Detecting cow behaviours associated with parturition using computer vision

John McDonagh

Computer Vision Laboratory

School of Computer Science

University of Nottingham

This dissertation is submitted in partial fulfilment of the conditions for the award of the degree of *Doctor of Philosophy*.

February 2023

Abstract

Monitoring of dairy cows and their calf during parturition is essential in determining if there are any associated problems for mother and offspring. This is a critical period in the productive life of the mother and offspring. A difficult and assisted calving can impact on the subsequent milk production, health and fertility of a cow, and its potential survival. Furthermore, an alert to the need for any assistance would enhance animal and stockperson wellbeing. Manual monitoring of animal behaviour from images has been used for decades, but is very labour intensive. Recent technological advances in the field of Computer Vision based on the technique of Deep Learning have emerged, which now makes automated monitoring of surveillance video feeds feasible. The benefits of using image analysis compared to other monitoring systems is that image analysis relies upon neither transponder attachments, nor invasive tools and may provide more information at a relatively low cost. Image analysis can also detect and track the calf, which is not possible using other monitoring methods. Using cameras to monitor animals is commonly used, however, automated detection of behaviours is new especially for livestock.

Using the latest state-of-the-art techniques in Computer Vision, and in particular the ground-breaking technique of Deep Learning, this thesis develops a vision-based model to detect the progress of parturition in dairy cows. A large-scale dataset of cow behaviour annotations was created, which included over 46 individual cow calvings and is approximately 690 hours of video footage with over 2.5k of video clips, each between 3-10 seconds. The model was trained on seven different behaviours, which included standing, walking, shuffle, lying, eating, drinking, and contractions while lying. The developed network correctly classified the seven behaviours with an accuracy of between 80 to 95%. The accuracy in predicting contractions while lying down was 83%, which in itself can be an early warning calving alert, as all cows start contractions one to two hours before giving birth. The performance of the model developed was also comparable to methods for human action classification using the Kinetics dataset.

Acknowledgements

The five years of this project has been a personal journey, which would not be possible without the support of my wife.

I am grateful for the support of my supervisors (Yorgos Tzimiropoulos, Matt Bell, Tony Pridmore and Pete Down) and technical staff at Nottingham, including Joe Best, for assistance on setting up camera surveillance equipment and data storage capabilities for the project. The ongoing use of the facilities and monitoring of the calving cows at University of Nottingham Dairy Centre made the data collection possible.

This project would not have been possible without a postgraduate scholarship from the EPSRC National Productivity Investment Fund. The work was also supported by the Douglas Bomford Trust (Alan Plom, David White and Paul Miller), which they mentored as part of their funding towards 'Cow Vision' work at Nottingham.

Contents

List of tables

List of Figures

Chapter 1

1 Introduction

The dairy sector has great importance to the UK economy. The 1.8 million cows in the UK produce about 14 million tonnes of milk each year (valued at £4bn), making it the tenth largest global milk producing country (Agriculture and Horticulture Development Board (AHDB), 2015). Dairy production has made large advances in efficiencies over the past 60 years as a result of changes in breeding, nutrition and management, and further improvements appear possible with technology (Bell and Tzmiropolous, 2018). In particular, the health and welfare of animals has great importance to consumers and is also important for the sustainability of milk production. Maintaining healthy cows is also of interest to the producer, with increased production, particularly later in life with better longevity (Bell et al., 2015).

1.1 The importance of parturition

The act of giving birth, medically referred to as parturition, unfolds in three distinct stages: cervical dilation, calf delivery, and the subsequent expulsion of the placenta. On occasion, this process encounters complications, leading to what is known as dystocia. Dystocia arises when the birthing process becomes challenging or obstructed, necessitating external assistance to facilitate delivery. Its primary origins can be traced to three key factors: a mismatch in size between the mother and calf, irregularities in calf presentation, positioning, or posture within the birth canal, and maternal factors, including conditions such as low calcium levels, medically termed hypocalcemia.

Close monitoring of cow behaviour during parturition is required by the stockperson to determine the progress of birth and the potential need for intervention. Typically, a group of pregnant animals would be observed at sporadic intervals with close monitoring as animals show key signs associated with imminent appearance of the young animal. Ultimately the most important outcome is the survival of the mother and her offspring, with problems potentially impacting on future lifetime performance for both. While some idea of expected calving date is often known by the stockperson, or estimated from time of insemination and gestation length, this estimate is often imprecise and requires some subjective judgement of stage of pregnancy.

Behavioural changes, such as standing and lying bouts, can give an indication to whether there is a need for assistance. The frequency of lying, standing and tail movements of an animal have been found to change in the period prior to calving in both dairy (Miedema et al., 2011) and beef cattle (Hyslop et al., 2008), and may give some indication of the need for assistance. Dystocia is fairly common in dairy cows and is a major cause of calf mortality (Lombard et al., 2007). Barrier et al. (2013) found that calves which survived dystocia had poorer welfare in the neonatal period and possibly beyond, with higher mortality and higher physiological stress. Although preventing dystocia is close to impossible, quick and timely intervention will help avoid the risk of poor health and welfare outcomes. Individual evaluation and continuous monitoring of dairy cows around the time of calving is important to identify any need for assistance or health problems as early as possible.

1.2 Use of technology to support a stockperson

Several sensor technologies (Wathes et al., 2008; Neethirajan, 2017) that can be used to monitor animals exist such as accelerometers, GPS, rumen boluses and temperature sensors (Figure 1.1). In terms of activity and behaviour, most research to date has focused on 3-dimensional accelerometer-based movement sensors (Diosdado et al. 2015, Rahman et al. 2018 and Benaissa et al. 2019), which are relatively cheap and simple to implement.

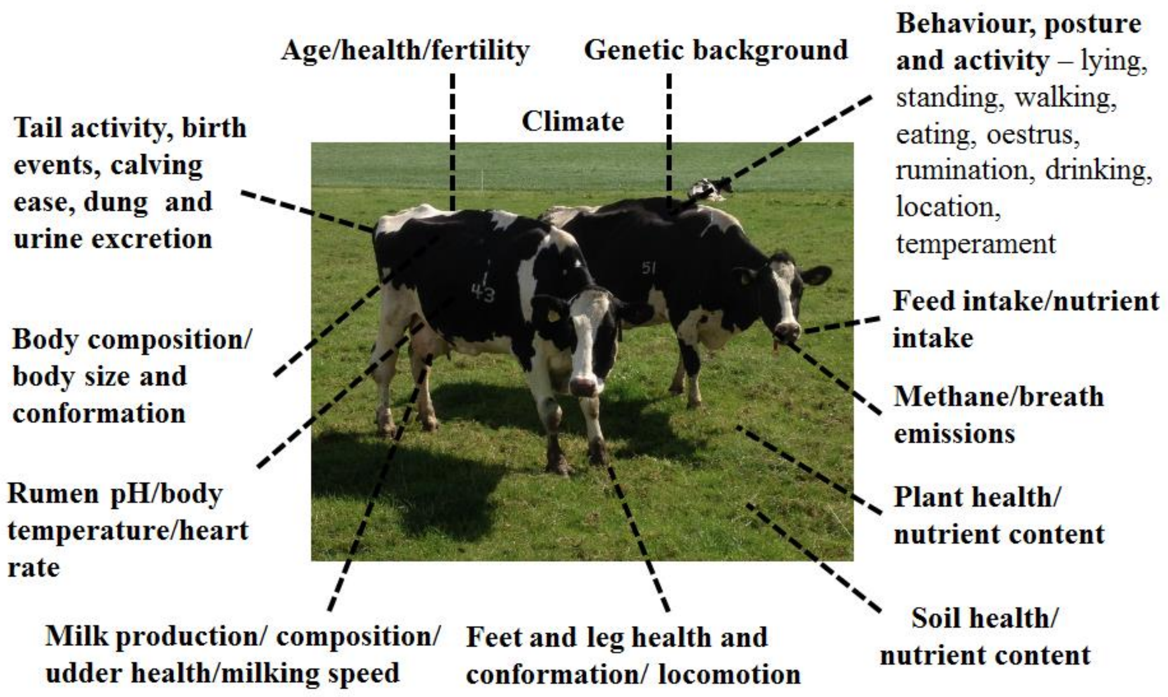
Age/health/fertility
Genetic background
Behaviour, posture and activity – lying, standing, walking, eating, oestrus, rumination, drinking, location, temperament
Climate
Tail activity, birth events, calving ease, dung and urine excretion
Feed intake/nutrient intake
Body composition/ body size and conformation
Methane/breath emissions
Plant health/ nutrient content
Rumen pH/body temperature/heart rate
Soil health/ nutrient content
Milk production/ composition/ udder health/milking speed
Feet and leg health and conformation/ locomotion

Figure 1.1. Potential data sources used to monitor and manage cows and their environment (Source: Bell and Tzmiropolous, 2018).

Less invasive technologies are emerging such as image analysis. Camera surveillance equipment has been used for decades in animal research to observe animals and their behaviours. Technologies that don't rely on human intervention, transponder attachments, or complex equipment (e.g. boluses, collars), may provide more information compared to other monitoring systems at a relatively low cost. For example, an image can capture location, pose and interactions between multiple objects. Also, existing movement or activity sensors, such as accelerometers, are calibrated using video image material. Accelerometers provide information on both body posture (standing, lying, walking) and activity, which are used as descriptors to define behaviours. Accelerometers have for several decades provided a useful tool to help farmers to identify oestrus activity in cows from clear peaks in activity signals (Wathes et al. 2008). The data from an animal mounted sensor can be acquired from the animal when they visit a common location such as milking station, feed and/or water trough. A potential disadvantage of video image monitoring of animals compared to animal mounted sensors is that that cameras are more suited to indoor environments and not outdoor conditions. However, drone mounted cameras and outdoor camera surveillance systems are becoming more common.

As financial pressures on farmers increases (Defra, 2018), each stockperson will be expected to look after more animals. Tools that can assist farmers in monitoring individual animals or groups will be beneficial to the animal and farmer. Enhanced monitoring tools will enable farm labour to be targeted towards those animals that need it most. For example, management at calving plays an important role in the subsequent health and reproductive performance of cattle during their lifetime (Bell et al., 2007). The development of precision monitoring of individual animals that are non-invasive, automated and produce results in real-time, such as digital image applications and online measurements, are becoming more available as 'machine learning' technologies develop and the cost of implementation on farms reduces. Such technologies have the potential to allow welfare and health issues to be detected quickly for more animals compared to more manual methods currently used, thus improving animal health and welfare outcomes.

1.3 Potential for image analysis technology

Recent technological advances in the field of Computer Vision based on the technique of deep learning (Krizhevsky et al., 2012, Girshick et al., 2014) have emerged which now makes automated monitoring of video feeds possible. Deep neural networks can be used for a number of animal monitoring tasks such as recognising the type of animals (recognition), detecting where the animals (and any other objects of interest) are located in the image (detection), localising their body parts, and even segmenting their exact shape (silhouette) from the image. Furthermore, advancements within the field of computer vision, namely action recognition have made it possible to capture spatiotemporal information across multiple frames with a high degree of certainty. With the recent introduction of non-local operations, that was proposed by

Wang et al. (2018), it is now possible to capture long-range dependencies directly by computing interactions between any two points, regardless of their distance from one another. With rapid developments in camera surveillance technology, machine learning and processing, and Computer Vision techniques, new objective methods to monitor animals are possible that can help improve early detection of health, fertility and welfare problems. The combination of sensors i.e. images with transponder technologies, may ultimately provide a more 'complete' approach to monitoring animal wellbeing.

To assist a stockperson at calving, and given the importance of a successful birth and potential need for intervention as mentioned above, a number of sensor technologies have been developed to focus on the task of birth detection. These technologies have largely been based on accelerometers and movement detection (Rutten et al., 2017; Giaretta et al., 2021). A potential alternative is Computer Vision (Cangar et al., 2008; Bell and Tzimiropoulos, 2018), which offers the opportunity to capture detailed behaviours associated with birth events (Table 1.1) and interactions (e.g. calf suckling, cow licking etc) among animals. A major benefit of automated image analysis is that it allows continuous monitoring for long-periods of time which is not possible for a stockperson.

Table 1.1. Ethogram of common behavioural states and events associated with parturition in cows.

State 1 (Posture)	State 2 (Behaviour)	Events (Behaviour)	Events (Parturition)
Stand	Eat	Vocalisation	Water bag
Walk	Ruminate	Contraction	Calf feet
Lie	Drink	Tail swish	Calf head
	Sniff/lick	Stamp/kick	Calf shoulders
	Circle/shuffle	Head turned	Calf hips
	Idle/other		Birth
			Cow licks calf

			Calf stands
			Calf sucks

1.4 Deep Learning

The continuous advancements in the field of deep learning have paved the way for innovative applications, one of which is animal behaviour recognition. Given the intricate patterns and diverse nuances in animal behaviours, specialised neural networks have become instrumental in deciphering and categorising them. In this section, we elucidate the foundational neural architectures that have made significant strides in the realm of animal behaviour recognition, discussing their core mechanisms, advantages, and limitations.

1.4.1 Neural Networks (FCN)

At the core of deep learning lies the conventional feed-forward neural network, commonly referred to as the fully connected network (FCN), which serves as the foundation for more sophisticated architectures. Feed forward neural networks are multi-layered fully connected networks, where each neuron in a layer is connected to all the neurons from the previous layer through weighted connections. The network consists of an input layer, multiple hidden layers, and an output layer. Data enters at the inputs and propagates forward through the network, layer by layer, until it reaches the output. A neuron calculates the weighted sum of its inputs, offset by a bias, and passes the resulting scalar value through a non-linear activation function. The output of the neuron is referred to as an activation. During training, the weights and biases are learned to optimise the network's performance.

1.4.2 Convolutional neural network (CNN)

While fully connected neural networks can be powerful, a notable drawback is their disregard for the spatial structure inherent in images. To address this limitation, Convolutional Neural Networks (CNNs) have been designed. These advanced neural architectures excel in tasks like image recognition, object detection, and facial and action recognition. CNNs comprise of convolutional layers, pooling layers, and fully connected layers. Central to a convolutional layer are filters, learnable weight matrices, that scan the input to extract critical features, resulting in a feature map. Such extraction is crucial for identifying patterns ranging from simple edges to complex textures. Subsequently, the pooling layer refines the feature map, diminishing its spatial dimensions through down-sampling. Finally, the fully connected layer harnesses these refined features, ensuring precise classification of the input image.

1.4.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) excel at processing sequences and temporal data, making them indispensable for tasks like natural language processing, time-series forecasting, and speech recognition. Central to an RNN is its memory mechanism, which allows it to retain past information and use it as context for future inputs, setting it apart from traditional feed-forward neural networks. This unique architecture, however, has its challenges, notably the issue of vanishing or exploding gradients. In essence, during training, the gradients can either diminish to nearly zero (vanishing) or escalate uncontrollably (exploding), leading to difficulty in learning long-range dependencies or unstable model training, respectively. To address these challenges, advanced RNN variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been developed (Hochreitern and Schmidhuber, 1997; Cho et al., 2014). These structures enhance RNNs with gated mechanisms to capture long-term temporal relationships more effectively. Thus, RNNs, with their inherent memory and improved architectures, adeptly process sequences, considering both immediate and historical context.

1.4.4 Transfer Learning and Pre-trained Models

Given the vast computational resources required to train deep networks, the concept of transfer learning has gained traction. Instead of training a model from scratch, transfer learning leverages pre-trained models, fine-tuning them for specific tasks. Many of these models, initially trained on extensive datasets, capture generic features that can be applicable across multiple domains. By adding task-specific layers and refining them, researchers can achieve state-of-the-art performance in animal behaviour recognition without the need for exhaustive training, making the entire process more efficient and accessible.

1.5 Background

Calving prediction stands as a cornerstone in dairy farming operations, crucial for optimising animal welfare and resource management. Precision in anticipating calving events is essential to ensure the well-being of both the cow and the calf and to manage the risks and complications associated with labour, such as dystocia. The incorporation of machine and deep learning technologies has led to significant innovations in creating models to predict calving events, offering transformative tools and insights for the dairy industry. This section provides a synthesis of key studies and their contributions to the field of calving predictions.

1.5.1 Behavioural Changes as Indicators

Miedema et al. (2011) conducted a study on the behaviour changes of twenty Holstein-Friesian cows over a 24hr period before calving. Video captured over this period was used to collect both the frequencies and durations of behaviours. The collected data was split into four equal segments of six-hours each, to determine the time when changes occurred. They found that the occurrences of lying bouts and tail raising greatly increased during the final six-hour period before calving and could potentially be used to predict calving. Similarly, Giaretta et al. (2021) also found increased tail movements as an important indicator of calving progress, along with decreased eating behaviour and rumination time.

A study on 32 dairy cows, which was conducted by Jensen (2012), showed that cow behaviour changed significantly between 2 to 6 hours prior to parturition. The authors found increases in lying bouts, restlessness, head activity and tail raising. A decrease in the eating and drinking duration was also observed. The authors suggested that changes in cow behaviours from 6 hours prior to calving could be used as an indicator of imminent calving.

Recently, a more extensive study by Titler et al. (2015) used electronic data loggers to record the behavioural activity of 132 dairy cows from 3 different herds. They recorded standing time, lying time, steps, lying bouts and duration of lying bouts prior to parturition. Their study found that there was an increase in steps and standing times, a decrease in lying time, shorter lying bouts and an increase in the number of lying bouts on the days leading up to the day of calving. They concluded that dairy cows approaching parturition show distinct behavioural changes which can be observed between 2 to 14 hours, averaging at 6 hours, before calf birth.

1.5.2 The Importance of Calving Prediction Models

Efficient and accurate prediction models are imperative for mitigating the risks of dystocia and for optimising resource allocation in dairy farming. Dystocia poses substantial health risks to both the cow and the calf, making predictive models vital for preventive and responsive measures. The advent of machine learning and deep learning has been pivotal in developing advanced models that offer improved predictive accuracy and insights in calving events.

1.5.3 Evolution of Predictive Models: Machine Learning to Deep Learning

1.5.3.1 Foundational Models

Fenlon et al. (2017) pioneered the development of models to estimate the level of assistance required during calving events, employing machine learning techniques such as multinomial regression, decision trees, random forests, and neural networks. Their work was foundational, showing that neural networks and multinomial regression models can predict up to 75% of calving events accurately.

1.5.3.2 Behavioural Data Integration

Borchers et al. (2017) advanced this field by using data on cattle behaviour and employing machine learning techniques, including random forest and linear discriminant analysis. The study illustrated the potential of machine learning in calving estimation but also highlighted the need for larger datasets and further refinement of models.

1.5.3.3 Advanced Sensor-Based Models

Incorporating sensor technologies provided a more granular perspective into the behaviours and physiological states associated with calving. Rutten et al. (2017) and Zehner et al. (2018) leveraged sensors to enhance model accuracy in predicting calving onset. Fadul et al. (2017) innovatively combined data from the RumiWatch noseband-sensor and a 3D-accelerometer to predict calving time in Holstein-Friesian cows, showing a clear connection between cow behaviour and the onset of calving. These advanced models highlighted the potential and challenges of sensor-based data, emphasising the need for minimising false positives for practical applicability.

1.5.3.4 Deep Learning Advancements

Keceli et al. (2020) marked a significant milestone by applying deep learning techniques, specifically Bi-directional Long Short-Term Memory (Bi-LSTM) method, showing enhanced classification accuracy in predicting the calving day over standard LSTM. The use of deep learning algorithms enabled more sophisticated analysis of patterns and improved the predictive accuracy of calving events, showcasing the potential of deep learning in this domain.

1.5.4 Comparative Analysis and Implications

1.5.4.1 Comparative Efficiency

Comparison across studies reveals an evolving trend towards more accurate and reliable models. While early models laid the groundwork, the integration of behavioural data, sensor technologies, and advanced deep learning techniques has significantly elevated the predictive capabilities. However, achieving a balance between accuracy and practical applicability remains a challenge.

1.5.4.2 Practical Implications

The advancements in predictive models offer tangible benefits to dairy farmers, allowing for optimised resource allocation and enhanced animal welfare. However, the translation of these models into practical, real-world solutions necessitates further research and development to minimise false positives and improve reliability.

1.5.4.3 Conclusion and Future Directions

The journey from machine learning models to the incorporation of deep learning signifies the ongoing evolution in calving prediction technologies. While substantial progress has been made, the pursuit for the ideal balance between model accuracy and real-world applicability continues. The future in this field lies in refining existing models, exploring new methodologies, and integrating diverse data sources to develop robust and reliable predictive tools that can revolutionise dairy farming practices.

1.6 Publications

McDonagh, J., Bell, M.J., Slinger, K.R., Tzimiropoulos, G. 2020. Predicting dairy cow behaviour using computer vision. BSAS proceedings, Nottingham, p.26.

McDonagh, J., Tzimiropoulos, G., Slinger, K.R., Huggett, Z.J., Down, P.M., Bell, M.J. 2021. Detecting dairy cow behaviour using vision technology. Agriculture, 11, 675.

Waters, B.E., McDonagh, J., Tzimiropoulos, G., Slinger, K.R., Huggett, Z.J., Bell, M.J. 2021. Changes in sheep behaviour before lambing. Agriculture, 11, 715.

Cavendish, B., McDonagh, J., Tzimiropoulos, G., Slinger, K.R., Huggett, Z.J., Bell, M.J. 2021. Changes in dairy cow behaviour with and without assistance at calving. Agriculture, 11, 722.

Chapter 2

2 Literature Review

2.1 Introduction

This chapter is devoted to offering an exhaustive examination of the literature pertinent to the focal theme of this thesis, which is employing deep learning for the prediction of parturition in dairy cows. Initially, the discussion will revolve around the methodologies related to object detection and action recognition, pivotal for discerning human activities within still images and video sequences, laying a foundational understanding for the application of such methods in animal behaviour analysis. Subsequently, the chapter will delve deeper into the principal area of our inquiry, which is livestock behaviour recognition utilising deep learning, with an emphasis on dairy cows and calving prediction through the application of machine learning and advanced deep learning techniques. The latter sections will spotlight the existent gaps in the literature, particularly focusing on the intersections of deep learning applications and calving prediction, highlighting the areas in need of further exploration and research in this multidisciplinary domain.

2.2 Object detection

Ever since the seminal work of Girshick et al. (2014) where they introduced their region-based convolutional neural network (R-CNN), the use of CNN-based object detectors has become the dominant paradigm in research. CNN-based object detectors can be broken down into two main categories: One-stage detectors, for instance, YOLO (Redmon et al. 2016), SSD (Liu et al. 2016). and RetinaNet (Lin et al. 2017), and two-stage detectors, which include R-FCN (Dai et al. 2016) and the R-CNN family of detectors (Girshick et al. 2014; Girshick 2015; Ren et al. 2015; He et al. 2017).

2.2.1 Two-stage object detection

Two-stage detectors typically divide the task of detecting objects into two phases: the first stage generates a sparse set of candidate object regions of interest, and the second stage refines and classifies each proposed region as either one of the predetermined foreground classes or as background. One of the most influential two-stage object detectors was R-CNN which was proposed by Girshick et al. (2014). They used Selective Search (Uijlings et al. 2013) to find the proposed object regions of interest. Subsequently, these identified regions underwent a transformation, being resized and adjusted to conform to a fixed square size, so that they could be forward propagated through a CNN to compute features. Then, class-specific linear SVMs were used to classify each region. Finally, to improve the localisation performance, bounding-box regressors were used on the classified regions to generate tighter bounding box coordinates.

Although R-CNN achieved excellent accuracy in object detection, it was not without its problems, as it was expensive to train, both in time and storage space. This was because training the parameters for CNN, SVM and the bounding-box regressors had to be done separately. Plus, it also took a long time in detecting objects, as it required a CNN forward pass for each object proposal.

To address these problems Girshick (2015) proposed Fast R-CNN. Instead of using a CNN to generate a feature map for each proposed region, the whole image was instead fed through a CNN to produce a single feature map for the entire image, which is then shared among all region proposals. Then, for each object proposal a region of interest (RoI) pooling layer extracted fixed size features. They also replaced the SVM classifier with a softmax layer for object classification and added a bounding box regression layer that ran parallel to the softmax layer. Doing this allowed the network to be fully trained using back propagation. Thus, training was 9 times faster than R-CNN and because the feature map was shared with all object proposals, Fast R-CNN also ran up to 213 times faster than R-CNN at processing an image.

Even with all these advancements, Fast R-CNN was still nowhere near real-time detection, this was because Selective Search (Uijlings et al. 2013) took around two seconds to generate the object proposal regions. To overcome this bottleneck Ren et al. (2015) proposed what they creatively called Faster R-CNN. Instead of using selective search, they opted to use a fully convolutional network (FCN) to generate object region proposals, which they aptly called the Region Proposal Network (RPN). RPN shares the feature map generated by Fast R-CNN and outputs region proposals that are then used by the Fast R-CNN detector. To train Faster R-CNN they adopted a four-step alternating training method to learn the shared features of the CNN backbone. By merging RPN and Fast R-CNN into a unified network they could detect objects in an arbitrary image within 200 milliseconds with the use of a GPU.

Non-maximum suppression (NMS) is used to refine the bounding boxes by eliminating redundant or overlapping bounding boxes around detected objects. NMS compares the confidence scores associated with these bounding boxes and retains only the one with the highest score for each detected object. Boxes that exhibit significant overlap with the chosen box are discarded. This process ensures that only the most confident and non-overlapping bounding boxes remain.

2.2.2 One-stage object detection

Although two-stage detectors are generally more accurate than their one-stage counterparts, they are still not without their problems. They are computationally intensive and hence cannot run in real-time, the pipeline is complex, plus it is difficult to optimise all the components in the network.

One of the first one-stage detectors, you only look once (YOLO) was proposed by Redmon et al. (2016). YOLO uses a single CNN to simultaneously predict both localisation and classification. They removed the object proposal network typically found in two-stage detectors and instead regress a grid-based object representation of an image. Typically, YOLO divides an image into an equally spaced grid, where each cell within the grid predicts a set number of bounding boxes and confidence scores for objects whose centre falls within it. Although YOLO was a lot faster than current two-stage detectors it was not as accurate, as it suffered from localisation errors and had low recall compared to region proposal networks. To rectify this problem Redmon and Farhadi (2016) introduced YOLO9000. The main difference from their previous version is that they removed the fully connected layers and used anchor boxes. Using these modifications, they were able to improve both speed and accuracy.

Anchor boxes comprise a predefined array of bounding boxes designed to depict objects with varying shapes and dimensions. Rather than relying on a single bounding box for each object, the algorithm opts for the anchor box that most closely aligns with the object's shape and size. This approach enhances the algorithm's capacity to make precise predictions regarding object locations and dimensions within an image.

Another notable one-stage detector is the single shot detector (SSD) which was introduced by Liu et al. (2016). Similar to Faster R-CNN’s anchors, they associate a set of default bounding boxes, of different aspect ratios to each cell of a feature map. Shape offsets and class confidence scores are predicted for each default box. But unlike Faster R-CNN these default bounding boxes are applied over multiple feature map scales that are generated at the end of the base network. Non-maximum suppression (NMS) is used to predict the final detections. To train the network the authors used hard negative mining to rectify the large imbalance between positive and negative training examples. Hard negative mining is a technique that focuses on identifying and giving priority to negative examples that were initially misclassified by the model with greater confidence. These challenging negative samples are particularly valuable for refining the model. By continually introducing these complex negative instances during training, it assists the model in improving its capacity to distinguish between positive and negative cases, consequently boosting its proficiency in making precise predictions.

Lin et al. (2017) investigated why one-stage detectors are less accurate than their two-stage counterparts. They discovered that during training, the main problem is that the massive foreground-background class imbalance overwhelms cross-entropy loss. To rectify this problem, they proposed a new loss function, which they called Focal Loss.

Focal Loss extends cross-entropy loss by adding a modulating term that focuses training on a sparse set of hard examples, while well classified examples are down weighted and only play a minor role during optimisation. To evaluate the effectiveness of focal loss, they designed a fully convolutional one-stage detector which they called RetinaNet. The authors found that not only could they match the speed of one-stage detectors, but also surpass the accuracy of state-of-the-art two-stage detectors.

2.2.3 Improving detection performance

Recent literature has started to focus more on improving detection performance. Two-stage detectors like Faster R-CNN opt to use a single scale feature map with multiple sized anchors to find objects of different scales. This offered a good trade-off between speed and accuracy but came at the expense of having lower performance at detecting small objects. To overcome this problem, Lin et al. (2017) proposed a new architecture which they called the Feature Pyramid Network (FPN). FPN exploits the inherent multi-scale, bottom-up pyramidal hierarchy of deep convolutional networks to construct feature pyramids with minimal extra cost. To generate high-level semantic feature maps at all scales, coarser-resolution feature maps are up-sampled in a top-down pathway and the missing high-resolution information is added via lateral connections to the corresponding layers in the bottom-up pyramid. FPN is independent of the CNN backbone architecture and used in combination with various object detection applications has shown state-of-the-art performance in detection, segmentation and person keypoint localisation.

