CAI 4104/6108: Machine Learning Engineering

Project Report: Bird Species Classification

Sakshi Pandey
(Point of Contact)
sakshi.pandey@ufl.edu

Satyajit Mohanty
satyajit.mohanty@ufl.edu

Avaneesh Khandekar
akhandekar@ufl.edu

Hamsini Sivalenka
hamsinisivalenka@ufl.edu

Jason Ang
jasonang@ufl.edu

December 1, 2024

1 Introduction

The accurate classification of bird species from images is not only a fascinating challenge in the field of computer vision but also a critical endeavor for biodiversity conservation, ecological research, and the enhancement of birdwatching experiences. With over 10,000 known bird species globally, each exhibiting unique morphological characteristics, the task demands sophisticated computational approaches capable of handling such diversity in features [5].

Our project capitalizes on the rich repository of bird images available on Kaggle, utilizing a dataset that spans 525 distinct bird species. This variety presents a unique opportunity to leverage advanced machine learning techniques, particularly Convolutional Neural Networks (CNNs), which are renowned for their effectiveness in image recognition tasks. By employing CNNs, we aim to develop a model that not only recognizes but also accurately differentiates between the subtle and overt distinctions across bird species.

To ensure the robustness and generalizability of our model, we incorporate K-fold cross-validation methods in our training process. This technique helps mitigate overfitting and ensures that our model performs consistently well across different subsets of data. Furthermore, we enhance our model's performance through transfer learning, utilizing pre-trained networks to capitalize on previously learned features from extensive and diverse image datasets. This approach is particularly beneficial in scenarios where data for certain classes is limited or imbalanced.

The significance of this project extends beyond the academic and technical realms. Accurately identifying bird species is crucial for monitoring biodiversity and ecosystem health. It aids conservationists in tracking species distribution and population changes, which can inform conservation strategies and policy decisions. For ecologists, understanding species interaction and habitat utilization is essential for ecological assessments and environmental impact studies. Additionally, for the growing community of birdwatching enthusiasts and citizen scientists, improving automated bird identification can greatly enhance their observational capabilities, contributing to larger-scale biodiversity monitoring projects through community science initiatives [4], [6].

2 Approach: Dataset(s) & Pipeline(s)

2.1 Dataset

The dataset consists of 84,635 images spanning 525 distinct bird species, making it a large and diverse collection crucial for training robust models. Each species contributes a minimum of 130 images, although the distribution is not uniform, introducing a class imbalance challenge. The dataset is meticulously partitioned into training, validation, and test sets with 2,625 images each for validation and testing.

2.2 Data Handling

The machine learning pipeline includes:

Data Inspection and Cleaning: Initial steps include inspecting the dataset for missing or corrupted images and inconsistencies in labeling. Each image's corresponding label (species) and, where applicable, the scientific name, are verified for consistency and completeness.

Splitting the Data: The dataset is divided into training, validation, and test sets. This split ensures that the model can be trained on a large portion of the data while also being independently validated and tested on unseen images to assess generalization capabilities.

Image Preprocessing: All images are resized to 224x224 pixels, a standard dimension for CNN inputs. Images are also normalized to have pixel values between 0 and 1, enhancing model training efficiency and stability.

2.3 Model Architecture - CNN

The CNN designed for this project follows a classic architecture pattern suitable for image classification tasks. Here's a breakdown of each layer and its function:

Input Layer: The model accepts images resized to 224x224 pixels with 3 color channels (RGB).

Convolutional and Pooling Layers: The model includes several convolutional layers, each followed by batch normalization and max pooling layers to extract and refine features from the input images.

First Convolutional Layer Conv2D: 32 filters, 3x3 kernel size, ReLU activation function, and padding set to 'same' to keep the output size the same as the input size.

BatchNormalization: Normalizes the activations from the previous layer, which helps in speeding up the training and improving model stability.

MaxPooling2D: 2x2 pool size, reduces the spatial dimensions (height and width) by half, minimizing the computational complexity and overfitting risk.

Second Convolutional Layer Conv2D: 64 filters, increases the depth to capture more complex features. Similar batch normalization and max pooling as the first layer.

