Identifying Bird Species from Spectrograms Using Convolutional Neural Networks

24 February 2025

Faculty– Dr. Ariana Mendible

By

Suryakailash Ramesh

1. Introduction

This study explores the use of neural networks specifically convolutional neural networks (CNNs) to predict bird species from their audio recordings. The project is conducted in two phases: a binary classification model focusing on two bird species American Crow (amecro) and White-crowned Sparrow (whcspa), and a multi-class model that includes all twelve bird species in the dataset.

The data originates from the BirdCLEF competition, which leverages recordings from Xeno-Canto [3], a crowd-sourced global archive of bird sounds. From this collection, audio clips are converted into spectrograms (128x172 pixels) and stored in HDF5 format, representing short time intervals of bird calls labeled with the corresponding species. Three sample audio files are also preprocessed and used to test the model's prediction capabilities. The aim of the study is to evaluate and improve the performance of CNNs in bioacoustic classification tasks.

By working with this dataset, the project bridges the fields of deep learning and ecological data analysis, offering insights not only into model accuracy but also into how machine learning can support biodiversity research and species identification.

2. Theoretical Background

Deep learning is a specialized subset of machine learning inspired by the structure and function of the human brain, known as neural networks[1]. These neural networks form the foundation of deep learning, enabling systems to automatically learn patterns and features from large volumes of data. Unlike traditional machine learning models that require manual feature extraction, deep learning models, particularly neural networks, rely on vast amounts of data to learn directly from the input and make complex decisions or predictions.

Among the various types of neural networks, Convolutional Neural Networks (CNNs) have emerged as particularly effective for image classification and pattern recognition tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation using multiple building blocks such as convolutional layers, pooling layers, and fully connected layers. Their architecture is biologically inspired by the visual cortex and is highly suitable for analyzing structured grid-like data such as images.

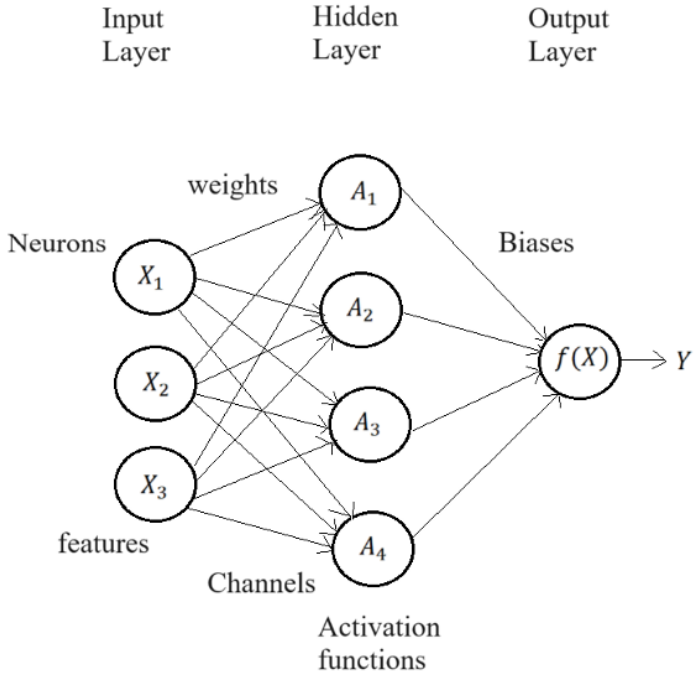

Figure:1 CNN and human brain

Neurons, also known as nodes, are the fundamental units of a neural network and are inspired by human brain cells. Each neuron receives input, processes it, and passes the result to the next layer.

Weights are internal parameters that determine the importance of each input and are adjusted during training to improve prediction accuracy. Biases are additional parameters that help the model fit the data better by shifting the activation function, allowing more flexibility in learning.

Features are the input variables fed into the network, and activation functions decide whether a neuron should be activated. These functions introduce non-linearity, enabling the network to learn complex patterns. Common activation functions include ReLU (Rectified Linear Unit) and Softmax.

