BIRD CALL CLASSIFICATION - USING DEEP LEARNING - CNNs

By HRISHABH KULKARNI

Github Link: https://github.com/Hrish-ProCoder/SML2_Hrishabh_Kulkarni

INTRODUCTION

This project explores the application of Deep Learning to classify bird species based on short audio clips using spectrograms. Bird call classification plays an important role in ecological monitoring and conservation efforts. It helps us or researchers track biodiversity and assess environmental health. The primary objective is to train models capable of accurately identifying bird calls among 12 species.

The **dataset consists** of `.mp3` files, converted into spectrograms. And additionally it has the External test data includes 3 `.mp3` clips for model testing.

Key Questions for my project are that will answer if CNNs distinguish Seattle bird species based on their vocal patterns? It also asks about which model architectures and hyperparameters leading to better classification performance?

Tools & Libraries Used: I used Python as the main programming language for this project. To build and train deep learning models, I used TensorFlow and Keras. For handling numerical operations and arrays, NumPy was used. Scikit-learn supports tasks like splitting data and measuring model performance. I used Matplotlib to create visualizations of the results. Librosa was used to process the audio data and extract useful features from the bird sound recordings.

Methods Applied in this project were Spectrogram generation and normalization, then CNN architecture with regularization and dropout. To improve model's performance I used Image Data Generator for augmentation. To test and compare the results implemented Model evaluation with accuracy, precision, recall, F1-score, and ROC AUC. I after preprocessing was facing issue of class imbalance. I then added class weighting to address issue of imbalanced data.

THEORETICAL BACKGROUND

Neural Networks: These are computational models which are inspired by human brain and neural structure. This contains interconnected layers which help to get and predict results. A neural network consists of an input layer, hidden layer with activation functions (like Relu, Sigmoid, etc.) and finally the output layer.

Convolutional Neural Networks (CNNs) are built and run on or designed for image-like data (e.g., spectrograms). This is because they extract spacial / important and temporal features effectively in proper order or hierarchy.

CNN apply filters to detect patterns in those images, then it is followed by pooling layers to reduce dimensionality.

Regularization and Stability used by me are **L2 Regularization** which Penalizes large weights to prevent overfitting. I also used **Batch Normalization** which Normalizes activations to stabilize training. Use of **Dropout** helped me Randomly disables neurons during training to improve generalization.

The Optimizers: which were used were Adam and RMS Prop.

- **Adam**: Combines momentum and RMSprop for adaptive learning
- **RMSProp**: Suitable for handling noisy or sparse data with moving average updates

Callbacks:

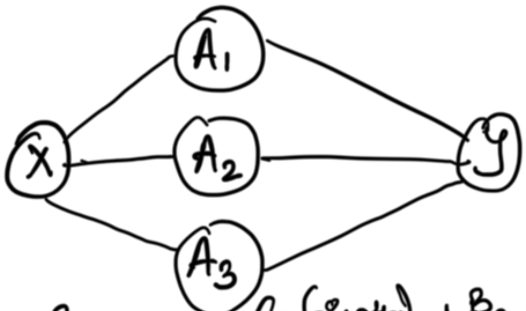
- **EarlyStopping**: Stops training when validation loss stops improving
- **ReduceLROnPlateau**: Lowers learning rate when a monitored metric stalls

Deep Learning

Neural Networks

- → Input Layer → Hidden Layer → Output Layer
- → connected by weights
- → Weights → are just Matrices
- → Hidden Layer has Activation Functions in 2^{nd} Layer
- → Each Layer has $\underbrace{\text{Activation Function}}_{\text{Any non-Linear functions}}$
- → First linear transformation, then in Hidden Layer (non-linear transformation)

Eg



Let's dive in an example

$$f(x) = \underset{B_0}{1.25} + \underset{B_1}{0.5}\left(\underset{A_1}{g(-8+0.4x)}\right) \underset{B_2}{-1}\left(\underset{A_2}{g(1+0.25x)}\right) \underset{B_3}{-0.75}\left(\underset{A_3}{g(-28+0.6x)}\right)$$

help with non-linear activation function.

$Ax+b$

Dropout Layer for Regularization.

Use blogspots – nodes at random are ignored.
– Randomly drops input & hidden Layer.

Sklearn → test train split

Sequential Neural Network → used Relu, input shape, output shape,

CNN

→ Divide image into diff. parts

⇒ Convolution Layer → filters
↳ vertical, horizontal, diagonal

eg. 3x3 filter

→ Filters smaller set of features from large image.
→ Eg. Relu Activation Function.

⇒ Pooling Layer → Avg./taking maximum

eg. Max pool of each layer & create a Matrics.

→ Used to condense information
→ Max. Pooling → helps maximize pooling in given block of image.

⇒ Data Augmentation → eg. zooming, shifting.

Use already done weights on your data/layers.

Document Classification :

1. Bag of Words

One Hot Coding

Sparse Matrix Representation → Saves ton of data → Saves loction of 1

Low Dimensional embeddings → Learns low dimensional of one hot encoded data. → Uses weights 0 and 1

Batch Size → 1/10 th of Dataset.

Downside of Bag Of Words

- No Content
- Text order is important
- Many unique words leading higher-dimensional model.

So, we make use of weights as 0 & 1 to better writes/make sentences:

- Bag Of n-grams

RNN (Recurrent Neural Network)

→ Considers ordering of inputs.
→ Learns sequences of Data.
→ Share weights with successive elements.

Applications

- Time Series Data
- Stock Price Prediction
- Weather Prediction
- Music Generation
- Video/Image Analysis/Generation
- Audio Recognision

Long Short Term Memory (LSTM)

→ This network considers longer past information into consideration.

Transformer :

→ Better at learning content.
→ Uses various attention modules
→ Example : • He took the umbrella with him in his bag.
• It was blue in color in that red storage.

→ So, here it helps keeping in mind the content that:
• It is umbrella &
• storage is bag

→ Tools like : BERT, GPT, Gemini

METHODOLOGY

Data Preparation

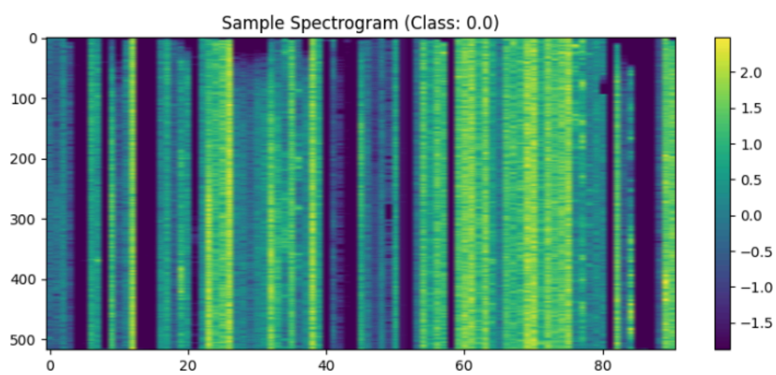
Audio clips were converted into spectrograms using Librosa, then normalized and cropped to a consistent length. The dataset was split into 76% training and 24% testing data. Augmentation techniques like horizontal/vertical shifts (10% range) were applied via Image Data Generator to improve generalization.

Binary Classification Model

Bird Species:

- 'whcspa' (White-crowned Sparrow)
- 'rewbla' (Red-winged Blackbird)

```
Class distribution - White-crowned Sparrow: 128, Red-winged Blackbird: 128
```



Understanding and finding: Here I have scaled my spectrogram from -2.0 to 1.5 suggesting improper normalization. I needed to scale between 0 and 1 to make the model consistent.

