

Automatic Annotation of Subsea Pipelines using Deep Learning

Anastasios Stamoulakatos

Centre for Intelligent Dynamic Communications
Electronic & Electrical Engineering Department
University of Strathclyde, Glasgow, UK

June 30, 2023

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Abstract

Subsea pipeline inspection is a crucial process for the oil and gas industry to ensure asset quality, as damage can lead to interruptions in production and pose an environmental threat. To this day, this process is based on human annotators that inspect an immense amount of visual and sensor data and manually annotate the events that occur during subsea pipeline surveys. This is a labour-intensive process, prone to human error, very costly for the oil and gas industry, and potentially unsafe for the annotators as it happens off-shore. This thesis proposes methodologies to automate visual inspection of subsea pipelines using Deep Learning (DL) which, in turn, can enable more robust, accurate, and faster inspections, allowing personnel to work on other more sophisticated tasks while reducing cost. To this end, the objectives of this thesis are: (i) developing a framework for subsea survey multi-label image classification and threshold search using Precision Recall (PR) curves, (ii) extending to subsea survey video classification and comparison between three Convolutional Neural Network (CNN)-based models, (iii) proposing a subsea survey texture adaptation methodology that combines the Swapping Autoencoder (SAE) architecture with a classifier module along with a characterisation of the domain shift between two subsea surveys recorded at different times and places.

For the first objective, a deep CNN ResNet-50 is used to automatically detect five subsea survey events; Anode, Exposure, Burial, Field Joint and Free Span, using only the centre video feed of a Remotely Operated Vehicle (ROV). To reduce the demands on the training time, a transfer learning approach is adopted where the feature extraction layers of the network are initialised using the weights of a network pre-trained on ImageNet. The network is then modified to allow for multi-label classification, allowing for the identification of events that appear concurrently in subsea surveys and re-trained

on subsea survey images. Different ResNet depths have been compared and ResNet-50 is selected as it provides the best balance between performance and number of parameters. An additional experiment is conducted to demonstrate the generalisation of the validation to the test sets when the data is split based on different events of a survey.

To extend this study to automatic video annotation, an evaluation of three models for classifying subsea survey video data is presented. A subsea survey video has been curated, and several regularisation techniques are investigated to address its challenges. The models include a traditional 2D CNN, IBN-ResNet50, which classifies individual frames and averages the predictions, along with a 3D IBN-ResNet50 and a 2D IBN-ResNet50-LSTM, which create a single prediction per video clip. Instance Batch Normalisation (IBN) is used between the convolutional layers of the models to improve performance with varying lighting conditions and changes in colour contrast in the surveys. Experimental results indicate that the 2D model outperforms the spatio-temporal models, particularly for short events. The experiment also suggests that a larger dataset would have been beneficial for the 3D model, but it would also require additional manual annotation.

For the third objective, three methods are tested to measure the adaptation of models from a source to a target survey. The first method compares variations of a ResNet-50 model with different normalisation layers. The second method proposes a two-step process combining an image-to-image translation solution (SAE) with a classifier module. The third method involves creating two synthesised datasets using SAE, and use them to increase the variability of the training source data. All methods present better or equal performance on the target surveys compared to the baseline ResNet-50 but they do not achieve supervised learning levels. To this end, further experimentation shows that adding 20% of target events to the source dataset is enough to boost performance in the target test set and reach the same levels as using all of the events. Finally, one method is proposed that measures the in- and out-of-domain shift between two surveys by examining the Fréchet Inception Distance (FID) scores between class-specific subsets.

Acknowledgements

First of all, I would like to thank the University of Strathclyde and my first supervisor, Dr. Christos Tachtatzis, for giving me the opportunity to study for the Ph.D. degree. Doing a Ph.D. can be challenging at times, and I could not have completed it without the supervision of Christos. During the last four years, Christos has been a great supervisor and mentor and has helped me develop both my research and my writing skills. We have had many meetings that involved technical discussions as well as editing the papers and this thesis. Our communication has had its ups and downs, especially during the Covid times, but overall I can definitely say that he has helped me grow as a person.

I would also like to thank the senior members of the CIDCOM group, and specifically Andy, Chris, and Javier, who helped me with coding and research-related questions, but also helped me settle in Scotland. Coming from abroad, the first few months of staying in a new country can be challenging, and Andy made sure that I had all the help I needed. In addition, I would like to thank my fellow Ph.D. students Miko, Tom, and Stathis for the technical discussions and their friendship. All of them had been listening to me complain when my research was not going well and gave me the moral support I needed.

Furthermore, I would like to thank my line manager at CMAC, Dr. Cameron Brown, who has allowed me to spend time editing this thesis while working on my new role. Working on two things at the same time can be challenging, but I have had the freedom to manage both. Another person who has helped me manage my time and prioritise my responsibilities is Bhavik, my thesis mentor.

Finally, I would like to thank my psychotherapist, Ms. Aggeliki Salion for support-

ing me during this period, my sister Elena who was always there when I needed to talk, and all my friends around Glasgow with whom I spent weekends going to gigs and enjoying hiking around Scotland. The support group is very important during a Ph.D., and I had the opportunity to meet some really cool people both at the university and outside of it, with whom I will be connected forever.

Contents

List of Figures	x
List of Tables	xvi
Acronyms	xix
1 Introduction	2
1.1 Subsea Pipeline Inspection and Challenges	2
1.2 Benefits of Automation	4
1.3 Research Questions	6
1.4 Research Contributions	7
1.4.1 Journals	8
1.4.2 Conferences	8
1.5 Thesis Outline	8
2 Literature Review	10
2.1 Introduction	10
2.2 Subsea Inspection using Multibeam Echosounder	11
2.3 Traditional Image Processing	12
2.4 Underwater Image Datasets	16
2.5 Challenges of Subsea Visual Footage	20
2.6 Deep Learning for Subsea Applications	21
2.6.1 Marine Structure Inspection	21
2.6.2 Posidonia Oceanica	22

Contents

2.6.3	Coral Reefs	22
2.6.4	Underwater Object Detection	23
2.6.5	Underwater Image Enhancement	23
2.7	Deep Learning for Subsea Pipeline Inspection	23
2.8	Deep Learning for Power Line and Manufacturing Inspection	28
2.8.1	Power Line Inspection	28
2.8.2	Manufacturing Inspection	29
2.8.3	Defect Inspection Datasets	30
2.9	Conclusions	31
3	Deep Learning Background	33
3.1	Introduction	33
3.2	Background on Neural Networks	33
3.3	Fully Connected Layers	34
3.3.1	Training of Neural Networks	35
3.3.2	Gradient Descent Variants	37
3.3.3	Momentum	38
3.3.4	Adam	38
3.3.5	Learning Rate and Schedulers	39
3.4	Convolutional Neural Networks	41
3.4.1	Convolution Layers	41
3.4.2	Downsampling	43
3.4.3	Activation Functions	44
3.4.4	Weight Initialisation	47
3.4.5	Normalisation	48
3.5	Recurrent Neural Networks	51
3.6	Overfitting and Underfitting	54
3.6.1	Bias-Variance Tradeoff	55
3.7	Regularisation	56
3.7.1	Weight Decay	57
3.7.2	Dropout	58

Contents

3.7.3	Label Smoothing	58
3.7.4	Early Stopping	59
3.7.5	Image Augmentation	59
3.8	Deep Convolutional Neural Networks	62
3.8.1	AlexNet	62
3.8.2	VGG	64
3.8.3	GoogleNet / Inceptionv1	64
3.8.4	ResNet	65
3.9	Transfer Learning	67
3.10	CNN Visualisation	69
3.11	Hyperparameter Tuning	71
3.12	Subsea Survey Multi-Class Image Classification	72
3.12.1	Dataset Description	73
3.12.2	Model Architecture	75
3.12.3	Multi-class Classification	77
3.12.4	Training Details	77
3.12.5	Data Augmentation	78
3.12.6	Model Performance Evaluation using a Hold-out (Test) set . . .	79
3.12.7	Performance Metrics	80
3.12.8	Results	83
3.13	Conclusions	86
4	Subsea Survey Multi-Label Image Classification	87
4.1	Introduction	87
4.2	Multi-Label Classification	88
4.3	Dataset Description	89
4.4	Model Architecture	89
4.5	Confidence Scores and Class Activation Maps	90
4.6	Threshold Tuning	93
4.7	Model Performance Evaluation and Cross Validation	95
4.8	Multi-Label Performance Metrics	97

Contents

4.9	Model Performance on Validation Sets	98
4.10	Model Performance on Test Set	101
4.11	Additional Experiments	104
4.11.1	Universal Thresholding	104
4.11.2	Multi-threshold using ROC curves	105
4.11.3	Randomly Initialised Network Weights	105
4.11.4	Effect of Model Size	107
4.11.5	Training with Instance Batch Normalisation Layer	108
4.12	Splitting Based on Subsea Survey Events	110
4.12.1	Generalisation Assessment - Dataset Split with and without Event Traceability)	113
4.13	Conclusions	115
5	Subsea Survey Video Classification	117
5.1	Introduction	117
5.2	Video Classification	118
5.2.1	3D CNN	120
5.2.2	2D CNN Encoder with RNN Decoder	122
5.3	Model Architecture	123
5.4	Dataset Exploration	124
5.5	Mitigating Label Imbalance	127
5.5.1	Sampling Sequences	130
5.5.2	Video Augmentation	132
5.6	Training Details	134
5.6.1	Focal Loss	136
5.6.2	Label Smoothing	137
5.7	Performance Evaluation Metrics	138
5.8	Results	138
5.8.1	Effect of dataset size in 3D model	141
5.9	Additional Experiments	142
5.9.1	Test Time Augmentation	142

Contents

5.9.2	Focal Loss: hyperparameter gamma	143
5.9.3	Label Smoothing	145
5.9.4	Cost-Sensitive Learning	146
5.10	Conclusions	147
6	Subsea Survey Shift and Adaptation	150
6.1	Introduction	150
6.2	ResNets with Different Intermediate Normalisation Layers	154
6.3	Subsea Survey Domain Adaptation	157
6.4	Two-Step Swapping Autoencoder plus Classifier for Cross Domain Clas- sification	159
6.4.1	Training Details of the First Step	161
6.4.2	Training Details of the Second Step	162
6.4.3	Results of Approaches A and B	164
6.5	Synthetic Dataset using SAE for Cross Domain Classification	167
6.5.1	Results of training with Synthesised Data	169
6.6	Domain Shift Assessment using Fréchet Inception Distance (FID)	170
6.6.1	In-Domain and Out-Of-Domain FID without Label Information .	171
6.6.2	In-Domain and Out-Of-Domain FID with Label Information . .	174
6.7	Adding Target Events to Source Dataset	178
6.8	Conclusions	180
7	Conclusions and Recommendations	182
7.1	Conclusions	182
7.2	Future Work	187
	Bibliography	190

List of Figures

1.1	Subsea Pipeline Inspection using Remotely Operated Vehicles	3
1.2	Survey Console Room and Event of Pipeline Exposure	4
1.3	Evolution of Subsea Pipeline Inspection	5
2.1	Subsea Applications	11
2.2	Inspection Taxonomy	28
3.1	Neural Network as a Function	34
3.2	Comparison of a biological and an artificial neuron.	34
3.3	A Fully Connected Neural Network	35
3.4	Training a Neural Network	36
3.5	Basic CNN architecture	41
3.6	2D Convolution	42
3.7	Convolution without stride and padding, convolution with stride of 2 and no padding, and convolution with padding of 2 and no stride [134] .	43
3.8	Max and Average Pooling	44
3.9	Sigmoid	45
3.10	ReLU	46
3.11	Leaky ReLU	46
3.12	Softmax	47
3.13	Unnormalised data lead to gradients of larger parameters dominating the updates, with normalisation parameters updated proportionally equal.	49
3.14	Normalisation techniques [149]	49

List of Figures

3.15	An unrolled recurrent neural network [153]	51
3.16	LSTM gates [153]	52
3.17	GRU gates [159]	54
3.18	Underfitting, Fitting, and Overfitting	55
3.19	Bias-Variance Tradeoff	56
3.20	L1 Weight Decay regularisation with three different values of λ for fitting a 16 th degree polynomial model to approximate a part of a cosine function	57
3.21	Dropout	58
3.22	Early Stopping [169]	60
3.23	Taxonomy of Image Augmentation Techniques	61
3.24	Examples of Texture Augmentation	61
3.25	Examples of Structure Augmentation	61
3.26	Neural Style Transfer using Johnson [172] Architecture. Here, the bot- tom image consists of \hat{y} , the top left image is both x and y_c , while the bottom right image is the y_s .	62
3.27	Comparison of deep CNNs on ImageNet (top-1 accuracy) [175]	63
3.28	AlexNet [176]	63
3.29	Inception Modules	64
3.30	Residual connection	66
3.31	Papers using ResNet [184]	67
3.32	The 64 3x3 convolutional kernels of the first VGG12 convolutional layer	70
3.33	Input bee image and activations of the 1st, 6th and 12th convolutional layers of VGG12	70
3.34	Class Activation Mapping [190]	71
3.35	Class Activation Map	71
3.36	Methodology Flow Diagram	72
3.37	Examples of events in subsea pipeline surveys with varying scene condi- tions; from left to right: Burial (B), Exposure (E), Anode (AN), Field Joint (FJ), Free Span (FS)	74
3.38	Event distribution of a total 23,570 frames of the complete dataset	74

List of Figures

3.39	ResNet-50 Architecture with modified head.	76
3.40	Figure presents how a series of the aforementioned augmentations can modify the original Exposure (E) sample seen on the left-hand side. For example, on the first augmented sample (top-left corner) a series of flipping, rotation, and brightness increase has been applied.	79
3.41	Binary Confusion Matrix	81
3.42	Multi-class Confusion Matrix [212]	81
3.43	Exposure samples and their predicted scores	84
3.44	Confusion Matrix	84
4.1	Multi-class and multi-label setting using one-hot encoded labels [218] . .	88
4.2	Label distribution of a total 23,570 frames of the complete dataset . . .	89
4.3	ResNet-50 Architecture with modified head and Sigmoid	90
4.4	From top to bottom: Ground Truth Label, Image, Class Activation Map (CAM) Heatmap, Predicted Confidence Scores for the five types of events	91
4.5	Example of ROC and PR Curves	94
4.6	Model training and evaluation process	95
4.7	5-fold Cross Validation	96
4.8	Monte Carlo Cross Validation	97
4.9	Steps for evaluating model's performance: (1) Validation Set (2) Feature Extraction (3) Classifier (4) PR Curves for optimal thresholds selection (5) Applying optimal thresholds (6) Comparison with Ground Truth . .	99
4.10	Precision Recall curves for all labels; the inset shows a zoomed version of the top-right corner	99
4.11	Confusion matrices on the test set for each class; Anode, Burial, Exposure, Field Joint and Free Span.	102
4.12	Aggregated metrics while the threshold value increases from 0 to 1 for all labels.	105
4.13	ROC curves for the five labels	106
4.14	ResNet original block and IBN-ResNet block [238]	109
4.15	Event Distribution	110

List of Figures

4.16	Examples of annotation regimes for ‘short’ events such as Field Joint (FJ) and Anode (AN) (a) single annotation (b) annotation with start and end delineations.	111
4.17	Frame distribution	112
5.1	Considered architectures	118
5.2	3D convolution [257]	121
5.3	Schematic diagram of 3D ResNet-50	122
5.4	3D ResNet block and 3D IBN-ResNet block	124
5.5	IBN-ResNet50 Encoder with Long Short Term Memory (LSTM) Decoder and Classifier	125
5.6	Distribution of events from the 2012 survey	125
5.7	Histograms of the duration of events of interest in seconds; Exposure (E), Burial (B), Free Span (FS), FJ and AN.	126
5.8	Distribution of video clips of the five events of interest	127
5.9	Frame sequences examples of the five events of interest	128
5.10	Histograms of the collected clips of events in frames	129
5.11	Total frames of the five events of interest	129
5.12	Label resampling [270]	130
5.13	Example of sampling frames and sequences from ‘long’ events of E, B, and FS	131
5.14	Example of sampling frames and sequences from ‘short’ events of AN and FJ	132
5.15	Example of a Temporal Elastic Transformation, where the middle part of the video (lowest level of the push-up) has been stretched and fit to 10 frames.	133
5.16	Example of an Elastic Transformation	134
5.17	Examples of Augmentations. The first sequence (a) shows a rotation-augmented Field Joint, while the second (b) is a horizontally flipped Exposure frame sequence	135
5.18	Focal Loss for different γ values	137

List of Figures

5.19	F1-Score for the Original and Larger Dataset	142
5.20	F1-Score of models trained with different Focal Loss gamma values	144
5.21	Anode Recall and Precision of models trained with different Focal Loss gamma values	145
5.22	F1-Score of models trained with different label smoothing	145
5.23	F1-Score for Balanced Sampling and Cost-Sensitive Learning	147
5.24	Anode Recall and Precision of models trained with Balanced Sampling and Cost-Sensitive Learning	148
6.1	Subsea survey data recorded in 2012 (first row) and 2016 (second row)	150
6.2	Events Frame Distribution for the two Subsea Surveys	151
6.3	Augmentations used during training	152
6.4	Samples of 2012 and 2016 surveys	155
6.5	Intermediate Normalisation	156
6.6	Swapping Autoencoder Architecture [284] on images from LSUN churches [294]	160
6.7	First Training Step. For the shake of simplicity, the Discriminator mod- ules are not added to the diagram.	161
6.8	SAE Reconstruction Results	162
6.9	SAE Reconstruction Results	162
6.10	Second Training Step	163
6.11	Steps required when using Image Translation for Domain Adaptation: Step 1 is to train and test a classifier in the source domain to get a baseline performance measurement. On Step 2, an image translation model can be trained to modify either the source or target domain data accordingly, these models only have access to the source labels. On Step 3, there is two options: (i) use the classifier model trained on data from the source domain, but when inferring in the target domain, the imagery is translated to the source domain, or (ii) retrain the classifier model with data from the source domain plus data translated from the target domain to the source; and then use this retrained model to perform conventional inference in the target domain.	168

List of Figures

6.12	Sample images of the Office-Home Dataset [310]. The dataset contains 65 classes from four domains - Art, Clipart, Product, Real World. Here, 5 example classes are shown.	172
6.13	Samples images from the GTA5 [239] and Cityscapes [240] datasets. . .	172
6.14	Office-Home FIDs	175
6.15	FID scores between domains and within domain. The bottom left quadrant marked by the black continuous line contains cross-domain scores. The dashed borders indicate comparisons for the same label. High values along the diagonal in the cross-domain region indicate a significant domain shift.	177
6.16	Free Span Events of 2016 Survey	177
6.17	Burial and Exposure Events of 2016 Survey	177
6.18	Survey Events	178
6.19	Aggregate F1-Score (F1-Total) on the source (blue) and target (orange) test sets by adding target events to the source training set plus FID scores between the hybrid and target datasets.	179

List of Tables

2.1	Pipeline Segmentation Work	13
2.2	Pipeline Contour Detection Work	14
2.3	Underwater Image Datasets	17
2.4	Deep Learning for Subsea Pipeline Inspection	24
2.5	Underwater Image Datasets	30
3.1	Test set metrics	85
4.1	Optimal label-based thresholds for the validation set	100
4.2	Aggregate performance of the five models, one for each fold	100
4.3	Metrics with optimal thresholds on the validation set.	101
4.4	Test set performance of individual labels and aggregate.	103
4.5	Test set performance of individual labels and aggregate using universal thresholding	104
4.6	Test set performance of individual labels and aggregate using ROC thresh- olding	106
4.7	Test set performance of individual labels and aggregate using a randomly initialised ResNet-50	107
4.8	Test the performance of the set of different sizes of ResNet models . . .	108
4.9	Test set performance of individual labels and aggregate using IBN-ResNet- 50	109
4.10	Test Set Performance on the Dataset of the previous Sections	113
4.11	Training, Validation, and Test Set Performance with Frame Splitting . .	114

List of Tables

4.12	Training, Validation, and Test Set Performance with Event Splitting . .	114
4.13	Average Performance per Test Fold.	115
5.1	Temporal Augmentations: 1 of those is selected randomly if not normal downsampling	132
5.2	Spatial Augmentations: Up to 3 of those are selected randomly per clip	134
5.3	2d Label-based Test Set Metrics	139
5.4	3D Label-Based Test Set Metrics	139
5.5	2D-LSTM Label-Based Test Set Metrics	140
5.6	Aggregated performance metrics for 2D, 3D CNN and 2D LSTM	140
5.7	Parameters and computation complexity of 2D and 3D Models	140
5.8	Performance of a 3D model trained on a larger training dataset of 29,347 samples	141
5.9	2d Label-Based with Test Time Augmentation	143
5.10	2D Test Set Metrics using BCE or FL with $\gamma = 0$	143
5.11	2D Test Set Metrics using FL with $\gamma = 0.5$	144
5.12	2D Test Set Metrics using FL with $\gamma = 5$	144
5.13	2d Test Set Metrics for a model trained with cost-sensitive learning . . .	147
6.1	ResNet-50; metrics on source and target surveys	153
6.2	Global Means and Standard Deviations of the 2 Surveys	155
6.3	ResNet-50; metrics on source and target surveys	157
6.4	IBN-ResNet-50; metrics on source and target surveys	157
6.5	GN-ResNet-50; metrics on source and target surveys	157
6.6	Models with their corresponding Accuracy on ImageNet and Parameters	160
6.7	ResNet-50; metrics on source and target surveys	165
6.8	SAE encoder plus classifier; metrics in source and target surveys	165
6.9	Approach A (target texture); metrics in source and target surveys . . .	166
6.10	Approach B (only structure); metrics in source and target surveys . . .	166
6.11	ResNet-50 with and without synthesised data; metrics in source and target surveys	170

List of Tables

6.12	IBN-ResNet-50 with and without synthesised data; metrics in source and target surveys	170
6.13	GN-ResNet-50 with and without synthesised data; metrics in source and target surveys	170
6.14	FID between Domains	173
6.15	FID within Domains	174

List of Tables

Acronyms

Adam Adaptive Moment Estimation. 38, 39

AI Artificial Intelligence. 33

AN Anode. xiv, 73, 89, 111, 112, 125–127, 139, 141, 143, 145, 146, 151, 153

AUV Autonomous Underwater Vehicle. 15, 22, 23, 29

B Burial. xiv, 73, 89, 111, 126, 127, 139, 146, 151, 176

BCE Binary Cross Entropy. 90, 135, 136, 143, 146

BGD Batch Gradient Descent. 37

BN Batch Normalisation. 49, 50, 66, 75, 104, 108, 109, 123, 155, 156, 164

CAD Computer Aided Design. 23

CAM Class Activation Map. xiii, 69, 90, 91

CCE Categorical Cross Entropy. 77

CE Cross Entropy. 36, 77, 136

CLR Cyclic Learning Rate. 40

CNN Convolutional Neural Network. ii, iii, xi, xii, 9, 22–24, 29, 33, 41, 42, 59, 62, 63, 67–69, 77, 86, 89, 117–124, 134–136, 138, 140, 142, 146, 183

CPU Central Processing Unit. 188

Acronyms

CSV Comma Separated Value. 72

CV Cross Validation. 80, 95, 96, 101, 114

DL Deep Learning. ii, 4, 6–8, 10, 11, 21–23, 28–30, 33, 41, 44, 54, 62, 67, 71, 72, 86, 117, 119, 150, 183, 186, 188

E Exposure. xiv, 73, 89, 93, 111, 126, 127, 139, 143, 151, 153, 176

EMR Exact Match Ratio. 98, 103, 104, 109, 114, 140

FCNN Fully Convolutional Neural Network. 22

FID Fréchet Inception Distance. iii, xvi, 7, 9, 65, 154, 167, 171–179, 186

FJ Field Joint. xiv, 73, 89, 102, 111, 112, 125–127, 139, 141, 146, 151

FL Focal Loss. 135, 136, 142, 143, 146, 152

fMRI fractional Magnetic Resonance Imaging. 120

FN False Negative. 80, 82, 83, 93, 98, 102, 123, 138, 139

FNR False Negative Rate. 102, 103

FP False Positive. 80, 82, 83, 93, 94, 98, 102, 123, 138, 139

FPR False Positive Rate. 93, 102, 103, 105

FS Free Span. xiv, 73, 89, 111, 125–127, 139, 141, 143, 151, 153, 176

GAN Generative Adversarial Network. 23, 29, 42, 159, 161, 167, 171, 187

GN Group Normalisation. 51, 154–156

GPU Graphics Processing Unit. 107, 131, 136, 140, 188

GRU Gated Recurrent Unit. 52, 53

IBN Instance Batch Normalisation. iii, 104, 122, 123, 154–156

Acronyms

- ILSVRC** ImageNet Large Scale Visual Recognition Challenge. 62
- IMR** Inspection, Maintenance and Repair. 4
- IMU** Inertial Measurement Unit. 2
- IN** Instance Normalisation. 50, 108, 109, 123, 155
- KP** Kilometer Point. 112
- LN** Layer Normalization. 51
- LReLU** Leaky ReLU. 46
- LSTM** Long Short Term Memory. xiv, 52–54, 118, 122–125, 140
- MBE** Multibeam Echosounder. 2, 3, 8, 10–12, 188
- ML** Machine Learning. 29, 41, 54–56, 130
- MLP** Multilayer Perceptron. 24
- MSE** Mean Square Error. 36, 54, 55
- NAS** Neural Architecture Search. 71, 124
- NMS** Non-Maximum Suppression. 15
- NN** Neural Network. 8, 24, 33–36, 39, 44, 47, 48, 51, 58, 131
- NST** Neural Style Transfer. 60, 167
- OF** Optical Flow. 120
- PCA** Principal Component Analysis. 22
- PR** Precision Recall. ii, xiii, 7, 93, 94, 98–100, 104, 105, 138, 153, 186
- QC** Quality Control. 3, 4, 74, 110, 139, 182

Acronyms

RANSAC Random Sample Consensus. 15

ReLU Rectified Linear Unit. 45, 46, 75

RGB Red Green Blue. 15

RNN Recurrent Neural Network. 51, 52, 120, 124

ROC Receiver Operating Characteristic. 93, 94, 104, 105

ROV Remotely Operated Vehicle. ii, 2, 3, 7, 10, 20–22, 78, 92, 132, 133, 137, 140, 150, 151, 176, 182, 188

S Structure. 162, 164

SAE Swapping Autoencoder. ii, iii, xviii, 7, 153, 159–162, 164–167, 169, 186

SAS Synthetic Aperture Sonars. 12

SGD Stochastic Gradient Descent. 37, 38

SSD Single Shot Detector. 28, 29

T Texture. 162, 164

TLS Total Least Square. 15

TN True Negative. 80, 93, 98, 102

TP True Positive. 80, 82, 83, 98, 102, 138

TPR True Positive Rate. 93, 105

TTA Test Time Augmentation. 142

Acronyms

Chapter 1

Introduction

1.1 Subsea Pipeline Inspection and Challenges

Oil and gas operators are governed by regulations that require frequent visual inspections of subsea pipelines and platforms to assess the condition and risks of these assets. These surveys are carried out on an annual or biannual basis and the integrity of the asset is checked to ensure there are no leaks or damage through corrosion, impact from natural causes, or debris and other objects (e.g. fishing nets). Therefore, the useful life of the pipeline assets is extended and the risk of failure is reduced, which may have significant consequences for both the natural and human environment.

ROVs are commonly used for inspections of subsea pipelines and power transmission cables [1] as seen in Figure 1.1. In a typical inspection, a surface vessel deploys a ROV which is piloted over the pipeline, collecting survey data from multiple sensors/instruments. A typical survey dataset comprises of: 1) video footage recorded from three camera angles (left/port, centre, and right/starboard), 2) Inertial Measurement Unit (IMU) data to capture the orientation of the ROV, 3) Multibeam Echosounder (MBE) data to map the seabed surface, and 4) magnetic pipe tracker to record the pipe location when it is buried below the seabed [2], but can also consist of information from more instruments.

These data are inspected by survey supervisors on the vessel to determine the overall condition of the pipeline and to ensure the installation is secure. The first annotation



Figure 1.1: Subsea Pipeline Inspection using Remotely Operated Vehicles

procedure is performed by trained Data Coordinators on the vessel while the data are captured; a Data Coordinator provides an ongoing commentary on the video feed received from a ROV. Therefore, the speed at which the ROV is piloted is limited by the rate at which the human can vocalise the presence of an event on audio commentary rather than a limitation of the craft. Typical vessel speeds whilst conducting surveys can be from 2 knots to 4 knots instead of 6 to 8 which is their maximum speed, adding to the time required to conduct surveys, driving the cost of the inspection further up.

During an inspection, apart from the real-time commentary, Data Coordinators press buttons to create timestamps to annotate events of interest such as pipeline Exposure, Burial, Field Joints, Anodes, Free Spans and others less regular such as boulders, nets, and debris. The annotation is a tedious and intense process, as inspectors have to watch several monitors for hours as seen in Figure 1.2, and therefore, prone to human error [3] as they become fatigued and distracted, which can lead to missed events or incorrect labelling.

For Exposure and Free Span events, the annotators do not rely solely on video footage, but have information from more instruments, such as MBE which maps the seabed terrain. This makes the annotation of these events more consistent, while the Anode and Field Joint events could be missed during the real-time annotation, and therefore, after the initial annotations, the video and commentary are subject to Quality Control (QC), either while the survey is ongoing or once completed, creating a

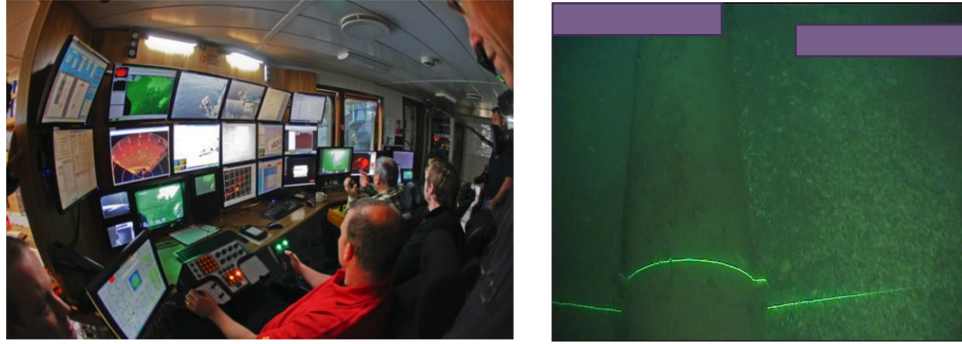


Figure 1.2: Survey Console Room and Event of Pipeline Exposure

bottleneck in the speed of processing and reporting. The annotations are verified again by the QC Data Coordinators, before generating the client report. Routinely, QC Data Coordinators have annotation data from previous surveys conducted on the same assets, which assist in the new survey; this eliminates any missed events, especially the Anode and Field Joint events.

The data from the non-visual and visual ROV sensors can complement each other to obtain accurate event detection. However, in this thesis only the central camera video feed of the ROV has been used, and datasets are curated to assist the training of DL image and video classification models, because the three cameras are not perfectly synchronised with the data provided by the industrial partner N-SEA. Similar methodologies can be applied that use image data from left and right ROV cameras.

1.2 Benefits of Automation

Although there have been many advances in subsea technology, Inspection, Maintenance and Repair (IMR) of subsea pipelines still require significant human intervention as seen in as illustrated in Figure 1.3. This can lead to inconsistencies on the annotations that depend on the specific expert annotators. Recent developments in the field of computer vision, led by significant enhancements provided by DL architectures, enable the automatic recognition of survey events from visual footage. This work aims to investigate several DL techniques for image analysis, such as multi-class and multi-label image classification, video classification, and domain adaptation, that can automate

the annotation process in subsea pipeline inspection.

This automation can transform pipeline inspection operations; allowing for faster annotations, as the ROVs will be operated in their full speed capacity and potentially more accurate as human annotators can be inconsistent. In addition, it will also minimise and simplify human intervention, freeing up the operator to perform more adequate tasks while reducing the presence of offshore staff and the concomitant cost and safety risks. Therefore, both the time and the cost of the whole process will be reduced.

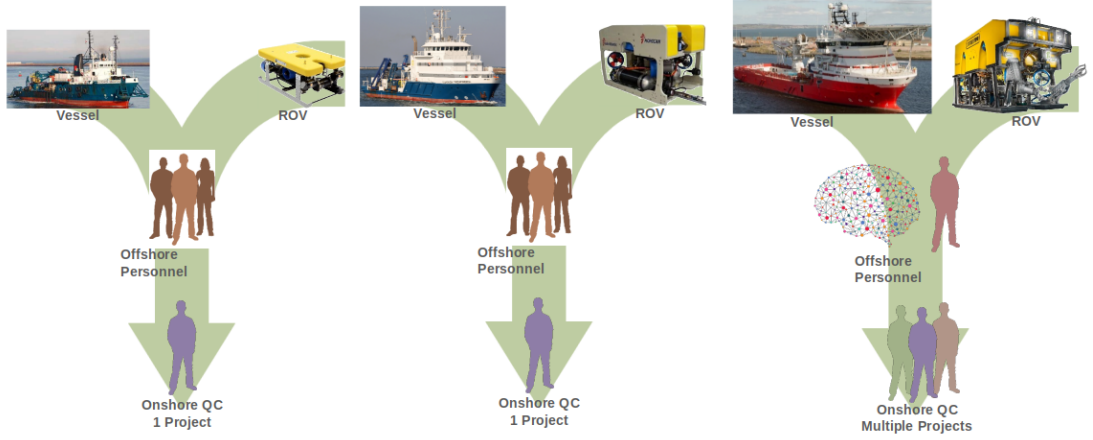


Figure 1.3: Evolution of Subsea Pipeline Inspection

For subsea pipeline inspection, the acceptable level of performance using AI can be particularly high due to the critical nature of maintaining the integrity of these pipelines. The specific performance requirements for subsea pipeline inspection using AI can depend on various factors, including industry standards, regulations, and the consequences of pipeline failures. While there is no universally defined acceptable level of performance, it is generally expected that AI-based inspection systems for subsea pipelines should strive for high accuracy rates. This work demonstrates that levels of over 90% of accuracy can be obtained, however, this also depends on the performed task and dataset; image classification, video classification, domain adaptation.

1.3 Research Questions

Towards automating the process of visual pipeline inspection the questions this thesis tries to answer are:

- The events recorded in the subsea pipeline survey are not mutually exclusive. For example, in order for a Field Joint (FJ) to be detected, the pipeline must be exposed (Exposure (E)). The events are described in Section 3.12.1. In addition, there are rare occasions that three events can occur at the same time; during Exposure (E), a Free-Span (FS) event is detected and an Anode (AN) is visible during this FS. Can events that occur simultaneously in frames of subsea pipeline surveys be recognised at the same time using a DL image classifier? In addition, what is good practise in this application to measure generalisation between training, validation, and testing sets?
- Subsea survey videos have a frame rate of 25 fps. Using an image classifier to perform predictions on each frame of a video clip results in fluctuated predictions, which can lead to False Positives (FP) or False Negatives (FN). To reduce these fluctuations, models that take multiple frames as input are examined. Which model is more appropriate to use in this application that operates directly on frame sequences instead of individual frames?
- A significant concern when integrating an automatic annotation framework in the inspection process is the transferability of the DL model on new surveys. To this end, can a model architecture or a dataset be adapted to enable it to sustain performance between domains? How can the shift between two survey datasets be measured? Finally, if a model is to be re-trained with a mix of events from both domains, how many events must be manually annotated to build a high performing model for the target domain?

1.4 Research Contributions

This thesis presents an analysis of various DL techniques that aim to automate subsea pipeline inspection using data from the central camera of the ROV. The main contributions of this thesis are as follows.

- Develops a framework that converts multi-class into multi-label image classification to detect five events of interest in subsea survey images, and utilises threshold search using PR curves to balance the precision and recall trade-off. This consists of the first work that detects events of Burial, Field Joint, Anode and Free Span in a multi-label fashion.
- Explores the effect of dataset splitting, i.e. on a per-event or per-frame basis, and highlights the need to split the dataset based on events to obtain accurate generalisation performance.
- Develops three models that fuse spatial and temporal information to perform classification of video segments in the subsea survey context and compares them in terms of performance and computation needs. This work is the first that operates directly in subsea pipeline video clips instead of individual frames.
- Develops a two-step domain adaptation methodology that combines the SAE architecture with a classifier module, along with data synthesis and retraining to take advantage of texture information from the target inspection dataset. There is no other work present in the literature that examines domain adaptation in the subsea survey context.
- Proposes a methodology for in- and out-of-domain shift characterisation using FID and further examines the percentages of target events that need to be added to the source dataset to improve performance on the target.
- Datasets that can provide a benchmark for subsequent studies¹. The datasets have been curated for the needs of this work from the raw data provided by

¹Subject to data owner permission.

Chapter 1. Introduction

the industrial partner N-SEA. The methodology for curating these datasets is described in Section 4.12. Furthermore, as new DL models with higher performance and reduced need for computational power become available, the proposed frameworks and datasets can be readily used to train evaluate these architectures.

These main contributions have been published as follows.

1.4.1 Journals

- Stamoulakatos, A.; Cardona, J.; McCaig, C.; Murray, D.; Filius, H.; Atkinson, R.; Bellekens, X.; Andonovic, I.; Lazaridis, P.; Hamilton, A.; Hossain, M.M.; Di Caterina, G.; Tachtatzis, C. Automatic Annotation of Subsea Pipelines Using Deep Learning. *Sensors* 2020, 20, 674. <https://doi.org/10.3390/s20030674> [4]

1.4.2 Conferences

- Stamoulakatos, A.; Cardona, J.; Michie, C.; Andonovic, I.; Lazaridis, P.; Bellekens, X.; Atkinson, R.; Hossain, M.M.; Tachtatzis, C. "A Comparison of the Performance of 2D and 3D Convolutional Neural Networks for Subsea Survey Video Classification," *OCEANS 2021: San Diego – Porto*, 2021, pp. 1-10, <https://doi.org/10.23919/OCEANS44145.2021.9706125> [5]

1.5 Thesis Outline

The content of this thesis is as follows.

- **Chapter 2** reviews existing work on automating subsea pipeline inspection using MBE and visual sensors and describes the challenges that exist in subsea visual footage, highlighting the need to use DL instead of traditional image processing. Furthermore, it provides existing DL methodologies used in other inspection fields to showcase the rise of deep learning in inspection processes.
- **Chapter 3** provides an extended DL literature review, along with the methodology required for DL tasks. It contains information on the fundamentals of Neural

Networks (NNs) and CNNs, as well as a multi-class subsea pipeline survey image classification workflow.

- **Chapter 4** introduces a methodology for automatic annotation of subsea pipeline survey frames consisting of multi-label image classification. It also provides a methodology for threshold selection and a comparison of model sizes. Furthermore, a comparison of the generalisation of models trained with and without event information is presented.
- **Chapter 5** moves forward from image to video classification. A comparison of three spatio-temporal architectures that are implemented with the goal of automating subsea survey video annotations is provided. Additionally, several techniques are illustrated to mitigate the challenges of the subsea survey video dataset created for the needs of this work.
- **Chapter 6** addresses the issue of domain shift between two subsea pipeline surveys and measures adaptation performance from source to target surveys. An experiment was carried out to compare the intermediate normalisation layers of the ResNet-50 architecture. Second, a two-step domain adaptation technique that combines an image-translation architecture with a classifier module is explored in an attempt to create a model that is trained on domain-invariant features. Finally, a new dataset is synthesised and, when added to the training, it is proven to be beneficial for cross-domain generalisation. A method is examined that identifies the domain shift between the two subsea pipeline surveys that uses FID, as well as an experiment that indicates that a sufficient number of target survey events should be added to the source training set to achieve better performance in the target survey.
- **Chapter 7** provides a summary of the contributions made in Chapters 4, 5, 6. Suggestions for potential future directions are also provided in this final chapter.

Chapter 2

Literature Review

2.1 Introduction

The objective of this chapter is to provide a review of the work that aims to automate the subsea pipeline inspection process. The automation of subsea pipeline inspection uses data collected from the instrumentation mounted on a ROV. These may include sonars, magnetometers, a still camera, a manipulator or cutting arm, water samplers, and instruments that measure water clarity, light penetration, and temperature [6], [7].

Figure 2.1 provides a diagram showing the connections between the subsea pipeline inspection field and other subfields that derive from it or provide information on it. The subsea pipeline inspection automation works can be divided into studies that use the feed from non-visual (MBE) and visual sensors (cameras). This thesis focusses on methodologies that use the camera feed, but a review on the MBE sensor is also provided in Section 2.2. In image-related studies, the methodologies can be divided into traditional image processing (Section 2.3), which has two main methods with pipeline segmentation and contour detection, and DL approaches (Section 2.7).

In addition, the challenges in underwater visual footage created by the dynamic nature of the subsea environment are explored in Section 2.5, and it is made evident that they have led to the increasing use of DL for other subsea applications such as the identification of coral reefs and *Posidonia Oceanica* [8], image enhancement, inspection of marine structures and detection of debris and fish objects that are presented in Sec-

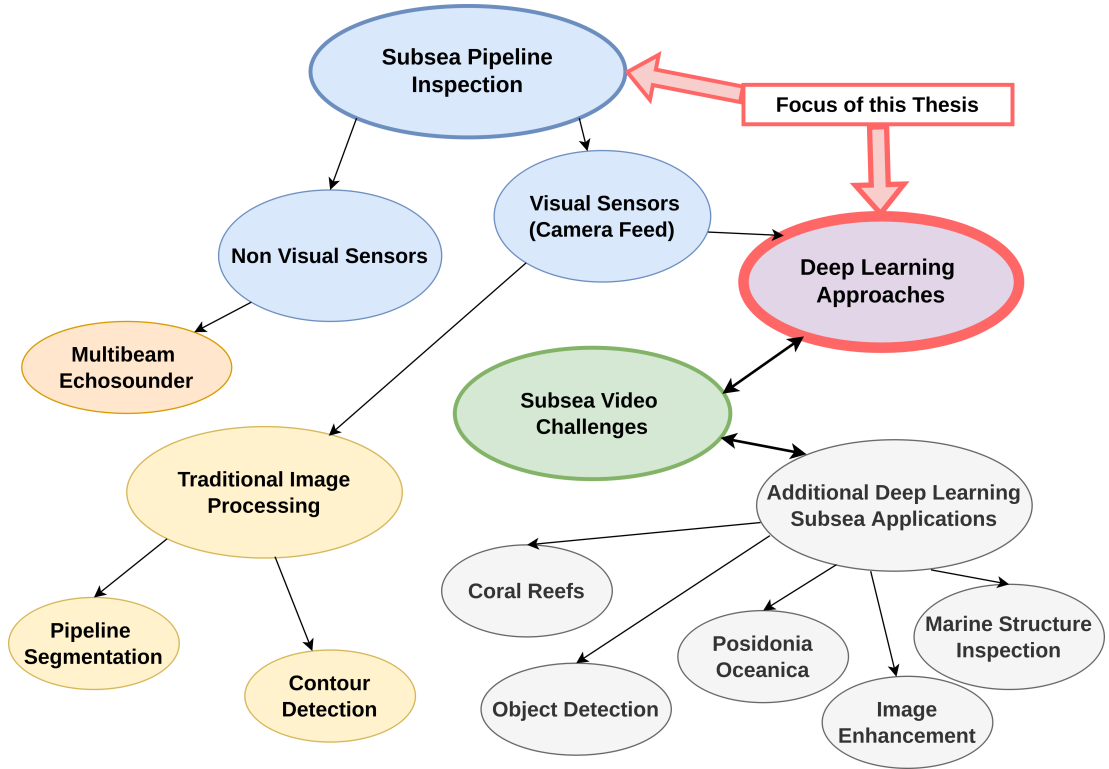


Figure 2.1: Subsea Applications

tion 2.6. Finally, an overview of DL work for power line and manufacturing inspection is presented to highlight the increase of DL in inspection processes in Section 2.8.

2.2 Subsea Inspection using Multibeam Echosounder

The literature on pipeline tracking using MBE is scarce due to the commercial / private nature of this application. AUTOTRACKER [9] is one of the main projects that employed a MBE for pipeline tracking in real scenarios. Petillot *et al.* [10] used a sidescan sonar for the initial detection of the pipe at long range. Once detected, it is tracked in MBE using a Metropolis-Hastings Monte Carlo [11] method, where the likelihood distribution is computed by fitting the pipeline model to MBE data. Another study by Pavin [12] described a fuzzy logic-based methodology for pipe detection in MBE, where the likelihood of each data point belonging to the pipe is calculated based on its height with respect to adjacent data points. Wide-band sonar capabilities have been

explored for cable and pipeline tracking by Pailhas *et al.* [13], where they argue that its low-frequency operation makes it less appealing for profiling applications. Paull *et al.* [14] provided a review of the trade-offs of different sonar paradigms where they argued that a MBE is more suitable for tracking subsea pipelines at close range, while a sidescan sonar can have a blind spot at the centre, which is undesirable as other sensors collecting survey data need to be at close range over the pipe. Furthermore, they argued that Synthetic Aperture Sonars (SAS) are not suitable for pipeline tracking missions, as they are limited to tight prescribed speeds and, therefore, this comes with decreased manoeuvrability, while they are better for imaging at longer ranges. Bharti *et al.* [15] used MBE data for pipeline detection and orientation estimation, while they implemented a method to filter out false positives. The proposed framework is tested in both simulated and real scenarios, and they conclude that it performs well at close range, but a sidescan sonar can be used to complement this approach at long range.

Multi-beam echosounder approaches are widely utilized in subsea pipeline surveys for their effectiveness in detecting and tracking pipelines. However, their limitation lies in their inability to directly capture and analyze visual information. These systems rely on acoustic signals to generate maps of the seafloor and pipelines, making them unsuitable for detecting events of interest with visual features, which can provide essential insights into the condition of the pipeline and identify potential issues.

Visual-based techniques, such as traditional image processing and Deep Learning (DL), offer a solution by leveraging visual information for event detection and classification. Integrating both approaches can lead to more comprehensive subsea pipeline inspections and improved understanding of the pipeline's condition.

2.3 Traditional Image Processing

Many approaches by the AUV navigation community reported to date use traditional signal processing on subsea image data. This work on optical camera-based subsea pipeline detection can be categorised into pipeline segmentation and this presented in Table 2.1, where a mask is generated, and pipeline contour detection which is presented

in Table 2.2, or a combination of both.

Table 2.1: Pipeline Segmentation Work

Authors	Method	Year
Hallset [16]	They used edge detection followed by the region growth algorithm [17] to create a segmentation mask that produces multiple segments. A rectangle matching algorithm was used on these segments to obtain the final detection.	1991
Grau <i>et al.</i> [18]	They presented a machine learning-based algorithm for cable detection, which is equally applicable to pipelines; it first uses some sample images to learn the features using a clustering algorithm, then a feature matching-based segmentation is done in the input image.	1998
Antich and Ortiz [19]	They presented a histogram-based cable segmentation in which the input image is divided into multiple sections to speed up the process. The cable segmentation was followed by contour detection, where the linearity was checked to ensure that these points were obtained from a cable.	2003
Rizzini <i>et al.</i> [20]	They presented another histogram-based algorithm and clustering.	2015
Khan <i>et al.</i> [21]	They presented a technique that performs image dehazing prior to using an entropy-based method for segmentation of the pipelines along with a variant of edge detection followed by Hough transform [22].	2017
Khan <i>et al.</i> [23]	They reported on a method for image enhancement of underwater pipelines using wavelet autoencoding and K-means to cluster corrosion segments.	2018

Table 2.2: Pipeline Contour Detection Work

Authors	Method	Year
Conte <i>et al.</i> [24]	They developed a real-time vision-based detection system for underwater pipelines using edge-based image processing to detect pipeline contours and a Kalman [25] filter that utilises the navigation data to reduce the effect of motion disturbances.	1997
Zingaretti <i>et al.</i> [3]	They used a contour detection algorithm and a mathematical model where the pipeline was modelled using two straight lines. They split the image into horizontal strips and individually process each segment to detect the end points of the pipeline using edge detection. Line fitting was performed at these points to obtain the contour of the pipeline.	1998
Ortiz <i>et al.</i> [26]	They proposed a method to identify the contours of the submarine cable in tandem with a linear Kalman [25] filter to predict the contours in the following frame.	2002
Asif <i>et al.</i> [27]	They proposed a pipeline tracking method that uses the Bresenham line algorithm and B-Spline to detect noise-free pipeline contours.	2006
Narimani <i>et al.</i> [28]	They proposed a pipeline and cable tracking technique to improve vehicle navigation by converting images to greyscale and applying the Hough transformation to determine the angle between vehicle and pipeline; subsequently, they used this as a reference to an adaptive sliding mode controller.	2009
Zhang <i>et al.</i> [29]	They improved Asif's method [27] by using green light that has a minor absorption effect to address the problem of illumination.	2012

Drews <i>et al.</i> [30]	They added morphological dilation to the edge detection output to enhance edges before applying the Hough transform.	2012
Jacobi <i>et al.</i> [31,32]	They proposed a pipeline tracking method for Autonomous Underwater Vehicle (AUV) guidance through the fusion of optical, magnetic and acoustic sensors applied on simulated pipeline data.	2013, 2014
Jacobi and Kari-manzira [32]	They did not focus solely on camera data, they fused data from multiple sensors to generate a feature map and then used the Hough transform and contour detection to locate the pipe with respect to AUV.	2014
Petraglia <i>et al.</i> [33]	They proposed a method for pipeline tracking, in which, after the initial preprocessing of Red Green Blue (RGB) pipeline images, the pipeline boundaries are detected by first filtering the edges through Non-Maximum Suppression (NMS) to eliminate horizontal line segments followed by Random Sample Consensus (RANSAC) and Total Least Square (TLS) to group segments.	2017

Traditional image processing techniques for subsea pipeline event detection have limitations when compared to DL approaches. One of the main drawbacks is their reliance on hand-crafted features and manual hyperparameter tuning. This requires experts to manually design and extract features, which can be time-consuming and require domain-specific knowledge. Additionally, manual tuning of hyperparameters is challenging as the optimal values depend on the specific subsea dynamic environment.

Furthermore, traditional image processing methods have inadequate support for subsea inspection in different fields. The hand-crafted features and manual tuning are typically tailored to specific conditions or environments, making them less adaptable to different scenarios. Subsea environments are highly dynamic, with varying lighting conditions, water turbidity, and other factors affecting image quality. Traditional

methods struggle to adapt to these changes since they lack the ability to learn and adjust their features automatically.

The techniques proposed in traditional image processing for subsea pipeline event detection have primarily focused on pipeline tracking. While tracking the pipeline is an important task, it is not sufficient for comprehensive event detection and annotation. Events of interest in subsea inspection can include anomalies, damages, leaks, anodes, field joints or other critical occurrences that require attention.

In contrast, DL approaches offer significant advantages in subsea pipeline event detection. Deep learning models, such as CNNs that are used in this thesis, can automatically learn features directly from the input data. This eliminates the need for manual feature engineering and allows the models to adapt and generalise to different subsea dynamic environments. Deep learning models can also learn to detect and classify various events of interest, as they can capture complex patterns and dependencies in the data.

Moreover, DL models have the potential to continually improve their generalisation performance through training on large datasets. As more subsea inspection data becomes available, DL models can be trained to become more accurate and robust in detecting and annotating events. This adaptability and learning capability make DL approaches more suitable for subsea inspection in different fields, where the conditions and events may vary widely.

2.4 Underwater Image Datasets

The availability of subsea pipeline imagery for training DL models in automated subsea pipeline inspection is limited due to the commercial and private nature of this application. As a result, publicly available subsea pipeline imagery is scarce, posing a significant challenge for developing DL models specifically tailored to this domain. In this thesis, this limitation is addressed by collaborating with the industrial partner N-SEA and curating the data from their proprietary sources as described in Section 4.12.

Although accessing subsea pipeline imagery is challenging, publicly available datasets have played a crucial role in advancing DL techniques for marine inspection and other

underwater applications. These datasets focus on tasks such as object detection, segmentation, and image restoration in underwater environments. While they may not directly represent subsea pipeline inspection scenarios, they serve as valuable resources for training and evaluating DL models in related underwater applications and they are presented in Table 2.3.

Table 2.3: Underwater Image Datasets

Dataset	Description	Task
Detecting Underwater Objects (DUO) [34]	DUO is a dataset for Underwater object detection for robot picking. The dataset contains a collection of diverse underwater images with more rational annotations.	Object Detection
UDD [35]	UDD is an underwater open-sea farm object detection dataset. UDD consists of 3 categories (seacucumber, seaurchin, and scallop) with 2,227 images.	Object Detection
FathomNet [36]	FathomNet is an open-source image database that can be used to train, test, and validate state-of-the-art artificial intelligence algorithms to understand the ocean and its inhabitants. Inspired by annotated image databases such as ImageNet and COCO, FathomNet aims to establish the same kind of reference data set for images of ocean life. The long-term goal of FathomNet is to aggregate $> 1k$ fully annotated and localised images per marine species of Animalia ($> 200k$), with the ability to expand and include other underwater concepts (e.g., substrate type, equipment, debris, etc.) for training and validating machine learning models.	Object Detection

DeepFish [37]	DeepFish as a benchmark suite with a large-scale dataset to train and test methods for several computer vision tasks. The dataset consists of approximately 40 thousand images collected underwater from 20 habitats in the marine environments of tropical Australia. It contains classification labels as well as point-level and segmentation labels to have a more comprehensive fish analysis benchmark. These labels enable models to learn to automatically monitor fish count, identify their locations, and estimate their sizes.	Classification, Segmentation, Object Detection
TrashCan [38]	The TrashCan dataset is an instance-segmentation dataset of underwater trash. It comprises annotated images (7,212 images) that contain observations of trash, ROVs, and a wide variety of underwater flora and fauna. The annotations in this dataset take the format of instance segmentation annotations: bitmaps containing a mask marking which pixels in the image contain each object. The imagery in TrashCan is sourced from the J-EDI (JAMSTEC E-Library of Deep-sea Images) dataset, curated by the Japan Agency of Marine Earth Science and Technology (JAMSTEC).	Instance Segmentation

Segmentation of Underwater Imagery (SUIM) [39]	The SUIM dataset contains over 1500 images with pixel annotations for eight object categories: fish (vertebrates), reefs (invertebrates), aquatic plants, wrecks/ruins, human divers, robots, and sea-floor. The images have been rigorously collected during oceanic explorations and human-robot collaborative experiments, and annotated by human participants.	Instance Segmentation
Large Scale Underwater Image Dataset (LSUI) [40]	LSUI dataset includes 5004 image pairs, which involve richer underwater scenes (lighting conditions, water types and target categories) and better visual quality reference images than the existing ones.	Image Restoration
Underwater Image Enhancement Benchmark Dataset (UIEB) [41]	UIEB includes 950 real-world underwater images, 890 of which have the corresponding reference images.	Image Restoration
Heron Island Coral Reef Dataset (ICRD) [42]	HICRD is a large-scale real underwater image dataset for underwater image restoration. There are 2000 reference restored images and 6003 original underwater images in the unpaired training set.	Image Restoration

MSRB (Marine Snow Removal Benchmarking) [43]	MSRB is a benchmarking dataset for marine snow removal of underwater images. Marine snow is one of the main degradation sources of underwater images that are caused by small particles, e.g., organic matter and sand, between the underwater scene and photosensors. The dataset consists of large-scale pairs of ground-truth and degraded images to calculate objective qualities for marine snow removal and to train a deep neural network. We propose two marine snow removal tasks using the dataset and show the first benchmarking results of marine snow removal.	Image Restoration
--	--	-------------------

2.5 Challenges of Subsea Visual Footage

From a technical perspective, there are several challenges to be addressed to automate the process of annotating subsea pipeline surveys at a level of precision similar to that of the inspection engineer. They come as a consequence of two main factors, poor image quality and camera motion. The lack of contrast due to scarce illumination and the presence of suspended particles in the water (e.g., sand, algae) are typical problems in underwater imaging [44]. Furthermore, as a result of ROV manoeuvres and the consequent camera movement, the position and orientation of the pipeline in the image plane may vary greatly compared to the previous frame [3].

