Title: Bird Species Identification using Deep Learning on Spectrograms of Sound

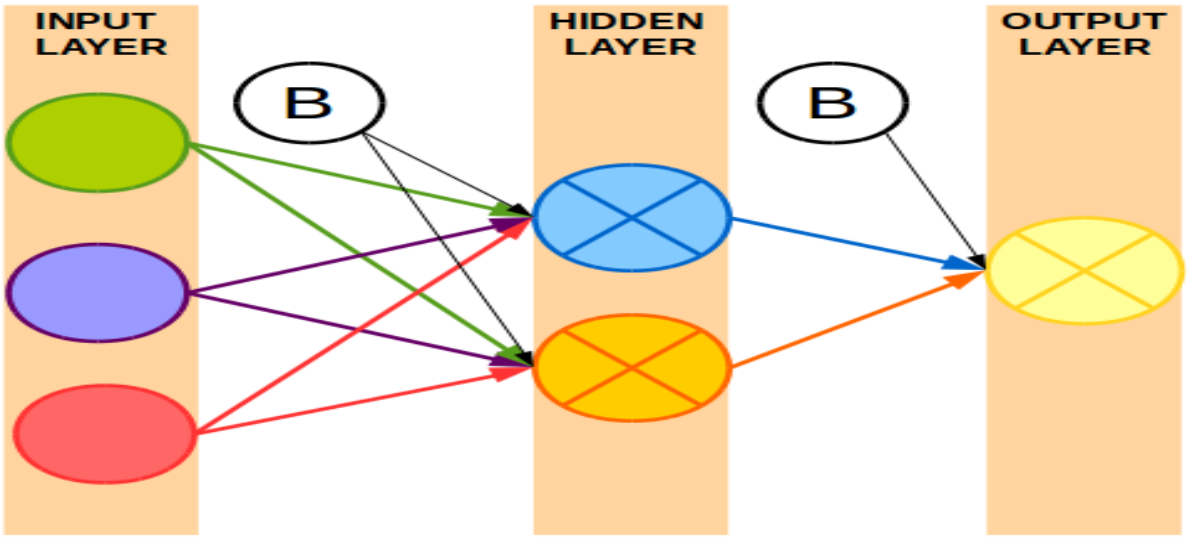
Abstract

This is a project that entails bird species classification into their own species based on their unique vocalizations presented in the form of grayscale spectrogram images. The dataset consists of 12 bird species with varying numbers of audio-based spectrograms created (128, 517). These images convey frequency patterns that are utilized as discriminative features for species identification. A convolutional neural network (CNN) model is developed and trained to recognize these frequency-based representations for successful classification.

Introduction

The general aim here is to train a Convolutional Neural Network (CNN) from scratch to identify bird species based on their sounds. The dataset used has already preprocessed spectrogram images for 12 species, with each spectrogram being of size (128, 517), even though the number of samples for each species differs. This preprocessed data was taken from a public HDF5 file on GitHub, and the raw recordings were taken from the Xeno-Canto Bird Recordings Extended dataset on Kaggle, which had 240 bird species' sounds in total. The data used here are not normalized, and the final objective is also to classify the bird species from a direct MP3 file input. This approach allows us to explore ways in which features of sound can be used economically for species classification based on deep learning.

Theoretical Background



Layers in a CNN

Input Layer:
The input layer is the layer where the network gets the data. For an image, this layer accepts the pixel values (i.e., for a grayscale image, each pixel can have a value between 0 and 255).
Example: For an image of 28x28 pixels grayscale, the input would be a 28x28 matrix with a single channel (28x28x1).

Hidden Layers:
These layers extract features from the input data. In CNNs, the most important types of hidden layers are Convolutional Layers and Pooling Layers. There can be n number of hidden layers

Convolutional Layers:
These layers apply a convolution operation with filters (also known as kernels) to detect features like edges, textures, and patterns in the image. For instance, one filter might look for horizontal edges, and another might look for vertical edges.
The filter slides over the input image and produces a feature map.

Pooling Layers:
Pooling shrinks the spatial dimensions (height and width) of the input data, which helps reduce computational expense and overfitting.

Max pooling is common, where the maximum value in some region is taken, which is helpful for retaining the most prominent features.

Output Layer:
This layer makes the final prediction. In the case of binary classification, there will be a single node (with a sigmoid activation). In the case of multi-class classification, the output layer will have one node for every class (with a softmax activation function).

Nodes, Biases, and Weight Updates
Nodes: Each node in the network is responsible for computing the weighted sum of its inputs and passing that along through an activation function.

Biases: A bias is added to the weighted sum to change the activation function. It allows the model to better fit the data.

Weight Updates: During training, the biases and weights are updated by backpropagation. The network learns as it updates these values to minimize the error (or loss).

Forward Propagation: During forward propagation, the input data is passed through all the layers in the network (convolution, pooling, fully connected) to make a prediction. The model outputs a result based on the weights, biases, and activations of each layer.

