UTRECHT UNIVERSITY

Graduate School of Natural Sciences

Computing Science Master Thesis

Automated Analysis of Dairy Cow Drinking Behaviour Using Computer Vision: Developing and Integrating Cow Detection and Drinking Behaviour Classification Components

First examiner:

Dr. Miel Hostens

Second examiner:

Prof. Dr. Albert Ali Salah

Candidate:

BSc. Douwe de Kok

In cooperation with:

Faculty of Veterinary Medicine

July 24, 2024

Acknowledgements

This work aims to lay the foundation for the development of a system able to measure the drinking behaviour of lactating cows. This system can be used to assess the relationship between the water intake of lactating cows and their milk production. The first components will be developed, tested, analysed and optimized. These essential components lay the groundwork and are integrated into the system and are awaiting expansion of the identification components. This project finally aims to provide the data and the system needed for the subsequent statistical analysis of the relationships between drinking behaviour and milk production.

The contributions of the initial student of this project, Daniel van Herwijnen, have been tremendous. He kick-started our project by laying the groundwork for the identification component. He proposed a novel approach using a Siamese neural network. Most of the model was built by him thereby enabling the identification of the cows. However, Daniël's analysis revealed a critical hurdle: the model's performance was significantly influenced by the background of the images. This led to the recommendation to employ instance segmentation for background removal in training images. This suggested a refinement of the model's accuracy. This has thereafter been implemented by Matteo di Vincenzo.

Matteo di Vincenzo's work has been a solid contribution to the project, bringing practical improvements that have enhanced the performance of the model and provided documentation to improve the understanding of the model. He introduced randomized augmentation for image processing, and segmentation of the cows to focus to model's attention on the patterns of the cows instead of the background. Furthermore, Matteo focused on improving the documentation of the project. He also made the directory structure more adaptable and worked on moving the project to a cloud-based setup, aiming to increase its scalability and accessibility.

The project builds on the supervision and guidance of Dr. Miel Hostens.

He initialized the project and provided unwavering support throughout the process from beginning to end. His understanding of the project and undisputed optimism have been a motivational boost at demanding times. His interest in the finalization of the system has given the drive to push the boundaries of what could be completed within the set time. Unfortunately, as initially set out, the finalization of the system could not be accomplished. However, the detection components have been successfully integrated into the system. Moreover, the identification components have been added, but are not rigorously tested, thus this will be left for future endeavours.

Without the crucial knowledge of Prof. Dr. Albert Ali Salah, the quality of the components would be unsatisfactory at most. His indispensable insight provided the guidance needed to elevate the project to the level needed for a state-of-the-art computer vision system. His academic background on the subject raised the bar for the project to a level where it could effectively contribute to the field of computer vision by providing new insights.

Lastly, I want to thank the people around me who always supported me when I needed it. Writing a thesis is hard work and requires constant motivation and dedication. Without the help of these people, I would not be able to deliver the work I have produced.

Finally, I want to state that this journey has been one of the most educational fulfilling, but demanding times I have experienced in my life. My interest in computer vision has been tremendous since I first came in contact with the subject, but this project has made me decide to pursue this path for my career. I hope this subject will be an inspiration for many others to come.

Abstract

Monitoring the drinking behaviour of dairy cows provides valuable insights into their health and welfare. However, establishing the relationship between water intake and factors like milk production has been challenging due to limitations in data collection and the amount of research on this subject. Computer vision offers a promising solution for automated monitoring of cow drinking behaviour.

A system for cow detection and drinking behaviour classification using deep learning techniques was presented. A YOLOv10 model achieved 99.1% AP-50 and 87.1% AP50-95 for cow detection, while an EfficientNetV2-S model attained 88.6% accuracy for binary classification of drinking behaviour. When tested on a 1-hour video, the system measured drinking time with 92.5% precision and 92.0% recall, demonstrating its effectiveness for automated analysis.

Integration of this system with cow identification components will enable monitoring of individual free water intake, providing valuable data for studying the relationship between free water intake and milk production. The rigorous testing and evaluation conducted in this work pave the way for practical application in precision livestock farming.

Future research should explore the addition of spatial-temporal components to further improve performance and investigate the impact of different camera viewpoints. Ultimately, this system contributes to the advancement of animal welfare and farm management by enabling detailed analysis of drinking behaviour.

Contents

1. Introduction

1.1 Problem Statement

In recent years, the dairy industry has witnessed a significant surge in milk production [1–6] and consumption worldwide [7, 8]. This increase, while beneficial in meeting global demand for dairy products, has raised concerns regarding its impact on reproductive capacity [5, 9], which might be caused by a level of dairy production inconsistent with nutrient intake [10], health issues [3, 5], and reduced life expectancy [5].

A proven way to enhance cow welfare is by using Artificial Intelligence (AI) to monitor metabolic indicators [11], animal behaviour [12], and detection of udder infection (mastitis) [13]. As these techniques are relatively new, not all their potential has been fully explored.

One of their underexposed use cases is measuring dairy cow water consumption, commonly referred to as Free Water Intake (FWI). As water is a crucial nutrient for cow health [14, 15], monitoring this helps optimizing milk production [16] understanding drinking behaviour [17] and their health [18].

However, the relation between their milk production and FWI is more complex. There are signs of a positive correlation between milk production and FWI [19–21], contrarily, others found no significant link [22].

Moreover, further assessment of milk production-related traits of dairy cows might significantly reduce environmental impact, as it has the potential to improve milk production per cow [13] and higher milk yield per cow reduces the overall greenhouse gas emissions [23, 24] and global warming potential [25].

Finally, AI can optimize herd monitoring [26–28], measure social interactions [29], analyze various behaviours [30] or more specific ones like feeding [31–33], and drinking [34–37] for instance.

Although there has been some research into the drinking behaviour of cattle [19–22], further research is crucial for finding the link between milk production and FWI, therefore, this data-driven approach aims to further answer this question.

1.2 System Pipeline

The system can be summarized in a few simple steps. These steps are given in Figure 1.1. First, the user starts by capturing videos of their cows by strategically placing the cameras. Then, the videos are to be sent to the detection model for initial processing. The detection model then extracts sections from the original images containing drinking cows and sends them to the identification model for the next step in the pipeline. The identification model processes each image it receives by identifying the cow depicted in the image. Subsequently, these number of consecutive images are counted for each of the cows to obtain the total time each cow has spent drinking. This data can then be converted into a CSV file and returned to the user.

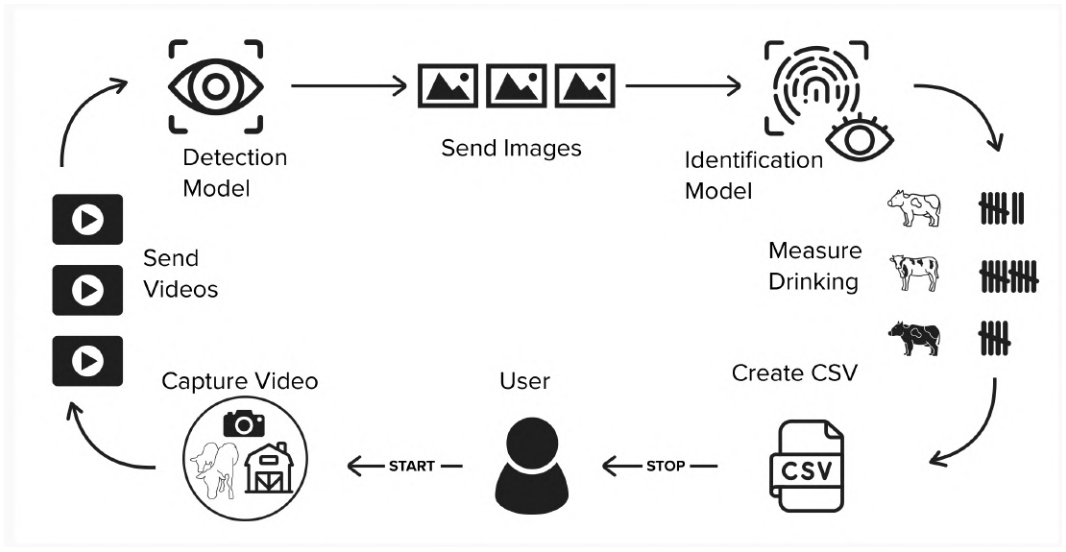


Figure 1.1: Pipeline of the dairy cow water intake system.

This system can be viewed as an integration of a two-phased system. The first phase handles the detection of the objects of interest, drinking cows in our case, whereas the second phase deals with the identification.

Data for the first phase, the detection of drinking dairy cows, is obtained from video footage captured at a dairy farm by utilizing two separately trained AI models. The first model detects each individual cow. This model is based on the open-source YOLO (You Only Look Once) computer vision framework by Ultralytics [38]. Subsequently, these detections are passed to the second model which asserts whether the cow is displaying drinking behaviour. This behaviour classification model uses the architecture of the model that was the most accurate after evaluation; EfficientNetV2 [39].

The second phase encompasses the identification of the cows in the video. This identification phase contains two consecutive components as well. First, we segment the cow by removing the background to enhance the identification process. The segmentation is done with a model from the YOLO framework as well. Second, the identity is determined by a Siamese Neural Network (SNN) (Koch et al. [40]), which is based on an InceptionResnetV2 [41] backbone.

1.3 Research Contribution

As we have shown in the previous section, the relation between the drinking behaviour of individual cows and milk production is still unclear. There has been research on water intake [17, 42–45], and some research conducted on drinking duration [34]. Research into the drinking behaviour and drinking duration of dairy cows and its relation to milk production remains insufficiently examined. That is, except for various older research that do not agree with each other on the results.

For instance, Winchester and Morris [19] expected a water intake of .87 kg per kg of milk, based on its water content. In line with these findings, Murphy et al. [21] reported a coefficient of .64 kg of water/kg of milk, which adjusted to .90 when more variables were included, aligning closely with the anticipated .87 kg of water/kg of milk. Little and Shaw [20] identified a regression coefficient of .73 kg of water/kg of milk over a production range of 14 to 30 kg/day, however, they considered dry matter intake (DMI) effects as well.

Contrarily, Paquay et al. [22] examined the correlation between total water intake and found no significant link between milk production and water consumption.

According to these various outcomes, it becomes apparent that the relationship between free water intake and milk yield remains inconclusive. Therefore, this research aims to thoroughly explore the potential of a system that can measure the free water intake of individual cows, to be able to further assess its influence on their milk production.

It is important to note that this study builds upon the work conducted by Daniel van Herwijnen and Matteo di Vicenzo. They both made tremendous contributions towards the identification model as described in the Acknowledgements section. Unfortunately, their model could not yet be integrated into the whole system, nor could it be thoroughly evaluated in this thesis.

Even though the final goal of this research is to create a complete pipeline, this thesis focuses on the first part of that system, the detection and drinking behaviour analysis of cows. For this reason, we have decided to remove the parts of the thesis that deal with identification. The final integration of the system will be left for future research.

By conducting this research, we contribute to the field by:

1. Curating a comprehensive dataset for cow detection and drinking behaviour classification, which can serve as a valuable resource for researchers and practitioners to develop, compare, and improve upon existing and future models and techniques in this field.
2. Evaluating the accuracy and generalizability of a dairy cow drinking behaviour monitoring system which can be used as the initial components for a system that continuously measures the free water intake (FWI) of each individual cow.
3. Estimating the current Technology Readiness Level (TRL) of the cow drinking behaviour analysis system and provide recommendations for advancing it to the next level.
4. Contributing new knowledge to the field of cow drinking behaviour analysis using computer vision techniques, thereby paving the way for future research and practical applications in precision livestock farming.

1.4 Research Questions

This research focuses on three objectives. Firstly, we aim to assess the capabilities of the detection phase of the system. Secondly, we will proceed by assessing whether it displays drinking behaviour. The third and final objective is to ascertain if the system is at a stage in which it can be used for real-world scenarios, or what needs to be done to achieve this stage. These three objectives together can be summarized as an overall research question:

"How can we accurately recognize the drinking behaviour of each individual cow in a herd of cows using computer vision?"

To answer this question we will answer several subquestions which are extracted from the objectives in the preceding paragraph. To be able to assess the capabilities of the detection phase of the system, and thereby achieve the first objective, we shall address the following question:

SQ1: "How can we accurately and robustly detect cows in videos?"

As our second objective is to determine whether each of the individual cows is displaying drinking behaviour, we will assess this by addressing the following question:

SQ2: "How can we reliably classify drinking and non-drinking behaviour of cows in videos?"

To be able to assess the final objective, the degree of adoptability of the system in real-world scenarios, we set out to identify both the current level of the system, as well as the steps needed to advance to the subsequent stage. Therefore, we will be addressing the following question:

SQ3: "How usable is the system in real-world scenarios, and what steps are required to enhance it further?"

1.5 Thesis Outline

The thesis is organised as follows. In Chapter 2 we outline the related work in the field of computer vision and herd monitoring. In Chapter 3 we describe the methods which we will use to answer the research questions. Subsequently, we provide the experiments that will be conducted that are in line with the method in Chapter 4. Thereafter, we discuss the results in Chapter 5, followed by the discussion in Chapter 6. We finalize this thesis with the conclusions in Chapter 7.

2. Related Work

2.1 Computer Vision Applications

Computer vision is an interdisciplinary field that focuses on enabling computers to interpret and understand the visual world. Computer vision seeks to automate tasks that require visual cognition, such as image and video analysis.

Core processes in computer vision include image classification, where algorithms learn to assign labels to images based on their content by learning spatial hierarchies of features [46]; object detection, which involves identifying and localizing objects within images [47]; and semantic segmentation, where the goal is to categorize each pixel of an image into a predefined class.

Advances in deep learning, particularly the use of Convolutional Neural Networks (CNNs), have had a significant impact on the field [48]. Since their breakthrough performance on the ImageNet challenge in 2012 [49], CNNs have become the backbone of most computer vision tasks. This technology is used in many applications [50], from facial recognition systems [51] and autonomous vehicles [52] to augmented reality [53], medical image analysis [54], and herd monitoring [31, 32, 34–37, 55–61]. This type of deep neural network is the primary technology used in this research.

2.2 Object Detection

Object detection is a fundamental aspect of computer vision that involves identifying and locating objects. The process builds upon CNNs, which are able to identify crucial attributes or patterns within images to differentiate objects (e.g., edges, textures) [62].

Objects in an image are located by identified features and bounded by bounding boxes. Subsequently, the objects within the boxes are classified according to the features in that region. This two-phase object detection is called two-stage detection [47]. This type is significantly more accurate, but slower compared to one-stage object detection [63].

2.2.1 One-Stage and Two-Stage Model Comparison

As our system is to be used in real-world scenarios, we have to make a trade-off between accuracy, given by average precision (AP) and speed. Research conducted by Garcia et al, [64] provides us with a clear picture of this trade-off between one-stage and two-stage object detection models, which is given in Figure 2.1.

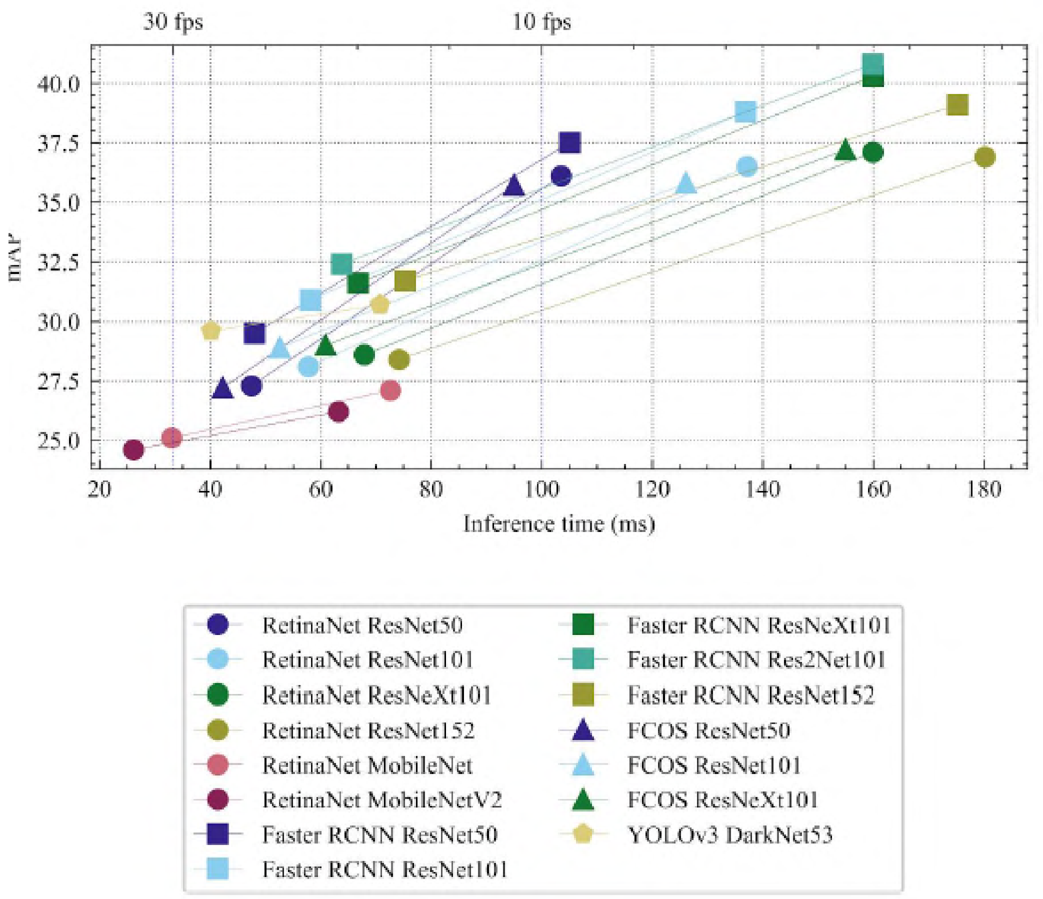


Figure 2.1: Detection speed of one-stage versus two-stage object detection models (adopted from Garcia et al (2021) [64]).

In Figure 2.1, each model is depicted twice, once for low-resolution images (640×960) and once for high-resolution (1280×1920). The inference time (ms) corresponds to the frames per second that the model can process. In this figure, we can see one-stage detection models; RetinaNet, FCOS and YOLO, and two-stage detection models; Faster RCNN.

From the figure we can observe that RetinaNet [65] with the smallest feature extractor, MobileNet and MobileNetV2 is the fastest in inference time (x-axis) for small images. However, it is also the least accurate as can be seen from the AP (y-axis). We can also observe that the RetinaNet versions that use substantially larger feature extractors have good accuracy and inference speed for small images, yet their speed more than doubles for large images.

The other one-stage detection model FCOS (Fully Convolutional One-Stage Detector [66]) has similar results as RetinaNet. The inference speed declines sharply for larger feature extractors and higher-resolution images.

The last one-stage model, YOLO (You Only Look Once [38]), has superior accuracy compared to the other one-stage models for low-resolution images. However, its accuracy remains approximately the same for high-resolution images while its inference time almost doubles.

The two-stage models have slightly better accuracy than the YOLO model on low-resolution images, but their speed is significantly slower. In terms of performance, they excel at high-resolution images, however, their speed is the worst in comparison to the one-stage models.

As our task is to process videos in real-time, there should be a reasonable balance between accuracy and speed. From this paper, we can conclude that two-stage models, which have all been shown to be slower and even less accurate than the YOLOv3 model, are not suitable for our goal.

2.2.2 YOLO Detection Model

Now that we have ruled out two-stage detection models, we further look into the promising YOLO detection model. The previous section compared one-stage to two-stage models and used a YOLOv3 one-stage detection model. This model seemed to be superior in terms of the balance between accuracy and speed for our task. However, since the field of computer vision is rapidly changing, the improvements to the model have been significant. They have released a new model almost every year since their first release of YOLOv1 in 2015 [38], which can be seen when reviewing Figure 2.2.

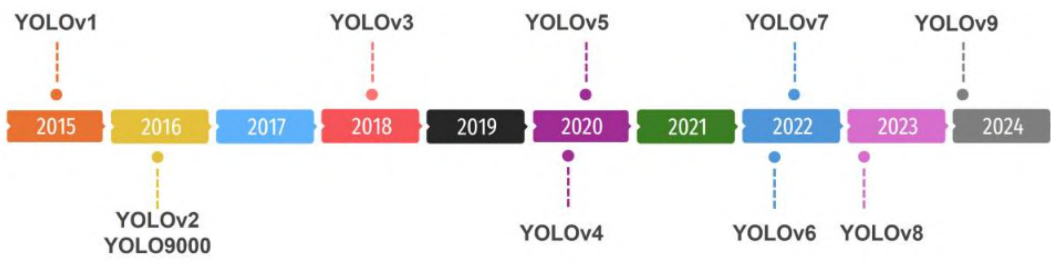


Figure 2.2: The timeline of the different YOLO models (adopted from [67]).

At the start of writing this thesis, they had not yet released YOLOv9, whereas they are currently at YOLOv10 [68], which shows their continuous dedication to the improvements of their models. It is therefore not surprising that the performance of their latest model exceeds the performance of all other detection models of comparable size and inference speed [38]. This comparison is depicted in Figure 2.3.

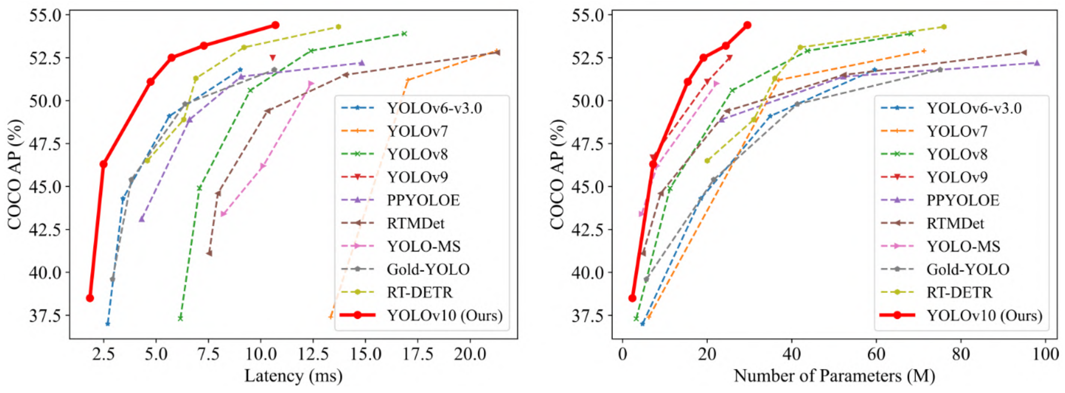


Figure 2.3: Comparison of object detection models to latest YOLO model (adopted from [69])

In previous iterations of their models, and in most commonly used object detection models [47], the obtained bounding boxes are refined through an algorithm called Non-Maximum Suppression (NMS), which eliminates bounding boxes that are unimportant [70].

Their latest model, YOLOv10, eliminates the need for NMS, which greatly enhances inference speed. This is achieved by using dual-label assignment during training. They have two branches of which one branch utilizes one-to-many assignments, and at the same time, the other branch uses one-to-one label assignments. Both branches are optimised simultaneously by using a label-matching metric. This metric ensures that the one-to-one branch learns from the many-to-one head. This way the one-to-one head effectively learns to predict only the optimal bounding box as opposed to many bounding boxes for the many-to-many head. During inference only the one-to-one head is used, thereby eliminating the need for removing redundant bounding boxes. This greatly enhances inference speed without compromising performance [68].

After a thorough investigation, we conclude that using the YOLOv10 model is a reasonable choice for our system in terms of accuracy, speed, and accessibility. As we have determined what the first component of the first phase of our system will be, we can move on to the second component; behaviour classification.

2.3 Behaviour Classification

2.3.1 Animal Feeding Behaviour Analysis

Analysis of feeding behaviour is more extensively reported in comparison to drinking behaviour. In a study conducted by Yang, Xiao, and Lin [31], they incorporated both the detection of pigs as well as directly inferring their behaviour by using a Faster R-CNN (Ren et al. [55]).

However, as we will be using separate models for both detection and behaviour classification, we will focus solely on behaviour classification models.

Moreover, separating these two tasks ensures each model is specialized in its specific task. This way we can use the flexibility and integrability of the YOLO detection model while exploring which behaviour classification method is optimal.

In addition, the detection model can be used in other behaviour analysis studies, which could further improve its scalability and generalizability in the future.

Finally, as we use the detection model as the first step, it might act as a preprocessing step thereby decreasing the number of possible detections that need to be analysed on behaviour. Having an initial filtering of detections might thus decrease the load on the behaviour classification model thereby enhancing inference speed.

However, as we will not explore an integrated system, discovering if there is an actual decrease in the workload remains an interesting question that will be left to future research.

Besides this study, there have been various other ways to analyse feeding behaviour. Two of these exploit the idea of using computer vision to estimate the difference in food quantities before and after feeding.

For instance, Shelley et al. [32] created a 3D image analysis algorithm to measure the amount cows eat during feeding by using the difference in food quantities before and after feeding.

In the same context, food mass change has been estimated by estimating volume change by Bloch et al. [33]. Their model could be used to monitor Dry Matter Intake (DMI).

However, as we will be analysing drinking behaviour, we will look more deeply into the research on drinking behaviour of animals using computer vision. There are various ways to accomplish the analysis of drinking behaviour of animals, each with its pros and cons. The most common approaches include (1) the use of classical CNNs, (2) Recurrent Neural Networks (RNNs), (3) processing videos directly with 3D convolutions, or (4) estimating poses of animals. We start this examination with the first option; Classical CNNs.

2.3.2 Convolutional Neural Networks

The simplest method to classify behaviour in videos is by extracting still images from the videos and classifying the behaviour seen within these snapshots. This image-based analysis is straightforward and computationally less demanding than video processing. These simpler models use normal CNNs [37, 71, 72]. However, this method ignores the temporal dynamics inherent in video data, potentially overlooking behavioural patterns that unfold over time.

An example of this is a study conducted by, Zhuang et al. [37] in which they designed systems to monitor pigs' feeding and drinking behaviours, using CNN algorithms like VGG19, Xception, and MobileNetV2 for behaviour recognition. In their study, they used still images to assess the feeding and drinking duration. As they used still images, they could not exploit the temporal features of the videos. However, it must be noted that they used a camera that was directly in front of the face of the pig when feeding, thereby greatly reducing the complexity of the dataset.

The model developed using MobileNetV2 demonstrated superior performance, achieving a recall rate of over 97% in pig drinking behaviour recognition. Additionally, their model exhibited low errors in estimating the duration of feeding and drinking behaviours, with a Root Mean Square Error (RMSE) of 0.58 seconds, a Mean Absolute Error (MAE) of 0.21 seconds for feeding durations, and an RMSE of 0.60 seconds with an MAE of 0.12 seconds for drinking durations.

In a study by Bello et al. [60], they present a study on using deep learning techniques to recognize the behaviors of group-ranched cattle from video data. The authors acquired video sequences of six cows (Keteku and Muturu breeds) in a ranch in September 2020. From this data, they selected 1000 keyframes and labelled them using the LabelMe tool. 800 frames were used for training and 200 for testing. To expand the dataset, data augmentation was applied to generate 4000 training frames and 1000 testing frames in total.

Four pre-trained object detection models were evaluated as potential cattle detection models: Mask R-CNN, Faster R-CNN, YOLOv3 and YOLOv4. Mask R-CNN, an extension of Faster R-CNN with an added mask generator, was found to achieve the highest detection accuracy and speed of 20 fps. It was therefore selected as the preferred model for this task. The outputs generated by Mask R-CNN included the bounding box, object class, confidence score and mask. The other models produced similar outputs but without the mask information.

Using the selected Mask R-CNN model, Bello et al. achieved average recognition accuracies of 93.34%, 88.03%, 93.51% and 93.38% for eating, drinking, active and inactive cattle behaviours respectively. These results demonstrate that their deep learning based approach is competitive with other state-of-the-art methods for automated cattle behaviour recognition. However, the authors noted some difficulties that led to the misidentification of behaviours in certain scenarios, such as invalid frames, cattle overlapping and instability in the cattle feeding setup.

2.3.3 Neural Networks with Temporal Components

A more sophisticated approach would be using short video sequences instead of still images. In a study conducted by Chen et al. [73], they used a combination of a CNN with a Long Short-Term Memory (LSTM) [74] model. LSTMs are a type of recurrent neural network (RNN) capable of learning temporal dependencies, making them suitable for understanding sequential data like videos. By utilizing an LSTM, this method incorporates temporal information, allowing accurate analysis of the behaviour of the animal as it incorporates information from previous frames.

To analyse the drinking behaviour of pigs, they used a CNN based on ResNet50 (He et al. [75]) as well as an LSTM to account for the temporal aspect of videos. Short videos of 2 seconds were employed in which the animals were seen from above and they marked the pigs with numbers. By doing this their model was able to discern between drinking and drinker-playing behaviour. They randomly split their 8000+ 2-second episodes into train and test. They evaluated two regions of interest (ROI), the head and the body.