In addition, images of surveys conducted in different areas and times contain different colours, viewing angles, highly variable illumination, and text overlays. It is also observed that the events themselves differ. Examples include Anodes that have different colours as a result of variations in the depletion rates and material, while the Field Joint structure differs and may vary in the depressions on the pipeline surface. In addition, marine growth and vegetation change with time and different locations

(seabed peculiarities are location-specific due to climate changes, etc.), cameras, lighting, and video capture perspectives (relative positioning of the pipeline landmarks to the camera field of view). This is further explored in Chapter 6.

Traditional image processing approaches such as contour detection and their variants, although suitable to localise the edges of the pipeline, require significant feature engineering to detect other events of interest such as Field Joints, Free Spans, and Anodes. Sea life, marine growth, seabed settlements, auxiliary structural elements, breakage in the external pipeline sheathing, and alien objects near the pipeline are possible sources of confusion in the detection of the pipeline contours. Furthermore, it is unclear how these algorithms perform in the absence of the pipeline (when the pipe is buried) or on changes in position and orientation as the ROV manoeuvres, both of which result in significant variations of the event appearance in the image plane.

2.6 Deep Learning for Subsea Applications

Apart from using DL to automate subsea pipeline inspection, there are various DL applications for other types of subsea visual footage, namely; marine structure inspection, *posidonia oceanica*, coral reefs, underwater object detection, and underwater image enhancement.

2.6.1 Marine Structure Inspection

Bonnin-Pascual and Ortiz [45] presented a framework to detect vessel defects. They precalculated and combined a range of multi-scale normalised feature maps with the use of Gaussian and Gabor pyramid filters. The framework was successfully tested on image mosaics during vessel inspection campaigns. Obyrne *et al.* [46] proposed the use of synthetic photorealistic imagery for training DL models that can be applied to detect biofouling on marine structures; SegNet [47] was trained using 2500 annotated synthetic images of size 960×540 pixels, where the images were rendered in a virtual underwater environment under a wide variety of conditions and feature biofouling of various sizes, shapes, and colours and each rendered image had a corresponding ground truth mask.

After training in synthetic imagery, SegNet was applied to segment new real-world images. Stanchev *et al.* [48] developed a website that allows quick annotations of underwater biological survey videos, the creation of a short tracking video for each annotation, the verification of existing annotations and tracking videos, and the training of NN models from existing annotations to automatically annotate new videos.

2.6.2 Posidonia Oceanica

Martin Abadal *et al.* [49] proposed a framework for the semantic segmentation of Posidonia Oceanica [8], where a fully convolutional network was implemented using a pre-trained on ImageNet VGG16 [50] as encoder and FCN8 [51] as decoder with Gaussian initialisation of its parameters and hyperparameter tuning. The model was successfully employed on a Turbot AUV for online meadow segmentation. Bonin-Font *et al.* [52] performed image feature extraction (168 features) and compared the performance of 14 classifiers in Posidonia Oceanica detection, mapping and quantification tasks. Principal Component Analysis (PCA) was applied on the best performing model (Logistic Model Trees) to select the 25 more relevant features and retrain the classifier.

2.6.3 Coral Reefs

Mahmood *et al.* [53] proposed a DL method for classifying coral reefs, trained with images of the sea floor that have been collected with the help of ROVs and AUVs. The proposed technique is then applied to unlabelled coral reef mosaics in the Abrolhos Islands, Western Australia, to automate the annotation of them. King *et al.* [54] compared the performance of five different CNN architectures in the formulation of patch-based CNN methods, where ResNet-152 [55] was found to perform the best in the annotated data set of underwater coral reef images. The results of four different Fully Convolutional Neural Network (FCNN) [56] models for the semantic segmentation of coral reef images were also compared and examined. Finally, a tool was developed for the fast generation of segmentation maps to serve as ground truth segmentation masks for the FCNN models. The FCNN architecture Deeplab [57] was observed to yield the best results for semantic segmentation of underwater coral reef images.

2.6.4 Underwater Object Detection

Jeon *et al.* [58] presented an approach for creating a dataset using a 3D Computer Aided Design (CAD) model for DL-based underwater object detection and pose estimation and a simple pose estimation network for underwater objects was introduced. They showed that object detection and pose estimation networks trained using synthetic data present potential for DL underwater approaches. Fulton *et al.* [59] compared DL algorithms on the task of detecting trash in realistic underwater environments, with the ultimate goal of exploring, mapping and extraction of such debris using AUVs. A large and publicly available debris dataset in open water locations was annotated for training CNN architectures for object detection. The trained networks were then evaluated on a set of images from other parts of that dataset, providing information on approaches for developing the detection capabilities of an AUV for underwater trash removal. Xu *et al.* [60] used YOLO [61] to recognize fish in underwater video using three very different datasets recorded at real-world water power sites.

2.6.5 Underwater Image Enhancement

Guo *et al.* [44] presented a DL framework that improves the quality of underwater images using a custom Generative Adversarial Network (GAN) [62]. Xie *et al.* [63] first split the underwater image in two representations of reflectance and illumination, concatenated it back by merging the colour channels and then employed a UNet [64] architecture using perceptual loss [65] to enhance the original image. Chen *et al.* [66] used two CNN branches to calculate a ‘Backscatter’ estimation and a ‘Direct-transmission’ estimation and then they obtained the enhanced image through reconstruction.

2.7 Deep Learning for Subsea Pipeline Inspection

Deep Learning can yield improved performance as it allows multiple processing layers to learn features by themselves, contrary to conventional ML approaches, which cannot process the data in their natural form. Given the importance of inspection of subsea pipelines, as well as the challenges mentioned above of subsea visual footage, only a few

methods have been proposed in recent years to automate different features of pipeline detection using DL and these are presented in Table 2.4.

Table 2.4: Deep Learning for Subsea Pipeline Inspection

Authors	Method	Year
Foresti <i>et al.</i> [67]	They proposed a NN to perform image segmentation of pipeline borders. They argue that this approach better addresses the lack of luminosity, since the NN takes into account the global properties of the images. A reasoning-based post-processing algorithm was then used to avoid false positives occurring from seaweed and other sea growth.	2000
Petraglia <i>et al.</i> [33]	They examined two NN architectures for classification of four types of events: inner coating exposure, algae, flange, and concrete blankets are compared. The first NN architecture utilises two convolutional and three fully connected layers, trained on segmented pipelines from the pre-processed images. The second architecture adopted a Multilayer Perceptron (MLP) with a single hidden layer, trained on features extracted from 3-level Wavelet decomposition. The results led to the conclusion that the CNN outperforms the MLP, without the need for manual feature extraction.	2017

Khan <i>et al.</i> [23]	This method addressed the challenges of underwater imaging by developing image restoration and enhancement algorithms to minimize blurring effects and enhance color and contrast. The algorithms are tested on experimentally collected and publicly available hazy underwater images, achieving reasonable accuracy in corrosion estimation. The enhanced colors in the imaging data aid in the corrosion estimation process, allowing for differentiation between corroded and non-corroded surface areas. The qualitative and quantitative analyses demonstrate promising results, supporting the integration of this method into a robotic system for real-time underwater pipeline corrosion inspection.	2018
Thum <i>et al.</i> [68]	This study proposed the use of deep CNN models for underwater cable image classification. Transfer learning and data augmentation techniques are applied to enhance classification accuracy. Among the tested models, MobileNetV2 [69] achieves the highest accuracy of 93.5% in classifying underwater cable images, with lower computational time.	2020
Medina <i>et al.</i> [70]	This study focused on two types of neural networks, namely U-Net and Deeplabv3+, for semantic segmentation. The findings revealed deep CNNs outperform traditional computer vision techniques. The comparative analysis between U-Net [71] and Deeplabv3+ [57] indicates that Deeplabv3+ achieves superior results and demonstrates robust performance in challenging underwater conditions.	2020

Bharti <i>et al.</i> [72]	This work finetuned U-Net [71] in a self-supervised setting using multibeam echosounder data for the detection and segmentation of subsea pipelines.	2020
Li <i>et al.</i> [73]	This paper presented a strategy for subsea pipeline identification using YOLOv5 [74]. The strategy utilizes acoustic images acquired by a Side Scan Sonar (SSS) and involves segmenting the bar image formed by SSS into sub-images. These sub-images are then inputted into a pre-trained YOLOv5 model for subsea pipeline extraction. The proposed strategy achieves a high average precision (AP) of 97.62% with a low time consumption of 304ms for a 10-second period bar image.	2021
Gasparovic <i>et al.</i> [75]	Six different CNN detectors were trained and tested, including variations of the YOLO [76] architecture (YOLOv4, YOLOv4-Tiny, CSP-YOLOv4, YOLOv4@Resnet, YOLOv4@DenseNet) and the Faster R-CNN [77] architecture. The models are evaluated in terms of detection accuracy, mean average precision (mAP), and processing speed measured in Frames Per Second (FPS) using a custom dataset of underwater pipeline images. The YOLOv4 model demonstrates superior performance for underwater pipeline object detection, achieving an mAP of 94.21% and real-time object detection capability.	2022

These works have made valuable contributions to the field by improving the accuracy and reliability of subsea pipeline inspection systems. However, it is worth noting that the scope of these works has been somewhat limited, primarily addressing segmentation and corrosion detection.

One area that has received limited attention in the existing literature is the iden-

tification of burial, anodes, field joints, and free spans in subsea pipelines. These aspects play a crucial role in the overall integrity of subsea pipelines and are essential for ensuring their long-term reliability. Surprisingly, there is a dearth of reported works specifically dedicated to addressing these important aspects in subsea pipeline detection.

Furthermore, another notable gap in the existing literature is the absence of research on video classification and domain adaptation for subsea pipelines. With the increasing availability of video data captured during underwater inspections, the need for accurate and efficient classification techniques becomes crucial. Despite this demand, no reported works have explored the application of video classification and domain adaptation specifically for subsea pipelines.

In light of these gaps in the literature, this thesis aims to fill the void by introducing novel methodologies and techniques that address the identification of burial, anodes, field joints, and free spans in subsea pipelines. Furthermore, this thesis also pioneers the application of video classification and domain adaptation techniques to the domain of subsea pipelines. By leveraging advancements in DL and computer vision, this research aims to develop robust algorithms capable of accurately classifying video data captured during underwater inspections, thereby enabling efficient automatic annotation of subsea pipeline survey.

In addition, one significant challenge that becomes evident is the lack of benchmark datasets in the field of subsea pipeline inspection for evaluating and comparing the performance of state-of-the-art techniques. The absence of standardized datasets poses a considerable hurdle when it comes to objectively assessing the effectiveness and robustness of different approaches. Consequently, the lack of benchmark datasets hinders the ability to make meaningful comparisons and impairs the overall progress in the field of subsea pipeline inspection.

2.8 Deep Learning for Power Line and Manufacturing Inspection

Recently, DL approaches have taken over inspection processes in different fields with the goal of automating them as the computational demands that these algorithms require have been provided. The work presented here is in the fields of power line inspection and manufacturing inspection. These works highlight the increasing use of DL in several inspection processes. Automation through DL can reduce the cost of these processes and, at the same time, produce more accurate and faster inspections. Figure 2.2 provides a diagram of inspection processes that use DL. Similar DL models and methods can be applied to a wide range of inspection processes with the same goal of making the process faster, more accurate, and continuous compared to human inspectors.

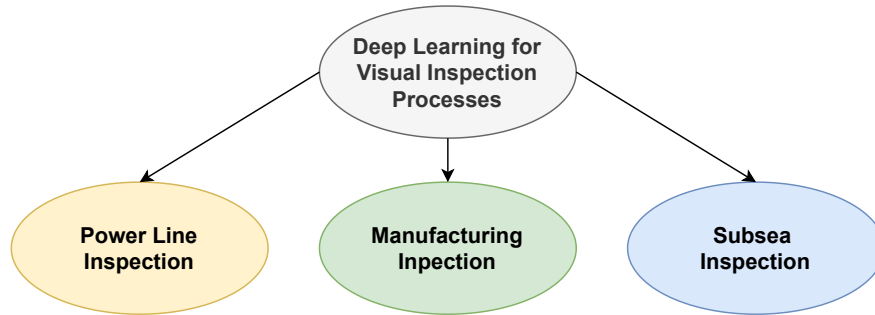


Figure 2.2: Inspection Taxonomy

2.8.1 Power Line Inspection

One inspection process that DL aims to automate is power line inspection, where models are used to perform object detection to identify specific features of power line assets and thus to make inspection faster and more continuous. Liu *et al.* [78] provided an in-depth discussion of DL technology in power line inspection, highlighting that after 2015, more than half of the publications on vision-based power line inspection use DL. Nguyen *et al.* [79] conducted a review of vision-based approaches for power line inspection and the potential role of DL and deployed a Single Shot Detector (SSD) [80]

on an AUV to detect components and faults in power lines [81]. Zhang *et al.* [82] detected electricity poles in Google Street View Imagery using RetinaNet [83] trained with 1,000 annotated images. Jalil *et al.* [84] used Faster-RCNN [85] to detect insulators in drone images. Miao *et al.* [86] implemented a custom SSD with MobileNet [87] as the backbone to detect insulators. Wan *et al.* [88] implemented and compared three custom object detectors to identify electrical fittings and transmission line defects. Vemula *et al.* [89] deployed and tested a Mask R-CNN model [90] on an AUV to detect power poles, insulators, and transformers. Jiang *et al.* [91] analysed and compared methods for infrared image recognition of patrol inspection of power equipment. Han *et al.* [92] used Fast-RCNN [93] to detect typical defects in the equipment of power substations, metre readings, infrared images, and humans. They also used a Kalman [25] filter to track the detected humans.

2.8.2 Manufacturing Inspection

The utilisation of DL and specifically of CNNs has been widely investigated for machinery fault diagnosis, and classification. Yang *et al.* [94] presented a review on the use of DL to detect defects in manufacturing and its challenges. Weimer *et al.* [95], designed a deep CNN architecture for defect detection, and different hyperparameters are examined to ensure the accuracy of the detection results. Masci *et. al* [96], compared a CNN that performs feature extraction directly from steel defect images with traditional Machine Learning (ML) (a combination of manual feature extraction and a multi-layer perceptron and support vector machine) and shows lower error rates in the former. Ren *et. al* [97] explored another CNN to automatically inspect dirties, scratches, burrs, and wears on surface parts, and the results show that it works properly with different types of defects on textured or non-textured surfaces. Park *et. al* [98] proposed a generic approach based on CNN to extract patch features and predict the area of the defect by thresholding and segmenting. CNNs have been also used in other applications of defect diagnosis such as bearing [99–101], gearbox [102] and rotors [103].

Liu *et al.* [104] utilised pix2pix [105] conditional GAN [106] to generate more samples and address the class imbalance that exists in industrial processes, and DenseNet [107]

to detect defects in these samples. Zheng *et al.* [108,109] used state-of-the-art DL methods to inspect surface defects in industrial products. In addition, they implemented a semi-supervised [110] framework to identify manufacturing defects while introducing a loss component and making extensive use of mix augmentations [111].

2.8.3 Defect Inspection Datasets

Defect inspection in different industries is vital for maintaining product quality, reducing costs, ensuring safety, complying with regulations, satisfying customers, and driving continuous improvement within an organization. In this Section, three inspection datasets are presented in Table 2.5. The methodologies of Chapters 4 and 6 can be applied to this dataset as well, as the task of multi-label image classification is the same with the subsea pipeline inspection.

Table 2.5: Underwater Image Datasets

Dataset	Description	Task
Synthetic Visual Inspections [112]	Synthetic visual inspection data of structural elements in bridges.	Regression (Deterioration State)
Pavementscapes [113]	Pavementscapes is a large-scale dataset to develop and evaluate methods for pavement damage segmentation. It is comprised of 4,000 images with a resolution of 1024×2048 , which have been recorded in the real-world pavement inspection projects with 15 different pavements. A total of 8,680 damage instances are manually labeled with six damage classes at the pixel level.	Instance Segmentation

Sewer-ML [114]	Sewer-ML is a sewer defect dataset. It contains 1.3 million images, from 75,618 videos collected from three Danish water utility companies over nine years. All videos have been annotated by licensed sewer inspectors following the Danish sewer inspection standard, Fotomanualen. This leads to consistent and reliable annotations, and a total of 17 annotated defect classes.	Multi-label Image Classification
----------------	--	----------------------------------

2.9 Conclusions

In conclusion, this literature review has highlighted the limitations of multi-beam echosounder approaches in subsea pipeline surveys, particularly their inability to capture and analyze visual information. It has emphasized the potential of integrating visual-based techniques, such as traditional image processing and deep learning (DL), to enhance subsea pipeline inspections and improve understanding of pipeline conditions.

Traditional image processing techniques have been found to have limitations due to their reliance on hand-crafted features and manual hyperparameter tuning, making them less adaptable and time-consuming. They also lack support for subsea inspection in different fields, where dynamic environmental factors can significantly impact image quality and require tailored approaches. Moreover, traditional methods primarily focus on pipeline tracking rather than comprehensive event detection and annotation.

In contrast, DL approaches offer significant advantages for subsea pipeline event detection. Deep learning models, such as convolutional neural networks (CNNs), can automatically learn features from input data, eliminating the need for manual feature engineering and allowing adaptation to diverse subsea environments. DL models can detect and classify various events of interest by capturing complex patterns and de-

dependencies in the data. Their adaptability and learning capability make them more suitable for subsea inspection in different fields.

While previous works have made valuable contributions to subsea pipeline inspection, there are several notable gaps in the existing literature. These gaps include the limited research on burial, anodes, field joints, and free spans identification in subsea pipelines, which are critical for ensuring long-term reliability. Additionally, the absence of studies on video classification and domain adaptation specifically for subsea pipelines is a significant gap, considering the increasing availability of video data from underwater inspections.

Addressing these gaps, this thesis aims to introduce novel methodologies and techniques for identifying burial, anodes, field joints, and free spans in subsea pipelines, as well as applying video classification and domain adaptation techniques to this domain. By leveraging advancements in DL and computer vision, the research aims to develop robust algorithms for accurate classification and efficient automatic annotation of subsea pipeline survey videos.

A significant challenge identified is the lack of benchmark datasets for evaluating and comparing the performance of state-of-the-art techniques in subsea pipeline inspection. The absence of standardized datasets hampers the ability to objectively assess the effectiveness and robustness of different approaches, hindering overall progress in the field.

In summary, this literature review has emphasized the potential of integrating visual-based techniques, particularly DL, to enhance subsea pipeline inspections. It has identified gaps in the existing literature, such as the limited focus on certain pipeline aspects and the absence of research on video classification and domain adaptation. The thesis aims to address these gaps and contribute to the advancement of subsea pipeline inspection methodologies. Furthermore, the need for benchmark datasets has been highlighted to facilitate objective evaluation and comparison of techniques in this field.

Chapter 3

Deep Learning Background

3.1 Introduction

This Chapter outlines the basic building blocks of deep CNNs, techniques, and methodologies that are used in this thesis to train and evaluate the performance of the developed models. In the context of automatic visual inspection of subsea pipelines, the work presented in this thesis focusses primarily on supervised learning. The remainder of this Chapter provides a detailed description of model architectures, the methodology for hyperparameter tuning, and the evaluation of model performance. Finally, the Chapter concludes with a case study on a subsea pipeline image classification task to demonstrate a complete DL methodology.

3.2 Background on Neural Networks

A NN is a computational model inspired by the human brain. Many of the recent advances that have been made in the field of Artificial Intelligence (AI) such as voice, image, and video recognition, music generation, use NNs. In simple terms, the goal of NNs is to find the function $F(x; \theta)$ with parameters θ (for example, weights and biases) that transform an input x into the predicted output \hat{y} (shown in Figure 3.1), while minimising the error to the ideal output \mathbf{y} .

The basic building block of a NN is a perceptron (also known as an artificial neuron), which mimics the structure of a biological neuron. The structure of the biological

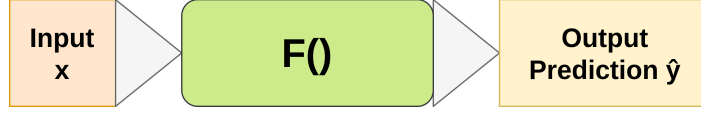


Figure 3.1: Neural Network as a Function

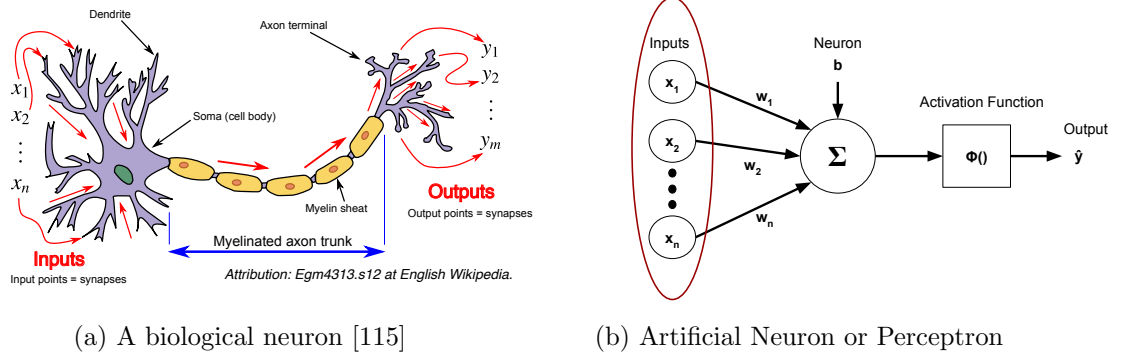


Figure 3.2: Comparison of a biological and an artificial neuron.

and artificial neurons is contrasted in Figure 3.2). The neuron receives one or more inputs; dendrite and sums them through the soma (body) to produce an activation potential that is transmitted via the axon (activation function; typically non-linear) to the synapses (output). The shape and structure of the dendrite is captured in the artificial neuron as weight multiplication. Formally, a neuron or perceptron is a function that takes an input vector $\mathbf{x}\{x_1, x_2, \dots, x_n\}$ and computes a dot product with the weight vector $\mathbf{w}\{w_1, w_2, \dots, w_n\}$ plus the addition of a bias b . ϕ is an activation function that provides non-linearity to the network. Typical activation functions are described in Section 3.4.3. Artificial neurons can then be combined to architect complex network topology of stacked layers.

$$\hat{y} = \phi \left(b + \sum_{i=1}^n w_i \cdot x_i \right) \quad (3.1)$$

3.3 Fully Connected Layers

In a fully connected layer, each neuron has full connections to all activations in the previous layer and each connection has its own weight, as seen in Figure 3.3. A NN computation in each layer can be considered as a cascaded chain of linear matrix multi-

plications followed by typically non-linear activation functions, where the input of each layer is the output of the previous one, hence transforming the input space. Therefore, neural networks solve the problem of transforming the input \mathbf{x} to the predicted output $\hat{\mathbf{y}}$. Learning is the process of finding the parameters values that minimise the error between the NN output $\hat{\mathbf{y}}$ and the true output \mathbf{y} . Therefore, the network output is determined based on the parameters of all interconnected layers.

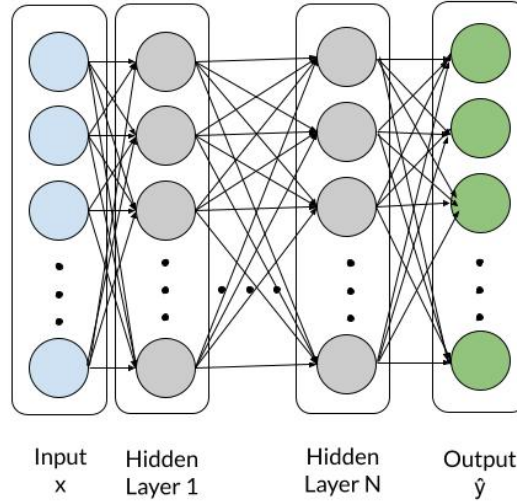


Figure 3.3: A Fully Connected Neural Network

3.3.1 Training of Neural Networks

The training process is an optimisation problem. Parameters are learnt during model training through sample data and optimising (minimising) the output error [116]. Figure 3.4 provides a logical diagram of the NN training process in a supervised setting where the input data \mathbf{x} and their corresponding labels \mathbf{y} are available. In supervised learning, the loss function calculates the error between the network output $\hat{\mathbf{y}}$ and the ideal solution \mathbf{y} for the current set of parameters θ . The goal is to minimise a loss function indicated by $\mathcal{L}(\theta)$ or $\mathcal{J}(\theta)$. Loss is a non-negative value and, as it decreases, the performance of the model improves. An important aspect of supervised learning is the choice of an appropriate loss function for the task. For example, it is common to

use Mean Square Error (MSE) loss for regression tasks, and Cross Entropy (CE) loss for classification tasks.

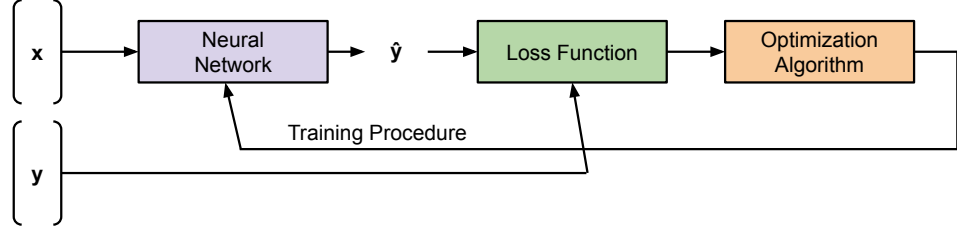


Figure 3.4: Training a Neural Network

Loss functions should be differentiable to facilitate learning through backpropagation. Backpropagation, short for “backward propagation of errors”, is an iterative algorithm that calculates the gradient of the error function $\nabla \mathcal{J}(\theta^{(t)})$ with respect to the NN parameters $\theta^{(t)}$ at iteration t . The “backward” part of the name is due to the fact that the gradient calculation proceeds backward through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last [116]. The implementation of the backward pass is based on the chain rule.

The parameters $\theta^{(t)}$ in iteration t are updated using an optimisation algorithm as seen in Algorithm 3.3.1. Gradient descent [117] aims to find the minimum of a function; in this case, the loss function. To find the minimum of a function using gradient descent, we take steps proportional to the negative gradient of the loss function at the current point $\Delta\theta^{(t)} \propto -\nabla \mathcal{J}(\theta^{(t)})$. First, in the forward pass phase, all input data \mathbf{X} is carried through the entire network to produce an output with the current parameters $\theta^{(t)}$. The error between the network output $\hat{\mathbf{y}}$ and the known target \mathbf{y} is calculated as $\mathcal{J}(\theta^{(t)}; x, y)$, and then the backpropagation estimates its gradients with respect to the model parameters $\nabla \mathcal{J}(\theta^{(t)})$. The parameters are then updated accordingly with $\theta^{(t+1)} = \theta^{(t)} + \eta \Delta\theta^{(t)}$ where η is the learning rate that controls the step size and is described in more detail in Section 3.3.5. The forward and backward passes process is repeated over multiple iterations T , known as epochs.

Algorithm 3.3.1 The batch gradient descent algorithm.

Input: initial weights $\theta^{(0)}$, number of epochs T **Output:** final weights $\theta^{(T)}$

1. select learning rate η
 2. **for** $t = 0$ **to** $T - 1$
 3. forward pass to compute $\mathcal{J}(\theta^{(t)}; x, y)$
 4. estimate $\nabla \mathcal{J}(\theta^{(t)})$ using chain rule
 5. estimate $\Delta \theta^{(t)} \propto -\nabla \mathcal{J}(\theta^{(t)})$
 6. update $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{J}(\theta^{(t)}) = \theta^{(t)} + \eta \Delta \theta^{(t)}$
 7. **return** $\theta^{(T)}$
-

3.3.2 Gradient Descent Variants

The Algorithm 3.3.1 is the Batch Gradient Descent (BGD) where all data are pushed through the network and gradients are used to backpropagate. The advantage of this approach is that an optimal (true) gradient can be computed and the error is relatively stable between iterations. On the other hand, the BGD is more computationally inefficient in cases where the training dataset is large. Large datasets typically do not fit in memory and require multiple loading of data from storage in batches before gradients can be accumulated to perform parameter updates.

Other Gradient Descent Variants [117] are available that mitigate the limitation of BGD and, in particular: Stochastic Gradient Descent (SGD) and Mini-BGD. In SGD data samples are presented one by one to the network. When each sample is pushed through the network, the gradients are computed and the network parameters are updated through backpropagation. Hence, SGD provides fast computation per pass and frequent updates at the expense of inconsistent navigation of the loss surface and pathway to the minimum. This, in turn, may lead to challenges in settling to the minimum, while frequent updates could result in more computation.

Mini-BGD, combines BGD and SGD where a portion (a mini-batch) of the data is pushed through the network and the average gradients from that portion are used for backpropagation. Mini-batch size is the number of samples used to compute gradients after which the parameter updates happen. The number of steps to complete one epoch is called iterations and is equal to the size of the training data divided by the mini-batch size [118]. Although Mini-BGD requires an additional hyperparameter (mini-batch size),

it accumulates the error within each mini-batch offering intermediate update frequency and high computational efficiency. These reasons make Mini-Batch Gradient Descent one of the most popular optimisation algorithms used in practise.

3.3.3 Momentum

Momentum [119] is an extension of the gradient descent that introduces inertia into the learning process leading to faster convergence. Furthermore, it reduces the high variance experienced by SGD by accelerating convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction [120]. The momentum terms modify the parameter updates:

$$\theta_{t+1} = \theta_t + u_{t+1} \tag{3.2}$$

$$u_{t+1} = \gamma u_t - \eta \nabla J(\theta_t) \tag{3.3}$$

Here γ is the momentum hyperparameter and u_t is the last update to θ . The momentum term γ is usually set to 0.9 or a similar value. Consideration must be given when selecting the momentum hyperparameter, since high values could lead to oscillations around the loss minimum.

3.3.4 Adam

Adaptive Moment Estimation (Adam) [121] is another extension that uses Momentum and Adaptive Learning Rates for each parameter update to increase convergence speed. The adaptive learning rate technique has also been utilised in other variants such as AdaGrad, AdaDelta, and RMSprop [122]. Part of the intuition for adaptive learning rate is starting with big steps and finishing with small steps. This allows for faster movement on the loss surface initially, and as the learning rate decays, smaller steps are taken, allowing for faster convergence and not surpassing the global minimum. The learning is changed dynamically using two additional terms, an exponentially decaying average of past gradients and an exponentially decaying average of past squared

gradients.

$$g_{t+1} = \nabla J(\theta_t) \quad (3.4)$$

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_{t+1} \quad (3.5)$$

$$u_{t+1} = \beta_2 m_t + (1 - \beta_2) (g_{t+1})^2 \quad (3.6)$$

Here m and u are estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients, respectively. As m_0 and u_0 are initialised as vectors of zeros, they are biased towards zero) [121]. The bias-corrected first and second moment estimates are:

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1)^t} \quad (3.7)$$

$$\hat{u}_t = \frac{u_t}{(1 - \beta_2)^t} \quad (3.8)$$

This leads to the optimisation step:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_{t+1}} + \epsilon} \hat{m}_{t+1} \quad (3.9)$$

The authors of Adam [121] propose default values of 0.9 for β_1 , 0.999 for β_2 and 10^{-8} for ϵ . Adam [121] is one of the most popular optimisers and is used to optimise the models in this thesis.

3.3.5 Learning Rate and Schedulers

The learning rate η is an important hyperparameter when setting up a NN. It defines the step size when updating network parameters and, consequently, directly affects convergence and its speed [123]. Low learning rates result in small step sizes and smooth convergence, but at the expense of convergence speed. Large learning rates, on the other hand, cause drastic updates on the parameters and undesirable divergent behaviour in the loss function. Traditionally, the learning rate is selected to be constant throughout the training; however, approaches where the learning rate varies through a

schedule or through learning conditions offer faster convergence and better approximate (reach) the location of the minimum.

The common learning rate scheduling approach is through decaying, where the learning rate is reduced by a predefined percentage after a set number of training epochs [124]. This approach is based on the intuition that larger steps are required at the beginning of the training process, while smaller steps to converge to the minimum are required towards the end of the training. However, this approach does not take into account the state of the learning process and the reduction is fixed at set times even if the training process is far from the minimum. An alternative approach that observes the performance metrics of the network is the ‘Reduce on Plateau’ scheduler, where the learning rate is reduced only when the performance metrics have stopped improving. In practise, consecutive epochs may momentarily result in constant metrics, and a second parameter called patience is used to decide when the learning process has remained static before proceeding to a learning rate reduction. Usually, the performance metric to use to drive the learning rate scheduler is the validation loss [125].

A frequently occurring issue during training is convergence to local minima. A technique that attempts to mitigate convergence to local minima through learning rate scheduling is the Cyclic Learning Rate (CLR) [126]. This scheduler sets the learning rate for each parameter group according to the CLR policy. The policy cycles the learning rate between two boundaries with a constant frequency. The distance between the two boundaries can be scaled on a per-iteration or per-epoch basis. The CLR policy changes the learning rate after every mini-batch.

One-cycle policy [127] is another scheduling technique that anneals the learning rate profile from an initial learning rate to some maximum learning rate and then from that maximum learning rate to some minimum learning rate much lower than the initial learning rate. The one-cycle policy annealing process has been shown to converge faster [127] and this scheduler is used for the training of the models in this thesis.

3.4 Convolutional Neural Networks

In the last decade, deep learning methods have been shown to outperform previous state-of-the-art ML techniques in several fields, with computer vision being one of the most prominent cases. CNNs and variations of them have achieved the best performance in tasks such as object detection, face recognition, action recognition, and human pose estimation [128]. Figure 3.5 shows the common architecture of a CNN model and its different layers, which are reviewed in this section.

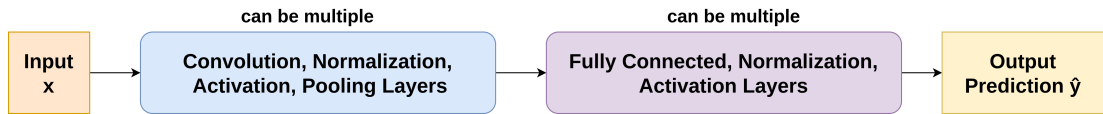


Figure 3.5: Basic CNN architecture

CNNs mainly consist of series of convolution and pooling (downsampling) layers and finally one or more fully connected layers. Subsequent architectures have replaced the fixed pooling layers by adjusting the stride of the convolutional layers, providing higher flexibility and better computational efficiency [129].

3.4.1 Convolution Layers

Convolutional layers are the staple layer for DL CNNs, primarily due to their ability to increase parameter sharing while maintaining the extraction of spatial feature maps. When dealing with high-dimensional input, such as images, it is impractical to use fully connected layers that connect neurons to all neurons in the previous layer.

Parameter sharing is the sharing of weights by all neurons in a particular feature map. This helps reduce the number of parameters in the whole system and makes the computation more efficient. If the detection of a horizontal edge is important at some location in the image, it should intuitively be useful at some other location in the image due to the translation-invariant structure of images. In convolutional layers, each neuron is connected to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron. The extent of connectivity along the depth axis (number of filters) is always equal to the depth

of the input volume. The connections are local in 2D space (along width and height), preserving spatial arrangement but extend to the full depth (i.e., all colour channels). These properties make CNNs suitable for handling natural images and hence suitable for the automatic annotation of subsea pipelines.

A convolution layer comprises a set of multiple independent filters or kernels. Each filter is independently convolved with the input by sliding the filter across the input extend (stride), and the layer output results in feature maps which are passed to through the activation function and then to subsequent layers. The convolution is defined as:

$$h_{xy} = \sum_i \sum_j w_{ij} u_{(x+i)(y+j)} \quad (3.10)$$

where h_{xy} is a feature map value in (x, y) , w_{ij} is a kernel (filter) weight and $u_{(x+i)(y+j)}$ is an input value at $(x+i, y+j)$. Figure 3.6 shows an example of a 2D convolution, where I is the input image, K is the kernel or filter, and the output $O = I * K$ is the feature map. Since multiple filters are used to convolve the input, an input image or feature map will become a stack of filtered images [130].

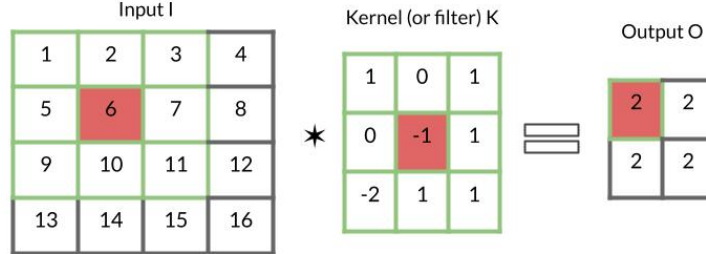


Figure 3.6: 2D Convolution

Transposed convolutions are used to upsample the input feature map to a desired output feature map. Similarly to the standard convolutional layer, the transposed convolutional layer is also defined by the padding and stride. Transposed convolutions are used in Encoder-Decoder architectures such as Unet [71] for image segmentation, in GAN [131] for image generation, and in image translation architectures [132].

3.4.2 Downsampling

As the network progresses, the inputs are downsampled in height and width dimensions, but their depth increases with the feature maps produced. One way to control the downsampling using the convolutional layers is by selecting the stride and padding hyperparameters. Stride denotes how many pixels the filter moves when convolved with the entire input. Naturally, the output size is smaller than that of the input. In the case where it is useful to maintain the dimensions of the input in the output, padding is used. Padding is a process of adding information (pixels) to the input [133]. This process is demonstrated in Figure 3.7 where a 3×3 kernel is convolved with the input. The downsampling operation can be tuned by changing the stride; bigger strides will lead to smaller outputs and vice versa.

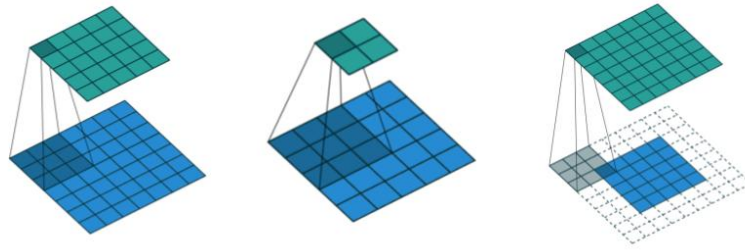


Figure 3.7: Convolution without stride and padding, convolution with stride of 2 and no padding, and convolution with padding of 2 and no stride [134]

Another downsampling option is the use of a Pooling layer which reduces the spatial dimensions of the input image to pass it to the next convolutional layer [128]. The pooling layer does not affect the depth dimension of the image. This operation reduces the size of the feature maps, which leads to a simultaneous loss of information. However, this loss acts as a regularisation to mitigate against overfitting (see Section 3.6) while at the same time leads to less computational overhead for the upcoming layers of the network. Average pooling and max pooling are the most commonly used strategies. The Max Pooling uses intensity values from the input to construct the shrunk image, and therefore, is used after convolutions. On the other hand, average pooling alters the intensity values and is used before the fully connected layers to obtain the final feature map. Figure 3.8 provides two examples of Max and Average Pooling.

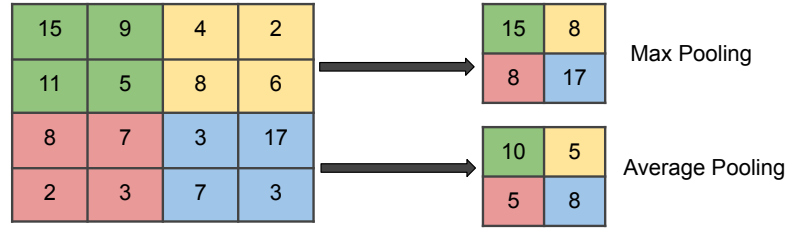


Figure 3.8: Max and Average Pooling

3.4.3 Activation Functions

The use of convolutional or fully connected layers are linear operations (matrix multiplication and summation). Typically an activation layer can be used after these layers to introduce non-linearity to a system [135, 136] and are added to help the NN learn complex patterns in the data.

Sigmoid

The Sigmoid function (Equation 3.11) is a logistic function, which means that it scales the input to an output ranging between 0 and 1. Hence, it always produces a non-negative value as output. Thus, it is not a zero-centred activation function which is desirable [137]. However, during backpropagation, the use of Sigmoid can lead to vanishing gradients if the weights are large because of its derivative (shown on the right-hand side of Figure 3.9). The vanishing gradient problem happens if the derivatives are small, then the gradient will decrease exponentially through backpropagation until it eventually vanishes. The accumulation of small gradients results in a system that is incapable of learning meaningful insights, as the weights and biases of the initial layers will not be updated effectively. When the gradient becomes nearly 0, the network stops training [138, 139]. This effect is more pronounced when multiple layers are cascaded, and there Sigmoid activations are avoided in DL models.

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \quad (3.11)$$

Typically Sigmoid activation functions is suitable after the last linear layer of a

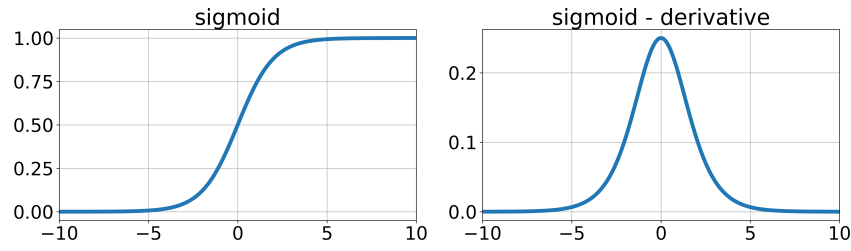


Figure 3.9: Sigmoid

classification network for multi-label or multi-class classification problems because it provides non-negative (0-1 output) and can be loosely interpreted as probabilities. Typically this is used for either binary classification models (with one neuron at the output) or multi-label classifiers. For multi-label a threshold can be defined to discretise the label predictions; the threshold can be either a single value for all labels or a multi-threshold (different value for each label) [140, 141].

ReLU and variants

Rectified Linear Unit (ReLU) activation function consists of two linear sections as shown in Figure 3.10 (however non-linear as a whole) with corresponding gradients of 0 or 1. The ReLU activation (Equation 3.12) allows the network to train faster (owing to computational efficiency) [142]. It also helps mitigate the problem of vanishing gradients. The ReLU layer applies the function of Figure 3.10 to all values in the input volume, changing all negative activations to 0. This layer increases the non-linear properties of the model and the overall network without affecting the receptive fields of the convolutional layer. ReLU has been one of the most widely used activation functions since it has been proposed [143].

$$ReLU(x) = \max(0, x) \quad (3.12)$$

Any input to the ReLU function that is less than zero generates zero as output. As a result, parts of the input that have negative weighted sums do not contribute to the whole process. This is also called the dying ReLU problem, and the neurons that

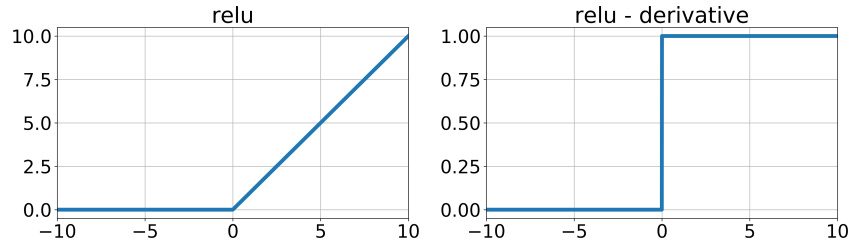


Figure 3.10: ReLU

are deactivated by ReLU are called dead neurons. The dying ReLU problem leads to a new variant of ReLU called Leaky ReLU or LReLU [144].

The Leaky ReLU (LReLU) function (Equation 3.13) is continuous and not bounded. It is computationally very cheap and is zero-centred activation function. The LReLU function allows for a small part of negative units instead of pushing them to zero, as does the ReLU.

$$LReLU(x) = \begin{cases} 0.01x & \text{for } x \leq 0 \\ x & \text{otherwise} \end{cases} \quad (3.13)$$

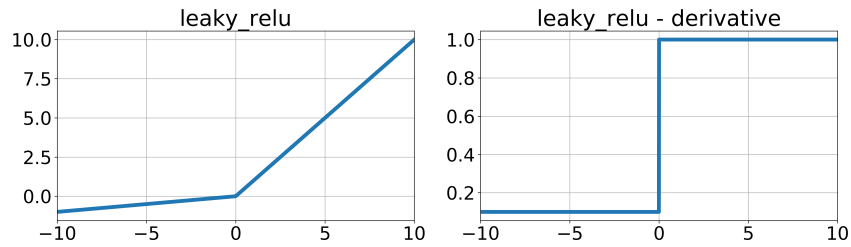


Figure 3.11: Leaky ReLU

Softmax

In multi-class classification Softmax activation (Equation 3.14) is used after the last linear layer to convert the logits/scores of every class to probability-like values that add up to 1. The Softmax output is large if the input score (called logit) is large. In the same way, its output is small if the score is small. The proportion is not uniform, as

Softmax is exponential. This means that it can increase the differences by pushing one result closer to 1 while pushing another closer to 0. In multi-class image classification tasks, after Softmax, the class with the highest score is chosen as the final prediction (*argmax*).

$$\text{Softmax}(x) = \frac{\exp(x)}{\sum \exp(x)} \quad (3.14)$$

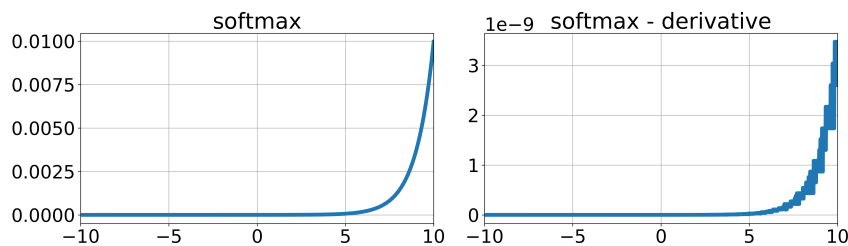


Figure 3.12: Softmax

3.4.4 Weight Initialisation

Since the weights of a model are updated during training, they must be assigned initial values before training begins. Weight initialisation is the assignment of initial values to the weights of a NN. The aim of weight initialisation is to prevent the layer activation outputs from exploding or vanishing during training a deep NN. Loosely speaking, if the weights and gradients are large, chained multiplications will dramatically increase the risk of overflow. Similarly, if the weights and gradients are small, the chained multiplications will lead to extremely small values (near zero), which cannot be represented using the finite resolution. Both of these cases will result in ineffective learning.

Traditionally, weight initialisation was performed using random small values. For instance [145], utilised a $\mathcal{U}(-\frac{2.4}{n_i}, \frac{2.4}{n_i})$; where n_i is the number of incoming network connections; however, subsequent studies have identified that this is an ineffective method to initialise NNs [146, 147].

Xavier [146] proposed a weight initialisation method that prevents gradients from vanishing for tanh and sigmoid activations where weights are selected from a range:

$$\mathcal{U}[(\frac{-\sqrt{6}}{\sqrt{n_i + n_j}}, \frac{\sqrt{6}}{\sqrt{n_i + n_j}})] \quad (3.15)$$

where n_i is the number of incoming network connections, n_j is the number of outgoing network connections from that layer n , and U is the uniform distribution. Subsequent studies by He [147] identified that Xavier initialisation does not work well with ReLU activation functions. Instead, He initialisation proposes that weights are selected from range:

$$\mathcal{N}[(\frac{-\sqrt{6}}{\sqrt{2 * n_i}}, \frac{\sqrt{6}}{\sqrt{2 * n_i}})] \quad (3.16)$$

where n_i is the number of incoming network connections in the layer n , and N is the normal distribution. In this thesis, when models are trained from scratch, the Kaiming He initialisation is used.

3.4.5 Normalisation

Before passing the input data to NN, the data need to be normalised, as normalisation-induced scaling and shifting are helpful for gradient-based learning. This is because equivalent updates are made to the network weights for all input dimensions, allowing a stable learning process [148] and faster convergence, as seen in Figure 3.13. If the input training vectors are not normalised, the ranges of feature values would likely be different for each feature, and thus weight updates will be dominated from the larger dimension. This leads to oscillating loss and hence moving slowly towards the minimum and increased convergence time. Therefore, the data should be normalised before being used as input into NN or any gradient-based algorithm.

Intermediate Normalisation

Normalising the input of the network is a well-established technique to improve the convergence properties of a network. Similarly, normalisation can be performed as intermediate layers of a NN. The activation layer has a dimension of $N \times C \times H \times W$ where N is the mini-batch size, C is the number of channels (filters) in that layer, H

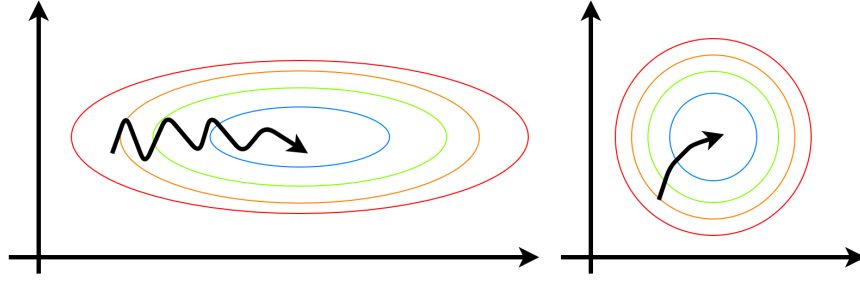


Figure 3.13: Unnormalised data lead to gradients of larger parameters dominating the updates, with normalisation parameters updated proportionally equal.

is the height of each activation map, and W is the width of each activation map. Normalisation techniques differ in the way μ and σ are calculated, as shown in Figure 3.14, with blue cubes used to calculate these statistics.

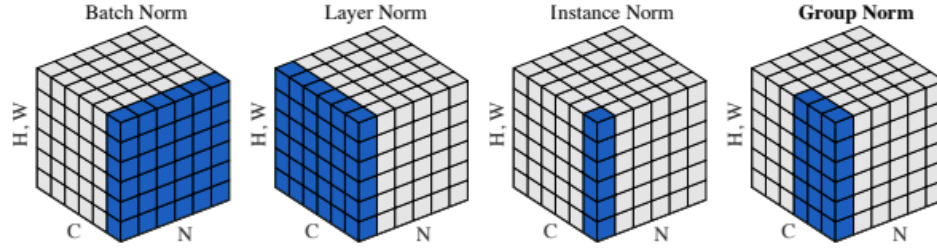


Figure 3.14: Normalisation techniques [149]

Batch Normalisation (BN) can be considered as preprocessing at each layer of the network. In BN, the mean (μ) and variance (σ) are calculated for each individual channel in all samples and in both spatial dimensions as shown in Equation 3.17 and Equation 3.18.

$$\mu_c = \frac{1}{NHW} \sum_{i=1}^N \sum_{j=1}^H \sum_{k=1}^W x_{ijk} \quad (3.17)$$

$$\sigma_c^2 = \frac{1}{NHW} \sum_{i=1}^N \sum_{j=1}^H \sum_{k=1}^W (x_{ijk} - \mu_c)^2 \quad (3.18)$$

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \quad (3.19)$$

However, normalisation of activations (Equation 3.19) is not sufficient, because it can alter activations and disrupt useful patterns that the network learns. Therefore, normalised activations are scaled and shifted to allow them to learn useful discriminative representations, as seen in Equation 3.20.

$$y = \gamma \hat{x} + \beta \quad (3.20)$$

where γ and β are the learnable parameters that are updated during training. In BN, during inference, the μ and σ of the whole population are used for normalisation instead of batch statistics.

BN restricts the distribution of output activations that help the network produce better gradients for weight updates, and therefore it enables a higher learning rate and faster convergence by reducing the internal covariate shift that occurs during training [150]. The covariate shift refers to the change in the distribution of layer activations as the parameters are updated during training. When the distribution changes, the layers try to adapt to the new distribution, which slows the training process and results in an increase in the convergence time. Furthermore, BN allows each layer of a network to learn independently of other layers, thus reducing overfitting. Similarly to Dropout (see Section 3.7.2), it adds some noise to the activations of each hidden layer. Therefore, with added BN, less Dropout can be used. However, it is common to use BN together with Dropout for regularisation.

In Instance Normalisation (IN), the mean and variance for each individual channel are calculated for each individual sample in both spatial dimensions [151]. Thus, IN allows filtering out instance-specific contrast information from the content. Unlike BN, the IN layer is applied at test time as well (due to the non-dependence of mini-batch). Motivation of IN was investigated in stylization literature [151] and was used to disentangle the contrast of the input image with the generated stylised image. The μ and σ in IN are calculated as:

$$\mu_{nc} = \frac{1}{HW} \sum_{j=1}^H \sum_{k=1}^W x_{jk} \quad (3.21)$$

$$\sigma_{nc}^2 = \frac{1}{HW} \sum_{j=1}^H \sum_{k=1}^W (x_{jk} - \mu_{nc})^2 \quad (3.22)$$

In Layer Normalization (LN), mean and variance are calculated for each individual sample in all channels and in both spatial dimensions [152] and are defined as:

$$\mu_n = \frac{1}{CHW} \sum_{i=1}^C \sum_{j=1}^H \sum_{k=1}^W x_{ijk} \quad (3.23)$$

$$\sigma_n^2 = \frac{1}{CHW} \sum_{i=1}^C \sum_{j=1}^H \sum_{k=1}^W (x_{ijk} - \mu_n)^2 \quad (3.24)$$

Similarly to LN, Group Normalisation (GN) is also applied along the direction of the features, but unlike LN, it divides the features into certain groups and normalises each group separately [149]. GN can exploit the channel dependence by dividing channels into groups (see Figure 3.14). Since each group of channels (instead of all of them) is assumed to have a shared mean and variance, the model has the flexibility of learning a different distribution for each group.

