The University of Sussex
School of Engineering & Informatics

Masters Dissertation

Programme: MSc in Artificial Intelligence and Adaptive Systems

Candidate number: 276201

Title of Dissertation: Automated Approach to Processing Camera Trap Data Using Machine Learning (VGG16 and YOLOv8 Model Implementation)

Approximate number of words: 7670

Supervisor(s): Dr Ivor Simpson

Date submitted 10/09/2024	**Time submitted** 15:50

Automated Approach to Processing Camera Trap Data Using Machine Learning

VGG16 and YOLOv8 Model Implementation

Candidate Number: 276201

Supervisor: Dr Ivor Simpson

Artificial Intelligence & Adaptive Systems MSc

Department of Informatics

2024

UNIVERSITY OF SUSSEX

Acknowledgements

I am deeply thankful to my supervisor, Dr Ivor Simpons, for the invaluable guidance, support and encouragement throughout this project. His expertise and insights have been instrumental in shaping this thesis.

Abstract

Ecological studies of animals provide valuable insights into biodiversity and the functioning of ecosystems. By tracking animal populations over time, researchers can find how they are affected by environmental changes, such as climate change, pollution, and more impact wildlife. However, manual data collection and analysis can be tedious and prone to human error. We developed VGG16 and YOLOv8 models to automate the animal classification and counting process. Additionally, we integrated SlowFast with YOLOv8 for action recognition. Grad-CAM++ was employed to visualise and understand the predictions made by the VGG16 model. DeepSORT was utilised to enhance tracking and counting for YOLOv8. The YOLOv8 outperformed VGG16 in detecting and counting animals but faced challenges with small-sized animals and poor lighting conditions, and VGG16 misclassified animals and struggled with counting. The action recognition proved insufficient due to using human action labels. Being an object detection model, the YOLO model showed great promise for future work, where fine-tuning is needed to achieve higher performance.

Contents

1 – Introduction

Camera traps have become an indispensable tool in ecological research, helping researchers to monitor wildlife and biodiversity (Wearn and Glover-Kapfer, 2019). This provides valuable data for studying what animals are seen, the number of animals of the same categories, and the animal behaviour. As the amount of data collected increases, it becomes very time-consuming for a person to label each video manually. There is another issue where some details can be missed, such as misidentifying an animal or counting the animals correctly. This could be due to poor lighting conditions or human error during the label. This limitation emphasises the necessity for automated solutions to efficiently process and analyse the data. So, what can be done to reduce the errors and increase the productivity of this workflow?

Advancements in machine learning and artificial intelligence have greatly improved exponentially over the past years (McKinsey & Company, 2022). Image classification and object detection models have come a long way and have made big strides in accurately identifying objects in images and video (Abbas and Singh, 2018). Leveraging these models can improve workflow and productivity by incorporating them into our work, especially for ecological studies of animals, where there can be 1000 hours of data that need to be analysed and labelled. This process can be automated rather than manually labelled by hand.

In collaboration with the School of Life Sciences and Dr Christopher Sandom, they have provided video data of camera traps placed around the surrounding areas of the University of Sussex, capturing animals ranging from longhorn cattle to red foxes to rabbits. Currently, they have master students who hand-label each individual video data, which is time-consuming and will lead to mistakes, as found later in this project.

The objective of this project:

1. Animal Classification – Automate the process of identifying the animal seen in the videos.
2. Animal Count – Automate the process of counting the number of animals of each class seen in the video.
3. Animal Action Recognition – Automate the process of identifying the behaviour of the animals seen in the video.

To complete these tasks, two different approaches were devised:

Approach 1: A fine-tuned VGG16 model was implemented for the animal classification and counting task. The VGG16 model could not be used for the action recognition task due to the architecture limitation in handling temporal information, which is why approach 2 was implemented.

Approach 2: YOLOv8 was implemented to overcome the limitations and improve the accuracy achieved by VGG16. YOLOv8 model was implemented for the animal classification and counting tasks in combination with the SlowFast action recognition task.

In the implementation of these models, we study their capabilities and see if they can improve the accuracy and efficiency of wildlife monitoring, which can eliminate the need for manual labelling and reduce human labelling errors.

2 – Literature Review

In this Literature Review, different existing research and implementations are explored to provide a foundational understanding of the study of animal detection, counting, and action recognition in videos. This review will delve into various approaches that have been developed over the years, from traditional image classification methods to modern deep learning techniques, such as object detection and action recognition models. By analysing these studies, we seek to identify current trends and challenges in the research, which will provide valuable insights for our own study.

In the early stages of research on animal detection, image classification models such as ResNet and VGG were widely implemented. These models have shown considerable success in accurately classifying animals from images. For example, in a study by Mahardi et al., (2020), they fine-tuned pre-trained VGG16 and VGG19 models on a custom dataset consisting of 21 classes of dog and cat breeds. The researchers found the fine-tuned VGG16 model achieved a training accuracy of 98.47% and a testing accuracy of 83.68%, while the VGG19 model achieved a marginally higher training accuracy of 98.59% and a testing accuracy of 84.07%. These results highlight the effectiveness of VGG architectures in accurately classifying breeds even with a limited amount of data. However, when applied to unseen test data, there was a noticeable drop in performance, indicating potential areas for further refinement to enhance generalisation.

In another study by Le et al., (2022), VGG-UNet and VGG16 were combined for an open-set shark detection and recognition to see if the model could detect and classify known and unknown sharks from their dataset, where the model achieved an overall accuracy of 81%. However, this study also highlights the need for further improvements, particularly in distinguishing between similar individual subjects, which remains a significant challenge in animal detection. While these studies demonstrate the potential of CNN-based models for animal classification, limitations of traditional classification models, particularly in generalisation and distinguishing between closely related classes, are evident in both papers. These challenges have encouraged further exploration into more advanced object detection models.

Building on these early successes in image classification, research has further progressed, with object detection models like YOLO (You Only Look Once) and Faster R-CNN emerging to bring further significant advancements with capabilities such as real-time object detection and identifying multiple object precise positions within a frame. A study by Zhong et al., (2022), evaluated YOLOv3, YOLOv5, and YOLOR models for their speed and accuracy. They found that YOLOv3 demonstrated the highest precision, especially in detecting small objects like fish, and YOLOv5 was the most efficient execution time, while YOLOR showed both precision and speed, offering an adaptable option for various detection scenarios.

In another study (Tan et al., (2022), researchers set out to see the effectiveness of detecting and classifying animals in camera trap images using YOLOv3, Faster R-CNN and RetinaNet. The YOLOv3 model achieved an accuracy of 85% with an inference time of 20 ms, RetinaNet achieved an accuracy of 89% with an inference time of 35 ms, and Faster R-CNN had the highest accuracy at 92%, but with a longer inference time of 60 ms. A study by Martha et al., (2023) also evaluated and compared the effectiveness of YOLOv3, SSD (Single Shot Multibox Detector) and Faster R-CNN in real-time cow detection. The YOLOv3 was the most effective model, showcasing a good balance between accuracy and real-time performance, achieving an accuracy of 87.5% and an inference time of 22 ms, the SSD model achieved an accuracy of 75.3% with an inference time of 15 ms. In comparison, Faster R-CNN had the highest accuracy at 91.2% but with a significantly slower inference time of 58 ms.

These studies have shown the capabilities of different object detection models, with YOLO being very effective in real-time monitoring, and models like Faster R-CNN, despite their higher accuracy, are more suited to scenarios where speed is less necessary.

Counting objects in videos is another ongoing research area aimed at developing more robust algorithms that can handle a broader range of scenarios, such as denser crowds, overlapping objects, and changing environmental conditions. In a study by Wu et al (2023), an improved method for accurately counting wheat ears was developed. The researchers proposed several improvements to the YOLOv7 model, including the integration of ODConv, GCNet, and CA. The researchers discovered that the enhancements significantly improved the model's detection accuracy, where a mean average precision (mAP) of 96.2% was achieved, a 2.5% improvement over the original YOLOv7 model. Moreover, the enhanced model exhibited a higher precision of 93.5% and recall rates of 92.4%. The model also showed improved tracking accuracy when used with DeepSORT, highlighting the model's robustness. However, the study identified areas for further refinement, as some wheat ears were missed or misidentified due to severe occlusion and poor lighting conditions. Irregular motion speeds in the video data posed another challenge for consistent tracking.

Similarly, Kumar et al. (2021) conducted a study to develop an efficient model for multiple object tracking and counting using YOLOv4 and DeepSORT. They found that YOLOv4 and DeepSORT are very effective, particularly in environments with moderate to high object density. However, the study also revealed limitations, particularly in poor lighting conditions and bad weather, which affected detection accuracy. Both studies indicated that these models are robust, but their performance may be significantly impacted by difficult environmental conditions such as poor lighting and bad weather.

While these studies focused on plant and general object counting, their findings have important implications for animal counting in videos, especially in challenging environmental conditions.

As research on animal detection and object counting has advanced, the focus has expanded to understanding and recognising behaviours in videos. Models such as I3D (Inflated 3D ConvNet) and SlowFast have shown remarkable abilities in accurately recognising video behaviours. In a study by Nguyen et al. (2019), the effectiveness of an I3D and the R(2+1)D model for recognising mouse behaviours in videos was evaluated. The I3D model achieved an accuracy of 96.9%, and the R(2+1)D model achieved an accuracy of 96.3%. However, both models struggle to identify behaviours that are similar to other behaviours, such as drinking and eating. This study has shown further research is needed.

This Literature Review has examined different approaches to animal detection, counting, and action recognition in videos. While significant progress has been made in these areas, challenges remain. Image classification models have shown high accuracy but struggle with generalisation with unseen data. Object detection models can detect multiple objects in real-time but face challenges in complex environmental conditions. Counting methods have improved but still struggle with occlusions and poor lighting. Action recognition models show promise but have difficulty distinguishing between similar behaviours.

This project aims to implement and evaluate these existing approaches in a program for wildlife video analysis. By applying established models, we aim to evaluate their collective effectiveness in real-world wildlife monitoring scenarios. This integration of proven methods will help us investigate how these different models can work together and identify any practical challenges or limitations in their combined application to automated wildlife monitoring.

3 – Methodology

This section will discuss the methodologies used to detect, count, and recognise the behaviour of animals in videos. Two distinct approaches were developed for this project, along with the implementation of a graphical user interface (GUI). Each approach is detailed, with the reasoning behind each design decision.

The first approach, a pre-trained VGG16 model fine-tuned on custom data, was implemented using PyTorch. It is known for its architecture, which is very effective in image classification tasks, as seen in studies in the Literature Review. This approach was implemented for animal classification and counting. The second approach, Ultralytics YOLOv8, was implemented for animal classification and counting, and the SlowFast model was incorporated with YOLOv8 for action recognition. The second approach was implemented to achieve greater accuracy, further discussed in the results section. A GUI was designed to allow users to select folders with videos to process and visualise the results. Frameworks used for data processing, evaluation, and visualisation of results include Pandas (pandas, 2024), Scikit-learn (scikit-learn, 2024), Matplotlib (matplotlib, 2024), and Seaborn (seaborn, 2024).

3.1 – Data Preparation

The Department of Life Sciences provided 6,915 video data with labels, but due to storage limitations, only 420 video data files were selected for this project's training and validation for VGG16. For the 420 video data, there were 14 class subjects, as seen in Table 1.

Table 1: Training Data Animal Class from the

Subject	
Longhorn Cattle	Red Fox
Human	No Animal
Horse Rider	Red Deer
Roe Deer	Pheasant
Fallow Deer	Rabbit
Dog	Jackdaw
Bird	Magpie

51 test data were selected to see how the fine-tuned VGG16 and YOLO8 perform. Table 2 shows the subjects seen in the test data.

Table 2: Test Data Animal Class

Subject	
Longhorn Cattle	Red Fox
Bird	Fallow Deer
Rabbit	Deer spp
Person	NA

It is important to note that there are some labelling issues where original annotators have mislabelled and not been consistent with labelling, leading to some mistakes. For example, some annotators have labelled animals not seen in the video as ‘No Animal’ whereas some annotators have labelled as ‘NA’, or when there is a person present in the video, some annotators have labelled it as ‘Human’ and some as ‘Person’. This issue was addressed by mapping the labels, which also helps to work the models' performance metrics, discussed further in the Evaluation of the Model within the method section.

Frame Extraction

To fine-tune the pre-trained VGG16 model, individual frames from the video data are required to process for animal classification and counting. Therefore, frames need to be extracted from the videos. Using OpenCV to save each frame as a separate image file, frames were extracted from the training video data at regular intervals of 5 seconds, ensuring sufficient data was available for training the VGG16 model. The extracted frames for the VGG16 approach are split into training and validation sets, which are split into 80/20 using scikit-learn's train_test_split function (scikit-learn, 2024). The reason for allocating 80% of the data to training was the low amount of initial training data available. So, it was decided to have the most available data for training the VGG16.

The table below shows the distribution of images across the training and validation sets for each subject.

Table 3: Training and Validation Data Split - Image Counts per Subject

Subject	Train Count	Validation Count
Longhorn Cattle	1552	388
Human	50	13
Horse Rider	5	2
Roe Deer	71	18
Fallow Deer	40	10
Dog	6	1
Bird	50	13
Red Fox	21	5
No Animal	278	70
Red Deer	6	1
Pheasant	6	1
Rabbit	37	10
Jackdaw	6	1
Magpie	6	1

Table 3 above shows an imbalance between the subjects, with Longhorn Cattle data having a higher count than any other subjects. To overcome this issue, data augmentation was applied during training the VGG16, which is further discussed in the implementation of VGG16.

