Counting Convolutional Neural Networks for Classification for Wildlife Conservation

Anna Edmonds
Stanford University
edmondsa@stanford.edu

Cheyenne Ali Sadeghi
Stanford University
csadeghi@stanford.edu

Vivek Brahmatewari
Stanford University
viranvi@stanford.edu

1. Abstract

The classification of animal species from images is crucial for wildlife conservation, biodiversity studies, and automated monitoring systems. Accurate image classification quickens species identification, enhancing the efficiency and accuracy of ecological studies. In this study, we address this problem by employing four convolutional neural networks (CNNs): EfficientNetB7, ResNet, Inception V3, and DenseNet, to classify images depicting ten animal categories. Through comparative analysis, our goal is to identify the most effective architecture for this classification task. We build upon foundational insights from previous research in animal image classification and leverage transfer learning to develop accurate classifiers, even with limited datasets. Our experiments demonstrate that EfficientNetB7 outperforms ResNet and DenseNet, achieving the highest accuracy while maintaining computational efficiency. This underscores the effectiveness of transfer learning in enhancing classification accuracy and highlights the potential of deep learning techniques for biodiversity conservation and wildlife monitoring. Our findings provide valuable insights for future research in optimizing deep learning models for real-world applications, such as wildlife trapcams, and contribute to the preservation of global biodiversity.

2. Introduction

The classification of animal species from images is a significant problem with implications for wildlife conservation, biodiversity studies, and automated monitoring systems [9]. Accurate image classification facilitates researchers in swiftly identifying and cataloging species, thereby enhancing the efficiency and accuracy of ecological studies [8]. This problem is important as it can greatly aid conservation efforts and scientific research by providing reliable data on species distribution and population by using the Shanon Index to verify biodiversity once we have the classification and count of species [16].

The input to the algorithm developed in this study is a set of images, each depicting a few animals in one image from one of ten categories: dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, and elephant. To achieve the classification, three different convolutional neural networks (CNNs) were employed: EfficientNetB7, ResNet, and DenseNet. The output of the algorithm is the predicted animal category for each input image. By comparing these models, the goal was to identify the most effective architecture for this classification task. The effectiveness of the models will be identified by comparing their F1 scores and top-1 test accuracies.

.

3. Related Work

The current literature provided a strong foundation for us when utilizing deep learning techniques in animal image classification. One article in particular, titled "Animal image identification and classification using deep neural networks techniques" (2023) presented a broad review of various neural networks, such as CNNs and their applications in animal image classification, establishing the general landscape and challenges of this task [1]. Similarly, "Experimental Evaluation of Systematic Animal Classification System using Advanced Deep Learning Principle" (2024) and "Insights and approaches using deep learning to classify wildlife" (2019) discuss models like VGG, ResNet, and generic CNNs [10][6]. These articles provide historical perspectives on the evolution of these models and detailed evaluation metrics, such as precision, recall, and F1-score, that are crucial for benchmarking and understanding model performance in animal classification tasks. Our approach for metric analysis parallels this, as we also use F1-score, precision and recall.

For our Inception V3 and DenseNet models, we acquired code from the Szegedy et al. and Huang et al. arxiv papers, respectively, that first introduced this architecture. However, we took a much different approach then the one specified in each paper. The Inception V3 model used by Szegedy et al. was trained with stochastic gradient utilizing the TensorFlow distributed machine with batch size 32

for 100 epochs. Their earlier experiments used momentum with a decay of 0.9, and their best models used RMSProp with decay of 0.9 and $\epsilon = 0.1$. They used a learning rate of 0.045, decayed every two epoch using an exponential rate of 0.94. In addition, gradient clipping with threshold 2.0 was found to be useful to stabilize the training. Finally, their model evaluations were performed using a running average of the parameters computed over time [12].

For the DenseNet models trained by Huang et al., on CIFAR and SVHN they trained using batch size 64 for 300 and 40 epochs, respectively. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. On ImageNet, they trained models for 90 epochs with a batch size of 256. The learning rate is set to 0.1 initially, and is lowered by 10 times at epoch 30 and 60 [3].

In comparison, when we trained our DenseNet models, we started with a learning rate of 0.00001 for training and tuning, but ultimately changed it to 0.001 for better results. We trained for 10 and 50 epochs and tuned for 25 and 50 epochs, for a total of 4 combinations. In our training setup, we employed a dynamic learning rate adjustment strategy using the ReduceLROnPlateau callback. This method reduces the learning rate by a factor of 0.2 if there is no improvement in the validation loss for three consecutive epochs, with the learning rate never going below a minimum threshold of 1e-6. This approach helps in fine-tuning the model more effectively by making smaller updates to the weights when progress in loss reduction stalls, thereby aiding in achieving a more precise convergence.

Focusing on the other models, "Animal Classification and Recognition Using Deep Learning EfficientNetB4" (2023) highlights the efficiency and accuracy of the EfficientNet family, particularly EfficientNetB4, in classifying animal images [5]. The study demonstrates the model's capability to achieve high accuracy with fewer parameters compared to traditional models, though it does not provide a direct comparison with other advanced architectures like ResNet and DenseNet. On the other hand, "AnimNet: An Animal Classification Network using Deep Learning" (2021) introduces a custom network tailored specifically for animal classification, showcasing how specialized architectures can be optimized for this domain to improve accuracy and efficiency [2]. However, these studies often lack comprehensive comparisons with other advanced models, which limits their ability to provide a complete performance assessment.

Comparative analyses and the use of transfer learning are well-covered in several studies. "Improving the Accuracy of Animal Species Classification in Camera Trap Images Using Transfer Learning" (2024) conducts a thorough comparison of models including ResNet, DenseNet, and others using transfer learning techniques[17]. This study provides robust performance analyses, demonstrating how transfer learning can significantly enhance model accuracy on camera trap images by leveraging pre-trained weights. Similarly, "Animal Species Recognition with Deep Convolutional Neural Networks from Ecological Camera Trap Images" (2023) evaluates VGG, ResNet, and generic CNNs, offering detailed evaluation metrics such as confusion matrices and ROC curves to assess model performance[7]. Furthermore, "Identifying animal species in camera trap images using deep learning and citizen science" (2018) and "Automated Recognition of Wild Animal Species in Camera Trap Images Using Deep Learning Models" (2023) emphasize the importance of transfer learning in improving model generalization and provide technical insights into model architecture and training processes[14][**?**]. These articles highlight the strengths and weaknesses of different models, the significance of transfer learning, and the necessity of using appropriate metrics, making them highly relevant for your project comparing EfficientNetB7, ResNet, and DenseNet for animal image classification.

4. Methods

To determine the most effective model for animal image classification, multiple pre-trained models were used from Keras to train an overarching transfer learning model. Afterwards, the transfer learning models were tested on a test set and the accuracies of each model were compared.