The traditional greedy non-maximum suppression (NMS) algorithm that has been used in face detection and object detection algorithms over the past decade, keeps the bounding box with the highest confidence score and suppresses all other boxes around it within a predetermined threshold. But this also causes problems in crowded scenes, as no matter what the threshold is set at, there will always be a trade-off between precision and recall. To overcome this problem, Bodla et al. (2017) proposed an extension to greedy NMS which they called Soft-NMS. Instead of pruning detections that are greater than the set threshold, they instead use a continuous Gaussian penalty function to decay the detection scores. Soft-NMS is a direct replacement for the standard greedy NMS function and requires no training. It has also shown consistent improvements over greedy NMS in state-of-the-art object detection algorithms, in both the Pascal VOC (Everingham et al. 2010) and Microsoft COCO (Lin et al. 2014) datasets.

CNNs that are used in object detection are inherently rigid in nature, they are initialised at start up and don’t change during their lifetime. For example, the receptive field of a convolution layer samples a feature map with a fixed size, RoIPool and RoIAlign perform pooling on regions of non-uniform size to extract fixed sized feature maps. To make CNNs more robust to geometric transformations Dai et al. (2017) introduced a new form of convolution and pooling which they named deformable convolution and deformable RoI pooling.

For deformable convolution, to enable free form deformation of the sampling grid of the standard convolutional layer, they attach a sibling convolutional layer to learn the 2D offsets for each input; these learned offsets are then added to the standard convolution sampling grid. The deformable RoI pooling module also consists of 2 parts: a regular RoI pooling layer and a fully connected layer that learns the 2D offsets of the bin positions. The bin offsets are normalised so that learning is invariant to RoI size. These learned offsets force the network to focus its attention onto the objects.

Using deformable modules have shown around a 12% relative improvement over their standard counterparts on the Microsoft Coco dataset. As they have the same inputs and outputs, they are a straightforward replacement and can easily be trained end-to-end by standard back-propagation.

2.2.4 Semantic Segmentation

The goal of semantic segmentation is to partition an arbitrary image into groups of predetermined classes, this is achieved by assigning a class label to every pixel location within the image.

Long et al. (2015) were the first to popularise the use of fully convolutional networks (FCN) for semantic segmentation. They repurposed image classification networks for the task of semantic segmentation by transforming the fully connected layers into convolutional layers. Since the spatial resolution of the feature map generated from the FCN is significantly reduced, a single deconvolutional layer is used to produce pixel wise predictions. But, because spatial information is lost in the pooling layers, the produced predictions were too coarse. To rectify this problem, skip connections are used to capture lower-level features which are upsampled and fused together with the prediction map to produce detailed segmentations.

Pooling layers help classification networks because the size of the receptive field is effectively increased. But this has a detrimental effect in segmentation because they also reduce the resolution, which is needed for dense prediction. To solve this problem Yu and Koltun (2015) used a series of dilated convolutional layers with increasing size inside what the authors call a context module, to aggregate multi-scale contextual information. Dilated convolution supports exponential expansion of the receptive field without decreasing resolution or coverage.

Recently Zhao et al. (2017) introduced the Pyramid Scene Parsing Network (PSPNet). In a similar fashion to Yu and Koltun (2015), PSPNet incorporates dilated convolutions on the ResNet architecture to extract the feature map. To capture both local and global context information, the feature map is propagated through what the authors call a pyramid pooling module. Finally, a convolutional layer is applied to the output to generate the final prediction map. To optimise the learning process a weighted Softmax auxiliary loss is added to the 4th stage of the ResNet architecture.

2.2.5 Instance Segmentation

While object detection only needs to classify and provide a tightly fitting bounding box around individual objects for localisation in an arbitrary image, instance segmentation also requires that the object within the bounding box must be accurately segmented from the background. This is an extremely challenging problem within the field of computer vision and a lot of recent research has been done to try to solve this conundrum.

He et al. (2017) proposed Mask R-CNN which extended the Faster R-CNN framework by adding a third branch to the detector stage, which predicts a binary mask for each region from the class labels. This mask layer is a fully convolutional network (FCN) that runs parallel to the classification and bounding-box layers, it also shares the feature maps that are generated from the FPN network.

When the authors first ran Mask R-CNN on the original Faster R-CNN architecture, they found that the regions that were selected by the RoIPool layer were misaligned from the regions in the original image. The reason for this is because RoIPool performs coarse spatial quantization for feature extraction. To solve this problem, they proposed RoIAlign, where in each bin, four regularly sampled locations are computed using bi-linear interpolation and then average pooling is used to obtain the final value of the bin.

To improve the information flow in proposal-based instance segmentation frameworks like Mask R-CNN, Lui et al. (2018) proposed the Path Aggregation Network (PANet) which extends Mask R-CNN in three ways. Firstly, they extended FPN by attaching a new bottom-up path with lateral connections from the existing generated FPN feature maps, this shortened the flow of information between low-layer and top-level features. Secondly, they proposed adaptive feature pooling. For each proposal, instead of using RoIAlign to extract a single feature grid from only one layer of the feature pyramid, they instead used RoIAlign to extract a feature grid from each layer. This enables them to capture both higher localisation accuracy and richer context information. The extracted feature grids are then fused together after going through the first layer in the detection and mask sub-networks. Lastly, noting that the properties of convolutional layers and fully connected layers differ from each other. FCN predicts each pixel based on the local receptive field and FC layers are location sensitive. They attach an extra branch consisting of two convolutional layers and a single fully connected layer to the mask sub-network for the purpose of predicting a class-agnostic mask. Because these two branches capture different views for each proposal, when fused together they produce better mask predictions.

2.3 Action Recognition

Inspired by the performance of convolutional neural networks (CNNs) in image classification and object detection, there has recently been a lot of research in the field of human action recognition, where a variety of architectures for tackling this problem have emerged. Action recognition involves the classification of actions within short video clips where the action may or may not be performed across the entire duration of the clip. Research based on deep learning can be divided into three main categories: two-stream convolutional neural networks, 3D convolutional neural networks and recurrent neural networks.

2.3.1 Single stream network

A problem with 2D convolutional networks is that they do not inherently model temporal information. In an attempt to rectify this problem, Karpathy et al. (2014) used a 2D CNN to investigate multiple ways to fuse temporal information from consecutive video frames. The authors investigated early, late and slow fusion methods, which are explained below and found slow fusion to perform the best. Using a model pre-trained on their sports-1M dataset the authors also ran various transfer learning experiments on the UCF-101 activity recognition dataset (Soomro et al., 2012). They experimented with training from scratch, fine-tuning the top layer, top 3 layers and all layers of the network and found fine-tuning the top 3 layers to perform the best. Training from scratch leads to abysmal performance due to massive overfitting.

- **Early fusion** combines in the first layer of the network by convolving over ten frames.
- **Late fusion** uses two 2D convolutional networks with shared parameters, spaced 15 frames apart, were the spatio-temporal features are obtained from merging the two streams in the first fully connected layer.
- **Slow fusion** is a balance between the early and late fusion methods, where fusion is done at multiple stages in a pyramid fashion.

Despite extensive experimentations, both 2D (Karpathy et al., 2014) and 3D (Tran et al., 2015) Single stream networks performed significantly worse than traditional handcrafted features such as Improved Dense Trajectories (Wang and Schmid, 2013).

2.3.2 Two-stream networks

Two-stream networks which were originally proposed by Simonyan and Zisserman (2014) modelled the motion features in the form of stacked optical flow vectors. Instead of a single network of spatial context they employed two separate networks, one for performing image classification based on static video frames and the other based on optical flow. The authors ran various experiments for input into the temporal net and found bi-directional optical flow stacked across 10 successive frames to perform the best. The two streams were trained separately and the softmax scores were combined using a multi-class linear support vector machine (SVM) (Crammer and Singer, 2002).

This approach was later extended by Feichtenhofer et al. (2016), where they explored better methods to fuse the spatial and temporal streams. They found that combining both the spatial and temporal streams in the convolutional layer to perform better than fusing at the softmax layer. They also discovered that using 3D pooling instead of 2D after the fusion layer increased performance.

As noted by Wang et al. (2016), the problem with two-stream networks is their inability to model long-range temporal structure. To overcome this problem the authors introduced the Temporal Segment Network (TSN) video-level framework. Instead of randomly sampling across the entire video, the authors suggest dividing the input video into a number of equal length segments and then randomly sampling a short snippet from each segment. Each snippet produces its own preliminary spatial and temporal prediction. The authors explored multiple strategies (max pooling, average pooling and weighted average) for final prediction at video-level and found using average pooling on the separate spatial and temporal streams to perform the best. To counter the problem of overfitting, due to the small dataset sizes, the authors established the use of various regularisation techniques such as pre-training, batch normalisation and dropout.

Girdhar et al. (2017) presented a novel trainable pooling layer named ActionVLAD that aggregates convolutional feature descriptors across both time and space. The authors also investigated various strategies for combining the appearance and motion streams. They found that having a separate ActionVLAD pooling layer at the end of each convolutional stream and then combining in the classification layer to work best.

In order to achieve real-time action recognition in two-stream networks, Zhang et al. (2018) replaced the optical flow sequence with the already encoded motion vectors of the video stream for input to the CNN. Because of noise and lack of fine details in motion vectors, directly replacing an optical flow CNN with a motion vector CNN leads to a loss in accuracy. Therefore, because optical flow and motion vector share inherent similar structures, during training the authors used the optical flow CNN as a teacher network to transfer knowledge to the motion vector CNN. During inference only the motion vector CNN is used to process the video. They were able to achieve a 20x speedup compared to the traditional two-stream approaches.

As mentioned above, the problem with motion vectors is that they are noisy and have inaccurate motion patterns, which leads to inferior accuracy compared to optical flow. But the problem with optical flow is that it is computationally expensive, which restricts action recognition from becoming real-time. To overcome these issues Zhu et al. (2018) presented a novel CNN architecture to capture motion information between adjacent frames which they named MotionNet. They used a CNN to learn optical flow in an unsupervised manner, which is then concatenated into the temporal stream. The two-stream network is then fine-tuned to predict action recognition classes. They showed similar accuracy to traditional two-stream methods (Simonyan and Zisserman, 2014; Wang et al., 2016) but were also 10x faster.

2.3.3 3D convolutional neural networks

One of the earliest attempts of using 3D convolutional neural networks for action recognition was that of Ji et al. (2010). Using stacked video frames as input to the 3D network they were able to extract features from both the spatial and temporal dimensions, thereby capturing the motion information encoded in multiple adjacent frames.

Tran et al. (2015) showed that deep 3D convolutional networks can simultaneously model both appearance and motion information and were more suitable for spatiotemporal feature learning compared to their 2D counterparts. They also investigated different kernel temporal depth settings and found that using a 3x3x3 sized convolutional kernel for all layers to give the best overall performance. Using deconvolutional layers to interpret the decisions of the network, they discovered that 3D CNNs initially focus on spatial appearance over the first few frames, then tracked the salient motion in subsequent frames.

Carreira and Zisserman (2017) proposed a two-stream inflated 3D convolutional network (I3D). They converted the image classification Inception-V1 architecture (Szegedy et al., 2014) into a spatiotemporal feature extractor by repeating 2D filters along the temporal dimension. This allowed the network to reuse 2D filters that are pre-trained on ImageNet. They also showed that pretraining on the Kinetics dataset (Kay et al., 2017) improves action recognition accuracy on different datasets.

Tran et al. (2018) showed that separating 3D convolutional filters (t, w, h where t is the number of frames and w and h are the width and height of the filter) into 2D spatial (1, w, h) and 1D temporal (t, 1,1) filters significantly increases accuracy. The advantages of separating 3D convolutions into 2D and 1D operations is two-fold: it enables the network to double the number of nonlinearities, which increases the complexity of functions that can be represented by the model and secondly, optimisation is made easier because of reduced number of parameters and the separate spatial and temporal components.

Another work similar to Tran et al. (2018) was that of Qiu et al. (2017) were they proposed Pseudo-3D Residual Net (P3D ResNet). The authors adapted the Bottleneck block design of ResNet (He et al., 2015) for video. They experimented with three different types of Spatial-temporal forms: spatial followed by temporal, spatial and temporal in parallel, the outputs of both are accumulated and spatial followed by temporal with a skip connection from the spatial convolution that is accumulated with the output of the spatial temporal sequence. The authors showed that interleaving the three blocks in sequence throughout the network gives the best performance.

One of the best-known filtering algorithms for denoising images is the non-local means filter which was proposed by Buades et al. (2005). To denoise an image the non-local means filter replaces the value of a pixel by the weighted average of all the pixel intensities in the image. The value of the weights depends on how similar these pixel neighbourhoods are to the target pixel neighbourhood, where similar neighbourhoods are given a large weighted value and neighbourhoods that are very different from the target have small weights applied to them. This allows pixels that are far away from the target (non-local) to contribute to the filtered response.

Convolution and pooling in a CNN are local operators. In order for a standard convolutional neural network to see a wider portion of an image the convolutions need to be stacked which then widens the receptive field. The problem with using this method is that it can cause optimisation difficulties and it is not computationally efficient. In order to overcome this problem for capturing long-range dependencies in CNNs, Wang, et al. (2018) proposed the Non-Local block which covers the entire area of the image. Their Non-Local block is inspired by the non-local means filter (Buades et al., 2005). The non-local block computes a response at a given position as the weighted average of the features at all locations. The non-local block can be easily applied into existing CNNs and significantly improves the performance for video classification tasks.

The current state-of-the-art in video action recognition and localisation is the SlowFast network, which was proposed by Feichtenhofer et al. (2019). The SlowFast network consists of a slow pathway with a low frame rate to capture spatial semantics and a fast pathway with a high frame rate to capture motion information at a fine temporal resolution. Both pathways operate on the same video clip and are fused together with lateral connections from the fast to the slow pathway. The SlowFast network is learned end-to-end, as it does not need to compute optical flow. Also, because the raw video is fed through the network at two different temporal rates, both pathways are able to learn their own expertise on video modelling. The authors showed that the performance of the SlowFast network can be further improved with the use of non-local blocks.

2.3.4 Recurrent neural networks

Recurrent neural networks (RNN) have been applied with incredible success in fields such as speech recognition, translation, language modelling, image captioning and video description and recognition. Given that a video is essentially a sequence of image frames it would be fair to assume that RNNs would offer a natural solution to the problem of action recognition. Below are RNNs that use RGB and optical flow as input to the network. Newer works such as that proposed by Yang et al. (2018) use skeleton based action recognition methods.

As shown by Bengio et al. (1994) vanilla RNNs do not preserve information over a long time, this is due to the vanishing gradient effect, where the gradient gets smaller with each layer until it is too small to affect the deepest layers of the network. To overcome this long-term dependency problem Hochreiter and Schmidhuber (1997) introduced the long short-term memory network (LSTM) which has become the standard module used in modern RNNs.

Ng et al. (2015) compared various convolutional temporal feature pooling architectures and using LSTM on top of CNNs. For temporal pooling the authors investigated six types of methods and found max pooling over the final convolutional layer across the video frames to perform the best. For the RNN architecture they used a deep five-layer LSTM model that is connected to the output of the underlying CNN. They showed that the LSTM model outperformed temporal pooling both on raw frames and fused with optical flow. Whereas, temporal pooling showed no noticeable gains when fused with optical flow. The authors concluded that a sophisticated sequencing architecture like LSTM is needed to take advantage of optical flow.

Donahue et al. (2016) proposed long-term recurrent convolutional networks (LRCN) to map variable-length video frames to variable length outputs. Similar to Ng et al. the authors use LSTM on top of CNNs but used end-to-end training of the entire architecture. The authors experimented with RGB and optical flow as input choices and found that RGB was better at classifying objects present in the scene, while optical flow was better with motion. Therefore, because RGB and optical flow signals are complementary to one another, the authors showed that a weighted average score based on both inputs gave the best performance.

Another hybrid network using CNNs and LSTMs was proposed by Wu et al. (2015). They used a two-stream CNN (Simonyan and Zisserman, 2014) to extract the spatial and short-term motion features from the video frames, which are then fed into their respective LSTM networks to model long-term temporal dependencies. To further improve classification, the authors proposed a regularised feature fusion neural network which captures the correlations between spatial and motion features.

2.4 Deep Learning and Animal Behaviour Recognition

The fusion of deep learning methodologies with animal behaviour analysis is marking a groundbreaking era in the field of livestock monitoring, with a distinctive impact on the study of dairy cows. These advanced technologies, predominantly convolutional neural networks (CNNs), are unlocking nuanced understandings of dairy cow behaviours, allowing for precise predictions and insights into crucial aspects such as calving. This synthesis of technology and animal science is reshaping livestock management approaches, offering more nuanced, refined, and impactful strategies specifically tailored to optimise the health and productivity of dairy cows. This section endeavours to present a detailed overview of pioneering research and innovations that leverage deep learning in decoding the complexities of behaviour recognition in dairy cows, highlighting the evolving paradigms and their transformative potential in dairy farming, especially in calving prediction and monitoring.

2.4.1 Sow and Pig Behaviour Recognition

Focusing on the mounting behaviour in pigs Li et al. (2019) introduced an efficient learning algorithm to identify the mounting behaviour of pigs based on the data characteristics of visible light images. The algorithm consists of three parts. Namely, a pig segmentation network, eigenvectors extraction and kernel-extreme learning machine (KELM) (Huang et al., 2004; Huang et al., 2006) classification. The pig segmentation network based on Mask Region-based Convolutional Neural Network (Mask R-CNN) (He et al., 2018) was used to extract individual pigs in the frames. Eigenvectors were extracted from the bounding box coordinates and mask of each pig. They extracted the perimeter and the half-body area (HBA) of each pig in the mask as well as the distance between the centre point of every bounding-box in the image as an eigenvector. Subsequently, the eigenvectors were classified with KELM which is a kind of machine learning algorithm based on a feedforward neural network, to determine whether mounting behaviour has occurred. The authors demonstrated that this method can not only effectively identify mounting behaviour, but also overcome the challenges of segmenting pigs that are partially occluded, stuck together or have similar colours to the background.

Highlighting the potential of CNNs in the livestock sector Chen et al. (2020) proposed a method to monitor the feeding behaviour of multiple pigs and measure their individual feeding times. The authors recorded video of two pens of pigs over a three-day period. The video from pen 1 was split into 70% training and 30% validation, while the video from pen 2 was used for testing. The authors used the Convolutional Neural Network Xception architecture (Chollet, 2017) to extract spatial features, which were then passed into a Long Short-term Memory (LSTM) framework to extract spatial-temporal features. A fully connected layer and softmax function were used for classification. To identify the individual pigs at feeding time, the authors used an image processing algorithm based on maximum entropy segmentation, Hue, Saturation and Value (HSV) colour space transformation, and template matching to capture the circularity of the head, the ratio of the head to the feeding sub-region, the accumulated pixels of the head motion, and the distance from the head to the number on the pig’s back. The authors found that they could recognise feeding behaviour with an accuracy of 98.4% and could correctly recognise each of the eight individual pigs 98.5% of the time.

Yang et al. (2020) showcased a framework for recognising the daily behaviours of lactating sows using both image and motion analysis techniques from still images and video.

The framework recognises drinking, feeding, nursing, moving, medium active, and inactive behaviours of three sows. They extracted spatial features such as the circularity of the head and overlapping area of the head and feeding region using a Fully Convolutional Network (FCN). They further defined the motion intensity of the head as temporal features using optical flow vectors. The spatial and temporal features were input into a hierarchical classifier for behaviour recognition. The final recognition results were obtained by a temporal-correlation-based correction module for promoting the recognition rate. They tested with over 26 hours of video from within a loose pen environment and measured the time spent per behaviour with a reliability of over 88%.

2.4.2 Dairy Cow Behaviour Recognition

The study by Li et al. (2019) tested three deep cascaded CNN models, namely a convolutional pose machine model, a stacked hourglass model and a convolutional heatmap regression model, to estimate cattle pose from RGB images that were taken under real cattle farm conditions. The cow's body was annotated with 16 keypoints. A square region with a single cow at its centre was cropped from each image and resized to 256 pixels. The authors used data augmentation techniques such as image rotation, horizontal flip and colour conversion to reduce overfitting during training. They found that the stacked hourglass model performed the best, but it could only provide rough estimations for some poses, such as lying, getting up and twisted postures.

Jiang et al. (2020) investigated how to capture the spatio-temporal structure of typical dairy cow lameness actions, which are brief and distinct in nature, and learned action representations with convolutional neural networks. However, these representations are often learned at the level of a few video frames, which fails to capture the full temporal extent of the actions. Therefore, the authors learned video representations using neural networks with single-stream long-term optical flow convolution. The experimental results demonstrate that single-stream long-term optical flow convolution network models with longer temporal extents enhance the accuracy of dairy cow lameness action recognition. The authors also explored the effects of different low-level representations, such as raw pixel values and optical flow vector fields and revealed the importance of high-quality optical flow estimation for learning precise dairy cow lameness action models. They achieved an accuracy of 98.24% on a dairy cow lameness action video set, which comprises 1080 total dairy cow videos, randomly split into 756 training and 324 test videos.

The study by Wu et al. (2020) applied the object detection model YOLOv3 (You Only Look Once, Version 3) (Redmon and Farhadi, 2018) to obtain the leg region coordinates of the cow in each video frame. They then computed the step size of the front and rear legs of the cow based on the leg coordinates and built a relative step size feature vector. The authors tested Long Short-Term Memory (LSTM), support vector machine (SVM), K-Nearest Neighbour (KNN) and decision tree classifier (DTC) algorithms and found that LSTM achieved the best accuracy with a score of 98.57% in detecting lameness (2.93% higher than its closest competitor SVM). They also showed that a bidirectional LSTM performed slightly better but required more hardware resources. Moreover, the authors showed that a pure deep learning method performed slightly better than LSTM but was less suitable for interpretation and diagnosis of lameness.

The study by Achour et al. (2020) applied four CNN modules for recognising feeding behaviour and identifying individual cows. The first CNN module detected whether a cow was present in the feeder zone or not. The second CNN module determined the state of the cow (standing or feeding). The third CNN module assessed the availability and category (5 types) of food in the feeder. The fourth CNN module was devoted to individual identification of the dairy cow. A fixed camera looking down into the feeding zone was used to obtain 7265 images of seventeen Holstein dairy cows from a commercial farm over 5 days during feeding times. Their test results achieved 100% accuracy in food detection and categorisation, 92% accuracy in state classification and 97% accuracy in individual identification. The authors investigated three different methods for identifying individual cows: CNN4, Hybrid CNN-SVM and Multiple CNN classification. They found that Multiple CNN classification performed the best as it used extra features for cows that were entirely black or white, such as ear shape or horn presence.

2.4.3 Multiple Behaviour Recognition in Dairy Cows

Recently deep learning has been used for recognition of multiple behaviours in cows. Fuentes et al. (2020) were among the first to apply deep learning to animal behaviour recognition. They proposed a method for hierarchical cattle behaviour recognition using spatio-temporal information. Their model consists of three parts, a frame-level detector to generate the ROIs, which are then used to extract temporal-context features (3D-CNN) and motion information (optical flow). They demonstrated that the system could recognise 15 different types of hierarchical activities that are divided into 3 groups: 1. individual activities (walking, standing, resting, eating, sleeping, standing up, lying down and self-grooming); 2. group activities (fighting, feeding, social licking and mounting); 3. parts actions (moving head, ruminating and tail swish).

Yin et al. (2020) developed an EfficientNet-LSTM model to track the motion behaviour of a single cow. The authors analysed five types of motion behaviours, namely walking, standing, lying, feeding and drinking. First, they used EfficientNet-B0 (Tan et al., 2020) to extract the spatial features. Then, they used a bidirectional feature pyramid network (BiFPN) and a fusion layer to fully capture the characteristics of different behaviour information. Finally, they sent the cow behaviour feature information to the bidirectional long short-term memory (BiLSTM) module (Graves and Schmidhuber, 2005), which incorporated the attention mechanism, to achieve fast and accurate recognition of individual cow motion behaviour. The authors applied a sliding window mechanism where they continuously sampled 60 frames of a video sequence to both predict and locate behaviours. They used a step of 30 frames to ensure 50% of the sequence was repeated in each sample. They demonstrated that their model had a recognition accuracy of 95.2% in undivided long behaviour videos.

Wu et al. (2021) proposed the fusion of a convolutional neural network and long short-term memory (CNN-LSTM) to recognise the basic behaviours of a single cow (walking, standing, lying, ruminating and drinking). Firstly, they opted to use VGG-16 (Simonyan and Zisserman, 2015) trained on ImageNet (Deng et al., 2009) as the network backbone to extract the feature vector sequence corresponding to each video that was collected from a low-quality monitoring camera in the cattle farm. Then, the extracted features are passed through a Bi-LSTM classification model to extract semantic information of the time series data in two directions. The results of which was finally passed through a fully connected layer and a Softmax layer is used to predict results. Their dataset comprised of a total of 4566 videos which was captured from the outdoor exercise area of the Keyuan Dairy Farm. The duration of each video was between 10 to 55 seconds. Each video had only one behaviour label. The dataset was split 70% training and 30% testing. For the 1370 test videos, the average recognition accuracy of the proposed algorithm for the five behaviours was 97.59%. To verify the effectiveness of the VGG16 feature extraction network, the authors experimented with five different feature extraction networks, namely: VGG-19, ResNet-18, ResNet-101, MobileNet V2 and DenseNet-201, (Simonyan and Zisserman, 2015; He et al., 2015; Sandler et al., 2019; Huang et al., 2018) which all gave lower average recognition scores of 97.51%, 95.62%, 95.40%, 94.74% and 95.25%, respectively. As stated by the authors the dataset consists of class imbalance, where some classes may be up to dozens of times that of other classes. This data imbalance was not considered in their study.

2.4.4 Calving prediction

Fenlon et al. (2017) developed four models to estimate the level of assistance needed for calving events (no assistance, slight assistance, or veterinary assistance). The authors used data from 2,076 calving events from 10 dairy farms, where 19.9% of the events required slight assistance and 5.9% required veterinary assistance. The models were based on four machine learning techniques: multinomial regression, decision trees, random forests, and neural networks. The neural network and multinomial regression models had the highest accuracy, predicting 75% of the events correctly, with errors of 3.7% and 4.5% in the predicted probabilities, respectively.

Borchers et al. (2017) conducted a study to estimate calving events using machine learning techniques. The authors evaluated three techniques: random forest, linear discriminant analysis, and neural network. They used data on the number of steps, lying time, standing time, lying bouts, and total motion to create alerts eight hours before calving. The neural network achieved the highest accuracy, with a specificity and sensitivity of 80.4% and 82.8%, respectively, when including rumination data. Without rumination data, the specificity and sensitivity were 83.8% and 79.2%, respectively. These findings suggest that machine learning can be effective for calving estimation. However, the sample size was limited to 53 events, so a larger test dataset is required. The authors also found that rumination time and lying time decreased gradually during the prepartum period (days before birth) and reached the lowest levels on the day before calving.

Fadul et al. (2017) used a combination of data obtained from the RumiWatch noseband-sensor which monitors several variables of ingestive behaviour (Zehner et al., 2012) and 3D-accelerometer to predict the calving time in Holstein-Friesian cows. The authors fitted the cows with the sensors 10 days before the expected calving day and used two models to predict the calving time, one for primiparous cows (first calving) and the other for multiparous cows (given birth more than once). Their study showed that lying bouts increased and rumination decreased similarly in both groups, and that there was a clear connection between cow behaviour and the onset of calving.

Rutten et al. (2017) used a sensor in an ear tag to record cumulative activity, rumination activity, feeding activity, and temperature on an hourly basis over 24 hours before calving. 417 calvings were documented using camera images taken at 5-minute intervals, with 114 calving moments linked to sensor data. Two logit models were formulated: one relying solely on the expected calving date and another incorporating additional sensor data. The latter model proved more effective, with a sensitivity rate of 36.4% at a 1% false positive rate, compared to the former's 9.1% sensitivity. Furthermore, at a 1% false positive rate, the combined model had a sensitivity of 21.2% for a one-hour window and 42.4% for a three-hour window, highlighting the challenge of pinpointing the exact hour of calving onset. Nonetheless, the sensor data significantly enhanced the accuracy of predicting the calving's commencement over predictions solely based on expected calving dates. This offers farmers an invaluable tool, suggesting closer supervision of cows in imminent labour.

