



Automated detection of animals in camera traps using deep learning in South Africa

S Mamapule

 **orcid.org 0000-0002-6927-2157**

Dissertation accepted in fulfilment of the requirements for
the degree Master of Science in Computer Science at
the North-West University

Supervisor: Prof BM Esiefarienrhe

Co-supervisor: Dr IC Obagbuwa

Graduation: July 2024

The bottom half of the cover features abstract, flowing blue and teal wave patterns that complement the top section's design.

Declaration

I, **Siyabonga Mamapule**, student number **31205879**, declare that the study ***Automated detection and counting of animals in camera traps using deep learning in South Africa***, submitted to North-West University, Mafikeng Campus for an MSc in Computer Science, and Information Systems study, is my work and has not been submitted before to any institution. All the information used has been acknowledged both in the text and in references.



Signature.....

Date... 01/12/2023

Siyabonga Mamapule

Approval



Signature.....

Date.....01/12/2023.....

Supervisor: Prof. M. B. Esiefarienrhe

Acknowledgements

First and foremost, I would like to appreciate the support from the School of Computer Science and Information Technology at North-West University. I am grateful for allowing me to work under your guidance on my Master of Science in Computer Science.

I would like to acknowledge and express my gratitude to my research supervisor, Prof. BM Esiefarienrhe, for his wise counsel and compassion during the entire process of the study, from the inception of the research proposal through the preparation of the research report.

I would like to thank my co-supervisor, Dr Ibidun Obagbuwa, for her advice and support in completing both the experimentation and research report during this research journey.

Abstract

In the field of ecology, counting of animals to estimate population size and types of species is important for wildlife conservation. This includes analysing massive volumes of image, video or audio/acoustic data and traditional counting techniques. Automating the process of identifying, classifying, and counting animals is helpful as it phases out the tedious manual tasks of counting and labelling. The intention of this dissertation was to address manual identification and counting methods of images, by implementing an automated solution using computer vision and deep learning. This study applied a classification model to classify species, and trained an object detection model using deep convolutional neural networks, to automatically identify and determine the count of four mammal species in 1,504 images extracted from camera traps. The image classification model reported a classification accuracy of 99%, and the YOLOv8 object detection model automatically detected buffalo, elephant, rhino, and zebra school mean average precision 50 of 89%, and mean average precision 50-95 of 72.2%, and provided an accurate count over all animal classes. The results of the study showed that the application of computer vision and deep learning methods on data-scarce and data-enriched scenarios respectively, can save biologists and ecologists an enormous amount of time used on time-consuming manual methods of analysis and counting. Additionally, these automated identification techniques can contribute towards enhancing wildlife conservation and informing future studies.

Keywords : Biodiversity Monitoring, Camera traps, Image classification, Deep Learning

Table of Contents

Chapter 1: Introduction.....	1
1.0 Background information to the study	1
1.1 Statement of the problem.....	2
1.2 Aim of the study.....	3
1.3 Objectives.....	3
1.4 Research Question(s).....	4
1.5 Motivation	4
1.6 Contributions	5
1.7 Location of the Study	5
1.8 Limitations	6
1.9 Overview of the Thesis	6
1.10 Conclusion of the Chapter	8
Chapter 2: Literature Review	9
2.1 Introduction.....	9
2.1.1 Overview of the Chapter.....	10
2.2 Importance of ecological monitoring for wildlife conservation.....	11
2.3 Camera trap Images.....	12
2.4 Utilising Machine Learning in the field of Computer Vision.....	13
2.4.1 Feature Extraction	15
2.4.2 Classification	16
2.5 Deep Learning	17
2.5.1 Neuron.....	19
2.5.2 Fundamental Neural Network.....	20
2.5.3 Convolutional Neural Network.....	22
2.5.4 Challenges in Deep Learning	24
2.6 Deep Learning methods for wildlife conservation.....	25
2.7 Research Gap	26
2.8 Conclusion of the Chapter	27
Chapter 3: Methodology	29
3.1 Introduction.....	29
3.1.1 Overview of the chapter.....	30

3.2	Theoretical framework	30
3.2.1	Awareness of problem	31
3.2.2	Suggestion and Development	31
3.2.3	Evaluation	31
3.2.4	Conclusion	32
3.3	Research Design	32
3.3.1	Data partitioning.....	33
3.3.2	Data Augmentation.....	33
3.3.3	Defining model architecture.	34
3.3.4	Deep learning toolkit	34
3.4	Data collection	35
3.4.1	South African Wildlife Dataset.....	35
3.4.2	Transfer Learning	36
3.5	Analysis techniques	36
3.5.1	Convolutional Layer.....	36
3.5.2	Pooling Layer	37
3.5.3	Fully Connected Layer.....	38
3.5.4	Activation functions	38
3.5.5	Regularisation	39
3.5.6	Optimisation Algorithm.....	41
3.6	Model Evaluation	43
3.6.1	Classification	43
3.6.2	Object detection.....	46
3.7	Conclusion of the chapter	49
Chapter 4:	Data and Algorithm identification.....	50
4.1	Introduction.....	50
4.1.2	Overview of the chapter.....	51
4.2	Data Summary.....	51
4.3	Partitioning the data into training, validation, and test sets.....	53
4.3.1	Dataset division for image classification	54
4.3.2	Dataset partitioning for object detection.....	55
4.4	Data Augmentation	58

4.4.1	Augmentation using Torchvision.....	58
4.4.2	Augmentation using Keras.....	60
4.5	Configuration of the CNN architectures.....	62
4.6	Image Classification.....	64
4.6.1	Alexnet Architecture	64
4.6.2	Residual Network (ResNet) Architecture	66
4.6.3	VGG16 Architecture	68
4.6.4	CNN model training for image classification.	70
4.7	Object Detection	72
4.7.1	The YOLOv8 algorithm	72
4.7.2	Bounding Boxes for training set	74
4.7.3	Hyperparameters	75
4.7.4	YOLOv8 Model Assessment.....	76
4.8	Computational Resources.....	77
4.9	Conclusion.....	78
Chapter 5: Results and Discussions		79
5.1	Image Classification results.....	79
5.1.1	Evaluation of image classification model performance.....	80
5.1.2	Alexnet results.....	82
5.1.3	Results of ResNet models.....	84
5.1.4	VGG16 results.....	89
5.2	Object detection results	93
5.2.1	Bounding box analysis.....	93
5.2.2	YOLOv8 Model training output	95
5.2.3	YOLOv8 Model Validation	97
5.2.4	YOLOv8 Model Test.....	101
5.3	YOLOv8 Sample predictions.....	104
5.4	Conclusion.....	111
Chapter 6: Conclusion		113
6.1	Conclusion.....	113
6.2	Answers to the research questions	115
6.3	Future works.....	116

References	118
Appendix A - Codes / Data Collection Tools	127
Appendix B - Ethics Certificate	128
Appendix C - Editing Certificate.....	129
Appendix D - Similarity Report.....	130

List of Tables

Table 1: Performs of CNN architectures on Imagenet.	23
Table 2: Image distribution according to the train, validation, and test data splits .	54
Table 3: Label information of a text file.....	55
Table 4: YOLO annotation classes that represent the animal species.	56
Table 5: Data partitioning summary for object detection	58
Table 6: Summary of augmentation parameters used for Torchvision.	59
Table 7: Summary of parameters used in Keras augmentation.....	61
Table 8: Parameters for Alexnet architecture	65
Table 9: Parameter information of ResNets	67
Table 10: Configuration of the VGG16 architecture.....	69
Table 11: Hyperparameter configurations for classification experiments	71
Table 12: Hyperparameter configuration for the object detection model	76
Table 13: Computational Resources	77
Table 14: Overall performance of CNN classifiers.....	92
Table 15: Evaluation of the model's performance on the validation set.....	97
Table 16: Overall model performance on the test set.....	101
Table 17: Detailed model performance for each class	101

List of Figures

Figure 1: Overview of the Study	6
Figure 2: Overview of Chapter 2.....	10
Figure 3: A computer vision channel used for identifying animal species.....	15
Figure 4: Different structures of biological and artificial neural networks.	18
Figure 5: Basic artificial neuron [44]	19
Figure 6: Fundamental structure of a Neural Network.....	20
Figure 7: Overview of Chapter 3.....	30
Figure 8: Design Science Research Activities of the First Cycle: [64].	30
Figure 9: Overview of the South African Wildlife Dataset	35
Figure 10: Representation of Average and Max pooling methods [70].....	37
Figure 11: Designs of different activation functions [72]	39
Figure 12: The Dropout approach [74]	40
Figure 13: CNN architecture for image classification.....	43
Figure 14: Structure of Confusion matrix [82]	44
Figure 15: Depiction of IoU across different thresholds [87]	48
Figure 16: Overview of Chapter 4.....	51
Figure 17: Animal Classes. 17.A Buffalo class, 17.B represents the elephant class, 17.C depicts the rhino class, and 17.D shows the zebra class.....	52
Figure 18: Summary of Dataset.....	53
Figure 19: Data Summary for image classification after partitioning.....	55
Figure 20: Bounding Box images. A shows animal images without bounding boxes and 20.B shows an image of an object with the corresponding bounding boxes. .	57
Figure 21: Output of the Torchvision augmentation technique	60
Figure 22: Output of the Keras ImageDataGenerator.....	61
Figure 23: Species identification structure of the CNN architecture	62
Figure 24: Configuration of a CNN architecture.....	63
Figure 25: Alexnet architecture.....	64
Figure 26: Residual block [49]	66
Figure 27: Structure of VGG16 architecture [52]	68
Figure 28: Structure of the YOLOv8 object detector.....	73
Figure 29: Training batch 1.....	74
Figure 30: Training batch 2.....	75
Figure 31: Alexnet results: 31.A Accuracy curve, 31.B Loss curve, and 31.C Confusion matrix	82
Figure 32: Alexnet prediction results on rhino class	83
Figure 33: ResNet-18 results: 33.A Accuracy curve, 33.B Loss curve, and 33.C Confusion matrix	84
Figure 34: ResNet-34 results: 34.A Accuracy curve, 34.B Loss curve, and 34.C Confusion matrix	85
Figure 35: ResNet-50 results: 35.A Accuracy curve, 35.B Loss curve, and 35.C Confusion matrix	86

Figure 36: ResNet-18 sample prediction results.....	87
Figure 37: ResNet-34 sample prediction results.....	88
Figure 38: ResNet-50 sample prediction results.....	89
Figure 39: VGG16 model training. 39.A depicts the training vs validation accuracy curve, the training and validation loss in 39.B, and the confusion matrix was illustrated in Figure 39.C.	90
Figure 40: Prediction of VGG16 on testing sample	91
Figure 41: Object distribution within the dataset.....	94
Figure 42: YOLOv8 training progress.....	95
Figure 43: Epoch visualisation during training.....	97
Figure 44: Precision-Confidence and Recall-Confidence curves on validation set.	99
Figure 45: F1-Confidence curve and the Confusion Matrix on the validation set.	100
Figure 46: Precision-Confidence and Recall-Confidence curves on the test set .	102
Figure 47: F1-Confidence curve and Confusion Matrix on the test set.....	103
Figure 48: YOLOv8 detection on buffalo class	104
Figure 49: Detection results of buffalo when appearing small and blurry.	105
Figure 50: YOLOv8 detections on elephant class.	106
Figure 51: Detection results of elephants appearing with other animal classes and animal count.	107
Figure 52: YOLOv8 detection on rhino class.....	108
Figure 53: Detections of rhino appearing in images that consists different animal classes.	108
Figure 54: YOLOv8 detection on the zebra class.....	109
Figure 55: Detection of zebras in different images that show multiple animal species.	110

List of Outputs

1. Siyabonga Mamapule, Ibidun Christiana Obagbuwa, Michael Esiefarienrhe
“Automatic identification and counting of South Africa wildlife species in camera traps using deep learning Techniques” (Manuscript in preparation) - Research Article
2. Siyabonga Mamapule, Ibidun Christiana Obagbuwa, Michael Esiefarienrhe
“Automatic Species Identification on camera trap images using Deep Learning approach” (Manuscript in preparation) -Systematic review article.

Chapter 1: Introduction

1.0 Background information to the study

Camera traps are commonly used in ecological studies to analyse animal behaviour and calculate densities, relative abundance, or habitation in single and multi-species research [1]. Images obtained from camera traps have become increasingly significant and necessary in ecological management studies due to their ability to closely monitor animal activity and behaviour [2]. In addition, camera traps can record additional image information that includes specific animal features such as age, sex, or stripe pattern. Identifying specific features, for instance, can enable assessment of species abundance or density using geographic capture-recapture techniques [3].

South Africa, making up only 0.8% of the planet's terrestrial surface, is among the 18 "megadiverse countries". According to Conservation International, these nations harbour most of the planet's endemic species [4]. South Africa has an extensive and diverse biodiversity, and a long history of wildlife conservation management. Monitoring biodiversity and ecosystems is important for South Africa – a biologically diverse nation with various mammal species that are endangered by anthropogenic activities that include poaching, hunting, and climate change. Like other African nations, South Africa lacks information on conservation of various species, and 17% of the mammal species are endangered and threatened with extinction [5]. As a result, the nation serves as a testing ground for various hypotheses.

In response to the above challenges, the network Snapshot Safari (www.snapshotsafari.org) was formed. Snapshot Safari is an extensive international camera trap network that was created to explore and track the biodiversity and ecology processes of animals in eastern and southern Africa [5]. Snapshot Safari is one of the biggest camera trapping networks in the globe. However, there is little

information available on the conservation status of several species, and some South African regions lack core biodiversity data required to evaluate the success of ongoing conservation [6]. Systematic long-term monitoring of various species and animal populations is essential for making informed decisions. However, the amount of data collected by camera traps has increased much to a point that it requires technical capacity to analyse such enormous datasets. Due to the enormous amount of image data produced by camera traps, the application of artificial intelligence (AI) for analysing images using deep learning has received much interest in recent times.

Deep learning, a branch of AI and machine learning that includes training artificial neural networks (ANNs) to learn from massive datasets [7]. Furthermore, from a technical perspective, deep learning algorithms are multi-layered neural networks. Neural networks are computer models that use highly interconnected processing units referred to as neurons, to jointly solve problems in a manner that is based on biological processes. These algorithms may classify images and identify objects after being trained on a pre-labelled dataset that specifically identifies a particular species or category of interest [1].

1.1 Statement of the problem

Nature is deteriorating at an alarming rate, and different human-caused actions have accelerated the loss of biodiversity [8]. These actions have resulted in a heightened loss of biodiversity, and threat to the existence of endangered species. Traditional wildlife monitoring uses techniques such as observation, seeking out tracks/signs, and the capture/mark/release of animals, all of which are prone to limitations [9]. These techniques require well-trained personnel to participate at acceptable levels of efficiency and are time consuming.

Gathering animal imagery data with camera traps is a non-invasive method for acquiring relative densities and determining population trends in animals over time [10, 11]. This allows the study to be remotely conducted on this subject by counting animals from the gathered images. Nonetheless, due to size and pixel variation,

images are often complex to analyse in their raw state, making it difficult to extract relevant information. As a result, the main approach for identifying and counting animals in images currently is human labour [11]. This hinders the development of ecological research that includes image processing. For instance, it took specialists and citizen scientists several years to manually classify millions of images [5, 12].

Camera traps (motion-sensor cameras) generate high volumes of image data capturing species, animal movement and management [9]. Considering that traditional wildlife monitoring techniques are limited in effectively analysing large volumes of image data, this study proposed the use of Deep Learning as a mechanism to automatically identify and count animal species. This includes animal monitoring and biodiversity examination to improve conservation management in South Africa.

1.2 Aim of the study

The aim of the study was to develop a deep learning model to identify, count and classify animal species using camera trap images.

1.3 Objectives

The objectives of this research were to:

- conduct investigation on image classification models for biodiversity monitoring using deep learning in order to identify research efforts in image classification and to identify research gaps,
- develop an automated image classification model for biodiversity monitoring using deep learning methods, and
- evaluate the performance of the deep learning models.

1.4 Research Question(s)

The study responded to the following research questions.

- What are the existing image classification models used in biodiversity monitoring through deep learning, and what are the current research efforts and gaps in this field?
- How can an automated image classification model be developed for biodiversity monitoring using deep learning methods?
- How can the performance of the deep learning models be evaluated?

1.5 Motivation

The visible impact of human activities such as overhunting, poaching, and habitat destruction cannot be neglected. This has not only threatened the future existence of animal species, but also raised questions on South Africa's conservation strategies to mitigate these issues, granted that conservation status for distinct animal species is limited. The implemented strategies for biodiversity conservation by ecologists have addressed some of the issues but there is still much to be done. This is because data retrieved from the camera traps comes in high volumes and has become an obstacle [7]. With enormous data recorded, the task of labelling the data has become tedious and time consuming, given that it is done manually [12]. This leads to difficulties in analysing the animal species and their behaviour, as well as monitoring their population, resulting in poor conservation throughput. Research action in automating the analysis procedure to label and categorise image data using deep learning is crucial and can assist in biodiversity monitoring and conservation management.

1.6 Contributions

The study aimed to contribute to the Machine/Deep Learning (a computer science division) research by developing a deep learning model for biodiversity monitoring in South Africa. The investigation on developing a deep learning model will contribute to the body of knowledge. Implementing cutting-edge methods in biodiversity monitoring as a strategy to tackle the issues that result from employing traditional ways of extracting information through human-labour, is both a creative and an imperative effort. With the disadvantages occasioned by traditional ways, the study approached the concerns of wildlife conservation (area of application) using deep learning mechanisms. Deep learning mechanisms include the extensive architectures of Convolutional Neural Networks (CNNs).

Convolutional Neural Networks (CNNs) are significant mechanisms in deep learning, especially in the domain of computer vision. CNNs are particularly engineered to analyse grid-like data, such as imagery data, by harnessing the strength of convolutional layers and providing capabilities that conventional ways cannot provide. This was done to sufficiently preprocess the image data collected as preparation for usage when conducting experiments. As a result, this constitutes the research's primary contribution of developing a deep learning model to identify, count and classify animal species in camera trap images to be employed for wildlife conservation.

1.7 Location of the Study

The present research is in the computer science domain. Computer science is a branch of research that explores how computers solve issues in real life [13]. At the core of the study is the use of deep learning (a computer science method) in the study of wildlife conservation (area of application). Deep learning is a subfield of machine learning that uses neural networks to identify patterns in data. It is a branch of AI that is inspired by the structure and function of the human brain.

Deep Learning can be used to identify, count, and describe animals from a large volume of camera trap data (images) to improve biodiversity monitoring and conservation [12]. In this study, camera trap imagery data provided by the South African Wildlife dataset is used.

1.8 Limitations

The data for use in this research was gathered with the goal of performing object detection and image classification of animal species in camera trap images. The dataset consisted of four animal classes commonly found in nature reserves around South Africa: the buffalo, elephant, rhino, and zebra. However, research can be expanded, and the principles used in this study can be used in other studies related to wildlife conservation and ecology. Furthermore, conducting research that includes other animal classes would open new ideas in conserving wildlife.

1.9 Overview of the Thesis



Figure 1: Overview of the Study

- **Chapter 1:** The chapter introduced relevant concepts of ecology, biodiversity monitoring and wildlife conservation, and demonstrated the need for deep learning methods, and for developing a model to identify, count, and classify animal species in South Africa.
- **Chapter 2:** Literature is reviewed according to five themes, namely, importance of ecological monitoring for wildlife conservation, camera trap images, utilising machine learning in the field of computer vision, deep learning, and deep learning methods for wildlife conservation.
- **Chapter 3:** This chapter discusses how the aim of the study was achieved by following a quantitative approach, firmly grounded in the positivist school of thought, where the incorporation of deductive reasoning and arguments were made with emphasis on developing and evaluating the deep learning model.
- **Chapter 4:** This chapter dwells on the experimentation methods that were described in chapter 3. The report discusses the techniques used in this study, this includes image processing for data preparation, in-dept details on the convolutional neural networks, and the hyperparameter adjustments conducted in identifying the proposed deep learning model.
- **Chapter 5:** The results generated by the deep learning models that were implemented in chapter 4 are reported in this chapter, based on the two identification tasks; image classification and object detection. The chapter provides discussions on the performance of the models.
- **Chapter 6:** This chapter provides a detailed discussion on the advantages of the results acquired in chapter 5, and sheds light on the future directions and improvements to deep learning for wildlife conservation in South Africa.

1.10 Conclusion of the Chapter

The introduction of the research demonstrated the significance of problem solving and deep learning techniques for biodiversity monitoring in managing wildlife conservation. The statement of the problem introduced root keywords such as biodiversity monitoring, camera traps, image classification, and deep learning. The subsequent chapters delve deeper into the keywords identified in the problem statement and the significance of each term to the aim of the study.

Chapter 2: Literature Review

2.1 Introduction

Biodiversity, which encompasses the variety of life forms on Earth at different levels ranging from genetics to ecosystems, entails the evolutionary, ecological, and cultural elements that support life in its diverse manifestations [14, 15]. It promotes ecological processes that lay the groundwork for essential ecosystem services, including the provision of food, water, maintenance of soil fertility, and control of pests and illnesses [16]. This makes biodiversity conservation essential for the coexistence of people and other living organisms within the ecosystem. Precise species identification serves as a foundation for systematic evaluation of extant mechanisms used in conservation and plays a decisive role in several biological research fields like agriculture, ecology, and medicine. Extensive research efforts conducted in the domain of computer vision and machine learning have generated numerous publications proposing and evaluating automated species identification techniques [11, 17].

In this chapter, a review of scholarly sources is undertaken to provide a comprehensive overview of the current state of knowledge. This review allows for the identification of relevant hypotheses, approaches, and research gaps within the existing body of literature. Additionally, it involves the application of proposed methodologies and scientific insights, presenting evidence to support theories and justifications, with the goal of establishing the validity of the study's findings. The study aims to respond to the concern of processing large amounts of imagery data to classify animals, as well as to assess biodiversity monitoring [12]. This work seeks to contribute to this expanding body of academic literature.

2.1.1 Overview of the Chapter



Figure 2: Overview of Chapter 2

According to Miao, et al. [10], the use of machine learning and deep learning for the analysis of biodiversity monitoring and animal classification offers several opportunities. This viewpoint is substantiated by a growing body of research on wildlife conservation, which employs deep learning and artificial intelligence mechanisms to propose innovative solutions for the complex challenges encountered in conservation management.

The objective of this chapter is to offer essential background information and establish the theoretical foundation for the study. In line with this objective, the chapter is organised as follows:

To begin with, section 2.2 provides an overview of ecological monitoring. Following that, section 2.3 delves into the utilisation of camera traps in wildlife conservation. Subsequently, section 2.4 conducts a literature review on the role of integrating machine learning techniques in the field of computer vision. Section 2.5 focuses on the deep learning methods, lastly section 2.6 describes the role of deep learning methods in wildlife conservation. The chapter concludes with the development of a deep learning model, indicating the contributions made by this section to the existing body of knowledge.

2.2 Importance of ecological monitoring for wildlife conservation.

Ecological monitoring is the process of gathering and analysing information on trends and conditions of ecosystems such as biotic (living) and abiotic (non-living) elements and relations between them [18]. The effects of loss of biodiversity are a concern, and it has become imperative to address the concerns with adequate conservation [4]. Wildlife community structures and the understanding of diversity within a focus area is vital for reducing current and future conservation warnings and improving management practices [9, 19]. Fundamentally, there is growing interest in monitoring wildlife biodiversity or population monitoring for maintaining ecological balance, protecting endangered species, and ensuring long term sustainability for South Africa's biodiversity and natural resources [19]. This is meant to minimise the dangers that come with ecological imbalances and avoid human-wildlife conflict and overpopulation of some species, or depletion of food resources.

In a case where animal species are not monitored, the impact can become disastrous over time. This may result in unavoidable overpopulation, causing animals to more likely trespass into human territories, resulting in conflicts between

humans and animals. As a result of the growing incidents, wildlife conservation authorities are highly motivated to enhance monitoring methods [20]. An efficient system to monitor wildlife populations can assist help in effectively managing the numbers of animal species. By implementing an effective monitoring system, conservationists and authorities can make informed decisions regarding habitat protection, population control measures, and conflict resolution. In addition, management of animal species guarantees the equilibrium of species within the ecosystem.

2.3 Camera trap Images

Over the past two decades, the deployment of motion-sensor cameras, or “camera-traps” in wild habitats, has revolutionised wildlife ecology and conservation [12]. Pardo Vargas, et al. [5] studied 31 wildlife areas in South Africa with aid from the Snapshot Safari – South Africa network, with 21 of these areas being permanent grids for continuous monitoring. A total of 1,408 cameras were set up for networks at fixed positions, while 83 were installed at roaming locations. The use of camera traps has become vital in conservation management. Camera traps are essential for wildlife conservation in South Africa, as they enable researchers and conservationists to gather important data on the presence, abundance, and behaviour of species in their natural habitats [21, 22]. Camera traps produce large amounts of images which makes it difficult to classify different animal species.

Although there are many advantages of camera traps [12], managing the amount of human-labour needed to process, analyse, and study large volumes of data is difficult. Comprehensive and current knowledge about the location and behaviour of animals in the natural environment could enhance the study and conservation of biodiversities [11]. Norouzzadeh, et al. [12] observed how to gather such data automatically, accurately, and affordably, which helps encourage the growth of “Big Data” studies in several domains, including wildlife biology, conservation biology, and zoology.

With the goal of classifying different species using camera traps, research on analysing large volumes of camera trap images to improve conservation management has emerged [22]. The majority of movement ecology research relies on a variety of sensors, primarily biologging equipment (e.g., Joo, et al. [23]). However, tracking studies are usually limited to sampling a small number of animals from the population due to related costs, and the practical difficulty of capturing and tagging animal species. In addition, populations of several species have low densities, their habitats are challenging to reach and assess, and the species are secretive [22]. Therefore, there are still important fundamental gaps in movement ecology [22, 24]. Potential knowledge gaps to improve biodiversity and monitoring include the use of AI to perform image processing to analyse population and animal movement patterns [25]. To ensure effective image processing, Pardo Vargas, et al. [22] combined citizen science initiatives with advanced machine learning for effective picture processing.

A substantial number of research projects have introduced the use of machine learning techniques, specifically deep learning for information extraction from camera trap data [26]. Although the traditional machine learning and some few deep learning techniques in wildlife and ecological monitoring research dates to early 2000s [11], literature demonstrates that machine learning methods are essential in resolving numerous intricate ecological problems, given the capabilities to analyse complex nonlinear data [9]. Some studies are involved with automatic classification, animal behaviour, and ecological monitoring [8]. The current technological advancements and integration of machine learning in the field of computer vision has led to enhanced image processing and analysis of high volumes of camera trap data in species identification studies.

2.4 Utilising Machine Learning in the field of Computer Vision

In the modern era, machine learning represents the most rapidly advancing area in computer science, expanding its significance across a wide range of diverse

disciplines like marketing, healthcare, manufacturing, information security, etc. [27]. The primary reason behind this substantial growth can be related to the union of three key factors: 1) the availability of faster and effective computer hardware resources, this includes extensive parallel processors and graphics processing units (GPUs); (2) the development of software algorithms intended to utilise the capabilities of these computational architectures; (3) the virtual endless availability of training data for specific problems, computerised images, digital documents, social platform content, or geolocation-based observation [28].

Machine learning (ML), a subset of artificial intelligence (AI), can accomplish tasks without explicit programming for the related solution [29, 30]. Alternatively, it gains knowledge from prior instances of the same task through a series of processes referred to as training. It can apply this gained knowledge to new data samples; a phase called inference [31]. Machine Learning is particularly beneficial for tasks requiring extraction of information from enormous and continually expanding datasets, making it well-equipped for scenarios where extant traditional analytical modelling faces difficulties, such as when performing analysis on visual material such as imagery and video data.

Computer Vision (CV) is a discipline within computer science, which focuses on collecting information from images and videos [32, 33, 34]. In the engineering domain, its aim is to automate operations analogous to those performed by the human visual system [35]. As a result, CV primarily involves creating artificial systems designed to address specific visual challenges, and as such, it employs techniques related to image processing and analysis [32]. Moreover, it is closely related to other fields such as machine learning and image recognition. The channel of machine learning for computer vision generally comprises two stages: feature extraction and classification. Various solutions related to species classification and detection through image processing and analysis have been widely surveyed in prior studies [12, 36, 37, 38, 39, 40]. Figure 3 illustrates a computer vision channel used

for identifying animal species, serving as example to clarify fundamental computer vision and human vision terminology.

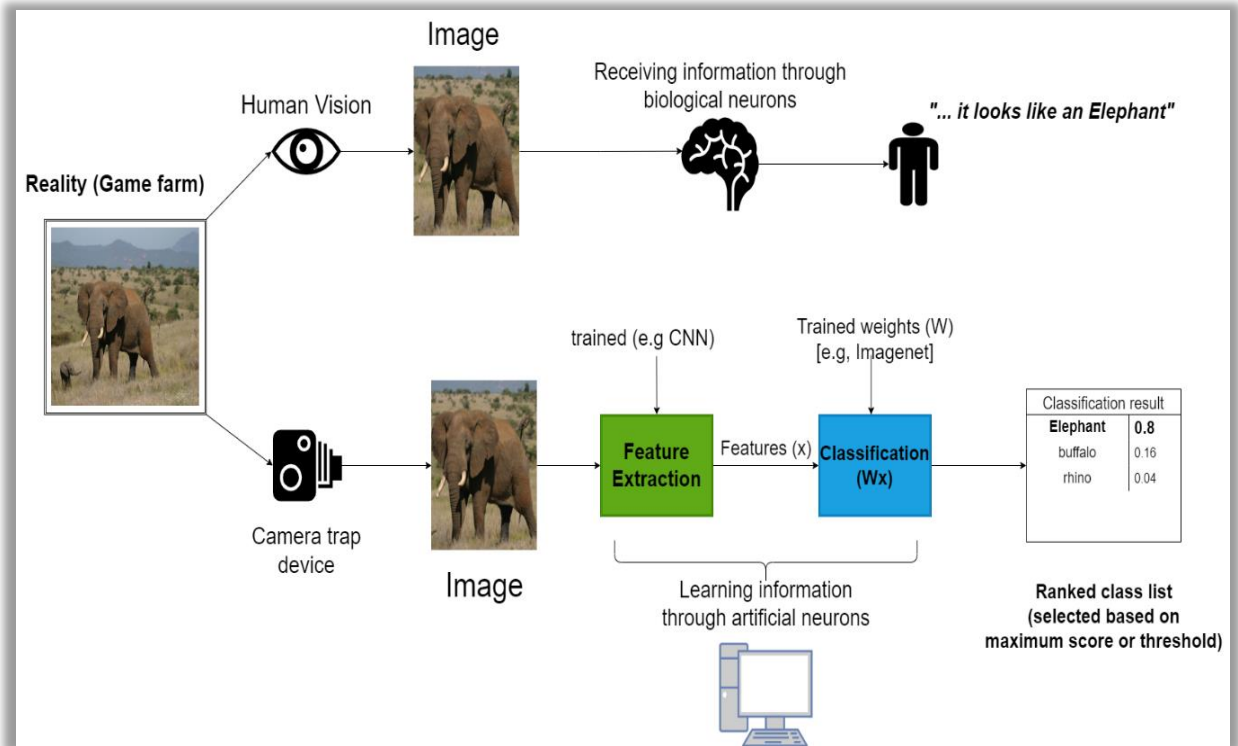


Figure 3: A computer vision channel used for identifying animal species [17].

2.4.1 Feature Extraction

Feature is a process of converting unprocessed data into insightful representations custom-made for a specific identification task. Images are usually made up of millions of pixels, each containing related colour information. To handle the processing of these high dimensional images, abstract features are computed. The generated features serve as quantified illustrations of the image, preserving important information for classification task, such as texture, colour, or shape details,

while removing insignificant data. Previously, features were typically designed by field experts through a relatively tedious, and to some extent, subjective manual procedure. For example, it was noticed that people have a sensitivity to edges when looking at an image. Several established CV algorithms comply to this paradigm and employ edge-based features, such as the well-known scale-invariant feature transform (SIFT). SIFT serves as a commonly used method for image comparison and object detection tasks. It efficiently identifies and describes unique image key points and allied characteristics for object recognition, providing significant advancements over extant techniques [41]. In the subsequent section, the classification result of feature extraction is revisited and its evolution in the rising era of deep learning is explored.

