Classification of Seattle Bird Sounds based on Neural Networks

Zahra Ahmadi Angali

Abstract

This study classifies the most common bird species in the Seattle area through the implementation of Neural Networks. Utilizing spectrograms derived from 10 sound clips of various lengths for each of the 12 bird species. This classification problem is examined through a binary classification approach on two selected species, a multi-classification using all 12 species, and classification on an unlabeled test set. The binary classification model using the CNN neural architecture and RNN is conducted on the two selected species of Blue Jay and Steller's Jay the model is optimized and tuned through various parameters resulting in the best-performing model with 16 batch sizes and 20 epochs giving an accuracy of 77% on the test set and 76% on the training. Various CNN models are also trained on the train set of the 12 species and the model's performance is evaluated through metrics like loss validation, accuracy, and validation accuracy, the optimal model with an accuracy of 58% on the train set and 87% on the training set. The best-performing model of the multi classification problem is then used for the classification of the unlabeled data set presenting Western Meadowlark species to be mostly detected with being present in the top 5 five species that are identified in all three audio set.

Introduction

Understanding bird calls is an important sector in comprehending the physiology, ecology, and habitats of birds and is a concern for ornithologists. The focus of this study is to identify the bird species utilizing spectrograms. The 12 species present are the following American Crow, Barn Swallow, Black-capped Chickadee, Blue Jay, Dark-eyed Junco, House Finch, Mallard, Northern Flicker, Red-winged BlackBird, Steller's Jay, Western Meadowlark, white-crowned Sparrow. Due to the nature of Neural Networks to comprehend complex data, they are suitable and possibly the only candidate to analyze audio types of data. When dealing with audio data the most common issue would be to differentiate the desired sound from all the other sounds present in the audio, demanding the need to optimize the model and constantly assessing its performance. The Convolutional Neural Network model is set to the task and trained for binary classification, multi-classification, and transfer learning as well. Through the utilization of deep learning

models new methods for studying bird habits and understanding their ecology can be introduced.

Theoretical Background

[1]Neural Networks

Neural networks are machine learning models that are influenced by the functioning of neurons found in the human brain. They are made up of interconnected nodes organized into multiple layers. During training, the connections between nodes are adjusted to reduce errors between predicted and actual outputs, resulting in the network to learn complex patterns. Once trained, neural networks can make predictions about new data. While highly efficient and sometimes transparent, their ability to learn from data makes them useful for tasks such as image recognition, natural language processing, and predictive modeling in a variety of fields.

[2]Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of neural network that does well at processing grid-like data. CNNs are intended to automatically learn a spatial structure of features from input data. Convolutional layers apply filters to the input to detect low-level features such as edges and shapes. These features are then combined by subsequent layers to detect more advanced features. CNNs also include pooling layers, which reduce the spatial dimensions of feature maps, lowering the complexity of computation. The final layers are usually fully connected, allowing the CNN to generate class probabilities.

[3]RNN

Recurrent Neural Networks (RNNs) are specialized neural networks that process sequential data such as text, speech, or time series. Unlike traditional networks, RNNs have connections that form loops, allowing information to be carried from one step to the next. This allows them to keep an internal memory or state that records context and patterns in the order. At each step, the RNN processes the present input and the previous step's state through its layers, producing an output and passing a changed state to the next step. This allows the network to gain an understanding of data as it goes on through each step.

Spectrogram

In short a spectrogram is an image of sound showing the spectrum of frequencies in a signal as it changes over time. It displays the frequency content of the signal along the vertical axis, with

low to high frequencies. The horizontal axis shows time. The intensity of each point on the spectrogram represents the amount of energy or amplitude present at that specific frequency and time. Spectrograms are widely used for analyzing and visualizing audio signals such as speech, music, and animal calls. They allow us to see how the frequency changes over time.

Activation Function

Activation functions are important components of artificial neural networks. They use the input values to determine whether a node or neuron in the network should be "activated" or not. Without activation functions, neural networks would be just like a linear model and limited in their capabilities. Activation functions introduce non-linearity, allowing neural networks to learn and comprehend complex patterns and relationships in data that aren't just straight lines. ReLU, sigmoid, and softmax are three common activation functions.

Methodology

Binary Classification

The processed spectrogram data is in an HDF5 format which makes the exploration of the data much easier while maintaining the tensor format. Due to the nature of the data, the spectrogram file is loaded using the h5py providing easy access and to label the data. A function is defined that loads and preprocesses data from an HDF5 file, with a focus on the binary classification of the target species (Blue Jay and Steller's Jay). Each target species is assigned a unique label. This is done by creating a dictionary called species_labels, in which a numerical label is given to each species. Moving on with the preprocessing stage for the binary classification for each species, the function extracts spectrogram data and adds it to the features list. Corresponding labels are added to the label list. This requires iterating through each slice of the spectrogram data which is a 3D array.

It is critical for neural network training to normalize the input data. Therefore, the features are converted to numpy arrays and normalized by dividing by the maximum value, ensuring that all values fall between 0 and one. Then the features are reshaped to include the number of slices (e.g., 256x343x1), ensuring that they meet the CNN's input requirements.

The preprocessing stage of the binary classification is followed by the split of the data into 30% test and 70% to ensure the model has enough data to learn and to test the model's performance on the test set, with a fixed random state that guarantees reproducibility.