Zehner et al. (2018) employed the RumiWatch noseband sensor (Zehner et al., 2012) to monitor the ingestive behaviour of 35 dairy cows and to formulate a predictive model for calving time using a Naïve Bayes classifier, a probabilistic machine learning model based on Bayes' theorem with strong independence assumptions. Data from three farms were used for model training and validation, 11 cows from farm 1 were used for training and 11 and 13 cows from farms 2 and 3 were used in two validation sets. The model's performance was gauged on an hourly basis for 168 hours leading up to calving. Although some sensor variables, especially those related to rumination behaviour, showed promising sensitivity and specificity values, the model was undermined by its low positive predictive value and high false positives. As such, even with satisfactory sensitivity and specificity, the authors showed that the model is not practical for real-world use. This highlights the need for comprehensive evaluations, as relying solely on sensitivity and specificity can be deceptive regarding a model's actual utility. The authors concluded that future research on calving detection should consider the promising predictive value of rumination behaviour.

Zaborski et al. (2019) conducted a study to compare the predictive performance of random forest and boosted trees for dystocia detection in dairy heifers (young female cows less than 2 years old that have never calved) and cows. Calf sex, calving age, calving season, gestation length, and sire breed were used as features for heifer calving events, while two additional features were used for cows, namely, previous calving difficulty, and lactation number. The authors used several datasets with varying proportions of easy and difficult calvings and found that boosted trees had better sensitivity than random forest when trained on sets with an increasing percentage of easy calvings. However, overall, they had a lower accuracy. They showed that boosted trees had higher accuracy in detecting difficult calvings but also generated a high number of false alarms. In all datasets boosted trees performed better at detecting dystocia but the overall accuracy was lower than random forests. This was due to random forests greater ability to correctly predict easy calvings. The authors showed that the most important predictors of calving difficulty were calving age, gestation length and previous calving difficulty. None of the models investigated in the study was good enough for practical application under field conditions. It can be also noted in a previous study Zaborski et al. (2017) used random forest for dystocia detection and was able to predict a high percentage of difficult calving events but was unable to detect dystocia in cows.

Keceli et al. (2020) showed that calving can be predicted by applying several behaviours of cattle, behavioural monitoring sensors, and machine learning models. The authors used the same dataset that was collected by Borchers et al. (2017) which consists of 53 cattle. The Bi-directional Long Short-Term Memory (Bi-LSTM) method (Graves and Schmidhuber, 2005) was applied to predict the calving day and was found to outperform the standard LSTM in classification accuracy. Additionally, the RusBoosted Tree classifier (Seiffert et al., 2010), which is an ensemble learning algorithm was effectively employed to predict the final 8 hours leading up to calving. The authors showed slightly better results for the day before calving using Bi-LSTM compared to Borchers et al. (2017).

2.4.5 Challenges and Improvements in Animal Behaviour Recognition

Qiao et al. (2019) proposed a four-step instance segmentation approach based on Mask R-CNN to address the challenges of cattle segmentation and contour extraction in a real feedlot environment. The four steps are: key frame extraction for detecting huge cattle motion frames, image enhancement to reduce illumination and shadow influence, cattle segmentation, and body contour extraction. The authors claim that their proposed framework outperformed the state-of-the-art SharpMask and DeepMask instance segmentation methods.

Tu et al. (2020) conducted a study on Mask Scoring R-CNN (Huang et al., 2019), which is an adaptation of the Mask R-CNN framework (He et al., 2017), to improve instance segmentation performance for grouped-housed pigs from front-view and top-view images. The authors showed that Mask Scoring R-CNN can robustly detect and segment multiple target pigs under group-housed pig natural scenes such as pigs overlapping, touching, occlusion and under uneven lighting conditions. The improvement in instance segmentation performance is achieved by adding a MASKIoU head that learns the quality of the predicted instance masks and feeds this information back into the network during training.

2.4.6 Gaps in Literature

While there exists substantial research on the utilisation of deep learning models in animal behaviour recognition, literature particularly focusing on dairy cows and aspects such as calving presents noticeable gaps.

Diversity and Representativity in Datasets: There is a substantial lack of diverse and representative datasets involving various breeds of dairy cows under a range of environmental and farming conditions, affecting the universality and adaptability of developed models in real-world scenarios, especially for predicting and monitoring calving events.

Comparative Studies on Model Efficacy: The current body of literature is deficient in studies that offer comparative analyses of different models under identical conditions, obscuring insights into which models are most effective for diverse behaviours and conditions inherent to dairy cows and calving.

Inherent Biases and Data Imbalances: There is a dearth of exhaustive studies addressing the ramifications of inherent biases and imbalances in datasets on the reliability and accuracy of models related to dairy cow behaviours and calving predictions.

Translational Research Deficiency: Practical applicability of models in diverse and unstructured dairy farm environments is largely unexplored, underscoring the need for more translational research to bridge the gap between theoretical advancements and their practical implementations in the field, especially in the domain of calving.

Calving-specific Insights: More refined research is needed to hone in on the intricate behaviours and signs exhibited by dairy cows during calving, as the literature is relatively scant in providing precise and detailed insights in this specific context, impacting the advancement of more specialised models.

2.4.7 Conclusion

The advent of deep learning technologies in animal behaviour analysis represents a revolutionary leap, especially in the nuanced realm of dairy cow behaviours and calving. By harnessing advanced computational models such as convolutional neural networks (CNNs), the field has been endowed with the capability to glean refined insights and make accurate predictions, potentially transforming the protocols for managing dairy livestock. This burgeoning synergy between technology and animal science is pivotal in establishing novel, precision-driven approaches in dairy livestock management, ensuring optimal health, productivity, and welfare of dairy cows. While the breakthroughs achieved are monumental, addressing the identified gaps in literature is crucial for pushing the boundaries of current knowledge, refining predictive models, and ensuring their seamless integration and adaptability in diverse, real-world dairy farming environments. The pursuit of these refinements and advancements is integral in elevating the standards of dairy farming, ensuring the welfare of the animals, and contributing to sustainable and ethical farming practices, particularly in the crucial area of calving.

Chapter 3

3 Developing an Annotated Video Dataset for Classifying Cow Behaviour

3.1 Introduction

In the convergence of scientific innovation, urgent necessity, and consequential impacts on animal health and welfare, the field of animal behaviour research stands out as a pivotal domain. However, there exists a notable under-exploration in terms of developing annotated video datasets within this field. Much of the existing research has predominantly relied on the use of accelerometer-based activity monitoring systems, as demonstrated by studies such as those by Diosdado et al. (2015), Rahman et al. (2018), and Benaissa et al. (2019). Recognising this significant gap in research methodologies—especially the conspicuous lack of annotated video datasets in animal behaviour research—this chapter introduces a novel, large-scale video dataset meticulously designed for the classification of cow behaviour. The inception of this dataset is driven by the conspicuous absence of any comparable resources that allow a detailed exploration of cow behaviours through videos.

3.2 Video acquisition

High-definition video cameras (5 Mp, 30 m IR, Hikvision HD Bullet; Hangzhou, China) were strategically positioned to record the behaviours of Holstein–Friesian dairy cows at the Nottingham University Dairy Centre (Sutton Bonington, Leicestershire, UK) in the period leading up to calving. Operating at 20 frames per second and with a resolution of 640x360 pixels, the cameras were set up to capture detailed observations of the cows' movements and behaviours.

To ensure comprehensive coverage, each of the three calving pens were equipped with two surveillance cameras. This setup secured 24-hour footage of 46 individual cows from April to June 2018. Each camera was angled at approximately 45 degrees and positioned at a height of 4 meters to ensure full coverage of the 10 m × 7 m area within each pen. Every pen was designed to accommodate a maximum of eight cows.

Several days before the anticipated calving, cows were relocated to one of the three dedicated calving pens for close monitoring of the entire calving process. Initially, six cameras were installed, but a decision was made to use just one camera from each pen, reducing the number to three. This modification was necessary due to the significant similarity in the video footage obtained from the multiple cameras. The adjustment allowed for a more streamlined and focused approach to data acquisition.

3.3 Graphical User Interface (GUI) Development

A robust and intuitively designed behaviour annotation tool was developed using Python 3.6 and OpenCV 3. Python was chosen as the preferred programming language due to its user-friendly nature, extensive libraries, and compatibility across various platforms including Windows, iOS, and Linux. OpenCV was selected for its proficiency in managing real-time scrolling through a video with the help of the incorporated navigation bar.

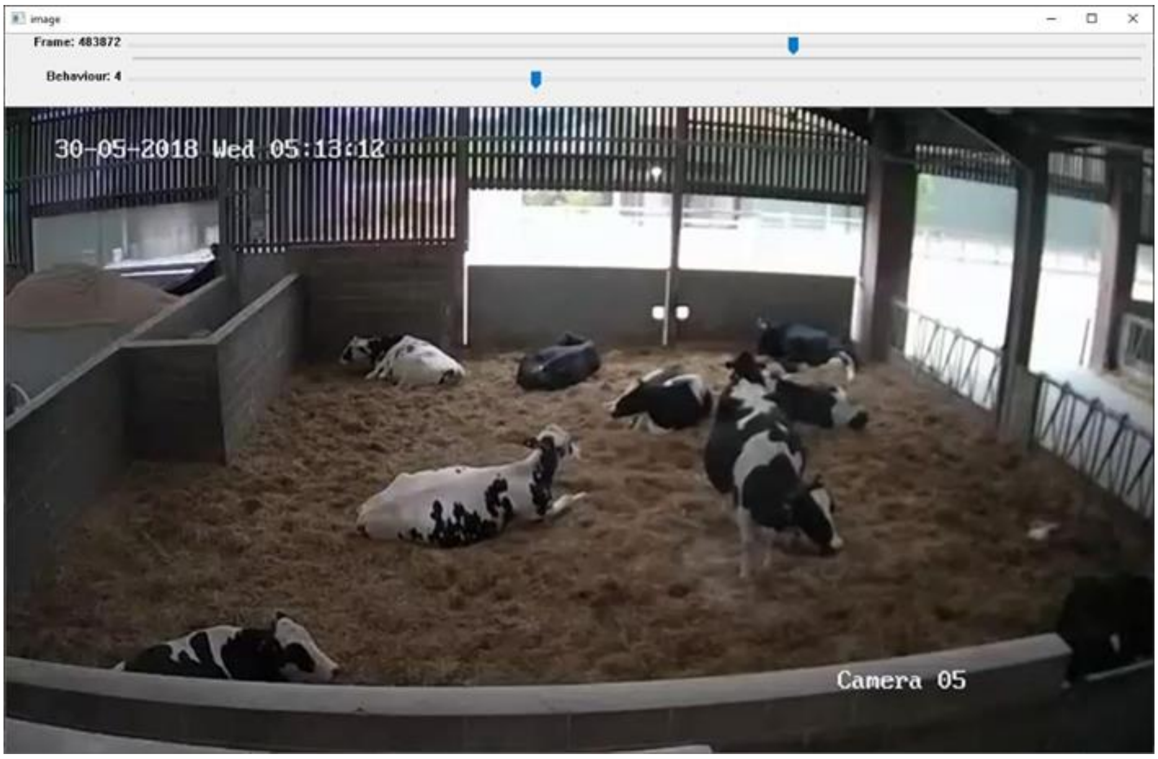

Figure 3.1 The Developed Graphical User Interface (GUI) of the behaviour annotation tool, showcasing the video display window, the navigation bar, and the behaviour display bar.

The developed GUI, as shown in Figure 3.1, is divided into three key sections. These are a video display window, a navigation bar, and a behaviour display bar. The navigation bar shows the current frame and allows for user-friendly scrolling through the entire video, while the behaviour display bar dynamically displays the behaviour of the cow in the current frame.

In addition to the navigation bar, the interface also allows more nuanced navigation through keyboard commands, offering two different speed choices to accommodate the precision needs of users in both forward and rewind directions. For instance, the left/right arrow keys facilitate one-second interval updates, and the up/down keys enable frame-by-frame updates. A prolonged key press repeats the action. The interface accommodates seven keys assigned for behaviours and two keys to designate the start and end frames of the behaviour, addressing instances where cows are not visible in the video due to blind spots, ensuring gaps in the annotated file.

Initially, we envisaged using Mask R-CNN (He et al., 2017) for cow detection and SORT (Bewley et al., 2016) for temporal tracking. However, due to inconsistencies and errors over extensive time frames, we pivoted towards manual tracking of cows giving birth. We leveraged the recorded time of birth in the video filename to pinpoint the exact time using the navigation bar. This allowed us to take screenshots of the cow from dual perspectives. These screenshots served as references to identify and annotate the cow at the start of the video.

3.4 Behavioural Annotation

The video recordings of each of the 46 cows that gave birth underwent meticulous annotation, spanning a timeframe from 10 hours before to 5 hours after calving. This meticulous process, conducted by three observers, resulted in a total of 19,191 individual behaviour observations from 690 hours of video. The behaviours recorded were classified into seven distinct categories as depicted in Table 3.1. The original dataset included two additional behaviours: standing contractions and birth. However, these were removed from the dataset due to limited data and poor performance, as discussed in Chapter 5.3.7.5.

Table 3.1 Classification of Recorded Behaviours: This table outlines the seven distinct categories of cow behaviours observed and annotated in this thesis, each labelled and described according to the specific actions or states exhibited by the cows.

Label	Behaviour	Description
1	Stand	The cow is still on all four legs
2	Lie	The midway transition of when the cow is about to lie down to when it starts to rise again
3	Walk	Movement of more than two steps
4	Shuffle	Cow circles on the spot or moves slightly with a step or two
5	Contractions	Visible straining while lying down
6	Eating	Cow puts its head through the feeding barrier until the moment it pulls

		its head back out from the feeding barrier
7	Drinking	Head is over the water trough and regular head movement towards the trough

For systematic storage and easy retrieval, the annotated data for each cow was catalogued in individual CSV files. The adopted format for these CSV files, representing the start frame, end frame, and behaviour, is illustrated in Table 3.2.

Table 3.2 Format of Annotated Data Storage: This table shows the adopted format for storing the annotated data in individual CSV files, representing the sequential arrangement of start frames, end frames, and the corresponding behaviour labels identified during the annotation process..

Start Frame	End Frame	Behaviour Label
553602	556724	7
556725	557555	2
557556	557697	4
557698	580880	1
580881	581004	4
581005	581077	1
581078	581157	4

3.5 Temporal segmentation

Utilising a bespoke Python script, we methodically segmented each video. For each saved CSV file, behaviours were categorically separated into individual lists and arranged in descending order based on their duration, calculated by subtracting the start frame from the end frame. Behaviours spanning fewer than 64 frames were excluded since our non-local network has a threshold capacity to read 64 frames simultaneously. Notably, a single contraction typically lasts around three seconds and might be recurrent or interspersed with brief pauses ranging between two to five seconds. To focus on genuine contractions and eliminate sequences with gaps, we organised these in ascending order based on their durations.

Subsequently, as depicted in step 3 of Figure 3.2, we performed random sampling to extract ten-second segments (equivalent to 200 frames) from behaviours whose durations surpassed ten seconds. This duration was chosen to enable comprehensive behaviour identification, especially considering cows predominantly exhibit minimal movement during prolonged activities such as standing or lying.

To refine our dataset further, we retained the top 60 clips from each behaviour list. This was especially relevant for contractions where shorter clips were more informative. Ensuring a well-distributed sample pool across the fifteen-hour period, fifteen samples were uniformly extracted from each behaviour list, as shown in step 4 of Figure 3.2. This sampling restriction was implemented in acknowledgment of the cows' limited activity range, ensuring the exclusion of overly similar video segments.

Finally, each video was allotted a unique directory, encompassing seven sub-directories, each attributed to a specific behaviour. Using the computed start and end frames, videos were further segmented, and relevant segments were stored in their respective directories. The file names, embedding the date, time, camera identifier, along with the start and end frames, contributed to an organised and navigable dataset.

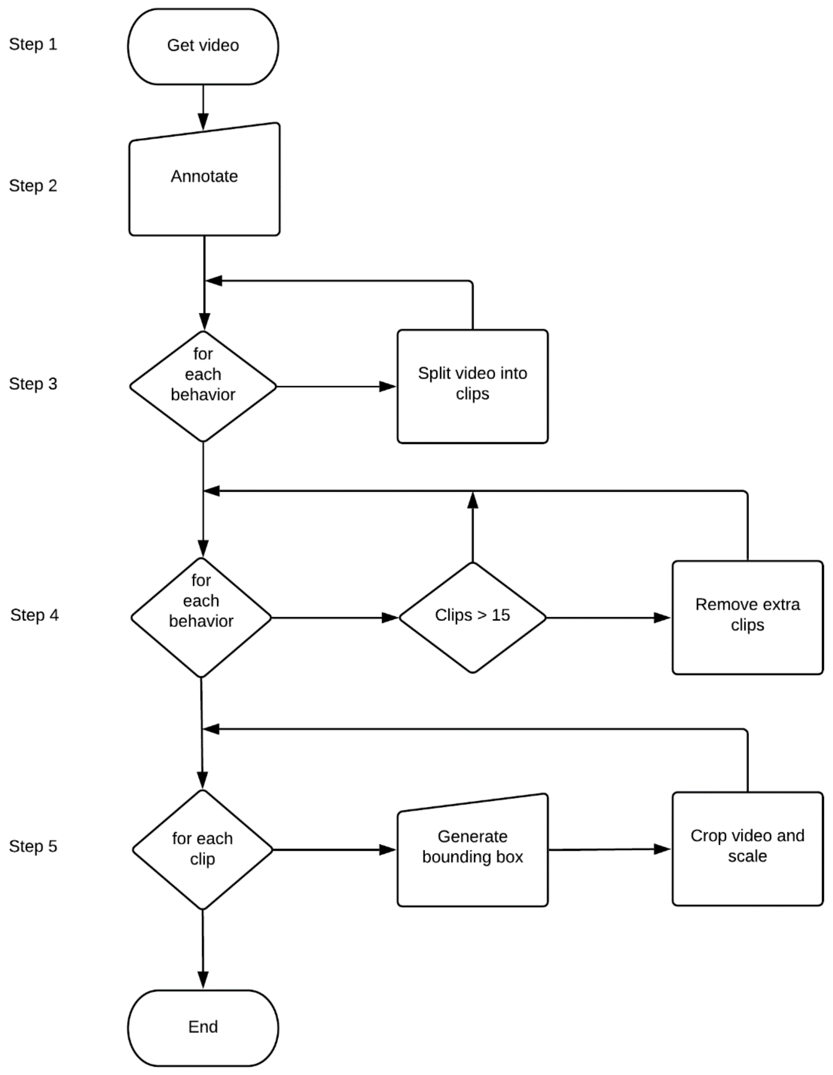

Figure 3.2 High-Level Overview of the Annotation Process: 1: Acquiring video. 2: Annotating said video. 3: Splitting the videos into 3 to 10 second segments. 4: Removing excess video clips. 5: Generating fixed size bounding and cropping and scaling video so its shorter side is 256 pixels.

3.6 Spatial Segmentation

Aligning with the protocols of the non-local network outlined by Wang et al. (2018), we adopted the use of a fixed-size bounding box designed to encapsulate the entirety of each cow that gives birth across all frames. This methodology replicates Wang et al.'s (2018) technique, wherein the complete frame was incorporated.

Explaining the final 5th step of Figure 3.2, given the now-condensed video segments, Mask R-CNN was deployed for the detection of cows, and SORT for their subsequent tracking. We then determined the maximum width and height of the bounding boxes corresponding to the annotated cow and resized all boxes from their central point. Any discrepancies or inaccuracies detected were corrected using the image annotation tool, ViTBAT (Biresaw et al., 2016).

The dimensions of the fixed-size bounding box were confined to the dimensions of the image as illustrated in Figure 3.3, avoiding any clipping at the frame's boundary. This allowed the seamless creation of mp4 videos from the list of conserved bounding boxes.

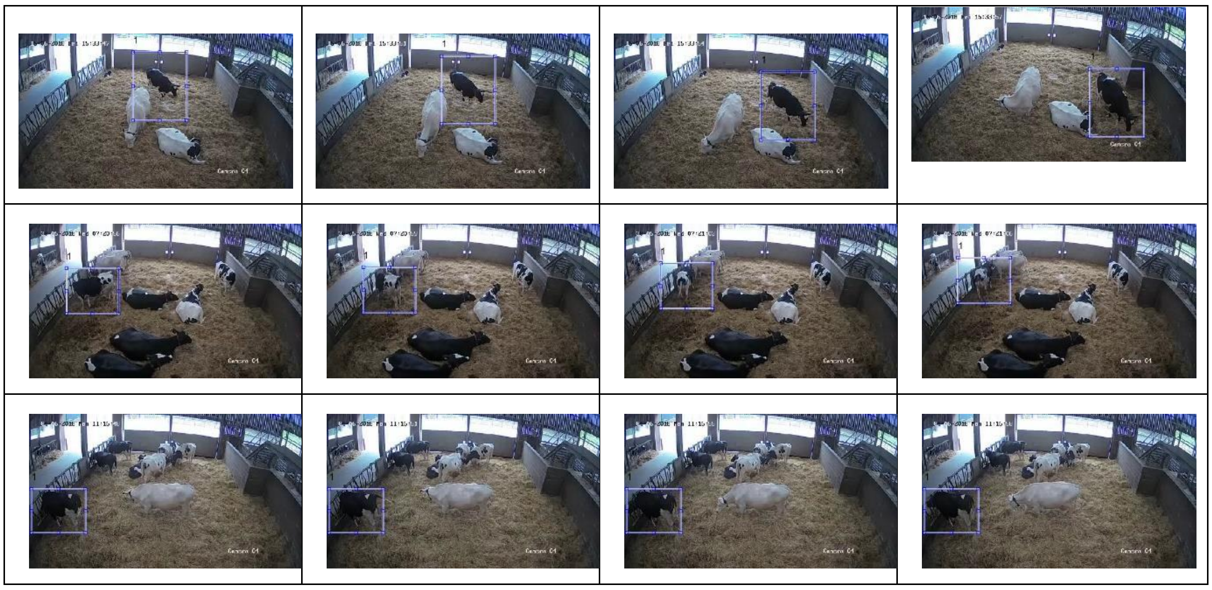

Figure 3.3 Example annotated behaviours before scaling and cropping. The behaviour of the top row is of a cow walking. Middle row shows a cow shuffling and bottom row is of a cow eating. There is a stride of 50 frames between each displayed image. The size of the bounding box does not change for the entire sequence.

The newly generated bounding boxes for the cow due to give birth were used to crop and scale each frame of the video clip, ensuring the smaller side measured 256 pixels, as illustrated in Figure 3.4. Given that the cropped images predominantly underwent upward scaling, we preferred bicubic interpolation over bilinear interpolation because it more effectively reduces artefacts and enhances image quality. The refined videos were then compressed using the H.264 codec. Inspired by the Kinetic dataset, ground truth labels were archived in a text file and systematically paired with their corresponding video filenames.

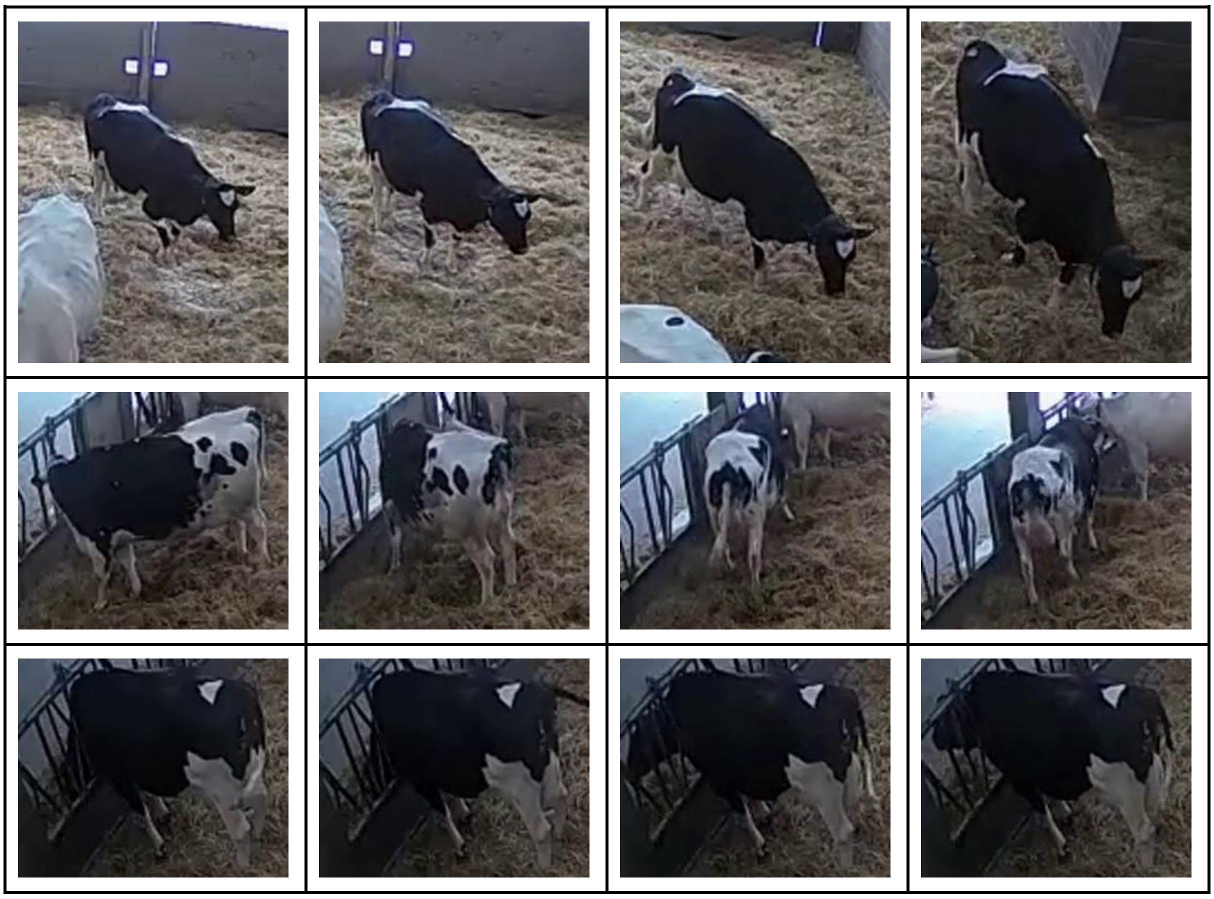
Figure 3.4 Example of cropped and scaled videos, where the smaller side is 256 pixels. Top row shows a cow walking. Middle row shows a cow shuffling and bottom row is of a cow eating. There is a stride of 50 frames between each displayed image.

3.7 Statistics

The dataset created focuses on seven classes of cow behaviour. For each behaviour, there are between 248 and 686 video clips. Each unique video in the dataset contains up to 15 clips per behaviour, with each clip lasting between 3 and 10 seconds.

The current version of the dataset hosts a total of 3,969 videos. These are further divided into two subsets: a training subset and a validation subset. The training subset comprises 3,186 videos, with 205 to 552 videos per class. The validation subset consists of 783 videos, with each class represented by 43 to 135 videos. To ensure no overlap between the subsets, they are distinct at the video level. This ensures that all clips from a single video are confined to either the training or the validation subset, but not both.

In terms of duration, the dataset represents 9 hours and 42 minutes of captured video data. This is split into approximately 7 hours and 48 minutes for training and 1 hour and 54 minutes for validation.

Below is a comprehensive breakdown of the clip count for each behaviour class in both the training and validation datasets, represented in Table 3.3.

Table 3.3 Distribution of Video Clips by Behaviour Class: This table showcases the allocation of video clips for each identified behaviour class within the training and validation subsets. It offers a clear view of the distribution and total number of clips available per behaviour, providing insight into the dataset's composition.

Label	Behaviour	Training	Validation	Total
1	Stand	552	134	686
2	Lie	522	135	657
3	Walk	496	134	630
4	Shuffle	518	134	652
5	Contractions	501	112	613
6	Eating	392	91	483
7	Drinking	205	43	248
	Totals	3186	783	3969

3.8 Challenges

The development of behaviour recognition through continuous camera surveillance in farm environments presents numerous challenges. This thesis has identified several potential sources of error in computer model predictions, highlighting the limitations of current vision-based monitoring, as detailed in Table 3.4 and illustrated in Figure 3.5.

Table 3.4 Potential causes of error in animal vision-based model predictions.