3.5 Recurrent Neural Networks

Another type of NN that is used for sequential data (text, time series, audio, video, etc.) is called a Recurrent Neural Network (RNN). A RNN can be thought of as multiple copies of the same network, each passing a message to the next, as seen in Figure 3.15 that demonstrates the unrolling of the RNN loop. In the figure below, a module of a NN A , takes an input x_t and outputs a value h_t , the loop allows information to be passed from one step of the network to the next.

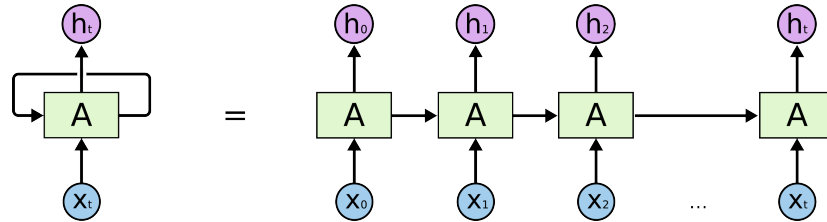


Figure 3.15: An unrolled recurrent neural network [153]

RNNs are capable of selectively processing and retaining sequential information, and in recent years, there has been a wide use of applying RNNs to problems and tasks that involve sequential data such as speech recognition, language modelling [154] and translation [155], video recognition, and image captioning [156]. However, RNNs cannot learn long-term dependencies because of the vanishing gradient problem, when the interval between the relevant input data points becomes too large, the gradient decreases, and the updates of the earlier layers do not contribute to learning.

To overcome this issue, two versions of RNN were created, namely Gated Recurrent Unit (GRU) and LSTM. LSTM was designed to overcome this problem by learning to select which information is relevant to remember [157]. Since its inception, LSTM has been modified to have internal mechanisms, called gates, that can regulate the flow of information. A schematic overview of a LSTM and its gates is shown in Figure 3.16 and can be described by the following equations 3.25, 3.26, 3.27, 3.28, 3.29.

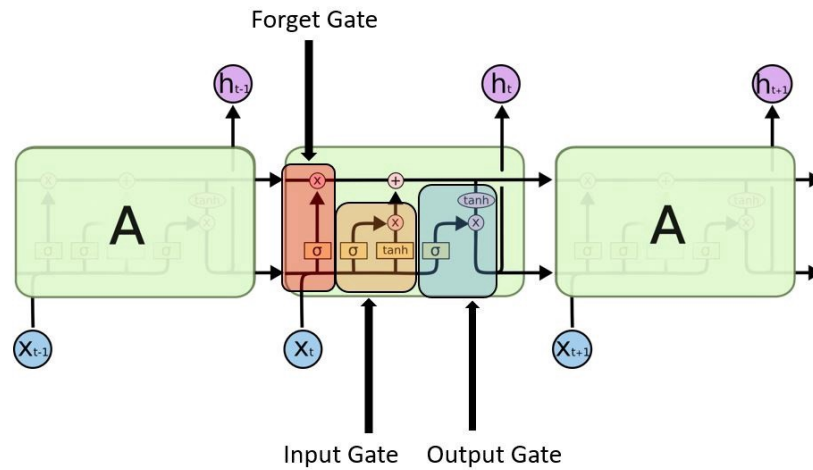


Figure 3.16: LSTM gates [153]

The forget gate discovers which details will be discarded from the block using the Sigmoid function. Using the previous state h_{t-1} and the input x_t , it outputs a number between 0 and 1 for each number in the cell state C_{t-1} . Values close to 1 indicate that the corresponding element in the previous cell state should be retained in the current

cell state, while values close to 0 indicate that the corresponding element should be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \quad (3.25)$$

The input gate discovers which input value should be used to modify the memory. The Sigmoid function decides which values to allow and the tanh function gives weight to the values passed, deciding their level of importance ranging from -1 to 1 .

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \quad (3.26)$$

$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t]) \quad (3.27)$$

The output gate uses the input and the memory of the block to decide the output. The function tanh gives weight to the values passed, deciding their level of importance that ranges from -1 to 1 and is multiplied by the Sigmoid output.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \quad (3.28)$$

$$h_t = o_t * \tanh(C_t) \quad (3.29)$$

In the above equations, σ and \tanh are applied element-wise. The input at time point t is x_t as input sequence data at time point t , and h_{t-1} the previous hidden state. The current cell state, C_t , is modified by contributions from the current input and previous hidden state through the input gate, forget gate, output gate and cell updates, denoted i_t , f_t , o_t and C_t , respectively. Having these gates, allows LSTM models to selectively keep or forget information about previous inputs, which makes them well-suited for tasks that require remembering long-term dependencies.

A GRU [158] is similar to a LSTM, but has only two gates; a reset gate and an update gate, and it notably lacks an output gate, as seen in Figure 3.17. Fewer parameters mean that GRUs are generally faster to train than their LSTM counterparts.

However, in this thesis, LSTM is chosen and used in Chapter 5 to model video data, due to its wider availability in DL libraries.

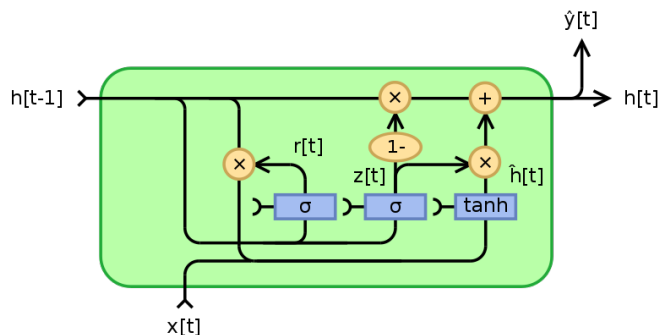


Figure 3.17: GRU gates [159]

3.6 Overfitting and Underfitting

Supervised ML is best understood as the approximation of a target function f that maps the input variables x to an output variable y . This characterisation describes the range of classification and prediction problems and the ML algorithms that can be used to address them. An important consideration in learning the target function from the training data is how well the model generalises to new unseen data. Overfitting is the situation where a model learns well the distribution of the training data but cannot generalise on unseen data. Model underfitting refers to a model that is too simple to capture the relationship in the data in both seen and unseen data. An underfit ML model can be easily detected, as it will have poor performance on the training data. The regression example in Figure 3.18 demonstrates the three cases (underfit, good fit, overfit) of trying to approximate a part of a cosine function with a polynomial of different degrees and MSE is used as the loss function for this task. On the left-hand side of Figure 3.18 it is shown that a linear function (polynomial of degree 1) is not sufficient to fit the training samples while also producing a high MSE for both the training and the testing data points. On the right-hand side, the 16th degree polynomial model overfits the training data, i.e. it learns the noise of the training data (very small training MSE) and does not generalise on unseen data (very

large testing MSE). A polynomial of degree 4 approximates the true function almost perfectly and produces similar errors for both the training and the testing data points. Scikit-learn [160] Python library is used for this example.

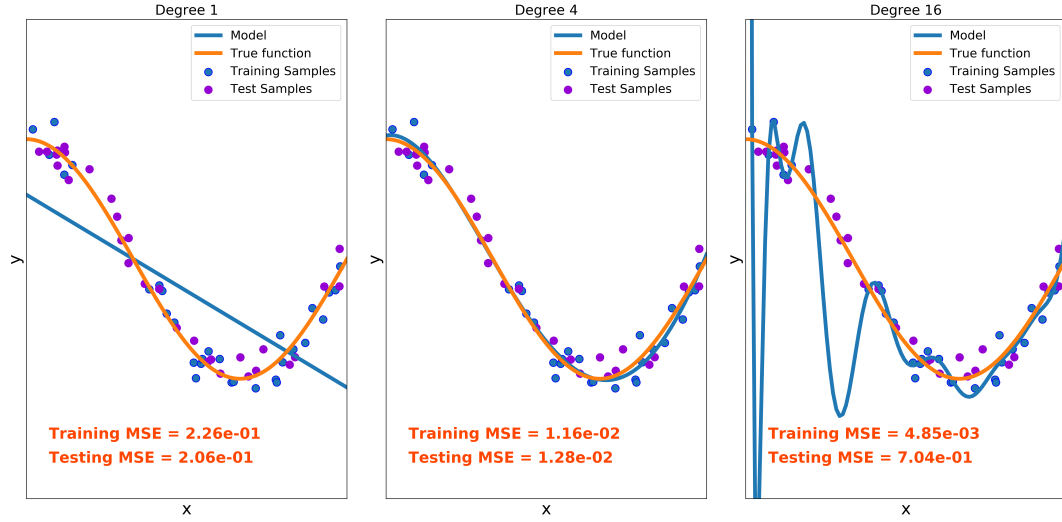


Figure 3.18: Underfitting, Fitting, and Overfitting

3.6.1 Bias-Variance Tradeoff

Bias occurs when an algorithm has limited flexibility to learn the true signal from a dataset. In Figure 3.18 the 1st degree model has a high bias, while the 16th degree model has a high variance. Variance refers to an algorithm's sensitivity to specific sets of training data. The prediction error for any ML algorithm can be broken down into three parts, namely bias error, variance error, and irreducible error.

$$y = f(x) + e \quad (3.30)$$

$$Error(x) = E[y - \hat{f}(x)]^2 \quad (3.31)$$

$$Error(x) = (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \sigma_e^2 \quad (3.32)$$

$$Error(x) = Bias^2 + Variance + Irreducible \quad (3.33)$$

When a model is too simple and has very few parameters, it may have a high bias

and low variance. On the other hand, when a model has a large number of parameters, then it is going to have a high variance and low bias. Therefore, it is desired to find the right balance without overfitting and underfitting the data. This trade-off in complexity results in a trade-off between bias and variance. An algorithm cannot be more complex and less complex at the same time. In Figure 3.19 the optimal point is where the bias and variance errors meet, the left side is the underfitting area, while the right side is the overfitting area.

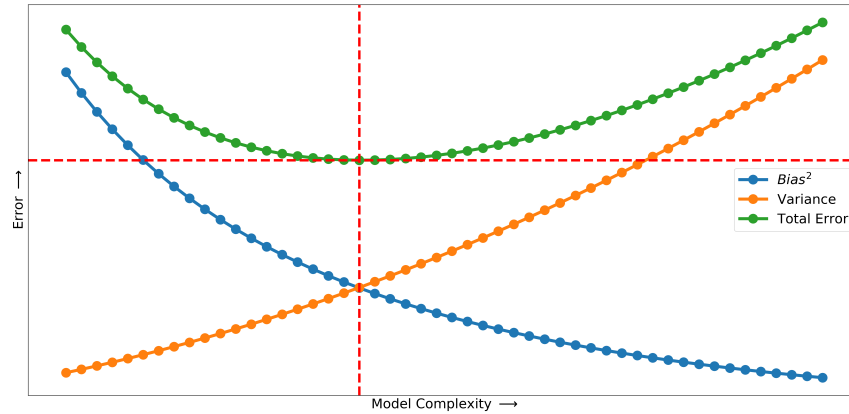


Figure 3.19: Bias-Variance Tradeoff

3.7 Regularisation

To tackle overfitting, models should generalise over the training data, and various regularisation techniques can be used. These techniques, which are used to reduce the error on the test set at the expense of an increase in the training error, are collectively known as regularisation [161]. This regularisation is often done by adding some extra constraints on a ML model, such as adding restrictions on the parameter values or by adding extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values. An effective regularizer optimised the trade-off by reducing variance significantly, while not overly increasing bias. Some techniques of regularisation are Weight Decay, Dropout, Data Augmentation, Label Smoothing, Early Stopping [162] and Data Augmentation.

3.7.1 Weight Decay

In regularisation, a new term is added to the loss function that penalises large weight values, which means that, in addition to being penalised for incorrect predictions, the model will also be penalised for having large weight values even if the predictions are correct. Shrinking the weights close to zero has the practical effect of effectively deactivating some of them. Therefore, Weight Decay ensures that the weights stay small and thus generalise better to new data. This regularisation term is often implemented as an $L1$ or $L2$ loss [163]. Hence, the loss function is modified to:

$$J'(\theta) = J(\theta) + \lambda\Omega(\theta) \quad (3.34)$$

$\Omega(\theta)$ is the regularisation term that is controlled by the regularisation coefficient λ . The effect of λ is shown in Figure 3.20, where the 16^{th} degree polynomial model is trained with $L1$ regularisation with three different values of λ . The model on the right-hand side has the lowest test error.

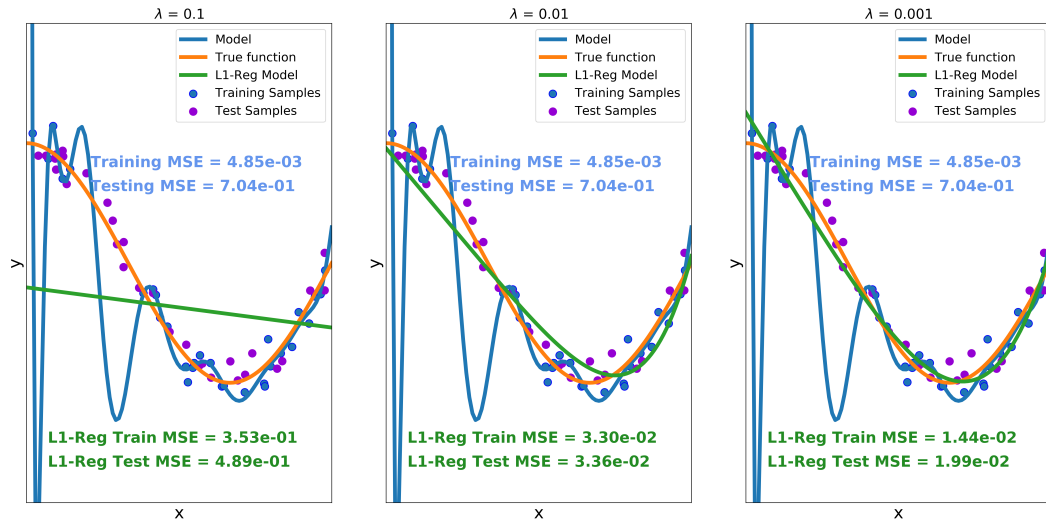


Figure 3.20: L1 Weight Decay regularisation with three different values of λ for fitting a 16^{th} degree polynomial model to approximate a part of a cosine function

3.7.2 Dropout

The Dropout layer, ‘drops out’ a random set of activations in that layer by setting them to zero. It encourages the network to be able to provide the right classification or output for a specific example, even if some of the activations are missing (dropped out), and this ensures that the network is not too ‘fitted’ to the training data. An important note is that this layer is only used during training and not during test time [164]. A common way to apply dropout to a NN is to deactivate a randomly selected hidden node and a randomly selected input node for each mini-batch of data; an example is shown in Figure 3.21. This is in practise implemented by drawing a random number from a uniform distribution and deactivating the neuron based on a threshold which is known as dropout probability. The dropout probability is an additional hyperparameter which must be considered during training.

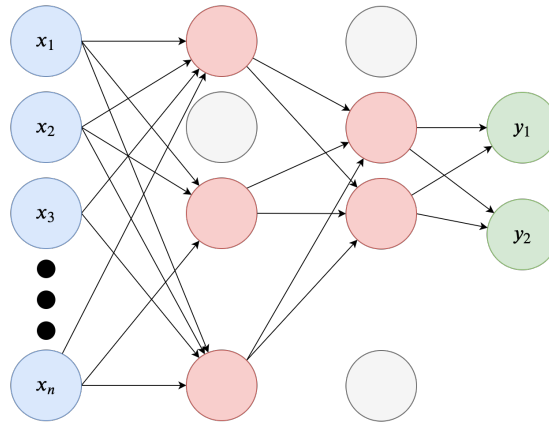


Figure 3.21: Dropout

3.7.3 Label Smoothing

In classification problems, sometimes models learn to predict training examples extremely confidently, which can result in poor generalisation. Label Smoothing is a regularisation technique that is used to prevent the model from predicting labels too confidently during training. It is also used to mitigate against incorrect assignment of labels in the dataset [165,166]. This is achieved by replacing the one-hot encoded label vector y_{hot} with a mixture of y_{hot} and the uniform distribution:

$$y_{ls} = (1 - a)y_{hot} + \frac{a}{K} \quad (3.35)$$

where K is the number of labels and a is a hyperparameter that determines the amount of smoothing. When $a = 0$, the original one-hot encoded y_{hot} is obtained. On the other hand, when $a = 1$, the uniform distribution is obtained.

3.7.4 Early Stopping

The core concept of early stopping is to stop the training phase once the performance in the validation set stops improving or becomes saturated [162]. The early stopping approach is demonstrated in Figure 3.22. Initially, both training and validation losses are decreasing, resulting in improvements in metrics of interest, such as accuracy or F1-Score. As training progresses further, the training loss keeps decreasing, but the validation loss starts to increase. The aim of early stopping is to detect the optimal point to cease training, and this uses the validation loss curve as the early stopping trigger. Naturally, there will be some staggering at various stages of the training process, and there may be temporal fluctuations in the validation loss. To mitigate from premature stopping while convergence is still on-going an additional parameter named ‘patience’ is used, which defines the number of epochs that the training continues before ceasing it. Therefore, early stopping can be used against overfitting, but can also provide additional benefits, such as shortening the validation cycle for hyperparameter tuning [167, 168].

3.7.5 Image Augmentation

More specifically for CNNs, another way to regularise them is by using image augmentation; by modifying the images in the dataset. Augmentation is widely used in computer vision tasks to increase the size of the dataset, to diversify samples, and thus make the model more robust and improve its generalisation capabilities [170]. Augmentation can be performed offline (creating more samples before training a model) or online during the training process. Furthermore, it plays a significant role when combined with techniques of oversampling to balance a long-tail dataset and thus mit-

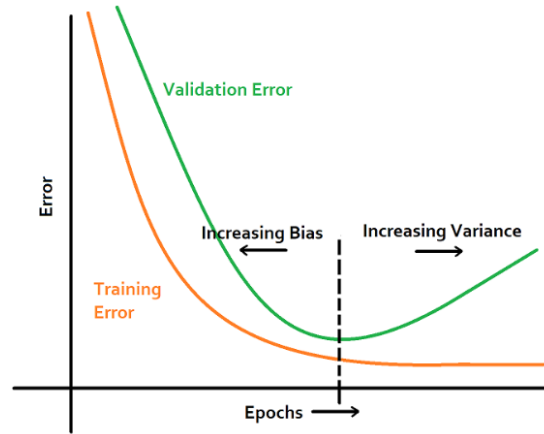


Figure 3.22: Early Stopping [169]

igate the problem of limited data and class imbalance. Augmentation techniques can be divided into subcategories, as seen in Figure 3.23. Image Processing based augmented examples of the Technology and Innovation Centre building of the University of Strathclyde are shown in Figure 3.24 and Figure 3.25 which have been created using the Albumentations library [171].

Figure 3.24 shows, Texture Augmentation techniques, also called style or pixel-level augmentation, can generally be viewed as colour distortion. More specifically, Blurring, Channel Shuffling, Colour Jittering, Noise, Hue Saturation are some of the texture (style) augmentations. Figure 3.25 shows Structure Augmentation which is also known as Content or Spatial-level Augmentation. It consists of Rotations, Flips, Crops, Scaling, Elastic Transformations, and Grid Distortions. In the case of subsea pipeline surveys, augmentation becomes essential to address the challenges of the subsea dynamic environment.

Another technique that can be used of image augmentation is called Neural Style Transfer (NST). In recent years, many NST works have been presented, where the style of a single image can be transferred to another image, as seen in Figure 3.26. One of the most popular works in NST is by Johnson *et al.* [172], they optimise a generator (Image Transform Net in Figure 3.26) by minimising both its content distance with the content image and its style distance with the style image using representations from different layers of the VGG [50] (externally pre-trained on ImageNet [173]). Content loss is the

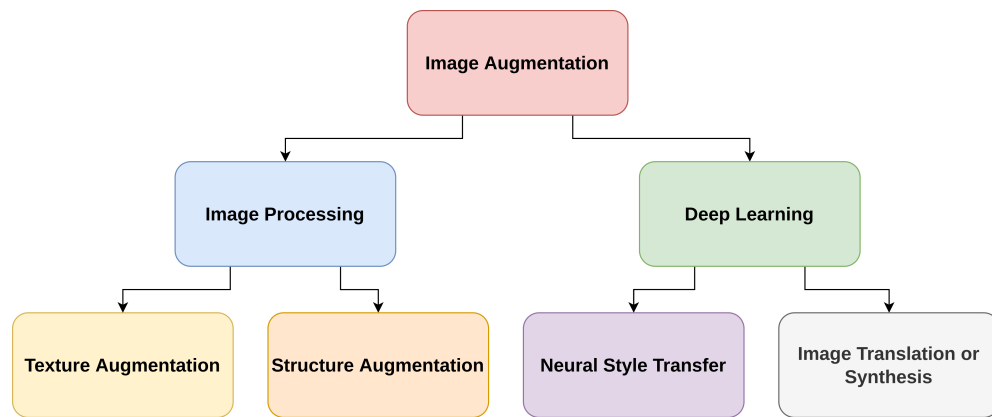


Figure 3.23: Taxonomy of Image Augmentation Techniques



Figure 3.24: Examples of Texture Augmentation



Figure 3.25: Examples of Structure Augmentation

Euclidean distance between the feature representations of the content image and the generated one. Style loss is the Frobenius distance between the Gram matrices of the generated and the style image representations. DL solutions for Image Translation and Synthesis are described in Chapter 6.

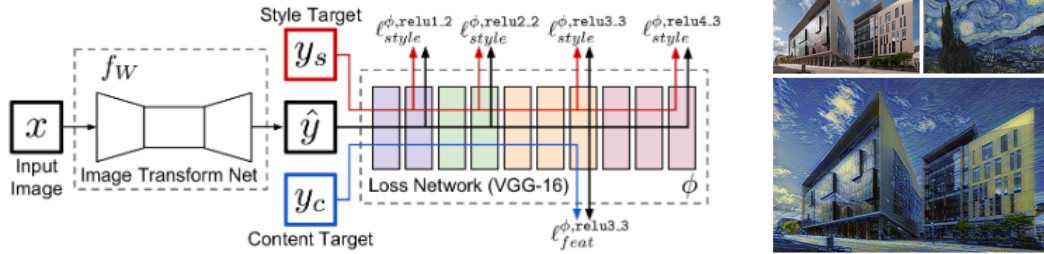


Figure 3.26: Neural Style Transfer using Johnson [172] Architecture. Here, the bottom image consists of \hat{y} , the top left image is both x and y_c , while the bottom right image is the y_s .

3.8 Deep Convolutional Neural Networks

CNNs have achieved outstanding results in image recognition. This section provides a review of several deep CNN architectures for image classification. Figure 3.27 shows the evolution of deep CNNs in both accuracy and model complexity in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [174].

3.8.1 AlexNet

AlexNet [176] was the first large-scale CNN model that led to the resurgence of deep CNNs in computer vision. This architecture won ILSVRC [174] in 2012 with a top-5 error of 15.3% by a large margin of 10.8% compared to the runner up and 9.6% from the winner of 2011. The main difference between the AlexNet architecture and its predecessors is the increased network depth, which leads to a significantly larger number of parameters. It consists of a total of eight parameter layers, among which the five initial layers are convolutional layers, while the later three layers are fully connected layers. The final fully connected layer classifies an input image into one of the thousand classes of the ImageNet dataset, and therefore contains 1,000 units. The

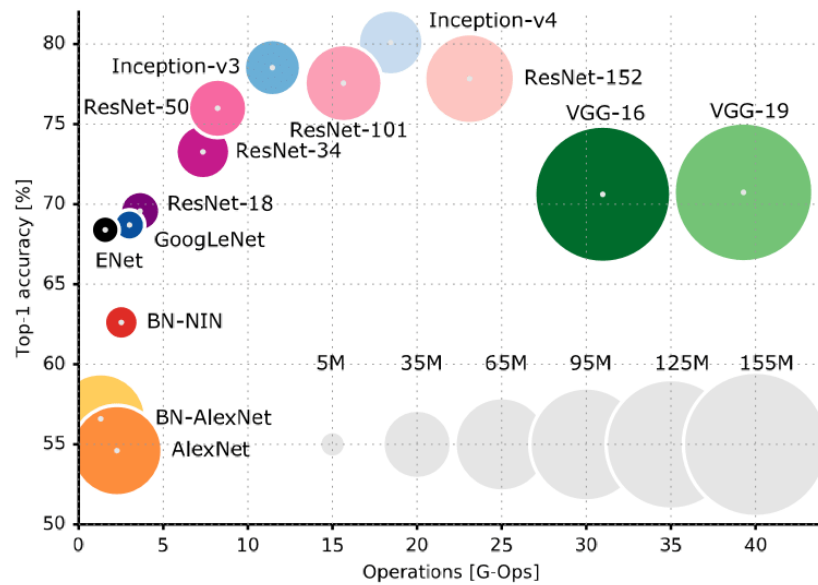


Figure 3.27: Comparison of deep CNNs on ImageNet (top-1 accuracy) [175]

filter sizes and the location of the max pooling layers are shown in Figure 3.28. Note that dropout is applied after the first two fully connected layers in the AlexNet architecture. Another distinguishing aspect of AlexNet is the use of the ReLU activation function after every convolutional and fully connected layer, which substantially improves the training efficiency compared to the saturating activation functions previously used, such as *tanh*.

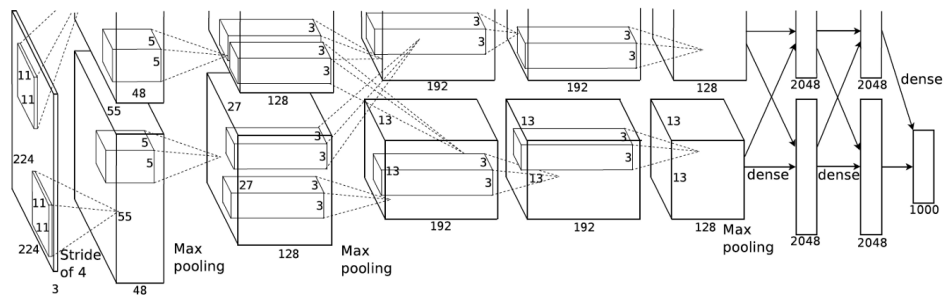


Figure 3.28: AlexNet [176]

3.8.2 VGG

In 2015 another more efficient and accurate network was implemented, called VGG [50]. The architecture of VGG consists of 5 convolutional blocks with max pooling in between and 3 full connected layers as a classifier head. Although VGG is based on AlexNet, it has several differences. Instead of using large receptive fields as in AlexNet (11x11 with a stride of 4), VGG uses very small receptive fields (3x3 with a stride of 1). The small-size convolution filters allow VGG to have a large number of layers and this leads to improved performance because deeper networks can learn more features. Similarly to AlexNet, it also uses activation dropouts in the first two fully connected layers to avoid overfitting. VGG16, where 16 is the number of layers, is also widely used in style transfer tasks as a feature extractor [172, 177].

3.8.3 GoogleNet / Inceptionv1

All the networks previously discussed consists of a sequential architecture with only a single path. Along this path, different types of layers, such as convolution, pooling, ReLU, dropout, and fully connected layers, are stacked on top of each other to create an architecture of desired depth. GoogleNet [178] is the first popular model that uses a more complex architecture with several network branches.

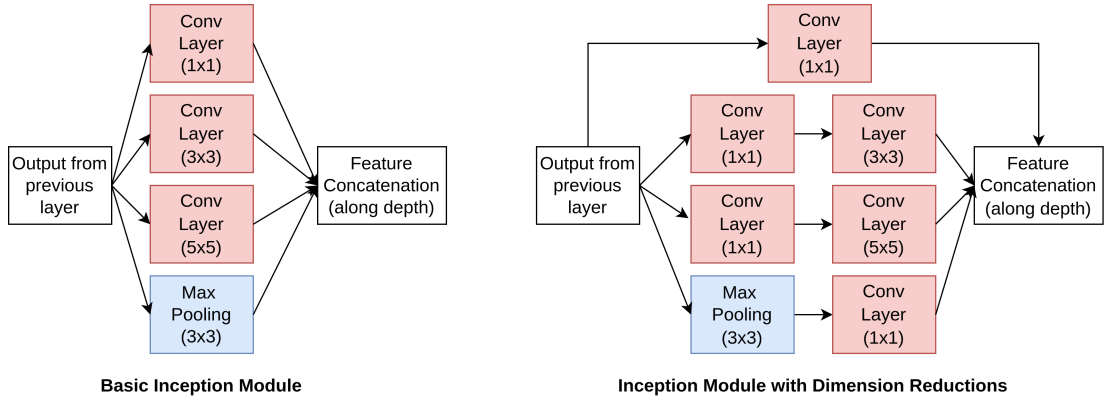


Figure 3.29: Inception Modules

GoogleNet consists of a total of 22 weight layers. The basic building block of the network is the ‘Inception Module’, shown in Figure 3.29 due to which the architecture

is also commonly called the ‘Inception Network’. Processing of this module occurs in parallel. All basic processing blocks that occur in a regular sequential convolutional network are placed in parallel, and their output feature representations are combined. The design of using multiple inception modules stacked together allows for the creation of a deep network without the need for carefully designing each individual layer for different stages of the network.

However, if all individual feature representations are concatenated from each individual block along the depth dimension, it will result in a very high-dimensional feature output. To overcome this problem, the full inception module performs dimensionality reduction before passing the input feature volume (say with dimensions $h \times w \times d$) through the 3×3 and 5×5 convolution filters. This dimensionality reduction is performed using a fully connected layer that is equivalent to a 1×1 convolution operation (right-hand side of Figure 3.29). Although convolution filters operate in the spatial domain (i.e., along the height and width of the input feature channels), a fully connected layer can combine information from multiple feature channels (i.e., along the depth dimension), and this leads to reduced feature dimensions.

The advantage of GoogleNet (Inception) is that the features are extracted using a range of filter sizes that correspond to different receptive fields and an encoding of features at multiple levels from the input. Similarly, there is a Max Pooling layer which down-samples the input to obtain a feature representation. All convolution layers in GoogleNet are followed by a ReLU non-linearity. In the end, these different complementary features are combined to obtain a more useful feature representation. Inceptionv3 [179] is also used as a feature extractor for the calculation of the FID score to assess dataset distribution shifts and discussed in more detail in Chapter 6.

3.8.4 ResNet

As the networks go deeper, their performance becomes saturated due to problems with vanishing or exploding gradients. ResNet [180] made it easier to train very deep networks by introducing an identity shortcut connection, as shown in Figure 3.30. These connections skip one or more layers, so their outputs are added to the outputs of the

stacked layers. In this way, the transformation function in a residual block is split into an identity term (which represents the input) and a residual term, which helps to focus on the transformation of the residue feature maps. In practise, such an architecture achieves stable learning of very deep models because the residual connections allow the model to learn an identity function, which ensures that the higher layer performs at least as well as the lower layers and not worse, while creating an alternate path through which the gradient flows, which mitigates the problem of the vanishing gradient [180]. A diagram of the entire ResNet-50 architecture used in this thesis can be seen in Figure 3.39 from Section 3.12. The weight layers in the residual block are followed by a BN and a ReLU activation layer. In this design, the identity mapping has to pass through the ReLU activation after addition with the output of the weight layers.

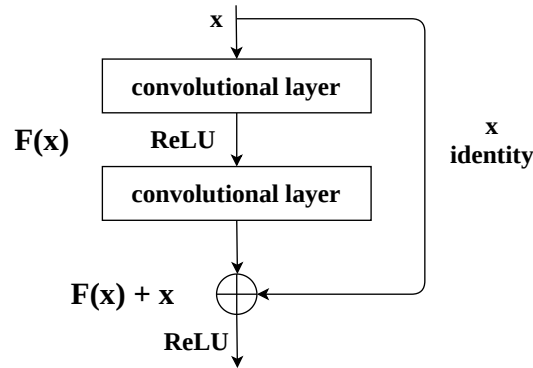


Figure 3.30: Residual connection

ResNet is one of the most popular state-of-the-art architectures that is used as an encoder for a wide range of applications (image segmentation, object detection, and self-supervised learning), including being a backbone network for the implementation of U-Net [71], RetinaNet [83], Faster R-CNN [77], Mask R-CNN [90] as well as SimCLR [181]. Furthermore, there are variations of ResNet, such as W-ResNet [182] and ResNeXt [183].

The reasons for choosing ResNet architectures for this work are listed below:

- Skip connections that allow Residual architectures to have a stable training, mitigating against vanishing gradients and making optimisation faster.

- ResNet have been successfully applied to a wide range of computer vision problems and have achieved state-of-the-art performance in several benchmarks [184]. In addition, residual connections allow the network to capture both low-level and high-level features, making ResNet models capable of learning hierarchical representations that generalise well to different datasets.
- Its wide availability of pre-trained models, with different depths (ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152) compared to similar architectures with fixed sizes like Inception.
- ResNet popularity in the DL community as shown in Figure 3.31.

Usage Over Time

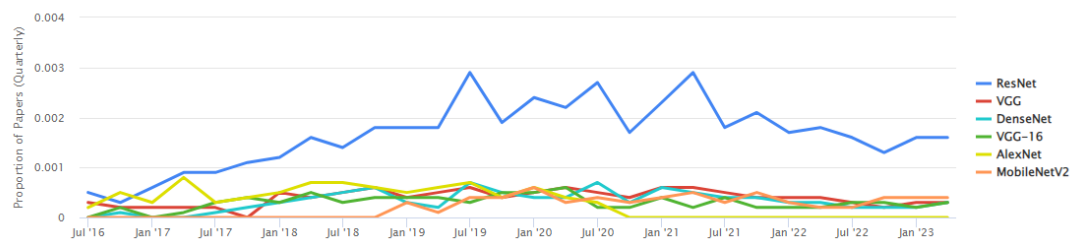


Figure 3.31: Papers using ResNet [184]

3.9 Transfer Learning

Deep CNNs, like ResNet, automatically extract features from domain-specific images, without any feature engineering techniques. Multiple layers work together to build an improved feature space. The initial layers learn first-order features (e.g. colour, edges, etc.), whereas the later layers learn higher-order features (specific to input dataset). In principle, a deep CNN can be initialised with random weights as described in Section 3.4.4. An alternative approach is to use weights from a pre-trained network on ImageNet [185] which is known as transfer learning. This latter approach is advantageous for two main reasons: (i) faster convergence; the pre-trained network weights are not random, and the weight values are already capable of extracting image-related

features (ii) weights values are obtained from a large data (larger than the target domain dataset) and hence feature maps are not overfitting to the dataset improving generalisation.

The transfer learning strategy is to take a CNN pre-trained on ImageNet [185], replace the last fully connected layer with a new classifier that has a number of output units corresponding to the classes of a particular task, compared to the 1,000 classes of ImageNet. During training, the weights of the convolutional layers (backbone) are kept fixed (frozen), and the gradients update only the classifier layers (head of the model). After initial training of the head, an option is to fine-tune the weights of the pre-trained network by continuing backpropagation to the backbone.

It is possible to train all layers in a single phase or to keep some of the earlier layers fixed and only fine-tune later layers in the backbone network [186]. Typically, the freezing of the backbone to train the head followed by staged unfreezing of the backbone is implemented, and the rationale and intuition are as follows: initially, when the new head is initialised with random weights, the errors and hence gradient will be large, which will lead to large weight updates of the backbone layers, detuning the network. However, training for a few epochs with the backbone frozen permits the head to converge to reasonable locations, and after unfreezing the backbone can be fine-tuned to improve performance further. In this thesis, both these approaches have been tried and it was found that a single stage training with all layers unfrozen works equally well.

In Discriminative Fine-tuning [186] different learning rates are used in different parts of the model, with lower learning rates in the earlier convolutional layers and higher in the latter. This technique is based on the fact that the earlier layers of the network learn general features, so the weights should not change much compared to the later layers, which learn more specific features of the task and need to be changed (updated) more. Additionally, the final layers (classifier) of the network must be trained at a higher learning rate, since the classification is different for a new task. Choosing a strategy depends on the size of the new dataset and its similarity [187] to ImageNet. However, starting with optimised ImageNet weights accelerates the convergence and

thus speeds up training.

3.10 CNN Visualisation

Visualisation of feature activation maps provides insight on regions that the model is focusing on to confirm that meaningful features are learnt. CNN Visualisation consists of visualising the convolution kernels, the feature or activation maps, and the final class activation maps [188]. Each convolutional layer of a CNN model contains different sets of filters. In Figure 3.32 the 64 3×3 kernels of the first convolutional layer of the VGG12 model are demonstrated. The model is pre-trained on ImageNet dataset. The initial layers capture more generic features, whereas the later layers capture more dataset-specific features.

The idea of visualising a feature map for a specific input image would be to understand what features of the input are detected or preserved in the feature maps. The expectation would be that the feature maps detect small or fine-grained detail. Feature maps are the result of applying filters to the input images. The output of the feature map from the prior layers could provide insight into the internal representation the model has of a specific input at a given point in the model. Figure 3.33 shows the first 16 activations after the first, middle, and last convolutional layer of the pre-trained VGG12 with the input image of Figure 3.33.

A deep CNN model consists of numerous convolutional layers and global average pooling is performed before the final output layer. To obtain the desired result, the resulting features are fed into a fully connected layer for the final prediction. By projecting the output layer weights back into the convolutional maps derived from the last Convolution Layer, the importance of the image regions is identifiable. This technique is called Class Activation Mapping. A CAM is the average of the feature maps produced after each convolutional layer and indicates the strongest features of the different labels or classes, as seen in Figure 3.34. CAM can be used to interpret the prediction decision made by CNNs [189, 190] and, therefore, can be used to help explain and debug the model. Other visualisation techniques have been implemented in [191–198]. Figure 3.35 shows the CAM of the bee image that a VGG12 (pre-trained

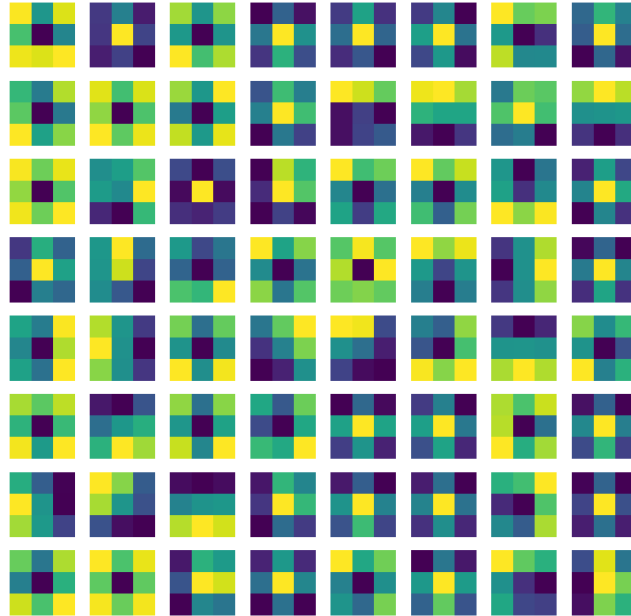


Figure 3.32: The 64 3x3 convolutional kernels of the first VGG12 convolutional layer

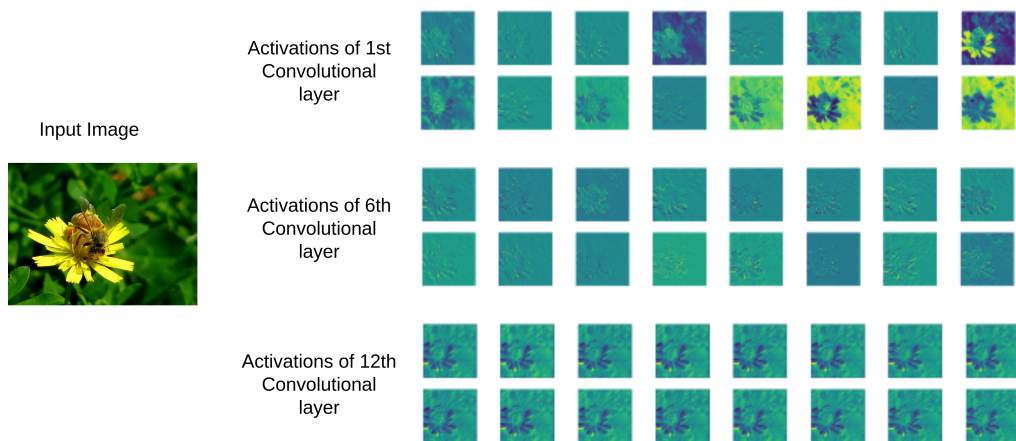


Figure 3.33: Input bee image and activations of the 1st, 6th and 12th convolutional layers of VGG12

on ImageNet) produces.

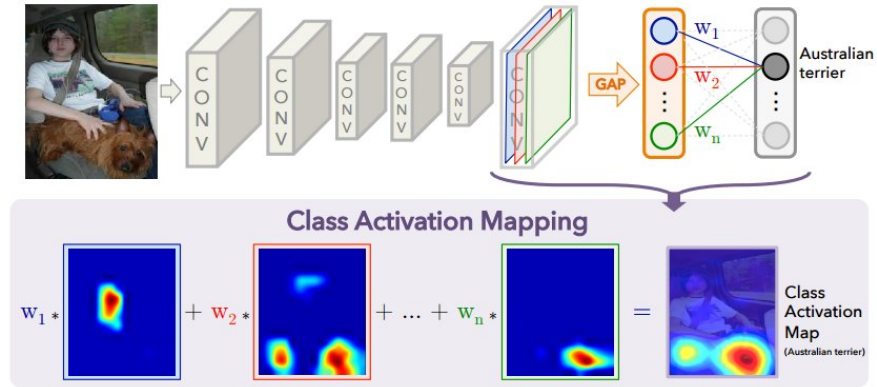


Figure 3.34: Class Activation Mapping [190]



Figure 3.35: Class Activation Map

3.11 Hyperparameter Tuning

A DL model can have multiple hyperparameters that must be configured to control the model complexity, its performance and convergence. The hyperparameters can be logically divided into two groups: (a) architectural hyperparameters and (b) training hyperparameters. The exploration of the first group of hyperparameters considers the engineering of network architecture (such as number of layers, depth, and width) with multiple formal approaches which are collectively referred to as Neural Architecture Search (NAS) [199]. In this work NAS has not been used because the models employed in the thesis, rely on established transfer learning architectures and the use of NAS

will negate the benefits provided by transfer learning in addition to the significant computation overhead to execute architectural search.

Examples for the second group of hyperparameters (training hyperparameters) are the learning rate, mini-batch size, weight decay, momentum, dropout, and number of epochs, which must be configured [167, 168]. Two common techniques to formally search the hyperparameter space are grid search and random search, however, these impose high computation overhead. Instead in this thesis, hyperparameters are tuned (a) experimentally, i.e. trial and error guided by model outputs and training trajectory, or (b) informed selection of hyperparameter values from the literature.

3.12 Subsea Survey Multi-Class Image Classification

This section presents a multi-class DL framework for the automation of visual inspection of subsea pipelines and is used as a case study to present the methodology that is also followed in the remainder of the thesis. The methodology followed is outlined in Figure 3.36.

The first step is to determine the appropriate dataset consisting of imagery along with labels for the annotations of interest. In this thesis, datasets curated by third parties are used in Chapter 3, 4, 6 while for Chapters 4, 5, 6 the datasets are also extracted from raw data; i.e. video files and file annotations Comma Separated Value (CSV). The detailed methodology and intricacies of preparing the dataset are presented in Section 4.12.

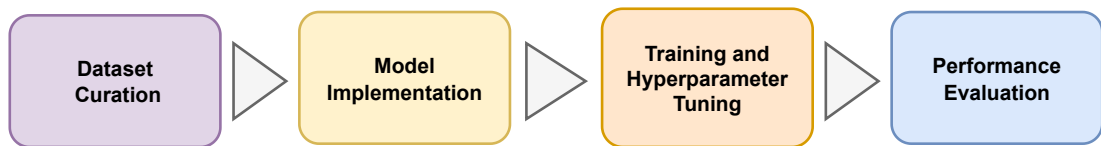


Figure 3.36: Methodology Flow Diagram

The case study starts with the dataset is description followed by the model architecture, the selection of hyperparameters, and the training process. Finally, the performance of the model is measured using the hold-out (test) set strategy.

3.12.1 Dataset Description

Subsea survey data was provided by the industrial partner N-Sea. The dataset provided from two North Sea surveys conducted in 2012 and 2016 covering 201 kilometres and 58 kilometres, respectively. Example survey frames, contained in the dataset of this work, with various lighting conditions, seabed characteristics, and parasites are shown in Figure 3.37. The dataset contains frames of the five events of interest described below.

- *Burial (B)*: the pipeline is buried beneath the seabed and thus protected.
- *Exposure (E)*: the pipeline is exposed; visible and prone to damage.
- *Anode (AN)*: pipeline bracelet anodes are specifically designed to protect subsea pipelines from corrosion [200]. Data Coordinators visually recognise Anodes by the banding that appears in the orthogonal direction of the pipeline; anodes do not have surface vegetation growth.
- *Field Joint (FJ)*: the point where two sections of the pipe meet and are joined, typically occurring every 12 metres. Data Coordinators recognise Field Joints as a result of the depression on the pipeline surface.
- *Free Span (FS)*: pipeline segments that are elevated and not supported by the seabed (either due to seabed erosion/scouring or due to uneven seabed during installation) pose a significant risk to the asset; currents or moving objects (debris, nets, etc.) could damage the pipeline. FSs are more apparent on the starboard and port video feeds; the centre camera is used to judge the seabed depth against the pipeline.

The dataset was provided by N-Sea in the form of individual frames extracted from real survey footage along with the corresponding event annotations. The dataset contains 23,570 frames in total, consisting of 5,985 frames of B, 4,236 frames of E, 6,119 frames of FJ, 2,494 frames of AN and 4,736 frames of FS. The event distribution of the extracted frames is shown in Figure 3.38. Note that all annotated data have

Chapter 3. Deep Learning Background

been checked for annotation correctness three times; one from the Data Coordinator on the vessel during the execution of the survey, subsequently on-shore by the QC personnel, and finally, after the frames are extracted, by a trained Data Coordinator who confirmed the annotations through manual inspection.

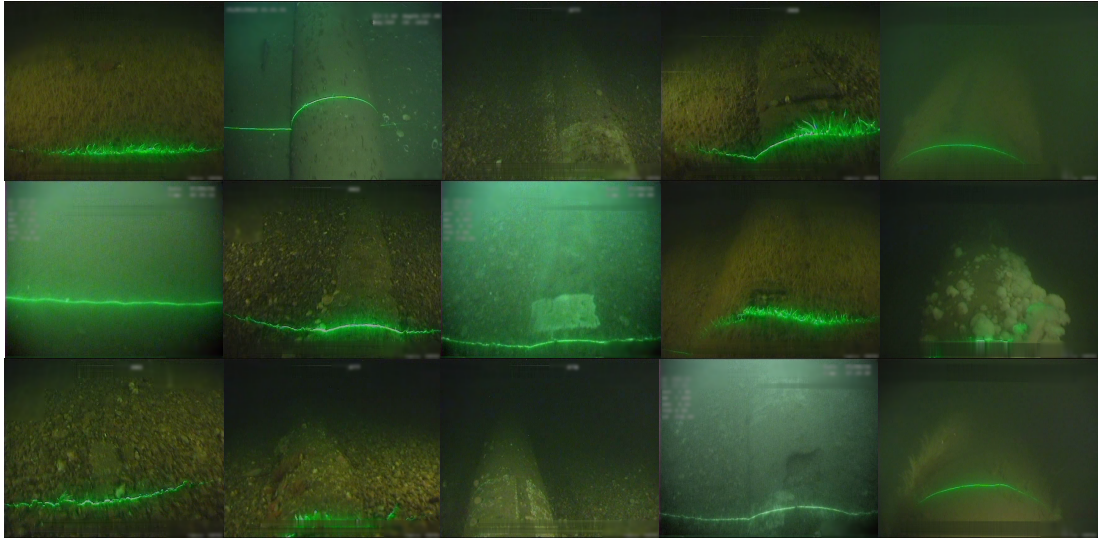


Figure 3.37: Examples of events in subsea pipeline surveys with varying scene conditions; from left to right: Burial (B), Exposure (E), Anode (AN), Field Joint (FJ), Free Span (FS)

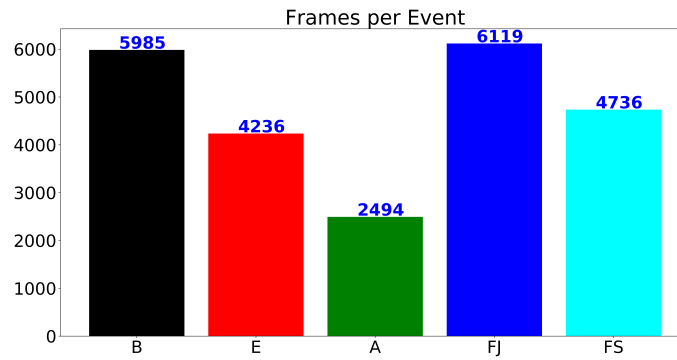


Figure 3.38: Event distribution of a total 23,570 frames of the complete dataset

3.12.2 Model Architecture

In this work, the ResNet-50 architecture is used that contains 25.6 M parameters. Typically, a network with high number of parameters and network depth demands a large training dataset to yield acceptable generalisation and performance. Creating a training dataset of this size is expensive and laborious. The alternative approach is to adopt a transfer learning methodology as described in Section 3.9, where a pre-trained network from a different domain is re-trained on data from the domain of interest (sub-sea pipeline inspection imagery in the present application). The pre-trained ResNet-50 network used is provided by PyTorch [201] trained on the ImageNet dataset [202].

The complete ResNet-50 architecture, shown in Figure 3.39, consists of five stages; each stage comprises multiple layers of convolutions, BN [150] and ReLU activations [142] that do not affect the receptive fields of the convolutional layers [142]. More importantly, the ResNet architecture uses the concept of skip (or identity) connections between stacked convolutional layers. These shortcut connections mitigate the problem of vanishing gradients in training deep architectures by allowing gradients to propagate through identity connections as discussed in Section 3.8.4. Maintaining the Feature Extraction layers is a standard methodology for application of transfer learning. In this case, all the layers in the feature extractor are kept identical with the exception of the final pooling layer.

After the fifth stage, an adaptive pooling layer is implemented that consists of Average and Max pooling, and then the features are flattened and concatenated before being fed to two fully connected (linear) layers, with the purpose of reducing the dimensionality of the features and making the dimensions equal to the number of output labels. Furthermore, BN and Dropout layers are introduced between the linear layers to regularise the Head/Classifier. The last linear layer is changed to five output neurons to match the number of events of interest. Then a Softmax activation is used to obtain the Cross Entropy Loss that is optimised during training; further details are described in the next Section 3.12.3.

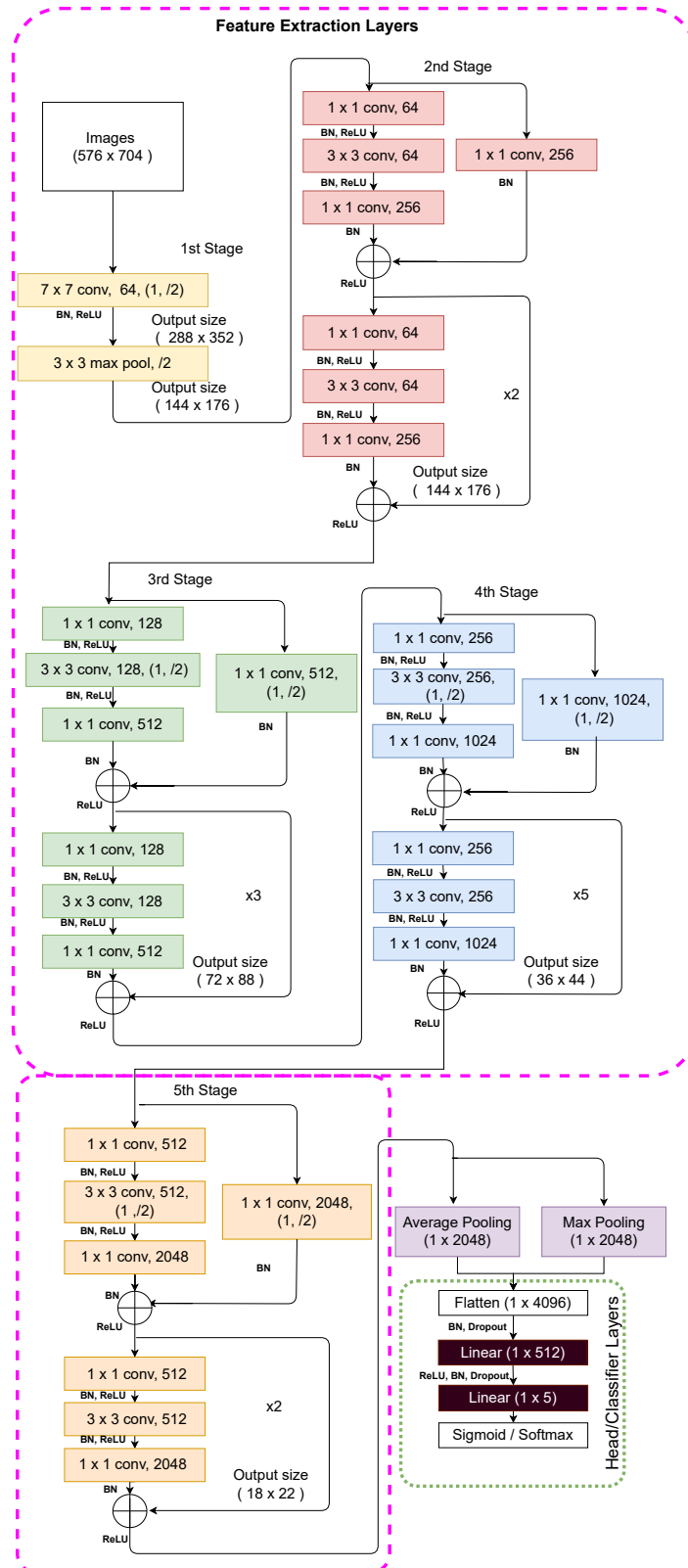


Figure 3.39: ResNet-50 Architecture with modified head.