Model Configurations:

- **Adam Config**:
    - Filters: [4, 8]
    - Dropout: 0.5
    - Dense Units: 16
    - L2 Regularization: 0.01
    - Optimizer: Adam (LR = 1e-5)
- **RMSprop Config**:
    - Filters: [4, 8]
    - Dropout: 0.2
    - Dense Units: 8
    - L2 Regularization: 0.02
    - Optimizer: RMSProp (LR = 1e-5)

Training Setup:

- Epochs: 60
- Batch Size: 16
- EarlyStopping (patience=5)
- ReduceLROnPlateau (factor=0.5)

```
Epoch 14/60
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - AUC: 0.7526 - Precision: 0.7431 - Recall: 0.7101 - accura
cy: 0.7211 - loss: 1.0675 - val_AUC: 0.7647 - val_Precision: 0.6842 - val_Recall: 0.7647 - val_accu
racy: 0.7368 - val_loss: 1.0664 - learning_rate: 2.5000e-06
Epoch 15/60
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - AUC: 0.6351 - Precision: 0.6420 - Recall: 0.6854 - accura
cy: 0.6444 - loss: 1.1028 - val_AUC: 0.6569 - val_Precision: 0.5652 - val_Recall: 0.7647 - val_accu
racy: 0.6316 - val_loss: 1.1247 - learning_rate: 2.5000e-06
Epoch 16/60
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - AUC: 0.6585 - Precision: 0.6064 - Recall: 0.6507 - accura
cy: 0.6215 - loss: 1.0945 - val_AUC: 0.7031 - val_Precision: 0.6875 - val_Recall: 0.6471 - val_accu
racy: 0.7105 - val_loss: 1.0898 - learning_rate: 1.2500e-06
Epoch 17/60
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - AUC: 0.6881 - Precision: 0.5954 - Recall: 0.6704 - accura
cy: 0.6088 - loss: 1.0875 - val_AUC: 0.7241 - val_Precision: 0.6000 - val_Recall: 0.5294 - val_accu
racy: 0.6316 - val_loss: 1.0684 - learning_rate: 1.2500e-06
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
```

Multi-Class Classification Model

All 12 bird species included

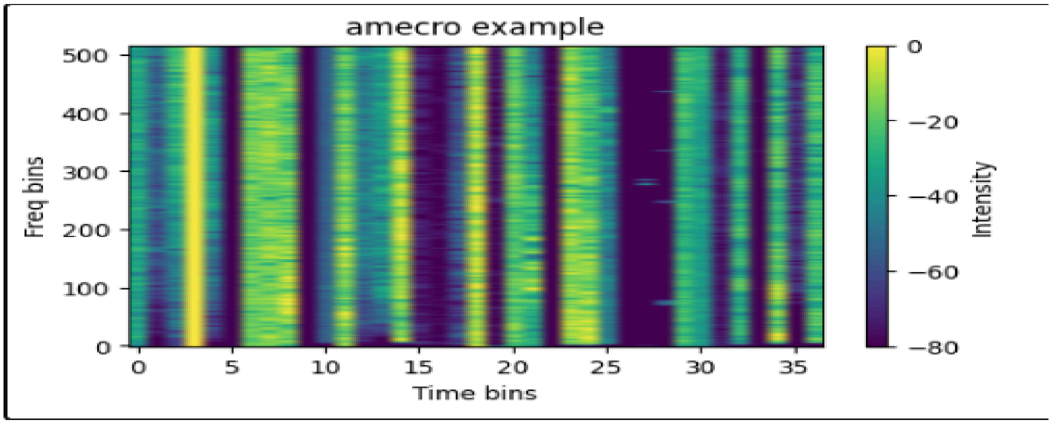
Configurations:

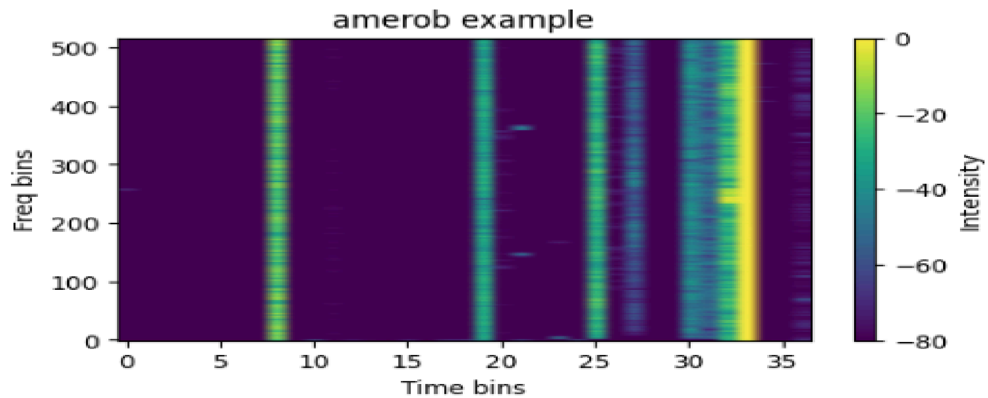
- **regularized_simple**:
  - Conv Layers: 2
  - Dense Units: 128
  - L2 Regularization: 0.001
  - Optimizer: Adam (LR = 1e-5)
- **deeper_regularized**:
  - Conv Layers: 4
  - Dense Units: 256
  - L2 Regularization: 0.002
  - Optimizer: RMSProp (LR = 1e-5)

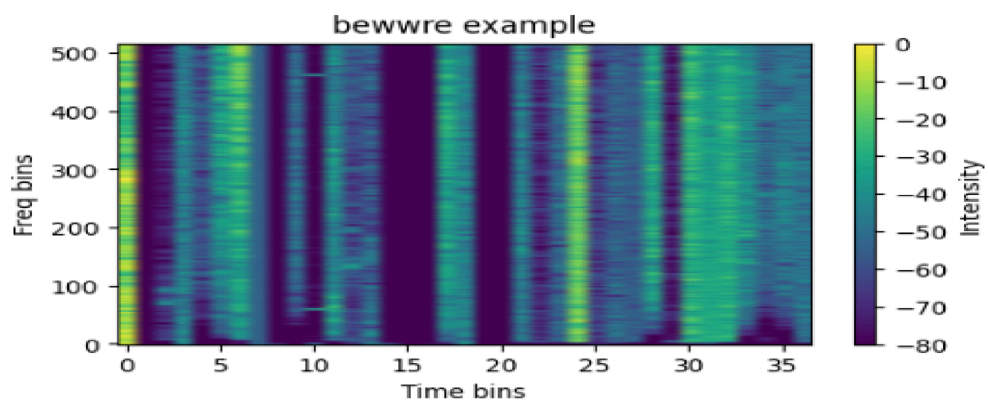
Training Setup:

- Epochs: 120
- Batch Size: 32
- EarlyStopping (patience=10)

- ReduceLROnPlateau (factor=0.5)







