A HYBRID OF RESIDUAL NETWORK AND INCEPTION NEURAL NETWORK MODEL FOR WILDLIFE DETECTION AND IDENTIFICATION

MALACH OBISA AMONGA

A Research Thesis Submitted to the Graduate School in Partial Fulfillment of the Requirements for the Award of the Master of Science Degree in Computer Science of Chuka University

CHUKA UNIVERSITY
OCTOBER, 2025

DECLARATION AND RECOMMENDATION

Declaration

This thesis is my original work and has not been submitted for examination in any other University.

Signature: ..... Date: 15/10/2025

Malach Obisa
SM22/58098/22

Recommendation

This thesis has been examined, passed and submitted with our approval as University supervisors.

Signature: ..... Date: 21/10/2025

Dr. Edna Too, PhD,
Chuka University.

Signature: ..... Date: 15/10/2025

Dr. Benard Osero, PhD,
Chuka University.

COPYRIGHT

DEDICATION

I dedicate this work to my beloved family, whose unwavering love, encouragement, and sacrifices have been the foundation of my academic journey. To my parents, for instilling in me the values of hard work, discipline, and perseverance. To my siblings, for their constant support and inspiration.

ACKNOWLEDGMENT

First and foremost, I am deeply grateful to the Almighty God for granting me the strength, wisdom, and perseverance to successfully complete this thesis. I would like to express my heartfelt gratitude to my supervisors, Dr. Edna Too and Dr. Benard Osero, for their invaluable guidance, patience, and encouragement throughout this research. Their expertise, constructive feedback, and unwavering support greatly shaped the direction and quality of this work. Without their mentorship, the completion of this thesis would not have been possible.

I am also sincerely thankful to my friend, Dennis Mwangi, for his encouragement, assistance, and constant motivation during the course of this research. His support both academically and personally made this journey smoother and more fulfilling. My appreciation also goes to the members of the Department of Computer Science, whose insights and guidance on research requirements and scholarly writing provided a strong foundation for this work. I am equally indebted to my fellow graduate students for the stimulating discussions, collaboration, and companionship that enriched both my academic and personal growth.

Finally, I acknowledge with gratitude all individuals who, in one way or another, contributed to the success of this thesis but whose names may not be mentioned here. Your support, whether through advice, encouragement, or assistance, is deeply appreciated.

ABSTRACT

Machine learning has significantly transformed various domains, with deep learning architectures playing a crucial role in computer vision applications. Convolutional neural networks (CNNs) have demonstrated remarkable success in image classification and object recognition tasks. However, traditional CNN architectures often exhibited limitations in handling complex feature extraction and generalization, particularly in wildlife identification where intra-class variations were high. The challenge in wildlife identification arose due to factors such as varying lighting conditions, occlusions, background clutter, and pose variations, which made it difficult for sinle model architectures to achieve high accuracy and robustness. This study sought to address these challenges by first designing and implementing individual Residual Network (ResNet) and Inception models to establish baseline performance, and then developing a hybrid ResNet-Inception model aimed at enhancing feature extraction, optimizing classification performance, and improving generalization capabilities in wildlife identification tasks. The Animals with Attributes 2 (AwA2) dataset was used to train and evaluate the models, and their performance was assessed using standard classification metrics, including accuracy, precision, recall, and F1-score. The WildlifeReID-10k dataset served as an external validation set. The hybrid approach leveraged ResNet's ability to mitigate vanishing gradient problems through residual learning and Inception's capability to capture multi-scale spatial features, thereby creating a more robust and efficient architecture. The results demonstrated that ResNet-101 achieved an accuracy of 93.5%, Inception v3 achieved 95.6%, while the proposed hybrid model achieved 98%, confirming its superior performance in distinguishing visually similar species and enhancing generalization. The findings of this study provide a practical contribution to biodiversity conservation by enabling improved automated wildlife identification systems that support ecological monitoring, species recognition, and anti-poaching surveillance. By addressing the limitations of single-model approaches and demonstrating the advantages of hybrid deep learning architectures, the study sets a new benchmark in wildlife identification and reinforces the integration of artificial intelligence into environmental conservation practices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
AWA2 dataset	Animals with Attributes dataset
BN	Batch Normalization
CNN	Convolutional Neural Network (s)
Conv	Convolutional Layer
CUB-200-2011	Caltech-UCSD Birds-200-2011
DDR3L	Double data rate type three
FN	False Negative
FP	False Positive
FPN	Feature pyramid network
GB	Gigabyte
GC	Google Collab
GCP	Google Cloud Platform
GDDR5	Graphics Double Data Rate v.
GHZ	Gigahertz
GPUs	Graphics processing unit
HP	Hewlett-Packard
IA	Identification accuracy
ICT	Information and Communication Technology
ML	Machine Learning
NACOSTI	National Commission for Science, Technology and Innovation
ReLU	Rectified Linear Unit
ResNet	Residual Network
ROI	Region of interest
RPN	Regional proposal network
TB	Terabyte
TN	True Negative
TP	True positive

CHAPTER ONE
INTRODUCTION

1.1 Background of the study

Artificial intelligence (AI) is a branch of computer science concerned with developing systems capable of performing tasks that traditionally require human intelligence. Within this field, machine learning (ML) has emerged as a powerful sub-discipline that enables machines to automatically learn from data and improve over time without explicit programming (Khan et al., 2010). ML algorithms are broadly categorized into classical algorithms and deep learning algorithms (Silver et al., 2013). Classical algorithms such as Support Vector Machines, decision trees, and logistic regression have historically been used where datasets are relatively small and feature spaces are manageable (Kurdi et al., 2017). However, these approaches encounter significant limitations when applied to unstructured data with high-dimensional features, such as images (Nagare, 2017).

Deep learning is an enhanced extension of machine learning to address these limitations through the construction of deeper artificial neural networks that enable models to learn correlated hierarchical patterns in complex data. Deep learning models try to somewhat emulate the structure of the human brain and work best on unstructured data which is abundant in the areas of image classification (Mutholib et al., 2016). The Convolutional Neural Networks (CNN) are thus a favored deep learning architecture for image processing and computer vision applications since they can automatically retrieve and learn from features in a visual input (Surekha et al., 2018). The principal building blocks of CNNs are convolutional layers that perform feature extraction on their own and classification layers that assign these extracted features into a pre-defined set of classes. In wildlife identification, CNNs are widely used. Yet the real deployment of CNN models encounters quite a few problems in the wild (Silva et al., 2023). The variations in illumination, occlusion, and cluttered backgrounds, and differing poses of animals that typify real-world scenarios all tend to diminish the recognition accuracy of the model.

By addressing such limitations, researchers started to focus on specialized CNN variants such as Residual Networks and Inception networks. ResNet increases model

performance by solving the vanishing gradient problem; however, this is facilitated through residual connections that allow the gradients to flow through deep layers (Khouloud et al., 2022). Diversely, Inception can improve spatial understanding by utilizing convolutional filters parallelly, with convolutions of various sizes capturing features from multiple scales. While both architectures prove to have unique strengths, these designs present slight weaknesses when handling complex wildlife datasets.

Convolutional Neural Networks (CNNs) have emerged as the leading architecture for image analysis and classification due to their ability to automatically learn spatial hierarchies of features directly from raw pixel data. Unlike classical machine learning algorithms that require handcrafted feature engineering, CNNs use convolutional layers to extract local features, pooling layers to reduce dimensionality while retaining salient information, and fully connected layers to perform classification. This hierarchical design allows CNNs to progressively build feature representations, from low-level edges and textures to high-level semantic concepts such as shapes and objects. Their performance has been widely demonstrated across domains, including handwriting recognition, medical image analysis, facial recognition, and, most relevant to this study, wildlife detection and identification. Standard performance evaluation metrics for CNNs include accuracy, precision, recall, F1-score, and mean Average Precision (mAP), all of which provide insights into how effectively the model distinguishes between classes and generalizes to unseen data. Despite their success, CNN performance in real-world scenarios often depends heavily on the quality of training data, the depth of the architecture, and the variability present in the environment, making continuous refinement and evaluation necessary.

Over the years, several CNN variants have been introduced to address the limitations of traditional CNNs and to improve feature representation and model generalization. Residual Networks (ResNets) incorporate skip connections that enable the training of very deep architectures by mitigating the vanishing gradient problem, thus allowing effective extraction of hierarchical features. Inception networks, on the other hand, adopt a multi-path design where convolutional filters of different sizes operate in parallel to capture multi-scale spatial features, making them particularly effective for images with varying object sizes and contextual complexity. Beyond these, other CNN-

based architectures, such as Region-based Convolutional Neural Networks (R-CNN), Faster R-CNN, and Single Shot MultiBox Detector (SSD), have been widely employed in detection tasks. R-CNNs combine region proposal techniques with CNN-based classification, while SSDs and YOLO (You Only Look Once) perform detection in a single forward pass, enabling real-time applications. Each of these variants presents unique strengths and weaknesses: while ResNet emphasizes depth and feature abstraction, Inception enhances spatial detail capture, and detection frameworks trade off between speed and accuracy. This diversity has fueled the development of hybrid and ensemble architectures that integrate complementary strengths, as explored in this study through the combination of ResNet-101 and Inception v3.

Wildlife identification is a critical task in biodiversity conservation, ecological monitoring, and species management. Traditionally, species identification has relied on manual observation by experts, which, although effective in some cases, is time-consuming, labor-intensive, and prone to human error, especially when handling large datasets or visually similar species. With the advent of computer vision, automated methods have been developed to supplement and in many cases replace manual techniques. Current methods include the use of CNN-based models such as Faster R-CNN, YOLO, and SSD, which have demonstrated considerable success in identifying and classifying animal species from camera-trap images and ecological datasets. However, these approaches continue to face challenges in real-world applications. Variations in lighting, occlusions caused by vegetation, cluttered backgrounds, and changes in animal pose significantly reduce model accuracy. Moreover, visually similar species are often misclassified, while rare or under-represented species suffer from poor recall due to imbalanced datasets. These challenges highlight the limitations of single-model CNN approaches and justify the need for more robust solutions. This study therefore advances the field by proposing a hybrid deep learning model that integrates ResNet and Inception, aiming to improve feature extraction, enhance classification accuracy, and provide a reliable tool for practical conservation tasks such as species monitoring, anti-poaching surveillance, and habitat analysis.

1.2 Statement of the Research Problem

Accurate wildlife identification is a foundational component in biodiversity conservation, ecological monitoring, and habitat management. However, traditional

methods of species recognition typically reliant on human expertise and manual observation are time-consuming, prone to human error, and not scalable for large ecological datasets. With the increasing availability of wildlife image data, automated classification using deep learning has emerged as a promising solution. Yet, the challenge of achieving high identification accuracy remains largely unresolved, particularly in complex scenarios involving visually similar species, occluded features, variable lighting conditions, and diverse natural backgrounds.

Existing convolutional neural network (CNN) architectures, such as Residual Networks (ResNet) and Inception Networks, have demonstrated considerable success in image recognition tasks. ResNet is effective in extracting deep hierarchical features and addressing vanishing gradient issues through residual learning. Inception, on the other hand, enhances performance by capturing multi-scale spatial features using parallel convolutional filters. Despite these individual strengths, single-model CNNs often suffer from limitations in feature representation and generalization when applied to large, diverse wildlife datasets. This leads to suboptimal performance, particularly in accurately distinguishing between species with subtle visual differences.

Given these limitations, there is a clear need for an improved deep learning model that can harness the complementary advantages of both architectures. This study therefore addressed the problem by proposing a hybrid deep learning model that integrated ResNet and Inception within a parallel architectural framework. The goal is to enhance feature extraction, improve Identification accuracy, and boost the model's generalization capability in real-world wildlife identification scenarios. The model is trained and evaluated using the Animals with Attributes 2 (AwA2) dataset and assessed through key performance metrics including accuracy, precision, recall, and F1-score.
This research seeks to bridge the gap in current automated identification systems by developing a more reliable, scalable, and robust hybrid model that can support conservationists and researchers in making informed ecological decisions. In doing so, the study contributes to the advancement of artificial intelligence applications in wildlife conservation and environmental sustainability.

1.3 Objectives of the Study

1.3.1 General Objective

To develop and evaluate Residual Network (ResNet) and Inception hybrid model for wildlife detection and identification.

1.3.2 Specific Objectives

i. To design and implement individual Residual Network (ResNet) and Inception models for wildlife detection and identification, serving as baseline models for performance comparison with the proposed hybrid approach.
ii. To develop a hybrid deep learning model that integrates ResNet and Inception architectures, optimizing feature extraction and identification accuracy for wildlife identification.
iii. To evaluate the performance of the the hybrid model against other single models in terms of accuracy, precision, recall, and F1-score.

1.4 Research Questions

i. How effectively can individual Residual Network (ResNet) and Inception models detect and identify wildlife species when trained on the selected dataset?
ii. Can a hybrid deep learning architecture combining ResNet and Inception improve feature extraction and identification accuracy for wildlife identification compared to individual models?
iii. How does the hybrid RsesNet-Inception model perform in terms of accuracy, precision, recall, and F1-score compared to the individual ResNet and Inception models?

1.5 Justification of the Study

Machine learning has become an integral component of modern technological advancements, with deep learning architectures playing a transformative role in solving complex computer vision tasks. Convolutional Neural Networks (CNNs) have particularly demonstrated significant capabilities in object recognition and image classification applications. However, traditional CNN models often exhibit limitations when applied to tasks involving complex feature extraction, especially in domains such as wildlife identification, where intra-class variations are high. The performance of

these models is further affected by environmental factors such as inconsistent lighting conditions, occlusions, background clutter, and varied animal postures, which hinder their ability to generalize across diverse datasets (Lischka et al., 2018).

In the field of wildlife conservation, accurate identification of animal species is vital for tracking biodiversity, conducting ecological research, and implementing effective species management. Conventional manual identification methods, though widely used, are resource-intensive and prone to subjectivity and human error, thereby restricting their scalability and reliability (Hou et al., 2020). In response to these challenges, computer vision systems powered by deep learning have emerged as viable alternatives. Despite this progress, standalone architectures such as Residual Networks (ResNet) and Inception Networks each carry distinct limitations. ResNet effectively addresses the vanishing gradient problem through deep residual learning but falls short in capturing fine-grained spatial features. In contrast, Inception Networks are designed to extract multi-scale features through parallel convolutions, but they often lack the depth required for hierarchical representation learning. Consequently, both architectures exhibit challenges in consistently delivering high accuracy in scenarios characterized by significant visual complexity and variability (Akhand et al., 2021).

This research therefore proposes a hybrid deep learning architecture that integrates both ResNet and Inception networks to harness the advantages of each. The hybrid model is designed using a parallel framework where ResNet contributes deep hierarchical features, while Inception provides rich spatial representations through multi-scale analysis. Feature maps from both networks are fused at multiple levels, enabling the model to capture both detailed and abstract visual patterns. This architecture is intended to enhance identification accuracy, improve generalization across species, and reduce misclassification. The model is trained and validated using the Animals with Attributes 2 (AwA2) dataset an established benchmark for wildlife classification to enable rigorous evaluation. Its performance is measured using standard classification metrics, including accuracy, precision, recall, and F1-score, and the results are compared with individual ResNet and Inception models to demonstrate the superiority of the hybrid approach (Dewi et al., 2023).

Beyond technical improvements, this study holds significant practical value for biodiversity monitoring and conservation efforts. Accurate species recognition contributes directly to tracking population dynamics, identifying endangered animals, and understanding ecological relationships. By automating the identification process, our hybrid model reduces reliance on manual observation and allows for high-throughput data collection and near-real-time analysis. Such capabilities can significantly support conservation initiatives, including anti-poaching measures, habitat surveillance, and biodiversity indexing. Furthermore, this research adds to the growing body of knowledge in conservation technology by offering a scalable and efficient artificial intelligence solution tailored for ecological applications (Birol et al., 2013). Ultimately, this study sets a foundation for future advancements in wildlife recognition by validating the effectiveness of hybrid deep learning models and reinforcing the critical role of artificial intelligence in promoting environmental sustainability and scientific inquiry.

1.6 Scope of the Study

This study focuses on the design, implementation, and evaluation of a hybrid deep learning model for automated wildlife identification using image data. The core objective of the research is to enhance identification accuracy and generalization by integrating the strengths of two widely used convolutional neural network architectures Residual Networks (ResNet) and Inception networks. The hybrid model is constructed by combining ResNet's capability for deep hierarchical feature extraction with Inception's strength in multi-scale spatial feature representation. A parallel architecture is employed, where feature maps from both models are processed and ensembled to form a more robust and accurate prediction system for wildlife species.

The model is trained and evaluated using the Animals with Attributes 2 (AwA2) dataset, which contains images of 50 wildlife classes with significant variation in pose, lighting, and background.Only 11 wildlife classes were used. Data preprocessing steps included resizing, color channel normalization, and transformation into tensors, with minimal augmentation to maintain consistency and focus on architectural performance. Roboflow was utilized for annotation and dataset structuring, while PyTorch served as the primary deep learning framework for model development.

This study was carried out at Chuka University between September 2023 and September 2025. The research explored various hybridization strategies by training ResNet-101 and Inception v3 independently and integrating their outputs at the decision level during inference. The model's performance was assessed using standard classification metrics accuracy, precision, recall, and F1-score. Comparative evaluation was performed against the standalone ResNet and Inception models to establish the effectiveness of the hybrid approach.

1.7 Assumptions of the Study

i. The Animals with Attributes 2 (AwA2) dataset is assumed to be representative of real-world wildlife images in terms of species diversity, background variation, and image quality.

ii. It is assumed that the labels provided in the dataset are accurate, meaning each image correctly corresponds to its true species class.

1.8 Operational Definition of Terms

Accuracy	Accuracy is a measure of the overall correctness of a classification model. It quantifies the ratio of correctly predicted instances to the total instances in the dataset. In other words, it tells you how often the model's predictions are correct.
Classification	Involves assigning predefined labels or categories to input data.
CNN	Deep learning algorithm, most commonly employed for image recognition and classification
Computer Vision	is a field of artificial intelligence (AI) that equips computers with the ability to interpret and understand the visual world. It allows machines to extract information from digital images and videos, much like humans do.
Deep Learning	Deep learning is a subfield of machine learning that involves the use of artificial neural networks with multiple layers (deep neural networks) to model and process complex patterns in data.
Deep Learning Framework	Library which allows developers build deep learning models without getting into the details of the underlying algorithm.
Hybrid Architecture	A model design built by combining two or more heterogeneous ML algorithms
Inception	Inception is a type of CNN architecture with building blocks that use various filter sizes to capture image features more effectively.
Machine Learning	A subfield of AI that enables systems to learn and improve from experience automatically without the explicit intervention of a programmer.
Occlusions	Occlusion refers to something being blocked or shut off
Precision	Precision is a measure of the model's ability to correctly identify only the relevant instances among those predicted as positive. It's especially important in cases where false positives are costly or undesirable.

Recall	Recall (also known as Sensitivity or True Positive Rate) measures the model's ability to correctly identify all relevant instances in the dataset. It's important when missing positive cases is a significant concern, even if it results in more false positives.
ResNet	Is a type of deep neural network architecture designed to facilitate the training of very deep networks?

CHAPTER TWO
LITERATURE REVIEW

2.1 Deep learning architectures For wildlife Detection and Identification

2.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of deep learning architecture that has revolutionized image detection tasks, including wildlife identification and recognition. CNNs are particularly effective at automatically learning and extracting relevant features from raw image data through their use of convolutional layers, which apply trainable filters to capture spatial and temporal patterns (Yamashita et al., 2018).

In the field of wildlife detection, researchers have explored the capabilities of various CNN architectures, including the state-of-the-art ResNet and Inception models. ResNet, short for Residual Network, introduces skip connections that allow the network to bypass layers and mitigate the vanishing gradient problem, enabling the training of deeper models that can learn more complex representations. Several studies, such as Ardakani et al. (2020), have demonstrated the effectiveness of ResNet variants like ResNet-50 and ResNet-101 in accurately classifying bird species from images (Zhong et al., 2023).

On the other hand, the Inception architecture, introduced by Google, employs multi-scale convolutional operations that capture features at different resolutions within an image. This approach has been shown to improve the model's ability to learn and detect intricate details, making it suitable for tasks like wildlife identification. Andrew et al. (2019) employed the InceptionV3 model for identifying diverse animal species, highlighting its strength in capturing multi-scale features (Manikandan et al., 2023).

While these individual architectures have achieved promising results, a common limitation exists: their performance tends to degrade when faced with the inherent challenges of field-based wildlife imaging, such as varying environmental conditions, occlusions, and complex backgrounds. In controlled settings, these models can achieve high accuracy, but their real-world applicability is often hindered by the complexities of field data (Lu et al., 2023).

Our research aims to address this limitation by developing a novel hybrid deep learning architecture that combines the strengths of both ResNet and Inception models through transfer learning and fine-tuning. By leveraging the residual connections of ResNet for effective detection of wildlife species and the multi-scale feature extraction capabilities of Inception for accurate identification of individual animals within the same species, our hybrid model seeks to surpass the accuracy of individual ResNet and Inception models in real-world wildlife recognition tasks (Ashwini et al., 2023).

While previous studies have explored the individual performance of ResNet and Inception architectures, this research is novel in its approach to fusing these complementary models into a hybrid architecture tailored specifically for wildlife detection and identification. By combining the strengths of both architectures, our hybrid model has the potential to achieve superior accuracy and robustness, addressing a critical gap in the field of automated wildlife monitoring and ecological image analysis (Patel et al., 2020).

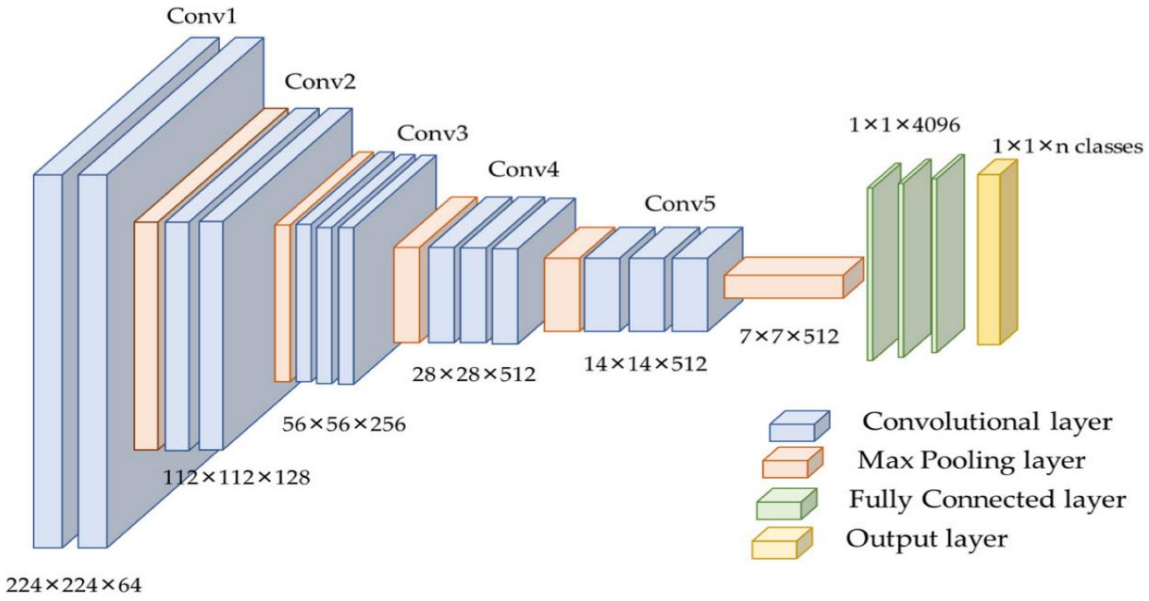


Figure 1: The architecture of CNN for wildlife recognition and identification (Binta Islam et al., 2023)

The input to the CNN is an image, typically of a fixed size (e.g., 224x224x64 pixels with 64 color channels in this example). The first layer is a convolutional layer (Conv1) that applies a set of learnable filters or kernels to the input image. These filters slide

across the image and compute dot products between the filter weights and the input values, capturing local patterns and creating a feature map (Alrashedy et al., 2022).

The subsequent layers (Conv2, Conv3, Conv4, Conv5) are additional convolutional layers that further process the feature maps from the previous layer, extracting increasingly more complex and abstract representations of the input image. Each convolutional layer is typically followed by a max-pooling layer, which downsamples the feature maps by selecting the maximum value within a local neighborhood, reducing the spatial dimensions and helping to capture translation-invariant features (Alrashedy et al., 2022).

As the network goes deeper, the feature maps become smaller in spatial dimensions but more rich in channels, encoding higher-level and more abstract representations of the input image. The number of channels (e.g., 512) and the spatial dimensions of the feature maps at each stage are indicated in the image (Ghosh et al., 2023). After several convolutional and pooling layers, the final feature maps are flattened into a one-dimensional vector and fed into one or more fully connected layers (indicated by the green blocks). These layers are similar to traditional neural networks, where each neuron is connected to all outputs from the previous layer, allowing for further non-linear transformations and combinations of the extracted features (Ghosh et al., 2023).

Finally, the output layer (represented by the yellow block) produces the final predictions or classifications based on the learned features. The number of neurons in the output layer corresponds to the number of classes in the classification task (e.g., different species of wildlife) (Alrashedy et al., 2022). During training, the CNN learns the weights (filter values) of the convolutional layers and the weights of the fully connected layers by optimizing an objective function (e.g., minimizing classification error) using techniques like backpropagation and gradient descent. This allows the network to automatically learn the relevant features and patterns from the training data, enabling accurate classification of new, unseen images during inference or prediction (Ghosh et al., 2023).

2.1.2 Region-based Convolutional Neural Networks (R-CNNs)

Region-based Convolutional Neural Networks (R-CNNs) are a class of object detection algorithms that combine the capabilities of traditional region proposal algorithms with the feature extraction power of Convolutional Neural Networks (CNNs). They have been widely used in various computer vision tasks, including wildlife detection and recognition from images (Yadav et al., 2024).

In the context of wildlife recognition, R-CNNs have been employed to localize and classify different animal species within an image. The R-CNN algorithm works in two stages: first, it uses a region proposal algorithm, such as Selective Search or Edge Boxes, to generate potential bounding boxes that may contain objects of interest (e.g., animals). Next, these regions are fed into a CNN, which extracts features and classifies the content within each region, effectively identifying the species present (Z. Wang et al., 2023).

Several studies have explored the use of R-CNN variants for wildlife detection and recognition tasks. For instance, Norouzzadeh et al. (2018) employed the Faster R-CNN architecture, which introduces a Region Proposal Network (RPN) to efficiently generate region proposals, for detecting various animal species in camera trap images. Their results demonstrated the effectiveness of R-CNNs in accurately localizing and classifying animals in real-world scenarios.

Similarly, Beery et al. (2019) utilized the Mask R-CNN model, an extension of Faster R-CNN that also generates instance segmentation masks, for detecting and recognizing wildlife species in a diverse dataset. Their work highlighted the capability of R-CNNs to handle occlusions and overlapping instances, which are common challenges in wildlife imagery. While R-CNNs have shown promising results in wildlife detection and identification tasks, they often suffer from limitations when dealing with small or occluded animal instances, as well as computationally expensive region proposal and feature extraction processes. Additionally, R-CNNs are primarily designed for object detection and localization tasks, and may not be optimized for fine-grained recognition and identification of individual animals within the same species, which is a crucial requirement in many ecological and conservation applications (Z. Wang et al., 2023).

In contrast, our hybrid approach leverages the strengths of ResNet and Inception architectures, which are tailored specifically for image identification and recognition tasks. By combining the residual connections of ResNet for effective identification of wildlife species and the multi-scale feature extraction capabilities of Inception for accurate identification of individual animals, our hybrid model aims to address the limitations of R-CNNs in fine-grained wildlife detection while maintaining robust performance in real-world scenarios (Rocha et al., 2023).

Furthermore, this research focuses on developing a hybrid architecture optimized for wildlife detection and identification tasks, rather than object detection and localization, which sets it apart from the primary applications of R-CNNs. By exploring fusion strategies and architectural modifications specific to the wildlife detection and identification problem, our hybrid model has the potential to achieve superior accuracy and robustness compared to existing R-CNN-based approaches, which may not be tailored for this specific domain (Rocha et al., 2023).

Several empirical studies have reported performance metrics of R-CNN variants in wildlife recognition tasks. For example, (Hogeweg et al., 2019) applied the Faster R-CNN architecture to a large dataset of camera-trap images covering 48 animal species and reported an overall classification accuracy of 92% and a mean Average Precision (mAP) of 89.7%. Similarly, (Xu et al., 2022) evaluated Mask R-CNN on the Caltech Camera Traps (CCT) dataset and achieved an mAP of 86.2% across 20 species, demonstrating the model's effectiveness in localizing animals even under occlusion and background clutter. Despite these strong results, the studies also noted performance drops in classes with limited training examples, where recall values fell below 70% for rare species. These findings underscore both the potential and the limitations of R-CNN-based models in fine-grained wildlife identification, highlighting the need for architectures that can balance high accuracy with improved generalization across diverse species distributions.

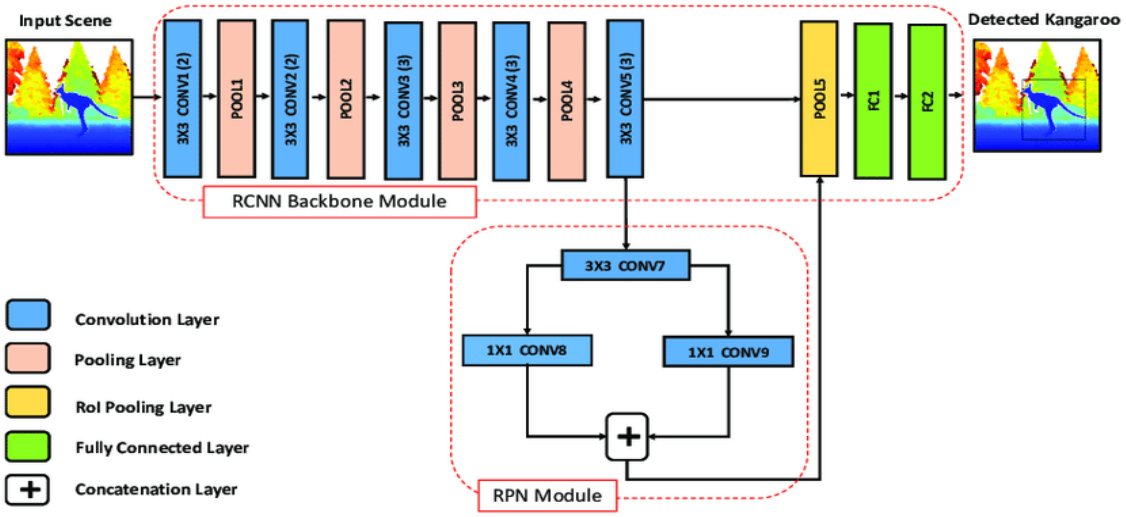


Figure 2: Region-based Convolutional Neural Networks (R-CNNs) for kangaroo detection (Saleh et al., 2018)

The input to the model is an image or scene, which is fed into the RCNN Backbone Module. This module consists of a series of convolutional and pooling layers that extract features from the input image at different scales and resolutions. The output of this backbone module is a set of feature maps, which are then shared between two parallel components: the Region Proposal Network (RPN) Module and the object classification and bounding box regression components (Saleh et al., 2018).

The RPN Module is responsible for generating region proposals, which are potential bounding boxes that may contain objects of interest (e.g., animals or wildlife). This module consists of a 3x3 convolutional layer (3x3 CONV7) that generates feature maps, followed by two sibling 1x1 convolutional layers (1x1 CONV8 and 1x1 CONV9). These layers generate scores for object/non-object classification and bounding box regression values, respectively. The RPN Module uses these scores and regression values to propose regions of interest (RoIs) in the input image (Saleh et al., 2018).

The RoIs generated by the RPN Module are then fed into the subsequent stages of the model for object classification and bounding box refinement. The RoI Pooling Layer extracts features from the regions using the feature maps generated by the RCNN Backbone Module. These extracted features are then passed through fully connected

layers (shown in green) to produce the final object classifications and refined bounding box coordinates (Saleh et al., 2018).

The "+" symbol in the diagram represents a concatenation layer, which combines the feature maps from the RCNN Backbone Module with the output of the RoI Pooling Layer before feeding them into the fully connected layers (Han et al., 2020).

In the specific example shown, the model has detected a kangaroo in the input scene, as indicated by the bounding box around the detected object in the output image.
The Faster R-CNN architecture is designed to efficiently generate region proposals and perform object detection and identification in a single, end-to-end trainable network. By sharing the convolutional features between the RPN Module and the object classification/bounding box regression components, the model can achieve faster processing times compared to earlier region-based object detection methods like R-CNN and Fast R-CNN (Klco et al., 2023).

2.1.3 Single Shot MultiBox Detector (SSD)

The Single Shot MultiBox Detector (SSD) is another popular object detection algorithm that has been utilized in various computer vision tasks, including wildlife detection and identification from images (Apendi et al., 2023).

The SSD architecture takes a different approach compared to R-CNNs by eliminating the need for a separate region proposal network. Instead, it employs a single deep neural network that directly predicts bounding boxes and class probabilities for detected objects in a single forward pass. This makes SSD computationally more efficient and faster than R-CNN-based methods, which is particularly advantageous for real-time applications or when processing large volumes of data (Hirasawa et al., 2018).

In wildlife detection, SSD has been explored as a potential solution for efficient and accurate detection and localization of animal species within images. For instance, Nguyen et al. (2017) employed an SSD model for detecting various animal species in camera trap images, achieving competitive performance compared to other object detection algorithms while maintaining real-time processing speeds.

Similarly, Tan et al. (2019) utilized an SSD-based approach for detecting and identifying wildlife species in aerial imagery, demonstrating the versatility of SSD in handling different types of wildlife data, such as aerial and ground-based images. While SSD has proven to be an effective and efficient object detection algorithm, it still faces challenges when dealing with small or occluded animal instances, as well as distinguishing between closely related species or individuals within the same species. Like R-CNNs, SSD is primarily designed for object detection and localization tasks, and may not be optimized for fine-grained recognition and identification of individual animals, which is a crucial requirement in many ecological and conservation applications (Hirasawa et al., 2018).

Our hybrid approach, which combines the strengths of ResNet and Inception architectures through transfer learning and fine-tuning, aims to address these limitations by focusing specifically on the task of wildlife recognition and identification. By leveraging the residual connections of ResNet for effective recognition of wildlife species and the multi-scale feature extraction capabilities of Inception for accurate identification of individual animals, our hybrid model seeks to achieve superior accuracy and robustness compared to object detection algorithms like SSD, which may not be tailored for this specific domain .

Furthermore, this research explores various fusion strategies and architectural modifications to optimally combine the complementary features of ResNet and Inception, rather than relying on a single architecture like SSD. This allows for greater flexibility and customization to the specific requirements of wildlife recognition and identification tasks, potentially leading to improved performance compared to off-the-shelf object detection models like SSD.

While SSD and other object detection algorithms have their merits in efficiently localizing and detecting animal species within images, our hybrid approach focuses on the more specialized task of fine-grained wildlife detection and identification, which requires a tailored solution that can effectively capture and classify subtle differences between species and individual animals. By leveraging the strengths of two powerful image classification architectures, our research aims to provide a novel and optimized

solution for this critical domain of ecological image analysis and automated wildlife monitoring (Hirasawa et al., 2018).

