UNIVERSITÀ DEGLI STUDI
DI MILANO

STATISTICAL METHODS FOR MACHINE LEARNING

Instructor: Nicolò Cesa-Bianchi

Project: Urban Sound Classification with Neural Networks

Dataset: **UrbanSound8k**

Student: Nallapaneni Aditya Sai(933570) - Data Science and Economics

Abstract:

This document illustrates a deep learning-based audio classification model. We address the problem of classifying the type of sound based on audio signals and their generated spectrograms, from labeled sounds belonging to 10 different folds and with 8732 real-world sound clips. In order to meet this challenge, we use a model based on Convolutional Neural Network (CNN). The audio was processed with Mel-frequency Cepstral Coefficients (MFCC) into what's commonly known as Mel spectrograms of Librosa .Our final CNN model we achieved good accuracy on the testing dataset.

DECLARATION:

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

CONTENTS

1. CONVOLUTION NEURAL NETWORKS

CNNs are a particular type of neural networks, which use the convolution oper- ation in one or more layers for the learning process. A CNN is composed by three main layers:

– **Convolutional Layer**: The convolutional layer is the one tasked with ap- plying the convolution operation on the input. This is done by passing a filter (or kernel) over the matricial input, computing the convolution value, and using the obtained result as the value of one cell of the output matrix (called feature map); the filter is then shifted by a predefined stride along its dimensions. The filters parameters are trained during the training process.

– **Detector layer**: In the detector layer, the output of the convolution is passed through a nonlinear function, usually a ReLU function.

– **Pooling layer**: The pooling layer is meant to reduce the dimensionality of data by combining the output of neuron clusters at one layer into one single neuron in the subsequent layer.

The **last layer** of the network is a fully connected one (a layer whose units are connected to every single unit from the previous one), which outputs the probability of the input to belong to each of the classes.

CNNs in a machine learning system show some advantages with respect to traditional fully connected neural networks, because they allow sparse interactions, parameters sharing and equivariant representations.

The reasons why we used CNNs in our approach is due to the intrinsic nature of audio signals. CNNs are extensively used with images and, since the spectrum of the audio is an actual picture of the signal, it is straightforward to see why CNNs are a good idea for such kind of input, being able to exploit the adjacency properties of audio signals and recognize patterns in the spectrum images that can properly represent each one of the classes taken into consideration.

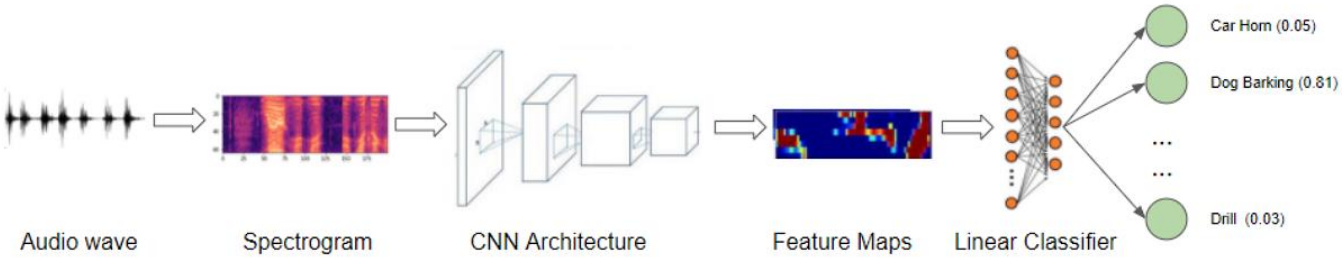


Fig 1

2. Libraries and Methods/techniques used:

We are using multiple packages for loading data, extracting features and model training.

NumPy and librosa for features extracting:
NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences. Librosa is a python package for music and audio analysis. It provides the buil- ding blocks necessary to create music information retrieval systems, and we used the Librosa audio package in python to extract features from the audio files in our dataset. The following are the features that we extracted: (??) Mel- Frequency Cepstral Coefficients (MFCC): Coefficients derived from a cepstral representation of the audio clip, (??) Mel-scaled spectrogram: Psychoacous?tic scales that capture the distances from low to high scale frequency, (??) Tonnetz: Representation of tonal space. Visualizations of the .wav files formats and the mel-scaled spectrograms for each class in the data set can.

