Bird Sound Classification using Deep Neural Networks

MOD002691 Final Project

SID: 1834432

Supervised by Mahdi Maktabdar Oghaz

Submitted in fulfillment of the requirements
for the degree BEng Computer Science

Faculty of Science and Engineering
Anglia Ruskin University
4/08/2023

Acknowledgement

This report work was supervised by Mahdi Maktabdar Oghaz, who has been a constant source of support. This work is also dedicated to my parents who have supported my studies throughout the years.

Table of Contents

List of Figures

List of Tables

Abstract

The classification of bird sounds has garnered considerable attention in recent times, primarily due to its wide-ranging applications in fields such as biodiversity monitoring, ornithology, and soundscape analysis. This paper provides a thorough examination of the utilisation of deep learning models for the automated classification of bird sounds. The proposed methodology leverages state-of-the-art deep learning architectures, including Convolutional Neural Networks (CNNs) and Long Term Short Term Memory(LSTM).Audio recordings from 10 commonly found bird species in Britain was used for training and evaluation the models and was sourced from the Xeno-Canto dataset.Mel-Frequency Cestrum(MFCCs) was used as the Feature extraction technique for the models.The best model achieved an accuracy of 99%.

Chapter 1

Introduction

1.1 Overview

Birds play a crucial role in maintaining ecosystem stability through their active involvement in pollen transfer and seed dispersal processes.Whelan et al. (2015).The identification of bird species based on their songs can be advantageous to both ecologists and ornithologists in the evaluation of biodiversity in conservatories and the examination of climate change.Birds communicate extensively using acoustics. They use only their sounds to communicate a variety of warnings about an approaching hazard and to recognise other birds in a flock.The identification of such acoustical activity, and its classification of different species of birds are important aspects in the recognition of a bird sounds.

But,over the course of the years there has been a global decline in bird populations(Bowler et al., 2019; Newton, 2004; Plard et al., 2020).This decline has increased the attention towards monitoring bird sounds continuously. The process of manually monitoring and analysing acoustic recordings has the potential to deliver precise results. Nevertheless, the considerable amount of time and effort involved in processing such recordings makes manual analysis impractical (Swiston and Mennill, 2009).The conventional methods used in bird sound surveillance are also expensive(Wimmer, Towsey, Roe and Williamson, 2013) since acoustic sensors could potentially store several gigabytes of compressed data each day without manual intervention(Brandes, 2008).Fortunately,nowadays the collection of Bird sounds on a Large-scale can be achieved using wireless sensor networks(Sugai et al., 2019; Wimmer, Towsey, Planitz, Williamson and Roe, 2013).

In recent years,there also has been notable developments in the fields of Machine Learning and Audio Signal Processing - leading to the emergence of Automated bird sound identification systems(Chandu et al., 2020; Cole et al., 2022). With the availability of Large datasets such as Xeno-Canto with over a million bird sounds, classification tasks involving bird sounds has seen an increase (Goëau et al., 2016; Incze et al., 2018; Zhang et al., 2019).The availability such extensive datasets has encouraged Bird Sound Challenges such as BirdCLEF and many others to gain popularity (Joly et al., 2014, 2015; Mesaros et al., 2017).

Various Machine Learning approaches have been explored to classify Bird sounds including unsupervised neural networks (Jancovic and Köküer, 2019; Michaud et al., 2023), Support vector machine (Tuncer et al..,2021,Fagerlund, 2007), decision trees (Neal et al., 2011; Lasseck, 2015), random forests (Bravo Sanchez et al.,2021) and Hidden Markov Model (Stastny et al.,(2013). At present, Convolutional Neural Network (CNN) (Permana et al., 2022; Xie et al., 2019) based architectures based on deep learning have demonstrated the highest level of the success in recognised bird call challenges that have become the state-of-the-art architectures.Recurrent Neural network(RNN) based architectures have recently started being explored in the field of Bird sound classification(Müller et al., 2018, Zhao et al., 2021,Krishnan et al.,2022),but it is yet to be applied extensively to Large scale bird sound classification tasks.

1.2 Research Aims

I. Use robust Feature extraction techniques to extract meaningful and relevant data from bird sound audio signals.

II. Build Deep Neural Network architectures and train the models to predict bird sounds.

III. Evaluate the models and select the model with the highest accuracy as the best model.

1.3 Research Objectives

I Pre-process data and minimise overfitting.

II Conduct rigorous evaluation experiments using appropriate metrics to evaluate and select the best model.

III Ultimately develop a robust classifier which provides accurate results in classification and is computationally fast.

1.4 Research Scope

The research focuses on Bird Sound Classification specifically.The scope of the study is defined as below:

I The sample of bird sounds used for training the models is limited to the most common species of birds found in UK.

II This study aims to comprehend and acquire knowledge pertaining to bird vocalisations, with a particular focus on the challenges associated with low accuracy and overfitting.

III State-of-the-art Deep learning architectures trained and evaluated.

1.5 Abridged Methodology

The methodology consisted of the steps 1)Pre-processing 2)Feature extraction 3)Model training 4)Evaluating model. The bird song dataset initially consisted of 88 bird classes but it was sampled down to 10 as the first step.Subsequently, features were extracted from the audio signals and then used for training the CNN and LSTM models.CNN model achieved a accuracy of 99% , outperforming the LSTM model, which attained an accuracy of 50%.

The work is organised as following.Chapter 2 is the reports findings after researching Bird Sound Classification techniques implemented by others,Chapter 3 is the Methadology which includes the steps taken to conduct the classification experiments,followed by Chapter 4 reporting the results from the experiments in the previous Chapter discussing the results and proposing solutions for future work.Finally, Chapter 5 is the Conclusion.

Chapter 2

Literature Review

The popularity of Automatic Bird Sound classification has increased recently with the introduction of Bird sound challenges like DCASE, BirdCLEF running yearly (Goëau et al., 2017; Mesaros et al., 2017). CNN has become the state-of-the-art architecture for such audio classification tasks as promising results have been produced across numerous studies(Abdallah et al., 2021; Mustaqeem and Kwon, 2019; Permana et al., 2022; Zhao et al., 2019).CNN-based Transfer Learning models like DenseNet(Liu et al., 2021,Kahl et al.), ResNET(Koh et al..,2019, Sankupellay et al., 2018) and VGG-16 (Noumida et al.,2021, Islam et al., 2019) have gained popularity for Bird sound classification tasks. RNN's also have been recently applied for classyfying bird sounds(Himawan et al., 2018,Liu et al., 2021).Bird Sound classification experiments all typically have a common feature extraction stage.

MFCCs are one of the most widely used feature extraction techniques for bird sound classification Anderson et al. (1996); Sundermeyer et al. (2012).In addition to CNNs, MFCCs have set the standard for feature extraction techniques because they produce very accurate results.(Rana et al.,2021;Şaşmaz et al., 2018).

Gupta et al(2021) conducted experiments using hybrid CNN models on the Cornell Bird Challenge' dataset which consisted of 284 bird species.All CNN based models ImageNET,VGG-16,ResNET were modelled on a single spectogram.They reported that as CNN complexity increase, the validation accuracies for the model increase as well.

RNNs capture temporal dependencies for sequence data.But, they suffer from vanishing gradients.As a solution to this,RNN units that implement a gating mechanism, such as a Long Short-Term Memory (LSTM) unit and gated recurrent unit (GRU) were introduced.Gupta et al compared different RNN models with hybrid CNN-LSTM and Transfer learning models with the best model achieving highest accuracy of 67%.They observed that increasing the size of the hidden state RNNs resulted in an increase in the validation accuracy but adding more layers to the RNN did not seem to improve the performance.The Hybrid LSTM model seemed to performed better than the VGG-16 model.

Support Vector Machine,another widely used model for bird sound classification tasks obtained 77.65% accuracy in predicting 46 kinds of birds in experiments conducted by Salamon et al.A feature dictionary was constructed by utilising logarithmic scale Mel spectrum species. By using SVM algorithm, accuracy of 93.96% was attained across a dataset comprising 43 distinct species of birds.

Mohanty et al.,(2020), used a Spiking Neural Network (SNN), a third-generation artificial neural network (ANN).Three types of feature extraction techniques were used: Mel Frequency Cepstral Coefficient (MFCC); Wavelet Transform;Permutation Pair Frequency Matrix (PPFM).The classification accuracy reached was 92%.

Most of the experiments on bird sound classification achieve high accuracies but with datasets with few bird classes/species. Large-scale classification tasks would require classifying a large number of classes and produce high accuracies.Also,the presence of background noise in large scale and results in low signal-to-noise ratio significantly affecting performance of Automated bird sound classification systems. (Stowell et al., 2017; Sumitani et al., 2019).Priyadarshani et al.,(2016) reported that denoising recordings using Wavelets can improve the model's classification performance.

Chapter 3

Methodology and Implementations

3.1 Overview

The Bird Song Classification models were build using Deep Learning Models - Convolutional Neural Networks(CNN) and Long Short-Term Memory(LSTM) for this project.The steps taken to build the model has been described in detail in this Section.The source code implementation for both the models has been added at the end of this paper in Appendix Section 5.The Jupyter notebook files with the names **BirdSoundClassifiedCNN.ipynb** and **BirdSoundClassifiedLSTM.ipynb** has been added along with the report for the CNN and LSTM implementations.The **dataset** folder contains the recordings for the **recordings** folder and the bird song labels in **birdsong_metadata.csv**.

3.2 Software and Dependencies

TensorFlow [1] a machine learning platform was used for building and training the machine learning models.Anaconda[2] and Jupyter notebook [3] are some of the tools used importing depencies and conducting the experiments.Python libraries like numpy[4],matplotlib[5], LibROSA[6] were some of the dependencies used for carry out Machine Learning and Audio Signal processing tasks.

3.3 Research Framework

The CNN and LSTM based Deep Learning approaches in this paper aims to classify bird sounds into their 10 classes.The step-by-step representation of how the project was con-

[1] `https://www.tensorflow.org/`
[2] `https://www.anaconda.com/`
[3] `https://jupyter.org/`
[4] `https://numpy.org/`
[5] `https://matplotlib.org/`
[6] `https://librosa.org/doc/latest/index.html`

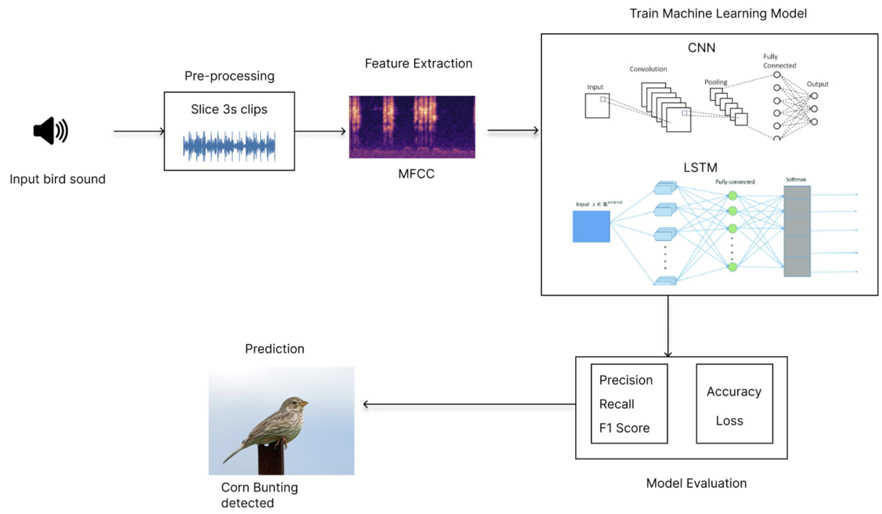

Figure 3.1: Research Framework for Bird Sound Classification

ducted has been shown in Fig 3.1.The first step in Section 3.3.2 involved Pre-processing the dataset of 88 bird types to get the top 10 bird types and the audio files,the MFCC features were then extracted from the audio recordings in Section3.3.3 followed by training Deep Learning models based on the extracted features in Sections 3.3.4 and 3.3.5. Finally, the metrics to measure model performances were defined in Section 3.3.6.