4.1. Loss Function

A softmax loss function was used during training. Softmax loss converts the raw output scores of a neural network into probability distributions over the class labels. This helps in clearly distinguishing between multiple classes by assigning higher probabilities to the correct class and lower probabilities to the incorrect ones. It also reacts effectively to blurry or distorted images. This quality makes softmax loss ideal for image classifying since raw image data is not typically consistent in resolution or quality.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

4.2. Transfer Learning

Transfer learning involves training a network on a base dataset , then transferring those features to a second network to be trained on a different dataset. Due to the vast computational and time resources needed to train a convolutional neural network from scratch, a transfer learning model was used for this image classifier. In this experiment, all models had been pre-trained with the ImageNet dataset. After a pre-trained model had been loaded into the classifier, all layers except for the input layer were frozen. VGG-16,

VGG-19, and Xception are typical pre-trained models used for image classifiers, however, the merits of EfficientNet-B7, ResNet152-V2, and DenseNet121 as pre-trained models in a transfer learning model were tested for this classifier. For all three models we attempted to set up Early Stopping, Learning Rate Reduction, and Model Checkpoint callbacks to monitor the training process and save the best model weights. After the pre-trained models were used to transfer their features in the initial training process, the model was fine-tuned with another training run in which the layers were unfrozen.

4.3. Models

4.3.1 EfficientNet B7

EfficientNet-B7 was used as a pre-trained model in training the image classifier. EfficientNet is a convolutional neural network architecture that uniformly scales depth, width, and resolution using a compound scaling method, as opposed to the typical approach of arbitrarily modifying network properties [[13]]. EfficientNet-B7, specifically, improves upon the complexity of the base EfficientNet-B0 by adding more layers. It's used in image classification because of its high accuracy without compromising computational efficiency. Figure 1 shows the architecture of EfficientNet-B7.

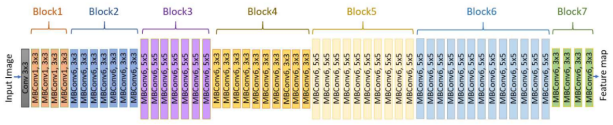

Figure 1. EfficientNet-B7 model architecture

4.3.2 ResNet152-V2

ResNet152V2 is one of the pre-trained convolutional neural network models we used for classifying the images and finding the best models. This model had several parameters, including the inclusion of the top fully connected layer, pre-trained weights from the ImageNet dataset, and an input shape. The architecture comprises multiple residual blocks that are organized into four main stages. Each stage of ResNet is responsible for processing specific spatial dimensions of the input image as seen in Figure 2. [4]

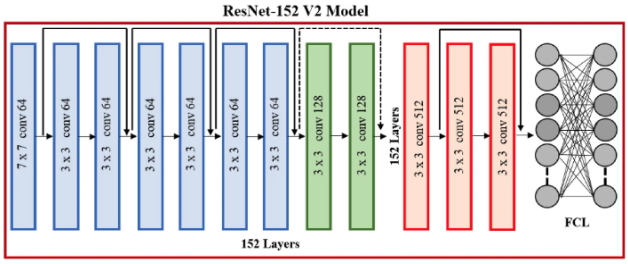

Figure 2. ResNet152-V2 Architecture

Each residual block features convolutional layers, batch normalization, and ReLU activations, enhanced by shortcut connections that ensure efficient gradient flow and mitigate the vanishing gradient problem. The network concludes with a global average pooling layer followed by a fully connected layer with a softmax activation function, which is particularly effective for classification tasks. For this model we compiled the model with sparse categorical cross-entropy as the loss function and used Adam optimizer.

4.4. DenseNets

The Dense Convolutional Network (DenseNet) enhances traditional convolutional networks by establishing direct connections from each layer to every other layer in a feed-forward manner. DenseNet's structure, with L(L+1)/2 connections mitigates the vanishing-gradient issue, boosts feature propagation, fosters feature reuse, and reduces the overall number of parameters. DenseNet eliminates the vanishing-gradient problem, as each layer has direct access to the gradients from the loss function and the original input signal. It is remarkably efficient, too. DenseNet also integrates features through transition layers that consist of batch normalization, a 1x1 convolution, and a 2x2 average pooling operation, which help in managing the feature map sizes and improving the computational efficiency. [3]

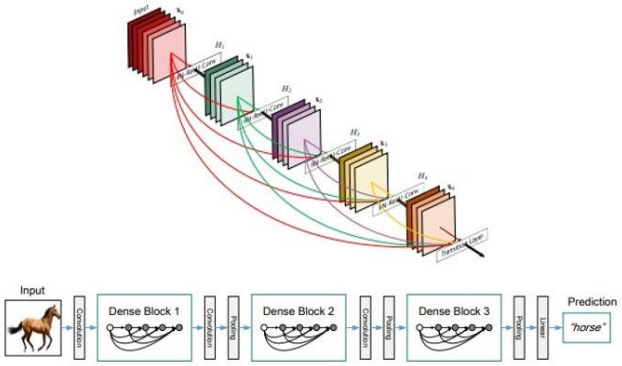

Figure 3. DenseNet 121 Architecture

DenseNet-121, DenseNet-169, and DenseNet-201, differ mainly in their depth, with 121, 169, and 201 layers,

respectively. Each variant is designed to handle increasingly complex image recognition tasks: DenseNet-121 offers a balance between efficiency and performance for simpler applications, DenseNet-169 provides enhanced accuracy for more complex tasks with its additional layers, and DenseNet-201, the deepest model, is best suited for highly detailed and nuanced tasks such as advanced medical imaging due to its superior feature extraction capabilities. We tested all three types.

5. Dataset

5.1. Dataset Description

The dataset utilized for this project comprises approximately 28,000 (224 x 224 pixel) images, each categorized into one of the ten specified animal classes as seen in Figure 5. These images were sourced from Google Images and verified to ensure quality by Corrado Alessio. To simulate real-world conditions, the dataset includes some erroneous data, representing images that users of an application might take. We had to identify the erroneous data to run our models appropriately.