TensorFlow and Keras:
Keras on the other hand, is a high-level neural networks library that is running on the top of TensorFlow, CNTK, and Theano. Using Keras in deep learning allows for easy and fast prototyping as well as running seamlessly on CPU and GPU.
TensorFlow is a comprehensive and flexible ecosystem of tools, libraries and other resources that provide workflows with high-level APIs. The framework offers various levels of concepts for you to choose the one you need to build and deploy machine learning models.

Sklearn for checking if any benefits would come from reducing features dimensions, As well as using it to as labels encoding and reporting the model

Batch Normalization:
A powerful technique that reshapes the distribution of activations (making them close to Normal Distribution), which improves the performance and stability of the network.
Batch Norm = (Activation – Mean Activation)/S.D
With Mean Activation close to 0 and Active Standard Deviation of the distribution close to 1.

Adam optimization:
Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Where as Stochastic gradient descent maintains a single Learning rate (termed alpha) for all weight updates and the learning rate does not change during training. learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.This method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

3. DATA EXPLORATION AND ANALYSIS

The dataset used in this project is UrbanSound8K Dataset. In this dataset, there are 8732 urban sound excerpts which are labeled and divided into 10 folders. Each audio file has a name formatted as [fsID]-[classID]- [occurrenceID]-[sliceID].wav, where

1. fsID is the Freesound ID of the recording from where this excerpt (slice) is taken.
2. classID is a numeric identifier of the sound class.
3. occurrenceID is a numeric identifier to distinguish different occurrences of the sound within the original recording.
4. sliceID is a numeric identifier to distinguish different slices taken from the same occurrence.

All the 8732 audio files are divided into 10 classes with different numbers of tracks. The 10 classes are *Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music*. Each of the classes is represented by a label.

Audio Signal: Audio Signal: A complex signal composed of multiple “single - frequency” sound waves. When sound is recorded, we only capture the resultant amplitude of those multiple waves. A signal is a variation of a certain quantity over time.

3.1 Fast Fourier Transform

An audio signal is composed of several single-frequency sound waves. When taking samples of the signal over time, we only capture the resulting amplitudes. The Fourier transform is mathematical formula that allows us to decompose a signal into its individual frequencies and the frequency’s ampli tude.

