

# Assessing the Efficacy of EfficientNet, Inception, and ResNet for Wildlife Species Identification

MSc Research Project  
Data Analytics

Preena Darshini  
x22238590

School of Computing  
National College of Ireland

Supervisor: Prof. Barry Haycock

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** .....Preena Darshini . ....

**Student ID:** .....x22238590.....

**Programme:** .....MSc Data Analytics..... **Year:** .....2024.....

**Module:** .....MSc Research Project .....

**Supervisor:** .....Professor Barry Haycock.....


**Submission Due Date:** .....12/08/2024.....

**Project Title:** ..... Assessing the Efficacy of EfficientNet, Inception, and ResNet for Wildlife Species Identification .....

**Word Count:** .....6186..... **Page Count:**.....20.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:**  .....

**Date:** 10/08/2024 .....

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input checked="" type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input checked="" type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input checked="" type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Assessing the Efficacy of EfficientNet, Inception, and ResNet for Wildlife Species Identification

Preena Darshini  
x22238590

## Abstract

In the current scenario, there is a lot of attention in the field of wildlife conservation to identify wildlife species in order to easily monitor biodiversity and aid ecologists in conservation efforts using deep learning techniques. This document discusses the use of EfficientNet, Inception, and ResNet to identify animal species on the iWildCam 2019 dataset. Challenges like class imbalance are handled. The models are trained and evaluated. The predictions from these models are combined with an ensemble to boost the performance. The ResNet model was fine-tuned to improve performance. The ensemble approach yielded an accuracy of 55.1%. The potential and limitations of these models are discussed with respect to the field of ecological research. Future work has also been suggested to improve the model accuracy which can handle complex imbalanced datasets.

## 1 Introduction

Detecting or counting of wildlife species is still a matter of interest for conservationists, biologists, and ecologists. Accurately identifying wildlife species using camera trap images is important for the conservation of wildlife and for monitoring the diversity and balance of the ecosystem. Camera traps are one of the most essential tools for wildlife conservationists and biologists as it provides images that can help with analysing the population distribution and understanding the mannerisms of different animal species (O’Connell et al., 2011). Initially, species identification or counting used to take place manually. Traditional or manual approaches also depended on the experience of personnel and manual monitoring. This manual operation and processing of images are both resource-consuming and time-consuming which has pushed for the need for developing automated systems using advanced deep learning models (Tabak et al., 2019). The need for this study is influenced by many factors. Firstly, there is an increase in human-fauna conflicts like vehicle collisions with animals. This endangers the safety of both humans and wildlife. Thus, efficient wildlife detection systems can help manage such conflicts by providing timely alerts (Sreedevi, K. L. and Edison, 2022). Secondly, precise monitoring of animal populations is key for conservation initiatives, especially amidst habitat destruction and climate change. Conservationists can make better-informed strategies using automated systems which are more reliable than manual methods (Berry et al., 2019).

This raises the research question: How can deep learning methods like EfficientNet, ResNet, and Inception be improved to enhance the accuracy in wildlife species identification?

The key or major objectives of this research include developing and evaluating deep learning models for identifying wildlife species and assessing the effectiveness of the ensemble technique in combining these models.

It is hypothesised that combining deep learning models through an ensemble technique will enhance the accuracy of animal identification.

This study contributes to the scientific literature on wildlife identification and monitoring by utilising models like EfficientNet, ResNet, and Inception for identifying animal species. ResNet has been fine-tuned to improve its accuracy and the results of each model have been used in an ensemble technique. This study provides an efficient approach or strategy to detect animals in the wild which can help in reducing human and wild animal conflicts and help in conservation initiatives.

The remainder of this report is arranged as follows: Related Work is presented in Section 2. Section 3 discusses about the Research Methodology. Section 4 and Section 5 discuss about the Design Specification and Implementation respectively. Section 6 talks about Evaluation. Finally, Section 7 discusses the Conclusion and talks about Future Work.

## **2 Related Work**

This section discusses about the literature review in different areas. Previous and current practices are discussed. The focus of this section is on critically reviewing different deep learning and ensemble techniques that have been used by various authors for identifying or classifying animal species.

### **2.1 Implementations using Deep Learning Methods**

There has been a significant improvement in the use of Convolutional Neural Networks (CNN) and other advanced deep learning methods. This section highlights reviews of various studies that have used these techniques for animal identification.

The study conducted by Sreedevi, K. L. and Edison (2022) focused on using convolutional layers that are separable depth-wise to develop a model that was used for detecting animals in the wild. The model utilised zero padding in order to keep up the edge characteristics and keep the output image size in control. This enabled the authors to achieve high accuracy in both detection (IoU 0.87) and classification (99.6%) which indicates robustness. The use of this technique made the model resource-efficient. However, it failed to discuss the model's ability to perform on real-world cluttered images. Wang, Y. et al. (2022) proposed two new approaches namely FilterDetector and DLEDetector. Both these methods were based on Microsoft MegaDetector V4. FilterDetector used Non-Maximum Suppression (NMS) with density filtering. DLEDetector uses an ensemble of Weighted Box Fusion (WBF) and binary classification models. These methods were computationally expensive and there was a reliance on the accuracy of the initial MegaDetector V4 results.

The task of animal classification was also performed using both supervised and unsupervised learning methods by Manohar et al. (2016). The authors of this study used maximum region merging segmentation algorithm to segment animals and Gabor features for extracting features. Linear Discriminant Analysis (LDA) was used for supervised learning

along with symbolic classifiers and Principal Component Analysis (PCA) was utilised for unsupervised learning along with K-means clustering. This study compares the two methods and concludes that supervised learning method achieves greater accuracy. The dataset used was small with just 2000 images. This could limit the generalisability of results.

The study conducted by Kanaga Priya et al. (2023) used a CNN-based approach to classify animals based on species. Convolutional layers were used to extract features and max-pooling along with fully connected layers was used for classification. The authors used transfer learning to fine-tune pre-trained models. This method achieved a high accuracy of 98%. However, it focused only on a limited number of species. Deng et al. (2023) proposed an automatic animal recognition model based on Faster R-CNN using TensorFlow. This technique was fast and accurate at detecting animals. The accuracy was 92.78%. This method requires further optimising to meet real-world scenarios.

The paper written by Abood et al. (2023) utilised YOLOv5 and Regions-Convolutional Neural Networks to detect object overlaps. The accuracy was 99.38%. This could be due to overfitting issues or the model actually performs very well. An enhanced YOLOv5 model was used by Wang, F. et al. (2022) which demonstrated higher accuracy and efficiency compared to the traditional YOLOv5. The authors had integrated Inception and ResNet for feature extraction.

The review written by Palanisamy and Ratnarajah (2021) highlights both the strengths and weaknesses of various deep learning techniques which provides a roadmap for future studies. Deep discussion related to technical details was not highlighted. Deep Convolutional Neural Networks (DCNN) which includes 3 convolutional layers and 3 max-pooling layers was used to detect wildlife species in North America (Chen et al., 2014). A comparison was done between DCNN and bag of visual words (BOW) model by these authors. DCNN outperformed BOW model indicating that it is superior as it handled large amounts of data. Simultaneously, DCNN requires a large amount of labelled training data which could be a limitation in certain scenarios. The use of CNN was not limited to detecting animal species. It was also used in food recognition.

