Saving Our Bacon

Applications of Deep Learning for Precision Pig Farming

Jake Cowton

School of Natural and Environmental Sciences
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

September 2020

Abstract

The research presented in this thesis focussed on how deep learning can be applied to the field of agriculture to enable precision livestock farming for pigs. This refers to the use of technology to automatically monitor, predict, and manage livestock. Increased consumer awareness of the welfare issues facing animals in the farming industry, combined with growing demand for high-quality produce, has resulted in a need for providing farmers with tools to improve and simplify animal care. The concept of precision livestock farming tackles these requirements, as it makes it possible to treat animals as individuals, rather than as batches. This translates to tailored care for each animal and the potential for higher-quality produce. As deep learning has shown rapidly increasing potential in recent years, this research explored and evaluated various architectures for applications in two distinct areas within pig farming. We began by demonstrating how deep learning methods can be used to monitor and model the environmental conditions in which pigs are living in order to forecast oncoming respiratory disease. Implementing this approach can mean earlier intervention than if simplify looking for clinical symptoms. However, as not all diseases are caused by environmental conditions, we also implemented and evaluated a full workflow for the localisation and tracking of individual pigs. This made it possible to extract behavioural metrics to better understand the wellbeing of each pig. Overall, this research shows that deep learning can be used to advance the agriculture industry towards better levels of care, which is valuable for all stakeholders.

Acknowledgements

I would like to begin by thanking my supervisory team for their support and encouragement over the past four years. Without their guidance, undertaking this research and completing this thesis would have been infinitely more challenging. In particular, I would like to thank Jaume Bacardit, who has guided me through the majority of my work and spent a great deal of time proofreading this thesis and the papers stemming from it.

A number of people have supported me throughout my work on this research. In particular, Ben whom I thank wholeheartedly for putting up with me for the past nine years. His friendship has made this journey all the more enjoyable and his support has played a huge role in keeping me going. I would also like to thank Sarah for taking the time to proofread this thesis. Her support throughout the last few weeks of writing has been incredible.

Finally, my thanks goes to my parents for their commitment and dedication to ensuring I reached my full potential, and for their unwavering love and support. The countless number of hours they spent working with me from a young age to ensure I achieved well in school is what has enabled me to go on to pursue a PhD. I will never be able to thank them enough for all that they have done for me.

The thesis has received funding from the European Union Seventh Framework Programme for Research, Technological Development and Demonstration under grant agreement 613574 (PROHEALTH) and the European Union Framework Programme for Research and Innovation Horizon 2020 under Grant 633531 (Feed-a-Gene).

Publications

- Cowton, J., Kyriazakis, I., Plötz, T., & Bacardit, J. (2018). A Combined Deep Learning Gru-Autoencoder For The Early Detection Of Respiratory Disease In Pigs Using Multiple Environmental Sensors. Sensors, 18(8), 2521.
- Cowton, J., Kyriazakis, I., & Bacardit, J. (2019). Automated Individual Pig Localisation, Tracking and Behaviour Metric Extraction Using Deep Learning. IEEE Access, 7, 108049-108060.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

An ever-increasing population [9] directly translates to an ever-increasing demand for food. Combined with an increasing consumption of meat per capita [10], it is immediately clear that animals play an integral part in the future of the world's food supply, with pigmeat being the most popular [10]. In order to supply the growing demand, the agricultural industry must increase farming intensity. Generally, this can be achieved in one of two ways: increase the number of animals farmed, or improve the performance of each animal. However, it is not enough to simply *produce more*.

The agricultural industry has a responsibility to the environment and to animal welfare, both of which must be taken into account when scaling intensity. Additionally, figures from 2015 show that many countries (including the vast majority of European and North American countries) exceed the target of no more than 50mg of antibiotics per kilogram of meat production [10–12]. A broadly agreed target that was set due to concerns that improper and excessive use will lead to antibiotic resistance in both livestock and humans [13]. This means further reductions in usage of antibiotics are still necessary despite the need for higher intensity farming. These concerns are not just academic; consumer awareness with regards to these issues has been growing rapidly [14, 15]. This has mounted even more pressure for solutions to be environmentally sustainable, whilst improving the overall welfare of animals. In summary, the agricultural industry is tasked with providing increasing quantities of high-quality food, in a way that reduces its environmental impact, reduces its overall antibiotic use, and improves the quality of life and welfare for animals, all whilst keeping costs as low as possible for the consumer.

This need for increased production, whilst also abiding by the limitations and requirements, fundamentally set by consumers, poses a substantial challenge to the industry, and was therefore the core motivation behind the research presented in this thesis. We sought implementations that embrace the concepts of Precision Livestock Farming (PLF) [14], and that assist, rather than replace, the humans involved in the management of livestock[1]. We explored, developed and evaluated an early-warning system for oncoming respiratory disease, along with a full workflow for providing farmers with behavioural information for each individual pig. A farmer that has access to this information is in a significantly better position to provide a level of care, not only at an individual level, but also at an earlier stage than if they were only looking for clinical symptoms.

[1] referred to as farmers throughout the remainder of this thesis

Applying machine learning to agricultural challenges is nothing new [16]. However, deep learning, the subcategory of machine learning we employ throughout this thesis, is a relatively new field whose full potential for applications in agriculture is still in the process of being realised. The remainder of this introductory chapter introduces and outlines what both PLF and deep learning are, and how the latter can be used in pursuit of concepts of the former. Due to the interdisciplinary nature of this thesis, Section 1.2 contains a high-level overview of some of the core concepts of deep learning that are required to understand the design decisions made throughout this thesis. This chapter concludes by setting out the overarching aims and objectives of this thesis and provides an overview of the subsequent chapters.

1.1. Precision Livestock Farming

PLF is a subset of precision agriculture that focuses specifically on livestock. The currently accepted definition notes it as a type of management strategy that combines temporal, spatial, and individual data with "other data" to inform management decisions in such a way that improves performance [17]. Performance, in this context, can be measured in various ways such as, but not limited to, efficiency, productivity and economical. As disease drastically hinders performance, using collected data to avoid outbreaks of disease is one of the many goals of PLF.

There are a wide variety of variables that can be measured for the purpose of informing management decisions. Commonly tracked variables typically include environmental [18], consumption [19], and general animal activity [20]. Temperature and humidity are commonly monitored environmental variables in livestock management [21]. CO_2 concentration is also often monitored as it can be used as a proxy for how well ventilated an area is (i.e. a higher concentration indicates low ventilation and vice versa). Knowing this can help estimate the concentration of ammonia on the air, which can increase the risk of respiratory disease [18]. Monitoring these variables is strongly recommended as they are known risk factors of various respiratory diseases [21].

Monitoring an animals consumption over time, along with their weight, allows farmers to calculate Feed Conversion Ratio (FCR). This ratio is a metric for understanding how well an animal is converting the food they are eating into mass (i.e. the performance of an animal). This is particularly important for growing pigs, as the main goal is to maximise the weight gain of a pig with as few resources possible in the shortest amount of time. An animal's FCR over time, alongside pigs of similar size and gender, can give a good indication of the animal's health [20, 22]. In addition to FCR, research has shown that water consumption, particularly changes in consumption pattern, is also a strong indicator of pig health [23, 24].

The implementations required for the monitoring of both the environmental and consumption variables described above are relatively straightforward, as they can all be directly measured using electronic sensors. Whereas, monitoring animal activity is more resource-intensive. Animal

activity monitoring is commonplace throughout research [25–27], as it can be used as a means to understand their behaviour [28]. This is valuable data as changes in an animal's behaviour is another potential indicator for the presence of disease [29]. Typically, this data is obtained by human observation, either in-person or through the use of a video recording, where the observer notes the behaviour of animals over time. This approach has its limitations, as different people can interpret the same data in different ways [30]. Research has also shown that different people can even evoke different reactions from animals [31], adding bias to the data collected. Furthermore, having a human observe an animal can be expensive. This is even more so the case if specialist knowledge is required to capture the required data, for example, to understand specific animal behaviours.

In the majority of cases, tracking the change in these values over time provides much greater insight than knowing the value at a single given time, as it is often the changes in these values that can indicate potential problems. However, taking FCR as an example, in a scenario where a single pig's FCR decreases, only a small change in the batch's average would be seen. This means several pigs need to show a decrease in FCR for a farmer to see a change that warrants intervention. Therefore, tracking metrics for individual animals, rather than as a batch, drastically increases the value of collected data. With individual-level data, the farmer can know precisely which pig is having issues and intervene at an earlier stage, which in many cases would cause less disruption, reduce the need for medication and make any necessary treatment more effective [32].

Tracking individual-level data is an attractive solution to some of the challenges facing livestock management. However, tracking these variables for individuals, rather than for an entire batch, drastically increases the challenges around implementation. For example, calculating FCR for a group of animals requires weighing the amount of food given to the group, then, at a given time, weighing a sample of the animals in the group and weighing how much feed remains to calculate total consumed feed. Making the same calculation for an individual requires knowing how much feed the individual consumed along with its weight at a given time, which is not feasible without constantly monitoring each animal.

Electronic sensors that can assist with individual tracking have rapidly become more accessible, and the accuracy of data they capture is sufficient for most use cases. The two main sensor types that are used are Radio-Frequency Identification (RFID) tags and high-resolution cameras. RFID tags, typically attached to a pig's ear, are used as a means of identifying specific pigs using RFID antennas, which can remotely detect the ID of a given tag (within a limited range depending on the type of sensor used). This approach has been used to calculate the feeding behaviour of individual pigs [33] by placing RFID antenna around the feeding area and detecting when specific pigs were in that area. There are limitations to this approach, as the authors of this research acknowledge. Due to the nature of RFID tags, only one pig can be detected at a time. In this paper, this limitation was managed by only allowing one a pig access to the feeder at a time, but this is a sub-optimal solution for implementation in commercial farms, as adapting feeding areas to allow only one animal access is impractical.

More recent research has shown RFID tags can also be used to track a pig's movements between certain areas. In this paper [34], each pig was assigned an ultrahigh-frequency RFID tag and its location was determined by a set of antennas placed in calculated "hotspot" areas. If a pig entered the hotspot, the antenna would detect it and log the pig's ID and time it was detected. Knowing the distances between the hotspots, the authors were able to infer a "virtual walking distance". This metric could be used to detect lameness in pigs by applying a linear moving window regression model, that was applied on a per-pig basis, that made predictions about expected behaviour. Deviations between predicted and actual measurements indicated a potential problem. This research demonstrated an ability to detect lameness in 32% of cases; a particularly low detection rate, which the authors attribute to the high variation between pigs. However, the authors argue that their approach acts as an effective way of tracking general, individual pig movements within a pen.

The location data collected from RFID-based methods is very coarse as it can only provide a sequence of times a pig was at one of several locations. Additionally, the only way to verify the captured data would be to either have a human presence or to install cameras to verify the RFID data. As discussed earlier in this section, research has used cameras as a means for monitoring pigs, which avoids in-person contact with animals. However, more recent research has made use of computer vision techniques to automatically monitor pig behaviour in video footage. Two main types of camera are typically used for this: 3D depth cameras that use infrared light [35–37] and 2D RGB (colour) cameras [38, 39]. Both types of cameras offer different benefits and drawbacks. Depth cameras offer much more information regarding the shape and size of the pig, which enables better performance in tasks such as weight estimation [40]. However, the main downsides of this hardware are that depth cameras are typically expensive and, as they use infrared light, no colour data can be captured. On the other hand, high-definition cameras are very inexpensive and, though they only produce a 2D image, they do capture colour data, which can be integral in telling two pigs apart for the purpose of identifying individuals. Currently, one of the main approaches to automatically processing the images recorded by cameras is to train a Convolutional Neural Network (CNN), a deep learning architecture, which we will discuss in the following section.

1.2. Deep Learning

In its most fundamental form, "deep learning" is a type of Artificial Neural Network (ANN) [41] that uses several hidden layers. This definition is more of a technicality as, in reality, the hidden layers in deep learning models are much more complex than traditional multi-layer perceptrons, which are comprised of what we now refer to as fully connected layers. In traditional machine learning methods, including shallow ANNs, extensive preprocessing and feature engineering is required to model the input data. Whereas in deep learning algorithms, the extensive and more

complex layers are relied upon to model data without the need for much preprocessing or feature engineering, if any at all.

Though research in this area has drastically increased over the past decade, the term itself is not new [42]. However, at that time, the large amount of computational power required for neural networks to achieve competitive results in reasonable time made them less desirable, as compute power was limited. This resulted in methods, such as support vector machines [43] and random forests [44], being able to achieve comparable, if not better, performance whilst requiring less computational resources. Falling into the "traditional machine learning" category, they required relatively extensive data preprocessing for them to be effective.

The necessary computing power was eventually more readily available [45, 46] and significant advances in the field were made. The following is a non-exhaustive, high-level list of some of the major advances within the field of deep learning. The application of backpropagation to CNNs, allowing them to outperform traditional computer vision approaches [47] was one of the first pieces of research that pushed deep learning architectures into the spotlight. This led to the development of LeNet [48], a top-performing CNN-based image classification architecture.

Around the same time, Long Short-Term Memory (LSTM) units were presented as a solution to the memory issues of vanilla Recurrent Neural Network (RNN) [2] implementations [49], enabling recollection of data from over 1,000 prior timepoints in the past. These architectures were particularly well-suited to handling sequential and time-series data as the temporality of data is explicitly handled by their recurrent architecture, rather than requiring the temporal relationship between values to be learned through training. In later research, they were augmented with an additional component referred to as “attention”, which were learned weights that improved performance in a number of temporal applications [50–52]. This was followed by the introduction of the transformer, an architecture which relied almost entirely upon attention weights [53]. These architectures are discussed in more detail in Sections 2.4.1 and 3.2.2 respectively.

The introduction of the Rectified Linear Unit (ReLU) activation function [54], one of the most used activation functions in deep learning, made significant strides to minimise the effect of the vanishing gradient problem. This is where small derivatives of each layer in the network are multiplied together causing the error that is propagated through the network to exponentially decrease. This causes low gradients in earlier layers, meaning very slow learning. Leaky ReLU proposed changes that further minimised the vanishing gradient problem for this activation function [55].

Later, batch normalisation [56] greatly improved the generalisability of a model by normalising inputs on a mini-batch level, enforcing a mean of 0 and a standard deviation of 1, ensuring that all mini-batches had the same distribution. The authors claimed that this is a form of regularisation, and therefore using batch normalisation mitigates the need for using “dropout”.

[2] we use "RNN" throughout this thesis to refer to all recurrent architectures including LSTMs and GRUs

This is another regularisation strategy where a proportion (often half) of the neuron activations are ignored [57]. Layer normalisation was later introduced for recurrent-based architectures [58], which applies the same theory as batch normalisation but to a different axis of the input data (Figure 1.1). All of these methods reduce training time and prevent overfitting for their respective architectures.

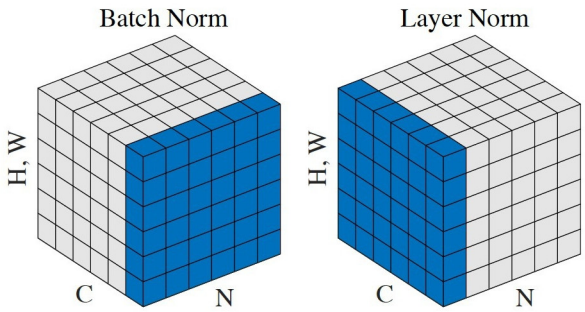


Figure 1.1 A visualisation of the data to which batch (left) and layer (right) normalisation is applied. Sourced from [1].

Finally, a number of optimisation algorithms were presented that improved the convergence of deep learning models. Before the use of deep learning models was widespread, AdaGrad [59] and stochastic gradient descent with momentum [60] were two of the most commonly used optimisation algorithms. The latter was commonly implemented with a learning rate scheduler that could modify the learning rate throughout training based on certain parameters (e.g. number of iterations/epochs or performance). One such method of learning rate scheduling reduces the learning rate using cosine annealing strategy [61]. The authors of this paper also included "warm restarts" that increase the learning rate after a given number of steps to avoid getting stuck in local optima. Since, RMSProp [62] and ADAM [63] have taken over as the two main optimisation algorithms; the latter was published as an improvement over the former and both, alike AdaGrad, are adaptive learning rates.

The combination of these advances in training strategy and greater compute power allowed deep learning implementations to become leading solutions to complex problems in many areas. This was most notably the case for areas such as computer vision [64], Natural Language Processing (NLP) [65] and data generation tasks [66].

Selecting the best architecture for a given task is a crucial part of building a high-performance solution, especially as architectures begin to be successfully used outside of their traditional applications, such as using CNNs for text translation [67]. However, identifying the best metric to measure performance is an equally important decision. Different metrics are often used to evaluate different parts of a system, such as one metric to evaluate the performance of a trained model on a test set and another metric to evaluate the performance of that model in a given application. When selecting which metric to use, it is crucial to understand not only the objective

of the system (e.g. classification vs regression), but also the data that it is using, as this can affect the appropriateness of a given metric. For example, in a balanced dataset used for classification, it is common to use classification accuracy as a metric for the performance of a model, as it is simple to calculate and interpret. However, for datasets where there are imbalanced classes, either in the number of samples per class or in the importance of one class over another, classification accuracy would not be appropriate as it does not account for class distribution. Precision and recall are often used in these circumstances [68], along with their harmonic mean, F_1, as it takes the class distribution into account rather than only comparing correct vs incorrect classifications. This feature of F_1 has led to it becoming one of the most widely used evaluation metrics for classification tasks in machine learning. Although, it is not always the most appropriate, as it does not account for correctly classified negative cases (true negatives). In applications where these are important, other metrics, such as Matthews Correlation Coefficient (MCC), may be better suited [69]. Therefore, we paid particular attention to the performance metrics we made use of throughout our research.

In this thesis, we focussed on two of the main categories of deep learning architectures: CNNs and RNNs. CNNs are primarily comprised of layers of convolving filters that are applied to input. They gained popularity for their exceptional performance in computer vision challenges such as image classification [70], object detection [71] and facial recognition [72]. More recently, they have been successfully applied to other applications such as natural language processing [73] and speech recognition [74]. RNNs, specifically the LSTM variant [49], garnered attention for its strong performance in sequence processing, especially in sequence-to-sequence tasks such as text translation for NLP [75] and time-series forecasting [76]. There are limited use cases for NLP in an agriculture setting [77]. However, sequence processing, the underpinning concept for many deep learning-based methods applied to NLP, is a valuable tool for modelling the time-series data that is collected about animals (Section 1.1).

1.3. Deep Learning in Agriculture

Traditional machine learning methods have been used throughout agriculture for some time; a substantial proportion of this has focussed on crop yield and disease forecasting [16]. More so, these implementations often inherently adopt the concepts of precision agriculture, not only by enabling the capture of richer and more specific data, but additionally by assisting in the processing of complex data such as images and videos. Support Vector Machines have commonly been used in these applications [78–80]. For example, in this particular paper [80], the authors implemented a Support Vector Machine (SVM) for the purpose of an early-warning system to alert farmers to upcoming decreases in egg production. In this, strong performance across a range of metrics is achieved by the SVM. However, the authors were required to handcraft the features they used as inputs to the model, whereas deep learning algorithms can be applied to data without the need for feature engineering, and often with performance gains. For example, CNNs can be

used to automatically detect fruits in images. When this is applied to an entire field, an estimate of the total number of fruits can be obtained. Furthermore, if a time-of-flight camera is used or if the distance from the camera to the fruit is known, then this can be translated to an estimated total yield based on the size and number of fruit. Recent research has used Faster R-CNN [81] to implement the detection component of this workflow [82]. However, a commonly observed challenge faced by CNN-based approaches is the vast variations in illumination conditions in real-world conditions. Though strong performance is still achievable, this is seen as a current limitation to the practical applications of these methods in the real-world [83].

CNN-based approaches are also well suited for weed detection [84]. In this implementation, the implemented CNN was capable of detecting weeds even in scenarios where the weeds were occluded, though it did not go so far as to identify the exact type of weed, which is useful for autonomously determining the correct weed control method. This implementation used DetectNet [85], essentially GoogLeNet [86] without the input and two of the output layers, to detect weeds in an image. This is a fully convolutional method (i.e. there are no fully connected layers) that takes an input image and outputs an image mask. That mask is then translated into bounding boxes using thresholding to show where weeds are. Similar research has made use of the You Only Look Once (YOLO) [87] architecture as a starting point for proposing a custom architecture for a detecting a specific type of weed [88]. In this implementation, synthetic training data is generated to make up for a lack of real-world data. This meant that within the training data there were variations of image brightness as there would be in real-world data. This approach tackles one of the major challenges in deep learning; a lack of annotated data.

Although the applications discussed thus far have focussed on CNNs, there are also implementations using RNN-based approaches. Satellite image data contains valuable data for understanding and classifying land cover and deep learning can be used to process this data [89, 90]. At first glance, this appears to be simple image data that could be processed by methods that perform well on spatial data (e.g. CNNs), but land cover changes throughout the year depending upon the season. Recent research has shown that because of this, methods that can account for temporal data (e.g. RNNs), as well as spatial, tend to outperform CNN-based alternatives [91] that can only assess images in a single context. Deep learning methods have also been applied to livestock in applications such as animal counting and classification [92, 93], growth forecasting [94], and activity recognition [95]. In the activity recognition paper, CNNs were used to process the input images and detect pigs. Assumptions were made based on the overlap between the head area and the feeding area to determine if the pig was feeding or not. However, this assumption does not always hold, as it is not uncommon for pigs to be in the feeding area but not eat, referred to as "non-nutritive visits". This is an area recent research has tackled using CNNs [96]

In all of the activity tracking methods discussed above, none track the identity of individual pigs, often reverting to painting an identity onto the pig itself [97, 95]. Deep learning applications have been successfully implemented and deployed for real-world use for the purpose of detecting, tracking and predicting human behaviour. In particular, research has achieved

implementations that track humans in video feeds [98], including across multiple cameras [99]. However, this is a challenging task when trying to apply similar techniques to pigs, as they are often indistinguishable from one another to the human eye. Humans, on the other hand, are easier to tell apart due to differences in appearance, most notably caused by clothing, which impacts both spatial and colour-based features. These are feature changes that CNNs perform well at detecting, making them well-positioned to be able to tell two humans apart based on their appearance.

Research into using these methods for pig identification has presented a solution through the use of identity tags attached to a pig's ear. These tags can be detected using CNNs and used to assign an identity to the pig [100]. The use of additional hardware that must be attached to the pig certainly overcomes the indistinguishability challenges, however, the installation and removal of tags adds to the workflow of day-to-day farming, which can affect adoption in real-world applications. One of the few successful solutions to identifying pigs that requires no additional hardware or markings on the pig has been through the use of a "tag box" [101]. This tag-box was defined using keypoint detection, around which a box was defined and HOG [102] features extracted for identification.

A substantial proportion of the challenges and solutions presented in this section relate to how deep learning can be used to process image data, and we have shown that there is increasingly extensive literature in this space. This is likely due to the fact that computer vision is one of the main areas where deep learning methods have excelled. However, there are also RNN-based applications for methods such as sequence processing [103–105]. In particular, this can be used for disease prediction using time-series data, though much of the literature has focussed on human applications [106]. One of the first papers to apply a recurrent architecture in this area [106], specifically, to use multivariate inputs, demonstrated that LSTM architectures have a clear ability to model environmental conditions, and used this to forecast the spread of flu. The environment and flu data used was weekly totals and preprocessing was applied to each of the variables independently to account for each of their respective lagged impact on the total flu count (e.g. temperature change may take x days to impact flu prevalence whereas humidity change may take y days). Theoretically, this is an unnecessary step as a well trained LSTM architecture should be capable of accounting for these offset relationships.

Other papers have also demonstrated the performance of RNN-based models on univariate time-series data [107, 108], though this is typically a less challenging scenario. The success of RNN-based models that have been applied to disease prediction in humans, though indicative of their general capability, does not mean that they would be successful in a livestock setting, as there are substantial differences between the two applications. Specifically for livestock housed indoors, the environment in which animals are kept is typically rigorously climate controlled to maintain optimal conditions [109]. This means that changes in the environmental conditions are typically on a much smaller scale than in open-air conditions. The literature shows that

environmental conditions do impact disease for livestock, flu or otherwise, so a model used to detect these changes needs to be able to do so for much subtler patterns and relationships.

Recent research has explored the performance of LSTM-based models in comparison with Autoregressive Integrated Moving Average (ARIMA) directly, however, these are frequently addressing financial stock prediction challenges [110, 111]. Interestingly, one particular paper [111] combines both CNN and LSTM-based architectures to create a single model for forecasting financial stock data. In this model, the LSTM observes the minute-by-minute stock price and the CNN processes a "candlestick chart", as an image, of the same data. Though the authors observed a performance improvement by combining the features produced by these two models, it begs the question of whether the sole LSTM model would have benefited from more in-depth fine-tuning as it contains a more granular representation of the data contained within the graph.

Overall, there are many applications for deep learning in agriculture, though some are still in their infancy as the benefits of deep learning for PLF become more widely evaluated and understood. The following section outlines the overarching aims of this thesis, along with an overview of the areas in which this research applies deep learning methodologies.

1.4. Thesis Outline

Two core aims drove the research presented in this thesis. Firstly, we aimed to investigate how deep learning methodologies can be implemented for use cases in operational farms; moving away from tightly constrained lab conditions. We evaluated our implementations under real-world conditions, on operational, commercial farms, in order to ensure the solutions we presented addressed all of the challenges. Secondly, we sought to demonstrate how these implementations could be used to make treating animals as individuals a more manageable task for farmers. Across the experimental chapters, this research addressed two broad applications: environmental monitoring for disease forecasting, and individual behaviour tracking. The first of these shows how deep learning can be used to process complex, imbalanced data and the second used deep learning to focus on animals as individuals.

We began this research by monitoring and modelling multiple environmental sensor data to provide an early warning system for oncoming increases in respiratory disease prevalence (Chapter 2). We compared our proposed recurrent-based method to other deep learning methods alongside traditional approaches. Our evaluation demonstrated that our architecture choice and choosing to treat disease detection as an anomaly detection problem, allowed it to achieve the best performance. As this method required substantial computational resources, further research was conducted to evaluate the benefits and drawbacks of alternative architectural variations that would use fewer resources (Chapter 3). Here we evaluated both CNN and transformer-based architectures against our RNN-based one, showing that, under the right conditions, CNNs can be powerful alternatives even when data is temporal.

In order to extract behaviour tracking metrics for individual pigs, we implemented a full workflow for the detection and tracking of pigs. This was achieved using standard colour cameras, without the need for additional hardware (e.g. RFID or ear tags), markings on pigs, or the definition of "tag boxes". The data produced by this workflow was then used to extract metrics regarding each pig's activity levels (Chapter 4). This data could be used by farmers to better understand the condition and needs of each pig. In order to enhance the tracking performance on an individual level, an additional module for this workflow was developed for the purpose of re-identifying pigs after prolonged occlusions (Chapter 5). We evaluated several CNN-based approaches that were capable of re-identifying pigs based on their appearance to re-assign seemingly newly detected pigs to their original identities. This thesis is then finalised by a general discussion of the research that was presented (Chapter 6), how it relates back to the material and challenges presented in this chapter, and proposes various avenues for further research. Relevant background material is covered both in this introductory chapter and in the relevant chapters rather than in a dedicated background material chapter.

Chapter 2. A Combined Deep Learning GRU-Autoencoder for the Early Detection of Respiratory Disease in Pigs Using Multiple Environmental Sensors

Abstract

We designed and evaluated an assumption-free, deep learning-based methodology for animal health monitoring, specifically for the early detection of respiratory disease in growing pigs based on environmental sensor data. Two recurrent neural networks (RNNs), each comprising gated recurrent units (GRUs), were used to create an autoencoder (GRU-AE) into which environmental data, collected from a variety of sensors, was processed to detect anomalies. An autoencoder is a type of network trained to reconstruct the data it is fed as input. By training the GRU-AE using environmental data that did not lead to an increase in respiratory disease prevalence, data that did not fit the pattern of "healthy environmental data" had a greater reconstruction error. All reconstruction errors were classified as either normal or anomalous using a threshold that was optimised using particle swarm optimisation (PSO), from which alerts are raised. The results from the GRU-AE method outperformed both classical statistical and other deep learning implementations that raised alerts when predictions deviated from actual observations. The results show that a change in the environment can result in occurrences of pigs showing symptoms of respiratory disease within 1–7 days, meaning that there is a period of time during which farmers can act to mitigate the negative effect of respiratory diseases, such as porcine reproductive and respiratory syndrome (PRRS), a common and destructive disease endemic in pigs.

2.1. Introduction

The forecasting of an oncoming health challenge in animals is fundamental to maintaining a high level of health and animal welfare. The earlier a disease is predicted, the sooner it can be dealt with, thus lowering the overall impact of the disease on both the animal and the farm, and increasing the likelihood of treatment success [32]. Respiratory diseases, such as porcine reproductive and respiratory syndrome (PRRS), pneumonia, and pleurisy are some of the most common types of disease found in commercial pig populations [21]. Therefore, it is highly

valuable to be able to predict occurrences of respiratory disease, which we define as the point in time where respiratory disease prevalence increases from 0 to greater than 0.

There are many factors that influence the contraction of respiratory diseases, in particular, the environment which the pigs inhabit. A number of the respiratory disease risk factors discussed in established research [21] - specifically, air quality, temperature, and humidity - are now able to be monitored continuously in real time and at a high resolution using inexpensive electronic sensors. Such data can be used to further understand the relationship between environmental conditions and respiratory disease.

In most situations, recording data continuously means storing data that predominantly describes normal circumstances, as anomalous events are less frequent. In an ideal situation, datasets are comprised of balanced data for each class. In the case of the data used in this chapter, where the classes are considered to be "healthy" and "not healthy", the classes are considered to be extremely imbalanced, due to the large amount of "healthy" data where no respiratory disease occurs. However, the large amount of normal ("healthy") data generated, allowed us to train a solid, robust model of normality. This meant that as environmental conditions were tracked over time, the model could be used to evaluate the data as it is received and any deviations from this model could be quantified. This quantification could be used to determine if that deviation indicated a potential problem that might cause occurrences of respiratory disease prevalence.

The objective of this chapter was to describe and evaluate a methodology for the early warning of environmental anomalies pertaining to respiratory disease in growing pigs. We implemented a recurrent-based autoencoder, built using gated recurrent units (GRUs) [112], that was trained to reconstruct the raw environmental sensor data which did not lead to an increase in respiratory disease prevalence within a batch of pigs. This GRU-autoencoder (GRU-AE) was then used to evaluate periods of sensor data and measure how close it is to normality. A threshold-based anomaly detector, whose parameters were optimised using particle swarm optimisation (PSO), used this score to determine whether to raise an alert for scores which are too great and therefore represent environmental conditions that do not fit in line with the data it was trained to represent.

The benefits of using such a method were that there was no need for hand-crafted features, as is typically required in classic machine learning and statistical approaches, meaning that raw sensor data could be used with minimal preprocessing. Also, no arbitrary thresholds were set, as the detection of anomalous events was wholly data-driven. In addition, there is no requirement for an in-depth understanding of the relationship between environmental conditions and pig health, as this is handled implicitly by the network. The drawback to this is that, because of the nature of RNN-based approaches, interpretability of the model is very poor. This work is structured as an anomaly detection solution, rather than, for example, predicting increases in respiratory disease prevalence; therefore, the method is able to perform on data consisting of low disease prevalence without affecting performance. However, the more cases of disease there

are, and the higher the prevalence at these times, the easier it is to validate the performance of a method.

Previous research has demonstrated the potential of a Long short-term memory (LSTM)-based encoder-decoder structure for multi-sensor anomaly detection [113], showing the methodology has good potential for using data which is not easily predictable [114]. This implementation used LSTM cells, rather than a standard artificial neural network (ANN) structure, in order to handle the temporal dimension of the data, and reconstruction error was used to determine deviations from normality. However, the results of their implementation on various datasets suffered distinctly either in precision or recall (defined in Section 2.4.1). This chapter revisits the design choices made in previous work, such as that by Malhotra et al., specifically by:

- exchanging long short-term memory cells for GRU cells
- introducing particle swarm optimisation to optimise an anomaly detector that determines whether the loss from the GRU-AE is anomalous
- changing how the overall system is evaluated to demonstrate a more balanced anomaly detection system

The remainder of the chapter is structured as follows. Firstly, Section 2.2 gives context for the approach taken. Section 2.3 explains the composition of the data and the preprocessing that was applied, followed by Section 2.4, which details how each of the models was constructed. Section 2.5 outlines the performance of the systems described, followed by the discussion in Section 2.6 and the conclusions drawn in Section 2.7.

2.2. Background

In recent years, agricultural research has begun to shift away from relying only on classical statistical approaches and started to regularly incorporate machine learning (ML) methodologies, such SVMs and Random Forest Classifiers, into the approach to data analysis [80, 115–119]. Recent work has seen SVMs being used, for example, to detect anomalies in the production curve of eggs produced by commercial hens, covering a range of prediction windows for the purpose of an early warning system [80]. Other research has applied ensemble learning methods (combined models) for the prediction of avian influenza prevalence using meteorological data, showing improved performance over support vector regression [115].

Classification methods have been used to separate normality from abnormality through the implementation of one-class classifiers (OCCs). SVM-based methods are very common in this approach, though there are many potential methods which can be applied. Models of this type usually have high sensitivity but a comparatively poor specificity [117], which is to be expected

given the natural imbalance in the datasets that these methodologies are commonly used for. However, these metrics are not always suitable, particularly if the true negatives drastically outweigh the number of false positives.

There are also approaches to anomaly detection outside of classification, such as clustering [120] and correlation analysis [121]. Clustering uses distance metrics to determine if a new point's distance from a cluster centroid exceeds an acceptable value: if this is the case, it is considered anomalous. However, methods falling into this category are typically unable to strike a reasonable balance between the quality of analysis and speed due to the innate temporal component of time-series data [122]. Conversely, correlation analysis uses similarity metrics to determine if new data is similar to known normal data.

Anomaly detection can be used in health applications to detect outbreaks of disease or other deviations from normality. A common methodology used for this task is Bayesian networks, which have been shown to outperform classical statistical methods, such as control charts, moving average models, and ANOVA regression [123]. In one simulation of an anthrax outbreak [124], the Bayesian network was able to correctly trigger an alarm within 2 days, in comparison to 9 days for the moving average model and 11–12 days for the others. Anomaly detection is very well suited to healthcare, as often there are not necessarily certain conditions that constitute a problem, but rather deviations from certain conditions. For example, when detecting arrhythmia through electrocardiography (ECG), it is not possible to have knowledge of all of the different types, as new types can occur in the future [125]. It is therefore necessary to understand what normality looks like, rather than what problems look like.

In health applications where sensors are used, there are two main challenges. Firstly, there is a need to focus not only on a high detection rate but also a low false positive rate [126], as, if this is high, alerts from this system will not be trusted. Secondly, though not in all cases, anomalies ought to be detected in near real-time [127] in order to provide a quick response so that interventions can be carried out.

Often, these approaches require a transformation to be carried out on the data in order to generate a useful feature representation [128–130]. However, deep learning-based methods do not require this. Raw data can be passed to a deep network without any hand-crafting of features, as this is handled intrinsically by the network [125], though this does depend upon a substantial amount of data to function effectively [131]. This is particularly valuable for high-dimensional data, as, provided the network is deep enough to model the data, all of the data can be utilised.

A relatively small amount of work in agriculture has made use of deep learning, which mostly focuses on image analysis and remote sensing [132]. This chapter demonstrates how deep learning methods, used in conjunction with evolutionary algorithms, can be used for multivariate time-series analysis for the purpose of anomaly detection.