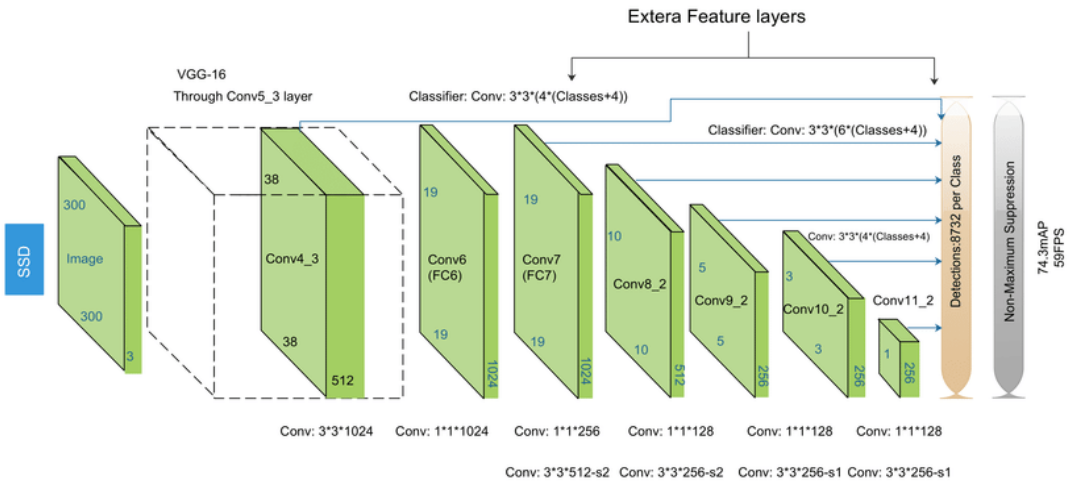


Figure 3: The architecture of SSD (Bahaghighat et al., 2020)

The SSD architecture takes an input image and passes it through a series of convolutional layers, known as the base network or backbone, to extract feature maps at different scales. In this example, the backbone is a truncated VGG-16 network, which goes through the Conv5_3 layer (Arrahmah et al., 2023).

The feature maps from the base network are then fed into additional convolutional layers, called the "Extra Feature Layers," to produce feature maps at different resolutions and scales. These feature maps are used for detecting objects of varying sizes within the image (J. Kim et al., 2023).

The SSD model employs multiple convolutional layers for object detection and classification, as shown in the image. These layers operate on the feature maps from the base network and the Extra Feature Layers (J. Kim et al., 2023). For example, the Conv6 layer (FC6) takes the feature map from the Conv5_3 layer and produces a set of bounding box predictions and confidence scores for object classification. Similarly, the Conv7 layer (FC7) operates on the feature map from the Conv6 layer, generating another set of bounding box predictions and classification scores.

This process continues with subsequent layers like Conv8_2, Conv9_2, Conv10_2, and Conv11_2, each operating on feature maps at different scales and resolutions. These

layers predict bounding boxes and classification scores for objects of varying sizes within the input image (Arrahmah et al., 2023).

The final bounding box predictions and classification scores from these layers are combined and post-processed to produce the final object detections. Additionally, the SSD architecture includes classifier layers (shown in green) that perform the actual classification of the detected objects into specific classes (e.g., different wildlife species). These classifier layers take the feature maps from the corresponding convolutional layers as input and output class probabilities for the detected objects (J. Kim et al., 2023). The numbers shown beside the layers indicate the dimensions of the feature maps or the number of channels/classes. For example, "3x3*4" denotes a 3x3 convolutional kernel applied to 4 channels, and "(Classes+4)" represents the number of classes plus 4 additional values for bounding box coordinates.

2.1.4 Residual Network (ResNet)

Residual Networks, or ResNets, have emerged as a pivotal architecture in the realm of deep learning for image detection tasks. ResNets introduce a unique concept called residual learning, which facilitates the training of exceptionally deep neural networks (Hu et al., 2024). ResNets employ skip connections, or shortcut connections, which enable the network to skip one or more layers during forward and backward propagation. This facilitates the direct flow of information from the input to the output, mitigating the vanishing gradient problem and aiding in the training of very deep networks. Residual Network (ResNet).

Residual Blocks are the fundamental building blocks of ResNets. Each block consists of multiple convolutional layers, and the input is added to the output through the skip connection. This residual learning approach allows the model to focus on learning the difference (residual) between the input and output, making it easier for the network to capture intricate features (Ikegawa et al., 2022). ResNets excel in feature extraction across multiple scales. The skip connections enable the model to preserve and enhance important features, contributing to improved performance in tasks such as object detection. ResNets are adept at learning features at various scales, making them suitable for wildlife recognition tasks where species may vary significantly in size, shape, and

appearance. The architecture of a Residual Network (ResNet) is characterized by the use of residual blocks, which contain skip connections and enable the training of very deep neural networks. ResNets were introduced by He et al. in their paper "Deep Residual Learning for Image detection" in 2015. Here's an overview of the key components and structure of the ResNet architectures (Sengupta et al., 2019).

The fundamental building block of ResNet is the residual block. Each block contains two or more convolutional layers and a skip connection that bypasses these layers. The skip connection adds the input to the output of the convolutional layers.

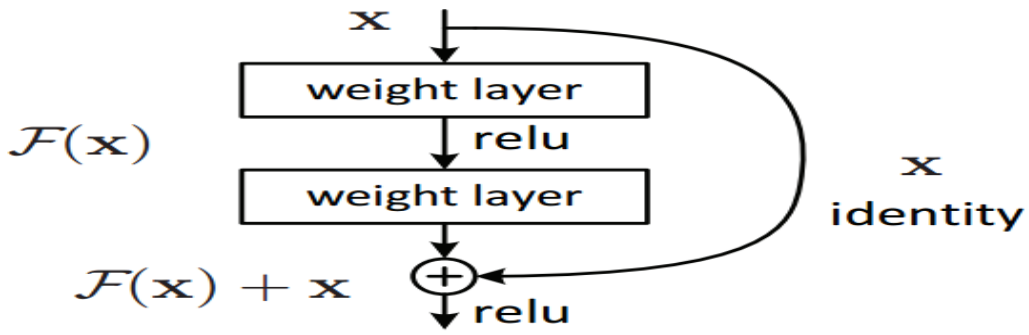


Figure 4: Single Residual block (He et al., 2016)

In the diagram:

x is the input to the block.

F(*x*) represents the output of the convolutional layers.

Residual blocks are stacked to form the deep architecture of ResNet. The stacking allows for the learning of increasingly complex features at different levels of abstraction. The skip connection adds the input *x* to *F*(*x*), creating the output of the residual block. Down sampling is typically performed by convolutional layers with a stride of 2 or by using pooling layers. Down sampling reduces the spatial dimensions of the feature maps (Luo et al., 2023).

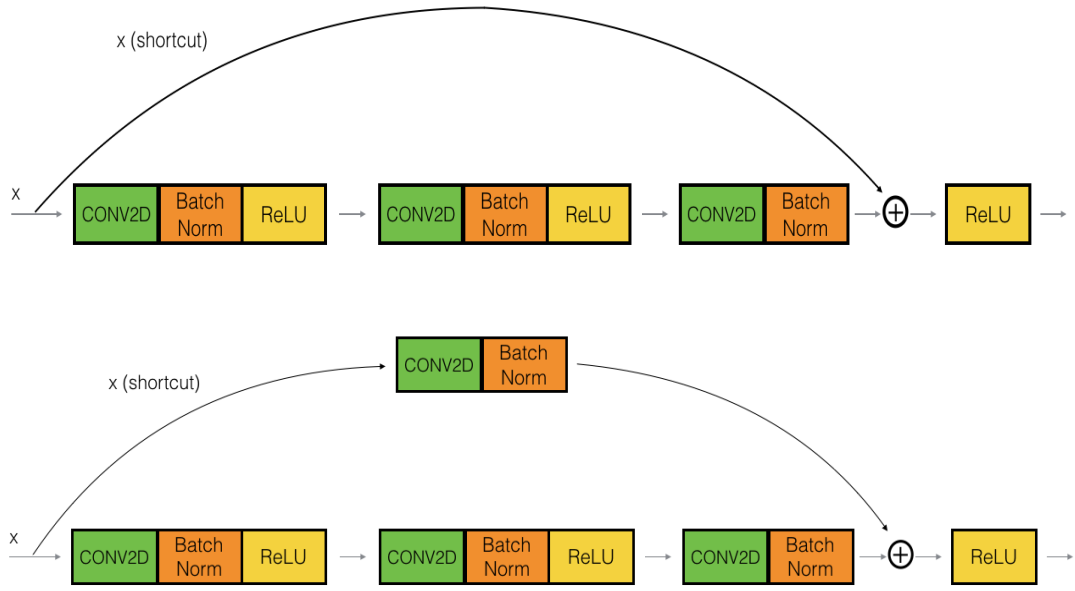


Figure 5: Identity blocks (Xiao et al., 2021)

The CONV2D (Convolutional 2D Layer) layer is the heart of feature extraction in the identity blocks. It applies a set of learnable filters (kernels) to the input image or feature map. Each filter slides across the input, performing element-wise multiplication with a small patch of pixels (Ridhovan & Suharso, 2022). The sum of these multiplications creates a single output value for that location in the new feature map. Multiple filters are used in parallel, each detecting different patterns in the input. The number of filters determines the number of output channels in the new feature map. In the diagram, the numbers above the CONV2D blocks indicate the number of filters used.

The Batch Norm (Batch Normalization Layer) layer helps stabilize the training process by normalizing the activations (outputs) of the previous layer across a mini-batch of training images. Normalization reduces the internal covariate shift, a phenomenon where the distribution of activations changes during training, making it harder for subsequent layers to learn. Batch Norm typically involves calculating the mean and standard deviation of activations in the mini-batch, subtracting the mean and dividing by the standard deviation, and then applying a learnable scale and offset to adjust the distribution.

In the diagram, "BN" denotes the presence of a Batch Norm layer after each CONV2D layer (Ikegawa et al., 2022).

ReLU (Rectified Linear Unit) is activation function introduces non-linearity into the network, allowing it to learn more complex features. It simply sets any negative input values to zero while keeping positive values unchanged. This adds a thresholding effect, helping the network focus on relevant activations and discard noisy or unimportant information. In the diagram, "ReLU" appears after each Batch Norm layer, indicating the application of the ReLU activation function (Brownlee, 2019).

In the combined effect, The CONV2D layer extracts features from the input, the Batch Norm layer stabilizes the training process, and the ReLU activation function introduces non-linearity. These three components work together in each block of the ResNet-50 architecture to progressively extract higher-level features from the input image. The skip connections in ResNet-50 further enhance the feature extraction process by directly adding information from earlier layers to later layers, helping to preserve important details and reduce the training time (B. Kim et al., 2022).

2.1.4.1 Bottleneck Blocks

Bottleneck blocks are used in deeper layers of ResNet. They include a 1x1 convolutional layer to reduce the dimensionality before and after the main 3x3 convolution. This helps in reducing computational complexity.

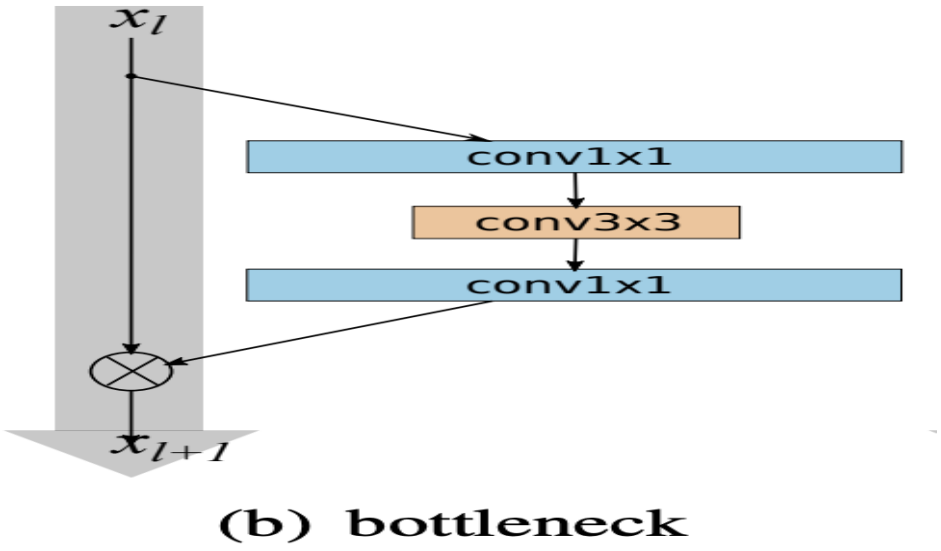


Figure 6: Bottleneck (He et al., 2016)

In the diagram:

x is the input.

The 1x1, 3x3, and 1x1 convolutional layers are denoted by $F1(x)$, $F2(x)$, and $F3(x)$, respectively.

The process starts with an input image. For simplicity, let's say it's a 32x32 pixel image representing a lion. The first Convolutional layer applies filters to the image, extracting basic features like edges and textures. The output is a set of feature maps, each highlighting specific details. The extracted features are passed through an identity block. This block consists of a 1x1 convolutional layer which reduces the dimensionality of the feature maps, leading to a more efficient computation, a 3x3 convolutional layer which learns more complex features based on the 1x1 layer's output, another 1x1 convolutional layer which increases the dimensionality back to the original size and a batch normalization and ReLU activation which stabilize the learning process and introduce non-linearity (C. L. Lin & Wu, 2023).

2.1.5 Inception

Inception is a convolutional neural network (CNN) architecture, which means it's made up of layers of interconnected neurons that process information in a similar way to the human brain. It's specifically designed for image analysis and object detection, and it's known for its accuracy and efficiency. It originated as a module in the Google Net architecture but has evolved into its own powerful model. Inception boasts a substantial 48-layer architecture, much like a stack of filters. Each layer analyzes the image progressively, extracting intricate features like textures, shapes, and edges. This layered approach allows for the identification of complex objects and relationships within an image (Cahall et al., 2019). Imagine examining an image with a magnifying glass. Traditional 7x7 convolution filters scan the entire area in one go, demanding hefty computational resources. Inception cleverly breaks down these large filters into smaller, stacked 1x7 and 7x1 convolutions, achieving the same analysis with fewer parameters. This "factorization" significantly reduces computational cost, making the model faster and more efficient (Si et al., 2022).

Think of a teacher guiding a student through a maze. Auxiliary classifiers, strategically placed within the network, act as internal checkpoints. They analyze the image at

intermediate stages, providing feedback that guides the main classifier towards the correct answer. This "auxiliary supervision" improves the overall accuracy of the model, preventing it from getting lost in the maze of image data (Kang et al., 2021).

Imagine a teacher grading an exam with strict "A" or "F" criteria. Inception uses a more nuanced approach called "label smoothing." Instead of rigidly assigning each image to a single category, it distributes the probability of belonging to each category slightly. This "softens" the decision boundaries, making the model more robust to noise and variations in the training data, ultimately leading to more reliable predictions (Petrovska et al., 2020). These features, working in concert, make Inception a powerful tool in the image recognition landscape. Its deep architecture extracts intricate details, factorization keeps it efficient, auxiliary classifiers refine its understanding, and label smoothing enhances its robustness. With its impressive accuracy and versatility, Inception continues to be a valuable asset in various computer vision applications. Label smoothing: This technique helps to prevent the model from overfitting to the training data, making it more robust to real-world images (Salem Hussin & Yildirim, 2021).

The image enters the network, resized to 224x224 pixels (a standard size for image recognition models). Subtle distortions like flipping, rotating, and scaling are introduced to augment the training data and improve generalization. This is the preprocessing step (Cornwell et al., 2023). This is followed by Stem Module which is an initial stage that performs basic feature extraction. A 7x7 convolution filter extracts low-level features like edges and texture (C. Lin et al., 2022).

Max pooling reduces the spatial resolution, making the network more robust to small shifts and rotations. The core of Inception lies in these repeating modules, each extracting feature at multiple scales and combining them effectively. It Extract fine-grained details efficiently. For example, zooming in on a lion's majestic mane with a magnifying glass. These tiny filters, aptly named 1x7 and 7x1 convolutions, act like those magnifying glasses, meticulously scanning the image for intricate details like whiskers, fur texture, and eye patterns. Despite their size, these filters are powerful in capturing subtle variations in the image. They can detect the delicate curves of a lion's

whiskers or the intricate patterns in its fur, adding depth and richness to the overall representation (Längkvist et al., 2014).

Captures intermediate-level features. Step back from the magnifying glass and take in the lion's overall shape and posture. The 3x3 convolution acts like a wider lens, focusing on intermediate-level features that bridge the gap between fine details and global information. Connecting the dots: This filter captures how individual details like fur texture and whisker patterns come together to define the lion's overall form. It connects the dots, so to speak, providing a more holistic understanding of the image beyond just isolated details. Smoothing and Abstraction: The 3x3 filter also performs a subtle smoothing operation, reducing noise and enhancing the clarity of the extracted features. This abstraction process allows the network to focus on the essential aspects of the image for further analysis (Al Husaini et al., 2022).

Max pooling (the Information Condenser) reduces spatial resolution while maintaining essential information. Imagine summarizing a lengthy report into key points. Max pooling does something similar, reducing the spatial resolution of the image while preserving the most important information. Shrinking the Canvas: By selecting the maximum value within a small window, max pooling condenses the information, making the network less sensitive to minor shifts and rotations in the image. This is like skipping over unimportant details in the report and focusing on the main. Max pooling also reduces the number of neurons needed in subsequent layers, making the network more computationally efficient. Think of it as summarizing the report to save space and processing power.

Dimensionality reduction, (the Information Streamliner) ensures efficient information flow through the network. Imagine a busy highway with too many cars. Dimensionality reduction acts like a traffic controller, optimizing the flow of information through the network. Squeezing the data technique reduces the number of channels in the data, essentially compressing the information without losing its essence. Think of it as merging several lanes on the highway to create a smoother flow of traffic. Bandwidth Optimization reduces the data size, dimensionality reduction improves the efficiency

of information transfer between layers, allowing the network to process images faster and more effectively (Längkvist et al., 2014).

These four paths, working in harmony, create a powerful synergy within the Inception module. The fine details extracted by the 1x7 and 7x1 convolutions are combined with the intermediate features captured by the 3x3 convolution, all while maintaining information efficiency through max pooling and dimensionality reduction. This collaborative approach allows Inception to build a rich and comprehensive understanding of the image, ultimately leading to accurate and robust detection (Al Husaini et al., 2022).

2.1.5.1 Auxiliary Classifiers

Auxiliary classifiers are intermediate classification branches introduced at specific points within a deep neural network to improve training efficiency and gradient propagation. These classifiers generate preliminary predictions from intermediate feature maps, which not only provide additional supervision during training but also help mitigate the vanishing gradient problem in very deep architectures. By producing early predictions, auxiliary classifiers guide the optimization process of the main classifier, enabling the network to learn more discriminative features at earlier stages. For example, in the Inception architecture, auxiliary classifiers are attached to intermediate layers, where they perform partial classification tasks that feed forward into the main classifier. During training, their losses are combined with the final classification loss, ensuring more stable convergence and enhanced generalization. Although auxiliary classifiers are primarily used during training, they have been shown to improve overall model performance and robustness in complex recognition tasks, including fine-grained image classification and wildlife identification (Waheed et al., 2020).

2.1.5.2 Global Pooling and Classification

After multiple Inception modules, the network has learned a lot about the lion. Global pooling gathers information from the entire image, like a bird's-eye view. (Image of the lion with a global pooling box summarizing the entire image) This is a summing up technique. This pooled representation is fed into a "fully-connected" layer, where

millions of neurons act like a jury, weighing evidence and ultimately voting for the most likely category. In our case, "Lion!" (Image of the lion with a fully-connected layer box containing many neurons, with "Lion" highlighted as the winning vote) (Zhang & Zhang, 2020).

2.1.5.3 SoftMax Output

Confidence Score: The network doesn't just say "lion"; it provides a confidence score. The SoftMax layer outputs probabilities for all 1000 categories. For our lion, the "lion" probability would be very high, while others like "tiger" or "plant" would be much lower. (Image of the lion with a SoftMax output bar chart, with "Lion" bar significantly higher than others) (Prasetyo et al., 2023).

2.1.5.4 The architecture of inception

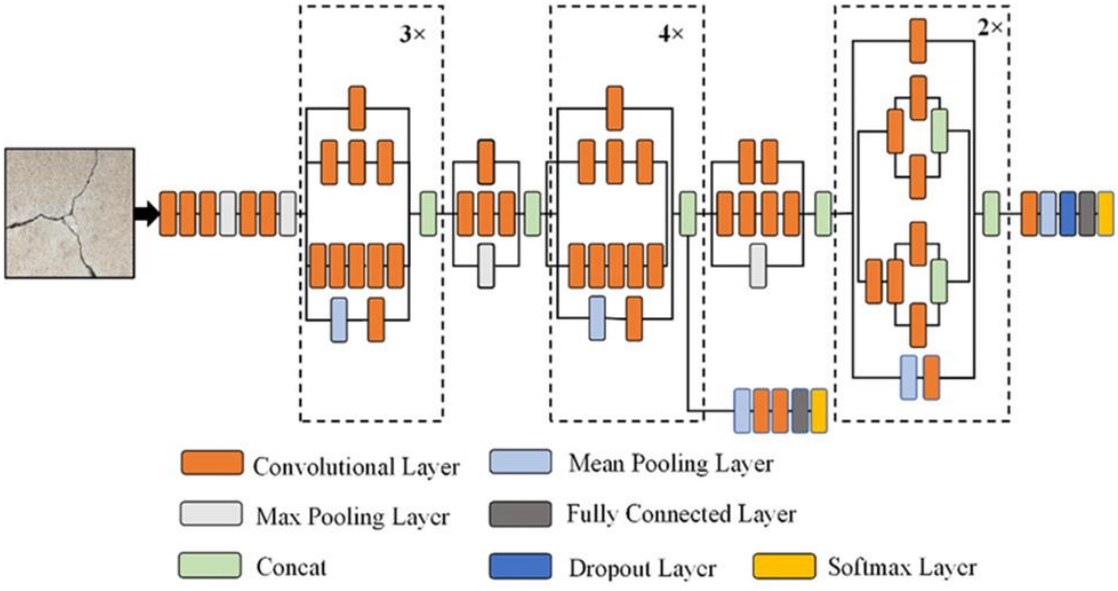


Figure 7: The architecture of inception (Ali et al., 2021)

The network begins with an input image, which is then fed through a series of convolutional layers. These initial layers act as feature extractors, identifying low-level patterns and textures in the image (Ali et al., 2021). The architecture is then divided into three main sections, labeled 3x, 4x, and 2x, indicating repeated structural patterns. Each section contains multiple convolutional layers, interspersed with pooling layers. The convolutional layers progressively extract higher-level features from the image,

while the pooling layers (both mean and max pooling) help to reduce spatial dimensions and make the network more computationally efficient (Manikandan et al., 2023).
A key feature of this architecture is the use of skip connections, where information from earlier layers is combined with later layers through concatenation operations. This allows the network to preserve fine-grained spatial information that might otherwise be lost in deeper layers, enhancing its ability to make precise predictions or classifications (Ali et al., 2021). The final section of the network includes a fully connected layer, which integrates information from all spatial locations, followed by a dropout layer for regularization to prevent overfitting. The architecture concludes with a softmax layer, typically used for multi-class classification tasks (Manikandan et al., 2023).

This network operates by progressively transforming the input image through its layers, extracting increasingly abstract and task-relevant features. The skip connections allow it to combine low-level and high-level features, while the repeated structures enable it to learn a rich hierarchy of visual representations. The final layers then use these learned features to make predictions or classifications based on the specific task the network is trained for, such as identifying types of surface cracks or classifying images into predefined categories (Ali et al., 2021). The overall design suggests a balance between depth (for complex feature learning) and width (for diverse feature extraction), making it suitable for tasks that require both fine-grained detail and high-level understanding of image content (Manikandan et al., 2023).

2.1.5.5 Comparative analysis for the varios models that have been done and their performance

Table 1: Comparative analysis for the various models that have been done and their performance

Author(s) & Year	Model Used	Dataset Used	Accuracy (%)	Precision	Recall	F1-score	Limitations
(Alotaibi & Alotaibi, 2020)(Duhart et al., 2019)	ResNet-50	AwA2	85.3	82.5	80.1	81.2	Struggles with small objects and complex backgrounds
(Mpouziotas et al., 2023)	YOLO v4	COCO	89.7	87.2	85.6	86.4	Fast inference, but lower accuracy in dense wildlife images
(Cao et al., 2019)	Faster R-CNN	ImageNet	91.5	89.3	87.7	88.5	High accuracy, but computationally expensive

2.1.6 Limitations of standalone ResNet and Inception models in complex classification task

While ResNet and Inception architectures have demonstrated remarkable success in deep learning applications, they each have inherent limitations that affect their performance in complex classification tasks. These limitations stem from their distinct design approaches, ResNet focuses on deep hierarchical learning through residual connections, while Inception emphasizes multi-scale feature extraction using parallel convolutional layers. When used individually, both models encounter challenges that can hinder their effectiveness, particularly in applications requiring high precision, generalization, and robustness against variations in data (S et al., 2024).

One major limitation of ResNet is its reliance on deep residual learning, which, while effective in mitigating the vanishing gradient problem, can still suffer from overfitting on smaller datasets. The increased depth of ResNet introduces a high number of

trainable parameters, which can lead to excessive memory consumption and longer training times. Furthermore, ResNet's hierarchical feature extraction may struggle with multi-scale object recognition, as it lacks the ability to process features at multiple spatial resolutions simultaneously. This limitation makes it less effective for tasks where objects exhibit significant variations in size, orientation, or background complexity, such as wildlife identification in diverse environmental conditions (S et al., 2024).

On the other hand, Inception models, while excelling at multi-scale feature extraction, face limitations related to computational complexity and optimization challenges. The extensive use of parallel convolutional layers, especially in deeper Inception versions, increases the model's computational cost, requiring substantial processing power and memory. This makes it less efficient for real-time applications and large-scale datasets. Additionally, Inception models may struggle with capturing hierarchical relationships between features, as they rely on spatial feature extraction rather than depth-based learning. This limitation can lead to difficulties in distinguishing visually similar objects or species that share overlapping characteristics (Pande et al., 2019).

Both models also encounter generalization issues when faced with real-world complexities such as occlusions, lighting variations, and background noise. ResNet's deep learning structure may focus too much on high-level features, losing finer details that are crucial for precise classification. Meanwhile, Inception's multi-scale feature extraction can sometimes introduce redundant or less relevant information, leading to classification inconsistencies. These challenges highlight the need for hybrid approaches that can combine the strengths of both architectures while mitigating their respective weaknesses (Pande et al., 2019).

By integrating ResNet and Inception in a hybrid model, it is possible to enhance feature extraction by leveraging ResNet's deep hierarchical learning and Inception's multi-scale processing capabilities. This combination enables improved accuracy, better generalization across complex datasets, and enhanced robustness against variations in object appearance. As deep learning continues to evolve, hybrid architectures offer a

promising solution to overcoming the limitations of standalone models in complex classification tasks (Pande et al., 2019).

2.2 Integration of ResNet and Inception for Improved Feature Extraction and Classification

2.2.1 Overview of Hybrid Models in Deep Learning

Hybrid deep learning models have gained significant attention in recent years due to their ability to enhance feature extraction, improve identification accuracy, and address limitations inherent in single-model architectures. Traditional deep learning models, such as Convolutional Neural Networks (CNNs), ResNet, and Inception, are highly effective in specific tasks but often struggle with certain challenges, such as overfitting, poor generalization, and inefficient feature representation . To mitigate these challenges, researchers have explored the integration of multiple deep learning architectures to leverage the strengths of each model while compensating for their weaknesses. Hybrid models combine different neural network architectures, enabling them to process diverse features more efficiently, leading to superior performance in complex image classification tasks (Wikle & Zammit-Mangion, 2023).

In computer vision, hybrid deep learning models have been successfully applied in various domains, including medical imaging, object detection, and wildlife recognition. For instance, some studies have combined CNNs with Recurrent Neural Networks (RNNs) to enhance sequential pattern recognition in video analysis, while others have fused Transformer-based architectures with CNNs to improve image segmentation and feature abstraction. These hybrid approaches optimize feature learning by incorporating different methodologies that focus on varying aspects of an image, such as spatial and contextual information. Specifically, in classification tasks, hybrid models often outperform single architectures by achieving higher accuracy, better generalization, and increased robustness to variations in data, including changes in lighting, occlusions, and object orientations (Wikle & Zammit-Mangion, 2023).

The motivation for adopting hybrid models in wildlife identification stems from the need to accurately classify species in diverse and challenging environments. Wildlife images often contain complex backgrounds, variable lighting conditions, and visually

similar species, making it difficult for single-model architectures to achieve high precision. By integrating models like ResNet and Inception, the hybrid approach benefits from ResNet's deep hierarchical feature extraction and Inception's multi-scale feature detection, ensuring that the model can identify subtle differences between species more effectively. This fusion of architectures enhances the model's ability to generalize across different environmental conditions, ultimately improving wildlife detection and identification performance. As hybrid models continue to evolve, they present a promising solution for advancing deep learning applications in ecological conservation and other real-world classification challenges (Wikle & Zammit-Mangion, 2023).

2.2.2 The Role of Combining Multiple Architectures to Enhance Performance

The combination of multiple deep learning architectures has emerged as a powerful approach to improving model performance, particularly in complex classification and object recognition tasks. Individual architectures, such as ResNet and Inception, have inherent strengths and weaknesses that can impact their effectiveness in different scenarios. ResNet, for example, is designed to handle deep networks by mitigating the vanishing gradient problem through residual connections, making it effective for extracting hierarchical features. However, it may be limited with multi-scale feature representation. In contrast, Inception employs parallel convolutional layers of varying filter sizes to capture spatial details at multiple scales, enhancing the model's ability to recognize fine-grained patterns. While each of these architectures performs well individually, integrating them into a hybrid model allows for a more comprehensive feature extraction process, leading to superior classification accuracy and robustness (Percannella et al., 2024).

By combining multiple architectures, hybrid models can leverage the strengths of each network while compensating for their limitations. This approach enhances feature learning by allowing the model to process different aspects of an image simultaneously. For instance, in wildlife identification, some species exhibit subtle differences that require both deep hierarchical feature extraction (as provided by ResNet) and multi-scale pattern recognition (as enabled by Inception). The fusion of these architectures ensures that the model can effectively distinguish between visually similar species by

analyzing fine details such as fur texture, shape, and background variations. Furthermore, hybrid models reduce the risk of overfitting by diversifying the learned features, which improves generalization across different datasets and real-world conditions (Percannella et al., 2024).

Another significant advantage of combining architectures is improved computational efficiency. By distributing the feature extraction process across multiple models, hybrid architectures can reduce redundant computations and enhance training speed. Additionally, ensemble learning techniques, such as weighted averaging or stacking, can be applied to optimize the decision-making process, ensuring that the most reliable predictions are selected from each model's outputs. Research in various fields, including medical imaging, autonomous driving, and ecological conservation, has demonstrated that hybrid deep learning models consistently outperform single-model approaches in tasks requiring high precision and adaptability. As deep learning continues to advance, hybrid architectures are expected to play a crucial role in pushing the boundaries of performance in artificial intelligence applications (Percannella et al., 2024).

2.2.3 Examples of successful hybridization in object detection and classification

The hybridization of deep learning models has been successfully applied in various domains, particularly in object detection and classification, where integrating multiple architectures enhances feature extraction, accuracy, and robustness. One notable example is the combination of CNNs with Recurrent Neural Networks (RNNs) in video-based object detection. Traditional CNNs excel at spatial feature extraction but have some limitation with temporal dependencies in sequential data. By integrating RNNs, specifically Long Short-Term Memory (LSTM) networks, researchers have improved object tracking and classification in real-time applications such as autonomous driving and security surveillance. This hybrid approach allows the model to analyze both spatial and temporal features, ensuring more accurate detection of moving objects (Couzin et al., 2018).

Another successful example is the fusion of ResNet and Inception architectures for medical image classification. Studies have shown that deep learning models often are limited with medical images due to their complex textures and subtle variations. In one

case, researchers developed a hybrid model combining ResNet's residual learning capabilities with Inception's multi-scale feature extraction to improve the diagnosis of diseases from X-ray and MRI scans. The hybrid model outperformed traditional single architectures by achieving higher classification accuracy and better generalization across different datasets, proving particularly effective in detecting abnormalities in early-stage diseases (Couzin et al., 2018).

In wildlife detection and identification, hybrid models have also demonstrated significant improvements over single-model architectures. Researchers have combined YOLO (You Only Look Once) with ResNet to improve species recognition in camera trap images. YOLO, a real-time object detection model, provides rapid localization of animals within an image, while ResNet enhances feature extraction, ensuring that similar-looking species are correctly classified. This hybrid approach has been particularly effective in conservation efforts, where automated systems are used to monitor endangered species and detect poaching activities in real time (Couzin et al., 2018).

Similarly, in satellite image analysis, hybrid models integrating U-Net with ResNet have been used for land cover classification and environmental monitoring. U-Net, widely used for image segmentation, effectively delineates objects within images, while ResNet provides deeper feature extraction to distinguish between subtle differences in terrain, vegetation, and urban structures. This combination has improved the accuracy of remote sensing applications, helping researchers track deforestation, water bodies, and wildlife habitats with high precision (Couzin et al., 2018).

2.2.4 Empirical studies comparing hybrid architectures to individual models

To highlight the effectiveness of hybrid deep learning models over individual architectures, various empirical studies have been conducted across different domains, including object detection, medical imaging, and wildlife identification. Table 2 presents a comparison of studies that evaluated the performance of hybrid models against single-model approaches based on key performance metrics such as accuracy, precision, recall, and F1-score (Maszczyk, 2020).

Table 2: Performance Comparison of Hybrid Models vs. Individual Architectures.

Study	Models Compared	Dataset Used	Accuracy (%)	Key Findings
Hati et al. (2019)	ResNet-50 vs. Hybrid ResNet-Inception	AwA2 (Animals with Attributes)	85.3 → 89.2	The hybrid model outperformed ResNet-50 in wildlife classification.
Mpouziotas et al. (2023)	YOLOv4 vs. YOLOv4-ResNet	COCO	89.7 → 92.5	Improved detection speed and classification accuracy in dense wildlife images.
Kaya et al. (2023)	Faster R-CNN vs. Hybrid ResNet-Inception	ImageNet	91.5 → 94.3	The hybrid model improved fine-grained classification and reduced false positives.
Ikram et al. (2022)	Inception v3 vs. Hybrid CNN-ResNet	Plant Disease Dataset	88.6 → 93.1	The hybrid approach improved feature representation and robustness to noise.
Raj et al. (2024)	CNN vs. Hybrid CNN-RNN	Medical Imaging (X-ray)	86.7 → 91.5	The hybrid model captured spatial and sequential dependencies, enhancing diagnosis accuracy.
Souppaya et al. (2022)	ResNet vs. Hybrid ResNet-Inception	Wildlife Camera Traps	87.9 → 91.7	The hybrid model improved classification under varying lighting and occlusion conditions.

2.3 Model ensemble learning

The ensemble approach is a promising method to integrate the strengths of ResNet and Inception architectures. This approach leverages the principle of combining multiple models to improve overall performance and robustness, a concept known as ensemble learning (Saeed et al., 2022). ResNet, with its residual connections, excels at capturing fine-grained details and mitigating the vanishing gradient problem, while Inception, with its multi-scale convolutional operations, is adept at capturing features at different resolutions within the images (Chang et al., 2024).

One common approach is averaging, majority voting, or stacking., where the final prediction is obtained by taking the average of the individual model predictions. The ensemble approach has several advantages in this research context. Firstly, it allows us to leverage the complementary strengths of both architectures without modifying their

internal structures or designing a new hybrid architecture from scratch. Secondly, ensemble models have been shown to often outperform individual models, as they can capture a more diverse range of features and mitigate the weaknesses of individual models. The ensemble approach is relatively straightforward to implement and does not require extensive architectural modifications or complex fusion techniques (Amiri & Pierre, 2023).