Problem	Cause of Error
Pose	A cow's pose changes not only in terms of its current behaviour, but also in terms of the direction it is facing from the camera. As a cow is a quadruped, this forces the model to have a much higher generalisation capability when compared to bipeds such as humans.
Similarity	Distinguishing between two or more cows is a very difficult task even for humans. This is because cows can often have similar colours or patch patterns on their bodies.
Occlusion	Parts of a cow can be hidden if behind other cows, such as when all bunched up while eating. The birth of the calf can also be occluded if the cow is facing towards the surveillance camera. Cows can also be partially hidden under bedding. Cows can even have self-occlusion, where the cow's body blocks the view to other parts such as the head. Spider webs can also blur/occlude cows while the camera is in infrared night vision mode.
Lighting	Natural light comes through the ventilation spaces, which can produce rectangular patches over the enclosure and on the cows. Over the course of the day, the brightness of the enclosure changes. In the evening artificial lighting is used, which gives an orange tint to the enclosure. Infrared night vision is used during night-time, which turns the video footage into black and white. While the camera is in night vision mode, it focuses on the centre of the pen and loses focus towards

	the extremities of the enclosure. Night vision also casts deep shadows off the cows that may confuse object detection.

3.9 Conclusion

This chapter has presented a comprehensive approach to developing a classified annotated video dataset for cow behaviour. The process of developing the annotated video dataset involved strategic video acquisition, development of a user-friendly graphical interface, and meticulous video segmentation. The use of high-definition cameras ensured detailed data capture. Additionally, the deployment of Python and OpenCV for GUI development, coupled with a custom Python script, facilitated precise video segmentation. The pivot towards manual tracking of cows due to give birth, while initially unplanned, proved to be a necessary methodological adjustment that allowed for more focused data acquisition. The resulting dataset, with its detailed annotations and careful segmentation, provides a valuable resource for further research in animal behaviour analysis, particularly in the context of dairy cows during calving. Additionally, it is particularly noteworthy that this dataset has not only filled a significant void in existing resources but has also played a crucial role in foundational studies by Cavendish et al. (2021) and McDonagh et al. (2021).

			Came	

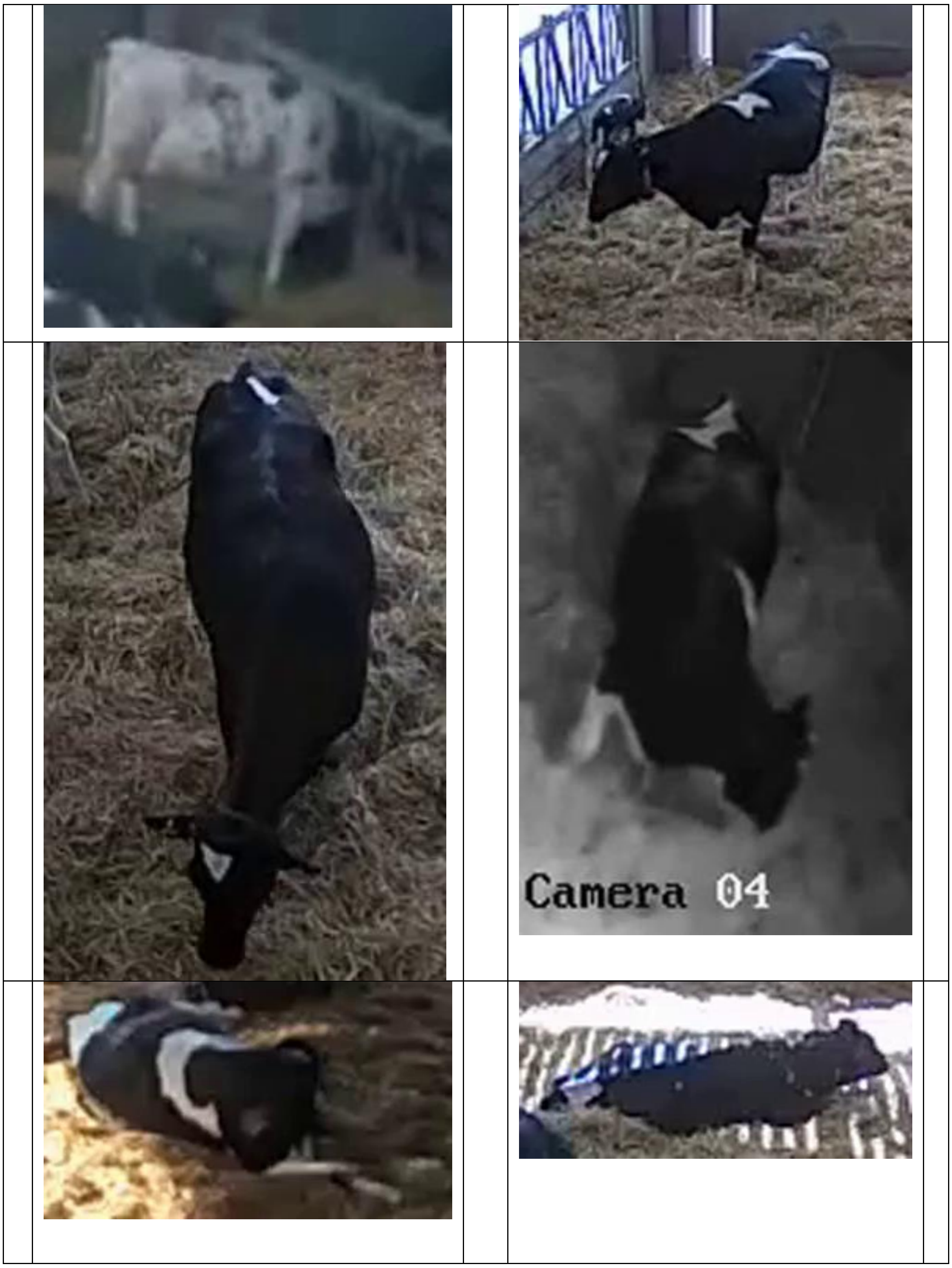

Figure 3.5 Various images from the training set illustrating the potential causes of error in animal vision-based model predictions: Pose, Lighting, Similarity, and Occlusion.

Chapter 4

4 Changes in cow behaviour during parturition

4.1 Summary

This chapter quantifies the changes in key cow behaviours observed prior to calving, focusing on behaviours such as standing, lying, walking, shuffling, eating, drinking, and contractions, recorded for each cow. The analyses are based on comma-separated value (CSV) files generated by video annotation, as outlined in Chapter 3.4. These files were used to determine the duration and frequency of behaviours for each individual cow that had complete 24-hour observations during the 9-hour period studied before giving birth. It is important to note that this analysis represents a subset of the full dataset, which contains annotations for 46 cows; however, only data for 35 cows were available at the time of this study by Cavendish et al. (2021).

4.2 Introduction

Determining the precise timing of parturition in cows is crucial, not only for the welfare of the cows but also for optimal farm management. Timely interventions can minimise calving complications, ensuring the safety of both the calf and the cow, and facilitating optimal lactation and reproductive performance post-calving. This chapter provides an extensive investigation into the behavioural modifications in cows during parturition, contributing to our understanding of how these predictable changes could be harnessed for practical applications, particularly in designing vision-based monitoring tools to precisely predict calving time.

In order to develop a comprehensive understanding, we examined a range of behaviours exhibited by cows in the lead-up to calving, such as standing, lying, walking, shuffling, eating, drinking, and contractions. Utilising CSV annotations as described in Chapter 3.4, we quantified the duration and frequency of each behaviour from a detailed examination of 35 cows, providing complete 24-hour observations for the 9-hour period immediately preceding parturition.

4.3 Objective

While earlier studies have explored predicting calving time by focusing predominantly on alterations in cow behaviour during parturition, the degree of success has been varied. Thus, there exists a need for a more nuanced and detailed understanding of the behavioural changes occurring in the pre-calving period. This thesis aims to characterise the behavioural patterns associated with calving in dairy cows by examining both the frequency and duration of specific behaviours, intending to enhance the effectiveness and reliability of vision-based monitoring systems for calving.

4.4 Method

Approval for this study was obtained from the University of Nottingham animal ethics committee before commencement of the study (approval number 198).

4.4.1 Data

Leveraging the comprehensive behavioural annotations discussed in Chapter 3.4, text files for 19,191 annotated behavioural observations were obtained for 35 cows that had complete behavioural observation from 9-hours prior to giving birth. The start of the continuous observation period was determined as 9-hours from when the calf was fully expelled at birth, using the video recording, and considered a time when no visual signs of calving behaviour were observed. Documented behaviours within this timeframe include standing, lying, walking, shuffling, experiencing contractions while lying, eating, and drinking.

4.4.2 Analysis

For the analysis, the 9-hour period prior to giving birth was split into three-hour time periods, with period three ending with the birth. The duration of behaviours in seconds and frequency were determined for each time period. A total of 735 behaviour records were obtained from 35 cows (35 × 7 behaviours × 3 time periods) and the mean for each cow determined for the analysis.

4.5 Results

Of the 35 calvings, 23% of calvings required assistance by farm staff, with all other calvings being unassisted. Differences were found in the duration of behaviour with the majority of time spent lying (0.49) or standing (0.35) with other behaviours being 0.04 or less across the 9-hour period (Figure 4.1). In the final three hours prior to calving, the proportion of time for lying and contractions noticeably increased, and the time spent standing, drinking and eating decreased (Figure 4.2).

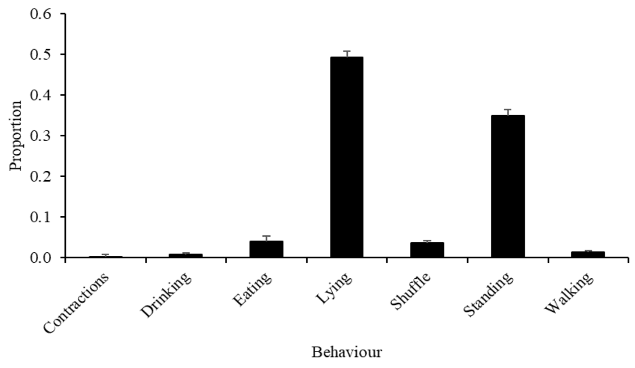

Figure 4.1. Mean (± s.e.) proportion of time dairy cows (n = 35) spent doing different behaviours during the 9-hours prior to calving (Source: Cavendish et al. (2021)).

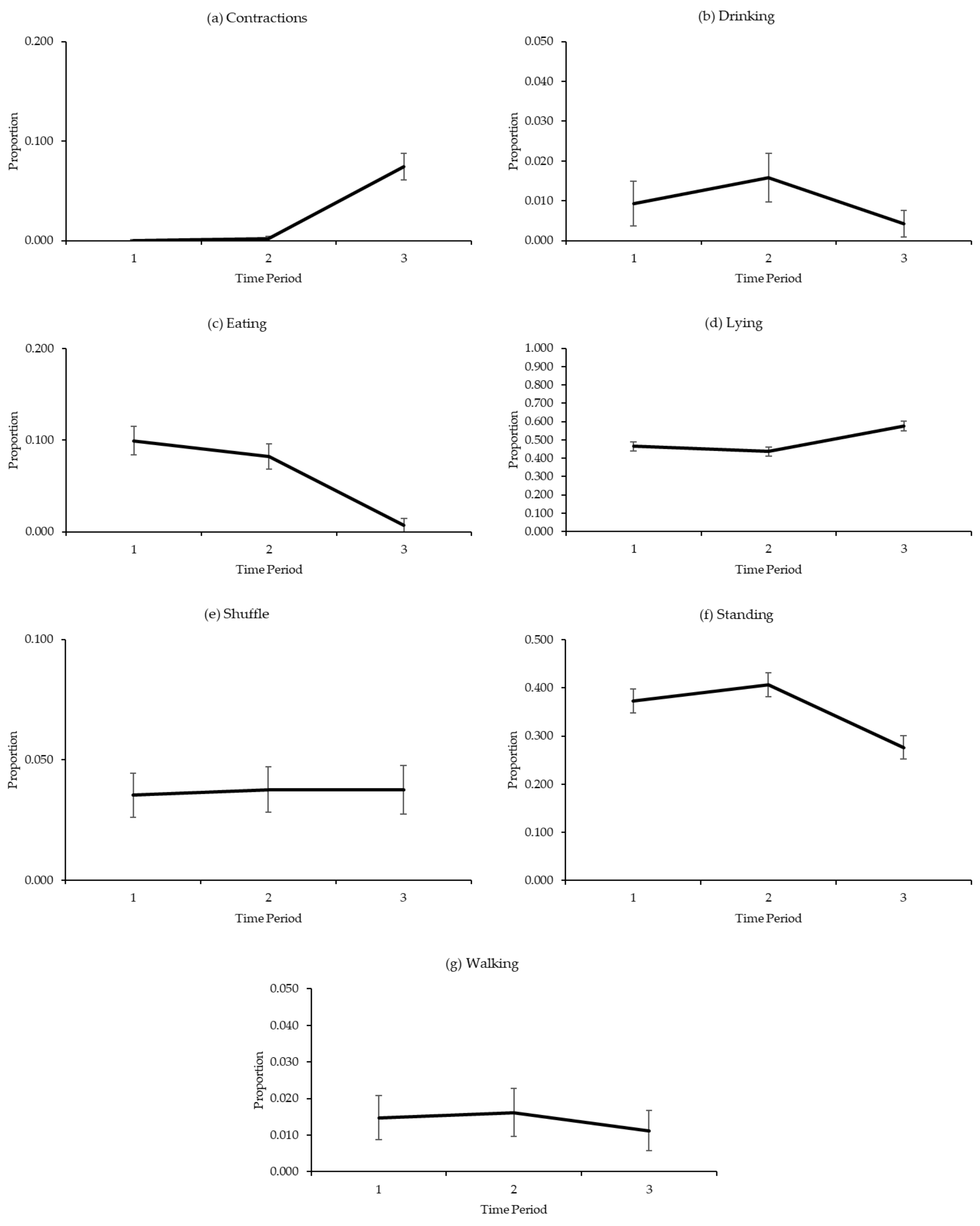

Figure 4.2. Mean (± s.e.) proportion of time that were (a) contractions, (b) drinking, (c) eating, (d) lying, (e) shuffle, (f) standing and (g) walking behaviour for dairy cow calvings (n = 35) in time periods one to three, with period three ending with the birth (Source: Cavendish et al. (2021)).

Differences were also found in the frequency of behaviours with standing (0.36) and shuffle (0.26) being most frequent, with other behaviours being 0.09 or less across the 9-hour period (Figure 4.3).

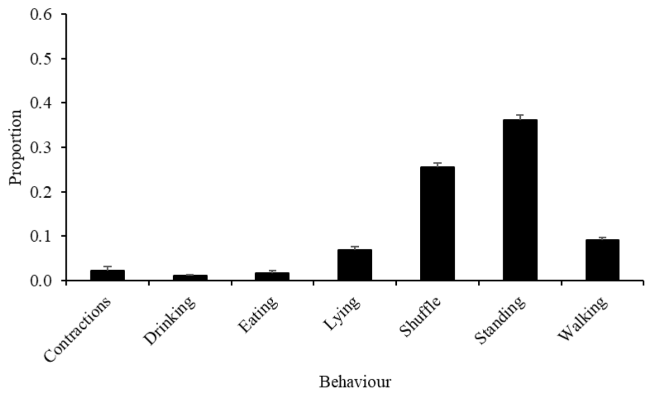

Figure 4.3. Predicted mean (± s.e.) proportion of observations for different dairy cow (n = 35) behaviours during the 9-hours prior to calving (Source: Cavendish et al. (2021)).

In the final three hours prior to calving, the frequency of lying and contraction bouts increased and the standing, shuffle, walking, drinking and eating bouts decreased (Figure 4.4).

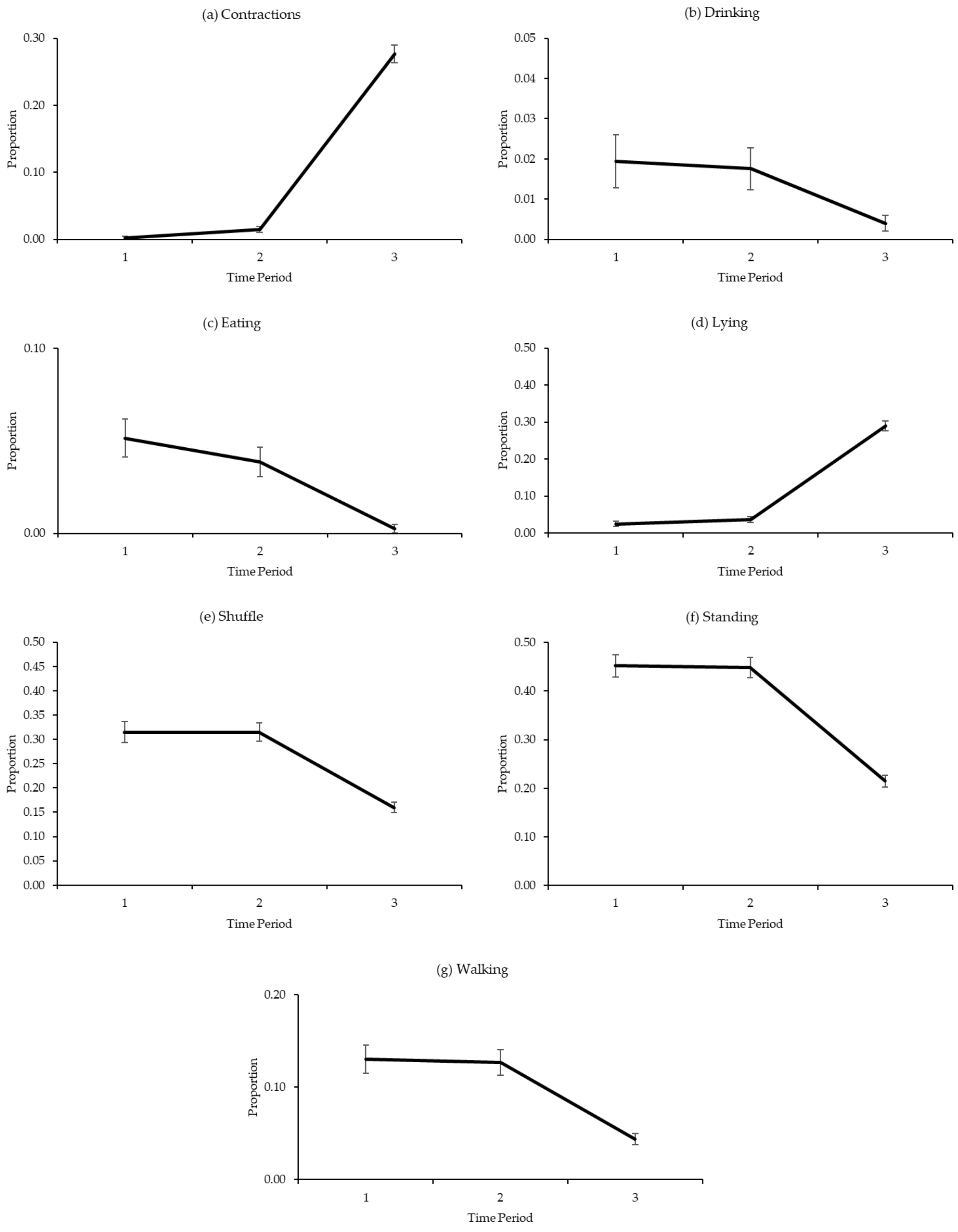

Figure 4.4. Mean (± s.e) proportion of observations that were (a) contractions, (b) drinking, (c) eating, (d) lying, (e) shuffle, (f) standing and (g) walking behaviour for dairy cow calvings (n = 35) in time periods one to three, with period three ending with the birth (Source: Cavendish et al. (2021)).

4.6 Discussion and conclusions

This study found that when monitoring calving the duration and frequency of lying and contraction bouts increased in the last three hours prior to birth compared to other time periods studied. Observing contractions and their increased frequency, along with increased frequency and time spent lying, can be used as indicators of progress in parturition. During the nine hours studied prior to calving, cows spent a large proportion of their time either lying or standing in their late pregnancy. Due to insufficient numbers of assisted births the difference in behavioural patterns between assisted and unassisted calvings was not studied, but have been found by others (Miedema et al., 2011; Schuenemann et al., 2011). Only 23% (8 of the 35 cows) in the current study needed assistance when calving, and therefore further observations of assisted births would be needed.

Cows tend to be lying when contractions are occurring. Lying is a highly motivated behaviour in dairy cows, with cows prioritising lying over other behaviours such as feeding, and especially after a period when these behaviours have been limited (Weary et al., 2008). Typically, cows when indoors will spend between 10–12 hours per day lying, and between eight and 10 hours per day grazing (Smid et al., 2020). In the current study cows spent about 12 h per day lying and eight hours standing, with more time spent lying and less time standing, drinking and eating as parturition progressed. The findings of the current study suggest the use of lying and standing transitions as a means for farmers and technology to detect the progress of parturition and imminent birth. Further behaviours such as tail movements and rumination time may have added to the current study since they are potentially visible on video footage.

Schuenemann et al. (2011) suggested that dystocic births are characterised by an increase in abdominal contractions for around 95 min until intervention is required. Therefore, if contractions can be tracked accurately, and potentially with technology, a prediction of dystocia could potentially be made given its importance in the monitoring of parturition. Electronic devices such as abdominal belts or intravaginal thermometers to detect uterine contractions and body temperature changes have been proposed as potential solutions (Rutten et al., 2017; Giaretta et al., 2021). During the current study, changes in behaviour were largely associated with standing, shuffle, walking and lying, with bouts of lying increasing in the period prior to calving, which is consistent with other studies (Huzzey et al., 2005). This potential restlessness is known to relate to discomfort in animals and may reflect late stages of pregnancy or boredom (Munksgaard et al., 2005).

In conclusion, cows spend a large proportion of their time either lying (0.49) or standing (0.35), with a higher frequency of standing (0.36) and shuffle (0.26) bouts than other behaviours. During the three-hours prior to calving, the duration and bouts of lying, including contractions, were higher than during other time periods. The monitoring of behavioural patterns (i.e. standing, lying and contraction bouts) could be used as an alert to the progress of parturition before the birth event occurs.

Chapter 5

5 Model for object detection and behaviour recognition.

This chapter is organised into two sections. Section 5.2 describes Mask R-CNN, and Section 5.3 focuses on the Non-Local Network.

5.1 Introduction

Section 5.2 delves into object detection through the lens of Mask R-CNN (He et al., 2017), providing deep insights into its intricate architecture and components. Various configurations and optimisations, such as the replacement of batch normalisation layers with group normalisation and the incorporation of a non-local block, are explored. The experiments utilise the MS COCO dataset (Lin et al. 2014), known for its extensive range and complexity in object detection, instance segmentation, and keypoint detection tasks.

Section 5.3 commences with a detailed description of the integration of the non-local block (Wang, et al., 2018) within the ResNet architecture (He et al., 2015), delineating the intricate mechanics and functionalities of this combined approach in the realm of behaviour recognition. Subsequent to this exposition, our implementation is rigorously evaluated against the Kinetics dataset (Kay et al., 2017), and the outcomes are meticulously compared with the results shown by Wang et al. (2018). Following this comparative analysis, the section proceeds to test the developed model on a subset of the cow dataset, as elaborated in Chapter 3.4, serving as a preliminary proof-of-concept examination. It is imperative to note that the results discussed herein pertain to this subset, and a more extensive analysis of the results obtained from the full dataset is reserved for Chapter 6.

5.2 Object detection

For object detection we use the state-of-the-art Mask R-CNN which was proposed by He et al. (2017). A high-level view of the architecture of Mask R-CNN is shown in Figure 5.1. In the next section we will describe each of the individual parts.

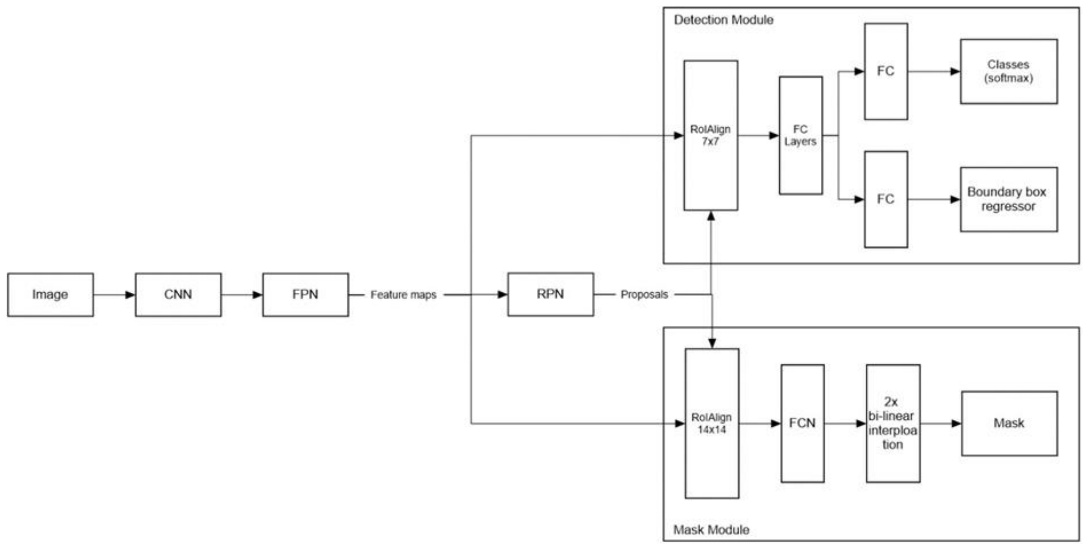

Figure 5.1. High-level view of Mask R-CNN architecture. The detection module does multiway object classification and bounding box refinement. The mask module uses a small fully convolutional network to segment the mask. RoIAlign is used to extract features

5.2.1 CNN

For the CNN we use the vanilla Resnet-50 architecture with some further improvements (Table 5.1). As shown by Wu and He (2018) batch normalisation (BN) performs poorly with small batch sizes. Therefore, similar to Detectron (FAIR, 2018), we replace all the batch normalisation (BN) layers of the Residual Network (ResNet) architecture (He et al., 2015) with group normalisation (GN) layers (Wu and He, 2018). Likewise, all convolution layers in the feature pyramid network (FPN) (Lin et a., 2017), Region of Interest (RoI) box head and RoI mask head are also followed by GN layers. Similar to Wang et al. (2018), we use a non-local block just before the last residual block of res4.

Table 5.1. ResNet-50 backbone architecture with a single non-local block, which is placed before the last residual block of res4. The convolutional kernels are HxW with the number of channels following. Residual blocks are shown in brackets with the number of times they are repeated displayed next to them.

	ResNet 50 backbone
conv1	7x7, 64 stride 2, padding 3
maxpool1	3x3, stride 2, padding 1
res2	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix} x3$
res3	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix} x4$
res4	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x5$ Non-local block $\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x1$
res5	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1,2048 \end{bmatrix} x3$

5.2.2 Feature Pyramid Network

The feature pyramid network (FPN) (Lin *et al.*, 2017), is used for feature extraction. It comprises of a bottom-up and top-down pathway as shown in Figure 5.2

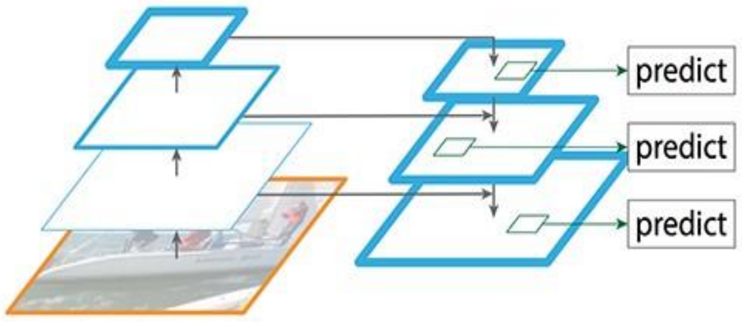

Figure 5.2. Feature pyramid network (FPN) improves feature extraction by adding a second pyramid that takes the high-level features from the first pyramid and passes them through the top-down pathway. Predictions are made independently at all levels in the pyramid (Source: Lin et al. (2017)).

FPN provides a top-down pathway to construct higher resolution layers from the semantic rich layer from the top of the bottom-up CNN. We use ResNet (He *et al*., 2016) as the CNN backbone and use {C2, C3, C4, C5} to denote the output of each layer's last residual block. The first layer is not used due to its large memory footprint. Lateral connections are added between reconstructed layers and the corresponding feature maps to help the detector more accurately predict locations.