2.3. Experimental Design

2.3.1. Data Collection

Data was collected for growing pigs from a variety of operating, commercial farms across Europe as part of an EU-funded multidisciplinary project. Each farm was represented by a number of batches of pigs (3–16 batches), where a batch consists of varying population sizes (40–1294 pigs), and the batches were monitored for varying lengths of time (14–145 days). The starting weight of the pigs was variable, ranging from 17.5 to 27.6 kg. The number of pigs per batch affected by respiratory disease was recorded on a daily basis. Respiratory disease prevalence was divided into four levels of severity, as defined by Zoetis's Individual Pig Care [133]. A pig could be either scored (identified as diseased but no treatment given) or treated for each level of severity. Health data, regarding the number of pigs showing symptoms of disease, was collected by a trained person that manually counted the number of pigs in each batch that showed symptoms of certain diseases, and classified it by the level of severity into which the symptoms fell. Given the nature of the system, pigs cannot be identified individually each day, so the number of pigs that are classified as showing symptoms can be new cases or pigs that were identified as such in the previous day.

The environmental data was collected using General Alert sensors [134], which were suspended in the centre of the room in which the batch was located. The sensors recorded the environmental features relevant to pig health [135]. Briefly, the system monitored temperature (a sensor range of -50 °C to +250 °C $\pm$ 0.05%), relative humidity (a sensor sensitivity of $\pm$2% and operating temperature range of -40 °C to +85 °C), and CO_2 concentration (with a range from 0 ppm to 5000 ppm $\pm$ 30 ppm or $\pm$3% of reading, operating temperature range from 0 °C to 50 °C, and humidity from 0% to 95%), which was sampled every minute.

2.3.2. Data Preprocessing

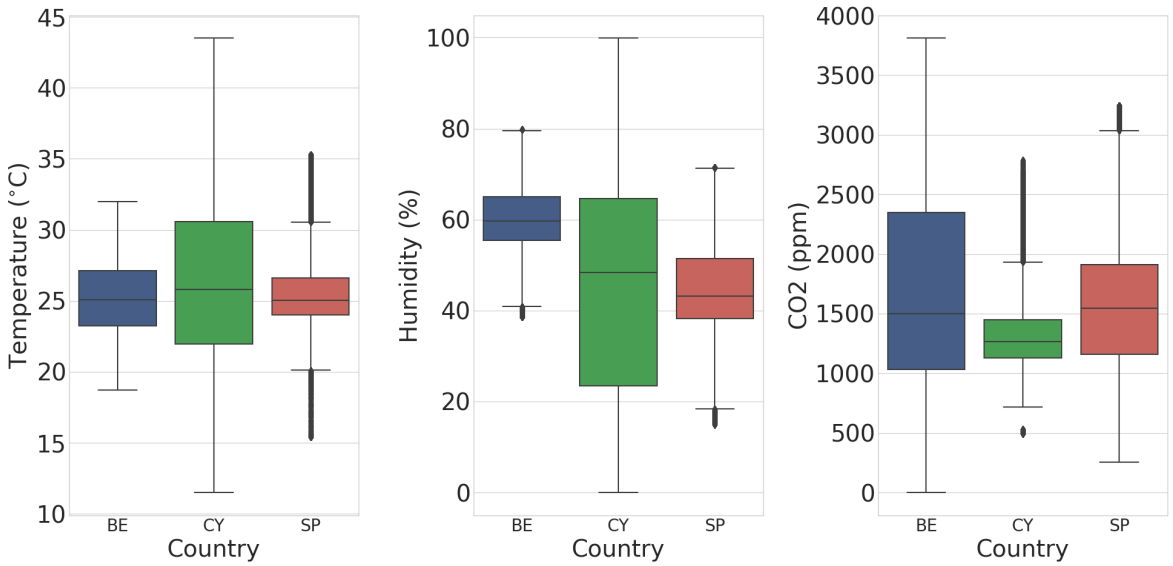


Figure 2.1 A boxplot describing the temperature, humidity, and CO_2 sensor data for the three countries included in the data collection—Belgium (BE), Cyprus (CY), and Spain (SP)—after preprocessing.

Incorrect sensor data was determined to be anything outside $\mu \pm 2\sigma$ per variable per batch and were removed from the data. This was in order to handle sensor errors, such as impossibly high values for temperature (e.g $70^{\circ}C$) and human interference, such as where pens were cleaned, but the humidity sensor was not sealed up correctly, allowing for extremely high values (e.g. 100%). It was not enough to simply set upper and lower boundaries for the sensors as very high and very low values are possible. Instead, this preprocessing step allowed for unrealistic spikes and troughs to be determined relative to the rest of the data for that batch.

Missing data induced both from the removal of outliers and due to hardware issues were filled using linear interpolation. This was deemed acceptable, as the environment has low short-term variance, meaning that, on average, the assumption of a linear change between two values was reasonable. The results of this preprocessing of sensor data are shown in Figure 2.1. The three sensor readings—temperature, humidity, and CO2—constitute the inputs to the models outlined in Section 2.4. The four severity levels were summed and used to calculate the proportion of pigs showing symptoms of respiratory disease on a given day, rather than looking at one specific level of severity). For our definition of prevalence, see Equation 2.1

$$\text{prevalence} = \frac{\text{Number of pigs showing any symptoms}}{\text{Total number of pigs}} \tag{2.1}$$

This was done because respiratory diseases rarely exceed the classification of light or mild, as in most cases they are treated before the disease develops into more severe symptoms. Therefore, there were very few cases of severe or irrecoverable respiratory disease across all farms. Even

with the aggregation of data, the level of prevalence was still very low, and it decreased even further over time (Figure 2.2). The sensor data was merged with the daily health data by upscaling the resolution of the latter by duplicating the values for the full 24 h period. All input was normalised to between 0 and 1.

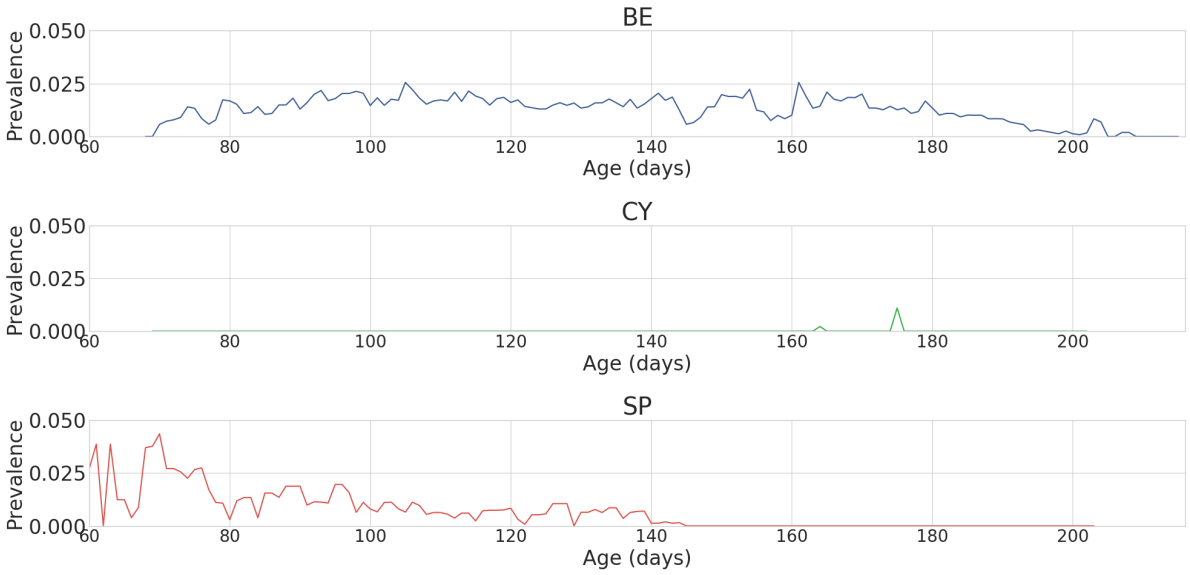


Figure 2.2 A graph showing the average prevalence of respiratory disease against age (in days) in all batches of growing pigs in Belgium (BE), Cyprus (CY), and Spain (SP) after preprocessing.

Separation of Assumed Healthy and Assumed Unhealthy Data

As this chapter investigates the effects of environmental conditions on health, it was necessary to account for the existence of a lag period between a change in environment and a change in health, as a change in environment will not immediately result in an outbreak of disease [136]. Lag periods of 14 and 21 days were both evaluated [136]; greater windows were not used, as this would greatly reduce the amount of data available for training. This lag period was referred to as the assumed unhealthy window, u. Using this window, the data was labelled as one of two sets: "assumed healthy" and "assumed unhealthy". Data from any point where the proportion of pigs showing symptoms was greater than 0 to u days after the disease prevalence decreases to 0 was labelled "assumed unhealthy"; all other data was labelled "assumed healthy". All data at this point was broken down into 30 minute "frames" using a sliding window (Figure 2.3).

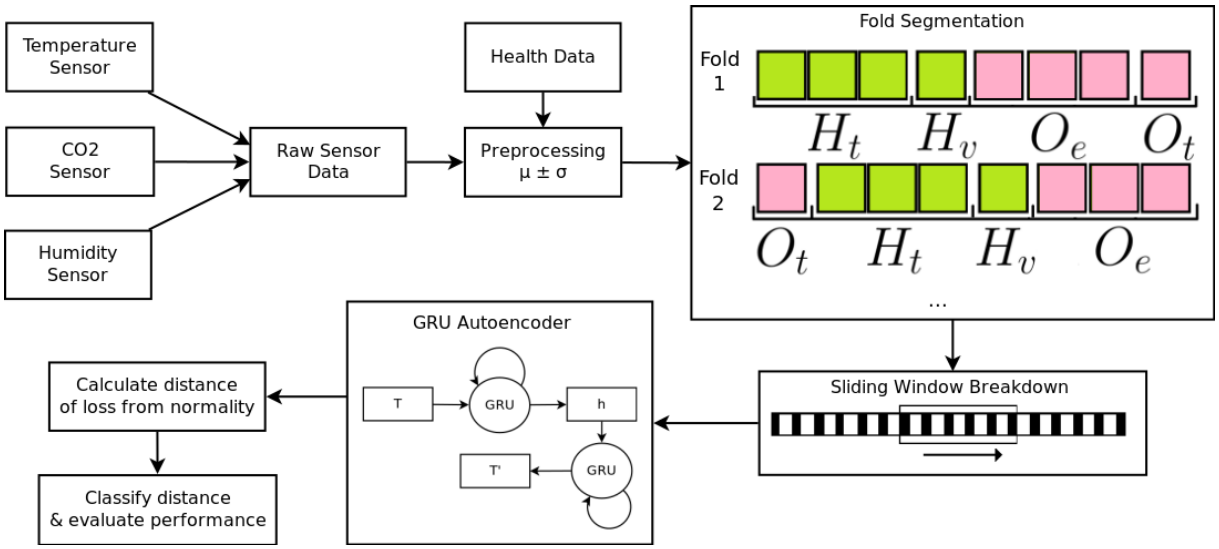


Figure 2.3 An overview of the entire processing workflow, from the raw sensor data to anomaly detection. Temperature, CO_2, and humidity sensors were preprocessed and merged with daily health data. This was then broken down into four folds—H_t and H_v (used for the GRU-AE), and O_e and O_t (used for the anomaly detection). Each of these folds was broken down using a sliding window sized at 30 min. Each window of 30 min was processed by the GRU-AE, and a reconstruction error was produced. The distance between this reconstruction error and normality was used to detect whether the window was anomalous or not using a threshold-based anomaly detector optimised using particle swarm optimisation.

Splitting Data into Train, Validation, and Test Sets

Leave-one-batch-out cross-validation was used to split the data into training and testing sets (Figure 2.3). Our method, detailed in Section 2.4.1, and the GRU-regression model (GRU-R) used for comparison, detailed in Section 2.4.3, were each composed of two parts: a GRU network and a threshold-based anomaly detector. Both the GRU network and anomaly detector required two sets of data for the full workflow. H_t, H_v were used for the GRU network, and O_e and O_t were used for the anomaly detector. H_t was used for the training of GRU networks and H_v was used to validate it. O_e was used to optimise an anomaly detector via PSO and O_t was the set the anomaly detector was tested on to attain an evaluation metric. There was no overlap between any of these sets.

O_t was a single, independent farm batch, O_e was composed of three randomly chosen batches, H_v contained only the assumed healthy data of another independent farm batch, and H_t contained only the assumed healthy data from all the remaining batches.

Each of the four datasets could be represented as a matrix $X \in \mathbb{R}^{n\times 3}$, where n is the number of minutes of data in a farm batch, and each X_n contains sensor data for temperature, humidity, and CO_2. This matrix was broken down into frames using a sliding window of size w, which moved one timepoint (1 min) per step, creating $T \in \mathbb{R}^{m\times w\times 3}$, where m was the number of frames in the set.

Preliminary Experiments Utilising Batch Normalisation

Batch normalisation is a deep learning technique that has shown to be very effective at training robust networks by performing a per batch normalization process on input data. It should be noted that the meaning of batch in this context differs from the farm batches used throughout the rest of the manuscript. In this context, batches are the subsets of the training data that are fed in blocks to the network being trained with the subsequent back-propagation-based error correction. In anomaly detection, the effect of this technique is not necessarily desirable and, in this particular application, based on preliminary experiments, was found to be detrimental to performance. This was determined to be because batch normalisation allows a model to deal with internal covariate shift by normalising each mini-batch by its mean and variance [137], meaning that the model is more capable of handling data it has never seen before. The decrease in performance when using batch normalisation showed that there is a requirement that the model be highly sensitive to slight variations in the data, and so it was not used in the final models.

2.4. Time-Series Early Warning Methods

2.4.1. GRU-Autoencoder

Autoencoders are trained to reconstruct the data input to them. Given an input tensor N, an encoding M is found which is used to create N'. The network is trained to minimise the loss between N and N'. The transformation of N to M is the encoder, and the attempted reconstruction of M to N' constitutes the decoder. Traditional autoencoders achieve this using an ANNs, where, typically, the M that is produced by this network has lower dimensionality than that of N. This way it acts as a type of dimensionality reduction technique. Most commonly the encoded output, M, is used as an input to some other model, such as an SVM, for tasks such as classification or regression. Autoencoders have shown good performance in assisting anomaly detection by using the feature representation M as the input to an OCC [138–140]. However, other research has shown that the reconstruction error between N and N' can be used to detect anomalous data [141].

In this particular paper, the authors applied a stacked denoising autoencoder [142] to a Latent Dirichlet Allocation (LDA) [143] representation of an SMS message, in order to determine if the text was “spam” or not. While the approach demonstrated the potential behind using the reconstruction error for anomaly detection, it did not make implicit use of the temporal aspect of natural language, relying instead upon a pre-processing step to extract important features; something that an RNN is capable of achieving when applied to raw data. Additionally, due to the nature of natural language, the method was only evaluated on univariate inputs.

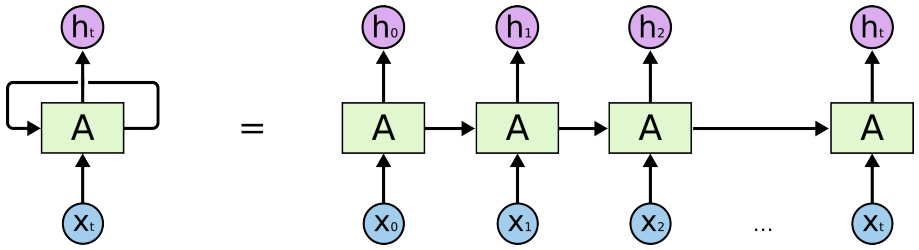


Figure 2.4 A visualisation of a rolled and unrolled recurrent neural network [2].

Unlike ANNs, the architecture of an RNN accounts for the temporality of data. Through the use of learned weights, each time-step in a time-series input is incorporated into a "hidden state" sequentially until all time-steps have been processed (Figure 2.4). At this point, the hidden state represents all of the data in the time-series. In the sense of an autoencoder, this would be the same as the feature representation M, which could subsequently be decoded back to the original time-series input to create an RNN-based autoencoder. This kind of recurrent architecture is known as a "vanilla" RNN, and although it constitutes a major milestone in deep learning research, this implementation faced challenges when trying to model long-term dependencies. If an application required knowledge of values far back in the input time-series, vanilla RNNs would typically underperform.

LSTM cells were introduced to solve the memory issues with vanilla RNNs, enabling recurrent architectures to be able to "remember" more than 1,000 time-steps of an input sequence [49]. This was achieved by adding a new state to work alongside the already existing hidden state called the cell state. Additionally, several "gates" were introduced to determine what data should be forgotten (the "forget" gate), what data should be remembered (the "input" gate), and what data should be output to the next cell (the "output" gate). This combination of an additional state and the introduction of gates allowed more granular control over how to process the input sequence, which is learned by the model through training.

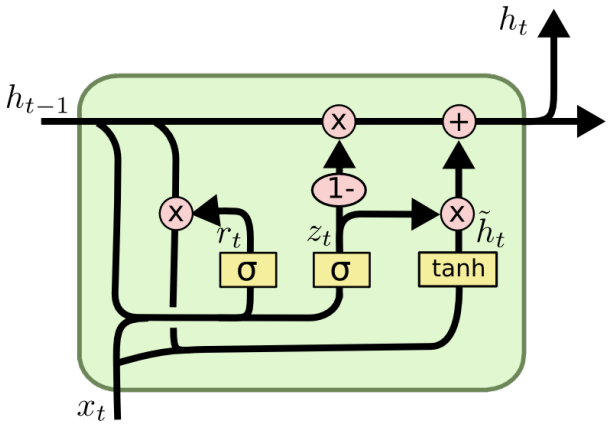


Figure 2.5 A visualisation of the internal architecture of a Gated Recurrent Unit [2].

$$z_t = \sigma(W_z \cdot [H_{t-1}, x_t]) \tag{2.2}$$

$$r_t = \sigma(W_r \cdot [H_{t-1}, x_t]) \tag{2.3}$$

$$\begin{aligned} \tilde{h_t} &= tanh(W \cdot [r_t * h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h_t} \end{aligned} \tag{2.4}$$

The improvements over vanilla RNNs that LSTMs introduced came at a cost. By increasing the complexity of the model, the training of LSTMs was slower. Further research resulted in the introduction of the Gated Recurrent Unit (GRU) [112] (Figure 2.5), which could be used in place of the LSTM cells. This implementation combined the "input" and "forget" gates to create the "update gate" (Equation 2.2), such that t is the current time point of the input sequence, W are the learned weights of the GRU, H is the hidden state and X is the input sequence. A "reset gate" was introduced to determine how much previously remembered information to forget (Equation 2.3), and the cell state was removed. The hidden state remained and was updated using Equation 2.4, such that z is the output of the update gate, and r is the output of the reset gate. This reduction in states and gates, along with other changes to how values are updated, simplified the processing of an input, which increased training speed, with limited impact on performance [144–147].

The implementation of the GRU-based autoencoder consisted of two multi-layer GRU networks — an encoder and a decoder — that were trained simultaneously. This is instead of the traditional ANN architecture (Figure 2.6), which does not intrinsically account for temporality in data within its architecture. The encoder learned a fixed-size representation M of an input N. The quality of this representation was measured by using it to attempt a reconstruction of N, $N\prime$, using the decoder. The model was optimised to minimise the reconstruction error, calculated using binary cross-entropy loss, between N and N'. N was reversed when calculating loss, as it creates short-term dependencies that result in an increased performance in sequence-to-sequence networks [148].

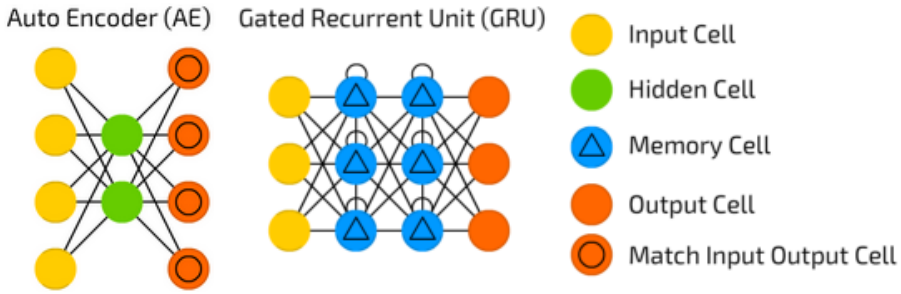


Figure 2.6 A visualisation of a classic autoencoder (**left**) and recurrent neural network consisting of gated recurrent units (GRUs) (**right**) [3]. These two methodologies are what we combined to create an autoencoder (AE) capable of utilising temporality.

H_t was used to train the GRU-AE using mini-batch RMSprop gradient descent [62]. This dataset contained only "assumed healthy" data, which resulted in the GRU-AE optimising towards a lower reconstruction error for this class of data and a higher reconstruction error for all other data. Hyper-parameters, hidden size, number of layers, mini-batch size, dropout, learning rate, and momentum were selected using grid search, and the parameters which resulted in the lowest reconstruction error for each fold's H_v were selected. The hidden size represents the amount of information the GRU-AE can store for each time series it is given. As there was no requirement for a large amount of memory for this task, the hidden size representation identified by the grid search was expected to be fairly small. The number of recurrent layers represents the capability of the GRU to create deep models and understanding of the time series. These are expensive in terms of computation and memory. The mini-batch size is how many sequences are passed forward through the network before back-propagating the loss. This is typically desired to be as large as possible to speed up training, though is entirely dependent on the other parameters and the amount of memory available. Dropout is a regularisation technique used to reduce the likelihood of over-fitting and is common in standard artificial neural networks. The learning rate and momentum are also used in standard artificial neural networks. They determine how drastically weights are changed during back-propagation. The selected hyper-parameters were then used to calculate reconstruction errors for all frames in O_e and O_t.

Once all reconstruction errors were calculated, the threshold-based anomaly detector was used to determine whether each reconstruction error of a frame was to be considered anomalous or not. This was estimated by calculating the Mahalanobis distance (Equation 2.5) between the reconstruction error of a given frame e and the distribution of the reconstruction errors for all frames of H_v, defined as R, where $\Sigma = COV(e,R)$. This was then smoothed using locally weighted smoothing (LOWESS) [149]. If the smoothed distance was above some threshold τ (the optimisation of which is described in Section 2.4.1), then it was considered an anomalous frame.

$$d = \sqrt{(e-\bar{R})^T \Sigma^{-1} (e-\bar{R})} \tag{2.5}$$

PSO-Optimised Anomaly Detection

Once all Mahalanobis distances were calculated for all frames, the next step was to optimise the threshold-based anomaly detector, capable of determining which frames should be considered anomalous, using PSO [150]—a meta-heuristic, which uses concepts of swarm intelligence to find a set of hyper-parameters that produce a near-optimal solution. In this method, an initial "population" of particles, that represent a set of potential solutions to a given "fitness" function, are randomly initialised with a fixed-length vector, representing the particle's position within a search space, and an initial "velocity". They are then each evaluated against the fitness function

and the best overall particle is saved as the "global best". If the score a particle achieves is the best it has achieved so far, it is saved as the "local best" for that particle. Each particle's position is then updated based upon its velocity, its local best, and its distance from the global best. This is repeated until some termination condition is reached, often a maximum or minimum fitness score, or a maximum number of iterations. The final global best particle represents a near-optimal set of hyper-parameters for the given fitness function.

For our application of PSO, consider the vector of Mahalanobis distances of all frames in a sequence $D = \{d_0, d_1, d_2, ..., d_n\}$, such that each d_i is the Mahalanobis distance between the reconstruction error of a frame at index i from the distribution of reconstruction errors resulting from the autoencoding of H_v. An alert was defined as any $d_i : d_{i-1} < \tau \wedge d_i \geq \tau$. The validity of an alert was assessed by determining whether respiratory disease prevalence increased from 0 within an acceptable forecasting window starting at $i + \alpha$ and ending at $\alpha + \beta$. If so, and if no other alert had been raised for this increase in prevalence, the alert was considered a true positive, else a false positive. Furthermore, consider the vector $F = \{f_0, f_1, f_2, ..., f_n\}$ such that each f_i contains the number of animals showing symptoms of respiratory disease at frame i. A false negative was determined to be any frame $i : f_{i-1} = 0 \wedge f_i > 0$ for which there is no alert raised within $i - \alpha$ and $i - \beta$.

Once a true positive is raised, no further true positives are counted for the period of time the first trust positive is raised. This is to stop models raising multiple "alerts" for the same period of time, artificially boosting performance metrics.

$$F = 2 \cdot MCC + \alpha - \beta \tag{2.6}$$

The parameters τ, α, and β of the detector were the parameters to be optimised using PSO. Although these parameters created a relatively small search space, τ being a continuous variable drove the decision behind using PSO over something more systematic, such as grid search, as it allowed for finer changes in τ to be evaluated. The particles were optimised for each fold independently in O_e, and the best values for each parameter were then tested on O_t for a final performance evaluation. Each particle was randomly initialised with three dimensions representing τ, α, and β. The fitness function (Equation 2.6) used by the PSO was designed to maximise both Matthews correlation coefficient MCC (Equation 2.7) and α whilst, at the same time, minimising β. Both α and β were normalised between 0 and 1 when entered into the fitness function.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(FP+FN)(TN+FP)(TN+FN)}} \tag{2.7}$$

MCC is a measure designed to evaluate the predictions of binary classifiers, which is especially robust in problems with high class imbalance, such as our dataset. MCC scores 1 for perfect classification, -1 for perfect misclassification, and 0 for the equivalent to random classification. The F-measure is commonly used in information/document retrieval, as it does not fall prey to the same problems, however, it does not include true negatives in the calculation. This is not an issue within document retrieval, as they are not important to that particular task; however, in a detection system, correctly classifying negatives (true negatives) is an integral objective of the system.

It was desirable to attain a high α, as this corresponds to how far in advance the alert of oncoming disease is raised, whereas β needs to be minimised, as this represents the period of time the alert is for. For example, given the values $\alpha = 7$ and $\beta = 1$, we know that when an alert is raised, it is expected that occurrences of respiratory disease will appear within 7–8 days. Given a different detector for the values $\alpha = 3$ and $\beta = 5$, we know that when an alert is raised, it is expected that occurrences of respiratory disease will appear within 3–8 days, a larger window and, therefore, more uncertainty. As we consider MCC to be the most important component of our fitness function, it has a larger weighting in Equation 2.6.

2.4.2. Other Metrics Used for Evaluation

Specificity, a commonly used evaluation metric, can produce misleading scores in datasets with a large number of negative examples (the typical scenario in anomaly detection tasks). Instead of specificity, we have used positive predictive value (also known as precision, Equation 2.8). For the sake of brevity, we identify this metric as precision throughout this chapter. We also made use of sensitivity (also known as recall, Equation 2.9), which we identify as recall throughout this chapter. We did not use precision or recall for the optimisation of the hyper-parameters of the detector. They were only used to evaluate the final performance of each method. It should be noted that neither of these metrics account for true negatives.

$$P = \frac{tp}{tp + fp} \tag{2.8}$$

$$R = \frac{tp}{tp + fn} \tag{2.9}$$

2.4.3. Methods Included for Comparison

All methods used were tested on identical test sets O_t. Once all methods for comparison were carried out, a Wilcoxon test was conducted to compare the precision, recall, and MCC between

the three comparison methods and the GRU-AE. These results were corrected for multiple observations using Holm's method [151].

Luminol

In order to provide a baseline for comparison of performance, LinkedIn's open-source, univariate time-series anomaly detection package, Luminol [152], was used. This library represents out-of-the-box anomaly detection, as well as methods which require data be univariate, often achieved using principal component analysis (PCA). This anomaly detection method uses a bitmap representation of a univariate time series, calculated using symbolic aggregate approximation (SAX)—a method for the discretisation of time-series data [153]—to detect anomalies [154]. This is implemented using two concatenated windows, a lagged window and a leading window, which slide across the time series. Each window is then converted into a SAX representation and subsequently into bitmaps based on the frequency of each SAX "subword". The distance between the two bitmaps is calculated and represents the anomaly score for the leading window. The calculated anomaly scores are then tested using a grid search-optimised threshold-based anomaly detector in the same way that the GRU-AE is assessed. In order to transform the three environmental variables into a univariate time series, the first principal component was taken.

Time-Series Regression with Autoregression Integrated Moving Average (ARIMA)

Autoregression integrated moving average (ARIMA) is a frequently used model for univariate time-series analysis in statistical analysis. The parameters for this model (p, the order of autoregressive model; d, the degree of differencing; and q, the order of the moving average model) were found using grid search on H_t, optimising for the lowest Akaike information criteria (AIC). This is a metric used for comparing the goodness of fit of statistical models, though it is not a measure of goodness of fit. Once the optimal parameters were found, H_v was used to fit the model. This was then used to forecast 30-minute windows, given the previous 180 minutes in O_e, a larger input window than is used for the GRU-AE. Windows where the mean square error between the predicted window and ground truth was beyond some threshold (τ) were considered an alert for the window $i+\alpha$ to $i+\alpha+\beta$. Values for τ, α, and β were optimised using grid search and, finally, tested on O_t. Similar to the Luminol method, the first principal component was taken to transform the three environmental variables into a univariate time series.

Time-Series Regression with GRU Network (GRU-R)

To provide context for how the GRU-AE performs with regards to an alternative deep learning methodology, a GRU network was used for regression. This network was trained to predict future values of the environmental sensors. By training this regression network on only healthy data H_t,

it learned to predict environmental variables for conditions which do not result in occurrences of respiratory disease. The model was trained to predict windows of 30 minutes, the same size as the windows used for the GRU-AE, but used the prior 180 minutes of data to make this forecast, similar to the ARIMA model. Windows where the mean square error was beyond some threshold (τ) were considered an alert for the window $i+\alpha$ to $i+\alpha+\beta$. Similar to the configuration laid out in Section 2.4.1, these parameters were optimised using PSO, which is trained using O_e, and finally evaluated on O_t.

2.5. Results

For the purpose of exploration, the GRU-AE's anomaly detector was initially optimised using a grid search with a fixed $\beta = 6$ with O_e used as the test set. For values from $\alpha = 10$ onwards, there was a steady and sustained decrease in the total number of alerts raised (Figure 2.7). This was indicative of the point where the acceptable forecast window extended beyond the data available, resulting in an increased portion of alerts that were unverifiable in terms of validity. This decline in alerts signified the maximum range of the acceptable forecast window $\alpha+\beta = 16$. This maximum endpoint was used to reduce the search space of the PSO, and, therefore, a particle was considered invalid if $\alpha+\beta > 16$. The search space was also restricted by stipulating that a particle was also invalid if it exceeded the following ranges: $0 \leq \tau \leq 20$, $1 \leq \alpha \leq 12$, and $1 \leq \beta \leq 6$. The limits on τ cover all feasible values; this was found through preliminary testing. β was limited to a maximum of 6, as a range greater than this was deemed to be too large.

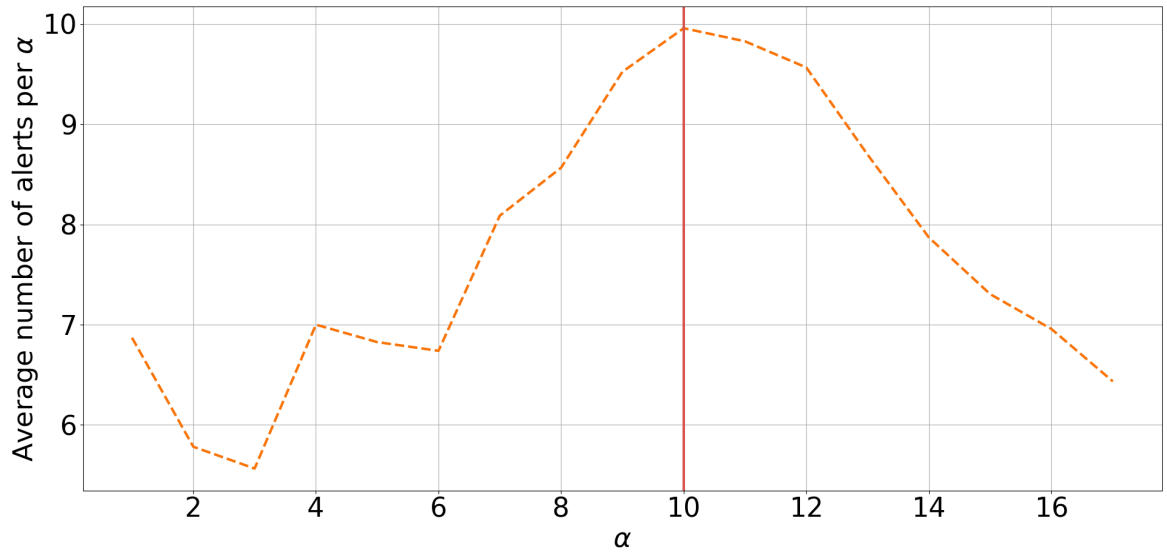


Figure 2.7 The average number of alerts raised, given an alert window start of α days and a size of $\beta = 6$, using the optimal threshold τ for each given α found using grid search on the anomaly detection used on Mahalanobis distances produced by a GRU-based autoencoder.

As outlined in Section 2.4.1, grid search was used to find the hyper-parameters for the GRU-AE. The grid search converged to the same set of parameters in almost all batches, so, for the sake of space, we report in Table 2.1 the most frequently identified values for all hyper-parameters. As expected, the hidden size was very low. This is because under healthy circumstances there is little fluctuation in the environment, and, therefore, only a small amount of memory is required. It is when there are deviations from this—frequent or drastic fluctuations—that occurrences of respiratory disease can be expected.

Hyper-Parameter	Value
Hidden size	4
No of layers	20
Mini-batch size	1024
Dropout	0.5
Learning rate	1×10^{-6}
Momentum	0.3

Table 2.1 The grid search-optimised hyper-parameters used to train a GRU-AE used on multidimensional time-series data.

Table 2.2 shows the batch-by-batch performance of each of the three methodologies; each row presents the performance of the three compared systems when a given batch is used for testing. The GRU-AE achieves better results than the other methods in terms of recall and MCC, though is outperformed by ARIMA and the GRU-R in terms of precision. When looking at the results of batches where there are cases of disease (i.e. "events > 0"), a higher precision is achieved in comparison to recall in the ARIMA and GRU-R models, indicating that these models are more conservative in their alerts. However, in the GRU-AE model, the solutions found tend to rely on maximising recall at the expense of precision, $\mu_P = 0.651, \mu_R = 1.000$. The substantially higher recall across all batches for the GRU-AE model indicates the model's capacity to identify negative changes in the environment that lead to an increase in the number of pigs with symptoms of respiratory disease; the lower precision of the model could indicate that not all negative changes in the environment result in occurrences of respiratory disease. This is backed up by the fact that there are other factors [155] that contribute to respiratory disease prevalence, not just the environmental measures used in this analysis.

Looking only at test batches where there were no events, the deep learning models excel in determining that there are no alerts to be raised, achieving the highest values for precision and recall. This indicated that the GRU is capable of modelling the underlying representation of "normality" very well, as alerts are not raised when they should not be. Comparing the batches where events = 0 and batches where events > 0, both of the GRU-based models decreased in performance in terms of precision, but only the GRU-AE was able to retain its high recall.