3.2 – Approach 1: VGG16

For the first approach, VGG16 was implemented using the PyTorch framework for animal classification and counting (Paszke et al., 2019). VGG16 was developed by the University of Oxford (Simonyan and Zisserman, 2015) to tackle the challenges faced with image classification. In the study, the researchers aimed to identify vegetable pest images. They found that the fine-tuned VGG16 achieved 99.90% and VGG19 achieved 99.99%. These models significantly outperformed both their non-fine-tuned counterparts and other advanced models, such as Inception V3 and ResNet50 (Ye et al., 2019). The VGG16 model was chosen over the VGG19 model for this task because of the marginal improvement in the accuracy, as seen in the study mentioned above and the study mentioned in the Literature Review (Mahardi et al., (2020)). VGG16 has a simpler architecture which will also require less computation and memory resources, leading to faster training.

Before training the VGG16 model, data augmentation was applied to the extracted frames. Data augmentation can help mitigate overfitting, improve generalisation, compensate for class imbalances, and improve the overall accuracy of the VGG16 model. In a study conducted by Thanapol et al., (2020), it was shown that data augmentation effectively improved the performance of convolutional neural networks (CNNs), even when the training data was limited. Their approach significantly reduced overfitting, leading to better generalisation. with a noticeable increase in the model accuracy, with test accuracy improving from 49.80% without data augmentation to 61.50% with data augmentation. In another study by Uswatun Khasanah, Utami, and Raharjo (2020), data augmentation was applied to a pre-trained VGG16 model to classify different types of Batik patterns. The VGG16 model initially achieved an accuracy of 95.83% without any data augmentation but improved to 98.96% with data augmentation.

These studies demonstrate a significant improvement in the performance of data augmentation on a VGG model. These improvements are evident in model accuracy, robustness, and generalisation, particularly when dealing with smaller datasets or solving complex image classification tasks.

Using PyTorch's 'torchvision.transforms' (PyTorch, 2024), data augmentation was applied, which included resizing the image to 512x512 pixels, which was done to balance computational efficiency and maintain sufficient detail in the images for accurate classification and to ensure consistency in the input dimensions. Random resized cropping during training to 448x448 pixels also gave variability in the framing of the animals, simulating different viewpoints. A 50% random flip was applied to increase the variability in the animal's orientation in the frames. These data augmentations will help artificially increase the diversity of training data, allowing the model to become more robust and achieve greater accuracy.

After the data augmentations were applied, the pixels of the imaged were normalised to the mean and standard deviation values of the ImageNet dataset, specifically [0.485, 0.456, 0.406] for the mean and [0.229, 0.224, 0.225] for the standard deviation, which done to match the original values of pre-trained ImageNet dataset (He et al., 2016).

VGG16 Model Architecture

The pre-trained VGG16 model was trained with weights from the ImageNet dataset, which includes 1,000 ImageNet classes. Instead of modifying the final fully connected layer to output 14 classes corresponding to the animal classes from the custom training dataset, the decision was made to fine-tune the entire VGG16 model. This choice was motivated by the small dataset size as well as the complex and nuanced differences between animal classes in the dataset, which may not be entirely captured by the pre-trained features alone. Allowing all layers to be updated during training, particularly the earlier layers responsible for capturing low-level features like edges and textures, enabled the model to adapt its feature extractors to the precise characteristics of animal classification and count tasks, leading to improved model accuracy. This approach is supported by a study in which researchers fine-tuned all layers of the VGG16 model on a small and specific dataset, resulting in higher accuracy in classification tasks (Giraddi et al., 2020).

VGG16 Model Training

A Cross-Entropy Loss function was used as it is well suited for multiclass classification problems (Demirkaya, Chen and Oymak, 2020). The VGG16 model was compiled using a Stochastic Gradient Descent (SGD) optimiser with momentum. Research by Keskar and Socher (2017), show that while adaptive optimisers like Adam perform well during the initial stages of training, SGD outperforms Adam in the later stages, leading to better generalisation performance. Another study by Qian (1999) demonstrated that using momentum in gradient descent helps improve convergence speed, leading to more efficient and effective training. The learning rate was consistent with the learning rate $\eta = 0.001$[1] and the momentum rate set to 0.9. This fixed learning rate has a good balance between speed and stability, enabling the model to learn effectively during training (Vitaly Bushaev, 2017).

[1] Learning Rate hyperparameter will affect how quickly the model learns during training.

The model was trained for 25 epochs with a batch size of 32. This was done to ensure robust convergence, providing time for the model to learn the patterns in the data while maintaining a good balance between computational efficiency and training stability. This approach will help the model learn effectively from the dataset without excessive memory usage. To ensure the consistency and reliability of the training process, the model was trained three times using the same hyperparameters. This approach ensured that similar training performance could be consistently achieved across multiple runs.

The training results are further delved in the Results section of the report.

VGG16 Classification

After the training is completed, the fine-tuned VGG16 model is loaded and ready for animal classification and counting of animals in new video data. Additional functions were added to improve the model's accuracy and processing of new data. When new data is loaded for the VGG16 model to make predictions, frames need to be extracted from the videos, so frames from 0, 10, 20, and 30 seconds of the video are extracted to match the same layout of the labelled data provided by the Life of Sciences department. The video data contains atmospheric pressure, temperature, date, time, and location of the video, which is visible on the video frames, as seen in Figure 1. This information is unnecessary for image classification. Therefore, the frames were cropped, which helps the model to focus only on the regions containing the animals.

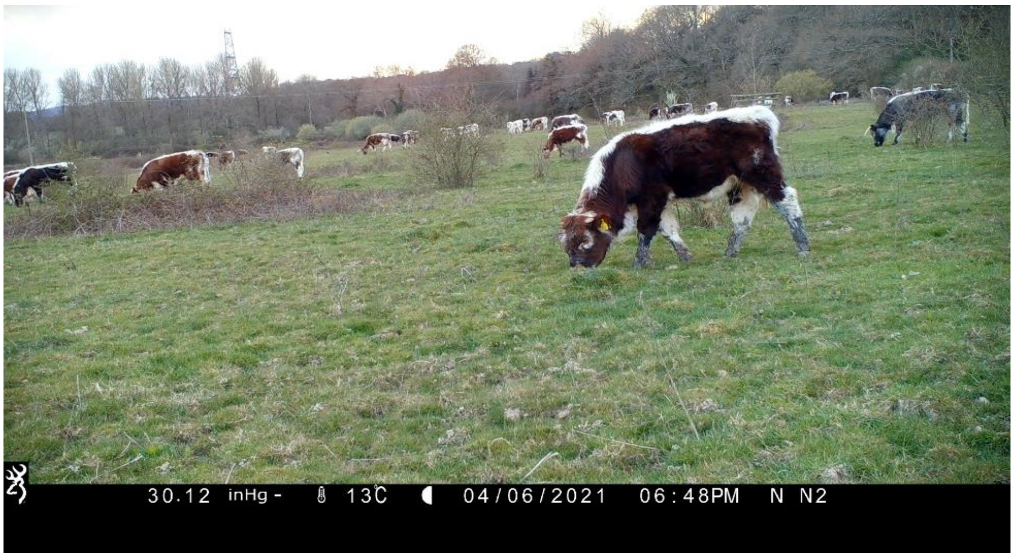


Figure 1: An example of a video frame captured by a camera trap, showing longhorn cattle grazing in a field, with additional data seen at the bottom of the frame containing atmospheric pressure, temperature, date, time, and location of the video.

Before the model processes the newly extracted images, they are resized to 512x512 pixels, converted to a tensor, and normalised using the same preprocessing steps to match the training process. This preprocessing ensures that the input data matches the model's expected format and scale, which is crucial for maintaining accuracy and consistency in predictions.

To better understand the VGG16 model predictions, Grad-CAM++ was integrated, which generates heatmaps that provide visual explanations for the model's prediction reasoning, highlighting the areas of the image that were most influential in the decision-making process (Chattopadhyay et al., 2018). Grad-CAM++ was specifically chosen over Grad-CAM due to its improved accuracy in handling complex scenes, which can be very beneficial for this dataset, where animals could be obscured by another object (Inbaraj et al., 2021). The heatmaps which are generated can also help identify potential areas where the model might be focusing incorrectly and be used as feedback for model improvement.

A threshold was set to ensure the reliability of the model's predictions, where the VGG16 model will only make predictions with a probability if the probability of the predicted class is higher than 10%. This was integrated to reduce false negatives by filtering out low-confidence predictions, which helps the model achieve greater accuracy and reliability in the animal classification and counting tasks. The processed images are stored with folder and filenames that include timestamps and predicted class labels. The predictions of the animal classification are saved in a CSV file, which includes the video file name, timestamp, predicted animal class, and the count of animals detected.

The results of the VGG16 model classification and predictions are further discussed in the Result section.

Figure 2 shows the workflow of VGG16 implementation.

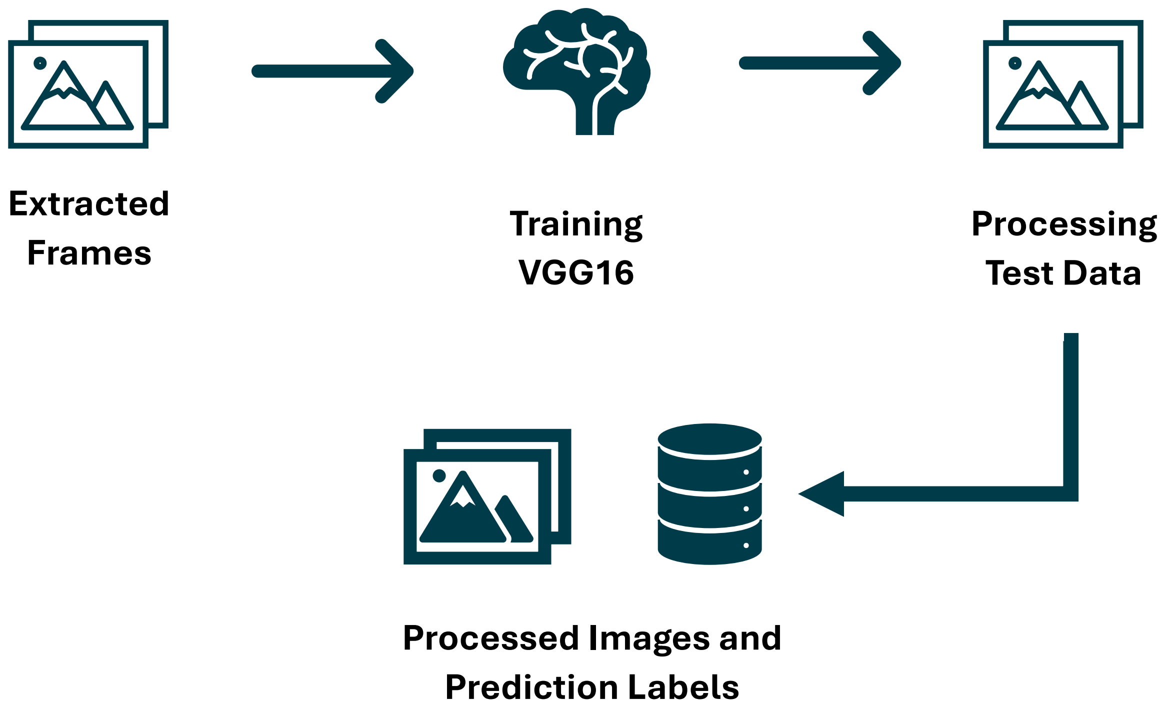


Figure 2: Overview of implementation of the VGG16 model. This diagram illustrates the implementation of the VGG16 model. First, extracted frames are passed to VGG16 to be fine-tuned. Then, the trained VGG16 model processes new data to perform animal classification and counting. Finally, the VGG16 saves the heatmap frames of the images and predictions are saved as CSV.

3.3 – Approach 2: YOLOv8

For the second approach, YOLOv8 was implemented using the Ultralytics framework (Ultralytics, 2024), which is pre-trained in the COCO dataset for the animal detection and count task. YOLOv8 was chosen for efficiency in processing speed and time while maintaining accuracy compared to other object detection models, such as Fast R-CNN, as seen in some studies in the Literature Review. YOLOv8 has improved speed and accuracy over the previous generations of YOLO models, as seen in Figure 3. Different variants of YOLOv8, as seen in the left graph Figure 3, have different mAP scores. The YOLOv8x model was selected for animal detection and count tasks due to a higher mAP score compared to the other variants of YOLOv8, as seen in Figure 3. The only drawback of using YOLOv8x is that it has a higher latency due to a higher number of parameters, leading to increased computational resources. However, running on the YOLOv8 model on RTX A4000 will be fine.

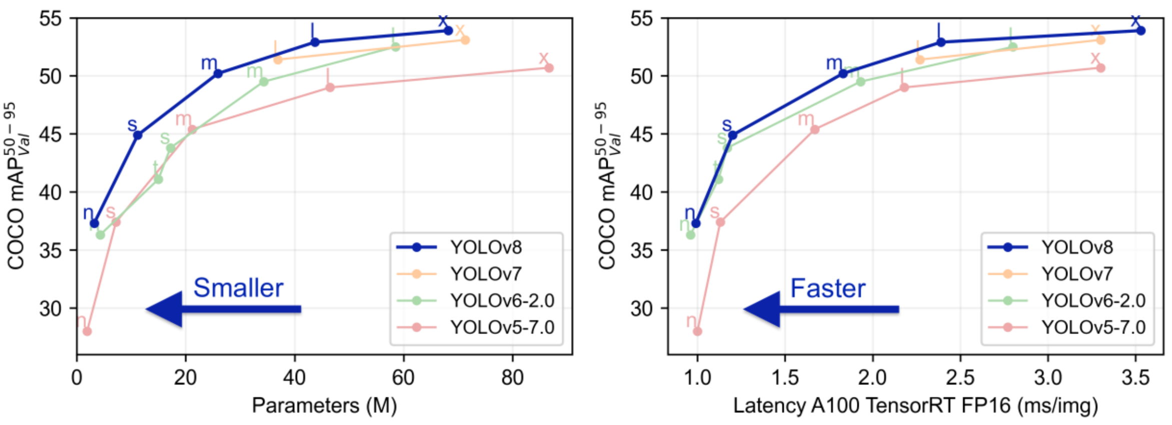


Figure 3: Ultralytics (2024) presents a performance comparison of the YOLOv8 model with earlier YOLO models using the COCO dataset. The graph on the left illustrates the correlation between the number of parameters (in millions) and the COCO mAP (mean Average Precision) for each model variant. Across all versions, YOLOv8 exhibits exceptional performance, achieving higher mAP scores than previous versions of the YOLO models.

