BIRD CALL DETECTION USING DEEP LEARNING

by

Cihan Yüksel

Submitted to Graduate School of Natural and Applied Sciences

in Partial Fulfillment of the Requirements

for the Degree of Master of Science in

Electrical and Electronics Engineering

Yeditepe University

2020

BIRD CALL DETECTION USING DEEP LEARNING

APPROVED BY:

Assoc. Prof. Dr. Engin Maşazade .....
(Thesis Supervisor)
(Marmara University)

Prof. Dr. Duygun Erol Barkana .....
(Yeditepe University)

Prof. Dr. Haluk Küçük .....
(Marmara University)

DATE OF APPROVAL: .... /.... /2020

ACKNOWLEDGEMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Engin Maşazade for all the support and education he has given me during my masters study. I would like to thank Prof. Dr. Duygun Erol Barkana for being a committee member and her support throughout my study at Yeditepe University. I would like to thank Prof. Dr. Haluk Küçük also for being a committee member. I would like to thank my family for their endless love and support. I would like to thank graduate student my friend Mert Lale for all the help he provided.

ABSTRACT

BIRD CALL DETECTION USING DEEP LEARNING

In this thesis, we compare different deep learning methods for bird sound detection. For this purpose, by using digital signal processing methods, our audio data sets containing recordings from multiple fields are turned into features as mel spectrogram images, mel frequency cepstral coefficients (MFCC) or gammatone frequency cepstral coefficients (GTCC). For our convolutional neural network (CNN), we use different layers such as input layer, convolution layer, normalization layer, activation layer, pooling layer, fully connected layer and classification layer. The gray scale mel spectrogram images are used to train our CNN for different parameter settings such as layer sizes, layer numbers, input sizes and training options. On the other hand, extracted gammatone frequency cepstral coefficients and mel frequency cepstral coefficients are used as features for recurrent neural network (RNN) based bidirectional and unidirectional long short term memory networks (LSTM). Both MFCC and GTCC are also used as input for a simple neural network algorithm. For both of our long short term memory networks, we use different number of LSTM for comparison. Accuracy of the detection is validated for all methods for different parameters using area under curve (AUC) of receiver operating characteristics.

ÖZET

DERİN ÖĞRENMEYİ KULLANARAK KUŞ ÖTÜŞÜ TESPİTİ

Bu tezde, kuş sesi tespiti için farklı derin öğrenme yöntemlerini karşılaştırıyoruz. Bu amaçla, dijital sinyal işleme yöntemleri kullanılarak, birden fazla alandan kayıt içeren ses veri setlerimiz, mel spektrogram görüntüleri, mel frekanslı sepstral katsayıları (MFCC) veya gammaton frekanslı sepstral katsayıları olarak dönüştürülmektedir. Evrişimli sinir ağımız (CNN) için girdi katmanı, evrişim katmanı, normalizasyon katmanı, aktivasyon katmanı, havuz katmanı, tam bağlantılı katman ve sınıflandırma katmanı gibi farklı katmanlar kullanılmaktadır. Gri tonlamalı mel spektrogram görüntüleri, CNN'imizi katman boyutları, katman sayısı, giriş boyutları ve eğitim seçenekleri gibi farklı parametre ayarları ile eğitmek için kullanılmaktadır. Öte yandan, çıkarılan gammaton frekanslı sepstral katsayıları ve mel frekanslı sepstral katsayıları, tekrarlayan sinir ağı (RNN) dayalı çift yönlü ve tek yönlü uzun kısa süreli bellek ağlarının (LSTM) özellikleri olarak kullanılmaktadır. Hem MFCC hem de GTCC, basit bir sinir ağı algoritması için girdi olarak da kullanılmaktadır. Her iki uzun kısa süreli bellek ağımızda, karşılaştırma için farklı sayıda LSTM kullanılmaktadır. Algılamanın doğruluğu, alıcı çalışma karakteristikleri eğrisinin altındaki alanı (AUC) hesaplama methodu kullanılarak farklı parametreler için tüm yöntemler için doğrulanmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\boldsymbol{a}$	Activation
$\boldsymbol{b}$	Input bias
$\boldsymbol{c}$	Output bias
Δ	Delta
f	Long short term memory forget gate
f_{hw}	Filter height and width
f_{Hz}	Center frequency
f_{Mel}	Mel frequency
f_s	Sample frequency
F_{spec}	Spectrogram frequency
g	Long short termmemory external input gate
$\boldsymbol{h}$	Hidden layer
L	Loss function
MF	Mel spectrum of frame
n_h	Input height
n_w	Input width
n_c	Number of channels
N_{mfcc}	Number of mel frequency cepstral coefficients
N_{gtcc}	Number of gammatone frequency cepstral coefficients
$NFFT$	Number of fast Fourier transform
$noverlap$	Number of overlapping sections in spectrogram
$numbands$	Number of mel filter banks
$\boldsymbol{o}$	Output layer
$\boldsymbol{V}$	Output weight matrix
p	Padding height and width
P_{spec}	Spectrogram power
q	Long short term memory output gate
R	Number of mel or gammatone filters

s	Long short term memory internal state gate
s_{hw}	Stride height and width
S_h	Spectrogram matrix height
S_w	Spectrogram matrix width
T_{spec}	Spectrogram time step
$\boldsymbol{U}$	Input weight matrix
$\boldsymbol{W}$	Hidden weight matrix
$window$	Hamming window value
X_{audio}	Audio file input
$\hat{\boldsymbol{y}}$	Target layer

AUC	Area under curve
BNN	Binarized neural network
BiLSTM	Bidirectional recurrent neural network
BiRNN	Bidirectional long short term memory
CNN	Convolutional neural network
CQT	Constant-Q transform
DCASE	Detection and classification of acoustic scenes and events
Elu	Exponential linear units
ERB	Equivalent rectangular bandwidth
FPR	False positive rate
FFT	Fast Fourier transform
FC	Fully connected
GT	Gammatone filters
GTCC	Gammatone frequency cepstral coefficients
Hz	Hertz
IMDS	Image data store
LSTM	Long short term memory
MFCC	Mel frequency cepstral coefficients
NN	Neural network
NMF	Non-negative matrix factorization

patternnet	Pattern recognition network
PNG	Portable network graphics
ROC	Receiver operating characteristic
ReLu	Rectified linear unit
RNN	Recurrent neural network
TPR	True positive rate
WAV	Waveform audio file format

1. INTRODUCTION

Locating birds by their sound has been very important for many environmental cases with scientific purposes [1]. Monitoring the acoustic detection of bird calls is critical to identify the change of population and its effects. Traditionally, the detection of bird presence or identification of the bird type has been performed manually by observing the field live or listening to recordings of the population environment. With the advancements of mobile recording devices, the amount of data to be processed manually has been increased to the point where an automated system becomes a necessity. Machine Learning algorithms have been widely used for classification or detection of such large data as presented in [2].

In this thesis, using the publicly available raw recordings of multiple bird calls or other environmental sounds in a given area, we seek to determine the best classification algorithm to detect whether there is a bird presence or not. Specifically we implement and compare four different machine learning methods including a convolutional neural network (CNN), pattern recognition network (patternnet), long short term memory (LSTM) classification network and bidirectional long short term memory (BiLSTM) classification network. In order to apply our dataset to these type of networks, a preprocessing becomes essential. To use the CNN architecture on waveform audio file formatted (wav) files, a conversion of audio files to images needs to be achieved by using a mel filtered spectrogram application. For our LSTM based networks and patternnet, these wav files are first subjected to thresholding, segmentation and feature extraction then extracted features are used as input to these networks.

1.1. LITERATURE SURVEY

Classification of environmental audio is a common task that can be approached with different methods. Recognition of sound includes the compilation of audio data, feature extraction, clustering and classification of related features. In audio processing and speech detection mel frequency cepstral coefficients (MFCC) have been commonly used as features since it was first proposed by Mermelstain and Davis [3]. A selection of audio recognition

features have been suggested, such as the MFCC with matching pursuit algorithm to receive time-frequency features [4]. Using time-frequency distribution with non-negative matrix factorization was proposed for environmental audio recognition [5].

MFCC features, to the issue of distinguishing between speech and music signals have been introduced with audio segmentation and Hidden Markov models for classification [6]. Between speech and music signals, pattern classification was achieved using Gaussian mixture models on pitch and amplitude features with delta cepstral coefficients [7].

For animal sound detection, conventional methods included using MFCC features with support vector machines as opposed to Gaussian mixture models [8]. For insect sound classification a combination of MFCC and linear frequency cepstral coefficients (LFCC) with mixture of their delta coefficients on support vector machines were proposed [9]. In more recent applications, for animal sound detection, combination of CNN and support vector machines were proposed after applying segmentation to audio database [10]. With the rise of LSTM networks specific applications on animal sound detection started with MFCC features [11].

During literature survey, we came across multiple works on bird audio detection. With conventional methods, detection of bird species by their sound identification was done with MFCC and pitch features using Gaussian mixture models with maximum a posteriori estimation [12]. Comparison of Gaussian mixture models with vector quantization on bird species detection with two dimensional cepstral coefficients were also investigated [13]. Using Hidden Markov models was a common approach for bird species detection with MFCC features [14, 15]. Like in other animal sound detection applications, for bird species classification support vector machine algorithms were tested [16].

In previous studies, specifically for Detection and Classification of Acoustic Scenes and Events (DCASE) 2018 Challenge on bird detection, different ideas were presented [2]. The challenge awarded the most successful report using modern techniques of machine learning on their provided datasets which is explained in detail at Section 3.2 of Chapter 3. With a special custom activation function a combination of CNN and recurrent neural network

(RNN) was created for features with MFCC as well as applying denoising to the audio with area under curve (AUC) the receiver operating characteristic (ROC) result as 85.67 percent [17]. Using different baseline methods with mel filtered spectrogram images for frequencies between 25 Hz to 11025 Hz, a CNN was presented with spectrogram windowing using Hann function [18]. Mini batch size, optimizer method and activation function all effect the outcome of training as well as different learning rates [18, 19]. During spectrogram application it was required to remove undesired frequencies and resizing was applied for improvement [20]. It is also possible to use Hamming windowing in mel spectrogram extractions as inputs for a combination of CNN and RNN of 3D convolution with results of 87.13 percent AUC for combined data and 88.70 percent during challenge evaluation [21]. Constant-Q transform (CQT) spectrogram of non-negative matrix factorization (NMF) values for binary classification resulted 80 percent accuracy [22]. The most common methods use CNN or recurrent neural network (RNN) designs however there are other works done using Binarized neural network (BNN) with gradient function of Hard tanh with a result of 88.75 percent and 68.60 percent [23]. A designing approach was used with AlexNet and VGG, two well known systems in machine learning, for different layer sizes with log mel energies as features on Python [24]. Combining different methods of deep learning and creating CNN+RNN algorithm was approached for training [25]. One of the most important things in neural networks seems to be the parameter optimization and ability to do cross-dataset training [26, 27]. Different pooling layers have different effects on some applications [20, 28].

For CNN applications, audio files were required to be converted into spectrogram images. Image sizes affect input size which cause decrement in speed. With spectrogram, unnecessary parts of the audio such as silence or other environmental sound segments are included in training, which complicates training and may result confusion to learning. In more recent studies, for bird sound classification applications, especially LSTM networks replace other conventional methods, given the fact that RNN and LSTM methods utilize time dependence of the audio samples and achieves ground breaking results [29, 30]. Ideas has also been proposed for video processing applications on detection of bird with LSTM networks [31].

1.2. LIST OF CONTRIBUTIONS

In this thesis, we study bird detection using multiple deep learning methods. Our CNN method is similar to the framework presented by Lasseck M. [20]. We extend the speech detection system to our bird calls detection system, and our LSTM networks architecture are built with two layer steps. During our literature survey, it came to our attention that in all the work done about the bird detection most important factor is to create a simple network which can work for different datasets globally and parameter optimization. Our contributions in this thesis are briefly listed as follows;

- We set up parameters and train the networks paternnet, CNN, LSTM and BiLSTM using the DCASE 2018 challenge datasets in order to distinguish between bird calls and other environmental sounds. Our classification results show that both the LSTM and BiLSTM networks outperform the patternet and CNN in terms of accuracy and AUC scores.
- Our numerical results show that respectively in LSTM and BiLSTM networks, gammatone frequency cepstral coefficient (GTCC) features and mel frequency cepstral coefficient features show slight improvement for cases with bigger training sets compared to small training sets.

1.3. THESIS ORGANISATION

The organisation of this thesis is given as follows.

In Chapter 2, we present the theoretical topic of machine learning. We explain deep learning and its subsection CNN and LSTM network. We study layers of the CNN and LSTM architecture designs and introduce option parameters. Furthermore, we explain mel spectrogram operation and Fourier transform effects on images. We further describe theoretical steps of MFCC and GTCC.

In Chapter 3, we present the audio datasets and their properties. We explain the necessary data preparation processes mel spectrogram application which we use in CNN method in

detail. Step by step, we introduce our modified CNN and LSTM architectures, training options and all related MATLAB operations. We also mention LSTM networks and pattern recognition network features and how they are obtained.

In Chapter 4, we present and compare our numerical classification results on bird call detection for our different methods. We also further investigate using comparative results.

In Chapter 5, we discuss our findings, reference to other applications in the literature.

In Chapter 6, we summarize our thesis, explain key results and limitations of our study. We finally give an outlook on possible research directions and key suggestions for improvements on future applications.

2. THEORETICAL BACKGROUND

In this chapter mel spectrogram, mel frequency cepstral coefficients (MFCC) and gammatone frequency cepstral coefficients (GTCC), machine learning, convolutional neural networks (CNN) and long short term memory (LSTM) networks are explained.

2.1. MEL SPECTROGRAM

A spectrogram is the visual portrayal of a signal as it changes over time for different frequencies. It can be used to show audio signals as 2-D or 3-D representations as an image. Using mel spectrogram, our audio datasets are converted and saved as images of their spectrogram values because CNN are mainly used in image processing. Processed images are used as input during network training. Digital signal processing using Fast Fourier Transform (FFT) is a common practice for spectrogram applications [32]. Any type of signal but specifically audio signal in our work, after applying sampling, is cut into overlapping portions. Since these signal samples are in time domain, by using FFT they are transformed into frequency domain and their frequency spectrum magnitudes are calculated. These frequency frames are applied to a mel filter bank and mel outputs are summed together. Fourier Frequency Transform effects can be minimized by overlaying, which means performing a window application. There are different types of functions for windowing. In our spectrogram operation as default we are using Hamming Function for all windowing operations [33].

2.2. MEL FREQUENCY CEPSTRAL COEFFICIENTS

In audio processing MFCC represent short term power spectrum of a sound signal and are commonly used as features. After applying Fourier transform on a signal, obtained spectrum powers are projected on a mel scale with window overlaps into mel frequencies. Power logs of each frequency are taken and $N_{mfcc} = 13$ MFCC are acquired by applying discrete cosine transform. Resulting amplitudes are the mel coefficients of the spectrum. In mel frequency cepstrum, bands of frequencies are distanced with equal spaces just like the auditory system

response in humans [34, 35]. Signal to MFFC process is seen in Fig. 2.1. There are $R = 32$ mel bands in mel filter. The relations between the mel frequency and normal frequency where center frequency f_{Hz} 1000 Hz is equal to 1000 mel is seen as,

$$f_{Mel} = 2595 \cdot \log_{10}(1 + \frac{f_{Hz}}{700}). \tag{2.1}$$

Each MFFC from $n = 1, 2, \ldots, N_{mfcc}$ for number of R filters can be calculated with,

$$\text{MFCC}[n] = \frac{1}{R} \sum_{r=1}^{R} \log(MF[r]) \cos\left[\frac{2\pi}{R}(r + \frac{1}{2})n\right]. \tag{2.2}$$

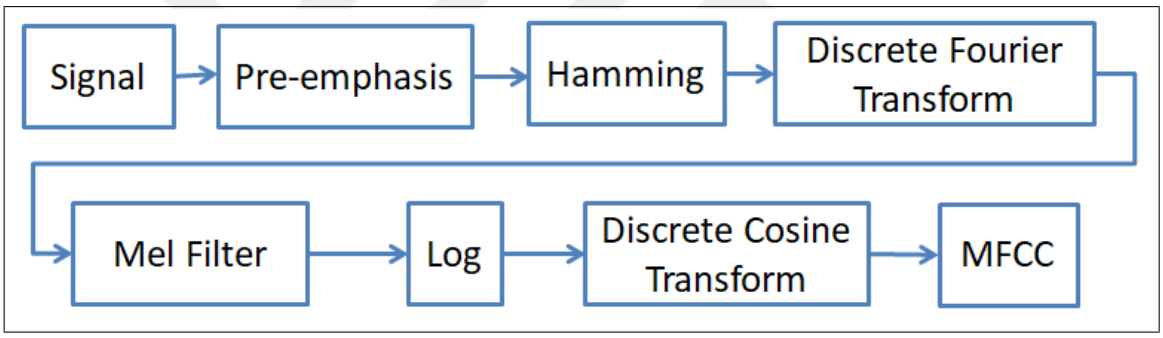

Figure 2.1. MFFC Calculation.

MFCC delta coefficients Δ_{MFCC} which are the differential of one coefficient to another at a t frame are calculated by the given formula below [36].

$$\Delta_{\text{MFCC}}[n] = \frac{\sum_{n=1}^{N} n(\text{MFCC}_{t+n} - \text{MFCC}_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.3}$$

MFCC delta-delta coefficients $\Delta\Delta_{\text{MFCC}}$ which are the differentiation of one delta coefficient

to another at a t frame are calculated by the given formula below [36].

$$\Delta\Delta_{\text{MFCC}}[n] = \frac{\sum_{n=1}^{N} n(\Delta_{t+n} - \Delta_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.4}$$

2.3. GAMMATONE FREQUENCY CEPSTRAL COEFFICIENTS

GTCC are defined as a modification of the MFCC. GTCC are using gammatone filters (GT), biologically inspired model of human auditory response, which performs a spectral analysis in the cochlea with equivalent rectangular bandwidth (ERB) bands [37]. Signal to GTCC process is seen in Fig. 2.2. Formula given below shows the relations between the equivalent rectangular bandwidth and center frequency.

$$f_{ERB} = 24.7 \cdot (4.37 \cdot (f_{Hz} + 1)) \tag{2.5}$$

For a given r th gammatone filter order with R gammatone filters, number of GTCC $N_{gtcc} = 13$ are calculated as seen in below [38].

$$\text{GTCC}[n] = \sqrt{\frac{2}{R}} \sum_{r=1}^{R} \log(GT[r]) \cos\left[\frac{r\pi}{R}(n - \frac{1}{2})\right] \tag{2.6}$$

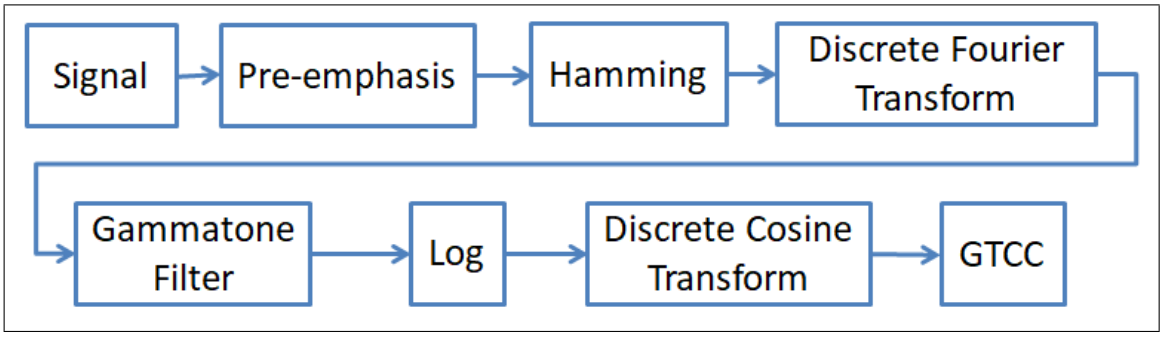

Figure 2.2. GTCC Calculation.

Table 2.1. Differences between MFCC and GTCC

Feature Type	MFCC	GTCC
Number of Bands	32	32
Frequency Scale	Mel	ERB
Nonlinear Operation	Log	Cubic Root
Time - Frequency Representation	Mel Spectrum	ERB spectrum

2.4. DEEP LEARNING AND NEURAL NETWORKS

Based on artificial neural networks one of the method varieties is called deep learning or deep structured learning. Deep learning models such as deep neural network, RNN, CNN are used in biomedical, computer, electrical and electronic sciences, medical applications and audio and image recognition. All these artificial neural networks are based on animalistic brain patterns and their process of information in their brains as seen in Fig. 2.3. In deep learning, each level figures out how to change its information input into a marginally increasingly unique and composite portrayal. Ideally the process itself can learn without any implicit teachings from us but this doesn't necessarily mean the manual tuning will be eliminated. Varying parameters or sizes in layers can demonstrate different levels of learning.

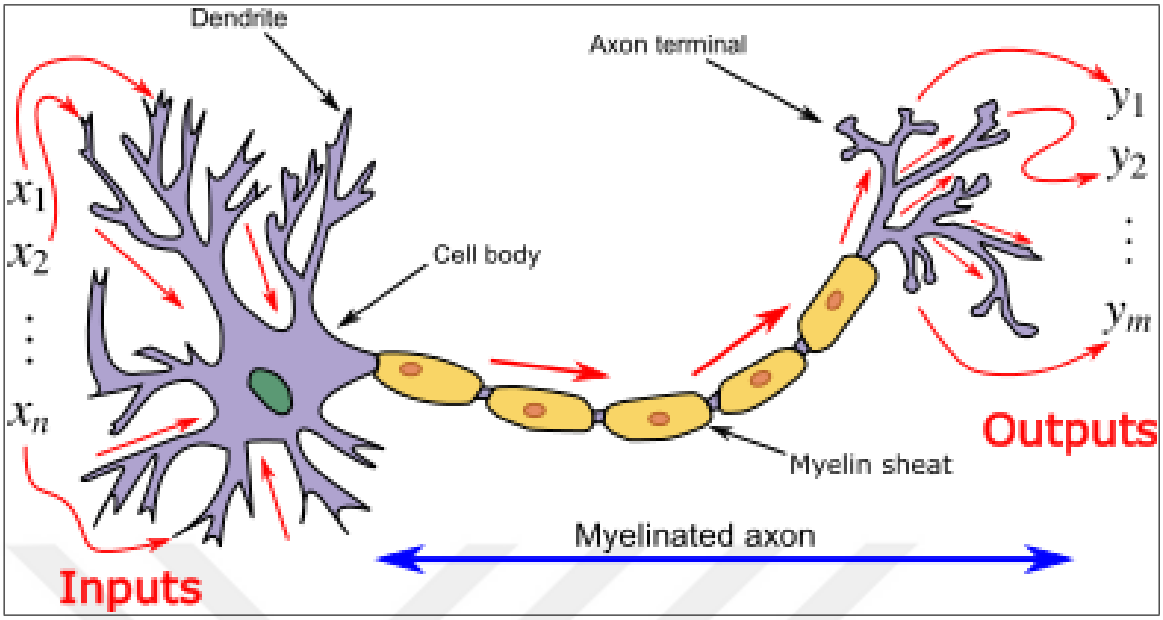

Figure 2.3. Structure of biological neuron and artificial representation [39].