ID	Events	Luminol			ARIMA			GRU-R			GRU-AE		
		P	R	MCC	P	R	MCC	P	R	MCC	P	R	MCC
5	0	1	1	1	1	1	1	1	1	1	0	1	0
9	0	0	1	0	1	1	1	1	1	1	0	1	0
10	0	0	1	0	1	1	1	0	1	0	1	1	1
14	0	0	1	0	1	1	1	1	1	1	1	1	1
16	0	0	1	0	0	1	0	1	1	1	1	1	1
17	0	0	1	0	1	1	1	1	1	1	1	1	1
18	0	0	1	0	0	1	0	1	1	1	1	1	1
19	0	1	1	1	1	1	1	1	1	1	1	1	1
24	0	1	1	1	1	1	1	1	1	1	1	1	1
26	4	1	1	1	0.053	0.500	0.162	1	0	0	0.250	1	0.500
27	5	0.031	0.667	0.144	1	0	0	0	0	1	0.111	1	0.333
28	3	1	0.667	0.816	1	0	0	1	0	0	1	1	1
29	1	1	1	1	1	1	1	0	1	0	0	1	0
30	0	0	1	0	0	1	0	0	1	0	1	1	1
31	1	0	1	0	1	1	1	1	1	1	0.050	1	0.224
32	0	0	1	0	1	1	1	1	1	1	1	1	1
Avg. Events = 0		0.273	1.000	0.273	0.727	1.000	0.727	0.818	1.000	0.818	0.818	1.000	0.818
Avg. Events > 0		0.606	0.867	0.592	0.811	0.500	0.432	0.600	0.400	0.400	0.282	1.000	0.411
Avg. All		0.377	0.958	0.373	0.753	0.844	0.635	0.750	0.813	0.688	0.651	1.000	0.691
I/B/C				9/2/5			5/1/10			5/0/11			3/3/10

Table 2.2 Table of results for the grid search-optimised implementation of LinkedIn's anomaly detection library (Luminol), the grid search-optimised autoregression integrated moving average (ARIMA) model, particle swarm optimisation (PSO)-optimised threshold-based anomaly detection of loss incurred from a GRU-based regression, and PSO-optimised threshold-based anomaly detection of Mahalanobis distance produced by a GRU-based autoencoder. The events column lists how many times the respiratory disease prevalence increased from 0 within the batch, P denotes the precision, and R is the recall of the model. These results are from the test data O_t. The final I/B/C row indicates the number of folds that were Incorrect ($MCC = 0.0$), Between ($0.0 < MCC < 1.0$), or Correct ($MCC = 1.0$)

The last row of Table 2.2 contains a high-level quantitative summary of the performance of all methods across batches. We have counted the number of batches in which each method obtains either an incorrect score of 0 for MCC (I), correct score of 1 for MCC (C), or in between 0 and 1 (B). In terms of the C count, Luminol clearly performs the worst of the methods, while the others have a similar score (10–11). At the same time, GRU-AE stands out as having the lowest count of I, three times lower than Luminol's, whilst simultaneously keeping a low count of B.

Numerically, at least, we can see that the difference in performance between Luminol and GRU-AE may be considered substantial. However, the results from the Wilcoxon test, implemented as described in Section 2.4.3, showed that the difference was not statistically significant for precision ($p = 0.325$), recall ($p = 0.100$), and MCC ($p = 0.285$). The Wilcoxon test also concluded there was no significant difference between the ARIMA model and GRU-AE, and GRU-R and GRU-AE models, in terms of any of the performance metrics. The inclusion of a statistical test to compare the performance of various architectures on a given task is not particularly common in deep learning research, despite it being a valuable step in presenting performance data transparently. Future research in deep learning should consider this when presenting multiple methods for comparison, regardless of whether one method appears to drastically outperform the others or not. The optimised α and β values for each batch are reported in Table 2.3.

ID	Events	Luminol		ARIMA		GRU-R		GRU-AE	
		α	β	α	β	α	β	α	β
5	0	12	4	12	1	1	6	1	1
9	0	12	4	12	1	1	6	1	2
10	0	12	4	12	1	1	6	1	2
14	0	12	4	12	1	1	6	1	4
16	0	12	4	12	1	1	6	3	1
17	0	12	4	12	1	1	6	1	5
18	0	12	4	12	1	1	6	3	5
19	0	12	4	12	1	1	6	1	5
24	0	12	4	12	1	1	6	1	6
26	4	12	4	12	1	2	6	3	6
27	5	12	4	12	4	4	6	1	6
28	3	12	4	12	1	5	6	8	6
29	1	12	4	12	2	1	6	2	6
30	0	12	4	12	1	1	6	4	6
31	1	12	4	12	1	1	6	1	6
32	0	12	4	12	1	1	6	8	5

Table 2.3 Table of results for the grid search-optimised implementation of LinkedIn's anomaly detection library (Luminol), the grid search-optimised ARIMA model, PSO-optimised threshold-based anomaly detection of loss incurred from a GRU-based regression, and PSO-optimised threshold-based anomaly detection of the Mahalanobis distance produced by a GRU-based autoencoder. α and β denote the start and length of the time window (in days), respectively, for which an alert is assessed, and τ is the threshold the loss/Mahalanobis distance needed to exceed in order to raise an alert. They are optimised using training data O_e; these results are from the test data O_t.

2.5.1. Case Study

The following case is presented for batch 27 exemplifies the characteristics of each of the methodologies applied to the problem. This batch was chosen as it contained the highest number of anomalous events in a single batch, making it one of the more challenging batches for a method to perform well on. The batch was analysed by a Luminol-powered, grid search-optimised anomaly detection (Figure 2.8), an ARIMA-powered, grid search-optimised anomaly detection (Figure 2.9, a GRU-R-powered, PSO-optimised anomaly detection (Figure 2.10), and a GRU-AE-powered, PSO-optimised anomaly detection (Figure 2.11). The first subplot for each figure shows respiratory disease prevalence in both analogue and binary form; the second plots the univariate time series that was used for anomaly detection and the alerts raised for it. In the case of the Luminol model, this was the first principal component of the temperature, humidity, and CO_2 time series. For the GRU-R and ARIMA models, the mean square error between predicted and ground truth was used; for the GRU-autoencoder, the binary cross-entropy loss between each input window and the model's recreation of it was used. The final subplot shows the results of the anomaly detection for each of the alerts, along with their alert windows.

The two deep learning-based models show signs of consistency in the underlying technology, as they both capture similar events within the environmental data, demonstrated by the common peaks and troughs seen in the loss. The Luminol-based model correctly produces alerts to some of the occurrences of respiratory disease prevalence, but these are severely outweighed by the

number of false positives that it also produces. The GRU-R and ARIMA models suffer from the inverse problem in that no alerts are raised, even though the latter appears at first glance to produce some significant anomalous points. These points rarely align with actual events, and the success of the ARIMA model is largely down to not raising many alerts at all. The GRU-autoencoder model, however, strikes a balance between these solutions, alerting to almost all of the occurrences of respiratory disease, albeit with some false positives, though far fewer than the Luminol model.

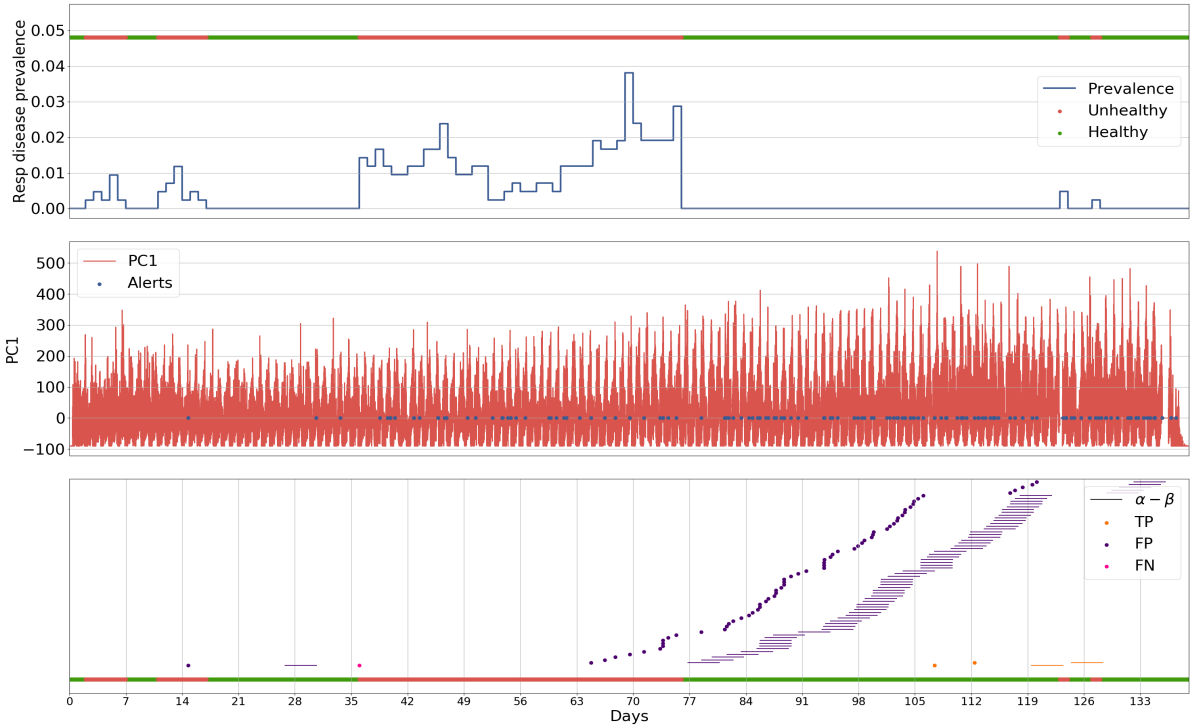


Figure 2.8 Results for batch 27 processed by the Luminol-powered anomaly detection, given its grid search-optimised parameters α (the start of the window for which an alert is for) and β (the length of the window), in relation to the prevalence of respiratory disease within a single batch.

Figure 2.9 Results for batch 27 processed by the ARIMA-powered anomaly detection, given its grid search-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

Figure 2.10 Results for batch 27 processed by the GRU-R-powered anomaly detection, given its PSO-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

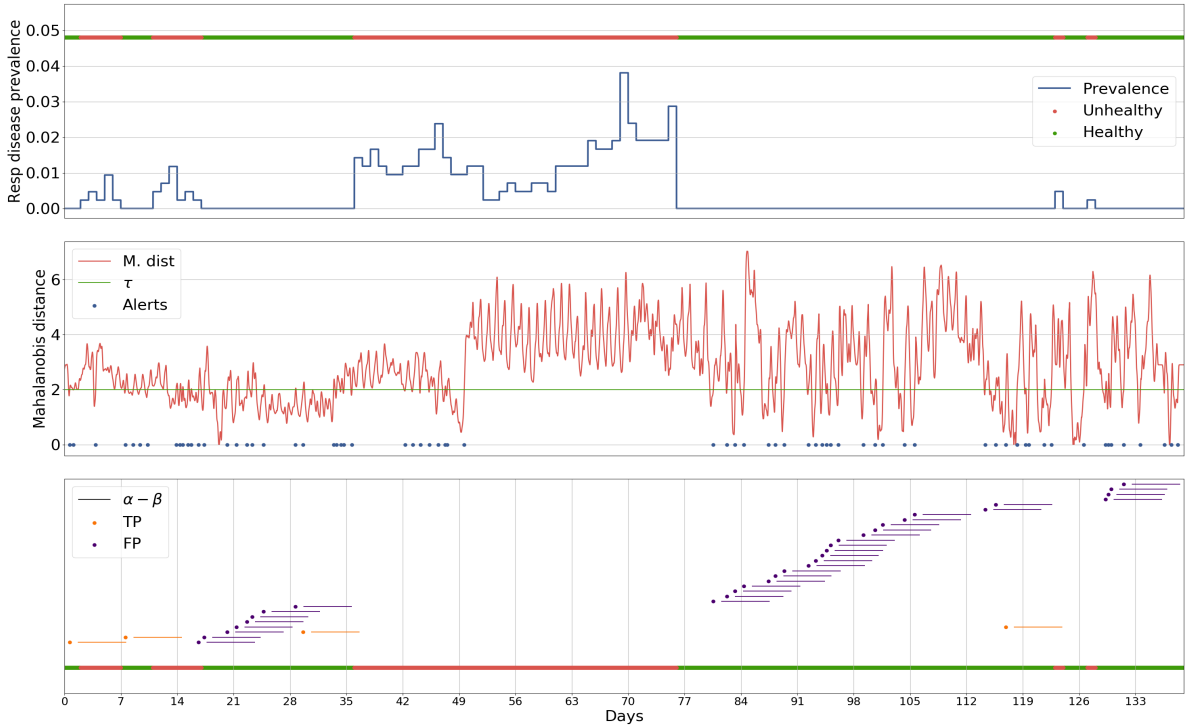


Figure 2.11 Results for batch 27 processed by the GRU-AE-powered anomaly detection, given its PSO-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

2.5.2. Influence of the Number of Hidden Layers on GRU-AE Performance

The number of layers used in the encoder and decoder affected what the GRU-AE was capable of modelling (Figure 2.12). Since the concept behind the model was that loss is reduced for "healthy" data, the perfect environment's loss would have a constant loss over time, therefore meaning the model architecture should be chosen to minimise the standard deviation of the loss in healthy data. Using a small number of layers, 1–15, there was little change in the ability of the GRU-AE to model "healthy data". Once the number of layers reached 20 for each half of the GRU-AE, the performance reached its maximum and would not reduce the standard deviation of loss any further.

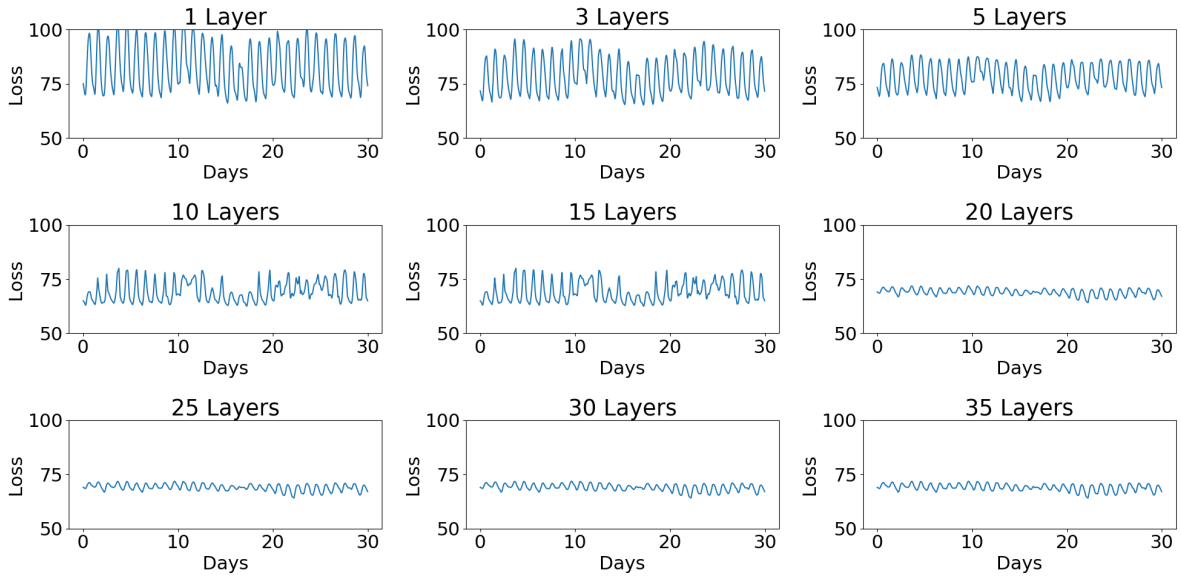


Figure 2.12 How different numbers of layers in a GRU-AE affect the ability to model environmental data. The smaller the standard deviation, the better the model performs.

2.5.3. Computation Time

Despite its lacking in performance, the Luminol-based model presented (Figure 2.8) was completed in a fraction of the time required to train the two GRU models presented. The Luminol model was fully trained within a period of 6 h and the ARIMA-based model was fully trained within 24 h. These models were run on a machine containing two Xeon E5-2699v4 processors (totalling 44 cores), whereas the two GRU models took just over 3 days on a similarly performing machine equipped with an Nvidia GTX 980 Ti.

2.6. Discussion

This chapter set out to develop and evaluate a methodology utilising recurrent neural networks comprising GRUs, and to apply the methodology to the early detection of respiratory disease in growing pigs. This network structure was used to build an autoencoder capable of accounting for the temporal aspect of data within the structure of the network itself, rather than solely through the structuring of the data. The GRU-AE specifically outperforms the GRU-R where there were events to be detected, resulting in a substantially higher recall for the GRU-AE.

The novelty of this work lies in the complete reassessment of the idea of a recurrent autoencoder for multi-sensor anomaly detection of Malhotra et al. [113]. The primary adjustments made were switching from LSTM cells, currently the most common unit in recurrent networks, to GRUs. This benefited the network as a whole, as GRUs are less complex, and therefore more efficient, whilst providing comparable performance [144–147]. A PSO-optimised anomaly

detector was also created for processing the output of the GRU-AE to determine if a window was anomalous or not. In addition to the methodological changes, this work provides a much more in-depth evaluation of the performance of the GRU-AE through the use of a case study and concrete comparisons with alternative methods, whilst also applying the methodology to real-world data from operating, commercial farms across Europe.

Given the recent trends in livestock keeping, there is a move towards automating significant portions of livestock management and monitoring in response to the ever-increasing demand for high-quality, sustainable, and safe meat production [156]. These changes in livestock management offer opportunities for automation in disease detection, such as the one developed here. A common approach to embracing precision livestock farming is the introduction of a variety of sensors. Small, relatively cheap, and high-resolution sensors can monitor environmental variables, such as those assessed in this study, and also capture large quantities of other data using sensors such as cameras [157] and microphones [158]. In the case of disease detection, any false negatives have the potential to lead to epidemics and therefore increase the disease risk of livestock systems. For this reason, a high precision ought to be sacrificed if a higher recall can be achieved, as false positives pose less danger than false negatives in this application. The GRU-AE presented in this chapter acts upon this, ensuring that recall is maximised as much as possible.

The motivation for developing such a methodology was to deal with the very low disease prevalence in the data that was collected. In this work, we concentrated on the detection of respiratory diseases. There were several reasons for this, including the fact that the prevalence of respiratory diseases was highest amongst this class of pigs in this dataset. We had similar data for other disorders that affect pigs, such as lameness and digestive diseases, but as their incidence was much lower than respiratory diseases, developing similar methodologies would have resulted in an insufficient number of occurrences to validate the model.

Steps were taken to mitigate the challenges posed by the data, such as summing all severity levels of respiratory disease (light, mild, severe, and irrecoverable), as described in Section 2.3.2. This was justified, as this methodology is for an early-warning system, which only needs to know if there is disease or if there is not. Therefore, looking at respiratory disease prevalence as a whole, rather than segmented into levels of severity, makes for a simpler dataset that is more focussed on the challenge being tackled. It should be noted that the methodology developed here can be applied to all types of respiratory diseases in pigs.

The detection of respiratory disease following a change in the environmental condition fits within the parameters derived from literature, such that we are not predicting oncoming disease before pathogen incubation (e.g., PRRS has an average incubation time of 14 days [136]). Both the GRU-R and GRU-AE are able to raise an alert for oncoming respiratory disease 1–7 days prior to pigs in a batch first showing symptoms. The Luminol model does not reflect this, however, this is likely due to its inability to find hyper-parameters that result in a strong MCC.

This causes the PSO to maximise the values for α and β, as there is no repercussion for doing so if the performance is poor regardless.

Due to the renewed popularity of recurrent-based models for sequence analysis, a result of their exceptional performance in fields such as natural language processing, there is a need for methods to increase the interpretability of these models. Work in this area has steadily grown [159, 160], and any work building upon this research might consider applying such methods to the model to gain a better understanding of the risk factors for respiratory disease in growing pigs based on the GRU-AE. As this method is assumption-free and works without considering differences between countries, additional work to validate that this multi-sensor anomaly detection scales beyond the domain of health within agriculture, thus validating its robustness, would be valuable to the wider community. However, the GRU-based models required substantially more resources than their classical statistical counterparts to achieve the necessary level of performance, therefore research ought to be carried out to address these arguably excessive requirements.

2.7. Conclusions

We have presented a deep learning-based model capable of raising an early warning to occurrences of respiratory disease in growing pigs. The proposed methodology of a GRU-based autoencoder combined with a PSO-optimised anomaly detector shows strong potential in its ability to detect anomalies that will lead to occurrences of respiratory disease in growing pigs, and it is robust to country-specific environments. This GRU-AE was able to outperform other comparable methodologies in precision, recall, and overall performance in both scenarios where there was a presence or an absence of disease.

Chapter 3. Deep Learning Architectures for Anomaly Detection In Multivariate Time Series Data

Abstract

Recurrent neural networks have long been touted as the best solution for modelling temporal data due to the fact that the concept of temporality is built into their architecture. However, because of this architectural design, the data must be processed sequentially, making parallelisation impossible, which has a substantial impact on performance for both training and inference. In our previous chapter, we proposed an architecture, comprised of two RNN-based models using GRU cells, to autoencode multivariate environmental sensor data for the purpose of anomaly detection. Although our architecture outperformed both classical statistical methods as well as another deep learning-based approach, the amount of training time required made it difficult to recommend for real-world use. In this chapter, we evaluate CNN and transformer-based architectures that could replace the GRU-AE in order to lower resource requirements, improve training time and still be able to reliably detect anomalies. We show that although CNNs have traditionally been used to solve spatial problems, they can be well suited to temporal applications through the use of causal convolutions. In our implementations, both the CNN and transformer-based architectures train faster and use less memory than the GRU-AE, though only the CNN-based architecture was able to maintain performance on our dataset.

3.1. Introduction

In the previous chapter we evaluated how a GRU-based autoencoder (GRU-AE) can be used to model multivariate time series data and thus be used to detect anomalies. The proposed GRU-AE was trained to create a robust representation of the multi-sensor data and outperformed alternative methods. However, this implementation relied on a recurrent neural network architecture, which is more computationally expensive than alternative deep learning architectures, as it is difficult to parallelise an inherently sequential task, which results in a performance bottleneck. At the time of writing the previous chapter, RNN-based approaches, typically using LSTM cells and often including attention mechanisms, were generally considered the best approach to time-series. Since, further research has shown that CNNs may be viable alternatives to RNNs in temporal applications [67]. Furthermore, transformer architectures have largely replaced RNN-based

architectures in NLP [161, 162], which is fundamentally a sequence processing task. Therefore, this chapter focussed on evaluating how convolutional and transformer-based architectures, both of which are non-recurrent architectures, can replace the GRU-AE used in the previous chapter to improve resource efficiency, whilst minimising the impact on anomaly detection performance.

Though CNNs have traditionally been used to solve various computer vision challenges such as multi-label classification [163] and object detection [81], they have since been used in a broader range of problem areas such as NLP [164] and time series classification [165]. More recently, transformers, an architecture consisting of "self-attention" layers [53], have been used with great success within NLP; outperforming both recurrent and convolutional-based networks [161, 166]. The transformer architecture has been particularly successful due to its ability to better model the context of inputs. For example, BERT [161], uses transformers to learn bidirectional representations of natural language, which gives it a deep contextual understanding of input sequences. More recently, GPT-3 was introduced [162], which uses a very large transformer architecture to create a comprehensive language model. This model can then be fine-tuned to achieve outstanding performance in various NLP challenges. Unlike recurrent architectures, CNNs and transformers do not have the concept of temporality explicitly built into their architecture and are therefore expected to learn this concept implicitly through training. This increases the challenges in training, as there is more that needs to be learned, but can simultaneously improve raw training speed as the task is no longer inherently sequential, allowing tasks such as sequence prediction to be parallelised.

This chapter focused on two models in particular, one CNN-based and one deriving from the original transformer. The CNN architecture evaluated was Sequential-U-Net (Seq-U-Net) [5], which uses causal convolutions to auto-encode time series data. It is a variant of the UNet architecture that was originally developed for medical image segmentation [4]. This method is described in more detail in Section 3.2.2. The transformer-derived architecture was Dual Self-Attention Network (DSANet) [7], which uses two branches containing self-attention blocks alongside an auto-regressive branch to forecast multivariate time-series data. This is a relatively unexplored application of transformers as most of their popularity comes from their applications in NLP. DSANet is described in more detail in Section 3.2.2

The main contribution of this chapter was in demonstrating how these methods are less resource-intensive to train with minimal performance compromises. This research makes use of identical training and testing data to the previous chapter in order to be able to make a direct comparison. The remainder of this chapter is structured as follows. Section 3.2 outlines the data, models and training methods used, where Section 3.2.2 describes the two methods evaluated in more detail. The results of this evaluation are presented in Section 3.3, followed by the discussion in Section 3.4 and conclusion in Section 3.5.

3.2. Materials and Methods

3.2.1. Data Description

The dataset used in this chapter was identical to the data used in the previous chapter. It consisted of environmental sensor data (temperature, relative humidity and CO_2 levels), which were recorded every minute and respiratory disease prevalence, which was recorded every day. The daily health data was remapped to minutely data by repeating values. The data was collected from a number of farms across Europe, where each farm had multiple "batches" of pigs, each of which consisted of 40-1294 pigs and lasted for 14-145 days.

Any timepoint in our dataset that had a respiratory disease prevalence of 0 and was not followed by an increase in respiratory disease prevalence within 2 weeks was referred to as "assumed healthy" Timepoints that were followed by an increase in respiratory disease prevalence within 2 weeks were referred to as "assumed unhealthy", along with any timepoints that coincided a respiratory disease prevalence of greater than 0.

Leave-one-batch-out cross-validation was used to train and evaluate both the models and the anomaly classifier, such that each "batch" was used as the final test set once. For each fold, the dataset was broken into four sets:

- H_t: the data used to train the deep learning models (only "assumed healthy" data).
- H_v: the data used as the baseline for normality to compare test data to (only "assumed healthy" data).
- O_e: the data used to optimise the parameters for the classifier.
- O_t: the data for an entire single batch of pigs for the test set.

The sets were then broken down further into 30 minutes segments using a sliding window, each of which constituted a single input to the model. The order of these windows was randomised for training.

3.2.2. Models and Training Methods

This section describes the two models, Seq-U-Net and DSANet, that were evaluated against the GRU-AE along with a description of the training methodology used for each model. The difference in model performance was tested for statistical significance using Wilcoxon tests that were corrected for multiple observations using Holm's method, as was done in the previous chapter.

Seq-UNet

Seq-U-Net [5] was a development on top of the original U-Net [4] architecture, which was designed for the purpose of biomedical image segmentation. Visualised in Figure 3.1, it was constructed as two main components that come together to create a single, fully-convolutional network (i.e. there are no fully connected layers). The first component, referred to as the contracting path, followed the standard pattern of a CNN that is used to extract a feature map from an input tensor (i.e. convolutional layers, using a ReLU activation followed by max pooling). The second component, referred to as the expansive path, took the feature map resulting from the contractive path and up-scaled it to the size of the original input. However, rather than just using transposed convolutional layers, where filters increase the size of the input rather than decrease it, the up-scaling also concatenated the feature map from the corresponding layer in the contracting path.

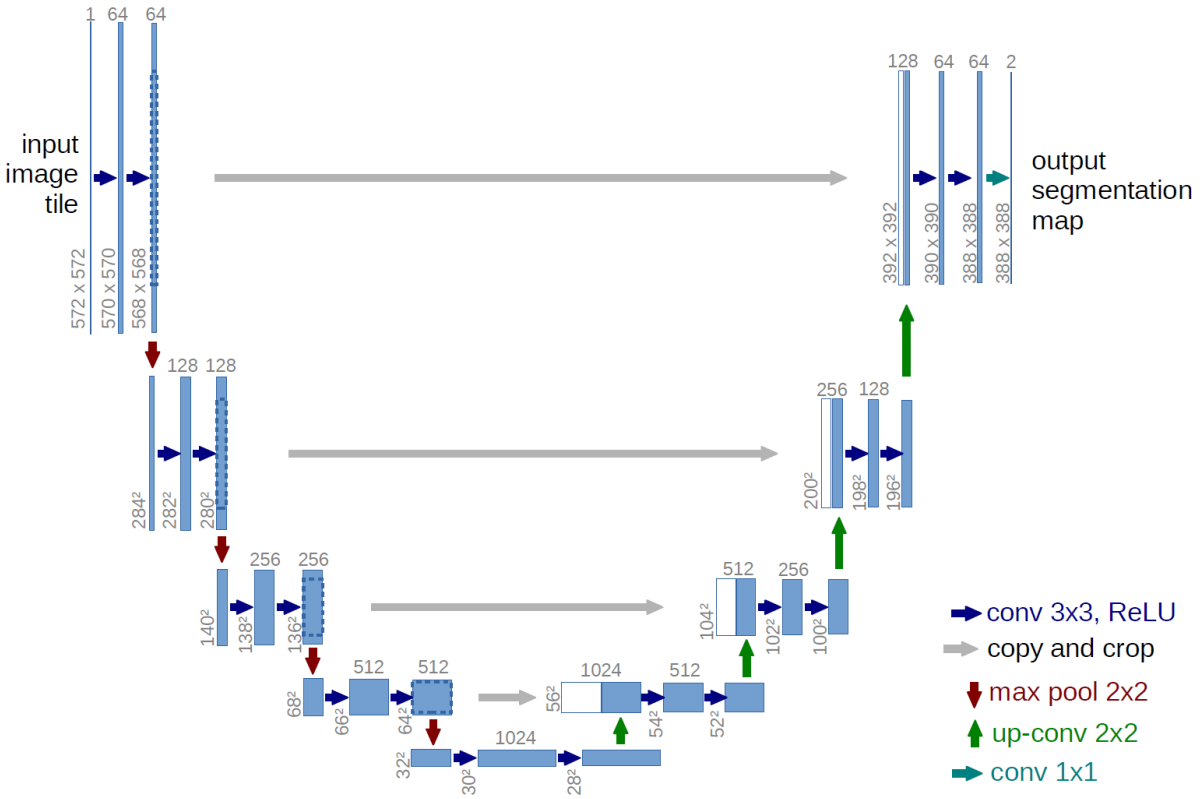


Figure 3.1 A visual representation of original U-Net architecture used for biomedical image segmentation [4].

As image data is non-directional, it is acceptable for a filter applied to a region of an image to use all values, as all surrounding values provide context. However, as time-series data is directional, when applying filters to a value at a given timepoint, the filter must not make use of any values from future timepoints. Therefore the implementation of Seq-U-Net adapted the underlying architecture of U-Net, such that the processing of time-series data using the convolutional filters applied at time point t did not include values from time points greater than t.

This was achieved by "padding" any filters that would otherwise process future values with zeros in order to exclude the values. This adapted architecture achieved similar performance metrics to models such as WaveNet [167] and Temporal Convolutional Network (TCN) [168] but with substantially lower resource requirements in terms of both time and memory.

In the implementation of the work in this chapter, Seq-U-Net was trained in the same manner as the GRU-AE from the previous chapter. The model was trained to recreate 30 minute windows of H_t, using a binary cross-entropy loss function, and subsequently, inference was performed on H_v. The reconstruction errors produced by this inference were used as the baseline for "normality". If the Mahalanobis distance between a new time window's reconstruction error and the distribution of reconstruction errors created by H_v increased beyond a certain threshold, τ, then that time window was determined to be anomalous. From this, it was possible to say that occurrences of respiratory disease prevalence were likely to occur within α to $\alpha + \beta$ days. The reconstruction errors for all time windows in O_e were calculated and the three parameters, τ, α and β were optimised using particle swarm optimisation (PSO). Finally, the model trained on H_t and classifier optimised using O_e were evaluated on O_t, and a corresponding precision, recall and MCC were calculated.

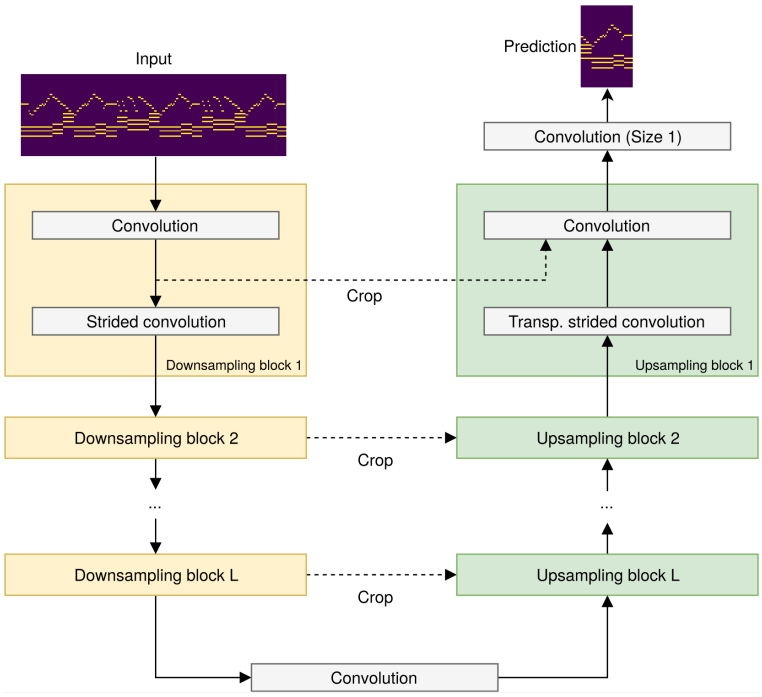


Figure 3.2 A visual representation of the Seq-U-Net architecture used for time-series autoencoding [5].

DSANet

The concept of "attention" in the encoder-decoder structures [169], that are commonly used in sequence prediction [170], was introduced as a means to overcome the limitations of the original LSTM and GRU-based recurrent architectures. The primary one being that all of the information contained in the input sequence had to be reduced to single hidden state, (i.e. the final hidden

state). Attention solved this by using "attention weights" in conjunction with all hidden states, rather than just the final one. These learned weights allowed for certain parts of all hidden states to have different levels of impact at each timepoint in the sequence, allowing for much longer-term memory. This came at the cost of larger model sizes and greater computational complexity.

Fundamentally, hidden states are simply approximations of an input sequence. Research in the area of question-answering models [6], which are models that are trained to answer questions about a body of text, took this viewpoint and showed that attention could instead be directly applied to the input sequence (Figure 3.3). This removed the need for hidden states altogether and conceptually replaced them with "Query", "Key" and "Value" vectors.

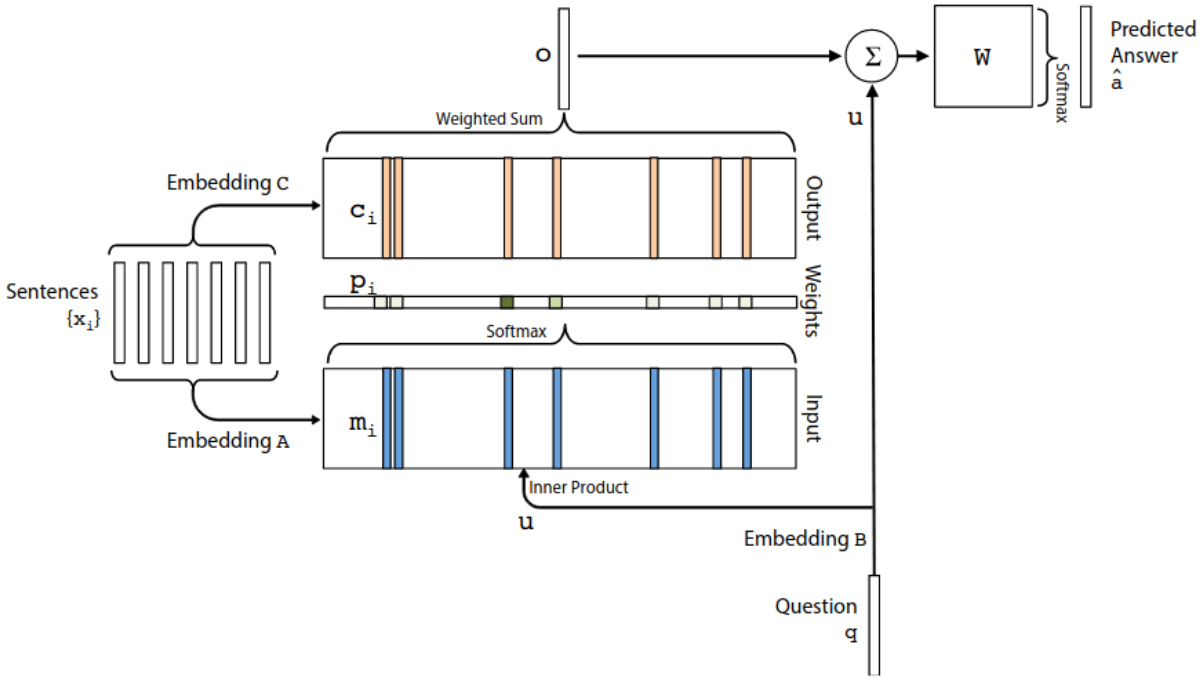


Figure 3.3 A visual representation of original End-to-End Memory architecture used for question answering [6]

.