3.12.3 Multi-class Classification

In this type of classification, each sample in the dataset belongs only to one of the C classes (five in this example). The CNN will have C output neurons that can be gathered in a vector s (scores). The target vector (ground truth) is an one-hot vector with 1 positive and $(C - 1)$ negative classes. The model predicts only one class or label. To perform multi-class classification the output of the last linear layer of the model is fed to a Softmax activation. Softmax converts the logits of every class to probabilities that add up to 1 as described in Section 3.4.3. The network is then trained to minimise the Categorical Cross Entropy (CCE) loss, which is essentially the activation of Softmax plus the loss CE. The CNN is trained to output the likelihood over the C classes for each image. CCE is used for multi-class classification.

$$CCE(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_i^C y_i \log \hat{y}_i \quad (3.36)$$

where \mathbf{y} is the one-hot encoded target y_i is the vector element at the location i , $\hat{\mathbf{y}}$ is the predicted vector output of the network and \hat{y}_i is the vector element at the location i that indicates the CNN score for the corresponding class i .

After training, the model outputs the class with the highest confidence score as the final prediction ($\arg \max$).

3.12.4 Training Details

In the present study, a deep CNN ResNet-50 [55] pre-trained on the ImageNet dataset [173] is implemented (see Figure 3.39). The rationale is that the initial layers of the pre-trained CNN are able to extract features that are generic for image classification tasks; e.g., edge detectors, colour blob detectors, etc. In the subsequent layers, network weights need to be fine-tuned to adapt to the specific features of the dataset under consideration. The network can be logically divided in two sections; the backbone feature extraction layers (enclosed in purple dashed lines in Figure 3.39) and head or classification layers (enclosed in green dashed line in Figure 3.39). The training process, unless otherwise stated, is as follows: The backbone is initialised with the pre-trained

ResNet-50 network weights distributed with PyTorch [203], while the head layers are randomly initialised with the He [147] method. The Adam optimiser [204, 205] is used for training with a mini-batch size of 10 and exponential decay parameters β_1 and β_2 equal to 0.9 and 0.99, respectively. A cyclic learning rate training [206] is used with a maximum learning rate of 10^{-3} . The cyclic learning rate allows fast convergence and avoids local minima [207] during training. The training is executed for 100 epochs and the network weights are saved in every epoch that the validation loss decreases. The converged model is considered the one which yields the lowest validation loss. The training and validation loss curves have been inspected and in all cases, the converged model is obtained well ahead the end of the training (i.e. the validation loss beyond that convergence point either remains constant or increasing). Given the high capacity of the network, the risk of overfitting the training set must be considered. Two measures are taken to prevent overfitting: regularisation through weight decay and online data augmentation which is described further in Section 3.12.5. For weight decay, the regularisation parameter λ is set to 0.01 for all layers. Training is carried out on a server equipped with two Nvidia GeForce RTX 2080 Ti, twelve Intel(R) Core(TM) i9-7960X CPU @ 2.80 GHz, and 128 GB RAM.

3.12.5 Data Augmentation

Online data augmentation is used to increase the variability of the dataset and improve the generalisation of the model by limiting overfitting [185]. A series of transformations is applied randomly to the training data, at every epoch, with a probability of 75% using the Albumentations [171] library. This means that more than one augmentation technique can be used per sample or no augmentations at all. The augmentation employed are Horizontal Flipping, Rotation (with maximum angle of 10 degrees), Scaling (with maximum variation of 1.05) and lighting alteration (Random Brightness and Contrast with maximum variation change of 0.1). Data augmentation renders the model more robust and adaptable to the artefacts created, for example, by the motion of the ROV during the survey. Examples of an augmented Exposure frame are shown in Figure 3.40. The resolution of the provided image is 576×704 and remains unchanged

to avoid information loss.

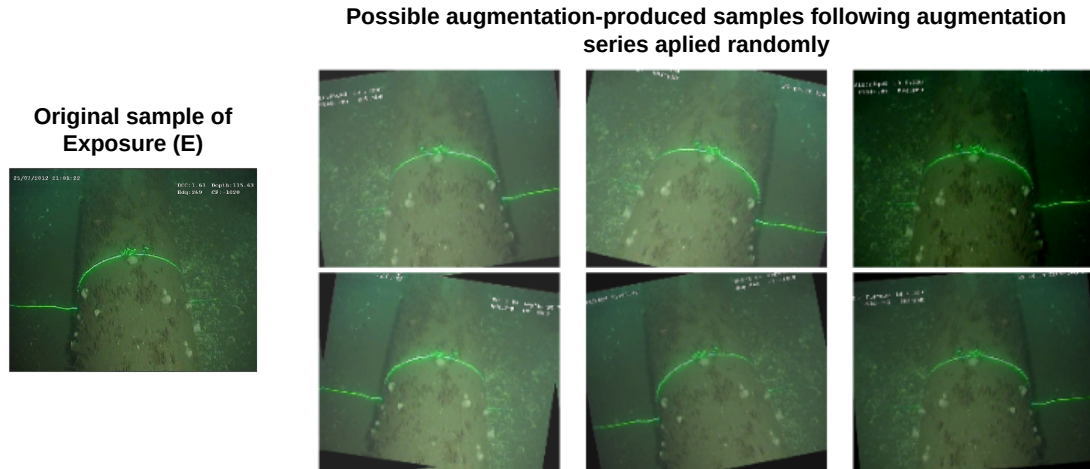


Figure 3.40: Figure presents how a series of the aforementioned augmentations can modify the original Exposure (E) sample seen on the left-hand side. For example, on the first augmented sample (top-left corner) a series of flipping, rotation, and brightness increase has been applied.

3.12.6 Model Performance Evaluation using a Hold-out (Test) set

While training a model is a key step, it is essential to know how the model performs in unseen data and consequently whether its predictions can be trusted for the target application. To evaluate the performance of a model in supervised learning, it is common to split the original dataset into three sets (60%, 20%, 20%) and calculate performance metrics on samples that are not used during the training phase. The training set (60%) is used to build predictive models. The validation set is used to assess the performance of the model built in the training phase, and it provides a metric for selecting the hyperparameters of a model. Finally, the test set, or unseen data, is used to assess the likely future performance of a model. If a model fits the training set much better than it fits the test set, overfitting is probably the cause. The purpose of hold-out evaluation is to test a model on different data than it was trained on and to provide an estimate of learning performance. This technique can be associated with high variability, as differences in the training and test dataset can result in meaningful differences in the performance estimate, depending on how representative the test set is to the train-

ing data. Cross Validation (CV) validation techniques can offer a better measure and quantify the variability between training validation and test sets, and two techniques (Monte Carlo CV [208] and Nested CV [209]) are utilised in Chapter 4. However, the purpose of this section is to present the limitations of multi-class classification for this application, and thus the simple train, validation, test split is used.

3.12.7 Performance Metrics

Performance metrics are used to evaluate and compare different classification models or to analyse the behaviour of the same model when different hyperparameters are tuned [210]. In a multi-class classification problem, when making a prediction, there are four possible outcomes for each class:

- **True Positive (TP):** The model predicts that the sample belongs to a class, and it does.
- **False Positive (FP):** The model predicts that the sample belongs to a class, but it does not (Type I error).
- **False Negative (FN):** The model predicts that the sample does not belong to a class, but it does (Type II error).
- **True Negative (TN):** The model predicts that the sample does not belong to a class, and it does not.

For a 2-class (binary) classification problem when these four outcomes are rearranged in a matrix, the Confusion Matrix is created [211], as seen in Figure 3.41. The diagonal elements represent the number of points for which the predicted label matches the true label, while anything off the diagonal is mislabeled by the classifier.

For a multi-class (higher than 2) problem the Confusion Matrix can be extended. An example for classification with n classes and the class k is showcased in Figure 3.42. When considering the class k ($0 \leq k \leq n$), the four different classification results can be obtained: TP (green), TN (orange), FP (brown), and FN (red).

The common multi-class classification performance metrics and their definitions are:

		Prediction	
		Positive	Negative
Ground Truth	Positive	TP	FP
	Negative	FN	TN

Figure 3.41: Binary Confusion Matrix

		Estimate		
		$c_0 \dots c_{k-1}$	c_k	$c_{k+1} \dots c_n$
annotated ground truth	$c_{k+1} \dots c_n$	TN	FP	TN
	c_k	FN	TP	FN
	$c_0 \dots c_{k-1}$	TN	FP	TN

TN

TP

FN

FP

true negative

true positive

false negative

false positive

Figure 3.42: Multi-class Confusion Matrix [212]

- **Accuracy:** The accuracy provides the amount of correctly classified samples by relating the number of correctly classified to the overall number of samples.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.37)$$

- **Precision, Positive Predictive Value:** Precision is the fraction of TP elements divided by the total number of positively predicted samples. Precision expresses the proportion of all samples predicted positive that are actually positive. In other words, Precision shows how much the model can be trusted when it predicts an individual as positive.

$$Precision = \frac{TP}{TP + FP} \quad (3.38)$$

- **Recall, Sensitivity, Hit Rate, True Positive Rate (TPR):** Recall is the fraction of TP elements divided by the total number of positively classified samples. Recall measures the predictive accuracy of the model for the positive class: intuitively, it measures the ability of the model to find all the positive samples in the dataset.

$$Recall = \frac{TP}{TP + FN} \quad (3.39)$$

- **False Positive Rate (FPR), Fall-Out:** FPR is calculated as the ratio between the number of negative events wrongly categorized as positive (FP) and the total number of actual negative events (regardless of classification).

$$FPR = \frac{FP}{FP + TN} \quad (3.40)$$

- **False Negative Rate (FNR), Miss Rate:** FNR is calculated as the ratio between the number of positive events wrongly categorized as negative (FN) and the total number of actual positive events (regardless of classification).

$$FNR = \frac{FN}{FN + TP} \quad (3.41)$$

- **F1-Score:** F1-Score is the harmonic mean of Precision and Recall. It is a measure of the precision of a test that considers both the precision and the recall of the test to calculate the score.

$$F1-Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3.42)$$

The F1-Score is an important metric as it balances Precision and Recall. The Recall of a class is increased when this class is over-predicted, which in turn results in a lower Precision for that class. Likewise, the Precision of a class can be boosted by predicting that class only when high certainty (small number of FP). However, this reduces the Recall of that class.

When it comes to multi-class cases, F1-Score should involve all classes giving rise to two different metrics: ‘Micro’ F1-Score and ‘Macro’ F1-Score [213]. Micro-averaging is used to compute the total metrics over all classes, where metrics are calculated globally by counting the total TP, FN and FP. Macro-averaging is used when it is desirable for all classes to be treated as equals, regardless of the possible imbalance; metrics are calculated for each class, and their unweighted mean is found, which does not take class imbalance into account.

3.12.8 Results

In multi-class classification the model selects the class with the highest probability as the final prediction, ignoring the possibility of other concurrent events. In Figure 3.43, two Exposure samples from the test are illustrated along with their predicted scores probabilities obtained after the Softmax layer. To make the final prediction, the class with the highest probability is chosen. These images are examples of False Positive classifications to highlight the confusion between the Exposure and Field Joint classes.

Wild vegetation on the surface of the pipeline can be a potential cause of visual confusion between the Field Joint and the Exposure class, but also the Anode class, as seen by the output confidence scores of the left example of Figure 3.43. Although the pipeline is exposed when Anode or a Field Joint is visible the opposite is not true. There

Chapter 3. Deep Learning Background

is need to crate a model that can identify the Exposure class with higher confidence as this is an event that occurs in the majority of the times during a subsea pipeline survey.

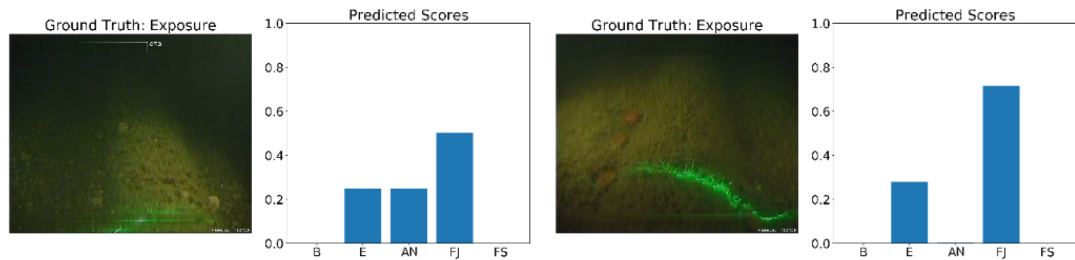


Figure 3.43: Exposure samples and their predicted scores

This is further evident when inspecting the confusion matrix in Figure 3.44 and observing the actual values versus the predicted outcomes for Exposure. Many instances of the Exposure class are confused with the Field Joint and Anode classes, and that is expected, as the pipeline has to be exposed and visible for an Anode or a Field Joint event to be present and both of these classes are not mutually exclusive to the Exposure. The confusion is smaller for the Anodes compared to Field Joints and this is attributed to the typically distinct colour of Anodes.

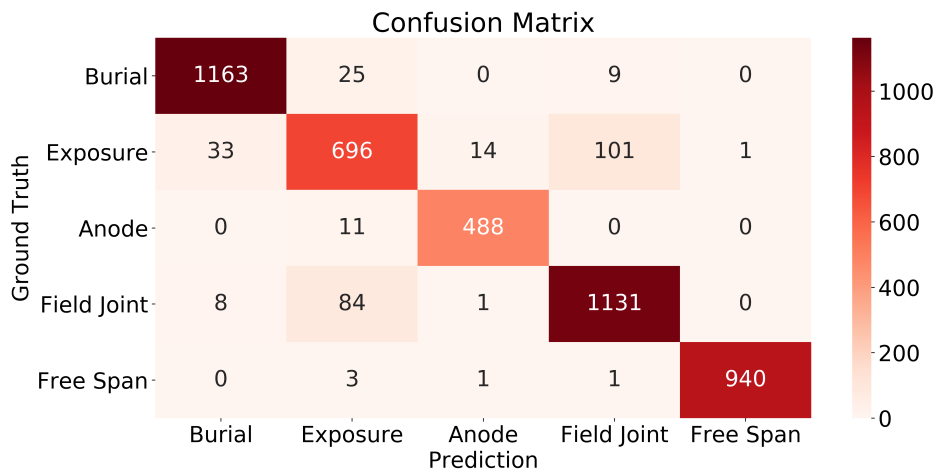


Figure 3.44: Confusion Matrix

Multi-class classification assumes mutually exclusive classes which is not the case in subsea pipeline inspection. Assuming that each image belong to only one class leads

to incorrect predictions or loss of important information when multiple events occur together, and this is the case for the 84 images of Field Joint that were reclassified as Exposure (False Negatives for the Field Joint class) and the 101 images of Exposure that were classified as Field Joints (False Positives for the Field Joint class) in the Confusion Matrix of Figure 3.44. Consequently, important details can be overlooked or misinterpreted.

Table 3.1 presents the classification report which contains Precision, Recall and F1-Score per class and the overall micro averages. Note that in the multi-class setting the averaged Accuracy, is the same as Precision, Recall and F1-Score ‘Micro’ averaged. As can be seen in Table 3.1 the Exposure class is the one with the lowest metric scores and this is attributed to the fact that the Exposure class is a superset of other events such as Field Joints and Anodes. Naturally, multi-class modelling would be more appropriate for mutually exclusive labels which is not the case in this scenario where Exposure is the superset of other classes and hence a multi-label image classification approach would be more appropriate and presented in Chapter 4.

In multi-label classification, a threshold can be used to determine the presence or absence of specific events. Each label’s predicted probability is compared against this threshold, and if it exceeds the threshold, the event is considered present. This approach enables flexibility in deciding the significance or severity of each event based on the threshold value.

Table 3.1: Test set metrics

Event	Accuracy	Precision	Recall	F1-Score
Burial	0.971	0.966	0.972	0.969
Exposure	0.823	0.850	0.824	0.837
Anode	0.977	0.968	0.978	0.973
Field Joint	0.924	0.911	0.924	0.917
Free Span	0.994	0.999	0.995	0.997
‘Micro’ Averaged	0.938	0.938	0.938	0.938

The motivation to produce these results is firstly to showcase all the steps of a DL methodology as shown in Figure 3.36, namely; dataset curation model implementation, training and hyperparameter tuning as well as performance evaluation. Secondly, this

methodology is setting the landscape to explore multi-label classification in the next Chapter, indicating the limitations of multi-class for this particular application.

3.13 Conclusions

This Chapter outlined the basic building blocks of deep CNNs, techniques, and methodologies that are used in this thesis to train and evaluate the performance of the developed models. In addition, it provided a detailed description of model architectures, the methodology for training, and the evaluation of model performance, all of which form the basis of the methodology employed in the rest of the thesis. Finally, the Chapter concludes with a case study on a subsea pipeline image classification task to demonstrate a complete DL methodology. The results of multi-class image classification indicate that there is confusion between concurrent events such as Field Joints and Exposure in subsea pipeline survey images. This provides motivation to explore multi-label image classification for subsea surveys where the model learns to associate more than one labels with one image and that is presented in Chapter 4.

Chapter 4

Subsea Survey Multi-Label Image Classification

4.1 Introduction

This chapter presents a contribution to the field of subsea pipeline surveys through the introduction of an automatic image annotation framework. The framework uses multi-label classification to identify and classify five key events of interest in subsea pipeline survey video footage: Exposure, Burial, Field Joint, Anode, and Free Span. This novel methodology represents the first application of multi-label classification in this domain, marking a significant advancement in the field, addressing the challenges of multi-class image classification previously identified, and demonstrating the capability to identify multiple concurrent events within a single frame. However, its potential extends beyond subsea pipeline surveys to other fields where concurrent events occur during inspection processes, making it a versatile approach with broad applicability. By leveraging transfer learning and deep convolutional neural networks (CNNs), the framework achieves automatic classification of the aforementioned events by analysing raw images from subsea pipeline surveys. Overcoming the difficulties posed by underwater imagery, such as dynamic ROV motion, low lighting conditions, sand agitation, sea life, and vegetation, is made possible through the use of data augmentation techniques and the integration of IBN layers into the ResNet architecture. The effectiveness of the

framework is validated through its evaluation on subsea survey frames obtained from an operational-class ROV, providing real-world applicability and demonstrating its practicality. Furthermore, the generalisation capabilities of the framework are showcased through an additional experiment, highlighting its ability to generalise from validation to test sets when the splitting of the data has been performed based on different events of a survey. Furthermore, the framework shows promise for real-time detection in a 25 fps video when embedded in an ROV. This capability opens opportunities for immediate event identification and response during inspections, enhancing efficiency and enabling timely decision-making. The framework’s ability to process data in real-time, combined with its high performance in automatic image annotation, highlights its potential as a valuable tool for subsea pipeline surveys and other industries with similar inspection requirements. By automating the annotation process and enabling rapid event detection, this methodology paves the way for improved efficiency, accuracy, and safety in various fields.

4.2 Multi-Label Classification

Multi-label image classification is widely used in the scene classification domain, where images may belong to multiple semantic classes [214–217], as illustrated in Figure 4.1.

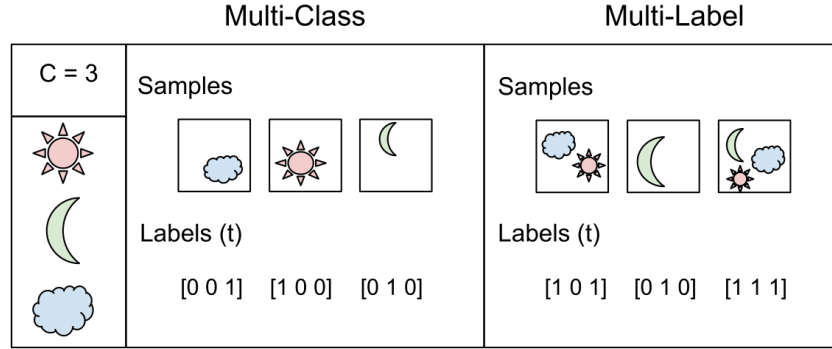


Figure 4.1: Multi-class and multi-label setting using one-hot encoded labels [218]

Multi-label classification is chosen in this study because the events that are recorded during a pipeline survey are not mutually exclusive. The pipelines are either buried beneath the seabed or exposed and thus visible. However, additional events, such as

field Joints, anodes, and free spans, are only observable when the pipeline is exposed. There are also some events in which three tags can coexist, e.g. when an anode is visible during a free span.

4.3 Dataset Description

The dataset utilised in this Chapter is identical to the dataset described in Section 3.12.1, with the difference that the labels are converted from single-label event to multi-label events. In particular, all events annotated with Anode, Field Joint and Free Span are changed to contain the Exposure as an additional label. This essentially results in a data distribution as shown in Figure 4.2 that matches the distribution presented in Figure 3.38. In summary, the dataset contains 23,570 frames in total, consisting of 5,985 frames of B, 4,236 frames of E, 6,119 frames of E and FJ, 2,494 frames of E and AN and 4,736 frames of E and FS. Note that the correctness of the original annotations has been checked three times, as described in Section 3.12.1.

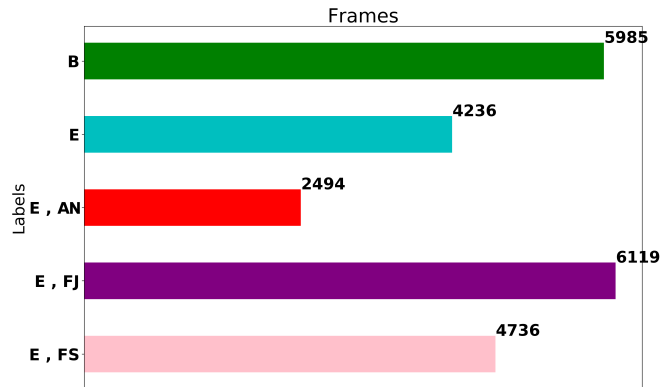


Figure 4.2: Label distribution of a total 23,570 frames of the complete dataset

4.4 Model Architecture

The CNN used in the study is based on that used in Chapter 3 that utilised the ResNet-50 architecture [55] backbone with a head modified to permit multi-label prediction. The complete architecture is shown in Figure 4.3. Furthermore, the last activation

layer is changed from Softmax to Sigmoid to perform multi-label classification as in Equation 3.11.

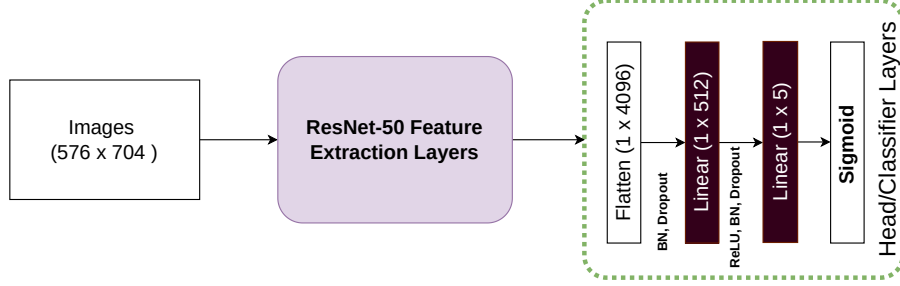


Figure 4.3: ResNet-50 Architecture with modified head and Sigmoid

The training details (initialisation, number of epochs, optimizer, etc.) are identical to those used in Section 3.12.4. In the multi-label case, the loss used is the sum of Binary Cross Entropy (BCE) for all labels, which is computed as follows:

$$BCE(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{i=1}^C [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \quad (4.1)$$

where C is the number of labels, \mathbf{y} is the one-hot encoded target (1 when the label is present in the ground truth data and 0 otherwise), y_i is the vector element at the location i , $\hat{\mathbf{y}}$ is the predicted vector output of the network and \hat{y}_i is the vector element at the location i that indicates the confidence level for the corresponding label.

4.5 Confidence Scores and Class Activation Maps

After training, the model predicts five confidence scores for the five event labels. This section provides a qualitative and visual evaluation of these confidence scores along with illustrations of CAM [219] that indicate which regions of the images are taken primarily into account during classifier predictions [220]. CNN models consist of convolutional layers and global average pooling, with Class Activation Mapping (CAM) used to identify important image regions. CAM involves projecting output layer weights back into convolutional maps, highlighting strong features for different classes and it is useful for interpreting CNN decisions, as described in Section 3.10.

In Figure 4.4 the predicted confidence scores are plotted along with the CAM illustrations, while the ground truth label is written on the top of the images. The confidence scores for each label for these samples are plotted in bar charts; these can provide a measure of confidence for the predictions.

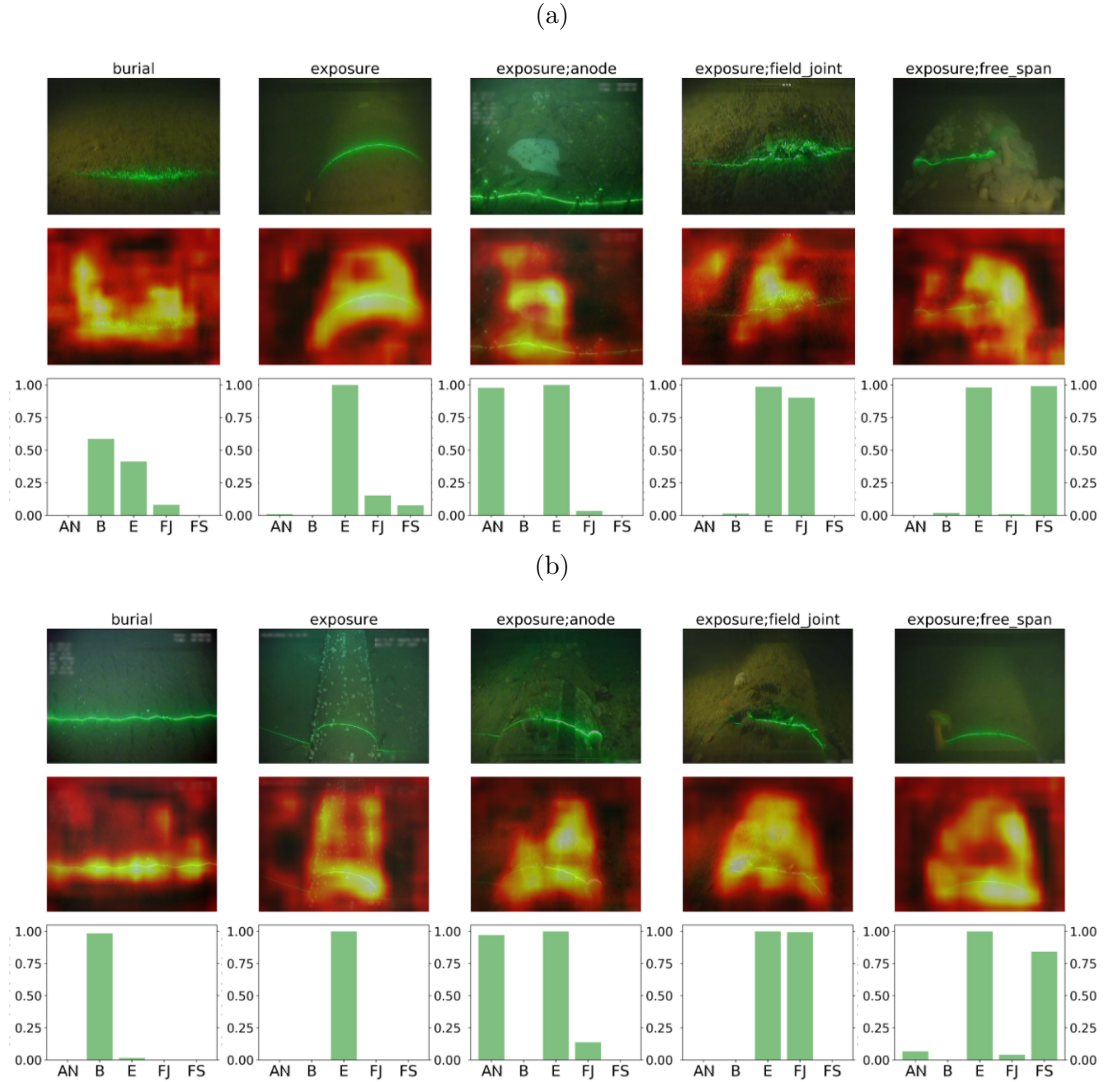


Figure 4.4: From top to bottom: Ground Truth Label, Image, CAM Heatmap, Predicted Confidence Scores for the five types of events

In the second row of Figure 4.4 the relevance of different features for each class is illustrated using Class Activation Maps (CAM). When the pipeline is buried, the most relevant feature for the Burial class is the straight laser line, as indicated by the CAM

heatmap. This suggests that the model focuses on this feature to classify an image as belonging to the Burial class.

On the other hand, when the pipeline is exposed, the most strong feature identified is the curved nature of the laser line. Additionally, the model also focuses on the cylindrical shape and well-defined edges of the pipeline when it is exposed. These features, along with the curved laser line, contribute to accurately classifying images as belonging to the Exposure class.

For the Field Joint class, the unstructured depression or hole in the middle of the pipeline becomes an important feature. Similarly, for the Anode class, the dominant feature is identified as the characteristic white bracelet. The CAM can emphasize this feature, aiding in the classification process. Interestingly, the Anode of Figure 4.4b does not have a white color but the features that are picked up are similar to the Anode of Figure 4.4a indicating the robustness of the model.

In the case of the Free Span class, the absence of the green laser line on the left and right sides of the pipeline becomes a significant feature. The CAM can highlight regions where the laser line is not visible, assisting in the classification of images as Free Span. Additionally, the description mentions that the dark background is the least heated region, which implies that the CAM can potentially capture this information as well.

In summary, this illustration can show how different classes have specific features that are relevant for classification, and the CAM can effectively highlight these features, aiding in the interpretation of the model’s decision-making process.

Furthermore, these examples, which have been intentionally extracted from the test dataset, highlight the large variation in the image scenes. Taking into account the entire dataset, these variations include differences in colour (green, brown, grey), type of seabed (sand or gravel), vegetation (low or high), and distance and orientation of ROV with respect to the seabed. These examples emphasise the different imaging conditions present during subsea pipeline surveys, including poor lighting conditions and alien elements such as fish, vegetation, or sand suspension. These different examples constitute a natural augmentation of the dataset, which increases its variability and leads to a model with enhanced generalisation ability.

4.6 Threshold Tuning

In multi-label classification, after the final Sigmoid activation layer, a vector is obtained with degrees of confidence that each label is associated with the input image. To obtain the final prediction, a threshold must be defined to make the output discrete; 1 if the confidence score exceeds the threshold; 0 otherwise. The threshold can be defined using the same value for all labels or by defining multiple thresholds, one for each label [141, 221, 222]. These thresholds are hyperparameters of the model that can be tuned using a validation set and evaluated on a test set. Selecting thresholds allows for a trade-off in errors made by the model, such as the number of FP compared to the number of FN. This is required when using models where the cost of one error outweighs the cost of other types of error. This must be taken into account, especially for problems where an unequal distribution of labels is present within the dataset [223]. Two diagnostic tools that help in the interpretation of confidence scores for multi-label classification predictive modelling problems are Receiver Operating Characteristic (ROC) curves and PR curves [224].

ROC curves is a plot of the False Positive Rate (FPR) (x-axis) versus the True Positive Rate (TPR) (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. The TPR shows the performance of the model in predicting the positive class when the actual result is positive. The FPR summarises how often a positive class is predicted when the actual outcome is negative. A model and a threshold with ideal performance will be placed at the point (0, 1) of the ROC. By plotting the ROC curve for the trained model with the predictions in the validation set, optimal thresholds can be chosen for each of the labels that provide a desirable balance between the TPR and the FPR.

The disadvantage of the ROC in the multi-label setting is that imbalance could affect threshold selection. For example, consider that the E label is present in most samples (almost 80%) and therefore a high number TN exists for the other labels that affect FPR.

An alternative tool that does not employ TN is the PR curves that utilise Precision

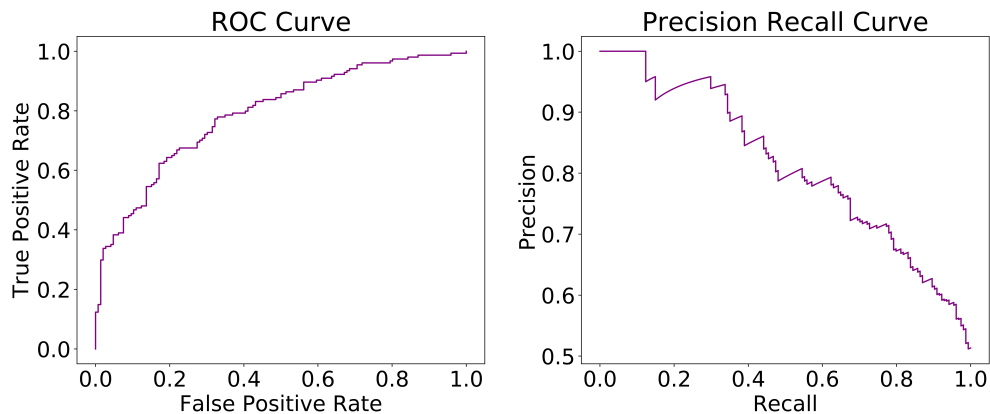


Figure 4.5: Example of ROC and PR Curves

and Recall. PR curves are more appropriate for cases where there is an imbalance in observations between classes [225]. A PR curve represents a PR ratio for different threshold values, with Precision on y-axis and Recall on x-axis. The optimal threshold for each label is defined as the point that achieves the best balance between Precision and Recall and therefore corresponds to the closest point to the upper right corner of the graph (coordinate $(1, 1)$). It is equally important to maximise Precision and Recall to provide the maximum F1-score.

The primary evaluation methodology in this Chapter uses the PR curve and individual thresholds per label, because there is an imbalance between classes. The selection of thresholds is a means of adjusting the sensitivity of the model for each label. Low thresholds will lead to high detection sensitivity at the expense of FP, while high thresholds will reduce FP at the expense of missed positives [226]. The five threshold values make up the model hyperparameters and the PR curves are used to determine the optimal values, as illustrated in Figure 4.6. Note that the definition of optimal thresholds is executed solely using the validation set, only containing images that were not seen during the training phase.

Additional experiments are presented in Sections 4.11.1 and 4.11.2 using a universal threshold for the five labels and ROC curves demonstrating that the use of the PR curves provides the best performance in this scenario of imbalance in the multi-label setting.

4.7 Model Performance Evaluation and Cross Validation

The training, validation and testing methodology to evaluate the performance of the proposed network is shown in Figure 4.6. The complete dataset contains 23,570 frames with annotation according to the label distribution shown in Figure 4.2. Initially, 20% of the frames in the dataset, are selected in a stratified fashion and set aside to be used as a test (hold-out) set and in the evaluation of the performance of the model after training/validation and hyperparameter tuning.

Stratified sampling is a sampling method that reduces sampling error in cases where the population can be partitioned into subgroups, here the five labels. Stratified sampling is performed by dividing the population into homogeneous subgroups, called strata, and then applying random sampling within each subgroup [227]. As a result, the test set is representative of the population, since the percentage of each subgroup is preserved. This methodology produces a test set of 4,714 frames with a label distribution approximately equal to that shown in Figure 4.2.

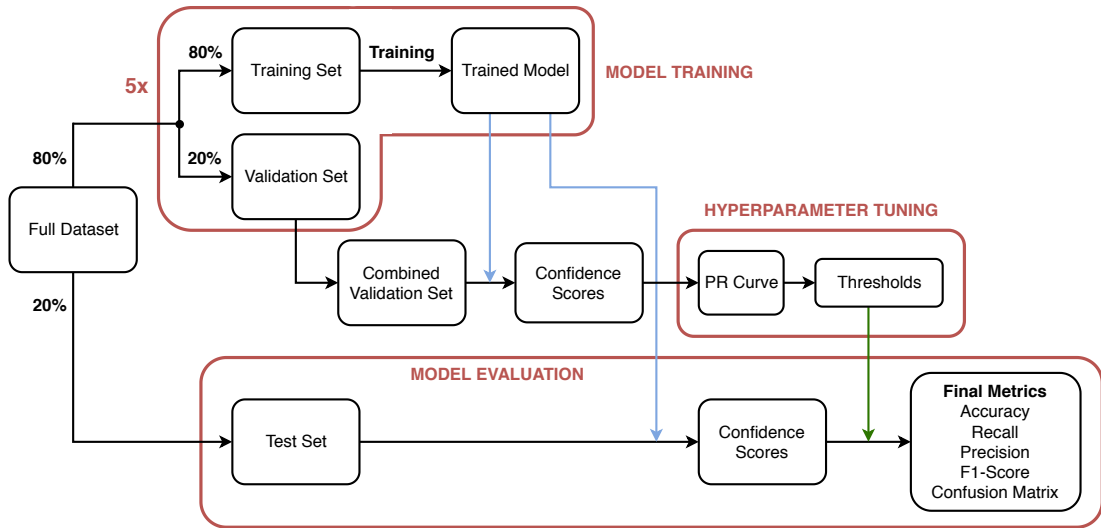


Figure 4.6: Model training and evaluation process

The remaining 80% (18,856 frames) of the dataset are used to perform CV. CV is a technique used to test the ability of a model to predict unseen data [228]. The most basic CV, known as k -fold CV divides the training data into k folds, and the model trains on $k - 1$ folds and is validated using the remaining one fold, as illustrated in

Figure 4.7. The process is repeated for k iterations so that each fold is in the validation set once. CV can be used to analyse the generalisation of the classification model, the average error is used for all iterations to evaluate the model. Using a model with better CV performance is always preferable. Similarly, CV can also be used to adjust the hyperparameters. The final evaluation is performed on the test set using the model with the highest performance among the folds.

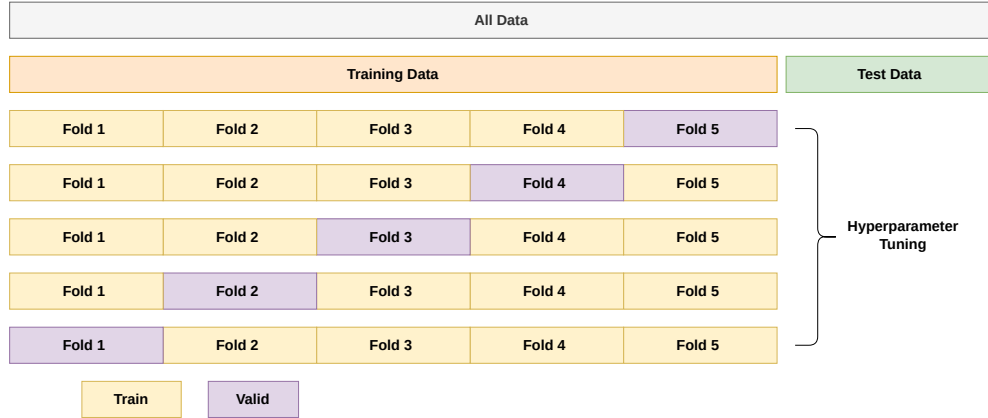


Figure 4.7: 5-fold Cross Validation

An alternative is Monte Carlo [208, 229] CV, also known as random subsampling CV. In this approach, the training data are split randomly k times using stratified sampling, as seen in Figure 4.8. For each iteration, the percentage of train validation split may be different. The model is trained on the training data for that iteration, and the validation metrics are calculated using the validation set. This process is repeated for multiple iterations, and the mean and standard deviation of the performance metrics in the validation set are calculated. The same data can be selected more than once in the validation set or can be excluded. Therefore, Monte Carlo CV results in higher bias but low variance, compared to the K-fold validation, where each sample is tested (it belongs to the validation set) exactly once.

Here, Monte Carlo CV is used for the remaining 80% of the dataset as seen in Figure 4.6, with stratified splits of 80/20%, that is, 80% of the data (15,085 frames) is used to train the model and the performance is validated on the remaining 20%; as a validation set (3,771 frames). The process is repeated multiple times (5 in this

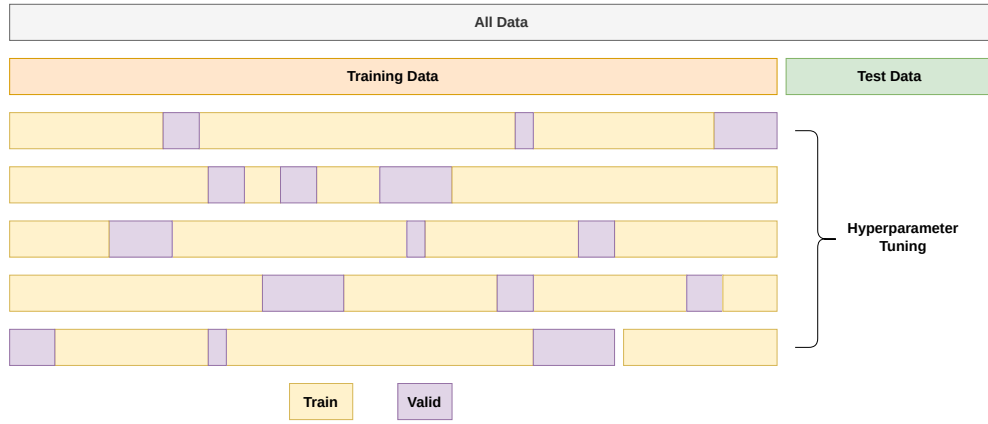


Figure 4.8: Monte Carlo Cross Validation

study) to evaluate the variability of the trained models and their performance on the validation sets. After hyperparameter tuning (threshold selection), the performance of the model is obtained in the test set to ensure representative performance on unseen data.

4.8 Multi-Label Performance Metrics

In a multi-class problem formulation, classes are mutually exclusive. In other words, under the condition of mutual exclusivity, each training example can belong only to one class. Measurement of the performance of a multi-label classifier is more challenging than single-label classification, because each sample can be associated with more than one label simultaneously [140]. In multi-label classification, a misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual labels is considered better than a prediction that contains many incorrect labels; i.e., in an Exposure Anode event, predicting one out of the two labels correctly is better than predicting Exposure, Anode, Burial, Field Joint and Free Span simultaneously. Therefore, evaluation metrics for multi-label classification should not only be calculated for the entire validation dataset but also for each label individually [230]. In particular, the following metrics are of interest, namely: Accuracy, Recall, Precision, F1-Score, presented in Section 3.12.7.

When computing metrics for a single label the problem is considered as a binary,

one-versus-rest classification [230]. For aggregate performance (Precision, Recall, F1-Score), then metrics are calculated globally over all instances in the validation set, by counting the total TP, FN, and FP and the ‘Micro’ averaging [231] as described in Section 3.12.7. The exception is for aggregate Accuracy; in this case, successful classification counts are used only after all the labels have been identified correctly, commonly also known as Exact Match Ratio (EMR). This is a stricter metric, compared to average Accuracy and does not differentiate between partially incorrect and complete incorrect predictions. Formally, the EMR is defined as:

$$\text{Exact Match Ratio} = \frac{1}{n} \sum_{i=1}^n \mathbf{I}(\mathbf{y}_i = \hat{\mathbf{y}}_i) \quad (4.2)$$

where $\mathbf{I}(\mathbf{y}_i = \hat{\mathbf{y}}_i)$ is the indicator function equal to 1 only when every element in the vector \mathbf{y}_i is equal to every element in $\hat{\mathbf{y}}_i$ and n is the number of input samples. Note that for a binary classification (i.e. individual labels), this reduces to Accuracy (Equation 3.37).

4.9 Model Performance on Validation Sets

Steps 1-4 in Figure 4.9 illustrate the process followed to obtain the optimal threshold selection in the validation set. Note that, due to the use of 5-fold Monte Carlo cross-validation, five different models are trained, one for each validation fold. The predictions obtained from the five independent models on the five different validation folds are concatenated and used to determine the optimum set of thresholds/hyperparameters. PR curves for every label can be generated to evaluate the performance of the classifier with increasing threshold values. For each threshold value, the final set of predictions is evaluated against the corresponding ground truths on an individual label basis to identify each prediction as TP, FP, TN or FN. The Precision and Recall of the classifier are then calculated using Equations 3.39 and 3.38 (Step 4 in Figure 4.9) and are shown in Figure 4.10.

The optimal threshold is defined as the point that achieves the best balance between Precision and Recall and therefore corresponds to the closest point to the upper right

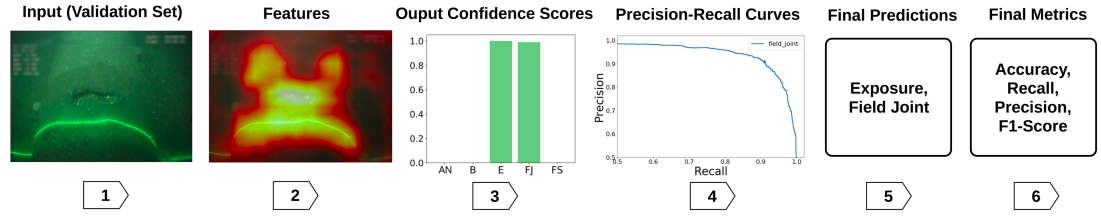


Figure 4.9: Steps for evaluating model's performance: (1) Validation Set (2) Feature Extraction (3) Classifier (4) PR Curves for optimal thresholds selection (5) Applying optimal thresholds (6) Comparison with Ground Truth

corner of the graph (coordinate (1,1)). The strategy to define the optimal threshold was selected because in this application it is equally important to maximise Precision and Recall, and hence maximise F1-Score.

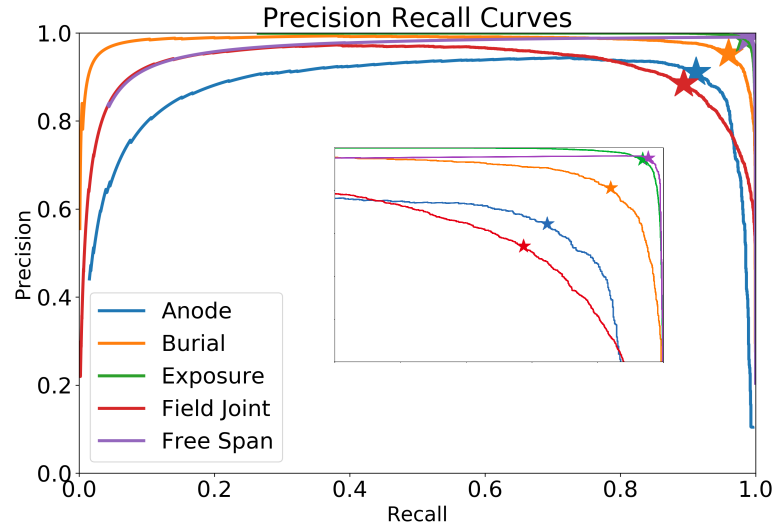


Figure 4.10: Precision Recall curves for all labels; the inset shows a zoomed version of the top-right corner

Applying the methodology for the five types of events (Anode, Burial, Exposure, Field Joint, and Free Span), results in the PR curves shown in Figure 4.10. The optimal thresholds are at the locations depicted by the stars (“*”) caret on the graph and yield thresholds for each event type, summarised in Table 4.1. Using the optimal thresholds identified by the PR curves, the aggregate performance metrics (Equations 3.37-4.2) for each model in its corresponding validation fold are shown in Table 4.2. The highest

performing fold, fold 3, is highlighted in bold and it is used in Section 4.10 to obtain performance for the test set.

The 3rd-fold model performs best for the validation set of this fold and it is assumed that it will perform best for the hold-out test set. This is not necessarily the case in general; however, the additional experiments that have been performed in Sections 4.12.1 showcase the low Standard Deviation in the performance of the different models on the same testing set as well as the low Standard Deviation performance across different tests sets, indicating that this choice is not hurtful.

Table 4.1: Optimal label-based thresholds for the validation set

Event	Anode	Burial	Exposure	Field Joint	Free Span
Threshold	0.357	0.367	0.632	0.542	0.430

Table 4.2: Aggregate performance of the five models, one for each fold

Fold #	EMR	Precision	Recall	F1-Score
1	0.907	0.958	0.961	0.960
2	0.890	0.949	0.956	0.953
3	0.920	0.972	0.961	0.967
4	0.914	0.962	0.967	0.964
5	0.899	0.954	0.958	0.956

Similarly, the average performance of the five models for each type of event is shown in Table 4.3 along with the standard deviation for each metric. Field Joints are the most challenging class with the lowest F1-Score of 88.9%, this is expected given that such events are often difficult to distinguish due to subtle features. Furthermore, this is evident in Figure 4.10 where the Field Joint PR curve resides further away from the ideal performance; point (1, 1) compared to other events. The curve also indicates higher sensitivity to the threshold selection as the locations near the optimal threshold “*” have higher slopes. This leads to higher standard deviation in the Precision, Recall metrics presented in Table 4.3. Similar observations are valid for the Anode event. On the other extreme, Free Span and Exposure show high performance, with F1-Score of 98.8% and 98.5%, respectively and the corresponding PR curves support that lower sensitivity in the threshold selection. Overall, the aggregate ‘Micro’ F1-Score across all

labels is 96%.

Table 4.3: Metrics with optimal thresholds on the validation set.

Event	Threshold	EMR		Recall		Precision		F1-Score	
		Average	Std	Average	Std	Average	Std	Average	Std
Anode	0.357	0.981	0.006	0.910	0.028	0.912	0.046	0.911	0.028
Burial	0.367	0.978	0.001	0.959	0.011	0.953	0.013	0.956	0.004
Exposure	0.632	0.978	0.001	0.984	0.004	0.986	0.003	0.985	0.001
Field Joint	0.542	0.942	0.008	0.893	0.020	0.885	0.024	0.889	0.015
Free Span	0.430	0.995	0.002	0.988	0.002	0.988	0.013	0.988	0.007
Aggregate		0.906	0.011	0.961	0.004	0.959	0.008	0.960	0.005

4.10 Model Performance on Test Set

To ensure that thresholds are not biased towards the validation set and obtain the final model performance, evaluation is carried out on a previously unseen (hold-out) test set (Figure 4.6); that is, images that have not been used for training, validation, or hyperparameter tuning. CV has yielded five different models and the model selected for the final test is the one that provides the highest F1-Score; in this case it is the third-fold model, shown in bold in Table 4.2.

Figure 4.11 shows the confusion matrices for each label, obtained using the final model on the test set. Confusion matrices, commonly employed in multi-class classification scenarios, are not typically utilised for multi-label classification due to the simultaneous association of multiple labels with each instance. Consequently, the conventional confusion matrix, designed for mutually exclusive classes, proves less suitable for evaluating multi-label classification models directly. The conventional representation assigns each cell of the confusion matrix to the count of instances belonging to a specific combination of predicted and actual classes. However, in multi-label classification, instances can possess multiple predicted and actual labels, rendering the creation of a straightforward confusion matrix representation challenging.

To visually depict the per‘one versus the rest’ approach widely used in multi-class classification can be adapted. This approach treats each label as a distinct binary classification problem, allowing the creation of separate confusion matrices for each

Chapter 4. Subsea Survey Multi-Label Image Classification

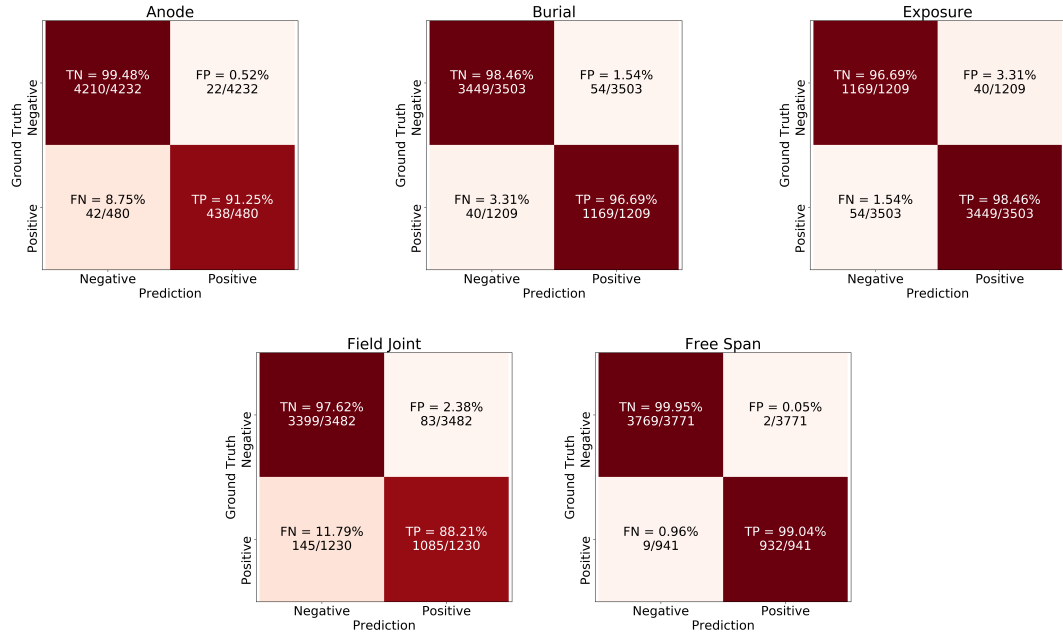


Figure 4.11: Confusion matrices on the test set for each class; Anode, Burial, Exposure, Field Joint and Free Span.

label. Consequently, the model performance for the individual labels can be analysed independently, facilitating the identification of its strengths and weaknesses [232].

Each label is considered positive if it is present in the image frame and negative otherwise. The confusion matrices show the absolute number of frames and the percentage of TN, FP, FN, TP. For instance, the total number of frames in the test set is 4,712 frames with 480 frames associated with the label ‘Anode’ and 4,232 are without. From the 480 frames that are labelled as ‘Anode’ (positive frames), 438 (91.25%) have been correctly identified by the model (TP) and 42 (8.75%) have been missed (FN). In terms of FP, 22 frames have been incorrectly identified as anodes out of 4,232, that is, a FPR of 0.52%.

From the confusion matrices, the Field Joints are the most challenging label with a False Negative Rate (FNR) of 11.79% and a FPR of 2.38%. FJ mis-classifications can be attributed to visual artefacts in the imagery, for example, small rocks or vegetation. It should be noted that these metrics are obtained on a single-frame basis and for a practical deployment, the classifier will be applied in a video stream with frame rate

of 25 fps, the probability that these artefacts appear in all frames is reduced hence reducing the FNR. Similarly, the FPR can be reduced by aggregating predictions from adjacent frames and these approaches are explored in Chapter 5, albeit on a slightly different dataset that contain consecutive frames. The performance of the network per label is summarised in Table 4.4 with an overall, the EMR of the network is 91.9% and the F1-Score is 96.6%.

Table 4.4: Test set performance of individual labels and aggregate.