The right graph illustrates the COCO mAP against inference latency, measured on an A100 GPU with TensorRT in FP16. Again, all versions of YOLOv8 outperform previous versions of the YOLO models by delivering faster inference times while maintaining high mAP scores.

To enhance the YOLOv8 model's capability for accurate animal counting and action recognition tracking, DeepSORT was implemented using the deep_sort_realtime library. DeepSORT assigns unique IDs to each animal detected in the videos, ensuring that individual animals are consistently tracked, even when the animal becomes occluded, moves out of the frame, or the YOLOv8 model briefly loses the detection of the animal (Wojke, Bewley and Paulus, 2017). DeepSORT ensures the same animal is not counted multiple times and ensures that action recognition is correctly assigned to the specific animal. In a study by Bai (2023), DeepSORT showcased enhanced tracking performance, helping the YOLO model to keep track of multiple objects even with occlusion.

The hyperparameter setting for DeepSORT is as follows: 'max_age' was set to 25, which keeps track of the animal for 25 frames when detection is lost. 'n_init' was set to 2, which requires the model to see an animal for 2 consecutive frames before it starts tracking, which prevents brief detections from being counted as valid which reduces false positive detections. The 'nms_max_overlap' is set to 1.1 to allow for some overlap between the bounding boxes of the animals, as there will be scenarios where animals could be moved in front of each other and be occluded by one another.

The YOLOv8 model is not fine-tuned on custom data due to the absence of bounding box annotations in the custom dataset, which is only labelled in a CSV format. As a result, YOLOv8 pre-trained on the

COCO dataset is used. The COCO dataset includes 80 object categories featuring various animals, such as cows, horses, birds, and dogs (Lin et al., 2014). These categories are highly relevant and closely align with the objects in this task, making the COCO dataset a suitable choice.

A full list of the 80 labels can be found in Appendix A.

For the action recognition task, SlowFast using PyTorch was integrated with YOLOv8. SlowFast outperforms other action recognition models, such as I3D and C3D, with a significant difference in accuracy (Feichtenhofer et al., 2019). Additionally, a study by Hassan, Elgabry and Elsayed Hemayed, (2021) further demonstrated the effectiveness of SlowFast in dynamic sign language recognition, where it achieved a Top 1 accuracy of 79.34% on the WLASL 300 dataset, outperforming the I3D model, which achieved an accuracy of 56.14%. There are two variants of the SlowFast, 'R50' and 'R101'. For this action recognition task, 'R101' was used due to having higher accuracy on Top 1 scores.

The SlowFast model used the Kinetics-400 labels, which contain 400 labels for human actions (Kay et al., 2017). These labels were chosen due to limitations, such as restrictions on installing some packages and libraries on the university computer. For example, the installation of Detectron2, necessary for processing the Animal Kingdom Dataset containing various animal actions for training, was restricted and, therefore, was unable to be used. However, the Kinetics-400 labels include labels such as running and eating, which could be detected by the SlowFast model.

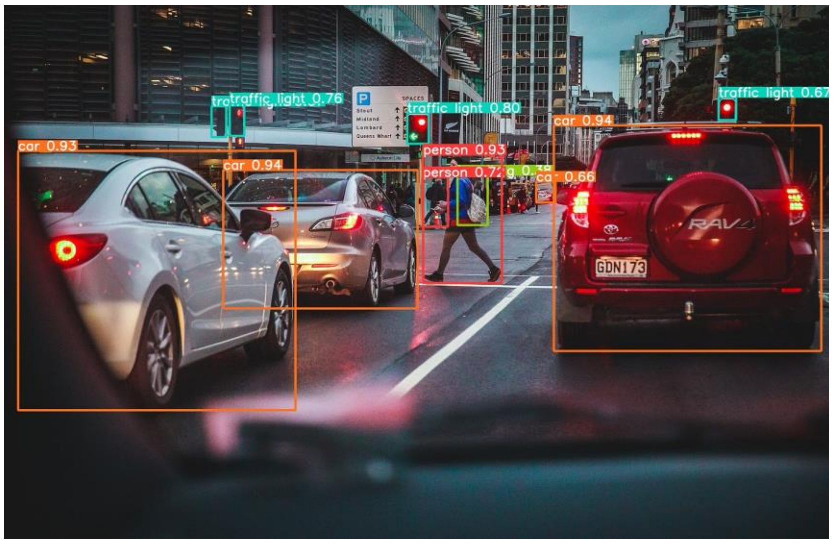


Figure 4: The original image was taken by James Coleman and is available on Unsplash. Dorfer (2023) applied the YOLOv8 model to the image. The image shows the detection of different objects, such as cars, traffic lights, and pedestrians, with confidence scores for each detection.

Figure 4 shows an example of YOLOv8 annotation applied to an image. The YOLOv8 for this task will produce similar annotations with extra details with object ID, an action recognised by the SlowFast model.

YOLOv8 with SlowFast Classification

For the YOLOv8 model, detections are made only when the confidence score is greater than the threshold of 0.5. This confidence threshold helps filter out low-confidence predictions so that only high-confidence detections are processed. This helps the model reduce the probability of false positives, which is important in maintaining high accuracy and reliability of the detection process.

Before passing raw video data into the SlowFast model, preprocessing is applied on each frame. The frames are resized to 256x256 pixels to ensure consistency. 32 frames are evenly sampled to capture temporal actions. The pixel values of these frames are then normalised by dividing by 255, scaling them between 0 and 1. Moreover, they are normalised again using the SlowFast model's specified mean of [0.45, 0.45, 0.45] and standard deviation of [0.225, 0.225, 0.225].

The raw video is passed onto the YOLOv8 model to make predictions on what animals are seen in the video. The YOLOv8 model produces a bounding box, where there are labels for the object detected with the confidence score of the label, and an ID is assigned to the animal by DeepSORT to keep track of the animal. When another new animal is detected and assigned an ID by DeepSORT, the count for that particular animal type is incremented. SlowFast model will then make action predictions for the tracked animals. The video data with bounding box annotation is saved, and the CSV is plotted with folder and file name, subject (animal) detected by YOLOv8, Animal ID the ID assigned by DeepSORT, Behaviour of the animal by the SlowFast model and the count for each type of animal in the video. When no animal is detected, 'NA' is entered for that field, and the count is set to 0 in the CSV file.

The results of the YOLOv8 and SlowFast model classification and predictions are further discussed in the Result section.

Figure 5 shows the workflow of YOLOv8 implementation.

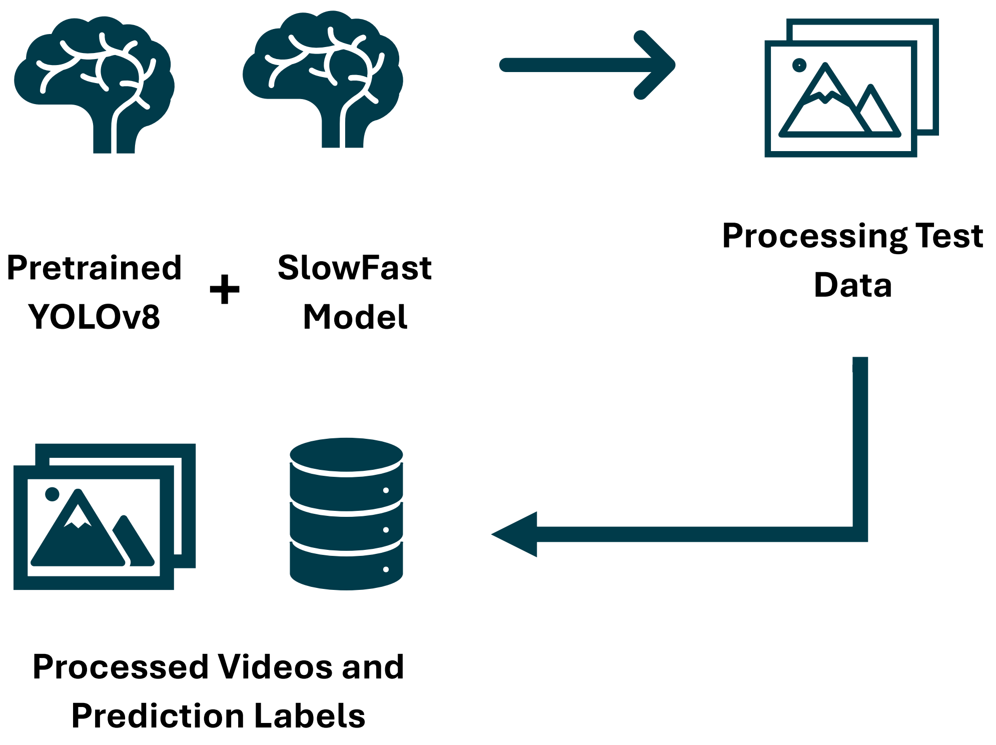


Figure 5: Overview of implementation of the YOLOv8 model. This diagram illustrates the workflow of the YOLOv8 and SlowFast model processing new data. YOLOv8 and the SlowFast model process new data for animal classification, counting and action recognition. The model at the end saves the processed videos with bounding box annotations, and predictions are saved as CSV.

3.4 – Graphical User Interface

To allow the users to upload folders of video data for processing either using VGG16 or YOLOv8 for animal detection and action recognition, a graphical user interface (GUI) was designed using the Tkinter library to make a user-friendly and intuitive, with midnight green, which is similar colour to the University of Sussex logo.

The GUI allows users to select a folder containing the videos to process, and the user can select an output folder where they want the processed images (if the user selects VGG16) or processed video (if the user selects YOLOv8) with the CSV classification predictions to be saved. The default model is set to YOLOv8 due to its higher performance (discussed further in the Results section why this was done), but users can select the VGG16 model if they want. Any errors will be printed. directly on the interface GUI in red to allow the users to know if there are any issues processing the videos.

Note: The selected folder for processing must contain a subfolder with the video files. The same structure applies to processing even just one video file. The structure should be as follows: Test Data/Folder1, Test Data/Folder2, where video files are in Folder1 and Folder2. This is done to follow the same naming rule as the labelled data, which is why the folder name is needed to save the processed video file and CSV predictions.

The user interface of the GUI:

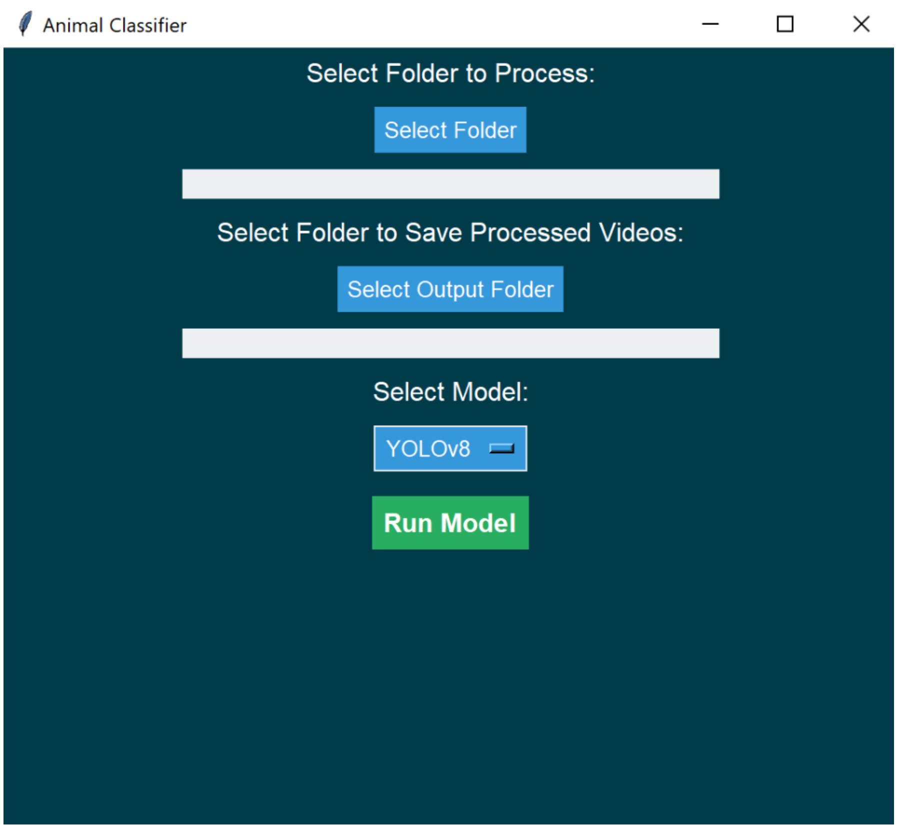


Figure 6: The GUI interface for video processing using either VGG16 or YOLOv8.

3.5 – Evaluation of the Models

To evaluate the VGG16 and YOLOv8 model's predictions for animal classification, counting and action recognition, using Scikit-learn, the accuracy, precision, recall, F1 scores and confusion matrices are worked out to compare and analyse how the model's predictions perform against the ground truth, using macro averaging to give all the class labels equal importance.

