Bird Image Classification using State of the Art Architectures and Transfer Learning

Edoardo Vaira[1] **Georgios Chavales**[2] **Han Fu**[3]

[1]edoardo.vaira@icloud.com [2]geochaval@gmail.com [3]han.fu.0251@student.lu.se

Abstract: Bird species identification is essential for various fields including ecology, conservation biology, ornithology, and biodiversity monitoring. Traditional identification methods, while valuable, are often labor-intensive and error-prone. This paper explores the use of deep learning, particularly Convolutional Neural Networks (CNNs), to automate bird species identification. Leveraging transfer learning from pre-trained models such as VGGNet, ResNet, and DenseNet, we developed models that significantly enhance the accuracy of bird species classification. We utilized a high-quality dataset sourced from eBird, comprising 507,000 images of 507 bird species from Sweden. This dataset was processed using YOLOv8 for image standardization. Our experiments demonstrated that DenseNet169 achieved the highest accuracy, outperforming other models. This research highlights the potential of deep learning models for effective bird species identification, suggesting avenues for further improvements in model robustness and real-time deployment.

1. Introduction

Bird species identification is fundamental in ecology, conservation biology, ornithology, and biodiversity monitoring. Accurate identification helps understand ecological dynamics and assess habitat health, providing insights into other organisms in the habitat. It is crucial for conserving endangered species by tracking and flagging them for protection. Bird watching has also become a popular recreational activity, offering relaxation and fascination through the observation of diverse bird species. Traditionally, bird identification relied on manual methods such as field guides, expert observations, and acoustic monitoring. While valuable, these methods are labor-intensive, time-consuming, and prone to human error and bias.

Recently, there has been growing interest in using artificial intelligence (AI) to automate bird species identification. One promising approach is the use of Convolutional Neural Networks (CNNs) [1], which have shown remarkable success in pattern recognition tasks like image classification. CNNs are trained on labeled images and can predict the class of an unlabeled input by learning to distinguish features across multiple layers. Although introduced theoretically in 1990, CNNs gained popularity in the past 10-15 years, driven by computational advancements and AlexNet's introduction [2]. In 2012, AlexNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), significantly outperforming traditional techniques with a top-5 accuracy of 83.6%, surpassing other submissions by an astonishing 10%, and benefiting from faster training times due to GPU implementation.

Following AlexNet's success, more sophisticated CNN models were developed, such as VGGNet [3], GoogLeNet (Inception) [4], ResNet [5], EfficientNet [6], and DenseNet [7]. VGGNet employs multiple 3x3 convolutional layers to increase depth and enhance feature extraction. GoogLeNet introduces the Inception module, optimizing computation with different-sized filters within the same layer. ResNet revolutionizes training deep networks with residual connections, enabling the training of much deeper networks by alleviating the vanishing gradient problem. EfficientNet scales CNNs systematically in depth, width, and resolution, while DenseNet enhances feature reuse by connecting each layer directly with every other layer in a feed-forward fashion.

These models have demonstrated improved performance in various computer vision tasks, including object detection, segmentation, and image recognition, further solidifying CNN dominance. They are widely used in their pre-trained forms for many computer vision problems through transfer learning. Pre-trained on large datasets like ImageNet [8], containing 1000 classes and approximately 1.2 million images, these models have learned to detect features across a wide variety of classes, generalizing exceptionally well to new classes and datasets. Transfer learning involves using these pre-trained models for feature extraction, modifying the output fully connected layers, and training only these layers. This approach saves significant time when training models on a novel dataset.

This paper addresses the challenge of bird species identification using transfer learning. Section 2 introduces our dataset, while Section 3 discusses the models and architectures explored. Section 4 presents the results of these models, and Section 5 concludes the paper and outlines further research directions.

2. Dataset

2.1 Data Source

First, we needed to find a reliable source for our data. We started with Kaggle [9], a well-known site for datasets. The one we downloaded was a good starting point to test our models on. But the image quantity and quality was poor. Additionally, the birds were randomly selected from around the world, making the dataset less useful for real-world applications.

Thus, we explored other sources and found eBird [10]. eBird is an online database of bird observations that covers the entire world since 2010. Users from around the globe can upload and share their bird sightings. The platform includes images, audio recordings, geographical data, descriptions, and many other statistics.

