

# APM 598: Deep Neural Networks

## MUSIC GENRE CLASSIFICATION USING NEURAL NETWORKS

### FINAL PROJECT REPORT

Vivek Verma  
Harishankar Prabhakaran  
Senthilkumar Murali  
Jennifer Rastegar

May 6, 2020

### Abstract

Audio data is becoming incredibly important, from having audio assistants like Siri and Alexa to self-driving cars that can interpret a broad range of stimuli that aren't just 'seeing' but even 'hearing'. We consume more audio information than anyone who lived before us. Audio files are discrete-time signals whose sample rate keeps going up as the quality of today's recordings increases, hence producing a densely packed amplitude values separated in time. Artificial Neural networks have the power to unpack the data and classify it. Along with the help from signal feature analysis, this project handles neural network techniques such as One Dimensional Convolutional Neural Network using direct amplitude data, and Two-Dimensional Convolutional Neural Network using the spectrogram images produced from the signals. This project is focused on classifying music by genres. A humongous amount of music files can be easily accessed through the internet and musical apps such as Spotify. With the increasing amount of its presence digitally, there is a growing demand for an excellent system to organize audio files. Moreover, detecting and grouping music of a similar genre is a keen part in music recommendation system and playlist generator. Using different implementations, classification of **81.62%** for ten musical genres is achieved.

**Index Terms** - Audio classification, feature extraction, musical genre classification, Convolutional Neural Networks (CNN).

## Introduction

A deep neural network is a great tool to understand data from complex systems. Classification of data sensed through vision is widespread and easily understandable, but audio signals are everywhere and analyzing them gives a better understanding of the systems. This need to classify the signals is the inspiration to this project, which is concentrated on classifying musical dataset. Even though the classification of music by genres is not mimicking a specific system, it lays the foundation for audio classification in general. The project is all about designing different neural networks to imitate the functions of a human ear, trained only to classify musical genres.

Musical GTZAN Dataset has been classified into ten categories: Blues, Classical, Country, Disco, Hip-hop, Jazz, Pop, Metal, Reggae, Rock. Even though the task of classification seems easy for a human just by hearing, people who have had prior experience or interest in music can do so. On the contrary, Neural networks have to be taught how to recognize different features of an audio signal, to find patterns, to group them accordingly and eventually to classify any new data given to it.

Audio classification is not a new topic, and research on this topic has been going on for almost two decades. This project is another small increment in that progress; instead of adding more layers to improve performance, the focus is mainly on implementing different architectures of convolutional neural network with least number of parameters and less computation power but giving the same performance.

Classification using the features obtained from predefined functions, 1-D-CNN, and 2-D-CNN are implemented. Various configurations are analyzed; the results are compared and presented.

## Data Description

GTZAN Dataset (Tzanetakis & Cook, 2002) is composed of 1000 half a minute audio excerpts. It contains ten genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in .wav format. It is the most used dataset for Music Genre Recognition by machine learning algorithms. The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, to represent a variety of recording conditions.

It is available to download from <http://marsyas.info/downloads/datasets.html>

## MODE 1: Feature Extraction & Artificial Neural Network

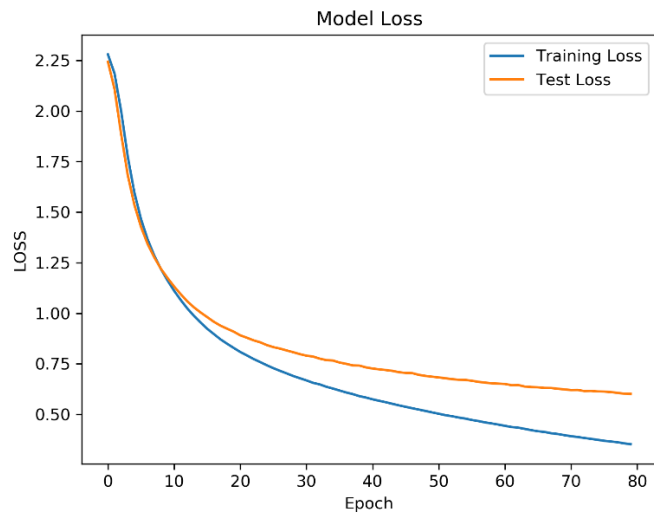
Feature extraction is the way of finding numerical representation that can be used to describe an audio signal. The primary intention of this process is to recognize patterns from a vast amount of data. Once the features are pulled out, Neural Networks can be used directly using the numerical data.

