BIRD CALL CLASSIFICATION - USING DEEP LEARNING

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. The primary objective is to train models capable of accurately identifying bird calls among 12 species.

- The dataset includes `.mp3` files, converted into spectrograms.
- External test data includes 3 `.mp3` clips for model testing.

Key Questions:

- Can CNNs distinguish Seattle bird species based on their vocal patterns?
- Which model architectures and hyperparameters lead to better classification performance?

Tools & Libraries Used:

- Python, TensorFlow, Keras
- NumPy, Scikit-learn, Matplotlib
- Librosa for preprocessing

Methods Applied:

- Spectrogram generation and normalization
- CNN architecture with regularization and dropout
- ImageDataGenerator for augmentation
- Model evaluation with accuracy, precision, recall, F1-score, and ROC AUC
- Class weighting to address imbalanced data

THEORETICAL BACKGROUND

Convolutional Neural Networks (CNNs)

- Designed for image-like data (e.g., spectrograms)
- Extract spatial and temporal features effectively

Regularization and Stability:

- **L2 Regularization**: Penalizes large weights to prevent overfitting
- **Batch Normalization**: Normalizes activations to stabilize training
- **Dropout**: Randomly disables neurons during training to improve generalization

Optimizers:

- **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

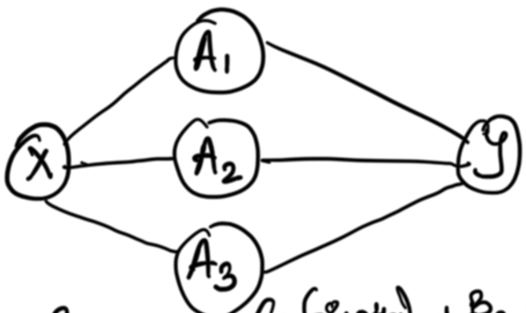
Neural Networks

- → 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 $\underbrace{\text{Activation Function}}$

  Any non-Linear functions

- → First linear transformation, then in Hidden Layer (non-linear transformation)

Eg



Let's dive in an example

$$\beta_0 + \beta_1 \left(g\frac{(-8+0.4x)}{A_1}\right) + \beta_2 \left(g\frac{(1+0.25x)}{A_2}\right) + \beta_3 \left(g\frac{(-28+0.6x)}{A_3}\right)$$

$$f(x) = 1.25 + 0.5(\quad) - 1(\quad) - 0.75(\quad)$$

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:

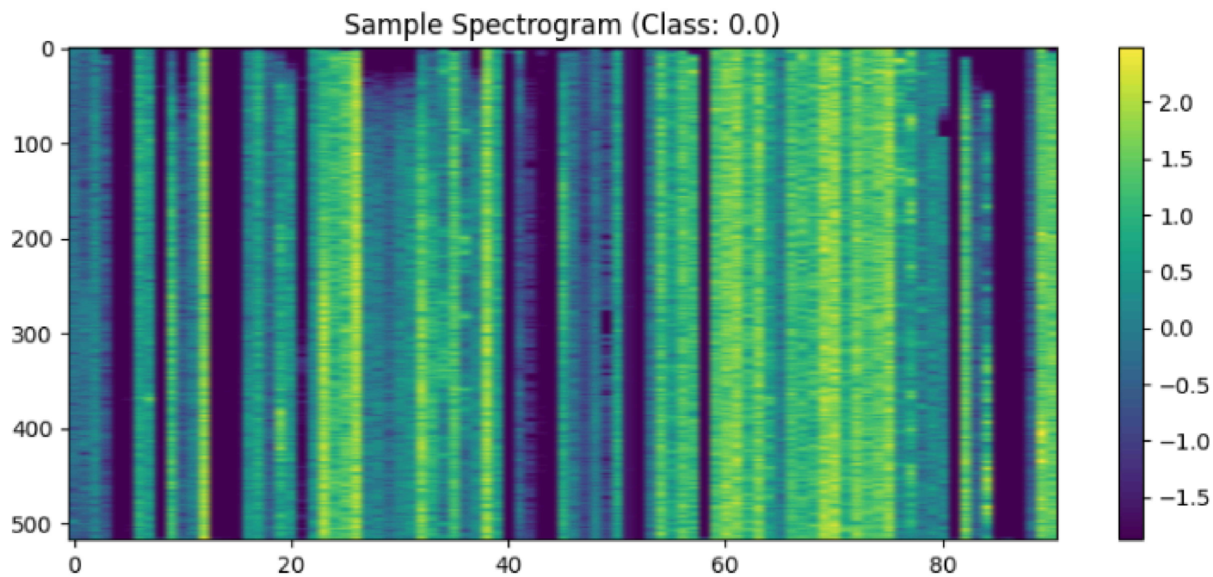
- Spectrograms read from `.hdf5` format
- Cropped to consistent time length and normalized
- Data split: 76% training / 24% testing
- Augmentation: `ImageDataGenerator` with `width_shift=0.1`, `height_shift=0.1`, `validation_split=0.2`

Binary Classification Model

Bird Species:

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

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



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

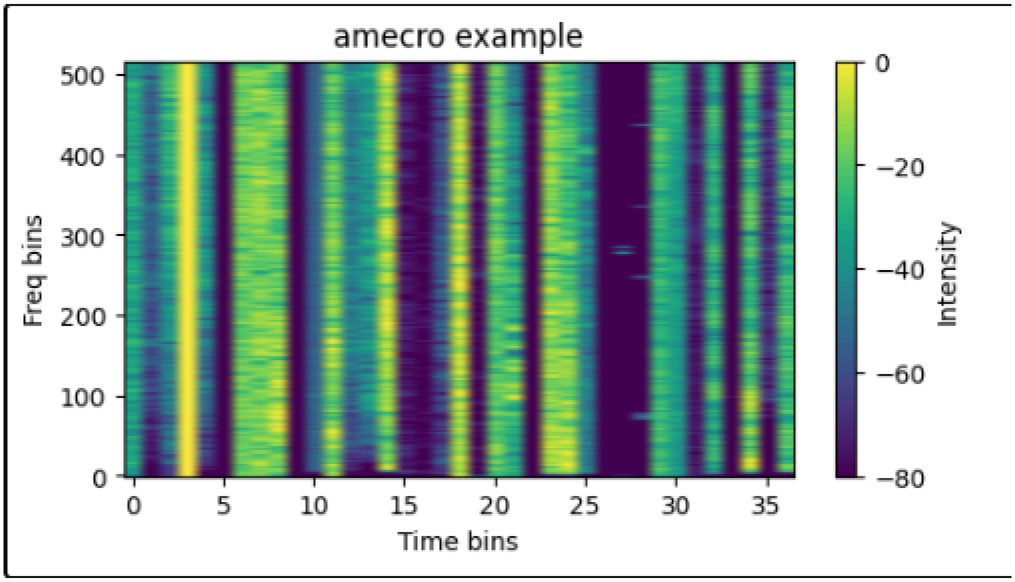
All 12 bird species included

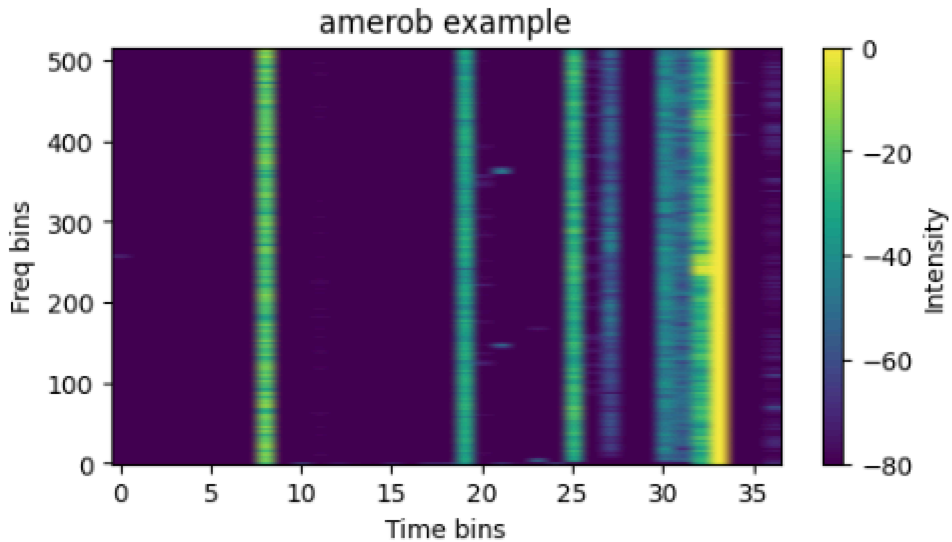
Configurations:

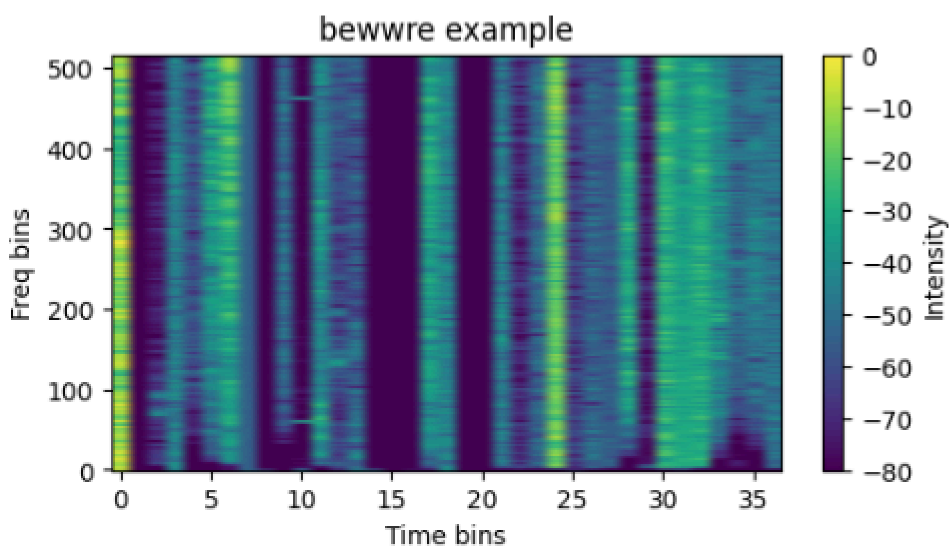
- **regularized_simple**:
  - Conv Layers: 2
  - Dense Units: 128
  - L2 Regularization: 0.001
  - Optimizer: Adam (LR = 1e-5)
- **deeper_regularized**:
  - Conv Layers: 4
  - Dense Units: 256
  - L2 Regularization: 0.002
  - Optimizer: RMSProp (LR = 1e-5)

Training Setup:

- Epochs: 120
- Batch Size: 32
- EarlyStopping (patience=10)
- ReduceLROnPlateau (factor=0.5)