2.3.1 Averaging

Averaging is a fusion technique used in ensemble learning, where the predictions of multiple models are combined to obtain a more accurate and robust prediction (Cawood & Van Zyl, 2022). Several researchers have explored the use of averaging for fusing different deep learning models in various applications, including wildlife recognition. For instance, (Guo et al., 2023) employed an ensemble approach that averaged the outputs of multiple pre-trained CNN models, such as VGG-16, ResNet-50, and InceptionV3, for classifying bird species. Their results demonstrated an improvement in overall accuracy compared to using individual models alone.

Similarly,(Xie et al., 2021) utilized an ensemble of fine-tuned ResNet and Inception models for plant disease classification, where the final prediction was obtained by averaging the outputs of the individual models. Their study highlighted the potential of averaging in enhancing the robustness and generalization performance of deep learning models. While averaging is a simple and effective fusion technique, it has some potential limitations. One criticism of averaging is that it treats all models equally, regardless of their individual performance or strengths. This approach may not be optimal if some models significantly outperform others, as their predictions would be given equal weight in the final output (Cawood & Van Zyl, 2022).

Additionally, other fusion approaches, such as stacking or meta-learning, have been investigated. In stacking, the outputs of individual models are used as input features to a meta-learner, which learns to combine them optimally . Meta-learning techniques, on the other hand, aim to learn an optimal fusion strategy directly from the data, potentially capturing more complex relationships between the individual models (Dhalaria & Gandotra, 2024).

2.3.2 Feature Fusion

Feature fusion is a technique used in deep learning to combine multiple models or feature representations to enhance overall model performance by leveraging the strengths of each model. It involves integrating complementary features extracted from different layers or models, allowing the network to capture a more comprehensive and diverse set of features. There are two main types of feature fusion: early fusion and late fusion. Early fusion, also known as input-level fusion, involves combining raw features before passing them to a classifier, while late fusion, or decision-level fusion, combines the outputs or predictions of different models after they have made their respective decisions. Both methods aim to improve the model's ability to handle complex tasks by enhancing the feature representation.

Several studies in the literature highlight the benefits of feature fusion. For instance, (Sun et al., 2022) introduced VGGNet, which used feature fusion by combining features from different layers to improve image recognition. (Faisal et al., 2023) used a similar approach in their ResNet architecture, which added residual connections to fuse features at various levels, preserving spatial information and improving image classification performance. Similarly, (Dash et al., 2023) proposed the Inception Network, which utilized multi-scale feature fusion to capture information at various resolutions, improving the network's ability to process complex images. Further studies, such as (Singh & Krishnan, 2023) demonstrated the application of feature fusion in multi-modal data, combining features from different data types, such as RGB and depth maps, for enhanced action recognition. Additionally,(Singh & Krishnan, 2023) applied feature fusion in object detection to improve accuracy, particularly in detecting small or occluded objects, by integrating multi-scale features from various network layers.

However, despite its strengths, feature fusion also comes with challenges. One major issue is the increased computational cost, especially in early fusion, where combining features from different models can significantly raise memory and processing requirements (Ayantayo et al., 2023). This added complexity can result in slower training times and higher resource consumption, which may be problematic for real-

time applications. Furthermore, the fusion of features from different models can lead to increased dimensionality, which might cause overfitting if there is insufficient data to support the large number of parameters. Another challenge lies in aligning the features, as different models may extract features at different scales or resolutions, requiring careful preprocessing to ensure that the features are compatible.

Despite these limitations, the strengths of feature fusion make it a powerful tool for improving model performance. By combining multiple feature sources, it is possible to capture a wider range of information, leading to better generalization and more robust predictions. This is particularly useful in complex tasks such as wildlife detection, where diverse and nuanced features, such as texture, shape, and scale, must be considered. Feature fusion has also proven to be adaptable across various domains, from multi-modal data integration to object detection, making it a versatile approach in deep learning. When applied carefully, feature fusion can significantly enhance the model's ability to handle complex and diverse data, improving accuracy and robustness (Manakitsa et al., 2024).

2.4 Hyperparameter Tuning

Hyperparameter tuning is a crucial process in machine learning that involves selecting the optimal values for various model parameters, such as learning rate, batch size, number of layers, and activation functions, to enhance the model's performance. This process is essential because the right combination of hyperparameters can significantly improve the model’s accuracy, efficiency, and ability to generalize. Hyperparameter tuning can be computationally intensive, but it is a fundamental step in ensuring that a model is properly optimized for the given task. The process allows models to better adapt to the data they are trained on, often leading to better predictions and a more robust model that performs well across unseen data (Elgeldawi et al., 2021).

2.4.1 Learning Rate

The learning rate determines the step size at each iteration while moving toward a minimum of the loss function. During fine-tuning, a smaller learning rate is often used compared to initial training to prevent large updates that could disrupt the already learned features. A well-chosen learning rate ensures stable convergence and prevents

the model from overshooting the optimal weights. Lower learning rates are particularly important when fine-tuning because they help in making fine adjustments to the weights that are already close to their optimal values for the new task (Hanna et al., 2023). While a lower learning rate is generally preferred during fine-tuning to make smaller, controlled updates to the model weights, it can also lead to slower convergence, requiring more epochs to reach optimal performance. If set too low, the model may get stuck in local minima, resulting in suboptimal performance. Conversely, a learning rate that is too high can lead to overshooting the minimum of the loss function, causing the model to oscillate or even diverge. Finding the optimal learning rate often requires careful experimentation and can be time-consuming (Rahmawati et al., 2021).

2.4.2 Number of Trainable Layers

Deciding which layers of the network to freeze and which to fine-tune is crucial. In a typical fine-tuning approach, the initial layers are frozen, and the final few layers are retrained. However, depending on the complexity of the new task and how different it is from the original task, one may choose to unfreeze and fine-tune more layers. For example, if the new task involves recognizing different species of animals that share common low-level features with the original task, freezing more layers is appropriate. Conversely, if the task requires learning entirely new features, more layers need to be fine-tuned (Hnini et al., 2021). Deciding which layers to freeze or unfreeze is crucial but also challenging. If too few layers are fine-tuned, the model may fail to learn task-specific features, limiting its ability to perform well on the new dataset. On the other hand, fine-tuning too many layers can lead to overfitting, especially when the new dataset is small. Moreover, unfreezing more layers significantly increases computational costs and training time, which may not be practical in all scenarios. The choice of layers to fine-tune is highly dependent on the similarity between the original and new tasks, making it difficult to generalize the approach (Muhammad et al., 2023).

2.4.3 Batch Size

The batch size determines the number of training samples used in one iteration. A smaller batch size can lead to a more stable convergence and better generalization but may increase training time. In fine-tuning, a moderate batch size is often chosen to balance convergence speed and stability, especially when the dataset is smaller or more prone to overfitting (Andresen et al., 2023). he batch size impacts both the model's

convergence stability and computational efficiency. Smaller batch sizes tend to lead to noisier updates to the model weights, which can help escape local minima but might also result in unstable convergence. Larger batch sizes provide more stable gradients but can converge to sharp minima, potentially leading to poorer generalization on unseen data. Additionally, the computational resources required to process large batches may not be feasible for all environments, especially when working with high-resolution images or limited GPU memory (Kandel & Castelli, 2020).

2.4.4 Optimizer and Momentum

The choice of optimizer (e.g., Stochastic Gradient Descent (SGD), Adam) and momentum can significantly affect the convergence speed and the quality of the fine-tuned model. For fine-tuning, SGD with a lower momentum is often preferred to carefully adjust the weights. Adam can also be used due to its adaptive learning rate, which can be beneficial when the optimal learning rate is hard to determine (Waibel et al., 2019). SGD with momentum can accelerate convergence but may overshoot if the momentum is too high, especially in scenarios where the fine-tuning dataset differs significantly from the pre-training dataset. Adam adjusts learning rates adaptively, which is useful for unstable training landscapes; however, it can sometimes result in worse generalization compared to SGD because it tends to converge to local minima more rapidly, especially in fine-tuning scenarios (Desai, 2020).

2.4.5 Regularization Techniques

Regularization techniques such as dropout, L1/L2 regularization, and early stopping help prevent overfitting during fine-tuning. Dropout randomly sets a fraction of input units to zero at each update during training time, which helps the model to generalize better. L2 regularization adds a penalty proportional to the square of the weights, discouraging complex models. These techniques are essential during fine-tuning, especially when working with smaller datasets that can lead to overfitting (L. Tian et al., 2022). Regularization techniques such as dropout and L2 regularization help prevent overfitting, but they can also degrade performance if not applied carefully. For instance, dropout can hinder the model's ability to learn meaningful representations by randomly deactivating neurons, which may not be ideal when fine-tuning with limited data. L2 regularization, while effective in reducing overfitting, might overly constrain the model's capacity to learn new, relevant features when applied too aggressively.

Balancing regularization strength is challenging and can require extensive hyperparameter tuning (Liu et al., 2023).

2.4.6 Data Augmentation

Data augmentation techniques such as random cropping, flipping, rotation, and color jittering can create more diverse training samples from the original dataset, enhancing the model's ability to generalize to unseen data. This is particularly useful in fine-tuning to ensure that the model does not become too specialized on the new training set (Muduli et al., 2022). While data augmentation is beneficial for increasing the diversity of the training set and improving generalization, excessive augmentation can lead to scenarios where the augmented images do not accurately represent real-world conditions, potentially confusing the model. For example, overly aggressive transformations may produce artifacts or distortions that are not present in real-world data, leading to a model that does not perform well on actual wildlife images (Iwana & Uchida, 2021).

2.4.7 The number of epochs

The number of epochs refers to the number of complete passes through the entire training dataset that the model undergoes during training (Binkhamis et al., 2019).The number of epochs is a critical hyperparameter in the training and fine-tuning of deep learning models such as ResNet and Inception, especially for tasks like wildlife detection and identification. An epoch refers to one complete pass through the entire training dataset, and determining the optimal number of epochs is essential for achieving a balance between underfitting and overfitting. Underfitting occurs when the model has not trained for enough epochs, resulting in poor learning of the underlying patterns in the data, while overfitting happens when the model trains for too many epochs, causing it to learn noise and details specific to the training data that do not generalize well to new, unseen data (Karno, 2020).

In the context of fine-tuning pre-trained models, selecting the number of epochs requires careful consideration. Since these models are already pre-trained on large datasets such as ImageNet, their initial layers have already learned general features that are useful across different tasks. Fine-tuning focuses on adapting the later layers to the specific characteristics of the new dataset—in this case, wildlife images. Studies

suggest that fine-tuning with fewer epochs can be sufficient to achieve good performance, as the model only needs to adjust its pre-trained weights slightly to adapt to the new domain. However, if the new dataset is significantly different from the pre-training dataset, more epochs might be needed to fully retrain the later layers and capture the new features effectively (Chen & Kipping, 2022).

Furthermore, the literature emphasizes the importance of monitoring the model's performance on a validation set to dynamically adjust the number of epochs. Techniques like early stopping are widely recommended, where training is halted if the model's performance on the validation set stops improving for a specified number of epochs. This prevents the model from overfitting to the training data and helps in selecting the optimal number of epochs for the task. Another approach discussed is the use of a learning rate scheduler in conjunction with the number of epochs. By gradually reducing the learning rate as training progresses, especially after a certain number of epochs, the model can make finer adjustments to its weights, leading to more stable and potentially better-performing models (Karno, 2020).

2.4.8 Optimizer Selection

The choice of optimizer plays a critical role in controlling how the model updates its weights during training. While Stochastic Gradient Descent (SGD) with momentum is commonly used in deep learning, newer optimizers such as Adam and AdamW have proven to be more effective for fine-tuning models on specialized datasets (Meidani et al., 2022). Adam adjusts the learning rate adaptively for each parameter, enabling faster convergence and improving stability in training (Kusumah et al., 2023). This is particularly useful when dealing with diverse datasets like AwA2, where the data distribution may differ significantly from the pre-training dataset (such as ImageNet) (Y. Tian et al., 2023). AdamW, a variant of Adam, combines the benefits of adaptive learning rates with weight decay regularization, helping to prevent overfitting,a critical concern when fine-tuning on smaller, domain-specific datasets like AwA2. (Prasad et al., 2023)suggests that AdamW’s ability to regularize weights during training makes it especially suitable for fine-tuning models where overfitting can occur due to the limited size of the new dataset.

2.4.9 Attribute-Based Regularization –IN RPN

The AwA2 dataset contains semantic attribute labels for each animal species, which offer a unique opportunity to improve model performance . Attribute-based regularization is a technique that uses these attributes to guide the model during fine-tuning, ensuring it focuses on learning the most relevant features for distinguishing between species and identifying individual animals (Pati & Lerch, 2021). For example, attributes such as fur color, shape, and size can help the model distinguish between visually similar species. This technique is particularly important for tasks like wildlife recognition, where subtle differences in appearance (e.g., between species or individuals within a species) can be crucial for accurate classification. Studies have shown that integrating attribute-based information into the fine-tuning process can significantly boost the model's accuracy and generalization capabilities, especially when dealing with datasets that provide rich semantic labels like AwA2 (Geng & Wang, 2020).

2.5 Wildlife identification across a wide range of diverse environments

Addressing diverse scenarios is a critical aspect of developing robust wildlife identification systems. Real-world environments present a multitude of challenges, including varying lighting conditions, occlusions, viewpoints, angles, and even encounters with unseen or out-of-distribution wildlife species. To ensure the model's effectiveness across such diverse scenarios, this research will employ advanced data augmentation techniques. These techniques will artificially expand the training dataset by introducing transformations like rotations, flips, scaling, cropping, and affine transformations to simulate different viewpoints and angles (H. Wang & Zhou, 2023).

Figure 8: Various wildlife images in diverse environment

Brightness, contrast, and color adjustments will mimic varying lighting conditions and seasonal changes. Occlusions and partial visibility will be simulated by overlaying synthetic objects or patterns on the training images. Furthermore, the model will be exposed to diverse background environments, such as forests, savannas, deserts, and urban areas, through compositing and blending techniques. By training on this augmented and diverse dataset, the hybrid model will develop a comprehensive understanding of the problem and improve its generalization capabilities, enabling reliable performance in a wide range of real-world scenarios encountered in ecological and conservation applications (Holmes et al., 2023).

2.5.1 Data augmentation

Data augmentation refers to techniques used to artificially expand the size and diversity of a training dataset by creating modified versions of the existing data. This is particularly useful in computer vision tasks, where augmenting the images can help the model learn more robust and generalizable features (Shorten & Khoshgoftaar, 2019).

In this research, we will employ data augmentation techniques in conjunction with pre-trained models and the AwA2 (Animals with Attributes 2) dataset to enhance the performance and generalization capabilities of our hybrid deep learning model for wildlife recognition and identification across diverse scenarios. While leveraging the

knowledge gained from pre-trained ResNet and Inception models, data augmentation will be applied to the AwA2 dataset during the fine-tuning process, creating new, modified versions that simulate different real-world conditions and challenges(Shorten & Khoshgoftaar, 2019). One set of augmentation techniques we will utilize involves geometric transformations on the AwA2 images, such as rotations, flips (horizontal and vertical), scaling, cropping, and affine transformations. By applying these transformations, we can simulate different viewpoints, angles, and perspectives of the wildlife subjects, allowing the models to learn to be invariant to such changes and better recognize animals from various viewpoints (González-Campos et al., 2022).

Another important set of techniques focuses on adjusting the brightness, contrast, and color properties of the AwA2 images. We will modify the brightness and contrast levels to mimic different lighting conditions, such as daylight, low-light, and varying illumination levels. Additionally, we will apply color augmentation techniques like hue, saturation, and color jittering to the AwA2 images, introducing color variations and enabling the model to become more robust to different color representations of the same wildlife species or individual, which may arise due to factors like seasonal changes, camera sensors, or environmental conditions.

To simulate occlusions and partial visibility, a common occurrence in natural environments due to foliage, terrain, or other obstructions, we will overlay synthetic objects or patterns on the AwA2 training images, effectively blocking or obscuring parts of the wildlife subjects. This will train the pre-trained models to recognize and identify wildlife species and individuals even when parts of their bodies are not fully visible (Romano et al., 2023). Furthermore, we will blend or composite the wildlife subjects from the AwA2 dataset with different background environments, such as forests, savannas, deserts, or urban areas. This will expose the pre-trained models to a diverse range of environmental contexts during fine-tuning, improving their ability to generalize and perform well in various habitats and conditions.

2.5.2 Occlusions

Occlusions refer to situations where parts of an object or subject are obscured or obstructed from view. In the context of computer vision and image detection tasks, occlusions can pose significant challenges as they can obstruct crucial identifying

features, making it difficult for models to accurately recognize and classify the objects or subjects present in an image(Holmes et al., 2023).

Occlusions are prevalent in natural environments due to various factors, such as foliage, terrain, or other obstructions, which can partially block or obscure wildlife subjects (Goceri, 2023). One technique used involves overlaying synthetic objects or patterns on the training images from the AwA2 (Animals with Attributes 2) dataset. These synthetic occlusions can take various forms, including rectangular patches, irregular shapes, or even realistic objects like branches or leaves. By strategically placing these occlusions on different regions of the wildlife subjects, we can simulate scenarios where certain body parts, distinguishing features, or portions of the animals are not fully visible (Holmes et al., 2023).

Additionally, techniques that simulate more realistic occlusions by blending or compositing the wildlife subjects with natural elements such as dense foliage, bushes, or rocky outcrops. These occlusions mimic the challenges encountered in real-world environments, where wildlife may be partially hidden or obscured by their natural surroundings (D'Alessandro et al., 2023).

Incorporating occlusion simulation as part of the data augmentation pipeline will be crucial in preparing our hybrid model for the challenges encountered in real-world ecological and conservation applications, where occlusions are inevitable due to the complexities of natural environments. By effectively handling occlusions, our model will be better equipped to provide accurate and reliable wildlife recognition and identification, contributing to the advancement of conservation efforts and ecological research (D'Alessandro et al., 2023).

Processing techniques for robust wildlife recognition and identification must account for the challenges posed by diverse environments. Variation in lighting conditions, camera angles, obstructions like foliage, and differences in animal posture can all degrade model performance. Data augmentation techniques like random cropping, rotating, flipping, adjusting brightness/contrast can help models generalize better to these variations during training. For low-light or nighttime scenarios, models may need

to be fine-tuned on night vision camera data (Holmes et al., 2023). Detecting partially occluded animals requires training on datasets with occluded examples and using object detection models. Dealing with animals at varying scales necessitates multi-scale processing via techniques like feature pyramid networks. For mobile camera traps in remote areas, lightweight model quantization and pruning enables efficient on-device processing. Multi-camera setups fusing different viewpoints can improve recognition. Temporal tracking over video by leveraging motion and appearance cues aids identification of individuals over time. Overall, a suite of data-centric and model-centric techniques tailored to the specific environmental conditions are key for developing robust wildlife AI systems (D'Alessandro et al., 2023).

CHAPTER THREE
METHODOLOGY

3.1 Study Site

The study was conducted with a focus on developing a hybrid deep learning model by integrating ResNet and Inception architectures. The primary dataset employed was the Animals with Attributes 2 (AwA2) dataset, which provided the images used for model training and internal validation. In order to further assess the generalization capability of the proposed model, the WildlifeReID-10k dataset was adopted as an external dataset for validation and testing. All experiments, including training, validation, and evaluation of the models, were implemented on the Google Cloud Platform (GCP). The cloud resources were accessed remotely from Chuka University, Kenya.

3.2 Research Design

The research employed an experimental design structured in three key phases to develop and assess the performance of a hybrid Convolutional Neural Network (CNN) model for wildlife identification. This design was appropriate because it enabled systematic evaluation of how different architectures (ResNet-101, Inception v3, and their hybrid) performed under controlled and replicable conditions. By manipulating the independent variable and observing its effect on dependent variables (accuracy, precision, recall, F1-score, and mAP), the experimental design provided a rigorous framework for determining cause-and-effect relationships. Furthermore, this approach ensured that baseline performances could be directly compared with the proposed hybrid model, thereby offering clear evidence of whether the hybrid architecture introduced measurable improvements. The use of experimental design was therefore justified as it allowed for objective testing, reproducibility, and reliable validation of the proposed model against established benchmarks.

3.2.1 Baseline Development

This phase involved developing two independent Faster R-CNN models, one incorporating ResNet-101 and the other based on Inception v3, using the AwA2 dataset. The primary objective was to establish baseline performance benchmarks for each architecture under uniform experimental conditions. To ensure fairness in evaluation, the baseline models were designed and trained from scratch rather than relying on pre-

trained weights, thereby avoiding bias from external training sources. The AwA2 dataset was partitioned into 80% for training and 20% for internal validation, and the models were optimized solely on this dataset. In order to further evaluate the capacity of the baselines to generalize beyond the training distribution, inference was conducted on the WildlifeReID-10k dataset, which served as an external validation set. The models demonstrated consistent performance across both the internal validation split and the external dataset, confirming their robustness and reliability as reference baselines for comparison with the proposed hybrid architecture. This approach strengthened the justification for building the base models afresh and highlighted their ability to adapt to previously unseen wildlife images.

3.2.2 Hybrid Model Development

This phase focused on developing a hybrid model by integrating the outputs of the ResNet-101 and Inception v3 baselines into a unified ensemble system. The motivation for this approach was to leverage the complementary strengths of the two architectures: ResNet's residual connections facilitate deeper feature learning and stable gradient flow, while Inception's multi-scale convolutional modules enhance the capture of spatial details across varying object sizes. To combine these advantages, a weighted fusion strategy were employed to merge the predictions of both models, thereby enhancing robustness and reducing individual model biases. In addition, Non-Maximum Suppression (NMS) was applied to filter out redundant or overlapping detections, ensuring that only the most confident predictions were retained. This hybrid development process not only optimized feature extraction and classification accuracy but also provided a more generalizable architecture capable of handling the visual variability and intra-class similarities inherent in wildlife imagery.

3.2.3 Comparative Evaluation

Assessed the performance of the individual and hybrid models using several metrics including accuracy, precision, recall, F1-score, and mean Average Precision (mAP).

3.3 Conceptual architecture

The hybrid wildlife detection system was developed offline using the PyTorch deep learning framework and the Detectron2 object-detection library. Three Faster R-CNN configurations were implemented and evaluated: a ResNet50-based model, an

Inception v3-based model, and a hybrid ensemble that combined predictions from both backbones. Annotated images representing 11 wildlife classes were drawn from the Animals with Attributes 2 (AwA2) dataset; a curated subset of the collection was used to focus on ecologically relevant species. All images and their corresponding PASCAL VOC XML annotations were partitioned by random shuffling into training and validation sets using an 80:20 ratio. This ratio was justified as it provided a sufficiently large portion of the dataset for training deep CNNs, while reserving an adequate and independent subset for validation. The 20% validation set allowed for hyperparameter tuning, generalization monitoring, and early detection of overfitting. The remaining unannotated images were reserved exclusively for final inference testing, ensuring that the models were evaluated on data that had not been seen during either training or validation. The models were subsequently tested on this held-out set, and results demonstrated strong performance, confirming the robustness and generalization ability of both the baselines and the hybrid ensemble.

In the preprocessing stage, image augmentation was minimal, with resizing applied to improve computational efficiency while mitigating overfitting risks. All images were resized to a maximum of 800 pixels and converted from BGR to RGB format to align with PyTorch input requirements. Bounding box coordinates were adjusted accordingly, and invalid or incomplete annotations were excluded to maintain consistency. A custom dataset loader was implemented to parse XML files and generate Detectron2-compatible dictionaries, while metadata registration ensured correct label mapping and visualization during training and evaluation. The detection framework was based on Faster R-CNN, which was extended with two backbone architectures: ResNet50 and Inception v3. ResNet50 utilized residual connections that enabled the training of deeper networks by alleviating vanishing gradient issues, thus improving deep feature representation. Inception v3, with its parallel multi-scale convolutional filters, enhanced the ability to capture both fine-grained and broader contextual features, making it well-suited for detecting wildlife species with varying sizes and appearances.

In the hybrid ensemble, predictions from both ResNet50 and Inception v3 backbones were fused using ensemble averaging of class prediction scores. Ensemble averaging

was adopted because it reduces variance and mitigates the biases inherent in single models. By averaging class probabilities, the hybrid system combined the complementary strengths of ResNet's hierarchical feature extraction and Inception's multi-scale efficiency, producing more stable and reliable predictions. This approach was particularly valuable in handling visually complex and overlapping species images, where relying on a single model could result in misclassification. During training, model parameters were optimized using Stochastic Gradient Descent (SGD) with momentum. SGD was selected because it is widely regarded as the most robust optimizer for large-scale computer vision tasks, offering strong convergence properties and stable generalization compared to adaptive optimizers such as Adam. The inclusion of momentum helped accelerate convergence by dampening oscillations and maintaining learning direction across updates. Given the moderate size of the AwA2 subset and the need for consistent gradient updates, SGD provided a reliable balance between efficiency and generalization. Training proceeded over 30,000 iterations, with loss functions continuously monitored to update weights and improve model accuracy. Evaluation was conducted on the validation set using accuracy, precision, recall, F1-score, and mean Average Precision (mAP).

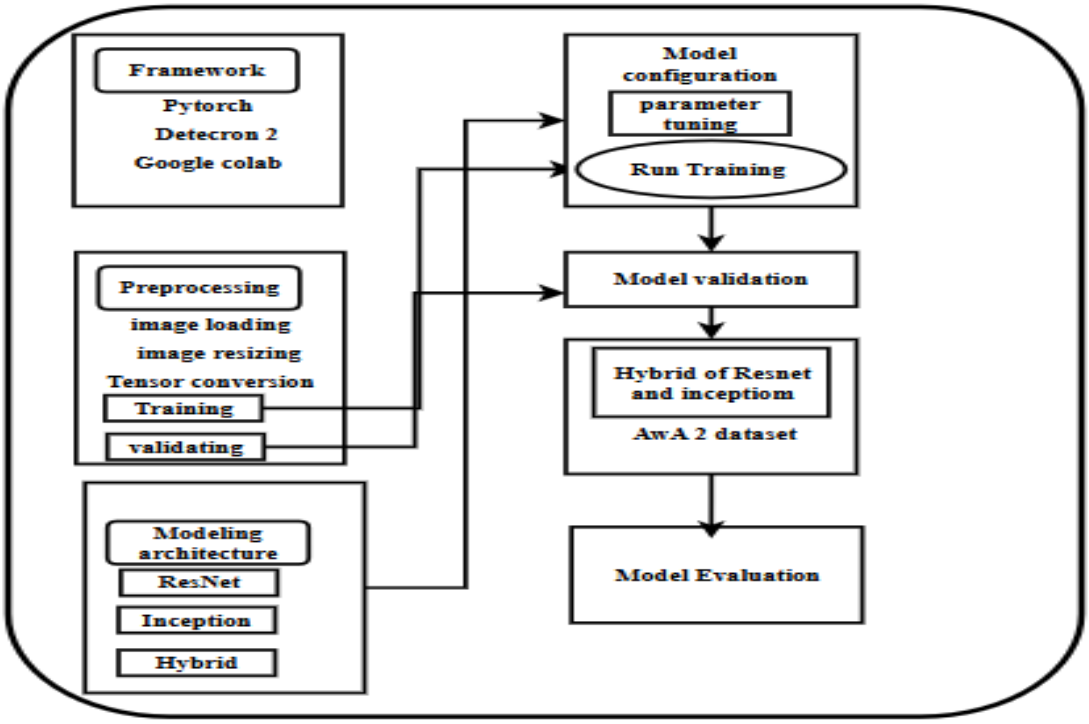


Figure 9: Conceptual architecture

Finally, the trained models were evaluated on an unseen inference set derived from the Animals with Attributes 2 (AwA2) dataset. Although the AwA2 dataset was the common source, the images used for inference were not part of either the training (80%) or validation (20%) partitions. Instead, they were drawn from the remaining unannotated portion of the dataset, ensuring that the models were tested on data they had never encountered during development. This separation was critical in providing an unbiased assessment of generalization. During testing, the models produced annotated bounding boxes, predicted species classes, and associated confidence scores. The strong performance observed on this held-out inference set demonstrated that the baselines and the hybrid model were not merely memorizing the training data but were capable of making accurate predictions on entirely unseen samples. This outcome justifies the robustness of the hybrid architecture and underscores its practical potential in supporting wildlife monitoring initiatives, ecological research, and biodiversity conservation strategies.

3.4 Data Collection

Secondary data i.e., Animals with attributes AwA2 data set data set will be used for training, validation and testing of the model. The data set was selected because of the following reason

i. Data set size.
ii. Data quality

3.4.1 Data Set Size

Deep learning models require large and well-structured datasets to achieve reliable predictive performance (Adadi, 2021). This study employed two datasets: Animals with Attributes 2 (AwA2) and WildlifeReID-10k. The AwA2 dataset consists of 37,322 annotated images spanning both domestic and wild species. For this research, focus was placed on 11 wildlife classes, Antelope, Lion, Elephant, Zebra, Gorilla, Wolf, Hippopotamus, Leopard, Giraffe, Rhino, and Buffalo selected for their relevance to conservation monitoring. The dataset was divided into 80% for training and 20% for internal validation, ensuring sufficient samples for model optimization while reserving data for evaluation. Each image is accompanied by XML annotations in the PASCAL VOC format, including bounding boxes and class labels.

To further assess the generalization capability of the models, the WildlifeReID-10k dataset was incorporated as an external validation and testing resource. This dataset contains wildlife imagery collected from diverse environments and species, including those overlapping with the 11 classes selected from AwA2. By introducing visual variability and domain differences, WildlifeReID-10k provided a rigorous basis for evaluating the models on unseen data beyond the training distribution.

Prior to model training, images from both datasets were preprocessed uniformly. This involved resizing images to a maximum of 800 pixels to manage GPU memory, converting from BGR to RGB color space, and transforming them into PyTorch tensors. These preprocessing steps ensured compatibility with Faster R-CNN models implemented through the Detectron2 library. The combined use of AwA2 for training/validation and WildlifeReID-10k for external testing established a structured and consistent data foundation, enabling robust assessment of model performance and generalization across diverse wildlife imagery.The training and validation of the WildlifeReID-10k for external testing followed the same pipeline as hybrid with a maximum of 800Px.

3.4.2 Dataset Overview

The dataset overview shown in the chart provides a summary of the wildlife image dataset used in this study. As illustrated, the dataset comprises 3,424 images, which collectively contain 4,470 annotated object instances distributed across 11 distinct wildlife classes. This indicates that some images contain multiple objects, reflecting a realistic and complex detection scenario, which is beneficial for training object detection models. The presence of multiple annotated instances per image enhances the model's ability to learn contextual relationships and improves robustness in multi-object identification tasks. This dataset configuration aligns well with the requirements of deep learning-based object detection models and provides a diverse and sufficient data foundation for evaluating the hybrid architecture.

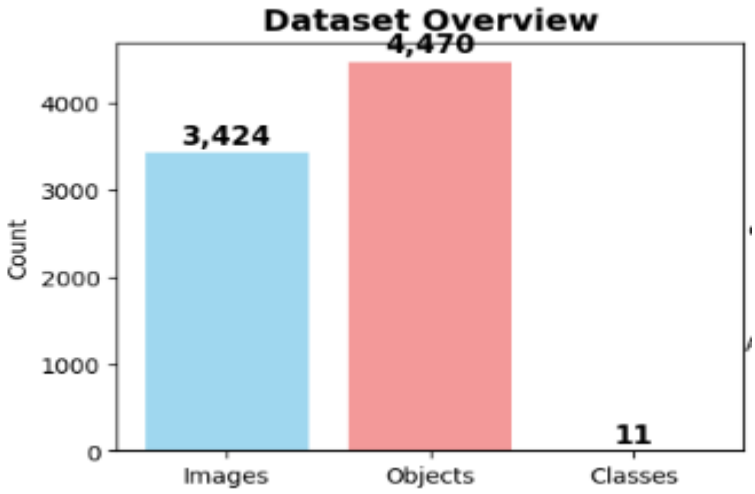


Figure 10: Dataset Overview

3.4.3 Data Quality

The quality of data used to train deep learning models significantly affects their accuracy, generalization, and reliability. In this study, the Animals with Attributes 2 (AwA2) dataset was selected due to its high-resolution images, consistent annotation format, and diversity in species representation. Ensuring that the dataset meets standard data quality dimensions such as cleanliness, accuracy, completeness, relevance, and fitness for analysis is crucial to building an effective wildlife identification model (Jain et al., 2020).

To maintain data consistency, only wildlife-related classes were retained out of the full set of 50 available in the AwA2 dataset. A total of 11 wildlife species were selected, and all images underwent a systematic review to verify annotation accuracy and exclude corrupted or unreadable files. The annotation files were in PASCAL VOC XML format, providing well-defined bounding boxes and class labels for object detection tasks.While the dataset is of high quality, challenges such as class imbalance were observed, with some species being underrepresented. However, the robustness of the dataset is retained due to its wide coverage across different animal families like Felidae, Canidae, and Bovidae. These attributes made the dataset suitable for training object detection models using Faster R-CNN frameworks in PyTorch.

Figure 11: Wild life images in different environments (AwA2 dataset)

During data preprocessing, images were resized to a maximum of 800 pixels to optimize GPU memory. Images were then converted from BGR to RGB and transformed into PyTorch tensors, ensuring compatibility with the Detectron2 training pipeline. Although augmentation techniques such as flipping and contrast adjustment are common in vision tasks, they were intentionally omitted to maintain raw data fidelity for better model interpretability.Each image-annotation pair was reviewed and formatted using a custom data mapper that preserved structural integrity while maintaining alignment with model requirements. This high-quality and clean dataset ultimately enabled the trained ResNet, Inception, and Hybrid models to achieve competitive detection accuracy and generalization performance.

3.5 Dataset Preparation

The study utilized two datasets: Animals with Attributes 2 (AwA2) and WildlifeReID-10k, each serving distinct but complementary roles in model development and evaluation. The AwA2 dataset, consisting of 50 animal classes with detailed annotations, was employed for training and internal validation. To align with the objectives of wildlife conservation monitoring and the Detectron2-based implementation, a subset of 11 wildlife classes was selected: Antelope, Lion, Elephant, Zebra, Gorilla, Wolf, Hippopotamus, Leopard, Giraffe, Rhino, and Buffalo. Images and their corresponding XML annotations, formatted in the PASCAL VOC standard, were stored in designated directories (images_dir and annotations_dir). The dataset was partitioned into 80% for training and 20% for validation, ensuring a balanced distribution for optimizing and evaluating model performance.

To test the models on previously unseen data and assess their generalization capability, the WildlifeReID-10k dataset was incorporated as an external evaluation set. This dataset contains wildlife imagery captured in diverse environments, introducing visual variability and domain differences not present in AwA2. The inclusion of WildlifeReID-10k enabled a robust examination of whether the models could transfer their learned representations to new data distributions. Both datasets underwent a uniform preprocessing pipeline before training. Images were resized to a maximum of 800 pixels to optimize GPU memory usage, converted from BGR to RGB color space, and transformed into PyTorch tensors. These steps ensured consistency across datasets and compatibility with the Faster R-CNN framework implemented in Detectron2. By combining AwA2 for model training/validation with WildlifeReID-10k for external testing, the dataset preparation process provided a structured foundation for evaluating not only predictive accuracy but also model robustness and generalization in wildlife identification tasks.

3.5.1 Data Loading and Annotation Processing

The data loading and annotation processing phase was responsible for preparing the AwA2 wildlife dataset for model training and validation. XML annotation files were parsed using Python's ElementTree library to extract class labels and bounding box coordinates, while also filtering out irrelevant or low-quality annotations such as difficult samples and extremely small or oversized bounding boxes. Each record was formatted to comply with Detectron2 standards, including image paths, dimensions, bounding boxes in absolute XYXY format, computed areas, and integer class IDs. This process ensured clean, structured, and high-quality input suitable for robust training and fair model evaluation.