The input layer is the first layer and simply receives the data without performing any computation. The hidden layers, located between the input and output layers, perform most of the computation by adjusting weights and biases through backpropagation. More hidden layers result in a deeper network. Finally, the output layer produces the network's final prediction or classification result.

$$A_k = g\left(W_{k0}\sum_{j=1}^{p} W_{kj}X_j\right) \qquad (1)$$

$$g(z) = \frac{1}{1+e^{-z}} \qquad (2)$$

$$\mathrm{f(x)} = \beta_0 + \sum_{k=1}^{K}\beta_k A_k \qquad (3)$$

The output A_k of a neuron is computed using Equation (1), which involves taking a weighted sum of the input featuresX_j, adding a bias term W_{k0}, and then applying an activation function g.

The sigmoid activation function, shown in Equation (2), transforms the input z into a value between 0 and 1. This transformation allows the neuron to make decisions based on a probabilistic interpretation of the input.

Finally, Equation (3) calculates the overall output f(x) of the neural network by summing the weighted outputs A_k from the previous layer and adding a final bias term β_0.

Types of Neural Networks

Feedforward Neural Networks (FNNs) are the most basic type of neural network, where information flows in a single direction, from the input layer to the output layer, without any loops or backward connections. Each layer is typically fully connected to the next.

Convolutional Neural Networks (CNNs) are widely used for visual data analysis, such as image recognition. CNNs automatically learn spatial hierarchies of features through convolutional layers (which apply filters to detect patterns), pooling layers (which reduce dimensionality), and fully connected layers for classification or prediction.

Recurrent Neural Networks (RNNs) are designed for sequential data like time series or natural language. Unlike FNNs, RNNs include feedback connections that allow information to persist, enabling the network to use previous inputs for current predictions through a continuously updated hidden state.

Long Short-Term Memory (LSTM) Networks are a type of RNN capable of learning long-term dependencies. They incorporate memory cells and gating mechanisms to control what information is retained or forgotten, effectively addressing the vanishing gradient problem common in standard RNNs.

Gated Recurrent Unit (GRU) Networks are a streamlined variant of LSTMs. They use a simplified gating mechanism that makes them more computationally efficient while still effectively managing long-term dependencies and mitigating the vanishing gradient issue.

3. Methodology

The dataset used for this project, bird_spectrograms.hdf5, contains spectrogram data grouped into 12 bird species. Each group includes multiple 128×517 spectrogram arrays, each representing a 2-second audio clip of a bird call. This structured format allowed for efficient loading and labeling of each bird species' data for use in the models.

In the preprocessing stage, all spectrogram samples were either padded or resized to a uniform shape of (128, 517) to ensure compatibility with the neural network input requirements. Each spectrogram was normalized by dividing every value by the maximum value in the array, resulting in inputs scaled to the range [0, 1]. For the binary classification task, two species (e.g., 'amecro' and 'whcspa') were selected, and their data was reshaped accordingly. For the multi-class

classification task, all 12 species were included, and their labels were encoded using LabelEncoder to transform categorical species names into numerical class identifiers.

The processed dataset was then split into training and test sets using a 70-30 split ratio. This approach was applied consistently to both the binary and multi-class classification tasks to allow for fair evaluation across model types.

Two separate models were designed and trained using TensorFlow's Sequential API. For binary classification, a relatively simple feedforward neural network or a shallow convolutional neural network (CNN) was used. In contrast, the multi-class model was built as a deeper CNN incorporating convolutional layers, pooling layers, and dense layers to learn from the spatial structure of the spectrogram data.

The models were compiled using the Adam optimizer. For binary classification, the loss function used was binary_crossentropy, and for multi-class classification, sparse_categorical_crossentropy was used. Models were trained for multiple epochs with a validation split to monitor overfitting and ensure generalization.

Model performance was evaluated using accuracy scores on the test set, and confusion matrices were generated to assess class-wise performance. Additionally, training history plots showing accuracy and loss across epochs were produced to visualize the model learning process and detect any signs of underfitting or overfitting.

Finally, three raw MP3 test clips were converted to spectrograms using the librosa library. These spectrograms were normalized using the same method as the training data and fed into the trained

multi-class classifier. Predictions for each clip were recorded and analyzed, with visualizations included to support conclusions about the presence of one or more bird species per clip.