In a neural network there are mathematical functions called neurons that are artificially created just as a representation of biological neurons in the brain which receives a single or multiple inputs and creates an output based on its activation function. From these activation functions we are collecting weights that change as long as the process of learning continues. Organization of neurons are generally in multiple layers for different architectures. Various layers may perform various types of changes depending on their input of info. Information start travelling from the input layer, which is the first layer, than into the output layer, which is the last layer, by passing thorough the network, perhaps crossing to the other layers on numerous occasions.

2.5. CONVOLUTIONAL NEURAL NETWORKS

The engineering of a CNN is intended to exploit the 2D structure of an info picture (or other 2D information, for example, a sound signal) [40]. This is accomplished with weights associations and connections pursued by some type of pooling followed by a activation unit finishing on a fully connected layer. An advantage of CNN is that they are simpler to prepare and have numerous less parameters than fully connected systems with a similar number of

hidden units. In CNN a common approach to back propagation is to use gradient based optimization.

2.5.1. Layers of CNN

In a CNN architecture there are three main layers named input layer, output layer and hidden layer. CNN are made of convolutional and subsampling layers generally feed into fully connected layers [41, 42].

2.5.1.1. Input Layer

First step creating a CNN is creating an input layer which uses images as inputs to the network and applies data normalization for all the images in the image data set. It is important to specify the image size and number of channels which depend on the colour of the image carefully. If there are a lot of input features in the first hidden layer depending on the number of neurons, this will create an enormous weight matrix which cause over fitting [43].

2.5.1.2. Hidden Layer

All the layers in between the input layer and output layer is called hidden layer. Once a training architecture is set there is no need to know or rewrite hidden layers, but only to feed an input and receive the output.

Two Dimensional Convolution Layer: First hidden layer of the CNN architecture. A convolutional layer connects to the input layer. Depending on the complexity of the data a CNN can be made out of multiple convolutional layers. Even though method to apply the filter is mathematically called cross correlation, in the deep learning literature by convention it is called convolution.

- **Padding**: Assume we have a $n_h \times n_w$ size image and we try to apply our $f_{hw} \times f_{hw}$ sized convolutional filter, the output size of the first layer will become $(n_h - f_{hw} +$

$1) \times (n_w - f_{hw} + 1)$. The down side of this operation is that every time a filter is applied to the input such as this, image starts to shrink until it becomes very small for processing. The second downside is that pixels on the corners of the image are used less in the output image than other pixels located through the input image, causing loss of information on the edges of the image. To solve these problems, padding operation is applied to input image. An additional pixel border with a size of p, zero valued elements of padding will make the input image size as $(n_h + 2p) \times (n_w + 2p)$ so the output image will become $(n_h + 2p - f_{hw} + 1) \times (n_w + 2p - f_{hw} + 1)$, preserving the size of the original input. There are two types of padding operation called valid padding and same Padding. In more details if valid padding is applied there would be no padding to the image and output would shrink and if same padding is applied we would receive the output image size as same as the input image size. During the same padding since the input size $n_h \times n_w$ would be equal to output size which is $(n_h + 2p - f_{hw} + 1) \times (n_w + 2p - f_{hw} + 1)$, it will return the p value as $(f_{hw} - 1)/2$. Because of this by convention f_{hw} is always an odd number [44].

- **Stride**: A stride is the step size by which the convolution filter moves on the input image. For an $n_h \times n_w$ image and we apply our $f_{hw} \times f_{hw}$ filter with a padding of p, output size becomes as seen in (2.7). If the fraction is not an integer it is rounded to the closest number [44].

$$[\frac{n_h + 2p - f_{hw}}{s_{hw}} + 1] \times [\frac{n_w + 2p - f_{hw}}{s_{hw}} + 1] \tag{2.7}$$

Three Dimensional Convolution Layer: Instead of using gray scaled images it is possible to use RGB image in CNN. The RGB image would have a size of $n_h \times n_w \times n_c$ where the n_c corresponds to red green and blue channels. Important thing to keep in mind if using RGB image the number of channels in the filter must match the number of channels making it $f_{hw} \times f_{hw} \times n_c$ size [44].

Pooling Layer: In many of the cases in CNN, pooling layers are used regularly. The fundamental principal behind the pooling layer is to combine indistinguishably similar features into one by dividing its input into regions of rectangular pooling in order to perform

a down-sampling on each region depending on the type of pooling. Addition to their improvements to the calculations speed, applying a pooling layer also causes some of the features in detection to become more powerful. It is important to keep in mind that pooling layer itself does no help to gradient descend algorithms for actual learning on its own. When the sizes of the pooling layers are created, all it will do is down sample for fixed calculations. The pooling enables almost no change to representations when components in the past layer fluctuate in position and appearance. There are different types for pooling operation. In neural network applications max pooling is used more than compared to average pooling [41].

- **Max Pooling**: As the name suggest it takes the max values from the each filter [45].
- **Average Pooling**: As the name suggest it takes the average of values from the each filter [46].

In order to define the sizes for pooling layer, the same principles from filter sizes, padding and stride can be applied. In most common applications as often times, used filter size and stride size is taken as equals to two where the value for padding is taken as zero. This practice is done to diminish its input sizes of height and width with a factor of two. Since the objective is to shrink its input in to a different size output, in a pooling layer almost no padding is ever applied.

Batch Normalization Layer: All the inputs to the normalization layer are normalized according to a mini-batch. A normalization layer is used between the rectified linear unit layer or any type of non-linearity and the layer of convolution to increase the training speed of the network. The normalization process is done firstly by mini batch mean subtraction and mini batch standard deviation division to channel activations from the convolution layer. Then using the learned parameters from every update during the training such as offset to shift and scale factor to scale its inputs, normalization layer modifies the activations. Normalization layer is used to make optimization easier by learning rate increment causing the network to learn faster. At the end of training mean and variance of the full training is calculated and recorded in properties. During validation process these recorded mean and variances are used for normalization of the activations rather than the mini batch

mean and mini batch variance. Likewise, normalization also helps network layers to learn independently [47].

Non-linear Activation Layer: As mentioned before an activation function is used in neurons to create an output for summed weights. In between the convolutional layer and batch normalization layers there is always a non-linear activation function. The rectified linear unit (ReLu) function layer apply a threshold to every input element if any of the value is smaller than zero is fixed to zero. This layer has no effect on the input size. There are different types of activation functions with the more common one being the rectified linear unit.

- **ReLu**: If any value of input is smaller than zero it is fixed to zero [48].

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.8}$$

- **Noisy ReLU**: Created by adding a Gaussian noise to ReLu equation [49].
- **Leaky Relu**: If any value of input is smaller than zero, the value is multiplied by a fixed scalar [50].

$$f(x) = \begin{cases} x, & x \geq 0 \\ scale \times x, & x < 0 \end{cases} \tag{2.9}$$

- **Clipped Relu**: If any value of input is smaller than zero it is fixed to zero. When values over clipping ceiling, it is set to that clipping ceiling [51].

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x < ceiling \\ ceiling, & x \geq ceiling \end{cases} \tag{2.10}$$

- **Elu**: If any value of input is smaller than zero Exponential linear units (Elu) gets the

mean activations closer to zero [52].

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases} \tag{2.11}$$

- **tanh**: As the name suggest input layers are applied the tanh function [53].

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.12}$$

Fully Connected Layer: Fully connected (FC) layers are created after multiple convolutional and pooling layers in the neural network. Previous layer activations are fully connected to neurons in the FC layer. In order to detect larger patterns inside the input image, a fully connected layer tries to merge locally learned informations from the previous layers. During the classification applications, final fully connected layer merges these informations for classification of correct labels which is equal to the number of labels from the data set. The input from the previous layer is multiplied by a matrix of weights with addition of a bias vector. During the regression applications response variable numbers must be same with the output size. FC layers can be converted to convolutional layers given desired filter sizes [54].

2.5.1.3. Output Layer

The last layer in the CNN architecture is the output layer [41].

Softmax Layer: In order to normalize the outputs from the last fully connected layer a softmax layer consisting on sum of positive integers is used which can then become the probabilities of classification layer for single or multi class classification problems [55].

Classification Layer: In multi class classification applications in order to calculate the cross entropy loss a classification layer is needed after the softmax layer. During training, values are taken from the softmax function and for every label of classes that are exclusive to each other they are send to cross entropy function [56].

Regression Layer: In CNN networks if the problem is a regression type rather than a classification problem, this layer calculates half mean squared error loss. Compared to classification layer, a regression layer must be attached to the final fully connected layer without a softmax layer. During training, the layer calculates the mean loss of the mini batch observations [57].

2.6. RECURRENT NEURAL NETWORKS

In deep learning, RNN are used for sequential data processing [40]. Generally feed forward networks that are fully connected have different weight parameters for each feature used as inputs but in a recurrent neural network, since there are multiple time steps, the same weights are shared. In RNN each time step output is a function of the previous time step and they are all created by using the exact same update rule that are applied to all time steps. This allows the recurrent operation to share parameters through a very deep network. For practical reasons in RNN, training is done in mini batches with different sequence lengths. The time steps in RNN do not have to be real time but can also mean the position in the sequence. RNN with the presence of time involved data can go forward or backward in sequences if all of the training sequence is inspected before going into network. The unfolded version of RNN neuron connection is seen in Fig. 2.4.

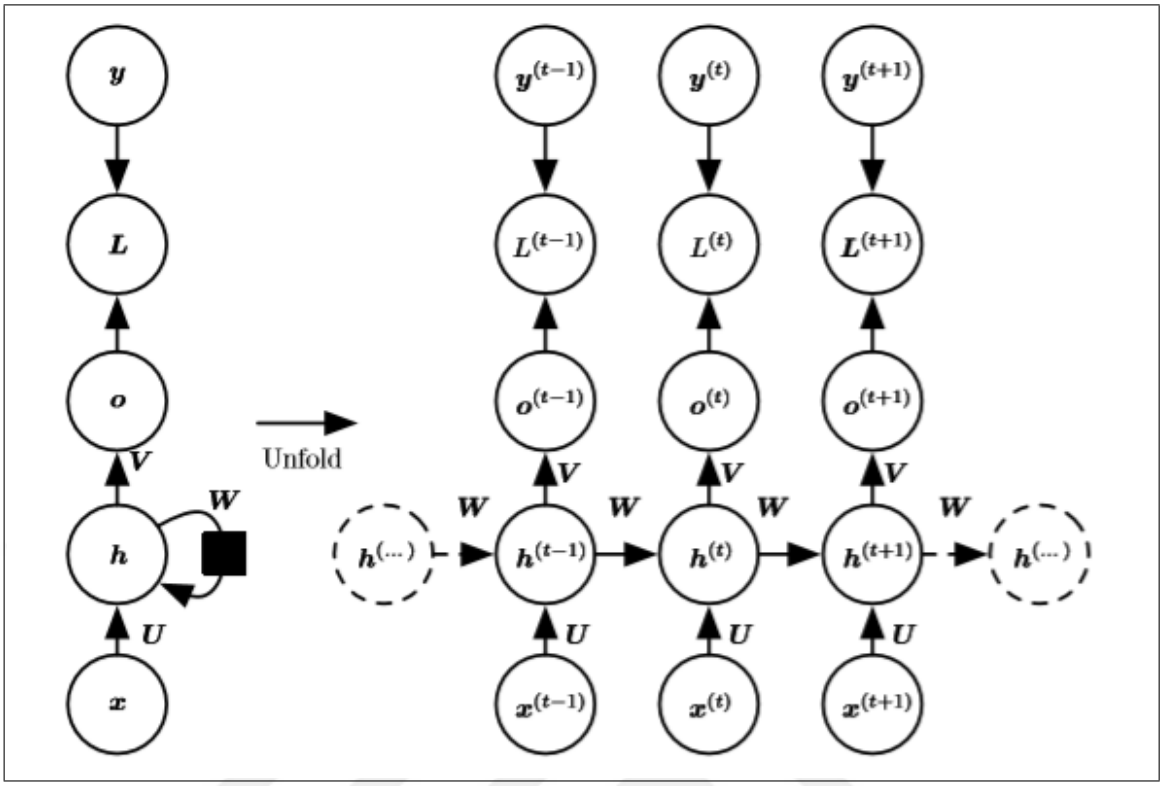

Figure 2.4. RNN [40].

In RNN sequence of $\boldsymbol{x}$ values as inputs are mapped to matching sequence of $\boldsymbol{o}$ values as outputs. The loss function L estimates the error from the comparing training target y to the output value o which are the unnormalized log probabilities after using a softmax function. This operation is done internally by the loss function using the predicted values of $\hat{\boldsymbol{y}} = softmax(o)$. A weight matrix $\boldsymbol{U}$ is characterizing connections of input to hidden, a weight matrix $\boldsymbol{W}$ is characterizing connections of hidden to hidden and a weight matrix $\boldsymbol{V}$ is characterizing connections of hidden to output during forward propagation. In order to start the forward propagation $h(0)$ initial state needs to be declared. The following equation is applied for $t = 1$ to $t = \tau$ in each time step with $\boldsymbol{b}$ and $\boldsymbol{c}$ as bias vectors with respective weights to each connection.

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{2.13}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}) \tag{2.14}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{2.15}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)}) \tag{2.16}$$

In theory, in a feed forward recurrent network, one can use almost any loss function depending on the task. Commonly output of the RNN are interpreted as distribution of probabilities using a cross-entropy loss. In Gaussian unit distributions, mean squared error and cross-entropy loss is associated just like in networks with feed forward.

2.6.1. Bidirectional Recurrent Neural Networks

As mentioned before, it is possible to design RNN that are moving both forward and backward in time sequences. This is done by combining two RNN where one starts from the beginning of the sequences and the second one starts from the end sequence. Fig. 2.5 shows the connections in a bidirectional recurrent neural networks (BiRNN) where forward moving first RNN state is shown by $\boldsymbol{h}^{(t)}$ and backward moving second RNN state is shown by $\boldsymbol{g}^{(t)}$. This operation works well for input x values at time t. The idea of bidirectional movement can be applied to 2-D inputs of images by using 4 different directed RNN going left, right, up, down. RNN compared to a CNN on images are harder to train but between features if network is taught to carry the information, it will enable lateral interactions of long-range in the same mapping [58, 59].

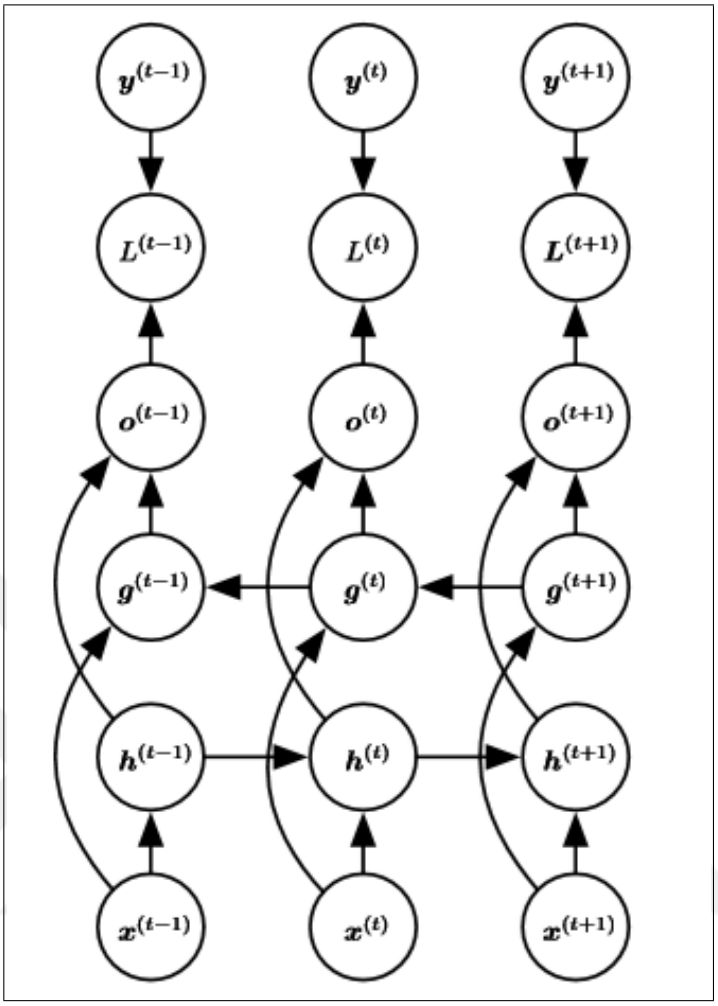

Figure 2.5. BiRNN [40].

2.7. LONG SHORT TERM MEMORY

LSTM networks are specially designed RNN which are first introduced in 1997 [60]. In long sequenced RNN a common technical problem is vanishing gradients during back propagation step. LSTM are created to deal with this gradient problem. Compared to single tanh simple design in RNN, LSTM have four different interactive connections. Rather than a unit that essentially applies non linearity to inputs and recurrent units, LSTM layers deploy self looped cells. Input and output numbers of all cells are equal just like in RNN but they also have more parameters for controlling the information flow as seen in Fig. 2.6. These hidden units control the gate weights by self-loop.

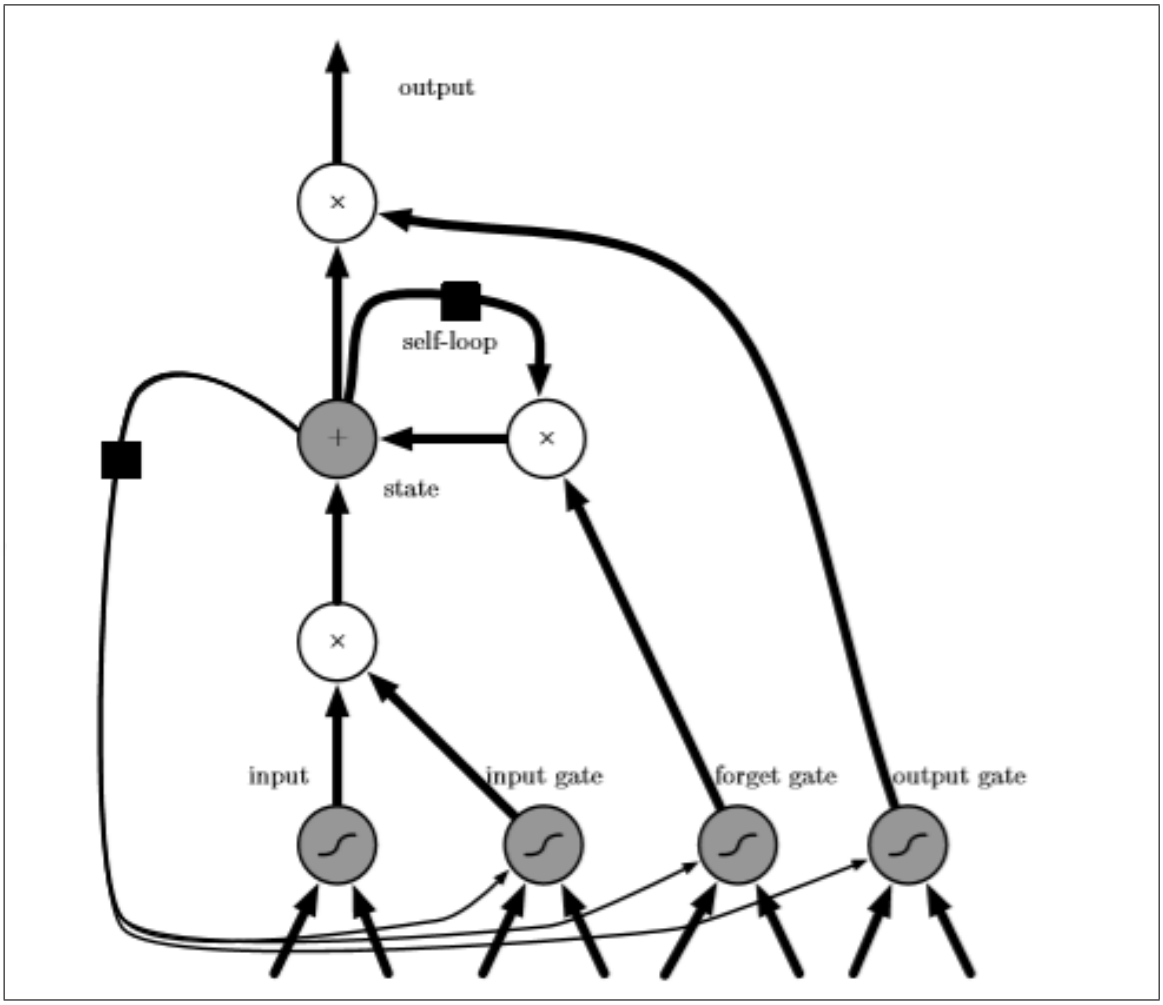

Figure 2.6. LSTM [40].

To update this self-loop, controlling forget gate $f_i^{(t)}$ with t time step of i cell is defined as,

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{ij}^f x_j^{(t)} + \sum_j W_{ij}^f h_j^{(t-1)}\right), \tag{2.17}$$

where calculations with input vector $x^{(t)}$, bias $\boldsymbol{b}^f$ and weights $\boldsymbol{U}^f$ and $\boldsymbol{W}^f$ values are specific. The internal state formulation is written as seen below with forget gate calculations included.

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{ij} x_j^{(t)} + \sum_j W_{ij}^f h_j^{(t-1)}\right). \tag{2.18}$$

The similar computation of external input gate unit $g_i^{(t)}$ with its special parameters are shown below as,

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{ij}^g x_j^{(t)} + \sum_j W_{ij}^g h_j^{(t-1)}\right). \tag{2.19}$$

The output gate $q_i^{(t)}$, using a sigmoid unit with its own respective bias $\boldsymbol{b}^o$, weights $\boldsymbol{U}^o$ and $\boldsymbol{W}^o$ is written as,

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}, \tag{2.20}$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{ij}^o x_j^{(t)} + \sum_j W_{ij}^o h_j^{(t-1)}\right). \tag{2.21}$$

2.7.1. Layers of LSTM

In a LSTM network architecture the main layers are sequence input layer, LSTM layer and output layer. LSTM networks are made of sequenced input layer followed by single or multiple LSTM or BiLSTM layers with drop out layers in between generally feed into fully connected layers with softmax and clasification layer or a regression output [61]. Fully connected and classification layers are based on the same principle in CNN system as mentioned in Section 2.5.

2.7.1.1. Sequence Input Layer

In order to use LSTM networks the training inputs needs to be data of sequences. A sequence data is input in to network by a sequence input layer [62]. Number of input size is decided depending on the number of features in a single sequence. It is possible to apply hyper parameter normalization and extract means and standard deviation for normalization.

