New Deep Learning Techniques for Image Enhancement and Classification

By

Alexander Ulrichsen

Supervised by

Professor Paul Murray & Professor Stephen Marshall

In the fulfilment of the requirement for the degree of

Doctor of Philosophy

Centre for excellence in Signal & Image Processing

Department of Electronic & Electrical Engineering

University of Strathclyde

United Kingdom

Contents

Declaration of Copyright

Acknowledgements

I would like to thank the following people for their support throughout my studies.

Thanks to my first supervisor, Paul Murray, for his supervisory support and the opportunities he has provided me throughout my PhD studies. I was first introduced to Paul through the Drug Discovery Vertically Integrated Project I was involved with during my third year of my undergraduate studies. This was an exciting project where the objective was to develop algorithms to detect Streptomyces strains that had the potential to be developed into new antibiotics from hyperspectral images. This project ignited my passion for research and inspired me to pursue a PhD after my Masters degree. I am especially grateful to Paul for providing me with the opportunity to perform my research in conjunction with Peacock Technology Limited. This industry research experience was invaluable to me and I developed many key skills in addition to making many great connections.

I would also like to thank my second supervisor, Stephen Marshall, for his support throughout my PhD studies. I am particularly grateful to Steve for putting me in touch with Moncef Gabbouj and enabling my exchange visit to Tampere University, Finland, to study Operational Neural Networks for three months.

I would like to show my great appreciation for Andrew Peacock who provided me with opportunity to perform my research alongside his company, Peacock Technology Limited. This industrial collaboration has taught me many valuable skills that will help me throughout my career and I have made many great connections with the fantastic team at Peacock Technology.

I am grateful to all the employees at Peacock Technology who have supported me throughout my studies and I would like to give a special thanks to Brian Lee. Brian has greatly supported me by providing me with the resources I needed including datasets and computing power, as well as offering a wealth of technical support and advice whenever I needed it. I have learnt many valuable skills from Brian and he has greatly helped me in my professional development.

I would like to thank Moncef Gabbouj for providing me with the opportunity to come and study with him at Tampere University, Finland, for three months. I learnt a great deal during this visit and also got to experience life in Finland.

I greatly appreciate the work that both Thomas De Kerf and David Dunphy undertook to provide me with hyperspectral image super-resolution data. These datasets require technical expertise and require a large amount of time and effort to capture. These datasets have been of great value to my research, and I am very grateful for their efforts.

I would like to show appreciation to my initial supervisor, Jinchang Ren, who sadly left the University a few months into the project, for providing me with the opportunity to undertake a PhD.

I also would like to thank my fellow students, particularly those within the HSI lab, who helped make my PhD experience as enjoyable as it was.

Finally, I would like to thank my family for supporting me throughout the years and pushing me to excel in my studies. A special mention goes to my brother Finlay who made use of his artistic and Photoshop skills to help design high-quality images for me to use in my academic writing. Another special mention goes to my mother, who kindly spent a lot of time helping me proofread my thesis.

Some of the results in this thesis were obtained using ARCHIE-WeSt High Performance Computer (www.archie-west.ac.uk).

Abstract

Deep learning has revolutionised the field of image processing, enabling significant advancements in tasks such as classification, object detection, and image enhancement. However, several critical challenges persist, hindering its broader applicability and efficiency. This thesis investigates solutions to key issues, including the generalisation of models to new classes, the scalability of deep learning systems constrained by their substantial size, and the limitations of supervised learning in acquiring labelled data at scale. Additionally, it explores innovative approaches to improve image enhancement, with a focus on reconstruction fidelity and computational efficiency. This research contributes novel model architectures, training techniques and insights to the development of robust, efficient, and versatile deep learning frameworks for image processing.

One such field where the generalisation of models is critical is the dairy industry. Automated identification of individual cattle is a valuable tool for modern dairy farming, enabling increased operational scale and the potential for advanced health and welfare monitoring systems. Existing identification methods, such as Radio Frequency Identification tags, achieve only around 90% accuracy, are prone to detachment, and require specific scanning locations. Recently, deep learning-based identification systems have gained attention for their ability to overcome these limitations. This thesis explores deep learning techniques for training cattle identification models using data acquired in a controlled milking parlour environment, aiming to enhance existing Radio Frequency Identification systems. Through similarity learning, models are trained to produce embeddings that enable identification of cows not present in the training set. Novel new class analysis is conducted to evaluate model performance in realistic scenarios where herd composition changes over time, demonstrating the generalisation capacity of this technique. Furthermore, cattle identification models trained on controlled milking parlour data excel in similar domains but struggle to generalise to free-moving barn environments, where labelling data at scale is impractical. To address this, novel self-supervised learning techniques are proposed in this thesis to facilitate domain

adaptation. These techniques leverage detection and tracking models to generate weak labels from unlabelled barn data, which are then utilised in triplet loss functions during training, achieving significant performance improvements over existing self-supervised approaches.

Another field which has seen massive advancements due to breakthroughs in deep learning is Hyperspectral imaging. Hyperspectral imaging is a valuable tool in remote sensing applications as its spectral properties offer rich insights into the materials present within each captured hyperspectral image. However, this spectral detail typically comes at the cost of reduced spatial resolution. To address this, Super-Resolution techniques are often used to recover lost spatial detail and improve the overall quality of hyperspectral images. Despite their potential, several challenges persist in this domain, including issues related to data quality caused by sensor noise and the spectral response of sensors. However, the most critical challenge specific to Super-Resolution, is the lack of paired high- and low-resolution training data. As a result, existing methods often rely on artificially generating low-resolution image pairs, leading to suboptimal performance in real-world scenarios. To address these limitations, this thesis introduces several key contributions to the field of Hyperspectral Image Super-Resolution, including a novel paired high- and low-resolution dataset, novel preprocessing techniques, and a novel analysis of models trained using synthetic downsampling methods and evaluated on the proposed datasets.

The power of deep learning models comes from their ability to learn highly complex non-linear functions. However, the non-linear components of a deep neural network come from the activation functions used between layers, meaning that to achieve the necessary non-linear complexity, networks have to be sufficiently deep, resulting in large computational demands. Self-Organised Operational Neural Networks (Self-ONNs) have recently been proposed, which tackle this issue by making the linear filters of traditional Convolutional Neural Networks (CNNs) learnable non-linear functions, meaning that the same theoretical non-linear complexity can be achieved in a much shallower network. To address computational challenges, novel Self-ONN architectures are proposed. Architectures are proposed for both cattle identification tasks as well as Hyperspectral Image Super-Resolution to demonstrate both the power and versatility of such models. The results presented in this thesis show that more parameter-efficient Self-ONN models can achieve performance on par with larger CNN models and in certain cases, even outperform them.

This thesis presents a comprehensive exploration of deep learning methodologies tailored for practical applications in image processing, offering contributions that span cattle identification, Hyperspectral Image Super-Resolution, and self-supervised learning for domain adaptation, paving the way for more robust and scalable solutions.

List of Acronyms

Adam Adaptive Moment Estimation. 57, 78, 86

BYOL Bootstrap Your Own Latent. 66, 67, 153–155, 161, 162, 167

CNN Convolutional Neural Network. 9, 10, 22–24, 28, 36–38, 40, 43, 55–57, 60–65, 68, 71, 83, 85, 87–89, 98, 102, 104, 122, 128, 153, 160, 163, 165–168

DNN Deep Neural Network. 42, 70, 141, 164

ERGAS Error Relative Global Adimensional de Synthèse. 2, 14, 50, 125, 131, 132, 195

GAN Generative Adversarial Network. 62–64

GPU Graphics Processing Unit. 55, 87

HR high-resolution. 24, 62, 95–97, 100, 106, 107, 129, 133, 135

HSI Hyperspectral Image. 2, 3, 9, 10, 13, 15, 21–26, 29, 49, 62–64, 89–91, 94, 95, 98, 99, 106–109, 116–118, 122, 124, 125, 128, 129, 132, 138, 139, 162, 164, 165, 168, 169

KNN K-Nearest Neighbours. 41, 45, 58, 59, 81, 82, 86, 87, 140, 149, 158

LR low-resolution. 22, 24, 25, 61–63, 94–97, 99, 100, 106, 107, 120, 122, 124, 128–133, 135, 139

MAE Mean Absolute Error. 131

MSE Mean Squared Error. 48, 91, 100

ONN Operational Neural Network. 2, 24, 64, 65

OOD out-of-distribution. 68, 70, 71

PSNR	Peak Signal-to-Noise Ratio. 14, 48, 115, 125, 131–133, 193
ReLU	Rectified Linear Unit. 35, 36, 57, 98, 99
RFID	Radio Frequency Identification. 8, 18, 19, 73–77, 84, 140, 142, 143, 146, 147, 151, 156, 166
RGB	Red, Green and Blue. 51, 54, 63, 98, 116, 165
RMSE	Root Mean Squared Error. 50
SAM	Spectral Angle Mapper. 14, 49, 115, 125, 131, 132, 194
Self-ONN	Self-Organised Operational Neural Network. 1, 9, 10, 17, 20, 23, 24, 26, 38–40, 65, 72, 73, 83, 85, 87–89, 98, 99, 101–104, 114, 121, 122, 129, 138, 139, 157, 160, 162–164, 166–169
SGD	Stochastic Gradient Descent. 34
SISR	Single-Image Super-Resolution. 22
SNV	Standard Normal Variate. 118–121, 125–128, 139
SR	Super-Resolution. 2, 3, 9, 10, 14, 15, 17, 23–26, 29, 49, 61–64, 89, 90, 95, 98, 100, 103, 108, 109, 116, 117, 121, 124–126, 128, 129, 132, 138, 139, 162, 164, 165, 168, 169
SSIM	Structural Similarity Index Measure. 14, 48, 49, 100, 115, 125, 131–133, 194
Tanh	Hyperbolic Tangent. 35, 36, 99, 122
YOLO	You Only Look Once. 151, 156

List of Figures

List of Tables

1 Introduction

In recent years, deep learning has revolutionised the field of image processing, enabling significant advancements in tasks such as classification, object detection, and image enhancement. Despite these breakthroughs, several challenges remain that limit the broader applicability and efficiency of these methods. This PhD thesis investigates innovative approaches to address some of these key challenges, including the generalisation of models to new classes, the scalability of deep learning systems constrained by their substantial computational demands, and the limitations imposed by supervised learning in acquiring labelled data at scale. The thesis also focuses on improving image enhancement techniques, particularly in terms of reconstruction fidelity and computational efficiency.

This research aims to develop robust, efficient, and versatile deep learning frameworks by introducing novel contributions to data preprocessing, neural network architectures, and unsupervised training algorithms, alongside the creation of a new dataset. The work focuses on practical applications, including the automated identification of individual cattle in the dairy industry and Super-Resolution techniques for Hyperspectral Imaging. These applications present significant challenges, such as dynamic class variations, data quality issues, and difficulties in acquiring labelled data. To address these challenges, this thesis proposes novel solutions, including generalised models capable of performing effectively on new classes, advanced preprocessing techniques to mitigate noise within the data, unsupervised training methods that leverage weak labels more effectively, and novel Self-Organised Operational Neural Network (Self-ONN) architectures which not only enhance performance but also improve parameter efficiency.

1.1 Application Areas

This thesis applies the proposed models and algorithms to two different application areas: individual dairy cattle identification in an agricultural technology setting; and resolution enhancement of hyperspectral images. Each of these application areas presents its own unique challenges in which specific methods have been developed for each. However, both application areas have a common theme exploring the utilisation of advanced models containing non-linear filters, which will be discussed further in Section 2.3.

1.1.1 Dairy Agritech Background

Noninvasive continual monitoring of dairy cattle to assess animal welfare and milk production is of great value to the dairy industry [2]. The advancement of such technology enables the effective management of larger herd sizes and consequently greater profits. Furthermore, such precision dairy technology could provide insight into the health of the animals and provide early signals for health conditions, consequently improving welfare.

A crucial component of such a system is the identification of individual cattle. Conventional techniques for identifying individual dairy cows have largely relied on physical contact methods, including ear tags, branding, and embedded Radio Frequency Identification (RFID) technology [3]. The ear tag and branding methods require manual identification, which provides little value to precision dairy technology in modern farming practices and often causes stress in cows [4]. Livestock farming frequently employs embedded RFID technology for individual identification [5]. Although a significant improvement over manual identification methods, it still has limitations as it requires the cows to wear electronic devices that can be lost. Furthermore, it requires the use of specific RFID readers, meaning that identification can only be performed at specific locations. These devices are typically around £2,500 per unit, so having several units to perform identification at many locations is very costly. Furthermore, it is not physically possible to use them for full identification coverage in a barn.

The advancement of artificial intelligence and deep neural networks in recent years has made the development of such a system theoretically possible through the use of non-invasive cameras and vision systems. Such a system not only eliminates the need for any artificial physical markings or identifiers to be attached to the cow but also greatly increases the effectiveness of the identification system as the process can be automated and allows for identification at any location where there is camera coverage. In an indoor barn, multiple cameras can be installed on the ceiling looking down, which can provide video, and thus identification coverage of the entire barn. Each individual camera not only provides a far greater identification radius than an RFID scanner, but is also significantly cheaper and provides far more information, which can also be used for other downstream tasks such as behaviour monitoring.

Each individual cow has unique biological characteristics that a machine learning model can exploit to perform identification [6]. Like human beings, cattle have unique faces and identification can easily be performed on close-up images of the cow's face [7]. Much like a human fingerprint, cow muzzles are unique and additionally remain consistent over time [8], making them a good region for visual identification. However, the issue with performing identification on the face or the muzzle is that it is challenging to acquire the images for identification as the cow has to be sufficiently close to the camera while also not being occluded by other objects, which is common in side-on view images in a barn setting. Holstein cows are the most common breed of dairy cows in UK and many other countries [9]. Holstein cows have distinct black and white patterns on their bodies that are unique to each individual cow, making it possible to perform identification on these patterns from a top-down view such as the one shown in Figure 1.1. Furthermore, identification can be performed from a much greater distance from the camera since the body is much larger than the face or muzzle, and occlusion is also much less likely to occur in a top-down view, greatly improving the frequency at which identification can be performed.

The ability to continually identify individual cattle from any location accurately has significant implications for the dairy industry. By enabling the continual monitoring of individual cows, such systems can provide crucial insights into health, behavior, feed intake and milk production metrics, which are essential for precision dairy farming. This not only

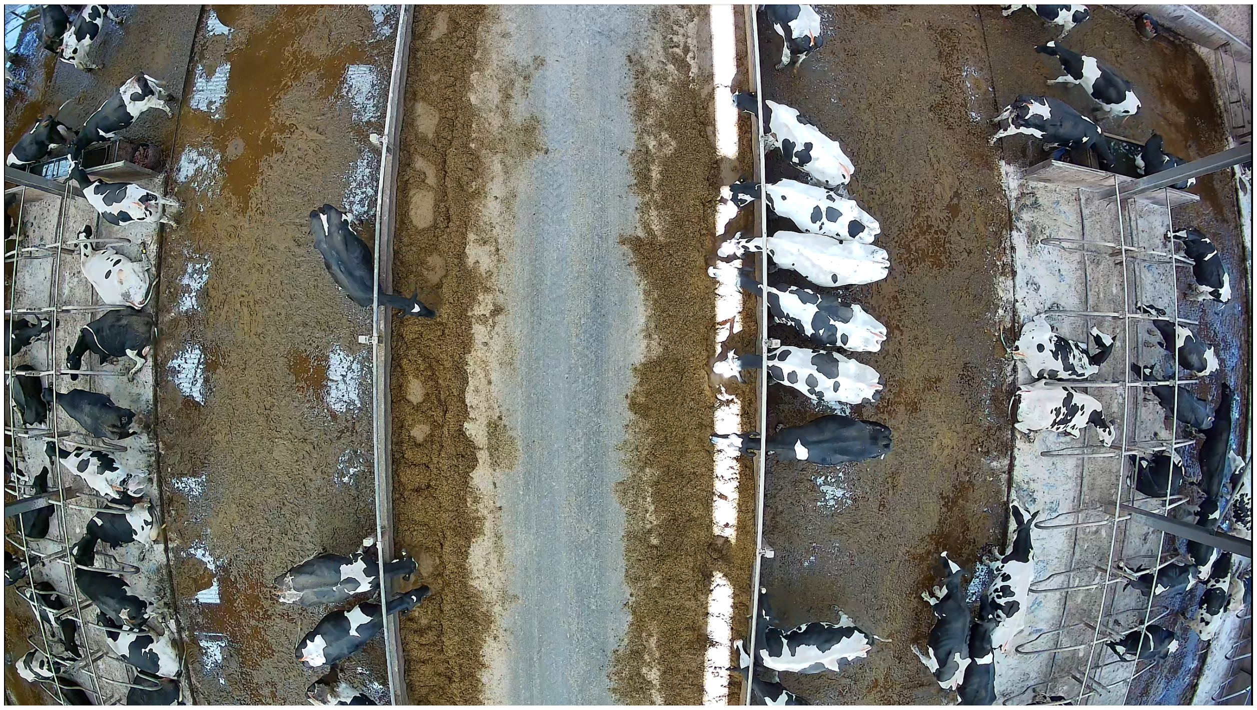

Figure 1.1: Cow Barn Image

enhances animal welfare through early detection and intervention of health conditions but also improves operational efficiency and profitability. In addition, noninvasive identification eliminates the need for physical contact or wearable devices, reducing stress and minimising the risk of loss or damage to equipment.

Developing a reliable system for cattle identification poses several challenges, including variations in environmental conditions, occlusion, and the need for scalability to accommodate large herds. This thesis addresses these challenges using top-down view data captured by cameras covering entire barns, overcoming the limitations of face- or muzzle-based identification methods and providing a practical, effective solution for real-world applications. Similarity learning techniques are employed to enable identification of cows not included in the training dataset, and novel new class analysis is conducted to assess the effectiveness of these models in practical scenarios where new cows are continuously introduced to the herd. Additionally, this work proposes novel Self-Organised Operational Neural Network (Self-ONN) architectures that enhance network parameter efficiency without compromising identification performance. To address domain adaptation challenges, self-supervised learning techniques are introduced, bridging the gap between easily labelled data from milking parlours and more challenging, unlabelled data from barns. The techniques presented in this thesis aim to advance precision dairy technology, fostering more sustainable and welfare-oriented farming practices.

1.1.2 Hyperspectral Imaging

Hyperspectral imaging has emerged as a pivotal technique in remote sensing, offering unparalleled insights into material properties. This technology operates by capturing a wide spectrum of light beyond the visible range, enabling the precise identification and analysis of materials, chemical compositions, and physical conditions within the captured scene, and is useful for many applications including agriculture, mining, and environmental monitoring [10, 11, 12].

The intrinsic value of hyperspectral imaging lies in its capacity to provide detailed spectral information for every pixel in an image. Unlike conventional imaging, which captures only three colour bands (red, green, and blue), hyperspectral sensors can capture hundreds of narrow and contiguous spectral bands. This rich spectral information is instrumental in distinguishing between materials with similar visual appearances but different spectral signatures, thereby enhancing material classification, environmental monitoring, and other critical applications [13].

However, the acquisition of high-quality Hyperspectral Images (HSIs) presents significant technical challenges. A fundamental trade-off exists between spectral and spatial resolution due to current sensor limitations. While hyperspectral sensors provide extensive spectral information, they often do so at the cost of spatial resolution. This limitation arises because increasing the spectral resolution — capturing more spectral bands — typically reduces the number of photons captured per band, leading to lower spatial resolution [14].

The diminished spatial resolution adversely impacts automated image processing tasks such as segmentation, object detection, and classification. These tasks rely on precise spatial details to accurately identify and analyse objects within the scene. The lack of high spatial resolution can lead to inaccuracies and reduced performance of these algorithms, limiting the potential of hyperspectral imaging in various applications.

To address this challenge, Single-Image Super-Resolution (SISR) has been proposed as a promising solution. SISR is an advanced image processing technique designed to enhance the spatial resolution of an image using only the available low-resolution (LR) data, without the need for additional auxiliary information. By employing sophisticated algorithms, SISR reconstructs high-resolution images from their lower-resolution counterparts, effectively recovering lost spatial details while preserving spectral integrity. This process involves complex modeling and machine learning techniques, including Convolutional Neural Networks and deep learning, which have shown great potential in learning the intricate mappings between low and high-resolution images [15, 16, 17].

The development and refinement of SISR techniques are of paramount importance in the field of hyperspectral imaging. By improving the spatial resolution of HSIs, researchers and practitioners can significantly enhance the performance of subsequent image processing tasks. Enhanced spatial details facilitate more accurate material classification, better environmental monitoring, and improved detection and analysis in a myriad of applications. As such, advancements in HSI-SISR directly contribute to the broader utilisation and effectiveness of hyperspectral imaging in tackling complex and diverse challenges across various domains.

State-of-the-art techniques for SISR predominantly leverage deep neural networks, typically training models in a supervised fashion using paired high- and low-resolution images. However, obtaining such paired datasets is particularly challenging, and as of this writing and to the best of the author's knowledge, no such dataset exists in the hyperspectral space. In contrast, unpaired single images are relatively easy to acquire, leading most existing methods to rely on artificial downsampling techniques to generate low-resolution counterparts from high-resolution images. These approaches, however, make assumptions about the true downsampling process, which is inherently unknown. Consequently, models trained on such synthetic data often suffer from diminished performance in real-world applications.

In this thesis, a novel paired HSI-SR dataset is introduced, comprising real high- and low-resolution image pairs. Novel analysis of this dataset highlights the advantages of training on genuine data pairs and the limitations of using synthetically generated downsampled data. Furthermore, a key challenge in working with HSI data — its susceptibility to noise and the spectral response characteristics of sensors — is addressed through novel preprocessing techniques. These techniques enhance the quality of the input data and improve SR performance. Finally, novel Self-ONN architectures are proposed, demonstrating significant performance improvements over traditional CNN architectures for HSI-SR.

1.2 Original Contributions of the Work

It is believed that the novel contributions of this work are:

- New class evaluation for dairy cattle identification (Chapter 4)
- Self-ONN architectures for classification tasks trained using similarity learning (Chapter 4)
- Self-ONN architectures for HSI-SR tasks (Chapter 5)
- Data normalisation strategies for HSI-SR (Chapter 5)
- Analysis of artificial downsampling techniques for HSI-SR (Chapter 5)
- Self-supervised techniques for real-world cattle identification (Chapter 6)
- Automated data gathering techniques for use in self-supervised training (Chapter 6)

These novel contributions will now be explained further.

In Chapter 4 of this thesis, similarity learning is used to train various classification and identification models. The first novel contribution is the analysis of how well models trained with this technique can identify cattle not contained within the training set, a capability that models trained with traditional classification techniques do not possess. Analysis is performed in a small-scale setting to evaluate the robustness of features learned in a limited data setting. Additionally, analysis is performed in a more realistic larger-scale setting to simulate the performance in a more practical setting.

The second novel contribution of this thesis is Self-ONN architectures to enhance the non-linear capacity of popular classification Convolutional Neural Network (CNN) models. Self-ONNs transform the linear filters of a CNN to non-linear filters, thus the classification models proposed have more powerful feature extraction capabilities. Other modifications to the networks are also made including different activation functions and initialisation strategies. Furthermore, the increased feature extraction capacity allows for a significant reduction in the number of filters required by the networks, improving the network efficiency.

In Chapter 5, more novel Operational Neural Network architectures are proposed, this time for the task of Hyperspectral Image Super-Resolution. The non-linear network filters offer significant performance improvements for this task, offering improved definition, particularly in object edges, within the recovered images. Various feature normalisation techniques are investigated to examine the ways in which these affect the performance of the proposed models.

The fourth novel contribution of this thesis is improved normalisation techniques to pre-process Hyperspectral Image data for Super-Resolution training. The proposed techniques involve normalising the data in a band-wise manner and applying a small amount of outlier removal to more evenly distribute the data. This has several benefits, including allowing the model to focus more evenly on all components of the input data, while also simplifying the problem space.

The fifth novel contribution of this thesis is the analysis of artificial downsampling strategies for Hyperspectral Image Super-Resolution training. Ideally, Super-Resolution models are trained with paired low-resolution and high-resolution data. However, this is difficult to acquire, particularly for Hyperspectral Images, so downsampling techniques are often applied to artificially generate a low-resolution pair for a given Hyperspectral Image. Two novel datasets were acquired in collaboration with two other researchers that capture real low- and high-resolution paired images. Analysis is performed on how well models trained on data generated with popular downsampling techniques perform when applied to real data, indicating the efficacy of models trained with these techniques in the real world.

In Chapter 6, Self-Supervised techniques are proposed to train individual dairy cattle identification models on unlabelled data acquired from aerial barn cameras - the desired deployment domain for this application. Standard self-supervised techniques train the model on individual images at any given training step. The techniques proposed exploit the properties of the data in this application to allow the model to learn from multiple training images at a given training step, leading to greatly improved performance.

The self-supervised techniques proposed in Chapter 6 require specific properties to be present in the training data to exploit these properties within the algorithm. The final contribution of this thesis is the algorithms used to extract the data for the proposed self-supervised techniques, allowing for large quantities of training data to be gathered in an automated fashion.

1.3 Organisation of Thesis

The remainder of this thesis is organised as follows.

Chapter 2 outlines the key techniques and metrics used throughout this thesis. This chapter is intended to present enough information so that the reader can grasp the content of this thesis and its contributions without having to consult additional sources.

Chapter 3 provides an extensive literature review on the relevant research background and context for the techniques proposed in this thesis. It covers the topics of deep learning for image processing tasks such as classification and SR, the relevant background topics for this thesis including cow identification and hyperspectral imaging, and finally the two key techniques used in this thesis, which are similarity learning and Self-Organised Operational Neural Networks (Self-ONNs).

Chapter 4 explores the application of similarity learning to the specific task of supervised identification of individual dairy cattle. A novel evaluation of how well the developed models can identify cows that were not included in the training dataset is conducted. In addition, novel Self-Organised Operational Neural Network models are proposed, in order to enhance the parameter efficiency of cow identification models.

Chapter 5 delves into the topic of Hyperspectral Image Super-Resolution through the application of deep learning methods. Novel Self-Organised Operational Neural Network architectures are introduced, which are designed to improve Super-Resolution performance across a diverse range of Hyperspectral Image datasets. The chapter also presents novel normalisation strategies aimed at boosting super-resolution effectiveness over multiple images. Furthermore, a novel and comprehensive analysis of the synthetic downsampling methods typically used for Hyperspectral Image Super-Resolution is conducted, assessing their performance when applied to a novel real data set.

Chapter 6 examines strategies to close the domain gap between the dairy cattle identification models introduced in Chapter 4 and their real-world deployment in barn environments. Novel innovative self-supervised methods designed to address this challenge are proposed, along with techniques for the automatic generation of extensive datasets that these methods can leverage. In addition, the chapter examines the use of Self-Organised Operational Neural Network architectures to improve model performance.

Chapter 7 offers concluding remarks and recommends directions for future research.

2 Technical Background

In recent years, the fields of artificial intelligence (AI) and machine learning (ML) have transformed the landscape of computer vision and image processing, with deep learning emerging as the dominant paradigm. This chapter provides an overview of the technical foundations and advancements in deep learning for image processing, giving the reader the necessary background for the research contributions discussed in later chapters. This chapter also highlights the trends and breakthroughs that have cemented deep learning's position at the forefront of this domain.

AI encompasses a broad spectrum of techniques aimed at creating systems capable of performing tasks traditionally requiring human intelligence, such as decision-making, problem-solving, and perception. Within AI, machine learning represents a subset focused on enabling machines to learn from data and improve their performance without being explicitly programmed. ML methods are traditionally categorised into supervised, unsupervised, and reinforcement learning, each characterised by the type of data and feedback utilised during the training process.

Supervised learning relies on labelled datasets, where input-output pairs are used to train models to make predictions or classifications. Supervised learning is potentially the most effective form of training as the objective is well defined since the desired output is known for each individual data point from its label. However, the major drawback with this approach is that it is often very costly to obtain said labels, making it impractical in many scenarios. In contrast, unsupervised learning operates without explicit labels, uncovering patterns or structures within data, such as clustering or dimensionality reduction. This means that data is much more easily obtained since labels are not required, though the absence of labels makes the training objective much harder to define and consequently results in difficulty achieving the desired objective. Self-supervised learning aims to bridge the gap between supervised and unsupervised learning by by creating its own supervisory signals from the data itself. Examples of this include predicting masked words in a sentence or completing a missing image portion, where the objective is to force the model to learn underlying structures and features without the need for human-provided labels. Reinforcement learning explores how agents can learn optimal behaviors by interacting with an environment and receiving rewards or penalties based on their actions. This technique is potentially very powerful as agents can learn their own strategies to complete complex tasks without any human intervention. However, similar to unsupervised learning, the objective in reinforcement learning is often very challenging to define making it hard to achieve the desired results.

Among the branches of ML, deep learning has achieved unprecedented success, particularly in computer vision tasks. Leveraging neural networks with multiple layers, deep learning algorithms have demonstrated exceptional capabilities in extracting hierarchical features from raw data. This has enabled breakthroughs in areas such as object detection, image segmentation, and image classification, far surpassing traditional computer vision techniques. The advent of large-scale datasets, increased computational power, and advanced neural network architectures has further accelerated the adoption and dominance of deep learning in image processing.

Due to this success, almost all state-of-the art techniques for image processing and computer vision tasks leverage deep learning and many notable trends have emerged. One prominent direction is the focus on more efficient and lightweight models designed to perform effectively on resource-constrained devices. Simultaneously, transformer architectures, initially introduced for natural language processing, have gained traction and popularity over more traditional Convolutional Neural Networks, particularly for large-scale data tasks. Additionally, there is a growing emphasis on self-supervised and unsupervised learning approaches, which reduce the dependency on large labelled datasets by learning meaningful representations from unlabelled data. Generative models, including Generative Adversarial Networks (GANs) and diffusion models, have also gained traction, enabling high-quality image synthesis, enhancement, and restoration. Finally, due to the black box nature of neural networks, there is a huge amount of interest and research being conducted on explainability to better understand how neural networks come to make their decisions. These trends collectively reflect a shift toward more versatile, scalable, accessible, and transparent deep learning solutions for complex image processing challenges.

The remainder of this chapter delves into the technical aspects of deep learning, with a particular focus on its applications to dairy cattle identification and Hyperspectral Image Super-Resolution. It examines key architectures, training methodologies, and challenges, while also highlighting emerging trends that are shaping the future of this field. This chapter aims to provide a foundation for understanding the novel contributions explored in subsequent chapters of this thesis.

2.1 Deep Learning

Deep learning is a branch of machine learning that uses artificial neural networks (inspired by the human brain) with multiple layers to model complex patterns in data. These networks learn to perform tasks by considering examples, generally without task-specific programming. The model is composed of many layers of neurons that transform the input data as it passes through the network as shown in Figure 2.1. The first layer processes the input directly, and the following layers operate on the output of the previous layer, abstracting higher-level features of the data. Each neuron in the network's layers is associated with a set of weights and a bias. When an input is fed into the network, it is multiplied by these weights, which essentially determines the strength or importance of the input features in relation to the task the network is trying to learn.

Mathematically, the pre-activation output $\mathbf{z}$ of a layer l, is the matrix multiplication of the weights $\mathbf{W}$ at layer l and the output $\mathbf{a}$ of the previous layer $l-1$, plus the bias, $\mathbf{b}$ at layer l as defined in Eq. (2.1):

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \tag{2.1}$$

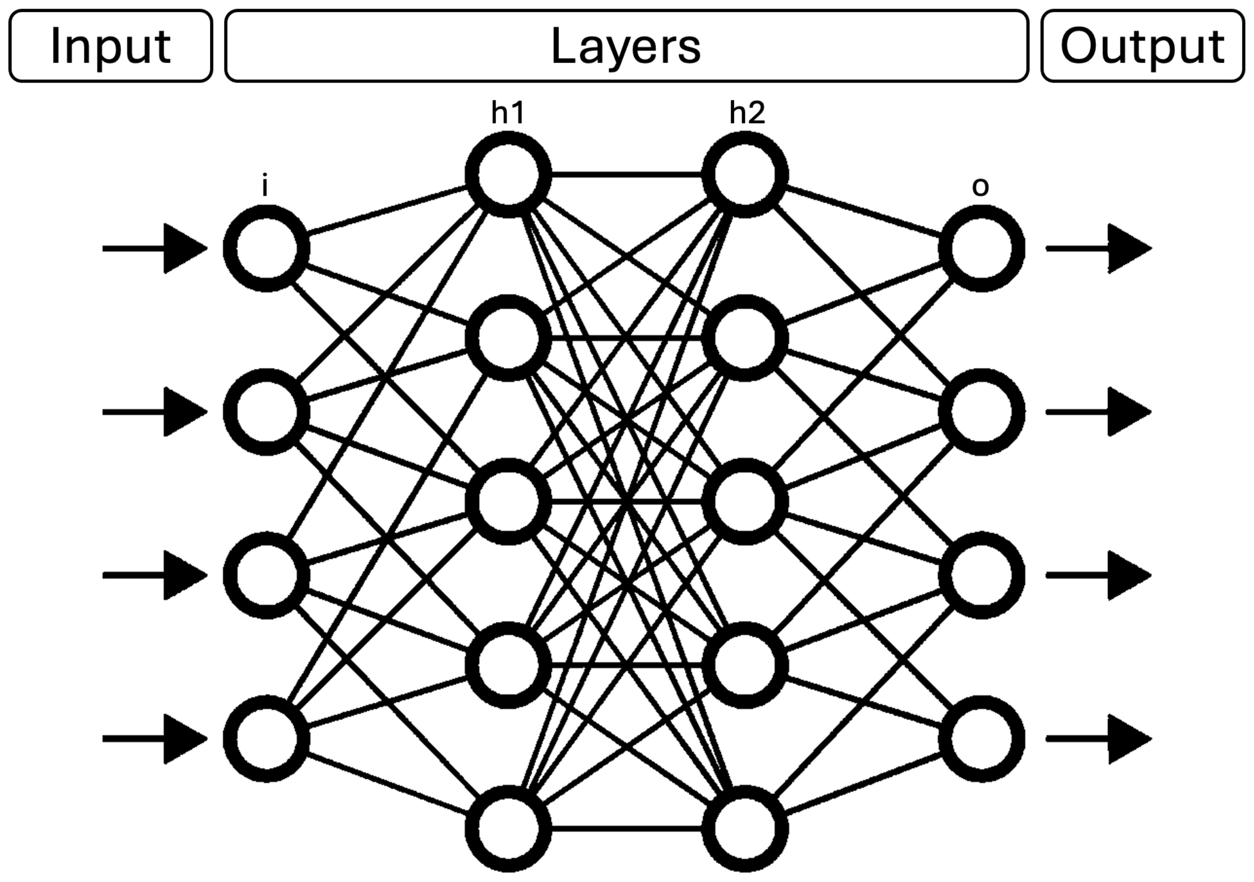

Figure 2.1: Neural Network Diagram

The pre-activation output, $\mathbf{z}$ is then passed to an activation function g to get the activation output $\mathbf{a}$ as definded in Eq. (2.2):

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]}) \tag{2.2}$$

This process is repeated for each layer iteratively until the final output layer is reached ($l = L$). For the first layer ($l = 0$), $\mathbf{a}^{[l-1]}$ is the input to the network. The final network output $\mathbf{y}$ is defined as:

$$\mathbf{y} = \mathbf{a}^{[L]} \tag{2.3}$$

During the training process, the network adjusts these weights based on the feedback it receives in terms of how well it performs (i.e. the loss). This adjustment is done through a process called backpropagation and is optimised using algorithms based on gradient descent.

The goal of training a neural network is to find the optimal set of weights that minimises the loss function, meaning the network's predictions are as close as possible to the actual desired values. In essence, the weights are learned in such a way that the network is able to extract meaningful features from the data that the network then uses to make predictions or decisions based on the input data it receives.

In supervised deep learning, the learning process relies on labelled datasets, where the correct output for each input is used to compute the loss — a measure of the difference between the predicted and true labelled values. The effectiveness of a deep learning model depends significantly on the size and quality of the training dataset, as larger datasets provide more diverse examples for the model to learn from. This diversity helps mitigate the risk of overfitting, where a model performs well on training data but fails to generalise to new unseen data. Achieving generalisation, where the model performs effectively on new data, is the most important goal in the development of deep learning models and is closely tied to the quality and comprehensiveness of the training dataset.

2.1.1 Training, Validation, and Testing

When deep learning models are developed using supervised learning, datasets are typically divided into three distinct portions: training, validation, and testing. Each of these partitions serves a specific purpose in the model development process. The training set is used to teach the model and is typically composed of a randomly selected subset of around 70% of the entire dataset. During this stage, the model 'sees' and learns from these data samples, adjusting its weights and biases to minimise the error in its predictions via backpropagation. The goal is to optimise performance on the training data and learn the underlying patterns and relationships in the dataset.

The validation set is a separate random subset of data, typically containing around 15% of the total data, used to monitor the model's performance after each full iteration of training, known as an epoch. Unlike the training set, the model does not learn from the validation set. Instead, this partition helps assess whether the model is generalising well to unseen data or overfitting to the training set. Overfitting occurs when the model becomes overly specialised to the training data, resulting in diminished performance on new data. A clear indicator of overfitting is when the model's performance continues to improve on the training set but stagnates or deteriorates on the validation set. In practice, the validation set is also used to identify the optimal set of model parameters, often employing techniques like early stopping. Early stopping ensures that training halts when validation performance ceases to improve, thereby preventing unnecessary overfitting.

Finally, the test set is reserved for evaluating the performance of the finalised model and comprises the remaining data not used in the training and validation subsets and is also typically around 15% of the overall data. Unlike the validation set, the testing set is not used at any point during the training process. This separation ensures that the evaluation provides an unbiased estimate of the model's ability to generalise to entirely new data. Since the validation set is used throughout training to guide decisions, it cannot offer a truly independent measure of the model's performance. Therefore, the testing set acts as a final benchmark, verifying the model's effectiveness in real-world scenarios. If the dataset is sufficiently large, the training, validation, and testing splits are typically fixed. However, when the dataset is relatively small, cross-validation is often employed: the model is trained multiple times with different splits, and the results are averaged to reduce bias and improve statistical reliability.

2.1.2 Loss Functions

Loss functions, also known as cost functions, measure the discrepancy between the model's predictions and the actual labels. They guide the training process by indicating how far off the predictions are from the target labels. Common loss functions in image-based tasks include Mean Squared Error (MSE) for image-to-image mapping tasks, Cross-Entropy Loss for classification tasks, and Intersection over Union (IoU) for segmentation tasks. The choice of loss function depends on the specific task and has a significant impact on the efficiency and effectiveness of the learning process.

2.1.3 Backpropagation & Gradient Descent

During the training process, a neural network learns by adjusting its weights based on feedback regarding its performance. This feedback comes from the loss function, which measures the difference between the predicted output of the network and the actual target values. To optimise the weights, the network uses an iterative process called backpropagation, which involves calculating the gradients of the loss function with respect to each weight. These gradients indicate how much each weight contributed to the error and are used to update the weights in a way that minimises overall loss.

The backpropagation algorithm works by applying the chain rule to propagate the error back through the network, starting from the output layer and moving toward the input layer. For each weight w, the gradient of the loss $\mathcal{L}$ is computed as:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial w} \tag{2.4}$$

Where $\frac{\partial \mathcal{L}}{\partial w}$ is the gradient of the loss with respect to the weight, $\frac{\partial \mathcal{L}}{\partial y}$ is the derivative of the loss with respect to the network's output, and $\frac{\partial y}{\partial w}$ is the derivative of the output with respect to the weight w.

Gradients are calculated for all weights in the network and, once they are determined, the weights are updated using an optimisation algorithm such as gradient descent. In standard gradient descent, the weight update rule is as follows:

$$w \leftarrow w - \eta \cdot \frac{\partial \mathcal{L}}{\partial w} \tag{2.5}$$

Where w is the weight to be updated, η is the learning rate, a hyperparameter that controls the step size of the update, and $\frac{\partial \mathcal{L}}{\partial w}$ is the gradient of the loss with respect to the weight. This process is repeated iteratively for each batch (a random subsample) of training data, gradually reducing the loss function by updating the weights in the direction of steepest descent (i.e., the negative gradient). Over time, this leads to a more accurate model, as the weights converge to values that minimise the error between the predicted and actual outputs.

In summary, backpropagation efficiently computes the gradients needed for training, and gradient descent is the algorithm that uses these gradients to iteratively adjust the weights of the network, ultimately optimising the model's performance.

2.1.4 Optimisation Algorithms

Gradient descent, defined in Eq. 2.5, is the foundational optimisation algorithm for neural networks in which the gradient is calculated using the entire training dataset at each iteration. However, this method is computationally expensive for large datasets and is slow to converge. Stochastic Gradient Descent (SGD) is more commonly used in practice, where the gradient is computed using either a single training example or a mini-batch of training examples. While this introduces more noise, it greatly speeds up convergence times, though is potentially more susceptible to converging at non-global minima.

Although basic gradient descent provides a foundation for weight updates, several more advanced optimisation algorithms have been developed to further improve convergence speed and stability. These algorithms often incorporate additional techniques such as momentum, adaptive learning rates, and the use of past gradients. Some examples include:

- Adagrad: Adapts the learning rate for each parameter based on historical gradients, making it useful for sparse data.
- RMSprop: Addresses the problem of rapidly decaying learning rates in Adagrad by using a moving average of squared gradients to normalise the updates.

- Adam (Adaptive Moment Estimation): Combines the benefits of momentum and adaptive learning rates by maintaining both the first and second moments of the gradients. Adam is one of the most widely used optimisers due to its efficiency and effectiveness across a wide range of problems.

Although Adam and other advanced algorithms typically perform better in practice, they build on the core idea of gradient descent and offer optimisations to improve the speed and accuracy of weight updates, especially for large and complex models.

2.1.5 Activation Functions

An activation function in the context of deep neural networks is a mathematical construct that introduces non-linearity into the output of a neuron. This function is applied to the output of each neural network layer, transforming the linear input into a non-linear output. The use of activation functions is critical in the learning process of a neural network, as it provides the ability for the network to represent complex non-linear relationships in the data through its layers.

There are a vast amount of activation functions present in the literature. Three popular activation functions used in this thesis include Sigmoid, Hyperbolic Tangent (Tanh), and Rectified Linear Unit (ReLU). Each activation function has specific advantages and limitations that influence the performance and suitability of the network for various tasks.

Sigmoid Function: The Sigmoid function, defined in Eq. (2.6) maps the input values into a range between 0 and 1. This characteristic makes it suitable for applications where the model needs to predict probabilities as outputs, and is thus commonly applied to the final layer of classification models. However, it is less commonly used in hidden layers in deep networks because of issues such as vanishing gradients, impeding the network's learning process.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

Hyperbolic Tangent (Tanh) Function: The Tanh function, defined in Eq. (2.7), is similar to the sigmoid function, but the output values range from -1 to 1. This makes it more effective than Sigmoid in certain contexts, as it centres the data, improving the learning efficiency for subsequent layers. However, Tanh also suffers from the vanishing gradient problem.

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

Rectified Linear Unit (ReLU) Function: ReLU, defined in Eq. (2.8) has become the most widely used activation function in deep learning models, particularly in convolutional neural networks. The application directly outputs the input if it is positive; otherwise, it outputs zero. ReLU is favoured for its computational simplicity and its ability to mitigate the vanishing gradient problem, thus enabling models to learn faster and perform better.

$$\text{ReLU}(x) = \max(0, x) \tag{2.8}$$

A plot of each activation function can be seen in Figure 2.2.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks, which, as the name suggests, utilises the convolution operation with learnable filter parameters, which is defined in Eq. (2.9):

$$(f * g)(i, j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} f(i + m, j + n) \cdot g(m, n) \tag{2.9}$$

Where f is the input image or the input feature map. g is the kernel or filter that is

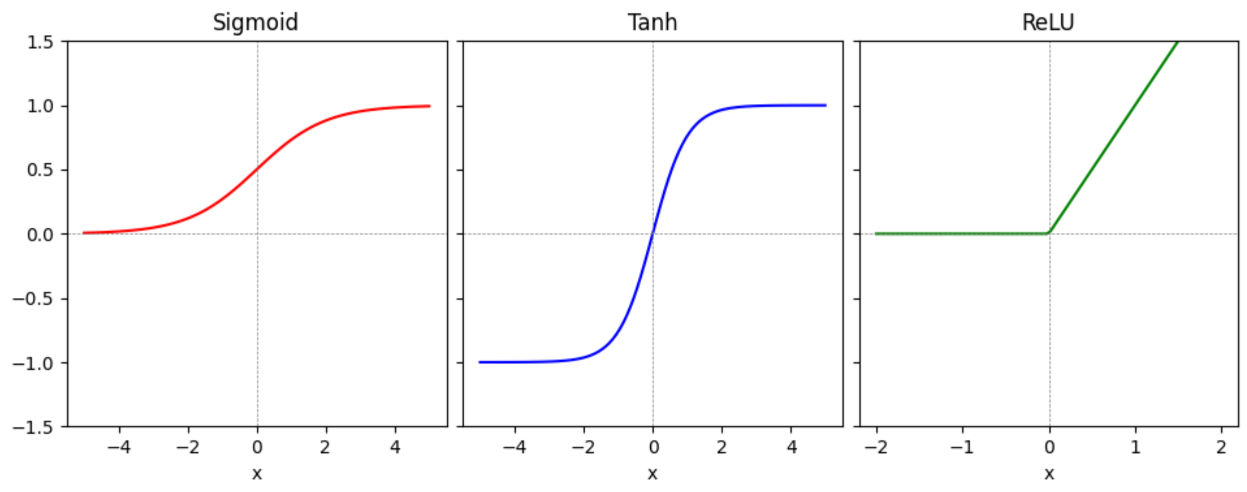

Figure 2.2: Activation Function Plots

applied to the image, which contains the learnable parameters of the network. i and j are the coordinates in the output feature map. m and n are the coordinates in the learnable kernel g.