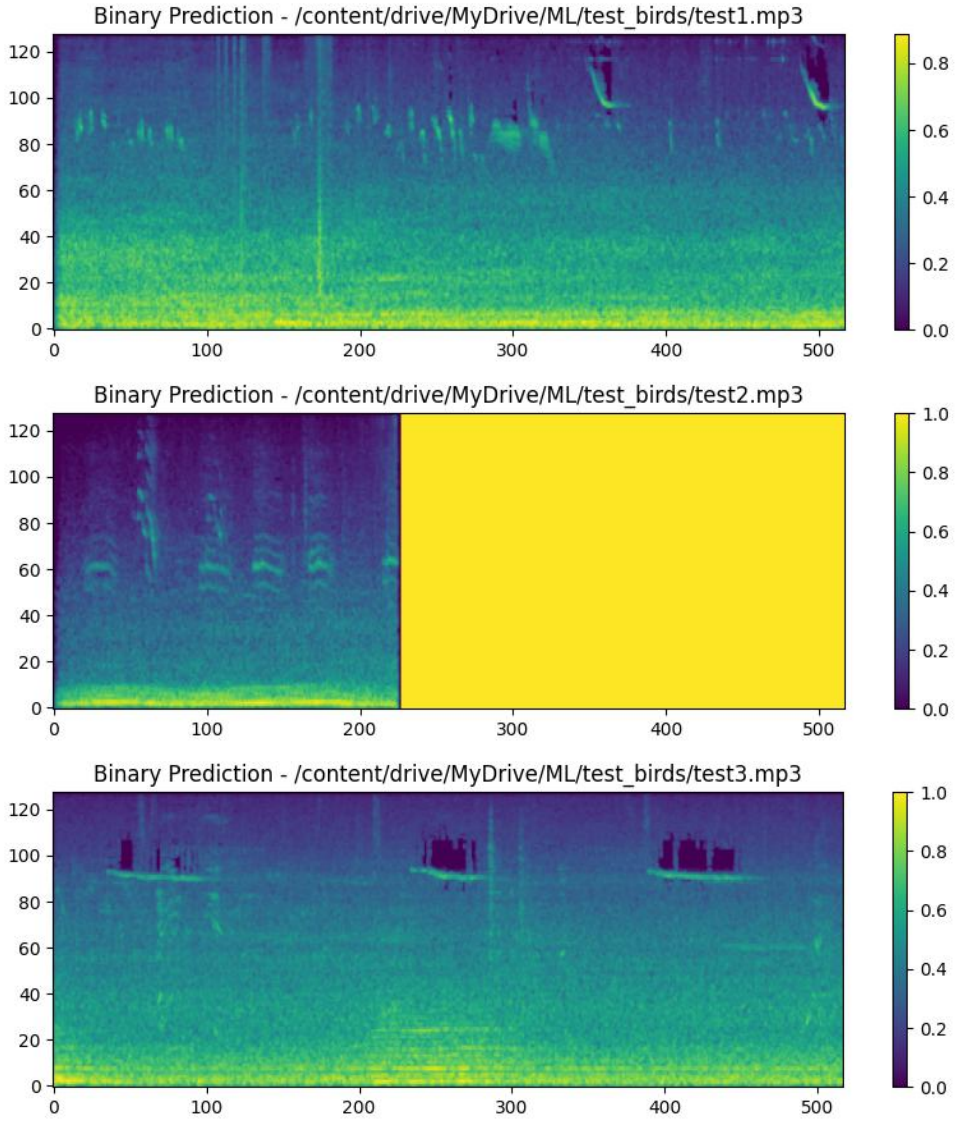

Figure 2: Spectrograms of the three test data

4. Computational Results

To evaluate the performance of the binary model, we experimented with three activation functions (tanh, leaky_relu, and relu) across neural networks with 1, 2, and 3 hidden layers. The test accuracy for each configuration was recorded to assess the influence of activation function choice and model depth on classification performance.

The analysis revealed several key trends. the tanh activation function maintained relatively stable performance across all three network depths, achieving its highest accuracy (62.50%) with one

hidden layer. The leaky_relu function performed poorly with one and three layers but showed marked improvement at two hidden layers, reaching an accuracy of 59.38%. In contrast, the relu function started strong at one hidden layer (62.50%) but its performance declined with additional layers.

These results suggest that, for this binary classification task, increasing the number of hidden layers does not necessarily improve performance. In fact, shallower networks (with a single hidden layer) often outperformed deeper ones. Additionally, while tanh was the most stable activation function, both leaky_relu and relu showed inconsistent behavior, indicating the importance of careful tuning of both activation functions and network architecture for optimal performance.