2.4.2 Classification

Classification involves associating an input image with a specific predefined class [42]. A class is represented by a set of feature values captured during the training stage. In the classification phase, models operate under the assumption that the image is composed of different features like colour, texture, shape, and facial features; with each feature collection corresponding to one of several distinct classes. The result of this stage is either the identification of the animal or finding the most similar match. It involves two critical phases: training and testing. During the training phase, unique attributes of common image features are assigned to a classification category. Subsequently, in the testing phase, the predefined classes are employed to classify images.

The result of feature extraction generally takes the structure of a vector (as illustrated by 'x', see Figure 3), which afterward, is transformed into a confidence score using a classification model. Based on the utilisation context of the process, the score may either be correlated to a threshold to find out the absolute presence or absence of an object. For instance, detecting the presence of an animal or a particular object in an image, or comparing it to other scores to make a distinct

evaluation between object classes (e.g., detecting the animal species). Well distinguished classification techniques comprise machine learning models like Decision Trees, Random Forest, Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs).

Features acquired from images display what the model perceives in an image, and selection is highly dependent on the specific problem and object being analysed. Previously, the process of determining unique features was performed manually, and it was critical for achieving adequate classification results. However, the drawback was the time consumption and the subjectively driven task performed by field experts. In addition, there were limits to accurately extract various features manually. Hence, there is long-standing research for a method that can perform automatic identification of acceptable features for a problem, without a predefined science of reasoning. ANNs address this problem by automating the process of feature extraction through learning an acceptable depiction of data from gathering samples and constructing an effective model on their own. The automated feature extraction process has proven to be highly effective in computer vision duties, and in modern architectures for image classification, image retrieval and object detection [17]. However, the current popularity and expansion of deep learning methods can be integrated into the high-tech advancements accomplished in gathering massive datasets, and in the subsequent need for correlative and potent data modelling mechanisms.

2.5 Deep Learning

Deep learning is an expansion of frequently used ANNs, which are mathematical models influenced by the learning principle observed in biological neural networks, such as the central nervous system found in animals, specifically their brain. As reference, the human brain typically consists of around 86 billion neurons [43]. In terms of biology, each neuron comprises a cell body, a set of dendrites responsible for transmitting electrochemical data into the cell, and an axon that conveys

electrochemical data out of the cell. A neuron generates an outcome through its axon, firing each time when the cumulative result of its inputs surpasses a specific threshold. The axon of one neuron can affect the dendrites of another neuron at intersections known as synapses. Several synapses have a good influence on the dendrites, prompting its neuron to fire, while other synapses generate a negative impact, hindering the neuron from firing. Figure 4 shows the different structures of biological and artificial neural networks.

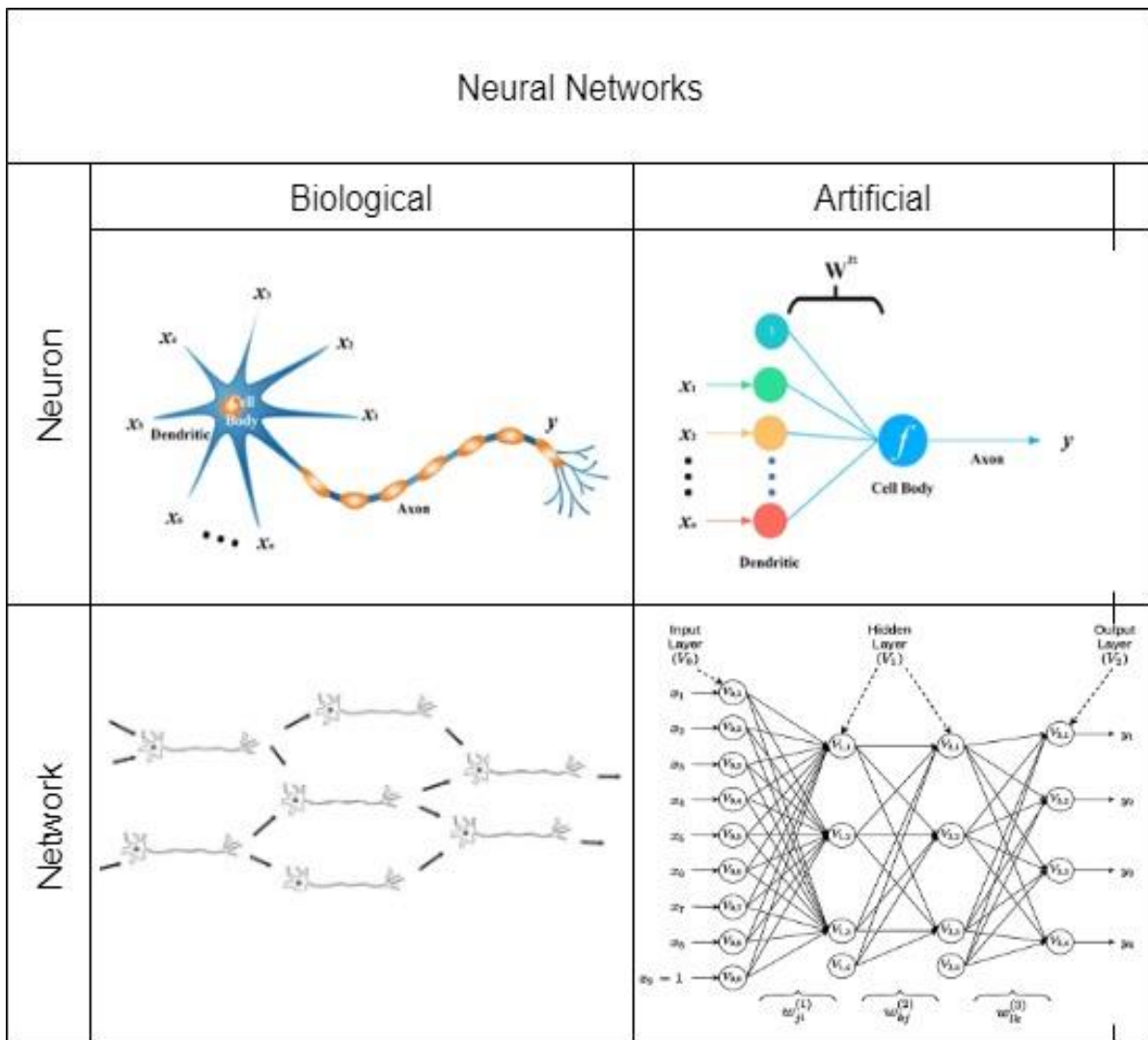


Figure 4: Different structures of biological and artificial neural networks [43, 44].

2.5.1 Neuron

The biological nervous system consists of a complex network made up of numerous neurons. In a similar manner, neurons serve as fundamental processing components within artificial neural networks. The central operating principle involves employing multiple input to mathematical transformations to generate an output value (see Figure 5). The mathematical transformation relation between the input signal and the result value is shown in equation 1.

$$f\left(b + \sum_{i=1}^n (x_i \times w_i)\right) \quad (1)$$

The activation or threshold function $f(.)$ can either be ReLU, Sigmoid, Tanh, etc. Fundamentally, different ANNs apply distinct activation functions, this means various activation functions are employed based on the specific classifier in use.

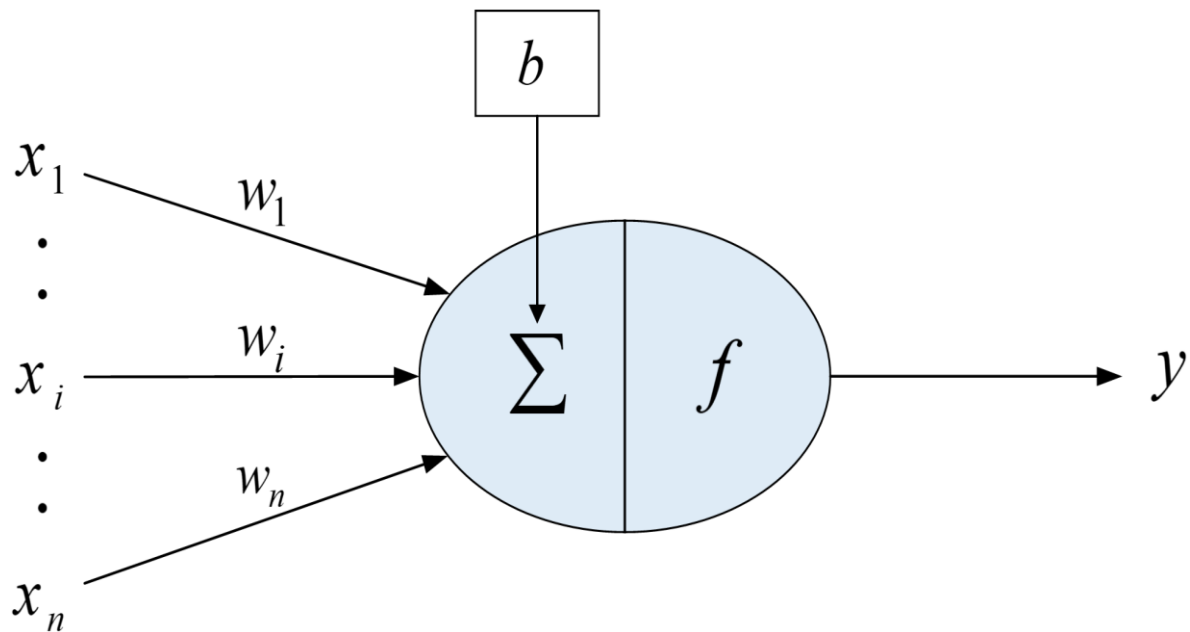


Figure 5: Basic artificial neuron [44]

2.5.2 Fundamental Neural Network

With machine learning algorithms created with the purpose of identifying patterns within data for predictive ambitions, neural networks serve a similar role but with a higher level of complexity. The term “neural network” draws its inspiration from its similarity to the human brain, given its functional resemblance to the fundamental unit of the brain, the neuron [45]. In contrast to a biological brain, where neurons form connections with any other neuron within a particular physical range, ANNs consist of a fixed and predefined number of layers and connections. Neural networks consist of numerous layers of linked neurons [44]. A layer indicates a group of neurons, with individual neurons processing information from the preceding layer before transmitting the output to the neurons in the succeeding layers, progressing from left to right. Figure 6 shows the fundamental architecture of a neural network.

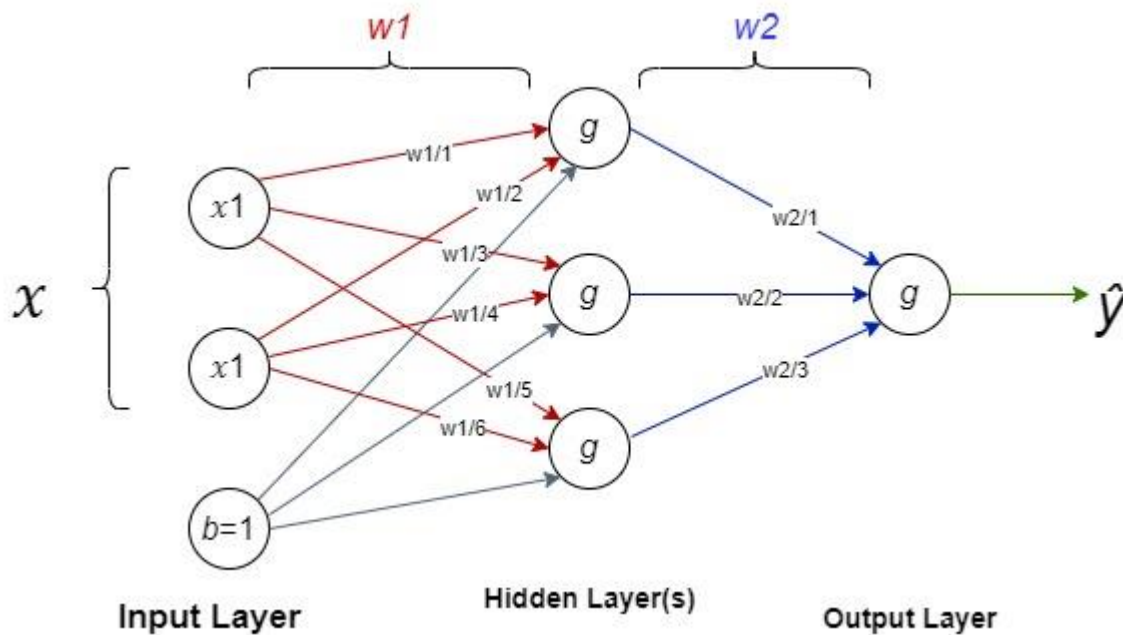


Figure 6: Fundamental structure of a Neural Network [44].

Neural networks (see Figure 6) are structured with three primary layers: an input layer, one or more hidden layers, and an output layer. The input layer initiates the network's operation by accepting and transmitting the input data to the hidden layers. The succeeding hidden layers consists of neurons, and the connections between neurons across layers are established through weights (denoted W^n). The weights, which are real-numbered parameters within the neural network, play a crucial role of altering the input data as it progresses through the hidden layers. Each time a neuron receives an input, it undergoes a process in which it is multiplied by the associated weight value, followed by the inclusion of the bias quantity.

Weights and bias quantity serve as parameters that undergo data estimation based on the available data. In the beginning stages, both weights and bias are assigned as random real-numbered values [46]. These weights play a crucial role in signifying the intensity of connections within the network, and they determine the magnitude to which modification in a neuron will influence the output. In simpler terms, a lower weight has a minimal impact on the outcome of a neuron, whereas a larger weight exerts a more substantial influence on a neuron's output, and this, in turn, has a direct significance on the final outcome.

Bias value, also known as an offset quantity, is typically set to 1. The purpose of the bias value is to fine-tune the result of the weighted sum of inputs, enhancing the model's adaptability to match the data more efficiently. Bias can be considered related to the intercept, denoted as 'b', in the linear function $y = Wx + b$. In this equation, 'b' permits the line shift of both upwards and downwards, enabling the model to better adapt to the data. In the absence of 'b,' the line would be constrained to advance through the origin (0,0), restricting its mobility and leading to a less precise fit to the data. In addition, it allows the model to accommodate instances where all input data points are zero.

Finally, the data undergoes an additional transformation using an activation function denoted as " $F(g)$ ", which is applied within the neurons.

In general, when a neural network comprises just a single hidden layer, it is categorised as a “shallow” neural network. Alternatively, when it includes multiple hidden layers, it falls into the category of “deep” neural networks, hence the term “deep learning” [47]. In such networks, as the number of neurons and layers increases, the capacity to identify and detect intricate features and patterns within the data also escalates. This occurrence is referred to as a “feature hierarchy”, where abstraction and complexity grows as neurons accumulate and reconstruct features from the preceding layers [43]. The type of neural network commonly used for deep learning in image processing is the Convolutional Neural Network (CNN).

2.5.3 Convolutional Neural Network

Convolutional Neural Networks (CNNs) presents a category of neural networks mainly recognised for their efficiency in handling image and video data [43]. CNNs consist of one or many convolutional layers, very much alike to a conventional multilayer neural network (see Figure 6). The CNN architecture is specifically designed to leverage the 2-dimensional framework of input images. It accomplishes this through local connections, tied weights, and subsequent pooling operations, which form features that are translation invariant. CNNs recognise the extant spatial design of the inputs and can capture intricate image features by using filters.

Research into CNNs dates to the early 1980s, as evidenced by the work of Kohonen [48]. The first notable practical implementation of CNNs appeared with the development of LeNet [43] for recognising handwritten digits. However, regardless of these early achievements, the extensive adoption of CNNs did not gain strength until significant advancements in parallel computing systems converged with different new methods for effective training. After a period, a pivotal moment occurred in the year 2012 when Krizhevsky, et al. [49] made a significant contribution to the well-known ImageNet Challenge (ILSVRC). Their CNN, referred to as Alexnet, secured victory in the contest by lowering the classification error rate from 26% to 15%. In the subsequent years, various networks continued to advance, including

renowned ones such as Visual Geometry Group network (VGGnet) [50] and Residual Network (ResNet) [51]. Table 1 shows the CNN performance on the ImageNet challenge.

Table 1: Performance of CNN architectures on Imagenet.

Name	Layers	Top-5 Error (%)	References
AlexNet	8	15.3	Krizhevsky, et al. [49]
VGG Net	19	7.3	Simonyan and Zisserman [52]
ResNet	152	3.6	He, et al. [51]

While CNNs have clearly emerged as the leading networks in many image-based species recognition projects, the specific model is not the significant element that determines a successful result. When examining projects that use the same network architecture for a related challenge, the results of the model performance usually differ. A frequently overlooked aspect is the significant contributions of field expertise in tackling the task, which expands further outside than adding more layers to a CNN architecture. Scholars accomplishing exceptional performance with deep learning algorithms distinguish themselves by factors beyond network architecture. These include specialised design of image preprocessing and augmentation of training data. Chen, et al. [53] highlighted that these methods require a deep grasp and comprehension of the study and data used.

Data preprocessing and augmentation are significant factors, but not determinants of overall model accuracy and robustness. For instance, constructing architectures that integrate specific properties custom-made to the task can yield exceptional results compared to traditional CNNs. A distinct example is the use of multi-scale network [54]. Through the process of image cropping of several parts to acquire different pieces of insightful information, in-depth data about a particular animal species (i.e., zebra) can be gathered. Smaller-scale patches can capture animal-

specific information such as teeth, whereas larger-scale patches offer background surrounding the animal. Furthermore, general underestimated aspects of a network framework include the receptive field and input capacity. Receptive field is the region in the input scope that commit to a distinct result block [17]. The choice regarding the input size should be made after careful consideration of the appropriate resolution and context required to address the subject matter. Other further considerations that one should keep in mind are the challenges that arise when undertaking a study in the field of deep learning.

2.5.4 Challenges in Deep Learning

Selecting a learning rate.

The learning rate (LR) refers to the frequency at which a model's parameters are updated during the training phase. LR plays a pivotal role in controlling the speed of the model, and is a hyperparameter that requires tuning, particularly with certain optimisation algorithms like ADAM [55]. If the LR is set too low, it can lead to slow convergence, or get stuck at a sub-optimal solution. However, if it is set too high, it can cause the objective function to diverge.

Selecting a Network Size

Preventing model overfitting can be achieved by ensuring that the model is not overly intricate. This involves lowering the quantity of layers and blocks in the network, and effectively lowering the number of trainable parameters as well. However, it is important for the model to have an adequate number of parameters to grasp the complexity of the dataset. Thus, determining the desirable equilibrium between model simplicity and complexity is a matter of empirical experimentation in ML. There is no specific blueprint to accurately establish the optimal model intricacy particularly on the data alone. Hence, it is necessary to explore different model architectures and assess their performance on the validation set [56].

2.6 Deep Learning methods for wildlife conservation

Deep learning possesses the mechanisms to analyse big data systematically and develop comparative models in large datasets. This can be a vital method for ecologists to effectively manage, analyse, and evaluate biological related information [7, 8]. Scholars attempted to employ deep learning to automatically identify species and discard images that do not display animal species, which significantly reduces the amount of human effort and time [57].

Studies on harnessing deep learning to automatically gather important attributes to identify, count, and label wildlife species on large wildlife datasets (i.e., the SS dataset) have proven to be vital for ecological research and wildlife conservation. Gathering attributes from raw data frequently improves performance [58]. This is because these attributes can be conveyed to other fields with small-scale datasets [59]. It tends to be time consuming to practice standard methods to design attributes compared to a generic algorithm that learns attributes that consequently improves performance on various types of data (e.g., text and sound [60]).

Yu, et al. [61] performed feature or attribute extraction with sparse programming spatial pyramid matching, and utilised linear support vector machine to classify images. To fulfil an 82% accuracy, their approach required manual imagery cropping, which consequently needed significant human intervention. Chen, et al. [53] employed CNNs to entirely automate the animal identification task, although they primarily performed the technique on a small image dataset consisting of 20 000 images and 20 classes. Moreover, they managed to achieve an accuracy of 38% for their technique, which indicates significant potential for improvement. However, Chen, et al. [53] discovered that, given substantial volume of training data, DNNs surpass the conventional Bag of Words method.

Norouzzadeh, et al. [12] conducted a study on automatic image classification using multitask models and were able to accurately classify images of animals with a

classification accuracy of 93.8%. Similar to their research was Gomez, et al.'s [36], which analysed DNNs on the SS dataset. They primarily conducted the task of identifying the species, while Norouzzadeh, et al. [12] additionally made an effort to count animals, assess their behaviour, and determine whether there was presence of the offspring. With the focus on the species-identification tasks conducted by the two studies ([12] and [36]), the models' performance of their networks was 92.0% and 57% respectively.

Schneider, et al. [62] successfully resolved the issue of achieving single label output for multi-species images through training object detection models such as Faster R-CNN. Object detection and identification can locate and classify target objects in an image and present the findings. This will further enhance the capacity of camera data processing [62]. Further to that, Tan, et al. [8] discusses several studies that argue that in complex natural habitats, locating animals would initially serve as the foundation for enhancing classification accuracy.

Matin, et al. [26] performed transfer learning using VGG16 and ResNet50 on ImageNet dataset to classify animal species. The reason behind employing these model architectures was the extensive image processing capabilities each of them possessed in classifying animals on a larger dataset. Vecvanags, et al. [63] assessed the effectiveness of RetinaNet and Faster R-CNN, which offer technological assistance for efficient monitoring and animal control. In the discipline of deep learning, various object identification models have recently been put forth, and an increasing number of studies have centred on these ecological applications.

2.7 Research Gap

While object detection offers potential advancements, an automated model is required to recognise an individual it has already identified to accurately estimate population parameters. Human analysis of camera trap re-identification procedures is prone to inevitable bias and has led to questioning the accuracy of human efforts on re-identification of animal species using camera trap data. This gap leads to

minimal efforts made for biodiversity monitoring, and subsequently, slow improvement of wildlife conservation in South Africa. This study aimed to address the obstacle by conducting animal re-identification specifically on single and multiple-species recognition using deep learning methods in South Africa. To my knowledge, no previous study has shown the use of deep learning on wildlife conservation (area of application) in South Africa. [36].

2.8 Conclusion of the Chapter

Establishing precise information about the species identity and geographical range holds a critical role for the continuous progress between humanity and the conservation of biodiversity. Research on species identification relied heavily on structural assessments provided by systematic studies. In the present day, accurate species identification is mainly within the field of systematic experts and well-trained technicians, as it requires specialised skills honed through comprehensive experience. However, the number of experts is dwindling significantly, which has led to an expanding demand for alternate and precise identification techniques that can be applied by individuals with no special expertise.

The existing studies on developing a deep learning model for animal classification reveal that the adoption of deep learning techniques is still in its early stages, particularly in South Africa, as indicated by recent literature. Furthermore, there is a notable scarcity of research in the field of deep learning concerning the processing of animal imagery data, especially for biodiversity monitoring purposes. By integrating deep learning methods into the development of such a model, more transparent and comprehensive mechanisms can be established. Additionally, these methods offer fast processing capabilities and play a pivotal role in creating an effective deep learning model that can drive conservation action.

The subsequent chapter of the study elucidates how the concepts discussed in the literature review were methodologically incorporated to achieve the study's

objectives. Furthermore, it provides a clear explanation of how deep learning approaches were employed to construct the animal classification model, providing insights into the research process and its potential contributions to wildlife conservation.

Chapter 3: Methodology

3.1 Introduction

The primary objective of this study is the development and assessment of a proposed animal classification model applying extensive deep learning techniques, specifically a convolutional neural network. This novel model is designed to process a dataset of camera trap images featuring four distinct animal classes: buffalo, elephant, rhino, and zebra. To construct the animal classification model, Tensorflow, Pytorch, Keras, and other python libraries are used in this study. The fundamental purpose of this model is to employ convolutional neural networks to accurately identify, count, and classify animals based on their respective classes.

The study adopts a quantitative approach, firmly grounded in the positivist school of thought, where the use of deductive reasoning and arguments is used with emphasis on developing and evaluating the proposed deep learning model. The research outcomes presented in this study were obtained through experimental data collected for classification and object detection tasks.

3.1.1 Overview of the chapter

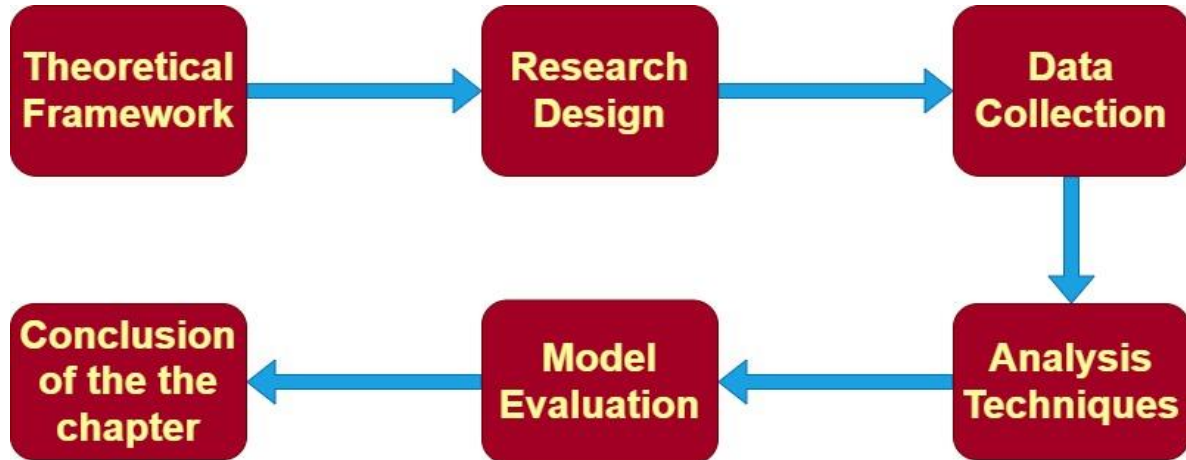


Figure 7: Overview of Chapter 3

3.2 Theoretical framework

Design Science is a systematic approach for creating Supervised Machine Learning (SML) models aimed at addressing identified issues, advancing research, evaluating designs, and disseminating findings to the appropriate audience [64]. These artifacts encompass elements like constructs, models, procedures, and instances. They may also include social innovations or novel aspects of technological, social, and information services. In essence, this definition encompasses any constructed entity that presents a solution to a recognised research problem. The design science adopted by Peffers, et al [64] is illustrated in Figure 8.

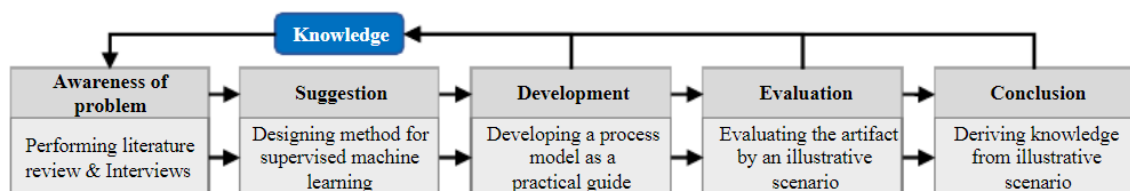


Figure 8: Design Science Research Activities of the First Cycle: [64].

The five phases of the design science and their application to this work are discussed below.

3.2.1 Awareness of problem

The first phase is the problem awareness phase. In this phase, the problem that required to be addressed was identified (see section 1.1). Then it was followed by an in-depth literature review, whereby the current body of literature revealed pertinent hypotheses, approaches, and research gaps. Furthermore, it entails implementing suggested methodologies and scientific perspectives, offering substantial proof for theories and justifications, all with the aim of validating the findings of the study.

3.2.2 Suggestion and Development

In these phases, data and algorithm techniques are presented to accomplish the aim of developing a deep learning model (see section 3.3). The goal is to create a deep learning model capable of automatically detecting and counting animals in image data. The image data goes through various image processing techniques to ready it for experimentation and prediction using the proposed Convolutional Neural Network (CNN) architectures. These CNN architectures are tailored to execute image identification tasks through the fine-tuning of parameters.

3.2.3 Evaluation

The fourth phase of the design science involves a discussion on the evaluation methods used in this study. Evaluation methods are discussed comprehensively in section 3.6. These methods accommodate both classification and object detection tasks, by providing extensive deep learning to assess the performance of both classifiers and object detector models respectively.

3.2.4 Conclusion

The fifth phase is the conclusion. In this phase, the outcome of the study is presented through a written research report.

In the beginning stages of the study, core keywords were introduced, then a survey was conducted on species identification methods to develop and evaluate the proposed deep learning model using CNN architectures and mechanisms. The aim is to investigate the requirements needed to build a deep learning model that addressed wildlife conservation challenges related to animal species classification. The intention behind leveraging deep learning techniques is to effectively tackle the specified problem and enhance biodiversity management. The South African Wildlife dataset will be employed in the development of the proposed deep learning classification model. Throughout the investigation, data analyses will be carried out, with a particular focus on testing the hypothesis and drawing conclusions from the results obtained.

3.3 Research Design

This study will be conducted under a class of machine learning referred to as supervised learning. Supervised learning is a class of ML that entails the process of learning from pre-labelled inputs (x), which serve as the desired targets. Each training sample consists of input values (vectors) and one or many matching specified output values (y). The aim of this learning approach is to achieve precise animal classification and detection, given that it holds a significant position in the field of machine learning and plays a crucial role in image processing and analysis. In accomplishing the research objectives, a series of steps will be undertaken. Among these steps is the data preparation phase, which is important for training and testing purposes of the proposed classification and object detection models. This study will use the South African Wildlife Dataset to train the proposed models, with the dataset consisting four distinct animal classes. The data will be partitioned into

three segments: (1) one for training data, (2) the second for validation, and (3) the last one for testing image data.

3.3.1 Data partitioning

Convolutional Neural Network models are designed to analyse statistical patterns within data, particularly image features. Furthermore, it is critical to maintain a consistent statistical allocation when partitioning data for both training and evaluation objectives. During the training phase, the network acquires relevant features and passes through continuous monitoring on the validation subset with a goal to mitigate overfitting.

Afterwards, the model goes through testing using an entirely different set of samples from the overall dataset, that were not employed during the previous stages of training and validation. Careful measures will be taken to ensure that the data split process is generated randomly and without any overlaps. This precautionary measure prevents any potential bias or any improper advantages that could appear if testing is carelessly included in the training sample. Furthermore, a dedicated set of test data will be reserved for each class of animal species used in the camera trap data, catering for both classification and object detection tasks. The remaining samples will be split into training and validation sets.

3.3.2 Data Augmentation

To facilitate effective model training, it is imperative to ensure uniform image size for the images in the dataset. Therefore, all images are required to be resized to a standardised dimension (i.e., identical heights and widths) convenient for model training. The quality of the model depends on two aspects: the image dataset size and image dimensions [17]. CNNs generally operate optimally with pixel numbers varying from 0 to 1. Using data augmentation for deep learning models proved to be significant, particularly when integrated with CV tasks such as image classification and object detection.

Image augmentation is a method that artificially creates diverse variation of input images, effectively encompassing a wide spectrum of potential alterations. By incorporating augmentation, the model will be exposed to a broad array of transformations during the training stage, allowing it to learn and recognise objects in different forms and shapes [26]. Furthermore, this method enhances the model's capabilities for generalisation, allowing it to make accurate predictions even on unobserved images belonging to the target classes.

In the case of the image data, which is generally made up of a variety of sizes; data augmentation will be vital to yield optimal results and reduce the risk of overfitting. Therefore, the normalisation process will be used to rescale pixel values from a range of 0 to 255. This is done using the Transforms package to pass input images through a series of transforms, to better augment the data, increase the training data size, and reveal data variations to the models.

3.3.3 Defining model architecture.

A critical step when conducting a study that uses CNN architecture is to define the proposed models' architecture. In the first step of this phase, the number of convolution layers, fully connected layers, pooling layers, activation function for each layer, etc will be outlined, as indicated by Norouzzadeh, et al [12]. These parameters play a pivotal role in determining the effectiveness required for both the classification and object detection tasks.

3.3.4 Deep learning toolkit

During the model training phase, the training set will be employed. To achieve that, this study employed extensive tools such as Tensorflow, Pytorch and Keras. Tensorflow is an open-source software created by Google focused on machine learning and deep neural networks using data flow graphs. Its versatile capabilities in automatic differentiation and parameter sharing allows the support of different types of CNN architectures [64].