The use of the convolution operation makes CNNs well suited to image processing tasks, as they are able to learn complex spatial hierarchies of features from two-dimensional inputs. Like regular linear neural networks, CNNs are comprised of many layers, where each layer operates on the outputs of the previous layer, abstracting higher and higher level features as shown in Figure 2.3. The early layers typically detect low-level features such as edges, the middle layers abstract these into higher-level patterns like shapes, and the later layers combine these abstractions to recognise complex, task-specific features. Depending on the task the model is to be used for, a few linear layers are often added to the end of the network to produce numerical outputs, though for certain tasks, the direct output of the final convolutional layers is sometimes used.

CNNs are invariant to the scale and translation of the input, meaning they can recognise objects regardless of their position or size in the image, a property that standard linear layers do not possess. However, a known limitation of CNNs is that they are not rotation-invariant. This limitation can be mitigated for in applications where the object orientation within the image is unknown, by training the model with a large number of rotated views

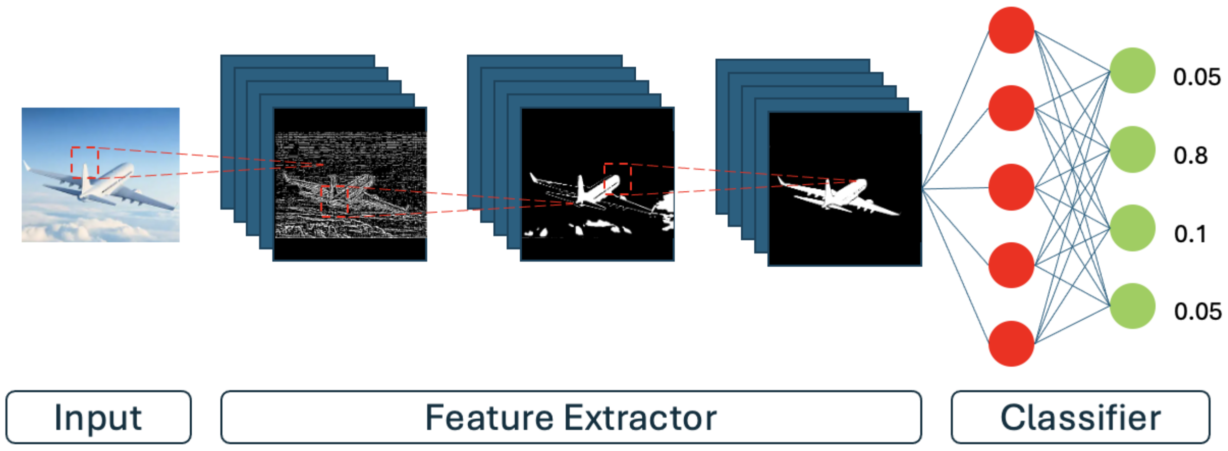

Figure 2.3: Convolutional Neural Network

during training, but this is still an inherent weakness in certain applications. Like standard linear Neural Networks, CNNs rely on activation functions after each layer to provide non-linear operations to learn complex features. However, due to the memory constraints that processing images impose, downsampling techniques are often employed to reduce the size of feature maps between layers. This downsampling can take the form of a pooling layer, shown in Figure 2.4, where the feature map is partitioned into a set of non-overlapping rectangles and, for each such sub-region, outputs a value most commonly determined by the average or the maximum of the sub-region, which is known as average and max pooling respectively. Alternatively, a stride parameter S can be introduced to the convolution operation itself, where the stride determines the number of pixels by which the filter shifts over the input data after each operation, resulting in an output feature map reduced by a factor S in each direction.

2.3 Self-Organised Operational Neural Networks

Self-Organised Operational Neural Networks (Self-ONNs) are a superset of CNNs which extend the linear multiplication operation of each convolutional kernel node to a non-linear function as represented in Figure 2.5.

The non-linear functions at each node are known as a nodal operator and are approxi-

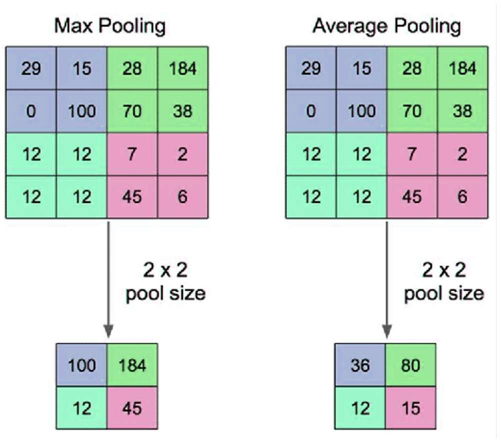

Figure 2.4: Max and Average Pooling Examples

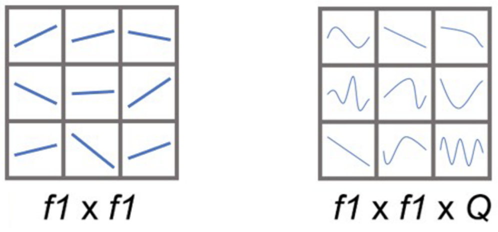

Figure 2.5: CNN Filter (left) vs Self-ONN Filter (right).

mated using MacLaurin series expansions defined in Eq. (2.10):

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n \tag{2.10}$$

In practice, the ∞ upper bound of the summation term in Eq. (2.10) is replaced by a hyperparameter Q which determines the number of terms present in the MacLaurin series approximation. In the Self-ONN formulation, the standard function symbol f in Eq. (2.10) is replaced by the nodal operator symbol $\mathbf{\Psi}$. To make $\mathbf{\Psi}$ learnable, the $\frac{f^{(n)}(0)}{n!}$ term in Eq. (2.10) is replaced by a learnable weight vector $\boldsymbol{w}$ to give:

$$\mathbf{\Psi}(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_Q x^Q \tag{2.11}$$

In practice, w_0 is removed and replaced by the bias for the entire layer. Note that a Q of 1 would make a self-operational layer identical to a convolutional layer. For practical use of Self-ONNs, it is desirable to restrict the data passed to each self-operational layer to the range $[-1, 1]$ otherwise the data can become exponentially large when passed through the layer due to the high-order terms present in the nodal operator, causing training instability. Furthermore, constraining data to this range keeps the data close to 0, which is where the MacLaurin series function approximations are theoretically most accurate.

To train a Self-ONN, the standard back-propagation algorithm is used, as is the case with all other deep learning models.

2.3.1 Parametric Analysis

The additional non-linear capacity of Self-ONNs compared to a CNN comes at the cost of additional network parameters. The introduction of the parameter Q to make the polynomial in Eq. (2.10) finite gives:

$$f(x) = \sum_{n=1}^{Q} \frac{f^{(n)}(0)}{n!} x^n \tag{2.12}$$

Higher Q values yield more accurate function approximations, but also increase total network parameters as the Q value directly equates to the multiplication of parameters compared to a standard convolutional filter. The number of parameters in the convolutional layers of a CNN can be calculated using the following equation:

$$\#\ \text{parameters} = \sum_{l=0}^{L-1} (n_l \times m_l \times f_l + 1) \times f_{l+1} \tag{2.13}$$

where L is the number of layers, n_l, m_l is the number of rows and columns in the convolutional filters at layer l, f is the number of filters and the constant 1 accounts for the bias for each filter. Note, that on the first layer, i.e. $l = 0$, the number of filters from the previous layer $(l-1)$ is given by the number of channels of the input image. To compute the number of parameters of a Self-ONN, the number of filters in the previous layer, f, in Eq. (2.13) is simply multiplied by Q to give:

$$\# \text{ parameters} = \sum_{l=0}^{L-1} (n_l \times m_l \times f_l \times Q + 1) \times f_{l+1} \tag{2.14}$$

2.4 Normalisation

Normalisation is the process of adjusting and scaling data to fit within a certain range to improve the performance and efficiency of machine learning algorithms. This involves scaling all the numerical features in a dataset to a common scale without distorting differences in the range of values. Normalising data ensures that numerical features with different scales do not affect the given algorithm's performance. For example, many algorithms, such as support vector machines (SVMs) and K-Nearest Neighbours (KNN), compute distances between data points, so having features on the same scale allows the algorithm to learn more effectively. In addition, algorithms that use gradient descent as an optimisation technique converge faster with normalised features.

The two most common types of normalisation are:

Min-Max Normalisation (Rescaling): This method scales and transforms the features to a range between a new minimum and maximum value, typically between $[0, 1]$, or $[-1, 1]$. The transformation is given in Eq. (2.15):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.15}$$

where X is the original data, X_{min} is the minimum value of the data, and X_{max} is the maximum value of the data.

Standard Normal Variate (Standardisation or Z-score Normalisation): This method adjusts the data to have a 0 mean and unit variance, defined in Eq. (2.16):

$$X_{norm} = \frac{(X - \mu)}{\sigma} \tag{2.16}$$

Where μ is the mean of the data X and σ is the standard deviation of the data X. This approach centres the data around a mean of 0 and has a standard deviation of 1. However, this technique does not bind values to a specific range, which can be a problematic for algorithms expecting input data within bounded intervals.

In addition to normalising the dataset, it is also common to normalise data passed between layers in DNNs. Some of the commonly used techniques are detailed below.

- **L1 Normalisation** involves making the sum of absolute values of the data equal to 1.
- **L2 Normalisation** scales the components of the data so that the sum of the squares of the components is equal to 1. In the context of machine learning, L1 and L2 normalisation are more often associated with regularisation to prevent overfitting by penalising large coefficients, although they can also be used for standard normalisation operations.
- **Batch Normalisation** normalises data across each neuron or channel individually using the standard normal variate with the mean and variance of the neuron/channel across the batch. A running mean and variance are updated with each batch which is used at inference time. Batch normalisation addresses the issue of internal covariate shift, where the distribution of each layer's inputs changes as the parameters of the previous layers change during training, making the training process faster and more stable, and has seen use in a broad range of deep learning applications.
- **Instance Normalisation** normalises data using the standard normal variate on each individual channel separately with the mean and standard deviation computed across the channel. It is primarily used in style transfer applications. Since instance norm

computes the statistics across the channels individually, it can only be applied to CNNs as linear layers would not provide enough data points to compute statistics with a single neuron.

- **Layer Normalisation** normalises data across the feature dimension using the standard normal variate. It is particularly useful in applications where training must be performed with a single sample at a time.
- **Group Normalisation** divides the channels or neurons into groups and normalises the data with the standard normal variate with mean and variance calculated from each group. Group normalisation's performance is less dependent on the batch size, making it useful in situations where it is desirable to use small batches due to memory constraints. It is a compromise between instance normalisation and layer normalisation, providing benefits in a wide range of network architectures.

Batch, instance, layer, and group normalisation all have the optional addition of learnable parameters to scale and add a bias term to the normalised data.

2.5 Similarity Learning

Similarity learning, also known as metric learning, refers to a type of learning where the goal is to learn a representation of the data such that similar items are represented by close points in the embedding space, and dissimilar items are represented by distant points. This approach is particularly useful in tasks that involve finding relationships between data points, such as face recognition, item recommendation, and clustering. This is achieved by training models on examples of similar (positive pairs) and dissimilar (negative pairs) items, and optimising a loss function that encourages the model to represent these relationships correctly in the learned embedding space.

The most common approaches and techniques in similarity learning include:

Contrastive Loss: This method involves pairs of examples. The loss function is designed to minimise the distance between embeddings of similar pairs while ensuring that the distance between embeddings of dissimilar pairs is greater than a margin. This encourages the model to cluster similar items closer together, while pushing dissimilar items apart. Contrastive Loss is defined in Eq. (2.17):

$$L = \frac{1}{2}\left((1-Y)\cdot D^2 + Y\cdot \max(0, m-D)^2\right) \tag{2.17}$$

Where Y is 0 if the embeddings pair are of the same class and 1 if not, D is the euclidean distance between the embeddings pair, and m is a margin parameter specifying the minimum separation distance between dissimilar pairs.

Triplet Loss: This method extends the idea of contrastive loss by considering triplets of examples: an anchor, a positive example (similar to the anchor), and a negative example (dissimilar to the anchor). The goal is to learn embeddings such that the anchor is closer to the positive example than to the negative example by some represented by α as demonstrated in Figure 2.6.

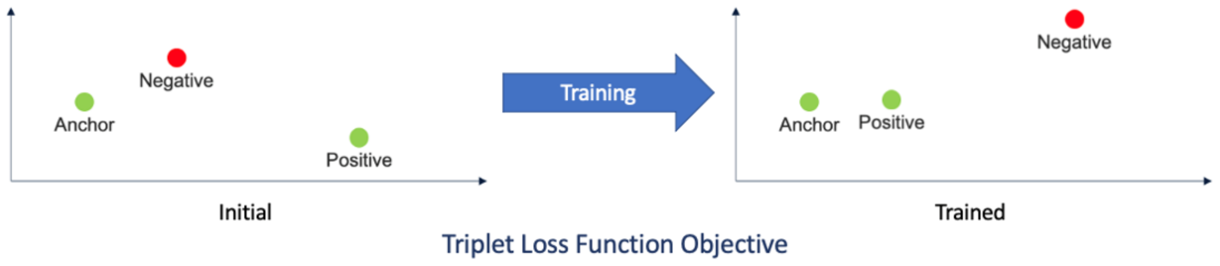

Figure 2.6: Triplet Loss Function Objective

This objective is mathematically expressed in Eq. (2.18):

$$L = \sum_{i}^{N} \left[\, ||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha \right] \tag{2.18}$$

Where $f(x_i^a)$ is the anchor embedding, $f(x_i^p)$ and $f(x_i^n)$ is the negative embedding

The effectiveness of similarity learning in deep learning applications relies on the ability to learn rich, meaningful embeddings that capture the underlying semantics or relationships in the data which often produce better feature representations than other training methods.

2.6 K-Nearest Neighbours

A K-Nearest Neighbours (KNN) classifier is a simple, yet effective machine learning algorithm that belongs to the family of instance-based, or lazy, learning algorithms. It operates on the basic principle of feature similarity, which means that it classifies new cases based on how closely they resemble existing cases in the training dataset.

The KNN classifier works by identifying the k instances in the training dataset that are nearest to the new instance, based on a distance metric (such as Euclidean, Manhattan, or Hamming distance). The classification of the new instance is then determined by a majority vote among its k nearest neighbours, with the new instance assigned to the most common class among its nearest neighbours. An example of a KNN classifier can be seen in Figure 2.7.

2.7 Evaluation Metrics

The evaluation metrics that are used throughout this thesis are defined in this section.

2.7.1 Accuracy

Accuracy is a commonly used evaluation metric for classification tasks. It is defined as the proportion of correct predictions made by the model out of all predictions made and is defined in Eq. (2.19):

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2.19}$$

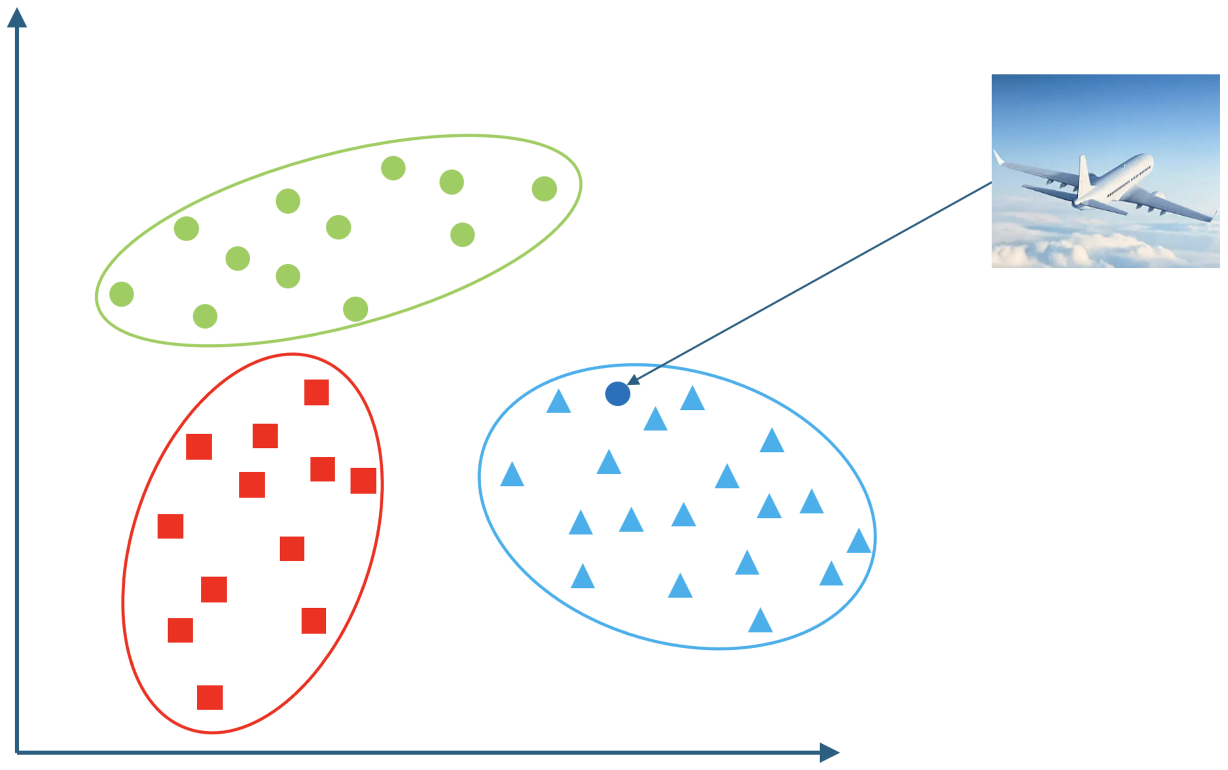

Figure 2.7: KNN Classifier Example.
The blue circle represents the embedding representation of the input image to be classified. All remaining shapes represent the KNN's reference embeddings of known examples. Since the input image's embedding is closest to the blue triangles, then the image would be classified as this class.

Accuracy is expressed as a percentage, where 0% represents the worst possible model performance with no correct classifications, and 100% represents perfect model performance with no incorrect classifications.

2.7.2 F1 Score

The F1 score is another commonly used evaluation metric for classification tasks. The F1 score considers the type of error the model makes, either a false positive or a false negative. A false positive occurs when the model incorrectly predicts the positive class and a false negative occurs when the model incorrectly predicts the negative class. True positives and negatives are where the model correctly predicts the positive or negative class, respectively. The F1 score is two times the product of the precision and recall divided by the sum of the precision and recall as defined in Eq. (2.20):

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.20}$$

Where Precision and Recall are defined in Eq. (2.21) and Eq. (2.22) respectively:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.21}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.22}$$

The F1 Score is especially useful in situations where there are imbalanced classes or when the cost of false positives and false negatives is high. The F1 score ranges between a value of 0 and 1 where the highest possible value is desired.

2.7.3 Silhouette Score

The silhouette score is a metric used to measure how well a clustering algorithm clusters the data. Its value gives a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette score is defined in Eq. (2.23):

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.23}$$

Where $s(i)$ is the silhouette score for a single data point, $a(i)$ is the average distance of i to the other points in the same cluster, and $b(i)$ is the smallest average distance of i to all points in any other cluster, of which i is not a member.

The silhouette score ranges from -1 to 1. A high silhouette score indicates that the data point i is well matched to its own cluster and poorly matched to neighboring clusters. The silhouette score is typically averaged across all data points in the test set and this is how it will be used in this thesis.

2.7.4 Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) is a widely adopted metric in the domain of image processing for quantifying the quality of reconstructed images in relation to their original counterparts. Primarily, the PSNR gauges the severity of distortion present within the reconstructed image. The PSNR for an original image I and its distorted counterpart K is defined in Eq. (2.24):

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right) \tag{2.24}$$

Where MAX_I is the maximum possible pixel intensity of the image. The Mean Squared Error (MSE) between the original and distorted images is defined in Eq. (2.25):

$$\mathrm{MSE} = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j) - K(i,j)]^2 \tag{2.25}$$

Where, $I(i,j)$ and $K(i,j)$ denote the pixel intensities at the position i, j in the original and distorted images, respectively. In addition, (M) and (N) represent the dimensions of the images.

A higher PSNR score suggests reduced distortion and a more accurate reconstruction of the original image. Given its interpretability and relevance, PSNR remains a pivotal measure in image quality assessment studies across several domains.

2.7.5 Structural Similarity Index Measure

The Structural Similarity Index Measure (SSIM) is an advanced metric designed to assess image quality similar to human quality perception. Unlike traditional error summation methods such as MSE, SSIM is designed to evaluate changes in structural information, luminance, and texture contrast. These three components aim to emulate the human visual system's perception, recognizing that humans are particularly sensitive to structural changes in an image. Given two images, x and y, with their local means μ_x and μ_y, local variances

σ_x^2 and σ_y^2, and cross-covariance σ_{xy}, the SSIM index is defined in Eq. (2.26):

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.26}$$

Here, C_1 and C_2 are small constants introduced to avoid instability when the denominators are close to zero. They can be expressed as:

$$C_1 = (k_1 L)^2 \quad \text{and} \quad C_2 = (k_2 L)^2 \tag{2.27}$$

Where L signifies the dynamic range of pixel intensities (255 for an 8-bit image) and $k_1 = 0.01$ and $k_2 = 0.03$ are standard constants.

In practice, SSIM is computed within local windows, instead of the entire image, to account for changes in local patterns and structures. The overall SSIM score is computed by averaging all individual local window SSIM scores.

2.7.6 Spectral Angle Mapper

The Spectral Angle Mapper (SAM) is a geometric-based approach commonly employed in hyperspectral remote sensing applications, primarily for identifying and classifying spectra in hyperspectral images. It works by treating the spectra as vectors in a space with dimensionality equivalent to the number of spectral bands and computes the angle between these vectors. This angle serves as a measure of spectral similarity and is particularly advantageous due to its insensitivity to vector magnitude, making it suitable for comparing material reflectance spectra observed under varying illumination conditions. While SAM is traditionally used for spectral identification and classification, it is also useful as an evaluation metric to quantify the similarity between reconstructed and ground truth spectra in the context of Hyperspectral Image Super-Resolution. Given two spectra, $\mathbf{r}$ and $\mathbf{s}$ represented as vectors, the spectral angle, θ, is defined in Eq. (2.28):

$$\theta = \arccos\left(\frac{\mathbf{r}\cdot\mathbf{s}}{\|\mathbf{r}\|\|\mathbf{s}\|}\right) \tag{2.28}$$

Where $\mathbf{r}\cdot\mathbf{s}$ represents the dot product of the two vectors, $\|\mathbf{r}\|$ and $\|\mathbf{s}\|$ represent the magnitudes of vectors $\mathbf{r}$ and $\mathbf{s}$, respectively.

A smaller spectral angle indicates higher similarity between the two spectra. A zero angle indicates identical spectra, while an angle of 90 degrees suggests orthogonality or maximum dissimilarity.

2.7.7 Error Relative Global Adimensional de Synthèse

The Error Relative Global Adimensional de Synthèse (ERGAS) is a widely utilised metric in the remote sensing domain, primarily for the assessment of the quality of fused satellite imagery but is also useful for assessing single-image super-resolution performance. This metric gauges the relative error between original and reconstructed images and offers a dimensionless score, facilitating interpretability.

The ERGAS metric is defined in Eq. (2.29):

$$\text{ERGAS} = 100 \times \frac{r}{\bar{X}} \times \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{RMSE_i}{\bar{X}_i}\right)^2} \tag{2.29}$$

Where, r is the spatial resolution ratio between the coarser and finer resolution images. n is the number of spectral bands. $RMSE_i$ is the root mean square error of the i^{th} band. $\bar{X}_i$ represents the mean intensity of the i^{th} band in the reference image. $\bar{X}$ is the overall mean intensity across all bands in the reference image.

A lower ERGAS value indicates superior performance of the reconstruction technique, with zero representing perfect fusion.

2.8 Self-Supervised Learning

Self-supervised learning is a machine learning paradigm that leverages unlabeled data to learn useful feature representations without requiring explicit manual annotations. Unlike supervised learning, which depends on large volumes of labeled examples, self-supervised learning defines an auxiliary or pretext task for which supervisory signals can be generated automatically from the inherent structure of the data. By solving this pretext task, the model acquires representations that transfer effectively to downstream tasks such as classification, detection, or segmentation.

Typical pretext tasks vary depending on the data modality. In computer vision, examples include predicting the relative position of image patches, solving jigsaw-like arrangements, colourising greyscale images, or learning invariances via contrastive methods. In natural language processing, the common task is predicting masked tokens or the next word/sentence.

Formally, self-supervised learning optimises an objective function associated with the pretext task during training. Once trained, the learned encoder or feature extractor can be fine-tuned or directly applied to target tasks, often achieving competitive or superior performance compared to fully supervised baselines, particularly when labelled data are scarce. This paradigm is thus particularly valuable in domains where manual annotation is expensive, subjective, or infeasible.

2.9 Hyperspectral Imaging

Hyperspectral imaging is a sensing technique that captures images across a large number of narrow, contiguous spectral bands, extending beyond the three broad channels of visible Red, Green and Blue (RGB). By recording both the spatial and spectral information of a scene, hyperspectral imaging enables the identification and quantification of materials based on their unique spectral signatures. The acquisition of hyperspectral data is a hardware-intensive process that combines optical dispersion, scanning mechanisms, and sensitive detectors to produce three-dimensional data cubes, with two axes representing spatial information and the third representing spectral content.

2.9.1 Spectral Separation Mechanisms

At the core of hyperspectral imaging hardware is the ability to separate incoming light into multiple wavelength channels. This is typically achieved using one of three approaches.

- **Dispersive elements:** Gratings and prisms are commonly used to diffract or refract incoming light into constituent wavelengths. Gratings provide high spectral resolution and can be tuned for specific wavelength ranges, while prisms exploit material dispersion properties to achieve separation.

- **Interference filters:** Tunable filters, such as liquid crystal tunable filters (LCTFs) and acousto-optic tunable filters (AOTFs), selectively transmit narrow wavelength bands while blocking others. By electronically tuning the filter, sequential images can be captured at different spectral bands without moving parts.

- **Filter mosaics:** More recent filter-on-chip technologies integrate narrowband filters directly on the sensor, with each pixel sensitive to a different wavelength. This allows single-shot hyperspectral capture, though often at reduced spectral resolution compared to scanning-based systems.

2.9.2 Image Formation Strategies

Different imaging architectures have been developed depending on how spatial and spectral information are acquired simultaneously.

- **Spatial scanning (pushbroom):** Being the most widely used in airborne and satellite hyperspectral imaging, pushbroom sensors use a slit to image one spatial line at a time. A dispersive element separates light into wavelengths, and as the platform moves, successive lines are recorded to build the data cube. This design provides high spectral resolution but requires stable motion control.

- **Spectral scanning (tunable filters):** In this configuration, a full 2D spatial image is captured at one wavelength at a time. Sequential tuning produces the full hyperspectral cube. This is well suited to stationary scenes but may suffer from motion artefacts.
- **Snapshot systems:** Using filter mosaics, image slicers, or coded apertures, snapshot systems capture the full cube in a single exposure. These systems are advantageous for dynamic scenes, such as biomedical imaging, but usually involve trade-offs in spectral or spatial resolution.

2.9.3 Detectors and Sensor Technology

The detectors used in hyperspectral imaging hardware depend on the spectral range of interest. Silicon-based CCD and CMOS sensors are commonly used for visible to near-infrared (VNIR, 400–1000 nm). For short-wave infrared (SWIR, 1000–2500 nm), indium gallium arsenide (InGaAs) detectors are employed due to their higher quantum efficiency. Mid-wave and long-wave infrared imaging (MWIR/LWIR) often requires cooled detectors such as mercury cadmium telluride (MCT), though uncooled microbolometers are also used for cost-sensitive applications.

2.9.4 Calibration and Image Capture

Once light is dispersed and detected, raw sensor output must undergo calibration to form usable hyperspectral images. Calibration includes radiometric correction, to account for sensor sensitivity variations; spectral calibration, to ensure accurate wavelength registration; and geometric calibration, to correct optical distortions. In practice, imaging sessions include reference measurements such as white panels (for reflectance normalisation) and dark frames (to remove sensor noise).

2.9.5 Data Cube Representation

The final result of hyperspectral image acquisition is the hyperspectral data cube, where two axes correspond to spatial dimensions (x, y) and the third axis corresponds to the spectral dimension (λ). Each pixel therefore contains a full spectral profile across potentially hundreds of contiguous bands, enabling fine-grained material classification, anomaly detection, and quantitative analysis beyond the capabilities of conventional RGB imaging.

3 Literature review

3.1 Deep Learning in Image Processing

The Convolutional Neural Network (CNN) was first popularised when the AlexNet model [18] exceeded the previous state of the art on the ImageNet LSVRC-2010 challenge [19] by over 10% Top-1 error. The concept of the CNN had been around for some time prior to this paper but performance was always limited due to the computational cost of such models. The authors of the AlexNet paper were the first to accelerate deep learning training by utilising Graphics Processing Units (GPUs), which allowed them to train a large model containing 60 million parameters and exploit the full power of the CNN which was previously computationally infeasible. To further exploit the success of AlexNet, the authors of [20] proposed a much deeper model with 16 and 19 layer variants named VGG. This model uses smaller 3x3 convolutional kernels, though despite the smaller filter sizes, the model is able to learn much higher-level features since it is significantly deeper than AlexNet and each layer builds upon the features from the previous layer, resulting in a significant performance improvement.

As neural networks grow deeper in size, they become affected by a phenomenon known as the vanishing gradient problem. This is where the gradients in the early layers become very small as the gradient is propagated backward from the output layer to the input layer. When the gradients become very small, or ”vanish”, the weights of the early layers in the network do not get updated effectively, meaning that those layers learn very slowly, or sometimes not at all. The authors of [21] propose an even deeper model named ResNet, but alleviate the vanishing gradient problem by introducing residual connections. This is where the input of a layer (or a block of layers) is added to its output, effectively allowing the creation of a skip connection. This means that each layer or block learns the difference between the input and the desired output rather than the direct input-to-output mapping, making optimisation smoother and improving gradient flow. The authors of [22] proposed ResNeXt which incorporates the idea of ensembles (aggregating outputs from multiple models) into the ResNet architecture which creates several smaller ResNet-like branches within the model to improve performance. Drawing inspiration from the success of the attention module within transformer models [23], the authors of [24] proposed squeeze and excitation networks which incorporate CNN specific attention modules into popular architectures such as ResNet and ResNeXt to improve performance.

Vision transformers [25] have recently emerged as a competitor to the more traditional CNN, outperforming the latter on certain vision tasks. Part of the reason for the success of CNN models is that they posses many inductive biases that make them well suited to many image tasks such as local connectivity, translation equivariance, parameter sharing and hierarchical feature learning [26]. However, these inductive biases many not be favourable in certain contexts. One of the key components of a transformer is the attention mechanism which enables the model to focus on relevant parts of the input data when making predictions. It assigns weights to different input elements (tokens) based on their relevance to a specific task, enabling the model to capture relationships between input tokens, regardless of their position in the sequence. Transformers therefore do not possess any of the inductive biases that CNNs possess as all data relationships have to be learned, allowing them to learn longer-range dependencies more easily within images and learn these dependencies more flexibly via self-attention. Consequently, this absence of inductive bias means that transformers require far greater quantities of training data to converge, typically in the order of millions of images. Transformers also posses other challenges with computation due to the quadratic nature of the attention mechanism, meaning that compromises have to be made in terms of the input dimensions. However, it has been shown that transformers generally outperform CNNs in situations where there are vast quantities of training data and computational power [27].

The authors of [28] propose ConvNeXt, a CNN-based model that integrates several design principles inspired by the success of transformers. These include adopting wider layers and receptive fields, fewer activations and normalisations, inverting dimensions, replacing batch normalisation with layer normalisation, and replacing the ReLU activation function with GeLU [29]. By incorporating these elements, ConvNeXt demonstrates that CNNs can achieve performance comparable to transformers while retaining the inherent advantages of CNNs, such as translation invariance and computational efficiency.

The AlexNet paper [18] uses the Stochastic Gradient Descent (SGD) optimiser to optimise their network. One problem with gradient descent optimisation is how best to set the learning rate. A learning rate too small can take a large amount of time to converge and is prone to getting stuck in local minima. A learning rate too large may miss the true minima or take too large a step in the wrong direction and consequently fail to converge. Adaptive Moment Estimation (Adam) [30] was proposed to adaptively adjust the effective learning rate for each network parameter based on adaptive estimates of lower-order moments. This optimiser has proven to be highly effective and has become the optimiser of choice among most deep learning researchers and practitioners, along with its variants. Regularisation terms such as weight decay are often added to optimisers to help mitigate overfitting. However, weight decay does not work in adaptive optimisers such as Adam due to its inequivalence to L2 regularisation. The authors of [31] therefore decouple the weight decay term from the optimisation step w.r.t. the loss for improved generalisation performance in their AdamW optimiser. More recent optimisers such as Sharpness-aware Minimisation (SAM) [32] have been proposed to improve generalisation performance by considering the sharpness of the loss. However, these approaches are computationally expensive since they require two gradient computations per optimisation step, which also introduces additional hyperparameters, making it more difficult to tune [33].

3.2 Similarity Learning and Triplet Loss

Image classification is one of the most common applications for CNNs and the standard approach is to train a neural network with a predefined number of classes represented by the output layer dimensions. The network is trained by passing images with known class values and the network learns to maximise the probability of the output dimension representing the class, while minimising all other indexes. The network architecture consists of several feature extraction layers, followed by a classification head consisting of two or more linear layers. The drawback with this approach is that the classification head parameters are learned in an indirect and inefficient way, typically requiring very large layer sizes [34]. Furthermore, the resulting network representations will not necessarily generalise well to new classes, meaning that if new classes are to be introduced, the entire network will require to be retrained and not just the final classification layer.

Similarity learning, also referred to as metric learning, addresses these limitations by shifting the learning objective from predicting class probabilities to mapping images into a highly discriminative embedding space. An embedding is a dense vector representation that encapsulates the most relevant features of the data for a specific task. The training process focuses on clustering embeddings of the same class closely in Euclidean space while ensuring embeddings of different classes are well separated. This approach enables the model to learn more robust features and achieve a more direct and effective training process [34].

Contrastive loss [35] was one of the techniques proposed to formulate the training objective in this way. The loss function aims to minimise the distance between same-class embeddings and maximise the distance between embeddings of different classes. [34] improved this idea by considering triplets of embeddings through their triplet loss function, expressed in Eq. (2.18). As the name suggests, this loss function takes three embeddings as input: an anchor, a positive, and a negative, where the anchor and positive are two different embeddings of the same class, and the negative is an embedding of a different class. The loss function is satisfied when the positive is closer to the anchor than the negative in euclidean space by a margin α. Once the model is trained, the embeddings are typically classified by a KNN classifier [36], although other forms of classifier, such as a single layer neural network, can also be used. Since the embeddings produced by the model are just representations of the input image and not fixed class probability values - as is the case with a standard classification network - the model can embed new classes without the need for retraining. Only the second stage KNN classifier would need to be altered, which requires negligible computation time compared to retraining an entire classification neural network.

However, triplets can become less informative as training progresses, leading to slow convergence. The authors of [37] proposed N-Pair Loss which computes triplet combinations between every negative pair for each anchor-positive pair instead of using a single negative pair for each anchor-positive pair. This improves convergence, however, it is hugely inefficient to compute all the negative pairs in such a fashion. The authors restrict each batch to contain only a single anchor-positive pair for each class to ease the computational burden, though it remains significant. To improve triplet selection while not significantly increasing the computational burden, the authors of [38] propose to select triplets based on both their self-similarity but also their relative similarity to multiple points. However, this introduces additional hyperparameters to tune the selection weights. [39] introduces a lifted structure loss which introduces a pairwise distance metric and selects samples based on this distance from both the anchor and the positive pairs to improve performance. This introduces additional overhead due to the computation of all pairs.

Pair-based methods such as triplet loss and its variants can be slow to converge and difficult to acquire informative pairs for training. [40] proposes ProxyNCA loss which artificially creates learnable artificial embeddings representing classes and the model learns to adjust the anchor image in embedding space based on these proxies to improve convergence times. ProxyNCA++ loss is proposed in [41] which improves the ProxyNCA loss by considering a proxy assignment probability score within the loss function. Proxy-based losses address the issue of having to find informative pairs in pair-based methods. However, they do not exploit fine-grained semantic relationships between samples in the same way that pair-based methods do. The authors of [42] therefore proposes Proxy Anchor loss which combines pair-based and proxy-based losses by forming an anchor proxy and computing pair-wise distances from all other embeddings in the batch relative to the proxy. The use of proxies improves convergence time and robustness to noise while also exploiting fine-grained data relations due to the pair-based loss of the proxy relative to all other batch embeddings. Another issue with proxy-based methods is that they do not consider the implicit hierarchy of categories in real-world datasets. The authors of [43] therefore propose hierarchical proxy-based loss which forms coarse proxies on top of the fine proxies to form the hierarchy which is determined using k-means clustering and is continually updated until convergence.

3.3 Cow Identification

Cattle identification plays a crucial role in many agricultural technology applications. Due to recent advances in deep learning, there has been a great deal of research exploring non-invasive methods for identifying cattle through visual features. Most methods focus on distinct anatomical features of the animals, such as their body patterns, facial characteristics, and muzzle prints [44]. Each cow's face presents a unique set of features, akin to human faces, providing a reliable means for individual identification. Similarly, a cow's muzzle, with its unique and time-invariant patterns, serves as an analogous biometric trait to human fingerprints, offering a robust basis for identification purposes [45]. In addition, the distinctive black and white patterns found on Holstein cattle further enable their identification through body patterns.

Several studies have explored these approaches with varying focus and technological applications. Identification from close-up cameras in feeding areas is a common approach. Identification has been performed through the localisation and classification of ear tags [46]. Facial recognition technologies have been applied to cows [47, 48], where the faces are typically localised via a YOLO [49] detection model before applying a CNN model for identification. Identification of muzzle images has also been performed both using conventional image processing techniques such as SURF [50] as well as deep learning [51]. However, the main problem with the mentioned approaches is the reliance on a close-up image of the area of interest, limiting the area of which identification can be performed over.

Identifying cattle in free-moving environments like barns or fields introduces significant challenges due to variations in lighting, positioning, and camera angles. [52] perform identification of cattle from various side-on cameras in a field, achieving over 99% accuracy using CNN models. However, their method performs identification on images containing a single cow, so it is unclear how the model would perform with multiple cows present in the image or with occlusion. Similarly, [53] performs identification from side-on views using a cascaded approach using DeepOtsu for image binarization and an EfficientNet classification model, enhancing training speed and efficiency. Again, it is unclear how this approach can handle occlusions. The [54] address this issue by performing identification from aerial barn cameras. Their approach involves classifying features extracted from binary masks of patterns, segmented by an improved Mask-RCNN model. Though effective under certain conditions, this method's performance can falter with less distinct patterns and is hindered by its non-end-to-end training nature, increasing development costs. Alternatively, [55] applied a ResNet model trained with deep metric learning for identification in a similar outdoor environment, revealing the potential for robust identification solutions from a single end-to-end CNN model. In [56], the authors identify non-Holstein cattle from videos using a joint CNN and LSTM architecture. However, other studies have shown that Holstein identification can be performed reliably on image data, making this approach relatively computationally expensive.

3.4 Super-Resolution Techniques

The vast majority of modern Super-Resolution (SR) methods utilise deep learning in some capacity. CNNs were initially the most popular choice that started with SRCNN [16]. Since then, many improved CNNs for SR have been proposed. [57] proposed a model which performs feature extraction in the LR dimensions before using deconvolution to learn the interpolation function up to the target resolution. [58] proposed the far deeper VDSR model to improve performance over SRCNN. The authors also incorporated a global residual connection to simplify the SR problem and help alleviate the vanishing gradient problem as well as improve convergence times.

One of the main challenges with SR is that it is particularly challenging to recover the high-frequency information that produces the fine details which are lost in the LR domain. Several models using the Generative Adversarial Network (GAN) framework [59] have been proposed to recover these details by learning the distribution of the high-resolution (HR) images. SRGAN [60] was the first proposed GAN for SR which produced more detailed images than previous methods at the time. However, SRGAN was prone to producing aesthetically unappealing artefacts, so ESRGAN [61] made improvements to the model architecture, adversarial loss, and perceptual loss functions of SRGAN, to further improve visual quality and reduce artefacts. Due to the synthetic generation of LR pairs during training, this model's performance still dropped when applied to real-world data. To alleviate this issue, Real-ESRGAN [15] was proposed, which introduces a more complex high-order degradation model to better simulate real-world data and improve performance on real-world data.

In recent years vision transformers [25] have grown in popularity for SR. SwinIR is proposed in [62] which leverages the Swin Transformer architecture to achieve superior SR results over CNNs. The authors of [63] proposed a hybrid Transformer CNN model named ESRT which improves long range dependencies between similar local patches within the image. The authors of [64] propose a Transformer in Transformer Network (TnTViT-G) for the task of guidance super-resolution. However, the main drawback of using transformer architectures is that they often require vast amounts of training data to produce good results.

3.4.1 Hyperspectral Image Super-Resolution

Hyperspectral imaging stands as a pivotal technology in remote sensing, playing a critical role in various domains, including material classification, mineral exploration, and environmental monitoring [10]. The success of automated post-processing tasks closely depends on the image's spatial and spectral resolutions. However, obtaining a high-quality Hyperspectral

Image (HSI) that boasts high spectral and spatial resolutions is challenging due to sensor limitations [13]. This often results in a trade-off, where an increase in spectral resolution comes at the cost of reduced spatial resolution [14]. Enhancing the lost spatial resolution is thus essential for improving the efficiency of post-processing operations.

In a manner akin to Super-Resolution (SR) in RGB imaging, Hyperspectral Image Super-Resolution (HSI-SR) aims to construct Hyperspectral Image (HSI) HSIs from low-resolution (LR) counterparts. This capability becomes particularly important due to the inherent spectral-spatial resolution trade-off associated with hyperspectral imaging. Through SR, it is possible to improve spatial resolution without compromising spectral quality. HSI-SR methodologies can be divided into two primary categories: single-image HSI-SR [65, 66], and fusion-based HSI-SR [67, 68, 69]. Given that fusion-based approaches necessitate an additional image from a different modality — a requirement not met by the datasets discussed in this thesis — this review will exclusively focus on single-image HSI-SR. Herein, the term single-image HSI-SR will be used synonymously with HSI-SR.

The authors of [70] were the first to introduce CNNs to HSI-SR by applying transfer learning from a 2D RGB SR CNN model. However, the 2D convolution operations used within this model do not fully exploit the spectral correlation between spectral bands. Therefore, the authors of [71] proposed a 3D CNN model to extract spatial and spectral features together for improved performance and spectral fidelity. Since then, many more 3D models have been proposed [72, 73]. The authors of [17] proposed a 3D CNN model for HSI-SR which uses dilated convolution and their proposed non-linear mapping blocks to improve efficiency. Given the large memory requirements of 3D CNNs, feature extraction is generally performed in the LR space before being interpolated to the target size by deconvolution [74] or pixel shuffling [75], reducing memory usage. However, applying deconvolution to extracted features is more challenging than simply performing SR on an input image already interpolated to the target resolution.