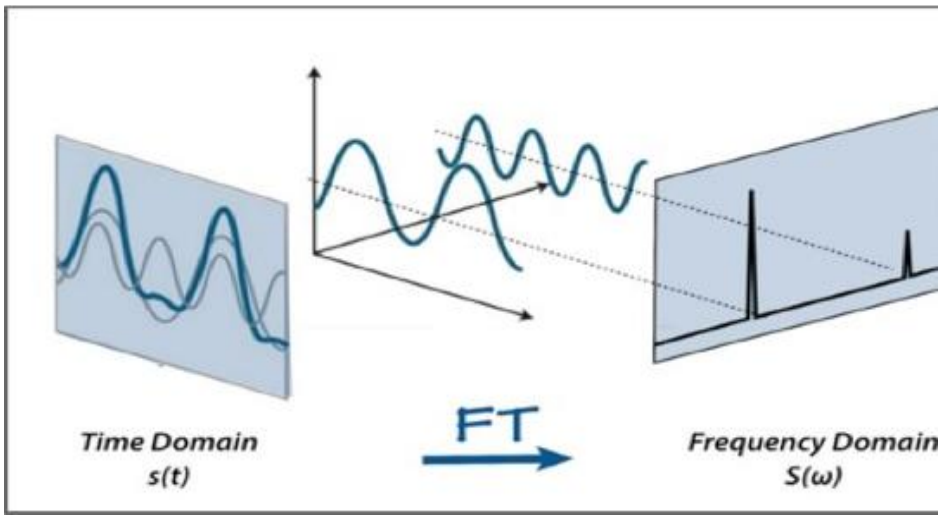


Fig 2

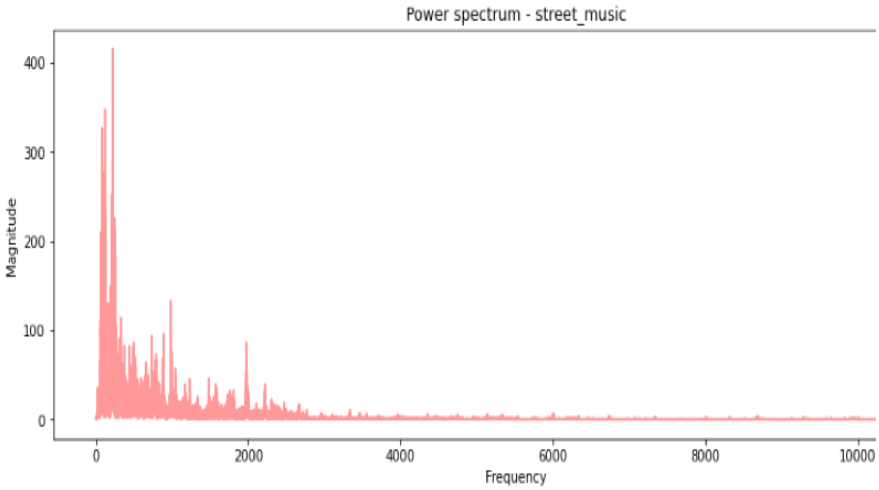


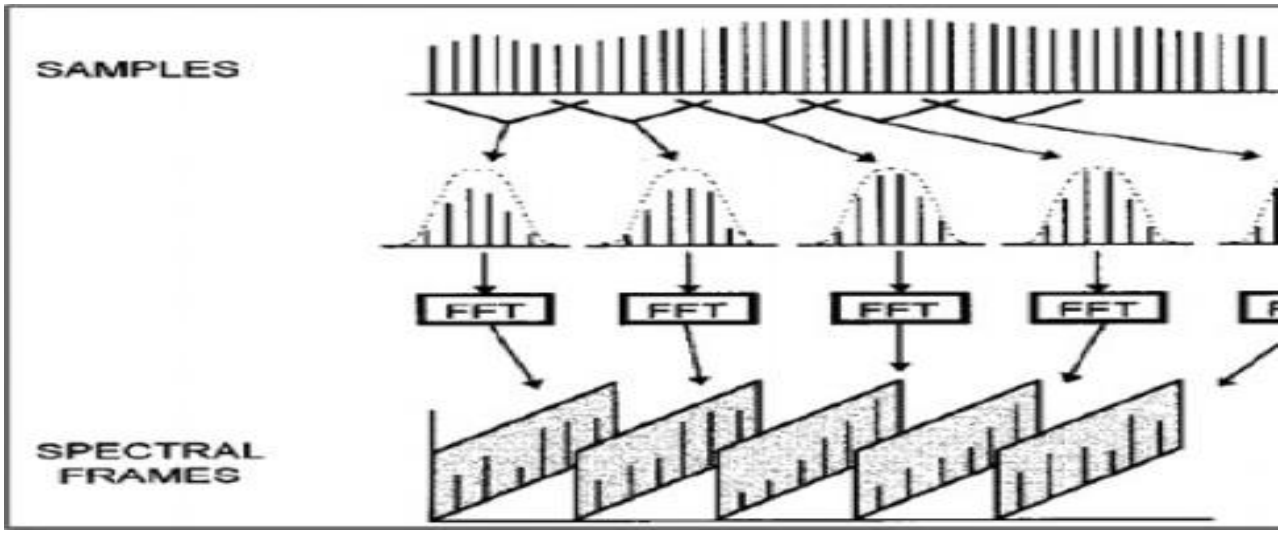
Fig 3

It converts the signal from the time domain into the frequency domain. The result is called a spectrum. Basically, Fourier Transform decomposes complex periodic sound into a sum of sine waves oscillating at different frequencies. Following is the spectrum I got after applying Fourier Transformation of audio from street music class which is Fig3.

But, when we apply FFT, we lose time information. And, if we lose time information then our system won't be able to tell which music sound played first if we use only frequencies as a feature. And thus we will go for the next transformation.