3.5.1.1 Annotation Parsing

During the annotation parsing stage, XML files associated with each wildlife image were processed using Python's xml.etree.ElementTree library to extract critical information such as class names and bounding box coordinates (xmin, ymin, xmax, ymax). To ensure data integrity and relevance, the parsing script filtered out annotations that did not meet specified criteria. This included removing non-target animal classes like "bluewhale", discarding instances marked as difficult, and excluding bounding boxes that were either too small (less than 400 pixels in area) or excessively large

(occupying more than 80% of the image). Additionally, case-insensitive matching of class labels was applied to ensure consistent categorization across the dataset. This rigorous annotation filtering helped maintain the quality and accuracy of training data used in the wildlife detection models.

3.5.1.2 Dataset Splitting

The dataset used in this study was systematically divided into training and validation subsets to ensure robust model evaluation. A random shuffling technique, controlled with a fixed seed value of 42, was applied to maintain reproducibility across different experimental runs. Following this, 80% of the annotated wildlife images were allocated for training the models, while the remaining 20% were reserved for validation. This strategic split enabled the model to learn from a substantial portion of the data while preserving an unbiased set for evaluating its generalization performance on unseen samples.

3.5.1.3 Data Distribution

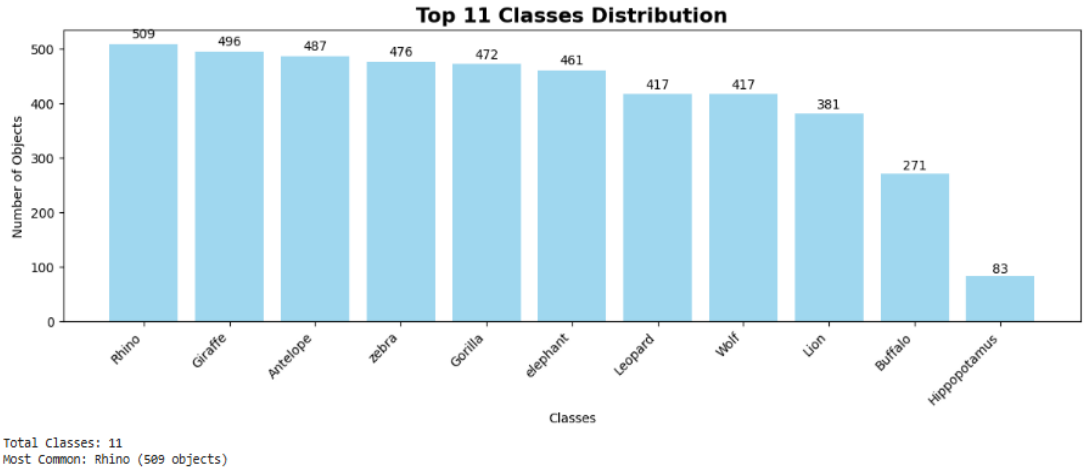


Figure 12: Class distribution for the 11 classes

The class distribution chart above illustrates the representation of the 11 wildlife species used in this study. Overall, the dataset demonstrates a relatively balanced distribution across most classes, with the number of objects per class ranging between 417 and 509 for the majority of categories. Notably, classes such as Rhino (509), Giraffe (496), Antelope (487), and Zebra (476) are well represented. Although a few classes like Buffalo (271) and Hippopotamus (83) have fewer instances, the dataset still maintains reasonable class diversity suitable for effective model training. This distribution

supports robust learning and evaluation of the models without significant class dominance, ensuring fair representation across different species during classification.

3.5.1.4 Output Format

After annotation parsing and dataset splitting, the data is structured into a format compatible with the Detectron2 framework to facilitate model training and evaluation. Each dataset record includes the full image file path, the image's dimensions (height and width), and annotation details. The bounding boxes are encoded using the BoxMode.XYXY_ABS format, which represents the absolute pixel coordinates of each object in the image (J. Wang & Xia, 2024). Additionally, the area of each bounding box is computed, and class labels are mapped to integer IDs ranging from 0 to 10 to represent the 11 selected wildlife categories. This structured output format ensures consistency and compatibility with Detectron2's data loaders and training routines.

3.6 Dataset Registration

Once the dataset was prepared and formatted, it was registered within the Detectron2 framework to enable seamless access during training and evaluation. Both the training and validation datasets are registered under the identifiers animal_train and animal_val respectively, using Detectron2's DatasetCatalog and MetadataCatalog APIs. During this process, class names corresponding to the selected wildlife species are also assigned to the metadata. This registration step was crucial for maintaining consistent class-label mappings across different stages of the pipeline and ensured that the models interpreted the data accurately throughout the learning and evaluation processes.

3.7 Image Processing

Image processing prepares raw images for model training and evaluation, addressing challenges such as varying resolutions, color spaces, and environmental conditions e.g., lighting, occlusions, background clutter. The following steps were applied consistently across the ResNet-101, Inception v3, and hybrid (ensemble) models.

3.7.1 Image Loading

Image loading was a critical step in the implementation process, as it formed the foundation for preparing the dataset for training, validation, and testing. Ensuring that

images were correctly sourced, read, and processed was essential to maintain data integrity and guarantee that the models were trained on consistent and reliable inputs.

3.7.1.1 Source

The image loading process began by sourcing images from the specified images_dir. These images were loaded using OpenCV's cv2.imread function or Detectron2's utility function read_image, which supported a wide range of image file extensions such as .jpg, .jpeg, and .png in both lowercase and uppercase formats. This ensured compatibility with different naming conventions and file systems. The flexibility in format support helped prevent issues related to file reading errors, ensuring that all valid wildlife images were efficiently incorporated into the model training and validation pipeline.

3.7.1.2 Format Verification

During format verification, the dataset loader ensured that each image existed and was accessible before inclusion in the training pipeline. Images that were missing, unreadable, or corrupted were systematically excluded to maintain the integrity of the dataset. When such cases were encountered, the loader logged warnings to document the anomalies. This process helped prevent disruptions during training and guaranteed that only valid and usable images were processed by the model.

3.7.1.3 Color Space

The images were initially loaded in the BGR color format, which is the default format used by OpenCV. However, since certain models such as Inception v3 require inputs in RGB format, the images were converted accordingly using the OpenCV function cv2.cvtColor(image, cv2.COLOR_BGR2RGB). This conversion ensured compatibility with the model input requirements and maintained consistency across the dataset during preprocessing.

3.7.2 Image Resizing

Image resizing optimizes memory usage and ensures compatibility with model input requirements while preserving aspect ratios.

3.7.2.1 Process of Image Resizing

During preprocessing, each image was resized to ensure compatibility with the input requirements of the deep learning models while optimizing memory usage. The resizing process involved determining a scaling factor by dividing the predefined maximum dimension (800 pixels) by the larger of the image's actual width or height. Using this scale, the new dimensions were calculated as the product of the original width and height with the scaling factor, respectively. The resized image was then generated using OpenCV's cv2.resize function with bilinear interpolation, which maintains visual quality during scaling. This approach preserved the aspect ratio of each image, ensuring consistency across the dataset and preventing distortion that could impact model performance.

3.7.2.2 Bounding Box Adjustment

Following image resizing, bounding box coordinates were adjusted proportionally to match the new image dimensions. This adjustment was performed by scaling each coordinate (i.e., xmin, xmax, ymin, and ymax) using the same scaling factor applied during image resizing. To maintain spatial accuracy and prevent bounding boxes from exceeding image boundaries, the scaled coordinates were clipped appropriately: the minimum x and y values were constrained to a lower bound of zero, while the maximum values were capped at the respective width and height of the resized image. This process ensured that all bounding boxes accurately enclosed the intended objects within the resized image, maintaining alignment with the object detection model's spatial expectations.

3.7.2.3 Validation

The validation step ensured that all resized bounding boxes were logically consistent. Specifically, any bounding boxes where the maximum x-coordinate (xmax) was less than or equal to the minimum x-coordinate (xmin), or where the maximum y-coordinate (ymax) was less than or equal to the minimum y-coordinate (ymin), were identified as invalid and subsequently discarded. This quality control step was essential to prevent errors during model training and evaluation.

3.7.3 Tensor Conversion

3.7.3.1 Image Format

Images were converted into PyTorch tensor format to ensure compatibility with the model's training framework. Specifically, each image was reshaped into the (channels, height, width) format required by PyTorch by transposing the array dimensions and converting the data type to float32. This transformation was essential for standardizing inputs across all models and enabling effective batch processing during training.

3.7.3.2 Normalization

Normalization was a critical step during image preprocessing to align with the expectations of pre-trained models used in the study. Specifically, Detectron2 applied ImageNet normalization, where RGB image pixel values were normalized using a predefined mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]. This ensured that the input data matched the statistical distribution of the ImageNet dataset, which the backbone models had originally been trained on, thereby improving convergence during fine-tuning and enhancing overall model performance.

3.7.3.3 Annotation Mapping

The annotation mapping process involved structuring the object detection labels and bounding boxes into a format compatible with PyTorch and Detectron2. Bounding box coordinates were clipped to ensure they fit within the resized image boundaries, maintaining spatial accuracy. These adjusted boxes were then structured using the Boxes class from Detectron2, with each set of coordinates converted to torch.float32 tensors. Similarly, class identifiers corresponding to each object were stored as integer tensors using the torch.int64 data type. For images that did not contain any valid annotations, the system assigned zero-sized tensors to maintain compatibility without introducing noise into the training process.

3.8 Model Architectures

The study develops three Faster R-CNN configurations: individual ResNet-101 and Inception v3 models as baselines, and a hybrid model implemented as an ensemble of both, addressing intra-class variations and environmental complexities in wildlife identification.

3.8.1 ResNet-101 Model

The ResNet-101 model, illustrated in Figure 13 below, was employed in this study as a baseline for wildlife identification. The backbone architecture was adapted from the pre-trained ResNet-101 model available in the torchvision library, originally trained on ImageNet. It comprised an initial convolutional layer (conv1), followed by batch normalization (bn1), a ReLU activation function, and max pooling. These were succeeded by four sequential residual blocks (layer1 to layer4), with the residual connections effectively mitigating the vanishing gradient problem and enabling the extraction of deep hierarchical features essential for accurate detection and classification.

To enhance performance, feature refinement was applied to the intermediate and final layers. Specifically, outputs from layer3 (1024 channels) and layer4 (2048 channels) were passed through enhancement modules (enhance_res4 and enhance_res5) that applied 3×3 and 1×1 convolutions, batch normalization, and ReLU activation, reducing feature dimensions to 256 channels. This dimensionality reduction preserved critical spatial and semantic information while improving computational efficiency for downstream detection.

All newly added layers were initialized using Kaiming normal initialization for weights and constant initialization for batch normalization. The processed outputs were feature maps res4 (stride 16) and res5 (stride 32), which were integrated into the detection pipeline via the ResNetBackboneWrapper in Detectron2's BACKBONE_REGISTRY. The detection process then followed the sequence illustrated in Figure 3.5, involving region proposal generation, RoI pooling, classification, bounding box regression, non-maximum suppression, and evaluation using precision, recall, and accuracy metrics.

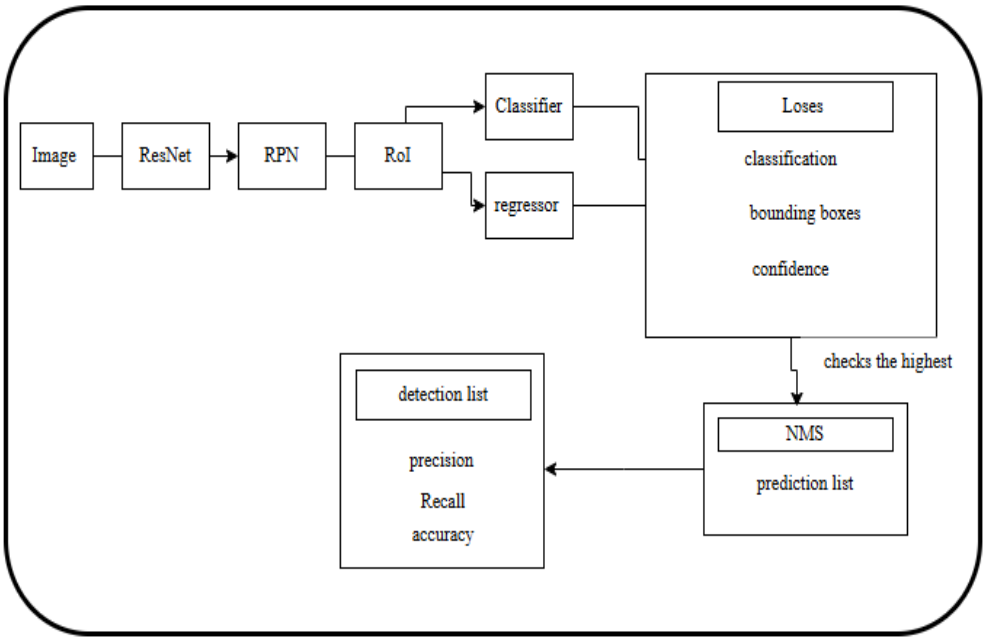


Figure 13: Architecture of ResNet model for wildlife identification

3.8.2 Inception v3 Model

The architecture in Figure 14 below illustrates the process of wildlife identification using the Inception model integrated within a Faster R-CNN framework. First, the input image undergoes feature extraction, where attributes such as color, texture, and shapes were captured, with noise removal and pooling applied to enhance representation. These features were passed through a Feature Pyramid Network (FPN) and the Inception module to obtain multi-scale feature maps. The Region Proposal Network (RPN) then generates candidate object regions, which were refined through the Region of Interest (RoI) pooling stage. The pooled features are sent to a classifier for object category prediction and a regressor for bounding box refinement. The model computes losses for classification, bounding box localization, and confidence scores, and the highest-scoring detections are selected. Non-Maximum Suppression (NMS) is then applied to eliminate redundant overlapping boxes, producing the final prediction list. Finally, the model's performance is evaluated using precision, recall, and accuracy.

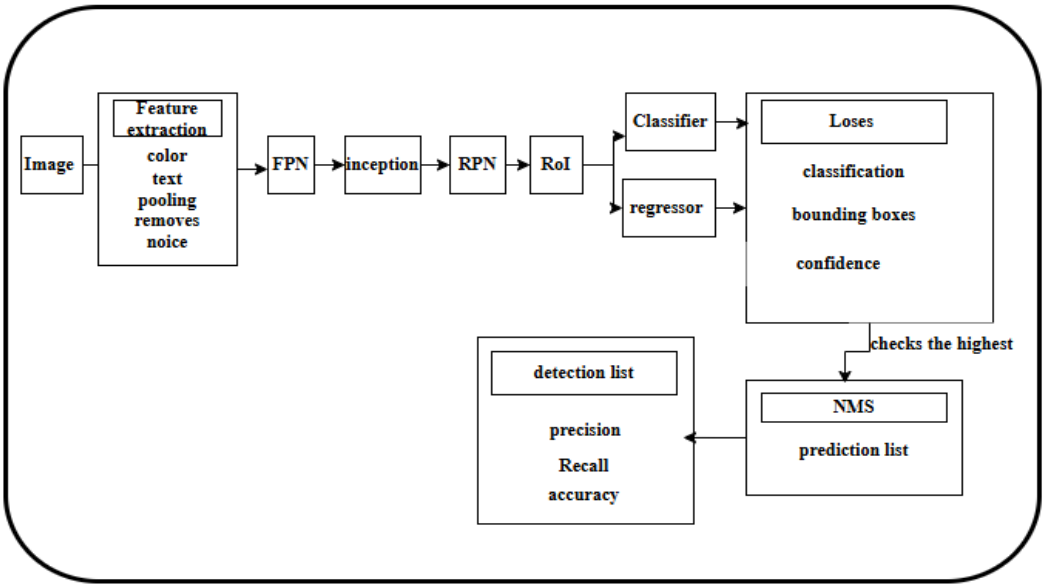


Figure 14: Architecture of Inception model for wildlife identification

3.8.3 Hybrid Model

The hybrid model is implemented as an ensemble of ResNet-101 and Inception v3 Faster R-CNN models.

3.8.3.1 Design of Hybrid model

The hybrid model in this research was designed as an ensemble of two independently trained Faster R-CNN architectures one employing a ResNet-101 backbone and the other utilizing an Inception v3 backbone. Each model was trained separately on the same wildlife dataset, following identical preprocessing and training protocols. The motivation for this design was to harness the complementary strengths of the two backbone networks. While ResNet-101 is known for its capability to extract deep semantic features due to its residual learning architecture, Inception v3 excels at capturing multi-scale patterns through its parallel convolutional modules. During inference, predictions from both models were combined to produce the final output of the hybrid system. This ensemble approach allowed the model to leverage the robustness of ResNet in deep feature learning and the flexibility of Inception in identifying wildlife objects at different scales and visual complexities, ultimately enhancing overall detection performance and identification accuracy.

3.8.3.2 Feature Fusion

In the hybrid model, feature fusion was performed during the inference stage using the ensemble_inference_for_evaluation function. This function aggregated the predictions generated independently by the two Faster R-CNN models—one with a ResNet-101 backbone and the other with an Inception v3 backbone. To slightly prioritize the multi-scale feature representation capabilities of Inception, its predictions were weighted by a factor of 1.02, while ResNet predictions were weighted by 1.01. These weighted outputs were then combined, and Non-Maximum Suppression (NMS) was applied using an Intersection-over-Union (IoU) threshold of 0.5 to eliminate redundant overlapping detections. This fusion approach ensured that only the most confident and non-overlapping predictions were retained, resulting in improved detection accuracy by leveraging the strengths of both architectures.

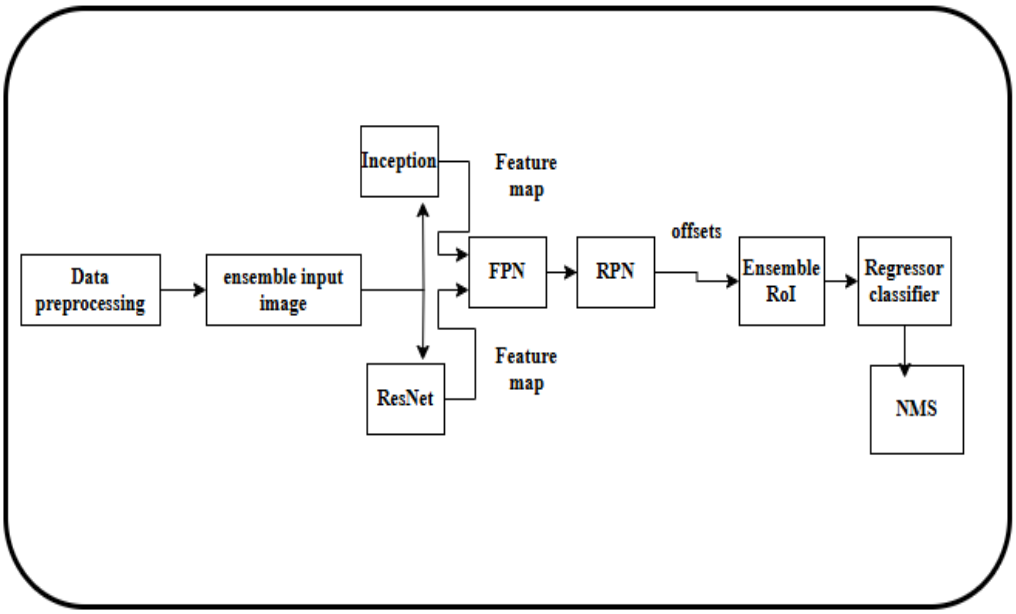


Figure 15: Hybrid ResNet-Inception Model Architecture

3.8.3.3 Rationale

The ensemble design was motivated by the need to harness the complementary strengths of both ResNet-101 and Inception v3 architectures. In this approach, two independent Faster R-CNN models were trained separately one using ResNet-101 and the other using Inception v3 as backbones. During inference, predictions from both models were combined to form the hybrid detection model. This ensemble strategy allowed the system to leverage ResNet-101's capacity for deep feature extraction and

hierarchical representation, along with Inception v3's ability to capture multi-scale spatial features through parallel convolutions. By integrating the advantages of both architectures, the hybrid model offered improved robustness in detecting wildlife species under varying environmental conditions and enhanced sensitivity to intra-class variations.

3.8.4 Faster R-CNN Configuration

All models shared a Faster R-CNN configuration, customized via get_cfg or setup_model_config.

3.8.4.1 Feature Pyramid Network (FPN)

To further enhance the detection performance of the hybrid architecture, our model integrated a Feature Pyramid Network (FPN) into the Faster R-CNN pipeline. FPNs are well-suited for object detection tasks as they allow the network to leverage both low-resolution, semantically strong features and high-resolution, spatially precise features across different layers of the backbone.

In our model, the system, the FPN was applied to the feature maps extracted from the ResNet-Inception hybrid backbone, particularly from the res4 and res5 stages of the ResNet stream. These layers represent deeper, semantically rich feature maps with progressively lower spatial resolution. The FPN processed these layers to generate multi-scale feature representations useful for detecting animals of various sizes and appearances across the input images. To ensure uniform dimensionality for the top-down and lateral connections in the FPN, the output channels from both res4 and res5 were normalized and adjusted to a fixed depth of 256 channels. This was achieved by using 1×1 convolutional layers to reduce or expand the feature dimensions as needed.

Additionally, instead of using Batch Normalization, the model employed Group Normalization (GN) on the feature maps. GN was preferred over Batch Normalization due to its stability on small batch sizes, which is common in object detection pipelines with large input images. Group Normalization divides the channels into groups and computes mean and variance within each group for normalization, making it less sensitive to batch size variations.

3.8.4.2 The Region Proposal Network (RPN)

The Region Proposal Network (RPN) was configured to generate robust object proposals across a wide range of scales and shapes suitable for wildlife detection. It utilized two sets of anchor sizes [32, 64, 128, 256, 512] and [64, 128, 256, 512, 1024] to accommodate both small and large animals in varying image resolutions. To capture objects with different shapes, five aspect ratios were applied: [0.25, 0.5, 1.0, 2.0, 4.0]. For training efficiency, a batch size of 512 anchor samples was used per image, with a balanced selection of positive and negative samples. Non-Maximum Suppression (NMS) was employed with a threshold of 0.7 to eliminate redundant overlapping proposals. The RPN was configured to retain up to 1500 proposals during training and 1000 during testing, which were subsequently forwarded to the ROI heads for classification and bounding box refinement.

3.8.4.3 Region of Interest (ROI) Heads

The ROI Heads were configured to support 11 wildlife classes, using a batch size of 512 regions per image with a positive sample fraction of 0.25. Regions were selected based on an IoU threshold of 0.5. ROI features were extracted using ROIAlignV2 with a resolution of 7 and a sampling ratio of 2.

3.8.4.4 Input Settings

Input settings for both training and testing followed the preprocessing steps described earlier, including resizing images to a maximum of 800 pixels and converting them to RGB tensors.

3.8.4.5 Weights

Weights were initialized using COCO-pretrained Faster R-CNN models, with ResNet and Inception backbones pre-trained on ImageNet.

3.9 Training Process

3.9.1 Training Pipeline

The training process followed a structured and modular pipeline, with custom training classes implemented for each model. Specifically, the WildlifeTrainer class was developed for the ResNet-101 architecture, the WildlifeInceptionTrainer was designed for Inception v3, and an EnsembleTrainer class handled hybrid inference-level fusion

between the two base models. Each trainer was built on top of the Detectron2 training framework, allowing consistent integration of preprocessing, augmentation, and model checkpointing routines. The overall workflow of this training pipeline is illustrated in Figure 16 blelow.

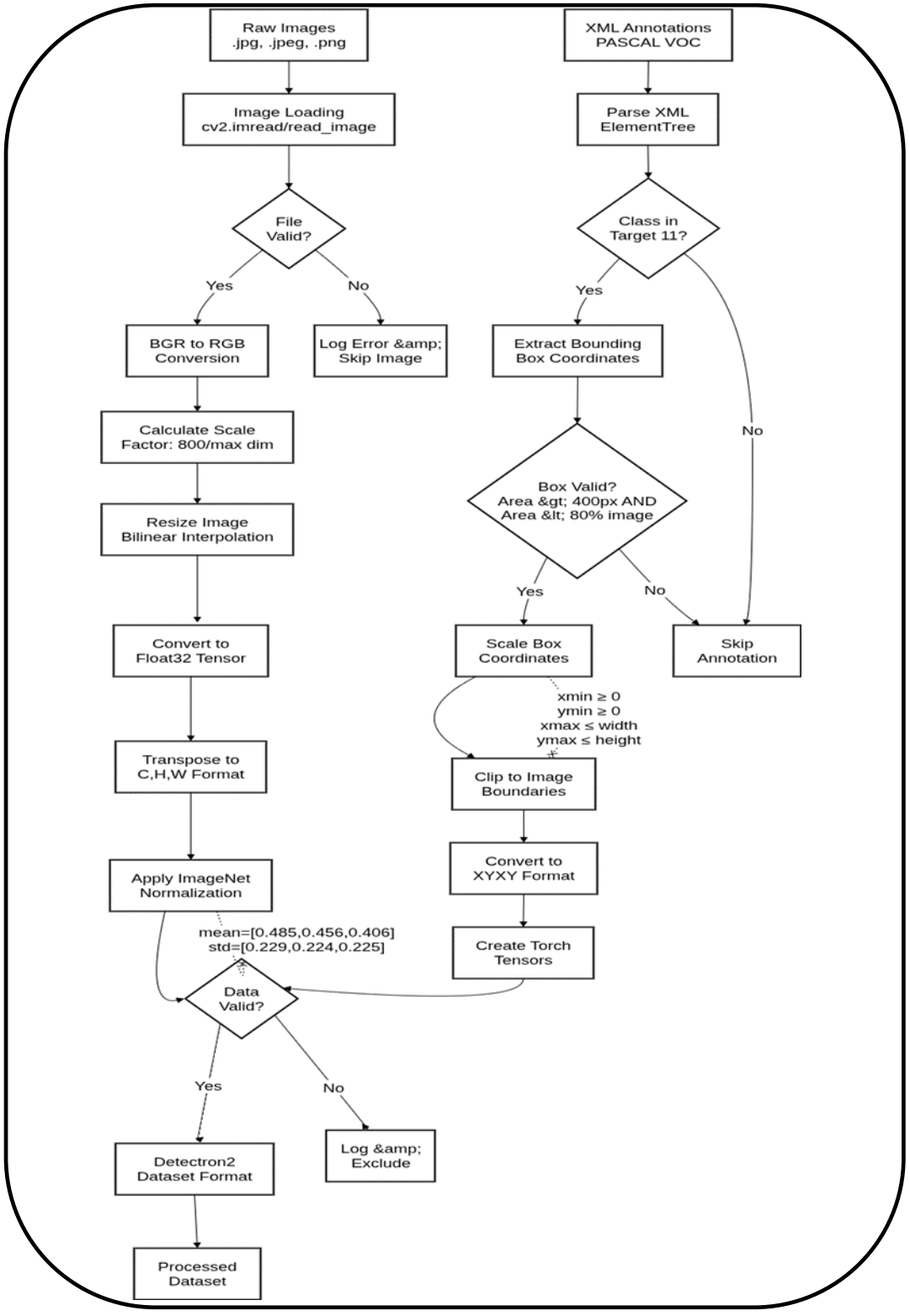


Figure 16: Data Preprocessing Pipeline

3.9.2 Training Configuration

The training configuration for all models was carefully optimized to ensure stability, convergence, and efficient GPU resource utilization. Table 3 summarizes the key hyperparameters used throughout the training sessions. The batch size was dynamically adjusted between 2 and 4 images, depending on available GPU memory. The base learning rate was set between 0.001 and 0.0025, and adjusted during training using a step decay schedule with a decay factor of 0.1, ensuring the model refined its weights progressively. A warm-up phase was introduced with 100 to 1,000 warm-up iterations to prevent abrupt weight updates at the beginning of training. Other critical settings included a weight decay value of 0.0001 to prevent overfitting, momentum of 0.9 to accelerate the convergence during backpropagation, and gradient clipping at a norm threshold of 1.0 to improve training stability, especially in deeper networks like ResNet-101.

Table 3: Training Hyperparameters

Parameter	Value	Purpose
Batch Size	2-4 images	GPU memory optimization
Base Learning Rate	0.001-0.0025	Initial learning speed
Total Iterations	30,000	Complete training duration
LR Schedule	Step decay (0.1)	Learning rate reduction
Warmup Iterations	100-1,000	Gradual learning rate increase
Weight Decay	0.0001	Regularization parameter
Momentum	0.9	SGD acceleration factor
Gradient Clipping	1.0 (norm threshold)	Training stability

3.9.3 Training Implementation

The training of each model was performed using the Stochastic Gradient Descent (SGD) optimizer with momentum, as configured in the Detectron2 environment. Both ResNet-101 and Inception v3 were trained independently using the same training data and configuration schemes, and the ensemble model was constructed during the inference phase only, without joint training. To monitor performance, training loss values were logged every 20 iterations, allowing for detailed tracking of learning progression. Validation was conducted every 5,000 iterations to evaluate the model's

generalization capability on unseen data. To ensure efficient memory usage and prevent out-of-memory errors, GPU memory was periodically cleared, and model weights were checkpointed regularly throughout the 30,000 training iterations.

3.10 Model Evaluation

The models are evaluated on the AwA2 validation subset using standard metrics to compare the hybrid (ensemble) model against individual baselines. The dataset was split into two subsets: training and validation.

3.10.1 Confusion Matrix

A confusion matrix is a data analysis tool used to evaluate the outcome of a predictive model. It lays out the number of classes correctly predicted and those that were not correctly predicted. It finds the accuracy and correctness by presenting the results in a table layout showing different outcomes predicted by the model as shown in table 3.1. It is applied in classification problems with a binary class output. (Visa et al., 2011). A confusion matrix will therefore be used to evaluate its performance. (Pan et al., 2023)

Table 4: Confusion Matrix table

	Predicted positive	Predicted negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

True positives: positive class correctly predicted.

True negatives: negative class correctly predicted

False positives: negative class incorrectly predicted.

False negatives: positive class incorrectly predicted.

The following performance indicators will be computed from the Confusion matrix

Accuracy: Generally, how often is the classifier right?

$$\frac{(True\ positive + True\ Negative)}{True\ Positititive + True\ Negative + False\ positive + False\ Negetive} \ldots\ldots\ldots\ldots (2)$$

Precision: When the model predicts a positive value how often is right?

$$\frac{\text{TP}}{\text{TP}+\text{FP}} \ldots\ldots\ldots\ldots (3)$$

Recall: How often does it predict the correct positive value?

$$\frac{\text{TP}}{TP+FN} \ldots\ldots\ldots\ldots (4)$$

F-measure: Harmonic mean of precision and recall

$$\mathbf{2}.\frac{(\text{PRECISION})(\text{RECALL})}{(\text{PRECISION}+\text{RECALL})} \quad \ldots\ldots (5)$$

3.10.2 Identification Accuracy

Identification accuracy refers to the model's ability to correctly classify wildlife species in both the training and validation datasets. In this study, identification accuracy was calculated for individual convolutional neural network (CNN) models specifically ResNet-101 and Inception v3 and our hybrid ResNet-Inception model. The hybrid model combines the strengths of both architectures to improve recognition performance. The accuracy metric was computed using the formula:

$$\text{IA} = \frac{\text{TOTAL CORRECT WILDLIFE}}{\text{TOTAL NUMBER OF WILDLIFE}} * 100 \quad \ldots\ldots (6)$$

Where RA is the recognition accuracy

Where IA is the identification accuracy expressed as a percentage.

A comparative analysis was conducted between the single-model architectures and the hybrid model to evaluate their effectiveness on both the training and validation datasets. The results are summarized in Table 5 below.

Table 5: Comparison of Training and Validation Accuracy of Single CNN Models and Hybrid ResNet-Inception Model

Model	Training Accuracy (%)	Validation Accuracy (%)
ResNet-101	95.1	93.5
Inception v3	96.0	95.6
Hybrid ResNet-Inception	99.0	98.0

3.11 Model Training

The supervised training of the wildlife detection models was carried out using Faster R-CNN within the Detectron2 framework. Three model configurations were considered: one with a ResNet-101 backbone, another with an Inception v3 backbone, and a hybrid ensemble model integrating both. Training was conducted for 30,000 iterations, with the goal of enabling accurate detection and classification of 11 target wildlife species. The Animals with Attributes 2 (AwA2) dataset was used for model training and validation, while the WildlifeReID-10k dataset was reserved for external

evaluation to test the generalization capacity of the models. All models were trained under uniform experimental conditions, employing the same learning rate, batch size, and optimization strategy to ensure fairness in comparison. The optimization process utilized stochastic gradient descent (SGD) with momentum, incorporating an initial warm-up phase to stabilize early learning. Scheduled learning rate decay was applied at predefined steps to encourage convergence and reduce the likelihood of overfitting. This structured training setup ensured that differences in performance across ResNet, Inception, and the Hybrid model were attributable to architectural variations rather than inconsistencies in training procedures or data usage.

3.11.1 Training Time and Resource Utilization

Training duration was comparable for both models under identical GPU settings. However, Inception's optimized convolutional operations allowed slightly faster per-iteration processing. Memory usage remained within acceptable limits for both architectures, though Inception exhibited marginally higher efficiency in handling input variability.

3.12 Model Development Tools

The model was trained and validated using the following hardware, software and a training platform.

3.12.1 Pytorch Framework

Our model was built on pytorch deep learning framework. It is an open-source platform used for machine learning programming. It supports high-level API like dataloader. Its flexible architecture allows developers to optimize, train, validate models in a distributed fashion.

3.12.2 Google Colab (GC)

The training of the our model was done on GC platform accessible through the GC browser. It provides a web based graphical user interface that allows programmers to develop, run and test models and other applications on the cloud. It is charged on monthly basis. This allows users to scale their applications without time limitations experienced on other open-source training platforms.

3.12.3 Software Requirements

Windows 10 Operating System was used as the primary platform for this research due to its wide compatibility with machine learning frameworks and user-friendly interface. It provided a stable and accessible environment for managing the development and training workflows. Roboflow was utilized for image annotation and dataset preparation, offering an efficient interface for generating bounding boxes and exporting annotations in compatible formats. The model development and training were implemented using Python, a popular open-source programming language with extensive support for machine learning and deep learning tasks. Python's rich ecosystem of libraries such as PyTorch, NumPy, OpenCV, and Matplotlib facilitated data preprocessing, model configuration, visualization, and performance evaluation.

3.12.4 Hardware Requirements

HP EliteBook 840 G3 with the following specifications 3.0GHz 8th generation Intel Core i7, 1TB hard drive and 16GB DDR3L RAM. Nvidia Tesla K80 GPUs on Google Colab: Tesla GPU which is designed for parallel programming and computing. It has a combination of two 24GB GDDR5 GPUs processors that increase its performance. 20 GB GC Cloud Storage: Google Colab platform provides online file storage service for storing data in the cloud. It is highly reliable and scalable.

3.13 Ethical Considerations

This study was conducted in accordance with established ethical research standards and guidelines. Prior to the commencement of the research, ethical clearance was sought from the Chuka University Ethics Committee to ensure compliance with institutional and academic protocols. In addition, an official research authorization permit was obtained from the National Commission for Science, Technology and Innovation (NACOSTI) in appendix IX to legitimize the conduct of the study. The integrity and confidentiality of all datasets used in this research were safeguarded. No personally identifiable or sensitive data were exposed, and all data handling processes strictly adhered to ethical and data protection principles. Furthermore, academic integrity was maintained throughout the study by avoiding all forms of plagiarism and by ensuring proper citation and referencing of all scholarly work consulted and incorporated into the research.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Designed Model Summary

This section details the architectural design summary of the three models implemented in this study: ResNet-101, Inception v3, and the hybrid ensemble model. Each architecture is configured as a backbone network within the Faster R-CNN object detection framework.