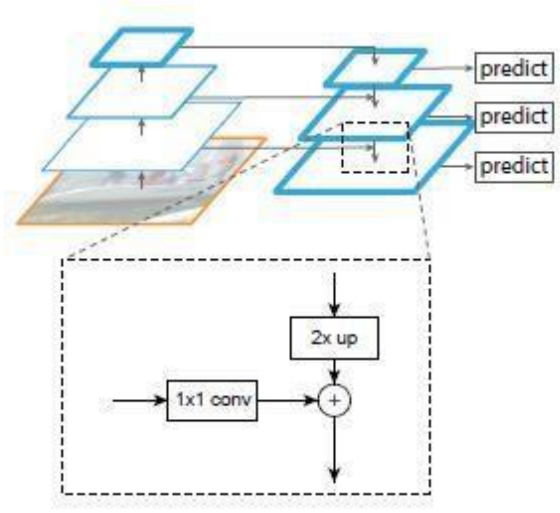

Figure 5.3. Building block showing the lateral connection and top-down pathway merged by elementwise addition (Source: Lin et al. (2017)).

Figure 5.3. illustrates the building block that is used in an iterative manner to construct the top-down feature maps. The coarser-resolution feature map is up sampled with nearest neighbour, by a factor of 2. To reduce the channel dimensionality to match that of the up sampled feature map, a 1x1 convolutional layer is applied to the corresponding bottom-up map. The two feature maps are then merged together with element-wise addition. To start the iterative process a 1x1 convolutional layer is applied to C5 to produce the coarsest resolution map. Finally, to reduce the aliasing effects of up sampling, a 3x3 convolutional layer is applied to each of the merged maps to produce the final feature maps denoted as {P2, P3, P4, P5}. A final layer, P6 is added to the FPN by applying maxpool with kernel size of 1 and stride of 2 to P5. Because the same classifier and box regressor is shared for all levels in the FPN, all feature maps have 256-d output channels. There are also no non-linearities attached to any convolutional layers. All weights are initialised using Xavier initialisation and biases are set to zero.

5.2.3 Region Proposal Network

To generate region proposals (Ren et al., 2015) a small subnetwork is evaluated on dense 3x3 sliding windows, preforming class-agnostic box classification and bounding box regression. The subnetwork consists of a 3x3 convolutional layer with ReLU, followed by two separate 1x1 convolutional layers for classification and regression.

Sigmoid is used to estimate the probability of the box being an object of interest. All weights are randomly initialised with a zero-mean Gaussian distribution with standard deviation of 0.01 and all biases are initialised to zero. This module is attached to each level of the FPN.

Each proposal has an associated reference box, called an anchor which is centred at the sliding window in question. the anchors have areas of {32^2, 64^2, 128^2, 256^2, 512^2} pixels on {P2, P3, P4, P5, P6} respectively and aspect ratios of {1:2, 1:1, 2:1}.

For each FPN level, we follow He et al. (2017) where the top 1000 classifications (2000 for training), box regression deltas and their associated anchors are kept. The anchors are transformed into RoI proposals p with:

$$p_x = x^d * w^a + x^a, \quad p_y = y^d * h^a + y^a \tag{5.1}$$

$$p_w = exp^{w^d} * w^a, \quad p_h = exp^{h^d} * h^a \tag{5.2}$$

where x, y, w and h denote the box's centre coordinates and its width and height. Variables X^a and X^d are for the anchor box and predicted box regression deltas respectively. The proposals are then clipped to the image window and a lose fitting non-maximum suppression (NMS) with threshold of 0.7 is applied based on their classification scores.

Next, the proposals from all FPN layers are combined and the top 1000 (2000 for training) are retained. If the network is in evaluation mode, then these RoI proposals are assigned to their appropriate FPN levels {P2, P3, P4, P5} according to their scale, with the following equation:

$$k = \left\lfloor k_0 + log_2\left(\frac{\sqrt{wh}}{224}\right)\right\rfloor \tag{5.3}$$

where k is the P_k layer of the FPN, 224 is the canonical ImageNet pre-training size and k_0 is the target level on which a RoI with $w * h$ = 224^2 should be mapped into, He *et al.* (2017) set this value to 4. In training mode, RoIs are first sampled from the 2000 proposals, before been assigned their appropriate FPN levels.

5.2.3.1 Box RoIs and targets

A total of 512 RoIs per image are sampled from the 2000 proposals. Of these, up to a maximum of 128 are foreground (IoU of at least 0.5 with the ground truth box) and the rest background. The box regression targets t are then calculated using:

$$t_x = 10 * (x^{gt} - x)/w, \qquad t_y = 10 * (y^{gt} - y)/h \tag{5.4}$$

$$t_w = 5 * log(w^{gt}/w), \qquad t_h = 5 * log(h^{gt}/h) \tag{5.5}$$

where x, y, w and h denote the box's centre coordinates and its width and height. Variables X and X^{gt} are for the RoI and ground-truth box respectively. The values 10 and 5 are for normalising the box regression targets (FAIR, 2018).

5.2.3.2 Mask RoIs and targets

Mask R-CNN also uses the above foreground RoIs for training and mask targets are calculated as the intersection between these foreground RoIs and their corresponding ground-truth masks.

5.2.4 Detection module

The assigned RoIs and FPN feature maps are then passed into the detection module for multiway object classification and bounding box refinement. RoIAlign (He *et al*. 2017) is used to pool features from the assigned FPN levels of the RoIs. The size of the extracted features is [N, 256, 7, 7], where N is the total number of RoIs. The extracted features are passed through two hidden 1024-d fully connected (fc) layers. The output of which is then passed through a classification and box regression layer, which run parallel to each other. Both hidden layers are followed by ReLU activations. The fc layers are initialised with Xavier initialisation. The classification layer is initialised with a zero-mean Gaussian distribution with standard deviation of 0.01 and the box regression layer with standard deviation of 0.001. All biases are initialised to zero. In inference mode the classification scores are passed through a softmax activation layer to obtain the predicted probabilities for each class.

5.2.5 Mask module

RoIAlign (He *et al.* 2017) is used to pool features from RoIs and their assigned FPN levels. The size of the extracted features is [N, 256, 14, 14], where N is the total number of RoIs. The features are passed through a fully convolutional network (FCN), which consists of four 3x3 convolutional layers, a 2x2 deconvolutional layer with stride of 2 and a 1x1 output convolutional layer with output channels equal to the number of classes. all hidden convolutions/deconvolutions have 256 output channels and are followed by ReLU. In inference mode the output of the FCN is passed through a sigmoid activation. All convolutional/deconvolutional layers are initialised with MSRA and zero bias.

5.2.6 Inference

The 1,000 proposals that are returned from the detection module are scaled back to image space, refined using equations, then clipped to the image window.

$$r_x = \frac{x^d}{10} * w + x, \qquad r_y = \frac{y^d}{10} * h + y \tag{5.6}$$

$$r_w = exp^{\frac{w^d}{5}} * w, \qquad r_h = exp^{\frac{h^d}{5}} * h \tag{5.7}$$

x, y, w and h denote the box's centre coordinates and its width and height. Variable X and X^d are for the bounding boxes and box regression deltas respectively. Next, all bounding boxes with classification scores less than 0.05 are rejected. This is the value chosen by He *et al.* (2017) to balance obtaining high recall against having too many low precision detections that will slow down inference post-processing steps. Then, per class NMS with threshold of 0.5 is used to remove duplicates based on their classification scores. To accommodate the maximum number of detections that is established for the COCO dataset, the top 100 detections (over all classes) are kept per image. Finally, the refined boxes are assigned to their appropriate FPN levels {P2, P3, P4, P5} according to their scale using equation (5.3), before being applied to the mask module. The predicted class that is returned from the detection module is used to select the mask, which is then resized to the RoI size and binarized at a threshold of 0.5.

5.2.7 Development and Tuning of a Cow Detection Dataset

As discussed in Chapter 3, videos were captured to observe the movements and behaviours of cows. From these videos, 193 still images were uniformly extracted from three cameras: 53 images from the first camera, 70 from the fourth camera, and the final 70 from the fifth camera.

Each camera was positioned to record varying numbers of cows in different pens. Camera 1 captured scenes where the number of cows fluctuated between 5 and 7 throughout the day. In contrast, Camera 4 constantly depicted 8 cows, and the number of cows in the frames from Camera 5 varied between 4 and 10.

To identify objects of interest in the images, we employed Inkscape, a vector graphics software, during the annotation process. All the annotated images were then saved in the PNG format, with selected examples presented in Figure 5.4.

Following the preparation of our dataset, we initiated the implementation of Mask R-CNN by integrating it with weights pre-trained on the MS COCO dataset. Post-initialisation, we embarked on fine-tuning the Mask R-CNN model specifically for detecting cows. This entailed adjusting the model parameters to attune the pre-trained model to the distinct task of recognising and segmenting cows within our set of images. This fine-tuning is pivotal as it enables the model to adapt effectively to our specific dataset, ensuring precise and dependable detection of cows in the sampled images.

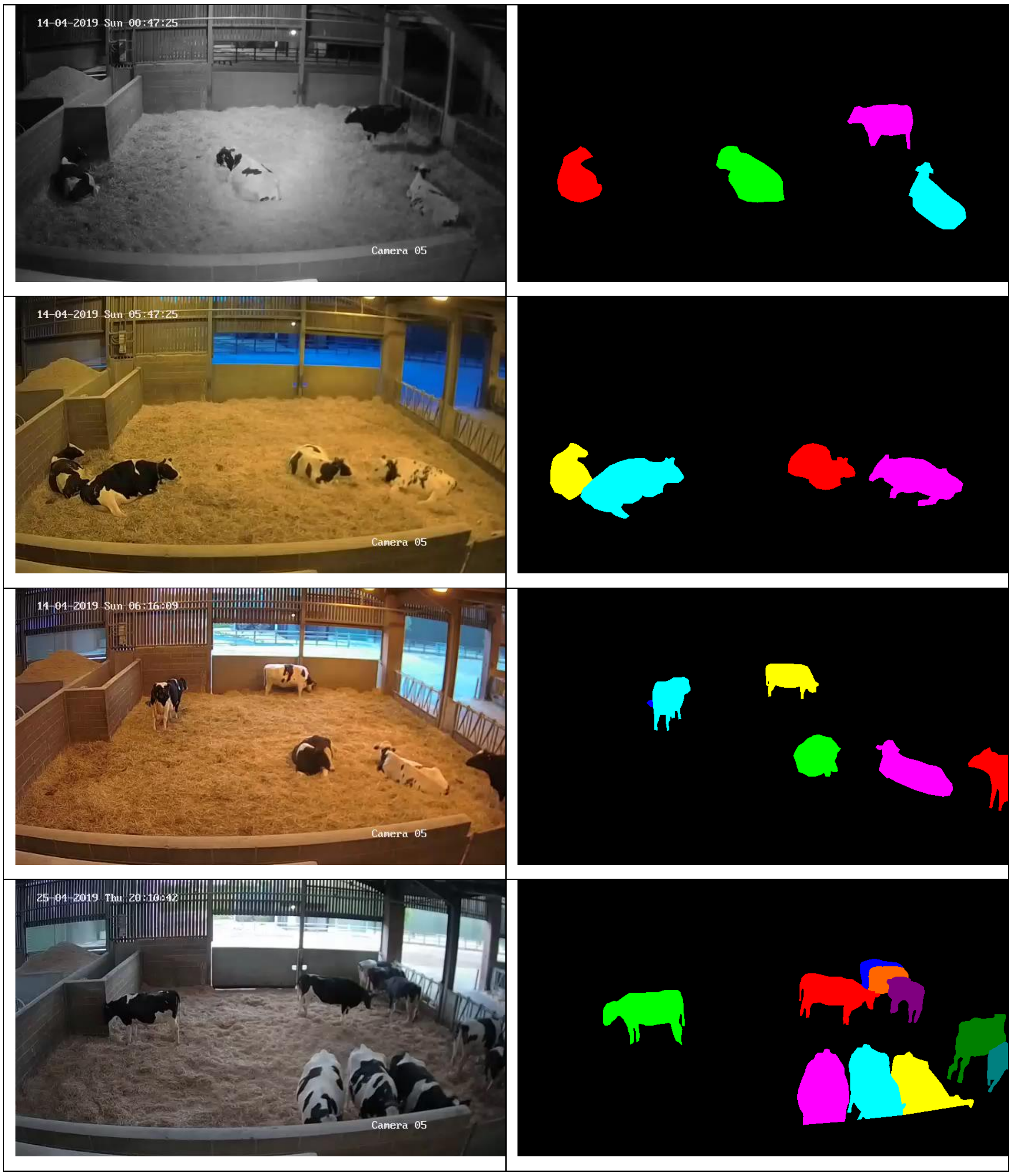

Figure 5.4 Selected Images from Camera 5. The left-hand side depicts variations in lighting conditions encountered in the dataset, and the right-hand side displays their corresponding annotations.

5.2.8 Training

The model is trained end-to-end using synchronised SGD training on 2 GPU's, with an effective mini-batch size of 4 (2 images per GPU) and trained for a total of 360k iterations. The learning rate is set to 0.005 for the first 240k iterations, 0.0005 for the next 80k and 0.00005 for the final 40k iterations. This is equivalent to FAIR (2018), were they use 8 GPU's, with an effective mini-batch size of 16, 90k iterations total and a starting learning rate 0f 0.2. As proposed by He *et al*. (2017), we use a weight decay of 0.0001 and momentum of 0.9. Images are resized so that their shorter edge is 800 pixels. To reduce overfitting on the cow dataset, we resize images such that their shorter edge is randomly sampled from the set [640, 672, 704, 736, 768, 800]. Each image has 512 sampled RoIs, with a ratio of 1:4 of positives to negatives.

5.2.8.1 Training RPN

Anchors are generated for each image, of these any anchors that have an intersection over union (IoU) greater than 0.7 with any ground-truth bounding box are considered foreground. The highest IoU to each ground-truth bounding box is also considered as foreground. Anchors with IoU less than 0.3 for all ground-truth bounding boxes are used as background. The maximum number of anchors used for training RPN is 256, of these a maximum of 128 are foreground, background makes up the rest. The box regression targets t are calculated using:

$$t_x = (x^{gt} - x^a)/w^a, \quad t_y = (y^{gt} - y^a)/h^a \tag{5.8}$$

$$t_w = log(w^{gt}/w^a), \quad t_h = log(h^{gt}/h^a) \tag{5.9}$$

where x, y, w and h denote the box's centre coordinates and its width and height. Variables X^a and X^{gt} are for the anchor and ground-truth box respectively.

5.2.9 Loss

The multi-task loss function of Mask R-CNN combines the loss of classification, bounding box regression and segmentation:

$$L = \sum_{p=2}^{6} L_{rpn\_cls}^{p} + \sum_{p=2}^{6} L_{rpn\_box}^{p} + L_{detection\_cls} + L_{detection\_box} + L_{mask} \quad (5.10)$$

L_{rpn_box} and $L_{bdetection\_box}$ are the losses for box regression and is calculated as the average smooth L1 loss between predicted boxes and box targets, where only positive samples contribute to the loss. L_{rpn_cls} the classification loss is calculated as a sigmoid binary cross-entropy loss between the predicted output and the ground-truth labels, which is averaged over all non-ignored targets. $L_{detection\_cls}$ is the cross-entropy loss between the predicted classifications and the ground-truth labels. L_{mask} is a per-pixel sigmoid and mean binary cross-entropy loss, which only includes the kth mask if the RoI is associated with the ground-truth class k.

5.2.10 Results

5.2.10.1 Microsoft Common objects in context dataset

The Microsoft Common objects in context (MS COCO) dataset (Lin *et al.* 2014) is one of the most challenging datasets that is used for instance segmentation, object detection and keypoint detection. It is designed for the purpose of detecting and segmenting objects that are found within their natural environment. The 2017 variant of the dataset consists of 118k training images and 5k images for validation. There are also 40k test images that are split into two equal groups (test-dev and test-challenge). Ground-truth labels for both test sets are not publicly available.

MS COCO dataset consists of 80 classes for object detection and instance segmentation which are split into 12 categories as follows:

- **Accessory**: backpack, handbag, suitcase, tie, umbrella.
- **Animal**: bear, bird, cat, cow, dog, elephant, giraffe, horse, sheep, zebra.
- **Appliance**: microwave, oven, refrigerator, sink, toaster.

- **Electronic**: cell phone, keyboard, laptop, mouse, remote, tv monitor.
- **Food**: apple, banana, broccoli, cake, carrot, donut, hotdog, orange, pizza, sandwich.
- **Furniture**: bed, chair, couch, dining table, potted plant, toilet.
- **Indoor**: book, clock, hair drier, scissors, teddy bear, toothbrush, vase.
- **Kitchen**: bottle, bowl, cup, fork, knife, spoon, wine glass.
- **Outdoor**: bench, fire hydrant, parking meter, stop sign, traffic light.
- **Person**: person.
- **Sports**: baseball, frisbee, kite, skateboard, skis, snowboard, sports ball, surfboard, tennis racket.
- **Vehicle**: airplane, bicycle, boat, bus, car, motorcycle, train, truck.

There are on average 3.5 classes and 7.7 instances per image. 10% of images contain only one class per image. The dataset comprises of approximately 41% small objects, 34% medium sized objects and 24% large scale objects.

5.2.10.2 MS COCO evaluation metrics

We adhere to the conventional evaluation metrics outlined by MS COCO for assessing models in object detection and instance segmentation tasks, focusing particularly on Mean Average Precision (mAP). mAP serves as a critical metric for assessing the effectiveness of object detection and instance segmentation models. For each category of object, the model's precision (equation 5.11) is gauged at various levels of recall (equation 5.12) and subsequently averaged to compute the Average Precision (AP) specific to that category (equation 5.13). Following this, the calculated APs for every category are averaged to determine the mAP (equation 5.14), offering a consolidated view of the model’s overall efficacy across diverse object categories. This singular metric encapsulates the model's capability to accurately identify and localise objects, facilitating a detailed and comparative evaluation of various models against the COCO dataset standards.

$$P = \frac{T_p}{T_P + F_p} \tag{5.11}$$

$$R = \frac{T_p}{T_p + F_n} \tag{5.12}$$

$$AP = \sum_n \quad (R_n - R_n - 1)P_n \tag{5.13}$$

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{5.14}$$

Table 5.2. MC COCO evaluation metrics.

Mean Average Precision (mAP)	
mAP$_{50:95}$	mAP at IoU = [0.50: 0.05: 0.95]
mAP$_{50}$	mAP at IoU = 0.50 (Pascal VOC metric)
mAP$_{75}$	mAP at IoU = 0.75 (strict metric)

As shown in Table 5.2, to reward detectors with better localisation, MS COCO uses the mAP$_{50:95}$ metric, which takes an average over 10 Intersection over Union (IoU) thresholds, that range from 0.5 to 0.95 with a step of 0.05, this is the main metric used for evaluation. MS COCO also uses two other metrics. mAP$_{50}$ is the standard pascal VOC challenge metric, where only a 50% overlap with the ground-truth bounding box is needed to be classed as detected and AP$_{75}$ which is a stricter version of Pascal VOC, where an overlap of 75% and above is considered as detected.

Table 5.3. Mask R-CNN with ResNet-50 and FPN. Results are for COCO object detection (box) and instance segmentation (mask). Top row shows results reported by Facebook's Detectron. Second row displays our PyTorch implementation of the baseline Mask R-CNN. Third row is

our Mask R-CNN with added group normalisation (GN) layers and bottom row is the same as the third row with an added non-local block. Best results are shown in bold lettering.

method	$AP^{box}_{50:95}$	AP^{box}_{50}	AP^{box}_{75}	$AP^{mask}_{50:95}$	AP^{mask}_{50}	AP^{mask}_{75}
Detectron	0.377	0.592	0.409	0.339	0.558	0.358
R50	0.379	0.595	0.414	0.345	0.562	0.365
R50_GN	**0.390**	**0.601**	**0.427**	0.349	**0.568**	0.372
R50_GN_NL	**0.390**	**0.601**	0.423	**0.350**	**0.568**	**0.373**

As can be observed in Table 5.3 adding GN layers significantly improves both bounding box detection and instance segmentation. While inserting a non-local block just before the last residual block of res4 only slightly improves instance segmentation and gives slightly worse results for object detection.

5.2.11 Conclusion

The exploration of Mask R-CNN on the MS COCO dataset has yielded insightful revelations regarding the impact of different architectural elements on model performance. The results, primarily reflected in the detailed comparison in Table 5.3, offer a nuanced understanding of how each modification and enhancement influences the model's effectiveness in object detection and instance segmentation tasks.

The baseline, established by Facebook’s Detectron, served as a reference point, demonstrating scores of 0.377 and 0.339 for object detection and instance segmentation, respectively. A slight improvement was observed with our PyTorch implementation of the baseline Mask R-CNN, indicating the potential for optimisation even within established frameworks.

The most significant revelation came with the incorporation of Group Normalisation (GN) layers, which led to a considerable enhancement in model performance, evidenced by an increase to 0.390 in object detection and to 0.349 in instance segmentation. This underscores the pivotal role of normalisation techniques in refining model stability and convergence, ultimately leading to more accurate and reliable object detection and instance segmentation.

Conversely, the introduction of a non-local block did not yield a comparable enhancement, suggesting that the ability of non-local blocks to capture long-range dependencies doesn't uniformly translate to substantial improvements in every context.

5.3 Behaviour recognition

Non-local blocks, also known as non-local neural networks or self-attention mechanisms, have garnered substantial interest within the realm of deep learning, particularly in fields such as natural language processing and computer vision. These blocks introduce a unique capability for facilitating non-local interactions among various spatial or temporal positions within the input data. They excel at capturing extensive dependencies within data, setting them apart from traditional convolutional or recurrent layers, which are bound by their local receptive fields. Instead, non-local blocks can forge connections between distant elements found in input sequences or images. Furthermore, their versatility shines as they find application in both spatial and temporal data processing. In computer vision, they play pivotal roles in tasks like image classification, object detection, and semantic segmentation. Similarly, in video analysis, they prove invaluable for modelling long-range temporal dependencies, a critical aspect of recognising intricate actions and events.

5.3.1 Non-local bock

Non-local operation computes the response at a position as a weighted sum of the features at all positions in the input feature maps and is defined as follows:

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f\left(x_i, x_j\right) g\left(x_j\right) \tag{5.15}$$

where x is the input features, y is the output features (same size as x), i is the current position of interest, j enumerates over all possible positions, $C(x)$ is the normalisation factor $C(x) = \sum_{\forall j} f(x_i, x_j)$, g is a linear embedding $g(x_j) = W_g x_j$ where W_g is learned weight matrix and $f(x_i, x_j)$ is a pairwise function that computes the scalar between the feature at location i and those at all possible positions j. Wang, et al. (2018), experiment with four types of pairwise functions, namely Gaussian eq. (5.16), embedded Gaussian eq. (5.17), dot product eq. (5.18) and concatenation eq. (5.19), they found the latter three all produced similar performance.

The Pairwise functions are described as follows:

Gaussian	$f(x_i, x_j) = e^{x_i^T x_j}$	(5.16)
Embedded Gaussian	$f(x_i, x_j) = e^{\theta(x_i)^T \varphi(x_j)}$	(5.17)
Dot Product	$f(x_i, x_j) = \theta(x_i)^T \varphi(x_j)$	(5.18)
Concatenation	$f(x_i, x_j) = ReLU(w_f^T [\theta(x_i), \varphi(x_j)])$	(5.19)

In eq. (5.19) $[.,.]$ denotes concatenation and w_f is a weight vector which projects the concatenated vector to a scalar. To simplify the gradient computation in eq. (5.18) and eq. (5.19), Wang, et al. (2018) set the normalisation factor to $C(x) = N$, where N is the number of positions in x.

For our work we chose the embedded gaussian function. The reasons for this are threefold: firstly, it is the default function used in the paper; secondly, in Facebook Research Github repository, it is the only pairwise function that they display results; thirdly, it is the only trained model that they give access to download and training a new model from scratch would take far too long.

The embedded Gaussian function is described in eq. (5.17), where $\theta(x_i) = W_\theta x_i$ and $\varphi(x_j) = W_\varphi x_j$ are two linear embeddings.

For any given i, $\frac{1}{C(x)} f(x_i, x_j) = \frac{e^{\theta(x_i)^T \varphi(x_j)}}{\sum_{\forall j} e^{\theta(x_i)^T \varphi(x_j)}}$ becomes a softmax computation long the j dimension.

Wang, et al. (2018), noted that the self-attention module presented by Vaswani et al. (2017) is a special case of non-local operations in the embedded Gaussian version. Therefore, following Vaswani et al. (2017), the self-attention form can then be described as:

$$y = softmax\left(\frac{(W_\theta x)^T W_\varphi x}{\sqrt{d_\theta}}\right) W_g x \tag{5.20}$$

To counter the softmax function producing extremely small gradients in the attention maps, the dot-products of $(W_\theta x)^T W_\varphi x$ are scaled by $1/\sqrt{d_\theta}$ where d_θ is the number channels used in the bottleneck.

Finally, the nonlocal operation is then wrapped into a non-local block and is defined as follows:

$$z_i = W_z y_i + x_i \tag{5.21}$$

where y_i is the genetic function given in eq. (5.15), W_z computes a position-wise embedding on y_i, restoring the number of channels to that of x and x_i denotes the residual connection. Adding the residual connection allows the non-local block to be added into a pre-trained network as an identity block if the weights W_z are initialised to zero.

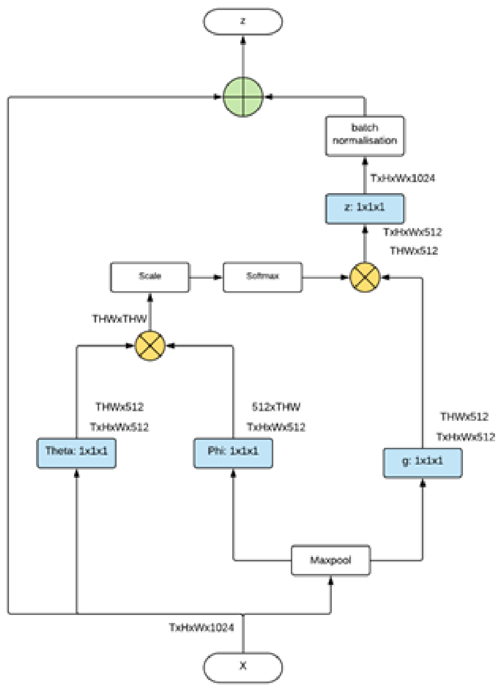

Figure 5.5. A spacetime non-local block (Embedded Gaussian) used in videos. $\otimes$ denotes matrix multiplication and $\oplus$ denotes element-wise sum. Blue boxes denote 1x1x1 convolutions. The feature maps are shown as the shape of their tensors (temporal, height, w width, channels) and reshaping is performed when noted. A maxpool layer with a kernel and stride of 1x2x2 is used to half the spatial dimensions of W_φ and W_g. The dot-products of the matrix multiplication $W_\theta W_\varphi$ are scaled by $1/\sqrt{d_\theta}$ where d_θ is the number channels used in the bottleneck. Softmax is performed on each row.

A spacetime non-local block refers to a specific type of non-local block that is designed to capture interactions and dependencies not only in the spatial dimensions of data (e.g., within an image) but also across the temporal dimension (e.g., across frames in a video sequence). As illustrated in Figure 5.5, W_θ, W_φ, W_g and W_z are learned weight matrices that are implemented as 1x1x1 convolutions. Similar to the bottleneck design of ResNet (He et al., 2015), W_θ, W_φ and W_g have half the number of channels of X. This reduces the computation of the non-local block by around a half. W_z computes position-wise embedding on y_i and restores the channel dimension to that of X.

As shown in Figure 5.5, to speed up the pairwise computation, we use the same subsampling trick that is used by Wang, et al. (2018), were a maxpool layer with a kernel and stride of 1x2x2 is used to half the spatial dimensions of both W_{φ} and W_g. Adding the maxpool layer reduces the pairwise computation by up to a quarter.

5.3.2 ResNet

The ResNet architecture uses 3-layer bottleneck building blocks with shortcut connections, known as residual blocks (Figure 9.11). An identity shortcut is used if the input and output dimensions are equal, otherwise a projection shortcut is used to match dimensions. Element-wise addition is used to join the shortcut connection to the output of the block. The residual blocks are shown in brackets in Table 5.4 and how many times they are repeated is displayed next to them. Unlike in the original paper (Wang et al., 2018) that uses 32-frame input clips, we opt for 8-frame input clips. the reason for this is to significantly improve training/testing speed.