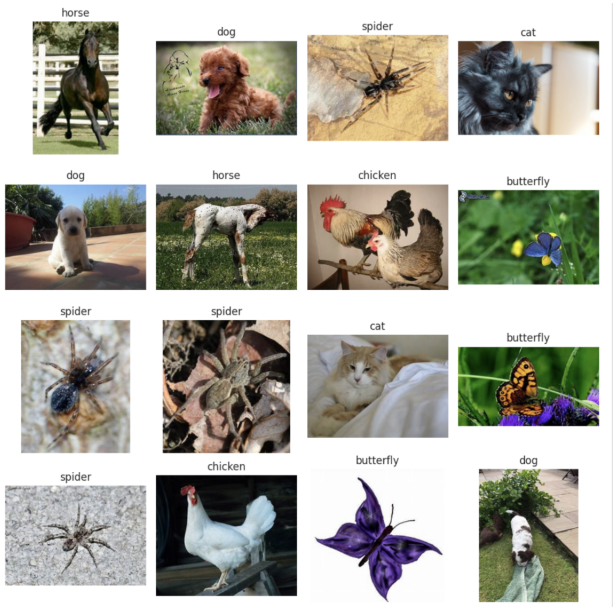

Figure 4. Classification of Dataset and Labels

The data is organized into a main directory with subfolders for each animal category, with the number of images per category ranging from 2,000 to 5,000, however, there was a greater amount of images for butterfly, spider, and dog than for the other classes which is seen in Figure 14 in the Appendix. This diverse collection includes variations in lighting, background, and pose, providing a robust foundation for training and evaluating image classification models.

5.2. Data Preprocessing

Several preprocessing steps were applied to the data before training the models. Images were synthesized into a dataframe and split into training and test sets using an 80-20 ratio. The ImageDataGenerator from Keras was utilized for data augmentation and preprocessing, including splitting the data into training, validation, and test subsets. The data generator also facilitated the splitting of the dataset into training and validation sets, with 20% of the data reserved for validation. For the test data, we used a separate generator. Each model took in its own preprocess input. We initially usde the 'train_test_split' function from 'sklearn.model_selection' to split the data into training and test datasets. The train dataframe contains 80% of the data and is used for both training and validation. The test data frame contains the remaining 20% of the data and is used for testing. From the training dataframe, it futher split, 80% for the training, and 20% for the validation. We load the data into the model using the flow_from_dataframe method. This validation split was critical for monitoring the model's performance on unseen data during training and ensuring that the model did not overfit.

5.3. Data Augmentation

To further enhance the training process, data augmentation techniques were employed as part of the data preprocessing. For this project, the 'ImageDataGenerator' class from Keras was employed to apply a series of image augmentations. The augmentations applied included rescaling, horizontal flipping, vertical flipping, and rotation. Specifically, re-scaling was performed by normalizing pixel values to the [0, 1] range, which aids in faster training and better performance. Additionally, images were randomly rotated within a 20-degree range to enhance the model's robustness to rotational variations as seen in the Appendix in Figure 15.

6. Experimental Results and Discussion

6.1. Result Metrics

Top-1 accuracy was used as the metric when training the model and during testing. Top-1 accuracy score refers to the percent of images for which the correct class was within the top-1 predicted scores.

The F1 score is a metric commonly used in classification tasks, particularly when dealing with imbalanced datasets where the class distribution is uneven. It combines precision and recall into a single value, providing a balanced assessment of a classifier's performance.

Precision measures the proportion of true positive predictions (correctly classified instances) out of all positive predictions made by the model. It is calculated as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions out of all actual positive instances in the dataset. It is calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The F1 score is the harmonic mean of precision and recall, given by:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score considers both false positives and false negatives, making it particularly useful when the cost of misclassification is high or when there is an imbalance between classes. It provides a single metric that balances the trade-off between precision and recall, offering a more comprehensive evaluation of a classifier's performance compared to accuracy alone.

In a confusion matrix, the rows represent the instances belonging to the actual classes, while the columns represent the instances assigned to the predicted classes. A very accurate model will have a saturated matrix diagonal.

6.2. Hyperparameters

A learning rate of 0.00001 was used during the transfer learning and model fine-tuning steps. This learning rate was chosen because the pre-trained models are already loaded with useful features, thus a small learning rate was chosen to avoid disrupting the features. A learning rate reduction callback was also used in both the transfer learning and the model fine-tuning steps. This allowed for dynamic changing of the learning rate if the validation losses were to plateau. For the optimizer, the adam optimizer was used. The mini-batch size was chosen to be 32. This size achieves a balance between computational efficiency and stable gradient estimates, allowing for more efficient use of GPU memory and computational resources. Lastly, no cross-validation was performed, as it was deemed too computationally expensive for this model. However, for all three DenseNet models, a learning rate of 0.001 was used during the transfer learning while a 0.00001 learning rate was used for the model fine-tuning steps. This achieved higher test accuracy results for all three models. Also, for the Inception V3, learning rates of 0.00001, 0.0001, 0.001, and 0.002 were all tried for the transfer step but all failed to produce accurate results.

6.3. DenseNet Results

All DenseNet models were run with four different epoch value combinations. Not only would this allow us to determine the optimal values for these parameters, but also allows for efficient evaluation of the model's performance under various training durations, providing insights into the balance between training complexity and accuracy gains. This systematic approach ensures that the model is not only accurate but also optimized for computational efficiency, preventing overfitting and unnecessary resource expenditure. The results of DenseNet-121 and DenseNet-169 can be found in the appendix. Here we just focus on DenseNet-201, the best performing of the trio tested. Before tuning the model we get a validation loss of 0.274 and a validation accuracy of 90.41%. The pair consisting of 50 transfer learning epochs and 25 tuning epochs performed the best after tuning, with a test accuracy of 94.44%.

Figure 5. Validation and training curves for accuracy (left) and loss(right) of the DenseNet-201 model

Per usual, the difference between the training and validation curves indicates a lack of underfiting and overfiting as a result of this model. With an F1 score of 0.92, it proves to be a decently reliable image classifier. This accurate model has, as expected, a clean, colored confusion matrix diagonal.

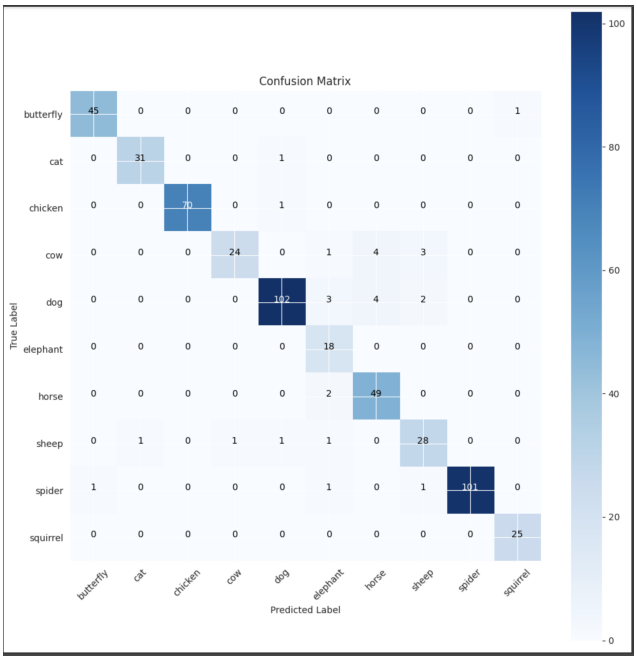

Figure 6. Confusion matrix of the DenseNet-201 model

In this display, we present both the predicted and actual labels for a randomly selected set of images. All predicted labels accurately match the actual labels, showcasing the model's effectiveness. As we initially hypothesized, the DenseNet-201 model outperformed other DenseNet variants. This superior performance can likely be attributed to its increased number of layers, which allow for deeper and more detailed processing of features.

