Bird Sound Classification Using Deep Learning

Abstract

Bird Sound Classification works by classifying bird species from audio recordings.In this process, we proccess raw bird sound recordings into spectograms and train convolutional neural networks (CNN) for both binary and multiclass classification tasks.The binary model achieves a 100% accuracy and the multiclass model achieves an accuracy of 71.9%; most of the misclassifications occurredwith Dark-eyed Junco and Spotted Towhee. These results show that CNN's are highly effective for bird sound classification with a fair share of errors in identifying birds.In future work we will foucs on improving the accuracy while making sure the model does not memorize data.Models being able to memorize the data is highly possble considering the size of the data.We can also work on expanding the model to give us a better accuracy with noisy data(background noises).

Introduction

Bird Sound Classifier built using neural networks can be a great tool for exploring, biodiversity monitoring, and research.For many years ornithologists and biologists have worked with visual observations and experience to identify bird species. This can sometimes be time consuming, will require expertise and is prone to errors/misclassifiction. The major issue faced by them is processing the huge amount of data they collect on any bord species manually and getting meaningful insights.So to tackle this problem we've built a scalable method that can classify bird species form their sound.

With recent advances in machine learning, deep learning it is now pssible to automate this entire process of identifying the bird with a great accuracy and efficiency. Convolutional neural networks can learn complex features from spectogram of bird calls which is a more efficient and accurate method compared to traditional methods. Even with all these advancements, challenges are not avoidable. This ranges form class imbalance, imroper training data and difficulty faced with birds that sound very similar to each other.

We aim to automate identifying bird calls by developing a pipepline to preprocess raw audio clips into spectograms which can be suitable for maching learning.We use

these spectograms to train and evaluate neural networks for both binary and multiclass classification tasks and analyze the results.

The dataset we use contains audio clips of 12 bird species.Each clip is subsampled to 22050Hz and first 3 secinds of the sound clip are selected and then a spectogram is produced fr each 2 second window resulting in a 256(frequency) x 343(time) image of the bird call.All bird calls for all clips in a given species are saved individually and number of samples for all birds is 128 each which is 1536 in total.

Theoretical Background

Neural networks are computational models inspired by the human brain, consisting of interconnected layers of nodes (“neurons”). Each neuron computes a weighted sum of its inputs, applies a nonlinear activation function, and passes the result to the next layer, allowing the network to learn complex, nonlinear relationships in data.

Common activation functions include sigmoid, tanh, ReLU, and softmax. The sigmoid function outputs values between 0 and 1, making it suitable for binary classification, while the softmax function transforms outputs into a probability distribution over multiple classes.

Overfitting occurs when a model learns noise or random fluctuations in the training data. Regularization techniques such as dropout layers, which randomly ignore a fraction of neurons during training, help improve generalization.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed to process grid-like data such as images and spectrograms. CNNs consist of several key components:

- Convolutional layers: Apply learnable filters to local regions of the input, extracting features such as edges or frequency contours.
- Pooling layers: Downsample feature maps, providing translation invariance and reducing dimensionality.
- Dense (fully connected) layers: Aggregate features for final classification.

CNNs learn hierarchical features: early layers detect simple patterns, while deeper layers combine these into more complex structures.

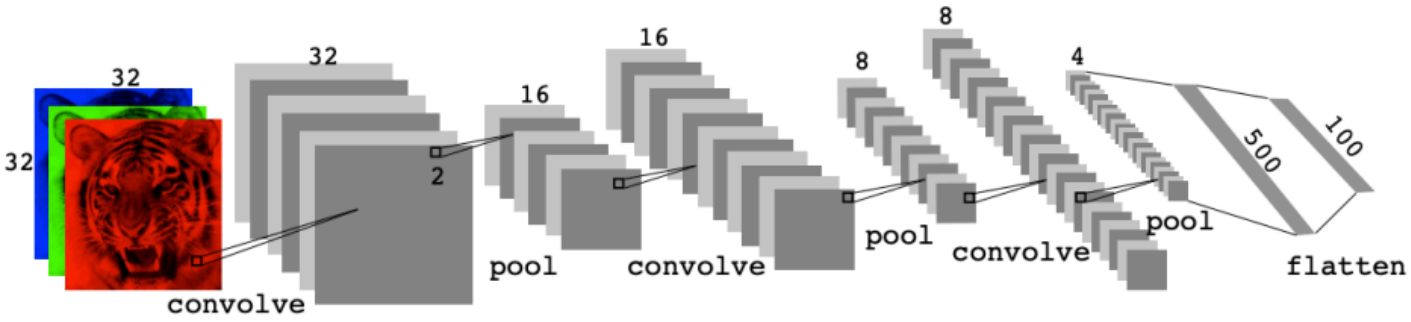

Fig 1- CNN Architecture

Spectrograms & Audio Preprocessing

A spectrogram is a time-frequency representation of an audio signal, created by applying a windowed Fast Fourier Transform (FFT) to short segments of the signal.