Table 5.4. ResNet-50 non-local network. The dimensions of the output maps are CxTxHxW, where C is the number of channels and T,H,W are the temporal, height and width dimensions. The dimensions of the pooling layers are TxHxW. The convolutional kernels are HxW with the number of channels following. residual blocks are shown in brackets. The input is $3x8x224x224$.

ResNet-50		
	layer	Output size
conv1	7x7, 64 stride 1,2,2	64x8x112x112
maxpool1	1x3x3, stride 1,2,2	64x8x55x55
res2	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix} x3$	256x8x55x55
maxpool2	2x1x1, stride 2,1,1	256x4x55x55
res3	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix} x2$ Non-local block $\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix} x2$ Non-local block	512x4x28x28
res4	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x2$	1024x4x14x14

	Non-local block $\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x2$ Non-local block $\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x2$ Non-local block	
res5	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix} x3$	2048x4x7x7
average pool	4x7x7	2048x1x1x1
fc	number of categories	

5.3.3 Non-local network

Table 5.4 shows our implementation of the non-local neural network using the ResNet-50 architecture. The RGB input video clip has 8 frames all of 224x224 pixels. The kernels of all convolutions are essentially 2D kernels that process the input frame-by-frame but are implemented as 3D kernels with a dimension of 1xkxk, where k is the size of the 2D kernel. This allows us to use available ResNet weights that have been pre-trained on ImageNet. The maxpool2 layer is used to halve the temporal dimension. Similar to Wang et al. (2018), we use five non-local blocks in total, 2 in res3 and 3 in res4. In both res3 and res4 each non-local block is separated by two residual blocks.

5.3.4 Training

The non-local network uses weights that are pre-trained on ImageNet (Deng et al., 2009). As illustrated in Figure 5.6. the 8-frame input clips are generated by randomly cropping out 64 consecutive frames from the training video and then keeping 8 frames that are separated by a stride of 8 frames. The spatial size is fixed to 224 pixels squared, which is randomly cropped from a video or its horizontal flip, whose shorter side is randomly scaled between 256 to 320 pixels.

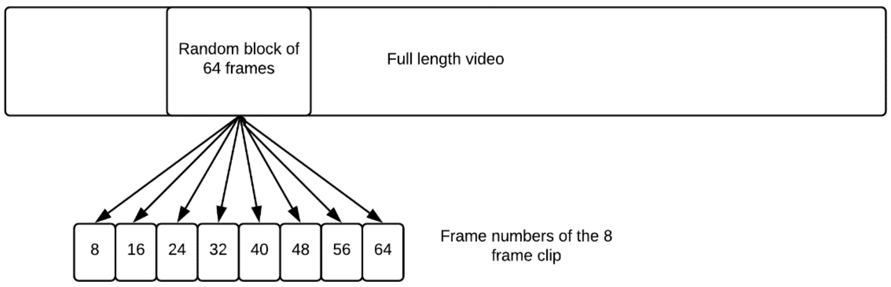

Figure 5.6. Temporal sampling of each video clip with eight evenly spaced frames being selected from a block of 64 consecutive frames.

We train our model on 2-GPUs with an effective mini-batch size of 16 (8 clips per GPU). Our model is trained for a total of 1,200k iterations. The learning rate starts at 0.0025 and is reduced by a factor of 10 at the 600k and 1,000k interval marks. This is equivalent to Wang et al. (2018), where they use 8-GPU's, with an effective mini-batch size of 64, 400k iterations total and a starting learning rate of 0.01.

Similar to Wang et al. (2018), we adopt a weight decay of 0.0001, momentum of 0.9 and add a dropout layer with a drop out ratio of 0.5 after the average pooling layer. To reduce overfitting, the model is fine-tuned using Batch Normalisation. This is in contrast to the typical methods of fine-tuning ResNet architectures for object detection, where the batch normalisation layers are frozen.

For the Non-Local block, the convolutions W_θ, W_φ and W_g are initialised with a zero-mean Gaussian distribution with a standard deviation of 0.01. W_z is initialised with a zero-mean Gaussian distribution with a standard deviation of zero. Following Goyal et al. (2018), we set the scale parameter of the batch normalisation layer to zero. The reason for this is because it causes the forward/backward signal initially to propagate through the identity shortcut, which improves the models output by easing optimisation at the beginning of training.

5.3.5 Loss function

The loss function for the non-local network is a cross-entropy loss between the predicted output and the ground-truth labels and is calculated as follows:

$$L = -\sum_{i} y_i' \, log(y_i) \tag{5.22}$$

where y' is the ground truth probability vector and y corresponds to the predicted probability vector.

5.3.6 Inference

Following Wang et al. (2018), we perform spatially fully convolutional inference on videos where the shorter side is resized to 256 pixels. 3 crops of 256x256 are used to cover the entire spatial size. 10 evenly spaced clips are sampled along the temporal dimension of the video (Figure 5.7) and softmax scores are computed on each of these clips. The final predicted output is just the average of all the softmax scores.

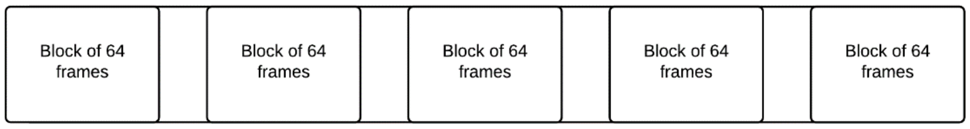

Figure 5.7. Ten clips of eight frames are sampled from blocks (64 frames) which are evenly sampled over the entire video. Each clip produces its own score, and the final output is the average of all the scores (a total of 5 blocks are shown for illustration purposes).

5.3.7 Results

To evaluate the performance of our system, all tests are done using 2 NVIDIA GeForce GTX 1080Ti GPUs, on a PC running the Linux operating system with 32 gigabytes of ram.

As the original code was written using Caffe2 (which is now deprecated), we first check to make sure our PyTorch implementation achieves the same results as those reported by Facebook Research, on the Kinetics dataset, which is described below. Next, to make sure the training part of our code works correctly, we train a model from scratch using ImageNet pre-trained weights which we downloaded from Facebook Research. The results for the above experiments are shown in Table 5.5. Finally, known that our code produces similar results to that of the original Caffe2 implementation, we investigate using ImageNet and Kinetics pre-trained weights for initialising the network with different dropout ratios, results of which are shown in Table 5.7.

5.3.7.1 Kinetics Human Action dataset

The Kinetics Human Action Video dataset, which was proposed by Kay et al. (2017) is one of the most challenging datasets for human action recognition. It is a large-scale dataset and contains approximately 306,245 unique videos which are all sourced from YouTube. The videos are mostly of homemade quality, which enables a great degree in variation in resolution, illumination, camera viewpoint and frame rate.

Kinetics consists of 400 human action classes where each class has around 400 to 1150 unique video clips. Each clip lasts for around 10 seconds and is labelled with a single class. The dataset is split into 246,245 video clips for training (250-1000 videos per class), 20,000 clips for validation (50 videos per class) and 40,000 clips for testing (100 videos per class).

The actions cover a broad range of classes from laughing to person-person interactions such as hugging and kissing, as well as person-object interactions such as playing bagpipes and mowing the lawn.

5.3.7.2 Classification accuracy top-1 and top-5

The Kinetics dataset measures its classification performance using both a top-1 and top-5 measure. The reason for using a top-5 metric as well as a top-1 is because each video clip is only labelled for a single class, but it is possible that a clip can contain multiple actions.

Top-1: Top-1 classification accuracy shows the percentage of how many times the highest probability predicted by the network is the same as the target label.

Top-5: Top-5 classification accuracy is measured by how many times the target label is within the 5 highest classes predicted by the network.

In both cases, the top score is computed as the times a predicted label matches the target label, divided by the total number of classes.

5.3.7.3 Changes from paper

Due to expired YouTube URLs (Uniform Resource Locator), it is not possible to train on the full dataset that was used in the paper. We instead train our model using 232,303 out of the 246,245 training videos and test on 19,684 validation video clips.

We downloaded the dataset from Facebook Research's official Github repository. In order to make the overall size of the dataset smaller, they used FFmpeg (FFmpeg.org, 2019) to rescale all videos to a height of 256 pixels. The total size of the rescaled dataset is 134GB. There are 5% fewer training videos compared to the original dataset. The only change we made was to remove all videos that have fewer than 64 frames, as this is the size of a video clip used for training and testing.

5.3.7.4 Kinetics Results

Table 5.5 shows the top-1 and top-5 results, using 5 non-local blocks on a ResNet-50 backbone. The GPUs column shows how many were used to train the model. Input frames are taken over 64 consecutive frames, using a stride of 2 for 32 input frames and a stride of 8 for 8 input frames. Number of iterations for the trained model are worked out as follows:

$$iterations = \frac{300k * 8}{2} = 1200k = 1{,}200{,}000$$

where 8 is the original number of GPUs used for training, 2 is how many GPUs are used to train the new model and k = 1000.

Table 5.5. Results using 5 non-local blocks within a ResNet-50 backbone. Results are shown for the non-local networks paper (top row), official Facebook Github repository (second row) and converted PyTorch code using original model that is converted to PyTorch (third row) and our model which was trained on ImageNet (bottom row).

Kinetics dataset results					
ResNet-50 (5 non-local blocks)					
platform	Iterations	Input frames	GPUs	Top-1	Top-5
Paper					
Paper	400k	32	8	73.8	91.0
Original Model					
Caffe2	400k	8	8	74.4	91.4
PyTorch	400k	8	8	74.1	91.4
Trained Model					
PyTorch	1,200k	8	2	71.1	89.4

As shown in Table 5.5, the paper section shows the results reported in the Non-local Neural Networks paper (Wang et al., 2018) for 5 non-local blocks. The original paper uses 32 input frames with a stride of 2. In the original model section, the top row shows Facebook Caffe 2 official results (taken from GitHub) and the second row shows converted PyTorch results. Same model is used for both versions. To significantly improve both training and testing times, the input frames per clip are reduced to just 8 with a stride of 8. As can been seen, this also has the added benefit of improving both top-1 and top-5 results. Missing videos in the test set most likely contribute to the difference in top-1 results, as the original model was trained and tested on the complete dataset. The trained model section is trained on PyTorch, with 2 GPUs, using ImageNet pre-trained weights. We used only 1,200k iterations (equivalent to 300k iterations if using 8 GPUs) instead of 400k. The reason for this is because of the amount of time it takes to train a model. Using 8 GPUs, it takes approximately 6 days to train 400k iterations.

The reason why there is a 3% difference in top-1 results between the original and trained models is most likely due to a smaller training set (approximately 10k less videos), fewer iterations and number of GPUs used to train the model. This can be seen in Table 5.6, where Facebook Research experimented with using both fewer iterations and GPUs. The results shown in Table 5.6 do not use any non-local blocks and was implemented using 3D convolutions.

Table 5.6. Results taken from Facebook research for training with fewer iterations and GPUs. top row shows the baseline model, trained with 400k iterations and 8 GPUs. Second row uses the same setup as the baseline but is trained with 100k fewer iterations. Bottom row is equivalent to the middle row but uses 4 GPUs instead of 8 for training.

	iterations	input frames	GPUs	Top1	Top5
Baseline	400k	8	8	73.4	90.9
Fewer iterations	300k	8	8	73.2	90.8
Less GPUs	600k	8	4	73.0	90.4

As can be seen in Table 5.6, using a shorter training schedule (8-GPU 400k vs 8-GPU 300k), there is only a slight drop in accuracy compared to the baseline model of 0.2 for top-1 and 0.1 for top-5 scores. Whereas lowering the number of iterations and GPUs used for training (8-GPU 400k vs 4-GPU 600k), shown in bottom row, further reduces the accuracy compared to the baseline model by 0.4 for top-1 and 0.5 for top-5 scores.

5.3.7.5 Cow Results and Discussion

As outlined in Chapter 5.1, the subset of the dataset explored in this section is derived from the comprehensive dataset depicted in Chapter 3.4. This subset encompasses behaviours from 35 cows and includes two additional behaviour classes, culminating in a total of nine classes: contractions while standing and birth. Given the limited classes in the cow dataset, we employ top-2 classification accuracy to capture clips that display multiple actions, such as lying and contractions while lying.

The training leverages two GPUs, running for a total of 5,000 iterations (approximately 39 epochs), which is where the validation/training error stabilises. Results are acquired using 8 input frames per clip with a temporal stride of 8 frames.

We undertake investigations using different initialisation parameters under two scenarios: pre-training on ImageNet and pre-training on Kinetics, with both fine-tuned on our cow dataset. For each setting, we experiment with setting the dropout regularization ratio to 0.5 and 0.9.

Table 5.7 displays the results acquired using pre-trained ImageNet and Kinetics weights for initialisation, fine-tuned on our proof-of-concept cow dataset, with the top section trained with a dropout ratio of 0.5 and the bottom section using 0.9. Best results are shown in bold lettering.

Pre-trained weights	Top1	Top2
Dropout ratio 0.5		
ImageNet	70.36	86.03
Kinetics	72.57	89.44
Dropout ratio 0.9		
ImageNet	70.70	88.59
Kinetics	**73.76**	**91.31**

From the results presented in Table 5.7, it is evident that pre-training on the Kinetics dataset outperforms ImageNet. This superior performance is attributed to the Kinetics being trained for motion, whereas ImageNet is only trained on spatial information (still images). Additionally, a higher dropout ratio notably improves generalization. A detailed breakdown of the top-1 behaviours, pre-trained on Kinetics with a dropout ratio of 0.9, is presented in Table 5.8.

Further investigations into removing the temporal pooling layer (maxpool2 in Table 5.4) were conducted to maintain the temporal dimension consistency throughout the network. However, the results were suboptimal, achieving only a top-1 score of 56.22% and top-2 of 80.58%.

Table 5.8 provides a detailed breakdown of top-1 behaviours for the proof-of-concept dataset, pre-trained on the Kinetics dataset, with a dropout of 0.9. It depicts the target row, output row, and the percentage row, showcasing the number of video clips tested for each behaviour, the count of correctly classified behaviours by the model, and the actual percentage of correctly classified instances for each individual behaviour, respectively.

	Stand	Lie	Walk	Shuffle	Contractions stand	Contractions lie	Birth	Eating	Drinking
Target	90	90	87	90	9	90	18	75	38
Output	70	71	63	63	0	74	0	65	27
Percentage	77.78	78.89	72.41	70.00	00.00	82.22	00.00	86.67	71.05

The variances in behaviour detection accuracy, demonstrated in Table 5.8, reveal that contractions while lying have an accuracy of 82.22%, sufficient to predict the birth of a calf, as a cow usually starts contractions approximately 1 to 2 hours prior to giving birth. Other behaviours also yielded reasonable accuracy, with some exceptions due to inadequate data or the rarity of the behaviour. However, inaccuracies in categories like standing contractions and births, due to occlusions and misclassifications, necessitated the removal of these categories from the final dataset.

5.3.8 Conclusion

Initial testing showed crucial insights into the performance of our system and its comparative efficacy against pre-established benchmarks. We observed remarkable reliability and consistency in our PyTorch implementation compared to the original Caffe2 platform.

While our system effectively mirrored the results of the original model in many aspects, the unavoidable constraints associated with the availability of the complete dataset and required computational resources resulted in a subtle compromise in the model's performance. These discrepancies are elucidated by variations in the top-1 and top-5 scores which, although nominal, illustrate the inevitable impact of reduced datasets and altered training schedules.

The exploration and analysis of the proof-of-concept cow dataset highlighted the superior efficacy of Kinetics pre-training for behavioural classification due to its emphasis on motion-based learning, in contrast to the spatially oriented ImageNet. Enhanced generalisation was achieved with higher dropout ratios, contributing to more accurate predictions of impending calving events based on exhibited behaviours.

However, despite successes in most behaviour classifications, challenges encountered in annotating rare behaviours like standing contractions and births due to inadequate data, occlusions, and the inherent difficulty in annotating such behaviours underscored the need for more robust data collection and annotation methodologies. The decision to exclude the problematic categories from the final dataset is a compromise to maintain the reliability and accuracy of the model, emphasising the significance of comprehensive and accurate data collection and annotation in future works.

Chapter 6

6 Automated monitoring of dairy cows to determine the progress of parturition.

6.1 Summary

The aim of this study was to investigate the application of existing image recognition techniques for predicting the behaviour of dairy cows. As discussed in Chapter 3, a total of 46 dairy cows were continuously monitored under 24-hour video surveillance prior to calving. Videos were annotated according to the procedures and tools delineated in Chapters 3.3 to 3.6, noting behaviours such as standing, lying, walking, shuffling, eating, drinking, and contractions while lying, for each cow from 10 hours prior to calving. As indicated in Chapter 3.7, a total of 3,969 video clips were recorded, with 3,186 used for training and 783 used for validation. A non-local neural network was then trained and validated on these video clips, each representing specific behaviours. The results of this study revealed that the non-local network accurately classified the seven behaviours in over 80% of cases within the validated dataset. Specifically, the network accurately detected birth contractions 83% of the time, serving as a potential early warning system for calving since all cows exhibit contractions several hours prior to giving birth. This approach to behaviour recognition utilising video cameras can assist livestock management.

6.2 Introduction

As public concern about animal welfare and livestock management intensifies, the agricultural sector is pressed to adopt innovative solutions that enhance animal well-being and bolster public trust in agricultural methodologies. The contemporary landscape is witnessing livestock handlers responsible for managing more extensive quantities of animals due to escalating operational costs, including labour, and the challenges of acquiring skilled farm workers, coinciding with the trend of increasing average sizes of dairy herds.

In addressing these evolving challenges, the agricultural domain has seen the introduction of sophisticated digital camera systems, equipped with 24/7 video surveillance features. These innovations empower farmers to keep tabs on their livestock from afar while tending to other vital farm duties. Utilising cameras for the observational study of animal behaviour is a practice rooted in decades of application, primarily focused on animals in controlled settings like dairy cows (Lawrence and Stott, 2009).

The importance of uninterrupted animal monitoring is paramount to maintaining animal welfare and survival (Wathes et al., 2008). The advent of automated image analysis methods has ushered in the capability for continuous monitoring that only requires users to interpret system outputs, a level of sustained observation that surpasses human capabilities.

The amalgamation of computer vision with artificial intelligence, particularly deep learning, has broadened the scope in the field of animal observation. These advanced technologies are capable of a multitude of functions including recognising animals, detecting objects, localising body parts, and segmenting animal forms within images. Adaptations specifically designed for video analysis can discern specific animal actions such as standing, lying down, walking, eating, and drinking (Cangar et al., 2008).

The merits of image analysis are evident; it circumvents the necessity for human intervention and obviates the need for intrusive equipment like collars or transponders. It also tends to offer more in-depth insights compared to other monitoring systems while being cost-effective. However, the efficacy of these advancements is contingent upon access to a substantial volume of high-quality imagery, a prerequisite acknowledged by specialists in the domain (Tian et al., 2020).

Vision-based observation is versatile, applicable to individual animals as well as groups, whether they are herds, flocks, or parent-offspring units. Persistent individual animal monitoring technology has the potential to yield unbiased assessments of unusual behavioural patterns, enabling timely responses and enhancing the alertness of stockpersons.

The objective of this study was to investigate the application of existing image recognition techniques to predict the behaviour of dairy cows. This study amassed a substantial collection of high-quality video images representing a variety of cow behaviours, with the workflow detailed in Chapter 3. A dataset of this nature was notably absent but was essential for training a computer vision model in the current study.

6.3 Materials and methods

Approval for this study was obtained from the University of Nottingham animal ethics committee before commencement (approval number 151).

6.3.1 Computer vision model used for behaviour recognition.

In this chapter, we employ the finalised cow behaviour dataset—detailed in Chapter 3—and the non-local network (Wang et al., 2018) discussed in Chapter 5.3, incorporating the ResNet-50 architecture (He et al., 2015), to investigate dairy cow behaviour recognition. As outlined in Chapter 5.3.7.5, we initialise the non-local network with a dropout ratio of 0.9 and weights pre-trained on the Kinetics image dataset (Kay et al., 2017). The use of pre-trained weights is established to enhance action recognition (Carreira et al., 2017). All other settings for training and inference are consistent with those discussed in Chapter 5.3.

6.4 Results and discussion

Despite scientific value, pressing need and direct impact on animal health and welfare, very little attention has been paid in developing an annotated video dataset of dairy cow behaviours. Most research to date has been based on wearable accelerometer-based activity monitoring sensors (Diosdado et al., 2015; Rahman et al., 2018; Benaissa et al., 2019). We present a novel, extensive video dataset designed for cow behaviour classification, described in detail in Chapter 3. To advance vision-based technologies such as animal behaviour recognition, there is a pressing need for image banks enriched with high-quality (accurate and high-resolution) images, as emphasised by previous studies (Tian et al., 2020). This study demonstrates the feasibility of automated monitoring of cows during parturition, a capability crucial for aiding stockpersons and promoting animal welfare, especially for high-value animals.

Our finalised dataset, detailed in Chapter 3, encompasses nearly 4,000 video clips, each ranging from 3 to 10 seconds, depicting individual behaviours of pregnant dairy cows before calving. The total footage amounted to over 9 hours and 42 minutes, divided into approximately 7 hours and 48 minutes for training and 1 hour and 54 minutes for validation.

In the realm of computer vision, action recognition models have achieved significant success when applied to human subjects (Wang et al., 2018). As demonstrated in Chapter 5.3.7.5, a model pre-trained on the Kinetics dataset (Kay et al., 2017)—a collection specifically designed for recognising human actions—can be effectively repurposed to identify behaviours in dairy cows.

As outlined in Table 6.1, our investigations indicate an 83% accuracy in identifying contractions while lying down, which is pivotal for predicting calving since contractions typically begin 1 to 2 hours before the actual birth. The model achieved over 84% accuracy in recognising behaviours such as standing, lying, eating, and drinking, crucial for effective animal well-being monitoring. Alterations in the duration or frequency of these identified behaviours can act as pivotal indicators of impending parturition, as elaborated in Chapter 4.6.

Notably, behaviours like eating and drinking were detected with over 90% accuracy, serving as significant indicators of potential health issues (Weary et al., 2009).

Table 6.1. Evaluation of model predictions against validation dataset. 1 The target row shows how many video clips were tested for each behaviour. 2 Output row shows how many behaviour video clips the model classified correctly. 3 The percentage of target behaviour video clips correctly classified.

	Stand	**Lie**	**Walk**	**Shuffle**	**Contractions**	**Eating**	**Drinking**
Target[1]	134	135	134	134	112	91	43
Output[2]	113	122	107	108	93	86	40
Accuracy[3]	84	90	80	80	83	95	93

As well as working with cows, the proposed computer vision approach could be adapted for other livestock species such as pigs, poultry, sheep, and horses to predict birth and identify behaviour patterns or behaviours that occur over many hours, which may be missed by subjective and observational sampling. Furthermore, because the calving pen is continuously monitored, it should also be possible to detect and track the behaviours of the mother and its newborn offspring, which is not feasible using standard predictive animal monitoring applications that are currently being used by the livestock industry.

6.5 Conclusion

We show that computer vision can be successfully applied to predict individual dairy cow behaviours with an accuracy of 80% or more for the behaviours studied. This approach could be used for early detection of abnormal behaviour in animals, birth events and the need for assistance. Computer vision technology may help a stockperson make more timely decisions based on the continuous tracking of individuals within groups of animals.

Chapter 7

7 General discussion and conclusions

7.1 Application of computer vision for birth detection

While still developing, the automatic prediction of individual animal behaviour and welfare of animals has great promise for non-invasive animal monitoring and farm management use. Furthermore, a camera vision-based monitoring system may not only support farmers with routine management tasks but also support farm assurance schemes as a repeatable, reliable and objective measure across different farm environments for animal welfare reporting. As a management tool, the monitoring of cows at calving is essential to ensure a successful birth, and this activity can be supported by computer vision software.

Cavendish et al. (2021), found that cows display a noticeable uptick in the frequency and duration of lying and contractions three hours before calving. These behaviours are reliable indicators of imminent parturition. Most calvings in this study were unassisted, with only 23% requiring intervention. This emphasises the importance of correctly identifying labour signs. The behaviours shown during this period also reflect the animals' comfort and well-being, with increased restlessness indicating discomfort from the later stages of pregnancy. Since cows prioritise lying, recognising changes in this pattern is pivotal for monitoring parturition.

While Cavendish et al.'s (2021) study was thorough, observing more behaviours like tail movements and rumination time, visible on video, might offer a more detailed understanding of pre-calving cow behaviours. These added parameters could significantly improve predictions of impending parturition. Also, Given the limited number of cows needing assistance in this study, more research is imperative to understand the behavioural differences between assisted and unassisted calvings, which can provide invaluable insights into cows potentially facing calving complications.

For the non-local network, we demonstrated that pre-training using the Kinetics dataset yielded superior results compared to pre-training on ImageNet. This is attributed to the fact that the Kinetics dataset, being an action recognition video dataset, places emphasis on motion-based learning, whereas ImageNet, being a static image dataset, is spatially oriented. We also showed that the non-local network used in this study could correctly classify seven key behaviours 80% or more of the time. While changes in standing and lying behaviours can serve as indicators of parturition, a computer vision model can detect birth contractions, thereby reducing false positives in calving alerts, and increasing the reliability of predictions. The current study found that contractions could be detected and correctly predicted 83% of the time, providing a potential early warning calving alert, as all cows start contractions several hours prior to giving birth.

We further improved upon vanilla Mask R-CNN by adding Group normalisation (GN) which was proposed by Wu and He (2018). As with Detectron (FAIR, 2018), we replaced all batch normalisation (BN) layers of the Residual Network (ResNet) architecture (He et al., 2015) with GN layers. Similarly, all convolution layers in the feature pyramid network (FPN) (Lin et al., 2017), Region of Interest (RoI) box head and RoI mask head are followed by GN layers. The incorporation of GN layers led to a considerable enhancement in model performance, evidenced by an increase from 0.377 to 0.390 in object detection and from 0.339 to 0.349 in instance segmentation. This underscores the pivotal role of normalisation techniques in refining model stability and convergence, ultimately leading to more accurate and reliable object detection and instance segmentation.

Similar to Wang et al. (2018), we also investigated with using a non-local block just before the last residual block of res4 (Table 5.1). But, instead of using the standard Mask R-CNN, we opted to use the improved Mask R-CNN with GN layers. We report results for the above experimentations using Mask R-CNN with a ResNet-50 backbone and FPN. The introduction of a non-local block yielded only a marginal enhancement, improving segmentation from 0.349 to 0.350, with object detection remaining constant. This suggests that the ability of non-local blocks to capture long-range dependencies does not uniformly translate to significant improvements in all contexts.

We annotate a behaviour dataset of 46 individual calvings, that can be used by animal behaviourists. All recordings start 10 hours before parturition and continue for a further 5 hours afterwards, as this is the critical period for the health and welfare of the cow and of its calf. This results in a total of approximately 690 hours of annotated behaviours.

To our knowledge we have built and annotated the only video dataset that is devoted to cow behaviour during parturition. This video dataset is generated from the previously mentioned behaviour dataset. It encompasses seven different behaviours (stand, lie, walk, shuffle, drink, eat and lying contractions) which are recorded over 46 calvings. There is a total of 3,969 videos clips which have been annotated so far, where each clip lasts between 3 to 10 seconds, this results in 9 hours and 42 minutes of annotated video data. The recorded video clips in our dataset are generated to offer complete real-world conditions for cow behaviours within a working daily farm and cover all possible situations such as pose, illumination and occlusions.

We benchmark our video dataset for behaviour recognition using the state-of-the-art human action recognition algorithm, namely, the non-local network, which was proposed by Wang et al. (2018). Results indicate that the current dataset is challenging for current state-of-the-art methods that are developed for human action recognition.

7.2 Conclusion

This study contributes to the ongoing exploration of computer vision in livestock monitoring, centring on the detailed examination of calving. By leveraging a highly specialised and annotated behaviour dataset, along with insights from the Cavendish et al. (2021) study, it has been possible to unearth nuanced understandings of bovine behaviour during parturition. The application of a non-local network, particularly with the implementation of the Kinetics dataset, has proven to be integral, enhancing the capability to discern and predict subtle behaviours with heightened accuracy. These insights and methodologies are not just enhancements but are transformative, opening avenues for profound advancements in animal welfare, sustainable livestock management, and informed agricultural practices, steering towards a future where innovation aligns seamlessly with ethical and humane coexistence.

7.3 Future Directions

This exploration into automated livestock monitoring via deep learning has unveiled extensive prospects for further research and development. Moving forward, our agenda for future work is extensive and diversified, aiming to address several crucial factors.