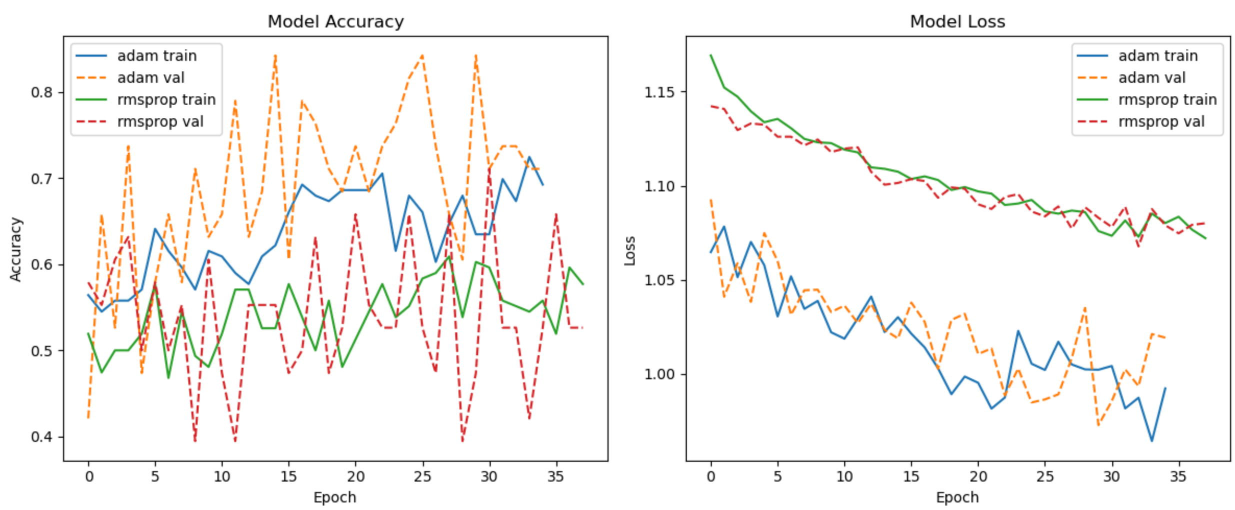
RESULTS AND COMPARISON

Table 1: Final Binary Classification Metrics

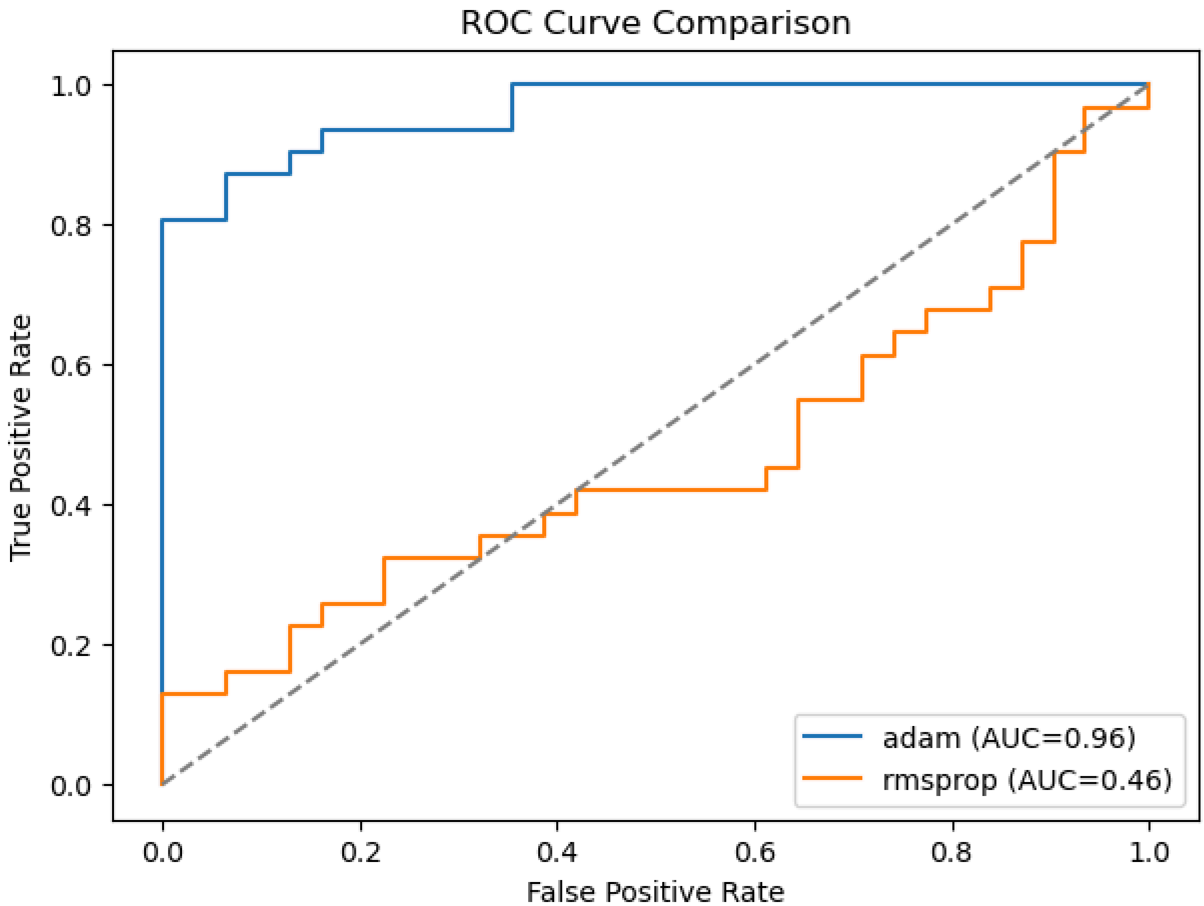
Model performs better with Adam optimizer with accuracy of 60%.

OPTIMIZER	ACCURACY	PRECISION	RECALL	F1 SCORE	AUC ROC	SPECIFICITY	TRAINING TIME (MIN)
Adam	**60**	**56**	**90.32**	**69.14**	**59.21**	29.03	50
RMS Prop	31	28.57	25.81	27.12	28.10	**35.48**	**30**

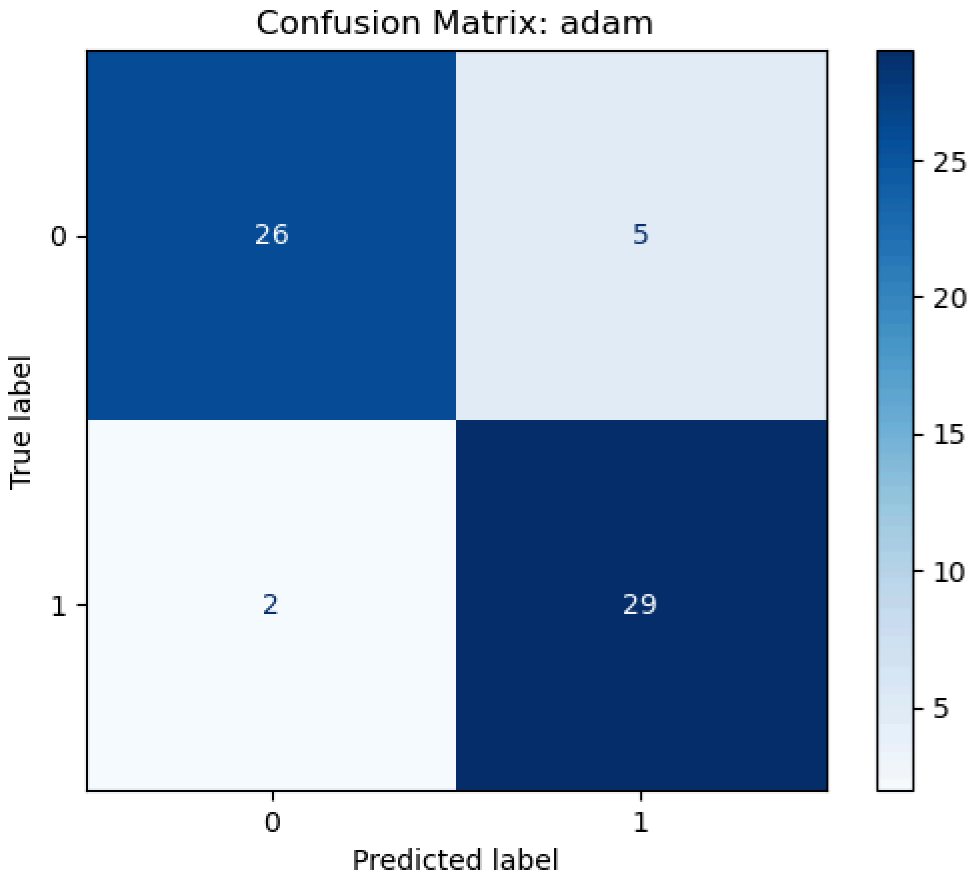
These below are plots for Model's Accuracy and Loss for Binary Classification of Data:



This ROC curve shows that the model trained with Adam performed much better (AUC = 0.96). It is compared to the model with RMSprop (AUC = 0.46).