3.3.1 Dataset

There are numerous online resources available that provide access to datasets containing bird sounds. One notable example is Xeno Canto [7], a platform that provides a wide range of freely accessible bird vocalisations contributed by users. Since 2014, it has made a noteworthy contribution to BirdCLEF and its influence on the recognition of bird sounds.

The dataset used for this project is the British bird sound[8] , a subset gathered from the Xeno Canto dataset.It contains bird sound recordings collected by 68 separate bird enthusiasts across 88 birds commonly heard in the United Kingdom.The graphical distribution of the dataset with the bird names on the x-axis and the number of instance for the top 30 birds on the y-axis is presented in 3.2.

3.3.2 Pre-Processing

The dataset exhibited class imbalances i.e. it had different numbers of instances across classes.The imbalance in the dataset could result in Overfitting ultimatly Chawla (2010). Downsampling is an effective way to deal with imbalanced datasets - it lowers the number of instances of certain classes with large numbers of instances to match ones with lower

[7] https://xeno-canto.org/
[8] https://www.kaggle.com/datasets/rtatman/british-birdsong-dataset

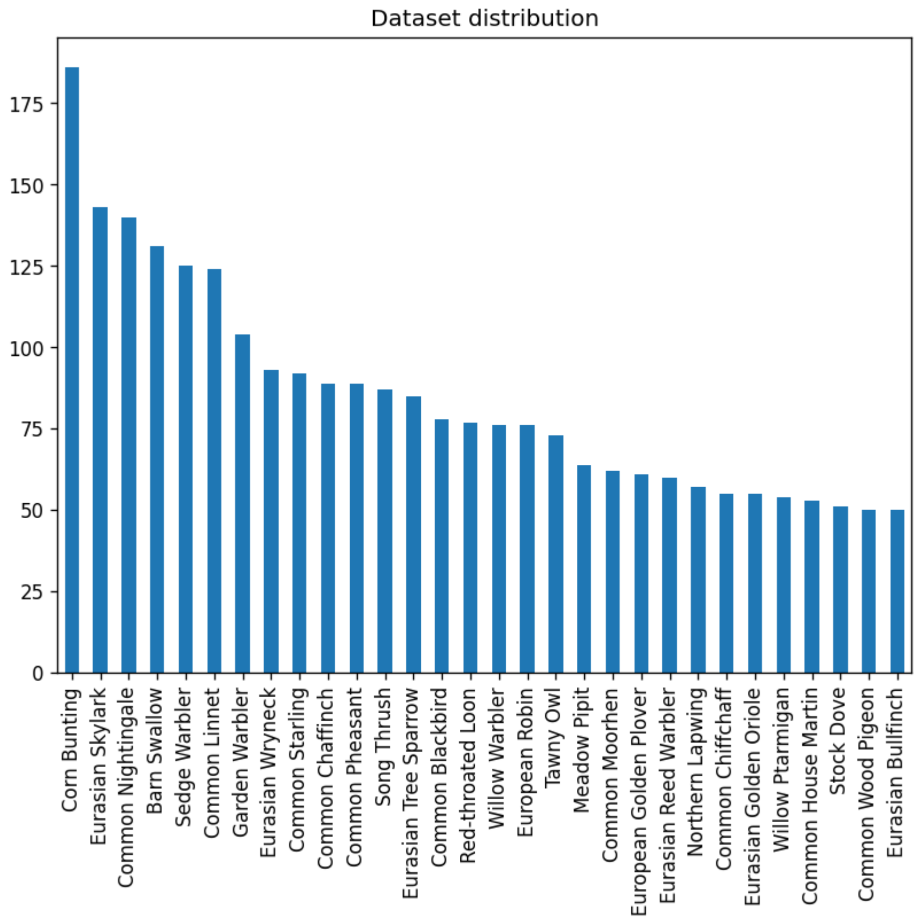

Figure 3.2: Dataset distribution showing the top 30 bird sounds

numbers of instances (Huang et al., 2006).

The implementation of this method is straightforward and has demonstrated superior efficacy compared to an alternative approaches for reducing imbalances (Drummond et al., 2003).Nevertheless,this approach suffers from a significant drawback, useful instances are discarded during the training process.

Another commonly used approach for handling imbalanced datasets is upsampling, which duplicates instances from classes with a low number of instances in order to match classes with high instances (Huang et al., 2006).It holds an advantage of not discarding instances that may contain useful data. Nevertheless, this approach has previously demonstrated a tendency to result in overfitting and can be computationally demanding when dealing with a substantial number of instances and classes. Kumar and Sheshadri (2012).

Downsampling was selected as a solution to tackle the dataset imbalance problem for this project.This approach was selected since the original dataset had few instances of bird sounds,88 total types of birds.This is comparatively low compared to tasks like the Bird-CLEF 2017,a challenge that involved classifying 1500 classes of bird sounds (Goëau et al., 2017).Classes with too fewer instances than 89 were removed.The number of instances of some of the classes were relatively very low compared such as Marsh Warbler, Rock Dove

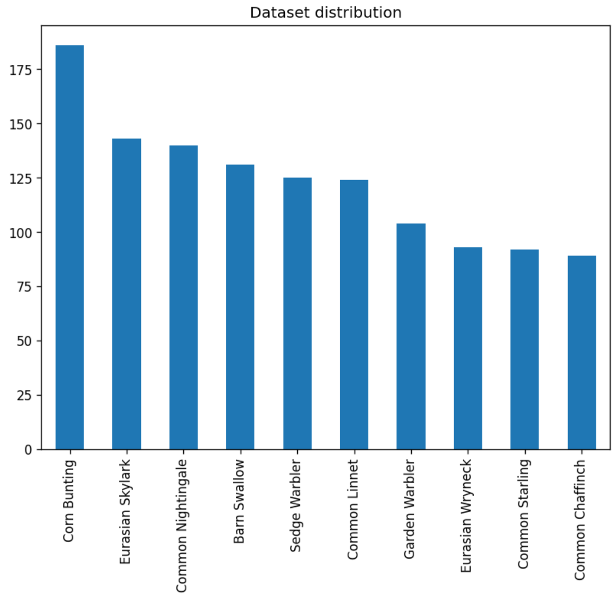

Figure 3.3: Graph shows the 10 bird classes with the most number of instances in the dataset

Table 3.1: Bird sound classes with the lowest instances

Bird name	Number of Instances
Common Cuckoo	16
Great Spotted Woodpecker	15
Lesser Whitethroat	15
Common Redpoll	12
Dunlin	12
Rock Dove	10
Marsh Warbler	9

and others;ranging between 9 and 16 as seen in Table 3.1.The top ten classes with the highest number of instances were considered.Ultimately,the final dataset for training/testing consisted of 10 classes and has been depicted in the graph in Fig 3.3.The x-axis depicts the bird name classes and the y-axis is the number of instances per class.The dataset was split into training and testing with a 80% and 20% percentage respectively.Training set then consisted of 981 samples and test set consisted of 246 samples.

3.3.3 Feature Extraction

The audio data from the bird sound samples cannot be directly processed by Machine Learning models.The conversion of audio signal into visual representations is a necessary step for classification tasks.The representation of audio is in the form of an audio signal, which can be mathematically represented as a function of frequency and time.Mel-spectrogram is a visual representation of such audio signals utilising Mel-scale,making them effectively capture the distribution of energy in a manner that corresponds to human auditory system's frequency perception.Mel-spectrograms utilise the Mel-scale to transform the frequency axis into a scale that is more perceptually significant, as opposed to utilising a linear frequency scale.The Mel-scale utilised leverges it's ability to approximate the human auditory system's response to various frequencies.It can be computed given frequency in Hz & perceptual frequency scale in Mels. Mel-frequency scaling is linear frequency spacing for frequencies up to 1000Hz,logarithmic spacing is used exceeding 1000Hz.The formula to convert frequency(Hz) into $\text{mel}_{frequency}$(Mels) and can be computed using Equation 3.1.

$$mel_{frequency} = 2595 \log_{10}(\frac{frequency}{700} + 1) \tag{3.1}$$

Mel-frequency cepstral coefficients (MFCC) is a feature representation technique has widely been used in the field of Audio Signal processing (Abduh et al., 2020; Mushtaq and Su, 2020; Şaşmaz and Tek, 2018).It is obtained by applying a transformation to the Mel-spectrogram.The Mel-spectrogram is utilised as an intermediary representation, with the MFCC being derived through the application of additional transformations to the Mel spectrogram. MFCC's are first obtained by applying the Short-Time Fourier Transform (STFT) to the audio signal to obtain a power spectrum.The STFT algorithm partitions the audio signal into discrete and overlapping frames.Mel filter-banks are subsequently applied to individual frames.Mel filter-banks can be characterised as a collection of triangular filters, each filter associates with a distinct frequency range based on the Mel-scale.These filters are placed on the linear frequency scale and weighted according to the Mel-scale and sum of the energy within each Mel filter's frequency range then results in the Mel-spectrogram.

The sound samples had varying lengths and in order to make them uniform the samples were sliced into smaller 5 second clips. The audio clips were sampled at 22.05 kHz. LibROSA was used to extract the MFCCs from audio signals. Mel-spectrograms were computed with 2048 window size for each STFT frame, 512 samples between successive frames(hop size),128 mel-bank filters. **librosa.power_to_db()** function was utilised to convert the mel-spectrogram to decibel (dB) scale. It applies the formula as shown in Eq 3.2,p is the reference power. By default, ref is set to the maximum power in the spectrogram.MFCCs are then computed for the CNN architecture with 40 MFCC coefficients.The LibROSA **specshow** function was used to display the waveplots,subsequent Mel-spectogram and MFCC for each of the 10 birds in Fig 3.4.

$$dB = 10log_{10}(\frac{melspectogram}{p}) \tag{3.2}$$

The MFCCs for the LSTM model were computed with 255-point window size, 512 hop size, 13 MFCC coefficients and 5 Mel-bands.Similar to CNN,**specshow** function was used to display the waveplots and MFCC representations for each of the 10 birds in Fig 3.5.

The output MFCCs were two-dimensional arrays representing the computed MFCCs.The rows correspond to the different MFCC coefficients and the columns correspond to the frames.The extracted MFCC features are then reshaped into a Two-Dimensional matrix to be suitable for CNN input.The rows of the matrix represent frames and columns represent features.The resulting feature representation,a sequence of frames containing Mel-spectrogram features along with other information served as input to the CNN and LSTM based models.