Backward Propagation: After completing forward propagation, the model will match its output with the actual output and calculate the loss (error). Based on that error, it will update the weights and biases via gradient descent in such a manner that it decreases the loss

Activation Functions

Sigmoid: Used in binary classification issues. It squeezes the output between 0 and 1, therefore it could be used to predict probabilities between two classes.

Softmax: Used in multi-class classification. It normalizes raw output values into probabilities by ensuring the sum of all probabilities is equal to 1. It provides a probability for each class.

Loss Functions

Binary Cross-Entropy: Used in binary classification problems, this loss function measures the difference between predicted probabilities (from sigmoid) and true labels.

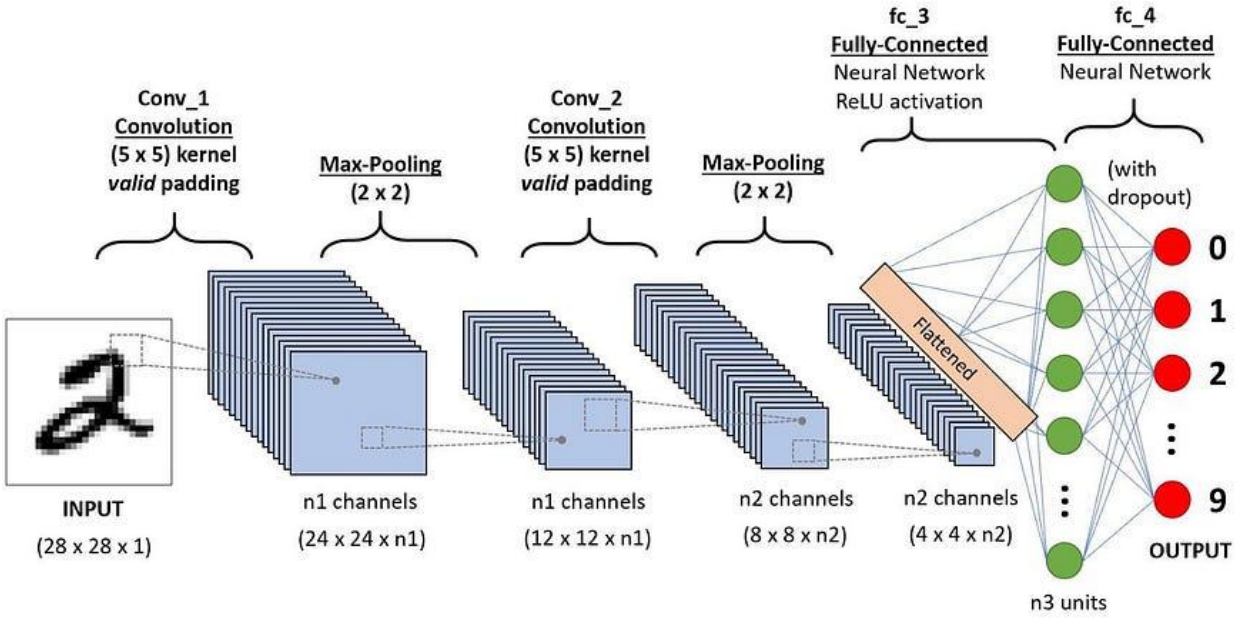
Categorical Cross-Entropy: For multi-class classification issues, this loss function is utilized if softmax is employed to output the class probabilities.

Dropout: A technique to prevent overfitting. During training, random nodes are "dropped out", which causes the model not to rely too much on any one node and improves generalization.

L2 Regularization: This adds a penalty term to the loss function, which penalizes large weights. This avoids overfitting as the model now prefers simpler models.

Learning Rate: Learning rate determines the magnitude of the step that the model takes when it updates the weights and bias during training. Too high a learning rate can cause the model to overshoot the optimal values, while too low a learning rate can cause the model to learn slowly, and it can take an eternity for it to converge. A correct learning rate needs to be used to train a model correctly.

Early Stopping: Early stopping is a method of preventing overfitting.
It halts the training when the performance on
the validation set ceases to increase for some number of epochs, such that the model won't overfit to the training set.



Pooling in CNNs
In max pooling, a window slides across the feature map and selects the maximum of every window. This reduces the spatial data dimensions of the data, making the model computationally lighter.

Methodology:
Data Overview:

- The dataset consists of 12 bird species with frequency information associated with each species.
- Total frequency samples:1886 frequency samples altogether.

Data Preparation and Processing:
The MP3 files containing bird sound are processed in order to derive the Mel spectrograms in the dB scale. Specific time slices of the MP3 files are chosen such that the bird sound is clearly audible. The following time slices have been chosen:

- test1.mp3: 7-12 seconds
- test2.mp3: 1-4 seconds
- test3.mp3: 6-13 seconds

Conversion to HDF5:

- The MP3 files are converted to HDF5 format for easier access and storage. The audio clip data of the spectrograms is extracted from the audio clips, and each clip's data is converted to a Mel spectrogram using the librosa library. The spectrograms are saved into an HDF5 file.
- The information in the HDF5 file is grayscale and hence is normalized by dividing the pixel values by 255. It rescales the values to 0 to 1 range so that the model has a better chance of learning meaningful features during training.