Ali and Subhi (2018) used CNN with multiple pooling layers to detect food in a dataset consisting of local Malaysian food items. This study showed that CNNs achieved higher accuracy for detection and recognition tasks. It has been proven again that CNNs are excellent at identification and classification tasks in many image recognition domains. Sherin et al. (2024) used CNN based approach to detect bird species and achieved an accuracy of 97.21%. The use of Flask web application could limit the real-time application in case the environment is resource constrained.

The studies reviewed above reveal the potential of CNNs and deep learning techniques in species identification and classification. There are various challenges like generalisability of data, resources and computational constraints. These challenges motivate the need to develop robust models that can handle large and diverse datasets.

## **2.2 Ensemble Techniques and Advanced Algorithms**

Many researchers employed ensemble techniques to increase the accuracy of species detection models. This section reviews the studies that have used ensemble approaches.

As mentioned previously, Wang, Y. et al. (2022) used ensemble methods by combining Weighted Box Fusion (WBF) and binary classifier. The approach increased the detection accuracy effectively. Deep ensemble learning and transfer learning techniques were employed by Samarasinghe and Lakmali (2023) to classify bees, snakes, and wasps. A training accuracy of 93% was achieved which indicated robustness of the method. The ensemble model can increase computational complexity which means there could be a demand for more processing power and memory.

Classical machine learning algorithms like SVM, Logistic Regression, KNN, and Decision Trees along with ensemble methods like Random Forest and AdaBoost were used to identify endangered species like Markhor, Asiatic Black Bear, and Snow Leopard (Kumar et al., 2022). With the small number of classes and images per class. Logistic regression and Random Forest performed well. The generalisability of the model was not discussed given the small size of the dataset.

An ensemble of deep learning models was used to classify and identify bird species from snapshot images from camera traps (Jahan and Ganesan, 2024). The authors used CNNs and ANN to train and classify bird species. The output of these were combined as a snapshot ensemble technique to form a single model. The accuracy of the model was 98.7%. The dataset was very small with just 500 images which could limit its generalisability in real-world cases.

An ensemble of three deep learning models was employed namely, VGG16, VGG19, and ResNet50 to identify butterfly species (Nadarashalingham et al., 2023). The dataset size was again limited to only 2,182 images which may still not capture all the variability. An interesting new classification principle called Semantic Learning Principle for Animal Classification (SLPAC) was proposed by Mary Diana et al. (2024) and its results were compared with CNN. SLPAC outperformed regular CNNs. This also requires high computational resources and does not discuss the scalability of this method.

Instead of pixel-by-pixel comparison difference calculations to indicate the presence of felines particularly ocelots, the squared window method was employed (Figueroa et al., 2014). This method was fast and applicable to real-time applications. However, this method also requires the consecutive images to be captured by the same camera which might not always be possible.

The use of ensemble techniques seems promising in relation to species identification. The limitations mentioned must be addressed to handle real-world scenarios and applications.

## **3 Research Methodology**

### **3.1 Research Justification**

CNNs have become standardised for tasks such as image classification as it can quite efficiently extract features from images. Krizhevsky et al. (2017) made CNNs popular using AlexNet architecture. Ever since then, the use of sophisticated models like EfficientNet, ResNet, and Inception are on the rise.

### **3.1.1 EfficientNet**

For this research, EfficientNet was chosen as it efficiently balances the network depth, width, and resolution providing high accuracy using less computational resources (Tan and Le, 2019). It utilises compound coefficient to scale the depth, width, and resolution of the neural network (Hossen et al., 2023).

### **3.1.2 ResNet**

The ResNet model is suitable for this research as it allows training of deep networks by solving the vanishing gradient issue (He et al., 2016). This model is known for its ability to perform well as well as for its robustness with an increase in the depth of the network.

### **3.1.3 Inception**

Inception is also called as GoogLeNet as it makes use of parallel convolutions using varying kernel sizes to capture features at multiple scales within images (Szegedy et al., 2015). It handles complex images well especially where the objects vary significantly in size.

### **3.1.4 Justification**

The main reason for the choice of using these three models lies in its ability to classify images efficiently and its adaptability to various scales. Based on previous studies, these models when used individually or with other models had the capability to tackle challenges in identifying wildlife in scenarios with changes in image quality, complexity, and size of the object. Hence, these models have been used in combination with an ensemble technique to assess if this combination can improve the efficacy of identifying wildlife species on the iWildCam 2019 dataset.

## **3.2 Steps followed in the Research Work**

This section describes the research procedure using the Knowledge Discovery in Databases (KDD) methodology which was proposed by Fayyad et al. (1996). This research ensures that a systematic approach is followed to discover patterns and build predictive models to identify wildlife species.

### **3.2.1 Literature Review**

Existing studies were reviewed to better understand the current state-of-the-art solutions. EfficientNet, ResNet, and Inception have been used as the primary models along with an ensemble technique.

### 3.2.2 Data Collection

The iWildCam 2019 dataset was obtained from Kaggle (Beery, Morris and Perona, 2019). It consisted of 196,086 training images and 153,730 testing images.

### 3.2.3 Data Preprocessing

Loading of the dataset, standardizing images, data augmentation or transformation, and class imbalance issues were handled.

### 3.2.4 Model Training and Ensemble

Pre-trained EfficientNet, ResNet, and Inception were used and the outputs of the models were combined using a simple weighted average ensemble technique.

### 3.2.5 Model Evaluation

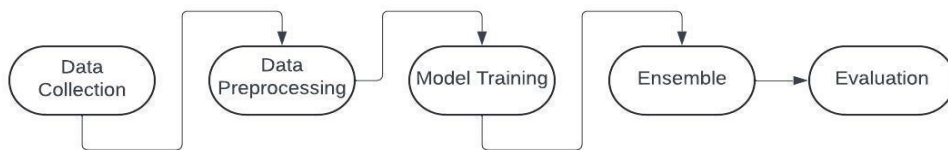
Pre-trained EfficientNet, ResNet, and Inception models were evaluated using performance metrics like accuracy, precision, recall, and F1-score. The performance of each model and the ensemble technique were analysed.

## 4 Design Specification

This section outlines the architecture, requirements, and framework used for the implementation.

### 4.1 System Architecture

This architecture of the system for wildlife species identification involves various layers such as data collection layer, data preprocessing layer, model training layer, ensemble layer, and evaluation layer as shown in Figure 1 below.



**Figure 1: System Architecture – Layers**

#### 4.1.1 Data Collection

For this research, a publicly available dataset called iWildCam 2019 was sourced from Kaggle (Beery, Morris and Perona, 2019). The dataset consists of images of wild animals captured using camera traps set up at various locations. Caltech Camera Traps dataset was used for training and Idaho Camera Traps was used for testing purposes. The training dataset consists of 196,086 images from 138 locations in the southern part of California. This dataset contains both images of wild animals and empty images where no animals are present. Due to



the large number of images and different locations, the training dataset can be considered as diverse and rich making it suitable for training various deep learning models. The test dataset contains 153,730 images from 100 places in Idaho and contains a mixture of images similar to the training dataset. The dataset consists of 'train\_images' and 'test\_images' in zip file format which was extracted and used.