Model Architecture:

A Convolutional Neural Network (CNN) is defined for binary classification. That accepts the input shape (256, 343, 1). Three Conv2D layers use ReLU activation and (3x3) filters. Each convolutional layer is followed by a MaxPooling2D layer that reduces the dimensionality. Followed by a flattening layer followed by a dense layer with 128 neurons and ReLU activation, followed by an output layer with a single neuron and sigmoid activation, for binary classification. In the layers of some of the models dropouts of 0.5 and 0.25 are also defined to prevent overfitting. Along With defining padding equal to ‘same’ for some models.

The model is later compiled with the Adam optimizer and binary cross-entropy loss since it is a binary classification. Accuracy is used as an assessment metric. The models are trained over various number of epochs and batch sizes to obtain the best results. Early stopping is also defined to prevent overfitting, validation loss is also monitored the best weights is restored if validation performance does not improve after 5 epochs. Lastly, the model's performance is evaluated across both the training and testing sets. A performance over the number of epochs is presented with Metrics like loss and accuracy that are printed for both sets to evaluate the model's performance.

For the Recurrent Neural Network (RNN) the data is reshped that is compatible with the input shape of an RNN as for the model architecture it consists of two Long Short-Term Memory (LSTM) layers, followed by two dense layers. LSTM layers are effective at detecting temporal dependencies in sequential data. The final dense layer performs binary classification using a sigmoid activation function.

The model is later compiled with the Adam optimizer and binary cross-entropy loss since it is a binary classification. Accuracy is used as an assessment metric. The models are trained over various number of epochs and batch sizes to obtain the best results. Early stopping is also defined to prevent overfitting, validation loss is also monitored the best weights is restored if validation performance does not improve after 5 epochs. Lastly the model's performance is evaluated across both the training and testing sets. A performance over the number of epochs is presented with Metrics like loss and accuracy that are printed for both sets to evaluate the model's performance.

Multi-Classification

The preprocessing stage for the multi-classification is similar of the binary classification this time instead of only two target species that were chosen for the binary classification all 12 species are chosen. Followed by the split of the data into 30% test and 70% to ensure the model has enough data to learn and to test the model's performance on the test set, with a fixed random state that guarantees reproducibility.

Model Architecture:

A Convolutional Neural Network (CNN) is defined for binary classification. That accepts the input shape (256, 343, 1). Three Conv2D layers use ReLU activation and (3x3) filters. Each convolutional layer is followed by a MaxPooling2D layer that reduces the dimensionality. Followed by a flattening layer followed by a dense layer with 128 neurons and ReLU activation, followed by an output layer with a single neuron and softmax activation, for multi classification.
In the layers of some of the models dropouts of 0.5 and 0.25 are also defined to prevent overfitting. Along With defining padding equal to ‘same’ for some models.
.

The model is later compiled with the Adam optimizer and binary cross-entropy loss since it is a binary classification. Accuracy is used as an assessment metric. The models are trained over various number of epochs and batch sizes to obtain the best results. Early stopping is also defined to prevent overfitting, validation loss is also monitored the best weights is restored if validation performance does not improve after 5 epochs. Lastly, the model's performance is evaluated across both the training and testing sets. A performance over the number of epochs is presented with Metrics like loss and accuracy that are printed for both sets to evaluate the model's performance.

Unlabeled Data

The Root Mean Square (RMS) energy of the audio signal is calculated the loud parts of the audio which are likely to contain bird calls are identified. And then audio segments that are louder than a predefined threshold (20 decibels below the peak)are identified. This helps to separate potential bird calls from background noise. The audio segment is divided into smaller windows (2second windows). Later on the spectrogram is padded or trimmed to ensure it has a fixed width of 343 frames. This is done to maintain a consistent input shape for the neural network. Each spectrogram is reshaped to add a number of sample dimension, converting it to the shape (256, 343, 1).

Using the best model from the multi-classification the unlabeled data set is predicted. With the same parameters only the input data is ensured to be compatible with the model and as the data set is unlabeled the probability for the most likely species to be heard is presented.

Computational Results

Binary Classification

For the binary classification of the two selected species, two main model architectures of CNN and RNN are implemented with different batch sizes and epochs and for some a dropout value is also defined the results of the models are presented in the table below.

The first model with a CNN architecture and 20 epochs and 32 batch size has a high training accuracy of 85.71%, indicating that it fits the data well. However, the test accuracy is significantly lower, at 59.26%, indicating that the model is not able to generalize to new data well. This difference between train and test accuracy indicates overfitting, in which the model learns the training data too well but fails to perform well on new data.

With adding dropout to this model the training accuracy is reduced to 77%, indicating a more regularized model. The test accuracy increases to 66%, indicating improved generalization. The lower test loss of 0.63 compared to model1 suggests more accurate predictions on the test set. Dropout appears to have reduced overfitting and improved the model's performance on unseen data.

By changing the batch size to 16 of the CNN model it can be seen that the model has good accuracies of 76% for train and 77% for train, with the difference between the accuracies being close indicating good generalization. The low test loss of 0.55 indicates that the model predicts accurately on the test set. The smaller batch size of 16 may have helped to achieve this balance, allowing the model to learn more effectively from the data.