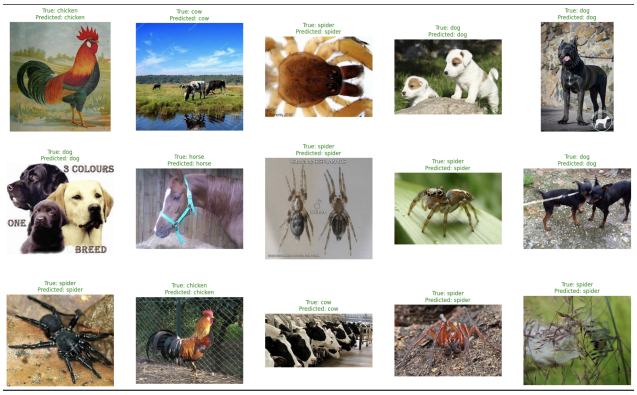

Figure 7. Predicted labels of the DenseNet-201 model for a test set

6.4. EfficientNet-B7 Results

The EfficientNet-B7 based model was run with four different epoch value combinations to determine the optimal values for these parameters. Shown in Table 6 in the appendix, the pair consisting of 10 transfer learning epochs and 25 tuning epochs performed the best, with a test accuracy of 97.32%.

After the model was run with these hyperparameters, the plots shown in figure 8 were produced. The plots in figure 8 indicate a healthy difference between the training and validation curves. This means that there is neither underfiting nor overfiting as a result of this model. Figure 31 in the appendix shows the classification report for the EfficientNet-B7 based model. With an F1 score of 0.9645, it proves to be a very reliable image classifier.

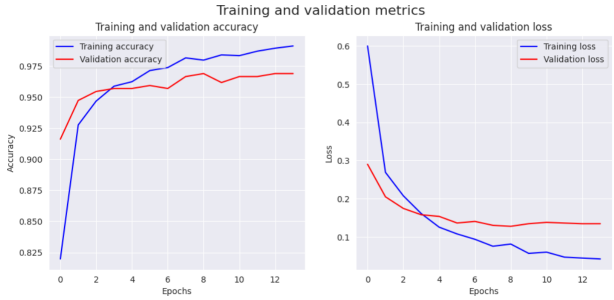

Figure 8. Validation and training curves for accuracy (left) and loss(right) of the EfficientNet-B7 model

A confusion matrix can help visualize the performance of a classification model. In Figure 9, the confusion matrix of the EfficientNet-B7 model shows the frequencies of the predicted label matching the true label. A very accurate model will have a colored confusion matrix diagonal.

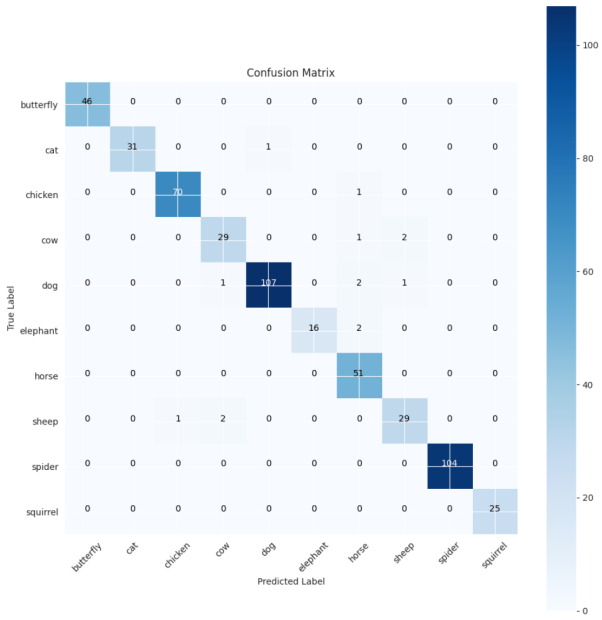

Figure 9. Confusion matrix of the EfficientNet-B7 model

Figure 10 shows the predicted and the actual labels of a random set of images. All of the predicted labels are correct, showing the model's robustness.

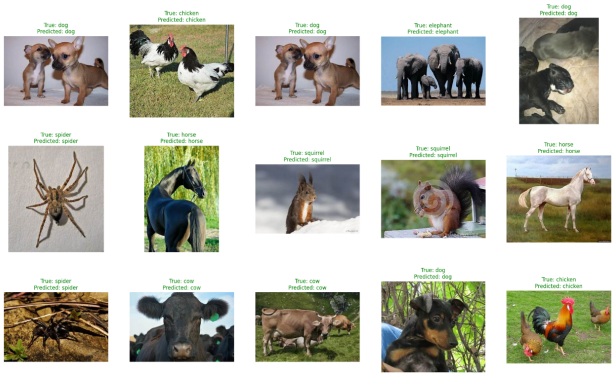

Figure 10. Predicted labels of the EfficientNet-B7 model for a test set

6.5. ResNet Results

The ResNet-based model was run with four different epoch value combinations to determine the optimal values for these parameters. As shown in Table 5 in the Appendix, the pair consisting of 50 transfer learning epochs and 25 tuning epochs performed the best, with a test accuracy of 95.93

Before tuning the model we get a validation loss of 0.389 and a validation accuracy of 86.71% as seen in Figure 29 in the appendix. After tuning the model we get a validation loss of 0.373 and a validation accuracy of 89.25%.

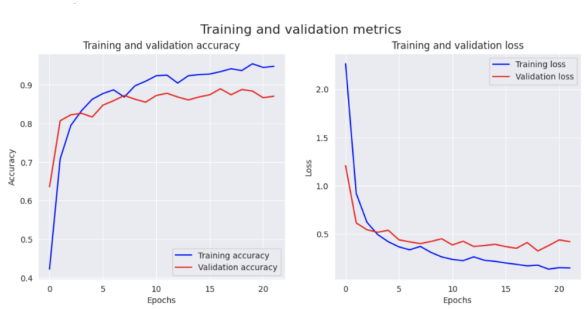

Figure 11. Validation and training curves for accuracy (left) and loss(right) of the ResNet152-V2 model

We evidently see that after doing hyperparameter tuning we have an increase in validation accuracy but not to the same extent that EfficientNet had.