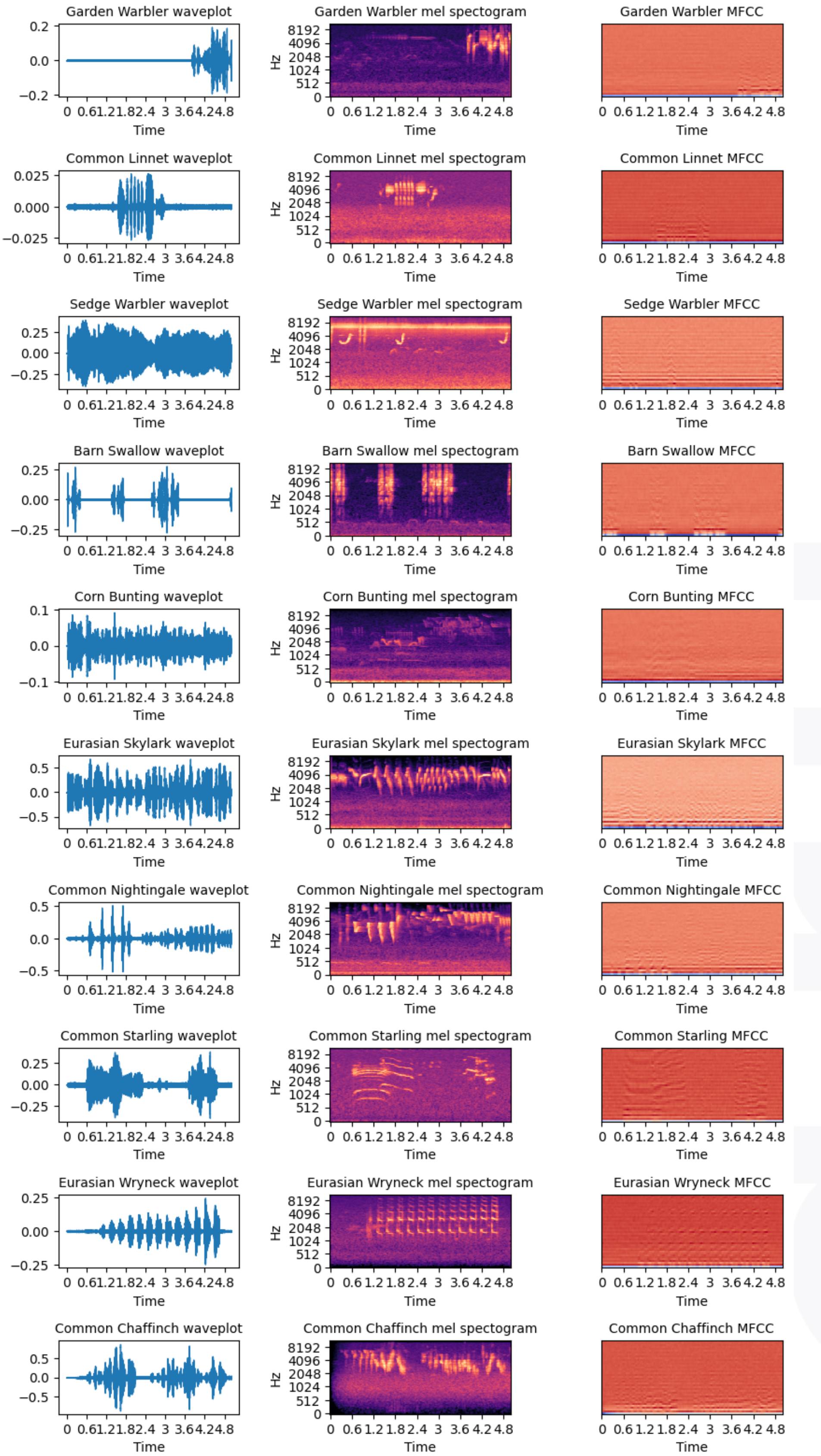

Figure 3.4: The waveplots,Mel-spectograms and MFCCs representations displayed for 10 bird sounds the MFCC-CNN architecture

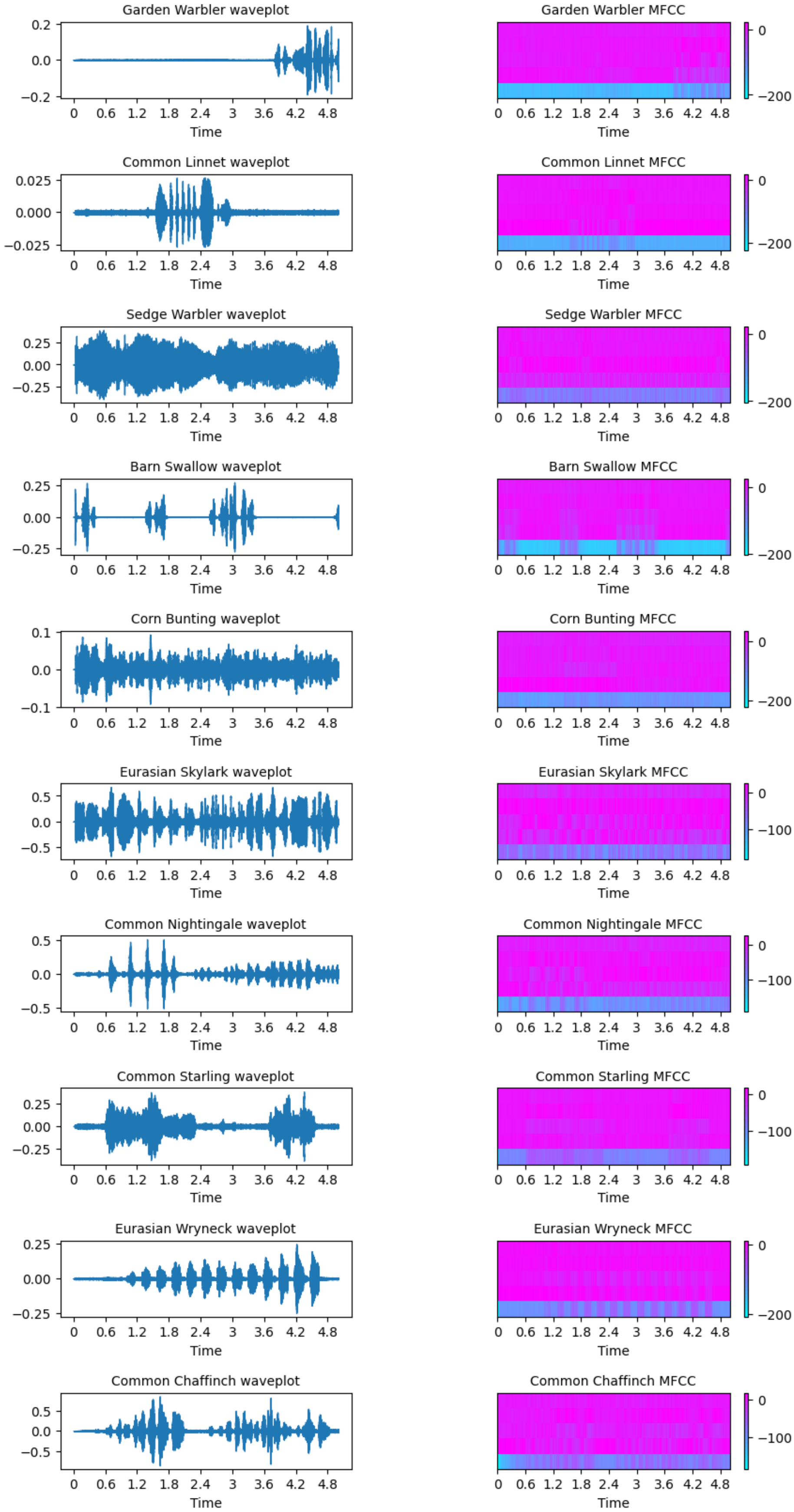

Figure 3.5: The waveplots and MFCCs representations displayed for 10 bird sounds the LSTM-CNN architecture

3.3.4 Convolutional Neural Network

Convolutional Neural Networks(CNN) have been widely used for bird sound classification tasks (Hidayat et al., 2021; Kahl et al., 2017*b*; Koh et al., 2019*b*).Convolutional Neural Networks (CNNs) have been engineered to effectively analyse and extract significant features from data by leveraging the power of convolutional layers. The utilisation of Convolutional Neural Networks (CNNs) in audio signal processing offers significant advantages due to their ability to proficiently capture and model local dependencies and hierarchies present in the data. Stacking a series of Convolutional layers can help the neural network acquire progressively complex features. The hierarchical representation enables the model to effectively identify significant patterns and attributes within the audio signals, thereby resulting in enhanced performance in tasks such as audio classification.

A conventional CNN structure typically comprises three primary layers: the Fully Connected Layer,Convolution Layer and Pooling Layer as shown in Fig 4.4.In addition to the previously mentioned main layers, a CNN can also use optional layers to address overfitting and minimise training time, such as a Batch Normalisation Layer.The Dropout Layer prevents overfitting by at random set a portion of the data's weights to zero.

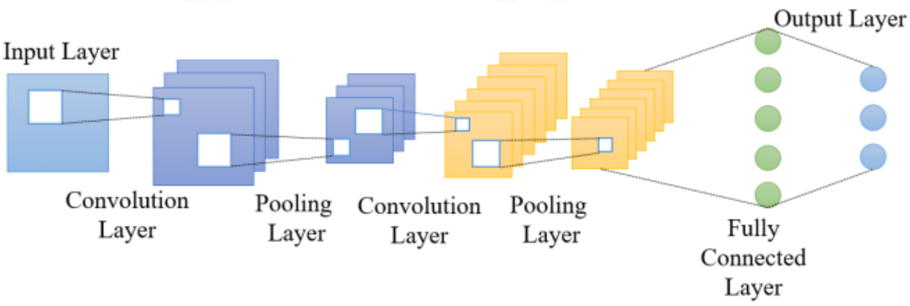

Figure 3.6: CNN architecture

Convolutional Layer

The Convolutional layer within a CNN designed for audio processing, utilising MFCC input, executes convolution operations on the input tensor. Each convolutional kernel is selectively applied to a designated region of the input. Upon reaching a convolution layer, the layer proceeds to convolve each filter across the spatial dimensionality of the data, resulting in the generation of a two-dimensional feature map. The verification of neuron outputs, which are connected to localised regions of the input, can be achieved by utilising the convolution layer. This involves calculating the scalar product between the weights of the neurons and the corresponding area of the input volume.

An activation function is applied to each convolutional layer.The Rectified Linear Unit (ReLU) is widely recognised as one of the most prominent activation functions utilised in the field of Deep Learning. The rectified linear unit, often abbreviated as ReLU, seeks to apply an elementwise activation function to the output of the activation produced by the preceding layer. The activation is computed by applying a threshold of 0 to the input in the ReLU function.ReLU function outputs a value of 0 if the input is less than 0, and outputs the original data if the input is greater than or equal to 0.It is mathematically represented as Eq 3.3.

$$f(x) = max(0, x) \tag{3.3}$$

Pooling Layer

Pooling Layer,is the subsequent to Convolutional Layer.The pooling operation is applied separately to each channel of the input data.The Layer aims to systematically reduce the dimensionality of the data, thereby decreasing the number of parameters and the complexity of the model and increasing efficiency.This approach helps address the issue of overfitting and reduces computational costs.

Adding **GlobalAveragePooling2D()** after the convolutional layers is another common approach in CNN architectures.The process involves the reduction of spatial dimensions in the feature maps by computing the average values within each channel. This operation produces a vector of fixed lengthThe spatial dimensions undergo a process of collapse, leading to the formation of a one-dimensional tensor.This enables the model to prioritise the most significant features.The subsequent layers that are fully connected can then proceed to process the pooled features in order to perform classification.

Fully Connected Layer

Following the application of multiple convolutional and pooling layers, the extracted features are subsequently transmitted to fully connected layers. These layers collect high-level features and make subsequent predictions by utilising the extracted information.The features extracted from preceding layers are consolidated and utilised for classification purposes.

The Fully Connected Layer is comprised of a network of interconnected neurons. Every individual neuron in the preceding layer is connected to a distinct neuron in the subsequent layer.The output of the final Fully Connected Layer is subsequently fed into an activation function, which determines the class labels.The activation function type of the output Dense layer is softmax, which generates a probability distribution for each class.The softmax function is utilised to generate a probability distribution for the n output classes.The function can be represented as Equation 3.4.

$$\alpha(c)_j = \frac{e_j^c}{\sum_{k=1}^{K} e^{c_k}} \tag{3.4}$$

Designing the CNN

The 2D CNN designed for this project has multiple Convolusional,Pooling and Dropout layers stacked and one Fully Connected layer at the end. The Convolution and Pooling kernels are 2x2 in dimension.The Dropout values for the first three CNN structures used a Dropout value of 0.2 and fourth 0.5.The number of the filters in the first,second layers,third and fourth layers were 16,32,64 and 128 respectively.The Fully connected Dense layer has 10 neurons corresponding to 10 classes followed by the softmax activation.