The accuracy, precision, recall, and F1 scores for animal counting and behaviour recognition are calculated after checking if the correct animal is identified. This is done to give a more accurate examination of the model's performance in both tasks and reduce the false positives in counting and behaviour recognition for objects which are not relevant to this project.

The accuracy will give a clearer indication of the overall correctness of the models in the predictions of the animal classification, count and behaviour. The precision and recall will show how well the model is predicting. The F1 Score will give better insight into how the models get the right predictions, with a higher score illustrating predictions with minimal false positives and false negatives. The confusion matrices will help even further to better understand the prediction distribution across different classes from each model, which can highlight any classes the models struggle with.

Calculating the Scores

When preparing the test data (ground truth) for evaluation, some discrepancies were identified in counting the animals for some videos. A new file was created to address this issue where animal counts were manually recounted. It is important to note that there still could be errors with the counts. This should be taken into consideration when analysing the results.

The 'Subjects' field from the predictions is standardised to uppercase for consistency. As mentioned earlier in Data Preparation, there are some wrong labelling for some animals. To overcome this issue, for the VGG16 model, the 'HUMAN' label was mapped to 'PERSON', and the 'NO ANIMAL' label was mapped to 'NA'. The YOLOv8 model, which used the COCO dataset, has different labels and does not match the ground truth labels, whereas the YOLOv8 model could have predicted 'cow', but the ground truth label would be 'Longhorn Cattle', the COCO dataset does not have specific cow breads like longhorn cattle. Therefore, for YOLOv8, 'cow' is mapped to 'LONGHORN CATTLE'. The labels were then merged with the ground truth labels based on the video folder and file name.

For the behaviour metrics, only the ground truth labels were used. This is because the YOLOv8 model predicted 62 behaviour labels, making the confusion matrix difficult to analyse. Therefore, the ground truth labels will make the confusion matrix easier to interpret.

Note: Please read the README file before running the code to install all the necessary packages and libraries.

4 – Results

In this section, performance analysis of the models developed for animal classification, counting, and action recognition are presented. First, the training and validation performance of the VGG16 model is discussed, followed by a comprehensive evaluation of both the VGG16 and YOLOv8 models on the test data. This will include metrics such as accuracy, precision, recall, F1 scores, and a confusion matrix to provide a deep understanding of the effectiveness of the models and highlight their respective strengths and weaknesses.

4.1 – VGG16 Training Performance

To ensure the consistency and reliability of the training process, the model was trained three times using the same hyperparameters. This process helps to reduce the impact of potential anomalies, such as random weight initialisation or batch selection and confirms the stability of the model across the three runs.

Run 1

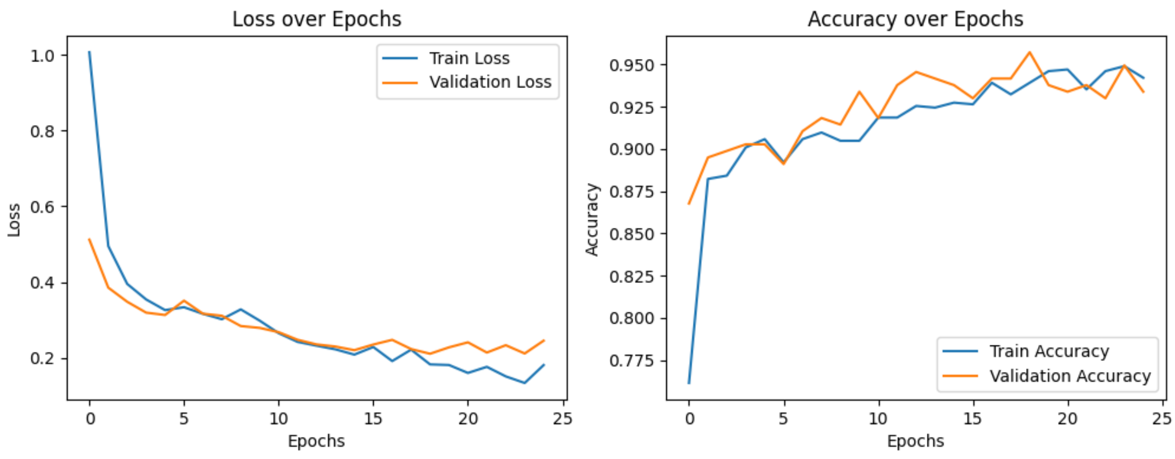


Figure 7: Training Run 1. The graph on the left shows loss over epochs, and the graph on the right shows accuracy over epochs.

The first training run shows a steady decrease in both training and validation loss, with the training loss reaching 20% and the validation loss of 30% by the end of the training. The training and validation accuracy follow a similar pattern where rapid increases are seen. Both improved in accuracy over the epochs, with the final training accuracy of 93% and validation accuracy of 92%.

Run 2

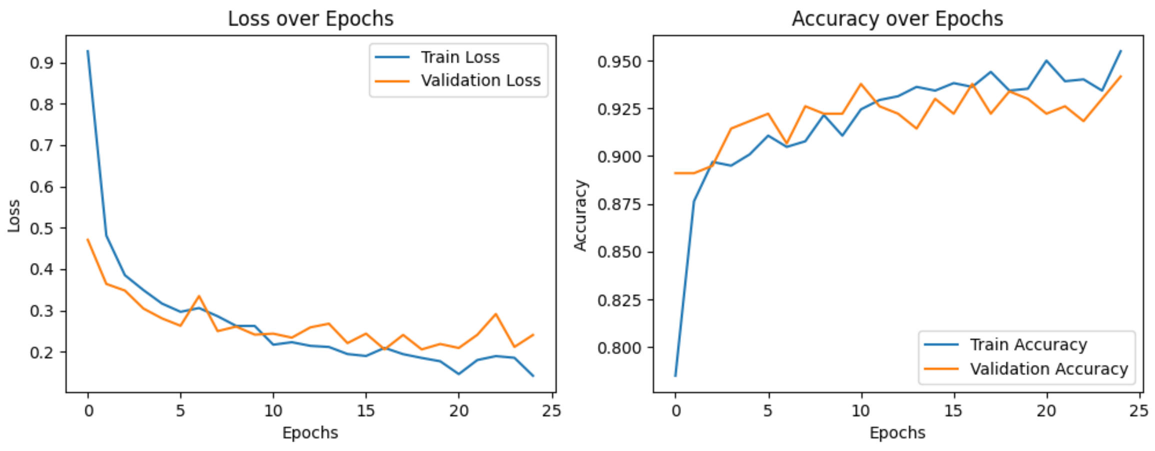


Figure 8: Training Run 2. The graph on the left shows loss over epochs, and the graph on the right shows accuracy over epochs.

The second run is very similar to the first run, following the same pattern of both loss and accuracy. However, this run achieved lower final loss values, with a training loss of 17% and a validation loss of 27%. A higher training and validation accuracy was achieved of 95% and 93%, higher than the first run.

Run 3

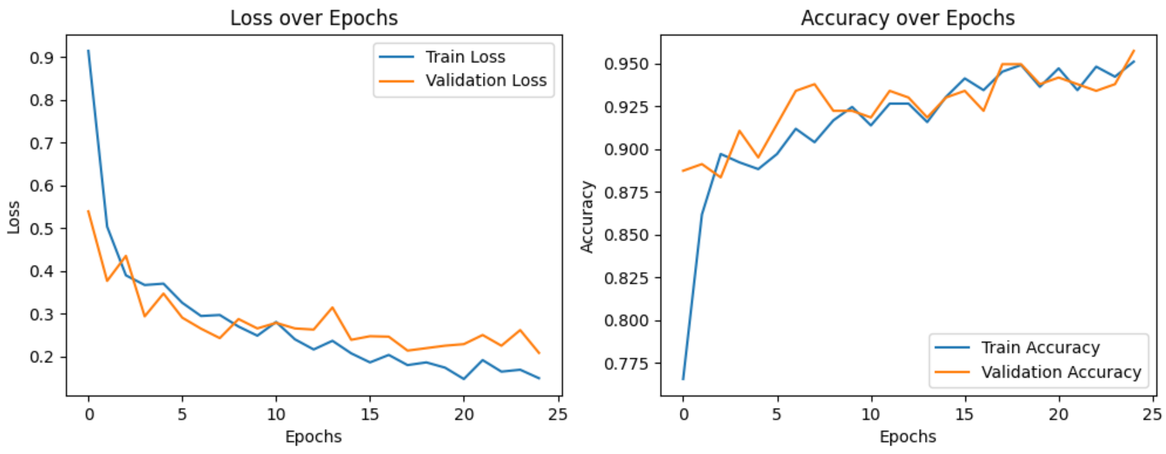


Figure 9: Training Run 3. The graph on the left shows loss over epochs, and the graph on the right shows accuracy over epochs.

The third training run showed similar training characteristics to previous training runs. The final training loss is 18%, while the validation loss settles at about 23%, showing the smallest gap between training and validation loss, suggesting better generalisation compared to the previous runs. This run also achieved the highest training accuracy of 95% and validation accuracy of 95.5%, indicating exceptional generalisation.

The VGG16 model demonstrated good training performance across the three runs, with low training loss and higher training accuracy, indicating that the model was learning effectively and generalising well to unseen validation data. The third run achieved the highest accuracy from the training and was therefore used for the animal classification and count task.

4.2 – VGG16 Model Performance on Test Data

Animal Classification and Count Metrics

Table 4: Performance metrics for the VGG16 model

Task	Accuracy	Precision	Recall	F1 Score
Animal Classification	0.34	0.31	0.90	0.29
Animal Count	0.04	0.69	0.18	0.02

Note: The metrics are calculated only on correctly classified animals for the count scores. If the model did not correctly classify the animal, the count is not included in the metric calculations.

For the animal classification task, the VGG16 model achieved a low accuracy of 34%. The model could identify different classes of animals, with a recall of 90%, which indicates the model is very good at identifying different animal classes, but with a 31% precision, only a small amount of the classes is correctly identified, which has led to the low F1 Score of 29%.

For the counting task, the VGG16 achieved an accuracy of 4% and the F1 Score of 2%, which shows the model struggles to count the animals correctly. The high precision of 69% shows the model is correct in some cases with the count, but this is most likely due to the test data video mostly having only one animal in the video.

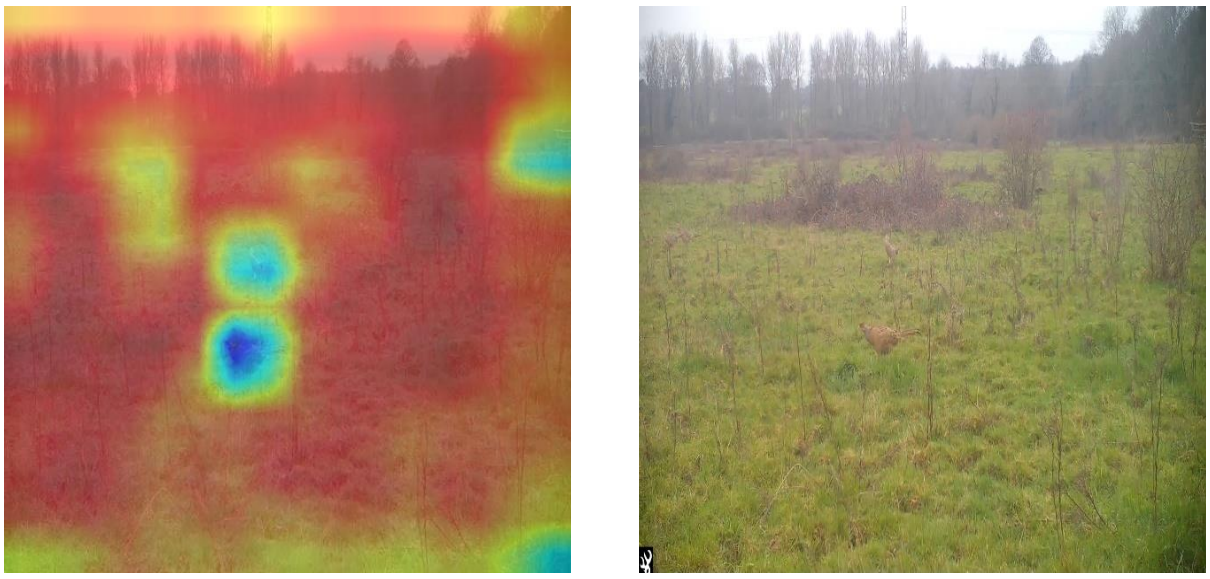
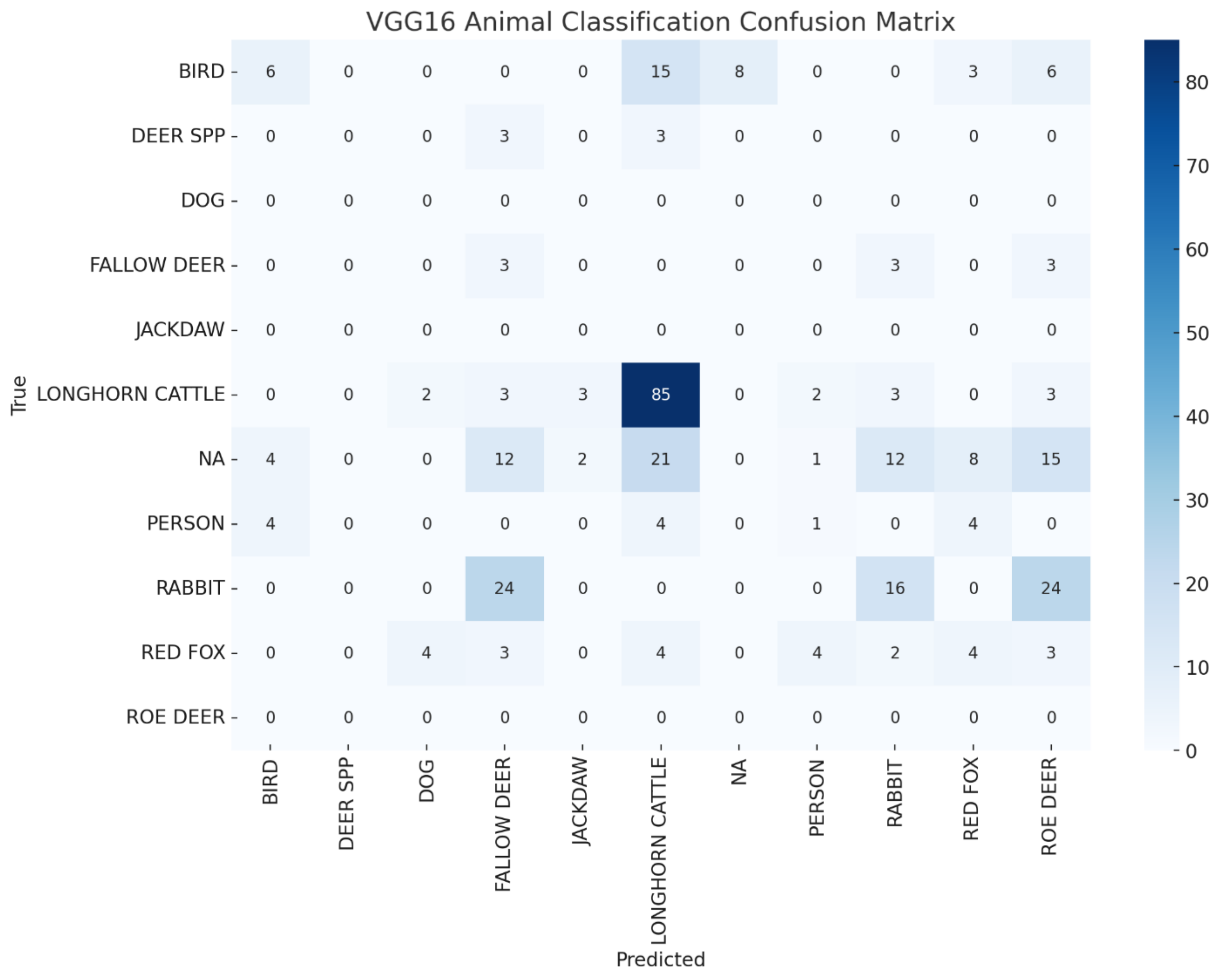