The GAN approach to SR has also been proposed for HSI-SR. The authors of [72] propose a GAN containing band attention modules. The authors of [76] propose a GAN architecture with an enhanced spatial attention module and a refined spectral attention module along with an attention-enhanced generative loss to produce more detailed images. However, the fine-details predicted by GANs are often visually appealing, but not objectively accurate, meaning the loss functions are often heavily constrained with traditional pixel-wise loss functions.

Vision Transformers have recently been introduced to HSI-SR. The authors of [77] propose a hybrid transformer and 3D CNN model which consists of a transformer branch to improve spatial reconstruction and a 3D CNN branch to enhance spectral reconstruction. The two branches are connected through several interactive attention units which help to share learned features between the two branches. Due to the fact that the number of token multiplications is quadratic relative to the sequence length within transformers, memory saving tricks generally have to be applied for practical use in the context of HSI-SR. The authors of [78] propose ESSAformer which is a transformer architecture that uses an efficient SCC kernel-based self-attention mechanism to relieve the computational burden of performing self-attention on HSI. The common issue of transformers' vast data requirements still applies in the HSI-SR domain, which combined with their large memory requirements, limits their practicality in this domain.

3.5 Operational Neural Network

Advances in deep learning have resulted in CNNs dominating many computer vision fields. Part of the reason for their success is their ability to learn complex non-linear operations which can extract discriminative features from a given image. However, convolution itself is a linear operation and the non-linear components of the networks are solely provided by the activation functions used after each convolutional layer in the network. This means that CNNs often have to be very deep in order to have the necessary non-linear capacity and diversity to learn the complex function of the learning problem.

Recently, Operational Neural Networks (ONNs) [79, 80] were proposed to address this issue by extending the idea of the Generalised Operational Perceptron [81] to convolutional models by incorporating non-linear nodal and pooling functions that replace the sole convolution operation with any non-linear operator. The addition of these functions incorporates significantly more non-linear components to the network than a traditional CNN, increasing its theoretical non-linear capacity. However, these additional non-linear operations are hard coded and thus cannot be changed during training. This means that the functions need to be searched for, which is computationally expensive, and the search space is limited to the function set, which may not contain the optimal function or functions.

The authors of [79] then extended upon their original work to address these limitations by proposing Self-Organised Operational Neural Network (Self-ONN) [82] which makes the linear filters of a standard CNN non-linear through the use of MacLaurin series expansions, rather than applying hard-coded functions. Such non-linear filters for each kernel element are learnable during training, and thus eliminate the need for an exhaustive search to find the optimal functions. Furthermore, almost any function can theoretically be approximated using MacLaurin series expansions, which means that a Self-ONN is not limited to a specified function set, allowing for a comprehensive non-linear search space. These improvements mean that Self-ONNs are far more computationally efficient than their standard ONN counterparts, with greater theoretical non-linear capacity than both their ONN and CNN counterparts. This additional complexity comes at the cost of each filter requiring more parameters. However, the network size of a Self-ONN can be much smaller than that of a CNN to have the same or increased theoretical non-linear capacity, allowing for the overall model to have fewer parameters than a CNN despite each individual filter containing more parameters. In many applications [83, 84, 85, 86, 87, 88] Self-ONNs have been shown to outperform the deeper and more complex CNNs with far fewer parameters.

3.6 Self-Supervised Learning

Self-supervised learning is a type of machine learning that allows models to learn from the data itself without relying on labels. This is typically achieved by training the model to learn one part of any given input from another, turning the unsupervised problem into a supervised problem by auto-generating the labels. This is particularly useful in situations where labelled data is scarce or expensive to obtain, which is a common challenge in many machine learning applications.

The two main methods to perform self-supervised learning are invariance-based methods and generative methods. Invariance-based methods aim to produce similar embeddings for two or more views of the same image whereas generative methods aim to predict or reconstruct removed portions of the input image.

SimCLR [89] is a well known invariance-based method which works by adding a multilayer perceptron head to the given network architecture to train. The model is then passed augmented views of data samples and minimises the distance between augmented views from the same data sample while maximising the distance between these views and a different data sample via a contrastive loss function. The authors then make several improvements in their updated SimCLRv2 [90] by introducing deeper models with selective kernel channel-wise attention, introducing a memory bank for enhanced negative sampling, and using a deeper head where finetuning is applied to the middle layer of the head. These methods were shown to be effective, providing ImageNet performance on par with supervised methods at the time. However, there are several challenges with the proposed approach including sensitivity to the augmentations and batch size used, and challenges with negative sampling, particularly when the number of classes in the dataset is small.

To overcome these limitations, Bootstrap Your Own Latent (BYOL) [91] was proposed, which uses an online and a target network to predict similar representations of two augmented views of the same image. A predictor module is added to the end of the online network and the target network's weights are updated with the exponential moving average of the online network's parameters, in order to prevent representation collapse. Since BYOL does not rely on negative samples, it is more robust to the selected augmentations and batch size, providing performance gains over contrastive methods such as SimCLR.

Time contrastive learning [92] is another example of an invariance-based method, which exploits close and distant frames within videos to produce positive and negative pairs to be used in a triplet loss function. The obvious weakness with this approach is the reliance on video data.

DINO [93] builds upon BYOL by training a student (online) network and a teacher (target) network to produce the same class probabilities for augmented views of a given image but without using the additional predictor module, as is used in BYOL. The teacher network is updated using an exponential moving average of the student network's weights and the teacher model's outputs are centred with the exponential moving average of the batches to avoid representation collapse, and a small temperature term is added to the softmax layer of both models to also help prevent representation collapse. These improvements offer faster training speeds and improved performance over BYOL. DINOV2 [94] improves upon the original DINO method by introducing the patch level objective proposed in iBOT [95] where the student network is passed masked patches and the teacher is passed the unmasked patches. Other improvements are also made including the use of KoLeo regularisation to provide a better masking strategy during training, gradually increasing the resolution to 518x518 towards the end of training and several efficiency improvements to train at scale. iBOT [95] uses a patch level objective function where a teacher network is passed the original image and the student network is passed a masked version of the image. The student network then learns to recover the masked tokens based on the teacher network output where the models are jointly optimised via momentum updates.

Generative methods aim to reconstruct an image, which is either masked or compressed, to learn rich feature representation of the training images. This method is particularly well suited to transformer architectures where input patches can easily be removed which also has the added benefit of improving training times. Some of the previously mentioned invariance-based methods incorporate ideas of generative methods to further enhance performance.

Masked Autoencoding [96] is a prime example of a generative method, which exploits an encoder-decoder framework to produce an encoding of an image where a large portion of the image is deleted, before decoding this encoding to reconstruct the full image. This technique has been proven to be a highly effective feature extractor pretraining step as it does not rely on negative sampling or careful augmentation strategies and it forces the model to learn highly general features. However, this method was designed for transformer architectures and its reliance on positional embeddings means it is not applicable to CNNs.

The authors of [97] proposed an image-based joint-embedding predictive architecture (I-JEPA), to predict representations of various target blocks from a context block within the same image. By utilising a predictor model and computing the loss on the image representations instead of the pixels directly, the model learns better semantic information and greatly improves the performance of downstream tasks over other generative methods and is competitive with invariance-based methods while being far more computationally efficient.

3.7 Domain Shift

Domain shift refers to the phenomenon in which there is a distribution shift between the training (source) data and the test (target) data. This is a common problem when deploying models in practical settings, as out-of-distribution (OOD) scenarios are commonly encountered. Since most statistical learning algorithms assume that source and target data are independent and identically distributed, performance in these practical settings, where OOD examples are frequently encountered, often deteriorates significantly [98]. Domain shift is a problem in almost every application, but it has seen a particular focus in the areas of object recognition, semantic segmentation [99], face recognition, medical imaging [100], sentiment classification, speech recognition, and reinforcement learning [101].

3.7.1 Domain Adaptation

The simplest solution to address the domain shift problem is to utilise some of the target domain data in training. This approach is referred to as domain adaptation, which aims to create a model capable of performing well in a target domain by training with a readily available source domain and either a limited amount of labelled target domain data, a large amount of unlabelled target domain data, or some combination.

To address the challenge of domain adaptation, the authors of [102] propose a classification and contrastive semantic alignment (CCSA) loss function to train a model for use on target domain data by using very limited amount of labelled target domain data combined with an abundance of source domain data. Their CCSA loss combines a standard classification loss to be used on the source domain with a contrastive semantic loss, which brings feature representations of same-class source and target domain samples close together while separating opposing-class representations of source and target samples. They showed that their method greatly outperforms unsupervised domain adaptation techniques in challenging domains where there is a large covariate shift between the source and target domains. The authors of [103] utilise multiple classification heads with weights sampled from several learnable distributions to improve local alignment and offer improved domain adaptation in an unsupervised manner. In [104], the authors merge target domains to form a more realistic compound target domain where they leverage both a class encoder and a domain encoder to help the model learn to ignore factors across domains that are irrelevant to classification, which they show improves performance when the model is applied to new challenging domains.

[105] propose Memory-based Multi-Source Meta-Learning which computes a memory-based identification loss combined with a triplet loss on the meta-train data before updating the network and computing the meta-test loss which is enhanced via meta-batch normalisation. The meta-train and meta-test losses are then combined to update the original weights of the network.

However, the main limitation with domain adaptation methods is the reliance on the target domain data, which is not always available or, indeed, known.

3.7.2 Domain Generalisation

To address the domain shift issue in the scenario where there are no available target domain data, domain generalisation was introduced. The objective in domain generalisation (DG) is to train a model using data from one or several related but distinct source domains so that the model can effectively be applied to any OOD target domain. Depending on whether the model is trained on data from a single source domain or multiple source domains, domain generalisation can be categorised as either single-source or multi-source. Approaches to domain generalisation include using some form of augmentation strategy, aligning source domain distributions, or meta-learning, in order to produce a domain-invariant model.

DNNs can tend to be vulnerable to texture changes and small perturbations. To help make models more robust to this and consequently improve the model's ability to generalise across domains, the authors of [106] propose a data augmentation technique called RandConv. This technique applies a single layer CNN with random weights and kernel size to an input image to alter its texture will retaining the shape of the object(s) within the original image. They also propose a mix variant which mixes this augmented image with the original image by a factor α. Although RandConv reduces the performance slightly on the original domain, the authors show that the performance on new domains improves significantly, especially in more challenging domains. However, as the kernel size in the RandConv method increases, the semantic information deteriorates. To overcome this issue, Progressive Random Convolution [107] (ProRandConv) was introduced which uses multiple convolution layers with small kernel sizes, which better preserves the semantic information while producing richer texture diversity. Other modifications such as deformable convolution and random affine transformations are introduced to further enrich the texture diversity.

Research has shown that visual domains can often be characterised by image styles encapsulated within instance-level feature statistics in shallow CNN layers. To exploit this finding, the authors of [108] proposed MixStyle which probabilistically mixes feature statistics between training instances within the early layers of a CNN to synthesise novel domains and create more diversity in the training distributions without making any modifications to the training data itself.

In contrast to domain adaptation, the main challenge with domain generalisation is the absence of target data, which means that there is no guarantee that the methods will generalise well to OOD samples. Furthermore, many domain generalisation methods rely on domain labels for training, limiting their practicality in many applications where such labels are either unavailable or difficult to define.

4 Image Classification with Similarity Learning

In this chapter, the challenge of image classification capable of handling dynamically changing classes is addressed. Traditional classification models are typically trained to recognise a fixed set of classes, making them unsuitable for scenarios where new classes may emerge over time or where previously unseen data must be classified without retraining the model. This limitation is particularly problematic in applications that require real-time adaptability and scalability, such as monitoring systems, dynamic inventory tracking, and biometric identification. To overcome this challenge, novel solutions are proposed based on similarity learning [34], a technique originally introduced by Google for facial recognition. Unlike traditional classification approaches, similarity learning projects data into a high-dimensional embedding space, enabling flexible and scalable classification that is not restricted to a fixed set of predefined classes as new classes are simply projected into a new position within the embedding space.

To validate the proposed solution, the real-world application of individual dairy cattle identification is selected. This application serves as an excellent test case for the generalised problem of handling changing classes due to the unique characteristics of dairy farming. Herds are dynamic, with animals being added and removed over time, making it impractical to rely on fixed-class classifiers. Moreover, accurate individual identification is critical for various dairy farming tasks, such as tracking health, monitoring productivity, and ensuring animal welfare, highlighting the research importance of this application. The embeddings produced by similarity learning enable the model to identify cattle not included in the training dataset, demonstrating its ability to generalise to new classes. Novel analysis of this capability is conducted in this chapter.

In addition, this chapter introduces novel Self-Organised Operational Neural Network (Self-ONN) architectures to improve the parameter efficiency of identification models, addressing computational constraints often encountered in large-scale deployments. By combining the adaptability of similarity learning with the efficiency of Self-ONNs, the proposed approach offers a scalable and practical solution to the broader problem of dynamically changing classification tasks, while also offering highly accurate cattle identification models.

The novel contributions of this chapter are listed as follows:

- New class evaluation for dairy cattle identification
- Self-ONN architectures for dairy cattle identification trained using similarity learning

4.1 Datasets

Datasets containing top-down views of individual cattle were collected with the industrial sponsor of this project - Peacock Technology Limited. Initially, a relatively small dataset containing 8055 images of 537 cows was collected which will be referred to as the CowID-537 dataset. Then, later in the project, when more resources were available, a much larger dataset consisting of 1785 cows and 161663 images was collected, which will be referred to as the CowID-1785 dataset.

A brief overview of all the datasets used in this chapter is shown in Table 4.1.

Dataset Name	Number of Images	Number of Classes	Images Per Cow	Labelled
CowID-537	8055	537	15	Yes
CowID-1785	161163	1785	17 - 134	Yes

Table 4.1: Cow Identification Datasets Overview.

4.1.1 CowID-537 Dataset

Cow image data was acquired by positioning an IP camera above the Radio Frequency Identification (RFID) scanner in the rotary milking parlour within the target farm as shown in Figure 4.1. Each new RFID detected by the scanner triggered a sequence of three image captures at fixed time intervals to capture the cow at different angles. In the parlour, the cows are placed in compact parallel pens; this means that for each image capture there are up to 3 cows present in the image since the camera was centred around the RFID scanner.

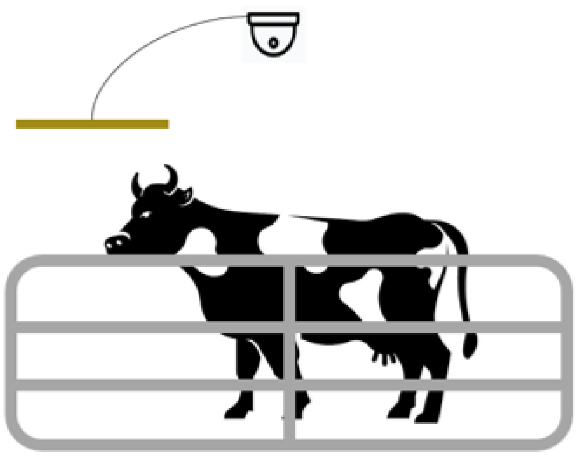

Figure 4.1: Milking Parlour Data Capture Setup
Cows are put into rotary milking parlour pens and a single camera and RFID scanner capture images of the cow from above when the pen passes under.

Since the camera would capture a larger area than just the cow itself, as can be seen in Figure 4.2, it was necessary to develop an object detection model to isolate the cow from the entire image. A subset of the data was annotated with bounding boxes and used to train a YOLOv3 [49] model to detect cows. The developed YOLOv3 model was then applied to all the acquired images and all detections (including detections of other cows and erroneous detections), as shown in Figure 4.2, were saved into folders corresponding to the triggered RFID. Due to the RFID scanners not being 100% accurate and the fact that cows would occasionally move their heads into adjacent stalls and trigger the scanner, human verification was then applied to remove all incorrect cows and erroneous detections from each RFID folder. In cases where it was unclear which cow was the correct cow in any given folder, the entire RFID folder was discarded. The final dataset consisted of 537 unique cows, each with 15 unique images.

4.1.2 CowID-1785 Dataset

Another larger cow dataset was acquired from the same farm at a later date to expand and improve upon the previous dataset. This dataset was acquired with the same camera and

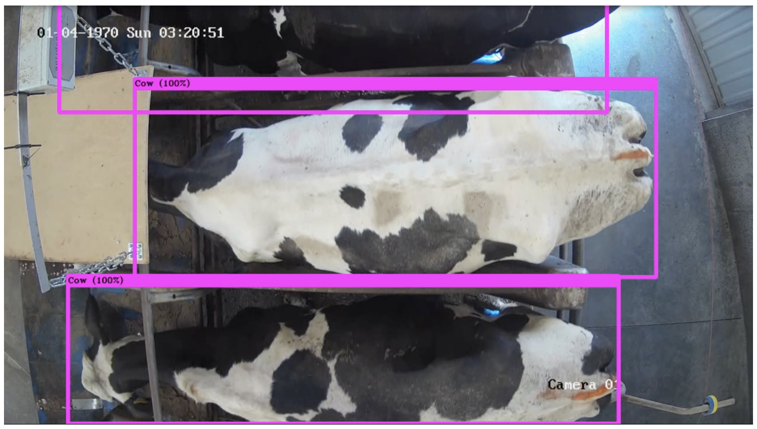

Figure 4.2: Data Cropping with YOLO

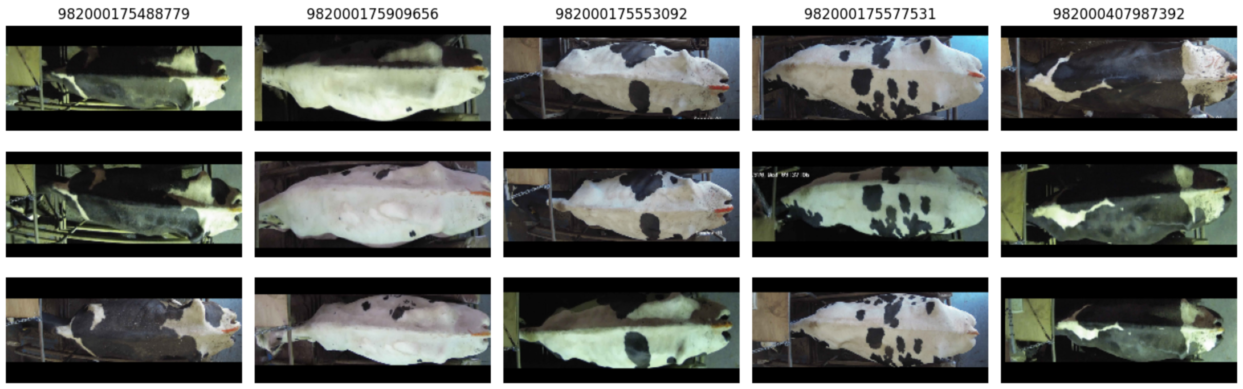

Figure 4.3: CowID-537 Dataset Examples

RFID scanner setup as the CowID-537 dataset but captured images of a total of 1785 cows with between 17 and 134 images per cow. The capture process for this dataset was more reliable than the previous process as the previously developed YOLOV3 model was employed on each image capture as shown in Figure 4.2 and only the centre cow detection was saved to the folder corresponding to the RFID trigger. Thus reducing the occurrence of erroneous cows being contained within a capture folder.

Human verification was still required to verify the clean the data as it was still possible that an incorrect cow could appear in a given capture folder for the same reasons as the CowID-537 dataset. The human verification process was different for this dataset and a new verification policy was adopted so as not to exclude cows that were extremely challenging for humans to distinguish between. This could be more reliably achieved due to the improved capture process making it less likely that erroneous cows were contained within each folder compared to the capture process on the CowID-537 dataset.

The dataset was divided between five separate people to validate, where each person was assigned a different portion of the cows to validate. Each person would go through each image for each individual cow and remove images of any obviously wrong cows in a given RFID folder. Any images that were not obviously wrong were left in, including any images that were challenging to tell whether it was a different cow or the cow itself was dirty as seen in Figure 4.4, covering some of its features. This resulted in no single cow being completely removed from the dataset, keeping all the potentially more challenging cows in. The entire process took around 8 hours per person.

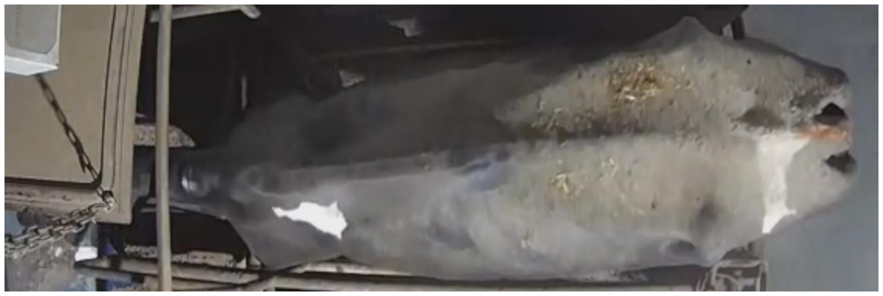

Figure 4.4: Dirty Cow Example

All the images were passed to an identification model (trained on the CowID-537 dataset) after each person had gone through and removed all their obvious erroneous cows. The model provided suggestions for potential mistakes made by the five verifying people. Three of the people then reviewed these suggestions together and again removed the obvious erroneous cows that had been missed initially.

This model filtering process also highlighted that some cows had duplicate RFIDs, i.e. that the same cow would have a certain number of images under one RFID and more under another. This was likely due to said cow having lost its RFID tag and then having it replaced with another tag with a separate RFID. On this assumption, all RFID pairs where there was not at least two days of overlap (as there could be one day of overlap if the RFID tag was lost and replaced on the same day) between the image captures were highlighted. All these potential pairs were manually reviewed and 7 pairs were identified as the same cow. These pairs were then merged into the folder with the newest RFID for the given cow. Some examples of the CowID-1785 dataset can be seen in Figure 4.5.

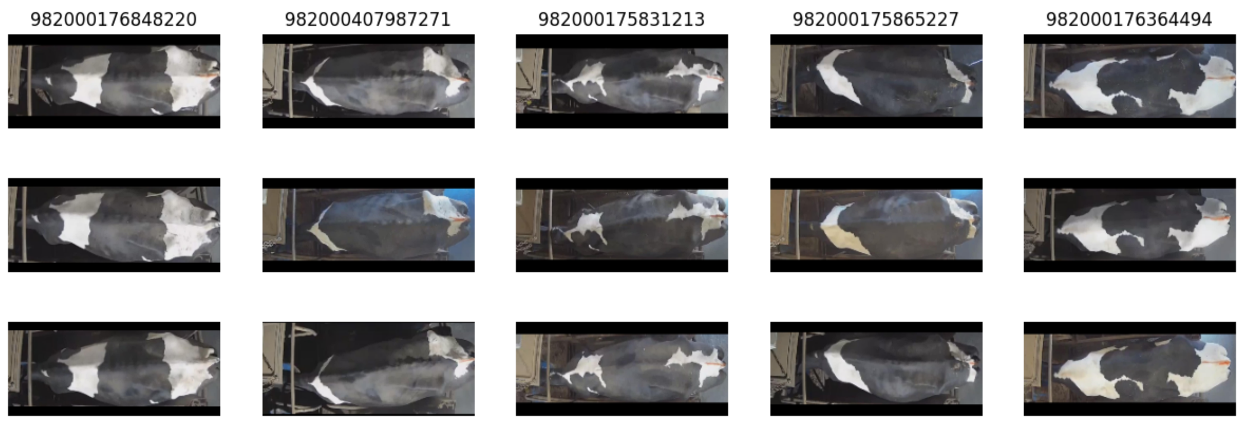

Figure 4.5: CowID-1785 Dataset Examples

4.2 Cow Identification with Similarity Learning and New Class Evaluation

Similarity learning is a training technique that focuses on producing meaningful embeddings rather than maximising class probability outputs. In this approach, the model learns to generate embeddings that are close in Euclidean distance for samples of the same class and farther apart for different classes. This method enables the model to learn more robust features by optimising its representation space directly.

A key advantage of similarity learning is that it does not constrain the model to a fixed number of classes. Since the model produces embeddings rather than class-specific outputs, it can generate representations for new classes that the model has not been trained on, making it highly adaptable. In this section, cow identification models are developed using similarity learning, and a novel analysis of their performance on new classes - classes not contained within the training dataset - is conducted. This evaluation assesses the real-world effectiveness of these models in dynamic herd environments where individual cows change over time.

4.2.1 Methodology

The CowID-537 dataset was used for the experiments in this section where 500 cows were used for training and the remaining 37 were reserved for new class evaluation. Due to the limited size of the dataset, 3 fold cross-validation was applied to prevent selection bias, where on each fold a different random subset of 37 cows was selected for new class evaluation. Since this dataset contains 15 images per cow, the images were split into 3 random groups of 5 images where each group would be used in one of the cross validation folds for validation and testing (where 3 of the 5 images were used for validation and 2 for testing). The split of the dataset can be seen in Figure 4.6 with some example images shown in Figure 4.7.

In the initial paper where similarity learning was first used for facial recognition [34], the authors use an Inception ResNet model which has a carefully designed topology for their facial recognition task, which provides excellent results. However, due to this specific design, it does not perform as well when applied to other tasks as the more generic models such as ResNet [21] and VGG [20], which essentially have a stacked block-type architecture designed for general-purpose classification. One of the reasons that Inception ResNet models can offer superior performance to their more generic rivals, is because they use a split-transform-merge strategy, which essentially gives the effect of an ensemble as multiple feature extractors are applied to the input and the results are combined. The ResNeXt architecture [22] exploits this technique, but instead of using specially designed, different feature extractors, it uses a reduced ResNet architecture on each branch, which has been shown to offer performance improvements while not requiring the network to be specially designed for any given task. For this reason, the ResNeXt architecture was selected for experiments as it is a general purpose architecture with the added benefit from split-transform-merging.

The Adam optimiser [30] was used for all experiments as this is a powerful, diverse and well tested optimiser that is commonly used by many researchers and practitioners.

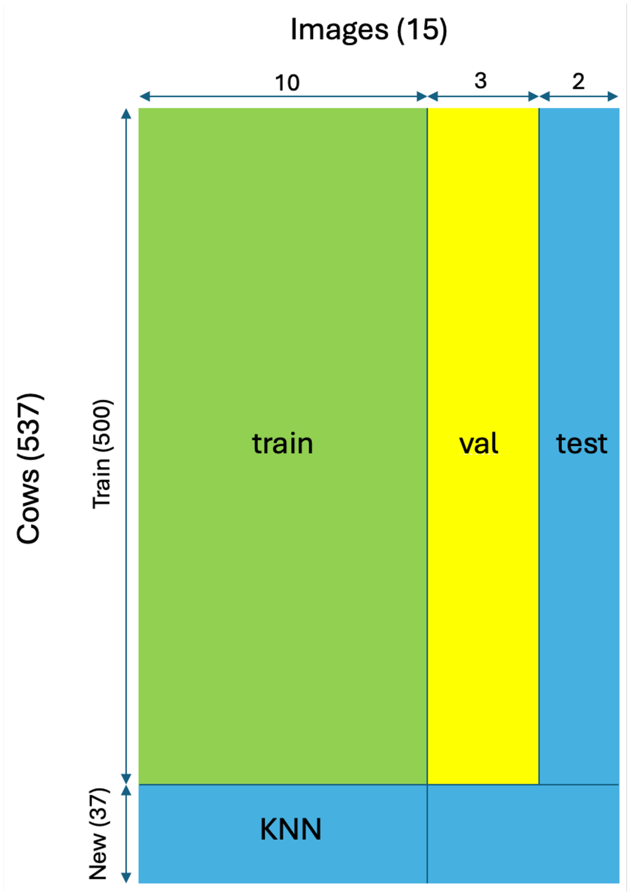

Figure 4.6: CowID-537 Dataset Split.
The green portion represents images that were used to train the model. The yellow portion represents images that were used to validate the model during training. The blues portions represent images that were reserved from the training process that the model has never seen before.

The triplet loss function [34] was selected as this is a well tested similarity learning loss function that can take advantage of fine-grained data relations, is computationally efficient and contains few hyperparameters. Online triplet mining [34] was applied to select triplet combinations from each mini-batch fed to the model where each batch contained images from 25 random cows with 4 random images per cow to ensure that a sufficient quantity of potential triplet combinations was available for selection. The hard and hardest triplet selection strategies were explored where the hard triplet selection strategy is defined as selecting all triplet combinations within the mini batch that satisfy Eq. (2.18) and hardest triplet selection is defined as the triplet combination producing the largest value from Eq.

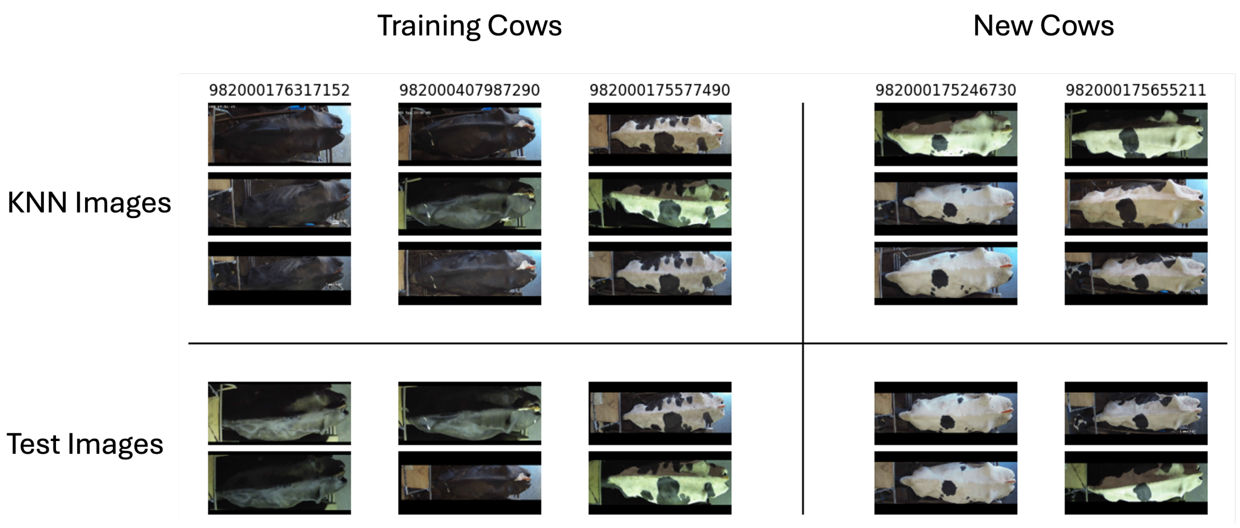

Figure 4.7: Training Cows vs New Cows Example.
Top left are images of cows used in training and also to fit the KNN for evaluation. Bottom left are images reserved for testing of the cows used in training. Top right are images used to fit the KNN of cows not used in training. Bottom left are test images for cows not used in training.

(2.18).

Random augmentations were applied to each image before it was passed to the model during training, as this is standard practice during deep learning training to improve performance and generalisation capacity. The augmentations used during training can be seen in Figure 4.8 and the parameters used are shown in Table 4.2 where the probability of any augmentation being applied to a given training image was 80%.

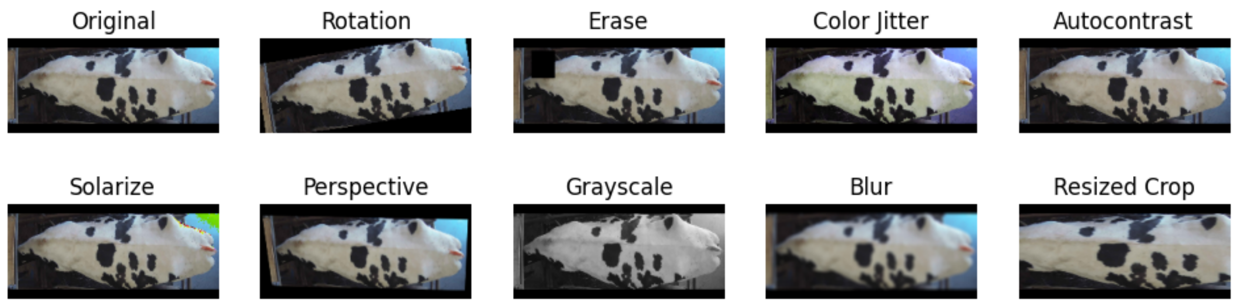

Figure 4.8: Training Augmentation Examples.

Random search optimisation [109] was used to tune the training parameters as this has been shown to be far more efficient than grid search while achieving similar results. The

Table 4.2: Augmentation parameters used during cow identification training.

Augmentation	Probability (%)	Parameters
Rotation	100	$[-10, 10]$ degrees
Erase	50	$[0.02, 0.05]$ scale, $[0.3, 3.3]$ ratio
Colour Jitter	100	$[0.8, 1.2]$ brightness, $[1.8, 1.2]$ contrast, $[0.85, 1.15]$ saturation, $[-0.1, 0.1]$ hue
Autocontrast	50	-
Solarize	50	0.95 threshold
Perspective	50	0.15 distortion scale
Greyscale	10	-
Blur	50	5 kernel size, $[0.1, 2.0]$ sigma
Resized Crop	50	$[0.8, 1.2]$ scale, $[0.75, 1.33]$ ratio

parameters that were optimised include the learning rate, learning rate decay, gradient accumulation frequency (in batches), the triplet margin, and the use of either hard or hardest triplet selection strategy.

Early stopping was applied to prevent overfitting by selecting the model weights from the epoch that produced the highest validation accuracy. Testing was then applied by embedding all 10 training images for each cow and then using these embeddings to fit a KNN classifier and then perform testing on the reserved test images. A K value of 5 was selected for the KNN as each cow contained 10 embeddings in the KNN embedding space, so a value of 5 would strike a good balance between reducing the impact of noise on the results, while also not requiring all 10 embeddings to be used in the prediction. For the new class evaluation, a similar approach was taken where 10 random images from the reserved new cows/classes (as no images were used for training) were embedded and used to fit the 5NN classifier. All 5 remaining images were used for testing, as none were used for validation.

The entire training and evaluation framework can be seen in Figure 4.9.

4.2.2 Experimental Results

For each cross-validation fold, 50 random search iterations were applied to tune the hyperparameters. The results can be seen in Table 4.3. Since many optimisation iterations did not yield good results due to the random nature of the optimisation algorithm, only the 3

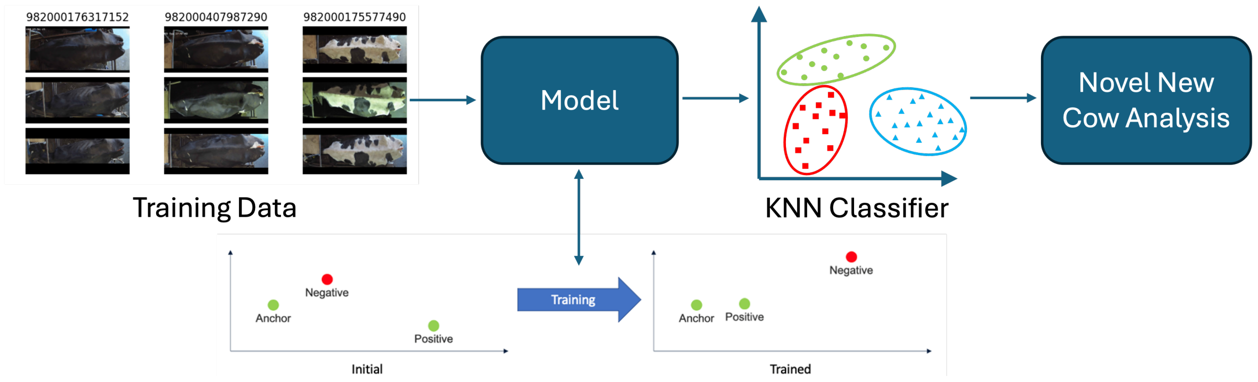

Figure 4.9: Cow Identification Framework.

best results for each fold are reported in Table 4.3 for the sake of clarity. The entire set of results is shown in Appendix 7.5.

Table 4.3: CowID-537 Hyperparameter Optimisation Results.

Learning Rate	Learning Rate Step	Accum Iter	Triplet Margin	Hardest Triplets	Accuracy	F1 Score	New Class Accuracy	New Class F1 Score
Fold 1								
0.00018	529	1	0.1	FALSE	98.27	0.982	92.43	0.94
0.000636	568	45	0.2	FALSE	98.2	0.981	90.81	0.922
0.000749	947	3	1	TRUE	97.67	0.975	81.08	0.861
Fold 2								
0.000912	490	10	0.2	FALSE	98.6	0.986	94.05	0.955
0.00036	488	4	0.5	TRUE	98.4	0.983	94.05	0.943
0.000794	354	2	0.1	FALSE	97.47	0.973	94.59	0.95
Fold 3								
0.00081	641	1	0.1	TRUE	99	0.99	94.05	0.962
0.000611	170	1	0.1	TRUE	98.73	0.987	91.35	0.941
0.000813	215	1	0.1	TRUE	98.33	0.983	93.51	0.952

Results from random search optimisation on each cross-validation fold on the CowID-537 dataset. Only the top three random search iterations (in terms of accuracy) are reported for each fold where the iterations are ordered top to bottom in terms of accuracy. Each fold consists of a unique split of 500 training classes and 37 reserved new classes.
Learning Rate, Learning Rate Step, Accum Iter (gradient accumulation frequency in batches), Triplet Margin and Hardest Triplets are the training hyperparameters used.
Accuracy and F1 Score are calculated on the test images of the 500 cows used during training.
New Class Accuracy and New Class F1 Score are computed on the test images of the reserved 37 cows (Figures 4.6 & 4.7) that are not used during training. All 537 cows are used to fit the KNN so that each of the 37 test cows may be classified as any of the total 537 cows.

4.2.3 Discussion

The results from Table 4.3 show that consistent performance is achieved across each of the cross-validations folds, with test accuracy of up to 99% on the standard test set and up to 94.05% on the new cow test set. Although there is around a 5% accuracy drop in the new cow test results compared to the standard test set results, the new cow performance is still very good considering the model does not have prior information about these cows from training. This demonstrates the efficacy of this training technique to learn robust and general features to distinguish between individual cattle even with a very limited amount of training data present. Another interesting point to note is that there is a great deal of variation in the hyperparameter values between each of the best optimisation runs, indicating that the performance of this method has little sensitivity to the training hyperparameters.

4.3 Operational Neural Networks for Parameter Efficient Classification

A larger cow identification dataset was later acquired from the same target farm as the CowID-537 dataset, containing a far greater number of cows and images than the previous version. This newer dataset, referred to as the CowID-1785 dataset and presented in Table 4.1, contains over three times as many cows as the CowID-537 dataset, representing a more challenging problem. In response, novel Self-ONN extensions of the CNN models used in the previous section are proposed which are shown to provide on par performance, but with improved parameter efficiency. These novel models are also trained using similarity learning, which has not been performed previously. Further new class analysis is also conducted to examine the generalisation capabilities of the proposed models. The entire training and evaluation framework remains the same as in Figure 4.9, although in this section novel contributions are also made to the model.

4.3.1 Methodology

To train and evaluate each model the CowID-1785 dataset was split into a training, validation, and test portion where 70% of the images for each cow were used for training, 15% for validation and the remaining 15% for testing. The same training, validation and testing split was used for all experiments and in this case cross validation was not used due to the size of the dataset. To test the model's ability to perform on new classes in a similar manner to the experiments conducted on the CowID-537 dataset, the CowID-537 dataset was again utilised. However, instead of only using 37 of the 537 cows, as previously done, the entire dataset was used for new class testing in these experiments. Since both the CowID-537 and CowID-1785 datasets were captured from the same barn, there are cows that appear in both datasets. However, since cows can have their RFIDs changed over time, it is not guaranteed that the same cow will contain the same RFID in both datasets, meaning that training cannot be performed on both datasets simultaneously. Hence, the CowID-537 dataset was selected for new class testing. Although there are cows contained in both datasets, there is still a large portion of cows in the CowID-537 dataset that are not contained within the CowID-1785 dataset, meaning this evaluation will still provide valuable insights into the trained models' ability to generalise to new classes. Furthermore, using a mix of cows used and cows not used in training is more representative of a scenario when such a model would be deployed.

Experiments were conducted using the ResNeXt architecture applied to the CowID-537 dataset. In addition, the squeeze-and-excitation (SE) extension [24] of the ResNeXt model [22] was also selected for experiments. SE modules enhance the representational power of a network by explicitly modeling channel-wise dependencies. They achieve this by first "squeezing" global spatial information into a channel descriptor through global average pooling, and then "exciting" the channels by learning adaptive weights that recalibrate each channel's importance. This mechanism allows the network to emphasise informative features and suppress less useful ones, leading to improved performance with only a minimal increase in the number of parameters compared to the standard ResNeXt model. This model will be

Table 4.4: Cow Identification Model Parameters

Model	Embedding Size	# Parameters
ResNeXt	32	23045472
	128	23242176
SE_ResNeXt	32	25576464
	128	25773168
SE_ResONeXt	32	17894488
	128	17992888

referred to as SE_ResNeXt. Furthermore, a novel Self-ONN variant the SE_ResNeXt architecture, referred to as SE_ResONeXt is proposed which extends the convolutional layers of this models to more powerful non-linear Self-Operational layers [82]. These layers introduce additional higher-order terms to each node in each convolutional filter to turn said node into a learnable MacLaurin series function approximation with Q terms. A Q value of 3 was selected for all proposed Self-ONN as this provided a good balance between function approximation and parameter increase as each Q term introduces a new parameter to each node. However, a Q of 3 still nearly triples the number of parameters the network contains (derived from Eq. (2.14). Therefore, the number of filters in each layer of the proposed Self-ONN model was halved in order to bring down the number of network parameters to below what the equivalent CNN architectures contain. The total number of parameters per model is reported in Table 4.4.

Due to the fact that the MacLaurin series expansions are used within Self-ONN models, learned function approximations are only accurate close to 0. Therefore, Tanh activation functions were used instead of ReLU functions within each proposed Self-ONN model to ensure data passing between layers was bound to the interval $[-1, 1]$. This also helps stabilise training as it ensures no large values occur as a result of raising a value greater than 1 to a large power within the MacLaurin series expansions.

The training parameters for these experiments were selected based on the results from the CowID-537 dataset experiments presented in Table 4.3. An initial learning rate of 0.001 was selected as this was slightly higher than the average value of the learning rates presented in Table 4.3 to encourage faster convergence on the larger dataset. A learning rate decay of a factor of 10 was applied at epoch 500 as this is approximately the average decay step from the earlier results. A 0.5 triplet margin was selected as this is slightly above the average triplet margin in Table 4.3, encouraging more separation between the embeddings of different classes. The hard triplet mining strategy was selected as there is roughly a 50/50 split between the use of this strategy and the hardest strategy in Table 4.3, so this strategy was selected as it considers more triplet combinations in each training step. The more modern AdamW optimiser [31] is used with a weight decay of 0.005 as this has been shown to provide better generalisation capacity than the standard Adam optimiser.

Hyperparameter optimisation was not applied, as the CowID-1785 dataset takes significantly longer to train. Furthermore, the results in Table 4.3 suggest that the training technique is fairly insensitive to hyperparameters, indicating that hyperparameter optimisation may not be strictly necessary. The random augmentations from Table 4.2 were applied again to each raw image before being passed to the model with a probability 80% that any augmentation would be applied. Note, a fresh random augmentation combination is applied to the raw image every time it is passed to the model during training.

Each model was trained until the validation accuracy had not improved for 250 epochs, where the final weights used for testing were selected from the epoch producing the highest validation accuracy. Training times for each individual model would range between 1 and 11 hours and would on average take around 3.5 hours. Once training had completed, all training images were embedded and used to fit a KNN classifier which was used to perform classification on the test images. K values of 3, 5, and 7 were tested and the best result was reported. New class testing was performed in a similar manner where 10 of the images from each cow in the CowID-537 dataset were embedded and used to fit a KNN classifier which then performed classification on all of the remaining images. Again, a K value of 3, 5 and 7 was tested, with the best result being reported. Inference times for an individual image were near instant and the total inference time for all test images for each model, including the time it takes to fit the KNN, was less than a minute.

4.3.2 Experimental Results

The results of the experiments carried out on the CowID-1785 dataset can be seen in Table 4.5.

Table 4.5: CowID-1785 Results.

Model	Training Classes				New Classes			
	NN	Accuracy (%)	F1 Score	Silhouette Score	NN	Accuracy (%)	F1 Score	Silhouette Score
ResNeXt_128	3	99.693	0.9969	0.7672	3	92.588	0.9235	0.3973
SE_ResNeXt_128	3	99.701	0.997	0.774	3	91.62	0.9154	0.3889
SE_ResONext_128_q3	7	99.602	0.99602	0.76219	3	88.715	0.8849	0.3009
ResNeXt_32	3	99.689	0.9969	0.7718	3	92.775	0.9258	**0.4074**
SE_ResNeXt_32	7	**99.73**	**0.9973**	**0.7899**	3	**92.924**	**0.9268**	0.4036
SE_ResONeXt_32_q3	7	99.709	0.9971	0.7691	3	89.013	0.8854	0.3122

Models with O in the middle and q3 at the end are the Self-ONN architectures and all others are CNN. The numbers 32 and 128 denoted at the end of each model name represent the output embedding size of the model.