Pytorch is a python-based deep learning platform created by Facebook's AI research team. It simplifies the development of deep learning models by utilising an API. Opposed to the approach of many other well-known deep learning frameworks that rely on static computation graphs, Pytorch uses dynamic computation, providing enhanced capabilities and agility for architecture construction [65]. Keras, a python-based high-standard API structured for neural networks, is compatible with both CPU and GPU processing [66]. Its main function is on enabling rapid experimentation. In this study, Keras, in conjunction with Tensorflow, will be used to construct deep learning models.

3.4 Data collection

3.4.1 South African Wildlife Dataset

The South African Wildlife dataset comprises four animal species classes: Buffalo, Elephant, Rhino, and Zebra, with each class consisting of 376 images and 376 text files, totalling 1,504 images and 1,506 text files. In the dataset, the text files use yolo annotation format, necessitating the construction of an object detection model. Figure 9 shows the overview of the image dataset employed in this study.



Figure 9: Overview of the South African Wildlife Dataset

3.4.2 Transfer Learning

In this study, a transfer learning technique will be used to retrain deep learning models with the intention of evaluating their performance in species identification in terms of both accuracy and capability [67]. Deep learning models comprise a layered architecture with distinct layers that learn complex features from animal features [12]. These layers are ultimately linked to a fully connected layer to generate the outcome. In transfer learning, the layered architecture is leveraged to employ pre-trained models like Alexnet, ResNets, and VGG. With the final classification layers of these pre-trained removed, it allows models to serve as established feature extractors. This approach seeks to enhance animal classification performance while reducing training time.

3.5 Analysis techniques

Convolutional Neural Network

A convolutional neural network (CNN) functions as a feedforward neural network [68]. It incorporates a feature extractor comprising a convolutional layer and a pooling layer. This combination effectively merges lower-level features with more advanced ones. This automatic extraction of features minimises the urgency for manual intervention, resulting in enhanced efficiency and speed [69]. Commonly, the architecture of the CNN is composed of a convolutional layer, ReLU layer (a pooling layer), a fully connected layer, and an output layer. The central aspect of CNN that sets it apart from a regular neural network is the handling of image structure during processing. The fundamental layers of CNN are outlined below.

3.5.1 Convolutional Layer

Serving as the initial stage, this layer accepts input images and captures diverse features. This layer executes the convolution mathematical operation between the input image and a filter with specific dimensions ($N \times N$). Through sliding the filter

across the input image ($N \times N$), it calculates the dot product between the filter and sections of the input image corresponding to the filter's size. The result is a Feature Map, containing details about the image's edges and corners. This resultant feature map is then transmitted to a subsequent layer, which uses it to learn different features from the input image.

3.5.2 Pooling Layer

The pooling layer serves to reduce the dimensions of the feature map. Two extensive pooling methods are employed for dimensionality reduction: max pooling and average pooling. Max pooling involves calculating the maximum value within each patch of the feature map, while average pooling calculates the average value for each patch within the feature map. The pooling layer acts as a bridge connecting the Convolutional Layer and the Fully Connected (FC) layer. As shown in Figure 10, there are two methods of pooling. Moreover, the extent of reduction in input size relies on the dimensions and stride of the pooling layer.

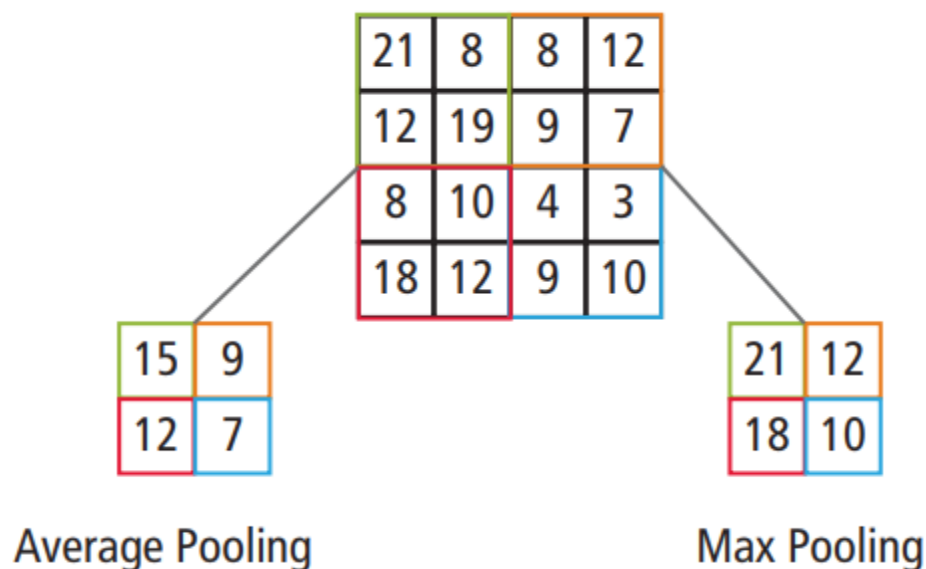


Figure 10: Representation of Average and Max pooling methods [70]

3.5.3 Fully Connected Layer

The Fully Connected (FC) layer establishes connections among neurons across adjacent layers, comprising neurons, weights, and biases. Often positioned near the previous layers of a CNN architecture, the output layer takes precedence. In this phase, input images from prior layers are flattened and conveyed to the FC layer. The flattened vector subsequently proceeds through additional FC layers, where mathematical operations are generally carried out. It is at this juncture that the classification procedure initiates.

3.5.4 Activation functions

An important element within the CNN architecture is the activation function. This study employed this function to impart dynamism to the network, enabling it to extract complex information from the data and depict non-linear and convoluted mappings between input and output [71]. A key aspect of an activation function is its differentiability, which enables the application of backpropagation for error computation. This supports optimising weights by way of techniques like Gradient Descent or other optimisation methods to minimise errors. Several widely used activation functions comprise ReLU, Softmax, tanH, and Sigmoid. Figure 11 illustrates various activation functions along with their respective mathematical representations.

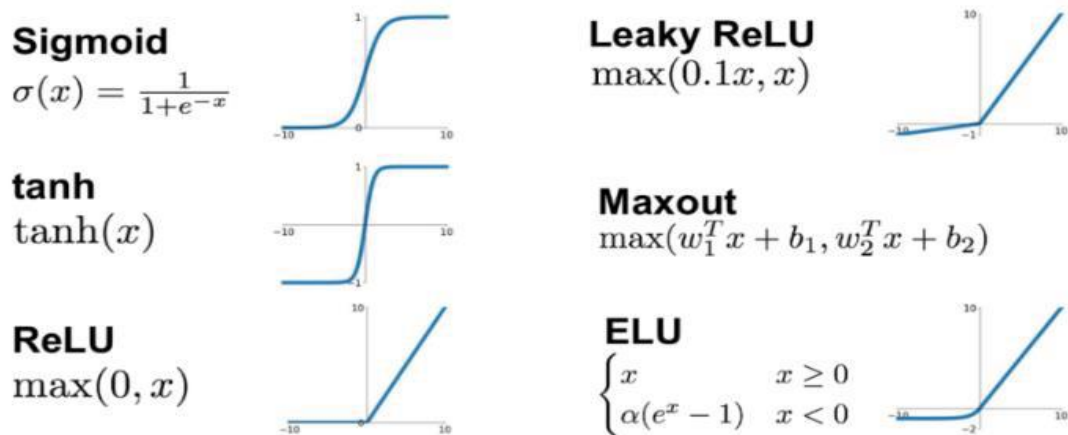


Figure 11: Designs of different activation functions [72]

Each of these functions serves a specific role. The sigmoid and softmax functions are preferred for binary classification, whereas softmax is often used for multi-class classification.

3.5.5 Regularisation

Regularisation encompasses a set of techniques targeted at decreasing generalisation error. Techniques like data augmentation (see section 3.3.2), dropout, early stopping, and batch normalisation are significant in improving generalisation.

Dropout

To mitigate the potential of overfitting, a handful of strategies are employed. Among these approaches, is the use of dropout [73]. In this layer, units are randomly selected, and their weights are set to zero; effectively excluding them from both backpropagation and the forward pass. Additional methods encompass regularisation and the expansion of the training dataset using techniques such as label preservation. In comparison to regularisation, dropout excels in accelerating

the training process and minimising the susceptibility to overfitting. Figure 12 shows the dropout approach.

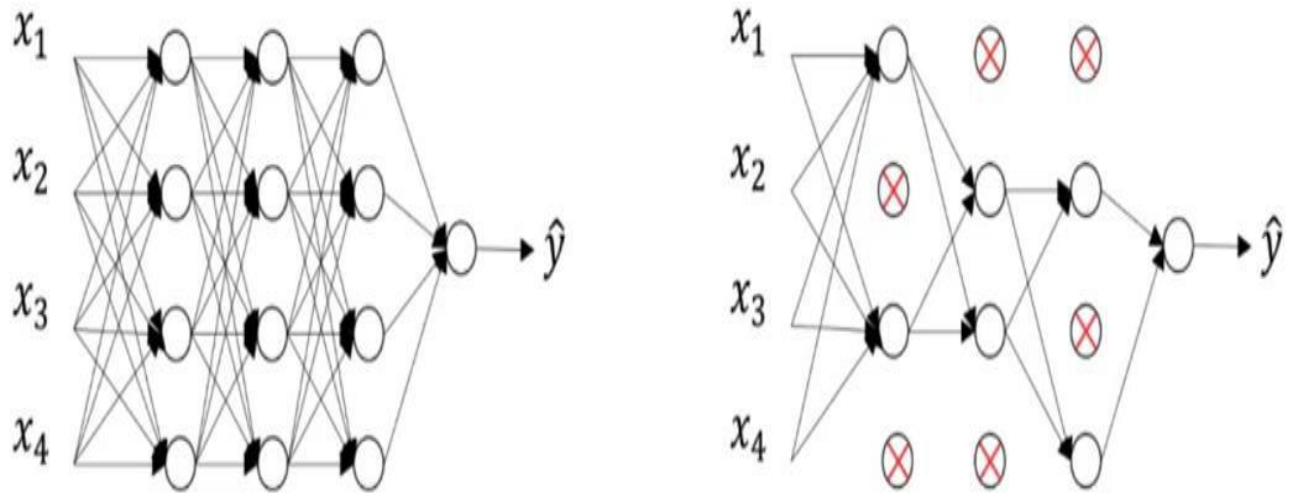


Figure 12: The Dropout approach [74]

Early stopping

During training a CNN architecture, there is a possibility for the training process to continue for a lengthy period, potentially resulting in overfitting. During the training phase, the training error tends to depreciate gradually until a given point is fulfilled. At that turning point, the validation error starts to rise, indicating the onset of overfitting. The concept of early stopping aims to mitigate this by either freezing model parameters and pausing the training the moment overfitting is detected, or as an alternative, by saving the parameters at a certain point and resuming training later [75]. The model is paused by observing its performance metrics, such as accuracy and loss on the validation data [50]. A callback function is responsible for observing the performance of the model during the specified set of procedures [76]. Over time, this function will intervene by stopping the training process when it detects that the model is no longer bettering its accuracy.

Batch normalisation

Before commencing model training, it is customary to normalise the input data between hidden layers. This is done by initialising the input data with zero mean and variance. As the training progresses, the parameters of every layer are influenced by the previous layer, given that the outcome of the previous layer becomes the input for the current layer. Consequently, the parameters of each layer constantly adapt to new value distributions. This phenomenon is referred to as covariate shift. Therefore, the first normalisation is lost, which in turn slows down the training process. To address this, batch normalisation was introduced [77].

Batch normalisation reestablishes parameter normalisation for individual small-scale batch, thereby normalising every layer. Therefore, parameter numbers become more durable, allowing for the use of higher learning rates and reduced sensitivity to initialisation parameter. In return, this promotes faster training and acts as a regularisation technique [77].

3.5.6 Optimisation Algorithm

The optimisation algorithms employed in machine learning adhere to an iterative approach, where enhancements are accomplished through an iterative process of computing the same actions multiple times. This means the outcome of each iteration becomes the input for the following iteration and increase in accuracy generally occur continuously as the algorithm advances. In this context, the number of epochs is defined. An epoch is the sum of times the model is revealed to the training set. Each epoch encompasses both a forward and backward propagation involving all the training samples. Since the model learns by adjusting its weights, training takes place over multiple epochs. Defining the number of epochs is a hyperparameter that requires tuning, and it plays a crucial role in determining whether the model might underfit, overfit, or achieve optimal performance [78].

Root Mean Squared Probability (RMSprop)

The RMSprop is one of the frequently used adaptive stochastic algorithms for training DNNs [79]. RMSprop represents a modification of Adaptive Gradient Descent (Adagrad) [80], where it maintains a rolling average of gradients. It accumulates gradients into exponentially weighted average, effectively removing previous gradient information and primarily preserves current knowledge of the gradient.

Adaptive Moment Estimation (Adam)

Adam is one of the most prominent used optimisation approaches or training algorithm [55]. This technique is widely recognised in the latest optimisation enhancements in deep learning. Adam possesses two primary advantages of both momentum and RMSprop [79]. It employs squared gradients to adjust the LR like the RMSprop and integrates a moving average of the gradient, like momentum. The mathematical representation of the ADAM optimiser for image classification tasks is shown in equation 2.

$$W_{ij}^t = W_{ij}^{t-1} - \frac{\eta}{\sqrt{E[\delta^2]^t + \epsilon}} * E[\delta^2]^t \quad (2)$$

Figure 13 shows a CNN architecture example designed for image classification.

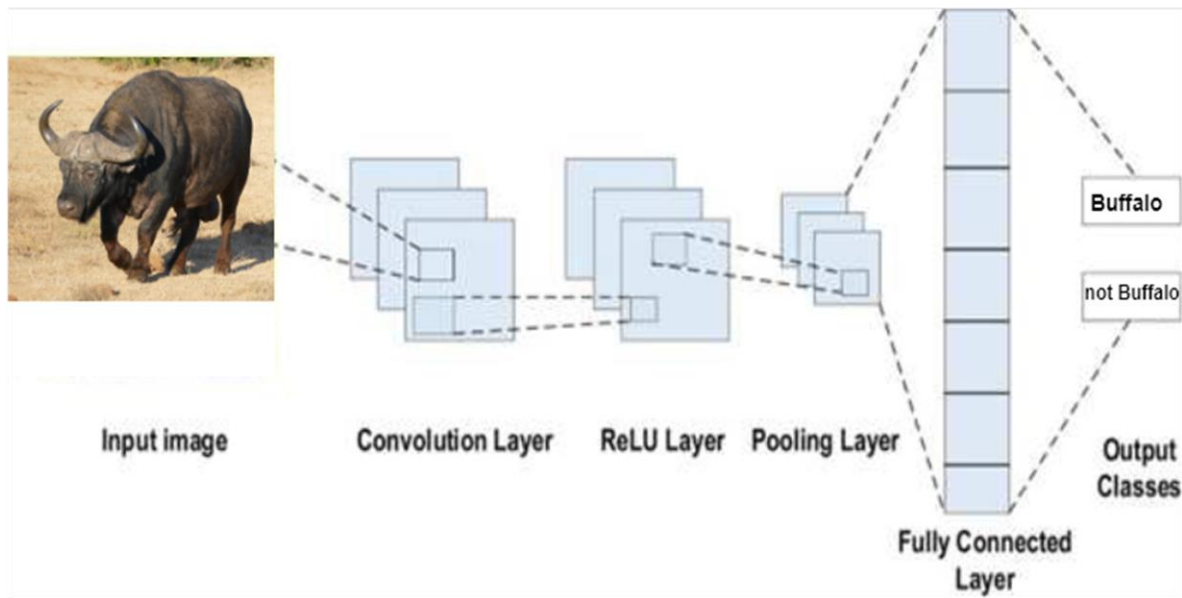


Figure 13: CNN architecture for image classification

3.6 Model Evaluation

Deep learning model undergoes an evaluation process to assess its algorithmic performance. The selection of performance metrics varies depending on the task performed and dataset used. In the context of classification tasks, metrics such as accuracy, precision, recall (sensitivity), and F1-score will be employed. Meanwhile, for the object detection task, evaluation metrics such as precision, recall, intersection over union (IoU), and mean average precision (mAP) are used.

3.6.1 Classification

Accuracy is defined as the proportion of accurate predictions out of the sum of predictions produced [81]. Nonetheless, significant performance results do not guarantee that the model is optimal, particularly when conducting experiments on imbalanced datasets. To guarantee model performance, various other metrics like precision, recall, and F1-score are employed as well.

The use of a confusion matrix is a well-known approach to assess the performance of individual classes within classification models. It allows for a detailed comparison of the quantity of accurately and inaccurately predicted occurrences for every category within the model. Furthermore, in the confusion matrix, the rows symbolise the predicted class occurrences, whereas the columns symbolise the actual class for those occurrences. Figure 14 shows the confusion matrix structure.

		Prediction Results	
		Positive (PP)	Negative (PN)
Actual Observations	Positive (P)	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

Figure 14: Structure of Confusion matrix [82]

True positives (TP) describe accurately predicted instances of the positive class, whereby the predicted result aligns with the actual positive class. In a similar manner, true negatives (TN) denote instances that were correctly predictive of the negative class. False positives (FP) correspond to instances that were incorrectly predicted as belonging to the positive class, meaning the predicted result was different from the expected outcome. Like the FPs, false negatives (FN) were instances that were inaccurately predicted as part of the negative class.

Accuracy can be computed using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

Precision is defined as the proportion of accurately predicted positive observations out of the total predicted positive observations [83]. It can be computed using the following formula:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

Recall, also referred to as sensitivity, represents the proportion of correctly predicted positive observations relative to the total number of observations within the actual class [83]. It can be calculated by using the subsequent formula:

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

F1-score is a metric that signifies the weighted average of precision and recall. Hence, it considers both FP and FN in its calculation.

$$F1 - score = 2 \left(\frac{Recall \times Precision}{Recall + Precision} \right) \quad (6)$$

In computer vision classification models, such as those used for object detection, the goal is to categorise (recognise the object within the image and determine its class) and pinpoint (forecast the coordinates of the bounding box including the object if it appears in the image). That way, it becomes necessary to use a metric that can assess the performance of both classification and localisation, especially concerning the use bounding boxes within image for object detection.

3.6.2 Object detection

Evaluation metrics for object detection are used to assess the effectiveness of detection models across different fields and applications [84]. In the context of object detection, these evaluation metrics assessed the proximity of the detected bounding boxes to the actual ground-truth bounding boxes. This assessment is conducted individually for each object class through quantifying the degree of overlap between the predicted and actual ground-truth regions.

Confidence threshold

Confidence score, associated with each detected object by the model signifies the odds that a bounding box encompasses an object of the specific class, as predicted by the classifier [85]. Anchor boxes are predefined rectangular boxes represented by specific dimensions in terms of height and width.

The confidence threshold signifies the minimum level of confidence required by the model to consider an object detected [86]. All predicted bounding boxes

exceeding this confidence threshold are categorised as positive boxes, whereas those falling below it are classified as negative.

Intersection over Union

Intersection over Union (IoU) is a vital metric for object detection models [87]. IoU calculates the ratio between the intersection and union of two bounding boxes. One of these bounding boxes represents the ground-truth, and the other one illustrates the predicted bounding box. An IoU value of one signifies a perfect overlap between the predicted and the ground-truth bounding boxes. IoU is calculated using the following formula:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (7)$$

For example, when IoU is greater than 0.5, object detection is categorised as a true positive. However, when IoU is less than 0.5, then the detection is considered inaccurate and categorised as a false positive. If there exists a ground-truth object but was undetected, it is labelled a false negative. True negative is where there is no predicted object. Nonetheless, within object detection it is not used. Illustration of IoU is shown in Figure 15.

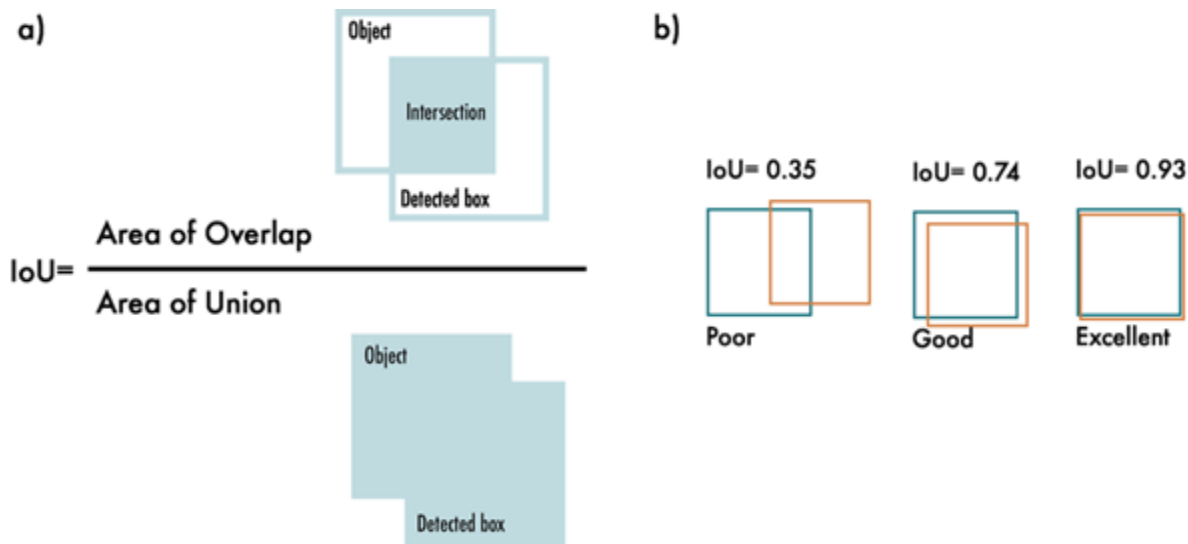


Figure 15: Depiction of IoU across different thresholds [87]

Mean Average Precision

Mean average precision (mAP) relies on precision and recall metrics as a foundation for its computations. Class predictions are linked with a confidence threshold, and the adjustment of the threshold includes reciprocity between precision and recall metrics. For instance, when a model requires to prioritise FPs over FNs, the confidence will be set higher to accomplish greater at the cost of reducing recall.

Plotting precision and recall values against the models' confidence thresholds generates the precision-recall curve. This curve takes account of the variety in confidence thresholds.

Average precision (AP) is designed to assess the precision of the detector across the entire range of recall [88]. As a result, it gives preference to techniques that have precision over unified recall, as opposed to detectors with precision curves that approach the top-right corner. In simpler terms, AP does not aim at maximum performance, but rather assesses the overall performance of the detectors. The AP employs the IoU measure to evaluate the quality of predicted bounding boxes. Below is the formula used to compute AP.

$$AP = \sum_n (R_n - R_{n-1})P_n \quad (8)$$

where R_n and P_n are the precision and recall at n th threshold.

The mAP represents the mean of the average precision scores calculated for all classes [89]. Mathematically, it can be expressed as follows:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (9)$$

where AP_i denotes the average precision for a specific class, while N represents the total amount of classes.

3.7 Conclusion of the chapter

This chapter outlined the methodology of the study, which is firmly rooted in a positivist school of thought, emphasising the use of deductive reasoning for a construction and assessment of the proposed deep learning model. The model is based on a convolutional neural network. The chapter discussed the research design, outlining the steps necessary to accomplish the research objectives. For this research, the South African Wildlife image dataset was acquired from the Kaggle repository, which consists of four distinct animal species (mammals). The architecture of the CNN model architecture included key components such as convolutional layers, pooling layers, dropout, and activation functions. The performance results derived from the experiments evaluated using a confusion matrix for classification models and assessment metrics such as precision, recall, IoU, and mAP for object detection tasks.

Chapter 4: Data and Algorithm identification

4.1 Introduction

The capability of deep learning architectures is primarily reliant on the data and the robustness of the algorithms used. In this chapter, the pivotal components of this study are examined, concentrating on the data source and algorithmic design that were the basis of the experiments. This was done through the introduction of data preprocessing methods and algorithm used for analysis to perform species identification tasks.

4.1.2 Overview of the chapter

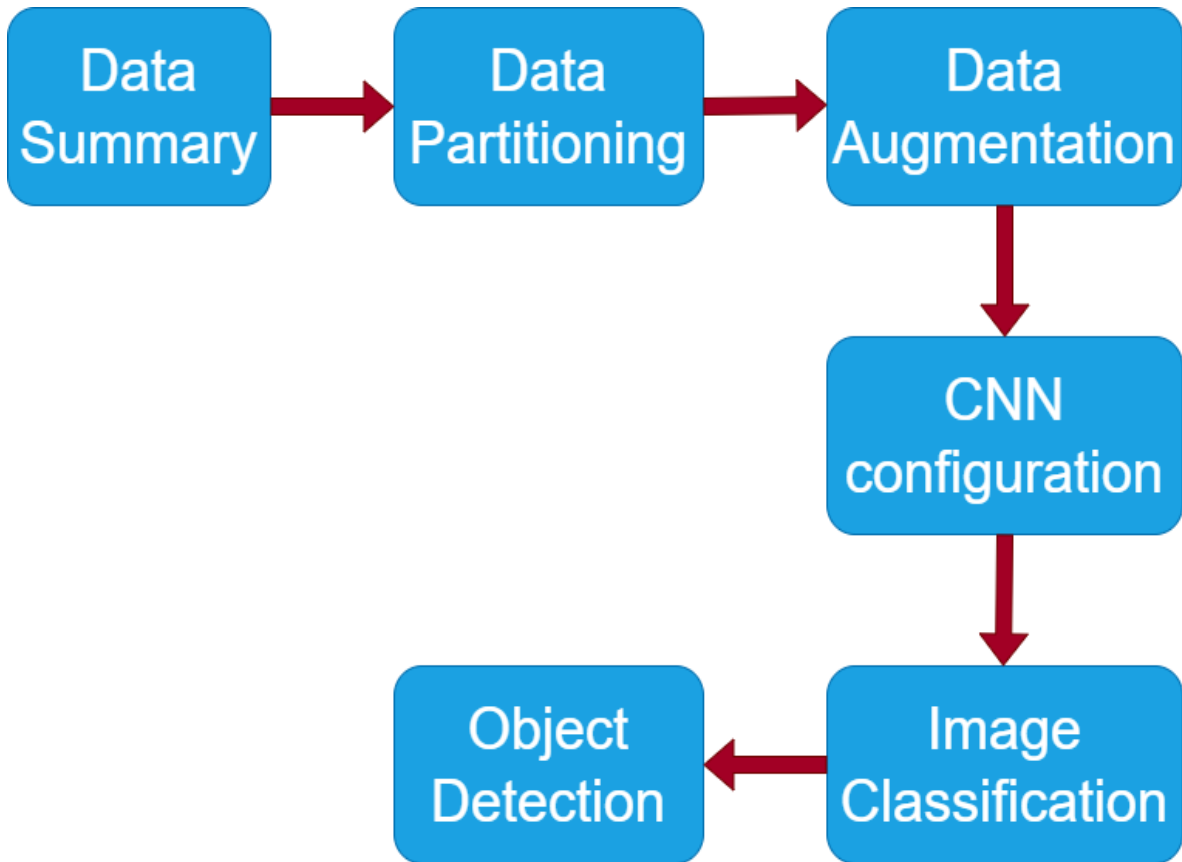


Figure 16: Overview of Chapter 4

4. 2 Data Summary

The data used in this study was gathered from the Kaggle repository (<https://www.kaggle.com/>). The dataset includes four animal classes: Buffalo, Elephant, Rhino, and Zebra. The four animal classes of the dataset are shown in Figure 17.

A



B



C



D



Figure 17: Animal Classes. 17.A Buffalo class, 17.B represents the elephant class, 17.C depicts the rhino class, and 17.D shows the zebra class.

The total number of images within this dataset was 1,504 (see section 3.4.1). There was a variety of image dimensions in this dataset, with image resolution ranging from (550 x 417) to (2592 x 1944), all stored in a jpg format. All images had a corresponding text file stored in a yolo annotation format vital for the object detection tasks. These images served as a foundation for the development of a deep learning model, that required appropriate image processing techniques to accurately perform

species classification and object detection tasks. Figure 18 shows the summary of the dataset.

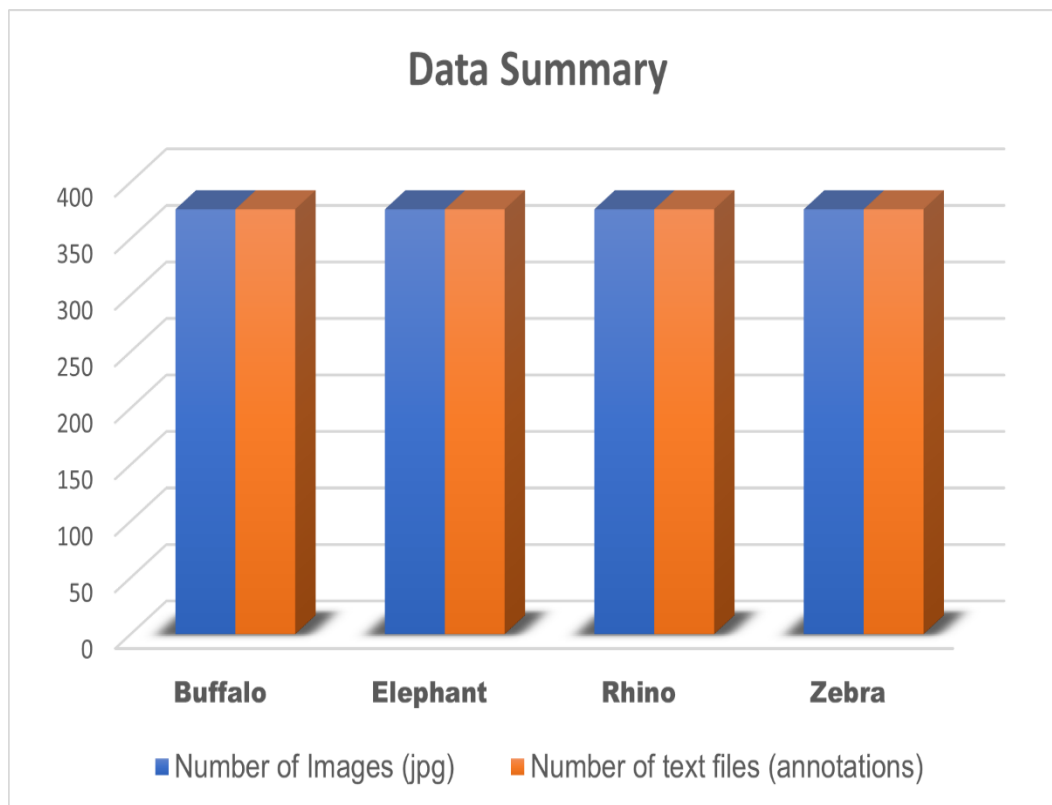


Figure 18: Summary of Dataset

4.3 Partitioning the data into training, validation, and test sets.

CNN models measure statistical patterns in the data, which include image features. It is imperative to maintain consistent statistical distribution for training and testing purposes (see section 3.3.1). With the goal of partitioning data in three sets to allow fair network learning of image features and testing, and the reduction of overfitting, the dataset was split into training, validation, and test sets using the gathered images. In ML, it is a general practice to divide the dataset into either an 80:10:10 or 70:20:10 ratio split for the training, validation, and test sets, respectively [90]. This

division was implemented to ensure that the validation and test sets were entirely apart and unseen in the training data.

4.3.1 Dataset division for image classification

The manner of training and validation data separation in image classification and object detection models varies from their distinct data loading framework. In image classification, the training, validation, and test sets are organised into separate directories, and their division is done randomly in the code according to specific sequence, such as the 70:20:10 split. Table 2 shows the distribution of images and their corresponding classes according to the train, validation, and test set data splits.

Table 2: Image distribution according to the train, validation, and test data splits

Dataset	Buffalo	Elephant	Rhino	Zebra	Total of Images
Train	264	264	264	264	1,056
validation	72	72	72	72	288
test	40	40	40	40	160

The dataset summary after the data partitioning for image classification models has the outcome of the animal classes for the respective data subsets as shown in Figure 19. The sum of images for each set included 1,056 images for training, 288 images for validation, and 160 images for the test set.