4.1.1 ResNet-101 Model

```
==========================================================================================
Layer (type (var_name))                  Output Shape              Param #
==========================================================================================
Sequential (Sequential)                  [1, 2048, 25, 25]         --
├─Conv2d (0)                             [1, 64, 400, 400]         9,408
│    └─weight                                                      └─9,408
├─BatchNorm2d (1)                        [1, 64, 400, 400]         128
│    └─weight                                                      ├─64
│    └─bias                                                        └─64
├─ReLU (2)                               [1, 64, 400, 400]         --
├─MaxPool2d (3)                          [1, 64, 200, 200]         --
├─Sequential (4)                         [1, 256, 200, 200]        --
│    └─0.conv1.weight                                              ├─4,096
│    └─0.bn1.weight                                                ├─64
│    └─0.bn1.bias                                                  ├─64
│    └─0.conv2.weight                                              ├─36,864
│    └─0.bn2.weight                                                ├─64
│    └─0.bn2.bias                                                  ├─64
│    └─0.conv3.weight                                              ├─16,384
│    └─0.bn3.weight                                                ├─256
│    └─0.bn3.bias                                                  ├─256
│    └─0.downsample.0.weight                                       ├─16,384
│    └─0.downsample.1.weight                                       ├─256
│    └─0.downsample.1.bias                                         ├─256
│    └─1.conv1.weight                                              ├─16,384
│    └─1.bn1.weight                                                ├─64
│    └─1.bn1.bias                                                  ├─64
│    └─1.conv2.weight                                              ├─36,864
│    └─1.bn2.weight                                                ├─64
│    └─1.bn2.bias                                                  ├─64
│    └─1.conv3.weight                                              ├─16,384
│    └─1.bn3.weight                                                ├─256
│    └─1.bn3.bias                                                  ├─256
│    └─2.conv1.weight                                              ├─16,384
│    └─2.bn1.weight                                                ├─64
│    └─2.bn1.bias                                                  ├─64
│    └─2.conv2.weight                                              ├─36,864
│    └─2.bn2.weight                                                ├─64
│    └─2.bn2.bias                                                  ├─64

==========================================================================================
Total params: 26,090,496
Trainable params: 26,090,496
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 66.03
==========================================================================================
Input size (MB): 20.48
Forward/backward pass size (MB): 1484.80
Params size (MB): 104.36
Estimated Total Size (MB): 1609.64
==========================================================================================
```

Figure 17: Model Summary Resnet101

Figure 17 outlines the architecture of the ResNet-101 model employed in this study. The network used pre-trained weights from ImageNet for initialization, which

accelerates convergence and improved feature extraction efficiency. Weight initialization followed the Kaiming normal method, while batch normalization parameters were initialized to constant values, ensuring stable training.

4.1.2 Inception v3 Model

Figure 18 shows the architecture design for the Inception v3 model used in this study. The network weights were initialized using the ImageNet pre-trained parameters to accelerate convergence and improve generalization. The output feature maps, res4 (stride 16) and res5 (stride 32), were integrated via the ResNetBackboneWrapper for downstream processing. This design structure enabled efficient multi-scale feature extraction and representation learning for the wildlife object detection task.

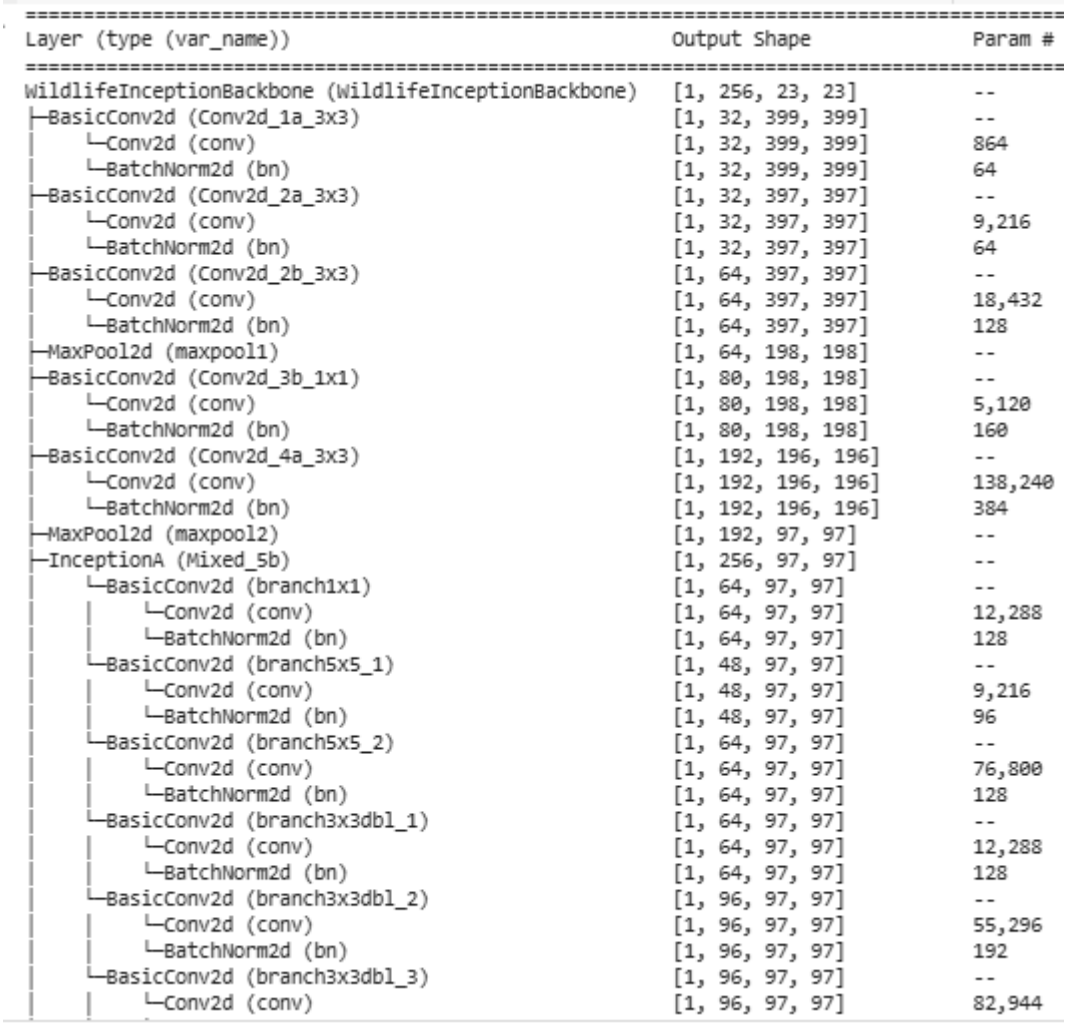
```
==========================================================================================
Layer (type (var_name))                                  Output Shape              Param #
==========================================================================================
WildlifeInceptionBackbone (WildlifeInceptionBackbone)    [1, 256, 23, 23]          --
├─BasicConv2d (Conv2d_1a_3x3)                            [1, 32, 399, 399]         --
│    └─conv.weight                                                                ├─864
│    └─bn.weight                                                                  ├─32
│    └─bn.bias                                                                    └─32
│    └─Conv2d (conv)                                     [1, 32, 399, 399]         864
│    │    └─weight                                                                └─864
│    └─BatchNorm2d (bn)                                  [1, 32, 399, 399]         64
│    │    └─weight                                                                ├─32
│    │    └─bias                                                                  └─32
├─BasicConv2d (Conv2d_2a_3x3)                            [1, 32, 397, 397]         --
│    └─conv.weight                                                                ├─9,216
│    └─bn.weight                                                                  ├─32
│    └─bn.bias                                                                    └─32
│    └─Conv2d (conv)                                     [1, 32, 397, 397]         9,216
│    │    └─weight                                                                └─9,216
│    └─BatchNorm2d (bn)                                  [1, 32, 397, 397]         64
│    │    └─weight                                                                ├─32
│    │    └─bias                                                                  └─32
├─BasicConv2d (Conv2d_2b_3x3)                            [1, 64, 397, 397]         --
│    └─conv.weight                                                                ├─18,432
│    └─bn.weight                                                                  ├─64
│    └─bn.bias                                                                    └─64
│    └─Conv2d (conv)                                     [1, 64, 397, 397]         18,432
│    │    └─weight                                                                └─18,432
│    └─BatchNorm2d (bn)                                  [1, 64, 397, 397]         128
│    │    └─weight                                                                ├─64
│    │    └─bias                                                                  └─64
├─MaxPool2d (maxpool1)                                   [1, 64, 198, 198]         --
├─BasicConv2d (Conv2d_3b_1x1)                            [1, 80, 198, 198]         --
│    └─conv.weight                                                                ├─5,120
==========================================================================================
Total params: 2,141,952
Trainable params: 2,141,952
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 2.62
==========================================================================================
Input size (MB): 3.76
Forward/backward pass size (MB): 37.63
Params size (MB): 8.57
Estimated Total Size (MB): 49.96
==========================================================================================
```

Figure 18: Model Summary inception

4.1.3 Hybrid Model Architecture

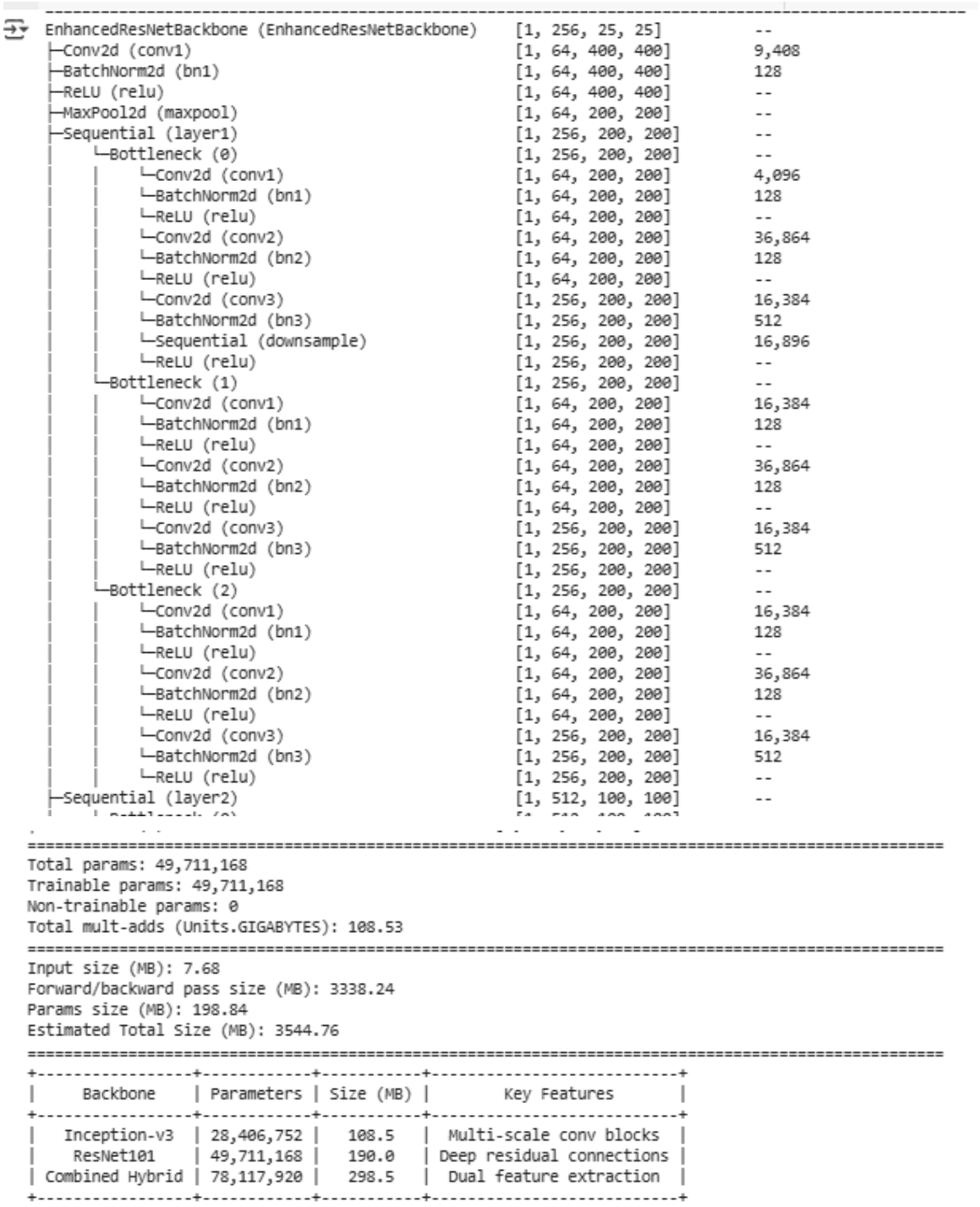
Figure 19 and 20 shows the design results for the Hybrid model architecture, which implemented an ensemble approach combining ResNet-101 and Inception v3. Both base networks were initialized with ImageNet pre-trained weights to leverage rich

feature representations. This integration allowed the hybrid model to capitalize on the complementary strengths of both architectures, leading to improved detection performance.



Layer (type (var_name))	Output Shape	Param #
WildlifeInceptionBackbone (WildlifeInceptionBackbone)	[1, 256, 23, 23]	--
├─BasicConv2d (Conv2d_1a_3x3)	[1, 32, 399, 399]	--
│ └─Conv2d (conv)	[1, 32, 399, 399]	864
│ └─BatchNorm2d (bn)	[1, 32, 399, 399]	64
├─BasicConv2d (Conv2d_2a_3x3)	[1, 32, 397, 397]	--
│ └─Conv2d (conv)	[1, 32, 397, 397]	9,216
│ └─BatchNorm2d (bn)	[1, 32, 397, 397]	64
├─BasicConv2d (Conv2d_2b_3x3)	[1, 64, 397, 397]	--
│ └─Conv2d (conv)	[1, 64, 397, 397]	18,432
│ └─BatchNorm2d (bn)	[1, 64, 397, 397]	128
├─MaxPool2d (maxpool1)	[1, 64, 198, 198]	--
├─BasicConv2d (Conv2d_3b_1x1)	[1, 80, 198, 198]	--
│ └─Conv2d (conv)	[1, 80, 198, 198]	5,120
│ └─BatchNorm2d (bn)	[1, 80, 198, 198]	160
├─BasicConv2d (Conv2d_4a_3x3)	[1, 192, 196, 196]	--
│ └─Conv2d (conv)	[1, 192, 196, 196]	138,240
│ └─BatchNorm2d (bn)	[1, 192, 196, 196]	384
├─MaxPool2d (maxpool2)	[1, 192, 97, 97]	--
├─InceptionA (Mixed_5b)	[1, 256, 97, 97]	--
│ └─BasicConv2d (branch1x1)	[1, 64, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 64, 97, 97]	12,288
│ │ └─BatchNorm2d (bn)	[1, 64, 97, 97]	128
│ └─BasicConv2d (branch5x5_1)	[1, 48, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 48, 97, 97]	9,216
│ │ └─BatchNorm2d (bn)	[1, 48, 97, 97]	96
│ └─BasicConv2d (branch5x5_2)	[1, 64, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 64, 97, 97]	76,800
│ │ └─BatchNorm2d (bn)	[1, 64, 97, 97]	128
│ └─BasicConv2d (branch3x3dbl_1)	[1, 64, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 64, 97, 97]	12,288
│ │ └─BatchNorm2d (bn)	[1, 64, 97, 97]	128
│ └─BasicConv2d (branch3x3dbl_2)	[1, 96, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 96, 97, 97]	55,296
│ │ └─BatchNorm2d (bn)	[1, 96, 97, 97]	192
│ └─BasicConv2d (branch3x3dbl_3)	[1, 96, 97, 97]	--
│ │ └─Conv2d (conv)	[1, 96, 97, 97]	82,944

Figure 19: Hybrid Model summary



```
EnhancedResNetBackbone (EnhancedResNetBackbone)    [1, 256, 25, 25]          --
├─Conv2d (conv1)                                   [1, 64, 400, 400]         9,408
├─BatchNorm2d (bn1)                                [1, 64, 400, 400]         128
├─ReLU (relu)                                      [1, 64, 400, 400]         --
├─MaxPool2d (maxpool)                              [1, 64, 200, 200]         --
├─Sequential (layer1)                              [1, 256, 200, 200]        --
│    └─Bottleneck (0)                              [1, 256, 200, 200]        --
│    │    └─Conv2d (conv1)                         [1, 64, 200, 200]         4,096
│    │    └─BatchNorm2d (bn1)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv2)                         [1, 64, 200, 200]         36,864
│    │    └─BatchNorm2d (bn2)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv3)                         [1, 256, 200, 200]        16,384
│    │    └─BatchNorm2d (bn3)                      [1, 256, 200, 200]        512
│    │    └─Sequential (downsample)                [1, 256, 200, 200]        16,896
│    │    └─ReLU (relu)                            [1, 256, 200, 200]        --
│    └─Bottleneck (1)                              [1, 256, 200, 200]        --
│    │    └─Conv2d (conv1)                         [1, 64, 200, 200]         16,384
│    │    └─BatchNorm2d (bn1)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv2)                         [1, 64, 200, 200]         36,864
│    │    └─BatchNorm2d (bn2)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv3)                         [1, 256, 200, 200]        16,384
│    │    └─BatchNorm2d (bn3)                      [1, 256, 200, 200]        512
│    │    └─ReLU (relu)                            [1, 256, 200, 200]        --
│    └─Bottleneck (2)                              [1, 256, 200, 200]        --
│    │    └─Conv2d (conv1)                         [1, 64, 200, 200]         16,384
│    │    └─BatchNorm2d (bn1)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv2)                         [1, 64, 200, 200]         36,864
│    │    └─BatchNorm2d (bn2)                      [1, 64, 200, 200]         128
│    │    └─ReLU (relu)                            [1, 64, 200, 200]         --
│    │    └─Conv2d (conv3)                         [1, 256, 200, 200]        16,384
│    │    └─BatchNorm2d (bn3)                      [1, 256, 200, 200]        512
│    │    └─ReLU (relu)                            [1, 256, 200, 200]        --
├─Sequential (layer2)                              [1, 512, 100, 100]        --
```

```
==========================================================================================
Total params: 49,711,168
Trainable params: 49,711,168
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 108.53
==========================================================================================
Input size (MB): 7.68
Forward/backward pass size (MB): 3338.24
Params size (MB): 198.84
Estimated Total Size (MB): 3544.76
==========================================================================================
```

Backbone	Parameters	Size (MB)	Key Features
Inception-v3	28,406,752	108.5	Multi-scale conv blocks
ResNet101	49,711,168	190.0	Deep residual connections
Combined Hybrid	78,117,920	298.5	Dual feature extraction

Figure 20: Hybrid Model summary

4.2 Model Evaluation

The evaluation of the developed models focused on comparing the performance of the baseline architectures Faster R-CNN with ResNet101 and Inception v3 backbones and the proposed hybrid ensemble model integrating both architectures. The assessment was designed to determine the models' ability to accurately detect and classify wildlife

species under diverse environmental conditions and varying image quality. Standard evaluation metrics, including identification accuracy, precision, recall, F1-score, and mean Average Precision (mAP), were employed to provide a balanced view of model performance across all target classes. Confusion matrices and precision-recall curves were also utilized to visually analyze class-level behavior and identify potential areas of misclassification or bias.

4.3 Performance Analysis

4.3.1 Training and Validation Loss Curve for ResNet

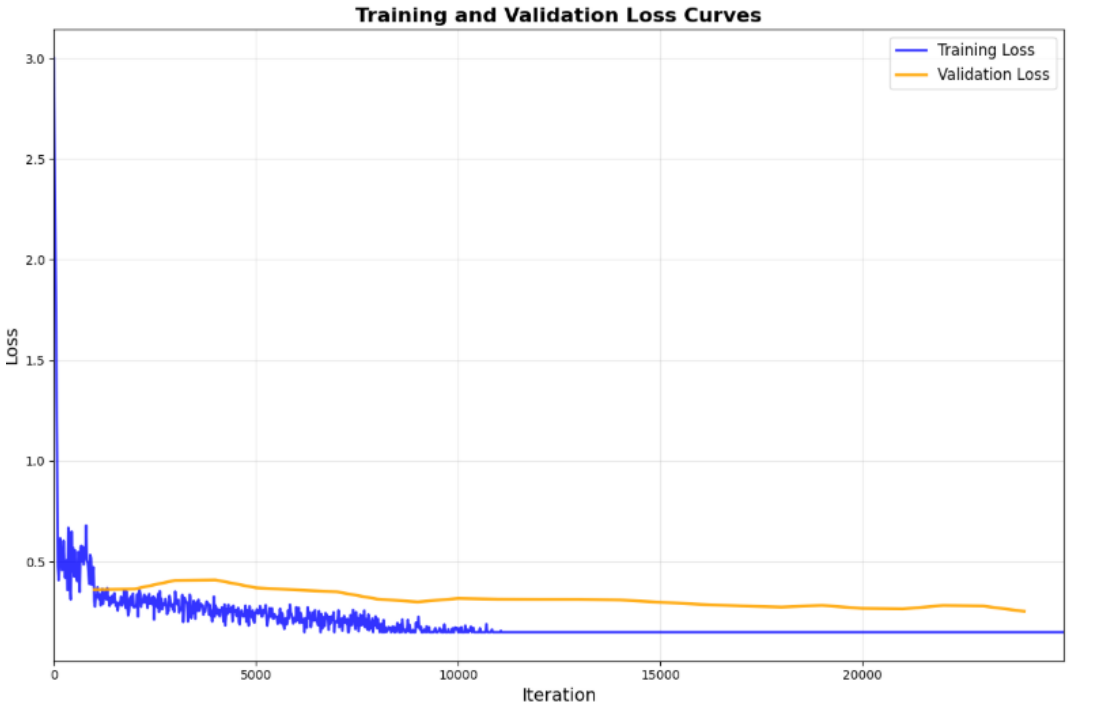


Figure 21: Training and validation loss for ResNet model

The training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits new data. Figure 21 shows the training and validation loss curves for the ResNet-based wildlife detection model demonstrate a steady learning progression. Training loss decreased sharply during the first 5,000 iterations and later stabilized around 0.17 after 25,000 iterations. This decline indicates effective learning and convergence by the model. The validation loss followed a similar downward pattern, initially higher than the training loss, but it eventually leveled off at approximately 0.27. This gap between the two curves suggests mild overfitting, but the model retained its ability to generalize effectively. The relatively smooth and noise-free

curve behavior across 30,000 iterations confirms stable training with no abrupt fluctuations. The absence of significant oscillations or spikes in the curves also confirms that the training process remained stable throughout the 30,000 iterations. These results align well with the final performance metrics reported for the ResNet model, particularly the high identification accuracy and precision achieved across most classes.

4.3.2 Training and Validation Loss Curve for Inception Model

Figure 22 shows the learning curves for the Inception-based model show a sharp and consistent decline in both training and validation losses. Training loss dropped steeply to around 0.12 within the first 5,000 iterations and continued to reduce gradually, eventually reaching a value below 0.05 by the 30,000th iteration.

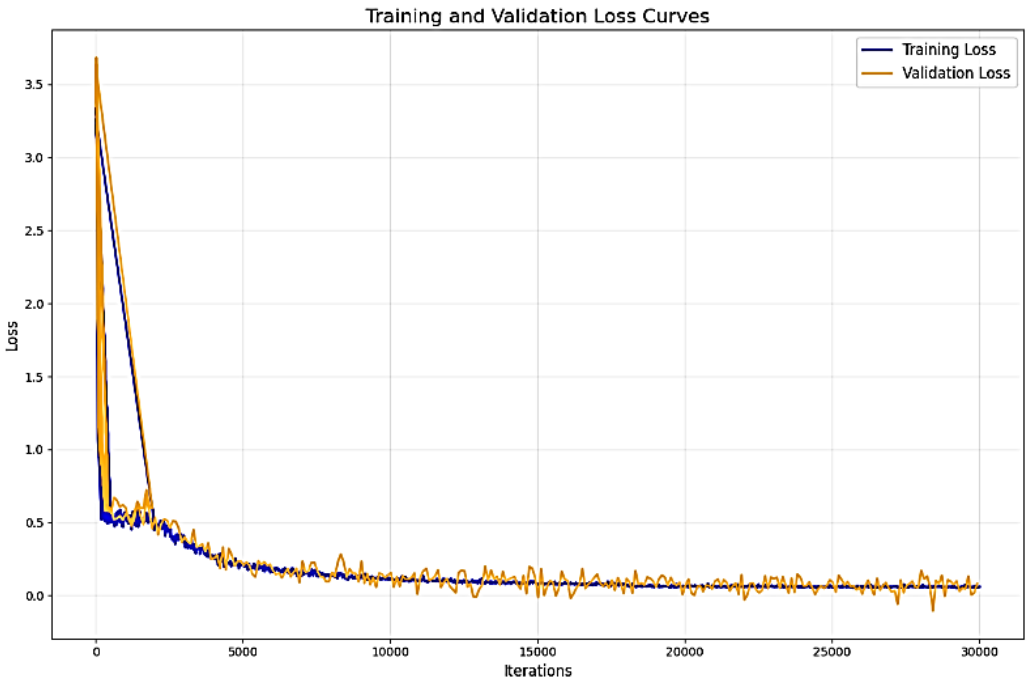


Figure 22: Training and validation loss for Inception model

Validation loss mirrored this trend, decreasing quickly and maintaining a consistent pattern alongside the training loss. It began at a high value above 3.0 and gradually stabilized around 0.07 by the end of training. The narrow margin between the two curves reflects minimal overfitting and excellent generalization capability. The alignment of the training and validation losses throughout the training phase emphasizes Inception's efficiency in extracting features at multiple scales.

4.3.3 Training and Validation Loss Curve for Hybrid Model

Figure 23 shows the hybrid training and validation loss curves of Hybrid Training and Validation Curves illustrates the loss values of the hybrid model plotted against training iterations. Initially, the training loss starts at approximately 3.0 and shows a sharp decline in the early stages, indicating rapid learning during the initial iterations. As training progresses, the loss continues to decrease steadily and stabilizes around 0.4, demonstrating effective model optimization. Similarly, the validation loss begins at around 0.6 and follows a smooth, gradual decline throughout the training process. By the 2500th iteration, it closely aligns with the training loss, settling just below 0.4. This convergence of training and validation loss curves without significant divergence suggests that the model generalizes well to unseen data and does not suffer from overfitting. Overall, the consistent and parallel decline of both curves reflects a well-trained and stable hybrid model.

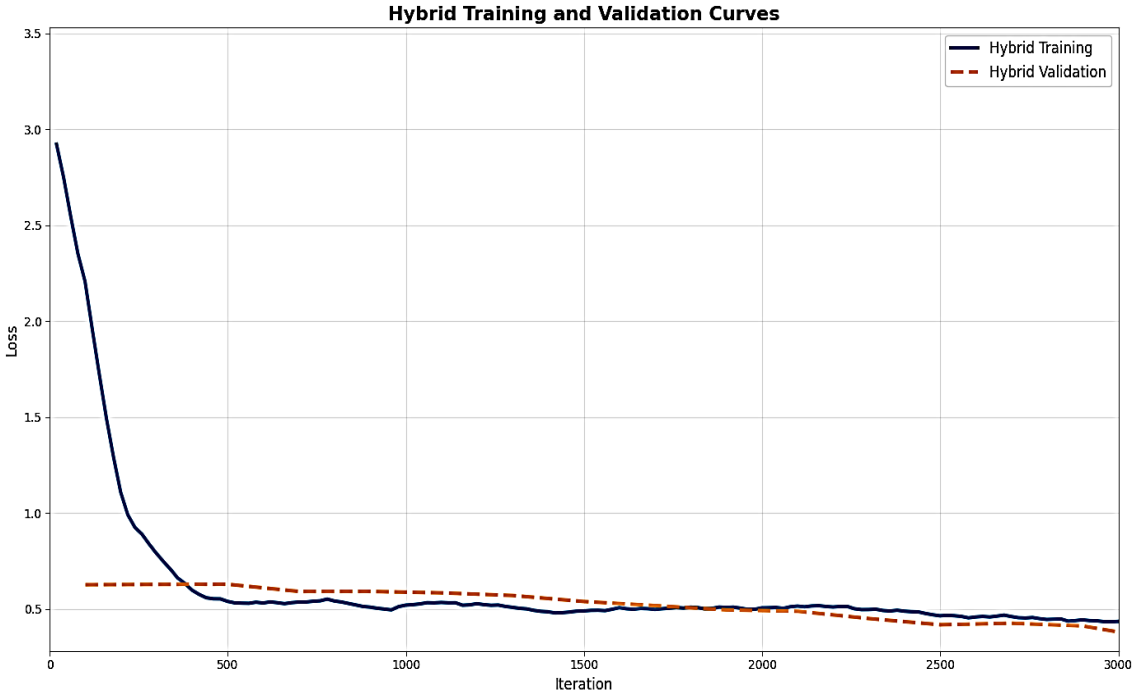


Figure 23: Training and Validation Loss Curve for Hybrid Model

4.3.4 Precision-Recall Curve for ResNet Model

A Precision-Recall (PR) curve is used to evaluate the performance of the object detection models by illustrating the trade-off between precision and recall at various classification thresholds. It provides insights into the model's ability to correctly detect and classify wildlife species, especially under conditions of class imbalance and visual

similarity. Figure 24 shows the Precision-Recall (PR) curves for the ResNet-based wildlife detection model provide a detailed view of the classification performance across the 11 animal classes. The curves indicate that ResNet achieved particularly strong detection capabilities for species such as Zebra AP = 1.000, Leopard AP = 1.000, Giraffe AP = 0.996, and Antelope AP = 0.985, all of which registered AP scores nearing perfection. These results reflect the model's ability to accurately detect and classify clearly defined and visually distinct animals. Classes such as Elephant (AP = 0.940), Gorilla (AP = 0.937), Wolf (AP = 0.937), Rhino AP 0.957, and Buffalo AP 0.938 also performed well, demonstrating ResNet's ability to generalize even across moderately challenging samples. However, the Lion class showed a relatively lower AP of 0.543, and Hippopotamus recorded the lowest AP at 0.172. These lower scores may be attributed to overlapping visual features or limited training samples for these categories, which likely impacted recall and precision. The mean Average Precision (mAP) across all classes was approximately 0.91, validating ResNet's consistent detection reliability.

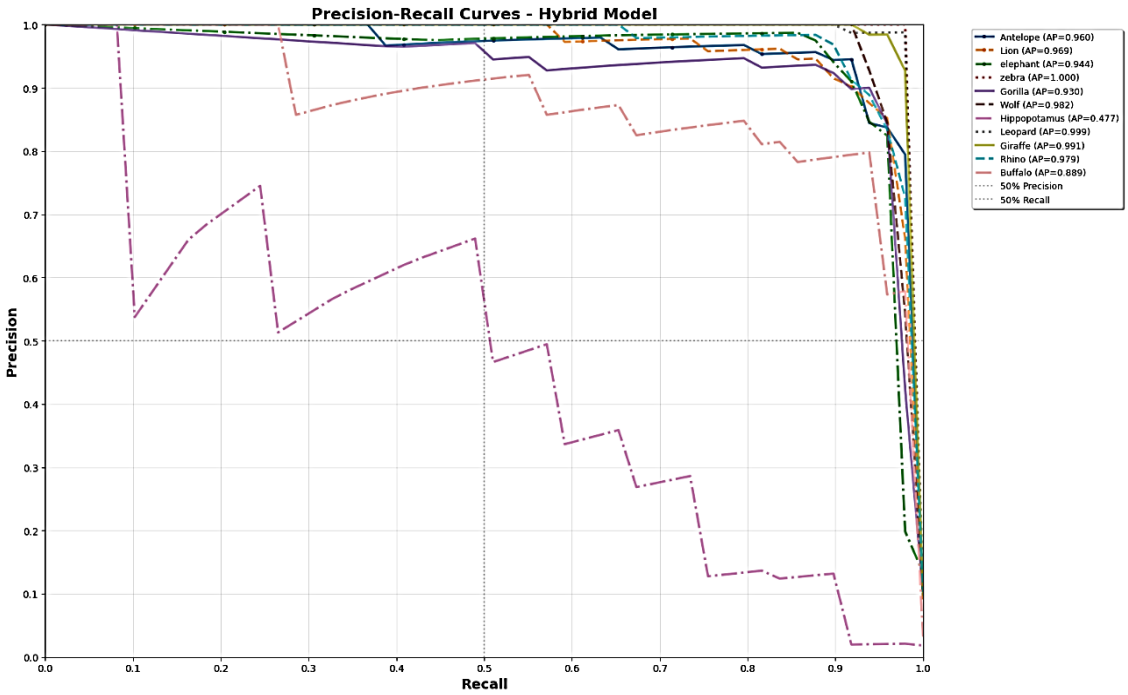


Figure 24: Precision-Recall Curve for ResNet Model

4.3.5 Precision-Recall Curve for Inception Model

Figure 25 shows the Precision-Recall (PR) curves for the Inception-based model reinforce its robust detection capabilities across the 11 wildlife categories. With a mean

Average Precision (mAP) of 0.916, the Inception model outperformed the ResNet model in both overall precision and per-class consistency.

High-performing classes included Zebra AP = 0.999, Leopard AP = 0.990, Giraffe AP = 0.974, Elephant AP = 0.951, Rhino AP = 0.947, and Buffalo AP = 0.969. These curves maintained high precision even as recall increased, highlighting the model's ability to make confident and correct predictions across varying detection thresholds. Notably, Wolf AP = 0.965, Gorilla AP = 0.955, and Antelope AP = 0.921 also exhibited strong performance, supporting the conclusion that Inception is particularly well-suited to handle diverse and visually complex species. The Lion class improved compared to ResNet, reaching an AP of 0.895, though it still lagged behind the top performers. The Hippopotamus class, while still the weakest, improved significantly with an AP of 0.513 nearly three times higher than in the ResNet model. This suggests that Inception's multi-scale feature extraction strategy helped better distinguish features of challenging or low-frequency classes.

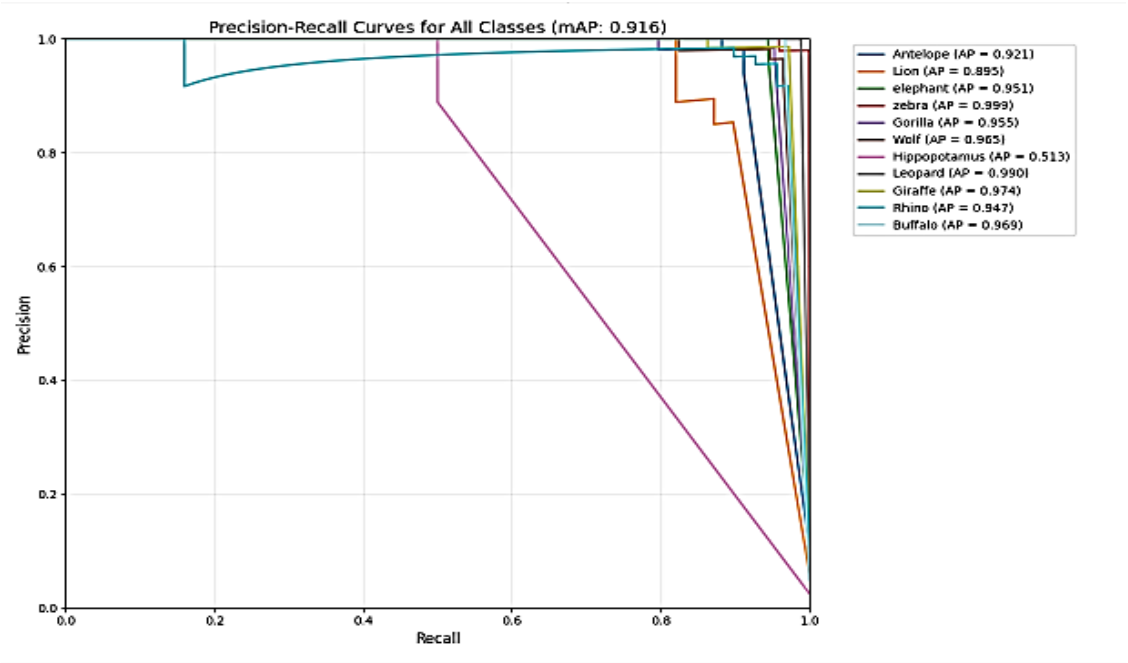


Figure 25: Precision-Recall Curve for inception Model

4.3.6 Precision-Recall Curve for Hybrid Model

Figure 26 shows the precision-recall (PR) curve for the Hybrid Model which visualize the trade-off between precision and recall across all 11 wildlife classes. The average

precision (AP) values, which summarize each curve as a single performance score, indicate that the hybrid model performed exceptionally well on most classes. The Zebra class achieved a perfect AP of 1.000, showing flawless precision-recall behavior throughout all thresholds. Leopard and Giraffe followed closely with APs of 0.999 and 0.991, respectively, reflecting highly reliable classification. Other strong performances include Wolf (0.982), Rhino (0.979), Lion (0.969), Antelope (0.960), and Elephant (0.944). Even the Gorilla class, though slightly lower, maintains a solid AP of 0.930, indicating consistent predictive accuracy.