Figure 10: (File: N_N 3_2021_03_02/IMG_0165 at zero seconds). The left image shows the Grad-CAM++ heatmap and the right image shows the same image without the heatmap.

As seen in Figure 10, the VGG16 model is able to detect the bird seen at the centre of the image but struggles with the birds seen in the background, as proven by the VGG16 Grad-CAM++ image, where there is no heatmap for the other birds. Another problem with this particle video prediction is that the VGG16 predicted this animal as a Longhorn Cattle, not a bird. This misclassification was a common problem across the different video files, with the Longhorn Cattle being the most commonly identified animal. This example explains why the accuracy of animal classification and animal count is very low.

Animal Classification Confusion Matrix



True \ Predicted	BIRD	DEER SPP	DOG	FALLOW DEER	JACKDAW	LONGHORN CATTLE	NA	PERSON	RABBIT	RED FOX	ROE DEER
BIRD	6	0	0	0	0	15	8	0	0	3	6
DEER SPP	0	0	0	3	0	3	0	0	0	0	0
DOG	0	0	0	0	0	0	0	0	0	0	0
FALLOW DEER	0	0	0	3	0	0	0	0	3	0	3
JACKDAW	0	0	0	0	0	0	0	0	0	0	0
LONGHORN CATTLE	0	0	2	3	3	85	0	2	3	0	3
NA	4	0	0	12	2	21	0	1	12	8	15
PERSON	4	0	0	0	0	4	0	1	0	4	0
RABBIT	0	0	0	24	0	0	0	0	16	0	24
RED FOX	0	0	4	3	0	4	0	4	2	4	3
ROE DEER	0	0	0	0	0	0	0	0	0	0	0

Figure 11: The confusion matrix for the VGG16 Animal Classification task.

Figure 11 shows that there is a high bias for ‘Longhorn Cattle’ labels, where there are 85 instances that are correctly identified but also have misclassified another animal for Longhorn Cattle. For example, for the ‘Bird’ class, the model predicted ‘Longhorn Cattle’ in 15 instances. The model struggles to identify NA (No Animal) labels where it fails to correctly class a video where there are no animals. This misclassification continues with other class labels. For the ‘Rabbit’ true labels, the model predicted ‘Fallow Deer’. This showcases the model is struggling to find distinctive features from the video files to correctly classify different animals.

Animal Count Confusion Matrix

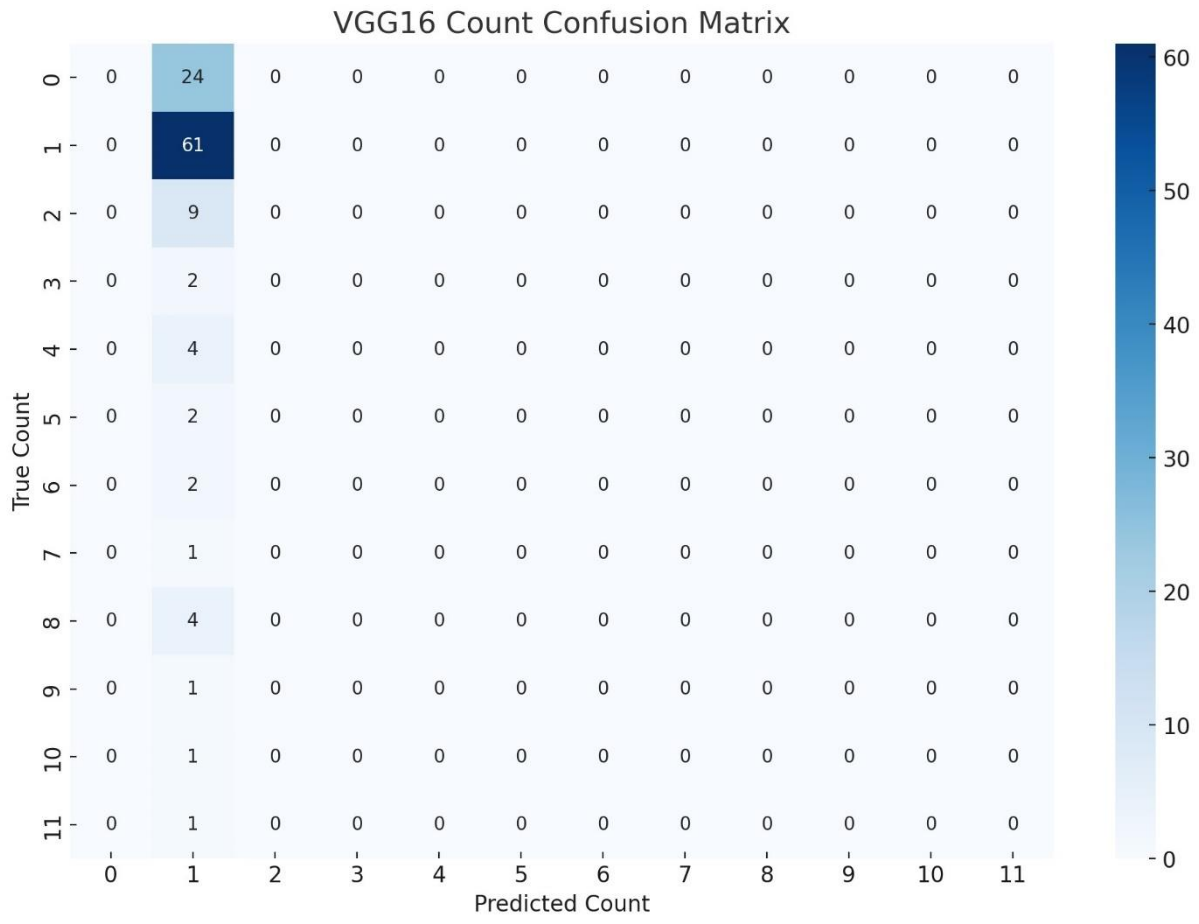


Figure 12: The confusion matrix for the VGG16 Animal Count task.

The confusion matrix shows the VGG16 model only makes 1 prediction count, ‘1’. The model fails to identify any count other than ‘1’. For this particular test dataset, most video files only contained one count of animals in the video, which explains the high precision score. If the dataset contains counts other than the value of ‘1’, then the metrics scores will decrease even further.

4.3 – YOLOv8 Model Performance on Test Data

Animal Classification, Count and Action Recognition Metrics

Table 5: Performance metrics for the YOLOv8 model

Task	Accuracy	Precision	Recall	F1 Score
Animal Classification	0.51	0.27	0.91	0.21
Animal Count	0.10	0.36	0.45	0.02
Action Recognition	0.09	1	0.71	0.71

Note: The metrics are calculated only on correctly classified animals for the count and behaviour scores. If the model did not correctly classify the animal, the count is not included in the metric calculations. As mentioned in the Method section, the behaviour metrics only used the ground true labels.

The YOLOv8 model has performance much better than the VGG16 model, with an accuracy of 51%. The model achieved a recall score of 91%, which illustrates that this model can detect multiple animals within the videos, but with a precision of 27%, many detected animals are misclassified, which led to a low F1 score of 21%. For the counting task, the YOLOv8 model achieved an accuracy of 10%, again an improvement over the VGG16. However, an F1 score of 2% shows that the model struggles to correctly identify the number of animals seen in the video. The action recognition for YOLOv8 produced an accuracy of 9%. This is because of the use of human labels rather than animal action labels. The 100% precision is achieved by using the ground true label and calculating the metric score only where the YOLOv8 has correctly identified the animals.

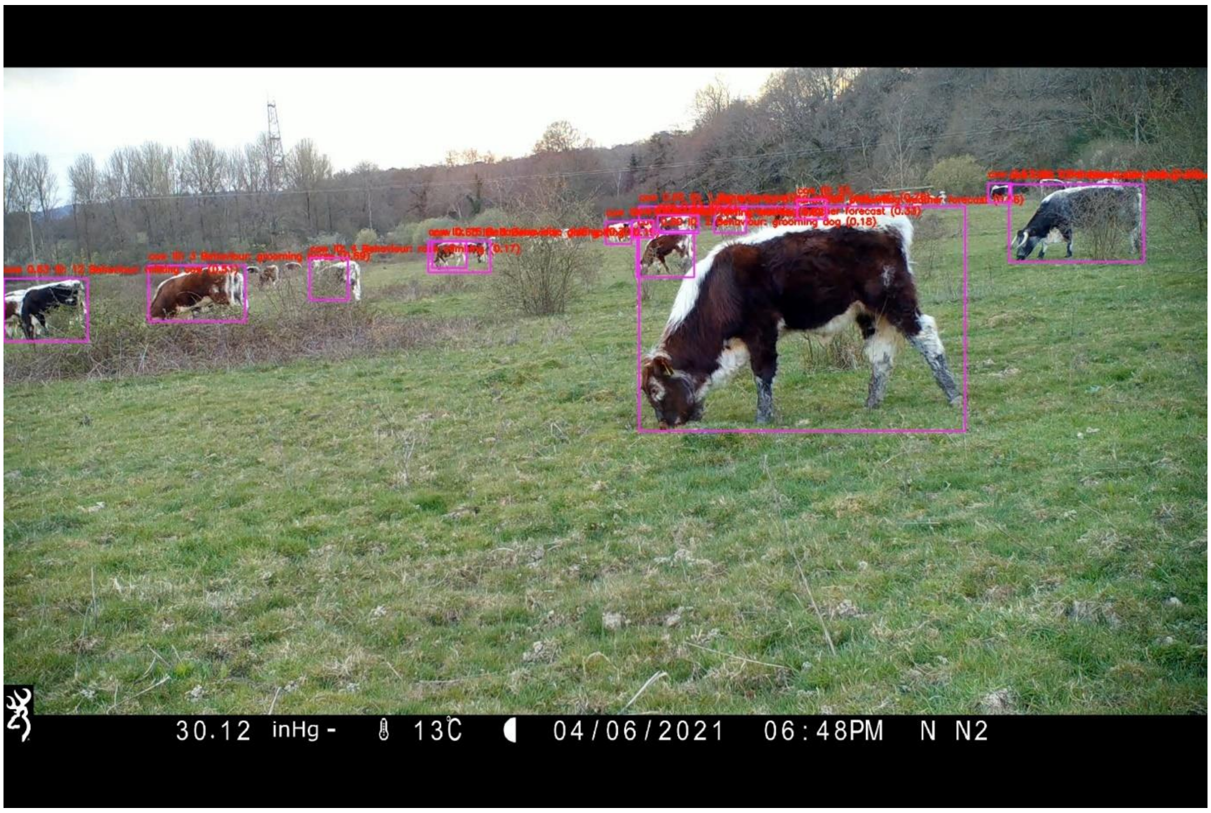
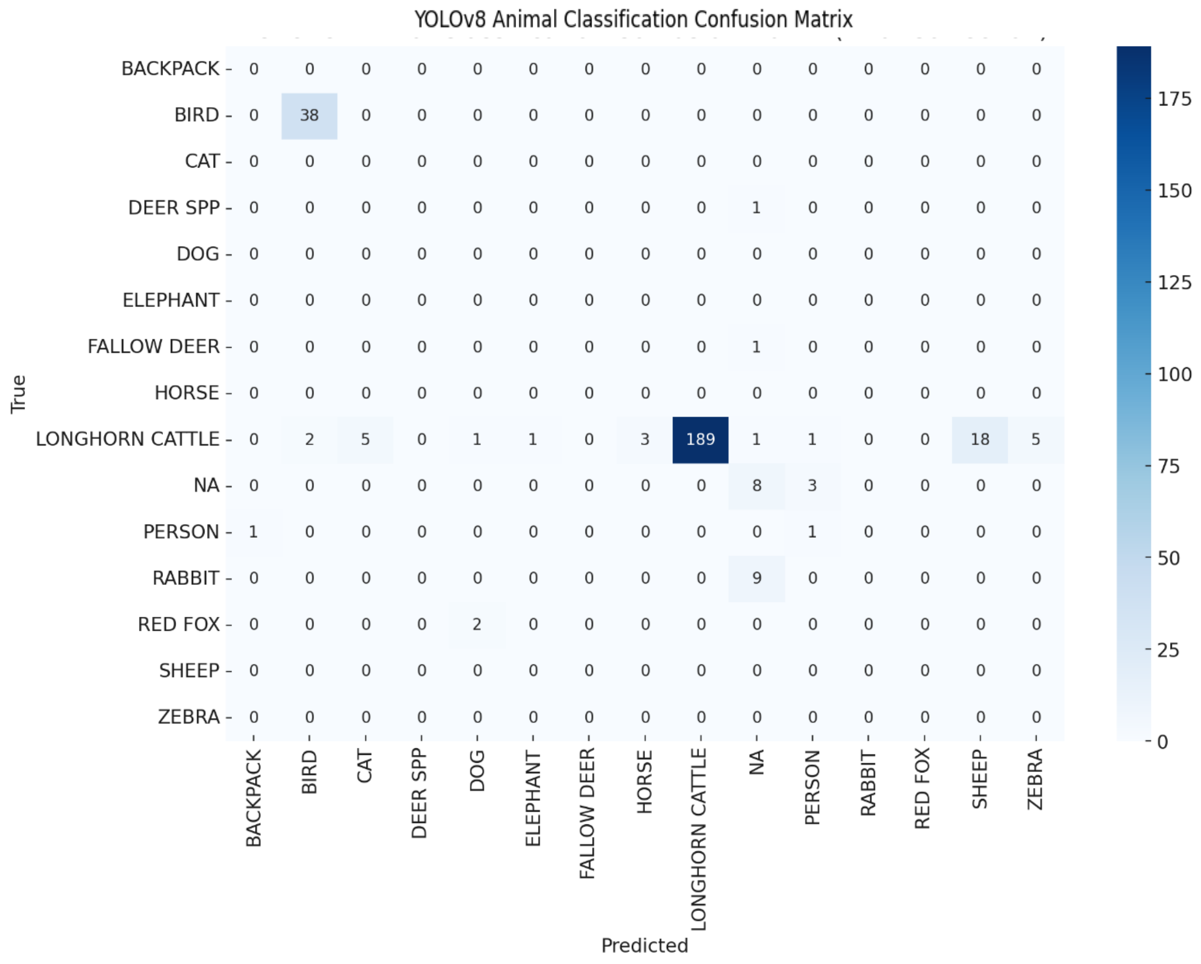