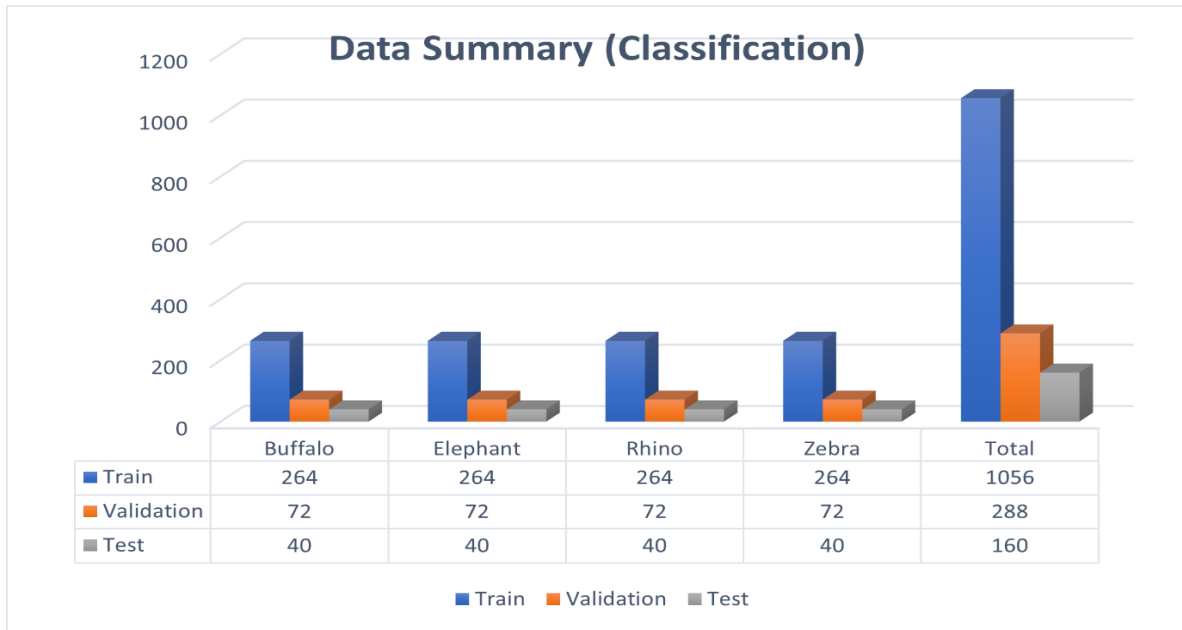


Figure 19: Data Summary for image classification after partitioning

4.3.2 Dataset partitioning for object detection.

For object detection, partitioning was applied to both images and text files that were gathered, unlike for image classification whereby only images were used. The text files included a YOLO labelling format, where each detectable instance was listed on individual lines along with their respective classes. Table 3 displays the contents of the text file, primarily storing information about the label category and the coordinates of the bounding box.

Table 3: Label information of a text file

Class	x	y	w	h
0	0.717969	0.605405	0.351563	0.324324
1	0.314063	0.474324	0.434375	0.510811
3	0.235938	0.657432	0.148438	0.366216

Individual lines in the YOLO annotation format text file presents information about an object, including its class and position. The first column indicates the object's class, while the subsequent four columns provide information about its position, specifically the central coordinates (x, y) and the dimensions (w: width, and h: heights) of the object. Each image has an associated text file, and multiple object classes could be present withing a single file. Table 3 serves as perfect example as it illustrates label category and coordinates for three distinct classes which included buffalo, elephant, and zebra. However, the rhino class coordinates were not shown because it was not in the image. Details regarding the YOLO annotation classes discussed in this study are shown on Table 4.

Table 4: YOLO annotation classes that represent the animal species.

Annotation Class Information of YOLO	
0	Buffalo
1	Elephant
2	Rhino
3	Zebra

Figure 20 Illustrates animal images A: without bounding boxes, and B: images of animals with the corresponding bounding boxes.

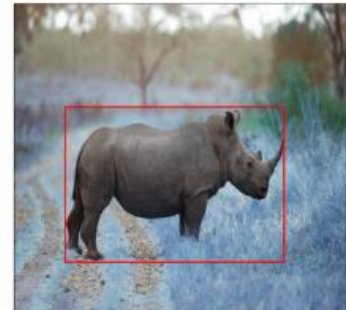
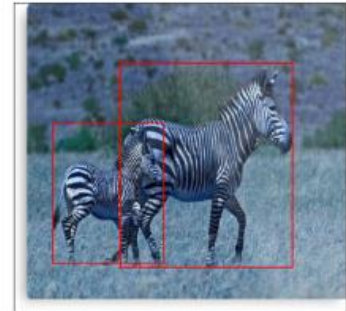
A**B**

Figure 20: Bounding Box images. A shows animal images without bounding boxes and 20.B shows an image of an object with the corresponding bounding boxes.

The dataset consisted of a total of 1,504 images as illustrated in Table 5. The training, validation and test sets were split in a 79:20:10 ratio, resulting in 1,189 images allocated for training, 297 for validation, and 18 for the test set. These sets were employed for testing and evaluating the proposed algorithms. During the partitioning process, it is vital to aim for a balance in the number of samples across

classes withing the training, validation, and test sets, while also striving to maintain consistent data allocation.

Table 5: Data partitioning summary for object detection

Dataset	Buffalo	Elephant	Rhino	Zebra	Total of Images
Train	295	297	298	299	1,189
validation	74	74	75	74	297
test	7	5	3	3	18

4.4 Data Augmentation

The critical step into building an optimal model for species identification lies in the preprocessing phase. Using augmentation, the model was fed with a variety of transformation so that it learns features in various forms during the training phase. This approach assisted in enhancing the generalisation ability of the models to predict unseen images of target classes. Furthermore, it aided in minimising the risk of overfitting. Two augmentation toolkits were used, given that classification tasks were performed using both Keras and Pytorch (see section 3.3.4)

4.4.1 Augmentation using Torchvision.

In this study, augmentation was performed using the Torchvision library in python for data manipulation to enhance model training of Alexnet and ResNet architectures. Torchvision uses transforms, which are image transformations offered by the *torchvision.transform* module, used for offline preprocessing and augmentation tasks [91]. These transforms are chained together using composed classes to allow full control over the transformations. In this chained pipeline approach, a sequence of transformation, which includes resize, rotation,

horizontalflip, converting to tensor and image normalisation were created. Table 6 summarises parameter and a variety of augmentations using Torchvision applied in the dataset.

Table 6: Summary of augmentation parameters used for Torchvision.

Augmentation method	Parameters for Training set	Parameters for Validation set	Parameters for Test set	Species
Resize	Size=256	Size=256	Size=256	All
Rotation	15	-	-	All
HorizontalFlip	True	False	False	All
CenterCrop	Size=224	Size=224	Size=224	All
ToTensor	True	True	True	All
Normalize	[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]	[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]	[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]	All

Figure 21 shows the output of the Torchvision augmentation technique on random animal classes in the training dataset.



Figure 21: Output of the Torchvision augmentation technique

4.4.2 Augmentation using Keras.

For the Keras library, *keras.ImageDataGenerator* was used to perform resize to enhance image species size, quality, and shape to better improve model training. Keras was used to augment data for the VGG16 architecture. Through the creation of the *ImageDataGenerator* class, the type of augmentations to apply for the data was defined. This included loading the data by specifying the directory, defined width, and the height parameters of images for the model training, validation, and testing tasks. The *class_mode* parameter was set to none since there was no use of labels (annotations) for the classification tasks. Table 7 summarises parameter and a variety of augmentations using Keras *ImageDataGenerator* class applied in the dataset.

Table 7: Summary of parameters used in Keras augmentation.

Augmentation method	Parameters for Training set	Parameters for Validation set	Parameters for Test set	Species
Specified directory	True	True	True	All
Target size (width and height)	Width = 224 and height = 224	Width = 224 and height = 224	Width = 224 and height = 224	All
Class mode	False	False	False	All

Figure 22 illustrates the outcome of the Keras ImageDataGenerator technique on the different animal classes.

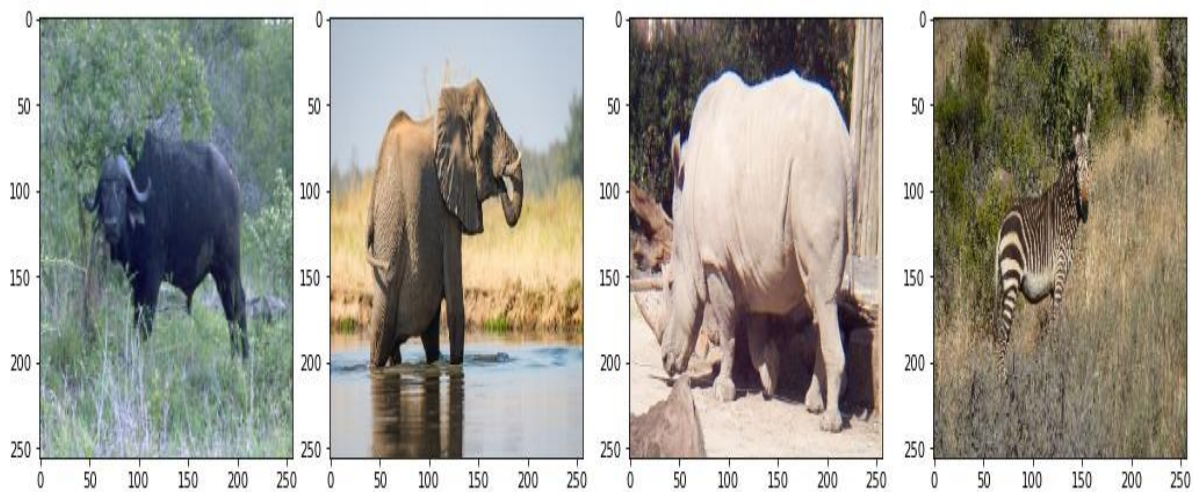


Figure 22: Output of the Keras ImageDataGenerator

4.5 Configuration of the CNN architectures

In this study, the three distinct CNN architectures were used, namely, Alexnet, ResNet, and VGG16 for the image classification and YOLOv8 for object detection.

Classification architectures were carefully chosen and evaluated regarding the available training data and the applicability of transfer learning. Initially, both Alexnet and ResNets were built and trained using augmented data generated by the torchvision library, while the VGG16 architecture was trained using augmented data produced by the Keras ImageDataGenerator. To enhance performance, various data augmentation techniques were integrated into the preprocessing stage of these CNN architectures, as detailed in section 4.3.

The sequential convolutional component of the CNN algorithm remained consistent across all three models. However, notable differences transpire when implementing the two distinct types of data augmentation methods explored in this study. The purpose of applying these two distinct augmentation techniques was exploring the impact of augmentation strategies on the camera trap dataset, and assessing the performance, in pursuit of an improved classification solution. Figure 23 shows the overall species identification structure of the CNN architecture implemented in this study.

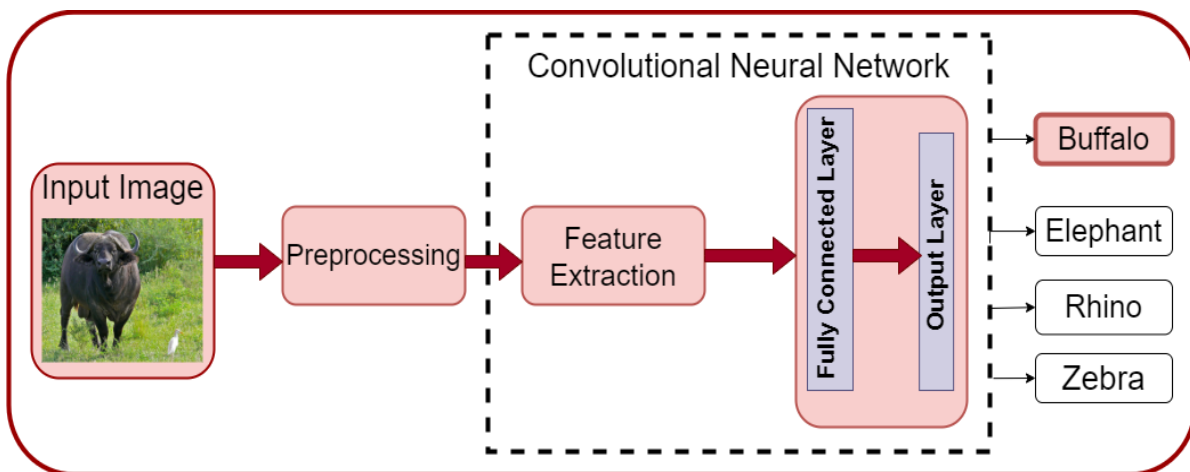


Figure 23: Species identification structure of the CNN architecture

The structure of the CNN network involves arranging a sequence of Convolutional layers (Conv) and subsampling layers (consisting of Batch Normalization, Activation, Pooling, and Dropout), then followed by Fully Connected (FC) layers in the end.

The conv layer, serving as the fundamental component of the CNN, uses learnable filters to extract localised features within images. Collectively, Batch Normalisation (BN) layers are introduced after each conv layer, executing the normalisation of each training mini batch with the application of lifted learning rates [77]. This step improves accuracy and accelerates the training process. Each BN layer is succeeded by an Activation layer, particularly ReLU, which establishes non-linearity, enabling the architecture to learn intricate data connections. Succeeding the activation layer are the Pooling (Max Pool) layers. These layers are integrated to compress or generalise feature descriptions by selecting the highest input values. Following the pooling layers are the dropout layers, which are incorporated to mitigate overfitting and enhance the generalisation capability in CNNs [70]. The 2D images are then transformed into 1D vectors for the convenience of FC layers. Finally, two Dense layers, along with a dropout layer and a sigmoid activation function, are added to the architecture. Figure 24 shows the configuration of a CNN architecture.

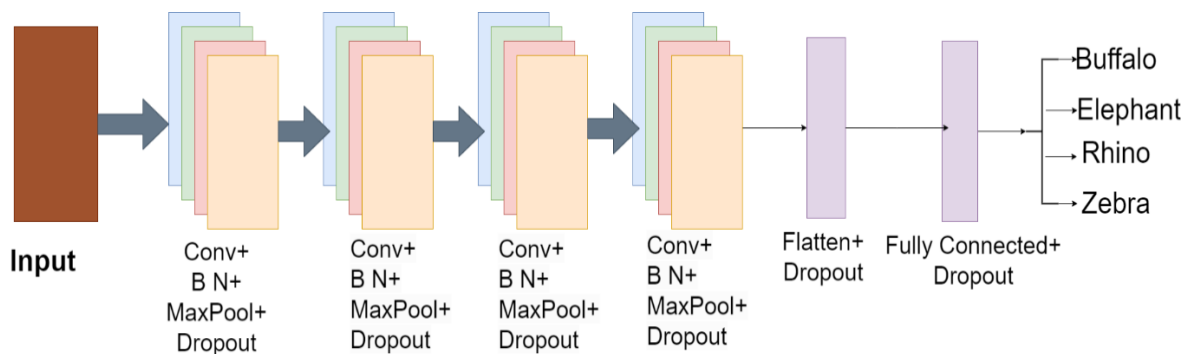


Figure 24: Configuration of a CNN architecture

Since the study aimed to identify an optimal CNN model for species identification, various model parameters were adjusted independently in each experiment, and the

outcomes were compared to determine the model with the highest obtainable accuracy. The three architectures employed in this study varied in terms of the number of layers and parameters. The specific design of these models for classification and object detection tasks is detailed in the subsequent sections.

4.6 Image Classification

Conducting image classification experiments by integrating transfer learning (see section 3.4.2) with CNNs enhances performance in a different and unaccustomed task, by leveraging previously acquired knowledge from related tasks. This technique has proven to be exceptionally beneficial in the context of species image analysis, tackling the issues that arise with using limited data, and computational resources. However, it is worth noting that, in several studies, the configuration of transfer learning has been employed to address problems outside the species identification spectrum.

4.6.1 Alexnet Architecture

In the domain of deep Convolutional Neural Network (DCNN) architecture, Alexnet holds a distinguished position [49], owing to its cutting-edge achievements in the fields of image recognition and classification. The total count of layers is 25, encompassing convolutional layers, pooling layers, a pair of ReLU activations, normalization, fully connected layers, softmax layers, and more [92, 93]. The Alexnet architecture is illustrated in Figure 25.

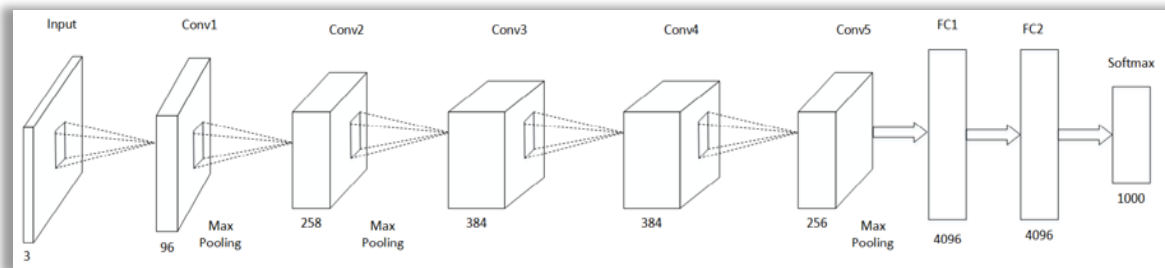


Figure 25: Alexnet architecture

The Alexnet structure encompasses 62 million trainable parameters. The network is proposed to directly handle RGB images, consistently processed in batches of dimensions $227 \times 227 \times 3$. The resulting outputs yield a probability vector size of 1000×1 , with each element matching to a class [94]. In this study, there was a total of 4 outputs representing individual classes conducted in the experiment. Table 8 below displays the parameters and hyperparameters for the Alexnet architecture used in this study.

Table 8: Parameters for Alexnet architecture

Layer	Map	Tensor Size	Kernel size	Stride	Activation	Parameters
Input image	1	227x227x3				
Convolutional 1	96	55x55x96	11x11	4	ReLU	34,944
Max Pooling	96	27x27x96	3x3	2	ReLU	0
Convolutional 2	256	27x27x256	5x5	1	ReLU	614,656
Max Pooling	256	13x13x256	3x3	2	ReLU	0
Convolutional 3	384	13x13x384	3x3	1	ReLU	885,120
Convolutional 4	384	13x13x384	3x3	1	ReLU	1,327,488
Convolutional 5	256	13x13x256	3x3	1	ReLU	884,992
Max Pooling	256	6x6x256	3x3	2	ReLU	0
Dense 1	-	4096x1	-	-	ReLU	37,752,832
Dense 2	-	4096x1	-	-	ReLU	16,781,312
Dense 3	-	4096x1	-	-	ReLU	4,097,000
Output Dense	-	1000(4)	-	-	Softmax	-

4.6.2 Residual Network (ResNet) Architecture

The current configurations of the neural network designs constructs layers to approximate some mapping function from an input image to a target specified as $H(x)$. Fundamentally, it approximates the residual of this function that can be considered as $F(x) = H(x) - x$. To recover the original function, one simply adds the residual back in the form of input: $F(x) + x$. Fundamentally, these layers are referred to as Residual Blocks, and are given the task to estimate the residuals and then re-insert the immediate input to reconstruct the actual target function.

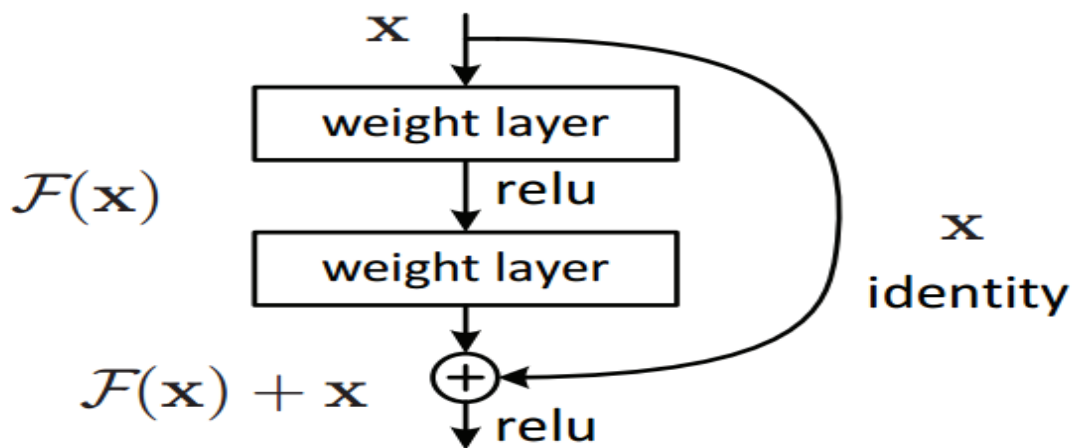


Figure 26: Residual block [49]

As shown in Figure 26, the architecture of this block features a skip-connection within the network. This skip-connection incorporates the input of stacked layers to the output of the same stack of layers. It is imperative to note that $F(x)$ and x may not have the same dimensionality, requiring a linear projection to match the dimensions of the original input x . This concept of residual learning has led to the enhancements of different ResNet-based models, which vary in the quantity of layers and compositions of building blocks. These networks employ residual blocks, whereby each possesses a skip connection incorporated within their framework. Notable

examples include ResNet34 and ResNet50 networks, which employ distinct types of residual blocks.

Residual Networks (ResNets) [95] are deep convolutional networks that incorporate a key concept by skipping blocks of convolutional layers through shortcut connections. These networks employ building blocks referred to as ‘bottleneck’ blocks, which adhere to two fundamental design rules; (1) maintaining an equal count of filters across layers for identical output feature map sizes, and (2) doubling the number of filters if the feature map size is halved. There exist various versions of ResNet architectures denoted by the term ‘XX’, which indicate the count of layers. For instance, ResNet50 contains 50 layers within the network. The table below provides information about the network’s parameters, totalling 24 million, all of which can be trained [95, 96]. Details such as padding, stride, input layer, and output layers are shown in Table 9. While the default output of 1000 is indicated in the table, for this study’s experiment, it is 4, the same as Alexnet.

Table 9: Parameter information of ResNets

Layer	Output size	18-layer	34-layer	50-layer
Convolutional 1	112×112	$7 \times 7, 64$, stride 2		
Convolutional 2_x	56×56	3×3 , max pool, stride 2		
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Convolutional 3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
Convolutional 4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$

Convolutional 5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9

4.6.3 VGG16 Architecture

In this study, a pre-trained VGG16 CNN model was employed. This model was fine-tuned by freezing certain layers to mitigate overfitting, granted that the dataset employed in this study was small. The VGG16 model consisted of 16 convolutional layers. As shown in Table 10, the input image for this network possessed a dimensionality of $(224 \times 224 \times 3)$ and consisted of 16 convolutional layers with a fixed-size filters of (3×3) and 5 max-pooling layers with a size of (3×3) throughout the entire network. At the top, the network integrated two fully connected layers together with a softmax output layer. The VGG16 model is considered an extensive network, encompassing approximately 138 million parameters. Its framework comprises multiple stacks of convolutional layers to develop DCNNs that excel at gathering hidden features. The architecture of the VGG16 network is illustrated in Figure 27.

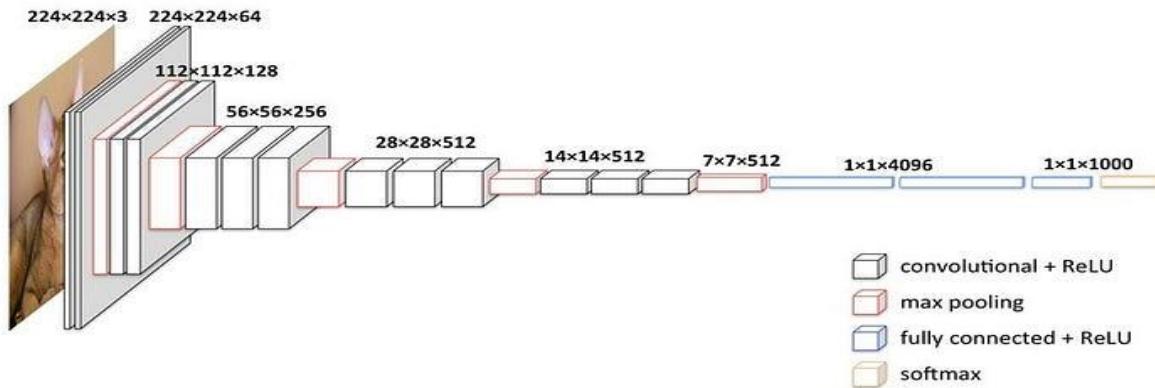


Figure 27: Structure of VGG16 architecture [52]

Table 10: Configuration of the VGG16 architecture

VGG-16 Configuration
16 weight layers
Input (224×224) RGB images
Convolutional 3-64 Convolutional 3-64
Max pool
Convolutional 3-128 Convolutional 3-128
Max pool
Convolutional 3-256 Convolutional 3-256 Convolutional 1-256
Max pool
Convolutional 3-512 Convolutional 3-512 Convolutional 1-512
Max pool
Convolutional 3-512 Convolutional 3-512 Convolutional 1-512
Max pool
FC-4096
FC-4096
FC-1000 (4)
softmax

4.6.4 CNN model training for image classification.

The process of DL is iterative, which allows the adjustment of different parameters to optimise the training speed and performance of the learning operation. These parameters include epochs, learning rate, batch size, and dropout values. Parameters can be tuned to enhance the performance of the CNN model. This study undertook extensive investigations to identify effective hyperparameters for evaluating accuracy and obtaining a suitable learning curve. This section explores the key hyperparameters that were used across the classification tasks.

Batch size

Batch size is the quantity of samples chosen from the training dataset, that are processed before the internal parameters of the model are modified [97]. Utilising mini-batch training accelerates the training procedure and facilitates rapid convergence. For the classifiers employed in this, batch sizes of 32, 34, 50 were selected for the model training.

Count of Epochs

Epochs are referred to as the number of times the complete training set goes through a CNN (see section 3.8). Generally, a CNN model is required up to a certain number of epochs depending mainly on the size of the dataset. Given the limited sample size in the gathered camera trap data, the network was run for 30 epochs across classifiers.

Learning rate

Learning rate (LR) serves as a vital hyperparameter in fine-tuning CNNs, facilitating the extent to which the weights of the optimising algorithm or optimiser are adjusted [80]. In this study, the LR of 0.0001 ($1e - 4$) was selected across all classifiers as it has proven to yield optimal outcomes for classification tasks.

Dropout

Dropout is another regularisation approach to mitigate overfitting (see section 3.6). The value of dropout was also fine-tuned following every convolutional and FC layers. For the Alexnet architecture, the dropout of 0.3 was consistent for all layers, for ResNet architectures, a consistent dropout of 0.4 was used across all layers, and for the VGG16 architecture, a dropout of 0.5 following convolutional layers and 0.3 following FC layer.

The selection for hyperparameter values for configuring the classifiers was a challenge due to the changing results based on the selected values. Hence, multiple experiments with various parameter values until optimal overall performance was achieved, were conducted. The table below shows the values of hyperparameters for classifiers.

Table 11: Hyperparameter configurations for classification experiments

CNN Models	Data	Batch Size	Count of Epochs	Learning Rate	Dropout	Optimizer
Alexnet	Train:70% Val:20% Test:10%	Train:32 Val:32 Test:32	30	0.0001 (1e-4)	Conv:03 FC:03	ADAM
ResNet-18	Train:70% Val:20% Test:10%	Train:32 Val:32 Test:32	30	0.0001 (1e-4)	Conv:04 FC:04	ADAM
ResNet-34	Train:70% Val:20% Test:10%	Train:34 Val:34 Test:34	30	0.0001 (1e-4)	Conv:04 FC:04	ADAM
Resnet-50	Train:70% Val:20% Test:10%	Train:50 Val:50 Test:50	30	0.0001 (1e-4)	Conv:04 FC:04	ADAM

VGG16	Train:70% Val:20% Test:10%	Train:50 Val:50 Test:50	30	0.0001 (1e-4)	Conv:05 FC:03	RMSprop
--------------	----------------------------------	-------------------------------	----	------------------	------------------	---------

4.7 Object Detection

As stated in section 3, the study conducted experiments by performing image classification and object detection tasks. In this section, the in-depth framework of the object detector was discussed. In the domain of DCNNs, You Only Look Once (YOLO) is a widely established object detection model. YOLO comprises multiple iterations, with each successive version building on the previous version. This study focused on the use of YOLOv8 model as the selected object detector. The details of the YOLOv8 detector are elaborated in the section below.

4.7.1 The YOLOv8 algorithm

YOLOv8 [98] shows an advanced version of the YOLO framework. It is developed through the integration of foundational approaches offered in YOLOv4 [99]. It incorporates components such as the Cross Stage Partial (CSP) approach from Wang et.al [100], the fusion of Path Aggregation Network (PAN) [101] and Feature Pyramid Network (FPN) [102], which is widely referred to as the PAN-FPN, as well as the spatial pyramid pooling fast (SPPF) module.

YOLOv8 encompasses distinct-scale models grounded on scaling coefficients, a feature inspired by YOLOv5 to accommodate diverse requirements of different tasks. While maintaining the fundamental concepts of YOLOv5, the YOLOv8 integrates the Coarse-To-Fine (C2F) module [103] that is inspired by the Efficient Layer Aggregation Network (ELAN) concept in the architecture of YOLOv7 [104]. Components that make up the architecture of the YOLOv8 include the input, backbone, neck, and the output.

Input

This primarily focuses on various augmentation techniques such as colour perturbation, spatial perturbation, mosaic, and mix-up. With the combination of these data augmentation techniques, different sets of images are joined together after processing a single image. This approach improves the diversity of object perspectives and refines the variety of image backgrounds. Figure 28 shows the YOLOv8 structure.

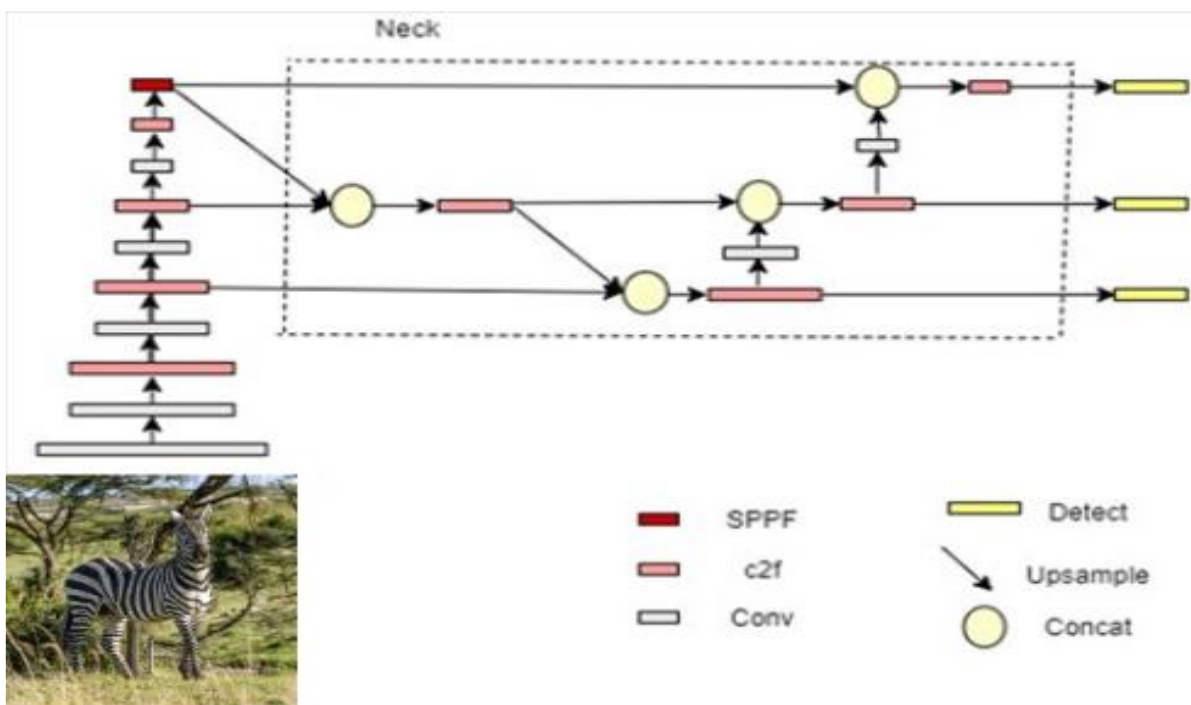


Figure 28: Structure of the YOLOv8 object detector

Backbone

The backbone comprises convolutional layers, the C2F layer, and the SPPF layer. The design of the C2F differs from the C3 module in YOLOv5, drawing inspiration from the ELAN concept. This modification effectively enhances gradient propagation and speeds up the network convergence. The SPPF maintains the structure from YOLOv5.