It can be seen that by adding dropout to Model 3 does not lead to better results as the train accuracy remains the same, but the test accuracy drops significantly, suggesting that the dropout may have been too aggressive, leading to underfitting when compared with the previous model. The higher test loss confirms that the model's performance on the test set is not optimal.
Increasing the number of epochs to 30 has improved train accuracy while maintaining high test accuracy, indicating that the model benefited from additional training iterations. The test loss is still moderate, implying effective learning and improved generalization.

The RNN model's train and test accuracy are balanced with 66% train and 73% test , indicating good generalization and the gap between the train and test is not significant enough for it to be a severe case of overfitting. The test accuracy is comparable to CNN with dropout models, but with slightly lower loss values. This indicates that the RNN model is effective for this classification task, offering a viable alternative to CNNs with comparable performance.

Models (Epochs, batch size)	Train Acc	Test Acc	Test Loss	Train Loss
model1.CNN(20, 32)	85.71%	59.26%	0.7663	0.5062
model2.CNN(20, 32), Dropout	77%	66%	0.63	0.59
model3.CNN(20, 16)	76%	77%	0.55	0.49
model3.CNN(20, 16),Dropout	76%	55%	0.68	0.64
model3.CNN(30, 16)	80%	74%	0.63	0.57
model1.RNN(20, 32)	73%	66%	0.57	0.54

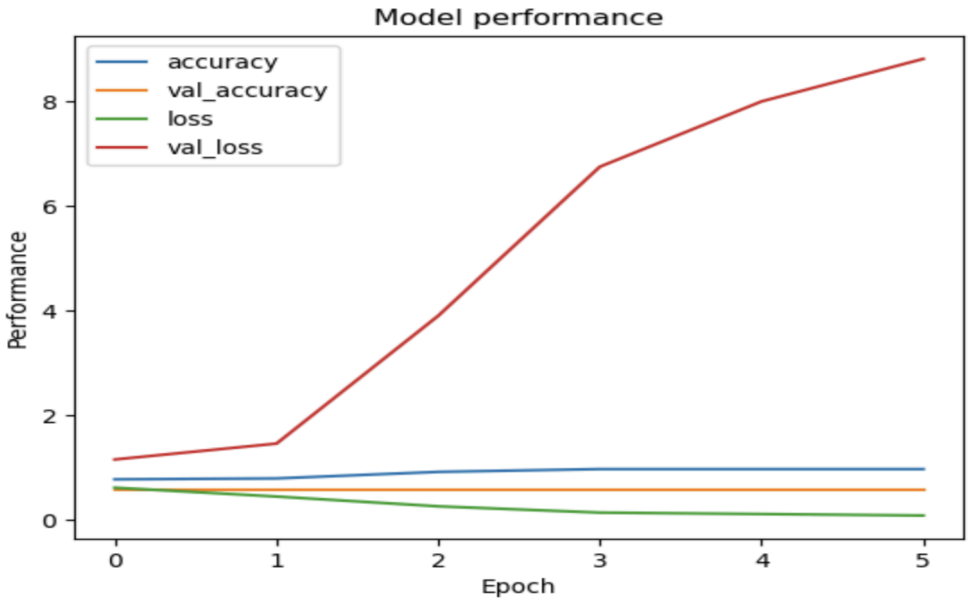

Figure 1:CNN model for binary classification with 20 epochs and 16 batch size

The above graph indicates the performance of accuracy validation, accuracy, loss and validation loss of the model with the most optimal performance on the training and test with accuracies of 76% and 77% respectively.

Multi Classification

Convolutional Neural Network is implemented. The CNN models for the multi-classification of the 12 species are presented below table. The first model with 20 epochs and 32 batch sizes along with padding in its layers, achieves a moderate training accuracy of 79%, indicating that it has learned to classify the training data adequately. However, the test accuracy is significantly lower (54%), indicating that the model does not generalize well to new data. The high test loss of 1.3 suggests that the model's predictions on the test set are inaccurate, which is a sign of overfitting. The model fits the training data well but does not perform adequately on the test data.

The second model with 20 epochs and 32 batch size outperforms the previous one, with a training accuracy of 84%, indicating improved learning from the training data. The test accuracy also rises to 62%, indicating better generalization to new data than the first model. The test loss of 0.46 is significantly lower than that of the first model, indicating that the test set predictions are more accurate indicating to the model's generalization ability has improved,

For the third model it can be said, that it achieves the highest training accuracy of 87%, indicating that it learns very effectively from the training data. Also, the test accuracy of 74% is also the highest of the three models, indicating strong generalization to new data. However, the test loss is relatively higher at 0.86, indicating some variability in the model's predictions. The reduction in batch size from 32 to 16 likely resulted in the model learning more effectively, leading to improved overall performance.