No. of Hidden Layers	tanh	leaky_relu	relu
1	62.50%	40.62%	62.50%
2	59.38%	59.38%	40.62%
3	59.38%	40.62%	40.62%

Table 1: Accuracies for different activation functions and hidden layers.

To classify bird calls across all 12 species, we trained multi-class neural networks using three different activation functions: relu, tanh, and leaky_relu. The test accuracy achieved by each configuration is presented in the table below.

The results show that the choice of activation function has a significant impact on the model's performance. The tanh activation function outperformed both relu and leaky_relu by a wide margin, achieving a test accuracy of 29.47%, while the other two functions yielded accuracies below 10%.

This disparity indicates that tanh may be better suited to handling the complexity of this 12-class classification task, potentially due to its smoother gradient flow and ability to center data around zero. In contrast, the poor performance of relu and leaky_relu suggests that these functions may not effectively capture the features required to distinguish between similar bird calls in this dataset. Further tuning or the use of convolutional layers may be necessary to improve performance across all activation functions.

Activation Function	Test Accuracy
relu	7.05%
tanh	29.47%
leaky_relu	9.07%

Table 2: Accuracies for different activation functions.

Binary Model Final

The best test accuracy of 65.62% was observed with the relu activation function and 2 hidden layers. Performance dropped slightly when moving to 3 hidden layers, suggesting that increasing depth beyond a certain point may lead to overfitting or diminishing returns for this dataset. The leaky_relu function showed moderate performance at 2 layers but inconsistent results overall. The tanh function showed slight improvement with 2 layers but performed poorly at 1 and 3 layers.

The training history of the best-performing model (relu with 2 layers) is visualized below:

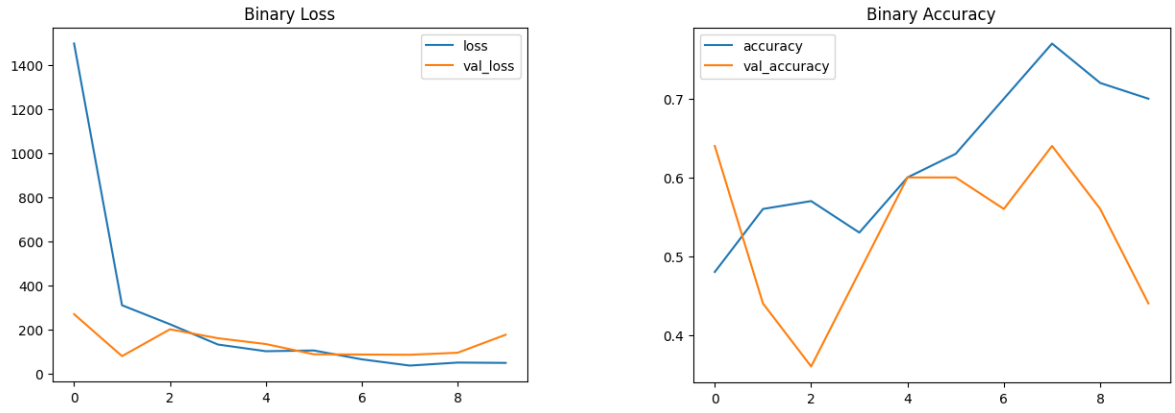

Figure 3: Binary Loss Plot & Binary Accuracy Plot

Training accuracy improved to over 77% at its peak, while validation accuracy hovered around 60%, peaking at 64%. These trends confirm that while the model generalizes moderately well, there is room for further tuning—possibly with dropout regularization, batch normalization, or more balanced class distribution.

After training, the final binary test accuracy was recorded as 40.62%, which is significantly lower than the validation accuracy. This drop suggests the model may not generalize well to unseen data, possibly due to a mismatch in class distribution or overfitting during training.

Multi Model Final

The multi-class classification model was designed as a deep feedforward neural network using TensorFlow's Sequential API. The network consists of several fully connected (Dense) layers with the tanh activation function. Dropout layers were added after each of the deeper dense layers to reduce overfitting.