4.3.3 Discussion

The results from Table 4.5 show that all models produce very high accuracy on the training classes. There is a similar performance drop observed in the experiments conducted on the CowID-537 dataset in terms of the new class accuracy, though this performance is still very good all things considered. The Self-ONN models perform only marginally worse than their CNN counterparts, with a reduction of less than 0.1% in training class accuracy. However, this slight performance drop is outweighed by an approximately 30% reduction in parameters. This parameter efficiency also translates into substantially lower GPU memory requirements, enabling inference on smaller and more cost-effective GPUs - a particularly valuable advantage for agritech applications, where models are often deployed at the edge. The Self-ONN models do however produce a larger performance drop on the new classes, suggesting that the additional non-linear complexity of these models may make them slightly more prone to overfitting, even with less total parameters.

It is also observed that using a smaller embedding size improves model performance, especially on the new classes. This is somewhat counter-intuitive as one would expect that it is easier to separate embeddings in a larger space. However, the smaller embedding space may force the model to make more efficient use of the embedding space, consequently improving performance. Furthermore, the smaller embedding size reduces the overall model parameters slightly, potentially improving generalisation performance which is evident in the improved new class performance.

4.4 Summary

In this chapter similarity learning was explored for individual Holstein cow identification. First, similarity learning experiments were conducted on the CowID-537 dataset where novel new class analysis was performed. The results showed that models trained in this way can generalise well to classes the model has not been trained on, though there is a small drop in performance relative to the classes the model has been trained on.

Experiments were later conducted on the larger CowID-1785 dataset where novel Self-ONN models were proposed and these models were trained using similarity learning, which has never before been done. The results demonstrated that the proposed Self-ONNs models provided similar performance to the CNN models used in experiments, but with a roughly 30% reduction in the number of parameters, greatly improving the per-parameter performance, with similar performance to the full-sized comparison CNN models.

5 Image Enhancement with Non-Linear Filters and Improved Preprocessing

In this chapter, Self-Organised Operational Neural Network (Self-ONN) models are extended to Hyperspectral Imaging. Several novel Self-ONN models are proposed where traditional CNN-based Hyperspectral Image (HSI) Super-Resolution (SR) models are enhanced by replacing their convolutional filters with more powerful non-linear Self-Operational filters, offering both performance improvements and overall network parameter savings. Novel improvements are also made to the data preprocessing pipeline to handle HSI data more effectively and improve reconstruction performance. The models are first evaluated on a single HSI, as is commonly done in the literature. Subsequently, performance is then evaluated on the less commonly used, but substantially larger, ICONES dataset [1]. Finally, a large novel dataset acquired in collaboration with Thomas De Kerf, University of Antwerp, Belgium, and David Dunphy, University of Strathclyde, Scotland, is used to evaluate both performance, and additionally the issues with training using synthetic downsampling techniques.

The novel contributions of this chapter are listed as follows:

- Novel Self-ONNs models are proposed for HSI-SR with analysis on the effects that normalisation and residual connections have on these networks
- Novel preprocessing techniques using the standard normal variate and noise removal are proposed for HSI-SR
- Evaluation of how HSI-SR models trained with traditional synthetic downsampling processes fail to generalise to real-world data

Each contribution is presented in its own sub-section along with the results associated to the contribution.

Table 5.1: Small HSI Dataset Information

Dataset	Sensor	Image Dimensions	Channels	Resolution	Location
Pavia University	ROSIS	610×340	103	$1.3m$	Pavia, Italy
Salinas	AVRIS	512×217	204	$3.7m$	Salinas Vally, California
Cuprite	AVRIS	512×614	224	-	Las Vegas, Nevada
Urban	-	307×307	210	$2m$	-

Example images for each dataset can be seen in Figure 5.1.

5.1 Datasets

Several different datasets are used for experiments within this chapter which will be outlined within this subsection. Four commonly used, publicly available, remote sensing datasets are first used which are named Pavia University, Salinas, Cuprite, and Urban. Each of these datasets consist of a single HSI and are commonly used in the HSI-SR literature. Details for the four small HSI remote sensing datasets [110, 111] used in this chapter can be seen in Table 5.1. Bands affected by water absorption or sensor noise have already been removed from each of these datasets.

It is well known that the performance of deep learning models is strongly correlated with the amount of quality data used to train the model [112]. The much larger ICONES remote sensing dataset [1] was therefore also used for experiments in this chapter. This is a large available HSI dataset, consisting of a total of 486 HSIs captured from the NASA Jet Propulsion Laboratory's Airborne Visible InfraRed Imaging Spectrometer (AVIRIS). Each image has spatial dimensions of 300x300 pixels with 224 contiguous spectral channels (bands) ranging between 365 and 2497 nanometers. The dataset consists of nine categories of capture scene which can be seen in Table 5.2 along with the breakdown of the number of images contained within each category.

Certain bands in HSIs are corrupted by water absorption noise and sensor noise. Many single HSI datasets come with metadata indicating these bands or simply have these bands already removed. However, such metadata does not exist for the ICONES dataset and so preprocessing steps had to be performed to detect and remove these bands. To detect

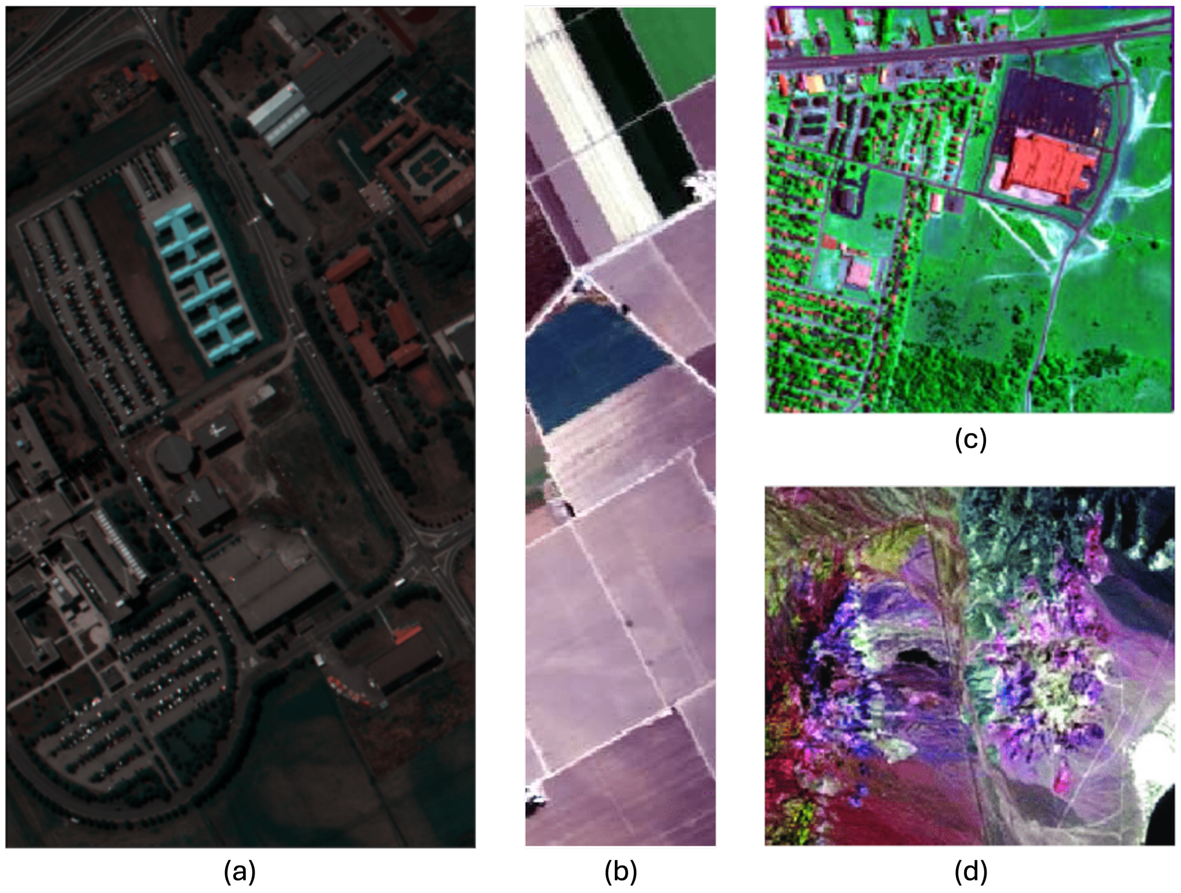

Figure 5.1: Small HSI Datasets False Colour Images.
(a) Pavia University, (b) Salinas, (c) Urban Datasets and (d) Cuprite.

the noisy bands present in the images, a simplified version of the algorithm proposed in [113] was used to detect the noisy bands. Since the spectral resolution of the ICONES dataset is approximately 9.52 nm, there is little difference in features between adjacent bands. Therefore, a channel was deemed noisy if the MSE between the current channel and either of the adjacent channels was greater than 0.005 since the clean bands would be similar and the noisy bands would be very dissimilar. However, it is worth noting that with a lower spectral resolution, the MSE value would potentially need to be increased, or this approach may not work at, all due to greater differences in features between bands. To ensure that the brighter bands were not disproportionately penalised, the minmax norm was applied to each channel individually before computing the MSE. Example clean and noisy bands can

Table 5.2: ICONES Dataset Information

Category	Images	Images Used
Agriculture	50	38
Cloud	29	28
Desert	54	53
Dense-Urban	73	73
Forest	69	50
Mountain	53	52
Ocean	68	14
Snow	55	55
Wetland	35	27

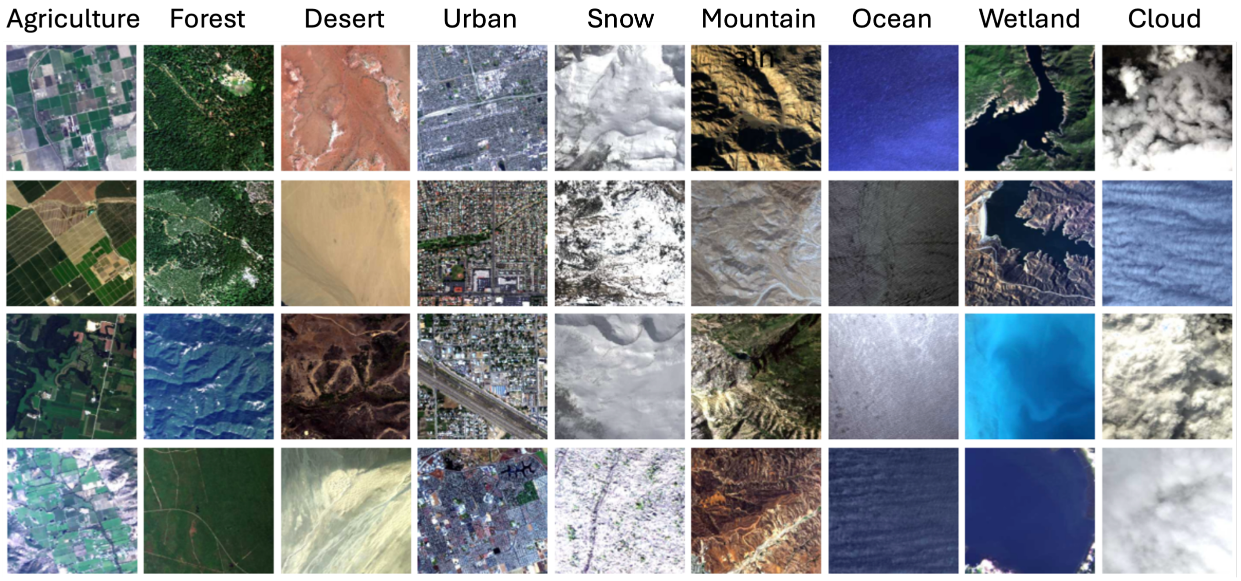

Figure 5.2: ICONES Dataset False-Colour Examples [1].

be seen in Figure 5.3.

From this, bands 1-5, 104-116, 152-172, and 221-224 were removed to ensure that there was minimal noise present in the training data. Therefore, each training image was split into 3 band groups of bands 6-103, 117-151, and 173-220 and random 100x100 pixel patches with 32 bands were extracted. This ensures that training patches are selected with contiguous bands. For validation and testing images, patches were extracted at all 9 100×100 evenly spaced spatial locations within the 300×300 image with contiguous 32 band groupings at bands 7-39, 39-71, 71-103, 118-150, and 179-211 to ensure that validation and test patches

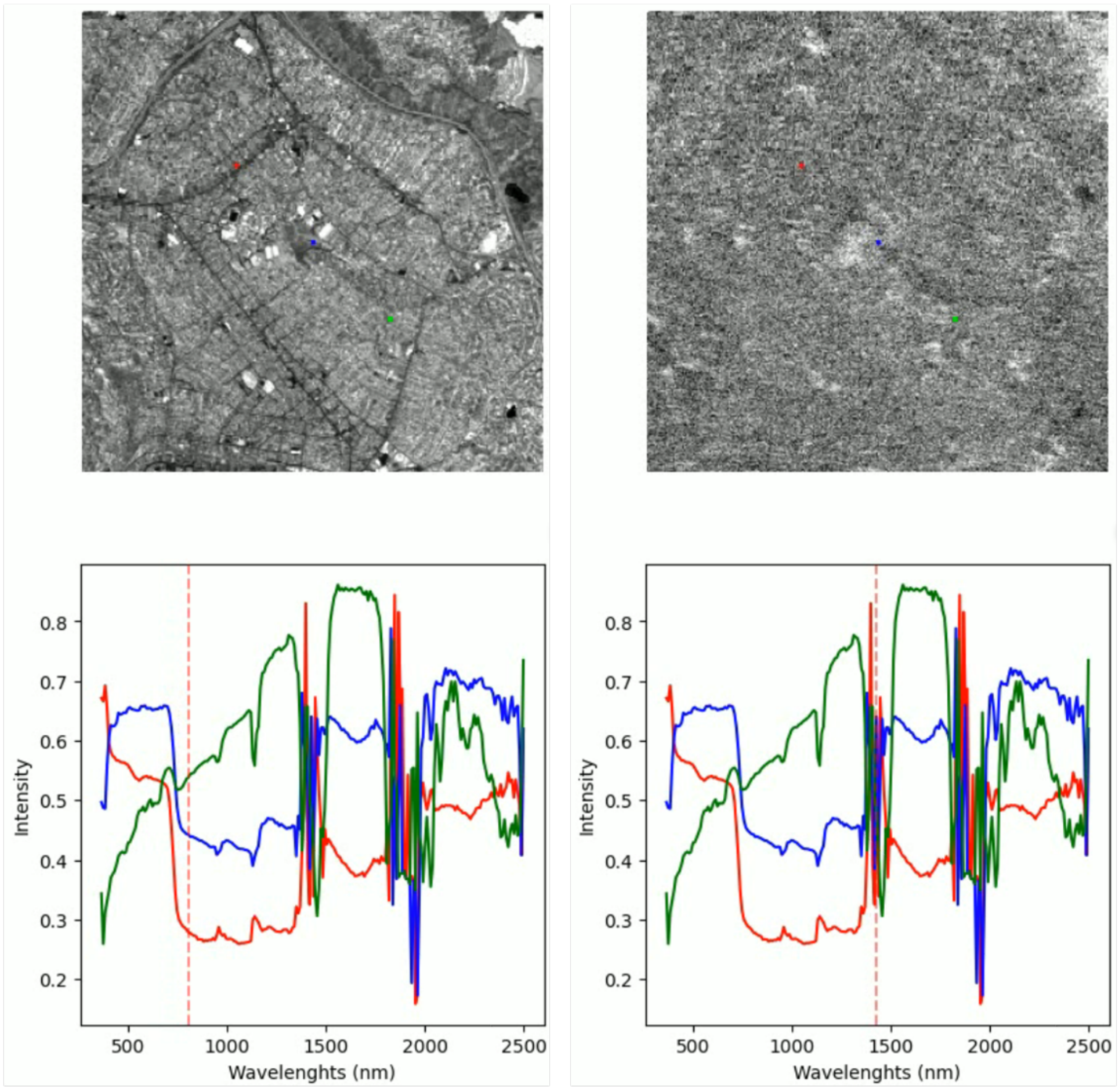

Figure 5.3: Clean and Noisy Band Examples.
Left: Clean Band Example at 802nm. Right: Noisy Band Example at 1461nm. Spectra at the 3 coloured pixel locations are shown on the bottom with the vertical red dotten line representing the wavelength of the band of the top image.

remained consistent. Therefore, each validation and testing image produced 45 patches. The spatial dimensions of 100×100 were selected to ensure maximum data usage for validation and testing by obtaining patches covering the entire image. The band dimension of 32 was also selected to maximise the data usage for validation and testing by allowing for as many

Table 5.3: Indian Pines Dataset Information

Dataset	Sensor	Image Dimensions	Channels	Resolution	Location
Indian Pines	AVRIS	145×145	200	$20m$	Indiana, USA

consistent large chunks of contiguous bands to be used within each noise-free band grouping. Furthermore, the patch size of $100 \times 100 \times 32$ kept patch sizes consistent to allow for batching and were also small enough to avoid GPU memory issues during training, particularly with the 3D models used for experiments. The images heavily corrupted by noise were removed, which was defined as images with more than 40 detected noisy bands. Of the remaining files, 70% of the images in each category were used for training, 15% for validation, and 15% for testing.

In addition, the Indian Pines [114] dataset is used in this chapter for a small target detection experiment. Details of the dataset can be found in Table 5.3 and a true colour image of the dataset alongside the materials ground truth can be seen in Figure 5.4.

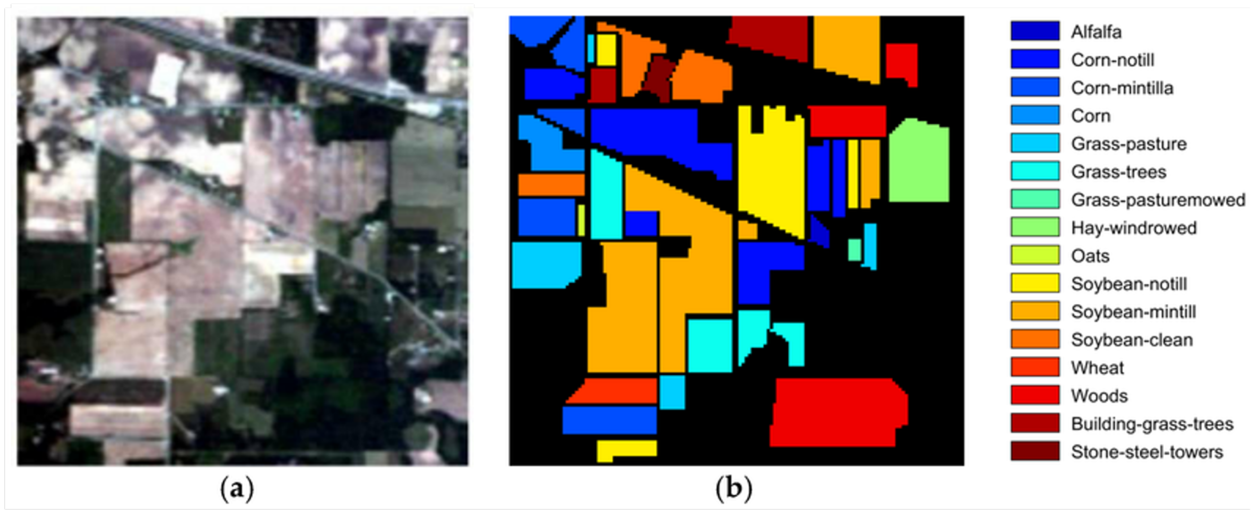

Figure 5.4: Indian Pines Dataset
a) True colour composite image. b) Materials Ground Truth.

All mentioned remote sensing datasets are unpaired, meaning that every LR pair to each HSI within the datasets must be generated for training. Gaussian downsampling was used to generate LR image pairs from the given high-resolution targets in all experiments, which is defined in Eq. (5.1):

$$I_{LR} = (I_{HR} * k) \downarrow_s + n \tag{5.1}$$

where $k \in R^2$ is a 2D degradation kernel, * is a spatial convolution, $\downarrow_s$ is a decimation operation with a stride s, and n is a noise term. No noise was added in any of the experiments, so the parameter n is ignored.

However, downsampling images to synthetically create LR pairs imposes many assumptions on the true low- to high-resolution image relationship, which is known to be detrimental to performance [115, 116]. To address this issue, a novel dataset was acquired in collaboration with Thomas De Kerf and David Dunphy. This dataset consists of true pairs of low- and high-resolution HSIs acquired by two different methods. The first method was acquired by capturing two images of the same scene with the same camera but with different magnification lenses. The data was thus named the *lens data*. The second method involved capturing two images of the same scene with two different sensors, which was named the *sensor data*. Details for each type of paired data can be seen in Table 5.4 with examples of the Lens Dataset shown in Figure 5.5 and examples of the Sensor Dataset shown in Figure 5.6.

Table 5.4: Paired HSI Dataset Information

Name	Sensor(s)	Magnification	Scenes
Lens	Specim FX17	4x	31
Sensor	Headwall Micro-Hyperspec VNIR-E & Hamamatsu C8484-05 CCD	4x	50

For the Lens dataset, HR images (4× magnification) were captured with a 12° lens at 3172 × 640 pixels (0.104 pixel), while LR images used a 38° lens at 793×160 pixels (0.416 mm/pixel).

5.2 Operational Neural Networks for Hyperspectral Single-Image Super-Resolution

In the literature, HSI-SR models are typically trained on a dataset comprised of a single HSI. Most recent methods propose deep networks [117, 77, 72]. However, deep networks tend to

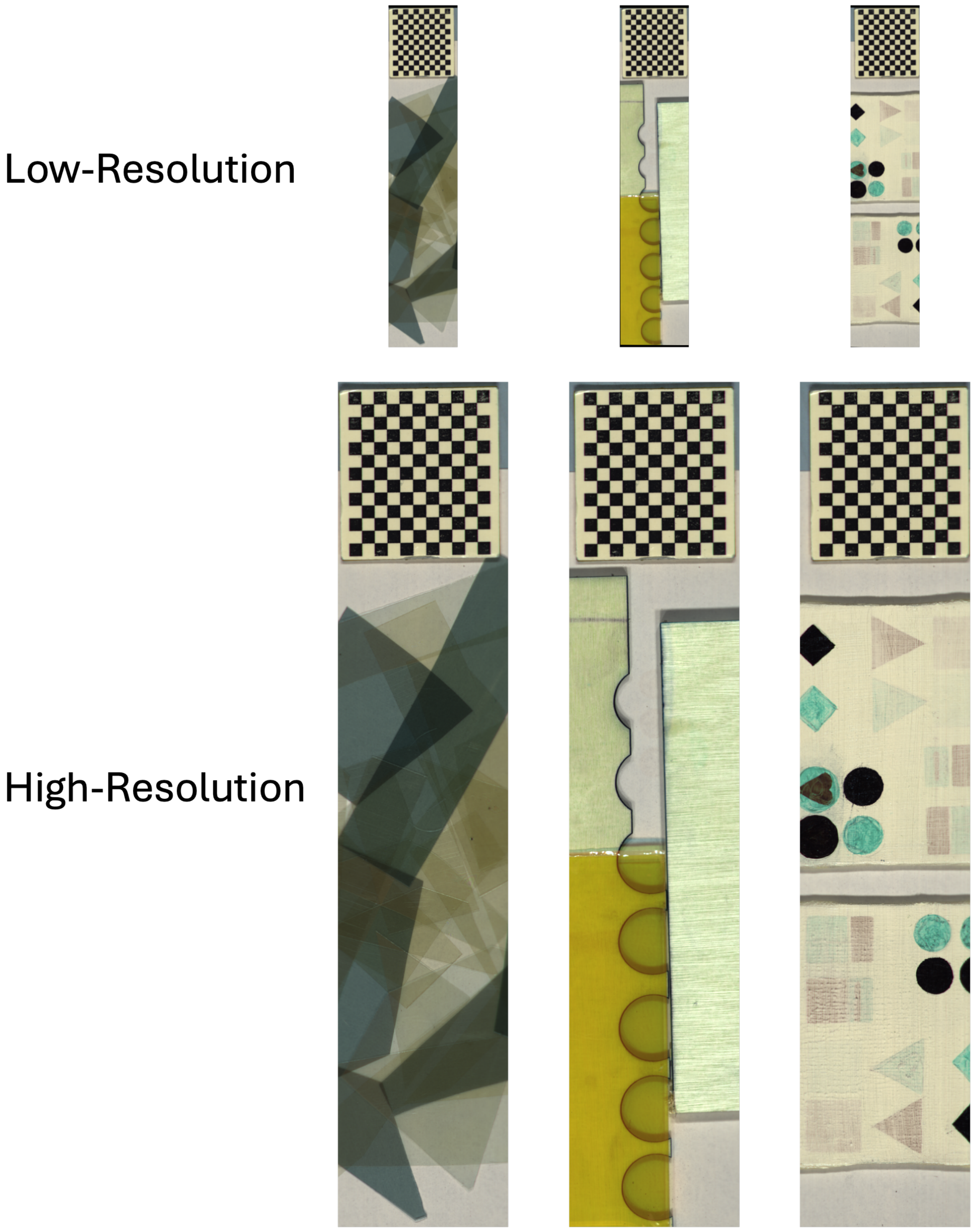

Figure 5.5: Lens Dataset Examples.
Top image are scenes captured with the LR lens and bottom images are the paired scenes captured with the HR lens.

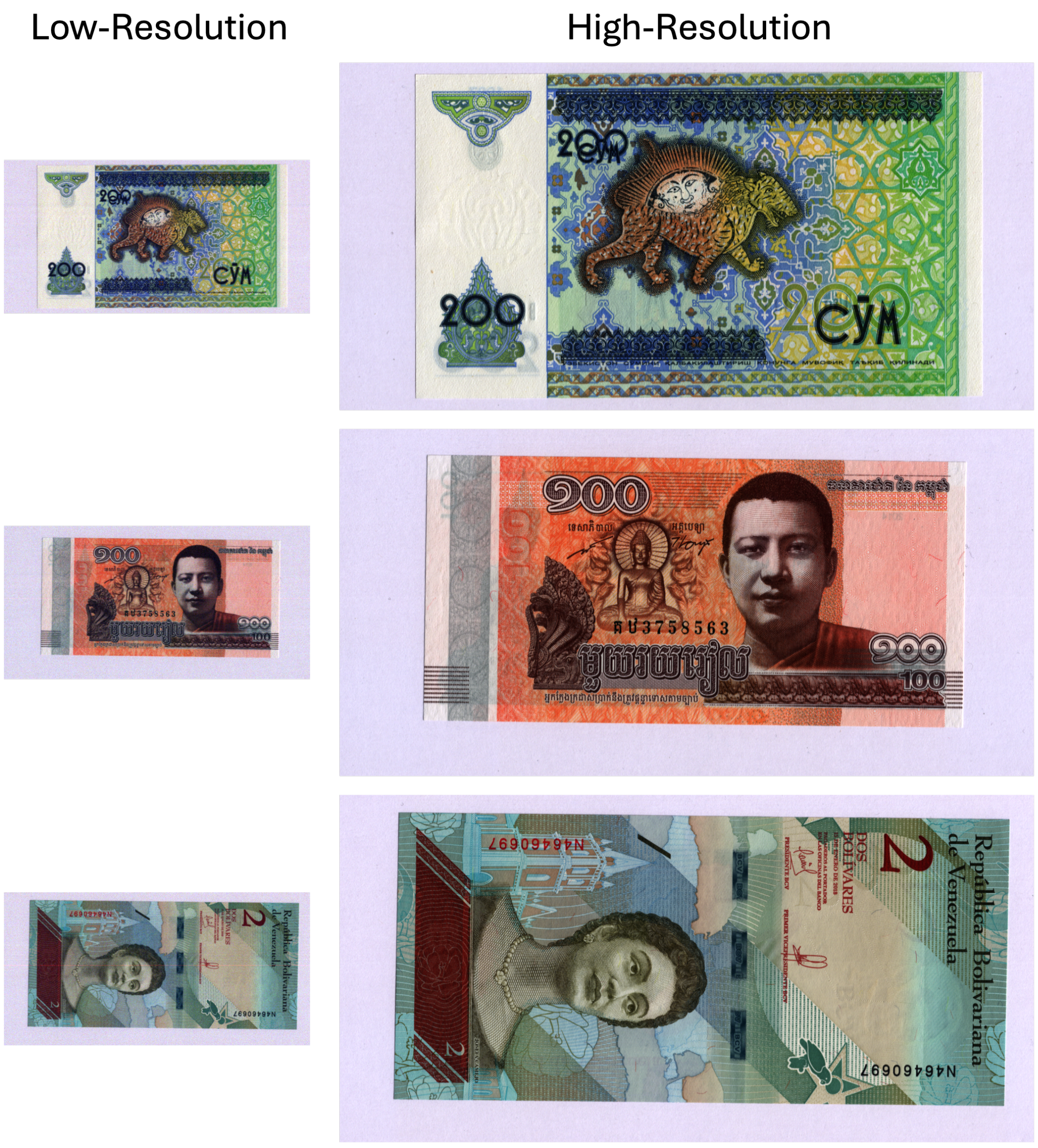

Figure 5.6: Sensor Dataset Examples.
Left images are bank note images captured with the LR sensor and right images are the paired bank note images captured with the HR sensor.

be prone to overfitting on small datasets, which limits their performance and suitability in such a scenario. In this section, an alternative approach is proposed where very shallow models are employed, which use the recently proposed Self-ONN [82] non-linear layers to improve performance over traditional CNN models even with a reduced number of network parameters. In this chapter, the popular SR network, SRCNN [16], was extended for use on HSIs. Then a Self-ONN model based on this architecture is proposed by replacing the convolutional layers with operational layers. Furthermore, another Self-ONN variant of this model with a reduced number of filters is proposed to demonstrate the non-linear capacity of operational layers over convolutional layers. Experiments are conducted on the small Pavia University, Cuprite, Salinas, and Urban remote sensing datasets [111, 110]. Furthermore, the effects residual connections and various normalisation types have on Self-ONN performance are investigated, as this has not been previously explored.

5.2.1 Methodolody

A novel extension of the SRCNN [16] super-resolution model is proposed for application to HSIs. Specifically, this adaptation modifies the SRCNN architecture to accommodate hyperspectral data by increasing the number of input and output channels from three (as used in RGB images) to match the number of wavelength bands in the target HSI. Additionally, a novel Self-ONN-based variant of this modified SRCNN model, named SRONN, is introduced. SRCNN, illustrated in Figure 5.7, is a relatively compact architecture consisting of three convolutional layers, each followed by a ReLU activation function, except for the output layer, which does not use an activation function. Despite the existence of more advanced models, SRCNN was chosen for its simplicity and widespread adoption. Its straightforward structure enables a clear and controlled comparison between CNN and Self-ONN models, ensuring that any performance improvements can be attributed specifically to the non-linear filters of the Self-ONN rather than other auxiliary network components. Furthermore, this architecture provides a suitable foundation for evaluating the impact of additional components, such as

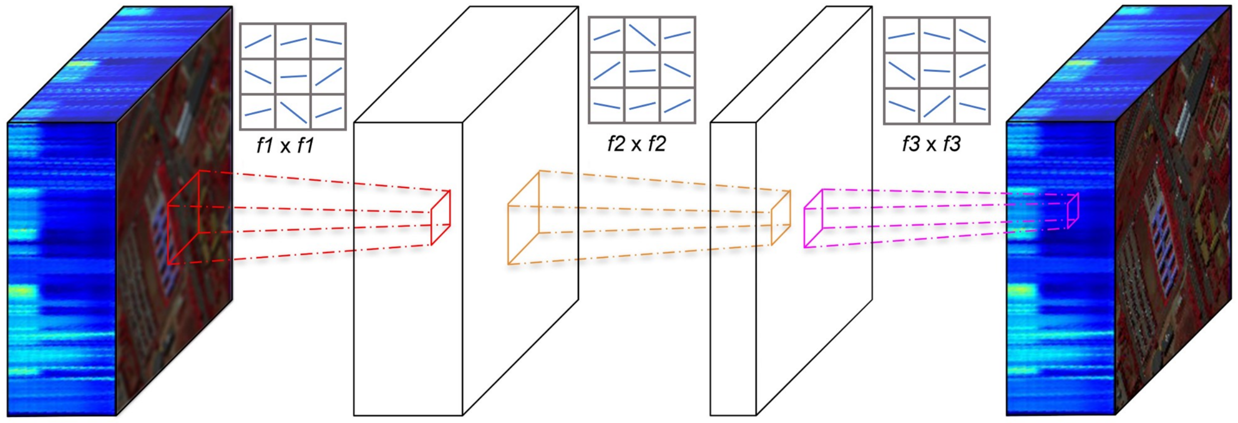

Figure 5.7: SRCNN Architecture.
SRCNN model representation consisting of 3 convolutional layers with filter sizes f1 x f1, f2 x f2, and f3 x f3.

residual connections and normalisation layers, within Self-ONN models. The shallow nature of SRCNN also reduces the risk of overfitting, which is particularly advantageous when working with small single HSI datasets.

The proposed SRONN model retains the same structural configuration as SRCNN, as shown in Figure 5.8, but replaces the convolutional layers with Self-ONN layers. A key characteristic of Self-ONNs is the need to restrict the data passed between layers within the range of -1 to 1 to prevent exponentially large values, a consequence of the inherent non-linearity of the model. To achieve this, SRONN employs Hyperbolic Tangent (Tanh) activation functions after the first and second operational layers instead of the ReLU activations used in SRCNN. The Tanh function, defined in Eq. (2.7), naturally bounds its output between -1 and 1, making it an ideal choice to maintain numerical stability within the SRONN architecture.

Each model was evaluated on four different HSI datasets: Pavia University; Salinas; Cuprite; Urban. Details for each dataset [111, 110] can be seen in Table 5.1.

Eq. (5.1) was used to generate the LR pairs to each of the training patches with $2\times$ subsampling, $\downarrow_s$, and a Gaussian blur with a sigma value of 0.8943 for k as is done in [118]. No noise was added, so the parameter n is ignored. Each generated LR tile was then bilinearly

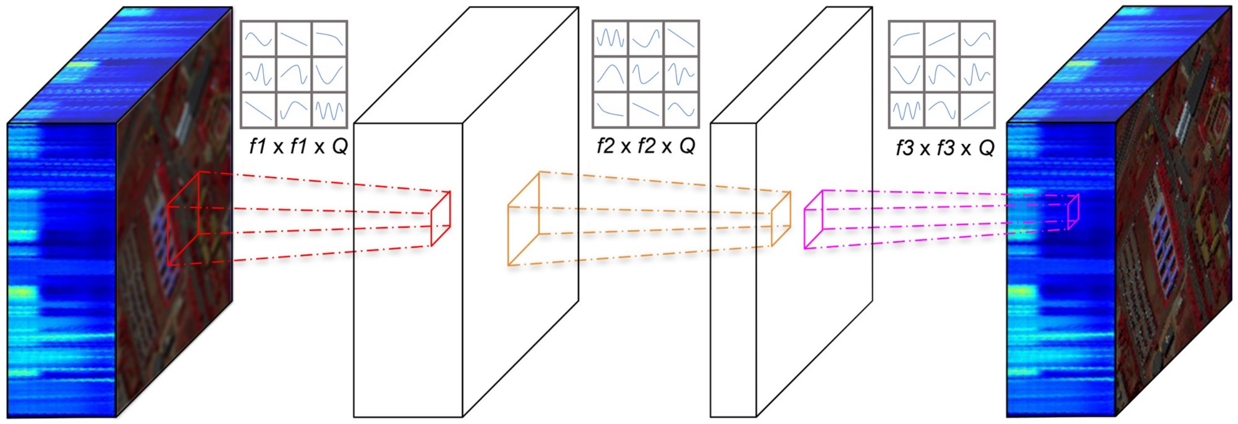

Figure 5.8: SRONN Architecture.
SRONN model representation consisting of 3 self-operational layers with filter sizes (f1 x f1 x Q), (f2 x f2 x Q), and (f3 x f3 x Q). Note, each filter element is a learnable non-linear function, enhancing its theoretical learning capacity over a standard CNN where each filter element is a learnable linear function.

interpolated back to the size of the original tile so the model could perform Super-Resolution by recovering the information at the desired output resolution. The model would then be trained with the LR tile as input and the original HR tile as the target. A scale factor of 2x was selected as the datasets being used are very small in size, making it infeasible to go beyond this scale factor. Each dataset was preprocessed with min–max normalisation and then divided into 64×64 pixel tiles, maintaining the entire wavelength spectrum. 70% of the tiles were utilised for training, 15% for validation and 15% were reserved for testing.

Each model was trained for 50000 epochs to guarantee network convergence, and the weights from the epoch that produced the highest SSIM validation score were used for testing. The Adam optimiser [30] with default parameters except for the learning rate was selected. Each model was initially trained with a learning rate of 10^-4, which was decreased by a factor of 10 at epochs 5000 and 40000. Two following runs were then completed where the starting learning rate and the epoch milestones — where the learning rate was decreased by a factor of 10 — were manually adjusted in an attempt to improve the performance. The MSE loss function was selected as this is a simple and common loss function for SR. The weights of the models were initialised with a normal distribution with a gain of 0.02. For all experiments, the entire training dataset was forward propagated through the model at once, so there was no need to adjust the batch size.

5.2.2 Parametric Analysis

Self-ONNs gain their additional non-linear complexity through the use of MacLaurin series expansions as shown in Eq. (2.10).

In practice, the 0th term in the expansion is the bias. Therefore, the 0th term can be disregarded from the filter approximation. The order of the polynomial should be finite in practice so the number of terms is supplied to the network by a parameter Q. This makes the expansion for an ONN as follows:

$$f(x) = \sum_{n=1}^{Q} \frac{f^{(n)}(0)}{n!} x^n \tag{5.2}$$

Note that when the Q value is 1, it is the exact equivalent of a standard convolutional layer. Higher Q values yield more accurate function approximations but at the cost of additional parameters as the Q value directly equates to the multiplication in parameters over a standard convolutional filter. The number of parameters in the convolutional layers of a CNN can be calculated using the following equation:

$$\#\text{ parameters} = \sum_{l=0}^{L-1} (n_l \times m_l \times f_l + 1) \times f_{l+1} \tag{2.13}$$

where L is the number of layers, n_l, m_l is the number of rows and columns in the convolutional filters at layer l, f is the number of filters and the constant 1 accounts for the bias for each filter. Note, that on the first layer, i.e. $l = 0$, the number of filters from the previous layer ($l-1$) is given by the number of channels of the input image. To compute the number of parameters of a Self-ONN, the number of filters of the previous layer is simply multiplied by Q in Eq. (2.13) to give:

$$\#\text{ parameters} = \sum_{l=0}^{L-1} (n_l \times m_l \times f_l \times Q + 1) \times f_{l+1} \tag{2.14}$$

The proposed SRONN model will therefore have approximately Q times more parameters than the SRCNN model. To ensure a fair comparison between CNN and Self-ONN, a low Q value was selected. The minimum Q value is 2, as a Q value of 1 is the equivalent of a CNN. However, a Q value of 2 would only add one non-linear term to Eq. (5.2), limiting the non-linear function approximation capacity. To enhance this capacity, a Q value of 3 was selected in all experiments, which introduces a second non-linear term to Eq. (5.2), significantly improving the non-linear function approximation while still keeping the parameter increase relatively low. It is also worth noting that going much beyond this Q value will likely have diminishing performance returns relative to the parameter increase and may even be detrimental to performance due to the increased training difficulty, especially on small datasets. However, a Q value of 3 still means that each SRONN model has around three times more parameters than its equivalent SRCNN model. For a fair comparison, a Self-ONN model with the same number of layers as SRCNN but with four times fewer filters per layer was proposed. This model has between 26.5% and 28.2% fewer parameters than SRCNN, depending on the required input and output channels of the dataset. This model will be referred to as small SRONN or sSRONN.

To implement a Self-ONN layer in practice a standard convolutional layer can simply be extended by increasing the number of input channels by a factor of Q and passing the input concatenated with the input raised to the power n up to Q. The convolutional layer will then apply its weights to all the MacLaurin series terms and perform the required summation of the terms, providing the non-linear learnable MacLaurin series approximation.

5.2.3 Normalisation and Residual Connections

Due to the recent proposal of Self-ONNs [82], techniques commonly applied to CNNs to improve results have been studied little on Self-ONNs. Various types of normalisation layer are incorporated into the proposed SRONN and sSRONN model, including L1, L2, instance [119], and batch [120] normalisation, to examine the effects these have on Self-ONN performance. The effects of adding a residual connection to the models is also studied, which connects the output of the model directly to the input so that the model learns the residual rather than the direct mapping as performed in [121].

The proposed Self-ONN model with additional normalisation and residual connection configurations is illustrated in Figure 5.9.

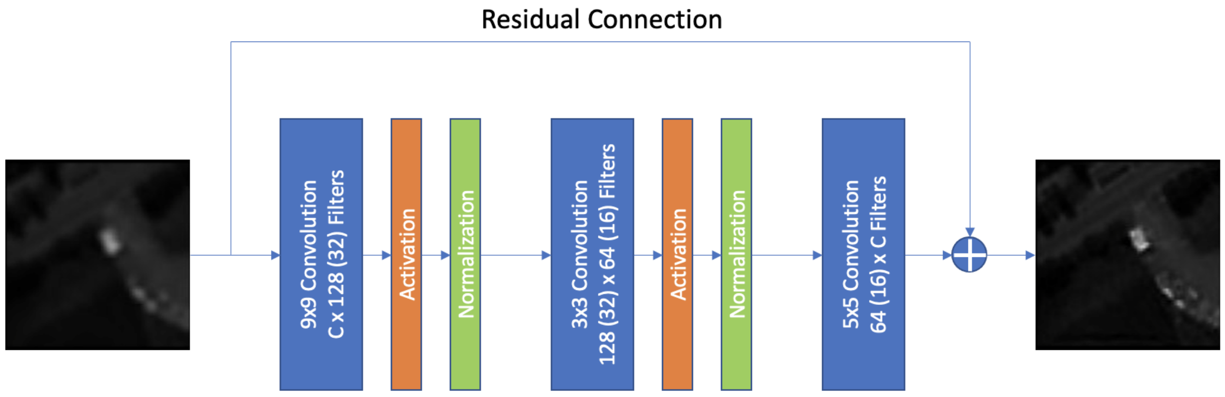

Figure 5.9: General Model Architecture.
C represents the number of channels in the hyperspectral image. Values in brackets represent the number of filters in the compact sSRONN model. SRCNN and SRONN variants have Cx128, 128x64, and 64xC filters in each respective layer. sSRONN variant has Cx32, 32x16, and 16xC filters in each respective layer. The normalisation type depends on the experiment and in some experiments, there is no normalisation, in which case the normalisation layers are skipped. The residual connection is also removed in experiments where it is not applied.

5.2.4 Results

The SRCNN models are first compared against the SRONN and sSRONN models without normalisation for a direct and fair comparison. The results can be seen in Table 5.5 and example outputs on the Pavia University dataset from the models with and without residual connections can be seen in Figures 5.10 and 5.11, respectively. True SR outputs, i.e., where there is no target image and SR is performed on the original data (no downsampling), on the Pavia University dataset can be seen in Figures 5.12 and 5.13.

Experiments using normalisation layers are only conducted on the Self-ONN models, since normalisation has been widely studied in CNNs. The results from adding various normalisation layer types to the Self-ONN models are presented in separate tables for each dataset. Results for the Cuprite dataset are shown in Table 5.6, Pavia University in Table 5.7, Salinas in Table 5.8, and Urban in Table 5.9.

For the three training iterations of each model on each dataset, the results from the best iteration are presented in the tables.

Table 5.5: Results from Standard Models with no Normalisation.