#### 4.1.2 Preprocessing Data

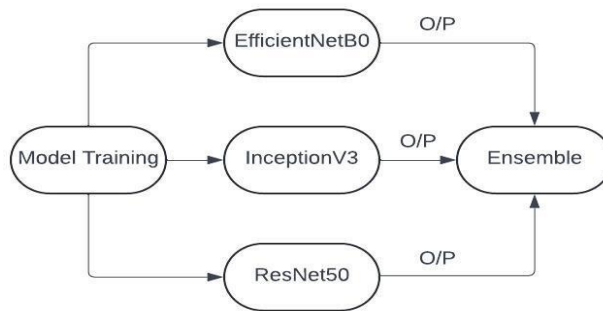
The filenames of images in both train and test datasets were loaded for easy access to the data and for manipulation. The data from both files were loaded into pandas DataFrames which consists of metadata about those images including class labels and file names. Basic information and structure of the DataFrames are checked to better understand the class distributions and composition of data. Random sample of images from the training dataset is displayed to understand the quality and variety of images. In order to make the data more interpretable, class names were matched with the category IDs in the training DataFrame.

#### 4.1.3 Transforming Data

The training dataset was split into train and validation subsets so that the model could be validated and evaluated on data that is unseen. The class imbalance issue was taken care. Rescaling of image pixel values to a range between 0 and 1 was performed. The classes categorised as empty were not removed as even empty images can also be categorised as valid images or a meaningful category.

#### 4.1.4 Model Building and Evaluation

The number and indices of classes were set for training the models which would help in configuring the final layer of the neural network. The training and validation data generators were created to manage loading and preprocessing of the images batch-wise. The Figure 2 shows that the model training layer involves training three models namely, EfficientNetB0, InceptionV3, and ResNet50.



**Figure 2: Model Training Layer expansion**

All three models were trained using both train and validation sets and finally tested using the test data. Each of the models was evaluated using metrics like precision, accuracy, recall, confusion matrix, and F1 score. The results from the three models were combined using the ensemble technique.

## 4.2 Requirements

The requirements for this research have been listed below.

### 4.2.1 Data Requirements

A diverse dataset is needed which represents a wide variety of animals in the wild. Metadata associated with the image dataset like the class labels must be available.

### 4.2.2 Software Requirements

Python programming language was used for data preprocessing, development of the models, and evaluation of this research. Python provides various libraries for building and handling complex models and large datasets. The computing platform used was Jupyter Notebook. Various deep learning libraries have been utilised. TensorFlow framework and Keras API were used for developing deep learning models. Libraries like pandas and numpy were used for manipulating and preprocessing the data. Visualisation libraries used were matplotlib and seaborn. Python Imaging Library was used for handling various image file formats. Scikit-learn was used for data preprocessing, model evaluation using various performance metrics, dataset splitting, computing the class weights, and confusion matrices. TensorFlow Hub library was used to load the pre-trained models like EfficientNetB0, ResNet50, and InceptionV3. The load\_model was used to load a previously saved keras file. Adam is an optimisation algorithm and EarlyStopping was used to prevent overfitting by stopping the training process by monitoring the validation set.

### 4.2.3 Hardware Requirements

The system used for this project contains Intel® Iris® Xe Graphics which is a high-performance GPU. Hence, training the complex deep learning models on a large dataset was efficient and less time-consuming.

## 5 Implementation

This section discusses the implementation process of the proposed solution to identify wildlife species. As discussed previously, the dataset was downloaded, the zip files were extracted and the necessary libraries were installed and imported.

### 5.1 Data Availability and Class Definitions

This step involves checking if the data is available in the specified directory. A list of files and folders was displayed. The files ‘train.csv’ and ‘test.csv’ contained the metadata. The folders ‘train\_images’ and ‘test\_images’ contained the images. A Python dictionary was created so that the numerical labels could be mapped to the species names. This will make interpretation of the model results easier. The paths to the training and test images were defined and listed. There were 196,086 training images and 153,730 test images.

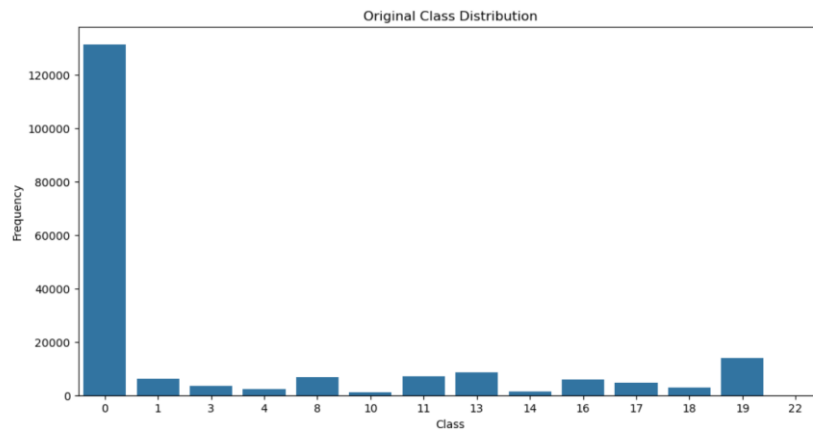
## 5.2 Loading and Exploring the Data

The metadata in the CSV files were loaded in Python DataFrames named ‘train\_df’ and ‘test\_df’. The first few rows of the DataFrames were displayed. It also showed that there were no missing values. A sample of 16 training images was displayed.

## 5.3 Preparing and Transforming the Data

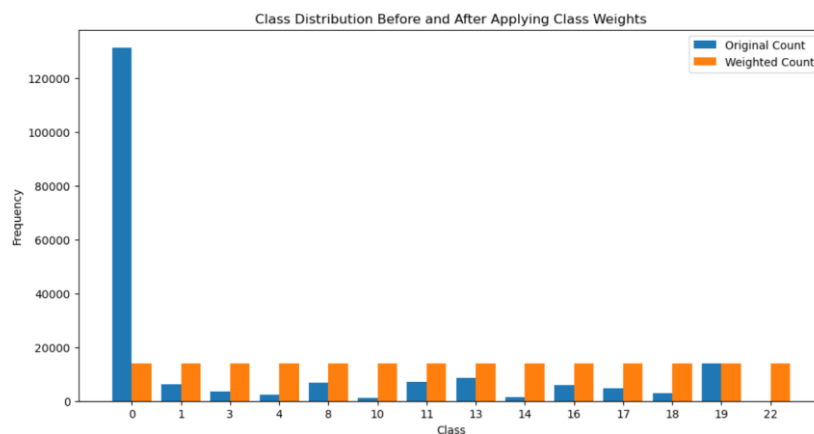
The training DataFrame initially did not contain the class names or category names. This was appended as a new column in the training DataFrame. It mapped the category IDs to the class names making the data more understandable. The file paths for the image files in the DataFrame were updated to the complete path of each image file for easy access during training. Once this was done, each image was linked to its metadata.

The next step was to analyse the class distribution. The class distribution of training data was plotted using the bar graph as shown in Figure 3.



**Figure 3: Class Distribution Bar Graph**

The training dataset was split into train and validation subsets. A Python dictionary containing class weights was generated. The class imbalance issue was taken care of by calculating the weights inversely proportional to the frequencies of each class to avoid bias and was visualised using a bar plot as shown in Figure 4.



**Figure 4: Class Distribution Bar Graph after applying class weights**

## 5.4 Preprocessing the Data

The Data Generators were set up for the training and validation sets using the ImageDataGenerator class. This class was used to rescale and normalise the pixel values to [0, 1] range to make feeding into the neural network easier. The images in the training and validation data generators were resized to 128x128 pixels. The training generator had 117,780 image files with 14 classes. The validation generator contained 39,259 images. Later on, a separate test data generator was also set up. The number of unique classes in the dataset was identified to be 14.