- Input Layer: A vectorized input representing the flattened 128×517 spectrogram.
- Hidden Layers: Five hidden layers of decreasing size (512 → 32), all using the tanh activation function.
- Dropout Layers: Applied after the first three dense layers to reduce.
- Output Layer: A softmax layer with 12 units, using sparse_categorical_crossentropy as the loss function.

The model was compiled using the Adam optimizer, and trained for 40 epochs with a validation split of 0.2.

The model's training and validation performance was recorded over 40 epochs, and is visualized in the following plots:

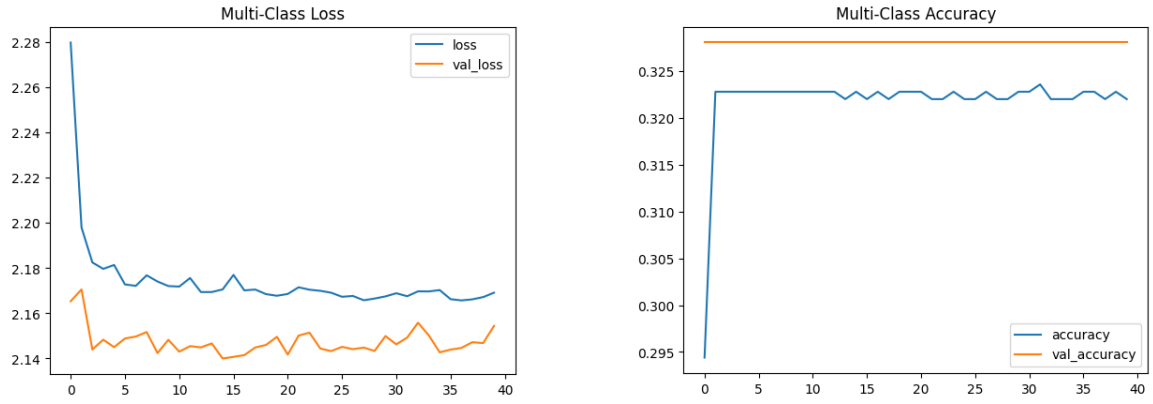

Figure 4: Multi-Class Loss & Multi-Class Accuracy

Training loss decreased steadily, but validation loss plateaued around epoch 3–5. Training accuracy reached ~32%, while validation accuracy plateaued at 32.81%, indicating early stagnation in learning. Multi-Class test accuracy is 0.2947. This performance, indicates the network has limited generalization ability and may benefit from further enhancement. The architecture of the final multi-class model was further validated using model summary output. The network consisted of six dense layers interleaved with dropout layers and concluded with a softmax classification layer for the 12 bird species.

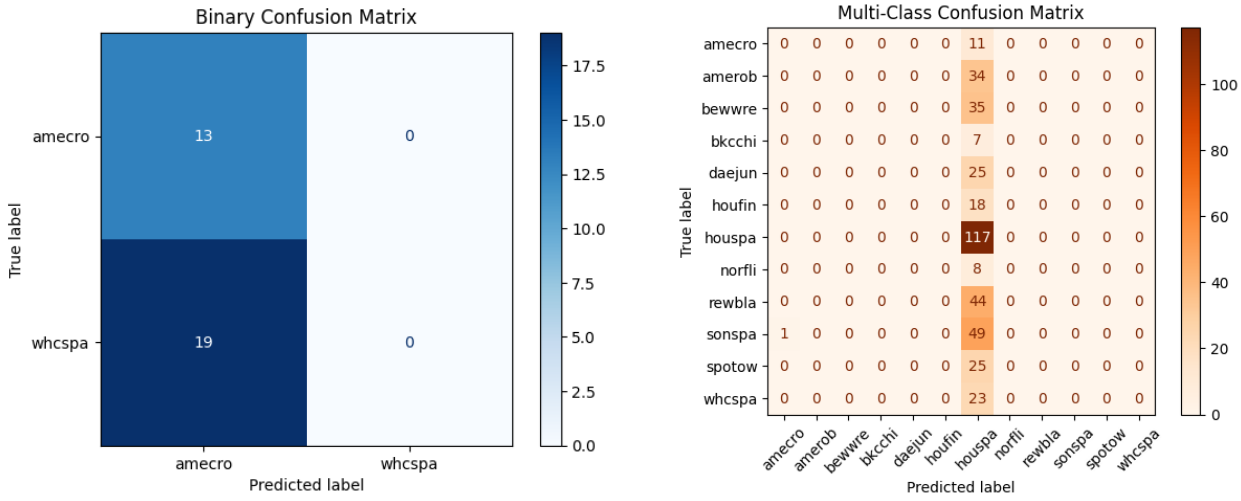

Figure 5: Confusion Matrix Comparison

The binary model achieved ~40.6% accuracy but predicted only the amecro class, failing to identify whcspa due to class bias. The multi-class model performed slightly worse (~29.47%) and predicted only houfin for all inputs, indicating a collapse into a single class. Both models