Below is the confusion metrics which tells that the Adam model correctly predicted most class 1 samples and made fewer mistakes overall.
The Adam model shows high accuracy with few errors.
The RMSprop model struggled with class 1, misclassifying most of them.
Adam performs better.



Confusion Matrix: adam
True label
Predicted label
0
1
0
1
26
5
2
29
25
20
15
10
5

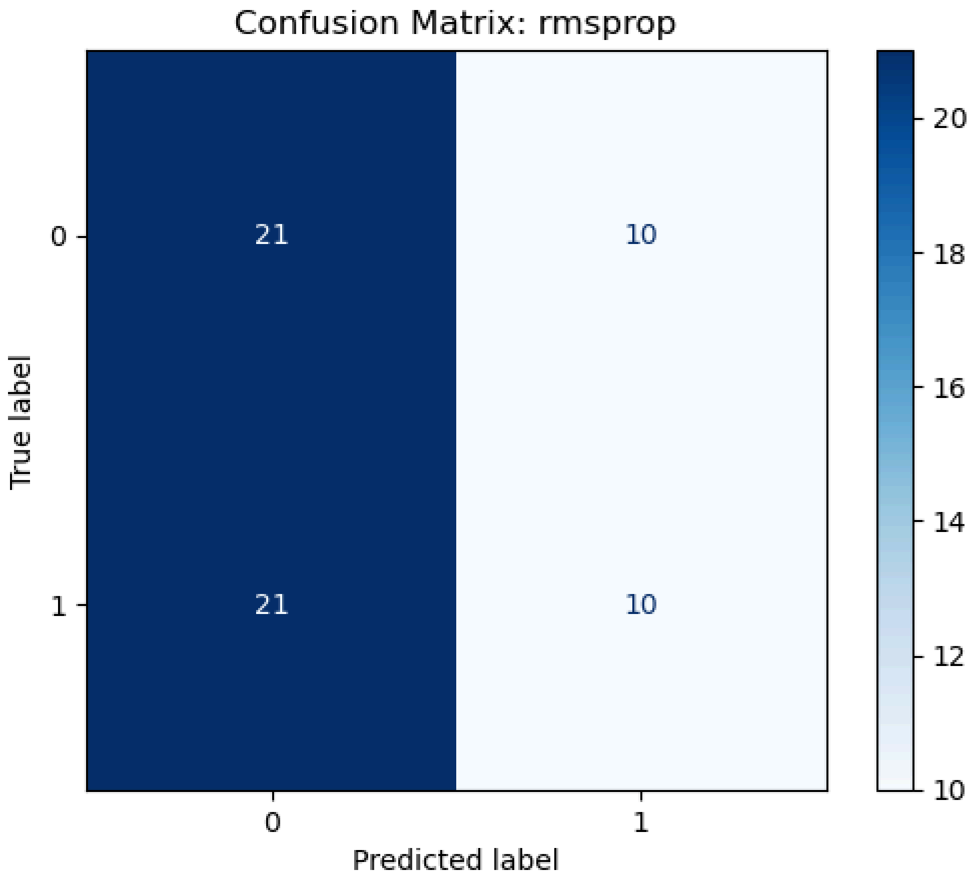
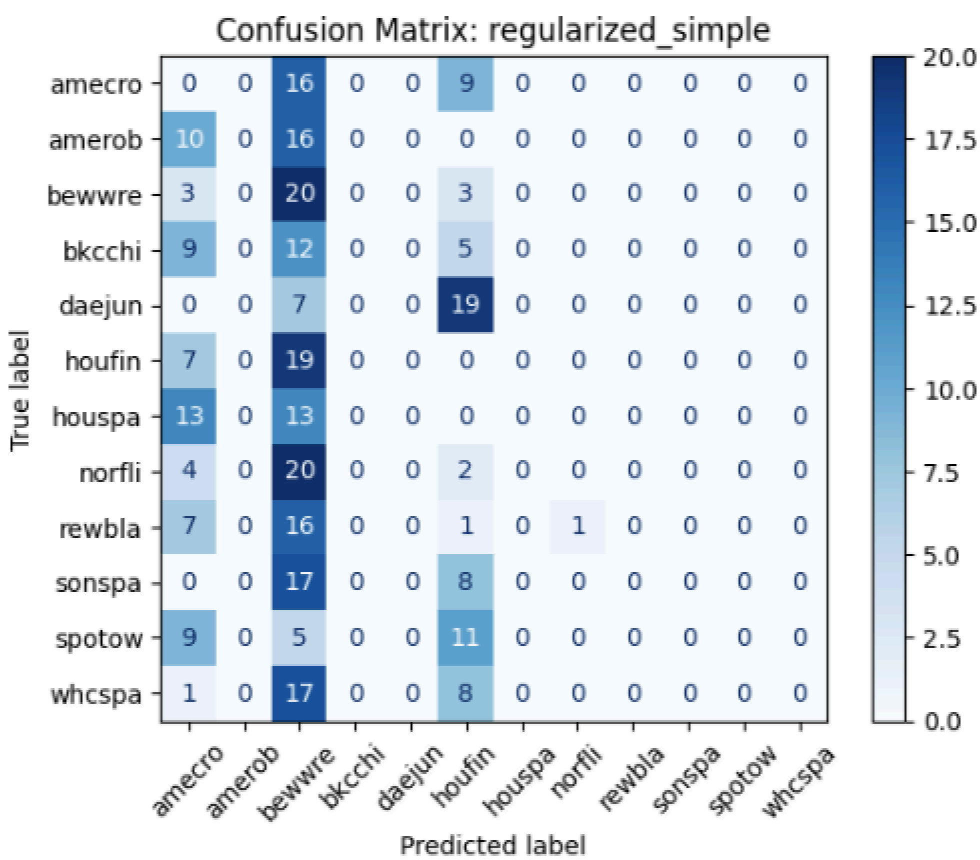


Table 2: Final Multiclass Classification Metrics

Deeper Regularized which contains the RMS Prop as optimizer is the best for muti-class classification as it is dominating the evaluation metrics with accuracy of 32.47 %.
In comparison to simple Adam optimizer with lower accuracy of 6.5% and other metrics.