2.7.1.2. LSTM Layer

Long-term relations in time series and sequence data between time stages is learned by LSTM layer. The layer is made of cell state which has informations that are learned from earlier time steps and output state containing a t timed output of the layer. At every steps of time, the layer either removes or adds new information, updating from the previous gates [63].

Bidirectional Long Short Term Memory Layer: BiLSTM Layer extracts long-term relations in time series and sequence data between time stages in a bidirectional way [63]. By default all LSTM layers in MATLAB are using $\tanh$ function for state activations while using sigmoid function for gate activations. All LSTM layers requires defining of number of hidden units, the hidden state which are corresponding to the information saved between time steps.

2.8. AREA UNDER THE CURVE

Measuring the performance of a machine learning system is a critical task in determination of the best working systems. Using the standard accuracy evaluation in classification models, correct number of predictions divided by number of total prediction, is a possible application but area under curve (AUC) of receiver operating characteristics (ROC) is a more common approach [65]. Outcome of a machine learning system depending on the target class can have 4 different prediction classes in a confusion matrix as seen in Fig. 2.7.

- **True Positive (TP)**: The system prediction is possitive class where target is positive class.
- **False Positive (FP)**: The system prediction is possitive class where target is negative class.
- **True Negative (TN)**: The system prediction is negative class where target is negative class.
- **False Negative (FN)**: The system prediction is negative class where target is positive class.

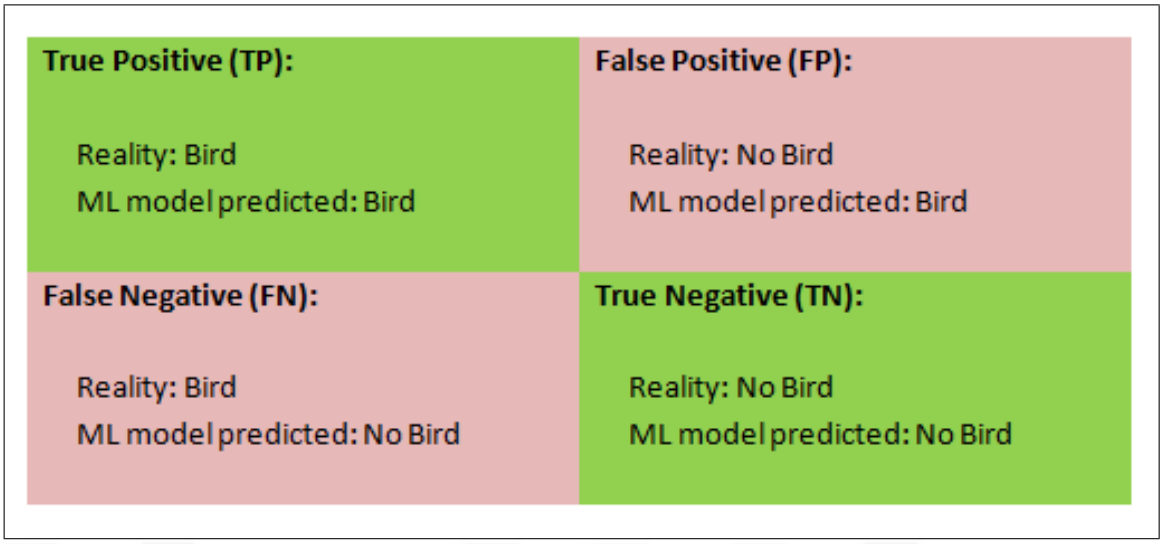

Figure 2.7. Confusion matrix [66].

In AUC operations classification performance evaluations are done at different threshold levels according to ROC curve probality results. AUC informs us how good is the system at understanding different classes. In order to plot ROC, true positive rate (TPR) also known as sensitivity and false positive rate (FPR) are required. The area left under the ROC curve as seen in Fig. 2.8 at different thresholds gives us the AUC score where the score range is from 0 to 1. If a system predicts all the targets wrong the AUC score yields 0 but if all the predictions are correct AUC score would yield 1. If a system returns AUC as 50 percent the system can not separate classes from each other. To calculate the area under the TPR and FPR values at different thresholds j equation can be given as,

$$\mathrm{AUC} = \sum_{j=1}^{J} \left(\frac{\mathrm{TPR}_j + \mathrm{TPR}_{j-1}}{2} \right) \left(\mathrm{FPR}_j - \mathrm{FPR}_{j-1} \right). \tag{2.22}$$

where J represents the total number of thresholds used to find the AUC.

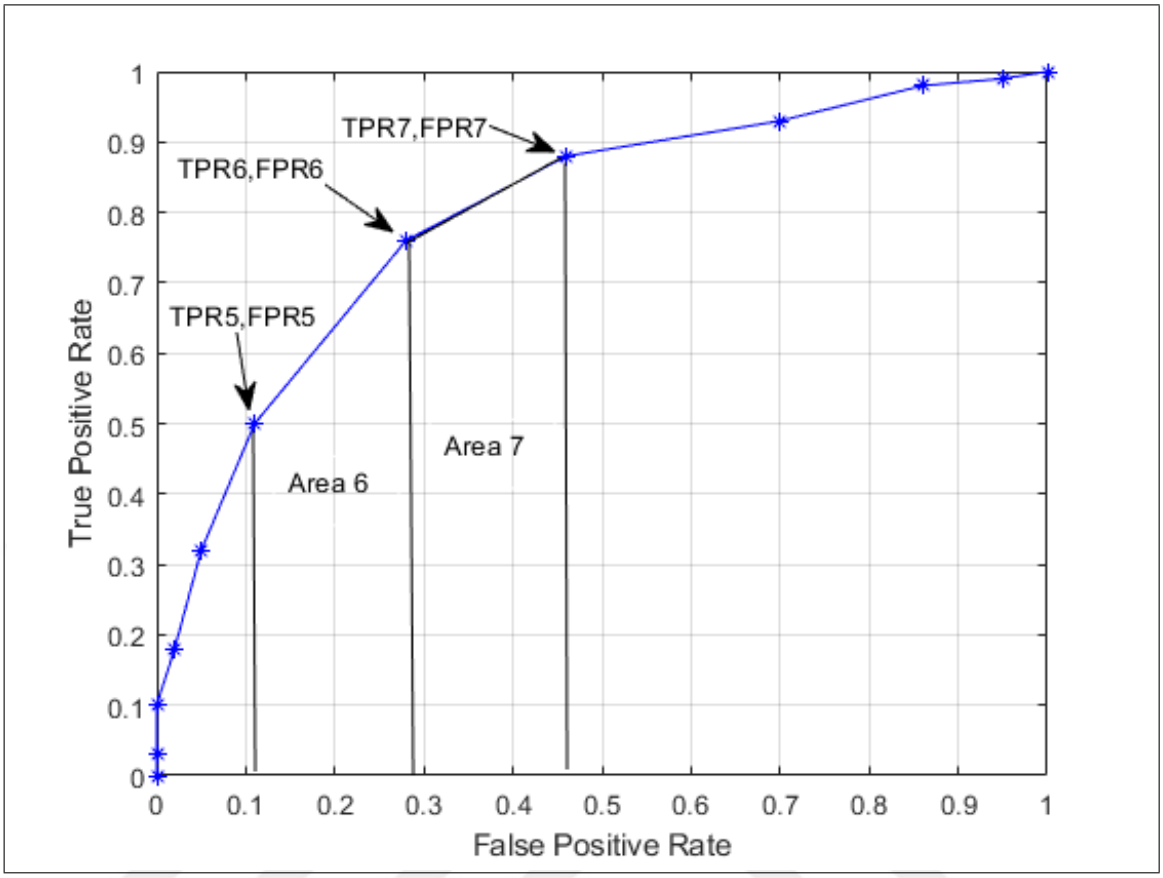

Figure 2.8. Area under curve.

3. EXPERIMENTAL PROCESS

In this chapter the datasets and our experimental process in MATLAB are explained.

3.1. MATLAB TOOLBOX

Since MATLAB is not a free program, there are certain toolboxes required for audio processing and deep learning. We use the student version of MATLAB and also purchased the deep learning toolbox for all machine learning applications. Audio toolbox for feature extraction, speech detection, thresholding and digital signal processing system toolbox for digital signal processing applications are also used.

3.2. DATASETS

The audio sets which we use in our thesis was provided by the DCASE 2018 Challenge and the audio sets can be accessed online [2, 67]. There are three different datasets available for usage named: freefield1010, warblrb10k, BirdVox-DCASE-20k.

These audio sets were recorded in two main methods called remote monitoring and crowdsourced monitoring. For remote monitoring type recording, which was also called passive recording, fixed and known recording equipments were used. For crowdsourced monitoring type recording, which was done in the style of active recording, the equipment used was uncontrolled meaning it can be anything from an expensive microphone to mobile phones. Crowdsource recordings as the name suggest were supplied by the crowds. Recording locations and data quantity also differs for the data set, as in freefield1010 was recorded worldwide for a quantity of 7960 audio files, warblrb10k was recorded in United Kingdom for a quantity of 8000 audio files and BirdVox-DCASE-20k was recorded in United States near New York, Ithaca for a quantity of 20000 audio files. All the audio files were divided into 10 sec clips of .wav format and labelled manually by the DCASE as 1 or 0 depending on the bird presence. Originally there are also unlabelled audio files in the DCASE databank but for validation purposes we only use the labelled data. The summary

of the data can be seen in the Table 3.1.

Table 3.1. Datasets

Name	Location	Recording Type	Label	Number of Samples
warblrb10k	United Kingdom	Crowdsourced	0-1	8000
freefield1010	Worldwide	Crowdsourced	0-1	7960
BirdVox-DCASE-20k	United States	Remote	0-1	20000

3.3. METHODS

For CNN, MFCC or GTCC processes there are different methods to extract features as well as preparing the audio datasets. In order to determine the effect of dataset size on validation accuracy, two sets of 5000 and 15400 audio files are used. In both sets of 5000 and 15400, same audio files are used as training and validation sets. For all operations the freefield1010 and warblrb10k audio datasets are merged into a single folder. The folder names are labelled as ”Bird” or ”No Bird” according to their bird call presence. In all training models 85 percent of whole set is used for training and 10 percent is used for validation.

3.3.1. CNN Model

In MATLAB environment an algorithm to read the audio data for sample frequency of 44100 Hz is written. After audio sampling process for sample time of 10 seconds with sample frequency of 44100 Hz yields us audio matrix of size 441000 $\times$1. Audio files from the merged datasets are either a little longer or shorter then 10 seconds so with the algorithm they are all fixed to 10 seconds of same portion. CNN model with MATLAB requires images as input variables. For this purpose using mel filtered spectrogram, audio files are required to be converted into images. For mel spectrogram analysis the built in MATLAB command ”melSpectrogram” is used [68]. During this operation Hamming window length ($window$)

is set as 60 ms, overlap length ($noverlap$) is set as 40 ms, number of Fourier transforms ($NFFT$) is set to 4096 and number of mel filter banks ($numbands$) is set as 32. These values created a mel spectrogram matrix S. The result is a matrix of 32×498 according to formulas given below.

$$S_h = numbands \tag{3.1}$$

$$S_w = \frac{(length(X_{audio}) - noverlap)}{(window - noverlap)} \tag{3.2}$$

$$S = S_h \times S_w \tag{3.3}$$

Figure 3.1. Mel spectrogram ”Bird” Train (a) 1, (b) 2, (c) 3, (d) 4.

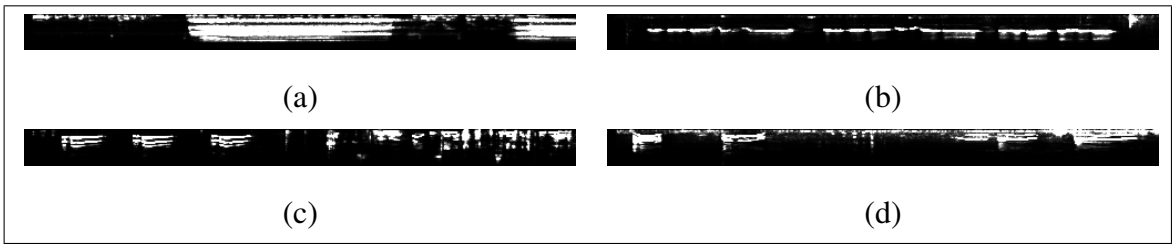

Figure 3.2. Mel spectrogram ”No Bird” Train (a) 2126, (b) 2127, (c) 2128, (d) 2129.

Mel spectrogram matrix S is saved as gray images with MATLAB built in command ”imwrite” into correct labelled folders. Using the S values mel spectrogram images are created as seen in Fig. 3.1 and Fig. 3.2. Gray images are moved into a special folder named

”Bird” or ”No Bird” depending on their bird label. An image data store (imds) is created for convolutional neural network model and imds takes the labels from the folder names.

Fig. 3.3 shows the CNN architecture for a single layer. Our CNN network input layer takes a 32×498 sized image from imds and starts applying convolution layer of size $f \times f$ with specified number of filters with same padding. Output of the convolution layer is feed into batch normalization layer then into ReLu Layer. This process is repeated 3 times, every time filter sizes and number of filters change. First convolution layer is using 3×3 of 8 filters, second convolution layer is using 3×3 of 16 filters and last convolution layer is using 3×3 of 32 filters. Final output of the third layer is applied to fully connected layer with only 2 class labels before finally passing into softmax layer with classification layer.

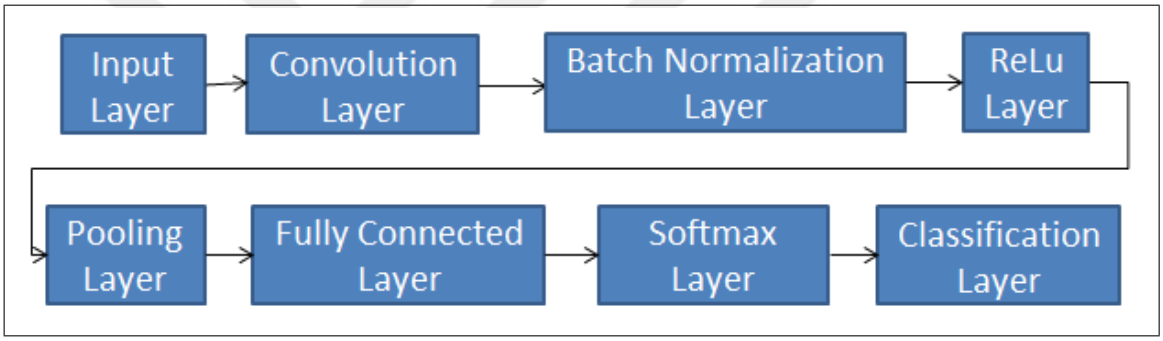

Figure 3.3. Layers of a convolutional neural network.

3.3.2. Bird Call Segmentation

Before extracting MFCC or GTCC features in order to remove silence and non bird call segments a thresholding method is applied. All audio files are broken into 50 ms of non overlapping frames. Each frame is used to calculate signal energy and spectral centroid values. Threshold values for energy is decided as mean of calculated energy values from the frames and centroid threshold is decided as 10 kHz because bird calls typically have frequencies between 1 kHz to 8 kHz [69]. Frames with energy values greater than threshold energy and centroid values less than threshold centroid is accepted as the bird call region. After thresholding intersecting bird call segments are merged together. In Fig. 3.4 threshold effects and bird call regions can be seen on a single audio sample.

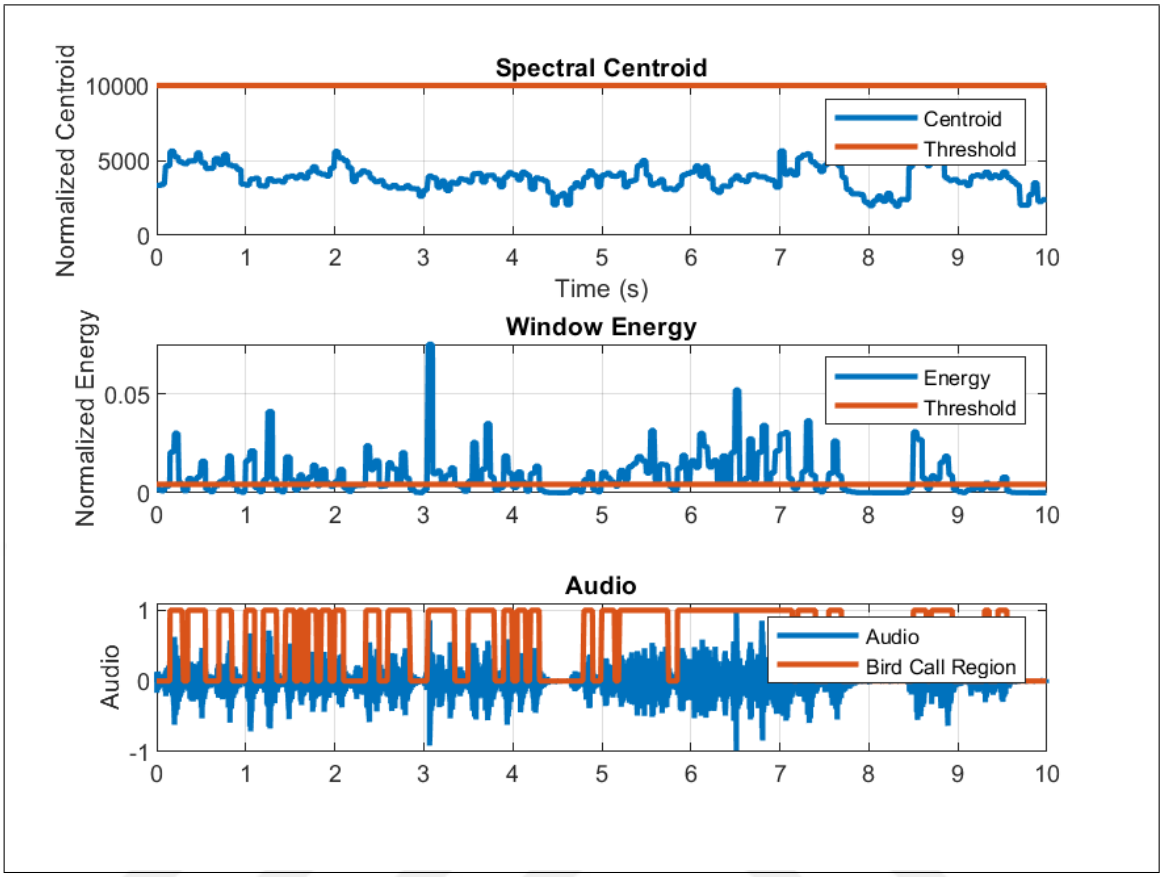

Figure 3.4. Energy and centroid threshold on audio.

3.3.3. Feature Extraction

To extract MFCC and GTCC features MATLAB built in function ”audioFeatureExtractor” is used [70]. Feature extraction is done for a Hamming window length of 30 ms with 75 percent overlapping frames. Using the extractor MFCC and GTCC coefficients are returned with delta, deltadelta, slope, flux, centroid, entropy, pitch, harmonic ratio values respectively as seen in Table 3.2. These vectorized features are used for each training coefficient to calculate the mean and standard deviation which is then used to normalize all features. Normalized features are sequenced into 20 time steps with 10 overlaps with their corresponding labels according to which label folder they came from. The outcome which each item being a 45×20 matrix is a cell array of 463464 items for training and 52184 items for validation in 5000 audio files and 1377810 items for training and 147734 items for validation in 15400

audio files. Furthermore patternnet requires these items to be vectorized for any operation where LSTM networks can use them as they are for input layers. Extracted features for training, validation and testing are all using different audio files from same dataset.

Table 3.2. MFCC and GTCC Features [70].

Feature Name	**Number of Coefficients**
MFCC or GTCC	13
MFCC or GTCC Delta	13
MFCC or GTCC DeltaDelta	13
Spectral Slope	1
Spectral Flux	1
Spectral Centroid	1
Spectral Entropy	1
Pitch	1
Harmonic Ratio	1

3.3.4. Patternnet Model

All features are loaded into the MATLAB workspace as a single cell matrix consisting vectorized 900 items for each time steps. For 10 hidden layers and a training function, built in command ”patternnet” is called to create the network [71]. In order to train the network all features are required to be used as inputs according to class targets and these targets need to be set as binary values in different array collums. For ”Bird” target values are set as [1 0] for each time step and [0 1] for all ”No Bird” targets. Network divides the targets into three sets as training validation and testing. By default this dividing operation is done randomly but since our aim is to keep the training and validation features fixed for all operations indices are set manually respectively to number of items in all 3 feature sets. Training features correspond to total of 900×463464 items for 5000 audio files. Training operation can not be done in patternnet with 15400 audio files because MATLAB returns an ”out of memory error” for an input matrix size of 900×1377810. Default settings of

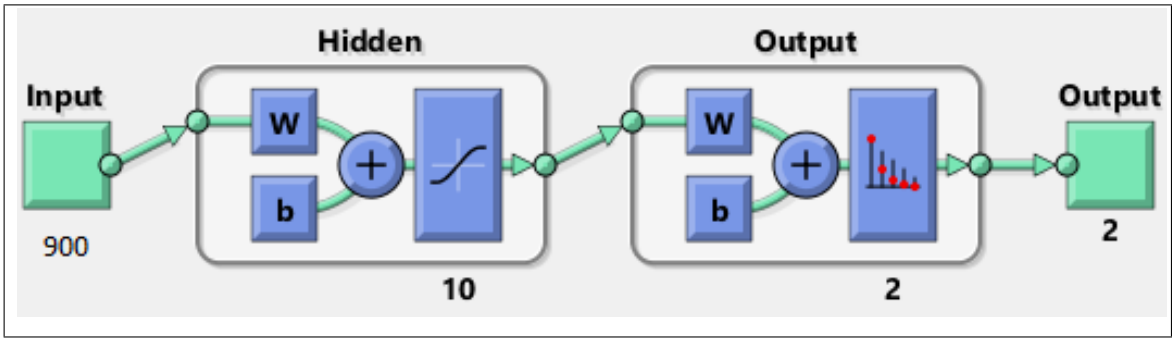

Figure 3.5. Pattern recognition network graphical diagram [72].

patternnet also uses scaled conjugate gradient backpropagation as its training function and cross-entropy as its performance validation function. Softmax outputs neurons and in each hidden layer sigmoid activation function is used as seen in Fig. 3.5.

3.3.5. LSTM Model

There are 2 different LSTM models in our work. First model is a single unidirectional LSTM network based on work done with Japanese vowels data set example on MATLAB [73, 74]. Depending on different number of layers or batch sizes used during the training there are 3 different versions of japan model named japan1, japan2 and japan4. In japan1 model sequence input layer takes the first feature matrix and feeds into lstmlayer with 100 hidden units as output mode sequence. Output of the lstmlayer before feeding into another lstmlayer with 100 hidden units as output mode last is followed by a dropoutlayer with a given probability of 0.1. Last lstmlayer is feed into fullyconnected layer with 2 classes a softmax layer and lastly a classification layer. In japan2 model sequence input layer takes the first feature matrix and feeds into lstmlayer with 100 hidden units as output mode last. There is only a single lstmlayer and no dropoutlayer. In japan2 lstmlayer is feed into fullyconnected layer with 2 classes a softmax layer and lastly a classification layer. In japan4 model layers are structured in the same architecture as japan1 system as seen in Fig. 3.6. Main difference between japan1 and japan4 is that during training options where japan1 and japan 2 is using max epoch size as 4 and mini batch size as 128, japan4 system is using max epoch size as 4 and mini batch size as 8192.