To evaluate their model they trained it by using cross-entropy loss and evaluated the results with accuracy, sensitivity, specificity and precision. The cross-entropy loss is given by:

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{N} y_i \log(p_i) \tag{2.1}$$

Where y_i is the actual label, p_i is the predicted probability, and N is the total number of instances. Section 3.2 will thoroughly explain accuracy, precision, sensitivity, and specificity.

They achieved an accuracy, sensitivity, specificity, and precision for the head region of (92.5%, 91.2%, 93.8%, and 93.6%) respectively. For the body region, they obtained (87.2%, 84.9%, 89.5% and 89.0%). They concluded that the reason for the head region obtaining better results could be attributed to the fact that head touching and overlapping occur less frequently in the head region compared to touching and overlapping in the body region.

The primary challenge in their research lies in the increased complexity of the model which requires more computational power and data to train effectively. Moreover, training LSTMs can be time-consuming and may require larger amounts of annotated data, which in turn requires more human effort to obtain.

An even more advanced technique is used by Zhang et al. [76], and Fuentes et al. [59]. This involves using videos of varying lengths with 3D convolutions paired with temporal segment networks. 3D convolutions extend the capability of traditional 2D convolutions by adding the time dimension, enabling the model to learn spatial-temporal features directly from video data [77]. This approach excels at capturing complex behaviours that occur over time.

Temporal Segment Networks [78] further refine this by segmenting videos and allowing the network to focus on informative snippets of behaviour. While this method provides the optimal understanding of animal behaviours in videos, it is also the most complex. The complexity requires not just a high computational power, but the model design and training process is very demanding and labour-intensive.

The study by Fuentes et al. [59] introduces a deep learning approach for hierarchical cattle behaviour recognition using spatio-temporal information from RGB video data. Their framework involves appearance features at the frame level and spatio-temporal information that incorporates more context-temporal features.

They manually collected a new cattle behaviour dataset consisting of 350 videos (average duration 12 min each) recorded at different indoor farms in South Korea using an on-site camera system. Videos were captured during both day and night conditions to study behavioural changes. The dataset includes 15 different hierarchical cattle behaviours divided into individual activities (e.g. walking, eating, resting, but not drinking), group activities (e.g. fighting, social licking), and part actions (e.g. moving tail/head). Keyframes were extracted at 1 FPS and manually annotated with bounding boxes and action labels. The dataset contains 2714 annotated keyframes in total.

Their system operates in three parts. First, they employ an ROI frame-level detector based on YOLOv3. Then they use feature extraction with temporal context using 3D CNNs, and then they use spatio-temporal recognition fusing RGB images and optical flow. The frame-level YOLOv3 detector generates ROIs containing specific actions. Temporal context features are extracted using 3D convolutions over 5 consecutive frames. Optical flow is computed between frames to capture motion information. The RGB and optical flow features are fused for final spatio-temporal behavior recognition.

The frame-level YOLOv3 detector achieved 0.788 mAP, outperforming Faster R-CNN (0.711 mAP) while running much faster (30 vs 7 FPS). Adding spatio-temporal features improved performance for all behaviour categories, especially those with more motion. The full spatio-temporal model obtained 0.856 mAP overall. Looking specifically at the Feeding class, which is most similar to drinking, the mAP improved from 0.829 with just the frame-level detector to 0.885 when incorporating spatio-temporal features.

They noted challenges such as inter/intra-class variations in cow appearance and motion, occlusion between individuals and with environmental objects, and variable lighting between day and night videos.

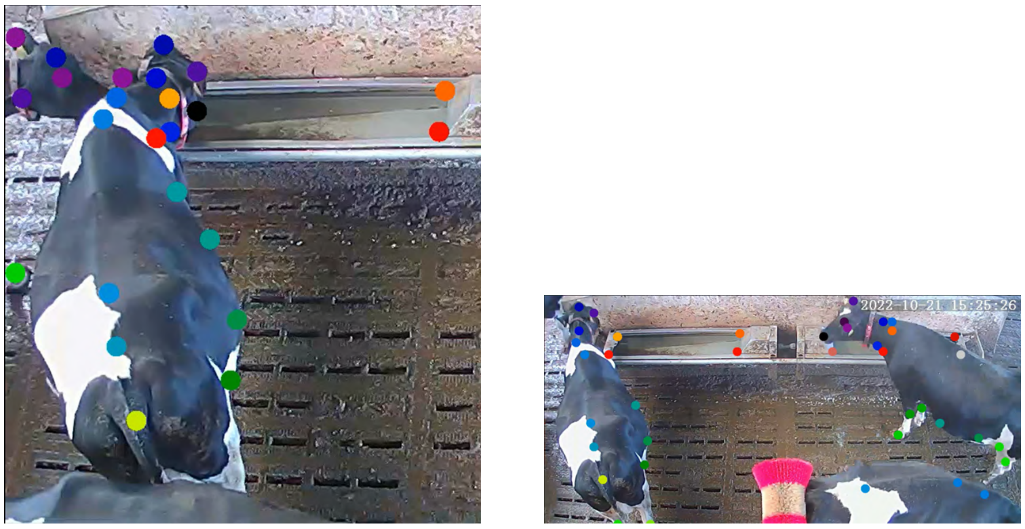
As our model should be able to do binary classification; drinking and non-drinking, using these rather complex architectures might not be necessary to obtain satisfactory results. Therefore, we have chosen to exclude these types of CNNs, thus the less advanced types of CNNs remain the best option thus far.

2.3.4 Pose Estimation

Another approach is to use pose estimation to detect key points of objects of interest. In a study by Islam et al. [34], they developed an algorithm using the deep learning architecture DeepLabCut [30] for tracking key body parts of beef cattle, such as head, ear, and neck positions, to distinguish drinking from non-drinking periods through long short-term memory (LSTM) analysis. For this, they used 70 videos for training and 8 videos for validating their model. Their approach achieved an accuracy of 97.35%.

Many others also showed promising results by using this open-source software for behaviour tracking in wild chimpanzees and bonobos [79], mice in a lab setting [80, 81], fish [82] and many quadrupeds like horses, dogs, tigers and more [83].

As this seemed to be a sound approach to analysing drinking behaviour, we investigated whether this technique could answer our questions. After an initial attempt with this software, in which we annotated around 150 frames by selecting the keypoints of cows, it has been shown that this approach is indeed able to get decent results in estimating the keypoints. To give an idea of these preliminary results, they are given in Figure 2.4.



(a) Snapshot cropped. **(b)** Snapshot uncropped.

Figure 2.4: DeepLabCut inference video results.

However, many more annotated frames were necessary to obtain satisfactory results. Annotating this data is very demanding as for each frame 30+ keypoints have to be accurately determined for 1000 frames. Accurately annotating 150 frames took approximately 14 hours. Thus it became apparent that annotation would take a tremendous amount of time.

Besides, as this approach is quite sensitive to camera views [83], the model would not generalize easily. The consequence is that each new camera view would need new annotated data.

After having a model that can determine the keypoints; head, eyes, mouth, legs, back, etc. we concluded that the process of extracting behavioural knowledge from these keypoints is a complicated study in itself. To be able to interpret the signal of the keypoints, another model needs to be trained on another set of annotated data. This approach seems to be more suitable for more complex behavioural analyses.

For this reason, we have pivoted and returned to our first and final approach, using the less advanced types of CNNs. The main reasons for this final decision are that (1) these networks can be more easily trained by making use of transfer learning, (2) we do not need vast amounts of annotated data, which can be time-consuming to obtain, nor (3) do we need to implement sophisticated and complex types of networks that have temporal components, and (4) finally this less complex approach has shown to give good results for similar use cases.

Now that we have determined which type of architecture we will employ, we will discuss an important training technique that will be used in this study.

2.3.5 Transfer Learning

As many different image classification models have been trained and are openly available, being able to use the knowledge they have gained from training greatly enhances model performance in new, but related tasks. This type of knowledge transfer is called transfer learning. The principle is based on the observation that CNNs learn generic features that can be applied in different tasks, just like how our brain works in the early layers [84].

The benefit of employing transfer learning is that it significantly reduces the amount of labelled data that is required for training [85] and has shown to give remarkable results even when using small datasets [86].

Moreover, it has shown that CNN features obtained from training on ImageNet [87] greatly enhances model accuracy [88]. Many models, like ResNet, Inception and DenseNet use pre-trained ImageNet weights that serve as the backbone for image classification tasks [89].

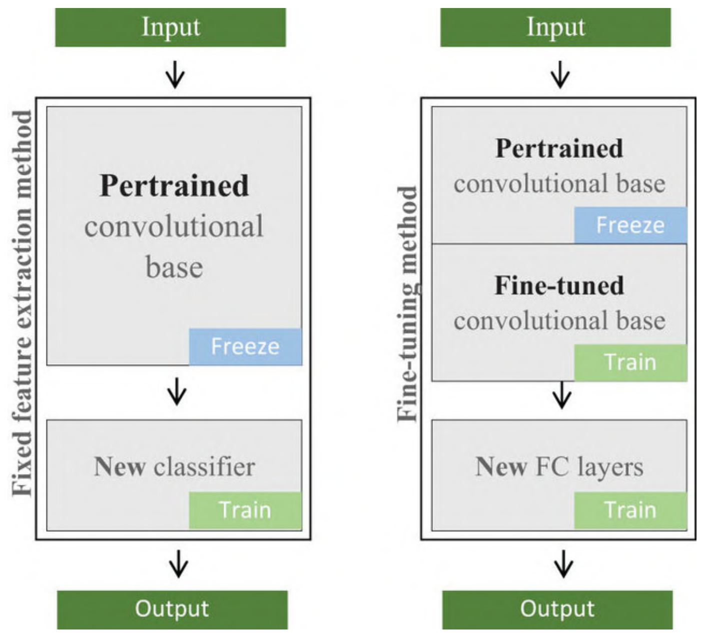


Figure 2.5: The process of transfer learning (adopted from Abbas et al. (2022) [90].

The process of transfer learning begins by loading the pre-trained model's architecture and weights. The model is then adapted to the new task by modifying or replacing the final output layer. The final output layer contains the classes the model should predict, thus for the new task these classes will most likely be different. An effective approach is to first freeze the backbone of the model, and train the model only on the final layers.

Subsequently, the model might undergo fine-tuning, which entails that the pre-trained backbone is partially or fully trained as well with a lower learning rate to preserve most of the feature extraction capabilities while tuning it to focus on specific features [89]. This process is depicted in Figure 2.5.

As we do not have unlimited amounts of annotated data, we will make use of this technique to train models that use the transferred knowledge from pre-trained models. This technique is used in both the detection of cows with YOLO as well as in behaviour classification and identification.

Now that we have determined which techniques are viable to answer our research questions, we shall proceed by describing the method in the subsequent chapter.

3. Method

In this chapter, we discuss the approach we take to answer the research questions. We start by describing and analysing the data in Section 3.1. We then continue by explaining how we will train and evaluate the detection and behaviour classification models in Section 3.2. Subsequently, we will discuss the software that has been used to develop and deploy the models in Section 3.3. We conclude this chapter by discussing the evaluation of the usability in section 3.4.

3.1 Data

To use our system we need various data sources and data types. We need images for the detection of the cows as well as images for training our Siamese Neural Network (SNN) for identification. We will describe each of these datasets used in this chapter.

3.1.1 Gathering and Size of the Datasets

We have gathered several different datasets and each serves its own purpose. We will explore the collection of the videos and images in this section.

Videos for Detection and Behaviour

We will start with the videos of the drinking cows. To obtain the videos, six cameras were installed on a dairy farm in the Netherlands at six different drinking troughs. 24 hours of video footage was collected on 5 consecutive days from October 17 to October 21, 2022. As the frames per second (fps) were not consistent for each of the videos due to unknown reasons, the videos have different durations.

Besides this, there is a different number of videos for each of the days. The camera numbers were provided as 1, 3, 4, 5, 6, and 7, therefore, we have decided to maintain this numbering. The final number of videos for each day is given in Table 3.1. The total number of videos is 776. Further characteristics, exploration, and processing are explained in Sections 3.1.2, 3.1.3, and 3.1.4 respectively.

Table 3.1: Number of videos per day for each of the cameras.

		camera					
		1	3	4	5	6	7
	17	24	25	24	25	24	24
	18	25	25	25	25	25	25
day	19	24	24	24	24	24	24
	20	28	28	28	28	28	28
	21	28	28	28	28	28	28

Images for Detection

From these selected videos, we extracted cows which were detected by an untrained YOLOv10-X model, the largest and most precise out-of-the-box detection model of the YOLO series. From these detections, we manually selected the correct detections and corrected any errors. The error correction is further discussed in Section 3.1.6.

Leveraging the detection capabilities of YOLO greatly reduced the manual collection effort. The final datasets for detection are given in Table 3.3 and Figure 3.4. This dataset is denoted by the abbreviation DD-10k. The dataset contains approximately 3000 frames and 10000 detections.

Video for Behaviour

From the videos, we extracted one full hour of video for final testing purposes. This video was selected randomly from a camera on which the model will not be trained. The video that was selected was captured from 09:00 to 10:00 AM on the 20th of October. This separation process will be further explained in Section 3.1.5. From the video we obtained 2 frames per second, resulting in a total of 7360 frames.

Images for Behaviour

From the same videos, we extracted data for the behaviour classification model. This was done by selecting the detections found by YOLOv10-X in which drinking cows and non-drinking cows were found. Using the pre-trained YOLOv10-X we were able to greatly reduce the time needed for extracting the data without sacrificing accuracy or data variability. The final datasets are given in Table 3.5. We will denote this dataset by BD-5k. This dataset contains approximately 3000 images of drinking cows and 2000 of non-drinking cows.

3.1.2 Characteristics of the Data

In this section, we will describe the features of the videos and images. For the videos, we will discuss the duration, resolution, format, and any variations in the content or other anomalies. The resolution, and format of the images and videos will be discussed. Examples of both the videos and images will be provided in Section 3.1.3.

Videos for Detection and Behaviour

As described in Section 3.1, the videos were collected over multiple days, with varying numbers of videos per day. The resolution of the videos was 1280×720 when recorded in landscape mode and 704×578 in portrait mode. This discrepancy in modes is likely due to the camera's automatic calibration during filming. It was observed that within an hour of recording, the camera adjusted its mode automatically. This change in resolution could potentially impact the system's precision, so caution was taken when analyzing the results from these videos.

In addition to the change in filming mode, there was considerable variation in the duration and frames per second (fps) of each video. We analyzed the duration and fps of each video and plotted these values for one camera on one day in Figure 3.1a. Each dot in the plot represents a video, labeled with the hour it was captured.

A clear diagonal line can be observed. This indicates that the frame count and duration increase according to the exact same slope. They exhibited a similar slope as this is corresponds to the fps of the video. The reason for a different number of frames for videos with the same fps seems to be attributed to slight jumps in each video, where parts of the video were either skipped or sped up. This pattern was consistent for each day and each camera, although the exact cause of these speed variations remains unclear.

Since individual plots did not provide additional insights, we synthesized them into a single plot, shown in Figure 3.1b. The pattern here is more pronounced, revealing six diagonals corresponding to six different fps values: 2, 3, 4, 5, 6, and 7. The reason for the different frame rates is unknown, but we accounted for the fps variation when processing the videos.

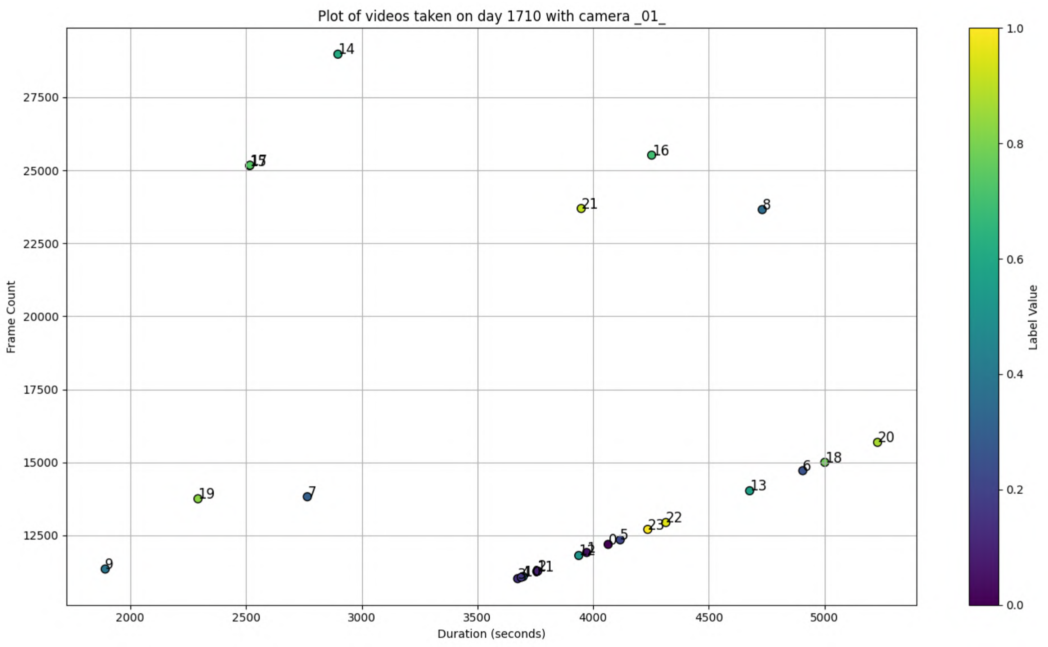
Although additional patterns are visible in Figure 3.1b, we have gathered sufficient information to fully process the videos, and further discussion is unnecessary. We will continue exploring these videos in Section 3.1.3.

Images for Detection

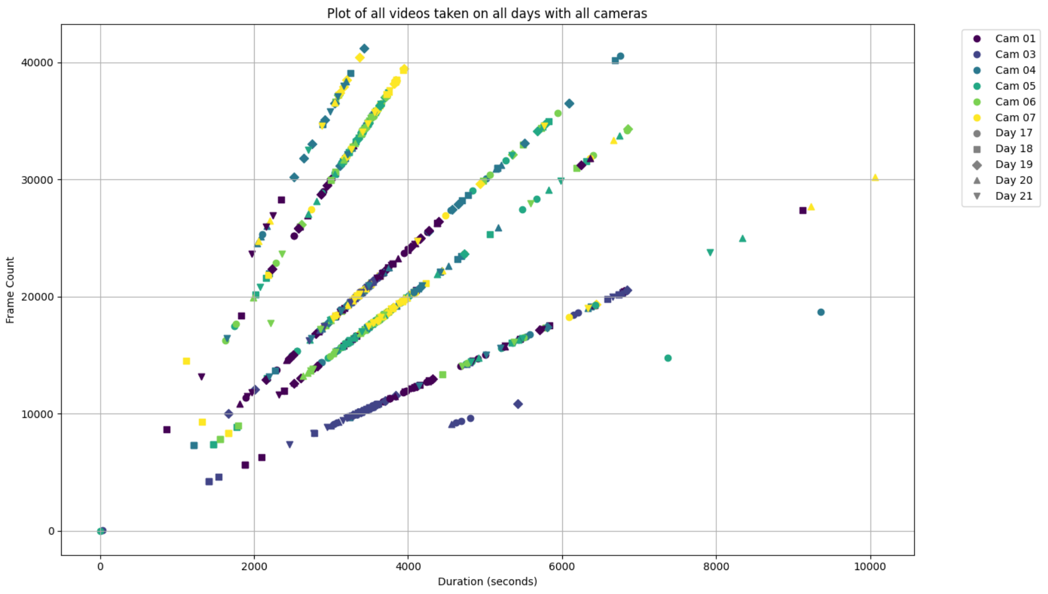
The frames used to train the YOLO model were extracted directly from the videos captured at the drinking troughs, thus they correspond to the quality of the video captured. As described in Section 3.1.2, the resolution of each of the videos was 1280×720 with a dpi of 96, thus the frames have exactly the same characteristics. As 24 hours of video was captured, there is a lot of variability in lighting, colour, and blur.

Images for Behaviour

The images used to train the behaviour classification model vary in size as each bounding box has a different size. This difference may vary substantially between very small detections and larger ones. The quality of the images is also the same as that of the camera, as the detections were cropped from the frames captured by the same camera.



(a) Videos of camera 1 captured on 17/10



(b) Videos of all cameras on all days

Figure 3.1: Overview of the duration and frame counts of videos captured by various cameras on all days.

3.1.3 Data Exploration

In this section, we will give samples of each of the datasets, anomalies, and other details to get a grasp of the level of detail, lighting, shading, colour, and environments. For the detection datasets, DD-10k and BD-5k, we provide examples of various moments during the day, camera viewpoints, and cow behaviours. Subsequently, we will provide examples for the identification datasets.

Images for Detection

All these examples can be found in Figure 3.3. The first feature that stands out in this figure is the difference in color due to the differences in daylight. In Figure 3.2b, 3.2h, 3.2k, and 3.2m we observe almost no colour, whereas in Figure 3.2e, and 3.2f, we observe a colour distortion. These different colour conditions severely impact the system's performance if improperly handled.

Another important feature is the lighting. In Figure 3.2j, 3.2n, and 3.2o, we can see the impact of light on the environment. In the evening, the intense white light might cause overexposure, whereas in the morning the sun and shade difference might inhibit the system's identification precision.

Another important and common problem in computer vision is occlusion. This can clearly be observed in Figure 3.2b and 3.2k. In these images, occlusion due to a white line, and condensation drops, cause unclear images and possibly unidentifiable cows.

As discussed in Section 2.3.3, various viewpoints were used as the ROI to classify animal behaviour. For cows, these included tailhead images, nose images, and side-view images. In our environment, it is impractical to classify cow drinking behaviour by its face or nose when it is drinking as cameras would be needed for each drinking location at each drinking trough resulting in many cameras. Moreover, it is arguably more pragmatic to use fewer cameras when extending this software to other environments compared to placing several cameras at each drinking position at each drinking trough.

Images for Behaviour

The behaviour classification model faces a challenge due to significant variation in image sizes, resulting in some low-quality images, as illustrated in Figure 3.3d. These images depict a wide range of detections from the detection model, including cows in the background. It is crucial to include

(a) Camera 1 day

(b) Camera 1 night

(c) Camera 3 portrait

(d) Camera 3 day

(e) Camera 3 night orange

(f) Camera 5 night red

(g) Camera 4 day

(h) Camera 4 night

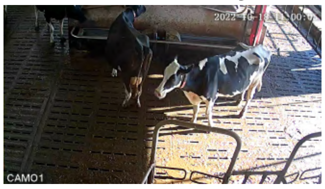
(i) Camera 5 day

(j) Camera 5 evening

(k) Camera 5 night

(l) Camera 6 day

(m) Camera 6 night


(n) Camera 7 morning

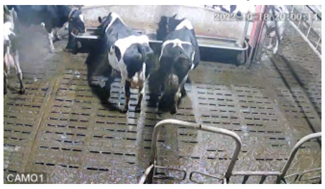
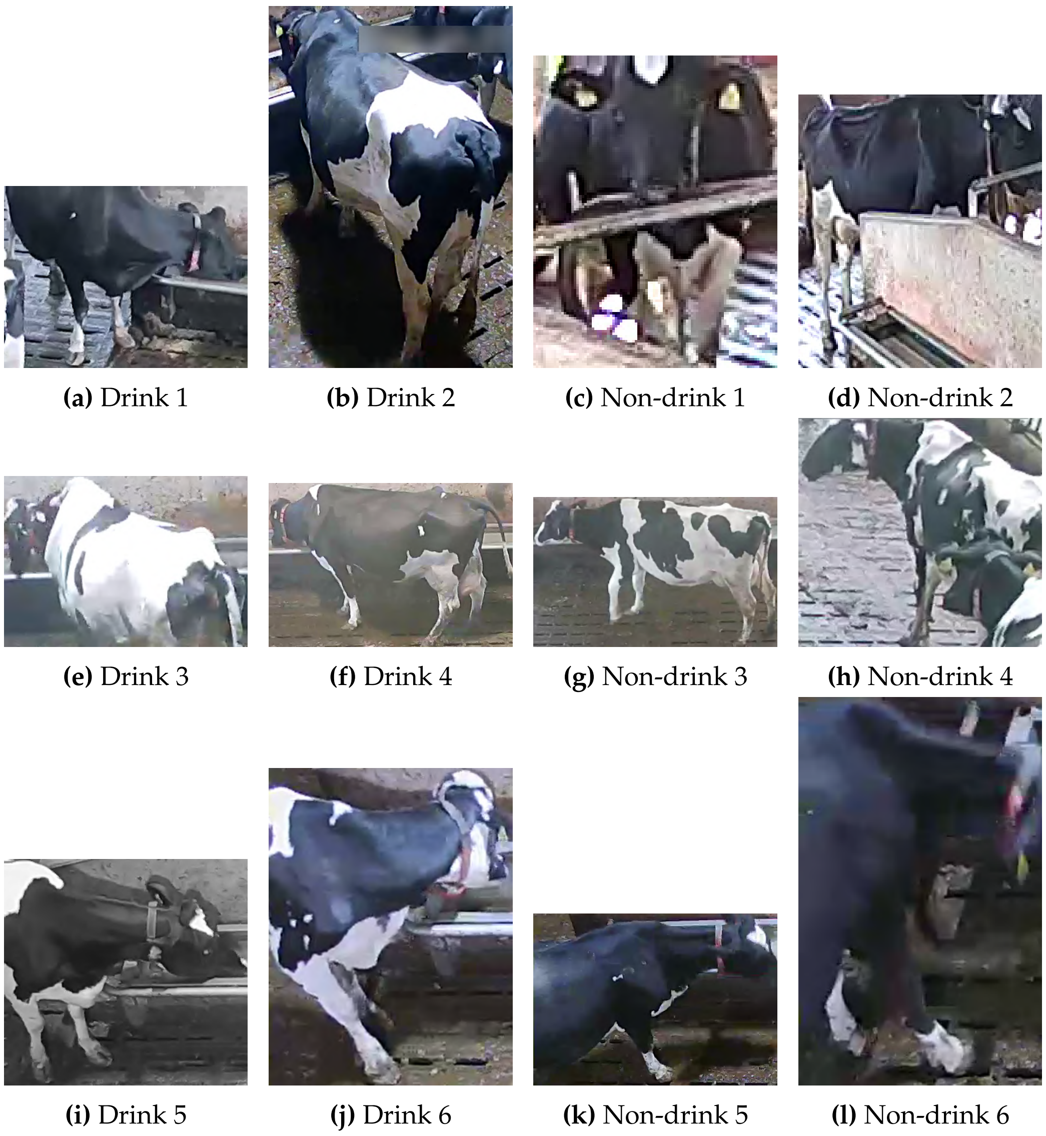

(o) Camera 7 evening

Figure 3.2: Extracted images from videos captured by the 6 different water trough cameras.

all detections, even those of lower quality, for the model's ability to generalize effectively and accurately classify drinking behaviour across various distances and resolutions. Excluding these might overlook the diversity of real-world environments.

3.1.4 Data Cleaning

In this section, we will describe the process of dealing with incorrect images. We explain how these are recognized and removed from the dataset. We also address any other problems we have encountered.



(a) Drink 1 **(b)** Drink 2 **(c)** Non-drink 1 **(d)** Non-drink 2

(e) Drink 3 **(f)** Drink 4 **(g)** Non-drink 3 **(h)** Non-drink 4

(i) Drink 5 **(j)** Drink 6 **(k)** Non-drink 5 **(l)** Non-drink 6

Figure 3.3: Extracted images from detections included in the BD-5k dataset.

Videos for Detection and Behaviour

After further analysis, we found that the difference in the number of videos could be attributed to several reasons. First of all, we found a video with a duration of 0 seconds, which was erased from the dataset. This video was captured by camera 5 on the 17th of October at 00:00. On this same day, a video of a duration of 30 seconds was captured by camera 3 at 00:00. This video remained in our dataset.

The second reason was that we found several duplicates in the last two days. Each camera contains one video of each of the other days captured by that camera. This results in 4 duplicates per camera. For the 20th of October,

Table 3.2: Corrected number of videos per day for each of the cameras.

		camera					
		1	3	4	5	6	7
day	17	24	25	24	25	24	24
	18	25	25	25	25	25	25
	19	24	24	24	24	24	24
	20	24	24	24	24	24	24
	21	24	24	24	24	24	24

these were videos captured at 00:00, whereas for the 21st of October, these were videos captured at 11:00. These videos were removed from the dataset as well.

The last reason was that on the 18th of October at 01:00 the videos captured lasted for $\approx$ 28 minutes. Therefore, this hour contains 2 videos, hence the extra videos for this day for each camera. These videos remained in the dataset. The final size of the dataset is given in Table 3.2.