Testing:
In evaluation, the trained model is used to predict which species is closest to the given spectrogram from the MP3 files. The spectrograms are graphed, and the prediction is done using the model. The predicted species is printed along with the confidence level for each species, and a bar chart is given showing the prediction confidence.

Key Training Techniques

1. **Early Stopping:**
   Early stopping is implemented to prevent the model from overfitting. It monitors the validation loss and halts training when the validation performance stops improving for a set number of epochs. This ensures that the model doesn't continue training unnecessarily and helps it generalize better. Details on how early stopping works, are referred from TensorFlow's EarlyStopping documentation.
2. **Reducing Learning Rate on Plateau:**
   ReduceLROnPlateau is used to reduce the learning rate when the validation loss stops improving. This allows the model to make finer adjustments in the later stages of training, improving convergence. The following stackoverflow post gave a better idea Post
3. **Model Checkpoint:**
   Model checkpointing is employed to save the best-performing model based on the validation loss during training. This ensures that the best version of the model is saved and used for evaluation or deployment. Usage of model checkpoint was found in StackOverflow discussion on saving model checkpoints in TensorFlow.

Methodology:

Binary Classification Methodology

In this binary classification task, we are training a Convolutional Neural Network (CNN) to classify two bird species: **'amerob'** and **'rewbla'**. The model uses spectrograms of bird calls converted into Mel scale representations. Below is the overview of the architecture and training methodology.

CNN Architecture

The CNN architecture for binary classification consists of several distinct layers that help the model learn complex patterns from the input spectrograms:

1. **Input Layer**:The model takes input spectrograms of shape (128, 517, 1), representing grayscale images with dimensions 128x517 pixels.
2. **Convolutional Layers**:
    - The model uses **three convolutional layers**:
        - **First Conv Layer**: 32 filters of size 5x5, ReLU activation, and L2 regularization.
        - **Second Conv Layer**: 64 filters of size 3x3, ReLU activation, and L2 regularization.
        - **Third Conv Layer**: 128 filters of size 3x3, ReLU activation, and L2 regularization.
    - **Max-Pooling Layers** follow each convolutional layer to reduce the spatial dimensions and retain key features.
3. **Dropout Layers**:After each convolutional block, a **dropout layer** with a rate of **0.5** is applied to prevent overfitting. This helps ensure the model generalizes well by randomly deactivating neurons during training.

4. **Fully Connected (Dense) Layer**:

   One Dense Layer: A fully connected layer with 512 neurons and ReLU activation. This layer enables the model to learn high-level features from the convolutional layers.

   The output from the fully connected layer is then passed through a **final dense layer** with a single neuron and a **sigmoid activation** function, which outputs a probability score for the binary classification task (0 or 1).

Model Parameters:

- **Learning Rate**: The Adam optimizer is used with a learning rate of **1e-4**.
- **L2 Regularization**: Applied to all convolutional and dense layers with a regularization strength of **0.02** to prevent large weight values and overfitting.
- **Dropout Rate**: Set to **0.5** for preventing overfitting and improving generalization.

Training Methodology

1. **Data Preparation**: The data is loaded from the HDF5 file, where spectrograms are normalized (values scaled between 0 and 1). The dataset is split into **training**, **validation**, and **test** sets.
2. **Early Stopping**:The model uses **early stopping** to stop training when the validation loss stops improving for 10 consecutive epochs, preventing overfitting.
3. **Learning Rate Scheduling**: ReduceLROnPlateau is used to reduce the learning rate by half if the validation loss doesn't improve for 5 epochs. This helps the model converge more efficiently during training.
4. **Model Checkpointing**:The best model is saved during training using **model checkpointing** based on validation loss. This ensures that the final model is the one that performs the best on the validation set.

Weight Updates:

During training, the **weights and biases** of the model are updated using **backpropagation** with **gradient descent**. The model adjusts the weights to minimize the loss (binary cross-entropy) and improve the classification accuracy. The optimizer (Adam) computes the gradients of the loss function with respect to each parameter and adjusts the parameters to reduce the error.

Accuracy and Loss:

During training, the **accuracy** and **loss** metrics are tracked for both training and validation sets. The **accuracy curve** shows how the model's performance improves over epochs, while the **loss curve** tracks the model's ability to minimize the error. Both curves help assess the model's progress and detect overfitting or underfitting.

Evaluation:

After training, the model is evaluated on the **test set**, providing the final **test accuracy** and **AUC (Area Under the Curve)**, which indicates the model's performance in classifying the two species. The final model is then saved for future use.