- Subsampling: Audio is resampled to 22050 Hz.
- Windowing: The first 3 seconds are selected
- Each 2-second window is transformed into a spectrogram (e.g., 256 frequency bins × 343 time steps).
- Resizing: Spectrograms are resized to 343×256 for model input.

Performance Metrics and Regularization

- Accuracy: Fraction of correct predictions.
- Mean Absolute Error (MAE): Average absolute difference between predicted and true values.
- Confusion Matrix: Visualizes true vs. predicted labels, highlighting which classes are confused.
- Precision, Recall, F1-score: Important for imbalanced datasets.
- Overfitting Prevention:
    - Dropout layers: Randomly ignore a fraction of neurons during training
    - Validation split: Hold out part of the data for model evaluation.
    - Early stopping: Halt training when validation loss stops improving.

Methodology

Data Processing

- Preprocessing Pipeline:
  - Each .mp3 audio file is loaded using librosa and resampled to 22050 Hz.
  - For each file, the first 3 seconds are extracted
  - Spectrograms are resized to 256 (frequency) × 343 (time) pixels.
  - All spectrograms are normalized to the range [0, 1].
  - The final dataset is balanced: exactly 128 spectrograms per species (12 species, 1536 total).
  - All spectrograms and labels are saved in HDF5 format (data/bird_spectrogram.hdf5).
- Label Encoding:
  - For multiclass classification, species labels are one-hot encoded (12-dimensional vectors).
  - For binary classification, labels are 0 or 1.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y_encoded = le.fit_transform(y)
print("Encoded species labels:", dict(zip(le.classes_, le.transform(le.classes_))))

```

```
Encoded species labels: {'amecro': 0, 'amerob': 1, 'bewwre': 2, 'bkcchi': 3, 'daejun': 4, 'houfin': 5, 'houspa': 6, 'norfli': 7, 'rewbla': 8, 'sonspa': 9, 'spotow': 10, 'whcspa': 11}
```

Fig 2- Label Encoding

- Data Splitting:
  - The dataset is split into training (70%), validation (15%), and test (15%) sets using stratified sampling to maintain class balance.

- Splits are stored in bird_spectrogram_splits.hdf5.

5. Split Data into Training, Validation, and Test Sets

```
from sklearn.model_selection import train_test_split

# First split off test set (15%)
X_temp, X_test, y_temp, y_test = train_test_split(
    X_norm, y_encoded, test_size=0.15, stratify=y_encoded, random_state=42
)
# Then split remaining into train and validation (validation = 15% of original)
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.1765, stratify=y_temp, random_state=42
)

print(f"Train: {X_train.shape}, Validation: {X_val.shape}, Test: {X_test.shape}")

```