As seen in Figure 30 in the Appendix, the overall F1-score of the model is 0.948, indicating a high level of accuracy and robustness in classifying images across various categories. The F1-score, which balances precision and recall, demonstrates that the model performs exceptionally well in differentiating between the classes, even those with fewer training examples. This can be further seen when we look at the predicted labels vs. the true labels in Figure 19 below where out of the 15 images one was predicted incorrectly.

The majority of predictions align correctly along the diagonal of the confusion matrix as seen in Figure X, signifying high accuracy. For instance, butterflies, cats, chickens, cows, and spiders are predominantly classified correctly, reflecting the model's robust feature learning for these categories. Specific metrics reinforce this, with F1-scores of 0.96 for butterflies, 0.98 for cats, and 0.97 for spiders, underscoring high precision and recall which is seen in Figure 30 in the Appendix.

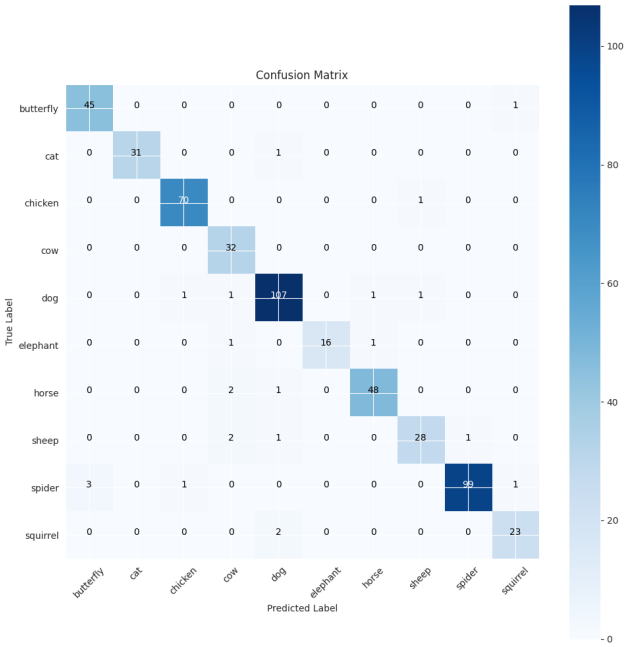

Figure 12. Confusion Matrix ResNet152-V2 model

However, the confusion matrix as seen in Figure 12 also uncovers some misclassifications. For example, a few dogs are misclassified as cats, cows, horses, and elephants, and some spiders are mistaken for butterflies and chickens. The elephant and squirrel categories show slightly higher misclassification rates, indicating potential areas for refinement. Elephants, with an F1-score of 0.94, and squirrels, with an F1-score of 0.92, illustrate these challenges as seen in Figure 29.

Figure 13. Predicted labels of the ResNet152-V2 model for a test set

6.6. Comparison of Models

Overall the four models did comparatively well, EfficientNet, ResNet, Inception V3, and DenseNet as expected. Among the tested models, Inception V3 struggled significantly with the complexity of the dataset, likely due to its architecture not being well-suited for the task. DenseNet models showed improved performance with increasing depth, where DenseNet 201 outperformed DenseNet 121 and 169 because of its ability to capture more complex features. ResNet excelled in capturing complex features owing to its deep architecture and residual learning mechanism. EfficientNet-B7 outperformed all models due to its balanced scaling of depth, width, and resolution, capturing complex features effectively. In terms of hyperparameter sensitivity, Inception V3's performance was significantly affected by hyperparameter tuning, indicating high sensitivity. Conversely, DenseNet demonstrated robustness to different epoch values, with consistent performance improvements through tuning. ResNet maintained high performance across various hyperparameter settings, suggesting stable training dynamics. EfficientNet-B7 showed excellent performance with minimal sensitivity to hyperparameter changes, indicating robust training dynamics.

Model	Test Accuracy (%)	F1 Score
DenseNet 121	92.15	0.915
DenseNet 169	93.30	0.913
DenseNet 201	94.44	0.926
ResNet	95.93	0.948
EfficientNet-B7	97.32	0.964

Table 1. Test Accuracy and F1 Scores of Different Models

In terms of accuracy and F1 score, Inception V3 had the lowest test accuracy and F1 score, indicating overall poor performance. DenseNet 121 and DenseNet 169 had similar F1 scores, respectively of 0.9159 and 0.9134 and highest test accuracy of 92.15% and 93.30%. These were outperformed by the deeper DenseNet variants, DenseNet 201 achieving a test accuracy of 94.44% and an F1 score of 0.92. ResNet achieved high accuracy and an F1 score of 0.948 and highest test accuracy of 95.93%, indicating better performance than DenseNet. However, EfficientNet-B7 achieved the highest accuracy and F1 score, with a test accuracy of 97.32% and an F1 score of 0.9645, demonstrating the best overall performance. Among the four models, EfficientNet-B7 stands out as the most effective for this classification task, followed by ResNet and DenseNet 201. Inception V3, while architecturally innovative, did not perform well due to its sensitivity to image size and hyperparameters. Both DenseNet and ResNet models demonstrated strong feature learning capabilities, with EfficientNet-B7's balanced scaling providing an edge in handling complex datasets.

7. Conclusion and Future Work

In conclusion, our study comparing the performance of EfficientNet-B7, ResNet, Inception V3, and DenseNet-121, 169, and 201 for animal image classification resulted in EfficientNet-B7 producing the most accurate results while maintaining high computational efficiency. Leveraging its optimized architecture and compound scaling method, EfficientNet-B7 demonstrates superior performance in accurately classifying animal images, albeit by a few percentage points from DenseNets and Resnet, making it a promising candidate for scientists who perhaps do not want to use the ImageNet or CIFAR datasets. Moreover, our findings underscore the effectiveness of transfer learning in developing accurate image classifiers, even with limited datasets. By fine-tuning pre-trained models, we can harness the knowledge learned from large datasets and apply it to smaller, domain-specific tasks, significantly improving classification accuracy. Furthermore, we could do this fine-tuning of the models if we change to torch instead as we could change the forwards and backward pass where there was limited work that could be down with Keras. Furthermore, for future work, we plan to ensure an equal number of images for each class. The current dataset had a significantly higher number of images for dogs and spiders compared to the other classes, which may have contributed to the higher overall accuracy. Equalizing the number of images across classes will help mitigate this imbalance and provide a more accurate assessment of model performance.

Future work could explore further optimizations of EfficientNet-B7 and investigate its performance on larger and more diverse datasets. Additionally, research efforts could focus on adapting the model for real-time applications and deploying it in resource-constrained environments, such as wildlife trap cameras. The implications of our results extend beyond the realm of deep learning, offering valuable insights into biodiversity conservation and wildlife monitoring. Accurate animal classification using

trap camera imagery can provide researchers and conservationists with invaluable data for assessing biodiversity, understanding ecosystem dynamics, and informing conservation strategies. [11] By leveraging advanced deep learning techniques like EfficientNet-B7 and transfer learning, we can enhance our ability to monitor and protect wildlife populations, ultimately contributing to the preservation of global biodiversity.