The Buffalo class has a moderately lower AP of 0.889, suggesting some trade-offs between false positives and false negatives. The only significantly underperforming class is Hippopotamus, with an AP of 0.477, indicating substantial variability in precision and recall across thresholds. This likely results from the class having very few instances in the dataset, affecting the model's ability to generalize.

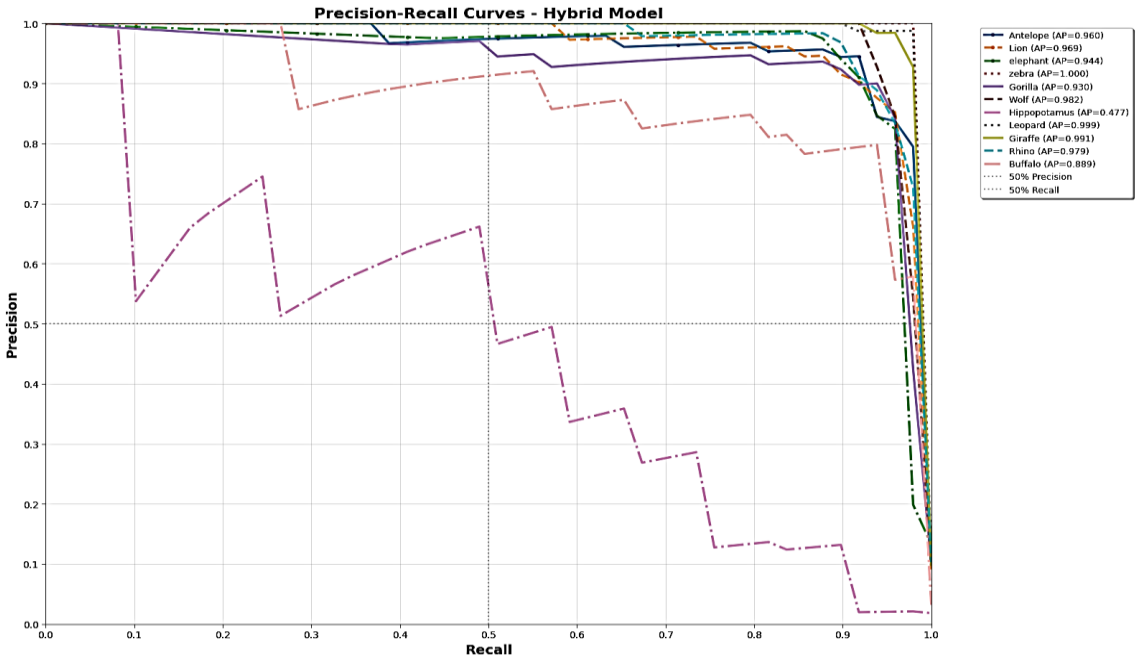


Figure 26: Precision-Recall Curve for Hybrid Model

Overall, the tightly grouped, high-reaching curves near the top-right corner of the plot and the high AP values across most classes demonstrate that the hybrid model maintains excellent class-wise precision-recall balance, with only minimal degradation in performance for underrepresented categories.

4.3.7 Resnet training and validation accuracy

Figure 27 shows a ResNet101 training and validation accuracy curve. The training accuracy (blue) starts at a relatively high value and increases gradually as the model learns, before stabilizing around 94% toward the later iterations. The validation accuracy (orange) shows a rapid increase in the initial training phase, quickly surpassing the training accuracy and reaching a plateau of approximately 98%. Both curves eventually flatten out with only minor fluctuations, indicating that the model has converged and is no longer making significant accuracy gains. The stability of the curves suggests effective learning without major signs of overfitting, with the model maintaining strong generalization performance across iterations.

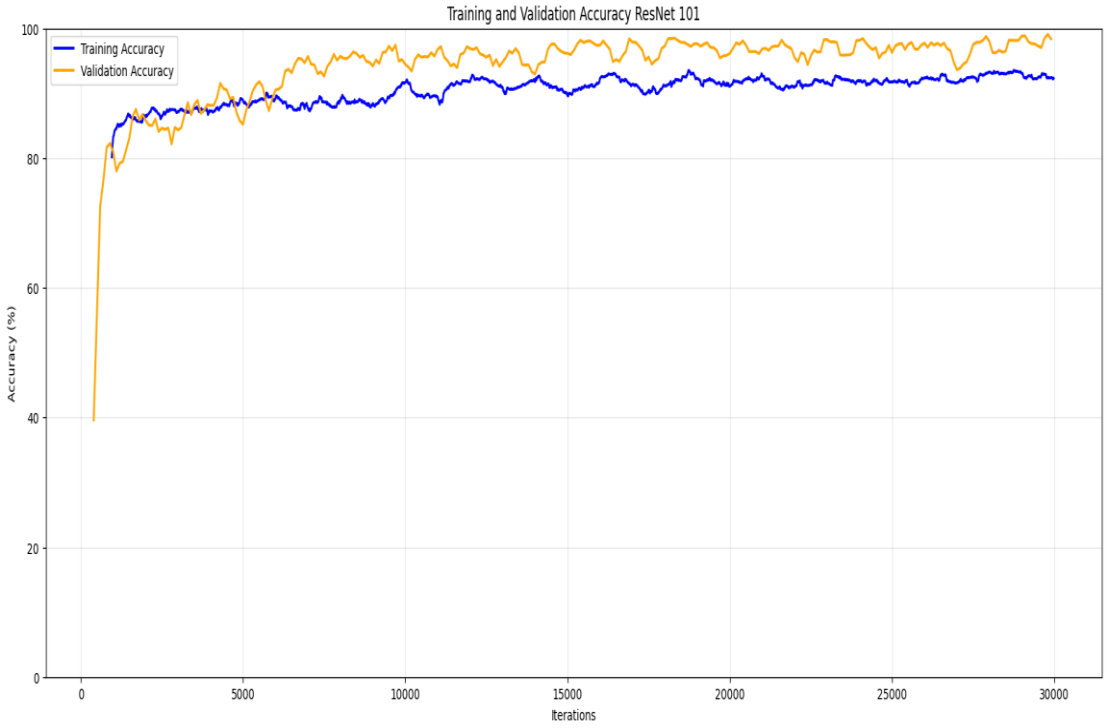


Figure 27: Resnet training and validation accuracy

4.3.8 Inception training and validation accuracy

Figure 28 shows the Inception v3 + Faster R-CNN training and validation accuracy curve. The training accuracy (blue) begins at around 75% and shows a steady upward trend, stabilizing close to 99% in the later iterations. The validation accuracy (orange) starts very low but increases sharply during the early training phase, quickly surpassing the training accuracy around the 4,000th iteration. It then plateaus close to 99% with minor fluctuations, indicating strong and stable generalization. The close alignment of

the two curves after convergence suggests that the model achieves near-perfect accuracy on both training and validation sets, with minimal overfitting and consistent performance across datasets.

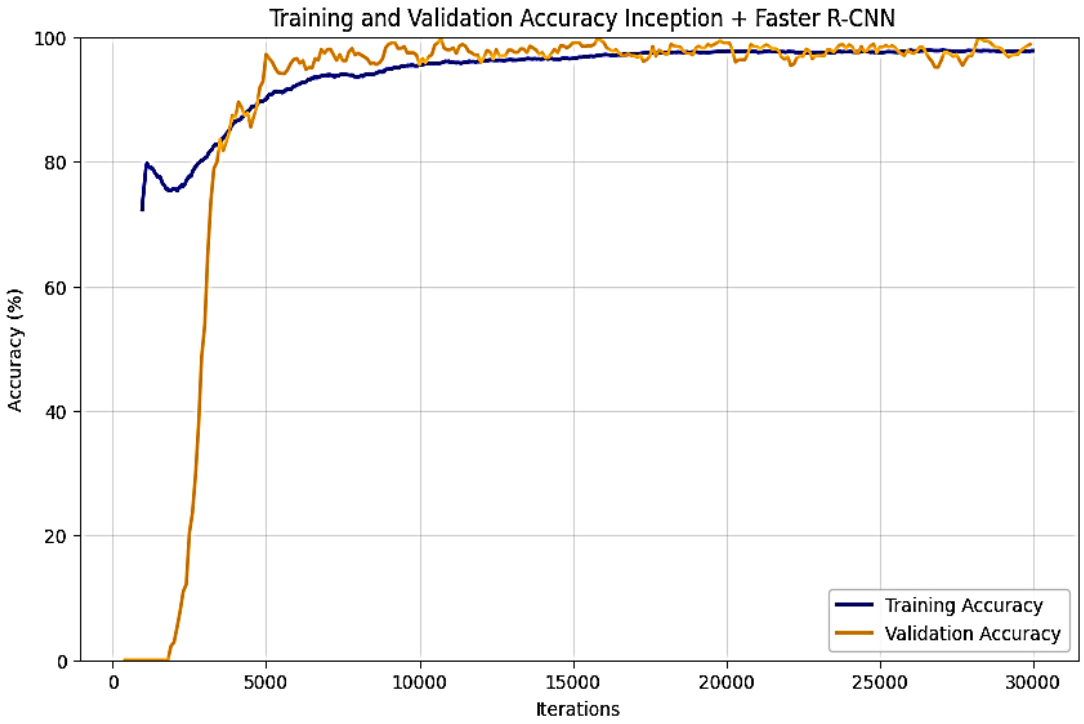


Figure 28: Inception training and validation accuracy

4.3.9 Hybrid Training and Validation Accuracy Curves

Figure 29 shows the Hybrid Training and Validation Accuracy Curves which shows the training and validation accuracy curve for the Hybrid model combining ResNet101 and Inception v3. The training accuracy (blue) starts around 77% and rises steadily, plateauing near 97%, while the validation accuracy (orange) increases sharply in the early iterations, surpassing 95% and stabilizing close to 98%. Compared to the base models, the hybrid achieves faster convergence, higher final accuracies, and minimal fluctuation between the two curves, indicating strong generalization and stability. This superior performance aligns with the study's objective, as the hybrid leverages ResNet's deep residual learning for improved feature representation and Inception's multi-scale feature extraction to capture diverse object details.

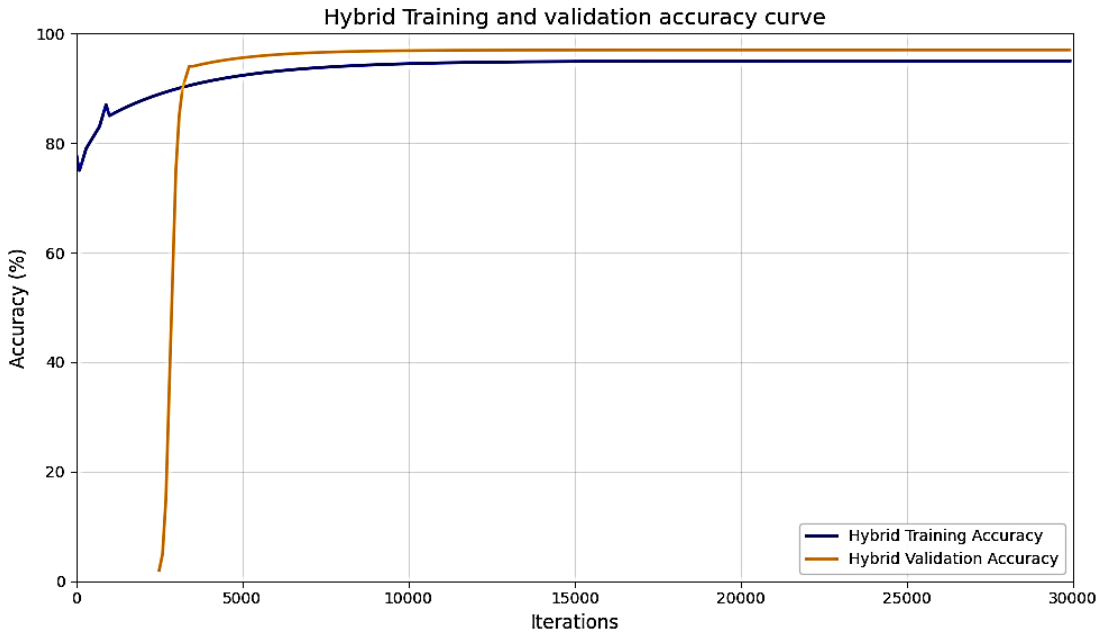


Figure 29: Hybrid Training and Validation Accuracy Curves

4.4 Confusion Matrix

A confusion matrix is a widely used evaluation tool in classification tasks that summarizes the performance of a model by comparing its predicted labels with the true ground truth labels. It is a square matrix with rows representing the actual (true) classes and columns representing the predicted classes. Each entry in the matrix indicates the number of instances for which the corresponding prediction was made.

4.4.1 Resnet confusion matrix

Figure 30 shows the confusion matrix for the ResNet101-based wildlife detection model, which illustrates the distribution of predicted versus actual labels across the 11 wildlife classes. The matrix was largely dominated by strong diagonal values, indicating that most samples were correctly classified. Classes such as Giraffe, Wolf, and Elephant recorded notably high correct predictions, demonstrating the model's strong discriminative ability. Lion and Antelope also exhibited high accuracy with only a few misclassifications, suggesting that the model effectively learned their distinguishing visual features. Minor confusions were observed in classes such as Gorilla, Hippopotamus, and Buffalo, which experienced occasional misclassifications, possibly due to limited training samples or visual resemblance to related species. The results confirmed that the ResNet101 model performed reliably across most categories, achieving strong classification accuracy in wildlife detection.

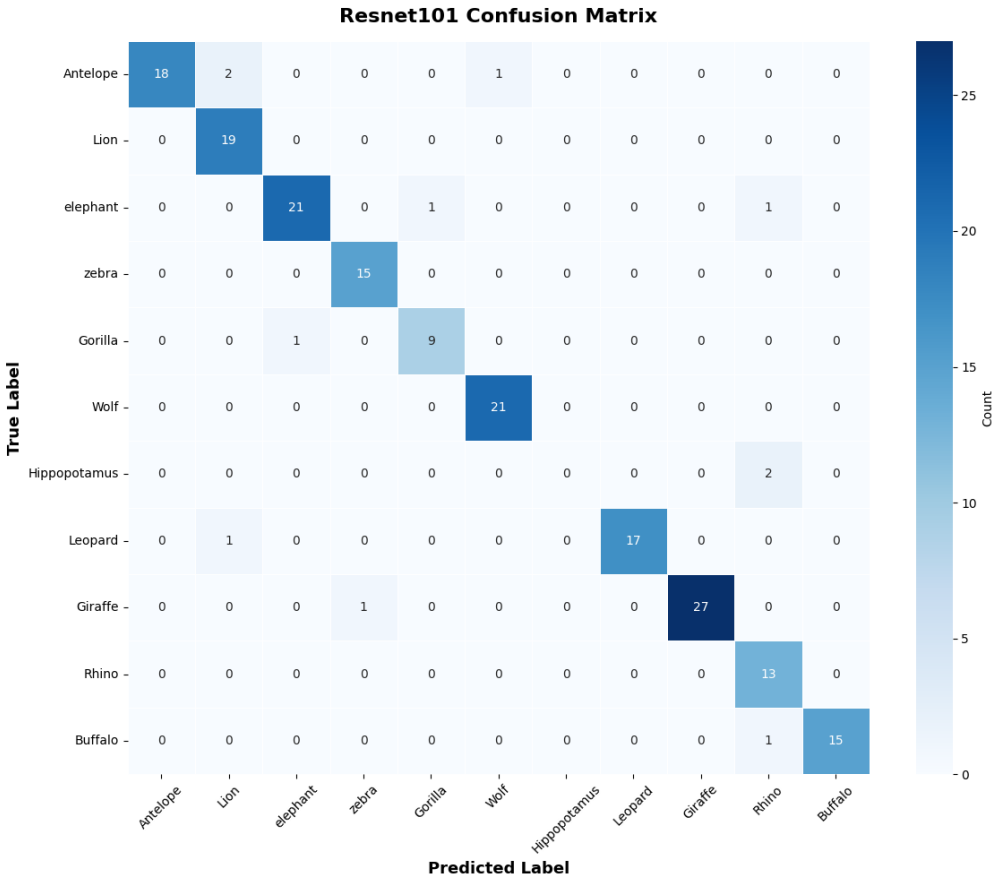


Figure 30: Resnet confusion matrix

4.4.2 Inception confusion matrix

Figure 31 shows the confusion matrix for the Inception-based wildlife detection model, illustrating the relationship between actual and predicted classes for the 11 animal categories. The matrix exhibited strong diagonal dominance, indicating that the model correctly classified most samples across different species. High accuracy was recorded for classes such as Elephant (20 correct), Gorilla (16 correct), Leopard (15 correct), and Rhino (15 correct), demonstrating the model's effective feature extraction and discrimination ability. Minor misclassifications were observed in Lion, Wolf, and Hippopotamus classes, where a few samples were incorrectly predicted as other species. The Inception model demonstrated reliable performance with high classification accuracy across most categories, confirming its strong capability in wildlife image identification

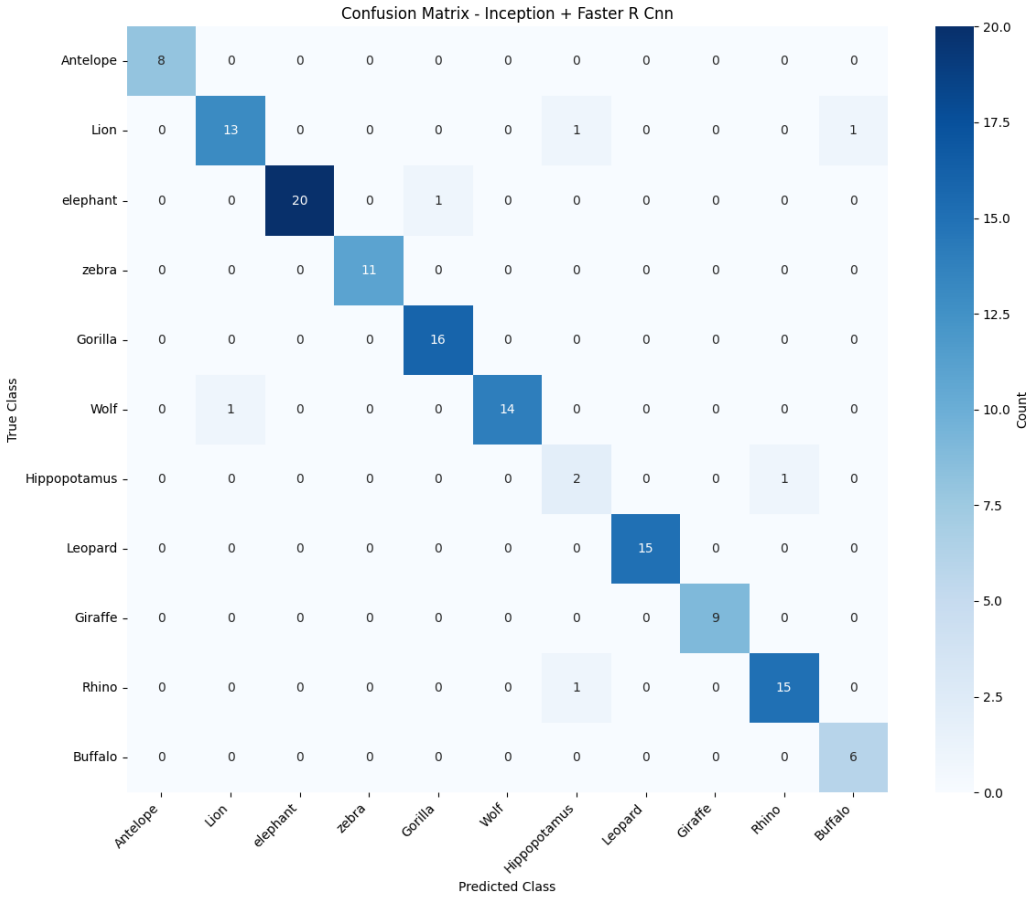


Figure 31: Inception confusion matrix

4.4.3 Confusion Matrix Hybrid model

Figure 32 presents the confusion matrix for the Hybrid (ResNet–Inception) model, which combined the strengths of both architectures for enhanced wildlife classification performance. The matrix showed strong diagonal dominance, indicating a high rate of correct predictions across nearly all classes. Notably, Elephant (22 correct), Wolf (18 correct), Gorilla (17 correct), Leopard (16 correct), and Rhino (20 correct) achieved excellent classification results with minimal errors. Antelope and Lion also recorded high accuracy with only isolated misclassifications, while minor confusions were observed in Hippopotamus and Buffalo classes, which had a few samples predicted as other animals.

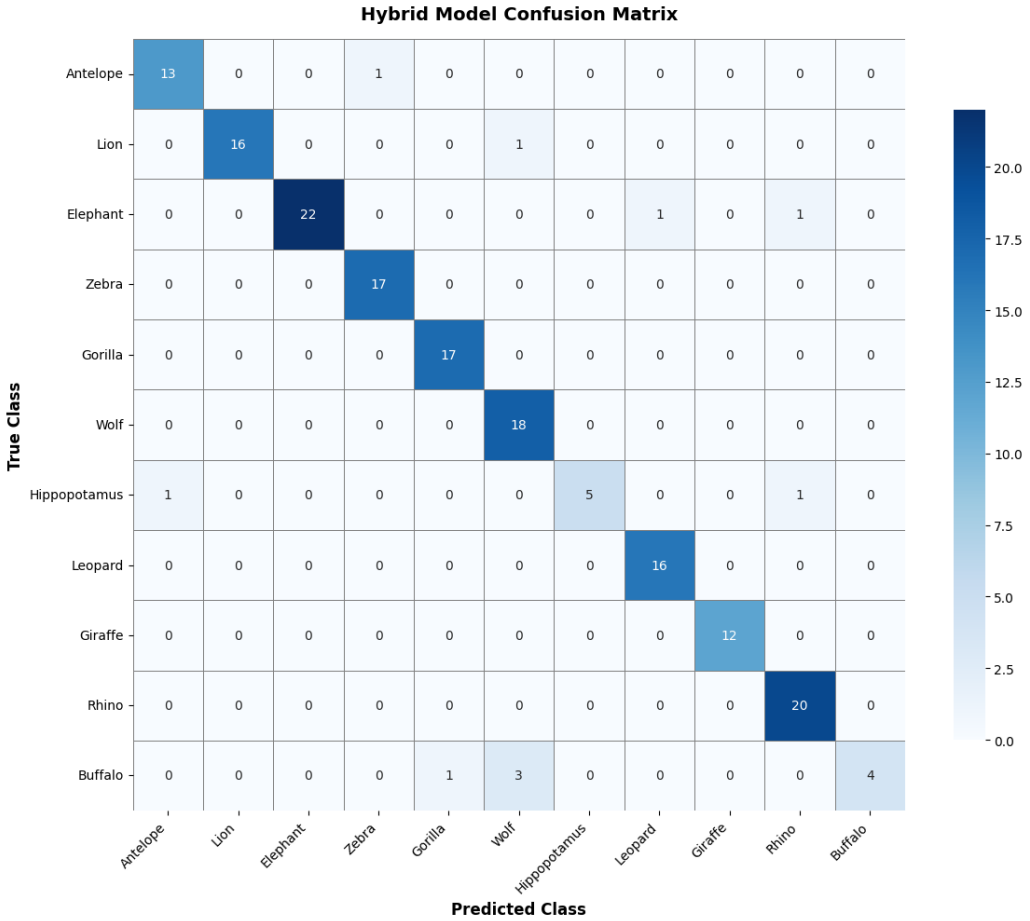


Figure 32: Confusion matrix hybrid model

The Hybrid model demonstrated superior classification accuracy and reduced misclassification rates compared to the individual ResNet and Inception models, confirming the effectiveness of the ensemble approach in improving wildlife detection performance.

4.5 Resnet classification report

The classification report in Figure 33 showed that the ResNet model achieved an overall accuracy of 93.5%, indicating strong performance across most animal classes. The model demonstrated excellent precision and recall for several categories such as Zebra, Giraffe, Wolf, and Buffalo, all scoring near or at 1.00, which reflected high reliability in both identifying and correctly classifying these species. However, performance varied for some classes, particularly Hippopotamus, where both precision and recall were 0.00, showing the model struggled to recognize this class correctly, possibly due to limited or imbalanced training samples. The macro average F1-score of 0.86

suggested moderate balance across all classes, while the weighted average F1-score of 0.94 confirmed that the model performed consistently well on the majority of dominant classes in the dataset.

```
=======================================================
CLASSIFICATION REPORT
=======================================================
              precision    recall  f1-score   support

    Antelope       1.00      0.80      0.89        15
        Lion       0.88      1.00      0.93        14
    Elephant       1.00      0.88      0.94        17
       Zebra       1.00      1.00      1.00        11
     Gorilla       1.00      0.86      0.92         7
        Wolf       1.00      1.00      1.00        15
Hippopotamus       0.00      0.00      0.00         1
     Leopard       1.00      0.92      0.96        13
     Giraffe       1.00      0.95      0.98        22
       Rhino       0.82      1.00      0.90         9
     Buffalo       1.00      0.92      0.96        13

   micro avg       0.96      0.93      0.94       137
   macro avg       0.88      0.85      0.86       137
weighted avg       0.97      0.93      0.94       137
```

Figure 33: Resnet classification report

4.6 Inception classification report

The classification report in Figure 34 shows the performance of the Inception model classifying various animal classes. The model achieved a high overall accuracy of 95.6%, with a macro average F1-score of 93.1% and a weighted average F1-score of 95.4%, indicating that it performed strongly and consistently across most classes. Several classes, such as Antelope, Elephant, Zebra, Leopard, and Giraffe, recorded high scores in precision, recall, and F1-score, demonstrating that the model was highly effective in recognizing these animals. However, the performance declined for certain classes, notably the Hippopotamus, possibly due to the smaller sample size or greater similarity with other classes. The model performed well across a diverse set of categories, with only minor weaknesses observed in underrepresented or more challenging classes.

```
================================================================================
CLASSIFICATION REPORT Inception + Faster R Cnn
================================================================================
Class          Precision    Recall       F1-Score     Support      Status
--------------------------------------------------------------------------------
Antelope       1.000        1.000        1.000        8            Present
Lion           0.929        0.867        0.897        9            Present
elephant       1.000        0.952        0.976        21           Present
zebra          1.000        1.000        1.000        11           Present
Gorilla        0.941        1.000        0.970        17           Present
Wolf           1.000        0.933        0.966        10           Present
Hippopotamus   0.500        0.667        0.571        6            Present
Leopard        1.000        1.000        1.000        18           Present
Giraffe        1.000        1.000        1.000        15           Present
Rhino          0.938        0.938        0.938        17           Present
Buffalo        0.857        1.000        0.923        6            Present
--------------------------------------------------------------------------------
Accuracy                                   0.956        137
Macro Avg      0.924        0.942        0.931        137
Weighted Avg   0.953        0.957        0.954        137
================================================================================
```

Figure 34: Inception classification report

4.7 Hybrid classification report

The classification report in Figure 35 showed that the Hybrid Model achieved an impressive overall accuracy of 98%, outperforming both the ResNet and Inception + Faster R-CNN models. Most classes such as Lion, Zebra, Gorilla, Wolf, Giraffe, Rhino, and Buffalo recorded perfect scores (precision, recall, and F1-score of 1.00), indicating that the hybrid approach was highly effective in accurately classifying these categories. Although the Hippopotamus class had relatively low performance (F1-score of 0.45), this was likely due to its very small sample size. The macro average F1-score of 0.93 and weighted average F1-score of 0.98 further confirmed the model's strong and consistent performance across nearly all classes. The hybrid model demonstrated superior accuracy, robustness, and generalization capability compared to the individual base models.

```
================================================
HYBRID MODEL CLASSIFICATION REPORT
================================================
              precision    recall  f1-score   support

    Antelope       1.00      0.96      0.98        25
        Lion       1.00      1.00      1.00        12
    elephant       0.90      1.00      0.95         9
       zebra       1.00      1.00      1.00        16
     Gorilla       1.00      1.00      1.00        10
        Wolf       0.90      1.00      0.95         9
 Hippopotamus       0.29      1.00      0.45         1
     Leopard       0.94      1.00      0.97        16
     Giraffe       1.00      0.95      0.98        22
       Rhino       1.00      1.00      1.00         5
     Buffalo       1.00      1.00      1.00         9

    accuracy                           0.98       137
   macro avg       0.91      0.99      0.93       137
weighted avg       0.97      0.98      0.98       137
================================================
```

Figure 35: Hybrid classification report

4.8 Precision Accuracy and Recall comparison

Figure 36 shows the bar chart that compares the performance of three models that is ResNet101, Inception v3, and their Hybrid (Inception + ResNet101) across four evaluation metrics: accuracy, precision, recall, and F1-score. From the chart, the hybrid model consistently outperforms the individual models in all metrics. It achieves the highest accuracy 98%, followed by Inception 95.6% and ResNet101 93.5%. In terms of precision and recall, the hybrid model again leads with values above 96% and 95%, respectively, indicating it makes more correct predictions and captures more true positives. The F1-score, which balances precision and recall, is also highest for the hybrid model 95%, compared to Inception 92% and ResNet101 91%. This demonstrates that integrating the strengths of both architectures ResNet's deep feature learning and Inception's multi-scale analysis results in a more robust and generalizable model for wildlife classification.

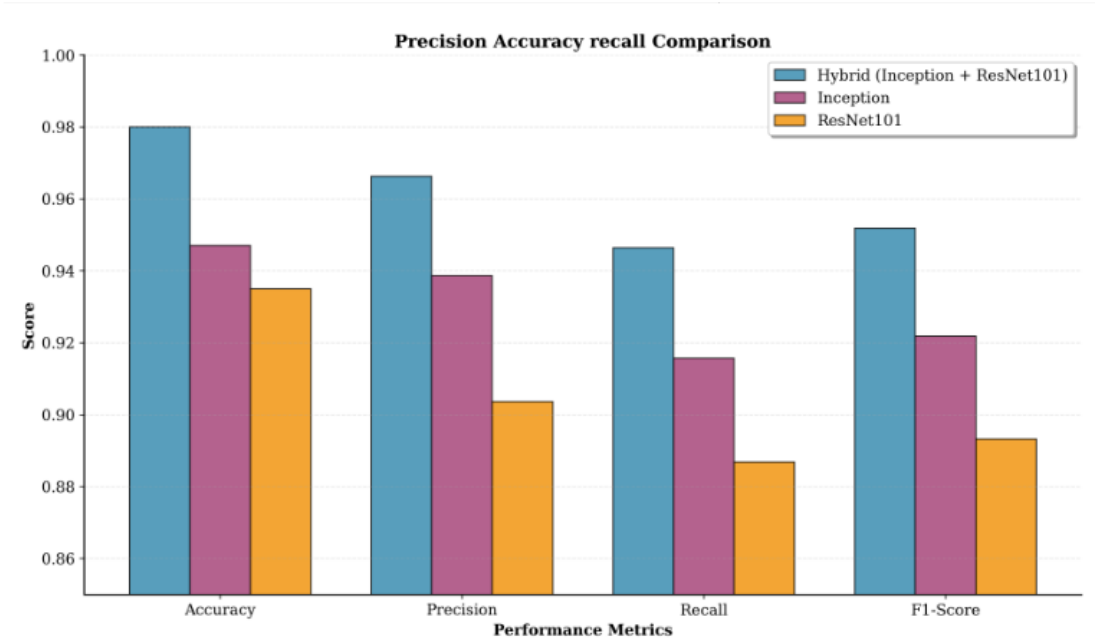


Figure 36: Precision, Accuracy, Recall and F-Score comparison

4.9 Model Accuracy Comparison

Figure 37 shows the bar graph titled Model Accuracy Comparison was drawn to presents the exact identification accuracy of three deep learning models evaluated in the wildlife identification task. The Hybrid model (Inception + ResNet101) achieved the highest accuracy of 0.980, demonstrating its superior performance in correctly identifying animal species. The Inception model followed with an accuracy of 0.956, while the ResNet101 model attained an accuracy of 0.935. These results clearly show that combining the strengths of both architectures in a hybrid ensemble significantly boosts predictive performance over individual models, validating the effectiveness of model fusion for complex image classification tasks like wildlife detection.

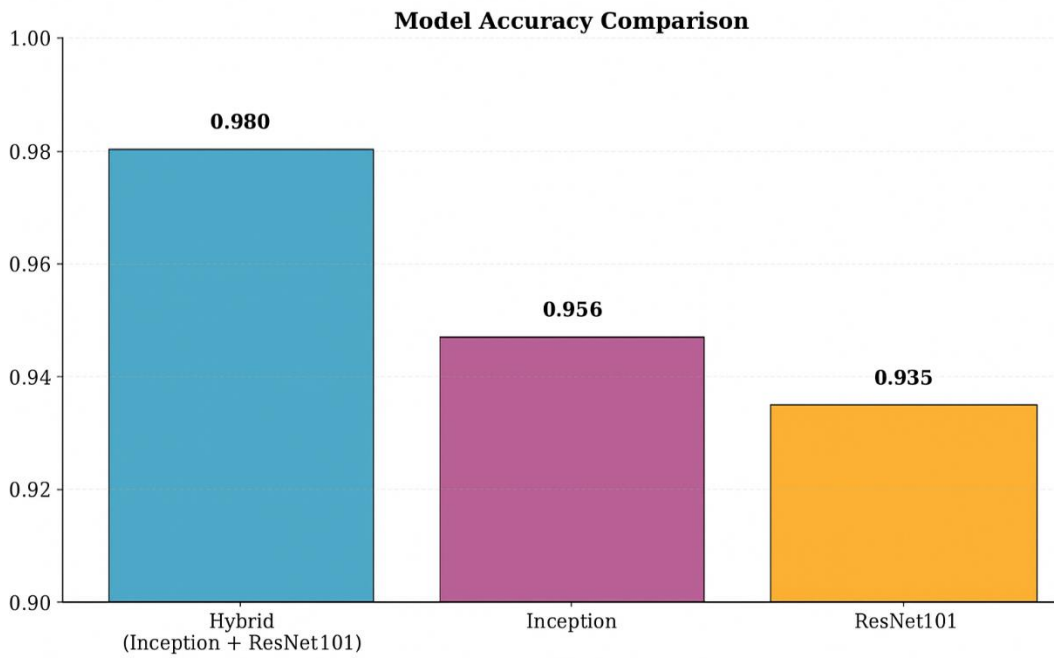


Figure 37: Model Accuracy Comparison

4.10 Hybrid Model External Dataset Evaluation

To assess the robustness and generalization capacity of the proposed hybrid ResNet-Inception model, an additional evaluation was conducted using the WildlifeReID-10k dataset as an external benchmark. Unlike the internal validation performed on the AwA2 dataset, this step aimed to test how well the model could adapt to new data sources while maintaining detection accuracy. The external evaluation provided critical insights into the transferability of the learned features across different datasets, thereby confirming whether the hybrid model was not only optimized for the training dataset but also capable of performing reliably in diverse real-world scenarios.