Figure 13: (N_N 3_2021_05_04-IMG_0109 video file.) YOLOv8 processed video. Detecting animals with a bounding box, including animal detected, animal ID, and the action of the animal. The video data provides clearer labels, which can be found on OneDrive.

The YOLOv8 model detects multiple objects well and classifies them as ‘cow’. The model also does well in keeping track of animals even when they are obscured by another animal, but it struggles to detect multiple animals seen in the far back of the video due to their smaller size. The action recognition labels, based on human labels, are not suited to accurately identify the behaviour as the model struggles to correctly identify the animal's behaviour.

Animal Classification Confusion Matrix

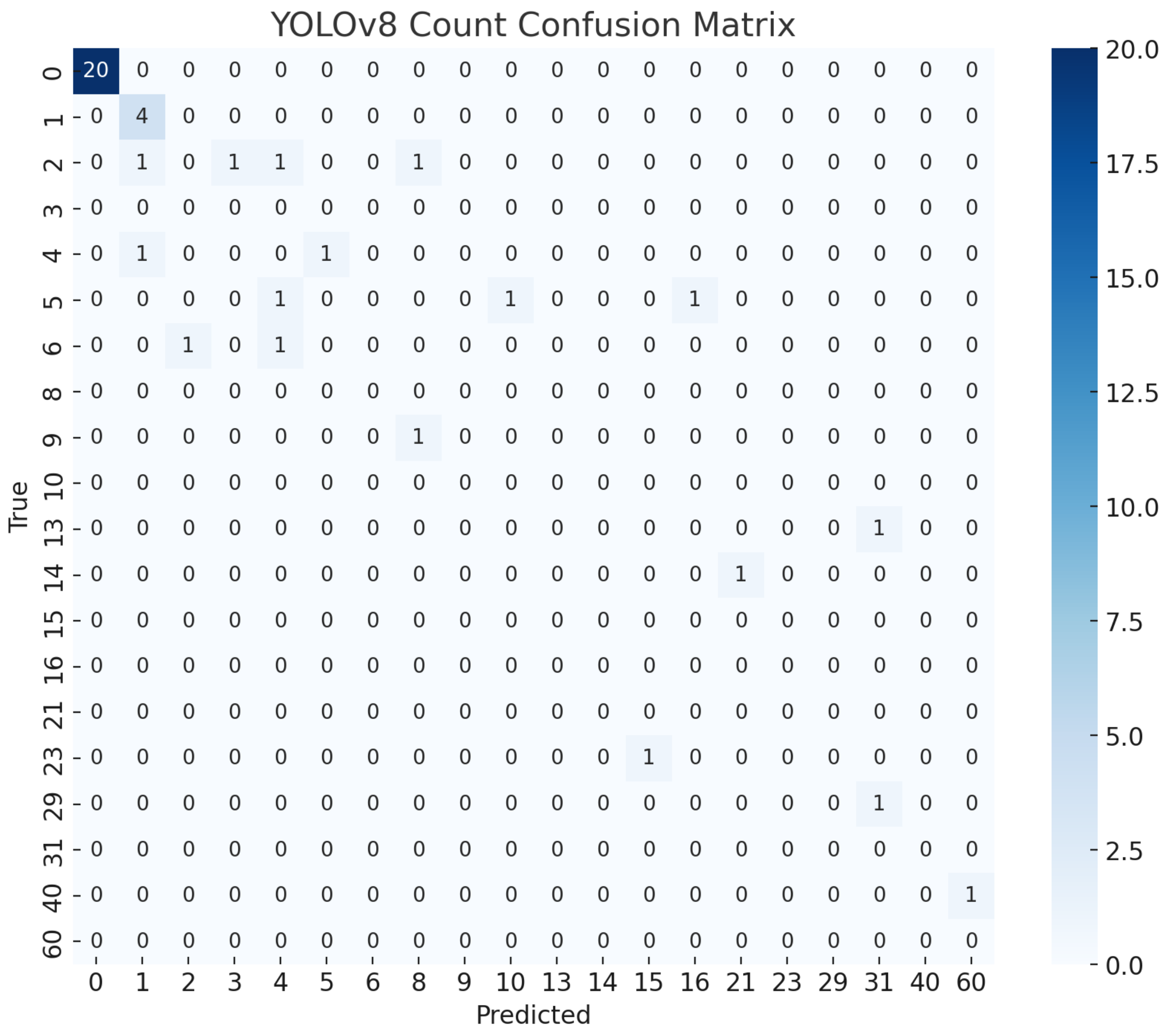


True \ Predicted	BACKPACK	BIRD	CAT	DEER SPP	DOG	ELEPHANT	FALLOW DEER	HORSE	LONGHORN CATTLE	NA	PERSON	RABBIT	RED FOX	SHEEP	ZEBRA
BACKPACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BIRD	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0
CAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DEER SPP	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
DOG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ELEPHANT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FALLOW DEER	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
HORSE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LONGHORN CATTLE	0	2	5	0	1	1	0	3	189	1	1	0	0	18	5
NA	0	0	0	0	0	0	0	0	0	8	3	0	0	0	0
PERSON	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
RABBIT	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0
RED FOX	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
SHEEP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ZEBRA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14: The confusion matrix for the YOLOv8 Animal Classification task.

The YOLOv8 has predicted 189 correct ‘Longhorn Cattle’ (cow) labels with only 7 misclassified, which is a precision of 96.4% for the ‘Longhorn Cattle’. Similarly, all ‘Birds’ labels have been correctly classed. However, the YOLOv8 model has identified animals that are not seen in test videos, such as ‘Zebra’ and ‘Elephant’, indicating some false positives. The model also struggled to identify ‘Rabbit’ in the videos. This is because the rabbits are small and only seen at night in the test video. The model also struggled to identify ‘Deer Spp’, which was also seen at night.

Animal Count Confusion Matrix

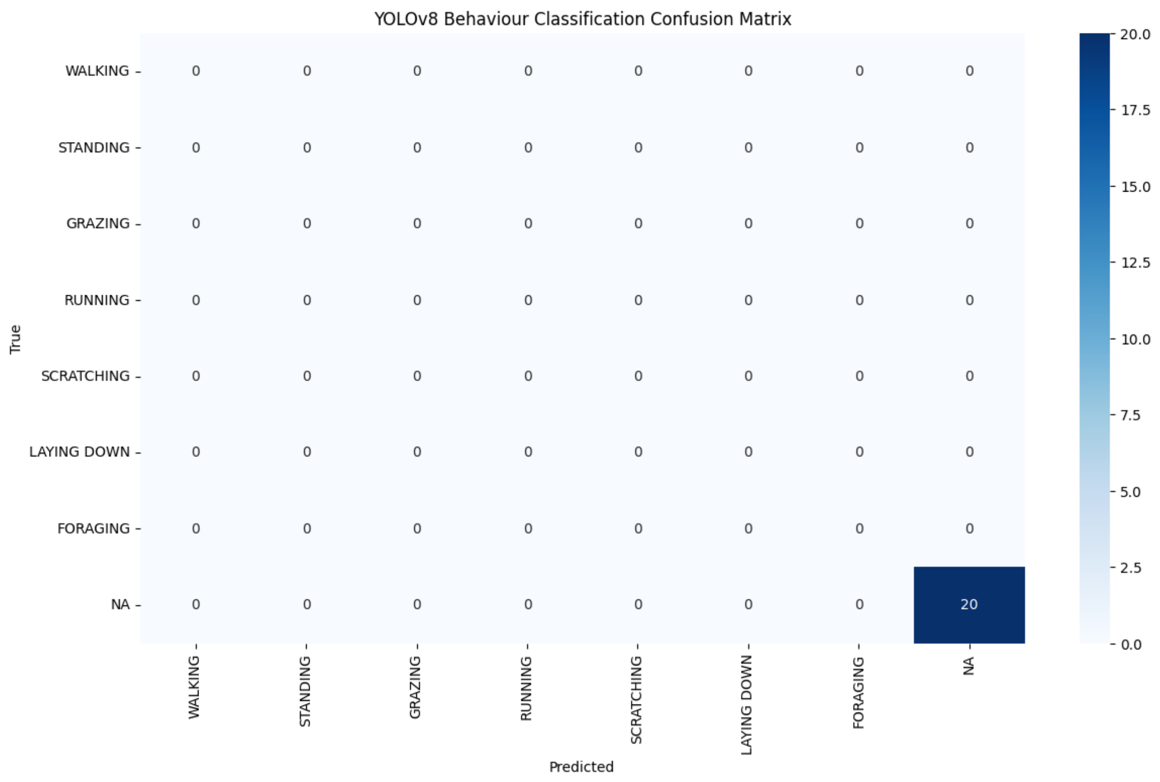


True \ Predicted	0	1	2	3	4	5	6	8	9	10	13	14	15	16	21	23	29	31	40	60
0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15: The confusion matrix for the YOLOv8 Animal Count task

The model was able to correctly identify all the NA counts, but in most cases, the model struggled. For example, in one case, the model predicted 60, but the right count was 40, and this discrepancy was seen throughout. Even though the YOLOv8 model can classify some animals correctly, the model struggles with the correct count of the animals as the model fails to identify some animals that are further at the back of the video, due to being small in size.

Animal Behaviour Confusion Matrix



True \ Predicted	WALKING	STANDING	GRAZING	RUNNING	SCRATCHING	LAYING DOWN	FORAGING	NA
WALKING	0	0	0	0	0	0	0	0
STANDING	0	0	0	0	0	0	0	0
GRAZING	0	0	0	0	0	0	0	0
RUNNING	0	0	0	0	0	0	0	0
SCRATCHING	0	0	0	0	0	0	0	0
LAYING DOWN	0	0	0	0	0	0	0	0
FORAGING	0	0	0	0	0	0	0	0
NA	0	0	0	0	0	0	0	20

Figure 16: The confusion matrix for the YOLOv8 with SlowFast Animal Action Recognition task.

Other than the ‘NA’, the model was not able to predict any labels associated with the ground truth label, which explains the precision of 100% seen in the metric scores. This data shows that even with human labels that contain running and walking, the model could not translate those labels to the animals, highlighting the need for animal-specific action labels.

As the YOLOv8 had overall higher accuracy than VGG16, it is set as the default model in the GUI.

4.4 – VGG16 and YOLOv8 Samples on Test Data

Here are some sample processed data from VGG16 and YOLOv8. This will show the differences in the model predictions.