8. Contributions and Acknowledgements

Vivek implemented the solution, data importation, and evaluation. Vivek also trained and tuned the EfficientNet-B7 based model, and contributed to report writing. Anna helped with the data pre-processing augmentation and then implemented and trained the ResNet base model and contributed to the entire report writing. Cheyenne helped with implementing and training the Inception V3 and DenseNet models and contributed to the report writing.

Code based on a Kaggle Notebook [15]

A. Appendix

A.1. Data Representation

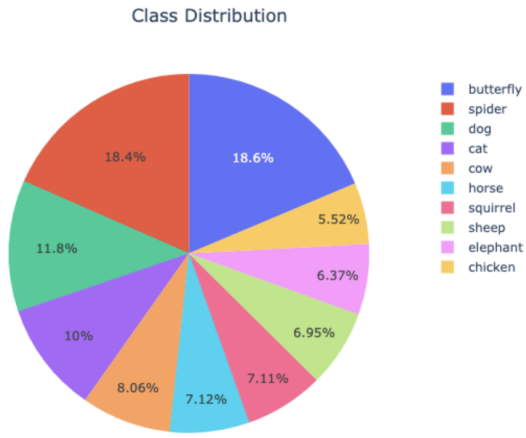

Figure 14. Classification of Dataset and Labels

Figure 15. Sample set of data after augmentation

A.2. Inception V3

Inception V3 is a convolutional neural network that has the factorization of larger convolutions into smaller ones, which helps reduce the parameter count and computational expense while maintaining network depth and capability. The architecture also employs label smoothing as a regularization technique to prevent the model from becoming

overly confident during training—a common cause of overfitting. This enhances its ability to generalize. [12]

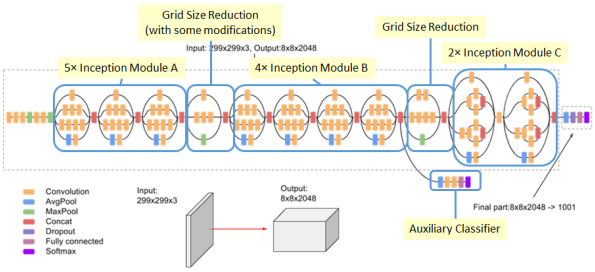

Figure 16. Inception V3 Architecture

Inception V3 utilizes auxiliary classifiers to assist in training, stabilizing the training process across the network's depth. The network's design structures convolutional filters to minimize redundancy and maximize processing speed. This includes the use of 1x1 convolutions to perform dimension reduction before applying more computationally expensive 3x3 and 5x5 convolutions.

A.2.1 Inception V3 Results

Inception V3 performed the worst out of all tested models. We hypothesize that this is because it expects a (299, 299) image size while our images were all of pixel size (224, 224). Inception V3's architecture may also not align well with the complexity or scale of features in the dataset. Additionally, the model's sensitivity to hyperparameters such as learning rate and batch size could be affecting its training effectiveness. Inception V3 is designed to be a balanced model in terms of width and depth, and our dataset contains highly complex features that require deeper or wider networks to capture effectively. Models we tested like DenseNet-201, which are deeper and allow more complex feature propagation, performed better. Finally, the architecture of Inception V3, with its mixture of convolutions of varying sizes at each module, is aimed at capturing information at various scales. It is possible that the scale of important features in the dataset did not align well with these sizes, resulting in the model struggling to effectively learn these features.

A.3. DenseNet-121

A.3.1 Results

Table 2. DenseNet-121 Hyperparameter Optimization

Transfer Learning Epochs	Tuning Epochs	Test Accuracy (%)
10	50 (early stopping at 6)	92.15
50	50 (early stopping at 10)	91.95
10	25 (early stopping at 13)	91.00
50	25 (early stopping at 6)	91.67

Before tuning, the validation accuracy of our DenseNet-121 model was 86.33% with 0.4123 validation loss.

Figure 17. Validation and training curves before tuning for DenseNet-121

Shown in the Table, the pair consisting of 50 transfer learning epochs and 50 tuning epochs performed the best after tuning, with a test accuracy of 91.95%. This was determined to be the worse performing of all DenseNet models.

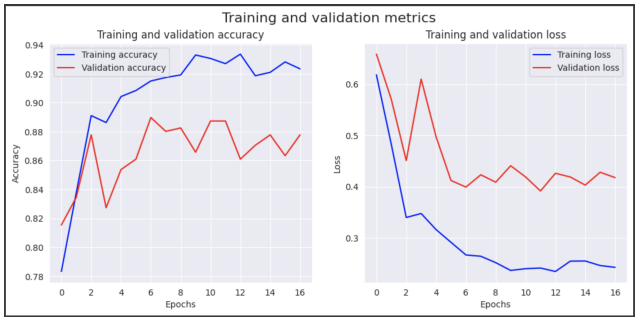

Figure 18. Validation and training curves for accuracy (left) and loss(right) of DenseNet-121

The difference between the training and validation curves indicates a lack of both underfiting and overfiting

as a result of this model. After the model was run with these hyperparameters, the plots shown in these figure were produced. With an F1 score of 0.9159, it proves to be a decently reliable image classifier. The next figure shows the classification report for the DenseNet-121 model.

F1 Score: 0.9159102349331707

	precision	recall	f1-score	support
butterfly	0.88	0.98	0.93	46
cat	1.00	0.97	0.98	32
chicken	1.00	0.97	0.99	71
cow	0.96	0.72	0.82	32
dog	0.97	0.93	0.95	111
elephant	0.78	1.00	0.88	18
horse	0.77	0.98	0.86	51
sheep	0.84	0.81	0.83	32
spider	1.00	0.93	0.97	104
squirrel	0.96	0.96	0.96	25
accuracy			0.93	522
macro avg	0.92	0.93	0.92	522
weighted avg	0.94	0.93	0.93	522

Figure 19. F1 score and classification report for DenseNet-121

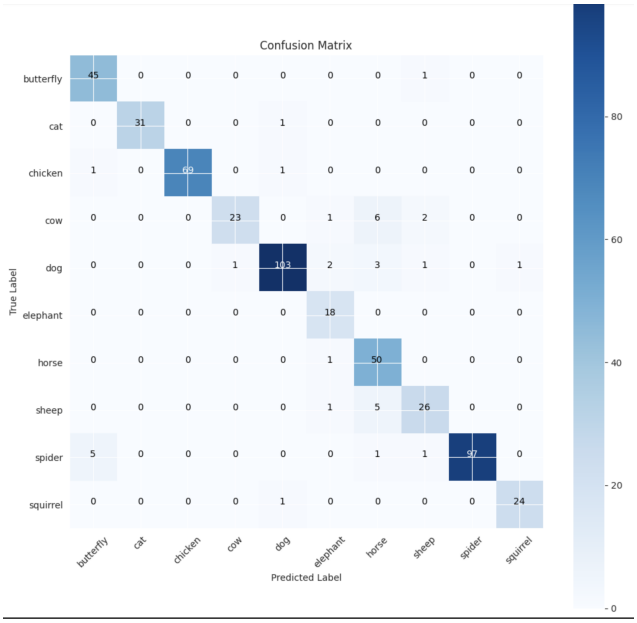

Figure 20. Confusion matrix of DenseNet-121

The last figure shows the predicted and the actual labels of a random set of images. Four of the predicted labels are incorrect, revealing a weakness in the model's ability to generalize.