Multi-Class Classification Methodology

In this multi-class classification task, we are training a Convolutional Neural Network (CNN) to classify bird species from spectrogram data. The model is designed to classify data into multiple categories, using **softmax activation** in the final output layer for multi-class prediction. Below is the overview of the architecture and training methodology for this model.

CNN Architecture

The CNN architecture for multi-class classification consists of multiple layers to extract features from the spectrograms and predict one of several possible bird species:

1. **Input Layer**:
    - The input data consists of spectrograms with a shape of (128, 517, 1), representing grayscale images.

2. **Convolutional Layers**:
    - The model contains **three convolutional layers**:
        - **First Conv Layer**: 32 filters of size 5x5, ReLU activation, and L2 regularization.
        - **Second Conv Layer**: 64 filters of size 3x3, ReLU activation, and L2 regularization.
        - **Third Conv Layer**: 128 filters of size 3x3, ReLU activation, and L2 regularization.
    - **Max-Pooling Layers** follow each convolutional layer to reduce the spatial dimensions, helping the model focus on the most important features.
3. **Dropout Layer**:
    - After each convolutional block, a **dropout layer** with a rate of **0.3** is used to prevent overfitting, allowing the model to generalize better.
4. **Fully Connected (Dense) Layer**:
    - **One Dense Layer**: A fully connected layer with 512 neurons and ReLU activation. This layer connects the convolutional feature maps to the output layer.
    - The output from this dense layer is passed through a final dense layer with **softmax activation**, which outputs probabilities for each class in the classification task.

Model Parameters:

- **Learning Rate**: The Adam optimizer is used with a learning rate of **1e-5**.
- **L2 Regularization**: Applied to all convolutional and dense layers with a regularization strength of **0.02** to prevent large weights and overfitting.
- **Dropout Rate**: Set to **0.3** to reduce overfitting and improve generalization.

Data Preparation and Generator:

1. **Data Loading**:The spectrograms are loaded from an HDF5 file, normalized (scaled between 0 and 1), and split into training, validation, and test sets.
2. **Data Generator**: A data generator is used to yield batches of data during training. This is helpful for large datasets, as it generates batches on the fly.

Training Methodology

1. **Early Stopping**: The model uses **early stopping** to halt training if the validation loss does not improve for 10 consecutive epochs, preventing overfitting.
2. **Learning Rate Scheduling**: **ReduceLROnPlateau** reduces the learning rate by a factor of 0.5 if the validation loss stops improving for 5 epochs, helping the model converge more efficiently.
3. **Model Checkpointing**: It saves the best model based on validation loss during training. This ensures that the final model is the one that performs the best on unseen data.

Weight Updates:

During training, the **weights** and **biases** of the model are updated using **backpropagation** and **gradient descent**. The optimizer computes the gradients of the loss function (categorical cross-entropy) with respect to each parameter and adjusts the parameters to minimize the error. This is how the model learns to improve its predictions.

Accuracy and Loss:

During training, the **accuracy** and **loss** metrics are tracked for both training and validation sets. The **accuracy curve** shows how the model's performance improves over epochs, while the **loss curve** tracks the model's ability to minimize the error.

Evaluation:

After training, the model is evaluated on the **test set**, providing the final **test accuracy** and **AUC (Area Under the Curve)**. The final model is saved for future use.

Discussion

Limitations and Challenges Faced

During this assignment, I encountered various challenges in working with the CNN model for binary and multi-class problems.

1. Processing Grayscale Images:

Initially, the proper processing of grayscale images for the model was unclear. The spectrograms were in grayscale, and it was necessary to determine how to normalize the image data so that the model could effectively comprehend it. After conducting some research, it was found that dividing the pixel values by 255 would normalize the image data to a range of 0 to 1. This step was crucial because it enabled the model to learn more meaningful features from the data. For a deeper understanding of how this normalization improves model performance, a relevant StackOverflow discussion was consulted.

2. Training Without Early Stopping:

Early stopping was not being utilized initially during training, and the model was trained for only 300 epochs. Without early stopping, training would have taken almost an hour, and the model's performance was suboptimal. The accuracy curve indicated evident overfitting, with the spikes in the training accuracy being very high compared to being flat in the validation accuracy. This discrepancy asserted that the model was not properly generalizing and was memorizing the training data instead. To counter this, early stopping was used to prevent overfitting, and training was halted when validation accuracy stopped improving.

3.Training Time:

The binary classification model was trained for approximately 23 minutes for 200 epochs with learning rate 1e-4 and early stopping patience of 5.

The multi-class classification model, however, took around 40 minutes to train with the same parameters but for 200 iterations as well. The increased training duration in the multi-class model is due to the greater number of classes and task complexity involved in the classification.

Hard Species to Predict

Certain species were more difficult to predict due to issues in data quality and overlapping characteristics:

- houspa: The large dataset (630 samples) had inconsistent spectrograms, making it tougher for the model to capture distinctive features.
- norfli: Due to the low number of samples (37), the model struggled to generalize owing to the small amount of data.
- bkcchi: Similarly, the small dataset (45 samples) made it difficult for the model to separate this species properly.

