APS360
Project Final Report

Project Title: Instrument Classifier

Github link
https://github.com/APS360-instrument-classifier/project

Team 38

Krutarth Patel
Malhar Shah
Mohammad Ahmed
Shanthosh Sivayogalingam

Word Count: 2403

1.0 Introduction

In today's world, music is a big component in our daily lives. Regardless of the genre, language, or the medium of the music, it has a very drastic and profound impact on our lives. Music can be very complex in that it can contain several instruments played together at a fast pace. However, many listeners may not be able to accurately decipher which ones are actually being used.

Our goal with this project is to be able to create a Machine Learning model that can take in an instrumental audio file and output the list of instruments being used as accurately as possible. We believe this will bridge the gap where people can now visibly see and classify what instruments are involved in their favourite instrumentals. Taking advantage of supervised learning, we want to be able to create a Convolutional Neural Network that can generalize the different sounds created by various instruments and apply it to any instrumental audio the person wants. With our model, we will precisely identify and classify this information and present it to the user.

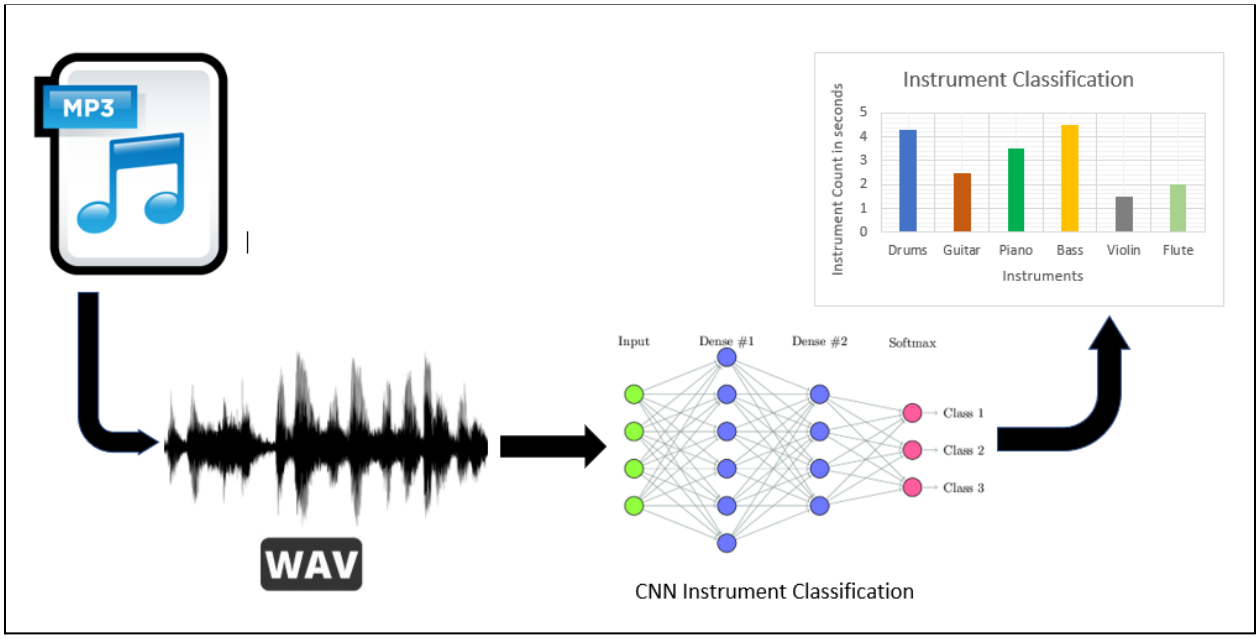

Figure 1.0 - Process from audio file to instrument classification

2.0 Background & Related Work

A few researchers from MIT released PixelPlayer which is an AI system that has similar functionalities to our project. [1] The system takes in a video rather than an audio file identifies the instrument sounds within the input. This software is composed of three neural networks which are used to analyze the video, audio and separating the sounds. The networks detect the pixels from which the sounds are coming from and use the different components in the analysis. These networks were trained over 60 hours worth of videos and can identify 20 different instruments.

Another similar product is NSynth Super which is an instrument made by Google [2]. Thisproduct replicates different sounds instruments can make using an NSynth algorithm and uses a deep neural network to generate sound. Furthermore, it learns the core qualities of individual sounds and then has the capability to generate completely new sounds by combining various sounds. This is useful for artists and musicians because it allows them to be creative and explore different sounds that they could incorporate into their work.

3.0 Data Processing

Our data consists of audio files with sounds from different instruments. It was gathered from a Kaggle data [3][4] set which consisted of 9473 audio files with samples of different sounds. There were a lot of different sources of sounds in the dataset that weren't instruments. There were audio samples of fireworks, applause, cough and many other sounds (Figure 3.0). Thus, the data was reduced to 3000 samples with each instrument having 300 samples (Figure 3.1).

Acoustic_guitar	300
Applause	300
Bark	239
Bass_drum	300
Burping_or_eructation	210
Bus	109
Cello	300
Chime	115
Clarinet	300
Computer_keyboard	119
Cough	243
Cowbell	191
Double_bass	300
Drawer_open_or_close	158
Electric_piano	150
Fart	300
Finger_snapping	117
Fireworks	300
Flute	300
Glockenspiel	94

Figure 3.0 - Audio files with non-instrumental sounds

label	
Acoustic_guitar	300
Bass_drum	300
Cello	300
Clarinet	300
Double_bass	300
Flute	300
Hi-hat	300
Saxophone	300
Trumpet	300
Violin_or_fiddle	300

Figure 3.1 - New data with only instruments

The audio files in our set were of different durations, and since it's important that the input to a CNN be of the same size, we decided to limit the audio file duration to 2 seconds. Samples with duration of less than 2 seconds were discarded and left us with 2194 audio files. However, the samples per instrument weren't equally distributed after the duration limit which resulted in some instruments having more samples than others. Thus, we decided to duplicate some files for the instruments that had less samples compared to others.

After the duplication, the total sample size increased from 2194 to 2655 samples (Figure 3.2). This was important in order to minimize any input bias. Afterwards, the data was split into training, validation, and testing set with 70/15/15 ratio with the training set having 1858 samples, and the validation and testing set each having 398 samples.

Acoustic_guitar	275	Acoustic_guitar	275
Bass_drum	80	Bass_drum	240
Cello	286	Cello	286
Clarinet	290	Clarinet	290
Double_bass	205	Double_bass	255
Flute	270	Flute	270
Hi-hat	141	Hi-hat	282
Saxophone	250	Saxophone	250
Trumpet	207	Trumpet	257
Violin_or_fiddle	190	Violin_or_fiddle	250

Figure 3.2 - Duplicated samples to reduce input bias

To convert our audio files to images we used the librosa and python_speech_features python packages. Librosa provided the librosa.load function which we used to convert each audio clip into an array of time series values that we plotted to create waveform representations shown in Figure 3.3. Further, we used these arrays and the mfcc function to extract complex features (i.e. pitch, frequency distribution) and created heat maps to represent this data. The heat maps, shown in Figure 3.4, are what we used as inputs for our CNN.

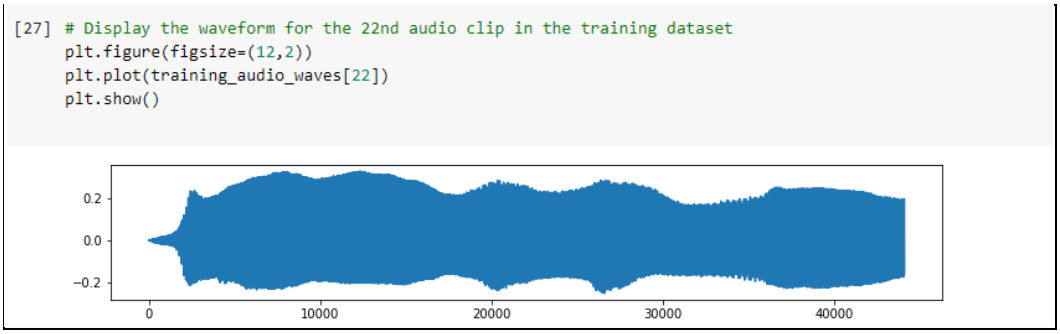

Figure 3.3 - Waveform Representation of an audio .wav file

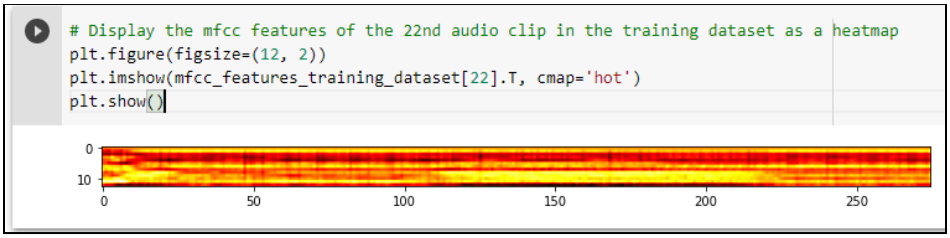

Figure 3.4: Heatmap of an audio .wav file

4.0 Architecture

For our final model, we decided to use a CNN architecture because we wanted to be able to use its ability to identify the subtle features different instruments have in their sound. Our CNN receives MFCC hot encodings in the form of a list. Each list contains information regarding the pitch of the instrument for the duration of the sound.

This leads to a convolutional layer which uses a ReLU activation function followed by a max pooling layer. These two layers are repeated. The kernel size of the convolutional layers is 4x4, while the stride is 2. The kernel size for the 2D max pooling layer is 2x2.

Next, I have a dropout layer which is used to prevent the model from overfitting. After, the network is flattened and it connects to 2 fully connected linear layers which leads to the outer layer using a Softmax classification to predict from the 10 instruments. The 2 fully connected layers have 128 neurons and 64 neurons, respectively.

For the training of the model, I used a Cross Entropy Loss function with Adam Optimizer. The model was trained over 30 epochs. Using these parameters, we were able to achieve the highest validation accuracy of 80%.

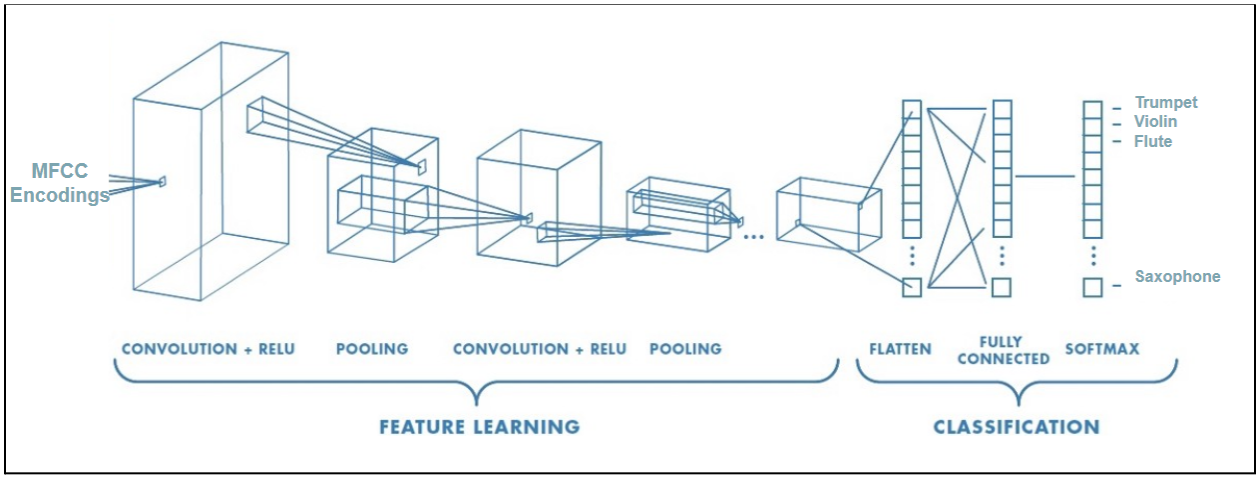

Figure 4.0 - Illustration of our CNN's model architecture

5.0 Baseline Model

For the baseline model, we decided to use a simple Fully Connected ANN architecture. This model also receives MFCC hot encodings in the form of a list. This is followed by 3 hidden layers which leads to the outer layer that uses a Softmax classification to predict which of the 10 instruments was being played in the audio clip. The hidden layers are 128, 64 and 32 neurons respectively.

The model uses ReLU activation function. For the training process, I used a Cross Entropy Loss function with Adam Optimizer. The model was trained over 30 epochs. Using these parameters, we were able to achieve a validation accuracy of 72%.

This architecture was chosen for our model because of its simplicity and ease of training. Furthermore, the model produced fairly good results and it was a good benchmark point to compare our primary model with. The major flaw with this model was that this model was overfitting to the training data.

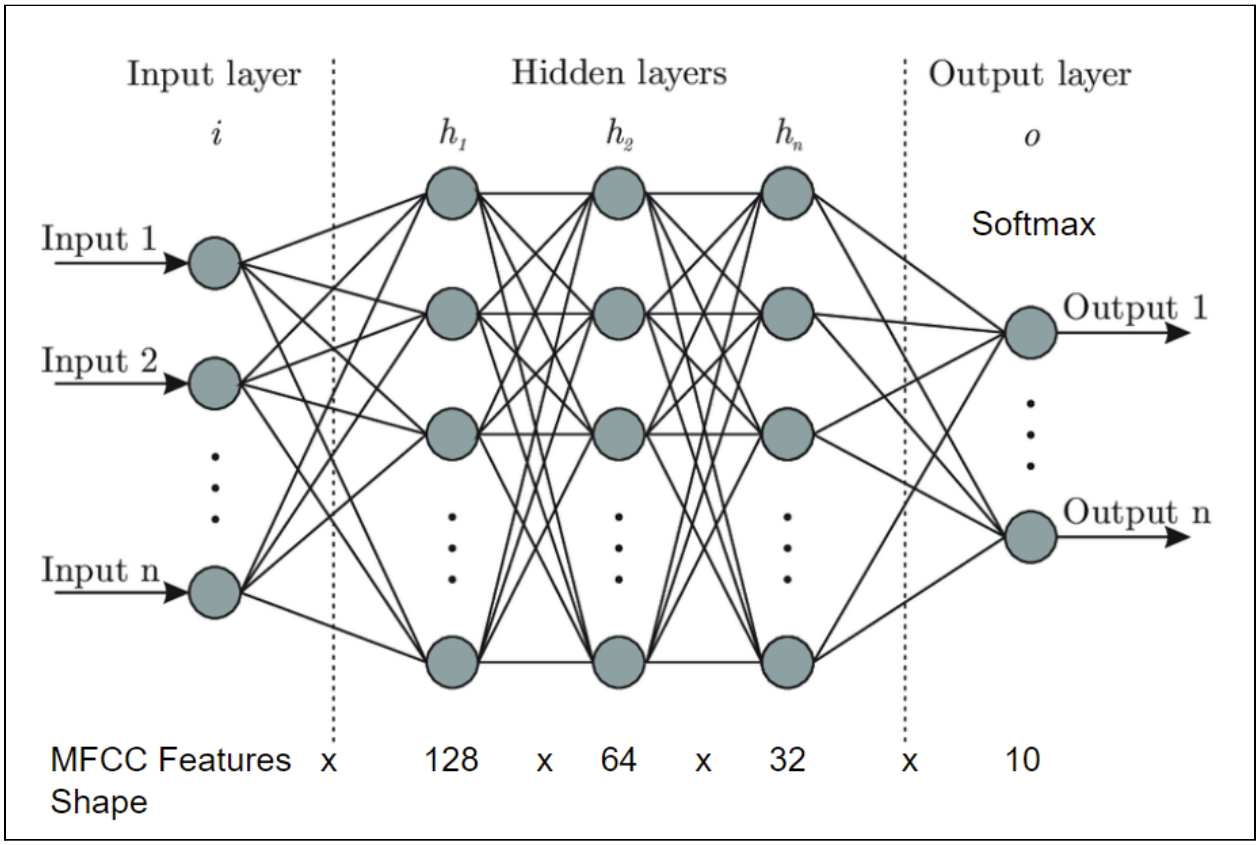

Figure 5.0 - Illustration of our Fully Connected Baseline Model

6.0 Quantitative Results

CNN networks are known for achieving high accuracy in image processing tasks. CNN's automatically detect important features by itself, which in our case would be extracting the distinctive features from each heatmap for each instrument class we used.

Our final CNN model was able to achieve a training accuracy of about 90.01% and a validation accuracy of about 80.06%. The two figures below represent the validation and training accuracies for both our baseline model and our final CNN architecture.

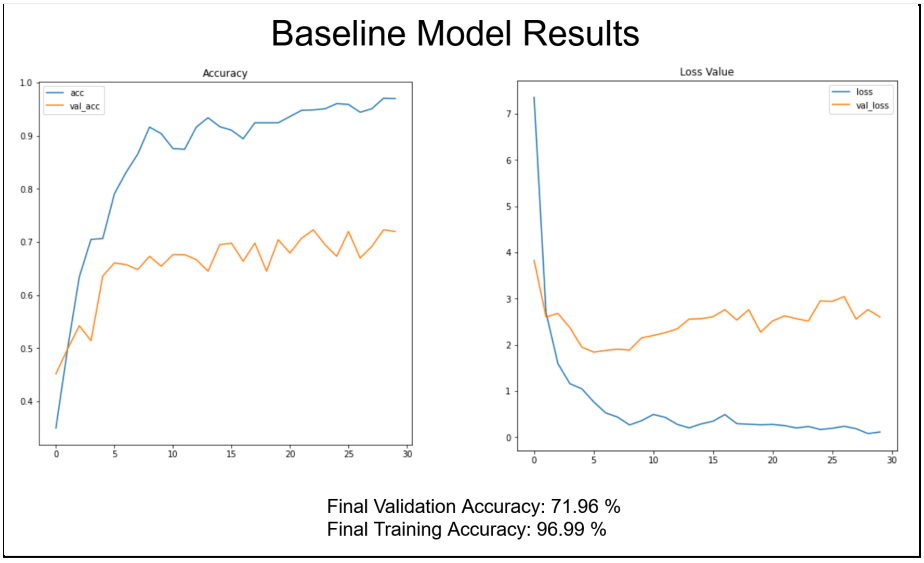

Figure 6.0 - Baseline Model Results

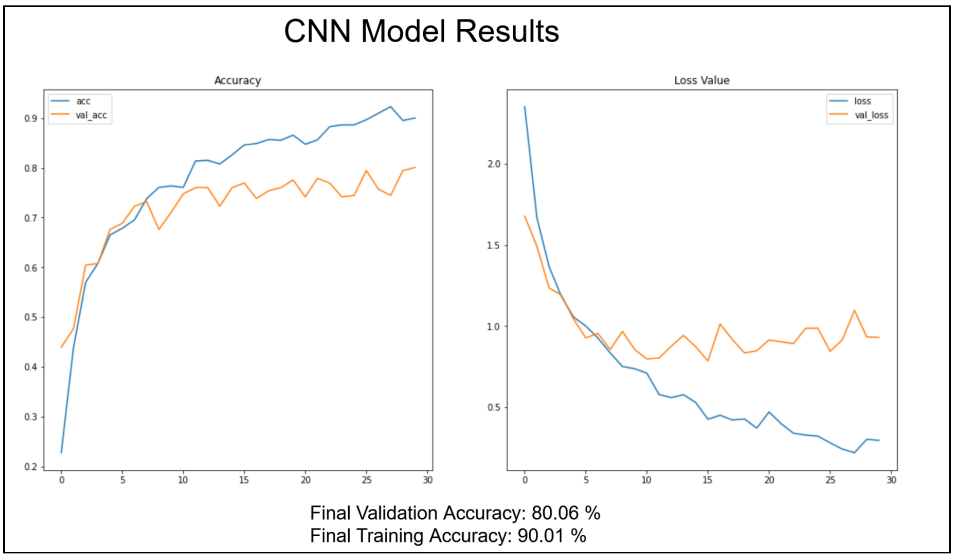

Figure 6.1 - CNN Model Results

Our model and the baseline model both have neural networks and have learnable weights and biases; however, our CNN model is far more efficient when it comes to classifying images. Fully connected networks tend to have a lot more weights along with a large number of parameters which causes training time to slow down along with chances of overfitting.

When looking at our results, this was the case for our baseline model. When comparing the validation loss between both models, our CNN model had a significantly lower loss (0.928 vs 2.6029). We believe the reason behind the difference is because our baseline model seems to be memorizing the correct solutions rather than picking up on trends or patterns which is causing the model to overfit. In addition to the validation loss, the validation accuracy for our baseline model is significantly lower than its training accuracy which backs up the belief of the baseline model overfitting. If you take a look at the figures above, you can note that the validation accuracy and loss graphs are more spread out for the baseline model than our CNN model.

Our CNN model does a better job than our baseline model because it makes an explicit assumption that the inputs are images. This allows our model to be trained so it can extract the best features from our images whereas the fully connected network makes zero assumptions on the inputs making it more general and weaker. The positive effects of this assumption can be seen as the final validation accuracy improved approximately 8% after using a CNN model instead of our baseline model.

7.0 Qualitative Results

Overall, the model performed well. This can be seen through the confusion matrix (Figure 7.0) which shows that the model correctly predicted the instruments majority of the time. The model can get confused when instruments that can sound similar are present. For example, clarinet and saxophone, flute and clarinet, and cello and double bass are instruments that may sound similar which resulted in the model getting confused. This may be resolved by having more samples for similar sounding instruments. Likewise, instruments that have a unique sound and don't sound similar to other instruments are more accurately predicted. For example bass drum and hi-hat have a unique sound compared to the other instruments which resulted in very accurate predictions.

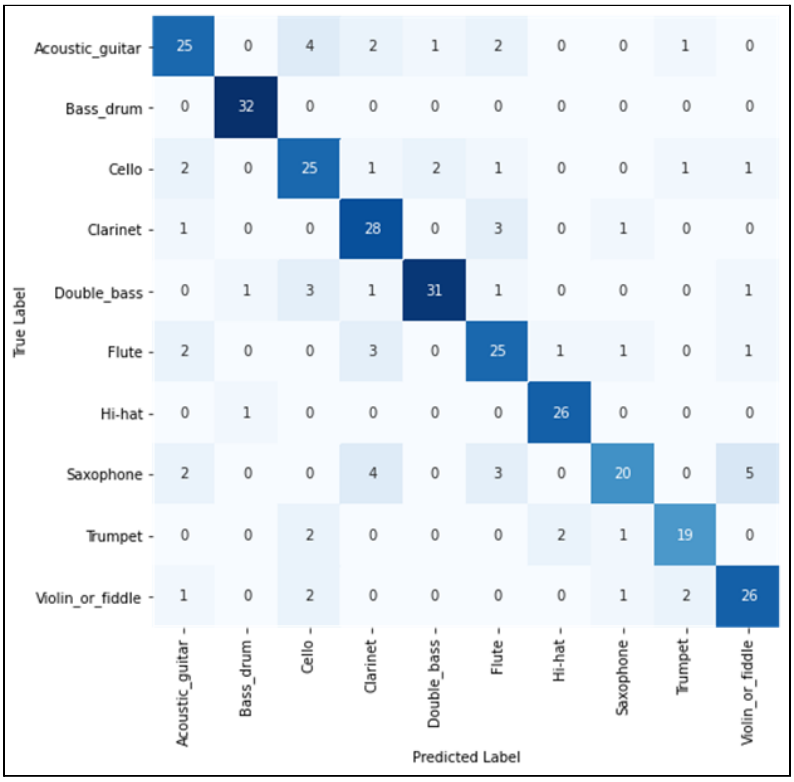

Figure 7.0 - Confusion matrix

8.0 Evaluate Model on New Data

In order to make sure our results from testing the model would be unbiased, the team kept the testing dataset untouched in a separate file throughout the course of training our model. In general, to test new samples of audio clips the new data must be all composed of .wav files of individual instruments, and they must be processed to get their heat map representations. Once new samples have been processed, then they can simply be passed into the trained model as inputs.

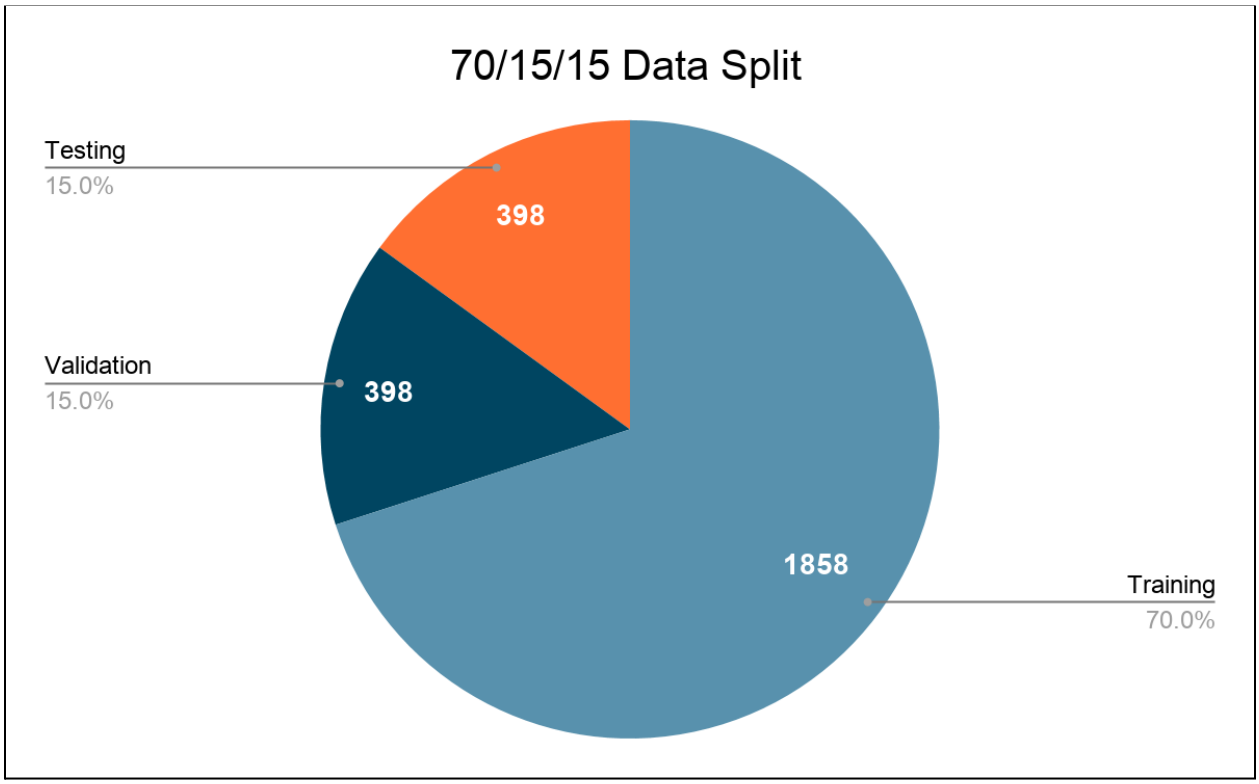

Figure 8.0 - Distribution of Data

For our testing dataset, is 15% of the entire dataset and has 398 unused audio samples. After completing the testing, from Figure 8.1 , you can see that the model is achieving a testing accuracy of . This is a clear indication that our model is performing well with new unseen data and is able to generalize. We did not use these samples to influence the tuning of the hyperparameters, yet we were able to achieve a testing accuracy extremely close to the validation accuracy.

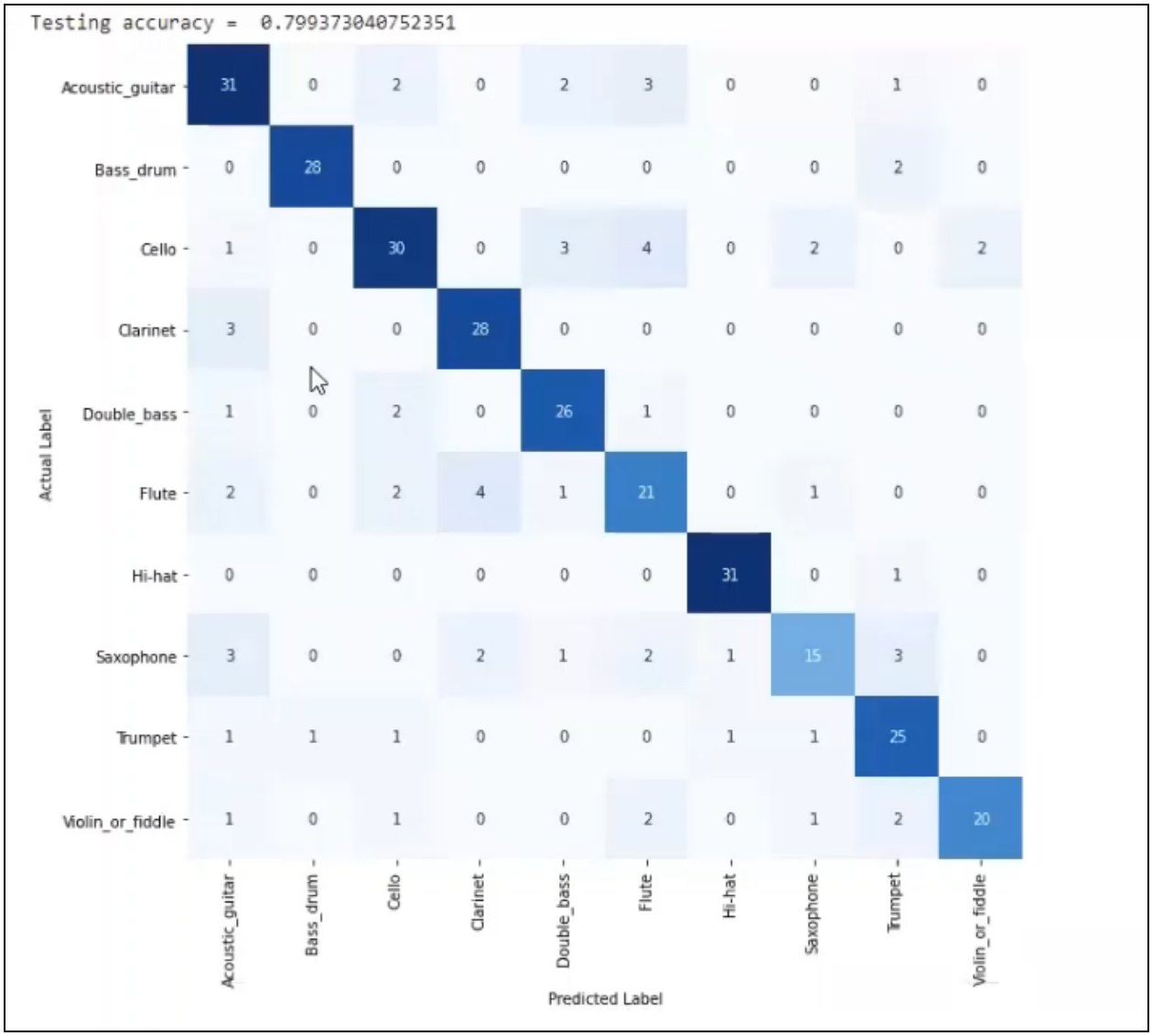

Figure 8.1 - Model Accuracy and Confusion Matrix on Testing Data

9.0 Discussion

Overall, it seems like the model is performing well because its testing accuracy was very similar to our validation accuracy while training (Figure 8.1). This shows us that the model is generalizing using our training data instead of overfitting/memorizing. As for the confusion matrix, shown in Figure 8.1, it relates back to our Quantitative analysis in the prior section, instruments with similar noises were not classified as accurately compared to those with distinct sounds. Similarly, an interesting point that can be made is that we only chose 1 specific way to visually represent our audio. There are multiple ways audio files can be displayed along with various features we can extract from them. This was definitely a major part in the performance of our model; with more defined features/visual images the results may have been better.

Throughout the course of the project, the team was able to learn the importance of data processing in developing AI models. In our case, it was a vital part of our project because making sure the data was processed correctly would make it much easier to train/build our model. Further, we were able to apply our knowledge of tuning hyperparameters in a unique setting where it taught us how important minor changes can directly impact the performance of our model.

A final observation we made while training the different models was that the accuracy and loss values would vary while re-training on the same parameters. Therefore, some of the numbers in this document may no longer match the values in the codebase exactly.

10.0 Ethical Considerations

Our ML model will be using instrumentals from online and as a result, we would need to consider the ethical standards behind using copyright music. In Canada, the law states that music is copyrighted “if its author is still living, or if the author died less than 50 years ago” [5], along with other details. This means for our model we need to limit our test set to non-copyright instrumentals. Another ethical consideration we need to make is to use a variety of styles of music each instrument can play along with different origins of those styles. This would make it so our model can generalize to all styles of music and to prevent the possibility it is inaccurate at predicting instruments within instrumentals that have different tempos/rhythm.

11.0 Project Difficulty/Quality

Overall we would say our model is a bit above average in terms of difficulty. The problem our model is trying to tackle is a single instrument classification problem as we're trying to classify an instrument being played in an audio file from a set of instruments. The data processing that went into our project was quite extensive as there were many preliminary tasks that needed to be done before sending in our data into our CNN as input.

The following tasks were done when processing our data:

- Collected audio data files with sounds from a Kaggle data set
- Eliminated audio files that were not instruments
- Discarded audio files that were not at least 2 seconds long
- Limited each audio file to 2 seconds
- Duplicated certain instruments samples that had less samples compared to others
- Split the data set into training, validation and testing set
- Converted audio files to images using librosa and python_speech_features
- Using the waveform representation created heat maps to represent this data

Due to our data processing being lengthy (mentioned above), we believe this is a bit more complex than what was needed to be done during the labs. For example in lab 3 all that needed to be sent into the CNN were the pictures of cats and dogs without much required data processing.

As mentioned earlier, our final CNN model was able to achieve a training accuracy of about 90.01% and a validation accuracy of about 80.06%. Based on the level of difficulty of our project we feel like our results meet our expectations. Depending on the quality of our heatmaps and the features extracted from each audio file, we may have been able to achieve a higher level of accuracy.

In the future if we wanted to make our project a bit more difficult, we could look into classifying multiple instruments from an audio file that contains several instruments being played. The level of difficulty would increase because our problem would then turn into a multi classification problem.

12.0 References

[1] “PixelPlayer - Identify and Extract Musical Instrument Sounds from Videos with MIT's AI,” Available:https://www.analyticsvidhya.com/blog/2018/07/pixelplayer-mit-open-source-ai-identifies-separates-instrument-sounds-video/. [Accessed: 9-Feb-2021].

[2] “Google is making Music with Machine Learning - and has released the code on GitHub,” [Online]. Available: https://www.analyticsvidhya.com/blog/2018/03/google-making-music-help-machine-learning/. [Accessed: 9-Feb-2021].

[3] "Freesound General-Purpose Audio Tagging Challenge | Kaggle", *Kaggle.com*, 2021. [Online]. Available: https://www.kaggle.com/c/freesound-audio-tagging/data. [Accessed: 9- Feb- 2021].

[4] M. Shah, "Wavfiles of Instruments' audio", *Kaggle.com*, 2021. [Online]. Available: https://www.kaggle.com/mayur1999/wavfiles-of-instruments-audio. [Accessed: 9- Feb- 2021].

[5] "Welcome to CMRRA", *Cmrra.ca*, 2021. [Online]. Available: https://www.cmrra.ca/. [Accessed: 10- Feb- 2021].