Figure 21. Predicted labels of DenseNet-121 for a test set

A.4. DenseNet-169

A.4.1 Results

A.4.2 DenseNet 169

Table 3. DenseNet-169 Hyperparameter Optimization

Transfer Learning Epochs	Tuning Epochs	Test Accuracy (%)
10	50 (early stopping at 11)	90.61
50	50 (early stopping at 12)	93.30
10	25 (early stopping at 7)	92.34
50	25 (early stopping at 9)	90.80

Before tuning the model we get a validation loss of 0.274 and a validation accuracy of 90.41%.

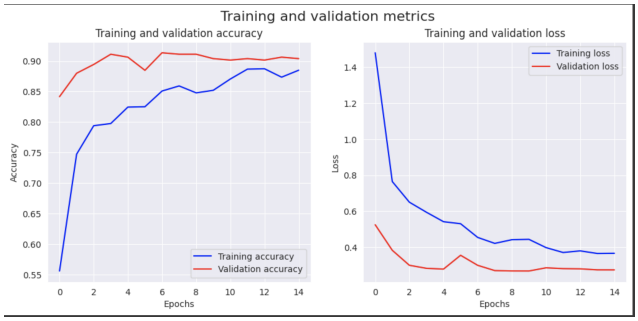

Figure 22. Validation and training curves before tuning

Shown in Table 2, the pair consisting of 50 transfer learning epochs and 50 tuning epochs performed the best after tuning, with a test accuracy of 93.30%.

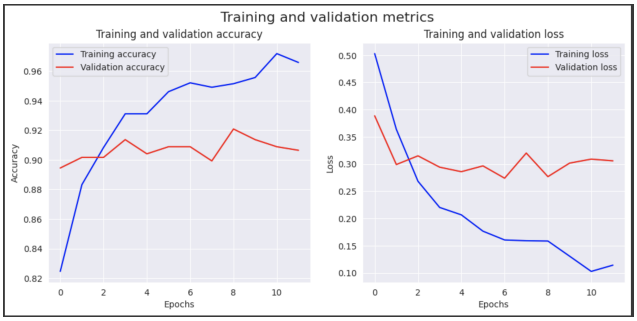

Figure 23. Validation and training curves for accuracy (left) and loss(right) of the DenseNet-169 model

The close alignment between the training and validation curves suggests that the model exhibits neither underfitting nor overfitting.

After executing the model with the specified hyperparameters, the resulting plots are depicted in the figures below.

With an F1 score of 0.9134, it demonstrates considerable reliability as an image classifier, and its confusion matrix validates this.

Figure 15 presents the classification report for the DenseNet-169 model.

F1 Score: 0.9134247866540737

	precision	recall	f1-score	support
butterfly	0.96	0.93	0.95	46
cat	0.94	1.00	0.97	32
chicken	0.99	1.00	0.99	71
cow	0.86	0.78	0.82	32
dog	0.99	0.89	0.94	111
elephant	0.81	0.94	0.87	18
horse	0.89	0.96	0.92	51
sheep	0.74	0.91	0.82	32
spider	0.99	0.95	0.97	104
squirrel	0.85	0.92	0.88	25
accuracy			0.93	522
macro avg	0.90	0.93	0.91	522
weighted avg	0.94	0.93	0.93	522

Figure 24. F1 score and classification report for the DenseNet-169 model

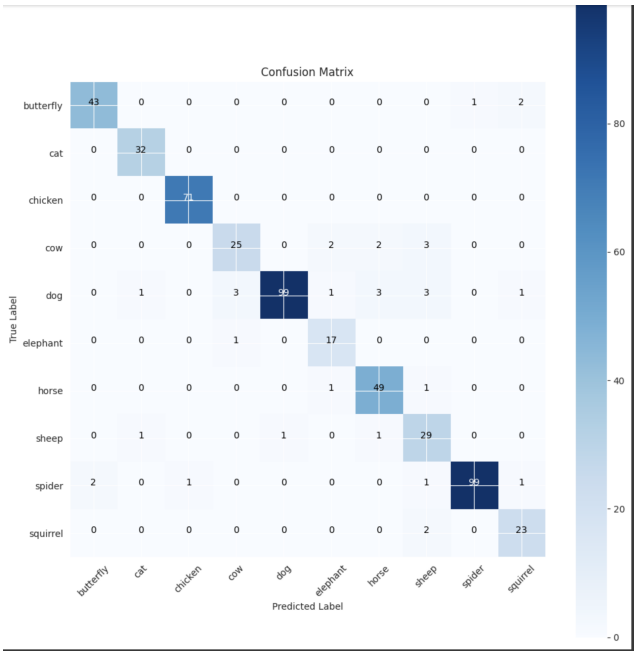

Figure 25. Confusion matrix of the DenseNet-169 model

The last figure shows the predicted and the actual labels of a random set of images. All but two of the predicted labels are correct, showing the model's robustness.

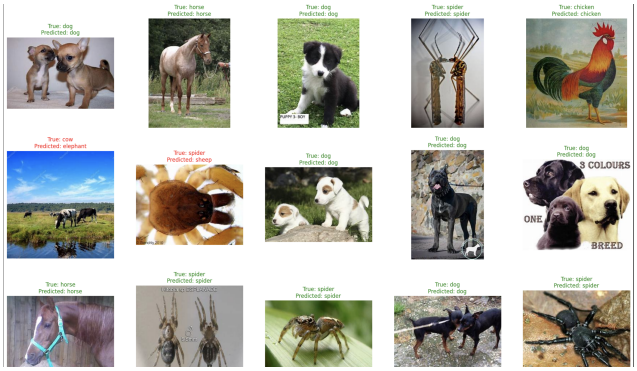

Figure 26. Predicted labels of the DenseNet-169 model for a test set

Table 4. DenseNet 201 Hyperparameter Optimization

Transfer Learning Epochs	Tuning Epochs	Test Accuracy (%)
10	50 (early stopping at 14)	92.15
50	50 (early stopping at 17)	93.10
10	25 (early stopping at 10)	92.34
50	25 (early stopping at 12)	94.44