Neck

The neck of the network integrates a design that combines elements of PAN and FPN. Features from adjoining layers are connected and serve as input to the C2F module. As the features are transferred from both top to bottom and from bottom to top, first-class semantic features are merged with lower-class features.

Output

The output stage accomplishes the separation of object detection and classification. Lower-class features are used to capture information regarding small target objects, whereas first-class features are relevant to larger target objects. Individual detection layer generates a result vector consisting of location and matching class information.

4.7.2 Bounding Boxes for training set

The dataset discussed in section 4.3.2 contains images labelled in the YOLO format. This format uses a text file (.txt) for every image, which contains annotation specific to that image. These annotations outline details such as the class ID, object coordinates, height, and width for individual labelled object. Illustrations of the first and second training batches are captured in Figure 29 and Figure 30 respectively.

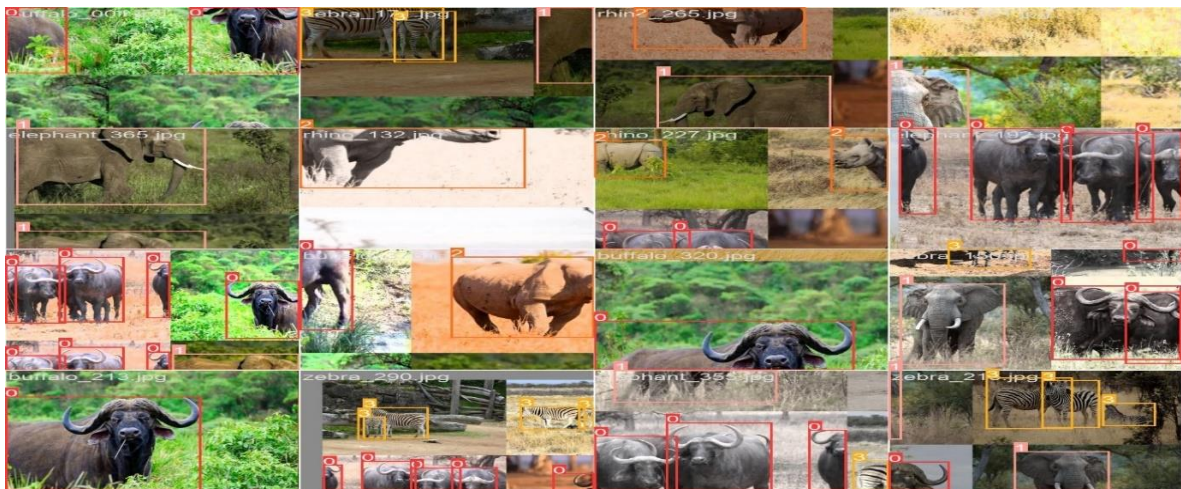


Figure 29: Training batch 1

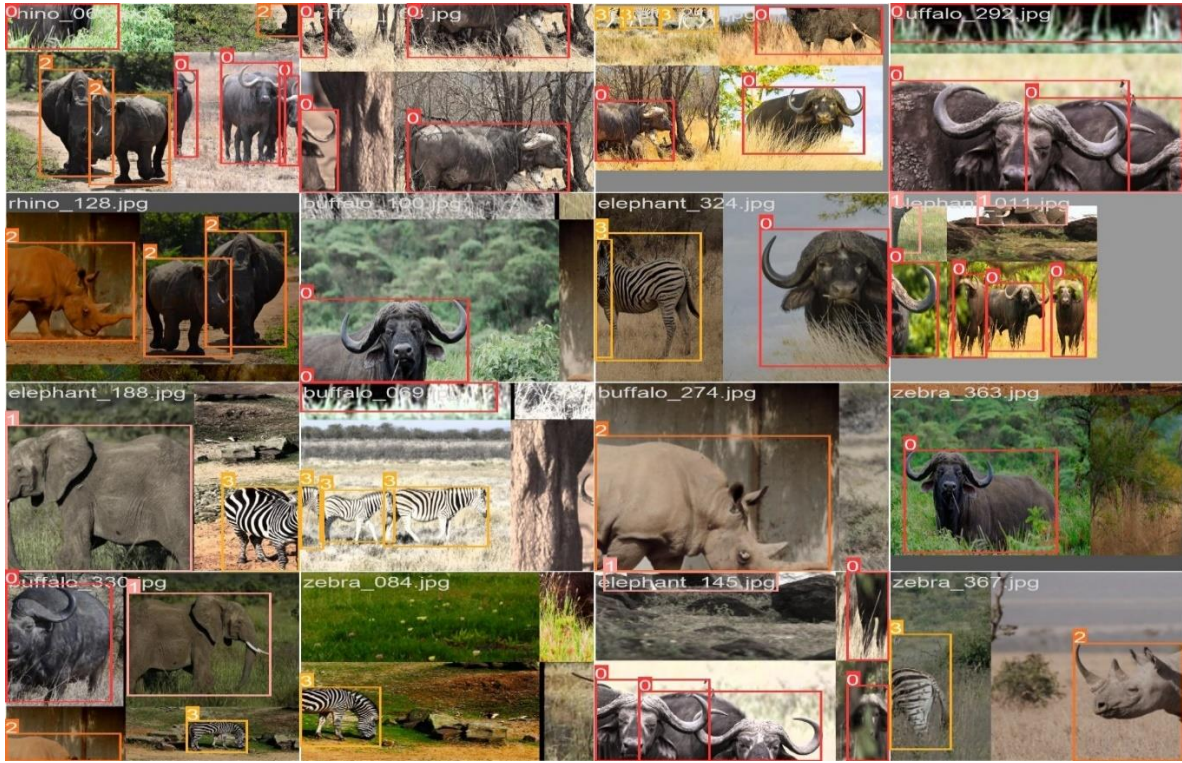


Figure 30: Training batch 2

4.7.3 Hyperparameters

As outlined in the previous section that the object detection was conducted using YOLOv8 model architecture. During the training phase, it is important to fine-tune hyperparameters of the model to achieve optimal performance. Nonetheless, due to the computational drawbacks, performing comprehensive experimentation of all possible hyperparameters is not feasible. Therefore, for the model architecture detailed in the previous section, this study focused on using fixed hyperparameters. Hyperparameters included a total of 100 epochs, Adam optimizer. Training parameters included a LR of 0.01, batch size of 16, momentum of 0.937 and weight decay of 0.0005. Table 12 shows the details of the hyperparameters.

Table 12: Hyperparameter configuration for the object detection model

Hyperparameter	Value
Epochs	100
Batch Size	16
Optimizer	Adam
Learning Rate	0.01
Momentum	0.937
Weight Decay	0.0005
Input Size	640 × 640

4.7.4 YOLOv8 Model Assessment

The YOLOv8 model was evaluated applying the proposed performance metrics for object detection. These include precision, recall, IoU, and mean average precision (mAP). Detailed description of these metrics was outlined in section 3.6.2.

Confidence threshold

When employing an object detection model, the confidence threshold is established to decide whether a detection is retained or removed, depending on the confidence score associated with each prediction. The confidence threshold chosen for this model was 0.0001.

IoU

The IoU (see section 3.6.2) plays a critical role in deciding what the model considers a TP detection depending on the degree of overlap between the predicted bounding box and the ground truth bounding box. In the study, an IoU threshold of 0.5 was selected. This will require a situation where at least 50% of the predicted bounding box overlaps with the ground truth bounding, it was observed as a true positive

detection. This decision aligned with the aim of species counting through the detection process, highlighting the identification of the object in the counting process.

Counting Assessment

The count is derived by tallying the model's detection for each animal class through the detected bounding boxes.

Weights selection

To establish the optimal weights for the model, an assessment of the mAP values was done. As the training process advanced, the model weights were updated after each backward pass. It is beneficial to take note of the number of epochs and continuous weight updates during the training phase. Once the training is completed, the model is evaluated using the mAP for each iteration of the model.

4.8 Computational Resources

Both classification and object detection models were coded using Python programming language. The table below shows the hardware and environment parameters.

Table 13: Computational Resources

Parameters	Configuration
CPU	13th Gen Intel Core (TM) i7-13700F
Operating System	Windows 11
Python version	3.11.4
torch	2.0.1

4.9 Conclusion

This chapter delved into the identification of the data and algorithms implemented in this work. The data gathered underwent preprocessing using techniques that allowed it to be fed into the deep learning models for classification and object detection. These techniques included the data partition approach, which is a common practice to separate the data into three sets, which included the training, validation, and test sets. After partitioning, the augmentation method was used to apply several transformation mechanisms that would allow the model to see the data from a variety of transformation to improve the classification models' generalisation capabilities. Augmentation was performed using the Keras ImageDataGenerator and Torchvision transforms libraries.

After the data preparation, deep learning algorithms were selected to perform classification and object detection tasks. These algorithms included pre-trained classifiers such as Alexnet, ResNets, and VGG16. YOLOv8 was employed for object detection. These CNN architectures underwent structural configurations, as well as defining of the parameters and hyperparameter that would equip the models in accomplishing the goal of the study. In addition, the chapter focused on model evaluation for each model to observe its performance during the experimentation phase when performing species identification tasks.

Chapter 5: Results and Discussions

This chapter presents the outcomes of the image classification and object detection tasks discussed in this dissertation. Section 5.1 details the findings from the deep learning classification models, whereas Section 5.2 focuses on the results of the object detection model.

5.1 Image Classification results.

While the primary objective of this study was to identify distinct animal species within a camera trap dataset, the preceding phase of the research started by constructing and training CNN classifiers using the collected image data. The primary objective was to build a model for multiclass species recognition. This phase provided a comprehensive learning experience by generating results and providing insights into the complexities of image analysis.

During the construction of the multi-classification model, the training comprised four different animal species: buffalo, elephant, rhino, and zebra. The dataset included 1,344 samples for training and validation and 160 samples for testing.

Hyperparameter permutations were employed to ensure that the CNN classifiers were equipped to effectively extract features from each class. For the Alexnet, ResNet-18, ResNet-34, ResNet-50 and VGG16 models incorporated in this study, a final layer was altered by integrating the model with the transfer learning and changed the output features from 1000 to 4, given the number of animal classes, and adding a final softmax filter for the output of neurons. This technique increases the performance of the model because it involved the reduction of these neurons within the dense layer.

The model performance of a model can be assessed by how well it recognises unseen data from the dataset sample allocation used for the training phase. This is

essentially a measure of the CNN classifier's capability to generalise. In the circumstance of experimentation with camera trap images, once a model is trained, its performance undergoes validation using a particular set of samples designed to evaluate its ability to generalize across the trained set. The evaluation of the classification models involved a unique testing sample, and the results were used to produce a confusion matrix, which provides an analysis of the classification predictions, including both correct and incorrect label estimations.

In addition to the learning curve and confusion matrix, the model was employed to predict unseen images from the saved model. After training, the Alexnet, ResNet 18, 34, and 50 were saved using the Torch function *torch.save*, which stored the models that were built using Pytorch. For the VGG16 model, this study employed the Keras function *model.save*, which stores the weights of the model in a file. Thereafter, the *load_model* function was used to load the model directly to predict new and unseen images that were not part of the training and validation sets. This was done for testing purposes of a separate set of data samples that included 160 images for all the animal classes. The following sections will delve into the analysis of the results for the Alexnet, ResNets and VGG16 models.

5.1.1 Evaluation of image classification model performance

During experimentation, the issue of working with a small dataset was eminent, so it required techniques that assisted classifiers with knowledge and equipped models with the fundamental mechanisms to carry out the related tasks. Prior to training, image samples were preprocessed through the employment of distinct data augmentation tool to encourage variety of this technique. During model training, the transfer learning approach was implemented to aid in providing necessary assistance, allowing models to serve as established feature extractors through the replacements of final layers. This improved the model's capability within reduced training time.

Several hyperparameters were fine-tuned to construct an optimal model. These hyperparameters included batch size, number of epochs, learning rate, dropout, and optimizer. Dropout layers used in the experimentation incorporated values of 0.3, 0.4, and 0.5. The Alexnet results accomplished using 0.3 across the CNN layers, ResNets employing 0.4 across CNN layers, and VGG16 architecture, used a dropout of 0.5 following convolutional layers and 0.3 following FC layer. For Pytorch model architectures, which include Alexnet and ResNets, the Negative Log-Likelihood function [105] was used to compile their architectures and employed Adam optimizer. The VGG16 was constructed with categorical cross-entropy loss function and RMSprop was employed as the optimiser. The learning rate of 0.0001 was used for all models. The batch size ranged from 32 for Alexnet and ResNet-18, 34 for ResNet-34, and 50 for ResNet-50 and VGG16.

The results were displayed using the learning curve and confusion matrix. Learning curves in the context of CNNs are visual representations that show how a deep learning model performs over a period when it is trained on a train set [90]. The training and validation accuracy was measured to assess the models' accuracy of accurately predicting instances on the training and validation datasets as well as the number of samples that were classified correctly by the model. The training and validation loss assesses the error between predicted and actual values in the training set, as well as how good the model generalises to the data points unseen during training. The loss played a critical role in evaluating the models' ability to execute accurate predictions on new and unseen data. In this study, illustrated curves of accuracy and loss were plotted against the number of epochs. These curves were implemented with an objective to observe the training progress of each model throughout the stated number of epochs.

5.1.2 Alexnet results

Results for Alexnet are illustrated in Figures 31.A, 31.B, and 31.C.

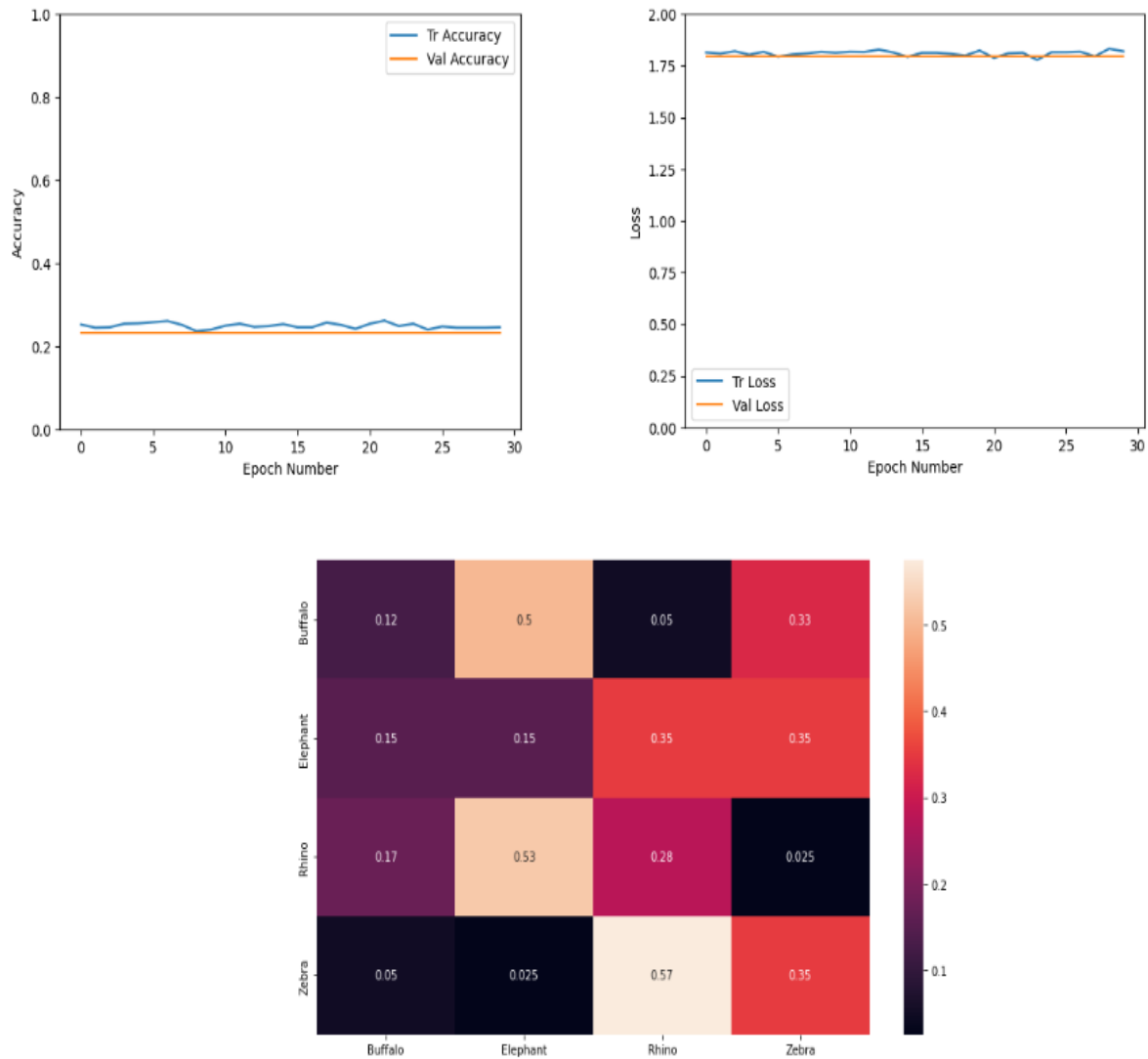


Figure 31: Alexnet results: 31.A Accuracy curve, 31.B Loss curve, and 31.C Confusion matrix

For classification, a good fit is described by training and validation loss curves that decrease and stabilise with a minimal gap between the final loss values. The learning

curves results shown in Figures 31.A and 31.B, as well as the confusion matrix in Figure 31.C present the results of the multiclassification experiment.

The model encountered challenges in distinguishing between and among species in the test dataset. Among the four species, zebra and rhino predictions exhibited relatively better performance, whereas buffalo and elephant predictions were notably poorer. Unfortunately, the model did not achieve the desired level of accuracy, as it misclassified elephants as rhinos and zebras, and misclassified buffalo samples as zebras. Figure 32 shows the same predictions of Alexnet on the rhino class.

```
Prediction 1 : rhino , Score: 0.4765948  
Prediction 2 : buffalo , Score: 0.23578915  
Prediction 3 : zebra , Score: 0.15797041
```

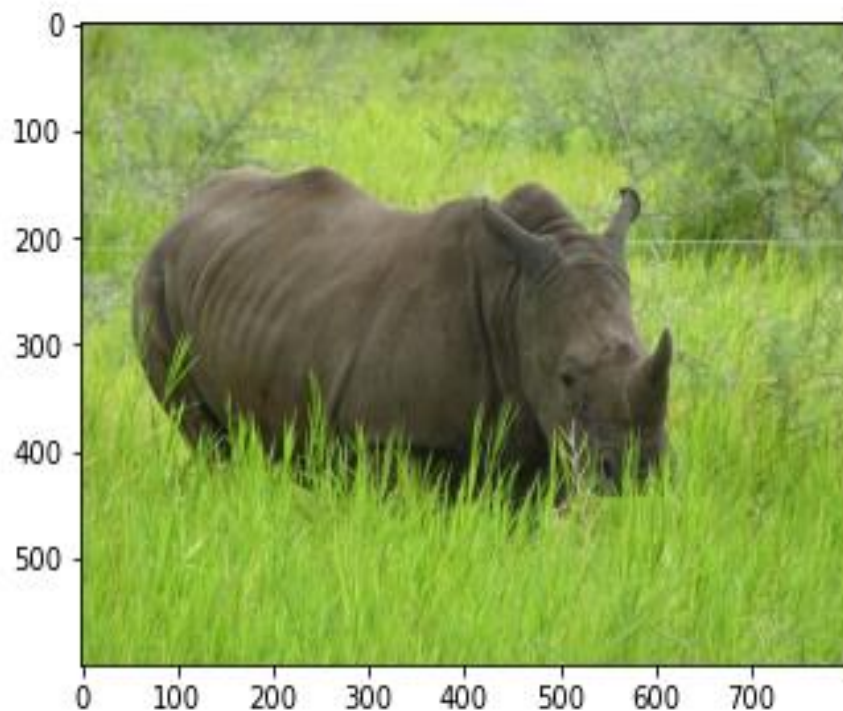


Figure 32: Alexnet prediction results on rhino class

5.1.3 Results of ResNet models

The ResNet architectures, which included ResNet-18, 34, and 50 were employed in this study for the classification experiments. For the classification, ResNets performed better than the Alexnet model. The results of ResNet models were also using the learning curve and confusion matrix to analyse performance for distinct animal species that were predicted. Figures 33, 34 and 35 show the results of ResNet architectures 18, 34, and 50 respectively.

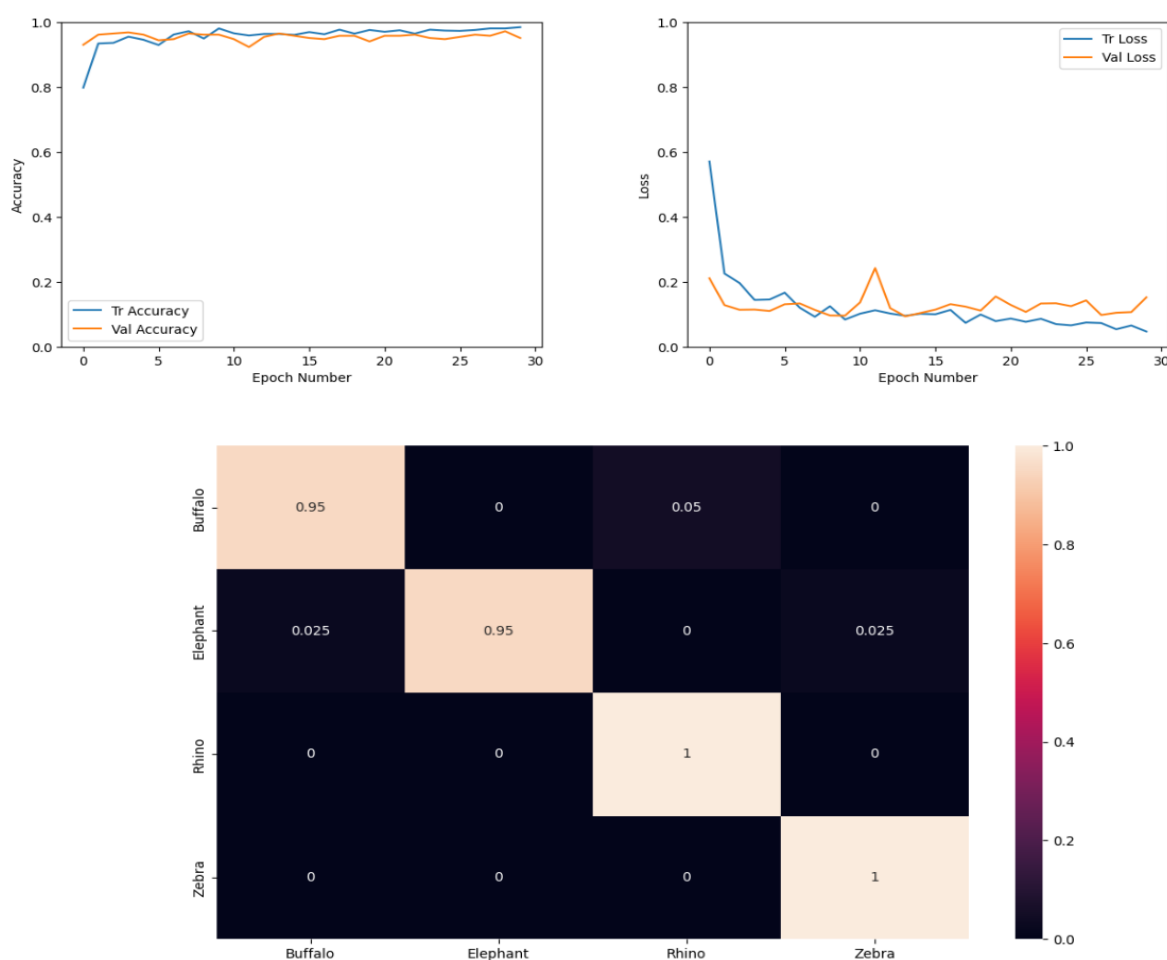


Figure 33: ResNet-18 results: 33.A Accuracy curve, 33.B Loss curve, and 33.C Confusion matrix

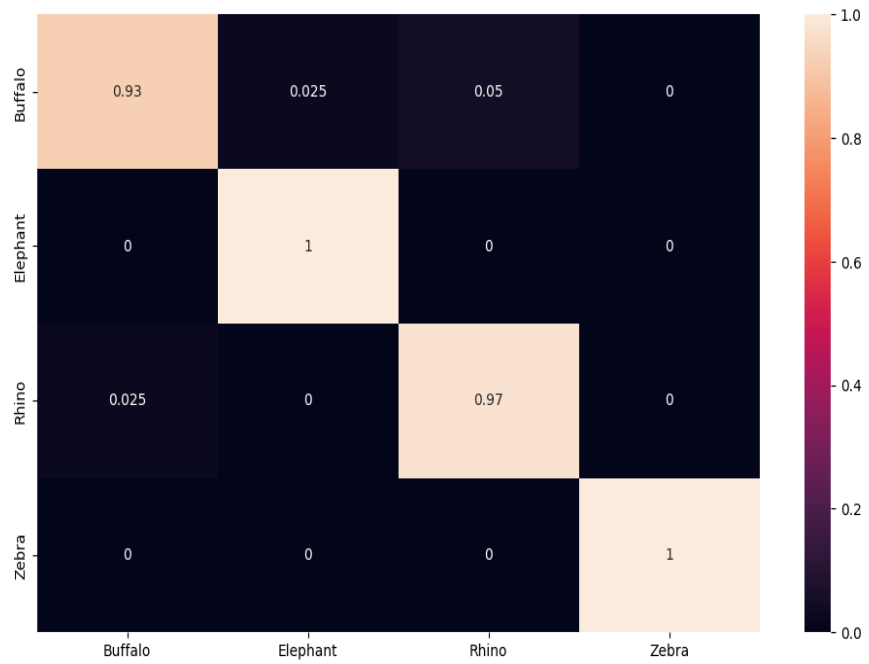
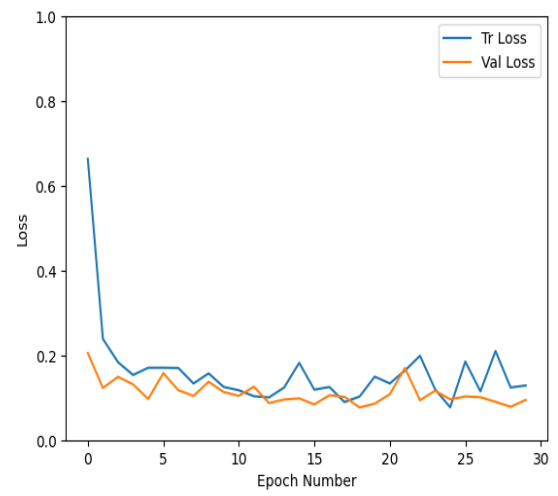
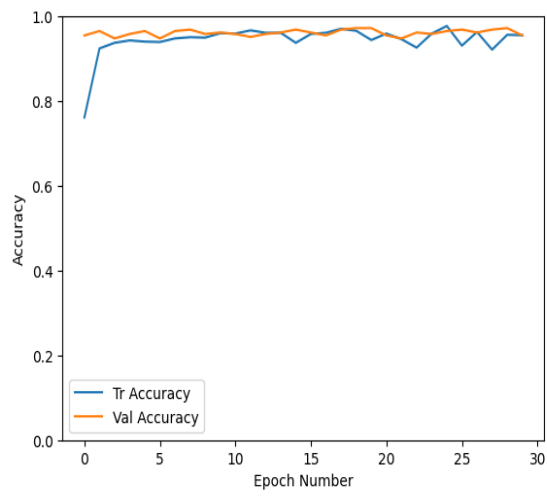


Figure 34: ResNet-34 results: 34.A Accuracy curve, 34.B Loss curve, and 34.C Confusion matrix

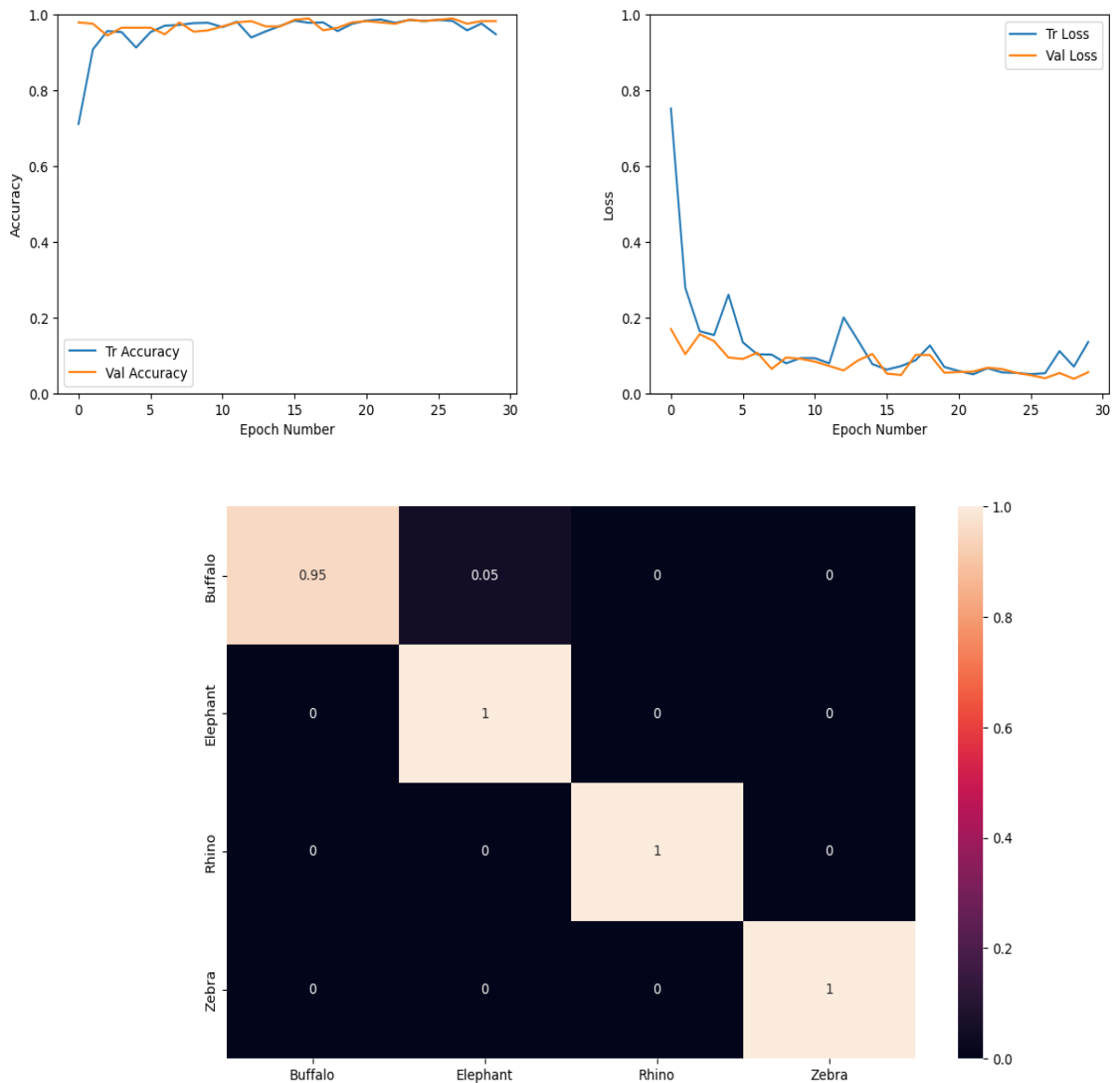


Figure 35: ResNet-50 results: 35.A Accuracy curve, 35.B Loss curve, and 35.C Confusion matrix

The learning curves and confusion matrix results for ResNet-18 are depicted in Figures 33.A, 33.B, and 33.C, while the learning curves and confusion matrix results for ResNet-34 can be seen in Figures 34.A, 34.B and 34.C. Lastly, the results for ResNet-50 are displayed in Figures 35.A, 35.B, along with the confusion matrix in

Figure 35.C. These figures summarise the performance of the ResNet architectures in the multi-classification experiment.