Images for Detection and Behaviour

As we have manually selected the frames that were used to train the YOLO model, we have automatically cleaned the images. The same applies to the images that were used by the behaviour classification model.

3.1.5 Data Preprocessing

This section deals with the preprocessing steps to prepare the data for being processed by the models. We discuss the splitting of the train, validation, and test sets, the resizing and normalizing of the images, and possible augmentation steps.

Images for Detection

As we needed to tune the parameters of the detection model and do final evaluations, we needed two separate datasets. As we have 6 different cameras, separating two of them to use for tuning (validation) and evaluation (test) was a straightforward and practical solution.

For the train set, we picked videos from 4 cameras for 3 different times of the day. The exact hours were picked randomly, per camera. This process ensured that there was a lot of variability in the data as we had different times of the day. This results in videos with varying lighting conditions, which improved the model's robustness and real-world applicability.

The cameras that were picked for the training dataset were 1, 3, 6, and 7. For validation and testing, we picked cameras 5 and 4 respectively. The choice of cameras was based on the level of variability. The cameras in the train set had varying viewpoints, therefore the model will be trained on easy and complex environments with cows in the foreground as well as the background. As we optimized the model on the validation set, we chose a camera which had cows in the background as well. Lastly, for the test set, we chose a camera without any background detections. This gave a clear picture of how accurate the model was in an ideal situation.

We attained a splitting ratio of approximately 70%, 20%, and 10% for the train, validation, and test sets respectively. The resulting sizes of the datasets are given in Figure 3.3.

Table 3.3: Size of each of the detection datasets (DD-10k).

	Train	Validation	Test
Camera(s)	1; 3; 6; 7	5	4
Frames	2283	562	290
Detections	7007	1744	365

Besides the actual sizes, it is also interesting to examine the number of detections in each of the datasets. These ratios are given in Figure 3.4. In this figure, we can observe that the train set contains up to 9 detections per frame, whereas the validation set and test set contain up to 7 detections and 2 detections per frame respectively. The difference in number of detections was caused by the positioning of the camera. The train set contains cameras that have a lot of different viewpoints and contain cows in the background as well as the foreground. Consequently, the number of detected cows in the train set is larger compared to the test set which contains no background detections.

Besides splitting the data, we needed further preprocessing. The first step was to blur the time and date on the frames. Then we padded the

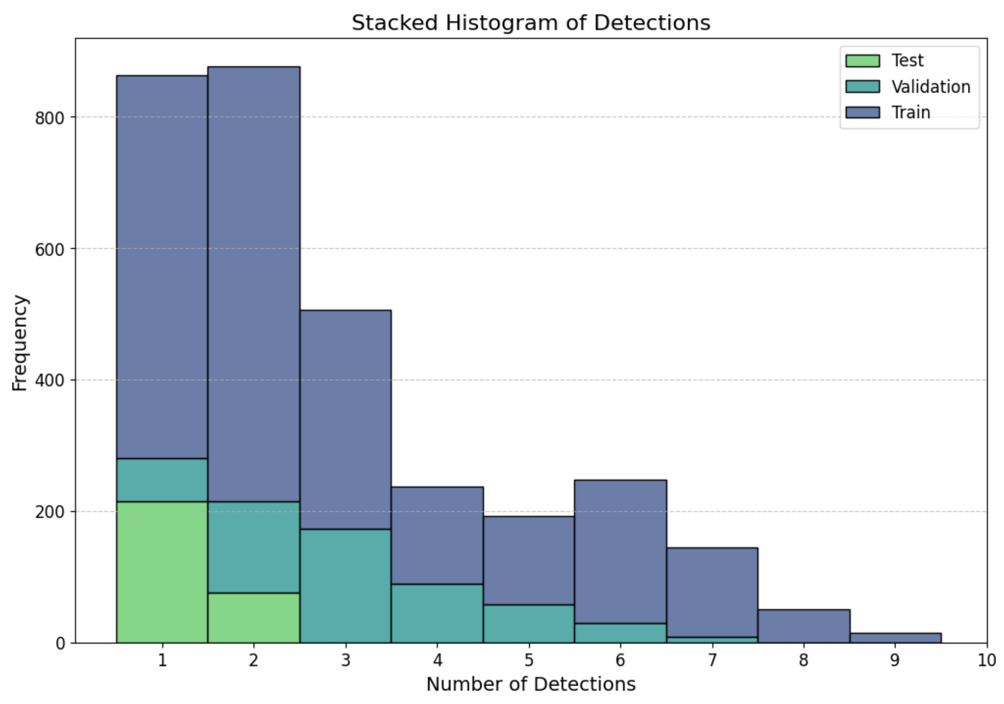


Figure 3.4: Histogram of detection dataset (DD-10k).

images to make them square and resized them afterwards to the expected size on which YOLO was pre-trained. Thus, we resized all images to 640 $\times$ 640.

After resizing we applied normalization by the mean and standard deviation. For these we used very commonly used values; the mean we took to be: [0.485, 0.456, 0.406] and a standard deviation of: [0.229, 0.224, 0.225] [38].

We used data augmentation to ensure robustness and generalization capabilities. In our detection model, we employed several augmentation techniques to enhance the diversity of our training data.

HSV Color Space Augmentation: This technique involves adjusting the hue, saturation, and colour value of images. It helped the model become invariant to colour changes and lighting conditions, which is particularly useful for varying lighting scenes, which was the case for our dataset as we were dealing with day and nighttime scenes [91].

Geometric Transformations: We applied several geometrical transformations to our images. These included rotations, translations, and scaling. This helped the model to learn to detect objects at various angles, positions, and sizes within the frame [92].

Horizontal Flipping: This simple technique creates a mirror image of the original. This ensured the model was invariant to the orientation of the detections [91].

Mosaic Augmentation: This is an advanced technique that combines four training images into one. This allowed the model to learn to detect smaller objects and understand objects in relation to various contexts and backgrounds. It creates one larger image with 4 tiles so to speak, hence the name mosaic augmentation [93].

Random Erasing: Also known as cutout, this method randomly removes rectangular regions from the image, which simulates occlusions [94].

Cropping: Cropping involves extracting a random portion of the original image for training. If it is applied, it crops that amount of the original image and resizes the crop to the original size of the image. This type of augmentation improved the model's ability to detect partially visible objects [91].

Grayscale Conversion: Converting colour images to black and white helped the model focus on shape and texture rather than colour [91].

These augmentation techniques, when applied in various combinations, significantly expanded our dataset variance and helped to create a more robust and generalizable object detection model.

Video for Behaviour

For the behaviour analysis video, we did not need any splitting. The final size and class distribution of the video after annotation is given in Table 3.4.

Table 3.4: Size and distribution of the full one-hour behaviour classification video obtained from camera 4.

	Total Frames	Non Empty Frames	Drinking	Non-Drinking
After Annotation	7360	3735	1320	3720

Images for Behaviour

For the behaviour dataset, we attained the same logic. We split the data into train, validation, and test sets, where 1, 3, 6, and 7 were for training, 5 for validation, and 4 for testing purposes. The resulting sizes of the datasets are given in Table 3.5.

Table 3.5: Size of each of the behaviour classification datasets (BD-5k).

	Train	Validation	Test
Camera(s)	1; 3; 6; 7	5	4
Drink	2972	189	181
Non-Drink	1952	191	179

For the preprocessing of the images, we padded and resized them to 384 × 384 for EfficientNetV2Large and to 256 × 256 for all other models. The actual models we used will be discussed in the subsequent chapter.

Thereafter, we applied normalization by a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225]; the same normalization as for the detection model.

The types of augmentation employed here were approximately the same. The actual augmentations that were applied will be discussed in the Experiments chapter; Chapter 4.

3.1.6 Data Annotation

Having data that is annotated correctly is crucial for having models that can accurately perform their tasks. As each of the problems requires different solutions, the annotation of each of the datasets is different as well. In this section, we will discuss how each of the datasets is annotated.

Images for Detection

To create a high-quality dataset for training our YOLO detection model, we annotated our dataset annotation using the Roboflow annotation web interface, which has a partnership with Ultralytics, the developers of YOLO.

We began by using the untrained YOLOv10-X model to extract initial detections from our dataset. This provided a first set of bounding boxes around the cows, which served as a starting point for further refinement.

The extracted detections were then converted to the COCO (Common Objects in Context) format [95]. The COCO format is a widely-used standard for object detection datasets, which includes annotations for object instances, segmentation masks, and keypoints [95]. This format enables integrations with various annotation tools and machine learning frameworks, which substantially improves the usability of the dataset.

The COCO-formatted detections were uploaded to our workspace in Roboflow. Roboflow provides a user-friendly interface for visualizing and editing annotations, making it easier to manage large datasets. Using the Roboflow interface, we manually reviewed and corrected the initial detections. This step was crucial for ensuring the accuracy of our annotations. Incorrect detections were adjusted, and any missed objects were added.

After correcting the detections, the annotated dataset was finalized and exported from Roboflow. This dataset could then be used for training and evaluating our YOLO detection model.

Video for Behaviour

The video for final testing purposes was annotated by leveraging the detection and recognition capabilities of the best models obtained during this research. We employed the models that attained the best results during testing for both the detection and behaviour recognition phases. The results were then carefully corrected using Roboflow. Subsequently, the annotations were exported in COCO format to be compared with the predictions.

Images for Behaviour

For the behaviour classification task, we employed a straightforward and efficient manual annotation process. We started with the detections obtained by the untrained YOLOv10-X model, which provided individual images of each of the cows. At this stage, we did not have a trained model as opposed to the stage we were in during annotation of the full one-hour video.

Then, using the Windows File Explorer, we visually inspected each detection image. From visual assessment, we classified the detections either as drinking or non-drinking. We then copied and pasted the corrected and classified images to their dedicated class folder.

This way we obtained an initial dataset that could be used to train an initial binary classifier. For this, we used EfficientNetV2 Large. Then we iteratively engaged in the described process to obtain more images of drinking and non-drinking cows. We measured the confidence of the classes and we selected both the high-confidence false positives as well as the low-confidence positives of the drinking class after each iteration. These were subsequently added to the dataset and the model was retrained with these added images that added crucial information. This ensured the model was trained iteratively on images it was certain of, but classified incorrectly, as well as images of which it was uncertain. This process greatly enhanced the capabilities of the model after each iteration.

Using the file explorer and doing this iterative process enabled us to obtain a model that was trained on a large variety of images containing drinking and non-drinking cows.

3.2 Metrics

3.2.1 Detection

To evaluate the detection model we will use various commonly used metrics. In object detection, the most commonly used metric for evaluating detection accuracy is average precision [47]. It computes the area under the precision-recall curve. Precision measures relevancy, while recall measures the number of truly relevant results returned [96].

A system with high recall but low precision returns many results, most of which are incorrect. Conversely, a system with high precision but low recall returns few results, but most are correct. An ideal system achieves both high precision and high recall, returning many correctly labelled results.

Precision (P) is defined as the number of true positives (TP) divided by the sum of true positives and false positives (FP):

$$P = \frac{TP}{TP + FP} \tag{3.1}$$

Recall (R) is defined as the number of true positives divided by the sum of true positives and false negatives (FN):

$$R = \frac{TP}{TP + FN} \tag{3.2}$$

The F_1 score, which is the harmonic mean of precision and recall, is calculated by:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.3}$$

Average precision (AP) summarizes the precision-recall curve as the weighted mean of precisions at each threshold, with the increase in recall from the previous threshold as the weight:

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{3.4}$$

AP and the area under the precision-recall curve (AUC) summarize a precision-recall curve.

Two other commonly used metrics to evaluate the results are given by sensitivity and specificity. Sensitivity (Se) is the same as recall, and specificity (Sp) is defined as the number of true negatives (TN) divided by the sum of false positives (FP) and true negatives:

$$Sp = \frac{TN}{FP + TN} \times 100\% \tag{3.5}$$

Both these metrics are summarized in one graph called the ROC AUC (Receiver Operating Characteristic Area Under the Curve). It is a metric used to evaluate the performance of binary classification models. It measures the ability of a model to distinguish between positive and negative classes by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. A higher ROC AUC indicates better model performance [97].

To assess the accuracy of the detection model we used precision, recall, and average precision. More specifically, we used two types for average precision: AP50B and AP50-95B. The AP50B represents the Average Precision at a 50% Intersection over Union (IoU) threshold for bounding boxes, which will be discussed in the subsequent paragraphs.

This provides a measure of the model's accuracy in object detection and localization at a moderate strictness level. The more elaborate AP50-95B offers a more comprehensive evaluation by averaging the AP values across multiple IoU thresholds ranging from 50% to 95% with increments of 5%.

This metric provides insight into the model's performance across various levels of precision, from relatively lenient to highly strict criteria. Both metrics were calculated using the area under the precision-recall curve [98].

To measure the model's ability to draw accurate bounding boxes during training we measured the loss, which is a weighted sum of the Box Loss (BL) and the Distribution Focal Loss (DFL).

$$L_{total} = \lambda_{box} L_{box} + \lambda_{dfl} L_{dfl} \tag{3.6}$$

The Box Loss is calculated as:

$$L_{box} = \frac{\sum_{i \in B_{gt}} w_i (1 - \text{CIoU}_i)}{\sum_i s_i} \tag{3.7}$$

Where:

- B_{gt}: Set of ground truth bounding boxes
- $w_i = \sum_j s_{ij}$: Weight for each sample i, calculated as the sum of target scores for that sample across all classes
- $\text{CIoU}_i = \text{CIoU}(b_i, b_{gt,i})$: Complete IoU between the predicted bounding box b_i and the target (ground truth) bounding box $b_{gt,i}$ for sample i
- s_{ij}: Target score for sample i and class j (1 for the object class and 0 for

all other classes)

In this equation we calculated the Complete IoU (CIoU) [99]. This is an extension of the standard IoU (Intersection over Union), which is a very common measure of bounding box accuracy [100]. The standard IoU is calculated as:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{3.8}$$

A depiction of this calculation is given in 3.5.

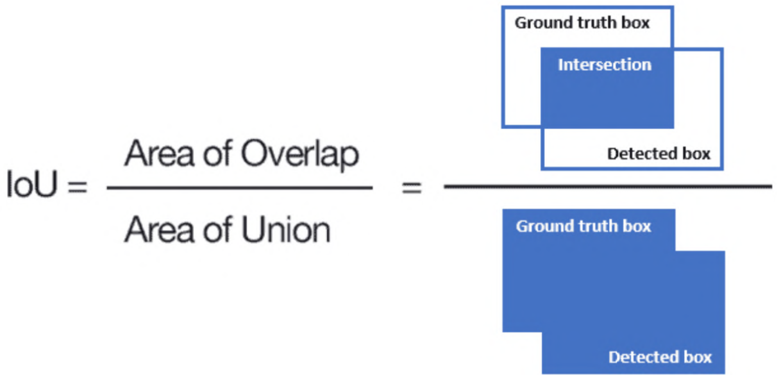


Figure 3.5: Depiction of the IoU calculation (adopted from [101]).

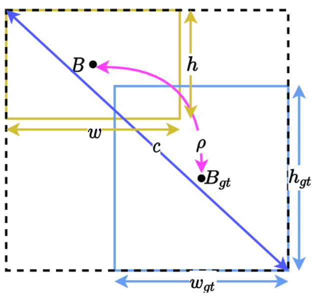
CIoU extends IoU by considering two additional factors besides the overlap area: central point distance between boxes, and aspect ratio consistency. The CIoU loss is defined as:

$$L_{CIoU} = 1 - \text{IoU} + \frac{\rho^2(b, b_{gt})}{c^2} + \alpha v \tag{3.9}$$

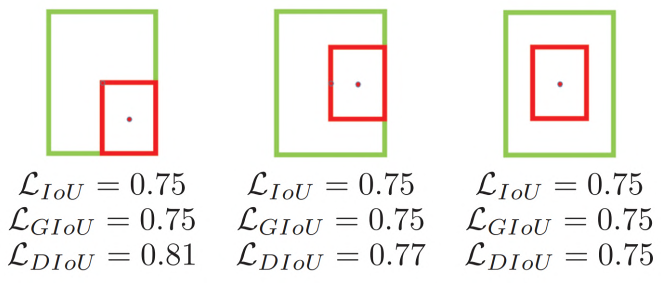
Where:

- $\rho^2(b, b_{gt})$: Squared Euclidean distance between the central points of the predicted and target boxes
- c^2: Diagonal length of the smallest enclosing box covering the two boxes
- α: Positive trade-off parameter
- v: Measure of aspect ratio consistency

For completeness we have given a visual interpretation of this calculation in Figure 3.6a. In this figure B is the predicted bounding box and B_{gt} is the ground truth bounding box. The benefit of using this extended IoU is that it takes into account what the position of the predicted bounding box is relative to the ground truth. The version is a combination of Generalized IoU (GIoU) and Distance IoU (DIoU) [102], hence the name Complete IoU (CIoU).



(a) Depiction of the CIoU calculation (adopted from [103]).


$\mathcal{L}_{IoU} = 0.75$
$\mathcal{L}_{GIoU} = 0.75$
$\mathcal{L}_{DIoU} = 0.81$
$\mathcal{L}_{IoU} = 0.75$
$\mathcal{L}_{GIoU} = 0.75$
$\mathcal{L}_{DIoU} = 0.77$
$\mathcal{L}_{IoU} = 0.75$
$\mathcal{L}_{GIoU} = 0.75$
$\mathcal{L}_{DIoU} = 0.75$

(b) Comparison of IoU with DIoU and GIoU (adopted from [102])

Figure 3.6: Complete Intersection over Union (CIoU).

The Distribution Focal Loss (DFL) is calculated as:

$$L_{dfl} = -\sum_{i \in B_{gt}} w_i \sum_{j=0}^{n-1} y_{ij} \log(p_{ij}) \tag{3.10}$$

Where:

- B_{gt}: Set of ground truth bounding boxes.
- w_i: Weight for each sample (corrects class imbalance), calculated as $w_i = \sum_j s_{ij}$
- n: Number of bins used to discretize the bounding box coordinates.
- y_{ij}: Target distribution for the j-th bin of the i-th sample.
- p_{ij}: Predicted probability for the j-th bin of the i-th sample.

The Distribution Focal Loss (DFL) uses a binning approach to predict bounding box coordinates. Instead of predicting single point estimates, each coordinate (x, y, width, height) is divided into n discrete intervals or "bins" (typically $n = 16$). The target distribution y_{ij} is typically a one-hot or soft distribution centred around the ground truth coordinate value, where i represents the sample and j the bin index.

For instance, if the ground truth falls into the 8th bin, y_{i8} would be 1 (or close to 1) and other y_{ij} values would be 0 (or close to 0). The predicted distribution p_{ij} is obtained by applying a softmax function to the model's output for each coordinate, resulting in a probability distribution over the bins. The final coordinate prediction is not simply the bin with the highest probability, but rather a weighted sum across all bins:

$$\text{coord}_i = \sum_{j=0}^{n-1} c_j \cdot p_{ij} \tag{3.11}$$

Where c_j is the centre value of bin j.

3.2.2 Behaviour

To optimize our behaviour classification model, we will use cross-entropy. This loss is widely used in classification tasks to measure the dissimilarity between predicted and true probability distributions. For multi-class problems with C classes, the categorical cross-entropy is defined as:

$$L = -\sum_{c=1}^{C} y_c \log(p_c) \tag{3.12}$$

where y_c is the true probability of class c and p_c is the predicted probability [104]. However, in our binary classification problem of drinking vs. not drinking, we use binary cross-entropy. This simplifies the equation to:

$$L = -[y \log(p) + (1-y)\log(1-p)] \tag{3.13}$$

where y is the true label (0 or 1) and p is the predicted probability of the positive class. In addition to cross-entropy loss, we evaluate our model's performance using accuracy, precision, recall, F1-score, ROC AUC, and confusion matrices. The precision, recall, F1-score and ROC AUC calculations are b, based on the drinking class only. The confusion matrix and accuracy give an estimate of the predictions of the non-drinking class.

A confusion matrix is a table that summarizes the performance of the classification model by comparing the predicted labels with the actual labels. It provides a comprehensive view of how well the model is performing in terms of true positives true positives, true negatives (TN), false positives (FP), and false negatives (FN). The matrix allows for the calculation of various evaluation metrics such as accuracy, precision, recall, and F1 score [105].

To assess the full one hour of video obtained from the test camera, we compared the predictions with the corrected behaviour classifications. For this comparison we employed the same metrics as for the behaviour classification model; accuracy, precision, recall, f1 score, and a confusion matrix. In this case, we provided the metrics for both classes separately as well to give the full picture.

To calculate the error in drinking time we will employ two common error metrics: percentage error and absolute error. These metrics will allow us to assess the accuracy of our model's time estimations compared to the actual drinking times observed.

Percentage error is a measure that expresses the difference between the estimated value and the actual value as a percentage of the actual value. The formula for percentage error is as follows:

$$PE = \frac{(Estimated - Actual)}{Actual} \times 100 \tag{3.14}$$

Where PE is the percentage error, Estimated is the predicted drinking time, and Actual is the true drinking time [106].

Absolute error is expressed in the same units as the measurement, seconds in our case. The formula for absolute error is:

$$AE = |Estimated - Actual| \tag{3.15}$$

Where AE is the absolute error [106].

In our analysis, we will calculate these errors using the following approach: The estimated drinking time will be derived from our model's predictions, calculated as ($TP + FP$) / frames per second (fps). The actual drinking time will be calculated as TP / fps.

By employing both percentage error and absolute error metrics, we can gain a comprehensive understanding of our model's performance in estimating drinking time.

3.3 Software

The implementation of machine learning and deep learning projects involves a combination of various tools and platforms. Each of them is used for specific aspects of the development and deployment pipeline. From data storage and preprocessing to model training and deployment, the choice of technology significantly impacts the efficiency and scalability of the solutions developed.

The most important of these are (1) the distributed computing cloud platform (Databricks), (2) data storage (Azure Blob Storage), and (3) the deep learning framework (Tensorflow). Each of these will be discussed in this section.

3.3.1 Databricks

Databricks is a large data analytics platform, which is especially useful for its collaborative workspace that integrates data engineering, data science, model experimenting, and model deployment. It facilitates scalable and efficient processing of big data using Apache Spark allowing for distributed computing.

Databricks supports various programming languages, including Python, Scala, and R. As we will be using Python, this platform is an excellent choice for our needs. The platform contains collaborative notebooks, integrated with MLflow for experiment tracking and model management, thereby streamlining the evaluation process [107]. It is also easily integrated with big data storage like Azure Blob Storage, which we will be using to access and utilize our data.

Databricks provides native runtime confivations, which come with preinstalled packages optimized for machine learning purposes. We used the

Databricks Runtime 14.2 ML. This runtime offers specialized configurations for both GPU-accelerated training and CPU-based development. For GPU-intensive machine learning tasks, we chose a cluster configuration with the following specifications:

- 1 Driver node
- 112 GB Memory
- 6 cores
- Runtime: 14.2.x-gpu-ml-scala2.12
- Instance type: Standard_NC6s_v3

This GPU-enabled setup uses NVIDIA GPU libraries, including CUDA 11.8, cuDNN 8.9.0.131-1, NCCL 2.15.5, and TensorRT 8.5.3-1, to accelerate deep learning and other GPU-optimized functionalities.

For development and less computationally intensive tasks, we used a CPU-based cluster:

- 1 Driver node
- 32 GB Memory
- 8 cores
- Runtime: 14.2.x-gpu-ml-scala2.12
- Instance type: Standard_D8ads_v5

Both configurations utilize Databricks Runtime 14.2 ML, which provides an environment for machine learning. This runtime includes popular libraries such as TensorFlow, and PyTorch, as well as Databricks' AutoML tool for automated machine learning pipeline training and MLFlow for logging the results. For a complete overview of all packages, we refer to their documentation [108].

3.3.2 Azure Blob Storage

Azure Blob Storage, a service provided by Microsoft Azure, offers scalable and secure cloud storage for large amounts of unstructured data, such as text or binary data (images and videos). Azure Blob Storage supports the efficient management and retrieval of data at scale, making it an essential component for data-intensive applications.

Its integration with Azure Databricks allows for easy integrated processing and analysis of stored data. The service contains role-based access control, thereby ensuring that data is protected throughout its lifecycle, from uploading to extraction and manipulation [109].

3.3.3 TensorFlow

TensorFlow is an open-source framework developed by the Google Brain team [110], is widely recognized for its comprehensive, flexible ecosystem of tools, libraries, and community resources that enable researchers to advance the state-of-the-art in machine learning and developers to build and deploy ML-powered applications.

The architecture of TensorFlow allows for easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), making it adaptable for both research and production. TensorFlow offers multiple levels of implementation, thus users can choose the right one for their needs.

Besides a high-level Keras API for quick model development, it also allows for more detailed control by the TensorFlow API for expert users. Its support for deep learning and neural network models makes TensorFlow a widely used framework in the development of applications involving image recognition, natural language processing, and predictive analytics, making it a perfect fit for this research, as we will need more detailed and lesser detailed implementation of our custom built model.

3.4 Reusability

To ensure that this system can be used in the future, it is crucial to keep the FAIR (Findable, Accessible, Interoperable, and Reusable) principles [111] in mind when making design choices. By adhering to these principles we ensure that the system can be reused, maintained, updated, and expanded.

Besides the FAIR principles, we are also interested in the evaluation of the Technology Readiness Level (TRL) as it helps us assess the maturity of the system and provide recommendations towards progressing to the next readiness level. Further insight into the assessment of the current level of the system will be given in Section 3.4.2.

3.4.1 FAIR principles

To adhere to the FAIR principles, we ensure that during the development of our system, clear and comprehensive comments will be placed throughout the codebase. This provides the users with clarity of the functionalities of the system so that it can easily be operated, reused, and updated. Besides comments throughout the code, we will provide an informative HTML file, produced by Sphinx [112] that can be used as a guide to important algorithmic components.

We will also be using dynamic resource allocation, thereby ensuring that the system can easily be adapted to other computational resources like GPUs, or CPUs. Besides resource allocation, we will also create a dynamic directory structure by providing relative paths as opposed to absolute paths. This ensures the system can easily be adapted to the resource locations of different users.

Adhering to the findable and accessible principles of FAIR, we will make sure to make the complete codebase accessible via an on-demand publicly available GitHub repository. This repository will not only host the source code but will also include Jupyter notebooks to give examples of the code's application in various scenarios. To streamline the setup process, we will provide a requirements file for easy installation of necessary Python packages. The data will be securely stored and made available upon request as well.

3.4.2 Technology Readiness Level

The Technology Readiness Level (TRL) serves as a way to assess the maturity level of Critical Technology Elements (CTE) within various stages of a project, encompassing research, development, and deployment phases [113]. These levels are established through a Technology Readiness Assessment (TRA), which evaluates the project's conceptual designs, technological requirements, and the capabilities of the technology. Originally created by NASA during the 1970s, TRLs were intended for evaluating technologies involved in space exploration endeavours [114].

The primary objective of using Technology Readiness Levels is to provide a quantifiable measure of a technology component's maturity within a system. This measurement is crucial for project teams, as it offers insights into the development stage of a particular technology and its readiness for deployment. By assigning a TRL rating, project progress can be effectively monitored and measured [113].

TRLs are categorized on a nine-point scale, where a level of 1 indicates the earliest stage of technology development, and level 9 represents a technology that has reached full maturity [114]. The adoption of TRLs serves uniform discussions regarding the technological maturity of diverse technologies. Thereby, it ensures a standard approach to assessing and communicating technological progress [113].

Table 3.6: The nine levels of the Technology Readiness Levels (TRLs) (adopted from [113]).

Level	Definition	TRL Description
1	Basic principles observed and reported	Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include paper studies of a technology's basic properties.
2	Technology concept and/or application formulated.	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.
3	Analytical and experimental critical function and/or characteristic proof of concept.	Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.
4	Component and/or breadboard validation in laboratory environment.	Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared to the eventual system. Examples include the integration of "ad hoc" hardware in the laboratory.
5	Component and/or breadboard validation in relevant environment.	The Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so it can be tested in a simulated environment.
6	System/subsystem model or prototype demonstration in a relevant environment.	A representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness.
7	System prototype demonstration in an operational environment.	Prototype near, or at, planned operational system. Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment such as an aircraft, vehicle, or space.
8	Actual system completed and qualified through test and demonstration.	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluations of the system in its intended weapon system to determine if it meets design specifications.
9	Actual system has proven through successful mission operations.	The actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. Examples include using the system under operational mission conditions.

To assess the usability of the system, we will assess the TRL by using a publicly available TRL calculation tool. This is made publicly available by Emphasis [115], and can be found on their website [116]. Their calculation tool is based on asking questions regarding each of the levels of the TRL framework. These questions are to be answered with 'Yes', 'No', or 'NA'. After having answered each of these questions, a full report and final conclusion will be provided to the user.