## 5.5 Training the Models – EfficientNet, Inception, and ResNet

The EfficientNetB0 pre-trained model was loaded from the TensorFlow Keras. The custom output layer contains the number of classes in the dataset i.e. 14. The global average pooling layer and dense layer with softmax activation were used for building the model. The training data generator was used to train the model. Early stopping was employed to prevent overfitting as shown in Figure 5. Class weights were added to handle the imbalance. The training and validation accuracy and loss were plotted to monitor the performance of the model as shown in Figure 6.

```
Epoch 1/20
C:\Users\Preena_Rahul\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: You should call 'super().__init__(**kwargs)' in its constructor. '**kwargs' can include 'workers', 'use_multiprocessing', 'max_queue_size' to 'fit()', as they will be ignored.
  self._warn_if_super_not_called()
100/100 188s 2s/step - accuracy: 0.4945 - loss: 21.8243 - val_accuracy: 0.6969 - val_loss: 1.4025
Epoch 2/20
100/100 155s 2s/step - accuracy: 0.6873 - loss: 8.3806 - val_accuracy: 0.7259 - val_loss: 1.2024
Epoch 3/20
100/100 145s 1s/step - accuracy: 0.7160 - loss: 6.4711 - val_accuracy: 0.7325 - val_loss: 1.1998
Epoch 4/20
100/100 132s 1s/step - accuracy: 0.7310 - loss: 6.0424 - val_accuracy: 0.7394 - val_loss: 1.1634
Epoch 5/20
100/100 122s 1s/step - accuracy: 0.7357 - loss: 5.3720 - val_accuracy: 0.7409 - val_loss: 1.1450
Epoch 6/20
100/100 90s 910ms/step - accuracy: 0.7305 - loss: 5.0594 - val_accuracy: 0.7487 - val_loss: 1.0929
Epoch 7/20
100/100 93s 930ms/step - accuracy: 0.7384 - loss: 5.6726 - val_accuracy: 0.7544 - val_loss: 1.0614
Epoch 8/20
100/100 109s 1s/step - accuracy: 0.7383 - loss: 5.6707 - val_accuracy: 0.7563 - val_loss: 1.0576
Epoch 9/20
100/100 107s 1s/step - accuracy: 0.7540 - loss: 4.6866 - val_accuracy: 0.7725 - val_loss: 0.9534
Epoch 10/20
100/100 109s 1s/step - accuracy: 0.7421 - loss: 6.1157 - val_accuracy: 0.7725 - val_loss: 0.9383
Epoch 11/20
100/100 102s 1s/step - accuracy: 0.7505 - loss: 4.7478 - val_accuracy: 0.7747 - val_loss: 0.9433
Epoch 12/20
100/100 102s 1s/step - accuracy: 0.7645 - loss: 4.2004 - val_accuracy: 0.7719 - val_loss: 0.9663
Epoch 13/20
100/100 95s 953ms/step - accuracy: 0.7728 - loss: 4.4069 - val_accuracy: 0.7602 - val_loss: 1.0358
Epoch 13: early stopping
```

Figure 5: EfficientNet Training process

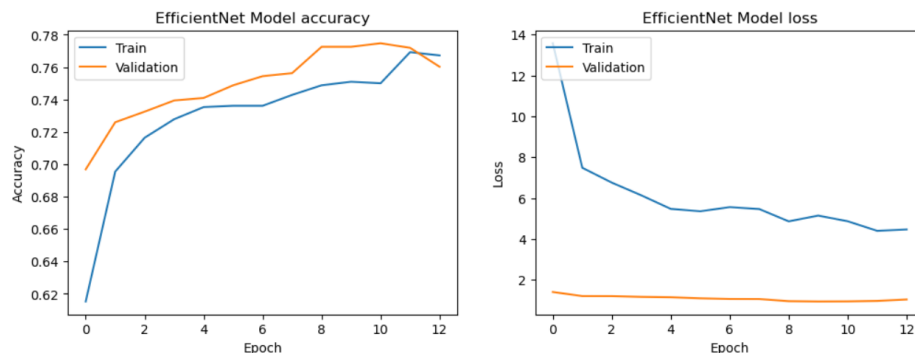
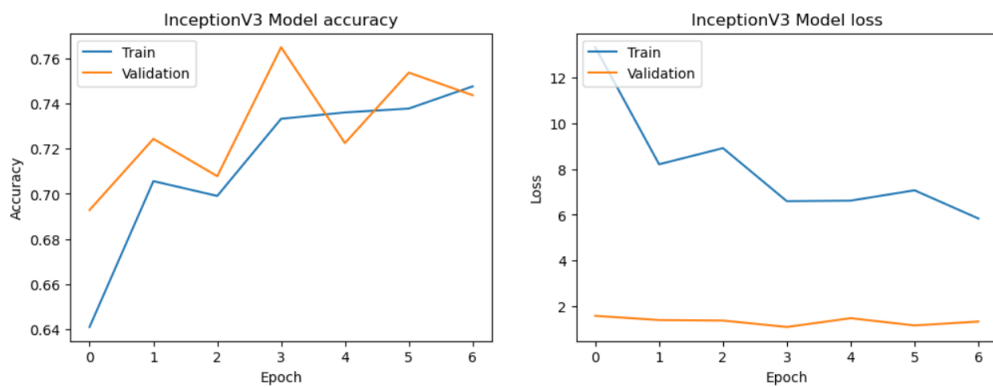


Figure 6: EfficientNet Accuracy and Loss

The InceptionV3 model was loaded from the TensorFlow Keras using pre-trained weights. Similar to the EfficientNetB0 model, the final output layer consisted of number of classes and the global average pooling layer and dense layer were used to build the model with softmax activation. The model was trained using the training and validation data generators and the training process is seen in Figure 7. Again, class weights were added to handle the imbalance. The training and validation accuracy and loss graphs were plotted as shown in Figure 8.

```
Epoch 1/20
100/100 — 179s 2s/step - accuracy: 0.5816 - loss: 18.7419 - val_accuracy: 0.6928 - val_loss: 1.5909
Epoch 2/20
100/100 — 154s 2s/step - accuracy: 0.6981 - loss: 8.9457 - val_accuracy: 0.7244 - val_loss: 1.4066
Epoch 3/20
100/100 — 147s 1s/step - accuracy: 0.7081 - loss: 8.2684 - val_accuracy: 0.7078 - val_loss: 1.3839
Epoch 4/20
100/100 — 144s 1s/step - accuracy: 0.7175 - loss: 7.2262 - val_accuracy: 0.7650 - val_loss: 1.1036
Epoch 5/20
100/100 — 132s 1s/step - accuracy: 0.7357 - loss: 6.5525 - val_accuracy: 0.7225 - val_loss: 1.4873
Epoch 6/20
100/100 — 108s 1s/step - accuracy: 0.7372 - loss: 7.2042 - val_accuracy: 0.7538 - val_loss: 1.1707
Epoch 7/20
100/100 — 109s 1s/step - accuracy: 0.7520 - loss: 5.5905 - val_accuracy: 0.7437 - val_loss: 1.3396
Epoch 7: early stopping
```