```
Train: (1074, 343, 256), Validation: (231, 343, 256), Test: (231, 343, 256)
```

Fig 3- Data Splitting

Model Implementation

- Binary Classification Model:
  - Input: 256×343×1 spectrogram.
  - Architecture:
    - Conv2D (8 filters, 3×3, ReLU) → MaxPooling2D (2×2)
    - Conv2D (16 filters, 3×3, ReLU) → MaxPooling2D (2×2)
    - Conv2D (32 filters, 3×3, ReLU) → MaxPooling2D (2×2)
    - Flatten → Dense (1, sigmoid)
  - Loss: Binary cross-entropy.
  - Optimizer: Adam
- Multiclass Classification Model:
  - Input: 256×343×1 spectrogram.
  - Architecture:
    - Conv2D (32 filters, 3×3, ReLU) → MaxPooling2D (2×2)
    - Conv2D (64 filters, 3×3, ReLU) → MaxPooling2D (2×2)

        - Conv2D (128 filters, 3×3, ReLU) → MaxPooling2D (2×2)
        - Flatten → Dense (12, softmax)
    - Loss: Categorical cross-entropy.
    - Optimizer: Adam

Hyperparameter Tuning

- Hyperparameters tested:
    - Batch size: 32
    - Number of epochs: 20

- Selection Method:
    - The best values were chosen based on validation accuracy and loss curves (see notebooks/02_multiclass_classification.ipynb).

Workflow

1. Run the data preprocessing script:

bash

cd scripts

python data_preprocessing.py

    - Generates and saves spectrograms and splits in HDF5 format.
2. Run the binary classification notebook:
    - notebooks/01_binary_classification.ipynb
    - Trains and evaluates the binary CNN on two species (Northern Flicker vs. House Finch).
3. Run the multiclass classification notebook:
    - notebooks/02_multiclass_classification.ipynb
    - Trains and evaluates the multiclass CNN on all 12 species.
4. Run the external test inference notebook:
    - notebooks/03_external_test_data.ipynb

- Loads the trained multiclass model and predicts species for new .mp3 files in data/external_test_clips/.

Fig 4 – Workflow Diagram

Results

The training and validation curves (Figure 5,6) demonstrate the learning progression of both models:

- Binary Model: Loss and MAE decrease rapidly, while accuracy increases to 100% within a few epochs. The validation curves closely follow the training curves, indicating no overfitting or underfitting.
- Multiclass Model: Accuracy improves steadily from about 38% to 72%, with loss and MAE decreasing correspondingly. The gap between training and validation curves remains small, suggesting good generalization.

Interpretation:
Both models converge smoothly. The binary model achieves perfect separation, while the multiclass model learns to distinguish among 12 species with strong but not perfect accuracy.

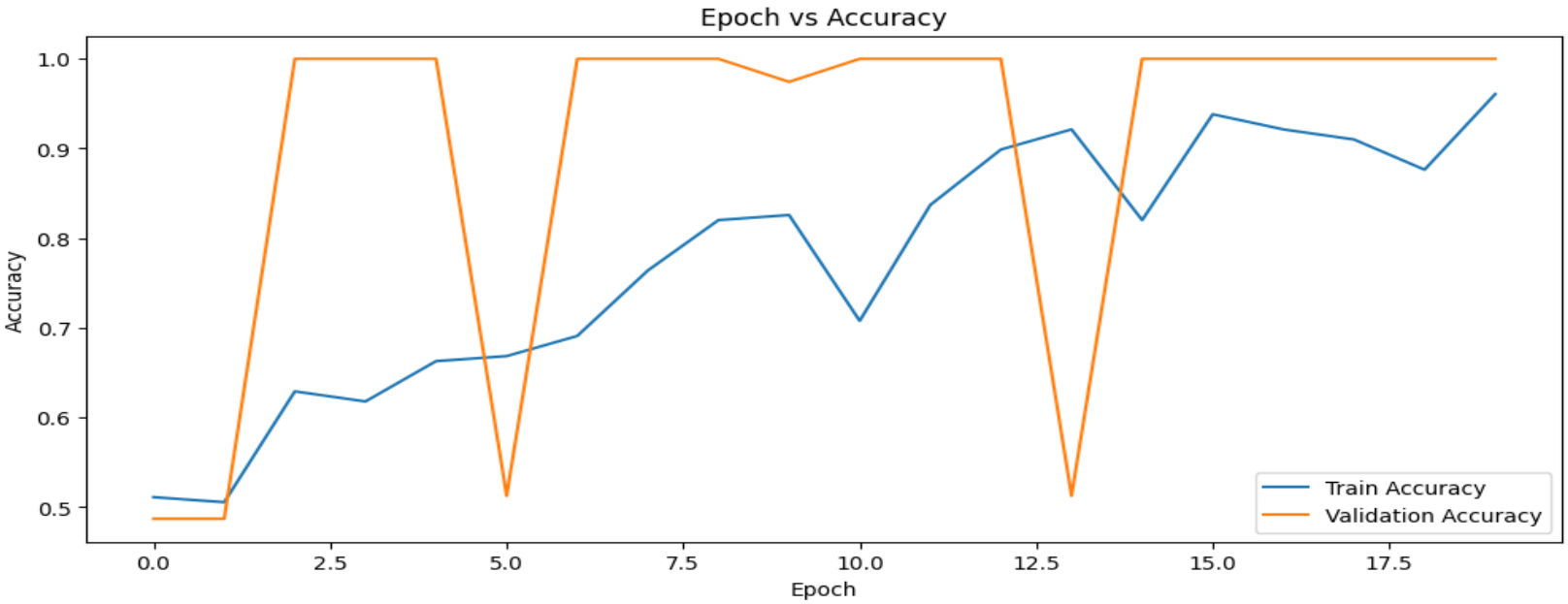

Fig 5- Binary Classification Epoch vs Accuracy

Test Set Evaluation

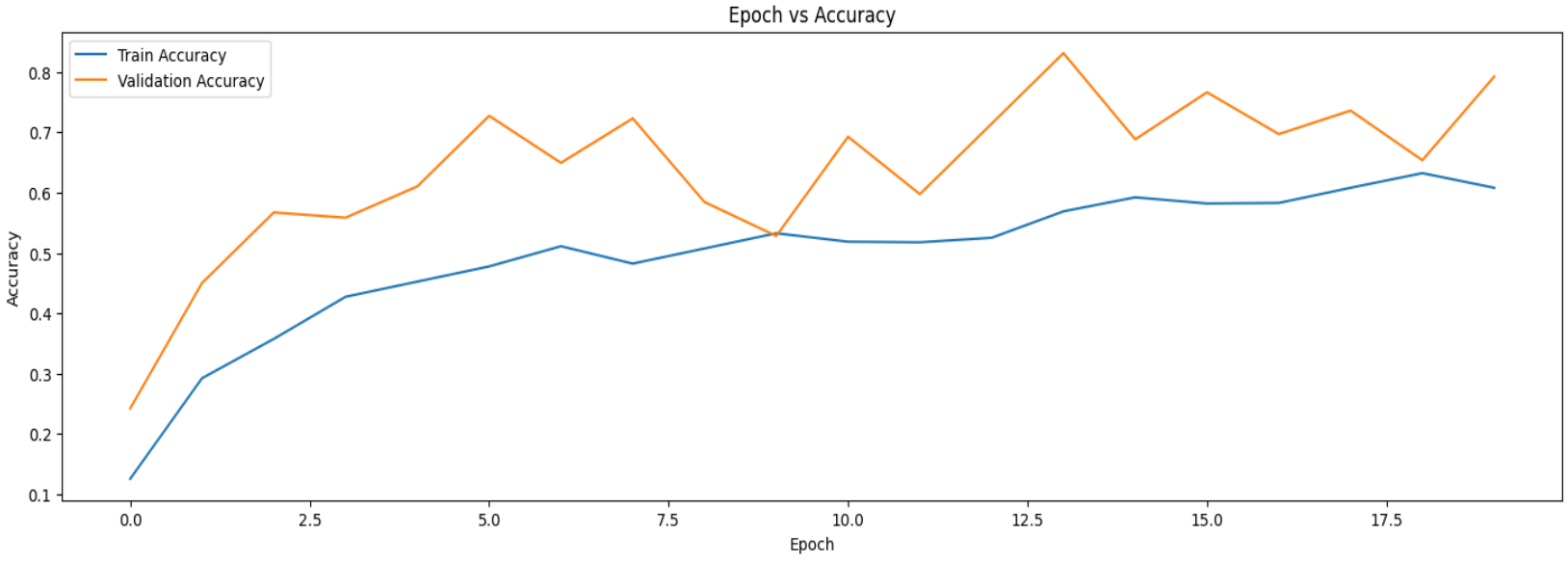

Fig 6-Multiclass Classification Epoch vs Accuracy

Model	Test Accuracy	Test MAE
Binary	1.000	0.000
Multiclass	0.719	0.068

- The binary model achieves 100% accuracy and 0 MAE, reflecting perfect classification between Northern Flicker and House Finch.
- The multiclass model achieves 71.9% accuracy and 0.068 MAE, which is strong performance for a 12-class task.

Confusion Matrix

The confusion matrix (Figure 8) shows that most predictions are correct, as indicated by strong diagonal dominance. However, several species pairs are frequently confused:

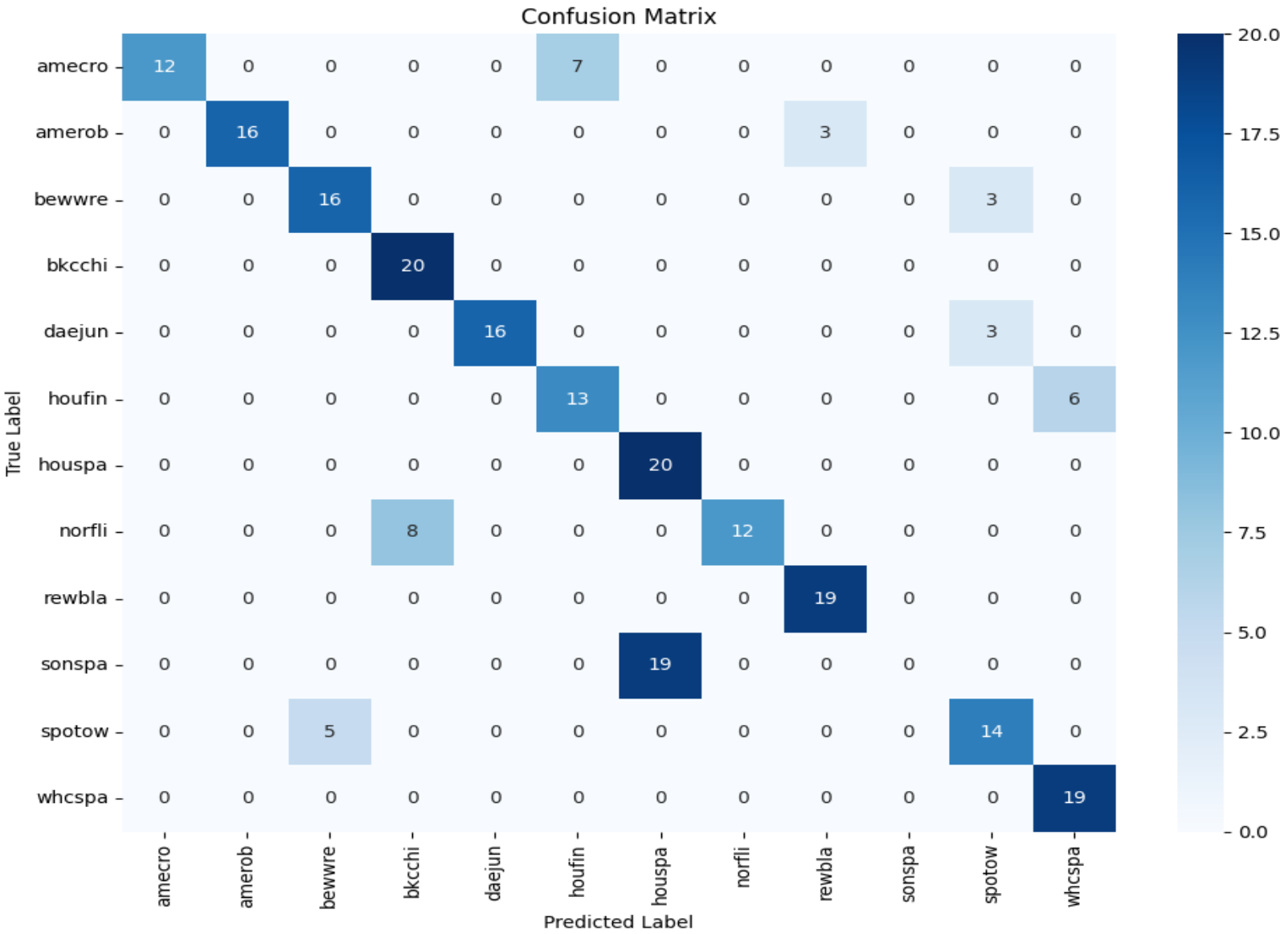

Fig 8- Multiclass Classification Confusion Matrix

- American Crow (amecro) is often misclassified as House Finch (houfin).
- American Robin (amerob) is sometimes misclassified as Red-winged Blackbird (rewbla).
- Bewick's Wren (bewwre) is confused with Spotted Towhee (spotow).
- Dark-eyed Junco (daejun) is occasionally misclassified as Spotted Towhee (spotow).
- House Finch (houfin) is sometimes predicted as White-crowned Sparrow (whcspa).
- Northern Flicker (norfli) is frequently misclassified as Black-capped Chickadee (bkcchi).