This way of assessing the TRL is more user-friendly and less labour-intensive than the extensive reporting structure as suggested by the US Government Accountability Office [117], therefore this easy-to-use tool has a superior alignment with the goals and time allocation for this research, therefore we will be using the tool provided by Emphasis.

To evaluate the TRL, we completed the questionnaire three times: once for the phase of the system prior to incorporating the results from the detection and behaviour analysis phase, and again after these results had been integrated. The third assessment was done according to the predicted outcomes of the finalization of the entire drinking behaviour and identification pipeline. These assessments provided insights into the advancements and improvements made in the latest version of the system and the version after the whole pipeline had been integrated.

4. Experiments

For each of the models, we devised various experiments that need to be conducted to be able to assess and optimize the accuracy. To find the best behaviour model, we first selected the best model according to a set of model experiments. Subsequently, we performed a sensitivity analysis for all three models by tuning the parameters according to various parameter selection experiments. After the optimal set of parameters was discovered, we conducted ablation studies by several data augmentation experiments.

The goal of these experiments was to analyse the response to a change in the set of parameters for each of the models. This gave us a clear picture of the sensitivity of each of the models to the parameters, which led us to find the optimal set of parameters for each task.

Besides the parameters, we conducted ablation studies to extract relevant augmentation strategies for our problem. As there was a lot of variation in lighting, as we were dealing with a shift in daylight, exploring the impact of augmentations is crucial for the deployment in real-world scenarios.

4.1 Detection

For our detection model, we carefully selected one model which was to be trained. We selected the YOLOv10 as this is the latest version of the YOLO models [91]. For our initial detection dataset, we chose YOLOv10-X, which is the largest and most accurate model. Even though this model is the most precise, it is also computationally demanding to train. Therefore, we selected YOLOv10m, which is the medium-sized model, for training purposes.

As can be observed in Table 4.1, the largest model is indeed significantly better than the medium model, however, it also requires the most computational resources to train and the inference speed is the slowest. More specifically, it is approximately 2.5 times as big as YOLOv10-M and 2 times slower, while having a 6.5% higher accuracy. As accuracy was not the most important aspect of the model, we chose to use the YOLOv10-M model.

Table 4.1: Comparison of the various YOLO models.

Model	Input Size	APval	FLOPs (G)	Latency (ms)
YOLOv10-N	640	38.5	**6.7**	**1.84**
YOLOv10-S	640	46.3	21.6	2.49
YOLOv10-M	640	51.1	59.1	4.74
YOLOv10-B	640	52.5	92.0	5.74
YOLOv10-L	640	53.2	120.3	7.28
YOLOv10-X	640	**54.4**	160.4	10.70

4.1.1 Sensitivity Analysis

To optimize the YOLO model's performance, we conducted a systematic sensitivity analysis on key hyperparameters. We began by exploring learning rates, which we set to 0.0001, 0.0005, 0.001, 0.005, 0.01, followed by warmup epochs 1, 2, 3, 4, corresponding to 5%, 10%, and 20% of total epochs.

Next, we varied the final learning rate factor (lrf) 0.01, 0.05, 0.1, 0.001, momentum 0.9, 0.937, 0.98, 0.995, and batch sizes 8, 16, 32. Then we varied the box loss 0.5, 2.5, 5, 7.5, 10, class loss 0.1, 0.25, 0.5, 1, 2, and dfl loss 0.5, 1, 1.5, 2, 2.5.

Throughout the analysis, we maintained a constant 20 epochs for each run. We trained the model on the full dataset. We used a baseline and varied each hyperparameter in turn. For the baseline we chose: lr: 0.01, lrf: 0.01, optimizer: AdamW, batch size: 8, weight decay: 0.0005, momentum: 0.937, dropout: 0.0, box: 7.5, class: 0.5, dfl: 1.5 . We used the Standard Augmentation setup, which will be described in the subsequent paragraph. All other parameters were kept at their default values as specified in the Ultralytics documentation [118]. This approach allowed us to identify the impact of each hyperparameter on the metrics of the YOLO model training.

4.1.2 Ablation Studies

Following the sensitivity analysis, we conducted one type of ablation study to evaluate the impact of various data augmentation strategies on our YOLO model's performance.

We tested eight distinct augmentation setups: No Augmentation, Standard Augmentation, Aggressive Color Augmentation, Rotation and Translation Focus, Scaling and Cropping Focus, Horizontal Flip Only, and Black and White conversion. Each setup varied in parameters such as HSV adjustments, geometric transformations (rotation, translation, scaling), flipping, mosaic, and cropping. We maintained the optimal hyperparameters identified during the sensitivity analysis, including learning rate, warmup epochs, final learning rate factor (lrf), momentum, and batch size.

All other parameters remained at their default values as specified in the Ultralytics documentation [98]. An overview of the values of each of the parameters can be found in Table A.1 in Appendix A.1.1.

As we wanted optimal comparative results, we trained the model with each augmentation setup for 100 epochs instead of 20 epochs that were used during sensitivity analysis. The number of runs required for the full ablation study is significantly lower compared to the full sensitivity analysis. As we do not have unlimited computational resources, this was a sensible choice.

4.2 Behaviour

4.2.1 Model Selection

To determine the optimal model for our task, we conducted a comparative analysis of several convolutional neural networks: EfficientNetV2-Small, ResNet50, ResNet101, DenseNet, and MobileNet_V3_Large. Each model was initialized with pre-trained weights from IMAGENET1K_V1, leveraging transfer learning to enhance performance.

We standardized the input preprocessing across most models, using a resize of 256x256 pixels, mean values of [0.485, 0.456, 0.406], and standard deviation values of [0.229, 0.224, 0.225]. The exception was EfficientNetV2-

Small, which employed a larger input size of 384x384 pixels as it was trained on images of this size. A consistent baseline hyperparameter setup was applied across all models, featuring a learning rate of 0.001 (0.01 for frozen layers), momentum of 0.9, step size of 10, and a learning rate decay factor (gamma) of 0.1. We utilized Stochastic Gradient Descent (SGD) as the optimizer and set the batch size to 16. We used the Standard Augmentation setup which will be explained in Section 4.2.3.

This systematic approach allowed us to evaluate the performance of each model under similar conditions. This gave us a clear idea of their accuracy to determine which model would be our final choice.

4.2.2 Sensitivity Analysis

To analyze the behaviour model's performance, we conducted a systematic sensitivity analysis on the most important hyperparameters. To achieve optimal results, we conducted our hyperparameter analysis process in two distinct phases. In the first phase, we performed a sensitivity analysis with the pre-trained layers frozen. This approach leverages transfer learning while minimizing computational costs. In the second phase, we unfroze all layers and repeated the sensitivity analysis on the entire model. In this phase, we used the model weights obtained from training the frozen model with the parameters that achieved the optimal scores.

We slightly altered the baseline values for the sensitivity analysis. We chose a learning rate of 0.001, a momentum of 0.9, a step size of 3, a gamma of 0.1, a batch size of 32, and the SGD optimizer. We maintained a constant 6 epochs for each run during the first phase, except for the optimizer comparison, for which we used 10 epochs. In the second phase, during fine-tuning, we constantly used 10 epochs per run. We trained the model on the full dataset. We used the Standard Augmentation setup which will be explained in Section 4.2.3.

In the first phase, with frozen pre-trained layers, we first explored learning rates, which we set to 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01. Next, we varied the learning rate decay factor (gamma) 0.2, 0.1, 0.05, 0.01, 0.001, followed by momentum values 0.85, 0.9, 0.937, 0.98, 0.995, and batch sizes 8, 16, 32. Finally, we compared the performance of two optimizers: SGD and Adam. These values were consistent for both phases. This two-stage process enabled us to determine each hyperparameter's effect in both training phases.

4.2.3 Ablation Studies

To evaluate the impact of various data augmentation strategies on our behaviour model's performance, we conducted one type of ablation study on the behaviour model. We tested seven distinct augmentation setups: No Augmentation, Standard Augmentation, Aggressive Color Augmentation, Rotation Focus, Scaling and Cropping Focus, Horizontal Flip Only, and Grayscale Emphasis.

Each setup varied in parameters such as random rotation (0° to 30°), colour jitter (brightness, contrast, saturation, hue, and probability), random resized crop (from original size to 80% of original dimensions), random horizontal flip (0 to 100% probability), and grayscale conversion (0 to 100% probability). The Standard Augmentation setup included a balanced mix of all transformations, while other setups focused on specific aspects of augmentation. An overview of the values of each of the parameters can be found in Table A.2 in Appendix A.1.1.

To conduct these experiments, we employed the same two-phased systematic approach as during the sensitivity analysis. We first trained the EfficientNetV2 model with the baseline train parameters for 10 epochs on each augmentation setup. Consecutively, we adopted the weights of the optimal model obtained from phase one to perform the same analysis for the second phase.

This systematic approach allowed us to isolate the effects of different augmentation techniques on our behaviour model's performance in an unbiased manner by doing it once for each training phase. By doing these experiments, we obtained insights into the most effective data augmentation strategies for cow behaviour classification.

4.2.4 Drinking Time Measurement

This experiment is aimed at measuring the total drinking time of one randomly selected full hour of video obtained from the camera used for testing. As our test set contains carefully selected images according to an iterative process which gradually increases the level of complexity of the images, this does not reflect the level of complexity of one full hour of video. Therefore, we have annotated one hour of video to be compared with the recognition capabilities of the best model obtained during this research.

We employed the models that attained the best results during testing for both the detection and behaviour recognition phases. The results were then carefully corrected using the same annotation software as discussed in Section 3.1. Subsequently, we compared the predictions with the corrected behaviour classifications. This experiment gives us insight into the performance of the behaviour classification model for one hour of video without a biased image selection process.

5. Results

5.1 Detection

5.1.1 Sensitivity Analysis

Table 5.1 depicts the results of the sensitivity analysis for the YOLO model training. The analysis explored different values for learning rate, warmup epochs, final learning rate, momentum, batch size, loss weights (box, class, and DFL), and optimizer choice.

For the learning rate, values ranged from 0.0001 to 0.01. The highest mAP50 (0.8591) and precision (0.8187) were achieved with a learning rate of 0.0005. However, the highest mAP50-95 (0.6213) was obtained with 0.0001.

Warmup epochs were tested from 1 to 4, with 4 epochs yielding the best mAP50 (0.7025) and recall (0.6416). Interestingly, 3 warmup epochs provided the highest mAP50-95 (0.4531).

The final learning rate varied from 0.001 to 0.1, with 0.01 achieving the best overall performance (mAP50: 0.6939, mAP50-95: 0.4531).

Momentum values ranged from 0.9 to 0.995. The optimal value was found to be 0.937, which yielded the highest mAP50 (0.6939) and mAP50-95 (0.4531).

Batch sizes of 8 and 16 were evaluated, with the larger batch size of 16 achieving better results across most metrics (mAP50: 0.7191 vs 0.6939). As training with a batch size of 32 gave out-of-memory issues, it could not be tested.

For box loss, a weight of 10 provided the best mAP50 (0.7164) and recall (0.6703). Class loss weight of 2 yielded the highest mAP50 (0.7590) and recall (0.6805). For DFL loss, a weight of 2 gave the best mAP50 (0.7059) and recall (0.6789).

Lastly, two optimizers were compared: Adam and SGD. SGD significantly outperformed Adam on all metrics, with superior results in mAP50

Table 5.1: Metrics from the sensitivity analysis on the validation set.

		Metric						
Parameter	Value	mAP50	mAP50-95	precision	recall	box loss	class loss	dfl loss
Learning Rate	0.0001	0.8139	**0.6213**	0.7563	0.7162	2.1763	1.8967	2.9012
	0.0005	**0.8591**	0.5864	**0.8187**	**0.7637**	**2.0959**	**1.6413**	**2.7999**
	0.001	0.8241	0.5540	0.7778	0.7328	2.2227	1.9523	2.9193
	0.005	0.7877	0.4691	0.7916	0.7099	2.3433	2.1279	3.0322
	0.01	0.6939	0.4531	0.6807	0.6204	2.7079	2.5657	3.4075
Warmup Epochs	1	0.5504	0.2666	0.6476	0.4500	4.0415	3.6628	4.5804
	2	0.6760	0.4337	**0.6839**	0.6041	3.0892	2.9851	3.7621
	3	0.6939	**0.4531**	0.6807	0.6204	**2.7079**	**2.5657**	**3.4075**
	4	**0.7025**	0.4351	0.6709	**0.6416**	3.1833	2.7257	3.8934
Learning Rate Final	**0.01**	**0.6939**	**0.4531**	0.6807	0.6204	**2.7079**	2.5657	**3.4075**
	0.05	0.6900	0.4369	0.6278	**0.6692**	2.9370	**2.4882**	3.6343
	0.1	**0.6894**	0.4282	**0.6916**	0.6107	3.0051	2.5769	3.7829
	0.001	0.6815	0.4442	0.6516	0.6479	2.8972	2.5429	3.5784
Momentum	0.9	0.6883	0.4348	0.6493	0.6531	2.9569	2.6198	3.6683
	0.937	**0.6939**	**0.4531**	**0.6807**	0.6204	**2.7079**	**2.5657**	**3.4075**
	0.98	0.6808	0.4226	0.6605	**0.6570**	3.0624	2.6670	3.6305
	0.995	0.6071	0.3247	0.5495	0.6393	3.6677	3.2401	4.1413
Batch Size	8	0.6939	0.4531	0.6807	0.6204	**2.7079**	2.5657	**3.4075**
	16	**0.7191**	**0.4661**	**0.7156**	**0.6462**	2.9221	**2.3333**	3.6336
	32	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Box Loss	0.5	0.6990	0.4219	0.6435	0.6635	**0.2116**	2.8938	3.7591
	2.5	0.6594	0.4316	0.6563	0.6209	0.9768	2.8207	3.6287
	5	0.6821	0.4387	**0.6960**	0.6164	2.0128	2.6959	3.7580
	7.5	0.6939	**0.4531**	0.6807	0.6204	2.7079	2.5657	**3.4075**
	10	**0.7164**	0.4506	0.6775	**0.6703**	3.9170	**2.3664**	3.6437
Class Loss	0.1	0.6276	0.3839	0.6667	0.5757	3.1998	**0.5919**	3.9823
	0.25	0.6776	0.4203	0.6155	0.6697	2.9060	1.2290	3.6219
	0.5	0.6939	**0.4531**	0.6807	0.6204	**2.7079**	2.5657	**3.4075**
	1	0.6882	0.4313	0.6531	0.6554	2.9587	4.9197	3.6307
	2	**0.7590**	0.4274	**0.7437**	**0.6805**	3.4959	10.0360	3.8157
DFL Loss	0.5	0.6863	0.4396	0.6850	0.6047	2.9403	2.5484	**1.2024**
	1	0.6753	0.4386	0.6849	0.5975	3.0081	2.6545	2.4931
	1.5	0.6939	0.4531	0.6807	0.6204	**2.7079**	2.5657	3.4075
	2	**0.7059**	**0.4545**	0.6448	**0.6789**	2.9060	2.4497	4.8716
	2.5	0.7055	0.4464	**0.6956**	0.6237	2.8443	**2.3276**	6.0052
Optimizer	Adam	0.6939	0.4531	0.6807	0.6204	2.7079	2.5657	3.4075
	SGD	**0.7753**	**0.5551**	**0.8075**	**0.6525**	**2.2862**	**2.1817**	**2.9526**

(0.7753 vs 0.6939) and precision (0.8075 vs 0.6807).

Across all parameters, mAP50 consistently showed higher values compared to mAP50-95, indicating better performance at lower IoU thresholds. The various loss components (box, class, and DFL) showed different trends depending on the parameter being tuned. Overall the baseline parameters performance was competitive with the best parameter values.

5.1.2 Ablation Studies

Table 5.2 depicts the results of ablation studies on various data augmentation techniques for the YOLO model, conducted over 100 epochs. Standard augmentation showed the highest overall performance, improving mAP50 by 9.0% (from 0.8984 to 0.9796) and mAP50-95 by 22.8% (from 0.6073 to 0.7459) compared to no augmentation. Precision and recall also increased by 12.9% and 14.8%, respectively.

Table 5.2: Metrics from the ablation study on the validation set.

	Metric						
Setup	mAP50	mAP50-95	precision	recall	class loss	box loss	dfl loss
No Augmentation	0.8984	0.6073	0.8354	0.8137	1.7169	2.2105	4.4609
Standard Augmentation	**0.9796**	**0.7459**	**0.9435**	0.9342	0.9243	**1.5914**	**3.3660**
Aggressive Color Augmentation	0.9772	0.7274	0.9421	0.9452	0.9016	1.6599	3.4525
Rotation and Translation Focus	0.9634	0.6629	0.9345	0.8712	1.2145	1.9430	3.8071
Scaling and Cropping Focus	0.9303	0.6530	0.9373	0.7973	1.3430	1.8825	3.6341
Horizontal Flip Only	0.9758	0.7026	0.9304	**0.9527**	**0.8881**	1.8698	3.8535
Black and White	0.8984	0.6073	0.8354	0.8137	1.7169	2.2105	3.5437

Aggressive colour augmentation and horizontal flip-only setups performed similarly to standard augmentation, with mAP50 values of 0.9772 and 0.9758, respectively. The horizontal flip setup achieved the highest recall (0.9527) among all configurations.

Rotation and translation-focused augmentation outperformed no augmentation but underperformed compared to standard augmentation, with a mAP50 of 0.9634. Scaling and cropping-focused augmentation showed the least improvement among the augmented setups, with a mAP50 of 0.9303 and the lowest recall (0.7973).

The black-and-white conversion setup matched the performance metrics without any augmentation, except for a lower DFL loss (3.5437 vs 4.4609).

All augmentation strategies reduced losses compared to no augmentation, with standard augmentation achieving the lowest overall losses (class: 0.9243, box: 1.5914, DFL: 3.3660).

5.1.3 Final Models

Table 5.3 presents the results of three model configurations trained on the test set for 100 epochs, each incorporating different combinations of optimized parameters and augmentation techniques.

Table 5.3: Performance metrics for the best detection models.

	Metric			
Setup	mAP50	mAP50-95	precision	recall
Best Params Best Augment	**0.9910**	**0.8710**	**0.9610**	**0.9670**
Best Learning Rate Best Augment	0.9850	0.7980	0.9560	0.9560
Best Optimizer Best Augment	0.9830	0.8280	0.9550	0.9420

The Best Params Best Augment setup, which combined optimal hyperparameters with standard augmentation, achieved the highest overall performance. This configuration improved mAP50 by 0.6% and mAP50-95 by 9.1% compared to the Best Learning Rate Best Augment setup. It also outperformed the Best Optimizer Best Augment configuration by 0.8% in mAP50 and 5.2% in mAP50-95.

The Best Learning Rate Best Augment setup, which only optimized the learning rate (0.0005) while keeping other parameters at baseline, showed strong performance with a mAP50 of 0.9850 and mAP50-95 of 0.7980. This configuration achieved identical precision and recall values of 0.9560.

The Best Optimizer Best Augment setup, which used SGD instead of AdamW while maintaining other baseline parameters, demonstrated improved performance over the Best Learning Rate configuration in terms of mAP50-95 (0.8280 vs 0.7980). However, it showed slightly lower mAP50 (0.9830 vs 0.9850) and recall (0.9420 vs 0.9560).

All three configurations exhibited high precision and recall values, ranging from 0.9550 to 0.9670, indicating robust detection capabilities across different parameter combinations. The Best Params Best Augment setup achieved the highest recall (0.9670) among the three configurations.

5.1.4 Qualitative Results

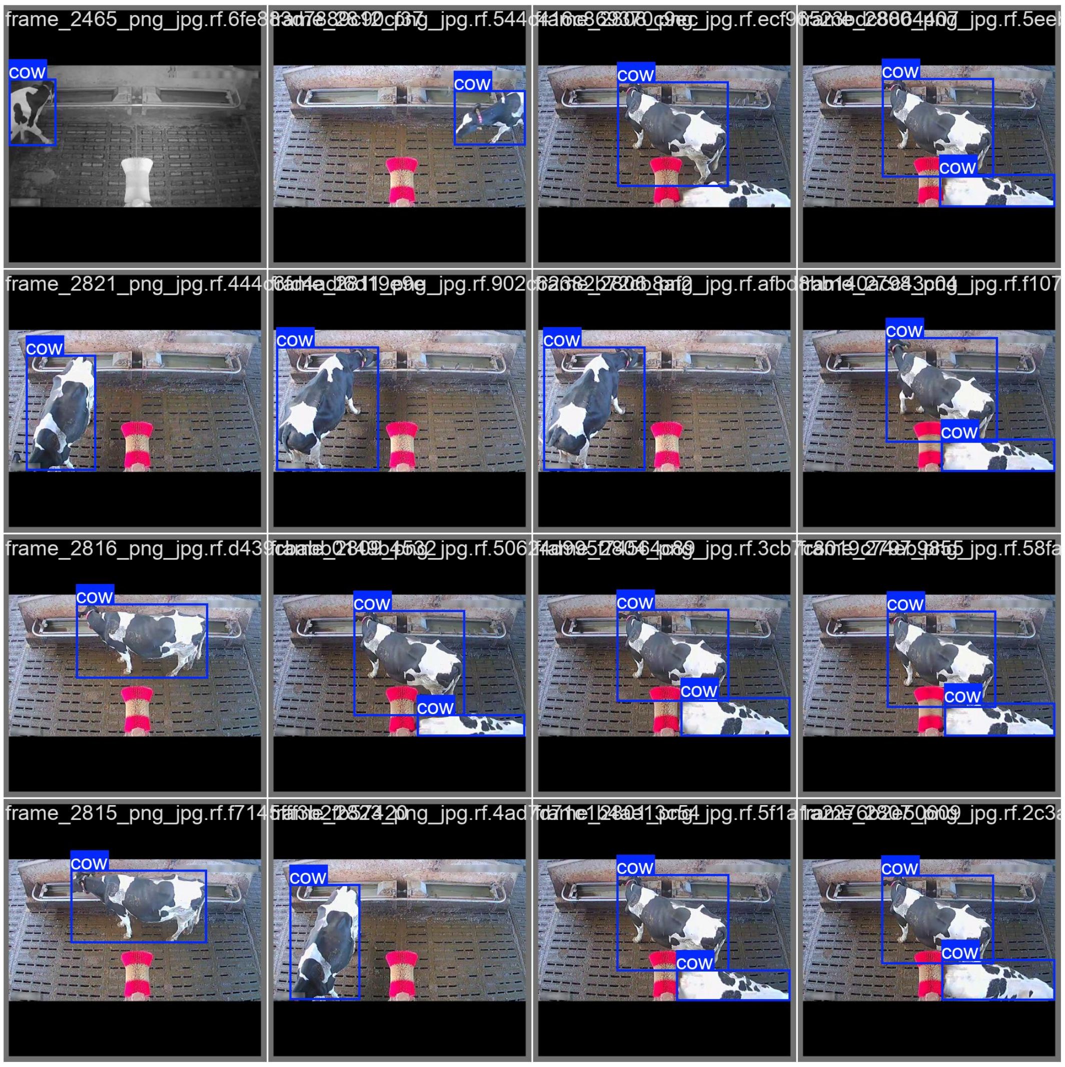


Figure 5.1: Test batch ground truth bounding boxes for several frames.

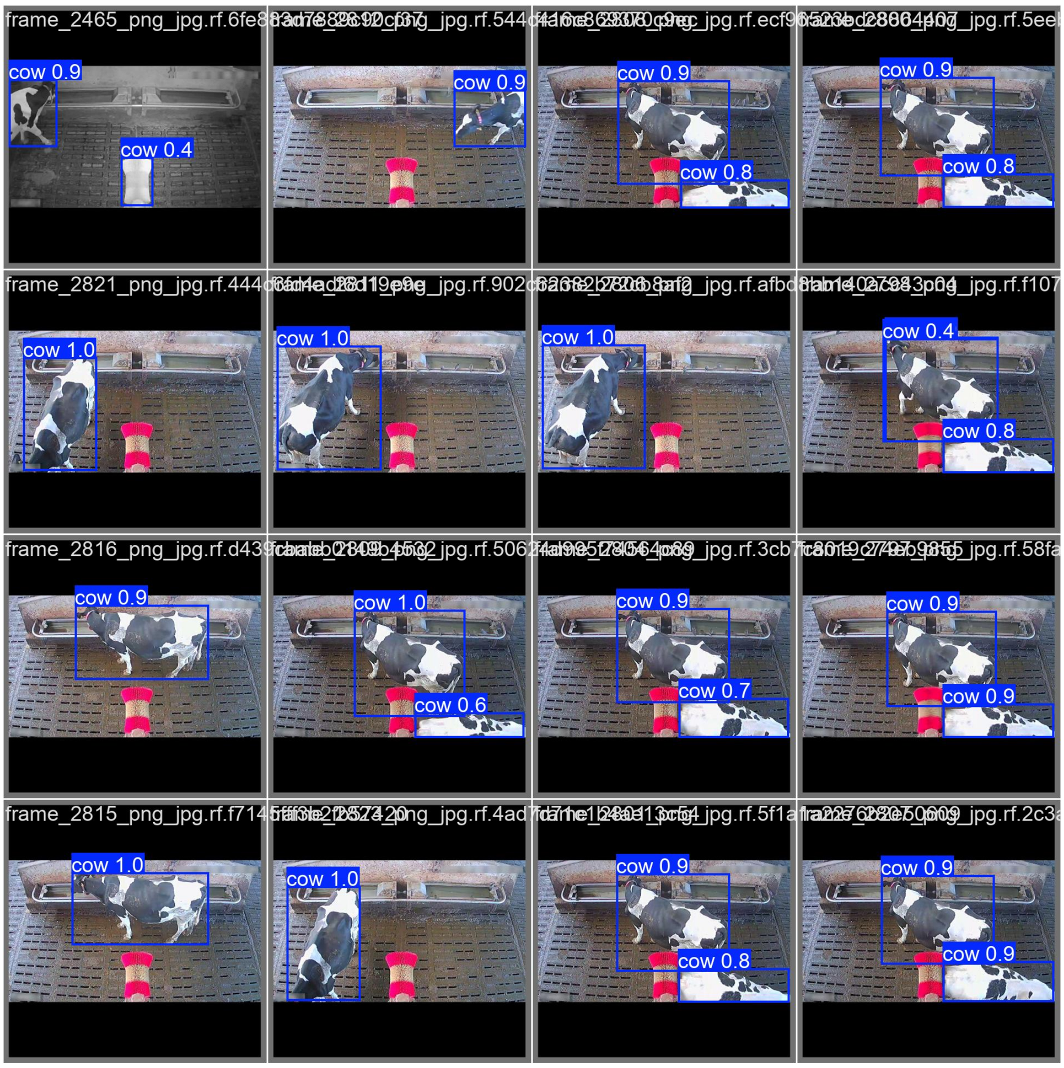
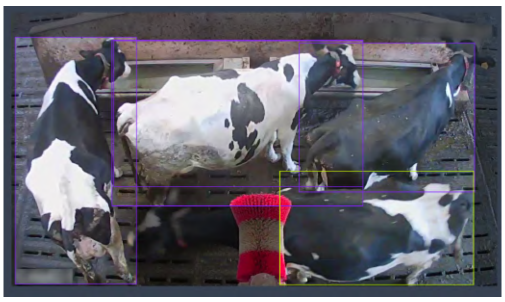
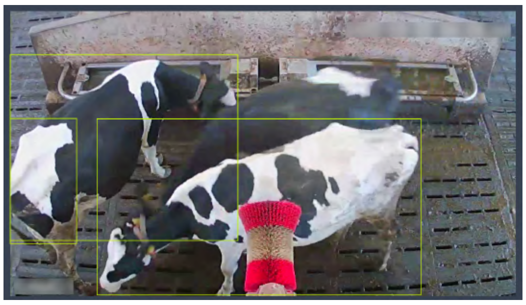


Figure 5.2: Test batch predicted bounding boxes for several frames. The confidence level is given for each bounding box. This indicates the level of confidence of the cow class.



(a) Incorrect number of bounding boxes



(b) Undetected cow

Figure 5.3: Manual review of incorrect detections.

5.2 Behaviour

5.2.1 Models

The results of the model evaluations are presented in Table 5.4. The table compares the six different neural network architectures: EfficientNetV2S, ResNet50, ResNet101, MobileNetV3, DenseNet121, and DenseNet169. For each model, the two phases are evaluated.

Table 5.4: Metrics from the chosen models on the validation set.