Third Convolutional Layer Conv2D: 128 filters, further increases the depth for complex feature extraction. Followed by batch normalization and max pooling.

Flattening Layers: Converts the 3D output of the last pooling layer into a 1D vector. This is necessary to transition from feature extraction layers (convolutional and pooling) to classification layers (dense).

Dense Layers: Following the flattening layer, the model includes several dense layers that serve as the classification portion of the network:

Dense: 512 units, ReLU activation function. This fully connected layer uses the features extracted by the conv layers to determine the final output.

Dropout: Set to 0.5, it randomly sets input units to 0 at each update during training time, which helps to prevent overfitting.

Optimiser: Adam, known for its efficiency and adaptive learning rate capabilities. Loss Function: Categorical crossentropy, suitable for multi-class classification problems. Metrics: Accuracy, to monitor the training and validation performance.

2.4 Model Architecture - CNN With Transfer Learning

The model architecture utilizes a blend of transfer learning and custom layers:

Transfer Learning with EfficientNetB0[3]: The EfficientNetB0 architecture, pre-trained on the ImageNet dataset, serves as the base model [3]. Its initial layers capture universal image features like edges and textures, which are relevant across various image recognition tasks, including bird classification.

Custom Top Layers: After the pre-trained layers, custom layers are added to tailor the model to the specific task of bird species classification. This includes:

Global Average Pooling 2D: Reduces the spatial dimensions of the feature map to a single vector per map, minimizing the number of parameters and helping in reducing overfitting.

Dense Layers: Several fully connected layers follow, including a large dense layer with 2048 units and ReLU activation for non-linear transformation, and a dropout layer to prevent overfitting.

Output Layer: The final layer is a dense layer with 525 units (one for each bird species) and softmax activation to output a probability distribution over the classes.

2.5 Training Strategy

The training process is designed to optimize accuracy while preventing overfitting:

Optimization and Loss: The Adam optimizer is used for its efficient computation and adaptive learning rate capabilities, coupled with categorical cross-entropy loss to handle the multi-class classification.

Callbacks: These are the callbacks we used

Early Stopping: Monitors the validation loss and stops training if it does not improve for a set number of epochs, helping in preventing overfitting. We used a patience of 3 meaning that our model will stop training if there is no improvement in the last 3 epochs.

Model Checkpoint: Saves the best model based on validation accuracy, ensuring that the best performing model is retained.

Reduce Learning Rate on Plateau: Reduces the learning rate when the validation loss plateaus, allowing for finer adjustments in weights that could lead to better model performance. We used a LR factor of 0.2, which means it decreases it to 20% of its current value with a patience of 2 epochs and LR lower bound of 0.001.

K-fold Cross-Validation: Implemented to validate the robustness of the model. The dataset is split into K subsets, with the model being trained on K-1 subsets and validated on the remaining subset. This process is repeated K times with each subset used exactly once as the validation data. This method provides a thorough indication of how well the model is likely to perform on unseen data.

3 Evaluation Methodology

The project utilizes these metrics to evaluate the performance of the classification model:

Accuracy: Measures the proportion of correctly predicted images out of the total number of predictions. It's a straightforward indicator of overall performance but may not always reflect the true predictive power of the model, especially in datasets with imbalanced classes.

Macro F1 Score: This metric calculates the F1 score (the harmonic mean of precision and recall) for each class and then takes the average. It is particularly useful in datasets with imbalanced classes because it treats each class equally, regardless of its frequency.

AUC-ROC (Area Under the Receiver Operating Characteristic Curve): Measures the ability of the classifier to distinguish between classes. An AUC of 1 represents a perfect model, while an AUC of 0.5 represents a worthless model. For multi-class classification, the AUC-ROC is typically calculated for each class against all other classes and then averaged.

3.1 Model Evaluation Techniques

The evaluation of the model employs a mix of techniques to ensure comprehensive testing:

Training-Validation Split: During model training, data is split into a training set and a validation set. The training set is used to fit the model, while the validation set is used to monitor the model's performance on unseen data, preventing overfitting and tuning hyperparameters.