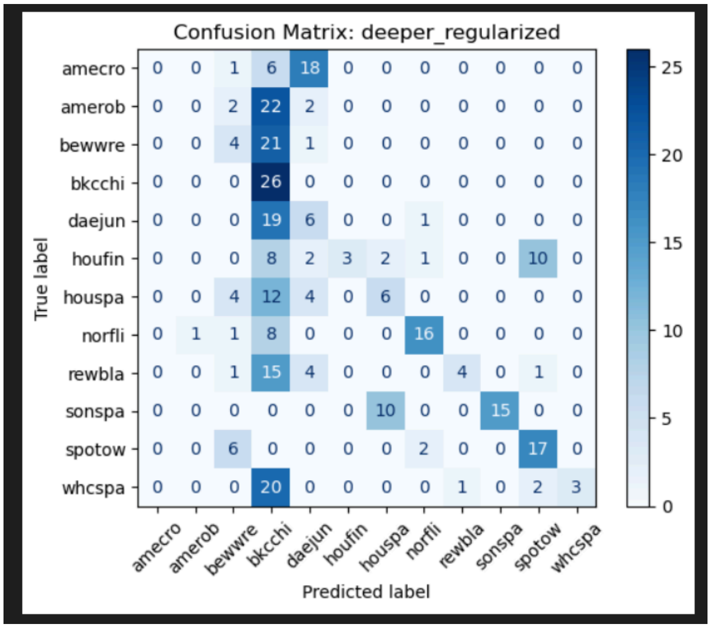
OPTIMIZER	ACCURACY	PRECISION	RECALL	F1 SCORE	AUC ROC	TRAINING TIME (MIN)
Deeper Regularized (RMSProp)	**32.47**	**50.31**	**32.51**	**30.58**	**86**	124.73
Simple Regularized (Adam)	6.5	0.56	6.4	1.03	47.11	**60.08**

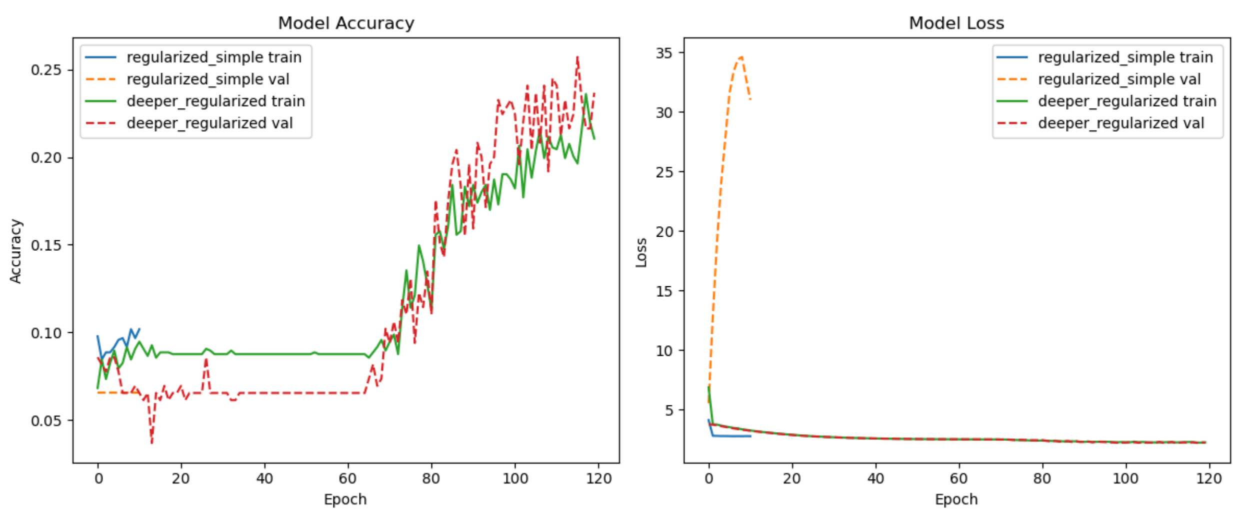
This confusion matrix shows that the model struggles to distinguish between many bird species.
It is misclassifying several into bewwre and houfin.
Secies like bewwre and daejun are predicted well, others like amecro, bkcchi, and whcspa are often confused.



```
Epoch 116/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 88ms/step - accuracy: 0.1980 - auc: 0.6865 - loss: 2.2865 - precision: 0.5489 - recall: 0.0210 - val_accuracy: 0.2571 - val_auc: 0.7094 - val_loss: 2.22
Epoch 117/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.2301 - auc: 0.6863 - loss: 2.2457 - precision: 0.6316 - recall: 0.0224 - val_accuracy: 0.2327 - val_auc: 0.6738 - val_loss: 2.25
Epoch 118/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.2488 - auc: 0.7046 - loss: 2.2181 - precision: 0.7114 - recall: 0.0336 - val_accuracy: 0.2163 - val_auc: 0.7040 - val_loss: 2.21
Epoch 119/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.2173 - auc: 0.7031 - loss: 2.2159 - precision: 0.7115 - recall: 0.0378 - val_accuracy: 0.2163 - val_auc: 0.6912 - val_loss: 2.25
Epoch 120/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.2147 - auc: 0.6999 - loss: 2.2350 - precision: 0.5789 - recall: 0.0232 - val_accuracy: 0.2367 - val_auc: 0.7146 - val_loss: 2.22
 8/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
2025-05-21 16:03:30.260744: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:961] PluggableGraphOptimizer failed: INVALID_ARGUMENT: Failed to deserialize the `graph_buf`.
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step
```

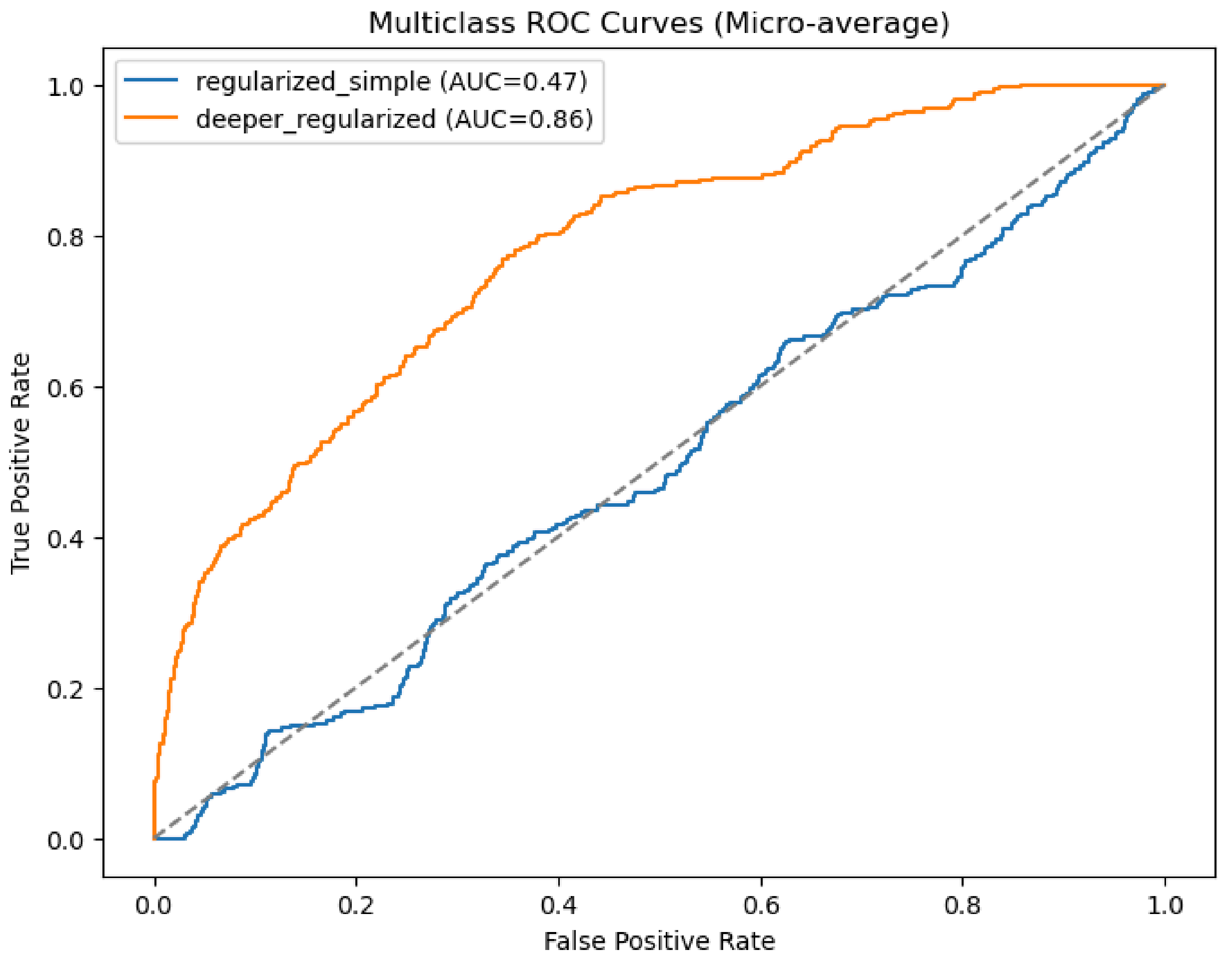
Confusion Matrix: The **deeper_regularized** model improves classification for several species like bewwre, daejun, norfil showing more accurate predictions compared to the previous model.