amecro example
Freq bins
Time bins
Intensity

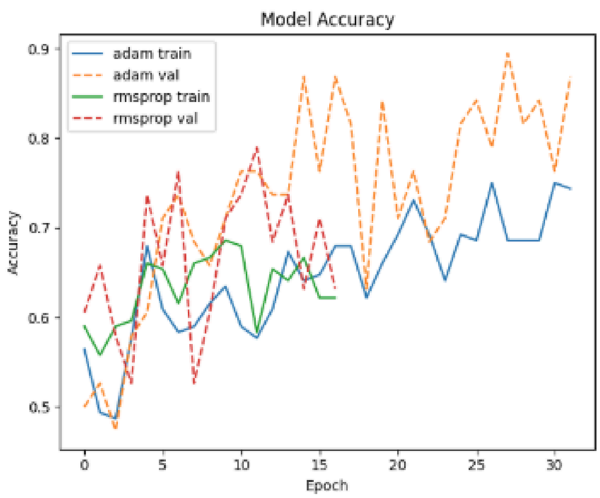

amerob example
Freq bins
Time bins
Intensity

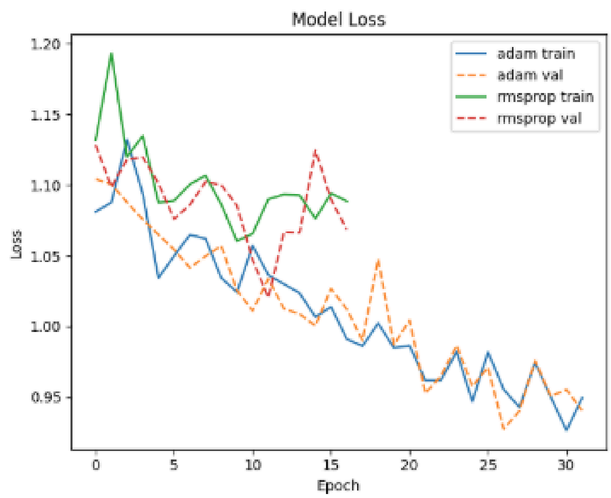

bewwre example
Freq bins
Time bins
Intensity

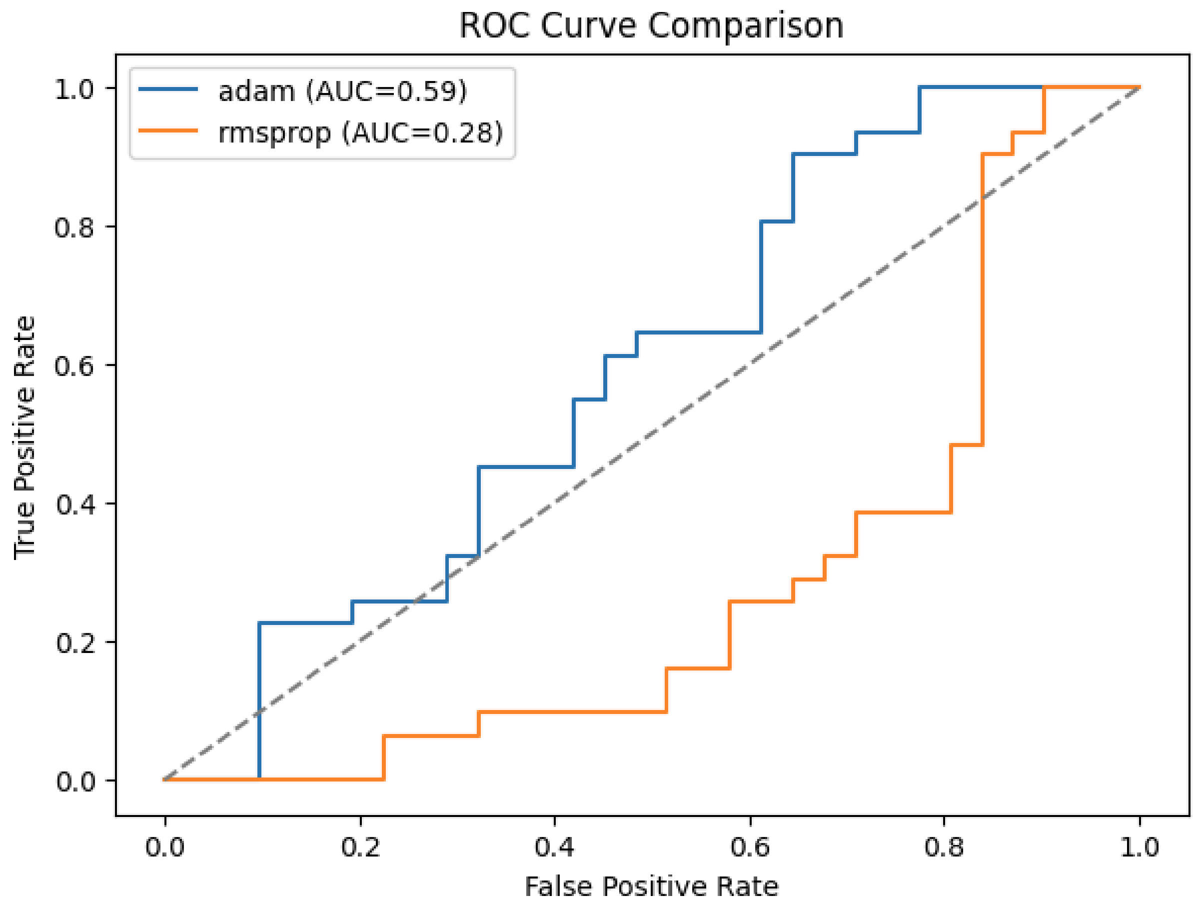
RESULTS AND COMPARISON

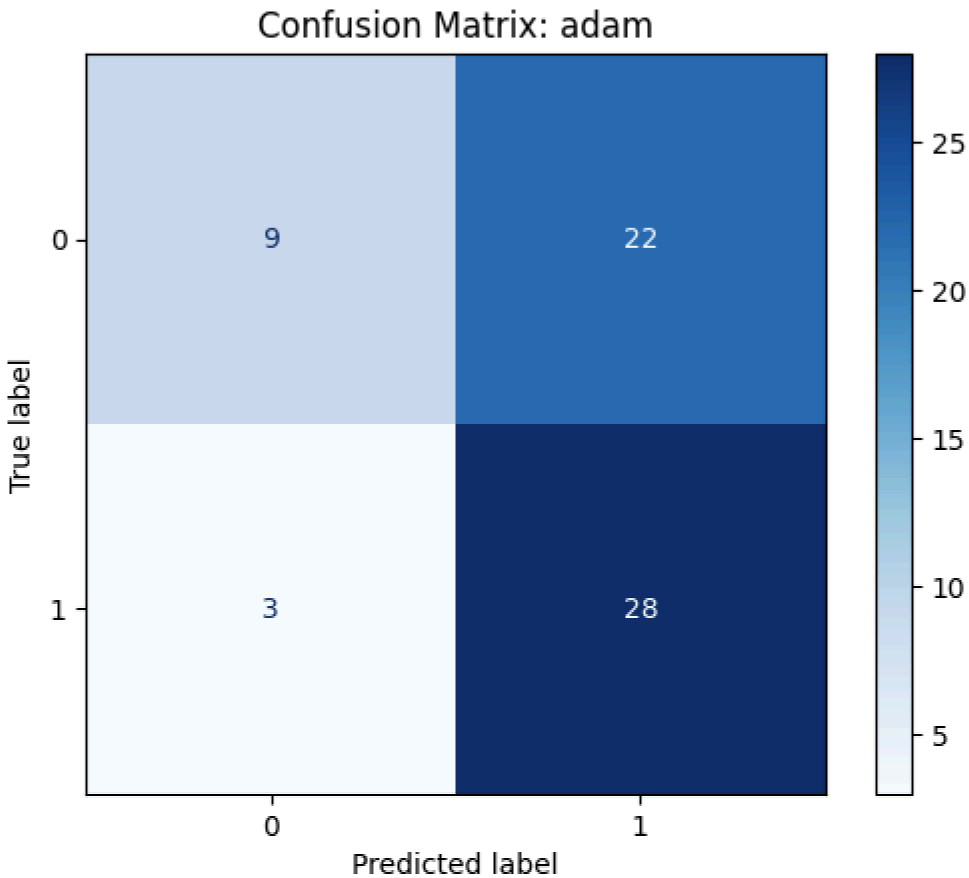
Table 1: Final Binary Classification Metrics

Config	Accuracy	Precision	Recall	F1 Score	ROC AUC	Specificity
adam	0.5968	0.5600	0.9032	0.6914	**0.5921**	0.2903
rmsprop	0.3065	0.2857	0.2581	0.2712	0.2810	**0.3548**






ROC Curve Comparison
adam (AUC=0.59)
rmsprop (AUC=0.28)
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate



Confusion Matrix: adam
True label
Predicted label
0
1
0
1
9
22
3
28
25
20
15
10
5