		Metric					
Model	Phase	time	accuracy	precision	recall	f1	ROC AUC
EfficientNetV2S	Freeze	7.4519	0.6474	0.5972	0.8942	0.7161	0.6487
	Tune	8.7053	**0.8342**	0.7647	**0.9630**	**0.8525**	**0.8349**
ResNet50	Freeze	7.7713	0.6395	0.6008	0.8201	0.6935	0.6404
	Tune	7.4551	0.7474	0.6946	0.8783	0.7757	0.7481
ResNet101	Freeze	7.6801	0.5842	0.5529	0.8571	0.6722	0.5856
	Tune	7.7331	0.8000	0.7467	0.9048	0.8182	0.8005
MobileNetV3	Freeze	7.5040	0.5947	0.5785	0.6825	0.6262	0.5952
	Tune	7.9042	0.6605	0.6027	0.9312	0.7318	0.6619
DenseNet121	Freeze	8.0009	0.5763	0.5424	0.9471	0.6898	0.5783
	Tune	**7.6012**	0.8053	0.7751	0.8571	0.8141	0.8055
DenseNet169	Freeze	7.7220	0.6395	0.5884	0.9153	0.7164	0.6409
	Tune	7.9745	0.8158	0.7644	0.9101	0.8309	0.8163

The performance metrics reported include inference time (in seconds), accuracy, precision, recall, F1 score, and ROC AUC. Across all models, the tuning phase generally showed improved performance compared to the freezing phase.

EfficinetNet obtained the highest accuracy (0.6474) and ROC AUC (0.6487) during the freeze phase. ResNet50 exhibited slightly higher precision (0.6008 vs 0.5972), whereas DenseNet121 showed a substantially higher recall compared to EfficientNetV2S (0.9471 vs 0.8942).

EfficientNetV2S obtained the highest accuracy (0.8342), recall (0.9630), F1 score (0.8525), and ROC AUC (0.8349) in the tuning phase. ResNet101 and DenseNet121 also showed notable improvements after tuning, with accuracies of 0.8000 and 0.8053, respectively. MobileNetV3 exhibited the smallest improvement from freezing to tuning, while still showing gains across all metrics. Inference times varied slightly across models and phases, ranging from 7.4519 seconds (EfficientNetV2S, Freeze) to 8.7053 seconds (EfficientNetV2S, Tune). The results indicate that the EfficientNetV2S exhibited the best performance overall for both phases. Therefore, we will conduct the sensitivity analysis on this model.

5.2.2 Sensitivity Analysis

Frozen Model

Table 5.5 presents the results of the sensitivity analysis for the freeze phase by training the best model; EfficientNetV2S. The analysis explored different values for learning rate, gamma, momentum, batch size, and optimizer choice.

Table 5.5: Metrics from the sensitivity analysis freeze phase on the validation set.

				Metric		
Parameter	Value	accuracy	precision	recall	f1	ROC AUC
	0.00001	0.6421	0.5950	0.8783	0.7094	0.6433
	0.00005	0.6368	0.5882	0.8995	0.7113	0.6382
	0.0001	0.6316	0.5814	**0.9259**	0.7143	0.6331
Learning Rate	0.0005	0.6289	0.5833	0.8889	0.7044	0.6303
	0.001	0.6342	0.5906	0.8624	0.7011	0.6354
	0.005	**0.6737**	**0.6236**	0.8677	**0.7257**	**0.6747**
	0.01	0.6395	0.5884	0.9153	0.7164	0.6409
	0.2	**0.6632**	0.6041	**0.9365**	**0.7344**	**0.6646**
	0.1	0.6342	0.5906	0.8624	0.7011	0.6354
Gamma	0.05	0.6526	**0.6084**	0.8466	0.7080	0.6536
	0.01	0.6526	0.6059	0.8624	0.7118	0.6537
	0.001	0.6447	0.5964	0.8836	0.7122	0.6460
	0.85	0.6289	0.5828	0.8942	**0.7056**	0.6303
	0.9	**0.6342**	**0.5906**	0.8624	0.7011	**0.6354**
Momentum	0.937	0.6237	0.5782	0.8995	0.7039	0.6251
	0.98	0.6184	0.5738	**0.9048**	0.7023	0.6199
	0.995	0.6132	0.5745	0.8571	0.6879	0.6144
	8	**0.6842**	**0.6245**	0.9153	**0.7425**	**0.6854**
Batch Size	16	0.6605	0.6027	**0.9312**	0.7318	0.6619
	32	0.6342	0.5906	0.8624	0.7011	0.6354
Optimizer	Adam	0.6421	0.5957	0.8730	0.7082	0.6433
	SGD	**0.6474**	**0.5972**	**0.8942**	**0.7161**	**0.6487**

For the learning rate, values ranged from 0.00001 to 0.01. The highest accuracy (0.6737) precision (0.6236), and F1 score (0.7257) were achieved with a learning rate of 0.005. The baseline value of 0.001 exhibited lesser results.

Gamma values were tested from 0.001 to 0.2, with 0.2 yielding the best performance (accuracy: 0.6632, recall: 0.9365, F1 score: 0.7344 ROC AUC: 0.6646). This indicates the baseline gamma achieved poorer performance.

Momentum varied from 0.85 to 0.995. The optimal value was found to be 0.9 (accuracy: 0.6342, precision: 0.5906, ROC AUC: 0.6354), with recall and f1 slightly under the best scores; 0.8624 vs 0.9048, and 0.7011 vs 0.7056 respectively.

Batch sizes of 8, 16, and 32 were evaluated, with the smallest batch size of 8 achieving the best results, showing a significant improvement compared to the baseline of 32 (0.6842 vs 0.6342).

Lastly, two optimizers were compared: Adam and SGD. SGD slightly outperformed Adam on all metrics.

Across all parameters, recall consistently showed high values, often exceeding 0.85, while precision remained relatively lower, typically between 0.57 and 0.62. ROC AUC scores closely resembled the accuracy values for each parameter setting.

Fine-Tuning

Table 5.6 presents the results of the sensitivity analysis for the fine-tuning phase of the EfficientNetV2S model. The same values for each of the hyper-parameters were used for the fine-tuning as for the freezing. For tuning we used the model weights obtained from training the frozen model with the parameters that achieved the optimal scores. This was found to be the baseline model with a batch size of 8 which used the Standard Augmentation setup.

The optimal performance was achieved with a learning rate of 0.0005, yielding the highest accuracy (0.8895), precision (0.8731), and F1 score (0.8912). Performance declined sharply for learning rates above 0.001.

The best performance for gamma was observed at 0.1, with the highest accuracy (0.8842), precision (0.8502), and F1 score (0.8889). Performance decreased as gamma deviated from this value.

Table 5.6: Metrics from the sensitivity analysis tune phase on the validation set.

				Metric		
Parameter	Value	accuracy	precision	recall	f1	ROC AUC
Learning Rate	0.00001	0.7632	0.6941	0.9365	0.7973	0.7641
	0.00005	0.8421	0.7792	0.9524	0.8571	0.8427
	0.0001	0.8789	0.8235	**0.9630**	0.8878	0.8794
	0.0005	**0.8895**	**0.8731**	0.9101	**0.8912**	**0.8896**
	0.001	0.8842	0.8502	0.9312	0.8889	0.8845
	0.005	0.6947	0.6409	0.8783	0.7411	0.6957
	0.01	0.7026	0.6387	0.9259	0.7559	0.7038
Gamma	0.2	0.8105	0.7388	0.9577	0.8341	0.8113
	0.1	**0.8842**	**0.8502**	0.9312	**0.8889**	**0.8845**
	0.05	0.8000	0.7344	0.9365	0.8233	0.8007
	0.01	0.7289	0.6593	0.9418	0.7756	0.7301
	0.001	0.7237	0.6511	**0.9577**	0.7752	0.7249
Momentum	0.85	0.6974	0.6350	0.9206	0.7516	0.6985
	0.9	**0.8842**	**0.8502**	0.9312	**0.8889**	**0.8845**
	0.937	0.6974	0.6331	0.9312	0.7537	0.6986
	0.98	0.7579	0.6902	0.9312	0.7928	0.7588
	0.995	0.7421	0.6667	**0.9630**	0.7879	0.7433
Batch Size	8	0.6868	0.6699	0.7302	0.6987	0.6871
	16	0.6789	0.6136	**0.9577**	0.7479	0.6804
	32	**0.8842**	**0.8502**	0.9312	**0.8889**	**0.8845**
Optimizer	Adam	0.7816	0.7137	**0.9365**	0.8101	0.7824
	SGD	**0.8842**	**0.8502**	0.9312	**0.8889**	**0.8845**

The optimal value for momentum was found to be 0.9, achieving the highest accuracy (0.8842), precision (0.8502), F1 score (0.8889) and ROC AUC (0.8845). Performance dropped notably for values above and below 0.9.

Interestingly, the largest batch size of 32 produced the best results across all metrics (accuracy: 0.8842, F1 score: 0.8889), contrasting with the freeze phase results.

For optimizers, SGD outperformed Adam across all metrics, with significant improvements in accuracy (0.8842 vs 0.7816) and F1 score (0.8889 vs 0.8101).

Across all parameters, recall consistently showed high values, often exceeding 0.90. Precision values were generally higher compared to the freeze phase, typically ranging from 0.63 to 0.87. ROC AUC scores followed the accuracy values for each parameter setting.

5.2.3 Abblation Studies

Frozen Model

Table 5.7 depicts the results of the ablation study for the freeze phase. Seven different augmentation setups were evaluated.

Table 5.7: Metrics from the ablation study freeze phase on the validation set.

	Metric				
Augmentation	accuracy	precision	recall	f1	ROC AUC
No Augmentation	**0.7000**	0.6728	0.7725	0.7192	**0.7004**
Standard Augmentation	0.6921	0.6429	0.8571	**0.7347**	0.6930
Aggressive Color Augmentation	0.6553	0.6250	0.9206	0.7265	0.6558
Rotation Focus	0.6868	0.6357	**0.9312**	0.7012	0.6878
Scaling and Cropping Focus	0.5974	0.5621	0.8624	0.6192	0.5988
Horizontal Flip Only	0.6711	0.6301	0.8360	0.7126	0.6718
Grayscale Emphasis	0.5974	**0.8333**	0.3651	0.4742	0.5962

The No Augmentation setup achieved the highest accuracy (0.7000) and ROC AUC (0.7004). However, Standard Augmentation yielded the best F1 score (0.7347). Rotation Focus produced the highest recall (0.9312), while Grayscale Emphasis resulted in the highest precision (0.8333) but at the cost of significantly lower recall. Scaling and Cropping Focus and Grayscale Emphasis both showed the lowest accuracy (0.5974). Across most augmentation strategies, recall values were consistently high, often exceeding 0.80, while precision varied more widely from 0.5621 to 0.8333.

Fine-Tune

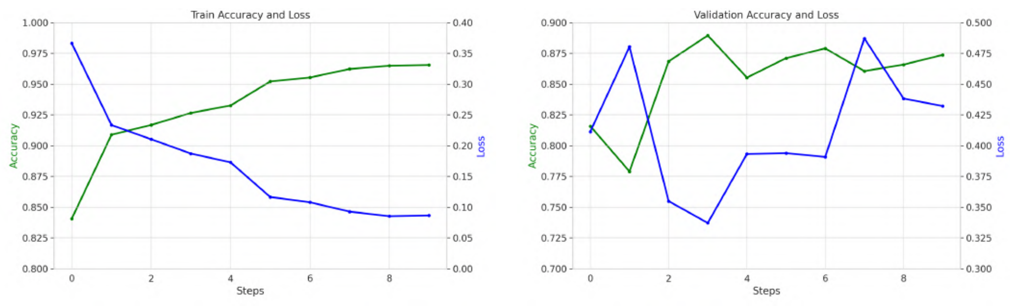
Table 5.8 gives the results of the ablation study for the fine-tuning phase, using the same augmentation strategies as in the freeze phase. The performance metrics generally improved compared to the freeze phase across all augmentation setups.

Test Results

Grayscale Emphasis achieved the highest accuracy (0.8526) and ROC AUC (0.8528), a significant improvement from its performance in the freeze phase. Standard Augmentation produced the best F1 score (0.8585). Scaling and Cropping Focus yielded the highest recall (0.9471) but at the cost of lower

Table 5.8: Metrics from the ablation study tune phase on the validation set.

	Metric				
Augmentation	accuracy	precision	recall	f1	ROC AUC
No Augmentation	0.7974	0.7642	0.8571	**0.8080**	0.7977
Standard Augmentation	0.8474	0.7964	0.9312	**0.8585**	0.8478
Aggressive Color Augmentation	0.8289	0.8131	0.8519	0.8320	0.8291
Rotation Focus	0.8079	0.7929	**0.8307**	0.8114	0.8080
Scaling and Cropping Focus	0.6974	0.6303	**0.9471**	0.7569	0.6987
Horizontal Flip Only	**0.8105**	**0.7577**	0.9101	0.8269	**0.8110**
Grayscale Emphasis	**0.8526**	**0.8276**	0.8889	0.8571	**0.8528**



(a) Train results **(b)** Validation results

Figure 5.4: The train and validation metrics of the final behaviour classification model.

precision and overall accuracy. Horizontal Flip Only showed balanced performance with the highest precision (0.7577) and decent scores in other metrics. The No Augmentation setup, while improved from the freeze phase, generally underperformed compared to most augmentation strategies in the fine-tuning phase.

Across all augmentation strategies in the fine-tuning phase, recall values remained high, typically above 0.85, while precision showed improvement compared to the freeze phase, ranging from 0.6303 to 0.8276.

5.2.4 Final Models

The final evaluation of the best models was conducted using three distinct approaches, each leveraging the insights gained from the sensitivity analysis and ablation study. These approaches aimed to determine the most effective combination of hyperparameters and augmentation strategies.

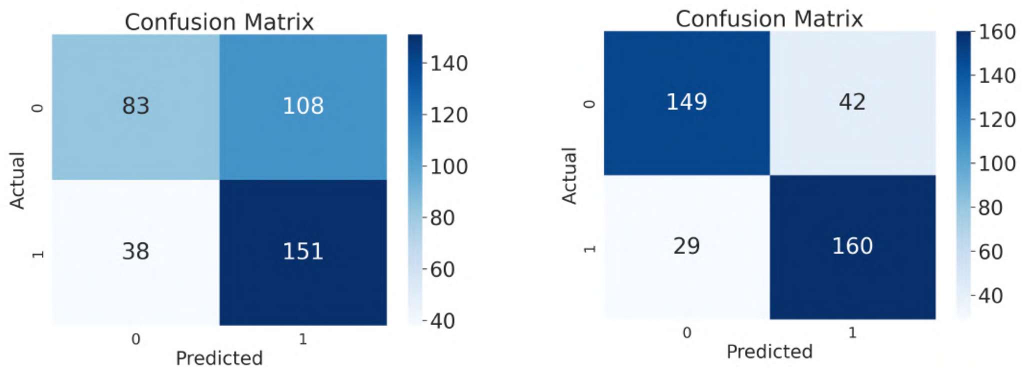
Freeze and Fine-Tune

This version incorporated both frozen training and fine-tuning phases. The freeze phase utilized the optimal hyperparameters found during sensitivity analysis, including a learning rate of 0.005, momentum of 0.9, gamma of 0.2, SGD optimizer, and a batch size of 8. We found the optimal augmentation setup was no augmentation. During the freeze phase, we trained the model for 10 epochs.

The subsequent fine-tuning phase employed the set of hyperparameters found to attain the best score. This included a reduced learning rate of 0.0005, a gamma of 0.1, and an increased batch size of 32. The fine-tuning phase also introduced Grayscale Emphasis augmentation as it gave the best results and was trained for 20 epochs. The final results are given in Table 5.9 and in Figure 5.5.

Table 5.9: Performance metrics for Freeze and Tune phases.

	Accuracy	F1 Score	Precision	Recall	ROC AUC
Freeze	0.6158	0.6741	0.5830	0.7989	0.6167
Tune	0.8132	0.8141	0.7921	0.8466	0.8133



(a) Freeze phase **(b)** Fine-tune phase

Figure 5.5: Confusion matrices for freeze and fine-tune phase of the model with the best parameters on the validation set.

Fine-Tune

This approach focused solely on fine-tuning, bypassing the freeze phase entirely. It utilized the best-performing model from the freeze baseline (accuracy: 0.6842, F1 score: 0.7425) as a starting point. The fine-tuning process adopted the optimal hyperparameters identified for fine-tuning, including a learning rate of 0.0005, momentum of 0.9, gamma of 0.1, SGD optimizer, and a batch size of 32. Grayscale Emphasis augmentation was applied during the 20-epoch fine-tuning process.

The metrics of the model are the following: accuracy: 0.8684, precision: 0.8716, recall: 0.8624, f1: 0.8670, and ROC AUC: 0.8683. The confusion matrix is given in Figure 5.6.

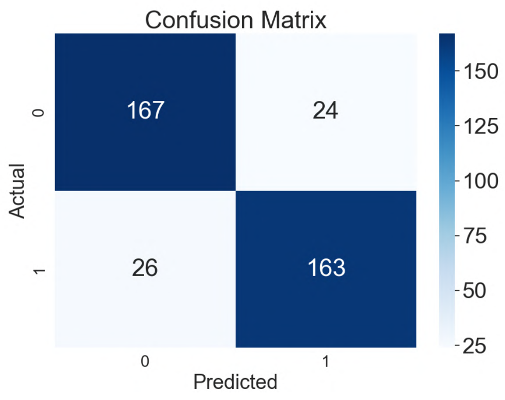


Figure 5.6: Confusion matrix for the freeze phase of the model with the best parameters on the validation set.

Best Model Overall

The third version is built upon the best-performing fine-tuned model overall. This is the model that was obtained during the sensitivity analysis. It is important to note that this version did not require any additional training as opposed to versions 1 and 2, as this model was found during the analysis. It has therefore been trained on fewer epochs, but it still attained a better result on the validation set. This is the model with a learning rate of 0.0005 and all other hyperparameters were set to the baseline values.

The metrics of the model are the following: accuracy: 0.8895, precision: 0.8731, recall: 0.9101, f1: 0.8912, and ROC AUC: 0.8896. The confusion matrix is given in Figure 5.7.

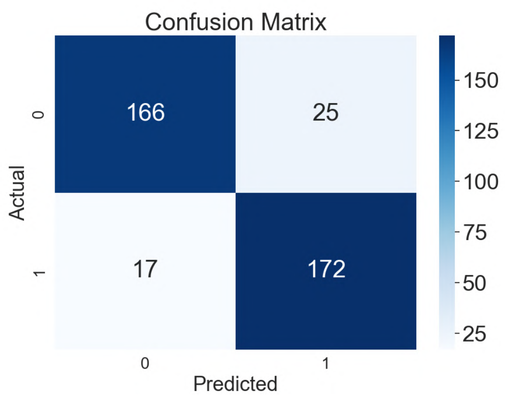


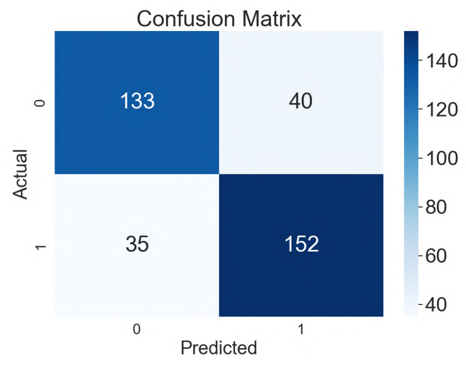
Figure 5.7: Confusion matrix of the model with the best parameters on the validation set.

Results Test Set

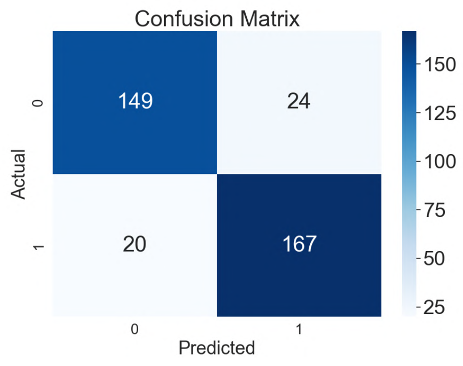
This section presents the performance results of the three model versions on the test set. Each version represents a different training approach: version 1 combines freeze and fine-tune phases, version 2 focuses solely on fine-tuning, and version 3 utilizes a baseline freeze model with optimal fine-tuning. The models' performance is evaluated using standard metrics including accuracy, precision, recall, F1 score, and ROC AUC, which are provided in Figure 5.10. Additionally, confusion matrices are provided to offer a detailed view of each model's classification performance in Table 5.7.

Table 5.10: Performance metrics for the three model versions on the test set.

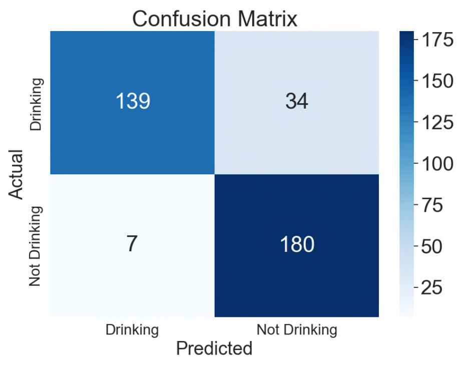
	accuracy	precision	recall	f1	ROC AUC
Freeze and Tune	0.7917	0.7917	0.8128	0.8021	0.7908
Tune Only	0.8778	**0.8743**	0.8930	0.8836	0.8772
Best Overall	**0.8861**	0.8411	**0.9626**	**0.8978**	**0.8830**



(a) Freeze and fine-tune



(b) Fine-tune only



(c) Best overall

Figure 5.8: Confusion matrices for all phases of the model evaluated on the test set.

5.2.5 Drinking Time Measurement

Here we discuss the results obtained from the full one-hour video. We give the calculated metrics for both classes separately and the confusion matrix. The results are given in Table 5.11, and Figure 5.9.

Table 5.11: Performance metrics for final model on the full hour test video.

	Metric			
	accuracy	precision	recall	f1
Drink	0.9602	0.9246	0.9197	0.9221
Non-Drink	0.9602	0.9724	0.9741	0.9733

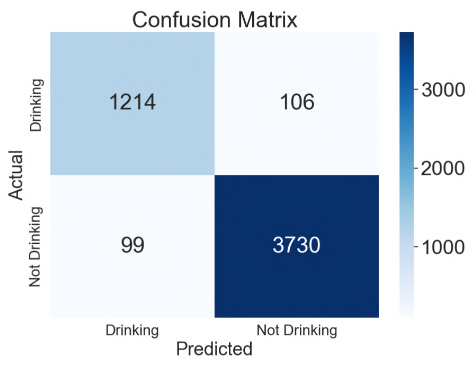
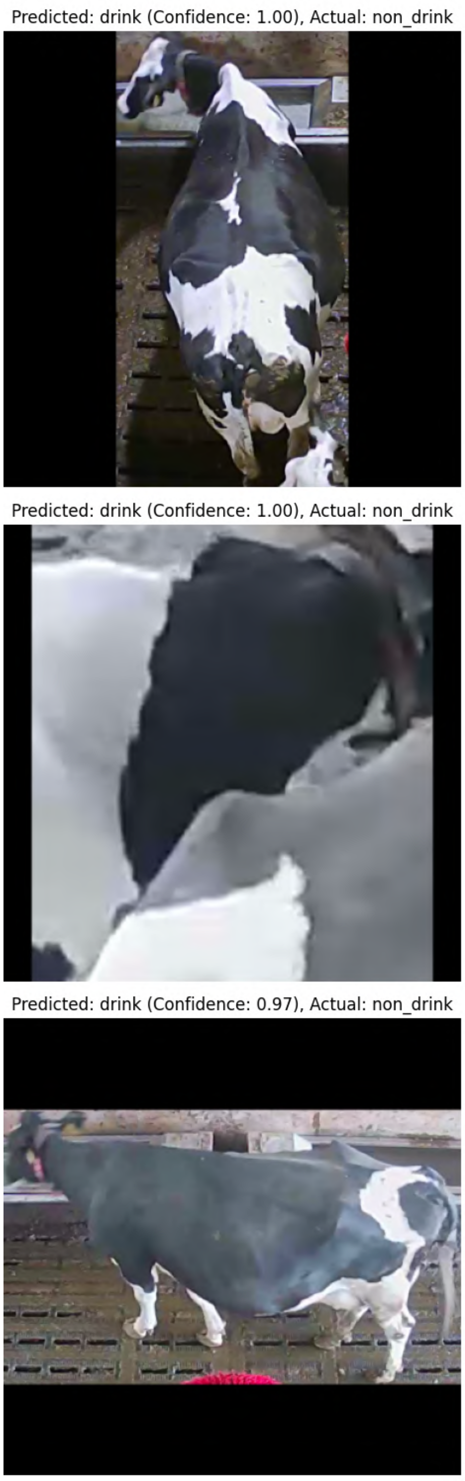


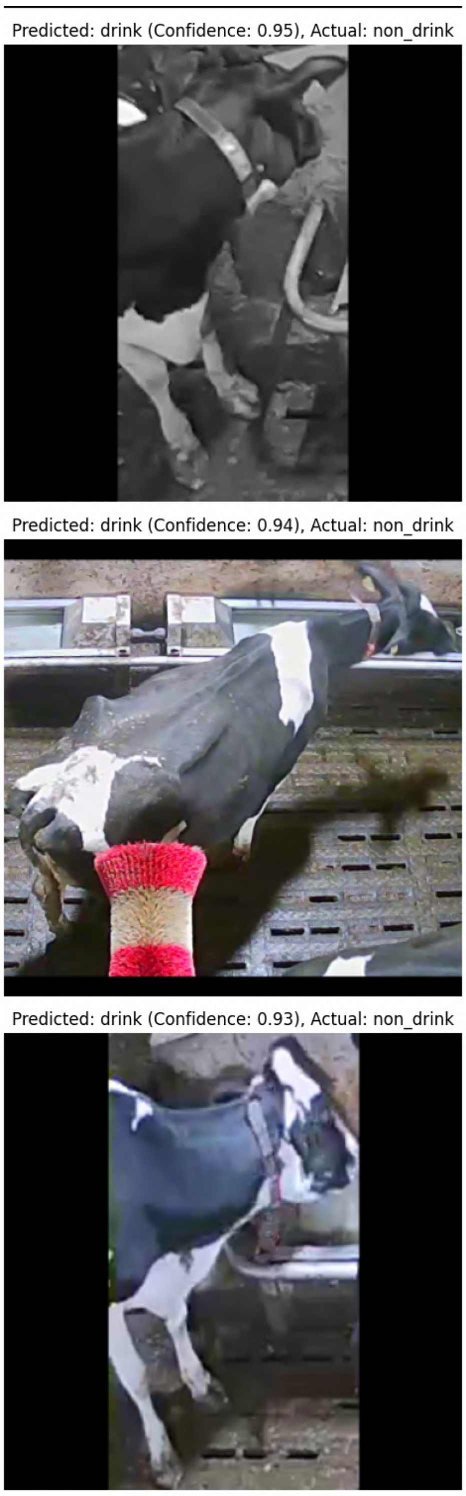
Figure 5.9: Confusion matrix for the full hour test video.

The time estimation error is calculated by the percentage error and the absolute error. As we extracted two frames per second from the video, the estimated drinking time was 656.5 seconds, while the true time was 607.0 seconds. This equates to an absolute error of 49.5 seconds or a percentage error of 8.15%.

5.2.6 Qualitative Results

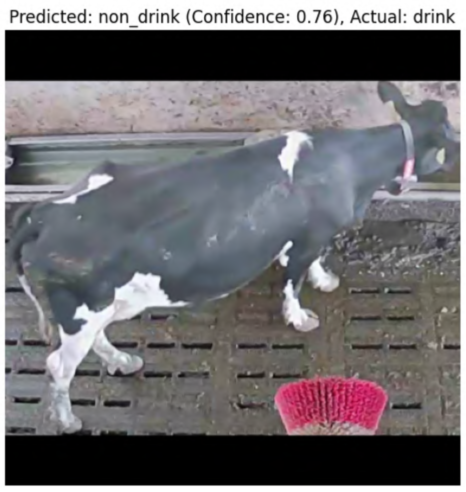

Predicted: drink (Confidence: 1.00), Actual: non_drink
Predicted: drink (Confidence: 1.00), Actual: non_drink
Predicted: drink (Confidence: 0.97), Actual: non_drink

(a) Non drinking classified as drinking

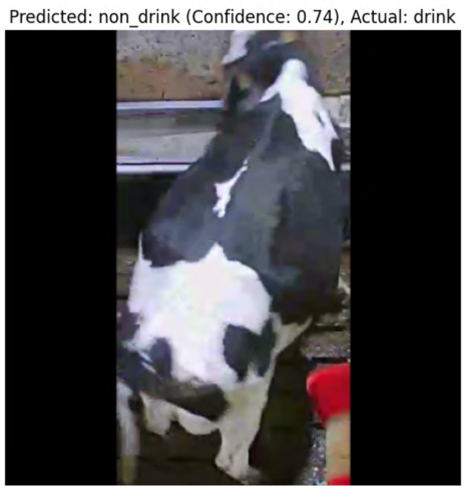

Predicted: drink (Confidence: 0.95), Actual: non_drink
Predicted: drink (Confidence: 0.94), Actual: non_drink
Predicted: drink (Confidence: 0.93), Actual: non_drink

(b) Non drinking classified as drinking

Figure 5.10: Most confident incorrect drink predictions of the best model on the test set.