The CNN model was trained by optimizing the categorical cross-entropy between predictions and targets with adaptive moment estimation (Adam) . This network was trained using Adam optimizer with a learning rate of 10^{-4} Categorical cross entropy was utilized as the loss function.The model is trained for 50 epochs in batch sizes of is 32 samples.The overall architecture of the CNN is shown in **Table 3.2.**

Table 3.2: CNN Architecture for Bird sound Classification

Layer	Type	Kernel Size	Number of Filters	Activation
Conv1	Convolutional Max Pooling Dropout	2×2 2×2 0.2	16	ReLU
Conv2	Convolutional Max Pooling Dropout	2×2 2×2 0.2	32	ReLU
Conv3	Convolutional Max Pooling Dropout	2×2 2×2 0.2	64	ReLU
Conv4	Convolutional Max Pooling Dropout Global Average Pooling	2×2 2×2 0.5	128	ReLU
Fully Connected	Dense	-	512	Softmax

3.3.5 Long Short-Term Memory

The Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that was developed by Hochreiter and Schmidhuber in 1997.The architecture demonstrates notable efficacy in processing sequential and time-dependent data(Zhao et al., 2019).

The LSTM architecture is comprised of a singular component, referred to as the memory unit or LSTM unit. The LSTM unit comprises four distinct feedforward neural networks. Each of these neural networks is composed of an input layer and an output layer. In each of these neural networks, there exists a connection between the input neurons and all of the output neurons. Consequently, the LSTM unit is composed of four fully connected layers. Three out of the four feedforward neural networks are tasked with the responsibility of information selection. The three gates in question are commonly referred to as the forget gate, the input gate, and the output gate. These three gates serve to execute the three standard operations related to memory management: the removal of data from memory (referred to as the forget gate), the addition of new data into memory (known as the input gate), and the utilisation of data already stored in memory (designated as the output gate), with the purpose of selectively retaining or discarding information over a period of time.The fourth neural network, referred to as the candidate memory, is employed for the purpose of generating fresh candidate data that can subsequently be incorporated into the memory.A single LSTM unit has been depicted in Figure 3.7

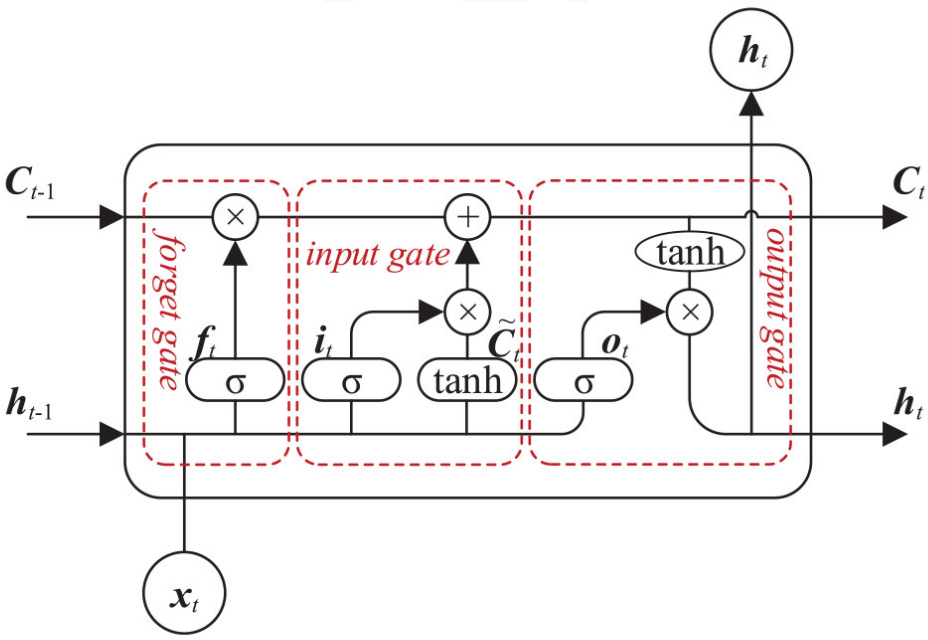

Figure 3.7: Architecture of a LSTM unit(Zhao et al., 2020)

The primary element of a Long Short-Term Memory (LSTM) is the memory cell, which facilitates the network's ability to retain and retrieve information over extended periods. The capacity to preserve and propagate information over an extended period is crucial for capturing long-term dependencies in auditory signals. The utilisation of memory cells and gating mechanisms in LSTM models enables the selective updating and utilisation of information from previous time steps, while effectively filtering out irrelevant information.

The typical result of the LSTM layer(s) is a series of concealed states that symbolise the processed sequential data. In order to generate final predictions or conduct classification, it is common practise to flatten the output obtained from the LSTM layers into a one-dimensional vector and subsequently input it into one or multiple dense layers. The subsequent dense layers that follow the LSTM layers can be conceptualised as a conventional feed-forward neural network. In a dense layer, every neuron is linked to each neuron in the preceding layer, resulting in a fully connected architecture.ReLU activation function on Dense Layers helps the network learn complex relationships and can mitigate the vanishing gradient problem by allowing for faster and more efficient training(Li et al., 2018; Sundermeyer et al., 2012).

Designing the LSTM

The LSTM architecture was designed with 64 LSTM units along with L2 regularization to the layers' kernel weights with a regularization strength of 0.01 as the first layer. This was taken as a measure to prevent overfitting by penalizing large weight in the input matrix(Abdallah et al., 2021).The LSTM Layer was followed by a Dropout layer with a value of 0.2, which randomly sets the weights of a portion of the data to zero to prevent overfitting.The subsequent layer used was a Dense Fully connected Layer with 128 neurons followed by another Dropout layer with same value of 0.2 as previous Dropout layer. The final layer consisted of a Fully Connected Dense Layer with a size of 10, which outputs a probability for each class is 10 neurons with a softmax activation function to predict 10 bird classes.

The model was trained with loss function as categorical cross-entropy and applied Adam optimizer with a learning rate of 10^{-4} The network was trained in batch sizes of 32 samples over 100 epochs.

3.3.6 Evaluation metrics

In this project,the classification models were evaluated using precision,recall,F1 score and accuracy.For each class, the False Positive (FP) occurs when the other classes are falsely identified as being the positive (current) class. False Negative (FN) occurs when the current class is falsely identified as being one of the other class. Using these metrics the Precision, Recall and F1 score can be further calculated.

Accuracy is a measure of the overall correctness of the model's predictions. It calculates the proportion of correctly classified instances (both True Positives and True Negatives) out of the total number of instances.However, it may not accurately reflect the true performance in datasets that exhibit class imbalance, where one class significantly outweighs the other.For such cases, precision and recall are useful metrics to consider.

Precision is a metric applied in the context of binary classification to measure the proportion of True Positive(TP) predictions or the correctly predicted positives out of all positive predictions. In alternative terms, precision provides an indication of the model's ability to minimise the occurrence of False Positives(FP).Precision can be computed using **Equation 3.5**.

Recall measures the proportion of True Positive(TP) predictions out of all actual positive instances in the dataset. Assessing the model's ability to accurately detect positive cases is instrumental in gaining insights into its performance.Recall can be computed using **Equation 3.6**.

In simple terms,Recall refers to the proportion of instances that are both relevant and accurately identified. Precision refers to the proportion of instances that are accurately classified as belonging to the correct class(Buckland and Gey, 1994).

The F1 score is another widely used performance metric.The metric integrates precision and recall metrics into a single measure, thereby offering a well-balanced assessment of the model's effectiveness.The F1 score can be computed using **Equation 3.7**.

The confusion matrix is a tabular representation that effectively illustrates the performance evaluation of a classification algorithm.It represents the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).It supports in comprehending the performance of the model and is useful in calculating precision and recall.The diagonal entries in the matrix show the number of times the classes were correctly identified.A Confusion matrix representation has been shown in **Table 3.3.**

$$Precision = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{TruePositive}{TotalActualPositive} \tag{3.5}$$

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{TruePositive}{TotalPredictedPositive} \tag{3.6}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.7}$$

	Actual Class	
Predicted Class	**True Positive (TP)**	**False Positive (FP)**
	False Negative (FN)	**True Negative (TN)**

Table 3.3: Confusion Matrix

Chapter 4

Results and Discussion

This section presents the results obtained from CNN and LSTM experiments for Bird Sound Classification and analysis of the results.The original dataset consisted of 88 bird types.Since the dataset had an imbalance, Downsampling technique was applied to prevent Overfitting.Dataset was downsampled to 10 classes of bird ultimately.The training and test sets for both the models were trained with of 981 and tested on 246 samples each.The results are evaluated based on the Evaluation metrics described in Section 3.3.6 of this paper.

4.1 CNN Results

The results from the CNN model has been presented in Table 4.1, Figures 4.1 and 4.2.The CNN model was trained in 32 batches and 50 epochs.The model reached an accuracy of 99%.The accuracy and loss curves in Figures 4.1 and 4.2 show no signs of Overfitting i.e the loss and accuracy curves are in close proximity, showing signs of a "fit model".After 12 epochs of training the model, the training and validation accuracies recorded were 95% and 96% and had similar difference in values throughout.

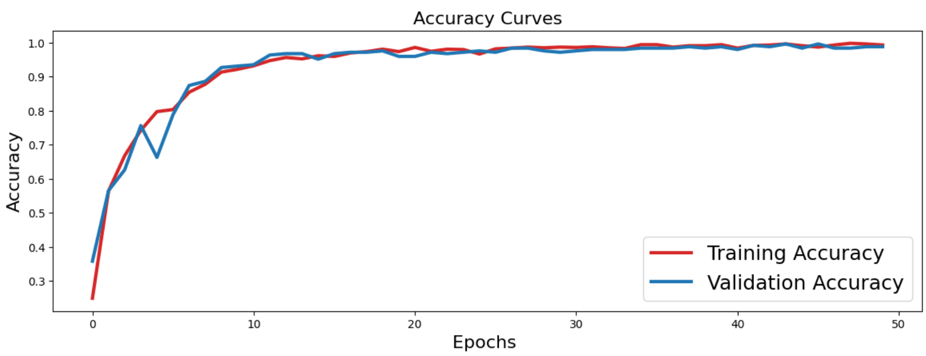

Figure 4.1: Accuracy curve CNN

Table 4.1: Evaluating CNN model

	Precision	Recall	F1-score	Support
Barn Swallow	0.94	1.00	0.97	29
Common Chaffinch	1.00	0.96	0.98	23
Common Linnet	1.00	1.00	1.00	21
Common Nightingale	1.00	1.00	1.00	33
Common Starling	1.00	1.00	1.00	17
Corn Bunting	1.00	1.00	1.00	32
Eurasian Skylark	1.00	0.97	0.98	33
Eurasian Wryneck	1.00	1.00	1.00	18
Garden Warbler	0.94	0.94	0.94	17
Sedge Warbler	1.00	1.00	1.00	23
Accuracy			0.99	246
Macro avg	0.99	0.99	0.99	246
Weighted avg	0.99	0.99	0.99	246

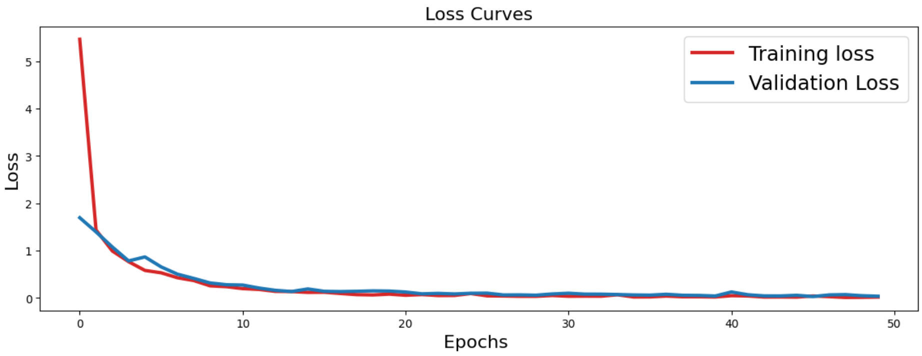

Figure 4.2: Loss curve CNN

4.2 LSTM Results

The results from the LSTM model has been presented in Table 4.1,Figures 4.1 and 4.2.The model was trained in 32 batches and 100 epochs.The model reached a maximum accuracy of 50%. After just 12 epochs the LSTM training accuracy reached 92% but the validation accuracy was 38%. The loss curve shown in Figures 4.1 and 4.2 clearly show signs of overfitting,evidenced by the substantial disparity between the training and validation curves. This observation suggests that the model has been effectively trained, but it is unlikely to exhibit satisfactory performance when presented with new, unseen data. Adding more training examples using Upsampling could resolve the issue of Overfitting (Huang et al., 2006).