Question-answering models are given a body of text and a question about that text as input and are trained to generate a correct answer to the question. Using this architecture, we can create an analogy of what these new vectors represent. In this application, the "Key" and "Value" vectors are two different encodings of the input body of text, created by two different sets of weights, one for each vector. The "Key" vector is used to approximate the relationship between the input and the query, whereas the "Value" vector represents the input. The "Query" vector is an encoding of the question that is being asked about the body of text. From these new vectors, the "Attention" weights that used to be applied to hidden states, became the inner product of the "Query" and "Key" vectors, instead of an independently trained set of weights. The sum of the "Value" vector, weighted by this new attention vector, gave the final output sequence from the model, the answer to the question. The use of these vectors in this way allowed sequence

forecasting to overcome their sequential nature. Instead, the problem is presented as a memory querying (e.g. lookup) task, where a representation of the query is "looked up" in a representation of the body of text in order to produce an answer.

From this foundation, the transformer architecture was developed which described “self-attention” [53]. This refers to allowing a model, too look at other words in a sequence to influence how the current word being processed is encoded. This concept was developed further, with the introduction of the “multi-headed self-attention” module [171]. This module adds several trainable weights to the “Query”, “Key”, “Value” vectors, allowing for multiple outputs per timepoint, rather than just one that is produced by the usual self-attention module.

Finally, to account for the removal of the recurrent architecture, which was well suited to handle the temporal nature of sequences, positional encodings were introduced. These are vectors that are added to a word embedding that indicates a word’s position within a sequence. However, the model must learn that the resulting embedding contains the temporality of the input sequence, as it is not built into the architecture as in recurrent networks. The combination of these approaches are generally referred to as “transformers” and have shown strong performance in NLP-based tasks such as text generation [162].

The DSANet [7] used in this chapter was built on top of the original transformer architecture. As natural language generation is similar to time-series regression, it was reasonable to hypothesise that transformers would perform well in time-series forecasting, a relatively unexplored application of this architecture. DSANet was able to outperform several other machine-learning approaches to multivariate time-series forecasting. It used input sequences of lengths ranging from 32–128 days in order to forecast windows ranging from 3–24 days in the future.

The DSANet architecture (Figure 3.4) was comprised of 3 main pathways: local, global, and auto-regressive, which are combined to produce the forecasted values. Unlike recurrent approaches, this method produces all forecasted values in parallel, rather than sequentially. The global pathway applies a fixed number of convolutional filters, whose size is equal to the length of the input sequence, such that each row in the output feature map corresponds to one of the univariate sequences that make up the full multivariate input. This output is then passed to a fixed number of self-attention modules which include position-wise feed-forward networks. The local pathway does the same as the global except that the size of the convolutional filter is smaller and max pooling is applied to the resulting feature maps to produce a resulting feature map the same size as that of the global pathway. These two feature maps are then combined using a fully connected layer that outputs a vector the size of the forecasting window, which is finally summed with the resulting vector of a classic multivariate auto-regressive model [172]. The authors justify the use of this auto-regressive pathway as a way to account for linear relationships that might be missed by the convolutional and self-attention modules due to their inherent non-linearity.

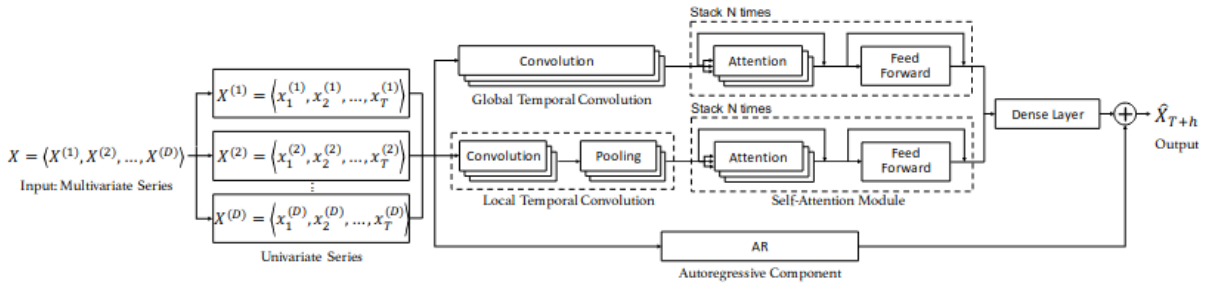


Figure 3.4 A visual representation of the DSANet architecture used for multivariate time-series forecasting [7].

In the implementation of this chapter, DSANet was trained using the same approach as GRU-R from the previous chapter. The forecasting model was trained to forecast windows of 30 minutes from the H_t dataset, given the previous 180 minutes, and used H_v as its validation set to evaluate the performance of the model after each epoch. The forecasting was therefore only trained to forecast data that did not result in occurrences of respiratory disease. This meant that if forecasted data deviated from the ground truth by more than a given amount (τ), it could be labelled as anomalous; if it wasn't anomalous, the model would have more accurately predicted it. As with Seq-U-Net, the parameters τ, α and β used for the classifier were optimised using PSO, which used O_e to find optimal values. The resulting parameters were then used to evaluate the model and classifier on O_t and a corresponding precision, recall, and MCC were calculated.

3.3. Results

3.3.1. Training Parameters

Table 3.1 outlines the hyper-parameters that were used for GRU-AE, Seq-U-Net and DSANet. The values for GRU-AE are identical to that in the previous chapter and were selected using grid search. Additionally, the hyper-parameters used for the PSO were also identical to that of the previous chapter. The hyper-parameters selected for Seq-U-Net and DSANet were determined manually.

There were two main changes to the training strategy implemented for both Seq-U-Net and DSANet in comparison to the GRU-AE. Firstly, the AMSGrad [173] variant of the Adam [63] optimiser was used in place of RMSProp. Adam combined the properties RMSProp and stochastic gradient descent with momentum and bias correction, to produce a more robust optimiser. In further research, AMSGrad took a step further by adding a "long-term memory" of past gradients, solving some of the convergence challenges faced by Adam [173].

	GRU-AE	Seq-UNet	DSANet
Hidden Size	4	256	64
No of Layers	20	4	4
Batch Size	1024	1024	1024
Dropout	0.5	0.5	0.5
Optimiser	RMSProp	AMSGrad	AMSGrad
Scheduler	Plateau	Cosine Annealing	Cosine Annealing
Learning Rate	1×10^{-6}	1×10^{-5}	1×10^{-3}
Momentum	0.3	N/A	N/A

Table 3.1 Hyper-parameters used for GRU-AE, Seq-U-Net and DSANet that are used in the training of all models.

Secondly, the GRU-AE used a learning rate scheduler that decreased the learning rate if performance plateaued after a given a number of steps, helping to avoid local maximums. For Seq-U-Net and DSANet, a cosine annealing-based scheduler was used that decreases the learning rate in line with a cosine annealing schedule that converges the learning rate to 0. As AMSGrad is an adaptive learning rate algorithm, calculating a learning rate for each training parameter every iteration, it could be argued that it did not need a learning rate scheduler as it carries out its own adjustments. However, by introducing a learning rate scheduler, the maximum learning rate the adaptive value can take is gradually lowered after each step in the training process. This meant that the learning rate could be initialised at a higher value in order to reduce loss more drastically in the early stages of training, whilst still benefiting from the adaptive learning rate set by AMSGrad.

3.3.2. Resource Efficiency

Table 3.2 shows the time it took to train each of the models along with the maximum amount of GPU memory required. These measurements were taken on a machine equipped with an Intel Xeon E5-2690 and a single Nvidia P100. Both Seq-U-Net and DSANet were substantially less resource-intensive, being both faster to train and used less memory to do so. Alongside the performance bottleneck present in recurrent neural networks described in section 3.1, this was also due to the lower number of layers required to construct an effective model (see Table 3.1).

	GRU-AE	Seq-UNet	DSANet
Time (hours)	77	7	6
Memory (GB)	3.4	1.9	2.5

Table 3.2 The time (hours) and memory (GB) required to train the GRU-AE, Seq-U-Net and DSANet models on all batches of data.

ID	Events	GRU-AE			Seq-U-Net			DSANet		
		P	R	MCC	P	R	MCC	P	R	MCC
5	0	0	1	0	0	1	0	1	1	1
9	0	0	1	0	1	1	1	1	1	1
10	0	1	1	1	1	1	1	1	1	1
14	0	1	1	1	1	1	1	1	1	1
16	0	1	1	1	1	1	1	1	1	1
17	0	1	1	1	1	1	1	1	1	1
18	0	1	1	1	1	1	1	1	1	1
19	0	1	1	1	1	1	1	1	1	1
24	0	1	1	1	1	1	1	1	1	1
26	4	0.250	1	0.500	1	1	1	1	0	0
27	5	0.111	1	0.333	0.167	1	0.408	1	0	0
28	3	1	1	1	1	1	1	1	0	0
29	1	0	1	0	0	1	0	1	0	0
30	0	1	1	1	0	1	0	1	1	1
31	1	0.050	1	0.224	0.029	1	0.169	1	1	1
32	0	1	1	1	0	1	0	1	1	1
Avg 0 Events		0.818	1.000	0.818	0.727	1.000	0.727	1.000	1.000	1.000
Avg >0 Events		0.282	1.000	0.411	0.439	1.000	0.515	1.000	0.200	0.200
Avg All		0.651	1.000	0.691	0.637	1.000	0.661	1.000	0.750	0.750
I/B/C				3/3/10			4/2/10			4/0/12

Table 3.3 Table of results for the PSO-optimised threshold-based anomaly detection of Mahalanobis distance produced by a GRU-based autoencoder, PSO-optimised threshold-based anomaly detection of Mahalanobis distance produced by a Sequential U-Net, and particle swarm optimisation (PSO)-optimised threshold-based anomaly detection of loss incurred from a DSANet. The events column lists how many times the respiratory disease prevalence increased from 0 within the batch, P denotes the precision, and R is the recall of the model. These results are from the test data O_t. The final I/B/C row indicates the number of folds that were Incorrect ($MCC = 0.0$), Between ($0.0 < MCC < 1.0$), or Correct ($MCC = 1.0$).

3.3.3. Test Results

Table 3.3 shows the performance of the GRU-AE, Seq-U-Net and DSANet for each of the individual batches. The final four rows in the table show different summarisations for each model used. The first summary row shows the average precision, recall and MCC for batches where no respiratory disease prevalence occurred in the ground-truth data; the second shows the inverse, batches where respiratory disease prevalence was greater than 0. The penultimate row shows the average precision, recall and MCC across all batches. To better understand the overall MCC, the final row shows a high-level summary, for which the values I, B and C were created. I was the number of batches where the overall MCC was 0, B was the number of batches where overall MCC was greater than 0 but less than 1 and C was the number of batches where overall MCC was 1. Each of these summary metrics provides a different insight into model performance, so must be assessed together in order to properly evaluate the performance of each of the models. In addition to the performance metrics, the values for α and β, that were selected by the PSO are shown in Table 3.4.

Firstly, regarding batches where there were no cases of respiratory disease, both the GRU-AE and DSANet models performed very well across all metrics, with Seq-U-Net trailing behind. In terms of the precision and recall for these batches, the GRU-AE and DSANet were both well balanced, whereas Seq-U-Net's main underperformance was in terms of precision. This indicates that Seq-U-Net raised more false positives for these batches, misclassifying normal windows as anomalous.

However, for the batches where there were cases of respiratory disease, DSANet drastically underperformed, missing all but one of the anomalous time windows. This is likely a result of the PSO either maximising or minimising the value for τ so as to minimise the number of alerts raised, rather than to find the correct balance. The GRU-AE from the previous chapter achieved perfect recall for these batches. This was at the expense of precision, indicating that although there were no false negatives, a substantial number of its alerts were false positives. Seq-U-Net's poor performance on "normal" batches appears to have allowed it to achieve the highest average MCC, despite its poor precision, for batches where there was respiratory disease. This was a notable improvement over the GRU-AE.

Despite the apparent performance improvement, the Wilcoxon test, corrected for multiple observations, showed no statistically significant difference between any of the models overall MCC. Although DSANet scored an MCC of 1.0 for many batches, it did not detect the anomalous periods in 80% of the batches where they occurred. The GRU-AE outperformed Seq-U-Net when respiratory disease prevalence was 0. However, Seq-U-Net struck a better overall balance, accepting poorer performance when respiratory disease prevalence was 0, for increased performance for batches with respiratory disease.

ID	Events	GRU-AE		Seq-UNet		DSANet	
		α	β	α	β	α	β
5	0	1	1	1	6	1	6
9	0	1	2	1	6	1	6
10	0	1	2	1	6	1	6
14	0	1	4	1	6	1	6
16	0	3	1	1	6	1	6
17	0	1	5	1	6	1	6
18	0	3	5	1	6	1	6
19	0	1	5	1	6	1	6
24	0	1	6	1	6	1	6
26	4	3	6	1	6	1	6
27	5	1	6	1	6	10	6
28	3	8	6	1	6	1	6
29	1	2	6	1	6	1	6
30	0	4	6	1	6	4	6
31	1	1	6	4	6	1	6
32	0	8	5	1	6	1	6

Table 3.4 Table of results for the PSO-optimised threshold-based anomaly detection of the Mahalanobis distance produced by a GRU-based autoencoder, Seq-UNet and DSANet. α and β denote the start and length of the time window (in days), respectively, for which an alert is assessed, and τ is the threshold the loss/Mahalanobis distance needed to exceed in order to raise an alert. They are optimised using training data O_e; these results are from the test data O_t.

The PSO-optimised values for α and β were consistent for both Seq-U-Net and DSANet, typically raising an alert for oncoming respiratory disease 1 to 7 days prior to an actual increase. The GRU-AE achieved numerous higher values for α and typically lower values for β, indicating that it was able to forecast occurrences of respiratory disease prevalence further into the future and for a more specific time window.

3.3.4. Case Study

As in the previous chapter, figures 3.5, 3.6, and 3.7 show a summary visualization of each of the models performance on batch 27 from Table 3.3. This batch was chosen as it shows a range of characteristics that can be found across all batches, such as quick successive periods of disease, both long and short periods time where there was disease, long and short periods time where there was no disease and both high and low levels of disease.

The visualisations for each model are broken down into 3 graphs. The topmost graph shows the ground truth respiratory disease prevalence each day, which has been translated into a single bar which represents where respiratory disease prevalence is 0 (green) and respiratory disease prevalence is greater than 0 (red). The second graph shows the Mahalanobis distance between

the output of each model for a given time window of O_t and the baseline output of the model produced by H_v along with the threshold, τ, that was selected by PSO. There are also markings to show alerts, which are defined as timepoints where the Mahalanobis distance goes from below τ to above τ. The final graph shows these alerts and their corresponding alert window using the models respective α and β values, and how they relate to periods of respiratory disease. These are coloured to distinguish true positives, false positives and false negatives.

The Mahalanobis distance of the GRU-AE and Seq-U-Net were very similar. Seq-U-Net had fewer false positives than the GRU-AE, particularly after the elongated period of increased respiratory disease prevalence. It does not detect the first two anomalous periods early enough, though does raise alerts after the clinical symptoms of respiratory disease start showing. This exemplifies the results presented in Section 3.3.3, showing Seq-U-Net raises fewer false positives than the GRU-AE at the cost of increased false negatives.

The output produced by DSANet helps to better understand the poor performance it achieved as threshold τ is set very low, meaning an alert is never raised for this batch. This is symptomatic of there not being enough of an increase in the Mahalanobis distance when respiratory disease prevalence was greater than 0. Something even the GRU-R approach, presented in the previous chapter, did not suffer from.

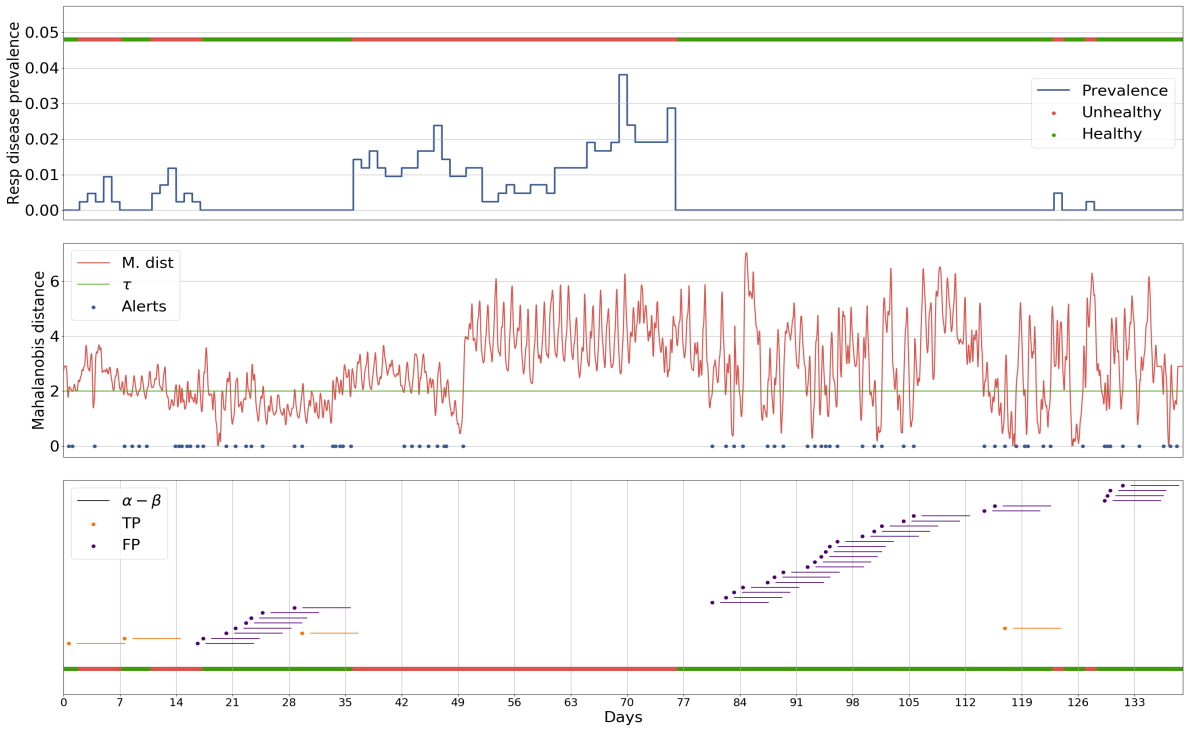


Figure 3.5 Results for batch 27 processed by the GRU-autoencoder-powered anomaly detection, given its PSO-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

Figure 3.6 Results for batch 27 processed by the Seq-U-Net-powered anomaly detection, given its grid search-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

Figure 3.7 Results for batch 27 processed by the DSANet-powered anomaly detection, given its PSO-optimised parameters α (the start of the window for which an alert is for), β (the length of the window), and τ (the threshold which the loss between predicted values and actual values must cross to be considered an alert), in relation to the prevalence of respiratory disease within a single batch.

3.4. Discussion

In the previous chapter, we presented an approach to multivariate time-series anomaly detection that could be used to detect conditions that were likely to result in occurrences of respiratory disease. Although the proposed GRU-AE method performed well, it required a substantial amount of resources, both in terms of training time and memory. The methods applied for comparison to this method were able to achieve similar, albeit slightly poorer, performance metrics using a fraction of the time and memory necessary for the GRU-AE. Therefore, the objective of this chapter was to evaluate how alternative deep learning-based models could be used in place of the GRU-AE, that would require fewer resources, whilst minimising the impact on performance metrics.

Both of the architectures evaluated as alternatives to the GRU-AE required less time and memory resources in order to be able to detect environmental scenarios that would likely lead to occurrences of respiratory disease. However, only Seq-U-Net, the CNN-based architecture, was able to do so whilst also limiting the impact on performance relative to the GRU-AE. The poor performance of the DSANet was determined to likely be due to the inability of the PSO to find a balance between the three parameters it was optimising. A potential solution to this would be to use multi-objective optimisation, rather than compressing the different objectives into a single fitness function [174]. Therefore, future work may consider this as an alternative approach to optimising the parameters of the anomaly detector.

Seq-U-Net achieved a higher MCC than the GRU-AE for batches where there were cases of respiratory disease, but a lower overall MCC. This appears to have been caused by allowing more false positives, as is indicated by the lower overall precision, whilst still achieving high overall recall. Though this trade-off is not ideal, the substantial reduction in necessary resources creates a case for choosing Seq-U-Net over the GRU-AE.

The general approach of this work is not without flaws. The main limitation is that there are factors other than the environmental conditions within a housing unit that contribute to the prevalence of respiratory disease [155]. This means it is not feasible to expect a model only using these variables as inputs to detect all upcoming occurrences of respiratory disease. Though this should be taken into account when assessing the performance of the models, both in this and the previous chapter; future research should seek to include additional variables that represent known risk factors identified in the literature such as heard size and density [175]. This data, alongside the environmental sensor data, would allow the model to better contextualise the sensor data.

The core contribution of this chapter was in the evaluation of how a convolutional-based model can match the performance of a recurrent-based model at modelling time-series data, whilst taking a fraction of the time and memory to train despite not being explicitly aware of temporal

nature of the data. Resource-intensive deep learning solutions have become commonplace [162], which can make building new approaches on top of existing solutions inaccessible. As this implementation reduces resource requirements, it simplifies the process of iterating and tuning models, whilst also improving the accessibility to deep learning-based anomaly detection. Furthermore, as single-board compute devices become more developed [176, 177], smaller models that can easily fit onto these devices will become more attractive, as they can more easily be deployed into remote locations where poor infrastructure limits access to cloud-based services. As data collection must be done on farms that are typically in these types of locations, this is an important factor to consider when developing new technologies in this field. Due to the reduced requirements of the work presented in this chapter, the feasibility of deploying the workflow we have presented into production into a rural setting is both realistic and affordable, even for smaller farms.

This proposed architecture for anomaly detection is not limited to only applications in pig farming. The general workflow we have presented could not only be adapted to function in a similar context for different species, but also in entirely different fields that require multivariate time-series anomaly detection. The performance of our architecture would certainly vary between use cases, but, at its core, the approach is entirely agnostic to the application. Further research might consider implementing this general workflow as a solution in a broader range of challenges and evaluating its performance.

3.5. Conclusion

This chapter evaluates two alternatives to the recurrent-based architecture presented in the previous chapter for the purpose of creating an early warning system for occurrences of respiratory disease. We evaluated one CNN-based and one transformer inspired architecture and demonstrated how a CNN-based approach, augmented with causal convolutions, can drastically reduce the resource requirements for training a model on multivariate sequential data compared to a recurrent-based approach. Although the end results were impacted, the benefits of faster training and lower memory requirements make it an attractive option for "edge"-based machine learning. This maximises the feasibility of deployment in operating, commercial farms, whilst also delivering an effective anomaly detection solution.

Chapter 4. Automated Individual Pig Localisation, Tracking And Behaviour Metric Extraction Using Deep Learning

Abstract

Individual pig tracking is key to stepping away from group-level treatment and towards individual pig care. By doing so we can monitor individual pig behaviour changes over time and use these as indicators of health and well-being, which, in turn, will assist in the early detection of disease allowing for earlier and more effective intervention. However, it is a much more computationally challenging than performing this task at the group level, particularly as mistakes in identification and tracking accumulate over time. We combine a CNN-based object detection method, Faster Region-based convolutional neural network (Faster R-CNN), with two potential real-time multi-object tracking methods in order to create a complete system that can autonomously localise and track individual pigs allowing for the extraction of metrics pertaining to individual pig behaviours from RGB cameras. We evaluate two different transfer learning strategies to adapt Faster R-CNN to our custom-built pig detection dataset that is more challenging than conventional tracking benchmark datasets. We are able to localise pigs in individual frames with 0.901 mean average precision (mAP), which then allows us to track individual pigs across video footage with 92% Multi-Object Tracking Accuracy (MOTA) and 73.4% Identity F1-Score (IDF1), and re-identify them after occlusions and dropped frames with 0.862 mAP (0.788 Rank 1 cumulative matching characteristic (CMC)). From these tracks we extract individual behavioural metrics for total distance travelled, time spent idle, and average speed with less than 0.015 mean squared error (MSE) for each. Changes in all these behavioural metrics have value in the detection of pig health and wellbeing.

4.1. Introduction

The ability to monitor the behaviour of animals, in particular, how said behaviour changes over time and under varying circumstances, provides us with the knowledge that can assist in identifying problems before they become serious, or even life-threatening, and enhances the success of intervention[157, 178]. Continuous human observation of animals is impractical due to the effort required and the enormity of the scale of modern pig livestock units. As a consequence, farm staff usually resort to brief observations that are only able to detect substantial

changes or clinical signs. Thus they fail to detect subtle changes in behaviour that usually precede clinical signs of disease; this results in late intervention [157, 35].

Recurrent research developed a method that enables the measuring of behavioural traits in groups of pigs [35]. Despite the usefulness of having group-level pig behaviour measures, there are several advantages in being able to automatically detect individual behaviours and identify pigs that may be at risk or are challenged [178] as this enables more personalised treatment plans. Targeted and selective animal treatment may result in furthering the trend towards reducing antimicrobial input in livestock systems [179]. However, individual-level methods create data that is sparser and hence more sensitive to potential estimation errors [180]. Moreover, errors in pig identification and tracking can propagate to later frames, which is an issue that does not affect group-level measurements. In addition to the clinical advantages, treating pigs as individuals also helps to assuage consumer concerns regarding animal welfare.

In this chapter, we present an approach to detect and track individual pigs kept in groups, using inexpensive, colour (RGB) cameras in commercial farm conditions. The approach does not use individual pig identifiers, such as RFID tags, due to their impracticality and industry concerns over their use, mainly associated with their retrieval at the abattoir, potential residues and cost. Additionally, this approach does not change the day-to-day operations of farmers by requiring them to make and record regular observations or install and remove identity tags and markers.

We make use of a Faster Regions with CNN features (Faster R-CNN) [81] architecture adapted to our domain using transfer learning in order to detect the location of pigs in each frame of an RGB video. These detections are then used to generate identities that are then processed by multi-object tracking (MOT) algorithms. In this chapter we present two approaches to tracking the identities of pigs between frames, one which uses trajectory-based prediction and association [181], and one which uses both trajectory and similarity in visual appearance [182]. We then calculate behaviour measures from the tracks to illustrate how we can extract valuable and actionable knowledge from these algorithms in our concrete domain of application.

The main contribution of our method is that it provides a full workflow to localise, track, and extract behavioural metrics of individual pigs using only an RGB camera, in real-time, without the need for additional hardware (such as ID tags) or visual aids (such as IDs marked directly on pigs), whilst also still being feasible to install in a commercial farm. We also demonstrate that, despite the very similar visual appearance of the pigs in our dataset, deep learning methodologies can be used to generate feature vectors capable of discriminating between identities of pigs. We achieve this using the entire bounding box produced by the Faster R-CNN, rather than defining a smaller area to identify pigs as recent research has done [101]. This is key to understanding the potential applications of deep learning in precision livestock farming.

The remainder of this chapter is structured as follows. Section 4.2.1 describes the datasets that were used for training and evaluation in our models. Section 4.2.2 describes the detection method used followed by Section 4.2.3 which outlines the two methods used for multi-object tracking, and how the deep association metric model was trained. Section 4.2.4 describes how we use these tracks to extract behaviour metrics for individual pigs and is followed by Section 4.2.5 which describes how each of the components of our method is evaluated. Section 4.3 and Section 4.4 outline and discuss the results of each of the components respectively. We finalise with the conclusion in Section 4.5.

4.2. Materials & Methods

In this section, we describe the data used to train and evaluate our methods for detection, tracking and extracting behavioural metrics, the model used to detect the location of pigs in an image, the tracking methods that were used to assign the detections to identities, and how we analysed the tracks to extract behavioural metrics. We also describe the means by which we evaluate these methods.

4.2.1. Dataset Descriptions

Three standard datasets were used in order to implement all of the components of our work: ImageNet [183], an image classification dataset commonly used to pre-train a CNN that processes images; Pascal Visual Object Classes Challenge 2007 [184], commonly used as a benchmark dataset for object detection and classification; and Motion Analysis and Re-identification Set (MARS) [185], a video-based person re-identification (Re-ID) dataset. Additionally, we used three of our own, manually-annotated, pig datasets: one for detection (Section 4.2.1), one for tracking (Section 4.2.1), and one for re-identification (Section 4.2.1).

We selected distance travelled, time spent idle, and average speed for behavioural metrics to extract from individual pig tracks as they best inform us about a pigs activity levels, an valuable metric for understanding pig health and wellbeing.

The images used in these pig datasets were collected at Newcastle University's Cockle Park farm. Because the images were collected from pigs husbanded under farm conditions, there was no need for ethical approval. The building used houses 4 pens, each containing 20 pigs of the same age and of balanced weight. The pigs were recorded using the RGB sensor within the Microsoft Kinect v2 (resolution: 1920×1080, field of view: 84.1×53.8, focal length: 3.29, shutter speed: 14ms, max frame rate: 30 FPS) mounted on the ceiling of the room, pointing downwards. Although we used this specific sensor, for the purpose of the methods presented in this work, any RGB camera with similar parameters would be suitable as we do not use any of the specialised features of the Kinect. The camera was tilted at an angle (rather than perpendicular

to the floor) so that the camera covered half of one of the pens. There is some overlap in the images with the adjacent pen (Figure 4.1), separated by a red wall, however, we removed any detections in this area. Due to technical constraints of the data capturing infrastructure, we were not able to record long, continuous segments of video so were limited to a theoretical maximum of 10 minutes, which, although short, is longer than a human observer would realistically spend watching the animals.

Pig Detection Dataset

Figure 4.1 A example image from the pig dataset where pigs are densely packed into one area with corresponding ground-truth annotations.

Our manually-annotated (Figure 4.1) pig dataset consisted of 1,646 images (50% used for training and 50% used for testing) consisting of 9,014 annotations. All data used in this dataset came from the same camera and the same batch of pigs, though was collected across different days and times. The days used for creating the test set are not included in the training set in order to create a level of separation. This dataset contained roughly 6 times fewer images and nearly 3 times fewer annotations than Pascal Visual Object Classes Challenge 2007 (VOC), hence indicating that our dataset has a higher annotation density (number of objects per image). The pig detection dataset suffered from a number of complexities that are not found in the VOC dataset. The challenges largely stem from the fact the data was collected on a commercial farm environment. For example, the reason there were more annotations per image in this dataset was because the objects were often very densely packed into a small space within the image 4.1. This made the separation of the pigs a very difficult task, even when carrying out the manual annotations.

Secondly, the quality of the images in the pig dataset was substantially lower than that of the images in VOC. This was because the images were captured on a live farm, which is extremely dusty. Additionally, images can sometimes be overexposed due to natural light entering through windows. Having this natural light is a requirement in order to comply with UK legislation. When contrasted with the very high-quality level of photographs used in VOC there was a very significant difference. Because the features of an object were much more difficult to identify due to the lower image quality, models trained on this data are at a disadvantage as these are typically what is used in order to create a generalisable model.

These factors combined show that the pig dataset was a substantially more difficult dataset than VOC, which is something that previous research has had to overcome by employing post-processing methods [38]. In order to quantify how our implementation performs in these conditions, in addition to testing on the whole test set, we also assessed the detection performance independently for images containing: many pigs, densely packed pigs, overexposed images, and low-light images (Figure 4.2).

Images were classed as containing many pigs if more than 10 pigs were in the image, as this meant more than half the pigs were in less than half the pen space (4% of the test set). As for the densely packed pigs segment, images were placed here when more than 4 bounding boxes were overlapping (43% of the test set). Images were determined to be overexposed by manual annotation (11% of the test set). Finally, the low-light segment was made up of images where the average brightness of a pig was lower than 100 (4% of the test set). Pig brightness was calculated by converting a bounding box containing a pig to greyscale and taking the average pixel intensity.

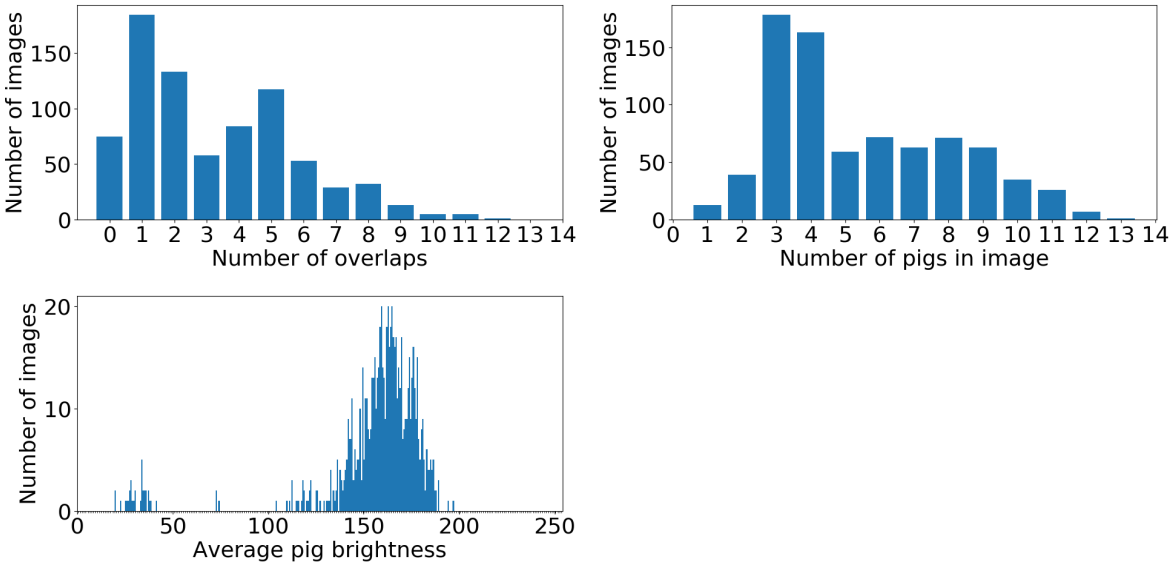


Figure 4.2 Distributions of the number of overlapping bounding boxes per image (top left), the number of pigs per image (top right) and the average brightness of a pig per image (bottom left) within the test set.

As can be seen in Figure 4.1, the camera's field of view overlapped with the adjacent pig pen. We therefore ignored any detections from this area of the camera. This was achieved by setting a threshold on the Y-axis, along the red wall, and ignoring any detections to the left of it.

Pig Tracking Dataset

Images from the detection dataset were used to create another dataset specifically for tracking. This consisted of a single 7.8 minute video recorded at an average of 4 frames per second (FPS) ($\sim$ 1,874 frames) from a single camera covering half a pig pen (the maximum was 10 FPS. Each frame was annotated, in the same way as the detection dataset, however IDs were given to each pig that persisted from frame to frame. Due to the nature of the recording environment, in particular the hardware used, the recording varied in its FPS and regularly dropped frames throughout the recording. This resulted in some drastic changes, frame-to-frame, in some situations (e.g. a pig appear in the middle of the pen, or extremely quickly moving pigs).

Moreover, as pigs leave and later re-enter the scene, we cannot continuously track them for the whole length of the video. This has resulted in a set of 25 manually curated, unique tracking IDs. Some tracks last for most of the video while some others are fairly short. The duration of each track is represented in Figure 4.3

Pig Re-Identification Dataset

In addition to this single-camera tracking dataset, we gathered a separate, dual-camera pig Re-ID dataset (Figure 4.4), structured similarly to MARS [185], a dataset for person Re-ID. This pig Re-ID dataset consisted of 25 pig identities, where each identity had an average of 280 images, totalling 5,653 images (60% for training, 40% for testing). All annotations were resized to be 128 x 256 for processing by a CNN (outlined in Section 4.2.3). This is a difficult Re-ID dataset as, when compared to that of a person Re-ID dataset, such as MARS, where clothes strongly distinguish two people apart, pigs look very similar to one another (Figure 4.4).