Dataset Diversification and Enhancement: A pivotal area is the extensive diversification and enrichment of the dataset, incorporating a spectrum of dairy cow breeds and environmental contexts to foster model robustness and versatility. A richer, more nuanced dataset can also include more anomalies in behaviour patterns, vital for enhancing early detection capabilities of health and stress-related anomalies in cows.

Cross-Species Adaptation and Application: An exploration into the model's adaptability and efficacy across various livestock species is crucial, with an aim to formulate a universal animal behaviour monitoring system, thus broadening the spectrum of its applicability across the agricultural domain.

Real-Time Alerting and Monitoring Development: The evolution of the model into a real-time alerting and monitoring system is essential, providing instantaneous notifications and insights into abnormal behaviours or calving events, thereby enabling timely interventions.

Refinement in Recognition Techniques: Pursuing advancements and innovations in image recognition and computer vision techniques is vital to the continual enhancement of model accuracy and reliability in predicting behavioural patterns.

Correlative Studies on Behaviour and Health: Investigating the correlations between observable behaviours and livestock health to formulate predictive models for early disease detection is essential for ensuring the welfare of the animals through timely interventions.

User Interface and Training Enhancement: There’s a need to develop intuitive user interfaces coupled with comprehensive training modules, ensuring the effective deployment and interpretation of the system’s output by end-users.

Sustainability and Ethical Consideration Integration: The embedding of sustainability and ethical practices within the technological development and deployment processes is crucial to ensure the holistic well-being of the animals and alignment with sustainable livestock management.

Development of Automated Intervention Mechanisms: Exploring automated intervention mechanisms triggered by abnormal behaviours detected by the system will facilitate immediate responses in critical and emergency situations.

Enhancement of Model Robustness: Enhancing model robustness to accommodate varying lighting, orientations, and occlusions is crucial to ensure consistent and reliable performance across diverse operational conditions.

Integration with Existing Management Systems: Creating integration solutions for seamless incorporation of developed models with existing farm management software is critical for providing a unified and streamlined solution to livestock managers.

Incorporation of User Feedback Loops: Establishing mechanisms for continuous user feedback integration will be pivotal for the ongoing refinement and enhancement of models and systems, aligning development with practical needs and field challenges.

By addressing these multifaceted dimensions of future work, there is promising potential to bring unprecedented advancements in the field of automated livestock monitoring, contributing substantially to animal welfare, sustainable agriculture, and optimised livestock management practices.

8 References

Achour, B.; Belkadi, M.; Filali, I.; Laghrouche, M.; Lahdir, M. 2020. Image analysis for individual identification and feeding behaviour monitoring of dairy cows based on convolutional neural networks (cnn). Biosyst. Eng., 198, 31–49.

Barrier, A.C., Haskell, M.J., Birch, S., Bagnall, A., Bell, D.J., Dickinson, J., Macrae, A.I., Dwyer, C.M. 2013. The impact of dystocia on dairy calf health, welfare, performance and survival. Vet. J., 195, 86–90, doi:10.1016/j.tvjl.2012.07.031.

Bell, M. J., Garnsworthy, P. C., Stott, A. W., and Pryce, J. E. 2015. The effect of changing cow production and fitness traits on profit and greenhouse gas emissions from UK Dairy systems. J. Agric. Sci. 153, 138–151. doi: 10.1017/S0021859614000847.

Bell, M. J., and Roberts, D. J. 2007. The impact of uterine infection on a dairy cow's performance. Theriogenology, 68, 1074–1079. doi: 10.1016/j.theriogenology.2007.08.010.

Bell, M.J. Tzimiropoulos, G. 2018. Novel monitoring systems to obtain dairy cattle phenotypes associated with sustainable production. Front. Sustain. Food Syst., 2, 31, doi:10.3389/fsufs.2018.00031.

Benaissa, S., Tuyttens, F., Plets, D., Pessemier, T., Trogh, J., Tanghe, E., Martens, L., Vandaele, L., Van Nuffel, A. and Joseph, W. 2019. On the use of on-cow accelerometers for the classification of behaviours in dairy barns. Res. Vet. Sci., 125, 425–433.

Bengio, Y., Simard, P. and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5, 157–166.

Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B. 2016. Simple Online and Realtime Tracking. arXiv:1602.00763.

Biresaw, T., Nawaz, T., Ferryman, J. and Dell, A. 2016. ViTBAT: Video Tracking and Behavior Annotation Tool. In IEEE.

Bodla, N., Singh, B., Chellappa, R. and Davis, L.S. 2017. Improving Object Detection With One Line of Code. arXiv:1704.04503v2.

Borchers, M.R.; Chang, Y.M.; Proudfoot, K.L.; Wadsworth, B.A.; Stone, A.E.; Bewley, J.M. 2017. Machine-learning-based calving prediction from activity, lying, and ruminating behaviors in dairy cattle. J. Dairy Sci., 100, 5664–5674.

Buades, A., Coll, B. and Morel, J-M. 2005. A non-local algorithm for image denoising. In Computer Vision and Pattern Recognition (CVPR).

Cangar, Ö., Leroy, T., Guarino, M., Vranken, E., Fallon, R., Lenehan, J., Mee, J., Berckmans, D. 2008. Automatic real-time monitoring of locomotion and posture behaviour of pregnant cows to calving using online image analysis. Comput. Electron. Agric., 64, 53–60, doi:10.1016/j.compag.2008.05.014.

Carreira, J. and Zisserman, A. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. arXiv:1705.07750v3 [cs.CV].

Crammer, K. and Singer, Y. 2002. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. JMLR, 2, 265–292.

Dai, J., Li, Y., He, K. and Sun, J. 2016. R-FCN: Object Detection via Region-based Fully Convolutional Networks. arXiv:1605.06409v2.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H. and Wei, Y. 2017. Deformable Convolutional Networks. arXiv:1703.06211v3.

Defra, 2018. The Future Farming and Environment Evidence Compendium. Available online at:

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/683972/future-farming-environment-evidence.pdf. [Accessed 20 Feb. 2023].

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In CVRP.

Diosdado, J., Barker, Z., Hodges, H., Amory, J., Croft, D., Bell, N. and Codling, E. 2015. Classification of behaviour in housed dairy cows using an accelerometer-based activity monitoring system. Anim. Biotelemetry, 3, 15, doi:10.1186/s40317-015-0045-8.

Donahue, J., Hendricks, L., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K. and Darrell, T. 2016. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. arXiv:1411.4389v4 [cs.CV].

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J. and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. IJCV, 88(2), 303-338.

Fadul, M.; Christopher, B.; Alsaaod, M.; Hasler, J.; Alexander, S.; Adrian, S.; Hirsbrunner, G. 2017. Prediction of calving time in dairy cattle. Anim. Reprod. Sci., 187, 37–46.

FAIR, 2018. Detectron. https://github.com/facebookresearch/Detectron.

Feichtenhofer, C., Fan, H., Malik, J. and He, K. 2019. SlowFast Networks for Video Recognition. arXiv:1812.03982v2 [cs.CV].

Feichtenhofer, C., Pinz, A. and Zisserman, A. 2016. Convolutional Two-Stream Network Fusion for Video Action Recognition. arXiv:1604.06573v2 [cs.CV].

Fenlon, C., O'Grady, L., Mee, J., Butler, S., Doherty, M. and Dunnion, J. 2017. A comparison of 4 predictive models of calving assistance and difficulty in dairy heifers and cows. Journal of Dairy Science. 100, 12. 9746-9758.

FFmpeg.org, 2019. FFmpeg. [online] Available at: https://ffmpeg.org [Accessed 10 Sep. 2019].

Fuentes, A.; Yoon, S.; Park, J.; Park, D.S. 2020. Deep learning-based hierarchical cattle behavior recognition with spatio-temporal information. Comput. Electron. Agric., 177, 105627.

Giaretta, E., Marliani, G., Postiglione, G., Magazzù, G., Pantò, F., Mari, G., Formigoni, A., Accorsi, P.A. and Mordenti, A. 2021. Calving time identified by the automatic detection of tail movements and rumination time, and observation of cow behavioural changes. Animal, 15, 100071, doi:10.1016/j.animal.2020.100071.

Girdhar, R., Ramanan, D., Gupta, A., Sivic, J. and Russell, B. 2017. ActionVLAD: Learning spatio-temporal aggregation for action classification. arXiv:1704.02895 [cs.CV].

Girshick, R. 2015. Fast R-CNN. arXiv:1504.08083v2.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524v5.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June; arXiv:1311.2524.

Glorot, X. and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. [PDF]. http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

Goyal, P., Doll´ar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K. 2018. Accurate, Large Minibatch SGD: Training ImageNet in 1 hour. arXiv:1706.02677v2 [cs.CV].

Graves, A. and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5):602–610.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. 2017. Mask R-CNN. arXiv:1703.06870v3.

He, K., Zhang, X., Ren, S. and Sun, J. 2015. Deep Residual Learning for Image Recognition. https://arxiv.org/pdf/1512.03385.pdf.

He, K., Zhang, X., Ren, S. and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852v1.

Heuritech. 2016. Deep CNN and Weak Supervision Learning for visual recognition. https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learningmeetup-5/

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. Neural Computation, 9, 1735–1780.

Huang, G.B.; Zhu, Q.Y.; Siew, C.K. 2004. Extreme learning machine: A new learning scheme of feedforward neural networks. Neural Netw., 2, 985–990.

Huang, G.; Zhu, Q.; Siew, C. 2006 Extreme learning machine: Theory and applications. Neurocomputing, 70, 489–501.

Huang, G., Liu, Z. and Maaten, L. 2018. Densely Connected Convolutional Networks. https://arxiv.org/pdf/1608.06993.pdf.

Huang, Z., Huang, L., Gong, Y., Huang, C. and Wang, X. 2019. Mask Scoring R-CNN. arXiv:1903.00241v1 [cs.CV]. CVPR, 6409-6418.

Huzzey, J.M., von Keyserlingk, M.A.G. and Weary, D.M. 2005. Changes in Feeding, Drinking, and Standing Behavior of Dairy Cows During the Transition Period. J. Dairy Sci., 88, 2454–2461, doi:10.3168/jds.S0022-0302(05)72923-4.

Hyslop, J., Ross, D., Bell, M. and Dwyer, C. 2008. Observations on the time course of calving events in unassisted multiparous spring calving suckler cows housed in a straw bedded yard.

In Proceedings of the British Society of Animal Science; Cambridge University Press: Cambridge, UK, p.158.

Ioffe, S. and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167v3.

Jensen, M.B. 2012. Behaviour around the time of calving in dairy cows. Appl. Anim. Behav. Sci., 139, 195–202, doi:10.1016/j.applanim.2012.04.002.

Ji, S., Xu, w., Yang, M. and Yu, K. 2010. 3D convolutional neural networks for human action recognition. In International Conference on Machine Learning (ICML).

Jiang, B., Yin, X., Song, H., 2020. Single-stream long-term optical flow convolution network for action recognition of lameness dairy cow. Comput. Electron. Agric. 175, 105536. https://doi.org/10.1016/j.compag.2020.105536.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L. 2014. Large-scale Video Classification with Convolutional Neural Networks. In CVPR.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vi-jayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suletman, M. and Zisserman, A. 2017. The kinetics human action video dataset. arXiv:1705.06950 [cs.CV].

Krizhevsky, A., Sutskever, I., Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst., 25, 1097–1105.

Lawrence, A., Stott, A. 2009. Profiting from animal welfare: An animal-based perspective. J. R. Agric. Soc. Engl., 170, 40–47.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. [PDF]. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf.

Li, D., Chen, Y., Zhang, K. and Li, Z. 2019. Mounting Behaviour Recognition for Pigs Based on Deep Learning. Sensors, 19(22), 4924; https://doi.org/10.3390/s19224924.

Li, X., Cai, C., Zhang, R., Ju, L., He, J., 2019. Deep cascaded convolutional models for cattle pose estimation. Comput. Electron. Agric. 164, 104885. https://doi.org/10.1016/j.compag.2019.104885.

Lin, M., Chen, Q. and Yan, S. 2013. Network In Network. arXiv:1312.4400v3.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. 2017. Feature Pyramid Networks for Object Detection. arXiv:1612.03144v2.

Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P. 2017. Focal Loss for Dense Object Detection. arXiv:1708.02002v2.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. 2014. Microsoft COCO: Common objects in context. arXiv:1405.0312v3.

Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J. 2018. Path Aggregation Network for Instance Segmentation. arXiv:1803.01534v1.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C. and Reed, S. 2016. SSD: Single shot multibox detector. arXiv:1512.02325v5.

Lombard, J.E.; Garry, F.B.; Tomlinson, S.M.; Garber, L.P. Impacts of Dystocia on Health and Survival of Dairy Calves. J. Dairy Sci. **2007**, 90, 1751–1760, doi:10.3168/jds.2006-295.

Long, J., Shelhamer, E. and Darrell, T. 2015. Fully Convolutional Networks for Semantic Segmentation. arXiv:1605.06211v1.

Luo, W., Li, Y., Urtasun, R. and Zemel, R. 2017. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. arXiv:1701.04128v2.

Miedema, H.M., Cockram, M.S., Dwyer, C.M., MacRae, A.I. 2011. Changes in the behaviour of dairy cows during the 24h before normal calving compared with behaviour during late pregnancy. Appl. Animal Behav. Sci., 131, 8–14.

Munksgaard, L., Jensen, M.B., Pedersen, L.J., Hansen, S.W. and Matthews, L. 2005. Quantifying behavioural priorities—Effects of time constraints on behaviour of dairy cows, Bos taurus. Appl. Anim. Behav. Sci., 92, 3–14, doi:10.1016/j.applanim.2004.11.005.

Nair, V. and Hinton, G.E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6419&rep=rep1&type =pdf.

Neethirajan, S. 2017. Recent advances in wearable sensors for animal health management. Sens. Bio-Sensing Res. 12, 15–29. doi: 10.1016/j.sbsr.2016.11.004.

Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R. and Toderici, G. 2015. Beyond Short Snippets: Deep Networks for Video Classification. arXiv:1503.08909v2 [cs.CV].

Qiao, Y., Truman, M. and Sukkarieh, S., 2019. Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming.

Qiu, Z., Yao, T. and Mei, T. 2017. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. arXiv:1711.1030 [cs.CV].

Rahman, A., Smith, D., Little, B., Ingham, A., Greenwood, P., Bishop-Hurley, G. 2018. Cattle behaviour classification from collar, halter, and ear tag sensors. Inf. Process. Agric., 5, 124–133, doi:10.1016/j.inpa.2017.10.001.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. 2016. You only look once: Unified, real-time object detection. arXiv:1506.02640v5.

Redmon, J. and Farhadi, A. 2016. Yolo9000: Better, faster, stronger. arXiv:1612.08242v1.

Redmon, J. and Farhadi, A. 2018. YOLOv3: An Incremental Improvement. arXiv:1804.02767v1.

Ren, S., He, K., Girshick, R. and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497v3.

Rutten, C.J., Kamphuis, C., Hogeveen, H., Huijps, K., Nielen, M. and Steeneveld, W. 2017. Sensor data on cow activity, rumination, and ear temperature improve prediction of the start of calving in dairy cows. Comput. Electron. Agric., 132, 108–118, doi:10.1016/j.compag.2016.11.009.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L. 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://arxiv.org/pdf/1801.04381.pdf.

Schuenemann, G.M., Nieto, I., Bas, S., Galvão, K.N. and Workman, J. 2011. Assessment of calving progress and reference times for obstetric intervention during dystocia in Holstein dairy cows. J. Dairy Sci., 94, 5494–5501, doi:10.3168/jds.2011-4436.

Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J. and Napolitano, A. 2010. Rusboost: A hybrid approach to alleviating class imbalance, IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans 40 (1) (2010) 185–197. doi:10.1109/Tsmca.2009.2029559.

Simonyam, K. and Zisserman, A. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. arXiv:1406.2199v2 [cs.CV].

Simonyan, K and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556v6.

Simonyan, K. and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/pdf/1409.1556.pdf.

Smid, A.-M.C., Weary, D.M., von Keyserlingk, M.A.G. 2020. The influence of different types of outdoor access on dairy cattle behavior. Front. Vet. Sci., 7, 257, doi:10.3389/fvets.2020.00257.

Soomro, K., Zamir, A. and Shah, M. 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. arXiv:1212.0402 [cs.CV].

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15. Pp.1929-1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. 2014. Going Deeper with Convolutions. arXiv:1409.4842v1.

Tan, M., Pang, P., Le, Q., 2020. EfficientDet: Scalable and Efficient Object Detection. In: IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/arXiv:1911.09070.

Tian, H., Wang, T., Liu, Y., Qiao, X., Li, Y. 2020. Computer vision technology in agricultural automation—A review. Inf. Process. Agric., 7, 1–19, doi:10.1016/j.inpa.2019.09.006.

Titler, M., Maquivar, M.G., Bas, B., Rajala-Schultz, P.J., Gordon, E., McCullough, K., Federico, P. and Schuenemann, G.M. 2015. Prediction of parturition in Holstein dairy cattle using electronic data loggers. Journal of Dairy Science, 98(8), pp.5304-5312.

Tran, D., Bourdev, L., Fergus, R., Torresani, L. and Paluri, M. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. arXiv:1412.0767v4 [cs.CV].

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y. and Paluri, M. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. arXiv:1711.11248v3 [cs.CV].

Tu, S., Liu, H., Li, J., Huang, J., Li, B., Pang, J., and Xue, Y. 2020. Instance Segmentation Based on Mask Scoring R-CNN for Group-housed Pigs. ICCEA, pp. 458-462.

Uijlings, J., Van de Sande, K., Gevers, T. and Smeulders, A. 2013. Selective Search for Object Recognition. IJCV.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. and Polosukhin, I. 2017. Attention is all you need. arXiv:1706.03762v5 [cs.CL].

Wang, H., and Schmid, C. 2013. Action Recognition with Improved Trajectories. IEEE International Conference on Computer Vision, 3551-3558.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X. and Gool, L. 2016. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. arXiv:1608.00859v1 [cs.CV].

Wang, X., Girshick, R., Gupta, A., He, K. 2018. Non-Local Neural Networks. arXiv:1711.07971v3[cs.CV].

Wathes, C.M., Kristensen, H.H., Aerts, J.-M., Berckmans, D. 2008. Is precision livestock farming an engineer's daydream or nightmare, an animal's friend or foe, and a farmer's panacea or pitfall? Comput. Electron. Agric., 64, 2–10, doi:10.1016/j.compag.2008.05.005.

Weary, D.M., Huzzey, J.M., von Keyserlingk, M.A.G. 2009. Board-invited review: Using behavior to predict and identify ill health in animals. J. Anim. Sci., 87, 770–777, doi:10.2527/jas.2008-1297.

Wu, D., Wu, Q., Yin, X., Jiang, B., Wang, H., He, D., Song, H., 2020. Lameness detection of dairy cows based on the YOLOv3 deep learning algorithm and a relative step size characteristic vector. Biosyst. Eng. 189, 150–163.

Wu, D., Wang, Y, Han, M., Song, L., Shang, Y., Zhang, X. and Song, H. 2021. Using a CNN-LSTM for basic behaviors detection of a single dairy cow in a complex environment. Comput. Electron. Agric. 182, 106016. https://doi.org/10.1016/j.compag.2021.106016.

Wu, Y. and He, K. 2018. Group Normalization. arXiv:1803.08484v3[cs.CV].

Wu, Z., Wang, X., Jiang, Y., Ye, H. and Xue, X. 2015. Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification. arXiv:1504.01561v1 [cs.CV].

Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, H. 2017. Aggregated Residual Transformations for Deep Neural Networks. arXiv:1611.05431v2.

Yang, Z., Li, Y., Yang, J. and Luo, J. 2018. Action Recognition with Spatio-Temporal Visual Attention on Skeleton Image Sequences. arXiv:1801.10304v2 [cs.CV].

Yin, X.; Wu, D.; Shang, Y.; Jiang, B.; Song, H. 2020. Using an EfficientNet-LSTM for the recognition of single Cow's motion behaviours in a complicated environment. Comput. Electron. Agric., 177, 105707.

Yu, F. and Koltun, V. 2015. Multi-Scale Context Aggregation by Dilated Convolutions. arXiv:1511.07122.

Zaborski, D., Proskura, W., Grzesiak, W., Szatkowska, I. and Jedrzejczak-Silicka, M. 2017. Use of random forest for dystocia detection in dairy cattle. https://literatur.thuenen.de/digbib_extern/dn059657.pdf.

Zaborski, D., Proskura, W., Grzesiak, W. and Rozanska-Zawieja, J. 2019. The comparison between random forest and boosted trees for dystocia detection in dairy cows. Comput. Electron. Agric. 163, 104856.

Zehner, N., Niederhauser, J.J., Nydegger, F., Grothmann, A., Keller, M., Hoch, M. and Schick, M. 2012. Validation of a new health monitoring system (RumiWatch) for combined automatic measurement of rumination, feed intake, water intake and locomotion in dairy cows. In: Proceedings of International Conference of Agricultural Engineering CIGR-Ageng 2012, p. C0438.

Zhang, B., Wang, L., Wang, Z., Qiao, Y. and Wang, H. 2018. Real-Time Action Recognition with Deeply-Transferred Motion Vector CNNs. IEEE Transactions on Image Processing, 27(5), pp.2326-2339.

Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. 2017. Pyramid Scene Parsing Network. arXiv:1612.01105v2.

Zhu, Y., Lan, Z., Newsam, S. and Hauptmann, A. 2018. Hidden Two-Stream Convolutional Networks for Action Recognition. arXiv:1704.00389v4 [cs.CV].

9 Appendix

9.1 Neural networks

Feed forward neural networks as illustrated in Figure 9.1 are multi-layered fully connected networks. Neural networks consist of many neurons (also called nodes/units) which are organised in layers, where each neuron in a layer has a weighted connection to all the neurons from the previous layer. Neural networks consist of an input layer, multiple hidden layers and an output layer. Data enters at the inputs and forward propagates through the network layer by layer until it arrives at the output.

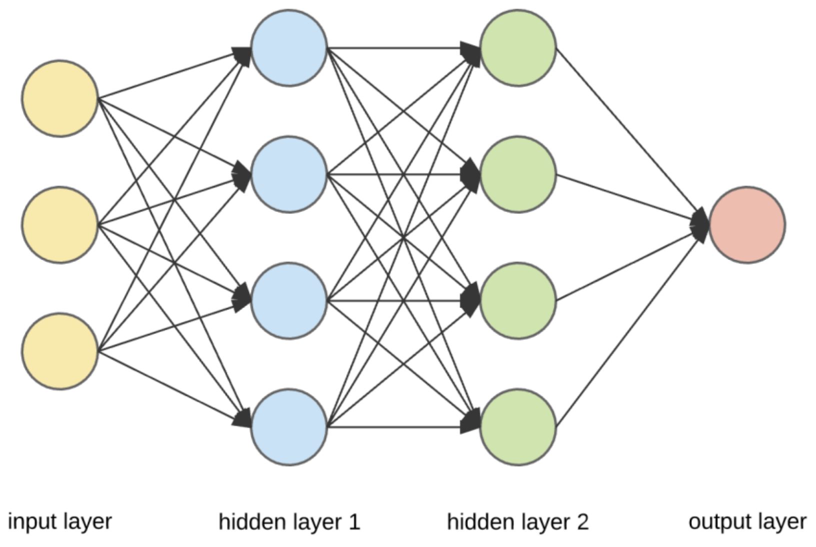

Figure 9.1. A 3-layer Neural network with three inputs, two hidden layers with four neurons in each layer and an output layer with one neuron.

A neuron as shown in Figure 9.2 takes the weighted sum of its inputs, offset by a bias and passes the resulting scalar value through a non-linear activation function. The output of the neuron is called an activation. The weights and biases are learned during training.

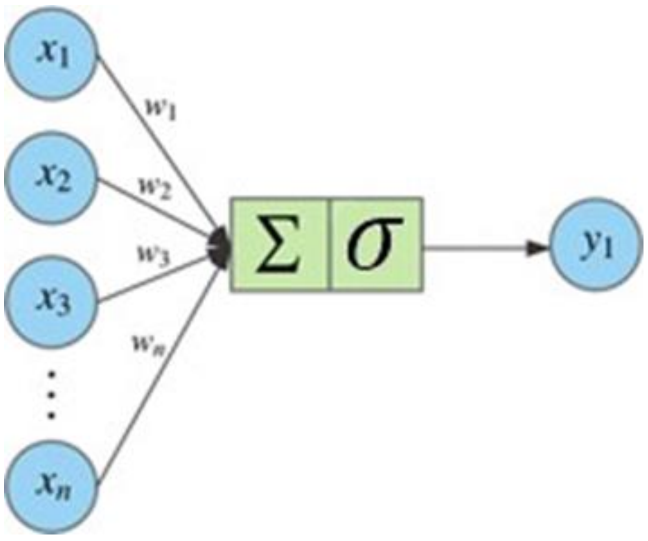

Figure 9.2. Schematic representation of an artificial neuron.

The equation for a neuron is given as follows:

$$y = \sigma\left(b + \sum_{i=1}^{n} w_i x_i\right) = \sigma(b + w^T x) \tag{9.1}$$

were x is the input to the node, w are the associated weights, b is the bias and σ represents the non-linearity function.

9.2 Convolutional neural network (CNN)

, plus it would take a huge number of parameters to characterise the network. Convolutional layers on the other hand have shared parameters. Plus, each neuron will only have local connectivity to a local region of the whole input volume. The extent of the connectivity along the depth axis is the same as depth of the input volume. This is known as the local receptive field of the neuron.

Kernel weight parameters are shared across all receptive fields. The depth of a kernel is the same as the number of channels in the input volume and the number of channels in the output volume equals the number of kernels used. Each kernel detects different features.

$$conv(I,K)_{xy}^{l} = \sigma\left(b^{l} + \sum_{m=1}^{h}\sum_{n=1}^{w}\sum_{c=1}^{d} K_{mnc}^{l} \cdot I_{x+m-1,y+n-1,c}\right) \tag{9.2}$$

As shown in equation (9.2), the activation for each neuron in a layer l is calculated by taking the dot product of the kernel K and its associated receptive field I, then offsetting the result with a bias b which is then passed through a non-linearity σ. h, w and d denote the height, width and depth of the volume.

9.3 Nonlinear activation functions

The purpose of nonlinear activation functions is to introduce non-linearity into the network, therefore extending the kind of functions that can be represented within the network. If we were to use just linear activation functions, then no matter how many hidden layers are used, the network would behave like a single perceptron, as the sum of the layers would just give another linear function. Below are the main types of nonlinear activation functions.

9.3.1 Sigmoid

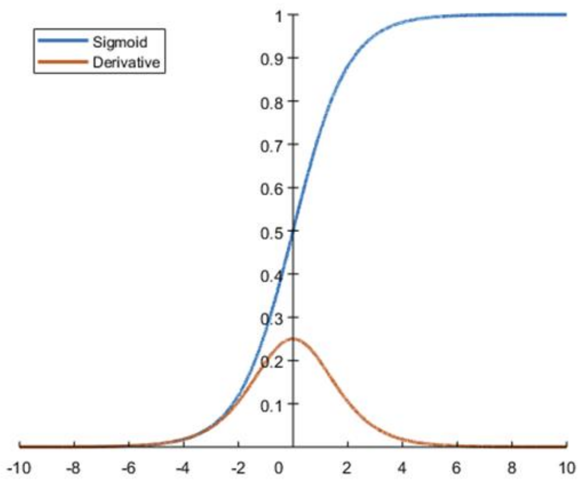

Figure 9.3. Sigmoid (blue) and its derivative (red).

The sigmoid function also known as the logistic function which is shown in equation (9.3) and its derivative in equation (9.4) takes a real number as input and maps the output value in the range [0, 1].

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \tag{9.2}$$

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x)) \tag{9.3}$$

As can be seen in Figure 9.3, the sigmoid function saturates at either end which gives rise to the problem of vanishing gradients (gradients within these regions is almost zero) which causes the network to either not converge or learn very slowly. Sigmoid is mainly used as a binary classifier on the output layer.

9.3.2 Hyperbolic tangent (Tanh)

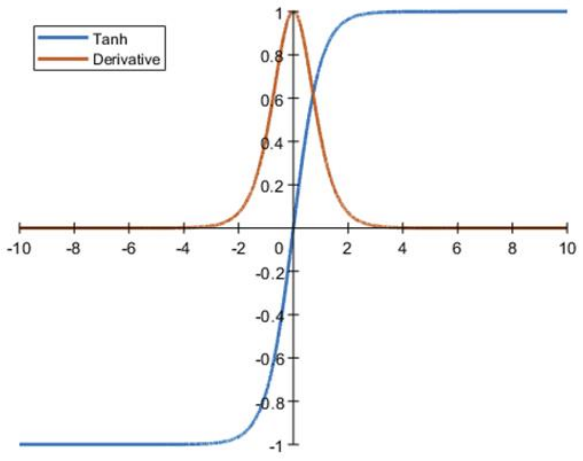

Figure 9.4. Tanh (blue) and its derivative (red).

The tanh activation function which is shown in equation (9.5) and its derivative in equation (9.6) is similar to the sigmoid function, but as can be observed in Figure 9.4, it maps the output values to the range [−1, 1].

$$f(x) = tanh(x) = \frac{sinh\,(x)}{cosh\,(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{9.4}$$

$$\frac{d}{dx} tanh(x) = 1 - tanh^2(x) \tag{9.5}$$

Like the sigmoid function, it also suffers from vanishing gradients, but unlike sigmoid, optimization is easier as its output is zero-centred. In recent years tanh has been replaced by ReLU as the main activation function used in convolutional neural networks (CNNs).

9.3.3 ReLU

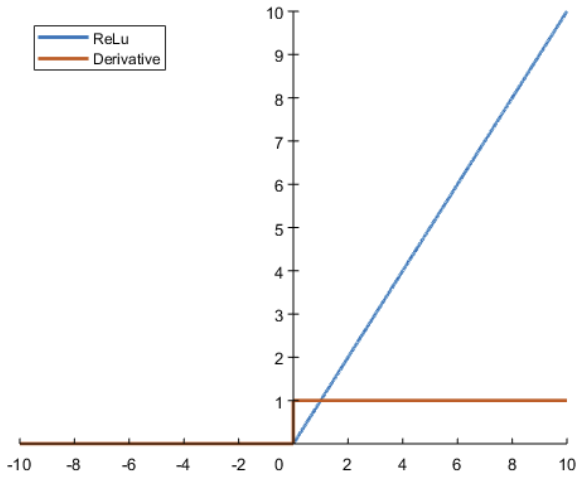

Figure 9.5. ReLU (blue) and its derivative (red).

The Rectified linear unit (ReLU) was originally proposed by Nair and Hinton (2010). It has been shown by Krizhevsky et al. (2012) that using ReLU activations to train deep convolutional neural networks is six times faster than using equivalent tanh units. ReLU is simple to implement and is shown in equation (9.7) with its derivative in equation (9.8). The output values are in the range [0, inf] and as can be seen in Figure 9.5, ReLU does not suffer from vanishing gradients.

$$f(x) = \begin{cases} x \text{ for } x \geq 0 \\ 0 \text{ for } x < 0 \end{cases} = \max(0, x) \tag{9.6}$$

$$\frac{d}{dx} f(x) = \begin{cases} 1 \text{ for } x \geq 0 \\ 0 \text{ for } x < 0 \end{cases} \tag{9.7}$$

ReLU does not activate all the neurons at the same time as negative input values will be converted to zero, which causes the gradient to also be zero. Because of this the weights do not get updated during backpropagation and effectively create dead neurons. This is actually beneficial, as it creates a sparse and efficient network. The limitation of ReLU is that it should only be used within the hidden layers of a network.

It is possible for too many neurons to become dead neurons, this is known as the dying ReLU problem. Leaky ReLU, shown in equation (9.9) and its derivative equation (9.10), addresses this issue by multiplying negative values by a small constant value a = 0.01, which gives the neuron a chance to recover.

$$f(x) = \begin{cases} x \text{ for } x \geq 0 \\ ax \text{ for } x < 0 \end{cases} \tag{9.8}$$

$$\frac{d}{dx} f(x) = \begin{cases} 1 \text{ for } x \geq 0 \\ a \text{ for } x < 0 \end{cases} \tag{9.9}$$

9.3.4 Softmax

The problem with a sigmoid function is that it is only a binary classifier. To handle the probability distribution of multiple classes, the softmax activation function, which is shown in equation (9.11), is generally used. Softmax takes a N-dimensional vector of real numbers and transforms it into a vector of real numbers within the range [0,1], where the sum of the resultant vector has the value of one.

$$f(x_c) = \frac{e^{x_c}}{\sum_{i=1}^{n} e^{x_i}} \tag{9.10}$$

9.4 Receptive Field

Every neuron in a convolutional layer is assigned a local region on the activation map produced from the previous layer. This local region is called the receptive field. The receptive field of the first convolutional layer is the kernel size. The effective receptive field of deeper layers with respect to the input image, depends on kernel sizes and convolutional strides of all previous convolutional layers. It is possible to keep track of the effective receptive field for each convolutional layer by using the following equation:

$$r_{out} = r_{in} + (k - 1) * d \tag{9.11}$$

where r denotes the receptive field, k equals the kernel size and d is the distance between two cells in the kernel. The first convolutional layer is the input layer and always has r_{in} = 1 and d = 1.

Luo et al. (2017) has shown that the effective receptive field does not have a uniform distribution, but in fact has a gaussian distribution which only occupies a small fraction of the full theoretical receptive field.

9.5 Zero padding

Given a kernel with size greater than one, if the convolution layer had no padding, then the spatial dimension of the resultant activation maps would decrease with each convolution. Zero padding is added to preserve the spatial dimensions of the activation map. This allows us to design deeper networks and improves performance by keeping information at the borders. The formula to calculate padding p is as follows, assuming the kernel k is square.

$$p = \left\lfloor \frac{k}{2} \right\rfloor \tag{9.12}$$

Knowing the padding we can then calculate the dimensions of the output volume as follows:

$$I_{out} = \left\lfloor \frac{I_{in} - k + 2p}{s} \right\rfloor + 1 \tag{9.13}$$

where l_{in} and l_{out} are the number of input and output features and s is the convolution stride size.

9.6 Max Pooling

In CNNs max pooling is mainly used to reduce the spatial dimension of feature maps by eliminating non-maximal values. This helps reduce computation on the network, reduces overfitting and makes the model resilient to minor shifts in position and rotation.

Max pooling consists of splitting the input into bins and returning the maximum value from each bin. There are other types of pooling such as average pooling which takes the mean of each bin as output and L2-norm pooling where the output is the square root of the sum of the values squared.

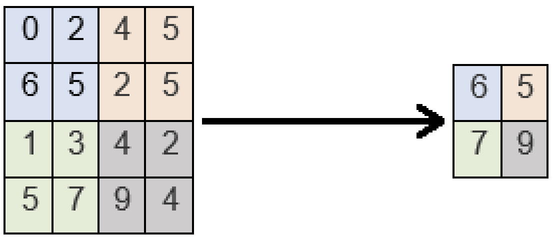

Figure 9.6. Max pooling with a kernel of size 2x2 and a stride of 2.

A pooling layer generally has a kernel and stride of similar length. Figure 9.6 shows a 2x2 kernel with a stride of 2 applied to down-sample a 4x4 feature map. Equation (9.15) shows the formula to calculate the output dimensions of a feature map. The depth dimension does not change.

$$I_{out} = \left\lfloor \frac{I_{in} - k}{s} \right\rfloor + 1 \tag{9.14}$$

where I_{in} and I_{out} are the number of input and output features, k is the kernel size and s is the convolution stride size.

9.7 Batch normalisation

Batch normalisation (BN) proposed by Ioffe and Szegedy (2015) is a technique for improving the performance and stability of neural networks. The idea behind BN is that instead of just normalising the input to the network, we also for each mini-batch normalise each hidden layer independently within the network (zero mean and unit variance). A learned offset β and scaling factor y are then applied, which can provide the ability to recover the original output of the current layer, if needed. Equation (9.18) shows batch normalisation, the mean μ and variance σ^2 are calculated per-dimension over the min-batch.

$$\mu = \frac{1}{m} \sum_i z_i \tag{9.15}$$

$$\sigma^2 = \frac{1}{m} \sum_i (z_i - \mu)^2 \tag{9.16}$$

$$\tilde{z}_i = \frac{z_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} * \gamma + \beta \tag{9.17}$$

where m is the size of the mini-batch, z_i is the values of the hidden layer before the activation function and ε is a small positive constant to prevent division by zero.

During inference, the mean and standard-deviation for each activation is computed on the whole training dataset, which replaces the ones computed on mini-batches. It also has the added benefit of helping prevent sigmoid and tanh activations from saturating.

9.8 Fully-connected layer

Fully-connected layers (also known as Dense layers) in a convolutional network are mainly used for classification. Similar to a multi-layer perceptron, each neuron in the current layer is connected to all the weighted activations from the previous layer. For classification a softmax activation function is used in the output layer.

9.9 Weight initialisation

Having a proper weight initialisation strategy is critical to how well a model converges. Initialising a network with arbitrary weights can slow down or completely stall the learning process. If the weights are initialised too small, then the variance of the input signal starts diminishing as it gets deeper into the network. Likewise, if they are initialised with too large a value, the variance increases rapidly with each passing layer.

9.9.1 Xavier initialisation

Glorot and Bengio (2010) proposed an initialisation scheme called Xavier which enabled learning of deep networks to converge substantially faster than previous methods. Xavier tries to make the variance of the outputs of a layer to be equal to the variance of its inputs and is given by the equation:

$$Var(W) = \frac{1}{n_{in}} \tag{9.18}$$

where $Var(W)$ is the variance of the weights for a layer, initialised with a zero mean normal distribution and n_{in} is the number of weights connected to the neuron, from the previous layer.

9.9.2 MSRA initialisation

MSRA, which was proposed by He et al. (2015) extends Xavier initialisation for layers that are followed by a ReLU non-linear activation. Because ReLU is zero for half of its input, to keep the variance constant, MSRA doubles the size of the weight variance and is given by the following equation:

$$Var(W) = \frac{2}{n_{in}} \tag{9.19}$$

9.10 Loss functions

During training we supply our model with training data and its associated ground truth labels. The training data is forward propagated through the network and the output is the predicted probability. The ground truth label for the associated training data and the predicted output are then passed into a loss function which calculates the error. The loss is the error between what the network is predicting for the input data against what the ground truth label for that data actually is. There are many loss functions, below are two examples.

9.10.1 Cross-entropy loss

Cross-entropy loss measures how alike two probability distributions are. The ground truth distributions are generally expressed in terms of a one-hot distribution. The equation is given below

$$L = -\sum_i y_i' \, log(y_i) \tag{9.20}$$

where y' is the ground truth probability vector and y corresponds to the predicted probability vector. The loss L is how far away the prediction is from the ground truth.

9.10.2 Smooth L1 loss

The smooth L1 loss is used for regression. It is similar to the mean squared error loss but is more robust to the presence of outliers. It uses a squared term if the absolute element-wise error goes below one, otherwise a L1 term is used. Unlike L1 loss, smooth L1 loss is differentiable when the element-wise error is zero and is defined as follows:

$$smoothL1(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}0.5 * (y_i - \hat{y}_i)^2, & if\ |y_i - \hat{y}_i| < 1\\ |y_i - \hat{y}_i| - 0.5, & otherwise\end{cases} \tag{9.21}$$

where y is the vector of ground truth values, $\hat{y}$ is the vector of the predicted values and n equals the size of the vector.

9.11 Backpropagation

Backpropagation is the tool that gradient descent uses to calculate the gradient of the loss function. For each training example, the network computes the predicted output and all its associated losses. Next, all the losses are summed up to generate the total error. Then the backpropagation algorithm uses the chain-rule from calculus, to compute the partial derivatives $\frac{\partial E}{\partial w}$ of the cost function E with respect to each weight w in the network.

Given a neuron which is defined as follows:

$$a_j^k = \sigma(z_j^k) = \sigma\left(\sum_l w_{lj}^k a_l^{k-1}\right) \tag{9.22}$$

where σ is the non-linear activation function and z_j^k is the weighted sum of the outputs a from the previous layer. The partial derivative of the error function E with respect to weight w_{ij}^k can be solved using the chain rule as follows:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial z_j^k} * \frac{\partial z_j^k}{\partial w_{ij}^k} \tag{9.23}$$

If we denote the error term as follows:

$$\delta_j^k \equiv \frac{\partial E}{\partial z_j^k} \tag{9.24}$$

and the partial derivative of the second term is calculated by:

$$\frac{\partial z_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_l w_{lj}^k a_l^{k-1}\right) = a_i^{k-1} \tag{9.25}$$

Then, the partial derivative of the error function E with respect to a weight w_{ij}^k is given by:

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k a_i^{k-1} \tag{9.26}$$

Where the error term δ_j^k is node j in layer k and output a_i^{k-1} is node i in layer $k-1$.

Given the network in Figure 9.1, we can calculate the partial derivative of the error function E to the weights in the output layer as follows:

$$\delta_1^k = \left(\sigma\left(z_1^k\right) - y\right)\sigma'\left(z_1^k\right) = (\hat{y} - y)\sigma'\left(z_1^k\right) \tag{9.27}$$

$$\frac{\partial E}{\partial w_{i1}^{k}} = \delta_1^k a_i^{k-1} = (\hat{y} - y)\sigma'\left(z_1^k\right)a_i^{k-1} \tag{9.28}$$

Then, for each hidden layer, we can recursively use the following formula, as we backpropagate thought the network:

$$\delta_j^k = \sum\nolimits_l \delta_l^{k+1} w_{jl}^{k+1} \sigma'\left(a_j^k\right) = \sigma'\left(z_j^k\right) \sum\nolimits_l w_{jl}^{k+1} \delta_l^{k+1} \tag{9.29}$$

$$\frac{\partial E}{\partial w_{ij}^{k}} = \delta_j^k a_i^{k-1} = \sigma'\left(z_j^k\right) a_i^{k-1} \sum\nolimits_l w_{jl}^{k+1} \delta_l^{k+1} \tag{9.30}$$

9.12 Stochastic gradient descent

To train a network model we are essentially trying to solve an optimisation problem. We are trying to optimise the weights within the model. The most widely used optimization algorithm is called stochastic gradient descent (SGD) and the particular variant of SGD that is used is called mini-batch gradient descent. The objective of SGD is to, in an iterative manner, minimise the error for any given loss function based on that function's gradient. The update rule is given by the following formula:

$$w^{t+1} = w^t - \eta \nabla \mathrm{f}(w^t) \tag{9.31}$$

where η is the learning rate, $\nabla f(w)$ is the gradient of the loss function $f(w)$ with respect to the weights w and the iteration is given by the notation t.

9.12.1 Learning rate

The learning rate determines the size of the steps the algorithm takes down the gradient on the error curve. Too small a rate leads to slow convergence. While too large a value, can cause the loss function to fluctuate around the minima or even diverge.

9.12.2 Momentum

Momentum is an extension to SGD which uses a moving average gradient instead of the immediate gradient at each time step. Its purpose is to help reduce the risk of SGD getting trapped in local minima. It also helps dampen oscillations, which leads to faster convergence of the loss function. Update equations of momentum for iteration t is shown as follows:

$$v^{t+1} = \gamma v^{t} - \eta \nabla \mathrm{f}(w^{t}) \tag{9.32}$$

$$w^{t+1} = w^{t} + v^{t+1} \tag{9.33}$$

where v is the velocity vector, γ is the momentum coefficient with range [0,1], η is the learning rate and $\nabla \mathrm{f}(w^{t})$ is the gradient of the loss function.

9.12.3 Weight decay

In SDG, weight decay is a regularisation term that causes the weights to decay in proportion to their size. This ensures the weights stay small. Which is crucial for avoiding overfitting. SGD with momentum and weight decay is described as follows:

$$v^{t+1} = \gamma v^{t} - \lambda\eta w^{t} - \eta\nabla f(w^{t}) \tag{9.34}$$

$$w^{t+1} = w^{t} + v^{t+1} \tag{9.35}$$

where weight decay is given by the term $-\lambda\eta w^{t}$ where λ is the weight decay coefficient.

9.13 CNN architectures

There have been a lot of CNN architectures developed since the 1990's. In this section, we describe the most well-known ones, along with their contributions.

9.13.1 LeNet-5

The use of CNN's can be dated back as early as the 1990's with one of the best known architectures of the time being LeNet-5 which was proposed by LeCun et al. (1998). LeNet-5 was used for digit recognition.

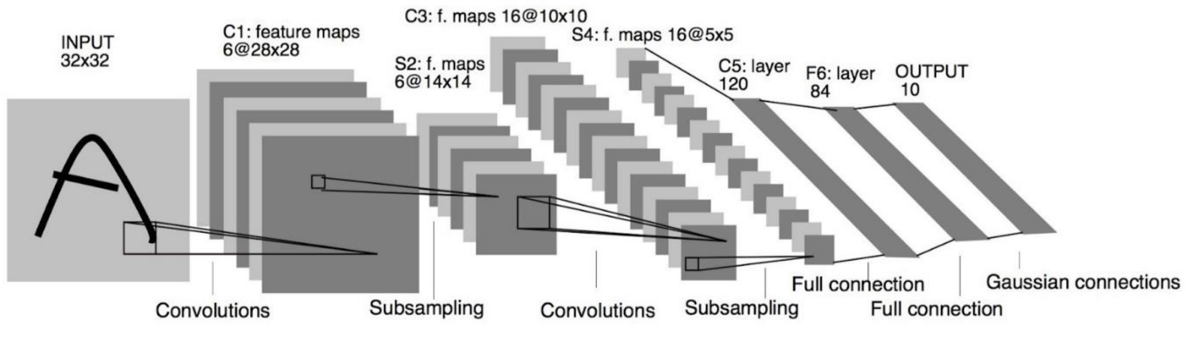

Figure 9.7. Lenet-5 CNN architecture, used for digit recognition (Source: LeCun et al. (1998)).

As illustrated in Figure 9.7, the architecture of Lenet-5 consists of 7 layers, where each convolutional layer is followed by a pooling layer and a non-linearity. Fully connected layers are used as the final classifier. To help accelerate learning, the input pixels were normalised with zero mean and unit variance.

9.13.2 AlexNet

The seminal work of Krizhevsky et al. (2012) which introduced the deep convolutional network Alexnet, popularised the use of convolutional networks within the field of computer vision. In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 they placed first with a test error rate which was over 10% better than the second-best entry.

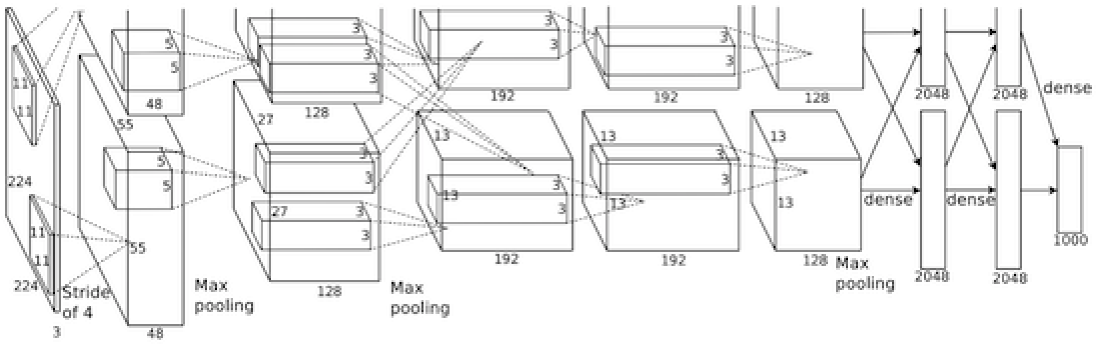

Figure 9.8. Alexnet CNN architecture. Training is done on 2 GPUs. One GPU executes the top half of the diagram and the other the bottom half. (Source: Krizhevsky et al. (2012)).

Alexnet, shown in Figure 9.8, consists of five convolutional layers and three fully-connected layers. It popularised the use of ReLU as a non-linearity, as it showed improved performance over tanh. To help reduce overfitting they used overlapping pooling and a regularisation method called dropout (Srivastava et al., 2014) in the fully connected layers. The third, fourth and fifth convolutional layers and their non-linearities are stacked together.

9.13.3 VGG Net

VGG Net by Simonyan and Zisserman (2014) showed that the depth of a network was a critical factor for good performance. They also introduced the use of using multiple 3x3 convolutional/non-linearity layers in sequence to increase the effective size of the receptive field. This made the decision function more discriminative and also decreased the number of parameters.

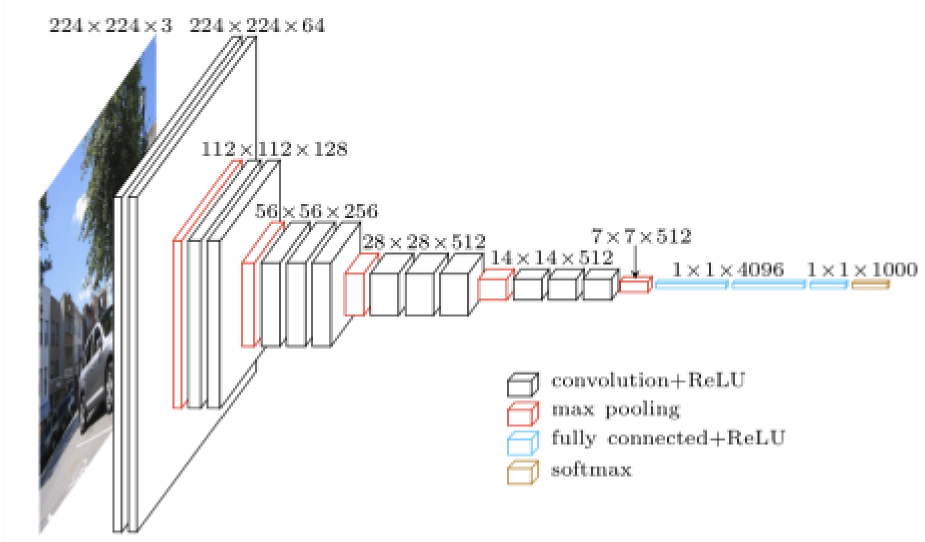

Figure 9.9. VGG16 Net architecture consists of 13 convolutional layers and 3 fully connected layers used for classification. (Source: Heuritech (2016)).

Figure 9.9 shows VGG16 Net, the deeper variant VGG19 Net consisted of 16 convolutional layers and 3 fully connected layers used for classification.

9.13.4 GoogleLeNet

Another notable CNN architecture is GoogleLeNet which was proposed by Szegedy et al. (2014). GoogleLeNet achieved a top-5 error rate of 6.67% and ranked first in the ILSVRC 2014 challenge. Their network was 22 layers deep and comprised of, at its core, what they term Inception modules. There are nine of these modules which are stacked on top of one another with occasional max-pooling layers to reduce the dimensionality of the feature maps. It is interesting to note that GoogleLeNet uses twelve times fewer parameters than AlexNet.

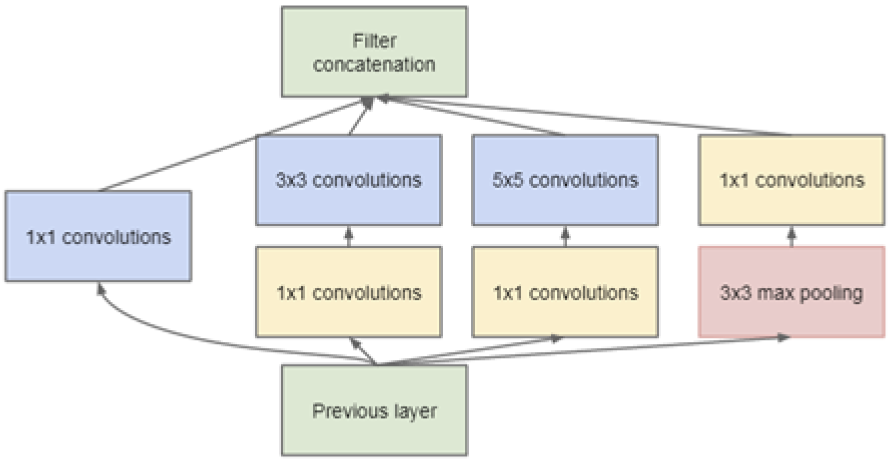

Figure 9.10. Inception module (Source: Szegedy et al. (2014)).

As can be observed in Figure 9.10, the inception module is a set of parallel convolution and pooling layers. Similar to Lin et al. (2013) they use 1x1 convolutional layers to reduce the dimensionality of inputs to the more expensive larger kernel layers. This method of reducing the dimensionality is commonly referred to as a bottleneck. The output from each of the convolutional pathways is then concatenated together and used as input to the next inception module.

9.13.5 Resnet

As networks got deeper, training them effectively becomes challenging due to vanishing and exploding gradients which results in the network's inability to learn the desired mapping between inputs and outputs efficiently. Another observed phenomenon is the degradation problem, where the training accuracy would saturate and then quickly degrade. This was due to the added layers not being able to learn the identity mapping effectively. To solve these problems, He et al. (2015) proposed the residual learning framework (ResNet). They trained networks with a depth of up to 152 layers, while still having a lower complexity than VGG. Using an ensemble of residual networks, they achieved a 3.57% top-5 error on the ImageNet test set and was the winning architecture in the ILSVRC 2015 classification competition.

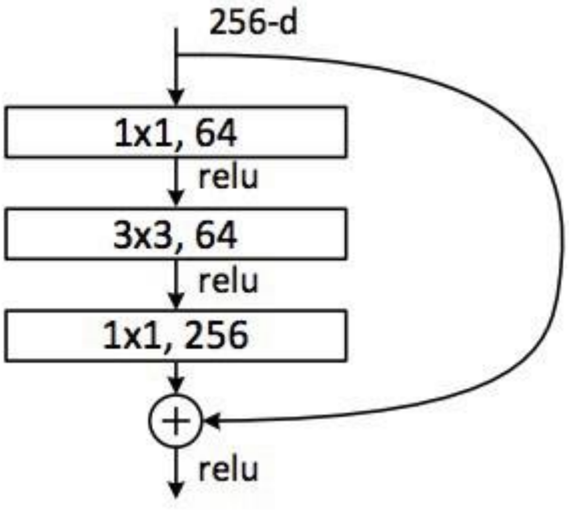

Figure 9.11. Residual learning. Bottleneck building block used in ResNet-50/101/152 (Source: He et al. (2015)).

As illustrated in Figure 9.11, ResNet-50/101/152 uses 3-layer bottleneck building blocks with shortcut connections. An identity shortcut is used if input and output dimensions are equal, otherwise a projection shortcut is used to match dimensions. To reduce the complexity, they first use a 1x1 convolutional layer to reduce the depth to a quarter of the input, this is followed by a 3x3 convolutional layer. Then, another 1x1 convolutional layer is used to restore the original depth. Finally, an element-wise addition is used to join the identity shortcut connection to the output of the block. Shortcut connections allow the output of one layer to bypass one or more layers and be summed up with the output of a subsequent layer. This helps in creating a kind of direct link between the input and output, making it easier for the network to learn the identity function when needed. Shortcut connections do not introduce additional parameters or computational complexity.

9.13.6 ResNeXt

Recently, Xie et al. (2017) proposed a multi-branch architecture based on ResNet which they named ResNeXt. Their building block structure follows a split-transform-merge paradigm, similar to that of Szegedy et al. (2014) but with some key differences. Notably, as illustrated in Figure 9.12, the topology of each pathway is identical and the outputs from the pathways are merged together via element-wise addition.

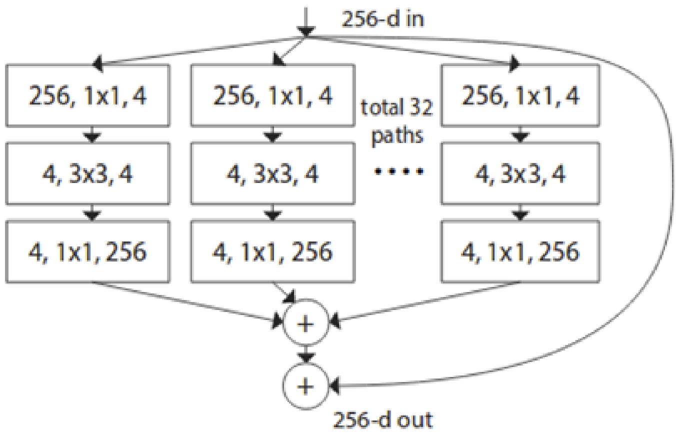

Figure 9.12. ResNeXt building block with cardinality equal to 32. A layer is shown as number of input channels, kernel size and number of output channels (Source: Xie et al. (2017)).

To adjust the number of pathways, the authors introduced a new hyper-parameter which they termed cardinality. Experiments done by the authors show that accuracy can be gained more efficiently by increasing cardinality, compared to going deeper or wider. ResNeXt placed 2nd in the ILSVRC 2016 classification competition with an ensemble result of 3.03% top-5 error on the ImageNet test set.