AUDIO TAGGING USING DIGITAL SIGNAL PROCESSING AND DEEP LEARNING

Ayush Kumar Shaw
19095023, B. Tech
Department of Electronics Engineering
Indian Institute of Technology(BHU)
Varanasi-221005
ayush.kumarshaw.ece19@itbhu.ac.in

Rochak Bhardwaj
19095083, B. Tech
Department of Electronics Engineering
Indian Institute of Technology(BHU)
Varanasi-221005
rochak.bhardwaj.ece19@itbhu.ac.in

Supervisor: Dr. M. Thottappan
Associate Professor
Department of Electronics Engineering
Indian Institute of Technology(BHU)
Varanasi-221005
mthottappan.ece@itbhu.ac.in

ABSTRACT

Audio Tagging is one of the preliminary tasks involved in Music information Retrieval(MIR). It refers to identifying the different instruments/sources present in audio clips at various instances. This has a huge number of applications in the music industry like adding tags to various audio clips automatically, which can enhance the way of searching these audio tracks. We use a Convolutional Neural Network and a Recurrent Neural Network to perform Audio Tagging after using Digital Signal Processing algorithms on the raw audio waveform and compare their performances.
Keywords- audio tagging, neural networks, signal processing

1 INTRODUCTION

Audio Tagging is an application of pattern recognition and machine learning in which an audio signal is mapped to a corresponding sound event. Audio Tagging is a very wide area of research, with one of its most widely used instances being Music Auto Tagging or Music Genre Classification. Music genres are helpful for grouping songs and artists to help listeners discover and explore new music. The problem involves, given an audio track, to identify the different instruments/vocals being used in any segment of the track. With the rise of learning based

methods in the past decade, it has been found that audio tagging can be performed very efficiently using deep learning models trained on enough data. Because of the scope of this project, we decided that, rather than choosing a single model and maximizing our results, we would attempt various approaches to the problem, comparing the results of each approach to decide which model would be the most promising for this task.

2 BACKGROUND

Audio Tagging has been done by researchers using spectrograms as input. These spectrograms are produced by taking the Fourier transformations of the raw waveform and efficiently represent the audio as a time-frequency plot which can be used directly as input to a CNN. Most of the ongoing work in this field uses mel-spectrograms as inputs to Deep Neural Networks as they can represent audio features that can be used in genre classification very well.
Mel Frequency Cepstral Coefficients(MFCC) is widely used as features for deep learning models in music genre classification and automatic speech recognition(ASR). Convolutional Neural Networks have been commonly used by researchers to perform audio tagging and have shown exceptional results. Generally the pre-trained weights of huge neural networks like VGG are used in Audio Tagging models using Transfer learning which requires lots of computational power. We have used a custom neural network architecture and trained it from scratch to tag audio tracks using a given set of tags. This custom architecture is much smaller and has fewer parameters and hence it is much faster and can be used to make real-time inferences from an incoming stream of audio data.

3 DATASET

We have used a subset of the Free Sound General Purpose Audio Tagging Challenge Dataset, which is available on Kaggle. The original Dataset consists of 41 diverse categories and is very huge. So we have selected 10 different categories and used its data as our dataset to test our approach, which include the following:

- ➔ Acoustic Guitar
- ➔ Bass Drum
- ➔ Cello
- ➔ Clarinet
- ➔ Double Bass
- ➔ Flute
- ➔ Hi-hat
- ➔ Saxophone
- ➔ Snare Drum
- ➔ Violin or Fiddle

Our dataset consists of 300 wav files consisting audio sampled at 44.1kHz. Each wav file consists of 1s of audio belonging to one of the above categories. The dataset is imbalanced i.e. it does not have the same amount of training data for each class.