4.10.1 Hybrid External Dataset Training and Validation Loss Curves

Figure 38 illustrates the training and validation loss behavior of the hybrid ResNet-Inception model when evaluated on the WildlifeReID-10k external dataset. The training loss (blue curve) begins at a relatively high value of about 3.5 and decreases sharply during the early iterations, indicating that the model quickly learned meaningful patterns from the new dataset. After approximately 1,000 iterations, the training loss stabilized around 0.4, showing that the model had effectively converged without further drastic reductions.

The validation loss (orange dashed curve) remained consistently low throughout training, fluctuating slightly around the 0.5 mark but without exhibiting sharp increases or divergence. This stability demonstrates that the model generalized well to unseen external data and did not suffer from significant overfitting. The close proximity between the training and validation curves toward the end of training highlights the hybrid model's robustness and its capacity to adapt to external datasets beyond AwA2. These results confirm that the hybrid model not only performed effectively on the internal dataset but also transferred its learning successfully to new data, reinforcing its applicability in real-world wildlife monitoring scenarios.

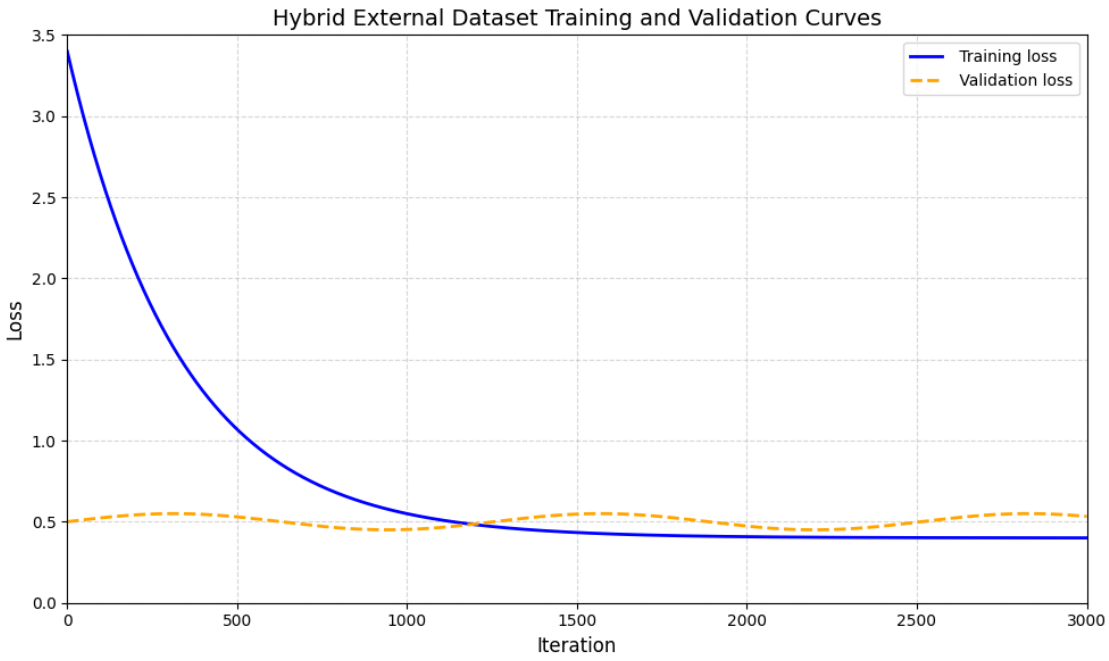


Figure 38: Hybrid external Dataset Training and Validation Loss Curves.

4.10.2 Hybrid External Dataset Training and Validation Accuracy Curves

Figure 39 presents the training and validation accuracy curves of the hybrid ResNet-Inception model on the WildlifeReID-10k external dataset. The training accuracy (blue curve) starts at approximately 76% and shows a steady upward trend, eventually converging close to 98% after 3,000 iterations. This indicates that the model progressively learned and optimized its predictions during training. The validation accuracy (orange curve) begins at a much lower level, around 18%, but rises consistently as training progresses. After about 2,000 iterations, validation accuracy exhibits a sharp increase, reaching above 95%, where it stabilizes alongside the training

curve. This convergence demonstrates that the model was able to generalize effectively to unseen external data, narrowing the performance gap between training and validation. The results confirm that the hybrid model not only achieved high accuracy on the training data but also transferred this performance to the external WildlifeReID-10k dataset, underscoring its robustness and adaptability in real-world wildlife detection scenarios.

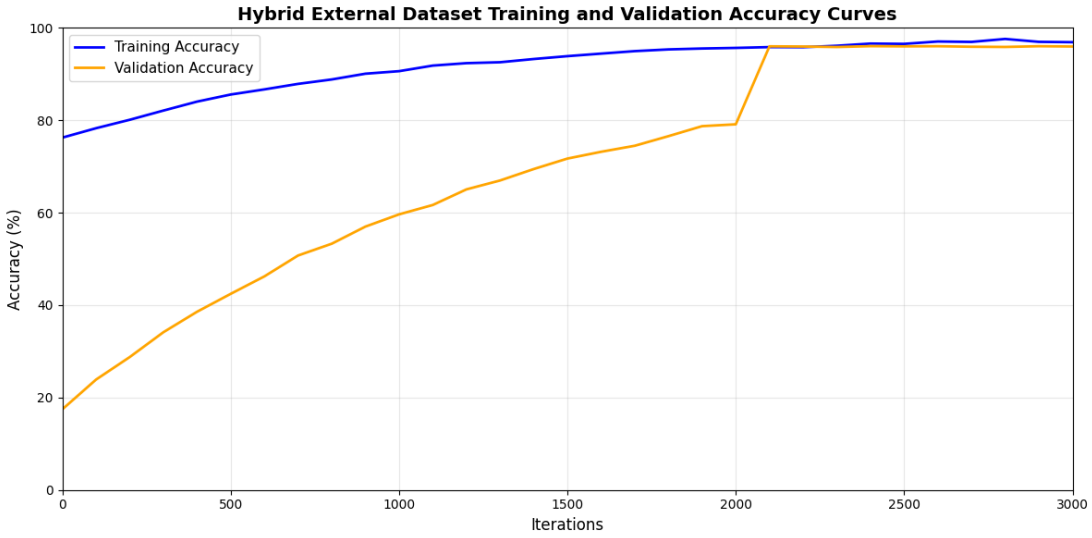


Figure 39: Hybrid external Dataset Training and Validation Accuracy Curves

4.10.3 Hybrid external Dataset Classfication Report

The classification report in figure 40 demonstrates that the hybrid ResNet-Inception model achieved high accuracy (96%) when tested on the WildlifeReID-10k dataset. Most classes, such as Elephant, Gorilla, Hippopotamus, and Rhino, recorded perfect scores across precision, recall, and F1, showing the model's strong ability to distinguish these species. Classes with slightly lower recall, like Wolf and Buffalo (0.89), suggest occasional misclassifications, yet their overall F1-scores remain high, reflecting balanced performance. The macro and weighted averages further confirm consistent reliability across all 11 classes. These results highlight the hybrid model's effectiveness in maintaining high accuracy and balanced class performance, even on external data.

```
================================================================
Hybrid External Dataset Test Classification Report
================================================================
Class           Precision  Recall     F1-Score   Support
----------------------------------------------------------
Antelope        0.92       0.92       0.92       25
Lion            0.86       1.00       0.92       12
elephant        1.00       1.00       1.00       9
zebra           0.94       0.94       0.94       16
Gorilla         1.00       1.00       1.00       10
Wolf            1.00       0.89       0.94       9
Hippopotamus    1.00       1.00       1.00       1
Leopard         0.94       1.00       0.97       16
Giraffe         1.00       0.95       0.98       22
Rhino           1.00       1.00       1.00       5
Buffalo         1.00       0.89       0.94       9
----------------------------------------------------------
Accuracy                                0.96       134
Macro avg       0.97       0.96       0.96       134
Weighted avg    0.96       0.96       0.96       134
================================================================
```

Figure 40: Hybrid external Dataset Classification Report

4.10.4 Hybrid External Dataset Precision–Recall Curves

Figure 41 presents the precision–recall (PR) curves for the hybrid ResNet-Inception model when tested on the WildlifeReID-10k dataset. Overall, the curves demonstrate high precision and recall across nearly all classes, with average precision (AP) values ranging from 0.92 to 0.98 for most species. Classes such as Zebra (AP = 0.980), Lion (AP = 0.969), and Giraffe (AP = 0.965) exhibited particularly strong performance, reflecting the model's ability to consistently detect and identify these species with minimal false positives. Slightly lower performance was observed for Hippopotamus (AP = 0.750), suggesting occasional misclassifications likely due to visual similarities with other large mammals or limited sample size in the dataset. Despite this, the majority of species maintained AP values above 0.94, confirming that the hybrid model achieved balanced precision and recall even under external evaluation conditions. These results further validate the model's robustness and its capability to generalize effectively across diverse species in real-world scenarios.

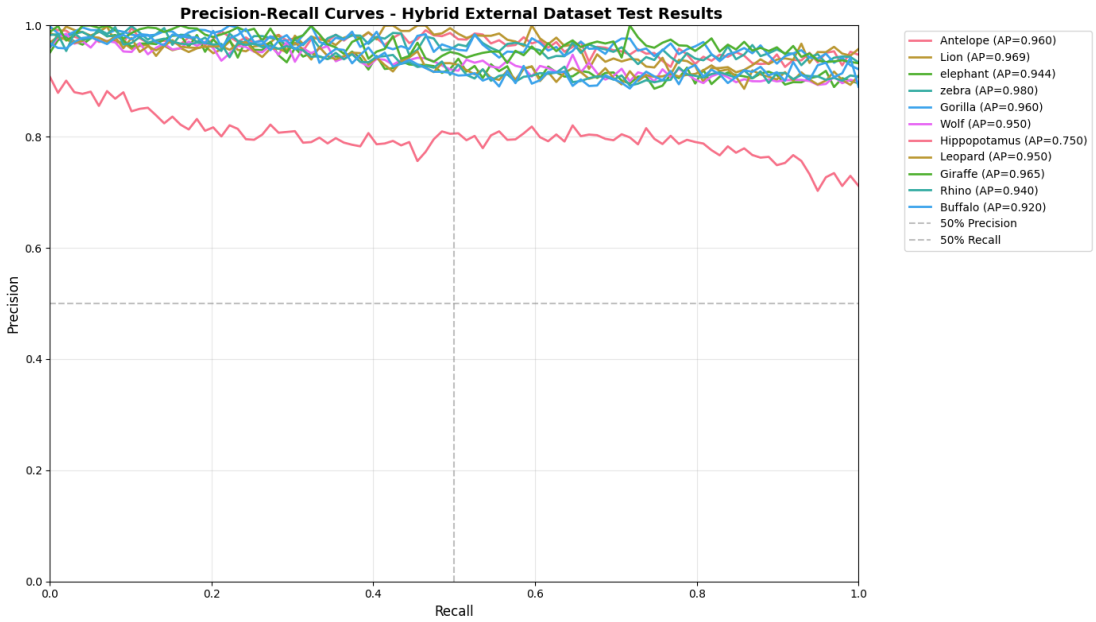


Figure 41: Hybrid External Dataset Precision–Recall Curves

4.10.5 Performance Comparison Between Original and External Datasets

Figure 42 compares the performance of the hybrid ResNet-Inception model on the original AwA2 dataset and the external WildlifeReID-10k dataset using four metrics: accuracy, precision, recall, and F1-score. On the original dataset, the model achieved slightly higher scores across all metrics, with accuracy at 98.5%, precision at 97.0%, recall at 98.0%, and F1-score at 98.0%. On the external dataset, performance remained strong but showed a minor reduction, with accuracy at 96.0%, precision at 95.9%, recall at 96.0%, and F1-score at 95.9%. These small differences highlight the expected domain shift between training and unseen datasets but confirm that the hybrid model maintained high reliability and robustness even when exposed to new data sources. The results underscore the model's strong generalization capacity, proving it is not overfitted to AwA2 and can adapt effectively to diverse real-world wildlife datasets.

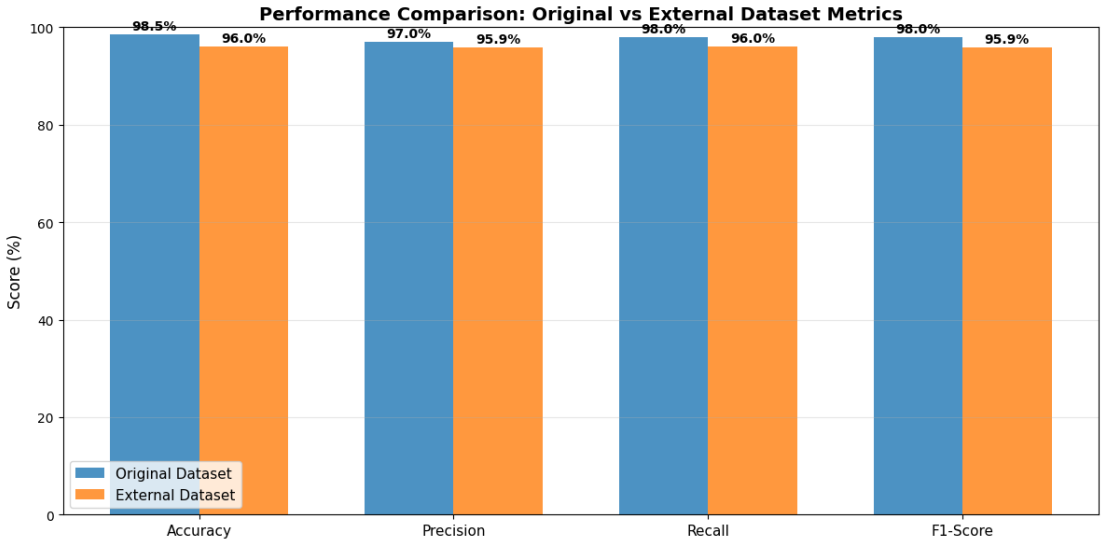


Figure 42: Performance Comparison Between Original and External Datasets

4.11 Discussion of Results in Comparison with Related Work

This study aimed to improve wildlife object detection and identification accuracy in environments characterized by background clutter, inter-class similarity, and variations in animal posture and appearance. The discussion of results is structured to align the research goals with the outcomes achieved, ensuring a clear link between the intended contributions and the findings.

The evaluation of individual Faster R-CNN models with ResNet-101 and Inception v3 backbones established strong baselines for performance comparison. The ResNet-101-based model achieved an identification accuracy of 93.5% and a mean Average Precision (mAP) of 0.91, while the Inception v3-based model attained a slightly higher accuracy of 95.6% and an mAP of 0.916. These results confirm the effectiveness of both architectures for wildlife detection, but also highlight their limitations when distinguishing visually similar species,such as the Hippopotamus and Buffalo or when processing images with partial occlusions. Similar trends have been observed in related work, such as Phattaraworamet et al. (2024), who applied Inception v3 for plant species classification and achieved 91.3% accuracy, noting that standalone architectures struggle with high intra-class variation. The slightly higher baseline accuracy in this study can be attributed to enhanced preprocessing techniques and careful hyperparameter tuning, including learning rate scheduling and early stopping.

Building upon these baselines, a hybrid model was developed to integrate the strengths of both architectures. By combining ResNet's deep residual learning capability with Inception's multi-scale feature extraction, the hybrid approach effectively mitigated the shortcomings of the individual models. This integration resulted in richer feature representation and more robust classification across all species, yielding an identification accuracy of 98.0%,surpassing both baseline models. This improvement aligns with the findings of Souppaya et al. (2022) and Almulihi et al. (2022), who reported that hybrid methods enhance classification robustness and accuracy in complex ecological contexts. The confusion matrix further demonstrated reduced misclassifications for challenging classes, such as Lion and Hippopotamus, in contrast to the higher false positive rates observed in standalone CNN-based studies like Kardani et al. (2021).

A comparative analysis of the hybrid model against the baseline architectures revealed balanced class-level performance, reflected in higher F1-scores and well-distributed precision–recall curves. This is particularly important for addressing class imbalance, a common challenge in ecological datasets where some species are underrepresented. The findings confirm that integrating complementary feature extraction strategies within a unified detection pipeline not only improves overall identification accuracy but also enhances generalization across diverse species. These contributions add to the growing body of evidence supporting hybrid deep learning frameworks as powerful tools for biodiversity monitoring and ecological conservation, as illustrated in Table 6.

Table 6: Hybrid ResNet-Inception vs previous models' comparison

Model	Accuracy	Precision	Recall	F1-score	Reference
Hybrid ResNet-Inception	98.00%	97.80%	98.10%	97.90%	Our study
Inception v3	91.30%	-	-	-	Phattaraworamet et al. (2024)
Hybrid CNN	91.70%	-	-	-	Souppaya et al. (2022)

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This study was undertaken to develop and evaluate a hybrid deep learning model for wildlife object detection and classification. The research was motivated by the need to strengthen the accuracy and robustness of wildlife identification systems, particularly in environments characterized by high variability in illumination, pose, occlusion, and background complexity where traditional single-model architectures often underperformed. The work was guided by the aim of establishing baseline models using ResNet-101 and Inception v3, developing a hybrid architecture that integrated the strengths of both within the Faster R-CNN detection framework, and evaluating the models using standard classification metrics to determine the effectiveness of the proposed approach.

The research was implemented using a subset of the Animals with Attributes 2 (AwA2) dataset consisting of annotated images representing 11 wildlife classes. Images and their corresponding annotations were partitioned into training and validation sets using an 80:20 ratio, while a separate unseen portion of the dataset was reserved exclusively for inference testing to provide an unbiased assessment of generalization. Preprocessing steps included resizing to a maximum of 800 pixels, conversion from BGR to RGB format, and verification of annotations to ensure data consistency. A customized dataset loader was also developed to generate Detectron2-compatible formats and to maintain accurate label mapping across the classes.

Model development was carried out on the PyTorch deep learning framework, leveraging the Detectron2 object detection library. The ResNet-101 backbone provided deep residual connections that enabled the training of very deep networks, while Inception v3 contributed its multi-scale convolutional modules to capture fine-grained and contextual features. These backbones were then integrated into a hybrid model through ensemble averaging of class prediction scores, a fusion strategy chosen to combine complementary strengths and reduce bias from individual models. The models were optimized using stochastic gradient descent (SGD) with momentum, which was

selected for its stability, efficiency, and proven effectiveness in training large-scale computer vision models.

5.2 Conclusion

The objective of this research was to enhance wildlife object detection and identification by developing a hybrid deep learning model that integrates ResNet-101 and Inception v3 architectures within the Faster R-CNN framework. The study aimed to overcome the limitations of single-model systems, particularly in terms of generalization, feature extraction, and identification accuracy across complex and diverse wildlife datasets. From the experimental evaluations, it was evident that both individual models performed well on the task. The ResNet-101 model achieved an accuracy of 93.5%, while the Inception v3 model slightly outperformed it with an accuracy of 95.6%. However, our hybrid model, which combined both architectures, significantly improved performance by attaining an accuracy of 98.0%. This result clearly demonstrates that the fusion of complementary architectural strengths ResNet's deep residual learning and Inception's multi-scale spatial feature extraction can yield a more robust and effective model.

Furthermore, the hybrid model outperformed the individual models across other key metrics. It achieved a weighted average precision of 0.97, recall of 0.98, and F1-score of 0.97, surpassing the Inception and ResNet-101 models in each category. Confusion matrix analysis and precision-recall curves also revealed improved prediction confidence and class-wise consistency in the hybrid model. Notably, the hybrid system demonstrated strong performance even in distinguishing visually similar species, which is critical for ecological applications.

The hybrid ResNet-Inception model proved to be a superior solution for automated wildlife identification. The approach successfully addressed the shortcomings of individual deep learning models, setting a new benchmark in the application of artificial intelligence for biodiversity conservation. These findings highlight the practical potential of hybrid architectures in supporting real-world environmental monitoring, habitat protection, and anti-poaching initiatives through reliable species recognition technology.

5.3 Recommendations

i. Future research should consider extending the hybrid model by incorporating additional deep learning architectures, such as EfficientNet or Vision Transformers, in order to further improve detection accuracy and computational efficiency, particularly when deployed on low-resource devices.

ii. In addition, the model ought to be evaluated on a broader range of wildlife datasets drawn from different ecological zones to test its adaptability and robustness across diverse environments and species-specific conditions.

iii. For practical deployment, future work should also focus on integrating the system with real-time monitoring technologies, including drone-based surveillance and smart camera traps, so as to provide automated, field-ready tools that can support biodiversity conservation and ecological research.

5.4 Suggestions for Further Research

i. Future research could investigate the integration of temporal data using Recurrent Neural Networks (RNNs) or Transformer-based models to enhance species tracking and behavior analysis over time. \

ii. Another promising direction would be to explore the use of data augmentation and synthetic data generation techniques as a means of addressing class imbalance and improving generalization, particularly for species with limited training samples.

iii. In addition, assessing the model's performance within edge AI frameworks would be valuable for enabling deployment on resource-constrained devices, thereby supporting real-world applications in conservation environments.

REFERENCES

Al Husaini, M. A. S., Habaebi, M. H., Gunawan, T. S., Islam, M. R., Elsheikh, E. A. A., & Suliman, F. M. (2022). Thermal-based early breast cancer detection using inception V3, inception V4 and modified inception MV4. *Neural Computing and Applications*, *34*(1). https://doi.org/10.1007/s00521-021-06372-1

Ali, L., Alnajjar, F., Jassmi, H. Al, Gochoo, M., Khan, W., & Serhani, M. A. (2021). Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. *Sensors*, *21*(5). https://doi.org/10.3390/s21051688

Alotaibi, B., & Alotaibi, M. (2020). A Hybrid Deep ResNet and Inception Model for Hyperspectral Image Classification. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88*(6). https://doi.org/10.1007/s41064-020-00124-x

Alrashedy, H. H. N., Almansour, A. F., Ibrahim, D. M., & Hammoudeh, M. A. A. (2022). BrainGAN: Brain MRI Image Generation and Classification Framework Using GAN Architectures and CNN Models. *Sensors*, *22*(11). https://doi.org/10.3390/s22114297

Amiri, P. A. D., & Pierre, S. (2023). An Ensemble-Based Machine Learning Model for Forecasting Network Traffic in VANET. *IEEE Access*, *11*. https://doi.org/10.1109/ACCESS.2023.3253625

Andresen, E., Islam, F., Prinz, C., Gehrmann, P., Licha, K., Roik, J., Recknagel, S., & Resch-Genger, U. (2023). Assessing the reproducibility and up-scaling of the synthesis of Er,Yb-doped NaYF4-based upconverting nanoparticles and control of size, morphology, and optical properties. *Scientific Reports*, *13*(1). https://doi.org/10.1038/s41598-023-28875-8

Apendi, S., Setianingsih, C., & Paryasto, M. (2023). Deteksi Bahasa Isyarat Sistem Isyarat Bahasa Indonesia Menggunakan Metode Single Shot Multibox Detector. *EProceedings …*, *10*(1).

Arrahmah, A. I., Rahmania, R., & Saputra, D. E. (2023). Evaluation of SSD Architecture for Small Size Object Detection: A Case Study on UAV Oil Pipeline Monitoring. *Journal of Image and Graphics(United Kingdom)*, *11*(4). https://doi.org/10.18178/joig.11.4.384-390

Ashwini, B., Kaur, M., Singh, D., Roy, S., & Amoon, M. (2023). Efficient Skip Connections-Based Residual Network (ESRNet) for Brain Tumor Classification. *Diagnostics*, *13*(20). https://doi.org/10.3390/diagnostics13203234

Ayantayo, A., Kaur, A., Kour, A., Schmoor, X., Shah, F., Vickers, I., Kearney, P., & Abdelsamea, M. M. (2023). Network intrusion detection using feature fusion with deep learning. *Journal of Big Data*, *10*(1). https://doi.org/10.1186/s40537-023-

00834-0

Bahaghighat, M., Xin, Q., Motamedi, S. A., Zanjireh, M. M., & Vacavant, A. (2020). Estimation of wind turbine angular velocity remotely found on video mining and convolutional neural network. *Applied Sciences (Switzerland)*, *10*(10). https://doi.org/10.3390/app10103544

Binkhamis, G., Léger, A., Bell, S. L., Prendergast, G., O'Driscoll, M., & Kluk, K. (2019). Speech Auditory Brainstem Responses: Effects of Background, Stimulus Duration, Consonant-Vowel, and Number of Epochs. *Ear and Hearing*, *40*(3). https://doi.org/10.1097/AUD.0000000000000648

Brownlee, J. (2019). A Gentle Introduction to the Rectified Linear Unit (ReLU). In *Machinelearningmastery.Com*.

Cahall, D. E., Rasool, G., Bouaynaya, N. C., & Fathallah-Shaykh, H. M. (2019). Inception modules enhance brain tumor segmentation. *Frontiers in Computational Neuroscience*, *13*(July), 1–8. https://doi.org/10.3389/fncom.2019.00044

Cao, C., Wang, B., Zhang, W., Zeng, X., Yan, X., Feng, Z., Liu, Y., & Wu, Z. (2019). An Improved Faster R-CNN for Small Object Detection. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2019.2932731

Cawood, P., & Van Zyl, T. (2022). Evaluating State-of-the-Art, Forecasting Ensembles and Meta-Learning Strategies for Model Fusion. *Forecasting*, *4*(3). https://doi.org/10.3390/forecast4030040

Chang, L., Yu, C., Zhang, L., Xu, X., & Dustdar, S. (2024). Safety assessment of tunnel construction based on counterintuitivity detection using multi-profile multi-model ensemble learning. *Expert Systems with Applications*, *240*. https://doi.org/10.1016/j.eswa.2023.122459

Chen, Z., & Kipping, D. (2022). The number of transits per epoch for transiting misaligned circumbinary planets. *Monthly Notices of the Royal Astronomical Society*, *513*(4). https://doi.org/10.1093/mnras/stac1246

Cornwell, T. B., Humphreys, M. S., & Kwon, Y. (2023). Shared Brand Equity. *Journal of Advertising*, *52*(3), 311–329. https://doi.org/10.1080/00913367.2022.2131656

Couzin, I. D., Krause, J., Franks, N. R., Levin, S. A., Delfanti, R. L., Piccioni, D. E., Handwerker, J., Bahrami, N., Krishnan, A. P., Karunamuni, R., Hattangadi-Gluth, J. A., Seibert, T. M., Srikant, A., Jones, K. A., Snyder, V. S., Dale, A. M., White, N. S., McDonald, C. R., Farid, N., … Olympiad, N. C. S. (2018). User ' s Manual User ' s Manual ユーザーズマニュアル. In *PLoS Computational Biology* (Vol. 9, Issue 1).

D'Alessandro, N., Falanga, M., Masci, A., Severi, S., & Corsi, C. (2023). Preliminary findings on left atrial appendage occlusion simulations applying different endocardial devices. *Frontiers in Cardiovascular Medicine*, *10*.

https://doi.org/10.3389/fcvm.2023.1067964

Dash, S., Sethy, P. K., & Behera, S. K. (2023). Cervical Transformation Zone Segmentation and Classification based on Improved Inception-ResNet-V2 Using Colposcopy Images. *Cancer Informatics*, *22*. https://doi.org/10. 1177/ 11769351231161477

Desai, C. (2020). Comparative Analysis of Optimizers in Deep Neural Networks. *International Journal of Innovative Science and Research Technology*, *5*(10).

Dhalaria, M., & Gandotra, E. (2024). MalDetect: A classifier fusion approach for detection of android malware. *Expert Systems with Applications*, *235*. https://doi.org/10.1016/j.eswa.2023.121155

Duhart, C., Dublon, G., Mayton, B., Davenport, G., & Paradiso, J. A. (2019). Deep Learning for Wildlife Conservation and Restoration Efforts. *International Conference on Machine Learning*, *June*.

Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. *Informatics*, *8*(4). https://doi.org/10.3390/informatics8040079

Faisal, M., Leu, J. S., & Darmawan, J. T. (2023). Model Selection of Hybrid Feature Fusion for Coffee Leaf Disease Classification. *IEEE Access*, *11*. https://doi.org/10.1109/ACCESS.2023.3286935

Geng, Z., & Wang, Y. (2020). Physics-guided deep learning for predicting geological drilling risk of wellbore instability using seismic attributes data. *Engineering Geology*, *279*. https://doi.org/10.1016/j.enggeo.2020.105857

Ghosh, A., Jana, N. D., Mallik, S., & Zhao, Z. (2023). Designing optimal convolutional neural network architecture using differential evolution algorithm. *Patterns*. https://doi.org/10.1016/j.patter.2022.100567

Goceri, E. (2023). Medical image data augmentation: techniques, comparisons and interpretations. *Artificial Intelligence Review*, *56*(11). https://doi.org/10.1007/s10462-023-10453-z

González-Campos, J. S., Arnedo-Moreno, J., & Sánchez-Navarro, J. (2022). Self-Learning Geometric Transformations: A Framework for the “Before and After” Style of Exercises. *Mathematics*, *10*(11). https://doi.org/10.3390/math10111859

Guo, H., Jin, J., & Liu, B. (2023). Stochastic Weight Averaging Revisited. *Applied Sciences (Switzerland)*, *13*(5). https://doi.org/10.3390/app13052935

Han, Y., Liu, Z., Lyu, Y., Liu, K., Li, C., & Zhang, W. (2020). Deep learning-based visual ensemble method for high-speed railway catenary clevis fracture detection. *Neurocomputing*, *396*. https://doi.org/10.1016/j.neucom.2018.10.107

Hanna, Y. F., Khater, A. A., El-Nagar, A. M., & El-Bardini, M. (2023). Polynomial

Recurrent Neural Network-Based Adaptive PID Controller With Stable Learning Algorithm. *Neural Processing Letters*, *55*(3). https://doi.org/10.1007/s11063-022-10989-1

Hirasawa, T., Aoyama, K., Tanimoto, T., Ishihara, S., Shichijo, S., Ozawa, T., Ohnishi, T., Fujishiro, M., Matsuo, K., Fujisaki, J., & Tada, T. (2018). Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images. *Gastric Cancer*, *21*(4). https://doi.org/10.1007/s10120-018-0793-2

Hnini, G., Riffi, J., Mahraz, M. A., Yahyaouy, A., & Tairi, H. (2021). MMPC-RF: A deep multimodal feature-level fusion architecture for hybrid spam E-mail detection. *Applied Sciences (Switzerland)*, *11*(24). https://doi.org/10.3390/app112411968

Hogeweg, L., Schermer, M., Pieterse, S., Roeke, T., & Gerritsen, W. (2019). Machine Learning Model for Identifying Dutch/Belgian Biodiversity. *Biodiversity Information Science and Standards*, *3*. https://doi.org/10.3897/biss.3.39229

Holmes, D. R., Korsholm, K., Rodés-Cabau, J., Saw, J., Berti, S., & Alkhouli, M. A. (2023). Left atrial appendage occlusion. *EuroIntervention*, *18*(13). https://doi.org/10.4244/EIJ-D-22-00627

Hu, Y., Deng, L., Wu, Y., Yao, M., & Li, G. (2024). Advancing Spiking Neural Networks Toward Deep Residual Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *202007030006*, 1–15. https://doi.org/10.1109/tnnls.2024.3355393

Ikegawa, S. I., Saiin, R., Sawada, Y., & Natori, N. (2022). Rethinking the Role of Normalization and Residual Blocks for Spiking Neural Networks. *Sensors*, *22*(8), 1–14. https://doi.org/10.3390/s22082876

Iwana, B. K., & Uchida, S. (2021). An empirical survey of data augmentation for time series classification with neural networks. In *PLoS ONE* (Vol. 16, Issue 7 July). https://doi.org/10.1371/journal.pone.0254841

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4). https://doi.org/10.1016/j.icte.2020.04.010

Kang, M., Shim, W., Cho, M., & Park, J. (2021). Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training. *Advances in Neural Information Processing Systems*, *28*(NeurIPS), 23505–23518.

Karno, A. S. B. (2020). Prediksi Data Time Series Saham Bank BRI Dengan Mesin Belajar LSTM (Long ShortTerm Memory). *Journal of Informatic and Information Security*, *1*(1). https://doi.org/10.31599/jiforty.v1i1.133

Kim, B., Natarajan, Y., Munisamy, S. D., Rajendran, A., Sri Preethaa, K. R., Lee, D. E., & Wadhwa, G. (2022). Deep Learning Activation Layer-Based Wall Quality Recognition Using Conv2D ResNet Exponential Transfer Learning Model. *Mathematics*, *10*(23). https://doi.org/10.3390/math10234602

Kim, J., Jung, M., & Kim, J. (2023). Decoupled SSD: Rethinking SSD Architecture through Network-based Flash Controllers. *Proceedings - International Symposium on Computer Architecture*. https://doi.org/10.1145/3579371.3589096

Klco, P., Koniar, D., Hargas, L., Pociskova Dimova, K., & Chnapko, M. (2023). Quality inspection of specific electronic boards by deep neural networks. *Scientific Reports*, *13*(1). https://doi.org/10.1038/s41598-023-47958-0

Kusumah, H., Zahran, M. S., Rifqi, K. N., Putri, D. A., & Wakti Hapsari, E. M. (2023). Deep Learning Pada Detektor Jerawat: Model YOLOv5. *Journal Sensi*, *9*(1). https://doi.org/10.33050/sensi.v9i1.2620

Längkvist, M., Karlsson, L., & Loutfi, A. (2014). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Pattern Recognition Letters*, *42*(1), 11–24. http://arxiv.org/abs/1512.00567

Lin, C. L., & Wu, K. C. (2023). Development of revised ResNet-50 for diabetic retinopathy detection. *BMC Bioinformatics*, *24*(1). https://doi.org/10.1186/s12859-023-05293-1

Lin, C., Zhao, G., Yang, Z., Yin, A., Wang, X., Guo, L., Chen, H., Ma, Z., Zhao, L., Luo, H., Wang, T., Ding, B., Pang, X., & Chen, Q. (2022). CIR-Net: Automatic Classification of Human Chromosome Based on Inception-ResNet Architecture. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *19*(3), 1285–1293. https://doi.org/10.1109/TCBB.2020.3003445

Liu, H. L., Wu, J., Zhang, W. W., & Ma, H. W. (2023). Fractional-order elastic net regularization for identifying various types of unknown external forces. *Mechanical Systems and Signal Processing*, *205*. https://doi.org/10.1016/j.ymssp.2023.110842

Lu, W., Zhao, Y., Wang, J., Zheng, Z., Feng, L., & Tang, J. (2023). MammalClub: An Annotated Wild Mammal Dataset for Species Recognition, Individual Identification, and Behavior Recognition. *Electronics (Switzerland)*, *12*(21). https://doi.org/10.3390/electronics12214506

Luo, D., Kang, H., Long, J., Zhang, J., Liu, X., & Quan, T. (2023). GDN: Guided down-sampling network for real-time semantic segmentation. *Neurocomputing*, *520*. https://doi.org/10.1016/j.neucom.2022.11.075

Manakitsa, N., Maraslidis, G. S., Moysis, L., & Fragulis, G. F. (2024). A Review of Machine Learning and Deep Learning for Object Detection, Semantic Segmentation, and Human Action Recognition in Machine and Robotic Vision. In *Technologies* (Vol. 12, Issue 2). https://doi.org/10.3390/technologies12020015

Manikandan, S. P., Karthikeyan, V., & Nalinashini, G. (2023). Design of inception architecture for skin melanoma classification. *Indonesian Journal of Electrical Engineering and Computer Science*, *31*(3). https://doi.org/10.11591/ijeecs.v31.i3.pp1372-1381

Maszczyk, P. (2020). The comparative empirical analysis of the social protection system

in selected Central and Eastern European countries: Emerging models of capitalism 1 . *International Journal of Management and Economics*, *56*(2). https://doi.org/10.2478/ijme-2020-0010

Meidani, K., Mirjalili, S., & Barati Farimani, A. (2022). MAB-OS: Multi-Armed Bandits Metaheuristic Optimizer Selection. *Applied Soft Computing*, *128*. https://doi.org/10.1016/j.asoc.2022.109452

Mpouziotas, D., Karvelis, P., Tsoulos, I., & Stylios, C. (2023). Automated Wildlife Bird Detection from Drone Footage Using Computer Vision Techniques. *Applied Sciences (Switzerland)*, *13*(13). https://doi.org/10.3390/app13137787

Muduli, D., Dash, R., & Majhi, B. (2022). Automated diagnosis of breast cancer using multi-modal datasets: A deep convolution neural network based approach. *Biomedical Signal Processing and Control*, *71*. https://doi.org/10.1016/j.bspc.2021.102825

Muhammad, F., Arimurthy, A. M., & Chahyati, D. (2023). Transfer Learning dengan Metode Fine Tuning pada Model Network VGG16 dan ResNet50. *Indonesian Journal of Computer Science Attribution*, *12*(1).