4.2.2. Pig Detection Method

Regions with CNN features (R-CNN) was originally proposed in order to use a high-capacity CNN for region proposals for the purpose of object detection [186]. This was very successful but suffered from very slow training times as there were several models that contributed to the overall system, meaning they needed to be trained individually. An improvement to this method was proposed called Fast Regions with CNN features (Fast R-CNN), which made use of Region of Interest (ROI) pooling to speed up the assessment of region proposals [187]. This offered another substantial performance boost, but the initial region proposals were still created by a separate

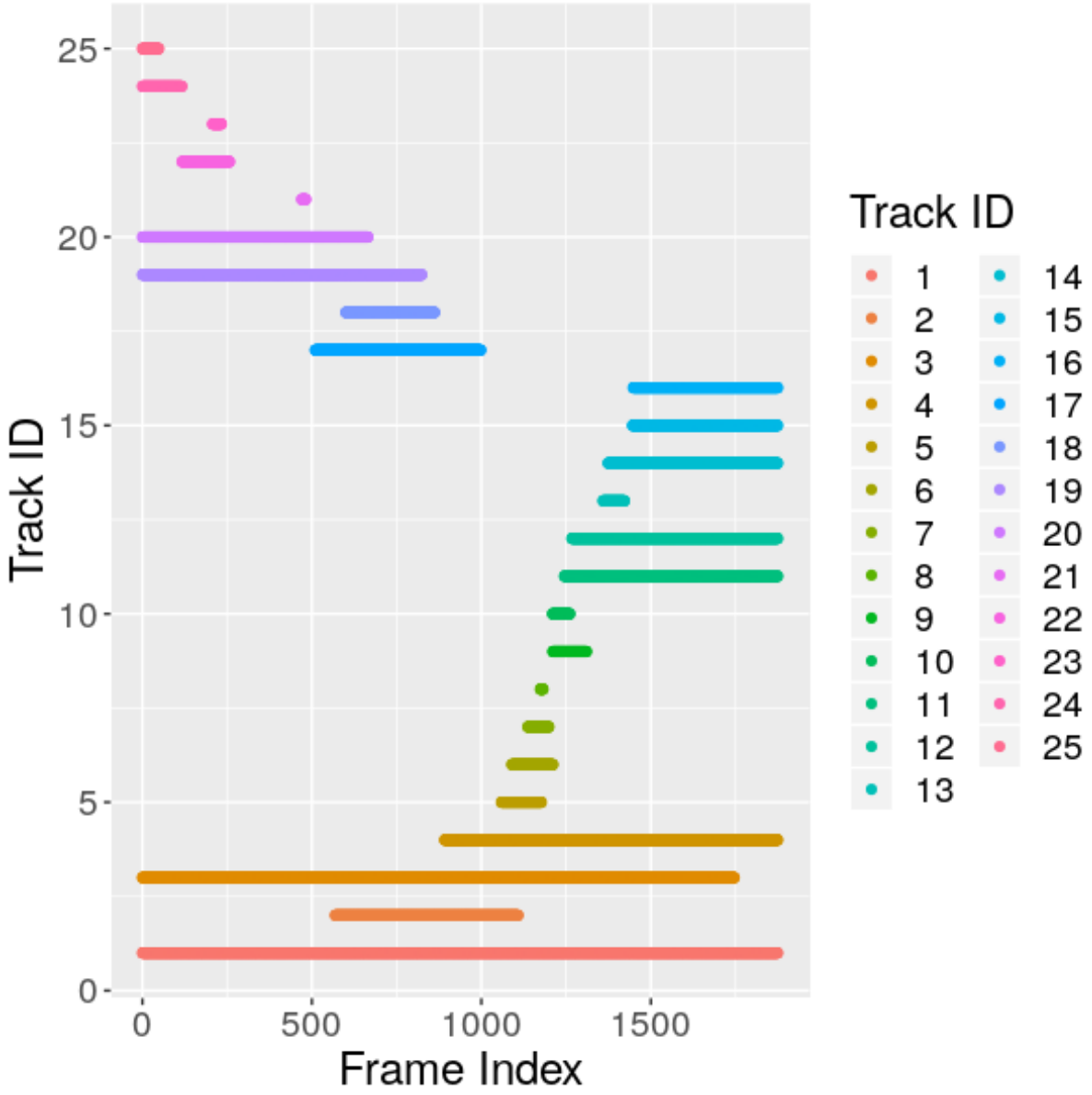


Figure 4.3 Representation of the manually annotated pig tracks. The Y-axis shows the ground truth pig ID, the X-axis shows the frames during which the pig was visible. Once a pig left the camera, it was not re-identified and was therefore given a new ID.

model. Faster R-CNN [81] solved this problem by introducing a region proposal network to the system which allowed for the whole system (extracting features from images and proposing and classifying regions) to be trained as one single model (Figure 4.5). This reduced the training time whilst still achieving strong performance.

The feature extraction layers of the Faster R-CNN can be pre-trained using large datasets such as ImageNet. This has shown to be a very powerful way of initially training CNNs [188] and is the standard approach for initialising these layers [70, 189, 190]. The remainder of the network requires an annotated dataset of bounding boxes around objects in images along with their class, such as VOC. When doing further transfer learning with these networks (e.g. training the network to identify a different set of classes), the fully connected (FC) layers are often where the changes are made. This has been shown to perform well in applications such as

Figure 4.4 Top: A sample of two identities of the MARS dataset for person re-identification. **Bottom**: A sample of two identities of the pig re-identification dataset.

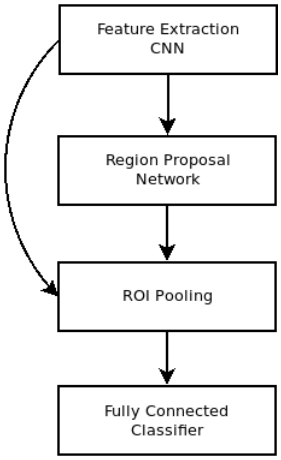


Figure 4.5 *An overview of the Faster R-CNN structure*

self-driving cars [191] where the classes in the target dataset differ from the source dataset, and detecting guns in images and video [192]. Recent research has also shown that when the target domain is dissimilar to the source domain, the fully-connected layers are vital to successful visual representation transfer [193], meaning their weights need to be preserved as part of the transfer learning.

The Faster R-CNN model, used in our application to detect pig locations in images, is comprised of 3 main components: the feature extractor, responsible for creating a fixed-length feature vector from an input image; the Region Proposal Network (RPN), which selects regions that are likely to contain an object based on the feature vector; and the FC layers, which classifies each of the regions into one of the classes. In our implementation, we used ResNet-101 [70], a popular image classification architecture, where 101 denotes the number of layers used. This architecture was chosen due to its improved performance over VGG-16 [64], another commonly used network in similar applications, when applied to multiple object detection datasets sets

such as VOC ($\sim$ 75 mean average precision (mAP) vs $\sim$ 70 mAP depending upon configuration) and Common Objects in Context (COCO) [194]($\sim$ 36 mAP vs $\sim$ 30 mAP depending upon configuration). Other variations of the ResNet architecture that use a different number of layers are also used, where fewer layers improve speed, while more layers typically improve performance. We deemed the 101 variant to be the best balance between speed and performance for this application.

Once the feature vector is created, it is passed to the RPN along with anchor boxes. These are pre-defined boxes that are used to start the process of determining where objects are within an image. The RPN proposes the regions of the image that are likely to be objects as opposed to background. These proposed regions are then passed through an ROI warping layer [195], an improved variation of ROI pooling, along with the feature vector from the feature extractor, which allows for the proposed regions to be represented as equal sizes allowing for significantly faster processing. The output from this layer is another fixed-length vector which is finally passed to the fully-connected layers which carry out the classification of each region.

For the training of our models, we used the parameters that perform best on the VOC dataset (Table 4.1) using stochastic gradient descent. An Nvidia Tesla P100 graphics card was used for both training and inference.

Hyper-Parameter	Value
Learning rate (LR)	0.001
LR Decay step	5
LR Decay multiplier	0.1
Batch Size	8

Table 4.1 The parameters used for the Faster R-CNN that perform best on the VOC dataset.

We made use of all three datasets in order to build this model. The feature extraction layers of the model were pre-trained on ImageNet. This is common when training any CNN on images as the dataset is extremely large ($\sim$ 14 million images) and therefore takes a substantial amount of time to train from randomly initialised weights (a week in some cases, dependent upon hardware). The RPN and FC layers of the Faster R-CNN were trained using two datasets: firstly VOC, followed by the pig detection dataset. It was expected that the densely-packed nature of the pigs at certain time points in the dataset would be particularly a problem for the Faster R-CNN as it uses non-maximum suppression (NMS) to filter overlapping bounding boxes which is typically enacted when Intersection over Union (IoU) > 0.7 [81] (described in Section 4.2.5).

Before we could train the model on the pig dataset after training on VOC, it was necessary to modify the model architecture to account for the change in the number of potential classes. This change in model architecture is referred to as transfer learning, where a model is trained to solve one task, in order to help it solve a different, but related problem. There are multiple ways this can be implemented, two common approaches would be to train the RPN and FC layers on VOC, then modify the FC layers to account for the different classes in the pig dataset (20 in VOC, 1 in

the pig dataset). Alternatively, an additional layer can be added to the final FC layer after training on VOC which has only one class outcome. This results in two output nodes (1 for pig and 1 for background), which creates an additional 78 trainable parameters in our fully-connected layers. Despite this slight increase in the number of parameters, we used the transfer learning strategy that adds an extra layer as it performed better than the other evaluated strategies (Section 4.3.1).

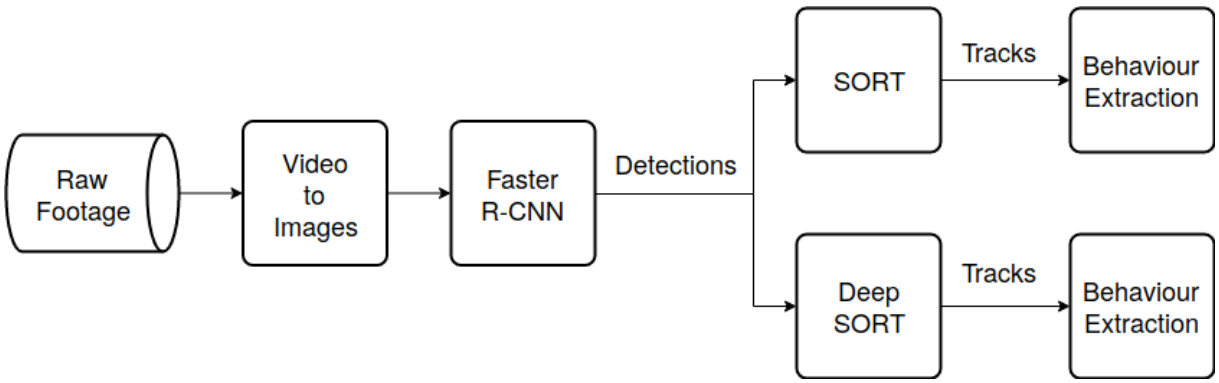


Figure 4.6 A breakdown of the full workflow of our implementation from the video footage of a pig pen, to the behavioural metrics we extract from the tracking methods.

4.2.3. Pig Tracking Methods

Once the Faster R-CNN detected the pigs' locations in each frame, it was necessary to track the identity of each pig between frames. To achieve this, we evaluated two alternative strategies for this task.

Distance-based Tracking (SORT)

For our baseline tracker, we used Simple Online and Real-time Tracking (SORT) [181]. This method combines the Kalman filter [196] with the Hungarian algorithm [197] to create a multi-object tracking algorithm that uses object trajectory to associate new detections with a tracklet. While the Kalman filter assumes that the process it is predicting (e.g. the tracklet of a pig) is not random, and the movements of pigs are arguably arbitrary, it may seem that the Kalman filter is not an ideal choice for a tracking mechanism in this application. However, as the distance an individual pig moves between frames is relatively small, it was not expected that this would cause problems for the tracking of the pigs despite not meeting the main assumption of the Kalman filter. The main benefit of using the Kalman filter over other methods, such as a particle filter [198, 199], is that the Kalman filter is much simpler, making it less computationally expensive.

This method is highly dependent upon accurate object detection, as only the location of objects is used, and is an entirely unsupervised method. As no training data is required, it can be applied directly to the detections found for the tracking test data set outlined in Section 4.2.1.

The Kalman filter uses the previous locations of an object to predict the next most likely location; this is done for all objects independently. Once detections are produced for the next frame, their centre points are calculated and they are assigned to the nearest location that the Kalman filter predicted by means of the Hungarian algorithm. One of the main challenges for this approach is when two objects are near each other as only a small deviation between the Kalman filter and ground truth can result in an identity switch (IDSW). Section 4.2.5 provides a more detailed description of the values and metrics used to evaluate this method.

Distance & Visual-based Tracking (Deep SORT)

We also made use of Deep SORT [182], which, alongside the Kalman filter & Hungarian algorithm used in SORT, uses a learned association metric to determine if two images of a pig contain the same pig or not. In this method, when identities of objects are being matched between frames, both the trajectory prediction and the association metric are used to determine if two objects in different frames are the same. Because this method is able to identify if a "newly" identified pig actually belongs to an identity that has already been established, it is much more capable of handling long-term occlusions and corruptions in the dataset (e.g. dropped frames), which is not uncommon in the datasets we used. This method is configured to only use the previous 480 images (2 minutes at 4 FPS) so that it can consistently track a pig as it grows. If all previous known images of the pig were stored it could degrade performance as pigs change in appearance over time.

Layer No.	Name	Size/Stride	Output
1	Conv	$3 \times 3/1$	$32 \times 128 \times 64$
2	Conv	$3 \times 3/1$	$32 \times 128 \times 64$
3	Max Pool	$3 \times 3/2$	$32 \times 64 \times 32$
4	Residual	$3 \times 3/1$	$32 \times 64 \times 32$
5	Residual	$3 \times 3/1$	$32 \times 64 \times 32$
6	Residual	$3 \times 3/2$	$64 \times 32 \times 16$
7	Residual	$3 \times 3/1$	$64 \times 32 \times 16$
8	Residual	$3 \times 3/2$	$128 \times 16 \times 8$
9	Residual	$3 \times 3/1$	$128 \times 16 \times 8$
10	FC	-	128
11	l_2Norm	-	128

Table 4.2 An overview of the CNN architecture used to produce the association metric for pig re-identification. This is trained using the MARS dataset followed by fine-tuning on our own pig re-identification dataset. The cosine softmax classification layer is not shown in this table as it is removed for inference.

This association metric is learned using a small CNN-based model (Table 4.2), which uses a re-parametrisation of the softmax classifier that includes a measure of cosine similarity in the representation space, which is a 1×128 vector, initially developed for person Re-ID [200]. Where an input is a cropped image of a detected object, the network is trained to minimise the

cross-entropy loss of the class predictions generated by the model and the true label distribution. This means that the final feature representation (the output of the l_2 layer) implicitly learns to maximise inter-class separation. Once the model is trained, the cosine softmax classification layer is discarded and feature vectors can then be compared with the feature vectors that are stored for each existing identity so that is can be assigned to the identity with the closest match. However, cosine similarity is not the only factor used. In order for the identity assignment to be made, it must also appear within range of the predicted location produced by the Kalman filter for that identity, thereby taking both visual appearance and trajectory into account. Training is carried out by feeding forward a single "query" image, which is compared to a range of "gallery" images. Where there are multiple cameras in the dataset, the gallery images are taken from different cameras. The performance of this model discussed in Section 4.2.5.

The original implementation of the CNN that generates the association metric was created for the purpose of person Re-ID [200]. This research area focuses on being able to identify if two images or videos of a person are the same person. The key applications of this research focus on facial recognition and datasets where the objective is to track individual people within a crowd, allowing for individuals to become temporarily occluded and then recovering their original tracking identity.

As we used this method for tracking pigs, which look very similar to one another, rather than humans, our application is considered much more challenging than human Re-ID. Our implementation was pre-trained on MARS to learn the basic concepts of Re-ID, followed by training using our custom pig Re-ID dataset (Section 4.2.1) to optimise the method to be able to identify identical pigs in a commercial setting. The CNN was trained using the Adam optimiser [63] with a low learning rate of 0.00001 and a batch size of 128.

4.2.4. Behavioural Metrics Extraction

Once tracks were established for individual pigs, we were able to derive behavioural metrics pertaining to each individual track. We measured average speed, total distance covered and time spent idle as these values quantify how active pigs are. Change in activity has been linked to several pig health and welfare challenges, with the general trend being that such challenges tend to decrease the levels of activity in individuals [157, 35]. We track activity levels by measuring idle time, which is the reciprocal of active time.

In order to calculatethe distance travelled and idle time we used the Euclidean distance between the centre-point of each detection box from frame to frame. We did not convert these measurements into real-world distances as it is simpler to calculate and because we have variable FPS, which we did not store when recording the footage. This, for example, makes it not possible to accurately estimate speed. This causes some discrepancies, albeit small, as the camera we used was not perfectly perpendicular to the floor; it was set at an angle in order to have full

coverage of the pen. The amount of time a pig spent idle was defined by the amount of time where the pig moves no more than 4 pixels between frames.

4.2.5. Evaluation

Detection Evaluation

Intersection over Union (Equation 4.1) was used to assess how accurate a predicted bounding box was in comparison with the ground truth (localisation performance), which was calculated using Equation 4.1. The higher the IoU, the more accurate the bounding box is (Figure 4.7). Rather than using the threshold of 0.5 to determine whether the IoU of a predicted bounding box is accurate, which is the standard proscribed in the Pascal Visual Object Classes Challenge 2007 challenge, we required an IoU ≥ 0.6. This was due to the fact that there are often many pigs in close proximity to one another, so there was a need to ensure that bounding boxes produced by the detector were as close to the ground truth as possible.

$$\text{IoU} = \frac{\text{Area of overlap between bounding boxes}}{\text{Are of union between bounding boxes}} \tag{4.1}$$

In order to summarise the performance of the detector, we used mAP a standard metric that is used to evaluate the performance of object detection methods [81, 186, 187, 201].

Figure 4.7 Examples of how IoU is calculated. **Left:** Poor performance IoU = 0.4034. **Middle:** Good performance, IoU = 0.7330. **Right:** Excellent performance, IoU = 0.9264.

Deep Association Metric Evaluation

In order to evaluate the performance of the association metric learned by the CNN outlined in Section 4.2.3, we made use of the overall mAP of the classifier and the Cumulative Matching

Characteristic (CMC). It is common to evaluate CMC ranks 1 through to 20 [202], however, as we only have 25 identities in our pig Re-ID dataset, it was not possible to do this. As with many $1:m$ identification systems, for our application, the CMC at rank 1 is the most crucial [203, 204]. We therefore focus predominantly on this metric, though we additionally report CMC ranks 3 and 5, in order to better show how our model is generally performing.

Tracking Evaluation

We used the widely accepted metrics outlined in the 2016 MOT Challenge [205] in order to evaluate the performance of our tracker, implemented using the py-motmetrics library [206]. The tracking performance measure we used was multi-object tracker accuracy (MOTA) (Equation 4.2), the most commonly used metric to benchmark MOT solutions, as it accounts for the three types of error that occur: false negative (FN), false postive (FP) and IDSW. False negatives are defined as an object that is not tracked, false positives are defined as tracked objects which should not be and identity switches are when two objects that should be tracked swap identities. Fragmentations are defined as the number of times an identity switches from "tracked" to "not tracked".

$$\text{MOTA} = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t GT_t} \tag{4.2}$$

However, as MOTA has shortcomings in terms of how it accounts for identity switches [8], we also report tracking metrics using IDF1, which is also included as a metric in the MOT Challenge, as this provides a much more global way to assess the performance of the tracking system in terms of its ability to track identities.

Behaviour Metrics Extraction Evaluation

All detected tracks were grouped respective to the ground truth pig ID to which they belonged. For total distance travelled and time spent idle, the frame-to-frame estimations belonging to each pig were summed and speed was averaged. All behavioural metric estimations were evaluated by normalising the values to between 0 and 1, followed by calculating mean squared error (MSE) (Equation 4.3), where i is the pig ID, Y is the ground truth behaviour and $\hat{Y}$ is the predicted behaviour.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4.3}$$

Additionally, we calculated the absolute error for each of the behaviour metrics for each pig identity and carried out a paired Wilcoxon test [207] on them to determine if the absolute errors for each method were significantly different.

4.3. Results

4.3.1. Detection Results

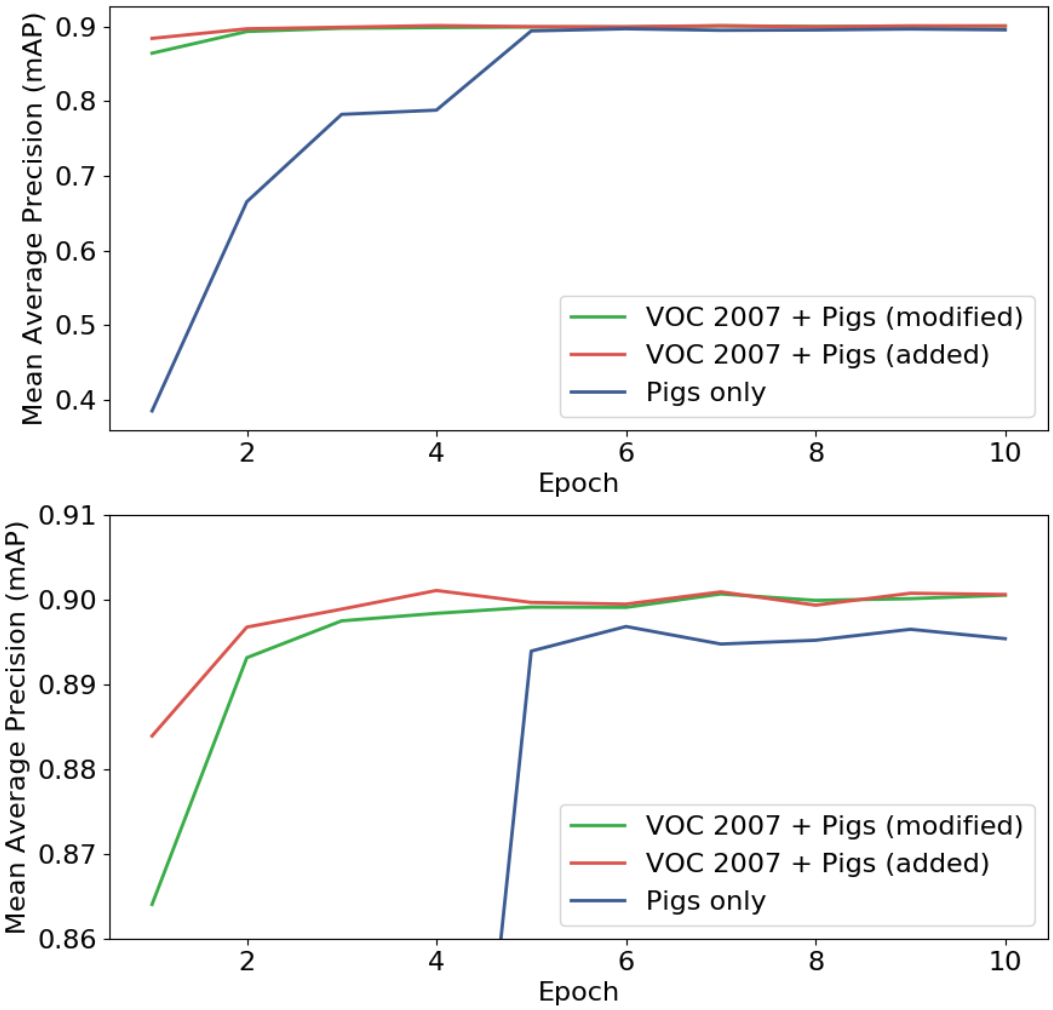


Figure 4.8 Top: Performance of Faster R-CNN models trained on a dataset of pigs in a live farm using 2 methods of transfer learning from a model pre-trained on VOC: adding an additional fully-connected layer and modifying the final fully-connected layers along with a model trained only on the pig data. **Bottom:** The same data zoomed in to highlight the difference between the two similarly performing models.

Results from the Faster R-CNN (Figure 4.8) show that the performance difference between adding an FC layer or modifying the final FC layers was negligible for our application. This was

also the case for detection (inference) speed, as both models had an average inference time of 80ms per frame. Both models that were pre-trained on VOC outperformed the model which was only trained on the pig data in terms of performance and speed of learning, hence showing the effectiveness of the transfer learning strategy.

The model that was only pre-trained on ImageNet (but not on VOC) achieved good results (mAP $= 0.894$) but never reached the same level of performance as the other two models (mAP ~ 0.901). This is as expected as the other models are not only pre-trained on more data, but also on a related task, which is highly beneficial in transfer learning, but nonetheless the inclusion of this option in our comparison provides us with a useful performance baseline.

Figure 4.9 Four sample images from our pig detection test set processed by the Faster R-CNN with the feature extraction layers pre-trained on ImageNet, the rest pre-trained on VOC and an additional fully-connected layer for the pig dataset. Detections to the left of the red wall are ignored. The top left image is from the low-light test segment. The top right image is from the densely packed test-segment. The bottom left image is from the overexposed test segment. The bottom right image is from the "many pigs" test segment.

Figure 4.9 shows examples of detections made from each of the test segments defined in Section 4.2.1. The top left image shows the Faster R-CNN correctly detecting 2 pigs from the low-light test segment. The top right image shows that the model was capable of detecting pigs from images in the densely packed pigs test segment. The bottom left image exemplifies issues relating to overexposure caused by strong sunlight which distorts the edges of the pigs, making them more difficult to detect. The model does appear to have suffered from the camera being at an angle rather than pointing directly down. This causes some pigs at the top of the images to be hidden behind other pigs in such a way that distorts their shape and makes them undetectable. This is particularly noticeable in the bottom right image, which is from the test segment for images containing many pigs.

In order to proceed with adding MOT tracking to our method, we selected the model which added an additional, final layer to the FC layers. We used this model to evaluate the test segments individually (Table 4.3).

Test Segment	mAP
Many pigs	0.905
Densely packed	0.906
Overexposed	0.906
Low-light	0.850

Table 4.3 The parameters used for the Faster R-CNN that perform best on the VOC dataset.

Test segments with many, densely packed, and overexposed pigs performed in line with the rest of the dataset. However, images that suffer from low-light relatively under-perform. This is mainly due to the fact that Faster R-CNN is an image-based detection method, which requires light in order to detect objects. Nonetheless, the implementation does perform moderately well in low-light conditions.

4.3.2. Association Metric Learning

As described in Section 4.2.3, the CNN used to learn the association metric was trained using MARS followed by our own pig Re-ID dataset (Section 4.2.1). Compared to the MARS dataset (1.1 million images), our pig Re-ID dataset is very small (5,653 images). Despite this, the fine-tuning increased the rank 1, 3, and 5 CMC by 17%, 15%, and 15% respectively, achieving a rank 1 CMC of 0.788. The additional training also increased the overall mAP from 0.760 to 0.862. This increase in performance is to be expected, as we are fine-tuning, but it is valuable to understand by how much the additional training has improved performance.

4.3.3. Tracking Results

As discussed, the SORT tracker is heavily dependent upon accurate detections. Therefore, as the detector was capable of achieving a high level of mAP (0.901), it was expected that the tracker would perform similarly well.

The direct output of the SORT tracker is a series of IDs, which then are mapped to our manually-annotated tracks (Section 4.2.1). From this mapping process, we can identify when the tracker detects an object that does not exist (FP), when the tracker is not able to detect an existing object (FN), or when the identifiers of two tracks are switched (IDSW) [205, 206, 208]. The result of this implementation was a large number of "tracklets" (partial tracks), subsets of which belong to individual pig identities.

SORT achieved a score of 95.1% MOTA. In total there were 153 FPs, 331 FNs, 50 IDSWs and 56 fragmentations. The average number of consecutive frames which perfectly tracked all pigs was 21.746 frames (5.437 seconds), with a maximum of 208 frames (52.000 seconds). The average length of time an individual pig could be perfectly tracked for was 129.358 frames (32.339 seconds), where the maximum was 981 frames (4 minutes). To further characterise the results of our method, Table 4.4 reports, for each of the 25 unique pig IDs, the percentage of frames for which such ID was correctly tracked. Five out of the 25 IDs had a perfect tracking score of 1.00, and 21 out of 25 had at least 0.9 successful tracking proportion. One ID had a very poor score of less than 0.5.

From the metrics derived, we can see that the general implementation works well (95% MOTA), but the occasional dropping of frames caused by the implemented recording system seriously impacts the continuity of IDs given to pigs between frames which is better represented by the 70.3% IDF1 score.

		SORT				**Deep SORT**			
ID	F	T ↓	C ↑	S ↓	FN ↓	T ↓	C ↑	S ↓	FN ↓
1	1875	11	0.97	10	48	6	0.95	4	97
2	540	2	0.99	0	5	3	0.96	1	20
3	1747	6	0.98	4	33	6	0.96	4	73
4	983	2	1.00	0	2	2	0.99	0	8
5	119	2	0.99	0	1	4	0.92	2	8
6	122	3	0.95	1	5	3	0.80	1	23
7	63	1	1.00	0	0	2	0.92	0	5
8	9	3	0.33	1	5	2	0.33	0	6
9	100	2	0.97	0	3	3	0.91	1	8
10	52	1	1.00	0	0	1	1.00	0	0
11	630	7	0.97	5	16	5	0.92	3	48
12	608	18	0.79	16	110	7	0.81	5	111
13	64	4	0.83	2	9	4	0.61	2	23
14	501	2	0.99	0	3	2	0.98	0	9
15	429	3	0.94	1	25	4	0.87	2	53
16	427	6	0.90	4	38	5	0.69	3	130
17	489	1	1.00	0	0	1	1.00	0	0
18	263	3	0.96	1	9	3	0.92	1	20
19	825	3	1.00	1	3	3	0.97	1	22
20	666	5	0.98	3	9	4	0.96	2	25
21	13	2	0.77	0	3	2	0.08	0	12
22	141	3	0.98	1	2	3	0.94	1	7
23	27	2	0.93	0	2	2	0.70	0	8
24	117	1	1.00	0	0	2	0.92	0	9
25	49	1	1.00	0	0	2	0.82	0	9
Avg.		3.76	0.90	2	13.2	3.24	0.84	1.32	29.3

Table 4.4 Results of the SORT & Deep SORT tracking algorithm used to track individual pigs. ID is the ground truth ID for a pig, F is the ground truth for how many frames the pig was visible, T are the number of tracklets the method created for each individual pig, C is the percent of the ground truth tracks that were tracked by the method, S is the number of identity switches that occurred, FN is the number of false negatives (pig not detected). The arrows indicate whether lower or higher is better. There were also 153 and 105 total False Positives (a pig was detected that did not exist) for SORT and Deep SORT respectively.

Unlike SORT, Deep SORT is less reliant upon accurate detections, though it does still require them to be of good quality as it still makes partial use of the Kalman filter to make assignment decisions between the existing tracklets and detections from the following frame. Deep SORT achieved a score of 92.1% MOTA. In total there were 105 FPs, 734 FNs, 33 IDSWs, and 40

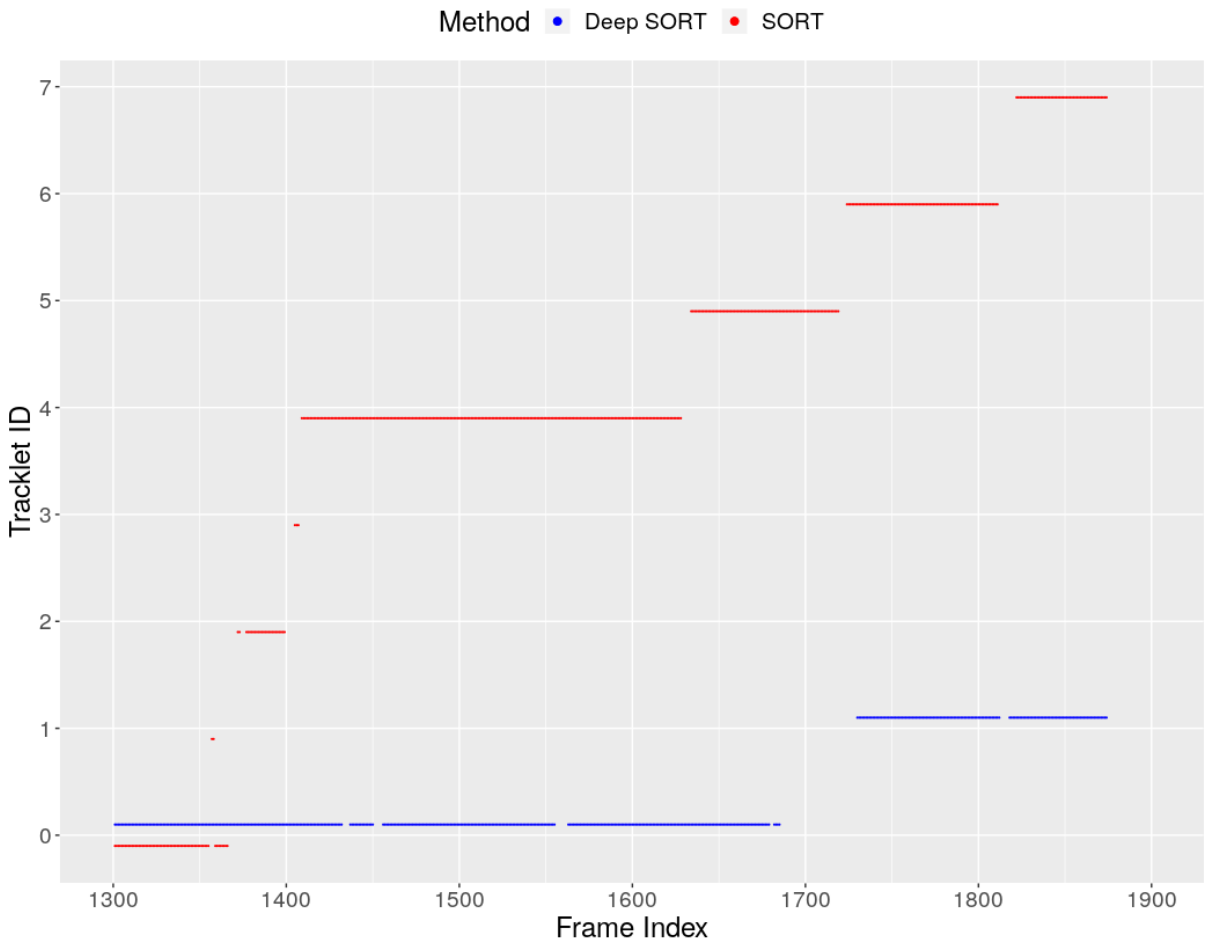


Figure 4.10 Representation of the detected tracklets for pig 1 from frame 1300 to 1875. This pig was visible for all frames, but showing the detail at this segment of frames was not possible if we showed all the tracklets from all frames. The Y-axis shows the tracklet IDs (which are independent for each method), the X-axis shows the frames during which the pig was visible. Red represents SORT generated tracklets, blue represents Deep SORT generated tracklets.

fragmentations. The average number of consecutive frames which perfectly tracked all pigs was 24.163 frames (6.041 seconds), with a maximum of 208 frames (52.000 seconds). The average length of time an individual pig could be perfectly tracked for was 197.882 frames(49.471 seconds), where the maximum was 981 frames (4 minutes). Two out of the 25 IDs had a perfect tracking score of 1.00, and 16 out of 25 had at least 0.9 successful tracking proportion. Two IDs had a very poor score of less than 0.5.