Dataset	Model	Residual	# parameters	lr	lr steps	PSNR ↑	SSIM ↑	SAM ↓
Cuprite	SRCNN	no	2754976	10^{-4}	100k	27.799	**0.9766**	10.136
	SRONN		8264096	10^{-4}	2.5k	**27.882**	0.9743	**10.044**
	sSRONN		**2024720**	10^{-4}	15k	27.863	0.9746	10.061
	SRCNN	yes	2754976	10^{-4}	5k, 40k	27.783	0.9731	10.118
	SRONN		8264096	10^{-4}	2.5k	27.927	0.9774	9.993
	sSRONN		**2024720**	10^{-4}	2.5k	**27.959**	**0.9775**	**9.961**
Pavia University	SRCNN	no	1306727	10^{-4}	5k, 40k	35.396	0.977	4.346
	SRONN		3919591	10^{-4}	2.5k	**35.857**	**0.9775**	**4.209**
	sSRONN		**938503**	10^{-4}	50k	35.693	0.9768	4.606
	SRCNN	yes	1306727	10^{-4}	2.5k, 10k, 30k	35.597	0.9768	4.388
	SRONN		3919591	10^{-4}	5k, 40k	35.914	**0.9783**	4.056
	sSRONN		**938503**	10^{-4}	5k, 40k	**35.926**	0.9782	**4.033**
Salinas	SRCNN	no	2515596	10^{-4}	5k	**44.074**	**0.9943**	**1.462**
	SRONN		7545996	10^{-4}	2.5k	43.941	0.994	1.549
	sSRONN		**1845180**	10^{-4}	5k, 40k	43.558	0.9937	1.622
	SRCNN	yes	2515596	10^{-4}	5k, 40k	44.025	0.9941	1.517
	SRONN		7545996	10^{-4}	10k	44.223	0.9944	1.461
	sSRONN		**1845180**	10^{-4}	4.5k, 30k	**44.286**	**0.9945**	**1.412**
Urban	SRCNN	no	2587410	10^{-4}	5k, 40k	25.231	0.8878	14.811
	SRONN		7761426	10^{-4}	5k, 40k	**25.941**	**0.8935**	**13.94**
	sSRONN		**1899042**	10^{-4}	3k	25.818	0.8912	14.22
	SRCNN	yes	2587410	10^{-5}	5k, 40k	25.872	0.8916	13.958
	SRONN		7761426	10^{-4}	2k	25.892	**0.8999**	**13.613**
	sSRONN		**1899042**	10^{-4}	4k	**26.065**	0.8963	13.681

Bold values are the overall best value for the given metric. Values in italics are the best values for the given model in the absence of a bold value.

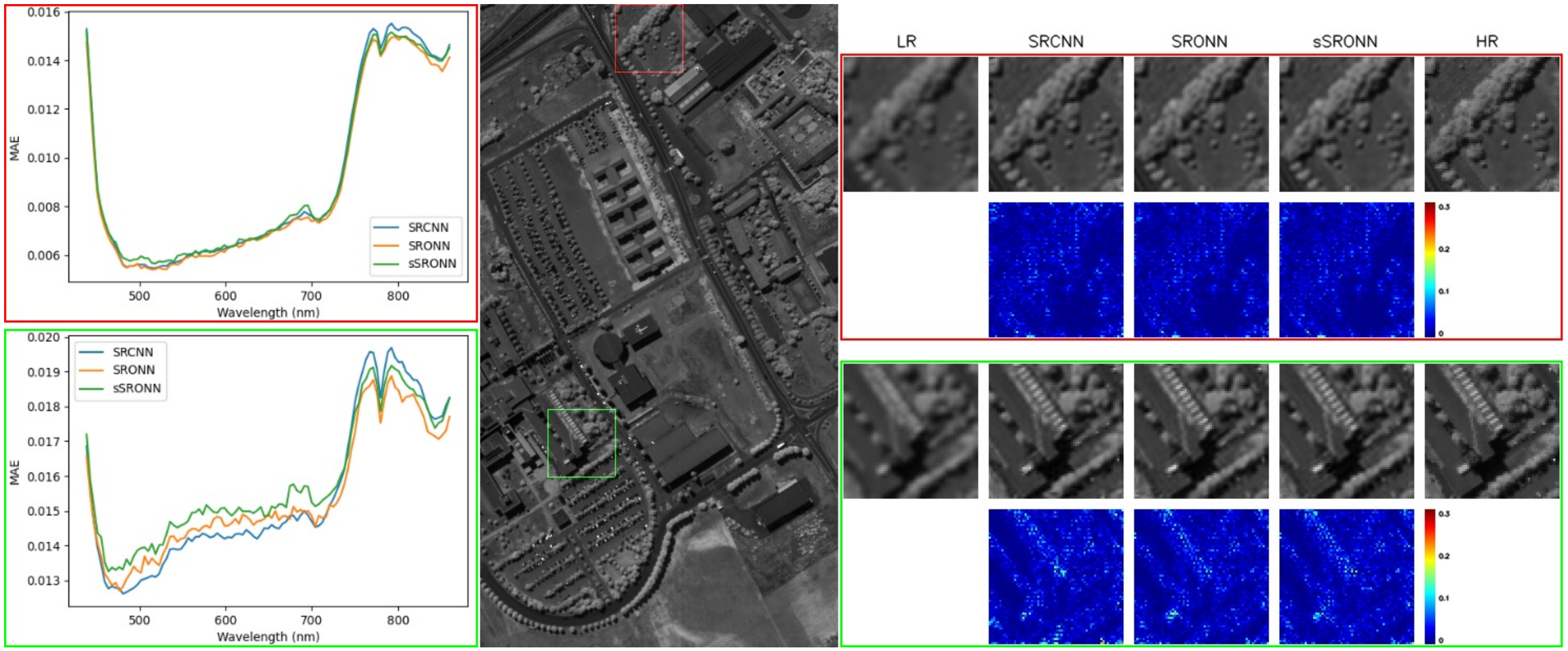

Figure 5.10: No Residual Connection Pavia University Results
Output of models with no residual connection or normalisation on the Pavia University dataset. The mean absolute error between the predicted and true spectra across the patch is shown on the left. Slice 80 of the original HSI is shown in the middle. LR, predictions, HR and the absolute difference between prediction and HR are shown on the right.

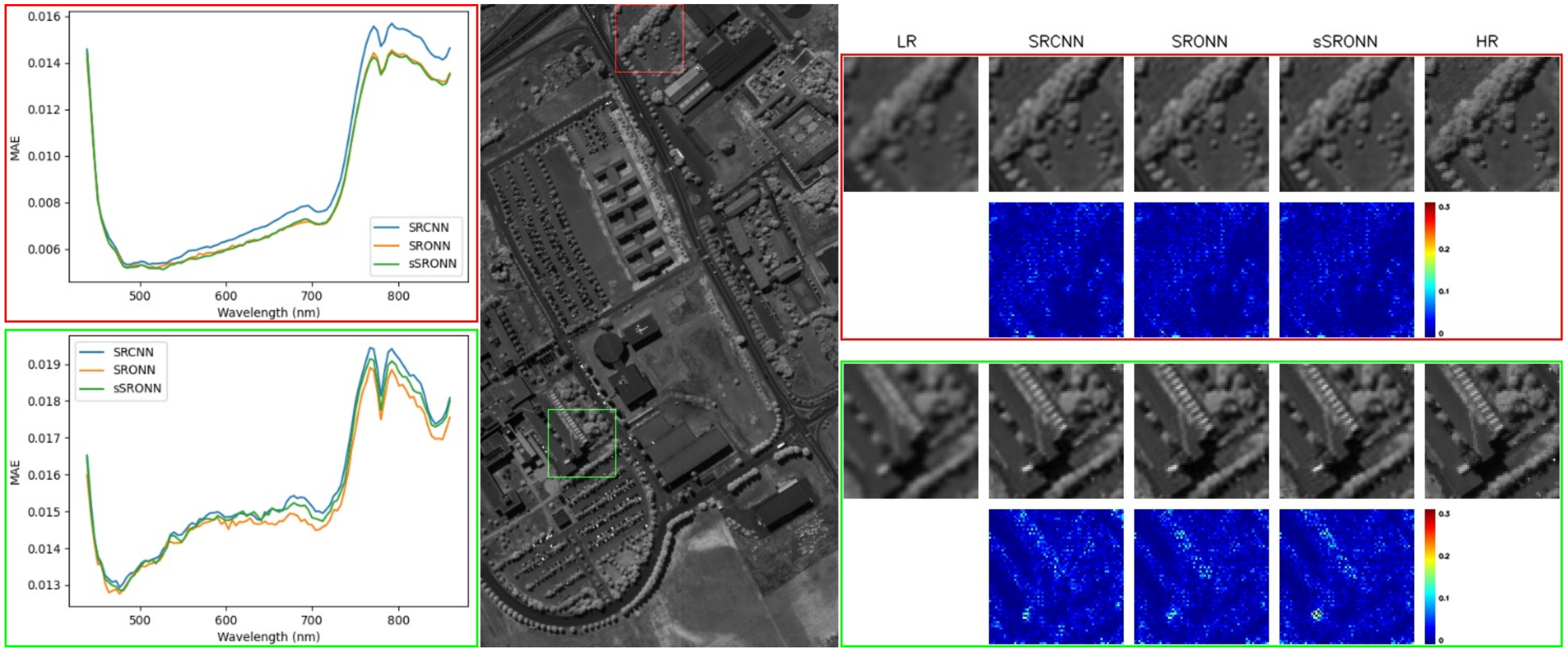

Figure 5.11: Residual Connection Pavia University Results
Output of models with a residual connection on the Pavia University dataset. The mean absolute error between the predicted and true spectra across the patch is shown on the left. Slice 80 of the original HSI is shown in the middle. LR, predictions, HR and the absolute difference between prediction and HR are shown on the right.

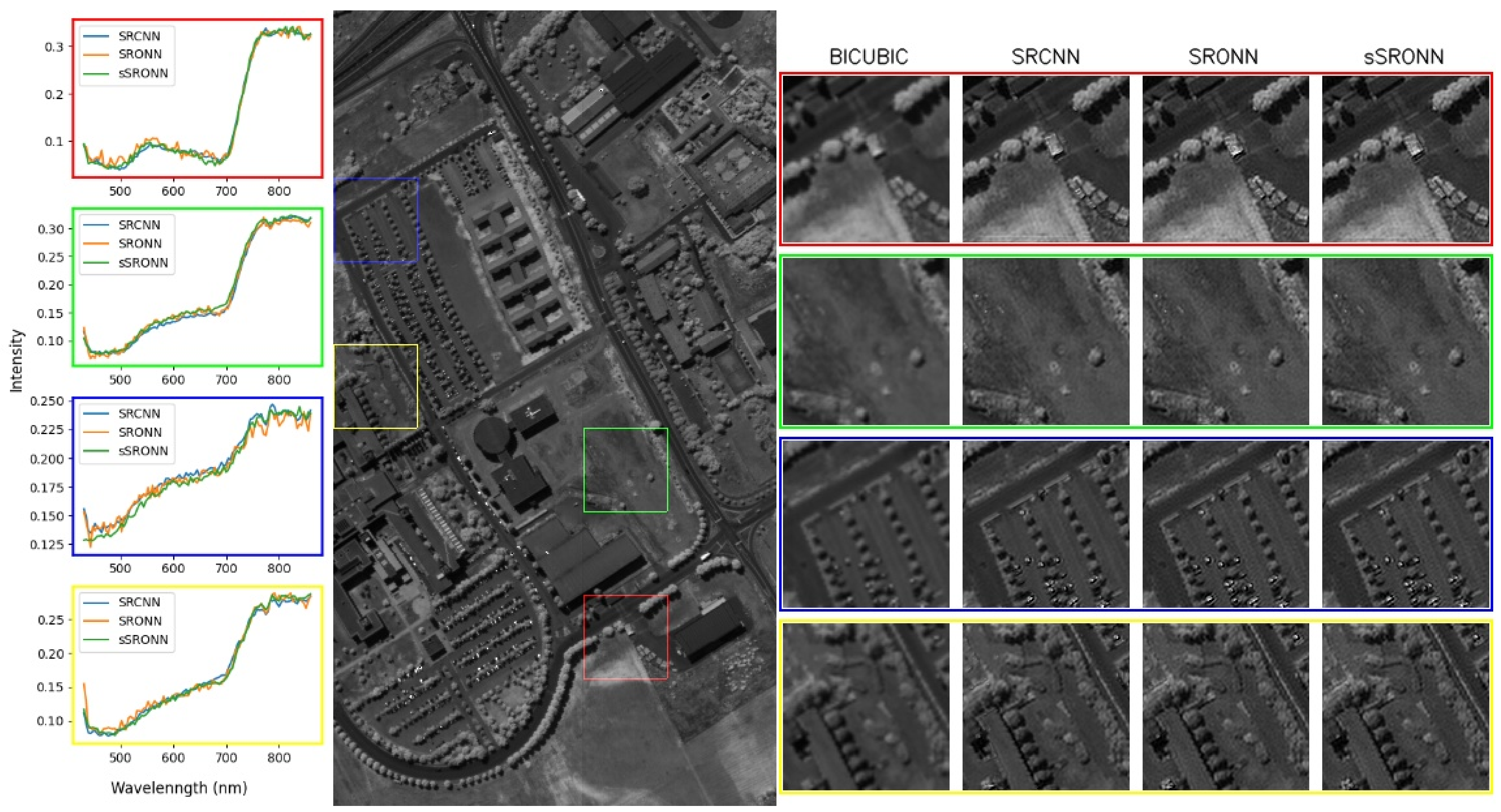

Figure 5.12: No Residual Connection Pavia University True SR Results
True SR output of models with no residual connection on slice 80 of the Pavia University dataset. Spectral plots of the centre pixel of each coloured image patch for each model output are shown on the left. The original HSI is shown in the middle. Test tiles bilinearly interpolated up to 2x their original size and SR results on the interpolated tiles are shown on the right.

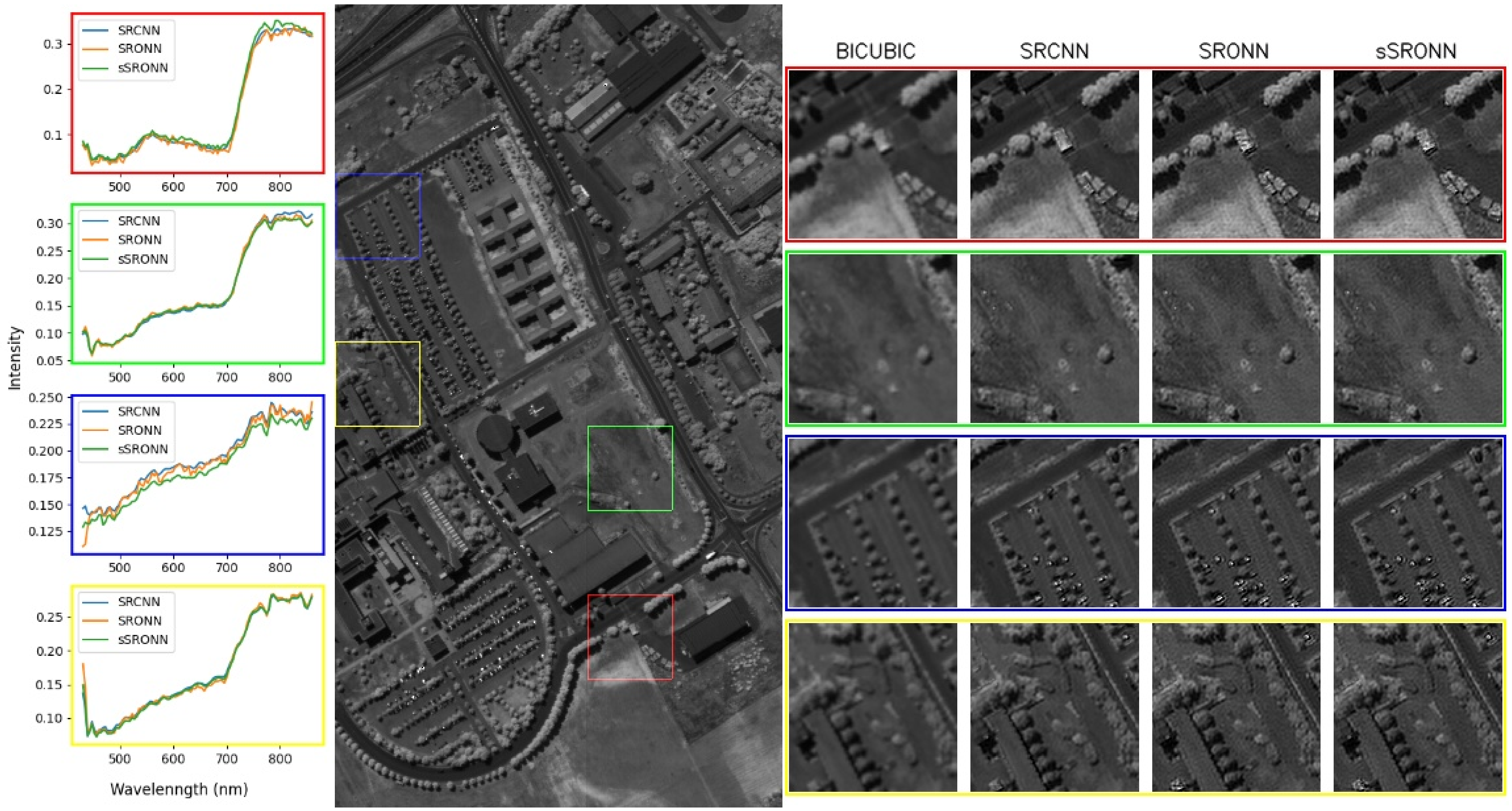

Figure 5.13: Residual Connection Pavia University True SR Results
True SR output of models with a residual connection on slice 80 of the Pavia University dataset. Spectral plots of the centre pixel of each coloured image patch for each model output are shown on the left. The original HSI is shown in the middle. Test tiles bilinearly interpolated up to 2x their original size and SR results on the interpolated tiles are shown on the right.

Table 5.6: Normalisation Results on Cuprite Dataset.

Model	Residual	Normalisation	# parameters	lr	lr steps	PSNR ↑	SSIM ↑	SAM ↓
SRCNN	no	none	2754976	10^{-4}	100k	*27.799*	*0.9766*	10.136
	yes	none	2754976	10^{-4}	5k, 40k	27.783	0.9731	*10.118*
SRONN	no	batch	8264480	10^{-4}	50k	26.998	0.9522	10.959
		instance	8264096	10^{-4}	25k	26.248	0.9296	11.744
		L1	8264096	10^{-4}	50k	27.506	0.971	10.438
		L2	8264096	10^{-4}	10k	27.921	**0.9807**	9.99
		none	8264096	10^{-4}	2.5k	27.882	0.9743	10.044
	yes	batch	8264480	10^{-4}	5k, 40k	26.968	0.9501	11.041
		instance	8264096	10^{-4}	50k	27.324	0.9626	10.662
		L1	8264096	10^{-4}	50k	27.911	0.9761	10.005
		L2	8264096	10^{-4}	5k, 40k	*27.939*	0.9774	*9.98*
		none	8264096	10^{-4}	2.5k	27.927	0.9774	9.993
sSRONN	no	batch	2024816	10^{-4}	50k	27.562	0.9689	10.371
		instance	**2024720**	10^{-4}	50k	26.56	0.9501	11.286
		L1	**2024720**	10^{-4}	50k	26.448	0.9607	11.787
		L2	**2024720**	10^{-4}	50k	27.886	0.9758	10.029
		none	**2024720**	10^{-4}	15k	27.863	0.9746	10.061
	yes	batch	2024816	10^{-4}	50k	27.823	0.9732	10.104
		instance	**2024720**	10^{-4}	50k	27.699	0.9701	10.242
		L1	**2024720**	10^{-4}	5k, 40k	27.372	0.9708	10.628
		L2	**2024720**	10^{-4}	2.5k, 35k	27.956	*0.9775*	**9.96**
		none	**2024720**	10^{-4}	2.5k	**27.959**	*0.9775*	9.961

Bold values are the overall best value for the given metric. Values in italics are the best values for the given model in the absence of a bold value.

Table 5.7: Normalisation Results on Pavia University Dataset.

Model	Residual	Normalisation	# parameters	lr	lr steps	PSNR ↑	SSIM ↑	SAM ↓
SRCNN	no	none	1306727	10^{-4}	5k, 40k	35.396	0.977	4.346
	yes	none	1306727	10^{-4}	2.5k, 10k, 30k	35.597	0.9768	4.388
SRONN	no	batch	3919975	10^{-4}	10k	34.103	0.965	6.013
		instance	3919591	10^{-4}	20k	27.385	0.8828	11.12
		L1	3919591	10^{-4}	50k	34.475	0.9713	4.956
		L2	3919591	10^{-4}	5k	35.16	0.9756	4.495
		none	3919591	10^{-4}	2.5k	35.857	0.9775	4.209
	yes	batch	3919975	10^{-4}	10k	34.705	0.9688	5.242
		instance	3919591	10^{-4}	50k	32.456	0.95	6.277
		L1	3919591	10^{-4}	50k	35.828	0.9775	4.288
		L2	3919591	10^{-4}	10k, 20k, 30k	**36.069**	**0.9785**	*4.055*
		none	3919591	10^{-4}	5k, 40k	35.914	0.9783	4.056
sSRONN	no	batch	938599	10^{-4}	20k	34.441	0.9681	5.323
		instance	**938503**	10^{-4}	50k	27.792	0.8957	11.675
		L1	**938503**	10^{-3}	50k	33.884	0.9655	5.453
		L2	**938503**	10^{-4}	50k	34.934	0.9741	4.708
		none	**938503**	10^{-4}	50k	35.693	0.9768	4.606
	yes	batch	938599	10^{-4}	20k	35.126	0.972	4.878
		instance	**938503**	10^{-4}	50k	32.559	0.9518	6.524
		L1	**938503**	10^{-4}	50k	35.672	0.9756	4.338
		L2	**938503**	10^{-4}	10k, 30k	*36.001*	0.9779	4.118
		none	**938503**	10^{-4}	5k, 40k	35.926	*0.9782*	**4.033**

Bold values are the overall best value for the given metric. Values in italics are the best values for the given model in the absence of a bold value.

Table 5.8: Normalisation Results on Salinas Dataset.

Model	Residual	Normalisation	# parameters	lr	lr steps	PSNR ↑	SSIM ↑	SAM ↓
SRCNN	no	none	2515596	10^{-4}	5k	*44.074*	*0.9943*	*1.462*
	yes	none	2515596	10^{-4}	5k, 40k	44.025	0.9941	1.517
SRONN	no	batch	7546380	10^{-4}	5k, 40k	37.767	0.9754	3.79
		instance	7545996	10^{-4}	50k	23.106	0.7458	18.164
		L1	7545996	10^{-4}	50k	36.285	0.9887	2.523
		L2	7545996	10^{-4}	3.5k	38.077	0.9918	2.082
		none	7545996	10^{-4}	2.5k	43.941	0.994	1.549
	yes	batch	7546380	10^{-3}	50k	42.656	0.9918	1.632
		instance	7545996	10^{-3}	50k	32.233	0.9529	7.342
		L1	7545996	10^{-4}	50k	43.923	0.9937	1.422
		L2	7545996	10^{-4}	5k, 40k	44.12	0.9943	**1.4**
		none	7545996	10^{-4}	10k	*44.223*	*0.9944*	1.461
sSRONN	no	batch	1845276	10^{-4}	20k	41.029	0.9879	2.455
		instance	**1845180**	10^{-4}	30k	26.377	0.8708	11.622
		L1	**1845180**	10^{-3}	50k	34.107	0.9801	3.147
		L2	**1845180**	10^{-4}	50k	37.75	0.9913	2.287
		none	**1845180**	10^{-4}	5k, 40k	43.558	0.9937	1.622
	yes	batch	1845276	10^{-4}	50k	42.429	0.991	1.918
		instance	**1845180**	10^{-3}	20k	39.24	0.9843	2.532
		L1	**1845180**	10^{-4}	50k	43.73	0.9935	1.446
		L2	**1845180**	10^{-4}	5k	44.039	0.9943	1.42
		none	**1845180**	10^{-4}	4.5k, 30k	**44.286**	**0.9945**	*1.412*

Bold values are the overall best value for the given metric. Values in italics are the best values for the given model in the absence of a bold value.

Table 5.9: Normalisation Results on Urban Dataset.

Model	Residual	Normalisation	# parameters	lr	lr steps	PSNR ↑	SSIM ↑	SAM ↓
SRCNN	no	none	2587410	10^{-4}	5k, 40k	25.231	0.8878	14.811
	yes	none	2587410	10^{-5}	5k, 40k	*25.872*	*0.8916*	*13.958*
SRONN	no	batch	7761810	10^{-4}	5k, 40k	22.853	0.7566	19.631
		instance	7761426	10^{-4}	5k, 40k	20.775	0.6761	22.462
		L1	7761426	10^{-4}	50k	25.082	0.8675	15.33
		L2	7761426	10^{-4}	50k	25.48	0.8905	14.332
		none	7761426	10^{-4}	5k, 40k	25.941	0.8935	13.94
	yes	batch	7761810	10^{-6}	50k	24.558	0.8345	15.995
		instance	7761426	10^{-4}	50k	23.953	0.8437	16.478
		L1	7761426	10^{-4}	50k	26.09	0.8959	13.515
		L2	7761426	10^{-4}	5k, 40k	**26.116**	**0.9023**	**13.42**
		none	7761426	10^{-4}	2k	25.892	0.8999	13.613
sSRONN	no	batch	1899138	10^{-4}	5k, 40k	23.38	0.8022	18.544
		instance	**1899042**	10^{-5}	5k, 40k	19.992	0.6549	22.723
		L1	**1899042**	10^{-4}	50k	24.352	0.8338	16.279
		L2	**1899042**	10^{-5}	50k	25.4	0.8809	14.812
		none	**1899042**	10^{-4}	3k	25.818	0.8912	14.22
	yes	batch	1899138	10^{-4}	5k, 40k	24.53	0.8345	16.218
		instance	**1899042**	10^{-4}	5k, 40k	24.75	0.8434	15.372
		L1	**1899042**	10^{-4}	50k	25.918	0.8895	13.783
		L2	**1899042**	10^{-5}	50k	26.019	*0.8964*	13.752
		none	**1899042**	10^{-4}	4k	*26.065*	0.8963	*13.681*

Bold values are the overall best value for the given metric. Values in italics are the best values for the given model in the absence of a bold value.

5.2.5 Discussion

The results from Table 5.5 reveal that the base SRONN models without a residual connection generally offer a slight improvement over the SRCNN model that also lacks a residual connection. However, an exception to this trend occurs specifically in the Salinas dataset, where the SRCNN model without a residual connection outperformed the corresponding SRONN models across all metrics. Note that this improved performance is confined only to the Salinas dataset and is not representative of the overall trend observed across the other three experimental datasets. It is hypothesised that this may be due to the Self-ONN models having a more complex search space to navigate and optimise, owing to the non-linear nature of the filters, thus causing more difficulty in converging compared to the simpler SRCNN model.

The results from Table 5.5 also show that adding a residual connection provides significant improvement to both Self-ONN models, resulting in both the SRONN and sSRONN models outperforming the SRCNN models across all metrics on all datasets. The addition of a residual connection improved all metrics across all datasets for both sSRONN and SRONN except for PSNR on the Urban dataset for the SRONN model, where a slight decrease was observed. The residual connection has a lesser impact on the results of SRCNN, only offering improvement in some cases, which is likely due to the model not being complex enough to see any consistent performance improvement from a residual connection. The residual connection performance improvement on the spectra can be very clearly observed in the mean absolute error spectral plots in Figures 5.10 and 5.11. In Figure 5.10, the SRONN model tends to have better spectral reconstruction at the higher wavelengths while the SRCNN model is usually better at the lower wavelengths, but from that plot, it is visually difficult to say which is better overall except that they are both better than the sSRONN model. However, when a residual connection is added, the mean absolute error spectral plots in Figure 5.11 quite conclusively show that both ONN models provide superior spectral reconstruction than the SRCNN model.

Interestingly, the sSRONN model generally saw greater performance improvements from the addition of a residual connection than the SRONN model, which is counterintuitive as the sSRONN optimisation search space is significantly smaller than the search space of the SRONN model. One explanation for this could be that the sSRONN model might be slightly under-parameterised for direct image-to-image mapping. However, it may have sufficient parameters to learn the residual, resulting in a bigger performance improvement when the residual connection is added to the model. The larger SRONN model, which may be well-parameterised for image-to-image mapping but slightly over-parameterised for residual learning, does not see as much of a performance improvement as the smaller sSRONN model.

Since both SRONN and sSRONN outperform SRCNN, this demonstrates the power of the non-linear filters over the standard linear convolutional filters. The non-linear filters provide the operational layer with an enhanced ability to produce sharper edges and thus produce sharper contrast between pixels resulting in a more detailed output image, which is evident in the resulting images shown in Figures 5.11 and 5.13.

The results in Tables 5.6, 5.7, 5.8, and 5.9 show the effects of incorporating normalisation layers into the SRONN and sSRONN models are largely varied and highly dataset dependent. It appears that normalisation has a greater impact on the datasets with larger spatial dimensions. L2 normalisation was found to be the most effective, providing a slight performance boost to the SRONN model across all metrics on the Cuprite, Pavia University and Urban datasets while boosting the SAM on the Salinas dataset. For the sSRONN model, the performance improvement from adding L2 normalisation is less significant, providing only a performance boost to SSIM and SAM on the Cuprite dataset, PSNR on the Pavia University dataset and SSIM on the Urban dataset. No performance improvement was provided by using L2 normalisation over no normalisation on the Salinas dataset. The results show that normalisation is generally more effective when utilised in conjunction with a residual connection. This is likely due to the fact the normalisation layers will normalise the data around a zero mean which makes it more difficult for the models without a residual connection to map the zero mean feature maps to the true mean of the output. However, when a residual connection is introduced, the model learns the residual between the input and the target, which should have a mean near zero. Therefore, normalisation may offer a greater benefit in this scenario as it assists the model in transforming the data to the target mean, rather than moving it away from the target mean. Interestingly, instance normalisation was found to be especially detrimental to all results. This could be because instance normalisation normalises each channel individually which may have an adverse effect on the channel dependencies.

5.3 Data Normalisation Techniques for Large Hyperspectral Image Datasets

It is well known that the performance of deep learning models is strongly correlated with the amount of quality data used to train the model [112]. Despite this, most HSI-SR methods use only a single well-known benchmark HSI for training [77, 72, 76] divided into several patches for training, testing, and validation. In this section of the chapter, SR experiments are conducted on the ICONES dataset [1] which contains hundreds of HSIs - far larger than any other publicly available HSI dataset.

To the best of the author's knowledge, there have been no studies on how best to preprocess HSI data for SR model training, despite HSI data being far more complex and noisy than standard RGB image data [13]. This is potentially due to methods using only a single HSI for SR training not requiring robust preprocessing methods since these datasets are often processed manually and the statistics of a single HSI will remain fairly consistent across patches. However, this is not the case within the ICONES dataset. To address this, novel data normalisation techniques are proposed to better prepare the challenging HSI data and to improve feature representation between images.

5.3.1 Methodology

The ICONES dataset [1] was selected for experiments as this is a large publicly available HSI dataset. Details for how the data were prepared can be found in Section 5.1.

The signal strength in a HSI varies in the spectral domain, which causes many bands to be much lower intensity compared to others, as Figure 5.14 demonstrates. Therefore, passing this data directly to a model for training can be problematic as the model will put more focus on the brighter bands, that will impose more of a penalty on the loss function due to their higher pixel ranges.

The majority of HSI-SR papers do not explicitly state the ways in which they preprocess their data, and therefore in this thesis it is assumed that no preprocessing is applied other than basic minmax normalisation, which is commonly done in deep learning applications for image processing tasks.

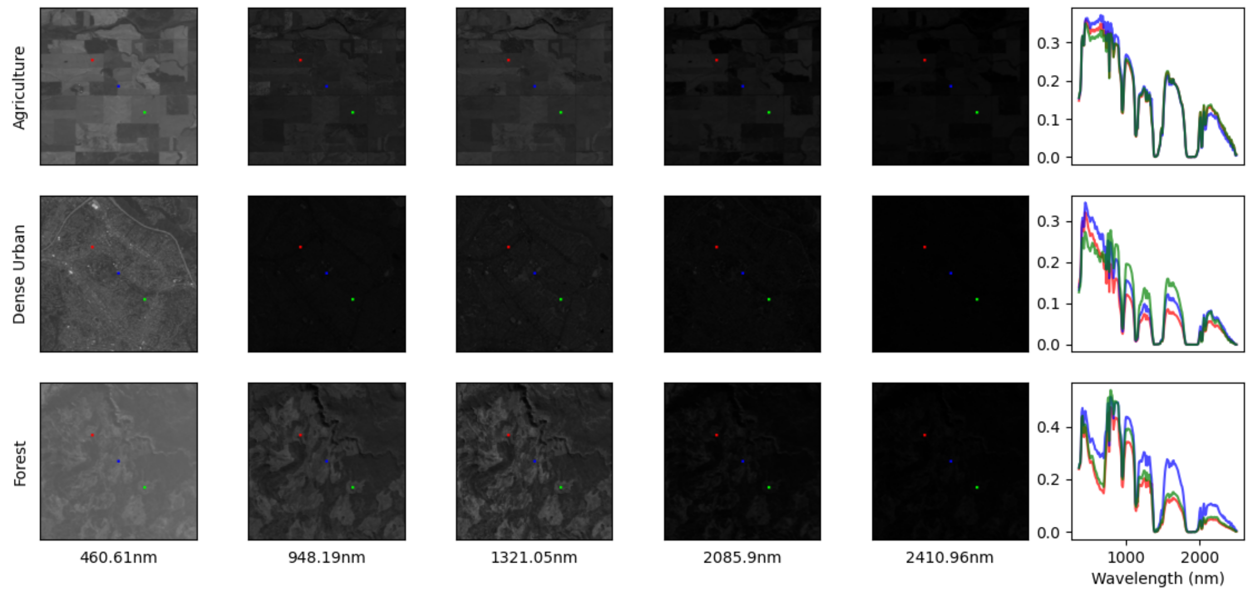

Figure 5.14: ICONES Dataset Examples
Dataset examples at several wavelengths with spectral plots from the highlighted pixel regions. The colour of the spectral plot corresponds to the same colour pixel for the images in the same row.

Therefore, the effects of various preprocessing techniques are examined, and two novel preprocessing techniques based on the standard normal variate are proposed. The preprocessing techniques employed are:

- *Global Minmax*
- *Patch Minmax*
- *Patch Standard Normal Variate (Patch SNV)*
- *Band Minmax*
- *Band Standard Normal Variate (Band SNV)*

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.15}$$

$$X_{norm} = \frac{(X - \mu)}{\sigma} \tag{2.16}$$

Global Minmax is the minimal processing approach, where only minmax normalisation, Eq. (2.15), is applied to each hyperspectral image across the entire image. During training, 100x100x32 patches are extracted from the HSIs to pass to the model. *Patch Minmax* is therefore minmax normalisation applied to the extracted patch instead of to the entire HSI. The *Band Minmax* algorithm is taken from [122], which applies minmax normalisation individually to each band in the extracted patch as defined in Eq. (5.3):

$$X'_{i,j,b} = \frac{X_{i,j,b} - X_{\mathrm{min(b)}}}{X_{\mathrm{max(b)}} - X_{\mathrm{min(b)}}} \tag{5.3}$$

An alternative approach to normalise data is to use the Standard Normal Variate (SNV) as defined in Eq. (2.16). This normalisation technique has seen widespread adoption for various feature normalisation techniques [120, 123, 119, 124] and has been shown to greatly improve performance and convergence times. Two novel normalisation methods based on the SNV are proposed. For the proposed SNV-based preprocessing algorithms, the data are first normalised using Eq. (2.16). This is applied across the patch for *Patch SNV* and for each band in the patch for *Band SNV*. The empirical rule states that 99.7% of the data in a normal distribution are encapsulated within the first three standard deviations from the mean. Therefore, data outside this range are clipped to perform denoising by removing extreme outliers from the data, which helps to prevent any particularly bright pixels from saturating the data. Finally, the data is rescaled back to the range of 0 to 1 so that it is consistent with the other preprocessing techniques. The first proposed SNV-based preprocessing technique which is applied to the extracted patches, named *Patch SNV*, is defined in Eq. 5.4:

$$X' = \text{clip}\left(\frac{(X-\mu)\,\frac{1}{3}\sigma + 1}{2}\right) \tag{5.4}$$

Here, μ and σ are the mean and standard deviation across the patch, respectively, and the clip function is defined in the equation. (5.5):

$$\text{clip}(x) = \begin{cases} a & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ b & \text{if } x > 1 \end{cases} \tag{5.5}$$

Note that without the clip operation, after the data are rescaled, the data would be identical to the data processed by the *Patch Minmax* method. The clip operation is therefore necessary to retain an approximately consistent mean.

The second proposed SNV-based preprocessing technique is applied individually to each band of the extracted patches. This technique is named *Band SNV* and is defined in Eq. (5.6):

$$X'_{i,j,b} = \text{clip}\left(\frac{(X_{i,j,b}-\mu_b)\,\frac{1}{3}\sigma_b + 1}{2}\right) \tag{5.6}$$

Example false-colour image outputs of the data after each preprocessing technique has been applied, along with their respective pixel intensity histograms, can be seen in Figure 5.15. The figure shows that each different preprocessing technique offers different enhancements to the patches. As evidenced by the colourful false colour images they produce, both the band methods have a more even distribution of values across each band, rather than having one particularly bright band dominate. Both SNV approaches produce histograms that are much more similar across different patches, which is hypothesised to improve feature representation between images.

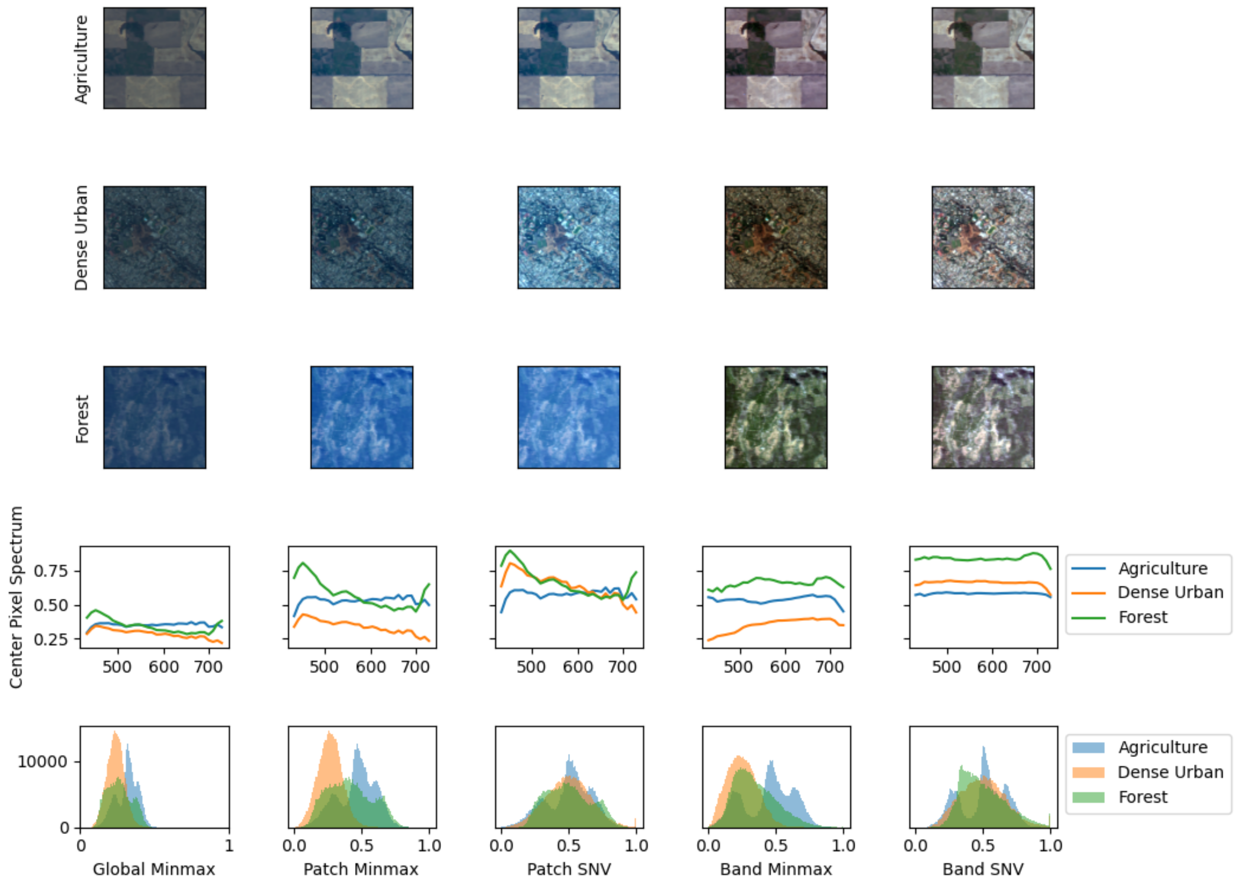

Figure 5.15: ICONES Training Patch Examples
False colour training patch examples with each experimental normalisation technique. Each patch image contains wavelengths: 460.61nm, 604.01nm, 756.98nm for blue, green and red channels respectively.

After the normalisation step, Gaussian downsampling was used to generate the LR pair for the given patch, which is defined in Eq. (5.1):

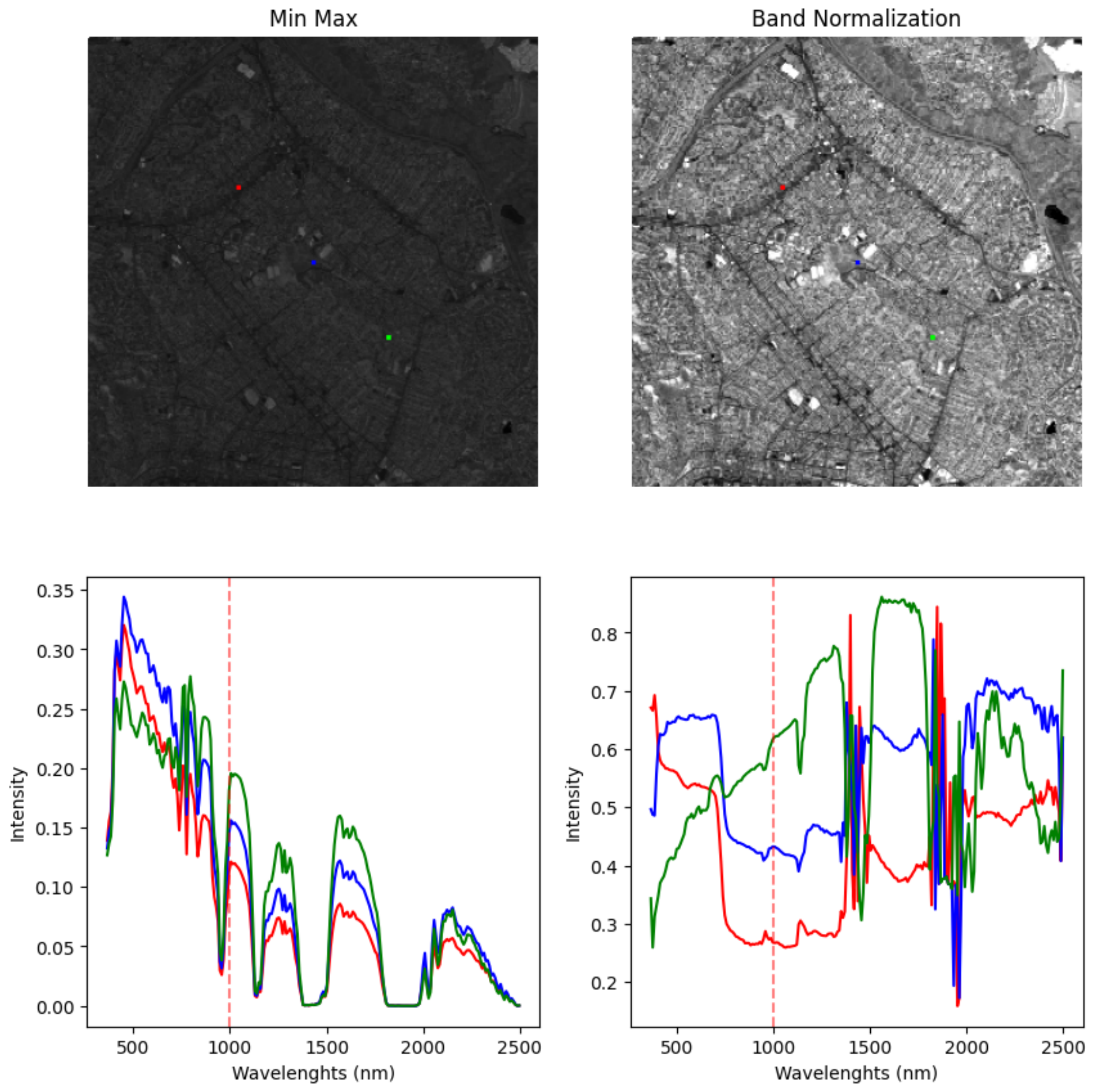

Figure 5.16: Normalisation Examples
Example Image of *Global Minmax* normalisation and *Band SNV* preprocessing at 1000nm.