LibROSA is a python package for music and audio analysis. It offers many functions to extract features directly from audio data. Some of the features used in this project are listed below.

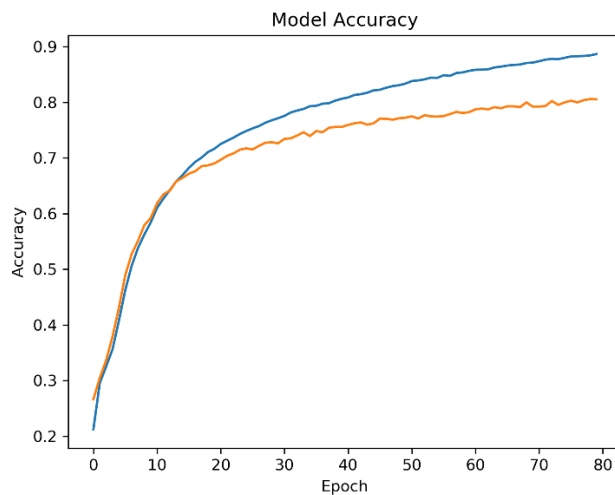
Features functions	Definition
chroma_stft	Compute a chromagram from a waveform or power spectrogram.
rms	Compute root-mean-square (RMS) value for each frame
spectral_centroid	Compute the spectral centroid $C_t = \frac{\sum_{n=1}^N M_t[n] * n}{\sum_{n=1}^N M_t[n]} \quad (1)$ Where, $M_t[n]$ is the magnitude of the Fourier transform at frame $t$ and frequency bin.
spectral_bandwidth	Compute p'th-order spectral bandwidth
spectral_rolloff	Compute roll-off frequency $\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^N M_t[n]. \quad (2)$
zero_crossing_rate	Compute the zero-crossing rate of an audio time series $Z_t = \frac{1}{2} \sum_{n=1}^N  sign(x[n]) - sign(x[n-1])  \quad (3)$ where $x[n]$ is the time domain signal for frame $t$ .
tempo	Estimate the tempo (beats per minute)
mfcc	Mel-frequency cepstral coefficients (MFCCs)

Features data for cut 3-second audio files are available online for GTZAN dataset. This is in CSV format and holds 57 features (mean and variance values). These features can also be extracted using the LibROSA package. Both mean and variance of these numerical features are used as the input to a simple neural network with just linear transformations with ReLu activation function. Since the number of inputs is less, it takes a very short time to train the system. An artificial neural network with three hidden layers classified the test data with 80.56% accuracy.

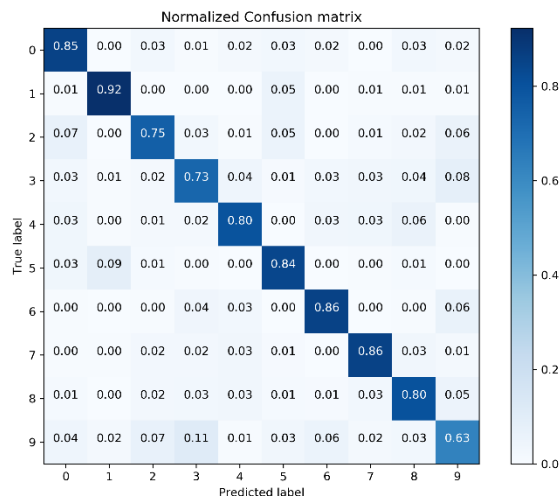
## Graphs (Feature Extraction & Artificial Neural Network):



**Figure 1:** Training and Test Loss for the model obtained by feature Extraction.



**Figure 2:** Training and Test Accuracy for the model obtained by feature extraction.




**Figure 3:** Confusion Matrix for the model obtained by feature extraction.

## **MODE 2: 1-D Amplitude values of Audio & Convolutional Neural Network**

Another way to input data from the audio signal is to get the amplitude values directly for each time step and feed it into a one-dimensional CNN. The main hindrance is that most audio files (including the GTZAN dataset) have a high sample rate. There are two ways we can get around this problem. One is to reduce the sample rate, and the other is to cut the length of each audio sample in the dataset. Reducing the sampling rate leads to the loss of many important features of music. Therefore, the audio samples are cut into 2-second pieces with 1-second overlap. Doing this also increases the size of the data set. Cut audio length, and the overlap percentage plays a significant role in the quality of the classification process; Features expressed over time longer than the cut audio length, will be lost if the overlap is not included.

Many research papers were studied to get a good 1-D CNN architecture and a paper on environmental sound classification (Abdoli, Cardinal, & Koerich, 2019), which discusses different architectures considering several input sizes. They proposed CNN architectures which only needs a small number of parameters comparatively, and thus the volume of data essential for training is also reduced.

### **1-D CNN Architecture used in this project**



<b><i>Layers (in order)</i></b>	<b><i>Strides</i></b>	<b><i># of Filters</i></b>	<b><i>Filter Size</i></b>
<b>convolutional layer 1</b>	2	16	64
<b>pooling layer 1</b>	8	16	8
<b>convolutional layer 2</b>	2	32	32
<b>pooling layer 2</b>	8	32	8
<b>convolutional layer 3</b>	2	64	16
<b>convolutional layer 4</b>	2	128	8
<b>convolutional layer 5</b>	2	256	4
<b>pooling layer 3</b>	4	256	4
<b>Two fully connected layers with 128 and 64 neurons (dropout = 0.25)</b>			
<b>Output layer where a softmax activation function is used</b>			

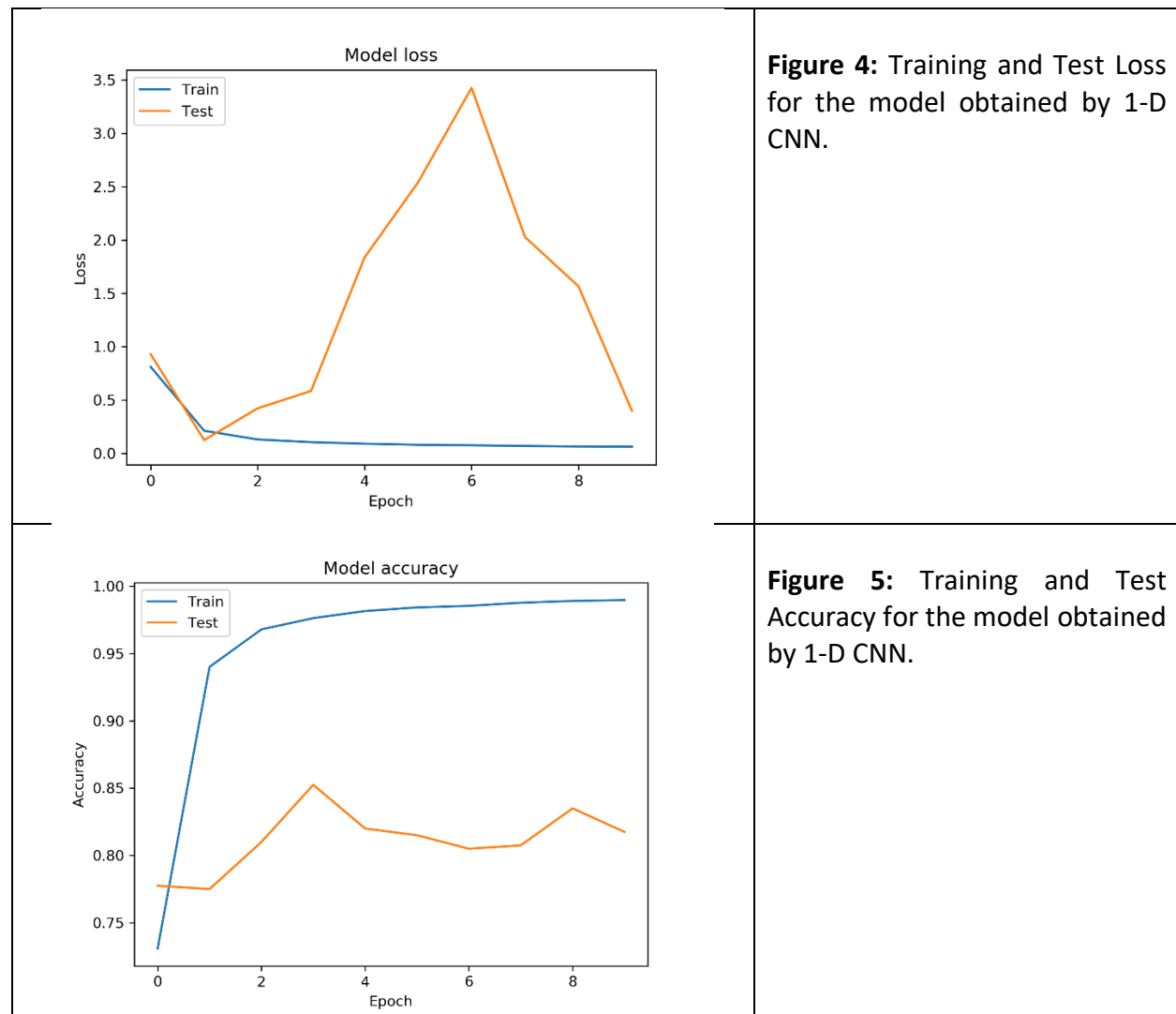
The ReLU activation function  $h(x) = \max(x, 0)$  is used for all layers. To decrease over-fitting, batch normalization is used after the activation function of each convolution layer.

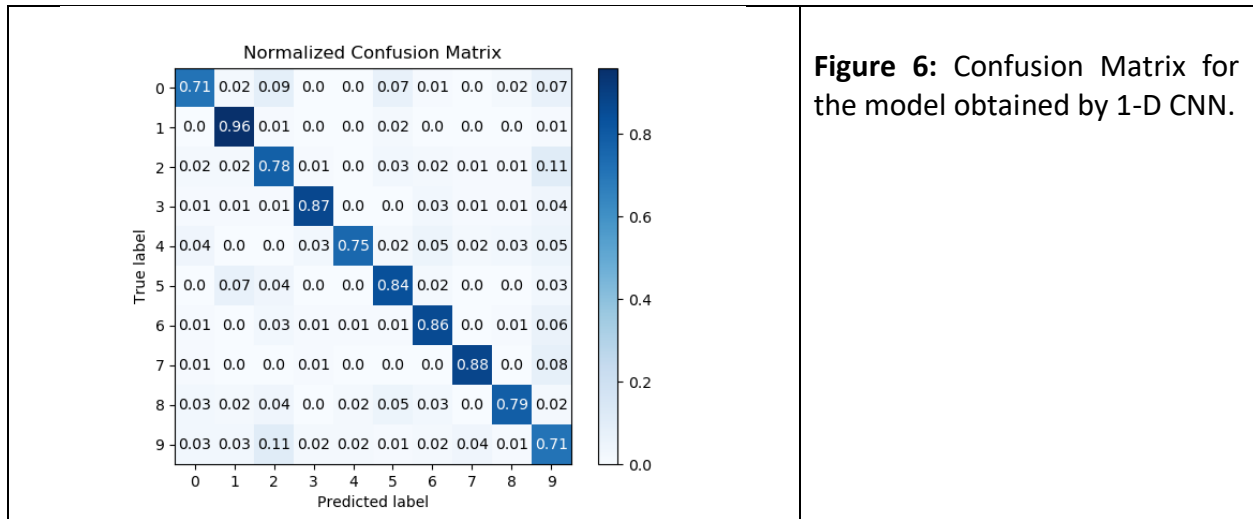
The number of parameters gets reduced by using the network given above and by increasing strides.

## Factors impacting 1-D Classification

- **Kernels** can be viewed as feature detectors in convolutional layers, and its size determines the range of a feature map it can detect. Increasing the kernel size also increases the parameters, and in turn, requires more computation. Therefore, a balance between feature detection efficiency and computation should be maintained.
- **Pooling**, in a very broad sense, is reducing the sample rate by fusing the amplitude values. This helps in finding the basic music elements such as overtone, timbre, pitch and amplitude.
- Linear operations are not adequate to represent all the organic audio features. **ReLU**s brings out sparse feature representations in the hidden layers by introducing non-linear behavior.

## Graphs (1-D Amplitude values of Audio & Convolutional Neural Network):





### **MODE 3: 2-D Mel-Spectrogram & Convolutional Neural Network**

**Fast Fourier transform** decomposes a signal into its distinct frequencies present in the signal and the frequency's amplitude, called as the spectrum. It extracts most of the information of a signal if it is periodic and it does not change over time. All music signals vary over time (nonperiodic); therefore, the spectrum is calculated on many-windowed segments of the signal, called a short-time Fourier transform. A spectrogram is a bunch of the Fourier transform put together, to visually represent a signal's varying amplitude over time at different frequencies. **Mel scale** is a unit of frequency such that equal distances in frequencies sounded equally distant to the listener. Spectrogram, whose frequency axis is in Mel Scale is Mel-Spectrogram.

Highly efficient image classification neural networks exist, and spectrogram image has all the information of an audio file. Therefore, classifying the spectrograms is equivalent to classifying the actual audio files. First 20 seconds of each audio file is divided into ten 2 second clips and converted into a spectrogram. The images stored separately are used by the image classification network.

A residual neural network or **ResNet** builds on constructs known from pyramidal cells in the cerebral cortex. They achieve this by using skip connections to jump over some layers.

In 2D Convolution, a kernel slides over the 2D input data, performing an element-wise multiplication with the part of the input. The output features are the weighted sums, which defines various features from the spectrogram.

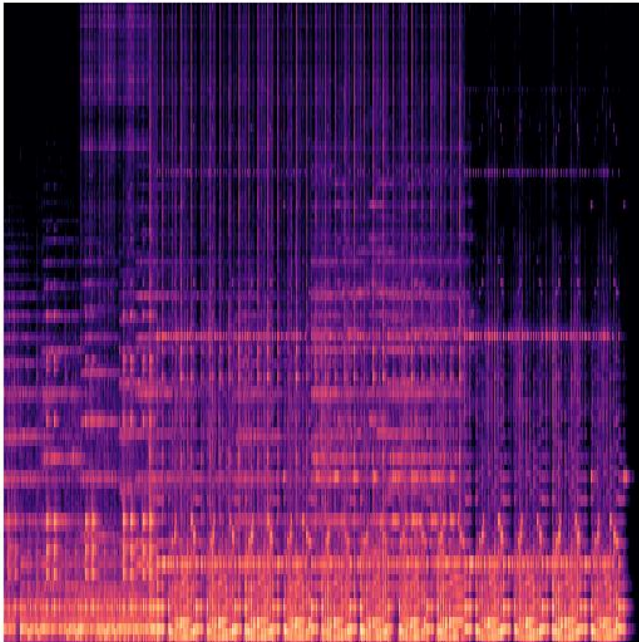
## 2-D CNN ResNet Architecture used

```

Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
AdaptiveAvgPool2d(output_size=(1, 1))
Linear(in_features=512, out_features=10, bias=True)

```

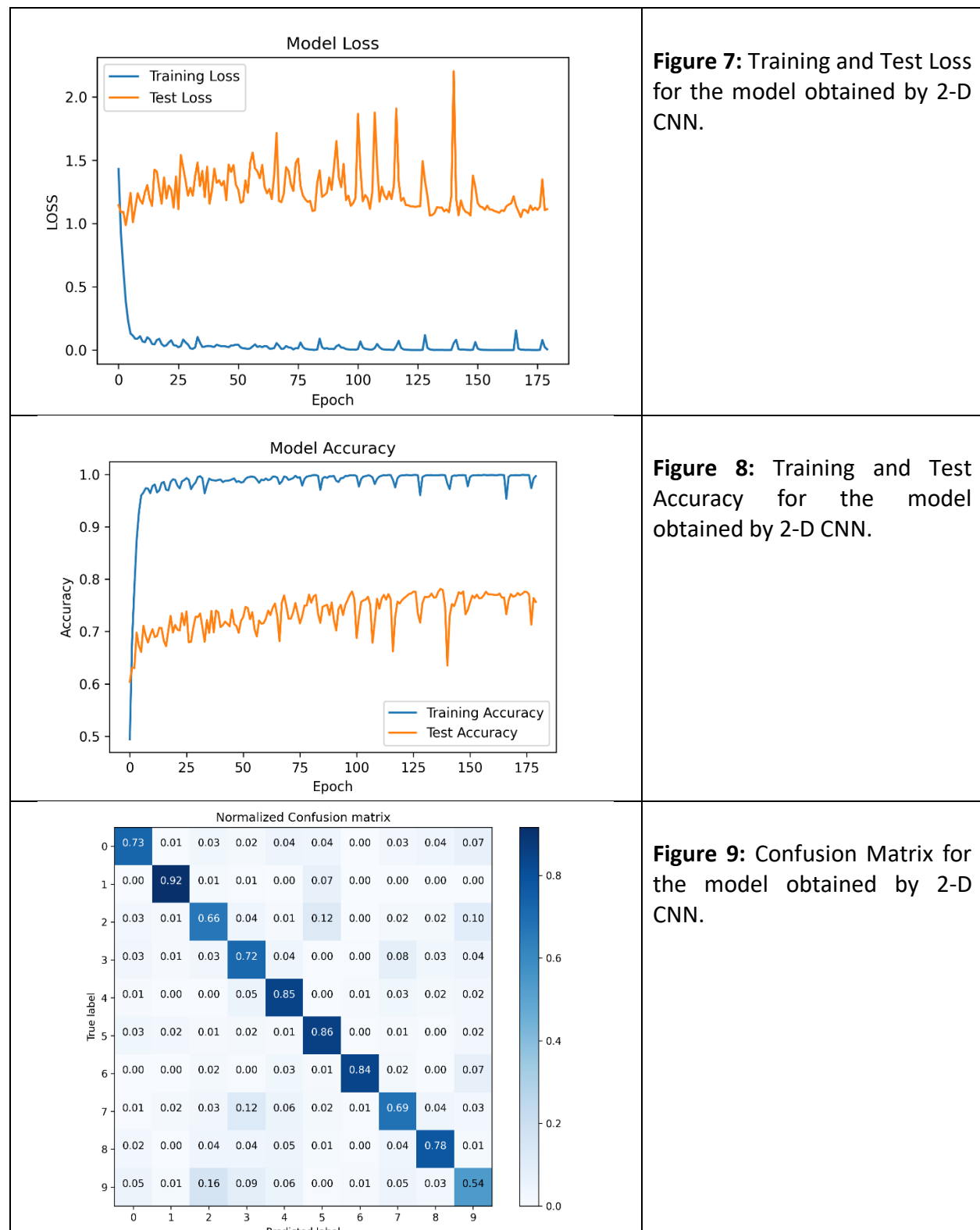
**Figure 10:** ResNet architecture, similar to (He, Zhang, Ren, & Sun, 2015); The ReLU activation function  $h(x) = \max(x, 0)$  is used for all layers. To decrease over-fitting, batch normalization is used after the activation function of each convolution layer.



**Figure 11:** Spectrogram Image obtained from sample audio.



## Graphs (2-D Mel-Spectrogram & Convolutional Neural Network):



## **Conclusion**

Classification using the features obtained from predefined functions, 1-D-CNN, and 2-D-CNN are implemented. The maximum accuracy values obtained from three different modes of implementation are given in the table below.

Mode	Percentage Accuracy
Feature Extraction	80.56%
1-D Convolutional NN	81.62%
2-D Convolutional NN	75.65%

The lowest number of trainable parameters achieved for the model produced by 1-D Convolutional NN is 388,378. The 1-D CNN also produced the best model with an accuracy of 81.67%, followed by Feature Extracted NN with 80.56%. And 2-D CNN (ResNet-18) with 75.65%.

2-D CNN was expected to produce a highly accurate model, but the networks used did not give good accuracy.

With respect to the computation time for training, Feature Extracted NN was the fastest (within a few minutes). It is because the input size is only the numerical values of the features.

## **Future Work:**

In 1-D CNN, gammatone filter banks can be used in place of the first convolution layer. Gammatone filter banks mimic auditory filters in the auditory system. Implementing this will extract more defined features, making the network more efficient.

We can use other datasets like UrbanSound8k, Million Song Dataset because the GTZAN is a really old dataset.

We can explore the 2D convolution method more and rectify how to fine-tune parameters or include another better CNN network to get more accuracy.

## References

- Tzanetakis, G., & Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*.
- Abdoli, S., Cardinal, P., & Koerich, A. (2019). End-to-End Environmental Sound Classification using a 1DConvolutional Neural Network. *Department of Software and IT Engineering, École de Technologie Supérieure, Université du Québec*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Microsoft Research*.