For this paper, we needed two things: images of the birds and their scientific names. These would be the inputs and outputs for our models respectively. An important feature that eBird offers is that users can rate the images. Which means that the highest-quality images have high ratings. We will explain why this feature was indispensable further on. Given these advantages listed above, we chose eBird as our data source.

2.2 Data Collection

To get the actual data from eBird, we wrote a scraper (program designed to extract data from websites) in Python. The scraper follows three customizable steps:

1. **Geographical Selection:** We can select the geographical area from which to get the data. Thus, we can create datasets tailored for specific cities, regions, countries, or even the entire world.

2. **Quantity Selection:** We can specify the number of images for each species. With this feature, we did not have to apply any data augmentation techniques. We always had the option to download extra images.

3. **Quality Selection:** Earlier, we talked about how users on eBird can rate each other's images. This feature comes in handy now. Our scraper downloads the highest-rated images by default. Additionally, eBird stores every image in various resolutions (160, 320, 480, 640, 900, 1200, 1800, 2400). This gave us the option to choose the resolution for our data.

At the end of this process, we have our dataset. The scraper stores each species in a separate folder, labeled with its scientific name. Each folder contains the specified number of images with the chosen quality. And the whole dataset is tailored for a specific geographical location.

Here is an example of such an image:

Figure 1. Coracias Garrulus

2.3 Data Standardization with YOLOv8

All the images downloaded from eBird have different dimensions. This is not ideal because our models need images with the same aspect ratio. To fix this, we standardized all the images to a square aspect ratio using YOLOv8 [11], a state-of-the-art object detection algorithm.

Here is how we used YOLOv8 to handle this issue:

- **Bird Detection:** YOLOv8 scans each image, detects birds, and marks them with a bounding box. It works by dividing the image into a grid, with each grid cell predicting bounding boxes and confidence scores for objects within that cell. Using a single neural network, it performs this task in one pass, making it fast and efficient. This makes the algorithm convenient for handling large amounts of data.

- **Selecting the Main Subject:** In images with multiple birds, the algorithm selects the bird with the highest detection confidence.

- **Cropping:** The image is then cropped to a square frame centered on the selected bird.

- **Exclusion of Non-Detections:** If YOLOv8 fails to find a bird, the image gets removed from the dataset.

You can see an example of this process in Figure 2.

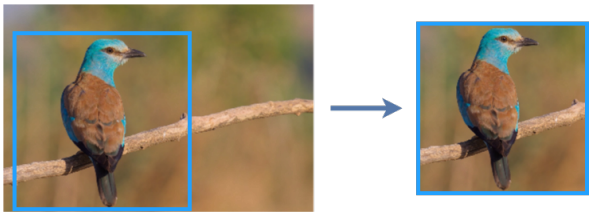

Figure 2. Cropping images using YOLOv8

2.4 Results

Given all the steps we have listed, this is how we generated our dataset for this project. We chose Sweden as our geographical area of interest, which is home to 507 bird species. For each species, we downloaded 1,000 images, totaling 507,000

images. We chose a resolution of 480, which was enough for effective model training. All of this resulted in a 20 GB dataset.

Next, we passed the dataset through YOLOv8 to crop the images, ensuring that all the images featured the bird and had the correct aspect ratio. Some images were also removed when YOLOv8 failed to detect any bird. This process resulted in an average of 940 images per class, with the smallest class having 513 images. So, the dataset size went from 20 GB to 5 GB.

At the end of this entire process, we obtained a high-quality dataset ready for our models to train on.

3. Modelling

In this paper in order to face the problem of bird species identification, we employ transfer learning. Transfer learning is a machine learning technique where a model trained on one task is adapted or fine-tuned for a different, but related, task. In the context of Computer Vision, transfer learning relates to using pre-trained CNN models, which have been trained on large-scale datasets, such as ImageNet, as feature extractors, instead of starting the learning process from scratch. ImageNet is a dataset that contains 1000 classes and approximately 1.2 million images. Therefore, these models have learned from a wide variety of pictures to detect features and patterns and have tuned the filters of their layers to act as these feature extractors. When using pre-trained models as feature extractors, we usually replace the last fully connected layer with a completely new one and we either freeze the feature extractor's weights and train the last fully connected layer on our new dataset, or we finetune the entire network on our new dataset. This is proven to yield a better performance fast, drastically saving time, as we do not need to train from the start deep models with millions of parameters. In this paper, we have used 3 kind of models: ResNet, VGGNet and DenseNet.