Although there was almost double the number of FNs, there was a 32% decrease in the average number of IDSWs and a 31% decrease in the number of false positives raised by the system. Despite the increase in FNs being substantial, and therefore also decreasing the proportion of track coverage, it was considered a fair trade-off, as a FN is much easier for the system to recover from than an IDSW. This is because an IDSW tends to be permanent, where a FN may only be a for a few frames, after which, the identity can be recovered. Because of this ability to recover from FNs and the decrease in IDSWs the introduction of the visual Re-ID component within Deep SORT results in an increase in IDF1 to 73.4% and a 30% decrease the

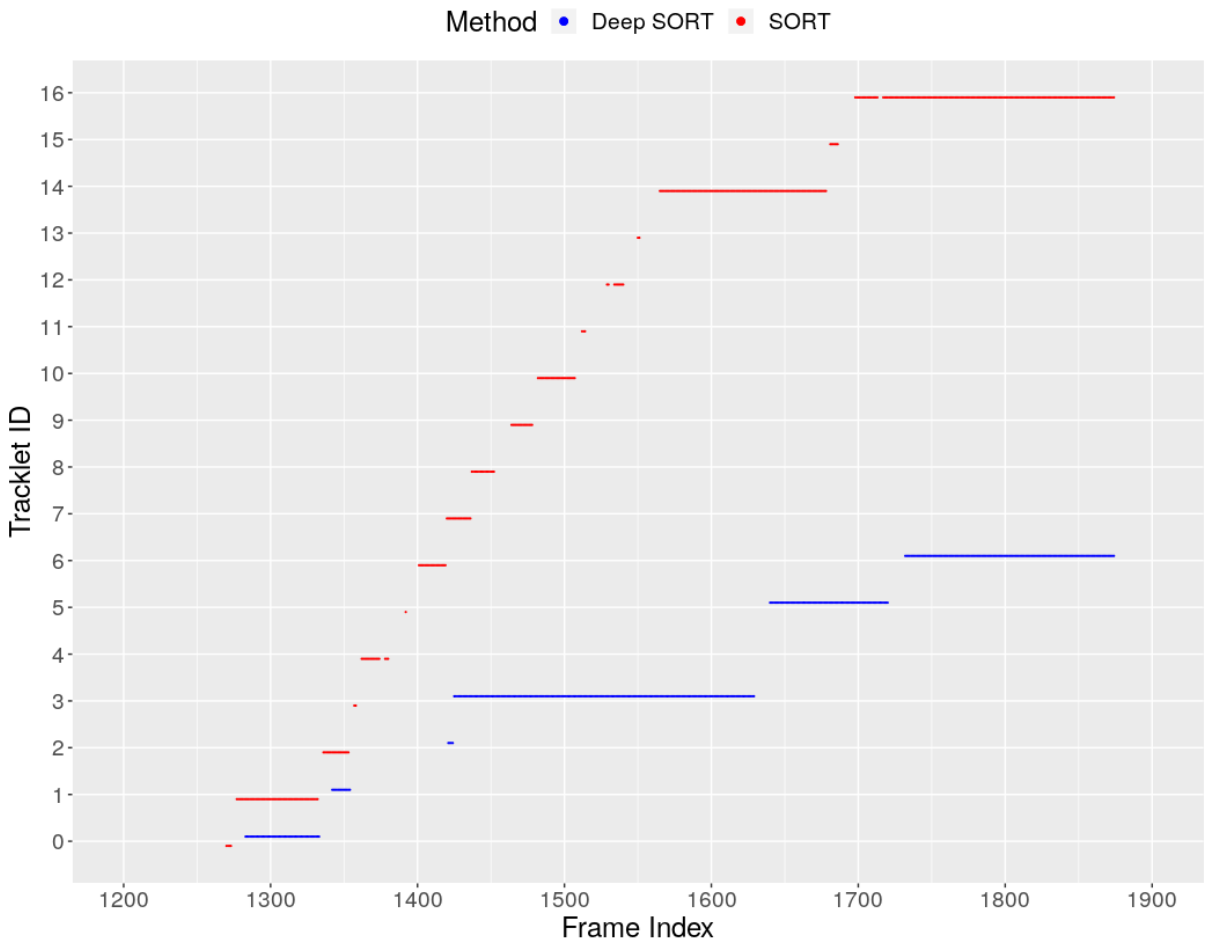


Figure 4.11 Representation of the detected tracklets for pig 12 for all the frames it was visible. The Y-axis shows the tracklet IDs (which are independent for each method), the X-axis shows the frames during which the pig was visible. Red represents SORT generated tracklets, blue represents Deep SORT generated tracklets.

number of fragmentations, which increases the average length of a track increases by 17.132 (+16%) and the average number of frames in which all pigs are perfectly tracked increases by 0.604 seconds (+11%). The maximum number of frames where all pigs are perfectly tracked, and the maximum frame length remains unchanged from those reported for SORT.

Naturally, FPs, FNs and IDSWs are to be avoided, as they each decrease the general performance of the tracker. In our application, FPs are commonly caused by substantial changes in object location between frames, often cause by dropped frames, or by the occasional rogue detection produced by the object detector. This results in tracklets for pigs that don't exist that are subsequently analysed to extract behaviour metrics. As this is not a real pig, it is unlikely that the tracks produced by false positives would result in behaviour metrics similar to that of pigs, which would likely produce a spurious alert to the farmer.

While FPs are undesirable, it is far less damaging to raise unnecessary alerts than to miss alerts that should be raised as is the case with FNs. A high number of FNs means that objects are not being tracked, which translates to periods of time where a pigs behaviour is not being

analysed for potential problems. As long as periods of FNs are relatively short, Deep SORT is able to recover from them. This is why the increase in FNs from SORT to Deep SORT is not a concern. In fact, it is a by-product of the implementation that requires the first n frames of a "new" tracklet to be left unassigned so that a robust history of visual appearance can be built. This helps to prevent spurious appearance metrics from damaging existing tracklets. This is why our results showed a lower track completion for Deep SORT despite a lower average number of tracklets and IDSW per pig. A post-processing step could be introduced to mitigate the impact of this effect in the final results. On the other hand, IDSWs are much more challenging to detect without ground-truth data and are subsequently very challenging to recover from. A high number of IDSWs renders the tracker much more unreliable and can result in one pig being treated for another pigs illness, leaving the ill pig untreated.

These challenges can all be exacerbated by a poorly performing CNN used to determine the visual similarity of two objects. This is particularly a concern in our application as the pigs are very similar in appearance, meaning it is easier to mistake two different pigs as the same pig, resulting in IDSWs across large distances. If we had been unable to achieve a high mAP in the CNN component of this tracker, one option to mitigate the negative effects of this would be to give more weight to the Kalman filter component of Deep SORT. The decrease in average IDSWs from 2 to 1.32 from SORT to Deep SORT respectively is a major performance improvement and indicates that the cosine metric learner was well trained. The combination of a strong mAP, Rank 1 CMC and reduced number of IDSWs indicates that Deep SORT is capable of being deployed into commercial farm environments to track medium-sized objects that are indistinguishable by eye. The main drawback of this approach is the number of frames per second that can be processed, 20 FPS compared to SORTs 260 FPS. Although being able to track at higher frame rates would likely improve performance of any tracker, 20 FPS is still more than the average FPS we were able to record video on a commercial farm (4 FPS) and was therefore more than capable of processing our footage in real-time.

Case Studies

In Figure 4.10 we show an example of Deep SORT's ability to consistently recover tracks between missed detections enables it to outperform SORT. The visualisation is of the tracklets generated for pig 1, the longest tracked pig in our dataset. The first tracklet in the visualisation, beginning at frame 1300, was lost and subsequently recovered 3 times when using Deep SORT. SORT is capable of recovery, but this is only possible when the detection is lost very briefly. This is why the gaps between recovered tracklets are consistently small, whereas Deep SORT can handle longer drops in detections, which is why Deep SORT is a much more robust method. This is also visible in Figure 4.11, which shows the detected tracklets for pig 12, the ID which benefited from the greatest reduction in IDSWs by using Deep SORT. SORT generated 18 tracklets for this identity, whereas Deep SORT only generated 7. We can see that tracklets

generated under Deep SORT were much more stable than that of SORT; all whilst having greater track coverage (Table 4.4).

4.3.4. Behaviour Metrics Extraction Results

	Distance			**Avg. Speed**			**Idle Time**		
ID	True	SORT	Deep SORT	True	SORT	Deep SORT	True	SORT	Deep SORT
1	7811	10868	12024	16	260	136	395	290	266
2	7368	6762	5830	53	50	75	79	42	52
3	3514	5638	5879	8	149	145	399	346	338
4	2224	3026	2814	9	12	11	226	214	213
5	1135	1498	2843	38	34	128	21	21	24
6	1745	1390	1050	56	94	81	18	10	10
7	397	579	520	25	35	34	12	5	5
8	264	1498	182	105	34	25	1	21	4
9	303	592	1256	11	21	60	22	18	26
10	179	366	316	10	21	19	11	10	10
11	1385	2791	3050	8	300	94	145	116	126
12	9749	5722	5346	63	798	217	103	47	155
13	1470	823	1619	90	132	219	8	5	40
14	994	1946	1841	7	15	14	115	89	88
15	1807	2922	2154	16	67	49	95	50	53
16	2720	2649	1733	25	172	172	91	51	40
17	229	1181	1174	1	9	9	116	109	108
18	2381	2090	1981	35	83	84	46	38	37
19	1393	1975	1755	6	32	26	191	181	178
20	2976	3916	3922	17	172	116	142	96	95
21	297	268	546	88	89	32	1	0	6
22	551	1713	1661	15	54	56	26	39	38
23	764	730	523	110	100	85	0	0	0
24	917	1125	1092	31	38	39	20	16	15
25	717	740	670	58	58	61	5	3	3
MSE ↓	-	0.010	0.015	-	0.148	0.008	-	0.003	0.008
Wilcoxon Test P-Value	-		**0.221**	-		**0.037**	-		**0.331**

Table 4.5 Results of the behaviour extractions and the ground truth associated with them. Results are shown for SORT and Deep SORT. Distance is measured as the number of pixels travelled, the average speed is measured as the average number of pixels travelled per second, and idle time is measured as the number of seconds a pig did not move more than 4 pixels. These results are normalised and the mean squared error (MSE) is shown for each (lower is better). The absolute error between the estimated behaviour and true behaviour for each method and metric is calculated; the number of IDs where this error is below a threshold is counted (higher is better).

The behavioural metrics extracted from both SORT and Deep SORT tracks are shown in Table 4.5. Total distance and time spent idle scored well (0.010 MSE and 0.003 MSE respectively), however the average speed estimations substantially underscored (0.148 MSE). In particular, the average speed calculations were overestimated for the tracklets in almost all cases (80% of pigs). Estimations for pig 12 across all metrics derived from SORT were largely incorrect. This is reflected in Table 4.4, as this ID incurred a high number of IDSWs (16). The metrics derived from Deep SORT, however, are more accurate for this pig due to the considerable decrease in IDSWs (5).

This relationship between a high number of IDSWs and poor estimations of behavioural metrics, specifically time spent idle and average speed, can be seen through all pigs (e.g. pigs 1 and 12). A higher number of IDSWs results in an overestimation of average speed. This is confirmed by the improved behaviour extraction when using Deep SORT (Table 4.5),

which has a significantly lower error for average speed, which is why there is a substantial performance improvement for this metric when using Deep SORT (0.008 MSE, -95%). However, this relationship is much less consistent when calculating the total distance travelled. The total distance (0.015 MSE) and time spent idle (0.008 MSE) metrics extracted from Deep SORT show no statistically significant performance change over SORT.

4.4. Discussion

With the increasing use of deep learning for multi-object tracking [209], particularly in crowd analysis (i.e. people tracking), it is valuable to evaluate the extent to which these methods can be applied to more difficult applications. Applications can be more difficult for varying reasons such as: low FPS recordings, poor image quality, and similar-looking objects.

Individual pig tracking on a commercial farm is an example of an application where all of these challenges can be found. This is mainly due to fact that the available infrastructure is limited and pigs are much less distinguishable from one another when compared to humans, especially with clothing taken into account. Deep learning methods for computer vision are much less common in the agricultural field, which tends to rely on classical signal processing methods, especially in the area of individual animal tracking. The ability to track individual pigs in real-time is key to creating a full system that can provide the information required for their management, including early detection of disease.

Previous research has focussed on how depth cameras can be utilised to track pigs under the challenging conditions presented by farm environments, such as poor image quality and low network bandwidth for data collection [35, 37, 36, 210, 211]. These methods have been able to achieve good tracking results (89% MOTA [35]). However, they rely upon accurate depth sensor data which is only achievable at short distances, which limits the distance a camera can be placed from an object. This limits the field of view, and thus the pen coverages of a single depth-camera meaning more cameras are required to cover a large area when compared to the RGB cameras we use.

More recent research has applied deep learning models to the tracking of pigs [95] using object detection models, similar to the implementation used in our method, on RGB images, obtaining 89.58%. However, this depends upon IDs being sprayed on the pig and detecting each pig as a separate class. This is not feasible or practical in a commercial setting due to the large number of pigs that may reside within a pen. Moreover, the sprayed markers do not hold for long and therefore would need to be continually reapplied for reliable tracking. Secondly, the model needs to be specifically trained on each numerical ID that it needs to track, which impedes the generalisability of the method.

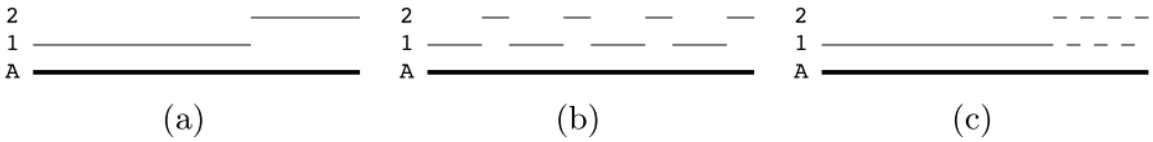


Figure 4.12 "Where there is one true identity A (thick line, with time in the horizontal direction), a tracker may mistakenly compute identities 1 and 2 (thin lines) broken into two fragments (a) or into eight (b, c). Identity 1 covers 67% of the true identity's trajectory in (a) and (b), and 83% of it in (c). Current measures charge one fragmentation error to (a) and 7 to each of (b) and (c). Our proposed measure charges 33% of the length of A to each of (a) and (b), and 17% to (c)." (This figure and caption are from [8]).

YOLO [71] is a commonly used method for object detection that is popular for its fast inference times was considered for use in our application. However, despite the real-time inference, its performance on detecting smaller objects is much poorer than that of Faster R-CNN. YOLOv3 does attempt to rectify this, but it does so at the cost of poorer performance on larger objects relative to Faster R-CNN [87]. As we wanted our method to generalise to different environments such as taller buildings where the camera is mounted higher, thus making pigs smaller in the images, and because we do not deal with high FPS videos, we decided against using it.

The use of standardised metrics for MOT in all applications is key to summarising how well methods perform in various domains and applications as concisely and as accurately as possible. Of the research discussed above, several report MOTA, but none report IDF1. This poses a risk of skewing the effectiveness of the tracking method that has been implemented, by misrepresenting the effect of IDSWs within the system. Figure 4.12 (taken from [8]) demonstrates an example where there are varying examples of IDSW. Based on these tracklets, MOTA would favour (a), as it has the fewest number of fragmentations; where IDF1 would favour (c) it provides a greater identity coverage (83% vs 67%). Hence, we believe that in our context, the under-reported IDF1 metric is a more informative tracking quality metric than the more widely used MOTA.

Our method was able to detect, track and extract behavioural metrics of individual pigs in real-time using a Faster R-CNN for localisation and Deep SORT for tracking. Alternative methods for tracking do exist (e.g. a kernelized correlation filter [212]), however, we chose SORT and Deep SORT as they provide a good side-by-side comparison of trajectory-based tracking and a combined trajectory and visual-based tracker respectively. This results in our main contribution, which is a complete, end-to-end system that can process raw images and produce behavioural metrics for individual pigs, whilst maintaining the identity of pigs between frames. We relied solely on footage recorded from an inexpensive RGB camera, as opposed to expensive 3D depth cameras, recording at an average of only 4 FPS and a resolution of 1920 x 1080. This low FPS was a limitation of the hardware that was used, as it was responsible for other background tasks which limited the number of frames that could be captured.

From this data, we were able to achieve comparable performance to previous research in terms of MOTA (95%), with the improvement that our method can do so without the use of

additional or expensive hardware, such as RFID tags and depth cameras, or visual aids, such as painted IDs. We were also able to use this data to reliably re-identify pigs when they become occluded or the detector fails to localise them (0.862 mAP), which enables us to achieve an IDF1 of 73.4%. This use of visual appearance when assigning identities at runtime proved valuable, as the method was able to more accurately determine the average speed of a pig, without compromising on other metrics, as this is easier to do when the number of IDSWs is lower. As our data was restricted to the pig sizes that were available, we were not able to verify whether the method will work as pigs grow.

Where other research focussed on performance aspects (e.g. weight) of individual pigs [213], we looked specifically at how active pigs are by extracting movement-related behavioural metrics from the generated tracks. We recorded pig behaviours for up to a theoretical maximum of 10 minute video segments. This is equivalent to the behavioural method of scan sampling, where all of the actions of all animals are recorded for intervals in order to obtain behavioural metrics for individuals within a group [214]. We have shown not only that we are able to successfully localise and track pigs from challenging naturalistic settings (commercial farms), but also that we are able to successfully extract a range of domain-relevant, useful knowledge from the outputs of fairly generic object localisation and tracking algorithms as the ones that we have used. One of the advantages of the method is that we are able to extract several behaviours from the same pig in real-time. This is beneficial, as it is suggested that a combination of behavioural metrics might be a better indicator of pig health than a change in a single behaviour [215].

4.5. Conclusion

We have implemented a system to detect and track pigs in a commercial farm setting using deep learning that allows us to track pigs for up to 4 minutes, with an MOTA of 92% and IDF1 of 73.4% without the use of additional hardware or visual aids. The tracks derived from this system are able to be used to calculate behavioural metrics for total distance travelled, average speed and time spent idle for individual pigs. The length of the identified tracks is mostly limited by the length of the realistic video we could use to evaluate the method, due to technical constraints in the data capturing system. However, this method generates a set of tracklets that (mostly) successfully cover parts of the annotated tracks and in cases where detections are missed, is capable of recovering the identity. Overall, our work shows how deep learning algorithms enable the development of a relatively inexpensive pig monitoring system that can provide useful information to characterise pig's behaviour by applying transfer learning strategies on top of a standard object localisation method such as Faster R-CNN. The literature shows how such descriptors enable the creation of more personalised pig treatment plans [157, 178] which in turn decrease disease risk and reduce the use of medication, whilst maintaining animal performance.

Chapter 5. Pig Tracklet Stitching for Improved Individual Pig Tracking Using Deep Re-Identification

Abstract

As commercial pig farming moves to embrace the concepts of Precision Livestock Farming, providing care on an individual pig level, rather than as groups, is vital. Human observation is often used to monitor to health and wellbeing of pigs, either in-person or using video footage, this can very resource-intensive and does not scale well for applications in larger farms. In our previous chapter, we presented a complete workflow for the detection and tracking of individual pigs within an operating, commercial farm. Through this implementation, we discovered performance issues relating to the recovery of identities after long-term occlusions, which we attributed to our tracking algorithm partially relying upon trajectory as well as visual appearance. Deep learning has shown good performance in the tracking of humans across multiple cameras, using CNNs as a means for re-identification. However, humans are often easily distinguishable from one another, often due to differences in clothing, whereas pigs are typically indistinguishable by eye. Therefore, the work in this chapter focused on evaluating the re-identification methods typically used to track humans across multiple cameras for the purpose of re-identifying pigs after long-term occlusions. We showed that the latest developments in CNN-based feature extraction for images can be used to improve the tracking of individual pigs by using it to assign seemingly newly detected pigs to their original identities, thereby increasing the IDF1 tracking metric of the tracker and lowering the number of identity switches.

5.1. Introduction

In the previous chapter, we developed a method to localise, track, and extract behavioural metrics on individual pigs, using a regular RGB camera, in an effort to move towards a more individual level of care that does not require human supervision. We evaluated how a Faster R-CNN could be used to automatically detect pigs within images, and then track these detections in videos using both a trajectory-based method (SORT)[182] and a method that combined trajectory and visual appearance (Deep SORT)[200]. We demonstrated that despite pigs often being indistinguishable from one another by eye, CNNs can be used to enhance trajectory-based methods in order to re-identify pigs after events that would otherwise cause their original identity to be lost. However,

this was only achievable after brief gaps in detections; long occlusions typically resulted in the existing identity being lost and a new identity being assigned to a pig.

In this chapter, we have evaluated several state-of-the-art, CNN-based Re-ID models on a dataset of pig identities. We have demonstrated, firstly, how such models can be used to re-identify pigs based on their appearance alone and secondly, how these models can be used to recover from the long-term occlusions that Deep SORT was not capable of recovering from on its own. This was achieved without the need for additional hardware (e.g. RFID tags), physically marking the pigs (e.g. drawing numbers on their back), or to define a "tag-box" which can be used to identify pigs based on a reduced portion of the pig [101]. Additionally, we show that, despite being trained on high-quality images, the methods generalise well enough to work on significantly lower resolution images.

The primary contribution of this chapter is demonstrating that existing tracking methods can be augmented without changes to their original implementation, to improve their multi-object tracking capability, increasing IDF1 and decreasing the number of identity switches, whilst still being able to operate in real-time. We achieved improved results without the need for additional hardware (e.g. RIFD tags) or physical markings (e.g. identities painted onto pigs). Secondly, though Re-ID methods that only use visual information are commonplace in tracking humans [200, 185, 203], they have not previously been used on pigs, which are often indistinguishable to the human eye. Therefore, our dataset provides a benchmark for how far CNN-based Re-ID models can be pushed. We have shown that even when using this challenging, low-resolution data, several of the CNN-bsaed Re-ID methods we evaluated were able to achieve very high performance in terms of mAP (0.737) and CMC at Rank-1 (0.957).

Section 5.2 describes the datasets, both benchmark and custom, that were used to carry out this work. Section 5.3 outlines the training and evaluation procedures for the methods used for detection, tracking and tracklet stitching. Section 5.4 and Section 5.5 present and discuss the results respectively, followed by our conclusion in Section 5.6.

5.2. Dataset Descriptions

This research made use of three, standard benchmark datasets: ImageNet [183], used for pre-training our detection and re-identification models; Pascal Visual Object Classes Challenge 2007 [184], used for pre-training of our detection model; and Motion Analysis and Re-identification Set [185], used for pre-training of our re-identification models. Additionally, 3 custom-built datasets were used: one for detection (training and testing), one for re-identification (training and validation) and finally one for tracking (the overall test set), a subset of which was used to test the re-identification models.

ImageNet is an image classification dataset consisting of around 14m images of objects each falling into more than 10k categories. VOC is an object localisation and classification dataset consisting of 5,304 images and 9,507 objects that are split evenly between training and test sets. Within this dataset, there are 20 different classes. Finally, MARS is a video dataset for person re-identification that was recorded across 20 cameras. It consists of 625 human identities for training (8,298 tracklets) and 636 for testing (12,180 tracklets). The identities are spread across 6 cameras, though not all identities span all cameras. Each identity has a set of short, sequential images of an individual identity, referred to as tracklets, from a varying number of the cameras.

All images used in the detection and re-identification custom datasets were recorded using the RGB sensor of a Microsoft Kinect v2 (resolution: 1920×1080, field of view: 84.1×53.8, focal length: 3.29, shutter speed: 14ms, maximum frame rate: 30 FPS). The tracking dataset was recorded on a 640×360 RGB camera with all other parameters kept the same. No special features of the Kinect camera were used and any other RGB sensor with similar parameters would achieve similar results. All cameras were mounted to the ceiling, pointing at the floor, at a slight angle (rather than perpendicular to the floor) in order to capture the entirety of half of the pen. The building used was comprised of 4 pens, each housing 20 pigs of the same age and similar weight. The pigs used in the detection and re-identification datasets are the same pigs captured on different days in the same pen, whereas the tracking dataset was a different cohort of pigs in a different pen. The images used in these pig datasets were collected at Newcastle University's Cockle Park farm and fell into 5 distinct categories: densely packed, overexposed, low-light, many pigs, or normal. Because the images were collected from pigs husbanded under farm conditions, there was no need for ethical approval.

5.2.1. Pig Detection Dataset

As described in Section 5.1, our custom detection dataset is identical to that used in our previous chapter. The dataset is comprised of 1,646 images, each manually annotated to note the location of each pig in each image, resulting in 9,014 annotations (Figure 5.1). The images are of 20 pigs across differing days and times and were all sourced from a single camera. The images were split equally between training and testing data, where days used for test data did not overlap with training data.

5.2.2. Re-Identification Datasets

As part of the previous chapter, we also built a pig Re-ID dataset consisting of 20 pigs across 2 cameras covering a whole pen. This dataset contained an average of 280 images per identity. We extended this dataset with additional footage to contain an average of 885 images per identity. These identities were broken down as 15 identities for training and 5 identities for validation

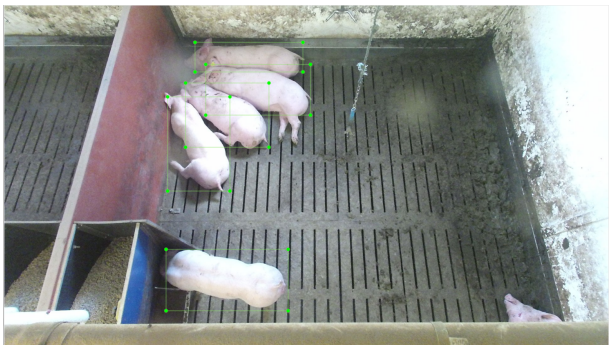


Figure 5.1 An example of an image used within our manually annotated pig detection dataset. This particular example shows how, in some images, the pigs are densely packed into one area.

(840 query tracklets & 3366 gallery tracklets). As our pig Re-ID dataset was video-based, it was broken down into sequences of 10 images using a sliding window.

We structured the dataset similarly to MARS by creating tracklets such that

$$X = \{\{x_{t+i}, \ldots, x_{t+10+i}\}, \ldots\}$$

where t is the frame number of the video the image is sourced from and i is the tracklet number. As in MARS, tracklets that spanned cameras were discarded.

Though the structure and design of our Re-ID dataset are similar to that of MARS, it is a distinctly more challenging dataset. The most notable reason was the difficulty in distinguishing identities from one another (Figure 5.2). Where with person re-identification, certain cues can be used, such as clothing colour and gait, this is not possible in our dataset. Though pigs will naturally be different colours, the amount of variation is substantially lower than that within human clothing.

Figure 5.2 Top: A sample of two identities of the MARS dataset for person re-identification. **Bottom**: A sample of two identities of the pig re-identification dataset.

Typically, methods used for re-identification are dependent upon a large number of identities in order for the distance metric to be maximised between two feature vectors of different pigs. As our dataset consists of considerably fewer identities than are in MARS (625 in MARS and 20 in the pig Re-ID dataset), it makes generalisation a challenging task.

We used a separate test set to evaluate the performance of each model that was comprised of a subset of the identities from dataset outlined in the following section (Section 5.2.3).

5.2.3. Pig Tracking Dataset

Unlike the other custom datasets, the dataset used for evaluating the final tracking result and re-identification models was not developed as part of our previous work. This new dataset consisted of frames extracted from a 9.3 minute video recorded from a single camera recorded at a significantly lower resolution (640×360). We use a lower resolution for this testing dataset in order to evaluate how well the methods will generalise from being trained on data with a high resolution (1920×1080), though in our implementation the images were resized to 1920×1080 using bilinear interpolation.

Each frame was manually annotated with the location and identity of each pig in order for the pig to be tracked between frames (Figure 5.3). As the camera did not cover the entire pen area, if a pig left the field of view of the camera, it was not possible to reassign its previous identity, meaning a new identity had to be assigned. Therefore the 20 pigs used in this dataset resulted in 74 identities. This means that there are a large number of pigs with multiple identities in the ground truth data. We, therefore, selected the 6 identities that lasted the longest period of time and manually verified that they were not the same pig, to be used as the test segment for the dataset outlined in Section 5.2.2.

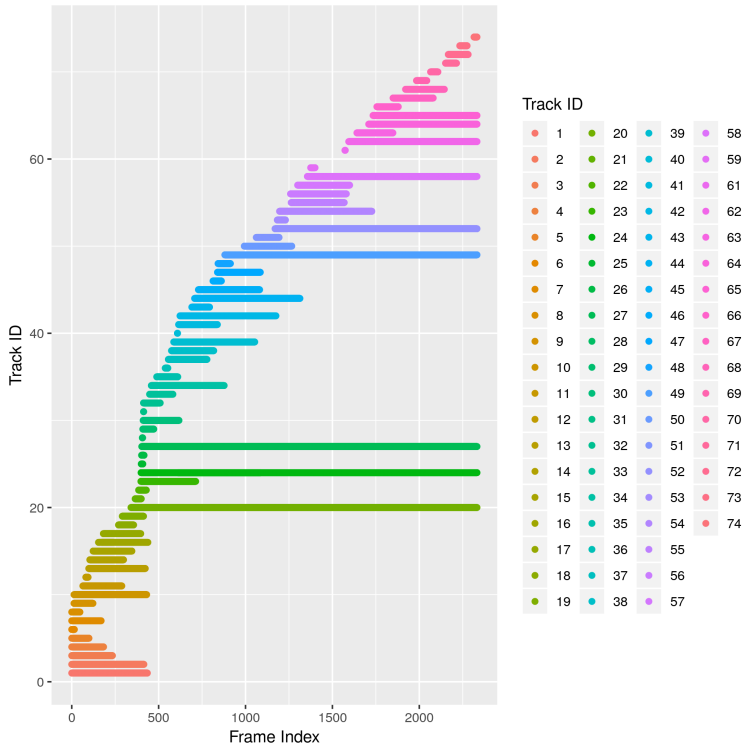


Figure 5.3 A representation of the manually annotated pig tracks used in the pig tracking dataset. The Y-axis shows the ground-truth pig ID, the X-axis shows the frames during which the pig was visible. Once a pig left the camera, it was not possible to re-identify it, and was therefore given a new ID.

The FPS of the video was variable (averaging 4 FPS) as background jobs running alongside image capturing affected the performance. High FPS in video footage makes tracking objects easier as the distance an object can move between frames is much lower. However, due to technical limitations of the infrastructure used to record this data, it was not possible to record at an average frame rate higher than 4.

5.3. Methods Applied

5.3.1. Pig Detection & Tracking

In this section, we will briefly cover the detection and tracking methods used to create the baseline tracking that our tracklet stitching methods would be applied to. This is the same as the Deep SORT configuration in our previous chapter. All models were trained on an Nvidia Tesla P100 graphics card.

Pig Detection

Faster R-CNN was used to detect the location of pigs within each frame. This method consists of 3 main components: feature extraction, RPN and FC layers. ResNet-101 [70] was used for the feature extraction layer and was initialised using weights pre-trained on ImageNet. The whole model was then trained on VOC using stochastic gradient descent (learning rate: 0.001, learning rate decay step: 5, learning rate decay multiplier: 0.1, batch size: 8).

Transfer learning was then used to continue the training of the model on the pig detection dataset, outlined in Section 5.2.1. This was achieved by adding an extra FC layer to the model with two output nodes (one for "pig" and one for "background"), which created an additional 78 trainable parameters; a relatively insignificant amount. Our previous chapter demonstrated that this worked marginally better than modifying the final layer.

Pig Tracking

In order to track pigs between frames, we implemented Deep SORT, which utilised both the trajectory of pigs and similarity in their visual appearance, in order to retain identities between frames. This method used a combination of a Kalman filter [196] and the Hungarian algorithm [197] to calculate the distance metric. Similarity in visual appearance was evaluated using a lightweight CNN, which used a re-parametrisation of the softmax classifier that included a measure of cosine similarity in the representation space [200]. This model was initially trained on MARS using the ADAM [63] optimiser with a learning rate of 1×10^{-4}, using cosine decay

(decaying the learning rate to 1×10^{-12} over 25,000 steps). This was then followed by training on the pig re-identification dataset outlined in Section 5.2.2 using transfer learning with the same hyper-parameters, with the exception of cosine decay, which was implemented over 9,000 steps due to the smaller size of the pig Re-ID dataset compared to MARS.

The small size of the model (2.8×10^6 trainable parameters) allowed it to be used alongside the distance metric whilst still allowing tracking to be executed in real-time. Smaller models are typically faster but are restricted in the amount of information they can extract from an image, which typically results in poorer performance. However, a model that is too large increases the likelihood of overfitting, so a balance is necessary.

5.3.2. Pig Tracklet Stitching

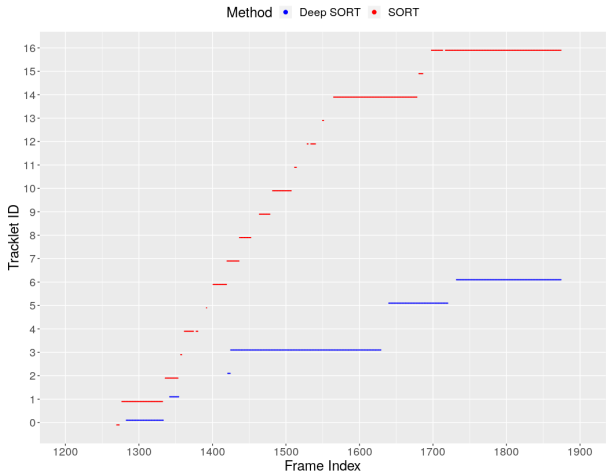


Figure 5.4 An example of a pig identity tracked by Deep SORT and SORT that is made up of several tracklets.

As Deep SORT uses similarity in visual appearance, as well as a distance metric, it was capable of reassigning a pig's original identity following an IDSW or missed detections at times where SORT could not (Figure 5.4). However, it was not capable of this when gaps in tracking occurred for an extended period of time.

To combat this, Deep SORT can be configured to allow more time between missed detections by increasing the number of reference images stored for each known identity. However, we found that doing so increases the number of IDSWs, as the more historic images that were stored for a given identity, the lower the intra-class cosine distance between pigs, which makes it difficult to distinguish between two pigs.

When a new identity is created by Deep SORT, it means one of two things: either a new pig has entered the field of view that has not previously been seen, or a pig that has already been seen has mistakenly been assigned a new identity. We, therefore, present a tracklet stitching approach that need only be ran when a new identity is identified (Algorithm 1). This method takes any new

identity and compares it to all identities that have been lost since the beginning of tracking. If a match is found where the cosine distance between two identities is below a threshold τ, the pig is reassigned its original identity. In addition to this, every n frames we compared the identities in the current frame with the identities being tracked n frames ago to assess whether there had been any identity switches. Both n and τ are optimised using grid search for each of the models.

```
L ← lost identities;
E ← existing identities;
τ ← similarity threshold;
while new frame do
    deepSORT();
    foreach id i in all identities active in current frame I do
        if i not in E then
            foreach lost identity l in L do
                if reid(l,i) < τ then
                    t ← l;
                    break;
                end
            end
        end
    end
end
```

Algorithm 1: Pseudocode showing how a model used for tracklet stitching is integrated with Deep SORT.

In order to compare identities, we trained and evaluated 9 state-of-the-art image classification and re-identification models to evaluate the visual similarity between them. As we were dealing with identity recovery over an arbitrary period of time, it was not possible to use a physical distance metric. As the method was only applied when a new identity was generated and every n frames, it did not hinder the ability for the overall tracking to run in real-time. Additionally, as this approach ran independent of the underlying tracking method, this approach could be applied to any existing tracking method in order to improve identity retention.

Re-Identification Models & Training

In total, 9 models were trained and evaluated for use as the tracklet stitching model (Table 5.1). Each model's feature extraction component was loaded with their respective pre-trained weights from training on ImageNet and then trained using the MARS dataset.

At this stage, the FC layers were further trained on the pig Re-ID dataset (Section 5.2.2) whilst all other layers were "frozen" (made untrainable); the final FC layer was reconfigured to have the number of classes required for the new dataset. This is done to prevent harmful changes introduced by the final, reconfigured FC layer propagating to the other pre-trained layers and also decreases training time [187, 225, 226]. This trained model could then be adapted for inference

Model	Size
Densely Connected CNNs (DenseNet-121) [216]	7×10^6
Inception v4 [217]	41×10^6
Multi-Level Factorisation Network (MLFN) [218]	32×10^6
Multi-Scale Deep Network (Mudeep) [219]	134×10^6
Omni-Scale Network (OSNet) [220]	2×10^6
ResNet-50 [221]	24×10^6
ResNet with Mid-level Features (ResNet-MidFeat) [222]	27×10^6
Squeeze-and-Excitation Network (S&E ResNet) [223]	27×10^6
Xception [224]	20×10^6

Table 5.1 The models used for reassigning newly generated identities with previous lost ones where the pigs are the same and their number of trainable parameters.tab

by removing the final FC layer and using the new final layer as a feature vector for the input image.