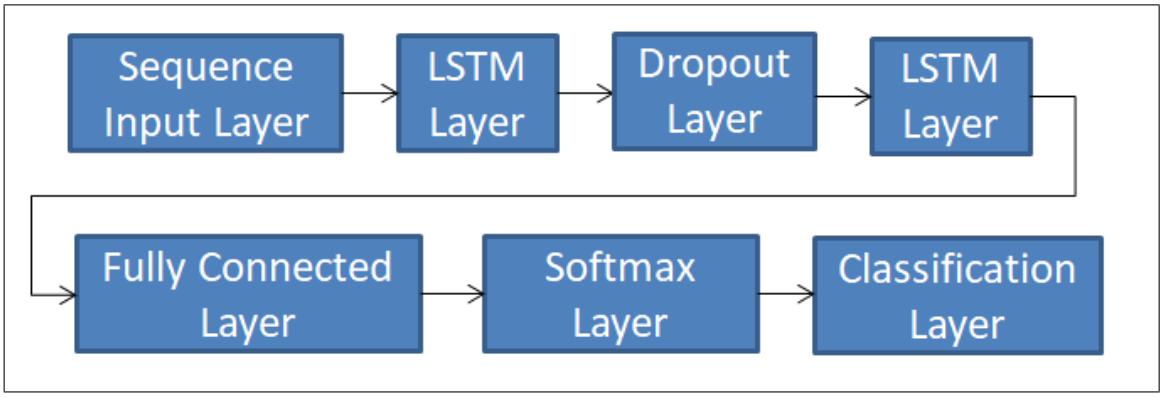

Figure 3.6. Layers of LSTM network model japan4.

Second model is a BiLSTM network which is based on work done with gender data set example on MATLAB [75, 76]. Similar to japan models for comparison there are 3 different BiLSTM network structures called gender1, gender2 and gender4. All gender systems are designed with same principles as in japan1, japan2 and japan4 with only difference being a bilstmlayer rather than a lstmlayer in system architecture as seen in Fig. 3.7.

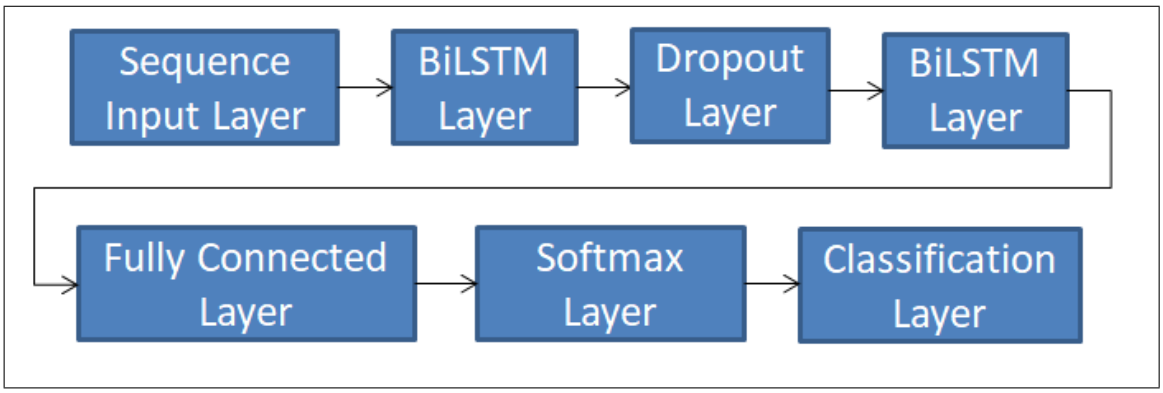

Figure 3.7. Layers of BiLSTM network model gender4.

3.3.5.1. Many to Many and Many to One

In our LSTM and BiLSTM models we combined many to many and many to one architecture during layer design according to our sequence problem type. In many to many applications all time step targets must be known in order to calculate the loss function from the cell output.

In many to one model only the last time steps target is known so only the outcome of the whole sequence can be seen and loss function of the last time step is calculated [77]. For an input vector of 45 features with 20 time steps in many to many and many to one model combined can be seen in Fig. 3.6. In order to connect many to many outputs to many to one input layer we used drop out layer in between layers.

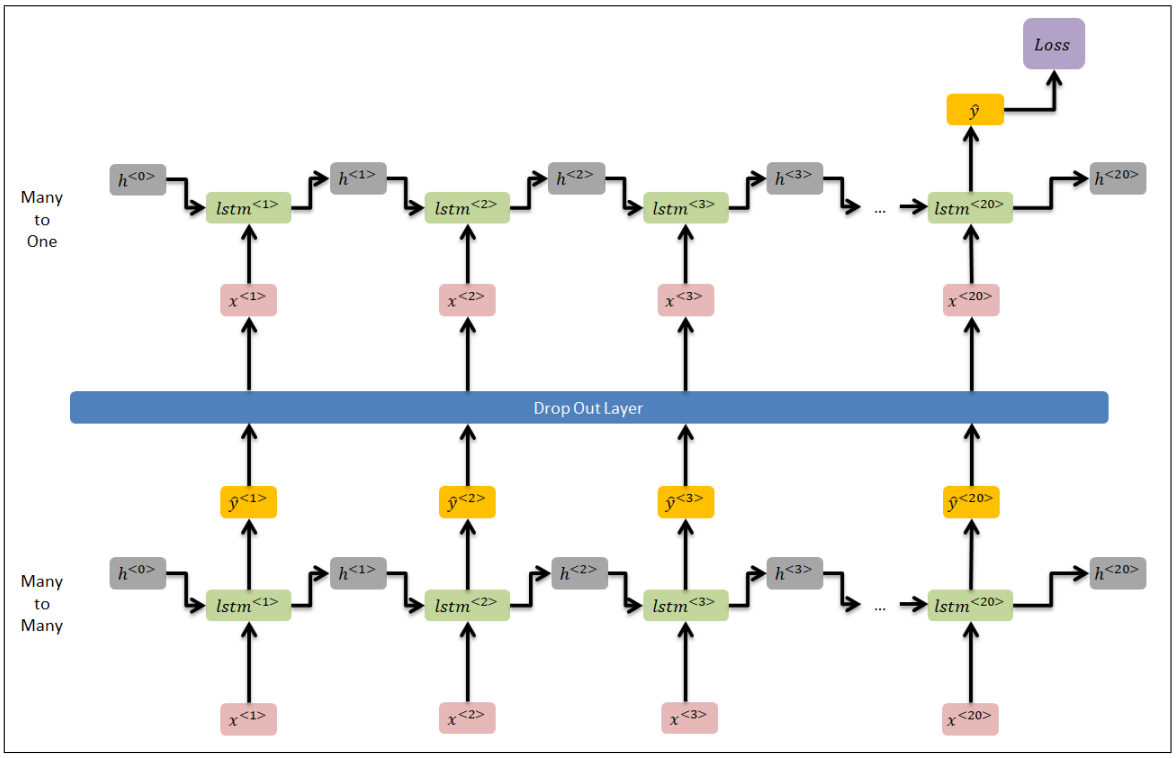

Figure 3.8. Many to many and many to one.

4. RESULTS

In this chapter our experiment results are explained individually according to specific models that are used. Then we present the overall results with each method together.

4.1. CNN RESULTS

Using CNN model 50 evaluations are done with 5000 and 15400 mel spectrogram images. 85 percent of files are taken as training and from the same dataset 10 percent of files are taken as validation. The accuracy plot is seen in Fig. 4.1 and AUC plot is seen in Fig. 4.2. CNN model for audio files dataset 5000 results in accuracy mean of 0.699 while CNN model for audio files dataset 15400 results in accuracy mean of 0.731 during validation. Also CNN model for audio files dataset 5000 performs a mean AUC score of 0.762 while CNN model for audio files dataset 15400 results in mean AUC score of 0.831 during validation.

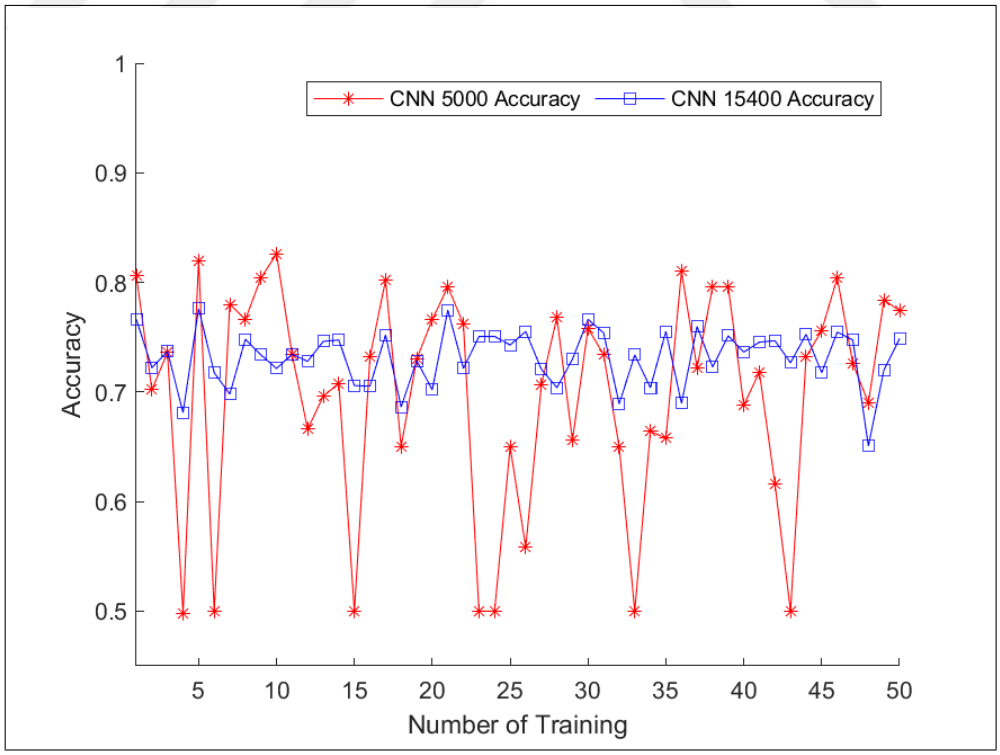

Figure 4.1. CNN accuracy scores for audio files 5000 vs 15400.

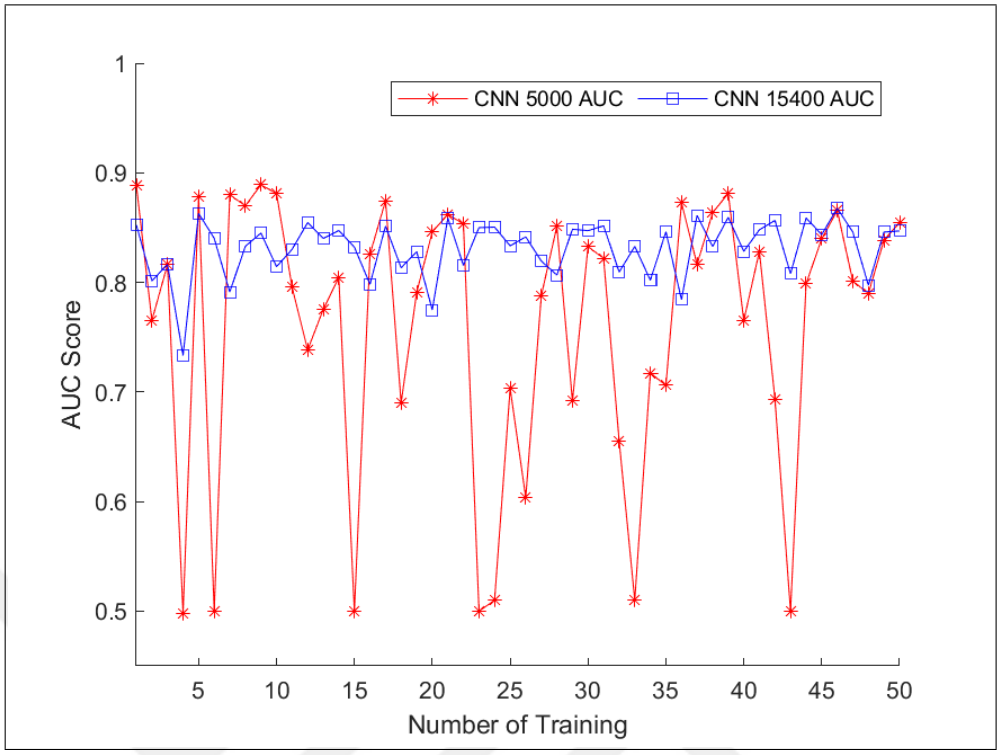

Figure 4.2. CNN AUC scores for audio files 5000 vs 15400.

Validation confusion matrix and ROC graphs for CNN model 5000 evaluation number 2 is seen in Fig. 4.3 and Fig. 4.4 while CNN 15400 evaluation number 1 is seen in Fig. 4.5 and Fig. 4.6. Confusion and ROC figures were selected because they represent a similar accuracy score closer to their mean values. In our CNN model increasing number of audio files in our dataset from 5000 to 15400 hence increasing the number of files for training, improves accuracy and AUC scores as seen in Fig. 4.1 and Fig. 4.2. Out of 250 files labelled as bird CNN 5000 predicts 158 files correctly as bird and 92 files wrongly as no bird while out of 250 files labelled as no bird CNN 5000 predicts 57 files as bird wrongly and 92 files as no bird correctly. Out of 770 files labelled as bird CNN 15400 predicts 584 files correctly as bird and 186 files wrongly as no bird while out of 770 files labelled as no bird CNN 5000 predicts 174 files as bird wrongly and 596 files as no bird correctly.

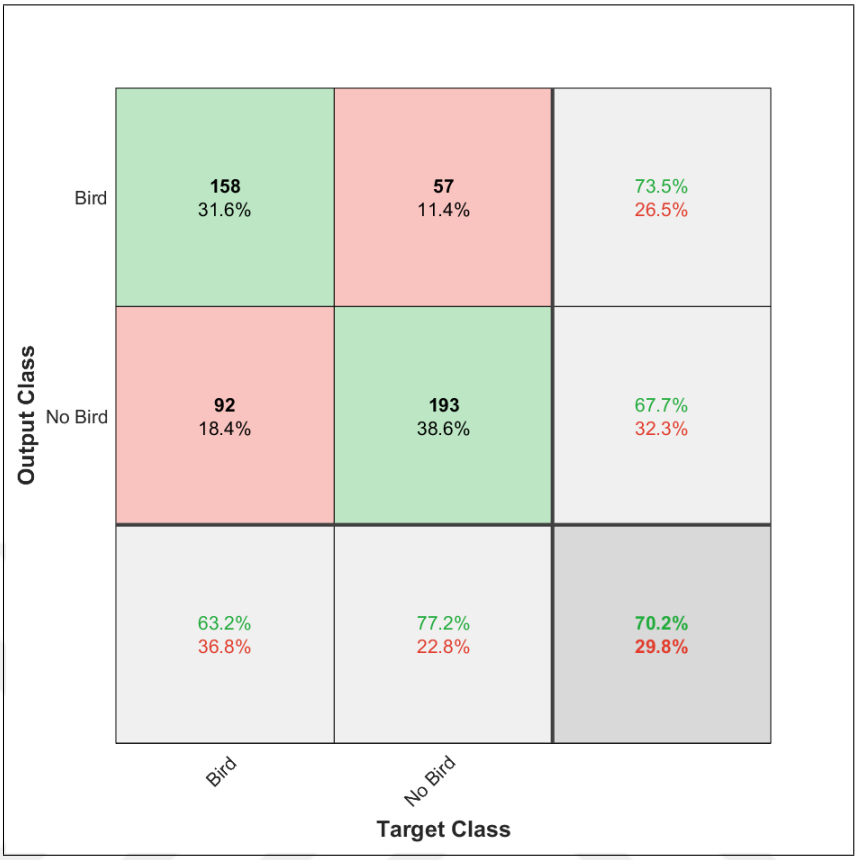

Figure 4.3. CNN 5000 trial 2 validation confusion matrix.

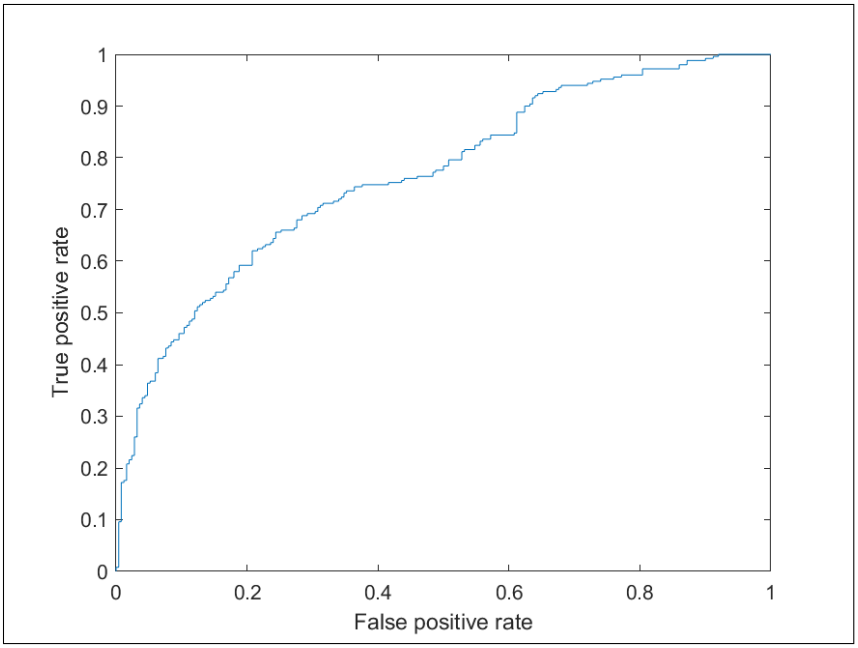

Figure 4.4. CNN 5000 trial 2 validation ROC graph.

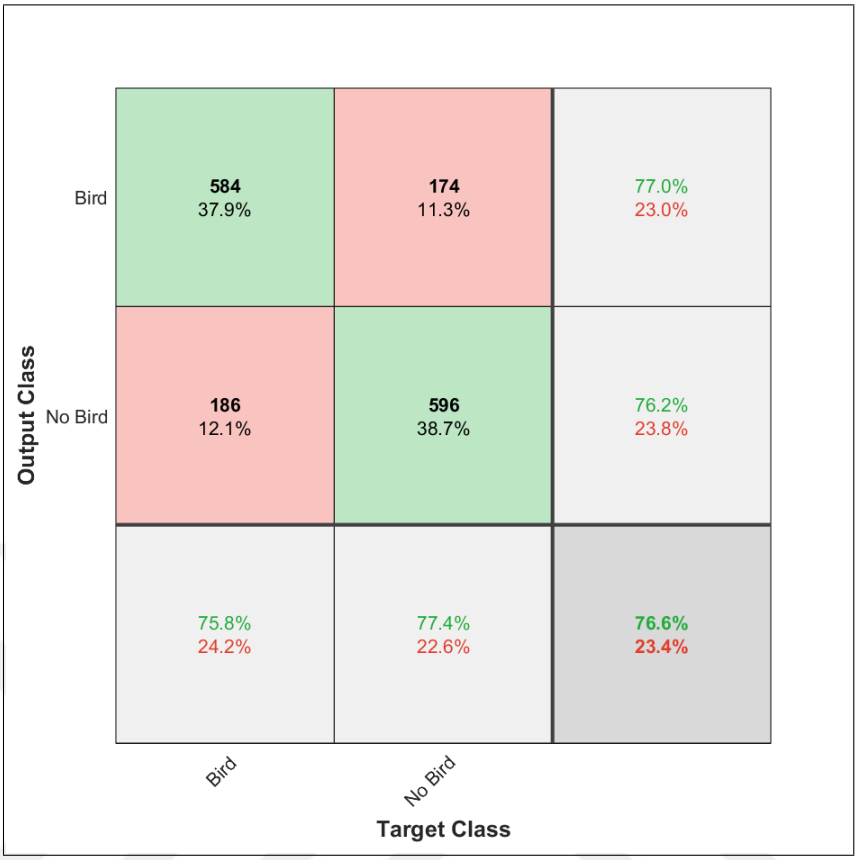

Figure 4.5. CNN 15400 trial 1 validation confusion matrix.

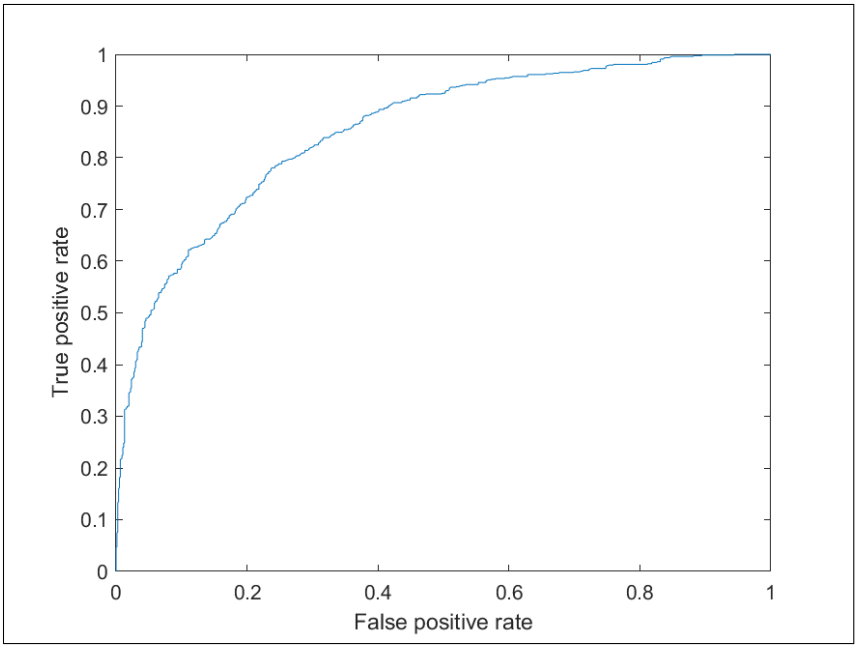

Figure 4.6. CNN 15400 trial 1 validation ROC graph.

Additional test are done with CNN model 15400 in order to investigate the effects of mini batch size parameter on accuracy. Accuracy results in mini batch sizes of 128, 256, 512, 1024, 2048 for an epoch size of 4 effects can be seen in Table 4.1. Our results shows that increasing mini batch size for a fixed number of epoch after some point starts to have negative effects on accuracy in our CNN Model. This is because number of iterations that update the weights during each epoch decreases causing errors in cost function minimization.

Table 4.1. CNN model 15400 different mini batch accuracy means.