In amerob and rewbla binary classification, the model performed poorly because their confidence levels were very similar, thus hard to distinguish. The predictions for these two species were typically almost equally probable, with amerob having a confidence of 44.6% and rewbla 55.4%.

Alternative Models for Neural Networks

For this exercise, there could have been different models applicable for bird species classification from audio data, yet the most appropriate is a neural network, especially a CNN, due to the nature of data (spectrograms) and the task nature.

Pretrained Models

Transfer Learning with pretrained models is a robust approach. Pretrained models on large image datasets, such as VGG16, ResNet, or Inception, can be used to classify bird species. The pretrained models have already optimized layers that are excellent at extracting image features, which may be a good fit for spectrograms as they essentially translate audio information into 2D image-like objects.

This approach allows leveraging powerful pre-trained weights, saving training time, and perhaps improving accuracy since the model already knows how to extract important features from visual data.

Audio-Specific Deep Learning Models

Alternatively, models like WaveNet or Raw Audio Neural Networks, specializing in audio classification, can be used. Such models operate on raw audio data (e.g., waveforms) and do not involve the conversion of data to spectrograms. This

may rule out information loss during the conversion phase to spectrograms, hence the model's performance should be improved while processing the audio data in its native form.

Dense Neural Networks:

A Dense Neural Network (DNN) would be another potential contender for this task. While CNNs are excellent at handling spatial data like images, a fully connected DNN can also be used, especially if the learned features from the spectrograms are already decent and well-preprocessed and dimensionally reduced. However, DNNs typically require more feature engineering and might not be as adept at handling tasks with spatial or sequential patterns like audio classification.

1. **Why CNN Makes Sense**:

- Since the work is working with spectrograms, which are essentially 2D pictures that detail the frequency makeup of sound as a function of time, a Convolutional Neural Network (CNN) is particularly well-adapted. CNNs are really designed for image classification, so they are ideally suited to work on tasks that involve inspecting 2D data such as spectrograms.
- CNNs are good at recognizing local patterns within data using sliding windows (filters), making them perfect for working with spectrograms. They can learn spatial hierarchies automatically and also pull-out suitable features like edges and textures, which is of the utmost importance in distinguishing between species of birds by their respective call features.

2. **Deep CNN Reference**:

   A paper was reviewed that highlights the advantages of using **Deep CNN models** over standard CNNs for tasks such as bird species classification. The paper emphasizes the effectiveness of deep CNNs in feature extraction from multi-dimensional data, showcasing their success in various image and pattern classification challenges. For further insights, the following paper provides a comprehensive discussion:
   Deep CNNs for Image and Pattern Classification

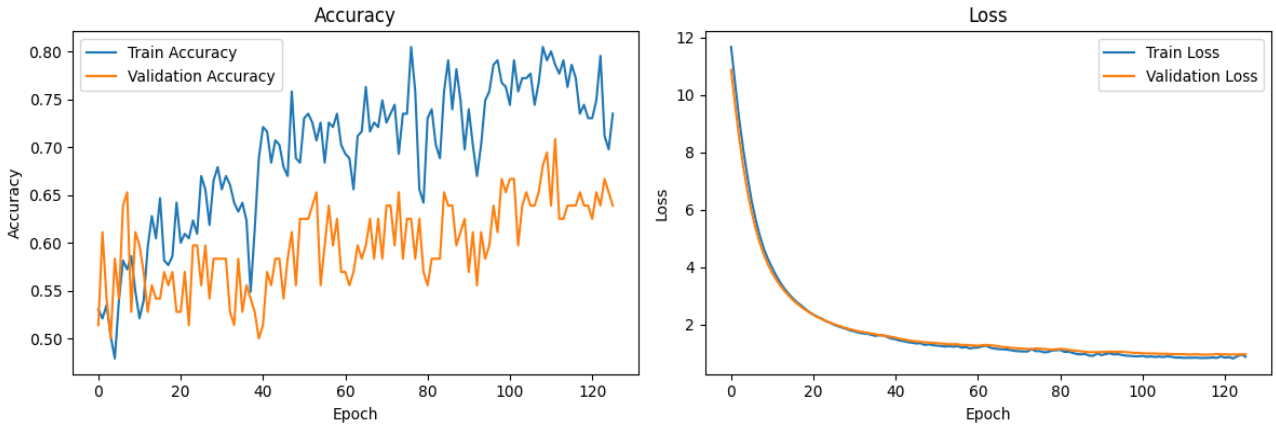
Result:

Binary Model:
The model was trained with only two species: **amerob** and **rewbla**.

Binary Model Best Model and Output:

First model:

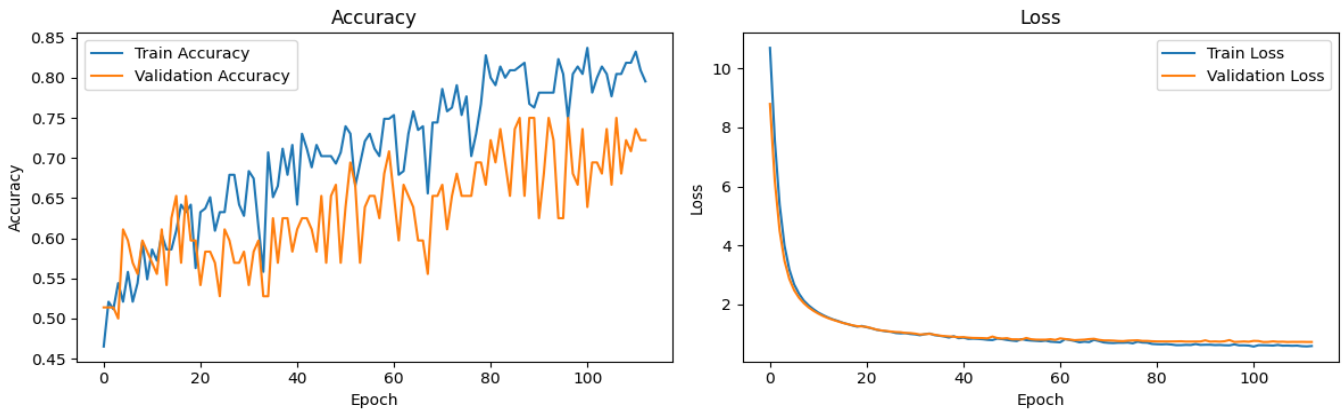
- The test accuracy is 62%, which is low.



For this model:

- The **first audio** in the test data is predicted as **rewbla** with **50%** confidence.
- The **second audio** is predicted as **rewbla** with **55%** confidence.
- The **third audio** is predicted as **amerob** with **52%** confidence.

Second Model:



- The test accuracy is **65%**, which is low but still better compared to the previous model.
- The training accuracy improves steadily, but the validation accuracy fluctuates and remains lower, suggesting some overfitting. The loss decreases for both training and validation, but the validation loss doesn't improve as much, indicating that the model isn't generalizing well to unseen data.

For this model:

- The **first audio** in the test data is predicted as **rewbla** with **57%** confidence.
- The **second audio** is predicted as **rewbla** with **70%** confidence.
- The **third audio** is predicted as **amerob** with **70%** confidence.

The model with the following parameters:

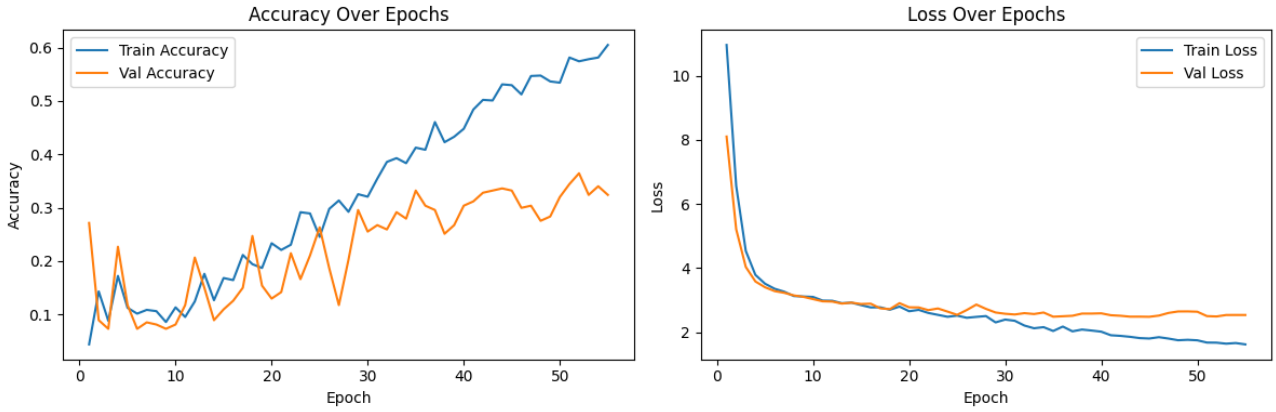
- **Batch size**: 32
- **Epochs**: 200
- **Learning rate**: 1e-4
- **Dropout rate**: 0.5
- **L2 regularization**: 0.01

performed reasonably well based on the confidence of the predictions from the test data. Despite the lower accuracy, the model showed a consistent trend in its predictions, with confidence levels reflecting its ability to classify the two species.

Multiclass Model:

The first model was created with the following parameters:

- **Batch size**: 64
- **Epochs**: 100
- **Learning rate**: 1e-4
- **Dropout rate**: 0.4
- **L2 regularization**: 0.01



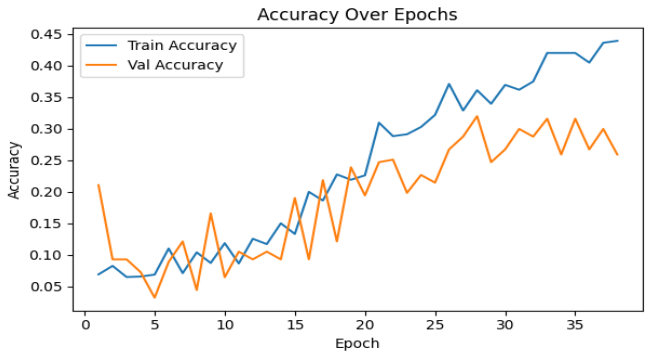
The model has 35% test accuracy but the validation accuracy and train accuracy curves are not matching, and the loss curve is also poor, indicating that the model is not performing well. This can be used to conclude that testing with audio data may not be suitable for this model. The values of confidence indicated by the bar chart do indicate, however, that the model is struggling to generalize, as the confidence values of the species predictions are not consistent.

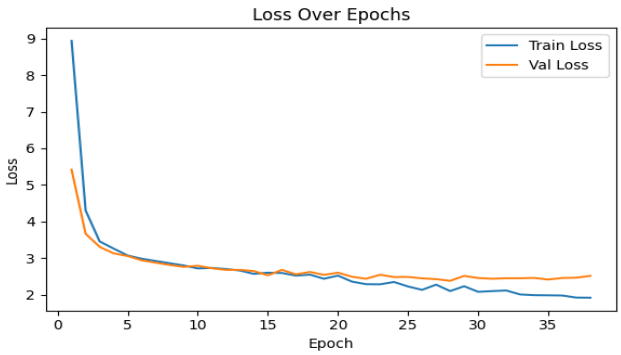
Second Model Results

The second model was trained with the following parameters:

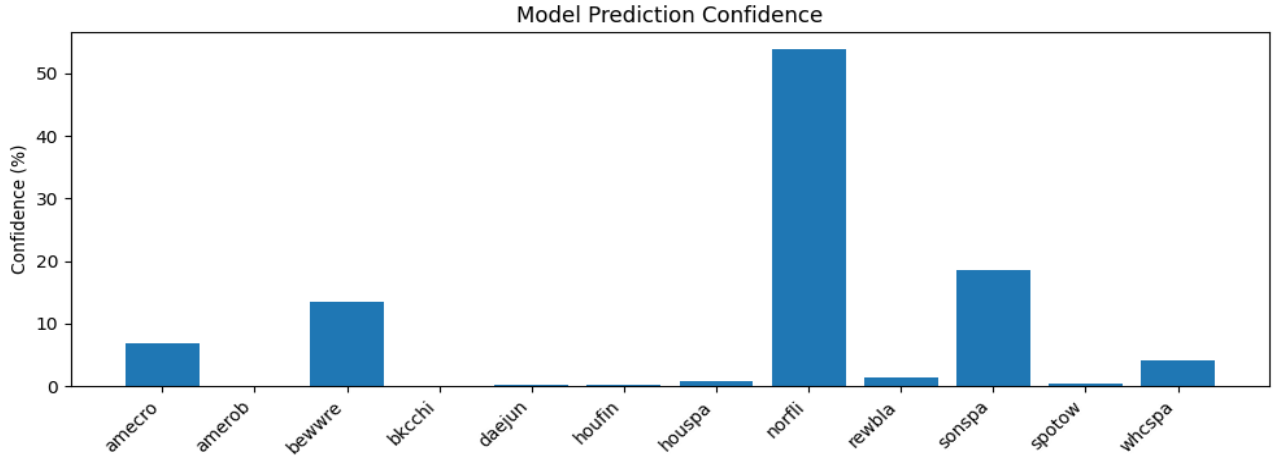
- Batch size: 32
- Epochs: 200
- Learning rate: 1e-4
- Dropout rate: 0.6
- L2 regularization: 0.01

The test accuracy of this model is 32%, which is higher than the first model's accuracy, 22%. The gap between the training and validation accuracy curves is a bit lower than that of the first model, but the model is not performing at its best.





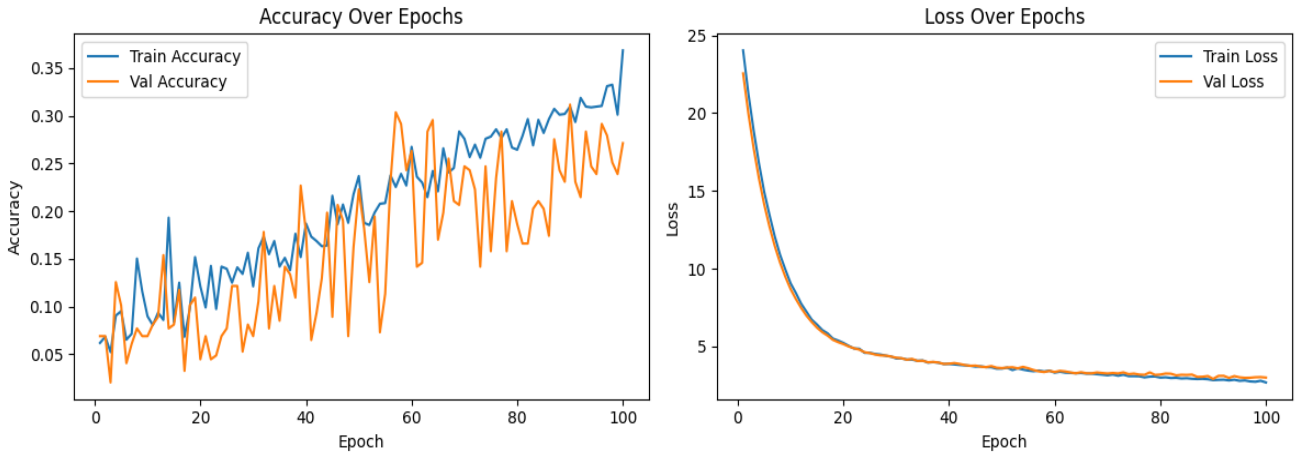
Prediction confidence is more balanced compared to the first model, with greater separation between species. While norfli is predicted with the highest confidence, the other species still have non-negligible confidence values, which means the model is slightly more generalized than the first.



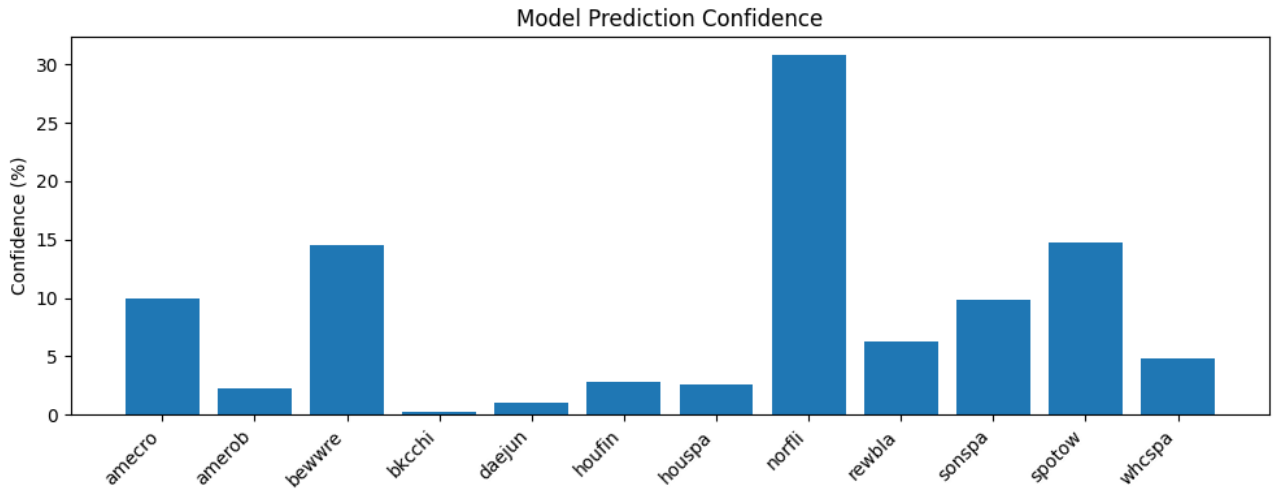
Third Model Results

The third model was trained on the following parameters:

- Batch size: 32
- Epochs: 200
- Learning rate: 1e-5
- Dropout rate: 0.3
- L2 regularization: 0.02



The prediction for audio 2 by model 3



This third model was more consistent than the previous ones
in both accuracy and generalization. Its steady training curve, consistent output, and improved performance make it a strong model for bird species classification based on spectrogram data.

References:

Advantage of dividing by 255
Stack Overflow. Why we should use gray scale for image processing. Retrieved from https://stackoverflow.com/questions/12752168/why-we-should-use-gray-scale-for-image-processing

Early Stopping Usage
TensorFlow. EarlyStopping callback. Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

Reducing Learning Rate on Plateau
TensorFlow. ReduceLROnPlateau callback. Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau

Model Checkpoint
Stack Overflow. (2021). Saving model checkpoint in TensorFlow. Retrieved from https://stackoverflow.com/questions/70723307/saving-model-checkpoint-in-tensorflow

Deep CNN Reference for Other Applications
Deep Convolutional Neural Networks in data classification and pattern recognition applications. https://www.mdpi.com/1099-4300/23/11/1507

Reference on Similar Application for Architecture and Solution Formation
Architectural design for bird species classification. Expert Systems with Applications https://www.sciencedirect.com/science/article/pii/S1574954120300637

Theoretical Background on CNN Architectures
Convolutional Neural Networks (CNN) Architectures Explained. Retrieved from https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243