deeper_regularized model shows a general upward trend in validation accuracy over epochs, outperforming the simpler model.

For Loss Plot, both models show decreasing loss.

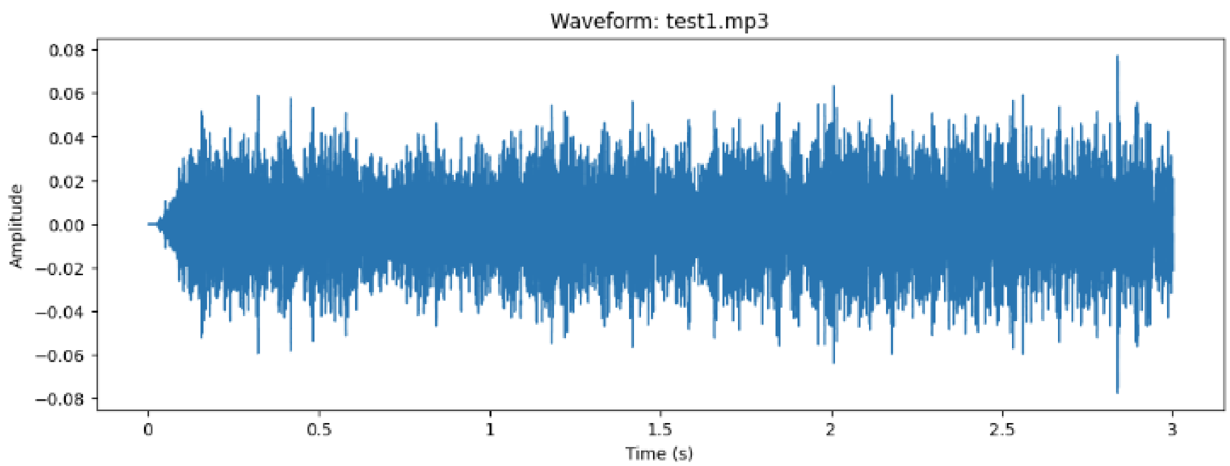


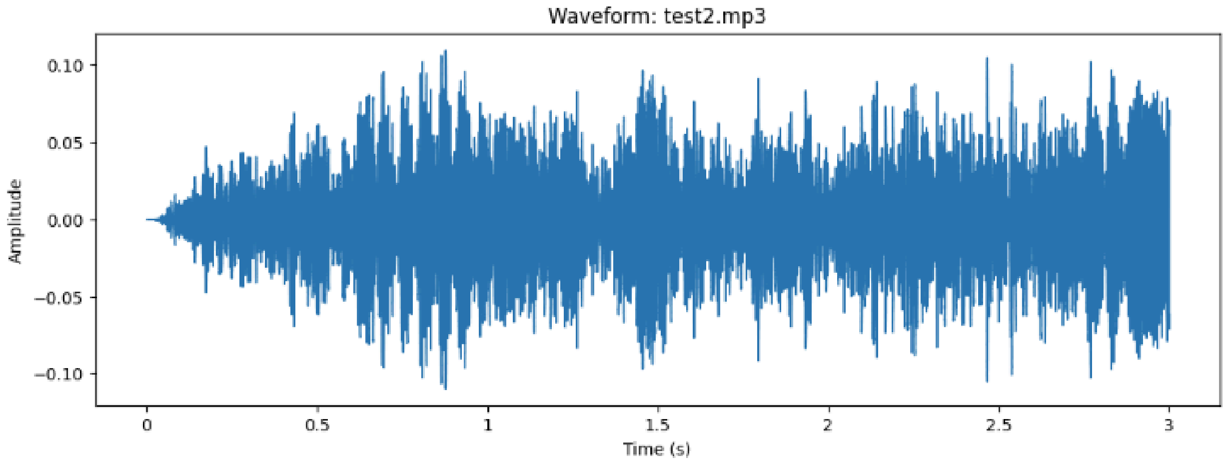
ROC curve shows that the deeper_regularized model performs significantly better than the regularized_simple model.

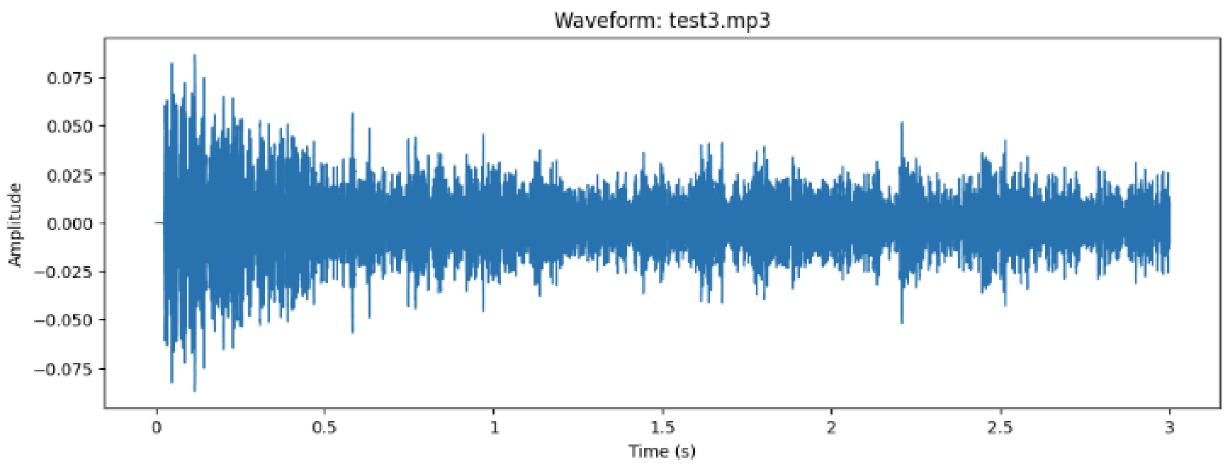
Deeper model clearly has stronger multiclass classification ability.