H

```
F1 Score: 0.9266300463181352
              precision    recall  f1-score   support

   butterfly       0.98      0.98      0.98        46
         cat       0.97      0.97      0.97        32
     chicken       1.00      0.99      0.99        71
         cow       0.96      0.75      0.84        32
         dog       0.97      0.92      0.94       111
    elephant       0.69      1.00      0.82        18
       horse       0.86      0.96      0.91        51
       sheep       0.82      0.88      0.85        32
      spider       1.00      0.97      0.99       104
    squirrel       0.96      1.00      0.98        25

    accuracy                           0.94       522
   macro avg       0.92      0.94      0.93       522
weighted avg       0.95      0.94      0.95       522
```

Figure 28. F1 score and classification report for DenseNet-201

A.5. DenseNet-201

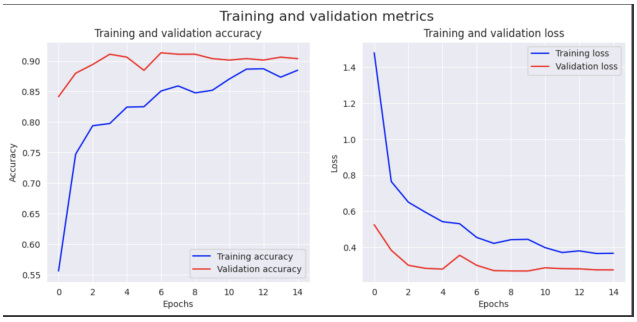

Figure 27. Validation and training curves before tuning for DenseNet 201

The next figure shows the classification report for the DenseNet-201 model.

A.6. ResNet

Figure 29. Validation and training curves before tuning

Table 5. ResNet Hyperparameter Optimization

Transfer Learning Epochs	Tuning Epochs	Test Accuracy (%)
10	50 (early stopping at 28)	92.79
50	50 (early stopping at 17)	94.83
10	25 (early stopping at 18)	93.32
50	25 (early stopping at 14)	95.93

```
F1 Score: 0.9480007303388612
              precision    recall  f1-score   support

   butterfly       0.94      0.98      0.96        46
         cat       1.00      0.97      0.98        32
     chicken       0.97      0.99      0.98        71
         cow       0.84      1.00      0.91        32
         dog       0.96      0.96      0.96       111
    elephant       1.00      0.89      0.94        18
       horse       0.96      0.94      0.95        51
       sheep       0.93      0.88      0.90        32
      spider       0.99      0.95      0.97       104
    squirrel       0.92      0.92      0.92        25

    accuracy                           0.96       522
   macro avg       0.95      0.95      0.95       522
weighted avg       0.96      0.96      0.96       522
```

Figure 30. F1 score and classification report

Table 6. EfficientNet Hyperparameter Optimization

Transfer Learning Epochs	Tuning Epochs	Test Accuracy (%)
10	50 (early stopping at 27)	95.79
50	50 (early stopping at 7)	94.83
10	25 (early stopping at 14)	97.32
50	25 (early stopping at 14)	96.93

A.7. EfficientNet-B7

F1 Score: 0.96453323418891

	precision	recall	f1-score	support
butterfly	1.00	1.00	1.00	46
cat	1.00	0.97	0.98	32
chicken	0.99	0.99	0.99	71
cow	0.91	0.91	0.91	32
dog	0.99	0.96	0.98	111
elephant	1.00	0.89	0.94	18
horse	0.89	1.00	0.94	51
sheep	0.91	0.91	0.91	32
spider	1.00	1.00	1.00	104
squirrel	1.00	1.00	1.00	25
accuracy			0.97	522
macro avg	0.97	0.96	0.96	522
weighted avg	0.97	0.97	0.97	522

Figure 31. F1 score and classification report of EfficientNet-B7 model

References

[1] T. Battu and D. S. Reddy Lakshmi. Animal image identification and classification using deep neural networks techniques. *Measurement: Sensors*, 25:100611, 2023. 1

[2] S. Binta Islam, D. Valles, T. J. Hibbitts, W. A. Ryberg, D. K. Walkup, and M. R. J. Forstner. Animal species recognition with deep convolutional neural networks from ecological camera trap images. *Animals*, 13(9), 2023. 2

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 3

[4] H. Malik, A. Naeem, A. Sadeghi-Niaraki, et al. Multi-classification deep learning models for detection of ulcerative colitis, polyps, and dyed-lifted polyps using wireless capsule endoscopy images. *Complex Intell. Syst.*, 10:2477–2497, 2024. Received 27 September 2022; Accepted 21 October 2023; Published 24 November 2023; Issue Date April 2024. 3

[5] P. Matapurkar. Animal classification and recognition using deep learning. *International Journal of Recent Development in Engineering and Technology*, 12(4):15–19, 2023. 2

[6] Z. Miao, K. Gaynor, J. Wang, et al. Insights and approaches using deep learning to classify wildlife. *Scientific Reports*, 9:8137, 2019. 1

[7] H. Nguyen, S. J. Maclagan, T. D. Nguyen, T. Nguyen, P. Flemons, K. Andrews, E. G. Ritchie, and D. Phung. Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 40–49, 2017. 2

[8] D. Oñoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 615–629, Cham, 2016. Springer International Publishing. 1

[9] S. Panigrahi, P. Maski, and A. Thondiyath. Real-time biodiversity analysis using deep-learning algorithms on mobile robotic platforms. *PeerJ Comput Sci*, 9:e1502, Aug 2023. 1

[10] Z. Song, W. Gong, C. Li, and T. T. Toe. Animals image classification method based on improved convolutional neural network. 10:1997–2001, 2022. 1

[11] Springer. A comprehensive overview of recent advances in deep learning techniques for wildlife conservation. *Complex Intelligent Systems*, 9, 2023. 9

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015. 2, 10

[13] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. Version 3, 23 Nov 2019. 3

[14] R. Thangaraj, S. Rajendar, S. M, R. S. K, S. Sasikumar, and C. L. Automated recognition of wild animal species in camera trap images using deep learning models. In *2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–5, 2023. 2

[15] V. Vence. Animal image classification using efficientnetb7. https://github.com/VinceVence/kaggle-notebooks/blob/75efd0162e3e7b198de97102f904cb1605757f97/animal-image-classification-using-efficientnetb7.ipynb. Accessed: 2024-06-09. 9

[16] Wildlife Conservation Society. Wildlife conservation society. https://www.wcs.org/, 2024. Accessed: 2024-05-17. 1

[17] M. Willi, R. T. Pitman, A. W. Cardoso, C. Locke, A. Swanson, A. Boyer, M. Veldthuis, and L. Fortson. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91. 2