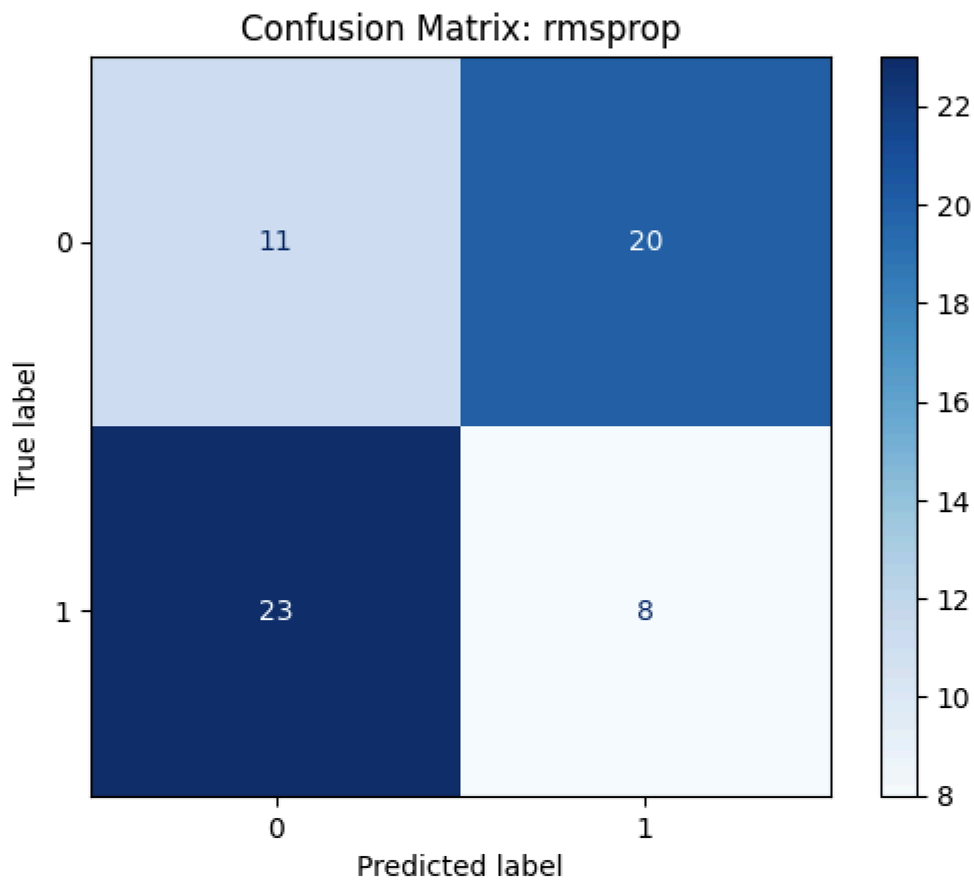
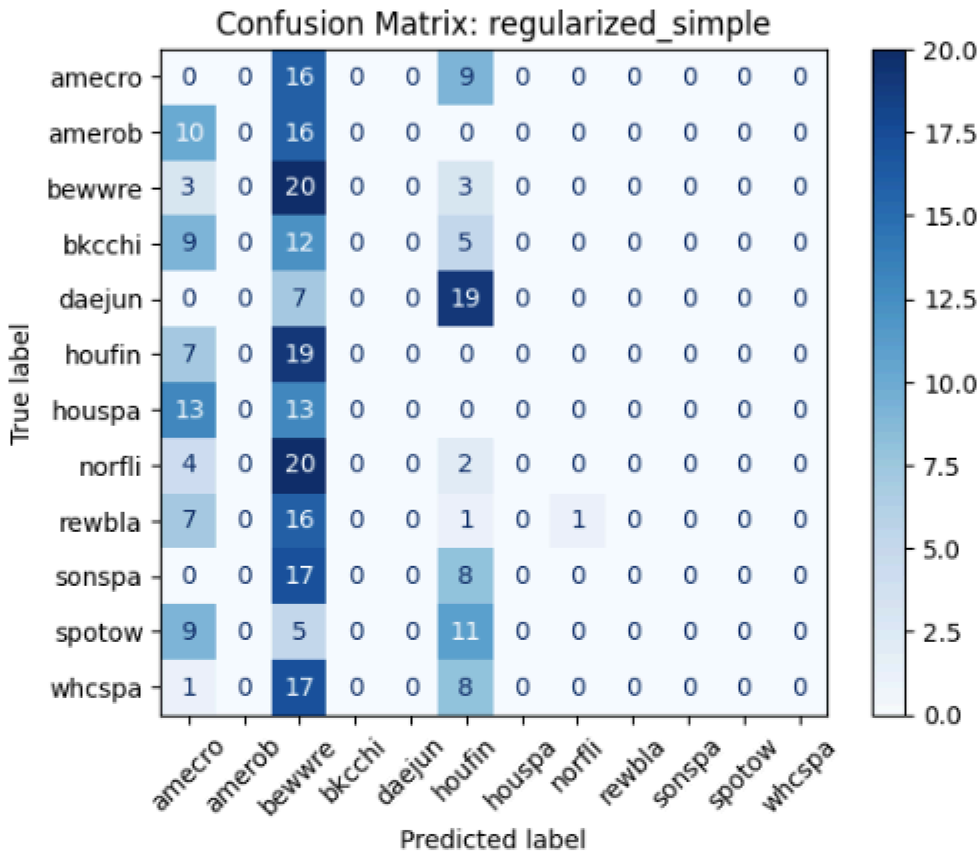
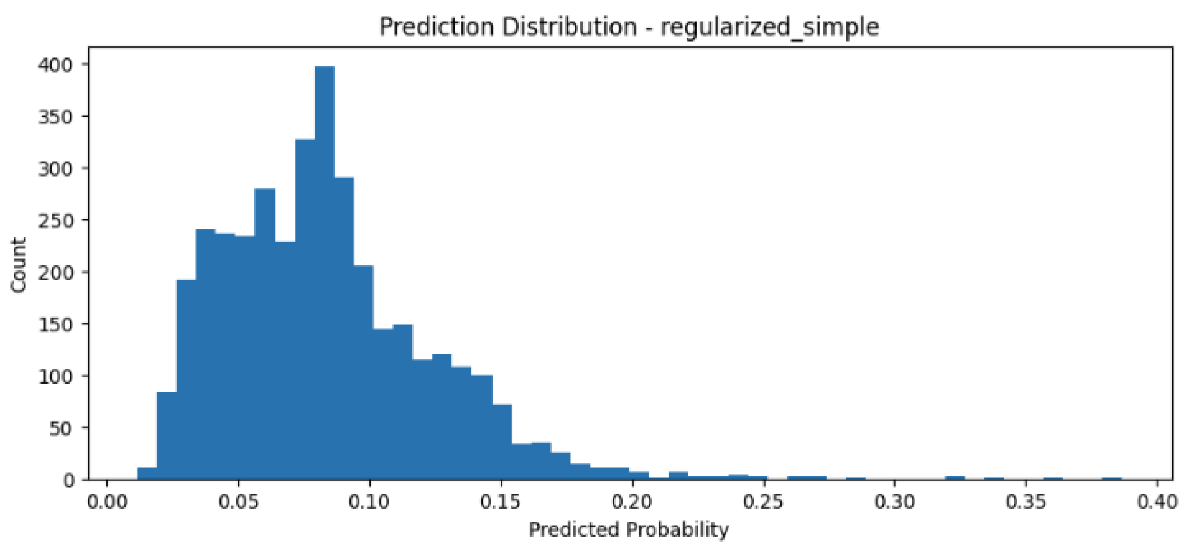


Table 2: Final Multiclass Classification Metrics

Config	Accuracy	Precision (Macro)	Recall (Macro)	F1 Score (Macro)	ROC AUC (Macro)	Training Time (min)