**Figure 7: Inception Training process**



**Figure 8: Inception Accuracy and Loss**

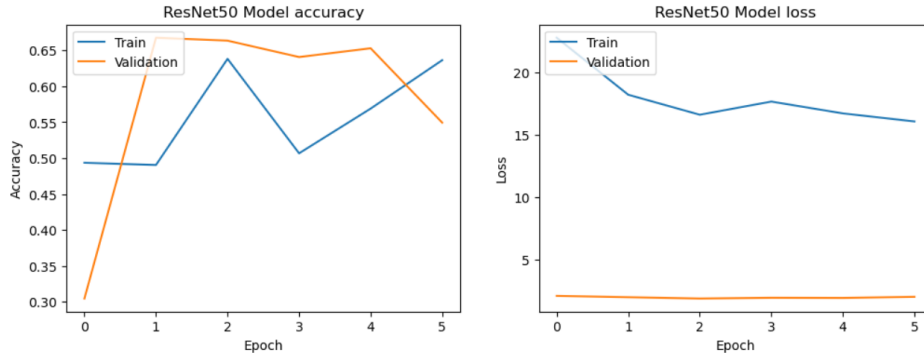
Similar to the EfficientNet and Inception model building and training process, the ResNet50 pre-trained model was loaded from TensorFlow Keras. The custom final layer consisted of the number of classes in the dataset. The global average pooling and the dense layer were used for building the model. The softmax activation method was used with Adam optimiser and the entire training process is shown in Figure 9. The training data generator was used to train the model on the training dataset and class imbalance was handled by applying class weights. The plots for model accuracy and loss on training and validation set are shown in Figure 10.

```

Epoch 1/20
100/100 — 257s 2s/step - accuracy: 0.4415 - loss: 29.2507 - val_accuracy: 0.3047 - val_loss: 2.0673
Epoch 2/20
100/100 — 234s 2s/step - accuracy: 0.4888 - loss: 17.9921 - val_accuracy: 0.6675 - val_loss: 1.9595
Epoch 3/20
100/100 — 226s 2s/step - accuracy: 0.6492 - loss: 17.1038 - val_accuracy: 0.6634 - val_loss: 1.8553
Epoch 4/20
100/100 — 222s 2s/step - accuracy: 0.4904 - loss: 18.8067 - val_accuracy: 0.6406 - val_loss: 1.9158
Epoch 5/20
100/100 — 208s 2s/step - accuracy: 0.6013 - loss: 16.4201 - val_accuracy: 0.6528 - val_loss: 1.9033
Epoch 6/20
100/100 — 175s 2s/step - accuracy: 0.6402 - loss: 16.2205 - val_accuracy: 0.5494 - val_loss: 1.9914
Epoch 6: early stopping

```

**Figure 9: ResNet Training process**



**Figure 10: ResNet Accuracy and Loss**

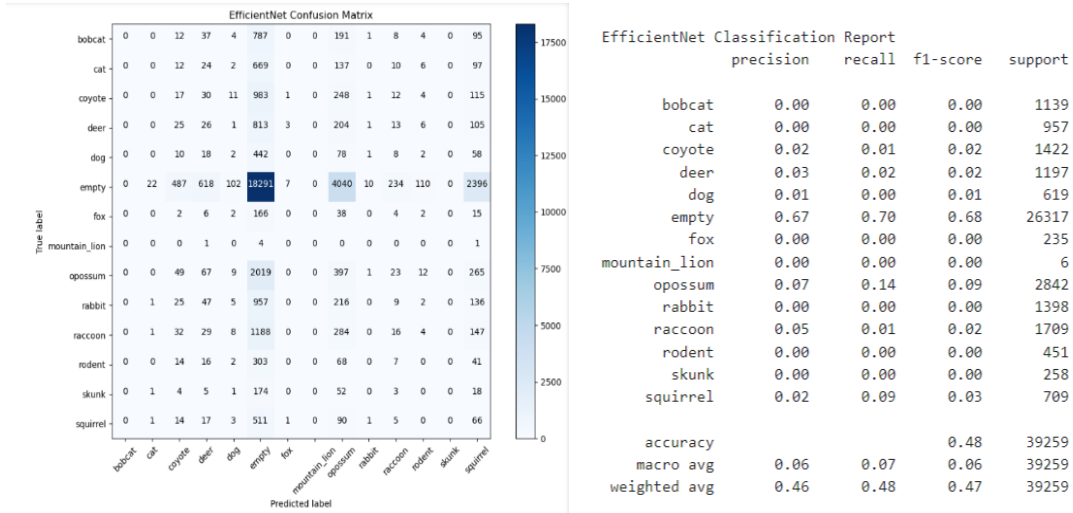
## 6 Evaluation

The next process after training the models is to evaluate the models on unseen data or test data. The subsections below discuss the performance of each model on the test dataset and the performance metrics.

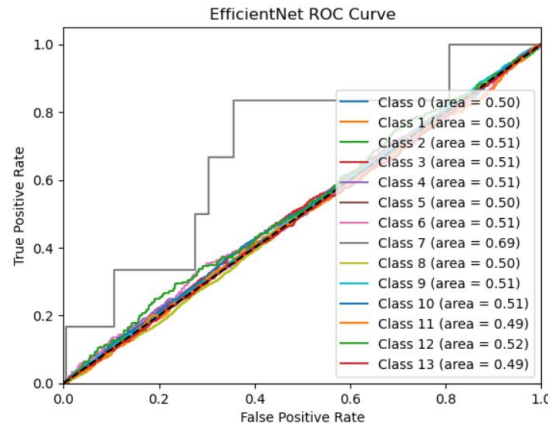
### 6.1 EfficientNet

As shown in the previous Figures 5 and 6, the maximum number of epochs was 20 but due to early stopping, the training stopped at epoch 13. Initially, training accuracy was 0.495 and gradually improved to 0.7778. The validation accuracy also follows a similar trend. The training loss was initially 21.82 and gradually decreased to 4.40. A similar trend was seen with validation loss. The model learns well however, in the end, there is a slight divergence indicating that when the model is beginning to fit the training data well, it is starting to overfit.

The test data generator is set. The test data does not contain the labels so the `class_mode` parameter was set to 'None'. The trained model was used for generating the predictions on the test set. The class labels were added to each image based on the highest probability. A random set of 30 images was displayed with the predicted labels. Finally, the model is saved as a '.keras' file. Confusion matrix, ROC curve and the Classification Report are shown in Figures 11 and 12.



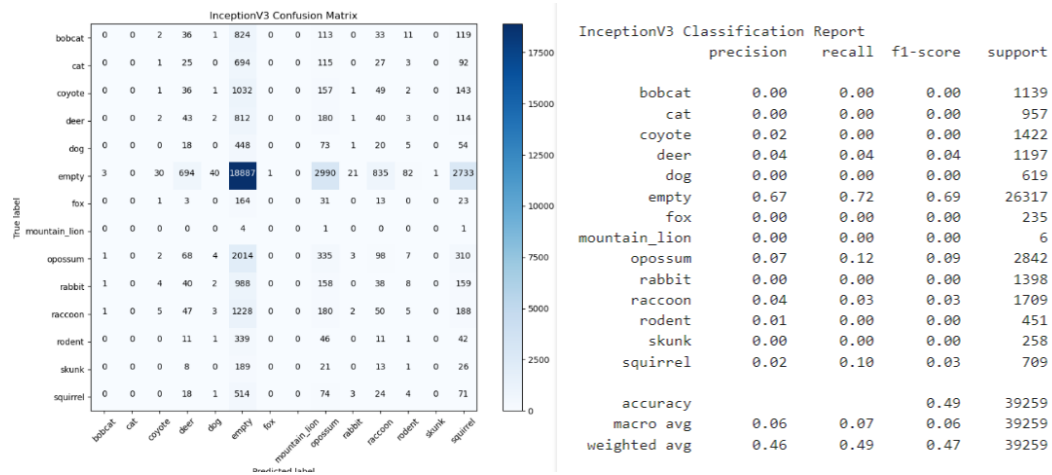
**Figure 11: Confusion Matrix and Classification Report for EfficientNet**