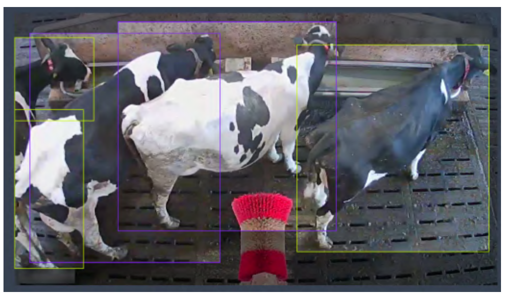


(a) Drinking classified as non drinking

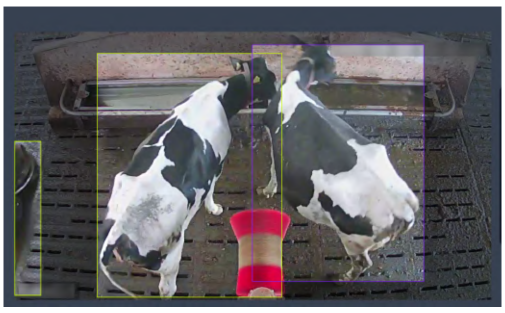


(b) Drinking classified as non drinking

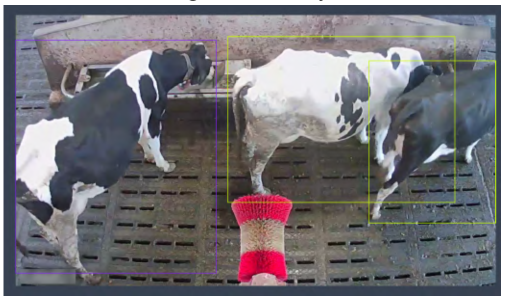
Figure 5.11: Most confident incorrect non-drink predictions of the best model on the test set.



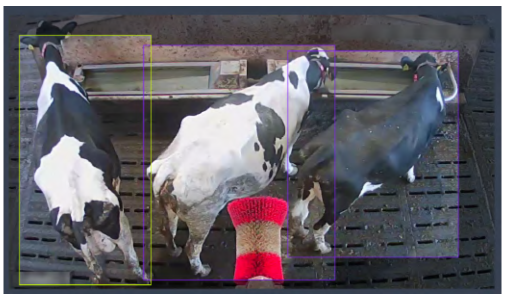
(a) Drinking incorrectly classified



(b) Drinking incorrectly classified



(c) Drinking incorrectly classified



(d) Drinking correctly classified

Figure 5.12: Manually selected correct and incorrect drinking behaviour classifications from the test set. A purple bounding box indicates drinking and a yellow bounding box indicates non-drinking.

5.3 Reusability

The Technology Readiness Level (TRL) assessment of the system yielded the following results. In the initial phase (Pre) where only the identification component was tested rigorously, 8 out of 53 questions were answered positively. After the behaviour analysis phase (Post), this increased to 13 positive answers. For the projected full system integration (Pipeline), 18 positive answers were recorded. The majority of affirmative responses were concentrated in the first section of questions, which primarily addressed the conceptual and early developmental stages of the technology. No positive responses were recorded for questions related to market assessment, intellectual property, or safety and deployment planning across all three phases.

6. Discussion

This thesis investigated the capabilities of computer vision models to detect cows and analyse their drinking behaviour to be able to assess the duration of their drinking. This research adds two important components to the final system that tracks the drinking duration of each cow in a farm environment. This system could measure the total time they are drinking in a day which facilitates research into the relation between milk production and drinking behaviour.

To accomplish this we devised two components of the system. The detection component is based on YOLO (You Only Look Once) and the behaviour analysis component uses EfficientNetV2-Small. After fully assessing the capabilities of each of these components they can be integrated into one system that can constantly track the drinking behaviour. A key challenge addressed in this study is the development of a generalizable model capable of deployment across diverse farm environments without necessitating extensive retraining.

The main research question is: "How can we accurately recognize drinking behaviour of each individual cow in a herd of cows using computer vision?". To address this question, we first needed to identify the components needed. After these were identified they were trained separately and evaluated on an unseen test set to analyse their performance in a new unseen environment. To do this we devised two sub-questions.

The first question; "How can we accurately and robustly detect cows in videos?" was explored through a detection experiment that validated optimal hyperparameters and augmentation strategies for the YOLO-based model.

The second question; "How can we reliably classify drinking and non-drinking behaviour of cows in videos?" was addressed by a classification experiment that identified the most effective model architecture, hyperparameters, and data augmentation techniques for behaviour analysis.

To assess the system's practical applicability and identify areas for further enhancement we answered the question: "How usable is the system in real-world scenarios, and what steps are required to enhance it further?" by employing the Technology Readiness Level (TRL) framework [119]. This evaluation provides insights into the system's current state of development and its potential for real-world implementation.

6.1 Findings and Interpretation

This section presents a comprehensive analysis of the experimental results, offering interpretations of model performance and discussing the implications of these findings for monitoring dairy cow drinking behaviour.

6.1.1 Detection

6.1.1.1 Sensitivity Analysis

To determine the optimal hyperparameters for the model, we conducted an extensive evaluation of various carefully selected hyperparameters and their respective values. This analysis revealed several noteworthy patterns, as illustrated in Figure 5.1. The learning rate emerged as a critical factor influencing model performance, with higher values leading to deterioration in performance. This observation emphasizes the importance of warmup epochs, during which the model begins training with a smaller learning rate. Extending the duration of the warmup phase has been shown to enhance the model's performance.

The impact of the final learning rate is less significant, as the majority of learning occurs during the initial epochs, as can be observed when evaluating the training results depicted in Figure A.2.

Momentum, which is intrinsically related to the learning rate, plays an important role in guiding the model towards the optimal solution. While it aids in accelerating convergence when moving in the correct direction, high momentum values can cause the model to converge to a local optimum or follow a sub-optimal path. This phenomenon likely explains why setting the momentum to an extremely high value (0.995) resulted in significantly poorer performance compared to other momentum values, which exhibited relatively minor differences in their effects.

Looking at the losses we can see that the mAP metrics improve with increasing box loss values, peaking at a value of 10. This suggests that a higher emphasis on box loss helps the model better localize objects, leading to more accurate detections. This also leads to higher precision and lower loss, which is as expected. No other clear patterns were observed.

For the class loss, we see that the mAP50 shows a significant increase at a class loss value of 2, indicating that a higher focus on class loss improves the model's ability to correctly classify objects. However, mAP50-95 does not follow a clear trend, suggesting that extreme values might lead to overfitting. Precision increases notably at higher class loss values, while recall improves but not as significantly. This indicates that the model becomes more confident in its correct classifications. Box loss increases with higher class loss values, indicating a trade-off where improving classification accuracy might slightly degrade localization performance.

For the DFL loss, we see that both mAP50 and mAP50-95 show improvement with increasing DFL loss values, peaking around 2 and 2.5. This suggests that a higher emphasis on DFL loss helps the model better focus on the most relevant features for detection.

As expected all losses decrease drastically when increasing the loss parameter value. We can see that having a clear balance between the losses is important as this enhances model performance.

Ablation Studies

From Figure 5.2, it is evident that the model's performance is significantly impaired when no augmentation is applied or when only black and white images are used. Augmentation addresses issues such as occlusion, which remain unresolved in the absence of augmentation.

Additionally, the orientation of the cows plays a critical role in model performance. The Horizontal Flip Only setup achieves results comparable to the best-performing setup, Standard Augmentation. This shows the necessity of horizontal flipping to achieve optimal results, indicating that despite the training set containing approximately 9000 detections, there are insufficient orientations for the model to generalize effectively.

While cropping may simulate occlusion, it can also introduce challenging examples, as illustrated in Figure A.1. In these instances, the model encounters nearly undetectable objects, such as in image [1,1]. Nevertheless, the model attained competitive results, suggesting that it has been trained on a sufficient number of detectable examples.

Colour is shown to be crucial when observing that the Black and White augmentation setup gives relatively poor results compared to Aggressive Color Augmentation.

Test Results

When comparing the test results to the validation results a striking pattern emerges. The results are depicted in Table 6.1.

Table 6.1: Comparison of model performance on validation and test sets

	Setup	mAP50	mAP50-95	precision	recall	class loss	box loss	DFL loss
Validation	Best Params Best Augment	0.8727	0.6578	0.8179	0.7884	2.6555	6.5433	3.4265
	Best Learning Rate Best Augment	**0.9874**	**0.8011**	**0.9601**	**0.9562**	**1.2948**	**0.7431**	**2.9894**
Test	**Best Params Best Augment**	**0.9910**	**0.8710**	**0.9610**	**0.9670**	-	-	-
	Best Learning Rate Best Augment	0.9850	0.7980	0.9560	0.9560	-	-	-

From Table 6.1, several patterns and anomalies emerge. The Best Params Best Augment setup on the validation set achieves a mAP50 of 0.8727 and a mAP50-95 of 0.6578, with precision and recall values of 0.8179 and 0.7884, respectively. On the test set, the performance improves significantly, with a mAP50 of 0.9910 and a mAP50-95 of 0.8710, and precision and recall values of 0.9610 and 0.9670, respectively. This indicates that the model generalizes well to unseen data.

The Best Learning Rate Best Augment setup shows excellent performance on the validation set, with a mAP50 of 0.9874 and a mAP50-95 of 0.8011, and precision and recall values of 0.9601 and 0.9562, respectively. However, on the test set, the performance slightly drops, with a mAP50 of 0.9850 and a mAP50-95 of 0.7980, and precision and recall values of 0.9560 and 0.9560, respectively.

The Best Params Best Augment setup has a higher class loss (2.6555) and box loss (6.5433) on the validation set compared to the Best Learning Rate Best Augment setup, 1.2948, and 0.7431 respectively. Despite this, it achieves better performance on the test set, suggesting that the model might be overfitting less to the training data. The Best Learning Rate Best Augment setup, while performing exceptionally well on the validation set, shows a slight decrease in performance on the test set, suggesting potential overfitting.

A notable pattern is a significant improvement in performance from the validation set to the test set for the Best Params Best Augment setup, which is unusual. Typically, one would expect the performance to be consistent or slightly lower on the test set due to the model encountering completely unseen data. This could be due to the positioning of the camera from which the images of the test set were composed. This set of images does not contain any background images, potentially decreasing the level of complexity of the dataset.

These observations highlight the importance of evaluating models on both validation and test sets to ensure robust performance and compare final results.

Tuning

To enhance the selection of optimal parameter values, we explored further tuning using evolutionary algorithms. Despite the potential of evolutionary algorithms to efficiently search large hyperparameter spaces [120], our attempts did not yield satisfactory solutions. We conducted three separate experiments, each spanning 20 epochs and 50 iterations, using a range of values consistent with those employed during the sensitivity analysis. Unfortunately, each attempt resulted in convergence to a local optimum, failing to achieve significant performance improvements.

Evolutionary algorithms operate by iteratively evolving a population of candidate solutions through mechanisms inspired by natural selection, such as mutation, crossover, and selection [121]. These algorithms are particularly effective for complex optimization problems with large search spaces. However, in our case, the size of the search space likely contributed to early convergence, preventing the discovery of globally optimal solutions.

Given the computational constraints, we decided to prioritize sensitivity analysis over exhaustive tuning. While evolutionary algorithms could theoretically be applied to tune augmentation strategies as well, the extensive computational resources required were not available. Consequently, we abandoned the idea of devising an experiment for augmentation tuning as we achieved satisfactory results from sensitivity analysis.

Detection Capabilities

As we closely examine Figures 5.1 and 5.2, several critical observations can be made. In the image [1,3] (indicating row 1, column 3) of Figure 5.1, representing the ground truth, it is evident that the cow at the bottom is not annotated. However, when comparing this with image [1,3] in Figure 5.2, we observe that the model successfully detects this cow. This discrepancy highlights the model's robustness, as it can identify objects even when the ground truth annotations are incomplete.

Additionally, in image [1,1] of Figure 5.2, the model demonstrates its capability to confidently detect cows under various nightly conditions. This indicates that the model's performance is not significantly affected by the lighting conditions in which the image was captured.

However, it is also notable that in the same image, the model erroneously identifies the backscratcher as a cow. This misclassification might be attributed to the white colour of the scratcher during nighttime. The model predicted this with low confidence, and thus, applying a confidence threshold of 0.5 would disregard this detection.

Setting the confidence threshold at this level can lead to problematic outcomes. While most cows in Figure 5.2 are detected with high confidence, this is not the case for the cow near the drinking trough in image [2,4], which is classified with a confidence of 0.4. Adhering to a 0.5 threshold would result in neglecting this detection. Given the importance of detecting cows exhibiting drinking behaviour, lowering the confidence threshold is not problematic, as our behaviour classification model can filter out non-relevant detections. Consequently, the backscratcher, despite having the same confidence level as the cow in image [2,4], would be filtered out. Figure 5.2 shows that no cow was left undetected, indicating the model's high precision.

By manually selecting more challenging detections, we can identify examples that the model struggles with, as illustrated in Figure 5.3. Despite YOLOv10's use of a single head during inference, which should eliminate the need for Non-Max Suppression (NMS) [70], the model still predicts some erroneous boxes. Addressing these issues can be achieved through two approaches: applying NMS as a post-processing step to eliminate erroneous boxes or utilizing the identification model's capabilities to infer unnecessary bounding boxes when the same cow is identified twice with high confidence. Future research should determine the preferred approach.

In addition to the erroneous bounding box in Figure 5.3b, we observe that one cow remains undetected, as it is occluded by the cow in front. Although this undetected cow is not engaged in drinking behaviour, it is crucial to address such scenarios. One potential solution is to add various examples containing occlusion, or augmenting the dataset with specific augmentation techniques. Employing harder augmentation techniques, such as random erase, cutout, and mosaic, may be beneficial [122]. Placing the camera directly above the drinking trough could prevent occlusion as well, but this slightly shifts the problem domain. Additionally, integrating tracking capabilities into the detection model, which is planned for future implementation, may resolve this issue [123].

Comparison to Other Models

To compare the capabilities of our best model, we have compared its results with 3 other studies that devised detection models for dairy cows. As direct comparison is hard due to the varying nature of the papers, caution must be taken. An overview of the results is given in Table 6.2.

In the study conducted by Moradeyo et al. [56], the researchers applied

Table 6.2: Detection model comparison.

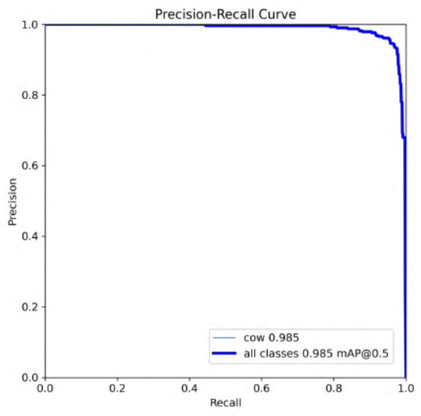
Model	Paper	mAP50	mAP50-95	Precision	Recall
YOLOv10	Our Model	0.9910	**0.8710**	**0.9610**	**0.9670**
YOLOv7	Moradeyo et al. [56]	0.9500	-	-	-
YOLOv5-EMA	Zhang et al. [58]	0.9510	-	0.9480	0.9030
Faster R-CNN	Andrew et al. [57]	**0.9960**	-	-	-

YOLOv7 to livestock image detection and segmentation tasks, focusing on cattle grazing behaviour, monitoring, and intrusions. Data was collected using a high-resolution video camera positioned on a high pole in a cattle ranch, capturing images of Nigerian beef cattle. The dataset consisted of 1050 images, with 80% used for training and 20% for testing. The model was trained for 55 epochs with a batch size of 16 and a confidence threshold of 0.5. The study reported a mAP of 0.95 for the YOLOv7 model.

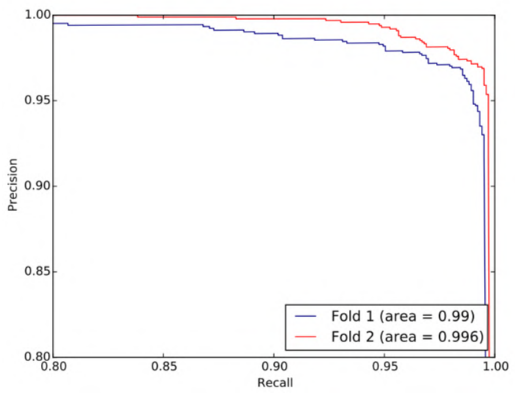
However, their research had several shortcomings. The dataset size was relatively small (1050 images), which may limit the generalizability of the model. Additionally, environmental conditions were controlled to reduce noise, which might not reflect real-world variability. Precision and recall metrics were not reported, making direct comparison difficult. Therefore, the results should be interpreted with caution due to the controlled environment and small dataset size. The model's performance in more diverse conditions might differ.

Andrew et al. [57] focused on visual localization and individual identification of Holstein Friesian cattle. They implemented cattle detection and localization using the VGG CNN-M 1024 network adapted for the Faster R-CNN framework. Their model was evaluated using a combination of datasets of inside and outside cattle seen from above, resulting in 1,077 images for 2-fold cross-validation. Their model achieved mAP50 values of 0.99 and 0.996 for the two folds, respectively.

We further compare the results with the precision-recall curves. The precision-recall curves for both models illustrate near-perfect performance in detecting cattle. The Faster R-CNN model's precision-recall curves, as shown in Figure 6.1b, exhibit mAP50 values of 0.99 and 0.996 for the two folds. Similarly, our YOLOv10 model's precision-recall curve demonstrates a mAP50 of 0.985, indicating comparable high-level performance.



(a) Our model



(b) Model by Andrew et al. [57]

Figure 6.1: Comparison of precision-recall curves of our model and the model by Andrew et al. [57].

Zhang et al. [58] proposed a novel YOLOv5-EMA model for accurate cattle body detection, incorporating the Efficient Multi-Scale Attention (EMA) module. The dataset included 8024 images of 113 cattle, captured from various angles and conditions. The model achieved an overall mAP@50 of 95.1% and a mAP@50 of 94.8% for cattle body detection. They also detected legs and heads and reported an overall precision, recall and f1 score for these classes of 94.8%, 90.3%, and 92.5%, respectively.

The dataset was relatively large and diverse, and they showed promising results for cattle detection as well as the detection of specific body parts by using an attention mechanism. They also reported the results without an attention mechanism, which were reported as 93.7%, 89.0%, and 91.3% for precision, recall and f1 score. The attention mechanism shows slight improvements compared to the model without the attention mechanism.

When comparing our YOLOv10 model to these studies, considering the differences in datasets, detection tasks, and reported metrics is crucial. Our model shows excellent performance in terms of mAP@50 and mAP@50-95, precision, and recall. However, the variability in datasets and specific detection tasks in the other studies means that direct comparisons should be made with caution.

It would be optimal to test our model on the datasets used in these studies to get conclusive answers. However, the results clearly indicate that our model performance is on par, or better than other cattle detection models.

6.1.2 Behaviour

Model Selection

To evaluate the results obtained by the model selection we will take another look at the results table given in Table 5.4. Several patterns emerge regarding the performance of the evaluated architectures. EfficientNetV2S achieves the highest overall performance across all metrics when fine-tuned, indicating its ability to effectively learn discriminative features for distinguishing between drinking and non-drinking behaviours. The architecture's efficient design, which is optimized through compound scaling, allows it to achieve high accuracy while maintaining computational efficiency [39].

ResNet50 and ResNet101 exhibit the expected pattern as the larger architecture performs significantly better. The same is true for the smaller and larger versions of DenseNet. A thing to notice, however, is that the accuracy, precision, f1, and ROC AUC of ResNet50 during freezing are higher compared to the same metrics of the larger version during freezing. The reason for this might be attributed to an incomplete random seeding initialization. Even though we set a random seed, in hindsight this should have been done for many processes. As we make use of many packages; Numpy, Pytorch, CUDA, dataset shuffling, and data augmentation, we should have set random seeds for all of these to eliminate any randomness [124]. This might be the reason why the performance results of the larger model and smaller model are not as expected.

A pattern that can be observed that applies to all models is that the precision is always lower than the accuracy and the recall. This shows that each model is superior in predicting when a cow is not drinking as opposed to when it is drinking. As accuracy is describes both recall and precision in one metric, it is logical that this is higher than precision due to the addition of the higher recall. The fact that recall is higher is not surprising either, as these images are generally harder to classify.

Another important metric is the inference time. The inference time analysis is impacted by the use of Databricks and Azure Blob Storage, introducing latency that reduces the inherent differences in inference speed among the architectures. To obtain reliable comparisons of computational efficiency, local evaluations are necessary [125]. Ideally, larger models like ResNet101 and DenseNet169 would have slower inference times compared to lightweight architectures like MobileNetV3.

Sensitivity Analysis

During the freeze phase of training the behaviour model, we conducted a sensitivity analysis to understand the impact of various hyperparameters on the model's performance. The hyperparameters investigated include the learning rate, gamma, momentum, batch size, and optimizer. Table 5.5 presents the results of this analysis.

Examining the learning rate, we observe that the model's performance is relatively stable across different values, with a slight improvement at higher learning rates. The highest accuracy, precision, and ROC AUC are achieved with a learning rate of 0.005. This suggests that the model benefits from a moderately high learning rate during the freeze phase, which is as expected as freezing often requires a relatively high learning rate [126].

For the gamma parameter, there is no real pattern to be observed. Momentum, however, exhibits a more pronounced impact on the model's performance. Higher momentum values lead to a slight decrease in accuracy and ROC AUC and accuracy. The best performance is achieved with a momentum of 0.9, resulting in an accuracy of 0.6342 and a ROC AUC of 0.6354. This indicates that a balanced momentum value is optimal for the freeze phase.

Regarding batch size, the model's performance improves with smaller batch sizes. The highest accuracy of 0.6842 and a ROC AUC of 0.6854 are obtained with a batch size of 8. This suggests that the model benefits from processing fewer samples in each batch during the freeze phase. This is interesting as it is the exact opposite of the pattern we observed for the detection model. The reason for this pattern might be because the model can focus more on a first set of examples, update its weights and then evaluate the next batch instead of having to focus on many examples at once.

Finally, comparing the Adam and SGD optimizers, we observe that SGD slightly outperforms Adam. SGD achieves an accuracy of 0.6474 and a ROC AUC of 0.6487, while Adam yields an accuracy of 0.6421 and a ROC AUC of 0.6433. This indicates that SGD is more suitable for optimizing the model during the freeze phase even though the differences are minimal. Overall, the sensitivity analysis of the freeze phase reveals that the model's performance is relatively robust to changes in hyperparameters.

It must also be noted that the metrics are quite low overall. This is because at this point the model only uses the features it extracted using the pre-trained ImageNet weights. During this frozen phase, it can not learn how to extract new features, therefore it might not be able to effectively use the features to discern more nuanced differences between images. As the ImageNet weights are obtained from training the model to discern between 1000 classes [87], these more subtle differences might not be effectively extracted using the features obtained from using the pre-trained weights.

Following the freeze phase, we proceeded to fine-tune the entire network, allowing all layers to adapt to the target task. During this phase, we conducted another sensitivity analysis to investigate the impact of various hyperparameters on the model's performance. Table 5.6 presents the results of this analysis.

Examining the learning rate, we observe that the model's performance improves significantly as the learning rate decreases, reaching a peak at 0.0005 with an accuracy of 0.8895 and a ROC AUC of 0.8896. However, further decreasing the learning rate leads to a sharp decline in performance, suggesting that excessively low learning rates can hinder the fine-tuning process. This is as expected as was previously discussed during the freeze phase [126].

The gamma parameter shows a different trend compared to the freeze phase, where no clear pattern emerged. Here we observe that a low gamma gives very poor results, whereas a higher value improves the metrics drastically. However the performance peaks at 0.1 and declines sharply again at 0.2. This suggests that a balanced gamma is crucial for the model to have time to learn, but not have a relatively high learning rate throughout the entire training session.

For the varying momentum and batch size, we see a strange pattern emerging as the performance of the baseline compared to the other values is superior. We would expect to see a steady increase or decrease in the performance of the model when increasing the values of momentum, but this is the case either as the performance is poor for 0.9 (0.6974), and then increases drastically for the baseline (0.8842) and is exactly the same for 0.937 as for 0.9 (0.6974). This leap in performance is most likely caused by the incorrect initialization of random seeds pre-training as discussed in Section 6.1.2. This is the most probable cause for the same pattern when varying the batch size as well.

We also see a substantial gap between the Adam and SGD optimizers when fine-tuning. This might be because of the difference in the way they operate. Several studies have shown that SGD tends to generalize better than Adam [127, 128], especially for image classification tasks. Although Adam may converge faster during training, SGD often achieves better performance on the test set. This difference in generalization could be more pronounced during fine-tuning, where the model is adapting to the specific task. If this were true, then we would see a lower train loss for the Adam optimizer compared to the SGD optimizer and a higher validation loss.

From the paper by Zhou et al. [127] we also found that Adam adapts the learning rate for each parameter based on its historical gradients, while SGD uses a single learning rate for all parameters. During fine-tuning, when the model is learning task-specific features, the adaptive learning rates of Adam may lead to overfitting or convergence to sharp minima. In contrast, SGD's single learning rate may provide a more stable and generalizable solution. This would therefore result in the same result; a lower train loss for Adam and a higher validation loss.

After further investigation, we found that the opposite is true. The train loss for Adam is 0.248 as opposed to 0.089 for SGD and for validation, Adam obtained a loss of 0.576 whereas SGD had a loss of 0.409. How these results are obtained remains puzzling as the literature indicates that the exact opposite should be observed. Our only logical explanation for this phenomenon is that it has to do with the incorrect setting of the random seeds with the same consequences as described earlier.

The reason why this difference is not observed during the freezing phase could be attributed to the fact that during freezing, only the last layer(s) of the model are trained, while the rest of the weights are fixed. This limits the model's ability to adapt to the specific task. The difference in generalization ability between SGD and Adam may not be as apparent in this scenario, as the model's capacity to overfit is restricted. However, a definitive correct answer to this question could not be uncovered.

Ablation Studies

The ablation study results reveal several patterns, of which the most intriguing pattern is that the No Augmentation setup achieves the best performance during the freeze phase. During the freeze phase of transfer learning, the pre-trained weights of the convolutional layers are fixed, and only the fully connected layers are trained. The pre-trained convolutional layers have already learned to extract meaningful features, and the model is able to leverage these features without the need for data augmentation. By freezing these layers, we preserve the learned representations and prevent them from being overwritten or distorted by the augmented data. Using no augmentation during freezing allows the fully connected layers to focus on learning the most relevant patterns without being influenced by added augmentations.

In contrast, during the fine-tuning phase data augmentation greatly enhances the model's generalization and performance. At this stage, the model has the flexibility to adapt its weights to the specific characteristics of features needed for correct classification, and data augmentation can help prevent overfitting and improve the model's robustness. This is the trend we observe when comparing the freezing and fine-tuning phases. No augmentation performs poorly when fine-tuning, but excels during the freezing phase.

Another striking observation is the poor performance of the Grayscale Emphasis setup during the freeze phase, which is in contrast to its excellent performance during fine-tuning. This can be attributed to the nature of transfer learning. During the freeze phase, the pre-trained weights are learned on colour images and may not be well-suited for extracting features from grayscale images thereby leading to suboptimal performance. However, during fine-tuning, the entire network adapts to the grayscale images, thus allowing the model to learn more discriminative features specific to grayscale images.

This aligns with the findings of Xie et al. [129], who demonstrated that pre-training on grayscale ImageNet can improve medical image classification. They demonstrate that using grayscale images with a model pre-trained on colour images can lead to suboptimal classification performance during the freeze phase. This occurs because the model's weights, optimized for extracting features from colour images, do not effectively generalize to grayscale data. However, during fine-tuning, the model can adapt its weights and learn features relevant to the grayscale domain, significantly improving its performance.

The Scaling and Cropping Focus setup yields relatively poor accuracy, precision, and F1 scores compared to other setups, particularly during fine-tuning. This can be explained by the challenging examples introduced by scaling and cropping augmentations, which can generate nearly undetectable objects, making it difficult for the model to learn robust features. The same phenomenon occurred during ablation studies of the detection model, thus these results are not surprising.

Test Results

When evaluating the test results of the final three models we see the opposite pattern emerging compared to the detection test results. When the final model is trained on the best hyperparameters and augmentation setup found during both phases and evaluated on the test set, the performance is poor compared to the results obtained during the sensitivity analysis. The model that was trained for fewer epochs and only varied in one hyperparameter (shown in Figure 5.4a) showed to have the best results in both the validation (Figure 5.6) as the test set (Figure 5.10).