deeper_regularized	**0.1039**	**0.1548**	**0.1050**	**0.0797**	**0.5044**	124.73
regularized_simple	0.0649	0.0094	0.0641	0.0163	0.4071	**60.08**



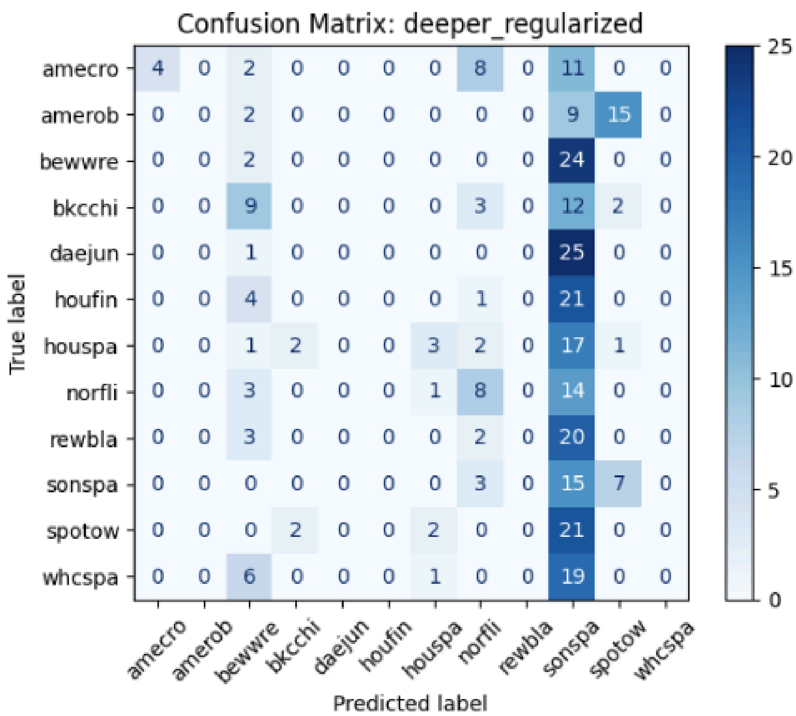
Confusion Matrix: regularized_simple
True label
Predicted label
amecro
amerob
bewwre
bkcchi
daejun
houfin
houspa
norfli
rewbla
sonspa
spotow
whcspa