Re-Identification Model Evaluation

When evaluating the performance of Re-ID models, our primary metric was the mAP as it gives a whole overview of how the method performs and is therefore a better indicator of how well the method might generalise. As secondary metrics, we evaluate the CMC at ranks 1, 3 and 5. These better describe how the method is performing specifically regarding gallery images that are deemed the most similar to query images. A higher mAP means that, given a query image, gallery images with the same identity would rank the most similar across all gallery images. Whereas a higher CMC at rank N indicates how often a correct gallery image is in the top N most similar to the query image. These are standard metrics for Re-ID models [185].

In order to evaluate the efficacy of each model on improving the tracking results produced by Deep SORT, we compared the change in IDF1 [8] and the number of IDSWs (how many times two identities are incorrectly swapped) from before and after tracklet stitching was applied. Though MOTA is often used to evaluate tracking performance, it does not properly take into account how well an identity is maintained, looking only at the number of IDSWs; we did not use it for our evaluation. A reduction in IDSWs is something that should be worked towards, but only if this also results in an increase in IDF1.

5.4. Results

5.4.1. Re-Identification Results

As outlined in Section 5.3, each model was initialised with pre-trained weights from training on ImageNet, trained on MARS, and then trained on the pig Re-ID dataset. All models were trained with a starting learning rate of 1×10^6.

A cosine annealing [61] was used to decay the learning rate. "Warm-restarts" were not used, which periodically increase the learning rate in an attempt to avoid local optima, and model loss was calculated using cross-entropy loss. Initially, the learning rate was decayed by multiplying it by 0.1 every other epoch. However, using this method, most models were unable to exceed 50% accuracy before plateauing. Introducing a more gradual decrease in the learning rate, by means of the cosine annealing, was a substantial improvement in performance in terms of both accuracy and loss.

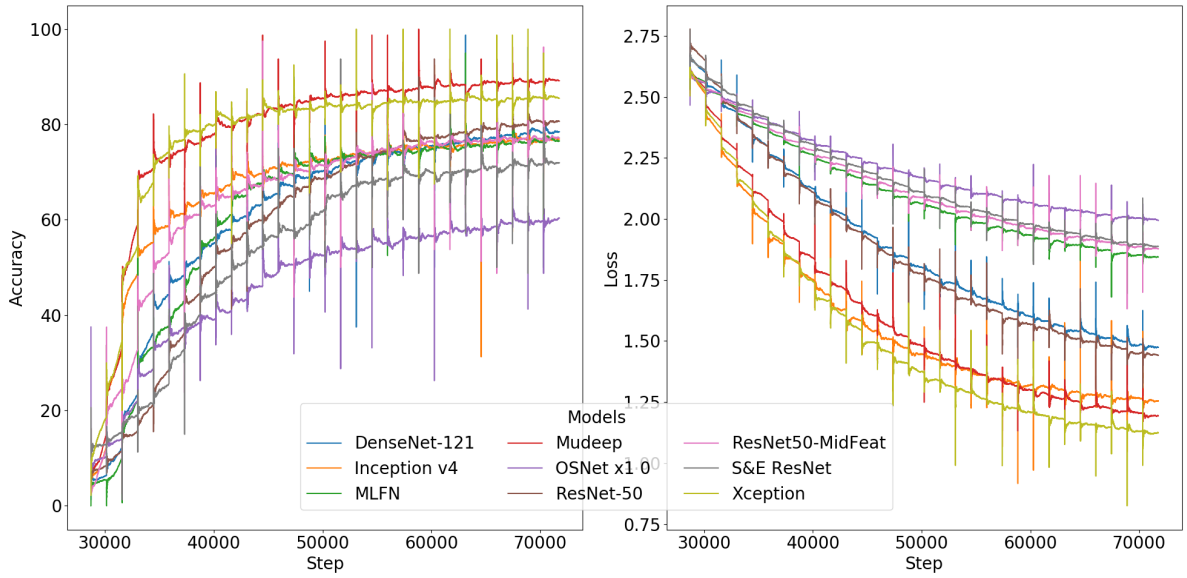


Figure 5.5 The training accuracy (left) and cross-entropy loss (right) for each of the evaluated models on the pig re-identification dataset with transfer learning from the MARS dataset.

Figure 5.5 shows the training loss and accuracy of each of the models used as part of our evaluation. The best model in terms of training accuracy, Mudeep, notably the biggest model evaluated, is not the same as the best model in terms of training loss, Xception. It is worth noting at this point that though accuracy is not the optimal metric to use when training data is not balanced, it is the simplest metric to compute for multi-class classification and, as we were only using this to evaluate training, and not testing, we felt it was appropriate to get a general idea of model training. Though accuracy trends upwards and loss trends downwards, they both do so with a notable amount of noise. This is a by-product of mini-batch gradient descent, which backpropagates loss after every mini-batch instead of at the end of an epoch as

with batch gradient descent. This is advantageous as noisier updating can help to avoid local minima, though comes at the cost of being more computationally expensive. For these models, we used a mini-batch size of 8, as this was the most we could fit on our GPU (Nvidia P100 16GB).

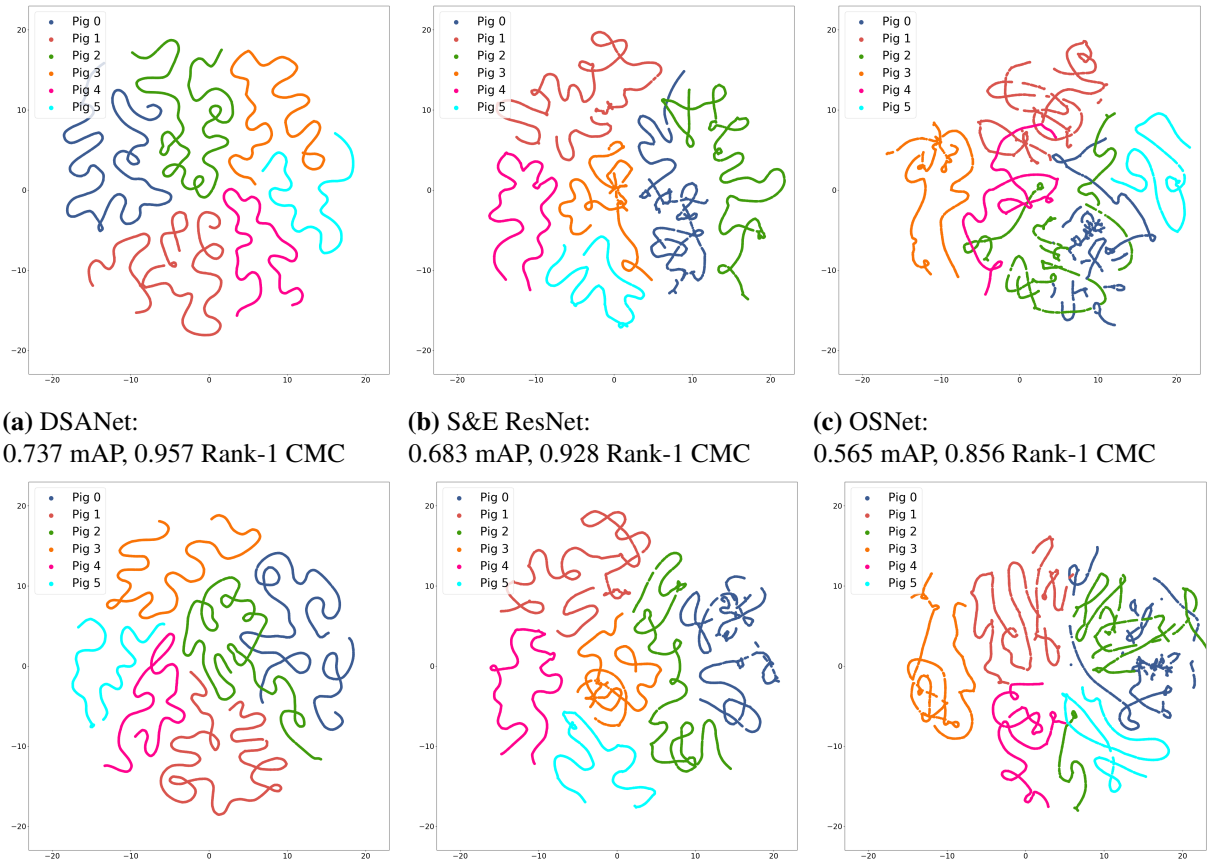
Model	mAP	Rank-1	Rank-3	Rank-5
Cosine Association Model	0.654	0.654	0.853	0.975
DenseNet-121	**0.737**	**0.957**	**0.984**	0.987
Inception v4	0.622	0.915	0.962	0.982
MLFN	0.720	0.947	0.977	0.983
Mudeep	0.606	0.942	0.972	0.977
OSNet	0.565	0.856	0.936	0.950
ResNet-50	0.723	0.948	0.973	0.984
ResNet-MidFeat	0.653	0.938	0.977	0.985
S&E ResNet	0.683	0.928	0.964	0.978
Xception	0.670	0.950	0.982	**0.992**

Table 5.2 The mAP and CMC at ranks 1, 3 and 5 of the nine re-identification models, trained on MARS and fine-tuned on our custom pig re-identification dataset, used for re-identifying pigs. We also include the cosine association model that is integrated as part of Deep SORT.

The performance of each model was clearer when they were evaluated on the Re-ID test data outline in Section 5.2.3. DenseNet-121 was the highest performing in terms of mAP and CMC at ranks 1 and 3 (Table 5.2). Despite a relatively average mAP, Xception achieved the second highest CMC at ranks 1 and 2, eventually outperforming DenseNet-121 at rank 5. This indicates that though Xception was able to perform well on the test set it was presented, it may be less capable of generalising to broader, more challenging datasets. Mudeep appears to be in a similar situation, scoring poorly in mAP, but well in CMC ranks.

When looking at the size of models used, there was no correlation between the size of a model and it's performance (both mAP and CMC). However, in extreme cases (i.e. Mudeep and OSNet), the model size appears to offer further detail to understand their performance. For example, Mudeep has nearly 10 times the number of trainable parameters than the other models, which, in line with its performance, suggests an inability to generalise to the test set. Similarly for OSNet, the model has over 10 times fewer parameters than the other models, which, for this particular application, suggests that it is too small to model the necessary features to carry out accurate pig Re-ID.

As part of the analysis of these results, we used Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [227] to reduce the dimensionality of the feature vectors produced by the Re-ID models of each image sequence to two dimensions. Figure 5.6a, b & c shows scatter plots of these two dimensions for three of the models: DenseNet-121, the best performing model; S&E ResNet, a mid-performing model; and OSNet, the poorest performing model, in terms of mAP and Rank-1 CMC. In these plots, the features appear to group into lines, where the plots of better performing models form longer lines than poorer



(a) DSANet:
0.737 mAP, 0.957 Rank-1 CMC

(b) S&E ResNet:
0.683 mAP, 0.928 Rank-1 CMC

(c) OSNet:
0.565 mAP, 0.856 Rank-1 CMC

(d) DenseNet:
0.737 mAP, 0.957 Rank-1 CMC

(e) S&E ResNet:
0.683 mAP, 0.928 Rank-1 CMC

(f) OSNet:
0.565 mAP, 0.856 Rank-1 CMC

Figure 5.6 UMAP scatter plots of the feature vectors of each image in the pig re-identification test set extracted from 3 differently performing models: DenseNet, the best performing model in terms of mAP and CMC at Rank-1; S&E ResNet, a mid-performing model; and OSNet, the poorest performing model. Plots **a**, **b**, and **c** use a different random seed to plots **d**, **e**, and **f**.

performing models. Unlike Principal Component Analysis (PCA), UMAP is non-deterministic, meaning that a different output vector is produced each time it is calculated despite the input being the same unless a random seed is set. We, therefore, produced duplicate versions of each of the UMAP plots (Figure 5.6d, e & f) and found that these groupings formed similarly in each. Substantial differences between the plots can be observed, however, we consistently see a correlation between the forming of lines within the scatter plot and model performance; the better a model performs, the more sequence-like groupings occur with an identities features.

5.4.2. Baseline Deep SORT Results

The cosine association metric learner was trained firstly on MARS, followed by our pig Re-ID dataset described in Section 5.2.2. After training on MARS the model achieved 0.661 mAP and 0.492, 0.775, and 0.933 CMC at ranks 1, 3 and 5 respectively on the pig Re-ID dataset. Following fine-tuning, this increased to 0.654 mAP, and 0.654, 0.853, and 0.975 CMC at ranks 1, 3 and 5 respectively. This network was subsequently integrated with Deep SORT and evaluated

on the tracking set described in Section 5.2.3 without any tracklet stitching. This resulted in a baseline of 0.841% IDF1 and 881 IDSWs. The tracklets produced by this method are the starting point for tracklet stitching that the Re-ID models are applied to.

5.4.3. Tracklet Stitching Results

The nine models that were trained were then applied to the tracks produced by Deep SORT as outlined in Section 5.3.2. Table 5.3 shows the results of applying each of the models to the task of tracklet stitching.

Model	IDF1↑	IDSWs↓
No Stitching	0.841	881
DenseNet-121	**0.846**	**869**
Inception v4	0.826	870
MLFN	0.808	**869**
Mudeep	0.826	870
OSNet	0.820	**869**
ResNet-50	0.817	**869**
ResNet-MidFeat	0.816	**869**
S&E ResNet	0.827	870
Xception	0.797	873

Table 5.3 The IDF1 (higher is better) and number of IDSWs (lower is better) of nine models used for stitching tracklets produced by Deep SORT along with the values associated with the original tracklets.

DenseNet-121 had the greatest effect on the base IDF1 and IDSWs produced by Deep SORT. This resulted in an increased average tracklet length from 28.44 seconds to 53.78 seconds, an 89% increase. Though they scored a lower IDF1 than the original output from Deep SORT, MLFN, OSNet, ResNet-50, and ResNet-MidFeat also reduced the number of IDSWs to the same level as DenseNet-121. This indicated that the number of frames tracked under a single identity was shorter when using these models.

5.5. Discussion

When compared to the high-quality data typically used in crowd analysis and vehicle tracking, there are additional challenges to detecting and tracking pigs on commercial farms. This is mainly caused by low FPS footage and poor image quality. The data itself is also more challenging as pigs are often densely packed into a small area and look very similar in appearance. However, as understanding the behaviour of pigs can help us to provide a more individually tailored level of care and thus improve welfare, research into how multi-object tracking can be applied to individual pig tracking on commercial farms in order to provide individual pig care is valuable.

Previous research has focussed on applications of multi-object tracking systems using both depth and colour cameras. In most cases, the MOTA is reported, often alongside multi-object tracker precision (MOTP) (a metric for measuring localisation rather than tracking), and good tracking results are reported. However, MOTA fails to properly quantify how well identities are maintained throughout tracking as it assesses the number of times an identity changes, rather than how long a tracker correctly identifies an object. MOTA is a measure of how well objects are tracked, but it does not accurately reflect how well identities are maintained.

In multi-object tracking, reliably maintaining an identity, is not always essential. Therefore one might choose to prioritise MOTA over IDF1. This is particularly the case if the aim is to gather group-level statistics. Previous research focussing on individual-level tracking [37], where tracks were able to be maintained for around 20 minutes, rely entirely on the notion that it is possible to re-identify pigs by eye. However, this is not always feasible due to how similar they often look. Therefore, in the case of individual pig tracking, maintaining the identity of a pig over time is the most important goal.

Typically, research in how multi-object tracking can be applied to pigs focusses directly on how the tracker can be improved [181, 182, 101, 228, 229, 37, 213], but this comes with a tradeoff. Generally speaking, improving re-identification capability either comes from an additional pre-processing step [101] or by improving the model that computes visual similarity (i.e. increasing model capacity) [182]. As both of these approaches are part of the pipeline of the tracker, they inherently increase the runtime of the method, meaning they reduce the FPS the tracker can process. For this reason, this research has focussed on how pigs can be automatically re-identified to "stitch" tracklets that are produced by real-time trackers that belong to the same ground truth identity. This way, re-identification runs in parallel to the tracking method and thus does not affect the performance of the tracker in terms of maximum FPS that can be processed. Therefore, the main contribution of this work is improving the performance of existing multi-object trackers, without the need for modifications to the tracking, additional hardware or physical markings on the pigs.

The models used were trained on a relatively small dataset consisting of only 20 identities. This added an additional challenge to the training of the models. In the task of Re-ID, it is desirable to have a large number of identities to train on so that the penultimate layer in the CNN, that acts as a Re-ID feature vector for an image, generalises well. We attribute the ability of the Re-ID models to generalise beyond the training data, despite the small number of identities, substantially to a large number of identities in the MARS dataset (625) that the models were trained on prior to our custom pig Re-ID dataset which was used for fine-tuning.

Additionally, as all of the methods we have implemented are CNNs, temporality is not explicitly programmed into the architecture as it is in recurrent neural networks. Instead, the image that is input to the model is 10 images concatenated. It is expected that the models will

learn the concept of temporality implicitly rather than through making it explicit as part of the architecture.

Of the models we trained on our pig Re-ID datasets and evaluated on our individual pig tracking dataset, the best model, DenseNet-121 (0.737 mAP, 0.957 rank 1 CMC), improved the IDF1 score from 0.841 to 0.846. MLFN, OSNet, ResNet-50, and ResNet-MidFeat achieved the same reduction in IDSWs as DenseNet-121, however, this came at the cost of a lower IDF1. A simple example of how this is possible is outlined in the previous chapter (Section 4.4).

Where other research focussed on metrics that favour reducing the number of negative events, we have made use of a more appropriate metric that more accurately evaluates a model based on its ability to maintain the correct identity of pigs as they are tracked within a pen on a commercial farm. We have used this metric to evaluate a real-time tracking method that uses visual appearance to improve identity retention. We then also used it to measure how implementing a re-identification model, running in parallel to the tracker, can improve the performance of the algorithm. The advantage of this approach is the ability to get this performance increase without requiring changes to the existing algorithm, affecting its runtime or requiring any additional hardware.

Regarding the limitations of this work, we acknowledge that the improvement gained in terms of IDF1 is relatively small. However, the improvement achieved by the DenseNet-121 model, in terms of IDF1, was on a test set substantially more challenging than the data it had been trained upon. Specifically, the MARS dataset consists of humans, where identities are typically much easier to tell apart, and both MARS and our own pig Re-ID dataset consisted of images of a high-resolution (1920×1080), where the test set was of a very lower resolution (640×360). Additionally, despite there only being 20 pigs in the pen used for collecting the final test data, 74 identities were created, as the data collection was carried out using a single camera that covered only half of the pen. If a pig left the field of view of the camera and then re-entered, it was assigned a new identity. This meant that it would be possible for the re-identification model to correctly determine that two images were of the same pig, but for the ground-truth data to count it as incorrect, as pigs have multiple identities. Despite this, our implementation is still able to achieve a CMC at Rank 1 of 0.957.

In retrospect, not allowing multiple identities for each pig to be created will give a much more accurate, and likely favourable, picture of the models' performance. Also, it would be more beneficial to improve the pig Re-ID dataset by increasing the number of identities at the cost of reducing the number of images per identity (i.e. it would be better to have 20 images of 885 identities, rather than the 885 images of 20 identities that made up our dataset).

5.6. Conclusion

We have implemented and evaluated multiple methods for pig re-identification in a commercial farm setting that can be used for stitching together tracklets produced by a multi-object tracker in parallel without affecting runtime even when the images are of very low resolution and the pigs are indistinguishable by eye. Despite the obstacles created by a challenging dataset, we were able to increase the IDF1 and reduce the number of IDSW of an existing tracker, that already contained a visual similarity component. Overall, this work demonstrates that without the need for additional hardware, preprocessing or increased runtimes, we can improve existing pig trackers based entirely on their visual appearance. In doing so, we can extract more accurate behavioural metrics on an individual level, which can give us a better understanding of a pigs health and wellbeing, without requiring a substantial amount of additional manpower.

Chapter 6. General Discussion

In this chapter, we re-visit the overarching aims of this thesis that were set out in Chapter 1, and outline the main research contributions that were developed throughout our core research chapters in pursuit of them. In addition to this, we have highlighted the limitations of these contributions, particularly relating to how current hardware impacts real-world adoption and implementation. We specify some of the specific challenges regarding the use of deep learning in agriculture, along with recommendations of how to best deploy our implementations in operating commercial farms. Finally, we discuss the importance of metric selection when implementing deep learning models, particularly in relation to the models described in this thesis, and present areas for further research that this thesis has laid the groundwork for.

6.1. Research Contributions

In the introductory chapter of this thesis (Chapter 1), we outlined what both PLF and deep learning are, and introduced some of the most recent and influential literature along with identifying the challenges that are faced both by PLF methods and specific deep learning approaches. Until recently, research in PLF has predominantly focussed on group-based metrics for livestock, as monitoring metrics on an individual level comes with extensive challenges. Additionally, there have been few implementations that exploit the most recent developments in the machine learning space. Therefore, this thesis set out to show that deep learning methodologies can be used to improve upon the approaches that have typically been relied upon. Our research has shown how the advancements in computer vision can be used to greatly enrich the data farmers can collect about individual animals, and how various deep learning architectures can be used to process multiple environmental sensors for early warning of oncoming disease.

Despite a wealth of research regarding applications of deep learning for computer vision challenges, we found a gap in the literature pertaining to the use of deep learning methodologies for disease forecasting in livestock. Though these methods had been applied to human disease prediction, though often using univariate data, we explained why this was not enough to assert that the approaches were directly transferable to livestock applications. We found that the current commonly implemented methods for agricultural applications were typically SVMs, ARIMA and a variety of random forest-based architectures. Therefore, we dedicated part of this thesis to identifying a robust solution for modelling the relationship between environmental conditions

and respiratory disease prevalence that could act as an early warning system. In addition, we sought solutions that used inexpensive hardware and required no additional modification beyond the initial set up.

Through our research presented in Chapters 2 and 3, we empirically demonstrated that there is enough information within the environmental data alone to make good estimates about respiratory disease prevalence and that deep learning methods are well suited to modelling this relationship. Our implementation used much higher resolution data than the existing literature in similar applications [230, 106], without the need for preprocessing, demonstrating that they are capable of processing raw, high-throughput data. Additionally, we showed how reframing the problem as an anomaly detection task, rather than a forecasting task, improved performance in RNN-based models, as it allows the model to benefit from the wealth of "normal" data. Finally, we showed that although CNN architectures do not account for the temporality of data explicitly in their architecture, they are still capable of learning this relationship through training, and achieved similar performance to RNN-based approaches but with substantially reduced resources. It would not be reasonable to expect a system that looks solely at environmental factors to be able to achieve near-perfect performance, as there are many other factors that affect disease prevalence [155]. Despite this, our models still achieve strong performance in detecting the environmental conditions that are likely to cause an increase in respiratory disease prevalence.

The inherent temporal nature of the RNN architecture makes a strong argument for using them to process time-series data, however, the substantially reduced resource requirements of both CNNs and transformers makes them very attractive. The research presented in Chapter 3 showed that these non-recurrent architectures are capable of maintaining the performance of their recurrent counterparts when modelling multivariate time-series data despite their reduced resource requirements and no explicit concept of temporality. These results, combined with the overwhelming popularity of both CNNs and transformers in time-series processing, are a strong indicator that, in some applications, RNNs can be replaced with these newer architectures [168]. The temporal characteristics built into the RNN should not be enough to make them the default model of choice for temporal data. This sentiment is generally established for applications processing univariate temporal data [167, 168], but our research shows that this can also be the case for multivariate temporal data.

Though applications of object detection in PLF, including classification, have been researched to some extent for both crops and livestock, there is limited literature addressing the long-term tracking of individual animals in video. Instead, projects that depend upon the monitoring of animals use human observation to note specific activities and behaviours. Putting the flaws with this approach aside (addressed in Section 1.1), human observation can work for capturing data for research, but it does not scale up for the purpose of monitoring a commercial farm, as there is a drastic difference in scale.

From the existing literature, we identified that though RFID tags had shown promising results in tracking the movements of pigs between designated "hotspots", it was only enough to provide a rough estimate of activity (e.g. pig 1 located in area A at time X then located in area B at time Y and so on). This meant that inferring further information from these measurements would be challenging, as large assumptions would have to be made about the pig's activity between these hotspots. Furthermore, there are also challenges for this hardware when there are multiple identities to detect. In addition to the performance issues, the need for RFID tags to be regularly installed and removed, along with configuring RFID antennas, is generally undesirable for real-world implementations, as it requires a change in the existing workflow for farmers, therefore adding friction to adoption. Additionally, the utilisation of this technology is not considered cost-effective within the farming industry. Based on these findings, our research sought a solution that could be implemented into commercial farms and that would not require any hardware beyond the initial installation. Any hardware that was required also had to be inexpensive to further reduce the barrier to entry for commercial farms.

The research presented in Chapters 4 and 5 were combined and ran in parallel to form a workflow for the long-term tracking of pigs using deep learning methodologies in all components of the system: detection, tracking, and re-identification. Our research has shown that multiple stages of training on benchmark datasets and fine-tuning on our custom-built pig detection dataset enables a Faster R-CNN architecture to detect pigs in images with strong performance, and that the visual appearance of a pig can be used to enhance the tracking of them between frames. This was achieved even in extremely challenging scenarios, such as times where pigs were densely packed together and across extreme variations in light intensity, both of which are common, challenging conditions for CNNs.

In addition to the detection and tracking, numerous CNN architectures were evaluated to determine their performance in pig re-identification and demonstrated that the best performing model, DenseNet, could be used to improve tracking performance. This re-identification model was implemented in such a way that it could be used alongside any tracking mechanism, as it was designed to run in parallel to an existing tracker so as to not impact performance in terms of FPS. This combination of tracking and Re-ID architectures created a foundation for tracking the behaviours of pigs as individual animals over prolonged periods of time, rather than as an entire batch.

The design of the tracking system, accounted for the generalisation capacity of our models, considering the different farms, or animals, this approach could be implemented in or applied to. Though we do not validate this in our research, by choosing the Faster R-CNN architecture over YOLO, the system could be installed in taller buildings or be used on smaller animals with limited effect on detection rates, as Faster R-CNN performs better on smaller objects. Additionally, the detector could be trained to detect any number of species in a single model, allowing a single model to be applied to various animals. However, regarding the re-identification and association metric components, variations in object size would likely impact performance unless a higher

resolution camera was used, as these were dependent on the finer details in images. Nonetheless, we would expect these models to also be able to re-identify multiple animals within a single model.

The performance of DenseNet in pig Re-ID demonstrated the impressive capability of CNN-based models. Despite the fact that the pigs used in our dataset were almost entirely indistinguishable from one another by eye, it achieved both a strong mAP and CMC at rank 1. Multi-object tracking is most commonly applied to humans (e.g. crowd analysis) and vehicles (e.g. self-driving cars) [205]. In both of these applications, an object's trajectory at any given time is typically a strong indicator of their future location. However, our research shows that the Re-ID workflow, typically reserved for multi-camera Re-ID, can go beyond this. The approach we presented validates that it is also well suited to the task of tracklet stitching, not only to recover from long-term occlusions, but also when the object movement is chaotic and objects are extremely similar in appearance.

6.2. Research Limitations

As with many interdisciplinary research projects, striking a balance between the various sides was not straightforward. In the case of this thesis, which was at the intersection of computer science and agriculture, it meant both sides needed to accept trade-offs. Our research focussed not only on introducing novel approaches to the challenges in PLF, but also on providing accessible solutions, by taking into account how the methods we presented could be deployed into operating, commercial farms. This created a drive to reduce computational resources where possible and ensure that any hardware that was necessary for implementation was affordable.

The main limitation of the work presented across Chapters 2 and 3 was that only environmental data from within the housing unit was used to train the model to understand "normality". Values for temperature, humidity and CO_2 only describe a narrow view of farm conditions. Factors beyond environmental conditions are known to impact disease prevalence, such as herd size, herd density, vaccination usage, weaning age and whether a farm has facilities to incinerate deceased pigs [175]. Providing these values as training data, alongside the environmental conditions within the housing unit, would likely improve the performance of the model, as it provides a more holistic view of the farm conditions.

A further limitation of the early-warning system for oncoming respiratory disease relates to the generalisability of the overall approach to pigs at different stages of growth. The data used to train the Seq-U-Net was captured from growing pigs, resulting in the model being limited to the ideal conditions for growing pigs specifically. Weaning pigs, which are younger and smaller than growing pigs, are much more susceptible to disease due to their immune system being less developed [231]. Therefore, a model that is capable of achieving similar results for pigs at this stage would be valuable. Although the general approach presented in Chapter 3 is likely capable

of modelling similar data captured from weaning pigs, in its current configuration, it would not be able to account for both growing and weaning pigs simultaneously. This is largely due to weaning pigs having different environmental needs and a lower tolerance to changes in their environmental conditions. However, accounting for the age of pigs alongside the other inputs would be a potential remedy to this challenge. It would enable the model to learn that the concept of normality changes depending upon the age of the pigs and would avoid the need for training different models for pigs of different ages.

Typically, farms are situated in remote locations where access to high-speed network and other infrastructure is severely limited, putting constraints on the types of solutions that can be used. Our experience regarding the collection of data throughout this research project made it clear that until substantial improvements are made to the network infrastructure available in remote locations, there will always be limitations on what can be achieved. For example, carrying out data collection on-site and streaming it to a cloud-based service for processing would not be workable as the network is often too unreliable.

As the computing power of single-board devices improves, such as Nvidia's Jetson devices [176] and Google's Coral devices [177], implementing "edge compute" solutions for deep learning methods becomes more feasible. Using this approach, solutions are deployed onto a device located on-site. This device both captures and processes the raw data, meaning only the relevant outputs need to be disseminated to external services. In the case of computer vision applications, this is typically several orders of magnitude less data, reducing the impact of poor infrastructure. These devices are still in their infancy, but significant improvements have been made in recent years to make "embedded machine learning" a viable method of deployment [176], and provides a real alternative to cloud-based solutions, particularly for remote locations. Currently, these devices have very limited memory, which was why our research to reduce the resource requirement of our work in Chapter 2 was so valuable, as the CNN-based approach (Section 3.2.2) reduced the memory requirement by almost half. This made the overall solution more feasible to be used in an edge-based deployment and memory is usually scarce.

The hardware decisions that needed to be made regarding our capturing of the environmental data were relatively simple. The necessary sensors are inexpensive and commercial, off-the-shelf units exist specifically for deployment in farms [134]. However, in almost all applications of computer vision, there is a need for the use of cameras and choosing the right one has a substantial impact on what can be achieved.

For our work in Chapters 4 and 5 we used the Microsoft Kinect v2 for image capturing (only the RGB sensor was used meaning any high-resolution camera could be used as a substitute). As discussed in Section 1.1, both RGB and depth sensors have been used in existing literature for applications relating to pig management, including activity monitoring, and both come with the respective benefits and drawbacks. One of the major downsides of using an RGB sensor is that they cannot be used at night. This is a particular issue in an agricultural setting as, by

law, all animals must be provided with an appropriate day/night cycle [232]. Although the work in this thesis achieved surprisingly good performance in very low-light conditions for pig detection (Section 4.3.1), it was not expected that this would be enough for carrying out reliable multi-object tracking at night. Depth sensors, on the other hand, can be used to overcome this, as they use infrared light, though they are not without drawbacks.

Our research showed that using the visual appearance of pigs can be used to enhance the performance of multi-pig tracking. Though research has been carried out with relation to pig tracking in 3D depth images [35], for our particular application, it would inhibit the performance of the tracking algorithm as there would be no access to colour data making it very difficult to have individual-level tracking. Research has begun to address the challenges of human re-identification using depth sensors, often focussing on applications where there is low lighting or the specific use case requires the capability to re-identify people after they change clothes [233]. However, this research is still very much tied to human applications, as approaches often rely upon estimating a skeleton for the person and using that to determine an identity. This is dependent on viewing the target human from a specific angle. An additional drawback of depth sensors is that their accuracy typically degrades the further away an object is. This can create challenges if a system needs to be installed in a building with a high ceiling or if large areas needs to be covered by the sensor's fields of view. Furthermore, depth sensors are typically more expensive than RGB, especially if they need to have an accurate, large operating range.

One of the benefits of RGB sensors over depth is that there was no need for any kind of sensor calibration, whereas this is typically a common requirement when using depth sensors. Through the research presented in Chapter 5, we showed that our detection and re-identification models trained on data captured on the Kinect v2 were able to detect, track, and re-identify pigs in images captured from a different RGB sensor that was recording at a resolution that was 3 times lower. These two sensors were installed in different locations and no explicit effort was made to ensure they were installed in a similar configuration (e.g. angle to the floor). The only consideration that was made was to ensure that they covered their designated area of the pen. This speaks for the generalisability of our implementation, meaning that it is possible for our models to be deployed on another farm without requiring any additional training or calibration to the new location.

6.3. Challenges to Deep Learning Approaches

One of the biggest criticisms of deep learning methodologies, with respect to classical statistical approaches and even traditional machine learning, is the sheer amount of data that is necessary for them to achieve their performance gains. This was part of the reason that the work we present in Chapters 2 and 3 treated our early-warning system as an anomaly detection problem, as a substantial majority of our data was "normal" data, so it made sense to make the most of this

situation. However, knowing which parts of the data were normal required extensive human annotation over a long period of time across multiple farms throughout Europe, which was only achievable due to the scale of the project. In the case of the image data used for this research, pre-training the part of the network responsible for extracting feature maps (often referred to as the "backbone") is a common step taken to reduce the need for large amounts of domain-specific training data.

In our work presented in Chapter 4, the ResNet backbone used in our object detection model was initialised with weights that were pre-trained on the ImageNet dataset. It is common to initialise with pre-trained weights, rather than manually pre-training, as the ImageNet dataset is extremely large and tuning a model to this dataset is a task in and of itself. This pre-training meant that this portion of the model had already been optimised to extract feature maps from images that could be used to classify it. Once the region proposal network and fully connected layers were added to this backbone, the model was then pre-trained on an object detection benchmark dataset (VOC) and subsequently on our own domain-specific object (pig) detection dataset. Without the ImageNet dataset for pre-training, our domain-specific dataset of around 1,000 images would not have been enough for accurate object detection. The amount of data that would need to be annotated to achieve this without ImageNet would have likely made our approach infeasible.

Even with an abundance of pre-training data, there was still a need for domain-specific data annotation, particularly for our research in re-identification, as we used a supervised training approach. Though more recent research has started to explore using similar architectures for unsupervised re-identification of objects, the decision to use a supervised training approach in this thesis was driven by the fact that our data was particularly challenging. This is particularly the case when compared to person re-identification as is often covered in the literature [234–237]. Before research can address an unsupervised approach for this application, it was necessary to understand the performance of a supervised approach. This gives an indication of the capability of a CNN-based approach for pig re-identification and sets a baseline for what can be achieved these architectures.

Another criticism of deep learning-based solutions is substantially longer training times, which is often a by-product of the quantity of data used to train the models. This issue is particularly noticeable in Chapter 2 of this thesis, where we compared deep learning methods with a classical statistical approach, ARIMA. In Section 2.5.3, our results showed that GRU models took several days, whereas ARIMA was trained within 24 hours. Looking at the performance improvement of the GRU-AE model over ARIMA, an argument could be made that it is not worth the additional training time. However, in the same way that our pig re-identification chapter laid the groundwork for future unsupervised approaches to this problem, the GRU-AE model laid the groundwork for exploring other autoencoder-based architectures that could remedy this flaw. In Chapter 3, we demonstrated that a similar approach using a

CNN could reduce the training time to a fraction of that of ARIMA with minimal effect on the performance of the model.