K-fold Cross-Validation: This method enhances the reliability of the model evaluation by reducing the variance associated with a single train-test split. The dataset is divided into 'K' subsets, and the model training and validation are repeated 'K' times. Each time, one of the 'K' subsets is used as the validation set, and the remaining 'K-1' subsets are used as training data. This process helps in assessing the model's stability and performance across different data subsets.

Test Set Evaluation: After the model has been trained and validated, it is finally evaluated on a separate test set that was not used during the training or validation phases. This step is crucial to gauge the model's real-world applicability and to ensure that the model generalizes well to new, unseen data.

Performance Visualization: Graphical representations such as ROC curves for each class, and loss-accuracy plots over epochs are used to visualize the model's performance across different metrics. These visualizations help in identifying classes that may require additional tuning or more data and in assessing the learning process over time.

3.2 Tools and Libraries

To implement and track these evaluations, various tools and libraries are used, including:

Scikit-learn for model evaluation metrics and cross-validation.

TensorFlow and Keras for building and training the models, including support for callbacks like EarlyStopping and ModelCheckpoint.

Matplotlib and Seaborn for generating plots that illustrate the model's training dynamics and performance.

4 Results

We first visualised 20 samples from the bird classification dataset, shown in Figure 1

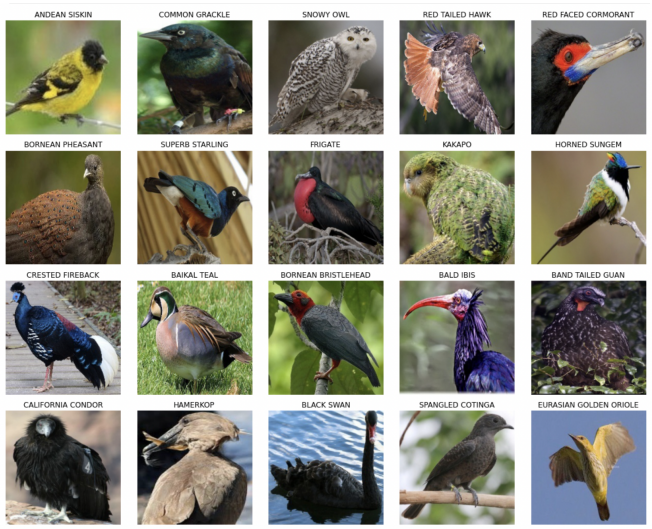

Figure 1: 20 Samples from dataset

We then proceed to check how the dataset is distributed amongst the top 20 labels out of the 525 labels, as shown in Figure 2

Figure 2: Distribution of top 20 labels from the dataset

Our experimental results showcased the effectiveness of CNN models with transfer learning techniques. Here are some detailed outcomes: We first implemented a CNN model without transfer learning and attempted to fine tune it, however this model fell short of the baseline previously observed with this dataset. The dataset baseline

on kaggle was 87% and our model achieved an accuracy of 77.5%. The model loss and accuracy of the CNN model is shown in Figure 3

