ML and AI Project Report

Group 3: Gustaf Cramer, Noam Etten, Keshav Ganesh
Caspar Lehmkühler, Luiza Zinca

May 29, 2024

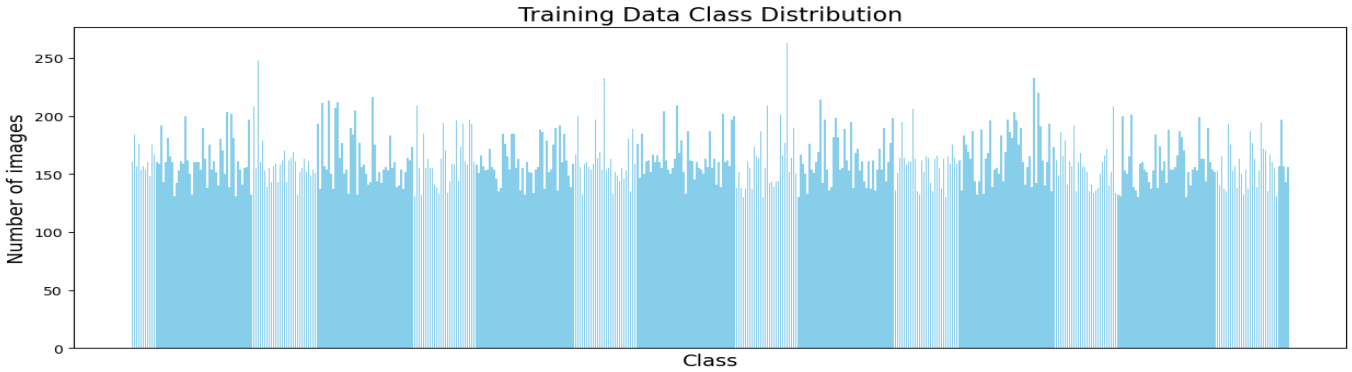
1 Introduction

In this project, we focus on the application of neural network architectures to the challenge of bird species recognition, a task that involves significant variability and complexity due to the diversity of features and environments. Utilizing a comprehensive dataset we aim to compare the performance of convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), to identify each species accurately. We expect to be able to achieve results in the high 90%'s since the dataset is adequately preprocessed for this task.

Our approach involves data preprocessing, augmentation to enhance model robustness, and hyperparameter optimization to fine-tune our models for optimal performance. By leveraging both traditional CNN architectures and the newer ViT models, we explore the strengths and limitations of each in handling high-dimensional, unstructured data. This report details our methodology, experiments, and findings, providing insights into the practical application of deep learning techniques in real-world image classification tasks.

2 Dataset

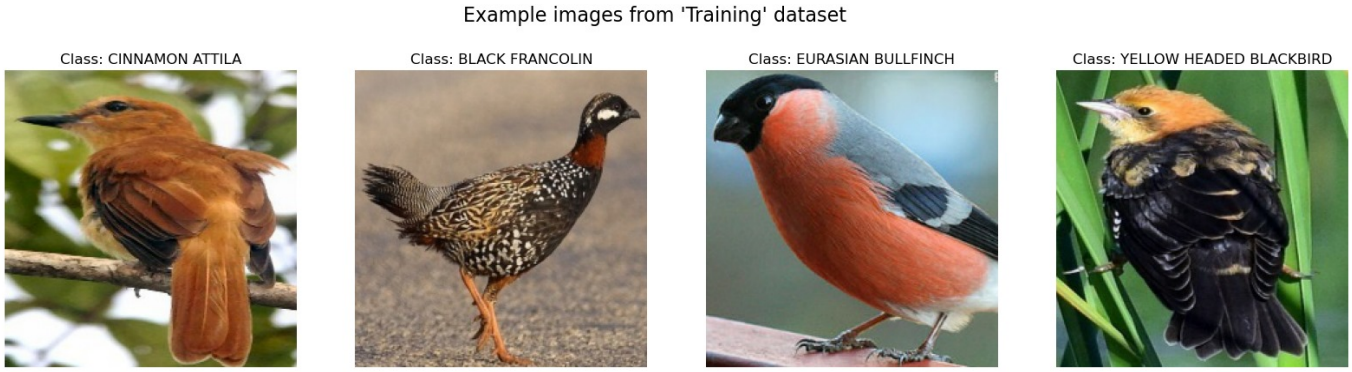
The dataset we are using was provided by Kaggle. It includes 84635 training images, 2625 test images(5 images per species) and 2625 validation images(5 images per species).