3.1 VGGNet

VGGNet is a CNN architecture proposed by the Visual Geometry Group at the University of Oxford. It is known for its simplicity and uniform architecture, consisting of multiple convolutional layers followed by fully connected layers. VGG architectures typically use small 3x3 convolutional filters with a stride of 1 and max-pooling layers to downsample feature maps. Despite its simplicity, VGG has achieved competitive performance on image classification tasks and served as a baseline for deeper architectures. The original VGG models, namely VGG16 and VGG19, have 16 and 19 weight layers, respectively. In this paper, we use the VGG19 version[3]. The model's architecture is depicted in Figure 3.

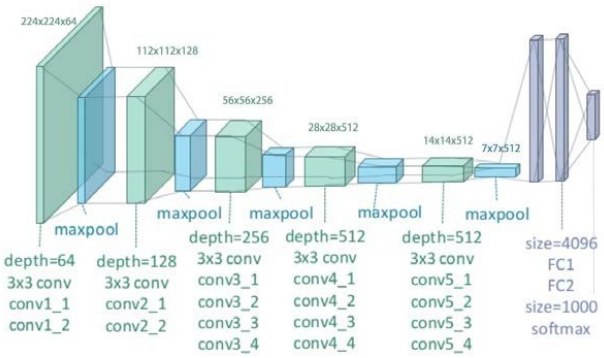

Figure 3. VGG19 architecture [12]

3.2 ResNet

ResNet (Residual Network) introduced the concept of residual connections, which alleviate the vanishing gradient problem in deep networks by enabling the direct flow of information through skip connections that bypass one or more layers. In very deep architectures, due to the nature of gradient descent, the weight gradient for the first layers of the model becomes close to zero, due to secuential derivations (vanishing gradient problem). ResNet, due to its skip connections counters this problem effectively, with weights of the earlier layers partaking in the derivatives of the later layers. Just like VGGNet, ResNet architectures come in different depths, including ResNet50, ResNet101, and ResNet152, which have 50, 101, and 152 weight layers, respectively. In this paper we use the ResNet101 version [5]. The model's architecture is depicted in Figure 4.

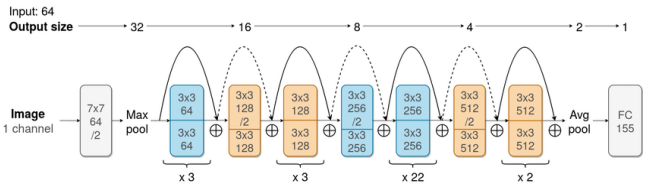

Figure 4. ResNet101 architecture [13]

3.3 DenseNet

DenseNet (Densely Connected Convolutional Network) is a CNN architecture that connects each layer to every other layer in a feed-forward fashion. With this, utilizing skip connections, it enhances gradient flow throughout the network like ResNet and has as a consequence dense feature maps. There are several versions of DenseNet, coming in different sizes and depths. In this paper we use the DenseNet169 version, which is the original version [7]. The model's architecture is depicted in Figure 5.

3.4 Ensemble model

Ensemble learning [15] is a machine learning technique that combines two or more individual models to produce better

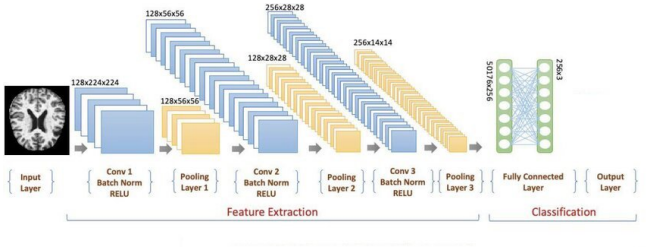

Figure 5. DenseNet169 architecture [14]

predictions. Also, ensemble learning could reduce the risk of overfitting due to the diversity of base models. In this paper, we choose ResNet101 and DenseNet169 as two base models. Then, we remove their classifier layers, concatenate the outputs of the penultimate layers and input it into a new classifier layer. After that, we train the ensemble model by freezing two base models' parameters and updating weights of the new classifier layer only. The model's architecture is depicted in Figure 6.