Mini Batch 128	**Mini Batch 256**	**Mini Batch 512**	**Mini Batch 1024**	**Mini Batch 2048**
0.729	0.733	0.711	0.698	0.603

There is no mini batch size of 4096 or greater value in our CNN model because further increments in mini batch size decreases number of iteration per epoch and increases memory requirements in MATLAB causing out of memory error. CNN model is the fastest out of all 4 models.

4.2. PATTERNNET RESULTS

Using patternnet model 50 evaluations are done with MFCC and GTCC features for 5000 audio files. 85 percent of files are taken as training and from the same dataset 10 percent of files are taken as validation. The accuracy plot is seen in Fig. 4.7 and AUC plot is seen in Fig. 4.8. Patternnet model for audio files 5000 with MFCC features results in accuracy mean of 0.770 while patternnet model for audio files 5000 with GTCC features results in accuracy mean of 0.818 during validation. Patternnet model for audio files 5000 with MFCC features performs a mean AUC score of 0.851 while patternnet model for audio files 5000 with GTCC features results in mean AUC score of 0.907 during validation.

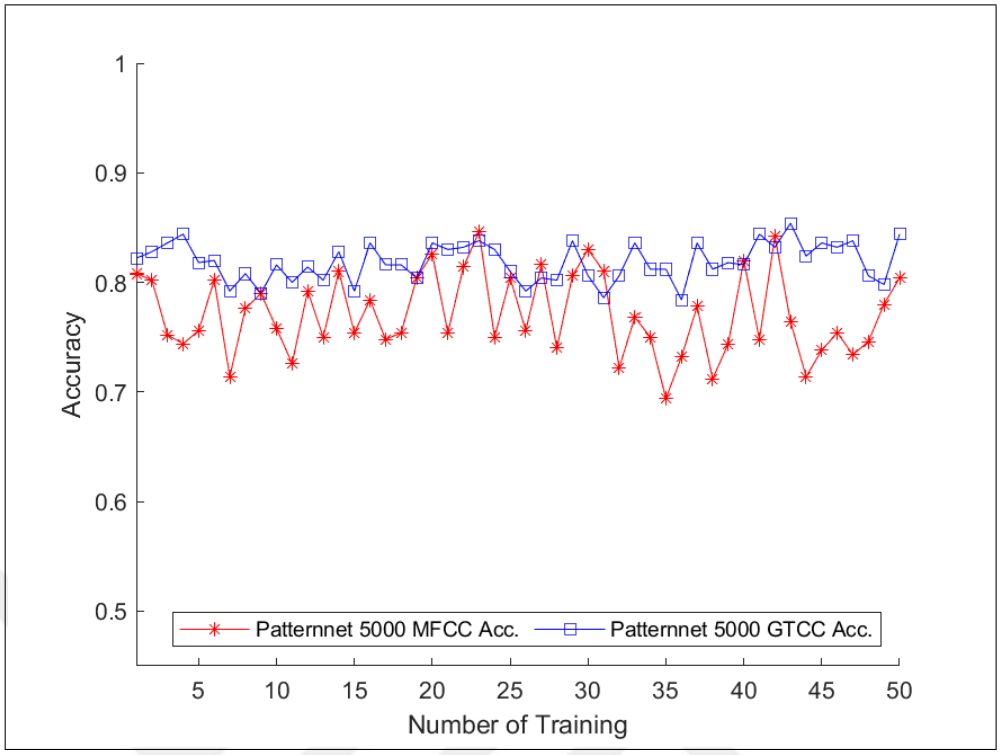

Figure 4.7. Patternnet accuracy values MFCC vs GTCC features for audio files 5000.

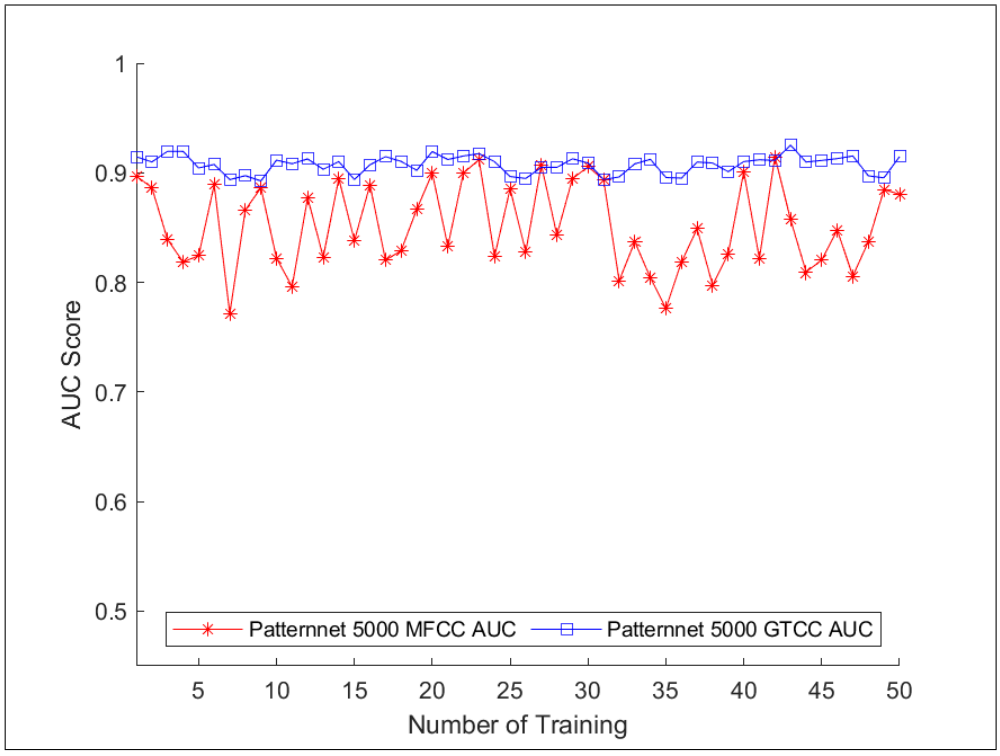

Figure 4.8. Patternnet AUC scores MFCC vs GTCC features for audio files 5000.

Validation confusion matrix and ROC graphs for patternnet model 5000 with MFCC features evaluation number 33 is seen in Fig. 4.9 and Fig. 4.10 while patternnet model 5000 with GTCC features evaluation number 5 is seen in Fig. 4.11 and Fig. 4.12. Confusion and ROC figures were selected because they represent a similar accuracy score closer to their mean values. In our Patternnet model using GTCC features during training compared to MFCC features improves accuracy and AUC scores as seen in Fig. 4.7 and Fig. 4.8. Out of 250 files labelled as bird patternnet 5000 with MFCC features predicts 203 files correctly as bird and 47 files wrongly as no bird while out of 250 files labelled as no bird patternnet 5000 with MFCC features predicts 69 files as bird wrongly and 181 files as no bird correctly. Out of 250 files labelled as bird patternnet 5000 with GTCC features predicts 238 files correctly as bird and 12 files wrongly as no bird while out of 250 files labelled as no bird patternnet 5000 with GTCC features predicts 79 files as bird wrongly and 171 files as no bird correctly.

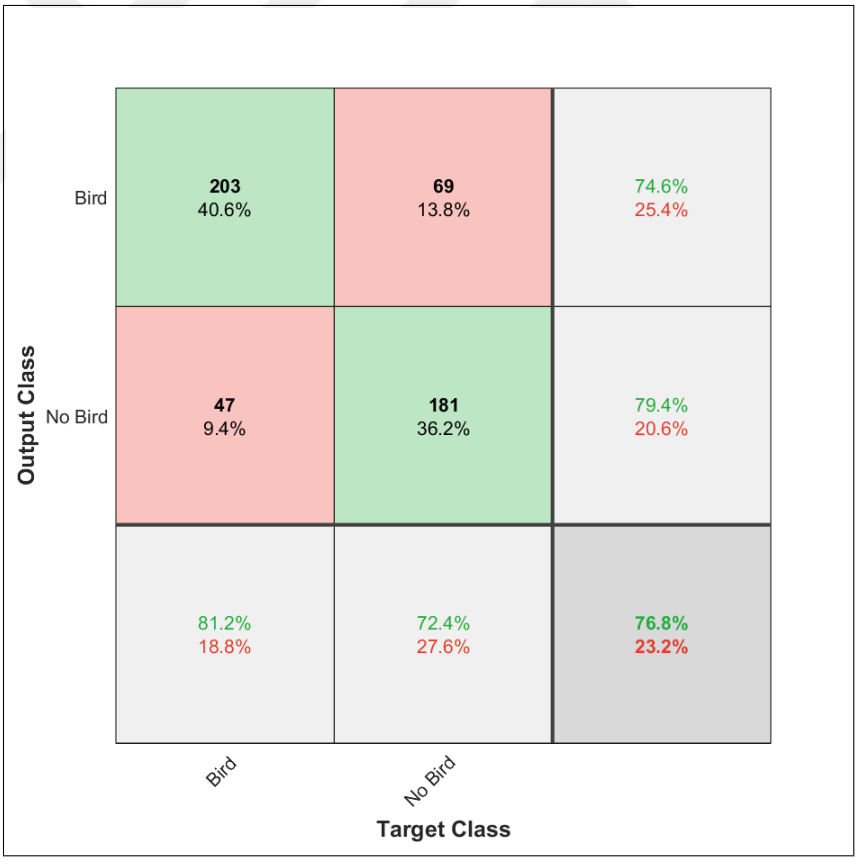

Figure 4.9. Patternnet 5000 MFCC trial 33 validation confusion matrix.

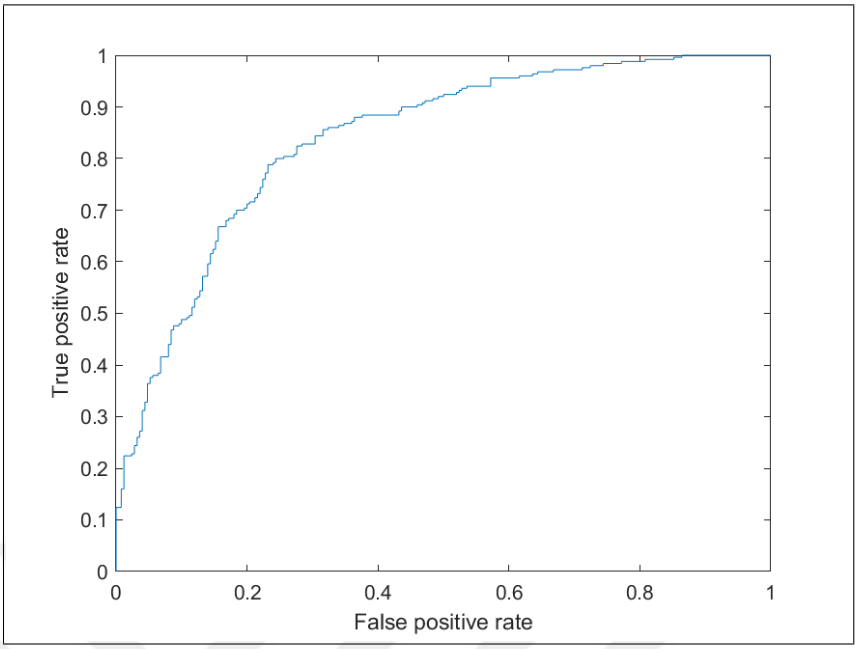

Figure 4.10. Patternnet 5000 MFCC trial 33 validation ROC graph.

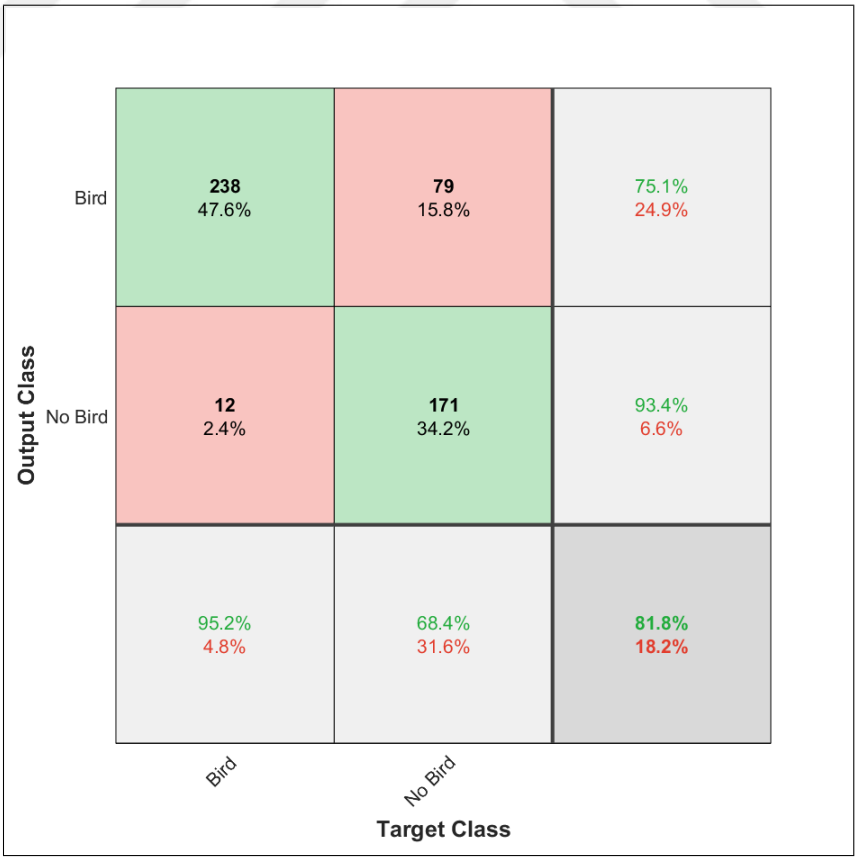

Figure 4.11. Patternnet 5000 GTCC trial 5 validation confusion matrix.

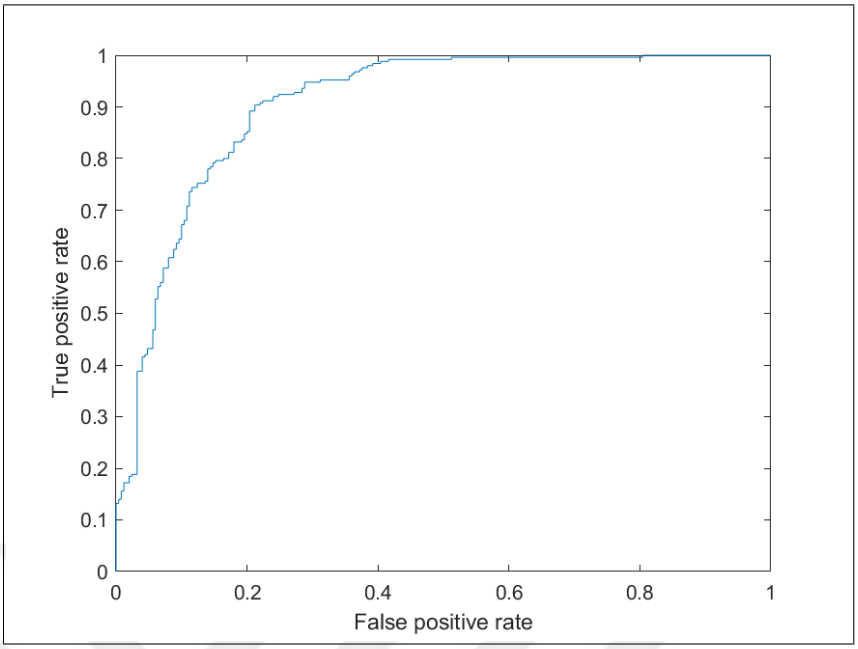

Figure 4.12. Patternnet 5000 GTCC trial 5 validation ROC graph.

There are no results for patternnet with 15400 audio files either with MFCC or GTCC features. During training operations MATLAB returns ”out of memory error” for both feature sets because of large matrix sizes.

4.3. JAPAN4 LSTM RESULTS

Using japan4 unidirectional LSTM model 50 evaluations are done with MFCC and GTCC features for 5000 and 15400 audio files. 85 percent of files are taken as training and from the same dataset percent of files are taken as validation. The accuracy plot is seen in Fig. 4.13 and AUC plot is seen in Fig. 4.14. Japan4 model for audio files 5000 with MFCC features results in accuracy mean of 0.872 while japan4 model for audio files 5000 with GTCC features results in accuracy mean of 0.875 during validation. Japan4 model for audio files 5000 with MFCC features performs a mean AUC score of 0.938 while japan4 model for audio files 5000 with GTCC features results in mean AUC score of 0.941 during validation. Japan4 model for audio files 15400 with MFCC features results in accuracy mean of 0.803 while japan4 model for audio files 15400 with GTCC features results in accuracy mean of

0.792 during validation. Japan4 model for audio files 15400 with MFCC features performs a mean AUC score of 0.905 while japan4 model for audio files 15400 with GTCC features results in mean AUC score of 0.907 during validation.

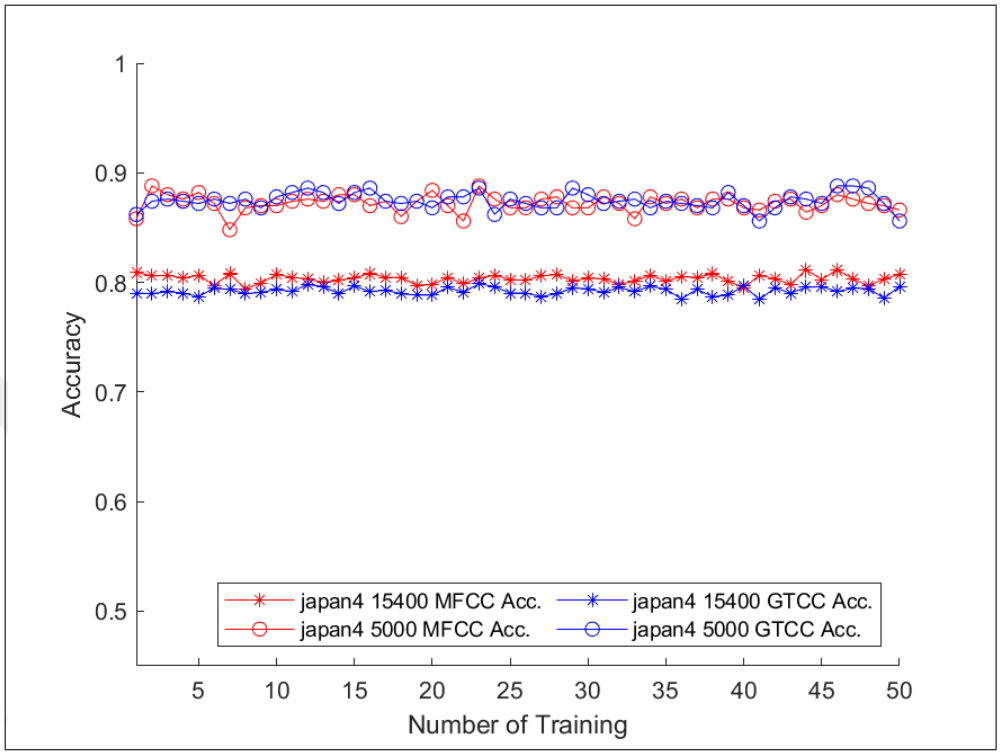

Figure 4.13. Japan4 accuracy scores MFCC vs GTCC for 5000 vs 15400 audio files.

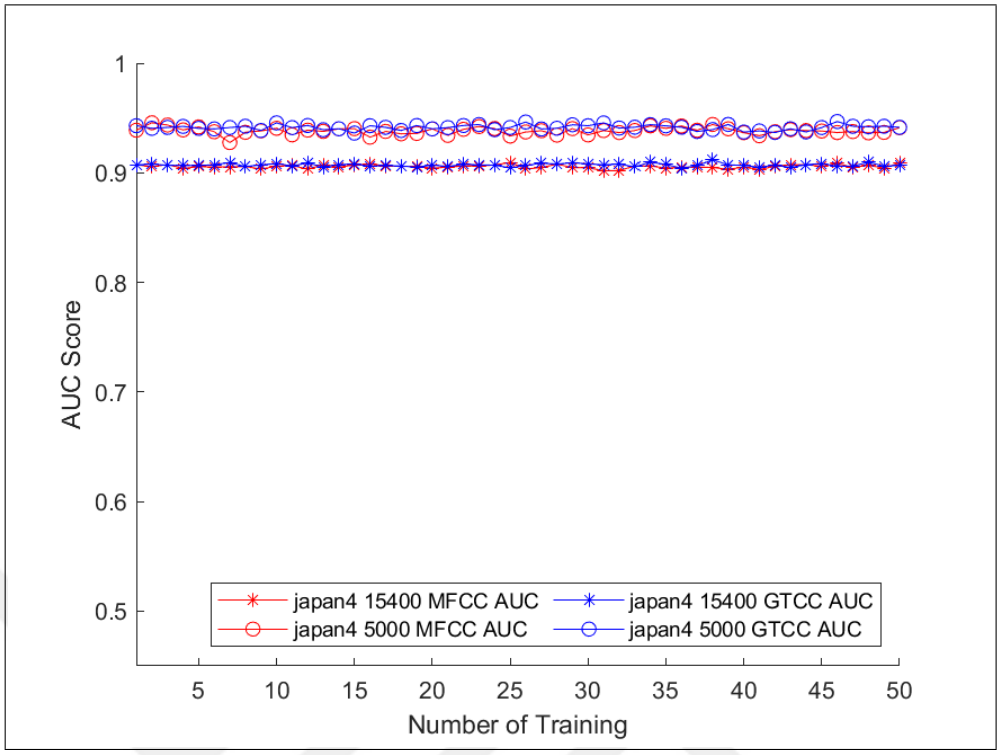

Figure 4.14. Japan4 AUC score MFCC vs GTCC for 5000 vs 15400 audio files.

Validation confusion matrix and ROC graphs for japan4 model 5000 with MFCC features evaluation number 6 is seen in Fig. 4.15 and Fig. 4.16 while japan4 model 5000 with GTCC features evaluation number 2 is seen in Fig. 4.17 and Fig. 4.18. Out of 250 files labelled as bird japan4 5000 with MFCC features predicts 234 files correctly as bird and 16 files wrongly as no bird while out of 250 files labelled as no bird japan4 5000 with MFCC features predicts 48 files as bird wrongly and 202 files as no bird correctly. Out of 250 files labelled as bird japan4 5000 with GTCC features predicts 239 files correctly as bird and 11 files wrongly as no bird while out of 250 files labelled as no bird japan4 5000 with GTCC features predicts 52 files as bird wrongly and 198 files as no bird correctly.