A short comment on the above figure: Although the class distribution appears to be relatively balanced, minor class imbalances can still cause the model to perform sub-optimally on underrepresented classes. This is because some classes are inherently more difficult to learn, requiring more examples to achieve

comparable performance. However, there is a simple fix for this, in the form of data augmentation, which enhances the robustness of the training process. This still leaves in some class imbalances, however, it leads to more data points in the minority classes.

The provided data has very high quality and contains one bird per image which takes up approximately 50% of the space. All entries in the dataset have dimensions 224x224x3 (3 for RGB colours).
Furthermore, the dataset includes a Metadata file labelled (Birds.csv) with 5 columns for file path, species label, scientific label, dataset and class ID.
The images were sourced from the web, analysed for quality and duplicates and then cropped to fit the requirements. Notably, this results in the data not being balanced: There are varying quantities of training images per class.

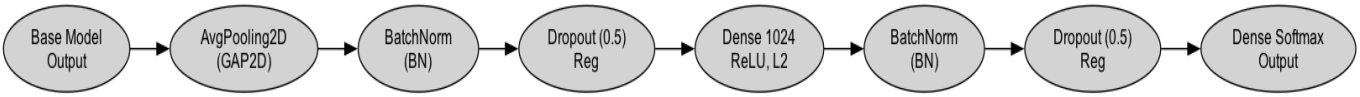


Firstly, we must note that the test and validation images were selected as the images with the highest quality. This means our models will perform unusually well on these sets. Secondly, male birds are overrepresented in the dataset, which might influence the prediction when attempted on female birds. A possible reason to use male birds for model training is that male birds are generally more colourful than female birds, making them easier to classify as their features are more pronounced. But clearly, this also introduces a bias since it makes it harder to predict female species members.

3 Models

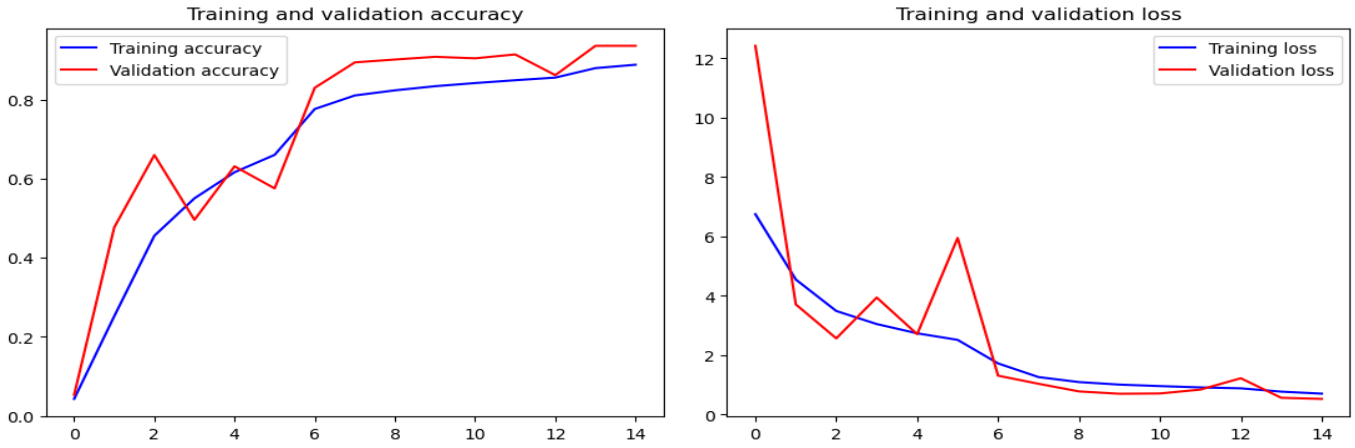
3.1 Convolutional Neural Networks

The first model we trained was a traditional CNN ResNet 50 model. We used this because it eliminates the vanishing gradient problem by utilising skip connections, allowing us to train very deep neural networks. Initially, we implemented some basic domain-specific data augmentation to the training set by scaling, rotating, shifting, zooming and flipping the images. After batching the data, we were ready to run our first models. Notably, we used a constant batch size (32) when training the models. In addition, to the standard ResNet model, we added custom NN layers on top:



Overall we ended up with approximately 26 million parameters. Initially, we set all of them to be trainable

since we did not know what results we would get and wanted to find a general benchmark. To improve the training process, we made use of callbacks, namely early stopping, a learning rate scheduler ("ReduceLROnPlateau") and a model checkpoint callback to save the best model. Since this is our initial model, we only trained the model for 15 epochs. We achieved a training and validation accuracy of 88% and 94%, respectively. Immediately we can see that even this initial model yields very good results.



The fact, that our performance on the validation set is higher, stems from the previously mentioned selection of validation images. After running the final model on the test set we achieved 96% accuracy and a loss of 0.46.

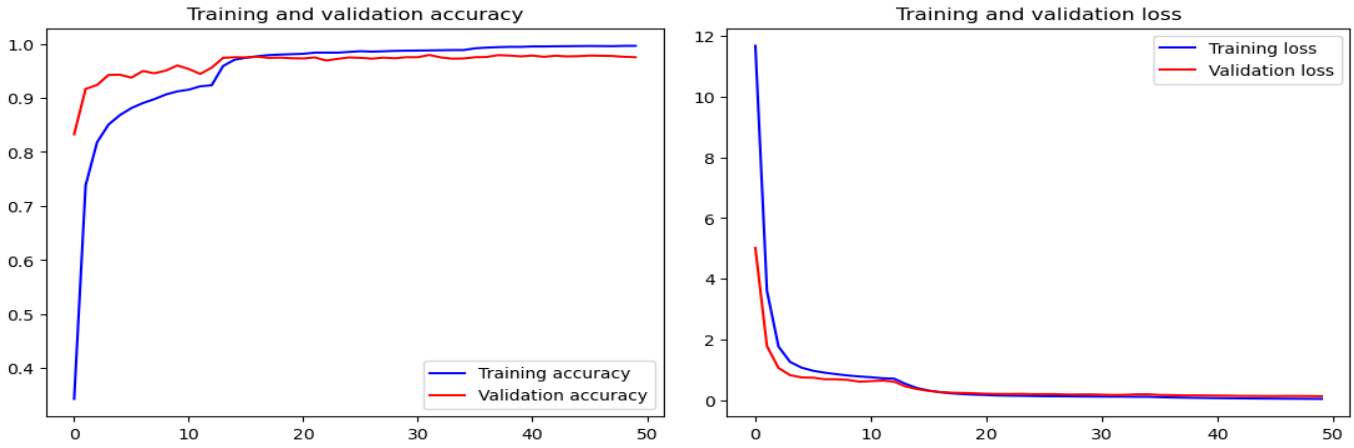
3.1.1 Hyperparameter Tuning

Although our initial results were strong, we were convinced that even some simple hyperparameter tuning would significantly improve results. The three parameters we were focusing on were "trainable layers", "dropout rate" and "initial learning rate". In order to reduce computational complexity we ran our parameter search for only 5 epochs using 25% of the training data. The goal of hyperparameter tuning was not to maximise accuracy but rather to find the optimal set of hyperparameters to train the model further.

The tuning method we used was Bayesian Optimisation. It works by building a probabilistic model of the function's behaviour and using it to make informed decisions about where to sample next, focusing on areas likely to offer improvement over current observations. The tuning method was run for 20 trials.
Our optimal parameter set was:

```
{'trainable_layers': 160, 'dropout_rate': 0.5, 'learning_rate': 0.0001}
```