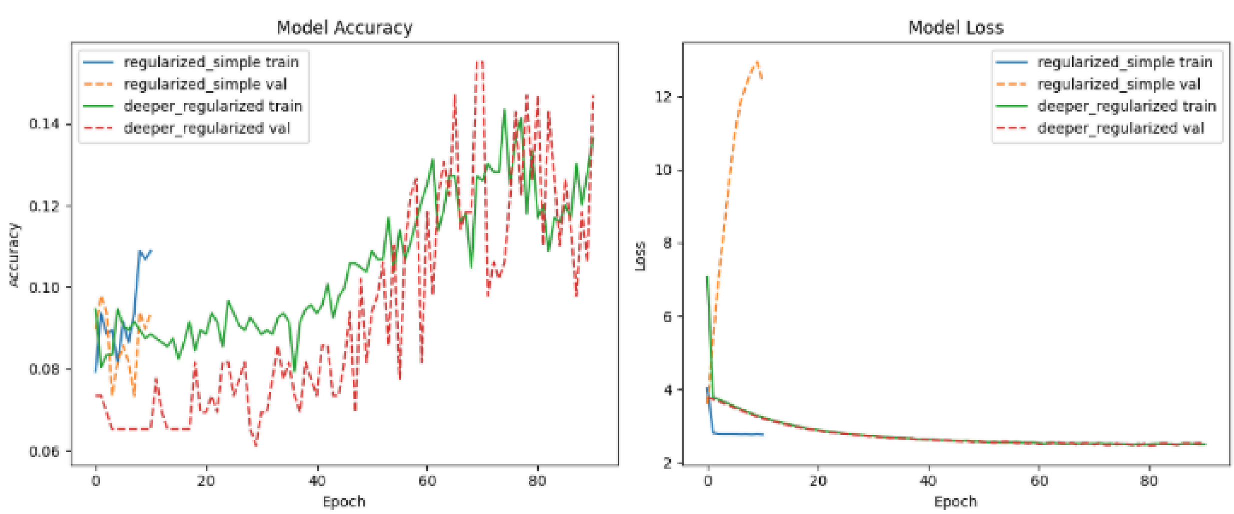


Prediction Distribution - regularized_simple
Count
Predicted Probability

```
Epoch 89/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 16s 506ms/step - accuracy: 0.1295 - auc: 0.5611 - loss: 2.5029 - precision: 0.8357 -
recall: 0.0174 - val_accuracy: 0.1184 - val_auc: 0.5728 - val_loss: 2.5163 - val_precision: 1.0000 - val_recall:
0.0122 - learning_rate: 1.0000e-06
Epoch 90/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 16s 523ms/step - accuracy: 0.1238 - auc: 0.5690 - loss: 2.4903 - precision: 0.7469 -
recall: 0.0197 - val_accuracy: 0.1061 - val_auc: 0.5943 - val_loss: 2.5275 - val_precision: 1.0000 - val_recall:
0.0082 - learning_rate: 1.0000e-06
Epoch 91/120
31/31 ━━━━━━━━━━━━━━━━━━━━ 20s 656ms/step - accuracy: 0.1436 - auc: 0.5806 - loss: 2.5009 - precision: 0.7296 -
recall: 0.0197 - val_accuracy: 0.1469 - val_auc: 0.5464 - val_loss: 2.5062 - val_precision: 1.0000 - val_recall:
0.0163 - learning_rate: 1.0000e-06
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 114ms/step

Confusion matrix for deeper_regularized:
```