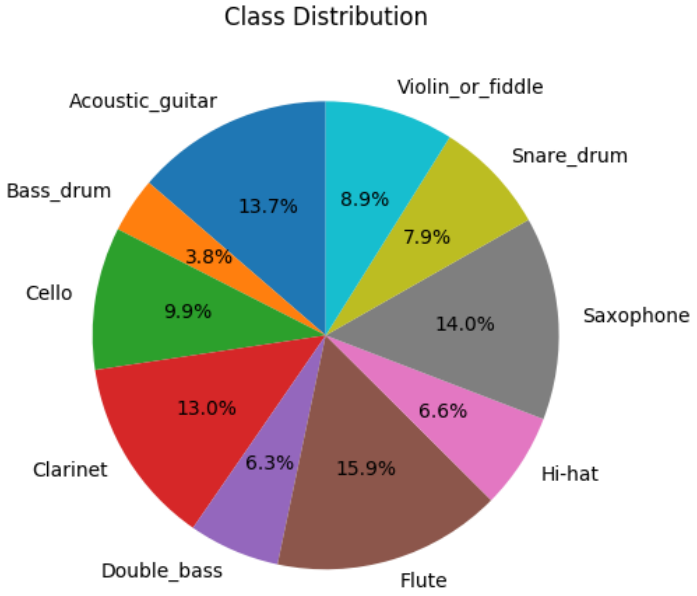


Figure 1: Class Distribution in dataset

4 PREPROCESSING AND FEATURE EXTRACTION

Preprocessing of raw audio is one of the most important parts of the entire pipeline. This involves converting the raw audio waveform to some meaningful form which can be used as features for our deep learning model. This is done in the following way:

Creating Signal Envelope to remove dead spaces

The first step of our audio processing pipeline involves downsampling the audio which was originally sampled at 44.1kHz. Audio sampled at this frequency would require more computation power, and hence we downsample the audio to 16kHz, which requires much less computation power in the subsequent steps without having any significant effect on the results of the model. This is because most of the significant frequency components and changes happen at lower frequencies in our dataset and so we can simply downsample the audio without compromising on the results. Generally raw audio files may contain a lot of dead spaces in the beginning or at the end, after the sound dies out. So, we create a signal envelope by taking the means of the values lying inside a sliding window and we discard the windows whose mean is less than a predefined threshold value. We have used a sliding window number of samples equal to 10% of the sampling rate and the threshold of 0.005 was selected after normalizing the values based on experimentation with different values of the threshold. This reduces the irrelevant data in the

audio files, while keeping only the meaningful and significant parts, and thereby cleans the raw signal.

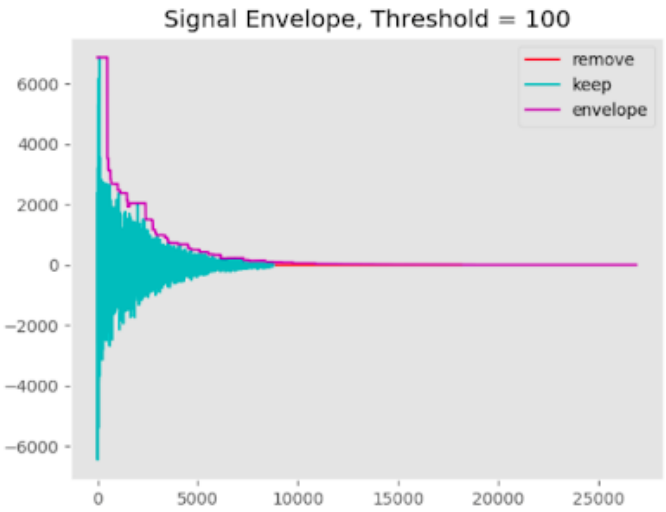
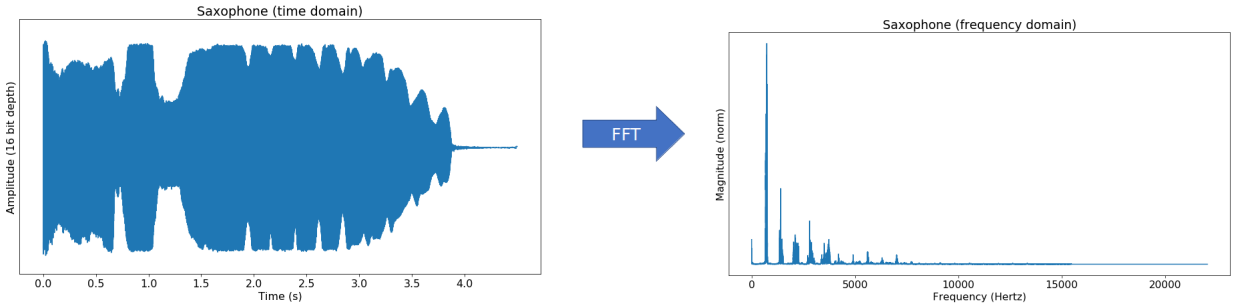


Figure 2: Signal Envelope

Framing and FFT

The next step of our pipeline involves taking the FFT of the signal to get a representation of the signal in the frequency domain. However, we cannot simply use the frequency domain representation in our models. So, we take the Short Time Fourier Transform(STFT) using N_FFT=512 samples and combine them using a Hann Window to prevent spectral leakage.