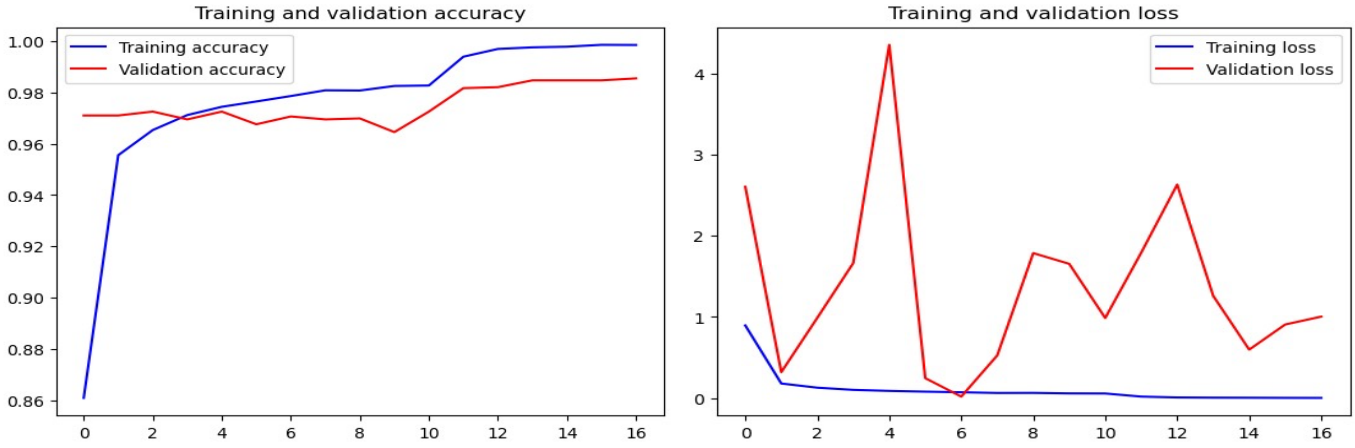
After performing the same data augmentation as above, we were ready to run the model again with these optimal parameters. This time we ran the model for 50 epochs and achieved even better results. Our training and validation accuracies were 99.63% and 97.52% respectively.



Our final test accuracy was at 98.90% with a loss of 0.12 (Only 29 images were misclassified).

3.2 Vision Transformers

Even though we already achieved amazing results with the CNN we wanted to also run a transformer model on the dataset. These were easier to implement with pytorch which is why we deviated from the tensorflow approach above.
As our base model, we used a Google Vision Transformer ("ViT Base Patch 16"). This model processes images by dividing them into fixed-size patches (16x16 pixels each), which are then linearly embedded and passed through a sequence of Transformer blocks. We one-hot encoded the labels, performed some simple data augmentation, and then ran the model. Since we expected the accuracy to plateau after some epochs, we implemented early stopping by counting how many consecutive epochs had a decreasing (or not increasing) validation loss. We ended up with the final model after 17 out of a total of 50 epochs. Our performance slightly increased compared to the CNN, with a final training and validation accuracy of 99.85% and 98.55%.



Here we can see some interesting results. While our accuracy is extremely high, the validation loss fluctuates a lot (it goes all the way down to 0.02 but also up to over 4). We are not exactly sure why this

happens. There can however be several reasons for this. Firstly, it is possible, that while our overall prediction on the validation set is correct, the model is not confident (it could for example be unsure between two species). This would lead to a high cross-entropy loss. These predictions on the validation set could fluctuate vastly between epochs and lead to different validation losses. Another explanation could be, that the data in the validation set is distributed slightly differently compared to the training set since we pre-selected the images with some quality criteria. Vision Transformers can be highly sensitive to such differences and produce varying losses. Lastly, we could negate these fluctuations by running it for more epochs or tuning the parameters more granularly. However, overall we achieve a very good performance. On the test set, our model predicted the labels with a high accuracy of 99.09% (In total, only 24 images were misclassified). Due to these amazingly accurate results and because it would require very extensive computation, we chose not to perform hyperparameter tuning on this model.

3.3 Missclassifications

For both models, the species of misclassified images were barely visually distinguishable from the correct label. In the graphics below we can see the actual class next to a random picture from the bird's predicted class.