A σ value of 1.6986 was selected for k as is done in [118] with $4\times$ subsampling for $\downarrow_s$. No noise was added, so the parameter n was ignored.

5.3.2 Experiments

Experiments were conducted on several popular types of deep learning models for hyperspectral single-image SR, incorporating various different architectures and layer types, including 2D and 3D convolution as well as transformers and Self-ONNs, to produce a representative evaluation of the proposed preprocessing techniques. The models selected for experiments were 3DHSRCNN [17], FSRCNN [57], ESSAformer [78] and the GAN model proposed in [72], which will be referred to as BAGAN. The deeper Fast Super-Resolution Convolutional Neural Network (FSRCNN) model was selected instead of SRCNN since the ICONES dataset is far larger than the individual HSI datasets used in the previous section, allowing for use of a deeper model. FSRCNN also performs feature extraction directly on the input LR patch, which is more computationally efficient compared to the SRCNN model, which performs feature extraction on the interpolated LR patch. Building off of the previously proposed SRONN model, another novel Self-ONN model is proposed which is based on the FSRCNN architecture and is named FSRONN. This model was proposed to provide a more complex architecture than the previous SRONN model, which is more suited to the larger ICONES dataset. Furthermore, the proposed model would allow for a direct Self-ONN to CNN comparison with the FSRCNN architecture. To create the FSRONN model, the FSRCNN architecture was taken and all 2D convolution layers were replaced with 2D self-operational layers, and the final deconvolution layer was replaced with a self-operational deconvolution layer, using a q value of 3 for each layer. The PReLU [125] activation functions were replaced with Tanh activation functions - a necessary modification to ensure that the data passed to each self-operational layer are bound between -1 and 1 [82]. Since the FSRCNN architecture does not use any auxiliary network components, any performance gains observed in FSRONN over FSRCNN can be solely attributed to the use of the self-operational filters.

Each model was trained on the ICONES dataset [1] a total of five times - once for each preprocessing method. Separate training parameters specific to each model were used, where the optimiser and loss function were selected from the papers in which the models were originally proposed, while tailoring the learning rate and learning rate decay step specific to the conducted experiments. Each model was trained for 1000 epochs, except for the ESSAformer model, which was trained for 1250 epochs, as it required slightly more time to converge. The training parameters used for each model remain consistent throughout the five training iterations for each preprocessing method. The training parameters used for each model can be seen in Table 5.10.

Table 5.10: Training Parameters

Model	Learning Rate	Learning Rate Reduction Epoch	Optimiser	Loss Function	Batch Size	Augmentation
FSRONN	10^{-3}	500	Adam	MAE	128	False
FSRCNN	10^{-3}	500	Adam	MAE	128	False
ESSAformer	10^{-4}	200	AdamW (5×10^{-3})	MAE	16	False
BAGAN	10^{-3}	500	Adam	BAGAN	64	False
3DHSRCNN	10^{-3}	-	Adam WD (10^{-4})	Charbonnier	64	True

During training, patches containing 100x100 pixels with 32 random contiguous bands were extracted from the original image. This was done for two reasons. First, to select bands only at indices between 6-103, 117-151, and 173-220, so as not to select any of the heavily corrupted bands outside these ranges. Second, to reduce the memory requirements during training. This means that random subsets of the bands were passed to the model on each forward pass. This is standard practice for models using 3D convolution, as the band indices themselves do not matter as long as they are contiguous, since the 3D filter will scan over all bands in the patch, and training with contiguous patches would allow for all bands to be passed to the model during inference. However, this is somewhat unconventional for models using 2D convolution or transformer layers, as the channel indices passed to the model are usually fixed. Nonetheless, the results show that this approach does not seem to have a significant impact on performance, as the 2D convolution and transformer models produce results comparable to the 3D models.

For a fair comparison, the normalisation process was reversed on the model's predicted outputs when computing the evaluation metrics. This means that all outputs are being evaluated in the *Global Minmax* format and will be evaluated with the same target data ranges. If the normalisation process were not reversed, normalisation methods using more distributed data ranges would be penalised more heavily in the evaluation metrics for the same percentage error as a preprocessing technique using a less distributed data range. Note that the normalised input is always passed to the model and the loss is computed using the direct model prediction and the normalised target image. The reversing process is *only* applied to the model outputs on the validation and testing data for the evaluation computation, and of course, the model was still passed the normalised inputs. For the *Global Minmax* case, no reversing process was applied because this represents the minimal preprocessing scenario to which all other normalisation techniques are ultimately reverted to.

To evaluate how a post-processing task may benefit from the proposed preprocessing methods, a simple target detection experiment was also performed on a HSI that was super-resolved using each preprocessing method with its relevant trained model.

5.3.3 Results

The experimental results are presented in Table 5.11. The results are also presented in four box plots to more clearly compare the overall performance across all models between the preprocessing methods for each evaluation metric individually. The box plot of the results can be seen in Figure 5.17. Training and validation plots are also shown in Appendix 7.5.

Target Detection

A small target detection experiment was performed to validate the effectiveness of the HSI-SR models trained using the standard and proposed preprocessing techniques in a real-world application. The Indian Pines [114] dataset, shown in Figure 5.4, was selected as it contains classification labels and is captured with the same type of sensor as the ICONES dataset used for SR training. The Modified Spectral Angle Mapper (MSAM) [126] method was used for classification using the mean of the spectra for each class from the original HSI as the reference spectra. Applying this method directly to the data (without Super-Resolution) yields an accuracy of 44.28%. The dataset is downsampled using the same process to generate the LR image pairs during the SR training. The LR pair was then normalised with each of the preprocessing algorithms used in the SR experiments and then super-resolved using the 3DHSRCNN model for the appropriate preprocessing technique and the normalisation

Table 5.11: HSI-SR Data Normalisation Techniques Results

Model Architecture	Data Preprocessing	SAM ↓	SSIM ↑	PSNR ↑	ERGAS ↓
FSRONN (589,644)	*Band Minmax*	1.6871	0.9153	40.8838	3.0151
	Band SNV	**1.6033**	**0.9162**	**41.3423**	**2.8379**
	Global Minmax	1.9693	0.9126	*40.244*	*3.1355*
	Patch Minmax	1.9447	0.9132	40.7437	3.1104
	Patch SNV	*2.2215*	*0.9112*	40.8811	3.0916
FSRCNN (196,685)	*Band Minmax*	1.792	0.9122	40.6766	3.0927
	Band SNV	**1.7263**	**0.9132**	**41.1015**	**2.9087**
	Global Minmax	2.1716	*0.909*	*39.8293*	*3.2507*
	Patch Minmax	2.1139	0.9099	40.5309	3.2337
	Patch SNV	*2.2494*	0.9101	40.8158	3.1092
ESSAformer (11,198,368)	*Band Minmax*	2.1599	0.9095	40.413	3.2123
	Band SNV	**1.8599**	**0.9143**	**41.179**	**2.8732**
	Global Minmax	*4.455*	*0.8913*	*37.4466*	*4.0431*
	Patch Minmax	3.0317	0.9038	39.6962	3.4386
	Patch SNV	3.1185	0.904	39.9497	3.3514
BAGAN (628,869)	*Band Minmax*	1.6305	0.9156	40.8509	3.0175
	Band SNV	**1.609**	**0.9166**	**41.3888**	**2.8269**
	Global Minmax	*2.675*	*0.9049*	*38.2925*	*3.3909*
	Patch Minmax	1.8557	0.9115	40.2405	3.1368
	Patch SNV	1.8878	0.9141	40.8661	2.9562
3DHSRCNN (110,741)	*Band Minmax*	2.0695	0.9133	40.6961	3.1819
	Band SNV	**1.6**	**0.9182**	**41.3877**	**2.8113**
	Global Minmax	*2.9889*	*0.8994*	*38.5912*	*3.5893*
	Patch Minmax	2.6713	0.9053	39.6063	3.369
	Patch SNV	2.5941	0.9078	39.9237	3.1666

Values in green and bold are the two best values for the given metric for each model.
Values in red and italics are the worst values for the given metric for each model.
Values in parentheses under the model names indicate the total number of network parameters.

process was then reversed. Target detection was then performed on the resulting image and the results are shown in Table 5.12.

5.3.4 Discussion

The results show that the SR models trained on data preprocessed with the proposed *Band SNV* method significantly outperform all other approaches in every metric for each model tested. The *Band Minmax* method produces better results than the remaining methods, suggesting that band-wise normalisation is much more effective for HSI-SR than global or

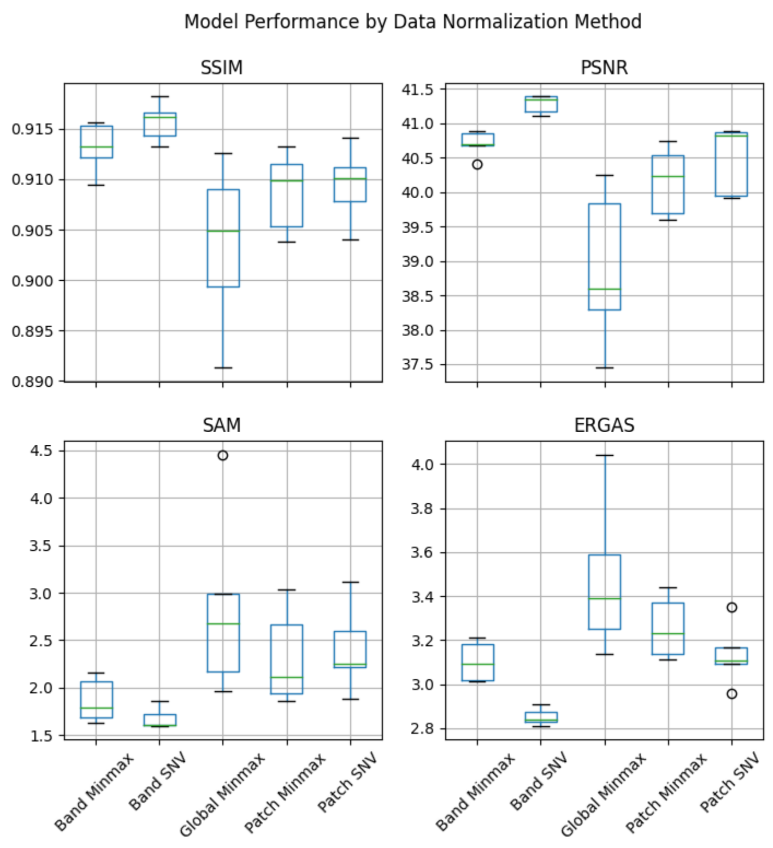

Figure 5.17: Box Plot of Model Results by Preprocessing Method.

patch normalisation. Conversely, the *Global Minmax* approach ranks lowest in performance, a predictable outcome given that this method applies the least processing out of all five methods. Additionally, the *Patch SNV* method generally outperforms the *Patch Minmax* method, indicating that applying a small amount of noise/outlier removal is beneficial to SR performance, since both SNV methods outperform their minmax counterparts.

The efficacy of these preprocessing methods can be attributed, in part, to their ability to provide more uniformly distributed intensity ranges across each image patch. This distribution leads to a more effective implementation of pixel-wise loss penalties, facilitating better model learning, as evidenced by the inverse correlation between the average loss values and the average validation metric values shown in Appendix 7.5. The *Global Minmax* technique, which normalises the entire image to a $[0, 1]$ range before selecting a patch, often results in

Table 5.12: Target Detection Results

Preprocessing Method	Accuracy (%)
Base	*44.28*
Global Minmax	40.03
Patch Minmax	39.21
Patch SNV	40.18
Band Minmax	**41.49**
Band SNV	41.15

Overall pixel classification accuracy from MSAM algorithm on images super-resolved with each different normalisation strategy.

patches with poorly distributed intensity ranges. In contrast, the *Patch Minmax* approach ensures that each patch has a $[0, 1]$ intensity range, although this method does not mitigate the impact of high-intensity outliers that can skew the patch's average intensity. The application of the *Patch SNV* method eliminates such outliers, standardising the average intensity of patches to 0.5 and ensuring a balanced intensity distribution.

However, due to the varying intensity distributions across different bands resulting from the spectral response of the hyperspectral camera, global or patch normalisation can lead to certain bands having narrower intensity ranges. This discrepancy causes the model to disproportionately focus on brighter bands which impose greater loss penalties. The *Band Minmax* normalisation strategy addresses this issue by ensuring that the intensity range for each band is bound to the range $[0, 1]$, allowing the network to focus more evenly on all bands. Furthermore, the *Band SNV* approach not only normalises the band data, but also removes outliers, setting a mean intensity of 0.5 for each band, further improving the uniformity of the intensity distribution. These variations in intensity distribution are illustrated in Figure 5.15.

In addition to improving pixel-wise loss function utilisation, it is also hypothesised that the proposed normalisation techniques simplify the problem space by improving feature representation between images. Making patches share more similar intensity ranges likely makes it easier for the model to extract more similar features. Furthermore, when outlier removal is applied, as is done with the SNV methods, this distribution is less likely to be skewed. For the band-wise methods, it also means that there is less likely to be a large variation in intensities between adjacent bands, making it potentially easier for a 3D CNN model to extract features by not having to adapt to potentially drastic intensity changes.

Furthermore, employing the Standard Normal Variate for normalisation between model layers has been shown to be very effective for deep learning applications in general [120]. The proposed SNV methods, which are somewhat similar to the Layer Norm [123] and the Instance Norm [119] operations, provide similar advantages, such as improved convergence times, training stability and overall performance.

The superior SR performance from the proposed methods also leads to improved accuracy in the target detection post-processing experiment. The proposed band-wise methods offer an accuracy boost of more than 1% compared to the basic global and *Patch Minmax* operations, indicating that these methods not only provide superior SR performance, but this performance boost is useful in a practical setting.

Finally, the proposed FSRONN model outperforms its CNN model equivalent, FSRCNN, across all metrics for each individual preprocessing technique. This demonstrates the power of the self-operational filters over the standard convolutional filters for this task. The FSRONN model also provides competitive performance with the other experimental models, although it is hard to identify if this is due to the self-operational layers due to the different architectural components of the other models.

5.4 Hyperspectral Image Super-Resolution on Real Data

One of the main challenges with SR is that it is particularly challenging to recover the high-frequency information lost in the LR domain, which makes SR an ill-posed problem. This issue is further exacerbated by the difficulty of acquiring paired training data. Due to the absence of paired HSI-SR data, most researchers in this field resort to synthetic downsampling methods (including the experiments conducted up to this point within this chapter), despite their known limitations [115, 116]. This section of the chapter addresses this issue by evaluating HSI-SR models on real paired datasets acquired in collaboration with Thomas De Kerf, University of Antwerp, Belgium, and David Dunphy, University of Strathclyde, Scotland.

5.4.1 Methodology

To estimate the actual improvements in using the real paired datasets over using common downsampling methods to generate LR data, a diverse group of super-resolution algorithms were evaluated. Experiments were conducted on the HSI-SR models 3DHSRCNN [17], the previously proposed SRONN, and the GAN-based model with a band-attention mechanism propsed in [72] which is referred to as BAGAN. This encompasses a variety of models including both 2D and 3D models, traditional and GAN-based frameworks, and Self-ONN [82].

The models were trained on the real HR and LR pairings, as well as artificial downsampling methods, including bicubic and Gaussian downsampling, as is commonly done in the HSI-SR community. Bicubic interpolation is defined in Eq. (5.7):

$$P_{out}(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} P_{in}(x+i, y+j) \cdot f(i) \cdot f(j) \tag{5.7}$$

where $P_{out}(x,y)$ represents the output pixel value at coordinates x, y, $P_{in}(x+i,y+j)$ represents the input pixels from the 4x4 grid surrounding the location x, y, and $f(i)$ and $f(j)$ are the cubic functions that determine how much influence each of the surrounding input pixels has on the final interpolated value of the output pixel.

Gaussian downsampling is defined in Eq. (5.1), which was used in all previous experiments within this chapter. A σ value of 1.6986 was selected for k, following the approach in [118], to represent how Gaussian downsampling is commonly applied within the community. Following the method of T. De Kerf [127], an optimised sigma value of 3.206 was used to examine whether this would offer improved performance. Downsampling using this value is referred to as Optimal Gaussian. 4× subsampling, $\downarrow_s$, was selected as this is the scale factor of the true data that would be used for performance evaluation. No noise was added, so the parameter n was ignored.

Models were trained by both downsampling the real high-resolution image but also by downsampling the real LR image and using the real LR as the target for training. The purpose of this was to simulate how super-resolution models are trained using unpaired images, and then evaluate the efficacy of such a training pipeline to enhance the target images themselves by enhancing the real LR image and comparing it to the real high-resolution image. These experiments were named Gausssian and Bicubic Bootstrap. An illustration of the training data pairs can be seen in Figure 5.18. For all experiments, testing and validation was always conducted on the test and validation portions of the real HR and LR image pairs, even when training on the synthetic downsampling configurations, as this represents the real-world scenario and is the configuration of interest when it comes to evaluation.

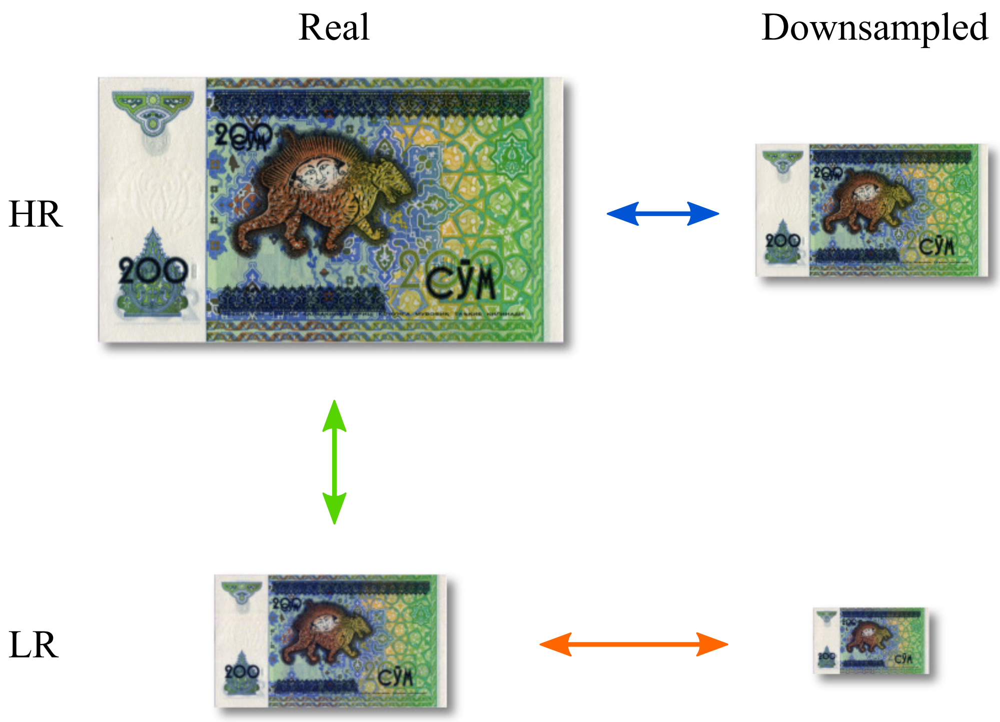

Figure 5.18: Training Image Pairs (not to scale)

Each model has its own set of unique training parameters, and these parameters were kept the same across all experiments on the different downsampling techniques. The original optimiser and loss function used to train each model when they were first proposed was selected. The learning rate and the epoch to reduce the learning rate by a factor of 10 was empirically set for these specific experiments. For the BAGAN and 3DHSRCNN modes, which both use 3D convolution, image patches containing 32 random channels were extracted and passed to the model during training. For the SRONN model, patches containing all 224 channels were extracted and passed to the model. All training parameters used for each model can be seen in Table 5.13.

Table 5.13: Training Parameters

Model	Channels	Learning Rate	Learning Rate Reduction Epoch	Optimiser	Loss Function
SRONN	224	10^{-4}	300	Adam	MAE
3DHSRCNN	32	10^{-3}	300	Adam WD (10^{-4})	Charbonnier
BAGAN	32	10^{-4}	300	Adam	BAGAN

Model performance was evaluated using the PSNR, SSIM, SAM and ERGAS [128] metrics.

5.4.2 Results

Of the 31 scenes available for the Lens Dataset, 23 were used for training, 4 for validation, and 4 for testing.

The experimental results for the Lens Dataset can be seen in Table 5.14 with example outputs from the best-performing model shown in Figure 5.19.

The results for the Sensor Dataset can be seen in Table 5.15 with example outputs from the best-performing model shown in Figure 5.20. Due to the large amount of sensor noise present in the LR images in this dataset, median filtering with a 3x3x3 kernel in addition to bicubic interpolation was also performed to form baseline metrics on the test data.

Table 5.14: Lens Dataset Results

Downsampling Method	Architecture	PSNR ↑	SSIM ↑	ERGAS ↓	SAM ↓
Real[1]	Bicubic Interpolation	33.5884	0.8746	2.3708	2.504
Real[1]	BAGAN	**35.5471**	**0.9099**	**1.8262**	3.0494
	3DHSRCNN	**34.9113**	**0.8874**	**1.9729**	3.3485
	SRONN	**35.5815**	**0.915**	**1.8211**	2.9236
Bicubic[2]	BAGAN	33.3865	0.8714	2.4137	2.5112
	3DHSRCNN	33.116	0.8641	2.4968	2.5447
	SRONN	33.472	0.8731	2.398	2.532
Gaussian[3]	BAGAN	32.4589	0.8674	2.6675	2.6152
	3DHSRCNN	32.6106	0.8644	2.6148	2.7109
	SRONN	32.792	0.8715	2.588	2.687
Optimal Gaussian[4]	BAGAN	32.595	**0.8766**	2.633	4.109
	3DHSRCNN	32.595	0.8647	2.602	3.768
	SRONN	33.392	**0.8894**	2.398	3.553
Bicubic Bootstrap[5]	BAGAN	33.3167	0.8693	2.4386	2.5347
	3DHSRCNN	33.0603	0.8625	2.5185	2.5501
	SRONN	33.393	0.8715	2.4	2.606
Gaussian Bootstrap[6]	BAGAN	32.272	0.8612	2.7284	2.8004
	3DHSRCNN	32.4324	0.8604	2.6567	2.7927
	SRONN	32.685	0.8735	2.609	2.681

Values in bold indicate superior performance to bicubic interpolation.
All test metrics are from the real HR and LR test set.
Downsampling method refers only to the training data.
[1] Real: the real HR and LR pair.
[2] Bicubic: the real HR paired with the bicubic downsampled HR image.
[3] Gaussian: the real HR paired with the gaussian downsampled HR image.
[2] Optimal Gaussian: the real HR paired with the gaussian downsampled HR image with a blur value optimised for this dataset.
[5] Bicubic Bootstrap: the real LR paired with the bicubic downsampled LR image.
[6] Gaussian Boostrap: the real LR paired with the gaussian downsampled LR image with standard blur value.

5.4.3 Discussion

The results from both datasets in Table 5.14 and Table 5.15 clearly show that training HSI-SR models using real low- and high-resolution data pairs significantly improves performance.

Synthetic Downsampling

In the case of the Lens Dataset, the models trained using the synthetic downsampling

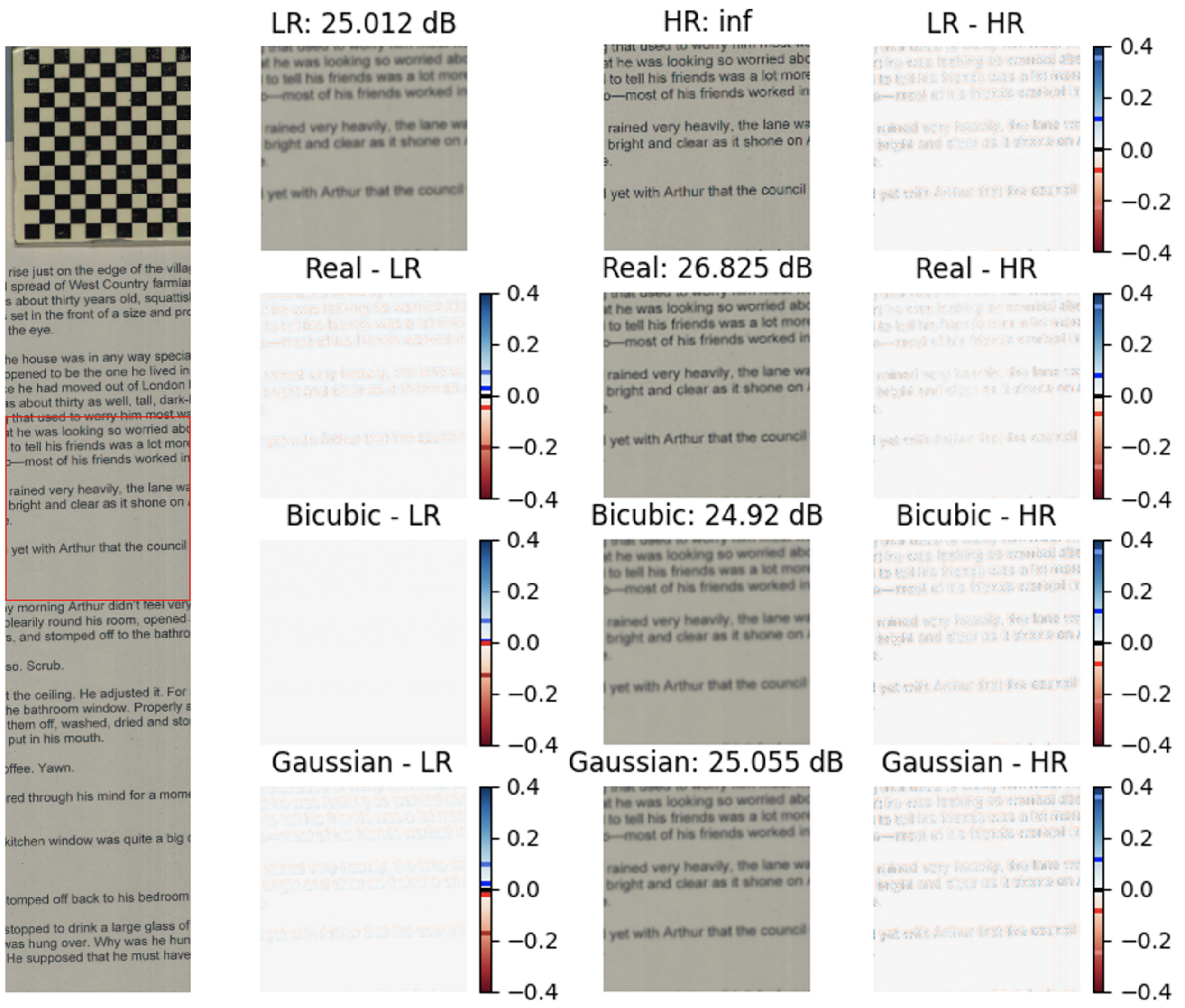

Figure 5.19: Lens Dataset SRONN Outputs.
Left: False-color image with red box indicating input patch location. **Middle-left**: LR patch (top) and difference between model predictions and LR patch, highlighting changes applied. **Middle-right**: Ground-truth HR patch (top) and SRONN predictions (trained on real, bicubic, and Gaussian downsampling); PSNR values shown. **Right**: Pixel-wise difference between predictions and ground truth to assess accuracy. Colour bars feature median (black line), 5th and 95th percentiles (dark red/blue), and range (light red/blue). Bootstrap experiment outputs are omitted for brevity.

processes consistently underperform simple bicubic interpolation when tested on the real pairing. The only exceptions to this are that the BAGAN and SRONN models show a very slight improvement in SSIM when trained using the optimal gaussian filter downsampling method, which itself depended on the real data pairing to get the optimal filter value. Though

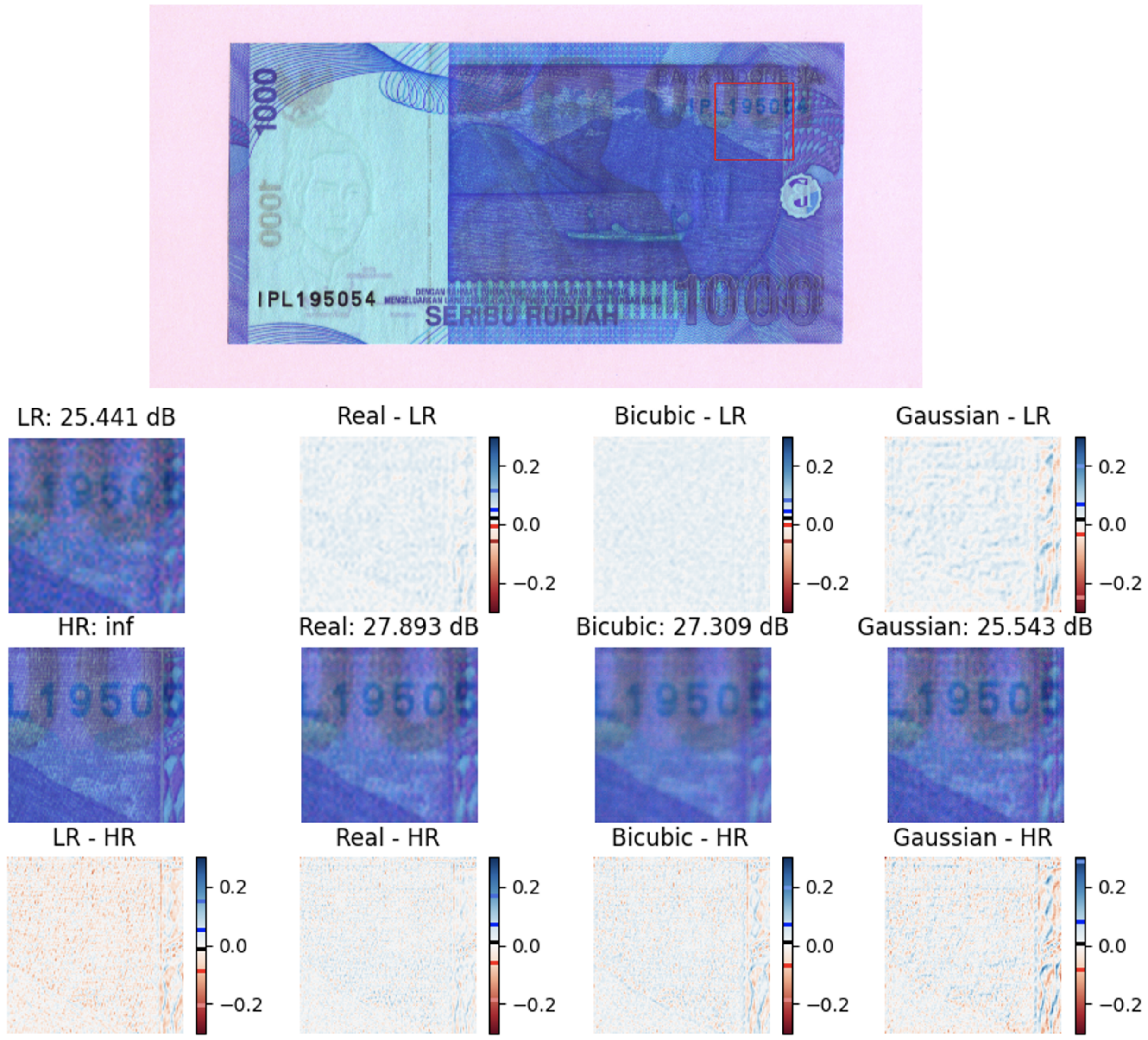

Figure 5.20: Sensor Dataset BAGAN Outputs.
Top: false-colour image with a red-annotated test patch. Smaller images include the interpolated LR patch (left), BAGAN predictions from the model trained with each downsampling method on the real test patch (middle), and the original HR patch (right), each annotated with PSNR values against the HR reference. Heatmaps above prediction images highlight model enhancements versus the LR input, while those below assess accuracy against the HR patch, including an LR and HR comparison at the bottom left. Colour bars indicate median (black line), 5th and 95th percentiles (dark red/blue), and extremes (light red/blue).

Table 5.15: Sensor Dataset Results.

Downsampling Method	Architecture	PSNR ↑	SSIM ↑	ERGAS ↓	SAM ↓
Real[1]	Bicubic Interpolation	24.704	0.5813	2.266	3.699
	Median Filtering & BI	25.633	0.685	2	3.036
Real[1]	BAGAN	**28.349**	**0.7558**	**1.567**	**2.625**
	3DHSRCNN	**28.263**	**0.7566**	**1.611**	**2.745**
	SRONN	**26.481**	0.6249	**1.953**	3.465
Bicubic[2]	BAGAN	**27.657**	**0.7071**	**1.682**	**2.776**
	3DHSRCNN	25.19	0.6271	2.128	3.313
	SRONN	24.701	0.5813	2.267	3.701
Gaussian[3]	BAGAN	**25.998**	0.6407	2.033	3.350
	3DHSRCNN	23.9923	0.5983	2.437	3.917
	SRONN	24.739	0.5818	2.261	3.713

Values in bold indicate superior performance to bicubic interpolation with median filtering.
All test metrics are from the real HR and LR test set.
Downsampling method refers only to the training data.
[1] Real: the real HR and LR pair.
[2] Bicubic: the real HR paired with the bicubic downsampled HR image.
[3] Gaussian: the real HR paired with the gaussian downsampled HR image.

these models still fail to outperform the bicubic benchmark across the remaining metrics. All other models trained on the synthetic data failed to outperform the bicubic benchmark across all metrics. These results confirm that training HSI-SR using synthetic downsampling techniques does not perform well on real data and thereby revealing the necessity of using real paired images for super-resolution training.

Similar to the results on the Lens Dataset, the models trained using the synthetic downsampling processes on the Sensor Dataset consistently underperform the bicubic interpolation benchmark. Only the BAGAN model outperforms bicubic interpolation with median filtering when trained on data synthetically generated low-resolution images. This may well be due to the adversarial loss function used to train the BAGAN model enabling the model to learn the low-noise distribution of the target high-resolution data and thus producing smoother outputs when fed a noisy (real) low-resolution input image, enabling better performance than bicubic interpolation with median filtering due to its denoising capacity rather than its super-resolution capacity. This is evident in Figure 5.20 as the output patch from the model trained with bicubic interpolation produces a PSNR value significantly better than the low-resolution image and somewhat close to the output from the model trained on the real pair, but the patch produced is much more blurry than the true high-resolution image and the predicted output of the model trained on the real data.

Real Data

Contrary to the synthetic downsampling, the models trained using the real high- and low-resolution image pairs consistently outperform the bicubic interpoplation benchmark by a significant margin, demonstrating the value of this data. This improved super-resolution is also evident in Figure 5.19 where the model trained on real data produces a significantly shaper and more detailed patch than the output of the two models trained using data downsampled with bicubic and gaussian downsampling.

Interestingly, the models trained on the real pairing do not outperform bicubic interpolation in terms of SAM and in fact perform worse than the models trained with artificial downsampling methods for this metric. This could be due to the fact that artificial downsampling methods produce a more detailed synthetic low-resolution image than the true low-resolution image, reducing the complexity of the low- to high-resolution mapping function learned by models trained with these data compared to models trained on the real low-resolution image. Thus, when the models trained on synthetically generated low resolution images are applied to real low-resolution data, they produce smoother and less detailed outputs, as shown in Figure 5.19 and evidenced by their low spatial metrics. This smoother output could also lead to ”cleaner” spectra and may be why they produce better SAM values than the models trained on the real data that are attempting to recover greater spectral detail. Though notably, these SAM values are still worse than the bicubic benchmark.

The models trained on the real data pair from the Sensor dataset consistently outperform the bicubic interpolation with median filtering baseline, and also consistently outperform the models trained with synthetic downsampling techniques by a significant margin, further demonstrating the value of training using real data pairs.

The SRONN model performs significantly worse on this dataset than the two other experimental models, which is most likely due to the 3D filters in the other models being better suited to tackling the higher levels of noise within the low-resolution patches of this dataset than the 2D filters used in the SRONN model. Interestingly, the BAGAN model trained with bicubic downsampling is able to outperform the SRONN model trained on the real data pair. This is likely due to a combination of the SRONN model's 2D filters and the reasons discussed in the previous sub-section. Despite this, all experimental models see a significant performance improvement across all metrics when trained using the real data pair over the synthetic downsampling methods.

Optimised Gaussian downsampling

Although the results from the models trained with the bicubic downsampling method objectively produce better results than the models trained with Gaussian downsampling on the Lens dataset, the bicubic downsampling methods produce results that are very similar to the bicubic interpolation baseline. All experimental models contain residual connections and in the case of the bicubic downsampling methods, models learn to output predictions very similar to the input as the blank ”Bicubic - LR” patch of Figure 5.19 shows. While the Gaussian downsampling method produces objectively worse metrics than the bicubic downsampling method, the non-blank ”Gaussian - LR” patch in Figure 5.19 shows that the model is in fact changing the input and learning something, though it lacks accuracy as the ”Gaussian - HR” patch of Figure 5.19 reveals. By optimising the σ value used in the Gaussian downsampling method, the performance of this method improves in terms of spatial metrics when the models are applied to real data, slightly outperforming bicubic interpolation in terms of SSIM. However, even with an optimised σ value, the performance of models trained on synthetic data remains significantly inferior to those trained on real image pairs. This gap underscores the limitations of synthetic downsampling methods, as they fail to capture the complexities and nuances of real-world data. Notably, the SAM metric deteriorates further with an optimised σ value, mirroring the results seen when models are trained on real data pairs. This deterioration highlights that, despite efforts to refine synthetic training data, only real images provide the necessary detail and variability for models to achieve superior performance, reinforcing the importance of using authentic datasets for effective training.

Bootstrap experiments

The bootstrap experiments presented in Table 5.14 produce fairly similar results to the standard artificial downsampling experiments, indicating that super-resolution performance may be translatable to higher scales. However, given that there are residual connections present in each experimental model and the models do not outperform bicubic interpolation, it is not conclusive whether or not the models are actually translatable across scales or if the models in fact just learn to output an image similar to the image interpolated with bicubic interpolation, making it appear scale-invariant. A model that offers improvements over bicubic interpolation would be required to draw any firm conclusions on this. Based on the results, an even lower-resolution data pair would have to be acquired in order to train the model on a real lower-scale data pair to outperform bicubic interpolation at this scale before evaluating on the higher scale. Such an experiment and data gathering are left for future work.

5.5 Summary

In this chapter, the topic of HSI-SR has been explored in depth using a broad array of datasets. First, HSI-SR was explored on small well-known datasets, as is commonly done in the literature, where novel Self-ONN models were proposed to improve both SR performance as well as parameter efficiency. Furthermore, a study was conducted to determine how residual connections and various normalisation layers affected performance within these models.

To further improve SR performance, experiments were then conducted on the much larger and less commonly used ICONES dataset, consisting of almost 500 HSIs. Novel preprocessing techniques were proposed to better prepare the data for training and improve SR performance on this dataset especially with the proposed *Band SNV* method. Furthermore, it was found that this technique not only improves performance, but additionally improves convergence time and narrows the performance gap between models, reducing the required model complexity for such a task. Another novel Self-ONN model was proposed for this task, and this model provided better performance than all other model types when the data were preprocessed minimally, even in a situation where the model was somewhat ill suited, as the channel indices passed to the model were not consistent. However, when the novel *Band SNV* preprocessing algorithm was used, this performance dominance was reduced and some other non-Self-ONN architectures were able to outperform the proposed FSRONN model, revealing the efficacy of the proposed preprocessing method.

Finally, to address the limitations of training SR algorithms using syntheic downsampling processes, experiments were carried out on a novel dataset acquired in collaboration with other researchers. This dataset consisted of real high- and low-resolution HSI pairs that could be utilised during training instead of having to rely on suboptimal synthetic downsampling processes. Experiments were carried out using both real data pairs and synthetic downsampling processes to artificially generate LR pairs to evaluate how these models trained on synthetic downsampling processes perform when evaluated on real data. It was found that not only do models trained with real image pairs significantly outperform models trained with synthetic downsampling processes, but models trained with synthetic downsampling processes often fail to even outperform simple bicubic interpolation when evaluated on real data.

6 Self-Supervised Cow Identification

Accurate and continual identification of individual cattle from any location holds great potential for the dairy industry. Such systems would allow for continuous monitoring of each cow, offering valuable insights into health, behavior, feed consumption, and milk production - key factors in precision dairy farming. This capability not only promotes animal welfare by enabling early detection and treatment of health issues but also enhances operational efficiency and profitability. However, in such a scenario, traditional identification methods such as RFID tags are not feasible as the cows are free-moving and the scanners can only identify in fixed locations.

This chapter investigates the ways in which cow identification through non-invasive aerial cameras can be performed in the real-world target domain of the barn shown in Figure 1.1. This domain is much more challenging than the milking parlour domain explored in Chapter 4 for two main reasons. Firstly, there is far greater variation among images due to the complete free movement of the cows, allowing for far greater positional variation and difference in the perspective of the cow relative to the camera, among other factors. Secondly, acquiring a large labelled dataset for supervised training is not feasible in this domain. This is due to the fact that a human labeller would have to look at each individual image within the dataset and make a correct identification of the presented cow out of the 1785 cows in the farm without any auxiliary information, which is near impossible and highly error prone. The proposed method for identification is therefore to use images from the milking parlour domain, where labels can be generated automatically via RFID scanners - to fit a KNN classifier to make predictions on barn images as Figure 6.1 demonstrates.

In an ideal scenario, models would simply be trained using the supervised datasets in the easily acquired milking parlour domain from Chapter 4 and directly applied to the barn domain. However, as will be shown in this chapter, this does not work, and thus alternative training strategies are explored to perform identification in this domain. This issue where training a model on data from a source domain (in this case the milking parlour)

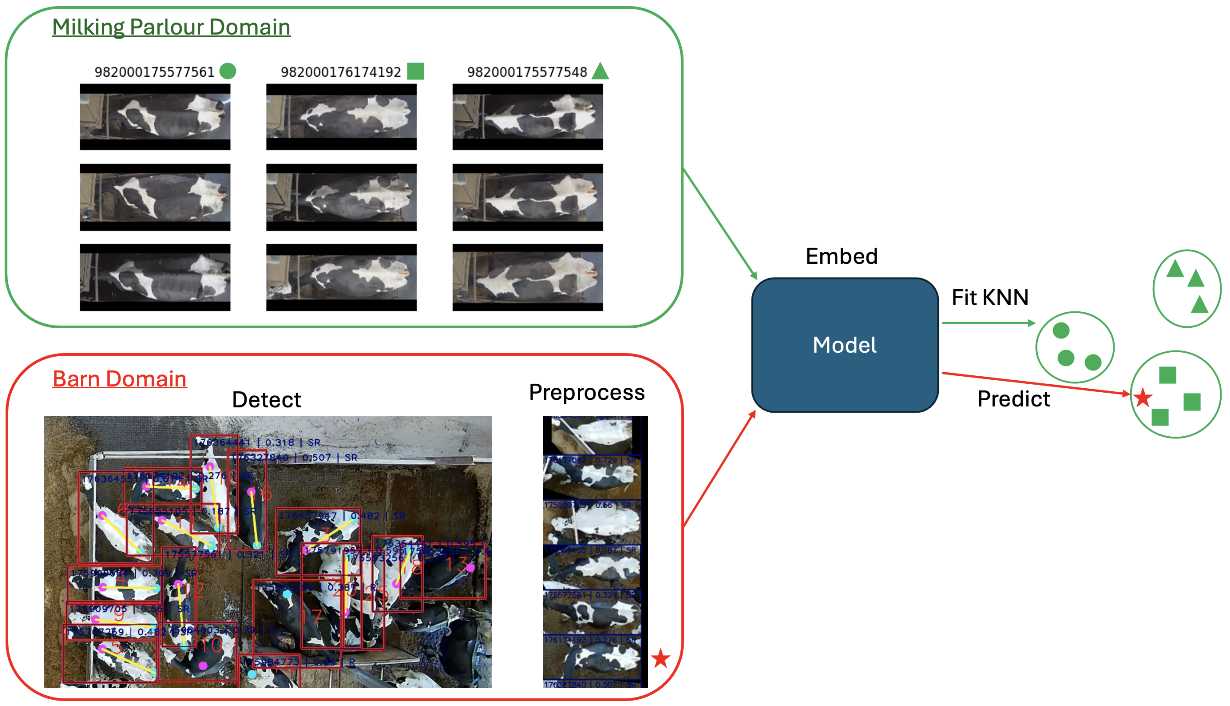

Figure 6.1: Barn Identification Pipeline

- where labelled data are available in abundance - does not perform well on the desired target domain (in this case the barn) is a common issue and an active area of research [98]. The most obvious solution is to train on data from the target domain. However, data from the labelled target domain are not always available in the quantity required to train a Deep Neural Network (DNN). Thus, labelled data from a similar (source) domain with abundant availability can be utilised for training. Domain adaptation aims to tackle this issue by training with a combination of source domain data with either a small amount of labelled target domain data (supervised domain adaptation) or a large amount of unlabelled target domain data (unsupervised domain adaptation). In this chapter, barn domain data is gathered in an automated fashion with weak labels, and two novel data gathering techniques combined with two novel self-supervised training techniques are proposed to perform domain adaptation between the milking parlour and barn domains.