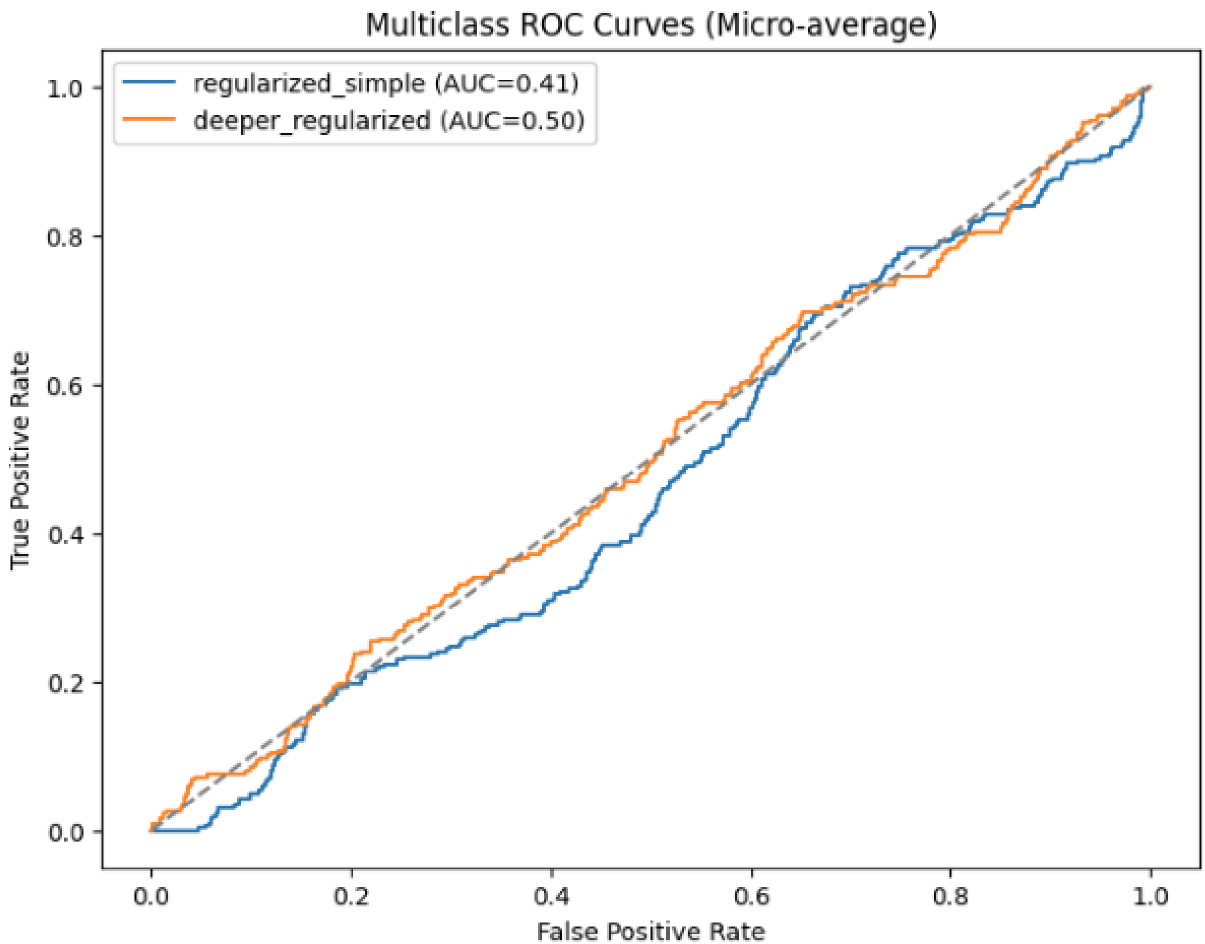
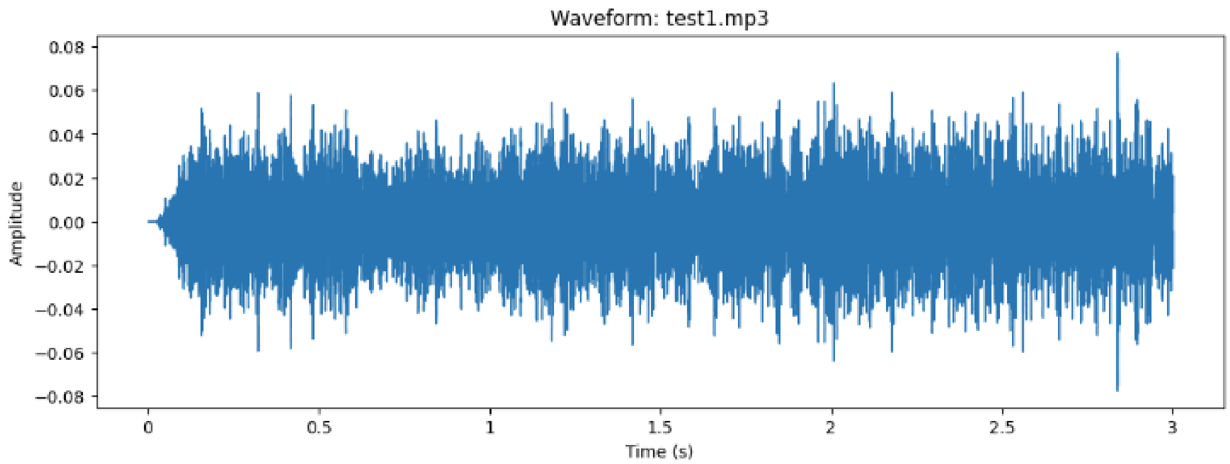
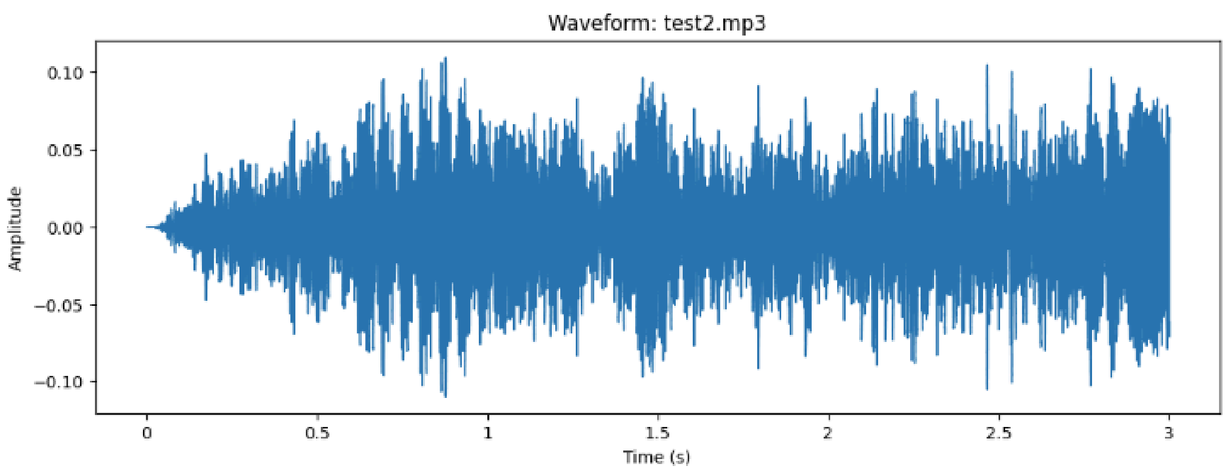