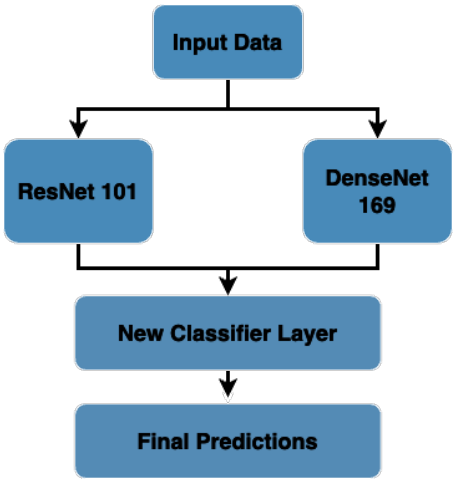

Figure 6. Ensemble model architecture

3.5 Data preprocessing and Implementation

The data preprocessing and the implementation of our networks were done by using the Python programming language. More specifically, we used the PyTorch library, which is a machine learning framework, providing a flexible and efficient platform for building and training neural networks.

After creating our own dataset, we load them using the Imagefolder and Dataloader PyTorch functions, which automatically resize all the images to 224x224 size, turns them into tensors, normalize them and transform the class names into one hot labels automatically. We also used the Sklearn library to divide our dataset into training and test splits. We make sure to use stratified splitting, having a balanced division of all classes into the train and test datasets. We perform a 80-20% split.

For each of the aforementioned models we removed their output layer and replaced it with a new fully connected layer that has as many outputs as the number of classes of our dataset. The previous layers are already good feature extractors. Nevertheless, we decided to unfreeze them and train the whole network. The output layer was trained from scratch with random weight initialization, while the previous layers of the models were finetuned using the pre-trained weights for initialization. For each input of the model, we use the predicted output with the biggest probability for training in the loss function.

For the training part, we used Adam [16] as our optimizer. The Adam optimizer (Adaptive Moment Estimation) is an extension of the stochastic gradient descent (SGD) [17] optimization algorithm commonly used for training neural networks. Adam combines the advantages of two other popular optimization algorithms AdaGrad and RMSProp. It adapts the learning rates for each parameter individually, taking into consideration the momentum, which is the moving average of the past gradients. The momentum is calculated as:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1}$$

where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ are the bias-corrected first and second moment estimates, respectively.

We set the learning rate to 0.003 initially for 3-5 epochs, in order to achieve fast convergence, and then reduced the learning rate to 0.001 to perform some finetuning. We used a batch size of 16. The loss function we used is the CrossEntropyLoss, which is the standard for multi class classification problems. The training epoch number was different for each model, ranging between [7,15]. It is also important to mention that all the training was performed using a NVIDIA RTX 3070 Ti GPU and a NVIDIA RTX 3060 GPU, significantly speeding the learning process and all the involved computations.

4. Experimental Results

The training loss and accuracy plots of the models are shown in Figures 7, 8, 9 and 10 below.