$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Figure 3: Fourier Transform

Because audio is a non stationary process, the FFT will produce distortions. To overcome this we can assume that the audio is a stationary process for a short period of time. Because of that we divide the signal into short frames(of 25ms). Each audio frame will be the same size as the FFT.

Also we want the frames to overlap(15ms overlap). We do that so that the frames will have some correlation between them and because we lose the information on the edges of each frame after applying a window function. Stacking these periodograms then gives us a spectrogram, which is a time-frequency representation of the waveform, and the intensity at a given point tells us the intensity of that frequency at that instant. We use a frame size of 25ms and a step size of 10ms with N_FFT=512 for computing the STFT.

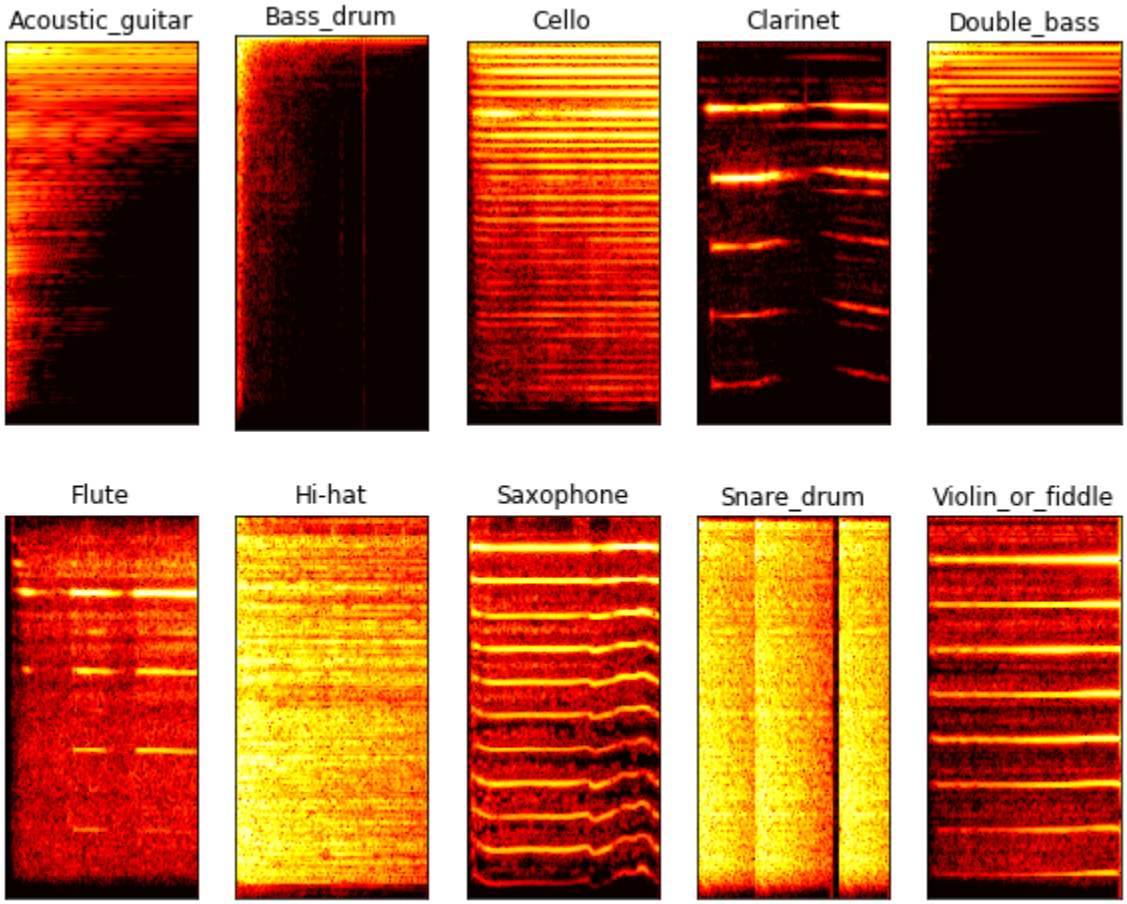


Figure 4:STFTs of the different instruments in the dataset

The MEL Scale

The MEL scale is the most commonly used alternate scale for frequency for deep learning applications. The MEL scale is a scale of pitches judged by listeners to be equal in distance one from another. Because of how humans perceive sound the MEL scale is a non-linear scale and the distances between the pitches increases with frequency.

$$M(f) = 1125\, ln(1 + f/700)$$

We compute the Mel spaced filterbank, which is a set of 26 triangular filters and apply these filters, typically on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale

aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. Each filter in the filter bank is triangular having a response of 1 at the center frequency and decrease linearly towards 0 till it reaches the center frequencies of the two adjacent filters where the response is 0, as shown in this figure:

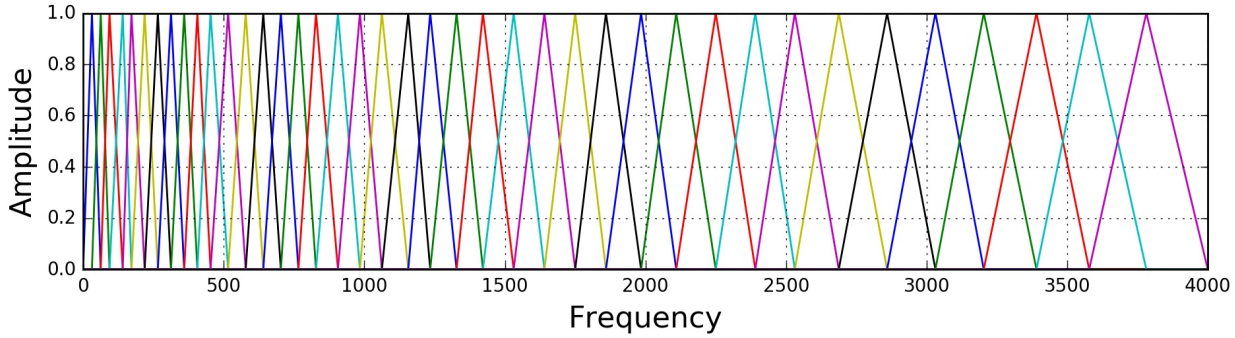


Figure 5:Filter Banks on a Mel Scale

Since our STFTs had a lot of overlap and hence the Mel filter bank coefficients are highly correlated which cannot act as good features for our deep learning model. Therefore, we can apply Discrete Cosine Transform (DCT) to decorrelate the filter bank coefficients and yield a compressed representation of the filter banks with the number of retained cepstral coefficients being 13. This finally gives us the MFCCs which will act as features for our deep learning models. These features represent the large structures in the spectrum eliminating the noise and unwanted data.

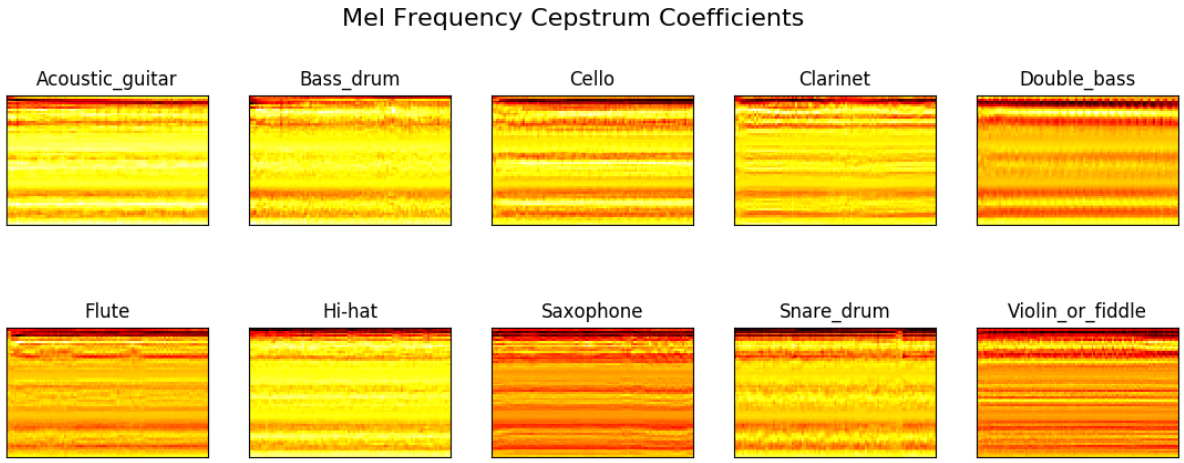


Figure 6:MFCCs for different instruments

5 METHODS

For classification, using the MFCCs as features, we tried using 2 different types of neural networks:

