Enhancing Remanufacturing Automation Using Deep Learning Approach

Chigozie Enyinna Nwankpa

A thesis submitted to the Department of Design Manufacturing and Engineering Management,

Faculty of Engineering,

University of Strathclyde Glasgow

In fulfilment for the award of Doctor of Philosophy

March 2022

Dedication

The work is dedicated to God almighty, my parents, Elder Godwin and Joy Nwankpa, my wife Loveth and children, Obianuri and Ebubechukwu, my siblings, and my extended family, without whom there would have been no success.

Acknowledgement

I want to appreciate God Almighty for His faithfulness to see me through to the end of this research. It has come with diverse challenges, but God faithfully delivered me. I want to appreciate my immediate and extended family, especially for providing the funding to support me through the research period, supporting my tuition, conference travels, and every other support required to complete the research. I remain indebted to you all.

Furthermore, I appreciate the support of my spouse Loveth and children, Obianuri and Ebubechukwu, for their understanding; despite the deprivations, they accepted to support me so far. To my Siblings, I appreciate your support and encouragement. On a sad note, my parents, Elder Godwin and Joy Nwankpa, who passed on to glory during this time. I feel inconsolable that you are not here to witness this joyous moment in our family. May your souls rest in perfect peace.

I appreciate your constructive criticisms and advice to my supervisors, Prof. Winifred Ijomah and Prof. Anthony Gachagan; without it, the work could last longer. I also acknowledge the following: Dr Carmelo Mineo, Dr Rahul Summan, Dr Gordon Dobie, Prof. Gareth Pierce, the Lab Technicians, especially Mr Fuad Warsame, who assisted in the data collection setup, and all the other University staff who helped in one way or another. I acknowledge your support. Also, to my colleagues who assisted me through their encouragement, advice and feedback, I could structure the research to a conclusion; I appreciate your support.

I appreciate your support and contributions of the research industry partners at Mackie Transmission Limited, led by Mr John Mackie, who provided the practitioner inputs in the research. I am grateful.

Finally, I appreciate the encouragement and prayers of everyone who supported the research through the good and dark moments, including Chigozie Ekwuonu, Kingsley Emeana, and Solomon Eze, among many others. God bless you all.

Publications

The author's publications during the research include three journal publications and six conference contributions, among other contributions that are not relevant to the thesis. These contributions are outlined as follows.

Journal Publications

- **Nwankpa, C.E***, Eze, S., Ijomah, W., Gachagan A, Marshall S "Achieving Remanufacturing Inspection Using Deep Learning" Journal of Remanufacturing vol 11, no. 2 pp. 89–105, 2020.
- **Nwankpa C.E***. "Advances in Optimisation Algorithms and Techniques for Deep Learning". Advances in Sc. Technology and Engineering Systems Journal, vol 5, no. 5 pp. 563–577, 2020.
- **Nwankpa, C.E*,** Ijomah, W. and Gachagan, A. "Design for Automated Inspection in Remanufacturing: A Discrete Event Simulation for Process Improvement," *Clean. Eng. Technol.*, vol. 4, 2021: 100199.

Conference Publications

- **Nwankpa, C.E***, Eze, S., Ijomah, W., Gachagan A, Marshall S. "Deep Learning-Based Vision Inspection System for Remanufacturing Application," 17th Int'l Conf. on Manuf. Res. 2019
- **Nwankpa, C.*,** Ijomah, W., Gachagan, A. and Marshall, S., "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning" Int'l Conf. on Comp Sci and Tech, 2020
- **Nwankpa, C.E*,** Eze, S., Ijomah, W., "Deep Learning-Based Automated Sorting System for Remanufacturing" 12th IEEE Green Technologies Conference 2020.
- Olisah M.C, **Nwankpa C***, Whitfield I, Ion W, "The exploration of collaborative supply chain factors in the Oil and Gas industry" British Academy of Management Conference, 2020.
- Olisah M.C, **Nwankpa C***, Whitfield I, Ion W, "The investigation of collaborative supply chain drivers in Oil and Gas industry" EurOMA Conference 2020.
- **Nwankpa, C.E*,** Ijomah, W. and Gachagan, A. "Artificial Intelligence for Process Control In Remanufacturing," Going Green Eco-design Conference 2021.

Author Declaration

I declare that except where specific references were cited in the thesis, the content is original and has not been submitted in part or whole for consideration for any other degree at this or any other University. This thesis is my original work and contains nothing that is the outcome of any research done in collaboration with others except collaborative publications from the study. Relevant seminars, workshops, and conferences were attended where the author presented some of the research outcomes, including visits to external institutions.

Signed: Chigozie Enyinna Nwankpa Date: 31st March 2022

"A wise man is strong; yes, a man of knowledge increases strength"

- King Solomon

Table of Contents

Dedication ii
Acknowledgement iii
Publications iv
Author Declaration v
List of Figures xiii
List of Tables xv
Abbreviations xvi
Abstract 1
CHAPTER ONE 2
INTRODUCTION AND BACKGROUND 2
1.0 Introduction 2
1.2 Research Background and Context 3
1.3 Deep Learning 4
1.4 Research Questions 5
1.5 Research Motivation and Justification 6
1.6 Scope of the Research 7
1.7 Research Design 8
1.8 Research Deliverables and Novelty 9
1.9 Research Beneficiaries 10
1.10 Contributions from the research 11
1.11 Thesis Structure 12
1.12 Chapter Summary 12
CHAPTER TWO 13
RESEARCH PHILOSOPHY AND DESIGN 13
2.0 Introduction 13
2.1 Research Design 13
2.1.1 Procedural Issues in the Research 15
a) Implementation Decision 15
b) Weighting Decision 16
c) Integration Decision 16
2.1.2 Research Delimitations 16
2.2 Research Philosophy 17
2.3 Rational for Adopting Mixed Method Research 18
2.4 Legitimacy of the research 19
2.4.1 Construct validity 19

2.4.2 Internal validity 20
2.4.3 External validity 20
2.4.4 Reliability 21
2.5 Vital Research Considerations 21
2.5.1 Researcher Involvement 21
2.5.2 Practitioner Needs 21
2.5.3 Domain of Research 23
2.5.4 Model Choice and Requirements. 24
2.5.5 Identification of Remanufacturing Processes for Modelling 24
2.6 Industrial Collaboration. 25
2.6.1 Benefits of the Collaboration 26
2.7 Data Collections 26
2.8 Research Data 27
2.8.1 Limitations of Existing Dataset 29
2.8.2 Remanufacturing Data for Deep Learning Research 29
2.8.3 Dataset Naming Convention 30
2.9 Chapter summary 31
CHAPTER THREE 32
LITERATURE REVIEW 32
3.0 Introduction 32
3.1 Overview of Remanufacturing 32
3.2 Benefits of Remanufacturing 33
3.3 Remanufacturing Automation 34
3.3.1 Challenges in Remanufacturing 35
3.3.1.1 Collections 35
3.3.1.2 Remanufacturing Process 36
3.3.1.3 Redistribution 37
3.4 Learning Models and Technologies as Solutions 37
3.5 General Learning Approaches 38
3.6 Learning Models in Remanufacturing 39
3.6.1 Operations Management 39
3.6.2 Forecasting 40
3.6.3 Factory Improvement 40
3.6.4 Decision Making and Support Systems 41
3.6.5 Remanufacturing Processes and Process Planning 42
3.6.6 Remanufacturing Technologies 43

3.7 Opportunities for Deep Learning 43
3.7.1 Operations Management 44
3.7.2 Forecasting 44
3.7.3 Factory Improvement 45
3.7.4 Decision Making and Support Systems 45
3.7.5 Remanufacturing Technologies 45
3.8 Suitability of Deep Learning Models 46
3.9 Deep Learning 48
3.9.1 Brief History of Deep Learning Research 51
3.9.2 Taxonomy of Deep Learning Methods 52
3.10 Deep Learning Architectures 53
3.10.1 Deep Unsupervised Learning Models 55
3.10.2 Deep Semi-Supervised Models 57
3.10.3 Deep Supervised Learning Models 58
3.11 Convolutional Neural Networks 60
3.11.1 Evolution of Neural Network Architectures 62
3.11.1.1 Spatial Exploitation 62
3.11.1.2 Depth and Width Exploitation 63
3.11.1.3 Multi-path Exploitation 65
3.11.1.4 Feature- Map Exploitation 65
3.11.1.5 Attention Exploitation 65
3.11.2 Components of the Convolutional Neural Networks 66
3.11.2.1 Convolutional Layer 66
3.11.2.2 Pooling Layers 67
3.11.2.3 Fully Connected Layers 69
2.11.2.4 Activation Functions 69
3.11.3 Loss Functions 76
3.11.3.1 Regression Loss Functions 77
3.11.3.2 Binary Classification Loss Functions 78
3.11.3.3 Multiclass Classification Loss Functions 79
3.11.4 Optimisation and Optimisation Functions 80
3.11.5 Regularisation Techniques 88
3.11.6 Evaluation Metrics 91
3.11.6.1 Accuracy 91
3.11.6.2 Error rate 92
3.11.6.3 Precision and Recall 92

3.11.6.4 $\boldsymbol{F\beta}$ Score.. 93
3.11.6.5 Receiver Operating Characteristics (ROC) 93
3.11.6.6 Area under the Curve (AUC) 94
3.11.6.7 Confusion Matrix 94
3.11.7 Applications of Deep Convolutional Neural Networks 95
3.12 Inspection Techniques in Remanufacturing.. 96
3.13 Sorting Systems in Remanufacturing... 100
3.14 Process Control Methods in Remanufacturing.. 102
3.15 Chapter Summary.. 105
CHAPTER FOUR 106
MODEL DESIGN AND APPLICATION TO INSPECTION IN REMANUFACTURING USING DEEP LEARNING 106
4.0 Introduction 106
4.1 Background to the Modelling and Development 106
4.1.1 Modelling Remanufacturing Processes.. 107
4.1.2 Model Development Boundaries.. 108
4.2 Research Model Design 108
4.3 Frameworks and Tools for Deploying Deep Architectures.. 110
4.4 Data Representation, Preparation and Pre-processing 111
4.4 1 Data Preparation.. 112
4.4.2 Splitting the Data.. 114
4.5 Learning Model and Development Considerations.. 115
4.5.1 The Computational Model Design 115
4.5.2 Model Parameters and Hyperparameters 116
4.5.3 Metric Selection 117
4.6 Computational Model Exploration.. 117
4.6.1 Transfer Learning.. 118
4.6.2 Novel Architecture 119
4.7 Learning Algorithms for Remanufacturing Application 120
4.7.1 Understanding the Architecture 121
4.7.2 Architectural Arrangement and Initialisation.. 123
4.7.3 Parameterising the mapping from Images to Label Scores.. 124
4.8 Modelling Surface Inspection in Remanufacturing Using Deep Learning 124
4.8.1 Inspection Applications and Data 125
4.9 Model Architecture, Parameters and Hyperparameters 126
4.10 Model Components, Hyperparameter Selection and Optimisation.. 127

4.11 Surface Fault Identification and Classification 128
4.11.1 Model Components and Hyperparameters 129
4.11.2 Model Selection for Transfer Learning 130
4.11.3 Model Training and Evaluations 130
4.11.4 Results and Discussions 138
4.12 Adapting the Developed Model to Torque Converter Component Inspection 139
4.12.1 Experiment and Model Training 140
4.12.2 Results and Discussions 140
4.13 Extending the Deep Learning Modelling to Achieve Automated Inspection 142
4.13.1 Benefits of Design for Automated Inspection 146
4.14 Chapter Summary 147
CHAPTER FIVE 148
MODELLING COMPONENT SORTING AND PROCESS CONTROL IN REMANUFACTURING USING DEEP LEARNING 148
5.0 Introduction 148
5.1 Modelling Sorting in Remanufacturing Using Deep Learning 148
5.2 Sorting Application and Data 149
5.3 Model Parameters / Hyperparameters and Modification 150
5.4 Experiment and Model Training 151
5.6 Results and Discussions 152
5.7 Modelling Process Control in Remanufacturing Using Deep Learning 155
5.7.1 Process Control Application and Data 155
5.8 Model Components, Hyperparameters and Modification 156
5.9 Experiment and Model Training 157
5.10 Results and Discussion 159
5.11 Chapter Summary 160
CHAPTER SIX 161
QUANTITATIVE ANALYSIS AND INDUSTRY FEEDBACK 161
6.0 Introduction 161
6.1 Analysis of the Learning Algorithm 162
6.2 Model Hyperparameters 162
6.2.1 Model Initialisation 164
6.2.2 Selection of Batch Size 166
6.2.3 Effect of Batch Normalisation on Model Performance 168
6.2.4 Selection of Activation Function 169
6.2.5 Selection of Learning Rate 171

6.2.6 Selection of Optimisation Techniques 172
6.3 Evaluating the Model Layers 174
6.4 Model Confirmatory Test 178
6.4.1 Transfer Learning 179
6.4.2 Training from Scratch Results 181
6.4.3 Developed Model Results 183
6.5 Cost Benefit Analysis 185
6.6 Basis for Testing Research Success 186
6.7 Model Research Validation 187
6.7.1 Experimental Validation 188
6.8 Industry Feedback 190
6.8.1 Model Validation Protocol 192
6.8.2 Industry Feedback Process 194
6.8.3 The Industry Validation Exercise 195
6.9 Results of Model Validation 195
6.10 Alterations to Enhance Clarity 196
6.11 Modelling Inspection, Sorting and Process Control in Remanufacturing 196
6.11.1 Descriptive Relevance 197
6.11.2 Goal Relevance 198
6.11.3 Operational Validity 198
6.11.4 Timeliness 199
6.11.5 Non-obviousness 199
6.12 Chapter Summary 200
CHAPTER SEVEN: CONCLUSION 201
7.0 Introduction 201
7.1 Achieving Research Objectives 201
7.2 Contributions to Knowledge and Research Originality 202
7.3 Recommendations and Future Research 203
7.4 Limitations of the Research 203
References 205
Appendix 1 Research code design tree 221
Appendix 2 Validation Results 223
Appendix 3A Developed inspection application model codes 225
Appendix 3B Developed sorting application model codes 226
Appendix 3C Developed process control model codes 227

List of Figures

Figure 1.1 Block diagram of a typical deep learning model 5
Figure 1.2 Cross-section of the core components of the torque converter units used as research data. 5
Figure 1.3 Block diagram of the evolved action research method in operations management adapted from [35]. 7
Figure 1.4 Research method visualisation 9
Figure 1.5 Research deliverables and contributions 10
Figure 2.1 Overall research structure 14
Figure 2.2 Study research design 15
Figure 2.3 Validated block diagram of the TC unit remanufacturing showing the focus processes as shades . 22
Figure 2.4 Adapted block diagram of the production and operations management system [59] 23
Figure 2.5 Schematic of the generic remanufacturing process. 24
Figure 3.1 Literature review ideation scope 32
Figure 3.2 Taxonomy of deep and machine learning models 48
Figure 3.3 Typical neural connections in a deep learning model 49
Figure 3.4 Typical neural network model 50
Figure 3.5 Taxonomy of deep learning architectures adapted from [113] 54
Figure 3.6 The auto-encoder 55
Figure 3.7 Components of a typical convolutional layer of a CNN showing the layer design approach 61
Figure 3.8 Sample convolution filters output used to learn different patterns in data [125] 61
Figure 3.9 Taxonomy of the CNN evolutions 62
Figure 3.10 The convolutional layer showing sparse interactions 66
Figure 3.11 The fully-connected layers showing the typical neural connections 69
Figure 3.12 Test response of the SiLU and dSiLU function [236] 71
Figure 3.13 Confusion matrix table 94
Figure 4.1 Model development stages 1078
Figure 4.2 Design conceptual model 1089
Figure 4.3 Actual model design 1090
Figure 4.4 Code development block diagram 109
Figure 4.5 Data splitting method 114
Figure 4.6 Model design approaches 118
Figure 4.7 Model modification stages showing the original model A, feature extraction model B, and the fine-tuned model C. 118
Figure 4.8 Flow diagram of the stages of achieving transfer learning 119
Figure 4.9 Architectural design block diagram 120
Figure 4.10 Sample of the inspection data 129
Figure 4.11 Model training pipeline 131
Figure 4.12 Pictorial representation showing model inputs, network, layers, loss function and optimiser 132
Figure 4.13 Pretrained VGGNet model performance on the surface inspection dataset 134
Figure 4.14 VGG model prediction visualisation using the confusion matrix 134
Figure 4.15 Training from the scratch loss and accuracy response on the VGGNet model 135
Figure 4.16 Model prediction visualisation for the training from scratch using the confusion matrix 136
Figure 4.17 Researcher developed model response on inspection data 137
Figure 4.18 Developed model prediction visualisation using the confusion matrix 137
Figure 4.19 A batch of samples for inspection II application used to train the model 139
Figure 4.20 Model training results on the Torque Converter surface inspection 141
Figure 4.21 Inspection II model predictions visualisation 141

Figure 4.22 Model inspection predictions result from the connected camera inputs 142
Figure 4.23 Automated inspection design approach. 143
Figure 4.24 High-level design approach of the design for automated inspection 144
Figure 5.1 Typical learning model block diagram 148
Figure 5.2 One batch of the original torque converter samples used to train the model 149
Figure 5.3 Sorting model training and validation responses 152
Figure 5.4 Sorting model confusion matrix 153
Figure 5.5 Sorting model predictions result from single-camera inputs 154
Figure 5.6 Model training results using the process control data 158
Figure 5.7 Cross-section of the model predictions from connected camera 160
Figure 6.1 Adapted Model analysis approach for comparing predictive classification tasks [329] 161
Figure 6.2 Effect of weight initialisation on performance: at the top - inspection I, middle - inspection II,

bottom - sorting. 165
Figure 6.3 Effect of batch size on performance: top - inspection I, middle - inspection II, bottom - sorting.. 167
Figure 6.4 Effect of batch-normalisation on performance: top - inspection I, middle - inspection II, bottom - sorting. 169
Figure 6.5 Effect of activations on performance: top - inspection I, middle - inspection II, bottom - sorting. 170
Figure 6.6 Effects of learning rates adapted from [256] 171
Figure 6.7 Effect of learning rate on performance: top - inspection I, middle - inspection II, bottom - sorting. 172
Figure 6.8 Effect of optimisers on performance: top - inspection I, middle - inspection II, bottom - sorting. 173
Figure 6.9 VGGNet Layer 1 block visualisation 175
Figure 6.10 VGGNet Layer 2 block visualisation 175
Figure 6.11 VGGNet Layer 3 block visualisation 175
Figure 6.12 VGGNet Layer 4 block visualisation 175
Figure 6.13 VGGNet Layer 5 block visualisation 176
Figure 6.14 Layer 1 of the researcher developed model 176
Figure 6.15 Layer 2 of the researcher developed model 177
Figure 6.16 Layer 3 of the researcher developed model 177
Figure 6.17 Layer 4 of the researcher developed model 177
Figure 6.18 VGG model prediction visualisation using the confusion matrix 180
Figure 6.19 Model prediction visualisation for the training from scratch using the confusion matrix 182
Figure 6.20 Developed model prediction visualisation using the confusion matrix 183
Figure 6.21 Comparison of the three models' final training accuracy 189
Figure 6.22 Comparison of the three models' final training losses 189
Figure 6.23 Model validation 190
Figure 6.24 The conceptual model design 190
Figure 6.25 Adapted Model validation Process [371] 192
Figure 6.26 Validation protocol 193
Figure 6.27 The model validation process 194
Figure 6.28 Identified camera blind spot for improvement 196
Figure 6.29 Formal model 196

List of Tables

Table 2.1 Tabular description of the recorded research data 30
Table 3.1 A comparison of different learning approaches adapted from [75]. 39
Table 4.1 Categorical encoding of model inputs............ 1134
Table 4.2 Model architecture for the inspection application............ 126
Table 4.3 Model I parameters and hyperparameters 129
Table 5.1 Model architecture optimised for the sorting application 150
Table 5.2 Model architecture optimised for the process control application 1567
Table 5.3 Model components and hyperparameters............ 15758
Table 6.1 Model parameter and hyperparameter definitions 1635
Table 6.2 Contingency table showing the Rater prediction probabilities 17879
Table 6.3 Evaluation of Kappa coefficient for the transfer learning model............ 1812
Table 6.4 Evaluation of Kappa coefficient for the VGGNet models training from scratch 1834
Table 6.5 Evaluation of Kappa coefficient for the developed model............ 1845
Table 6.6 Cost benefit problem selection matrix. 186
Table 6.7 Factor ranking for automation and no-automation............ 187
Table 6.8 Weighted factor ranking for automation and no-automation. 187
Table 6.9 Model data components and parameter outline 188
Table 6.10 Interpretation of the Likert scale feedback............ 199

Abbreviations

AI	Artificial intelligence
AM	Additive manufacturing
API	Application programming interface
ARM	Additive remanufacturing
AR	Augmented reality
AUC	Area under the ROC curve
AWS	Amazon Web Service
BGD	Batch gradient descent
BoL	Beginning-of-life
CC	Cloud computing
CNN	Convolutional neural networks
COBOTS	Collaborative robots
COCO	Common objects in context
CPS	Cyber-physical systems
CPU	Central processing unit
CUDA	Compute unified device architecture
DC	Decision tree
DiF	Diversity in Faces dataset
DL	Deep learning
DLT	Distributed ledger technology
DMEM	Design Manufacturing and Engineering Management
DAE	Deep Autoencoders
DBN	Deep belief networks
DDPG	Deep deterministic policy gradient
DNN	Deep neural networks
DQN	Deep Q networks
DRL	Deep reinforcement learning
DT	Digital twin
EHD	Edge histogram descriptor
ELU	Exponential linear units
EoL	End-of-life
FPGA	Field programmable gate arrays
GAN	Generative Adversarial Networks
GB	Gigabyte
GPU	Graphics processing units
GRU	Gated recurrent units
HOG	Histogram of oriented gradient
HORNN	Higher-order recurrent neural networks
ICT	Information and communication technology
IoT	Internet of things
KWh	Kilowatt-hour
LReLU	Leaky ReLU
LSTM	Long-short-term memory

MAE	Mean absolute error
MDP	Markov decision processes
ML	Machine learning
MLP	Multi-layer perceptron
MNIST	Modified National Institute of Standards and Technology
MoL	Middle-of-life
MSE	Mean square error
NAF	Normalising advantage functions
NDI	Non-destructive inspection
NN	Neural network
OEM	Original equipment manufacturer
PCA	Principal component analysis
PEID	Product embedded information device
PELU	Parametric ELU
PID	Proportional integral derivative
PIE	Pose, Illumination and Expression
PTAW	Plasma transferred arc welding
PReLU	Parametric ReLU
PLC	Product life cycle
POM	Productions and operation management
RAM	Random access memory
RBM	Restricted Boltzmann Machines
ReLU	Rectified Linear Units
RLReLU	Randomised leaky ReLU
RFID	Radio-frequency identification
RL	Reinforcement learning
RNN	Recurrent neural networks
RUL	Remaining useful life
ROC	Receiver operating characteristics.
SARSA	State-Action-Reward-State-Action
SELU	Scaled ELU
SIFT	Scale-invariant feature transform.
SGD	Stochastic gradient descent
SM	Subtractive manufacturing.
SURF	Speeded up robust features
SVM	Support vector machines
tf	TensorFlow
QR	Quick response
USB	Universal serial bus
VLSRC	Very-large-scale image recognition challenge
VOC	Visual Object Classes
XOR	Exclusive OR
WEEE	Waste Electrical and Electronic Equipment.

Abstract

In recent years, remanufacturing has significant interest from researchers and practitioners to improve efficiency through maximum value recovery of products at end-of-life (EoL). It is a process of returning used products, known as EoL products, to as-new condition with matching or higher warranty than the new products. However, these remanufacturing processes are complex and time-consuming to implement manually, causing reduced productivity and posing dangers to personnel. These challenges require automating the various remanufacturing process stages to achieve higher throughput, reduced lead time, cost and environmental impact while maximising economic gains. Besides, as highlighted by various research groups, there is currently a shortage of adequate remanufacturing-specific technologies to achieve full automation.

This research explores automating remanufacturing processes to improve competitiveness by analysing and developing deep learning-based models for automating different stages of the remanufacturing processes. Analysing deep learning algorithms represents a viable option to investigate and develop technologies with capabilities to overcome the outlined challenges. Deep learning involves using artificial neural networks to learn high-level abstractions in data. Deep learning (DL) models are inspired by human brains and have produced state-of-the-art results in pattern recognition, object detection and other applications. The research further investigates the empirical data of torque converter components recorded from a remanufacturing facility in Glasgow, UK, using the in-case and cross-case analysis to evaluate the remanufacturing inspection, sorting, and process control applications.

Nevertheless, the developed algorithm helped capture, pre-process, train, deploy and evaluate the performance of the respective processes. The experimental evaluation of the in-case and cross-case analysis using model prediction accuracy, misclassification rate, and model loss highlights that the developed models achieved a high prediction accuracy of above 99.9% across the sorting, inspection and process control applications. Furthermore, a low model loss between $3x10^{-3}$ and $1.3x10^{-5}$ was obtained alongside a misclassification rate that lies between 0.01% to 0.08% across the three applications investigated, thereby highlighting the capability of the developed deep learning algorithms to perform the sorting, process control and inspection in remanufacturing. The results demonstrate the viability of adopting deep learning-based algorithms in automating remanufacturing processes, achieving safer and more efficient remanufacturing.

Finally, this research is unique because it is the first to investigate using deep learning and qualitative torque-converter image data for modelling remanufacturing sorting, inspection and process control applications. It also delivers a custom computational model that has the potential to enhance remanufacturing automation when utilised. The findings and publications also benefit both academics and industrial practitioners. Furthermore, the model is easily adaptable to other remanufacturing applications with minor modifications to enhance process efficiency in today's workplaces.

CHAPTER ONE

INTRODUCTION AND BACKGROUND

1.0 Introduction

Environmental issues such as climate change have become a crucial discussion among experts in various fields to make more sustainable decisions about the current and future generations. These issues have spurred circular economy concepts that redefine how resources are managed, most importantly designing systems with the least waste and pollution, keeping products and materials in more prolonged use and many other approaches [1]. Despite these strategies, manufactured products still find their way to end-of-life (EoL), and the need to manage them at EoL becomes inevitable. To address these EoL products, reuse, remanufacturing, and recycling are among the dominant strategies to recover values from the EoL products, among other existing approaches [2]. The product recovery hierarchy has remanufacturing towards the top layers as it supports the reuse of products and components with the least additional raw materials [3].

Remanufacturing is the process of returning used products *"to at least original equipment manufacturer (OEM) performance specification from the customers perspective and giving the product a warranty that is at least equal to that of the newly manufactured equivalent"* [4]. It guarantees to return products with a warranty, matching that of a new product, proving more advantageous than other management approaches. More recently, remanufacturing is estimated to be worth about EURO 30 billion, and the industry continues to grow [3]. It is an end-of-life activity after a product has passed through its useful life. The product life cycle (PLC) consists of three vital stages, including the beginning-of-life (BoL), where the product is first designed and manufactured, and the middle-of-life (MoL), where the product is used, serviced and maintained. In the end-of-life (EoL) stage, used products are re-collected as cores, disassembled, remanufactured, recycled, reused and or disposed [5]. In these stages of the PLC, information flows across each process. However, the EoL stage suffers from significant information loss compared to the BoL and MoL stages, making most product decisions based on insufficient, inaccurate and incomplete product life cycle information with higher and increasing product complexity [6]. These make remanufacturing process activities more challenging to undertake.

Conversely, products for remanufacturing have vital characteristics, including: high recoverable value, stable product technology, good process technology, and must not suffer obsolescence [7], [8]. However, the design team's decision to remanufacture a product is an early thought during the product development stage. Remanufacturing makes it possible to recover some of the values invested in production at the end of its first life cycle. These decisions are based on the potential for value recovery against the cost of providing additional features to the product. The products are recycled or disposed if the cost cannot be recovered with some reasonable profit.

Nevertheless, researchers have outlined that the remanufacturing processes are complicated, manually performed and lack the suitable tools and methodologies to maximise profitability[9]–[11]. Moreover, using the existing approaches brings a time-delay bottleneck in remanufacturing, especially manual inspection, and the labour-intensive nature causes stress[12]–[14]. This research focuses on exploring the capabilities of deep learning algorithms to simplify these processes and enhance the tools and methods to achieve more sustainable remanufacturing. These technologies will provide vital benefits with their adoption, boost productivity, improve quality, increase capacity, and save operating costs [15].

1.2 Research Background and Context

The remanufacturing process is a complex process that returns used products to “as new” conditions in parts or complete with matching or greater warranty compared to the new products [16], [17]. It involves numerous processing stages, including identification, disassembly, inspection, cleaning, rework, reassembly and testing [18], [19]. Remanufacturing enhances the sustainable use of raw materials and provides environmentally safe productions with colossal energy-saving benefits [20], [21]. However, most of these processes are manually performed despite these potentials, making them slow, posing serious safety concerns to personnel, and unrepeatable results in most cases[6], [22]. These challenges result in reduced quality of remanufactured products, prolonged remanufacturing time, and increased costs, leading to significantly poorer value recovery from EoL streams. Besides, the availability of numerous simulation models that outline several novel technologies for improving remanufacturing process performances [23]–[25] and the validations that support productivity enhancements [26], [27] have not witnessed any success.

A solution to these concerns is developing process and hybrid automation technologies, identified as vital tools that can improve the remanufacturing process [6]. It explores technologies and systems to achieve safer, more repeatable, and improved results. This research examines the possibility of attaining hybrid technologies that could enhance remanufacturing processes, focusing on inspection, sorting, and process control. The problem statement highlights the need to explore the outlined challenges, including

- Remanufacturing processes are primarily manual and require experienced personnel to perform specific tasks efficiently [6], [22], drawing attention to new and emerging technologies for automating remanufacturing processes to reduce the dependence on expert judgement.
- The performance of production systems is always a critical concern, including remanufacturing; therefore, exploring approaches to make them repeatable is vital to enhance quality.
- Parameter tuning and optimisation are time-consuming, eliminating guesswork errors and saving time by automating feature detection algorithms using deep learning.

Besides considering these highlighted challenges, deep learning models offer a systematic process to achieve optimal performances by making the processes repeatable and continuous, improving service quality, and enhancing resource utilisation. This scope brings another big question that this research tries to answer, "How can emerging deep learning models automate the remanufacturing process or parts of the remanufacturing process?". This research explores possible solutions that could significantly automate remanufacturing or parts of the remanufacturing process since it is practically impossible to automate the entire remanufacturing process for all products without human interference.

1.3 Deep Learning

Deep learning refers to the acronym used to describe deep neural network models developed using artificial neural networks. Deep learning is a subfield of machine learning(ML) where computational models, composed of multiple processing layers, are used to learn representations in data with various abstraction layers [28]. It is an emerging technology that plays a significant role in enhancing industrial activities worldwide, and remanufacturing can leverage the benefits provided by the technology in other industries. Researchers outline that the deep learning field is driven by experimental findings rather than theory, with advances in algorithm design made possible through appropriate hardware and data [29]. However, modern enterprises are recently experiencing new revolutions from traditional manufacturing to intelligent manufacturing [30], with research on the impact of these technological advancements, especially for remanufacturing, attracting industry interest [31]. This research explores the potential of these deep learning algorithms to address identified remanufacturing challenges to meet the remanufacturing industry's specific needs. They have the potential to drive automation in remanufacturing; however, the general adoption of these technologies is still not vast across industries, including remanufacturing.

Nevertheless, artificial intelligence is a branch of computer science aiming to make computers perform up to human-style intelligence. AI systems use deductive logics whose rules depend on human ingenuity. Among artificial intelligence components, machine learning allows computers to learn data patterns without being explicitly programmed. Machine learning models also use statistical inference, where rules are inferred directly from data. It has significantly improved technological advances across diverse fields, with deep learning being the most significant driver of machine learning research lately, with massive state-of-the-art results [28].

Deep learning is inspired and modelled by the biological brain and thrives by learning high-level abstraction in data using multi-layered hierarchical architecture [28], [32]. A typical deep learning model schematic outline shows the stages to set up a standard deep learning-based system.

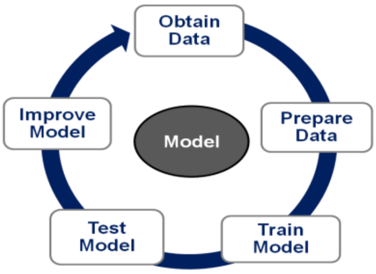


Figure 1.1 Block diagram of a typical deep learning model

These algorithms involve developing computational models, after which the obtained process data is modified to suit the model through a preparation process and used to train the model. Finally, the model is evaluated for performance and optimised through model improvement approaches. These stages are performed in all deep learning models to ensure that the model performs to desired standards.

Nevertheless, why should the potential of deep learning for remanufacturing be explored? To further highlight the need to study deep learning models, they do not require designers to develop and obtain hand-crafted features but automatically learn these features. It can also work on raw data and generalise well on different tasks, exceeding human-level recognition in predictive studies [28], [33]. The samples of the collected torque converter cores for remanufacturing used in this research are shown in Figure 1.2.

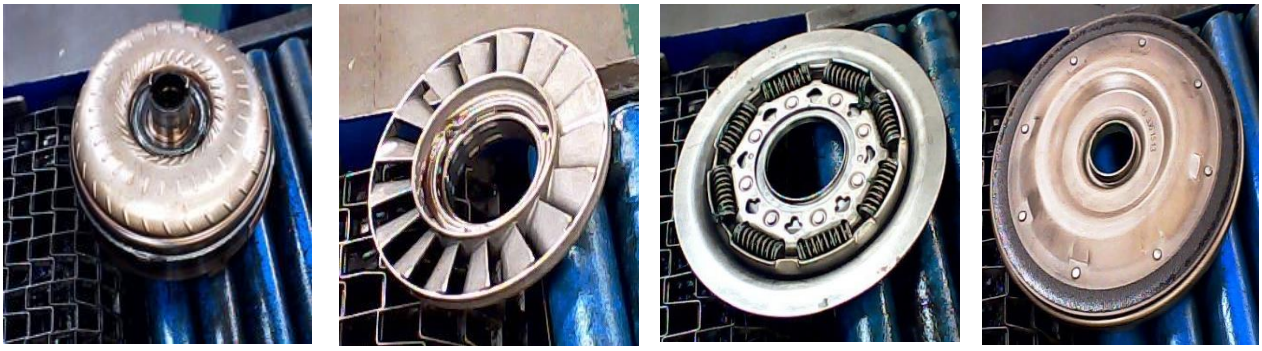


Figure 1.2 Cross-section of the core components of the torque converter units used as research data.

Besides, by exploring these DL concepts, novel solutions are produced to mitigate some of the associated hazards of remanufacturing processes, especially for automotive products, which are usually contaminated when returned.

1.4 Research Questions

This research builds on deep learning algorithms and models' ability to perform remarkably in diverse applications. Hence, this study explores how deep learning models can be deployed in different remanufacturing contexts to improve efficiency through automation. The following questions are set as guidelines to fulfil these aims. This research breaks the big question "How can deep learning enhance remanufacturing productivity?" into the following seven research questions to attain the aims and objectives of this study

Q1. What is the current level of deep learning applications in remanufacturing?

Q2. Can a novel method be developed using deep learning to automate various remanufacturing processes?

Q3. How can the understanding of the deep learning model results be improved?

Q4. How can the developed deep learning models be adaptable to other remanufacturing applications?

Q5. How can the study support future deep learning research in remanufacturing?

Nevertheless, these questions help to enhance the understanding of the existing deep learning literature and highlight the practical algorithms for use in the remanufacturing sector. It also outlines the techniques of applying and deploying the developed technologies in remanufacturing alongside discussions on the observed improvements achieved.

1.5 Research Motivation and Justification

This research is driven by the vital circular economy paradigms where sustainable habits and developments play significant roles in enhancing resource efficiency by reducing consumption of resources, materials, energy, and the corresponding environmental impacts while maintaining competitiveness in the global business environment. Moreover, remanufacturing refers to tools, technologies, systems, and knowledge-based methods to recover and reuse materials from end-of-life products [6]. As these technologies and techniques are not fully yet explored, it also outlines the need to delve further into the capabilities of emerging deep learning. Deep learning provides an abstract approach to learning patterns from data using computational models. The focus is exploring the deep learning algorithms in the remanufacturing context alongside the other enabling technologies.

Nevertheless, improving remanufacturing requires analysing the available process big data for understanding, using computational models capable of reading these extensive data and inferring helpful information from them. Machine learning, especially the deep learning subfield, has become the first choice and state-of-the-art for this application. The remarkable performance across almost every field of application draws further interest to explore remanufacturing applications. The following key factors inform the justification of the methods and techniques adopted in this research

- The opportunity to apply intelligent algorithms to remanufacturing and explore the challenges. However, integrating and achieving automated remanufacturing solutions has not been fully attained [22], [34].
- The available data limitations make them unsuitable for modelling remanufacturing processes as they were, thereby making it impossible to conduct empirical research on deep learning models for process-specific remanufacturing applications.
- To explore deep learning capabilities for remanufacturing by investigating the vast quantities of data produced by industrial processes to identify and understand the underlying trends.

- Exploring deep learning for remanufacturing productivity improvement by automating processes, increasing speed and accuracy since the algorithms have enhanced almost every application [28].
- To enhance the overall product quality by automation, eliminating total dependence on experts' judgement in the remanufacturing inspection, sorting and process control.
- To further investigate existing learning algorithms that have not been applied to specific remanufacturing tasks, primarily to perform inspection, sorting and process control.

1.6 Scope of the Research

The central theme of the study is to explore emerging deep learning technologies that could enhance the remanufacturing process. In addition, the general application of deep neural networks is investigated to provide empirical evidence that supports their application in remanufacturing. The research focuses on modelling inspection, sorting and process control in remanufacturing and considers the relationships between various model parameters on the performance of deep neural networks. Based on these findings, this research drives a comprehensive automated model for deploying deep neural network models for remanufacturing inspection, sorting and process control.

This scope is informed by the production and operations management action research theory-building approach that focuses on developing and applying various concepts to build knowledge [35]. Nevertheless, a holistic approach to action research includes the components of action research. The most significant considerations are finding specific areas of focus, obtaining industry collaborations to enhance transparency, data access, analysis, evaluation, and disseminating the findings from the research. This scope is highlighted in the dotted outline of Figure 1.3, the block diagram of action research in operations management.

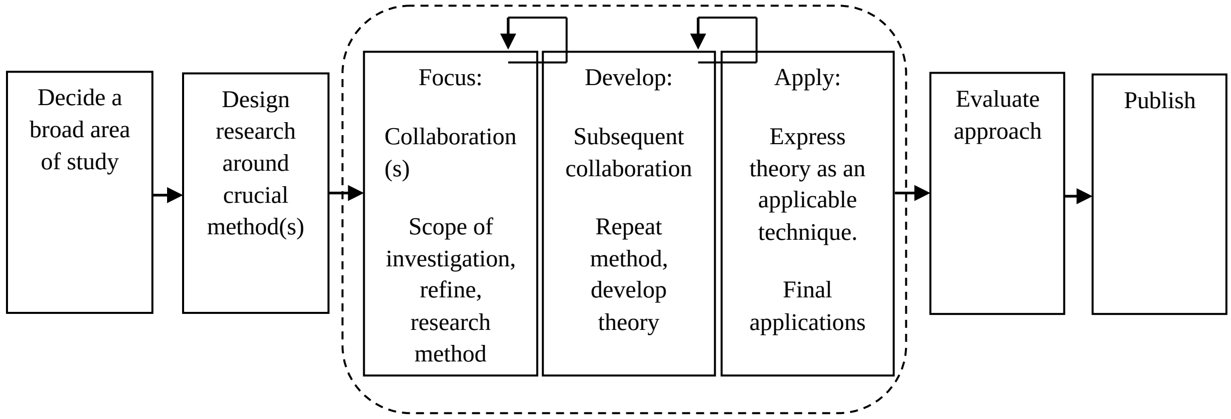

Decide a broad area of study
Design research around crucial method(s)
Focus:
Collaboration (s)
Scope of investigation, refine, research method
Develop:
Subsequent collaboration
Repeat method, develop theory
Apply:
Express theory as an applicable technique.
Final applications
Evaluate approach
Publish

Figure 1.3 Block diagram of the evolved action research method in operations management adapted from [35].

The role of collaboration highlights the process of two unrelated entities working together to determine how mutual goals, risks, information and resources are shared to achieve a common goal [36]. Understanding the collaboration components is vital to ensure that parties involved discern their roles throughout the process, transparently building trust and sharing information [37]. The primary benefits of this collaboration include providing the researcher with first-hand experience of the remanufacturing process, which is significant for understanding the processes, assessing the remanufacturing challenges from the practitioners' perspective, sharing knowledge to improve processes and providing access to the process data for model development and evaluation.

The rationale behind this scope was that only image data was collected, which would aid the evaluation of the different test cases during the research. These images were used to recognise parts from disassembly, inspection, and sorting. However, the disassembly requires additional specialised hardware, while the sorting and inspection applications require an actuator and visual sensors. The hardware requirements restrict the choice of sorting, inspection and process control for easy evaluation and validation.

1.7 Research Design

This research adopts the applied research method that focuses on obtaining empirical observation to solve critical societal problems. The use of applied research is the dissemination of the findings for ease of implementation, especially for practitioners, with authors suggesting the vital strength of the approach is the ability to test the results obtained from research [38]. Nevertheless, researchers added that applied research could benefit from building theories and testing the developed hypotheses [39], thereby expanding the scope of the basic knowledge to obtain additional values, usually for practitioners.

Moreover, the research design uses the sequential mixed research strategy, where the quantitative and qualitative research techniques alongside the strength of action research, to develop, understand and highlight the benefits of the emerging deep learning technologies in remanufacturing inspection, sorting, and process control. The action research perspective details the collaborations, scope, developments, and application of theories to build new knowledge alongside disseminating the findings

This research uses the in-case and cross-case analysis of various remanufacturing processes as an enquiry technique for developing new knowledge for applying deep learning models in remanufacturing. It follows an engineering research design approach that outlines that specific cases in engineering differ from the original case-study method with the contemporary component where historical data from any process cannot meet the definition of case-study research [39]. The in-case analysis focuses on the application familiarisation and thorough documentation of the process, while the cross-case highlights the differences and similarities in the models and results.

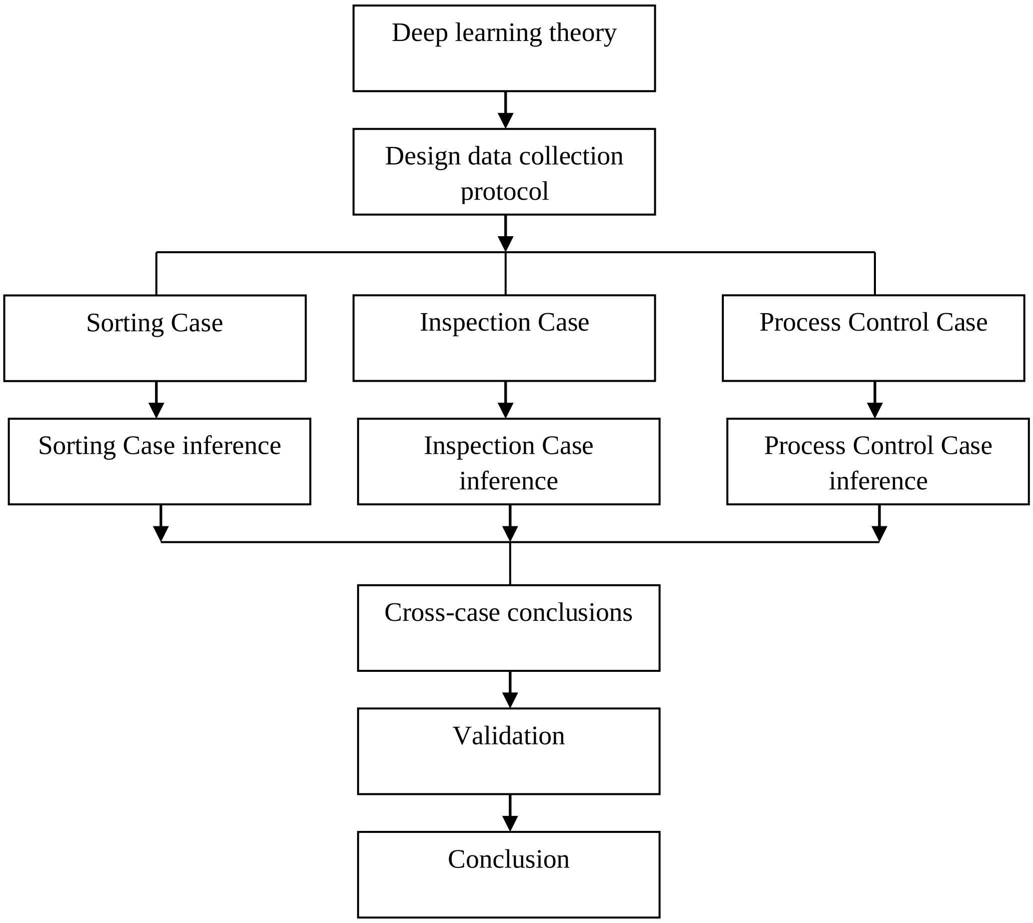
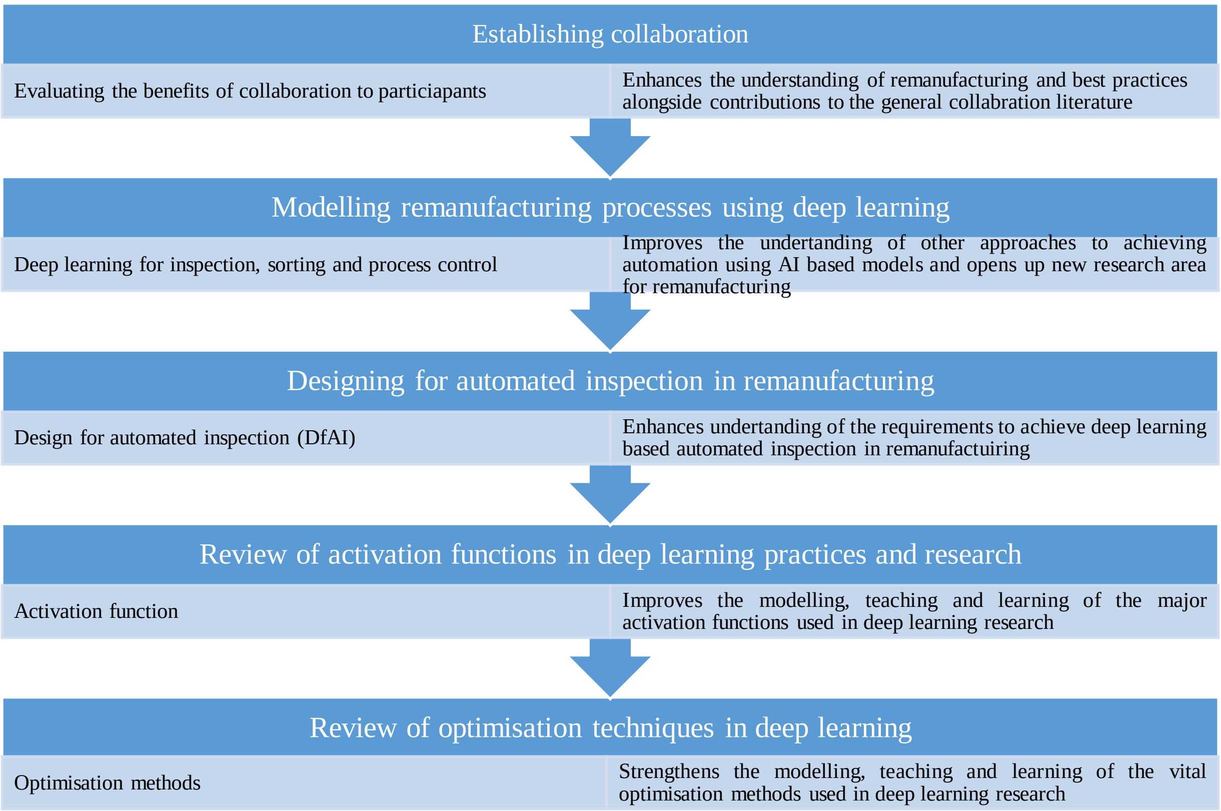


Figure 1.4 Research method visualisation

Moreover, the dependent sequential mixed method research uses deep learning algorithms for modelling inspection, sorting, and process control for remanufacturing applications. Afterwards, a validation by review assesses the usefulness of the developed automated systems for remanufacturing from a practitioner's perspective.

1.8 Research Deliverables and Novelty

The uniqueness of the research focuses on the method of achieving process automation using deep convolutional neural network models. Besides, the deliverables of the study include literature reviews on activation functions and optimisation techniques for deep learning alongside a CNN architecture for use in remanufacturing. Furthermore, it highlights various methods to model and deploy deep neural network models in remanufacturing inspection, sorting and process control applications. It also delivers convolutional neural network architecture that can quickly adapt to other remanufacturing processes. Nonetheless, another vital delivery of the research comes from industry collaboration. The study proposes improved process automation methods to enhance industry practices using deep learning and further outlines a process approach to achieve automated inspection, thereby providing tools that help decision-making before remanufacturing. Finally, the principal research deliverables and contributions to the body of knowledge are outlined in Figure 1.5.



Establishing collaboration

Evaluating the benefits of collaboration to particiapants

Enhances the understanding of remanufacturing and best practices alongside contributions to the general collabration literature

Modelling remanufacturing processes using deep learning

Deep learning for inspection, sorting and process control

Improves the undertanding of other approaches to achieving automation using AI based models and opens up new research area for remanufacturing

Designing for automated inspection in remanufacturing

Design for automated inspection (DfAI)

Enhances undertanding of the requirements to achieve deep learning based automated inspection in remanufactuiring

Review of activation functions in deep learning practices and research

Activation function

Improves the modelling, teaching and learning of the major activation functions used in deep learning research

Review of optimisation techniques in deep learning

Optimisation methods

Strengthens the modelling, teaching and learning of the vital optimisation methods used in deep learning research

Figure 1.5 Research deliverables and contributions

1.9 Research Beneficiaries

This research's beneficiaries are four-fold: academia, remanufacturing practitioners, machine learning practitioners, and supply chain practitioners. These include

- Academia benefits from current remanufacturing practices that have depended chiefly on simulations to highlight the viability of adopting emerging technologies; however, implementing these technologies has not been fully explored for practicality. The research helps to outline how to achieve practical deployment of deep learning models in remanufacturing, thereby enabling researchers to understand how to practically use them in remanufacturing applications, supplementing remanufacturing processes knowledge and improvement techniques.
- The research provides remanufacturing-specific tools for inspection, sorting, and process control for the remanufacturing practitioners, closing the gap in the scarcity of adequate technologies and tools for improvement.
- It also benefits the machine learning community by providing valuable insights on improving model designs and understanding in choosing model parameters, including the choice of activation functions and optimisation techniques for deep learning, summarised with published literature reviews from this research, among other unpublished findings.
- Supply chain practitioners also benefit from the contributions of this research by understanding the factors of supply chain collaboration, including trust, information sharing and other vital benefits

of cooperation. The benefits of this collaboration were helpful in understanding the essential elements that support collaboration, especially with the industry stakeholders

1.10 Contributions from the research

Research contributions often referred to as research gaps in the literature, are *"an area for which missing or insufficient information limits ability to reach a conclusion for a question"* [40]. These gaps constitute some research needs that limit the ability to make decisions. For remanufacturing, previous research focussed on understanding the processes involved in product remanufacture alongside the constraints and challenges faced by remanufacturers. Most importantly, In the field of inspection and sorting, researchers have shown the possibility of deploying emerging technologies, especially machine learning and deep learning, to improve some of the processes involved in remanufacturing products [41]. However, most of the existing machine learning research in the remanufacturing field is modelled as regression-based problems, where system performance predictions use numeric data, and the modelled outputs are also numeric [41], [42]. Conversely, the research models the remanufacturing inspection, process control, and sorting processes as a classification problem that use image data to achieve component and product inspection, sorting and process control.

The research provides numerous academic contributions to the field of remanufacturing and machine learning, which include:

- A published review on the optimisation techniques for deep learning.
- A published review on the activation function trends for deep learning
- Summary of the research progress in sorting, process control and inspection technologies used in remanufacturing to enhance understanding.
- Outlines novel approaches to modelling inspection, sorting and process control using deep learning methods to achieve automated inspection and process control in remanufacturing.
- A published framework for automating inspection in remanufacturing using the design for automated inspection.
- Developing a Python-based deep convolutional neural network model that performs sorting, inspection and process control in remanufacturing. Furthermore, these models enhance the automation of the remanufacturing sorting, inspection, and process control applications.
- A dataset of torque converter components for supporting further research on deep learning in remanufacturing.

Besides, the industrial deliverables include simplified methods of achieving component inspection, sorting and process control using computational models that can improve various remanufacturing processes and techniques, thereby enhancing efficiency and productivity.

Nevertheless, a summary of the research publications was included as publications on page iv.

1.11 Thesis Structure

The remaining parts of this thesis are structured in chapters, which include the following;

Chapter 2 presents a brief introduction to remanufacturing, automation, challenges, benefits of remanufacturing, and a brief introduction to deep learning algorithms. It further outlines the architectures of deep neural networks, focusing on the deep convolutional neural networks, their evolution and components, among others. Finally, the current inspection, sorting and process control methods in remanufacturing were discussed alongside the limitations.

Chapter 3 presents the research design and philosophical approaches used in the study. It also details the procedural issues encountered during the investigations—the rationale for adopting the sequential mixed-method research approach alongside the legitimacy of the research. Finally, the data collection methods and the vital research considerations were evaluated to attain the research objectives.

Chapter 4 presents the modelling and design techniques for using the deep convolutional neural network algorithms for modelling various remanufacturing processes. The design, data preparation, training and evaluation metrics used in modelling the remanufacturing applications are presented. It also explains the modelling approach of deep convolutional neural networks for the modelling inspection process in remanufacturing.

Chapter 5 presents the process adaptation approach for remanufacturing sorting and process control using the researcher-developed deep convolutional neural networks model. It details the modelling method and architectural modification to perform sorting and process control in remanufacturing alongside the in-case analysis and deductions from the respective models.

Chapter 6 presents the quantitative analysis of the developed model, where the cross-case analysis of the different model parameters was evaluated to obtain the optimal performance parameters used in the final model and the model industry feedback. Finally, the experimental validation and industry feedback interpretations supporting the research findings were discussed.

Chapter 7 provides the conclusions drawn from the investigation. In addition, it highlights the significant contributions of the research, recommendations and other areas of future research not explored by the research.

1.12 Chapter Summary

This chapter introduces remanufacturing, the benefits and challenges, and provides an overview of emerging deep learning technologies for improving remanufacturing. It further highlights the research background, context, aims and objectives, motivations, scope and beneficiaries. It also briefly discussed the research method and structure of the overall thesis.

CHAPTER TWO

RESEARCH PHILOSOPHY AND DESIGN

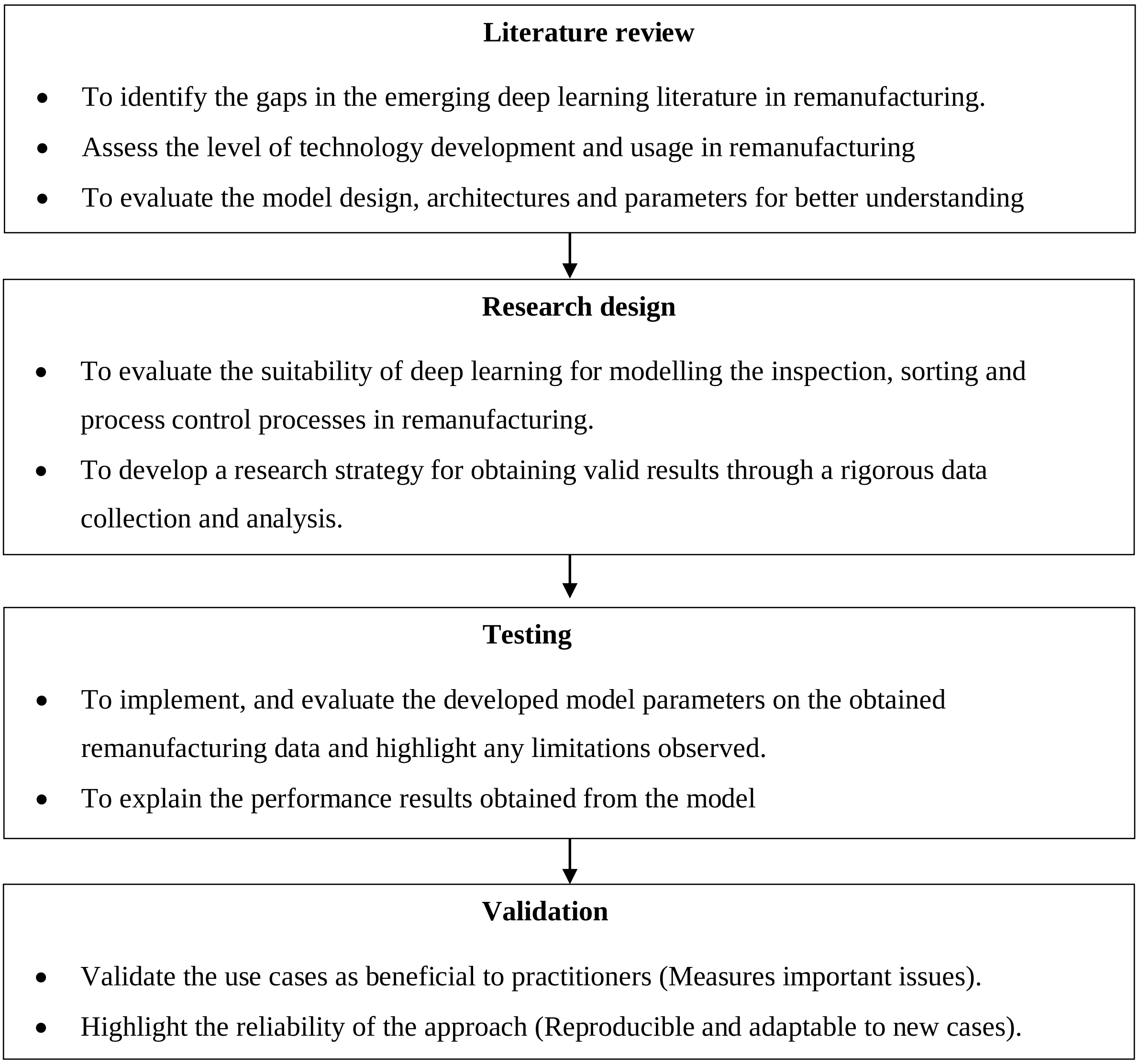
2.0 Introduction

This chapter discusses the research methodology and design approach. It outlines the rationale for adopting the applied research, focusing on empirical analysis to understand the best practices of deploying deep learning models in remanufacturing. The chapter addresses the research questions (Q3) on the specific remanufacturing processes that could be modelled using deep learning. It highlights how the research could support future remanufacturing research (Q5) by describing the created datasets to support future deep learning research.

2.1 Research Design

The research design refers to the science of performing specific research. The research design details the plans, procedures, assumptions, data collection methods, and analysis used in a given research. These plans include all the decisions made in the order that makes sense alongside the presentation order. On the other hand, science has been defined as a systematic and methodological approach to obtaining new knowledge [43]. Scientific research details the methods and principles that enable researchers to draw a valid and reliable conclusion from a study. The main benefit of the scientific approach to research is that it provides a structured approach for gathering, evaluating and reporting results in the research context. This technique allows researchers to design and present their research in the most logical approach.

Moreover, the research design also helps judge the quality of research according to specific logical steps: data dependability, credibility, trustworthiness, and conformability [44]. The approach to achieving the research objectives is summarised in the four stages of activities summarised in Figure 2.11.



Literature review

- To identify the gaps in the emerging deep learning literature in remanufacturing.
- Assess the level of technology development and usage in remanufacturing
- To evaluate the model design, architectures and parameters for better understanding

Research design

- To evaluate the suitability of deep learning for modelling the inspection, sorting and process control processes in remanufacturing.
- To develop a research strategy for obtaining valid results through a rigorous data collection and analysis.

Testing

- To implement, and evaluate the developed model parameters on the obtained remanufacturing data and highlight any limitations observed.
- To explain the performance results obtained from the model

Validation

- Validate the use cases as beneficial to practitioners (Measures important issues).
- Highlight the reliability of the approach (Reproducible and adaptable to new cases).

Figure 2.1 Overall research structure

Conversely, the mixed-method approach combines quantitative and qualitative research techniques, whose strength is higher than independently using qualitative or quantitative methods. Moreover, it involves the use of data from both views in tandem. The study's research design is outlined in the block diagram in Figure . The various process data for collection include images of objects recorded as videos and converted to images for further analysis and quantitative evaluation.

`

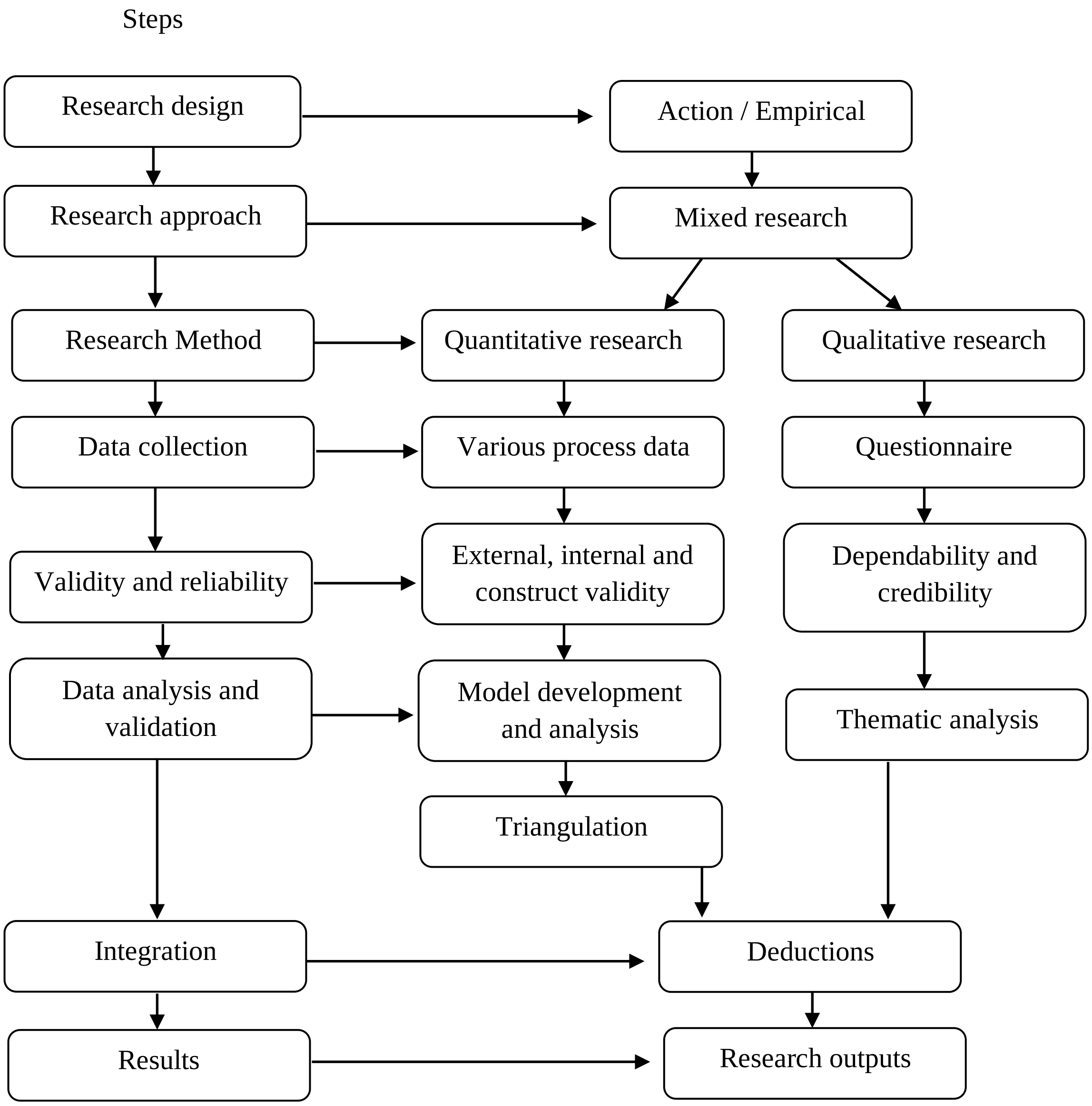


Figure 2.2 Study research design

2.1.1 Procedural Issues in the Research

This study adopts the mixed research method to explore both quantitative and qualitative approaches; however, there are several issues of procedure to manage. These include the issue of implementation, where a decision of which method is explored first, the weighting issue that decides the dominant method and finally, the integration issues, where the findings of the respective methods are combined [45], [46]. These procedural issues in the research are outlined as follows.

a) Implementation Decision

There are two approaches to implementation, including concurrent implementation and sequential implementation. The concurrent implementation involves performing both qualitative and quantitative analysis simultaneously. At the same time, the sequential allows one method to follow the other, either quantitative before qualitative or qualitative followed by the quantitative method [45]. This research adopts the sequential design as the research objective is to quantitatively develop and adapt the deep learning techniques across various remanufacturing processes and evaluate the performance. The qualitative study expands the scope through practitioner inputs by capturing their viewpoints and using them to enhance the understanding of the results. The quantitative stage models the remanufacturing inspection, sorting and process control alongside analysing the model's performance. At the same time, the qualitative study complements the results and their applicability to remanufacturing. These results are integrated to support the conclusions derived from the research.

b) Weighting Decision

Weighting refers to the magnitude of importance of the qualitative and quantitative methods to answer the research questions by assigning equal or unequal weight to the respective methods [45]. The research prioritised the quantitative evaluation method as it effectively addresses the vital research objectives with the qualitative evaluation used to enhance the credibility of the findings. The results from both evaluations were integrated to further the conclusions drawn from the study.

c) Integration Decision

The point of interface of the research is often referred to as the point of integration when the study's quantitative and qualitative components are combined. It represents one of the most crucial decisions in the research design, with researchers suggesting that the most common integration point is the results and analysis [47]. Integration refers to the stage in research where the findings of the qualitative and quantitative investigations merge [45], [46]. Without explicitly linking the findings of the two research methods, the research output becomes a collection of multiple research methods rather than mixed-method research. However, it has both components of a mixed research method. Hence, it is essential to link the methods at different stages of the research. The research integrates qualitative and quantitative data analysis and results. These integrations are described as follows.

- Data analysis - The modelling and quantitative evaluation findings provide a meaningful interpretation of the data. At the same time, the qualitative practitioner feedback helped validate the findings as worthy contributions to the body of knowledge.
- Results - The quantitative and qualitative methods were integrated to answer the research questions used in the study. In addition, the findings of the respective assessments were connected to enhance the understanding of the research in general.

2.1.2 Research Delimitations

This research focused on the exploration, design, development, analysis and testing of deep learning methods in remanufacturing and will only concentrate on deep learning algorithms. The deep convolutional neural network algorithms have shown state-of-the-art performance across domains. Therefore, the research focuses on convolutional neural networks for modelling inspection, process control and sorting in remanufacturing. The other statistical machine learning models are not considered with the deep learning algorithms discussed in Sections 3.9 and 3.10 of the literature review in chapter three. Moreover, these other models were not considered in the later stages of the research.

2.2 Research Philosophy

The research philosophy consists of four basic philosophical worldviews: positivism, constructivism, advocacy/participatory, and pragmatic worldviews [45]. These worldviews directly shape the general research approach. As presented by the researchers, an overview of these four viewpoints is as follows. The positivist worldview represents traditional research, with the assumptions true for quantitative analysis and not for qualitative research. Hence, it is often referred to as empirical science or postpositivist research.

Furthermore, the pragmatic approach also considers the reality of achieving the research goals, not just theoretical perspectives. It focuses on the situations, actions, and consequences rather than past conditions, emphasising the research problems to understand them further. On the other hand, the authors outline that the advocacy worldviews have political agendas intertwined with the research. In contrast, the social constructivist worldview refers to the social construction of reality where the research depends mainly on the participant's views of the situation under investigation. The advocacy and social constructivist worldviews have no relational relationship with this study. In contrast, the positivist and pragmatic worldviews form the basis of the research to evaluate the possibility of modelling various remanufacturing processes using deep learning.

The research follows a holistic deductive paradigm that emphasises five vital themes across quantitative and qualitative research methods: empirical enquiry, pragmatic enquiry, deductive analysis, and quantitative and qualitative research. The reality of these models is a pragmatic enquiry, while the generalised conclusions drawn from the results are deductive. The empirical enquiry refers to the modelling and analysis of the respective applications developed and used in the research, including the sorting, inspection and process control application data alongside the validation feedback.

Moreover, the quantitative research approach concerns the modelling and explanations of concepts and controls. It starts the investigation from the existing theories. In contrast, qualitative research concerns the interpretations of the feedback obtained after the model validation, with the researcher being distant from the subjects. However, the qualitative enquiry proponents believe that knowledge is constructed, which counters the quantitative proponents that knowledge is discovered by refining the existing theories [48]. The adopted quantitative approach involves the experimental enquiry into the use of deep neural networks in developing models for remanufacturing inspection, sorting, and process control applications, while the qualitative evaluation considers the output of the models alongside the practitioner feedback used for the validation of the research findings.

2.3 Rational for Adopting Mixed Method Research

Research methodology refers to assumptions, data collection, analysis, interpretation, and methods used to present research, making the findings open to critique and replication [49]. The research adopts the sequential mixed-method research approach, a hybrid technique where the researcher expands on one research method's findings using another. It starts with a quantitative analysis that explores concepts and theories, followed by a qualitative evaluation involving a few individuals or stakeholders and vice versa [44], [45]. However, researchers have outlined that mixed-method research can enhance understanding of more complicated research questions, with more substantial evidence from the broader data scope though it is more challenging to execute [44]. In addition, the mixed method offers the benefit of using the qualitative data to explore the quantitative findings further, augmenting the research findings and involving the research's community-based stakeholders [50]. Finally, the mixed method adopted in the current study gives the flexibility to develop deep learning models for remanufacturing applications and test the concepts in various cases to validate, refine and consolidate the results.

The quantitative component of the research evaluates the process data to explain the experimental observations between test variables using theories [45]. At the same time, the qualitative approach brings the practitioner's views on the obtained outcomes. The typical empirical research is prominent by the investigator's activities, which set the study's conditions alongside developing, building, and controlling the investigation's experimental conditions [51].

The empirical research method is considered as it offers the advantage of using direct observation and measurement of the considered process to make deductions about the process [49]. The research explores the case approach with five essential components, including an empirical inquiry, with real-life and contemporary components, using multiple sources of evidence alongside having no defined boundary between the context and phenomenon [39], [44]. It evaluates the "how" and "what" questions that the research poses [44], alongside combining the strengths of an applied and experimental study to the usefulness and application of knowledge [49]. However, the role of the researcher has become a crucial factor that highlights the difference between case research and other research methods, which considers the researcher's control over events. Multi-case research requires that the researcher works with the participants, similar to being involved in action research [39].

The research method is an experimental mixed-method research process that relies on the quantitative postpositivist research view while recognising that the qualitative approaches will benefit the research. The purpose of adopting the sequential dependent mixed research approach is to demystify the complexity of understanding the application of deep learning models in remanufacturing. The outcomes of the quantitative investigation are discussed further by academics and industrial experts to outline if the research objectives and outcomes were met or not.

Nonetheless, the research adopts a case-based approach used by other researchers, allowing for two stages of data analysis, including in-case analysis and cross-case analysis [268], alongside permitting data triangulation. The case approach is grounded in the lack of theoretical work in the deep learning field. Because of this lack of theory, most ground-breaking research is primarily based on heuristics arguments derived from specific case investigations, and the same method is adopted in the research. Moreover, another important reason for adopting case-based mixed-method research is the type of collected data and the best ways of making sense of it. The collected data were qualitative image data, while the numerical and computational analysis of the results is quantitative. Also, the validation protocol used to ascertain the research findings is qualitative, obtained as feedback from the questionnaires. Hence, the research objectives include investigating deep learning algorithms to enhance remanufacturing efficiency and productivity.

The multi-case investigation approach was adopted as the most appropriate method because research suggests that they provide greater generalisation and have a higher capability for creating and developing theories than single cases [52], [53]. In addition, it helps to model and evaluate systems on different real-world applications to determine their performance and generalisability. Perhaps, researchers have also outlined that implementing the simulation-based model results in remanufacturing has not witnessed appreciable deployment due to the practical implementation constraints, which are still lacking [6]. Therefore, this research outlines the practical steps and methods of deploying deep learning-based models in remanufacturing applications.

Nevertheless, research suggests two approaches to bridge the gap between theories and measurement: the top-down strategy, the theory-driven approach, and the bottom-up strategy, the data-driven approach [54]. The theories-driven approach starts with "the constructs and works towards the observable variables", while the data-driven method starts with the "observations and works towards the theoretical constructs". Furthermore, this research adopted the bottom-up approach where the process data are collected first and used to model the deep learning-based systems.

2.4 Legitimacy of the research

The legitimacy of research outlines the criteria used in research design to justify the research approaches. According to researchers, the research design outlines the logical set of statements, which helps judge the quality of a research design [44]. The author further highlights four critical design criteria for judging the quality of research design: construct validity, internal validity, external validity, and reliability. These criteria are described in the following subsections

2.4.1 Construct validity

Construct validity helps establish the chain of evidence using multiple sources of evidence. Construct validity issues arise when the researcher measures variables based on inadequate definitions [55]. It also measures the conformability of the correct operational measures in the research. This research construct adopts a data triangulation approach where various remanufacturing application cases of inspection, process control, and sorting were used to test the quality of research findings, strengthening the overall research's validity. The process-specific data were collected and used to train the model and, afterwards, compare the performance.

This research has limited construct validity threats. There was an explicit definition of the variables used in the literature review in chapter 2. The respective applications have all the factors well defined and presented in detail with their effect on the learning algorithm outlined.

2.4.2 Internal validity

Internal validity refers to the researcher's degree of confidence about the inferences and conclusions of the research by establishing causal relationships between variables [46]. In addition, internal validity outlines certain conditions that lead to other processes or systems behaviours. Therefore, researchers suggest that internal validity should be given special attention in experimental research. The vital approaches to internal validity include using logical models to build explanations, pattern matching, and addressing rival explanations [44]. The logical model approach was adopted to detail the relationships between different design stages, from the conceptual model to the final actual model, alongside explaining the different stages.

Nevertheless, internal validity is achieved through a complex model analysis, using various in-case analyses that enable control over variables. Nevertheless, the variables used for the research evaluation are based on theoretical foundations and findings from empirical research on applying deep learning models. The various control variables that impact the research's dependent variables, including prediction accuracy and error rate, were introduced to the model following other empirical research with interpretations and deductions presented to enhance internal validity. The control variables used to evaluate the impact of various thematic factors on model performance include:

- The model activation functions
- The optimisation methods
- The batch sizes
- the model parameter initialisation techniques
- The batch normalisation
- The loss functions

The developed model was tested against the above factors for performance evaluations and analysis with corresponding inferences and deductions outlined.

2.4.3 External validity

External validity refers to the ability of the research to be generalised across a population that is applying research findings in other instances of the phenomenon[44], [45]. It is another primary criterion for judging the quality of the research design, as it outlines the domain where the study findings can be generalised. The study adopted the mixed research method that combines the qualitative and quantitative research methods alongside data triangulation to obtain a robust method of improving external validity.

Moreover, the quantitative evaluation has a lesser external validity threat since there are limited data for the investigated processes. However, the external validity of the research was achieved by adapting the original model developed for the remanufacturing inspection process to other applications, including sorting and process control applications in remanufacturing. Besides, the research examined the relationship between several model parameters on the performance alongside the qualitative interpretation of the findings. By doing these, the consistent performance results across the various applications improve the overall validity of the research.

2.4.4 Reliability

Research reliability refers to the ability to obtain similar outcomes on research by repeating the study, thereby demonstrating that the study's findings can be repeated with the same results. The reliability of the research is achieved by adopting researchers' suggestions to either develop a case investigation protocol or a case database [44]. These techniques ensure that the model is made repeatable through the many documented operational steps taken during the research. The repeatability of the results is achieved through the set of results obtained from the individual cases of the model application that achieved significantly high prediction accuracy across the different remanufacturing inspection, sorting, and process control applications.

2.5 Vital Research Considerations

The researcher considered several factors to decide the scope of the investigation. The study adopts the cross-case analysis method for the reasons outlined in Section 1.7. Therefore, selecting a research method that supports the case analysis is required. The research method must be appropriate for use in the research domain and can obtain results that satisfy the needs of the remanufacturing practitioners. Besides, the vital considerations for the investigation include the researcher's involvement, the research domain, the practitioner's needs, the model choice and requirements and the choice of application cases.

2.5.1 Researcher Involvement

The researcher was not employed by the industry partner involved during this research. Perhaps, the research epistemology of the qualitative paradigm requires the principal investigator to interact with those being researched. The researcher interacted with the practitioners, which improved the researcher's understanding of the overall remanufacturing processes.

2.5.2 Practitioner Needs

The affiliate company is a vital part of the POM research as collaborators are critical to the success of POM research since they provide crucial practitioner feedback. The activities of this company include gearbox and torque converter (TC) remanufacture, diagnostics, overhaul and other automotive repair activities. The partners provide the researcher access to the industry process data for investigation. The research was also conducted primarily in the Mackie Transmission remanufacturing facility in Glasgow, UK, to further understand the challenges faced by practitioners. The routine activities helped identify other critical areas of need for the practitioners.