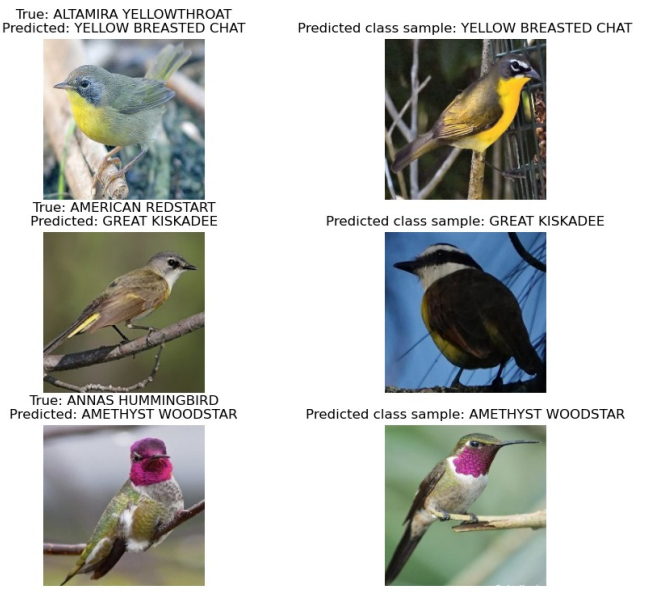


Figure 1: Predicted vs. True Class for CNN Model

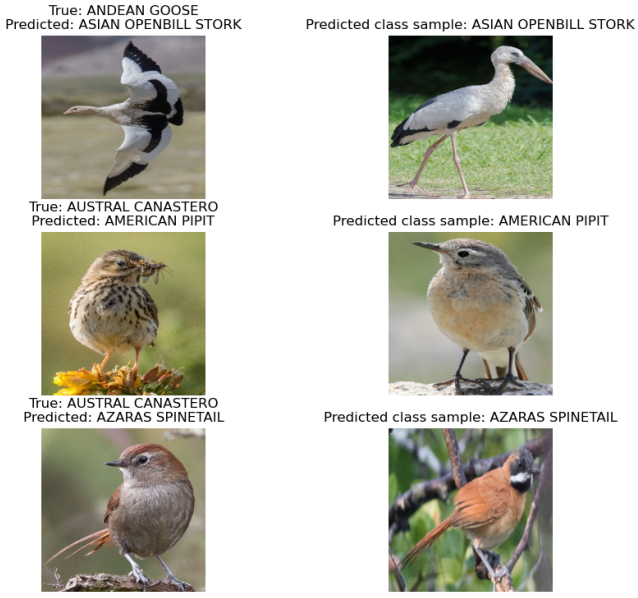


Figure 2: Predicted vs. True Class for ViT Model

3.4 Ensemble

After inspecting the misclassified images of both models (by checking the indices of the images), there were a lot of images which did not overlap, i.e., there were misclassifications for the ViT Model that were classified correctly using the CNN model, and vice versa. Therefore, an ensemble method should be able

to reassign all of these classifications to the right label. To implement this, we took the mean of the two probability vectors which were output by each classifier. This strategy worked well as the test accuracy improved to 99.35% (In total only 17 images were misclassified).

Obviously, this result is to be taken with a caveat. As previously discussed, the test set consisted of the highest quality, best images in the dataset, meaning that, in order to test our models even more rigorously, we should create a new train-test split and evaluate the model again.

4 Conclusion

Overall, we successfully applied Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) to classify 525 distinct bird species based on image data. Our experiments demonstrated that both models could achieve high accuracy, with the Vision Transformer marginally outperforming the CNN in both the training and validation phases. The ensemble approach, leveraging predictions from both models, further enhanced our results, achieving 99.35% accuracy on the test set.

These findings demonstrate the potential of advanced neural network architectures in handling complex image classification tasks, particularly in areas which require differentiation among highly similar categories. The performance of the ensemble model also highlights the value of combining different machine-learning strategies to improve prediction accuracy.

However, the performance on the pre-selected test set also poses questions about model generalizability. In future work, we could aim to evaluate these models on a more diverse and challenging set of images to more rigorously assess their practical utility in real-world scenarios.

Overall, this project not only advances our understanding of applying machine learning techniques to biological classification but also sets the stage for future research into more robust, adaptable, and accurate image recognition systems in ecological and conservation-related applications.

A Bibliography

- **Kaggle Dataset:** https://www.kaggle.com/datasets/gpiosenka/100-bird-species
- **ResNet50:** https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py
- **ViT Base Patch 16:** https://huggingface.co/google/vit-base-patch16-224
- **GitHub Repository:** https://github.com/Casparlehmkuehler/BirdSpecies