The novel contributions of this chapter are listed as follows:

- Self-supervised techniques for real-world dairy cattle identification

- Automated data gathering techniques for use in self-supervised dairy cattle identification training

6.1 Datasets

In this chapter, datasets are collected from barn videos in an automated fashion, exploiting various YOLO models and preprocessing algorithms to extract individual cow detections from barn video footage and create an identification dataset. Since there are no RFID matches to each detection, and human labelling is practically infeasible in this scenario due to the difficulty of manually identifying any given cow out of 1785 possible cows, there are no true labels for these barn identification datasets. However, due to the nature of the proposed data acquisition processes, there are weak labels available which provide contrastive information about the captures, which will be discussed in more detail in each relevant dataset subsection.

Since there are no true labels available for any of the barn domain datasets, the milking parlour supervised CowID-537 and CowID-1785 datasets from Section 4.1 were used for validation and testing. All training data used in this chapter is gathered from the barn domain and does not contain any true labels and can be gathered entirely automatically without any human intervention whatsoever. Meaning that in theory, training data can continuously be gathered.

All datasets used in this chapter (both unlabelled barn training data and labelled milking parlour validation/testing data) were acquired from the same farm. However, since the datasets were all acquired at different points in time, the cows present in each are not necessarily the same, though there is likely a great deal of overlap. The barn itself contains 100 fish eye cameras with full coverage of the barn and a slight amount of image overlap between adjacent cameras.

The objective in this chapter is to make predictions based on detections acquired in the barn domain and connect them to cow detections acquired from the milking parlour domain which have corresponding RFIDs.

6.1.1 Barn Dataset

The barn cameras were set to record and save 5 second clips at 50 minute intervals where all cameras would be recording simultaneously. In an ideal scenario, the cameras would be recording 24/7; however, this is not practical due to the vast amounts of data that this would produce.

A YOLOv8 [129] model developed for the general detection of Holstein cattle was used to extract individual cow detections from each video. The YOLOv8 model also produced detections for the shoulder and rear of the cattle as basic keypoints, but these keypoints were not linked to the cow detections and instead were treated as separate detections. On the assumption that each detection at a given instance in time is a unique cow, the detections for each camera were saved into a folder corresponding to the time the video was captured. To prevent multiple detections of the same cow from being saved in each folder, each video frame was passed to the YOLO model and the detections from the frame producing the highest number of detections were saved while the detections from the other frames were discarded. Note, the detections from each camera video for a given capture time would not necessarily be obtained from the exact same instance in time. However, due to the slow movement speed of the cows, it is highly unlikely that a cow would be able to transition between non-overlapping regions of the camera views within the 5 second window, causing multiple detections of the same cow for the given capture time. To prevent multiple detections of the same cow occurring in the camera overlap regions, the outer 5% of each video frame was discarded. The data gathering algorithm is shown in Algorithm 1. With this particular detection extraction technique, weak labels are generated as all the detections within a given capture time folder are unique cows, although the RFIDs of each individual cow/detection are unknown.

In order to reduce the complexity of the problem space and make the barn domain as

Algorithm 1 Barn Dataset Extraction Algorithm

```
for each capture time do
    Create Save Folder
    for each camera video do
        for each video frame do
            Cut frames by 5% from each side
            Get Detections from YOLOv8 Model
            if Frame has Most Detections in Video then
                Save Detections to Folder
            end if
        end for
    end for
end for
```

similar to the milking parlour domain as possible, preprocessing was applied to each cow detection such that they would have the same orientation as the milking parlour images. Each cow image in the CowID-1785 dataset was facing leftwards, so the keypoint detections from the barn frames were used to orient the cow detections leftwards. Since the keypoint detections were not linked to the cow detections, the algorithm detected if there were keypoint detections present within the cow detection box and used these keypoints. If there was no shoulder or rear keypoint present within the cow detection, then the detection was discarded. If there were multiple shoulder keypoints present, then the central most keypoint was selected as this would most likely be the keypoint belonging to the detected cow. If there were multiple rear detections, then the rear detection furthest from the shoulder (still within the cow detection box) was used. Once the orientation of the cow detection was corrected, the detection was then resized and cropped so that the resulting detection had a height of 100 pixels and a width of 224 pixels, consistent with the dimensions of the milking parlour images. The preprocessing algorithm is described in Algorithm 2. with an example output shown in Figure 6.2

However, this preprocessing algorithm was not perfect as there was no way to be sure that the rear and shoulder keypoints used for the orientation correction were indeed the correct keypoints for the given cow detection. Although the majority of corrections were

Algorithm 2 Detection Preprocessing Algorithm

```
for each cow detection do
    Get All Shoulder and Rear Detections with Cow Detection Box
    if No rear or shoulder detections then
        Discard Cow Detection
    else
        if Multiple Shoulder Detections then
            Select Central Most Shoulder Detection
        end if
        if Multiple Rear Detections then
            Select Rear Detection Furthest from Shoulder Detection
        end if
        Use Shoulder and Rear Detections to Correct Orientation
        Resize and Crop Detection to 224x100 Pixels
    end if
end for
```

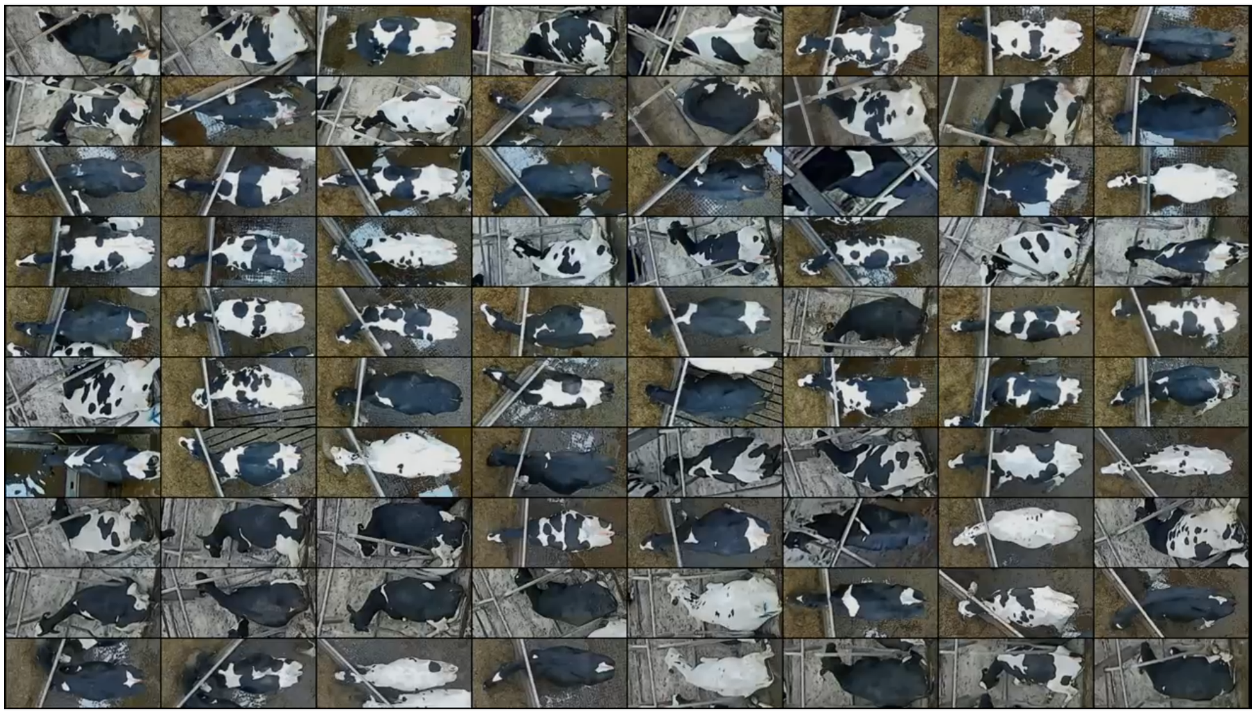

Figure 6.2: Barn Dataset Example Images
Example output of a folder of images acquired using algorithm 1 at a given instance in time and preprocessed using algorithm 2.

good, this drawback resulted in a small percentage of poor corrections.

6.1.2 Barn Track Dataset

To improve upon the non-keypoint YOLOv8 model used for the barn dataset that detected the shoulder and rear of a cow using bounding boxes, a YOLOv8 keypoint model was developed by the industrial sponsor, Peacock Technology Limited. In addition to cow bounding box detections, this new model produced proper keypoints for the rear and shoulders which were associated with the given cow detection instead of being treated as separate, unlinked, detections. This greatly improved the reliability of the preprocessing algorithm. Furthermore, a DeepSort [130] tracking model was used in conjunction with this YOLOv8 keypoint model so that cow detections could be tracked between frames to exploit more of the data.

For this dataset, 30 second clips were captured across all cameras at 6 hour intervals, again with all cameras capturing simultaneously. Each video was processed with the YoloV8 keypoint model and DeepSort track model, where 5% of the video was cropped from each side to avoid any overlap between videos. For each time instance, a new folder would be created. Each camera video for said time instance would be processed and a folder for each cow detection occurring within the first 3 frames would be created. These detections would be tracked throughout the video and the detections would be saved every 5 frames to its corresponding folder until the video was complete or the track was lost. Any new tracks after the first 3 frames would not be saved as there was no way of knowing if the new track was indeed a new cow, or a previously lost track, meaning that there would potentially be duplicate track folders for a given cow. Finally, any tracks with less than 4 images were discarded. The full algorithm is described in Algorithm 3. The detections acquired via this algorithm were again preprocessed with the algorithm described in Algorithm 2.

Since it is assumed that each track is a different cow for a given time instance, this dataset also contains weak labels as each track is different, but there are no RFIDs corresponding to the tracks. However, each track has multiple images which can be exploited during training.

Algorithm 3 Track Dataset Extraction Algorithm

```
for each time instance do
    Create Save Folder
    for each camera video do
        for each frame in video do
            Get Detections & Keypoints from YOLOv8 Model
            Track Detections with DeepSort
            Save Tracks Originating from 1st 3 Frames
        end for
    end for
end for
```

6.2 Domain Transfer Analysis

The best milking parlour cow identification model developed in Chapter 4, SE_ResNeXt_32, was used to perform embedding analysis. A small portion of data was gathered from the target farm barn cameras where cow tracking data was collected and extracted from the footage in a similar manner to way in which the Barn Track Dataset was acquired. The data were then manually matched to 28 RFIDs from the CowID-1785 dataset where each RFID contained a varying amount of images. The matching process was extremely difficult and time consuming, as each barn cow would need to be manually compared to all 1785 milking parlour cows, which combined with the great difference in appearance between the barn and milking parlour captures, made this process extremely challenging. Therefore, this small portion of labelled barn data was only used for domain transfer analysis and final testing of the trained models. Examples of the labelled barn data alongside their milking parlour captures can be seen in Figure 6.3. The labelled barn data was embedded using the best performing supervised model from Chapter 4 together with the milking parlour images for the corresponding RFIDs. These embeddings were reduced to two dimensions using T-distributed Stochastic Neighbour Embedding [131] and plotted to examine the clustering capacity of the trained model in both the milking parlour and the barn domains, which can be seen in Figure 6.4.

The figure shows that model has excellent clustering capacity on the milking parlour

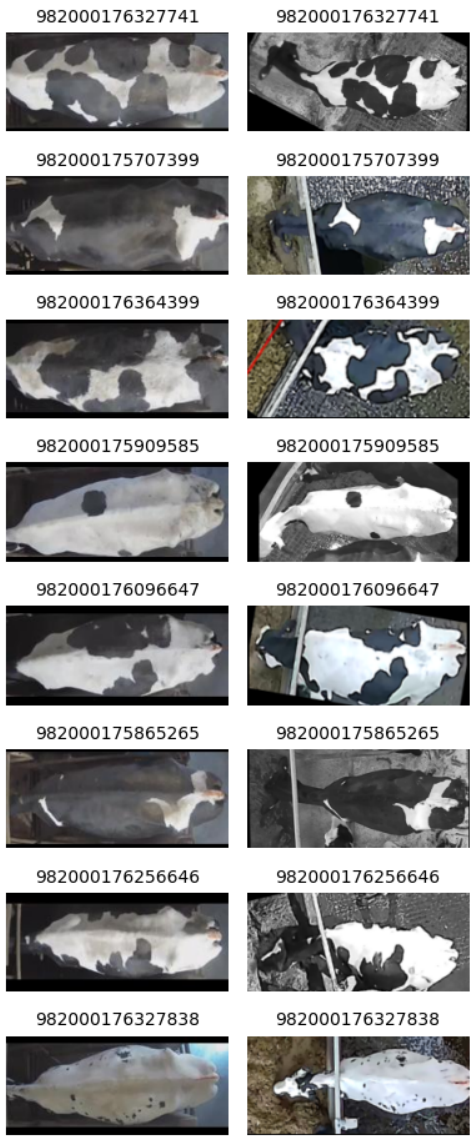

Figure 6.3: Example Labelled Barn Captures
(left) next to the milking parlour capture for the same cow (right).

images it was trained on and has reasonably good clustering capacity on the barn images, though not as good as the milking parlour. The key takeaway from the figure, however, is the poor overlap between the embeddings from the different domains, revealing that the model performs poorly in terms of producing well-clustered embeddings across domains.

To further analyse this model's cross-domain performance, the accuracy on the labelled barn data was computed in the same way accuracy was calculated for the experiments

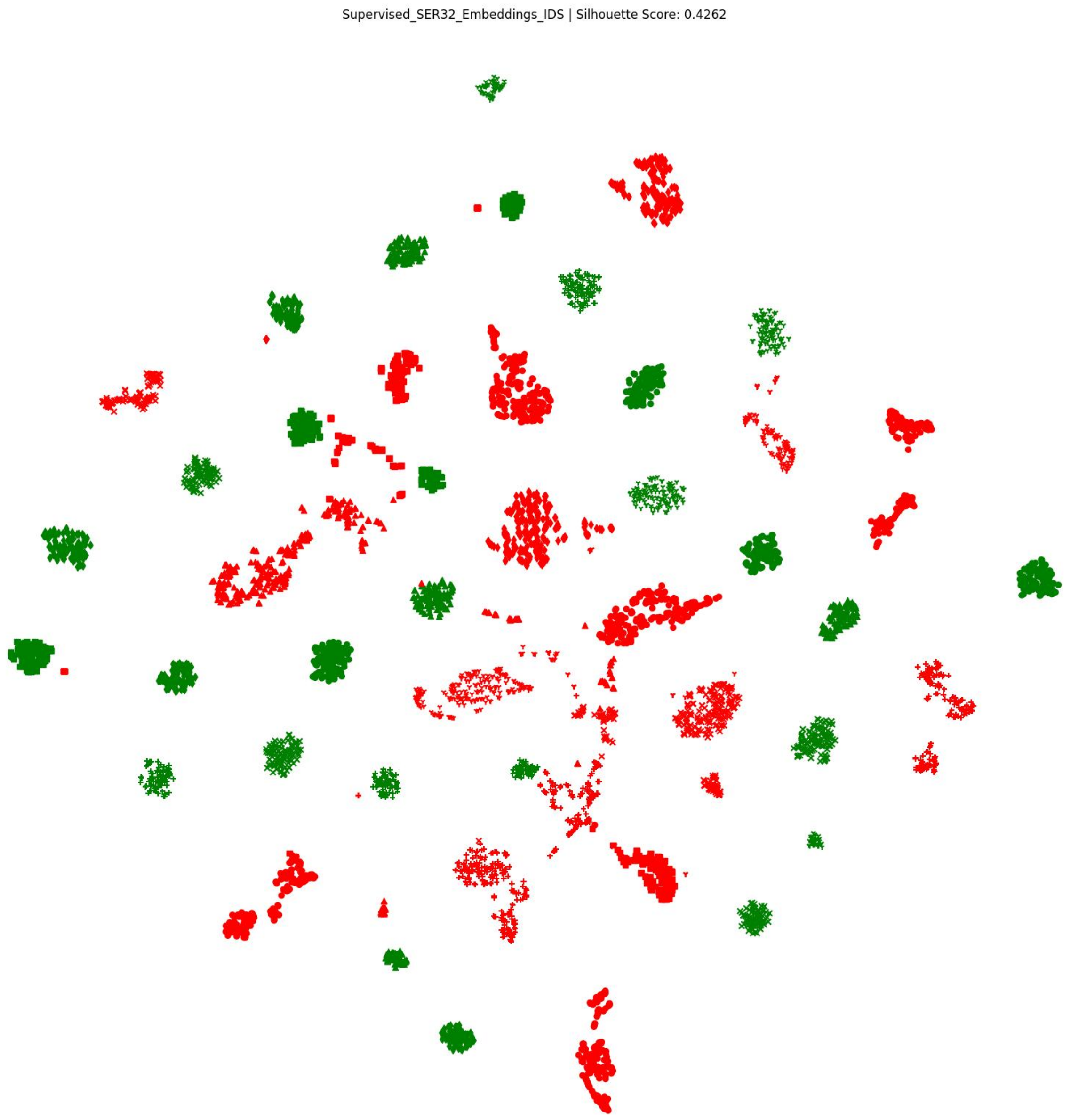

Figure 6.4: Supervised Model Embeddings Plot
Milking Parlour and Barn domain image embeddings from the supervised model. Point shapes represent different classes (though shapes are repeated across classes due to the limited number of shapes available)

conducted in Chapter 4. The images from the CowID-1785 dataset were embedded and used to fit a KNN classifier. The embeddings from the labelled barn data were classified using a K value of 3, 5 and 7 and the results from the best K value are reported in Table 6.1.

The results from Table 6.1 reveal that there is a huge deterioration in identification

Table 6.1: Supervised SE_ResNeXt_e32 Model Performance on Labelled Barn Data

Dataset	NN	Accuracy (%)	F1 Score	Silhouette
CowID-1785	7	99.73	0.9973	0.7899
Barn Identification	7	30.09	0.393	0.4514

performance when a model trained on milking parlour data is applied to barn data, falling from over 99.7% accuracy to less than 30.1%. This performance deterioration highlights the need for an alternative training strategy if identification is to be performed effectively in the barn setting.

6.3 Self-Supervised Similarity Learning-Based Training Algorithm

Most self-supervised algorithms learn from different parts or augmented views of the same image. However, to prevent representation collapse [132], such methods rely on either an online teacher network or negative sampling. Methods using the latter encounter issues to do with the negative sampling process and additionally do not guarantee that the negative samples are, in fact, from a different class. Due to the nature of the acquired barn dataset where all detections at a given instance in time are assumed to be different cows, negative examples which are guaranteed to be from a different class can be sampled from this unlabelled dataset. A novel self-supervised algorithm is proposed that takes advantage of this property of the barn dataset. The proposed algorithm samples batch images acquired from the same instance in time, guaranteeing that all samples are of different cows given that a cow cannot be in more than one location at given time. Positive pairs are generated via augmentation where these samples are utilised in the triplet loss function defined in Eq. (2.18) as represented in Figure 6.5. The proposed algorithm alleviates the issues encountered with alternative random negative sampling strategies, while also avoiding reliance on a student teacher framework. Furthermore, the training objective is similar to the supervised objective from Chapter 4, allowing for off-the-shelf classification evaluation without finetuning.

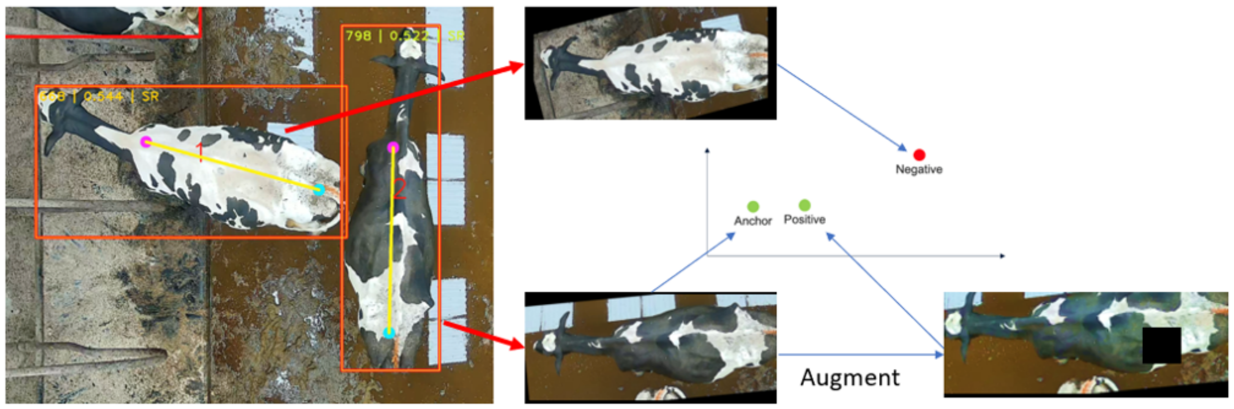

Figure 6.5: Proposed Self-Supervised Algorithm
Barn camera frame passed to YOLO model to detect cows present. Detections then preprocessed and utilised in training where the anchor detection is augmented to create the positive pair.

6.3.1 Methodology

To ensure that each image contained within a batch was a unique cow, any given training batch was formed using data from a single capture time, where 25 detections were randomly selected. Note, the triplet loss function does not require information on the specific classes; rather it only requires knowledge about which samples are the same or different in the current batch, therefore, the specific RFIDs do not matter, only that each cow is different and has a unique label which can be randomly assigned. Each image was then randomly augmented 3 times using the augmentations from Table 4.2 to create a selection of potential positive pairs where the augmented images share the same random label as the original. This created a total batch size of 100 images with four images per cow (original plus three randomly augmented images). Online triplet mining [34] was then applied to select the most difficult triplet combinations with a margin greater than 0.5. In every epoch, each time instance would be iterated over once where a batch would be formed by randomly selecting 25 images from that time instance and applying the mentioned augmentation strategy.

The training parameters for the self-supervised experiments were selected based on the experiments conducted on the CowID-1785 dataset from Chapter 4 due to the similarity between the applications. Most parameters were kept the same including the use of the

Table 6.2: Self-Supervised Algorithm Training Parameters

Parameter	Value
model	SE_ResNeXt50
triplet margin	0.5
triplet selection strategy	hard
optimiser	AdamW (lr 0.001, weight decay 0.005)
lr decay step	500
chances	300

hard triplet loss function with a triplet margin of 0.5, the AdamW optimiser with weight decay 0.005 and an initial learning rate of 0.001 which was decreased by a factor of 10 at epoch 500. Due to the validation set being of a different domain to the training data, the validation accuracy did not increase as steadily as the validation accuracy from the experiments in Chapter 4. Therefore, each experiment was given 300 epochs (50 more than the experiments in Chapter 4) to improve upon the previous best validation accuracy before termination, to account for this increased variance. The SE_ResNeXt50 model was selected for experiments as this was the best performing model from Chapter 4. Since there were no previous experiments conducted in this problem space and the best embedding size was unknown, an embedding size of 128 was used for these experiments, as this is the default embedding size used in [34]. The parameters used during training are summarised in Table 6.2.

Experiments were carried out on different combinations of data quantity and capture areas to evaluate the ways in which different portions of the barn affect performance. Data was captured from the right side of the barn where the cows are both lying down and standing, the middle of the barn, where the cows are mostly standing but far more congested, and from the entire barn. Data was captured for varying amounts of time to produce varying amounts of batches (capture instances) and each batch was processed to produce the maximum amount of detections possible as defined in Algorithm 1.

6.3.2 Experimental Results

Given that the acquired barn datasets used for training do not contain labels, portions of this data could not be used for validation and testing. Therefore, the CowID-1785 dataset from Chapter 4 was used for validation and testing as this is a fully labelled dataset that can be used to produce evaluation metrics. The experimental results can be seen in Table 6.3.

Table 6.3: Self-Supervised Training Algorithm Results

Version	Area	Images	Batches	Avg Images/Batch	Accuracy (%)
1	Right	4373	121	36.14	16.513
2	Right	12968	353	36.74	30.254
3	Middle	613	4	153.25	1.972
4	Middle	9288	78	119.08	30.975
5	Middle	30092	237	126.97	46.16
6	All	15330	58	264.31	39.412
7	All	33811	137	246.8	45.531
8	All	55573	224	248.09	49.785
9	All	79371	312	254.39	51.127
10	All	95726	374	255.95	52.063

Results presented in this table are from models trained on the Self-Supervised dataset and evaluated on the CowID-1785 dataset. Therefore, all metrics presented are from the CowID-1785 dataset since there are no labels for the Self-Supervised dataset.

Experiments were also performed using BYOL [91] - a popular self-supervised training method which uses two augmented views along with the student teacher network framework for training. Comparison to other common self-supervised methods could not be done due to either resource constraints from the extremely large batch sizes required [89, 90] or the algorithm's inherent design for transformer architectures [94, 95, 96, 97], making it incompatible with the experimental CNN architectures.

Since BYOL does not take negative samples into consideration random samples were selected for each minibatch without consideration of the capture time. BYOL produces two augmented views of each training sample so 50 random samples were passed to the model at any given time, equating to a total batch size of 100 to remain consistent with the

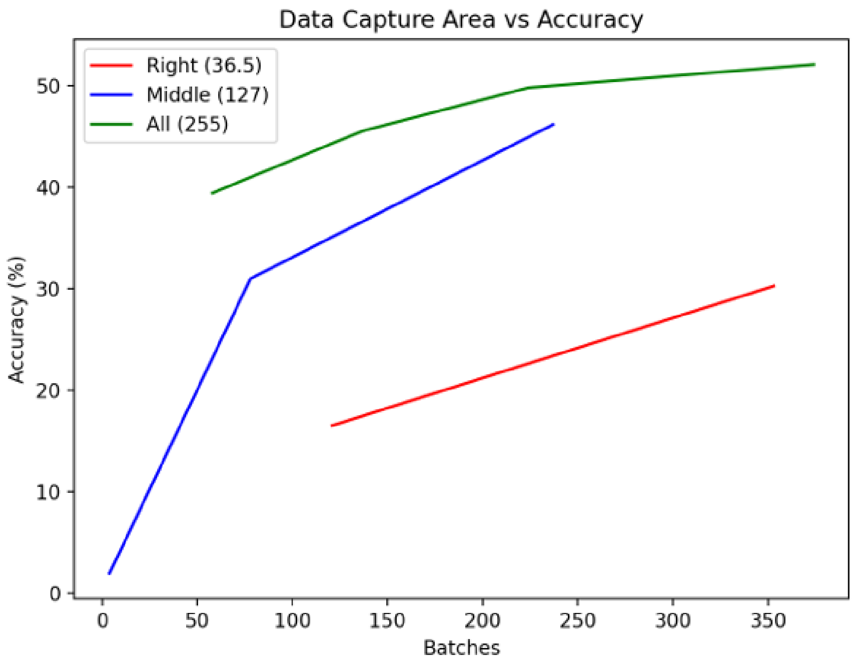

Figure 6.6: Barn Dataset Performance Trend

experiments on the proposed self-supervised algorithm. The same augmentation strategy used on the proposed self-supervised algorithm was also employed to remain consistent. The results can be seen in Table 6.4.

Table 6.4: Self-Supervised Method Test Accuracies on the CowID-1785 Dataset

Version	Method	Accuracy (%)
	BYOL (reg aug)	
5 (Middle)	BYOL	4.549
	Proposed	46.16
10 (All)	BYOL	7.397
	Proposed	52.063

6.3.3 Discussion

The results in Table 6.3 show that using data extracted from all cameras in the barn produces the best results, which is somewhat unsurprising as this produces the largest quantity of data. However, the middle portion of the barn produced the best results relative to the number of images captured, as the results from version 5 outperform the results from version 7 with fewer images. This could be due to the fact that the cow detections from the middle of the barn are more closely related to the milking parlour images used for evaluation, as the cows are predominantly standing in both domains. Finally, the results from the right side of the barn produce the worst results, which is likely due to a combination of a low number of detections per batch and the fact that the cows in this portion of the barn are predominantly lying down, increasing the domain gap between the target barn domain and the milking parlour source domain. The results indicate that if there are no storage constraints, then capturing data from the entire barn is the best strategy, though, if there are memory constraints, then the middle portion of the barn offers the best performance relative to the number of images available.

The results from Table 6.3 are plotted in Figure 6.6 which shows the performance of each capture region relative to the number of batches used. The plot reveals that the performance begins to plateau as the number of batches increases, indicating that the potential performance of this technique is limited. This is likely due to the limited variation that augmented positive pairings can produce. This means that the models produced by this approach likely have a strong discriminative ability due to the negative pairings, but likely have limited clustering ability on real samples due to the absence of true positive pairings.

However, the comparison between the proposed self-supervised algorithm and BYOL shown in Table 6.4 reveals that the proposed algorithm performs far better than BYOL. This is likely due to the use of negative samples enhancing the model's discriminative ability, as the negative sampling is the key difference between the two algorithms here. BYOL is therefore likely not able to effectively separate different class clusters, resulting in its exceptionally poor performance, so the proposed introduction of the negative sampling strategy provides a remarkable improvement.

6.4 Self-Supervised Similarity Learning-Based Training with Tracking Data

To address the limitations of the initial proposed self-supervised training algorithm's clustering ability, a tracking model was incorporated into the data extraction process so that multiple detections for a given cow could be extracted from the video data, allowing for the use of real positive pairs during training. The use of real positive pairs allows the model to encapsulate the true variation between samples of the same class and learn their similarities. Since the extracted track IDs do not correspond to any RFIDs and are not matched between the capture time instances, individual training batches were again formed using data from a single capture time as demonstrated in Figure 6.7.

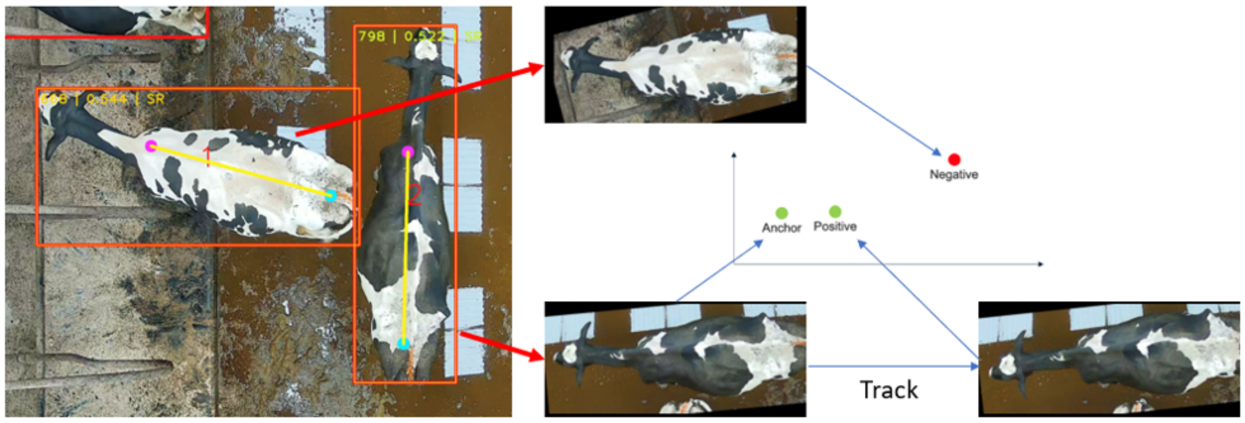

Figure 6.7: Proposed Self-Supervised Track Algorithm
Barn camera frames passed to YOLO model to detect cows present. Detections then preprocessed and utilised in training. The positive pair is selected from a different frame using the track ID of the anchor detection.

6.4.1 Methodology

To form a given training batch, 25 tracks were randomly selected from a given capture time and 4 images from each track were selected, similar to the batch formation done in Chapter 4. The triplet loss function was again used so that the known difference in classes associated with each individual track could be exploited to prevent representation collapse during training. Since positive pairs are sampled from track IDs, augmentation is not a necessary step within this algorithm. The less strict augmentation strategy used in Chapter 4 was therefore used again here to enhance training performance, where the augmentations from Table 4.2 are applied to each batch image with a probability of 80%.

Since the results from Table 6.3 show that using data from the entire barn is most effective, tracks acquired in all regions of the barn were used during training. The same training parameters in Table 6.2 were used. To investigate the efficacy of Self-ONN models in this task, experiments were conducted on the SE_ResONeXt50 model proposed in Chapter 4 along with the regular SE_ResNeXt50 model. Embedding sizes of 128 and 32 were also explored to examine how this affects performance. The standard ResNeXt model was excluded from experiments as the results from Chapter 4 show that the SE variant was superior.

The AdamW optimiser was again used with an initial learning rate of 0.001. Due to the self-supervised nature of the initial self-supervised algorithm, it was observed that training was less stable than the supervised experiments conducted in Chapter 4, making it particularly challenging to set a good learning rate decay epoch. The learning rate scheduling technique proposed in [133] was therefore selected since it uses cosine annealing with warm restarts, eliminating the need to set a decay step. The triplet loss function was of course used and the margin parameter was again set to 0.5.

6.4.2 Experimental Results

All models are trained on the Barn Track Dataset and evaluated on both the CowID-1785 dataset and the small portion of manually labelled barn data, where the CowID-1785 images are used as the reference embeddings for classification, therefore matching the barn images to the milking parlour domain. The experimental results are presented in Table 6.5. While at first glance these results may appear subpar, especially compared with the results from the earlier supervised model, it is worth noting that training in a self-supervised manner is far more challenging than training in a supervised manner, especially when evaluating on a supervised objective. Furthermore, the self-supervised models are not trained on any milking parlour data and therefore have no prior knowledge of this domain, so it is a somewhat unfair comparison of these models to the supervised model that has been trained in a supervised manner on data from this domain. Taking this into consideration, the results are much stronger than they may appear at first glance.

Table 6.5: Self-Supervised Track Model Performance on Labelled Barn Data.

Model	Milking Parlour Identification				Barn Identification			
	NN	Accuracy (%)	F1 Score	Silhouette	NN	Accuracy (%)	F1 Score	Silhouette
Supervised Model	7	99.73	0.9973	0.7899	7	30.094	0.393	0.4514
SE_ResNeXt_32	3	62.938	0.6278	-0.0066	3	22.96	0.2786	0.6747
SE_ResONeXt_32_q3	3	53.015	0.5286	-0.0565	5	15.473	0.2097	0.4312
SE_ResNeXt_128	3	63.548	0.6342	-0.01172	3	14.122	0.1875	0.6509
SE_ResONext_128_q3	3	51.398	0.513	-0.05949	3	17.645	0.214	0.6374

Results presented in this table are from models trained on the Self-Supervised Track dataset and evaluated on the CowID-1785 dataset along with the small portion of labelled barn data.
The reference embeddings for both evaluation sets are generated from the CowID-1785 dataset. Therefore, the barn identification data is being compared against embeddings from the milking parlour domain.

6.4.3 Discussion

The results show that the performance of the self-supervised models drops considerably compared to the supervised model on the CowID-1785 dataset. However, this is expected as the supervised model has been trained specifically for this task and from data within the same domain, so the performance of the self-supervised models is still fairly good on this dataset considering the models are not trained with strong labels or data from this domain. Interestingly, the barn identification performance also drops considerably with the self-supervised models when compared to the supervised model. This is likely due to the lack of knowledge of the milking parlour domain since images from the milking parlour are used to produce the reference embeddings for the KNN classifier and the model struggles to embed the barn images in regions close to the correct milking parlour images for classification. The supervised model has the opposite problem, where the lack of knowledge of the barn domain also makes it hard to embed the barn images close to the correct milking parlour images. Though, as the results show, it is slightly more effective at doing so than the self-supervised methods. This is likely due to the supervised milking parlour dataset containing far greater positive pair variation since the data is captured across several different days, encapsulating many different conditional variations such as lighting conditions and cow positions. While the barn datasets are also captured across several days, positive pairs can only be formed across a very narrow window of time as they must be formed from the same short video clip which will not capture much variation in the cow's position or the lighting conditions. Significantly extending the clip length would likely improve performance, but would require the additional capability to be able to track cows between cameras, which is left for future work. However, the self-supervised methods do improve the clustering performance on the barn domain, as evidenced by the increase in barn identification silhouette score when compared to the supervised model (with the exception of the SE_ResONeXt_32_q3 model), indicating stronger discrimination power in this domain, although it still fails to match the embeddings to the milking parlour domain. The poor silhouette score of the supervised model in the barn domain compared to the barn models is probably due to the absence of barn data during supervised training, demonstrating the value of using barn data in training.

Performance on the CowID-1785 dataset improves significantly compared to the first self-supervised method that does not take the tracking information into account, which demonstrates the value of using real positive pairs for training instead of relying on augmented views.

The main weakness with the methods presented is that there is no cross-domain supervision, meaning that there is no loss constraint to enforce learning of the desired objective to match embeddings from the barn domain to the milking parlour domain for identification. This absence of cross-domain supervision makes it very difficult for the model to learn the desired objective and consequently results in no barn identification performance improvement over models trained in a supervised fashion on only milking parlour domain data, which suffer the same problem. To overcome this issue, it is likely necessary to acquire properly labelled barn data. Despite the lack of improvement in barn identification performance, the relatively high milking parlour identification performance suggests that the proposed self-supervised methods may make a good pretraining step for milking parlour identification models.

Both Self-ONN models perform significantly worse than their CNN equivalents on the CowID-1785 dataset. This is potentially due to their more complex non-linear filters producing a more highly tailored function for the training domain that does not transfer over to different domains as easily as the simpler linear filters of a CNN. The results between Self-ONN and CNN are more mixed, where the smaller embedding CNN model performs best, though both Self-ONN models perform better than the larger embedding CNN model. However, the small size of this dataset combined with the great difficulty of matching embeddings across domains for the evaluation of this dataset likely introduces a lot of variance into these results. Since the CNN models have a higher average performance on this dataset than the Self-ONN models, this suggests that the CNN models are better suited to this task, though performance is still poor across both types of models, again likely due to the absence of cross-domain supervision.

6.5 Summary

This chapter explored the use of self-supervised training techniques to address the problem of identifying individual cows in a challenging barn environment. Traditional supervised training using milking parlour images proved ineffective in the barn domain due to a significant domain gap, as evidenced by a notable drop in identification accuracy. To address this issue, two novel self-supervised training pipelines were proposed that leverage the unique properties of barn data. More specifically, the proposed self-supervised techniques exploit weak labels obtained from time-synchronised detections and tracking information to improve training efficacy.

The first method used temporal information to ensure negative pair sampling, combined with standard augmentation to generate positive pairs, enabling effective similarity learning without the need for a teacher-student framework. Although this approach showed promising improvements over existing methods such as BYOL, its ability to learn intra-class variation was limited due to its reliance on augmented positives alone. The second approach addressed this issue by introducing tracking data to form real positive pairs, which significantly improved performance on milking parlour data and enhanced clustering within the barn domain. However, neither method yielded an improvement in cross-domain barn identification performance compared to supervised models trained solely on milking parlour data. This is likely due to the absence of a cross-domain constraint in the objective function, which is also lacking in the supervised milking parlour models.

Overall, while the proposed self-supervised methods demonstrate strong potential and enable fully automated dataset collection and training, their effectiveness in barn identification remains constrained by the lack of a cross-domain objective during training. The methods offer a valuable cow identification pretraining step due to the automated nature of data gathering, making them easily scalable, and they significantly outperform existing self-supervised techniques. An interesting direction for future research would be to integrate data from both barn and milking parlour domains and introduce a cross-domain constraint in the objective function to bridge the domain gap more explicitly.

7 Conclusions

The aim of this thesis was to develop novel techniques to enhance deep learning performance in various challenging image processing applications. Significant contributions were made to several aspects of deep learning, including data preprocessing, model architecture, and training algorithms, specifically applied to cow identification and Hyperspectral Image Super-Resolution, offering substantial advancements in each area.

Chapter 4 explored the use of similarity learning to train cow identification models. The properties of embeddings models produced by this technique were leveraged to perform classification on new classes. Novel analysis was performed to evaluate the models' performance on these new classes, simulating the scenario where new cows are introduced to a herd. In this Chapter, Self-ONN models were proposed to improve the parameter efficiency of the identification models by extending each filter to more powerful non-linear variants, significantly reducing the number of filters required to achieve similar performance. Novel Self-ONN models were also proposed in Chapter 5 to improve HSI-SR performance in a number of scenarios. Additional preprocessing techniques were proposed to further improve performance and analysis was conducted on a novel dataset gathered in collaboration with other researchers. In Chapter 6 novel self-supervised techniques were proposed to exploit weak labels present within data extracted from dairy cattle barn video footage to address the issue of barn identification. Two variants were explored where the first used an object detection model to generate weak labels and greatly improve self-supervised training performance over the popular BYOL technique, and a second variant which incorporated tracking information was also proposed to further improve performance.

This chapter draws several conclusions about the work presented in this thesis and suggests several directions for future research.

7.1 Image Classification with Similarity Learning

Chapter 4 focused on supervised learning approaches for cow identification in the relatively controlled milking parlour setting, where it is easily feasible to acquire labelled data. The results demonstrated exceptional performance in this setting, particularly when evaluating on the same cows used for training. Novel new class analysis was conducted on a reserved group of cows not present within the training set to simulate the scenario of new cows being introduced to a herd and examine how a production identification model might perform on these cows without retraining. Performance was found to be high on new cows the model has not been trained on, though a slight drop in performance was observed when compared to identification performance on cows present within the training set. This finding reveals that it is not necessary to retrain the identification model every single time a new cow is introduced to a herd, making the deployment of the identification models presented in this thesis much more practical than models trained with more traditional classification techniques which would necessitate retraining after every new cow introduced.

In addition to retraining frequency, another important practical consideration for identification models is the computational cost per identification, particularly in large barns. To address this consideration, various Self-ONN models were also proposed in Chapter 4. Self-ONN models extend the standard linear filters of a CNN to learnable non-linear function approximators, increasing their theoretical ability to learn complex non-linear functions. While this additional non-linear complexity introduces additional network parameters per filter, it was found that the overall number of filters the network requires can be reduced to achieve similar performance to an equivalent CNN identification model. The reduction in filters meant that the proposed Self-ONN models contained ∼30% less parameters and consequently this reduces the computation cost each time identification is performed.

7.2 Image Enhancement with Non-Linear Filters and Improved Preprocessing

Chapter 5 focused on exploring techniques for the challenging application area of HSI-SR, proposing several different techniques to address the various challenges associated with this area on a variety of datasets. Building off the parameter-efficient Self-ONN models proposed in Chapter 4, novel Self-ONN models were also proposed for this task, improving both parameter-efficiency as well as SR performance. First, a novel Self-ONN variant of the popular SRCNN model was proposed for use on small HSI datasets. The proposed SRONN model was found to consistently outperform the SRCNN model, especially when a global residual connection was incorporated into the model. However, due to the increased non-linear complexity of the proposed SRONN model, the model contained significantly more parameters than its SRCNN counterpart. To address this issue, another variant of this model, named sSRONN, was proposed, which contained four times fewer filters per layer than the initial SRONN and consequently contained significantly fewer parameters than the base SRCNN model. It was found that the proposed sSRONN model also consistently outperformed the SRCNN model and in some cases even outperformed the full SRONN model, demonstrating the power of the Self-Operational layers used within the models. Various normalisation layers were also incorporated into the proposed models to try to further performance, though this was found to have little effect.

Experiments were then carried out on the much larger ICONES dataset, although this dataset introduced new challenges due to the size and complexity of the dataset. Therefore, novel preprocessing algorithms were proposed to address these challenges. Several novel algorithms were proposed that incorporated the standard normal variate – a normalisation technique commonly used to normalise data between DNN layers – and provide small amounts of outlier removal to the data. A broad range of models were selected for experimentation, including a novel Self-ONN variant of the FRSCNN model, named FSRONN. With no normalisation techniques applied to the data, significant performance gaps were observed between the various models. However, when the proposed normalisation techniques were applied, it was found that not only does each model see a performance gain, but also the performance gap between the models was significantly reduced, indicating that the complexity of the problem space is significantly reduced when using the proposed normalisation techniques and by doing so, less powerful model architectures are required to achieve better performance. Furthermore, it was also found to be more effective to normalise each band in a given patch individually rather than normalising the entire patch. This indicates that band-wise normalisation better handles the data variations between bands, improving the quality and consistency of the data, and making it easier for the models to extract the necessary features from the data.