The ResNet architectures performed well, indicating that they were a good fit for the task, with only minor misclassifications. These models demonstrated minimal challenges during predictions among the species in the test dataset. Among the four animal species, overall predictions displayed relatively better performance, with very few instances where buffalo prediction were misclassified as elephants and rhinos. The models achieved the desired level of accuracy, and ResNet-50 particularly stood out among the ResNets proposed in this study.

Sample predictions of ResNet models

Figure 36 shows sample predictions of ResNet-18 on the test dataset.

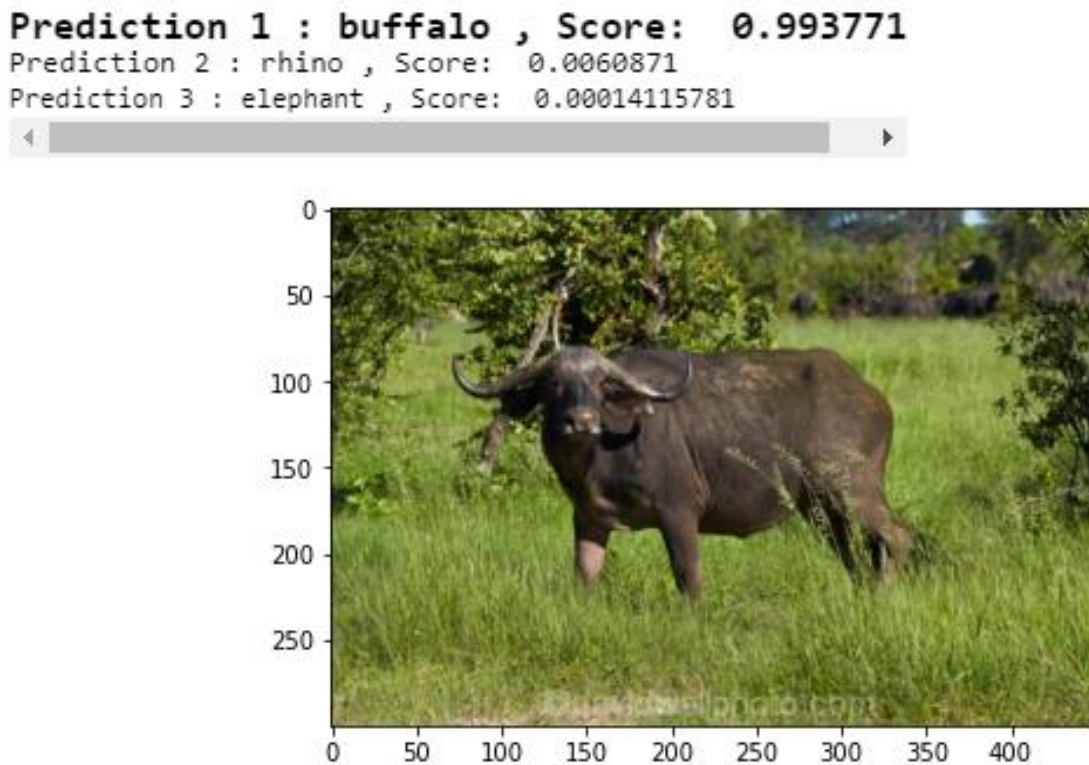


Figure 36: ResNet-18 sample prediction results

Figure 37 illustrates the sample predictions of ResNet-34 on the test dataset.

Prediction 1 : elephant , Score: 0.99474573

Prediction 2 : rhino , Score: 0.004964717

Prediction 3 : buffalo , Score: 0.0002394587

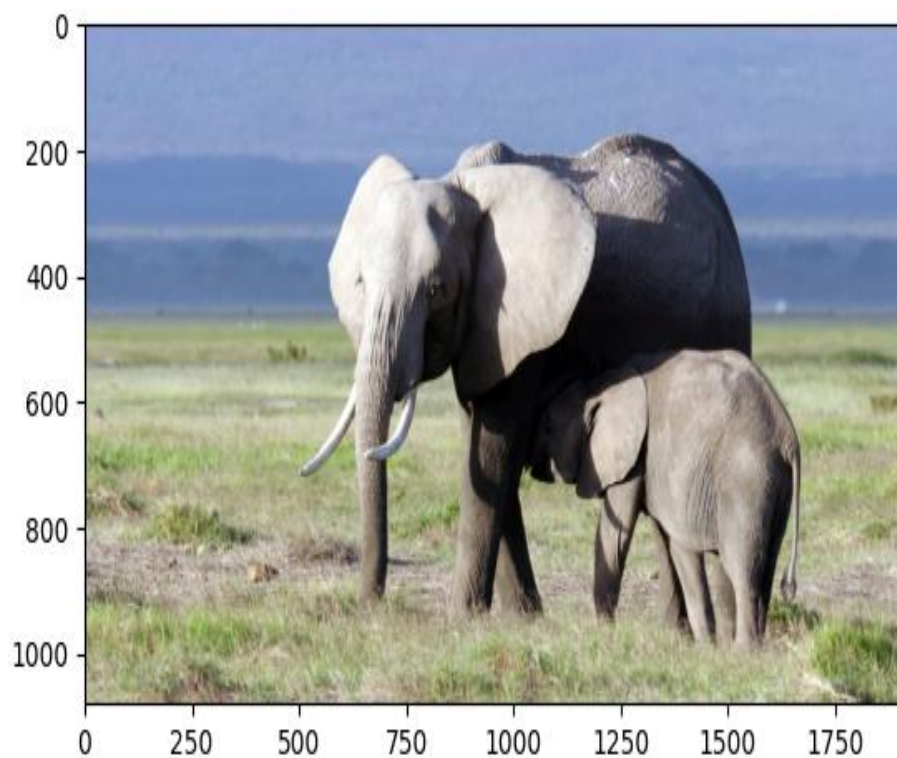


Figure 37: ResNet-34 sample prediction results

Figure 38 illustrates the sample predictions of ResNet-50 randomly on the test dataset.

Prediction 1 : zebra , Score: 0.9999995 
Prediction 2 : buffalo , Score: 4.8840184e-07
Prediction 3 : elephant , Score: 3.895697e-08

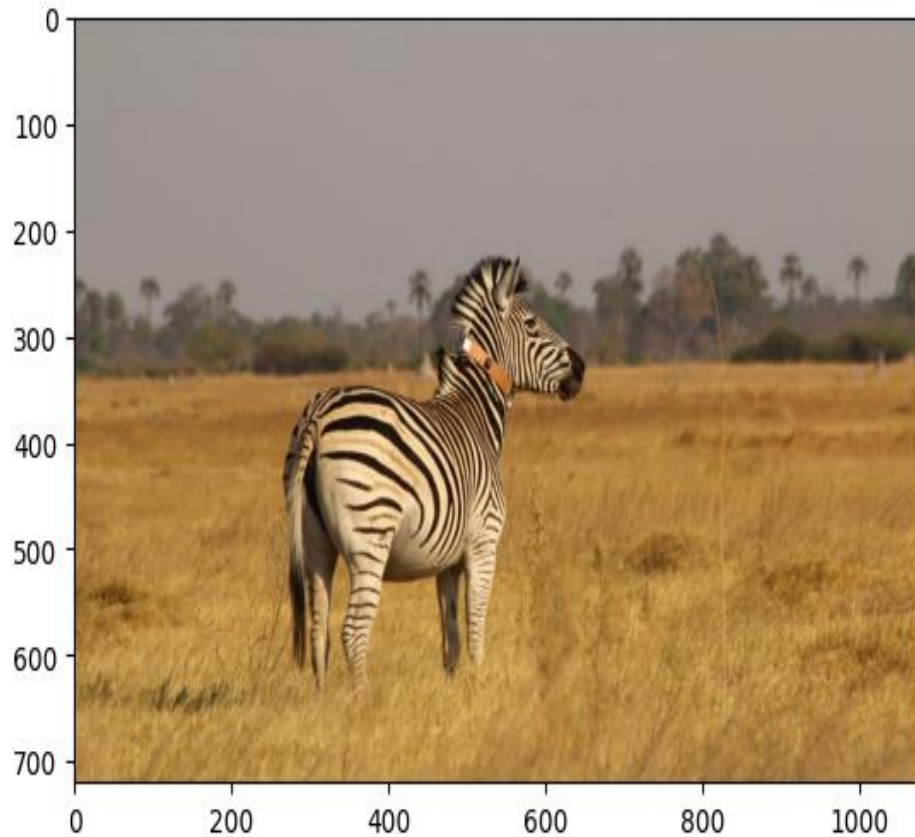


Figure 38: ResNet-50 sample prediction results

5.1.4 VGG16 results

The last architecture that was employed in this study's experimentations was the VGG16, which was implemented differently from the Alexnet and ResNets. The model was implemented using Keras rather than Pytorch, this included distinct

manner for fine-tuning hyperparameters as well. VGG16 performance was evaluated using the learning curves and confusion matrix to maintain consistency and fair comparisons with other model architectures in this study.

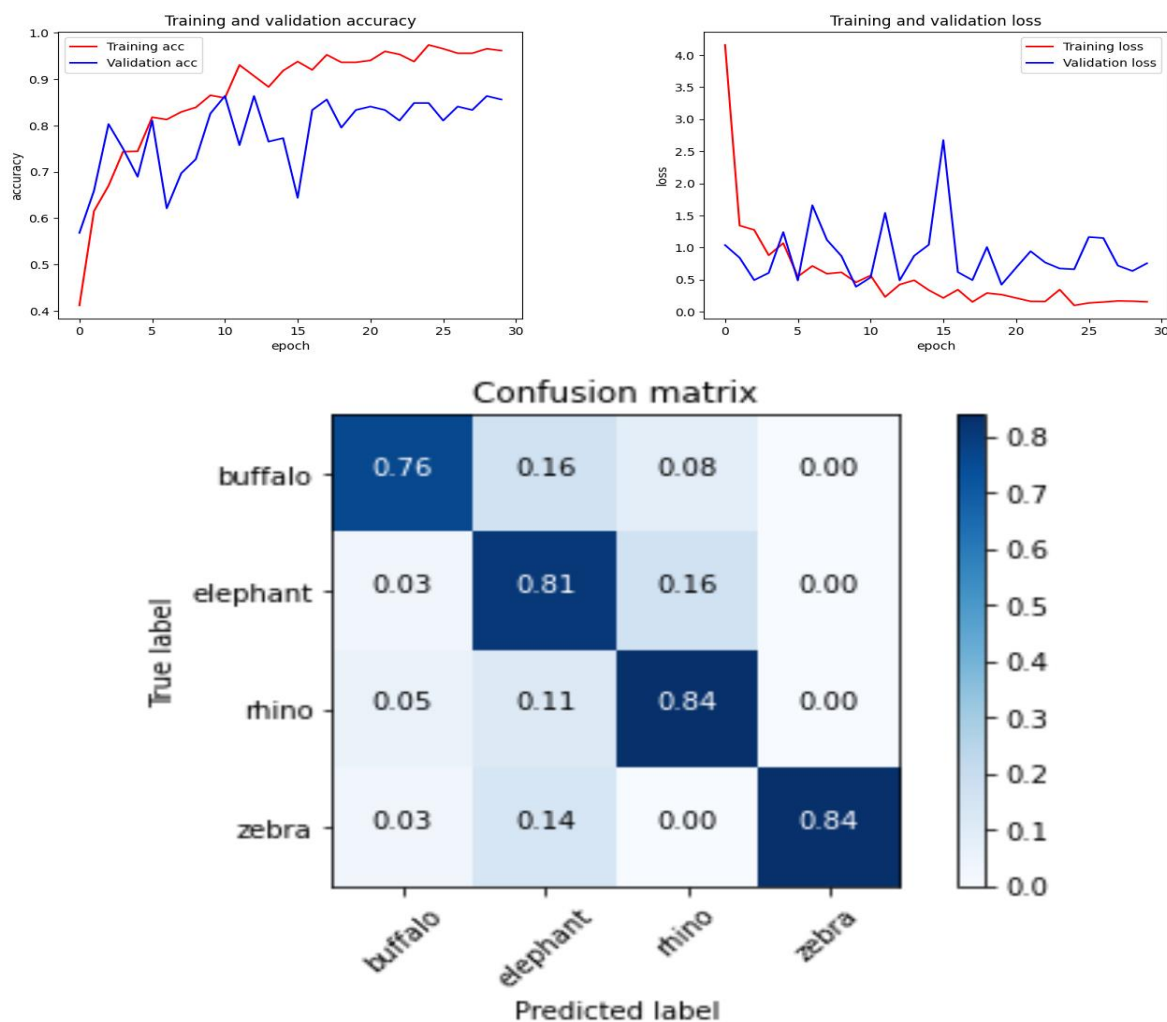


Figure 39: VGG16 model training. 39.A depicts the training vs validation accuracy curve, the training and validation loss in 39.B, and the confusion matrix was illustrated in Figure 39.C.

As shown in Figure 39, the training vs validation accuracy curve is depicted in 39.A, the training and validation loss in 39.B, and the confusion matrix is illustrated in 39.C. The VGG performed well, with both learning curves showing a good fit but with minor

inconsistencies. This model performed well on unseen data because of the high validation accuracy and low validation which are vital signs that symbolise good generalisation results. The minor inconsistencies reflected extensively in the confusion matrix whereby the overall prediction of the four distinct species was good, but there were some misclassifications of species. Misclassifications included buffalo samples predicted as elephant and elephants predicted as rhino. Figure 40 shows the prediction of VGG16 on the test set particularly on the buffalo class.

```
[INFO] loading and preprocessing image...
1/1 [=====] - 0s 125ms/step
1/1 [=====] - 0s 28ms/step
ID: 0, Label: buffalo 99.95%
ID: 1, Label: elephant 0.01%
ID: 2, Label: rhino 0.04%
ID: 3, Label: zebra 0.0%
Final Decision:
.
..
...
1/1 [=====] - 0s 33ms/step
ID: [9.9951410e-01 7.1252151e-05 3.9642048e-04 1.8261533e-05], Label: buffal
```

Out[34]:



Figure 40: Prediction of VGG16 on testing sample

The accuracy curves did not display much of overfitting since the validation curve aligns with the training curve. Furthermore, an assessment of the loss curves reveals a consistent decline towards a minimal value, with the validation loss curve starting to decrease after the fifth epoch and continuing until the end of the plot. In the case of ResNets and VGG16 models, the curves maintained a relatively stable learning process, while the Alexnet model showed a slight increase towards underfitting after 30 epochs. The table below summarises the outcomes of all classifiers.

Table 14: Overall performance of CNN classifiers

CNN model performance (Accuracy)			
Model	Training	Validation	Testing
Alexnet	26%	23%	24
ResNet-18	99%	96%	97%
ResNet-34	98%	97%	97%
ResNet-50	99%	98%	99%
VGG16	99%	85%	83%

The image classification predictions were conducted randomly among the four classes in the test set, and the first prediction which was tested on Alexnet performed poorly, and even on the best performing class of rhino it still had an output below 50%. For ResNet models, the first prediction was tested using Resnet-18 in the output shows the buffalo class being predicted with a score of 99.3%. This observation shows that the model performed well on the testing phase. ResNet-18 performed better than the Alexnet since the model performed well throughout the three phases.

However, compared to other ResNet versions, Resnet-18 trained much faster than the other two architectures, but came out last with a smaller margin based on performance. This was because models with more layers and higher batch sizes perform better, but with the disadvantage of having a higher training time. The same applied to ResNet-34 when compared to the ResNet-50 mode, whereby it came second to the deep-layered ResNet-50, with small difference in accuracy but performed better when it comes to training period. The VGG16, which was implemented on a different package, also displayed good results when compared to other architectures, especially ResNet architecture. It was much faster than all other architectures. However, with an accuracy of 99%, same as the ResNet architectures, it declined in accuracy on the validation and testing datasets. This made the ResNet-50 stand out among all classifier that were employed for image classification in this study.

5.2 Object detection results

This section presents the performance of the object detection model based on the validation and test sets. The following subsections provide comprehensive evaluation of the object detection model's performance and sample predictions for each animal class, buffalo, elephant, rhino, and zebra.

5.2.1 Bounding box analysis

A bounding box aids in defining a precise spatial location for a predicted object. Individual predicted object confined within a bounding box is described by five attributes: - x , y , w , h , and confidence (see section 4.3.2). The x and y values depict the centre coordinates of the box regarding the position of the cell's box. In a case where the centre falls outside a cell, that cell will not represent it since it is not responsible for that centre. Each cell is associated with only objects whose centres lie within it. The coordinates are normalised to $[0,1]$. The capacity of the box, indicated by the width (w) and height (h), is also normalised to the range $[0, 1]$ relative

to the overall image capacity. The presence or absence of an object is determined by the bounding box confidence score (denoted as C_i^j). The bounding box confidence score is represented in Equation X. In a case whereby the C_i^j values of bounding boxes exceed a certain threshold, the bounding boxes are displayed, if not, they are discarded.

$$C_i^j = P_{i,j} * IoU_{pred}^{truth} \quad (10)$$

Here, C_i^j represents the confidence score in the i th grid of the j th bounding box. $P_{i,j}$ represents a function that depends on the presence of an object. If there is an object on the i th grid of the j th of the box, then the $P_{i,j}=1$; otherwise $P_{i,j}=0$. The term $IoU_{truthpred}$ denotes the intersection over union between the ground truth box and the predicted bounding box.

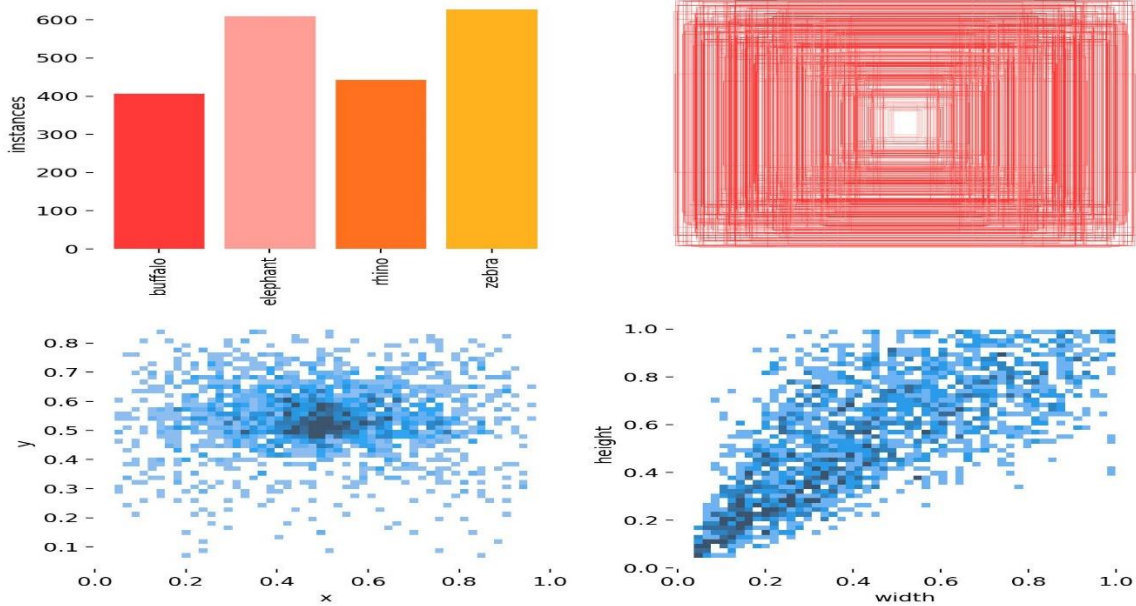


Figure 41: Object distribution within the dataset

The subfigures in Figure 41 are presented in a left-to-right, top-to-down order. Subfigure 41.A illustrates the object distribution in the dataset, revealing that zebras and elephants are the predominant objects. The subfigure 41.B shows the sizes of the object bounding boxes, with the centre coordinates of all object boxes fixed at one point. This signifies that the dataset contains significant number of small-sized objects. The third subfigure illustrates the dispersion of centre point coordinates for the object bounding boxes, demonstrating an even concentration of object centres across the image area. The fourth subfigure is a scatter plot depicting the width and height of the object bounding boxes, with the darkest regions in the bottom-left corner, emphasising the predominance of small object in the dataset.

5.2.2 YOLOv8 Model training output

The progress and outcomes of each phase in every epoch are presented on the screen to observe how the model learns and improves its performance from one epoch to another. As illustrated in Figure 42, the training progress of the YOLOv8 is displayed.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
11/100	0G	1.038	1.218	1.427	18	640: 100%	75/75 [14:29<00:00, 11.60s/it]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	297	530	0.776	0.683	0.761	0.515
8.03s/it]							
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
12/100	0G	0.9998	1.14	1.381	17	640: 100%	75/75 [14:17<00:00, 11.43s/it]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	297	530	0.78	0.701	0.809	0.569
7.87s/it]							
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
13/100	0G	0.9833	1.095	1.379	32	640: 100%	75/75 [14:28<00:00, 11.58s/it]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	297	530	0.837	0.769	0.864	0.616
7.90s/it]							

Figure 42: YOLOv8 training progress

For each epoch, a summary is provided for the training and validation stages. The outcomes of the training phase are shown in line 1 and 2, whereas the outcomes of validation are shown in line 3 and 4. Within the training phase, a significant calculation that involves evaluating the error in a loss function takes place to minimise the error rate. The most critical metrics in this calculation are box loss (box_loss) [106], classification loss (cls_loss) [107], and distribution focal loss (dfl_loss) [106]. Box_loss demonstrate the degree of error in detected bounding boxes, cls_loss quantifies the error regarding the classification of the detected bounding boxes, and dfl_loss enables the network to quickly concentrate on learning the probabilities of values in the locations of the target bounding box.

It is imperative to take note that the model might accurately detect the bounding box coordinates around the object but make errors in the identification of the object within that bounding box. For instance, in this study, the model may correctly determine the dimensions but inaccurately classify it, such as identifying buffalo as rhino but correctly assessing the size of the object. In terms of the model genuinely acquiring knowledge from the data, the reduction in these values were observed from one epoch to the next. As shown in the previous screenshot, the box_loss exhibited a decrease, with values of 1.038, 0.9998, and 0.9333. While the cls_loss experienced a similar decrease from 1.218 to 1.14, and down to 1.095. The dfl_loss shows a decrease, with values of 1.427, 1.381, and 1.379. Figure 43 shows the results of the training through epoch visualisation.

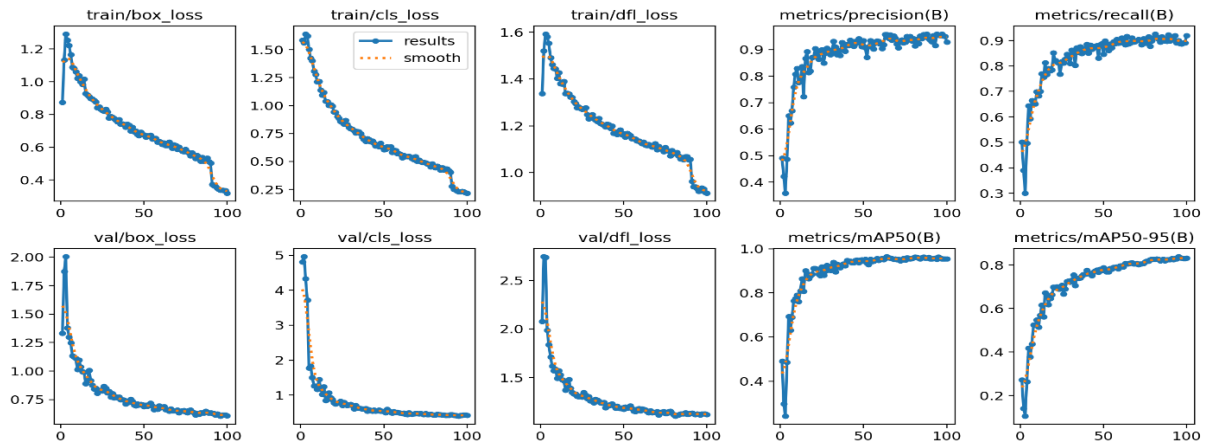


Figure 43: Epoch visualisation during training

5.2.3 YOLOv8 Model Validation

The YOLOv8 model was trained for approximately 24.066 hours over 100 epochs. At the 100th epoch, the model achieved the following performance results: a precision of 0.929, a recall of 0.92, a mAP (at an IoU threshold of 0.5) of 95.5%, and a mAP (at a spectrum of IoU thresholds of 0.5 to 0.95) of 83.3%. Table 15 depicts the evaluation of the model's detection performance on the validation set. The table includes the performance metrics such as Precision, Recall, mAP50, and mAP50-95 for the model. Furthermore, it provides the performance metrics value for individual animal classes as well.

Table 15: Evaluation of the model's performance on the validation set.

		Precision	Recall	mAP50 (%)	mAP50-95 (%)
	Validation	0.95	0.90	96	85
	Buffalo	0.96	0.865	86.5	95
	Elephant	0.957	0.879	95.4	80.4
	Rhino	0.963	0.955	97.7	88.1

YOLOv8	Zebra	0.935	0.904	96.5	82.3
---------------	-------	-------	-------	------	------

The model exhibits performance due to hyperparameter permutations and dataset configurations undertaken, which were discussed in detail on section 4.7.3. The results clearly indicate that the YOLOv8 model performs well in terms of precision, recall, and mAP. MAP50 assesses the model's ability to detect objects with a moderate overlap, using an IoU threshold of 0.5. Higher mAP50 scores indicate better performance. To offer a more thorough evaluation, mAP50-95 extends this evaluation by considering a spectrum of IoU thresholds, from 0.5 to 0.95.

It is imperative to note that the extent of performance varies among the different classes, with some classes achieving precision, recall, and mAP scores exceeding 90%, while others fall slightly below. Figure 44 shows the Precision-Confidence curve, Recall-Confidence curve, and Figure 45 shows F1-Confidence curve, and the confusion matrix of the model on the validation dataset.

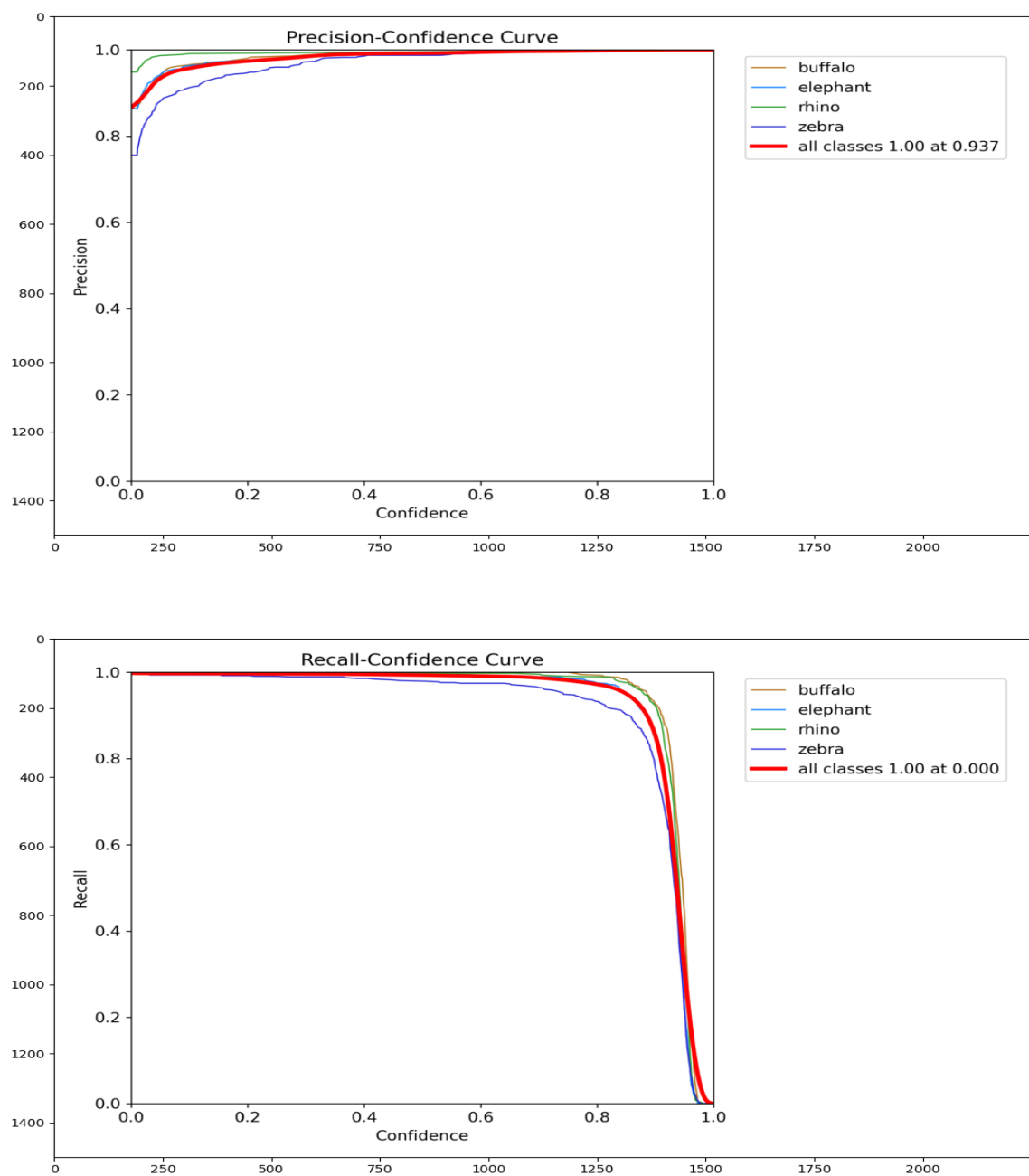


Figure 44: Precision-Confidence and Recall-Confidence curves on validation set.

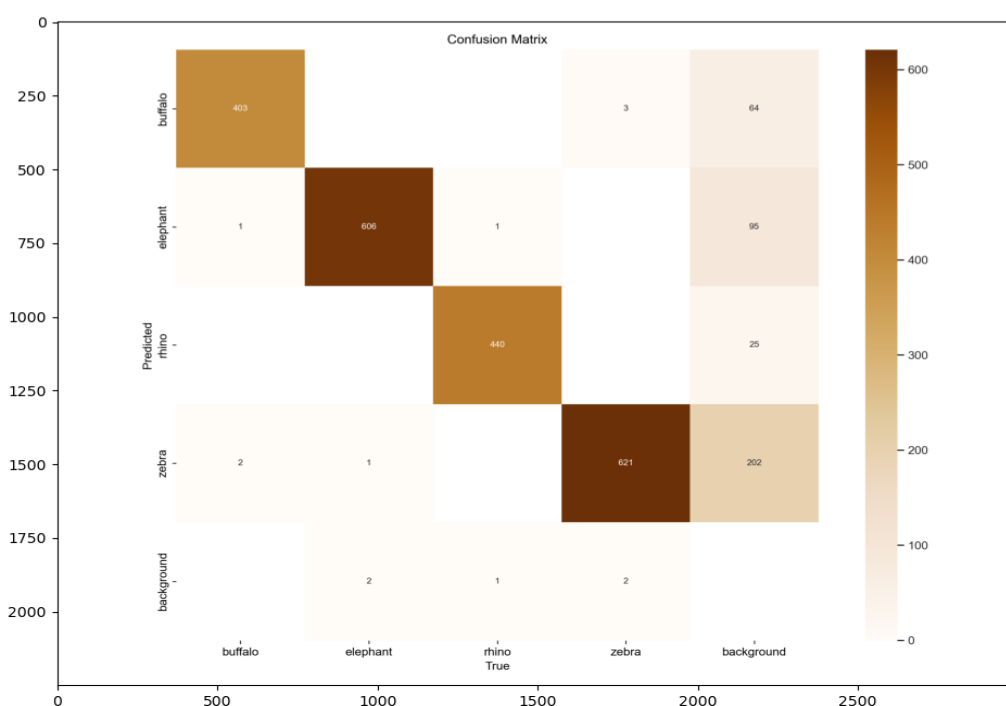
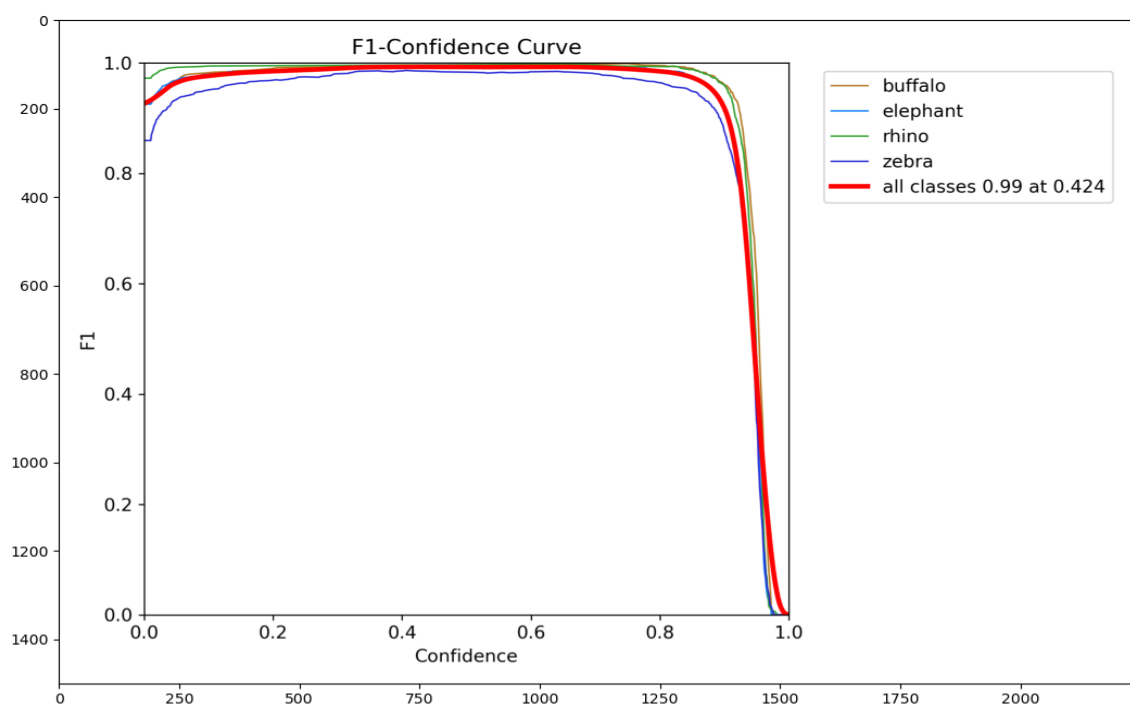


Figure 45: F1-Confidence curve and the Confusion Matrix on the validation set.