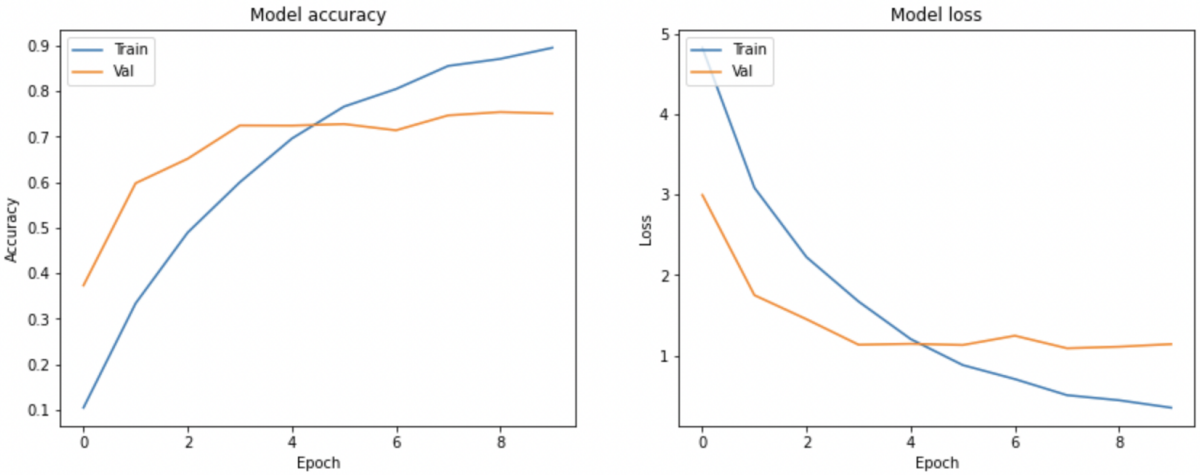

Figure 3: CNN Model loss and accuracy over multiple epochs

This led us to pivot to implementing transfer learning as suggested by Dr. Bindshaedler. Transfer learning significantly outperforms a basic CNN baseline in all key metrics in literature we observed, particularly with image datasets and multi-class classification.

4.1 Accuracy

The average accuracy achieved by the model over multiple folds of cross-validation provides a straightforward assessment of the model's overall performance. The K-fold cross-validation was used to mitigate any bias in the model evaluation due to random splits of the data.

Baseline Model Accuracy: The baseline model accuracy that kaggle dataset provided was 87%

Achieved Model Accuracy: The final model, enhanced with transfer learning (EfficientNetB0) and additional tuning, achieved an average accuracy of approximately 92.8% across the test sets, as shown in Table 1, indicating a robust performance against varied and unseen data.

4.2 Macro F1 Score

The macro F1 score is critical for datasets with imbalanced classes, as it gives equal weight to each class, ensuring that the model's performance reflects its ability to identify less frequent classes effectively. Formula:

$$\text{Macro F1} = \frac{1}{N}\sum_{i=1}^{N} F1_i \tag{1}$$

The model reached a macro F1 score of approximately 0.85, as shown in Table 2 demonstrating its competent handling of class imbalance and its ability to maintain precision and recall across classes.

4.3 AUC-ROC

The AUC-ROC score evaluates the model's ability to distinguish between classes, with a higher score indicating better performance. With transfer learning our model reached an AUC-ROC of 0.99994, as shown in Tanle 3. These metrics were captured over multiple training epochs, with improvements observed as the model fine-tuned its parameters on the bird species images. Overall our model outperformed the baseline previously observed for this dataset on Kaggle and showed a massive improvement in accuracy with transfer learning.

Table 1: Accuracy

Model	Accuracy
CNN	77.5%
CNN with k fold	73.7%
CNN with transfer learning	92.8%

Table 2: F1 score

Model	F1-score
CNN	0.77%
CNN with transfer learning	0.927%

5 Conclusions

The project's empirical findings demonstrate a clear superiority of the transfer learning model over standard CNN models across all performance metrics. This superiority underscores the inherent advantages of utilizing pre-trained networks, which have already learned general features from a diverse array of images. Such networks are particularly adept at adapting to specific tasks like bird species classification, where nuanced visual differences are pivotal ([3]).

However, the implementation of K-fold cross-validation revealed a slight decrease in accuracy for the CNN model, suggesting variability in model performance across different data subsets. This variability may indicate overfitting in models trained without cross-validation or a lack of generalization in the model's parameters across various segments of data. These findings highlight some intrinsic limitations of K-fold cross-validation, particularly its potential for data leakage and the challenges of applying it in scenarios with non-IID data ([2], [1]).

Despite these challenges, the transfer learning model exhibited robust performance, achieving high scores in both F1 and ROC AUC metrics. These results not only confirm the model's accuracy but also its ability to effectively handle class imbalance while maintaining high levels of precision and recall—qualities essential for reliable practical applications.