As outlined in the introductory chapter of this thesis, selecting the correct metric to evaluate a deep learning model is equally as important as selecting the correct architecture. Throughout this thesis, multiple decisions were made regarding the choice of metric. In Chapters 2 and 3 the decision was made to use precision and recall, as is typical in these cases, but rather than using F_1 we opted for MCC. This metric meant that the classifier had to correctly classify both positive and negative cases in order to achieve a good MCC, making it more suitable for our imbalanced dataset [69]. In Chapters 4 and 5, rather than using solely MOTA, we also reported IDF1. Where MOTA quantifies how well multiple objects are tracked, IDF1 quantifies how well multiple identities are tracked. In both the case of MCC and IDF1, our research has found them to be relatively under-utilised metrics in the existing literature. In the case of IDF1, this may be because identity tracking is not always pertinent to the particular application being evaluated. Nonetheless, it should be reported regardless, as it much more accurately represents the performance of multi-object tracking systems.

The development of deep learning frameworks such as PyTorch [238] and TensorFlow [239] have made implementing deep learning models substantially faster than using array libraries such as NumPy [240]. All of the work for this thesis was implemented using PyTorch. This decision was made at the time as PyTorch utilised a dynamic computation graph, as opposed to a static computation graph that was used by TensorFlow. At the cost of performance, a dynamic computation graph allows for an easier development cycle as they are easier to debug. Additionally, it simplified the process for allowing our RNN-based architectures to accept variable-length input sequences, as the graph is created for the data it is receiving at runtime, rather than requiring it to be specified beforehand. Since making that decision, both frameworks now support both dynamic and static computation graphs. There are also a plethora of additional libraries than can be used to improve experiment reproducibility [241] and to simplify the development process by helping with the tracking of experiments, their parameters, and their results [242].

6.4. Real-World Deployment

One of the main aims of this thesis was to account for how the work we have presented could be used in an operational, commercial farm. From a data perspective, we ensured that all the data we collected for model training and evaluation was obtained from working farms in order to ensure that we had to deal with as many of the challenges that a real-world implementation would face. Additionally, in order to demonstrate that our solutions could be implemented on any farm, we evaluated both the early-warning system from Chapters 2 and 3 and the pig tracking system from Chapters 4 and 5 under varying conditions. The early-warning system was evaluated on data

from farms across Europe in substantially different climates (Belgium, Cyprus and Spain) and the pig tracking system was evaluated on data from two different cameras operating at different resolutions, covering different locations.

With respect to the installation of hardware for the Seq-U-Net early warning system, temperature, humidity and CO_2 sensors would need to be installed and linked to a compute device. Despite training and evaluating the models on GPUs, a CPU would be sufficient for the deployment of the system, as inference is only done on a minutely basis. For the pig tracking system, our research only covered implementations with a single camera. Therefore, in an ideal situation, a single camera would cover the entire pen area, as having only a proportion of the pen covered introduces substantial challenges to re-identification of pigs; mostly in situations where the pigs leave and subsequently re-enter the field of view of the camera. In addition to the camera, a compute device equipped with a GPU would be necessary for this implementation, as it is not currently possible to do CPU-based object detection at a good enough frame rate that allows for reliable tracking of objects between frames. Both Deep SORT and the DenseNet re-identification model could be run on a CPU rather than GPU. However, for the latter of the two, this would depend upon the conditions at a given farm (e.g. lighting, camera angle, etc.). In poorer conditions, it would be expected that longer periods of lost detections would occur, which impacts how often the re-identification model would be needed.

Despite RGB sensors not requiring specific calibration or configuration, a proportion of the flaws in tracking the performance can likely be attributed to the angle at which the cameras were installed. In all of the camera installations that were used to create our own datasets, two cameras were installed next to each other on the ceiling, in the centre of the pen. In order to have coverage of the entire pen, the cameras were angled outwards, rather than pointing directly at the floor, meaning that it was more likely that a pig on the outskirts of the pen would be occluded by another pig next to it. This would undoubtedly negatively impact the performance of detection, and therefore also tracking. Although the re-identification model was introduced to recover from these situations, it is preferable to avoid these situations where possible. In a real-world deployment, we would recommend that cameras be installed so that the sensor is positioned perpendicular to the ground and, in order to still have good pen coverage, they should be mounted in the centre of the area the sensor is expected to cover, rather than the centre of the pen.

In all of the implementations proposed in this thesis, outside of the initial installation, no changes in day-to-day activity on the farm would be necessary. Where other methods might require the spraying of identifiers onto the backs of pigs, or the installation and removal of identity tags, our approaches adopt a "set-and-forget" approach. After the hardware is installed, an application could be deployment so that farmers receive notifications regarding potential problems to prompt them to investigate. This, combined with their own expertise, would minimise the amount of time needed to spend observing animals, allowing them to focus on more challenging tasks.

6.5. Future Research

Through the research presented in this thesis, we have created the potential for additional research that could build upon our findings and implementations to further utilise technology to make the adoption of PLF concepts easier for commercial farms. One such example of this could be using the early-warning system and pig tracking system in conjunction with one another. Implementing these systems alongside one another would give broader coverage of many of the variables that can alert to the signs of disease. Where the early-warning system could provide a general alert to poor environmental conditions for a batch of pigs, the tracking system could be used to identify any specific pigs that need intervention. Furthermore, our specific approach to anomaly detection in time-series data could be applied directly to the activity levels of each individual pigs. Where the training data would be a sequence of coordinates that represents "normal" activity levels.

As discussed in Section 6.2, a major factor holding back the performance of the environmental sensor anomaly detection is that there are other factors that contribute to the respiratory disease prevalence. Therefore, in addition to extending the anomaly detection model to other species, future research may consider including additional inputs to the model to improve coverage of the various factors that contribute to respiratory disease prevalence. Previous research has indicated that changes in food and water consumption is a strong indicator of a change in an animal's condition [23, 24], therefore incorporating this data, as well as the variables identified in Section 6.2, into the anomaly detection model would likely improve the overall early-warning system. This data can be recorded using flow rate sensors that can be calibrated to calculate the total amount of water consumed by animals for a given batch. It can be integrated into the model by adding an additional dimension to the input tensor, all other training and inference stages would remain as described.

The work presented throughout this thesis has been implemented such that it is species agnostic, meaning that minimal changes to the implementation would be necessary for it to be applied to animals other than pigs. Object detection architectures, such as the Faster R-CNN detector used in Chapters 4 and 5, are often proposed alongside performance metrics for the benchmark datasets such as VOC. These typically contain a range of objects such as household items, vehicles and various animals. Therefore, future work may look at collecting a multi-species dataset to create a single detection and tracking workflow that could be applied to various species. Variations in performance should be expected, as animals that look similar to one another would likely result in poorer performance in re-identification. However, understanding the limits of existing re-identification approaches, such as which species are difficult to re-identify and why, is a step towards developing improved architectures.

As our tracking system currently only supports a single camera, extending the algorithm to enable re-identification of animals between multiple cameras would be a major improvement to the system, allowing for tracking of animals in a much larger area and potentially even outside.

To achieve this using a supervised training approach, multiple images of each identity from various cameras would need to be collected. However, unsupervised methods may prove to be a better option, as they can improve the generalisability of the model to different settings as discussed in Section 6.3; their ability to perform on visually similar objects has not yet been fully determined.

6.6. Conclusion

In pursuit of the aims of this thesis, we have shown that deep learning has a lot to offer the agricultural industry regarding the monitoring and modelling of individual animals. In this thesis, we have proposed a deep learning, autoencoder-based approach to anomaly detection in multivariate time-series. This architecture was able to detect when environmental conditions are likely to result in an increase in respiratory disease prevalence in pigs. Our implementation generalises across multiple climates across Europe and does not require any pre-processing of sensor data, such as handcrafted features or time-series decomposition. Additionally, we have presented a full workflow for the detection and tracking of individual pigs in an operating, commercial farm that did not require any hardware beyond a standard RGB camera. This used a pig's appearance, in conjunction with its trajectory, to re-identify it after short-term interruptions to tracking A separate model was used to recover from long-term tracking interruptions, which only used a pig's appearance to associate it with an existing identity. This individual pig tracker was used to extract activity metrics for each pig, which could then be used by farmers to understand the health and welfare conditions of each pig. Despite our research focussing specifically on applications in pig farming, it is clear that the methods we have implemented and evaluated have applications to other species and more broadly within the field. Although there is often a great deal of unwarranted "hype" surrounding deep learning algorithms, the work presented in this thesis demonstrates that there is justified reason to be excited for the future of deep learning in Precision Livestock Farming.

References

[1] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[2] Christopher Olah. Understanding lstm networks, 2015.

[3] Fjodor Van Veen. *A Mostly complete chart of Neural Networks*. The Asimov Institute, 2016.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[5] Daniel Stoller, Mi Tian, Sebastian Ewert, and Simon Dixon. Seq-u-net: A one-dimensional causal u-net for efficient sequence modelling. *arXiv preprint arXiv:1911.06393*, 2019.

[6] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[7] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2129–2132, 2019.

[8] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[9] Hannah Ritchie Max Roser and Esteban Ortiz-Ospina. World population growth. *Our World in Data*, 2013. https://ourworldindata.org/world-population-growth.

[10] Hannah Ritchie. Meat and dairy production. *Our World in Data*, 2017. https://ourworldindata.org/meat-production.

[11] Thomas P Van Boeckel, Charles Brower, Marius Gilbert, Bryan T Grenfell, Simon A Levin, Timothy P Robinson, Aude Teillant, and Ramanan Laxminarayan. Global trends in antimicrobial use in food animals. *Proceedings of the National Academy of Sciences*, 112(18):5649–5654, 2015.

[12] European Surveillance of Veterinary Antimicrobial Consumption European Medicines Agency. Sales of veterinary antimicrobial agents in 30 european countries in 2015. *178p*, 2017.

[13] Christy Manyi-Loh, Sampson Mamphweli, Edson Meyer, and Anthony Okoh. Antibiotic use in agriculture and its consequential resistance in environmental sources: potential public health implications. *Molecules*, 23(4):795, 2018.

[14] Daniel Berckmans. Automatic on-line monitoring of animals by precision livestock farming. *Livestock production and society*, 287, 2006.

[15] Beth Clark, Gavin B Stewart, Luca A Panzone, I Kyriazakis, and Lynn J Frewer. A systematic review of public attitudes, perceptions and behaviours towards production diseases associated with farm animal welfare. *Journal of Agricultural and Environmental Ethics*, 29(3):455–478, 2016.

[16] Konstantinos G Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018.

[17] International Society for Precision Agriculture. Precision ag definition, 2019.

[18] Annelies Michiels, Sofie Piepers, T Ulens, N Van Ransbeeck, R Del Pozo Sacristán, Annelies Sierens, Freddy Haesebrouck, P Demeyer, and Dominiek Maes. Impact of particulate matter and ammonia on average daily weight gain, mortality and lung lesions in pigs. *Preventive veterinary medicine*, 121(1):99–107, 2015.

[19] Ilias Kyriazakis. Pathogen-induced anorexia: a herbivore strategy or an unavoidable consequence of infection? *Animal Production Science*, 54(9):1190–1197, 2014.

[20] ST Ahmed, H-S Mun, H Yoe, and C-J Yang. Monitoring of behavior using a video-recording system for recognition of salmonella infection in experimentally infected growing pigs. *Animal*, 9(1):115–121, 2015.

[21] Katharina DC Stärk. Epidemiological investigation of the influence of environmental risk factors on respiratory diseases in swine—a literature review. *The Veterinary Journal*, 159(1):37–56, 2000.

[22] Qingyun Li and John F Patience. Factors involved in the regulation of feed and energy intake of pigs. *Animal Feed Science and Technology*, 233:22–33, 2017.

[23] Thomas Nejsum Madsen and Anders Ringgaard Kristensen. A model for monitoring the condition of young pigs by their drinking behaviour. *Computers and electronics in agriculture*, 48(2):138–154, 2005.

[24] HM-L Andersen, Lise Dybkjær, and Mette S Herskin. Growing pigs' drinking behaviour: number of visits, duration, water intake and diurnal variation. *Animal*, 8(11):1881–1888, 2014.

[25] Janeen L Salak-Johnson, Deirdre L Anderson, and John J McGlone. Differential dose effects of central crf and effects of crf astressin on pig behavior. *Physiology & behavior*, 83(1):143–150, 2004.

[26] Jeffery Escobar, William G Van Alstine, David H Baker, and Rodney W Johnson. Behaviour of pigs with viral and bacterial pneumonia. *Applied Animal Behaviour Science*, 105(1-3):42–50, 2007.

[27] Marian Stamp Dawkins. *Observing animal behaviour: design and analysis of quantitative data*. Oxford University Press, 2007.

[28] Jarissa Maselyne, Wouter Saeys, Bart De Ketelaere, Petra Briene, Sam Millet, Frank Tuyttens, and Annelies Van Nuffel. How do fattening pigs spend their day?, 09 2014.

[29] Benjamin L Hart. Behavioral adaptations to pathogens and parasites: five strategies. *Neuroscience & Biobehavioral Reviews*, 14(3):273–294, 1990.

[30] David Fraser. Understanding animal welfare. *Acta Veterinaria Scandinavica*, 50(1):1–7, 2008.

[31] Paul H Hemsworth, Grahame J Coleman, John L Barnett, and Samantha Borg. Relationships between human-animal interactions and productivity of commercial dairy cows. *Journal of animal science*, 78(11):2821–2831, 2000.

[32] RG Huhn. Swine enzootic pneumonia: incidence and effect on rate of body weight gain. *American journal of veterinary research*, 31:1097–1108, 1970.

[33] TM Brown-Brandl and RA Eigenberg. Development of a livestock feeding behavior monitoring system. *Transactions of the ASABE*, 54(5):1913–1920, 2011.

[34] Anita Kapun, Felix Adrion, and Eva Gallmann. Activity analysis to detect lameness in pigs with a uhf-rfid system. In *10th International Livestock Environment Symposium (ILES X)*, page 1. American Society of Agricultural and Biological Engineers, 2018.

[35] Stephen G Matthews, Amy L Miller, Thomas PlÖtz, and Ilias Kyriazakis. Automated tracking to measure behavioural changes in pigs for health and welfare monitoring. *Scientific reports*, 7(1):1–12, 2017.

[36] Jinseong Kim, Yeonwoo Chung, Younchang Choi, Jaewon Sa, Heegon Kim, Yongwha Chung, Daihee Park, and Hakjae Kim. Depth-based detection of standing-pigs in moving noise environments. *Sensors*, 17(12):2757, 2017.

[37] Mateusz Mittek, Eric T Psota, Jay D Carlson, Lance C Pérez, Ty Schmidt, and Benny Mote. Tracking of group-housed pigs using multi-ellipsoid expectation maximisation. *IET Computer Vision*, 12(2):121–128, 2017.

[38] Miso Ju, Younchang Choi, Jihyun Seo, Jaewon Sa, Sungju Lee, Yongwha Chung, and Daihee Park. A kinect-based segmentation of touching-pigs for real-time monitoring. *Sensors*, 18(6):1746, 2018.

[39] Martin Riekert, Achim Klein, Felix Adrion, Christa Hoffmann, and Eva Gallmann. Automatically detecting pig position and posture by 2d camera imaging and deep learning. *Computers and Electronics in Agriculture*, 174:105391, 2020.

[40] Andrea Pezzuolo, Marcella Guarino, Luigi Sartori, Luciano A González, and Francesco Marinello. On-barn pig weight estimation based on body measurements by a kinect v1 depth camera. *Computers and Electronics in Agriculture*, 148:29–36, 2018.

[41] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[42] Rina Dechter. Learning while searching in constraint-satisfaction-problems. *AAAI*, pages 178–185, 01 1986.

[43] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[44] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[45] Dave Steinkraus, Ian Buck, and PY Simard. Using gpus for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120. IEEE, 2005.

[46] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[47] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[48] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[50] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[51] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen. Video captioning with attention-based lstm and semantic consistency. *IEEE Transactions on Multimedia*, 19(9):2045–2055, 2017.

[52] Yang Li, Ting Liu, Jing Jiang, and Liang Zhang. Hashtag recommendation with topical attention-based lstm. In *Proceedings of the 26th International Conference on Computational Linguistics*. Coling, 2016.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[54] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[55] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[57] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[58] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[59] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[61] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[62] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[65] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[66] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[67] Shiyao Wang, Minlie Huang, and Zhidong Deng. Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI*, pages 4468–4474, 2018.

[68] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

[69] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[71] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[72] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[73] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[74] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

[75] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[76] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[77] Gustavo Marques Mostaco, Icaro Ramires Costa De Souza, Leonardo Barreto Campos, and Carlos Eduardo Cugnasca. Agronomobot: a smart answering chatbot applied to agricultural sensor networks. In *14th international conference on precision agriculture*, volume 24, pages 1–13, 2018.

[78] Subhajit Sengupta and Won Suk Lee. Identification and determination of the number of immature green citrus fruit in a canopy under different ambient light conditions. *Biosystems Engineering*, 117:51–61, 2014.

[79] Chia-Lin Chung, Kai-Jyun Huang, Szu-Yu Chen, Ming-Hsing Lai, Yu-Chia Chen, and Yan-Fu Kuo. Detecting bakanae disease in rice seedlings by machine vision. *Computers and electronics in agriculture*, 121:404–411, 2016.

[80] Iván Ramírez Morales, Daniel Rivero Cebrián, Enrique Fernández Blanco, and Alejandro Pazos Sierra. Early warning in egg production curves from commercial hens: A svm approach. *Computers and Electronics in Agriculture*, 121:169–179, 2016.

[81] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[82] Suchet Bargoti and James Underwood. Deep fruit detection in orchards. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3633. IEEE, 2017.

[83] Nicolai Häni, Pravakar Roy, and Volkan Isler. A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, 37(2):263–282, 2020.

[84] Mads Dyrmann, Rasmus Nyholm Jørgensen, and Henrik Skov Midtiby. Roboweedsupport-detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. *Advances in Animal Biosciences: Precision Agriculture*, 8(2):842–847, 2017.

[85] J Barker, S Sarathy, and AT July. Detectnet: Deep neural network for object detection in digits. *Nvidia,(retrieved: 2020-09-10), https://web.archive.org/web/20200809183711/https://developer.nvidia.com/blog/detectnet-deep-neural-network-object-detection-digits/*, 2016.

[86] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[87] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[88] Junfeng Gao, Andrew P French, Michael P Pound, Yong He, Tony P Pridmore, and Jan G Pieters. Deep convolutional neural networks for image-based convolvulus sepium detection in sugar beet fields. *Plant Methods*, 16(1):1–12, 2020.

[89] Dino Ienco, Raffaele Gaetano, Claire Dupaquier, and Pierre Maurel. Land cover classification via multitemporal spatial data by deep recurrent neural networks. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1685–1689, 2017.

[90] Dinh Ho Tong Minh, Dino Ienco, Raffaele Gaetano, Nathalie Lalande, Emile Ndikumana, Faycal Osman, and Pierre Maurel. Deep recurrent neural networks for mapping winter vegetation quality coverage via multi-temporal sar sentinel-1. *arXiv preprint arXiv:1708.03694*, 2017.

[91] Marc Rußwurm and M Körner. Multi-temporal land cover classification with long short-term memory neural networks. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:551, 2017.

[92] Beibei Xu, Wensheng Wang, Greg Falzon, Paul Kwan, Leifeng Guo, Zhiguo Sun, and Chunlei Li. Livestock classification and counting in quadcopter aerial images using mask r-cnn. *International Journal of Remote Sensing*, pages 1–22, 2020.

[93] Liang Han, Pin Tao, and Ralph R Martin. Livestock detection in aerial images using a fully convolutional network. *Computational Visual Media*, 5(2):221–228, 2019.

[94] Theo GM Demmers, S Gauss, CM Wathes, Y Cao, DJ Parsons, et al. Simultaneous monitoring and control of pig growth and ammonia emissions. In *The Ninth International Livestock Environment Symposium (ILES IX). International Conference of Agricultural Engineering-CIGR-AgEng 2012: Agriculture and Engineering for a Healthier Life, Valencia, Spain, 8-12 July 2012*. CIGR-EurAgEng, 2012.

[95] Qiumei Yang, Deqin Xiao, and Sicong Lin. Feeding behavior recognition for group-housed pigs with the faster r-cnn. *Computers and Electronics in Agriculture*, 155:453–460, 2018.

[96] Ali Alameer, Ilias Kyriazakis, Hillary A Dalton, Amy L Miller, and Jaume Bacardit. Automatic recognition of feeding and foraging behaviour in pigs using deep learning. *Biosystems Engineering*, 197:91–104, 2020.

[97] Qiumei Yang, Deqin Xiao, and Genxing Zhang. An algorithm for pig detection and behavior recognition based on deep learning technique. *Animal Environment and Welfare*, 2017.

[98] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019.

[99] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6036–6046, 2018.

[100] Eric T Psota, Ty Schmidt, Benny Mote, and Lance C Pérez. Long-term tracking of group-housed livestock using keypoint detection and map estimation for individual animal identification. *Sensors*, 20(13):3670, 2020.

[101] Lei Zhang, Helen Gray, Xujiong Ye, Lisa Collins, and Nigel Allinson. Automatic individual pig detection and tracking in pig farms. *Sensors*, 19(5):1188, 2019.

[102] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[103] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. Stock market forecast using multivariate analysis with bidirectional and stacked (lstm, gru). In *2018 21st Saudi Computer Society National Computer Conference (NCC)*, pages 1–7. IEEE, 2018.

[104] Sajjad Ali Haider, Syed Rameez Naqvi, Tallha Akram, Gulfam Ahmad Umar, Aamir Shahzad, Muhammad Rafiq Sial, Shoaib Khaliq, and Muhammad Kamran. Lstm neural network based forecasting model for wheat production in pakistan. *Agronomy*, 9(2):72, 2019.

[105] Kiran M Sabu and TK Manoj Kumar. Predictive analytics in agriculture: Forecasting prices of arecanuts in kerala. *Procedia Computer Science*, 171:699–708, 2020.

[106] Siva R Venna, Amirhossein Tavanaei, Raju N Gottumukkala, Vijay V Raghavan, Anthony S Maida, and Stephen Nichols. A novel data-driven model for real-time influenza forecasting. *IEEE Access*, 7:7691–7701, 2018.

[107] G Wang, W Wei, J Jiang, C Ning, H Chen, J Huang, B Liang, N Zang, Y Liao, R Chen, et al. Application of a long short-term memory neural network: a burgeoning method of deep learning in forecasting hiv incidence in guangxi, china. *Epidemiology & Infection*, 147, 2019.

[108] Yongbin Wang, Chunjie Xu, Shengkui Zhang, Li Yang, Zhende Wang, Ying Zhu, and Juxiang Yuan. Development and evaluation of a deep learning approach for modeling seasonality and trends in hand-foot-mouth disease incidence in mainland china. *Scientific reports*, 9(1):1–15, 2019.

[109] Sébastien Fournel, Alain N Rousseau, and Benoit Laberge. Rethinking environment control strategy of confined animal housing systems through precision livestock farming. *Biosystems Engineering*, 155:96–123, 2017.

[110] Jian Cao, Zhi Li, and Jian Li. Financial time series forecasting model based on ceemdan and lstm. *Physica A: Statistical Mechanics and its Applications*, 519:127–139, 2019.

[111] Taewook Kim and Ha Young Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320, 2019.

[112] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[113] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148, 2016.

[114] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[115] Rama K Singh and Vikash C Sharma. Ensemble approach for zoonotic disease prediction using machine learning techniques, 2015.

[116] Pablo Valdes-Donoso, Kimberly VanderWaal, Lovell S Jarvis, Spencer R Wayne, and Andres M Perez. Using machine learning to predict swine movements within a regional program to improve control of infectious diseases in the us. *Frontiers in Veterinary Science*, 4, 2017.

[117] George Gomes Cabral and Adriano Lorena Inacio de Oliveira. One-class classification for heart disease diagnosis. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 2551–2556. IEEE, 2014.

[118] Qian Leng, Honggang Qi, Jun Miao, Wentao Zhu, and Guiping Su. One-class classification with extreme learning machine. *Mathematical problems in engineering*, 2015, 2015.

[119] Robin Thompson, Stephanie M Matheson, Thomas Plötz, Sandra A Edwards, and Ilias Kyriazakis. Porcine lie detectors: Automatic quantification of posture state and transitions in sows using inertial sensors. *Computers and electronics in agriculture*, 127:521–530, 2016.

[120] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14, 2007.

[121] Mohammed A Ambusaidi, Zhiyuan Tan, Xiangjian He, Priyadarsi Nanda, Liang Fu Lu, and Aruna Jamdagni. Intrusion detection method based on nonlinear correlation measure. *International Journal of Internet Protocol Technology 7*, 8(2-3):77–86, 2014.

[122] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[123] Weng-Keen Wong, Andrew W Moore, Gregory F Cooper, and Michael M Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 808–815, 2003.

[124] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. What's strange about recent events (wsare): An algorithm for the early detection of disease outbreaks. *Journal of Machine Learning Research*, 6(Dec):1961–1998, 2005.

[125] Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ecg time signals via deep long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–7. IEEE, 2015.

[126] Shah Ahsanul Haque, Mustafizur Rahman, and Syed Mahfuzul Aziz. Sensor anomaly detection in wireless sensor networks for healthcare. *Sensors*, 15(4):8764–8786, 2015.

[127] Bharadwaj Veeravalli, Chacko John Deepu, and DuyHoa Ngo. Real-time, personalized anomaly detection in streaming data for wearable healthcare devices. In *Handbook of Large-Scale Distributed Computing in Smart Healthcare*, pages 403–426. Springer, 2017.

[128] S Kanarachos, J Mathew, A Chroneos, and M Fitzpatrick. Anomaly detection in time series data using a combination of wavelets, neural networks and hilbert transform. In *IISA*, pages 1–6, 2015.

[129] Jinbo Li, Witold Pedrycz, and Iqbal Jamal. Multivariate time series anomaly detection: A framework of hidden markov models. *Applied Soft Computing*, 60:229–240, 2017.

[130] Mooi Choo Chuah and Fen Fu. Ecg anomaly detection via time series analysis. In *International Symposium on Parallel and Distributed Processing and Applications*, pages 123–135. Springer, 2007.

[131] Xue-Wen Chen and Xiaotong Lin. Big data deep learning: challenges and perspectives. *IEEE access*, 2:514–525, 2014.

[132] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[133] Zoetis. *Individual Pig Care*. Zoetis, 2012.

[134] General Alert. Application areas. http://www.general-alert.com/Applications, 2011. Last accessed 30/04/2018.

[135] Colin T Whittemore and Ilias Kyriazakis. *Whittemore's science and practice of pig production*. John Wiley & Sons, 2006.

[136] The World Organisation for Animal Health. *Infection With Porcine Reproductive And Respiratory Syndrome Virus*. The World Organisation for Animal Health, 2014.

[137] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[138] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[139] Jerone TA Andrews, Edward J Morton, and Lewis D Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 6(1):21, 2016.

[140] Ashfaqur Rahman, Daniel Smith, James Hills, Greg Bishop-Hurley, Dave Henry, and Richard Rawnsley. A comparison of autoencoder and statistical features for cattle behaviour classification. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 2954–2960. IEEE, 2016.

[141] Noura Al Moubayed, Toby Breckon, Peter Matthews, and A Stephen McGough. Sms spam filtering using probabilistic topic modelling and stacked denoising autoencoder. In *International Conference on Artificial Neural Networks*, pages 423–430. Springer, 2016.

[142] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[143] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[144] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[145] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[146] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[147] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[148] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[149] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[150] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[151] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[152] LinkedIn. Luminol. https://github.com/linkedin/luminol, 2016. Last accessed 30/04/2018.

[153] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[154] Li Wei, Nitin Kumar, Venkata Nishanth Lolla, Eamonn J Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. Assumption-free anomaly detection in time series. In *SSDBM*, volume 5, pages 237–242, 2005.

[155] Sten Mortensen, Henrik Stryhn, Rikke Søgaard, Anette Boklund, Katharina DC Stärk, Jette Christensen, and Preben Willeberg. Risk factors for infection of sow herds with porcine reproductive and respiratory syndrome (prrs) virus. *Preventive veterinary medicine*, 53(1):83–101, 2002.

[156] Daniel Berckmans. Precision livestock farming technologies for welfare management in intensive livestock systems. *Scientific and Technical Review of the Office International des Epizooties*, 33(1):189–196, 2014.

[157] Stephen G Matthews, Amy L Miller, James Clapp, Thomas Plötz, and Ilias Kyriazakis. Early detection of health and welfare compromises through automated detection of behavioural changes in pigs. *The Veterinary Journal*, 217:43–51, 2016.

[158] Yongwha Chung, Seunggeun Oh, Jonguk Lee, Daihee Park, Hong-Hee Chang, and Suk Kim. Automatic detection and recognition of pig wasting diseases using sound data in audio surveillance systems. *Sensors*, 13(10):12929–12942, 2013.

[159] Viktoriya Krakovna and Finale Doshi-Velez. Increasing the interpretability of recurrent neural networks using hidden markov models. *arXiv preprint arXiv:1606.05320*, 2016.

[160] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[161] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[162] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[163] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[164] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[165] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[166] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[167] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[168] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[169] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[170] Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 639–645, 2016.

[171] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[172] L Harrison, William D Penny, and Karl Friston. Multivariate autoregressive modeling of fmri time series. *Neuroimage*, 19(4):1477–1491, 2003.

[173] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[174] CA Coello Coello and M Salazar Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE, 2002.

[175] Martina Velasova, Pablo Alarcon, Susanna Williamson, and Barbara Wieland. Risk factors for porcine reproductive and respiratory syndrome virus infection and resulting challenges for effective disease surveillance. *BMC veterinary research*, 8(1):184, 2012.

[176] Sparsh Mittal. A survey on optimized implementation of deep learning models on the nvidia jetson platform. *Journal of Systems Architecture*, 97:428–442, 2019.

[177] Benjamin Hawks, Pradeep Jasal, Michael Wang, and Brian Nord. Real-time machine learning inferencing with edge computing devices from google and intel. Technical report, Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 2019.

[178] Sophie Helene Richter and Sara Hintze. From the individual to the population–and back again? emphasising the role of the individual in animal welfare science. *Applied Animal Behaviour Science*, 2018.

[179] Zoe Berk, Yan CSM Laurenson, Andrew B Forbes, and Ilias Kyriazakis. Modelling the consequences of targeted selective treatment strategies on performance and emergence of anthelmintic resistance amongst grazing calves. *International Journal for Parasitology: Drugs and Drug Resistance*, 6(3):258–271, 2016.

[180] Ahmet Yigit and Alptekin Temizel. Individual and group tracking with the evaluation of social interactions. *IET Computer Vision*, 11(3):255–263, 2016.

[181] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[182] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[183] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal Of Computer Vision*, 115(3):211–252, 2015.

[184] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html.

[185] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars: A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision*, pages 868–884. Springer, 2016.

[186] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[187] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[188] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[189] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2015.

[190] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015.

[191] Wendi Cai, Jiadie Li, Zhongzhao Xie, Tao Zhao, and LU Kang. Street object detection based on faster r-cnn. In *2018 37th Chinese Control Conference (CCC)*, pages 9500–9503. IEEE, 2018.

[192] Roberto Olmos, Siham Tabik, and Francisco Herrera. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, 275:66–72, 2018.

[193] Chen-Lin Zhang, Jian-Hao Luo, Xiu-Shen Wei, and Jianxin Wu. In defense of fully connected layers in visual representation transfer. In *Pacific Rim Conference on Multimedia*, pages 807–817. Springer, 2017.

[194] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[195] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[196] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[197] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[198] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 325(6):653–658, 1997.

[199] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.

[200] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 748–756. IEEE, 2018.

[201] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference On Computer Vision*, pages 21–37. Springer, 2016.

[202] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic matching using m-best solutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–145, 2016.

[203] Sakrapee Paisitkriangkrai, Chunhua Shen, and Anton Van Den Hengel. Learning to rank in person re-identification with metric ensembles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1846–1855, 2015.

[204] Xu Lan, Xiatian Zhu, and Shaogang Gong. Person search by multi-scale matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 536–552, 2018.

[205] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[206] Christoph Heindl. py-motmetrics. https://github.com/cheind/py-motmetrics, 2019.

[207] Edmund A Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1-2):203–224, 1965.

[208] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, volume 90, page 91. Citeseer, 2006.

[209] Vishwanath A Sindagi and Vishal M Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018.

[210] Jonguk Lee, Long Jin, Daihee Park, and Yongwha Chung. Automatic recognition of aggressive behavior in pigs using a kinect depth sensor. *Sensors*, 16(5):631, 2016.

[211] Jaewon Sa, Younchang Choi, Hanhaesol Lee, Yongwha Chung, Daihee Park, and Jinho Cho. Fast pig detection with a top-view camera under various illumination conditions. *Symmetry*, 11(2):266, 2019.

[212] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.

[213] Mateusz Mittek, Eric T Psota, Lance C Pérez, Ty Schmidt, and Benny Mote. Health monitoring of group-housed pigs using depth-enabled multi-object tracking. In *Proceedings of Int Conf Pattern Recognit, Workshop on Visual observation and analysis of Vertebrate And Insect Behavior*, 2016.

[214] Paul Martin, Paul Patrick Gordon Bateson, and Patrick Bateson. *Measuring behaviour: an introductory guide*. Cambridge University Press, 1993.

[215] Amy L Miller, Hillary A Dalton, Theo Kanellos, and Ilias Kyriazakis. How many pigs within a group need to be sick to lead to a diagnostic change in the group's behavior? *Journal Of Animal Science*, 97(5):1956–1966, 2019.

[216] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[217] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[218] Xiaobin Chang, Timothy M Hospedales, and Tao Xiang. Multi-level factorisation net for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2109–2118, 2018.

[219] Xuelin Qian, Yanwei Fu, Yu-Gang Jiang, Tao Xiang, and Xiangyang Xue. Multi-scale deep learning architectures for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5399–5408, 2017.

[220] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. *arXiv preprint arXiv:1905.00953*, 2019.

[221] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[222] Qian Yu, Xiaobin Chang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. The devil is in the middle: Exploiting mid-level representations for cross-domain instance matching. *arXiv preprint arXiv:1711.08106*, 2017.

[223] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[224] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[225] Abrar H Abdulnabi, Gang Wang, Jiwen Lu, and Kui Jia. Multi-task cnn model for attribute prediction. *IEEE Transactions on Multimedia*, 17(11):1949–1959, 2015.

[226] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*, 2017.

[227] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[228] Lei Zhang, Helen Gray, Xujiong Ye, Lisa Collins, and Nigel Allinson. Automatic individual pig detection and tracking in surveillance videos. *arXiv preprint arXiv:1812.04901*, 2018.

[229] Young-chul Yoon, Abhijeet Boragule, Young-min Song, Kwangjin Yoon, and Moongu Jeon. Online multi-object tracking with historical appearance matching and scene adaptive detection filtering. In *2018 15th IEEE International conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[230] Michael J Kane, Natalie Price, Matthew Scotch, and Peter Rabinowitz. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, 15(1):276, 2014.

[231] F Blecha, DS Pollman, and DA Nichols. Weaning pigs at an early age decreases cellular immunity. *Journal of Animal Science*, 56(2):396–400, 1983.

[232] Home Office. Code of practice for the housing and care of animals bred, supplied or used for scientific purposes, 2014.

[233] Ancong Wu, Wei-Shi Zheng, and Jian-Huang Lai. Robust depth-based person re-identification. *IEEE Transactions on Image Processing*, 26(6):2588–2603, 2017.

[234] Zimo Liu, Dong Wang, and Huchuan Lu. Stepwise metric promotion for unsupervised video person re-identification. In *Proceedings of the IEEE international conference on computer vision*, pages 2429–2438, 2017.

[235] Minxian Li, Xiatian Zhu, and Shaogang Gong. Unsupervised person re-identification by deep learning tracklet association. In *Proceedings of the European conference on computer vision (ECCV)*, pages 737–753, 2018.

[236] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Deep association learning for unsupervised video person re-identification. *arXiv preprint arXiv:1808.07301*, 2018.

[237] Minxian Li, Xiatian Zhu, and Shaogang Gong. Unsupervised tracklet person re-identification. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[238] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An

imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[239] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[240] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585:357–362, 2020.

[241] WA Falcon. Pytorch lightning. https://github.com/PyTorchLightning/pytorch-lightning, 2019.

[242] Weights & Biases. Weights & Biases. http://wandb.com/, 2017.