5.2.4 YOLOv8 Model Test

The object detection model underwent a final evaluation on the unseen test set to assess its performance. The model achieved a precision of 0.815, a recall of 0.869, an mAP50 of 89%, as well as an mAP50-95 of 72.2%. Table 16 shows the comprehensive model performance on the test set.

Table 16: Overall model performance on the test set

Model	Precision	Recall	mAP50 (%)	mAP50-95 (%)
YOLOv8	0.815	0.869	89	72.7

The model demonstrated strong performance on this challenging test set. This is attributed to the structure of the test set, which allowed single and multiple object test to be conducted. However, due to variations in class instances, some classes performed slightly poorly in the test set compared to the validation set. This decline in performance was due to the differences between the test set images and those in the training and validation sets. Detailed model performance for each class is outlined in Table 17.

Table 17: Detailed model performance for each class

YOLOv8		Precision	Recall	mAP50 (%)	mAP50-95 (%)
	Testing	0.815	0.869	89	72.2
	Buffalo	0.933	0.629	79	68.1
	Elephant	0.675	1	93.4	84.8
	Rhino	1	0.973	99.5	80.8
	Zebra	0.651	0.875	83.9	55.1

Table 17 gives a detailed summary of the model's performance on the test dataset. The generated results indicated a good model performance across the four classes, with precision ranging from 0.65 to 1 and recall from 0.63 to 1. The rhino class performed well, reaching the mAP50 of 99.5% and mAP50-90 of 80.8% despite the variations of class instances. The class with the least instances performed better, which in this case was the rhino class. Figure 46 shows the Precision-Confidence curve, Recall-Confidence curve and Figure 47 shows F1-confidence curve, and the confusion matrix of the model on the test dataset.

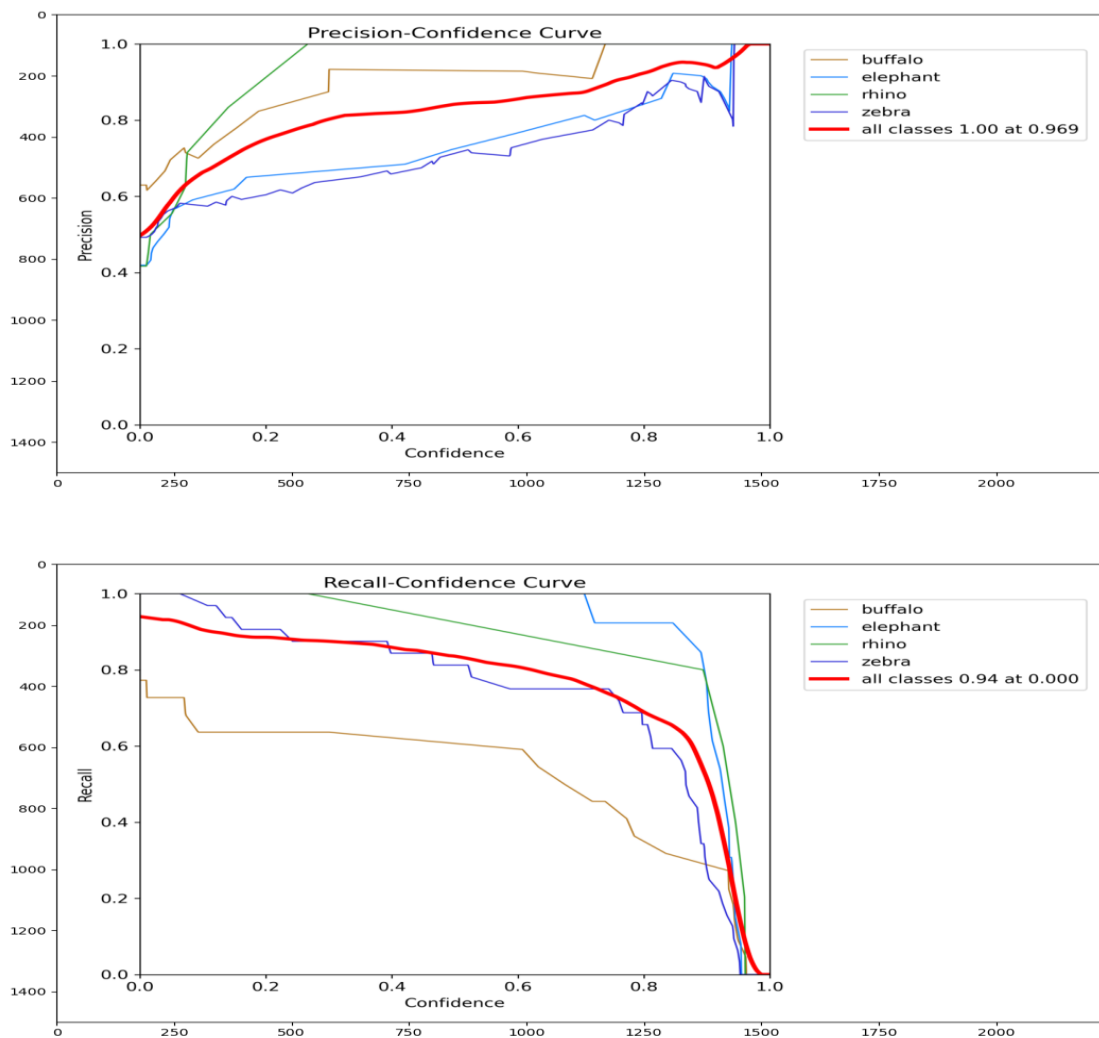


Figure 46: Precision-Confidence and Recall-Confidence curves on the test set

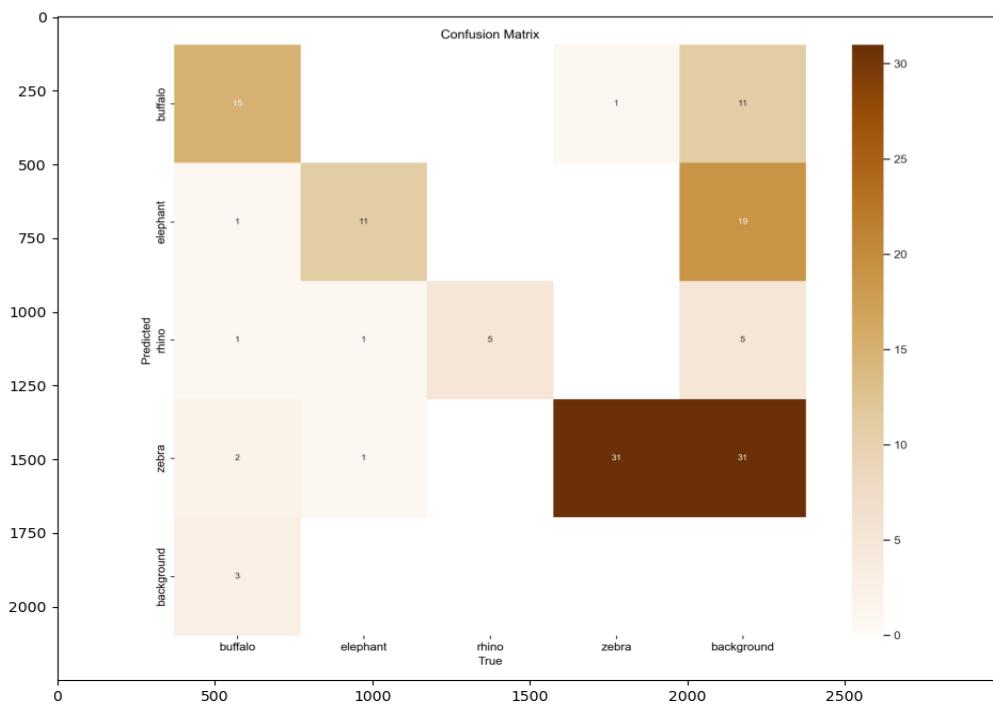
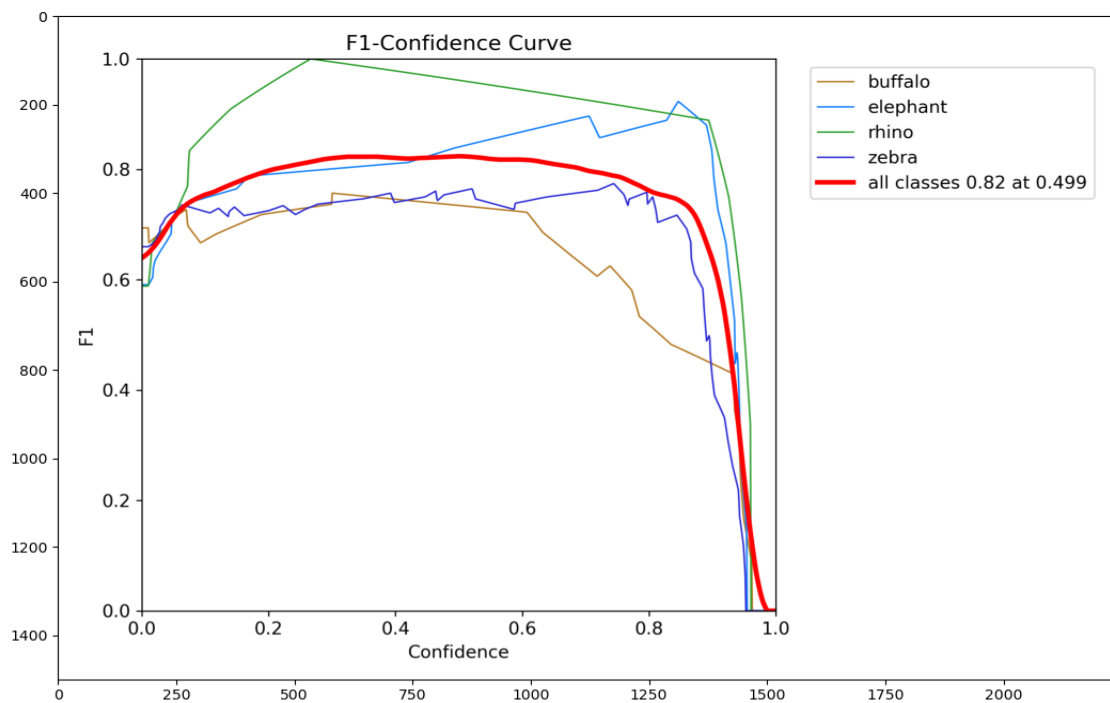


Figure 47: F1-Confidence curve and Confusion Matrix on the test set

The confusion matrix output reveals a few misclassifications of the predicted class to the actual class, with the rhino class being misclassified as background which was the least compared to other classes. The zebra class had the most misclassifications compared to the other classes. This was a result of having more instances within the test set, which had a higher chance to be misclassified. However, this did not reflect on the overall model accuracy when detecting objects.

5.3 YOLOv8 Sample predictions

To assess the predictive capabilities, the model was tested across all four classes of animals, including images containing multiple animal species of different classes. This evaluation aimed to determine whether the model exhibits robust detection abilities, even in scenarios where the image contains multiple animals of different species and features. Moreover, the prediction results provided a count for each species detecting in the image.

Buffalo

Figure 48 shows that the model was able to clearly detect the buffalo class.

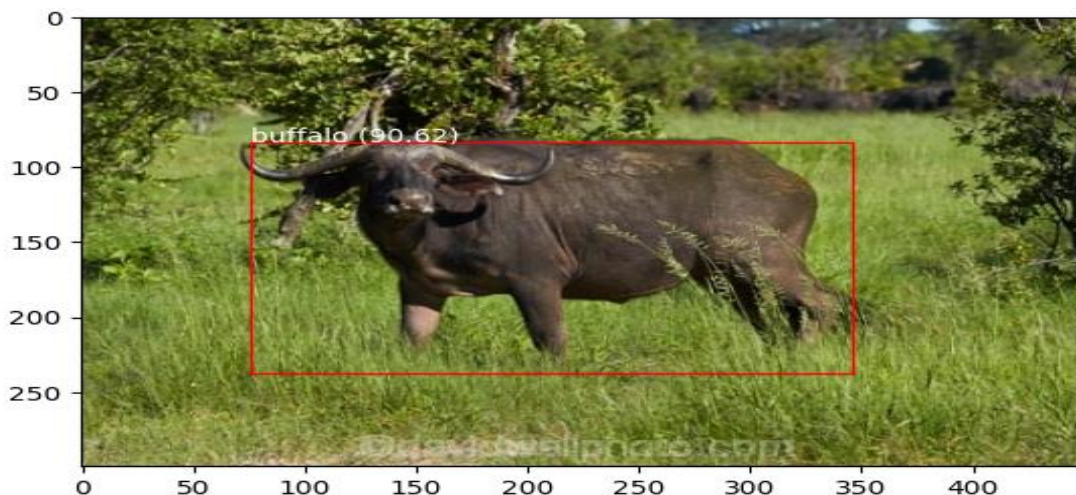


Figure 48: YOLOv8 detection on buffalo class

Instances of buffalo in the testing set were 22, making it the second lowest in the four animal classes. In some instances, buffalo objects were blurry and much smaller in size, especially when animals of different classes appear in the image. This resulted to a few inaccurate detections, which led to detecting some buffalo objects such as elephant or cow but correctly predicting the bounding boxes.

Granted that this model was a pretrained model, the cow class reflected the millions of images that were used to train the model, making it the more likely animal species that resembles features that appear in a buffalo. The output of the model count reflected the inaccurate detection of buffalo (Figure 49), having a total count of 1 zebra and 1 cow. This confirms that the model might need more training in the buffalo class, so that the model is equipped to detect the difference even on closely related species such as buffalo and cow. Figure 49 shows an incorrect detection of buffalo.

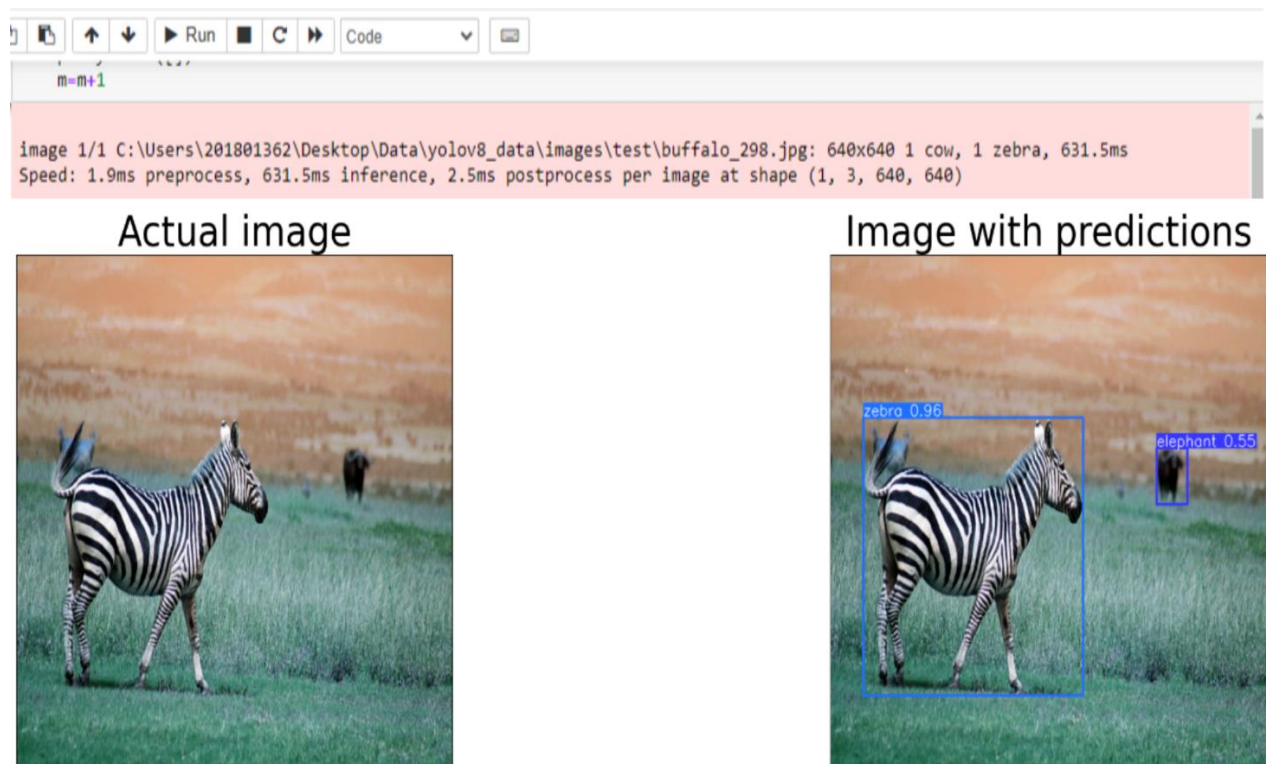


Figure 49: Detection results of buffalo when appearing small and blurry.

Elephant

The model was tested on the elephant class which consisted of 13 unseen instances in the test set. The model generated an accurate prediction on both single and multiple object detections on the test set for the elephant, which was an indicator that the model underwent adequate training in the elephant class. The animal count was accurate as well, given that the model correctly detected 2 elephant species that were in the image. Figure 50 shows the detection output of the elephant and animal count.

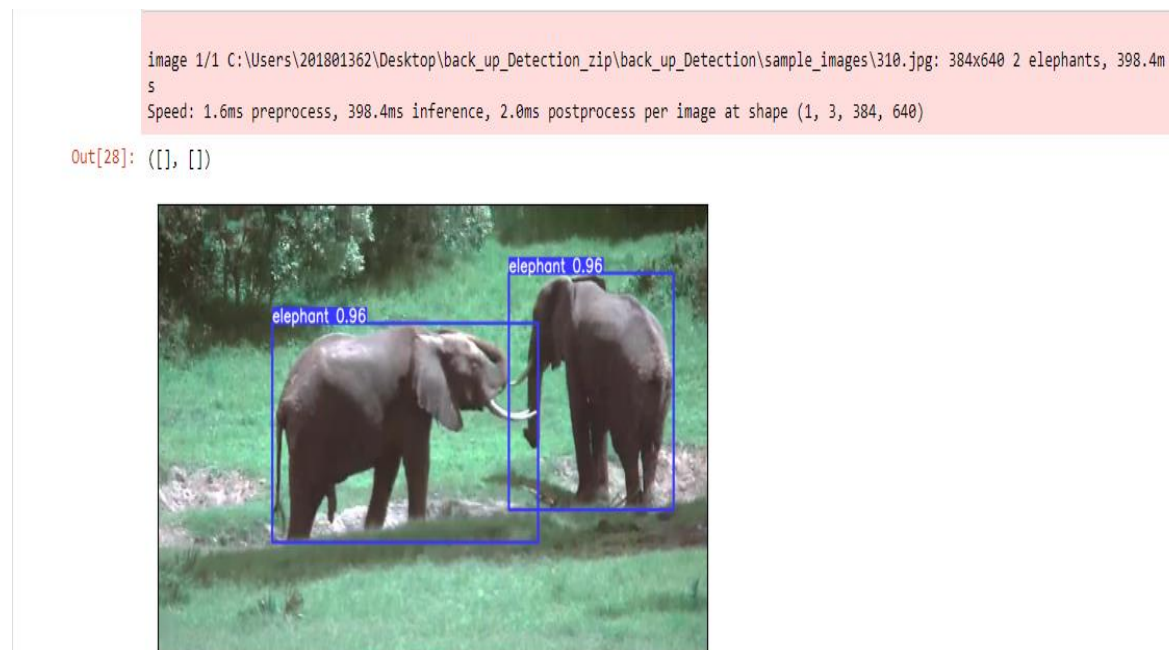


Figure 50: YOLOv8 detections on elephant class.

Figure 51 shows the results of elephant detection in an image that consists of different animal classes with the animal count for each image respectively.

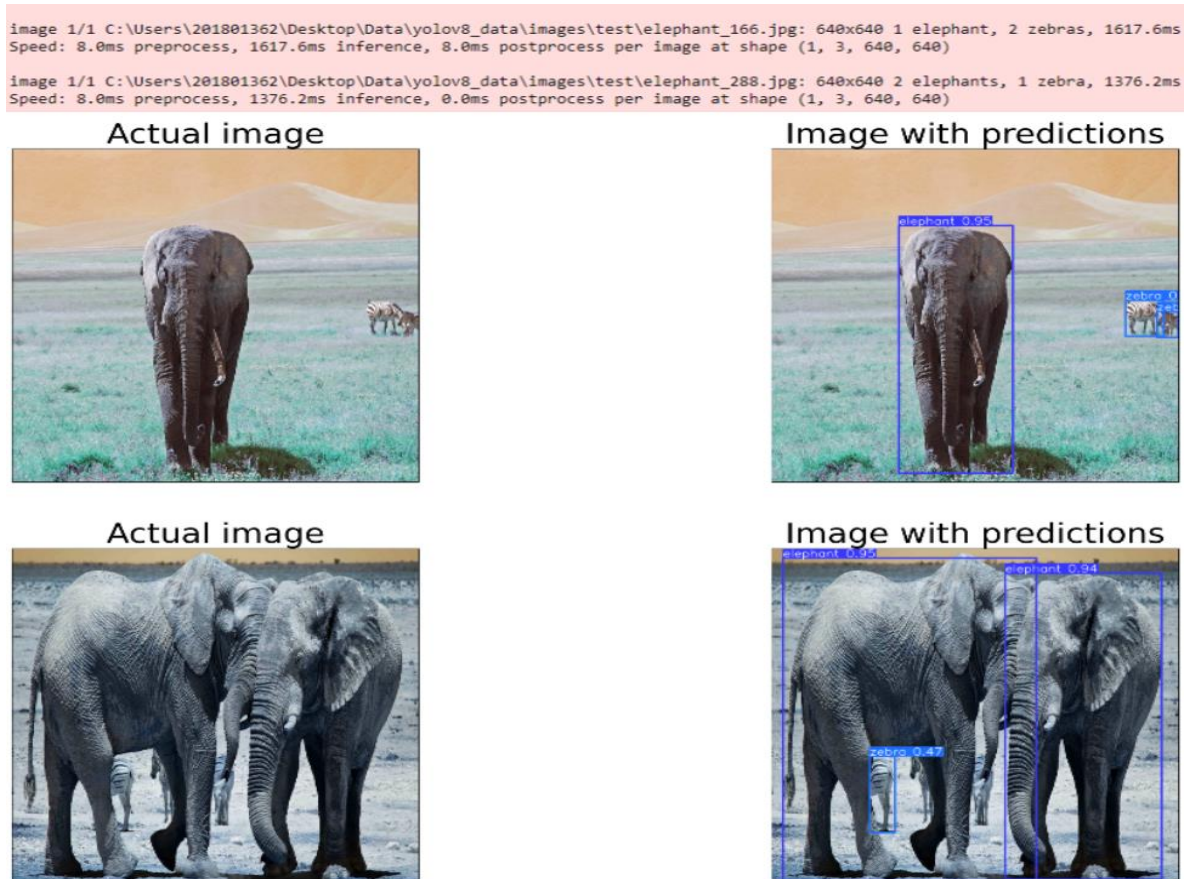


Figure 51: Detection results of elephants appearing with other animal classes and animal count.

Rhino

Predictions on the rhino class, which had 5 instances on the test set, were carried out. The results of rhino prediction were like the buffalo class whereby the model detected well on single species identification but had a few inaccurate detections when it came to a scenario where more than one animal class was visible in the image. Figure 52 shows the object detection predictions on the rhino class.

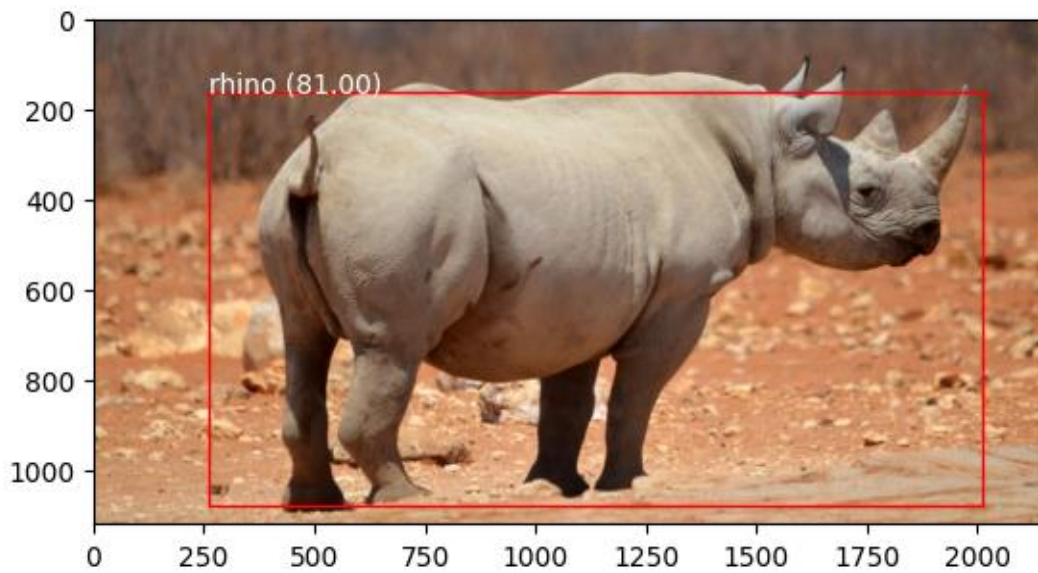


Figure 52: YOLOv8 detection on rhino class

The incorrect detection of rhino species was observed when we tested the model on some images. Figure 53 shows the rhino class prediction on images that consisted of distinct animal classes, resulting in an incorrect detection of rhino as elephant. This prediction generated a count of the incorrectly detected object of which resultantly was for the elephant class.

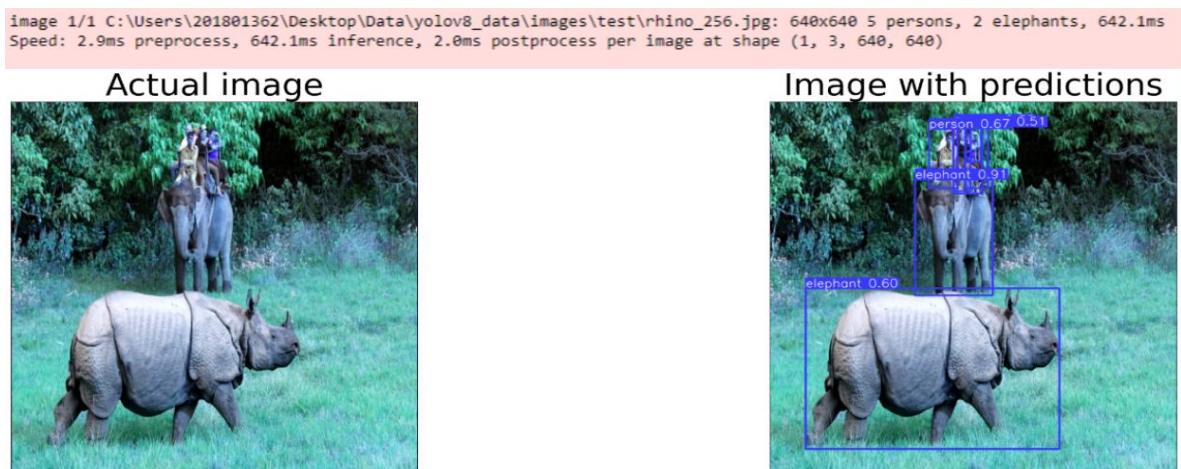


Figure 53: Detections of rhino appearing in images that consists different animal classes.

Zebra

The model was evaluated on the elephant class, which included a total of 32 instances from the test set. The model produced accurate predictions for the zebra class on both single and multiple species images. This successful performance suggested that the model had effective and substantial training to be able to perform recognition on the zebra class. Figure 54 shows the model's prediction results on the zebra class for single species detection provided with the total count of animals in an image, which was 20 zebras in this case.

```
image 1/1 C:\Users\201801362\Desktop\back_up_Detection_zip\back_up_Detection\sample_images\121.jpg: 480x640 20 zebras, 454.7ms  
Speed: 1.1ms preprocess, 454.7ms inference, 15.6ms postprocess per image at shape (1, 3, 480, 640)  
Out[65]: ([], [])
```

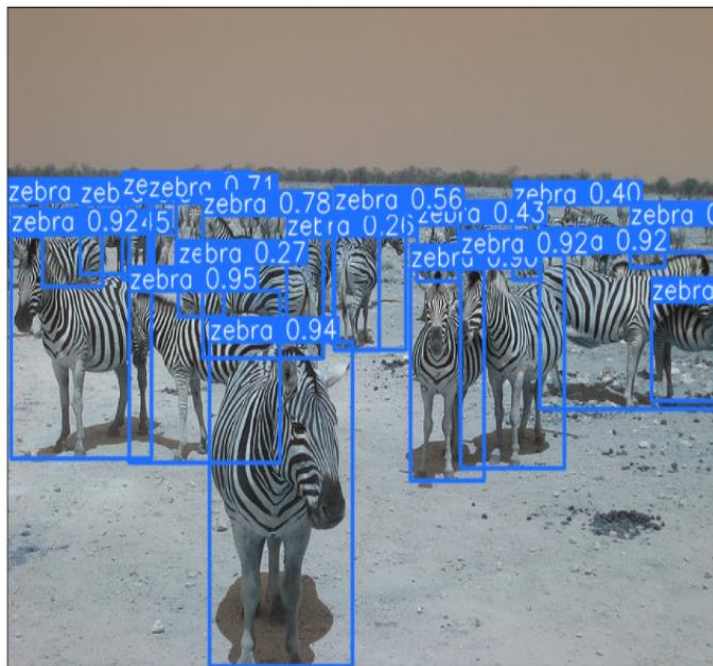


Figure 54: YOLOv8 detection on the zebra class

In Figure 55, the prediction output of the model was presented, particularly on the zebra class in the scenario where there are distinct animal classes appearing in the image.