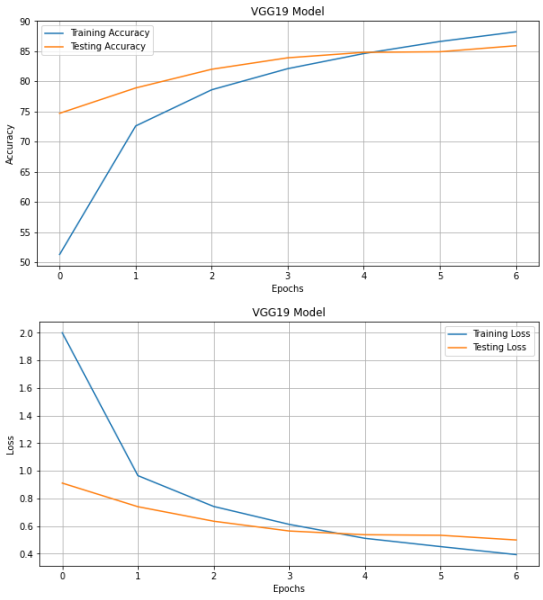

Figure 7. VGG19 plot

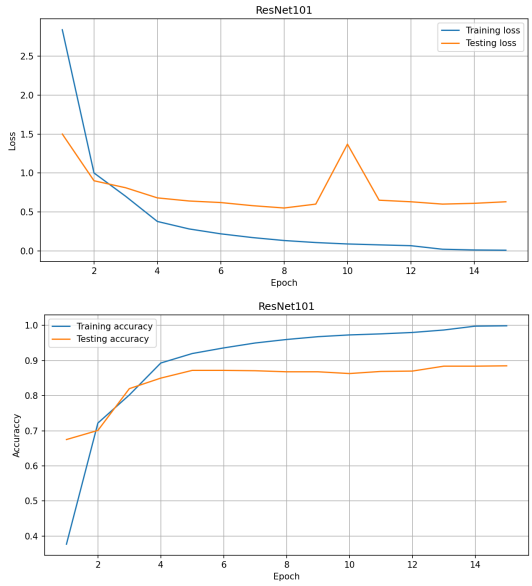

Figure 8. ResNet101 plot

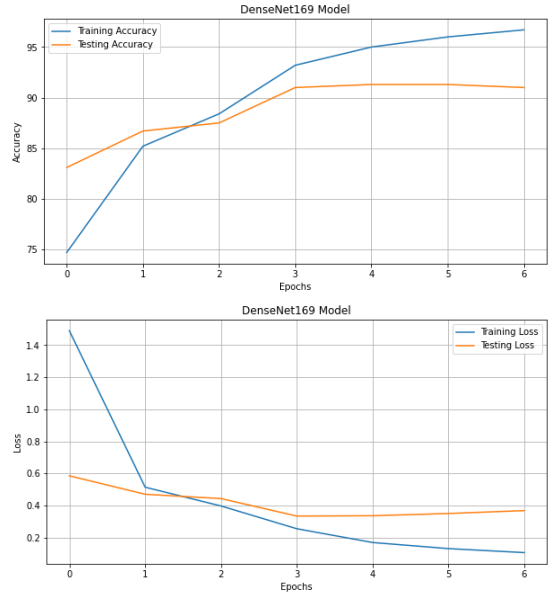

Figure 9. DenseNet169 plot

While there is a wide variety of bird species, many species share a lot of common traits and features, often making it hard to distinguish between close family classes. For this reason, we have decided instead of taking as a prediction only the highest probability of the output vector of our models to also take the top 5 probabilities. We noticed this approach in bird identification apps that already exist. As a performance metric to evaluate our models we chose the accuracy, which is the percentage of correct guesses. The results of our models are presented in Table 1 and depicted visually in 11.

From Table 1 and the model plots, we can easily deduct that all models fit almost perfectly on the training dataset. DenseNet169 achieves both the best top-1 and top-5 accuracy, with ResNet101 and the ensemble of them two being quite close(1-2% deviation).

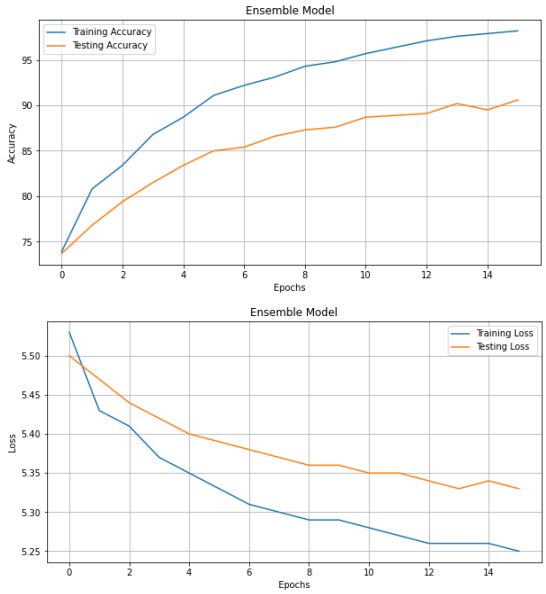

Figure 10. Ensemble Model plot

Table 1. Results on Swedish Birds dataset (500K)