Night Sample Data

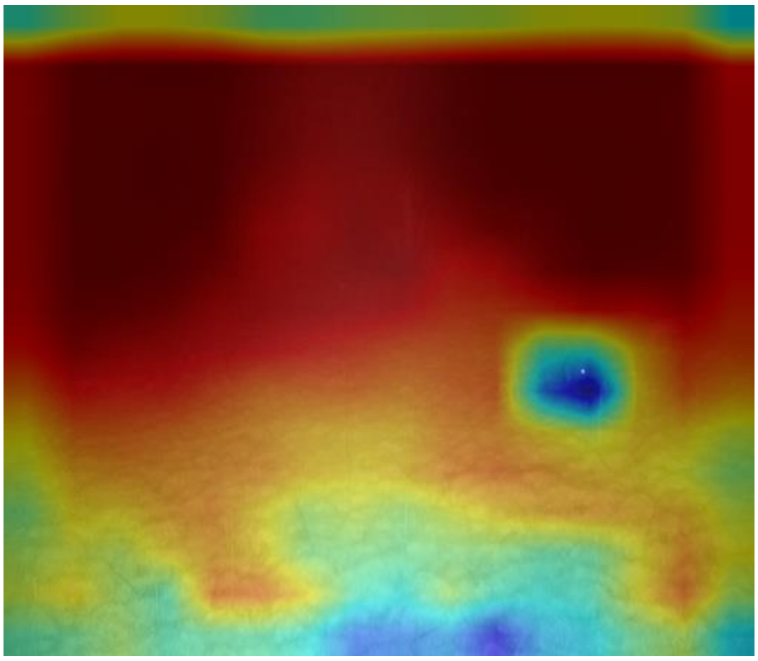


Figure 17: (N_N 3_2021_05_04-IMG_0006 at 10 seconds file). VGG16 processed an image that contained a rabbit with Grad-CAM++ applied.

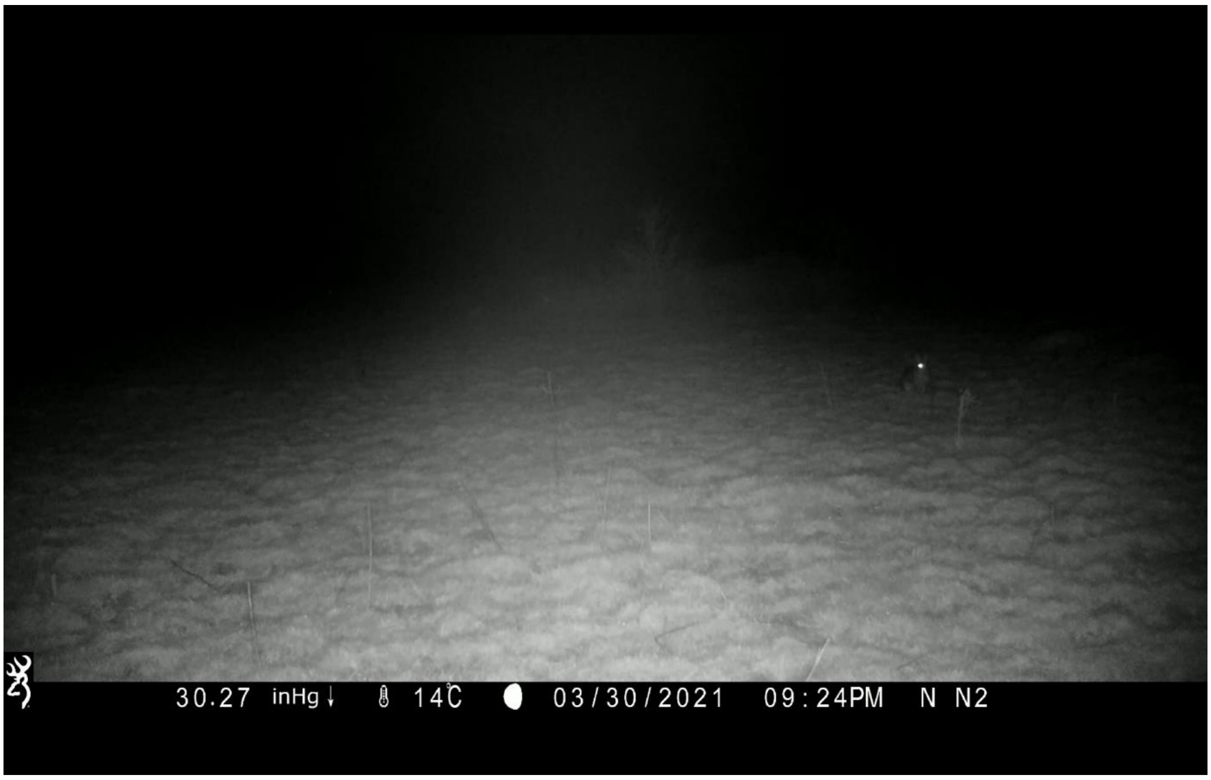


Figure 18: (N_N 3_2021_05_04-IMG_0006 at 10 seconds file). YOLOv8 processed test video which contains a rabbit.

Figures 17 and 18 show an example of a night camera trap video where a rabbit is seen. The VGG16 model could detect the rabbit, as illustrated by the heatmap. However, YOLOv8 could not detect any animals, likely due to the poor lighting conditions where the infrared light does not illuminate the entire frame. Table 6 shows the predictions from the models for this test video.

Table 6: VGG16 and YOLOv8 Prediction for Night Sample Data

Model	**Subject**	**Count**	**Behaviour**
VGG16	Rabbit	1	Not Applicable[2]
YOLOv8	NA	0	NA

Multiple Cow Sample

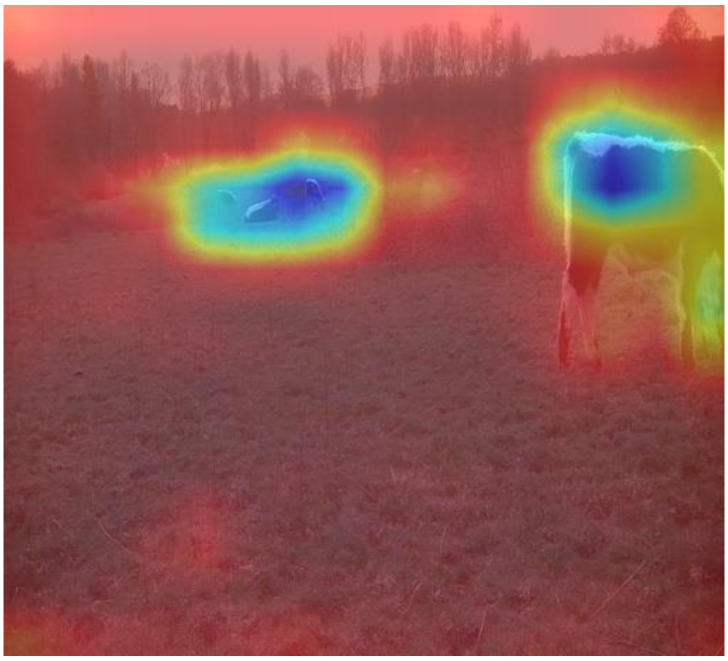


Figure 19: (N_N 3_2021_05_04-IMG_0211 at 20 seconds file). VGG16 processed an image that contained multiple longhorn cattle with Grad-CAM++ applied.

[2] VGG16 model does not have the capability to do action recognition in the videos.

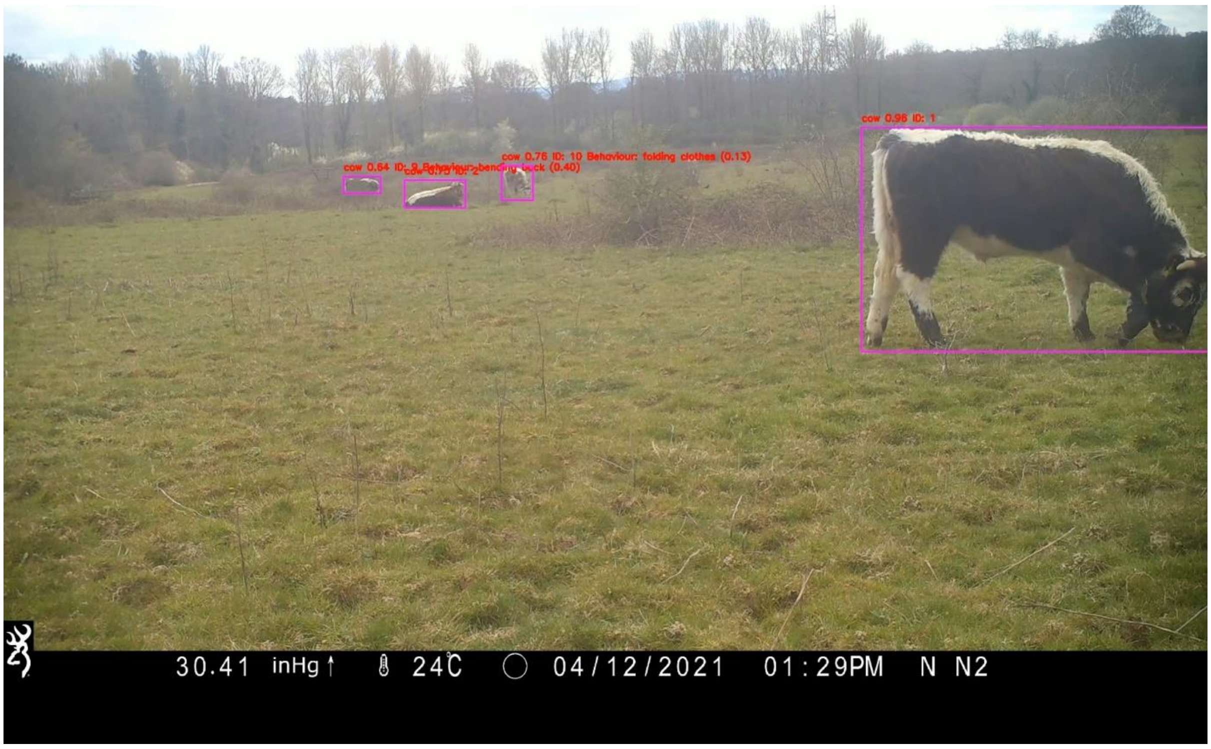

cow 0.96 ID: 1
cow 0.76 ID: 10 Behaviour: folding clothes (0.13)
cow 0.64 ID: 9 Behaviour: bending back (0.40)
cow 0.75 ID: 2
30.41 inHg
24℃
04/12/2021
01:29PM
N N2

Figure 20: (N_N 3_2021_05_04-IMG_0211 at 20 seconds file). YOLOv8 processed an image that contained multiple longhorn cattle.

VGG16 highlighted areas where the longhorn cattle are seen but missed other longhorn cattle seen in the background. However, the YOLOv8 model performed better than the VGG16, which detected multiple cows (longhorn cattle). Due to the bush covering the longhorn cattle seen in the background, the tracking was lost a few times and redetected, which led to duplicate counts for the same longhorn cattle, resulting in a wrong count for this test video. This suggests that further hyperparameter needs to be tuned, such as 'max_age', which needs to be increased.

Table 7: VGG16 and YOLOv8 Prediction for Multiple Cow Sample

Model	**Subject**	**Count**	**Behaviour**
VGG16	Longhorn Cattle	1	Not Applicable^2
YOLOv8	Cow (longhorn cattle)	10	See the CSV file

5 – Discussion

This project showcased the performance of VGG16 and YOLOv8 for animal classification, counting, and action recognition tasks. Both models had challenges dealing with each task, but they provided a fundamental understanding of the capabilities of each model, where YOLOv8 demonstrated clear advantages over VGG16.

The VGG16 model was fine-tuned and achieved excellent training performance but demonstrated moderate success in identifying different animals in the test data. The model particularly struggled with smaller objects and scenarios involving multiple animals, misclassifying small objects like rabbits. This is evident in the confusion matrix, where 'Longhorn Cattle' was misclassified despite having the highest amount of training data. This misclassification could be due to the overgeneralisation of features of 'Longhorn Cattle' and aggravated by the imbalance of training data where other class labels had a limited amount of data compared to 'Longhorn Cattle'. The counting task was another challenge for the VGG16 model. The Grad-CAM++ showed how the VGG16 model can miss out on animals across the video, especially the small animals (as seen with birds) and the animals seen in the background of the video, illustrating further fine-tuning is needed for the VGG16 model architecture, where the model can improve its accuracy in animal classification and counting other animals of the same class within the same video.

The YOLOv8 model performed better than VGG16, with the implementation of SlowFast for action recognition. The model's ability to detect multiple objects in the video was a significant advantage, where it achieved the highest recall for animal classification, illustrating the model's ability to identify various animals within a video. DeepSORT further helped with the tracking by keeping track of animals that get obscured or leave the video frame. However, the YOLOv8 model faced some challenges with false positives, detecting animals such as "Zebra" and "Elephant", which are not present in the test videos, resulting in a low precision score. This is due to using the COCO dataset and YOLOv8 model struggling to identify the animal correctly. With the counting task, YOLOv8 outperformed VGG16 but only marginally. The model miscounted many of the animals, usually not counting the animals seen in the background, due to the model being unable to detect them. It compensated for the scores where no animals were seen, which is good as it shows it is not detecting and counting an object that is not there. The YOLOv8 struggled to detect any animal at night or in foggy conditions as seen in Figure 18, affecting the accuracy in animal classification and count tasks. Using human labels for the action recognition task affected the metrics scores poorly, as the SlowFast model could not correctly label the behaviour of the animals except when there were no animals, which points out the need for animal-specific labels for this task.

6 – Future Work

As the YOLOv8 model performed well compared to VGG16, moving forward, incorporating the newer version of the YOLO models will help with overall performance as they will have higher detection performance. YOLOv9 (YOLOv9e variant) from Ultralytics achieves mAPval 50-95 of 55.6 compared to YOLOv8 (YOLOv8x variant) 53.9 (Ultralytics, 2024). This is a small improvement but can make a big difference in the accuracy of detecting the smaller animals and detecting animals in the dark.