Table 4.2: Evaluating LSTM model

	Precision	Recall	F1-score	Support
Barn Swallow	0.63	0.49	0.55	35
Common Chaffinch	0.19	0.17	0.18	23
Common Linnet	0.52	0.64	0.57	22
Common Nightingale	0.40	0.29	0.33	42
Common Starling	0.64	0.88	0.74	8
Corn Bunting	0.42	0.50	0.46	36
Eurasian Skylark	0.27	0.50	0.35	14
Eurasian Wryneck	1.00	0.76	0.86	29
Garden Warbler	0.38	0.50	0.43	10
Sedge Warbler	0.69	0.67	0.68	27
Accuracy			0.50	246
Macro avg	0.51	0.54	0.52	246
Weighted avg	0.53	0.50	0.51	246

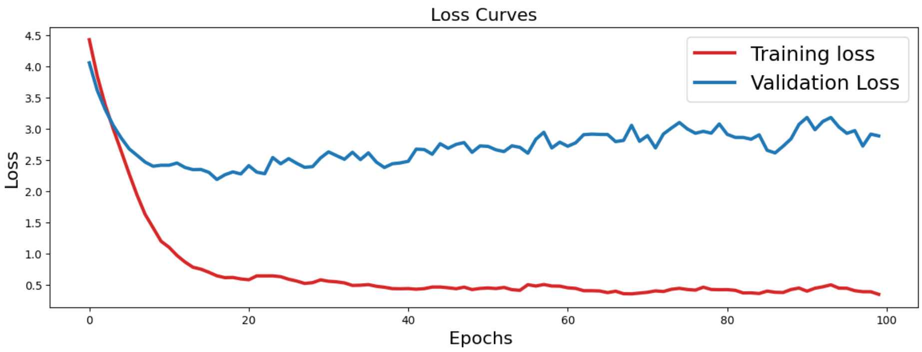

Figure 4.3: Accuracy curve LSTM

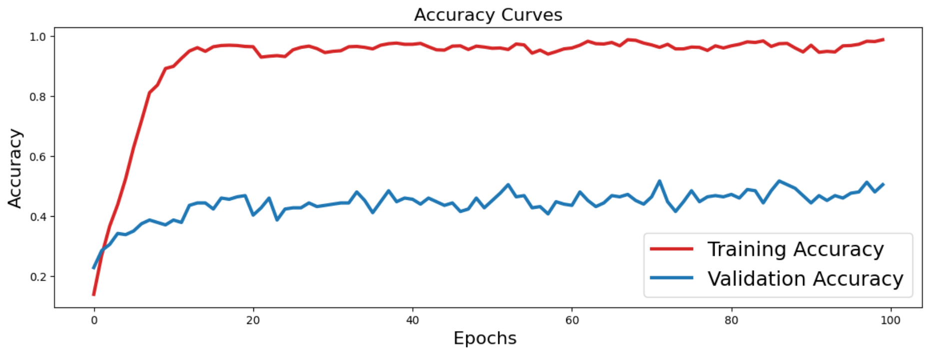

Figure 4.4: Loss curve LSTM

4.3 Evaluating Results

Compared to the LSTM model,the CNN loss and accuracy curves were in close proximity,indicating there was no over-fitting phenomenon and revealing the good fitting degree of the network.The LSTM model achieved an accuracy of 50%, while the CNN model achieved an accuracy of 99%.The CNN model outperformed the LSTM model in terms of overall accuracy.The model evaluation results with metrics precision, recall,F1-scores for each 10 bird classes was recorded in Tables 4.1 and 4.2.It was observed that both models achieved different classification performances for distinct bird classes.The support values in the Tables represent the number of instances of samples each class was assigned from the test set.It provides an understanding of the distribution of the classes and is useful for interpreting the performance of the model in the context of the class imbalance.The support values prove to be useful when interpreting results from a confusion matrix.

The confusion matrices were plotted as shown in Figures 4.5 and 4.6 for the CNN and LSTM models respectively.The x-axis of the matrix represents the Predicted bird class labels and the y-axis represents the actual bird class labels.The findings obtained from the CNN model demonstrate a strong alignment along the diagonal of the matrix,indicating a high degree of accurate identification for the majority of bird classes. This suggests that the CNN model is likely to produce precise results when tested on unseen data. Only a few samples were confused with other classes;1 Barn Swallow sample was confused with Common Chaffinch,1 Barn Swallow sample with Garden Warbler and 1 Garden Warbler with Eurasian skylark.

The LSTM model encountered a problem wherein several classes were consistently misidentified, as evidenced by the confusion matrix.It was observed that 13% Common Starling, 15% Sedge Warble, 24% samples of Eurasian Wryneck and 36% of Common Linnet were confused with other classes from the test set.But, 83% and 72 % samples of Common Chaffinch and Common Nightingale were heavily confused with other classes.Common Chaffinch had a recall of 19% and precision 17%, which was the lowest of all bird varieties.The F1-score was 18%,also lowest across other classes. The rest of the 4 classes were confused by percentage 50%.Compared to the CNN model,the performance is still satisfactory.The measures while conducting the experiments such as Downsampling, adding Dropout layers and adding Regularisation layer to the LSTM layer still produced 50% accuracy. The results obtained from the CNN model are comparable to the state-of-the-art architectures.Study conducted by Xie et al. produced an accuracy of 86.31% classifying 43 bird classes using CNN-Based architecture;Another study conducted by Permana et al. achieved 96.45% accuracy classifying local birds in Indonesia using CNN.According to the Table 4.1, the precision and recall scores all ranged between 94% to 100%.The CNN exhibited superior performance compared to the LSTM model across all evaluated metrics, establishing it as the more efficient model of the two.

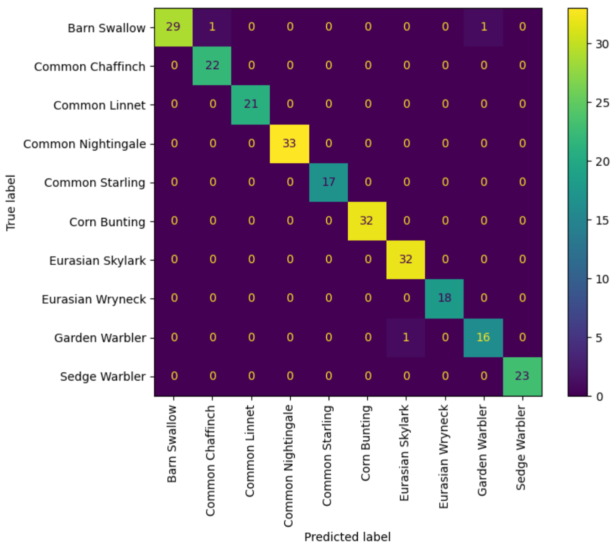

Figure 4.5: Confusion Matrix CNN

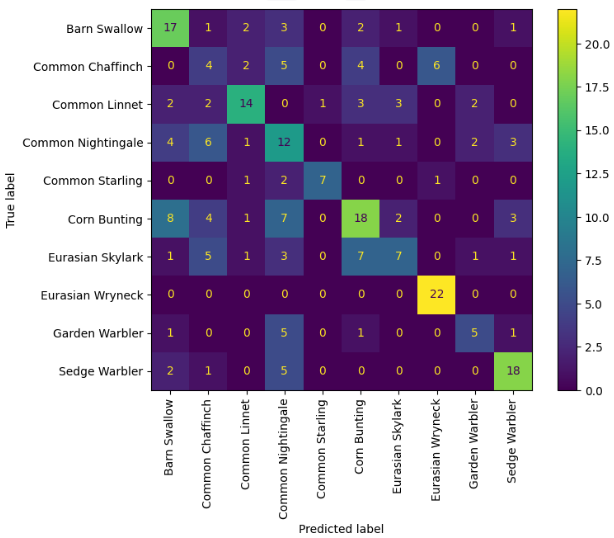

Figure 4.6: Confusion Matrix LSTM

4.4 Future Work

The CNN model demonstrated highly encouraging outcomes, exhibiting a remarkable accuracy rate of 99%.However, there exists potential for substantial enhancement of the performance of the LSTM model, which attained an accuracy of 50% and a weighted F1-score of 51%.

LSTM model was overfitted,training and validation accuracies had huge disparities during training. The Pre-processing technique could significantly improved applying Cross-validation to tackle overfitting.Cross-validation is a technique that guarantees the inclusion of every instance in the dataset for testing, validation, and training purposes within each fold;if a specific fold exhibits a higher accuracy, it does not impact the overall accuracy if the remaining folds demonstrate lower accuracies(Bengio and Grandvalet, 2003).The original dataset comprised 88 distinct bird types, which contrasts with the Large scale-bird sound classification tasks that have the potential to make substantial contributions to the field of bird sound classification.(Kahl et al., 2017*a*,*b*; Qian et al., 2015).Furthermore, the inclusion of additional training and testing samples for classes that have fewer instances has the potential to greatly improve the accuracy.

Chapter 5

Conclusion

In this project, the performances of two Deep Neural Networks was compared; namely CNN and LSTM.88 bird classes of commonly found birds in Britain was obtained from the Xeno-Canto dataset. The dataset observed to have few instances of some bird classes, a decision was made to discard classes with fewer instances than 89, to prevent overfitting at the Pre-Processing stage to get optimal results at the end. The CNN model outperformed the LSTM model achieving precision, recall,F1 score and overall accuracy all more than 94%. Comparatively, the LSTM model was overfitted after training for only 12 epochs, despite of the measures taken to prevent it including adding regularisation and dropout layers. Most of the original aims and objectives set in the beginning of the experiment were achieved. In the code implementation, "Corn Bunting" was predicted correctly for both the models after training and evaluation stages. Perhaps, conducting more experiments on the LSTM model and tuning hyperparameters further could potentially have shown better accuracy results. The LSTM model achieved an accuracy of 50% predicting the results of some bird classes like Barn swallow being confused with other classes 50% of the time. The CNN model had very few bird classes being for the others. Despite of 50% accuracy of the LSTM model, the confusion matrix revealed very high rates of confusion - 83% and 72 % bird classes of Common Chaffinch and Common Nightingale were confused with other classes.There is scope of improvement using techniques like Upsampling which involves increasing the number of instances of classes with few instances.Another improvement could be using Cross-Validation which is splitting the dataset into training,tesing and validation –increasing the chances of all instances of the dataset being included.