Model	Training		Testing	
	Loss	Acc. (%)	Top-1 Acc. (%)	Top-5 Acc. (%)
VGG19	0.393	88.2	85.9	96.9
ResNet101	0.007	99.9	88.5	97.2
DenseNet169	0.106	96.7	**91.1**	**98.2**
Ensemble	5.25	98.38	90.6	96.7

VGG19 achieved a training accuracy of 88.2 % and a testing top-1 accuracy of 85.9 %, resulting in a generalization gap of 2.3 %. ResNet101 had a training accuracy of 99.9 % and a testing top-1 accuracy of 88.5 %, with a generalization gap of 11.4 %. DenseNet169 obtained a training accuracy of 96.7 % and a testing top-1 accuracy of 91.1 %, leading to a generalization gap of 5.6 %. The Ensemble model reached a training accuracy of 98.38 % and a testing top-1 accuracy of 90.6 %, resulting in a generalization gap of 7.78 %.

1. **Training Loss** - ResNet101 has the lowest training loss, indicating it fits the training data very well. However, this can lead to overfitting.

2. **Training Accuracy** - ResNet101 also has the highest training accuracy, but the large generalization gap suggests it might be overfitting.

3. **Testing Accuracy** - DenseNet169 achieves the highest Top-1 and Top-5 testing accuracy, indicating it performs best on unseen data.

4. **Generalization Gap** - VGG19 has the smallest generalization gap, suggesting it generalizes well despite not having the highest testing accuracy.