Table 3: External Test Data Top-3 Predictions

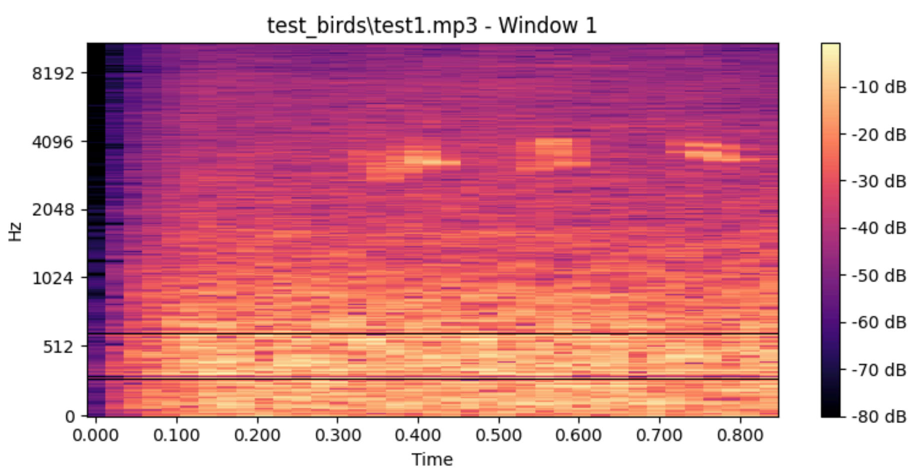
TEST CLIP	TOP 1 SPECIES	TOP 1 PROBABILITY	TOP 2 SPECIES	TOP 2 PROBABILITY	TOP 3 SPECIES	TOP 3 PROBABILITY
Test 1	Northern Flicker	34.81	Dark-eyed Junco	23.95	Black-capped Chickadee	17.91
Test 2	Northern Flicker	33.06	Dark-eyed Junco	23.74	Black-capped Chickadee	18.67
Test 3	Northern Flicker	32.17	Dark-eyed Junco	23.75	Black-capped Chickadee	19.46