Looking forward, the success of the transfer learning approach suggests valuable pathways for future research. Exploring more complex architectures or newer pre-trained models could further enhance model performance. Additionally, advanced data augmentation techniques, ensemble methods, and the application of interpretative mechanisms like Grad-CAM heatmaps could provide deeper insights into optimizing CNN networks specifically for bird classification tasks. In Figure 4 we see which parts of the image the network is utilizing to classify bird species.

Table 3: AUC ROC score

Model	ROC AUC score
CNN	0.9979%
CNN with transfer learning	0.9999%

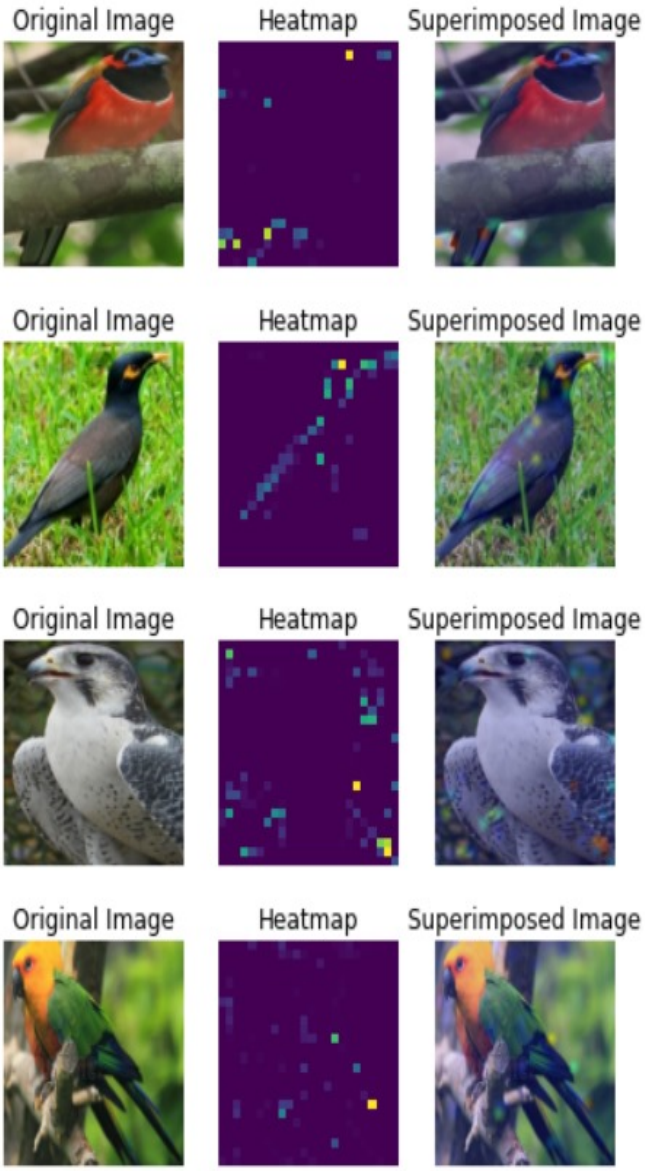

Figure 4: Grad-CAM

In conclusion, the findings robustly indicate that for complex image recognition tasks such as bird species classification, transfer learning provides significant advantages over models trained from scratch, particularly in terms of precision and robustness. This advantage is crucial for applications requiring high reliability and accuracy in environmental and conservation-related contexts.

References

[1] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 2:1137–1145, 1995.

[2] D. R. Roberts, V. Bahn, S. Ciuti, M. S. Boyce, J. Elith, G. Guillera-Arroita, S. Hauenstein, J. J. Lahoz-Monfort, B. Schröder, W. Thuiller, et al. Cross-validation: what does it estimate and how well does it do it? *Ecography*, 42(4):609–620, 2021.

[3] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[4] Peter Welinder et al. Caltech-ucsd birds 200: A dataset of bird images annotated with subspecies information. In *Sensor Networks and Information Processing*, 2015.

[5] Xian Xie et al. Deep learning for image-based bird species classification. *Journal of Machine Learning Research*, 2020.

[6] Shuiwang Zhang et al. Bird species recognition using deep learning: A visual attention-based approach. *Ecology and Evolution*, 2019.