The TC remanufacturing process is the primary focus of the investigation because the remanufacturing process data can be acquired with the available setup. Perhaps, when a customer returns a faulty TC, the company examines the unit according to the process activities outlined in Figure 2.3. However, the pre-remanufacturing activities include the job book-in, examination, initial quotation, and job confirmation. These activities are not discussed as they are out of the shop floor remanufacturing process.

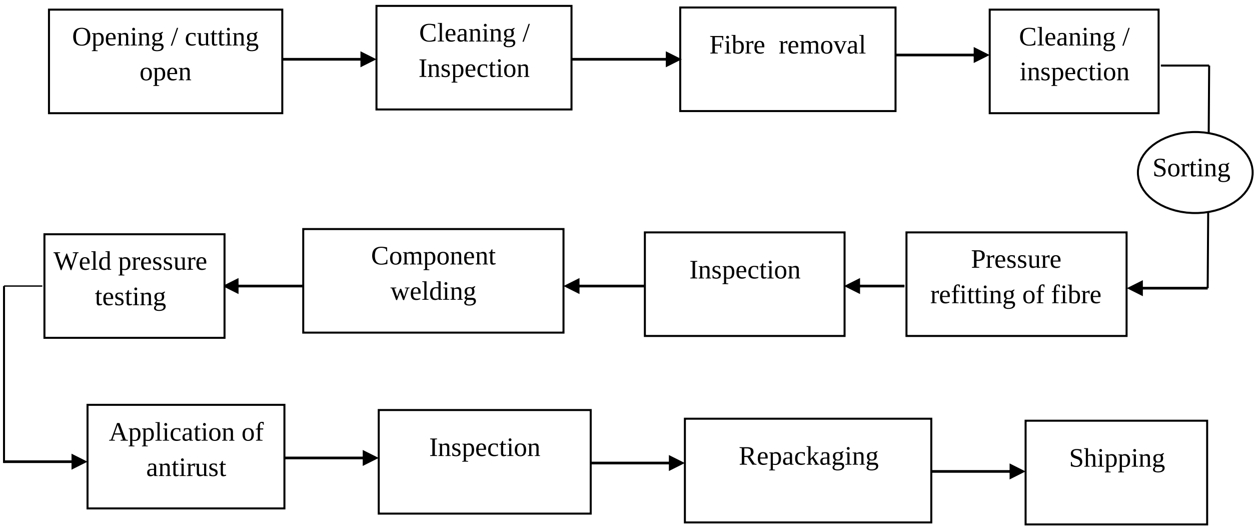


Figure 2.3 Validated block diagram of the TC unit remanufacturing showing the focus processes as shades

The TC remanufacturing process starts with the EoL unit opening, which is the stripping process of the product after the preliminary inspection of the core on arrival, intentionally omitted as part of the reverse logistics. Further mechanical activities include fibre removal, pressure refitting, component remanufacture or replacement, pressure testing, cleaning, and antirust application. However, automating these processes has been identified as a vital challenge to remanufacturing practitioners from the literature. Furthermore, researchers outlined that the remanufacturing processes are complicated, manually performed, and lack the tools and methodologies to achieve them [9]–[11]. Therefore, these practitioner needs inform the study's focus on developing tools to automate some of the manual processes in TC remanufacturing. Understanding these unexplored applications in the industry can improve the overall remanufacturing process efficiency.

Furthermore, the detailed description of these mechanical activities is not covered; however, the other processes, including sorting, inspection and process control during the TC remanufacturing, form a vital part of the automation investigation. The choice of these processes was informed by the capability to detach them from the entire loop for automation, thereby improving the speed of achieving the remanufacturing through semi-automated remanufacturing, where parts of a process can be automated. Nevertheless, the ability to model these processes with similar or identical data types was vital for using only image data to evaluate the performance of the developed models in remanufacturing product sorting, inspection, and process control. In addition, the process control application considers the post-cleaning inspection in remanufacturing, where inspection helps decide if the process would activate a pressurised drying system if there are waterlogs on the components.

2.5.3 Domain of Research

The domain of research is production and operation management (POM). Researchers have outlined the gaps between the theory of operations management and the practice [56], [57]. As the operations research domain considers data from the real world to investigate desired trends from specific research [58], these deviations of practical realities from theories continue to attract researchers to close the understanding. This research aims to close the gap in understanding the use of deep learning-based models in enhancing remanufacturing processes.

Conversely, as people are vital components of the operations management research domain, researchers have outlined the process approach to satisfy the needs of these people, especially customers. The process approach includes five vital stages: identifying the needs, analysing and designing a product or service to meet that need, obtaining the inputs to test the design, transforming it into a service or product, and finally, delivering the service or product [59]. The five-step process in the block diagram of the adapted POM approach is highlighted in Figure 2.4.

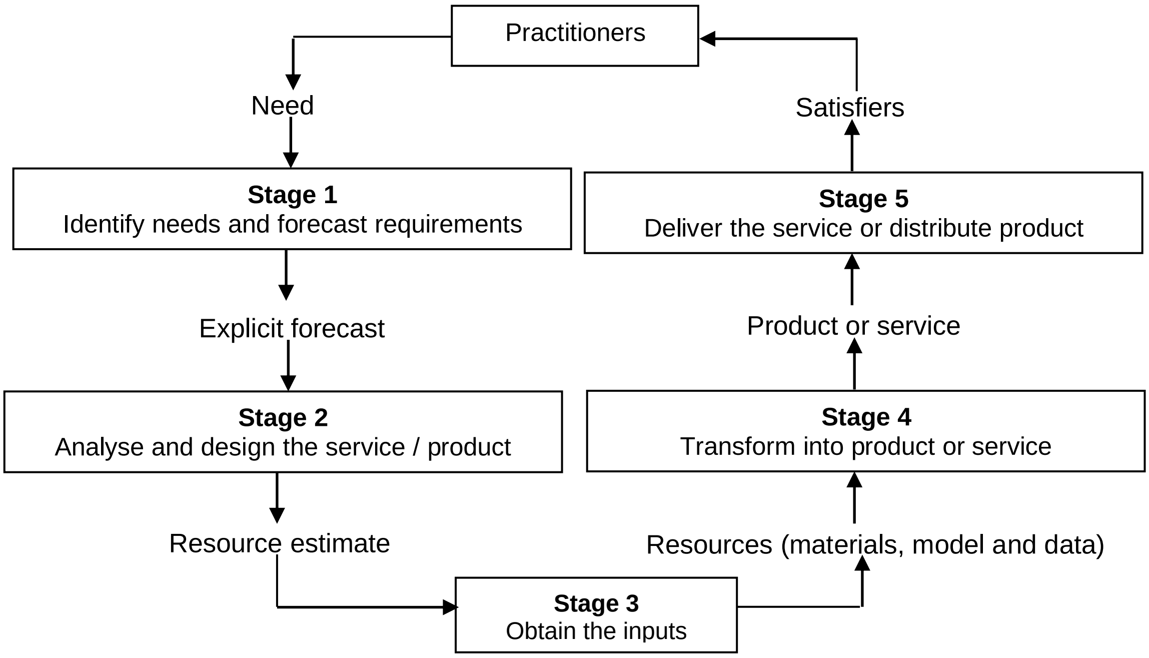

Practitioners
Need
Satisfiers
Stage 1
Identify needs and forecast requirements
Stage 5
Deliver the service or distribute product
Explicit forecast
Product or service
Stage 2
Analyse and design the service / product
Stage 4
Transform into product or service
Resource estimate
Resources (materials, model and data)
Stage 3
Obtain the inputs

Figure 2.4 Adapted block diagram of the production and operations management system [59]

Moreover, these needs of customers in POM are adaptable to the needs of practitioners in remanufacturing.

2.5.4 Model Choice and Requirements.

The choice of a computational model is vital as it considers the available data for modelling the processes. Since there is no available data for the processes to initiate the investigation, the researcher has to decide on the processes and the computational model to investigate based on the data collected from the remanufacturing facility where the research data is to be collected. Hence, the decision was made to record the samples' images for remanufacturing. The qualitative image data were recorded for processing by the model. The computational model that can process grid-like data becomes the most appropriate choice for the research, informing the selection of convolutional neural network models as they are suitable for analysing image data.

2.5.5 Identification of Remanufacturing Processes for Modelling

The generic remanufacturing process involves several steps that describe the entire process of returning a used product to 'as new' conditions with matching or higher warranty. The remanufacturing process involves identification, disassembly, cleaning, inspection, reconditioning, re-assembly, and final testing to arrive at a remanufactured product. This process is depicted in Figure 2.5.

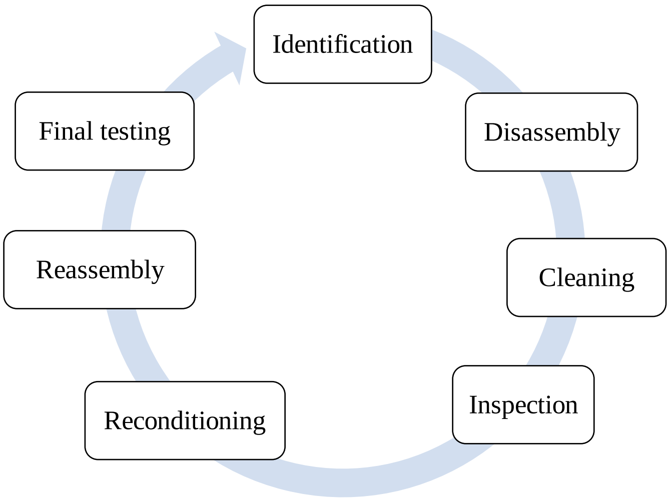


Figure 2.5 Schematic of the generic remanufacturing process.

The identification stage involves checking and reviewing a product to determine its make, model, and suitability for remanufacturing [60]. Unfortunately, the availability of all these product data is not readily available, making the application of these models in the identification stage unrealistic in the current settings. Besides, the disassembly stage separates the returned cores into single parts and is further classified into reusable and non-reusable products during the inspection. Early attempts to highlight whether a product is remanufacturable or not resulted in the development of significant importance for identifying the viability of remanufacturing, with researchers outlining that products for remanufacturing must meet the following conditions [61]:

- There must be a core for remanufacturing.
- The core cost is low compared to the actual value.

- The technology to restore to its original condition exist.
- Product technology is not evolving rapidly.
- The products are made to standard and with interchangeable parts.
- The remanufactured products sell for a high percentage of the original product market price.

Furthermore, these factors were considered when categorising a product for remanufacturing with subsequent stages following afterwards. Nevertheless, the inspection determines if the part condition has deviated from the original specifications. In addition, it ascertains the state of returned cores and makes the best and most profitable decisions about their future use [62]. Moreover, the cleaning stage involves numerous cleaning materials and methods, including water and high-pressure jet cleaners to degrease, de-rust and de-oiling the disassembled products for further work [63].

Nevertheless, the reconditioning, reassembly and final testing stages involve a series of repairs, replacements, testing, and coupling of the remanufactured parts into products. These final stages complete the remanufacturing process in a typical automotive remanufacturing setup. It is worth outlining that there could be additional or even lesser stages to achieve product remanufacture in some other industries. An industry case-by-case remanufacturing setup was investigated and outlined in the literature [64].

Conversely, an investigation into the automation of the entire remanufacturing process cannot be achieved in single research based on the complexities of the processes, thereby creating a limit to what is obtainable within the research. Therefore, selecting the specific remanufacturing processes for improvement through automation in the research is based on significant factors, most importantly, the ability to model the processes from the collected research data. The remanufacturing processes include the identification stage referenced as sorting, inspection and process control applications. This scope is informed mainly by the research design protocol in section 2.1, which outlines that only qualitative video data would be collected for further processing. Therefore, these aforementioned processes are suitable for modelling using the collected data.

2.6 Industrial Collaboration.

Collaboration refers to working across organisational boundaries to manage and build value-adding systems to meet customer needs [276]. It enhances knowledge sharing and the relationship between organisations, providing the platform to improve product offerings and deliver essential customer demands. Furthermore, the general supply chain collaboration literature outlines numerous crucial factors that enhance performance through collaboration, including information sharing, trust and information technology [37]. These factors were essential to achieving seamless collaboration during the research. Besides, researchers suggest that successful collaboration can be achieved when organisations evolve against the factors that hinder collaboration. These include a lack of trust, poor strategic planning, vision, and commitment, poor organisational culture, inadequate information sharing, and lack of standardised methods of measuring performance [37]. Understanding these vital factors enhanced the researcher's contributions while interacting with the practitioners during the research to achieve success.

Moreover, the case selection explored the possibility of collaborating with the industry, enabling the incorporation of experts with domain knowledge in the investigation. Hence, it will enhance the understanding and validation of the studied data.

2.6.1 Benefits of the Collaboration

The benefits of collaborating in remanufacturing research are tremendous, especially for participating institutions to share knowledge, meet customer needs, improve quality of work, enhance competitive advantage, improve delivery time, enhance product accessibility and improve revenue [37]. In addition, the research output on collaboration enhanced understanding of the requirements for successful collaboration among institutions, with significant potential to advance industrial practices.

Conversely, research benefit from the collaboration includes enhancing the understanding of remanufacturing processes, identifying vital processes for improvement, and accessing remanufacturing-specific data to advance future investigations on using various learning algorithms in modelling remanufacturing process challenges as well as establishing links to various remanufacturing industry stakeholders.

2.7 Data Collections

Data refers to any recorded factual material collected, processed, stored or used to justify or validate an original research result. It is also a collection of information used to approve or disapprove a research claim, theory or extend knowledge around a specific topic [49]. Research data can be qualitative, where verbal and non-verbal data, including questionnaires, documents, lab and field notes, audio, video, images, and transcripts or quantitative when numerically expressed or classified [49], [65]. Furthermore, for systems and processes, asking practitioners, observations and system documentation are vital sources of information [66].

Nevertheless, researchers highlight broad data collection techniques, including observation, secondary data, experiments, and derived forms [65]. However, the specific data collection type determines how to manage and store them for future use and processing. In industrial context, product conditions are recorded to properly assess and monitor health using various sensor systems, including cameras, ultrasonic, acoustic, accelerometer, current, thermocouple, radio-frequency tags, built-in encoders etc. [30], [279]. These sensor signals help to process system, product or component status[30], [67][30].

Conversely, the empirical data collected for this research were images of torque converter components obtained through the connected camera. The torque converter system is an assembly that primarily couples fluids found in automatic transmission engines, transferring rotational power from a prime mover to a driven load. It is located between the transmission and the engine flexplates. Besides, the data acquisition systems used for collecting the data include a universal serial bus (USB) camera of resolution 640 X 480 pixels, with lens F/2.0 and f=4mm, programmed to record the video of the object samples. To achieve the setup, a computer with pre-installed Python software and codes to open, record and capture a three-minute video of the samples from the camera and a clamp to fix the camera to a permanent position. It was necessary to minimise the inherent measurement errors where possible, as researchers highlight that most errors in research are caused by the data collection procedure [54], thereby enhancing repeatability. The respective videos provided the samples of each object class used to create the dataset, which helped train the developed learning algorithm, as detailed in Section 2.8.

Nevertheless, most of the collected data were videos of objects recorded from the experimental setup and converted to image data alongside the validation data. The location of the primary data collection includes the Mackie Automatic Transmission Limited Glasgow UK and the University of Strathclyde Design, Manufacturing and Engineering Management Workshop. The data collection involves recording videos of the samples using the researcher-developed Python algorithms and stored on a hard drive for further processing and analysis. In contrast, the validation data were returned through the validation questionnaires. The recorded visual data provides holistic conditional information about a product or component conditions. The videos are converted to images, with each image representing a data point and the collection of the data points making up the dataset.

Moreover, the data collection process of the torque converter components for remanufacturing considered the lighting and background of the actual remanufacturing operation, with the samples recorded directly on the conveyor systems during operation. This is informed by validated research that considering the operation background improves the model classification accuracy [86]. The video stream was set to always preview to quickly identify when the camera malfunctions while working.

2.8 Research Data

The research uses a collection of image data recorded using standard USB cameras. These collections of samples are often referred to as datasets. Datasets are generally a collection of examples or data points [68]. Researchers have outlined that the general characteristics of a given dataset fundamentally influence the behaviour of any reference model [69]. These examples constitute the experiences the learning systems will use during training, thereby attaining the ability to perform a given task. Furthermore, researchers have also suggested that the amount of skills required to deploy a deep learning-based model continues to reduce as more and more data becomes available to the models [68], making data an essential component for improving deep learning models.

Besides, dataset creation is a tedious job requiring a lot of time. It is expected to have information about its creation process and essential details about the makeup. The early datasets in general machine learning research include the Iris dataset, which contains measurements of different parts of 150 selected iris plants, with each plant corresponding to one example [70]. The features within this dataset are the respective sepal length and width alongside petal length and width measurements, representing three species of plants in the dataset. This dataset inspired numerous collections and annotations of research data, leading to the creation of other massive datasets, including ImageNet, COCO, CIFAR, FERET, and many others.

Moreover, the general computer vision challenges use standard datasets containing millions of images helpful in training deep learning model architectures for the specific application area. These datasets are characteristic of being huge in numbers and valuable for extracting features in the data. The first computer vision dataset is the classic handwritten image dataset of the Modified National Institute of Standards and Technology (MNIST), released in 1999. As the learning algorithms and techniques advance, this dataset has continued as a reference dataset. Other standardised datasets used in various state-of-art results reported in published computer vision articles include ImageNet [71], CIFAR [28], MNIST [72], [73], PASCAL VOC [74], COCO [75] etc. The MNIST dataset contains 70,000 handwritten images, with 60,000 training and 10,000 test samples [72], [73]. The MNIST was used for digit recognition tasks, but the visual recognition challenge was birthed due to advancements in algorithm development and the need to track the advancements.

Furthermore, the PASCAL Visual Object Classes challenge, known as the PASCAL VOC dataset, consists of two components: an annual competition with a workshop alongside a publicly available image with annotation of ground truth and standardised evaluation software. This competition started in 2005 and increased the number of objects used in the datasets in the subsequent years. The PASCAL VOC has five challenges: recognition, detection, segmentation, action classification, and person layout [76]. However, it is worth stating that the PASCAL VOC dataset is used to test new advancements in algorithm developments; however, the PASCAL VOC challenge has now finished [74].

Nevertheless, the early object recognition and image classification tasks used the large-scale visual recognition challenge (ILSVRC) dataset, also known as ImageNet. It is the pioneer dataset of millions of labelled images used for training and testing deep learning algorithms. Furthermore, this dataset is used to test the progress of computer vision applications for extensive scale image annotation and retrieval[71]. Besides, these standard datasets are usually grouped into two categories; the publicly available datasets and the annual competition datasets, where entrants train their algorithms using the provided training images and automatically annotate the test images as results, alongside submission to the evaluation server. After the competition, the results of the state-of-the-art algorithms are published, with the authors invited for insights.

In addition, the COCO dataset, the Common Objects in Context, is another standardised dataset for large-scale object detection, segmentation, and captioning, with ninety-one object classes and over two million labelled images. It represents objects in the natural environment with specialised features, including recognition in context, object segmentation and super-pixel segmentation [75]. In addition, the developers focus on finding objects within a scene from varied viewpoints [77]. Other more recent scene recognition datasets are the Places dataset, with over seven million labelled images [78] and the SUN dataset, with around a million labelled images for each of the ten scene categories and twenty object categories [79].

Conversely, It is worth highlighting that these datasets have continued to advance as the algorithms improve; however, the growth of the datasets has been slow, as suggested by researchers [79]. In addition, the contributions of the standard datasets have been massive, especially in stirring more interest in developing more advanced learning algorithms over the years. Furthermore, these datasets have provided robust techniques to detect and recognise objects, describe scenes alongside scene attributes.

However, it is also worth stating that these datasets consist of different data formats, including images, videos, texts, tabular forms etc. and have been used in traditional machine learning research. However, most of these datasets have been extensively used in deep learning applications from digit recognition, face recognition, gesture recognition, video classification, text characterisation and many other applications, with no dataset specific to the remanufacturing. This crucial observation provides another gap that the current research addresses by providing computer vision data for modelling remanufacturing processes, especially for sorting components of the torque converter system.

2.8.1 Limitations of Existing Dataset

The existing dataset for research in machine learning applications has recently been criticised for biases attributed to the makeup of the datasets, with other fields like remanufacturing not having field-specific datasets for deep learning research. However, these biases tend to cause the AI models to produce undesired results. To remedy these biases, IBM recently released a new dataset for face recognition research called the Diversity in Faces (DiF) dataset, containing one million human facial images [80]. This dataset aims to correct the biases in the current face recognition algorithms.

However, other research fields with similar biases in the current datasets or no datasets require improved, more balanced datasets to complement the performance of these state-of-the-art algorithms. The industry collaboration provides remanufacturing specific industry data for deep learning applications. The respective data for modelling the torque converter systems provide new data for inspecting and sorting torque converter components and units during remanufacturing.

2.8.2 Remanufacturing Data for Deep Learning Research

The remanufacturing dataset created by this research is described. The data collection process outlined in Section 2.7 details the data recording methods. The created dataset becomes the first public remanufacturing computer vision dataset of torque converter components and units for remanufacturing. Datasets generally have been highlighted as an integral part of object recognition research and the main reason for measuring techniques for comparing and evaluating the performances of algorithms over the years. Perhaps, datasets have also been identified as the main limiting factor that has constrained the focus of general learning-based research since the performance benchmark number has been used for evaluating successes [81].

Nonetheless, there is no dataset specific to the remanufacturing sector, which is a significant barrier to advancing the application of learning models in remanufacturing. Therefore, the research contributes the torque converter component dataset, consisting of 71560 image samples. This dataset will attract more researchers to investigate other applications of learning models to other remanufacturing sectors and processes. The summary of the recorded data used in the research investigation is presented in Table 2.1, showing the applications, data type, number of classes, the number of images, and the number of images per class used in the training and evaluation of the models.

Table 2.1 Tabular description of the recorded research data

Process	Data type	Number of classes	Number of Images	Number of images per class
Sorting	Images	20	71560	3578
Inspection I	Images	8	28800	3600
Inspection II	Images	8	28624	3578
Process control	Images	2	14312	7156

2.8.3 Dataset Naming Convention

The naming convention adapted for the collected data includes the actual component's name and a suffix of numbers denoting the unique sample in the collection. For example, the inspection I and inspection II cases had eight classes of sample images named as follows: Dry samples (DS1), (DS2), (DS3), (DS4), (DS5), and wet samples (Wet1), (Wet2), and (Wet3). Furthermore, the second eight samples considered for the surface inspection application were: no defect (Nodef), crack fault (CF), pitting fault (PF), rust fault (RF), pitting and crack (PnC), rust and crack (RnC), rust and pitting (RnP), alongside pitting rust and crack (PnRnC) defects. Besides, the sorting process had twenty (20) classes of sample input images named as follows; Damper1, Damper1, Damper3, Housing1, Housing2, Housing3, Impeller1, Impeller2, Impeller3, PressurePT1, PressurePT2, Reman1, Reman2, Reman3, Stator1, Stator2, Stator3, Turbine1, Turbine2, and Turbine3. Furthermore, the process control case had two (2) classes of sample input images named wet and dry with an additional three-number suffix. The samples had 7156 images in each object class used in the process control experiment.

Finally, these samples will be made available to support future research on applying deep learning models in remanufacturing, thereby providing the first remanufacturing-specific dataset for modelling deep learning-based inspection and process control applications. These primary data were explored to achieve the experimental aims of the research by interpreting and predicting the contents of the collected image samples using computational models. Furthermore, this research adopts the nonprobability sampling method, allowing for proper inference from the research data, ensuring an in-depth understanding and knowledge of the specific contexts used as test cases, and restricting generalisation on similar cases.

2.9 Chapter Summary

This chapter describes the research design method and research philosophy. It further outlines the rationale for selecting the sequential mixed method research approach alongside a description of the data collection approach limitations of existing data. Finally, it discusses the datasets used in the research. It also outlines the research domain, the researcher's involvement, and vital research considerations to achieve valid and reliable research outcomes.

CHAPTER THREE

LITERATURE REVIEW

3.0 Introduction

Chapter three provides the research literature review that first introduces remanufacturing, automation, benefits and challenges. It also introduced deep learning modelling, the generic learning models and their respective remanufacturing applications alongside the opportunities in remanufacturing. It also outlined the different deep learning modelling parameters to address the research question (Q3) on understanding and improving deep learning algorithms, including the architectures of deep neural networks (DNN) used for modelling various applications and their makeup. Hence, the application of these models is reviewed from remanufacturing perspective to understand the state of research in DCNN (Q1). The literature review conception approach of this research is outlined in Figure 3..

.

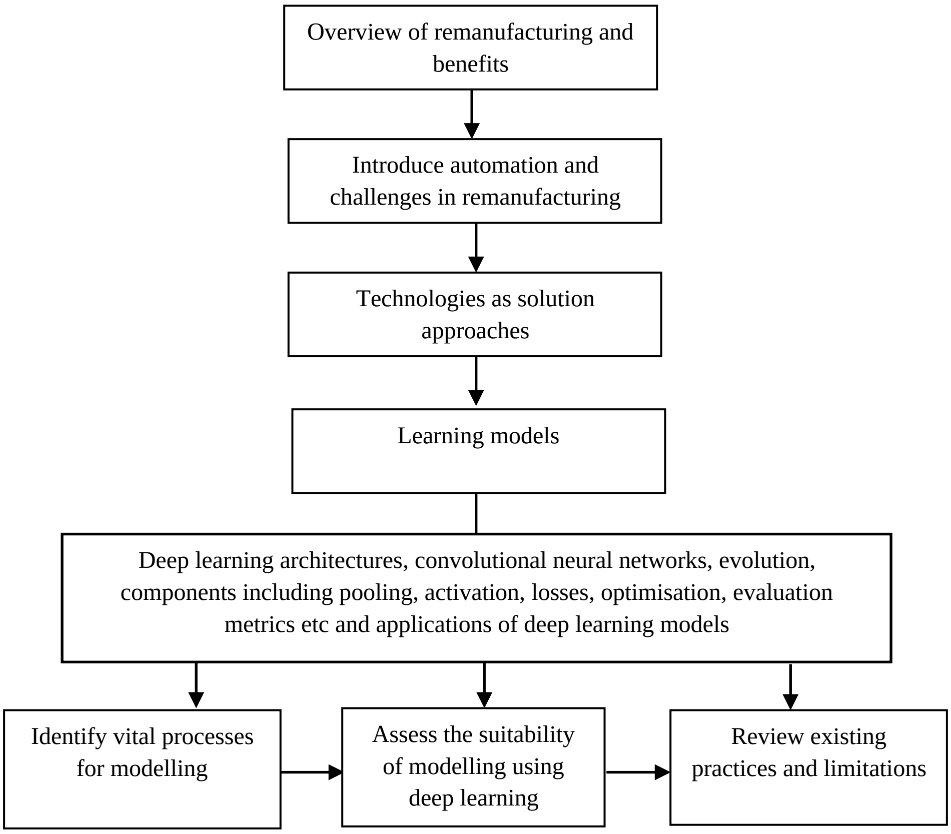

Overview of remanufacturing and benefits
Introduce automation and challenges in remanufacturing
Technologies as solution approaches
Learning models
Deep learning architectures, convolutional neural networks, evolution, components including pooling, activation, losses, optimisation, evaluation metrics etc and applications of deep learning models
Identify vital processes for modelling
Assess the suitability of modelling using deep learning
Review existing practices and limitations

Figure 3.1 Literature review ideation scope

3.1 Overview of Remanufacturing

Remanufacturing is an essential means of achieving sustainability in material, energy use and environmental protection[61] by restoring used products to as good as new quality, using only about 90% less material and one-sixth of the energy used for manufacturing equivalent new products [82]. Besides, It is a valuable strategy for continuing product usage that the manufacturers are no longer producing, supplies spare parts and manages warranty returns by industry operators [83], [84].

Remanufacturing has been described as an end-of-life activity to restore used products to "as new" condition with matching or more extended warranty. Remanufacturing has become a viable solution to increasing products' availability [85].

3.2 Benefits of Remanufacturing

Remanufacturing is an end-of-life strategy that provides numerous benefits, depending on the stakeholder in reference. These benefits are obtainable from different perspectives: the remanufacturer, customer and environmental benefits. The remanufacturers' gains include that It creates highly skilled jobs, improves profit margins, provides new manufacturing techniques, and creates a platform for better customer relationships through better trade-in opportunities [86], [87] and enhanced economic activities [88].

Nevertheless, environmental benefits arise from introducing the element of compliance with directives and regulations within territories. These have successfully increased the target level of recycling and reuse up to 95% as of 2015 [6]. Remanufacturing also maximises the added value throughout a given product's life cycle. It provides a platform to decrease the number of materials sent to landfills by reducing product waste, energy and material consumption, and carbon emissions into the environment through industrial activities[60], [86]. Specifically, remanufacturing reduces the number of raw materials consumed and energy used in the production process to about 10% to 15% of new materials and energy used for a typical remanufacturing activity[61].

Besides, remanufacturing customer benefits include providing superior quality products with good reliability compared to other product recovery techniques. The individual products are disassembled, assessed and restored independently or even replaced if the product cannot replicate original performance specifications[89], [90]. Furthermore, it enhances product availability, guarantees lower product prices, and provides flexibility in purchasing options when needed. It provides about 20% to 80% cost savings alternatives[87], [90] and serves as a source of spare parts highlighted from previous research[10]. It also offers economic benefits as the products are sold, on average, for much less than the price of equivalent new ones.

Despite these benefits provided by remanufacturing, other researchers have outlined that remanufacturing operations may not offer the vast gains anticipated. The expensive labour cost of remanufacturing since the procedures are human-intensive, the energy consumption and the carbon footprint to remanufacture a product have significant impacts [6], [91]. These authors suggest that emissions from transporting the products for remanufacturing and the effect of the chemicals used to clean products during remanufacturing are significant to ignore. Moreover, it is worth highlighting that not all remanufacturing operations use chemical cleaning techniques, which downplays the environmental pollution concerns. Furthermore, the supply chain concerns about transporting these products are case-specific as some remanufacturing facilities are located within the collection points, minimising the carbon footprint and enhancing the energy recovered from the process. Also, the considerable labour cost of remanufacturing has witnessed business owners sending most labour-intensive tasks to regions with lower wages. For example, Bosch performs her labour-intensive jobs in Ukraine and Slovakia while performing automation-intensive remanufacturing in Germany [6]. These downplayed issues highlight the need for further research to find new and novel approaches to enhance remanufacturing operations using emerging technologies like deep learning.

Nonetheless, the reverse logistics and the remanufacturing process pose severe challenges due to the lack of product information [92], the difficulty in disassembling the products, the indefinite quantity of returned products [93], the complex nature of the cleaning process [10], complicated nature of the remanufacturing process, uncertainty in ascertaining the condition of returned products [82], [94], challenges in reassembling the products and testing the products to verify that the quality meets the "as new" condition to mention a few. These and many other factors contribute to the challenges that must be addressed to enhance productivity in remanufacturing.

3.3 Remanufacturing Automation

Automation refers to using computer-aided systems and hardware such as sensors and programmable controllers to automate processes, reducing the dependence on human operators. To attain fully automated remanufacturing, systems should be adaptive to adjust to various product variations and conditions [95]. Furthermore, these systems sometimes allow collaborative work between humans and robots to interact, and the interaction has been defined as an interdisciplinary field of research. For example, robots can be deployed through collaboration and practical risk evaluation to perform hazardous operations while humans perform other cognitive and more flexible operations [96]. Besides, recent research has witnessed the automation of car remanufacturing case study using the human-robot collaboration where the sealant of an assembly was successfully performed using the cobots [97].

However, researchers have outlined vital challenges for practitioners in remanufacturing to achieve enhanced performance. These include the inconsistency of the quality of the remanufactured products and the labour-intensive nature of the remanufacturing process [98]. However, the systems' inputs are vital to address these challenges. Noteworthy are the two vital inputs to the production system, including the materials and labour, and these have been identified as the primary sources of poor performance [99]. Material productivity refers to using newer concepts, including material substitution techniques, to reduce components' input materials and weight, thereby improving performance and technologies.

In contrast, labour productivity entails using technologies, automation systems, and management methods across the production line to enhance throughput. Researchers outline that the manufacturing productivity sector has improved by more than three hundred percent in the past five decades due to improved labour productivity [99]. However, despite the outlined successes of these productivity initiatives, remanufacturing still lags in developing and adopting similar performance enhancement strategies, most notably the labour productivity techniques.

Conversely, the sources of inputs to the production system are vital control points to enhance performance. Recent research has focussed on techniques for enhancing material selection and usage during remanufacturing, including additive remanufacturing, also known as 3D printing [98]. At the same time, labour improvement is obtainable through different sustainable production approaches, including the use of active disassembly [100], procurement digitisation [101], pre-process inspection [102], automated inspection [103], electrochemical honing for the removal of hard mechanical alloys [104], cyber-physical systems [98], collaborative robots [105], and other technologies embedded in the Industry 4.0 framework. However, despite the technologies and processes specific activities to enhance performance, researchers outline that remanufacturing still suffers from poor automation due to a low-skilled workforce, lack of willingness to invest in automated systems, and lack of adequate tools and technologies, among others [6]. These challenges are broadly discussed as follows.

3.3.1 Challenges in Remanufacturing

The remanufacturing processes currently face crucial challenges that impede their overall throughput. These challenges have been discussed in detail in various remanufacturing literature. However, researchers classified the remanufacturing challenges into three categories: collections, often referred to as reverse logistics, remanufacturing process, and redistribution stages, differentiating the stages where these challenges appear in the overall cycle [106]. Perhaps, as there are no clear boundaries between these remanufacturing stages in practice, some of the difficulties in one step affect the subsequent stages.

3.3.1.1 Collections

The core acquisition and management constitute the first challenge that remanufacturing businesses have to deal with at the very beginning of the process. The primary core acquisition methods include volunteer-based returns and buy-back returns [107], ownership-based, service contract-based (leasing), deposit-based, credit-based (trade-in) and direct orders methods [108]. Some remanufacturers use these acquisition techniques independently and together to achieve the most profitable product collection. Nevertheless, the remanufacturing core management focuses on how products are managed, with the most critical decisions in procuring cores being the time, quality and quantity of cores.

Furthermore, another difficulty of core management is the complicated nature of reverse logistics as various groups are involved, including the OEMs, workshops, private suppliers, recycling, and disposal companies. Also, the challenges posed by the enormous logistics of the direct core supply by consumers and the scarcity of product life-cycle data helpful for predicting the remaining useful life [109].

Besides, inventory management is another critical challenge of the collections stage that remanufacturers have to manage. It involves predicting the supply and demand needs of the products alongside providing the balance for capital investment for products while keeping the stocks at an acceptable minimum to maximise profit. Research suggests that the available supplies must meet the short-term repairs needs of the remanufacturing process, and inventory management provides various methods of managing inventories, including make-to-order (MTO), make-to-stock (MTS), assembly-to-order (ATO), and the pull principle [89], [110]. However, the above inventory management techniques faced similar challenges of reducing or increasing stocks, causing avoidable costs for storage and disposal in the industry [111]. Perhaps researchers suggest that most remanufacturers have adopted mixed business models to reduce product demand and supply uncertainties, especially the MTO model [89]. Besides, research outlines that only one in three remanufacturers currently include prognosis in their inventory management [89], creating the need to develop more robust techniques to enhance remanufacturing efficiency, which is vital for production planning and control.

3.3.1.2 Remanufacturing Process

The remanufacturing processes constitute the most contributory factors to poor productivity in remanufacturing. These processes involve all the product remanufacture stages, including inspection, cleaning, sorting, disassembly, reconditioning, reassembly, testing, and storage. In addition, these processes have different inherent challenges that make the remanufacturing process difficult, including the small batch sizes of operations, vast product diversity, complicated production planning and disassembly, low degree of automation of processes, stochastic routing, and products not designed for remanufacturing, among others [89], [110].

Conversely, another area of challenge in remanufacturing is resource planning which includes activities to manage labour, raw materials and parts supply [107]. Besides, research also suggests that remanufacturers have managed the vast product varieties through different approaches, including the use of customised material requirement planning (MRP), theory of constraints (TOC) such as drum-buffer rope and classic inventory control methods, including economic reorder levels and reorder points and finally the Just-In-Time methods like the Kanban systems [89]. Furthermore, the disassembly process also introduces some complexities in the remanufacturing process. It directly impacts the production plans, material and resource plans, scheduling and shop floor controls and requires a reasonably high degree of coordination to improve productivity [89].

Perhaps, the most critical concern in the remanufacturing process is the complicated processes involved in the remanufacturing of EoL products, which are too broad to discuss independently. In addition, these concerns draw attention to the possibilities of incorporating new technologies to improve process automation in remanufacturing, focusing on the sub-processing to enhance efficiency and productivity.

3.3.1.3 Redistribution

The challenges of the remanufactured product redistribution are also an important consideration to achieve optimal productivity. These challenges arise from the uncertain demands for remanufactured products caused by crucial factors of the perceived differences between remanufactured and new products and the young state of the remanufacturing market in general [110]. These factors are also vital to maximising productivity as adequate efforts to provide storage facilities for remanufactured products that are not immediately dispatched are essential to enhance smooth operations alongside forecasting the size of the storage facility required.

Perhaps, the investigation of remanufacturing practitioners' challenges cannot be exhausted in single research; however, some of these challenges could be addressed using learning algorithms. Following the successes of the outlined prevalence of machine learning and digital automation in the manufacturing industry [112], remanufacturing can benefit from replicating compatible applications. However, as the remanufacturing processes are more complicated than the manufacturing processes, developing similar or new methods and technologies for automating remanufacturing processes is necessary to achieve holistic automation since the existing technologies in the manufacturing sector cannot work in the remanufacturing without adequate modifications. Therefore, exploring these new methods and technologies for improving remanufacturing is critical to addressing the technology gaps, alongside implementing and adopting the developed technologies. The investigation, design and implementation of these digital automation strategies in the remanufacturing industrial context supports the research on developing and deploying deep learning models, especially the convolutional neural networks in various remanufacturing applications. This context is informed by the excellent results of the deep convolutional neural network models, which have surpassed human-level accuracy in recognition tasks [33].

3.4 Learning Models and Technologies as Solutions

The learning models and technologies represent one of the solution methods for addressing crucial remanufacturing challenges. These approaches present vital opportunities for learning algorithms to be incorporated in systems design for remanufacturing process improvement, thereby enhancing productivity. These technologies will address specific productivity concerns and provide significant benefits, including insightful and data-driven decisions using product life cycle data, improving the efficiency and quality of remanufactured products. These digital technologies have played essential roles in the industrial landscape in the last decade and will continue to dominate even in the nearest future. The beneficial roles of these technologies have been outlined for the manufacturing industry, especially for digital automation, where researchers have highlighted that it is currently the general automation approach alongside machine learning models [112]. Besides, these technologies provide cheaper options for achieving process automation, enhancing overall efficiency when deployed.

Moreover, deep learning is a subfield of artificial intelligence (AI), an emerging technology that has witnessed tremendous applications across different industries, transforming the predictive capacity of learning models. AI refers to the simulation of intelligent behaviours by perceiving the environment, understanding the behaviours and responding to the perceived behaviours [68]. The AI models adapt artificial neural networks, machine learning, deep learning, reinforcement learning, and other technologies to learn the underlying patterns in data. The learning models research has advanced to address the challenges of early adoption of machine learning which includes processing raw data and automatically providing model features without manual inputs[28], [113], resulting in high-performance models with low resource and time investment [112]. These improvements reflect the current design methods that involve creating a model, preparing the data, training, evaluation, and deployment. The respective components of the pipeline consist of a set of codes that perform the specific task in the pipeline representing the typical deep learning modelling approach. These advances have inspired countless improvements across industries and encouraged more developments of new methods of improving processes and workflows.

3.5 General Learning Approaches

Learning models are mathematical algorithms that represent the relationship between different parts of a given data. These models map certain variables in the data to specific targets or responses. The learning models have adopted various approaches to achieve pattern learning. The general objective of these algorithms is to obtain a function that minimises some loss over specific data. These approaches are usually categorised based on the type of data features available to the learning algorithm. For example, the rule-based system uses hand-crafted features to obtain its corresponding output. The traditional learning approach uses similar hand-crafted feature designs to map features from input to output. Other techniques include representation learning, which uses mapped features to obtain the corresponding outputs. In contrast, the deep learning approach learns simple features from inputs and more complex features from the hidden layers, mapped together to get the output.

Nevertheless, the main difference between the traditional and other learning techniques to deep learning focuses on the feature extraction techniques. Most early design approaches were implemented successfully to classify images using hand-crafted features. However, these design approaches are inherently time-consuming and require immense domain knowledge and careful engineering of features [28]. A comparison of these learning paradigms is shown in Table , which outlines that the DL approaches use a layered learning structure where simpler features are learned by the initial layers, with the more complex features learned by the multiple hidden layers before feature mapping to obtain the output representation of the inputs.

Table 3.1 A comparison of different learning approaches adapted from [75].

Approach	Learning Steps				
Deep Learning	Inputs	Simple features	Complex features	Mapping from features	Output
Representation learning		Features	Mapping from features	Output	
Traditional machine learning		Hand-designed features	Mapping from features	Output	
Rule-based learning		Hand-designed features	Output		

3.6 Learning Models in Remanufacturing

The application of learning models in remanufacturing is not a new trend in remanufacturing research; however, deeper architectures are emerging as researchers investigate the more recent architectural advancements across different application areas. The deep architectures and other enabling emerging technologies, most importantly big data, overlap in actual implementation, suggesting that the learning-based models have other enabling technologies that support their deployment. Moreover, the remanufacturing sector has several challenges that require novel technologies to address across various remanufacturing stages, including core management, inventory management, product life cycle management, disassembly, process sequencing, material matching, and lean remanufacturing, among others [111]. The role of these learning algorithms in remanufacturing applications is presented to enhance inventory management, capacity planning, production planning, scheduling, forecasting, and many other benefits [8].

3.6.1 Operations Management

Learning algorithms have found various applications in remanufacturing operations management, including optimising reverse logistics, reliability and quality assurance [112] and redistribution. The reverse logistics involve managing returned products to capture value through remanufacturing, recycling, reuse and proper disposal [114]. Reverse logistics is mainly concerned with planning and forecasting product return quantities, probability and quality of product returns. It has witnessed the use of various artificial neural networks and neuro-fuzzy models [115], adaptive network fuzzy inference systems [116], Fuzzy Petri Net [117], and Fussy expert systems [118] to forecast product returns in remanufacturing. This modelling process enhances the planning of the product returns and collection processes. In addition, researchers also explored simulation models based on ordinal optimisation of remanufacturing process planning using machine learning methods [119], with the learning algorithms showing huge potentials in optimising reverse logistics however, the capabilities of these technologies is still an active research.

Another application of learning models in remanufacturing is inventory management. An adequately designed inventory system that meets the stock demand and supply improves overall productivity and throughput. The application accounts for the use of deep belief networks (DBN) to experimentally determine the feasibility of estimating the remaining useful life of the equipment. The authors successfully predicted the optimal remanufacturing time of mechanical transmission equipment [42], with the trained DBN model producing a prediction error of 27%. Despite the poor results, the application has enormous potential with advances in developing these computational models. Furthermore, another inventory management application of learning algorithms is the reinforcement learning (RL) approach in the planning and predicting the optimal strategy of maintaining service levels and switching between sources of materials during remanufacturing when core inventory is running low [120].

Nevertheless, using the computational models that focus on realising closed-loop product life cycles by enabling remanufacturing, reuse and customisation according to the customer needs is another excellent application of learning models. These models include customer specifications at an early stage in the value chain to meet the individual customer specifications and create the opportunity for product customisation [121]. Hence, this allows the remanufactured products to have new functionality and meet the original specifications.

3.6.2 Forecasting

Forecasting is another significant application of learning algorithms that plays a vital role in various remanufacturing aspects, including economic risk management, policymaking, and decision-making. Researchers have outlined two forecasting categories, casual and time series forecasts, which provide different benefits [122]. Time series forecasting is the dominant method due to the convenience of data collection, stability and high accuracy. At the same time, the authors identified causal forecasts as having inherent limitations due to the availability and reliability of independent variables.

Besides, some remanufacturing literature on learning algorithms has focused mainly on time series forecasting, which can benefit product returns and cost predictions, with the cost prediction model using semi-supervised learning, least-square support vectors regression algorithms considering failure characteristics, and the K-nearest neighbour algorithms to enhance forecast precision, being investigated [123]. Moreover, these learning models have also been helpful in the modelling and simulation of tyre remanufacturing for estimating product profit break-even points for different retreads [124]. The forecasting application is another area with massive potential for improving remanufacturing productivity and efficiency.

3.6.3 Factory Improvement

Factory improvement is another area that learning algorithms can enhance by using their vast capacity to model complicated processes, thereby improving them [112]. It involves creating additional functionalities to equipment to extend their use, with retrofitting being the essential use, where more functionalities are added to the products beyond the original state when manufactured [121]. It provides a cost-effective way of upgrading existing equipment with actuators and sensor systems, supporting sustainable remanufacturing. The learning models have been helpful for a decentralised identification system for components using mobile applications to enhance responsive on-site identification of parts based on their mobile photo [125], thereby strengthening the sorting process of products on return.

Furthermore, these models have also been helpful in the recent development of smart factories, where the data-driven simulation of the WEEE remanufacturing process for material flow behaviour during remanufacturing is modelled and simulated using data from the intelligent factory-like connected sensor systems to highlight the information requirements and service layers to collect process data [25]. The factory improvement represents another area where learning models can significantly improve, especially in mining data from connected sensor systems.

3.6.4 Decision-Making and Support Systems

Decision-making and support systems rank among the first tasks performed to achieve remanufacturing, starting from reverse logistics, identification, sorting, and other remanufacturing processes. Remanufacturing decision-making is another vital application area where learning models find a considerable advantage due to the industry's ever-increasing product and process data [112]. Deep learning can leverage these massive product data to provide the information that can improve products and processes by extracting data, logging, processing, and retrieval, thereby generating meaningful insight necessary to provide highly efficient and reliable results. Nevertheless, data-driven decisions highlight the benefits of using the overall product or process data in decision-making. It involves using learning models to access large quantities of data and making more informed decisions about the process from its data. It is essential to highlight that researchers have recently suggested that data-driven decisions and demand prediction systems are getting attention in remanufacturing [26], [34].

Furthermore, another application of learning models in remanufacturing decision-making is using reinforcement learning methods to evaluate the feasibility of remanufacturing using the rough set approach to establish the relationship between a remanufacturing plan and its feasibility. The RL algorithms helped enhance confidence in feasibility analysis to determine whether to remanufacture a product and the resource needed, thereby aiding resource planning [126].

Recently, remanufacturing decision-making has witnessed the integration of data and knowledge systems among the effective methods of enhancing remanufacturing decisions. For example, the data-driven product return forecast has seen the use of shallow multi-layer perceptron (MLP) and support vector machines (SVM) algorithms to model the consumer storage behaviour statistically for electronic wastes [127]. Furthermore, data mining and ML techniques have helped predict customer demand for remanufactured products in the electronics remanufacturing industry, proving to be another application of learning algorithms, with the effects of demand analysed using partial dependence plots [128]. In addition, the multidimensional, deep neural networks have helped predict the technological life of electrical and mechanical products, thereby estimating the rate of technological degradation and informing the decision for developing higher technology products to extend their technological life [129]. This application adopted an ensemble model incorporating a convolutional neural network and long short-term memory model designs to predict degradation rate.

Furthermore, researchers have recently investigated the benefits of using data mining techniques to enhance reverse logistics decision-making using computer vision, text mining, and ML concepts to plan the remanufacturing process [130]. Although the results look promising for optimising reverse logistics, they outlined that the work is still in progress.

However, using learning models in decision-making is one of the most critical applications to enhance remanufacturing. Therefore, the availability of tools to improve decision-making is vital to optimising remanufacturing.

3.6.5 Remanufacturing Processes and Process Planning

Process planning is a crucial aspect of remanufacturing that helps manage remanufacturing activities and significantly enhances process efficiency and automation [131]. Remanufacturing, in general, has various uncertainties associated with the processes, primarily due to the complex nature of the techniques. Therefore, process planning is another area where learning models find application across the multiple remanufacturing stages, including identification, sorting, disassembly, inspection, cleaning, reconditioning, and testing [112]. First, the disassembly stage is one of the most complicated stages in remanufacturing because of its manual and labour-intensive activities, and it remains an active research area to date. Researchers identified the ease of disassembly as a crucial factor in achieving remanufacturing automation alongside being disassembly-friendly, making the design for disassembly an essential consideration during the product design [132] to enhance remanufacturing. However, the remanufacturing processes have a minimal application of learning algorithms.

Conversely, remanufacturing has already benefitted from these algorithms in modelling scheduling problems where various algorithms, including the discrete Bees and the multi-objective harmony search algorithms, have been used to simulate scheduling optimisation problems and optimal disassembly sequence [13], [14], [133]. These applications help to generate an optimal disassembly sequence that streamlines the disassembly process, thereby enhancing efficiency. Furthermore, the learning models are helpful in scheduling optimisation, including the repair, maintenance and overhauling of products [134] using the ant colony, a swarm intelligent algorithm based on probabilistic techniques and in the constrained ordinal optimisation of the remanufacturing planning for estimating the feasibility plans thereby selecting the most effective plan(s) with high probability [119].

Nevertheless, researchers have also explored the reinforcement learning (RL) approach to model uncertainty and management in optimal disassembly process planning using the Petri-net modelling approach [135]. However, the model was limited to being dedicated to a particular product type in the remanufacturing facility, restricting its usage. Furthermore, learning models have also witnessed practical application in disassembly sequence generation, where a CNN model and disassembly rules helped achieve disassembly sequences [136]. Overall, these learning models help plan and schedule remanufacturing activities, effectively improving process efficiency.

3.6.6 Remanufacturing Technologies

The learning models have found tremendous applications across various stages of remanufacturing, and some of the use cases are outlined. The learning models have found application in the design of the remanufacturing inspection technology, where researchers investigate the use of the machine learning approach and Gaussian mixture probabilistic models for automating the detection of corrosion in components [137]. Furthermore, these models have also been helpful in the design of the vision inspection system for remanufacturing [103]. Also, these models have been used to simulate the direct energy deposition of titanium alloys in additive remanufacturing and manufacture, where the Taguchi experimental setup was used to obtain training and test examples for the artificial neural network (ANN) [138]. The method successfully determined the grain growth behaviour during the fabrication process, thereby enhancing the reliability of reconditioned components of a product.

Furthermore, the application of learning algorithms, including the deep and recurrent neural networks, has improved the prediction of thermal field distribution from laser scanning, improving the understanding of the residual stress and distortion distribution in laser-aided additive manufacturing, and adding new materials to a product during reconditioning [139].

3.7 Opportunities for Deep Learning

The remanufacturing application of most of the learning algorithms is composed of shallow architectures used for extracting features for training the learning algorithms. However, the capacity of these shallow architectures in modelling complicated processes is limited, thereby drawing further attention to the investigation of the deeper architectures for remanufacturing applications. This is evidenced by the scarcity of research publications on deep architectures, often described as deep learning in literature.

The deep learning applications in remanufacturing remain an active research area, primarily to manage the complexities inherent in remanufacturing systems and to explore the complete automation of remanufacturing processes that have not been achieved [34]. These algorithms have been incorporated in remanufacturing process planning; however, improvements can be obtained by fully exploring the emerging technologies' scope, including data mining, big data, and optical character recognition.

Besides, AI technologies, especially DL, can also help remanufacture data compression, improving the storage capacity of data management systems. It is another area focusing primarily on techniques to manage and store the product's MoL data, which significantly lags behind other research areas compared to the different sectors. Data compression has recently become a research focus for researchers investigating methods of incorporating general machine-learning models in remanufacturing [130]. In addition, with further research to maximise the storage of MoL data, practitioners can develop and deploy models to make predictive decisions from the product usage data to optimise usage by extracting and utilising product information using data mining methods and deep learning models.

These product usage data can also enhance pre-disassembly assessment and inspection of products with the learning models aiding effective decision-making on the products to accept for remanufacturing alongside their remaining useful life, thereby determining the probable cost of the returned products.

3.7.1 Operations Management

The opportunity of learning algorithms in operations management, including improving the redistribution network to enhance productivity, is another optimistic area; however, there must be available data for the redistribution processes to stand a chance of achieving good insight using learning models. Furthermore, the learning algorithms, alongside other enabling technologies like distributed ledger technology (DLT) and IoT technologies, can improve reverse logistics, especially with smart contracts, which guarantee transparent, secure and tamper-proof systems to monitor and manage the product return process [140]. Furthermore, this approach can enhance the recovery of products through incentives to customers who return their products by providing adequate product tracking, real-time assessment of product health, cost-saving for remanufacturers from buying products without economic value, and maximising storage facility. However, these have not been fully explored for remanufacturing and represent a future research direction for reverse remanufacturing logistics.

3.7.2 Forecasting

Forecasting is one of the most practical applications of learning models across their application domains. These models have found use cases in developing and deploying predictive systems in the industry. The emerging trend in deploying predictive algorithms in remanufacturing has witnessed learning algorithms useful for planning and forecasting purposes, thereby improving decision-making and management of remanufacturing operations. Forecasting is another area that has seen more applications as researchers understand these models, guaranteeing core availability through excellent core return forecast and enhancing overall process efficiency. However, remanufacturing applications of these models are broad and cut across many trends that have not been fully explored.

The specific applications that can benefit from the predictive capability of these learning models include the product identification, analysis and forecasting of the middle of life and product data, alongside the other process-specific data where predictive models can play a significant role in improving the overall productivity. The models can also be helpful in prognostics where remanufactured product health can be monitored in real-time, thereby helping to track the product's life cycle. Furthermore, another area where learning models have not been fully explored in remanufacturing is the prediction of the remanufactured product demand and supply factors of the market. However, further exploration of the collections and returns data is also essential.

3.7.3 Factory Improvement

Factory improvement is another area where the learning algorithms can benefit remanufacturing, especially in the remanufacturing shop floor design. It is an application area where the potential of emerging deep learning technologies, alongside other enabling technologies, including IoT, big data, and data mining, can gather product information, process, and store data. However, these technologies effectively extract useful information from the stored product and process information. In contrast, simulation technologies and CPS can provide prototype processes for evaluating, optimising, and developing reconfigurable remanufacturing, thereby enhancing productivity. Furthermore, these algorithms and technologies for modelling the remanufacturing shop floor activities can also benefit the factory operations through robotic disassembly sequence planning, improving throughput [13], [141], and other remanufacturing stages.

3.7.4 Decision-Making and Support Systems

Decision-making is one of the most practical applications of learning models in remanufacturing. It has witnessed numerous applications involving data-driven decisions based on these algorithms; however, there is no current research on improving remanufacturing data management and storage, which can preserve the product data for future data-driven decisions. Furthermore, these algorithms can play significant roles in enhancing the forecast of uncertainties in demand, supply, and quality of products, and estimates on inventory with appropriate data for modelling, thereby providing potential improvements for remanufacturing. The learning algorithms, alongside other enabling technologies, including CPS, big data, and IoT technologies, are helpful in developing a data-driven system for scheduling and inventory of real-time manufacturing processes, thereby providing adequate decision-making at every stage during remanufacturing [142]. Another opportunity for the learning algorithms to enhance productivity in remanufacturing is improving prognosis, especially in inventory management.

These represent vital areas of future research endeavours in remanufacturing that have not been fully explored and represent an area of future research.

3.7.5 Remanufacturing Technologies

The opportunities for learning models in developing new technologies are enormous, and it represents one of the most active research areas for learning models in recent times. However, the use of these models across the various stages of remanufacturing has not been fully explored yet; more recent investigations have witnessed the use of learning models in inspection, sorting, and process control [41], [103], [137], [143].

Nonetheless, developing innovative technologies for product identification is one of the essential technologies required in remanufacturing. These technologies will enhance the evaluation of products on return, alongside sorting products for remanufacturing. The learning algorithms and other enabling technologies that can improve product identification include RFIDs, IoT, ICT and wireless communication. Besides, there are several opportunities to strengthen remanufacturing using deep learning and other technologies, including IoT and CPS, to automate the processes to remotely monitor critical indicators, especially during identification, disassembly, cleaning, and reconditioning to assure quality [144].

Furthermore, another area where deep learning models can potentially enhance remanufacturing is automated robotic applications and machine tools, which are effective methods to flexibly adapt to changes in the products and processes during remanufacturing. Also, comparing the stages of remanufacturing, the use of deep learning and other AI technologies in the development of cleaning solutions, and testing reconditioned units for remanufacturing is another future research area as more investigation is needed to develop model solutions for testing and cleaning.

However, regardless of these opportunities offered by the learning algorithms, the potential application of remanufacturing is usually determined by the availability of data on the specific applications. Therefore, the areas of application where data could be obtained for training these deep learning models are vital in developing and deploying deep learning-based systems in remanufacturing.

3.8 Suitability of Deep Learning Models

The suitability of learning algorithms to model remanufacturing processes was a crucial consideration since remanufacturing processes are complicated. However, researchers have outlined that a possible approach to managing these complicated processes includes breaking down the complex processes into smaller functional units and developing automation systems for the smaller units, which could be cascaded together to achieve a fully automated system [145]. This approach is significant because if the process cannot be broken into sub-processes, automating the process is unlikely, thereby hindering productivity.

Nevertheless, deep learning is already defined in Section 1.3 as a new valuable technique for identifying patterns in data using hierarchical layers and potentially benefiting various industries. Furthermore, these models have recently attracted researchers' interest in the deep learning literature, suggesting that there is minimal research extending the applications of these technologies to remanufacturing compared to the manufacturing sectors. As the remanufacturing industry has not been fully explored, the study focuses on extending these applications to remanufacturing.

Moreover, the other remanufacturing stages and processes require various automation methods, including hardware and software automation. Specific processes, including disassembly and reassembly, need mainly necessary hardware to achieve process automation, including mechanical robots and other sensor systems [146]. However, different stages of remanufacturing, including inspection, process control and sorting, among others, can benefit from the software automation methods, which require a sensor system to perceive the signal and the low-level algorithms that process and provide the desired control [41].

Conversely, the learning model involves two vital components: algorithms and data. The algorithms are a sequence of instructions that learn underlying patterns in data without requiring explicit instructions [32]. It outlines how the models learn from data while the data is a collection of historical samples on a given process. A typical example of these models is linear regression, which helps predict a function's value based on a specific number of inputs. The linear regression model takes the form:

$$y = \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \ldots.. + \theta_n X_n + \beta \qquad \textbf{3.1}$$

Where the y is the output, X is the set of inputs, θ_1 *to* θ_n is a set of model parameters, and β is the bias term. The model tries to learn the relationship between the inputs and output as the data is fed to the model. The model obtains the appropriate values of the parameters θ_1 *to* θ_n using gradient descent optimisation. The optimisation techniques will be discussed in detail in the subsequent sections. The machine learning approach eliminates the need to explore model parameter values manually. Hence, these obtained weight parameters are used for predicting new values of the output y based on the values of the inputs X_1 *to* X_n. These models infer from training data some set of parameters θ that models the relationship between the target variable and some inputs. Mathematically, we can represent the model as follows

$$P_r(y|x;\theta) \qquad \textbf{3.2}$$

The model represents the probability of an output y given a vector of variables x, parameterised by θ. The machine learning modelling approach eliminates complicated conditional statements in direct programming, improving the model's overall performance. The taxonomy of the machine learning and deep learning modelling shown in Figure 6 highlights the crucial similarities of these models since the models work on similar types of data; however, the pre-processing stage and subsequent stages outline the vital differences between these models.

The machine learning models use hand-crafted features extracted using specific feature detection algorithms like the scale-invariant feature transform (SIFT) [147], speeded-up robust features (SURF) [148], and the histogram of oriented gradient (HOG) [149], thereby making the performance of these models depend on the experience of the designer of the feature extractor. Furthermore, the ML models use feature selection algorithms like principal component analysis (PCA), decision trees (DC) and support vector machines (SVM) etc., to learn the patterns in the data before inference [150]. In contrast, deep learning uses multilayer architectures, including convolutional neural networks (CNN) and recurrent neural networks (RNN), alongside the model hyperparameters, optimisers and loss functions to learn the underlying features in the data before evaluation automatically.

The working of the deep learning models differs considerably from the general traditional feature extraction techniques that use filters and statistical properties of the image, like a histogram, to detect features within a given image. The traditional recognition methods used either thresholding techniques, edge detection, contour geometry, template matching, keypoint feature matching, semantic features matching, scale-invariant feature transform, and histogram of oriented gradients techniques etc., to identify interest points within images and use the features to recognise the objects within the images using algorithms like SVMs [149].

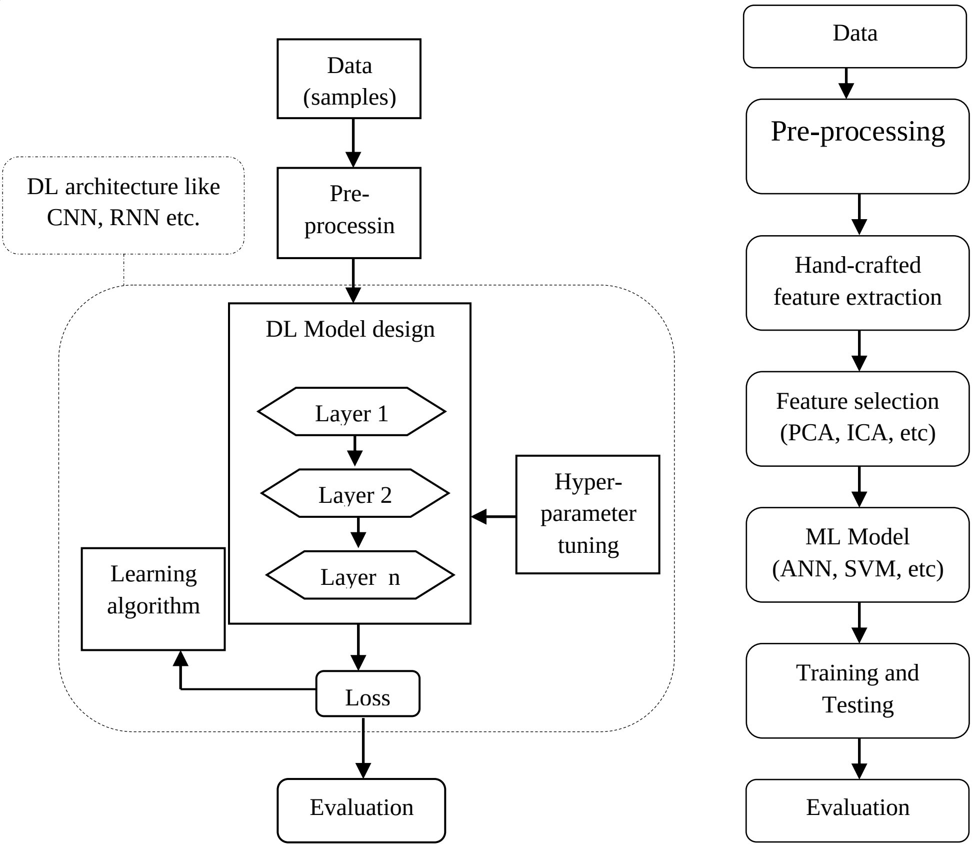


Figure 6 Taxonomy of deep and machine learning models

However, these techniques are time-consuming and suffer heavily from noise, and are primarily computationally expensive to implement. Hence, the deep learning models offer a speed advantage compared to the traditional learning approach, thus saving time to manually develop feature vectors that describe the objects within a scene before performing the classification. Furthermore, these DL models automatically learn these features without human interference. Therefore, automatic feature extraction ranks among the most significant advantages of DL as the model selects the features that best represent the available data, thereby improving inference results.

3.9 Deep Learning

Deep learning (DL) is a component of artificial intelligence research that uses hierarchical learning concepts to understand patterns in data. It uses the neuron as the basic building block and combines the neurons in parallel sequences to form layers. The neural network architecture design consists of the input, hidden, and output layers. The algorithm breaks down the input data into layers of abstraction, with the behaviours defined by the magnitude of the weights and the connections of the individual elements of the architecture. These weights are automatically modified during the training according to some specified learning rules until the model performs the desired tasks satisfactorily. Deep learning algorithms use an automatic feature extraction that differs from machine learning, which requires carefully designed features. The automatic feature extraction methods are not new since there are classic techniques of feature extraction, including singular value decomposition [151], principal component analysis [152], and non-negative matrix factorisation [153]. Besides, in the basic form, these algorithms obtain variables that are linear combinations of the old.

In contrast, deep learning algorithms learn non-linear combinations of variables, enabling more complicated modelling capability. This sequence of layers is helpful for mapping higher-level feature vectors from raw input images to the output layers [113]. The connections of a typical DNN architecture are depicted in Figure 7, highlighting the complicated nature of the interactions between neurons.

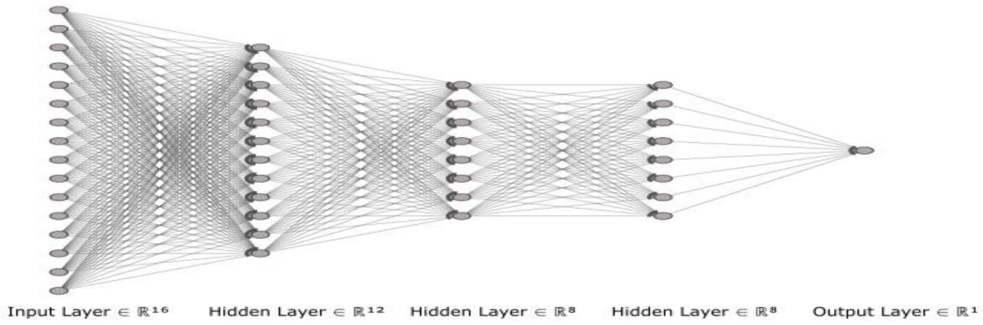

Input Layer $\in \mathbb{R}^{16}$
Hidden Layer $\in \mathbb{R}^{12}$
Hidden Layer $\in \mathbb{R}^{8}$
Hidden Layer $\in \mathbb{R}^{8}$
Output Layer $\in \mathbb{R}^{1}$

Figure 7 Typical neural connections in a deep learning model

The respective layers have several nodes connected to the previous layers, whose typical weights are adjusted during training (learning process). The magnitude of the weight parameter determines the changes in the strength of the signal to the specific connected neurons. Hence, the cascade of multiple layers creates the 'deep' networks, which refer to multiple layers of neurons stacked together. These features are learned when the micro-network strides over the presented input images to produce the feature maps [28], [154].

Moreover, the combinations of neurons produce neural networks, representing real-valued computations defined by some connected directed graphs [155]. The neural network nodes receive real numbers on their incoming edges, compute a function of these real numbers, and transmit the results to their outgoing edges. The root nodes perform their computations to the vector provided as inputs to the network, while the internal nodes compute their output to the output of other nodes. Hence, different nodes can add various functions to produce the desired outcomes. The distinctive characteristic of neural networks is that they can compute multiple layers to deliver results by combining an arbitrary number of non-linear operations. The typical neural network model is depicted in Figure 8, showing the inputs, weights, biases, activation function and outputs.

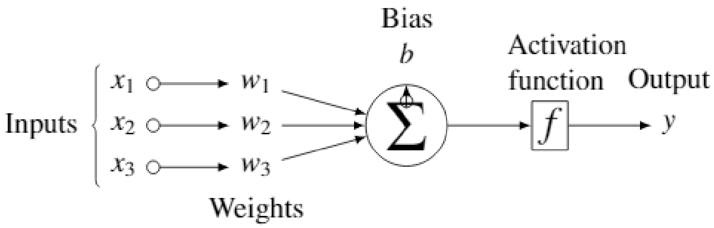


Figure 8 Typical neural network model

These models' output is obtained as a linear combination of the weighted sum of the inputs and biases. The vector product of the weights w and inputs x helps to get the model output; therefore, the inner product of the input and weight vectors denoted as h is given by

$$h = [x_1, x_2 \cdots x_n] . \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \tag{3.3}$$

The output of the model is obtained using the vector dot product where the output y is

$$y = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_n w_n + \mathrm{b} \tag{3.4}$$

The computation of the output of the model's respective layers is cascaded in multiple terms to obtain the very-deep neural network model. The deep learning architectures provide a significant advantage over the shallow architectures on complex learning tasks by stacking various linear and non-linear processing units in a layer-wise approach, providing the ability to learn complicated representations at multiple levels of abstraction. Hence, empirical research has proven mathematically that deep neural networks have more representational power. Furthermore, the deeper architectures gain more representation power by hierarchically composing shallow feature representations into deep model representations [155]. Besides, the advances in deep convolution neural network architectures have witnessed significant depth increments since the first architecture, LeNet, was used for digit recognition [156]. The depth of the newer architectures has multiplied rapidly, with AlexNet [157], VGGNet [158], GoogleNet [159], and ResNet [33] having eight, nineteen, twenty-two, and one hundred and two layers, respectively, among other architectures.

Deep learning has witnessed significant applications with unprecedented success rates across different domains, including image recognition, segmentation, video processing, object detection and natural language processing [32]. These successes have been attributed to the considerable research interests that have continued exploring the different approaches to enhance these models. These models allow learning from high dimensional raw data to automatically discover the underlying pattern, which is applicable in almost all fields with large historical data. These data are typical of most industrial processes, including remanufacturing processes. The ability of DNN to learn and generalise on unseen data after training gives it the ability to be deployed in new fields with new datasets, thereby enhancing the understanding of trends in the investigated data.

Conversely, the improved computational capability of newer hardware has aided the development and rapid deployment of DL models, especially the graphics processing units (GPU). Furthermore, recent advancements in image recognition have been attributed to the availability of large or big data and the ever-increasing computing power of computers, especially the graphic computing units (GPU), thereby facilitating the learning of very deep architectures. These advances led to the state-of-the-art results obtained in the annual Olympics of computer vision, known as the very-large-scale image recognition challenge (VLSRC) [160]. The GPUs are multi-processor graphics cards used widely in video games. They excel in the fast matrix and vector multiplications required for neural network training, thereby improving the learning speed by up to 50 or more [161]. In addition, GPUs have aided parallel computing, enhancing data access and computation speed. Similarly, the software is another primary driver of recent advances in deep learning. As a result, many newer toolboxes and models have been developed with improved code and techniques for implementing deep learning models.

Nevertheless, despite the benefits of improved performance provided by the deep neural network models, researchers suggest that even the smaller architectures can provide significant application enhancement, including the need for less communication to servers during distributed training, the need for smaller bandwidth to export models from the cloud and ease of deployment on field-programmable gate arrays (FPGA) and other memory limited hardware [162]. These challenges continue to drive the research on deep learning models to improve their architectural designs and model performances.

3.9.1 Brief History of Deep Learning Research

Artificial neural networks started in the 1940s after the first mathematical modelling of neurons[163]. The field attracted much more research interest until another remarkable breakthrough resulted in the perceptron, which used a single neuron to perform classification tasks [164]. Furthermore, the authors detailed the perceptron learning rule, which outlines how the perceptron works. However, a significant limitation of the perceptron is that it could not learn the exclusive OR (XOR) logic function. Further research continued, and the algorithm developments progressed until the back-propagation was proposed [165]. Additional application of the perceptron concepts continued. The use of neural networks for pattern recognition was first explored by Fukushima in 1980 when the self-organising neural network model that could recognise patterns based on geometric shape similarity without being affected by their positions was proposed [166]. This result, now known as the hierarchical multilayer neural networks, increased research interest in using neural networks for pattern recognition and further highlighted the huge potentials of neural networks.

Nevertheless, the back-propagation algorithm was reinvented to facilitate the training of neural networks[167]. The backpropagation algorithm provided the advantage of minimal preprocessing of the data before using them for training neural networks[168]. The researchers implemented the hierarchical architecture for object recognition and used backpropagation techniques to learn data representation. This approach involves an iterative adjustment of the weights of the hidden neural units, thereby minimising the difference measure between the actual and desired output vectors of a given network, creating new features about the inputs, captured by the interactions as the weights, which are used afterwards to learn a feature-based representation of objects by hidden layers [167]. These research results suggested that the hierarchical architecture outperforms the existing techniques for object recognition tasks.

However, the initial adoption of learning models were unsuccessful due to various challenges; limited data, when to stop the training, overfitting, wrong non-linearity models, little attention to the network initialisation parameters. These challenges contributed to the poor performance of these models, limiting development until 2006, when some remarkable breakthroughs in deep learning research manifested [28], [169], [170]. The field remains a very active research area in machine learning to date.

Conversely, the early machine learning techniques exploited shallow neural network architectures. Single neurons were used for signal processing, thereby containing a single non-linear feature transformation element with multiple inputs, where raw data conversion into problem-specific feature space is performed. These shallow neural networks include support vector machines, logistic regression, kernel regression, Gaussian mixture models, and hidden Markov models [171].

Besides, every instance found in any given dataset used by general learning algorithms is represented using the same feature set. However, these features can appear in binary, categorical or continuous forms [172]. The binary output produces one output from just two inputs. In contrast, the categorical output produces more than one output class during prediction alongside the continuous outputs with infinite numeric values.

3.9.2 Taxonomy of Deep Learning Methods

The general deep learning models use different learning approaches, including supervised, unsupervised, semi-supervised, and reinforcement learning. Supervised learning is the most common machine learning technique that uses labelled training examples to learn patterns in data and make accurate predictions. Furthermore, unsupervised learning uses unlabelled training examples to find the patterns in data. The architectures used in supervised models are mainly the convolutional and recurrent neural networks, while the unsupervised learning algorithms include other restricted Boltzmann machines, autoencoders, generative adversary networks (GAN), and some variants of RNN-based Long Short Term Memory (LSTM) and the reinforcement learning techniques.

Conversely, reinforcement learning techniques are models where the environment provides training information to the learning system[28], [172]. These algorithms learn interactions with the environment through actions, observations and rewards [173]. Besides, before selecting any actions by the agent (software or hardware), it must understand and have a befitting representation of its environment; thus, perception is a crucial problem that the agent must resolve before deciding on the optimal action to take. However, human experts provide the features of the environment to the reinforcement learning algorithms. Perhaps the features are learned automatically in some real-world applications to provide more accurate feature extraction. The RL algorithms allow an agent to learn by trial and error until a good understanding of the environment is achieved. The reinforcement learning technique is most often referred to as semi-supervised learning in some literature. It usually has restricted access to the optimisation function and, thus, interacts and queries it during learning to understand the process.

Besides, for a learning agent interacting with the environment, the number of parameters for optimisation determines the type of network to adopt. Models with fewer optimisation parameters use the reinforcement learning algorithm, while in models with many optimisation parameters, deep reinforcement learning techniques are adopted for the best results [170]. The RL agents are usually modelled as Markov decision processes (MDP), and depending on the states and actions spaces; the problem is modelled as infinite or finite MDP.

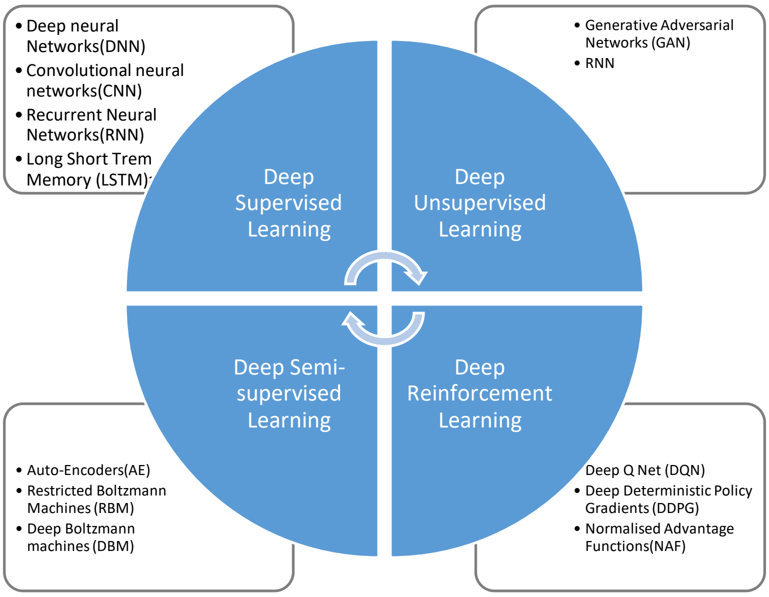
Nonetheless, most applications of machine learning models use supervised learning, and this learning model forms the basic theory of this research. It aims at performing classification tasks from labelled training examples. In addition, some general insights about the capabilities of deep learning, highlighted by an early study, produced the theoretical effectiveness of using deep learning, which attracted more research interest in deeper architectures [174]. Further research continued until 2012 when the first groundbreaking results on the application of deep CNN for image classification tasks [157]. Finally, the evolution of the learning models is discussed.

3.10 Deep Learning Architectures

The neural network architecture outlines the composition of the model with specifics on the number of units and the interconnection between units. The idea of a single neuron used for information processing is often referred to as the perceptron. The deep neural network refers to the multilayer stack of modules used to compute non-linear input-output mappings during learning [28]. The architecture also details the direction of the flow of signals in a model, from inputs to outputs.

The modules are subject to learning, with each in the multilayered stack transforming its input to increase its selectivity and invariance of the representation. The deep architectures of depth ranging from 5 to 20 can perform a highly complex transformation of its inputs to output, sensitive to minute details and distinguishing irrelevant and insensitive variations like lighting, background, surrounding objects, and pose [28].

Hence, good internal representations are hierarchical [175], and the architectures of the multilayer neural networks work in similar hierarchies for different types of data presented in the model. Furthermore, these models exploit compositional hierarchies where higher-level features are learned by combining lower-level features [28], [68]. For example, the deep architectures for speech and text exist alongside the images, with the make-up of the architectures differing significantly. The speech and text are obtained from sound inputs to phones, phonemes, syllables, words, and sentences. The method is similar to image inputs, where a local combination of pixels from the edges within an image, edges form motifs, motifs combine into parts, and finally, objects [28], [175]. The deep learning model taxonomy is shown in Figure 9.



• Deep neural Networks(DNN)
• Convolutional neural networks(CNN)
• Recurrent Neural Networks(RNN)
• Long Short Trem Memory (LSTM):
• Generative Adversarial Networks (GAN)
• RNN
Deep Supervised Learning
Deep Unsupervised Learning
Deep Semi-supervised Learning
Deep Reinforcement Learning
• Auto-Encoders(AE)
• Restricted Boltzmann Machines (RBM)
• Deep Boltzmann machines (DBM)
Deep Q Net (DQN)
• Deep Deterministic Policy Gradients (DDPG)
• Normalised Advantage Functions(NAF)

Figure 9 Taxonomy of deep learning architectures adapted from [113]

Conversely, the forward propagation of information is the multiplication of the given inputs by the model weights and biases before summing them together and applying the non-linearity function to produce the outputs for the given neural network. The deep architecture is a multilayer stack of simple modules that learn model parameters by computing the non-linear input-output mappings in data. The respective modules transform their input to increase the selectivity and invariance in representations[28]; thus, each architecture has numerous layers of non-linear processing elements, with the lower layer's output fed directly to the immediate higher layer [171]. The learning methods for deep architectures include multiple-layered neural networks [176], multi-layered graphical models [73], non-linear embedding algorithms [177] etc.

Consequently, the multi-layer neural networks use layers organized and arranged in a chain structure, with each layer being a function of the preceding layer. In this way, the model output is obtained by computing the outputs of the successive layers from the input layer to the second layers, and the following relationships give their outputs:

$$h^{(1)} - g^{(1)}\ (w^{(1)T} + x + b^{(1)})$$
$$h^{(2)} - g^{(2)}\ (w^{(2)T} + x + b^{(2)}) \qquad \textbf{3.5}$$

Where h = outputs of the layer, x = input, g = activation function, and b = bias. The multi-layer neural network is a straightforward approach achieved using stacking layers together and formed the basis for the model design adopted in the research.

Moreover, the bias term is helpful to shift the activation function to either the left or right or allow the activation function to move in either direction, as the case may be. Furthermore, it ensures that a positive output is obtained even when no input is applied to the neural network. These subsequent layers are represented by referencing the superscripts which represent the layers. These architectures discussed in this review include the feed-forward neural networks, restricted Boltzmann networks (RBN), recurrent neural networks (RNN), convolutional neural networks (CNN), deep belief networks (DBN), generative adversary networks (GAN) and autoencoders.

3.10.1 Deep Unsupervised Learning Models

Deep unsupervised models do not require labelled data for training, and they learn the vital characteristics in data to determine the underlying structure. For example, deep autoencoders, generative adversarial networks (GAN) and recurrent neural networks (RNN) are typical unsupervised neural network models and are helpful in clustering, dimensionality reduction and generative models applications [113].

Conversely, the deep autoencoder (DAE) is an unsupervised model that uses more than one hidden layer to learn the encoding of input data. The autoencoder neural networks are trained to copy input data to an output. They use sets of recognition weights to convert a given input vector to a code vector and a set of generative weights to convert the code vector into an approximate reconstruction of the input vector [178]. Pictorially, the autoencoder is shown in Figure 10, where the input vector is represented as x, the output r, and an internal representation h.

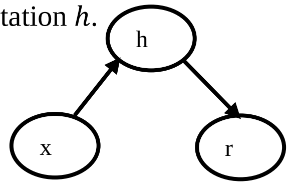


Figure 10 The auto-encoder

It consists of two-component units, namely the encoder function e, which maps the input x to an internal representation h, and a decoder function d, which maps the internal representation to the output r. The autoencoder output is given by

$$r(i) = x(i) \qquad \textbf{3.6}$$

Where r is the target or output, x is the input, i is an integer. During the DAE training, the network parameters are learned and compiled as feature vectors using one of the back-propagation techniques like the steepest descent, conjugate gradient method [171] and the recirculation techniques, with the newer technique, a biological training method rarely used for machine learning algorithms[179].

However, the DAE have the sparse variant, which has sparse features obtained by adding sparsity constraints to the hidden layer units, thereby modifying the loss function to

$$J_{\vartheta} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - x^{(i)}\right)^2 \ + \sum_{j}^{n} KL\left(p||p_j\right) \qquad \textbf{3.7}$$

Where n is the number of neurons in the hidden layer and KL is the divergence term. The $KL - divergence$ term with respect to the jth neuron is given as

$$KL(p||p_j) = p \ log\left(\frac{p}{p_j}\right) + \ (1 - P)log\ \left(\frac{1 - p}{1 - \ p_j}\right) \qquad \textbf{3.8}$$

Where p is a predefined sparse parameter that is close to zero and p_j denotes the average activation value of the jth neuron in the hidden layer over the entire training examples. The autoencoders produce a very sparse hidden representation of the input when the value of p is close to zero. Another variant of the autoencoder is the weight-decay added to a loss function to reduce the overfitting problem. The weight-decay AE is given by

$$J_{\vartheta} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - x^{(i)}\right)^2 + \tau \sum_{j=1}^{2}\left\|W^{(j)}\right\| \qquad \textbf{3.9}$$

Where τ is a hyperparameter that controls the decay strength. Nevertheless, the most significant strength of the AE models is that they are helpful in dimensionality reduction and can be modified to learn different representations; however, their main limitations include that they are not good at ignoring random noise in the training data, requires extensive training data, slow to train and fine-tune [113].

Besides, deep reinforcement learning (DRL) models are another form of unsupervised learning model with a foundation in Markov Decision Processes. These models learn behavioural policies by mapping states to actions, maximising cumulative rewards. In simplified form, the policy can be represented as a look-up table where the appropriate action for any state is listed. However, these listings are infeasible in more complicated environments and must be encoded as a parameterised function[180]. Furthermore, these models learn by punishments and rewards rather than explicit instructions [180], thereby understanding *"how to act in a dynamic environment from experience"* and responding by minimising some cost functions or maximising some payoff functions [181]. Besides, the learning process occurs by trial and error methods as the reinforcement signals are obtained from the experience of the interactions between the environment and the agent. A typical RL model with samples: $x_t \sim \rho,$ and agent forecasts: $\hat{y}_t = f\ (x_t)$, the probabilistic agent received cost: $rc_t \ \sim P(rc_t|x_t\ ,\ y_t)$. This is synonymous with semi-supervised models. These model derivations are beyond the scope covered as RL was not used in the research; however, the derivations can be found in the literature.

Furthermore, the different approaches of RL models include deep Q networks (DQN), Q - learning, deep deterministic policy gradient models (DDPG), normalising advantage functions(NAF), and the State-Action-Reward-State-Action (SARSA) methods [113]. The DQNs are the value learning models that learn utility values of the state and action pairs, often referred to as Q - values. The DQN models are the most used deep reinforcement learning models and have successfully trained self-driving cars [182]. However, a deeper review of the DQN architectures is not covered.

3.10.2 Deep Semi-Supervised Models

The deep semi-supervised models use labelled and unlabelled data in modelling and training tasks before inference. These models include the restricted Boltzmann Machines (RBM), which have two variants: deep belief networks (DBN) and deep Boltzmann machines. However, the respective semi-supervised models' characteristics distinguish them, and these characteristics are discussed as follows.

The RBM is one of the deep semi-supervised models that have found applications in dimensional reduction, regression, classification, filtering and feature learning applications. The restricted Boltzmann machines are a particular type of altered Boltzmann machines, consisting of a fixed number of two-valued units linked together by symmetrical connections [183]. These machines are symmetrically connected neuron-like network units that make stochastic decisions of being on or off in a probabilistic way. They are also used to discover fascinating features that represent complex regularities in a given training data [184]. The general Boltzmann machines are modelled based on parallel distributed computing techniques, where randomly initialised coefficients are helpful to modify the inputs. There are three different Boltzmann machines: conditional Boltzmann machines [185], higher-order, and mean-field Boltzmann machines. They also differ based on the network arrangement, either by conditional modelling or by the model initialisation technique. However, almost all Boltzmann machines have speed constraints, which could be addressed by restricting some network layers, thereby making some units invisible. These machines learn one hidden layer at a time, after which the activity of the hidden layer is applied for training subsequent restricted Boltzmann machines. This process is repeated to train as many Boltzmann machines as possible, producing the deep Boltzmann machine.

Conversely, the RBMs are primarily useful in unsupervised learning applications, especially in text classification, where the gradient-descent algorithm and an exponential loss function were used to tune the network, with active learning applied to train the text classifier [186]. However, perceptual learning and inference are simplified as the RBMs with no specific connections between their hidden units [187]. This stochastic dynamics of the RBMs can be described for a given unit entity i, having the opportunity to update its state in binary form; the first becomes computing the total available inputs x_i

which is the sum of the weights on interconnections w_{ij} and biases b_i coming from the active units, thereby producing the total input as

$$x_i = \sum_j w_j a_j + b_i \qquad \textbf{3.10}$$

where the variable a_j is the output state, and it is one if j is on and zero elsewhere. It turns out that the i unit turns on with a probabilistic logistic function as

$$P(a_i = 1) = \frac{1}{1 + e^{-x_i}} \qquad \textbf{3.11}$$

Perhaps, the main limitation of the DBM is the slower training process which limits the functionality and performance of the DBM models to mostly feature extraction applications

Conversely, deep belief networks are specialised semi-supervised models useful for efficient layer-by-layer top-down learning procedures with a generative weight that suggests how variables in one layer depend on the layer's variables above it [188]. The DBNs are composed of multi-layer stochastic latent variables, and the latent variable has binary values often referred to as feature detectors or hidden layers. The top two layers have undirected symmetric connections that form an associative memory. In contrast, the lower layers receive a top-down directed link from the layer directly above, with the lowest layer states representing the data vector. A significant property of the DBNs is that the layers are connected symmetrically, but there is no connection within the layers. The DBN also has a simple learning module containing an RBM, with visible units representing the data and a hidden layer that learns the high-order correlations. The unsupervised applications allow the DBNs to learn the features to reconstruct their inputs probabilistically. The pre-training stage learns the features, which are the initial weights parameters, while the fine-tuning modifies the architecture to achieve desired results. The DBNs models produce better results by treating the hidden vectors produced by the training data as the input for the subsequent learning modules [73].

Finally, the semi-supervised models have found applications in similar fields like the supervised learning models, where research outlines that these models have been successful in recognising images, generating images, video sequence analysis, motion capture data and analysis, dimensionality reduction as well as document retrieval [188], thereby making the semi-supervised models highly relevant in current and future applications.

3.10.3 Deep Supervised Learning Models

The supervised models use properly labelled data for training and inference on the computational models. The supervised models have found applications in feedforward, convolutional, and recurrent neural networks. These models are briefly introduced, their derivations and specific applications.

The feed-forward neural networks are algorithms where the inputs flow into the network and continue in one direction through the hidden layers until it reaches the output. The feed-forward networks have the values of any current node dependent on the previous layer nodes where x_i is the input layer, i is a function of x_{i-1} [68]. These networks are defined by the relationship that maps a fixed size input x to a fixed size output y, and are given by [28]

$$y = f(x; \boldsymbol{b}) \qquad \textbf{3.12}$$

These models are trained to learn the parameters $b,$ which best describes the function by minimizing its loss function $L(y, \hat{y})$ across the set of training data. The mapping function is a linear relationship between the input signal and the output; therefore, the output becomes

$$y = \alpha(W_i x_{i-1} + b_i) \qquad \textbf{3.13}$$

The mapping function can be re-written in linear algebraic form as $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{i-1} \end{bmatrix}$, where $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \end{bmatrix}$ and α is a nonlinear activation function. The feed-forward architecture with multiple hidden layers is often called the deep neural network.

Moreover, recurrent neural networks are specialised algorithms for sequence data like voice. It is more complicated than the feed-forward neural network and uses the same weights at every time slice to obtain inputs at every time portion of their operation. The RNN uses the parameter-sharing property of early machine learning models to extend the model applications and generalisation [68]. They can also remember the data in their hidden state for a long time since they have memories, although it is more challenging to train them to remember the hidden states. However, more recent algorithms have been successful in achieving this. The RNN usually has two sources of input, namely the recent past and present, combined to ascertain how best to respond to new datasets and a feedback loop connected to past decisions. The symmetrically connected networks use similar weights in both directions. However, their capability is more restricted because they obey an energy function.

Besides, the deep recurrent neural network recognises audio signals by first grouping them into low-level, and high-level frequencies or audio wave features grouped into phonemes and the phonemes grouped into words and the words grouped into phrases and sentences in a typical audio recognition system. The training of the RNN has successfully predicted the following sequences in data, especially the next character in a text [189] and the next word in a sentence [190]. When the RNN unfolds the time sequence in the data, it is visualised as a very deep feedforward neural network, with all the layers sharing similar weights. However, researchers have highlighted that learning and storing information about these long-term dependencies is challenging [191], creating room for improved approaches to succeed.

However, to remedy this, network augmentation with explicit memory was suggested. The long-short-term memory (LSTM) was proposed with special hidden units to remember the inputs for a long time using a memory cell [192]. The LSTM is one of the two main recurrent architectures, including the gated recurrent units (GRU). It consists of a cell that remembers values over an arbitrary time alongside input, output, and forget gates that control the flow of information in and out of the cell [192]. In contrast, the GRUs are gated mechanisms used as a forget gate and contain fewer parameters than the LSTM due to the lack of an output gate [193]. However, researchers have compared these architectures and highlighted that the GRU performs better than the LSTM with a fixed number of parameters for all models. The authors further assert that the performance metrics included CPU convergence time, generalisation, and parameter updates.

Finally, the major applications of the RNN architectures, including the LSTM and GRUs, have been in speech and audio modelling, natural language processing, and sentiment analysis; however other applications of excellent performance include the use of LSTM for the prediction of part quality in additive manufacturing [194]. Other notable applications include machine translation, automatic speech recognition [195], [196] and medical applications, where the RNN helped discover complex rules of biological protein application [197]. Finally, the closest application of the RNN to remanufacturing is the prognostic application, predicting the remaining useful life (RUL) of components [198] and bearings [199] alongside time series prediction[200].

3.11 Convolutional Neural Networks

The CNN is a feedforward neural network with multiple convolutional and pooling layers, helpful in providing end–to–end learning of the parameters of a given model. They are modelled according to the universal approximation theorem that a single-layer feedforward neural network can sufficiently represent any function given enough capacity. They are specialised neural network model for processing grid-like data and uses a mathematical operator known as convolution, which is a typical linear operation in one of the layers of the network instead of the general matrix multiplication used by the standard neural networks, in at least one of the layers of the models[68]. The convolution operation works by dividing an image into small slices, usually referred to as receptive fields. These smaller divisions help extract features from the images, thereby simplifying them. On the other hand, matrix multiplication involves separate parameters describing the interactions between component units, making each output unit interact with the input unit and causing the implementation of early neural networks to be computationally expensive. Moreover, making these architectures deeper to achieve higher accuracy has been a recent trend, giving birth to deep learning modelling.

Nevertheless, an attractive feature of the CNN models is the ability to exploit spatial and temporal correlation in data. The topology consists of multiple learning stages, including the convolutional, sub-sampling, and non-linear processing units [201]. The respective layers use a bank of kernels to perform numerous input transformations by extracting valuable features from locally correlated points.

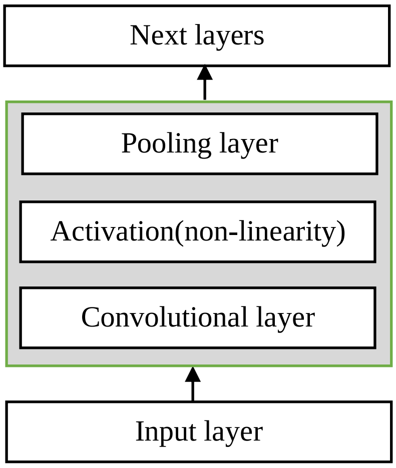


Figure 11 Components of a typical convolutional layer of a CNN showing the layer design approach

Moreover, empirical research outlines that given enough training data, CNNs can learn invariant representations in data to achieve and exceed human-level performance, as shown in Figure 12 [33].

Figure 12 Sample convolution filters output used to learn different patterns in data [125]

Besides, the 2012 annual Olympics of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) opened a new phase of CNN research as authors classified over one million images into 1000 classes using a deep convolutional neural network named AlexNet, setting a new world record accuracy in image recognition competition [157]. The research increased attention to using CNN in many other image classification problems. It produced other record-breaking architectures in the subsequent years, including ZFNet[202], GoogleNet[159], VGG16 and VGG19[158], and ResNet [33], to mention a few. The success of the CNNs has witnessed numerous valuable applications in real-world problems, including image recognition [157], [175], self-driving cars [182], object detection and segmentation [203], medical image analysis [204], emotion detection[205], remote sensing especially the synthetic aperture radar systems[206], [207], remanufacturing sorting [41] and inspection applications[143] among others.

Despite the early success and commercialisation of CNN, there was limited interest due to the limited amount of data, low computational capability of the available computers, and poor algorithms to compute the weights and biases of the neural networks. These challenges led to the evolution of CNN research, with researchers investigating methods of improving the design of CNN architectures to achieve improved performances. The components of these CNNs and their architectural evolutions are outlined in the subsequent sections.

3.11.1 Evolution of Neural Network Architectures

The architectures of the neural networks are a vital part of model development as these architectures influence the models' ability to generalise. However, research suggests that good generalisation ability is a function of developing architectures with a certain amount of prior knowledge about the problem. Nevertheless, the early research on CNN and their workings were not very clear on the internal workings of the CNN components, thereby treating them as black boxes [32], with the models having huge hyper-parameters and parameters including weights, biases, number of layers, number of neurons, stride, filter size, learning rate, activation functions and other hyperparameters. Moreover, recent research has improved and provided answers to the problematic research questions on CNN architecture, addressing the shortcomings of the previously proposed architectures and providing new structural formations.

The architectural evolution of CNNs highlights the advancements in structuring processing units and advanced block designs. In addition, these architectures also explored parameter optimisation methods, structural reformation, regularisation, and other techniques to improve model performance. These advances are outlined in the taxonomy CNN research. The description of the various design improvement methods is shown in Figure 13.

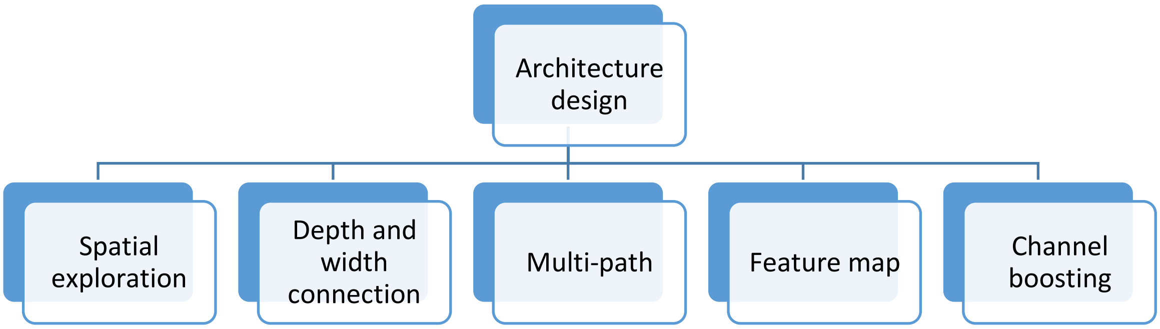


Figure 13 Taxonomy of the CNN evolutions

3.11.1.1 Spatial Exploitation

The spatial exploitation considers the neighbouring input pixels within the same locality and explores the correlations extracted using different kernels. These additional filters obtain different levels of detail, with the large filters extracting coarse-grain information while the small filters reveal fine-grained information. The early CNN research considered spatial filters to improve the CNN models alongside the relationship of the filters to the model's performance. These researchers observed that adjusting the model kernels improved results on the ability of the models to perform specific tasks. These spatial dimensions made new CNN architectures that became state-of-the-art in recognition tasks, including the LeNet [208], AlexNet [157], ZfNet [202], and GoogleNet [159] architectures, to mention a few. The LeNet5 architecture consists of two convolutional layers, two average pooling layers, a convolutional flattening layer, two fully connected layers and a softmax classification layer, used to perform digit recognition on 28 x 28 greyscale images [208] and was used for almost two decades before the recent explosion of the CNN research arrived, with another groundbreaking architecture called AlexNet. The AlexNet architecture is an eight-layer improvement of the LeNet, consisting of five convolutional layers and three fully connected layers, alongside some ReLU and softmax activation functions applied across each of the hidden layers of the architecture. The network was trained on input images of size *227 x 227 x 3*, with about a thousand object categories obtained from over one million images of the ImageNet database of images [71]. The architecture has some dropout applied just before the first and second fully connected layers to reduce overfitting and produced state-of-the-art results that surpassed the best-handcrafted image recognition and localisation entries [157].

Furthermore, these early CNN architectures explored the spatial details in data, including using different filters, padding, stride, and other hyperparameters to obtain state-of-the-art results with features extracted using large filters for more coarse details and smaller filters for fine-grained features. However, these architectures witnessed similar challenges because the results were based on trial and error because there were no clear reasons for the improved performances observed from the models [32]. Besides, other researchers propose using fixed topologies that are repeatable within the architecture and a handful in the design of VGGNet [158]. This architecture design method changed the CNN architecture design approach towards adopting the uniform layer design approach. Furthermore, deploying these repeatable units opened the door to developing innovative architectures that work in similar methods.

3.11.1.2 Depth and Width Exploitation

The depth and width of the CNN architecture is another vital architectural design method explored by researchers to enhance the understanding and performance of the architectural design of CNNs. The underlying assumption is that the deeper the architecture, the better the model's target approximation [209]. Besides, it is worthy of outlining that the deeper models have advanced the adoption of supervised learning research, with the early architectures exploring architectural depth in their design, including the network in network architecture [210], Highway networks [211], Inception-v1 also known as GoogLeNet [159], VGGNet [158] and the ResNet [33] architectures. These architectures focused on learning-rich features by exploring the model's width and depth. For example, the VGGNet and Inception architectures achieved the best performance at the 2014 ImageNet challenge using depth and width-based very deep architectures [158], [159]. However, as this depth increases, the gradient propagation diminishes, with increased computational cost, and longer training times, limiting the models. Therefore, researchers explored connecting intermediate layers to address these challenges, obtaining limited success.

Consequently, other methods, including the use of skip connections and gating mechanisms, provided improved performances of the deeper architectures [33], alongside the introduction of dropout in the residual blocks, which offers network regularisation [212], and the stochastic depth method that skips layers during training to reduce effective depth [213]. Nevertheless, the Highway network introduced depth and multipath in the design of CNNs. It increased the ease of information flow across several layers ranging from 50 to 900 [211], with the network having about 2.3 million parameters and a prediction accuracy of 92.24%.

Besides, the ResNet architecture is another depth and width-based model that uses identity shortcut connections to skip one or more layers during training. These identity shortcuts provide the advantage of not introducing additional parameters to the model alongside computational complexity and a higher prediction accuracy [33]. The ResNet model has the ReLU activation in the hidden layers alongside the softmax output layer [212] and was the first CNN architecture to surpass human-level performance with a record 5.6% top-5 recognition accuracy on the ImageNet challenge. Furthermore, the ResNet allows stacked model layers to fit a residual mapping rather than directly fit the desired mappings with multiple architectural variants, including ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. These variants have different convolutional layers, increasing the model's depth, complexity and accuracy.

Conversely, the width of the learning models has also been exploited in the design of model architectures since researchers suggested that stacking layers together may not learn the expected feature representations to improve the model learning power [32]. To address this challenge, researchers have focussed on the narrow architectural design approach towards achieving thinner and wider architectures. For example, the wide ResNet model explored the model width by introducing a factor k that controls the model width, providing improved performances compared to the residual networks [212]. Furthermore, the Pyramidal networks provided another architectural performance improvement compared to the depth-based methods, where the model width is extended per the residual units, thereby increasing the dimensions of the channel instead of the downsampling method used by the depth-based architectures [214].

The other width-based architectural improvement includes the Xception model, which uses depth-wise separable convolutions, decoupling the spatial and feature-map correlation and improving computational efficiency [215]. At the same time, the ResNeXt introduced cardinality, an additional dimension that describes the size of transformation used to split, transform and merge the model layers [216]. Perhaps, the newer architectural design outlined as the various versions of the original Inception and ResNet models were proposed to minimise the limitations of the original architectures, especially the computational burdens with Inception-v3, Inception-v4 and Inception-ResNet architectures [217], [218], while the Pyramidal network enhanced model generalisation [214], and the Xception and ResNeXt improved computational efficiency [215], [216].

3.11.1.3 Multi-path Exploitation

The multi-path model design, known as cross-layer connectivity, is another architectural design technique where a layer is systematically connected to another by skipping one or more intermediate layers, thereby creating a specialised path to the flow of information across layers. This technique was exploited in the design of Highway networks [211], DenseNet [219], ResNet [33] and dual-path networks, which use higher-order recurrent neural networks (HORNN) [220]. The multi-path design approach has become a dominant design approach in the architectural design of CNN. The cross-layer connectivity approach was inspired by the design of long short-term memory (LSTM) models, where gating units help decide the parameters that flow across layers. Furthermore, it partitions models into block units, which tries to resolve the vanishing gradient challenges of deeper architectures by making the gradients accessible in lower layers.

Moreover, the different methods used to establish cross-layer connectivity, often referred to as shortcut connections, include using skip connections [33], sub-sampling or zero-padding [211], a direct connection between layers using dense blocks [219], dual path networks [220], identity-mapping [221] and the $1\ x\ 1$ connection methods[32].

3.11.1.4 Feature- Map Exploitation

The model feature maps are essential components of every learning model since the models create features that represent the entire population during training and use these features to map the targets to the true labels in supervised learning models. Deep architectures have become a vast research area as deep learning models use the layered approach to identify data patterns, often called hierarchical patterns. A notable characteristic of the CNN models is that they perform excellently in automatically extracting discriminating features based on the tasks [28]. However, researchers suggest that not all feature maps support object discrimination in reality [222]; with enormous feature sets, model overfitting is likely due to noise. Therefore, selecting features is crucial for the model design to ensure that only supportive features are selected during training while the other features are dropped out.

3.11.1.5 Attention Exploitation

Attention networks are convolutional networks that focus on specific tasks at different time steps. The benefits of such mechanisms have been explored for image understanding, localisation and sequencing models. These models use an attention mechanism to obtain higher accuracy by finding global dependencies between data points without considering the distance between the input and output sequence, often referred to as the transformer [223]. The model architecture uses a gating function like the sigmoid and softmax units alongside a sequential model like the encoder-decoder structure to map a query and a key-value pair to an output where the queries, keys, values and outputs are all vectors [222]. The transformer network allows for increased parallelisation and achieves state-of-the-art performance in translation quality.

In summary, each architectural design approach offers some form of model improvement in the depth, width, space, paths, feature mapping, attention and gating, among others. However, none of the methods has achieved state-of-the-art performance on computational cost, accuracy, generalisation, etc. These suggest that providing the balance between these performance parameters is the most vital decision of the model designer. These decisions guide the design of the CNN architecture used in the research alongside other heuristics from the experiments.

3.11.2 Components of the Convolutional Neural Networks

The components of the CNN include the convolutional layer, activation, pooling, batch normalisation, dropout, and fully-connected layers alongside the inputs and output layers that make up the CNN architecture. Moreover, it is essential to highlight that all the outlined components of the convolutional neural networks are not present in every architecture since the designer chooses the layers to include based on the proposed model improvement. The convolutional layers learn the feature representations of the inputs using filters, usually referred to as kernels, used to compute the feature maps, with each neuron directly connected to its neighbouring neuron. The new features are obtained by convolving the input image and the kernel to get the set of features that are further propagated in the model. These components of the convolutional neural networks are as follows.

3.11.2.1 Convolutional Layer

The convolutional layer consists of several kernels used to learn and compute feature maps from a given input. A fundamental property of the convolutional layers is that the weights are shared, providing the advantages of reduced model complexity and ease of training the network by reducing the amount of time to learn model parameters due to the use of the receptive field of the NN and kernel size [224]. The individual neurons of the feature map are mapped to a region of neighbouring neurons in the preceding layers. Some new feature maps are obtained by convolving the inputs with a learned kernel and applying some element-wise activation function on the output.

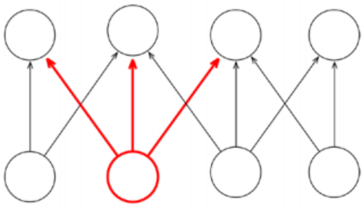


Figure 14 The convolutional layer showing sparse interactions

Besides, the kernel is shared by all the inputs' spatial locations to obtain these feature maps, and these features are computed mathematically. The feature value of a specific area (i, j) in the kth feature map of the lth layer is given by

$$Z_{i,j,k}^{l} = {w_k^l}^T x_{i,j}^l + b_k^l \qquad 3.14$$

Where $x_{i,j}^{l}$ is the input patch centred at location (i, j) of the lth layer, with w_k^l and b_k^l representing the respective weight vector and bias terms of the kth filter of the lth layer. There are numerous techniques for performing convolution operations, including tiled convolution, dilated convolution, transposed convolution [115], etc. The tiled convolution approach learns separate kernels within the same layer, while complex invariance is learned implicitly using square-root pooling over neighbouring units. The tiled convolution has a convolutional operation applied after every k-units where k represents the tile size that controls the distance of weight sharing [225]. Furthermore, the transposed convolution, often called deconvolution, is the opposite of vanilla convolution, which associates multiple input activations to a single activation. The deconvolution connects single input activation with numerous activations. It first upsamples the inputs by the padding and stride values before convolving the upsampled versions. The stride factor also provides a dilation factor for the input feature maps [226]. The deconvolution operation has been helpful in model visualisation [202], [226].

Nevertheless, the dilated convolution uses model hyperparameters in the convolutional layers to learn input patterns. For example, it uses zero padding between filter elements to increase the receptive field size of the model to capture more relevant information[227]. Besides, other compounded convolutional operations are used in the CNN architecture design, including the network architecture network alongside the inception module [210].

The CNN has the equivariant capability of the convolution operation, which assures that changes in input cause the same changes in the output, thereby making the operation translation invariant. In addition, the sparse interactions property also ensures that the memory storage space is maximised by using smaller kernels than the input data, guaranteeing that smaller data is available for storage, thereby improving the computation efficiency and reducing the convolutional computation cost. However, the convolution property does not guarantee invariance to all transformations, including changes in the image scale, rotation of an image and many others. However, an identified challenge with the convolution operation is the high computation cost, especially for very deep architectures, thereby consuming lots of memory resources. The pooling layers resolve this limitation of the convolutional layers by down-sampling the inputs.

3.11.2.2 Pooling Layers

Pooling reduces the computational cost between the model's convolutional layers [160]. It simplifies the layer output by non-linear downsampling, thereby reducing the number of parameters the model learns, building translation invariance and robustness to slight distortions by computing a max or average of the filter responses within the pool. The primary function of this layer is to reduce the spatial resolution of the feature map's size of the input. It is applied over space, scale and space alongside similar feature types [201]. These layers drop the CNN's computational burden by reducing the connections between respective convolutional layers, thereby reducing their size. Pooling operations generally introduce translation invariance in the images, making the models classify objects regardless of their position within the image. It is a vector-to-scalar transformation which operates on every local region in a photo by computing mostly the maximum or average of the pixels within an area and discarding the other remaining features.

There are different types of pooling layers, including maxpooling, average pooling, mixed pooling, l_p, Stochastic pooling, spatial pyramid pooling, spectral pooling, and multi-stage order-less pooling. The most common pooling techniques include max pooling, which computes the maximum value from a pooled region, and average pooling, which calculates the average value over a pooled area.

The ℓ_p pooling is a biologically inspired operation modelled after the average and max pooling techniques. The ℓ_p pooling is given by the ℓ_p norm of the pooling inputs obtainable using [228]

$$(|x_{Ii}|^p + \cdots + |x_{Il}|^p)^{1/p} \tag{3.15}$$

Where $x_{Ii\,,\cdots,\,x_{Il}}$ are the input nodes in the pool, with the value of p lying between 1 and ∞. However, the authors outlined that for $p \rightarrow \infty$, the ℓ_p pooling reduces to the ordinary max-pooling while for $p = 1$, it becomes the average pooling. Consequently, another pooling method named mixed pooling is a compound pooling technique that is similar to the ℓ_p pooling in terms of the form. The mixed pooling assigns a random value of 1 or 0 to parameter λ during the forward pass of the training. It uses the assigned number during backpropagation, indicating the choice of average or max-pooling. The mixed pooling is given by [229]

$$y_{kij} = \lambda \cdot max_{(p,q)\in\mathcal{R}_{ij}}\ x_{kpq} + (1-\lambda) \cdot \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} \tag{3.16}$$

Where λ is a random parameter, $y_{kij=}$ the output of the pooling operator for the kth feature map, x_{kpq} = elements at points (p, q) within the local pooling region, and $\mathcal{R}_{ij}$ = pool size.

Furthermore, stochastic pooling is another pooling approach inspired by dropout regularisation. The stochastic pooling selects the pool feature map response by sampling from a multinomial distribution obtained from the respective pooling region and randomly picks an activation that ensures that only non-maximal feature maps are selected. The stochastic pooling methods reduce the risk of model overfitting due to the stochastic nature [230]. Other pooling techniques include spatial pyramid pooling, which generates fixed-length representations regardless of the input sizes [231]. Spectral pooling uses frequency domain components of an image to generate feature maps [232] and multiscale order-less pooling, which considers local and global features of the inputs independently and uses it to provide feature maps for the model [233].

3.11.2.3 Fully Connected Layers

The fully-connected layer is the layer preceding the output and is often used as the output layer of some CNN models [210]. It is a global operation and valuable at the final layer in most CNN model designs for combining the non-linear selected features in the classification pipeline. The fully-connected layer is composed of a vector of k dimensions where the $k -$ parameter is the number of classes the model can predict. These classes of $k -$vector contain the probabilities of the respective predicted classes of the images under investigation. These fully connected layers depicted in Figure 15 are used to perform deductive reasoning from the learned features obtained using the convolutional and pooling layers.

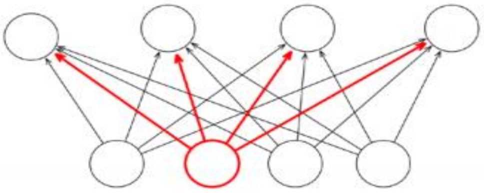


Figure 15 The fully-connected layers showing the typical neural connections

Moreover, the fully connected have much more connections than the convolutional layers. It takes a vector of arbitrary real-valued scores and squashes it into another vector, whose values will range from 0 - 1, with the sum equal to one for a softmax. These values represent the softmax activation output and the models' predictions. They achieve the needed high-level reasoning for the neural network [157], [158], [202]. After constructing the feature map hierarchy of the network, the model is fine-tuned, and the final layers are added such that each output neuron produces a conditional probability that maps the input image to a specific class with a non-linear function applied to the output. The output layer is the classification layer that usually contains a function that predicts the output. This function can be either the sigmoid or softmax function for a typical image-based application.

2.11.2.4 Activation Functions

Activation functions are functions that help the models to approximate any other functions or behaviour. Theoretically, a two-layer neural network can approximate any other function provided it contains a sufficient number of hidden units to achieve that. The position of the activation function in an architecture determines its function in the architecture. The most crucial work of these functions includes making decisions during the intricate pattern learning process alongside accelerating the learning process in the hidden layers. It achieves this by adding non-linearity to the learned features by further propagation[234]. The activations work by modifying the output of a feature map, given as

$$f_l^k = \alpha \left(O_l^k\right) \qquad \textbf{3.17}$$

where O_l^k is the output of a convolution layer, α is the activation function, and f_l^k is the transformed output. The literature outlines the numerous developed activation functions, including rectified linear units (ReLU), Sigmoid, HardSigmoid, Swish, Hyperbolic tangent, Softmax, Softplus, Maxout, and their variants, among other functions not mentioned. A worthy note is that the ReLU is the most used activation function in the hidden layers of the deeper architectures [234]. A comprehensive review of the various activation functions in deep learning research is found in the literature [234].

Moreover, a necessary condition for training the gradient-based models is that the activations are continuously differentiable, allowing the gradients to be computed, thus obtaining the parameters that minimise the loss function. If this condition is not satisfied, the gradient-based methods cannot succeed. These definite range helps to achieve more stable performance when using gradient-based techniques than the infinite range functions. Other significant factors in choosing an activation layer for deeper architectures include smooth, symmetric, behaving like identity functions around the origin [235]. An overview of the most crucial activation functions is outlined as follows.

a) Sigmoid Function

The Sigmoid is a non-linear AF used in neural networks to convert discrete signals to continuous signals, often referred to as a logistic function or squashing function in the literature [8]. The Sigmoid is a bounded differentiable function defined for real input values, with positive derivatives everywhere and a degree of smoothness [16]. The most vital property of the Sigmoid is that it produces numbers very close to 1 for large positive numbers, numbers very close to zero for large negative numbers, and numbers close to zero output numbers very close to 0.5. The Sigmoid is given by

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.18}$$

It often appears at most DL architectures' output layers and helps predict probabilistic outputs. The Sigmoid has been successfully applied to classification, logistic regression, and other domains, with researchers highlighting the most significant advantage as being easy to understand [17]. Moreover, researchers have suggested that the Sigmoid should be avoided when initializing neural networks, generally from small random weights [12]. Other crucial limitations of the Sigmoid include the sharp damp gradients during backpropagation from deeper hidden layers to the input layers, slow convergence, gradient saturation, and non-zero-centred output that causes the gradient updates to propagate in different directions. Nevertheless, newer activations have been investigated and proposed to manage these drawbacks suffered by the Sigmoid functions, including the HardSigmoid [236], Sigmoid weighted linear units (SiLU) and the derivative of the Sigmoid weighted linear units (dSiLU) [237], and the logistic Sigmoid [238] etc.

The HardSigmoid is a variant of the Sigmoid that offers improved computational cost compared to the original Sigmoid and finds practical applications in deep learning classification [236]. The HardSigmoid is obtainable using the function.

$$\boldsymbol{f(x) = clip\left(\frac{x+1}{2}, 0, 1\right) = \max\left(0, \min\left(1, \left(\frac{x+1}{2}\right)\right)\right)} \tag{3.19}$$

Conversely, Sigmoid's SiLU and dSiLU variants are reinforcement learning-based approximation functions. In the case of state-value-based learning, state vector $\boldsymbol{s}$, an RBM approximates to state-value energy function $\boldsymbol{V}$ by the expected negative energy of an RBM network. The activation $\boldsymbol{\alpha_k}$ of the $\boldsymbol{k-th}$ SiLU for inputs $\boldsymbol{z_k}$is obtained as the multiple of the input and the Sigmoid given by [237]

$$\boldsymbol{\alpha_k(z_k) = \ z_k\, \alpha\, (z_k)} \qquad \textbf{3.20}$$

Where $\boldsymbol{\alpha}$ is a Sigmoid function. However, the authors outlined that the substantial values of $\boldsymbol{z_k}$, the SiLU approximates to a ReLU function. In contrast, the dSiLU resembles an overshooting Sigmoid, comparable to the Sigmoid function and computed.

$$\boldsymbol{\alpha_k(z_k) = \ \alpha\, (z_k)\Big(1 + \ z_k\, \big(1 - \ \alpha(z_k)\big)\Big)} \qquad \textbf{3.21}$$

The dSiLU has a minimum value of $\boldsymbol{-0.1}$ and a maximum value of $\boldsymbol{1.1}$ for $\boldsymbol{z_k} \ \approx \ \boldsymbol{\pm 2.4}$. The typical response of the SiLU and dSiLU functions is shown in Figure 16.

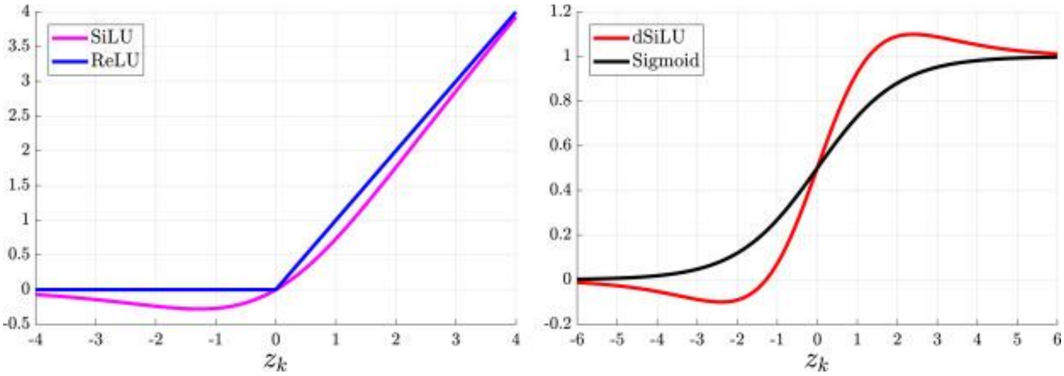


Figure 16 Test response of the SiLU and dSiLU function [237]

Besides, the logistic Sigmoid units (LSigmoid) is another activation proposed for the recurrent neural networks that address the problem of ReLU and LReLU in DBN applications [238]. These functions are not able to maximise the pre-training effects of RBNs. The LSigmoid addresses the vanishing gradients during backpropagation by combining the benefits of unsaturation from the leaky ReLU function. The LSigmoid is given by

$$\boldsymbol{f_I\ (x) = \begin{cases} \alpha(x - b) + \ Sigmoid(b)\ , & x \ \geq \ b \\ Sigmoid\ (x)\ \ , & -b\ <\ x\ <\ b \\ \alpha(x + \ b) + \ Sigmoid(b)\ , & x \ \leq -b \end{cases}} \qquad \textbf{3.22}$$

Where b = threshold and α = slope. Both of these parameters are usually preset. Finally, the Sigmoid is primarily helpful in binary classification problems; as such, it cannot work in multi-class problems, requiring other functions that can manage more than two class inputs. A function for multi-class prediction is the softmax function.

b) Softmax Function

The Softmax is a valuable function for multi-class neural computing that returns the probabilities of each class and the target class having the highest probability. It helps compute the probability distribution from a vectorof real numbers. The output of the Softmax function ranges between 0 and 1, with the sum of the probabilities equal to 1. The Softmax is obtained using the following relationship [68]

$$f(x_i) = \frac{e^{(x_i)}}{\sum_j e^{(x_j)}} = \frac{e^{(x_i)}}{e^{(x_1)} + e^{(x_2)} + \dots + e^{(x_n)}} \qquad \textbf{3.23}$$

Where j is a linear function of scores for values ranging from 1 to n. The softmax function appears at the output layer of almost all the major deep convolutional neural network architectures.

c) Softsign

Softsign is one of the earliest activation functions that has found application in deep learning research. It is a quadratic polynomial function that converges in polynomial form, and it differs from $tanh$ function, which converges exponentially. The Softsign is given by [239]

$$f(x) = \left(\frac{x}{|x| + 1}\right) \qquad \textbf{3.24}$$

Softsign is mainly applied in regression applications but has recently been useful in deep learning modelling of Text-to-Speech systems showing promising results [240].

d) Softplus

The Softplus is another activation function and a primitive of the Sigmoid. It can be viewed as a smoothened version of the ReLU activation, which has a non-zero gradient and smoothing properties, thereby stabilising the performance of the models [241]. The Softplus is given by

$$f(x) = \log(1 + e^x) \qquad \textbf{3.25}$$

Besides, comparing the Softplus function against the Sigmoid and ReLU functions suggests that the Softplus produced faster convergence with lesser epochs during training and finds practical application in speech recognition systems, among other applications [242].

e) Rectified Linear Units (ReLU)

The ReLU is a fast learning activation function used in almost all the existing deep learning architectures[28]. It is the most widely used function in deep learning research [234] due to its simplicity and reliability in deeper architectures [243]. It offers better performance and generalization in deep architectures than other activations [244], [245]. The ReLU represents a nearly linear function that preserves the properties of linear models, making them easy to optimise using gradient-descent techniques [68]. It performs a threshold operation on each input element where values are less than zero, setting them to zero. The ReLU is given by [243]

$$f(x) = \max(0.x) = \begin{cases} x_i\,, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \qquad \textbf{3.26}$$

The ReLU activation cuts off the values of the inputs less than zero, thereby forcing them to zero; however, it suffers from the vanishing gradient challenge. The ReLU and its variants have found significant application in different deep architectures, including therestricted Boltzmann machines [243] and the CNN architectures[33], [157], [159]. In the majority of the existing architectures, the ReLU function has been helpful in the hidden layer and another activation at the output layers of the network, especially in object recognition[33], [246], and speech recognition tasks [247]. The ReLU improves model computation speed since it does not perform exponentials and divisions [203] and introduces sparsity in the hidden layers as it squishes the values in the range of zero and maximum. Nevertheless, the ReLU function, like other activations, has some limitations, including being prone to overfitting. Researchers explored using the dropout technique to reduce the effects of overfitting, improving the performance of ReLU activation in very deep architectures [248]. Besides, the ReLU is sometimes fragile during training which causes some gradients todie, giving zero activation [68] and causing the weight updates not to activate future data points, hindering the learning. These challenges of the ReLU function were addressed by the newer variants of the ReLU, including the leaky ReLU that specifically addresses the dead neuron issues. The variants include the parametric ReLU [33], leaky ReLU [247], randomised ReLU [249], and displaced ReLU [250], among other variants.

Furthermore, the leaky ReLU (LReLU) is a variant of ReLU that introduced a slight negative slope to the original ReLU function, parameterised as α, to keep and sustain the weight updates during the entire propagation process [247]. The α parameter resolves the ReLUs dead neuron problems using a minimal constant value for the negative gradient in the range of 0.01. Therefore there will be no zero gradients at any point in time during model training. The LReLU is computed as follows.

$$f(x) = \alpha x + x = \begin{cases} x, & if\ x > 0 \\ \alpha x, & if\ x \leq 0 \end{cases} \qquad \textbf{3.27}$$

Nevertheless, the LReLU has identical results to the standard ReLU except for the non-zero gradient throughout the training process. Nevertheless, the parametric rectified linear units (PReLU) is another improved variant of the ReLU function where the negative part of the original ReLU is adaptively learned while the positive linear part is maintained. The PReLU is given [33]

$$f(x_i) = \begin{cases} x\ , & if\ x_i > 0 \\ \alpha_i x\ , & if\ x_i \leq 0 \end{cases} \qquad \textbf{3.28}$$

Where α_i is the learned negative slope control parameters. However, when the α_i parameter is zero; the PReLU becomes the same as the original ReLU. Moreover, another modification is the randomised leaky rectified linear units (RLReLU), which is a dynamic variant of leaky ReLU where α_{ji} is a random number sampled from a uniform distribution $U\,(l, u)$ and used to train the network. The RLReLU is given by

$$f(x_i) = \begin{cases} x_{ji}\ , & if\ x_{ji} > 0 \\ \alpha_{ji} x_{ji}\ , & if\ x_{ji} \leq 0 \end{cases} \qquad \textbf{3.29}$$

Where the $\alpha_i \sim U(l,u), l < u \; and \; l,u \in [0,1]$. Besides, the test phase averages all α_{ji} during the training without the dropout. The α_{ji} parameter is obtained as $\alpha_{ji} = \frac{l+u}{2}$. The test output is given by

$$y_{ji} = \frac{x_{ji}}{\left(\frac{l+u}{2}\right)} \qquad \textbf{3.30}$$

However, comparing the ReLU and some variants have been investigated on crucial classification datasets. As a result, researchers validate that the LReLU, RLReLU, and PReLU perform better than the ReLU on classification problems [249]. Yet, the ReLU remains the dominant function in the state-of-the-art architectures used in deep learning research.

f) Exponential Linear Units (ELU)

The ELU is another activation function developed to speed up the training of deep neural networks [251]. The ELU provides the crucial advantage of alleviating tle vanishing gradient problems of the ReLU function by using identity for positivevalues, thereby improving the learning characteristics. The negative values also push the mean unit activation closer to zero, improving learning speed and reducing the computational burden. The ELU is a reasonable alternative to ReLU as it reduces the bias shifts by shifting the mean activations towards zero during training. The ELU is given by

$$f(x) = \begin{cases} x \quad , & if \;\; x \; > \; 0 \\ \alpha \, e^{x} - 1, & if \;\; x \; \leq \; 0 \end{cases} \qquad \textbf{3.31}$$

Where $\alpha =$ hyperparameter that controls the saturation point for negative inputs. Besides, research suggests that the performance of ELU is significantly comparable to the ReLU and LReLU and even better in learning faster as well as generalisation [252]. However, an identified limitation of the ELU function is that it does not centre the values at zero. This limitation is addressed by the newer ELU variants, including the parametric ELU [251] and Scaled ELU [253] variants.

Nonetheless, the parametric exponential linear units (PELU) were proposed by [251] to address the zero-centring of values limitation of the ELU. It achieves this by reducing the bias shifts with additional parameters to control the gradient flow. The PELU function is given by

$$f(x) = \begin{cases} cx \quad , & if \;\; x \; > \; 0 \\ \alpha \, e^{\left(\frac{x}{b}\right)} - 1, & if \;\; x \; \leq \; 0 \end{cases} \qquad \textbf{3.32}$$

Where $\alpha, b, c > 0$. α controls the negative quadrant saturation, b controls the exponential decay scale, and c controls the changes in the positive quadrant slope [251]. The PELU is most useful in applications that require fewer bias shifts and vanishing gradients, like convolutional neural networks.

Conversely, the scaled exponential linear units (SELU) are another ELU variant introducing self-normalisation. The SELU is a scale multiple of the original ELU function with approximately zero mean and unit variance. Also, it converges towards unit variance and zero mean when propagated through multi-layers during training. The SELU enables a strong regularisation that allows robust feature learning in deep neural networks. The SELU is given by

$$f(x) = \tau \begin{pmatrix} x & , & if \ x > 0 \\ \alpha e^{x} - \alpha, & & if \ x \leq 0 \end{pmatrix} \qquad \textbf{3.33}$$

Where $\tau =$ scale factor, which ensures that the slope is large than one and α is a hyperparameter that controls the saturation of inputs. Moreover, the SELUs have positive and negative values for regulating the mean, a slope and are not affected by the vanishing gradient challenges as the ReLU and cannot be derived from other activations including the ReLU, scaled ReLU, Sigmoid, LReLU, among others [253].

g) Maxout

The Maxout function is an activation that generalises the ReLU and leaky ReLU, where the neurons inherit the properties of the ReLU and LReLU to avoid saturation and dying neurons during training. The Maxout has a non-linearity applied as a dot product of the data and the neural network's weights. The Maxout is given by [254]

$$f(x) = \max(w_1^T x + b_1, \quad w_2^T x + b_2) \qquad \textbf{3.34}$$

Where $b =$ biases, $w =$ weights, $T =$ transpose. However, the limitation of the Maxout is the computationally intensive nature of the function, as it doubles the number of parameters used in all neurons, increasing the number of parameters.

h) Swish

The Swish function is a compound function that combines the Sigmoid and the input to provide the hybrid Swish function. The Swish uses a reinforcement learning-based automatic search technique to achieve the activation, with better smoothness, non-monotonic and unbounded at the upper and bounded in the lower limits, producing better optimisation and generalisation. The Swish is derived as

$$f(x) = x \, .sigmoid(x) = \frac{x}{1 + e^{-x}} \qquad \textbf{3.35}$$

Perhaps the Swish's advantage is its simplicity and improved accuracy as it avoids the vanishing gradient while learning rich features. In addition, the authors outlined that Swish outperformed the ReLU on deep learning classification tasks.

Finally, activation function research has evolved significantly over the years. The more recent functions combine other existing functions to improve performance, including Softplus, Swish, dSILU, Maxout and PELU. Besides, the parameter learning functions have also become the new trend with PELU and PReLU functions. These AFs help to learn higher-order polynomials for deeper architectures, with their function in the architectures dependent on the position of the AFs in the respective architectures.

The specific roles of these activations differ, with researchers exploring the balance in the network's width, depth, and resolution to obtain improved performance. Moreover, the most vital activations outlined in the literature include the ReLU, Softmax, Sigmoid, Swish and SiLU, which have witnessed significant applications in CNN architectures, alongside the ReLU and Sigmoid having other applications in RNN and SAE architectures. Besides, the most used activation in the deep architectural designs is the ReLU which has witnessed applications in almost all the ImageNet winning CNN architectures, except the EfficientNet architecture, which adapted the SiLU and Swish activations.

The output layer has the activation helpful in making model predictions, including classification problems where the softmax, sigmoid, and SiLU functions have been previously used at the output of the deep CNN architecture [212], [255], [256]. The hidden layers also have different activations used in the design to ensure that the signals propagate to the output layer, especially the ReLU and Swish functions, which have been the dominant function in the deeper architectures.

Moreover, choosing the appropriate activation function for a given deep architecture requires the heuristic testing of the existing activations on the architectures and observing the performance of the models during training. The process ensures that the most efficient activation for specific architecture is easily selected and deployed in the desired deep learning applications.

3.11.3 Loss Functions

The loss functions are estimators that measure how well a model can predict the desired outcome. It uses the maximum likelihood framework as a helpful method to derive the best set of weights and performs well in optimising the weights. It uses a candidate solution that smoothly maps to a high-dimensional landscape to update the model weights iteratively. The loss function accepts the ground truth and the model's predictions and evaluates how well the model predicts the outcome. A higher value of loss means that the model performed poorly, and a low value implies that the model performed well during training. Selecting the appropriate loss function for a model is crucial in successfully training the model, with each loss function having different properties and capabilities. These properties define how the model learns, especially in managing outliers.

Moreover, deep learning theory generally adapts and uses statistical approaches to solve learning tasks. These tasks include training and generalisation of model performance of different applications. The overall foundational concepts of parameter estimation, bias and variance are handy for characterising the performance of a model, including overfitting, underfitting and generalisation [68].

Moreover, point estimates provide a single best prediction of the parameter of interest, which can be either a vector of parameters or a single parameter [68]. To distinguish a set of parameters from their actual values, the convention of a given point estimate θ is denoted by $\breve{\theta}$. These assumptions would differentiate a predicted values from the true values. The point estimators help determine the relationship between some inputs and the target variables. These estimators, often called function estimators or function approximators, help to predict a variable y from a set of input vectors x. For a function $f(x)$ that defines the relationship between x and y, it can be described as

$$y = f(x) + \varepsilon \qquad \textbf{3.36}$$

Where ε represents the part of y that is not predictable from x. The function estimation involves approximating f with a model. These approximations are typical in modelling even more extensive problems, including deep learning-based problems. The estimates often involve mapping a function from x to y or estimating a model parameter [68]. However, the vital sources of estimation error in results outlined by researchers include biases and variance. While the bias measures the deviation from the actual value, the variance measures the deviation from the expected estimator value that any data sampling can cause. Hence, if the true value obtained from training a model is y, the model's loss is the difference between the prediction and the actual value. This simple loss L becomes

$$L = f(x) - y \qquad \textbf{3.37}$$

Where x represents the inputs, therefore, to obtain the loss L over n items in a dataset, the average of all losses is

$$L = \frac{1}{n}\sum_{i=1}^{n} f(x^i) - y^i \qquad \textbf{3.38}$$

Besides, the above loss functions are adapted to various deep learning applications and grouped into three categories: regression, multiclass, and binary classification. These categories are briefly discussed as follows.

3.11.3.1 Regression Loss Functions

The regression application involves the modelling and prediction of real-value quantities. The loss function capable of predicting real-valued inputs includes the mean square error (MSE) and mean absolute error (MAE) and their variants. The mean square error is one of the most straightforward loss functions, and it models the average squared difference between the model's predicted output and the ground truth. The MSE helps predict continuous data when the outputs are numerical predictions, and it is obtained as follows

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \textbf{3.39}$$

Where n = model number of samples. The square functions ensure that the obtained outputs are always positive. Besides, the MSE offers the advantage that the model has few outlier predictions with huge errors since the MSE increases the error due to the square function. However, the major limitation of using the MSE is that if the model makes a slight mistake, the error is amplified by the squaring function, making the MSE unsuitable for models with many outliers.

Conversely, the mean absolute error is the average absolute difference between the model's predicted output and the ground truth. The MAE can never be a negative value, and it is given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad 3.40$$

Moreover, the main advantage of the MAE is that all the referenced errors are compiled on the same linear scale to produce the desired outputs. On the other hand, the MAE's limitation is that it ignores outliers instead of managing them, making the models prone to very poor decisions in some significant cases and not ideal for crucial applications.

Furthermore, the Huber loss function is another regression loss function that offers some common advantages over the MAE and MSE losses. It manages the outliers by balancing the MAE and MSE using a range of values that keep the loss for minor errors as a quadratic function and linear otherwise. The relationship gives the Huber loss.

$$L_\delta\big(y, f(x)\big) = \begin{cases} \frac{1}{2}\big(y - f(x)\big)^2 & , \quad for \ |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 \, , & otherwise \end{cases} \qquad 3.41$$

Where $L, y, and\ x$ represents the respective loss, actual outputs and inputs, δ is the delta parameter. The Huber loss is ideal when considering model outliers whose effects are negligible [68].

3.11.3.2 Binary Classification Loss Functions

The binary cross-entropy loss is the default function used for binary classification problems. It helps predict the probability output of a neural network with a single unit output layer. The binary cross-entropy is often referred to as log loss, logistic loss, and logarithmic loss in other literature [29], [68], [257]. The binary cross-entropy loss calculates a weighted sum of the feature array and bias, with the outputs logits produced using the sigmoid function.

$$\check{p} = \alpha(x^T\ \theta) \qquad 3.42$$

Where α is a sigmoid function that ensures the output takes either of the two states of 0 and 1. The Sigmoid function is described in detail in Section 3.11.2.4. Besides, as soon as the model estimates the probability of an instance belonging to a positive class, it predicts the class using the following rules;

$$\check{y} = \begin{cases} 0, & if\ \check{p}\ < 0.5 \\ 1, & if\ \check{p}\ \geq 0.5 \end{cases} \qquad 3.43$$

The binary cross-entropy loss is obtained by computing the function

$$l = -\sum_{i=1}^{n} y_i \log f(x_i, \theta) + (1 - y_i) \log(1 - f(x_i, \theta)) \qquad 3.44$$

Where l is the obtained model loss, the negative sign converts the overall computation to a positive quantity.

Moreover, the Hinge loss is another binary loss function used primarily on support vector machine classifiers whose labels are encoded as $1\ and - 1$. The Hinge loss is obtained as follows [258].

$$l(y) = \max{(0, 1 - t\,.\,y)} \qquad \textbf{3.45}$$

Where $y =$ raw output and $t =$ intended output given as $\pm$ 1. The Hinge loss tries to simplify the SVM algorithm by maximising the loss and has the squared version named Squared Hinge Loss, which computes the square of the obtained Hinge losses for a model. The squared Hinge loss is given by

$$l(y) = \sum_{i=0}^{n} (\max(0, 1 - \; y_i\,.\,\hat{y}_i)^2) \qquad \textbf{3.46}$$

However, the binary loss functions have the limitation that they can only support the binary classification, making them unsuitable for multiclass classification problems, which require the model's output to predict more than two outputs at a time.

3.11.3.3 Multiclass Classification Loss Functions

The multiclass problems are classification tasks modelled to predict that specific samples belong to one of more than two classes. They are modelled to predict the likelihood of the example belonging to each class. The categorical cross-entropy loss function is for multiclass applications that eliminate the class limitations of binary cross-entropy functions. It uses the softmax loss function to generalise and perform multi-class classification effectively. Besides, the categorical cross-entropy is a multiclass loss function useful when the target class are text-based labels [29]. For each instance, the model estimates the sample's probability of belonging to the class. The softmax loss predicts the class with the highest probability as the output, as described in Section 3.11.2.4. The cross-entropy loss is given by

$$L(X_i Y_i) = \; - \sum_{j=1}^{n} y_{ij} * log(p_{ij}) \qquad \textbf{3.47}$$

Where Y_i = labels or target vector encoded as one-hot as $(y_{i1}, y_{i2} \dots\, y_{in})$ and $p_{ij} = \; f(X_i)$ which outlines the probability that the element is in the class j. Besides, it is worth outlining that the cross-entropy is inversely proportional to the total probability of an event; thus, higher cross-entropy implies a lower chance for an event occurrence. The cross-entropy can be re-written as follows

$$L = \; -(ylog(\hat{y}) \; + \; (1 - y)log(1 - \; \hat{y})) \qquad \textbf{3.48}$$

The negative sum of the maximum likelihood produces the cross-entropy in which a low cross-entropy means the model performs well. In contrast, a high cross-entropy implies that the model is performing poorly. For example, given some probabilities, the cross-entropy of a set of events is low if the events are more likely to happen and significant if the event does not occur.

However, a limitation of the categorical cross-entropy function is the considerable memory required to store variables when the number of inputs is significantly large. The sparse categorical cross-entropy addresses the encoding challenge of categorical cross-entropy by encoding the targets as integers labels [29] and computing the loss using the output index as ground truth.

Conversely, the Kullback Leibler (KL) divergence loss is another multiclass loss function that measures the differences between various models' probability distribution, with zero divergences indicating identical models [68]. Perhaps, the KL divergence for two distributions, A and B, is given by

$$D_{KL}(A||B) = \begin{cases} -\sum_{x} A(x).\log\frac{B(x)}{A(x)} = \sum_{x} A(x).\log\frac{B(x)}{A(x)} \ , \ \mathit{discrete\ distributions} \\ -\int A(x).\log\frac{B(x)}{A(x)}.dx = \int A(x).\log\frac{B(x)}{A(x)}.dx\,, \mathit{continuous\ distributio} \end{cases} \tag{3.49}$$

The limitation of the KL divergence is that it is an asymmetric function. Therefore, it is not usable in simple classification problems or for estimating distance metrics. However, it can approximate more complex functions like the variational autoencoders.

Overall, the loss functions use the maximum likelihood for finding the best statistical estimates of parameters from training data. The interpretation of the maximum likelihood estimate is similar to minimising the dissimilarity between the empirical distribution described by the training set and the model probability distribution using KL divergence. The results correspond to minimising the cross-entropy of the distributions.

3.11.4 Optimisation and Optimisation Functions

Optimisation is an iterative process of comparing different model solutions to obtain satisfactory performance. It is helpful to evaluate a candidate's solution by either minimising or maximising an objective function by altering the values of the functions to get the solution with the lowest or highest scores, respectively [68]. The loss, error, and cost functions are different terms used to refer to the objective functions in literature. The standard notation to denote values that maximise or minimise a function is often outlined with a $superscript^*$ that is for minimisation, the function becomes

$$x^* = \arg\min f(x) \tag{3.50}$$

Where $x =$ input. Similarly, the maximisation is the same with the max function used in place of min

$$x^* = \arg\max f(x) \tag{3.51}$$

However, the optimisation of a function can produce several local minimums depending on the function's parameters. Still, it can only have one global minimum, which is the point that gives the absolute lowest value on the parameters. The obtained cost function reduces all the bad and good aspects of the complex model to a single scalar number that allows for the ranking of the candidate solution.

Furthermore, the optimisation techniques help improve the speed and memory performance of learning algorithms [259] and represent one of the numerous ways to improve deep learning models' performance, as outlined in the literature [260]. Perhaps, there are multiple methods of optimising models, especially for deep learning applications, including gradient, weight initialisation, data augmentation, batch normalisation, and shortcut connections [160]. A summary of these techniques is presented as follows.

a) Gradient Optimisation

The gradient is the rate at which change occurs over time. It measures the change in the output rate for a little change in the inputs. On the other hand, gradient descent (GD) outlines whether a function is decreasing or increasing at a particular point. It is an optimisation method that addresses specific challenges, especially when the gradient of a loss function with respect to each parameter helps to obtain the optimal direction to adjust the model parameters. The rate of this movement is determined when the learning rate is usually denoted by α. The learning rate is fixed to avoid changing too fast, causing overfitting or the model being too slow to converge. The gradient descent algorithm is given by

$$\theta \xrightarrow{yields} \theta - \alpha \frac{\partial L}{\partial \theta} \qquad \mathbf{3.52}$$

Where θ = parameter, $\alpha =$ learning rate. The first-order gradient computation produces a tangential line on a given error surface. These gradients are easy to compute and less time-consuming to converge, even on large datasets. Perhaps gradient propagation across very deep architectures during model training is challenging, justifying the need to modify the gradient to accept gradient updates. Several researchers have explored and continue to study techniques to resolve these challenges; the modifications to the gradient descent algorithm is active research, with numerous novel optimisation methods and algorithms proposed in recent time.

The batch gradient descent (BGD), also known as the vanilla gradient descent, is the default gradient descent algorithm that computes the gradient of a loss function with respect to each of the parameters of the entire training examples $\theta,$ before updating the model. The BGD is given by

$$\theta = \theta - \eta \, . \nabla_{\theta} \, J(\theta) \qquad \mathbf{3.53}$$

Where the $\nabla_{\theta} \, J(\theta) =$ the gradient term, $\eta =$ learning rate, $\theta =$ model parameters. However, a significant limitation of the BGD is that it is inherently slow and unsuitable for extensive training examples that cannot fit into the computer's memory and does not guarantee convergence at a global minimum for convex surfaces [261].

Conversely, stochastic gradient descent (SGD) was developed to address the limitation of BGD. The SGD resolves these limitations by randomly selecting the following training examples to update the trainable parameters, enhancing the speed. The SGD helps to take small steps in the direction of optimality, with the steps being stochastic and guaranteed to get to the minimum[160]. The SGD is given by

$$\theta_{t+1} = \theta_t - \eta_t \, \nabla_{\theta} \, L\big(\theta_t; \, x^{(t)}, y^{(t)}\big) \qquad \mathbf{3.54}$$

Where $x^{(t)} \, and \, y^{(t)}$ represents the selected examples, and the other parameters are the same as the BGD. The advantage brought by SGD is that instead of computing a gradient based on the aggregate across the entire dataset, the gradient is based on the data contained in a single batch and continues to update the loss, batch by batch, until the training is complete. However, a significant drawback of the SGD is that there is no adaptive way of obtaining the optimal learning rate. It also has the gradients tending to zero at some point during training, which is not ideal in very deep architectures and does not scale well on large datasets [262].

Moreover, the mini-batch gradient descent (MGD) optimisation aims to improve the training speed of deep architectures by performing an update after a mini-batch of training examples is passed. The MGD reduces the variance parameter updates, thereby enhancing convergence. The MGD is computed as

$$\theta \ = \ \theta - \ \eta \, . \nabla_{\theta} \left(\theta; \ x^{(i:i+n)}; \ y^{(i:i+n)}\right) \tag{3.55}$$

However, there are outlined limitations of the MGD, including that it does not guarantee excellent convergence and the difficulty in choosing an appropriate learning rate for the model, which is a parameter of the dataset used in training the model.

Furthermore, the stochastic gradient descent with momentum (SGDM) proposes to speed up the optimisation process based on the model dimensions. It involves accelerating the process by following the directions where the gradient is pointing while slowing the path where there is a sign of an inherent changing gradient. The SGDM is given by

$$v_{t+1} = \ \gamma v_t - \ \eta_t \, \nabla_{\theta^l} \left(\theta_t; \ x^{(t)}, y^{(t)}\right) ; \ \theta_{t+1} = \theta_t - \ v_{t+1} \tag{3.56}$$

Where $\gamma =$ momentum term, usually set to 0.9, $v_{t+1} =$ current velocity vector. However, the SGDM has a drawback because the learning rate is manually optimised, making it dependent on expert judgement.

Conversely, Nesterov's accelerated gradient (NAG) is another robust optimisation approach that provides better convergence than gradient descent-based optimisers[263]. The inspiration for the NAG algorithm is the Polyak classical momentum method of accelerating gradient descent that accumulates some velocity vectors in the direction of the continuous decreasing objective function[264]. The NAG algorithm is given by

$$v_{t+1} = \ \mu v_t - \ \epsilon \, \nabla f(\theta_t) \, ; \ \theta_{t+1} = \theta_t + \ v_{t+1} \tag{3.57}$$

Where $\epsilon =$ learning rate and it is always greater than zero, $\mu \ \in \ [0,1] =$ momentum coefficient, and $\nabla f(\theta_t) =$ the gradient at θ_t. The NAG updates are performed as follows

$$v_{t+1} = \ \gamma v_t - \ \epsilon \, \nabla f(\theta_t + \ \mu v_t) \, ; \ \theta_{t+1} = \theta_t - \ v_{t+1} \tag{3.58}$$

The NAG at first computes the known gradient θ_{t+1}, approximates the following steps by choosing an optimal step size and then moves in the direction of γv_t, which represents the past accumulated gradients, computes the current gradient and updates it accordingly. The NAG has a similar limitation to most other gradient-based optimisers, as the learning rate is manually fixed. However, the NAG optimisers inspired the development of the adaptive optimisers that have their learning rates as a learnable hyperparameter of the model.

The AdaGrad optimiser is an early adaptive optimiser with adaptive learning rates that update relative to the parameter updates during model training. The vital features of the AdaGrad include that it considers every model parameter when selecting the learning rates, thereby making it possible to increase or decrease learning rates depending on the model features and converges quicker than the gradient-based optimisers[265]. The AdaGrad modifies the learning rate η at every iteration of time step t for all parameters θ_i based on the past compiled gradients for θ_i. The AdaGrad update is computed as follows

$$g(t,i) = \nabla_\theta J\left(\theta_{t,i}\right);\ \Delta_{x_t} = -\frac{\eta}{\sqrt{\sum_{r=1}^{t} g_t^2}} .gt \qquad \textbf{3.59}$$

Where $g(t,i) =$ gradient of the loss function with respect to θ_i parameter at time step $t,\ \eta =$ global learning rate shared by all dimensions and the denominator gives the ι_2 norm of all past gradients on each dimension. Researchers identified the limitation of the AdaGrad as the continuous decay of the learning rate throughout the learning process [266].

Moreover, the aggressive reduction in AdaGrad was improved with AdaDelta. This new adaptive optimiser restricts the window of past gradients to a specific size denoted as w to update the learning rates. The AdaDelta uses the sum of the exponential decaying average of squared gradients to update the learning rate. The running average is given by

$$E[g^2]_t = \rho\, E[g^2]_{t-1} + (1-\rho)g_T^2 \qquad \textbf{3.60}$$

Where $\rho =$ decay constant. The update rule is given by

$$\Delta x_t = -\frac{RMS\ [\Delta x]_{t-1}}{RMS\ [g]_t} \qquad \textbf{3.61}$$

Where $t =$ time, $g =$ gradient, $\Delta = \ the$ sum of the numerator terms. The AdaDelta enables automatic learning rate fixing, lesser computation cost, robustness to noise due to large gradients and automatic hyperparameter tuning, making it easier to implement than the gradient descent optimisers [266].

Conversely, the root means square propagation (RMSProp) optimiser works similarly to AdaGrad but changes the gradient accumulation into a weighted moving average. The RMSProp modifies the learning rate into an exponentially decaying average of squared gradients given by [68], [267]

$$E[g^2] = E[g^2]_{t-1} + (1-\gamma)g_t^2\ ;\ \theta_{t-1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} .gt \qquad \textbf{3.62}$$

Where the $\eta =$ learning rate set to 0.001 and $\gamma = 0.9$

The RMSProp automatically adjusts the learning rate but has an identified limitation: it lacks the bias correction term, which causes large step sizes in practical applications, causing model divergence [267].

Besides, the Adam algorithm is another adaptive moment (Adam) optimisation technique that uses adaptive estimates of moments of lower-order degrees. Adam offers improved memory, computational efficiency and invariance to diagonal scaling of gradients and is suited for large-scale parameter optimisation. It combines the properties of AdaGrad and RMSProp to obtain the first and second-order moment estimates representing the uncentred variance and mean of the respective gradients. The gradient $\widehat{m_t}$ (mean) and squared gradient v_t (variance) is given by [268]

$$m_t = \frac{m_t}{1 - \beta_1^t} \;\; ; \;\; v_t = \frac{v_t}{1 - \beta_2^t} \qquad 3.63$$

Where $\beta_1, \beta_2 \in [0,1]$ are the hyperparameters that control the exponential decay rate of the moving averages. The gradient update is estimated directly from the moving averages of the first and second moments as

$$\theta_{t-1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \varepsilon}\widehat{m_t} \qquad 3.64$$

Where $\beta_1, \beta_2 = 0.9$ and 0.999 respectively and $\varepsilon = 10^{-8}$. The advantage of the Adam optimiser is that it converges fast and does not suffer from vanishing gradients. The AdaMax is a variant of Adam modelled using an infinity norm. It is sometimes superior to Adam in specific applications. The velocity parameter of the AdaMax scales the gradient inversely to the $\ell_2$norm of the past gradients v_t and current gradient $|g_t|^2$ terms. The gradient is given by

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\,|g_t|^2 \qquad 3.65$$

The AdaMax update rule is obtained as the maximum between the present and past gradients given by

$$\theta_{t-1} = \theta_t - \frac{\eta}{u_t}\widehat{m_t} \qquad 3.66$$

Where $\eta = 0.002,\ \beta_1 = 0.2, \beta_2 = 0.999,$ and the other parameters are the same for Adam.

Furthermore, the Adam algorithm has been improved in various ways, including rectifying the variance term. The rectified Adam (RAdam) addresses the enormous variance challenges in the early adaptive learning rates by reducing the variance of the model parameters. The rectified variance is given by

$$r_t = \sqrt{\frac{(p_t - 4)(p_t - 2)p_\infty}{(p_\infty - 4)(p_\infty - 2)p_t}} \qquad 3.67$$

Where the parameter $p_\infty \leq 4$. The authors outlined that the RAdam optimiser produced a comparable performance to Adam with fewer epochs, making it a faster optimiser [269]. Nevertheless, the Nesterov accelerated adaptive moment estimator (NAdam) is another improvement to the Adam optimiser. NAdam is a compound optimisation technique that combines the properties of the NAG and Adam optimisers to improve model optimisation by modifying the momentum term $\widehat{m_t}$ instead of including the momentum twice. The NAdam update rule becomes [260]

$$\widehat{m_t} \longleftarrow (1 - \mu_t)g_t + \mu_{t+1}m_t\,; \;\; \theta_t = \theta_{t-1} - \eta\frac{\widehat{m_t}}{\sqrt{v_t + \varepsilon}} \qquad 3.68$$

Moreover, another compound optimisation is the AMSGrad which combines the Adam and RMSProp optimisers as a moving average optimiser that guarantees faster convergence of learning models. AMSGrad uses a lower learning rate with slowly decaying gradients than Adam [270]. It also adopts the maximum of past squared gradients like the AdaMax instead of the most common exponential moving averages to update the optimiser parameters. The maximum of the past gradients is given by

$$\hat{v}_t = max(\widehat{v_{t-1}}, \hat{v}_t) \qquad \textbf{3.69}$$

The AMSGrad update rule is obtained as follows

$$\theta_{t-1} = \theta_t - \eta \frac{\eta}{\sqrt{v_t + \varepsilon}} m_t \qquad \textbf{3.70}$$

The AMSGrad shows promising results compared to the Adam optimiser, with researchers suggesting even better performance [270]. Perhaps, the Lookahead is another gradient-based optimisation technique that improves model performance by iteratively updating two sets of model weights. The Lookahead optimiser updates the model weights by choosing its search in the direction of the fast sequence of weights produced by the embedded optimiser. The weight update rule is given by [271]

$$\theta_{t,i+1} = \theta_{t,i} + A\big(L, \theta_{t,i-1}, d\big) \qquad \textbf{3.71}$$

Where $L =$ objective function, $A =$ optimisation method, $d =$ current mini-batch training examples.

Other optimisation techniques include the second-order derivatives, often referred to as the Hessian matrix approximations, swarm intelligence optimisers and parallel computing. The second-order optimisers have Newton's method, Quasi-Newton's method, and the sum of functions method [68]. These optimisation methods are inherently faster but computationally expensive, slower to compute, and not memory efficient. On the other hand, swarm intelligence optimisers are evolutionary, reliable and quick techniques for finding solutions to optimisation problems, inspired by biological methods of solving complex distributed computational problems using behavioural approaches of organisms. These behavioural approaches of ants, honey, wasps, bees, birds, and termites are the inspiration of the swarm optimisers, with the various biological optimisation methods including particle swarm optimisation [272], grey wolf optimiser [273], ant colony, firefly algorithms [274], artificial fish swarm optimisation etc. A detailed review of these optimisation techniques can be found in the literature [259].

Besides, parallel computing is another optimisation technique that improves model convergence by using multi-core tight coupling of processing units, ensuring low latency between the processor's computing gradient updates. A popular parallel computing method is the SGD parallelised method, which improves the SGD optimisation method for deep learning applications[275], [276]. Parallel computing can be synchronous or asynchronous depending on the configurations, with synchronous computing affected by slow computers on the network. In contrast, the asynchronous is not affected by the computer hardware issue. The parallelised SGD is obtained as follows.

$$v_i = SGD(c^1, \cdots, c^m, T, \eta, w_0) \qquad \textbf{3.72}$$

Where $T = the$ number of instances per machine, and the values of i lie between $i \epsilon 1 \cdots k$. The overall sum of the computer gradients aggregate becomes

$$v = \frac{1}{k} \sum_{i=1}^{k} v_i \quad \textbf{3.73}$$

Furthermore, parallel asynchronous computing provides improved training speed by distributing the processing to many central processing units (CPU) and graphics processing units (GPU). However, combining multiple (four) GPUs enhances training speed thrice compared to a single GPU [277].

In summary, the gradient-based SGD has been the dormant optimisation method for deep neural network applications and has offered good promise since its invention. It has also performed significantly better than the most adaptive optimisers for prolonged training time to tune the model hyperparameters [278], while the adaptive optimisers offer improved convergence speed. On the other hand, the second-order optimisers and the more recent swarm intelligence optimisation methods have limited applications in deep learning research. However, the latter is still a new research area. However, it is worth highlighting that no single optimisation technique offers the best performance on converge speed, accuracy, and model generalisation, thereby choosing the most appropriate optimiser heuristic. Therefore, a trial of the different optimisers as a model hyperparameter is the only guaranteed technique for selecting the best optimiser for an application.

b) Parameter Initialisation

The parameter initialisation for neural networks is another vital optimisation technique in neural network models. It determines the initial weights and biases for the chosen model and the overall performance of the neural network. Research suggests that deep neural networks have many parameters and a non-convex loss function response, making them challenging to train in real-time [68]. However, proper weight initialisation is crucial to training these models to achieve fast convergence [246], [263]. Model weights and biases are essential parameters of the neural networks alongside the learning rate, which determines how fast the gradient descent algorithm attains the global minimum and defines the starting point of a training process [29].

Furthermore, it helps the model avoid exploding and vanishing gradients during propagation[157], [263]. The primary goal of parameter initialisation is to break symmetry within the model's hidden layers. A proper initialisation ensures that signals are not inappropriately magnified or reduced but kept under control during the training [33].

Weight initialisation is a research area that focuses on the different techniques researchers adopt to improve model performance. The early weight initialisation approach includes sparse initialisation, where units are initialised to a constant non-zero value [279]. However, the arbitrary weight initialisation can slow down or stall the convergence process entirely due to the small variances received by the very deep layers, which reduces the back-propagation process during training [246].

Nevertheless, the Xavier initialisation technique is another weight initialisation approach that uses a scaled uniform distribution and assumes that the activations are linear to maintain variance across each layer. The Xavier initialisation is obtained as a random number with a uniform probability distribution that lies between $\frac{-1}{\sqrt{n}}$ and $\frac{1}{\sqrt{n}}$, with $n = \ the$ number of inputs to the node. The authors also proposed the normalised weight initialisation method that considers the number of inputs and output to the model's node to provide weight that can break symmetry during training successfully. The normalised weight initialisation also uses random numbers with a uniform probability that lies between $\frac{-\sqrt{6}}{\sqrt{n+m}}$ and $\frac{\sqrt{6}}{\sqrt{n+m}}$ where $n =$ number of inputs to the node and m = the number of outputs from the node [280], however, the major limitation of the Xavier initialisation methods was that it assumed that the models were all linear, while this is not the case when the model has rectified activations and suffers poor convergence with very deep architectures having over thirty layers [33].

Besides, the early attempt to use Gaussian distribution in weight initialisation saw researchers exploring the initialisation of model weights using Gaussian distribution with a standard deviation of 0.01 for weights and the biases set to 1 [157]. This method was not very successful as it suffers from poor convergence too [33], but an improvement in the use of the Gaussian distribution produced a new initialisation method for non-linear activations that used Gaussian distribution with $zero$ mean and standard deviation of $\sqrt{\frac{2}{n}}$ where $n =$ number of inputs to the node. This approach named Kaiming initialisation was successful to initialise the ReLU activations and further improve the performance of deep architectures.

More recently, a statistical weight initialisation technique using data statistics has been proposed. The data-dependent initialisation was used to initialise the network and tested on practical datasets. The authors reported better performance than other initialisation techniques on practical datasets [281]. However, a critical condition is that the learning rate must be fixed across all the layers during the training, making it prone to prolonged training time. Furthermore, the model learning rate must be set to a small value to prevent the model from converging at a local optimum level.

Another early initialisation approach was the orthonormal matrix initialisation, where a carefully selected scale factor accounts for the non-linearity [282]. It is a layer-sequential unit variance initialisation approach where the orthonormal matrix was helpful to pre-initialise the weight of the respective convolutional layers, with the first layer output normalised to 1 [246]. The researchers outlined that this approach performed better than the initialisation from a Gaussian distribution.

Conversely, another recent initialisation technique proposed is the decision trees initialisation technique for deep learning applications, especially for feedforward networks. This technique involves training a collection of decision trees and mapping them to a group of initialized neural networks, with the network structure determined by the tree structure. The technique has been used in training predictive models on complex datasets and produced promising results on regression and classification tasks. Numerous initialisation methods not covered in detail include the identity matrix technique and the variance scaling approaches [282].

Research suggests that choosing the uniform or Gaussian distribution does not matter much. However, the scale of the initial distribution is vital [2], with a more significant initial weight providing a higher symmetry-breaking effect, avoiding signal losses during the forward and backward passes alongside exploding gradients. Furthermore, selecting optimal weight and bias parameters alone cannot guarantee optimal performances. Most importantly, the model behaviour during training is dynamic, and the parameters are not only the biases and weights but critical for enhanced performances.

3.11.5 Regularisation Techniques

Regularisation addresses the overfitting challenge of the CNN models during model training. It is helpful to improve the overall performance of learning models [283], reduce overfitting and other essential parameters in neural networks that enhance model generalisation on unseen data. It achieves this by altering the learning algorithm to improve its performance by modifying the connections between sequential network layers to discourage co-adaptation [284]. Different techniques for regularising deep neural networks include regularisations at the respective nodes, data regularisation, and loss regularisation. These techniques provide various degrees of model improvement.

a) Loss Regularisation

The loss regularisation involves adding a cost to a model loss. The methods to achieve loss regularisation include L1 and L2 regularisation. These regularisation methods update the general cost function by adding an external penalty term. The L1 regularisation has the penalty term proportional to the absolute value of the weights, while the L2 has the penalty term proportional to the squared value of the weights. Loss values regularization is added by penalising the large weights, thus

$$\text{⋌} \sum_{l=1}^{k} (\theta)^2 \qquad \textbf{3.74}$$

Where ⋌ = weight decay hyperparameter that controls the strength of the regulariser. The regulariser approach term is often called L2 regulariser, with the regularisation and loss terms combined to give

$$L = \frac{1}{n} \sum -\log(s_j) + \text{⋌} \sum_{l=1}^{k} (\theta)^2 \qquad \textbf{3.75}$$

The effect of the L2 regulariser is that higher parameters will produce higher errors and would be less likely to be selected in the final parameter set.

b) Node Regularisation

Node regularisation involves modifying the parameters of a model node to enhance performance. These techniques of node regularisation include dropout, dropConnect, dropPath, scheduledDropPath and BinaryConnect during training. Dropout is the most common node regularisation method used in deep learning. It randomly omits or drops a subset of activations within each layer during the training of deep neural networks, thereby preventing co-adaptation of activations. The dropout proposes to reduce overfitting problems observed during training neural networks with state-of-the-art performances on supervised learning tasks [241]. The dropout helps determine the number of nodes to omit during model training and can be applied at a model's input or hidden layers. The authors suggest that dropout can also be set as a fixed probability independent of other units derived from the validation set of the model or a specific value of 0.5.

Moreover, the dropConnect is another regularisation method that randomly sets a subset of the model weights to zero during propagation. It drops the model connections rather than output units, with the drop probability obtained as $1 - \rho,$ making the dropConnect, a sparsely connected layer [285]. The DropConnect layer output is randomly selected during training with the output given by

$$r = \sigma((M * W)\, v) \qquad \textbf{3.76}$$

Where $M =$ binary matrix encoding the connection information, $W =$ weights, and $v =$ model input.

Conversely, binaryConnect regularisation is another recent technique that explores using binary weights to train neural networks. For each minibatch, the model randomly picks one of the two values of each weight, forward and backwards, and propagates it, not during parameter updates [236]. It is a method that constrains the model weight parameters into two values of either -1 or +1 during propagation. It is similar to dropConnect but uses Gaussian noise for binary sampling. The BinaryConnect weight is given by

$$w_b = \begin{cases} +1 & \text{with probability } \rho = \sigma(\mathrm{w}) \\ -1 & \text{with probability } 1 - \rho \end{cases} \qquad \textbf{3.77}$$

Where σ is the HardSigmoid function, the authors outlined that the BinaryConnect produced state-of-the-art results on two standard datasets.

Nevertheless, the ScheduledDropPath regularisation technique also improves the existing DropPath regularisation. It proposes an architecture designed to generalize on specific datasets during training and afterwards transfer the learned architectures with state-of-the-art results on available datasets [286]. Researchers first proposed transferring learned architectures using genetic algorithms to design deep learning structures [287]. DropPath first encodes some fixed-length binary string to represent each network structure and uses genetic approaches like crossover and mutation to search the available spaces efficiently. Next, they tested the genetic algorithm on CIFAR-10 and ImageNet datasets with notable poor performance results. However, they established that deep learning structures are learnable and transferrable despite poor performance.

Finally, other dropout techniques exist in the literature, including dropPath regularisation, where paths are dropped stochastically within a cell of fixed probability [284], spatialDropout adds another dropout layer before the 1 x 1 convolutional layer [288], and entropy regularisation for deep reinforcement learning models [289], among others. It is vital to outline that the majority of these dropout methods are only applicable during training to enhance generalisation and avoid model overfitting in most cases; however, at test time, all the activations are helpful to test the performance of a model.

c) Data Regularisation

The data regularisation involves modifying the data to achieve improved model performance. The methods of attaining data regularisation include batch training, data augmentation, data normalisation, etc. Data augmentation is another valuable model improvement technique in deep learning research. It is a process of sending batches of images and randomly applying a series of transformations on each of the images in the batch, replacing the initially obtained batches with the randomly transformed batches, and afterwards using the new batches to train the model. It involves creating more dataset samples with specific transforms, which helps the model better generalise when deployed to classify unseen data. The augmentation transforms include image translation, rotations, scaling, shearing and flipping[257]. Data augmentation is mainly used to increase the size of training examples. Still, researchers have suggested that it reduces the need for model regularisation, especially in deeper architectures where augmentation enhances model performance significantly, producing a reduced error rate in model predictions [213].

d) Batch Training and Normalisation

Batch normalisation is another valuable technique for addressing internal covariance shifts in feature maps. It standardises the inputs to subsequent layers of the neural network during training, improving the performance of the deep learning models [290]. The internal covariance shift refers to the changes in the value distribution within the hidden units that limit model convergence by enforcing a small learning rate [32]. The batch normalisation enhances the training speed of deep neural networks and model stability. It uses the first and second statistical moments (mean and variance) to normalise the activation vectors of hidden layers in a DNN. The batch normalisation can be applied before or after the non-linear activations in a neural network and is mainly implemented as a layer in most deep learning libraries.

Moreover, the mean (μ) and variance (σ) of the feature map of the mini-batch are respectively determined by the batch normalisation layer during training as

$$\mu = \frac{1}{n}\sum_i Z^{(i)} \, , \; \sigma = \frac{1}{n}\sum_i Z^{(i)} - \mu \qquad \textbf{3.78}$$

The transformed feature maps or activation vectors are then normalised using the relationship

$$Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}} \qquad \textbf{3.79}$$

Where $Z^{(i)}$ = input feature map, $Z_{norm}^{(i)}$ =normalised feature-map, $\epsilon =$ numerical stability constant. It ensures that the neuron's output maintains and follows a normal standard distribution across the batch of samples. The output of the batch normalisation layer is obtained by applying a linear transformation using

$$Z = \gamma * Z_{norm}^{(i)} + \beta \qquad \textbf{3.80}$$

Where γ and β are trainable parameters that modify the standard deviation and bias of the model, respectively. The model computes the mean and standard deviation of each batch iteration and then uses gradient descent to train the $\beta\ and\ \gamma$ parameters using an exponential moving average to give credence to the iteration.

e) Early Stopping

The early stopping technique is another regularisation method used in deep learning models. It is a simple and effective technique that allows for saving the best model validation weights during training and returning to the parameters set at a future time with the best weights parameters [68]. This technique has been one of the most successful regularisation techniques for training deep architectures.

Finally, it is worth outlining that regularisation is a heuristic process that enhances the model's generalisation ability. Perhaps, since it is heuristic, combining and adapting different regularisation techniques can achieve optimal performance.

3.11.6 Evaluation Metrics

The model evaluation metrics are parameters helpful in evaluating the performance of a given model. The metric functions are similar to the loss function; however, the output of the metrics is helpful during model training [29]. The metrics used to evaluate the performance of deep learning-based classification models include accuracy, precision, recall, $F1$ score, false positive rate, false negative rate, receiver operating characteristics and area under the curve [29], [68]. These metrics outline the various performances of any given model based on the known model parameters.

3.11.6.1 Accuracy

The accuracy of the model prediction is one of the most important metrics to outline the performance of a model on a given dataset. Accuracy is a measure of true and false positives in the model predictions. The evaluation of the accuracy of these models considers two different approaches that assess performance. The model accuracy describes the ratio of correct predictions to the total number of predictions. It highlights the proportion of the training examples predicted correctly by the model [68]. Accuracy is given by

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{T_P + T_n}{T_P + F_P + T_n + F_n} \qquad \textbf{3.81}$$

Where T_P = true positive, $T_n =$ true negative, $F_P =$ false positive, and $F_n =$ false negative, all obtained from the model predictions. The rank-1 and rank-5 accuracies are the two classification accuracy assessment methods found in deep learning classification literature. These different assessment criteria offer specific advantages that mainly depend on the dataset's size. The rank-1 accuracy in classification problems refers to the percentage of predictions where the top prediction from a test matches the exact ground truth label. The rank-1 accuracy is practical when testing classification problems where the number of inputs is few.

In contrast, rank-5 accuracy refers to accumulating the top-5 predictions from the developed model. All predictions in the top-5 are considered in the accuracy of this type of model. The top-5 is most helpful in classifying large datasets with hundreds to thousands of input predictions and has found typical applications in benchmark datasets [68].

Amongst these accuracy evaluation methods, the rank-1 accuracy was selected to reflect the size of our dataset, as there are fewer object classes to predict. At the same time, the top-5 accuracy was not considered as it would be meaningless; there are a few classes and samples in all the test datasets.

Moreover, the model misclassification M is the number of wrong classifications produced by a model. It is an of accuracy, and it is given by

$$M = 1 - Accuracy \qquad \textbf{3.82}$$

3.11.6.2 Error rate

The error rate is another metric that is useful to describe the performance of a given learning algorithm. It outlines the fraction of the training examples that the model predicted incorrectly [68]. The error rate is like probabilistic outcomes, with values ranging between 0 and 1. An incorrectly classified outcome gives a loss of 1, and a correctly classified outcome gives 0.

3.11.6.3 Precision and Recall

Precision and recall are other metrics used to evaluate models in deep-learning classification problems. The precision is the positive predictive value representing the fraction of the correct model predictions [68]. The relationship gives the model precision

$$Precision = \frac{T_P}{T_P + F_P} \qquad \textbf{3.83}$$

Conversely, the model recall is defined as the true positive rate of the model. It outlines the fraction of true events detected by the model and is often described as the model sensitivity [68]. The recall is given by

$$Recall = \frac{T_P}{T_P + \ F_N} \qquad \textbf{3.84}$$

It is essential to highlight that precision and recall are important classification metrics that are most helpful for evaluating models with an imbalanced dataset. These solely depend on the most important desired outcome of the model, either to have a low false positive or a low true negative.

3.11.6.4 F_β Score

The $\mathrm{F_\beta}$ score, often referred to as $\mathrm{F_{number}}$ is a measure of a model's test accuracy using precision and recall. It is another metric used in evaluating imbalanced classification tasks and is often referred to as a harmonic mean of the model's precision and recall. $\mathrm{F_\beta}$ can apply additional weights to the precision or recall to emphasise the model's precision or recall. The $\mathrm{F_\beta}$ can take multiple weight values depending on the application, and these numbers can range from zero to small positive numbers. The $\mathrm{F_\beta}$ is given by [291]

$$F = \frac{(1 + \ \beta^2) \ \times recall \ \times precision}{(\beta^2 \ \times precision) \ + recall} \qquad \textbf{3.85}$$

Where the β is the weight factor controlling the recall or precision more heavily. However, the perfect $\mathrm{F_\beta}$ is 1 highlighting the equal contribution of both the model's precision and recall, and is often referred to as $\mathrm{F_1}$ number. The $\mathrm{F_1}$ number is essential to summarise the performance of these models with a single number instead of creating the precision and recall curve. This $F1$ number is the harmonic mean of the model precision and recall, given by

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \qquad \textbf{3.86}$$

3.11.6.5 Receiver Operating Characteristics (ROC)

The receiver operating characteristics is another model assessment technique for selecting and interpreting the performance of binary classification models. It uses the false positive rate and true positive rate metrics to highlight the relationship between true positives and false positives. It also helps to visualise a confusion matrix at different thresholds. It shows the proportion of negative classes in a model classified as positive with the false positive rate displayed on the x-axis.

The ROC curve is obtained by plotting the true positive rate (TPR), often described as the model sensitivity, and it defines the proportion of positives adequately classified. The TPR is obtained using the following relationships

$$TPR = \frac{TP}{TP + FN} \qquad \textbf{3.87}$$

The false-positive rate (FPR) describes the portion of the negative classes incorrectly classified in a given model. The FPR is often referred to as a Type 1 error and is given by

$$FPR = \frac{FP}{FP+TN} = (1 - specificity) \tag{3.88}$$

Conversely, the true negative rate (TNR), often called specificity, describes the proportion of negatives that a given model correctly classifies. The specificity, also defined as the TNR, is obtainable using the relationship

$$TNR = \frac{TN}{TN + FP} \tag{3.89}$$

The false-negative rate describes the portion of the positive class incorrectly classified by the model. The FNR is often referred to as a Type II error and is obtained using the following relationship

$$FNR = \frac{FN}{TP + FN} \tag{3.90}$$

Conversely, it is desirable to have a model with a higher TPR and lower FNR to effectively classify the positive classes and a higher TNR with a lower FPR to classify the negative classes correctly.

3.11.6.6 Area under the Curve (AUC)

The AUC is another binary classification metric used to evaluate model performances. It differs from the ROC metric in that the AUC gives the average sensitivity for all possible specificity values and the average specificity values over all possible sensitivity values [292]. A classifier with an AUC of 1 produces a perfect classification of all the classes, an AUC of 0 outlines that the model got all predictions wrong, while an AUC of 0.5 shows that the model cannot distinguish the classes. The higher value of the AUC, the better the model performs at classifying the different instances.

3.11.6.7 Confusion Matrix

The confusion matrix is a table used to describe the performance of a classifier when true values are known. It is a model prediction visualisation tool that shows the model's ground truth and the predictions in matrix form. It uses the true positive, false positive, true negative and false negative metrics to visualise the model performance, with the diagonal shaded elements showing accurate predictions

	Predicted negative	Predicted positive
Actual negative	True Negative (TN)	False Positive (FP)
Actual positive	False Negative (FN)	True positive (TP)

Figure 17 Confusion matrix table

Finally, it is crucial to outline that choosing a performance metric is explicitly dependent on the application of the model. As researchers suggested, the model's intended behaviour is vital in selecting the evaluation metrics. However, since the applications are primarily classification-based models, the accuracy and error rate parameters are the most valuable metrics. Therefore, these performance metrics are used in assessing all the models investigated in the research.

The selected architecture will process images and produce a suitable output from the fully-connected layer. The general model output is vital since it can be probabilistic or non-probabilistic. For example, the softmax probabilistic output takes a vector of arbitrary real-value scores. Then, it squashes it into another vector, whose values will range from 0 - 1, with the sum equal to one. The non-probabilistic output uses the receiver operating characteristic (ROC) and area under the ROC curve (AUC) to outline the performance of a model.

Overall, the performance evaluation of the deep learning models uses the different metrics outlined; however, the choice of a metric depends on the application and the desired results. Accuracy and error rate are the most used metrics in classification problems. The accuracy metric is often the default metric because it is a single number comparison metric that enhances decision-making on model performance. On the other hand, precision and recall are two numbers, making them more difficult to compare model results and the ROC and AUC curves.

3.12 Applications of Deep Convolutional Neural Networks

Convolutional neural networks have found tremendous applications in diverse fields, from developing models for classifying noisy data, controlling dynamic systems, and recommending and predicting future actions or events. These applications are found across different domains, including object recognition, segmentation, object detection, optical character recognition (OCR), natural language processing, regression, medical image analysis, tracking, robotics, self-driving cars, facial recognition, smartphones, cameras, and so on [28]. However, CNN is primarily useful in object detection and recognition, widely modelled as classification problems. However, this myth is because of the state-of-the-art performance of CNN in classification problems since it was proposed and validated by different authors in literature[28], [156], [157].

The CNN recognition application includes identification and object classification modelled using human perception concepts, where some sensing elements are used to capture scenes for interpretation. These vision systems try to recognise objects by identifying and sometimes localising the objects where the three-dimensional coordinates of the objects are obtained [293]. Furthermore, object detection is similar to recognition. The model takes an input image, extracts regions using bottom-up region proposals, computes region proposals using CNN and finally performs a classification, usually with a classifier model like linear support vector machines (SVM). The detection system uses feature extraction techniques to identify regions of interest within the images and uses them to perform object detection [203].

Conversely, the regression application outlines the relationships between a response variable (output) and some predictor variables (inputs). These CNN applications have also found broader industrial applications in machine fault diagnosis [294], agricultural rocks classification [224], remanufacturing time of equipment prediction [42], sorting of components for remanufacturing [41], [143], and so on. Furthermore, optical character recognition is another application of CNNs that has become state-of-the-art in digit recognition. These models helped recognise banking notes, making it the first commercial application of the CNN models [156]. The OCR application brought other use cases of the CNN models in today's legal, insurance and general document digitisation applications.

Furthermore, facial recognition is another very recent application of CNN models where the CNNs are helpful in the effective extraction of facial features. Afterwards, these features are combined to achieve face recognition using CNN [295]. Moreover, the face recognition application has been extended to smartphones and cameras to identify faces in scenes before capture. Besides, medical image analysis is another application area where the CNN models have witnessed tremendous success, where they are helpful in image detection and predictive analysis of various medical data. This application helps patient diagnosis, drug discovery, precision medicine and predicting protein sequences [296], [297].

Finally, the CNN models have witnessed tremendous applications across domains, as outlined across industrial applications. These applications are increasing as researchers investigate different techniques for improving existing computational models across the domain. This trend will continue and likely birth even more applications of the CNN models. However, it is worth highlighting that the remanufacturing application of the CNN models is still in the infancy stage. It requires much more research to establish the extent and benefits that CNN models could bring to remanufacturing.

3.12.1 Inspection Application

Inspection is a quality control technique that identifies product or component non-conformities to assure quality and reliability [298]. It is a crucial stage in remanufacturing used to assess the economic value of a product and the reusability and reconditionability of such product [82], thereby enhancing remanufacturing process and inventory planning [299], [300]. In addition, it determines the extent of value recovery and reconditioning required to return the used product to "as new condition". Research suggests that the inspection process increases business profit by minimising the risk of loss for products that are not remanufacturable before disassembly [309] while mitigating the risks associated with uncertainty in core quality [301].

3.12.1.1 Inspection Techniques in Remanufacturing

The remanufacturing inspection consists of three stages: core acceptance, part, and final product testing [62]. These different stages of inspection offer different benefits and address different challenges. The core acceptance stage is a pre-disassembly inspection that sorts cores uneconomical to remanufacture, thereby determining the core model, the quality or identification of vital indicators that suggest its remanufacturability [94]. Conversely, the part inspection is a post-disassembly inspection that removes unusable parts of the product and confirms that the parts are reusable. It includes all forms of inspection performed during the remanufacturing process up to the reassembly before the final testing. Finally, the final product inspection ensures that the product is in complete working order and meets the desired quality before the warranty is placed on the product.

Moreover, there are three different methods of performing an inspection in remanufacturing, including manual inspection, automated inspection, and semi-automated inspection, also referred to as hybrid inspection in some literature [302]. The manual inspection involves human experts checking for defects and deciding to reject, rework or accept the product. Since its inception, manual inspection has been characterised by low quality and slow output. These inspection cells are usually integrated into the functional remanufacturing workshop floor [303] for easy core classification and to determine the extent of the core deterioration [304]. Perhaps, early research outlines that vision-based inspection systems are expensive to develop and deploy and further highlights that the primary driver of automation is the error-prone and slow expert inspection outcomes due to the massive production volumes, requiring 100 percent inspection, longer inspection time, and the high litigation costs if faulty products are delivered to customers [302].

Conversely, the hybrid inspection explores automating some of the tasks involved in the inspection instead of having experts perform all the activities. In contrast, automated inspection systems use sensors to capture process data, process the sensor signals, and classify based on the sensor data. The automated inspection does not include manual activities or human experts in the inspection process, with the two early inspection methods including feature matching and image subtraction [302]. The image subtraction matches a recorded image against a perfect image for similarities. However, the subtraction technique has minimal applications and is usually slow. In addition, some parts do not always match at end-of-life due to usage wear, making it very difficult to deploy in remanufacturing applications.

Similarly, the feature-matching approach compares selective features and the features corresponding to the perfect pattern. The feature matching approach is called local feature extraction or template matching. Perhaps, another approach to automated inspection includes the learning model approach, where a neural network is used to create a computational model that automatically learns the patterns in the data without human interference.

Nevertheless, researchers have adopted different techniques while developing systems for deployment in remanufacturing inspection. The methods include the metal magnetic memory technique, visual inspection, Taguchi method-3 pattern recognition, FUZZY Technique for Order Preference by Similarity to Ideal Situation (FUZZY TOPSIS), etc. These inspection methods are discussed in detail.

Firstly, the metal magnetic memory (MMM) is a non-destructive inspection technique helpful in evaluating the degree of damage to cores, interfaces and coatings of components and products in remanufacturing. The method's advantages include cheap and quick implementation and not requiring data preprocessing, making it suitable for evaluating early damages on cores for remanufacturing. Furthermore, it can assess different fault types, including crack length, plastic deformation, stress concentration and fatigue life of ferromagnetic materials [305]. This technique considers the MMM signals induced by the component damages, applied stress and frictional wear.

Besides, using the MMM has witnessed researchers investigating the MMM inspection technique for predicting the residual life of structural cores [306]. Researchers demonstrate the potential of MMM in detecting micro and macro-crack and predicting the residual useful life of a core, which is necessary to ensure that reused cores do not have inherent faults capable of causing premature failure. The MMM are particularly useful for detecting sub-surface faults in cores and have helped detect damage in remanufactured coatings using plasma transferred arc welding (PTAW) [307]. However, this method detects only crack faults on ferromagnetic materials, making them unsuitable for inspecting non-metallic materials, thereby limiting the use of MMM methods alone to achieve automated inspection without incorporating other techniques to inspect non-metallic cores.

Secondly, the Taguchi method-3 pattern recognition technique is another helpful method to identify features that enhance pre-processing inspection. It is a Mahalanobis-Taguchi System (MTS) technique that uses Mahalanobis distance to recognise multivariate data patterns and is implemented for inspecting automotive crankshaft remanufacturing [63]. The MTS approach has also been helpful in the diagnosis of freshwater quality to determine carbon steel corrosion [308]. However, the Taguchi system only applies to quantitative data, limiting the application to process having historical quantitative data.

Thirdly, the FUZZY method is a multi-criteria decision-making approach applied across domains to select and rank alternatives based on the weight of the criteria. Researchers highlight that the FUZZY method has a common characteristic: multiple objectives and multi-criteria usually conflict with each other [309]. It is another recent inspection approach extended to remanufacturing using the FUZZY Technique for Order Preference by Similarity to Ideal Situation (FUZZY TOPSIS), proposed as a valuable technique for selecting and ranking several possible alternatives using Euclidean distance measurement [310]. The principle assumes that the chosen inspection option will have the longest distance from the perfect negative solution and the shortest distance from the ideal positive solution. This inspection model uses core and component dimensions such as diameter, height, length, surface roughness, groove thickness, and other parameters to optimise the selection of used engine pistons for remanufacturing. However, the technique is complicated, as the majority of the product data is not inherently available, making it difficult to adopt in the remanufacturing context where there are heterogeneous products with different dimensions.

Finally, visual inspection is another NDT method that uses optical signals from a connected camera to perform product inspection. The cameras, lenses or mirrors produce images, which are a projection of three-dimensional (3D) scenes of the world points to two-dimensional (2D) points of known intensity value. These images represent the visual objects, scenes or persons produced by optical devices. Visual inspection is closely related to the image processing field of research, where scientists crave to extract possible information from captured images using computers. It involves three basic steps: importing the image, analysing or manipulating it, and predicting an output.

Perhaps, the image properties are modified to perform any form of image processing, highlighting the key features to be adjusted, including the noise and contrast [311], thereby helping to improve the quality of the images and enhancing object recognition from the scenes. However, low-level image processing is computationally expensive and time-consuming; thus, the need to use optimized hardware and software to enhance the processing time of the image processing system. To achieve this, the use of advanced computing systems that would implement these algorithms, with sizeable random access memory (RAM), central processing unit (CPU) and graphics processing unit (GPU) upgraded to meet the expected results, is recommended [312].

However, the aim of image processing can be summarised: detect interest regions and points, usually referred to as features points within images, and process the information using those features, saving resources including memory storage spaces, transmission time and bandwidth for networked devices, thereby producing optimal results from the available big data, obtainable in real-time systems.

These processing actions perform different forms of transformation on the image data depending on the capability of the developed algorithms, including image-to-image modifications like image enhancement, image-to-information transformations like feature extraction and pattern recognition, or information-to-image transformations like image reconstruction to produce the desired goals. These transformations help practitioners and developers to focus on specific aspects of the project for enhancement using various technologies.

Moreover, computer vision and machine learning are increasingly finding applications in different industries. They have achieved state-of-art in these new applications, including image analysis, classification, recognition, video analysis, natural language processing, and even recommender systems [28]. The remanufacturing application of this modelling technique has witnessed researchers investigating the use of image data for modelling remanufacturing inspection to obtain an automated visual inspection system for remanufacturing [137]. They used the Gaussian mixture model to train a machine learning-based inspection model to assess high-value components' surface corrosion [137]. Although this method successfully categorises corroded and non-corroded areas of tested engines, it was only able to detect one type of surface fault, making it unsuitable for holistic inspection of typical components in remanufacturing; that return multiple defects. Besides, a remanufacturing inspection application was also explored by researchers using ensemble learning, where automotive constant velocity joints were investigated and classified only wear defects [313]. This application is also limited by the number of faults evaluated.

Nevertheless, these existing inspection technology requirements complicate the inspection process, requiring new and more effective solutions. Hence, developing novel systems and algorithms that could enhance and optimise the inspection approaches becomes a research gap for improving inspection in remanufacturing.

3.12.2 Sorting Application

Sorting is another crucial stage of component remanufacturing where parts of a product or returned EoL products are identified and classified. It is usually performed at different remanufacturing stages, including pre-disassembly, during remanufacturing and post-remanufacturing stages. The use of deep learning models in remanufacturing sorting is outlined as follows.

3.12.2.1 Sorting Systems in Remanufacturing.

Convolutional neural network models are architectures used in deep learning research to investigate learning grid-like patterns from data. These models have been explored in various applications. Researchers investigate the application of CNN models for sorting, especially image recognition and object detection, which ranks among the most researched problems in computer vision and machine learning. Machine learning enables computational models to obtain high-level features in data automatically. In contrast, computer vision research enables visual object recognition from historical or real-time data. These fields support well-defined modelling approaches, especially computer vision problems where models are described in three well-defined pipelines, including the pre-processing, feature description and correspondence [314]. Besides, the design of sorting systems has adopted similarly defined pipelines in the different sorting approaches as researchers propose and justify their design methods. These design techniques include the use of the keypoint-based approach, the rule-based approach, the radio frequency identification, and the vision-based approach.

The keypoint-based approach is a low-level image processing technique where the local gradient information of an image is helpful to obtain features, also referred to as keypoints. It uses the changes in the image pixel local neighbourhood characteristic to obtain the key points used to describe the contents within an image. The keypoint feature extraction research has witnessed massive research over the years with several algorithms, including the histogram of oriented gradient (HOG) [149], scale-invariant feature transform (SIFT) [147], speed-up robust transform (SURF) [148], the Implicit Shape Model (ISM) and the Oriented FAST and rotated BRIEF (ORB) algorithms [315], to mention a few. These algorithms differ in their feature computation method, with the SIFT algorithm convolving images with different scales of Gaussian filters and approximating the Laplacian of the Gaussian using the Difference of Gaussian (DoG) technique. The minima and maxima of the obtained DoG are used as the SIFT key points.

Furthermore, the ISM technique is another feature description technique that uses a probabilistic recognition framework to obtain a category-specific segmentation. It is similar to the SIFT algorithm with a two-stage recognition approach that combines recognition and segmentation into a common probabilistic framework. The ISM recognition pipeline uses image patches extracted around interest points, compared against the Codebook, with matching patches casting a vote of probabilities to produce the object hypothesis. The object hypothesis is then refined to compute the category-specific segmentation [316]. The ISM model has numerous models for recognition that have been applied across domains in object recognition and tracking [316], [317]. Besides, the other keypoint based technique is the oriented FAST and rotated BRIEF algorithm, which combines a key point detector FAST, and the BRIEF binary feature descriptor to provide a robust, fast and free alternative to the patented SIFT descriptor, which is a computationally more expensive keypoint descriptor, for recognition and sorting applications [315]. However, an identified limitation of the keypoint-based feature extraction is the inherent time consumption and difficulty in designing the detectors since they are hand-crafted designs and require profound domain knowledge to develop a sorting system based on the models.

Conversely, the rule-based decision technique is a sorting method developed mainly for the automated identification of components for remanufacturing, proposed to reduce the challenges of the keypoints-based design method. The rule-based method is more straightforward to design than the keypoint method. It uses a recognition logic consisting of identification numbers, barcodes, and other inherent features like dimensions, weight, visual appearance, and volumetric representations [41]. These systems consist of a camera unit, a weight scale to obtain the input data to the system, and a vision classification algorithm that processes the images for the rule-based decision system. The parts are sorted into remanufacturable parts, recyclable parts and waste. The authors outlined that the rule-based sorting system could identify objects to an accuracy of 96% on the dataset used to train and test the design. However, a unique challenge for deploying the rule-based sorting system is that the automation provided by the system is time-consuming, as the returned cores are first tagged with bar codes before passing them to the sorting system.

Nevertheless, radio frequency identification tags (RFID) is another sorting method identified in remanufacturing. The technique uses RFID, a wireless communication technology that allows systems to read and identify distant electronic tags without requiring a battery in the tags [318]. The tags gather information about an embedded product, store and transfer relevant product data through an effective communication system that is processed to ascertain the product conditions. These devices used to gather data through this method is often referred to as product-embedded information device (PEID) [5]. This method offers crucial benefits compared to the rule-based approach that uses bar codes to gather and process product data quickly and accurately, alongside performing well in significantly harsh environmental conditions. However, a limitation of the RFID sorting system is the inaccuracies arising from missing data and reading errors, which makes them unsuitable for most applications [319].

Moreover, the learning-based sorting systems use machine learning models to recognise end-of-life products from video streams, thereby enhancing waste stream management [320]. This technique considers sorting product streams where the incoming items are recorded and recognised automatically, thereby predicting the required process for each sorted product. It primarily uses computer vision pipelines that require product pre-processing feature extraction. Pre-processing refers to the initial image conversion, resizing, denoising and normalisation, among others, while the feature description stage involves the identification of the interest points in the reference images, as well as the correspondence, where the input images are classified using the interest points also referred to as features[314].

The learning-based model design mainly uses convolutional neural networks (CNN), modelled as a supervised deep-learning problem where labelled data are used to train these predictive models. The learning approach has become very successful lately due to more efficient graphics processing units (GPU), improved optimisation methods, activation functions, regularisation, and augmentation techniques to generate more training examples and speed-up model training [28]. These improved developments have made CNNs, the dominant approach for recognition and detection tasks. Moreover, the advantages of the learning-based techniques include that there is no manual feature detection and extraction compared to the keypoint methods. They also do not suffer missing data as the RFID methods and do not require the bar codes like the rule-based techniques. These advantages support exploring the learning-based models as suitable for automating remanufacturing sorting.

3.12.3 Process Control Application

The process control application explores the use of deep learning algorithms for remanufacturing process control, especially the torque converter post-cleaning process, that aims to remove the contaminations on the surfaces of the cleaned Eol products.

3.12.3.1 Process Control Methods in Remanufacturing

The process control application enhances the inspection and repair of damaged components, improving their outlook. The cleaning process is also essential to develop automated processes during remanufacturing. Cleaning is a very complicated and costly process, identified as the second most expensive remanufacturing process after disassembly due to the inherent costs of detergents, machinery and electricity to put the machines to service[10]. The process generally consists of machine-assisted or semi-automated and manual cleaning processes, which are time-consuming and labour-intensive. Nonetheless, researchers have identified a significant concern for cleaning systems in measuring cleanliness [10]. In most industries, cleanliness is an expert judgment-based task, which depends on the worker's experience and the post-cleaning inspection is another manual process. The automation of the post-cleaning inspection is the focus, where vision sensors are used to verify the cleaning status.

Besides, this application aims to automate the process control by using convolutional neural networks to model and assess the post-cleaning process output of an automated cleaning system. It is achieved by obtaining data about the expected cleanliness condition, training a deep neural network model to classify the cleaned objects, and using the obtained output to control the subsequent processes. Researchers suggest that most quality inspection investigations focus on the surface detection of component issues while paying little attention to the process and control factors [321]. The consideration of the process control factors births a new remanufacturing application that explores deep convolutional neural networks for achieving process control in remanufacturing. This control approach is described as soft sensors in literature [322], [323]. Experts manually examine the torque converter system's post-cleaning inspection, with this method being an expensive inspection approach. An alternative decision-making tool is proposed to achieve the same result or even better. The CNN model decides the next process for activation, enhancing the system's post-inspection process and overall productivity.

Conversely, process control is crucial in industrial applications to ensure that the quality of products meets expectations and achieves consistent processing quality [324]. As remanufacturing aims to return used products to as-new conditions with a warranty, the differing quality of these end-of-life products is a crucial challenge, most significantly, to provide tools that can automatically assess the products' conditions quickly. Besides, novel technologies have been developed to automate various remanufacturing processes, namely disassembly, cleaning, inspection, sorting, reconditioning, and testing, to improve process efficiency. However, most of these technologies cannot handle substantial quantities of process data, making the traditional methods requiring prior knowledge of the systems impractical [325].

Nevertheless, emerging technologies like deep learning algorithms can unlock the various use cases of different technologies, primarily to achieve end-to-end digitisation of physical assets. Industry 4.0 represents the fourth industrial revolution. It increasingly enables the use of data to create higher value and customer benefits by connecting organisations, resources, and products alongside enhancing availability[15]. Nonetheless, process control technologies are a vital aspect of the industrial revolution, improving remanufacturing.

Besides, there are two process control forms: the data-driven models and the model-driven (first-principle models) [322], [326]. The model-driven control describes a process's physical and chemical backgrounds and uses the ideal steady-state process conditions in the model development [326]. In contrast, the software control uses computational models and historical process data to reflect real-time conditions in modelling processes. These different techniques provide various benefits based on the application. However, in industry 4.0, process automation has gained wider acceptance as more sensors and actuators are installed in process plants to gather process data, generating massive process data [325]. As these big data are obtained from the processes, it becomes paramount to develop tools and technologies that can leverage them to make insightful decisions. These technologies and tools include proportional integral derivative (PID) controllers, multivariate statistical analysis methods, and model-data integrated techniques [325].

PID control is the earliest and most successful technology for process-based industrial applications. It uses quantitative measurement methods where the input-output measurements from the plant and some controller parameters are adapted to achieve process control [325]. These process variables are good feedback loops to determine product quality rather than its product. This technique makes the PID control mainly used in most manufacturing applications. However, the online measurement of critical process variables has been identified as a significant limitation of the PID control method due to their economic (high cost of the sensing systems) and technical limitations [322], making them unsuitable in most remanufacturing applications.

Nevertheless, statistical control is another process control approach that depends on product usage data. These process data are described as data-rich but information-poor [327], with researchers suggesting that latent variables are suitable for characterising low-dimensional subspaces in such a scenario. The principal component regression and partial least squares are the most used approaches for managing industrial data correlations [322]. However, these methods are also limited since it requires vast amounts of data for proper generalisation. Furthermore, they can also provide tremendous benefits at the core collection stage, where product usage data help determine the remaining useful life of the product before core acceptance for remanufacturing. Besides, as the product usage data are not inherently available in remanufacturing [111], this constitutes a barrier to using the statistical approaches to adequately monitor and analyse the sensor measurements over time, making the statistical methods more useful in manufacturing applications.

Conversely, the data-driven measurement technique also uses historical data to control processes. The data-driven techniques often referred to as soft sensors in literature find practical applications in processes with massive historical data that can be used to model the soft sensor. These deep learning architectures consist of multiple layers of parameterised non-linear functions; the algorithms achieve better generalisation for highly dynamic non-linear systems [328]. The ability of deep learning algorithms to represent these highly dynamic functions is an attraction to the remanufacturing industry. The algorithms have also become the state-of-the-art method for modelling data-driven control systems, with applications in crude distillation [322], bioprocess fermentation [323], and other industrial cases already investigated.

Nevertheless, one of the practical difficulties encountered by soft sensors is the gradual degradation of the predictive accuracy of the systems due to changes in the state of the process, parameters and sensor drifts. These challenges are overcome by the adaptive nature of neural network weight updates after each training batch, ensuring optimal performance [329].

Perhaps, as the process data from the connected sensors increases due to the ever-increasing need to enhance the monitoring and control of systems, and processes, better decision-making tools that support insightful decisions using the process data become inevitable. Exploring the methods for using process data for process control involves developing and deploying computational models that can process these data to make insightful decisions. For example, soft sensor systems have been helpful in fault detection, process monitoring, and online prediction [326].

However, the remanufacturing processes have witnessed the software and hardware controls used to automate the laser remanufacturing process. The application has an embedded piezoelectric sensor, infrared thermometers, and a PID controller used in the design [330].

Nevertheless, to the author's knowledge, no standalone soft sensor applications are documented for remanufacturing processes; therefore, exploring these alternative qualitative controls for industrial remanufacturing applications is critical for enhancing various remanufacturing activities. However, the primary concern for soft sensor controls is the substantial computational cost of processing (training) the models necessary to achieve the control. They inherently apply to deep learning, computer vision, and other learning-based models that use high computing power to process the application's data. This limitation is often avoided by training the model once and deploying the trained model with the trained weights, with model retraining scheduled when there is a significant increase in the recorded process data.

3.13 Chapter Summary

The chapter provides an overview of remanufacturing the benefits, alongside the productivity issues in remanufacturing. The existing remanufacturing practices and limitations were reviewed to understand the current challenges and outline the technology-based solution in the deep learning modelling. It further reviews various deep learning modelling parameters, including activation function, optimisation techniques, regularisations, loss functions, and evaluation metrics, among others, thereby improving the understanding of the deep learning models. Finally, the technology's suitability was discussed alongside the deep learning architectures, evolution, components and applications of the deep convolutional neural network models, closing the gap in understanding the level of deep learning deployment in remanufacturing (Q1) and theory (Q3).

CHAPTER FOUR

MODEL DESIGN AND APPLICATION TO INSPECTION IN REMANUFACTURING

4.0 Introduction

The previous chapter presented the research design and the model choice and requirements. This chapter outlines the conceptual and actual design, frameworks for deploying deep learning models, dependencies, dataset preparation, model design considerations, development, and testing of the models. It also includes the model development assumptions used in the model design and details the interactions between the layers of the architecture during training/learning. This chapter answers the (Q3) question on developing new deep learning models to improve remanufacturing and applying the model to component inspection in remanufacturing (Q5). It compares the developed model to a state-of-the-art VGGNet architecture to evaluate its performance and applicability to two remanufacturing inspection applications. Finally, the in-case results and analysis are evaluated for proper deductions.

4.1 Background to the Modelling and Development

The modelling and development stage is an essential part of the investigation. It involves numerous activities that support the research's final goal, which concerns modelling and evaluating various remanufacturing processes using deep learning. The model design approach explores modelling the remanufacturing processes as a deep learning problem and analysing the results. Deep learning refers to the process of learning patterns from raw data without being explicitly programmed [28]. The modelling involves two activities which include

- Developing a learning algorithm.
- Performing the classification.

First, a learning model is a function that constructs a classifier given some examples and their classes. In contrast, a classifier is a function that, given any inputs, assigns the input to one of the provided classes [331]. The learning algorithm is also described as the computational model in the thesis. The existing remanufacturing application of learning models for object recognition applications was modelled as regression problems and used to classify cores and parts of a remanufacturing process [42], [63], [137]. However, other approaches to modelling machine learning problems besides regression include classification and Bayesian optimisation problems. Hence, the sorting, process control and inspection applications are explored as a classification problem while evaluating their respective performances. The respective activities of the model development are divided into six main stages, as outlined in Figure 18.1, showing the sequence of steps to obtain the model.

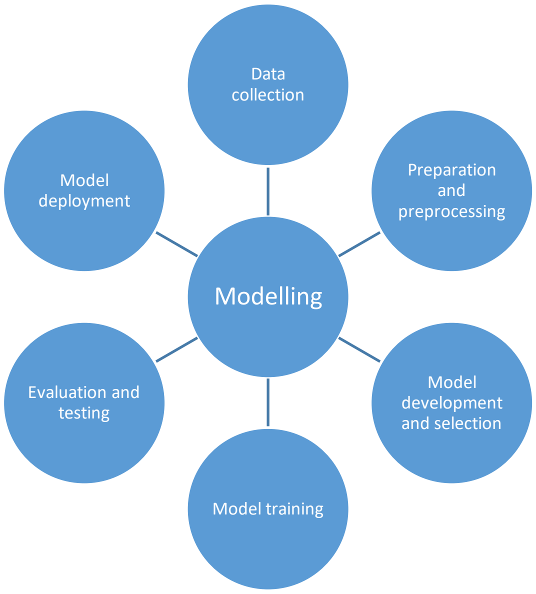


Figure 18.1 Model development stages

Furthermore, the data collection stage is where the videos of samples used for the research were collected. This process was discussed extensively in Section 2.7. The data preparation, processing, training, testing, and evaluation are other stages of the model design that make crucial decisions about the computational model. The preparation and pre-processing involve converting the data to images, resizing the images where necessary and creating the image split for training, validation, and testing of the model. The training stage outlines learning the features from the data and saving the best model weights for reuse. The evaluation and testing stage outlines the model's performance on the data and possible needs for improvement. The deployment stage is the final stage of the modelling, where the models are used in a real-time process for achieving process automation. The core model development and selection involve creating the computational algorithm, setting the initial hyperparameters and selecting the appropriate architectures for specific applications.

Conversely, the model consists of the convolutional neural network algorithm, kernel filters, pooling layers, activation function, loss function, and optimisation algorithm. It also includes other layers: stride, padding, flatten, dense, dropout, batch normalisation etc. Detailed discussions on the respective model development, frameworks and model parameters are presented in Sections 4.5 and 4.6

4.1.1 Modelling Remanufacturing Processes

The first step in modelling the research application of deep learning models to remanufacturing is to decide the type of problem that the algorithm would model. The decision is solely informed by the data type described in Section 2.8. The data for the research is solely qualitative image data; therefore, the deep convolutional neural network algorithm developed is modelled as a classification problem. Classification models have two main components: the scoring function that maps the raw data to specific class scores and a loss function that measures the agreement between the ground truth and the predicted labels. This classification problem is afterwards framed as an optimisation problem where the loss function is minimised with respect to the parameters of the scoring function. The algorithm is a data-driven model that depends on the vast collections of labelled research data that help train the model.

4.1.2 Model Development Boundaries

The research requires the data of the respective processes for adequate modelling of the processes. The inspection, sorting, and process control application data were recorded, labelled, pre-processed, and afterwards used to train a supervised deep convolutional neural network. The supervised learning algorithm is already defined as a computational model or function that constructs a classifier, given a set of examples and their classes. In contrast, a classifier is a computational model or function that, when presented with sample input, predicts or assigns the sample to one of the known $k -$ classes [331]. These definitions further highlight that the developed model is adaptable to make future predictions without holistic changes or modifications to the model. The most crucial goal of the developed model is to find the best classifier that sufficiently predicts an output with very high accuracy on unseen examples. Using these labelled process data provides the boundaries of the research to supervised learning modelling for all the processes and algorithms investigated in the research.

4.2 Research Model Design

The research model design is grouped into two stages: the conceptual and the actual models. The conceptual model for the research is depicted in Figure 4.19. It consists of the cameras (vision sensors) used to record the images of the process, the control scripts to capture image frames, and the computational model to process the images and predict the outputs used to control the process.

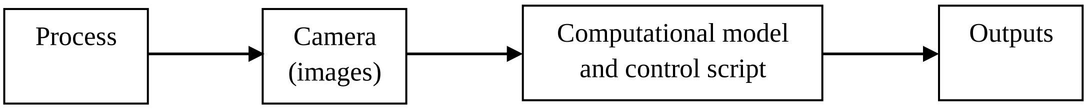


Figure 4.19 Design conceptual model

Besides, the actual model outlines the implementation approach of the remanufacturing application of deep learning models. The design outlines the considerations, including storage facilities, transport, and sensor systems. The other design components include the functions that pre-process the data, train the model, and make predictions using the provided metrics and various evaluations performed in the research. Another vital function is the time delay, which helps manage the transport system's speed for the model to predict the output. The time-delay function helps to control the number of objects the camera sees at any point as it controls the speed at which components reach the point of capture. The actual model design is shown in Figure 20.

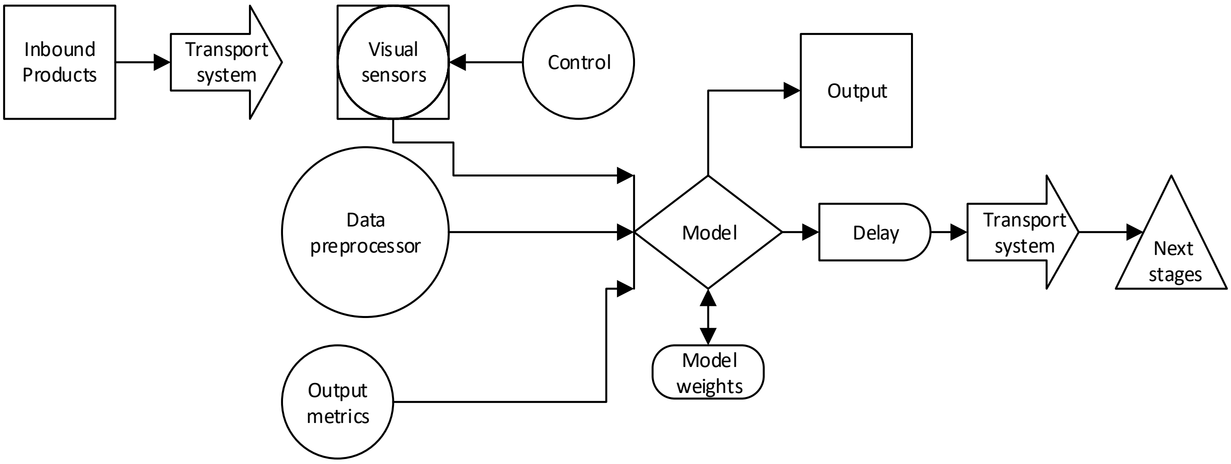


Figure 20 Actual model design

The code development of the design is grouped into layers depicted in the block diagram of **Error! Reference source not found.**. The layered structure simplifies the model code development into specific functions where the designs folder contains all the test codes for testing the model and analysis. Furthermore, the trained weight folder contains the serialised weights obtained after training the model, while the remanAI folder contains all the other functions, including the CNN design contained in the models, the algorithms used to operate the camera for data collection and split contained in the functions, the pre-processor function used to resize and convert the data into array for loading alongside an external datasets folder containing the datasets for the three application with their corresponding labels.

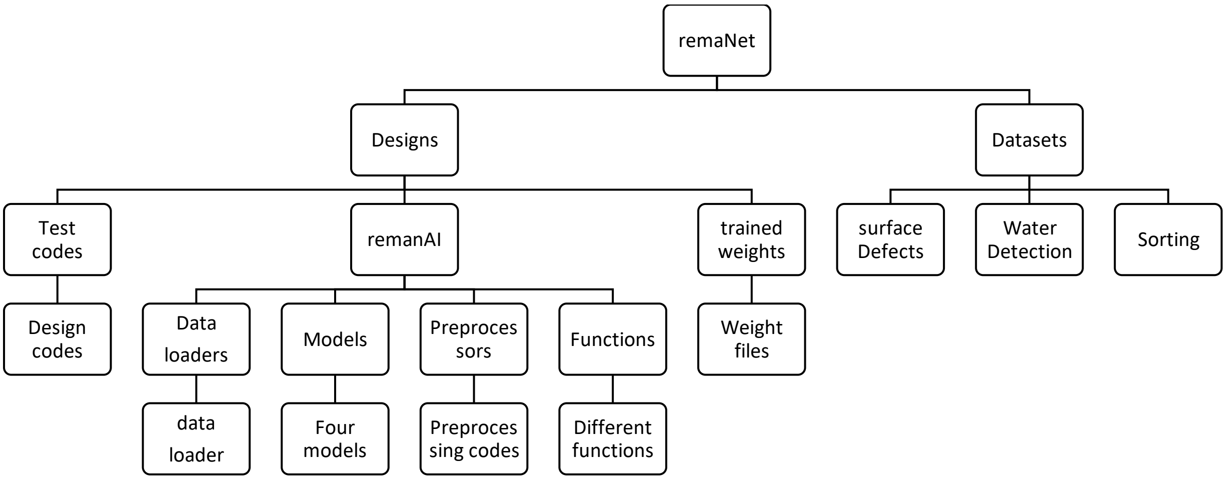


Figure 21 Code development block diagram

Furthermore, the code tree view of the model code development is shown in Appendix 1. The parameter names in blue represent the folders described in Appendix 1, while the executable code functions have the (.py) extension in the file outline. The dataset and the model analysis folder contents are intentionally hidden to save space, as the contents are massive for display.

Conversely, the tree shows the functional groupings of the codes developed and used to achieve the model presented in the research. The groups include the data loader group that contains the necessary commands to load the data and functions containing codes for recording the data, converting it to images and splitting the data. The Pre-processing contains commands primarily used for data splitting, resizing and reshaping alongside the model folder containing the developed neural network algorithms for the research. The frameworks for developing the models are described in the following section.

4.3 Frameworks and Tools for Deploying Deep Architectures

The design of a computational model involves writing the codes to achieve the properties of the chosen mathematical model. The existing tools useful for code development include Python software [332], MATLAB software [333], and R-software [334], among others. Besides, there are also major deep learning libraries and packages developed to enhance the ease of code development, including Caffe [335], Pytorch [336], Keras [337], MXNet [338], and TensorFlow [339] etc. The use of these libraries witnessed most of the deep architectures, modelled as black boxes in multiple-layer networks, hindered quality checks and interpretations at specified points within the data. However, deep architectures are powerful computational models that can quickly learn and represent patterns in high-dimensional data like images, texts, numeric and voice data. The need to understand these models' workings is paramount, and this research further adds to the understanding of these models.

Nonetheless, Python's software for developing the research models is an open-source software that helps developers create and integrate systems easily [293]. Python was chosen because it eliminates software licensing costs and has broader community support when there are code bugs. It is free and compatible with almost all systems, making it suitable for deployment on most computers. It has numerous valuable libraries of functions instead of re-investing all the functions needed in an application. These libraries are included and used in the model development. The model development dependencies include all existing libraries not developed during this research and other open-source libraries, including Keras, TensorFlow, Argparse, Scikit Learn, Matplotlib, Seaborn, os, NumPy, and OpenCV, among others.

Besides, the specific function of the libraries includes NumPy for numerical calculation and Argparse to run command-line codes while selecting the specific files at the exact location and passing other code-specific variables required for the algorithm to run successfully. In addition, TensorFlow library provides deep learning-specific libraries, including activation, dense, flatten, batch normalisation, conv2D, Maxpooling and Keras. Furthermore, Matplotlib and Seaborn provide the model responses; OS helps find the paths to load data and save the trained model. Furthermore, Scikit Learn provides split functions and evaluation metrics, and OpenCV provides image processing functions. The method for importing these dependencies is outlined in the architecture design codes in Appendices 2A, 2B and 2C.

Nevertheless, deploying and training deep network architectures is usually a tedious task primarily because of the enthusiasm shown in computer vision and beyond. Moreover, replicating the state-of-the-art research results in deep network-based systems has been identified as one of the most significant challenges for researchers. Therefore, developing tools for deploying deep networks is a core research area. The early frameworks or tools developed include Torch7, Keras, MATLAB Neural Network Toolbox, TensorFlow, MXNet, Microsoft Cognitive Toolkit and many other new application programming interfaces (API) and tools developed to date. Finally, TensorFlow and Torch7 were the final toolkits considered. They were developed with Python integration; Torch7 is a framework for fast numerical computation with a straightforward extension of its capabilities with library functions developed in Lua scripting language, implemented as a library written in clean C [340]. It can run on CPU and GPU, with the capability to train new architectures of deep networks. However, researchers highlighted its limitation that Torch7 is primarily helpful to prototype models but not for deployment [335], leaving TensorFlow as the final choice for developing ad deploying the research models.

Besides, TensorFlow is another end-to-end, highly scalable machine learning library used for deploying deep learning models in mobile, internet-of-things and production environments. It supports the modelling and experimentation of various machine-learning algorithms. It has found applications from research to industrial practitioners using TensorFlow and its application programming interface (API) for solving complex problems, including image classification, machine translation, speech recognition, and hardware optimisation [341]. The development of these computational tools has aided clear and convenient access to deep architectures, exploring novel training algorithms optimisation techniques and providing faster testing and deployment of deep learning-based models [335], [341]. The TensorFlow library was chosen because it provides an easy pathway to developing, testing, and deploying computational models, thereby reducing the burdens associated with model deployment.

4.4 Data Representation, Preparation and Pre-processing

The array structure is the primary data representation in neural networks, often called tensors. These tensors are usually matrices helpful in representing the inputs and outputs of the given model. Besides, there are two data representation forms: scalar and vector representations. The scalars are single numbers or tensors, described as rank-0 tensors having zero dimensions. In contrast, vectors are an array of numbers used to represent the parameters of a system. These vectors' dimensions range from one-dimensional (1D) quantities to n-dimensional quantities. A one-dimensional quantity is often referred to as a rank-1 tensor, two-dimensional quantities (2D) as rank-2, three-dimensional tensors (3D) as rank-3, four-dimensional (4D) tensors as rank-4 and up to five-dimensional tensors(5D) as rank-5. The typical deep learning applications use image data, 4D data consisting of samples, height, width, and channels, and 5D video data of shape parameters, including samples, frames, height, width, and channels [29].

Overall, these utilities grab the data, process the images using the data processor function, load the data using the data-loader function, convert the data to an array using the image-to-array processor, and finally rescale the pixels to lie between the values of 0 and 1. These utilities are required for all the applications developed in the research, and they ensure that the data becomes helpful in the learning algorithm. Nonetheless, the data for the classification model had a shuffle included before the split to ensure that all the samples in the training set appeared in the validation, ensuring that the entire class of data was randomly selected and used for the training of the models.

Conversely, data preparation is a critical stage of the research because understanding the pixel-level components of the image is necessary. Images are a multi-dimensional grid of values often referred to as picture elements or pixels. These pixels are the building blocks of images and represent the intensity of light in an image. Images are generally described by three parameters: width, height, and depth. The width parameter of an image represents the number of columns in the multi-dimensional matrix.

In contrast, the height parameter refers to the number of rows, and the depth represents the number of channels in the image [257]. Furthermore, these images are represented differently, with the grey-scale and colour images being the most common formats. The primary differentiation between the two image formats is the number of channels in the image, whereas grey-scale images have one channel representing the depth. The colour images have three channels representing depth. The grey-scale images have pixel values ranging from 0 as black to 255 as white, and the darker pixels are found close to 0. In contrast, the coloured images have three channels represented with the red, blue, and green (RGB) colour space and other colour spaces that specify different ordering approaches for the pixels.

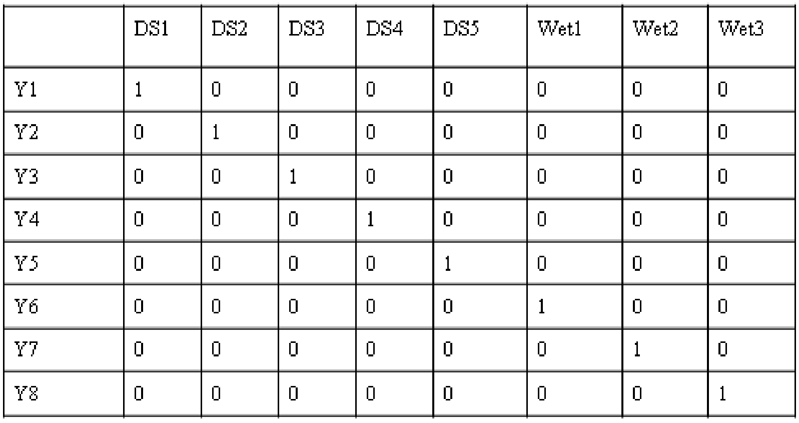
Besides, image understanding and contents remain a vital challenge to researchers, with various authors exploring image kernels which are small filters helpful in applying various effects on images through the convolution operation [257]. These kernels are specially designed filters that perform pre-assigned effects on an original image. Then, the neighbourhood pixels of the image are convolved with the kernel to obtain an output, which is the x-y centre coordinate of the kernel. The kernel effects applicable to images include blurring, smoothing, sharpening, embossing, edge detection, etc. and are essential to process these images.

4.4 1 Data Preparation

The data preparation is the first step to making the data suitable for the respective models. This stage includes structuring the data for the learning model and creating the datasets for the different applications. Data for modelling in machine learning are usually structured to suit the model architecture, which determines the expected data structure. For example, supervised learning requires the data to follow the table structure, with a value pair for each data point [257]. In the tabular structure, each feature is represented as a column and named for recognition, with the order of arrangement of the features not very important. Each item of these variables is represented in rows.

Moreover, the first stage of the preparation in supervised learning is the labelling of the data for the respective classes. These classes are then encoded as categorical variables, where each row and column represents the models' respective outputs. The typical encoding matrix is determined using the number of inputs, which determines the size of the matrix. Table 4.represents an 8-input system used to model the inspection application. The Scikit Learn Binariser was adopted for encoding the labels into categorical data using one-hot encoding. One-hot encoding is the process of converting the direct labels in text form into categories.

Table 4.1 Categorical encoding of model inputs



	DS1	DS2	DS3	DS4	DS5	Wet1	Wet2	Wet3
Y1	1	0	0	0	0	0	0	0
Y2	0	1	0	0	0	0	0	0
Y3	0	0	1	0	0	0	0	0
Y4	0	0	0	1	0	0	0	0
Y5	0	0	0	0	1	0	0	0
Y6	0	0	0	0	0	1	0	0
Y7	0	0	0	0	0	0	1	0
Y8	0	0	0	0	0	0	0	1

Furthermore, the second stage is the splitting stage since the datasets for machine learning and deep learning models are usually divided into training, test, and validation sets. The models are trained and evaluated on the training and validation sets, with the optimiser guiding the loss. The test set helps assess the model performance after training. The noteworthy point is that the test data is not used to train the model. If the performance evaluation results are reasonably high, the model is more likely to perform well in unseen data. Also, the validation set is another vital part of the data for evaluating the model during training and fine-tuning the model hyperparameters described in the subsequent sections.

Nonetheless, the pre-processing stage is another stage of data preparation, which includes data selection, cleaning, normalisation, transformation, cropping etc [342]. After the pre-processing stages, the data becomes ready for the computational model. The typical scaling approach in general machine learning includes normalisation and standardisation techniques. The normalisation entails rescaling the actual value range in the data to lie between 0 and 1. At the same time, standardisation refers to modifying the distribution of the attributes in the data to have zero mean and unit variance or standard deviation of 1 [257]. These data inputs were normalised after pre-processing by converting the images to float data type and dividing the pixels' range within the images by 255 to obtain the respective datasets for the model.

Conversely, the data format for the neural networks usually adopts one of the following formats: arrays, tensors, list of tensors for multiple inputs, a dictionary (dict) variable mapping input names to arrays or tensors when the model uses named inputs, a generator variable and TensorFlow data defined as tf.data that returns a tuple of inputs, inputs, targets and weights [337]. The above data structures helped present the data to the model during training by passing the images through the ImageToArrayProcessor function to obtain an image array.

Another critical parameter is setting the path where the data is stored. These paths can be relative paths, where the data is stored in the same folder with the model or absolute paths, where the data is stored in another location with the entire path directory used to call the data. The absolute path approach was used to store the data, while the Argparse and OS libraries were helpful in retrieval.

The CNN model requires fixed square images to be passed to the model during training. These images were obtained by resizing the images to 52 x 52 x 3 square pixels in the final data used across all the applications.

4.4.2 Splitting the Data

The data for training deep learning models are usually split into training validation and test sets. The training set helps the model achieve the lowest loss possible by adjusting the weight and bias parameters using gradient descent. The validation set is useful to fine-tune the model parameters, thereby selecting the most appropriate features representing the data for better decision-making and generalisation. Furthermore, it helps to direct where appropriate changes are to be made in the model parameters. Besides, the test set helps to evaluate the model's generalisation ability as these samples are kept for evaluation only. The data split is outlined in the block diagram of Figure 22, with the percentage of the split set to 70% and 30% for training, and validation sets, with the test set being the live camera feed.

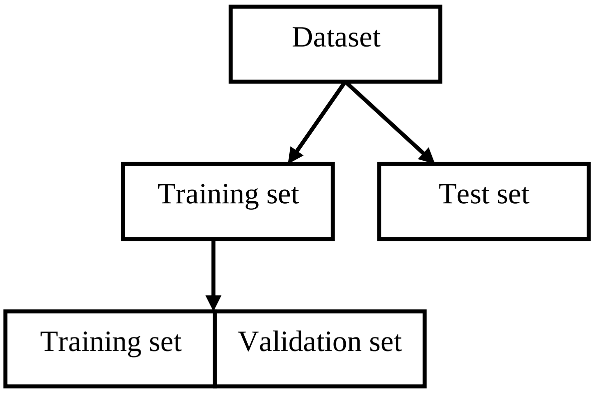


Figure 22 Data splitting method

Nonetheless, model training aims to achieve the most negligible loss and the highest accuracy, thereby obtaining the best model weight parameters for deployment in new or memory-deficient devices.

An observable way to evaluate performance is to observe the model validation loss and accuracy alongside the training loss and accuracy. These parameters start approximately at the same values and end at very similar values showing that the model learned vital features during the training. If the validation accuracy lags well behind the training accuracy, there is an overfitting problem in the model learning, and there is a need to explore possible solutions to it.

4.5 Learning Model and Development Considerations

The choice of model considerations has been outlined in Section 2.5 4, highlighting that the data is the primary consideration for the architecture. First, an appropriate architecture suitable for processing images is selected. The deep learning literature in Section 3.9 suggests that the convolutional neural network is the most suitable model for processing image data. However, to select a suitable CNN model, a hypothetical generalisation of the model's performance must be a function of the hyperparameters of the CNN architecture [343].

Besides, the considerations for general learning-based systems are primarily judged on the model's performance. These performances are accessed based on the ability of the algorithms to produce minimal training errors and high accuracy [68]. Therefore, the following design considerations for implementing deep learning-based models are outlined based primarily on the experiences of the author to investigate, design, and implement deep learning-based algorithms for remanufacturing applications. These vital considerations include but are not limited to the following:

- The choice of the computational model – Depends mainly on the type of data under investigation.
- The size and availability of training examples.
- The available computational resources for training the developed model – The availability of enhanced hardware is vital to train huge models.
- The place of model deployment after the design is crucial – Using lighter or smaller models for mobile applications is crucial when deploying to the cloud or memory-deficient computers.
- A good understanding of the model parameters and hyperparameters, including the learning rate, batch size, the number of epochs, dropout, loss functions, activations, regularization techniques etc., is essential [160] – This knowledge guides the choice of an appropriate model for specific use.

However, selecting the optimal parameters for the hyperparameter is usually an iterative process that requires a good understanding of the processes and careful parameter tuning to achieve significant results. These factors are not exhaustive, as there are other factors to consider in designing and implementing deep learning-based models since experience is always essential in successfully developing and deploying learning algorithms.

4.5.1 The Computational Model Design

The typical building block of a deep neural network is the neuron. The structure of these models consists of interconnected nodes of multiple layers, including the input, hidden, and output layers, with the layers connected to both the preceding and subsequent layers [68]. The output of the respective layers is usually weighted units followed by the activation functions, which are nonlinear functions that distinguish data that is not non-linearly separable. The neuron is a computational model that consists of units that accept input vector $x \in R^n$, the weight vectors $w \in R^n$ and the bias term b, with the output unit y described by the following relationship.

$$y = \alpha \left(\sum_{i=1}^{n} x_i \ . \ w_i + b \right) \qquad \textbf{4.1}$$

Furthermore, the design of the computational model uses the mathematical dot product to evaluate the product of two equal-length sequences of numbers to give a single number. Algebraically, this product gives the sum of the product of the two sequences under consideration. The dot product of two vectors $x_i = [x_1, x_2 \cdots x_n]$ and $w_i = [w_1, w_2 \cdots w_n]$ is given by

$$x.w = \sum_{i=1}^{n} x_i \cdot w_i = (x_1 w_1 + x_2 w_2 \ \ldots \ x_n w_n) \qquad \textbf{4.2}$$

Where $n =$ dimension of the vector space and $\sum =$ summation. However, matrices are generally described by their rows and columns. The appearance of a given matrix is a row matrix if the dot product can be written in matrix product where $x.w = xw^T$ where w^T is the transpose of matrix w. The computational model design defines the entire network's structural arrangement, including the type of layers, the number of layers, the width of the layers, the arrangement of the layers in the architecture, etc.

4.5.2 Model Parameters and Hyperparameters

The model parameters are intrinsic to a model and are optimised during model training. The model weights are the most crucial parameter in neural network models and are vital for optimisation. The model tries to obtain the optimal weights that return the best model prediction during training. In contrast, the hyperparameters are predefined before training, constraining the model to fit the specified data. The hyperparameters are not directly learnt by the learning algorithm and are very important to achieving a high-performing model as they minimise generalisation errors.

Furthermore, these parameters improve model generalisations and prevent overfitting during training. Finding the best parameters of the model is often referred to as hyperparameter optimisation. The best hyperparameters maximise the learning model's performance and differ depending on the dataset. The parameter search consists of searching the hyperparameter space for optimal parameters [68]. However, there is no specific formulated approach to obtain these best hyperparameters for any given model, requiring the heuristic approach to explore and find the parameters that maximise the model learning performance.

Model regularisation was also another critical consideration in the design. Early stopping regularisation was added to the algorithm to ensure that the model's best weights were saved during training and retained in case of subsequent performance depreciation.

4.5.3 Metric Selection

The choice of metrics for evaluating the performance of the developed models was solely dependent on the data considered in the research. The data investigated in the research is a balanced dataset across all the applications, which highlights that the ratio of training samples is equal across the respective classes, thereby informing the decision to use prediction accuracy for model evaluation. Other researchers have adapted this evaluation method to analyse deep learning-based automated surface inspection weld and wood defects [344].

Furthermore, most classification algorithms use the accuracy metric to evaluate the model performance, evidenced across the developed state-of-the-art models in image recognition challenges beginning from the AlexNet architecture, which won the first image recognition challenge using deep learning architecture [157]. The selection of the final evaluation metrics in those state-of-the-art models considered the number of classes in the data. It provided the top-1 accuracy and top-5 accuracy results that represent the model's single top predictions for each class and the top-5 predictions as results for over one thousand classes used to evaluate the model's performance [33], [217], [345]. The top-1 accuracy metric was chosen as the evaluation metric for the developed model because we have a limited number of classes in the data, with a maximum of 20 classes used in the research. The accuracy metric has been described in section 3.11.6.1 as

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{T_P + T_n}{T_P + F_P + T_n + F_n} \qquad \textbf{4.3}$$

Moreover, another metric used to evaluate the deep-learning model performance is the misclassification rate [344], [346]. The model misclassification rate (M) is obtained as

$$M = 1 - Accuracy \qquad \textbf{4.4}$$

In addition, the error rate is another metric used in the model evaluation. It has also been discussed in Section 3.11.6.2. Overall, the accuracy, error rate, and misclassification metrics helped quantify the performance of the developed models.

4.6 Computational Model Exploration

There are two approaches to model design: developing a novel architecture from scratch and the transfer learning method. The novel design approach allows the modeller to tweak the architectural parameters to obtain a high-performing model. In contrast, the transfer learning approach uses state-of-the-art architectures to evaluate the new problem. The pictorial representation of the model design method is shown in Figure 4..

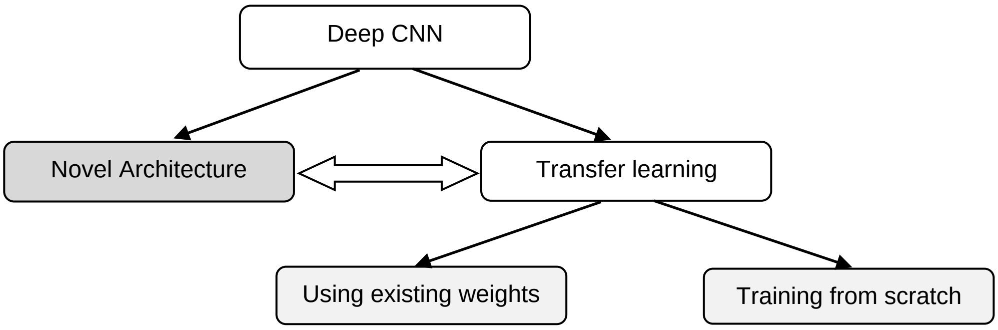


Figure 4.6 Model design approaches

4.6.1 Transfer Learning

Transfer learning is a method of adapting a developed model to new problems. Transfer learning can achieve by either fine-tuning or feature extraction. Fine-tuning is the act of modifying the current model output to fit another dataset. These methods differ based on the type and position of modification performed on the original architecture [257]. The feature extractor method involves the removal of the original architecture fully-connected layer and extracting the features from the data directly from the final pooling layer, with the other model parameters left unchanged. These feature extractors are primarily valuable on smaller datasets.

Similarly, the fine-tuning method involves replacing the fully-connected layer of the original architecture with a new fully-connected layer that fits the data parameters, with all other model parameters being frozen. Fine-tuning is an important technique to obtain classifiers from pre-trained CNN models on custom datasets. Researchers suggest that transfer learning instead of training from scratch improves results on limited data [347]. In most cases, the parameters of the new fully-connected layers are usually smaller than the original architecture as these layers usually have fewer classes than the original architecture. The pictorial representation of the model modifications is outlined on the block diagram in Figure 23, showing the original architecture of the feature extraction modification alongside the fine-tuned modifications.

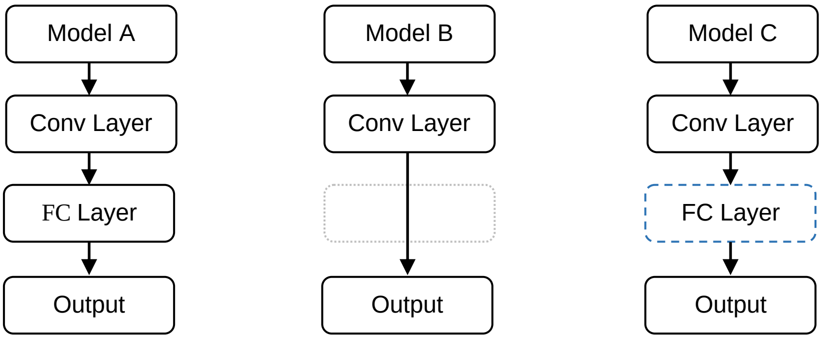


Figure 23 Model modification stages showing the original model A, feature extraction model B, and the fine-tuned model C.

The initial model exploration considers the transfer learning approach, where the final layers of the model architecture are replaced with the data-specific requirements and used to train the model. The vital stages of the transfer learning model detailed in Figure 24 include loading the pre-trained model modifying the final layers of the model, tweaking the model hyperparameters, training the model, evaluation and deployment.

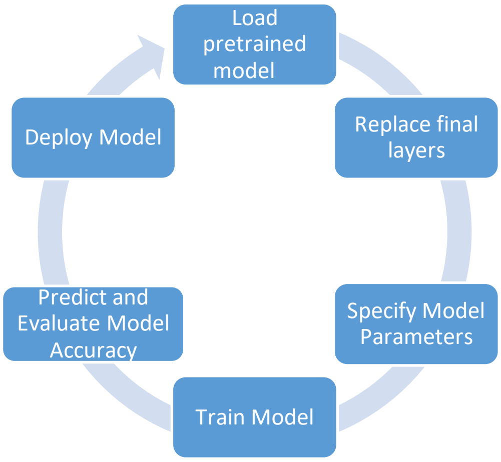


Figure 24 Flow diagram of the stages of achieving transfer learning

The transfer learning approach was first considered because it requires modifying a pre-trained model to learn new tasks, thereby transferring learned features using smaller training examples. Therefore, it is often referred to as fine-tuning, and it is much faster than training a new neural network model from scratch, with a very high computational burden. Moreover, the choice of the pre-trained model was based on the state-of-the-art performance of the pre-trained models from published research [348], [349]. Besides, the preliminary investigation considered the VGGNet architecture to evaluate the performance of transfer learning and training from scratch because the architecture was the first to explore the very deep depths for image recognition problems [158].

4.6.2 Novel Architecture

The architecture development stage involves all the model parameterisation approaches. The Keras model has two modelling techniques: the sequential and functional models. The sequential model helps stack layers of the model together where each layer has one input and out tensor respectively [337]. Conversely, the functional model is more flexible as it accepts a non-linear model definition where parameters can be shared with even multiple inputs and outputs. The model development process involves describing the task in a suitable form for modelling. The starting point for the general machine learning problem formulation is finding the hyper-plane or straight line that best fits the data points. Where the output (y) is predicted from a set of inputs (x), the relationship is given by the response is given by the straight-line equation $y = mx + c$ which has the machine learning equivalent of $y = Wx + b$ where the slope m is equivalent to the weights W, and the intercept c is equivalent to the biases b, and y and x represents the output and inputs respectively. Besides, the multiple inputs into the machine learning model can be represented in more than one dimension as

$$y = W_i x_i + b \qquad \textbf{4.5}$$

The sequential model is a linear stack of layers with the input shape specified before the first hidden layer. When the desired output of the model is expected without modification, the linear activation function, also known as pass-through, is applied to the model's expected output, causing no changes to the model output as the same signal is propagated [350]. However, the deep CNN models require the output to be non-linear, thereby requiring the non-linear activation functions applied at the output of the linear layers of the deep CNN architectures.

The novel architecture outlines the cascading of the new CNN architecture used to perform the remanufacturing inspection tasks to enhance productivity. These improvements can be obtained from either training improvement, memory improvement or the general model design techniques and optimised for the specific application. The development of the novel architecture involves using the knowledge of existing architectures to implement a simpler yet effective model that can perform at similar levels compared to the state-of-the-art. In addition, the new model guarantees lesser computational requirements and provides an application suitable for memory-deficient devices.

Moreover, the design backbone of the architecture consists of the layers as the building block that transforms data into a valuable form for further processing. These layers are banks of filters used to extract the representations in the data [29]. The new architecture is inspired by the need to investigate the effect of different model parameters on the performance of these models alongside the overall model parameter size. Since the existing architectures have a specific architectural design, modifying and naming them after the original is not ideal since the modification will perform like the original design. Therefore, the design considered vital factors outlined in the literature in Sections 3.11 and 4.5 to achieve the model used in this research.

4.7 Learning Algorithms for Remanufacturing Application

The convolutional neural network used in the research is a deep architecture that consists of a stack of multi-layer neural units that performs numerous dot products and linear combinations. The model layers are stacked from a few layers and units and gradually increase heuristically as the model performance is observed. The typical stack of the CNN model is described in the block diagram of Figure 254.9, showing the input, the filters or kernel, the convolutional layer, the pooling layer, the activation layers, the fully-connected layers (Fc), and the output or classification layer. The stacking of the multiple layers is represented as the hidden layers in the block diagram.

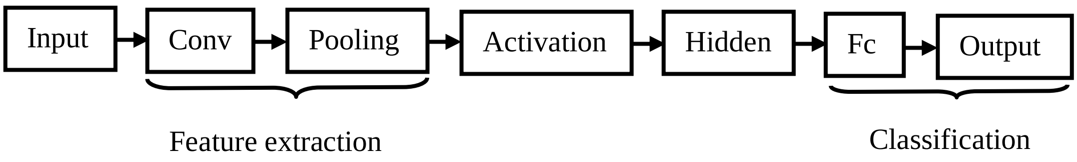


Figure 25 Architectural design block diagram

The model architecture comprises six layers, usually counted as layers with learnable parameters, including four convolutional layers and two fully connected layers. The pooling layers have not been weighed and are not included in the number of layer counts reported in the model. The model's configuration is outlined in Table 4.2.2, which outlines the model's layer-wise makeup, including the activation size of each component in the architecture alongside the number of parameters. The architecture was obtained from heuristically experimenting with the research data.

4.7.1 Understanding the Architecture

The architectural design is essential to determine the number of parameters expected in the model output. Therefore, understanding the architecture is crucial and helps in making vital decisions about the size of the kernel to use, the number of features expected and determining the overall computational requirements of the model. The block diagram design of the architecture is shown in Figure 25.9, with the following explanations of the essential components.

First, the convolutional layer is one of the essential layers in a CNN algorithm. It consists of four hyperparameters, including the number of kernels, the size of the kernels, the number of strides and the padding factor. These four parameters are valuable for the convolution processes of sliding the kernel over the input images. Besides, the kernel is a valuable filter for feature extraction from the images, with its height and width always being nearly square when used in CNN models [257]. It is a matrix that slides over the input data, performs a dot product with the sub-region of input and produces a dot product output or feature map, often called convoluted output. The kernel size is another crucial hyperparameter that must be set before training the models. Previous research recommended the kernel size of 3 x 3, and this dimension has been used in most recent state-of-the-art models to break the symmetry of parameters [158]. Finally, the stride factor controls the sliding of the kernel over an image.

Besides, the convolutional layer parameters used in the architecture include several filters of size 64, 32 and 16, a kernel size of 3 x 3, and stride one with zero padding. Besides, the padding and stride parameters are critical to down-sample model features to reduce the computational cost. The stride parameter determines the extent of shifting of the kernel, and it helps determine the rate of pixel down-sampling. The larger the stride, the smaller the convolutional layer output unless the input images are padded to maintain the input size. In addition, the padding determines the size of the inputs. When set to "same", a zero is added around the input borders to maintain the size. In contrast, when the padding is "valid", the actual size of the inputs is maintained with the trailing features outwitting the stride and kernel dropped at the right-hand end of the features map alongside the bottom of the feature map.

Conversely, the size of the feature map represents the output of each convolution operation, and it is determined as follows

$$\text{Feature map} = (W - F) + 1 \qquad \textbf{4..6}$$

Where $W = size\ of\ input, and\ F = \ size\ of\ kernel.$ Therefore the feature map of an image input of size 52 x 52, convolved by a 3 x 3 kernel, produces a CNN feature output of (52 - 3) +1 = 50 x 50 feature map without padding, thereby reducing the size of the output. This map assumes that stride is 1. Moreover, the padding concept was also introduced to manage the size of the input images. Padding introduces some additional pixels added to the convolutional filters to process the edge pixels outside the pixels in the image. It ensures that the size of the input is preserved. It fixes the border effect issues in input images to a CNN model, preserving information at the object's edges. Thus, the feature map with padding of one zero across the input images becomes

$$\text{Feature map } = \left(\frac{W-F\ +2P}{S}\right) + \ 1 = \left(\frac{52-3+2(1)}{1}\right) + 1 = 52 \text{ x } 52 \text{ features} \qquad \textbf{4.7}$$

$where\ P = \ padding,\ S = \ stride.$ However, the stride parameter moves across the image from left to right, top to bottom, with corresponding one-pixel changes in horizontal and vertical directions. It is worth outlining that the obtained features must be an integer to obtain a valid convolutional layer; otherwise, the implication is that the stride parameters are not properly set, and the neurons cannot be properly tiled to fit into the input volume [257].

Furthermore, the pooling layer is responsible for down-sampling the model size. It requires two hyperparameters that control the pooling operation, including the kernel size and the stride factor. Conversely, the pooling layers reduce the number of features detected by the model. It achieves this by summarising the input features with the local regions or patches using maximum value, ensuring that the learnt features become more robust, invariant, and sparse.

Perhaps, once the features are extracted, the obtained output features are flattened into a one-dimensional vector using the flattening layers, followed by the fully-connected layer. The fully-connected layer is the final layer in the architectural design before the output layer of the CNN. It applies a linear combination of the features from the previous layer before the final activation, flattened and dense layers. Finally, a SoftMax output corresponding to the number of model inputs is used to predict the output where the sum of the total scores is 1. The other model parameters include activation functions, categorical cross-entropy loss functions, batch normalisation, and dropout layers.

Nonetheless, the interaction of the model layers is the most vital design constraint for deep neural networks. These interactions are governed by modelling the behaviour of the respective layers by creating the types of input expected, the outputs, the number of layers, selecting the types of layers, specifying the stride, padding, selection of performance metrics etc, defines how the computational model processes the input and output.

4.7.2 Architectural Arrangement and Initialisation

Architectural arrangement and initialisation are the design consideration during the model development. It involves specifying the number of layers, types of layers and cascading method, number of filters, strides, etc., thereby determining the model's sequence and size of information flow. The components of the CNN model are described in Section 3.11.2 and the model design considerations.

The convolution layers are usually more than one to obtain a deep architecture, requiring more than one convolutional layer to obtain deep architecture to extract features from the images. Increasing the number of convolutional layers enhances the model feature detection capability; however, the larger the number of layers, the longer it takes the model to train, and the likelihood of overfitting increases. The number of convolutional layers used in the architecture is 4.

Besides, the pooling layers help reduce the computational cost of the model alongside model overfitting by reducing the input data dimensionality. For example, the average pooling layer was used to average all kernel values and produce a single score. In addition, a dropout layer was also added to the architecture, which randomly omits some nodes during model training to improve model generalisation and reduce overfitting. A description of the node regularisation techniques is outlined in Section 3.11.5. Finally, as the model's performance is observed, the dropout value used in most high-performing deep architectures is set heuristically between 25% and 50%.

Nonetheless, the model initialisations for the respective layers were of paramount importance as different initialisations for the linear and non-linear parts of the model were considered. The ReLU outputs were initialised using the He initialisation, while the Dense layers were initialised using the Glorot initialisation. The model initialisation was kept to the default Keras initialisation method, which uses the Glorot initialisation method that computes a uniform normal distribution by averaging the number of inputs to the layer, often described as fan_in ($F_{\_in}$) and the number of outputs from the layer known as fan_out ($F_{\_out}$), and taking the square root [280]. The uniform distribution provides a random value from a range of lower and upper limit values where every value has an equal probability of being drawn. The initialisation was achieved using the model parameters where the inputs to the layer (32) and the outputs (20) to obtain the limit of values of weights available for random selection as follows

$$Limits = \frac{\sqrt{2}}{F_{\_in} + F_{\_out}} \qquad \textbf{4.8}$$

$$Limits = \frac{\sqrt{2}}{64+20} = \pm 0.017$$

The initialisation restrains the initial weights parameter of the model to lie between ± 0.017, thereby keeping the model initialisation simple during training.

4.7.3 Parameterising the mapping from Images to Label Scores

The parameterisation of the mappings from the images to label scores starts with defining the scoring function that maps specific pixel values of an image to the confidence scores for each class. Let the training examples of images be represented as $x_i \in R^C$ with associated label y_i where $i = 1, \dots, N$ and $y_i \in 1, \dots K.$ When the number of training examples for the respective datasets is represented as N, from Table 2.1, the total number of samples used for the respective applications is N = 71560, N = 28800, N = 28624 and N = 14312. Also, let the number of distinct objects be represented as K where the applications have K = 20, K = 8, K = 8, and K = 2 for the four applications. The scoring function parameters include C represents the size of the images where C $= 52 * 52 * 3 = 8112$ pixels, and the categories K. Finally, the scoring function that maps the raw pixels to the class scores is defined as follows:

$$f : R^C \rightarrow R^K \qquad \textbf{4.9}$$

This function describes the expected classifier performance where the pixels in an image represented by the 8112 pixels are mapped directly to a specific category in the overall true class.

4.8 Modelling Surface Inspection in Remanufacturing Using Deep Learning

The modelling approach for the surface inspection application is developed as a multiclass classification problem that performs a classification as positive or negative for the respective classes. The multiclass classification differs from the binary classification problems modelling by the model's loss function and the output classifier, a SoftMax function. The parameter modifications are performed in the learning algorithm before model training and evaluation. The surface inspection application is modelled as a supervised learning problem using labelled data. It explores deep convolutional neural networks for developing sorting systems to categorise remanufacturing products and components.

Perhaps, it is worth outlining that the ML techniques were not considered in this study because researchers have outlined that some ML algorithms, including support vector machines (SVM) and Naïve Bayes algorithms, were not very efficient in recognising and classifying automotive components for remanufacturing after extracting features using scale-invariant feature transform (SIFT), and edge histogram descriptor (EHD) algorithms [351]. This research supported further investigation into the application of deeper architectures.

Conversely, surface faults refer to defects highlighting a suspected product abnormality, like what visual inspection experts identify during remanufacturing. Surface inspection mostly depends on image analysis, which is the most significant application in medical image analysis, especially image scanning. Computer vision techniques have more recently been attracting attention across industries, with researchers investigating the application of machine learning and deep learning for remanufacturing inspection using computer vision techniques [137]. However, the application was inefficient as it considered only corrosion faults, while the current application explored multiple fault recognition, including pitting, corrosion, crack, and other combination faults.

Nevertheless, the inspection data consists of objects recorded with varying surface defects for recognition using the research data collection setup. The model is a vision-based inspection system consisting of multi-layer convolutional neural networks that perform a quick and reliable real-time non-contact inspection. The system can check and sort components into scraps and remanufacturable parts and is helpful in the post-cleaning inspection, where products are inspected during any of the stages of the remanufacturing process, including cleaning, reconditioning or even final testing. The novelty of this method is that it considered the same objects having different surface conditions as the categories. In contrast, other existing vision-based applications consider objects of different categories, making them suitable for visual inspection and sorting systems for remanufacturing. The advantage of the developed model is that it can be easily incorporated into other remanufacturing stages with minor modifications.

4.8.1 Inspection Applications and Data

The investigation of deep convolutional neural network algorithms for automated surface inspection in remanufacturing consists of an object recognition system that aims to identify objects in the video stream. These object streams are afterwards inspected and sorted using the designed system. The inspection technique can be adopted for the pre-disassembly sorting, post-disassembly and other stages of operations during or after remanufacturing. The existing surface inspection techniques have been discussed extensively in Section 4.8 alongside the limitations of the existing methods. Finally, the development of the deep CNN-based surface inspection system for remanufacturing applications is presented.

Besides, the surface inspection application considers the torque converter system components and planar bars recorded during this research. The experiment investigates the possibility of detecting abnormalities in the objects from the video stream. The data consist of eight (8) categories of 3600 images per class, making up 28800 images. The distance between the camera and the objects on the conveyor system was limited to approximately 40" to ensure that the camera's coverage was restricted to one object at a time. As lighting contributes to visual sensing, the recordings were made in an industrial work setting to reduce lighting effects after development. The data consist of two separate eight (8) object categories of the torque converter components and some planar metal components used to evaluate the inspection application. These data compositions have been outlined in detail in Section 2.8.3 and used to train the supervised learning model.

4.9 Model Architecture, Parameters and Hyperparameters

The developed inspection model architecture consists of four convolutional filters of sizes $64, 32, 32$ and 16 and two fully connected layers of sizes 512 and 8, respectively. In addition, the architecture has the Swish activation and Maxpooling layers sandwiching the convolutional filters, with a dropout layer added across some of the hidden layers. The number of model parameters obtained from layers with learnable parameters amounts to 1,423,192, approximating to 1.423 million parameters that are learnable during training. The architectural makeup is depicted in Table 4.2.2.

Table 4.2 Model architecture for the inspection application

Layer Type	Output shape	Activation size	Parameters
Input	(None, 52,52,3)	8112	0
Conv2D	(None, 52,52,64)	173056	1792
Activation (Swish)	(None, 52,52,64)	173056	0
Conv2D	(None, 52,52,32)	86528	18464
Activation (Swish)	(None, 52,52,32)	86528	0
Maxpooling	(None, 26,26,32)	21632	0
Dropout	(None, 26,26,32)	21632	0
Conv2D	(None, 26,26,32)	21632	9248
Activation (Swish)	(None, 26,26,32)	21632	0
Conv2D	(None, 26,26,16)	10816	4624
Activation (Swish)	(None, 26,26,16)	10816	0
Maxpooling	(None, 13,13,16)	2704	0
Dropout	(None, 13,13,16)	2704	0
Flatten	(None, 2704)	2704	0
Dense	(None, 512)	512	1384960
Activation (Swish)	(None, 512)	512	0
Dropout	(None, 512)	512	0
Dense	(None, 8)	8	4104
Activation (SoftMax)	(None, 8)	8	0

Besides, understanding how the convolutional layer learns depends on knowing how the parameters move from one layer to another. These learnable parameters depend on the shape of the input and subsequent layers of the model. Furthermore, the number of parameters on every layer is determined by considering the layer kernel sizes, including the width $\boldsymbol{m}$, the height $\boldsymbol{n}$, the number of filters in the previous layer $\boldsymbol{d}$, the bias term $\boldsymbol{b}$ usually =1 for the respective filters, and the number of filters in the current layer under consideration $\boldsymbol{k}$. Therefore, the number of parameters $\boldsymbol{P}$ is a given [352]

$$P = \big((m * n * d) + b\big) * k$$
$$P = \big((m * n * d) + 1\big) * k \qquad \textbf{4.10}$$

Conversely, the total number of parameters of the developed model is estimated as follows with the kernel size of [3,3], which represents $\boldsymbol{m}, \boldsymbol{n}$, channels in input $\boldsymbol{d}$ =3, and current layer $\boldsymbol{k}$ = 64, giving the following layer outputs.

$$Layer\ 1 = \left((3 * 3 * 3) + 1\right) * 64 = 1792$$

$$Layer\ 2 = \left((3 * 3 * 64) + 1\right) * 32 = 18464$$

$$Layer\ 3 = \left((3 * 3 * 32) + 1\right) * 32 = 9248$$

$$Layer\ 4 = \left((3 * 3 * 32) + 1\right) * 16 = 4624$$

Nevertheless, the fully connected layer parameters are somewhat different from the above equations as some models have more than one fully connected layer. The fully connected layer's parameters can be determined using the activation (activ) size of the model as follows

$$P = current\ layer\ activ * previous\ layer\ activ + current\ layer\ activ \qquad \textbf{4.11}$$

$$Layer\ 5 = 512 * 2704 + 512 = 1384960$$

$$Layer\ 6 = 5 * 512 + 8\ \ = 4104$$

The overall architecture for implementing the deep learning inspection application in remanufacturing is attached as Appendix 2A.

4.10 Model Components, Hyperparameter Selection and Optimisation

The model hyperparameters are another essential factor in developing deep learning models. These hyperparameters include batch size, learning rate, etc. The proposed model consists of considerable weight and bias parameters that require optimisation to minimise the empirical classification errors on the labelled training data. A loss function that supports multiclass input is required, and the categorical cross-entropy loss function is selected and already defined in Section 3.10.3. The cross-entropy loss minimises the prediction error on the training data to achieve good predictions. The cross-entropy loss increases as the predicted probability deviates from the ground truth.

Generally, the performance of a model is measured from the error observed in the model response obtained after fitting the data on the model and following the difference between the model's training and test errors. The two important terms used to quantify these errors are bias and variance. The bias term refers to the error of the training data, while the variance relates to the error of the test data.

Conversely, model underfitting occurs when the model's error is very high compared to the training data (low accuracy) and performs poorly on the test data (low accuracy). Underfitting is also undesirable when training a model, as the model performance would not meet the desired performance levels. The property of the underfit model includes a high bias and high variance. Besides, model overfitting is the case of poor generalisation when a model performs well on training samples and poorly on unseen test samples, thereby obtaining low training errors during training (high accuracy) and high testing error during the testing stage (low accuracy). The overfit model has the property of low bias and high variance. Researchers have explored various methods to minimise overfitting early in the model design; overfitting was minimised using different model regularisation methods, including dropout and model checkpoint [160].

However, balancing the training and test error in the developed model is required as low training and test errors are desirable, constituting a model with high accuracy on both training and test data. Considering the bias-variance trade-off, a desirable model should have an ideal low bias and low variance property. This property of a developed model considers reducing the model estimated variance across samples by increasing the bias in the estimated parameters. It ensures that the developed models generalise well on unseen data.

4.11 Surface Fault Identification and Classification

The experiment involves investigating the identification of surface defects on planar metals. These defects are similar to those in EoL automotive parts, including inherent rusts, cracks, and pitting faults. The samples were obtained from the DMEM workshop at the University of Strathclyde Glasgow. The different fault conditions on the samples were recorded as labels, and a three-minute video of samples was recorded as training samples. The recorded videos were pre-processed into images and used for the inspection application. The machine for training the developed models consists of a graphics processing unit that enhances the speed of training the neural network models. In addition, it is a compute unified device architecture (CUDA) enabled NVIDIA GeForce RTX 2080 Super GPU hardware, useful to speed up the training of the models, thereby enhancing the training and model evaluation time. The other pre-processing activities on the data include resizing the data to suit the architecture input (52,52,3), splitting the data into training, validation and test sets, vectoring the data and labels by converting them into arrays for easy access and suitable for the chosen optimisation algorithm. The label vectorisation used in the model is the one-hot encoding which converts the class labels into categorical data of all zero vectors with a one (1) in place of the label index against using the integer tensor that transforms all the classes to integer values. The one-hot encoding allows the class labels to identify the respective predictions, enhancing model performance. The respective Keras utility for the pre-processing was used in the vectorising and encoding of the labels, while a data split function was created and used to partition the model data.

Moreover, this inspection application considers the identification of different surface fault conditions from samples of the objects with various inherent faults, including crack faults (CF), pitting faults (PF), rust faults (RF), and the combination faults: rust and crack (RnC), pitting and crack (PnC), pitting and rust (PnR), and rust, pitting and crack (PnRnC) faults. These inherent product faults are described in brief. Pitting faults are defects formed by the localisation of corrosion confined within a small area on a metal [353], while rusting occurs due to the exposure of metals to moisture and air, forming iron oxides. In contrast, crack defects originate at the surfaces and increase with continued stress [354]. Besides, the pre-processing for the designed inspection application includes converting the video streams into images and resizing the images to suit the model architecture, converting the data into arrays, and shuffling during the batch passes. The pre-processed images were used as data to train the model. A cross-section of the original samples used to train the model is depicted in Figure 26.

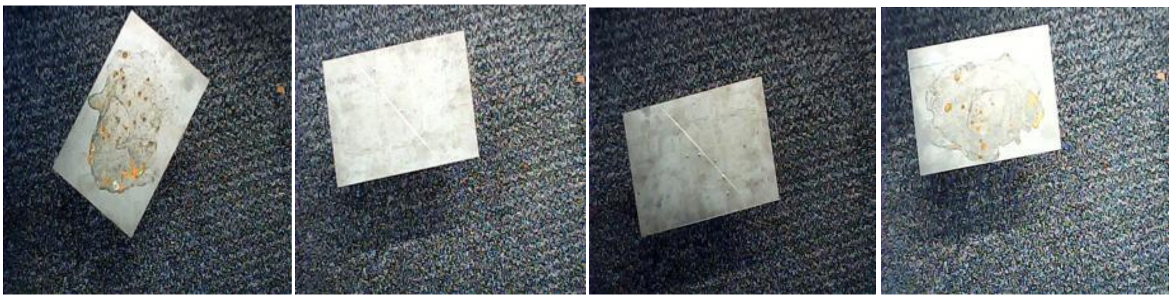


Figure 26.10 Sample of the inspection data

However, the CNN model requires a series of predefined parameters to optimise the generalisability and learning accuracy of the model. These parameters and hyperparameters are outlined as follows.

4.11.1 Model Components and Hyperparameters

The model hyperparameters of the CNN model used to investigate the sorting application includes the batch size, the number of epochs, learning rate, dropout, optimiser, activation functions, loss function, and evaluation metric. The vital components and hyperparameter set used are detailed in Table 4.. The dropout parameter for the model is set to 25% after the second hidden layer and 50% after the fourth hidden layer, which minimises the chances of overfitting. Furthermore, the Swish activation function was used in the hidden layers and a SoftMax activation at the output layer.

Table 4.3 Model I parameters and hyperparameters

Parameter/Hyperparameter	Value
Batch size	16
Epochs	50
Learning rate	0.005
Dropout	0.25/0.5
Optimiser	AdaMax
Activation	Swish and SoftMax
Loss	Categorical-crossentropy
Metrics	Accuracy

The planar object inspection data were used to train the VGGNet model using transfer learning, training from scratch and the newly developed model. The size of the available data is also important because the model does not exhaust the host device's memory. As there are approximately 5.5 Gigabytes (GB) of data for training and evaluation, the model data cannot fit into the computer random access memory (RAM); selecting an appropriate optimisation is necessary from the group of optimisation techniques discussed in Section 3.11.4. The batch gradient descent algorithm benefits from loading the data in

small fractions, often called mini-batch gradient, allowing the model to train all examples before updating the model parameters. The size of these mini-batches used is 16 images, and the model continues o loop over the samples until all the data is exhausted during the training.
Conversely, from the definitions in Section 2.4.3, the parameters $\theta,$ which represents the model's weights, are usually initialised before calling. The initialisation methods of the NN differ depending on the expected output and the different initialisation techniques discussed in Sections 3.11.4.

Nonetheless, transfer learning was first explored before developing a new architecture for comparison because the state-of-the-art models can perform the various classification tasks without significant modifications. Furthermore, the choice was informed by research that developing new architectures that compete against the current state-of-the-art is a very challenging task involving selecting numerous new hyperparameters and layer configurations [222].

4.11.2 Model Selection for Transfer Learning

The Visual Geometry Group (VGGNet) is one of the state-of-the-art deep CNN architectures selected to compare a transfer learning approach and training a model from scratch. In addition, the architecture was selected to explore transfer learning applications for remanufacturing because it is one of the first very deep CNN architectures that achieved state-of-art performance on large-scale datasets using replicated filters. The VGGNet architecture consists of the first two convolutional layers containing 64 and 128 respective 3 x 3 filters. The VGGNet uses a max-pool layer alongside a pool size of 2 and a stride of 2 in all the layers. The third, fourth and fifth layers have three convolutional layers with 256, 512 and 512 filters. The architecture also has the fifth and sixth layers as fully-connected layers. The fifth layer is flattened to produce 4096 units. The sixth fully connected layer contains eight (8) dense units and a SoftMax function used for classification. This model was selected and used to evaluate the initial inspection application.

4.11.3 Model Training and Evaluations

The model training involves optimising the model's weight parameters that ensure the best transfer of features from the inputs to the model's output. The training process ensures that the weight parameters are updated after each batch of the data passage. To achieve that, the model predicts the images in the batch, computes the loss value for those predictions given the actual data labels, and obtains the gradient of the loss function with respect to the model weights before updating the model weights by a minor factor in the direction opposite to the gradient.

Conversely, there are two broad methods of training deep learning models, including transfer learning and training from scratch. The process of training from scratch is a computationally more expensive technique than the transfer learning approach that uses pre-trained model weights trained with large datasets, including ImageNet and Microsoft COCO [75], [355]. However, to initiate the training process, the model is first compiled. The compilation involves the selection of an appropriate optimisation algorithm that ensures that the model updates itself based on the training data to improve its performance, a loss function that measures the distance between two probability distributions, including the predicted output and the true output, and finally a metric to evaluate the model performance [29]. The model training pipeline involves four vital stages that must be completed to learn the data features [29]. These stages are outlined in Figure 27 as follows.

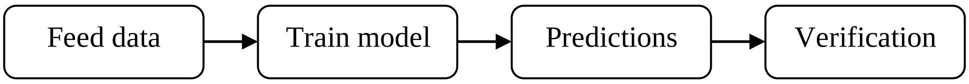


Figure 27 Model training pipeline

Moreover, the feature extraction begins the training process after the data is read into the model. It is a vital stage in traditional machine learning modelling that involves applying some computational algorithms to obtain the feature vectors that quantify the data. These vectors are hand-engineered in the traditional context and used to describe the contents of the presented input. However, there are numerous challenges to achieving effective feature learning, as outlined by researchers. These challenges include background clutter, illumination, occlusions, scale variation, viewpoint variation, deformation, and intra-class variation, among other factors [68], with various architectures proposed, especially CNN, to address these challenges, achieving optimal feature learning.

Nevertheless, effective training is obtained by optimising different modelling stages, including the data pre-processing, parameter initialisation, batch normalisation, architecture design, choices of activation functions, pooling techniques, regularisation techniques, and optimisation techniques. These considerations help to obtain the most optimised training for the model. The training process involves seven key stages highlighted as follows:

1) Get the batch from the training set.
2) Pass batch to the model.
3) Use backpropagation to calculate the model loss.
4) Use optimisation techniques to calculate the gradient of that loss function with respect to the model's weight.
5) Update the obtained weights using the gradient to reduce the loss.
6) Repeat the respective steps 1 to 5 above until one epoch is completed.
7) Repeat steps 1 to 6 above until the desired accuracy level is obtained.

Nevertheless, for a model with input x, weights W_1, W_2, biases b_1, b_2, layers L_1, L_2, and activation A, the training process involves using gradient descent to propagate the gradient of the loss forward and backwards through the model as follows

$$\frac{\partial \iota}{\partial W_1} = \frac{\partial L_1}{\partial W_1} \frac{\partial A}{\partial L_1} \frac{\partial L_2}{\partial A} \frac{\partial \iota}{\partial L_2} \tag{4.12}$$

The update of the weights with the learning rate σ is given by

$$W_1' = W_1 - \sigma \frac{\partial \iota}{\partial W_1} \qquad \textbf{4.13}$$

Nevertheless, the learning rate σ controls the speed of the model's training and obtaining an optimal learning rate is challenging. Therefore, minimal values of learning rates are used as guesses to train the network. However, It is important to note that very low learning rates might cause the model to freeze at some local minima. In contrast, high learning rates will likely cause poor convergence, making choosing an optimal learning rate a critical factor in successful training [68].

The AdaMax optimiser algorithm is selected with mini-batches, and the backpropagation is implemented to optimise the model's parameters. The number of epochs selected for training the model is based on heuristics. The model performance was observed, and the number of epochs was modified to achieve reasonably high prediction accuracy and a low error rate.

Nevertheless, the model predictions are a vector of the same length as the number of inputs, with the vector coefficient summing to 1 as the probability distribution is formed. The largest value in the probability distribution is the predicted class. The SoftMax function sums the entire vectors to 1 in a multi-class application, with the top probability score being the predicted class. This SoftMax is obtained from the relationship detailed in Section 3.11.2.4

$$\alpha(z)_k = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad j = 1, \ldots\ldots, K$$

Overall, the training minimises the model loss function, enhancing prediction accuracy. However, passing huge batch sizes is not recommended as it might be difficult for the data to fit into the computer RAM, thereby degrading performance. The pictorial representation of the learning process is shown in Figure 4.12.

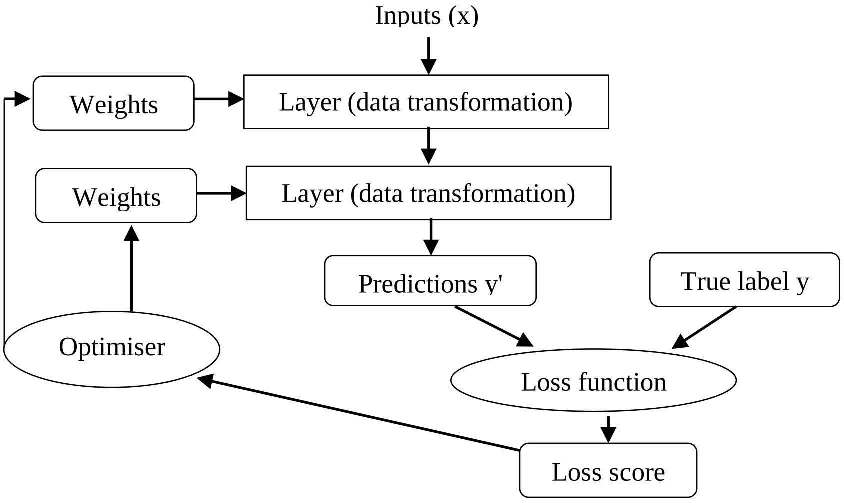


Figure 28 Pictorial representation showing model inputs, network, layers, loss function and optimiser

Nonetheless, the Keras callback function helps manage and make specific decisions during training. For example, it was helpful to save the model weights during training when the model accuracy improves and manage the training process if the training stops. The callback function also helps to specify the epoch you wish to restart the training [29].

I Training and Evaluation of the Pretrained Model

The pre-trained model considered was VGGNet architecture, once a state-of-the-art object recognition architecture with about 138.4 million parameters for an image size of 52 x 52 x 3. The 16-layer VGGNet model was selected and used to train the model using supervised learning. The model parameters and hyperparameters used for training and evaluating the VGGNet model are detailed in Table 4. and help train the model, including from scratch. The batch sizes reflect the number of images to load when the model performs forward and backward propagation of gradients which is highly computationally intensive work for the machine. In contrast, no gradients are propagated during evaluation, and the model can read many images.

Moreover, these models learn the data patterns during training, and the training process involves automatically extracting the samples' specific features to represent the image data patterns. The model uses the convolutional and pooling layers to learn the data patterns from the small, localised regions known as receptive fields. The first layer learns from the raw image pixels to motifs by detecting irregular edges to parts of objects, which are further combined in the hidden layers to reproduce the patterns as the underlying features [28]. During the training, the algorithm predicts the model's outputs after each epoch with two outputs computed for both the training and validation data, including training and validation accuracy and training and validation losses, with the model weights serialised after training. However, a critical aspect of modelling predictive systems is the evaluation stage. The evaluation requires estimating the model skill when predicting unseen data after training. Therefore, the performance of pre-trained models on the research data was first evaluated to visualise the goodness of fit for the remanufacturing applications, while the evaluation of the model was analysed to highlight the performance [331]. The performance of the pre-trained models was impressive, with the pre-trained VGGNet model producing a final training accuracy of 99.72% and a validation accuracy of 99.2% while classifying the faults on the metal bars. The results highlight that the model predictions the test data to a high accuracy level and generalises very well on the test data due to the high validation accuracy. Besides, generalisability is very important in evaluating the performance of predictive models.

Consequently, the model's training and validation errors are comparatively low since overfitting occurs when the validation error is significantly higher than the training error, as shown in Figure 4.129. These results highlight that the model has a low bias and low variance in predicting the classes of objects considered in the investigation.

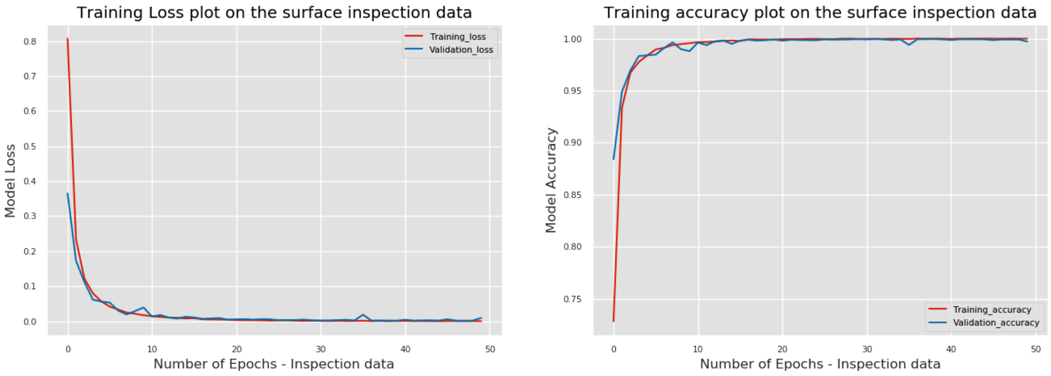


Figure 4.129 Pretrained VGGNet model performance on the surface inspection dataset

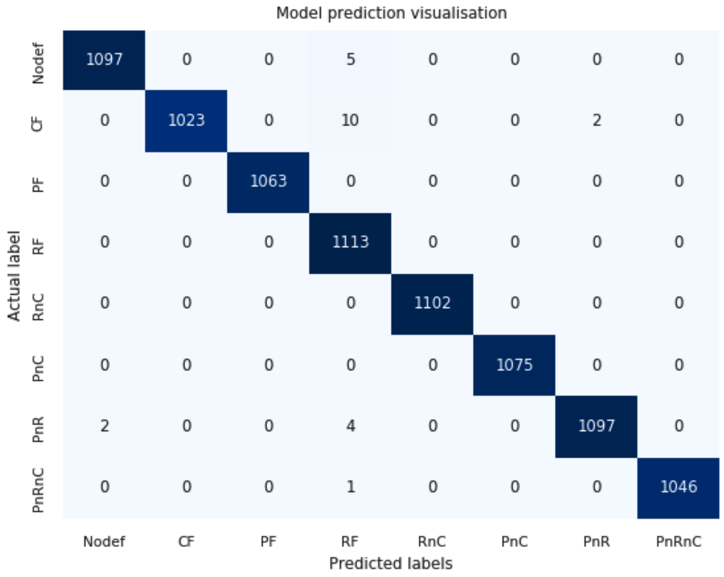
Furthermore, the pre-trained model misclassification defined in equation (2.80) is given by

$$M = 1 - Accuracy$$

$$M = 1 - 0.9972 = 0.0028$$

Overall, the pre-trained model produced a misclassification error of 0.28% on the inspection I data. These results highlight that the model can identify, inspect, and classify the products for remanufacturing with a very low error rate.

Nonetheless, the output is the testing is visualised using the confusion matrix, which shows the models misclassified twelve (12) crack defects to ten (10) pitting defects and two (2) pitting and rust defects, as outlined in Figure 304.14. Furthermore, another five (5) non-defective components were misclassified as rust defects, while six (6) pitting and rust defects were misclassified as four (4) rust defects and two (2) no defects. Finally, one pitting, rusting and crack-defected component was misclassified as rusting defects. These huge misclassifications highlight the pre-trained VGGNet model in its original form is inefficient to be deployed in a critical real-world remanufacturing application without improvement.


Model prediction visualisation
Actual label
Predicted labels

	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC
Nodef	1097	0	0	5	0	0	0	0
CF	0	1023	0	10	0	0	2	0
PF	0	0	1063	0	0	0	0	0
RF	0	0	0	1113	0	0	0	0
RnC	0	0	0	0	1102	0	0	0
PnC	0	0	0	0	0	1075	0	0
PnR	2	0	0	4	0	0	1097	0
PnRnC	0	0	0	1	0	0	0	1046

Figure 30 VGG model prediction visualisation using the confusion matrix

II Training from Scratch and Evaluation

The model training process from scratch was performed to understand the impact of training the model without considering the pr-trained weights. It is quicker and simpler to initiate as there are no existing model weight parameters to load; therefore, the pre-trained model was retrained using the same set of parameters. The training results from scratch produced a perfect prediction accuracy of 100% for training and validation and a zero-validation loss. In addition, the model's response shows similar low bias and low variance, as depicted in Figure 31.15.

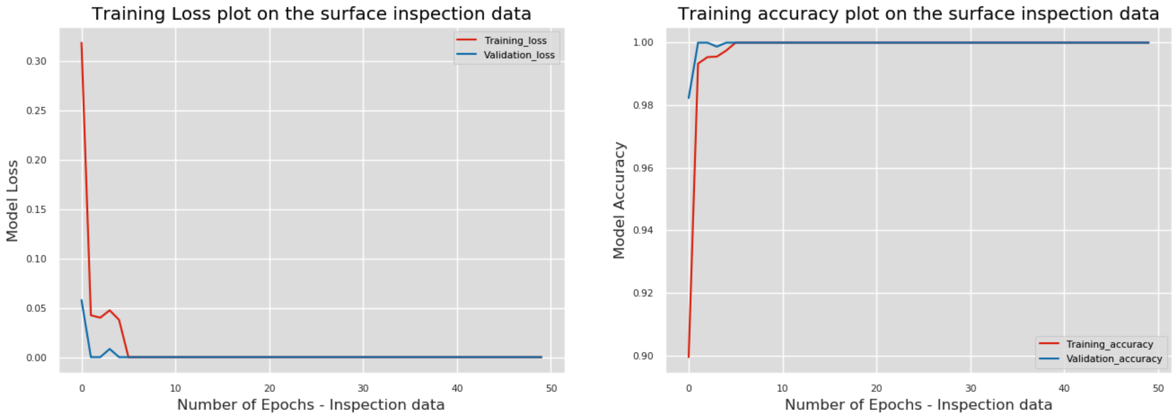


Figure 31 Training from the scratch loss and accuracy response on the VGGNet model

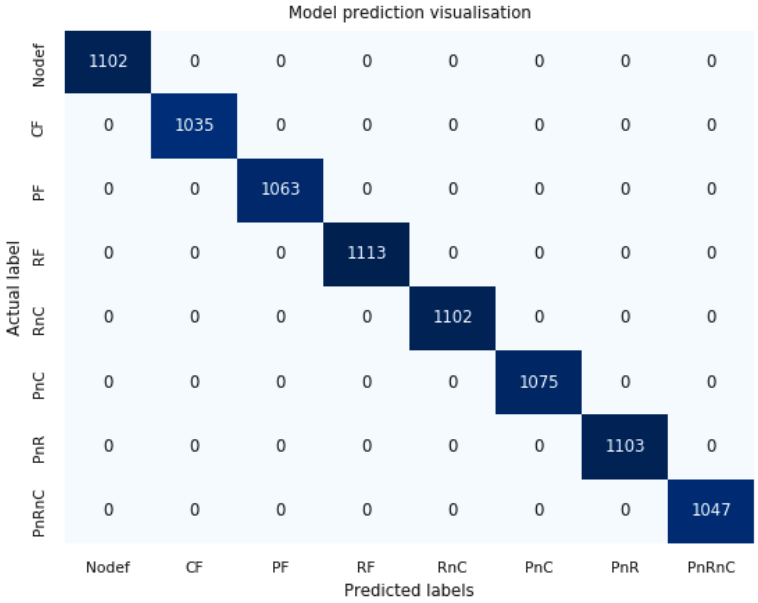
Moreover, the model training results highlight that the learning algorithm attained 100% training and validation accuracy before the 10th epoch, as shown in Figure 31.15. However, the model afterwards maintained the maximum performance, suggesting an excess capacity for the problem under investigation. Furthermore, the training and validation reached the lowest of zero before the 10th epoch and maintained the performance until the end, supporting the peak accuracy obtained during training. Finally, the zero-loss obtained also highlights that the model can generalise well on unseen data.

Furthermore, the VGGNet model misclassification from scratch is obtained as defined in equation (2.80) is obtained as

$$M = 1 - Accuracy$$

$$M = 1 - 1 = 0$$

Nonetheless, the testing output is visualised using the confusion matrix in Figure 326, which shows that the models achieved a perfect prediction across the entire test set, justifying the model's very high prediction accuracy.



	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC
Nodef	1102	0	0	0	0	0	0	0
CF	0	1035	0	0	0	0	0	0
PF	0	0	1063	0	0	0	0	0
RF	0	0	0	1113	0	0	0	0
RnC	0	0	0	0	1102	0	0	0
PnC	0	0	0	0	0	1075	0	0
PnR	0	0	0	0	0	0	1103	0
PnRnC	0	0	0	0	0	0	0	1047

Figure 32 Model prediction visualisation for the training from scratch using the confusion matrix

Consequently, the training of the VGGNet model produced a perfect prediction with no misclassification errors, thereby highlighting that the VGGNet model is likely too complicated for the dataset. Besides, the VGGNet had a wide architectural depth at its debut and produced a state-of-the-art performance in the ImageNet competition. Moreover, the model loss also shows a comparable low loss performance justifying the high model prediction accuracy observed after training the model. However, the perfect prediction results obtained from training from scratch account for the enormous computational cost of the training, which is a significant limitation as the model trained for more than seven hours compared to the transfer learning, which trained within an hour. Besides, an identified challenge of training from scratch was the prolonged training time, which increases the cost required to hire graphics processing units or rent a cloud-based GPU.

III Training the New Architecture and Evaluation

The model's training results in Figure 33.17 show that the mode accuracy peaked just before the 20th epoch and maintained a high accuracy over the 50 epochs that the model was trained. The newly developed model produced a top-1 training accuracy of 99.92% and a validation accuracy of 100%, highlighting the generalisability of the model on new data. Furthermore, the model training loss observed the lowest loss after about 15 epochs, with the loss stabilising to maintain the final training loss of about 0.0038 alongside the validation loss of 0.000013. These reported losses for the training and validation highlight that the model predictions follow the training accuracy, which peaked after about 17 epochs.

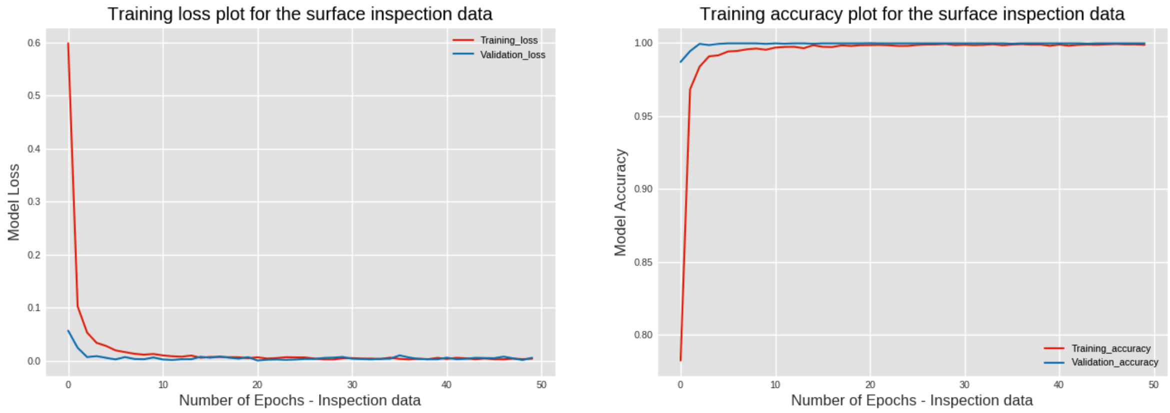


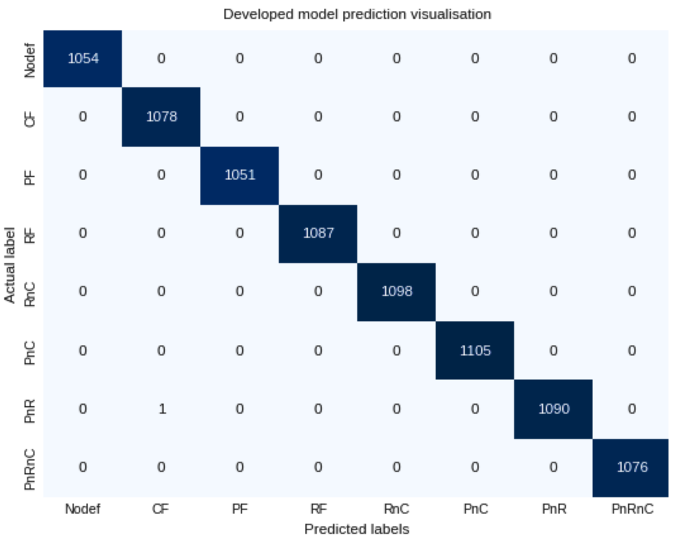
Figure 337 Researcher developed model response on inspection data

Furthermore, the misclassification error of the researcher developed model prediction is given by

$$M = 1 - Accuracy$$

$$M = 1 - 0.9992 = 0.0008$$

Moreover, the model's performance is significant as it produced a minor misclassification error of 0.08% on the torque converter dataset. Nevertheless, model testing is the final stage of deep neural network development before deployment. The testing involves loading the serialised model weights obtained after training and inferring the model performance using the separated test set. The model evaluation is obtained by computing and selecting the model's top predictions over the test images and comparing it against the actual value, obtaining the output as an un-normalised final model prediction. Finally, the test output is visualised using the confusion matrix in Figure 34.18. The table shows that the models misclassified a single pitting and crack defect to a crack defect in the entire predictions on the test set, justifying the very high prediction accuracy and low misclassification result produced by the model.



Developed model prediction visualisation

Actual label	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC
Nodef	1054	0	0	0	0	0	0	0
CF	0	1078	0	0	0	0	0	0
PF	0	0	1051	0	0	0	0	0
RF	0	0	0	1087	0	0	0	0
RnC	0	0	0	0	1098	0	0	0
PnC	0	0	0	0	0	1105	0	0
PnR	0	1	0	0	0	0	1090	0
PnRnC	0	0	0	0	0	0	0	1076

Figure 34 Developed model prediction visualisation using the confusion matrix

4.11.4 Results and Discussions

The deep learning modelling for remanufacturing process inspection highlights the necessary steps to model remanufacturing surface inspection using visual sensor data and deep convolutional neural network algorithms. The supervised modelling approach effectively represented the remanufacturing inspection, attaining a satisfactory performance across the two algorithms and three individual cases tested, including the pre-trained model, training from scratch, and using the researcher-developed algorithm. In addition, the top-1 prediction accuracy was used to evaluate the performance of the algorithms.

Besides, the challenges of obtaining rich features from the model highlighted by researchers [68] were addressed from the beginning of the model development. The measures and techniques used to address them include background clutter and illumination minimised by capturing the data on the conveyor system and in the actual work environment lighting, thereby ensuring that the images used to train the model will look similar during testing. Furthermore, the occlusion challenge was addressed by timing the product's arrival speed to the point of inspection, ensuring that the camera viewpoint is restrained to one object at a time.

Furthermore, scale and viewpoint variations were minimised using data augmentation, a low-level approach to introduce various transforms to the object before passing them to the model. Data augmentation primarily enhances the model's generalisation ability as the model sees slightly modified input versions of the input data. A significant feature of the augmentation class is that the data labels are not changed during the process. A utility to facilitate data augmentation is the Keras ImageDataGenerator, which accepts data, transforms them randomly, and returns the newly transformed data. The possible transforms used in computer vision augmentation include translation, rotations, horizontal flips, vertical flips, changes in scale, shearing, etc. Finally, the transformed data were used to train the models and evaluated on the unmodified test samples, which showed consistently high test accuracy with a noticeable reduction in the training accuracy. In summary, model generalisation was vital when training deeper neural network architectures and helped to improve model performance on unseen data.

Nonetheless, an ideal model should have a very high training and validation accuracy and low training and validation errors. These properties ensure the model performs well predicting new data and generalising well on unseen data. Nevertheless, the comparison of the model performance on the research data highlights that the training from scratch (VGGS) model produced the highest and maximum prediction accuracy of 100% on the surface inspection data. The newly developed model, alongside the VGGNet transfer learning models, produced a top-1 accuracy of 99.92% and 99.67%, respectively. These highlights that the models can effectively perform the desired inspection and further generalise to unseen data as the validation accuracy was significantly high across the three applications.

These performances were coherent across the three models, producing a low validation error on the test data.

However, despite the success of the pre-trained models, they have the limitation of high memory demands. The memory demands remain a vital challenge to deploying pre-trained models in remanufacturing and other applications that run on devices with limited storage. Therefore, it calls for exploring models that can achieve comparable results with smaller capacities. This exploration is worthy because the algorithms have many training parameters, making them significantly challenging to deploy in smaller applications with memory constraints. Nevertheless, it is worth outlining that the research-developed architecture has 1.423 million parameters making it far less model to train than the VGGNet architecture, which has 138.4 million parameters. These model parameters account for the required storage capacity of the trained model weights and the training time required to train and deploy the models.

4.12 Adapting the Developed Model to Torque Converter Component Inspection

This application considers the post-cleaning inspection in remanufacturing the torque converter units, ensuring that the components are appropriately dried after cleaning. The application assumes that the components are properly cleaned. The model is developed based on the process structure in the data collection facility, as there are no automated methods of returning the improperly cleaned parts to the cleaning system. The crucial application of the model helps to eliminate the need for expert inspection of the cleaning process, thereby enhancing productivity through the 100% guaranteed inspection provided by design. After cleaning, the model differentiates the dry and wet samples, named dry sample (DS1), DS2, DS3, DS4, DS5, Wet1, Wet2, and Wet3. A batch of the original samples used to train the model is depicted in Figure 35.19.

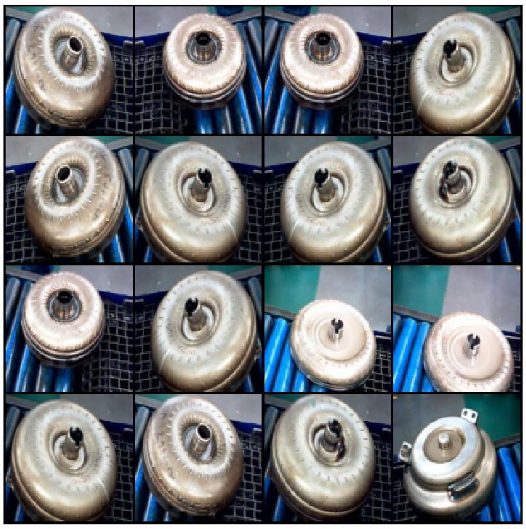


Figure 35 A batch of samples for inspection II application used to train the model

4.12.1 Experiment and Model Training

Adapting the developed model involves using the developed architecture without any modification since the product surface inspection data has the same number of classes and is in the same format. The data consist of eight (8) categories of 3578 images per class, making up the 28624 images, helpful in evaluating the adaption of the developed model to achieve automated component inspection during remanufacturing. The developed model was trained on the new remanufacturing post-cleaning inspection data of the torque converter components. The training adopted the supervised learning technique, where the data and labels were used to train the deep convolutional neural network model. The training tunes the weight parameters that enhance the model's performance using the model hyperparameters outlined in Table 4. and saved for future inference.

Furthermore, the training incorporates the AdaMax optimiser, a categorical cross-entropy loss. In addition, a small amount of dropout is applied to the hidden layers of the model, 25% and 50% in the first and second fully connected layers, to minimise overfitting, enhancing robustness. Besides, the saved file is stored in h5 format, a hierarchical data format (HDF) format used to store multidimensional arrays of scientific data. The h5 format enables the storage of the following model components, including the architecture, sets of weights, optimisers, loss and metrics used for model evaluation [356].

4.12.2 Results and Discussions

The modelling results in Figure 36.20 show that the deep learning technique can achieve surface inspection in remanufacturing applications, attaining significantly high performance using supervised learning. Notably, the researcher-developed architecture performs comparably to the state-of-art VGGNet model, with significantly lesser computational demands. The model produced a significant performance comparable to the initial test case, with a top-1 training accuracy of 99.97%. In addition, a validation accuracy of 100% was obtained, highlighting that the model generalises well on the new data. The results highlight that the model performs well on training and test data, guaranteeing good prediction and generalisation of unseen data.

Furthermore, the model's misclassification error is given by

$$M = 1 - Accuracy$$

$$M = 1 - 0.9997 = 0.0003$$

It is crucial to outline that the model's performance is impressive as it produced a misclassification error of 0.03% on the torque converter dataset, which is negligible.

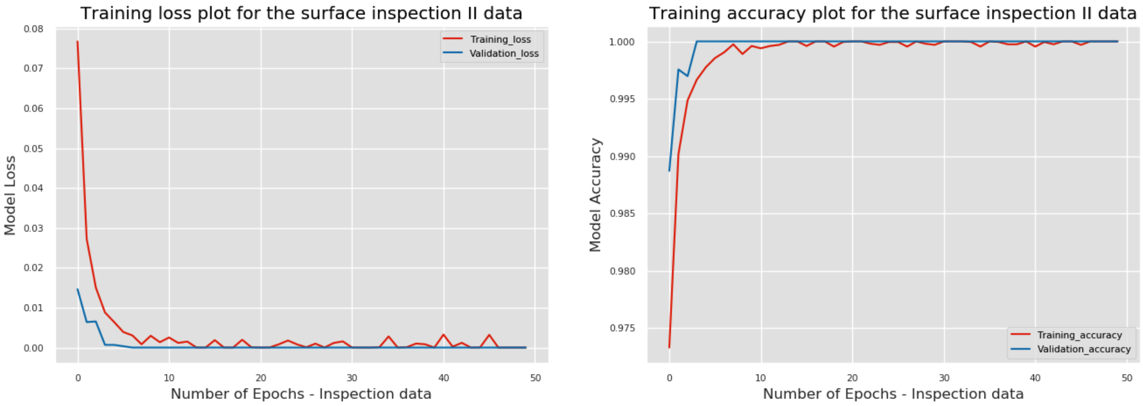
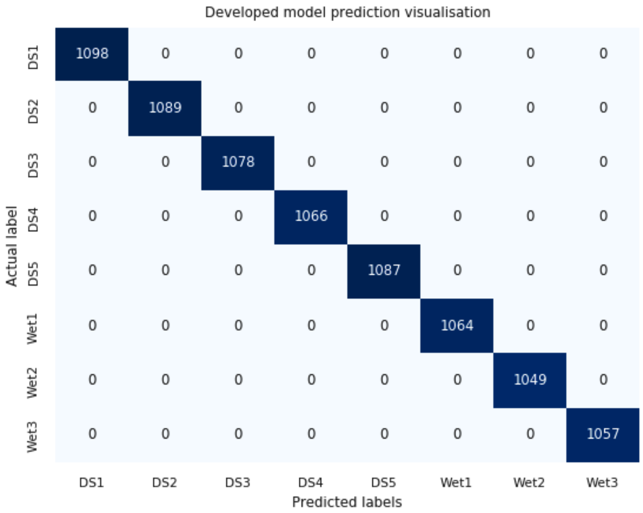


Figure 36 Model training results on the Torque Converter surface inspection

Conversely, the model performance visualisation results using the confusion matrix in Figure 371 highlight that the samples' predictions conformed with the obtained prediction accuracy of the Torque Converter inspection application, with no observable misclassifications.


Developed model prediction visualisation

Actual label \ Predicted labels	DS1	DS2	DS3	DS4	DS5	Wet1	Wet2	Wet3
DS1	1098	0	0	0	0	0	0	0
DS2	0	1089	0	0	0	0	0	0
DS3	0	0	1078	0	0	0	0	0
DS4	0	0	0	1066	0	0	0	0
DS5	0	0	0	0	1087	0	0	0
Wet1	0	0	0	0	0	1064	0	0
Wet2	0	0	0	0	0	0	1049	0
Wet3	0	0	0	0	0	0	0	1057

Figure 37 Inspection II model predictions visualisation

Besides, the inference from the connected camera focuses on a conveyor carrying cleaned end-of-life products for post-cleaning inspection. First, the surfaces are automatically inspected using the model and the connected camera to verify the surface dryness. Furthermore, the model predictions are visualised using OpenCV, a computer vision library that supports the execution of deep learning models. Then, the data is loaded, converted to an array, resized, and the serialised weights to predict the inputs. Finally, the model predictions were performed, and the predictions were displayed with custom texts, including the predicted class, position, text colour, and size inserted in the algorithm. In addition, the inspection model predictions from the live video feed outline the model's impressive performance, with the respective class predictions displayed in Figure 38.

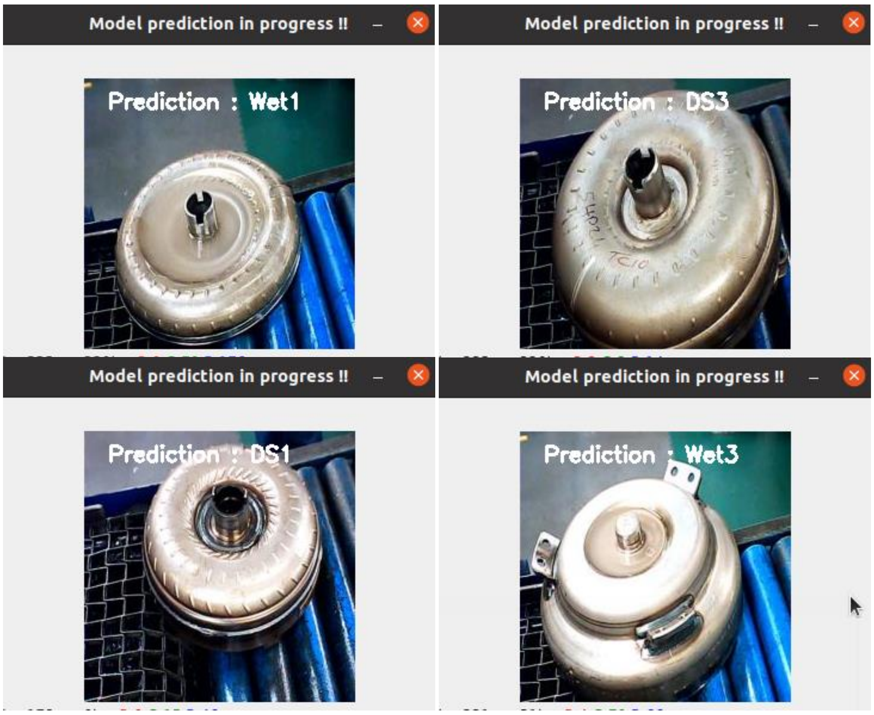
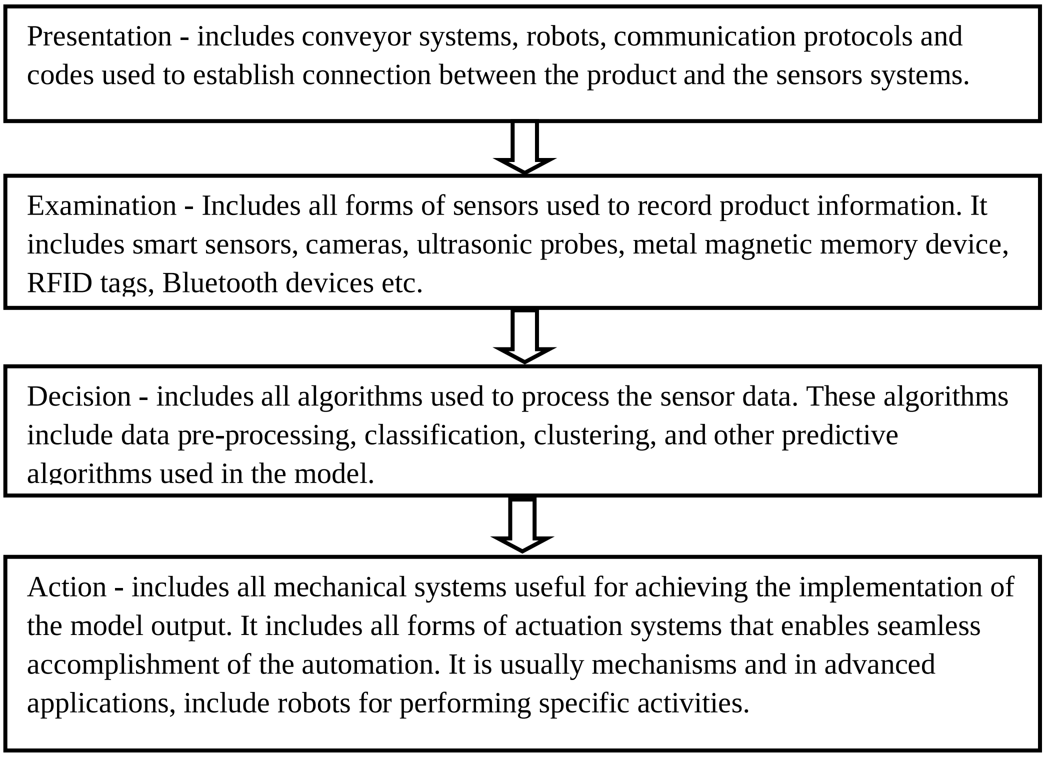

Model prediction in progress !!
Prediction : Wet1
Model prediction in progress !!
Prediction : DS3
Model prediction in progress !!
Prediction : DS1
Model prediction in progress !!
Prediction : Wet3

Figure 38 Model inspection predictions result from the connected camera inputs

These results highlight that the modelling approach effectively achieves surface inspection in remanufacturing. Furthermore, the method helps predict the surface conditions of the product using the camera data as inputs, thereby validating the effectiveness of the design and modelling to achieve surface inspection in remanufacturing.

4.13 Extending the Deep Learning Modelling to Achieve Automated Inspection

Sections 4.11 and 4.12 have outlined the various algorithms and techniques for achieving surface inspection in remanufacturing using two different surface-defected data. However, there are still subsurface defects that are inherent in some products as they return for remanufacturing. Therefore, the model's extension towards a holistic, automated inspection considers the inspection of surface and sub-surface defects on components and proposes a structured approach to achieving automated inspection in remanufacturing. The approach is described as a framework for achieving automated inspection in remanufacturing, summarised in four vital steps: presentation, examination, decision, and action stages. This sequence of activities is depicted in Figure 39.23.



Presentation - includes conveyor systems, robots, communication protocols and codes used to establish connection between the product and the sensors systems.

Examination - Includes all forms of sensors used to record product information. It includes smart sensors, cameras, ultrasonic probes, metal magnetic memory device, RFID tags, Bluetooth devices etc.

Decision - includes all algorithms used to process the sensor data. These algorithms include data pre-processing, classification, clustering, and other predictive algorithms used in the model.

Action - includes all mechanical systems useful for achieving the implementation of the model output. It includes all forms of actuation systems that enables seamless accomplishment of the automation. It is usually mechanisms and in advanced applications, include robots for performing specific activities.

Figure 39 Automated inspection design approach.

The presentation stage involves getting the products and components to the inspection point alongside enabling the inspection process. This step enhances automated inspection by using a conveyor transportation system and, where necessary, a robotic arm to enhance proper product inspection through picking and rotation to achieve a full view of the object. Furthermore, the examination stage evaluates the conforming features obtained through connected sensors' recordings. It uses sensors to collect data about the conditions of the components alongside historical data obtained from the products MoL. The sensors include visual, non-visual, ultrasonic, and other sensors that can detect internal defects in components. Besides, the decision stage can be likened to a binary decision to accept or reject the component or product, achieved using the learning algorithm. Therefore, the decision stage is critical in achieving automated inspection and forms the basis of modelling using deep learning algorithms. Lastly, the action stage uses an actuation system to activate the next remanufacturing sub-processes to sort, inspect, control, or exit the process.

However, to achieve the full automated inspection for remanufacturing, the examination stage is modified to include sensor systems that can assess surface and sub-surface defects on components, alongside any historical data about the product usage during the MoL. These modifications were investigated with the design for automated inspection proposed as the framework for achieving automated inspection in remanufacturing.

The proposed design for automated inspection captures the requirements to automate vital processes in remanufacturing using deep learning-based models and other associated technologies. The design for automated inspection (DfAI) model evaluates the automated inspection in remanufacturing, which differs from the automated inspection in manufacturing, most notably to evaluate and identify inherent internal properties of a product to undergo a new life cycle without failure. In addition, the integrity test is not always applicable in automated inspection in manufacturing, where the wear and strength properties of products are not verified before the product is returned for reuse during remanufacturing, thereby making the DfAI in remanufacturing a vital tool to enhance productivity.

The DfAI system details the integration of a product's historical data and status data to achieve automated inspection by inspecting the product's surface and sub-surface defects. The connected visual sensor systems detect surface defects, while non-destructive inspection (NDI) methods identify the sub-surface defects. In addition, other inherent usage information obtained from the IoT system is fused with the NDI and visual inspection results to obtain a holistic inspection system that can perform at every stage of remanufacturing, including the reverse logistic stage.

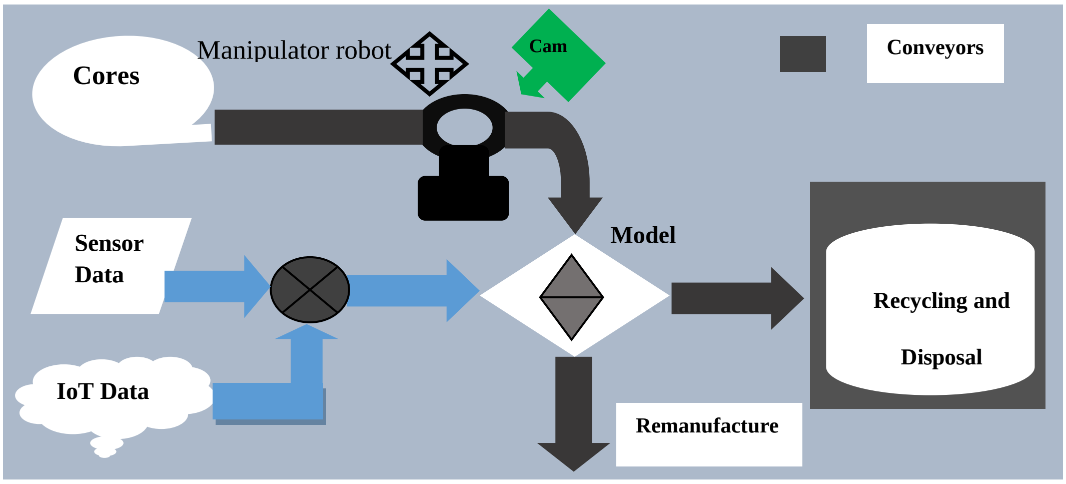


Figure 40 High-level design approach of the design for automated inspection

The vital components of the design for the automated inspection system in Figure 40.24 are summarised as follows:

- Cores - products returned for remanufacturing.
- Conveyors - The transport system of the products to the point where remanufacturing starts.
- Manipulator - Robot device used to achieve multiple viewpoints for the fixed visual sensor.
- Camera - A visual inspection sensor to record images/videos of products.
- Sensor data - The ultrasonic inspection sensor from sub-surface defected components
- IoT Data - The product's historical MoL (usage) data.
- Storage - Storage consists of products for recycling and disposal.
- Remanufacturing - Subsequent stages of product remanufacturing after acceptance.
- Model - The decision-making unit of the automated inspection model.

Nevertheless, the inspection faults anticipated by the model are pre-determined at the model design stage, and it helps in selecting the appropriate sensing devices and sensors used for recording the product conditions, which are labelled for use as training samples. Furthermore, the DfAI system involves six key stages: data loader, pre-processing, modelling and model selection, training, evaluation and improvement [28], [30], [103].

First, the data loader refers to the process of reading the data into the model. At the same time, the pre-processing step involves resizing the data to suit the model architectural requirements and partitioning the data into training, validation, and test sets, obtained as a percentage of the entire data and used for evaluating the model performance. Besides, the model is a computational algorithm consisting of multi-layer CNNs. The modelling involves creating the types of input, outputs, the type, and the number of layers, specifying the stride, padding, selection of performance metrics, etc. It also defines how the computational model processes the input and output. Finally, the model selection consists of choosing a state-of-the-art model and performing transfer learning using new data. Moreover, this approach to visual inspection modelling has successfully identified seven visible fault types, including rusting, crack, pitting, and other combination faults. The samples were recorded, pre-processed and used to train a deep convolutional neural network model for remanufacturing inspection [103], [344].

Furthermore, another crucial stage of non-destruction inspection is determining the products' structural integrity. The structural integrity test guarantees that the product for reuse can work optimally for another life cycle without failure, thereby making the test for components for reuse a vital part of remanufacturing. It ensures the product's internal characteristics are assessed before providing quality assurance. The design for testing in remanufacturing has been proposed and acts as a mathematical framework for achieving NDT inspection as a remanufacturing design consideration. The design for testing considers the components' shapes and data acquisition geometry to provide the advantage of enhancing in-service inspection and increasing the range of components suitable for remanufacturing [357].

Moreover, the sub-surface inspection involves adopting the design-for-testing method like the multi-objective optimisation that focuses on maximising the coverage of ultrasonic fields throughout components while also minimising the number of ultrasonic transducers used for the assessment. Researchers investigated the practical implementation of the design-for-testing to understand the optimal placement of ultrasonic sensors on the product boundary and the ultrasonic field coverage area using the PZFlex software [357], thereby confirming the product's shape is a vital design consideration. Furthermore, the design for testing enhances non-destructive testing, thereby improving integrity and certifying the quality of products after remanufacturing.

Nevertheless, internet-connected devices, also known as the internet of things (IoT), is another essential technology to enhance remanufacturing by providing connectivity and interaction between the cyber world and the physical devices. It allows for the recording and analysis of historical data about product conditions to make data-driven decisions. Recent advancements in hardware, software, and communication technologies have advanced IoT-connected sensing devices to provide observation and data measurements from the physical world [358]. The technology uses a more recent design of embedded chips to record product health data during use and decides the product conditions at end-of-life. The IoT data includes the middle-of-life data incorporated in the pre-disassembly inspection, which are only available to OEMs that remanufacture their products at end-of-life. Perhaps, since these MoL data are not readily available to third-party remanufacturers for almost every product for remanufacturing, making these data publicly available would enhance the design of automated inspection systems for remanufacturing applications.

Conversely, the design for automated inspection uses the photogrammetry approach requires the optimal camera placement for adequate coverage of the objects using the triangulation of multiple viewpoints [359], with the highest model prediction from the respective camera outputs returned as the final model prediction. The method was not implemented directly due to cost constraints. However, the implemented DfAI system incorporated a single camera with multiple low-level transformations, including rotations, flipping, width and height shifts, scaling etc., introduced to enhance the model's generalisation ability during training [360].

4.13.1 Benefits of Design for Automated Inspection

The benefits of the design for automated inspection include enhancing throughput, product inspection accuracy, and reducing workplace hazards associated with remanufacturing processes, alongside factory lead times [361]. These benefits are achievable using the DfAI approach, enhancing the remanufacturing process. Automating the remanufacturing processes, especially the inspection stage, reduces the complications in the process, alongside the efficient reuse of materials and components, thereby reducing the non-remanufacturable parts and waste, improving value recovery and quality assurance of remanufactured products and reducing remanufacturing cost [362]. The DfAI provides the platform to explore new process improvement methods and improve inventory management. These benefits have been outlined using a discrete event simulation of an automated inspection system for remanufacturing [361]. The DfAI framework addresses the following critical issues in remanufacturing

- Process requirements for achieving and deploying an automated inspection in remanufacturing.
- Outlines the high-level hardware setup requirements for automated inspection in remanufacturing.
- Understand the situational considerations for achieving automated remanufacturing inspection.
- Expand the methodologies for achieving automation inspection in remanufacturing.
- Enhance the understanding of the different levels of inspection in automated inspection.
- Apply deep learning techniques for automating processes in the remanufacturing industry.

In summary, the most significant advantage of the DfAI is that it can be incorporated into existing remanufacturing processes by re-engineering the systems, thereby creating new systems and designs for automated inspection.

4.14 Chapter Summary

This chapter presented the modelling and development approach, including the research design, frameworks, model dependencies, data preparation and pre-processing, computational model design and development considerations, training, and evaluation. It further outlined the deep learning-based approach for surface inspection applications in remanufacturing. It presented two different inspection cases and compared the performance of three models, including the pre-trained VGGNet, training from scratch, and the newly developed model. The model's performance suggests that the developed models can also be used at the initial point of product collection to assess the condition of components on arrival. The chapter has successfully addressed research questions (Q2) by modelling remanufacturing inspection using the developed algorithm, thereby automating the process and improving efficiency.

Besides, these inspection applications have already been helpful in the industrial post-cleaning inspection of torque converter remanufacturing. The application is new knowledge in the form of a learning algorithm that can be used in automated inspection processes in remanufacturing alongside the proposed design for the automated inspection framework. Furthermore, the chapter also presented a holistic, automated inspection technique achieved by extending the developed deep learning-based inspection application to include assessing and detecting sub-surface faults. The framework named design for automated inspection (DfAI) is another tool that helps remanufacturers quickly identify the automated inspection setup requirements.

CHAPTER FIVE

MODELLING COMPONENT SORTING AND PROCESS CONTROL IN REMANUFACTURING USING DEEP LEARNING

5.0 Introduction

This chapter focuses on adapting the researcher's deep learning algorithms to the remanufacturing sorting and process control applications. It addresses the research question (Q4), where the developed model is adapted in other remanufacturing applications. The chapter presents the design modification of the developed convolutional neural network model to achieve remanufacturing sorting and process control. The training and evaluation are explored on EoL products during remanufacturing, with the in-case results and analysis evaluated for proper deductions.

5.1 Modelling Sorting in Remanufacturing Using Deep Learning

The modelling approach for the sorting application is developed as a multiclass classification problem that performs a classification as positive or negative for the respective classes. The multiclass classification differs from the binary classification problems modelling across other applications, including the model's loss function and the output, which is a SoftMax function. These parameter modifications are performed on the learning algorithm before training and evaluating the model. The sorting application is modelled as a supervised learning problem using labelled data. It explores deep convolutional neural networks for developing sorting systems to categorise remanufacturing products and components. Nevertheless, the sorting application uses the deep learning recognition application, where specific parts are identified from the images of the components taken by a connected camera system. The typical pipeline of a machine learning model is depicted in Figure 415.1. It consists of the camera unit used to obtain data and a pre-processing unit that prepares and presents the data in the model format. During this training, the model learns the patterns in the data and the testing stage, where predictions are evaluated as the model output and finally improvement stage, where the model performance is enhanced after evaluation.

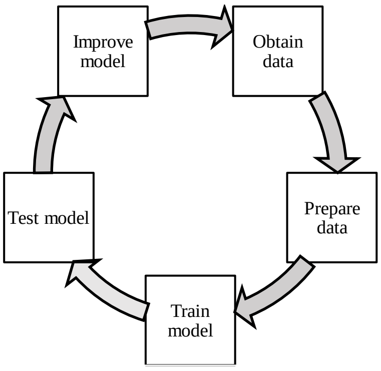


Figure 41 Typical learning model block diagram

5.2 Sorting Application and Data

The investigation of the use of DCNN for automated sorting in remanufacturing consists of an object recognition system that aims to identify objects in the video stream. These identified objects are afterwards sorted into categories using the designed system. The sorting technique can be adopted for pre-disassembly, post-disassembly, and other operations during or after remanufacturing. The existing sorting techniques have been discussed extensively in Section 3.12.2 alongside the limitations of the existing methods. Finally, the development of the deep CNN-based sorting system for remanufacturing applications is presented.

The sorting application considers the torque converter system components recorded during this research. The experiment investigates the possibility of detecting faults as the intended sorting solution is based on fault recognition. The distance between the camera and the objects on the conveyor system was limited to approximately 40" to ensure that the camera's coverage was restricted to one object at a time using the existing programmable logic controller's time delay. As lighting contributes to visual sensing, the recording was made in an industrial work setting to reduce lighting effects after development.

The data consist of twenty (20) object categories of the torque converter units, including the dampers, stators, housing, impellers, turbine, pressure plates and a whole torque converter system. The twenty (20) object classes corresponding to each object under consideration are Damper1, Damper1, Damper3, Housing1, Housing2, Housing3, Impeller1, Impeller2, Impeller3, PressurePT1, PressurePT2, Reman1, Reman2, Reman3, Stator1, Stator2, Stator3, Turbine1, Turbine2, Turbine3. The sample mini-batch is shown in Figure 42, which highlights a single batch of 16 images read by the model during training. The mini-batches were read until all the samples were taken into the model to complete the single epoch and afterwards repeated until the total epochs were covered.

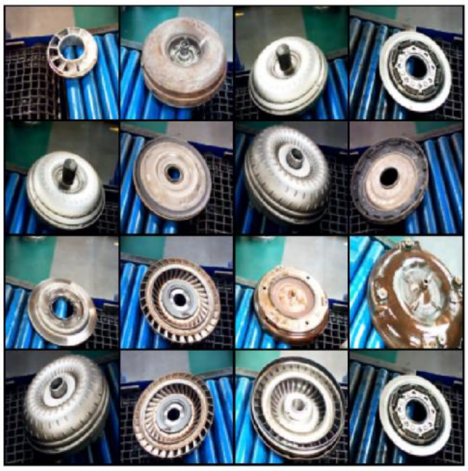


Figure 42 One batch of the original torque converter samples used to train the model

5.3 Model Parameters / Hyperparameters and Modification

The architectural modification for the CNN model used for the inspection application is necessary to adapt the developed model to perform component inspection in remanufacturing. The modified architecture used in the experiment is described in Table , highlighting the number of filters used in the input, hidden, and output layers of the model alongside the expected number of inputs and outputs specified as the target in the output (SoftMax) layer. The architecture consists of four convolutional layers of filter sizes 64, 48, 36 and 20 and two fully-connected layers of 512 and 20 filters. The number of model parameters amounts to 1,792,908 learnable parameters, which approximates 1.8 million parameters learnable during training. The architectural modifications and the calculations are outlined in Table , showing the shape modifications as the model learns the patterns in the data alongside the total number of parameters learned at each stage of the transformation.

Table 5.1 Model architecture optimised for the sorting application

Layer Type	Output shape	Activation size	Parameters
Input	(None, 52,52,3)	8112	0
Conv2D	(None, 52,52,64)	173056	1792
Activation (Swish)	(None, 52,52,64)	173056	0
Conv2D	(None, 52,52,48)	129792	27696
Activation (Swish)	(None, 52,52,48)	129792	0
Maxpooling	(None, 26,26,48)	32448	0
Dropout	(None, 26,26,48)	32448	0
Conv2D	(None, 26,26,36)	24336	15588
Activation (Swish)	(None, 26,26,36)	24336	0
Conv2D	(None, 26,26,20)	13520	6500
Activation (Swish)	(None, 26,26,20)	13520	0
Maxpooling	(None, 13,13,20)	3380	0
Dropout	(None, 13,13,20)	3380	0
Flatten	(None, 3380)	3380	0
Dense	(None, 512)	512	1731072
Activation (Swish)	(None, 512)	512	0
Dropout	(None, 512)	512	0
Dense	(None, 20)	20	10260
Activation (SoftMax)	(None, 20)	20	0

Consequently, like the original architecture, the width $\boldsymbol{m}$, the height $\boldsymbol{n}$, the number of filters in the previous layer $\boldsymbol{d}$, the bias term $\boldsymbol{b}$ and the number of filters in the current layer under consideration $\boldsymbol{k}$, all contribute to estimating the number of parameters in the model. The number of parameters $\boldsymbol{P}$ of the model is obtained using the kernel size of [3,3], which represents $\boldsymbol{m}, \boldsymbol{n}$, channels in input $\boldsymbol{d}$ =3, and current layer $\boldsymbol{k}$ = 64 [352]. The layer-specific parameters of the model are obtained using equation **4.12** as follows.

$$P = \left((m * n * d) + 1\right) * k$$

$$Layer\ 1 = \big((3 * 3 * 3) + 1\big) * 64 = 1792$$

$$Layer\ 2 = \big((3 * 3 * 64) + 1\big) * 48 = 27696$$

$$Layer\ 3 = \big((3 * 3 * 48) + 1\big) * 36 = 15588$$

$$Layer\ 4 = \big((3 * 3 * 36) + 1\big) * 20 = 6500$$

However, the fully connected are modified to suit the data for the remanufacturing sorting application with twenty classes; therefore, the layer parameters are determined using the activation size of the model using the relationship of equation **4.13**.

$$P = current\ layer\ activation * previous\ layer\ activations + current\ layer\ activations$$

$$Layer\ 5 = 512 * 3380 + 512 = 1731072$$

$$Layer\ 6 = 20 * 512 + 20\ = 10260$$

The total number of parameters becomes the sum of the respective layer parameters of the model, giving the 1,792,908 parameters. The overall architecture for implementing the deep learning inspection application in remanufacturing is attached as Appendix 2B.

5.4 Experiment and Model Training

Adapting the developed model for sorting in remanufacturing involves the modification of the developed architecture to suit the data for the sorting application. The data consist of twenty (20) categories of 3578 images per class, making up the 71560 images, helpful in evaluating the adaption of the developed model to achieve automated component sorting during remanufacturing. Besides, the developed model is a multilayer architecture consisting of filters and other components useful for learning patterns in each data. A supervised learning approach is used to train the model, where data and labels are required to train the models.

The training process is similar to the other applications where the model incorporates an optimiser that updates the model parameters, a scoring function that compares the predictions and true values and a metric to assess the performance, thereby obtaining the model's optimal weight and bias parameters after multiple iterations over the train set. These model layers must be compiled before the training is initiated. The compilation is achievable using compile method in Keras, which builds the model and is ready for training. The training process optimises the weight parameters that enhance the model's performance using the model hyperparameters outlined in Table 4..1, after which the model is saved for future inference after training. In addition, the model incorporates an AdaMax optimiser, a categorical cross-entropy loss, and the accuracy metric used to score the model's performance. Similarly, a 25% dropout is applied to the first fully-connected layers and a 50% dropout to the last fully-connected layers to minimise overfitting during training.

5.6 Results and Discussions

The evaluation involves performing predictions using the serialised model weights, passing the unseen test data and labels to the model, and evaluating the performance. The evaluation was performed across two stages: the test set and the live feed from a connected camera to return the number of correctly classified images. The evaluation results outline that the model features achieved high performance.

The model's training results in Figure 43 show that the accuracy peaked just after the 10th epoch and maintained high performance over the 50 epochs used to train the model; however, the performance was slightly degraded. Nevertheless, the sorting model accuracy of 99.99% was obtained during training alongside a validation accuracy of 100%. Moreover, the model loss obtained was negligible since the model's training, and validation losses were almost zero after the 40th epoch, supporting the model's high prediction results.

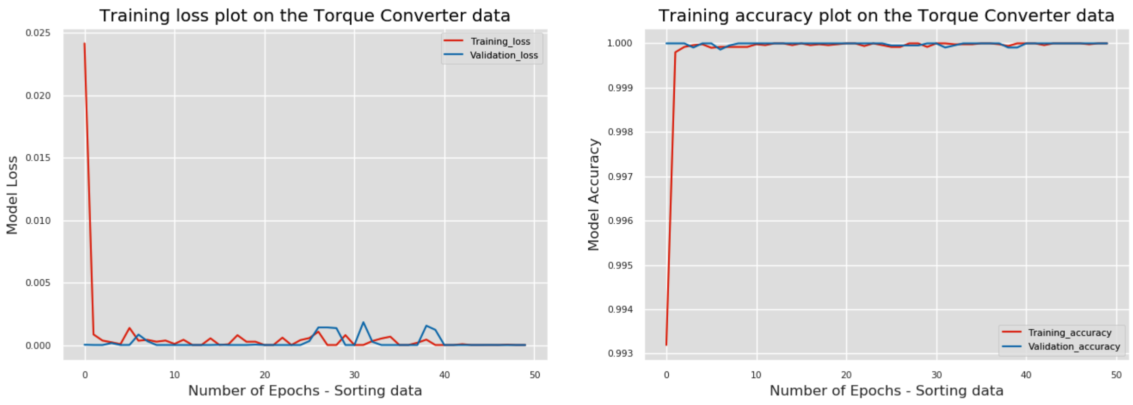


Figure 43 Sorting model training and validation responses

The misclassification of the researcher developed model prediction adapted for the sorting application is given by

$$M = 1 - Accuracy$$

$$M = 1 - 0.9999 = 0.0001$$

Conversely, the sorting misclassification of 0.01% obtained from the model predictions suggests that the model successfully sorted the Torque Converter components into categories.

Moreover, inference involves making predictions on a model using the serialised model weights. Finally, the model inference step includes passing the unseen data on the model to evaluate its generalisation ability. The test output was visualised using the confusion matrix in Figure 44, showing no misclassification in the sorting application, justifying the model's high prediction accuracy and low misclassification result.

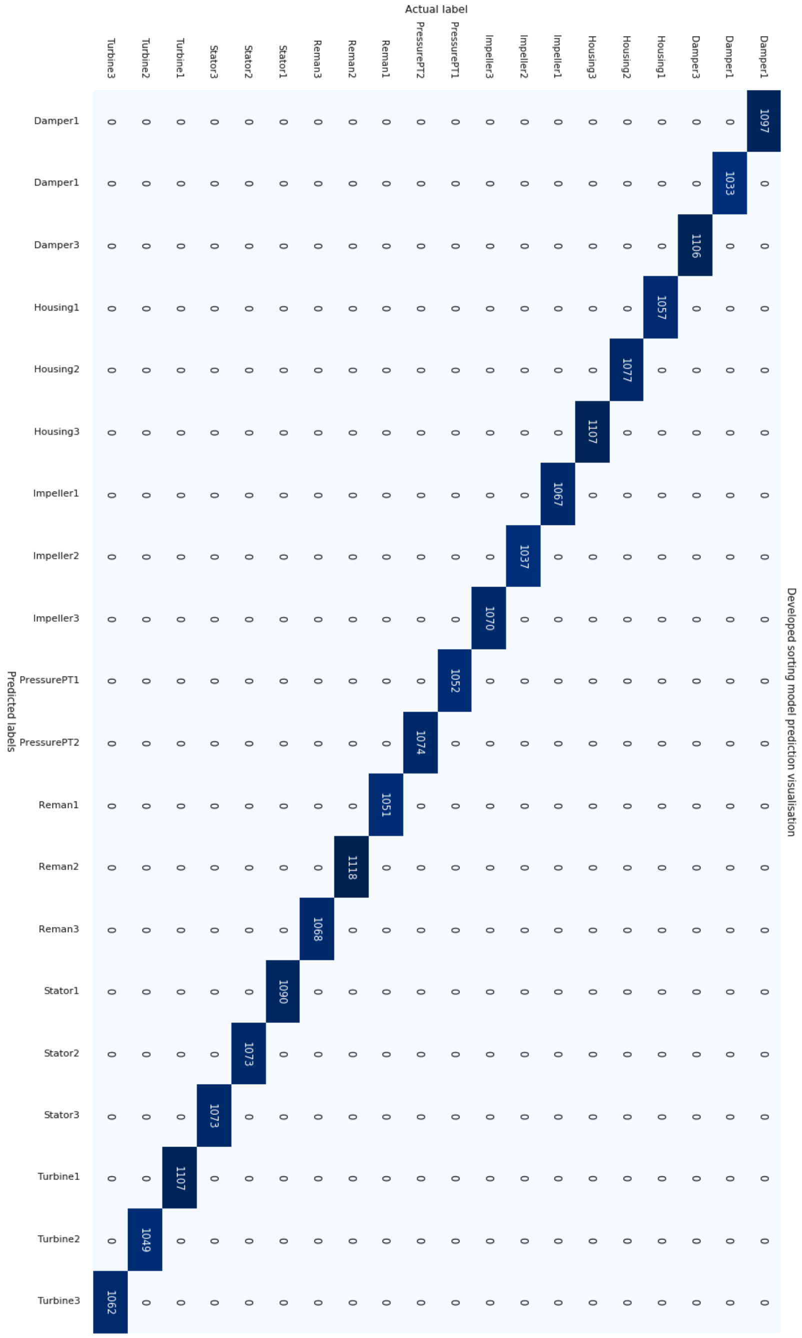
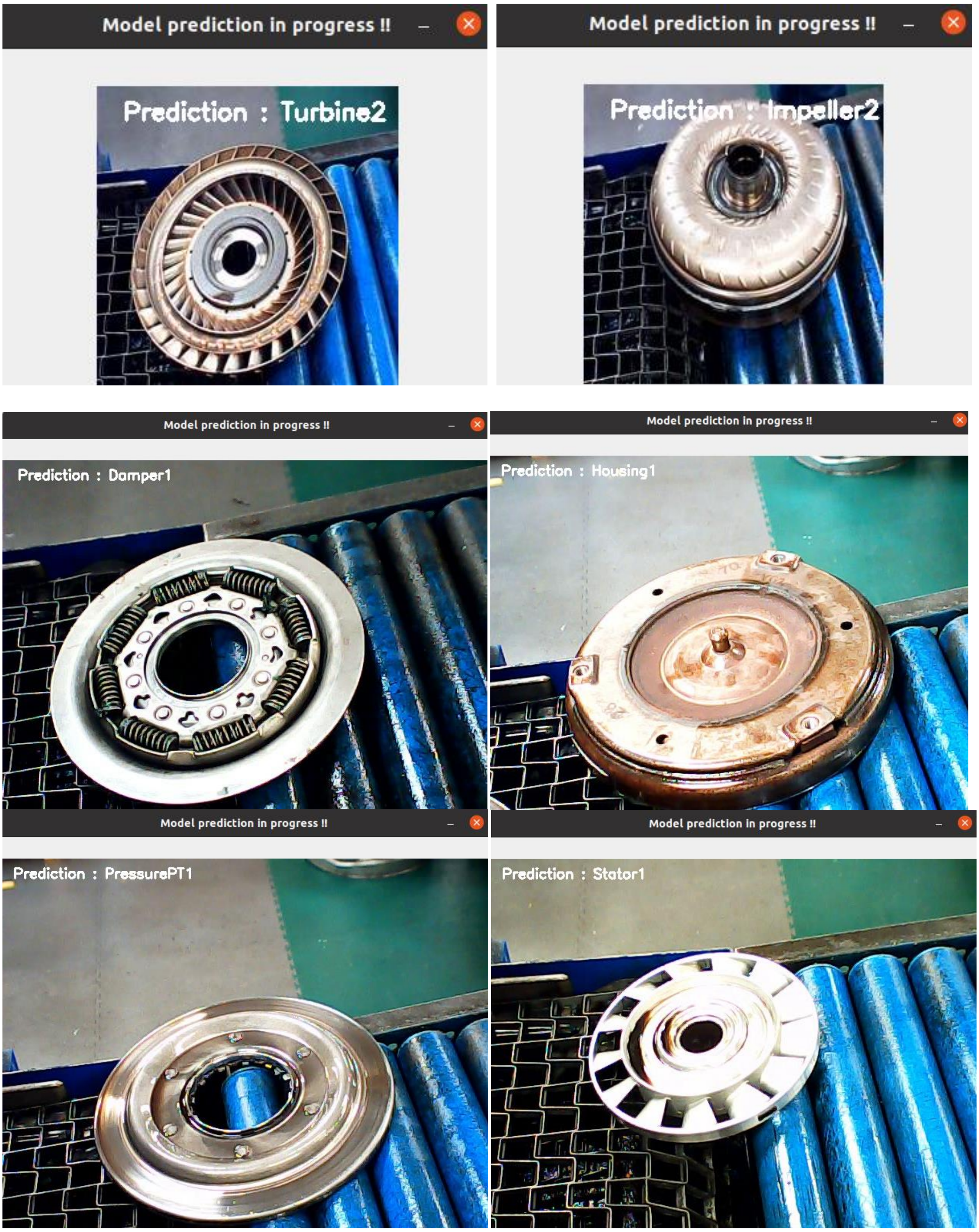


Figure 44 Sorting model confusion matrix.

Moreover, the inference from the connected camera focuses on a conveyor carrying the cleaned end-of-life products ready for the post-cleaning inspection, where the components are automatically identified and sorted into different classes using the model and the connected camera. The online testing helped visualise the model's predictions in real time, as shown in Figure 45. The codes to load the serialised model weights and make prediction was started, and the corresponding results from the live video feed were observed and recorded. As a result, the model accurately sorted the respective torque converter components into categories. A cross-section of the online sorting predictions is shown.



Model prediction in progress !!

Prediction : Turbine2

Model prediction in progress !!

Prediction : Impeller2

Model prediction in progress !!

Prediction : Damper1

Model prediction in progress !!

Prediction : Housing1

Model prediction in progress !!

Prediction : PressurePT1

Model prediction in progress !!

Prediction : Stator1

Figure 45 Sorting model predictions result from single-camera inputs

Nonetheless, the model can be incorporated primarily into both the pre-disassembly and post-disassembly inspection stages, where products are mixed during the reverse logistics and cleaning processes, prompting the need for an automated sorting process. The developed deep learning-based sorting system becomes a handy technology to automatically assess the product conditions and classify them into similar products.

Moreover, like every other data-hungry model, the sorting application requires a significant amount of training data, representing a challenge for applying CNN models for sorting in remanufacturing. Furthermore, the datasets used to train the models cannot contain all defects available by default for the industrial application under investigation, making holistic generalisation unreasonable. However, other authors have identified these similar limitations in using the CNN models in other industrial applications like Agriculture [363].

5.7 Modelling Process Control in Remanufacturing Using Deep Learning

The modelling approach for the process control application is developed as a binary classification problem that performs a classification as positive or negative for the respective classes. The binary classification differs from the multiclass problems modelled for the other applications, including the model's loss function and the output. These parameter modifications are performed on the learning algorithm before training and evaluating the model. The process control application is modelled as a supervised learning problem using labelled data.

5.7.1 Process Control Application and Data

The investigation of the use of DCNN for automated sorting in remanufacturing consists of an object recognition system that aims to identify objects in the video stream. These identified objects are afterwards used to trigger the actuation system to control a pressure valve to dry the cleaned torque converter units. The process control technique can be adopted for post-cleaning and pre-disassembly process control during or after remanufacturing. The existing process control techniques have been discussed extensively in Section 3.12.3 alongside the limitations of the existing methods. Finally, the development of the deep CNN-based process control system for remanufacturing applications is presented.

The process control application considers the torque converter system components recorded during this research. In addition, the experiment investigates the possibility of detecting wet surfaces after EoL product cleaning. The distance between the camera and the objects on the conveyor system was limited to approximately 40" to ensure that the camera's coverage was restricted to one object at a time using the existing programmable logic controller's time delay. As lighting contributes to visual sensing, the recording was made in an industrial work setting to reduce lighting effects after development.

The data consist of two (2) object categories of the torque converter units, including the wet and dry samples of the torque converter system. The two (2) object classes corresponding to each object under consideration are Wet and Dry. These labels helped identify and highlight the model predictions after training.

5.8 Model Components, Hyperparameters and Modification

The architectural modification for the CNN model used for the inspection application is necessary to adapt the developed model to perform process control in remanufacturing. The modified architecture used in the experiment is described as follows in Table 5.2, which highlights the number of filters used in the input, hidden, and output layers of the model, the expected number of inputs and outputs specified as the target in the output (Sigmoid) layer and the scoring function. The architecture consists of four convolutional layers of filter sizes 16, 16, 8 and 8, one fully-connected layer of size 512 and a dense filter. The architecture has a total of 697,761 parameters, all trainable parameters obtained by modifying the number of filters and the output layer, alongside monitoring the performance of the developed model.

Table 5.2 Model architecture optimised for the process control application

Layer Type	Output shape	Activation size	Parameters
Input	(None, 52,52,3)	8112	0
Conv2D	(None, 52,52,16)	43264	448
Activation (Swish)	(None, 52,52,16)	43264	0
Conv2D	(None, 52,52,16)	43264	2320
Activation (Swish)	(None, 52,52,16)	43264	0
Maxpooling	(None, 26,26,16)	10816	0
Dropout	(None, 26,26,16)	10816	0
Conv2D	(None, 26,26,8)	5408	1160
Activation (Swish)	(None, 26,26,8)	5408	0
Conv2D	(None, 26,26,8)	5408	584
Activation (Swish)	(None, 26,26,8)	5408	0
Maxpooling	(None, 13,13,8)	1352	0
Dropout	(None, 13,13,8)	1352	0
Flatten	(None, 1352)	1352	0
Dense	(None, 512)	512	692736
Activation (Swish)	(None, 512)	512	0
Dropout	(None, 512)	512	0
Dense	(None, 1)	1	513
Activation (Sigmoid)	(None, 1)	1	0

Besides, like in the original architecture, the width $\boldsymbol{m}$, the height $\boldsymbol{n}$, the number of filters in the previous layer $\boldsymbol{d}$, the bias $\boldsymbol{b}$ and the number of filters in the current layer under consideration $\boldsymbol{k}$ helped estimate the model's number of parameters. The number of parameters $\boldsymbol{P}$ of the model is obtained using the kernel size of [3,3], which represents $\boldsymbol{m}, \boldsymbol{n}$, channels in input $\boldsymbol{d}$ =3, bias term $\boldsymbol{b}$ = 1, and current layer $\boldsymbol{k}$ = 16 [352]. The model parameters are obtained using the relationship of equation **4.12** as follows

$$P = \big((m * n * d) + b\big) * k$$

$$Layer\ 1 = \big((3 * 3 * 3) + 1\big) * 16 = 448$$

$$Layer\ 2 = \big((3 * 3 * 16) + 1\big) * 16 = 2320$$

$$Layer\ 3 = \big((3 * 3 * 16) + 1\big) * 8 = 1160$$

$$Layer\ 4 = \big((3 * 3 * 8) + 1\big) * 8 = 584$$

Nevertheless, the fully connected are modified to suit the data for the process control application to suit the suit classes used in the investigation; therefore, the layer parameters are determined using the activation size of the model using the relationship of equation **4.13**.

$$P = current\ layer\ activation * previous\ layer\ activations + current\ layer\ activations$$

$$Layer\ 5 = 512 * 1352 + 512 = 692736$$

$$Layer\ 6 = 1 * 512 + 1\ \ = 513$$

The total number of parameters becomes the sum of the respective layer parameters of the model, giving 697,761 parameters. The other model parameters and hyperparameters useful for the training are outlined in Table 5.3. The most significant modification in the hyperparameters was the loss function and output activation. The binary cross-entropy loss scoring function was used alongside the Sigmoid squashing function at the output required for the two-class input data.

Table 5.3 Model components and hyperparameters

(Hyper)parameters	Value
Batch size	16
Epochs	50
Learning rate	0.005
Dropout	0.25/0.5
Activation	Swish and Sigmoid
Loss	Binary-crossentropy
Optimiser	AdaMax
Metric	Accuracy

The overall model architecture for implementing the deep learning inspection application in remanufacturing is attached as Appendix 2C.

5.9 Experiment and Model Training

Adapting the developed model for process control in remanufacturing involves the modification of the developed architecture to suit the data for the sorting application. The data consist of two (2) categories of 7156 images per class, making up the 14312 images, helpful in evaluating the adaption of the developed model to achieve automated process control during remanufacturing. The developed model is a multilayer architecture consisting of filters, convolutional and pooling layers, helpful in learning patterns in each data. A supervised learning approach is used to train the model, where data and labels are required to train the models. The model data was partitioned such that each class contained either the dry or wet samples. The test samples are loaded with the flow_from_directory function, which uses an alphabetical order to assign each class, making the class labels 0 for dry and 1 for wet samples. The test images are loaded at first and then resized to 52 x 52 x 3, which is the size of the inputs used to train the model.

The training process incorporates an optimiser that updates the model's parameters, a loss function that compares the predictions and actual values and a metric to assess the performance, thereby obtaining the model's best weight and bias parameters after multiple iterations over the training set. First, the model is compiled using the Keras build function before training. Next, the training process adjusts the weight parameters of the model using the model hyperparameters outlined in Table 5.3 until the specified epochs are complete, after which the model is serialised and saved for future inference after training. In addition, the model incorporates an AdaMax optimiser, a binary cross-entropy loss, and the accuracy metric used to score the model's performance. Besides, the dropout of 25% and 50% were applied to the model's first and final fully-connected layers to minimise overfitting.

Conversely, the model evaluation involves making predictions using the serialised model weights and testing the unseen data alongside predicting from connected camera sensor data. The serialised weights are loaded and used for classifying the wet and dry samples. However, the adapted process control application's performance was evaluated. As a result, the model produced final training and validation accuracies of 99.98% and 100%, respectively, while classifying the wet and dry components. The high prediction accuracy highlights that the model performed impressively on the test data. Furthermore, the high validation accuracy suggests that the model generalised well on unseen data, which is essential for deploying the developed model in real-time.

Consequently, Figure 46 shows that the model's training and validation errors are comparatively low. Moreover, towards the end of the training, the model produced approximately zero loss justifying the excellent performance. The validation and training errors are almost equal, highlighting a low bias and low variance in predicting the classes under investigation.

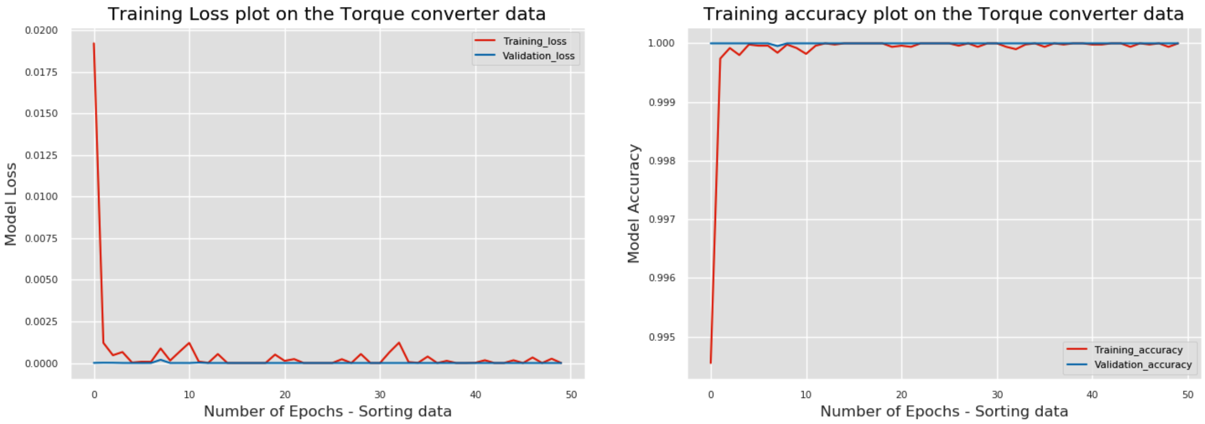


Figure 46 Model training results using the process control data

Nevertheless, the misclassification rate of the researcher developed model prediction adapted for process control is given by

$$M = 1 - Accuracy$$

$$M = 1 - 0.9998 = 0.0002$$

Overall, the model produced a minimal misclassification error of 0.02% on the process control data used in the first stage of the evaluation. These results further support that the model can achieve process control in remanufacturing, thereby attaining holistic process automation.

5.10 Results and Discussion

The method of adapting the deep learning techniques for process control in remanufacturing is investigated in the application. The researcher developed deep convolutional neural network model for recognising objects for process control to identify and classify wet parts that require the pressurised drying system to activate and dry the component. As a result, the process control application recognised various torque converter components, including dry and wet ones, with very high accuracy and used the results to control the valve for drying the components.

Conversely, the model prediction visualisation using OpenCV, a computer vision library, helped evaluate the performance of the developed model for process control during remanufacturing. First, the serialised weights are loaded and used to predict the inputs from the connected camera. Then, the model predictions were performed, with the predictions displayed using custom texts that show the predicted class. The text was also adjusted in position, colour, and size and inserted in the algorithm to show the predicted class around the top area in the original image. Besides, the inspection model predictions from the connected camera's live video feed highlight the model's performance, with some of the respective class predictions shown in Figure 47.

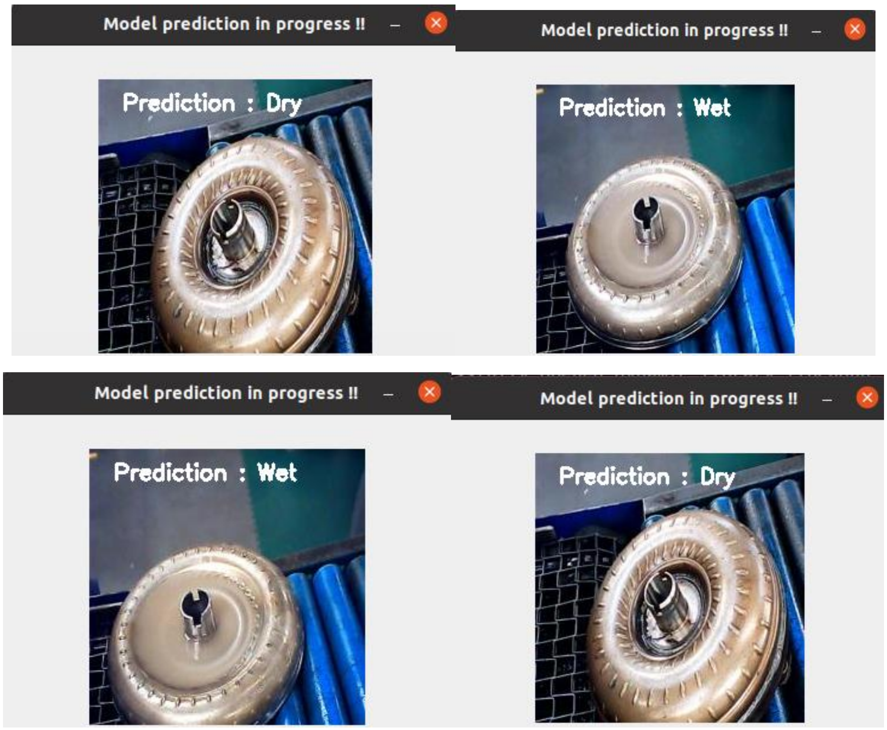

Model prediction in progress !!
Prediction : Dry
Model prediction in progress !!
Prediction : Wet
Model prediction in progress !!
Prediction : Wet
Model prediction in progress !!
Prediction : Dry

Figure 47 Cross-section of the model predictions from connected camera

5.11 Chapter Summary

This chapter outlined the process of adapting the developed deep learning model to achieve an automated sorting and process control in remanufacturing alongside evaluating the model performance using deep convolutional neural networks. The results highlight that the products and components for remanufacturing were successfully sorted into different categories using the developed model, suggesting that deep learning models can achieve automated visual sorting in remanufacturing. Besides, the model was also successfully adapted to remanufacturing process control and tested using the torque converter components. Finally, the chapter has successfully addressed research questions (Q4) by applying the newly developed model to the remanufacturing process control and sorting applications, automating the processes (Q2) and providing the benefit of process improvement.

CHAPTER SIX

QUANTITATIVE ANALYSIS AND INDUSTRY FEEDBACK

6.0 Introduction

This chapter evaluates the cross-case analysis of the various parameters of the developed models to assess and understand the model performance alongside addressing the research question Q3. Chapter 2 presented various reviews and summaries of different model parameters, including activation function, optimisation, batch normalisation, and other model regularisation methods to improve model performance and generalisation. The effect of these parameters on the different process data used to model the respective remanufacturing processes is evaluated. Furthermore, it highlights the research validation and verification exercises performed, which are a series of processes to assess the reliability and accuracy of computational models. Chapters 4 and 5 outlined the various deep learning models for remanufacturing sorting, inspection, and process control applications.

Besides, the analysis of the single-domain models highlights that the learning algorithm's performance and the classifier's performance are two broad approaches to testing classifier models [331]. This research analysis extends from the experimental design of the research. An experiment is *"a carefully worked-out and executed plan for data collection and analysis"* [364]. A properly designed experiment allows for the inference of causations. The plan for the model analysis is structured to consider vital parameters of the algorithm that can be analysed from the model to improve performance. The overview of the performances adapts the proposed taxonomy of statistical questions, as shown in Figure 48, which outlines the crucial questions that general learning models focus on [331]. These vital questions are structured from the model hyperparameter, which helps make critical decisions.

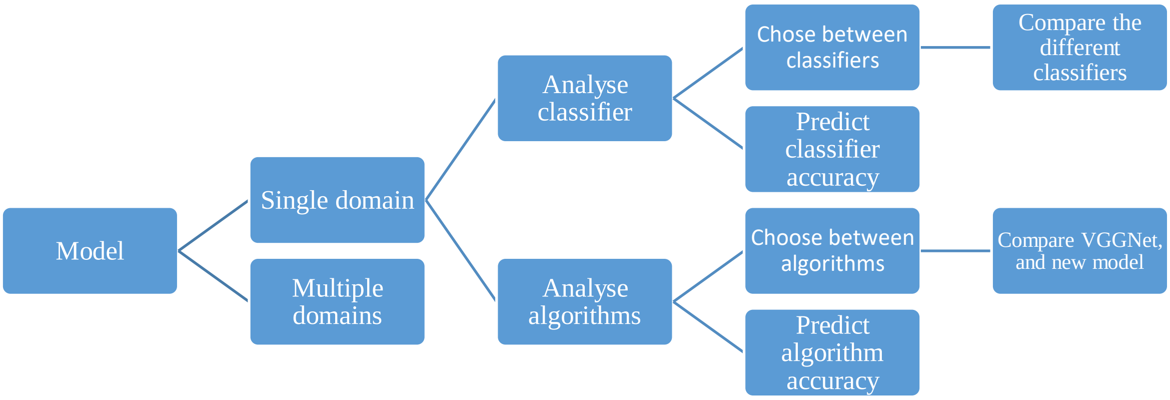

Model
Single domain
Multiple domains
Analyse classifier
Analyse algorithms
Chose between classifiers
Predict classifier accuracy
Choose between algorithms
Predict algorithm accuracy
Compare the different classifiers
Compare VGGNet, and new model

Figure 48.1 Adapted Model analysis approach for comparing predictive classification tasks [331]

It is worth outlining that both single and multiple domains consider similar questions; however, the specific applications make the difference. Unlike the single domains where the intense focus is to design an algorithm or classifier that can perform well in a particular task like the remanufacturing inspection, sorting, or process control tasks. The analysis of the single-domain models focuses on either of the following [331]:

- Performance of the learning algorithm
- Performance of the classifier

The developed algorithm's analysis focuses on the model's hyperparameters to support decisions made in the design and enhance understanding of the model. Besides, it is essential to highlight that the analysed models fall into single-domain applications that require algorithm modifications to work on different domains where necessary.

6.1 Analysis of the Learning Algorithm

The analysis of the developed model is performed to evaluate the model's performance. Researchers suggest that classification models require a minimal statistical analysis if there is enough data to accommodate keeping a test set out of the training set during training [331]. This approach changes the model design focus from providing statistical hypotheses to focusing on the model-specific parameters to evaluate the performance. The performance of the newly developed learning algorithm is evaluated and presented based on the crucial parameters that affect the performance of these models. The evaluation of the VGGNet used to explore the initial model is not considered because the VGGNet model was tested and optimised during development before they were released; therefore, altering the internal components of those architectures creates another new model that requires optimisation. This hyperparameter optimisation aims to achieve optimal performance from the model. The code tree for the model analysis is shown in Appendix 1B.

Nevertheless, the effect of different parameters and hyperparameters of the neural network has been evaluated by other researchers; however, the empirical investigation of the performance of CNN and RNN models when the architectural designs are modified validates that there are substantial performance fluctuations when changes in the number of hidden layers, batch size, and learning rate of a given model, thereby highlighting the need to consider these factors when designing the computational models [365].

6.2 Model Hyperparameters

The deep learning model hyperparameters are variables predefined before training a given deep learning model. These hyperparameters constrain the model to fit the provided data and are passed as arguments to the constructor used in the model estimators. The nature of model hyperparameters is also vital for determining how they are used. There are continuous, discrete, and categorical hyperparameters. The discrete hyperparameters are valuable for evaluating the number of estimators in ensemble learning models. Conversely, the categorical helps to implement model regularisation and loss functions while the continuous help to determine the penalisation coefficient and the number of sample splits. The nature of model hyperparameters effectively outlines the intervals of parameter search during the investigation. Moreover, the most significant tuning considerations include the total number of model parameters, the nature, the available computational resource, and the low effective dimensions. The low effective dimension avoids searching the hyperparameter spaces where the model performances do not increase.

Moreover, there are different methods of hyperparameter search optimisation, including manual search, grid search, random search, and Bayesian search, among others [366]. The manual search is an initial technique for finding the model parameters that produce a near-optimal performance. It provides an intuition of the vital area to focus the parameter search and the initial values for the grid search method. The manual search helps familiarise model hyperparameters and their effects while setting up a benchmark comparison model before optimisation. However, the manual search is often limited because it is time-consuming and lacks reproducibility due to the undocumented random combination of parameters at the initial testing stage. In addition, it does not explore the entire hyperparameter space or scale because a small part of the model parameters was used in the manual search.

Conversely, the various hyperparameters that describe the CNN models are outlined in Table 6.16.1, which make up the model architecture. These model parameters include the number of layers, dropout rate, optimisation technique, number of neurons per layer, activation functions, loss models, batch sizes, epochs, learning rate, verbose, output metrics, etc. Model parameter choices involve deciding the appropriate design configurations to achieve the research aims and objectives. The individual decisions at the design stage include the model architecture, activations, pooling and regularisation techniques, optimisation, and the frameworks to use. The model development considers these hyperparameters used to control the model performance.

Table 6.1 Model parameter and hyperparameter definitions

(Hyper)Parameters	Function
Layers	Layers describe the model topology of the given architecture.
Batch size	Number of samples to pass to the model at a given time
Epochs	Number of times to present samples to the model
Verbose	Debug parameters are used to control the display on the shell screen.
Learning rate	The parameter that controls the minimum step size that the model uses for moving toward a minimum loss
Dropout	A technique to drop node units randomly during training.
Optimiser	Process of finding the parameters that give the minimum or maximum output
Loss	The loss defines the objectives on what performance is evaluated
Metrics	The parameter used to determine the model performance include accuracy, error rate, false positive rate etc

However, an identified challenge of learning models is the heuristic nature of finding the hyperparameters since there are no specific formulas to obtain the best model parameters. Therefore, the choice of the number of hyperparameter combinations to test is critical. The higher the number of combinations, the better the performance at the cost of the improved computational burden. However, a cautious approach to increasing the parameters can lead to non-improved performances when the model capacity is reached.

6.2.1 Model Initialisation

Model initialisation is essential when training deep convolutional neural network models from scratch. This is because it helps to ensure the model parameters do not vanish during training, often referred to as a vanishing gradient, which results when specific model parameters are so small to propagate to the output layer of the model. Instead, the initialisation ensures that model parameters are kept within ranges during the entire training process using the variance of the distribution. Moreover, three experiments were performed on the sorting and inspection application datasets to evaluate the repeatability effects on different tests. The researcher-developed model was used, and the effect of both the Gorot and He initialisations methods were evaluated for both uniform and normal distributions. The uniform distribution has a constant probability of occurrence and the normal distribution; usually, a Gaussian distribution has zero mean and standard deviation of 1. Figure 2 shows the plots of the effect of initialisation methods. The results highlight a correlation between the model losses reducing across the four initialisation methods and the accuracy increasing across the four methods.

Furthermore, the two-inspection data produced identical performance on model loss and accuracy; however, the sorting application produced a slightly downgraded performance compared to the inspection applications. Moreover, these performances highlight that model initialisation slightly impacts the performance after training the designed architecture, although the two mainly used deep learning initialisation methods were considered. However, the He uniform initialisation works best on the architecture as it achieved very high accuracy and obtained the most negligible loss.

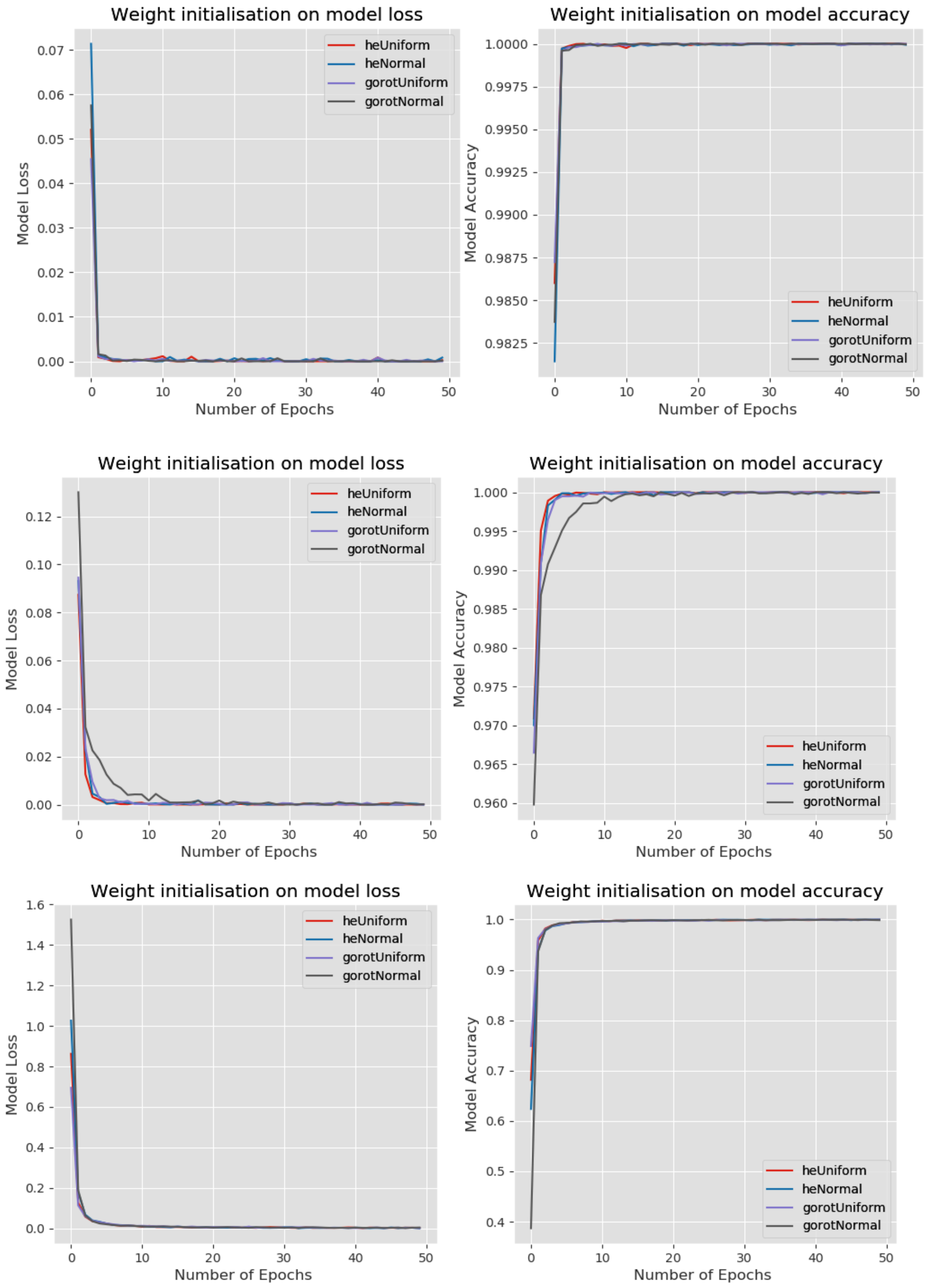


Figure 6.2 Effect of weight initialisation on performance: at the top - inspection I, middle - inspection II, bottom - sorting.

Furthermore, the He initialisation works better for both the uniform and normal distributions of weights used in the model evaluation, with the Gorot's normal initialisation producing the worst performance. On the other hand, the He uniform initialisation achieved the best performance on the sorting and inspection I data. In addition, it obtained the highest prediction accuracy and lowest loss, attaining significant results in the inspection II application. This result informs the choice of He uniform initialisation as the default initialisation of the architectures used in the research.

6.2.2 Selection of Batch Size

Batch training is a model regularisation method that enhances performance. The batch size is a mini-batch of training samples that make up a training batch, and it represents the number of samples read by the model per iteration before updating the model weights parameters. The batch size ensures that the computer's memory is not overwhelmed by the total samples passed at any given time during training, thereby maximising the use of the system memory. It also impacts the training speed as the larger batches increase training speed in contrast to the smaller batch sizes, causing an increase in training time. The stochastic gradient descent with parameters θ considers the model training as a non-convex optimisation problem that corresponds to the loss minimisation $L(\theta)$ with respect to the parameter θ where the loss is defined as the average loss per training example $L_i(\theta)$ over the entire training examples. The model loss is given by

$$L(\theta) = \frac{1}{m}\sum_{i=0}^{m} L_i(\theta)$$

Where $m =$ size of the training set. The batch size significantly affects the generalisation of a deep learning model alongside the training time, as tiny batch sizes increase the overall training time. The effect of batch size on the network's performance has been investigated by researchers with notable performances outlining that the batch size improves performance for more effective learning rates while lower learning rates produce good performances on smaller batch sizes [367].

The effect of the batch size on the three applications was evaluated. The results highlight the smaller batch sizes seem to perform well across the applications, with the batch size of 16 samples producing the best performances, as outlined in Figure . Besides the batch size of 16 representing the number of iterations to run one epoch for the 70% of the datasets used for the model training is given by

The number of iterations per epoch for the first inspection application is $\frac{20037}{16}$ = 1252 iterations /epoch.

The number of iterations per epoch for the sorting application is $\frac{50092}{16}$ = 3130 iterations/epoch.

The number of iterations per epoch for the second inspection application is $\frac{20160}{16}$ = 1260 iterations/ epoch. These iterations are completed before any model weight updates during training. This batch size was chosen and fixed in the training of the final models.

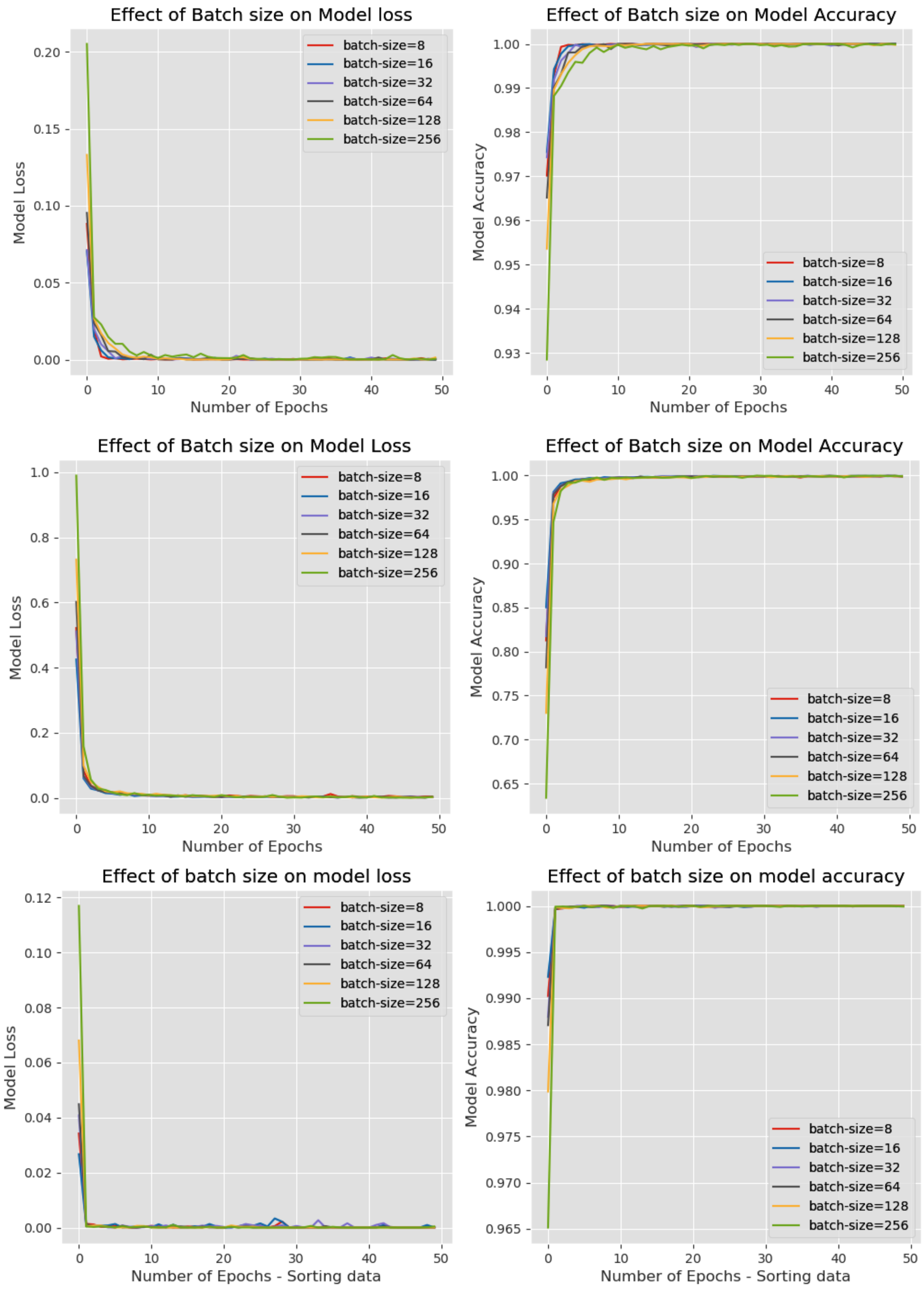


Figure 6.3 Effect of batch size on performance: top - inspection I, middle - inspection II, bottom - sorting.

However, the results suggest that the smaller batch sizes obtain better performance to attain peak accuracy alongside the least losses during training, highlighting that smaller batch sizes enhance optimal performance, with the larger batch sizes showing poor performance. However, balancing the speed and accuracy informs the choice of a batch size of 16 as it achieved comparable loss and accuracy to the smallest batch size.

6.2.3 Effect of Batch Normalisation on Model Performance

The batch normalisation aims to improve the training speed of the model by normalising the activations of the given input volume before propagating to the subsequent layers. The batch normalisation has been discussed in detail in Section 3.11.5, and It considers the mini-batch of the activations $Z^{(i)}$as the input feature map, to produce a normalised output as follows

$$Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}}$$

Where $Z_{norm}^{(i)}$ $=$normalised feature-map, $\epsilon =$ numerical stability constant usually set to 0.001. These features are controlled by the mean (μ) and variance (σ) of the mini-batch feature map obtained using the relationships

$$\mu = \frac{1}{n} \sum_i Z^{(i)}\ , \ \sigma = \frac{1}{n} \sum_i Z^{(i)} - \mu$$

However, the value of these selected parameters is of reasonable concern to the designer as significant computations cause numerical instability; thereby, the need to normalise the data becomes paramount. Normalisation entails scaling the values of the network weights to range between 0 and 1 or between -1 to 1. It is achieved by dividing every pixel in an image by 255, the maximum obtainable number of pixels. Normalisation makes the learning process seamless when we train the model.

A plot comparing the effect of batch normalisation on the model performance highlights that a slight prediction accuracy improvement is achieved across the three applications but is not very significant, as outlined in Figure 49.

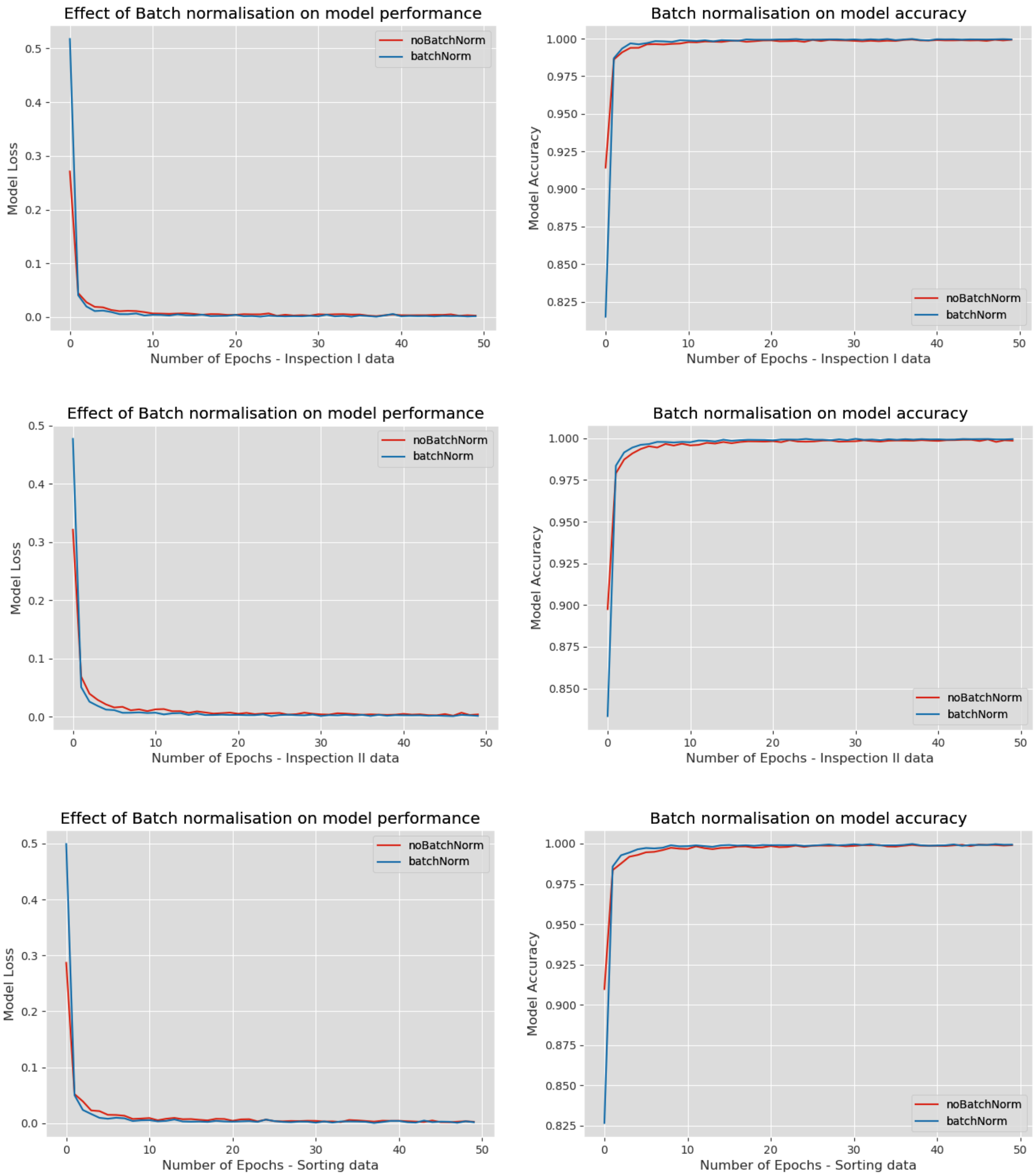


Figure 49 Effect of batch-normalisation on performance: top - inspection I, middle - inspection II, bottom - sorting.

6.2.4 Selection of Activation Function

The activation functions are vital parameters of deep convolutional neural network models. These functions convert the linear parameters of the model to non-linear parameters for further propagation to the subsequent layers, thereby ensuring that there are no dead neurons during training. The different activation functions have been reviewed in detail in section 3.11.2.4. However, the selection of the most appropriate activation function for the model was evaluated in this section. Exploring the different activations in the model helps select the most appropriate function for the designed architecture to attain optimal performance. The six selected activations were part of the principal identified functions used in deep learning research.

Nevertheless, investigating and plotting the six considered activations, including ReLU, Swish, Softsign, Softplus, ELU, and Selu, highlights that these functions compete significantly against each other except for the Softplus function, whose accuracy and loss responses were poorer than the other five activations. In addition, the ReLU, ELU, Selu, Swish and Softsign performed remarkably well; however, some functions produced slightly better performances, as shown in Figure 50. Moreover, the Swish function produced the best performance, with the highest prediction accuracy across the three datasets and the most negligible loss after training, thereby supporting the findings that the Swish is an emerging activation used in deep learning research [234]. Therefore, the Swish function was chosen as the activation function used in the researcher-designed architecture's hidden layers.

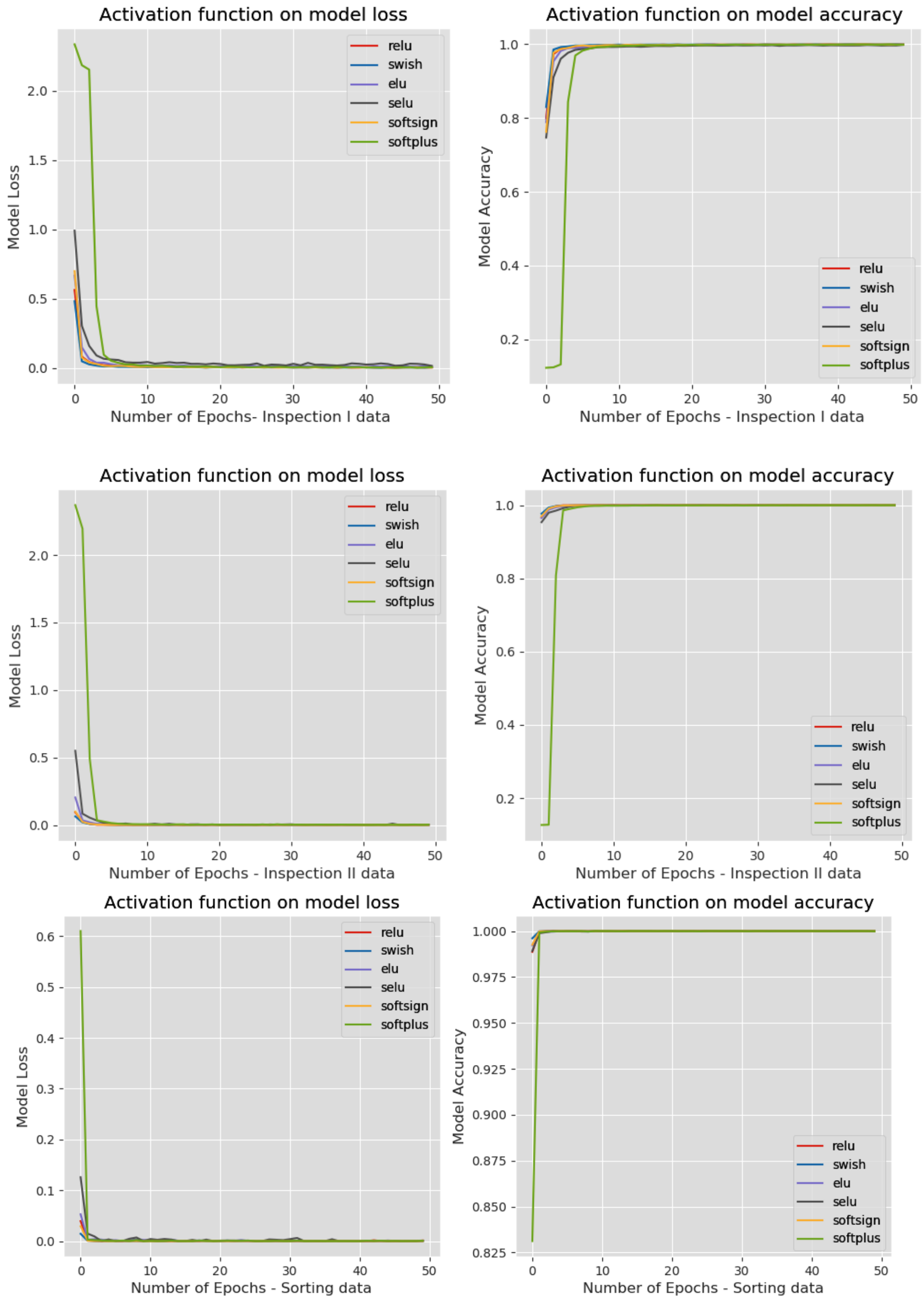


Figure 50 Effect of activations on performance: top - inspection I, middle - inspection II, bottom - sorting.

Perhaps, any activations that achieve peak performance at obtaining the most negligible loss and peak accuracy in the fastest possible time is the candidate for ideal activation, and the function selected was based on these vital performance benchmarks.

6.2.5 Selection of Learning Rate

The effect of the learning rate in the training of various learning algorithms is significant to achieving an optimal model. The learning rate helps the designer achieve a model that does not overfit or underfit the training data while converging to a local minimum, often called the best accuracy. The effect of these learning rates is best observed on the learning curves where an ideal learning rate achieves the least loss and the best accuracy during the neural network model training. In contrast, a high learning rate produces a model with a very high loss, and a low learning rate creates a model that takes a very long time to converge if it converges. The pictorial view of the effect of these learning rates is shown in Figure 51.

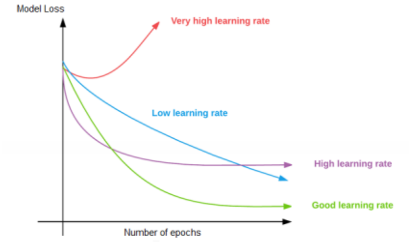


Figure 51 Effects of learning rates adapted from [257]

Moreover, the training process involves the backpropagation of the gradient of the loss function with respect to the model's weights and, afterwards, updating these weights with respect to the learning rate using the relationship.

$$W_1' = W_1 - \sigma \frac{\partial \iota}{\partial W_1}$$

Where $W_1' =$ new weight, $W_1 =$ original weights, $\sigma =$ learning rate, and $\partial \iota =$ loss function. The learning rate determines the decrease applied to the weight parameters during training and controls how long it takes to complete the training. Moreover, since the most important goal of model training is to obtain the best weights that produce the least losses, thereby minimising the model's error rate, it is important to evaluate the learning rate that would produce a high-performing model after training.

The heuristic process of selecting the learning rate for the model considered five learning rates obtained from a range of recommended rates from deep learning practitioners, ranging from 0.001 to 0.00001. The evaluation of these rates on the model data highlights that the lowest learning of 0.00001 requires a very long training time for convergence. At the same time, the highest rate of 0.1 also did not produce the most optimal performance, suggesting that the model did not reach the global minimum during training with the highest rate, as shown in the training response of Figure 52.

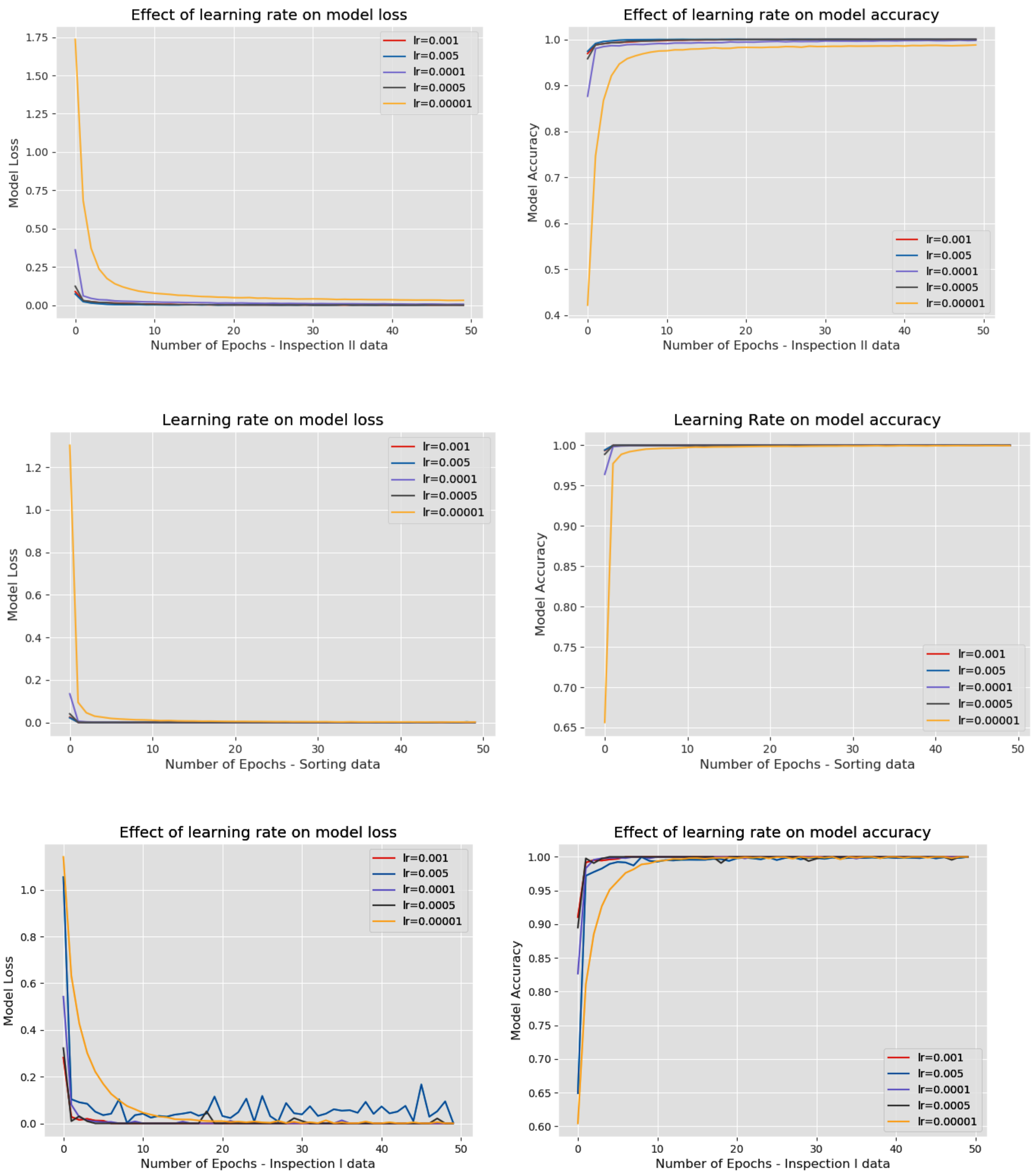


Figure 52 Effect of learning rate on performance: top - inspection I, middle - inspection II, bottom - sorting.

However, comparing the other four learning rates highlights that the rate of 0.005 was the most optimal learning rate, producing the lowest model loss after training. Furthermore, the model prediction accuracy comparison also justifies that the learning rate of 0.005 was the most optimal rate to achieve the highest accuracy at the quickest time at approximately five epochs, thereby choosing the model learning rate of 0.005 for the training of the subsequent models.

6.2.6 Selection of Optimisation Techniques

The effect of the optimisation method on the model's accuracy is outlined for different optimisation methods used in deep learning applications to evaluate the model performance. The different optimisers considered in the model include SGD, Adam, AdaMax, RMSProp, AdaGrad, and AdaDelta optimisers, among the outlined optimisers in Section 3.10.4.

Furthermore, the results highlight that the Adam and AdaMax functions achieved peaked performances early during training and maintained high accuracy until the end. Besides, the SGD and AdaGrad were quick and attained peak accuracy in less than five epochs, maintaining Adam's exact performance level. However, the Adagrad proved to have the slowest convergence and requires a longer time to achieve optimal loss and accuracy than the other optimisers. The overall performance of the optimiser is depicted in Figure 53.

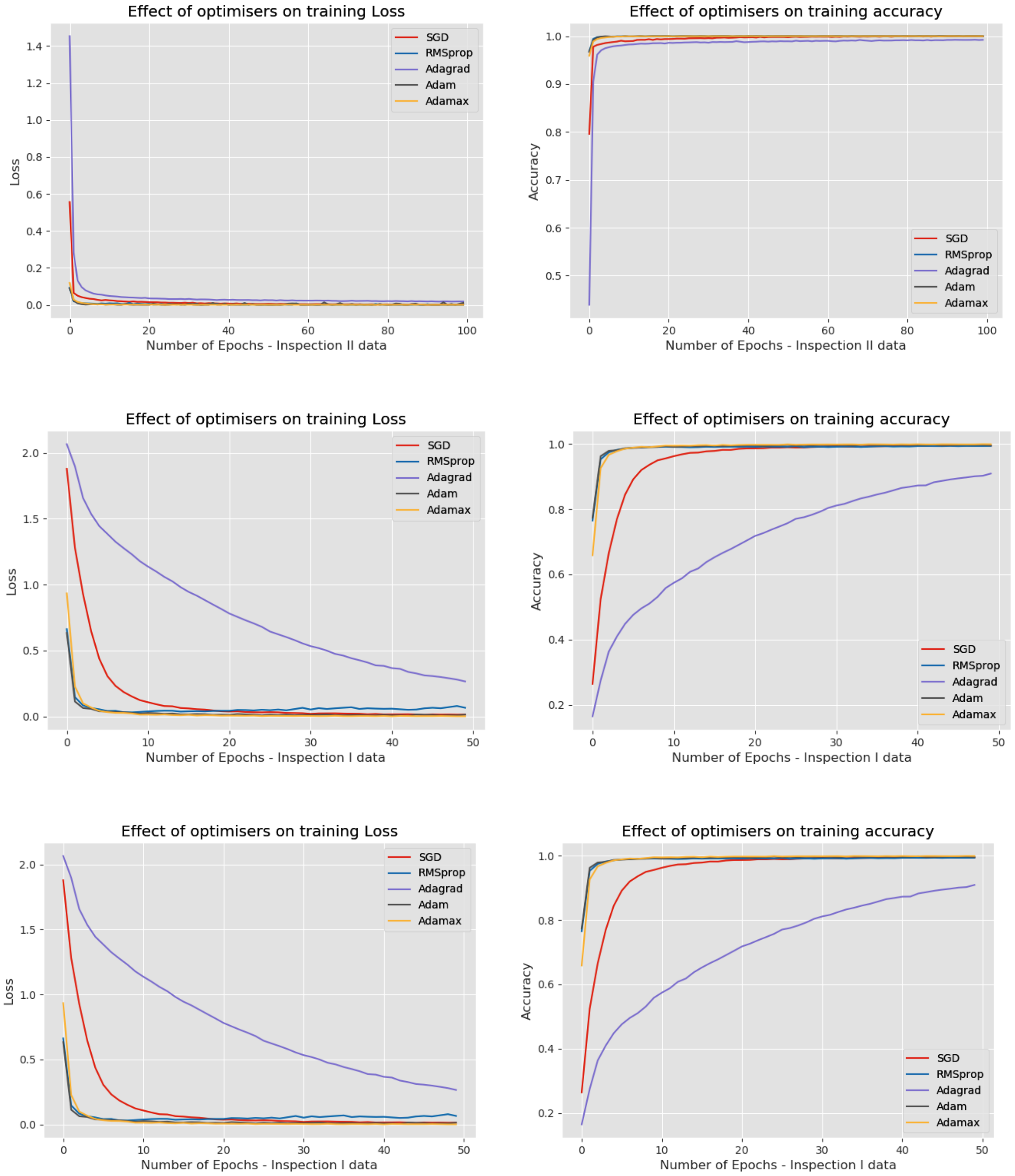


Figure 53 Effect of optimisers on performance: top - inspection I, middle - inspection II, bottom - sorting.

Similarly, the effect of the optimisation on the model loss is similar to the training accuracy. Moreover, training losses are inversely related to training accuracy; the larger the value of the models' loss, the lesser the accuracy of the model's predictions and the more work required to improve the model performance. The loss response suggests that the learning is progressive on all the optimisers as the losses consistently decrease; however, the Adagrad function produced the highest loss, suggesting that the optimiser is not performing well on the architecture. Perhaps, from the results, the most appropriate optimiser is the Adam family. The AdaMax function was selected as the optimiser for the developed CNN architecture and used in subsequent applications as it performed better than the original Adam on the architecture.

Finally, other model parameters evaluated during the investigation include the loss function, dropout, augmentation and shuffling. The dropout was added to the hidden layers to reduce overfitting, while the different augmentation methods added to the training data include zoom, flipping, and rotation. However, the augmentation introduced various effects that the fixed camera could not capture during data collection. The augmentation ensures that the model generalises well on unseen data during testing. Similarly, a shuffle effect was added to the training examples to ensure the randomness of the data during model training. The shuffle and augmentation together enhance the model generalisation.

6.3 Evaluating the Model Layers

The evaluation of the model layers is another analysis performed to ascertain that the developed model is suitable for the modelled tasks. The evaluation of the state-of-the-art VGGNet alongside the researcher developed models was studied to understand how the deep learning architectures derive the architectural patterns. Visualising the model learning across the five blocks of convolutional layers of the VGGNet architectures highlights learning patterns using deep architectures. The comparison of the state-of-the-art model with the researcher's developed model was performed to visualise the differences and similarities, especially since the VGGNet model has over 14 million learnable parameters compared to the 1.423 million learnable parameters in the researcher's developed model.

Moreover, the evaluation involves feeding the VGGNet model with an input image of the same dimension used to train the model, with the model's output taken from the model's respective hidden layers. The outputs of the five convolutional block layers of the VGGNet model were obtained as follows.

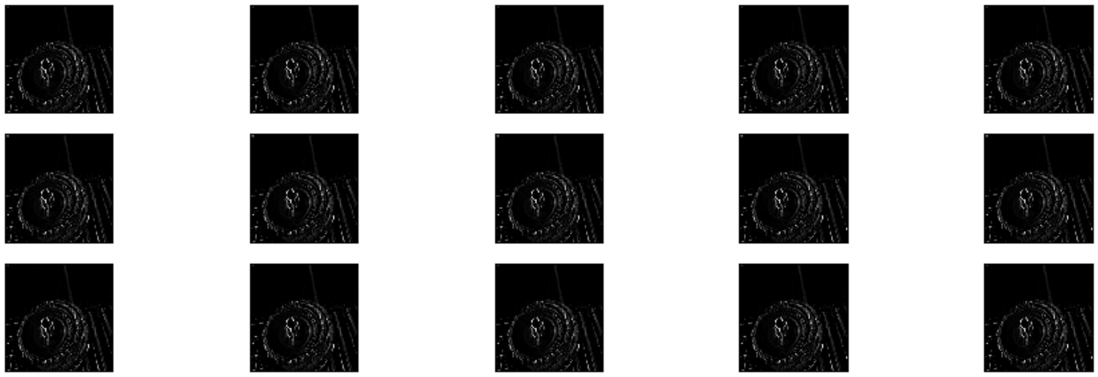


Figure 54 VGGNet Layer 1 block visualisation

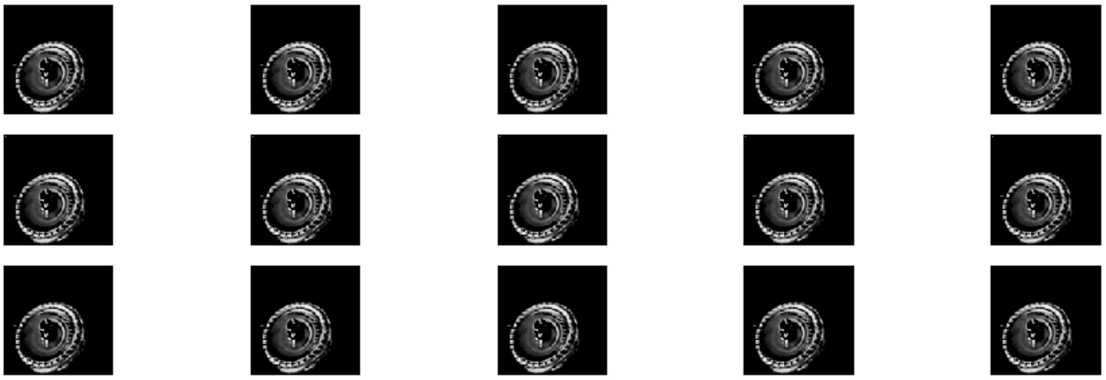


Figure 55 VGGNet Layer 2 block visualisation

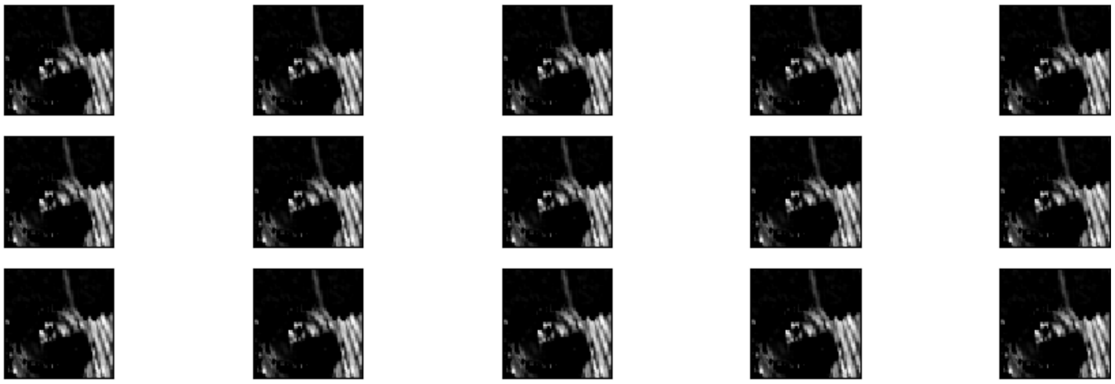


Figure 561 VGGNet Layer 3 block visualisation

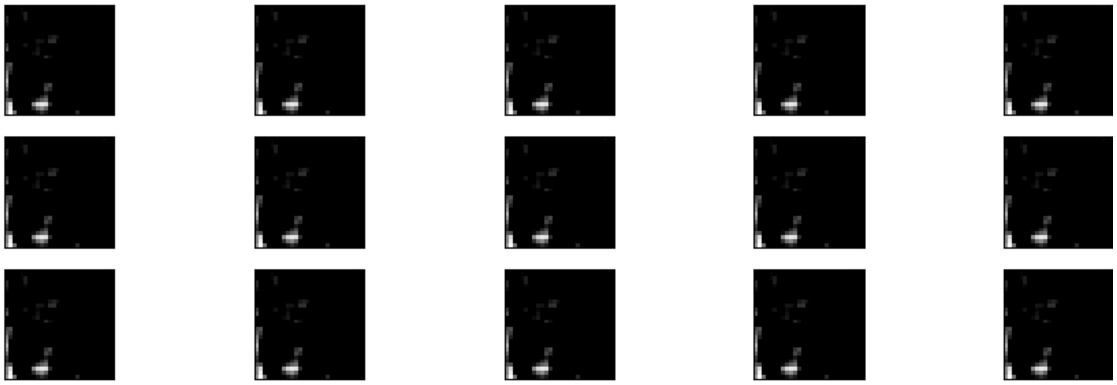


Figure 6.12 VGGNet Layer 4 block visualisation

Figure 573 VGGNet Layer 5 block visualisation

Hence, it is evident that the initial hidden layers act as edge detectors from Figure 54.9. The second hidden layer in Figure 55 obtained even more refined features than the first hidden layer. Perhaps, the third hidden layer has the least human-recognisable features in the images, as shown in Figure 56.11. The deeper hidden layers of Figure .12 and 6.13 have low-level features that can not be distinguishable by the human eyes.

Conversely, the outputs of the four hidden layers of the researcher developed model were also obtained to evaluate the performance of the hidden layer filters. The first hidden layer produced almost a replica of the VGGNet initial hidden layer output, as outlined in Figure 586.14. The other hidden layers also had similar features extracted and combined in Figure 596.15 and Figure 606.16 to obtain the final outputs shown in Figure 6.17. The model uses these rich features to understand the patterns in unseen examples.

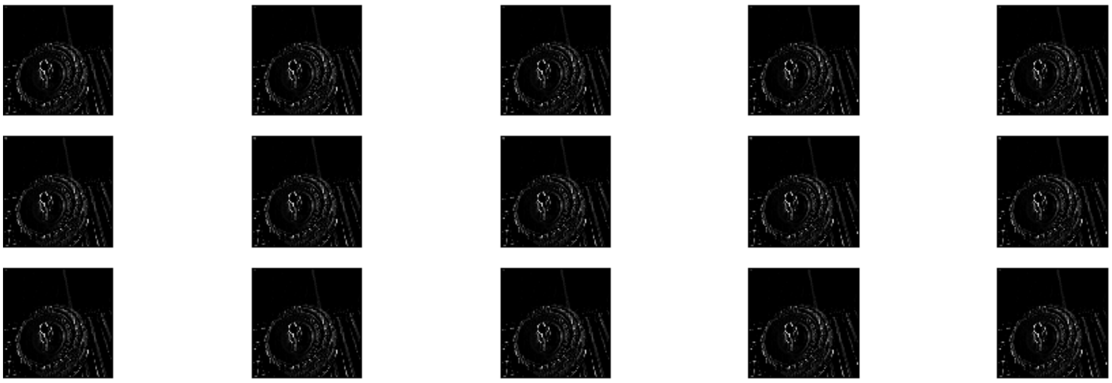


Figure 58 Layer 1 of the researcher developed model

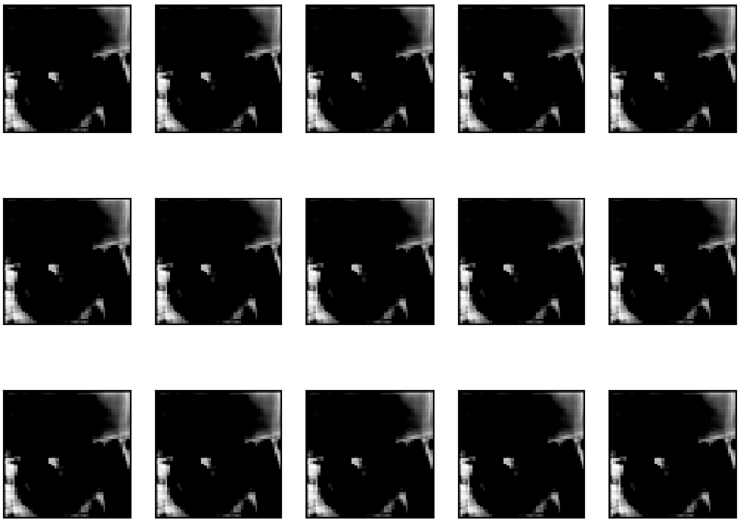


Figure 595 Layer 2 of the researcher developed model

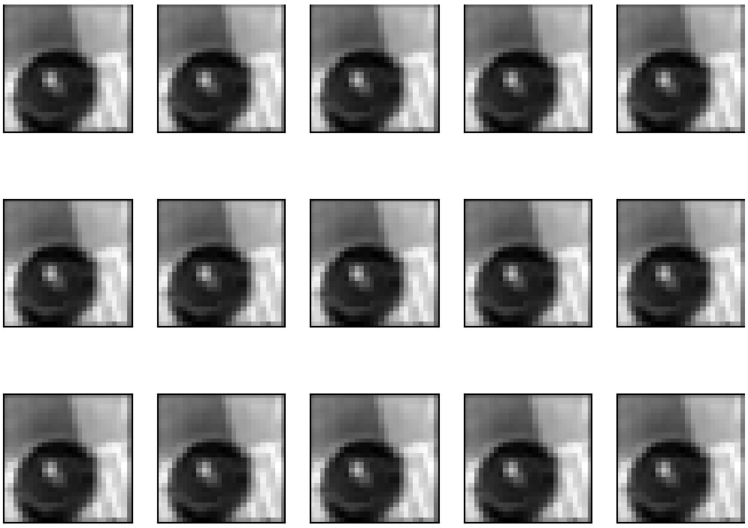


Figure 60 Layer 3 of the researcher developed model

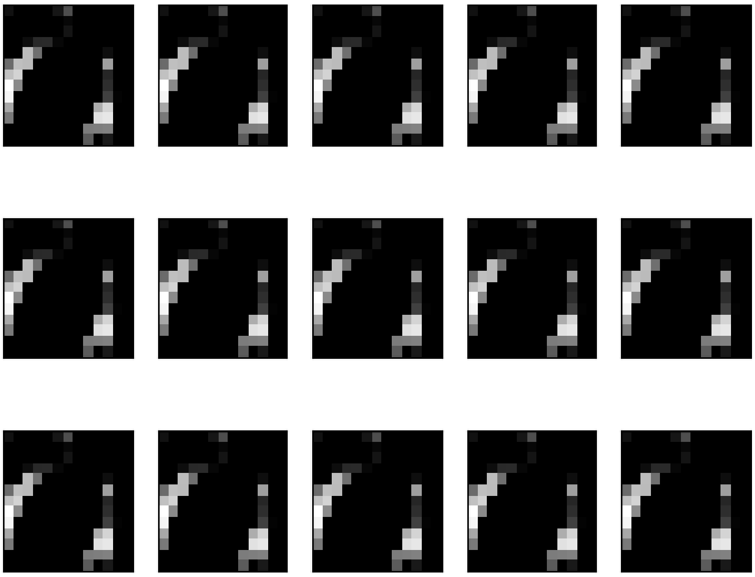


Figure 6.17 Layer 4 of the researcher developed model

The visualisation from both models highlights that the initial layers of the convolutional neural networks are similar to edge detectors that capture fine details about the objects. In contrast, the last layers are usually dark sheds of grey, which are unrecognisable by humans. Besides, the details in the hidden layers diminish as the depth of the model increases; however, the model learns these abstractions to better reconstruct the inputs during classification, thereby making it robust to identify and perform classification on unseen inputs. Lastly, the darker sheds found in the last layers of the models are inhibitory weights in the learned features. In contrast, the white square sheds represent the excited weights of the models and highlight that the researcher developed model successfully learned the patterns in the data, which is helpful to achieve high classification accuracy.

6.4 Model Confirmatory Test

The developed model's analysis is presented, including the compilation of the predictive results of the developed classifier, the training from scratch and transfer learning on the VGGNet architecture. Finally, the confirmatory test is presented to evaluate and relate the model's performance using the Kappa coefficient, which removes the possibility of the model predictions and random guesses agreeing alongside measuring the number of predictions that random guesses cannot explain.

The Kappa coefficient or statistic is a chance standardised and corrected measure of agreement between categorical scores produced by two raters and is valuable for representing agreements between raters on categorical variables. The Kappa statistic lies between 1 and -1, with 1 representing complete agreement and 0 or lower meaning chance agreement [368]. After removing the chance agreement, the Kappa coefficient represents the proportion of agreement between two observers. It is usually a scale proportion of each category used in the model.

The contingency table of the model prediction is used to obtain the parameters of the kappa coefficient, used to evaluate and obtain the Kappa coefficient. For a table consisting of N subjects assigned independently to one of the k-categories by two separate raters, with p_{ij} representing the portions of subjects that Rater I classified in category i and Rater II, classified as j, and $i, j = 1,2, \dots ,\ k$. The proportion of p_i and $p_{.j}$ are the frequencies of assignment into categories i and j, respectively for the Raters I and II. Perhaps, where the inputs to the contingency table are probabilities, the respective Rater category frequencies sum to one, as shown in Table 6.2.

Table 6.2 Contingency table showing the Rater prediction probabilities

Rater II					
Rater I	1	2	$\cdots$	k	Total
1	p_{11}	p_{12}	$\cdots$	p_{1k}	p_1
2	p_{21}	p_{22}	$\cdots$	p_{2k}	p_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
k	p_{k1}	p_{k2}	$\cdots$	p_{kk}	p_k
Total	$p_{.1}$	$p_{.2}$	$\cdots$	$p_{.k}$	1

Nevertheless, the diagonal proportions p_{ii} represents the portion of the subjects that both Raters I and II predictions agreed on the assignment. The overall proportion of the observed agreement is given by

$$p_o = \sum_{i=1}^{k} p_{ii}$$

Furthermore, the overall expected agreement by chance is given by

$$p_e = \sum_{i=1}^{k} p_{i.}p_{i}$$

The overall Kappa coefficient measures the degree of rater agreement and is obtained as follows

$$K = \frac{P_o - P_e}{1 - P_e} = \frac{Agreement - Expected\ agreement}{1 - Expected\ agreement}$$

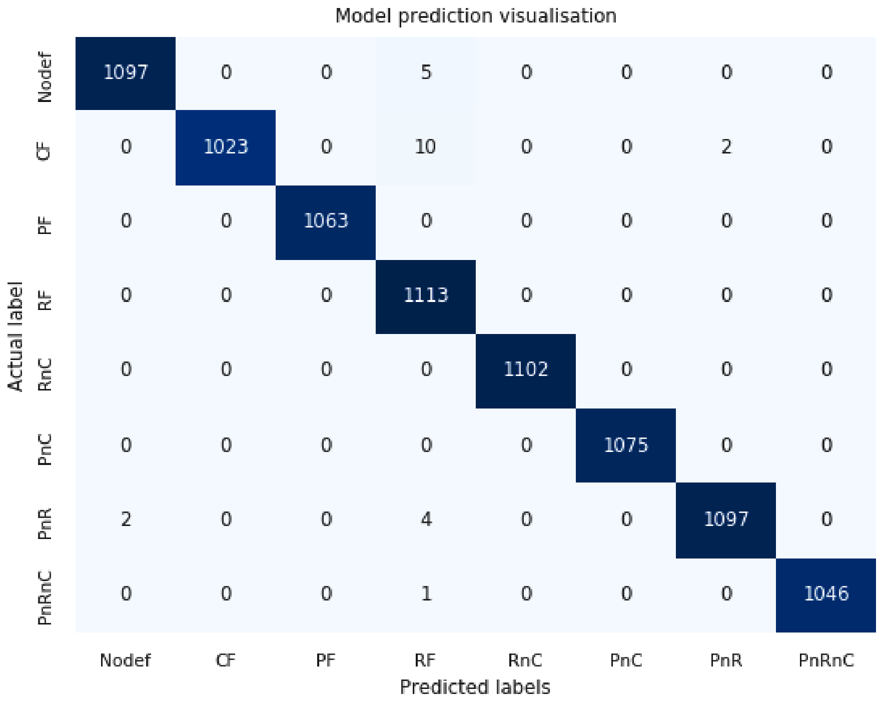
where the p_o is the observed proportion on which observers agree, p_e is the proportion of observation where agreement is expected by chance, $p_o - p_e$ is the proportion of agreement expected beyond chance, and $1 - p_e$ is the maximum possible agreement beyond by chance expectation [369].

Furthermore, this analysis of the model predictions highlights the patterns within the predictions. It also compares the agreement of the model's prediction to the ground truth (reality) to validate that the high prediction accuracy was correct.

The rationale for using the Kappa coefficient is that Kappa relates the predictions of two raters comparable to the model ground truth and the predictions, thereby making Kappa useful in the model confirmatory tests. Besides, the developed model and evaluation data meet the vital criteria for evaluating models using Kappa, including having a sample size of over one hundred, as outlined by researchers [370]. Finally, it is worth outlining that the contingency table requires conversion to the confusion matrix in supervised classification problems.

6.4.1 Transfer Learning

The VGGNet model transfer learning results outlined that the model predictions were 99.72% accurate; however, the evaluation of the prediction using the model statistics is considered to confirm the performance. The VGGNet model results shown using the confusion matrix visualisation in Figure 6.18 highlight the model's performance on the inspection data.


Model prediction visualisation
Actual label
Predicted labels

	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC
Nodef	1097	0	0	5	0	0	0	0
CF	0	1023	0	10	0	0	2	0
PF	0	0	1063	0	0	0	0	0
RF	0	0	0	1113	0	0	0	0
RnC	0	0	0	0	1102	0	0	0
PnC	0	0	0	0	0	1075	0	0
PnR	2	0	0	4	0	0	1097	0
PnRnC	0	0	0	1	0	0	0	1046

Figure 6.18 VGG model prediction visualisation using the confusion matrix

Conversely, from the model predictions confusion matrix, the overall proportion of the observed agreement of the transfer learning model is given by

$$p_o = \sum_{i=1}^{k} p_{ii} = 1097+1023+1063+1113+1102+1075+1097+1046 = 8616/8640$$

$$p_o = 0.9972$$

Also, the overall expected agreement by chance is given by

$$p_e = \sum_{i=1}^{k} p_{i.}p_{i}$$

$$p_e = 0.127 * 0.128 + 0.118 * 0.12 + 0.123 * 0.123 + 0.129 * 0.131 + 0.128 * 0.128 + 0.124 * 0.124 + 0.127 * 0.128 + 0.121 * 0.121$$

$$p_e = 0.016 + 0.014 + 0.015 + 0.017 + 0.016 + 0.015 + 0.016 + 0.015$$

The expected agreement by chance is $p_e = 0.125$

Hence, substituting the obtained expected agreement and expected agreement by chance, the Kappa coefficient becomes $K = \frac{P_o - P_e}{1 - P_e} = \frac{0.9972 - 0.125}{1 - 0.125}$ = 0.99.68

The Kappa coefficient of 0.9968 obtained from the model predictions highlights a significant agreement between the predictions and reality, validating that the model's performance is not by chance. The evaluation of Kappa is shown in Table 6.3.

Table 6.3 Evaluation of Kappa coefficient for the transfer learning model.

	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC	Sum 1	Pi=s
Nodef	1097	0	0	5	0	0	0	0	1102	0.128
CF	0	1023	0	10	0	0	0	2	1035	0.12
PF	0	0	1063	0	0	0	0	0	1063	0.123
RF	0	0	0	1113	0	0	0	0	1113	0.129
RnC	0	0	0	0	1102	0	0	0	1102	0.128
PnC	0	0	0	0	0	1075	0	0	1075	0.124
PnR	2	0	0	4	0	0	1097	0	1103	0.128
PnRnC	0	0	0	1	0	0	0	1046	1047	0.121
Sum 2	1099	1023	1063	1133	1102	1075	1099	1046	8640	
p.i	0.127	0.118	0.123	0.131	0.128	0.124	0.127	0.121		

Moreover, Table 6.3 shows that the classifier's predictions agree with the ground truth up to 99.72%, with the remaining 0.28% being by chance in percentage terms by deduction. This is because the Kappa agreement factor is the same as the accuracy of the model predictions in simple terms for balanced classification problems. A Kappa of less than zero means that the model is worse than chance and is a rare case when evaluating a model.

6.4.2 Training from Scratch Results

The results obtained from training the VGGNet from the scratch outline that the model performed excellently. It is due to the rich architectural make-up of the VGGNet model, having many more hidden layers in the architecture. Enabling the training of the hidden layers enhanced feature learning from the data, with the effect outlined as improved model performance and visible in the model confusion matrix of Figure .

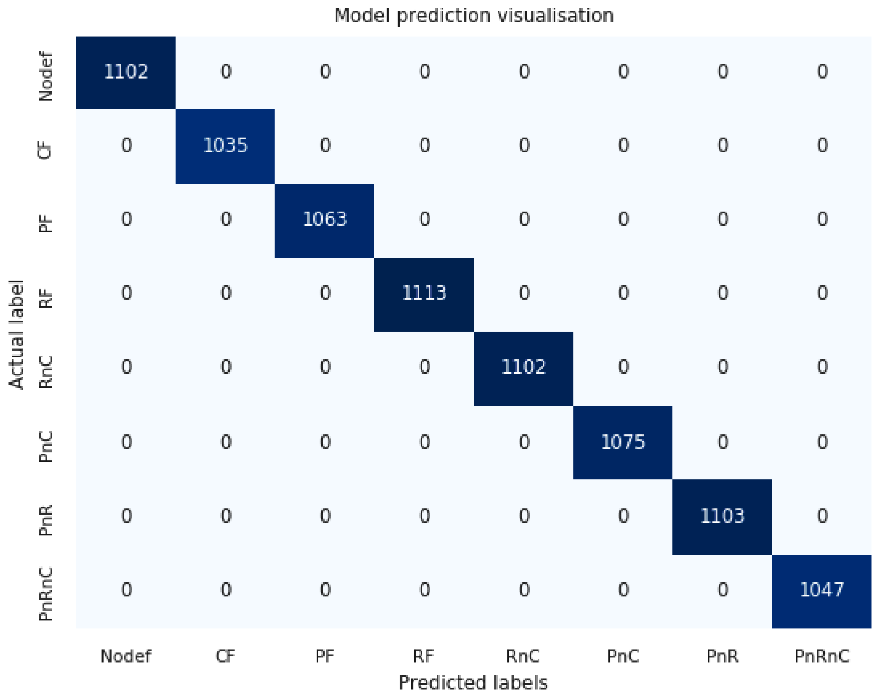


Figure 6.19 Model prediction visualisation for the training from scratch using the confusion matrix

However, in the model predictions confusion matrix of Figure .19, the overall proportion of the observed agreement of the researcher developed model is given by

$$p_o = \sum_{i=1}^{k} p_{ii} = 1102+1035+1063+1113+1102+1075+1103+1047 = 8640/8640$$

$$p_o = 1$$

Also, the model's overall expected agreement by chance is given by

$$p_e = 0.122 * 0.122 + 0.1198 * 0.1198 + 0.123 * 0.123 + 0.1288 * 0.1288 + 0.1276 * 0.1275 + 0.1244 * 0.1244 + 0.1277 * 0.1277 + 0.1212 * 0.1212$$

$$p_e = 0.0163 + 0.0144 + 0.0151 + 0.0166 + 0.0163 + 0.0155 + 0.0163 + 0.0147$$

The expected agreement by chance is $p_e = 0.1251$

Similarly, substituting the obtained expected agreement and expected agreement by chance, the Kappa coefficient becomes $K = \frac{P_o - P_e}{1 - P_e} = \frac{1-0.1251}{1-0.125} = 1$

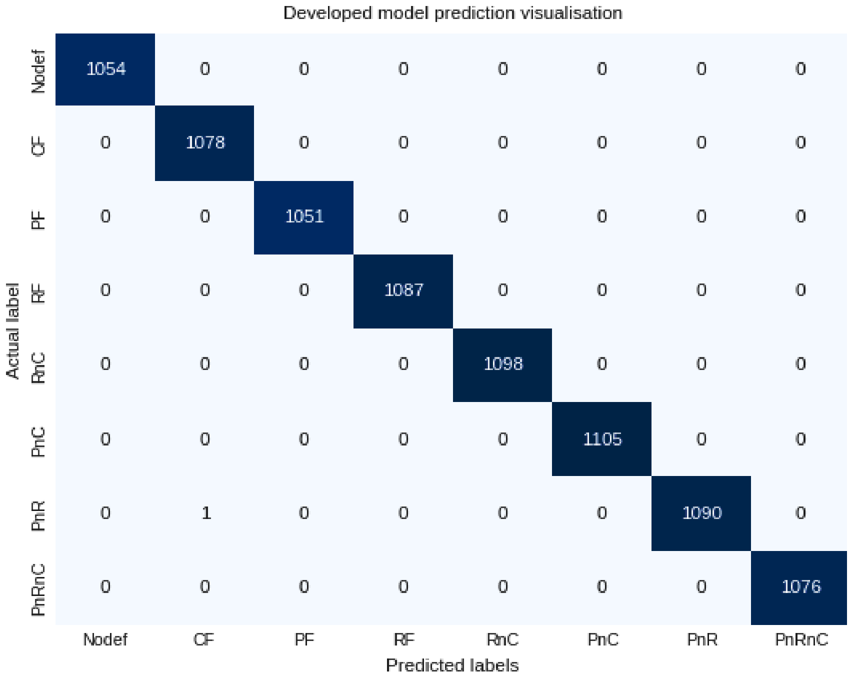
The Kappa coefficient of 1 obtained from training the model from scratch predictions outlines a perfect agreement between the model predictions and the reality, thereby suggesting that the model's performance is not by chance. The result also shows an improved performance compared to the pre-trained model. The evaluation of the Kappa coefficient from training the model from scratch is shown in Table 6.4.

Table 6.4 Evaluation of Kappa coefficient for the VGGNet models training from scratch

	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC	Sum 1	Pi
Nodef	1102	0	0	0	0	0	0	0	1102	0.1275
CF	0	1035	0	0	0	0	0	0	1035	0.1198
PF	0	0	1063	0	0	0	0	0	1063	0.1230
RF	0	0	0	1113	0	0	0	0	1113	0.1288
RnC	0	0	0	0	1102	0	0	0	1102	0.1276
PnC	0	0	0	0	0	1075	0	0	1075	0.1244
PnR	0	0	0	0	0	0	1103	0	1103	0.1277
PnRnC	0	0	0	0	0	0	0	1047	1047	0.1212
Sum 2	1102	1035	1063	1133	1102	1075	1103	1047	8640	
p.i	0.1276	0.1198	0.1230	0.1288	0.1276	0.1244	0.1277	0.1212		

6.4.3 Developed Model Results

The training time factor was not included as a metric for performance evaluation because the usability of the model was the most important factor considered. The ability of the model to replicate the inputs is vital and informs the decisions made during the design and testing of the models presented in this chapter.



	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC
Nodef	1054	0	0	0	0	0	0	0
CF	0	1078	0	0	0	0	0	0
PF	0	0	1051	0	0	0	0	0
RF	0	0	0	1087	0	0	0	0
RnC	0	0	0	0	1098	0	0	0
PnC	0	0	0	0	0	1105	0	0
PnR	0	1	0	0	0	0	1090	0
PnRnC	0	0	0	0	0	0	0	1076

Figure 61.20 Developed model prediction visualisation using the confusion matrix

Nonetheless, from the model predictions confusion matrix in Figure 6.20, the overall proportion of the observed agreement of the researcher developed model is given by

$$p_o = \sum_{i=1}^{k} p_{ii} = 1054+1079+1051+1087+1098+1105+1090+1076 = 8639/8640$$

$$p_o = 0.9999$$

Therefore, the overall expected agreement by chance is given by

$$p_e = 0.122 * 0.122 + 0.1249 * 0.1248 + 0.1216 * 0.1216 + 0.1258 * 0.1258 + 0.1271 * 0.1271 + 0.1279 * 0.1279 + 0.1262 * 0.1263 + 0.1245 * 0.1245$$

$$p_e = 0.0149 + 0.0156 + 0.0148 + 0.0158 + 0.0162 + 0.0164 + 0.0159 + 0.0155$$

The expected agreement by chance is $p_e = 0.125$

Similarly, substituting the obtained expected agreement and expected agreement by chance, the Kappa coefficient becomes $K = \frac{P_o - P_e}{1 - P_e} = \frac{0.9999-0.125}{1-0.125} = 0.9998$

The Kappa coefficient of 0.9998 obtained from the model predictions outlines a total agreement between the model predictions and reality, further validating that the model's performance is not by chance. The evaluation of the Kappa coefficient for the researcher developed model is depicted in Table 6.5.

Table 6.5 Evaluation of the Kappa coefficient for the developed model

	Nodef	CF	PF	RF	RnC	PnC	PnR	PnRnC	Sum 1	Pi
Nodef	1054	0	0	0	0	0	0	0	1054	0.1220
CF	0	1078	0	0	0	0	0	0	1078	0.1248
PF	0	0	1051	0	0	0	0	0	1051	0.1216
RF	0	0	0	1087	0	0	0	0	1087	0.1258
RnC	0	0	0	0	1098	0	0	0	1098	0.1271
PnC	0	0	0	0	0	1105	0	0	1105	0.1279
PnR	0	1	0	0	0	0	1090	0	1091	0.1263
PnRnC	0	0	0	0	0	0	0	1076	1076	0.1245
Sum 2	1054	1078	1051	1087	1098	1105	1090	1076	8640	
p.i	0.1220	0.1249	0.1216	0.1258	0.1271	0.1279	0.1262	0.1245		

A cross-case deduction to evaluate the model's performance across the three datasets highlights that the developed model is comparable to the state-of-the-art model in performance even with a less complicated architecture, providing a robust and faster computational model for remanufacturing applications. Hence, the prediction results from the three remanufacturing applications validate that the developed model performs comparably to the state-of-the-art models on the selected applications.

6.5 Cost Benefit Analysis

The cost of remanufacturing refers to the overall cost of achieving product or part remanufacturing. It is a total cost comprising pre-production, production, and overhead costs. These costs are combined to produce the overall remanufacturing cost [371]. Besides, the overall remanufacturing cost includes reverse engineering, data processing, setup, deposit, grinding, administration' machine maintenance, and other overhead costs. It is difficult to holistically support the cost-benefit estimation since most factors are inherent in manual and automated remanufacturing. Based on the above research proposition [371], the criterion eliminates the pre-production and overhead costs, leaving the remanufacturing production cost for further evaluation towards estimating the cost-benefit of the model and other non-quantifiable costs through the automation.

Moreover, the composition of these costs differs and depends on the processes involved in remanufacturing a given product. The cost-benefit estimation considers only the required modification of the existing remanufacturing cell to achieve automation for inspection, sorting and process control applications. The reported cost-benefit analysis helps evaluate the benefits of adopting the deep learning approach to automate remanufacturing processes. However, the cost-benefit analysis is unsuitable for this research because of the difficulties in quantifying the dollar cost of the benefits alongside predicting all the potential risks involved in remanufacturing, impacts and customer satisfaction.

6.5.1 Cost Benefit Justification.

The justification for exploring other methods of quantifying the impacts of automation remanufacturing is obtainable using the problem selection matrix, a vital tool for evaluating and selecting the best option between various choices. The problem selection matrix is often referred to as the Pugh matrix, decision matrix, grid analysis, decision grid and multi-attribute utility theory.

Nevertheless, the decision to automate the remanufacturing or not can be justified using the problem selection matrix considering six critical factors to support automating or not automating the remanufacturing processes investigated in the research. These vital factors include the quality of products, cost, exposure impact, especially the risk of accidents due to manual operations, time savings, the environmental impact of additional pollution from the new machines added for automation, and customer satisfaction. The problem selection matrix for the justification is outlined in Table 6.6.

Table 6.6 Cost-benefit problem selection matrix

	Quality	Cost	Exposure impact	Time savings	Environmental impact	Customer satisfaction
Automate						
No-automation						

Furthermore, by rating the factors on a scale of 1 to 5, where five is the best and one is the least, the research results are translated as follows:

- Automation provides consistent quality of products (5), a more expensive process (2), with minimal exposure to the risk of work accidents (4). It also offers better time savings (4), increased environmental pollution due to more energy requirements (2) and finally, better customer satisfaction as product remanufacturing is performed faster (4).

- No-automation provides less consistent quality of products due to poorer inspection (3), less expensive (4), high exposure to the risk of accidents working with slippery Eol products (2), taking longer time (2), and less additional environmental pollution (3). Finally, lesser customer satisfaction as product remanufacturing takes longer (3).

These criteria and their rankings are outlined in Table 6.7 as follows.

Table 6.7 Factor ranking for automation and no-automation

	Quality	Cost	Exposure impact	Time savings	Environmental impact	Customer satisfaction
Automate	5	2	4	4	2	4
No-automation	3	4	2	2	3	3

However, since some factors are deemed more critical, practitioners are mainly concerned with cost before other factors since most of them are third-party remanufacturers, while customers are concerned with quality. Therefore, a weighted score was helpful to estimate the potential impact of the various factors considered, not possible to evaluate in dollar terms to decide the better option to pursue.

Table 6.8 Weighted factor ranking for automation and no-automation

	Quality	Cost	Exposure impact	Time savings	Environmental impact	Customer satisfaction	Score
Weights	7	6	4	3	2	5	-
Automate	35	12	16	12	4	20	**97**
No-automation	21	24	8	6	6	25	90

Finally, the weighted score is obtained, which emphasizes the more essential considerations, helping to select the best option. The highest total score obtained gives the best option to adopt. From Table 6.8, it is evident that the benefit of automating the remanufacturing processes outweighs the no-automation option, thereby supporting the decision to automate remanufacturing processes using deep learning algorithms.

6.6 Basis for Testing Research Success

The basis for testing the research success is categorised into two sections to highlight the distinctness of the academic and practitioner needs of the study. First, academics assess the quality and credibility of the research based on crucial elements, the rigorous method of gathering high-quality data, analysis, and the credibility of the researcher, including training, status and presentation of self alongside the philosophical beliefs, including inductive analysis and holistic thinking among others [372]. This academic validity is achieved by collecting and analysing the data for two different inspection applications of deep learning models in remanufacturing described in Chapter 4 to achieve data triangulation. A synthesis of these applications was evaluated to identify the similarities and differences and how the different models affect the results, providing academic validity.

On the other hand, the research domain discussed in Section 2.5.3 outlines the research domain as production and operation management, which tries to bridge the gap between the theory of operations management and practices [56], [57]. The practitioner's needs are evaluated using the validation-by-review approach. This validation provides a platform to test the usefulness of research in the industry and highlight the practical relevance, with POM researchers suggesting practitioners as the frame of reference [373]. To consider practitioners as a reference, the authors outlined practitioners' vital needs, summarised in five key categories: descriptive relevance, goal relevance, operational validity, non-obviousness, and timeliness. The following explains the implications of the five needs of practitioners, namely:

- Descriptive relevance - Is the modelling approach a sufficient representation of the inspection, sorting and process control processes?
- Goal relevance - Is the model applicable to the stakeholders?
- Operational validity - Is the model presented so practitioners and academics can operate and use it?
- Timeliness - Is the model available when remanufacturers need them?
- Non-obviousness - Is the model a new knowledge or simple, common-sense knowledge available to practitioners?

6.7 Model Research Validation

The developed deep learning models for the various remanufacturing applications have had the model validation included at the design stage using a train test split, which is helpful to keep some parts of the data for testing. In contrast, the remaining are used for training and validation during training. The test set, usually referred to as the hold-out set in some literature, is used to assess the model generalisation on unseen data, which is the vital aim of model evaluation. However, the samples used in the research are not probabilistic; researchers suggest that statistical inference is not the appropriate method to generalise the results; instead, other techniques of generalising the results should be explored [54].

Moreover, Section 6.6 above outlines that research validation tests the researcher's quality and credibility, which helps to generalise the research findings. The validation compares the respective applications of deep learning models in remanufacturing on the similarities, differences, and methods affecting the results presented in Chapter 4, where two different inspection data were evaluated. This comparison is vital to achieve result generalisation and support the academic validations obtained from the review process during the review of the publication process of the respective chapters.

Table 9 shows the model parameters for the individual application of deep learning models in remanufacturing.

Table 6.9 Model data components and parameter outline

Process	Data type	Number of classes	Number of Images	Number of images per class	Number of inputs	Number of outputs
Sorting	Images	20	71560	3578	20	20
Inspection I	Images	8	28800	3600	8	8
Inspection II	Images	8	28624	3578	8	8
Process control	Images	2	14312	7156	2	1

Furthermore, Table 6.9 shows that the deep convolutional neural network model used in the investigation is similar in a great sense but requires slight architectural modifications at the input and output layers.

The control over the model's input solely depends on the new application. At the same time, the output depends on the design's expectations, where the desired outcome is coded into the model as observed in the sorting and process control applications that were evaluated using the same model. Besides, it is also evident that the respective applications have different input-output variables representing the individual application needs. Besides, the sorting model requires twenty categories, the two inspection applications require eight categories, and the process control requires only two input-output variables.

6.7.1 Experimental Validation

The experimental validation of the model considers the data triangulation approach, where multiple data sources were useful to evaluate the model's performance. The developed model considers the model accuracy and loss as two dependent variables, while the number of epochs was used as the independent variable. The other dependent variables used in the evaluations are the model's parameters that control its performance, including the type of optimisation, activation function, batch sizes, dropout, batch normalisation, and others. These parameters comparison provides the basis for the validation to infer the generalisability of the developed model.

Conversely, the architectural design parameters analysis presented in Chapter 7 outlines that the model parameters used in the respective applications performed relatively well across the various model parameters. However, the metric for evaluating the classification problem is predominantly the classification accuracy and loss, and these parameters were used to assess the data triangulation.

Moreover, the results highlight that the training from scratch obtained the best accuracy while transfer learning and the researcher-developed model progressively obtained high accuracy, highlighting that the developed model performs comparatively to the state-of-the-art VGGNet model on the test applications, as shown in Figure 62. The VGG and VGGS represent transfer learning and training from scratch.

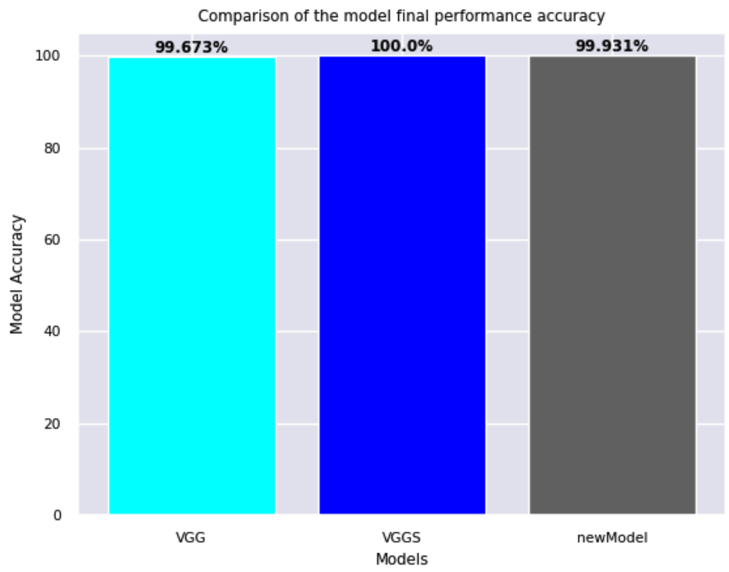


Figure 621 Comparison of the three models' final training accuracy

Consequently, a comparison of the final model losses shown in Figure 63 outlines that the VGGNet transfer learning (VGG) produced the highest model loss of the three models, confirming the poorer prediction accuracy obtained from using the VGGNet transfer learning modelling method. Besides, the training of the VGGNet model from scratch produced the lowest loss from the compared models, thereby suggesting that the training from scratch has over-capacity for the data, with the newly developed model producing a higher training loss compared to the VGGNet from Scratch. (VGGS).

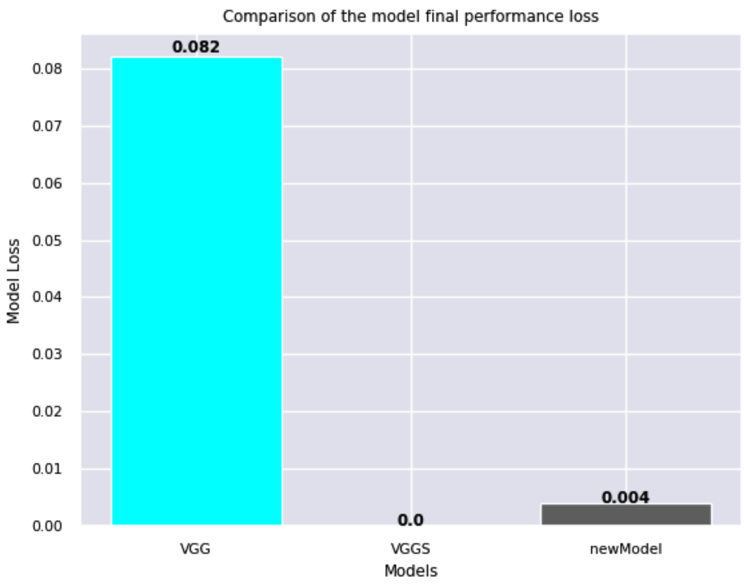


Figure 63 Comparison of the three models' final training losses

Finally, comparing the model's performance on the different datasets against vital model parameters suggests that the deep learning algorithms have successfully modelled the respective remanufacturing applications. The subsequent applications attest that the model can generalise alongside being adaptable to other remanufacturing-based applications.

6.8 Industry Feedback

Industry feedback is another type of validation on its merit. It is a *"process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model"* [374]. These user inputs come from the remanufacturing practitioners, and the validation technique adopted is the modelling-validation process, where model building and model validation are integrated into a single operation. The main goal of the validation process is to improve the model quality and, most importantly, to highlight the model's key components that need improvement and outline the specific stages of the model that has reached the predetermined benchmarks [375]. The modelling-validation process consists of four vital interrelated components: the problem statement, conceptual model, formal model, and solution, outlined in Figure 64.

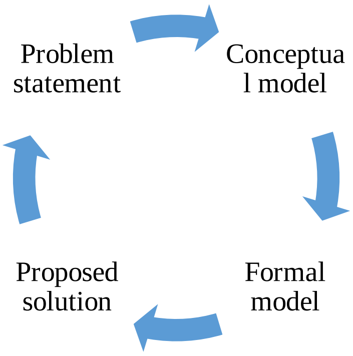


Figure 64 Model validation

The problem case refers to the poor productivity of remanufacturing processes and systems, leading to operational dissatisfaction with the performance. On the other hand, the conceptual model represents the developed mental picture of the problem investigated and the value judgement from both the decision-makers and model developers. For example, the conceptual model shown in Figure 6.24 describes the problem approach, the elements that would be excluded or included, and the level of aggregation. These components are captured in the problem or process under investigation before data collection, analysis and deployment.

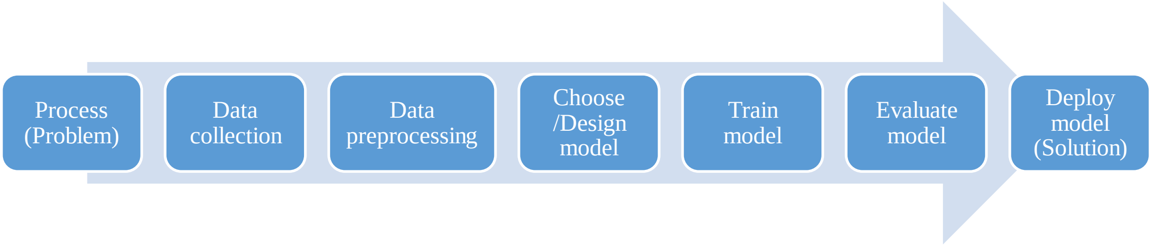


Figure 6.24 The conceptual model design

Moreover, the model proposed conceptual solution must first satisfy the design's conceptual validity, which refers to the relevance of the theories and assumptions underlying the conceptual model for the problem case. Finally, the problem is modelled to comprise the relationships and elements judged relevant by the end-users and conform to the available techniques and tools.

Besides, another vital requirement the conceptual model must satisfy includes the model's logical and data validity. Logical validity refers to the ability of the formal model to correctly and accurately describe the problems described by the conceptual model. It also relates the correctness of the translation from the conceptual model to a formal model, and it is primarily dependent on the translation language, which is the computational algorithms. This translation aims to maintain a faithful transfer of the critical elements of the process into the model and the verification of these vital elements enhances the logical validity. By contrast, data validity refers to the accuracy, sufficiency, appropriateness, and availability of the data within acceptable cost limits. The assessment of the difficulties involved in collecting and processing the process data and resolving the challenges alongside their impact are vital components of data validity.

Nonetheless, the formal model refers to translating the conceptual model into mathematical models for further investigation to obtain solutions useful for effective decision-making. Therefore, the formal model shown in Figure 6.29 must also meet the experimental validity requirements of the design. Moreover, experimental validity refers to the efficiency and quality of the proposed solution [375], and it highlights the efficiency of obtaining the desired solutions sensitive to changes in model parameters.

Finally, the solution is the output of the model validation process that forms the basis for recommending an answer to one of the problems under consideration. Therefore, it is expected to meet the operational validity goals, which refer to the users' ability to implement the theory's action implications by operating the independent variables.

Besides, the model-validation process involves multiple validation stages that include conceptual, logical, experimental, operational, and data validation, which requires the model developer to have reasonable knowledge and understanding of the acceptable levels of model validity [375]. These multiple validations are detailed in Figure 6.25.

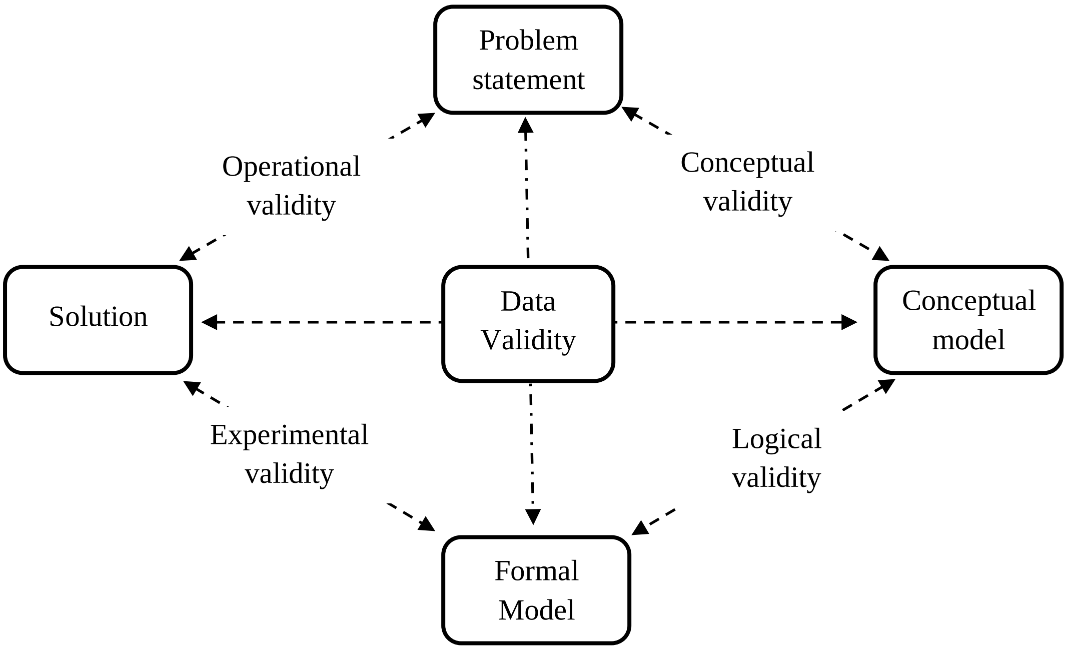
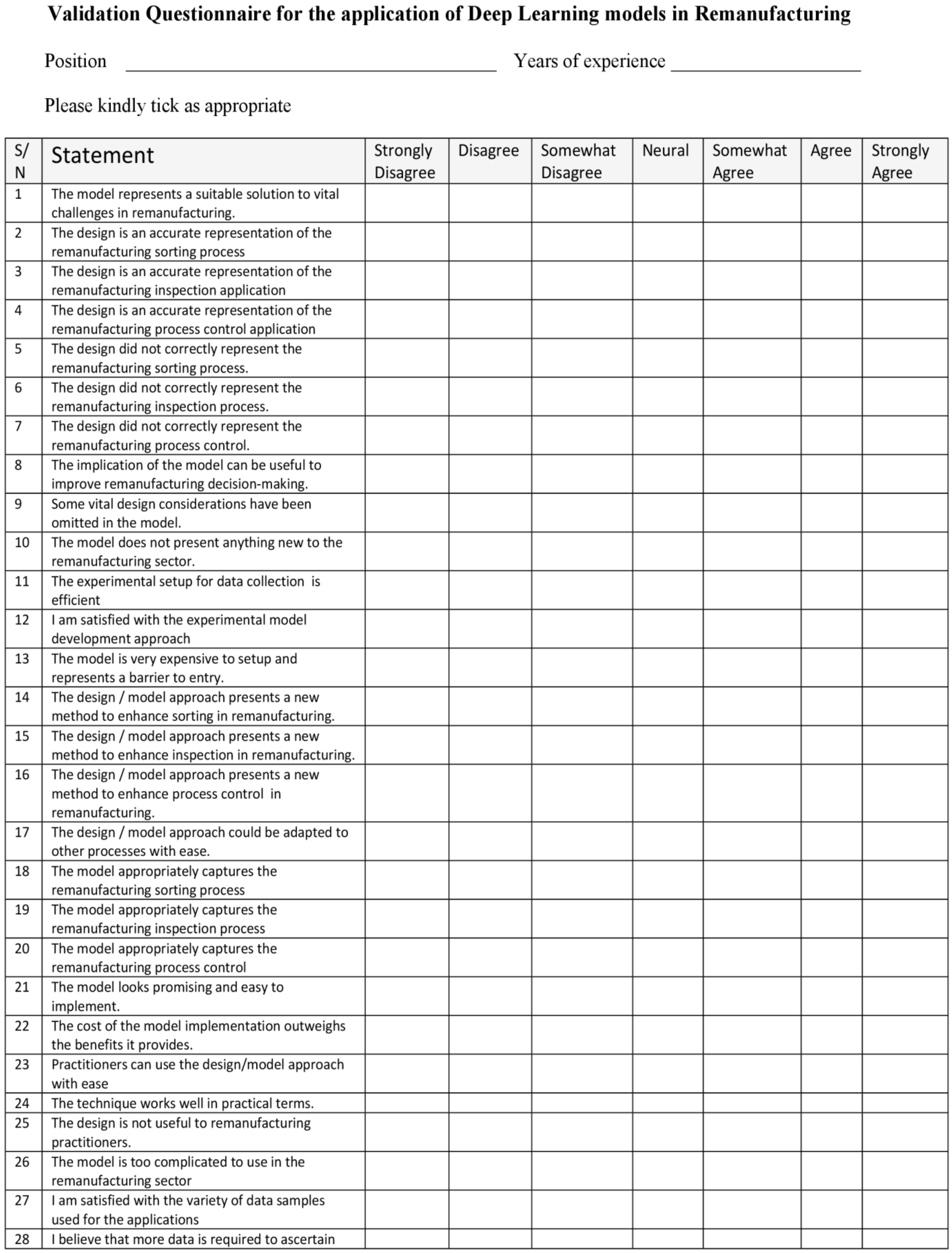


Figure 6.25 Adapted Model validation Process [375]

6.8.1 Model Validation Protocol

The validation adopts the questionnaire tool, a vital research tool for gathering primary data from respondents. It consists of a series of standardised questions for obtaining the same information from a group of individuals [44]. The available data from the questionnaires include awareness or knowledge, attributes, experiences, attitudes, and opinions from the practitioner's point of view. This feedback validates the model's effectiveness by confirming that each aspect attains the predetermined benchmark and highlighting possible areas for improvement through better modelling [375]. The Likert scale was chosen to obtain participants' feedback because it allows for degrees of opinion, either in agreement or disagreement, thereby aiding the analysis of the various views of evaluators on a piece of given information [376].

Generally, the design of questionnaires adopts two common types of questions; open-ended and close-ended question methods as outlined by researchers [45]. The kind of questionnaire to use provides specific advantages during analysis. The design uses the close-ended question approach to obtain the practitioner's feedback on questions that would help answer the validation questions, including data, experimental, logical, conceptual, and operational validity. These questions were combined in random order to obtain the validation protocol used in the research. Figure 6.26 shows the Industry feedback (validation) protocol used in the study.



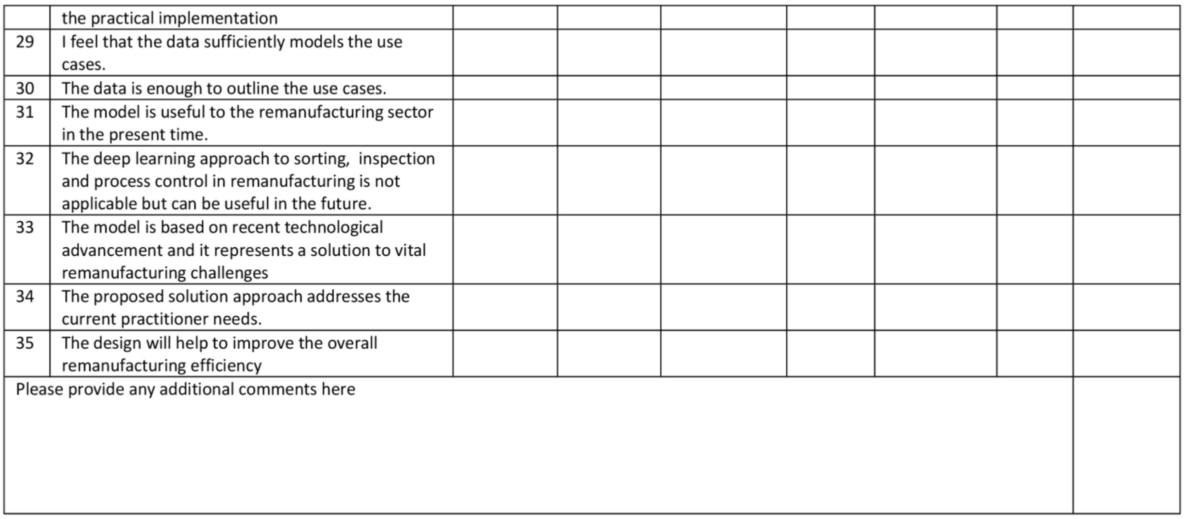
Validation Questionnaire for the application of Deep Learning models in Remanufacturing

Position ______________________________________ Years of experience ____________________

Please kindly tick as appropriate

S/N	Statement	Strongly Disagree	Disagree	Somewhat Disagree	Neural	Somewhat Agree	Agree	Strongly Agree
1	The model represents a suitable solution to vital challenges in remanufacturing.							
2	The design is an accurate representation of the remanufacturing sorting process							
3	The design is an accurate representation of the remanufacturing inspection application							
4	The design is an accurate representation of the remanufacturing process control application							
5	The design did not correctly represent the remanufacturing sorting process.							
6	The design did not correctly represent the remanufacturing inspection process.							
7	The design did not correctly represent the remanufacturing process control.							
8	The implication of the model can be useful to improve remanufacturing decision-making.							
9	Some vital design considerations have been omitted in the model.							
10	The model does not present anything new to the remanufacturing sector.							
11	The experimental setup for data collection is efficient							
12	I am satisfied with the experimental model development approach							
13	The model is very expensive to setup and represents a barrier to entry.							
14	The design / model approach presents a new method to enhance sorting in remanufacturing.							
15	The design / model approach presents a new method to enhance inspection in remanufacturing.							
16	The design / model approach presents a new method to enhance process control in remanufacturing.							
17	The design / model approach could be adapted to other processes with ease.							
18	The model appropriately captures the remanufacturing sorting process							
19	The model appropriately captures the remanufacturing inspection process							
20	The model appropriately captures the remanufacturing process control							
21	The model looks promising and easy to implement.							
22	The cost of the model implementation outweighs the benefits it provides.							
23	Practitioners can use the design/model approach with ease							
24	The technique works well in practical terms.							
25	The design is not useful to remanufacturing practitioners.							
26	The model is too complicated to use in the remanufacturing sector							
27	I am satisfied with the variety of data samples used for the applications							
28	I believe that more data is required to ascertain							

Figure 6.26 Industry feedback protocol



	the practical implementation							
29	I feel that the data sufficiently models the use cases.							
30	The data is enough to outline the use cases.							
31	The model is useful to the remanufacturing sector in the present time.							
32	The deep learning approach to sorting, inspection and process control in remanufacturing is not applicable but can be useful in the future.							
33	The model is based on recent technological advancement and it represents a solution to vital remanufacturing challenges							
34	The proposed solution approach addresses the current practitioner needs.							
35	The design will help to improve the overall remanufacturing efficiency							
Please provide any additional comments here								

Thank you for your time.

6.8.2 Industry Feedback Process

The validation plan was planned through email communication to Mackie Transmission Limited to arrange the day for the evaluation and discussion of the model performance and the expectations during the validation. The validation took place at the host company facility due to the tight schedules and loss of personal hours if the exercise was scheduled outside the facility.

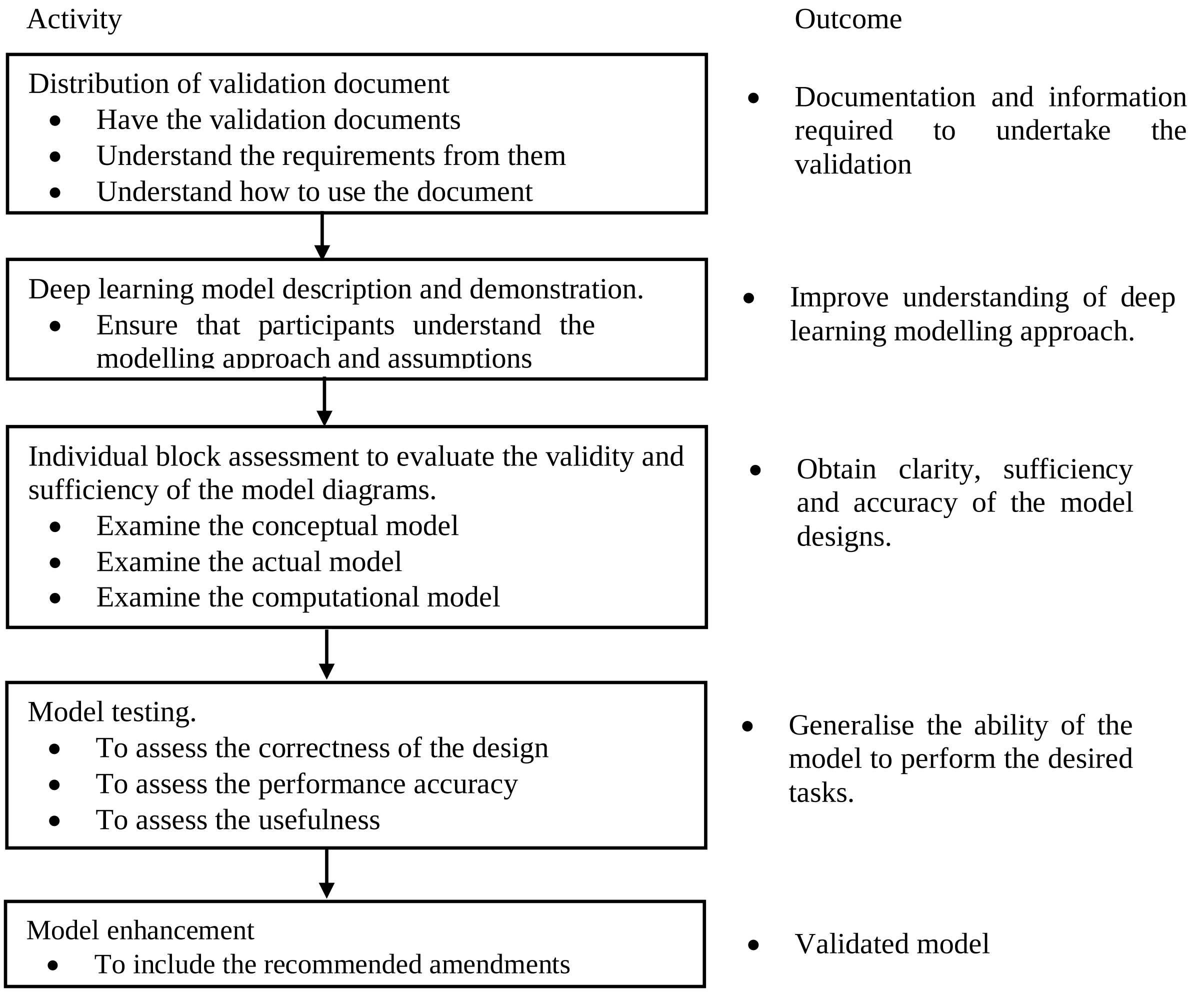

Activity
Outcome
Distribution of validation document
Have the validation documents
Understand the requirements from them
Understand how to use the document
Documentation and information required to undertake the validation
Deep learning model description and demonstration.
Ensure that participants understand the modelling approach and assumptions
Improve understanding of deep learning modelling approach.
Individual block assessment to evaluate the validity and sufficiency of the model diagrams.
Examine the conceptual model
Examine the actual model
Examine the computational model
Obtain clarity, sufficiency and accuracy of the model designs.
Model testing.
To assess the correctness of the design
To assess the performance accuracy
To assess the usefulness
Generalise the ability of the model to perform the desired tasks.
Model enhancement
To include the recommended amendments
Validated model

Figure 6.27 The industry feedback process

6.8.3 The Industry Validation Exercise

The validation exercise started with the author presenting the model design considerations obtained from the challenges identified from the literature and process observations to the five-person focus group of remanufacturing practitioners that work at Mackie transmission Limited. Next, the conceptual design, model, and actual design were presented before testing the developed model. The practitioners were also allowed to ask questions for further clarification, and afterwards, the questionnaires were issued. The author outlined the purpose of the questionnaires and described the methods of proper completion. The author allowed the practitioners to complete the questionnaires without bias and email the feedback to the author. Finally, the practitioners completed the questionnaires as a focus group and returned consensus feedback interpreted in the subsequent sections.

The questionnaire feedback sheet contained thirty-five questions used to record the participant feedback on using deep learning algorithms for modelling remanufacturing inspection, sorting and process control applications alongside a space for additional comments. The feedback assesses the model in three vital criteria for suitability, clarity and sufficiency. This process involves asking the participants the same question in different words about the model to assess their understanding. The last question was an empty comment box for the participants to document any additional information they wished to add. Once the feedback sheets were completed and returned, the data from the validation exercise were collated and used to improve the model. The completed feedback sheets are contained in Appendix 1.

6.9 Results of Model Validation

The validation feedback focus group believed that the model design described by the research accurately represents the remanufacturing inspection, sorting, and process control applications alongside the minor details involved in achieving the processes. This is highlighted by the information given in the validation feedback in Appendix 1. Besides, the focus group also agreed that the model presents new techniques to enhance the remanufacturing sorting, inspection and process control, and the modelling finds practical applications in remanufacturing.

Conversely, the sufficiency of the data for modelling the processes provided some conflicting feedback as the focus group outlined a moderately high acceptance that the data was sufficient for the use cases but were not sure if the variety of data was enough to generalise to other remanufacturing processes by giving an average or neutral support. However, they further suggested that more data was required to generalise the implementations. This is because of the other remanufacturing processes have not been explored. Therefore, more research is needed to accept the models as an efficient method of improving the other remanufacturing processes aside from the considered applications.

Besides, the focus group provided moderately high support for the experimental model development and setup for data collection; however, they outlined their concerns about the fixed camera blind spot and the cost of deploying these models as it could be expensive for smaller remanufacturing companies, constituting a vital limitation to adoption. This concern is not unexpected as the company is not very large to invest in such a technology. However, they unanimously agreed that the modelling approach is a recent technological advancement and serves as a solution to vital remanufacturing challenges.

6.10 Alterations to Enhance Clarity

The proposed alteration to the actual design included a manipulator (robot) fitted with end effectors to grip and flip components for the camera to capture the camera blind spot, indicated by the two upward arrows in Figure 6.2. This approach enhances the 360 degrees inspection since the model incorporated a single fixed camera during testing.

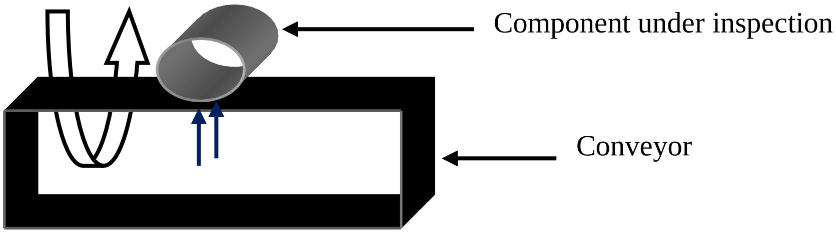


Figure 6.28 Identified camera blind spot for improvement

This alteration was necessary for the fixed camera to inspect the parts of the products placed directly on the conveyor to ascertain the product surface conditions in those underlying areas. The modified design approach is shown in Figure 6.29.

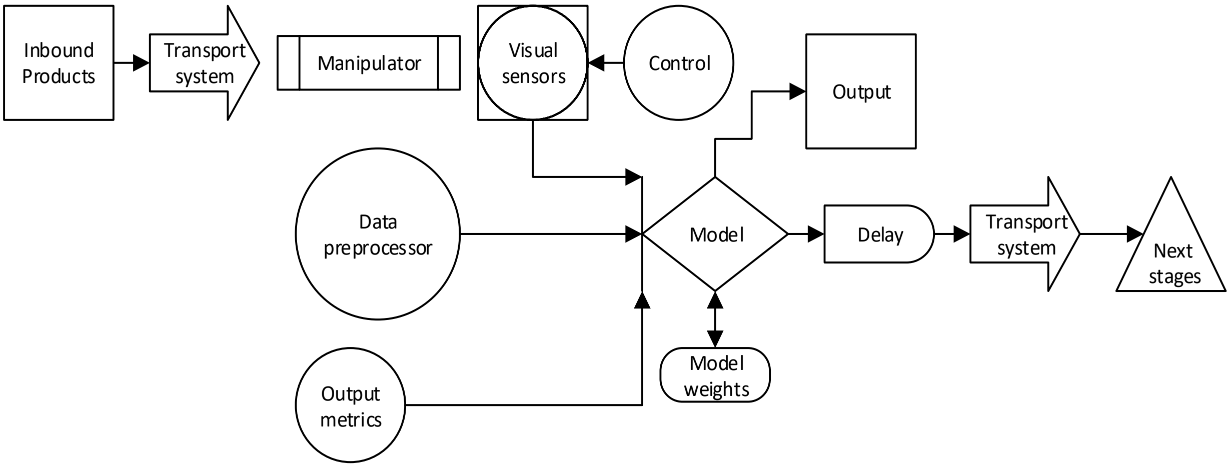


Figure 6.29 Formal model

6.11 Modelling Inspection, Sorting and Process Control in Remanufacturing

Before the validation exercise, the participants were unaware of deep learning modelling. However, they found the model approach understandable after the introductory presentation and modelling conception, design, testing and discussions about the final model. The researcher believed that clarity of the model was vital; however, the technical nature of their jobs is also contributory, as the designs were represented using different levels of block diagrams. The practitioner's opinion outlines that deep learning-based modelling was adequate for modelling the inspection, sorting and process control applications in remanufacturing.

These opinions were interpreted from the feedback received during the validation. The questionnaire results were converted from the 7-scale Likert of 1 to 7 to verbal interpretations. The lowest rank corresponds to the strongly disagreed inputs, and the highest represents the strongly agreed inputs. The respective windows with these verbal interpretations are shown in Table 6.10.

The window of each input is obtained as

$$window = \frac{range}{max} = \frac{7-1}{7} = 0.86$$

The window range helps to categorise the feedback into a scale that can be easily interpreted using verbal interpretation.

Table 6.10 Interpretation of the Likert scale feedback

Scale	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Range	1.0 - 1.86	1.87 -2.72	2.73 - 3.58	3.59 - 4.44	4.45 - 5.3	5.31 - 6.16	6.17 - 7.0
Response	Strongly Disagree	Disagree	Somewhat Disagree	Neural	Somewhat Agree	Agree	Strongly Agree
Verbal Interpretation	Very low	Low	Moderately low	Average	Moderately high	High	Very high

6.11.1 Descriptive Relevance

The descriptive relevance describes the accuracy of the research findings in expressing the phenomena witnessed by practitioners. The validation feedback outlines that the developed deep learning models sufficiently represent the sorting, process control and inspection process in remanufacturing and, therefore, could help describe the processes. The feedback that "the design is an accurate representation of the sorting, inspection and process control applications" received moderately high and high support for the model's representation of the processes. Also, the exact opposite of the question "the design does not correctly represent the remanufacturing inspection, sorting and process control" received holistic moderate low feedback across all the applications. Conversely, the practitioners outlined that the model omitted some essential design considerations to achieve a holistic view of the components during the inspection. Despite the minor improvement recommended by the practitioners to improve the robustness of the developed model, they felt that it did not constitute a significant error in the design. The suggested alteration is described in Section 6.10 alongside the author's improved amendments. These responses highlight that the practitioners believe that the modelling approach brings potential and generalisable methods that could enhance various process scenarios in remanufacturing.

Furthermore, the practitioners have a moderately high belief that the modelling approach is a recent technological advancement and represents a solution to vital remanufacturing challenges, thereby outlining that the model sufficiently presents crucial answers to some of their challenges. Overall, the deep learning models offer a new and practical approach to addressing remanufacturing inspection, sorting, and process control automation challenges, thereby enhancing productivity and efficiency.

6.11.2 Goal Relevance

Goal relevance refers to how relevant the model would be to the stakeholders. The stakeholders include everyone that has an interest in the research. Thomas (1979), cited in [373], outlined that research is helpful to practitioners if only research outcomes correspond to practitioners' concerns. It makes a case for the immediate applicability of the study, an essential component for the research to be valid.

Nevertheless, the practitioners believed that the deep learning modelling approach is an effective technique to enhance efficiency in remanufacturing as the feedback outlines a moderately high agreement to three of the questions on the model's usefulness. Furthermore, the practitioners also have a high agreement that "the model represents a suitable solution to vital challenges in remanufacturing", "the implication of the model can be useful to improve remanufacturing decision-making", and provided moderately low support that the model is not helpful to remanufacturing practitioners, confirming that the model can contribute significantly in automating various processes in remanufacturing. Finally, the practitioners also provided average support that the design will improve the overall remanufacturing efficiency.

Nevertheless, the practitioners also supported averagely that the modelling approach to inspection, sorting and process control is not applicable but can be helpful in the future. This feedback is not out of place for the case-study company as it is a small private establishment with a strict budget for technological expansion. The lack of funds was evidenced by their high agreement that "the cost of the model implementation outweighs the benefits", alongside their profound concern that "the model is expensive to set up and represents a barrier to entry", suggesting that they might not be willing to invest in the technology at present.

Overall, the practitioner's feedback across the five questions on the model's usefulness highlights that they have a moderately high understanding of the usefulness and benefits of the developed models in remanufacturing.

6.11.3 Operational Validity

Operational validity refers to the ease of practitioners using the new knowledge, and it outlines the ability of practitioners to understand alongside the ease of manipulating the developed models after development. The validation feedback sheet suggests that the practitioners misunderstood the operational validity questions. For example, the first question about the model adaptation, "the design/model approach could be adapted to other processes with ease," received a moderately high agreement alongside the practical use of the model, "the model works well in practical terms", which they provided an average agreement.

However, the practitioner feedback highlights their concerns about ease of implementation and usage when deployed, as they provided a moderately low agreement to the questions that relate to the usage and performance, including that "the model looks promising and easy to implement" and "practitioners can use the design/model with ease". In addition, they provided an average agreement that the model is too complicated to use in the remanufacturing sector. These feedbacks suggest that they are unclear about the model's parameters to modify when exploring new tasks and the requirements for the model deployment. This feedback is an obvious expectation because the author performed every requirement for setup and deployment before the validation, with the practitioners only available to see the demonstration and feedback on the outcomes. However, it also highlights the need for some domain knowledge to develop and deploy deep learning-based models that require additional modification for adaptation in other processes.

Overall, the practitioner feedback highlights that they understood the new knowledge produced by the deep learning-based modelling technique in remanufacturing.

6.11.4 Timeliness

Timeliness refers to the availability of the theory in times of need by practitioners to deal with their challenges [373]. The fact that new knowledge takes time to adopt is a common trend; however, the deep learning model's applicability and adoption were already found in various implementations in remanufacturing through this study, thereby making the model ready for deployment.

Moreover, the practitioners highlighted that these models present current solutions to their everyday challenges. They gave an overwhelming high agreement to the question that "the model is useful to the remanufacturing sector in the present time". They also provided average support that the model addresses the current practitioner needs through the question "the proposed solution approach addresses the current practitioner needs", suggesting that they agree that the model can resolve some of their automation challenges.

These responses outline that the practitioners appreciate the potential of deploying the deep learning models in remanufacturing, thereby validating that the proposed modelling approach can benefit remanufacturing practitioners in enhancing their productivity.

6.11.5 Non-obviousness

Non-obviousness refers to how a theory meets or exceeds practitioners' common sense of knowledge. For example, the practitioners were not familiar with deep learning modelling techniques before the research, suggesting they were not likely to consider using the model to automate parts of the process in their remanufacturing process. However, the validation feedback highlights a high agreement of the practitioners that the modelling method, presentation and discussions outlined new ways of automating sorting, inspection and process control in remanufacturing, thereby enhancing the process efficiency.

Hence, the practitioners disagreed that the model does not present anything new to the remanufacturing sector, highlighting the contributions of the research was significant and beyond their common-sense knowledge.

6.12 Chapter Summary

This chapter presented the analysis of the critical factors considered while selecting the parameters used in the developed algorithm alongside the performance of the developed model, including the type of initialisation, batch size, batch normalisation, activation, activations, loss functions, and optimisation techniques, among others. In addition, the chapter evaluated the learning process across each of the model layers for proper understanding. It also presented the developed model confirmatory test using the Kappa coefficient to highlight that the model performs and achieves component inspection, sorting and process control in remanufacturing, enhancing performance and productivity. The chapter further outlined the cost-benefit analysis of the developed model alongside the research industry feedback methods used to meet the different needs of the stakeholders in the research. Furthermore, the data triangulation validation helped evaluate the understanding of the use of deep learning models in remanufacturing alongside generalisability. Overall, the industry feedback highlights the practitioner's support that the developed models have the potential to enhance remanufacturing productivity and efficiency.

CHAPTER SEVEN: CONCLUSION

7.0 Introduction

Research has identified remanufacturing as a highly complex process that lacks effective techniques, tools and innovation to maximise value recovery due to the current manual processing. The manual state of remanufacturing processing causes efficiency losses due to reduced product quality, increased lead time and cost due to human errors, thereby requiring remanufacturers to explore techniques of improving performance to remain competitive compared to traditional manufacturing.

Consequently, the deep learning models already described as computational algorithms used to learn hierarchical patterns in data were explored to evaluate opportunities for improving remanufacturing using these models. It has achieved state-of-the-art performance across different applications and has continued to gain attention across new domains. The remanufacturing application of deep learning models has produced significant results, suggesting that these models can find even greater applications in remanufacturing with further investigation. The benefits of exploring deep learning-based models for remanufacturing applications are enormous, with different remanufacturing stages having various potential use cases.

The advantages of modelling remanufacturing processes using deep convolutional neural networks include producing a complete and consistent performance alongside the ability to modify these models to suit other applications with ease. The investigated use cases in remanufacturing identification, sorting, inspection, and process control have proved to be successful in achieving automated sorting, inspection and process control in remanufacturing [103], [143], [377], thereby enhancing the overall efficiency of the remanufacturing process. These automated methods and algorithms guarantee improved value recovery, reduced product cost, and reduced lead time to remanufacture end-of-life products, providing an effective alternative to purchasing products with up to a 30% discount.

7.1 Achieving Research Objectives

The research objectives explored various methods of improving the overall remanufacturing process's competitive advantage by analysing and developing deep convolutional neural network-based models for automating sorting, inspection, and process control applications. These objectives were achieved through the following activities:

- Identified the critical remanufacturing operational challenges that require automation.
- Evaluated the type of process data that could be collected.
- Evaluated if the processes could be modelled using the collected research data.
- Developed deep convolutional neural network architectures for identifying and predicting patterns in the various remanufacturing data.
- Analysed the model results for proper inference and deductions.

- Validated the findings of the research using data triangulation and industry feedback.
- Articulated and published the research findings for stakeholders, including remanufacturing practitioners and academics, to enhance understanding.
- Supported future research in deep learning modelling for remanufacturing by providing the torque converter datasets for further investigation.

Nonetheless, the crucial questions that the research addressed include understanding the current level of deep learning practice in remanufacturing, where the current applications of deep learning algorithms in remanufacturing were discussed in the literature review. Besides, the understanding of the deep learning modelling results was also enhanced through the findings from the published reviews on activation functions and optimisation techniques for deep learning. In addition, the developed model was investigated for the inspection of torque converter components for remanufacturing and adapted successfully to sorting and process control remanufacturing applications with little modifications using the different datasets for the specific application. Finally, the study supports future remanufacturing deep learning research by providing the torque converter dataset for further investigations of deep learning in remanufacturing.

7.2 Contributions to Knowledge and Research Originality

The research contributions to the body of knowledge are evident from the publications and conference contributions to enhancing the understanding of deep learning models and their applications to remanufacturing. These contributions include three (3) journal publications in Springer Journal of Remanufacturing, Elsevier's Cleaner Engineering and Technology Journal and the Advances in Science, Technology and Engineering Systems Journal, alongside another six (6) conference contributions, which have attracted over one thousand one hundred (1100) citations and counting. The primary deliverables of the research include

- A review paper on activation functions used in deep learning research supports understanding the role of activation functions in deep learning architectures.
- A journal review paper on the optimisation techniques used in deep learning research to support the selection and understanding of optimisers in deep learning architectures.
- A deep learning architecture for modelling various remanufacturing processes and used for modelling inspection, sorting and process control in remanufacturing.
- A dataset to support further deep learning research in remanufacturing.
- A robust framework for achieving automated inspection for remanufacturing.

The originality of the research outlines the facts from the literature that indicates that outlines the following contributions:

1. First, to compare various activation functions used in deep learning for further understanding.

2. Secondly, to provide a framework for automated inspection in remanufacturing named design for automated inspection (DfAI).
3. Thirdly, to explore the modelling inspection, sorting and process control for the torque converter components remanufacturing application using deep learning method.

7.3 Recommendations and Future Research

Adopting deep learning algorithms in remanufacturing is promising and requires further research to explore other use cases. However, implementing deep learning models for automating various remanufacturing processes can be enhanced by including at least three fixed cameras to record different viewpoints of the objects on arrival. This approach would improve the model's accuracy for various components and sizes. However, this research used one camera with data augmentation to achieve the deep learning-based system for sorting, process control and inspection applications due to the cost of obtaining multiple pieces of equipment for the project.

Nevertheless, the research also identified other potential aspects of remanufacturing that require further investigation by using deep learning models for modelling remanufacturing processes. These include

- Extending deep learning models to evaluate sub-surface product defects requires recording data about the product's internal properties using magnetic, ultrasonic, or eddy-current sensor systems. This application will complement the surface defects investigated in the research to achieve a holistic, automated inspection.
- Consider other sub-processes in remanufacturing, including disassembly, testing, and reassembly, to leverage the success of the deep learning models to improve the overall remanufacturing processes.
- Use historical numeric and text data from the MoL product usage information in decision-making. The stage will be enhanced with sensor-connected devices connected through IoT to log this product use data, improving the remanufacturing decision-making. However, as the MoL data are not readily available, this extension requires extensive planning.

7.4 Limitations of the Research

Certain limitations encountered during this research come from different research stages. These limitations include model development, use cases, generalisability, and integration of the developed model after the investigation. An outline of these limitations is presented

- The main limitation of the experimental research relates to the developed CNN architecture since the architecture has been optimised for classifying objects classes up to twenty objects. However, adapting the developed model to large applications with hundreds of classes would not be optimal. The performance will degrade since the number of kernels used to learn patterns in the data would be so small. Therefore, subjecting the model to a more significant number of inputs will require some modification to perform optimally.

- The use cases and generalisability were also vital since the general CNN models are data-intensive, making the models' performance depend hugely on obtaining quality data about the process under investigation. Since data is expensive to gather to obtain significant portions for providing the train, test, and validation samples, thereby constituting a research limitation.
- The number of practitioners consulted during the research constitutes another limitation, especially in accessing other forms of data for the investigation, thereby improving the use cases alongside obtaining a broader range of practitioner feedback. This limitation affected the depth and spread of data collected, which also impacted the generalisability and the number of practitioner inputs received as industry feedback since the industry feedback involved the members of staff of the host company.
- The system integration challenges include the cost of additional actuation hardware, accessories, and set-up for the respective processes. The expenses add to the cost of achieving product remanufacturing; however, the additional cost could be offset by the wages of hiring experts for the jobs, thereby achieving automation while saving costs in the future.
- Finally, other operational challenges to the integration include fixing the camera at a point of sight away from the conveyor system and achieving multi-views while capturing the data. The challenge of viewpoint is minimised by low-level data augmentation in the model, which helps to reduce the chances of poor generalisation, as discussed in Section 4.11.4. In addition, augmentation could be avoided if a robot is integrated into the design to pick and rotate the product 360 degrees while the camera captures the different viewpoints of the object. Other challenges include the high computational cost of training the deep learning models requires graphics processing units (GPUs). These memory units are expensive but reduce the time required to train the model on the high-dimensional image data used in the research. However, training cost is a single cost to cater for before deployment. It can be remedied by training the models in the cloud using Amazon Web Services (AWS), Google Colaboratory or other cloud computing sources for a small fraction of the cost of buying a GPU.

References

[1] X. Zhang, M. Zhang, H. Zhang, Z. Jiang, C. Liu, and W. Cai, "A review on energy, environment and economic assessment in remanufacturing based on life cycle assessment method," *Jor.of Clean. Prod.*, vol. 255, p. 120160, 2020.

[2] G. D. Hatcher, W. L. Ijomah, and J. F. C. Windmill, "Design for remanufacture: A literature review and future research needs," *Journ.of Clean. Prod.*, vol. 19, no. 17–18, pp. 2004–2014, 2011.

[3] T. E. Goltsos, A. A. Syntetos, and E. van der Laan, "Forecasting for remanufacturing: The effects of serialization," *J. Oper. Manag.*, vol. 65, no. 5, pp. 447–467, 2019.

[4] W. L. Ijomah, "Addressing decision making for remanufacturing operations and design-for-remanufacture," *Int. J. Sustain. Eng.*, vol. 2, no. 2, pp. 91–102, Jun. 2009.

[5] H.-B. Jun, J.-H. Shin, Y.-S. Kim, D. Kiritsis, and P. Xirouchakis, "A framework for RFID applications in product lifecycle management," *Int'l Jor. Comp. Integr. Manuf.*, vol. 22, no. 7, pp. 595–615, 2009.

[6] T. Tolio *et al.*, "Design, management and control of demanufacturing and remanufacturing systems," *CIRP Ann.*, vol. 66, no. 2, pp. 585–609, 2017.

[7] J. Wang, S. Prakash, Y. Joshi, and F. Liou, "Laser Aided Part Repair-A Review," *13th Annu. Solid Free. Fabr. Symp.*, pp. 57–64, 2002.

[8] M. Matsumoto, S. Yang, K. Martinsen, and Y. Kainuma, "Trends and research challenges in remanufacturing," *Int'l Jor. Prec. Eng. Manuf. Green Tech.*, vol. 3, no. 1, pp. 129–142, 2016.

[9] S. J. Ridley, W. L. Ijomah, and J. R. Corney, "Improving the efficiency of remanufacture through enhanced pre-processing inspection – a comprehensive study of over 2000 engines at Caterpillar remanufacturing, U.K.," *Prod. Plan. Control*, vol. 30, no. 4, pp. 259–270, 2019.

[10] J. . Gemage, Ijomah W.L, and J. Windmill, "What makes cleaning a costly operation in remanufacturing," in *11th Global Conf. on Sust. Manuf.*, 2015, pp. 219–223.

[11] C.-M. Lee, W.-S. Woo, and Y.-H. Roh, "Remanufacturing: Trends and Issues," *Int'l Jor. Prec. Eng. Manuf. Green Tech.*, vol. 4, no. 1, p. 113, 2017.

[12] P. Oyekola, A. Mohamed, and J. Pumwa, "Robotic model for unmanned crack and corrosion inspection," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 1, pp. 862–867, 2019.

[13] J. Liu, Z. Zhou, D. T. D. T. Pham, W. Xu, C. Ji, and Q. Liu, "Robotic disassembly sequence planning using enhanced discrete bees algorithm in remanufacturing," *Int'l Jor.of Prod. Res.*, vol. 56, no. 9, pp. 3134–3151, 2018.

[14] K. Li, Q. Liu, W. Xu, J. Liu, Z. Zhou, and H. Feng, "Sequence planning considering human fatigue for human-robot collaboration in disassembly," in *Procedia CIRP*, 2019, vol. 83, pp. 95–104.

[15] D. Peraković, M. Periša, and P. Zorić, "Challenges and issues of ICT in industry 4.0," in *Lecture Notes in Mech. Eng.*, Pleiades Publishing, 2020, pp. 259–269.

[16] D. A. P. P. Paterson, W. L. Ijomah, and J. F. C. C. Windmill, "End-of-life decision tool with emphasis on remanufacturing," *J. Clean. Prod.*, vol. 148, pp. 653–664, 2017.

[17] W. L. Ijomah, C. A. McMahon, G. P. Hammond, and S. T. Newman, "Development of design for remanufacturing guidelines to support sustainable manufacturing," *Robot. Comput. Integr. Manuf.*, vol. 23, no. 6, pp. 712–719, 2007.

[18] W. L. Ijomah, C. A. Mcmahon, G. P. Hammond, and S. T. Newman, "Development of robust design-for- remanufacturing guidelines to further the aims of sustainable development Development of robust design-for-remanufacturing guidelines to further the aims of sustainable development," *Int. J. Prod. Res.*, vol. 45, pp. 4513–4536, 2007.

[19] M. Thierry, M. Salomon, J. Van Nunen, and L. Van Wassenhove, "Strategic Issues in Product Recovery Management," *Calif. Manage. Rev.*, vol. 37, no. 2, pp. 114–136, 1995.

[20] W. L. Ijomah, S. J. Childe, G. P. Hammond, and C. A. McMahon, "A Robust Description and Tool for Remanufacturing: A Resource and Energy Recovery Strategy," in *4th Int' Symp. on Enviro. Consc. Design and Inv. Manufa.*, 2005, pp. 472–479.

[21] W. L. Ijomah, S. Childe, and C. McMahon, "Remanufacturing: A key strategy for sustainable development," *3rd Int'l Conf.on Des. Manuf. Sust. Dev.*, vol. 22, pp. 99–102, 2004.

[22] A. Aprilia, W. L. K. Nguyen, A. Khairyanto, W. C. Pang, S. B. Tor, and G. Seet, "Towards automated

remanufacturing process with additive manufacturing," in *Int'l Conf. on Progress in Additive Manuf.*, 2018, pp. 696–701.

[23] N. K. N. K. Dev, R. Shankar, and F. H. Qaiser, "Industry 4.0 and circular economy: Operational excellence for sustainable reverse supply chain performance," *Resour. Conserv. Recycl.*, vol. 153, p. 104583, 2020.

[24] O. Okorie, F. Charnley, A. Ehiagwina, D. Tiwari, and K. Salonitis, "Towards a simulation-based understanding of smart remanufacturing operations: a comparative analysis," *J. Remanufacturing*, 2020.

[25] P. Goodall, R. Sharpe, and A. West, "A data-driven simulation to support remanufacturing operations," *Comput. Ind.*, vol. 105, pp. 48–60, 2019.

[26] B. Surajit and A. Telukdarie, "Business Logistics Optimization Using Industry 4.0: Current Status and Opportunities," *IEEE Int'l Conf. Indus.Engi. Engi. Mgt.*, pp. 1558–1562, 2019.

[27] H. Tokucoglu, X. Chen, A. E. L. Rhalibi, and T. Opoz, "Sensor based cost modelling for a knowledge support system development," *IEEE Int'l Conf. Auto. Comp.*, pp. 1–6, 2019.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] François Chollet, *Deep Learning with Python*, 2nd ed. Manning Publications, 2021.

[30] J. Jiao, M. Zhao, J. Lin, and K. Liang, "A comprehensive review on convolutional neural network in machine fault diagnosis," *Neurocomputing*, vol. 417, pp. 36–63, 2020.

[31] S. Zahoor, W. Abdul-Kader, and M. Zain, "The prospect of smart-remanufacturing in automotive SMEs: A case study," in *Int'l Conf. on Indus.Eng. and Operations Mgt*, 2019, pp. 735–736.

[32] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *IEEE Int'l Conf. on Compu. Vision*, 2015, pp. 1026–1034.

[34] M. Kerin and D. T. Pham, "A review of emerging industry 4.0 technologies in remanufacturing," *Journal of Cleaner Production*, vol. 237. Elsevier, 2019.

[35] R. Westbrook, "Action research : a new paradigm for research in," *Int. J. Oper. Prod. Manag.*, vol. 15, no. 12, pp. 6–20, 1995.

[36] H. Allaoui, Y. Guo, and J. Sarkis, "Decision support for collaboration planning in sustainable supply chains," *J. Clean. Prod.*, vol. 229, pp. 761–774, 2019.

[37] M. C. Olisah, C. Nwankpa, I. Whitfield, and W. Ion, "The exploration of collaborative supply chain factors in the oil and gas industry," in *34th British Academy of Management*, 2020.

[38] R. L. Ackoff., *Scientific Method: Optimizing Applied Research Decisions.* New York: University of Chicago Press, 1962.

[39] C. Wohlin, "Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study?," *Inf. Softw. Technol.*, vol. 133, p. 106514, 2021.

[40] K. A. Robinson, I. J. Saldanha, and N. A. Mckoy, "Framework for identifying research gaps," *Natl. Collab. Cent. Methods Tools*, no. 2, pp. 1–3, 2012.

[41] M. Schlüter, C. Niebuhr, J. Lehr, and J. Krüger, "Vision-based Identification Service for Remanufacturing Sorting," *Procedia Manuf.*, vol. 21, pp. 384–391, 2018.

[42] L. Wang, X. Xia, J. Cao, and X. Liu, "Modeling and predicting remanufacturing time of equipment using deep belief networks," *Cluster Comput.*, vol. 22, no. s2, pp. 2677–2688, Dec. 2019.

[43] G. Marczyk, D. DeMatteo, and D. Festinger, *Essentials of Research Design and Methodology*. John Wiley and Sons Inc, 2005.

[44] R. K. Yin, *Case Study Research: Design and methods*, 5th ed. USA: SAGE Publications Inc., 2014.

[45] J. W. Creswell, *Research Design, Qualitative Quantitative and Mixed Methods*, 3rd ed. California: SAGE, 2009.

[46] A. Tashakkori, C. Teddlie, and C. B. Teddlie, *Mixed methodology: Combining qualitative and quantitative approaches*, vol. 46. Sage, 2018.

[47] J. Schoonenboom and R. B. Johnson, "How to Construct a Mixed Methods Research Design," *Kolner*

Z. Soz. Sozpsychol., vol. 69, pp. 107–131, 2017.

[48] Bubaker S., “Qualitative and Quantitative Case Study Research - Method on Social Science: Accounting Perspective,” *Int. J. Econ. Manag. Eng.*, vol. 10, no. 12, pp. 3849–3854, 2016.

[49] L. Given, “The SAGE Encyclopedia of Qualitative Research Methods.” SAGE, California, p. 490, 2008.

[50] J. Wisdom and J. W. Creswell, “Mixed methods: integrating quantitative and qualitative data collection and analysis while studying patient-centered medical home models,” *Rockv. Agency Healthc. Res. Qual.*, 2013.

[51] W. L. Neuman, *Social Research Methods: Qualitative and Quantitative Approaches*, 7th Ed. Pearson Education, 2014.

[52] K. M. Eisenhardt, “Building Theories from Case Study Research,” *Acad. Manag. Rev.*, vol. 14, no. 4, pp. 532–550, 1989.

[53] C. A. Romano, “Research Strategies for Small Business: A Case Study Approach,” *Int’l Small Bus. J. Res. Entrep.*, vol. 7, no. 4, pp. 35–43, 1989.

[54] E. D. de Leeuw, J. J. Hox, and D. A. Dillman, *International handbook of survey methodology*. London: Taylor and Francis, 2008.

[55] S. Modell, “Triangulation between case study and survey methods in management accounting research: An assessment of validity implications,” *Mgt. Account. Res.*, vol. 16, no. 2, pp. 231–254, 2005.

[56] G. K. Groff and T. B. Clark, “Commentary on ‘Productions/Operations Management: Agenda for the “80s,””” *Decis. Sci.*, vol. 12, no. 4, pp. 578–581, 1981.

[57] E. S. Buffa and L. Angeles, “Commentary on ‘Productions/Operations Management: Agenda for the “80s,””” *Decis. Sci.*, vol. 12, no. 4, pp. 1–2, 1981.

[58] B. B. Flynn, S. Sakakibara, R. G. Schroeder, K. A. Bates, and E. J. Flynn, “Empirical research methods in operations management,” *J. Oper. Manag.*, vol. 9, no. 2, pp. 250–284, 1990.

[59] R. . Khanna, *Productions and Operations Management*, 2nd ed. Delhi: PHI Learning, 2015.

[60] N. Nasr and M. Thurston, “Remanufacturing: A Key Enabler to Sustainable Product Systems,” in *CIRP International Conference on Lifecycle Engineering*, 2006, pp. 15–18.

[61] R. Steinhilper and Rolf Steinhilper, “Remanufacturing: The Ultimate Form of Recycling,” *J. Ind. Ecol.*, pp. 189–192, 1998.

[62] M. Errington and S. Childe, “A business process model of inspection in remanufacturing,” *Nat. Resour. Res.*, vol. 8, no. 3, pp. 219–232, 1999.

[63] A. M. Yazid, J. K. Rijal, M. S. Awaluddin, and E. Sari, “Pattern Recognition on Remanufacturing Automotive Component as Support Decision Making Using Mahalanobis-Taguchi System,” *Procedia CIRP*, vol. 26, pp. 258–263, 2015.

[64] S. Butzer and S. Schötz, “Map of Remanufacturing Processes Landscape,” 2016.

[65] G. Lancaster, *Research Methods in Management*, 1st ed. Elsevier, 2005.

[66] C.-H. Kuo, K. D. Dunn, and S. U. Randhawa, “A case study assessment of performance measurement in distribution centers,” *Ind. Mgt Data Syst.*, 1999.

[67] A. D. A. D. Joshi and S. M. S. M. Gupta, “Evaluation of design alternatives of End-Of-Life products using internet of things,” *Int’l Jor. Prod. Eco.*, vol. 208, pp. 281–293, 2019.

[68] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2017.

[69] T. Gebru *et al.*, “Datasheets for Datasets,” *CoRR abs/1803.09010*, Mar. 2018.

[70] R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[71] Imagenet, “Large Scale Visual Recognition Challenge (ILSVRC),” *Stanford Vision Lab*, 2018. .

[72] H. Wang and S. Bengio, “The MNIST Database of Handwritten Upper-case Letters,” 2002. .

[73] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[74] Mark Everingham, Luc van Gool, Chris Williams, John Winn, and Andrew Zisserman, “The PASCAL

Visual Object Classes Homepage,” *PASCAL-VOC*, 2018. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/. [Accessed: 07-Aug-2018].

[75] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, Springer, Cham, 2014, pp. 740–755.

[76] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes Challenge: A Retrospective,” *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[77] Tsung-Yi Lin *et al.*, “COCO - Common Objects in Context,” *COCO Dataset*, 2018. .

[78] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, “Learning Deep Features for Scene Recognition using Places Database,” in *Advances in NIPS*, 2014, pp. 487–495.

[79] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, “Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop,” *CoRR abs/1506.03365*, 2015.

[80] “IBM offers free 1m face dataset to combat bias,” *Biometric Technol. Today*, vol. 2019, no. 2, p. 1, Feb. 2019.

[81] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR 2011*, 2011, pp. 1521–1528.

[82] M. Errington and S. J. Childe, “A business process model of inspection in remanufacturing,” *J. Remanufacturing*, vol. 3, no. 1, 2013.

[83] C. Gray and M. Charter, “Remanufacturing and Product Design Designing for the 7th Generation,” 2007.

[84] M. A. Seitz, “A critical assessment of motives for product recovery: the case of engine remanufacturing,” *Jor. Clean. Prod.*, vol. 15, pp. 1147–1157, 2007.

[85] P. Goodall, E. Rosamond, and J. Harding, “A review of the state of the art in tools and techniques used to evaluate remanufacturing feasibility,” *J. Clean. Prod.*, vol. 81, pp. 1–15, Oct. 2014.

[86] D. Parker *et al.*, “Remanufacturing Market Study,” 2015.

[87] R. Steinhilper, “Recent trends and benefits of remanufacturing: From closed loop businesses to synergetic networks,” in *2nd Int’l Symp. on Environmentally Conscious Design and Inv. Manuf.*, 2001, pp. 481–488.

[88] Robert T. Lund, *Remanufacturing : The experience of the United States and implications for developing countries*. World Bank, 1984.

[89] V. D. R. Guide, “Production planning and control for remanufacturing: industry practice and research needs,” *Jor.of Oper. Mgt.*, vol. 18, no. 4, pp. 467–483, 2000.

[90] W. Ijomah, “A model-based definition of the generic remanufacturing business process,” University of Plymouth, 2002.

[91] M. Kosacka, “Sustainability in Remanufacturing Operations,” in *Sustainability in Remanufacturing Operations*, K. F. Golinska-Dawson P., Ed. Springer, 2018, pp. 25–45.

[92] J. Kurilova-Palisaitiene, E. Sundin, and B. Poksinska, “Remanufacturing challenges and possible lean improvements,” *Jor. Clean. Prod.*, vol. 172, pp. 3225–3236, 2018.

[93] A. Gungor and S. M. Gupta, “Disassembly sequence planning for products with defective parts in product recovery,” *Comput. Ind. Eng.*, vol. 35, no. 1–2, pp. 161–164, Oct. 1998.

[94] C. Zikopoulos, S. Panagiotidou, and G. Nenes, “The value of sampling inspection in a single-period remanufacturing system with stochastic returns yield,” *IFIP Adv. Inf. Commun. Technol.*, vol. 338 AICT, pp. 128–135, 2010.

[95] J. Pfrommer *et al.*, “An ontology for remanufacturing systems,” *Automatisierungstechnik*, vol. 70, no. 6, pp. 534–541, 2022.

[96] V. Gopinath and K. Johansen, “Understanding situational and mode awareness for safe human-robot collaboration: Case studies on assembly applications,” *Prod. Eng.*, vol. 13, no. 1, pp. 1–9, 2019.

[97] M. Hochwallner, E. Sundin, and K. Johansen, “Automation in Remanufacturing: Applying Sealant on a Car Component,” in *SPS2022*, IOS Press, 2022, pp. 147–158.

[98] N. C. Y. C. Y. Yeo, H. Pepin, and S. S. S. Yang, “Revolutionizing Technology Adoption for the Remanufacturing Industry,” *Procedia CIRP*, vol. 61, pp. 17–21, 2017.

[99] R. Giutini and K. Gaudette, "Remanufacturing: The next great opportunity for boosting US productivity," *Bus. Horiz.*, vol. 46, no. 6, pp. 41–48, 2003.

[100] J. D. Chiodo and W. L. Ijomah, "Use of active disassembly technology to improve remanufacturing productivity: automotive application," *Int. J. Comput. Integr. Manuf.*, vol. 27, no. 4, pp. 361–371, 2014.

[101] S. Bag, L. C. Wood, S. K. Mangla, and S. Luthra, "Procurement 4.0 and its implications on business process performance in a circular economy," *Resour. Conserv. Recycl.*, vol. 152, p. 104502, 2020.

[102] S. J. Ridley and W. Ijomah, "A novel pre-processing inspection methodology to enhance productivity in automotive product remanufacture: an industry-based research of 2196 engines," *Journal of Remanufacturing*, vol. 5, no. 1. SpringerOpen, 2015.

[103] C. Nwankpa, S. Eze, W. Ijomah, A. Gachagan, and S. Marshall, "Achieving remanufacturing inspection using deep learning," *J. Remanufacturing*, 2020.

[104] H. Singh and P. K. Jain, "Remanufacturing with ECH--a concept," *Procedia Eng.*, vol. 69, pp. 1100–1104, 2014.

[105] W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, and D. T. D. T. Pham, "Disassembly sequence planning using discrete Bees algorithm for human-robot collaboration in remanufacturing," *Robot. Comput. Integr. Manuf.*, vol. 62, 2020.

[106] P. Lundmark, E. Sundin, and M. Björkman, "Industrial challenges within the remanufacturing system," in *3rd Swedish Prod. Symp.*, 2009, pp. 132–138.

[107] S. Wei, O. Tang, and E. Sundin, "Core (product) Acquisition Management for remanufacturing: a review," *J. Remanufacturing*, vol. 5, no. 1, 2015.

[108] J. Östlin, E. Sundin, and M. Björkman, "Importance of closed-loop supply chain relationships for product remanufacturing," *Int. J. Prod. Econ.*, vol. 115, no. 2, pp. 336–348, 2008.

[109] E. Sundin and O. Dunbäck, "Reverse logistics challenges in remanufacturing of automotive mechatronic devices," *J. Remanufacturing*, vol. 3, no. 1, pp. 1–8, 2013.

[110] J. Kurilova-Palisaitiene and E. Sundin, "Challenges and opportunities of lean remanufacturing," *Int'l Jor.of Auto. Tech.*, vol. 8, no. 5, pp. 644–652, 2014.

[111] S. Butzer, D. Kemp, R. Steinhilper, and S. Schötz, "Identification of approaches for remanufacturing 4.0," in *IEEE Europ.Techn. and Eng. Mgt Summit*, 2017, pp. 1–6.

[112] M. Sharp, R. Ak, and T. Hedberg, "A survey of the advancing use and development of machine learning in smart manufacturing," *J. Manuf. Syst.*, vol. 48, pp. 170–179, 2018.

[113] M. Mishra, J. Nayakb, B. Naikc, and A. Abraham, "Deep learning in electrical utility industry: A comprehensive review of a decade of research." 2020.

[114] H. El Hachimi, M. Oubrich, and O. Souissi, "The optimization of Reverse Logistics activities: A Literature Review and Future Directions," in *IEEE Int'l Conf. on Techn. Mgt. Operations and Decisions*, 2018, pp. 18–24.

[115] T. Efendigil, S. Önüt, and C. Kahraman, "A decision support system for demand forecasting with artificial neural networks and neuro-fuzzy models: A comparative analysis," *Expert Syst. Appl.*, vol. 36, no. 3 PART 2, pp. 6697–6707, Apr. 2009.

[116] D. T. Kumar, H. Soleimani, and G. Kannan, "Forecasting return products in an integrated forward/reverse supply chain utilizing an ANFIS," *Int'l Jor. Appl. Maths. Comp.Sci.*, vol. 24, no. 3, pp. 669–682, 2014.

[117] J. Hanafi, S. Kara, and H. Kaebernick, "Generating Fuzzy Coloured Petri Net forecasting model to predict the return of products," in *IEEE Int' Symp. on Electronics and the Environment*, 2007, pp. 245–250.

[118] G. T. Temur, M. Balcilar, and B. Bolat, "A fuzzy expert system design for forecasting return quantity in reverse logistics network," *J. Enterp. Inf. Manag.*, vol. 27, no. 3, pp. 316–328, 2014.

[119] C. Song, X. Guan, Q. Zhao, and Q.-S. Q. S. Jia, "Remanufacturing planning based on constrained ordinal optimization," *Front. Electr. Electron. Eng. China*, vol. 6, no. 3, pp. 443–452, 2011.

[120] P. Shah, A. Gosavi, and R. Nagi, "A machine learning approach to optimise the usage of recycled material in a remanufacturing environment," *Int'l Jor. Prod. Res.*, vol. 48, no. 4, pp. 933–955, 2010.

[121] T. Stock and G. Seliger, “Opportunities of Sustainable Manufacturing in Industry 4.0,” *Procedia CIRP*, vol. 40, no. Icc, pp. 536–541, 2016.

[122] D. Wei, J. Wang, K. Ni, and G. Tang, “Research and application of a novel hybrid model based on a deep neural network combined with fuzzy time series for energy forecasting,” *Energies*, vol. 12, no. 18, p. 3588, 2019.

[123] X. Zhang, X. Ao, Z. Jiang, H. Zhang, and W. Cai, “A remanufacturing cost prediction model of used parts considering failure characteristics,” *Robot. Comput. Integr. Manuf.*, vol. 59, no. July 2018, pp. 291–296, 2019.

[124] W. Abdul-Kader and M. S. Haque, “Sustainable tyre remanufacturing: An agent-based simulation modelling approach,” *Int. J. Sustain. Eng.*, vol. 4, no. 4, pp. 330–347, 2011.

[125] J. Lehr, M. Schlüter, and J. Krüger, “Decentralised identification of used exchange parts with a mobile application,” *Int. J. Sustain. Manuf.*, vol. 4, no. 2–4, pp. 150–164, 2020.

[126] C. Song, X. Guan, Q. Zhao, and Y.-C. Ho, “Machine Learning Approach for Determining Feasible Plans of a Remanufacturing System,” *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 3, pp. 262–275, 2005.

[127] M. Sabbaghi, B. Esmaeilian, A. Raihanian Mashhadi, S. Behdad, and W. Cade, “An investigation of used electronics return flows: A data-driven approach to capture and predict consumers storage and utilization behavior,” *Waste Manag.*, vol. 36, no. 2015, pp. 305–315, 2015.

[128] T. Van Nguyen, L. Zhou, A. Y. L. Chong, B. Li, and X. Pu, “Predicting customer demand for remanufactured products: A data-mining approach,” *Eur. Jor. Oper. Res.*, vol. 281, no. 3, pp. 543–558, 2020.

[129] J. Yang, Z. Jiang, S. Zhu, and H. Zhang, “Data-driven technological life prediction of mechanical and electrical products based on Multidimensional Deep Neural Network: Functional perspective,” *Jor. Manuf. Syst.*, vol. 64, pp. 53–67, 2022.

[130] J. Dekhtiar, A. Durupt, M. Bricogne, D. Kiritsis, H. Rowson, and B. Eynard, “Toward an Extensive Data Integration to Address Reverse Engineering Issues,” in *IFIP Advances in Info. and Comm. Tech.*, 2016, pp. 478–487.

[131] R. Zhang, S. K. Ong, and A. Y. C. Nee, “A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling,” *Appl. Soft Comput.*, vol. 37, pp. 521–532, 2015.

[132] A. Priyono, W. Ijomah, and U. Bititci, “Disassembly for remanufacturing: A systematic literature review, new model development and future research needs,” *J. Ind. Eng. Manag.*, vol. 9, no. 4, p. 899, Nov. 2016.

[133] L. Liu, Q. Zhang, and Y. Sha, “Research on Remanufacturing Scheduling optimization with Uncertain Process,” in *5th Int’l Conf. on Autom., Cont. and Rob. Eng.*, 2020, pp. 439–443.

[134] L. V. Tran, B. H. Huynh, and H. Akhtar, “Ant colony optimization algorithm for Maintenance, Repair and Overhaul scheduling optimization in the context of Industrie 4.0,” *Appl. Sci.*, vol. 9, no. 22, p. 4815, 2019.

[135] S. A. Reveliotis, “Uncertainty management in optimal disassembly planning through learning-based strategies,” *IIE Trans. (Institute Ind. Eng.*, vol. 39, no. 6, pp. 645–658, 2007.

[136] T. Gu, “Automation disassembly sequence generation based on visual recognition and rules in remanufacturing,” in *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, 2021, pp. 428–433.

[137] T. Gibbons, G. Pierce, K. Worden, and I. Antoniadou, “A Gaussian mixture model for automated corrosion detection in remanufacturing.,” in *16th Int’l Conf. on Manuf. Research*, 2018.

[138] J. Li, M. Sage, X. Guan, M. Brochu, Y. Zhao, and Y. F. Fiona, “Machine Learning-Enabled Competitive Grain Growth Behavior Study in Directed Energy Deposition Fabricated Ti6Al4V,” *Jom*, vol. 72, no. 1, pp. 458–464, 2020.

[139] K. Ren, Y. Chew, Y. F. F. Zhang, J. Y. H. Fuh, and G. J. J. Bi, “Thermal field prediction for laser scanning paths in laser aided additive manufacturing by physics-based machine learning,” *Comput. Methods Appl. Mech. Eng.*, vol. 362, p. 112734, 2020.

[140] M. Andoni *et al.*, “Blockchain technology in the energy sector: A systematic review of challenges and opportunities,” *Renew. Sustain. Energy Rev.*, vol. 100, pp. 143–174, Feb. 2019.

[141] C. Yang, W. Xu, J. Liu, B. Yao, and Y. Hu, “Robotic Disassembly Sequence Planning Considering

Robotic Movement State Based on Deep Reinforcement Learning," in *IEEE 25th Int'l Conf. on Comp. Supp. Coop. Work in Design*, 2022, pp. 183–189.

[142] D. A. D. A. Rossit, F. Tohmé, and M. Frutos, "A data-driven scheduling approach to smart manufacturing," *Jor. Ind. Info. Integr.*, vol. 15, pp. 69–79, 2019.

[143] C. Nwankpa, S. Eze, and W. L. Ijomah, "Deep Learning Based Visual Automated Sorting System for Remanufacturing," in *IEEE Green Techn. Conf. (GreenTech)*, 2020, pp. 196–198.

[144] E. Manavalan and K. Jayakrishna, "A review of Internet of Things (IoT) embedded sustainable supply chain for industry4.0 requirements," *Comput. Ind. Eng.*, vol. 127, pp. 925–953, 2019.

[145] D. T. Sturrock and C. D. Pegden, "Introduction to SIMAN.pdf," in *Winter Simulation Conference*, 1990, pp. 109–114.

[146] A. Khan, C. Mineo, G. Dobie, C. Macleod, and G. Pierce, "Vision guided robotic inspection for parts in manufacturing and remanufacturing industry," *J. Remanufacturing*, 2020.

[147] David G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Int'l Conf. on Computer Vision-Volume 2*, 1999, p. 1150.

[148] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[149] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893.

[150] G. A. Ruz, P. A. Estévez, and P. A. Ramírez, "Automated visual inspection system for wood defect classification using computational intelligence techniques," *Int. J. Syst. Sci.*, vol. 40, no. 2, pp. 163–172, 2009.

[151] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A practical approach to microarray data analysis*, Springer, 2003, pp. 91–109.

[152] R. Bro and A. K. Smilde, "Principal component analysis," *Anal. Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[153] M. F. Rabbi, C. Pizzolato, D. G. Lloyd, C. P. Carty, D. Devaprakash, and L. E. Diamond, "Non-negative matrix factorisation is the most appropriate method for extraction of muscle synergies in walking and running," *Sci. Rep.*, vol. 10, no. 1, pp. 1–11, 2020.

[154] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN Transfer Learning for Image Classification," in *Advances in Intelligent Systems and Computing*, 2018, vol. 840, pp. 191–203.

[155] M. Telgarsky, "Benefits of depth in neural networks," in *MLR*, 2016, vol. 49, pp. 1–23.

[156] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[157] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. 25*, 2012.

[158] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.

[159] C. Szegedy *et al.*, "Going deeper with convolutions," in *IEEE Conf.on Comp. Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[160] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, 2017.

[161] J. Schmidhuber and Jürgen Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[162] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," in *ICLR*, 2017.

[163] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[164] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Optimisation in the Brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 19–27, 1958.

[165] A. E. Bryson, W. F. Denham, and S. E. Dreyfus, "Optimal Programming Problems with Inequality

Constraints," *AIAA J.*, vol. 1, no. 11, pp. 2544–2550, 1963.

[166] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[167] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[168] Y. Le Cun *et al.*, "Handwritten Digit Recognition with a Back-Propagation Network," in *NIPS*, 1989, pp. 396–404.

[169] M. Gheisari, G. Wang, and M. Z. A. Bhuiyan, "A Survey on Deep Learning in Big Data," in *IEEE Int'l Conf. on Com. Sci. and Eng. (CSE) and IEEE Int'l Conf. on Embedded and Ubiquitous Comp. (EUC)*, 2017, pp. 173–180.

[170] M. Z. Alom *et al.*, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," *CoRR abs / 1803.01164*, 2018.

[171] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *Trans. Signal Inf. Process.*, vol. 3, p. 2, 2014.

[172] Ilias Maglogiannis, K. Karpouzis, M. Wallace, and J. Soldatos, *Emerging artificial intelligence applications in computer engineering : Real word AI systems with applications in eHealth, HCI, information retrieval and pervasive technologies*. IOS Press, 2007.

[173] S. S. Mousavi, M. Schukat, and E. Howley, "Deep Reinforcement Learning: An Overview," in *SAI Intelligent Systems Conference*, 2016, pp. 426–440.

[174] Y. Bengio and Yoshua, "Learning Deep Architectures for AI," *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[175] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010, pp. 253–256.

[176] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *25th int'l conf. on Machine learning*, 2008, pp. 1096–1103.

[177] J. Weston, F. Ratle, and R. Collobert, "Deep Learning via Semi-Supervised Embedding," in *ICML*, 2008, pp. 1168–1175.

[178] and R. S. Z. Hinton, Geoffrey E., "Autoencoders, Minimum Description Length and Helmholtz free Energy," *Advances in Neural Information Processing Systems*, vol. 3, no. 3., 1994.

[179] G. E. Hinton and J. L. McClelland, "Learning Representations by Recirculation," in *Neural Information Processing Systems (NIPS 1987)*, 1987, pp. 358–366.

[180] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement Learning, Fast and Slow," *Trends in Cognitive Sciences*, vol. 23, no. 5. Elsevier, pp. 408–422, 2019.

[181] E. Yang and D. Gu, "Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey," *Tech. Rep.*, 2004.

[182] S. O. Ali Chishti, S. Riaz, M. Bilal Zaib, and M. Nauman, "Self-Driving Cars Using CNN and Q-Learning," *Proc. 21st Int. Multi Top. Conf. INMIC 2018*, 2018.

[183] R. M. Neal, "Connectionist learning of belief networks," *Artif. Intell.*, vol. 56, no. 1, pp. 71–113, 1992.

[184] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cogn. Sci.*, vol. 9, no. 1, pp. 147–169, 1985.

[185] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *18th Int'l Conf. on Machine Learning*, 2001, p. 643.

[186] S. Zhou, Q. Chen, and X. Wang, "Active deep learning method for semi-supervised sentiment classification," *Neurocomputing*, vol. 120, pp. 536–546, 2013.

[187] G. Hinton, "Where Do Features Come From?," *Cogn. Sci.*, vol. 38, no. 6, pp. 1078–1101, Aug. 2014.

[188] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[189] I. Sutskever, J. Martens, and G. Hinton, "Generating Text with Recurrent Neural Networks," in *28th Int'l Conf. on Machine Learning*, 2011, pp. 1017–1024.

[190] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and

their Compositionality," in *Advances in NIPS*, 2013, pp. 3111–3119.

[191] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[192] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[193] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation:Encoder–Decoder Approaches," in *8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.

[194] J. Zhang, P. Wang, and R. X. Gao, "Modeling of Layer-wise Additive Manufacturing for Part Quality Prediction," *Procedia Manuf.*, vol. 16, pp. 155–162, 2018.

[195] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE Int'l Conf.on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[196] X. Chen, X. Liu, Y. Wang, M. J. F. Gales, and P. C. Woodland, "Efficient Training and Evaluation of Recurrent Neural Network Language Models for Automatic Speech Recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 11, pp. 2146–2157, 2016.

[197] S. T. Hill *et al.*, "A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential.," *Nucleic Acids Res.*, vol. 46, no. 16, pp. 8105–8113, 2018.

[198] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 275, pp. 167–179, 2018.

[199] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[200] S. Sivakumar and S. Sivakumar, "Marginally Stable Triangular Recurrent Neural Network Architecture for Time Series Prediction," *IEEE Trans. Cybern.*, vol. 48, no. 10, pp. 2836–2850, 2018.

[201] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *Proceedings of the IEEE International Conference on Computer Vision*, 2009, pp. 2146–2153.

[202] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833.

[203] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[204] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.

[205] Z. Fei *et al.*, "Deep Convolution Network Based Emotion Analysis towards Mental Health Care," *Neurocomputing*, vol. 388, pp. 212–227, 2020.

[206] Z. Yue *et al.*, "A Novel Semi-Supervised Convolutional Neural Network Method for Synthetic Aperture Radar Image Recognition," *Cognit. Comput.*, pp. 1–12, 2019.

[207] F. Gao, Z. Yue, J. Wang, J. Sun, E. Yang, and H. Zhou, "A Novel Active Semisupervised Convolutional Neural Network Algorithm for SAR Image Recognition," *Comput. Intell. Neurosci.*, vol. 2017, pp. 1–8, 2017.

[208] Y. Le Cun *et al.*, "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 41–46, 1989.

[209] Y. Bengio, "Deep learning of representations: Looking forward," in *Int'l conf. on statistical lang. and speech processing*, 2013, pp. 1–37.

[210] M. Lin, Q. Chen, and S. Yan, "Network In Network," *CoRR abs/1312.4400*, 2013.

[211] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway Networks," *CoRR abs/1505.00387*, 2015.

[212] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conf. on Computer Vision and Pattern Recog.*, 2016, pp. 770–778.

[213] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*, 2016, vol. 9908 LNCS, pp. 646–661.

[214] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[215] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *30th IEEE Conf. on Comp. Vision and Pattern Recog.*, 2017, pp. 1800–1807.

[216] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” *30th IEEE Conf. Comp. Vis. Pattern Recog.*, pp. 5987–5995, 2017.

[217] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, “Rethinking the Inception Architecture for Computer Vision,” in *IEEE Conf on Computer Vision and Pattern Recog.*, 2016, pp. 2818–2826.

[218] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *31st AAAI conference on artificial intelligence*, 2017.

[219] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, L. Van Der Maaten, and K. Q. Weinberger, *Densely Connected Convolutional Networks*. IEEE, pp. 2261–2269.

[220] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, “Dual Path Networks,” in *31st Conference on Neural Information Processing Systems*, 2017, pp. 1–9.

[221] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European conference on computer vision*, 2016, pp. 630–645.

[222] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *IEEE Conf.on Comp. Vision and Pattern Recog.*, 2018, pp. 7132–7141.

[223] A. Vaswani *et al.*, “Attention Is All You Need,” in *31st Conf. on Neural Info. Proc. Systems*, 2017.

[224] A. Ferreira and G. Giraldi, “Convolutional Neural Network approaches to granite tiles classification,” *Expert Syst. Appl.*, vol. 84, pp. 1–11, 2017.

[225] J. Ngiam, Z. Chen, D. Chia, P. Koh, Q. Le, and A. Ng, “Tiled convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 23, 2010.

[226] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[227] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *Int’l Conf. on Learning Representation*, 2016.

[228] J. B. Estrach, A. Szlam, and Y. LeCun, “Signal recovery from Pooling Representations,” in *31st Int’l Conf. on Machine Learning*, 2014, vol. 32, no. 2, pp. 307–315.

[229] D. Yu, H. Wang, P. Chen, and Z. Wei, “Mixed pooling for convolutional neural networks,” in *International conference on rough sets and knowledge technology*, 2014, pp. 364–375.

[230] M. D. Zeiler and R. Fergus, “Stochastic pooling for regularization of deep convolutional neural networks,” *CoRR abs/1301.3557*, 2013.

[231] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 346–361.

[232] O. Rippel, J. Snoek, and R. P. Adams, “Spectral representations for convolutional neural networks,” in *29th Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.

[233] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, “Aggregating local image descriptors into compact codes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, 2011.

[234] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,” in *2nd Int’ Conf. on Comput.Sci. and Tech.*, 2020, pp. 124–133.

[235] N. Ketkar, *Deep Learning with Python - A Hands-on Introduction*. Bangalore: Apress, 2017.

[236] M. Courbariaux, Y. Bengio, and J.-P. David, “BinaryConnect: Training Deep Neural Networks with binary weights during propagations,” *CoRR abs/1511.00363*, 2015.

[237] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural Networks*, vol. 107, pp. 3–11, Nov. 2018.

[238] Y. Qin, X. Wang, and J. Zou, “The Optimized Deep Belief Networks with Improved Logistic Sigmoid Units and Their Application in Fault Diagnosis for Planetary Gearboxes of Wind Turbines,” *IEEE Trans. Ind. Electron.*, vol. 66, no. 5, pp. 3814–3824, 2019.

[239] J. Turian, J. Bergstra, and Y. Bengio, “Quadratic features and deep architectures for chunking,” in *Human Language Technologies: The 2009 Annual Conf. of the North American Chapter of the Association for Comp. Linguistics, Companion Volume: Short Papers*, 2009, pp. 245–248.

[240] W. Ping *et al.*, “Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning,” in *International Conference on Learning Representations - ICLR*, 2018, vol. 79, no. 14, pp. 1094–1099.

[241] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[242] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li, “Improving deep neural networks using soft plus units,” in *Int’l Joint Conf. on Neural Networks*, 2015, pp. 1–4.

[243] Vinod Nair and Geoffrey E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 807–814.

[244] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” in *ICASSP*, 2013.

[245] M. D. Zeiler *et al.*, “On rectified linear units for speech processing,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3517–3521.

[246] D. Mishkin and J. Matas, “All you need is a good init,” *CoRR abs/1511.06422*, Nov. 2016.

[247] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier Nonlinearities Improve Neural Network Acoustic Models,” in *International Conference on Machine Learning (icml)*, 2013.

[248] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *14th Int’l Conf. on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[249] B. Xu, N. Wang, H. Kong, T. Chen, and M. Li, “Empirical Evaluation of Rectified Activations in Convolution Network,” 2015.

[250] D. Macêdo, C. Zanchettin, A. L. I. Oliveira, and T. Ludermir, “Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study,” *Expert Syst. Appl.*, vol. 124, pp. 271–281, 2019.

[251] L. Trottier, P. Giguere, and B. Chaib-draa, “Parametric Exponential Linear Unit for Deep Convolutional Neural Networks,” in *ICLR*, 2017.

[252] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” *CoRR abs / 1511.07289*, 2015.

[253] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-Normalizing Neural Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[254] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in *30th Int’l Conf. on Machine Learning*, 2013, pp. 1319--1327.

[255] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” 2019.

[256] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *33rd AAAI Conf. on Artifi. Intelli.*, 2019, vol. 33, no. 01, pp. 4780–4789.

[257] A. Rosebrock, *Deep Learning for Computer Vision with Python*, 2nd ed. PyImageSearch, 2018.

[258] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, “Are loss functions all the same?,” *Neural Comput.*, vol. 16, no. 5, pp. 1063–1076, 2004.

[259] C. E. Nwankpa, “Advances in Optimisation Algorithms and Techniques for Deep Learning,” *Adv. Sci. Technol. Eng. Syst. J.*, vol. 5, no. 5, pp. 563–577, 2020.

[260] T. Dozat, “Incorporating Nesterov Momentum into Adam,” in *ICLR*, 2016.

[261] Sabastian Ruder, “An Overview of Gradient Descent Optimization Algorithms,” *CoRR abs / 1609.04747*, pp. 1–14, 2017.

[262] S. Sigtia and S. Dixon, “Improved music feature learning with deep neural networks,” in *IEEE int’l conf. on acoustics, speech and signal processing*, 2014, pp. 6959–6963.

[263] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of Machine Learning Research*, 2013, pp. 1139–1147.

[264] Herbert Robbins and Sutton Monro, “A Stochastic Approximation Method,” *Ann. Math. Stat.*, vol. 22,

no. 3, pp. 400–407, 1951.

[265] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[266] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *CoRR abs / 1212.5701,* 2012.

[267] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *CoRR, abs/1412.6980,* 2014.

[268] D. Lee and K. Myung, “Read my lips, login to the virtual world,” in *IEEE Int’l Conf. on Consumer Electronics,* 2017, pp. 434–435.

[269] L. Liu *et al.*, “On the Variance of the Adaptive Learning Rate and Beyond,” in *ICLR,* 2020.

[270] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” in *ICLR,* 2018.

[271] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, “Lookahead optimizer: k steps forward, 1 step back,” in *Neural Information Processing Systems*, 2019.

[272] K. Lan *et al.*, “Multi-view convolutional neural network with leader and long-tail particle swarm optimizer for enhancing heart disease and breast cancer detection,” *INDIA INTL. Congr. Comput. Intell.*, 2018.

[273] X. Chen, F. Kopsaftopoulos, Q. Wu, H. Ren, and F. K. Chang, “A self-adaptive 1D convolutional neural network for flight-state identification,” *Sensors,* vol. 19, no. 2, 2019.

[274] M. Revathi, I. Jasmine Selvakumari Jeya, and S. N. Deepa, “Deep learning-based soft computing model for image classification application,” *Soft Comput.*, vol. 24.

[275] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized Stochastic Gradient Descent,” in *Advances in Neural Information Processing Systems 23 (NIPS),* 2010, pp. 2595–2603.

[276] M. Zinkevich, A. J. Smola, and J. Langford, “Slow learners are fast,” in *NIPS,* 2009, pp. 2331–2339.

[277] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, “A fast parallel SGD for matrix factorization in shared memory systems,” in *7th ACM Conf. on Recommender systems,* 2013, pp. 249–256.

[278] C. Garbin, X. Zhu, and O. Marques, “Dropout vs. batch normalization: an empirical study of their impact to deep learning,” *Multimed. Tools Appl.*, pp. 1–39, 2020.

[279] J. Martens, “Deep learning via hessian-free optimization.,” in *ICML,* 2010, vol. 27, pp. 735–742.

[280] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Int’l Conf. on Artifi. Intelli. and Statistics*, 2010, pp. 249–256.

[281] S. Koturwar and S. Merchant, “Weight Initialization of Deep Neural Networks(DNNs) using Data Statistics,” *CoRR abs/1710.10570,* 2017.

[282] A. M. Saxe, J. L. McClelland, and S. Ganguli, “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks,” *Int’l Conf. Learn. Represent.*, 2013.

[283] D. D. E. Wong, S. A. Fuglsang, J. Hjortkjær, E. Ceolini, M. Slaney, and A. De Cheveigne, “A comparison of regularization methods in forward and backward models for auditory attention decoding,” *Front. Neurosci.*, vol. 12, p. 531, 2018.

[284] G. Larsson, M. Maire, and G. Shakhnarovich, “FractalNet: Ultra-Deep Neural Networks without Residuals,” *CoRR abs/1605.07648,* May 2016.

[285] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of Neural Networks using DropConnect,” in *30th Int’l Conf.on Machine Learning*, 2013, vol. 28, no. 3, pp. 1058–1066.

[286] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” *CoRR abs /1707.07012,* 2017.

[287] L. Xie and A. Yuille, “Genetic CNN,” *IEEE Int’l Conf.on Comput. Vis.*, pp. 1388–1397, 2017.

[288] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *IEEE conf. on computer vision and pattern recog.*, 2015, pp. 648–656.

[289] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning,” in *33rd Int’l Conf. on Machine Learning,* 2016, vol. 48, pp. 1928–1937.

[290] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *CoRR abs/1502.03167,* 2015.

[291] G. Hripcsak and A. S. Rothschild, “Agreement, the F-measure, and reliability in information retrieval,” *J. Am. Med. Informatics Assoc.*, vol. 12, no. 3, pp. 296–298, 2005.

[292] X.-H. Zhou, D. K. McClish, and N. A. Obuchowski, *Statistical methods in diagnostic medicine*, vol. 569. John Wiley \& Sons, 2009.

[293] S. Vongbunyong and W. H. Chen, “Vision System In: Disassembly Automation. Sustainable Production, Life Cycle Engineering and Management.,” Springer, Cham, 2015, pp. 55–93.

[294] J. Zhang, S. Yi, G. U. O. Liang, G. A. O. Hongli, H. Xin, and S. Hongliang, “A new bearing fault diagnosis method based on modified convolutional neural networks,” *Chinese J. Aeronaut.*, vol. 33, no. 2, pp. 439–447, 2020.

[295] P. Lu, B. Song, and L. Xu, “Human face recognition based on convolutional neural network and augmented dataset,” *Syst. Sci. \& Control Eng.*, vol. 9, no. 2, pp. 29–37, 2021.

[296] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning,” *Nat. Biotechnol.*, vol. 33, no. 8, pp. 831–838, 2015.

[297] S. Zhang *et al.*, “A deep learning framework for modelling structural features of RNA-binding protein targets,” *Nucleic Acids Res.*, vol. 44, no. 4, pp. e32–e32, Feb. 2016.

[298] S. T. M. Kin, S. K. Ong, and A. Y. C. Nee, “Remanufacturing Process Planning,” *Procedia CIRP*, vol. 15, pp. 189–194, 2014.

[299] M. A. Ilgin and S. M. Gupta, *Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art*, vol. 91, no. 3. Elsevier Ltd, 2010, pp. 563–591.

[300] R. Subramoniam, D. Huisingh, and R. B. Chinnam, “Remanufacturing for the automotive aftermarket-strategic factors: literature review and future research needs,” *J. Clean. Prod.*, vol. 17, no. 13, pp. 1163–1174, 2009.

[301] A. Robotis, T. Boyaci, and V. Verter, “Investing in reusability of products of uncertain remanufacturing cost: The role of inspection capabilities,” *Int’l J. Prod. Econ.*, vol. 140, no. 1, pp. 385–395, 2012.

[302] P. Kopardekar, A. Mital, and S. Anand, “Manual, Hybrid and Automated Inspection Literature and Current Research,” *Integr. Manuf. Syst.*, vol. 4, no. 1, pp. 18–29, 1993.

[303] H. K. Aksoy and S. M. Gupta, “Buffer allocation plan for a remanufacturing cell,” *Comput. Ind. Eng.*, vol. 48, no. 3, pp. 657–677, 2005.

[304] M. Errington, “Remanufacturing Inspection Models,” 2006.

[305] H. Huang, Z. Qian, and Z. Liu, *Metal Magnetic Memory Technique and Its Applications in Remanufacturing*. 2021.

[306] Y. Zhang, D. Zhou, P. Jiang, and H. Zhang, “The state-of-the-art surveys for application of metal magnetic memory testing in remanufacturing,” *Adv. Mater. Res.*, vol. 301–303, pp. 366–372, 2011.

[307] H. Huang, Z. Qian, and Z. Liu, “Detection of Damage in Remanufactured Coating,” in *Metal Magnetic Memory Technique and Its Applications in Remanufacturing*, Springer, 2021, pp. 169–179.

[308] M. Itagaki, E. Takamiya, K. Watanabe, T. Nukaga, and F. Umemura, “Diagnosis of quality of fresh water for carbon steel corrosion by Mahalanobis distance,” *Corros. Sci.*, vol. 49, no. 8, pp. 3408–3420, 2007.

[309] S. NĂdĂban, S. Dzitac, and I. Dzitac, “Fuzzy TOPSIS: A General View,” *Procedia Comput. Sci.*, vol. 91, pp. 823–831, 2016.

[310] M. Ramachandran., U. Ragavendran, and V. Fegade, “Selection of used piston for remanufacturing using fuzzy TOPSis optimization,” in *Fuzzy systems and data mining IV : Proceedings of FSDM*, 2018, pp. 61–67.

[311] F. Russo, “An image enhancement technique combining sharpening and noise reduction,” *IEEE Trans. Instrum. Meas.*, vol. 51, no. 4, pp. 824–828, Aug. 2002.

[312] Ruigang Yang and M. Pollefeys, “Multi-resolution real-time stereo on commodity graphics hardware,” in *IEEE Comp.Society Conf. on Computer Vision and Pattern Recog.*, pp. I211–I217.

[313] F. A. Saiz, G. Alfaro, and I. Barandiaran, “An Inspection and Classification System for Automotive Component Remanufacturing Industry Based on Ensemble Learning,” *Information*, vol. 12, no. 12, p. 489, 2021.

[314] S. Krig, *Computer Vision Metrics Textbook Edition Survey, Taxonomy and Analysis of Computer*

Vision, Visual Neuroscience, and Deep Learning, 2nd ed. Springer Nature, 2016.

[315] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[316] B. Leibe, A. Leonardis, and B. Schiele, "An Implicit Shape Model for Combined Object Categorization and Segmentation," in *Toward Category-Level Object Recognition. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2006, pp. 508–524.

[317] Kai Jungling and M. Arens, "Feature-based person detection beyond the visible spectrum," in *IEEE Comp. Society Conf.on Computer Vision and Pattern Recog.*, 2009, pp. 30–37.

[318] B. Nath, F. Reynolds, and R. Want, "RFID Technology and Applications," *IEEE Pervasive Comput.*, vol. 5, no. 1, pp. 22–24, 2006.

[319] W.-J. Gao, B. Xing, and T. Marwala, "Teaching - Learning-based optimization approach for enhancing remanufacturability pre-evaluation system's reliability," in *IEEE Symposium on Swarm Intelligence*, 2013, pp. 235–239.

[320] M. L. Dering and C. S. Tucker, "A Computer Vision Approach for Automatically Mining and Classifying End of Life Products and Components," in *20th Design for Manuf. and the Life Cycle Conf.; 9th Int'l Conf. on Micro- and Nanosystems*, 2015.

[321] X. Li, S. Siahpour, J. Lee, Y. Wang, and J. Shi, "Deep learning-based intelligent process monitoring of directed energy deposition in additive manufacturing with thermal images," *Procedia Manuf.*, vol. 48, pp. 643–649, 2020.

[322] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *J. Process Control*, vol. 24, no. 3, pp. 223–233, 2014.

[323] V. Gopakumar, S. Tiwari, and I. Rahman, "A deep learning based data-driven soft sensor for bioprocesses," *Biochem. Eng. J.*, vol. 136, pp. 28–39, 2018.

[324] D. Hu and R. Kovacevic, "Sensing, modeling and control for laser-based additive manufacturing," *Int'l Jor. Mach. Tools Manuf.*, vol. 43, no. 1, pp. 51–60, 2003.

[325] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 657–667, Jan. 2015.

[326] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Comput. Chem. Eng.*, vol. 33, no. 4, pp. 795–814, 2009.

[327] D. Dong and T. J. McAvoy, "Nonlinear principal component analysis—based on principal curves and neural networks," *Comput. \& Chem. Eng.*, vol. 20, no. 1, pp. 65–78, 1996.

[328] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Adv. Neural Inf. Process. Syst.*, vol. 19, 2006.

[329] H. Kaneko and K. Funatsu, "Adaptive soft sensor model using online support vector regression with time variable and discussion of appropriate hyperparameter settings and window size," *Comput. \& Chem. Eng.*, vol. 58, pp. 288–297, 2013.

[330] X. D. Hu, F. Z. Kong, and J. H. Yao, "Development of monitoring and control system for laser remanufacturing," in *Applied Mechanics and Materials*, 2011, vol. 44–47, pp. 81–85.

[331] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

[332] Python - Foundation, "Python Programming," *python.org*, 2022. [Online]. Available: https://www.python.org/. [Accessed: 03-May-2022].

[333] Mathworks, "MATLAB - MathWorks - MATLAB & Simulink," 2022. [Online]. Available: https://uk.mathworks.com/products/matlab.html. [Accessed: 03-May-2022].

[334] R-Foundation, "R: The R Project for Statistical Computing," 2022. [Online]. Available: https://www.r-project.org/. [Accessed: 03-May-2022].

[335] Y. Jia *et al.*, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *CEUR Workshop Proceedings*, 2014, vol. 1436, pp. 675–678.

[336] G. C. Adam Paszke, Sam Gross, Soumith Chintala, "PyTorch - From Research to Production (An open source Deep Learning Platform)," *Pytorch.org*, 2022. [Online]. Available: https://pytorch.org/. [Accessed: 03-May-2022].

[337] François Chollet, “Keras - The Python Deep Learning library,” 2022. [Online]. Available: https://keras.io/. [Accessed: 03-May-2022].

[338] Apache - Software - Foundation, “MXNet: A Scalable Deep Learning Framework,” 2022. [Online]. Available: https://mxnet.apache.org/. [Accessed: 03-May-2022].

[339] Google-Brain, “TensorFlow,” *Google Brain Team*, 2022. [Online]. Available: https://www.tensorflow.org/. [Accessed: 03-May-2022].

[340] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A Matlab-like Environment for Machine Learning,” in *BigLearn, NIPS Workshop*, 2011.

[341] M. Abadi *et al.*, “TensorFlow: A System for Large-Scale Machine Learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016.

[342] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, “Data Preprocessing for Supervised Learning,” *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 1, no. 12, pp. 111–117, 2007.

[343] Z. Li, L. Jin, C. Yang, and Z. Zhong, “Hyperparameter search for deep convolutional neural network using effect factors,” in *IEEE China Sum. and Int’l Conf.on Sig. and Info. Proc.*, 2015, pp. 782–786.

[344] R. Ren, T. Hung, and K. C. Tan, “A Generic Deep-Learning-Based Approach for Automated Surface Inspection,” *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 929–940, 2018.

[345] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2015.

[346] O. Silvén, M. Niskanen, and H. Kauppinen, “Wood inspection with non-supervised clustering,” *Mach. Vis. Appl.*, vol. 13, no. 5, pp. 275–285, 2003.

[347] J. Krüger, J. Lehr, M. Schlüter, and N. Bischoff, “Deep learning for part identification based on inherent features,” *CIRP Ann.*, vol. 68, no. 1, pp. 9--12, 2019.

[348] J. Donahue *et al.*, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,” in *Int’l Conf. on machine learning*, 2014, pp. 647--655.

[349] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” in *IEEE Conf. on Comp. Vision and Pattern Recog.*, 2014, pp. 3606–3613.

[350] Tensorflow, “Tensorflow Documentation,” 2022.

[351] S. Hu, X. Zhang, H. Liao, X. Liang, M. Zheng, and S. Behdad, “Deep learning and machine learning techniques to classify electrical and electronic equipment,” in *Int’l Design Eng. Techn. Conf. and Comp. and Info. in Eng. Conf.*, 2021, vol. 85413.

[352] A. Ng, “Convolutional neural networks,” 2020.

[353] B. N. Popov, *Corrosion Engineering: Principles and Solved Problems*. Elsevier Inc., 2015.

[354] P. Broberg, “Surface crack detection in welds using thermography,” *NDT E Int’l*, vol. 57, pp. 69–73, 2013.

[355] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int’l Jor. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[356] TensorFlow, “Save and load Keras models | TensorFlow Core,” Jan-2022. [Online]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize. [Accessed: 22-Mar-2022].

[357] K. M. M. Tant, A. J. Mulholland, A. Curtis, and W. L. Ijomah, “Design-for-testing for improved remanufacturability,” *J. Remanufacturing*, vol. 9, no. 1, pp. 61–72, 2019.

[358] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, “Machine learning for internet of things data analysis: a survey,” *Digit. Commun. Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[359] G. Olague and R. Mohr, “Optimal camera placement for accurate reconstruction,” *Pattern Recognit.*, vol. 35, no. 4, pp. 927–944, 2002.

[360] L. Taylor and G. Nitschke, “Improving Deep Learning using Generic Data Augmentation,” 2017.

[361] C. E. Nwankpa, W. Ijomah, and A. Gachagan, “Design for automated inspection in remanufacturing: A discrete event simulation for process improvement,” *Clean. Engi. Techn.*, vol. 4, p. 100199, 2021.

[362] M. U. R. Siddiqi *et al.*, *Low cost three-dimensional virtual model construction for remanufacturing industry*, vol. 9, no. 2. Springer, 2019, pp. 129–139.

[363] P. Sharma, Y. P. S. Berwal, and W. Ghai, “Performance analysis of deep learning CNN models for disease detection in plants using image segmentation,” *Information Processing in Agriculture*, vol. 7, no. 4. pp. 566–574, 2020.

[364] G. Keppel, *Design and Analysis: A Researcher’s Handbook*. Prentice-Hall Inc, 1991.

[365] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” *CoRR abs / 1702.01923*, 2017.

[366] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization,” in *Advances in Neural Information Processing Systems 24*, 2011, pp. 2546–2554.

[367] I. Kandel and M. Castelli, “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,” *ICT Express*, vol. 6, no. 4, pp. 312–315, 2020.

[368] J. Cohen, “A coefficient of agreement for nominal scales,” *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[369] K. J. Berry and P. W. Mielke, “A Generalization of Cohen’s Kappa Agreement Measure to Interval Measurement and Multiple Raters,” *Educ. Psychol. Meas.*, vol. 48, no. 4, pp. 921–933, 1988.

[370] H. J. Flack, V.F., Afifif, A. A., Lachenbruch, P.A., Schouten, “Sample size determinations for two rater kappa,” *Psychometrika*, vol. 53, no. 3, pp. 321–325, 1988.

[371] Y. Xu and W. Feng, “Develop a cost model to evaluate the economic benefit of remanufacturing based on specific technique,” *J. Remanufacturing*, vol. 4, no. 1, 2014.

[372] M. Q. Patton, “Enhancing the quality and credibility of qualitative analysis.,” *Health Serv. Res.*, vol. 34, no. 5 Pt II, pp. 1189–1208, 1999.

[373] K. W. Thomas and W. G. Tymon, “Necessary Properties of Relevant Research: Lessons from Recent Criticisms of the Organizational Sciences.,” *Acad. Manag. Rev.*, vol. 7, no. 3, pp. 345–352, 1982.

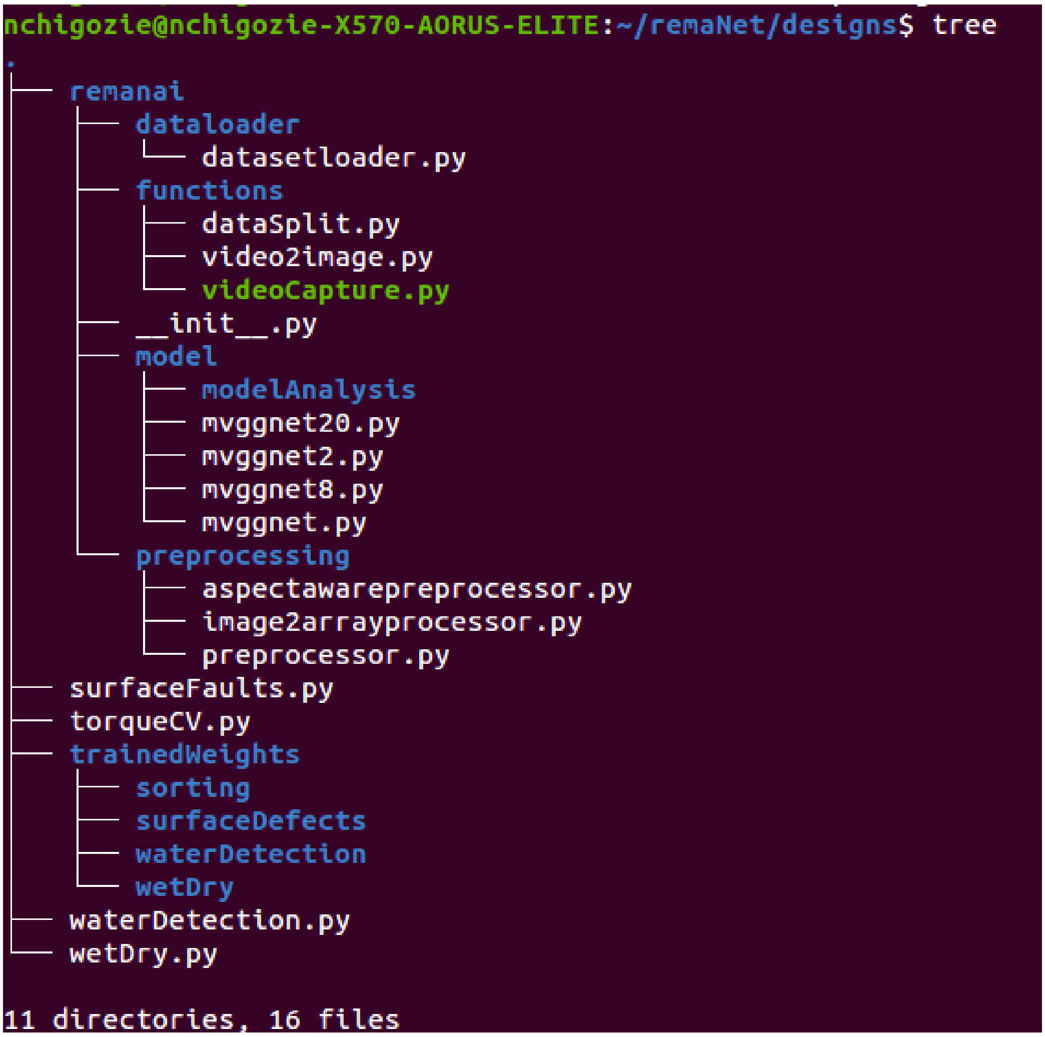
[374] W. L. Oberkampf and T. G. Trucano, “Verification and validation benchmarks,” *Nucl. Eng. Des.*, vol. 238, no. 3, pp. 716–743, 2008.

[375] M. Landry, J. L. Malouin, and M. Oral, “Model validation in operations research,” *Eur. J. Oper. Res.*, vol. 14, no. 3, pp. 207–220, 1983.

[376] S. Jamieson, “Likert scales: How to (ab)use them,” *Med. Educ.*, vol. 38, no. 12, pp. 1217–1218, 2004.

[377] C. Nwankpa, W. Ijomah, and A. Gachagan, “Artificial Intelligence for Process Control In Remanufacturing,” in *12th International Symposium on Environmentally Conscious Design and Inverse Manufacturing (Ecodesign)*, 2021.

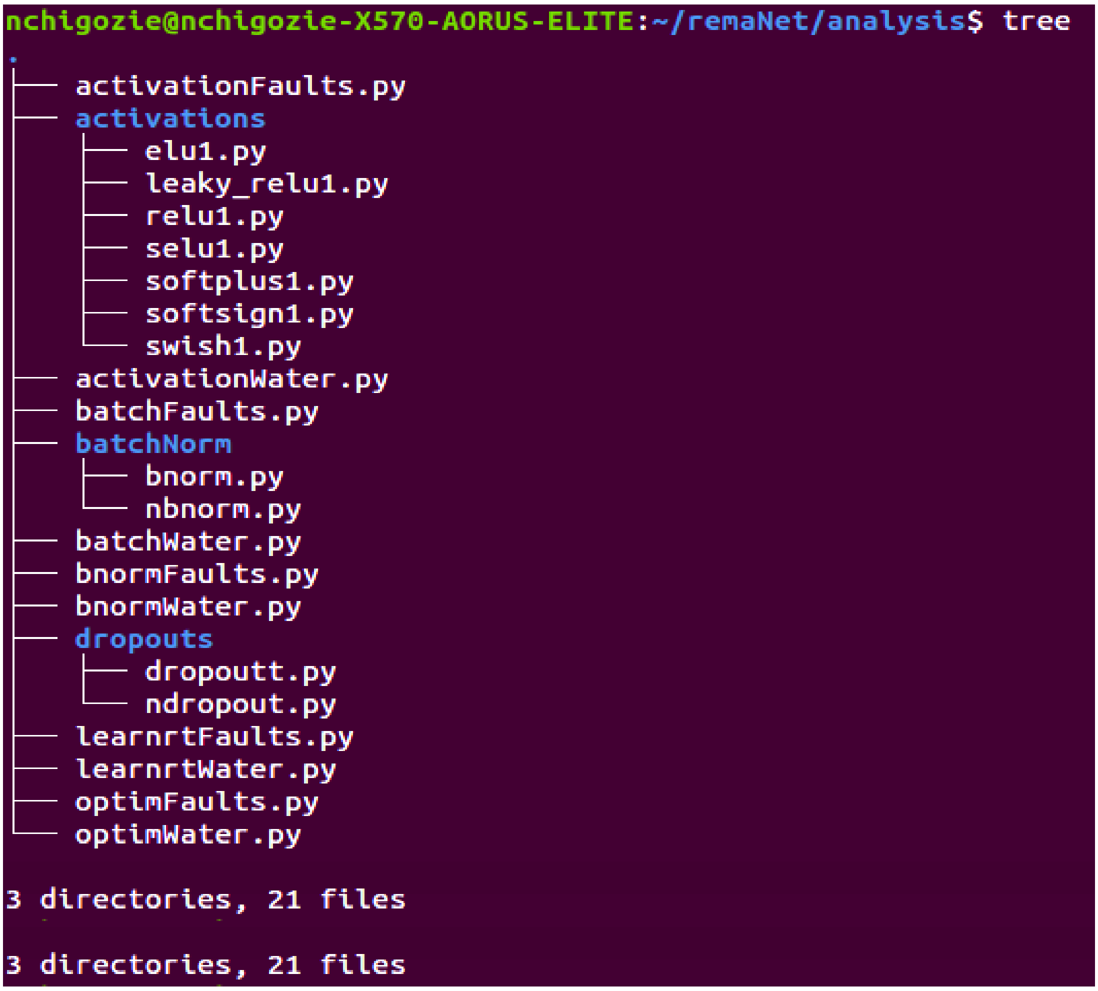
Appendix 1A Research code design tree



```
nchigozie@nchigozie-X570-AORUS-ELITE:~/remaNet/designs$ tree
.
├── remanai
│   ├── dataloader
│   │   └── datasetloader.py
│   ├── functions
│   │   ├── dataSplit.py
│   │   ├── video2image.py
│   │   └── videoCapture.py
│   ├── __init__.py
│   ├── model
│   │   ├── modelAnalysis
│   │   ├── mvggnet20.py
│   │   ├── mvggnet2.py
│   │   ├── mvggnet8.py
│   │   └── mvggnet.py
│   └── preprocessing
│       ├── aspectawarepreprocessor.py
│       ├── image2arrayprocessor.py
│       └── preprocessor.py
├── surfaceFaults.py
├── torqueCV.py
├── trainedWeights
│   ├── sorting
│   ├── surfaceDefects
│   ├── waterDetection
│   └── wetDry
├── waterDetection.py
└── wetDry.py

11 directories, 16 files
```

Appendix 1B Model analysis code tree

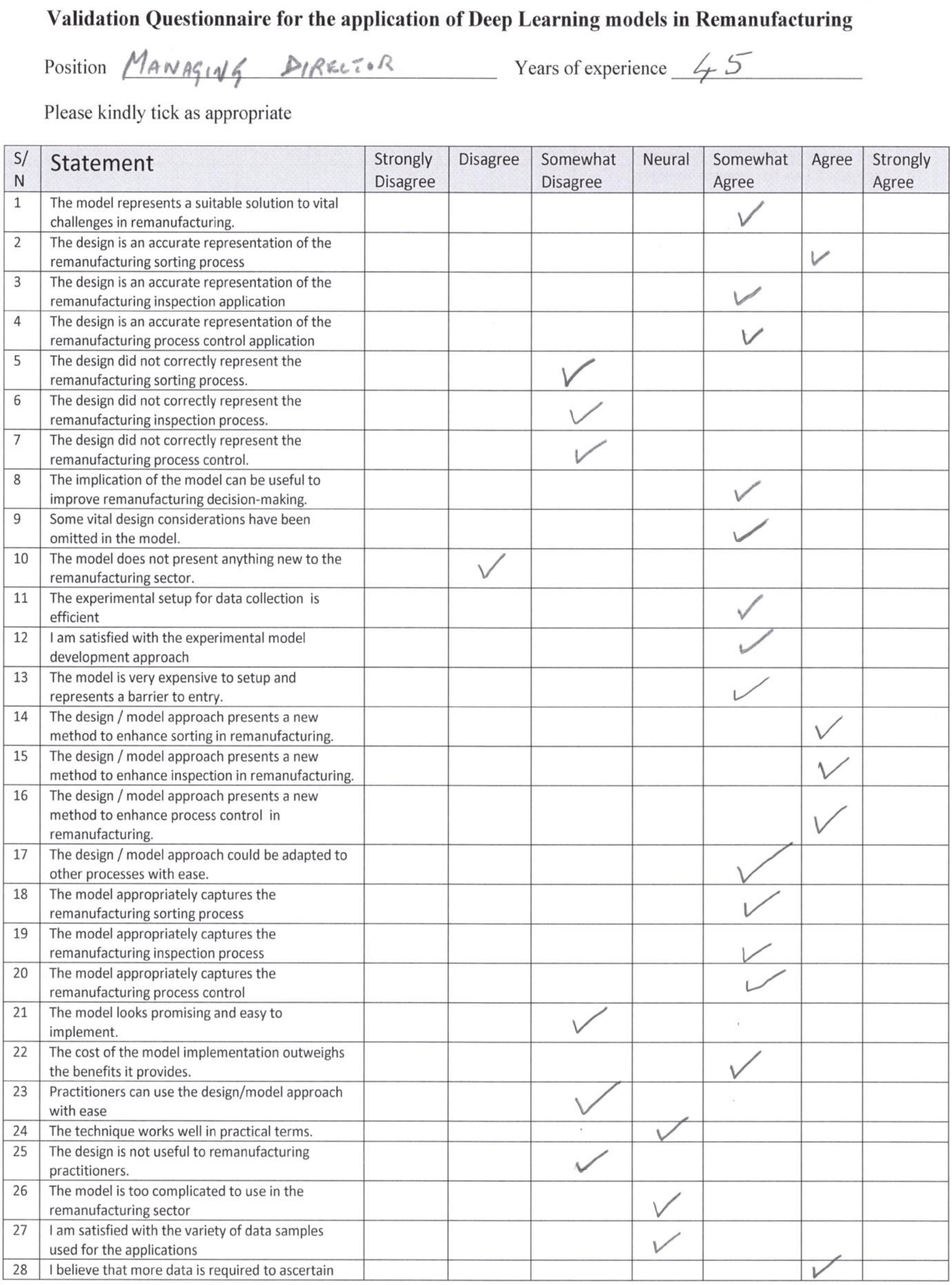


```
nchigozie@nchigozie-X570-AORUS-ELITE:~/remaNet/analysis$ tree
.
├── activationFaults.py
├── activations
│   ├── elu1.py
│   ├── leaky_relu1.py
│   ├── relu1.py
│   ├── selu1.py
│   ├── softplus1.py
│   ├── softsign1.py
│   └── swish1.py
├── activationWater.py
├── batchFaults.py
├── batchNorm
│   ├── bnorm.py
│   └── nbnorm.py
├── batchWater.py
├── bnormFaults.py
├── bnormWater.py
├── dropouts
│   ├── dropoutt.py
│   └── ndropout.py
├── learnrtFaults.py
├── learnrtWater.py
├── optimFaults.py
└── optimWater.py

3 directories, 21 files

3 directories, 21 files
```

Appendix 2 Validation Results



Validation Questionnaire for the application of Deep Learning models in Remanufacturing

Position MANAGING DIRECTOR Years of experience 45

Please kindly tick as appropriate

S/N	Statement	Strongly Disagree	Disagree	Somewhat Disagree	Neural	Somewhat Agree	Agree	Strongly Agree
1	The model represents a suitable solution to vital challenges in remanufacturing.					✓		
2	The design is an accurate representation of the remanufacturing sorting process						✓	
3	The design is an accurate representation of the remanufacturing inspection application					✓		
4	The design is an accurate representation of the remanufacturing process control application					✓		
5	The design did not correctly represent the remanufacturing sorting process.			✓				
6	The design did not correctly represent the remanufacturing inspection process.			✓				
7	The design did not correctly represent the remanufacturing process control.			✓				
8	The implication of the model can be useful to improve remanufacturing decision-making.					✓		
9	Some vital design considerations have been omitted in the model.					✓		
10	The model does not present anything new to the remanufacturing sector.		✓					
11	The experimental setup for data collection is efficient					✓		
12	I am satisfied with the experimental model development approach					✓		
13	The model is very expensive to setup and represents a barrier to entry.					✓		
14	The design / model approach presents a new method to enhance sorting in remanufacturing.						✓	
15	The design / model approach presents a new method to enhance inspection in remanufacturing.						✓	
16	The design / model approach presents a new method to enhance process control in remanufacturing.						✓	
17	The design / model approach could be adapted to other processes with ease.					✓		
18	The model appropriately captures the remanufacturing sorting process					✓		
19	The model appropriately captures the remanufacturing inspection process					✓		
20	The model appropriately captures the remanufacturing process control					✓		
21	The model looks promising and easy to implement.			✓				
22	The cost of the model implementation outweighs the benefits it provides.					✓		
23	Practitioners can use the design/model approach with ease			✓				
24	The technique works well in practical terms.				✓			
25	The design is not useful to remanufacturing practitioners.			✓				
26	The model is too complicated to use in the remanufacturing sector				✓			
27	I am satisfied with the variety of data samples used for the applications				✓			
28	I believe that more data is required to ascertain						✓	

	the practical implementation						✓	
29	I feel that the data sufficiently models the use cases.					✓		
30	The data is enough to outline the use cases.				✓			
31	The model is useful to the remanufacturing sector in the present time.						✓	
32	The deep learning approach to sorting, inspection and process control in remanufacturing is not applicable but can be useful in the future.				✓			
33	The model is based on recent technological advancement and it represents a solution to vital remanufacturing challenges					✓		
34	The proposed solution approach addresses the current practitioner needs.				✓			
35	The design will help to improve the overall remanufacturing efficiency				✓			
Please provide any additional comments here								

Thank you for your time.

Appendix 3A Developed inspection application model codes

```
#Import dependencies
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.models import Sequential
from tensorflow import keras
from keras import backend
import tensorflow as tf
initialiser = tf.keras.initializers.HeUniform(seed=1)
activationType = "swish"
class Model:
    @staticmethod
    def build(width, height, depth, classes):
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1    # if "channels first",
        if backend.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1
            model.add(Conv2D(64, (3, 3), padding="same", kernel_initializer=initialiser,
           bias_initializer='zeros', input_shape=inputShape))
        model.add(Activation(activationType))
        model.add(Conv2D(32, (3, 3), padding="same", kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Conv2D(32, (3, 3), padding="same", kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(Conv2D(16, (3, 3), padding="same", kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.5))
        model.add(Flatten())
        model.add(Dense(512))
        model.add(Activation(activationType))
        model.add(Dropout(0.5))
        model.add(Dense(8))
        model.add(Activation("softmax"))
        return model
```

Appendix 3B Developed sorting application model codes

```
#Import dependencies
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.models import Sequential
from tensorflow import keras
from keras import backend
import tensorflow as tf
initialiser = tf.keras.initializers.HeUniform(seed=1)
activationType = "swish"
class Model:
    @staticmethod
    def build(width, height, depth, classes):
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1  # if "channels first"
        if backend.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1
        model.add(Conv2D(64, (3, 3), padding="same", kernel_initializer=initialiser,
              bias_initializer='zeros', input_shape=inputShape))
        model.add(Activation(activationType))
        model.add(Conv2D(48, (3, 3), padding="same",kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Conv2D(36, (3, 3), padding="same", kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(Conv2D(20, (3, 3), padding="same", kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Flatten())
        model.add(Dense(1024))
        model.add(Activation(activationType))
        model.add(Dropout(0.5))
        model.add(Dense(20))
        model.add(Activation("softmax"))
        print(model.summary())
        return model
```

Appendix 3C Developed process control model codes

```
#Import dependencies
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.models import Sequential
from tensorflow import keras
from keras import backend
import tensorflow as tf
initialiser = tf.keras.initializers.HeUniform(seed=1)
activationType = "swish"
class Model:
    @staticmethod
    def build(width, height, depth, classes):
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1      # if "channels first"
        if backend.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1
        model.add(Conv2D(16, (3, 3), padding="same",
kernel_initializer=initialiser,bias_initializer='zeros',input_shape=inputShape))
        model.add(Activation(activationType))
        model.add(Conv2D(16, (3, 3), padding="same",kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Conv2D(8, (3, 3), padding="same",kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(Conv2D(8, (3, 3), padding="same",kernel_initializer=initialiser,bias_initializer='zeros'))
        model.add(Activation(activationType))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.5))
        model.add(Flatten())
        model.add(Dense(512))
        model.add(Activation(activationType))
        model.add(Dropout(0.5))
        model.add(Dense(1))
        model.add(Activation("sigmoid"))
        print(model.summary())
        return model
```