The implementation of the both the CNN and LSTM models included feature extraction using MFCCs, training and evaluating the models successfully. It was a great new learning experience dealing with audio signal processing and Machine learning.It strengthened my understanding of Machine learning architectures and Python. The project was up to satisfaction as majority of the aims were met. Future work aims to build a more efficient LSTM classification network. The project only included 10 classes of birds, it would be ideal to next implement using a larger dataset with more classes. Also, to introduce methods to reduce overfitting for large datasets and data augmention techniques to reduce noise from samples.A fusion of LSTM with CNN based frameworks like ResNET could also be considered.

Bibliography

Abdallah, M., An Le Khac, N., Jahromi, H. and Delia Jurcut, A. (2021), A hybrid cnn-lstm based approach for anomaly detection systems in sdns, *in* 'Proceedings of the 16th International Conference on Availability, Reliability and Security', pp. 1–7.

Abduh, Z., Nehary, E. A., Wahed, M. A. and Kadah, Y. M. (2020), 'Classification of heart sounds using fractional fourier transform based mel-frequency spectral coefficients and traditional classifiers', *Biomedical Signal Processing and Control* **57**, 101788.

Anderson, S. E., Dave, A. S. and Margoliash, D. (1996), 'Template-based automatic recognition of birdsong syllables from continuous recordings', *The Journal of the Acoustical Society of America* **100**(2), 1209–1219.

Bengio, Y. and Grandvalet, Y. (2003), 'No unbiased estimator of the variance of k-fold cross-validation', *Advances in Neural Information Processing Systems* **16**.

Bowler, D. E., Heldbjerg, H., Fox, A. D., de Jong, M. and Böhning-Gaese, K. (2019), 'Long-term declines of european insectivorous bird populations and potential causes', *Conservation Biology* **33**(5), 1120–1130.

Brandes, T. S. (2008), 'Automated sound recording and analysis techniques for bird surveys and conservation', *Bird Conservation International* **18**(S1), S163–S173.

Buckland, M. and Gey, F. (1994), 'The relationship between recall and precision', *Journal of the American society for information science* **45**(1), 12–19.

Chandu, B., Munikoti, A., Murthy, K. S., Murthy, G. and Nagaraj, C. (2020), Automated bird species identification using audio signal processing and neural networks, *in* '2020 International Conference on Artificial Intelligence and Signal Processing (AISP)', IEEE, pp. 1–5.

Chawla, N. V. (2010), 'Data mining for imbalanced datasets: An overview', *Data mining and knowledge discovery handbook* pp. 875–886.

Cole, J. S., Michel, N. L., Emerson, S. A. and Siegel, R. B. (2022), 'Automated bird sound classifications of long-duration recordings produce occupancy model outputs similar to manually annotated data', *Ornithological Applications* **124**(2), duac003.

Drummond, C., Holte, R. C. et al. (2003), C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, *in* ‘Workshop on learning from imbalanced datasets II’, Vol. 11, pp. 1–8.

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R. and Joly, A. (2016), Lifeclef bird identification task 2016: The arrival of deep learning, *in* ‘CLEF: Conference and Labs of the Evaluation Forum’, number 1609, pp. 440–449.

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R. and Joly, A. (2017), Lifeclef bird identification task 2017, *in* ‘CLEF: Conference and Labs of the Evaluation Forum’, number 1866.

Hidayat, A. A., Cenggoro, T. W. and Pardamean, B. (2021), ‘Convolutional neural networks for scops owl sound classification’, *Procedia Computer Science* **179**, 81–87.

Hochreiter, S. and Schmidhuber, J. (1997), ‘Long short-term memory’, *Neural computation* **9**(8), 1735–1780.

Huang, K., Yang, H., King, I. and Lyu, M. R. (2006), ‘Imbalanced learning with a biased minimax probability machine’, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **36**(4), 913–923.

Incze, A., Jancsó, H.-B., Szilágyi, Z., Farkas, A. and Sulyok, C. (2018), Bird sound recognition using a convolutional neural network, *in* ‘2018 IEEE 16th international symposium on intelligent systems and informatics (SISY)’, IEEE, pp. 000295–000300.

Jancovic, P. and Köküer, M. (2019), ‘Bird species recognition using unsupervised modeling of individual vocalization elements’, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**(5), 932–947.

Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.-P., Planque, R., Rauber, A., Fisher, R. and Müller, H. (2014), Lifeclef 2014: multimedia life species identification challenges, *in* ‘Information Access Evaluation. Multilinguality, Multimodality, and Interaction: 5th International Conference of the CLEF Initiative, CLEF 2014, Sheffield, UK, September 15-18, 2014. Proceedings 5’, Springer, pp. 229–249.

Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.-P., Planqué, R., Rauber, A., Palazzo, S., Fisher, B. et al. (2015), Lifeclef 2015: multimedia life species identification challenges, *in* ‘Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association, CLEF’15, Toulouse, France, September 8-11, 2015, Proceedings 6’, Springer, pp. 462–483.

Kahl, S., Wilhelm-Stein, T., Hussein, H., Klinck, H., Kowerko, D., Ritter, M. and Eibl, M. (2017*a*), ‘Large-scale bird sound classification using convolutional neural networks.’, *CLEF (working notes)* **1866**.

Kahl, S., Wilhelm-Stein, T., Hussein, H., Klinck, H., Kowerko, D., Ritter, M. and Eibl,

M. (2017*b*), 'Large-scale bird sound classification using convolutional neural networks.', *CLEF (working notes)* **1866**.

Koh, C.-Y., Chang, J.-Y., Tai, C.-L., Huang, D.-Y., Hsieh, H.-H. and Liu, Y.-W. (2019*a*), Bird sound classification using convolutional neural networks., *in* 'CLEF (Working Notes)'.

Koh, C.-Y., Chang, J.-Y., Tai, C.-L., Huang, D.-Y., Hsieh, H.-H. and Liu, Y.-W. (2019*b*), Bird sound classification using convolutional neural networks., *in* 'Clef (working notes)'.

Kumar, M. and Sheshadri, H. (2012), 'On the classification of imbalanced datasets', *International Journal of Computer Applications* **44**(8), 1–7.

Li, S., Li, W., Cook, C., Zhu, C. and Gao, Y. (2018), Independently recurrent neural network (indrnn): Building a longer and deeper rnn, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 5457–5466.

Mesaros, A., Heittola, T., Diment, A., Elizalde, B., Shah, A., Vincent, E., Raj, B. and Virtanen, T. (2017), Dcase 2017 challenge setup: Tasks, datasets and baseline system, *in* 'DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events'.

Michaud, F., Sueur, J., Le Cesne, M. and Haupert, S. (2023), 'Unsupervised classification to improve the quality of a bird song recording dataset', *Ecological Informatics* **74**, 101952. **URL:** *https://www.sciencedirect.com/science/article/pii/S1574954122004022*

Mushtaq, Z. and Su, S.-F. (2020), 'Environmental sound classification using a regularized deep convolutional neural network with data augmentation', *Applied Acoustics* **167**, 107389.

Mustaqeem and Kwon, S. (2019), 'A cnn-assisted enhanced audio signal processing for speech emotion recognition', *Sensors* **20**(1), 183.

Newton, I. (2004), 'The recent declines of farmland bird populations in britain: an appraisal of causal factors and conservation actions', *Ibis* **146**(4), 579–600.

Permana, S. D. H., Saputra, G., Arifitama, B., Caesarendra, W., Rahim, R. et al. (2022), 'Classification of bird sounds as an early warning method of forest fires using convolutional neural network (cnn) algorithm', *Journal of King Saud University-Computer and Information Sciences* **34**(7), 4345–4357.

Plard, F., Arlettaz, R., Jacot, A. and Schaub, M. (2020), 'Disentangling the spatial and temporal causes of decline in a bird population', *Ecology and evolution* **10**(14), 6906–6918.

Qian, K., Zhang, Z., Ringeval, F. and Schuller, B. (2015), Bird sounds classification by large scale acoustic features and extreme learning machine, *in* '2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)', pp. 1317–1321.

Şaşmaz, E. and Tek, F. B. (2018), Animal sound classification using a convolutional neural network, *in* '2018 3rd International Conference on Computer Science and Engineering (UBMK)', IEEE, pp. 625–629.

Stowell, D., Benetos, E. and Gill, L. F. (2017), 'On-bird sound recordings: automatic acoustic recognition of activities and contexts', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **25**(6), 1193–1206.

Sugai, L. S. M., Silva, T. S. F., Ribeiro Jr, J. W. and Llusia, D. (2019), 'Terrestrial passive acoustic monitoring: review and perspectives', *BioScience* **69**(1), 15–25.

Sumitani, S., Suzuki, R., Chiba, N., Matsubayashi, S., Arita, T., Nakadai, K. and Okuno, H. G. (2019), An integrated framework for field recording, localization, classification and annotation of birdsongs using robot audition techniques—harkbird 2.0, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 8246–8250.

Sundermeyer, M., Schlüter, R. and Ney, H. (2012), Lstm neural networks for language modeling, *in* 'Thirteenth annual conference of the international speech communication association'.

Swiston, K. A. and Mennill, D. J. (2009), 'Comparison of manual and automated methods for identifying target sounds in audio recordings of pileated, pale-billed, and putative ivory-billed woodpeckers', *Journal of Field Ornithology* **80**(1), 42–50.

Tuncer, T., Akbal, E. and Dogan, S. (2021), 'Multileveled ternary pattern and iterative relieff based bird sound classification', *Applied Acoustics* **176**, 107866.
URL: *https://www.sciencedirect.com/science/article/pii/S0003682X20309713*

Whelan, C. J., Şekercioğlu, Ç. H. and Wenny, D. G. (2015), 'Why birds matter: from economic ornithology to ecosystem services', *Journal of Ornithology* **156**, 227–238.

Wimmer, J., Towsey, M., Planitz, B., Williamson, I. and Roe, P. (2013), 'Analysing environmental acoustic data through collaboration and automation', *Future Generation Computer Systems* **29**(2), 560–568.

Wimmer, J., Towsey, M., Roe, P. and Williamson, I. (2013), 'Sampling environmental acoustic recordings to determine bird species richness', *Ecological Applications* **23**(6), 1419–1428.

Xie, J., Hu, K., Zhu, M., Yu, J. and Zhu, Q. (2019), 'Investigation of different cnn-based models for improved bird sound classification', *IEEE Access* **7**, 175353–175361.

Zhang, X., Chen, A., Zhou, G., Zhang, Z., Huang, X. and Qiang, X. (2019), 'Spectrogram-frame linear network and continuous frame sequence for bird sound classification', *Ecological Informatics* **54**, 101009.

Zhao, J., Mao, X. and Chen, L. (2019), 'Speech emotion recognition using deep 1d & 2d cnn lstm networks', *Biomedical signal processing and control* **47**, 312–323.

Lu, L., Li, S.Z. and Zhang, H.J., 2001, August. Content-based audio segmentation using support vector machines. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*(pp. 749-752). IEEE.Vancouver

Fagerlund, S., 2007. Bird species recognition using support vector machines.*EURASIP Journal on Advances in Signal Processing*,2007, pp.1-8.

Neal, L., Briggs, F., Raich, R. and Fern, X.Z., 2011. Time-frequency segmentation of bird song in noisy acoustic environments. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE.

Lasseck, M., n.d. *Improved Automatic Bird Identification through Decision Tree based Feature Selection and Bagging.*

Bravo Sanchez, F.J., Hossain, M.R., English, N.B. and Moore, S.T., 2021. *Bioacoustic classification of avian calls from raw sound waveforms with an open-source deep learning architecture.* Scientific Reports, [online] 11(1), p.15733.