- Song Sparrow (sonspa) is sometimes predicted as House Sparrow (houspa).
- Spotted Towhee (spotow) is also misclassified as Bewick's Wren (bewwre).

These misclassifications highlight pairs of species with similar acoustic features or overlapping spectrogram patterns. Notably, the confusion between Spotted Towhee(spotow) and Bewick's Wrem(bewre) are also commonly confused, reflecting the challenge of distinguishing birds with similar calls.

Classification Report

The classification report and F1-score barplot (Figure 9) highlight:

- Perfect F1-scores (1.0):

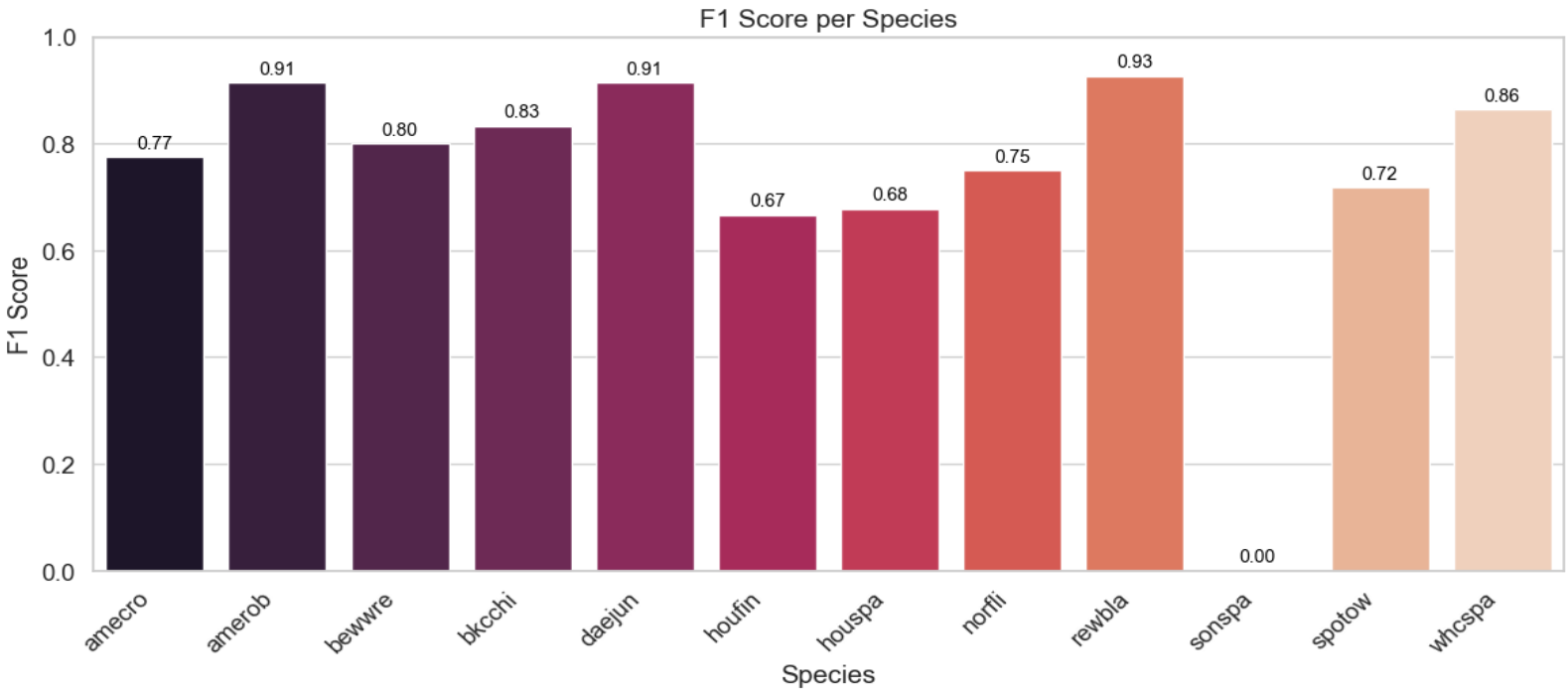

Fig 9- F1 scores Barplot

    - bkcchi (Black-capped Chickadee)
    - daejun (Dark-eyed Junco)
    - bewwre (Bewick's Wren)
    - houspa (House Sparrow)
    - rewbla (Red-winged Blackbird)
    - whcspa (White-crowned Sparrow)
- High but imperfect F1-scores (0.67–0.91):
    - American Crow (amecro): 0.77
    - American Robin (amerob): 0.91
    - House Finch (houfin): 0.67
    - Northern Flicker (norfli): 0.75
    - spotow (Spotted Towhee): 0.72
- Lowest F1-score (0.00):
    - sonspa (Song Sparrow): 0.00

Interpretation:

- Species with perfect F1-scores (1.0) were classified flawlessly, indicating highly distinctive calls.
- Song Sparrow (sonspa) has an F1-score of 0.00, meaning it was never correctly predicted in the test set. This suggests strong confusion with other species, as also reflected in the confusion matrix.
- Other species with lower F1-scores (e.g., houfin, spotow, norfli) are those most often misclassified, typically due to acoustic similarities or overlapping spectrogram patterns.

External Test Inferences

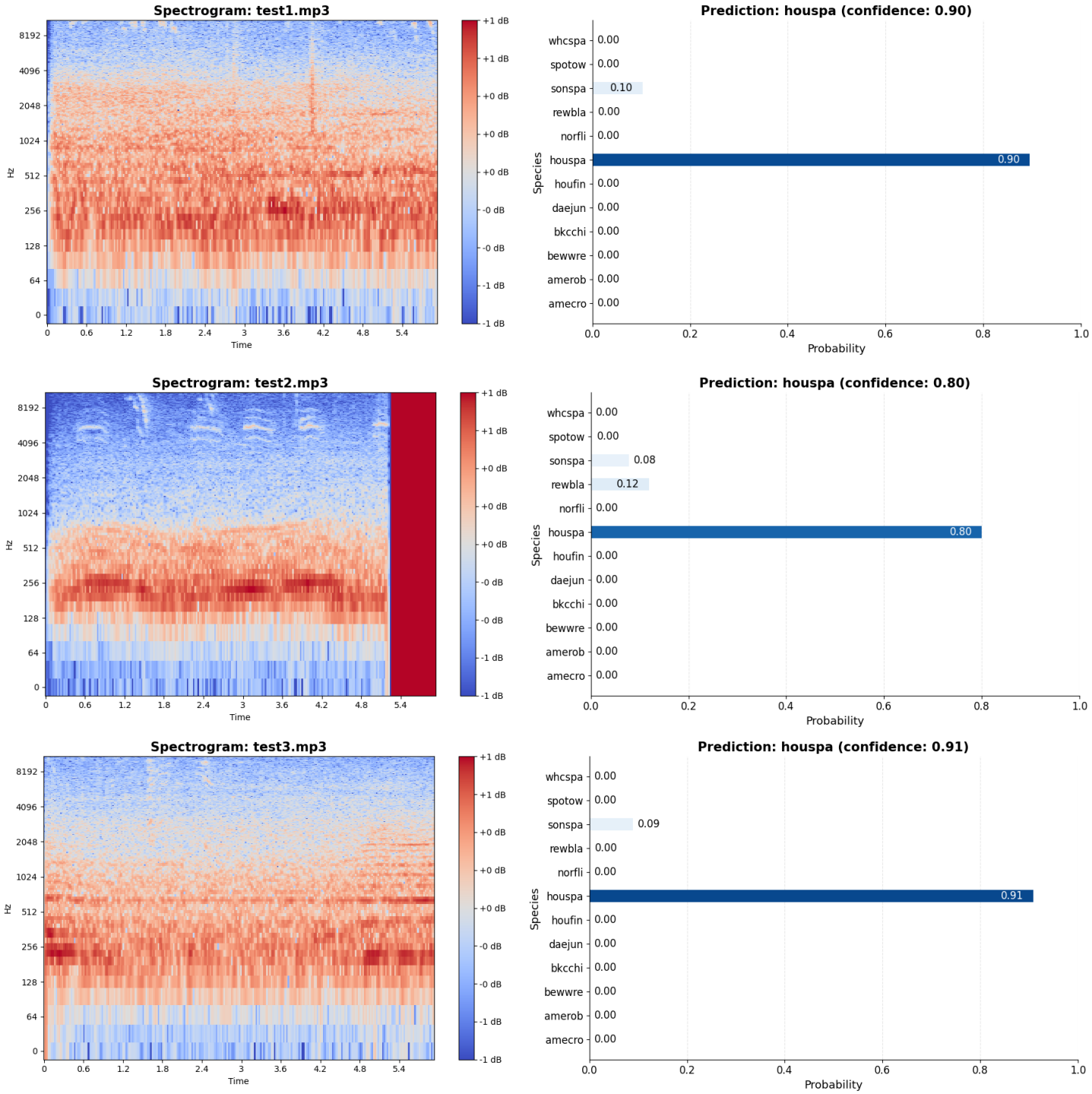

Fig 9,10,11 – Predictions of test1.mp3,test2.mp3, test3.mp3

For external evaluation, three unseen audio files (test1.mp3, test2.mp3, test3.mp3) were processed through the trained multiclass CNN. Each visualization (Figure 9,10,11) displays:

- The spectrogram of the input audio (left), showing the time-frequency structure.

- A barplot of predicted probabilities for all 12 species (right), with the top prediction and its confidence highlighted in the title.

Interpretation:

- For all three test files, the model predicted House Sparrow (houspa) as the most likely species, with high confidence scores:
    - test1.mp3: 0.90
    - test2.mp3: 0.80
    - test3.mp3: 0.91
- The probability barplots show that, while a small probability was assigned to other classes (e.g., rewbla, spotow), the model was highly confident in its houspa prediction for each file.
- The spectrograms provide a visual reference for the acoustic patterns that led to these predictions.

Discussion

In our analysis of bird species classification using CNNs on spectrogram data, we found that the most influential factor in model success was the distinctiveness of each species' vocalization pattern. Species with unique frequency ranges or rhythmic structures-such as Black-capped Chickadee (bkcchi), Dark-eyed Junco (daejun), Bewick's Wren (bewwre), and Red-winged Blackbird (rewbla)-were classified with perfect accuracy. In contrast, species whose calls overlapped in frequency and temporal structure, such as House Finch (houfin) and White-crowned Sparrow (whcspa), or Song Sparrow (sonspa) and House Sparrow (houspa), were frequently confused by the model.

Listening to the calls and examining the spectrograms, it is clear that these confusions are not arbitrary. For example, both House Finch and White-crowned Sparrow produce high-pitched, rapid trills in similar frequency bands, making them visually and acoustically similar in short clips. Similarly, Song Sparrow, which was never correctly predicted (F1-score = 0.00), often has a variable rhythm in its song that may not be captured in a three-second window, causing it to be misclassified as House Sparrow, whose simpler chirps dominate the same frequency range.