The previously mentioned experiments in Chapter 5 were conducted on publicly available HSI datasets, where low-resolution training pairs were required to be synthetically generated. However, synthetic generation of low-resolution pairs imposes several assumptions on the complex real high- to low-resolution relationship – something that has been shown to be detrimental to performance in the RGB-SR field. To address the limitations of training HSI-SR models on synthetically downsampled data, a novel HSI dataset was gathered in collaboration with two other researchers, which consisted of real high- and low-resolution HSI pairs for experimentation. Experiments were conducted by training models both on the real high- and low-resolution image pairs, but also by synthetically downsampling the high-resolution image to simulate the low-resolution generation process commonly used in the field. All experiments were evaluated using the real high- and low-resolution image pairs to evaluate true super-resolution performance and it was found that models trained using synthetic downsampling processes failed to outperform simple bicubic interpolation when applied to real data in the majority of cases. However, when training using real data pairs, models significantly outperform bicubic interpolation, revealing the importance of training with real data. For the experiments, a broad range of models were used including deep CNN architectures in addition to the shallow SRONN model proposed earlier in the

Chapter. It was found that the proposed SRONN model outperformed other larger and more complex CNN models on the noise-free lens datasets, again demonstrating the power of the non-linear filters within Self-ONN models. Interestingly though, the other experimental models were found to outperform the proposed SRONN model on the sensor dataset. This is likely due to the higher noise levels present within the low-resolution sensor making it more challenging for the shallower SRONN model to extract and map the features from the low-resolution sensor to the high-resolution sensor compared to the other deeper CNN models. Furthermore, the CNN models used in the experiments contained 3D filters which perhaps allowed them to more easily extract features from the noisy data than the 2D SRONN model, though further analysis is required to draw any firm conclusions on this. However, although the SRONN model did not preform as well in this case, it offers a significant performance improvement in the case where the low- and high-resolution images contain the same noise characteristics, despite only operating in two dimensions and being architecturally simpler than the comparison CNN models.

7.3 Self-Supervised Cow Identification

The cow identification models developed in Chapter 4 perform extremely well when identifying cattle in the milking parlour domain – the same domain used to train the models. However, it is also valuable to agritech applications to be able to perform identification in a free moving barn. It was found that the identification models trained on milking parlour domain data were not transferable to the barn domain due the increased complexity of the barn domain. Furthermore, barn domain data cannot be labelled using RFID scanners, making acquiring labelled barn data for training practically infeasible. Chapter 6 addressed these challenges by proposing novel self-supervised algorithms to take advantage of the large quantities of unlabelled data available. The proposed methods leverage the inherent weak labels resulting from the proposed data acquisition processes to produce triplet combinations. The first unsupervised algorithm proposed leveraged an object detection model to extract cow detections from the barn camera footage and worked off the assumption that all detections at a given instance in time were unique cows and could therefore be used to form negative pairs for the triplet loss function. Positive pairs were formed by augmenting each detection, similar to the popular BYOL algorithm. The proposed algorithm was found to significantly outperform the comparison BYOL algorithm due to its significantly improved discriminative ability resulting from the negative pairs used in training. However, it was also found that performance begins to plateau with this algorithm as more training data is added, which was hypothesised to be due to the lack of variation captured in the augmented positive pairs. To address this limitation, a second variant of the algorithm was proposed which incorporates a tracking model to match cow detections between frames in short video footage, allowing for positive pairs to be formed using multiple frames from the same detection and not having to rely on augmentation. This variant of the algorithm was found to boost identification accuracy by over 10%.

Experiments were also conducted on the Self-ONN models proposed in Chapter 4. However, it was found that these models performed significantly worse than their CNN equivalents in this scenario. This is likely because the non-linear filters caused the Self-ONN models to be more tailored to the barn training domain and less generalisable to the milking parlour domain used to generate the test metrics. Since the CNN filters are simpler, they are likely slightly better suited to making the domain jump to boost evaluation performance, though further research is required to draw any firm conclusions on this.

The self-supervised algorithms were found to significantly outperform the BYOL self-supervised method, which does not take advantage of the weak labels present within the proposed data extraction methods. It was found that incorporating the tracking information into this algorithm provides a significant boost to performance on the milking parlour dataset over the method that uses detections alone. However, it was found that barn identification does not improve with this method over the supervised models from Chapter 4. This may be due to the very limited size of the barn identification test set, though it is most likely due to the absence of cross-domain supervision, making it challenging to learn the desired objective. Despite this, the proposed algorithms show promise on the milking parlour evaluation, suggesting that they may make a valuable scalable pretraining step for supervised identification models.

7.4 Further Work

Despite the fact that the results presented in Chapter 4 may make it seem like cow identification in a milking parlour environment is a solved problem, there is still potential for further optimisations. One such example is the model size. The fact that the models using smaller embedding sizes presented in Chapter 4 outperformed the models with larger embedding sizes, particularly on new cows, suggests that reducing the model size or complexity may provide generalisation benefits and may even improve cross-domain performance. Furthermore, another interesting direction for research would be to explore augmentation techniques to improve cross-domain performance, potentially allowing for milking parlour models to be transferrable to the barn domain, which also relates nicely to the work presented in Chapter 6.

An empirical observation made throughout the work conducted in this thesis is that Self-ONN models are generally more challenging to train and require more careful hyperparameter tuning. Another valuable direction for future work would be to address this increased training difficulty, perhaps through the use of regularisation terms or more advanced optimisers. Additionally, it would be interesting to explore the use of 3 dimensional Self-ONN filters for HSI-SR models to see if this offers performance improvements to Self-ONN in the same way that 3D filters offer improvement to CNN models. Of course, this comes with memory consumption challenges, and so may necessitate novel computation methods for such 3D Self-ONN layers.

Although the proposed self-supervised algorithms in Chapter 6 offer a valuable pretraining step to train identification models using easily attainable large quantities of unlabelled data, there is still a great deal of room for improvement when it comes to barn identification via similarity measurements between barn images and milking parlour images. The proposed self-supervised algorithms do not exploit the milking parlour data available, so a promising direction for future work would be to leverage data from both domains during training using domain adaptation techniques. Though, of course, the obvious challenge with this is acquiring the labels for the barn domain. Therefore, a potentially promising route would be to explore a human in the loop approach which attempts to predict IDs from the unlabelled barn data while training on the labelled milking parlour data and have these predictions verified by the human. When a correct prediction is verified, the barn data can be added to the training data which in turn would improve the barn ID performance and result in more correct barn predictions being verified and added to the training data, positively feeding back and improving performance over time. Another promising avenue for performance improvement would be to encapsulate more variation in the data by significantly extending the tracking time for individual cattle. This would require the development of a cross-camera tracking algorithm to be able to track the animal as it moves across the entire barn instead of within a single camera as is currently done in the existing algorithm.

7.5 Final Remarks

In summary, this thesis has introduced novel methodologies that advance the state-of-the-art in deep learning for challenging image processing tasks, with particular emphasis on cattle identification and HSI-SR. Across the diverse domains explored, this work has demonstrated how innovations in Self-ONN architectures, data normalisation strategies, and self-supervised learning can yield practical and efficient solutions to problems that have direct scientific and real-world impact. While several avenues for improvement and extension remain, the contributions made here establish a solid foundation upon which future research can build, not just within the agricultural technology domain and HSI-SR, but also in broader applications of deep learning. Ultimately, the findings of this thesis highlight the value of bridging methodological innovation with application-driven challenges, and they provide a pathway towards more robust, scalable, and transferable machine learning systems.

References

[1] O. Ben-Ahmed, T. Urruty, N. Richard, and C. Fernandez-Maloigne, "Toward content-based hyperspectral remote sensing image retrieval (cb-hrsir): A preliminary study based on spectral sensitivity functions," *Remote Sensing*, vol. 11, no. 5, p. 600, 2019.

[2] D. Berckmans, "Precision livestock farming technologies for welfare management in intensive livestock systems," *Rev. Sci. Tech*, vol. 33, no. 1, pp. 189–196, 2014.

[3] M. G. Cappai, N. G. Rubiu, G. Nieddu, M. Bitti, and W. Pinna, "Analysis of fieldwork activities during milk production recording in dairy ewes by means of individual ear tag (et) alone or plus rfid based electronic identification (eid)," *Computers and Electronics in Agriculture*, vol. 144, pp. 324–328, 2018.

[4] A. I. Awad, "From classical methods to animal biometrics: A review on cattle identification and tracking," *Computers and Electronics in Agriculture*, vol. 123, pp. 423–435, 2016.

[5] A. K. Singh, S. Ghosh, B. Roy, D. K. Tiwari, and R. Baghel, "Application of radio frequency identification (rfid) technology in dairy herd management," *Int. J. Livestock Res*, vol. 4, pp. 10–19, 2014.

[6] S. Kumar, S. K. Singh, R. S. Singh, A. K. Singh, and S. Tiwari, "Real-time recognition of cattle using animal biometrics," *Journal of Real-Time Image Processing*, vol. 13, pp. 505–526, 2017.

[7] S. Kumar, S. Tiwari, and S. K. Singh, "Face recognition of cattle: can it be done?," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 86, pp. 137–148, 2016.

[8] W. Kusakunniran and T. Chaiviroonjaroen, "Automatic cattle identification based on multi-channel lbp on muzzle images," in *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, pp. 1–5, IEEE, 2018.

[9] "Education - dairy cows." Holstein UK `https://ukcows.com/holsteinUK/publicweb/Education/HUK_Edu_DairyCows.aspx?cmh=66#:~:text=Holsteins%20are%20large%20cattle%20with,high%20volumes%20of%20milk%20efficiently.`

[10] "Hyperspectral remote sensing." The University of Texas at Austin `http://www.csr.utexas.edu/projects/rs/hrs/hyper.html` (accessed Apr. 5, 2022).

[11] K. Berger, J. Verrelst, J.-B. Féret, Z. Wang, M. Wocher, M. Strathmann, M. Danner, W. Mauser, and T. Hank, "Crop nitrogen monitoring: Recent progress and principal developments in the context of imaging spectroscopy missions," *Remote Sensing of Environment*, vol. 242, p. 111758, 2020.

[12] B. Zhang, L. Zhao, and X. Zhang, "Three-dimensional convolutional neural network model for tree species classification using airborne hyperspectral images," *Remote Sensing of Environment*, vol. 247, p. 111938, 2020.

[13] A. G. Villafranca, J. Corbera, F. Martín, and J. F. Marchán, "Limitations of hyperspectral earth observation on small satellites," *Journal of Small Satellites*, vol. 1, no. 1, pp. 19–29, 2012.

[14] D. J. Brady, *Optical imaging and spectroscopy.* John Wiley & Sons, 2009.

[15] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1905–1914, 2021.

[16] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[17] W. Liu and J. Lee, "An efficient residual learning neural network for hyperspectral image superresolution," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 4, pp. 1240–1253, 2019.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[19] 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.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[26] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun, "Convit: Improving vision transformers with soft convolutional inductive biases," in *International conference on machine learning*, pp. 2286–2296, PMLR, 2021.

[27] L. Deininger, B. Stimpel, A. Yuce, S. Abbasi-Sureshjani, S. Schönenberger, P. Ocampo, K. Korski, and F. Gaire, "A comparative study between vision transformers and cnns in digital pathology," *arXiv preprint arXiv:2206.00389*, 2022.

[28] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

[29] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[32] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *International Conference on Learning Representations*, 2020.

[33] H. Sun, L. Shen, Q. Zhong, L. Ding, S. Chen, J. Sun, J. Li, G. Sun, and D. Tao, "Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks," *Neural Networks*, vol. 169, pp. 506–519, 2024.

[34] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[35] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE, 2005.

[36] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[37] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in neural information processing systems*, vol. 29, 2016.

[38] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5022–5030, 2019.

[39] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

[40] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE international conference on computer vision*, pp. 360–368, 2017.

[41] E. W. Teh, T. DeVries, and G. W. Taylor, "Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pp. 448–464, Springer, 2020.

[42] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238–3247, 2020.

[43] Z. Yang, M. Bastan, X. Zhu, D. Gray, and D. Samaras, "Hierarchical proxy-based loss for deep metric learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1859–1868, 2022.

[44] M. S. Mahmud, A. Zahid, A. K. Das, M. Muzammil, and M. U. Khan, "A systematic literature review on deep learning applications for precision cattle farming," *Computers and Electronics in Agriculture*, vol. 187, p. 106313, 2021.

[45] H. Minagawa, T. Fujimura, M. Ichiyanagi, K. Tanaka, *et al.*, "Identification of beef cattle by analyzing images of their muzzle patterns lifted on paper.," in *AFITA 2002: Asian agricultural information technology & management. Proceedings of the Third Asian Conference for Information Technology in Agriculture, Beijing, China, 26-28 October, 2002*, pp. 596–600, China Agricultural Scientech Press, 2002.

[46] T. T. Zin, M. Z. Pwint, P. T. Seint, S. Thant, S. Misawa, K. Sumi, and K. Yoshida, "Automatic cow location tracking system using ear tag visual analysis," *Sensors*, vol. 20, no. 12, p. 3564, 2020.

[47] L. Yao, Z. Hu, C. Liu, H. Liu, Y. Kuang, and Y. Gao, "Cow face detection and recognition based on automatic feature extraction algorithm," in *Proceedings of the ACM turing celebration conference-china*, pp. 1–5, 2019.

[48] Y. Kawagoe, I. Kobayashi, and T. T. Zin, “Facial region analysis for individual identification of cows and feeding time estimation,” *Agriculture*, vol. 13, no. 5, p. 1016, 2023.

[49] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[50] A. I. Awad and M. Hassaballah, “Bag-of-visual-words for cattle identification from muzzle print images,” *Applied Sciences*, vol. 9, no. 22, p. 4914, 2019.

[51] R.-W. Bello, A. Z. H. TALIB, and A. S. A. B. MOHAMED, “Deep learning-based architectures for recognition of cow using cow nose image pattern,” *Gazi University Journal of Science*, vol. 33, no. 3, pp. 831–844, 2020.

[52] F. de Lima Weber, V. A. de Moraes Weber, G. V. Menezes, A. d. S. O. Junior, D. A. Alves, M. V. M. de Oliveira, E. T. Matsubara, H. Pistori, and U. G. P. de Abreu, “Recognition of pantaneira cattle breed using computer vision and convolutional neural networks,” *Computers and Electronics in Agriculture*, vol. 175, p. 105548, 2020.

[53] R. Zhang, J. Ji, K. Zhao, J. Wang, M. Zhang, and M. Wang, “A cascaded individual cow identification method based on deepotsu and efficientnet,” *Agriculture*, vol. 13, no. 2, p. 279, 2023.

[54] J. Xiao, G. Liu, K. Wang, and Y. Si, “Cow identification in free-stall barns based on an improved mask r-cnn and an svm,” *Computers and Electronics in Agriculture*, vol. 194, p. 106738, 2022.

[55] W. Andrew, J. Gao, S. Mullan, N. Campbell, A. W. Dowsey, and T. Burghardt, “Visual identification of individual holstein-friesian cattle via deep metric learning,” *Computers and Electronics in Agriculture*, vol. 185, p. 106133, 2021.

[56] Y. Qiao, D. Su, H. Kong, S. Sukkarieh, S. Lomax, and C. Clark, "Bilstm-based individual cattle identification for automated precision livestock farming," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 967–972, IEEE, 2020.

[57] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 391–407, Springer, 2016.

[58] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[60] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[61] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.

[62] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1833–1844, 2021.

[63] Z. Lu, J. Li, H. Liu, C. Huang, L. Zhang, and T. Zeng, “Transformer for single image super-resolution,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 457–466, 2022.

[64] A. Mehri, P. Behjati, and A. D. Sappa, “Tntvit-g: Transformer in transformer network for guidance super resolution,” *IEEE Access*, vol. 11, pp. 11529–11540, 2023.

[65] J. Jiang, H. Sun, X. Liu, and J. Ma, “Learning spatial-spectral prior for super-resolution of hyperspectral imagery,” *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1082–1096, 2020.

[66] C. Zhang, M. Zhang, Y. Li, X. Gao, and S. Qiu, “Difference curvature multidimensional network for hyperspectral image super-resolution,” *Remote Sensing*, vol. 13, no. 17, p. 3455, 2021.

[67] L. Loncan, L. B. De Almeida, J. M. Bioucas-Dias, X. Briottet, J. Chanussot, N. Dobigeon, S. Fabre, W. Liao, G. A. Licciardi, M. Simoes, *et al.*, “Hyperspectral pansharpening: A review,” *IEEE Geoscience and remote sensing magazine*, vol. 3, no. 3, pp. 27–46, 2015.

[68] Y. Zheng, J. Li, Y. Li, J. Guo, X. Wu, and J. Chanussot, “Hyperspectral pansharpening using deep prior and dual attention residual network,” *IEEE transactions on geoscience and remote sensing*, vol. 58, no. 11, pp. 8059–8076, 2020.

[69] S. Li, R. Dian, L. Fang, and J. M. Bioucas-Dias, “Fusing hyperspectral and multispectral images via coupled sparse tensor factorization,” *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4118–4130, 2018.

[70] Y. Yuan, X. Zheng, and X. Lu, “Hyperspectral image superresolution by transfer learning,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 5, pp. 1963–1974, 2017.

[71] S. Mei, X. Yuan, J. Ji, Y. Zhang, S. Wan, and Q. Du, "Hyperspectral image spatial super-resolution via 3d full convolutional neural network," *Remote Sensing*, vol. 9, no. 11, p. 1139, 2017.

[72] J. Li, R. Cui, B. Li, R. Song, Y. Li, Y. Dai, and Q. Du, "Hyperspectral image super-resolution by band attention through adversarial learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 4304–4318, 2020.

[73] P. V. Arun, K. M. Buddhiraju, A. Porwal, and J. Chanussot, "Cnn-based super-resolution of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 9, pp. 6106–6121, 2020.

[74] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[75] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

[76] B. Wang, Y. Zhang, Y. Feng, B. Xie, and S. Mei, "Attention-enhanced generative adversarial network for hyperspectral imagery spatial super-resolution," *Remote Sensing*, vol. 15, no. 14, p. 3644, 2023.

[77] Y. Liu, J. Hu, X. Kang, J. Luo, and S. Fan, "Interactformer: Interactive transformer and cnn for hyperspectral image super-resolution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[78] M. Zhang, C. Zhang, Q. Zhang, J. Guo, X. Gao, and J. Zhang, "Essaformer: Efficient transformer for hyperspectral image super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23073–23084, 2023.

[79] S. Kiranyaz, T. Ince, A. Iosifidis, and M. Gabbouj, "Operational neural networks," *Neural Computing and Applications*, vol. 32, no. 11, pp. 6645–6668, 2020.

[80] J. Malik, S. Kiranyaz, and M. Gabbouj, "Fastonn–python based open-source gpu implementation for operational neural networks," *arXiv preprint arXiv:2006.02267*, 2020.

[81] S. Kiranyaz, T. Ince, A. Iosifidis, and M. Gabbouj, "Generalized model of biological neural networks: progressive operational perceptrons," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2477–2485, IEEE, 2017.

[82] S. Kiranyaz, J. Malik, H. B. Abdallah, T. Ince, A. Iosifidis, and M. Gabbouj, "Self-organized operational neural networks with generative neurons," *Neural Networks*, vol. 140, pp. 294–308, 2021.

[83] M. Gabbouj, S. Kiranyaz, J. Malik, M. U. Zahid, T. Ince, M. E. Chowdhury, A. Khandakar, and A. Tahir, "Robust peak detection for holter ecgs by self-organized operational neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[84] J. Malik, O. C. Devecioglu, S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1d self-operational neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 5, pp. 1788–1801, 2021.

[85] O. C. Devecioglu, J. Malik, T. Ince, S. Kiranyaz, E. Atalay, and M. Gabbouj, "Real-time glaucoma detection from digital fundus images using self-onns," *IEEE Access*, vol. 9, pp. 140031–140041, 2021.

[86] M. U. Zahid, S. Kiranyaz, and M. Gabbouj, "Global ecg classification by self-operational neural networks with feature injection," *IEEE Transactions on Biomedical Engineering*, vol. 70, no. 1, pp. 205–215, 2022.

[87] T. Ince, S. Kiranyaz, O. C. Devecioglu, M. S. Khan, M. Chowdhury, and M. Gabbouj, “Blind restoration of real-world audio by 1d operational gans,” *arXiv preprint arXiv:2212.14618*, 2022.

[88] J. Malik, S. Kiranyaz, M. Yamac, and M. Gabbouj, “Bm3d vs 2-layer onn,” in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1994–1998, IEEE, 2021.

[89] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[90] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” *Advances in neural information processing systems*, vol. 33, pp. 22243–22255, 2020.

[91] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, “Bootstrap your own latent-a new approach to self-supervised learning,” *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[92] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, “Time-contrastive networks: Self-supervised learning from video,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[93] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

[94] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[95] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong, "ibot: Image bert pre-training with online tokenizer," *arXiv preprint arXiv:2111.07832*, 2021.

[96] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

[97] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas, "Self-supervised learning from images with a joint-embedding predictive architecture," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

[98] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[99] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*, pp. 1989–1998, Pmlr, 2018.

[100] Q. Liu, Q. Dou, and P.-A. Heng, "Shape-aware meta-learning for generalizing prostate mri segmentation to unseen domains," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*, pp. 475–485, Springer, 2020.

[101] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617, IEEE, 2021.

[102] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, “Unified deep supervised domain adaptation and generalization,” in *Proceedings of the IEEE international conference on computer vision*, pp. 5715–5725, 2017.

[103] Z. Lu, Y. Yang, X. Zhu, C. Liu, Y.-Z. Song, and T. Xiang, “Stochastic classifiers for unsupervised domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9111–9120, 2020.

[104] Z. Liu, Z. Miao, X. Pan, X. Zhan, D. Lin, S. X. Yu, and B. Gong, “Open compound domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12406–12415, 2020.

[105] Y. Zhao, Z. Zhong, F. Yang, Z. Luo, Y. Lin, S. Li, and N. Sebe, “Learning to generalize unseen domains via memory-based multi-source meta-learning for person re-identification,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6277–6286, 2021.

[106] Z. Xu, D. Liu, J. Yang, C. Raffel, and M. Niethammer, “Robust and generalizable visual representation learning via random convolutions,” in *International Conference on Learning Representations*, 2020.

[107] S. Choi, D. Das, S. Choi, S. Yang, H. Park, and S. Yun, “Progressive random convolutions for single domain generalization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10312–10322, 2023.

[108] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, “Mixstyle neural networks for domain generalization and adaptation,” *International Journal of Computer Vision*, pp. 1–15, 2023.

[109] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization.,” *Journal of machine learning research*, vol. 13, no. 2, 2012.

[110] "Hyperspectral remote sensing scenes." University of the Basque Country `https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes` (accessed Mar. 16, 2022).

[111] "Remote sensing datasets." Remote Sensing Laboratory School of Surveying and Geospatial Engineering `https://rslab.ut.ac.ir/data` (accessed Mar. 4, 2022).

[112] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *electronics*, vol. 8, no. 3, p. 292, 2019.

[113] D. B. Bhushan, V. Sowmya, M. S. Manikandan, and K. Soman, "An effective preprocessing algorithm for detecting noisy spectral bands in hyperspectral imagery," in *2011 International symposium on ocean electronics*, pp. 34–39, IEEE, 2011.

[114] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3," Sep 2015.

[115] J. Cai, H. Zeng, H. Yong, Z. Cao, and L. Zhang, "Toward real-world single image super-resolution: A new benchmark and a new model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3086–3095, 2019.

[116] K. Zhang, J. Liang, L. Van Gool, and R. Timofte, "Designing a practical degradation model for deep blind image super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4791–4800, 2021.

[117] J.-F. Hu, T.-Z. Huang, L.-J. Deng, T.-X. Jiang, G. Vivone, and J. Chanussot, "Hyperspectral image super-resolution via deep spatiospectral attention convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7251–7265, 2021.

[118] L. Wang, T. Bi, and Y. Shi, "A frequency-separated 3D-CNN for hyperspectral image super-resolution," *IEEE Access*, vol. 8, pp. 86367–86379, 2020.

[119] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[120] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[121] T.-A. Song, S. R. Chowdhury, K. Kim, K. Gong, G. El Fakhri, Q. Li, and J. Dutta, "Super-resolution pet using a very deep convolutional neural network," in *2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*, pp. 1–2, IEEE, 2018.

[122] F. Cao, Z. Yang, J. Ren, M. Jiang, and W.-K. Ling, "Does normalization methods play a role for hyperspectral image classification?," *arXiv preprint arXiv:1710.02939*, 2017.

[123] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[124] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[125] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[126] S. Kodama, I. Takeda, and Y. Yamaguchi, "Mapping of hydrothermally altered rocks using the modified spectral angle mapper (msam) method and aster swir data.," *International Journal of Geoinformatics*, vol. 6, no. 1, 2010.

[127] A. Ulrichsen, T. De Kerf, D. Dunphy, P. Murray, S. Vanlanduit, and S. Marshall, "A true hyperspectral image super-resolution dataset," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Workshops*, pp. 4412–4421, June 2025.

[128] L. Wald, *Data fusion: definitions and architectures: fusion of images of different spatial resolutions*. Presses des MINES, 2002.

[129] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023.

[130] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[131] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[132] A. C. Li, A. A. Efros, and D. Pathak, "Understanding collapse in non-contrastive siamese representation learning," in *European Conference on Computer Vision*, pp. 490–505, Springer, 2022.

[133] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2018.

Publications by the Author

- A. Ulrichsen, P. Murray, S. Marshall, M. Gabbouj, S. Kiranyaz, M. Yamaç, and N. Abu-raed, “Operational neural networks for parameter-efficient hyperspectral single-image super-resolution,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2023.

- A. Ulrichsen, P. Murray, S. Marshall, B. Lee, and M. Rutter, “Cow identification network trained with similarity learning,” in *European Conference on Precision Livestock Farming*, 2022.

- A. Ulrichsen, P. Murray, S. Marshall, B. Lee, and M. Rutter, “Camera-based automatic cow identification using deep neural networks,” in *The 55th Congress of the International Society of Applied Ethology — Animal Behaviour and Beyond*, 2022.

- A. Gilmour, A. Ulrichsen, W. Jackson, M. Tabatabaeipour, G. Dobie, C. N. Macleod, P. Murray, and B. Karkera, “Using phased array ultrasound to localize probes during the inspection of welds,” *IEEE Open Journal of Instrumentation and Measurement*, 2023.

- T. De Kerf, A. Ulrichsen, P. Scheunders, P. Murray, and S. Vanlanduit, “A hyperspectral super-resolution dataset for the validation of super-resolution methods,” in *IEEE Whispers*, 2023.

- A. Ulrichsen, T. De Kerf, D. Dunphy, P. Murray, S. Vanlanduit, and S. Marshall, “A true hyperspectral image super-resolution dataset,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2025, pp. 4412–4421.

Appendices

Full Cow Identification Results

LR	LR Step	Accum Iter	Triplet Margin	Hardest Triplets	Accuracy (%)	F1 Score	New Class Accuracy (%)	New Class F1 Score
0.000717	66	182	0.2	TRUE	1.27	0.01	0.54	0.009
0.000126	131	14	0.5	TRUE	1.73	0.009	0.54	0.006
0.000826	150	81	4	FALSE	15.53	0.135	7.03	0.107
0.000314	82	8	3	TRUE	2.13	0.016	2.7	0.042
0.000749	947	3	1	TRUE	97.67	0.975	81.08	0.861
0.000362	252	66	5	TRUE	2.73	0.02	2.7	0.042
0.000539	47	2	0.5	FALSE	83.27	0.828	76.76	0.821
0.000172	457	228	0.1	TRUE	1.33	0.008	0	0
0.000313	48	216	0.2	FALSE	28.8	0.268	29.19	0.375
0.000577	655	2	2	TRUE	90.27	0.903	91.35	0.934
0.00048	92	1	5	FALSE	14.2	0.121	7.03	0.105
0.000705	206	36	4	TRUE	19.33	0.176	12.43	0.171
0.000199	24	171	5	FALSE	3.73	0.032	3.78	0.055
0.000773	20	40	0.1	FALSE	18.8	0.168	22.7	0.293
0.000838	32	1	2	FALSE	8.53	0.074	7.57	0.113
0.000953	26	207	0.1	TRUE	2.2	0.014	3.24	0.049
0.000551	256	9	4	FALSE	21.33	0.193	8.65	0.134
0.000004	22	8	0.2	FALSE	1.73	0.014	0.54	0.008
0.000616	155	62	1	FALSE	55.13	0.536	41.08	0.501
0.000194	153	38	0.1	TRUE	1.13	0.006	0	0
0.000923	20	3	0.1	FALSE	80.27	0.795	74.05	0.802
0.000169	93	2	0.1	FALSE	94.13	0.94	89.73	0.931
0.000236	123	6	4	FALSE	12.27	0.111	4.32	0.072
0.000158	208	6	2	FALSE	13.93	0.12	6.49	0.091
0.000023	41	5	3	FALSE	2.73	0.018	0.54	0.008
0.000636	568	45	0.2	FALSE	98.2	0.981	90.81	0.922
0.000092	122	8	0.1	FALSE	87.47	0.872	85.95	0.894
0.000286	36	19	0.2	TRUE	0.53	0.003	1.62	0.027
0.000894	33	4	4	FALSE	9.8	0.087	7.03	0.105
0.000087	95	9	0.1	FALSE	81.4	0.806	75.14	0.817
0.000549	186	5	0.2	TRUE	95.33	0.951	91.89	0.948
0.000446	271	18	0.2	TRUE	33.53	0.32	28.65	0.362
0.000583	15	255	4	TRUE	1.47	0.011	0	0
0.000667	104	72	0.1	FALSE	88.93	0.883	86.49	0.915
0.000289	860	238	0.5	FALSE	95.13	0.95	88.65	0.905
0.000686	396	163	2	FALSE	21.73	0.195	8.65	0.133
0.000275	17	169	1	TRUE	0.6	0.006	0	0
0.000141	10	105	3	TRUE	0.53	0.004	0	0
0.000363	29	3	4	TRUE	2.07	0.017	0.54	0.009
0.000219	934	54	5	TRUE	29.27	0.279	28.11	0.368
0.00018	529	1	0.1	FALSE	98.27	0.982	92.43	0.94
0.000708	204	11	3	TRUE	73.8	0.731	76.22	0.819
0.000756	469	13	4	TRUE	92.2	0.92	89.19	0.925
0.000124	244	105	0.2	TRUE	0.67	0.004	0.54	0.007
0.000917	12	7	5	TRUE	1.73	0.01	1.08	0.017
0.000895	13	31	2	FALSE	7	0.058	7.03	0.102
0.000414	750	32	4	TRUE	93.8	0.935	88.65	0.924
0.000529	38	15	0.1	FALSE	81.27	0.808	82.7	0.871
0.000093	193	3	2	FALSE	10.4	0.091	3.78	0.054
0.000459	186	51	0.5	TRUE	3.27	0.028	4.86	0.07

Table 7.1: Fold 1 Full Results.

LR	LR Step	Accum Iter	Triplet Margin	Hardest Triplets	Accuracy (%)	F1 Score	New Class Accuracy (%)	New Class F1 Score
0.00017	511	118	0.5	FALSE	90.53	0.9	82.7	0.852
0.000584	991	6	3	FALSE	36.2	0.339	16.76	0.232
0.000836	647	1	5	TRUE	89.47	0.893	87.03	0.899
0.0003	196	1	3	FALSE	22	0.206	12.97	0.175
0.000567	114	3	4	FALSE	20.53	0.185	13.51	0.184
0.000575	31	1	5	TRUE	25.6	0.238	25.41	0.336
0.000038	33	2	0.2	FALSE	24.27	0.226	22.7	0.307
0.000059	14	14	3	FALSE	2.13	0.015	0.54	0.009
0.000192	143	39	2	TRUE	1	0.007	1.62	0.026
0.000312	21	9	0.1	FALSE	62.2	0.608	54.05	0.639
0.000722	180	1	4	FALSE	22.6	0.21	18.38	0.258
0.000293	30	19	5	FALSE	6.33	0.052	5.95	0.09
0.000673	106	2	0.1	FALSE	94.8	0.945	93.51	0.953
0.000745	25	155	3	FALSE	6.93	0.058	4.86	0.074
0.000622	112	57	4	TRUE	2.07	0.015	1.08	0.014
0.000347	202	9	5	TRUE	34.33	0.33	29.73	0.387
0.00036	488	4	0.5	TRUE	98.4	0.983	94.05	0.943
0.000087	214	1	0.5	FALSE	93.4	0.932	87.57	0.905
0.000584	51	1	4	FALSE	12.87	0.11	8.65	0.127
0.000794	354	2	0.1	FALSE	97.47	0.973	94.59	0.95
0.000777	17	95	0.5	FALSE	7.33	0.06	5.95	0.086
0.000015	44	3	0.1	FALSE	3.07	0.021	2.7	0.041
0.000477	97	1	5	FALSE	13.2	0.116	7.57	0.112
0.000075	170	149	1	FALSE	10.8	0.096	5.41	0.077
0.000901	802	3	4	FALSE	35	0.327	19.46	0.261
0.0004	103	22	3	FALSE	11.53	0.105	3.24	0.049
0.000063	48	156	0.5	FALSE	2.47	0.016	3.24	0.045
0.000522	169	30	4	FALSE	18.47	0.172	9.73	0.144
0.000988	84	8	3	TRUE	37.13	0.359	34.05	0.429
0.000043	411	1	5	TRUE	95.4	0.953	91.35	0.936
0.000892	483	2	5	FALSE	27.87	0.257	22.7	0.279
0.000887	128	44	1	TRUE	4.6	0.036	3.78	0.062
0.00079	189	1	2	FALSE	16.13	0.141	18.92	0.245
0.000602	76	66	0.5	TRUE	1.07	0.005	1.08	0.017
0.000213	186	144	4	FALSE	10.67	0.096	8.11	0.115
0.00087	435	11	2	TRUE	93.73	0.937	90.81	0.935
0.00041	19	8	4	FALSE	8.13	0.069	4.32	0.072
0.000201	117	2	0.2	FALSE	95.27	0.952	90.27	0.922
0.000233	135	2	2	FALSE	15.13	0.135	8.11	0.123
0.000089	503	3	4	TRUE	95.13	0.949	89.73	0.915
0.000661	59	136	0.2	FALSE	50.4	0.483	41.08	0.508
0.000912	490	10	0.2	FALSE	98.6	0.986	94.05	0.955
0.000239	943	44	3	FALSE	25.33	0.241	8.11	0.113
0.000066	692	253	2	TRUE	1.6	0.01	0.54	0.009
0.000619	281	53	0.5	FALSE	94.67	0.942	89.19	0.913
0.00053	145	60	0.5	TRUE	1.67	0.012	0.54	0.007
0.000001	398	28	1	TRUE	0.27	0.002	0.54	0.008
0.000154	20	57	1	FALSE	3.4	0.026	2.7	0.045
0.000008	607	3	1	TRUE	4.2	0.029	2.7	0.044
0.000121	17	10	4	FALSE	2.87	0.022	2.7	0.042

Table 7.2: Fold 2 Full Results.

LR	LR Step	Accum Iter	Triplet Margin	Hardest Triplets	Accuracy (%)	F1 Score	New Class Accuracy (%)	New Class F1 Score
0.000217	23	154	5	FALSE	3.87	0.033	3.24	0.046
0.000233	369	113	0.2	TRUE	2.2	0.016	1.62	0.023
0.000363	17	147	0.5	TRUE	0.47	0.003	1.08	0.018
0.000469	14	3	4	FALSE	9.07	0.086	8.65	0.121
0.000541	155	247	4	TRUE	2.13	0.014	1.08	0.016
0.000121	239	88	0.1	TRUE	0.53	0.004	0	0
0.000798	258	18	3	TRUE	84.53	0.839	77.3	0.831
0.000961	32	70	0.5	FALSE	15.73	0.141	10.81	0.156
0.000813	215	1	0.1	TRUE	98.33	0.983	93.51	0.952
0.000872	156	27	5	TRUE	21.6	0.201	16.22	0.219
0.000382	39	43	0.1	FALSE	42.13	0.404	32.43	0.4
0.000286	330	5	3	TRUE	92.6	0.925	85.95	0.898
0.000774	304	200	0.1	TRUE	11.2	0.097	8.65	0.125
0.000829	366	26	1	FALSE	86	0.857	72.43	0.784
0.000665	24	3	5	FALSE	9.73	0.083	7.57	0.121
0.000928	388	74	0.5	TRUE	11.13	0.1	5.95	0.087
0.000313	289	1	0.1	FALSE	97.93	0.977	90.81	0.926
0.000256	922	5	5	FALSE	42.53	0.405	16.76	0.232
0.000618	12	1	0.1	FALSE	70.13	0.689	66.49	0.746
0.000932	331	72	2	TRUE	5.73	0.046	7.03	0.093
0.000375	284	7	0.1	TRUE	94.07	0.937	89.19	0.931
0.000204	29	28	3	TRUE	0.6	0.004	0	0
0.00012	130	65	2	TRUE	0.4	0.003	0	0
0.000754	465	84	3	TRUE	37	0.358	27.03	0.365
0.000885	217	5	5	FALSE	22.47	0.204	12.43	0.165
0.000632	691	3	2	FALSE	31.73	0.291	10.81	0.143
0.000774	255	3	0.2	TRUE	98.33	0.983	91.89	0.949
0.000196	296	31	4	TRUE	4.33	0.034	4.86	0.067
0.000092	660	11	2	TRUE	62.4	0.61	57.84	0.655
0.000541	431	158	1	TRUE	16.73	0.144	13.51	0.192
0.000546	55	2	0.1	TRUE	35.13	0.335	23.78	0.305
0.000828	104	6	5	FALSE	17.53	0.156	9.73	0.135
0.000574	24	82	0.5	FALSE	7.73	0.065	4.86	0.074
0.00027	256	148	4	FALSE	13	0.116	4.32	0.06
0.000554	145	4	1	TRUE	91.2	0.909	76.76	0.819
0.000339	217	5	1	TRUE	86.07	0.856	75.68	0.81
0.000857	25	5	0.5	FALSE	68.33	0.672	52.97	0.619
0.000543	55	3	5	TRUE	13.13	0.115	15.68	0.209
0.000099	71	27	3	FALSE	5.07	0.037	3.24	0.049
0.000668	14	250	0.5	FALSE	6.87	0.056	5.95	0.082
0.000611	170	1	0.1	TRUE	98.73	0.987	91.35	0.941
0.000928	435	9	5	FALSE	22.33	0.206	8.11	0.105
0.000081	20	117	1	TRUE	0.2	0.001	0	0
0.00081	641	1	0.1	TRUE	99	0.99	94.05	0.962
0.000446	177	24	2	FALSE	18.13	0.166	9.73	0.14
0.000104	23	43	4	FALSE	2.53	0.017	2.16	0.029
0.000567	635	3	2	FALSE	31.4	0.288	17.3	0.232
0.000061	18	9	3	FALSE	2	0.011	2.16	0.029
0.000413	38	10	1	TRUE	1.07	0.006	1.08	0.018
0.000332	22	30	1	TRUE	0.6	0.003	0	0

Table 7.3: Fold 3 Full Results.

Training & Validation Plots

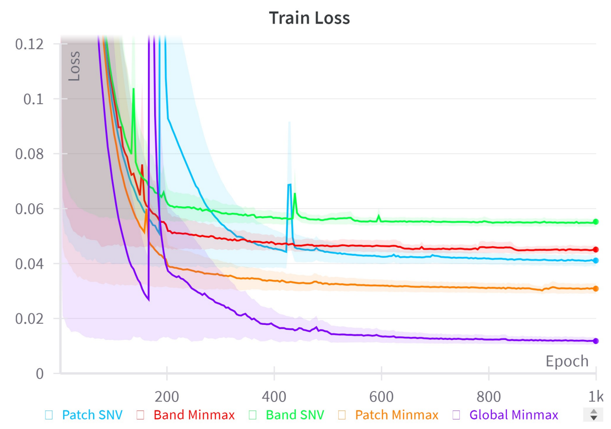

Figure 7.1: Training Loss Plots
Solid lines represent the mean values of all training runs for each preprocessing technique. Same colour clouds represent the standard error.

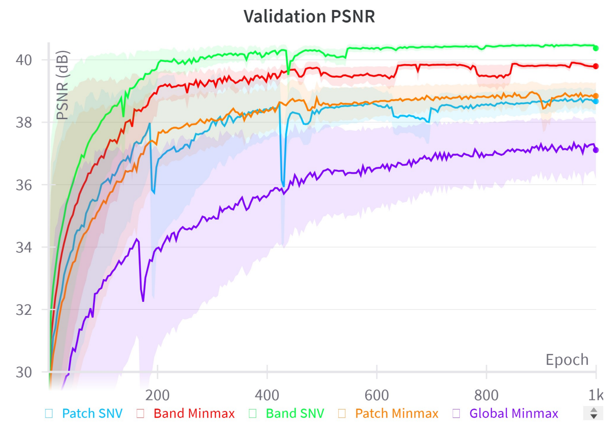

Figure 7.2: Validation PSNR Plots
Solid lines represent the mean values of all training runs for each preprocessing technique. Same colour clouds represent the standard error.

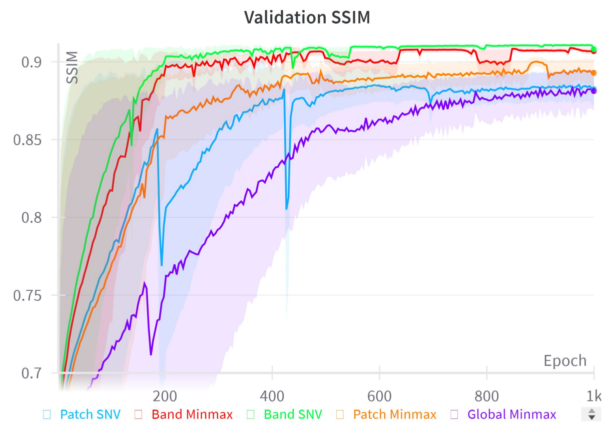

Figure 7.3: Validation SSIM Plots
Solid lines represent the mean values of all training runs for each preprocessing technique.
Same colour clouds represent the standard error.

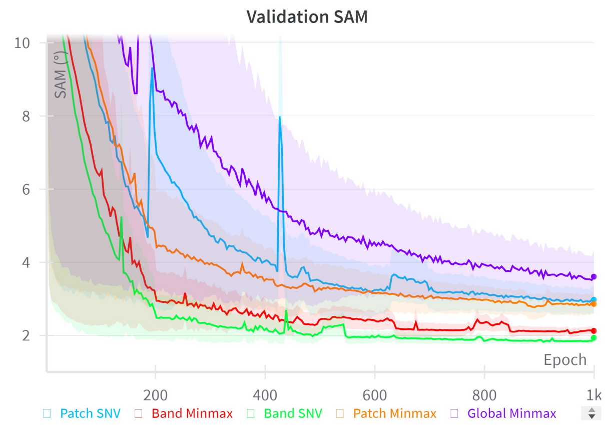

Figure 7.4: Validation SAM Plots
Solid lines represent the mean values of all training runs for each preprocessing technique.
Same colour clouds represent the standard error.

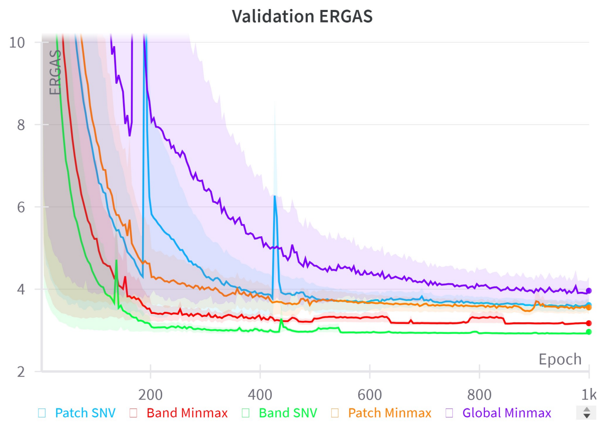

Figure 7.5: Validation ERGAS Plots
Solid lines represent the mean values of all training runs for each preprocessing technique. Same colour clouds represent the standard error.