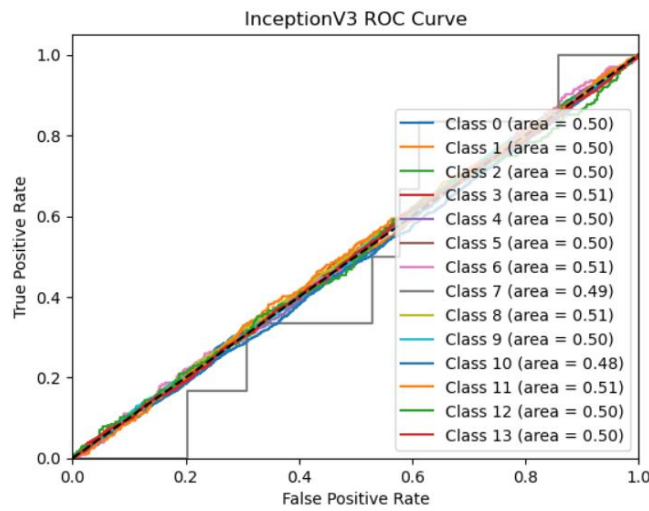
**Figure 12: ROC for EfficientNet**

## 6.2 Inception

As shown in the previous Figures 7 and 8, due to early stopping, the model was trained till epoch 7 to prevent overfitting. The training accuracy improved from 0.5816 to 0.7520. With some fluctuations, validation accuracy reached 0.7437. The training and validation loss was 5.5905 and 1.3396 respectively. Despite the fluctuations in validation accuracy and loss, the InceptionV3 model shows a positive learning trend. The test data generator is prepared for the test dataset and similar to EfficientNet, the 'class\_mode' is set to 'None'. Predictions for the test images are generated with class labels based on the predicted probability. A random sample of 30 images with the predicted labels is displayed. The model is saved. Confusion matrix, Classification Report, and ROC Curve for Inception are displayed in Figures 13 and 14.



**Figure 13: Confusion Matrix and Classification Report for Inception**



**Figure 14: ROC for Inception**

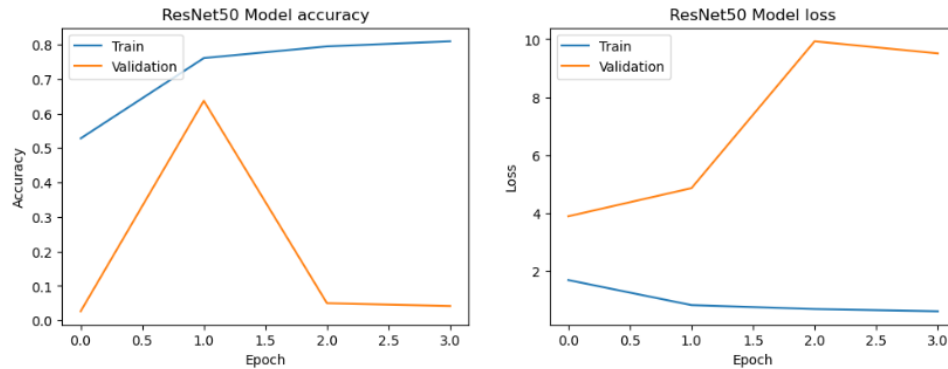
### 6.3 ResNet

From Figures 9 and 10, it can be seen that training stopped at epoch 6. The training accuracy was 0.6402 and loss was 16.2205. Validation accuracy was 0.5494 and loss was 1.9914. However, the model's performance is not consistent. To improve the performance, the model was fine-tuned and the training process is shown in Figure 15. From Figure 16, it can be seen that there is a possibility of overfitting even after fine-tuning the model as there is a sudden increase in validation loss and decrease in validation accuracy. Similar to EfficientNet and Inception, the test generator parameters are set and predictions are made on the test set. A random sample of 30 images with the predicted labels is displayed. The Confusion Matrix, Classification Report, and ROC Curve are shown in Figures 17 and 18.

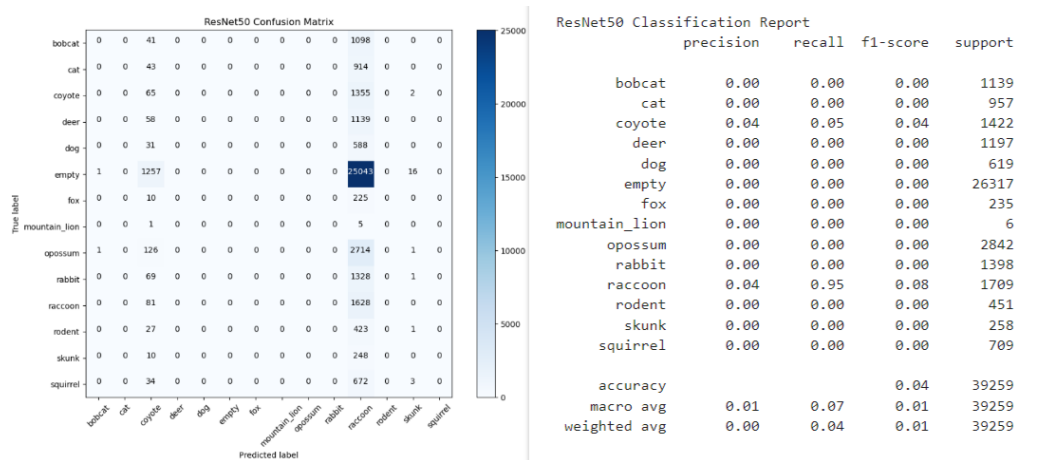


Epoch 1/50  
**100/100** ————— **441s** 4s/step - accuracy: 0.3174 - loss: 2.3359 - val\_accuracy: 0.0266 - val\_loss: 3.8891 - learning\_rate: 1.0000e-05  
Epoch 2/50  
**100/100** ————— **310s** 3s/step - accuracy: 0.7551 - loss: 0.8703 - val\_accuracy: 0.6369 - val\_loss: 4.8596 - learning\_rate: 1.0000e-05  
Epoch 3/50  
**100/100** ————— **356s** 4s/step - accuracy: 0.7917 - loss: 0.7083 - val\_accuracy: 0.0500 - val\_loss: 9.9254 - learning\_rate: 1.0000e-05  
Epoch 4/50  
**100/100** ————— **397s** 4s/step - accuracy: 0.8078 - loss: 0.6192 - val\_accuracy: 0.0419 - val\_loss: 9.5092 - learning\_rate: 1.0000e-05  
Epoch 4: early stopping

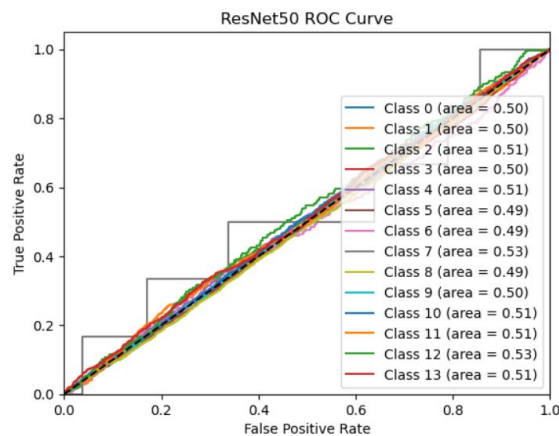
**Figure 15: ResNet Training process after fine-tuning**



**Figure 16: ResNet Accuracy and Loss after fine-tuning**



**Figure 17: Confusion Matrix and Classification Report for ResNet**



**Figure 18: ROC for ResNet**

## 6.4 Ensemble

The saved Keras files which contained the pre-trained models for EfficientNet, Inception, and ResNet were loaded and predictions were generated on the validation set. The predictions are averaged to create an ensemble prediction. The goal was to use the strengths of all three models and reduce the individual bias of each model. The ensemble model accuracy was calculated and was found to be 0.5510. The predictions were generated. A random sample of 30 images from the validation set was displayed with the predicted labels. The Confusion Matrix, Classification Report, and ROC curve are shown in Figures 19 and 20.