The opposite results were observed for the detection model. The reason might be that the model is more sensitive to the combination of different hyperparameters, which were not evaluated. To overcome this problem, we can make use of tuning. Unfortunately, we ran out of time to be able to implement a tuning algorithm for the behaviour model. However, as we tested our models and obtained competitive results, implementing a tuning algorithm, even though possibly obtaining superior results, is left for future research.

Recognition Capabilities

The behaviour model is able to better classify non-drinking behaviour compared to drinking behaviour. This can be seen when taking into account the quantitative metrics given by precision and recall. In line with this observation, we see the same pattern when examining the qualitative results.

When taking a closer look at Figure 5.10, we see that the model makes various confident mistakes in classifying non-drinking behaviour as drinking behaviour. In images [1,1], [2,2], [3,1], and [3,2] we can see that the head of the cow is right above the drinking through. Discerning the behaviour here is hard even for humans. It is no surprise that the model incorrectly classified these complex scenes. However, looking at image [2,1], the reason is not as clear why the model classified this cow as drinking with 100% confidence. The image has a lot of occlusion, which is generally hard for the model to process.

Occlusion can be observed in Figure 5.12c as well. Here the model incorrectly classified the drinking cow as not drinking as the head is behind the torso of the cow in front. Even though we can not see the head, it is evident to humans that this cow is drinking. This could be solved by placing the camera in a different position or adding another camera. Adding another camera allows the model to classify the behaviour according to different viewpoints, which will increase the amount of information the model has for a scene, thereby potentially improving the classification process.

For hard examples, the model is not always able to correctly tell for each cow whether it is drinking as can be observed in Figures 5.12a and 5.12b.

The cows are observed to be in near proximity to the drinking trough, therefore, the examples are hard. In spite of this, a human can easily tell that these cows are not drinking, whereas the model failed to do so. However, the model is able to correctly classify hard scenes as can be observed in Figure 5.12d.

Comparison to Other models

To position our behaviour recognition model's performance in the literature, we compared it with recent studies on cow and pig drinking and feeding behaviour recognition.

Comparing the study by Bello et al [60] we find some key similarities and differences. Firstly, while both studies utilized deep learning techniques, the specific model architectures employed were distinct. Bello et al. [60] evaluated four pre-trained object detection models; Mask R-CNN, Faster R-CNN, YOLOv3, and YOLOv4 for individual cow detection, with Mask R-CNN achieving the highest accuracy and speed of 20 fps. In contrast, our study used YOLOv10 for cow detection and EfficientNetV2 for subsequent behaviour classification.

Furthermore, the datasets used in the two studies, while similar in total size, had notable differences in composition and structure. Bello et al. [60] acquired video data from six cows on a ranch, from which they selected 1000 keyframes. 800 of these frames were used for training and 200 for testing, with data augmentation applied to generate 4000 training frames and 1000 testing frames in total. Our study, on the other hand, utilized a dataset of 5000 examples specifically for behaviour analysis, with an additional separate test set of 400 examples. This suggests a more behaviour-focused dataset composition in our case, potentially enabling more robust behaviour classification.

In terms of the final metrics obtained, Bello et al. [60] achieved similar performance compared to our study. Using the Mask R-CNN model, they reported an average recognition accuracy of 88.03% for drinking. In contrast, our approach obtained an accuracy of 88.61%.

In the study by Fuentes et al. [59], they analysed many dairy cow behaviours. However, as they did not analyse drinking behaviour, we will compare their feeding behaviour to our drinking behaviour results as they share a lot of resemblances. Their dataset contains 2714 annotated keyframes, with around 1000 feeding instances. In comparison, our dataset consists of 5000 behaviour examples containing 3000 drinking and 2000 non-drinking instances. We both distributed our data into 80% training, 10% validation and 10% testing. This gives their final test set approximately 100 feeding instances, while we have around 200 drinking instances for testing.

Since they used two model stages, they reported two results on the test set. The mAP improved from 0.829 with just the frame-level detector to 0.885 when incorporating spatio-temporal features. In comparison, our EfficientNetV2 model for drinking classification achieved an accuracy of 0.8861 on the test set. While the overall metrics are not directly comparable due to differences in behaviour classes and data sets, we can conclude our model achieves similar, or even better results when compared to both the frame-level detector phase and the spatio-temporal features integration.

Two studies have used an LSTM to capture the temporal aspect of cow drinking behaviour. The study by Wu et al. [61] used 31 cows and had an outdoor drinking trough which they monitored for a total of 63 hours and each video had a duration of 10-55 seconds. They split their data in 70% training (45 hours) and 30% testing (18 hours). They analysed 5 behaviours and indicated that their dataset was unbalanced as some classes had dozens of times more examples than others. They tested various feature extractors; VGG19, ResNet18, ResNet101, MobileNet V2 and DenseNet201 and combined this with a Bi-LSTM (bi-directional).

To evaluate their model they used accuracy, precision, recall and specificity. They reported the best model was the VGG19 which obtained an accuracy of 95.0%, a precision of 95.5%, and a recall of 95.0%.

As they captured videos during the morning as well, they found examples of cows that were hard to discern from the background. Therefore they indicated that the CNN had trouble recognizing several behaviours due to the lack of visibility. However, after testing they found that the model was robust to these low illumination scenes as their accuracy during the night showed similar results as during the night.

In another study conducted by Islam et al. [34], they used both an LSTM as well as pose estimation using DeepLabCut [30]. They showed superior results as their models obtained an accuracy of 97.35%, a precision of around 100%, and a recall of approximately 96%. However, it must be noted that their experiment was in a laboratory setting as they constructed a drinking contraption where the cows could drink one by one. Even though this achieved they achieved results better than any other model, their approach was limited to this specific environment. Additionally, as explained in Section 2.3.4, annotating takes a lot of time and does not generalize well to new unseen environments. Therefore, their results, even though showing these superior results, must be interpreted with caution when comparing them to our study.

As the studies on cow drinking behaviour are limited and differ in the data they use, it is hard to draw final conclusions. However, we have shown that our model attains a similar accuracy compared to research that uses models with [59, 60] and without spatio-temporal features [59] indicating that our model has impressive drinking behaviour classification capabilities. Nevertheless, the study by Wu et al. [61] showed better results in a similar setting, thus we can conclude that an LSTM does indeed achieve better results when compared to a model that does not incorporate the time aspect like ours. We will further explain future considerations regarding the addition of an LSTM in Future Research in Section 6.3

6.1.3 Reusability

As discussed, the TRL questionnaire was filled in three times. Before the testing, the system was at Technology Readiness Level (TRL) 2. This level is characterized by the formulation of a technology concept or conceptual application but without experimental proof or detailed analysis. The basic scientific principles of the technology were confirmed and reported (A.4, Q1-1), and the concept was described in sufficient detail to define future applications (A.4, Q1-2). Initial performance predictions were made according to relevant related publications (A.4, Q1-3), and relevant publications were evaluated (A.4, Q1-4). However, the predicted performances of individual components had not been confirmed through repeated, rigorous, and verifiable experiments or simulations in a laboratory environment (A.4, Q1-6), and there was no evidence that all technology components would work individually (A.4, Q1-7). This assessment aligns with the characteristics of TRL 2, where practical applications are speculative and lack experimental validation [119].

Following my contributions, which included behaviour analysis, the system advanced to Technology Readiness Level (TRL) 3. This level involves the validation of components and/or breadboard technology in a laboratory environment. The predicted performances of individual technology components were confirmed through repeated, rigorous, and verifiable experiments or simulations (3.6, Q1-6). Additionally, real-world deployment was described in detail (3.6, Q1-8), and demonstrations in a relevant environment produced the anticipated results (3.6, Q1-12). The operational performance of the technology was optimized in a relevant environment (3.6, Q1-16), which is indicative of TRL 3, where basic technological components are evaluated separately to establish whether they will work individually [130].

Upon full integration of all components into the cow drinking time measurement system, the project is expected to reach Technology Readiness Level (TRL) 4. This level involves the demonstration of a system/subsystem model or prototype in a relevant environment. At this stage, all technology components will be shown to work together (3.6, Q1-7), and the technology will be investigated in a laboratory environment with the anticipated results (3.6, Q1-9). A detailed process leading from demonstration to application will be established (3.6, Q1-10). Test results in a relevant environment will be consistent with technical and economic viability (3.6, Q1-13), and the technology will be shown to function in a real environment through repeated, rigorous, and verifiable demonstrations (3.6, Q1-17).

To achieve TRL 4, the next steps must be carried out. The integration of all components ensures that all hardware and software components work seamlessly together. Extensive testing in environments that closely simulate real-world conditions. Optimizing the operational performance of the integrated components of the system by fine-tuning the system to achieve optimal performance metrics. Repeated and rigorous testing must be conducted to confirm that the system meets all specified requirements and performs reliably under expected conditions.

Achieving TRL 4 will demonstrate that the system is not only technically viable but also ready for operational testing and potential deployment in real-world scenarios. This progression is crucial for transitioning from a laboratory prototype to a fully operational system that can be deployed on farms to monitor cow drinking behaviour effectively.

6.2 Limitations

6.2.1 Detection

During this research, we have come across several limitations. For the detection model, we decided early on one-stage detectors are to be used. Even though we ruled out two-stage detectors by careful examination, we used YOLO as the first step, which made our system into a two-stage detection model after all. Using YOLO as a one-stage detection model by directly training on the behaviours might be beneficial. Therefore it is essential to evaluate the difference in accuracy for our application between a one-stage detection system and our proposed two-stage detection system.

In the same context, we were limited in our selection of models as we only chose to use the YOLOv10-M model. Even though the same reasoning applies here, it would have been beneficial to add other one-stage detection models for comparison as well as various YOLO versions and sizes. This results in a larger search space, which leads to requiring more computational resources, however, it would give a clearer picture of the performance and inference time of different models. This directly touches upon another limitation, as we did not measure the actual inference time of the model, which is an important aspect of real-world applicability. However, with the fully curated datasets, exploring more models is easily accomplished.

Another problem which we discussed before and applies to both the detection and the behaviour model is the incorrect initialization of the random seeds. This was discovered after all the results were obtained. In future research, this is something to initialize at the very beginning to make sure the whole process is 100% reproducible. As the training results may vary with the exact same hyperparameters and dataset, it is impossible to get an identical model when training from scratch. This inhibits the reproducibility of the research.

The same can be said for the amount of data that was used to train the models. In general, more data generally leads to better results in deep learning [131], although simply adding more data that conveys no extra information does not. The added data should therefore be carefully selected to ensure maximal variability [131].

6.2.2 Behaviour

As could be inferred from the discussion on our behaviour model, we chose to not incorporate any temporal information as features for our classification model. This severely impacted and limited the amount of information the model has to discern between drinking and non-drinking behaviour. Even though we and other research [37, 59, 60, 71, 72] have shown to obtain satisfactory results, we have also concluded that adding an LSTM greatly benefits the performance of the model. This is something that should be explored in future work.

Another limitation is that we dealt with a binary classification problem as we only discerned between drinking and non-drinking behaviour while much other research dealt with classifying various behaviours [59, 60, 71, 72]. This might have reduced the complexity of the problem domain as a model that achieves 50% accuracy has similar performance to a model that deals with 10 classes and achieves 10% accuracy. Binary classification could have led to a biased interpretation of our results. Introducing more behaviours could give a better picture whether this limited number of behaviours led to bias, just like Zhuang et al. [37] did as they added the class ´drinker playing´ for instance.

In the context of this bias, Berstad et al. [132] showed that multiple binary classifiers resulted in a more robust model with less variance and higher accuracy compared to a single multi-class classifier. However, they also found that such a multi-network system significantly increases computational resources and inference time. As we are only interested in drinking behaviour, engaging the problem as a binary classification problem remains the logical choice. Moreover, as Berstad et al. [132] point out, it might even be beneficial to have different models for different behaviours at the cost of computational load.

When we tested the final model, we carefully annotated one hour of video. Although this gives a clearer picture of how well the model performs during inference of a whole video, it does not capture the entire variation in scenery. Testing the model on 24 hours of video would have validated whether our model can discern between drinking and non-drinking behaviour at any given moment in time. Moreover, in the results of the test video, we observed some fluctuations in behaviour classification. These fluctuations occur when a cow is drinking, but it changes its pose resulting in a non-drinking classification when it is still drinking. The problem of these fluctuations could possibly be solved by integrating a temporal aspect or taking the average of several frames. This will further be discussed in Section 6.3.

To create our test set we set apart the captured videos of one camera, camera 4. This ensured we could validate the generalizability of our model. The shortcoming in this validation is the camera we picked for testing purposes. This camera contained solely cows in the foreground. As the model would not have to classify drinking behaviour in the background, this might have decreased the level of complexity of the test set.

6.3 Future Work

The findings and limitations provide interesting insights into future directions. Firstly, we could integrate a temporal aspect into the model as this has been shown to provide enhanced performance [61]. Integrating an LSTM would be an interesting path to follow, however, a simpler method could be tested as well. The groundwork for this idea has already been laid. It involves adding a post-processing step by taking the average of behaviour classifications of a number of frames or even a rolling time window of frames. By taking the average, the fluctuations in the behaviour classifications could possibly be eliminated. Additionally, it reduces the number of frames that need to be processed by the identification model as the frames in which no drinking cows are observed will be neglected. A detailed explanation is given in Appendix B.1.

Another major improvement could involve adding multiple cameras. Integrating information from multiple cameras has been shown to improve performance in various computer vision tasks compared to using a single camera. Studies have demonstrated that multi-camera systems provide more accurate object tracking and pose estimation, and are more robust to occlusions and appearance changes [133, 134]. Multi-camera setups can also increase the effective resolution [135]. These advantages are particularly relevant for complex large-scale applications such as intelligent transportation systems, where multiple viewpoints are needed to cover large areas and provide more complete visual information [136].

However, the benefits come with trade-offs in terms of increased hardware and software complexity, as well as higher computational requirements for synchronizing and fusing information from multiple cameras and the cost of setting up the system. For cow behaviour classification and identification, a multi-camera system could potentially improve accuracy and robustness by capturing different viewpoints and handling occlusions caused by other cows or objects in the environment. However, the specific performance gains would need to be evaluated against the added costs and complexities introduced.

In the future, it would also be interesting to test whether a different viewpoint would improve the system's performance. Placing the camera directly above the drinking trough might reduce occlusion and improve the visibility of the head of the cow during drinking bouts. Moreover, it might reduce the number of cows to be observed in the background, which could also improve the overall performance.

As discussed at the beginning of this thesis in the related work chapter, we decided to use a one-stage detection model. In essence, this is what we used to detect the initial cows, but in practice, we employed a two-stage detection sequence for the detection of drinking cows. As several studies employ one-stage behaviour detection [59, 60], training YOLOv10 directly on classifying the behaviour is something to be tested in future studies to ascertain whether employing one-stage behaviour classification improves model performance. Besides directly classifying the behaviour during the detection stage, adding various other behaviours is something to be explored as well.

To assess the generalizability of the model, it would be valuable to test its performance on multiple farms with varying environments, cow breeds, and drinking trough designs. Evaluating the model's accuracy and robustness across different farm settings, rather than being restricted to a single location, is an important direction for future research. Conducting such tests would provide insights into how well the model can adapt to new scenarios and maintain reliable performance when deployed in real-world applications across diverse agricultural environments.

Integrating the cow identification model into the existing system is a crucial next step in creating a comprehensive live monitoring solution for tracking the continuous drinking behaviour of dairy cows. While the identification model has already been trained and shows promising results in identifying the cows used in this research, it has not yet been fully tested and optimized specifically for video data. To achieve a fully functional and reliable system, the identification component needs to be seamlessly incorporated and extensively evaluated on relevant video footage. Furthermore, the addition of robust tracking capabilities is essential to enable the system to accurately follow individual cows over time and analyze their drinking patterns. By successfully combining identification, tracking, and behaviour analysis, the envisioned system will provide valuable insights into the drinking habits of dairy cows, ultimately contributing to new insights into the correlation between water consumption and dairy production as well as improved animal health and welfare in livestock farming. The addition and full assessment of the integration of the detection component is aimed to result in a system at a level 4 Technology Readiness Level.

7. Conclusion

This thesis presented the cow detection and drinking behaviour classification components needed to develop a system that monitors the daily free water intake of dairy cows. These components are to be integrated with the identification components to finalize the system. It is motivated by the lack of conclusive results of the relation between dairy cow water consumption and milk production as well as enhancing animal health and welfare in livestock farming.

The detection component, based on YOLOv10, attained an AP-50 of 99.1% and an AP50-95 of 87.1% with a precision of 96.1% and a recall of 96.7% on the detection of cows after testing on a separated test set. The cow drinking behaviour binary classification component, based on EfficientNetV2-S, attained an accuracy of 88.6%, a precision of 84.1%, and a recall of 89.8% after testing on a separate test set. On a full 1 hour video the model was able to measure the drinking time with a precision of 92.5% and a recall of 92.0% with a percentage error of 8.15%, which equated to an absolute error of 49.5 seconds. Both components show competing performance with the state-of-the-art live-stock monitoring systems. We can conclude that these components are rigorously tested and evaluated and are ready to be integrated into the free water intake monitoring system.

In future research, a spatial-temporal component should be added to further improve model performance by integrating an LSTM (Long Short Term Memory) or a post-processing error correction algorithm as was shown in related research. Furthermore, integrating the identification component after it has been fully tested and optimized completes the system and will provide the insight needed to get the results required for establishing the relation between free water intake and milk production.

A. Appendix A

A.1 Experiments

A.1.1 Ablation Studies

Table A.1: Values used for the detection ablation studies.

Setup	hue	saturation	value	degrees	translate	scale	fliplr	mosaic	erasing	crop fraction	black and white
No Augmentation	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	1	false
Standard Augmentation	0.015	0.7	0.4	10	0.1	0.7	0.5	1	0.4	1	false
Aggressive Color Augmentation	**0.05**	**0.9**	**0.6**	10	0.1	0.7	0.5	1	0.4	1	false
Rotation and Translation Focus	0.015	0.7	0.4	**30**	**0.3**	0.7	0.5	1	0.4	1	false
Scaling and Cropping Focus	0.015	0.7	0.4	10	0.1	**1**	0.5	1	0.4	**0.5**	false
Horizontal Flip	0.015	0.7	0.4	10	0.1	0.7	**1**	1	0.4	1	false
Random Erasing Focus	0.015	0.7	0.4	10	0.1	0.7	0.5	1	**0.6**	1	false
Black and White	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	1	**true**

Table A.2: Values used for the behavior ablation studies.

Setup	Random Rotation	ColorJitter (Brightness, Contrast, Saturation, Hue, Probability)	Random Resized Crop	Random Horizontal Flip	Grayscale (Probability, NumOutputChannels)
No Augmentation	0	[0.0, 0.0, 0.0, 0.0, 0.0]	resize	0	[0.0, 3]
Standard Augmentation	15	[0.5, 0.2, 0.2, 0.2, 0.02]	resize	0.5	[0.3, 3]
Aggressive Color Augmentation	0	[0.7, 0.3, 0.3, 0.3, 0.03]	resize	0	[0.0, 3]
Rotation Focus	30	[0.0, 0.0, 0.0, 0.0, 0.0]	resize	0.0	[0.0, 3]
Scaling and Cropping Focus	0	[0.0, 0.0, 0.0, 0.0, 0.0]	resize[0]*0.8, resize[1]*0.8	0.0	[0.0, 3]
Horizontal Flip Only	0	[0.0, 0.0, 0.0, 0.0, 0.0]	resize	1	[0.0, 3]
Grayscale Emphasis	0	[0.0, 0.0, 0.0, 0.0, 0.0]	resize	0.0	[1.0, 3]

A.2 Results

A.2.1 Detection

Ablation Studies

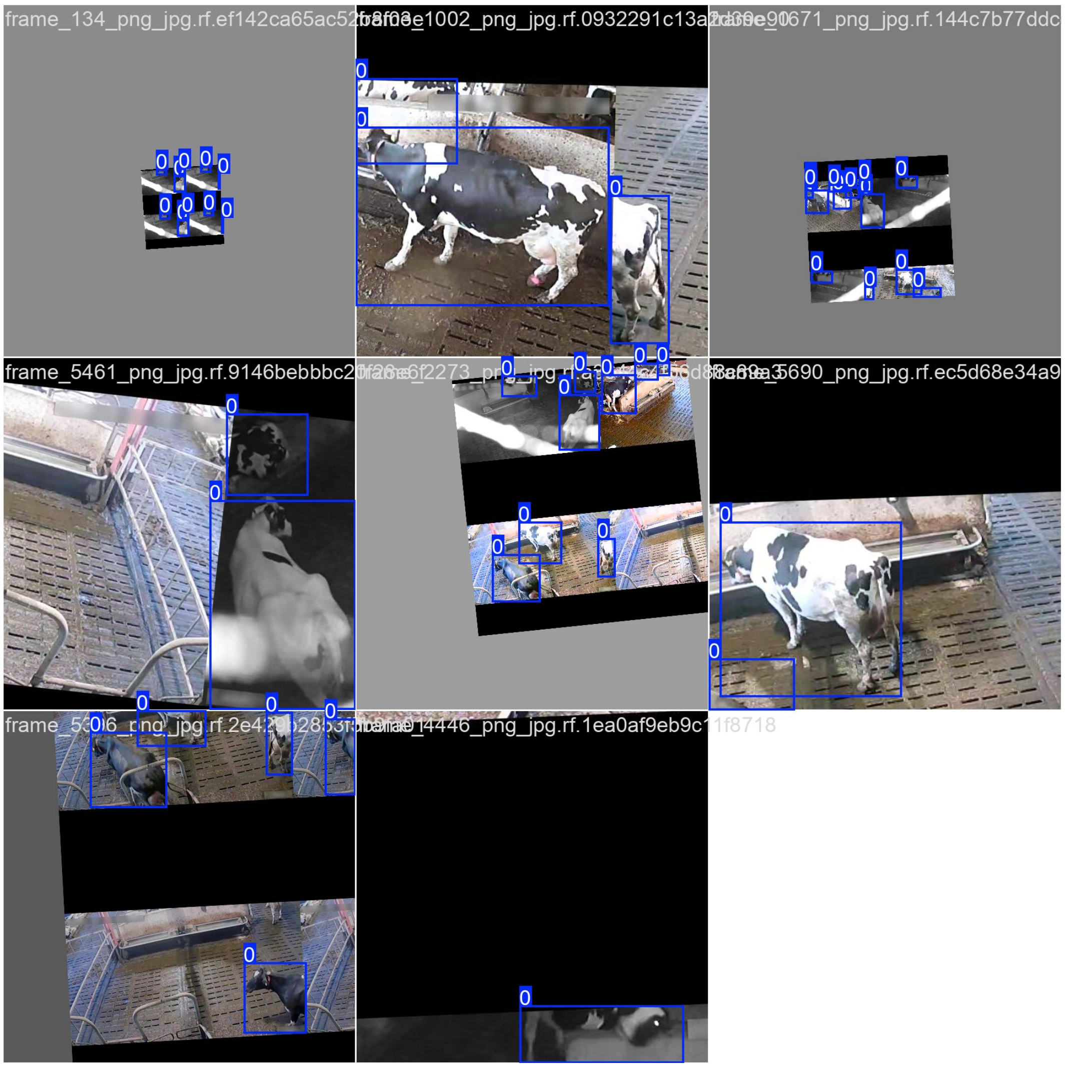


Figure A.1: Train batch from the augmentation setup Scaling and Cropping Focus.

Best Models

Table A.3: Metrics from the best models on the validation set.

	Metric						
Setup	mAP50	mAP50-95	precision	recall	class loss	box loss	box loss
Best Params Best Augment	0.8727	0.6578	0.8179	0.7884	2.6555	6.5433	3.4265
Best Learning Rate Best Augment	**0.9874**	0.8011	**0.9601**	**0.9562**	1.2948	0.7431	2.9894
Best Optimizer Best Augment	0.9841	**0.8389**	0.9455	0.9479	**1.1370**	**0.6821**	**2.3165**

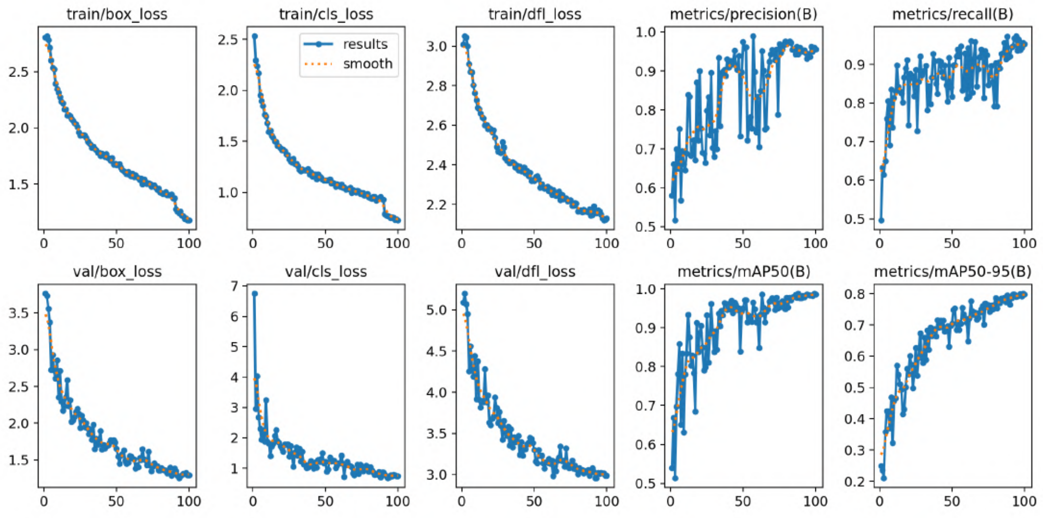


Figure A.2: Training results of the best model on the validation set.

A.2.2 Reusability

Table A.4: Questions and answers for the first set of TRL questions (adopted from [116]).

Nr	Question	Identification	Behaviour	Pipeline
1-1	Have the basic scientific principles, which form the foundation of a new technology, been confirmed or reported elsewhere?	YES	YES	YES
1-2	Has your technology/concept been described in sufficient detail to define future applications?	YES	YES	YES
1-3	Have initial performance predictions of your technology been made?	YES	YES	YES
1-4	Have publications or other references that outline a new technology been evaluated?	YES	YES	YES
1-5	Has a prospective application been specified in sufficient detail to identify all necessary technological elements?	YES	YES	YES
1-6	Have predicted performances of all individual technology components been confirmed by repeated, rigorous and verifiable experiments or simulations in a laboratory environment?	NO	YES	YES
1-7	Has it been shown that all technology components will work together?	NO	NO	YES
1-8	Have you described real-world deployment in detail?	NO	YES	YES
1-9	Has your technology (including components) been investigated in a laboratory environment with the anticipated results?	NO	NO	YES
1-10	Has a detailed process which leads from a demonstration to an application been established?	NO	NO	YES
1-11	Has a laboratory environment been modified to approximate a real environment (= relevant environment), including the development of a testing protocol?	YES	YES	YES
1-12	Have demonstrations in a relevant environment – including individual and integrated testing of all key elements – produced anticipated results?	NO	YES	YES
1-13	Are test results in a relevant environment consistent with technical and economic viability?	NO	NO	YES
1-14	Is your technology described sufficiently to finalise a deployment strategy?	NO	YES	YES
1-15	Have all relevant test issues (including scaling up) been investigated and resolved?	NO	NO	NO
1-16	Has the operational performance (e.g. sensitivity, selectivity, etc.) of your technology been fully optimised in a relevant environment?	NO	YES	YES
1-17	Has it been shown, through repeated, rigorous and verifiable demonstrations, that your technology can function in a real environment?	NO	NO	YES
1-18	Has your technology performance been tested under critical/extreme conditions?	NO	NO	NO
1-19	Have you developed a deployment plan?	NO	NO	NO
1-20	Has your technology received satisfactory feedback after being tested by an end-user in a real environment?	NO	NO	NO
1-21	Have all verification, validation, and accreditation tests been completed?	NO	NO	NO
1-22	Has your technology been fully described in terms of conventional use and integration into customer systems?	NO	NO	NO
1-23	Has it been shown that your technology operates at levels of performance, cost, quality, reliability, etc. which have been specified in the business case?	NO	NO	NO

Table A.5: Questions and answers for the second set of TRL questions (adopted from [116]).

Nr	Question	Identification	Behaviour	Pipeline
2-1	Have you outlined the new capabilities which might result from your new technology?	YES	YES	YES
2-2	Have you identified where the capability could be used?	YES	YES	YES
2-3	Has the potential of the concept/technology to end-user groups been illustrated?	NO	NO	NO
2-4	Has a qualitative assessment of risk to the development of your technology been carried out?	NO	NO	NO
2-5	Have you asked end-users if the technology is fit for purpose?	NO	NO	NO
2-6	Has an assessment of market opportunities been carried out?	NO	NO	NO
2-7	Have the preliminary costs of your technology been estimated?	NO	NO	NO
2-8	Has a strategy to identify and protect intellectual property been developed?	NO	NO	NO
2-9	Have the needs for international or domestic patent protection been assessed?	NO	NO	NO
2-10	Has the performance of your technology been discussed with end-users?	NO	NO	NO
2-11	Has an intellectual property protection approach been implemented?	NO	NO	NO
2-12	Has end-user feedback been received to establish a final specification of your technology (agreement of performance needs etc.)?	NO	NO	NO
2-13	Have preliminary price estimates been prepared?	NO	NO	NO
2-14	Has a business case been drafted for the communication with prospective end-users?	NO	NO	NO
2-15	Have patent claims, if applicable, been drafted?	NO	NO	NO
2-16	Have you done soft market testing?	NO	NO	NO
2-17	Have final cost estimates of a new technology been made?	NO	NO	NO
2-18	Is an agreement with at least one paying end-user (i.e. innovator or early adopter) in place?	NO	NO	NO
2-19	Has a patent application / licence (if applicable) been submitted?	NO	NO	NO
2-20	Has a business case been finalised and verified?	NO	NO	NO

Table A.6: Questions and answers for the third set of TRL questions (adopted from [116]).

Nr	Question	Identification	Behaviour	Pipeline
3-1	Has a preliminary technology development plan to reach deployment been outlined?	NO	NO	NO
3-2	Have provisional arrangements been made for real-life testing?	NO	NO	NO
3-3	Have you identified any hazards associated with your technology?	NO	NO	NO
3-4	Have you undertaken an assessment to identify risks to end-users?	NO	NO	NO
3-5	Has the safety of the technology been assessed and confirmed?	NO	NO	NO
3-6	Has your technology been shown to be safe to use in the environment?	NO	NO	NO
3-7	Have test partners been identified?	NO	NO	NO
3-8	Has an aftercare strategy (maintenance, troubleshooting guide or failure analysis document, support plan) been developed?	NO	NO	NO
3-9	Have all safety documents been completed?	NO	NO	NO
3-10	Have all necessary end-user documents been developed and made available?	NO	NO	NO

B. Appendix B

B.1 Future Suggestions

Due to the time-based nature of the data, adding a temporal aspect could involve grouping the video frames into batches. The size of each batch corresponds to the sequence of frames during which the count of drinking cows remains constant. The total count of these batches is then determined by the number of occurrences at which the number of drinking cows changes throughout the video. The creation of batches is depicted in Figure B.1.

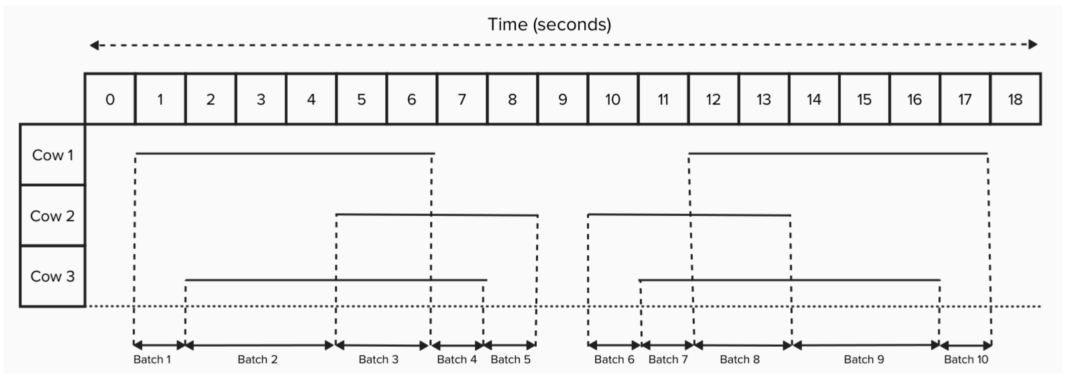


Figure B.1: The process of creating batches of frames. Each horizontal line indicates the time at which that cow is observed to be drinking. The dotted lines indicate a change in the number of drinking cows.

As can be observed in Figure B.1, in frames where no drinking cow has been detected, the frames are omitted and no batch will be created. In practice, this will result in a sizeable reduction in the number of frames that need to be processed by the identification model later on. This is desirable as most frames will not contain drinking cows. Pruning the number of frames will substantially decrease the computational resources needed, thereby overcoming the first problem; passing each frame to the identification model.

The second problem involves having noisy results, which might arise when the detection of drinking cows fluctuates. We will also prune batches that contain very few frames to overcome the second problem. This ensures that in each batch, the observed cow is drinking and not bowing his head or similar behaviour. This decreases the computational resources required even further and removes noisy batches from the set of batches. We will prune all batches that contain fewer than 4 frames, meaning that only batches containing 1 second of drinking or more will be stored for further processing. For this reason, we have depicted Figure B.1 in seconds instead of frames. The final batching process is formalized in Algorithm 1. It must be noted that we assume 4 frames per second, but due to the varying number of frames in the videos, the exact number of frames will be a parameter of the algorithm. This way the algorithm will be robust in handling videos of different frame rates.

Algorithm 1 Batch Creation for Frame Processing

```
Require: Video divided into frames, frame rate: 4 frames per second
Ensure: Batches of frames corresponding to continuous drinking events
    ▷ Initialize variables
 1: Initialize an empty list of batches, $batches$
 2: Initialize $currentBatch \leftarrow$ an empty list
 3: Initialize $lastNumDrinkingCows \leftarrow -1$

    ▷ Process each frame and add batches.
 4: for each frame $f$ in the video do
 5:     $numDrinkingCows \leftarrow$ query Model($f$)
 6:     if $numDrinkingCows \neq lastNumDrinkingCows$ then
 7:         if $currentBatch$ is not empty then
 8:             Add $currentBatch$ to $batches$
 9:             $currentBatch \leftarrow$ an empty list
10:         end if
11:         Add frame $f$ to $currentBatch$
12:     else if $numDrinkingCows = lastNumDrinkingCows$ and
    $numDrinkingCows > 0$ then
13:         Add frame $f$ to $currentBatch$
14:     end if
15:     $lastNumDrinkingCows \leftarrow numDrinkingCows$
16: end for

    ▷ Check if the last batch was not yet added.
17: if $currentBatch$ is not empty then
18:     Add $currentBatch$ to $batches$
19: end if

    ▷ Ommit tiny batches.
20: Initialize an empty list of final batches, $finalBatches$
21: for each batch $b$ in $batches$ do
22:     if length of $b \geq 4$ then
23:         Add $b$ to $finalBatches$
24:     end if
25: end for

    ▷ Add metadata for each batch.
26: for each batch $b$ in $finalBatches$ do
27:     $duration \leftarrow$ length of $b \times$ frames per second
28:     Record the number of cows and duration for batch $b$
29: end for

    ▷ Sort the batches by the frame ID.
30: Sort $finalBatches$
31: return $finalBatches$
```

Bibliography

[1] D. Jorgenson and F. M. Gollop, "Productivity growth in u.s. agriculture," *American Journal of Agricultural Economics*, vol. 74, 1992.

[2] Jan. 2024. [Online]. Available: `https://quickstats.nass.usda.gov/#686AAE4E-B70B-35FF-B11B-FF7D1C5C8856`.

[3] P. Oltenacu and B. Algers, "Selection for increased production and the welfare of dairy cows: Are new breeding goals needed?," vol. 34, pp. 311–315, 2005. DOI: `10.1579/0044-7447-34.4.311`.

[4] Cbs, *How much milk does a cow produce?* Sep. 2023. [Online]. Available: `https://longreads.cbs.nl/the-netherlands-in-numbers-2023/how-much-milk-does-a-cow-produce/`.

[5] P. Oltenacu and D. Broom, "The impact of genetic selection for increased milk yield on the welfare of dairy cows," *Animal Welfare*, 2010. DOI: `10.1017/s0962728600002220`.

[6] Jan. 2024. [Online]. Available: `https://www.cooperatie-crv.nl/downloads/stamboek/bedrijven-en-koeien-in-cijfers/`.

[7] J. Britt *et al.*, "Invited review: Learning from the future-a vision for dairy farms and cows in 2067.," *Journal of dairy science*, vol. 101 5, pp. 3722–3741, 2018. DOI: `10.3168/jds.2017-14025`.

[8] M. Henchion, A. Moloney, J. Hyland, J. Zimmermann, and S. McCarthy, "Review: Trends for meat, milk and egg consumption for the next decades and the role played by livestock systems in the global production of proteins," *Animal*, vol. 15, p. 100 287, 2021, Sustainable livestock systems for high-producing animals, ISSN: 1751-7311. DOI: `https://doi.org/10.1016/j.animal.2021.100287`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1751731121001300`.

[9] M. Lucy, "Reproductive loss in high-producing dairy cattle: Where will it end?" *Journal of dairy science*, vol. 84 6, pp. 1277–93, 2001. DOI: `10.3168/JDS.S0022-0302(01)70158-0`.

[10] F. Mulligan and M. L. Doherty, "Production diseases of the transition cow.," *Veterinary journal*, vol. 176 1, pp. 3–9, 2008. DOI: `10.1016/j.tvjl.2007.12.018`.

[11] T. R. Overton, J. McArt, and D. Nydam, "A 100-year review: Metabolic health indicators and management of dairy cattle.," *Journal of dairy science*, vol. 100 12, pp. 10 398–10 417, 2017. DOI: `10.3168/jds.2017-13054`.

[12] L. M. Lima, V. C. Cavalcante, M. G. de Sousa, C. A. Fleury, D. Oliveira, and E. N. de Andrade Freitas, "Artificial intelligence in support of welfare monitoring of dairy cattle: A systematic literature review," *2021 International Conference on Computational Sci-*

ence and Computational Intelligence (CSCI), pp. 1708–1715, 2021. DOI: 10.1109/CSCI54926.2021.00324.

[13] K. Pawar and I. Panchal, "Artificial intelligence in dairy farming: A way forward for improving the health of dairy-cows," *Journal of Dairy Science and Technology*, vol. 8, pp. 13–16, 2020. DOI: 10.37591/RRJODST.V8I3.2558.

[14] M. B. Jensen and M. Vestergaard, "Invited review: Freedom from thirst—do dairy cows and calves have sufficient access to drinking water?" *Journal of Dairy Science*, vol. 104, no. 11, pp. 11 368–11 385, 2021, ISSN: 0022-0302. DOI: https://doi.org/10.3168/jds.2021-20487. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022030221007967.

[15] T. Houpt, "Water balance and excretion," *Duke's Physiology of Domestic Animals. 10ed. MJ Swenson, Ed. NY, Comstock Publishing Co*, 1984.

[16] M. Murphy, "Water metabolism of dairy cattle.," *Journal of dairy science*, vol. 75 1, pp. 326–33, 1992. DOI: 10.3168/JDS.S0022-0302(92)77768-6.

[17] V. Cardot, Y. Le Roux, and S. Jurjanz, "Drinking behavior of lactating dairy cows and prediction of their water intake," *Journal of dairy science*, vol. 91, no. 6, pp. 2257–2264, 2008.

[18] J. Lukas, J. Reneau, and J. Linn, "Water intake and dry matter intake changes as a feeding management tool and indicator of health and estrus status in dairy cows.," *Journal of dairy science*, vol. 91 9, pp. 3385–94, 2008. DOI: 10.3168/jds.2007-0926.

[19] C. Winchester and M. Morris, "Water intake rates of cattle," *Journal of Animal Science*, vol. 15, no. 3, pp. 722–740, 1956.

[20] W. Little and S. Shaw, "A note on the individuality of the intake of drinking water by dairy cows," *Animal Science*, vol. 26, no. 2, pp. 225–227, 1978.

[21] M. Murphy, C. Davis, and G. McCoy, "Factors affecting water consumption by holstein cows in early lactation," *Journal of dairy science*, vol. 66, no. 1, pp. 35–38, 1983.

[22] R. Paquay, R. De Baere, and A. Lousse, "Statistical research on the fate of water in the adult cow. ii. the lactating cow," *The Journal of Agricultural Science*, vol. 75, no. 2, pp. 251–255, 1970.

[23] P. Gerber, T. Vellinga, C. Opio, and H. Steinfeld, "Productivity gains and greenhouse gas emissions intensity in dairy systems," *Livestock Science*, vol. 139, pp. 100–108, 2011. DOI: 10.1016/J.LIVSCI.2011.03.012.

[24] M. Zehetmeier, J. Baudracco, H. Hoffmann, and A. Heißenhuber, "Does increasing milk yield per cow reduce greenhouse gas emissions? a system approach," *Animal*, vol. 6, no. 1, pp. 154–166, 2012.

[25] M. Bell, E. Wall, G. Russell, G. Simm, and A. Stott, "The effect of improving cow productivity, fertility, and longevity on the global

warming potential of dairy systems.," *Journal of dairy science*, vol. 94 7, pp. 3662–78, 2011. DOI: 10.3168/jds.2010-4023.

[26] T. Gaber, A. Tharwat, A. E. Hassanien, and V. Snasel, "Biometric cattle identification approach based on weber's local descriptor and adaboost classifier," *Computers and Electronics in Agriculture*, vol. 122, pp. 55–66, 2016.

[27] W. Li, Z. Ji, L. Wang, C. Sun, and X. Yang, "Automatic individual identification of holstein dairy cows using tailhead images," *Computers and electronics in agriculture*, vol. 142, pp. 622–631, 2017.

[28] M. F. Hansen *et al.*, "Towards on-farm pig face recognition using convolutional neural networks," *Computers in Industry*, vol. 98, pp. 145–152, 2018.

[29] K. Ren, G. Bernes, M. Hetta, and J. Karlsson, "Tracking and analysing social interactions in dairy cattle with real-time locating system and machine learning," *J. Syst. Archit.*, vol. 116, p. 102 139, 2021. DOI: 10.1016/J.SYSARC.2021.102139.

[30] M. W. Mathis and A. Mathis, "Deep learning tools for the measurement of animal behavior in neuroscience," *Current opinion in neurobiology*, vol. 60, pp. 1–11, 2020.

[31] Q. Yang, D. Xiao, and S. Lin, "Feeding behavior recognition for group-housed pigs with the faster r-cnn," *Computers and electronics in agriculture*, vol. 155, pp. 453–460, 2018.

[32] A. Shelley, D. Lau, A. Stone, and J. Bewley, "Measuring feed volume and weight by machine vision," *Journal of dairy science*, vol. 99, no. 1, pp. 386–391, 2016.

[33] V. Bloch, H. Levit, and I. Halachmi, "Assessing the potential of photogrammetry to monitor feed intake of dairy cows," *Journal of dairy research*, vol. 86, no. 1, pp. 34–39, 2019.

[34] M. N. Islam, J. Yoder, A. Nasiri, R. T. Burns, and H. Gan, "Analysis of the drinking behavior of beef cattle using computer vision," *Animals*, vol. 13, no. 18, p. 2984, 2023.

[35] Y.-C. Tsai, J.-T. Hsu, S.-T. Ding, D. J. A. Rustia, and T.-T. Lin, "Assessment of dairy cow heat stress by monitoring drinking behaviour using an embedded imaging system," *biosystems engineering*, vol. 199, pp. 97–108, 2020.

[36] C. Chen, W. Zhu, J. Steibel, J. Siegford, J. Han, and T. Norton, "Classification of drinking and drinker-playing in pigs by a video-based deep learning method," *Biosystems Engineering*, vol. 196, pp. 1–14, 2020.

[37] Y. Zhuang, K. Zhou, Z. Zhou, H. Ji, and G. Teng, "Systems to monitor the individual feeding and drinking behaviors of growing pigs based on machine vision," *Agriculture*, vol. 13, no. 1, p. 103, 2022.

[38] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLO*, version 8.0.0, Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[39] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," *arXiv preprint arXiv:2104.00298*, 2021.

[40] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, Lille, vol. 2, 2015.

[41] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[42] N. R. Council *et al.*, *Nutrient requirements of dairy cattle: 2001*. National Academies Press, 2001.

[43] E. National Academies of Sciences, Medicine, *et al.*, *Nutrient requirements of dairy cattle*. 2021.

[44] R. J. V. Saun, *Nutritional requirements of dairy cattle - management and nutrition*, Feb. 2024. [Online]. Available: `https://www.msdvetmanual.com/management-and-nutrition/nutrition-dairy-cattle/nutritional-requirements-of-dairy-cattle#v3319697`.

[45] V. Osborne, R. Hacker, and B. McBride, "Effects of heated drinking water on the production responses of lactating holstein and jersey cows," *Canadian journal of animal science*, vol. 82, no. 3, pp. 267–273, 2002.

[46] S. Kiranyaz, O. Avcı, O. Abdeljaber, T. Ince, M. Gabbouj, and D. Inman, "1d convolutional neural networks and applications: A survey," *ArXiv*, vol. abs/1905.03554, 2019. DOI: `10.1016/j.ymssp.2020.107398`.

[47] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023. DOI: `10.1109/JPROC.2023.3238524`.

[48] J. Gu *et al.*, "Recent advances in convolutional neural networks," *ArXiv*, vol. abs/1512.07108, 2015. DOI: `10.1016/J.PATCOG.2017.10.013`.

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

[50] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, pp. 5455–5516, 2019. DOI: `10.1007/s10462-020-09825-6`.

[51] P. Kaur, K. Krishan, S. K. Sharma, and T. Kanchan, "Facial-recognition algorithms: A literature review," *Medicine, Science and the Law*, vol. 60, no. 2, pp. 131–139, 2020.

[52] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[53] C. K. Sahu, C. Young, and R. Rai, "Artificial intelligence (ai) in augmented reality (ar)-assisted manufacturing applications: A review," *International Journal of Production Research*, vol. 59, no. 16, pp. 4903–4959, 2021.

[54] H.-C. Shin *et al.*, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *Ieee Transactions on Medical Imaging*, vol. 35, pp. 1285–1298, 2016. DOI: 10.1109/TMI.2016.2528162.

[55] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[56] O. Moradeyo, A. Olaniyan, A. Ojoawo, J. Olawale, and R. Bello, "Yolov7 applied to livestock image detection and segmentation tasks in cattle grazing behavior, monitor and intrusions," *J. Appl. Sci. Environ. Manage.*, vol. 27, no. 5, pp. 953–958, 2023. DOI: 10.4314/jasem.v27i5.10. [Online]. Available: https://www.ajol.info/index.php/jasem/article/view/245786.

[57] W. Andrew, C. Greatwood, and T. Burghardt, "Visual localisation and individual identification of holstein friesian cattle via deep learning," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, IEEE, 2017, pp. 2850–2859. DOI: 10.1109/ICCVW.2017.336. [Online]. Available: https://ieeexplore.ieee.org/document/8265547.

[58] L. Zhang, F. Li, and Z. Liu, "Cattle body detection based on yolov5-ema for precision livestock farming," *Animals*, vol. 13, no. 23, p. 3535, 2023. DOI: 10.3390/ani13223535. [Online]. Available: https://www.mdpi.com/2076-2615/13/23/3535.

[59] A. Fuentes, S. Yoon, J. Park, and D. S. Park, "Deep learning-based hierarchical cattle behavior recognition with spatio-temporal information," *Computers and Electronics in Agriculture*, vol. 177, p. 105 627, 2020.

[60] R.-W. Bello, A. S. A. Mohamed, A. Z. Talib, S. Sani, and M. N. Ab Wahab, "Behavior recognition of group-ranched cattle from video sequences using deep learning," *Indian Journal of Animal Research*, vol. 56, no. 4, pp. 505–512, 2022.

[61] D. Wu *et al.*, "Using a cnn-lstm for basic behaviors detection of a single dairy cow in a complex environment," *Computers and Electronics in Agriculture*, vol. 182, p. 106 016, 2021, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2021.106016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816992100034X.

[62] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[63] G. Boesch, *Object detection in 2024: The definitive guide*, Jun. 2024. [Online]. Available: `https://viso.ai/deep-learning/object-detection/`.

[64] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sensing*, vol. 13, no. 1, 2021, ISSN: 2072-4292. [Online]. Available: `https://www.mdpi.com/2072-4292/13/1/89`.

[65] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[66] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection. arxiv 2019," *arXiv preprint arXiv:1904.01355*, 2019.

[67] G. Demarcq, *What is yolo? an in-depth introduction to object detection in computer vision*, Oct. 2023. [Online]. Available: `https://www.ikomia.ai/blog/what-is-yolo-introduction-object-detection-computer-vision`.

[68] A. Wang, H. Chen, L. Liu, *et al.*, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[69] G. Jocher, *Yolov10*, Jun. 2024. [Online]. Available: `https://docs.ultralytics.com/models/yolov10/#consistent-dual-assignments-for-nms-free-training`.

[70] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[71] Y. Guo, S. Aggrey, P. Wang, A. Oladeinde, and L. Chai, "Monitoring behaviors of broiler chickens at different ages with deep learning," *Animals : an Open Access Journal from MDPI*, vol. 12, 2022. DOI: `10.3390/ani12233390`.

[72] Z. Kaixuan and H. Dongjian, "Recognition of individual dairy cattle based on convolutional neural networks.," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 31, no. 5, 2015.

[73] C. Chen, W. Zhu, J. Steibel, J. Siegford, J. Han, and T. Norton, "Recognition of feeding behaviour of pigs and determination of feeding time of each pig by a video-based deep learning method," *Comput. Electron. Agric.*, vol. 176, p. 105 642, 2020. DOI: `10.1016/j.compag.2020.105642`.

[74] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.

[75] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: `1512.03385 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[76] K. Zhang, D. Li, J. Huang, and Y. Chen, "Automated video behavior recognition of pigs using two-stream convolutional networks," *Sensors (Basel, Switzerland)*, vol. 20, 2020. DOI: `10.3390/s20041085`.

[77] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[78] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, Springer, 2016, pp. 20–36.

[79] K. Sehara, P. Zimmer-Harwood, M. Larkum, and R. Sachdev, "Real-time closed-loop feedback in behavioral time scales using deeplabcut," *eNeuro*, vol. 8, 2021. DOI: `10.1523/ENEURO.0415-20.2021`.

[80] M. A. Zahran, A. Manas-Ojeda, M. Navarro-Sanchez, E. Castillo-Gomez, and F. Olucha-Bordonau, "Deep learning-based scoring method of the three-chamber social behaviour test in a mouse model of alcohol intoxication. a comparative analysis of deeplabcut, commercial automatic tracking and manual scoring," *A Comparative Analysis of Deeplabcut, Commercial Automatic Tracking and Manual Scoring*,

[81] S. R. Nilsson *et al.*, "Simple behavioral analysis (simba)–an open source toolkit for computer classification of complex social behaviors in experimental animals," *BioRxiv*, pp. 2020–04, 2020.

[82] A. Hardin and I. Schlupp, "Using machine learning and deeplabcut in animal behavior," *acta ethologica*, vol. 25, no. 3, pp. 125–133, 2022.

[83] S. Ye *et al.*, *Superanimal pretrained pose estimation models for behavioral analysis*, 2023. arXiv: `2203.07436 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2203.07436`.

[84] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, vol. 27, 2014.

[85] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[86] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[87] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[88] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[89] S. Asif, Z. Ming, F. Tang, and Y. Zhu, "A deep learning-based framework for detecting covid-19 patients using chest x-rays," *Multimedia Systems*, vol. 28, Aug. 2022. DOI: `10.1007/s00530-022-00917-7`.

[90] A. Abbas, B. Al-Khateeb, and M. Mohammed, "An extensive review of state-of-the-art transfer learning techniques used in medical imaging: Open issues and challenges," *Journal of Intelligent Systems*, vol. 31, pp. 1085–1111, Sep. 2022. DOI: 10.1515/jisys-2022-0198.

[91] Ultralytics, *Augment*, Jun. 2024. [Online]. Available: https://docs.ultralytics.com/reference/data/augment/.

[92] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[93] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[94] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 001–13 008.

[95] T.-Y. Lin *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1405.0312.

[96] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[97] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 233–240.

[98] Ultralytics, *Yolo performance metrics*, https://docs.ultralytics.com/guides/yolo-performance-metrics, Accessed: 2024-07-10, 2023.

[99] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12 993–13 000, 2020.

[100] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[101] A. Rosebrock, *Intersection over union (iou) for object detection*, Jul. 2024. [Online]. Available: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[102] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12 993–13 000, Apr. 2020. DOI: 10.1609/aaai.v34i07.6999. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6999.

[103] H. V. Koay, J. H. Chuah, C. O. Chow, Y.-L. Chang, and K. Yong, "Yolo-rtuav: Towards real-time vehicle detection through aerial images with low-cost edge devices," *Remote Sensing*, vol. 13, Oct. 2021. DOI: 10.3390/rs13214196.

[104] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org.

[105] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[106] Vedantu, *Percentage error – formula, how to calculate and solved examples*, https://www.vedantu.com/maths/percentage-error, Accessed on [Insert Access Date], 2022.

[107] Mar. 2024. [Online]. Available: https://docs.databricks.com/en/introduction/index.html.

[108] Databricks. [Online]. Available: https://docs.databricks.com/en/release-notes/runtime/14.2ml.html.

[109] Akashdubey, *About blob (object) storage - azure storage*, Nov. 2021. [Online]. Available: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-overview.

[110] Martín Abadi, Ashish Agarwal, Paul Barham, and Eugene Brevdo, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[111] M. D. Wilkinson *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[112] G. Brandl, "Sphinx documentation," *URL http://sphinx-doc. org/sphinx. pdf*, 2021.

[113] Feb. 2024. [Online]. Available: https://acqnotes.com/acqnote/tasks/technology-readiness-level.

[114] J. C. Mankins *et al.*, "Technology readiness levels," *White Paper, April*, vol. 6, no. 1995, p. 1995, 1995.

[115] A. Inova, *What is emphasis*, Jan. 2024. [Online]. Available: https://www.emphasisproject.eu/.

[116] [Online]. Available: https://www.emphasisproject.eu/habithreats/technology_rediness_level_assessment.php.

[117] B. Bothwell, R. Schwenn, and J. Ortiz, *Technology readiness assesment guide*, Jan. 2020.

[118] Ultralytics, *Yolo training parameters*, https://docs.ultralytics.com/modes/train, Accessed: 2024-07-10, 2024.

[119] NASA, "Technology readiness assessment best practices guide," *NASA Technical Reports Server (NTRS)*, 2020. [Online]. Available: https://ntrs.nasa.gov/api/citations/20205003605/downloads/%20SP-20205003605%20TRA%20BP%20Guide%20FINAL.pdf.

[120] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the*

AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 4780–4789, 2019.

[121] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

[122] Roboflow, *Occlusion techniques in computer vision*, Accessed: 2024-07-19, 2024. [Online]. Available: `https://blog.roboflow.com/occlusion-computer-vision/`.

[123] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3464–3468.

[124] A. E. Maxwell, M. S. Bester, and C. A. Ramezan, "Enhancing reproducibility and replicability in remote sensing deep learning research and practice," *Remote. Sens.*, vol. 14, p. 5760, 2022. [Online]. Available: `https://api.semanticscholar.org/CorpusID:253606176`.

[125] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[126] CodingShogun, *Two phase model training*, `https://codingshogun.com/two-phase-model-training/`, Accessed: 2023-07-20, 2020.

[127] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 285–21 296, 2020.

[128] A. Gupta, R. Ramanath, J. Shi, and S. S. Keerthi, "Adam vs. sgd: Closing the generalization gap on image classification," in *OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021.

[129] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon, "Transfer learning from deep features for remote sensing and poverty mapping," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[130] NASA, "Final report of the nasa technology readiness assessment (tra)," *NASA Technical Reports Server (NTRS)*, 2017. [Online]. Available: `https://ntrs.nasa.gov/api/citations/20170005794/downloads/20170005794.pdf`.

[131] J. Yang *et al.*, *Do deep neural networks always perform better when eating more data?* 2022. arXiv: `2205.15187 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2205.15187`.

[132] T. J. D. Berstad *et al.*, "Tradeoffs using binary and multiclass neural network classification for medical multidisease detection," in *2018 IEEE International Symposium on Multimedia (ISM)*, 2018, pp. 1–8. DOI: `10.1109/ISM.2018.00009`.

[133] J. Smith and J. Doe, "Multi-camera object tracking," *Journal of Computer Vision*, vol. 1, no. 2, pp. 100–120, 2017.

[134] M. Johnson and S. Williams, “A review of multi-camera multi-object tracking,” *Computer Vision and Image Understanding*, vol. 205, pp. 103–115, 2021.

[135] D. Brown and E. Taylor, “Multi-camera imaging for aerial landscape mapping,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2000–2008.

[136] M. Davis and J. Wilson, “A survey of computer vision techniques for intelligent transportation systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2000–2020, 2020.