Event	Threshold	EMR	Precision	Recall	F1-Score
Anode	0.357	0.986	0.952	0.912	0.931
Burial	0.367	0.980	0.955	0.966	0.961
Exposure	0.632	0.980	0.988	0.984	0.986
Field Joint	0.542	0.951	0.928	0.882	0.904
Free Span	0.430	0.997	0.997	0.990	0.994
Aggregate		0.919	0.972	0.960	0.966

Comparing these results with the Table 3.1 of Section 3.12.8 of the multi-class approach, it is indicated that the F1-Score of the individual label Exposure improves by ≈ 0.15 and this improves the aggregate F1-Score for the multi-label approach by 0.03. For the rest of the labels their individual performance remains similar apart from the Recall of the Field Joint label that is decreased as there are more False Positives for this label.

One advantage of multi-label classification is that the model can learn to associate multiple labels with an event and therefore the model can capture the relationships and dependencies between different labels.

In particular, the Exposure and Field Joint events have dependencies. By using multi-label classification, the model can learn to recognize these associations. For example, if an inspection identifies a specific defect, it may be indicative of other related defects or issues that should also be checked for. The model can learn to make these connections. This association increases the performance for the Exposure label but at the same time with a trade-off of having more False Positives for the Field Joint label.

4.11 Additional Experiments

The methodology described in this Chapter was used to execute further experiments to understand the effect of model design choices. In particular, the additional experimentation investigates, the effects of: universal thresholding of confidence scores, thresholding using ROC curves, exploring different methods of network weight initialisation, different depths of residual models and substituting the BN layers with IBN layers.

4.11.1 Universal Thresholding

One approach to obtain the final predicted labels is by applying a universal threshold to the classifier output confidence scores for the five labels. If the confidence score of a label is greater than this threshold, this label is detected and presented in the final prediction, otherwise this label (event) is not detected. Figure 4.12 presents the aggregated metrics when the threshold value increases from 0 to 1, for the test set. For a universal threshold with a of 0.375 Precision, Recall and F1-Score intersect and reach a value of 0.965 where EMR is 0.913 for the same threshold. These metrics present similar performance with the PR curves in terms of Precision, Recall and F1-Score as seen in Table 4.5, however, using PR curves to optimise individual thresholds improves the EMR to 0.919. In this specific application and task, using a universal thresholds results in a similar performance with the method of using individual thresholds per label, however, thresholding using PR or ROC curves can potential increase the performance in tasks where datasets are imbalanced.

Table 4.5: Test set performance of individual labels and aggregate using universal thresholding

Event	Threshold	EMR	Precision	Recall	F1-Score
Exposure	0.375	0.981	0.981	0.994	0.987
Burial	0.375	0.982	0.961	0.968	0.965
Field Joint	0.375	0.950	0.925	0.879	0.901
Anode	0.375	0.986	0.943	0.929	0.936
Free Span	0.375	0.994	0.979	0.995	0.987
Aggregate		0.913	0.966	0.966	0.966

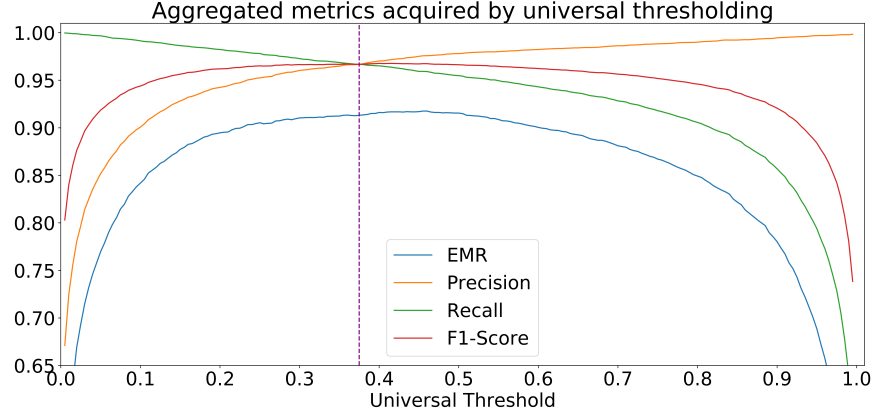


Figure 4.12: Aggregated metrics while the threshold value increases from 0 to 1 for all labels.

4.11.2 Multi-threshold using ROC curves

ROC curves for the multi-label classification can be used in a similar fashion to the binary classification by considering each label independently [233]. ROC curves summarise the trade-off between TPR and FPR for the model using a threshold per label [234]. As illustrated in Figures 4.13 the closest point to the top left corner of the graph gives the optimal threshold, since it minimises FPR and maximises TPR. Table 4.6 shows resultant threshold values using the ROC curves and the corresponding performance metrics on the test set. It is evident that the overall performance is lower when defining threshold values using the ROC curves compared to that of PR curves, and specifically the Precision of Anode and Field Joint labels drops significantly from 0.952 to 0.830 and from 0.928 to 0.866, respectively. This is attributed to the fact that the PR curves are more suitable for unbalanced datasets, which is the case in this setting.

4.11.3 Randomly Initialised Network Weights

For the results presented so far, the network weights were initialised with pre-trained weights obtained from a ResNet-50 trained on ImageNet. To evaluate the effect of transfer learning, the weights of the network were randomly initialised using the Kaiming initialization [147]. The results obtained are shown in Table 4.7. Although there

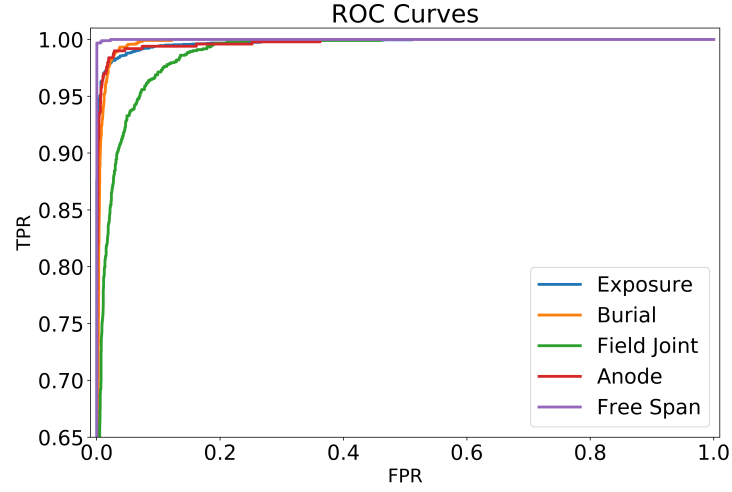


Figure 4.13: ROC curves for the five labels

Table 4.6: Test set performance of individual labels and aggregate using ROC thresholding

Event	Threshold	EMR	Precision	Recall	F1-Score
Anode	0.104	0.976	0.830	0.979	0.898
Burial	0.217	0.980	0.945	0.979	0.962
Exposure	0.786	0.980	0.993	0.980	0.986
Field Joint	0.198	0.947	0.866	0.941	0.902
Free Span	0.909	0.996	0.994	0.988	0.991
Aggregate		0.908	0.950	0.974	0.962

is no significant difference in performance between the randomly initialised networks compared to the ResNet-50 with pre-trained weights, the training iterations required to achieve this were significantly longer; the randomly initialised weights models converged at around ~ 90 epochs instead of ~ 55 , indicating that transfer learning reduces convergence time by almost two times.

Table 4.7: Test set performance of individual labels and aggregate using a randomly initialised ResNet-50

Event	Threshold	EMR	Precision	Recall	F1-Score
Anode	0.467	0.985	0.935	0.931	0.933
Burial	0.411	0.978	0.966	0.949	0.957
Exposure	0.588	0.978	0.982	0.988	0.985
Field Joint	0.367	0.942	0.897	0.878	0.887
Free Span	0.791	0.997	0.995	0.989	0.992
Aggregate		0.910	0.964	0.960	0.962

4.11.4 Effect of Model Size

Identical evaluation performance was carried out for ResNet models with 18, 34, 101, and 152 layers (in addition to the 50 mentioned above). The resulting performance metrics on the test set for each model size are summarised in Table 4.8. As the complexity and capacity of the model increases in the ResNet-50 architecture, the F1 scores initially improve. Further increases in the number of layers (that is, 101 and 152) result in performance degradation. Larger models tend to overfit faster. This is likely to occur given the training parameters are kept identical; i.e. number of epochs, regularisation coefficients, learning rates, etc., and altering these parameters may be necessary to achieve optimal prediction accuracy. Although larger networks have the potential to achieve a better F1-Score, as the number of layers increases, the number of parameters increases significantly along with the inference times. Note that the inference time reported in Table 4.8 is the average computation time on 100 frame predictions; i.e. 100 forward passes. For deeper networks, inference times are marginally within the limits of real-time operation. From these results, the ResNet-50 model is selected as it provides the best performance with inference time within the limits of real-time operation. The inference time was measured on an NVIDIA GeForce RTX 2080 Ti Graphics Processing

Unit (GPU).

Table 4.8: Test the performance of the set of different sizes of ResNet models

Network	# Parameters	Inference Time (ms)	EMR	Precision	Recall	F1-Score
ResNet-18	11,706,949	17.7	0.872	0.945	0.947	0.946
ResNet-34	21,815,109	20.8	0.903	0.953	0.966	0.960
ResNet-50	25,617,477	23.6	0.919	0.972	0.960	0.966
ResNet-101	44,609,605	31.2	0.916	0.956	0.973	0.965
ResNet-152	60,253,253	39.1	0.833	0.931	0.927	0.929

To achieve real-time event detection on a video with a frame rate of 25 frames per second (fps), it is necessary for the prediction on a single frame to be performed within a time constraint of 40 milliseconds (ms). This ensures that each frame can be processed and predictions generated within the specified timeframe, thereby keeping up with the video’s frame rate.

Adherence to this requirement is essential to maintain real-time analysis and to enable timely identification of events. It is imperative that the prediction time per frame be sufficiently fast, ensuring that delays or lag in the event detection process are avoided. This facilitates smooth and continuous processing that aligns with the video’s frame rate of 25fps, thus enabling efficient real-time event detection on the video.

4.11.5 Training with Instance Batch Normalisation Layer

Subsea pipeline survey frames can vary significantly in lighting conditions (brightness and contrast), as reflexions vary with depth and vegetation [29]. Style augmentations (random brightness and contrast changes) cannot completely mitigate these challenges. IN has been widely used in Style Transfer [235–237] and is used to learn features that are invariant to changes in appearance, such as colours, style, and virtuality / reality, while BN is essential for preserving content-related information. Pan *et al.* [238] combined IN with BN and modify the ResNet block as illustrated in the diagram in Figure 4.14. This increased the model performance without increasing the computational cost. The proposed IBN-Net achieves improvements comparable to other domain adaptation methods when applying trained networks to new domains [238]. For example, when training an IBN-Net model on GTA5 [239] dataset and applying it to the new

domain Cityscapes [240]. The combination of IN and BN leads to better generalisation by filtering out sample-specific contrast information while preserving content.

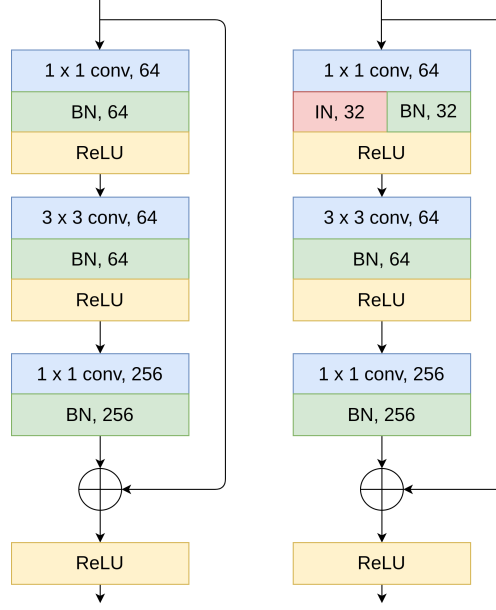


Figure 4.14: ResNet original block and IBN-ResNet block [238]

Changing the BN layers of ResNet-50 with a combination of IN and BN has been applied to the subsea automatic event annotation presented in the thesis, resulting in model performance improvements with no additional computation cost. As seen in Table 4.9 the F1-Score of the Field Joint label increases from 0.902 to 0.947 with the aggregate EMR increasing from 0.919 to 0.952 and the aggregate F1-Score from 0.969 to 0.979.

Table 4.9: Test set performance of individual labels and aggregate using IBN-ResNet-50

Event	Threshold	EMR	Precision	Recall	F1-Score
Anode	0.370	0.995	0.970	0.989	0.980
Burial	0.448	0.984	0.974	0.964	0.969
Exposure	0.563	0.985	0.988	0.991	0.990
Field Joint	0.483	0.972	0.945	0.949	0.947
Free Span	0.585	0.998	0.996	0.996	0.996
Aggregate		0.952	0.979	0.981	0.979

4.12 Splitting Based on Subsea Survey Events

The dataset used in this chapter comprises individual frames that were manually annotated, but the specific event associated with each frame, remains unknown. Consequently, it is highly probable that the frames within the dataset are consecutive or in close proximity to each other, indicating that they coincidentally belong to the same event. This is particularly problematic when trying to assess the generalisation of the model, because highly correlated frames can belong to the training, validation, and test (hold-out) sets. This phenomenon is known as leakage and could lead to an over-estimation of the model performance. To obtain a more representative generalisation performance the experimentation was repeated on a dataset that frames on the training, validation and test do not belong to the same events.

Survey dataset: Distribution of events

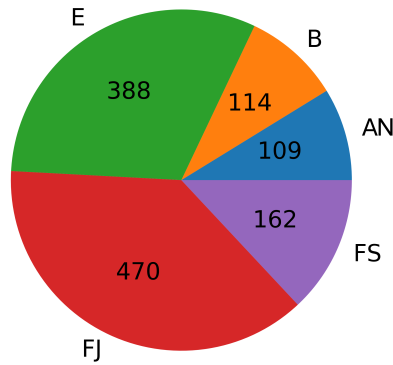


Figure 4.15: Event Distribution

A new video dataset was provided by N-Sea from two surveys conducted on 2012 and 2016. The survey data were provided as video files along with event annotations after the QC. The video files are provided in MPEG format with a resolution of 576×704 and a frame rate of 25 fps. The video filenames contained the timestamp for the start of the recording. The event distribution across the two surveys is shown in Figure 4.15. The annotation procedure followed by the data coordinator provides start

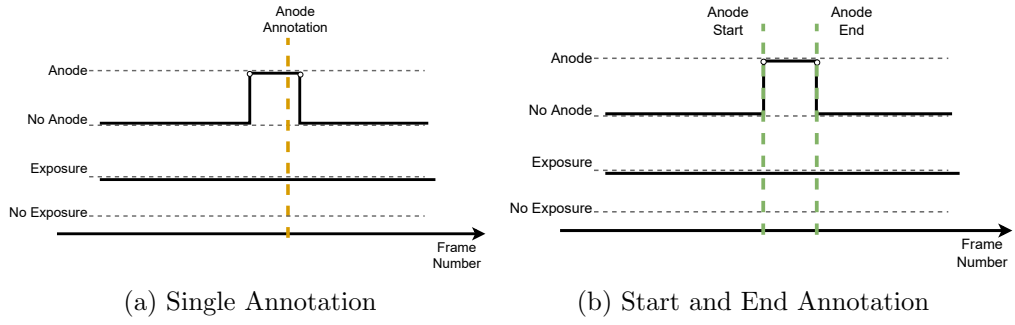


Figure 4.16: Examples of annotation regimes for ‘short’ events such as FJ and AN (a) single annotation (b) annotation with start and end delineations.

and end annotations for ‘long’ events such as FS and E (note that B is assumed to be between the E end and E start annotations), but provide a single annotation for ‘short’ events such as AN and FJ near their vicinity. An example of a single annotation for an anode event is shown in Figure 4.16a. The graph depicts the frame number on the horizontal axis the vertical axis shows if an event is present or not. In this video segment the pipeline is exposed and a single AN event. Extracting multiple frames from ‘short’ events is challenging because there is no clear delineation of when the event started or ended. Ideally, start and end annotation would be provided to allow use of all frames while the event is visible as shown in Figure 4.16b; where the green dashed lines mark the beginning the end of the event. Given that these annotations were not available, it was assumed that the single annotations for the ‘short’ events was located roughly in the middle of the event duration and that duration was approximately 3 s (these assumptions have shown to be appropriate based on the analysis presented in Figure 5.10 of Section 5.4). Based on these assumption, frames are extracted ± 1 s the location of the annotation for FJ and AN. Furthermore, in order to mitigate against heavy imbalance, the number of frames extracted was 3 and 8 for FJ and AN respectively. For ‘long’ events E, B and FS where the durations are multiple seconds, frames are extracted every 1 s. The event timestamps are used to extract frame sequences from the appropriate video file using OpenCV [241].

Another issue to be taken in consideration when extracting frames from raw data is the direction the survey is conducted. By convention the start and end annotation

events are made in ascending order of Kilometer Point (KP). KP measures the distance on a map from a given reference point which is typically the platform. The KP is an absolute reference of distance and for a pipeline that connects a platform to shore the KP will increase in that direction. If however, the survey conducted in the reverse direction the KP will be decreasing in the video footage. As a consequence the annotations will be observed in reverse; i.e. end first followed by start. To extract frames between the starting and ending timestamps, the order must be identified prior to extraction, as shown below.

- $Time \uparrow KP \uparrow \implies$ Extraction between the starting and ending timestamp
- $Time \uparrow KP \downarrow \implies$ Extraction between the ending and starting timestamp

It should be highlighted that extracted frames were extracted while maintaining the provenance of the frames (i.e. providing tractability on which frames belonging to which event). The resultant frame distribution is shown in Figure 4.17.

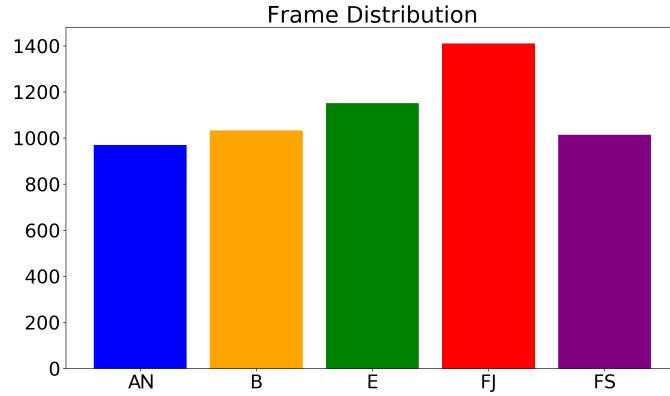


Figure 4.17: Frame distribution

The extracted dataset is four times smaller in size in terms of the number of extracted frames compared to the dataset used in the previous Sections. The label distribution in terms of frames is shown in Figure 4.17 and contains a total 5,579 frames. Furthermore, the event distribution is different across different events as seen in Figure 4.15 where there are 470 FJ events, while there are only 109 AN events. This

dataset has not been manually inspected on per frame basis to identify mislabeled frames. Given the potential noise (incorrect labeling) in the dataset and the smaller size, the overall performance of the model is expected to be lower compared to that presented in Section 4.10. A reminder of the results is presented in Table 4.10. Furthermore, the eliminating the leakage effect will result in a further lowering of performance, but obtaining a more representative indication of expected performance. In order to disentangle the effects from the noise and data set size from the leakage effect, two sets of experiments are conducted: 1) a model is trained following exactly the same procedure as in Section 4.10 i.e., without utilising event information, to identify the effect of noise and data set size 2) a model is trained splitting the training/validation and test datasets based on events (hence eliminating leakage).

Table 4.10: Test Set Performance on the Dataset of the previous Sections

EMR	Precision	Recall	F1-Score
0.910	0.964	0.960	0.962

4.12.1 Generalisation Assessment - Dataset Split with and without Event Traceability)

The raw data from the dataset presented in Section 4.12 was split in training/validation and test using the following two procedures:

- **Frame Splitting:** Randomly selecting frames without taking in consideration the event information.
- **Event Splitting:** Randomly selecting events ensuring that frames from each event can only belong to one of the training/validation or test splits.

The performance obtained from the first method to split the dataset based on Frame Splitting is presented in Table 4.11 and follows the splitting methodology presented in Section 4.7. The same performance evaluation methodology was repeated for the second method of Event Splitting and the results are presented in Table 4.12.

From Table 4.11, the F1-Score between training/validation and test is almost identical. At first glance, this would indicate that the three splits are representative and model training has been performed successfully, however, given that closely proximate frames, could be present in the three splits the generalisation performance could be overestimated. The Event Splitting method (Table 4.12) shows higher drop in the test set which is a more representative performance as there is no information leakage between the different datasets.

Table 4.11: Training, Validation, and Test Set Performance with Frame Splitting

Set	Metrics	EMR	Precision	Recall	F1-Score
Training	Average	0.795	0.925	0.932	0.917
	Std	0.029	0.015	0.006	0.008
Validation	Average	0.786	0.916	0.943	0.918
	Std	0.029	0.017	0.011	0.008
Test	Average	0.770	0.910	0.939	0.912
	Std	0.022	0.016	0.007	0.005

Table 4.12: Training, Validation, and Test Set Performance with Event Splitting

Set	Metrics	EMR	Precision	Recall	F1-Score
Training	Average	0.611	0.838	0.950	0.874
	Std	0.063	0.032	0.008	0.023
Validation	Average	0.547	0.808	0.912	0.839
	Std	0.065	0.036	0.003	0.022
Test	Average	0.502	0.774	0.893	0.810
	Std	0.060	0.038	0.013	0.027

To further explore how the selection of a test set affects the performance when the split of sets is performed with the information of events available, a 5-fold nested CV [209] is performed where 5 different test sets are extracted from the dataset. For each test set, a similar Monte Carlo approach has been performed, and the average performance for each test fold is presented in Table 4.13. The average aggregate F1-score varies from 0.806 to 0.829 while the average EMR varies from 0.48 to 0.52. The performance in all folds is similar and the standard deviation never exceeds 0.05 which makes the hold-out technique without CV representative for the task.

Table 4.13: Average Performance per Test Fold.

Test Fold	Metrics	EMR	Precision	Recall	F1-Score
1	Average	0.485	0.778	0.882	0.806
	Std	0.046	0.038	0.015	0.021
2	Average	0.523	0.798	0.902	0.829
	Std	0.040	0.013	0.013	0.006
3	Average	0.494	0.774	0.899	0.812
	Std	0.047	0.024	0.009	0.016
4	Average	0.521	0.785	0.894	0.818
	Std	0.016	0.008	0.021	0.012
5	Average	0.514	0.805	0.878	0.819
	Std	0.014	0.014	0.019	0.015
Average		0.507	0.788	0.891	0.817

4.13 Conclusions

This Chapter introduces a novel multi-label framework is proposed, which surpasses previous approaches by enabling the detection of concurrent events. This methodology, which is the first of its kind in the field, has significant implications not only for subsea surveys but also for other inspection industries where simultaneous events occur, such as sewer pipe inspection to detect defects [114].

The framework’s ability to accurately classify multiple events in real-time is a notable achievement. By achieving an inference time lower than 40 ms, it demonstrates the potential for efficient and timely event detection, providing valuable decision support in various inspection processes where subsea videos have a frame rate of 25 fps. This real-time capability can lead to improved survey execution speed and potentially reduce the need for extensive offshore personnel.

In addition, the transfer learning approach and the integration of IBN layers highlight the adaptability and effectiveness of the framework. By leveraging pre-trained models and addressing the challenges posed by varying lighting conditions, the framework exhibits robust performance and improved accuracy.

In summary, the contributions of this study include the pioneering multi-label framework for subsea surveys, its ability to detect concurrent events, the potential

for application in other inspection industries, and the real-time capabilities achieved. These advances open new avenues for automated video annotation, streamlining inspection processes, and advancing decision-making capabilities in various domains.

Chapter 5

Subsea Survey Video Classification

5.1 Introduction

Using DL image classification to automatically annotate subsea pipeline video surveys can facilitate the tedious and labour-intensive process, resulting in significant time and cost savings. The previous Chapter explored multi-label classification performance based on individual frame basis and this Chapter extends its application on frame sequences. An approach is to utilise the per frame classifier for video data to predict the annotations for individual frames and obtain a time series of predictions. However, at a typical frame rate of 25 fps, this will result in fluctuations of the predicted outputs. To mitigate against these fluctuations three architectures are considered that utilise the temporal nature of the frame sequences and shown in Figure 5.1. This Chapter represents a pioneering endeavor in the subsea domain as it is the first study to directly operate on videos rather than individual frames for automated annotation in subsea pipeline video surveys.

The first architecture (Figure 5.1 (a)) uses the multi-label classifier proposed in Chapter 4 followed by averaging of the confidence scores between individual frames and per label thresholding to obtain the final annotation. The second architecture (Figure 5.1 (b)) consists of 3D CNN accepting as an input a frame sequence and pro-

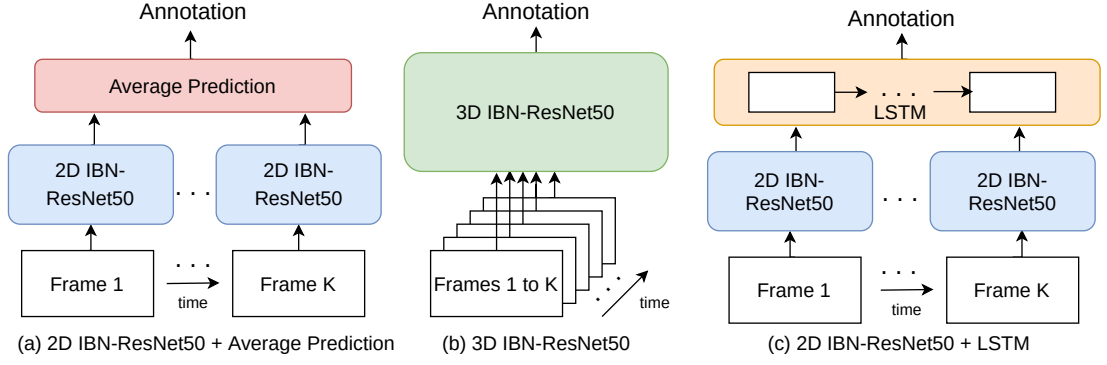


Figure 5.1: Considered architectures

ducing an annotation. The introduction of the 3D IBN-ResNet50 architecture in this study stands as a notable contribution. Leveraging the 3D functionality and incorporating IBN has not been investigated in any work before. Finally, the third architecture (Figure 5.1 (c)) contains a 2D CNN followed by a LSTM, accepting as input a frame sequence and predicting a single annotation. The three models considered here are compared based on their classification performance, model size, training and inference times.

To train these models a new dataset has been created from a survey containing event start and event end annotations. The process of compiling the dataset is described in detail in Section 5.4 and contains the five events of interest. The dataset has high class imbalance and label noise which is mitigated using balanced sampling and label smoothing techniques, spatial and temporal augmentation. This dataset can serve as a valuable resource for subsequent studies in subsea pipeline video analysis, providing a benchmark dataset for model comparison.

5.2 Video Classification

In video classification, frame sequences can be treated as a 3D volume that has spatial (x, y) and temporal (t) dimensions. Although video classification can be considered as aggregation of multiple frame image classification predictions, progress in video classification domain has been hindered by the difficulty to capture temporal context (in addition to spatial), high computational requirements, and datasets that

are limited in scope [242, 243].

Most of the available video datasets focus on human activity recognition. UCF101 [244] and Sports1M [245] with 101 and 487 classes respectively, have been the most popular video recognition datasets and are considered benchmarks for the human activity recognition task. More recently, Kinetics [246] was collected, which is a large-scale high-quality dataset of URL links to approximately 650,000 video clips that covers 700 human action classes, including human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. The largest multi-label video classification dataset is YouTube-8M [247, 248], a benchmark dataset for video understanding, where the main task is to determine the key topical themes of a video. YouTube videos are a good (albeit noisy) source of knowledge for diverse categories such as human activities, but also animals, foods, products, tourist attractions, games, etc. The aforementioned datasets mostly focus on the task of human activity recognition, and thus have limited scope.

One of the early works that utilised DL for video classification was from Karpathy *et al.* [245] who predicted the class of a complete video clip by fusing spatial and temporal information from video frames using 2D CNNs. They considered four approaches to execute the spatio-temporal fusion: (a) single frame fusion, (b) late fusion (c) early fusion and (d) slow fusion. Single frame fusion uses a single architecture that fuses information from all video clip frames at the last stage. Late fusion uses two networks with shared parameters, spaced 15 frames apart, and combines predictions at the end. Early fusion utilises a single network and combining information in the first layer by convolving over 10 frames. Slow fusion also utilises a single network involving fusion at multiple stages, balancing early and late fusion. The authors conclude that slow fusion is more advantageous but they also note that classification of video clips from a single frame provided high performance.

There are two basic types of network architectures to utilise the combined spatial and temporal information namely; utilise a single spatio-temporal model operating directly on 3D volume (3D CNN), or a 2D model extracting spatial features from individual frames followed by a recurrent model operating on the time dimension extracting

temporal features (2D CNN Encoder with RNN Encoder). A third approach is networks that utilise a spatial and temporal stream networks [249]. The spatial network typically utilise a single frame from the image sequence, while the temporal stream utilises Optical Flow (OF). OF is a mathematical estimate of movement in subsequent frames and can be described as densely calculated flow vectors for all pixels. Originally, networks increased performance using OF; however, these are not considered in this work because (a) they provide limited real-time capabilities, (b) OF targets scenarios where the camera is fixed while the scene is moving, while in the subsea survey scenario the scene is static and the camera is moving.

5.2.1 3D CNN

Though CNNs are used mainly for 2D images, they are also used for various forms of input data that can be represented by tensors of any rank. For example, 1D CNNs can be used for text and time series data (1D signal) [250], 2D CNNs for audio and image [251], 3D CNNs for video and volumetric data [252] and 4D CNNs for 3D video and fractional Magnetic Resonance Imaging (fMRI) [253]. With the advancements of low-cost computational power and 3D sensors, 3D computer vision is making its presence in many industrial and user applications such as surveillance, industrial inspection, and medical analysis [254]. 3D model recognition methods can be divided into three categories: (a) volumetric, where the depth of the object is calculated by various methods (in this category, datasets are designed with stereo camera, depth-based camera, or object (model) rendering); (b) view-based descriptors, where the shape of a 3D object is described with a collection of 2D projection; (c) motion-based, where the time interval between frames is used as a third dimension (videos) [255].

The 3D convolution is obtained by convolving a 3D filter kernel and a stack of multiple consecutive frames to produce the 3D cube, as seen in Figure 5.2. The feature maps present in the convolution layer are linked to multiple frames arranged continuously in the previous layer to capture temporal motion-related information. It is important to note that the 3D convolution kernel can select only one type of feature from the patch cuboid, provided that the kernel weights are duplicated across the patch cube. As in

2D CNNs, in a common design scheme of CNNs the number of feature maps increases as the layers increase. Therefore, multiple features are created from lower to higher levels [256].

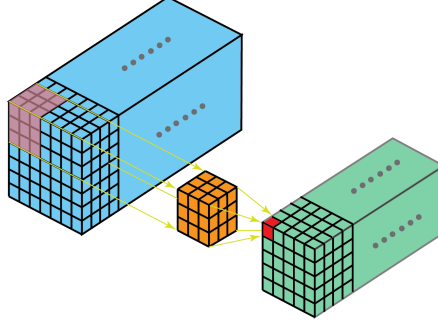


Figure 5.2: 3D convolution [257]

One issue with these models is that they have many more parameters than 2D CNNs because of the additional kernel dimension, and this makes them more computationally demanding to train. In addition, there are limited capabilities for transfer learning given the lack of pre-trained networks; these are widely available in the 2D domain [258].

Tran *et al.* [259] first explored the use of 3D convolution to process UCF-101 video data by adding a third dimension to the filters and without any pre-training. Hara *et al.* [260,261] continued this work by replacing 2D convolutions in common image-based architectures such as ResNet [180] with 3D convolutions, and examined the fine-tuning of Kinetics pre-trained 3D CNNs on UCF-101. Since pre-training on large-scale datasets is an effective way to achieve good performance levels on small datasets, it is expected that the deep 3D ResNets pre-trained on Kinetics would perform well on a relatively small UCF-101. Furthermore, the authors examined whether the transfer learning by deep 3D CNNs works effectively in similar domains. It has shown that the pre-trained 3D networks trained from scratch on the Kinetics dataset are more performant than networks pre-trained on other datasets, mainly due to relatively larger size of the Kinetics dataset. In this Chapter, transfer learning from the Kinetics dataset has not been considered because the action recognition domain presents has no similarities to subsea pipeline survey videos.

The 3D model implemented in this Chapter is inspired by the 3D ResNet of Hara *et*

al., but modifications have been made to the intermediate normalisation layers as described in Section 5.3. The 3D ResNet-50 model is illustrated in Figure 5.3.

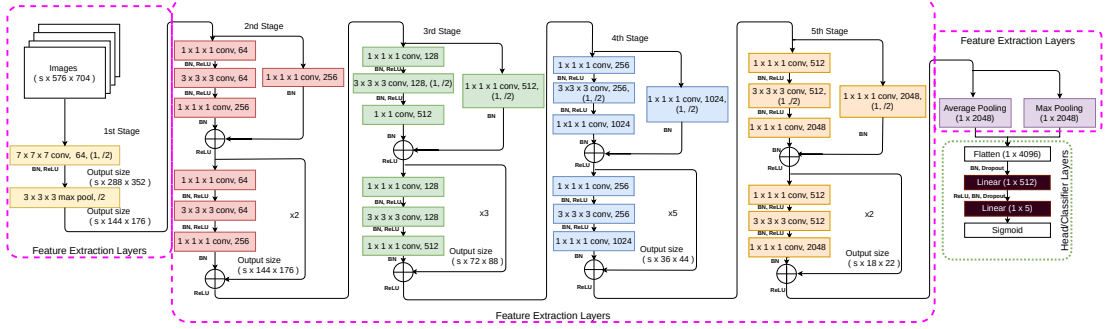


Figure 5.3: Schematic diagram of 3D ResNet-50

5.2.2 2D CNN Encoder with RNN Decoder

Due to the significant progress in the development of convolutional architectures to learn spatial features, many methodologies for videos (frame sequences) revolve around CNNs combined with additional temporal elements. An approach is to add a recurrent module to the CNN model, such as LSTM, which can encode the state and capture temporal ordering and long-term dependencies, as done by Joe Yue *et al.* [262] who implemented a network of stacked LSTMs to process the output of a pre-trained convolutional network and a two-step training approach. Donahue *et al.* [263] builds on the same idea of using LSTM blocks (decoder) after convolution blocks (encoder), but instead uses end-to-end training of the entire architecture, where during training, 16 frame clips are sampled from the video. Varol *et al.* [264] utilised longer clips of 60 frames which increase memory and computation requirements and compensate for the longer temporal range by reducing spatial resolution. Ullah *et al.* [265] proposed an action recognition method by extending the decoder to bidirectional LSTM networks [266]. Inspired by these works, a 2D CNN Encoder with an LSTM decoder is proposed that utilises 16 frames trained end-to-end. Furthermore, the CNN encoder uses the IBN architecture described in Section 5.3.

5.3 Model Architecture

This work compares the performance of three CNN architectures to annotate subsea survey frame sequences, as illustrated in Figure 5.1, along with an evaluation of their size and efficiency in training and testing times. For the 2D model, each frame in the sequence is passed through a 2D CNN to create a prediction for every frame. As the confidence of the network varies from frame to frame this could lead to sporadic FP and FN predictions which are physically not possible during the survey as the inter-frame spacing is too short. To mitigate these sporadic fluctuations, post-prediction filtering can be applied, such as averaging across the sequence to produce one final annotation. For the 3D and the LSTM models, the entire sequence is used as the input of a 3D CNN that outputs a single annotation and inherently reduces fluctuations.

Subsea video footage varies significantly across the pipeline length with various lighting conditions, seabed depths, sand and particle agitation, fouling, vegetation, etc. These attributes result in diverse contrasts and textures of pipeline objects and events of interest. IN has been widely used in the Style Transfer [235–237] literature, to make neural networks invariant to changes in texture and style. Furthermore, BN [267] is widely used to preserve content-related information. IBN [238] combines these two techniques to increase the robustness of the neural network by filtering out sample-specific contrast information while preserving contextual information. This motivates the development of 2D and 3D models that use ResNet-50 [55] and 3D ResNet-50 [260] respectively, both modified to incorporate IBN layers. In this work, the IBN ResNet architecture is expanded to use 3D convolution kernels that can capture both temporal and spatial information. Similarly to 2D IBN-ResNet50 [238], the layers of the model are changed to perform 3D convolution, instance, batch normalisation and pooling. The diagram of the 3D IBN-ResNet block is shown in Figure 5.4.

After the IBN-ResNet encoder, there is an adaptive pooling layer consisting of average and maximum pooling; then the features are flattened and concatenated before being fed into two fully connected (linear) layers. Furthermore, the BN and Dropout [164] layers are introduced between the linear layers to regularise the Head / Classifier. The

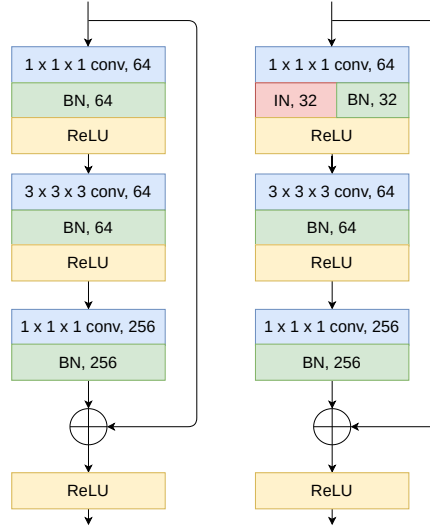


Figure 5.4: 3D ResNet block and 3D IBN-ResNet block

3D ResNet-50 model is illustrated in Figure 5.3.

Another model examined in this work is shown in Figure 5.5. The CNN encoder consists of an IBN-ResNet50, where the final linear layer is removed; that takes a sequence of frames as input and produces spatial 512-feature vectors. The RNN decoder in Figure 5.5 consists of one LSTM with 3 hidden layers of 512 units that capture temporal information. The entire sequence of vectors representing the input images serves as input for the LSTM decoder, which is followed by a linear layer for the final classification. During training, 16-frame sequences are sampled from video clips as described in Section 5.5.1. The architecture is trained end-to-end, and the training procedure is the same as for the 3D CNN model. The drawback of the LSTM decoder architecture is that it requires NAS to adjust the hyperparameters of the LSTM module, such as the number of hidden layers and the number of nodes in those layers.

5.4 Dataset Exploration

The raw dataset provided by N-Sea and presented in Section 4.12 contains two surveys from 2012 and 2016; however, both surveys contained only single event annotations instead of annotations with start and end (see Figure 4.16). This is particularly prob-

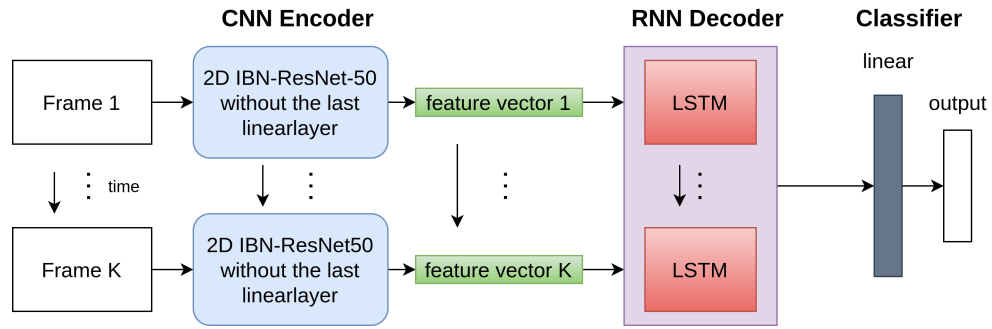


Figure 5.5: IBN-ResNet50 Encoder with LSTM Decoder and Classifier

lematic for video classification when blocks of frames are required. To mitigate this issue, N-Sea was asked to re-process the 2012 survey to include start and end annotations. Figure 5.6 shows the distribution of all events contained in the 2012 survey. Some events are labelled using different terminology; for example, Suspensions are FS and also contain other events such as Sea Marker, Ground Bags, which are not considered in this thesis. Most importantly, ‘short’ events have start and end event annotations, i.e. FJ and AN.

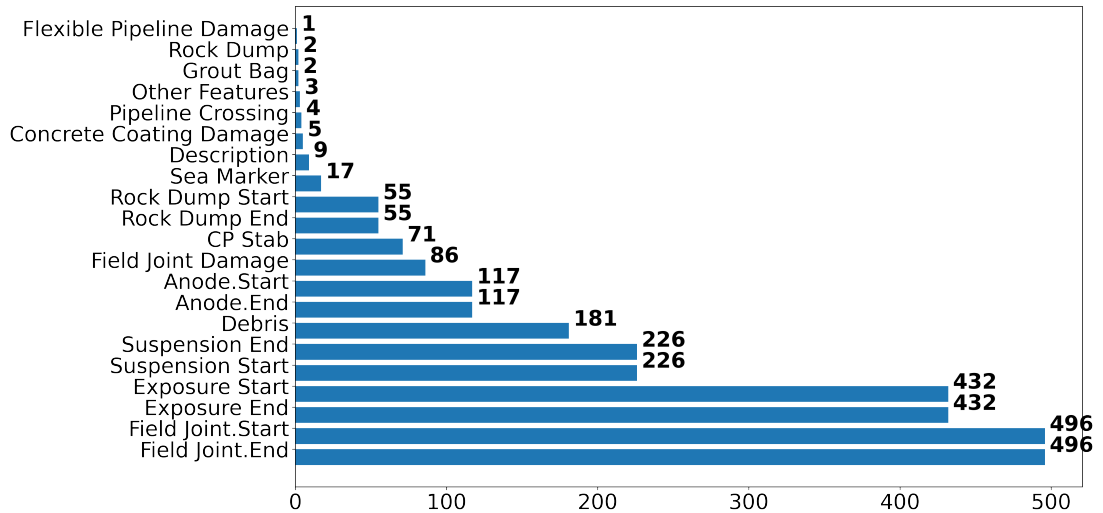


Figure 5.6: Distribution of events from the 2012 survey

The histograms of event duration in seconds are shown in Figure 5.7. The duration of ‘long’ events tends to be high, as the name states, and if all frames are to be extracted, this will dwarf the frame counts of ‘short’ events. If all frames from the long

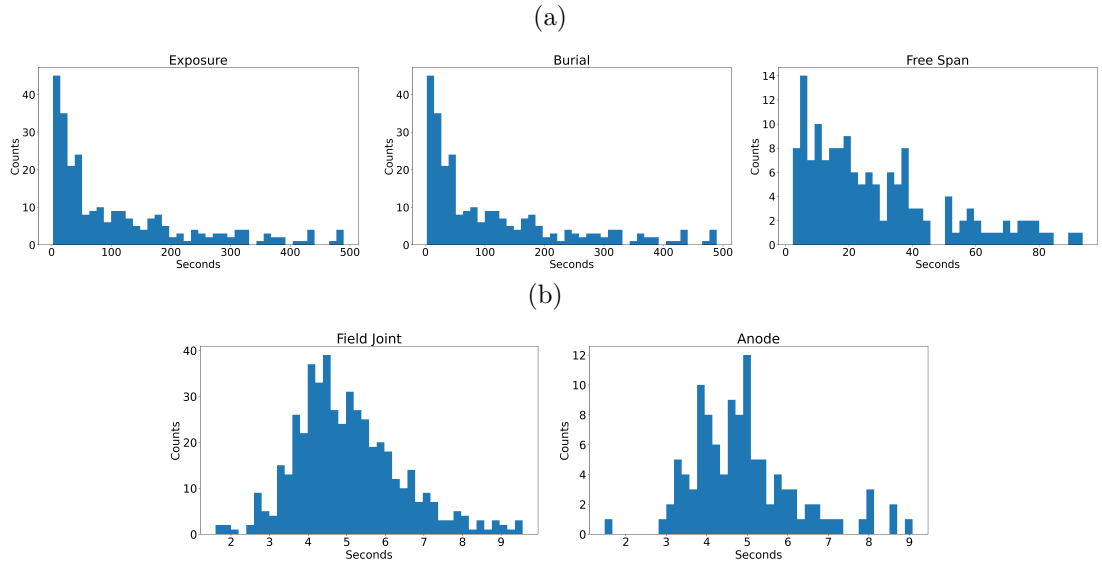


Figure 5.7: Histograms of the duration of events of interest in seconds; E, B, FS, FJ and AN.

events are extracted, this would create a highly imbalanced dataset with the events of Exposure, Burial and Free Span being over-represented compared with the ‘short’ events of Anode and Field Joint which last for an average of 100 frames. For that reason, frame extractions of ‘long’ occurs only for the first 400 frames; this value was selected after some initial experimentation during the dataset curation phase.

Furthermore, since the event frequency of E and B is high, not all events were used. For FJ and AN events are excluded if their duration is extremely short. Finally, for ‘long’ events, frames are extracted from the start annotation until the stop annotation or until another event becomes present; in which case extraction pauses and resumes after that event.

After extraction, the distribution of video clips (frame sequences) is shown in Figure 5.8. Hence, the data set contains 1,114 video clips in total, consisting of 105 clips of B and 979 exposure clips, of which 120 clips are single E, 477 clips contain FJ, 109 clips contain AN, and 273 clips contain FS. Example video clips of 16 frames are shown in Figure 5.9.

Furthermore, the histograms of frames per clip for the five events of interest are shown in Figure 5.10. The video clips of the five events have varying lengths, with 50

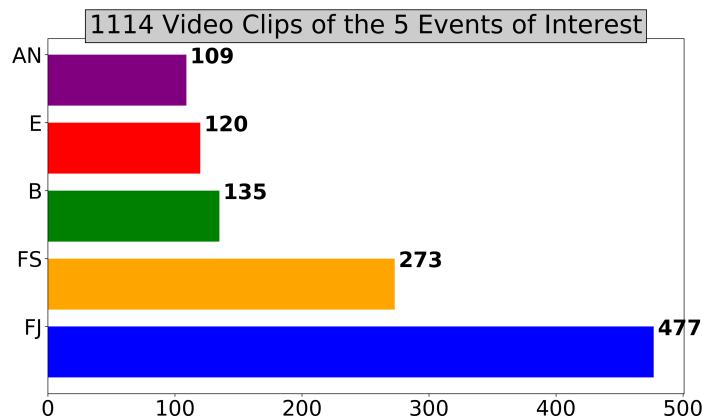


Figure 5.8: Distribution of video clips of the five events of interest

and 400 frames being the minimum and maximum durations, respectively. The ‘short’ events of FJ and AN last an average of 3-4 seconds (75-100 frames), while the events of B, E and FS last up to 16 seconds (400 frames). ‘Short’ events add some noise to the dataset because of the start and end timestamp annotations provided; examples were observed for AN events lasting approximately 100 frames, where the anode is only visible in the middle 70-80 frames. Extra frames in which the anode is not visible at the start and end of the event are considered to be the transition period.

The combination of the distribution of the video clip events with the frame histograms per event is shown in Figure 5.11, which shows the total frame distribution of the five events in the dataset. The entire dataset contains 227,334 frames. It is clear that AN is a minority class, while FS is a majority class. The other 3 classes (FJ, B, E) have a similar number of frames. As can be seen in Figure 5.11 there are far fewer frames containing the label AN, and it requires careful consideration to address the label imbalance, which is explored in more detail in Section 5.5.

5.5 Mitigating Label Imbalance

As illustrated in Figure 5.11, the dataset used in this work presents a high label imbalance. The two main approaches to mitigate the imbalance are data-level methods that operate on the training set to change the class distribution and cost-sensitive learning methods [268]. The latter methods keep the training dataset unchanged and adjust

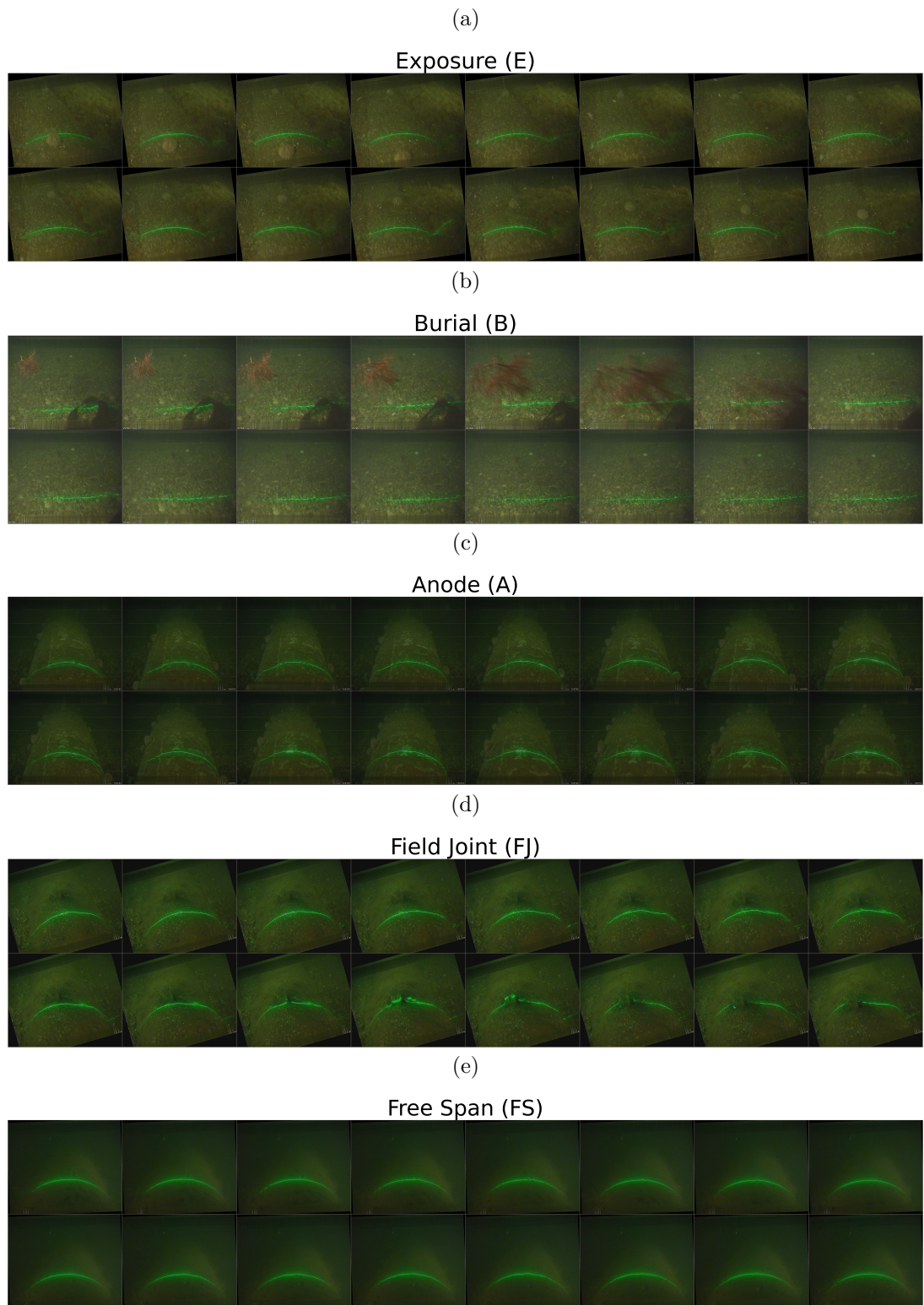


Figure 5.9: Frame sequences examples of the five events of interest

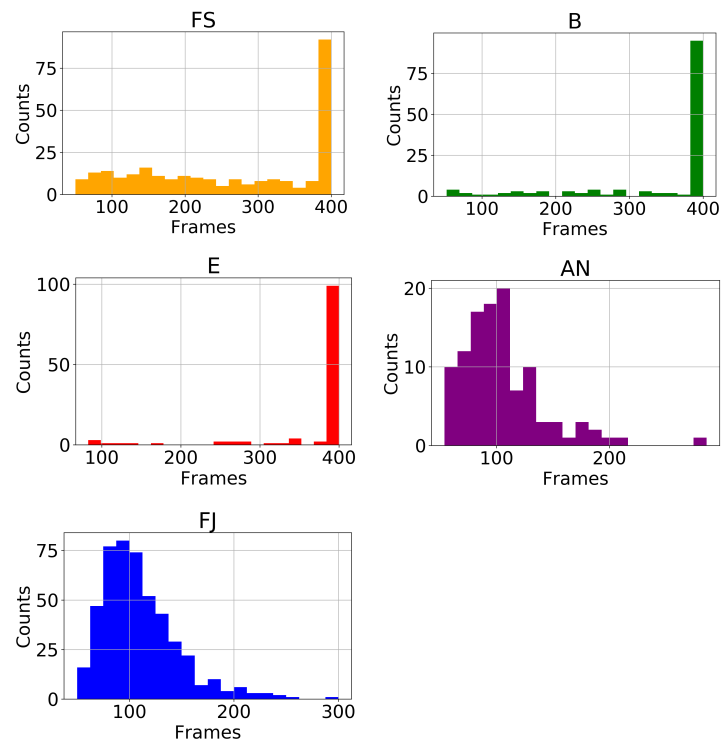


Figure 5.10: Histograms of the collected clips of events in frames

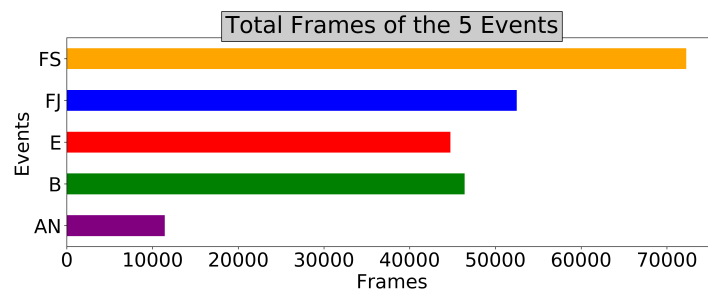


Figure 5.11: Total frames of the five events of interest

different class weights in the loss function. The data-level method has been used in this Chapter to tackle the label imbalance, but additional experimentation with cost-sensitive learning is presented in Section 5.9.4.

Data resampling is commonly used in ML works to balance datasets, by oversampling minority classes and undersampling majority classes [269], an example is shown in Figure 5.12. By altering the relative frequencies of the examples, resampling of the dataset enables training that does not underperform for minority classes. Oversampling adds repeated samples from minority classes, which could cause the model to overfit. To address this issue, oversampling is combined with image augmentation techniques.

In this Chapter, a balanced sampler has been developed that equalise the number of samples per event (label) in a mini batch. For example, for a mini batch size of 10, the mini batch contains 2 samples from each of the 5 events. A comparison between unbalanced, balanced sampling and cost-sensitive learning is provided in Section 5.9.4.