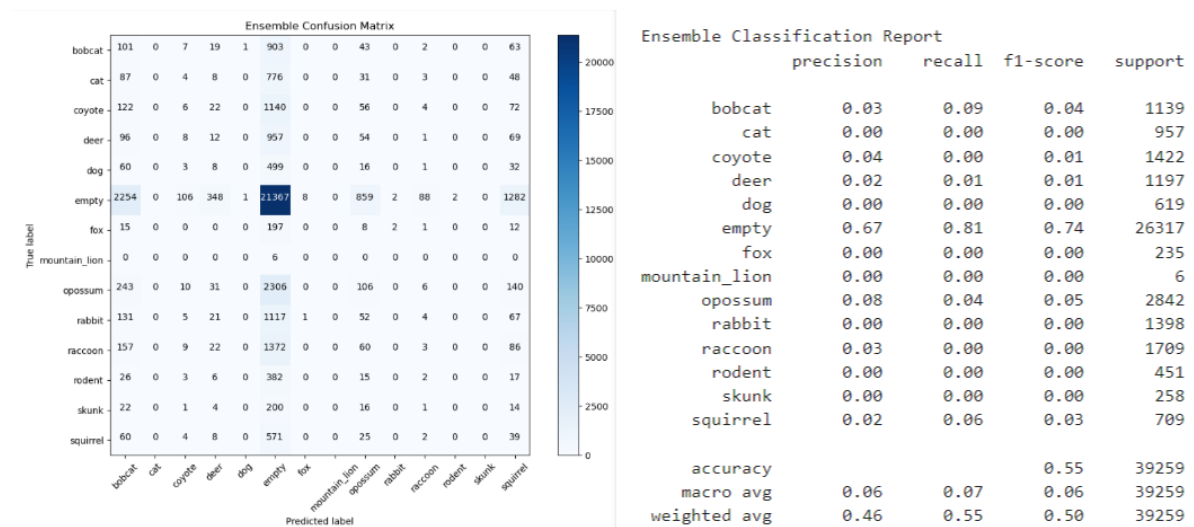


Figure 19: Confusion Matrix and Classification Report for Ensemble

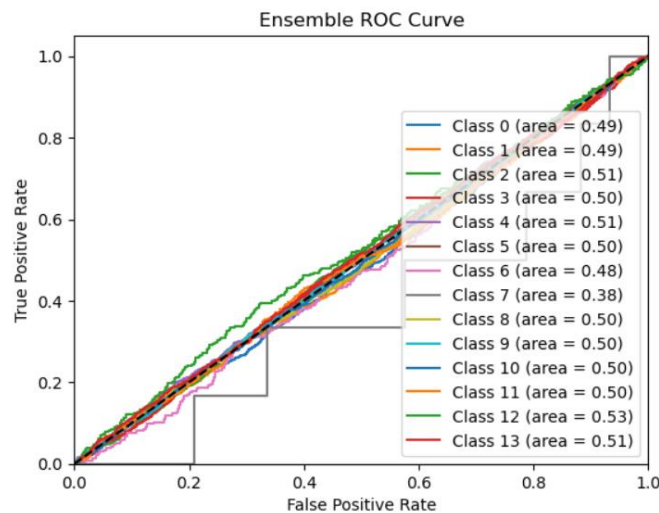


Figure 20: ROC for Ensemble

## 6.5 Discussion

EfficientNet Confusion Matrix and Classification Report from Figure 11 indicate that 'empty' category has the largest number of correct predictions (18,291). The categories 'opossum' and 'raccoon' have been often misclassified as 'empty'. This indicates bias to some degree. The precision for 'empty' class was 0.67, recall was 0.70, and F1-score was

0.68. Despite, handling the class imbalance issue while training the model, the performance of the model is average with an overall accuracy of 0.48. The AUC scores for most of the classes are close to 0.50 as seen in Figure 12. Most of the curves are marginally above the diagonal. The model's performance was not satisfactory.

The Inception Confusion Matrix and Classification Report shown in Figure 13 indicate that the highest number of correct predictions (18,887) was achieved for the 'empty' class. Mostly 'rabbit', 'raccoon', and 'opossum' were predicted as empty. Some empty images were also wrongly predicted as opossum and squirrel. In spite of managing class imbalance, the model still seems to be biased towards the empty class. The 'empty' class had the highest precision of 0.67, recall of 0.72, and F-1 score of 0.69. Classes like 'bobcat', 'cat', 'dog', and 'mountain\_lion' do not have any correct predictions. The overall accuracy is 0.49. Most of the classes have AUC around 0.50 as seen in Figure 14. The InceptionV3 model performs slightly better than the EfficientNetB0 in terms of the overall accuracy as it could identify certain classes better. InceptionV3 had slightly better performance on classes like 'squirrel' and 'raccoon'.

The ResNet50 Confusion Matrix and Classification report as shown in Figure 17 indicate that there are 25,043 correct predictions for empty class. Even after handling the class imbalance and fine-tuning the ResNet50 model, its performance was below average. The overall accuracy seems to be low with only 0.04. The ROC curve shown in Figure 18 indicates that the AUC for most of the classes is close to 0.50.

The Ensemble Confusion Matrix and Classification report seen in Figure 19 shows that 21,367 correct predictions for empty class is made. The ensemble seems to reduce the extreme bias towards empty class. Thus, proving that the ensemble helped balance the predictions to some extent. The overall accuracy is 0.551 with macro average F1-score of 0.06 which is very low and weighted average F1-score of 0.46 which is comparatively better. The F1-score for empty class was 0.74, recall was 0.81, and precision was 0.67. The ensemble model scores indicate that the model was quite good at predicting actual empty images as empty but also some non-empty images are classified as empty. From Figure 20, the AUC values are around 0.50 to 0.52 and the ROC curves are close to the diagonal indicating that the performance of the ensemble is average. Combining the predictions into an ensemble did increase the overall accuracy but the performance was still average.

The results from these models suggest that even though the class imbalance was handled and models were used individually for identifying species or even when fine-tuned or combined in an ensemble, performed only satisfactorily as the dataset was highly imbalanced. This can be seen in the existing literature that discusses the need for advanced data augmentation techniques, or the development of specialised architecture to handle such heavy class imbalances (Buda, Maki, and Mazurowski, 2018). The current implementation discussed is not yet ready for real-world deployment for the use of ecologists or wildlife conservationists where accurate animal identification is crucial for population count and conservation of biodiversity. Therefore, more sophisticated models or approaches must be explored before the application of these models in the real world.