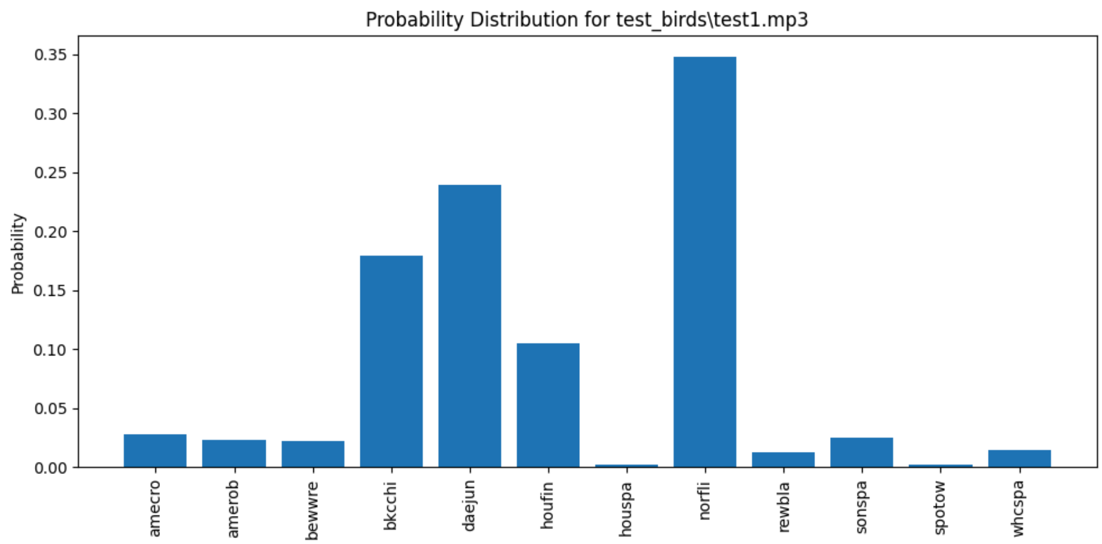


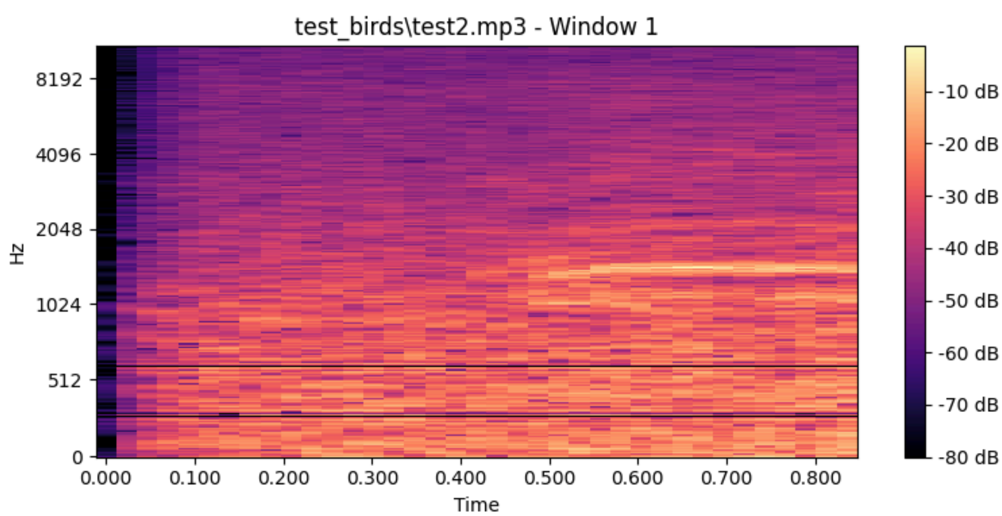


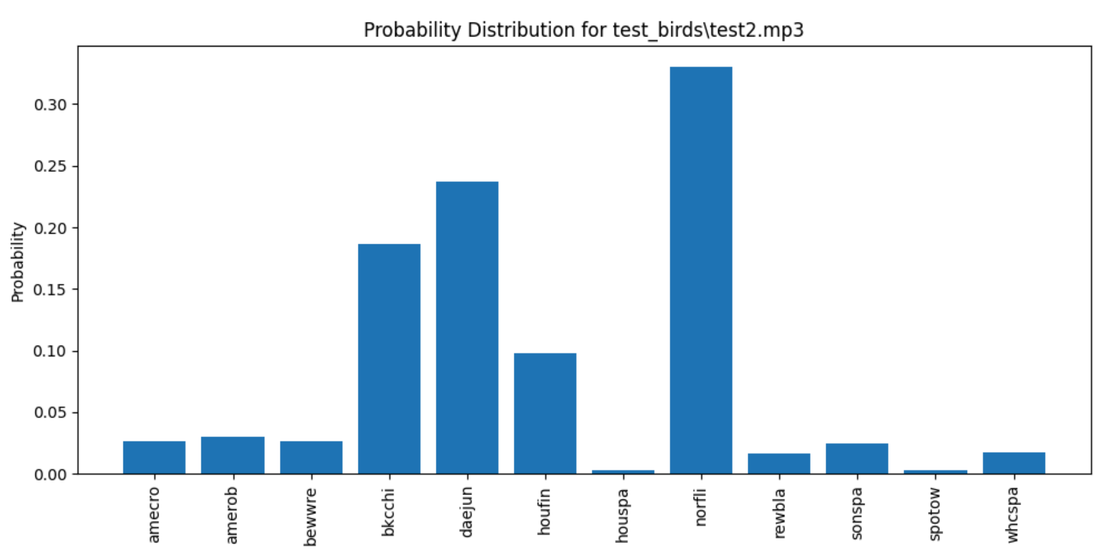


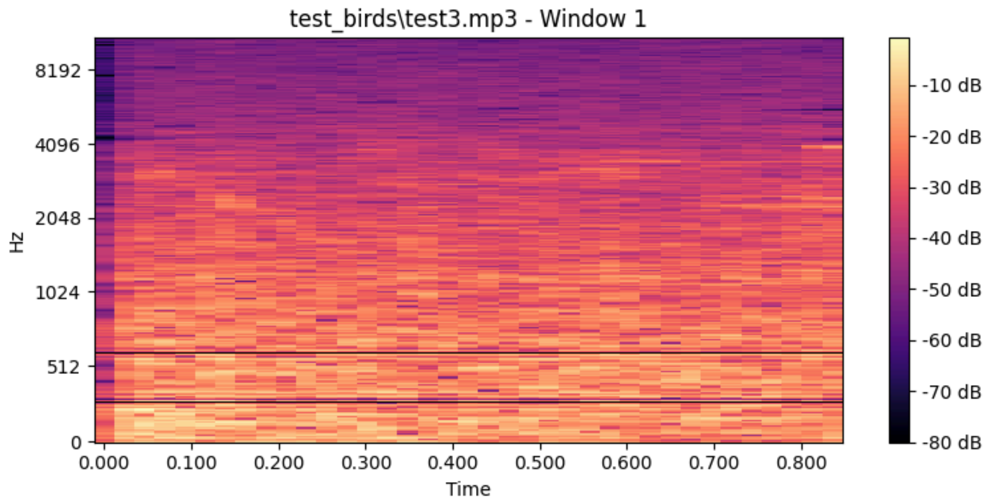
Now will be pasting results from test data fro predictions:

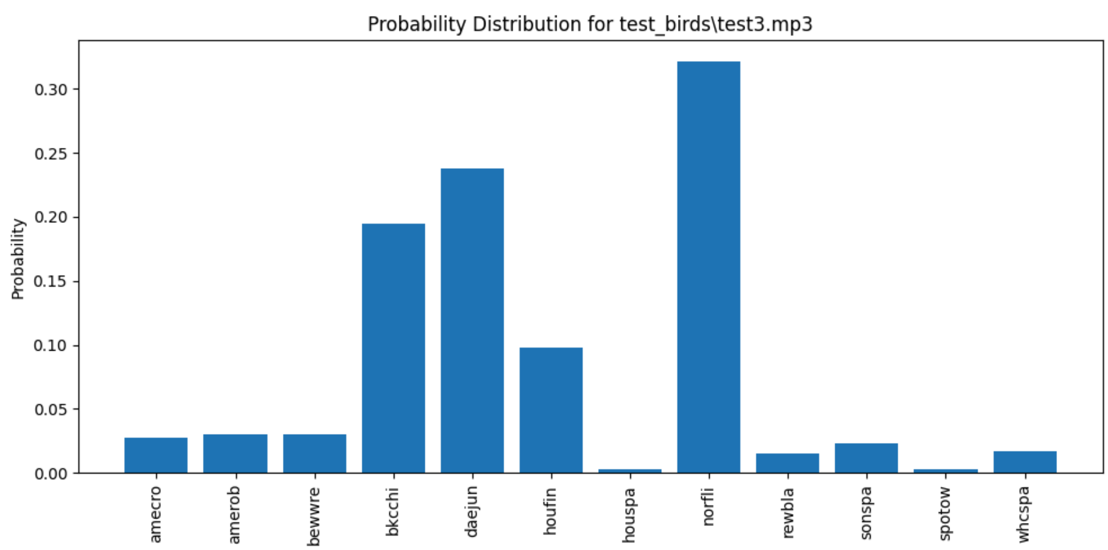



Probability Distribution for test_birds\test1.mp3
Probability
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
amecro
amerob
bewwre
bkcchi
daejun
houfin
houspa
norfli
rewbla
sonspa
spotow
whcspa


test_birds\test2.mp3 - Window 1
Hz
8192
4096
2048
1024
512
0
Time
0.000
0.100
0.200
0.300
0.400
0.500
0.600
0.700
0.800
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


Probability Distribution for test_birds\test2.mp3
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
amecro
amerob
bewwre
bkcchi
daejun
houfin
houspa
norfli
rewbla
sonspa
spotow
whcspa





- As you can see norfli bird is dominating clearly for all the test 1 and test 2 and test 3 data.
- It has the highest probability overall with 34, 33 and 32 % probability respectively for each test data.
- Followed by deajun and bkchi.
- Refer the comparison table in result section.

VISUAL INSIGHTS

Spectrogram & Waveform Analysis:

- Show time-frequency patterns of bird vocalizations.
- Variation among species is visible in frequency range and duration.

- As you can see norfli bird is dominating clearly

DISCUSSION

i. Training Time

- Binary: Adam trained faster and more effectively than RMSprop.
- Binary: Adam outperformed RMSprop due to better generalization
- Class weights and augmentation helped balance learning.
- Multi-class: deeper_regularized took significantly longer (~33 min).

ii. Challenges

- Initial overfitting (100% accuracy) resolved with regularization.
- Class imbalance (e.g., houspa dominating)
- Limited compute resources (no GPU) increased training time

iii. Observations

- Results vary across runs (despite seed) due to training dynamics.
- EarlyStopping helped reduce unnecessary epochs and saved time.

iv. Why CNN?

- Spectrograms are visual representations well-suited for CNN feature extraction.
- It can easily handled spectrogram in comparison to the other non-robust models.
- Helps in handling complex data.

v. Alternative Approaches:

- Use RNNs (e.g., LSTM, GRU) for temporal sequences
- Pretrained models like ResNet for better feature learning
- Lastly use of Transformers can be done which will ease the process as it retains/ remembers the pattern between the data.

vi. Limitations:

- In binary, there was limited data points/ sizes of data which at start affected showing a proper accuracy of 100%.
- Mixture of one or More species in an audio clip sometimes falsely predict the correct accuracy.
- External sound or background noise if any can alter the prediction or less the probability of detection.

CONCLUSION:

- This work demonstrates the feasibility of applying CNNs to classify bird species from spectrogram data.
- With appropriate tuning and preprocessing, even simple CNNs can effectively learn acoustic patterns.
- The model could be deployed in real-world bird monitoring systems, aiding ornithologists and conservationists.

FUTURE SCOPE:

- Expand and clean dataset
- Use transfer learning with pretrained models
- Use of Transformers to make it attentive and accurate at the same time.
- Explore RNNs to capture temporal data and its dependencies.
- Techniques to Handle environmental or external background noise. This will help to improve the robustness.

REFERENCES / CITATIONS:

[1] M. Abadi *et al*., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[2] F. Chollet *et al*., "Keras," 2015. [Online]. Available: https://keras.io/

[3] F. Pedregosa *et al*., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] B. McFee *et al*., "librosa: Audio and music signal analysis in Python," in *Proc. 14th Python in Science Conf. (SciPy 2015)*, 2015, pp. 18–25. [Online]. Available: https://librosa.org/

[5] *Deep Learning 1*, Canvas Seattle University. [Online]. Available: https://seattleu.instructure.com/courses/1621163/files/72015417?module_item_id=18402069. [Accessed: 28/04/2025].

[6] *Deep Learning 2*, Canvas Seattle University. [Online]. Available: https://seattleu.instructure.com/courses/1621163/files/72038418?module_item_id=18402075. [Accessed: 05/05/2025].