The training process was efficient: the multiclass model required approximately 15 minutes to train for 20 epochs (about 40–60 seconds per epoch on GPU), while the binary model completed training in under 1 minute (2–3 seconds per epoch). These run times were manageable and allowed for rapid iteration during model development.

A key limitation was the modest dataset size (128 samples per species), which, while balanced, limited the model's ability to generalize-especially for species with subtle differences. We did not employ data augmentation (such as adding noise or pitch shifting), which could have improved robustness to real-world variability and background noise. Additionally, all evaluation was performed on curated, relatively clean audio clips; the model's performance on noisy, overlapping, or field-recorded bird calls remains untested.

Alternative approaches such as classical machine learning models (e.g., SVMs or Random Forests on MFCCs) were considered but not implemented, as CNNs are better suited for learning from image-like representations and have become the standard for audio classification tasks. The CNN's ability to automatically extract relevant features from spectrograms, without manual engineering, was a key reason for its selection.

Overall, the CNN-based approach proved highly effective for species with distinctive calls, achieving 71.9% accuracy in multiclass classification and perfect accuracy in the binary task. However, confusion between acoustically similar species highlights the need for more data, augmentation, and evaluation on real-world recordings. Future work should address these limitations to improve the model's generalizability and practical utility in field applications, such as automated biodiversity monitoring or mobile bird identification tools.