To further improve the performance of the YOLO model, fine-tuning the model with custom data will be necessary. However, this requires the labelled data to be in bounding box annotation format. I highly advise the Department of Life Sciences and Dr Christopher Sandom to label the data with bounding box annotation. This will reduce false positives where the wrong animal, such as 'Zebra', is identified, as seen in the testing. For the action recognition task, SlowFast can also benefit from fine-tuning, as in this project, there were issues getting the animal labels from the Animal Kingdom Dataset to work due to library installation limitations and using human labels did not help with correct prediction of the animal behaviour, therefore training the SlowFast model on the custom animal label will allow for better behaviour predictions. All these suggestions will help to improve animal classification, count, and action recognition results.

This project sought ways to automate methods for labelling animals, as seen in videos. While there was some success, further improvement is essential to achieve higher accuracy in detecting the smaller animals, animals seen in the far back of the videos, and animals seen in poor lighting conditions such as at night. Fine-tuning the YOLO model is an important step in achieving higher accuracy. Additional refinement is also required for the action recognition task, where the current use of human action labels has resulted in poor results. Therefore, it is necessary to have specific animal action labels, which will help classify animal behaviour effectively. Integrating a newer version of YOLO for improved accuracy and efficiency, automation of wildlife video analysis will be possible with a small margin of errors.

.

Reference

Abbas, S.M. and Singh, Dr.S.N. (2018). *Region-based Object Detection and Classification using Faster R-CNN*. [online] IEEE Xplore. doi:https://doi.org/10.1109/CIACT.2018.8480413.

Bai, T. (2023). Multiple Object Tracking Based on YOLOv5 and Optimized DeepSORT Algorithm. *Journal of Physics Conference Series*, 2547(1), pp.012022–012022. doi:https://doi.org/10.1088/1742-6596/2547/1/012022.

Chattopadhay, A., Sarkar, A., Howlader, P. and Balasubramanian, V.N. (2018). Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. pp.839–847. doi:https://doi.org/10.1109/WACV.2018.00097.

Demirkaya, A., Chen, J. and Oymak, S. (2020). Exploring the Role of Loss Functions in Multiclass Classification. *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. doi:https://doi.org/10.1109/ciss48834.2020.1570627167.

Feichtenhofer, C., Fan, H., Malik, J. and He, K. (2019). SlowFast Networks for Video Recognition. *arXiv:1812.03982 [cs]*. [online] Available at: https://arxiv.org/abs/1812.03982.

Giraddi, S., Seeri, S., Hiremath, P.S. and G.N, J. (2020). *Flower Classification using Deep Learning models*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICSTCEE49637.2020.9277041.

Hassan, A., Elgabry, A. and Elsayed Hemayed (2021). Enhanced Dynamic Sign Language Recognition using SlowFast Networks. doi:https://doi.org/10.1109/icenco49852.2021.9698904.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [online] pp.770–778. doi:https://doi.org/10.1109/cvpr.2016.90.

Inbaraj, X.A., Villavicencio, C., Macrohon, J.J., Jeng, J.-H. and Hsieh, J.-G. (2021). Object Identification and Localization Using Grad-CAM++ with Mask Regional Convolution Neural Network. *Electronics*, 10(13), p.1541. doi:https://doi.org/10.3390/electronics10131541.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M. and Zisserman, A. (2017). The Kinetics Human Action Video Dataset. *arXiv:1705.06950 [cs]*. [online] Available at: https://arxiv.org/abs/1705.06950.

Keskar, N.S. and Socher, R. (2017). Improving Generalization Performance by Switching from Adam to SGD. *arXiv:1712.07628 [cs, math]*. [online] Available at: https://arxiv.org/abs/1712.07628.

Kumar, S., Vishal, Sharma, P. and Pal, N. (2021). *Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICAIS50930.2021.9395971.

Le, N.A., Moon, J., Lowe, C.G., Kim, H.-I. and Choi, S.-I. (2022). An Automated Framework Based on Deep Learning for Shark Recognition. *Journal of Marine Science and Engineering*, [online] 10(7), p.942. doi:https://doi.org/10.3390/jmse10070942.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Lawrence, Z.C. and Dollár, P. (2014). *Microsoft COCO: Common Objects in Context*. [online] arXiv.org. Available at: https://arxiv.org/abs/1405.0312.

Mahardi, Wang, I-Hung., Lee, K.-C. and Chang, S.-L. (2020). Images Classification of Dogs and Cats using Fine-Tuned VGG Models. doi:https://doi.org/10.1109/ecice50847.2020.9301918.

Martha, G.W., Mwangi, W., Aramvith, S. and Rimiru, R. (2023). Comparing deep learning object detection methods for real time cow detection. pp.1187–1192. doi:https://doi.org/10.1109/TENCON58879.2023.10322403.

matplotlib (2024). *Pyplot tutorial — Matplotlib 3.8.0 documentation*. [online] matplotlib.org. Available at: https://matplotlib.org/stable/tutorials/pyplot.html.

McKinsey & Company (2022). *The state of AI in 2022--and a half decade in review | mckinsey*. [online] www.mckinsey.com. Available at: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review.

Nguyen, N.G., Phan, D., Lumbanraja, F.R., Faisal, M.R., Abapihi, B., Purnama, B., Delimayanti, M.K., Mahmudah, K.R., Kubo, M. and Satou, K. (2019). Applying Deep Learning Models to Mouse Behavior Recognition. *Journal of Biomedical Science and Engineering*, 12(02), pp.183–196. doi:https://doi.org/10.4236/jbise.2019.122012.

pandas (2024). *pandas.DataFrame — pandas 1.2.4 documentation*. [online] pandas.pydata.org. Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L. and Bai, J. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. [online] arXiv.org. Available at: https://arxiv.org/abs/1912.01703.

PyTorch (2024). *torchvision.transforms — Torchvision master documentation*. [online] pytorch.org. Available at: https://pytorch.org/vision/stable/transforms.html.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, [online] 12, pp.145–151. doi:https://doi.org/10.1016/S0893-6080(98)00116-6.

scikit-learn (2024). *sklearn.model_selection.train_test_split — scikit-learn 0.20.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

seaborn (2024). *seaborn.heatmap — seaborn 0.10.1 documentation*. [online] seaborn.pydata.org. Available at: https://seaborn.pydata.org/generated/seaborn.heatmap.html.

Simonyan, K. and Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.1556.

Tan, M., Chao, W., Cheng, J.-K., Zhou, M., Ma, Y., Jiang, X., Ge, J., Yu, L. and Feng, L. (2022). Animal Detection and Classification from Camera Trap Images Using Different Mainstream Object Detection Architectures. *Animals*, [online] 12(15), p.1976. doi:https://doi.org/10.3390/ani12151976.

Thanapol, P., Lavangnananda, K., Bouvry, P., Pinel, F. and Leprévost, F. (2020). *Reducing Overfitting and Improving Generalization in Training Convolutional Neural Network (CNN) under Limited Sample Sizes in Image Recognition*. [online] IEEE Xplore. doi:https://doi.org/10.1109/InCIT50588.2020.9310787.

Ultralytics (2024a). *YOLOv8*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/models/yolov8/#usage-examples.

Ultralytics (2024b). *YOLOv9*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/models/yolov9/#performance-on-ms-coco-dataset.

Uswatun Khasanah, C., Utami, E. and Raharjo, S. (2020). *Implementation of Data Augmentation Using Convolutional Neural Network for Batik Classification*. [online] IEEE Xplore. doi:https://doi.org/10.1109/CITSM50537.2020.9268890.

Vitaly Bushaev (2017). *Stochastic Gradient Descent with momentum*. [online] Medium. Available at: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d.

Wearn, O.R. and Glover-Kapfer, P. (2019). Snap happy: camera traps are an effective sampling tool when compared with alternative methods. *Royal Society Open Science*, 6(3), p.181748. doi:https://doi.org/10.1098/rsos.181748.

Wojke, N., Bewley, A. and Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. *arXiv:1703.07402 [cs]*. [online] Available at: https://arxiv.org/abs/1703.07402.

Wu, T., Zhong, S., Chen, H. and Geng, X. (2023). Research on the Method of Counting Wheat Ears via Video Based on Improved YOLOv7 and DeepSort. *Sensors*, 23(10), pp.4880–4880. doi:https://doi.org/10.3390/s23104880.

Ye, H., Han, H., Zhu, L. and Duan, Q. (2019). Vegetable Pest Image Recognition Method Based on Improved VGG Convolution Neural Network. *Journal of Physics: Conference Series*, 1237, p.032018. doi:https://doi.org/10.1088/1742-6596/1237/3/032018.

Zhong, J., Li, M., Qin, J., Cui, Y., Yang, K. and Zhang, H.Y. (2022). REAL-TIME MARINE ANIMAL DETECTION USING YOLO-BASED DEEP LEARNING NETWORKS IN THE CORAL REEF ECOSYSTEM. XLVI-3/W1-2022, pp.301–306. doi:https://doi.org/10.5194/isprs-archives-xlvi-3-w1-2022-301-2022.

Image Reference

Dorfer, T.A. (2023). *Enhanced Object Detection: How To Effectively Implement YOLOv8*. [online] Medium. Available at: https://towardsdatascience.com/enhanced-object-detection-how-to-effectively-implement-yolov8-afd1bf6132ae.

Ultralytics (2024). *YOLOv8*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/models/yolov8/.

The diagrams of Figures 2 and 5 were created using Microsoft Word Icons

Code Reference

VGG16

GeeksforGeeks. (2024). *Python | Program to extract frames using OpenCV*. [online] Available at: https://www.geeksforgeeks.org/python-program-extract-frames-using-opencv/.

jacobgil (2024). *pytorch-grad-cam/tutorials/Class Activation Maps for Semantic Segmentation.ipynb at master · jacobgil/pytorch-grad-cam*. [online] GitHub. Available at: https://github.com/jacobgil/pytorch-grad-cam/blob/master/tutorials/Class%20Activation%20Maps%20for%20Semantic%20Segmentation.ipynb.

Pakhale, T. (2021). *Transfer Learning using VGG16 in Pytorch*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/transfer-learning-using-vgg16-in-pytorch/.

pytorch (2024). *Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.7.0 documentation*. [online] pytorch.org. Available at: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html.

YOLOv8 & SlowFast

Mosesdaudu (2023). *Yolov8, Sort and DeepSort - Mosesdaudu - Medium*. [online] Medium. Available at: https://medium.com/@mosesdaudu001/yolov8-sort-and-deepsort-9aa7059cf09c.

Pareek, N. (2023). *Using DeepSORT Object Tracker with YOLOv5*. [online] Kaggle.com. Available at: https://www.kaggle.com/code/nityampareek/using-deepsort-object-tracker-with-yolov5.

pytorch (2019). *hub/facebookresearch_pytorchvideo_slowfast.md at master · pytorch/hub*. [online] GitHub. Available at: https://github.com/pytorch/hub/blob/master/facebookresearch_pytorchvideo_slowfast.md.

PyTorch. (2024). *SlowFast*. [online] Available at: https://pytorch.org/hub/facebookresearch_pytorchvideo_slowfast/.

Bibliography

Ruiz, I., Porzi, L., Bulo, S.R., Kontschieder, P. and Serrat, J. (2021). Weakly Supervised Multi-Object Tracking and Segmentation. *Thecvf.com*, [online] pp.125–133. Available at: https://openaccess.thecvf.com/content/WACV2021W/AVV/html/Ruiz_Weakly_Supervised_Multi-Object_Tracking_and_Segmentation_WACVW_2021_paper.html.

Tsang, S.-H. (2021). *Review — Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks (Weakly….* [online] The Startup. Available at: https://medium.com/swlh/review-grad-cam-improved-visual-explanations-for-deep-convolutional-networks-weakly-760264e66bc6.

Wang, Y., Zhang, J., Kan, M., Shan, S. and Chen, X. (2020). *Self-supervised Equivariant Attention Mechanism for Weakly Supervised Semantic Segmentation*. [online] Available at: https://openaccess.thecvf.com/content_CVPR_2020/papers/Wang_Self-Supervised_Equivariant_Attention_Mechanism_for_Weakly_Supervised_Semantic_Segmentation_CVPR_2020_paper.pdf [Accessed 9 Sep. 2024].

Zubair (2024). *Grad-CAM Overview - Zubair - Medium*. [online] Medium. Available at: https://medium.com/@zakhtar2020/grad-cam-overview-f8f84edebe9d#:~:text=Class%2DDiscriminative%20Localization%3A%20Grad%2D.

Appendix A

YOLOv8 COCO Dataset Labels

1. person
2. bicycle
3. car
4. motorcycle
5. airplane
6. bus
7. train
8. truck
9. boat
10. traffic light
11. fire hydrant
12. stop sign
13. parking meter
14. bench
15. bird
16. cat
17. dog
18. horse
19. sheep
20. cow
21. elephant
22. bear
23. zebra
24. giraffe
25. backpack
26. umbrella
27. handbag
28. tie
29. suitcase
30. frisbee
31. skis
32. snowboard
33. sports ball
34. kite
35. baseball bat
36. baseball glove
37. skateboard
38. surfboard
39. tennis racket
40. bottle
41. wine glass
42. cup
43. fork
44. knife
45. spoon
46. bowl
47. banana
48. apple
49. sandwich
50. orange
51. broccoli
52. carrot
53. hot dog
54. pizza
55. donut
56. cake
57. chair
58. couch
59. potted plant
60. bed
61. dining table
62. toilet
63. TV
64. laptop
65. mouse
66. remote
67. keyboard
68. cell phone
69. microwave
70. oven
71. toaster
72. sink
73. refrigerator
74. book
75. clock
76. vase
77. scissors
78. teddy bear
79. hair drier
80. toothbrush