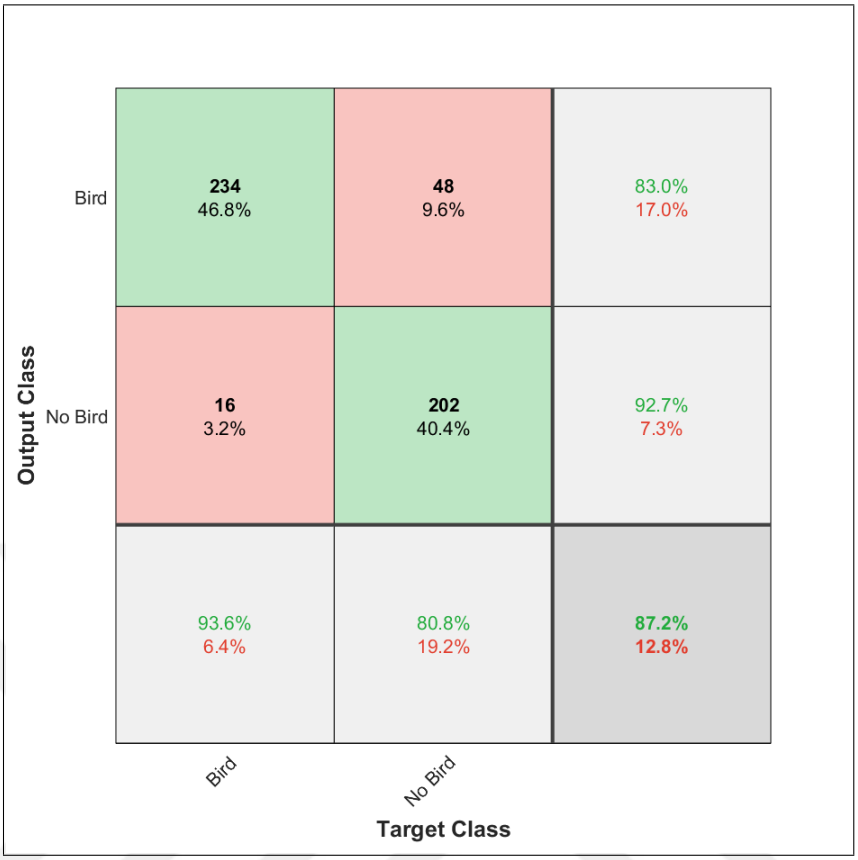

Figure 4.15. Japan4 5000 MFCC trial 6 validation confusion matrix.

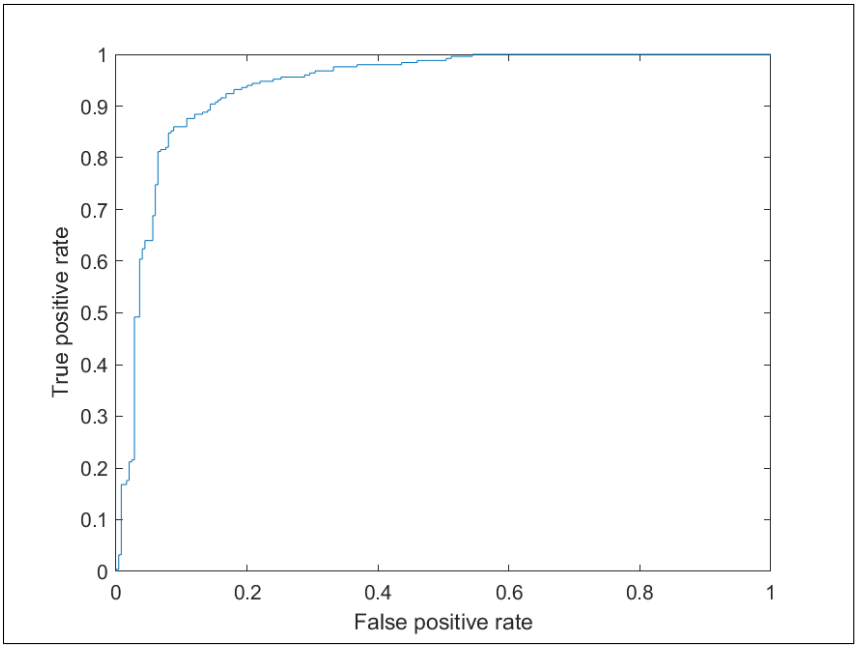

Figure 4.16. Japan4 5000 MFCC trial 6 validation ROC graph.

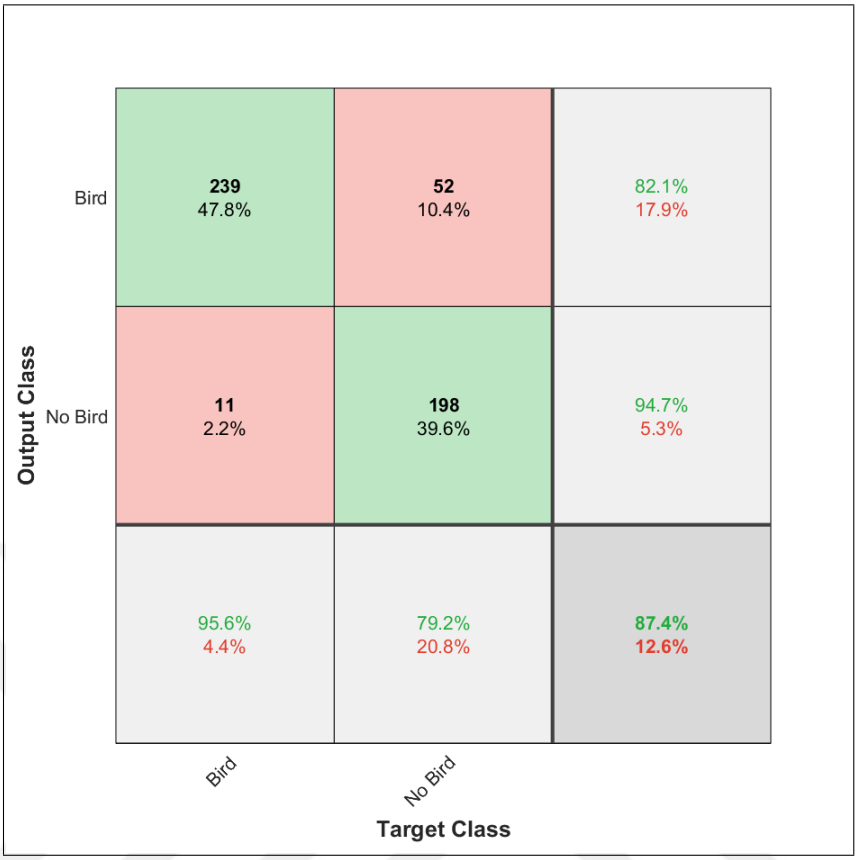

Figure 4.17. Japan4 5000 GTCC trial 5 validation confusion matrix.

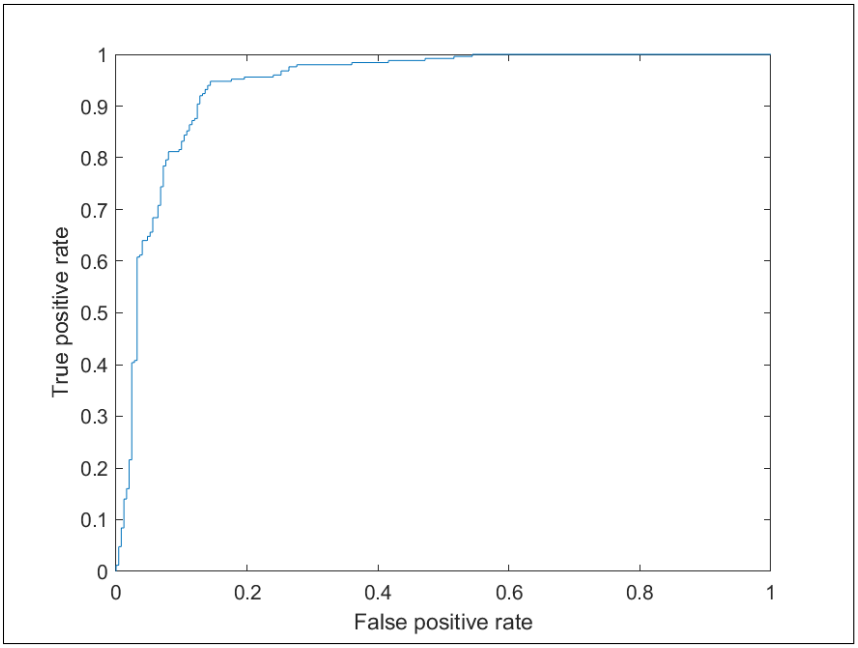

Figure 4.18. Japan4 5000 GTCC trial 5 validation ROC graph.

Validation confusion matrix and ROC graphs for japan4 model 15400 with MFCC features evaluation number 4 is seen in Fig. 4.19 and Fig. 4.20 while japan4 model 15400 with GTCC features evaluation number 3 is seen in Fig. 4.21 and Fig. 4.22. Out of 770 files labelled as bird japan4 15400 with MFCC features predicts 729 files correctly as bird and 41 files wrongly as no bird while out of 770 files labelled as no bird japan4 15400 with MFCC features predicts 261 files as bird wrongly and 509 files as no bird correctly. Out of 770 files labelled as bird japan4 15400 with GTCC features predicts 739 files correctly as bird and 31 files wrongly as no bird while out of 770 files labelled as no bird japan4 15400 with GTCC features predicts 290 files as bird wrongly and 480 files as no bird correctly.

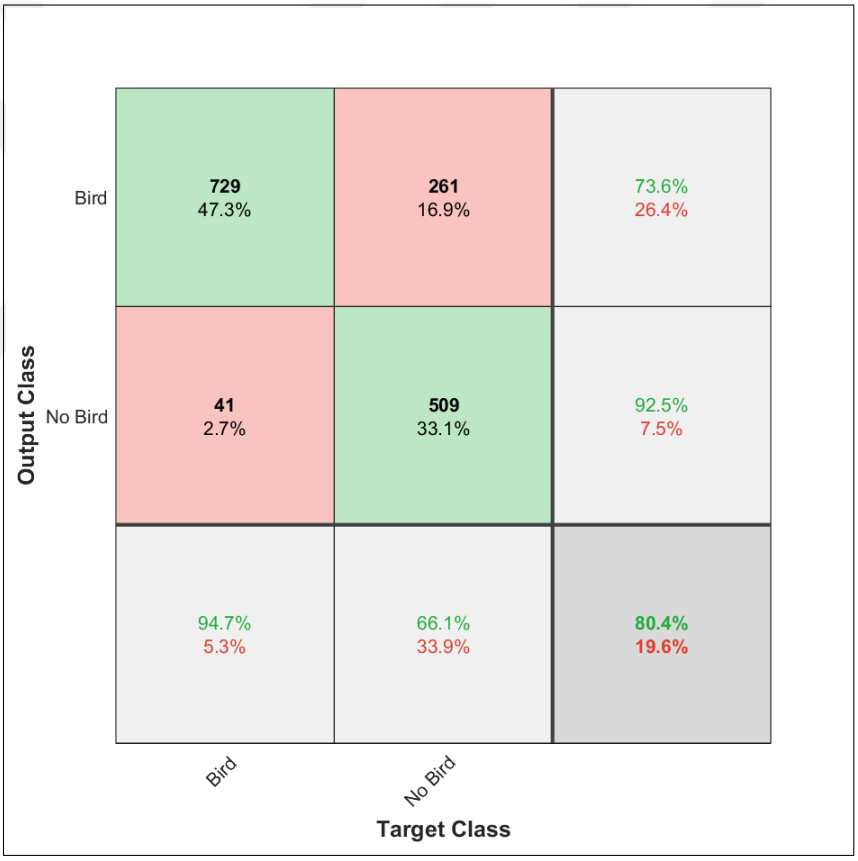

Figure 4.19. Japan4 15400 MFCC trial 4 validation confusion matrix.

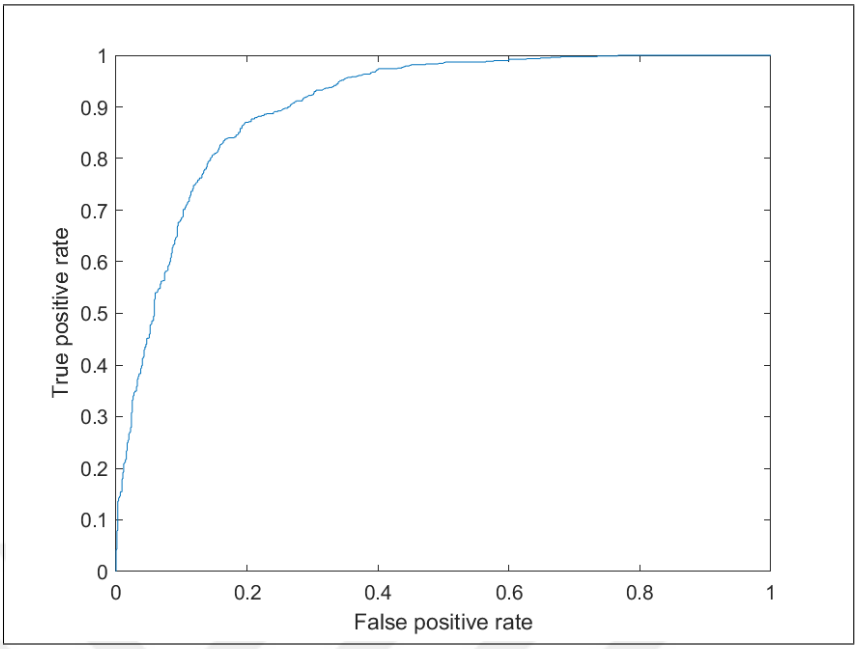

Figure 4.20. Japan4 15400 MFCC trial 4 validation ROC graph.

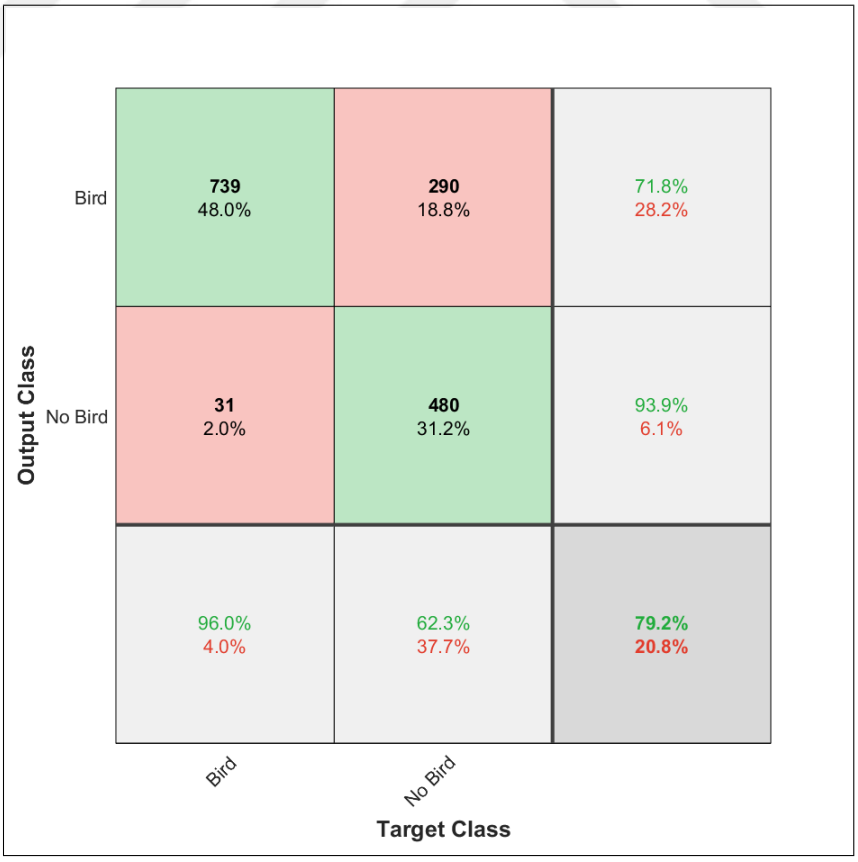

Figure 4.21. Japan4 15400 GTCC trial 3 validation confusion matrix.

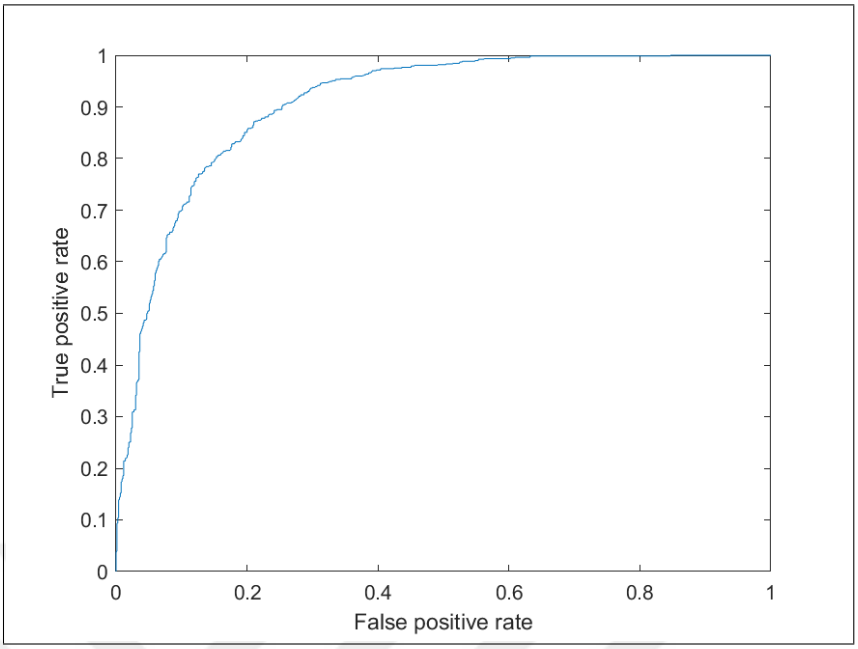

Figure 4.22. Japan4 15400 GTCC trial 3 validation ROC graph.

Confusion and ROC figures were selected because they represent a similar accuracy score closer to their mean values. In our japan4 model using GTCC features during training compared to MFCC features does not change accuracy or AUC scores but increasing number of audio files in our dataset from 5000 to 15400 hence increasing the number of files for training, decreases accuracy and AUC scores as seen in Fig. 4.13 and Fig. 4.14.

4.4. GENDER4 BILSTM RESULTS

Using gender4 bidirectional LSTM model 50 evaluations are done with MFCC and GTCC features for 5000 and 15400 audio files. 85 percent of files are taken as training and from the same dataset 10 percent of files are taken as validation. The accuracy plot is seen in Fig. 4.23 and AUC plot is seen in Fig. 4.24. Gender4 model for audio files 5000 with MFCC features results in accuracy mean of 0.871 while gender4 model for audio files 5000 with GTCC features results in accuracy mean of 0.875 during validation. gender4 model for audio files 5000 with MFCC features performs a mean AUC score of 0.938 while gender4 model for audio files 5000 with GTCC features results in mean AUC score of 0.942 during validation.

Gender4 model for audio files 15400 with MFCC features results in accuracy mean of 0.804 while gender4 model for audio files 15400 with GTCC features results in accuracy mean of 0.793 during validation. Gender4 model for audio files 15400 with MFCC features performs a mean AUC score of 0.907 while gender4 model for audio files 15400 with GTCC features results in mean AUC score of 0.908 during validation.

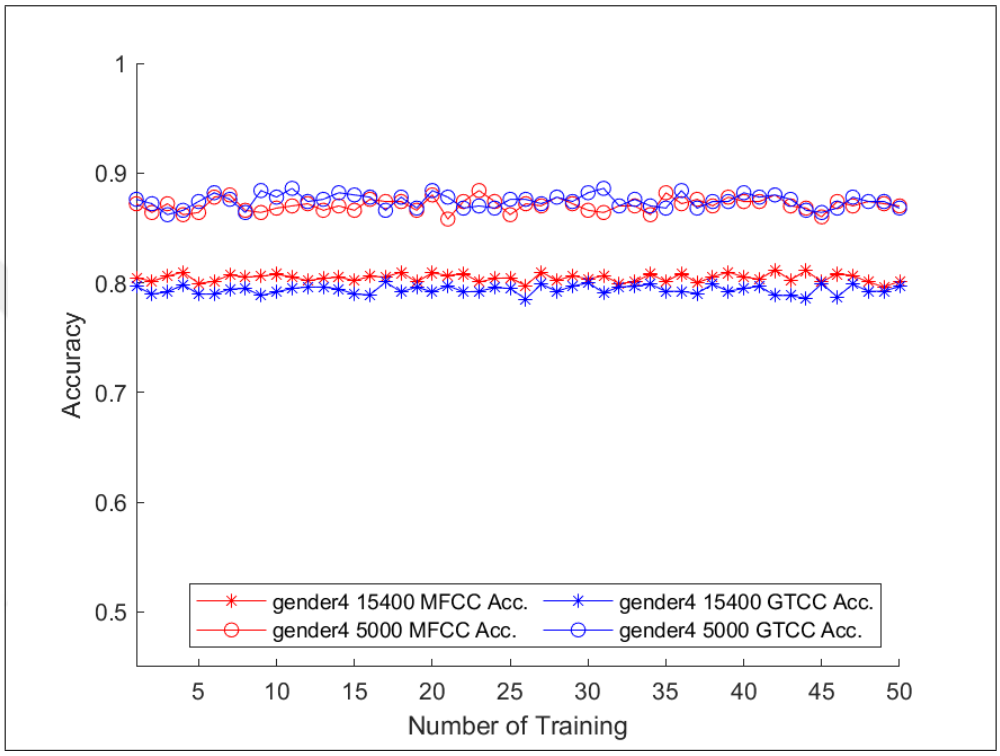

Figure 4.23. Gender4 accuracy scores MFCC vs GTCC for 5000 vs 15400 audio files.

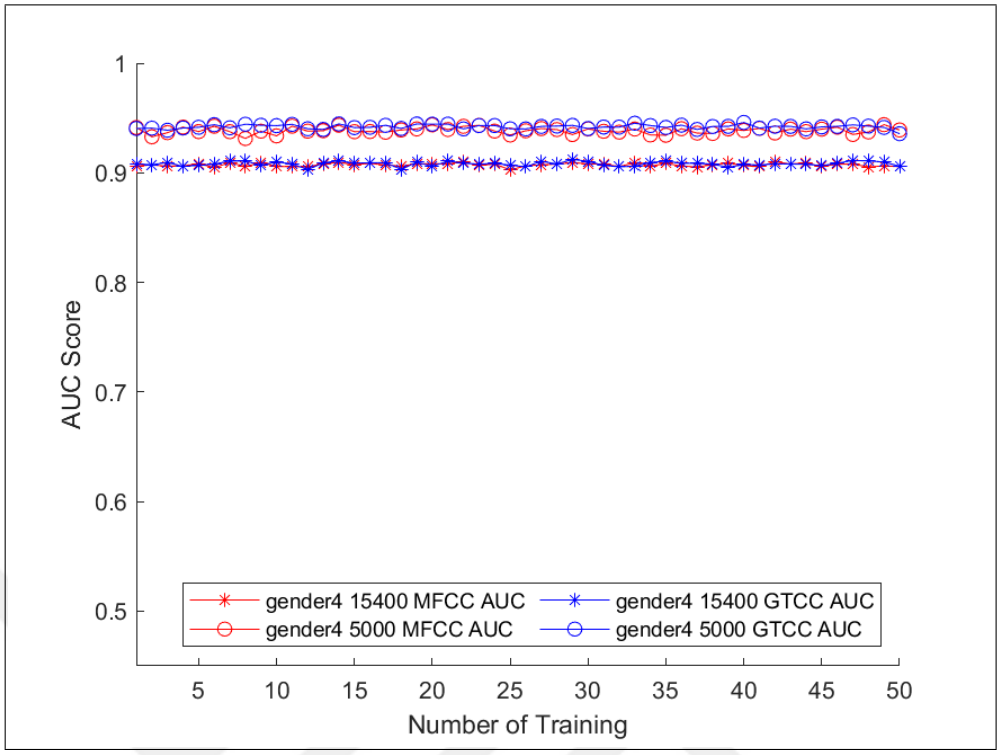

Figure 4.24. Gender4 AUC scores MFCC vs GTCC for 5000 vs 15400 audio files.