Convolutional Neural Network(CNN)
CNNs have given exceptional results in image related learning based tasks and they hence, we use a CNN for classifying our audio data, as the MFCC feature matrix can be processed by a CNN very efficiently. We have used a custom architecture having 4 Convolutional layers, with the number of filters increasing successively. We have used the RELU activation for the convolutional layers with SAME padding so as to preserve the dimensions of the matrix after passing through the convolutional layers. We have also added a dropout layer to prevent overfitting on the training data.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 9, 13, 16)         160
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 9, 13, 32)         4640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 13, 64)         18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 13, 128)        73856
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 6, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 4, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3072)              0
_________________________________________________________________
dense (Dense)                (None, 128)               393344
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 499,402
Trainable params: 499,402
Non-trainable params: 0
```

CNN Model Summary

We used the Adam Optimizer and categorical cross-entropy to train the network. We also assigned weights to different classes while training to handle the imbalance in the dataset, and to make a robust model. Using this network architecture, we attained a maximum validation

accuracy of 96.25%. Thus, this is a very light weight model and hence has a very less inference time and can be easily used for real time audio tagging.

Recurrent Neural Network(RNN)

We have used RNN to perform classification using the Long Short Term Memory(LSTM) layers. These layers have memory and perform very well in tasks like Sequence classifications, where the features in the beginning of sequences may be related to the features at a later portion in the sequence. Hence, RNNs can be a good choice for performing audio tagging. We have used 2 LSTM layers followed by a Dropout and Time Distributed Dense layers because LSTMs return entire sequences.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 9, 128)            72704
_________________________________________________________________
lstm_1 (LSTM)                (None, 9, 128)            131584
_________________________________________________________________
dropout (Dropout)            (None, 9, 128)            0
_________________________________________________________________
time_distributed (TimeDistri (None, 9, 64)             8256
_________________________________________________________________
time_distributed_1 (TimeDist (None, 9, 32)             2080
_________________________________________________________________
time_distributed_2 (TimeDist (None, 9, 16)             528
_________________________________________________________________
time_distributed_3 (TimeDist (None, 9, 8)              136
_________________________________________________________________
flatten (Flatten)            (None, 72)                0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                730
=================================================================
Total params: 216,018
Trainable params: 216,018
Non-trainable params: 0
```

RNN Model Summary

We used the Adam optimizer and the categorical cross-entropy loss for training the RNN as well. With the RNN, we attained a maximum validation accuracy of 86.16%, after training the network for 10 epochs.

6 APPLICATIONS

Automatic audio event detection is utilized in a host of applications, including surveillance, speech detection and audio segmentation. This task is also particularly challenging because it

involves large amounts of classes and multi-label classification. Audio tags are primarily used for the allotment of genres to music and thus enhancing the search process from a huge audio database. Our model is very lightweight and has an extremely low inference time due to its compact architecture and can be used in real time applications like surveillance. In addition, it can be used in general purpose audio tagging applications and can be tweaked easily to perform music genre classification/music tagging.

7 RESULTS / DISCUSSION

We performed several experiments varying the hyperparameters of the model like the learning rate and tried some variations in the model architecture as well. From the experiments, it was evident that the CNN gave better results than the RNN and also took comparatively less time for training. We trained the models on Google Colab platform and used python and its libraries like librosa and Tensorflow for audio preprocessing and model training. The MFCCs turned out to be a very good and compact representation of the audio waveform for tasks like audio tagging and helped us in getting validation accuracies upto 97% with a relatively simple model architecture. The modern day applications generally have a large number of classes as tags, and hence using a CNN would be one of the best ways to tag the audio tracks, as it can give very good accuracy without making the network too complex.

8 CONCLUSION / FUTURE WORK

In this work, we take a novel approach to audio tagging, which reduces the complexity of the automatic music tagging networks that are in use. We have used a custom network architecture and trained it from scratch using MFCCs as features, on a dataset consisting of 10 different classes. Additionally, we also trained a CNN and RNN to perform audio tagging using the same features and compared their results based on metrics such as validation accuracy.
This can be readily transferred to real life applications and can be extended to be used for real time audio tagging and in surveillance systems due to its simple architecture and low inference time. This network can be trained on larger datasets with more number of classes with a more complex network architecture to perform tasks like automatic music tagging.

The files and code used in the project can be found in the following GitHub repository:
https://github.com/aksayushx/Audio-Tagging

9 REFERENCES

[1] Multimodal Deep Learning for Music Genre Classification. Transactions of the International Society for Music Information Retrieval

[2] Music Genre Classification Using Spectral Analysis and Sparse Representation of the Signals.
[3] Practical Cryptography: A Guide to MFCC
[4] Audio Classification Series by Seth Adams
[5] Audio tagging with noisy labels and minimal supervision, Eduardo Fonseca
[6] Audio Tagging System for DCASE 2018
[7] ResearchGate, Short Time Fourier Transform Based Music Genre Classification, 2018.