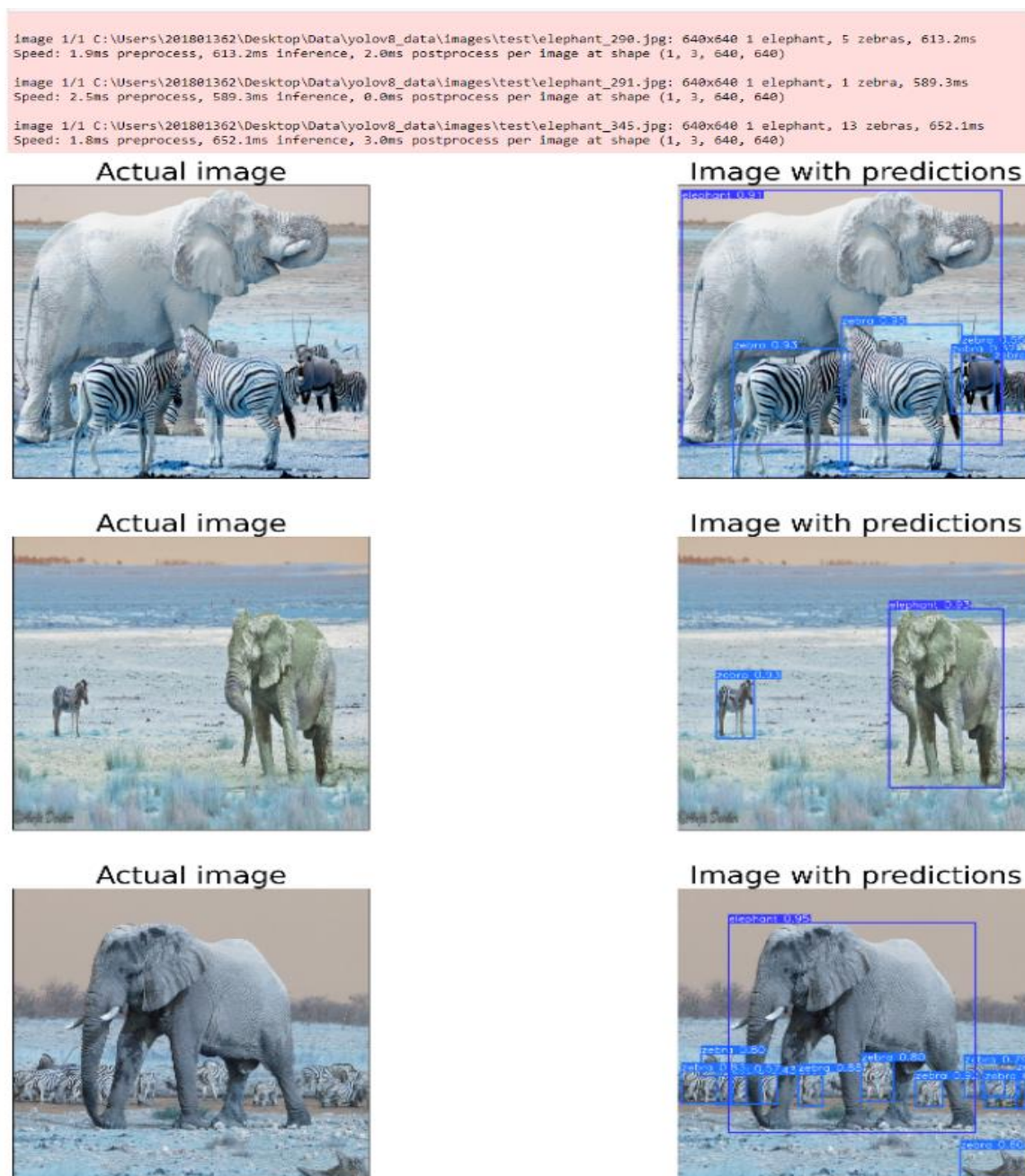


Figure 55: Detection of zebras in different images that show multiple animal species.

Figure 55 illustrated the species detection different classes, with an object to observe how the model performs on the zebra class. The count was displayed for every image that had predictions. The first image showed the count of 1 elephant and 5 zebras, 1 elephant and 1 zebra for the second image, and 1 elephant and 13 zebras for the last image. The output of the model on the zebra class was successful and it was consistent across other class predictions that were consisting of the zebra class. This was, not only observed on the identification, but also the animal count result, which was a clear indicator of the YOLOv8 capabilities on species recognition tasks.

5.4 Conclusion

The use of pre-trained models showed great potential for accurate image identification tasks, especially when performing experiments with insufficient image data. The results of these models are also a clear indication of the effect of data augmentation and different hyperparameter adjustments and fine-tuning on different models, which is a factor worth noting when determining the best model for image classification. The incorporation of distinct augmentation methods and hyperparameter permutation played a role in the performance of the models. Augmentation applied on Alexnet and ResNet architectures showed more image variation to the models which improved generalisation ability of images that allowed to the constructed of a robust model compared to the augmentation technique that was applied for the VGG16 even though it performed better than the Alexnet model. In the context of optimisers employed in this study, distinct optimisers for Pytorch implemented models and Keras implemented model, which is the VGG16, were employed. This showed the prowess of the ADAM optimiser which employs a combination of RMSprop and Stochastic Gradient Descent (SGD) optimisers.

The YOLOv8 displayed good detection and counting capabilities of animal classes used in this study. The prediction performance yielded good results, particularly on a dataset that had imbalanced animal instances, with some species appearing more than others in the images. However, using hyperparameter permutations and

substantial amount of training, the model was able to accurately detect and count animal species. Although there were a few incorrect detections of some images due to the size of the dataset, the model was still able to correctly count the detected objects in both single and multi-class species detection.

Chapter 6: Conclusion

The foundation of conservation efforts is the collection of data on resource abundance and population capacity, particularly in relation to population growth and reduction. The laborious human methods of processing vast amounts of imagery data with an object to gather counts and identify the various species categories to determine population sizes. These methods are time consuming given the volumes of images that require processing, and mistakes tend to happen where some images are misclassified as incorrect species, given that these tasks are performed by humans. This has left a room for improvement in the accuracy of species identification methods to make a positive impact in other domains such as ecological monitoring.

This dissertation addresses the challenge of manual counting animals in camera trap images by developing two distinct automated identification solutions, a deep learning image classification model and an object detection model. The dataset employed in this work had several minor complexities, given that the collected imagery data provided unsustantial number of images. Additionally, the imagery data consisted of few low-quality images, and images where some animals appeared to be small in blurry conditions. All these issues affected the clearness of the image. Moreover, some images appeared to consist of extra animal species within them. Thus, the dataset presented a challenge that required several extensive techniques like deep learning, which provided an iterative process, an advanced method of parameter permutation to generate desired prediction outcomes. Nonetheless, the difficulty of this method was the intricate computational resources required to withstand hundreds of distinct image features with hours of training.

6.1 Conclusion

The goal of the image classification models was to analyse image data by establishing a set of target classes, and then use unseen images to train the models

for identification. The result from the models were accessible for acquisition and interpretation. For a small experimental testing sample of four classes of mammal species, the classification model produced good prediction results, with 99% accuracy by the best performing model for the predicted label corresponding to the truth label.

This dissertation was closely linked to a study [108], which used RetinaNet to identify animals in images. In that study, they used 561 images from three species classes, and their proposed model generated an accuracy of 77% on the test and 86% on their best performing animal class of elephant. They emphasized the poor image quality that led to a low generalisation by the model. In this study, low quality was not predominant, and the number of images that were used led to a higher mAP in the test set. The proposed object detection model detected and counted four animal classes of mammal species. The model was experimented on 1,504 images, and yielded a precision of 0.815, a recall of 0.869, an mAP50 of 89%, and a mAP50-95 of 72.2% on the test set. Comprehensively, the elephant and zebra class performed better than the detection of buffalo and rhino on both single and multiple species detection tasks. Due to the model's pre-trained capabilities on cow or cattle images, it is challenging to identify some buffalo as cows, especially when it appears too small to see in a blurry image. The same for some rhino prediction given the resemblance to the dominating sized elephants in terms of skin colour, patterns, and texture.

Primarily, these findings demonstrate that biologists and ecologists may be able to conserve significant amount of time by using standard CV techniques and deep learning in both data-scarce and data-enriched scenarios. Instead of manually searching through vast amounts of images to identify, count, and classify animal species, deep learning methods provide extensive capabilities to optimally perform data collection tasks. Replacing tedious manual tasks by crucial scientific endeavours. Ultimately, the automated counting techniques described in this

dissertation had the potential to significantly enhance both wildlife conservation and future research.

6.2 Answers to the research questions

In chapter one, three research questions were asked in line with the three objectives (see section 1.3).

Research Question one: What are the existing image classification models used in biodiversity monitoring through deep learning, and what are the current research efforts and gaps in this field?

In conducting this research, a thorough understanding of the methods and approaches employed for species identification in image data from previous studies was crucial. Previous investigations into animal detection and classification utilised various techniques, including machine learning, computer vision (see section 2.4), and deep learning (see section 2.5). These approaches were employed to analyse images, enabling the learning of species-specific features for accurate identification, classification, and counting of animal species [12]. Comprehension of these methods addressed the initial research question.

Research Question two: How can an automated image classification model be developed for biodiversity monitoring using deep learning methods?

The research question was addressed by performing an investigation into deep learning and the toolkit it offers to analyse image data, perform predictions, and count animals. In developing a deep learning classification model, Convolution Neural Networks (CNNs) were used for image identification tasks. This included a combination of other Python libraries like Torchvision Transformations and Keras ImageDataGenerator to facilitate the preprocessing of image data and the development of a classification model to identify species. In this study, CNNs that were used included Alexnet, ResNet 18, 34, 50, and VGG16 for classification tasks.

YOLOv8 was used for the object detection task. CNNs were used to learn knowledge and provide relevant predictions.

Research Question three: How can the performance of the deep learning models be evaluated?

The third question was answered by using evaluation metrics to measure the performance of the deep learning models for both classification and object detection tasks. The performance of these architectures was evaluated using the accuracy, precision, recall, and F1-Score for classification and precision, recall, intersection over union (IoU), and mean average precision (mAP) object detection. The accuracy of the ResNet-50 on the test set was 99%. The YOLOv8 yielded a precision of 0.815, a recall of 0.869, an mAP50 of 89%, and a mAP50-95 of 72.2% on the test set. In conclusion, the results were satisfactory as they answered the third research question.

6.3 Future works

The results emanating from the deep learning model, as discussed in section 5 opened significant diversifications in accuracy, precision, recall, mAP50, mAP50-95, and F1-scores across different class, because of the hereditary differences in the detection complexities. After that, future research efforts should consider the development of different models tailored for each animal class to optimise the detection of performance for specific classes. Furthermore, it is imperative to take note of the choice of parameter selection that was implemented in this study, based on the restricted investigation within the realm of parameters. This restriction was mainly driven by the limitations of the computational resources. An expansion of these resources would allow a more extensive investigation for optimal parameter configurations. For example, this enabled investigations on a broader spectrum of values for key parameters such as batch size and learning, as well as other model

accuracy determinants such as dropout rates and the number of epochs. It is worth noting that providing a wider range of hyperparameter permutations is not only applicable to deep learning models, but also to the data.

Image data used in this study consisted of 1,504 images for both classification and object detection tasks, which was insignificant based on some CNN models in several classes. Moreover, increasing the number of data includes employing more than four animal classes, as well as the incorporation of video data as part of the dataset. In this manner, the model was exposed to a wide variety of data and was able to learn more animal features due to the different perspective it provides. Making it a clear indicator that significantly expanding the training and validation data stands a high chance to yield improved model performance.

Dataset repository such as the Labeled Information Library of Alexandria: Biology and Conservation (LILA) are standard platforms that offer distinct image datasets that encompass different objects and backgrounds. Moreover, these datasets include images found within nature reserves across South Africa, which provide real-world research scenarios pertaining to monitoring that address conservation challenges tailored to a specific nature reserve which has its unique terrain, biomes, and resources. This make it suited for research methods to be conducted on South African soil.

References

- [1] J. Vélez, W. McShea, H. Shamon, P. Castiblanco-Camacho, M. Tabak, C. Chalmers, P. Fergus and J. Fieberg, "An evaluation of platforms for processing camera-trap data using artificial intelligence," *Methods in Ecology and Evolution*, vol. 14, 2022.
- [2] M. Kutugata, J. Baumgardt, J. Goolsby and A. Racelis, "Automatic Camera-Trap Classification Using Wildlife-Specific Deep Learning in Nilgai Management," *Journal of Fish and Wildlife Management*, vol. 12, 2021.
- [3] B. C. Augustine, J. A. Royle, M. J. Kelly, C. B. Satter, R. S. Alonso, E. E. Boydston and K. R. Crooks, "Spatial Capture-Recapture with Partial Identity: An Application to Camera Traps," *The Annals of Applied Statistics*, vol. 12, no. 1, pp. 67-95, 2018.
- [4] B. van Wilgen, J. Measey, D. Richardson, J. Wilson and T. Zengeya, "Biological Invasions in South Africa: An Overview," in *Biological Invasions in South Africa*, Springer International Publishing, 2020, pp. 3-31.
- [5] L. E. Pardo Vargas, S. Bombaci, S. Huebner, M. Somers, H. Fritz, C. Downs, A. Guthmann, R. Hetem, M. Keith, A. le Roux, N. Mgqatsa, C. Packer, M. Palmer, D. Parker, M. Peel, R. Slotow and S. "Snapshot Safari: A large-scale collaborative to monitor Africa's remarkable biodiversity," *South African Journal of Science*, vol. 117, pp. 1-4, 2021.
- [6] D. Tilman, M. Clark, D. Williams, K. Kimmel, S. Polasky and C. Packer, "Future threats to biodiversity and pathways to their prevention," *Nature*, vol. 546, pp. 73-81, 2017.
- [7] S. Christin, E. Hervet and N. Lecomte, "Applications for deep learning in ecology," *Methods in Ecology and Evolution*, vol. 10, pp. 1632-1644, 2019.
- [8] M. Tan, W. Chao, J.-K. Cheng, M. Zhou, Y. Ma, X. Jiang, J. Ge, L. Yu and L. Feng, "Animal Detection and Classification from Camera Trap Images Using Different Mainstream Object Detection Architectures," *Animals*, vol. 12, 2022.
- [9] M. Adam, P. Tomášek, J. Lehejček, J. Trojan and T. Jůnek, "The Role of Citizen Science and Deep Learning in Camera Trapping," *Sustainability*, vol. 13, 2021.
- [10] Z. Miao, K. Gaynor, Z. Liu, O. Muellerklein, M. S. Norouzzadeh, A. McInturff, R. Bowie, R. Nathan, S. Yu and W. Getz, "Insights and approaches using deep learning to classify wildlife," *Scientific Reports*, vol. 9, 2019.
- [11] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, S. J. Sweeney, K. C. Vercauteren, N. P. Snow, J. M. Halseth, P. A. Di Salvo, J. S. Lewis, M. D. White, B. Teton, J. C. Beasley, P. E. Schlichting and Bou, "Machine learning to classify animal species in camera trap images: Applications in ecology," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 585-590, 2019.

- [12] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, C. Packer and J. Clune, "Automatically identifying wild animals in camera trap images with deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, 2018.
- [13] Z. Xu and J. Zhang, *Overview of Computer Science*, 2021, pp. 1-39.
- [14] A. McQuatters-Gollop, I. Mitchell, C. Vina-Herbon, J. Bedford, P. F. E. Addison, C. P. Lynam, P. N. Geetha, E. A. Vermeulan, K. Smit, D. T. I. Bayley, E. Morris-Webb, H. J. Niner and S. Otto, "From Science to Evidence – How Biodiversity Indicators Can Be Used for Effective Marine Conservation Policy and Management," *Frontiers in Marine Science*, vol. 6, 2019.
- [15] R. Kerry, F. J. Montalbo, R. Das, S. Patra, G. Mahapatra, G. Maurya, V. Nayak, A. Jena, K. Ukhurebor, R. C. Jena, S. Gouda, S. Majhi and J. Rout, "An overview of remote monitoring methods in biodiversity conservation," *Environmental Science and Pollution Research*, vol. 29, no. 53, pp. 1-43, 2022.
- [16] P. Whitehorn, L. Navarro, M. Schröter, M. Fernández, X. Rotllan-Puig and A. Marques, "Mainstreaming biodiversity: A review of national strategies," *Biological Conservation*, vol. 235, pp. 157-163, 2019.
- [17] J. Wäldchen and P. Mäder, "Machine learning for image based species identification," *Methods in Ecology and Evolution*, vol. 9, no. 11, pp. 2216-2225, 2018.
- [18] M. J. Feio, R. M. Hughes, M. Callisto, S. J. Nichols, O. N. Odume, B. R. Quintella, M. Kuemmerlen, F. C. Aguiar, S. F. Almeida, P. Alonso-EguíaLis, F. O. Arimoro, F. J. Dyer and Harding, "The Biological Assessment and Rehabilitation of the World's Rivers: An Overview," *Water*, vol. 13, 2021.
- [19] C. B. Anderson, "Biodiversity monitoring, earth observations and the ecology of scale," *Ecology letters*, vol. 21, pp. 1572-1585, 2018.
- [20] T. Petso, R. Jamisola and D. Mpoeleng, "Review on methods used for wildlife species and individual identification," *European Journal of Wildlife Research*, vol. 68, no. 3, 2022.
- [21] N. A. Gilbert, B. S. Pease, C. M. Anhalt-Depies, J. D. Clare, J. L. Stenglein, P. A. Townsend, T. R. { Deelen} and B. Zuckerberg, "Integrating harvest and camera trap data in species distribution models," *Biological Conservation*, vol. 258, pp. 109-147, 2021.
- [22] P. Palencia, J. Fernández-López, J. Vicente and P. Acevedo, "Innovations in movement and behavioural ecology from camera traps: Day range as model parameter," *Methods in Ecology and Evolution*, vol. 12, pp. 1201-1212, 2021.
- [23] R. Joo, M. Boone, T. Clay, S. Patrick, S. Clusella-Trullas and M. Basille, "Navigating through the R packages for movement," *Journal of Animal Ecology*, vol. 89, no. 1, pp. 248-267, 2019.
- [24] N. Owen-Smith, G. Hopcraft, T. Morrison, S. Chamaillé-Jammes, R. Hetem, E. Bennitt and F. Van Langevelde, "Movement ecology of large herbivores in African

savannas: current knowledge and gaps," *Mammal Review*, vol. 50, pp. 252-266, 2020.

- [25] I. Nandutu, M. Atemkeng and P. Okouma, "Integrating AI ethics in wildlife conservation AI systems in South Africa: A review, challenges, and future research agenda," *AI & SOCIETY*, vol. 38, p. 245–257, 2021.
- [26] M. Matin, T. Shrestha, V. Chitale and S. Thomas, "Exploring the potential of deep learning for classifying camera trap data of wildlife: A case study from Nepal," in *AGU Fall Meeting Abstracts*, International Centre for Integrated Mountain Development (ICIMOD), 2021.
- [27] M. H. Athmaja and V. Kavitha, "A survey of machine learning algorithms for big data analytics," in *International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, 2017.
- [28] S. Najam, J. Ahmed, S. Masood and C. M. Ahmed, "Run-Time Resource Management Controller for Power Efficiency of GP-GPU Architecture," *IEEE Access*, vol. 7, pp. 25493-25505, 2019.
- [29] P. Domingos, "A Few Useful Things to Know about Machine Learning," *Commun. ACM*, vol. 55, no. 10, p. 78–87, 2012.
- [30] C. Janiesch, P. Zschech and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685-695, 2021.
- [31] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021.
- [32] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience*, vol. 2018, p. 13, 2018.
- [33] B. G. Weinstein, "A computer vision for animal ecology," *Journal of Animal Ecology*, vol. 87, no. 3, pp. 533-545, 2018.
- [34] A. I. Khan and S. Al-Habsi, "Machine Learning in Computer Vision," *Procedia Computer Science*, vol. 167, pp. 1444-1451, 2020.
- [35] S. Xu, J. Wang, W. Shou, T. Ngo, A.-M. Sadick and X. Wang, "Computer Vision Techniques in Construction: A Critical Review," *Archives of Computational Methods in Engineering*, vol. 28, no. 5, pp. 3383-3397, 2021.
- [36] A. V. Gomez, S. Augusto and V. Francisco, "Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks," *Ecological Informatics*, vol. 41, pp. 24-32, 2017.
- [37] D. Bambil, H. Pistori, F. Bao, V. Weber, F. M. Alves, E. G. Gonçalves, L. F. de Alencar Figueiredo, U. G. Abreu, R. Arruda and I. M. Bortolotto, "Plant species identification using color learning resources, shape, texture, through machine learning and artificial neural networks," *Environment Systems & Decisions*, vol. 40, no. 4, p. 480–484, 2020.

- [38] J. Xie, W. Kong, Y. Lu, P. W. Grant and S. Xu, "KSRFB-net: detecting and identifying butterflies in ecological images based on human visual mechanism," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 10, p. 3143–3158, 2022.
- [39] B. K. Rai, S. Sharma, G. Kumar and K. Kishor, "Recognition of Different Bird Category Using Image Processing," *International Journal of Online & Biomedical Engineering*, vol. 18, no. 7, p. 101–114, 2022.
- [40] A. Popkov, F. Konstantinov, V. Neimorovets and A. Solodovnikov, "Machine learning for expert-level image-based identification of very similar species in the hyperdiverse plant bug family Miridae (Hemiptera: Heteroptera)," *Systematic Entomology*, vol. 47, no. 3, p. 487–503, 2022.
- [41] B. Wang, "Automatic Animal Species Identification Based on Camera Trapping Data," 2014.
- [42] P. C. Ravoora and S. T.S.B., "Deep Learning Methods for Multi-Species Animal Re-identification and Tracking – a Survey," *Computer Science Review*, vol. 38, pp. 100–289, 2020.
- [43] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [44] H. Wang, R. Czerminski and A. C. Jamieson, "Neural Networks and Deep Learning," in *The Machine Age of Customer Insight*, Emerald Publishing Limited, 2021, pp. 91–101.
- [45] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Q. Al-Dujaili, Y. Duan, O. Al-Shamma, J. I. Santamar, M. A. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 2196–1115, 2021.
- [46] Y. Ma and Y. Xie, "Evolutionary neural networks for deep learning: a review," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 10, pp. 3001–3018, 2022.
- [47] K. Sharifani and M. Amini, "Machine Learning and Deep Learning: A Review of Methods and Applications," *World Information Technology and Engineering Journal*, vol. 10, no. 07, pp. 3897–3904, 2023.
- [48] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.
- [49] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.
- [50] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015.

- [51] K. He, X. hang, S. Ren and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision - ECCV 2016*, 2016.
- [52] S. Karen and Z. Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015.
- [53] G. Chen, T. X. Han, Z. He, R. Kays and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," in *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, 2014.
- [54] Y. Wu, Y. Xia, Y. Song, Y. Zhang and W. Cai, "Multiscale Network Followed Network Model for Retinal Vessel Segmentation," in *Medical Image Computing and Computer Assisted Intervention -- MICCAI 2018*, 2018.
- [55] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.
- [56] D. a. Y. H. Hunter, I. M. S. Pukish, J. Kolbusz and B. M. Wilamowski, "Selection of Proper Neural Network Sizes and Architectures—A Comparative Study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228-240, 2012.
- [57] H. Yousif, J. Yuan, R. Kays and Z. He, "Animal Scanner: Software for classifying humans, animals, and empty frames in camera trap images," *Ecology and Evolution*, vol. 9, no. 4, pp. 1578-1589, 2019.
- [58] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.
- [59] 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, pp. 3320-3328, 2014.
- [60] S. Patil, S. Patil, I. Chankeshwara, H. Rapatwar and P. Waykule, "Speech Emotion Recognition System Using Recurrent Neural Network in Deep Learning," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 3, pp. 2332-2336, 2022.
- [61] X. Yu, W. Jiangping, R. Kays, P. Jansen, T. Wang and T. Huang, "Automated identification of animal species in camera trap images," *EURASIP Journal on Image and Video Processing*, vol. 52, no. 1, pp. 1-10, 2013.
- [62] S. Schneider, G. Taylor and S. Kremer, "Deep Learning Object Detection Methods for Ecological Camera Trap Data," in *2018 15th Conference on Computer and Robot Vision (CRV)*, Toronto, 2018.
- [63] A. Vecvanags, K. Aktas, I. Pavlovs, E. Avots, J. Filipovs, A. Brauns, G. Done, D. Jakovels and G. Anbarjafari, "Ungulate Detection and Species Classification from Camera Trap Images Using RetinaNet and Faster R-CNN," *Entropy*, vol. 3, no. 353, 2022.

- [64] A. Martín, B. Paul, C. Jianmin, C. Zhifeng, D. Andy, D. Jeffrey, D. Matthieu, G. Sanjay, I. Geoffrey, I. Michael, K. Manjunath, L. Josh, M. Rajat and M. Sherry, "{TensorFlow}: A System for {Large-Scale} Machine Learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, 2016.
- [65] P. Adam, G. Sam, C. Soumith, C. Gregory, Y. Edward, D. Zach, L. Zeming, D. Alban, A. Luca and L. Adam, "Automatic differentiation in PyTorch," 2017.
- [66] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio, "Theano: A CPU and GPU Math Compiler in Python," in *SciPy*, 2010.
- [67] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, "A Survey on Deep Transfer Learning," in *Artificial Neural Networks and Machine Learning -- ICANN 2018*, vol. 11141, Springer, Cham, 2018, pp. 270-279.
- [68] L. Shaojuan, W. Lizhi, J. Li1 and Y. Yuan, "Image Classification Algorithm Based on Improved AlexNet," in *Journal of Physics: Conference Series*, 2021.
- [69] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *European Conference on Computer Vision*, 2014.
- [70] S. Hijazi, R. Kumar, C. Rowen and others, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, vol. 9, no. 1, 2015.
- [71] S. Sharma, S. Sharma and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 4, no. 12, pp. 310-316, 2017.
- [72] B. Md Inzamam Ul Haque, "A Facial Expression Recognition Application Development Using Deep Convolutional Neural Network For Children With Autism Spectrum Disorder To Help Identify Human Emotions," *Texas State University*, August 2019.
- [73] G. Larsson, M. Maire and G. Shakhnarovich, "FractalNet: Ultra-Deep Neural Networks without Residuals," *arXiv preprint arXiv:1605.07648*, 2016.
- [74] N. S. a. G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929-1958, 2014.
- [75] L. Prechelt, "Early Stopping - But When?," in *Neural Networks: Tricks of the Trade*, 1998, pp. 55-69.
- [76] L. Lu, X. Meng, Z. Mao and G. E. Karniadakis, "DeepXDE: A Deep Learning Library for Solving Differential Equations," *SIAM Review*, vol. 63, no. 1, pp. 208-228, 2021.
- [77] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, Lille, PMLR, 2015, pp. 448-456.

- [78] D. Justus, J. Brennan, S. Bonner and A. S. McGough, "Predicting the Computational Cost of Deep Learning Models," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3873-3882.
- [79] R. Zaheer and H. Shaziya, "A Study of the Optimization Algorithms in Deep Learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 536-539.
- [80] H. B. McMahan and M. J. Streeter, "Adaptive Bound Optimization for Online Convex Optimization," *ArXiv*, 2010.
- [81] X. Deng, L. Qi, D. Yong and M. Sankaran, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Information Sciences*, Vol. 340-341, pp. 250-261, 2016.
- [82] J. Liang, "Confusion Matrix: Machine Learning," *POGIL Activity Clearinghouse*, vol. 3, no. 4, 2022.
- [83] M. Sokolova, N. Japkowicz and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *AI 2006: Advances in Artificial Intelligence*, Berlin, 2006.
- [84] Ł. Rybak and J. Dudczyk, "A Geometrical Divide of Data Particle in Gravitational Classification of Moons and Circles Data Sets," *Entropy*, vol. 22, no. 10, 2020.
- [85] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto and E. A. B. da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," *Electronics*, vol. 10, no. 3, 2021.
- [86] S. Wenkel, K. Alhazmi, T. Liiv, S. Alrshoud and M. Simon, "Confidence Score: The Forgotten Dimension of Object Detection Performance Evaluation," *Sensors*, vol. 21, no. 13, 2021.
- [87] H. R. Seyed, T. Nathan, G. JunYoung, S. Amir, R. Ian D. and S. Silvio, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 658-666, 2019.
- [88] K. Oksuz, B. Cam, E. Akbas and S. Kalkan, "Localization Recall Precision (LRP): A New Performance Metric for Object Detection," in *15th European Conference*, Munich, 2018.
- [89] M. Ahmed, K. A. Hashmi, A. Pagani, M. Liwicki, D. Stricker and M. Z. Afzal, "Survey and Performance Analysis of Deep Learning Based Object Detection in Challenging Environments," *Sensors*, vol. 21, no. 15, 2021.
- [90] S. B. Lee, X. Gui, M. Manquen and E. R. Hamilton, "Use of Training, Validation, and Test Sets for Developing Automated Classifiers in Quantitative Ethnography," in *Advances in Quantitative Ethnography*, vol. 1112, Springer International Publishing, 2019, pp. 117-127.

- [91] F. Pérez-García, R. Sparks and S. Ourselin, "TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, vol. 208, pp. 106-236, 2021.
- [92] X. a. Z. Y. a. C. L. a. Z. L. {Han, "Pre-Trained AlexNet Architecture with Pyramid Pooling and Supervision for High Spatial Resolution Remote Sensing Image Scene Classification," *Remote Sensing*, vol. 9, no. 9, p. 848, 2017.
- [93] A. Ullah, H. Elahi, Z. Sun, A. Khatoon and I. Ahmad, "Comparative Analysis of AlexNet, ResNet18 and SqueezeNet with Diverse Modification and Arduous Implementation," *ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING*, vol. 47, no. 2, pp. 2397–2417, 2021.
- [94] Y. Kazemi and S. Houghten, "A deep learning pipeline to classify different stages of Alzheimer's disease from fMRI data," in *2018 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, St. Louis, MO, USA, 2018.
- [95] J. Zhu and J. Song, "An Intelligent Classification Model for Surface Defects on Cement Concrete Bridges," *Applied Sciences*, vol. 10, no. 3, pp. 2076-3417, 2020.
- [96] W. Wang, M. Wang, H. Li, H. Zhao, K. Wang, C. He, J. Wang, S. Zheng and J. Chen, "Pavement crack image acquisition methods and crack extraction algorithms: A review," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 6, pp. 535-556, 2019.
- [97] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *ArXiv*, vol. abs/1803.09820, 2018.
- [98] G. a. C. A. a. Q. J. Jocher, "YOLO by Ultralytics. 2023," 2023.
- [99] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, L. Diaconu, J. Poznanski, L. Yu, P. Rai, R. Ferriday and others, "ultralytics/yolov5: v3.0," *SAO/NASA Astrophysics Data System*, 2020.
- [100] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen and J.-W. Hsieh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571-1580, 2019.
- [101] L. Shu, Q. Lu, Q. Haifang, S. Jianping and J. Jiaya, "Path Aggregation Network for Instance Segmentation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8759-8768, 2018.
- [102] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 936-944.

- [103] K.-Y. Jeng, Y.-C. Liu, Z. Y. Liu, J.-W. Wang, Y.-L. Chang, H.-T. Su and W. H. Hsu, "A Coarse-To-Fine (C2F) Representation for End-To-End 6-DoF Grasp Detection," in *Conference on Robot Learning*, 2020.
- [104] C. W. a. A. Bochkovski and H. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, 2023, pp. 7464-7475.
- [105] J. T. Barron, "A General and Adaptive Robust Loss Function," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4326-4334, 2017.
- [106] X. L. Yang, W. Wenhai, W. Lijun, C. Shuo, H. Xiaolin, L. Jun, T. Jinhui and Jian, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection," *ArXiv*, vol. abs/2006.04388, p. 14, 2020.
- [107] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," *ArXiv*, vol. abs/1702.05659, 2017.
- [108] J. A. J. Eikelboom, J. Wind, E. van de Ven, L. M. Kenana, B. Schroder, H. J. de Knecht, F. van Langevelde and H. H. T. Prins, "Improving the precision and accuracy of animal population estimates with aerial image object detection," *Methods in Ecology and Evolution*, vol. 10, no. 11, pp. 1875-1887, 2019.

Appendix A - Codes / Data Collection Tools

The code for the image classification and object detection models can be found in the following Google documents repository:

<https://drive.google.com/drive/folders/1eS9KCIsurWTLAOZ-9g3iqTFnNn9vBN7?usp=sharing>

Appendix B - Ethics Certificate

Appendix C - Editing Certificate

Appendix D - Similarity Report