## 7 Conclusion and Future Work

The main research question for this study is restated: How can deep learning models like EfficientNet, Inception, and ResNet be improved to enhance the accuracy in wildlife species identification? In order to answer this, the mentioned models were developed and trained and the predictions were combined in an ensemble to improve the overall identification performance. The ensemble performed better with an accuracy of 0.55 and a slightly better weighted average F1-score. Thus, combining models did improve the overall performance

and the hypothesis can be accepted. This study deemed to be partially successful in identifying animal species. Despite handling class imbalance during training, fine-tuning ResNet and implementing an ensemble of models, all models revealed a strong bias towards empty class images. While the ensemble did perform slightly better, it does not seem sufficient to classify all animal species correctly. There were certain limitations identified in this research study. The dataset was large but also highly imbalanced and diversity seemed to be limited. Even the train dataset and test dataset were from different locations, this could be a possible reason for a shift in the domains or variability in the type of images captured. Future research could include using advanced data augmentation techniques to create a balanced dataset. Specialised models tailored to handle class imbalances and train large datasets must be developed to identify species. This proposed future work could help make the identification of animal species more advanced and automated by addressing the challenges faced in this study.

## References

- Abood, B.S.Z., M., B.M., Almoossawi, Z., Shilpa, N. and Shakir, A.M., 2023. Revolutionizing Wildlife Monitoring: A Novel Approach to Camera Trap Image Analysis with YOLOv5. In *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, Tumkur, India, pp.1-6. doi: 10.1109/ICMNWC60182.2023.10435785.
- Beery, S., Morris, D. and Perona, P., 2019. The iWildCam 2019 Challenge Dataset. *arXiv (Cornell University)*. doi: <https://doi.org/10.48550/arxiv.1907.07617>.
- Buda, M., Maki, A. and Mazurowski, M.A., 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, pp.249-259. doi: <https://doi.org/10.1016/j.neunet.2018.07.011>.
- Chen, G., Han, T.X., He, Z., Kays, R. and Forrester, T., 2014. Deep convolutional neural network based species recognition for wild animal monitoring. In *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, France, pp.858-862. doi: 10.1109/ICIP.2014.7025172.
- Deng, C., Zhou, G. and Cai, Y., 2023. Wildlife Monitoring and Identification based on Faster R-CNN. In *2023 International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, Dalian, China, pp.638-642. doi: 10.1109/AEECA59734.2023.00119.
- Diana, S.M., Porselvi, R., Geetha, B., Vanitha, B., Muthukannan, K. and Ravi, S., 2024. Experimental Evaluation of Systematic Animal Classification System using Advanced Deep Learning Principle. In *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore, India, pp.1-6. doi: 10.1109/IITCEE59897.2024.10467699.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), pp.37-54.
- Figuerola, K., Camarena-Ibarrola, A., García, J. and Villela, H.T., 2014. Fast Automatic Detection of Wildlife in Images from Trap Cameras. In *Progress in Pattern Recognition*,

*Image Analysis, Computer Vision, and Applications*, 8827, pp.471-478. doi: 10.1007/978-3-319-12568-8\_57.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp.770-778. doi: 10.1109/CVPR.2016.90.

Hossen, M.R., Alfaz, N., Sami, A., Tanim, S.A., Bin Sarwar, T. and Islam, M.K., 2023. An EfficientNet to Classify Monkeypox-Comparable Skin Lesions Using Transfer Learning. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, Berlin, Germany, pp.1-6. doi: 10.1109/COINS57856.2023.10189311.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84-90. doi: 10.1145/3065386.

Kumar, R., Bhatti, S., Ruk, S.A. and Pathan, N., 2022. Identification of Endangered Animal species of Pakistan using Classical and Ensemble Approach. In *2022 2nd International Conference on Computing and Machine Intelligence (ICMI)*, Istanbul, Turkey, pp.1-5. doi: 10.1109/ICMI55296.2022.9873801.

K.S.M.E., A.R.D.K.M.E., S.P. and Senthilvelan, S., 2024. Birds Species Identification Using Deep Learning Model. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, Chennai, India, pp.1-7. doi: 10.1109/ADICS58448.2024.10533638.

Manohar, N., Sharath Kumar, Y.H. and Kumar, G.H., 2016. Supervised and unsupervised learning in animal classification. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, pp.156-161. doi: 10.1109/ICACCI.2016.7732040.

Nadarashalingham, T., Kumaralingam, L., Thanabalasingam, K., Thirunavukkarasu, J. and Ratnarajah, N., 2023. Automatic Identification of Sri Lankan Butterfly Families using Ensemble Deep Learning. In *2023 5th International Conference on Advancements in Computing (ICAC)*, Colombo, Sri Lanka, pp.633-638. doi: 10.1109/ICAC60630.2023.10417475.

O'Connell, A., Nichols, J.D. and Karanth, K.U., 2011. *Camera traps in animal ecology: Methods and analyses*. Springer. doi: 10.1007/978-4-431-99495-4.

Palanisamy, V. and Ratnarajah, N., 2021. Detection of Wildlife Animals using Deep Learning Approaches: A Systematic Review. In *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter)*, Colombo, Sri Lanka, pp.153-158. doi: 10.1109/ICter53630.2021.9774826.

Priya, P.K., Vaishnavi, T., Selvakumar, N., Kalyan, G.R. and Reethika, A., 2023. An Enhanced Animal Species Classification and Prediction Engine using CNN. In *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, Namakkal, India, pp.730-735. doi: 10.1109/ICECAA58104.2023.10212299.

Samarasinghe, B.S. and Lakmali, K.B.N., 2023. A Deep Ensemble Learning Approach for Venomous & Non-Venomous Serpents and Insect Species Identification. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, Ravet IN, India, pp.1-6. doi: 10.1109/ASIANCON58793.2023.10270077.

Shaik, F.J. and V.G., 2024. Automated Bird Detection using Snapshot Ensemble of Deep Learning Models. In *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore, India, pp.1-6. doi: 10.1109/IITCEE59897.2024.10467481.

S.K.L. and Edison, A., 2022. Wild Animal Detection using Deep learning. In *2022 IEEE 19th India Council International Conference (INDICON)*, Kochi, India, pp.1-5. doi: 10.1109/INDICON56171.2022.10039799.

Subhi, M.A. and Ali, S.M., 2018. A Deep Convolutional Neural Network for Food Detection and Recognition. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, Sarawak, Malaysia, pp.284-287. doi: 10.1109/IECBES.2018.8626720.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp.2818-2826. doi: 10.1109/CVPR.2016.308.

Tabak, M.A., Norouzzadeh, M.S., Wolfson, D.W., Sweeney, S.J., Vercauteren, K.C., Snow, N.P., Halseth, J.M., Di Salvo, P.A., Lewis, J.S., White, M.D., Teton, B., Beasley, J.C., Schlichting, P.E., Boughton, R.K., Wight, B., Newkirk, E.S., Ivan, J.S., Odell, E.A., Brook, R.K., Lukacs, P.M., Moeller, A.K., Mandeville, E.G., Clune, J. and Miller, R.S., 2018. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4), pp.585-590. doi: 10.1111/2041-210X.13120.

Tan, M. and Le, Q.V., 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv*, abs/1905.11946.

Wang, F., He, P., Zhang, T. and Liang, D., 2022. Wildlife detection algorithm based on Inv-YOLOv5m. In *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, Yantai, China, pp.1-6. doi: 10.1109/PHM-Yantai55411.2022.9941895.

Wang, Y., Zhang, Y., Feng, Y. and Shang, Y., 2022. Deep Learning Methods for Animal Counting in Camera Trap Images. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, Macao, China, pp.939-943. doi: 10.1109/ICTAI56018.2022.00143.