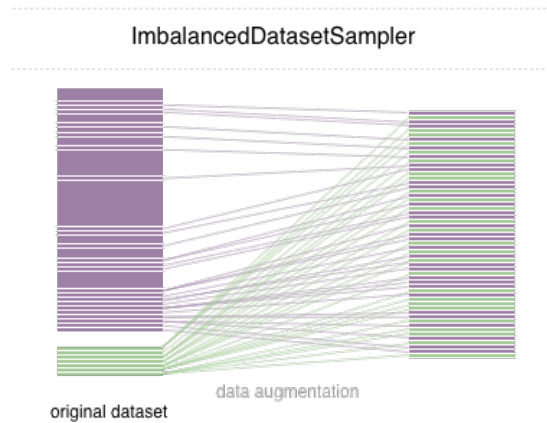


Figure 5.12: Label resampling [270]

5.5.1 Sampling Sequences

Initially, events are segmented to create video clips of 50 frames starting at a location within the duration of the event. In accordance with the frame rate of the subsea survey videos which 25 frames per second 50 frames correspond to 2 seconds. This number is chosen based on the intuition that 2 seconds are sufficient to detect the presence of an event in a subsea survey video, that has derived from manually inspecting some of

the videos provided by N-Sea. Furthermore, this number is supported by the video clip duration histograms of Figure 5.10 because the goal is to sample more than one examples for every video clip to increase the training variability.

Since there is a high volume of video footage for ‘long’ events, the starting locations of the 50-frame video clips are in fixed steps of 8 frames. An example is shown in Figure 5.13 where the blue dots represent the starting location of the 50-frame video clips. Initially an identical approach was used for ‘short’ events but manual inspection indicated that the events are usually visible in the middle of the video clip, but sometimes not present in the first few and last few frames. Hence, this could lead to a significant number of 50-frame video clips with no visible events in them. To minimise the effect such occurrences, the starting sampling location for ‘short’ event videos, has been biased towards the centre of the event using a normal distribution (with μ at the middle of the event and $\sigma = 15$); i.e. starting location in the middle of the event are more likely as shown in Figure 5.14. In the figure, the blue dots represent the starting location of the 50-frame video clips, and the purple line illustrates the bias applied when sampling starting locations. Note that in all cases, if the starting location is less than 50 frames before the end of the event video, these observations are discarded because there is no adequate number of frames to complete the video clip.

This sampling procedure results in segmenting the events and obtaining sequences of fixed length of 50 frames. Although the considered NN models can be directly trained on 50-frame sequences, there are GPU memory constraints. With current GPUs, a reasonable trade-off is to utilise sequences with 16 frames [263]. Reducing the 50-frame sequences to 16-frame sequences is achieved through downsampling during temporal augmentations.

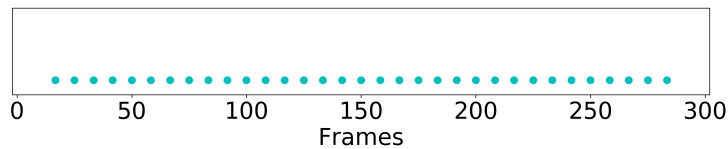


Figure 5.13: Example of sampling frames and sequences from ‘long’ events of E, B, and FS

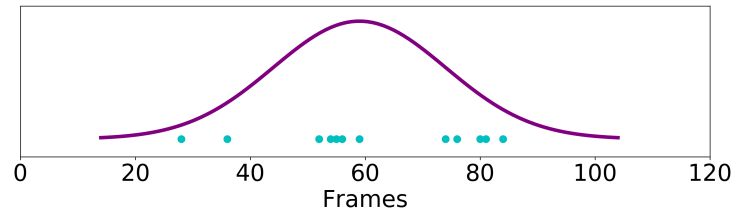


Figure 5.14: Example of sampling frames and sequences from ‘short’ events of AN and FJ

5.5.2 Video Augmentation

To produce robust video recognition models, which generalise well, video augmentation is used. Two categories of video augmentation are considered: (a) spatial, where image augmentation techniques are applied to the frames, and (b) temporal, where the time dimension is altered in different ways. Spatial and temporal augmentations are used to create models that learn spatially and temporally invariant features [271], respectively. The augmentations utilised are executed using the Vidaug [272] library.

To address the variability of ROV speed, two augmentation techniques are used: Inverse Order and Temporal Elastic Transformation which are presented in Table 5.1. When Inverse Order is used, the order of the sequence is inverted to address the issue of ROV changing its direction of travel during a survey. When Temporal Elastic Transformation is used, the time dimension stretches (normal) or shrinks (inverse) a video at the beginning, end, or middle part to simulate the ROV speed change. To showcase an example of Temporal Elastic Transformation a video of a person performing a push-up, consisting of 72 frames, is shown in Figure 5.15, as it can be more clearly observed compared to the subsea pipeline data.

Table 5.1: Temporal Augmentations: 1 of those is selected randomly if not normal downsampling

Augmentations	Parameters	Probability
Inverse Order		Always if selected
Temporal Elastic Transformation	Normal or Inverse	Random

Spatial augmentations are applied to create a more diverse dataset and tackle subsea

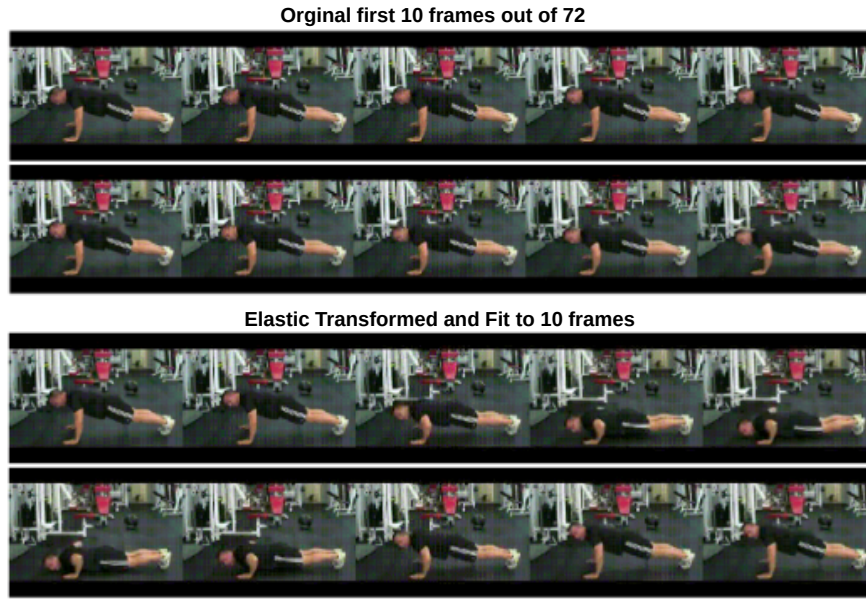


Figure 5.15: Example of a Temporal Elastic Transformation, where the middle part of the video (lowest level of the push-up) has been stretched and fit to 10 frames.

image challenges such as camera movement, varying lighting conditions, contrast, and blurriness from the seabed sand agitation. During training, every sample is altered with up to three of the transforms listed in Table 5.2; Random Rotation of the clip by a maximum of 20 degrees angle to address the ROV camera rotation, Gaussian Blur blurs the clip by filtering it with a Gaussian kernel to address the blurriness that the sand from the seabed creates in some clips, Horizontal Flipping, to tackle overfitting on the layout words that the inspection software writes to the frames, and ensuring that a mirror of any event should also be detected correctly, Increasing and decreasing the contrast and brightness, to address the varying lighting conditions that subsea surveys contain, by adding or multiplying pixel intensities with given value, and finally Elastic Transformation. Elastic Transformation creates local distortion in an image, by moving pixels using distortion fields [273]; the parameter α is the strength of the distortion field (higher values mean more ‘movement’ of pixels) and the parameter σ is the standard deviation of the Gaussian kernel used to smooth the distortion fields. An example is shown in Figure 5.16.

Example subsea frame sequences used in this work are illustrated in Figure 5.17.

Table 5.2: Spatial Augmentations: Up to 3 of those are selected randomly per clip

Augmentations	Parameters	Probability
Random Rotation	maximum angle: 20 degrees	Random
Gaussian Blur	σ of the Gaussian kernel: 0.1	Always if selected
Horizontal Flipping		Always if selected
Add	-15 or +15	Random
Multiply	1.5 or 0.75	Random
Elastic Transformation	$\alpha = 0.2$, $\sigma = 0.1$	Always if selected

It is important to note that spatial transforms are not used during the validation and testing phases.

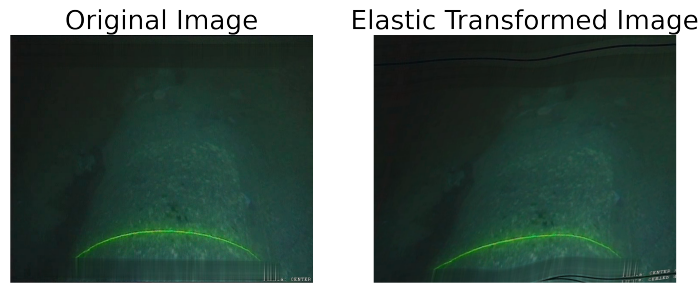


Figure 5.16: Example of an Elastic Transformation

5.6 Training Details

The Chapter presents a 2D CNN model and two spatio-temporal models. The two spatio-temporal models are trained and tested on 16-frame sequences. The 2D CNN is trained on single frames but it is tested on 16-frame sequences by averaging the individual frame predictions. Both single frames and frame sequences are labelled using a multi-label annotation approach because the events recorded during the pipeline survey are not mutually exclusive, in a similar fashion to Chapter 4.

After balanced sampling the number of observations in the training varies between the 2D CNN and the spatio-temporal models. The difference is caused from the fact that after sampling, the sampled sequences could exceed the total length of the video clip, and thus are discarded. In the experiments presented here, the 2D CNN is trained with 16,149 samples, the 3D CNN is trained with 14,876 and 2D-LSTM with 14,923

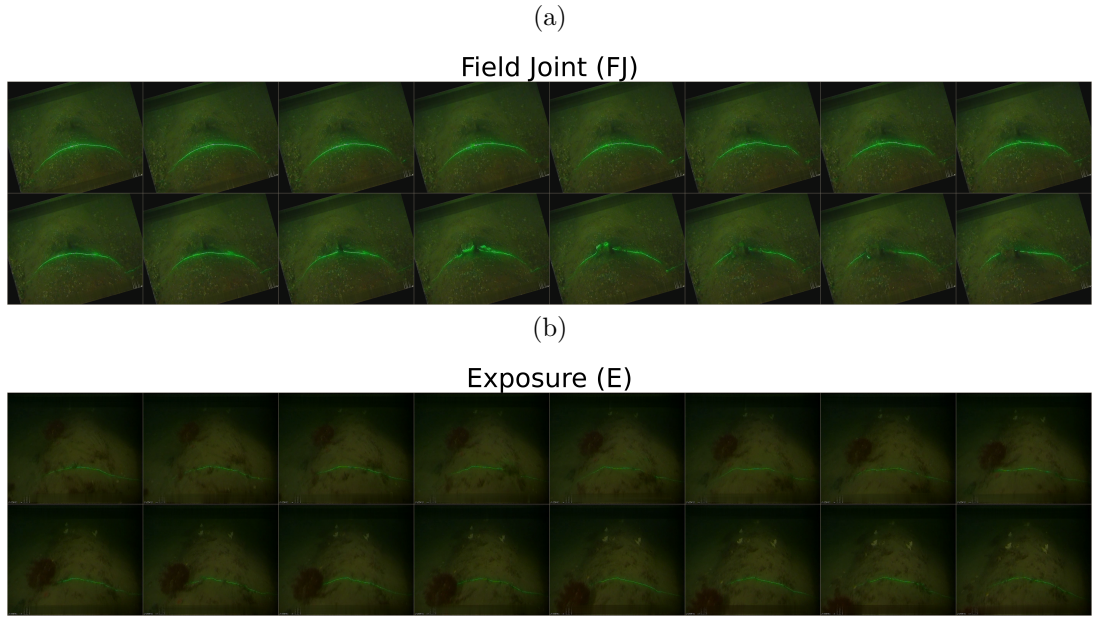


Figure 5.17: Examples of Augmentations. The first sequence (a) shows a rotation-augmented Field Joint, while the second (b) is a horizontally flipped Exposure frame sequence

16-frame sequences. All the remaining training details such as optimiser, weight decay and etc. are identical to that described in Section 3.12.4.

The loss function adopted is BCE with Focal Loss (FL) [274] which is further described in Section 5.6.1 and the models are further regularised by label smoothing [165] as described in Section 3.7.3 with specific details described in Section 5.6.2. The final models are saved based on the lowest validation loss. The 2D CNN is trained for 30 epochs and the model resulting in the lowest validation loss is acquired on epoch 17, whereas the 3D CNN and the 2D-LSTM are trained for 50 epochs and the best weights are acquired on epochs 27 and 22, respectively. The decision to train the 2D CNN for fewer epochs compared to the 3D CNN and 2D-LSTM models is based on the fact that the 2D CNN benefits from ImageNet pretraining which leads to faster convergence. The numbers of epochs were also manually explored during the development stage.

The 2D CNN converges faster due to starting with ImageNet weights, while the weights of the 3D are randomly initialised. Furthermore, the faster convergence can be attributed to the fact that both models utilise approximately the same number

of observations, but the 2D CNN has approximately half the parameters of the 3D counterpart. All models were trained on NVIDIA A40 GPUs with mini batch sizes of 40 and 10 for the 2D and the spatio-temporal models, respectively.

The validation and test sets are identical for all models and consist of 4,658 and 4,746 samples, respectively.

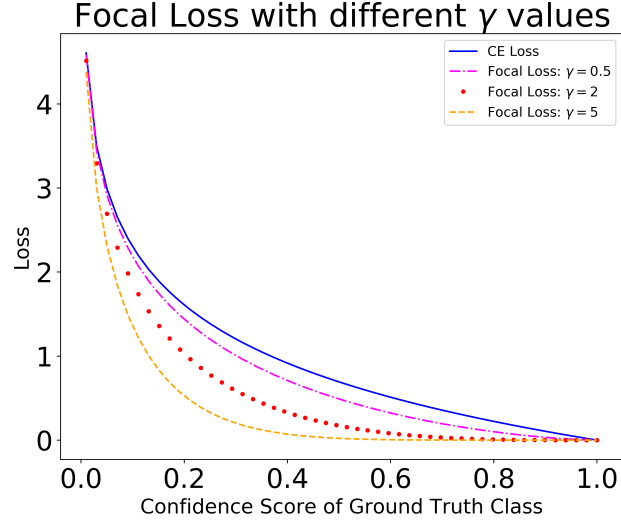
5.6.1 Focal Loss

FL was introduced by Lin *et al.* [274] to improve object detection when there is an extreme imbalance between the foreground and background classes. FL is a CE loss that weighs the contribution of each sample to the loss depending on the classification error. If a sample has already been correctly classified by CNN, its contribution to the loss decreases logarithmically. The rationale is that when the loss is made to implicitly focus on these problematic classes, the class imbalance can be addressed [275]. As a consequence, more attention is paid to hard, misclassified samples than to easy samples. The level of attention is controlled through the parameter γ , which modulates the multiplier strength to the standard loss. FL is defined as:

$$FL(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{i=1}^C [(1 - \hat{y}_i)^\gamma \cdot y_i \cdot \log(\hat{y}_i) + \hat{y}_i^\gamma \cdot (1 - y_i) \cdot \log(1 - \hat{y}_i)], \text{ with } \gamma \geq 0 \quad (5.1)$$

where C is the number of labels. When $\gamma = 0$, then FL becomes BCE (see Equation 4.1). When a sample is misclassified, \hat{y}_i (which represents the estimated confidence score of the model for the label i) is low and the multiplier $(1 - \hat{y}_i)$ is close to 1, so the loss remains unaffected. As $\hat{y}_i \rightarrow 1$, the multiplier approaches 0, and the loss for well-classified examples is down-weighted. The effect of the parameter γ on the loss is shown in Figure 5.18.

In this work, γ is set to 2. Additional experiments with other values of the hyperparameter γ can be found in Section 5.9.2.

Figure 5.18: Focal Loss for different γ values

5.6.2 Label Smoothing

Hard or noisy samples in the dataset could exist because of incorrect manual annotations, camera and ROV movement, as well as fish and vegetation. A method used in this work to address the noise in labels is label smoothing. Label smoothing is a regularisation technique for classification problems to prevent the model from predicting labels too confidently during training and consequently generalising poorly; this is also used when there is a wrong label assignment in the dataset [165, 166, 276]. The technique is implemented by replacing an one-hot encoded label vector y_{hot} which is discrete (either 0 or 1) with:

$$y_{ls} = (1 - a) \cdot y_{hot} + a/K \quad (5.2)$$

where $K = 5$ is the number of labels and a is a hyperparameter that determines the amount of smoothing. If $a = 0$, the original one-hot encoded y_{hot} is obtained, while if $a = 1$, the label scores follow the uniform distribution. In this work, after experimentation, a was selected as 0.1. An additional experiment without label smoothing is presented in Section 3.7.3.

5.7 Performance Evaluation Metrics

The performance evaluation is executed on a stratified split of the dataset; 60% of the events in the two surveys for training, 20% for validation and 20% for testing. This ensures that frames from a particular event only appear in one of these sets, and thus there is no leakage of information from one set to another. Following this split, 50-frame video clips are sampled and augmented for the training set; this is repeated for every mini-batch. For validation and test sets the sampling occurs only once for all models and no augmentations are applied.

The validation set is used to ensure there is no overfitting and to provide a test platform to search for an optimal confidence threshold for each label using PR curves [141, 225]. For every 16-frame input, the two spatio-temporal models produce one single output. In the case of 2D CNN, the threshold search occurs after averaging the 16 confidence score predictions. The holdout test set, or unseen data, is used to assess the final performance of the model after training and selecting optimal thresholds.

In this application, the evaluation metrics used for multi-label classification follow the methodology described in Sections 3.12.7 and 4.8, and it is identical to that used in Chapter 4. The only difference is that metrics are reported using the ‘Samples’ averaging instead of the ‘Micro’ averaging. This provides more representative results when there is an imbalance in labels [277]. The ‘Samples’ averaging calculates the metrics first for each label and then averaged, compared to ‘Micro’, which calculates metrics globally by counting the total TP, FN, and FP.

5.8 Results

The validation set consists of 4,669 samples and is used to find the optimal thresholds for the five labels using PR curves that balance the trade-off between Precision and Recall as discussed in Section 4.9. After these thresholds are set, the models are evaluated on a test set that contains 4,729 samples. The per-class performance, along with the corresponding thresholds for the 2D, 3D IBN-ResNet50 and 2D-LSTM models, are presented in Table 5.3, Table 5.4 and Table 5.5, respectively. For the 2D model, the

results of Table 5.3 indicate that for the events of E, FJ, FS, the classifier performs better than for the events of AN, B. The lowest performance is observed for the AN events and this can be justified considering that the AN class is the minority class. The B events are also mutually exclusive to E and all other classes, making it more difficult to predict.

Table 5.3: 2d Label-based Test Set Metrics

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.770	0.938	0.964	0.959	0.961
Burial	0.230	0.938	0.837	0.851	0.844
Field Joint	0.512	0.973	0.946	0.912	0.929
Anode	0.421	0.947	0.795	0.859	0.826
Free Span	0.813	0.991	1.000	0.968	0.983

For both the 3D IBN-ResNet50 and 2D-LSTM models, in Table 5.4 and Table 5.5, it can be seen that the models achieve similar performance for E, FS and B with each other and compared to the 2D model, however, for the events of AN and FJ the performance drops. This can be attributed to the fact that these events are more sensitive to the transition from exposure to the actual event that creates confusion (noise) in the dataset, leading to higher FP and FN, while averaging helps the 2D model mitigate this issue. In conclusion, for events of longer duration (E, B, FS) the 3D models perform better than for ‘short’ events such as AN and FJ.

For the inspection process of industrial subsea pipelines, it is more important not to miss any events and more forgiving to false positives. This can be justified considering that the survey results were again checked by the QC team. If necessary, high sensitivity can be obtained by decreasing the threshold for events of interest (e.g. AN).

Table 5.4: 3D Label-Based Test Set Metrics

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.657	0.935	0.943	0.978	0.960
Burial	0.280	0.933	0.865	0.781	0.821
Field Joint	0.424	0.893	0.714	0.748	0.731
Anode	0.280	0.870	0.562	0.492	0.525
Free Span	0.171	0.981	0.954	0.981	0.968

In general, the 2D model outperforms its spatio-temporal counterparts. All models

Table 5.5: 2D-LSTM Label-Based Test Set Metrics

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.532	0.892	0.933	0.932	0.932
Burial	0.441	0.891	0.714	0.736	0.725
Field Joint	0.607	0.912	0.769	0.775	0.772
Anode	0.661	0.896	0.689	0.531	0.600
Free Span	0.036	0.992	0.993	0.980	0.986

have been trained with a similar amount of samples, but the 3D models have a higher number of parameters. A larger dataset could help mitigate this issue, and this is confirmed in the additional experimentation presented in Section 5.9.

The aggregate metrics for all models are shown in Table 5.6 while the number of model parameters, training, and inference time are presented in Table 5.7. The 2D, 3D CNN and 2D-LSTM models are compared on the basis of their aggregated EMR, Precision, Recall, and F1-Score, training, and inference times. The aggregated evaluation metrics indicate that the 2D model followed by averaging outperforms the 2 spatio-temporal models overall. In addition, training the 2D CNN is significantly faster due to its fewer parameters (almost half as the 3D CNN) and the use of pre-trained ImageNet weights as initialisation makes the convergence faster. The inference time is comparable and similar for all models, while the GPU memory used during inference is less than 3 GB, which is acceptable for commodity GPU hardware. Consequently, all models could be deployed on a ROV with an embedded GPU system.

Table 5.6: Aggregated performance metrics for 2D, 3D CNN and 2D LSTM

Models	EMR	Precision	Recall	F1-Score
2D IBN-ResNet50-Avg	0.866	0.920	0.917	0.913
3D IBN-ResNet50	0.725	0.861	0.858	0.855
2D IBN-ResNet50-LSTM	0.743	0.852	0.838	0.836

Table 5.7: Parameters and computation complexity of 2D and 3D Models

Models	Parameters	Training Time (2 GPUs)	Inference Time (1 GPU)
2D IBN-ResNet50-Avg	25.6 M	5 mins per epoch	185 \pm 15 ms per 16 frames
3D IBN-ResNet50	45.5 M	95 mins per epoch	194 \pm 15 ms per 16 frames
2D IBN-ResNet50-LSTM	31.9 M	50 mins per epoch	152 \pm 10 ms per 16 frames

5.8.1 Effect of dataset size in 3D model

The results presented in the previous section indicate that the performance of the 3D model is not as high as that achieved by the 2D model. The hypothesis is that the model contains a large number of parameters and would benefit if a larger dataset was available. To examine whether a larger dataset would be beneficial for a 3D mode, another experiment has been carried out using 29,347 training frame sequences and compared with the results of the model trained previously in Section 5.8 with 14,876 samples. The test set performance metrics are shown in Table 5.8. Note that in this experiment only the training set has increased in size, while the validation and test sets used are those extracted in the previous experiments (i.e. equal number of samples and identical observations in each set). Figure 5.19 provides a comparison of F1-Score between the original and the larger dataset. When these test results are compared with those in Table 5.4, some improvements can be seen for AN, FJ and FS classes and an increase in the aggregate F1-Score from 0.855 to 0.868. However, the increase is not significant, as the data of the new larger dataset are sampled from the same clips as the original dataset, and therefore they do not add further new information and variety. For comparison, other large 3D models are trained on a very large dataset such as the Kinetics [278] benchmark, consisting of 650,000 clips. Therefore, further improvements are expected to be possible if more clips are added to the dataset.

Table 5.8: Performance of a 3D model trained on a larger training dataset of 29,347 samples

Event	Threshold	EMR	Precision	Recall	F1-Score
Exposure	0.533	0.897	0.906	0.956	0.930
Burial	0.370	0.894	0.835	0.776	0.805
Field Joint	0.424	0.910	0.715	0.788	0.750
Anode	0.297	0.899	0.640	0.526	0.577
Free Span	0.482	0.989	0.966	0.990	0.978
Aggregate		0.750	0.869	0.874	0.868

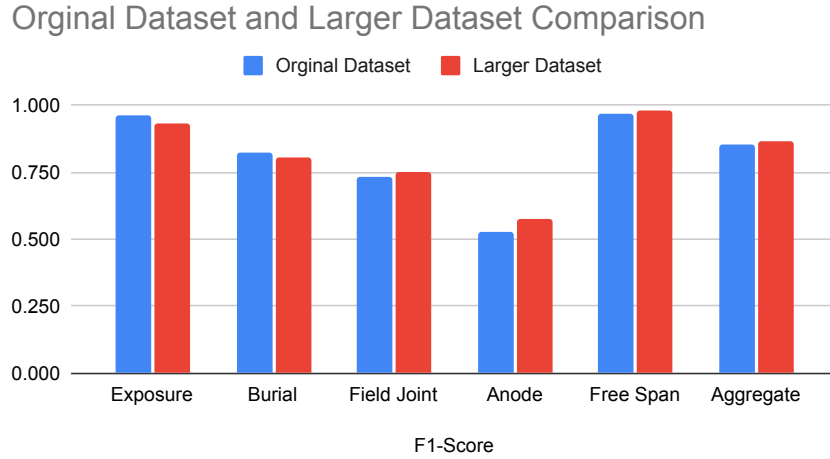


Figure 5.19: F1-Score for the Original and Larger Dataset

5.9 Additional Experiments

The same training and evaluation procedure has been followed to carry out further experimentation for the 2D CNN model, namely: Test Time Augmentation (TTA) [279], FL hyperparameter γ , label smoothing, cost-sensitive learning. In the following sections, the results of the test set metrics are provided in Tables along with bar chart visualisations.

5.9.1 Test Time Augmentation

Additional experiments have been performed on the 2D model, as it allows for faster training due to the fewer parameters needed. Image augmentation can also be applied when making predictions with a trained model to allow the model to make predictions for multiple different versions of each image in the test dataset. The predictions of the augmented images are then averaged, and this may lead to better predictive performance [280]. Table 5.9 shows the results acquired when TTA is applied with 5 augmented test folds. The techniques used are the same as those used during the training. Five augmented test folds were created, and the subsequent predictions were averaged. The results present a performance similar to that of the one-test fold and showcase the robustness of the model as predictions are made in the augmented sam-

ples. The reason why the performance did not increase is that the model is saved based on the lowest validation score, and the validation set is not augmented.

Table 5.9: 2d Label-Based with Test Time Augmentation

Event	Threshold	EMR	Precision	Recall	F1-Score
Exposure	0.770	0.922	0.938	0.954	0.946
Burial	0.230	0.922	0.877	0.840	0.858
Field Joint	0.512	0.972	0.925	0.915	0.920
Anode	0.421	0.941	0.736	0.856	0.792
Free Span	0.813	0.993	1.000	0.973	0.986
Aggregate		0.850	0.900	0.903	0.897

5.9.2 Focal Loss: hyperparameter gamma

Additional experiments carried out using three different γ values for the FL (see Equation 5.1). Table 5.10, Table 5.11 and Table 5.12 show the results for γ of 0, 0.5 and 5 respectively (note that the results for $\gamma = 2$ are presented in Table 5.3). The F1-Score of the models trained with FL and different γ values is also visualised in the bar chart of Figure 5.20. From these tables and the bar chart, it can be seen that the highest aggregate F1-Score is achieved for $\gamma = 2$. It is also clear that the F1-Score of the minority class AN improves as the γ increases; however, for the majority classes, for example E, the performance decreases. Evidently, when BCE loss is used ($\gamma = 0$) the performance of the minority labels is the poorest. On the other hand, when $\gamma = 5$ the performance of E and FS (majority labels) deteriorates. Finally, 5.21 illustrates how the use of FL with higher γ values improves the balance between Precision and Recall for the minority label Anode.

Table 5.10: 2D Test Set Metrics using BCE or FL with $\gamma = 0$

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.616	0.933	0.950	0.968	0.958
Burial	0.375	0.932	0.855	0.788	0.820
Field Joint	0.411	0.957	0.896	0.879	0.888
Anode	0.424	0.943	0.881	0.707	0.784
Free Span	0.552	0.999	1.000	0.996	0.998
Aggregate		0.846	0.917	0.900	0.902

Table 5.11: 2D Test Set Metrics using FL with $\gamma = 0.5$

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.706	0.934	0.956	0.953	0.954
Burial	0.295	0.935	0.881	0.888	0.885
Field Joint	0.519	0.960	0.874	0.897	0.885
Anode	0.477	0.951	0.785	0.864	0.822
Free Span	0.732	0.993	1.000	0.972	0.986
Aggregate		0.861	0.911	0.916	0.908

Table 5.12: 2D Test Set Metrics using FL with $\gamma = 5$

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.502	0.912	0.929	0.949	0.939
Burial	0.483	0.913	0.862	0.823	0.842
Field Joint	0.678	0.969	0.935	0.885	0.909
Anode	0.278	0.955	0.812	0.859	0.835
Free Span	0.831	0.993	1.000	0.975	0.987
Aggregate		0.847	0.896	0.893	0.890

F1-Scores of models trained with different Focal Loss gamma values

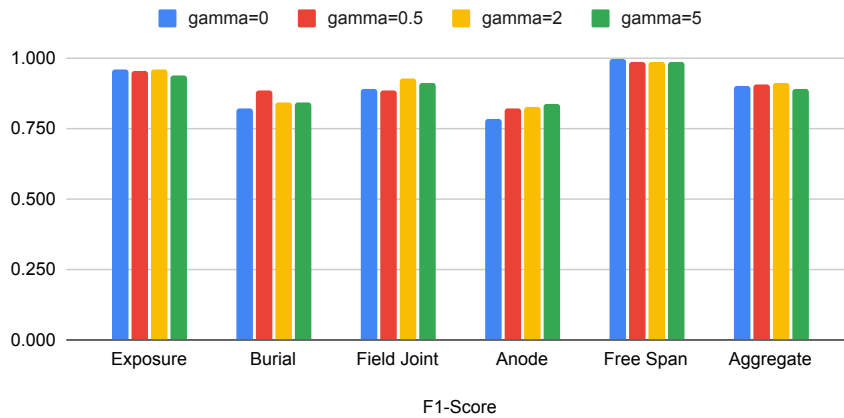


Figure 5.20: F1-Score of models trained with different Focal Loss gamma values

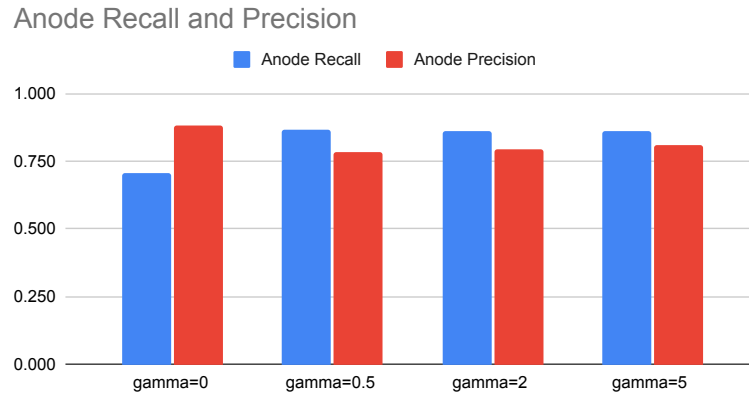


Figure 5.21: Anode Recall and Precision of models trained with different Focal Loss gamma values

5.9.3 Label Smoothing

To demonstrate how label smoothing affects performance, two experiments were carried out with $a = 0$ (no label smoothing) and $a = 0.4$ in Equation 5.2. Figure 5.22 presents the F1-Score acquired with these models on the test set and shows that for $a = 0.1$ the best aggregate performance is acquired in terms of F1-Score, which evaluates the selection of this value. One interesting observation is that the performance of the minority class AN improves when $a = 0.4$.

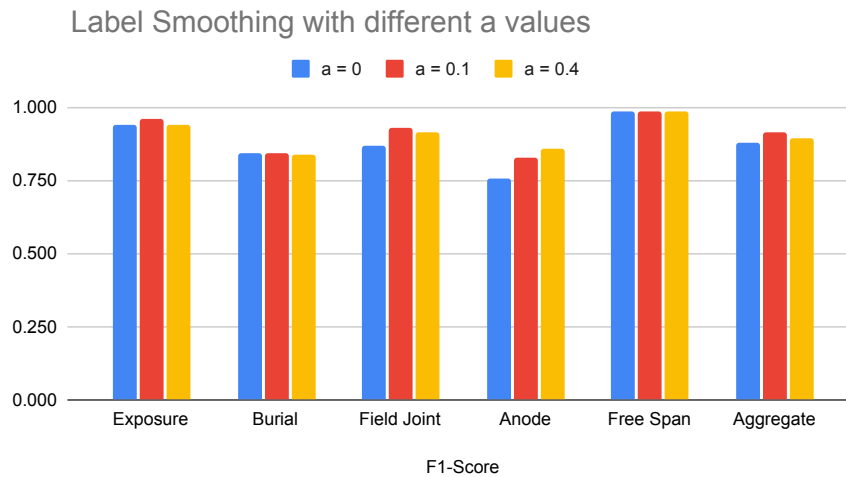


Figure 5.22: F1-Score of models trained with different label smoothing

5.9.4 Cost-Sensitive Learning

Cost-sensitive learning for imbalanced classification (long-tail datasets) focuses on assigning weights to the loss function of a CNN model [281, 282]. There are two options of assigning these weights, the first is to multiply the loss function with a weight-based cardinality of the classes and moderate the imbalance effect denoted by w_c in Equation 5.3. More specifically, the weights of the majority class are lower, whereas for the minority classes are higher, leading to balancing their contributions to the total loss.

The second is to trade off Recall and Precision by assigning weights to positive examples, denoted by p_c as in Equation 5.3. For example, if a dataset contains 100 positive and 300 negative examples of a label, then p_c for the class should be equal to $\frac{300}{100} = 3$. The loss would act as if the dataset contains $3 \times 100 = 300$ positive examples.

For simplicity, here the weighted BCE loss is presented. This can be used in addition to calculate FL.

$$\ell_c(x, y) = -w_c [p_c y_c \cdot \log \sigma(x_c) + (1 - y_c) \cdot \log(1 - \sigma(x_c))] \quad (5.3)$$

where ℓ_c is the loss function, c is the class number ($c > 1$ for the multi-label classification), w_c is the class weights and p_c is the weight of the positive prediction for class c .

In this experiment, a positive weight of 4 has been given to the labels B, FJ, AN, and the balanced sampler is discarded. As seen in Table 5.13, the performance for these three labels remains high but lower than when using a balanced sampler. Furthermore, weight values are new hyperparameters that require manual search, which causes a bottleneck in the training and evaluation pipeline, thus making the use of a balanced sampler a better choice.

Furthermore, to present a comparison between balanced sampling (Table 5.3) and cost-sensitive learning a bar chart is illustrated in Figure 5.23. For FJ and AN the F1-Scores are higher when balanced sampling is used, as well as the aggregate F1-Score. However, the difference is not significant, and this happens because both the validation and test sets are created without balanced sampling. Finally, 5.24 illustrates how the

Table 5.13: 2d Test Set Metrics for a model trained with cost-sensitive learning

Event	Threshold	Accuracy	Precision	Recall	F1-Score
Exposure	0.646	0.938	0.955	0.969	0.962
Burial	0.561	0.937	0.852	0.821	0.836
Field Joint	0.809	0.969	0.932	0.902	0.917
Anode	0.531	0.938	0.762	0.841	0.800
Free Span	0.794	0.994	1.000	0.979	0.989
Aggregate		0.852	0.914	0.916	0.909

use of balance sampling improves the balance between Precision and Recall for the minority label Anode.

Balanced Sampling and Cost-Sensitive Learning

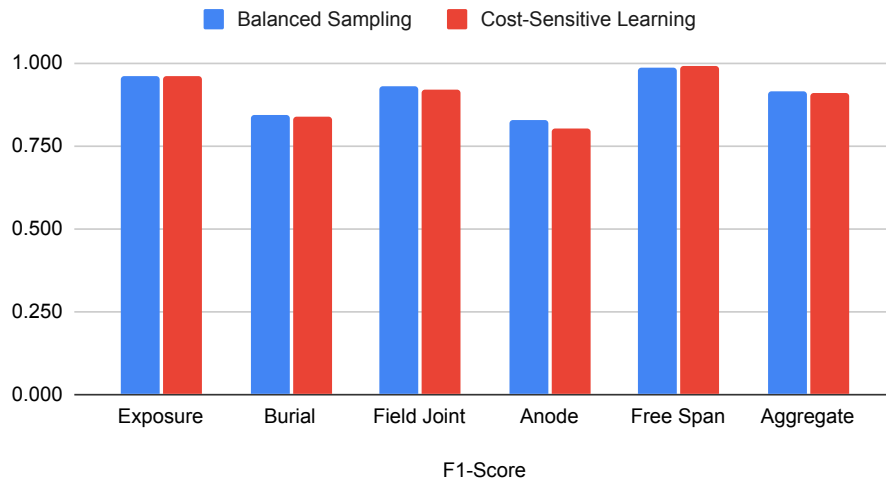


Figure 5.23: F1-Score for Balanced Sampling and Cost-Sensitive Learning

5.10 Conclusions

The presented Chapter makes several significant contributions in the context of subsea survey video analysis. Firstly, a subsea survey video dataset was meticulously created, addressing the challenges of label imbalance and annotation noise, with particular emphasis on handling ‘short’ events. This dataset serves as a valuable resource and can potentially be a benchmark for future studies.

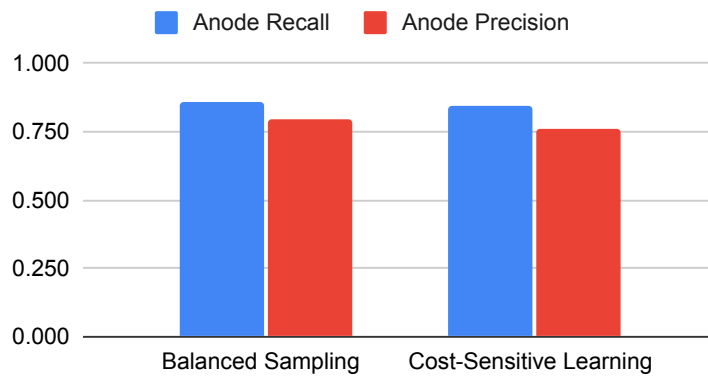


Figure 5.24: Anode Recall and Precision of models trained with Balanced Sampling and Cost-Sensitive Learning

Secondly, three CNN approaches were examined for automating the video annotation of subsea surveys. The first approach involved a 2D IBN-ResNet50, which classified individual frames and averaged the predictions. The second approach introduced a novel 3D IBN-ResNet50, leveraging 3D functionality along with IBN. The incorporation of the 3D IBN-ResNet50 model in this study represents a novel advancement in video analysis and introduces a new approach for subsea survey video classification. Finally, the third approach extended the standard 2D network with temporal awareness using the LSTM framework, resulting in the 2D IBN-ResNet50-LSTM model. Importantly, all three network architectures integrated IBN between the convolutional layers to enhance performance under varying lighting conditions and changes in color contrast.

The results obtained indicate that the 2D model exhibits superior performance, particularly for ‘short’ events such as Anode and Field Joint. Notably, the 2D model also offers advantages in terms of training efficiency and parameter reduction. The two spatio-temporal models demonstrated comparable performance levels. However, it should be noted that the larger models suffered from a high number of parameters, and their performance is expected to improve with a larger dataset size. Furthermore, additional sensitivity analysis experiments were conducted on the 2D model, exploring the impact of label smoothing, Focal Loss, and weighted loss. These experiments provided further insights into optimizing the model’s performance and demonstrated its adaptability to different configurations.

Importantly, this work represents an achievement as it is the first study to apply DL techniques directly to subsea pipeline video data, rather than individual frames. By doing so, it offers a practical solution for developing an intelligent decision support tool for video annotation in subsea pipeline analysis. This tool can be employed in conjunction with human operators to accelerate and enhance the annotation process, thus driving efficiency and effectiveness in subsea survey operations. The novelty and potential impact of this approach in the subsea survey domain cannot be overstated, opening up new avenues for research and development in the field of subsea video analysis.

The introduction of the 3D IBN-ResNet50 model, alongside the establishment of this benchmark dataset, underscores the significance and originality of this work. It sets a foundation for future researchers and practitioners to build upon our findings, refining video analysis techniques for subsea pipelines and driving progress in this important field.

Chapter 6

Subsea Survey Shift and Adaptation

6.1 Introduction

A significant concern when integrating an automatic annotation framework in the inspection process is the transferability of the DL model on new surveys. Naturally, different surveys will result in distribution shifts between the data available for training. These domain shifts can occur due to different locations that have specific biases from seabed peculiarities, sand agitation characteristics or water salinity and turbidity. This is exacerbated by the variability in ROV equipment, such as cameras and their mounting and lighting, which can change the perspectives of video capture. Even for surveys of the same pipeline could result in a change of appearance due to degradation, scouring, vegetation, etc.

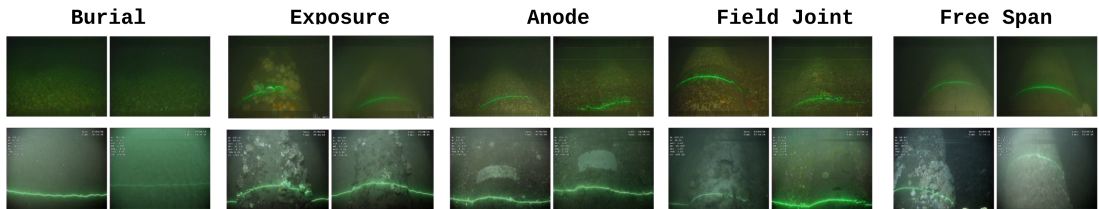


Figure 6.1: Subsea survey data recorded in 2012 (first row) and 2016 (second row)

Figure 6.1 shows Burial, Exposure, Anode, Field Joint and Free Span event ex-

amples from two surveys of two different sites, conducted in 2012 (top row) and 2016 (bottom row). These images illustrate that the two surveys differ both in texture and structure for all events of interest. The texture is different in terms of colour, contrast, illumination, and marine growth vegetation. For structure, there are differences in the text overlays and the visibility of laser lines caused by ROV camera positioning. More specifically, for the Anode event, the 2016 survey has an unobstructed view of the corrosion ring and is easier to distinguish compared to those in the 2012 survey. The Field Joints vary in depressions on the pipeline surfaces, with the 2016 survey being deeper and, consequently, easier to distinguish. These differences mean that a model trained on footage from one survey may perform significantly worse on footage from another survey. Examples frames from both surveys are shown in Figures 6.1 and 6.4.

To explore the significance of the distribution shift image classifiers were trained on the individual surveys and tested on the other survey. The classifiers are trained to make predictions on the five events of interest previously analysed namely; B, E, AN, FJ and FS.

Two datasets were created following the method described in Section 4.12 that maintain event information and consist of 2,828 and 2,751 samples from surveys conducted in 2012 and 2016, respectively. The two datasets have approximately equal number of frames from each label, as can be seen in Figure 6.2.

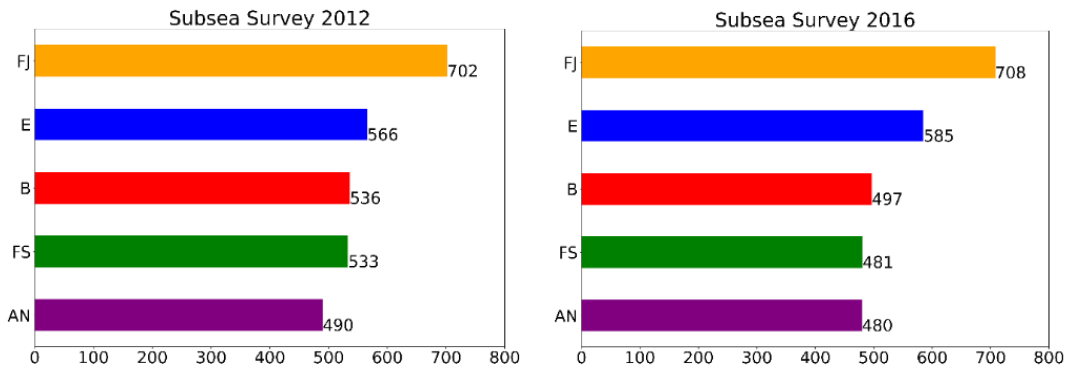
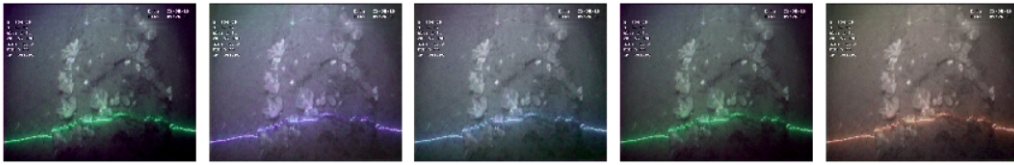


Figure 6.2: Events Frame Distribution for the two Subsea Surveys

The training scheme is almost identical to that described in Chapter 4 and has been adopted to showcase the differences in model performance when trained on data

from one survey (source) and tested on another (target). The model used to perform image classification is ResNet-50 [180]. For both surveys, the training routine is the same with maximum number of 100 epochs, Adam optimizer, a learning rate of 0.001, label smoothing with $\alpha = 0.1$ and employing FL. An one-cycle scheduler training has been used to accelerate convergence [126] while the optimal models are saved on the basis of the lowest validation loss within the 100 epochs of training. Both textural and spatial augmentations (Figure 6.3) are used during the training phase to regularise and increase model generalisation ability namely the augmentations used are: RGB Shift, Hue Saturation Value, Channel Shuffle, Random Contrast, Random Brightness, Blur, Median Blur, and Elastic Transformation (explained in Section 5.5.2), Random Rotation by a maximum of 20 degrees and Horizontal Flipping [283].

(a) Examples of Texture Augmentations



(b) Examples of Spatial Augmentations

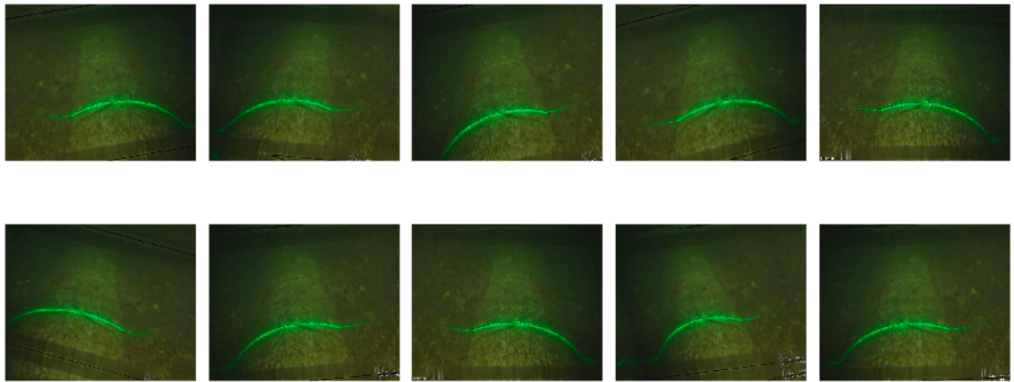


Figure 6.3: Augmentations used during training

The available data from each survey have been split in a stratified fashion in training,

validation, and testing sets with 60%, 20%, 20% of the samples, respectively. Frames of the same events are only contained in one of the splits to ensure no leakage (as explained in Section 4.12). When testing the trained model on the other (target) survey, inference is performed in the entire dataset. The models are trained in a multi-label setting and the metric used to assess model performance is the F1-Score [224]. In addition, the threshold search is performed using the PR curves as described in Section 4.9.

Table 6.1 summarises the results for two models, trained on 2012 or 2016 data. Their performance on each survey (2012 and 2016) is shown on per label basis as well as aggregate F1-Score. It can be observed that the aggregate F1-Score of both models drops significantly (approximately ~ 0.3) when tested on the new domain (i.e. model trained on 2012 and tested on 2016 and vice versa). From the per label results it can be observed that the performance for the E label is relatively sustained, while the AN and FS suffers the highest performance degradation. This could be attributed to the significantly different visual appearance of these events between surveys and that assumption will be further tested in this Section 6.6.2.

Table 6.1: ResNet-50; metrics on source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.927		0.988		0.522		0.666		0.772		0.864	
	2016	0.405	0.523	0.893	0.095	0.227	0.294	0.287	0.379	0.103	0.668	0.569	0.294
2016	2016	0.742		0.950		0.659		0.524		0.695		0.776	
	2012	0.508	0.234	0.753	0.196	0.120	0.539	0.425	0.099	0.216	0.478	0.510	0.266

In particular, the Chapter will investigate:

- *Can a model architecture or a dataset be adapted to enable it to sustain performance between domains?*

Naturally, the image statistics between surveys are expected to be different, a factor that could be accounted using various intermediate normalisation approaches. First, an experiment has been carried out to compare the intermediate normalisation layers of the ResNet-50 architecture. Second, a two-step domain adaptation technique that combines the SAE [284] architecture with a classifier module is explored in an attempt to create a model that is trained on domain-invariant features. Finally, a new dataset is synthesised and when added to the training it

is proven to be beneficial for cross-domain generalisation.

- *How can the shift between two survey datasets be measured and visualised?*

A method is examined that inspect and identify the domain shift [285] between the two subsea pipeline surveys. This technique is model-agnostic and uses FID to compare the distribution of the images from the first survey with the distribution of the images from a second survey. Furthermore, the FID between classes is computed to highlight the intra-class variation for in- and out-of-domain data.

- *If a model is to be re-trained with a mix of events from both domains, how many events must be manually annotated to build a high performing model for the target domain?*

To answer this question, the source domain model is re-trained by progressively adding events from the target domain to identify their effect in FID between the two datasets and the overall performance of the model in both domains.

6.2 ResNets with Different Intermediate Normalisation Layers

The calculation of the aggregate statistics of the images in the two survey datasets 2012 and 2016 per channel (shown in Table 6.2) highlights their difference in visual appearance. It can be seen that the mean values of three colour channels in the 2012 survey are lower than those of 2016. This is also visible by inspecting the frames between the two surveys shown in Figure 6.4. The 2012 survey appears to have a brown colour bias compared to the higher green values observed in the 2016 frames. Furthermore, the standard deviation of the channels for the 2016 data is higher, indicating lighting variations in the survey and the shift compared to the 2012 domain.

This difference in statistics provides motivation for experimentation with different intermediate normalisation layers, namely, IBN [238] and GN [149]. As stated in Section 4.11.5, intermediate normalisation layers can potentially improve the generalisability of a model [238]. Figure 6.5 shows the modifications in the ResNet-50 layers

Table 6.2: Global Means and Standard Deviations of the 2 Surveys

	Survey 2012			Survey 2016		
Mean	0.1793	0.2268	0.1113	0.2414	0.3410	0.2630
Standard Deviation	0.0732	0.0768	0.0426	0.1234	0.1466	0.1319
	R	G	B	R	G	B

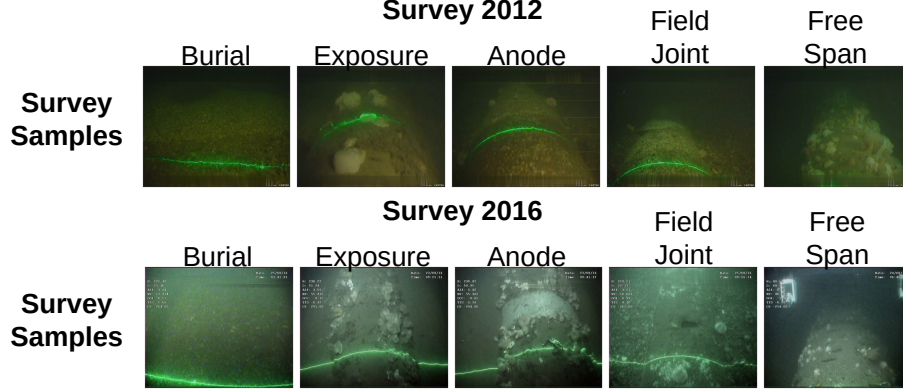


Figure 6.4: Samples of 2012 and 2016 surveys

(which utilise BN) to implement IBN and GN. IN is applied to a single image or (feature vector) while BN to a batch. Intuitively, IN allows one to remove instance-specific contrast information from the content image in a task such as image stylization. GN, on the other hand, can exploit channel dependence by dividing channels of an image (or feature vector) into certain groups and normalising each group separately [149] (see Figure 3.14 of Section 3.4.5). Therefore, since each group of channels is assumed to have a shared mean and variance, the model has the flexibility of learning a different distribution for each group. Another advantage of GN compared to BN is that it is independent of the mini-batch size, where BN can exhibit unstable behaviour when the mini-batch size is small [149]. The number of groups for GN is set to 32, which is the default number proposed [149].

Table 6.3 provides a reminder of the metrics acquired using ResNet-50 for the source and target surveys, which was shown in Section 6.1 of this Chapter. Table 6.4 and Table 6.5 present the metrics acquired using IBN-ResNet-50 and GN-ResNet-50, respectively.

IBN-ResNet-50 performs equally well with ResNet-50 in the 2012 source survey and

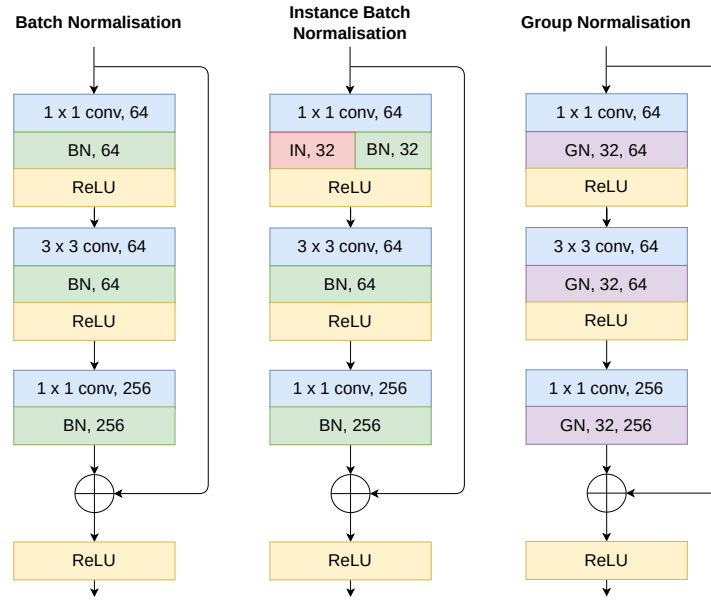


Figure 6.5: Intermediate Normalisation

slightly improves the performance on the target survey where the the F1-Total of the model trained on 2012 and tested on 2016 increases from 0.569 when BN is used to 0.575 when IBN is used. However, this is not similar in the opposite direction, where a model trained on 2016 and applied to 2012 with a shift in performance from 0.510 to 0.495. Interestingly, the source 2016 F1-Total improves from 0.776 to 0.835, indicating that IBN has a positive effect within the same survey.

For the results of GN-ResNet-50, the total F1-Score drops when the model is tested within the survey 2012, from 0.864 to 0.856 but increases from 0.776 to 0.790 within the source survey 2016. Similarly with IBN the F1-Total increases when applied to cross-domain testing from 2012 to 2016 but decreases in the opposite direction. The 2012 model applied to 2016 increases to 0.592 but the 2016 model applied on 2012 decreases to 0.479. This shows that both IBN and GN modifications boost cross-domain performance only in one direction for these two subsea survey datasets. However, they both boost performance within the 2016 survey.

Table 6.3: ResNet-50; metrics on source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.927	0.523	0.988	0.095	0.522	0.294	0.666	0.379	0.772	0.668	0.864	0.294
	2016	0.405		0.893		0.227		0.287		0.103		0.569	
2016	2016	0.742	0.234	0.950	0.196	0.659	0.539	0.524	0.099	0.695	0.478	0.776	0.266
	2012	0.508		0.753		0.120		0.425		0.216		0.510	

Table 6.4: IBN-ResNet-50; metrics on source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.967	0.810	0.991	0.085	0.450	0.163	0.638	0.420	0.786	0.702	0.863	0.287
	2016	0.156		0.906		0.286		0.217		0.083		0.575	
2016	2016	0.777	0.637	0.955	0.103	0.760	0.629	0.692	0.530	0.684	0.484	0.835	0.339
	2012	0.140		0.852		0.130		0.161		0.199		0.495	

Table 6.5: GN-ResNet-50; metrics on source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	1.000	0.950	0.998	0.099	0.429	0.242	0.672	0.550	0.617	0.481	0.856	0.263
	2016	0.049		0.898		0.187		0.121		0.135		0.592	
2016	2016	0.769	0.277	0.948	0.157	0.720	0.538	0.536	0.336	0.666	0.476	0.790	0.311
	2012	0.491		0.790		0.182		0.199		0.189		0.479	

6.3 Subsea Survey Domain Adaptation

Transfer learning serves as a fundamental technique for leveraging knowledge learned from a source domain (where labeled data is available) to improve performance in a target domain (where labeled data may be scarce or unavailable). This is achieved by utilizing pre-trained models on a large labeled dataset and fine-tuning them on the target domain dataset.

Domain adaptation, on the other hand, is a specific approach within transfer learning that focuses on mitigating the domain shift between the source and target domains. Domain adaptation deals with scenarios in which a model trained on a source dataset is used in the context of a different (but related) target dataset [286]. In general, domain adaptation methods use labelled data in one source domain to solve new tasks in a target domain without access to labels but where there is access to unlabelled data. It aims to align the statistical properties of the datasets from different domains to improve model performance in the target domain. In this study, a two-step domain adaptation methodology is developed, incorporating the Swapping Autoencoder architecture and data synthesis to address the domain shift in the subsea survey context.

To integrate the unlabeled dataset into the labeled dataset, the study likely employs unsupervised learning techniques. Unsupervised learning allows the model to learn patterns and representations from the data without explicit labels. By leveraging the information present in the unlabeled dataset, unsupervised learning methods such as clustering, autoencoders, or generative models can be employed to extract useful features and augment the labeled dataset. This incorporation of unlabeled data helps to enhance the model’s performance and generalization ability in the target domain.

Through the integration of the unlabeled dataset and the utilization of transfer learning and domain adaptation techniques, the study potentially uncovers hidden contributions within the chapter. These contributions could include improved model performance, enhanced generalization capabilities, and the discovery of previously unknown patterns or insights in the inspection process.

In the context of subsea surveys, one of the survey datasets is used as the source domain, while the other is used as the target domain to perform image classification. More specifically, in the scenario considered here, access to both images and labels of the source domain is available, but only unlabelled data are available for the target domain. This scenario is relevant given that the cost of obtaining annotated data remains expensive for subsea pipeline inspections.

Recent domain adaptation methods align source and target domains by creating domain-invariant feature representations, typically in the form of a feature extractor. A feature representation is domain-invariant if the features of both the source and target domains allow for similar performance across domains. Many works [287–290] of domain adaptation can be modelled with the general framework presented by Murez *et al.* [291] that makes use of several auxiliary networks and losses to help regularise the latent space, with the goal of making its embedding domain agnostic. The framework utilises different combinations of auxiliary networks consisting of encoders and decoders, a domain discriminator, a label classifier, and a discriminator for translated images. The loss can be a combination of losses such as: (i) identity (or reconstruction) loss, (ii) adversarial domain discrimination loss like in [287, 289], (iii) discriminator loss for translated images, (iv) cycle consistency loss [292], and (v) classification losses.

The authors show that by adjusting the loss terms, their framework can encompass many works as special cases; for example, it can simultaneously achieve image-to-image translation, domain discrimination, and domain adaptation. However, this framework is not considered further in this work because it requires experimentation with weights of the loss components which are task- and dataset-specific and the potential of GAN mode collapse [293].

6.4 Two-Step Swapping Autoencoder plus Classifier for Cross Domain Classification

Inspired by the work mentioned above (Murez *et. al* [291]), the method presented here uses image translation for unsupervised domain adaptation, using the SAE architecture, with the addition of a classifier module. SAE is a framework in which an encoder-decoder architecture is trained on datasets from one or multiple domains. The factorised representation derived by the encoder module provides an abstraction, and this can facilitate many different functionalities, such as texture swapping, style editing, and domain translation. The architecture of SAE is seen in Figure 6.6, where the style of one church image from the LSUN [294] dataset transforms the content church image.

This enables creating embeddings that use the source structure combined with the target texture or embeddings that are texture-agnostic (see Figure 6.10) and experiments have been performed with both. The rational behind this is that by creating texture-invariant embeddings, the model will be able to perform well on the target domain as the texture will incorporate lighting changes, sand, and blurriness from the seabed, different colours, and vegetation artefacts.

The encoder of SAE consists of four downsampling residual blocks and then the model has two branches: to obtain the embedding of the texture, there are two convolutional layers, followed by an average pooling (to remove spatial dimensions) and a dense layer leading to a 1×2048 texture vector, to obtain the embedding of the structure, one convolutional layer is used that outputs a structure code of $8 \times 16 \times 16$. The SAE encoder is within a similar range in parameter size to ResNet-50 (25 million

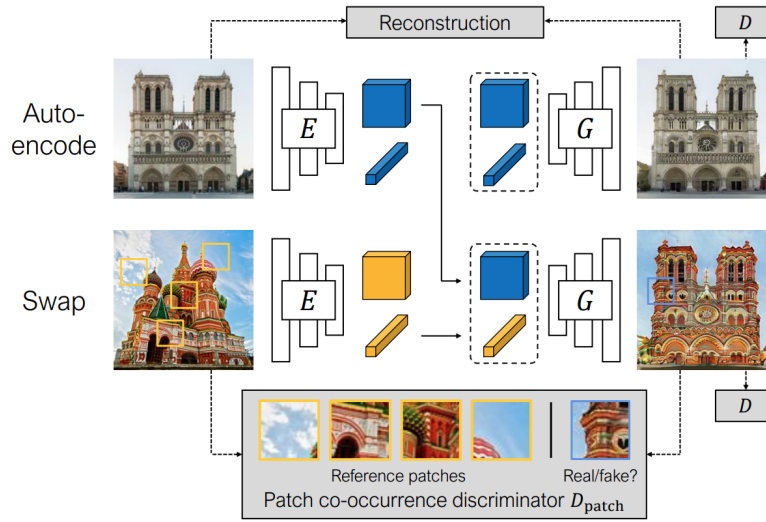


Figure 6.6: Swapping Autoencoder Architecture [284] on images from LSUN churches [294]

parameters) consisting of 32 million parameters. Encoders of this size provide similar performance to the ImageNet classification as seen in the Table 6.6.

Table 6.6: Models with their corresponding Accuracy on ImageNet and Parameters

Model	Accuracy	Parameters (approx.)
ResNet18	70%	11 million
ResNet34	73%	21 million
ResNet50	76%	25 million
ResNet101	77%	44 million
ResNet152	78%	60 million
DenseNet201	77%	20 million
InceptionV3	77%	23 million

The decoder (or generator) of SAE is a model with four residual blocks and four residual upsampling blocks, where the texture code is injected using the weight modulation/demodulation layer of StyleGAN2 [295]. Furthermore, the entire image discriminator architecture (D in Figure 6.6 is identical to the StyleGAN2 discriminator, ensuring that the output produced belongs to the domain of probable images (after training).

The patch co-occurrence discriminator enforces a match between the distribution

of patches in the texture source and the generated image. Each patch is first independently encoded with five downsampling residual blocks, one residual block, and one convolutional layer. The representations for the reference patches are averaged together and concatenated with the representation of the real/fake patch. The classification applies three dense layers to output the final prediction. The whole architecture is trained using three losses: (i) the reconstruction (or autoencoding) loss (top of Figure 6.6, (ii) the GAN losses from the the whole image discriminator D for both the reconstructed and the swapped image, and (iii) the patch co-occurrence loss.

6.4.1 Training Details of the First Step

Here, the entire architecture (SAE plus classifier module) is trained in two steps as shown in Figures 6.7, 6.10 to mitigate mode collapse. The first training step focuses on the SAE module and is executed in an unsupervised fashion without labels using all available data from the source and target surveys.

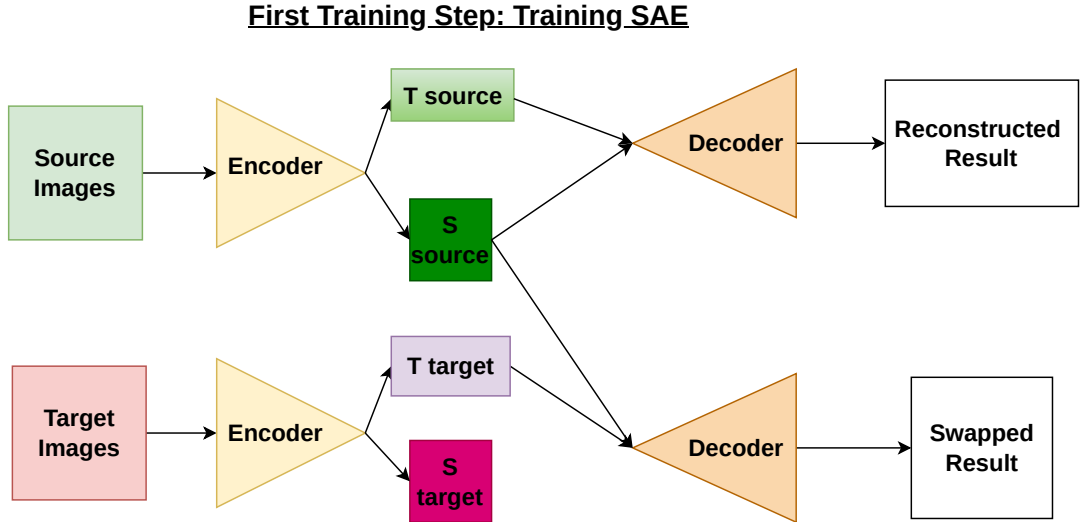


Figure 6.7: First Training Step. For the sake of simplicity, the Discriminator modules are not added to the diagram.

The SAE architecture is trained with all images from the source and target domains on the texture swapping and reconstruction tasks. The training set consists of all images from both domains, where a 20% subset is randomly chosen as the validation set.

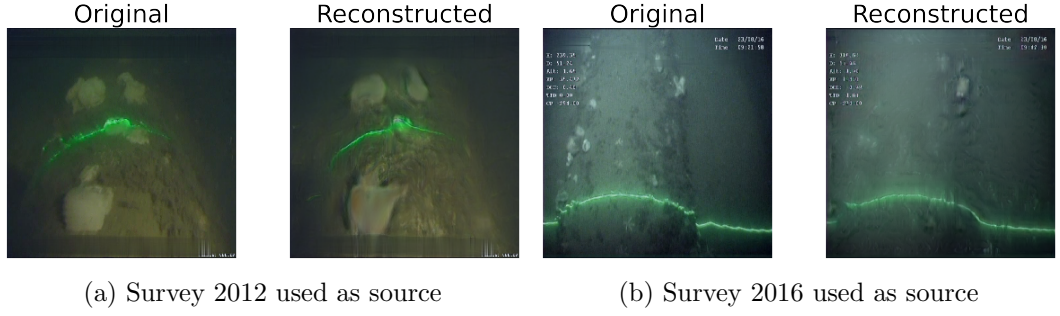


Figure 6.8: SAE Reconstruction Results

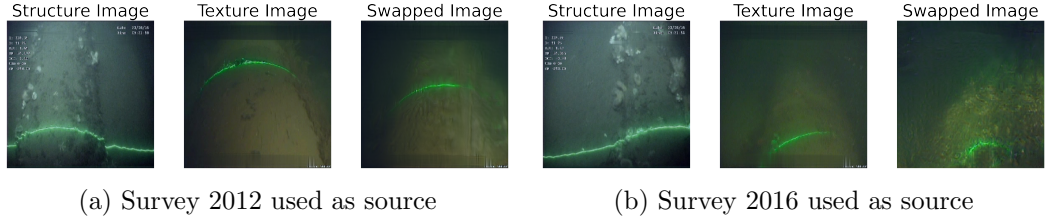


Figure 6.9: SAE Reconstruction Results

Training is carried out using Adam [204] optimiser and a learning rate of 0.001. Models are saved based on the lower validation loss, which is chosen to be the reconstruction loss component on the validation set images. The $L1$ loss [296] is used to measure the error between the original and reconstructed images. The images have been resized to 256×256 as this is a requirement of the original architecture to maintain the embeddings Structure (S) and Texture (T) of sizes 1×2048 when flattened. The size of the mini-batch is set to 4 while training the SAE module, and the number of epochs is set to 200. The best models are obtained in epochs 186 and 173 when the source survey is 2012 and 2016, respectively. Figure 6.8 and Figure 6.9 provide the results of the reconstructed and swapped output images. From these, it can be observed that the resulting encoder produces disentangled latent representations of texture and structure, which can then be used to train a classifier in a second subsequent step.

6.4.2 Training Details of the Second Step

In that second step, the encoder network is detached and a classifier is added. There are multiple approaches which the latent features could be utilised for the training the

classifier module in a supervised fashion as shown in Figure 6.10.

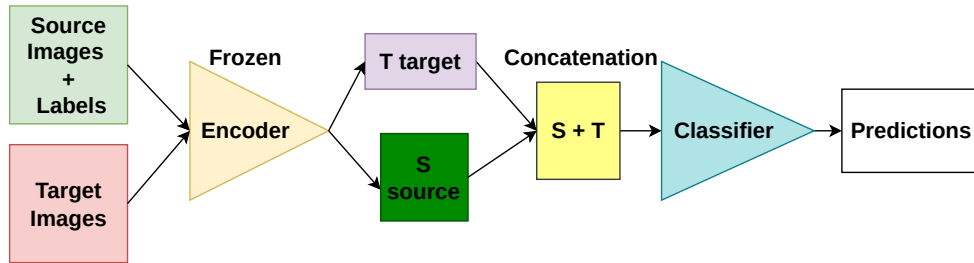
1. Approach A: One approach is to use the images and labels from the source survey and infuse them with the target domain texture. The classifier can then use the concatenation of the Structure embeddings from the source survey (on which labels are available) and the Texture representation from the target survey to adopt predictions to the target domain.
2. Approach B: The second approach is to use source survey images and labels, encode them and then only utilise the source Structure embeddings, bypassing the concatenation step, to make the classifier desensitised to texture.

The intuition for these approaches is that the encoder has learnt to disentangle the texture from the images, and therefore the created embeddings are either adjusted to the target domain or agnostic to the source subsea textures.

(a) Approach A

Second Training Step: Classifier Training with frozen SAE encoder

Approach A: Fusion of Target Texture with Source Structure



(b) Approach B

Second Training Step: Classifier Training with frozen SAE encoder

Experiment B: Removing Source Texture

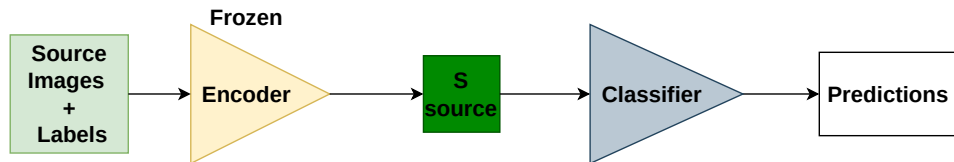


Figure 6.10: Second Training Step

In the experiments performed, the classifier architecture consists of two fully con-

nected (linear) layers with BN and Dropout [164] layers between them for regularisation purposes. S and T embeddings are concatenated into one with a size of 1×4096 which is passed to the classifier to obtain the output predictions. The training hyperparameters are the same for both experiments as described in Section 6.1.

For the second training step, the same data splitting methodology has been performed as described in Section 6.1 where the available source data from each survey have been split in a stratified fashion in training, validation, and testing sets with 60%, 20%, 20% of the samples, respectively, while ensuring that frames from the same events are contained only in one of the splits. The resultant model is trained on the source survey dataset, however, it can be tested on the entire target survey dataset. Although the imagery from the target dataset has been used for training the SAE architecture, the labels have not been used and therefore it constitutes a valid test set.

For both approaches, all possible combinations of freezing and unfreezing the encoder network and branches have been investigated and in the next Section the ones with the best results are presented. For Approach A, the encoder network is unfrozen, but both the Structure and Texture branches are kept frozen, while for Approach B, the encoder network and the Structure branch are unfrozen during training.

6.4.3 Results of Approaches A and B

To evaluate the performance of a domain adaptation model for the image classification task in the target domain, the F1-Score per class and aggregate is used in a fashion similar to Section 6.2.

For comparison, Table 6.7 presents the results acquired with ResNet-50, while 6.8 presents the results acquired with a model that makes use of the SAE encoder, used as the feature extractor, while the classifier is the same as in ResNet-50. The SAE plus classifier model is trained end-to-end, similarly to ResNet-50. When the SAE encoder is used as a feature extractor, the performance in the target domains is similar to the one of ResNet-50.

Table 6.9 presents the results obtained for the Approach A; i.e. the SAE model trained by fusing target Texture with source Structure. Table 6.10 shows the results

Table 6.7: ResNet-50; metrics on source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.927	0.523	0.988	0.095	0.522	0.294	0.666	0.379	0.772	0.668	0.864	0.294
	2016	0.405		0.893		0.227		0.287		0.103		0.569	
2016	2016	0.742	0.234	0.950	0.196	0.659	0.539	0.524	0.099	0.695	0.478	0.776	0.266
	2012	0.508		0.753		0.120		0.425		0.216		0.510	

Table 6.8: SAE encoder plus classifier; metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.990	0.990	0.994	0.093	0.547	0.267	0.731	0.381	0.786	0.424	0.873	0.335
	2016	0.000		0.900		0.280		0.350		0.362		0.537	
2016	2016	0.632	0.619	0.919	0.035	0.638	0.340	0.554	0.150	0.728	0.582	0.750	0.214
	2012	0.013		0.883		0.297		0.404		0.145		0.536	

obtained for Approach B; that is, when Texture information is removed and the Structure from the source is only used to train the classifier.

Tables 6.9 and 6.10 present similar patterns on the metrics the aggregate F1-Score. When evaluated within domain, the F1-Score for Approaches A and B is similar to both the ResNet-50 and SAE plus classifier model; for example, models trained on 2012 and testing on 2012 results in an F1-Score of approximately 0.86, whereas the 2016 to 2016 models have an F1-Score of approximately 0.75.

In addition, for the 2012 models tested on the 2016 survey, the performance is boosted by 0.02 and 0.04, for the Approaches A and B, respectively. Furthermore, in the opposite direction the performance of the models trained on 2016 but tested on 2012 is boosted with 0.07 and 0.03, for the two Approaches.

Approach A (target texture) achieves an improvement of 0.07 in the performance of the target 2016 survey, to reach an F1-Total of 0.608 compared to the 0.536 of Table 6.8. More specifically, the F1-B and F1-E of the target dataset are comparable with the ones of the source survey indicating that texture fusion can improve the pipeline detection in new surveys. The performance for the other three events is low indicating that texture fusion is not sufficient for detecting events that potentially have stronger structural features like the the Anode, Field Joint and Free Span.

With approach B (only structure) the drop in performance between source and target surveys is lower than when no adaptation techniques are used. An interesting observation is that while F1-B of the target survey improves the F1-FJ is lower. This

can be attributed to the fact that Burial frames are more textural images, as they are images of the seabed covered with sand and vegetation whereas the structure in frames of Field Joints is a string feature for detecting these events. Therefore, removing the texture from the latent space helps generalizing across domain for Burial but not for Field Joint as the model potentially overfits to the source dataset Field Joint structural features.

Overall, approach B indicates the potential importance of Structure over Texture for this specific application, as the performance of the models trained with only structural embeddings is almost identical with the performance of the models trained with both structural and textural embeddings for the source surveys, while there is an improvements for the F1-Total of the target surveys.

For both approaches the performance on the source surveys is similar to ResNet-50 and the SAE encoder plus classifier, while with both approaches, the performance on the target surveys is improved. Hence, the domain adaptation techniques can be seen as a regularisation permitting increased performance when applied across domains, while maintaining the same on the source domain.

Table 6.9: Approach A (target texture); metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.995	0.624	0.998	0.172	0.357	0.098	0.700	0.661	0.748	0.704	0.862	0.311
	2016	0.370		0.826		0.258		0.038		0.044		0.550	
2016	2016	0.717	0.096	0.942	0.043	0.512	0.126	0.531	0.336	0.637	0.454	0.765	0.156
	2012	0.620		0.898		0.385		0.195		0.182		0.608	

Table 6.10: Approach B (only structure); metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	1.000	0.662	1.000	0.141	0.690	0.347	0.467	0.688	0.676	0.579	0.851	0.281
	2016	0.337		0.858		0.120		0.002		0.097		0.570	
2016	2016	0.696	0.182	0.938	0.048	0.696	0.408	0.524	0.395	0.560	0.303	0.747	0.186
	2012	0.513		0.890		0.287		0.128		0.256		0.561	

The difference between Approach A and Approach B is that the first makes the model invariant to the target texture, while the second approach disregards texture all together. In both cases, the performance is improved compared to the ResNet-50 or the SAE plus classifier model, however, they do not reach the performance of models trained in a supervised fashion. This leads to the fact that factorising texture is helpful,

but is not adequate to sustain classifier performance and it is hypothesised that the structural differences between the events across the surveys are higher, leading the classifier to suffer confusion between events. To explore in-domain and out-of-domain similarities further investigation is performed using the FID.

6.5 Synthetic Dataset using SAE for Cross Domain Classification

Another potential solution is to use SAE to modify the source or target dataset to their counterpart domain [132, 297–299]. After training an image translation model, there are two main modes of operation as seen in Figure 6.11: (i) Use the classifier model trained on data from the source domain, but when inferring in the target domain, the imagery is translated to the source domain, or (ii) retrain the classifier model with data from the source domain plus data translated from the target domain to the source; and then use this retrained model to perform conventional inference in the target domain.

In both scenarios, only the labelled data from the source domain are used. The preliminary results on the first approach showed poor performance on the style-modified target dataset, so the second approach is explored here.

There is a plethora of solutions [151, 177, 292, 300–302] that have been proposed for image translation using either NST or GAN frameworks. NST is not explored here because there are two limitations; (i) NST only takes into account one reference image of a domain as a style, and (ii) it would be computationally challenging to retrain the model for the inference of every image. Multiple GAN architectures have been proposed that use paired [301] or unpaired images [292, 302] for training. However, in this work, the architecture SAE is chosen because it allows access to the disentanglement of structural and textural components (content and style) in the latent space, as shown in Figure 6.6.

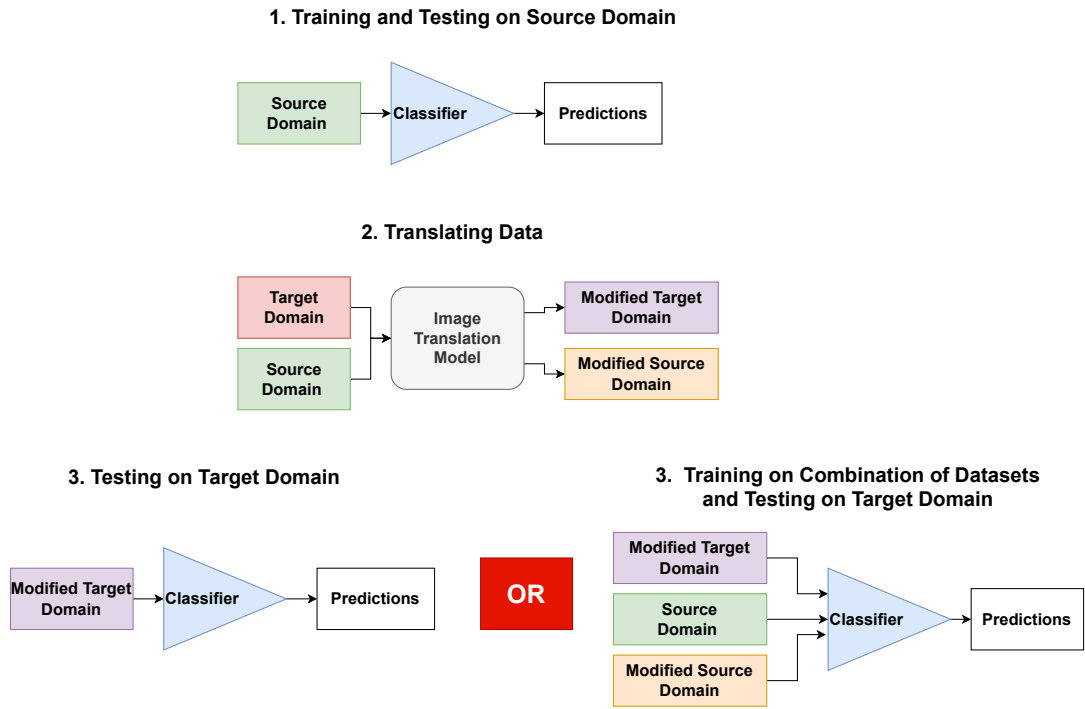


Figure 6.11: Steps required when using Image Translation for Domain Adaptation: Step 1 is to train and test a classifier in the source domain to get a baseline performance measurement. On Step 2, an image translation model can be trained to modify either the source or target domain data accordingly, these models only have access to the source labels. On Step 3, there is two options: (i) use the classifier model trained on data from the source domain, but when inferring in the target domain, the imagery is translated to the source domain, or (ii) retrain the classifier model with data from the source domain plus data translated from the target domain to the source; and then use this retrained model to perform conventional inference in the target domain.

6.5.1 Results of training with Synthesised Data

Using SAE two synthesised datasets were created. These synthesised survey datasets have been named ‘S12T16’ and ‘S16T12’, with ‘S’ representing the Structure and ‘T’ the Texture used, respectively. For example, ‘S12T16’ has the structure of survey 2012 but the texture of survey 2016 (see Figure 6.9). The label and the event information of the produced images are aligned with the source image that was used during the synthesis, whereas the texture image is randomly chosen from the target survey. This means that SAE translates images from the source domain to the target domain and the opposite while maintaining the semantic information present in the source labels; here the semantic information is the Structure (S) of an image which is aligned with an event label, while the Texture (T) is the only available information from the target domain, where images are available but not their corresponding labels. In that way, the event information for splitting the data is kept. Following the steps of Figure 6.11 the synthesised datasets were used to re-train new models along with the original source survey datasets. In one direction, when the source survey is the 2012 survey, the synthesized dataset ‘S12T16’ is added to the training while the testing remains the same. On the other direction, survey 2016 along with ‘S16T12’ has been used for the training. The models that are re-trained here, are based on ResNet and its variations and the training combinations are shown in Tables 6.11,6.12,6.13 where the results are also compared with the models of Section 6.2; the first two rows of each table provide reminders of the models trained only with the source surveys, while the next lines present the results of the models trained with both the source and the synthesised datasets.

For the ResNet-50 models trained with synthetic datasets, the results of target surveys show consistent improvement where aggregate performance on target datasets (F1-Total) increases by ~ 0.04 , indicating that increasing the training dataset with synthesised examples improves cross-survey generalisation. Interestingly, this is not the case for the IBN-ResNet that has been trained with the 2012 plus the S12T16 datasets, where the performance on the target 2016 survey drops, and for the GN-ResNet where the performance remains the same. The best performance on the target

Chapter 6. Subsea Survey Shift and Adaptation

surveys is acquired by ResNet with F1-Total of 0.601 and 0.542 in the target surveys 2016 and 2012 compared to 0.569 and 0.510, respectively.

Table 6.11: ResNet-50 with and without synthesised data; metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.927	0.523	0.988	0.095	0.522	0.294	0.666	0.379	0.772	0.668	0.864	0.294
	2016	0.405		0.893		0.227		0.287		0.103		0.569	
2016	2016	0.742	0.234	0.950	0.196	0.659	0.539	0.524	0.099	0.695	0.478	0.776	0.266
	2012	0.508		0.753		0.120		0.425		0.216		0.510	
2012, S12T16	2012	0.934	0.517	0.983	0.096	0.481	0.481	0.725	0.577	0.662	0.654	0.852	0.250
	2016	0.417		0.887		0.000		0.147		0.008		0.601	
2016, S16T12	2016	0.773	0.484	0.955	0.085	0.770	0.627	0.548	0.500	0.792	0.546	0.818	0.275
	2012	0.289		0.869		0.143		0.048		0.246		0.542	

Table 6.12: IBN-ResNet-50 with and without synthesised data; metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	0.967	0.810	0.991	0.085	0.450	0.163	0.638	0.420	0.786	0.702	0.863	0.287
	2016	0.156		0.906		0.286		0.217		0.083		0.575	
2016	2016	0.777	0.637	0.955	0.103	0.760	0.629	0.692	0.530	0.684	0.484	0.835	0.339
	2012	0.140		0.852		0.130		0.161		0.199		0.495	
2012, S12T16	2012	0.990	0.530	0.997	0.286	0.492	0.296	0.694	0.631	0.659	0.640	0.869	0.368
	2016	0.460		0.710		0.195		0.062		0.019		0.500	
2016, S16T12	2016	0.734	0.727	0.953	0.059	0.788	0.719	0.550	0.550	0.656	0.381	0.802	0.268
	2012	0.007		0.894		0.068		0.000		0.275		0.534	

Table 6.13: GN-ResNet-50 with and without synthesised data; metrics in source and target surveys

Train	Test	F1-B	Drop	F1-E	Drop	F1-AN	Drop	F1-FJ	Drop	F1-FS	Drop	F1-Total	Drop
2012	2012	1.000	0.950	0.998	0.099	0.429	0.242	0.672	0.550	0.617	0.481	0.856	0.263
	2016	0.049		0.898		0.187		0.121		0.135		0.592	
2016	2016	0.769	0.277	0.948	0.157	0.720	0.538	0.536	0.336	0.666	0.476	0.790	0.311
	2012	0.491		0.790		0.182		0.199		0.189		0.479	
2012, S12T16	2012	0.985	0.896	0.988	0.086	0.440	0.286	0.651	0.553	0.757	0.460	0.849	0.254
	2016	0.088		0.902		0.153		0.097		0.296		0.595	
2016, S16T12	2016	0.770	0.434	0.939	0.118	0.656	0.625	0.515	0.182	0.750	0.586	0.773	0.254
	2012	0.335		0.820		0.031		0.333		0.163		0.519	

6.6 Domain Shift Assessment using Fréchet Inception Distance (FID)

To gain further insight into the suboptimal performance of adaptation models, this section explores the domain shift between surveys and compares it with the Office-Home dataset [303]. Evaluating domain shift in subsea pipeline inspection datasets is

important, as differences in the new surveys lead to decreased model performance as shown in Section 6.4.3. To better understand the domain shift between the two survey datasets, a model-agnostic technique is examined which focuses on a FID metric. FID is based on an ImageNet [304] Inception v3 [179] model and allows comparisons of visual similarities. FID was introduced by [305] to quantitatively compare natural images with those generated by GAN [131, 306]. Here, it is proposed that FID can be used to measure the similarity between the class- and domain-specific subsets of pipeline survey datasets.

Instead of comparing images on a pixel-by-pixel basis, as in the $L2$ norm, FID compares the mean and standard deviation of one of the deeper layers in Inception v3. In this work the last layer has been used which is close to the output, and corresponds to dataset-specific objects instead of general features from the shallow layers near the input image. The FID authors argue that the later layers tend to mimic human perception for image similarity and is calculated using the following equation:

$$FID = d^2 = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2 * \sqrt{(C_1 * C_2)}) \quad (6.1)$$

The score is referred to as d^2 , showing that it is a distance and has squared units. μ_1 and μ_2 refer to the mean of the layer features for the first and second datasets. The C_1 and C_2 are the covariance matrices for the two feature vectors, often referred to as Σ . The $||\mu_1 - \mu_2||^2$ refers to the sum square difference between the two mean vectors. Tr refers to the linear algebra operation of the trace, which is the sum of elements along the main diagonal of the square matrix. The implementation of FID here is based on [307] and uses the last layer before the classification layer (i.e. after average pooling) of Inception v3, which has dimensionality of 2,048 and a batch size of 20. Lower FID scores indicate that two groups of images have similar characteristics, with a perfect score of 0.0 indicating that the two groups of images are identical.

6.6.1 In-Domain and Out-Of-Domain FID without Label Information

To get a sense of how FID varies in and between domains, the following two datasets from Domain Adaptation [286, 308] and Generalization [309] fields have been employed.

The first dataset is the Office-Home dataset [303] that consists of 65 classes and images from 4 domains; Art, Clipart, Product, Real World. Examples of images from this dataset are shown in Figure 6.12. The second dataset was originally developed for image segmentation and consists of 2,500 images of GTA5 [239] and 2,976 images of Cityscapes [240]. Example images are shown in Figure 6.13.





















Domains	Classes				
Art					
Clipart					
Product					
Real World					
	Bed	Bike	Bottle	Chair	Glasses

Figure 6.12: Sample images of the Office-Home Dataset [310]. The dataset contains 65 classes from four domains - Art, Clipart, Product, Real World. Here, 5 example classes are shown.



Figure 6.13: Samples images from the GTA5 [239] and Cityscapes [240] datasets.

The most straightforward but probably the least revealing method to calculate FID

is for all data between two domains ignoring the label information. These FID scores for the Office-Home dataset for all combinations of domains are shown in rows 1-6 in Table 6.14. Similarly, for the second dataset, the FID between the GTA-5 and Cityscapes domains is shown in row 7 of the table. Finally, the last row of the table shows the FID for the two subsea survey domains. To compute FID, all images in all datasets have been resized to 256×256 .

For the Office-Home dataset, the Product and Real World domains are the most similar, with the lowest FID of 29. These two domains contain natural products with similar appearance with different backgrounds (white in Product, variable in the Real World). The largest FID and hence dissimilarity between domains is obtained for Art and Clipart, which matches the intuition that art contains variable object textures and edges compared to the ones in the Clipart.

The FID between Cityscapes and GTA5 is high and similar to that obtained for the comparison of the complete subsea survey dataset for 2012 and 2016 Surveys. The high FID indicates that the imagery between these domains is dissimilar.

Table 6.14: FID between Domains

Row #	Domain 1	Domain 2	FID
1	Art	Clipart	66
2	Art	Product	64
3	Art	Real World	38
4	Clipart	Product	54
5	Clipart	Real Word	53
6	Product	Real World	29
7	Cityscapes	GTA5	105
8	Survey 2012	Survey 2016	102

In addition, Table 6.15 presents the FID scores that have been calculated within a domain. These are obtained by randomly splitting the images of the domain in half¹. This split is performed five times, and the mean FID is presented here. Interestingly, FID within the domains of the Office-Home dataset are higher than those of Cityscapes and Subsea Surveys, indicating that the Office-Home dataset has higher diversity within the domains compared to others. More specifically, the FID score within the Art domain

¹This is necessary to permit FID computation within domain

Table 6.15: FID within Domains

Row #	Domain 1	Domain 2	FID
1	Art	Art	53
2	Clipart	Clipart	25
3	Product	Product	22
4	Real World	Real World	26
5	Cityscapes	Cityscapes	17
6	GTA5	GTA5	22
7	Survey 2012	Survey 2012	11
8	Survey 2016	Survey 2016	10

is 53, whereas the FID score between Art and the Real World is 38. Another example is the FID between Product and Real World which is 29 compared to the FID within the Product domain which is 22. For the Office-Home dataset the FID between domains is lower compared to the Cityscapes and Subsea datasets which indicates that the Art, Clipart, Product and Real World domains are more similar to each other compared to the Cityscapes-GTA5 or the two subsea surveys.

This means that there is more diversity and dissimilarities within the domains but of Office-Home and higher similarities between domains compared to Subsea Surveys and GTA5-Cityscapes, which in turn leads to better generalisation and adaptation from one domain to another. Hence, this would support the observations of Section 6.2 and Section 6.4 that adaptation methods were not as effective in the subsea imagery of the 2012 and 2016 datasets.

6.6.2 In-Domain and Out-Of-Domain FID with Label Information

To gain further insight into the differences between domains and labels, the FID has been calculated for combinations of 5 individual labels within and between the domains Art - Real World and Product - Real World, as shown in Figure 6.14 for the Office-Home dataset (here 5 classes are shown out of 65). Accordingly, for the subsea domains, the comparison between domains and classes is shown in Figure 6.15. The data of each label and domain are split equally (50-50 split)² and the FID is computed for all

²This is necessary to permit FID computation within label and domain.

combinations of labels and domains. The process is repeated five times, and the mean FID is presented here.

The lower left quadrant of the figures shows the comparison between domains on a label basis, while the upper left and lower right triangular regions show the FID between labels for each domain, respectively. The external (main) diagonal annotated with dashed line shows the FID of the same label in the same domain, while the inner, smaller diagonal in the bottom left quadrant shows the same label between domains.

An apparent observation that can be made is that the external diagonal, where classes from the same domains are compared, in all Office-Home matrices typically consist of lower FID scores compared to the rest of the matrix. For the subsea surveys, low FID scores are not only observed on the main diagonal but also in other matrix positions. This is justified given that the labels in the subsea survey dataset are not mutually exclusive (as in the Office-Home dataset) and the task is multi-label classification; for instance the Exposure is the parent class of Anode, Field Joint, and Free Span. In particular all the classes except Burial contain characteristics from the Exposure class explaining to lower FID scores for some matrix locations.

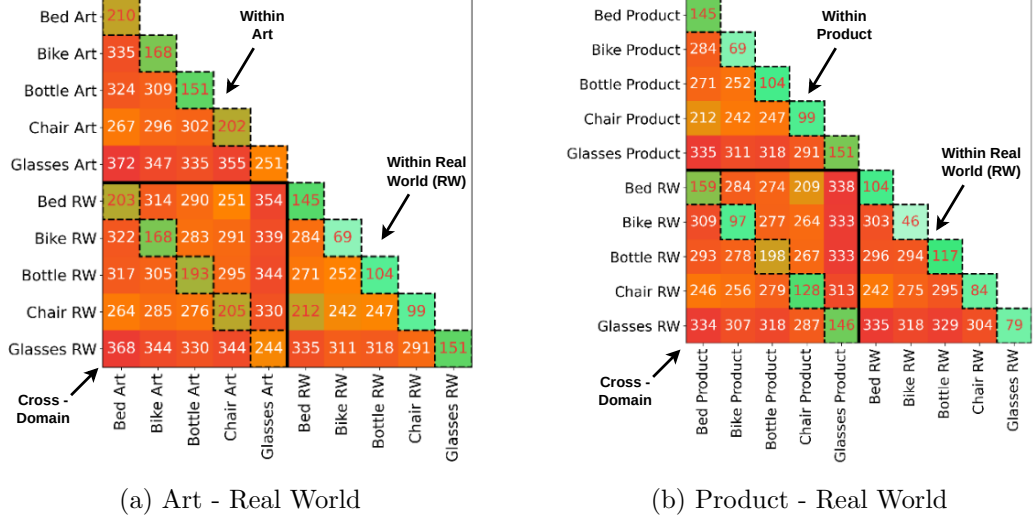


Figure 6.14: Office-Home FIDs

Furthermore, in the cases of Art - Real World and Product - Real World the FID scores of the internal diagonal (across domains) have magnitude which is twice at most

to the main diagonal (within domain) scores. For the Surveys '12 and '16 the internal diagonal has magnitude as high as eight times than those on the main diagonal (within domains). This highlights the data between domains are more divergent compared the Office-Home dataset and consequent it is more challenging for domain adaptation to function. For the Exposure class the FID is 92 between surveys which is of approximately of the same magnitude to that of the Office-Home dataset, domain adaptation offers the highest performance gains (for that event). In addition, although FID in Burial across domains is high, the performance is good because it is a mutually exclusive event with E.

For the two subsea surveys, the FID scores for the same label within the same survey are expected to be relatively low since the input dataset is derivative of the same set (i.e., are the result of the 50-50 split). This is the case as shown in Figure 6.15 with the exception of FS for the 2016 survey, where the FID is 2-3 times higher and equal to 98. This is attributed to the low number of FS events (9, Figure 6.18) that the 2016 survey contains and manual inspection of the video footage indicated that the variation in the appearance of events is high. The FS contain complex scenes with multiple pipeline junctions (crossing between pipelines) and obstructions from ROV equipment (Figure 6.16), unlike the traditional single pipeline imagery. This peculiarities make the FS events within the 2016 survey, distinct from all other events in both surveys and this is supported by the high FID scores showing in the bottom row of the matrix (as high as 201 for B, which is expected considering the visual differences of the two events, but it is also high for the FS 2012 event; score of 191).

Furthermore, by using FID scores it is possible to identify a number of class relationships more precisely. For example, additional classes corresponding to exposure subevents (Anode, Field Joint, Free Span) can be expected to share some similarity with the parent event of Exposure, at least more so than with the Burial event. Interestingly, the FID score between B'16 and E'16 is low which can be an indication of label noise in the dataset. This is verified by manual inspection as shown in Figure 6.17. This effect is not observed for the Survey 12 dataset.

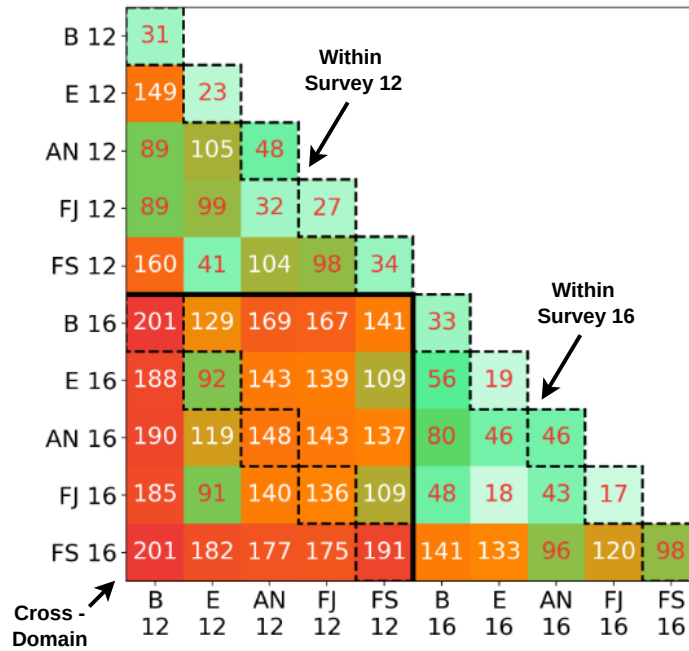


Figure 6.15: FID scores between domains and within domain. The bottom left quadrant marked by the black continuous line contains cross-domain scores. The dashed borders indicate comparisons for the same label. High values along the diagonal in the cross-domain region indicate a significant domain shift.



Figure 6.16: Free Span Events of 2016 Survey

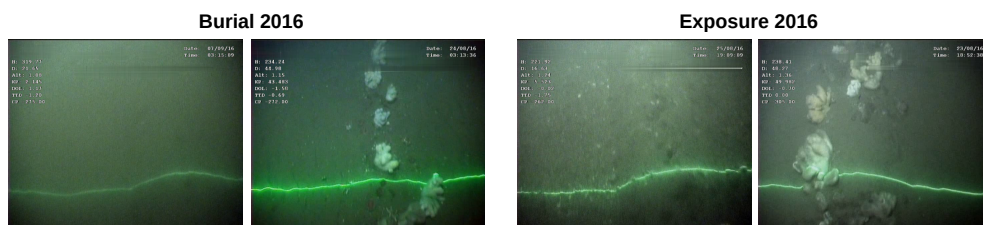


Figure 6.17: Burial and Exposure Events of 2016 Survey

6.7 Adding Target Events to Source Dataset

From the above experiments, it is evident that by creating texture-invariant models or training models with synthesised data, the performance in the target domain can be improved and here it was shown to obtain a boost of ~ 0.04 ; however, further improvements would be necessary for the models to become operational. The reduced performance is attributed to the fact that the structure of the new domain is different in many cases, and this cannot be solved in an unsupervised manner because the structure of a subsea pipeline image is critical for label inference. An approach is to manually annotate a segment of the target domain data and use them to retrain the model. The relationship between the amount of annotated data and performance is not clear, and a set of experiments was devised to gain insight into what the potential performance benefits would be. In these experiments, a random 50% of events from the target survey are kept as a hold-out test set, and the remaining 50% are progressively added to the training dataset in 5% increments from 0% to 50%. Figure 6.18 presents the number of events per survey. Then, 11 models are trained based on these hybrid datasets, with the same training routine as used throughout this Chapter, and their performance is measured on the hold-out target test set. Furthermore, the FID scores are calculated between the hybrid and target test sets.

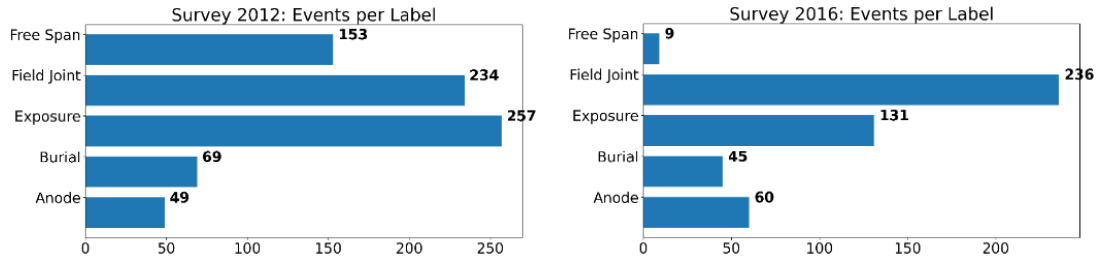


Figure 6.18: Survey Events

Figure 6.19 shows how the performance in the target test set (orange) improves as events from the target survey are added to the source survey training set, creating a hybrid set, as well as how the performance of the source test set is affected by that. Furthermore, it shows how the FID scores (red dashed line) decrease between

the hybrid and target sets as more events from the target survey are added to the source training set. This decrease is in agreement with the improved performance of the trained models. An interesting observation is that for both surveys there is a steep increase until 20% of target events are added to the source dataset where the F1-Score rises from ~ 0.58 to ~ 0.81 and then it remains constant with small fluctuations, which means that adding only 20% is enough to create a model that generalises between surveys. For the F1-Score of the source test sets, there is a very slight increase from 0.859 to 0.870, and from 0.806 to 0.819 in the test set of the surveys of 2012 and 2016, respectively, indicating that adding events from another survey can also have a positive effect in the source survey, as the model learns more general features. In the case of Figure 6.19b the performance of the model in the 2012 survey test set increases more than the performance in the 2016 survey.

Finally, when 20% of the target events are added to the source datasets, where the performance in the two surveys reaches above 0.80, the FID is ~ 80 . This shows that by calculating the FID scores, it is possible to obtain an estimate of how many events from the target survey need to be annotated and added to the source dataset to enable a model to perform optimally in a new target survey, without having to train new models, which is more time-consuming.

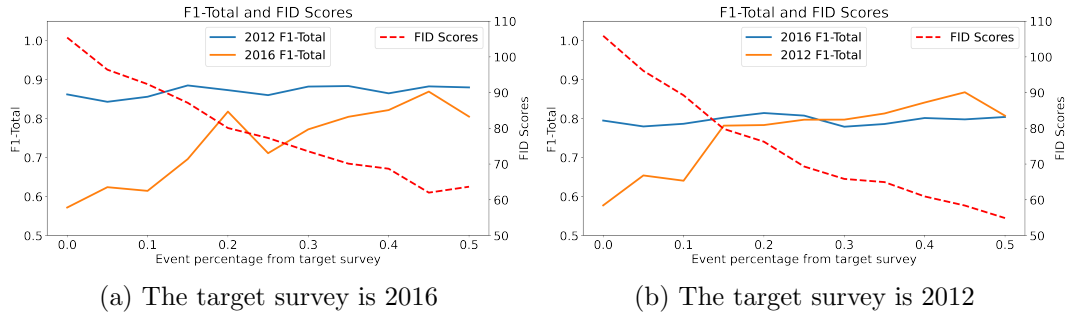


Figure 6.19: Aggregate F1-Score (F1-Total) on the source (blue) and target (orange) test sets by adding target events to the source training set plus FID scores between the hybrid and target datasets.

6.8 Conclusions

Domain shifts are particularly challenging in real-world datasets recorded in dynamic environments. Models implemented to classify subsea pipeline events have been shown to generalise poorly between surveys. To this day, there is no work that investigates the model transferability from one survey to another in the subsea context.

This Chapter presents a novel two-step domain adaptation methodology specifically designed for the subsea survey context. The methodology combines the Swapping Autoencoder (SAE) architecture with a classifier module, and incorporates data synthesis and retraining techniques to effectively leverage texture information from the target inspection dataset. Importantly, this research addresses a significant gap in the literature by being the first to explore domain adaptation in the subsea survey domain.

Although the proposed methods improve the performance on the target surveys, the performance of models trained in a supervised setting still offer higher performance. This is attributed to the structural differences of the events, for example, Anode. Different normalisation layers and the swapping of texture can mitigate against different imagery styles and lighting conditions. However, considering that the domain adaptation is executed in an unsupervised fashion, the performance obtained provides an initial baseline for annotations.

In addition to the proposed methodology, the study introduces a novel approach for characterizing both in-domain and out-of-domain shifts using the FID. This metric enables a quantitative assessment of the differences between the source and target domains, aiding in understanding the nature and extent of domain shift.

Furthermore, the study investigates the impact of incorporating target events into the source dataset to improve model performance on the target domain. By analyzing the percentages of target events required for enhanced performance, valuable insights are gained into the amount of manual annotation necessary for building a high-performing model in the target domain. Subsea pipeline inspection is a vital process within the oil and gas industry that combines structural condition assessment with the verification of regular containment surveys; therefore, supervised learning has been

proven to be the safest option.

In conclusion, this Chapter makes significant contributions to the field of subsea survey inspections by examining model transferability, adaptation, and domain shift. The proposed two-step domain adaptation methodology, combined with the utilization of the SAE architecture, data synthesis, and retraining, offers a robust framework for enhancing model performance in the subsea survey context. Moreover, the characterization of domain shift using FID and the analysis of target event percentages provide valuable insights into improving generalization and reducing the gap between different domains. This work stands as the first comprehensive investigation into the challenges of domain adaptation and model transferability in the subsea survey domain, paving the way for future advancements in this important area of research.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

Subsea pipeline inspection is a process of great importance for the oil and gas industry, as any potential danger can damage equipment and also pose a threat to the environment. ROVs are commonly used for inspections of subsea pipelines and power transmission cables [1]. They are submerged and driven above the pipeline, controlled through a cable that is connected to an off-shore vessel. ROVs collect inspection data from various sensors (visual, sonars / echosounders, laser scanning, magneto-metric sensors) [311]. Data are transmitted to offshore personnel to determine the overall condition of the pipeline and ensure that the installation is acceptable. Following the inspection video, the Data Coordinators record logs of key events during the survey both in real time and later in QC. They use timestamps to annotate events such as pipeline Anode, Exposure, Burial, Field Joint and Free Span. However, large amounts of data can cause challenges with manual annotation, as it is prone to human error, which is where the application of automatic annotation becomes necessary.

The thesis presented an approach towards the automation of the annotation process which enables fewer personnel working off-shore, increasing inspection speed, and hence reducing the survey cost, and increasing safety. In general, the techniques proposed in

the thesis will allow for more robust, accurate, and faster inspections. The impact of this work in practical contexts is that it develops the potential for an intelligent decision support tool to be used in conjunction with human operators to improve decision-making in the annotation process of subsea pipelines.

Chapter 2 presented work using DL in power line, manufacturing and subsea pipeline inspection processes and other underwater applications, along with the current challenges of underwater imaging that arise as a consequence of the dynamic subsea environment. Following this work, it was made evident that DL is the current approach in different fields, as the increasing computational power today can afford it. The methodology Chapter 3 outlined the basic building blocks of CNNs, along with techniques that were used in this thesis to train and evaluate the performance of the models developed. In addition, it concluded with a case study on a subsea pipeline image classification exploring multi-class image classification which provided motivation for the research question of Chapter 4.

Chapter 4 addresses the research question of whether events occurring simultaneously in frames of subsea pipeline surveys can be recognized at the same time using a Deep Learning (DL) image classifier. By proposing a framework that converts multi-class into multi-label image classification and utilising threshold search with Precision-Recall (PR) curves, the study aims to strike a balance between precision and recall. Multiple residual models of varying size have been tested, and the ResNet-50 architecture has been shown to balance the trade-off between performance and computation inference time.

This work represents the first multi-label approach applied to subsea video inspection, marking a significant contribution to the field. The investigation emphasizes the importance of dataset splitting based on events rather than frames, highlighting the need for measuring accurate generalisation performance. The results and insights gained from this study provide valuable guidelines for practitioners in the subsea inspection domain, helping to improve the accuracy and efficiency of event recognition in real-time subsea pipeline surveys.

Chapter 5 addresses the issue of fluctuated predictions in subsea survey videos

caused by using an image classifier on individual video frames. To mitigate this problem and improve classification accuracy, the study focusses on models that operate directly on frame sequences. The research question pertains to determining the most suitable model for this application, which effectively fuses spatial and temporal information. To this end, three models are developed and compared in terms of performance and computational requirements. The results indicated that the 2D model can outperform its spatio-temporal counterparts particularly for short events (i.e. Anode and Field Joint). Experimentation showed that a larger dataset would have been beneficial for the 3D counterparts at the additional expense of manual annotation.

This work represents a significant contribution as it is the first to operate directly on subsea pipeline video clips, rather than individual frames. The subsea survey video dataset created for this work can provide a benchmark for subsequent studies. In addition, the introduction of the novel 3D IBN-Net further differentiates this research, as it has not been employed in any previous work. The findings of this study offer valuable information to practitioners in the field, helping to select appropriate models to enhance the accuracy of classification and reduce fluctuations in subsea survey videos.

Chapter 6 addresses the concerns surrounding the transferability of DL models in the integration of an automatic annotation framework for the inspection process, particularly in the context of subsea surveys. The research investigates whether model architectures or datasets can be adapted to sustain performance between different domains. Furthermore, the study explores methods to measure the shift between two survey datasets and determine the number of manually annotated events required for building a high-performing model in the target domain.

To address these challenges, the study proposes a two-step domain adaptation methodology that combines the Swapping Autoencoder architecture with a classifier module. This approach leverages data synthesis and retraining to incorporate texture information from the target inspection dataset. Notably, this work represents the first exploration of domain adaptation in the context of subsea surveys, highlighting its novelty and significance. Additionally, the study introduces a methodology for characterizing in- and out-of-domain shift using FID scores. This measure provides insights

into the dissimilarity between the source and target datasets, facilitating a better understanding of the domain shift. Furthermore, the research investigates the percentage of target events that need to be added to the source dataset to improve model performance on the target domain. This analysis offers practical guidance on the amount of manual annotation required to build a high-performing model for a specific domain.

In summary, Chapter 6 contributes a comprehensive two-step domain adaptation methodology tailored for subsea survey applications, along with a novel approach for characterizing domain shift using FID. The findings provide valuable insights for practitioners seeking to integrate DL models into the inspection process and improve performance across different survey domains.

All models developed in this study have the potential to be deployed on a remotely operated vehicle (ROV) for real-time event detection. By leveraging these models on an ROV platform, operators can benefit from simultaneous event detection, enhancing the efficiency and accuracy of inspections in real-time scenarios. The ability to deploy these models on an ROV represents a significant practical implication, as it enables the integration of advanced deep learning techniques into existing inspection systems, ultimately leading to improved decision-making and operational outcomes.

The thesis makes contributions to methodologies that can be applied to automate visual inspection processes by identifying events of interest demonstrated in subsea pipeline surveys. In addition to their application in the subsea survey context, it is worth noting that all the methodologies discussed above can be applied to inspection processes where events occur simultaneously, irrespective of the specific domain. These methodologies hold the potential to assist in real-time inspection tasks by addressing the challenge of domain shift and enabling the sustained performance of DL models across different domains. By leveraging techniques such as multi-label image classification, video classification, domain adaptation, domain shift characterisation and determining the necessary manual annotation percentages, these methodologies provide a framework for improving classification accuracy and generalisation in real-time inspection scenarios. Consequently, these approaches have broader implications and can contribute to various inspection processes beyond the subsea domain, providing

valuable tools for practitioners aiming to deploy DL models in real-world applications with simultaneous event recognition.

More specifically, the contributions can be summarised as follows:

- Proposes a framework that converts multi-class into multi-label image classification applied to subsea video inspection, and utilises threshold search using PR curves to balance the precision and recall trade-off.
- Investigates the effect of dataset splitting, i.e. on per-event or per-frame basis, and highlights the need to split the dataset based on events to obtain accurate generalisation performance.
- Develops three models that fuse spatial and temporal information to perform classification of video segments in the subsea survey context.
- Proposes a two-step domain adaptation methodology that combines the SAE architecture with a classifier module, along with data synthesis and retraining to take advantage of the texture information from the target inspection dataset.
- Suggests a methodology for in- and out-of-domain shift characterisation using FID and further examines the percentages of target events that need to be added to the source dataset to improve performance on the target.
- Curation of datasets that could provide a benchmark for subsequent studies¹. Furthermore, as new DL models with higher performance and less computational needs become available, they can be readily trained and evaluated using the proposed datasets and methodologies developed.

One potential weakness of this work is the comparison with the state-of-the-art approaches, primarily due to the commercial nature of the dataset used. The availability of commercial datasets may impose limitations on accessing and comparing with existing state-of-the-art methods, as these methods are often evaluated on publicly available benchmark datasets. Consequently, it can be challenging to directly compare

¹Subject to data owner permission.

the proposed methodologies with the latest advancements in the field. However, despite this limitation, the study still provides valuable contributions by introducing novel approaches for multi-label image classification, video classification, domain adaptation, domain shift characterisation, and determining the required manual annotation percentages in the subsea survey context. These methodologies address important aspects of real-time inspection processes where events occur simultaneously, offering insights and techniques that can be applied to various domains, even if a direct comparison with state-of-the-art methods is not feasible due to dataset constraints. Future research efforts could focus on evaluating the proposed methodologies on publicly available datasets to further validate their performance and compare them with existing state-of-the-art approaches.

7.2 Future Work

Followed by the results and conclusions highlighted in this thesis, the potential directions for future research are summarised as follows:

- Training a model to perform automatic annotation of subsea pipelines requires a significant amount of labelled data. A promising future direction that could reduce the need for manual annotation is the use of self-supervised learning. Models such as SimCLR [312] that work based on similarity measurements can take advantage of large amounts of data from the video feed of subsea surveys without the need for labels. Additionally, to further increase the size and variability of the training dataset, publicly available underwater imagery datasets could be used along with synthetic data generated using GANs or Diffusion [313] models. Augmentation of the latent space before generation could also be investigated for the synthesis of new data. Then a small amount of subsea pipeline data could be annotated and used to fine-tune a classifier on top of the SimCLR encoder and perform a study of measuring the generalisability between surveys and datasets. Furthermore, another approach to reduce the need of manual annotation would be to use pseudo-labelling. More specifically, using a model that has been trained

on a source survey dataset to label a small subset of a target survey dataset, allows for acquiring more labelled data, which can in turn be used to retrain a new model.

- During most of the duration of a subsea pipeline survey, pipelines are exposed or buried beneath the seabed, whereas all other events occur less frequently. Therefore, the problem of automatic annotation of subsea pipelines could potentially be treated as an anomaly detection problem where a model is trained to identify the Exposure and Burial events that occur more frequently during a survey plus a third class of been unknown with an uncertainty percentage. Then, the unknown event or anomaly could be identified using separate smaller models that have been trained in a Siamese [314] fashion to measure similarities or differences with events that do not occur as frequently in a survey, such as Anodes, for example.
- Underwater images sometimes are very blurry as a result of the sand particles constantly moving due to waves, currents, and ROV motion. For this reason, during a subsea survey inspection, human annotators are dependent not only on the video feed of the ROV, but also on other sensors. For example, by combining the data camera and a sonar, an annotator can create a detailed map of the sea floor, highlighting potential hazards or areas of interest. Therefore, a multi-modal approach could be investigated in which the developed model can take as input both imaging and other sensory data such as MBE and fuse them, to improve the annotation performance. The fusion of the different sensors would require further investigation.
- Another study could investigate the deployment of the developed DL models on edge devices to produce real-time operations during a subsea survey along with the trade-off between the performance of the model, its computational efficiency, and the hardware requirements. The specific hardware requirements will depend on the complexity and size of the model, as well as the desired performance. They typically consist of a powerful Central Processing Unit (CPU) or GPU, as well as enough memory to store the model, and its inputs and devices, such as

Chapter 7. Conclusions and Recommendations

NVIDIA Jetson Nano, could be a potential solution. In addition, techniques that produce more computationally efficient models can be evaluated, such as model compression (pruning, quantisation, and low-rank factorisation), more efficient model architectures, such as a MobileNet [315], and distillation [316] where a smaller student model is trained to mimic the behaviour of a larger teacher model.

Bibliography

- [1] M. Ho, S. El-Borgi, D. Patil, and G. Song, “Inspection and monitoring systems subsea pipelines: {A} review paper,” *Structural Health Monitoring*, p. 147592171983771, apr 2019. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/1475921719837718>
- [2] A. G. Rumson, “The application of fully unmanned robotic systems for inspection of subsea pipelines,” *Ocean Engineering*, vol. 235, p. 109214, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0029801821006442>
- [3] P. Zingaretti and S. M. Zanolì, “Robust real-time detection of an underwater pipeline,” *Engineering Applications of Artificial Intelligence*, vol. 11, no. 2, pp. 257–268, Apr. 1998. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0952197697000018>
- [4] A. Stamoulakatos, J. Cardona, C. McCaig, D. Murray, H. Filius, R. Atkinson, X. Bellekens, C. Michie, I. Andonovic, P. Lazaridis, A. Hamilton, M. Hossain, G. Caterina, and C. Tachtatzis, “Automatic annotation of subsea pipelines using deep learning,” *Sensors (Switzerland)*, vol. 20, no. 3, 2020.
- [5] A. Stamoulakatos, J. Cardona, C. Michie, I. Andonovic, P. Lazaridis, X. Bellekens, R. Atkinson, M. M. Hossain, and C. Tachtatzis, “A comparison of the performance of 2d and 3d convolutional neural networks for subsea survey video classification,” in *OCEANS 2021: San Diego – Porto*, 2021, pp. 1–10.
- [6] J. A. Dowdeswell and R. D. Powell, “Instruments and methods: Submersible remotely operated vehicles (rovs) for investigations of the glacier-ocean-sediment interface,” *Journal of Glaciology*, vol. 42, no. 140, p. 176–183, 1996.
- [7] C. Mai, S. Pedersen, L. Hansen, K. L. Jepsen, and Z. Yang, “Subsea Infrastructure Inspection : A Review Study,” pp. 71–76, 2016.
- [8] “Seagrass meadows (*Posidonia oceanica*) distribution and trajectories of change,” *Scientific Reports*, vol. 5, pp. 1–14, 2015.
- [9] J. Evans, Y. Petillot, P. Redmond, M. Wilson, and D. Lane, “AUTOTRACKER: AUV embedded control architecture for autonomous pipeline and cable tracking,” *Oceans Conference Record (IEEE)*, vol. 5, pp. 2651–2658, 2003.
- [10] Y. Petillot, S. Reed, and J. Bell, “Real time AUV pipeline detection and tracking using side scan sonar and multi-beam echo-sounder,” in *Oceans ’02 MTS/IEEE*, vol. 1. IEEE, 2002, pp. 217–222. [Online]. Available: <http://ieeexplore.ieee.org/document/1193275/>
- [11] C. P. Robert, “The Metropolis – Hastings algorithm,” no. Mcmc, pp. 1–15, 1994.

Bibliography

- [12] A. M. Pavin, "The pipeline identification method basing on auv's echo-sounder data," in *OCEANS 2006*, 2006, pp. 1–6.
- [13] Y. Pailhas, C. Capus, K. Brown, and P. Moore, "Analysis and classification of broadband echoes using bio-inspired dolphin pulses," *The Journal of the Acoustical Society of America*, vol. 127, no. 6, pp. 3809–3820, 2010.
- [14] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014.
- [15] V. Bharti, D. Lane, and S. Wang, "Robust Subsea Pipeline Tracking with Noisy Multibeam Echosounder," *AUV 2018 - 2018 IEEE/OES Autonomous Underwater Vehicle Workshop, Proceedings*, 2018.
- [16] J. Hallset, "Simple vision tracking of pipelines for an autonomous underwater vehicle," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 2767–2772 vol.3.
- [17] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.
- [18] A. Grau, J. Climent, and J. Aranda, "Real-time architecture for cable tracking using texture descriptors," *Oceans Conference Record (IEEE)*, vol. 3, pp. 1496–1500, 1998.
- [19] J. Antich and A. Ortiz, "Underwater cable tracking by visual feedback," in *Pattern Recognition and Image Analysis*, F. J. Perales, A. J. C. Campilho, N. P. de la Blanca, and A. Sanfeliu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 53–61.
- [20] D. L. Rizzini, F. Kallasi, F. Oleari, and S. Caselli, "Investigation of vision-based underwater object detection with multiple datasets," *International Journal of Advanced Robotic Systems*, vol. 12, 2015.
- [21] A. Khan, S. S. A. Ali, F. Meriaudeau, A. S. Malik, L. S. Soon, and T. N. Seng, "Visual feedback-based heading control of autonomous underwater vehicle for pipeline corrosion inspection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, pp. 1–13, 2017.
- [22] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.
- [23] A. Khan, S. S. A. Ali, A. Anwer, S. H. Adil, and F. Meriaudeau, "Subsea pipeline corrosion estimation by restoring and enhancing degraded underwater images," *IEEE Access*, vol. 6, pp. 40 585–40 601, 2018.
- [24] G. Conte, S. Zanolli, A. M. Perdon, G. Tascini, and P. Zingaretti, "Automatic analysis of visual data in submarine pipeline inspection," in *OCEANS 96 MTS/IEEE Conference Proceedings. The Coastal Ocean - Prospects for the 21st Century*, vol. 3, Sep. 1996, pp. 1213–1219 vol.3.
- [25] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina, USA, Tech. Rep., 1995.
- [26] A. Ortiz, M. Simó, and G. Oliver, "A vision system for an underwater cable tracker," *Machine Vision and Applications*, vol. 13, no. 3, pp. 129–140, Jul. 2002. [Online]. Available: <http://link.springer.com/10.1007/s001380100065>

Bibliography

- [27] M. Asif and M. Rizal, “An Active Contour and Kalman Filter for Underwater Target Tracking and Navigation,” in *Mobile Robots: towards New Applications*, A. Lazinica, Ed. I-Tech Education and Publishing, December 2006. [Online]. Available: http://www.intechopen.com/books/mobile_robots_towards_new_applications/an_active_contour_and_kalman_filter_for_underwater_target_tracking_and_navigation
- [28] M. Narimani, S. Nazem, and M. Loueipour, “Robotics vision-based system for an underwater pipeline and cable tracker,” in *OCEANS 2009-EUROPE*. Bremen, Germany: IEEE, May 2009, pp. 1–6. [Online]. Available: <http://ieeexplore.ieee.org/document/5278327/>
- [29] T.-d. Zhang, W.-j. Zeng, L. Wan, and Z.-b. Qin, “Vision-based system of AUV for an underwater pipeline tracker,” *China Ocean Engineering*, vol. 26, no. 3, pp. 547–554, sep 2012. [Online]. Available: <http://link.springer.com/10.1007/s13344-012-0041-1>
- [30] P. Drews, V. Kuhn, and S. Gomes, “Tracking system for underwater inspection using computer vision,” *Proceedings - 2012 International Conference on Offshore and Marine Technology: Science and Innovation, NAVTEC 2012*, pp. 27–30, 2012.
- [31] M. Jacobi and D. Karimanzira, “Underwater pipeline and cable inspection using autonomous underwater vehicles,” in *2013 MTS/IEEE OCEANS - Bergen*. Bergen: IEEE, Jun. 2013, pp. 1–6. [Online]. Available: <http://ieeexplore.ieee.org/document/6608089/>
- [32] —, “Multi sensor underwater pipeline tracking with AUVs,” in *2014 Oceans - St. John's*. St. John's, NL: IEEE, Sep. 2014, pp. 1–6. [Online]. Available: <http://ieeexplore.ieee.org/document/7003013/>
- [33] F. R. Petraglia, R. Campos, J. G. R. C. Gomes, and M. R. Petraglia, “Pipeline tracking and event classification for an automatic inspection vision system,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. Baltimore, MD, USA: IEEE, May 2017, pp. 1–4. [Online]. Available: <http://ieeexplore.ieee.org/document/8050761/>
- [34] C. Liu, H. Li, S. Wang, M. Zhu, D. Wang, X. Fan, and Z. Wang, “A Dataset and Benchmark of Underwater Object Detection for Robot Picking,” *2021 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2021*, 2021.
- [35] C. Liu, Z. Wang, S. Wang, T. Tang, Y. Tao, C. Yang, H. Li, X. Liu, and X. Fan, “A New Dataset, Poisson GAN and AquaNet for Underwater Object Grabbing,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2831–2844, 2022.
- [36] K. Katija, E. Orenstein, B. Schlining, L. Lundsten, K. Barnard, G. Sainz, O. Boulais, M. Cromwell, E. Butler, B. Woodward, and K. L. Bell, “Fathom-Net: A global image database for enabling artificial intelligence in the ocean,” *Scientific Reports*, vol. 12, no. 1, pp. 1–19, 2022.
- [37] A. Saleh, I. H. Laradji, D. A. Konovalov, M. Bradley, D. Vazquez, and M. Sheaves, “A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis,” *Scientific Reports*, vol. 10, no. 1, pp. 1–10, 2020.

Bibliography

- [38] J. Hong, M. Fulton, and J. Sattar, “TrashCan: A Semantically-Segmented Dataset towards Visual Detection of Marine Debris,” pp. 1–6, 2020. [Online]. Available: <http://arxiv.org/abs/2007.08097>
- [39] M. J. Islam, C. Edge, Y. Xiao, P. Luo, M. Mehtaz, C. Morse, S. S. Enan, and J. Sattar, “Semantic segmentation of underwater imagery: Dataset and benchmark,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 1769–1776, 2020.
- [40] L. Peng, C. Zhu, and L. Bian, “U-shape Transformer for Underwater Image Enhancement,” vol. 14, no. 8, pp. 290–307, 2023.
- [41] C. Li, C. Guo, W. Ren, R. Cong, J. Hou, S. Kwong, and D. Tao, “An Underwater Image Enhancement Benchmark Dataset and beyond,” *IEEE Transactions on Image Processing*, vol. 29, pp. 4376–4389, 2020.
- [42] J. Han, M. Shoeiby, T. Malthus, E. Botha, J. Anstee, S. Anwar, R. Wei, M. A. Armin, H. Li, and L. Petersson, “Underwater Image Restoration via Contrastive Learning and a Real-World Dataset [†],” *Remote Sensing*, vol. 14, no. 17, pp. 1–15, 2022.
- [43] Y. Sato, T. Ueda, and Y. Tanaka, “Marine snow removal benchmarking dataset,” 2021.
- [44] Y. Guo, H. Li, and P. Zhuang, “Underwater Image Enhancement Using a Multiscale Dense Generative Adversarial Network,” *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 862–870, 2020.
- [45] F. Bonnin-Pascual and A. Ortiz, “A novel approach for defect detection on vessel structures using saliency-related features,” *Ocean Engineering*, vol. 149, pp. 397 – 408, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0029801817304778>
- [46] M. O’Byrne, V. Pakrashi, F. Schoefs, and a. B. Ghosh, “Semantic Segmentation of Underwater Imagery Using Deep Networks Trained on Synthetic Imagery,” *Journal of Marine Science and Engineering*, vol. 6, no. 3, p. 93, Aug. 2018. [Online]. Available: <http://www.mdpi.com/2077-1312/6/3/93>
- [47] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [48] L. Stanchev, H. Egbert, and B. Ruttenberg, “Automating Deep-Sea Video Annotation Using Machine Learning,” in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. San Diego, CA, USA: IEEE, Feb. 2020, pp. 17–24. [Online]. Available: <https://ieeexplore.ieee.org/document/9031469/>
- [49] M. Martin-Abadal, E. Guerrero-Font, F. Bonin-Font, and Y. Gonzalez-Cid, “Deep semantic segmentation in an auv for online posidonia oceanica meadows identification,” *IEEE Access*, vol. 6, pp. 60 956–60 967, 2018.
- [50] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.
- [51] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” 2015.

Bibliography

- [52] F. Bonin-Font, M. M. Campos, and G. O. Codina, "Towards visual detection, mapping and quantification of posidonia oceanica using a lightweight auv," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 500 – 505, 2016, 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2405896316320754>
- [53] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick, and R. Fisher, "Automatic annotation of coral reefs using deep learning," in *OCEANS 2016 MTS/IEEE Monterey*. Monterey, CA, USA: IEEE, Sep. 2016, pp. 1–5. [Online]. Available: <http://ieeexplore.ieee.org/document/7761105/>
- [54] A. King, S. M. Bhandarkar, and B. M. Hopkinson, "A Comparison of Deep Learning Methods for Semantic Segmentation of Coral Reef Survey Images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 1475–14758. [Online]. Available: <https://ieeexplore.ieee.org/document/8575347/>
- [55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [56] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.
- [57] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.
- [58] M. Jeon, Y. Lee, Y.-S. Shin, H. Jang, and A. Kim, "Underwater Object Detection and Pose Estimation using Deep Learning," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 78–81, 2019. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2405896319321718>
- [59] M. Fulton, J. Hong, M. J. Islam, and J. Sattar, "Robotic Detection of Marine Litter Using Deep Visual Detection Models," *arXiv:1804.01079 [cs]*, Sep. 2018, arXiv: 1804.01079. [Online]. Available: <http://arxiv.org/abs/1804.01079>
- [60] W. Xu and S. Matzner, "Underwater Fish Detection Using Deep Learning for Water Power Applications," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. Las Vegas, NV, USA: IEEE, Dec. 2018, pp. 313–318. [Online]. Available: <https://ieeexplore.ieee.org/document/8947884/>
- [61] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, 2016.
- [62] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

Bibliography

- [63] Y. Xie, Z. Yu, X. Yu, and B. Zheng, "Lighting the darkness in the sea: A deep learning model for underwater image enhancement," *Frontiers in Marine Science*, vol. 9, no. August, pp. 1–16, 2022.
- [64] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: <https://arxiv.org/abs/1505.04597>
- [65] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016. [Online]. Available: <https://arxiv.org/abs/1603.08155>
- [66] X. Chen, P. Zhang, L. Quan, C. Yi, and C. Lu, "Underwater Image Enhancement Algorithm Combining Deep Learning and Image Formation Model," *Jisuanji Gongcheng/Computer Engineering*, vol. 48, no. 2, pp. 243–249, 2022.
- [67] G. L. Foresti and S. Gentili, "Vision based system for object detection in underwater images," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 2, pp. 167–188, 2000.
- [68] G. W. Thum, S. H. Tang, S. A. Ahmad, and M. Alrifay, "Toward a Highly Accurate Classification of Underwater Cable Images via Deep Convolutional Neural Network," 2020.
- [69] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.
- [70] E. Medina, R. Campos, G. R. C. Gomes, M. R. Petraglia, and A. Petraglia, "Convolutional Neural Networks for Underwater Pipeline Segmentation using Imperfect Datasets," pp. 1585–1589, 2020.
- [71] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.
- [72] V. Bharti, D. Lane, and S. Wang, "Learning to Detect Subsea Pipelines with Deep Segmentation Network and Self-Supervision," *2020 Global Oceans 2020: Singapore - U.S. Gulf Coast*, 2020.
- [73] Y. Li, M. Wu, J. Guo, and Y. Huang, "A Strategy of Subsea Pipeline Identification with Sidescan Sonar based on YOLOV5 Model," *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, no. Iccas, pp. 500–505, 2021.
- [74] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios," 2021.
- [75] B. Gašparović, J. Lerga, G. Mauša, M. Ivašić-kos, B. Gašparović, J. Lerga, G. Mauša, M. Ivašić-kos, B. Gašparović, and G. Mauša, "Deep Learning Approach For Objects Detection in Underwater Pipeline Images Underwater Pipeline Images," *Applied Artificial Intelligence*, vol. 36, no. 01, 2022. [Online]. Available: <https://doi.org/10.1080/08839514.2022.2146853>
- [76] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.
- [77] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

Bibliography

- [78] X. Liu, X. Miao, H. Jiang, and J. Chen, "Data analysis in visual power line inspection: An in-depth review of deep learning for component detection and fault diagnosis," *Annual Reviews in Control*, vol. 50, p. 253–277, 2020. [Online]. Available: <http://dx.doi.org/10.1016/j.arcontrol.2020.09.002>
- [79] V. N. Nguyen, R. Jenssen, and D. Roverso, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 107 – 120, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0142061517324444>
- [80] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2
- [81] V. N. Nguyen, R. Jenssen, and D. Roverso, "Intelligent Monitoring and Inspection of Power Line Components Powered by UAVs and Deep Learning," *IEEE Power and Energy Technology Systems Journal*, vol. 6, no. 1, pp. 11–21, 2019.
- [82] W. Zhang, C. Witharana, W. Li, C. Zhang, X. Li, and J. Parent, "Using deep learning to identify utility poles with crossarms and estimate their locations from google street view images," *Sensors (Switzerland)*, 2018.
- [83] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.
- [84] B. Jalil, G. R. Leone, M. Martinelli, D. Moroni, M. A. Pascali, and A. Berton, "Fault Detection in Power Equipment via an Unmanned Aerial System Using Multi Modal Data," *Sensors*, 2019.
- [85] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN," 2017.
- [86] X. Miao, X. Liu, J. Chen, S. Zhuang, J. Fan, and H. Jiang, "Insulator detection in aerial images for transmission line inspection using single shot multibox detector," *IEEE Access*, 2019.
- [87] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.
- [88] N. Wan, X. Tang, S. Liu, J. Chen, K. Guo, L. Li, and S. Liu, "Transmission Line Image Object Detection Method Considering Fine-Grained Contexts," *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, no. Itnec, pp. 499–502, 2020.
- [89] S. Vemula and M. Frye, "Mask R-CNN powerline detector: A deep learning approach with applications to a UAV," *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2020-Octob, pp. 1–6, 2020.
- [90] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.
- [91] A. Jiang, N. Yan, B. Shen, C. Gu, H. Zhu, and H. Huang, "Research on Infrared Image Recognition Method of Power Equipment Based on Deep Learning," *7th IEEE International Conference on High Voltage Engineering and Application, ICHVE 2020 - Proceedings*, pp. 12–15, 2020.

Bibliography

- [92] R. Han, L. Liu, S. Liu, P. Jiang, Y. Han, and Z. Yang, "Research and application of substation intelligent inspection technology based on multi spectral image recognition," *7th IEEE International Conference on High Voltage Engineering and Application, ICHVE 2020 - Proceedings*, pp. 12–15, 2020.
- [93] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
- [94] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges," *Materials*, vol. 13, no. 24, pp. 1–23, 2020.
- [95] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals*, vol. 65, no. 1, pp. 417–420, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0007850616300725>
- [96] M. J., M. U., C. D., S. J., and F. G., "Steel defect classification with max-pooling convolutional neural networks [c] neural networks (ijcnn)," *The 2012 International Joint Conference on Neural Networks (IJCNN)*, p. 1 – 6, 2012, cited by: 5. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091730826&partnerID=40&md5=ca7435e1319fb3ee08ddd58acaf5452b>
- [97] R. Ren, T. Hung, and K. C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE Transactions on Cybernetics*, vol. 48, no. 3, pp. 929–940, 2018.
- [98] J. K. Park, B. K. Kwon, J. H. Park, and D. J. Kang, "Machine learning-based imaging system for surface defect inspection," *International Journal of Precision Engineering and Manufacturing - Green Technology*, vol. 3, no. 3, pp. 303–310, 2016.
- [99] X. Guo, L. Chen, and C. Shen, "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis," *Measurement*, vol. 93, pp. 490–502, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0263224116304249>
- [100] C. Lu, Z. Wang, and B. Zhou, "Intelligent fault diagnosis of rolling bearing using hierarchical convolutional network based health state classification," *Advanced Engineering Informatics*, vol. 32, pp. 139–151, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1474034616301148>
- [101] V. D., D. E., M. V., M. M., and F. A., "Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings," *Shock Vib*, vol. 2017, p. 1 – 29, 2017, cited by: 4. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85053418750&partnerID=40&md5=8dd563aed3b2f2be481b5f74f4a995f3>
- [102] P. Wang, Ananya, R. Yan, and R. X. Gao, "Virtualization and deep recognition for system fault classification," *Journal of Manufacturing Systems*, vol. 44, pp. 310–316, 2017, special Issue on Latest advancements in manufacturing systems at NAMRC 45. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0278612517300511>
- [103] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufer, S. Verstockt, R. Van de Walle, and S. Van Hoecke, "Convolutional neural network based fault detection for rotating machinery," *Journal of*

Bibliography

- Sound and Vibration*, vol. 377, pp. 331–345, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0022460X16301638>
- [104] Y. W. Lu, K. L. Liu, and C. Y. Hsu, “Conditional Generative Adversarial Network for Defect Classification with Class Imbalance,” *Proceedings - 2019 IEEE International Conference on Smart Manufacturing, Industrial and Logistics Engineering, SMILE 2019*, pp. 146–149, 2019.
 - [105] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” 2018.
 - [106] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014.
 - [107] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” 2018.
 - [108] X. Zheng, H. Wang, J. Chen, Y. Kong, and S. Zheng, “A Generic Semi-Supervised Deep Learning-Based Approach for Automated Surface Inspection,” *IEEE Access*, vol. 8, pp. 114 088–114 099, 2020.
 - [109] X. Zheng, S. Zheng, Y. Kong, and J. Chen, “Recent advances in surface defect inspection of industrial products using deep learning techniques,” *International Journal of Advanced Manufacturing Technology*, vol. 113, no. 1-2, pp. 35–58, 2021.
 - [110] X. J. Zhu, “Semi-supervised learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.
 - [111] D. Hendrycks*, N. Mu*, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple method to improve robustness and uncertainty under data shift,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=S1gmrxFvB>
 - [112] Z. Hamida, B. Laurent, and J. A. Goulet, “OpenIPDM: A probabilistic framework for estimating the deterioration and effect of interventions on bridges,” *SoftwareX*, vol. 18, 2022.
 - [113] Z. Tong, T. Ma, J. Huan, and W. Zhang, “Pavementscapes: a large-scale hierarchical image dataset for asphalt pavement damage segmentation,” 2022. [Online]. Available: <http://arxiv.org/abs/2208.00775>
 - [114] J. B. Haurum and T. B. Moeslund, “Sewer-ML: A multi-label sewer defect classification dataset and benchmark,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 13 451–13 462, 2021.
 - [115] “English: Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals. NOTE: the arrow head under the legend ”Axon terminal” in this png format is not rotated as in the display of the svg format. So use the png format for wiki articles.” Sep. 2018. [Online]. Available: <https://commons.wikimedia.org/wiki/File:Neuron3.png>
 - [116] P. Sharma and A. Singh, “Era of deep neural networks: A review,” in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Delhi: IEEE, Jul. 2017, pp. 1–5. [Online]. Available: <http://ieeexplore.ieee.org/document/8203938/>
 - [117] J. Zhang, “Gradient descent based optimization algorithms for deep learning models training,” 2019.

Bibliography

- [118] Y. You, I. Gitman, and B. Ginsburg, “Large Batch Training of Convolutional Networks,” *arXiv:1708.03888 [cs]*, Aug. 2017, arXiv: 1708.03888. [Online]. Available: <http://arxiv.org/abs/1708.03888>
- [119] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147. [Online]. Available: <https://proceedings.mlr.press/v28/sutskever13.html>
- [120] Z.-S. Huang and C. pei Lee, “Momentum as variance-reduced stochastic gradient,” 2022. [Online]. Available: <https://openreview.net/forum?id=kiwu8tcVf38>
- [121] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
- [122] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv:1609.04747 [cs]*, Sep. 2016, arXiv: 1609.04747. [Online]. Available: <http://arxiv.org/abs/1609.04747>
- [123] K. Nakamura, B. Derbel, K.-J. Won, and B.-W. Hong, “Learning-rate annealing methods for deep neural networks,” *Electronics*, vol. 10, no. 16, 2021. [Online]. Available: <https://www.mdpi.com/2079-9292/10/16/2029>
- [124] K. You, M. Long, J. Wang, and M. I. Jordan, “How does learning rate decay help modern neural networks?” 2020. [Online]. Available: <https://openreview.net/forum?id=r1eOnh4YPB>
- [125] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, “Demystifying learning rate policies for high accuracy training of deep neural networks,” 2019.
- [126] L. N. Smith, “Cyclical learning rates for training neural networks,” 2017.
- [127] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates,” 2018.
- [128] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep Learning for Computer Vision: A Brief Review,” *Computational Intelligence and Neuroscience*, vol. 2018, 2018.
- [129] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” 2015.
- [130] R. K. Sinha, R. Pandey, and R. Pattnaik, “Deep learning for computer vision tasks: A review,” 2018.
- [131] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.
- [132] Y. Pang, J. Lin, T. Qin, and Z. Chen, “Image-to-image translation: Methods and applications,” 2021.
- [133] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv:1603.07285 [cs, stat]*, Mar. 2016, arXiv: 1603.07285. [Online]. Available: <http://arxiv.org/abs/1603.07285>

Bibliography

- [134] vdumoulin, *Convolution arithmetic*, Dec 2021. [Online]. Available: <https://github.com/vdumoulin/conv-arithmetic>
- [135] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” 2018.
- [136] A. Panigrahi, A. Shetty, and N. Goyal, “Effect of activation functions on the training of overparametrized neural nets,” 2020.
- [137] L. Datta, “A survey on activation functions and their relation with xavier and he normal initialization,” *arXiv*, pp. 1–17, 2020.
- [138] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” 2013.
- [139] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [140] M. S. Sorower, “A literature survey on algorithms for multi-label learning,” 2010.
- [141] O. Gharroudi, H. Elghazel, and A. Aussem, “Ensemble Multi-label Classification: A Comparative Study on Threshold Selection and Voting Methods,” in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. Vietri sul Mare, Italy: IEEE, Nov. 2015, pp. 377–384. [Online]. Available: <http://ieeexplore.ieee.org/document/7372160/>
- [142] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10. Madison, WI, USA: Omnipress, 2010, p. 807–814.
- [143] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” 2017.
- [144] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” 2015.
- [145] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [146] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feed-forward neural networks,” *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.
- [147] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” 2015. [Online]. Available: <https://arxiv.org/abs/1502.01852>
- [148] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, “How does batch normalization help optimization?” 2019.
- [149] Y. Wu and K. He, “Group normalization,” 2018.
- [150] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv:1502.03167 [cs]*, Feb. 2015, arXiv: 1502.03167. [Online]. Available: <http://arxiv.org/abs/1502.03167>

Bibliography

- [151] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” 2017.
- [152] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016.
- [153] “Understanding LSTM Networks – colah’s blog.” [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [154] W. De Mulder, S. Bethard, and M.-F. Moens, “A survey on the application of recurrent neural networks to statistical language modeling,” *Computer Speech & Language*, vol. 30, no. 1, pp. 61–98, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S088523081400093X>
- [155] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” 2014. [Online]. Available: <https://arxiv.org/abs/1406.1078>
- [156] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar 2020. [Online]. Available: <http://dx.doi.org/10.1016/j.physd.2019.132306>
- [157] A. Patel, S. C. Van De Leemput, M. Prokop, B. Van Ginneken, and R. Mannesing, “Image Level Training and Prediction: Intracranial Hemorrhage Identification in 3D Non-Contrast CT,” *IEEE Access*, vol. 7, pp. 92 355–92 364, 2019.
- [158] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014. [Online]. Available: <https://arxiv.org/abs/1412.3555>
- [159] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” 2014. [Online]. Available: <https://arxiv.org/abs/1406.1078>
- [160] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [161] A. Ghods and D. J. Cook, “A survey of techniques all classifiers can learn from deep networks: Models, optimizations, and regularization,” 2019.
- [162] A. Ghods, D. J. Cook, R. Moradi, R. Berangi, and B. Minaei, “A survey of regularization strategies for deep models,” *Artificial Intelligence Review*, vol. 53, no. 6, pp. 3947–3986, 2020. [Online]. Available: <https://doi.org/10.1007/s10462-019-09784-7><http://arxiv.org/abs/1909.04791>
- [163] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” *arXiv:1711.05101 [cs, math]*, Nov. 2017, arXiv: 1711.05101. [Online]. Available: <http://arxiv.org/abs/1711.05101>
- [164] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>

Bibliography

- [165] L. Mao, “Label smoothing.” [Online]. Available: <https://leimao.github.io/blog/Label-Smoothing/>
- [166] J. Hou, H. Zeng, L. Cai, J. Zhu, J. Chen, and K. K. Ma, “Multi-label learning with multi-label smoothing regularization for vehicle re-identification,” *Neurocomputing*, vol. 345, no. February, pp. 15–22, 2019. [Online]. Available: <https://doi.org/10.1016/j.neucom.2018.11.088>
- [167] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, p. 295–316, Nov 2020. [Online]. Available: <http://dx.doi.org/10.1016/j.neucom.2020.07.061>
- [168] T. Yu and H. Zhu, “Hyper-parameter optimization: A review of algorithms and applications,” 2020.
- [169] R. JAIN, “Why “early-stopping” works as Regularization?” Feb. 2020. [Online]. Available: <https://medium.com/@rahuljain13101999/why-early-stopping-works-as-regularization-b9f0a6c2772>
- [170] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *Journal of Big Data*, vol. 6, no. 1, 2019. [Online]. Available: <https://doi.org/10.1186/s40537-019-0197-0>
- [171] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information*, vol. 11, no. 2, 2020. [Online]. Available: <https://www.mdpi.com/2078-2489/11/2/125>
- [172] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” 2016.
- [173] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, Jun. 2009, pp. 248–255. [Online]. Available: <http://ieeexplore.ieee.org/document/5206848/>
- [174] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [175] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.
- [176] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3098997.3065386>
- [177] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” 2017.
- [178] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2014. [Online]. Available: <https://arxiv.org/abs/1409.4842>
- [179] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” 2015.

Bibliography

- [180] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.
- [181] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, “A simple framework for contrastive learning of visual representations,” *CoRR*, vol. abs/2002.05709, 2020. [Online]. Available: <https://arxiv.org/abs/2002.05709>
- [182] S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2017.
- [183] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” 2017.
- [184] [Online]. Available: <https://paperswithcode.com/method/resnet>
- [185] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, Apr. 2016. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0925231215017634>
- [186] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” *arXiv:1801.06146 [cs, stat]*, Jan. 2018, arXiv: 1801.06146. [Online]. Available: <http://arxiv.org/abs/1801.06146>
- [187] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. [Online]. Available: <http://ieeexplore.ieee.org/document/5288526/>
- [188] E. Tjoa and C. Guan, “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 14, no. 8, pp. 1–21, 2020.
- [189] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, “Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks,” *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847, Mar. 2018, arXiv: 1710.11063. [Online]. Available: <http://arxiv.org/abs/1710.11063>
- [190] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” 2015.
- [191] J. Gildenblat and contributors, “Pytorch library for cam methods,” <https://github.com/jacobgil/pytorch-grad-cam>, 2021.
- [192] S. Srinivas and F. Fleuret, “Full-gradient representation for neural network visualization,” *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, pp. 1–15, 2019.
- [193] S. Desai and H. G. Ramaswamy, “Ablation-CAM: Visual explanations for deep convolutional network via gradient-free localization,” *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pp. 972–980, 2020.
- [194] M. B. Muhammad and M. Yeasin, “Eigen-CAM: Class Activation Map using Principal Components,” *Proceedings of the International Joint Conference on Neural Networks*, 2020.

Bibliography

- [195] P. T. Jiang, C. B. Zhang, Q. Hou, M. M. Cheng, and Y. Wei, “LayerCAM: Exploring hierarchical class activation maps for localization,” *IEEE Transactions on Image Processing*, vol. 30, no. 8, pp. 5875–5888, 2021.
- [196] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li, “Axiom-based grad-cam: Towards accurate visualization and explanation of cnns,” 2020.
- [197] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, “Score-CAM: Score-weighted visual explanations for convolutional neural networks,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 111–119, 2020.
- [198] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, 2020.
- [199] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: <http://jmlr.org/papers/v20/18-598.html>
- [200] H. Fang and M. Duan, “Submarine Pipelines and Pipeline Cable Engineering,” in *Offshore Operation Facilities*. Elsevier, 2014, pp. e1–e181. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/B9780123969774000068>
- [201] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [202] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [203] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *NIPS Autodiff Workshop*, 2017.
- [204] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [205] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=ryQu7f-RZ>
- [206] L. N. Smith, “Cyclical learning rates for training neural networks,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2017, pp. 464–472.
- [207] L. N. Smith, “Cyclical Learning Rates for Training Neural Networks,” *arXiv:1506.01186 [cs]*, Jun. 2015, arXiv: 1506.01186. [Online]. Available: <http://arxiv.org/abs/1506.01186>

Bibliography

- [208] S. Geisser, “The predictive sample reuse method with applications,” *Journal of the American Statistical Association*, vol. 70, no. 350, pp. 320–328, 1975. [Online]. Available: <http://www.jstor.org/stable/2285815>
- [209] G. C. Cawley and N. L. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.
- [210] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” 2020. [Online]. Available: <https://arxiv.org/abs/2008.05756>
- [211] —, “Metrics for multi-class classification: an overview,” 2020.
- [212] F. Krüger, “Activity, context, and plan recognition with computational causal behaviour models,” Ph.D. dissertation, 12 2016.
- [213] J. Opitz and S. Burst, “Macro f1 and macro f1,” 2021.
- [214] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, “Learning multi-label scene classification,” *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, Sep. 2004. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0031320304001074>
- [215] A. Zeggada and F. Melgani, “Multilabel classification of UAV images with Convolutional Neural Networks,” in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Beijing, China: IEEE, Jul. 2016, pp. 5083–5086. [Online]. Available: <http://ieeexplore.ieee.org/document/7730325/>
- [216] Y. Liu, L. Sheng, J. Shao, J. Yan, S. Xiang, and C. Pan, “Multi-Label Image Classification via Knowledge Distillation from Weakly-Supervised Detection,” *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, pp. 700–708, 2018, arXiv: 1809.05884. [Online]. Available: <http://arxiv.org/abs/1809.05884>
- [217] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. F. Wang, “Order-free rnn with visual attention for multi-label classification,” 2017.
- [218] “Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names.” [Online]. Available: https://gombru.github.io/2018/05/23/cross_entropy_loss/
- [219] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017, pp. 618–626. [Online]. Available: <http://ieeexplore.ieee.org/document/8237336/>
- [220] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, vol. 8689, pp. 818–833. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10590-1_53
- [221] M. S. Sorower, “A literature survey on algorithms for multi-label learning,” Oregon State University, Corvallis, Tech. Rep., 2010.

Bibliography

- [222] R. Al-Otaibi, P. Flach, and M. Kull, “Multi-label Classification: A Comparative Study on Threshold Selection Methods,” *First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD 2014*, 2014.
- [223] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, “Finding the Best Classification Threshold in Imbalanced Classification,” *Big Data Research*, vol. 5, pp. 2–8, 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.bdr.2015.12.001>
- [224] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, “Thresholding classifiers to maximize f1 score,” 2014.
- [225] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PLOS ONE*, vol. 10, no. 3, pp. 1–21, 03 2015. [Online]. Available: <https://doi.org/10.1371/journal.pone.0118432>
- [226] P. A. Flach and M. Kull, “Precision-recall-gain curves: Pr analysis done right,” in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’15. Cambridge, MA, USA: MIT Press, 2015, pp. 838–846. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2969239.2969333>
- [227] M. L. Patten, “Stratified Random Sampling,” *Understanding Research Methods*, pp. 67–68, 2021.
- [228] S. Raschka, “Model evaluation, model selection, and algorithm selection in machine learning,” 2020.
- [229] Q. S. Xu and Y. Z. Liang, “Monte Carlo cross validation,” *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1–11, 2001.
- [230] M.-L. Zhang and Z.-H. Zhou, “A Review on Multi-Label Learning Algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014. [Online]. Available: <http://ieeexplore.ieee.org/document/6471714/>
- [231] Y. Yang, “An evaluation of statistical approaches to text categorization,” *Inf. Retr.*, vol. 1, no. 1-2, pp. 69–90, May 1999. [Online]. Available: <https://doi.org/10.1023/A:1009982220290>
- [232] M. Heydarian, T. E. Doyle, and R. Samavi, “Mlcm: Multi-label confusion matrix,” *IEEE Access*, vol. 10, pp. 19 083–19 095, 2022.
- [233] T. Details, “One ROC Curve and Cutoff Analysis,” pp. 1–34.
- [234] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0031320396001422>
- [235] Y. Gao, Y. Ding, F. Wang, and H. Liang, “Attentional colorization networks with adaptive group-instance normalization,” *Information (Switzerland)*, vol. 11, no. 10, pp. 1–13, 2020.
- [236] H. Nam and H. E. Kim, “Batch-instance normalization for adaptively style-invariant neural networks,” *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 2558–2567, 2018.

Bibliography

- [237] X. Huang and S. Belongie, “Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 1510–1519, 2017.
- [238] X. Pan, P. Luo, J. Shi, and X. Tang, “Two at once: Enhancing learning and generalization capacities via ibn-net,” 2020.
- [239] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.
- [240] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” 2016.
- [241] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [242] I. Rodríguez-Moreno, J. M. Martínez-Otzeta, B. Sierra, I. Rodriguez, and E. Jauregi, “Video Activity Recognition: State-of-the-Art,” *Sensors*, vol. 19, no. 14, p. 3160, Jul. 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/14/3160>
- [243] J. Yu, B. Yang, J. Wang, J. Leader, D. Wilson, and J. Pu, “2D CNN versus 3D CNN for false-positive reduction in lung cancer screening,” *Journal of Medical Imaging*, vol. 7, no. 05, pp. 1–11, 2020.
- [244] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” 2012.
- [245] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, “Large-scale video classification with convolutional neural networks,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.
- [246] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, “The Kinetics Human Action Video Dataset,” *arXiv:1705.06950 [cs]*, May 2017, arXiv: 1705.06950. [Online]. Available: <http://arxiv.org/abs/1705.06950>
- [247] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, “YouTube-8M: A Large-Scale Video Classification Benchmark,” *arXiv:1609.08675 [cs]*, Sep. 2016, arXiv: 1609.08675. [Online]. Available: <http://arxiv.org/abs/1609.08675>
- [248] S. Garg, “Learning {Video} {Features} for {Multi}-label {Classification},” in *Computer {Vision} – {ECCV} 2018 {Workshops}*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, vol. 11132, pp. 325–337. [Online]. Available: http://link.springer.com/10.1007/978-3-030-11018-5_{-}30
- [249] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional Two-Stream Network Fusion for Video Action Recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, no. i, pp. 1933–1941, 2016.
- [250] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, “1d convolutional neural networks and applications: A survey,” 2019. [Online]. Available: <https://arxiv.org/abs/1905.03554>

Bibliography

- [251] K. Palanisamy, D. Singhania, and A. Yao, “Rethinking CNN Models for Audio Classification,” 2020. [Online]. Available: <http://arxiv.org/abs/2007.11154>
- [252] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, “Volumetric and multi-view cnns for object classification on 3d data,” 2016. [Online]. Available: <https://arxiv.org/abs/1604.03265>
- [253] Y. Zhao, X. Li, H. Huang, W. Zhang, S. Zhao, M. Makkie, M. Zhang, Q. Li, and T. Liu, “Four-dimensional modeling of fMRI data via spatio-temporal convolutional neural networks (ST-CNNs),” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 12, no. 3, pp. 451–460, sep 2020. [Online]. Available: <https://doi.org/10.1109%2Ftcds.2019.2916916>
- [254] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. Springer International Publishing, 2021, vol. 8, no. 1. [Online]. Available: <https://doi.org/10.1186/s40537-021-00444-8>
- [255] R. D. Singh, A. Mittal, and R. K. Bhatia, “3D convolutional neural network for object recognition: a review,” *Multimedia Tools and Applications*, vol. 78, no. 12, pp. 15 951–15 995, Jun. 2019. [Online]. Available: <http://link.springer.com/10.1007/s11042-018-6912-6>
- [256] J. Arunnehru, G. Chamundeeswari, and S. P. Bharathi, “Human Action Recognition using 3D Convolutional Neural Networks with 3D Motion Cuboids in Surveillance Videos,” *Procedia Computer Science*, vol. 133, pp. 471–477, 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1877050918310044>
- [257] S. Verma, “Understanding 1D and 3D Convolution Neural Network | Keras,” Oct. 2021. [Online]. Available: <https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610>
- [258] J. Carreira and A. Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,” *arXiv:1705.07750 [cs]*, Feb. 2018, arXiv: 1705.07750. [Online]. Available: <http://arxiv.org/abs/1705.07750>
- [259] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning Spatiotemporal Features with 3D Convolutional Networks,” *arXiv:1412.0767 [cs]*, Oct. 2015, arXiv: 1412.0767. [Online]. Available: <http://arxiv.org/abs/1412.0767>
- [260] K. Hara, H. Kataoka, and Y. Satoh, “Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE, Jun. 2018, pp. 6546–6555. [Online]. Available: <https://ieeexplore.ieee.org/document/8578783/>
- [261] —, “Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. Venice: IEEE, Oct. 2017, pp. 3154–3160. [Online]. Available: <http://ieeexplore.ieee.org/document/8265584/>
- [262] Joe Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,”

Bibliography

- in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, Jun. 2015, pp. 4694–4702. [Online]. Available: <http://ieeexplore.ieee.org/document/7299101/>
- [263] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-Term Recurrent Convolutional Networks for Visual Recognition and Description,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.
- [264] G. Varol, I. Laptev, and C. Schmid, “Long-Term Temporal Convolutions for Action Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.
- [265] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features,” *IEEE Access*, vol. 6, pp. 1155–1166, 2018. [Online]. Available: <http://ieeexplore.ieee.org/document/8121994/>
- [266] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, Jul. 2005. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0893608005001206>
- [267] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015.
- [268] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, pp. 249–259, 2018.
- [269] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002. [Online]. Available: <https://jair.org/index.php/jair/article/view/10302>
- [270] Ufoym, “Ufoym/imbalanced-dataset-sampler: A (pytorch) imbalanced dataset sampler for oversampling low frequent classes and undersampling high frequent ones.” [Online]. Available: <https://github.com/ufoym/imbalanced-dataset-sampler>
- [271] T. Kim, H. Lee, M. Cho, H. S. Lee, D. H. Cho, and S. Lee, “Learning temporally invariant and localizable features via data augmentation for video recognition,” 2020.
- [272] Okankop, “okankop/vidaug.” [Online]. Available: <https://github.com/okankop/vidaug>
- [273] P. Y. Simard, D. Steinkraus, and J. C. Platt, “0176_689_Patrice.P.Pdf,” *Microsoft Research*, no. Icdar, pp. 1–6, 2003.
- [274] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [275] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, “Calibrating deep neural networks using focal loss,” 2020.

Bibliography

- [276] R. Müller, S. Kornblith, and G. Hinton, “When does label smoothing help?” *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.
- [277] S. Godbole and S. Sarawagi, “Discriminative methods for multi-labeled classification,” in *Advances in Knowledge Discovery and Data Mining*, H. Dai, R. Srikant, and C. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 22–30.
- [278] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, “The kinetics human action video dataset,” 2017.
- [279] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Gutttag, “Better aggregation in test-time augmentation,” 2020. [Online]. Available: <https://arxiv.org/abs/2011.11156>
- [280] —, “Better aggregation in test-time augmentation,” 2021.
- [281] Y. Cui, M. Jia, T. Y. Lin, Y. Song, and S. Belongie, “Class-balanced loss based on effective number of samples,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 9260–9269, 2019.
- [282] C. X. Ling and V. S. Sheng, “Cost-Sensitive Learning and the Class Imbalance Problem,” *Encyclopedia of Machine Learning*, no. January, pp. 231–235, 2008. [Online]. Available: <http://www.springer.com/computer/ai/book/978-0-387-30768-8{%}5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.164.4418{%}&}rep=rep1{%}&}type=pdf>
- [283] A. B. nd A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, “Albumentations: fast and flexible image augmentations,” *ArXiv e-prints*, 2018.
- [284] T. Park, J.-Y. Zhu, O. Wang, J. Lu, E. Shechtman, A. A. Efros, and R. Zhang, “Swapping autoencoder for deep image manipulation,” 2020.
- [285] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, “Covariate Shift by Kernel Mean Matching,” *Dataset Shift in Machine Learning*, pp. 131–160, 2013.
- [286] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, “A brief review of domain adaptation,” 2020.
- [287] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” 2016.
- [288] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, “Deep reconstruction-classification networks for unsupervised domain adaptation,” 2016.
- [289] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” 2017.
- [290] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 95–104, 2017.

Bibliography

- [291] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, “Image to image translation for domain adaptation,” 2017.
- [292] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2242–2251, 2017.
- [293] Z. Zhang, M. Li, and J. Yu, “On the convergence and mode collapse of gan,” in *SIGGRAPH Asia 2018 Technical Briefs*, ser. SA '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3283254.3283282>
- [294] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop,” *arXiv preprint arXiv:1506.03365*, 2015.
- [295] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” 2020.
- [296] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for image restoration with neural networks,” *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.
- [297] I.-t.-i. T. A. Review and A. Alotaibi, “SS symmetry Deep Generative Adversarial Networks for,” 2020.
- [298] Y.-c. Chen and X. Xu, “Domain Adaptive Image-to-image Translation.”
- [299] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, “Domain Adaptive Faster R-CNN for Object Detection in the Wild,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3339–3348, 2018.
- [300] M. M. Cheng, X. C. Liu, J. Wang, S. P. Lu, Y. K. Lai, and P. L. Rosin, “Structure-Preserving Neural Style Transfer,” *IEEE Transactions on Image Processing*, vol. 29, pp. 909–920, 2020.
- [301] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5967–5976, 2017.
- [302] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, “Contrastive learning for unpaired image-to-image translation,” 2020.
- [303] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5018–5027.
- [304] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [305] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” 2018.
- [306] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” 2019.

Bibliography

- [307] M. Seitzer, “pytorch-fid: FID Score for PyTorch,” <https://github.com/mseitzer/pytorch-fid>, August 2020, version 0.1.1.
- [308] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, “Visda: The visual domain adaptation challenge,” 2017.
- [309] J. Wang, C. Lan, C. Liu, Y. Ouyang, W. Zeng, and T. Qin, “Generalizing to unseen domains: A survey on domain generalization,” 2021.
- [310] J. Wen, R. Liu, N. Zheng, Q. Zheng, Z. Gong, and J. Yuan, “Exploiting local feature patterns for unsupervised domain adaptation,” 2018.
- [311] C. Mai, S. Pedersen, L. Hansen, K. L. Jepsen, and Z. Yang, “Subsea infrastructure inspection: A review study,” in *2016 IEEE International Conference on Underwater System Technology: Theory and Applications (USYS)*, 2016, pp. 71–76.
- [312] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020. [Online]. Available: <https://arxiv.org/abs/2002.05709>
- [313] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” 2020. [Online]. Available: <https://arxiv.org/abs/2006.11239>
- [314] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” 2015.
- [315] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” 2017. [Online]. Available: <https://arxiv.org/abs/1704.04861>
- [316] S. Bhardwaj and M. M. Khapra, “I have seen enough: A teacher student network for video classification using fewer frames,” 2018. [Online]. Available: <https://arxiv.org/abs/1805.04668>