Models (Epochs, batch size)	**Train Acc**	**Test Acc**	**Test Loss**	**Train Loss**
model.CNN(20,32)with padding	80%	57%	1.3	0.65
model1.CNN(20, 32)	72%	55%	1.43	0.94
model1.CNN(20, 16)	87%	58%	1.28	0.50

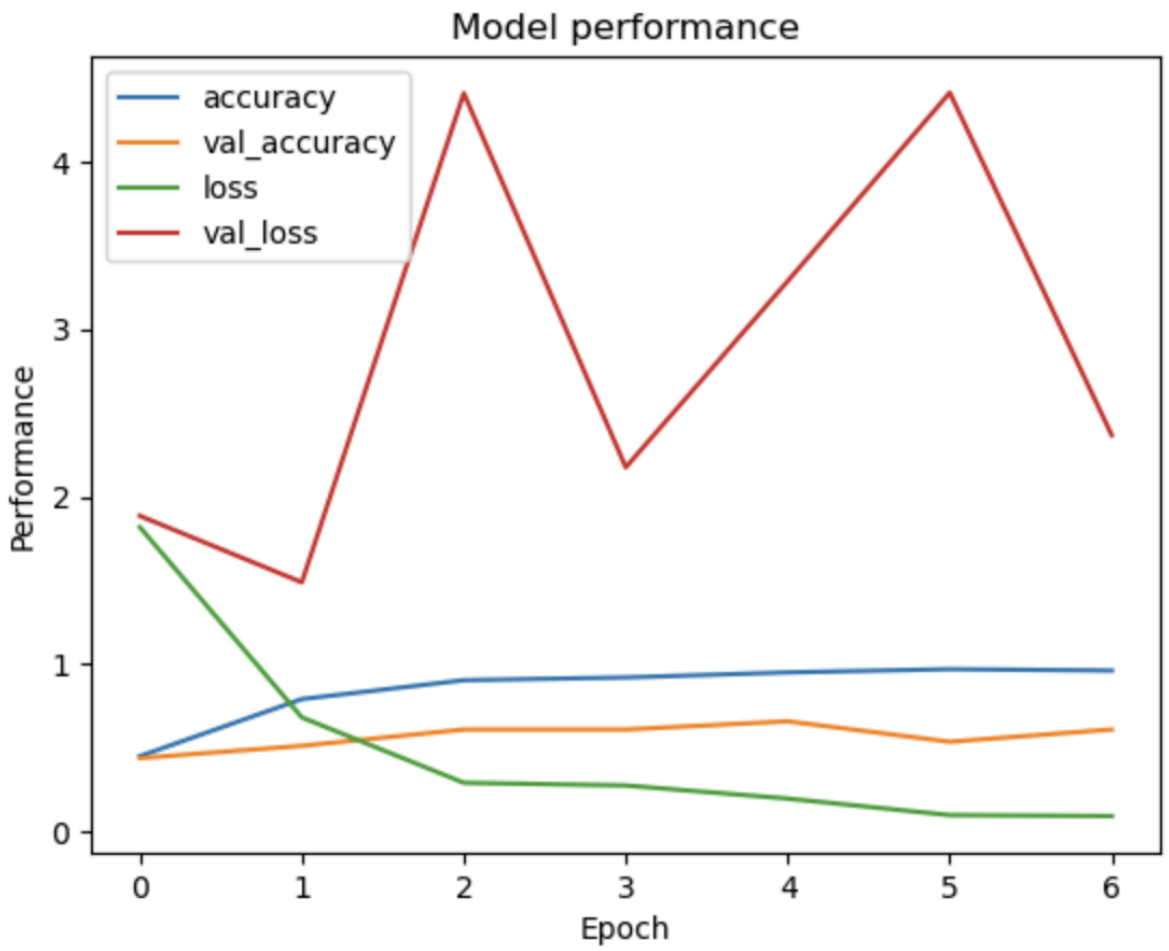

Figure 2:CNN model for multi-classification with 20 epochs and 16 batch size

The above graph demonstrates the performance of the model across epochs for the best-performing model for the multi-classification problem which gives 87% accuracy on the training set and 58% accuracy on the test set.

In the figure below the classification results of the best-performing classification model is presented and illustrating misclassifications.

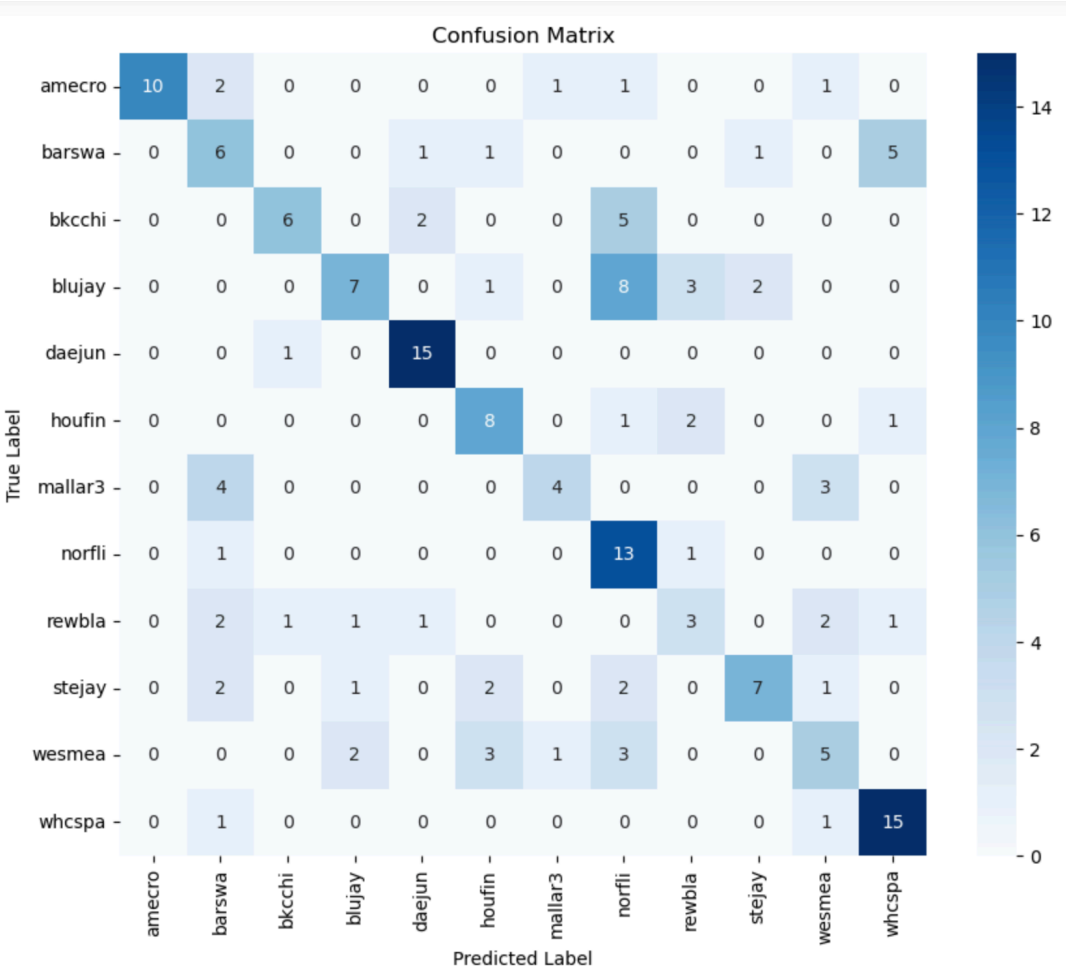

Figure 3:CNN model 10 epoch 512 batch size multi classification

External Data

With the implementation of the best performing model of the multi-classification that already has been trained and tuned on the 12 species and is now being tested on the processed spectrograms of the three unlabeled test audios, the probability of the 3 audios belonging to the possible species is presented below, for each of the models the results are given below. The models are conducted on the three audio files after their conversion to spectrograms.

Output of model one with 20 epochs and 32 batch size with padding in the layers:

Audio 3	wesmea	0.6165
Audio 3	blujay	0.1599
Audio 3	rewbla	0.0959
Audio 3	stejay	0.0766
Audio 3	amecro	0.0158

Audio 2	stejay	0.5361
Audio 2	wesmea	0.4639
Audio 2	norfli	0
Audio 2	blujay	0
Audio 2	rewbla	0

Audio 1	wesmea	0.7073
Audio 1	rewbla	0.1149
Audio 1	blujay	0.0712
Audio 1	stejay	0.0639
Audio 1	norfli	0.0205

Discussion

Several limitations were encountered during this research, one most significant one was the run-time to train the models. For models that were relatively more complex, having larger batch sizes and more epochs it took longer time, like 15 to 25 minutes to run therefore in order to tune and enhance the models it required better computational resources since, although the models performed well on the training data, there were indications of overfitting across all models even if the model was also performing quite well on the test as well.

Moreover when it comes to sound classification one major issue is to train a model that would direct the target sound in this case being able to accurately classify bird sounds from one another among species with similar spectrograms and species that have similar call patterns, or in the case of predicting the species on the unlabeled test data when there is noise and overlapping of various species. Due to challenges faced for classifying bird sounds whether it was a binary classification of the two species Blue jay and Steller jay and multi-classification for all 12 species the models resulted in low accuracy on the test set but higher accuracy on the training set indicating overfitting of the model and misclassification among the 12 species for multi-classification problem.

In order to better understand the performance of the models, for the binary classification the best-performing model with an accuracy of 76% on the train set and 77% on the test set is the one with 20 epochs and 16 batch sizes, from its performance over the number of epochs plot given as Figure1 it can be understood that The blue line being the training accuracy it can be seen that it increases slightly over and then being steady indicating that the model is learning and improving its performance on the training data. The orange line being the validation accuracy it can be said that it is almost constant indicating that the model is not generalizing well to the validation data. Giving the possibility of overfitting. Moreover, the training loss being the green line is decreasing steadily over each epoch, which is reasonable as the model optimizes its weights on the training data to reduce error. With the red line being the validation loss it can be seen that as the number of epochs increases it also increases which can be a point of concern suggesting overfitting.

As for the model best-performing model for the multi-classification the best performing model, is the model with 20 epochs and 16 batch sizes with accuracies of 87% on the train set and 58% on the test set, from its performance over the number of epochs plot given as Figure2 it can be understood that, The training accuracy (blue line) increases steadily over the epochs, stating that the model is learning the training data well. It can be stated that the validation accuracy orange line) remains somewhat constant and lower than the training accuracy, indicating that the model's generalizability to new data is limited. Moreover, The training loss (green line) decreases with each epoch, indicating that the model is minimizing the loss function on training data effectively. Lastly, the validation loss (red line) increases significantly and remains much higher than the training loss, indicating poor generalization and overfitting. There is a decrease and increase in the validation loss, implying that the model is sensitive to noise in the validation dataset. Resulting in model's poor performance in accurately classifying the 12 species leading to misclassification as demonstrated in Figure 3.

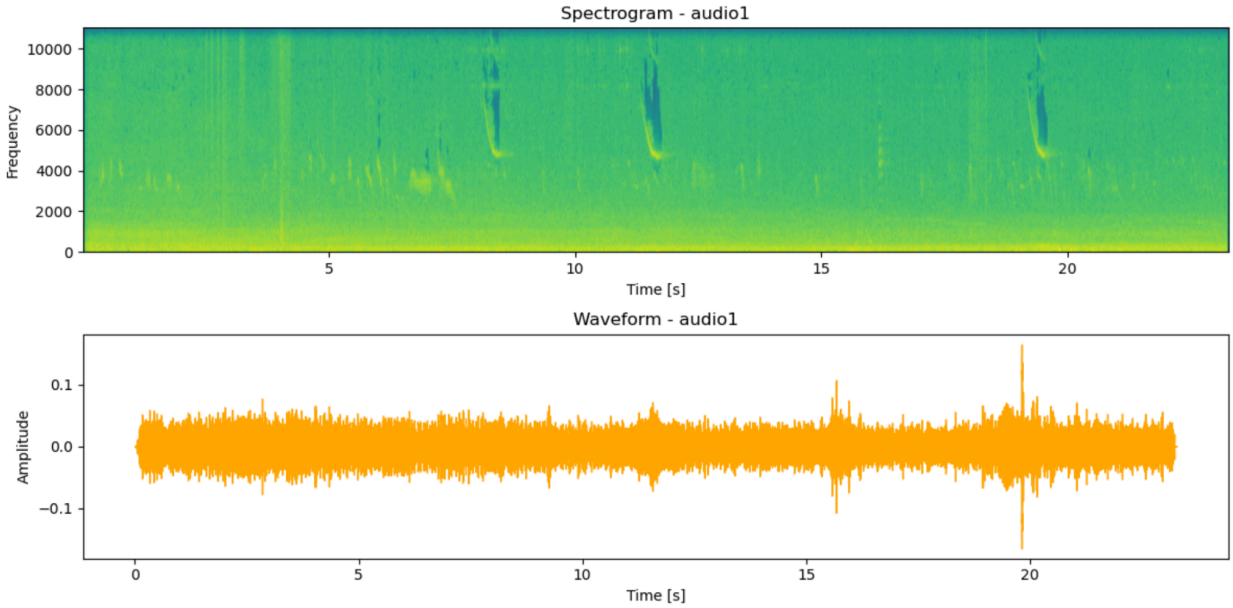

Figure 4:waveform and spectrogram of audio1

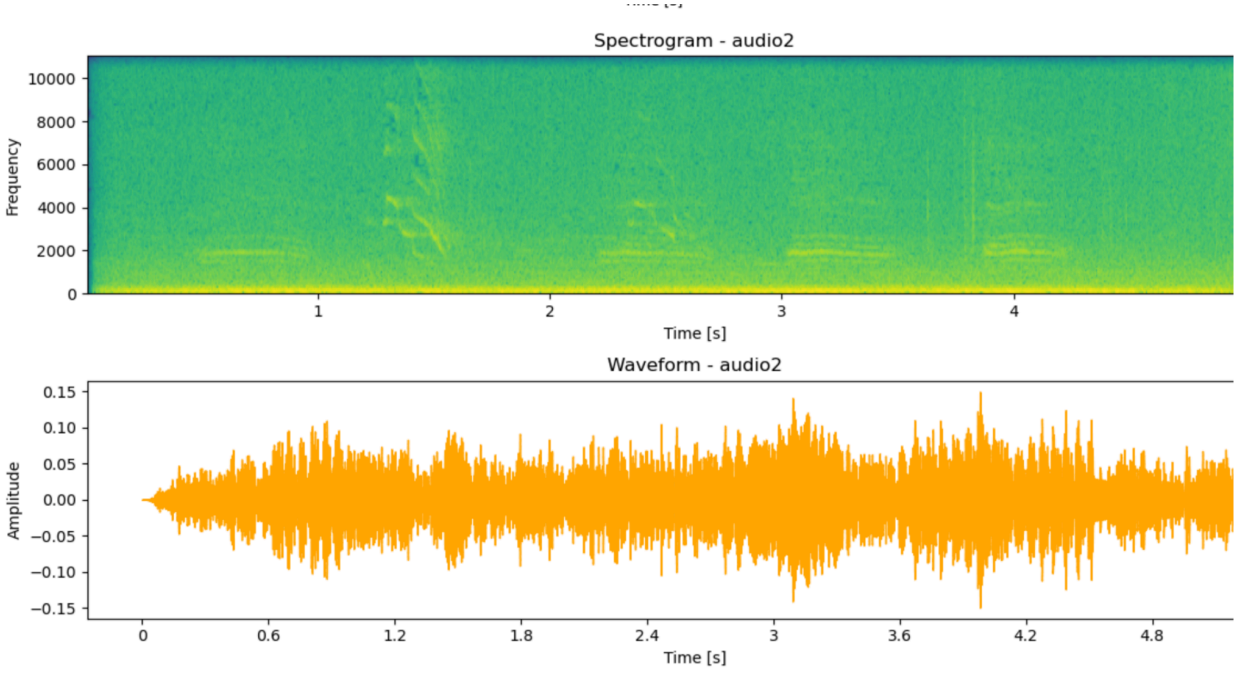

Figure 5:waveform and spectrogram of audio2

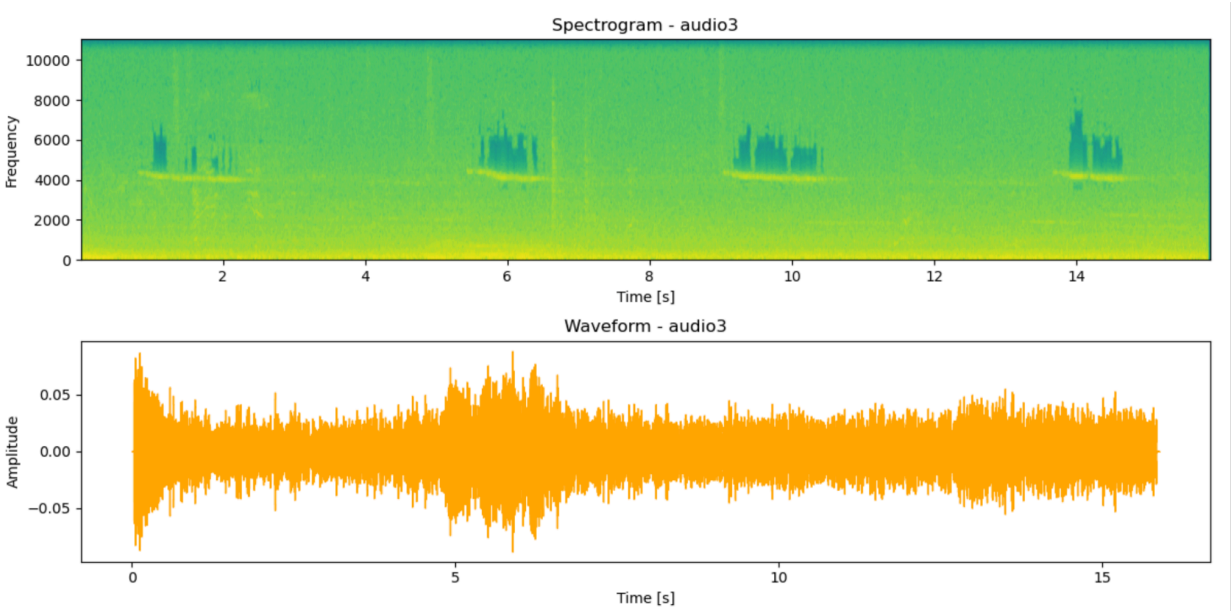

Figure 6:waveform and spectrogram of audio3

The frequency rates of each of the files can be seen in the above graphs. From the waveform and the display of the spectrograms for all three unlabeled audio files, we can understand that the second file is shorter resulting in the lowest number of spectrograms and therefore resulting in the model only being able to predict sounds for the two species of Steller jay and Western Meadowlark. And the first audio file with being 20 seconds would result in the highest number of spectrograms with the best performing model of the multi-classification being able to detect the sound of Western Meadowlark with the highest probability of 0.7 followed by Red-winged Blackbird, Blue Jay, StellerJay and Northern Flicker with low probabilities.

The multi-classification model was able to detect the sound of Western Meadowlark in the spectrograms of the third unlabeled audio set with the highest probability of 0.6 followed by Blue Jay, Steller Jay, Red-winged Blackbird and American crow with lower probabilities. Due to the overfitting nature of the model, it can be said during the training stage of the model on the train set the model had the ability to only learn the patterns of the above mentioned species as they are frequently detected through the three unlabeled audio file with favoring Western Meadowlark as it has been reported high probabilities for this species in all three audio set.

For sound classification problems transfer learning methods can also be implemented. Some pre-trained models that have been fine-tuned for the specific task of sound classification models are VGGish, Wavenet, and YAMNet. These models can be implemented on audio sets or spectrograms of the audio set. Furthermore, The reason why Neural Networks are the optimal choice for sound classification is that sound data are of complex nature, especially in the form of spectrograms can be highly dimensional making Neural Networks as the best fit.

Conclusion

To conclude in this research the implementation of Neural networks for sound classification of the most common 12 species in the Seattle area is assessed resulting in the binary classification of the two selected species of Blue jay and Steller jay, the best-performing model's accuracy 77% on the test and 76% on the train with indications of overfitting as for the multi classification of the 12 species sever overfitting for all models is detected with the best model performing 58% on the test set and 87% on the train. For the unlabeled test set it can be concluded that model mostly favored the Western Meadowlark species as it was predicted with high probability in all three audio set.

Furthermore, through this report the potential of implementing Neural Networks for birds sound classification in order to understand their ecology and habits can be highlighted. Continued research and development in this field show promise for improving the accuracy, efficiency, and applicability of machine learning approaches to biodiversity monitoring and preservation.

References

1. https://medium.com/@hasithsura/audio-classification-d37a82d6715 **" the liberosa library"**
2. https://medium.com/machinevision/overview-of-neural-networks-b86ce02ea3d1
3. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 **"NN"**
4. https://aws.amazon.com/what-is/recurrent-neural-network/#:~:text=A%20recurrent%20neural%20network%20(RNN,complex%20semantics%20and%20syntax%20rules. **"RNN"**
5. https://www.pnsn.org/spectrograms/what-is-a-spectrogram#:~:text=A%20spectrogram%20is%20a%20visual,energy%20levels%20vary%20over%20time. **"Spectrogram"**
6. https://www.v7labs.com/blog/neural-networks-activation-functions **"Activation function"**
7. https://www.kaggle.com/datasets/rohanrao/xeno-canto-bird-recordings-extended-a-m **"DataSet"**

Code Appendix

```
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
from matplotlib import image
from sklearn.metrics import confusion_matrix

import seaborn as sns

import librosa
import librosa.display
from tensorflow.keras.models import load_model

import os

from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelBinarizer, LabelEncoder
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score

from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.sequence import pad_sequences

from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import tensorflow as tf


from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten, Embedding, LSTM, SimpleRNN
from keras.applications import imagenet_utils
from keras.applications.imagenet_utils import preprocess_input
from keras.datasets import imdb

from scipy.sparse import csr_matrix
# Define CNN model for binary classification with dropout layers
model_dropout = Sequential([
    Input(shape=(256, 343, 1)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
```

```
    Dropout(0.25),  # Dropout layer with 25% dropout rate
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),  # Dropout layer with 25% dropout rate
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),  # Dropout layer with 25% dropout rate
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Dropout layer with 50% dropout rate
    Dense(1, activation='sigmoid')  # Using sigmoid activation for binary classification
])

# Compile and train the model
model_dropout.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
callbacks = [EarlyStopping(patience=5, monitor='val_loss', restore_best_weights=True)]
history = model_dropout.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.1,
callbacks=callbacks)

# Evaluate the model
test_loss, test_accuracy = model_dropout.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
train_loss, train_accuracy = model_dropout.evaluate(X_train, y_train)
print(f'Train Loss: {train_loss}, Train Accuracy: {train_accuracy}')
# Train the model

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.title('Model performance')
plt.xlabel('Epoch')
plt.ylabel('Performance')
plt.legend()
plt.show()



def create_spectrogram(audio_file_path, new_sample_rate=22050, min_loud_part_length=0.5,
bird_call_window_size=2,
                       n_fft=2048, hop_length=512):
    y, sr = librosa.load(audio_file_path, sr=new_sample_rate)
    energy = librosa.feature.rms(y=y, frame_length=n_fft, hop_length=hop_length)
    loud_parts = librosa.effects.split(y, top_db=20)
```

```
    spectrograms = []
    labels = []

    for start, end in loud_parts:
        segment_duration = (end - start) / sr
        if segment_duration > min_loud_part_length:
            for i in range(start, end, hop_length):
                bird_call_audio = y[i:i + int(bird_call_window_size * sr)]
                if len(bird_call_audio) == int(bird_call_window_size * sr):
                    spectrogram = librosa.feature.melspectrogram(y=bird_call_audio, sr=sr, n_fft=n_fft,
hop_length=hop_length, n_mels=256, fmax=8000)

                    if spectrogram.shape[1] < 343:
                        padding = np.zeros((256, 343 - spectrogram.shape[1]))
                        spectrogram = np.hstack((spectrogram, padding))
                    elif spectrogram.shape[1] > 343:
                        spectrogram = spectrogram[:, :343]

                    spectrogram = spectrogram[:, :, np.newaxis]

                    spectrograms.append(spectrogram)
                    labels.append(os.path.splitext(os.path.basename(audio_file_path))[0])

    return spectrograms, labels

folder_path = 'test_birds'

all_spectrograms = []
all_labels = []

for audio_file in os.listdir(folder_path):
    if audio_file.endswith('.mp3'):
        audio_file_path = os.path.join(folder_path, audio_file)
        spectrograms, labels = create_spectrogram(audio_file_path)
        all_spectrograms.extend(spectrograms)
        all_labels.extend(labels)

all_spectrograms = np.array(all_spectrograms)
all_labels = np.array(all_labels)
print(all_labels.shape)
print(all_spectrograms.shape)

model_path2 = 'cnn_multi_model3.keras'  # Update with the path to your saved model
```

```
model2 = load_model(model_path2)

model2.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
predictions = model2.predict(all_spectrograms)

class_to_species = {
    0: 'amecro', 1: 'barswa', 2: 'bkcchi', 3: 'blujay', 4: 'daejun', 5: 'houfin',
    6: 'mallar3', 7: 'norfli', 8: 'rewbla', 9: 'stejay', 10: 'wesmea', 11: 'whcspa'
}

file_names = [file.split('.')[0] for file in os.listdir(folder_path) if file.endswith('.mp3')]

# Aggregate predictions for each audio file
file_to_predictions = {file_name: [] for file_name in file_names}

for idx, (pred, label) in enumerate(zip(predictions, all_labels)):
    file_to_predictions[label].append(pred)

# Display the top five predictions for each test set
for file_name, species_probs_list in file_to_predictions.items():
    aggregated_probs = np.mean(species_probs_list, axis=0)  # Aggregate the predictions for the
file
    print("Original Label:", file_name)
    sorted_indices = np.argsort(aggregated_probs)[::-1][:5]
    for i, index in enumerate(sorted_indices):
        species = class_to_species[index]
        probability = aggregated_probs[index]
                print(f"Top {i+1} Predicted Bird Species: {species} - Class: {index} - Probability:
{probability:.4f}")
```