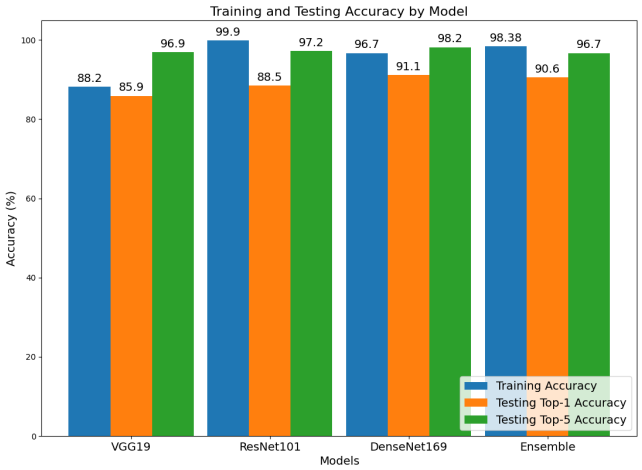

Figure 11. Results depicting the model accuracy on training and test data

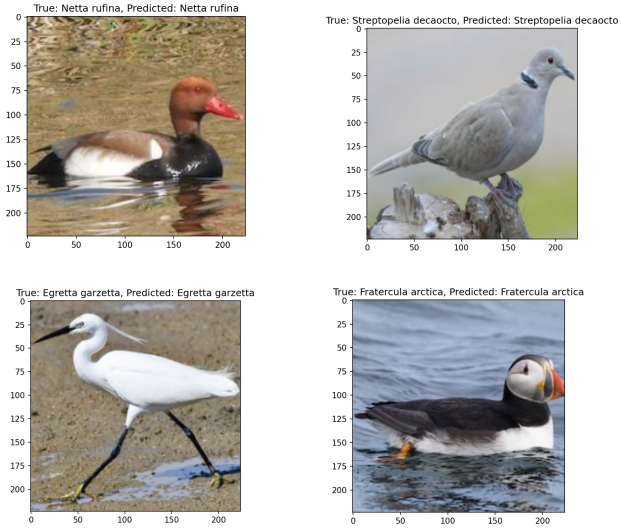

Figure 12. Correctly classified examples

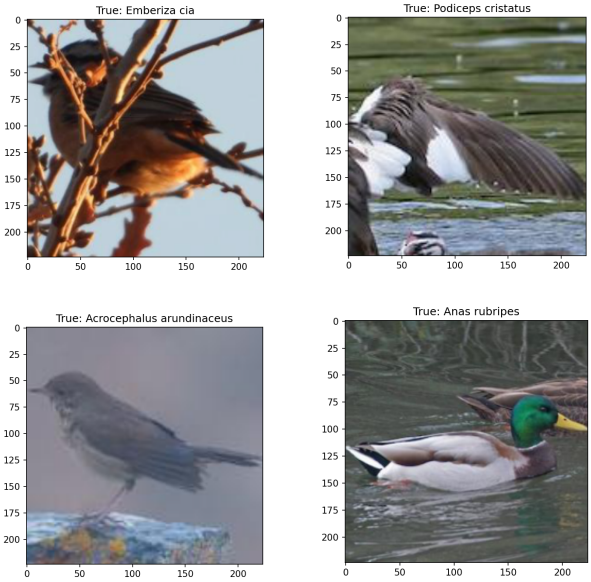

Figure 13. Incorrectly classified examples

Based on the testing accuracy and the generalization gap, DenseNet169 appears to be the best model. It has the highest testing accuracy, which is the most important metric for evaluating model performance on unseen data. Although its generalization gap is not the smallest, it balances well between fitting the training data and generalizing to new data.

Below we can see some prediction examples from the testing dataset using DenseNet169 in Figure 12 and 13. It is worth noting that for the misclassified images, the top-5 accuracy was used. Our model had a total of 1.8% of top-5 misclassifications. From the images we can understand that a high portion of that error is due to bad quality images, most likely as a result of an unsuccessful cropping or if the bird is too far away in the image and the final product has low quality. Also, some birds may be partially hidden by branches or other objects as shown in the top left plot in Figure 13, making it difficult for the model to identify key features. However, we can still notice that some clear high quality images can be misclassified. It's probably because significant variation within the same species can make it challenging for the model to learn consistent features or some bird species look very similar to one another.

5. Conclusion and Future Research

This paper has achieved satisfying and impressive results. Our experimental findings confirmed the effectiveness of our approach. In particular, DenseNet169 outperformed all other models. Our success stems from the high quality of our data and the robust models we employed. However, there is potential for further improvement. For instance, more sophisticated augmentation techniques could be implemented to create a more diverse training dataset which includes variations in rotation, color adjustments and adding synthetic noise. Also, we could develop techniques to make the model more robust to occlusions and complex backgrounds and experiment with more advanced neural network architectures, such as Vision Transformers(ViT) [18], which has shown promise in improving image classification tasks. Of course, we could use more powerful and recent GPUs, such as the NVIDIA A100 or V100, to improve the performance of our models. Also, we could deploy these models for real-time bird detection onto hardwares.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation applied to handwritten zip code recognition". *Neural Computation* **1**:4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". *Commun. ACM* **60**:6 (2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[3] K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[5] K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] M. Tan and Q. V. Le. *Efficientnet: rethinking model scaling for convolutional neural networks*. 2020. arXiv: 1905.11946 [cs.LG].

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. *Densely connected convolutional networks*. 2018. arXiv: 1608.06993 [cs.CV].

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: a large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[9] *Kaggle datasets*. https://www.kaggle.com/. Last Accessed: 2024-03-23.

[10] *Ebird: an online database of bird observations*. https://ebird.org/. Last Accessed: 2024-03-23.

[11] G. Jocher, A. Chaurasia, and J. Qiu. *Ultralytics yolov8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[12] Y. Zheng, C. Yang, and A. Merkulov. "Breast cancer screening using convolutional neural network and follow-up digital mammography". In: 2018, p. 4. DOI: 10.1117/12.2304564.

[13] J. Park, S. Wagner-Carena, S. Birrer, P. Marshall, J. Lin, and A. Roodman. *Large-scale gravitational lens modeling with bayesian neural networks for accurate and precise inference of the hubble constant*. 2020.

[14] B. Patil, M. Rani, S. Anakal, and A. Bhadrashetty. "Early detection of dementia using deep learning and image processing". *International Journal of Engineering and Manufacturing* **13** (2023), pp. 14–22. DOI: 10.5815/ijem.2023.01.02.

[15] A. Mohammed and R. Kora. "A comprehensive review on ensemble deep learning: opportunities and challenges". *Journal of King Saud University - Computer and Information Sciences* **35**:2 (2023), pp. 757–774. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2023.01.014. URL: https://www.sciencedirect.com/science/article/pii/S1319157823000228.

[16] D. P. Kingma and J. Ba. *Adam: a method for stochastic optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[17] S. Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An image is worth 16x16 words: transformers for image recognition at scale*. 2021. arXiv: 2010.11929 [cs.CV].