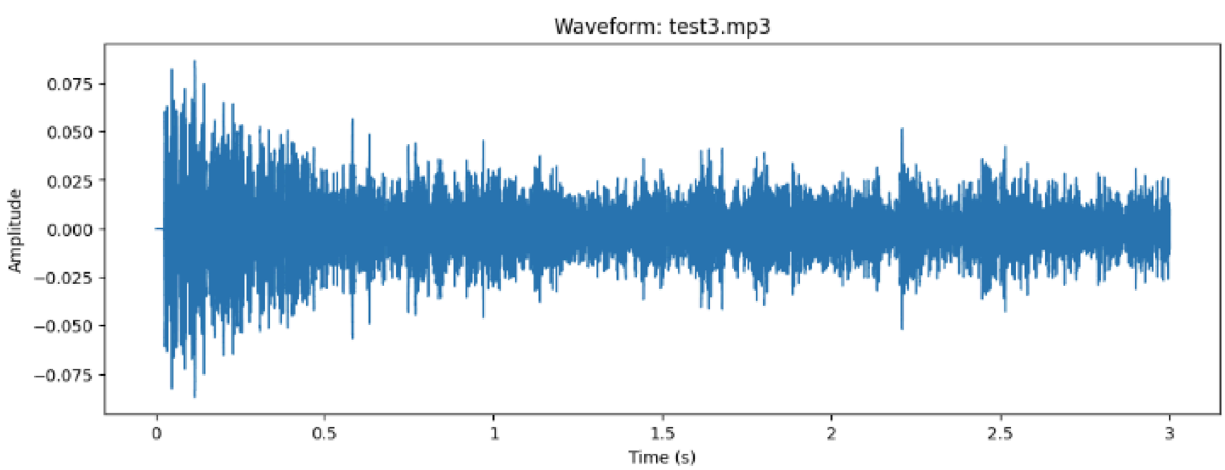
Table 3: External Test Data Top-3 Predictions

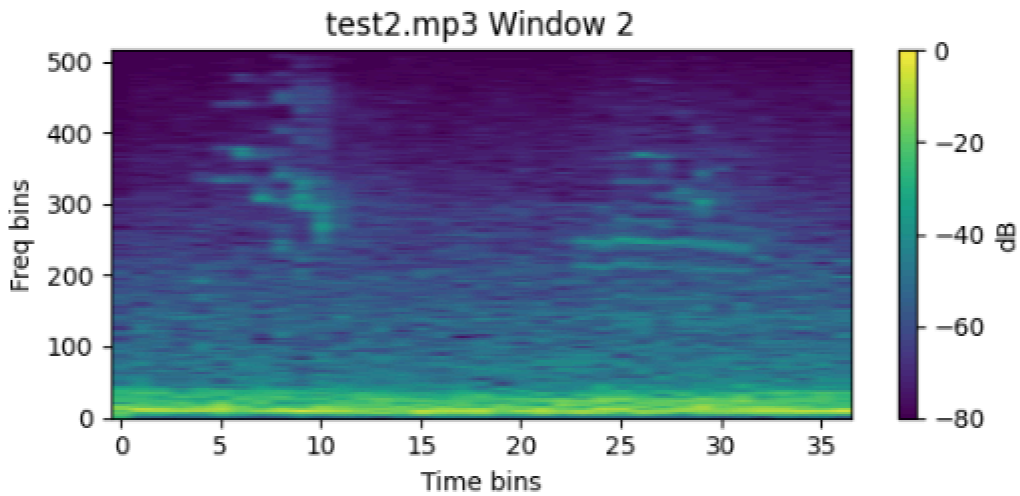
Clip	Top-1 Species	Top-1 Prob	Top-2 Species	Top-2 Prob	Top-3 Species	Top-3 Prob
test1.mp3	houspa	**0.4578**	bkcchi	0.3161	norfli	0.0773
test2.mp3	houspa	**0.2954**	bkcchi	0.2479	norfli	0.1041

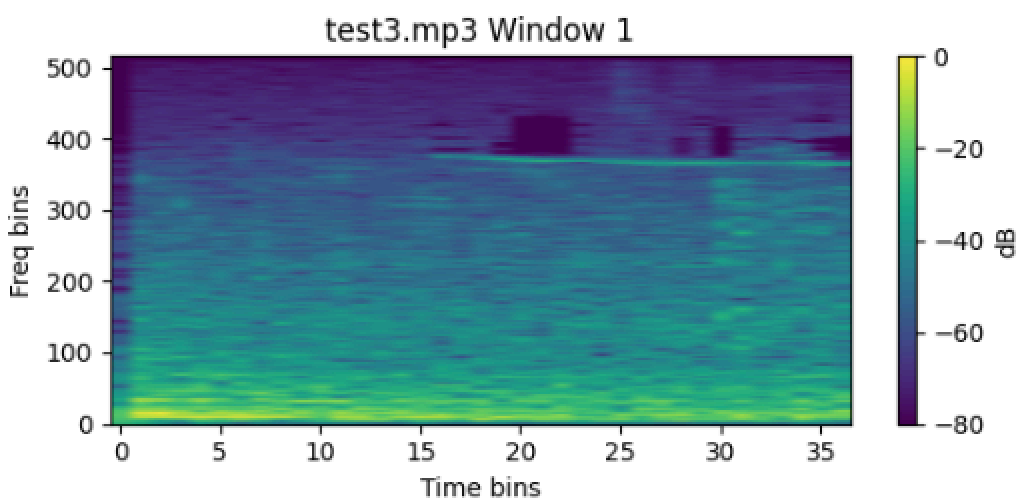
test3.mp3	houspa	**0.3678**	bkcchi	0.2828	norfli	0.0947











VISUAL INSIGHTS

Spectrogram & Waveform Analysis:

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

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)
- Similar-sounding birds (e.g., bewwre, bkcchi) caused confusion
- 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].