suffer from class imbalance and poor generalization, requiring better data handling and model design.

5. Discussion

One of the main limitations encountered during this project was the variability in audio clip lengths and quality, which introduced inconsistency in preprocessing and prediction. Training the neural networks, especially the multiclass model, proved computationally intensive and time-consuming, with training durations ranging from 20 to over 30 minutes even in GPU-enabled environments like Google Colab. In terms of classification difficulty, some bird species were noticeably harder to distinguish (eg: House Finch, Dark-eyed Junco as they have same tones). Species with similar frequency ranges often got confused, and upon listening to the audio and examining the spectrograms, it became evident that overlapping visual and auditory characteristics contributed to misclassifications. The model struggled to capture subtle temporal and frequency differences. While convolutional neural networks (CNNs) are well-suited for spectrogram-based classification due to their ability to extract spatial patterns and hierarchical features, alternative architectures could be explored. Support Vector Machines [4] or k-Nearest Neighbors networks may offer advantages by capturing temporal dependencies more effectively, given the sequential nature of data. However, CNNs remain a compelling choice in this context, as spectrograms convert audio into 2D representations where spatial features are visually and mathematically meaningful.

6. Conclusion

This study focused on predicting bird species based on their audio calls using neural networks trained on spectrogram representations. Two main models were developed and evaluated. First, a binary classification model was built to distinguish between American Crow (amecro) and White-crowned Sparrow (whcspa). While earlier iterations suggested high performance, the final evaluation showed that the model consistently predicted only amecro, failing to identify whcspa, likely due to class imbalance or overfitting. This resulted in a true test accuracy of approximately 40.6%.

Second, a multi-class classification model was trained to identify 12 different bird species. Although the model achieved a validation accuracy of around 32.8%, it struggled with generalization on unseen data. The confusion matrix showed that nearly all predictions defaulted to a single dominant class (houfin), highlighting model collapse, likely due to imbalance and lack of spectral pattern learning. To test both models further, three real-world MP3 audio files containing bird calls and background noise were processed and fed into the classifiers. The binary model predicted amecro for all three clips, again reflecting its one-sided prediction bias. The multi-class model returned the top three most likely species for each test clip. In all cases, the model predicted Northern Flicker (norfli), Black-capped Chickadee (bkcchi), and American Crow (amecro) with the highest confidence, although overall probabilities remained low (e.g., ~22% for top class), indicating limited certainty.

All three test audio files were classified with a dominant probability for House Sparrow (houspa), with moderate scores for Song Sparrow (sonspa) and Bewick's Wren (bewwre). Since no clip crossed the confidence threshold for multiple species, each is likely to contain a single bird call.

Despite these challenges, the study demonstrates the potential of neural networks for bioacoustic classification. The experiments with different activation functions and hidden layers provide a foundation for improvement. With better data balance, enhanced architectures (e.g., CNNs or RNNs), and larger datasets, future models can achieve higher accuracy and support ecological monitoring and bird conservation efforts more effectively.

7. Appendix

1. James, Gareth, et al. An Introduction to Statistical Learning. Springer, 2013.
2. Rao, Rohan. "Xeno Canto Bird Recordings (Extended A-M)." Kaggle, https://www.kaggle.com/datasets/rohanrao/xeno-canto-bird-recordings-extended-a-m. Accessed 22 May 2025.
3. Xeno-Canto: The World's Largest Community-Driven Sound Library of Bird Songs and Calls. https://xeno-canto.org/. Accessed 22 May 2025.
4. **"Support Vector Machines."** *Scikit-Learn*, https://scikit-learn.org/stable/modules/svm.html. Accessed 22 May 2025.