Validation confusion matrix and ROC graphs for gender4 model 5000 with MFCC features evaluation number 1 is seen in Fig. 4.25 and Fig. 4.26 while gender4 model 5000 with GTCC features evaluation number 5 is seen in Fig. 4.27 and Fig. 4.28. Out of 250 files labelled as bird gender4 5000 with MFCC features predicts 236 files correctly as bird and 14 files wrongly as no bird while out of 250 files labelled as no bird gender4 5000 with MFCC features predicts 50 files as bird wrongly and 200 files as no bird correctly. Out of 250 files labelled as bird gender4 5000 with GTCC features predicts 239 files correctly as bird and 11 files wrongly as no bird while out of 250 files labelled as no bird gender4 5000 with GTCC features predicts 52 files as bird wrongly and 198 files as no bird correctly.

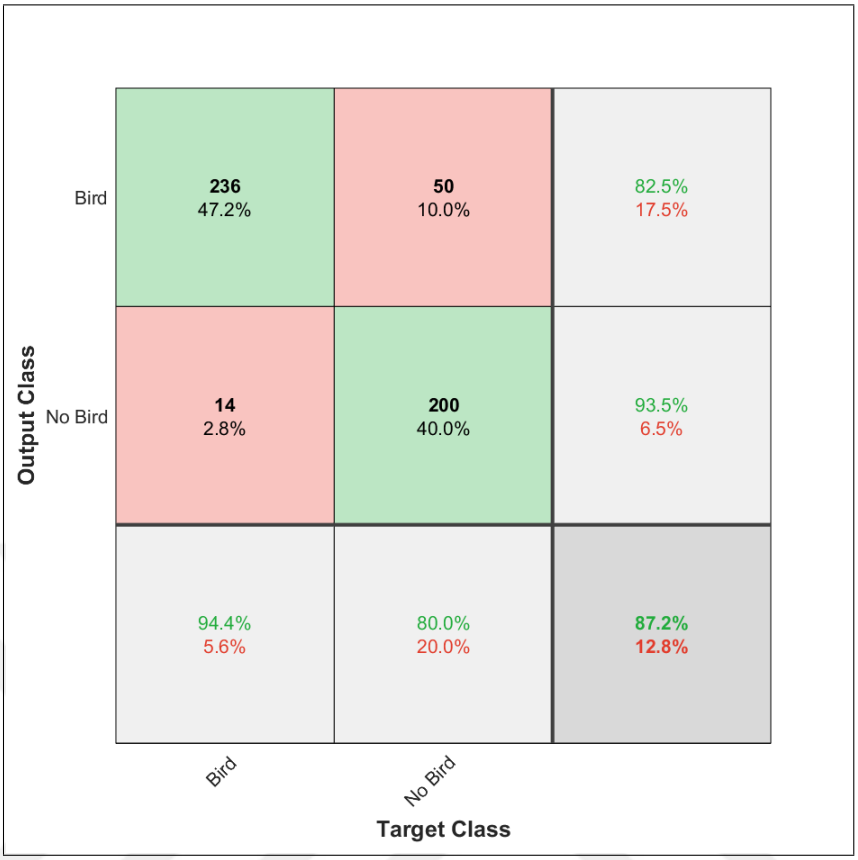

Figure 4.25. Gender4 5000 MFCC trial 1 validation confusion matrix.

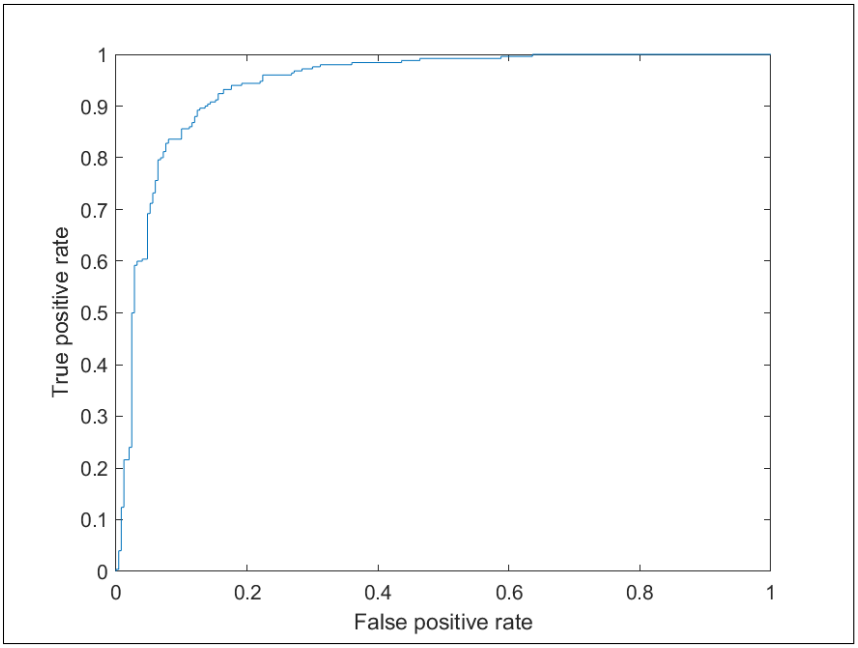

Figure 4.26. Gender4 5000 MFCC trial 1 validation ROC graph.

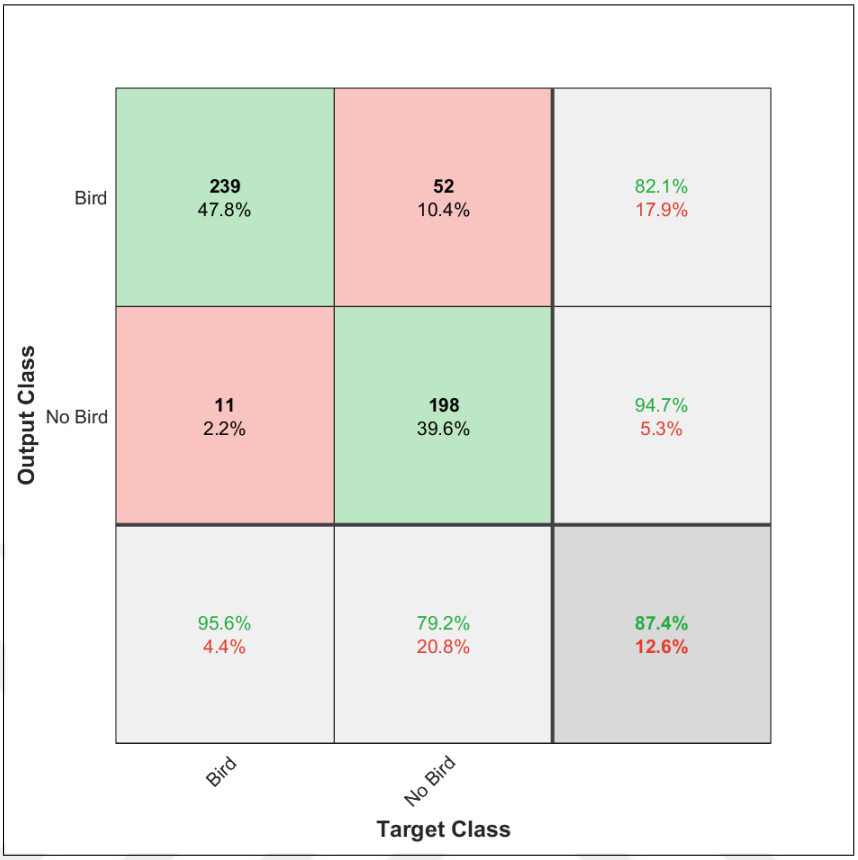

Figure 4.27. Gender4 5000 GTCC trial 5 validation confusion matrix.

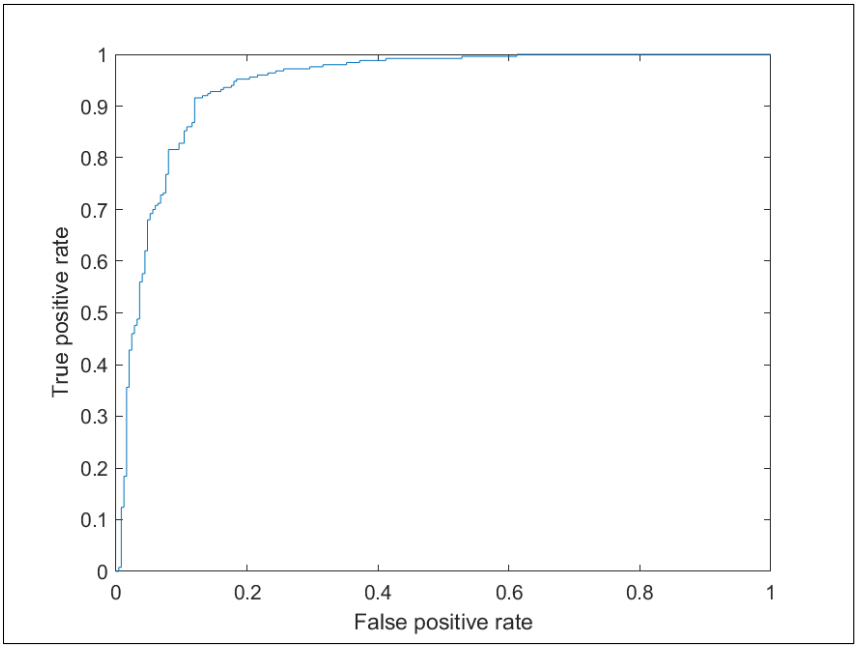

Figure 4.28. Gender4 5000 GTCC trial 5 validation ROC graph.

Validation confusion matrix and ROC graphs for gender4 model 15400 with MFCC features evaluation number 1 is seen in Fig. 4.29 and Fig. 4.30 while gender4 model 15400 with GTCC features evaluation number 23 is seen in Fig. 4.31 and Fig. 4.32. Out of 770 files labelled as bird gender4 15400 with MFCC features predicts 721 files correctly as bird and 49 files wrongly as no bird while out of 770 files labelled as no bird gender4 15400 with MFCC features predicts 252 files as bird wrongly and 518 files as no bird correctly. Out of 770 files labelled as bird gender4 15400 with GTCC features predicts 745 files correctly as bird and 25 files wrongly as no bird while out of 770 files labelled as no bird gender4 15400 with GTCC features predicts 295 files as bird wrongly and 475 files as no bird correctly.

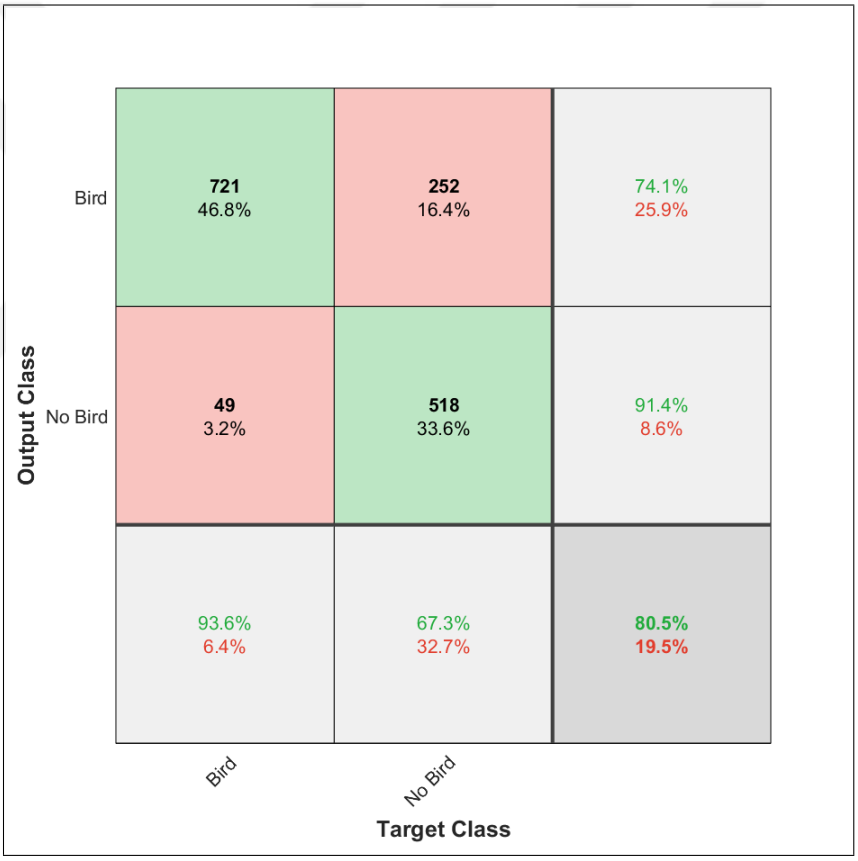

Figure 4.29. Gender4 15400 MFCC trial 1 validation confusion matrix.

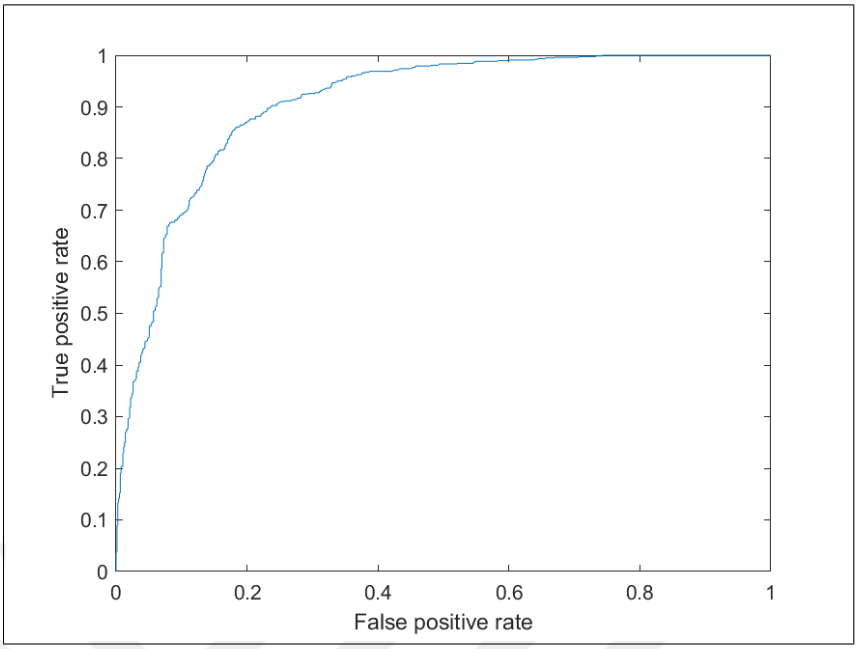

Figure 4.30. Gender4 15400 MFCC trial 1 validation ROC graph.

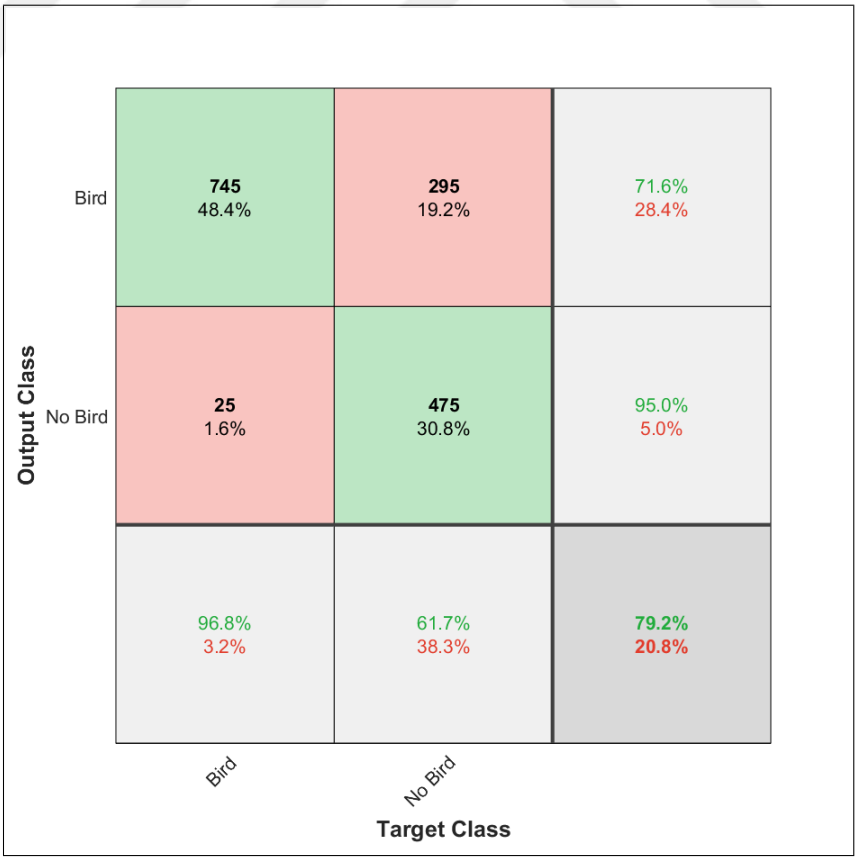

Figure 4.31. Gender4 15400 GTCC trial 23 validation confusion matrix.

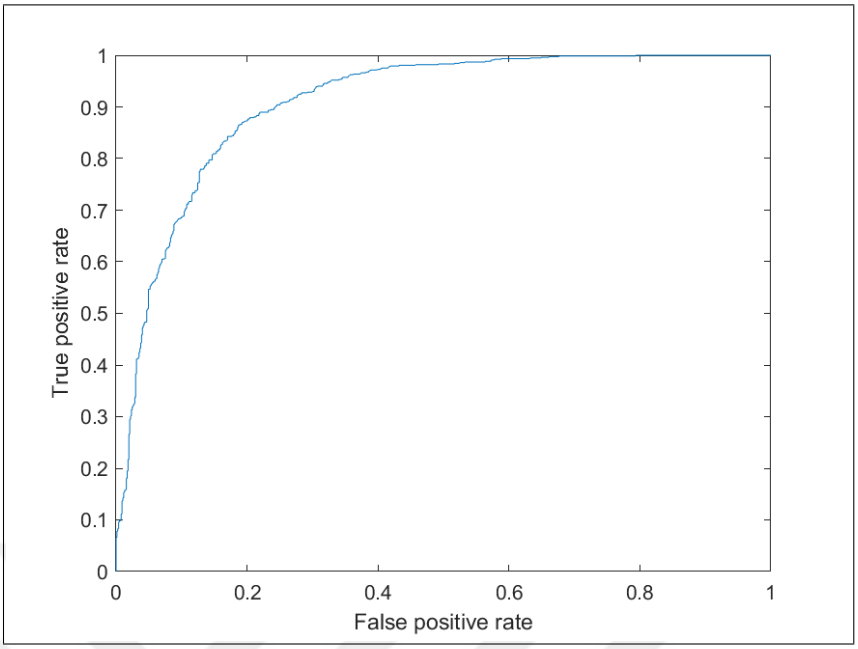

Figure 4.32. Gender4 15400 GTCC trial 23 validation ROC graph.

Confusion and ROC figures were selected because they represent a similar accuracy score closer to their mean values. In our gender4 model using GTCC features during training compared to MFCC features does not change accuracy or AUC scores but increasing number of audio files in our dataset from 5000 to 15400 hence increasing the number of files for training, decreases accuracy and AUC scores as seen in Fig. 4.23 and Fig. 4.23.

Overall results for both 5000 and 15400 audio files with which method is used with respective settings and layer sizes are as seen in Table 4.2 and Table 4.3. In Table 4.2 and Table 4.3 CNN model performs as the worst case in merged scores compared to patternnet, LSTM and BiLSTM methods where patternnet also falls behind both LSTM and BiLSTM models. Compared to MFCC, in both segmented and merged scores using GTCC improves patternnet scores in Table 4.2. There are no results for patternnet in Table 4.3 because of big training set causing out of memory. Even though improvements are seen in segmented scores when GTCC features are used, these improvements are not parallel in merged scores for LSTM and BiLSTM models. Classification results obtained with patternet are less than LSTM models because features in patternnet are used in vector formation, loosing the dependency between

sequential time steps, in LSTM models each time step is fed to corresponding input as a separate vector. In both tables merging the validation segments increase AUC and accuracy score for all models where it is possible to do so. There is a drop in scores between Table 4.2 and Table 4.3 but this drop does not reflect to effects of number of training because these values are gathered when number of training files are increased so are the number of validation files. For detailed effects of number of training files on fixed validation numbers please refer to comparative results and Table 4.4.

Table 4.2. Results for 5000 audio files.

Audio Files	**Feauture Type**	**Network Type**	**Epoch/ Mini Batch**	**Solver Type**	**Number of Layers**	**Layer Size Hidden Unit**	**Number of Training Runs**	**Accuracy and AUC Segmented**	**Accuracy and AUC Merged**
5000	Mel Spec.	CNN	4/128	sgdm 0.01	3 × CNN	8,16,32	50	-, -	0.700, 0.762
5000	MFCC	CNN	4/2048	sgdm 0.01	3 × CNN	8,16,32	10	0.770, 0.861	0.854, 0.936
5000	MFCC	patternnet	-	trainscg	-	10	50	0.732, 0.803	0.770, 0.851
5000	MFCC	japan1	4/128	adam 0.001	2 × LSTM	100,100	10	0.803, 0.888	0.872, 0.933
5000	MFCC	japan2	4/128	adam 0.001	1 × LSTM	100	10	0.800, 0.886	0.866, 0.929
5000	MFCC	japan4	4/8192	adam 0.001	2 × LSTM	100,100	50	0.823, 0.900	0.872, 0.938
5000	MFCC	gender1	4/128	adam 0.001	2 × BiLSTM	100,100	10	0.793, 0.880	0.858, 0.926
5000	MFCC	gender2	4/128	adam 0.001	1 × BiLSTM	100	10	0.802, 0.889	0.865, 0.933
5000	MFCC	gender4	4/8192	adam 0.001	2 × BiLSTM	100,100	50	0.824, 0.902	0.871, 0.939
5000	GTCC	patternnet	-	trainscg	-	10	50	0.803, 0.884	0.819, 0.907
5000	GTCC	japan1	4/128	adam 0.001	2 × LSTM	100,100	10	0.825, 0.911	0.873, 0.942
5000	GTCC	japan2	4/128	adam 0.001	1 × LSTM	100	10	0.823, 0.910	0.870, 0.940
5000	GTCC	japan4	4/8192	adam 0.001	2 × LSTM	100,100	50	0.848, 0.922	0.875, 0.941
5000	GTCC	gender1	4/128	adam 0.001	2 × BiLSTM	100,100	10	0.824, 0.912	0.873, 0.940
5000	GTCC	gender2	4/128	adam 0.001	1 × BiLSTM	100	10	0.823, 0.911	0.875, 0.940
5000	GTCC	gender4	4/8192	adam 0.001	2 × BiLSTM	100,100	50	0.849, 0.923	0.875, 0.942

Table 4.3. Results for 15400 audio files.