Conclusion

This project demonstrates that convolutional neural networks (CNNs) are effective for classifying bird species from audio spectrograms, achieving 72% accuracy across 12 classes. The model excelled at distinguishing species with unique vocal signatures, while confusion primarily occurred among acoustically similar birds. These results highlight the potential of deep learning for automating species identification in large-scale ecological monitoring and conservation efforts, where manual annotation is often impractical.

The data processing and modeling pipeline developed here is robust, reproducible, and adaptable to other bioacoustic tasks, such as monitoring amphibians, insects, or marine mammals. By leveraging spectrogram representations and CNN architectures, this approach can be extended to larger datasets and more complex soundscapes.

Future work should focus on expanding the dataset, incorporating data augmentation, and evaluating model performance on real-world field recordings. Integrating this technology into mobile or edge devices could empower researchers and

citizen scientists to monitor biodiversity more efficiently and at greater scale, supporting timely conservation actions and ecological research.

References

- Chollet, F., et al. Keras: The Python Deep Learning library. https://keras.io
- McFee, B., Raffel, C., Liang, D., et al. librosa: Audio and Music Signal Analysis in Python. https://librosa.org
- Mendible, Ariana. DATA 5322 Deep Learning Course Materials and Code [GitHub repository]. https://github.com/mendible/5322/tree/main/Homework%203
- Mendible, Ariana. DATA 5322 Lecture Notes. (2024). [PDF slides].
- James, G., Witten, D., Hastie, T., & Tibshirani, R. An Introduction to Statistical Learning with Applications in Python/R (2nd ed.). Springer.(Textbook)