Himawan, I., Towsey, M. and Roe, P., 2018. 3d convolution recurrent neural networks for bird sound detection. In Proceedings of the 3rd Workshop on Detection and Classification of Acoustic Scenes and Events (pp. 1-4).*Detection and Classification of Acoustic Scenes and Events.*

Mukherjee, R., Banerjee, D., Dey, K. and Ganguly, N., 2018. *Convolutional recurrent neural network based bird audio detection.DCASE challenge.*

Müller, L. and Marti, M., 2018, September. *Bird Sound Classification using a Bidirectional LSTM.* In CLEF (Working Notes).Vancouver

Krishnan, S., Khandelwal, P. and Garg, R., 2022. *Bird Species Classification: One Step at a Time.*CLEF Working Notes.Vancouver

Liu, H., Liu, C., Zhao, T. and Liu, Y., 2021, November. *Bird song classification based on improved bi-lstm-densenet network.* In 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE) (pp. 152-155). IEEE.

Gupta, G., Kshirsagar, M., Zhong, M., Gholami, S. and Ferres, J.L., 2021. *Comparing recurrent convolutional neural networks for large scale bird species classification.* Scientific reports, 11(1), p.17085.

Mohanty, R., Mallik, B.K. and Solanki, S.S., 2020. *Automatic bird species recognition system using neural network based on spike.* Applied Acoustics, 161, p.107177.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. *Learning representations by back-propagating errors. nature*, 323(6088), pp.533-536.

Williams, R.J. and Zipser, D., 1989. *A learning algorithm for continually running fully recurrent neural networks. Neural computation*, 1(2), pp.270-280.

Bengio, Y., Simard, P. and Frasconi, P., 1994. *Learning long-term dependencies with gradient descent is difficult.* IEEE transactions on neural networks, 5(2), pp.157-166.

Hochreiter, S. and Schmidhuber, J., 1997. *Long short-term memory. Neural computation*, 9(8), pp.1735-1780.

Himawan, I., Towsey, M. and Roe, P., 2018. *3d convolution recurrent neural networks for bird sound detection.* In Proceedings of the 3rd Workshop on Detection and Classification of Acoustic Scenes and Events (pp. 1-4). Detection and Classification of Acoustic Scenes and Events.

Tuncer, T., Akbal, E. and Dogan, S., 2021. *Multileveled ternary pattern and iterative ReliefF based bird sound classification.* Applied Acoustics, 176, p.107866.

Brooker, S.A., Stephens, P.A., Whittingham, M.J. and Willis, S.G., 2020. A*utomated detection and classification of birdsong: An ensemble approach.* Ecological Indicators, 117, p.106609.

Priyadarshani, N., Marsland, S. and Castro, I., 2018. *Automated birdsong recognition in complex acoustic environments: a review.* Journal of Avian Biology, https://doi.org/10.1111/jav.01447.

Priyadarshani, N., Marsland, S., Castro, I. and Punchihewa, A., 2016. *Birdsong Denoising Using Wavelets.* PLOS ONE, 11(1). https://doi.org/10.1371/journal.pone.0146790.

Stastny, J., Munk, M. and Juranek, L., 2018. *Automatic bird species recognition based on birds vocalization.* Eurasip Journal on Audio, Speech, and Music Processing, 2018(1). https://doi.org/10.1186/s13636-018-0143-7.

Stastny, J., Skorpil, V. and Fejfar, J., 2013. *Audio data classification by means of new algorithms.* In: 2013 36th International Conference on Telecommunications and Signal Processing, TSP 2013. pp.507–511. https://doi.org/10.1109/TSP.2013.6613984.

Rane, G., Rege, P.P. and Patole, R., 2021, August. *Bird Classification based on Bird Sounds.* In 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 1143-1147). IEEE.

Şaşmaz, E. and Tek, F.B., 2018, September. *Animal sound classification using a convolutional neural network.* In 2018 3rd International Conference on Computer Science and Engineering (UBMK) (pp. 625-629). IEEE.

Liu, H., Liu, C., Zhao, T. and Liu, Y., 2021, November. *Bird song classification based on improved bi-lstm-densenet network.* In 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE) (pp. 152-155). IEEE.

Sankupellay, M. and Konovalov, D., 2018, November. *Bird call recognition using deep convolutional neural network, ResNet-50.* In Proc. Acoustics (Vol. 7, No. 2018, pp. 1-8).

Noumida, A. and Rajan, R., 2021, July. *Deep learning-based automatic bird species identification from isolated recordings.* In 2021 8th International Conference on Smart Computing and Communications (ICSCC) (pp. 252-256). IEEE.

Islam, S., Khan, S.I.A., Abedin, M.M., Habibullah, K.M. and Das, A.K., 2019, July. *Bird species classification from an image using VGG-16 network.* In Proceedings of the 7th International Conference on Computer and Communications Management (pp. 38-42).

J. Salamon, J. P. Bello, A. Farnsworth and S. Kelling, "*Fusing shallow and deep learning for bioacoustic bird species classification*", Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp. 141-145, Mar. 2017.

J. Salamon, J. P. Bello, A. Farnsworth and S. Kelling, "*Fusing shallow and deep learning for bioacoustic bird species classification*", Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp. 141-145, Mar. 2017.

Appendix

Files

The iterim report has been added along with an A4 copy of the poster in their seperate folders called "INTERIM_REPORT" and "POSTER" respectively.

The source code with the implimentation has been added to "SOURCE_CODE".

Source Code

CNN

```
# install tensor flow
!pip install tensorflow
!pip install librosa==0.9.1
# install version 3.5.2 to enable subplots
!pip install matplotlib==3.5.2 --user

# import modules needed
import sys
import tensorflow
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot

%matplotlib inline
from matplotlib import pyplot
# import warnings filter
from warnings import simplefilter
# check for GPU availability
from tensorflow.python.client import device_lib

# import modules needed for audio processing
import librosa
import librosa.display
import IPython.display as ipd
#import modules for file names
```

```python
import glob

from sklearn.model_selection import train_test_split
from tqdm import tqdm

# imports for feature transformation
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import class_weight
from tensorflow.keras.utils import to_categorical
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
print("User Current Version:-", sys.version)
print(device_lib.list_local_devices())

matplotlib.__version__
# check if matplotlib downgraded to 3.5 to enable subplots

print(librosa.__version__)

# Dataset details
# 264 recordings from 88 species
# using pandas read the

birdsong = pd.read_csv('dataset/birdsong_metadata.csv')
birdsong = birdsong[["file_id", "english_cname"]] # only select 2 columns
birdsong

# add the csv data to a dictionary,
# get the file_id as "keys" and the "english_cname" or
#  the bird name as it's values
bird = birdsong.to_dict()
ids=list(bird['file_id'].values())
print('Number of instances: ',len(ids))
bird_name =list(bird['english_cname'].values())


# slicing the wave files into smaller clips of 5seconds
# all .flac bird sound files are located at working
# directory under dataset/recordings/*

dataset=[]
for filename in glob.iglob('dataset/recordings/*'):
    if (filename[-5:]=='.flac'):
        identity = filename.split('\\')[0][:-5]
        # Strip the first 21 characters
        identity = filename[21:]
        # Strip the last 5 characters
        identity = identity[:-5]
        index=ids.index(int(identity))
        label = bird_name[index]
```

```
        duration = librosa.get_duration(filename=filename)
        if duration>= 5:
            slice_size = 5 #slice the first 5 seconds off
            iterations = int((duration-slice_size)/(slice_size - 1))
            iterations += 1
            # initial_offset is the starting point of the recording
            initial_offset = (duration - ((iterations*(slice_size-1)) + 1))/2
            for i in range(iterations):
                # starting point of the slice from the
                # initial_offset of the recording
                offset = initial_offset + i*(slice_size-1)
                dataset.append({"filename": filename,
                                "label": label,
                               "offset": offset})
dataset = pd.DataFrame(dataset)
dataset.info()


# the sliced recordings offset values for each bird type as label,
# offset values and correspoding filename displayed
dataset = dataset[dataset['label'].isin(list(dataset.label.
                                            value_counts()[:10]
                                            .index))]
dataset.head(20)


# display the number of recording samples
# from the dataset per bird type of the total 10
dataset.label.value_counts()


# display the graph with the dataset distribution
# show the bird names on the x-axis and the number of recordings
# per bird on the y-axis
pyplot.figure(figsize=(8,6), dpi=120)
dataset.label.value_counts().plot(kind='bar', title="Dataset distribution")
plt.show()


# Split the data into training and test samples,
# 80% for training and 20% for testing
train, test = train_test_split(dataset, test_size=0.2, random_state=42)
print("Train: %i" % len(train))
print("Test: %i" % len(test))



# display the wave plot, mel spectograms
# and MFCC's for each of the 10 bird types
plt.figure(figsize=(10,60))
```

```python
idx = 0
for label in dataset.label.unique():
    y, sr = librosa.load(dataset[dataset.label==label].filename.iloc[1],
    duration=5)
    print(dataset[dataset.label==label].filename.iloc[1])

    # Wave plot
    idx+=1
    plt.subplot(30, 3, idx)
    plt.title("%s waveplot" % label,fontsize=10)
    librosa.display.waveshow(y, sr=sr)

    # Mel Spectrogram
    idx+=1
    plt.subplot(30, 3, idx)
    S = librosa.feature.melspectrogram(y, sr=sr,
                                    n_fft=2048,
                                    hop_length=512,
                                    n_mels=128)
    S_DB = librosa.power_to_db(S, ref=np.max)
    librosa.display.specshow(S_DB, sr=sr,
                          hop_length=512,
                          x_axis='time',
                          y_axis='mel')
    plt.title("%s mel spectogram" % label,fontsize=10)

    # MFCC
    idx+=1
    mfccs = librosa.feature.mfcc(S=librosa.power_to_db(S), n_mfcc=40)
    plt.subplot(30, 3, idx)
    librosa.display.specshow(mfccs, x_axis='time')
    plt.title("%s MFCC" % label,fontsize=10)
plt.subplots_adjust(hspace=1, wspace=0.5)
plt.show()


# Extract mfcc features for each of the processed audio recordings
def extract_features(audio_path,offset):
    y, sr = librosa.load(audio_path, offset=offset, duration=5)
    # function is used to compute the Mel-spectrogram using librosa
    S = librosa.feature.melspectrogram(y, sr=sr,
                                   n_fft=2048,
                                   hop_length=512,
                                   n_mels=128)
    #next extarct the MFCC's
    mfccs = librosa.feature.mfcc(S=librosa.power_to_db(S),
                                n_mfcc=40)
    return mfccs
```

```python

x_train = []
x_test = []

# append the extracted features extracted for each recording
# from the "train" and "test" datasets
# into testing and training arrays

for idx in tqdm(range(len(train))):
    x_train.append(extract_features(train.filename.iloc[idx],
                                    train.offset.iloc[idx]))

for idx in tqdm(range(len(test))):
    x_test.append(extract_features(test.filename.iloc[idx],
                                   test.offset.iloc[idx]))

x_test = (np.asarray(x_test))
x_train = (np.asarray(x_train))

print("X train:", train.shape)
print("X test:", test.shape)


# Encode Labels
encoder = LabelEncoder()
encoder.fit(train.label)

y_train = encoder.transform(train.label)
y_test = encoder.transform(test.label)

# Compute class weights
class_weights = class_weight.compute_class_weight(class_weight='balanced',
                                                  classes = np.unique(y_train),
                                                  y= y_train)


# x and y train and test shapes before reshaping
print("X train shape:", x_train.shape)
print("Y train shape:", y_train.shape)

print("X test shape:", x_test.shape)
print("Y train shape:", y_test.shape)


x_train = x_train.reshape(x_train.shape[0],
                          x_train.shape[1],
                          x_train.shape[2],1)
x_test =  x_test.reshape(x_test.shape[0],
                        x_test.shape[1],
                        x_test.shape[2], 1)
```

```python
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# x and y train and test shapes after reshaping
print("X train:", x_train.shape)
print("Y train:", y_train.shape)
print("X test:", x_test.shape)
print("Y test:", y_test.shape)


import tensorflow as tf
import keras

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D,Conv2D,MaxPooling2D,GlobalAveragePooling2D

# CNN MODEL
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2,
                 input_shape=(x_train.shape[1],
                              x_train.shape[2],
                              x_train.shape[3]),
                              activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=128, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.5))
model.add(GlobalAveragePooling2D())

model.add(Dense(len(encoder.classes_), activation='softmax'))

# PROVIDES MODEL SUMMARY
model.summary()


# using adam optimizer with learning rate(lr) 10^-3
adam = tf.keras.optimizers.Adam(learning_rate=0.001)
# compile model using categorical crossentropy, adam optimizer with lr 10^-3
# and metrics for measuring results as "accuray"
```

```python
model.compile(loss='categorical_crossentropy',
               metrics=['accuracy'],
               optimizer='adam')


# Train the model in 32 batches and for 50 epochs
from datetime import datetime
start = datetime.now()

history = model.fit(x_train, y_train,
              batch_size=32,
              epochs=50,
              validation_data=(x_test, y_test),
              shuffle=True)
duration = datetime.now() - start
print("Training completed in time: ", duration)


# Loss Curves
plt.figure(figsize=[14,10])
plt.subplot(211)
plt.plot(history.history['loss'],'#d62728',linewidth=3.0)
plt.plot(history.history['val_loss'],'#1f77b4',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves',fontsize=16)

# Accuracy Curves
plt.figure(figsize=[14,10])
plt.subplot(212)
plt.plot(history.history['accuracy'],'#d62728',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'#1f77b4',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)
plt.show()


# get accuracy score for the model
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])


from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import confusion_matrix