Audio Files	Feauture Type	Network Type	Epoch/ Mini Batch	Solver Type	Number of Layers	Layer Size Hidden Unit	Number of Training Runs	Accuracy and AUC Segmented	Accuracy and AUC Merged
15400	Mel Spec.	CNN	4/128	sgdm 0.01	3 × CNN	8,16,32	50	-, -	0.731, 0.831
15400	MFCC	CNN	-	-	-	-	-	-, -	-, -
15400	MFCC	patternnet	-	-	-	-	-	-, -	-, -
15400	MFCC	japan1	4/128	adam 0.001	2 × LSTM	100,100	10	0.772, 0.868	0.803, 0.909
15400	MFCC	japan2	4/128	adam 0.001	1 × LSTM	100	10	0.774, 0.867	0.808, 0.909
15400	MFCC	japan4	4/8192	adam 0.001	2 × LSTM	100,100	50	0.773, 0.865	0.803, 0.906
15400	MFCC	gender1	4/128	adam 0.001	2 × BiLSTM	100,100	10	0.774, 0.869	0.805, 0.909
15400	MFCC	gender2	4/128	adam 0.001	1 × BiLSTM	100	13	0.774, 0.867	0.809, 0.909
15400	MFCC	gender4	4/8192	adam 0.001	2 × BiLSTM	100,100	50	0.775, 0.868	0.805, 0.907
15400	GTCC	patternnet	-	-	-	-	-	-, -	-, -
15400	GTCC	japan1	4/128	adam 0.001	2 × LSTM	100,100	10	0.773, 0.869	0.798, 0.905
15400	GTCC	japan2	4/128	adam 0.001	1 × LSTM	100	10	0.773, 0.869	0.801, 0.906
15400	GTCC	japan4	4/8192	adam 0.001	2 × LSTM	100,100	0	0.774, 0.873	0.792, 0.907
15400	GTCC	gender1	4/128	adam 0.001	2 × BiLSTM	100,100	10	0.774, 0.871	0.799, 0.907
15400	GTCC	gender2	4/128	adam 0.001	1 × BiLSTM	100	10	0.772, 0.869	0.798, 0.905
15400	GTCC	gender4	4/8192	adam 0.001	2 × BiLSTM	100,100	50	0.774, 0.875	0.794, 0.908

4.5. COMPARATIVE RESULTS

To understand the effects of number of training files and training features on accuracy and AUC score, all systems are trained with 85 percent of their total audio sizes which corresponds to 4250 files for 5000 and 13090 files for 15400. Testing these systems with same 1540 files yields accuracy and AUC scores as seen in Table 4.4 for 50 evaluations.

Table 4.4. Comparison table for 1540 test files.

Training Files	**Test Files**	**Feauture Type**	**Network Type**	**Network Runs**	**Accuracy and AUC Merged**
4250	1540	Mel Spec.	CNN	50	0.650, 0.716
4250	1540	MFCC	patternnet	50	0.660, 0.738
4250	1540	MFCC	japan4	50	0.787, 0.883
4250	1540	MFCC	gender4	50	0.786, 0.885
4250	1540	GTCC	patternnet	50	0.730, 0.821
4250	1540	GTCC	japan4	50	0.804, 0.887
4250	1540	GTCC	gender4	50	0.805, 0.887
13090	1540	Mel Spec.	CNN	50	0.731, 0.831
13090	1540	MFCC	japan4	50	0.803, 0.906
13090	1540	MFCC	gender4	50	0.805, 0.907
13090	1540	GTCC	japan4	50	0.792, 0.907
13090	1540	GTCC	gender4	50	0.794, 0.908

As seen in Table 4.4, CNN model is still the worst case compared to all other models. Patternnet shows great improvement compared to MFCC features when GTCC features are used. It is not possible to use patternnet with big training files because of out of memory. Even though CNN model scores are increased for big number of training files, LSTM and BiLSTM models show no important change. In LSTM and BiLSTM there is no score difference for MFCC and GTCC features.

5. DISCUSSIONS

In this thesis we compared 4 different machine learning methods by combining the work done in literature on DCASE dataset which was mainly on CNN or CNN+RNN applications with speech detection methods proposed using LSTM [2, 75]. We succeeded better with LSTM applications but were not able to improve our results using BiLSTM model like in some cases in literature with different datasets [78].

In our work with CNN model we used raw data and recieved lower results then our LSTM model but application of preprocessing on audio files before mel spectrogram images were created to be used on CNN model might improve our results [79]. This might cause improvement to CNN models. Before feature extraction a better performing bird call detector might also improve our chances of bird related weight learning during training. Using MATLAB as our environment proved its own challenge in terms of toolbox expenses and memory usage. MATLAB applications rely heavy on hardware specially on memory and current deep learning methods use graphical processing units which are expensive but worked faster than central processing unit applications [80].

Time dependencies were captured with LSTM models which were better for audio where CNN models were more applicable with image data [81]. LSTM networks replaced other conventional methods, given the fact that RNN and LSTM methods utilize time dependence of the audio samples and achieves ground breaking results [30]. LSTM model efficiency compared to CNN model was much higher in our dataset applications [17, 18, 20–23].

For bird call detection conventional methods in the literature were focused mainly on MFCC, two dimensional coefficients or pitch levels where in our work we compared MFFC to GTCC extracted features and proved that GTCC showed little improvement on our datasetes [12–16].

Our classification results show that both the LSTM and BiLSTM networks outperform the patternet and CNN in terms of accuracy and AUC scores. In our current hardware set up

patternnet model caused out of memory making it inefficient with big data applications proving once more the LSTM models were better with audio applications.

6. CONCLUSION

In this thesis, our main purpose was to apply methods from the literature working on speech or image detection, in order to determine the best performing method for bird call detection in an audio. We compared 4 different machine learning methods on our dataset. Our methods contained a convolutional neural network, a simple neural network called patternnet, an unidirectional long short term memory network and a bidirectional long short term memory network.

First we adapted a speech detection based algorithm on our datasets to find the bird call segments in audio files for given energy and centroid thresholds. From these bird call segments, mel frequency cepstral coefficients and gammatone frequency cepstral coefficients were extracted. Extracted features were normalized according to mean and standard deviation. We adapted these features to be used as inputs in basic machine learning algorithm called patternet, unidirectional long short term memory network and bidirectional long short term memory network training and validation. Second we transformed our raw audio files into images using mel spectrogram. These mel images were used as inputs in convolutional neural network training and validation. In CNN and LSTM applications, to find best optimization we experimented with different parameters and layer sizes during training step. All models were validated using different number of audio files and then for same number of audio files. Confusion matrix and area under the receiver operating characteristic values were calculated.

Our results showed that compared to conventional methods machine learning methods performed better. For training with both low and high number of files CNN method was good for speed but with raw data was not successful compared to patternet or any LSTM method with extracted features falling really low in results. Patternnet system performed well with GTCC extracted features compared to MFCC features but was not able to work with large number of training files causing out of memory error in MATLAB making it inefficient with big data applications. Unidirectional LSTM model and bidirectional LSTM model with small number of files during training showed no difference between the MFCC

or GTCC features in terms of AUC score but slightly improved accuracy for GTCC features. Both LSTM models showed small improvement with large number of training files compared to low number of files yet MFCC and GTCC features showed no difference on any files sizes for results. LSTM models outperformed all other models and proved best for audio applications. Even though GTCC features performed well with patternnet and showed no difference compared to MFCC features in LSTM models lack of big file applications made it inefficient. With patternnet application features are used in vector formation loosing time dependency learnings, loosing the weights transitioning between the time steps. But LSTM network input features are in matrix formation preserving time steps where each column is used as vector input for each time step, improving result scores.

Increasing mini batch size in CNN model decreased our results a lot while increasing speed slightly. With LSTM models since number of input sequences increase enormously with feature extraction methods, mini batch size increment changed no accuracy or AUC but increased our training speed greatly. True generalization is still difficult at the moment and we were not able to combine different datasets for testing after training with a specific database. At our current level same dataset was required for both the training and testing applications.

Our results in CNN with MFCC features with 45×20 sequenced inputs as mel images showed that removing noise and enviromental sounds from mel images greatly improves accuracy and AUC scores for training with 5000 audio files but with 15400 files we received out of memory error for too many input images and can not make any assumptions for large number of files.

For any future use, implementation to python environment is heavily encouraged since it is both free and there are more applications and tutorials available in open source codes. Working with more complex systems to classify exact bird species as classes rather then general bird detection might be a better application for environmental works.

Implementation to microprocessor based systems can be handy for mobile applications but because of workload during training, a fusion centre or any type of wireless server will be

necessary until more capable devices are created. There are RNN or LSTM implementations being performed with FPGA processors on standalone devices specifically for speech detection [82–86]. In order to improve the design, several power efficient implementations and memory acceleration schemes are also being presented [87–91]. Such RNN or LSTM implementations can be extended to bird call or bird song classification.

REFERENCES

1. Ferreira N, Lins L, Fink D, Kelling S, Wood C, Freire J, Silva C. Birdvis: Visualizing and understanding bird populations. *IEEE Transactions on Visualization and Computer Graphics.* 2011;17(12):2374-83.

2. Stowell D, Wood M, Pamuła H, Stylianou Y, Glotin H. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *Methods in Ecology and Evolution.* 2019;10(3):368-80.

3. Davis S, Mermelstein P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 1980;28(4):357-66.

4. Chu S, Narayanan S, Kuo CC. Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing.* 2009;17(6):1142-58.

5. Ghoraani B, Sridhar K. Time–frequency matrix feature extraction and classification of environmental audio signals. *IEEE Transactions on Audio, Speech, and Language Processing.* 2011;19(7):2197-209.

6. Zhang T, Kuo CC. Hierarchical classification of audio data for archiving and retrieving. *IEEE International Conference on Acoustics, Speech and Signal Processing.* 1999: IEEE.

7. Carey MJ, Paris ES CC, Lloyd-Thomas H. A comparison of features for speech, music discrimination. *IEEE International Conference on Acoustics, Speech and Signal Processing.* 1999: IEEE.

8. Astuti W, Aibinu AM,Salami MJE, Akmelawati R, Muthalif AGA. Animal sound activity detection using multi-class support vector machines. *IEEE 4th International*

Conference on Mechatronics (ICOM). 2011: IEEE.

9. Noda JJ,Travieso CM, Sánchez-Rodríguez D, Dutta MK, Singh A. Using bioacoustic signals and support vector machine for automatic classification of insects. *IEEE 3rd International Conference on Signal Processing and Integrated Networks (SPIN).* 2016: IEEE.

10. Ko K, Park S, Ko H. Convolutional feature vectors and support vector machine for animal sound classification. *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).* 2018: IEEE.

11. Al Bashit A, Valles D. MFCC-based Houston toad call detection using LSTM. *IEEE International Symposium on Measurement and Control in Robotics (ISMCR).* 2019: IEEE.

12. Tsai WH, Xu YY, Lin WC. Bird species identification based on timbre and pitch features. *IEEE International Conference on Multimedia and Expo (ICME).* 2013: IEEE.

13. Lee CH, Han CC, Chuang CC. Automatic classification of bird Species from their sounds using two-dimensional cepstral coefficients. *IEEE Transactions on Audio, Speech, and Language Processing.* 2008;16(8):1541-50.

14. Kwan C, Mei G, Zhao X, Ren Z, Xu R, Stanford V, Rochet C, Aube J, Ho KC. Bird classification algorithms: Theory and experimental results. *IEEE International Conference on Acoustics, Speech, and Signal Processing.* 2004: IEEE.

15. Somervuo P, Harma A, Fagerlund S. Parametric representations of bird sounds for automatic species recognition. *IEEE Transactions on Audio, Speech and Language Processing.* 2006;14(6):2252-63.

16. Rai P, Golchha V, Srivastava A, Vyas, G, Mishra, S. An automatic classification of bird species using audio feature extraction and support vector machines. *International*

Conference on Inventive Computation Technologies (ICICT). 2016: IEEE.

17. Bai J, Wu R, Wang M, Li D, Li D, Han X, Wang Q, Liu Q, Wang B, Fu Z. CIAIC-BAD system for DCASE2018 challenge task3. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

18. Berger F, Freillinger W, Primus P, Reisinger W. Bird audio detection - DCASE 2018. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

19. Yu C, Hao Y, Yang W, Fu B. Author guidelines for DCASE 2018 challenge technical report. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

20. Lasseck M. Acoustic bird detection with deep convolutional neural networks. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

21. Himawan I, Towsey M, Roe P. 3D convolution recurrent neural networks for bird sound detection. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

22. Jamali S, Ahmadpanah J, Alipoor G. Bird audio detection using supervised weighted NMF. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

23. Song J, Li S. Bird audio detection using convolutional neural networks and binary neural networks. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

24. Kong Q, Iqbal T, Xu Y, Wang W, Plumbley MD. DCASE 2018 challenge surrey cross-task convolutional neural network baseline. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*. 2018: DCASE.

25. Mukherjee R, Banerjee D, Dey K, Ganguly N. Convolutional recurrent neural network based bird audio detection. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop.* 2018: DCASE.

26. Liaqat S, Bozorg N, Jose N, Conrey P, Tamasi A, Johnson MT. Domain tuning methods for bird audio detection. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop.* 2018: DCASE.

27. Tao L, Chen X. Bird audio detection for DCASE 2018 challenge technical report. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop.* 2018: DCASE.

28. Thakur A, Pankajakshan A, Rajan P. Learned aggregation in CNN: all-conv net for bird activity detection. *Detection and Classification of Acoustic Scenes and Events 2018 Workshop.* 2018: DCASE.

29. Ardakani IS, Hashimoto K. Encoding bird's trajectory using recurrent neural networks. *IEEE International Conference on Mechatronics and Automation (ICMA).* 2017: IEEE.

30. Müller L, Marti M. Bird sound classification using a bidirectional LSTM. 2018: CLEF.

31. Trinh TT, Yoshihashi R, Kawakami R, Iida M, Naemura T. Bird detection near wind turbines from high-resolution video using lstm networks. *In World Wind Energy Conference (WWEC).* 2016;2(5):6.

32. Zölzer U. *DAFX: Digital audio effects, 2nd edition.* West Sussex: John Wiley and Sons; 2002.

33. Weisstein EW. MathWorld - A Wolfram web resource 2019 [cited 2019 28 November]. Available from: https://mathworld.wolfram.com/HammingFunction.html.

34. Rabiner LR, Schafer RW. *Theory and applications of digital speech processing.* Upper Saddle River, NJ: Pearson; 2011.

35. Pertila P. Mel frequency cepstral coefficients (MFCCs) and gammatone filter banks 2019 [cited 2020 05 March]. Available from: http://www.cs.tut.fi/~sgn14006/PDF2015/S04-MFCC.pdf.

36. Lyons J. Mel frequency cepstral coefficient (MFCC) tutorial 2012 [cited 2020 05 March]. Available from: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs.

37. Valero X, Alias F. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *IEEE Transactions on Multimedia.* 2012;14(6):1684-9.

38. Shao Y, Jin Z, Wang D, Srinivasan S. An auditory-based feature for robust speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing.* 2009;4625-8.

39. Vu-Quoc L. Neuron3.png 2018 [cited 2019 28 November]. Available from: https://commons.wikimedia.org/wiki/File:Neuron3.png.

40. Goodfellow I, Bengio Y, Courville A. *Deep learning.* Cambridge, MA : MIT press; 2017.

41. The MathWorks, Inc. Specify layers of convolutional neural network 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html.

42. The MathWorks, Inc. List of deep learning layers 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ug/list-of-deep-learning-layers.html.

43. The MathWorks, Inc. image input layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html.

44. The MathWorks, Inc. convolution 2d layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html.

45. The MathWorks, Inc. Max pooling 2d layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling2dlayer.html.

46. The MathWorks, Inc. Average pooling 2d layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.averagepooling2dlayer.html.

47. The MathWorks, Inc. Batch normalization layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html.

48. The MathWorks, Inc. Rectified linear Unit layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.relulayer.html.

49. Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. *27th International Conference on Machine Learning*. 2010:ICML.

50. The MathWorks, Inc. Leaky rectified linear unit layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.leakyrelulayer.html.

51. The MathWorks, Inc. Clipped rectified linear unit layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/

nnet.cnn.layer.clippedrelulayer.html.

52. The MathWorks, Inc. Exponential linear unit layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.elulayer.html.

53. The MathWorks, Inc. Hyperbolic tangent layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.tanhlayer.html.

54. The MathWorks, Inc. Fully connected layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html.

55. The MathWorks, Inc. Softmax layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html.

56. The MathWorks, Inc. Classification output layer 2019 [cited 2019 28 November]. Available from: https//www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.classificationoutputlayer.html.

57. The MathWorks, Inc. Regression output layer 2019 [cited 2019 28 November]. Available from: https://www.mathworks.com/help/deeplearning/ref/regressionlayer.html.

58. Visin F, Kastner K, Cho K, Matteucci M, Courville A, Bengio Y. Renet: A recurrent neural network based alternative to convolutional networks. 2015;arXiv:1505.00393.

59. Kalchbrenner N, Danihelka I, Graves A. Grid long short-term memory. 2015;arXiv:1507.01526.

60. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation.*

1997;9(8):1735-80.

61. The MathWorks, Inc. Long short-term memory networks 2019 [cited 2020 26 March]. Available from: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html.

62. The MathWorks, Inc. Sequence input layer 2019 [cited 2020 26 March]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.sequenceinputlayer.html.

63. The MathWorks, Inc. Long short-term memory (LSTM) layer 2019 [cited 2020 26 March]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmlayer.html.

64. The MathWorks, Inc. Bidirectional long short-term memory (BiLSTM) layer 2019 [cited 2020 26 March]. Available from: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.bilstmlayer.html.

65. Google Developers. Classification: ROC curve and AUC 2020 [cited 2020 26 March]. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

66. Google Developers. Classification: True vs. false and positive vs. negative 2020 [cited 2020 26 March]. Available from: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative.

67. Stowell D, Stylianou Y, Wood M, Pamula H, Glotin H. Bird audio detection task description 2018 [cited 2019 28 November]. Available from: http://dcase.community/challenge2018/task-bird-audio-detection.

68. The MathWorks, Inc. Mel spectrogram 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/audio/ref/melspectrogram.html.

69. Cornell University. All About birds 2009 [cited 2020 08 March]. Available from: https://www.allaboutbirds.org/news/do-bird-songs-have-frequencies-higher-than-humans-can-hear/.

70. The MathWorks, Inc. Streamline audio feature extraction 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/audio/ref/audiofeatureextractor.html.

71. The MathWorks, Inc. Classify patterns with a shallow neural network 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/deeplearning/gs/classify-patterns-with-a-neural-network.html.

72. The MathWorks, Inc. network diagram 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/examples/nnet/win64/GsNprtoolExample-01.png.

73. The MathWorks, Inc. Create simple sequence classification network 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/deeplearning/gs/create-simple-sequence-classification-network.html.

74. Dua D, Graff C. UCI Machine learning repository 2019 [cited 2020 08 March]. Available from: http://archive.ics.uci.edu/ml.

75. The MathWorks, Inc. Classify gender using LSTM networks 2019 [cited 2020 08 March]. Available from: https://www.mathworks.com/help/deeplearning/gs/create-simple-sequence-classification-network.html.

76. The Mozilla Foundation. Common voice dataset 2019 [cited 2020 08 March]. Available from: https://voice.mozilla.org/en/datasets.

77. Amidi S. VIP Cheatsheet: Recurrent neural networks 2019 [cited 2020 29 April]. Available from: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

78. Jiang T, Xiao Q, Yin X. Music Generation Using Bidirectional Recurrent Network. *IEEE 2nd International Conference on Electronics Technology (ICET).* 2019: IEEE.

79. Yim J, Sohn KA. Enhancing the performance of convolutional neural networks on quality degraded datasets. *International Conference on Digital Image Computing: Techniques and Applications (DICTA).* 2017: IEEE.

80. Baykal S I, Bulut D, Sahingoz O K. Comparing deep learning performance on BigData by using CPUs and GPUs. *Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT).* 2018: IEEE.

81. Purwins H, Li B, Virtanen T, Schlüter J, Chang S Y, Sainath T. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing.* 2019;13(2):206-19.

82. Lee M, Hwang K, Park J, Choi S, Shin S, Sung W. FPGA-based low-power speech recognition with recurrent neural networks. *IEEE International Workshop on Signal Processing Systems (SiPS).* 2016: IEEE.

83. Zainab M, Usmani AR, Mehrban S, Hussain M. FPGA based implementations of RNN and CNN: A Brief Analysis. *International Conference on Innovative Computing (ICIC).* 2019: IEEE.

84. Hoffmann J, Guzmán ON, Kastner F, Janßen B, Hübner M. A survey on CNN and RNN implementations. *The Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications.* 2017: PESARO.

85. Han S, Kang J, Mao H, Hu Y, Li X, Li Y, Yang, H. Ese: Efficient speech recognition engine with sparse LSTM on FPGA. *International Symposium on Field-Programmable Gate Arrays.* 2017: ACM/SIGDA.

86. Rybalkin V, Pappalardo A, Ghaffar MM, Gambardella G, Wehn N, Blott M. FINN-L: Library extensions and design trade-off analysis for variable precision LSTM

networks on FPGAs.*28th International Conference on Field Programmable Logic and Applications (FPL.)* 2018: IEEE.

87. Gao C, Neil D, Ceolini E, Liu SC, Delbruck T. DeltaRNN: A power-efficient recurrent neural network accelerator. *International Symposium on Field-Programmable Gate Arrays.* 2018: ACM/SIGDA.

88. Wang Z, Lin J, Wang Z. Accelerating recurrent neural networks: A memory-efficient approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.* 2017;25(10):2763-75.

89. Gao C, Zhang F. FPGA-based accelerator for independently recurrent neural network. *IEEE 4th International Conference on Computer and Communications (ICCC).* 2018: IEEE.

90. Azari E, Vrudhula S. An energy-efficient reconfigurable LSTM accelerator for natural language processing. *IEEE International Conference on Big Data (Big Data).* 2018: IEEE.

91. Wang S, Lin P, Hu R, Wang H, He J, Huang Q, Chang S. Acceleration of LSTM with structured pruning method on FPGA. *IEEE Access.* 2019;7:62930-37