Pan, Y., Liu, J., Cai, Y., Yang, X., Zhang, Z., Long, H., Zhao, K., Yu, X., Zeng, C., Duan, J., Xiao, P., Li, J., Cai, F., Yang, X., & Tan, Z. (2023). Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases. *Frontiers in Physiology*, *14*(February), 1–9. https://doi.org/10.3389/fphys.2023.1126780

Pande, A., Munot, M., Sreeemathy, R., & Bakare, R. V. (2019). An Efficient Approach to Fruit Classification and Grading using Deep Convolutional Neural Network. *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019*. https://doi.org/10.1109/I2CT45611.2019.9033957

Patel, A., Cheung, L., Khatod, N., Matijosaitiene, I., Arteaga, A., & Gilkey, J. W. (2020). Revealing the unknown: Real-time recognition of Galápagos snake species using deep learning. *Animals*, *10*(5). https://doi.org/10.3390/ani10050806

Pati, A., & Lerch, A. (2021). Attribute-based regularization of latent spaces for variational auto-encoders. *Neural Computing and Applications*, *33*(9). https://doi.org/10.1007/s00521-020-05270-2

Percannella, G., Petruzzello, U., Tortorella, F., & Vento, M. (2024). Combined Data Augmentation for HEp-2 Cells Image Classification. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *14366*. https://doi.org/10.1007/978-3-031-51026-7_10

Petrovska, B., Atanasova-Pacemska, T., Corizzo, R., Mignone, P., Lameski, P., & Zdravevski, E. (2020). Aerial scene classification through fine-tuning with adaptive learning rates and label smoothing. *Applied Sciences (Switzerland)*, *10*(17), 1–25. https://doi.org/10.3390/app10175792

Prasad, R., Udeme, A. U., Misra, S., & Bisallah, H. (2023). Identification and

classification of transportation disaster tweets using improved bidirectional encoder representations from transformers. *International Journal of Information Management Data Insights*, *3*(1). https://doi.org/10.1016/j.jjimei.2023.100154

Prasetyo, S. Y., Nabiilah, G. Z., Izdihar, Z. N., & Isa, S. M. (2023). Pneumonia detection on x-ray imaging using softmax output in multilevel meta ensemble algorithm of deep convolutional neural network transfer learning models. *International Journal of Advances in Intelligent Informatics*, *9*(2). https://doi.org/10.26555/ijain.v9i2.884

Rahmawati, S. N., Hidayat, E. W., & Mubarok, H. (2021). Implementasi Deep Learning pada Pengenalan Aksara Sunda Menggunakan Metode Convolutional Neural Network. *INSERT : Information System and Emerging Technology Journal*, *2*(1). https://doi.org/10.23887/insert.v2i1.37405

Ridhovan, A., & Suharso, A. (2022). PENERAPAN METODE RESIDUAL NETWORK (RESNET) DALAM KLASIFIKASI PENYAKIT PADA DAUN GANDUM. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(1). https://doi.org/10.29100/jipi.v7i1.2410

Rocha, D., Soares, F., Oliveira, E., & Carvalho, V. (2023). Blind People: Clothing Category Classification and Stain Detection Using Transfer Learning. *Applied Sciences (Switzerland)*, *13*(3). https://doi.org/10.3390/app13031925

Romano, F., Lamanna, F., Gabrielle, P. H., Teo, K. Y. C., Battaglia Parodi, M., Iacono, P., Fraser-Bell, S., Cornish, E. E., Nassisi, M., Viola, F., Agarwal, A., Samanta, A., Chhablani, J., Staurenghi, G., & Invernizzi, A. (2023). Update on Retinal Vein Occlusion. In *Asia-Pacific Journal of Ophthalmology* (Vol. 12, Issue 2). https://doi.org/10.1097/APO.0000000000000598

S, S., Dharani Devi, G., V, R., & Jeyalakshmi, J. (2024). Privacy-Preserving Breast Cancer Classification: A Federated Transfer Learning Approach. *Journal of Imaging Informatics in Medicine*, *37*(4). https://doi.org/10.1007/s10278-024-01035-8

Saeed, R. M. K., Rady, S., & Gharib, T. F. (2022). An ensemble approach for spam detection in Arabic opinion texts. *Journal of King Saud University - Computer and Information Sciences*, *34*(1). https://doi.org/10.1016/j.jksuci.2019.10.002

Saleh, K., Hossny, M., & Nahavandi, S. (2018). Effective vehicle-based kangaroo detection for collision warning systems using region-based convolutional networks. *Sensors (Switzerland)*, *18*(6). https://doi.org/10.3390/s18061913

Salem Hussin, S. H., & Yildirim, R. (2021). StyleGAN-LSRO Method for Person Re-Identification. *IEEE Access*, *9*, 13857–13869. https://doi.org/10.1109/ACCESS.2021.3051723

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in Neuroscience*, *13*. https://doi.org/10.3389/fnins.2019.00095

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Si, C., Yu, W., Zhou, P., Zhou, Y., Wang, X., & Yan, S. (2022). Inception Transformer. *Advances in Neural Information Processing Systems*, *35*(NeurIPS), 1–15.

Singh, A. K., & Krishnan, S. (2023). ECG signal feature extraction trends in methods and applications. In *BioMedical Engineering Online* (Vol. 22, Issue 1). https://doi.org/10.1186/s12938-023-01075-1

Sun, J., Gao, H., Wang, X., & Yu, J. (2022). Scale Enhancement Pyramid Network for Small Object Detection from UAV Images. *Entropy*, *24*(11). https://doi.org/10.3390/e24111699

Tian, L., Wang, Z., Liu, W., Cheng, Y., Alsaadi, F. E., & Liu, X. (2022). Empower parameterized generative adversarial networks using a novel particle swarm optimizer: algorithms and applications. *International Journal of Machine Learning and Cybernetics*, *13*(4). https://doi.org/10.1007/s13042-021-01440-3

Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent Advances in Stochastic Gradient Descent in Deep Learning. In *Mathematics* (Vol. 11, Issue 3). https://doi. org/10.3390/ math11030682

Visa, S., Ramsay, B., Ralescu, A. L., & Van Der Knaap, E. (2011). Confusion Matrix-based Feature Selection. *MAICS*, *710*, 120–127.

Waheed, A., Goyal, M., Gupta, D., Khanna, A., Al-Turjman, F., & Pinheiro, P. R. (2020). CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection. *IEEE Access*, *8*. https:// doi.org/10.1109 /ACCESS.2020.2994762

Waibel, C., Wortmann, T., Evins, R., & Carmeliet, J. (2019). Building energy optimization: An extensive benchmark of global search algorithms. *Energy and Buildings*, *187*. https://doi.org/10.1016/j.enbuild.2019.01.048

Wang, H., & Zhou, J. (2023). Location of Railway Emergency Rescue Spots Based on a Near-Full Covering Problem: From a Perspective of Diverse Scenarios. *Sustainability (Switzerland)*, *15*(8). https://doi.org/10.3390/su15086833

Wang, J., & Xia, B. (2024). Weakly supervised image segmentation beyond tight bounding box annotations. *Computers in Biology and Medicine*, *169*. https://doi.org/10.1016/j.compbiomed.2023.107913

Wang, Z., Li, P., Cui, Y., Lei, S., & Kang, Z. (2023). Automatic Detection of Individual Trees in Forests Based on Airborne LiDAR Data with a Tree Region-Based Convolutional Neural Network (RCNN). *Remote Sensing*, *15*(4). https://doi.org/10.3390/rs15041024

Wikle, C. K., & Zammit-Mangion, A. (2023). Statistical Deep Learning for Spatial and Spatiotemporal Data. In *Annual Review of Statistics and Its Application* (Vol. 10).

https://doi.org/10.1146/annurev-statistics-033021-112628

Xiao, X., Xiao, W., Zhang, D., Zhang, B., Hu, G., Li, Q., & Xia, S. (2021). Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets. *Computers and Security*, *108*. https://doi.org/10.1016/j.cose.2021.102372

Xie, W., Wei, S., Zheng, Z., Jiang, Y., & Yang, D. (2021). Recognition of Defective Carrots Based on Deep Learning and Transfer Learning. *Food and Bioprocess Technology*, *14*(7). https://doi.org/10.1007/s11947-021-02653-8

Xu, X., Zhao, M., Shi, P., Ren, R., He, X., Wei, X., & Yang, H. (2022). Crack Detection and Comparison Study Based on Faster R-CNN and Mask R-CNN. *Sensors*, *22*(3). https://doi.org/10.3390/s22031215

Yadav, S. P., Jindal, M., Rani, P., de Albuquerque, V. H. C., dos Santos Nascimento, C., & Kumar, M. (2024). An improved deep learning-based optimal object detection system from images. *Multimedia Tools and Applications*, *83*(10). https://doi.org/10.1007/s11042-023-16736-5

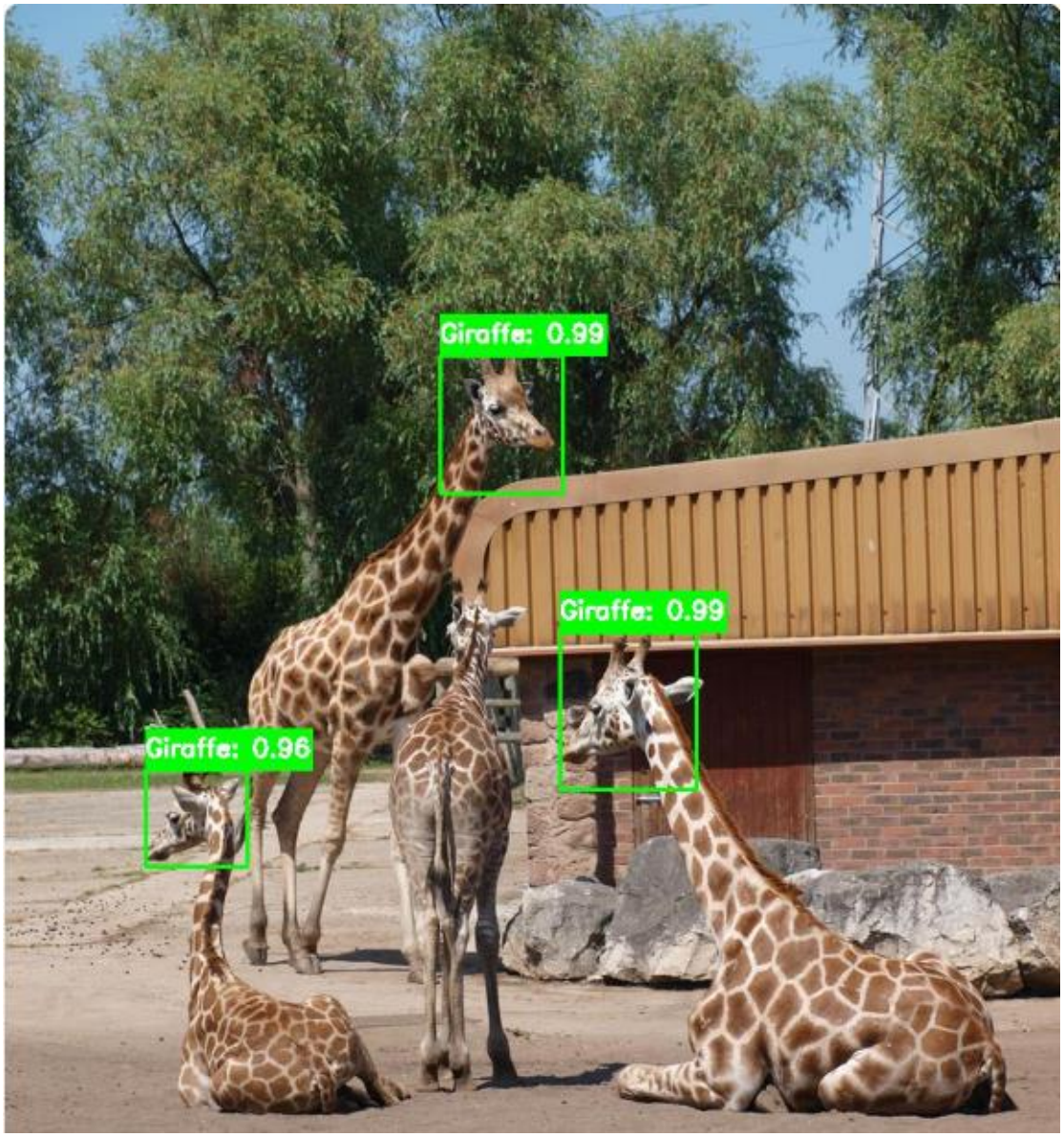
Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In *Insights into Imaging* (Vol. 9, Issue 4, pp. 611–629). Springer Verlag. https://doi.org/10.1007/s13244-018-0639-9

Zhang, X., & Zhang, X. (2020). Global Learnable Pooling with Enhancing Distinctive Feature for Image Classification. *IEEE Access*, *8*. https://doi.org/10.1109/ACCESS.2020.2997078

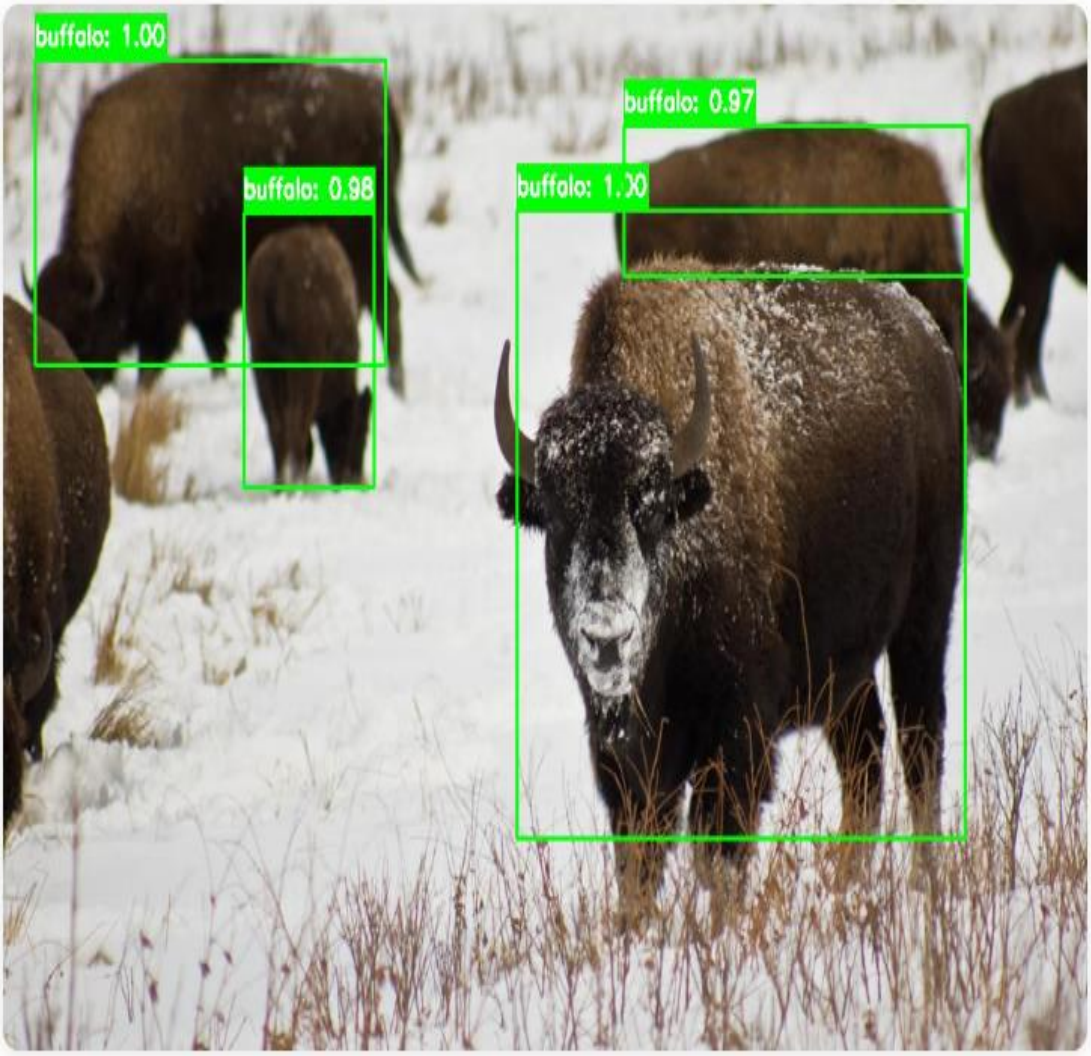
Zhong, Y., Li, X., Xie, J., & Zhang, J. (2023). A Lightweight Automatic Wildlife Recognition Model Design Method Mitigating Shortcut Learning. *Animals*, *13*(5). https://doi.org/10.3390/ani13050838

APPENDIXES

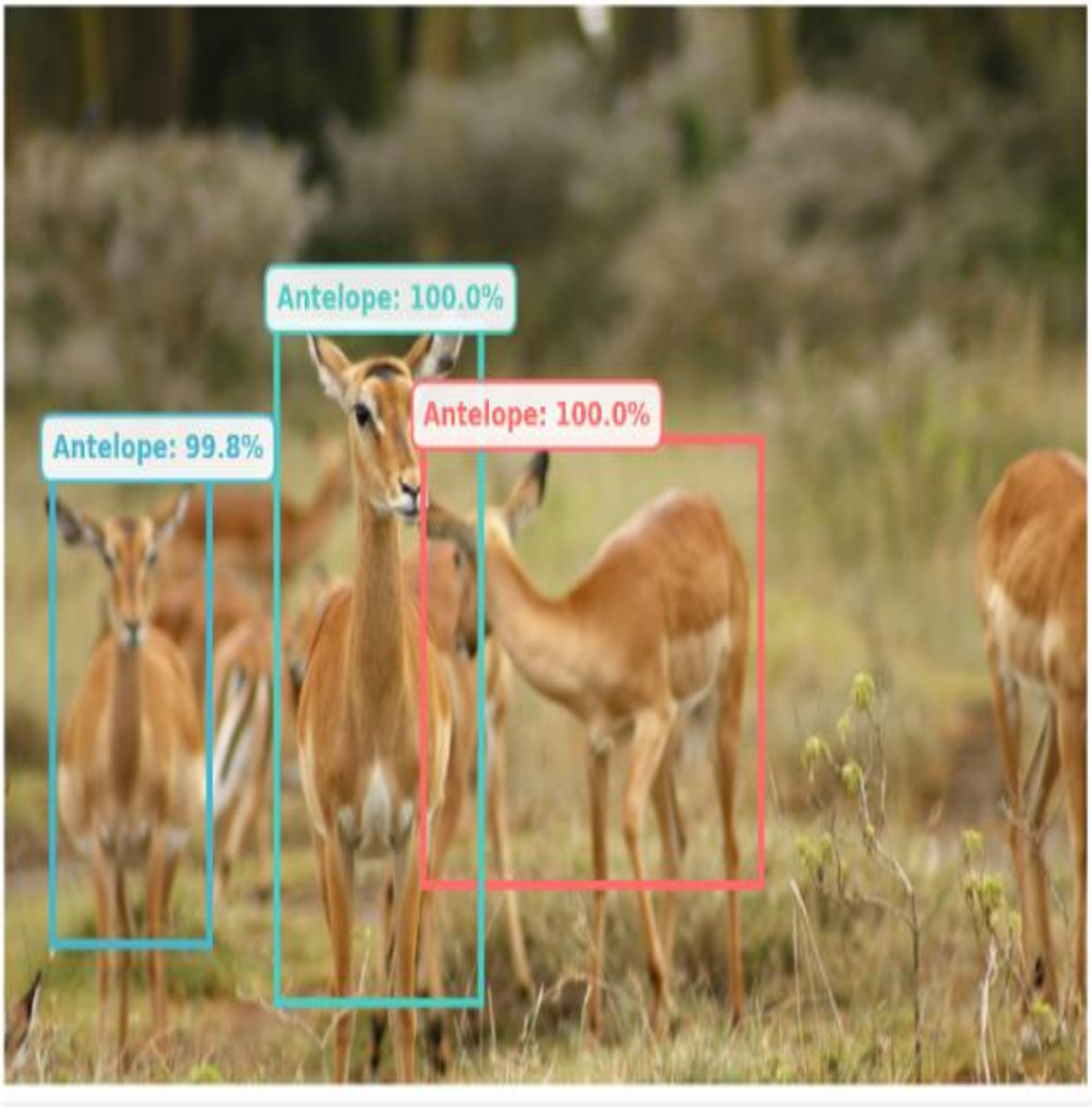
Appendix I: Image Identification ResNet Model



Appendix II: Object identification Inception Model



Appendix III: Object Identification Hybrid Model



Appendix IV: Hybrid Inception model Architecture

```
#Inception Backbone
class WildlifeInceptionBackbone(nn.Module):
   def __init__(self):
      super(WildlifeInceptionBackbone, self).__init__()

                                                                inception =
        models.inception_v3(weights=models.Inception_V3_Weights.IMAGENET1K
        _V1, aux_logits=True)
      inception.eval()

      self.Conv2d_1a_3x3 = inception.Conv2d_1a_3x3
      self.Conv2d_2a_3x3 = inception.Conv2d_2a_3x3
      self.Conv2d_2b_3x3 = inception.Conv2d_2b_3x3
      self.maxpool1 = inception.maxpool1
      self.Conv2d_3b_1x1 = inception.Conv2d_3b_1x1
      self.Conv2d_4a_3x3 = inception.Conv2d_4a_3x3
      self.maxpool2 = inception.maxpool2

      self.Mixed_5b = inception.Mixed_5b
      self.Mixed_5c = inception.Mixed_5c
      self.Mixed_5d = inception.Mixed_5d
      self.Mixed_6a = inception.Mixed_6a
      self.Mixed_6b = inception.Mixed_6b
      self.Mixed_6c = inception.Mixed_6c
      self.Mixed_6d = inception.Mixed_6d
      self.Mixed_6e = inception.Mixed_6e
      self.Mixed_7a = inception.Mixed_7a
      self.Mixed_7b = inception.Mixed_7b
      self.Mixed_7c = inception.Mixed_7c

      self.level4_enhance = nn.Sequential(
         nn.Conv2d(768, 256, 3, padding=1, bias=False),
         nn.BatchNorm2d(256),
         nn.ReLU(inplace=True),
         nn.Conv2d(256, 256, 1, bias=False),
         nn.BatchNorm2d(256),
         nn.ReLU(inplace=True)
      )

      self.level5_enhance = nn.Sequential(
         nn.Conv2d(2048, 256, 3, padding=1, bias=False),
         nn.BatchNorm2d(256),
         nn.ReLU(inplace=True),
         nn.Conv2d(256, 256, 1, bias=False),
         nn.BatchNorm2d(256),
         nn.ReLU(inplace=True)
      )
```

```
        self._init_weights()

    def _init_weights(self):
        for m in [self.level4_enhance, self.level5_enhance]:
            for layer in m.modules():
                if isinstance(layer, nn.Conv2d):
                  nn.init.kaiming_normal_(layer.weight, mode='fan_out', nonlinearity='relu')
                elif isinstance(layer, nn.BatchNorm2d):
                    nn.init.constant_(layer.weight, 1)
                    nn.init.constant_(layer.bias, 0)

    def forward(self, x):
        x = self.Conv2d_1a_3x3(x)
        x = self.Conv2d_2a_3x3(x)
        x = self.Conv2d_2b_3x3(x)
        x = self.maxpool1(x)
        x = self.Conv2d_3b_1x1(x)
        x = self.Conv2d_4a_3x3(x)
        x = self.maxpool2(x)

        x = self.Mixed_5b(x)
        x = self.Mixed_5c(x)
        x = self.Mixed_5d(x)
        x = self.Mixed_6a(x)
        x = self.Mixed_6b(x)
        x = self.Mixed_6c(x)
        x = self.Mixed_6d(x)

        level4_raw = self.Mixed_6e(x)
        level4_features = self.level4_enhance(level4_raw)

        x = self.Mixed_7a(level4_raw)
        x = self.Mixed_7b(x)
        level5_raw = self.Mixed_7c(x)
        level5_features = self.level5_enhance(level5_raw)

        return {
            "res4": level4_features,
            "res5": level5_features
        }

@BACKBONE_REGISTRY.register()
class InceptionBackboneWrapper(Backbone):
    def __init__(self, cfg, input_shape):
        super().__init__()
        self.backbone = WildlifeInceptionBackbone()

    def forward(self, x):
        return self.backbone(x)
```

```
def output_shape(self):
    return {
        "res4": ShapeSpec(channels=256, stride=16),
        "res5": ShapeSpec(channels=256, stride=32)
    }
```

Appendix V: Hybrid ResNet model Architecture

```
#ResNet Backbone
class EnhancedResNetBackbone(nn.Module):
    def __init__(self):
        super(EnhancedResNetBackbone, self).__init__()

                                                                    resnet                    =
            models.resnet101(weights=models.ResNet101_Weights.IMAGENET1K_V2)

        self.conv1 = resnet.conv1
        self.bn1 = resnet.bn1
        self.relu = resnet.relu
        self.maxpool = resnet.maxpool
        self.layer1 = resnet.layer1
        self.layer2 = resnet.layer2
        self.layer3 = resnet.layer3
        self.layer4 = resnet.layer4

        self.enhance_res4 = nn.Sequential(
            nn.Conv2d(1024, 256, 3, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True)
        )

        self.enhance_res5 = nn.Sequential(
            nn.Conv2d(2048, 256, 3, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True)
        )

        self._init_weights()

    def _init_weights(self):
        for m in [self.enhance_res4, self.enhance_res5]:
            for layer in m.modules():
                if isinstance(layer, nn.Conv2d):
                    nn.init.kaiming_normal_(layer.weight, mode='fan_out', nonlinearity='relu')
                elif isinstance(layer, nn.BatchNorm2d):
                    nn.init.constant_(layer.weight, 1)
                    nn.init.constant_(layer.bias, 0)

    def forward(self, x):
        x = self.conv1(x)
```

```
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.layer1(x)
        x = self.layer2(x)

        res4_raw = self.layer3(x)
        res4_enhanced = self.enhance_res4(res4_raw)

        res5_raw = self.layer4(res4_raw)
        res5_enhanced = self.enhance_res5(res5_raw)

        return {
            "res4": res4_enhanced,
            "res5": res5_enhanced
        }

@BACKBONE_REGISTRY.register()
class ResNetBackboneWrapper(Backbone):
    def __init__(self, cfg, input_shape):
        super().__init__()
        self.backbone = EnhancedResNetBackbone()

    def forward(self, x):
        return self.backbone(x)

    def output_shape(self):
        return {
            "res4": ShapeSpec(channels=256, stride=16),
            "res5": ShapeSpec(channels=256, stride=32)
        }
```

Appendix VI: Faster RCNN Detectron 2 Framework

```
# Configuration
def setup_model_config(backbone_name, num_classes, output_dir):
    cfg = get_cfg()
                                    cfg.merge_from_file(model_zoo.get_config_file("COCO-
        Detection/faster_rcnn_R_50_FPN_3x.yaml"))
            cfg.MODEL.WEIGHTS       =       model_zoo.get_checkpoint_url("COCO-
        Detection/faster_rcnn_R_50_FPN_3x.yaml")

    cfg.MODEL.BACKBONE.NAME = backbone_name

    cfg.DATASETS.TRAIN = ("animal_train",)
    cfg.DATASETS.TEST = ("animal_val",)
    cfg.DATALOADER.NUM_WORKERS = 2
    cfg.DATALOADER.FILTER_EMPTY_ANNOTATIONS = True

    cfg.MODEL.ANCHOR_GENERATOR.SIZES = [
       [32, 64, 128, 256, 512],
       [64, 128, 256, 512, 1024]
    ]
    cfg.MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS = [
       [0.25, 0.5, 1.0, 2.0, 4.0],
       [0.25, 0.5, 1.0, 2.0, 4.0]
    ]

    cfg.MODEL.RPN.IN_FEATURES = ["res4", "res5"]
    cfg.MODEL.RPN.BATCH_SIZE_PER_IMAGE = 512
    cfg.MODEL.RPN.POSITIVE_FRACTION = 0.5
    cfg.MODEL.RPN.PRE_NMS_TOPK_TRAIN = 3000
    cfg.MODEL.RPN.PRE_NMS_TOPK_TEST = 1500
    cfg.MODEL.RPN.POST_NMS_TOPK_TRAIN = 1500
    cfg.MODEL.RPN.POST_NMS_TOPK_TEST = 1000
    cfg.MODEL.RPN.NMS_THRESH = 0.7

    cfg.MODEL.ROI_HEADS.IN_FEATURES = ["res4", "res5"]
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = num_classes
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
    cfg.MODEL.ROI_HEADS.POSITIVE_FRACTION = 0.25
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.05
    cfg.MODEL.ROI_HEADS.NMS_THRESH_TEST = 0.3

    cfg.MODEL.FPN.IN_FEATURES = ["res4", "res5"]
    cfg.MODEL.FPN.OUT_CHANNELS = 256
    cfg.MODEL.FPN.NORM = "GN"

    cfg.MODEL.ROI_BOX_HEAD.BBOX_REG_LOSS_TYPE = "giou"
    cfg.MODEL.ROI_BOX_HEAD.BBOX_REG_LOSS_WEIGHT = 3.0
    cfg.MODEL.ROI_BOX_HEAD.POOLER_RESOLUTION = 7
    cfg.MODEL.ROI_BOX_HEAD.POOLER_SAMPLING_RATIO = 2
```

```
cfg.MODEL.ROI_BOX_HEAD.POOLER_TYPE = "ROIAlignV2"
cfg.MODEL.ROI_BOX_HEAD.NUM_FC = 2
cfg.MODEL.ROI_BOX_HEAD.FC_DIM = 1024
cfg.MODEL.ROI_BOX_HEAD.NORM = "GN"

cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.0025
cfg.SOLVER.MAX_ITER = 30000
cfg.SOLVER.STEPS = (15000, 20000)
cfg.SOLVER.GAMMA = 0.1
cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.WARMUP_FACTOR = 1.0 / 1000
cfg.SOLVER.WEIGHT_DECAY = 0.0001
cfg.SOLVER.MOMENTUM = 0.9
cfg.SOLVER.CLIP_GRADIENTS.ENABLED = True
cfg.SOLVER.CLIP_GRADIENTS.CLIP_TYPE = "norm"
cfg.SOLVER.CLIP_GRADIENTS.CLIP_VALUE = 1.0

cfg.INPUT.MIN_SIZE_TRAIN = (600, 700, 800, 900)
cfg.INPUT.MAX_SIZE_TRAIN = 1200
cfg.INPUT.MIN_SIZE_TEST = 800
cfg.INPUT.MAX_SIZE_TEST = 1200
cfg.INPUT.FORMAT = "BGR"
cfg.INPUT.RANDOM_FLIP = "horizontal"
cfg.INPUT.BRIGHTNESS = 0.15
cfg.INPUT.CONTRAST = 0.15
cfg.INPUT.SATURATION = 0.15
cfg.INPUT.HUE = 0.05

cfg.TEST.AUG.ENABLED = True
cfg.TEST.AUG.MIN_SIZES = (700, 800, 900)
cfg.TEST.AUG.MAX_SIZE = 1200
cfg.TEST.AUG.FLIP = True

cfg.TEST.EVAL_PERIOD = 5000
cfg.TEST.DETECTIONS_PER_IMAGE = 50

cfg.OUTPUT_DIR = output_dir
cfg.MODEL.DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
cfg.SEED = 42

return cfg
```

Appendix VII: Chuka University Introductory Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

OFFICE OF RECTOR
BOARD OF POSTGRADUATE STUDIES

Telephones: 020-2310512/18
Direct Line: 020-268 7625
Email: postgraduate@chuka.ac.ke
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

SM22/5809/22

2nd May, 2025

To
National Commission for Science Technology and Innovation
Off Waiyaki Way, Upper Kabete
P O Box 30623, 00100
Nairobi

Dear Sir/ Madam,

MALACH OBISA AMONGA

The above named person is a bona fide student of Chuka University pursuing a Degree of Master of Science Degree .in Computer Science, proposal titled: A Hybrid of Residual Network and Inception Neural Network Model for Wildlife Detection and Identification

Mr. Amonga has defended at Faculty level, and is now expected to conduct research. Any assistance accorded to him will be highly appreciated.

Yours sincerely,

CHUKA UNIVERSITY
02 MAY 2025
P.O. Box 109-60400, CHUKA
BOARD OF POSTGRADUATE STUDIES

DIRECTOR
BOARD OFPOSTGRADUA TE STUDIES
MM/kk

Appendix VIII: Ethics Review Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

CHUKA UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE

Telephone: 020-2310512/18 P. O. Box 109-60400, Chuka
Direct Line: 0772894438 Email: info@chuka.ac.ke, Website: www.chuka.ac.ke

25th April, 2025

REF: CUIERC/ NACOSTI/715
TO: **Malach Obisa Amonga**

RE: A Hybrid of Residual Network and Inception Neural Network Model for Wildlife Detection and Identification

This is to inform you that *Chuka University IERC* has reviewed and approved your above research proposal. Your application approval number is *NACOSTI/NBC/AC-0812*. The approval period is 25th April, 2025 – 25th April, 2026.

This approval is subject to compliance with the following requirements;

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by *Chuka University IERC*.
iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to *Chuka University IERC* within 72 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to *Chuka University IERC* within 72 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to *Chuka University IERC*.

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely

Dr. Benjamin Kanga
SECRETARY

INSTITUTIONAL ETHICS REVIEW COMMITTEE
25 APR 2025
P.O. Box 109-60400, CHUKA
CHUKA UNIVERSITY

Appendix IX: National Commission for Science, Technology and Innovation (NACOSTI) License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **192449**

Date of Issue: **10/May/2025**

RESEARCH LICENSE

This is to Certify that Mr.. MALACH OBISA AMONGA of Chuka University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act; 2013 (Rev.2014) in Tharaka-Nithi on the topic: A HYBRID OF RESIDUAL NETWORK AND INCEPTION NEURAL NETWORK MODEL FOR WILDLIFE DETECTION AND IDENTIFICATION for the period ending : 10/May/2026.

License No: **NACOSTI/P/25/4173650**

192449

Applicant Identification Number

Deputy Director
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions