Bird Sound Classification Using Deep Learning

Abstract

This report delves deep into the application of neural networks for classifying bird species based on the sounds it makes. The data used are the spectrograms generated from 3-seconds audio clips across twelve bird species which are taken from the Xen-Canto a crowd-sourced birds sounds archive. Using this data, I have developed binary and multi-class convolutional neural network models and compared variation in architecture and hyperparameters to get the most effective architecture. Model performance is evaluated using accuracy and confusion matrix. The report also demonstrates the potential and limitations of neural networks for this specific bird species identification application.

Introduction

In this project, Neural networks are used to classify bird species based on their sounds. The primary objective is to explore how deep learning models, mainly Convolutional Neural Networks(CNN) , can be applied to spectrograms (Time-Frequency representation) for effective classification of birds.

The data is collected from the Xeno-Canto bird recordings which contains recordings of 264 species. Out of all, 12 bird species are chosen for this project and high quality sounds are selected for each species. Spectrograms were then generated from the first 3 seconds of each sound clip which are fed as an input to the neural network.

The project explores two major classification tasks:

- Binary Classification: Distinguish between American robin (amerob) and red-winged blackbird (rewbla).
- Multiclass Classification: Distinguish between all 12 bird species [American crow (amerco), American Robin (amerob), Bewicks Wren (bewwre), Black-capped chickadee bkcchi), dark-eyed junco (daejun), house finch (houfin), house sparrow (houspa), northern flicker (norfli), red-winged blackbird (rewbla), song sparrow (sonspa), spotted towhee (spotow), white-crowned sparrow (whcspa)].

For the above classification tasks, I implemented and trained CNN with different hyperparameters tuning.

Theoretical Background

Deep learning is a rapidly evolving field in machine learning and artificial intelligence, with neural networks being the foundational block.

	Traditional Machine Learning	Deep Learning
Feature Extraction	Required	Not Required
Dataset	Deals with small datasets	Larger datasets are required
Computation complexity	Low	Require high computing

Most Common Deep Learning Techniques are Deep Neural Network (DNN) which can solve complex problems and works well with structured data, Convolutional Neural Network (CNN) are used for the data which can be represented in a grid-like or sequential form where neighboring values are meaningful, Recurrent Neural Network (RNN) for time series, and also works well with language or audio data.

Neural networks are inspired by the human brain which typically consists of interconnected nodes (neurons) arranged in layers. The network receives an input vector of p variables X = (X1, X2, … Xp) and builds a nonlinear function f(X) to predict the response Y.

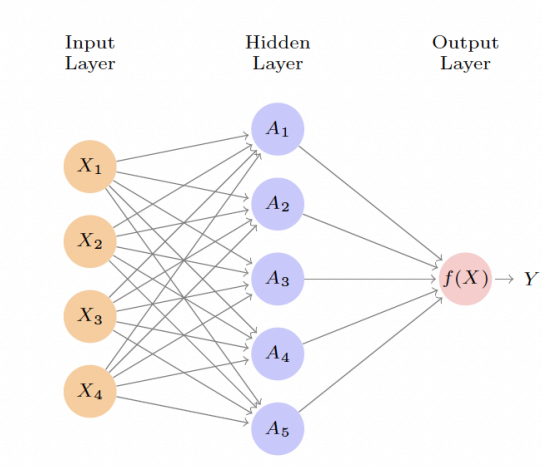


Figure 1: Neural network with single hidden layer

The hidden layer computes activations Ak = hk(X) that are nonlinear transformations of linear combinations of the inputs X1, X2,...,Xp. Hence these Ak are not directly observed. The output layer is a linear model that uses these activations Ak as inputs, resulting in a function f(X). There are many different types of activation functions. Some examples are:

- Linear
- ReLU: The preferred choice in modern neural networks. A ReLU activation can be computed and stored more efficiently than a sigmoid activation.

$$g(z) = max(0, z)$$; where z is the input

- Sigmoid : This is the function used in logistic regression to convert function into probabilities between zero and one.

$$g(z) = \frac{e^z}{1 + e^z}$$

There are many hyperparameters in the neural networks which determine the network structure and the variables which determine how the network is trained. These hyperparameters are set before training(before optimizing the weights and bias).

Pre-determined hyperparameters are:

Number of neurons in the input layer

- This will depend on the data that we are using for the training. For instance if we have n features in the dataset we will have n input neurons. If we have the image as an input with n*n pixels, we will have input neurons.

Number of neurons in the output layer

- This will also depend on the data that we are using for the training. For instance if we are doing binary classification or regression, single output will be sufficient. If we are having multi-class or multi-label class classification problems, we need a number of output neurons as classes/labels.

Hyperparameters that need tuning

Number of hidden layers

- The more hidden layers we have, the deeper the network will be. In general having more layers will help network more than having more number of neurons in the layer.

Number of neurons in the hidden layers

- Each hidden layer can have a different number of neurons. This will be changed based on how the network is performing.

Activation Function

- Each layer can have any type of activation function except the input layer and moreover the activation function of hidden layers cannot be linear.

Optimization Algorithms

- This algorithm determines how the network updates its weights. Most common algorithms are Stochastic Gradient Descent(SGD), Adam, RMSProp, AdaGrad.

Loss Function

- The loss (cost) function is what we're trying to minimize during training. This changes depending on the problem we have.

Batch size

- Batch size is the amount of data points we put in each batch while training the network.

Number of epochs

- This is basically how many times we run the whole dataset through the network. This helps the network learn the dataset a little bit better after each epoch.

Learning Rate

- Learning rate determines how fast/slow steps need to take towards the direction set by gradient descent. Small learning rate slows down the learning and too big might overshoot the optimum value.

Regularization

- This helps reduce overfitting. There are many ways to do regularization (L1, L2, Dropout,etc.) these often come with their own hyperparameters.

There are various types of neural networks each with their own architecture and characteristics. In this project I have used convolutional neural networks.

Convolutional Neural Network(CNN)

These are a special family of neural networks evolved for classifying images and have shown spectacular success on a wide range of problems. This combines two specialized types of hidden layers, called convolution layers and pooling layers. Convolution layers search for instances of patterns in the image, whereas pooling layers downsample these to select a subset.

Methodology

Data Preparation

The data is collected from Xeno-Canto, a crowd-sourced bird sounds archive. There are many sound clips available and out of all sound clips with high quality of 12 species are collected. The sound clip is then subsampled to 22050 Hz and the first 3 seconds of the sound clip are selected. The spectrogram is derived from an audio signal for each 2-second window, resulting in a 128(frequency) x 517 (time) image of the bird call. All bird calls for all clips in a given species are saved individually which results in an uneven number of samples in each species ranging from 37-630 samples. These spectrograms are used as an input to the neural network. The dataset is split into 80% training set and 20% testing set.

For the binary classification task I used spectrograms of American Robin (amerob) and Red-winged Blackbird(rewbla). Whereas for multi-class classification I used spectrograms for all the 12 bird species and labelled each class from 0 to 11.

Binary Classification

In this classification task, two birds American Robin (amerob), Red-winged Blackbird (rewbla) are being classified. Three different models are used for this classification task. The first model is the basic one with one hidden layer and ReLU activation function and one single dense layer with sigmoid activation function since this is a binary classification. The optimization algorithm used is Adam, loss function is the binarycrossentropy and metrics is the accuracy. The subsequent models are created by adding dropout rate, increasing the number of epochs and also increasing the number of hidden layers for better generalization. Early stopping is included to avoid overtraining by stopping when validation loss stops improving.

Model1

Neural Network Architecture

- Two convolutional blocks:
    - Each includes Conv2D, Maxpooling2D.
    - The number of filters increases progressively: 32 → 64
- Flatten which converts features to 1D vector.
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = ‘sigmoid’).

Training Configuration:

- Optimizer: Adam with a learning rate of 0.0001 for stable convergence
- Loss Function: binary_crossentropy
- EarlyStopping with patience = 10
- Epochs = 10, batch_size = 16

Model2

- Two convolutional blocks:
    - Each includes Conv2D → MaxPooling2D → Dropout(rate = 0.2)
    - The number of filters increases progressively: 32 → 64
- Flatten which converts features to 1D vector.
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = ‘sigmoid’).
- Epochs = 20, batch_size = 16

Model3

- Four convolutional blocks:
    - Each includes Conv2D → BatchNormalization() → MaxPooling2D → Dropout(rate = 0.2)
    - The number of filters increases progressively: 16 → 32 → 64 → 128
- GlobalAveragePooling2D ()
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = 'sigmoid').
- Epochs = 50, batch_size = 16

Training Configuration:

- Optimizer: Adam with a learning rate of 0.0001 for stable convergence
- Loss Function: binary_crossentropy
- EarlyStopping with patience = 10
- Validation Split: 20% of the training data is used for internal validation.

Multi class Classification

In this classification task, 12 bird species are being classified. To prepare the spectrogram data for training a convolutional neural network in a 12-class bird species classification task, each species shape was stored as a spectrogram array of shape (128, 517, N) where N is the number of recordings for that species. Each spectrogram was transposed and reshaped to a consistent format (N, 128, 517, 1) 1 is for the single grayscale channel. Class labels were numerically encoded and one-hot encoded for multi-class classification. This pipeline ensured that all data was consistently formatted, labeled, and ready for training using categorical cross-entropy loss. Three different models were created.

Model1

The model performed well in the binary classification is used as the base model. The architecture configuration is below

Neural Network Architecture

- Four convolutional blocks:
    - Each includes Conv2D → BatchNormalization → Maxpolling2D → Dropout rate(0.25)
    - The number of filters increases progressively: 16 → 32 → 64 → 128
- GlobalAveragePooling2D is used instead of flattening, reducing overfitting by averaging feature maps spatially
- A dense layer with 128 neurons precedes the final classification layer.

- The final Dense(12, activation = ‘softmax’) outputs probabilities over 12 classes.

Training Configuration:

- Optimizer: Adam with a learning rate of 0.0001 for stable convergence
- Loss Function: categorical_crossentropy, appropriate for multi-class classification with one-hot labels
- EarlyStopping with patience = 10 halts training if no improvement in validation loss is observed.
- Epochs = 30, batch_size = 16

Model2

- Six convolutional blocks:
    - Each block includes Conv2D → BatchNormalization → MaxPooling2D → Dropout(0.2)
    - The number of filters increases progressively 16 → 32 → 64 → 128 → 256 → 512
- GlobalAveragePooling2D is used instead of flatten to reduce the number of parameters.
- A dense layer of 128 neurons with ReLU activation function
- The final Dense(12, activation = ‘softmax’)
- Learning rate = 0.0001, epochs = 30, batch_size = 16

Model3

- Six convolutional blocks:
    - Each block includes Conv2D → BatchNormalization → MaxPooling2D → Dropout(0.2)
    - The number of filters increases progressively 16 → 32 → 64 → 128 → 256 → 512
- GlobalAveragePooling2D is used instead of flatten to reduce the number of parameters.
- A dense layer of 128 neurons with ReLU activation function
- The final Dense(12, activation = ‘softmax’)
- Learning rate = 0.0005, epochs = 100, batch_size = 32

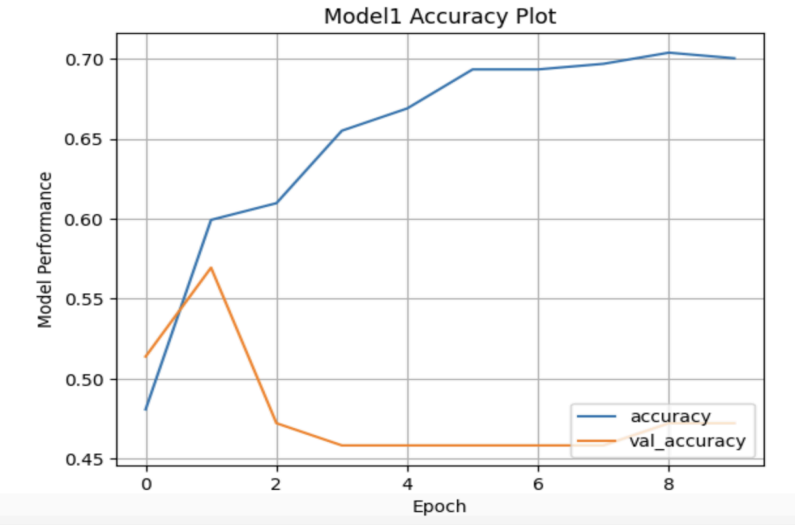
Computational Results

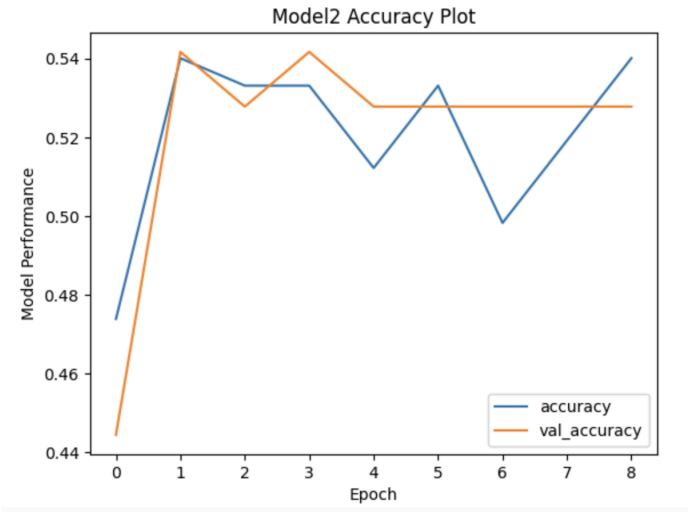
Binary Classification

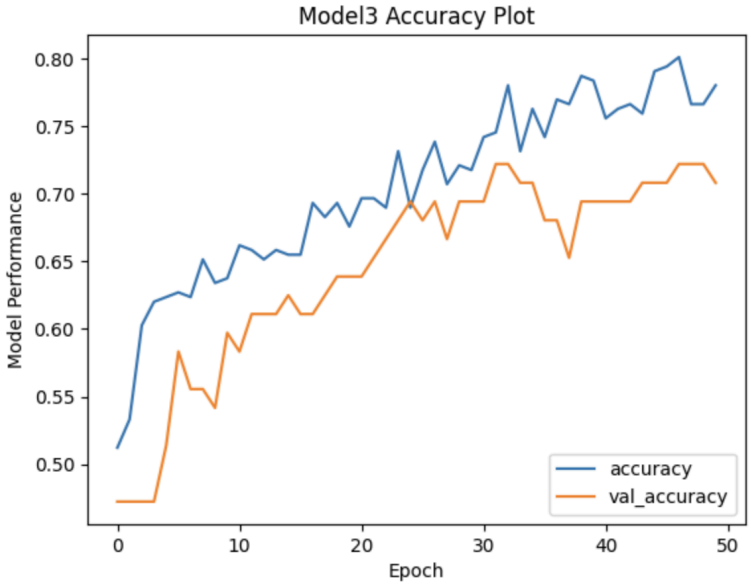
Metric	Model1	Model2	Model3
Test Accuracy	47.22%	54.16%	72.08%
Train Accuracy	70.38%	52.26%	74.21%
Precision	American Robin - 0.33 Red-winged Blackbird - 0.50	American Robin - 1.00 Red-winged Blackbird - 0.54	American Robin - 0.77 Red-winged Blackbird - 0.66
Recall	American Robin - 0.12 Red-winged Blackbird - 0.79	American Robin - 0.03 Red-winged Blackbird - 1.00	American Robin - 0.50 Red-winged Blackbird - 0.87
Time	1.40s	3:26 min	12.59 min

From the above table we can see the model 1 has 47.22% accuracy. While recall for Red-winged Blackbird is relatively high (0.79), it under-performed for American Robin with very low recall(0.12), which shows the model is biased towards the class1(Red-winged Blackbird). In case of model2 a precision of 1.00 and recall of 0.03 for American Robin shows that the model is only predicting this class resulting in no false positives but missing all the true positive cases. This is likely due to data with fewer samples of each class which is why it is failing to detect most instances of American Robin despite being perfectly accurate when the model does.
The model3 has performed well with better accuracy which is achieved by adding more layers and increasing the learning rate. American Robin recall is improved to 0.5, precision is 0.77 and overall accuracy is 72.08%. This model effectively handles class imbalance by maintaining the balance between the recall and precision.

Though the model 3 performed well it took much longer time to run when compared to the basic model in my local computer. But the time taken to run is reduced by running the code on T4 GPU in google colab.







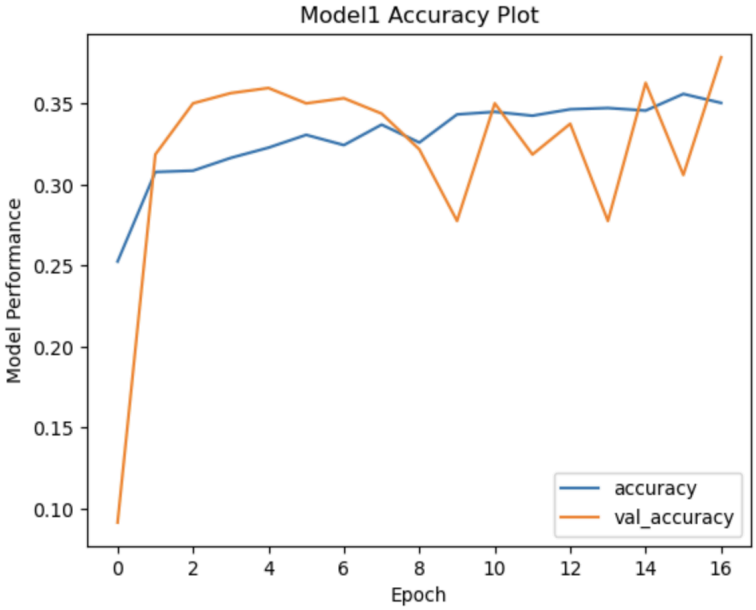
Above are the plots of training and validation accuracy across epochs for each model. Model 1 is the basic one but shows overfitting with poor generalization training accuracy increases while validation accuracy drops early which is confirmed with the large gap between the two plots. Also the model converges early at around 3-4 epochs.
Model2 is the improvisation of model1 which slightly performed better than model 1.
Model 3 outperformed in terms of accuracy and recall. The overfitting is reduced when compared to the other two models.

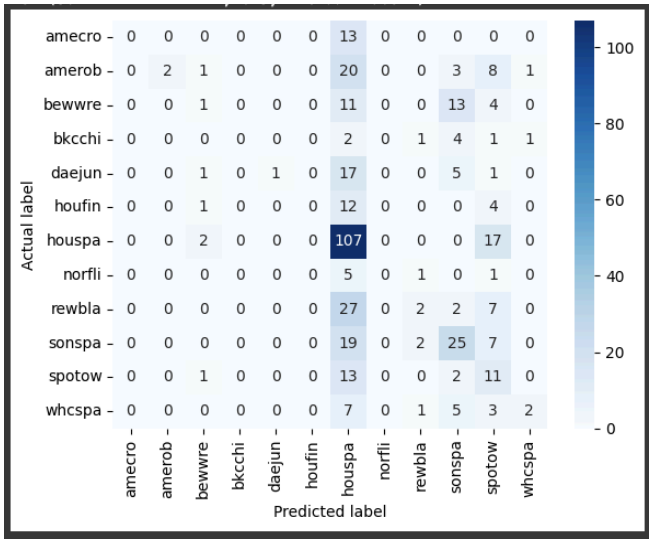
Multi-class Classification

This multi-class classification model tried to distinguish between 12 different bird species. Above is the table with the results of three multi class models with performance comparison.

Metric	Model1	Model2	Model3
Test Accuracy	38%	49.8%	61%
val_auc	0.7865	0.8597	0.8864
Test_loss	1.9586	1.6315	1.05913
Time	5.90 min	15.49min	45.66min

Model 1

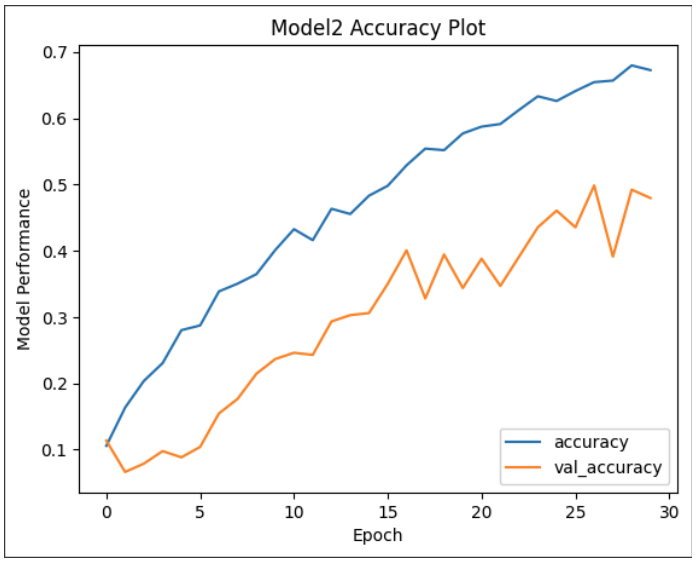


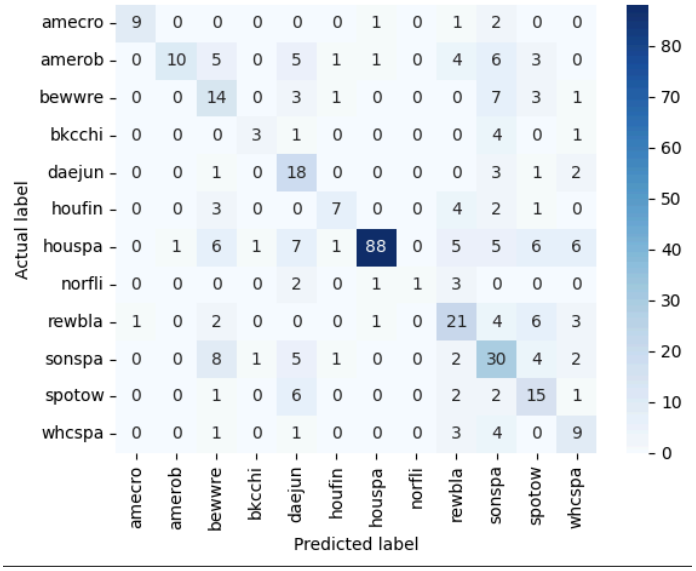


Model 1 did terribly with 38% accuracy and had an AUC of 0.7865. Despite the low accuracy the mode is still relatively good at distinguishing between classes and assigning higher probabilities to correct classes. If we see the confusion matrix, most of the species classified by the model are House Sparrow which is due to the class imbalance. Minority classes such as American Crow, Northern Flicker,

Black-capped Chickadee are never correctly classified. This reflects the model bias towards the majority classes.

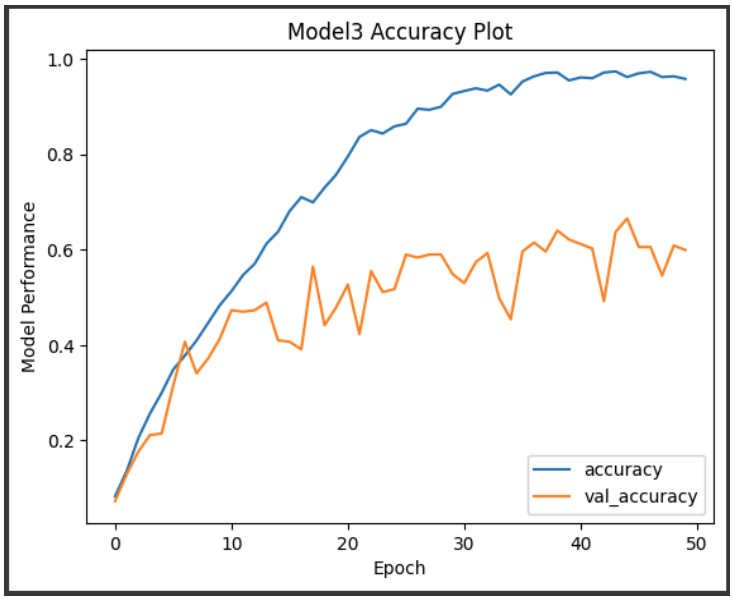
Model 2

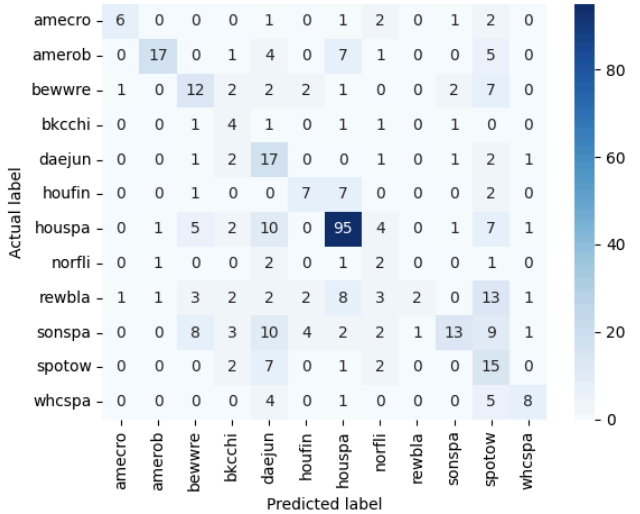




Model2 achieved 49.8% test accuracy and a strong AUC of 0.8597 which shows improved ranking of class probabilities. These results shows that the deeper architecture with more hidden layers significantly improves the model's ability. Also, from the confusion matrix we can see predictions are now more distributed across classes. Though results shows balanced learning but still underperforms on some majority classes.

Model 3



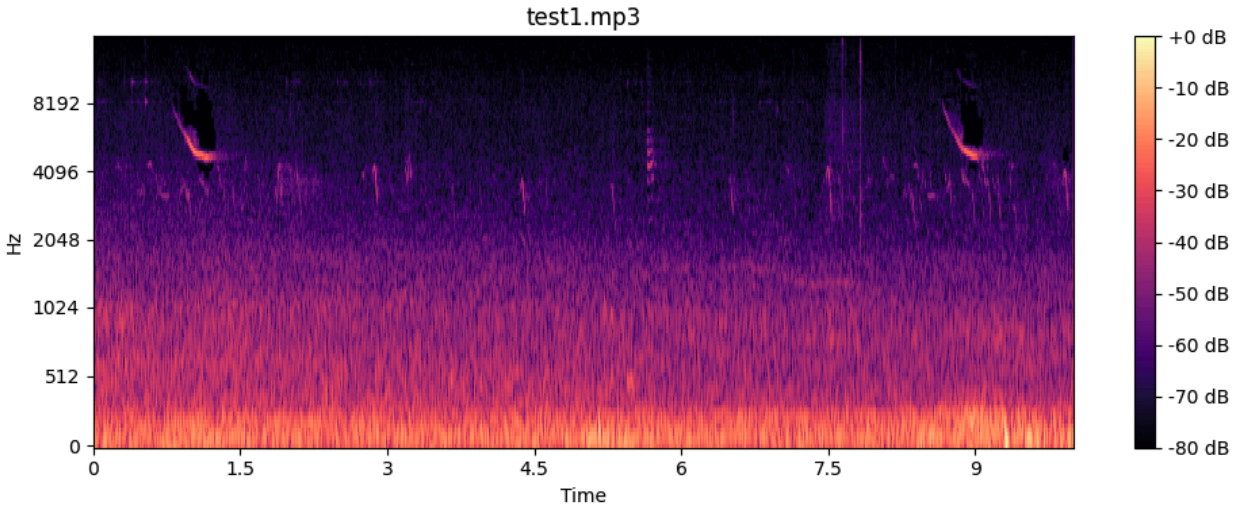


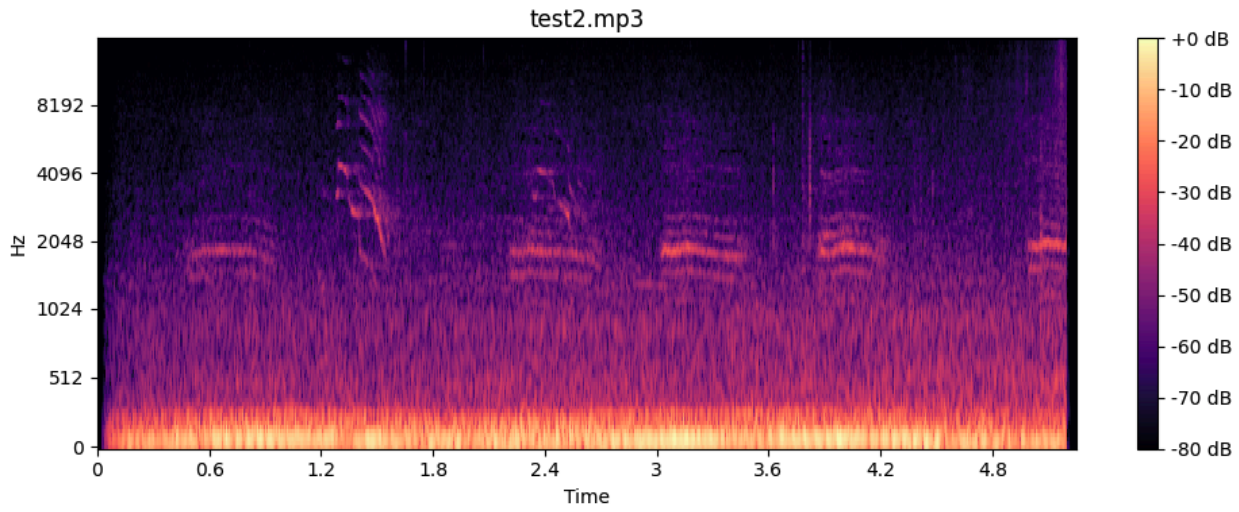
Overall Model3 performs well with accuracy of 61% and AUC of 0.8864. The model achieved well balanced and accurate predictions across all species. The training parameters of model3 with longer training, higher learning rate and increasing the batch_size from 16(model2) to 32(model 3), the model tried to extract and generalize complex patterns from the spectrogram data.

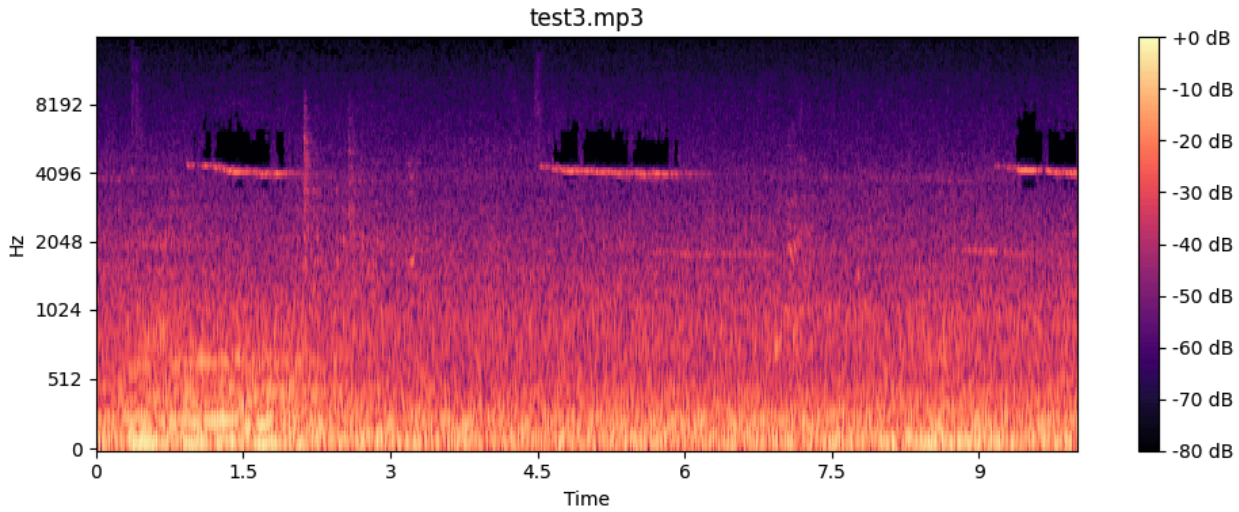
Test Data Predictions

The raw MP3 audio clips are segmented into 3-second windows, converted into spectrograms. These spectrograms are converted to grayscale and resized. Then predictions are made using the model3 which achieved better accuracy in multi-class classification.

File	Prediction 1	Prediction 2	Prediction 3
Test1.mp3	Northern Flicker - 0.25	Dark-eyed Junco - 0.21	White-crowned sparrow - 0.18
Test2.mp3	Dark-eyed Junco - 0.39	Northern Flicker - 0.23	American Crow - 0.15
Test3.mp3	White-crowned sparrow - 0.29	Northern Flicker- 0.25	Song Sparrow - 0.10







From the above spectrograms and prediction results, For Test1.mp3 spectrogram shows multiple distinct frequency patterns at different time intervals, this indicates more than one bird call could be contained in the clip. This illustration is supported by prediction probabilities as shown in the table.

Test2.mp3 has a more consistent spectrogram pattern typically characteristic of single bird song. The model assigned the highest probability to Dark-eyed junco with 0.39.

Test3.mp3 also contains irregular patterns in between which indicates there might be multiple bird calls in the audio. This is also supported by similar probability values for white-crowned sparrows(0.29) and Northern Flicker(0.25).

Discussion

In this project I have explored a range of convolutional neural network architectures with varying hidden layers and hyperparameters to classify bird species using spectrogram data. Through this I observed how different hyperparameters, including learning rate, batch size, number of epochs influence overfitting, convergence and generalization. For instance in multi-class classification, Model1 with fewer number of hidden layers, epochs and batch size overfitted and performed poorly on the test data set. In contrast, Model3 with six convolutional blocks, higher learning rate, larger batch size and higher number of epochs achieved better generalization with a test accuracy of 61% and validation AUC of 0.886.

In addition to this to stabilize learning and reduce overfitting, I incorporated Batch Normalization and Dropout (0.2 - 0.3) across all layers. Also incorporated early stopping with patience = 10 to halt training if the validation loss doesn't decrease for 10 epochs. Furthermore, due to the class imbalance in the dataset I have used class weights to help the model generalize well.

One of the limitations I encountered while working was the training time required for complex neural networks. In particular, the CNN with 6 convolutional blocks and 100 epochs took 45 mins to train using my macbook. To address overfitting, I tried using early stopping with more epochs(>150) which took more than 90 mins.

Neural Network for this application makes sense as the network can learn and extract features from the complex spectrogram data. In order to increase the performance of the model by using a transfer learning approach using pre-trained models like VGGish, PANNS, etc.

Conclusion

In conclusion, this project delves deep into the effectiveness of convolutional neural networks to classify bird species based on the spectrogram data. The binary classification model performed well with four convolutional blocks each followed by batch normalization, max pooling and dropout(0.2). Whereas in case of multi-class classification model results revealed that due to limited number of samples to the neural network can significantly hinder the model performance.

Despite the challenges and limitations, this project provides insights for further improvements and shows how changing the hyperparameters in the neural network can lead to better performance.

References

1. *James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An Introduction to Statistical Learning with Applications in Python. (Original work published 2023)* https://hastie.su.domains/ISLP/ISLP_website.pdf.download.html
2. *Rao, R. (n.d.). Xeno-canto Bird Recordings Extended (A-M)https://www.kaggle.com/datasets/rohanrao/xeno-canto-bird-recordings-extended-a-m*
3. Keras https://keras.io/