```

```python
# Display precision,f1 score,recall scores
predictions = model.predict(x_test, verbose=1)

y_true, y_pred = [],[]
classes = encoder.classes_
for idx, prediction in enumerate(predictions):
    y_true.append(classes[np.argmax(y_test[idx])])
    y_pred.append(classes[np.argmax(prediction)])

print(classification_report(y_pred, y_true))

classes

import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_true, y_pred)
labels_vertical = "\n".join(classes)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
                                            display_labels=classes)
fig, ax = plt.subplots(figsize=(8, 6))
cm_display.plot(ax=ax)
# Rotate the x-axis labels to make them vertical
plt.xticks(rotation=90)
plt.show()

model_name = "birdSoundClassifier.h5"
model.save(model_name)


# load model
from keras.models import load_model
model = load_model("birdSoundClassifier.h5")

# load and evaluate a saved model
# File path for the recording we want to be classy
classify_file = "dataset/recordings/xc123168.flac"
x_test = []
x_test.append(extract_features(classify_file,0.5))
x_test = np.asarray(x_test)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
pred = model.predict(x_test,verbose=1)
print(pred)

# predict a sample using the trained model
pred_class = model.predict(x_test)
index = np.argmax(pred_class, axis=1)
print(classes[index])
```

```
# verify the bird name using it's file_id
actual = birdsong.loc[birdsong['file_id'] == 123168]
actual
```

Code 1:Deep Learning model using MFCC-CNN

LSTM

```

# install tensor flow
!pip install tensorflow

# install version 3.5.2 to enable subplots
!pip install matplotlib==3.5.2 --user

print(matplotlib.__version__)

# install version 3.5.2 to enable subplots
!pip install librosa==0.9.1

# check if matplotlib downgraded to 3.5 to enable subplots
print(librosa.__version__)

# import modules needed
import sys
# import warnings filter
from warnings import simplefilter
# check for GPU availability
from tensorflow.python.client import device_lib

# imports
import tensorflow
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
import matplotlib.pyplot
from keras.models import load_model
import numpy as np
from tqdm import tqdm

#import modules for file names
import glob

# import modules needed for audio processing
import librosa
```

```
import librosa.display
import IPython.display as ipd

# imports for feature transformation
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import class_weight
from tensorflow.keras.utils import to_categorical

# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
print("User Current Version:-", sys.version)
print(device_lib.list_local_devices())

# Dataset details
# 264 recordings from 88 species
# using pandas read the

birdsong = pd.read_csv('dataset/birdsong_metadata.csv')
 # only select 2 columns
birdsong = birdsong[["file_id", "english_cname"]]
birdsong

# add the csv data to a dictionary, get the file_id as "keys"
#  and the "english_cname" or the bird name as it's values
bird = birdsong.to_dict()
ids=list(bird['file_id'].values())
print('Number of instances: ',len(ids))
bird_name =list(bird['english_cname'].values())

# slicing the wave files into smaller clips of 5seconds
# all .flac bird sound files are located at working
#  directory under dataset/recordings/*
dataset=[]
for filename in glob.iglob('dataset/recordings/*'):
    if (filename[-5:]=='.flac'):
        identity = filename.split('\\')[0][:-5]
        # Strip the first 21 characters
        identity = filename[21:]
        # Strip the last 5 characters
        identity = identity[:-5]
        index=ids.index(int(identity))
        label = bird_name[index]
        duration = librosa.get_duration(filename=filename)
        if duration>= 5:
            slice_size = 5 #slice the first 5 seconds off
            iterations = int((duration-slice_size)/(slice_size - 1))
            iterations += 1
            # initial_offset is the starting point of the recording
            initial_offset = (duration - ((iterations*(slice_size-1)) + 1))/2
            for i in range(iterations):
```

```
                # starting point of the slice from the initial_offset
                #  of the recording
                offset = initial_offset + i*(slice_size-1)
                dataset.append({"filename": filename,
                                "label": label,
                                "offset": offset})
dataset = pd.DataFrame(dataset)
dataset.info()

# the sliced recordings offset values for each bird type as label,
# offset values and correspoding filename displayed
dataset = dataset[dataset['label'].isin(list(dataset
                                             .label.value_counts()[:10]
                                             .index))]
dataset.head(20)

# display the number of recording samples from the dataset per bird type
dataset.label.value_counts()


# display the graph with the dataset distribution
# show the bird names on the x-axis and the number of
# recordings per bird on the y-axis
pyplot.figure(figsize=(8,6), dpi=120)
dataset.label.value_counts().plot(kind='bar', title="Dataset distribution")
plt.show()


# Split the data into training and test samples,
# 80% for training and 20% for testing
train, test = train_test_split(dataset, test_size=0.2, random_state=123)
print("Train: %i" % len(train))
print("Test: %i" % len(test))


# display the waveplots and the MFCC respectively for each of the 10 bird types
plt.figure(figsize=(10,40))
idx = 0
for label in dataset.label.unique():
    y, sr = librosa.load(dataset[dataset.label==label]
                         .filename.iloc[1], duration=5)
    print(dataset[dataset.label==label].filename.iloc[1])

    # Wave plot
    idx+=1
    plt.subplot(10, 2, idx)
    plt.title("%s waveplot" % label,fontsize=10)
    librosa.display.waveshow(y, sr=sr)

    # MFCC
```

```python
    idx+=1
    mfccs = librosa.feature.mfcc(y,n_fft=255,hop_length=512,n_mfcc=13,n_mels=5)
    plt.subplot(10, 2, idx)
    librosa.display.specshow(mfccs,sr=sr,cmap='cool',
                             hop_length=512,
                             x_axis='time')
    plt.title("%s MFCC" % label,fontsize=10)
    plt.colorbar()

plt.subplots_adjust(hspace=0.5, wspace=0.5)
plt.show()

# Extract mfcc features for each of the processed audio recordings
def extract_features(audio_path,offset):
    y, sr = librosa.load(audio_path, offset=offset, duration=5)
    # function is used to compute the MFCC's using librosa
    mfccs = librosa.feature.mfcc(y,n_fft=255,
                                 hop_length=512,
                                 n_mfcc=13,
                                 n_mels=5)
    return mfccs

x_train = []
x_test = []

# append the extracted features extracted for each recording
#  from the "train" and "test" datasets
# into testing and training arrays

for idx in tqdm(range(len(train))):
    x_train.append(extract_features(train.filename.iloc[idx],
                                    train.offset.iloc[idx]))

for idx in tqdm(range(len(test))):
    x_test.append(extract_features(test.filename.iloc[idx],
                                   test.offset.iloc[idx]))

x_test = (np.asarray(x_test))
x_train = (np.asarray(x_train))

print("X train:", train.shape)
print("X test:", test.shape)

# Encode Labels
encoder = LabelEncoder()
encoder.fit(train.label)

y_train = encoder.transform(train.label)
y_test = encoder.transform(test.label)

```

```
# Compute class weights
class_weights = class_weight.compute_class_weight(class_weight='balanced',
                                                  classes = np.unique(y_train),
                                                  y= y_train)

# x and y train and test shapes before reshaping
print("X train shape:", x_train.shape)
print("Y train shape:", y_train.shape)

print("X test shape:", x_test.shape)
print("Y train shape:", y_test.shape)


x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2])
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2])
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# x and y train and test shapes after reshaping
print("X train:", x_train.shape)
print("Y train:", y_train.shape)
print("X test:", x_test.shape)
print("Y test:", y_test.shape)

from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from tensorflow.keras import regularizers

model = Sequential([
    LSTM(64, return_sequences=False, input_shape=(5,216),
         kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(10, activation='softmax')
])

# PROVIDES MODEL SUMMARY
model.summary()

# using adam optimizer with learning rate(lr) 10^-3
adam = tensorflow.keras.optimizers.Adam(learning_rate=0.001)
# compile model using categorical crossentropy, adam optimizer
#  with lr 10^-3 and metrics for measuring results as "accuray"
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train the model in 32 batches and for 100 epochs
from datetime import datetime
```

```python
start = datetime.now()
history = model.fit(x_train, y_train,
              batch_size=32,
              epochs=100,
              validation_data=(x_test, y_test),
              shuffle=True)

duration = datetime.now() - start
print("Training completed in time: ", duration)

# Loss Curves
plt.figure(figsize=[14,10])
plt.subplot(211)
plt.plot(history.history['loss'],'#d62728',linewidth=3.0)
plt.plot(history.history['val_loss'],'#1f77b4',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves',fontsize=16)

# Accuracy Curves
plt.figure(figsize=[14,10])
plt.subplot(212)
plt.plot(history.history['accuracy'],'#d62728',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'#1f77b4',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)

# get accuracy score for the model
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# Display precision,f1 score,recall scores

predictions = model.predict(x_test, verbose=1)

y_true, y_pred = [],[]
classes = encoder.classes_
for idx, prediction in enumerate(predictions):
    y_true.append(classes[np.argmax(y_test[idx])])
    y_pred.append(classes[np.argmax(prediction)])

print(classification_report(y_pred, y_true))
```

```

classes

import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_true, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
                                            display_labels=classes)
fig, ax = plt.subplots(figsize=(8, 6))
cm_display.plot(ax=ax)
# Rotate the x-axis labels to make them vertical
plt.xticks(rotation=90)
plt.show()

model_name = "birdSoundClassifier.h5"
model.save(model_name)

# load model
from keras.models import load_model
model = load_model("birdSoundClassifier.h5")

# load and evaluate a saved model
# File path for the recording we want to be classy

classify_file = "dataset/recordings/xc123168.flac"
x_test = []
x_test.append(extract_features(classify_file,0.5))
x_test = np.asarray(x_test)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
pred = model.predict(x_test,verbose=1)
print(pred)

# predict a sample using the trained model
pred_class = model.predict(x_test)
index = np.argmax(pred_class, axis=1)
print(classes[index])

# verify the bird name using it's file_id
actual = birdsong.loc[birdsong['file_id'] == 123168]
actual

```

Code 2:Deep Learning Model using MFCC-LSTM