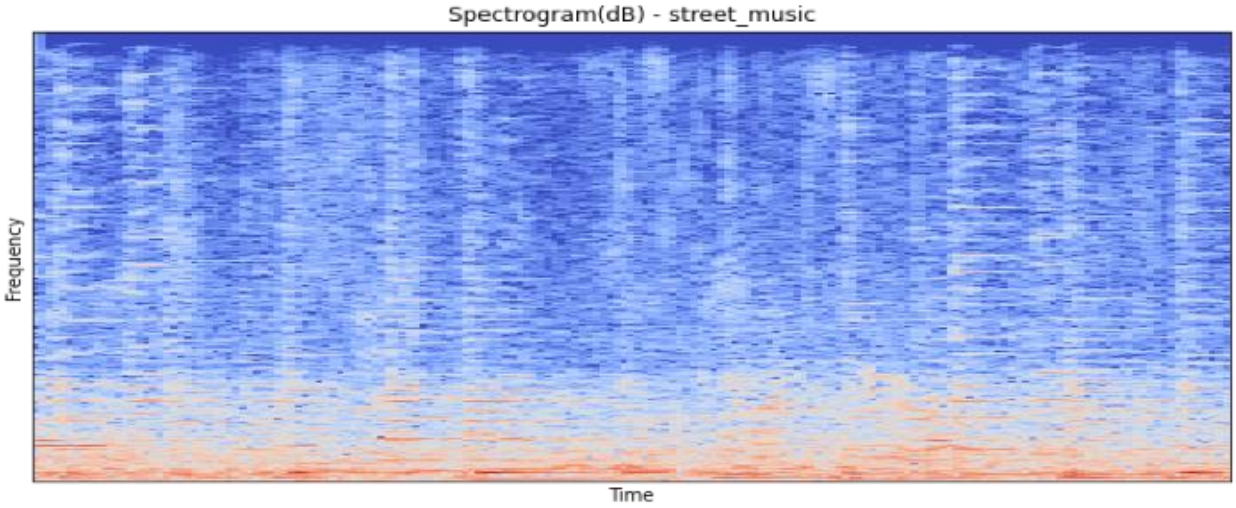
3.2 Spectrograms & Short Term Fourier Transform(STFT)

Spectrograms are introduced because we were losing time information after applying Fast Fourier Transformation. A spectrogram shows how the frequency content of a signal changes over time and can be calculated from the time domain signal. Short Term Fourier Transform comes to our rescue here.



A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given music signals.
How Spectrogram Preserves Time Information?

When we perform STFT on any audio data, the first step is to break the entire audio signal into smaller frames (or windows) and then for each window we calculate FFT. This way we will be getting frequencies for each window and window number will represent time information. Generally, window size is between 20ms to 30ms long. These frames must overlap with each other such that we do not lose any information. Generally, overlapping of 25% to 75% is done. Following is the Spectrogram after applying STFT on our one of the audio file from street music and car horn class:



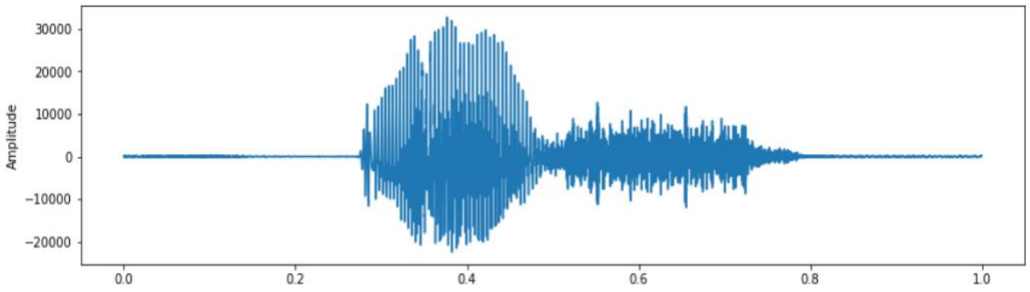
As a result of STFT, we got a spectrogram which gives us information about magnitude as a function of frequency and time. Audio that sounds drastically different, results in drastically different spectrograms.
Steps involved in extracting features from audio data :

1. I took samples of air pressure over time to digitally represent an audio signal.
2. Mapped the audio signal from the time domain to the frequency domain using the fast Fourier transform, and we performed this on overlapping windowed segments of the audio signal.
3. Converted the y-axis (frequency) to a log scale and the color dimension (amplitude) to decibels to form the spectrogram.
4. Mapped the y-axis (frequency) onto the mel scale to form the mel spectrogram.

4. DESCRIPTION OF EXPERIMENTS

Data Preprocessing and Feature Extraction

UrbanSound8K dataset has 8732 music files and each file is less than or equal to 4 seconds long. Now, to train the CNN model, I have extracted MFCC features from these audio files. Entire database of extracted features was created and stored in the drive All audio samples are in .wav format and are sampled at discrete periods of time at the standard sampling rate (44.1kHz meaning 44,100 samples per second). The bit depth determines how detailed the sample will be (typically 16-bit samples can range to about 65 thousand amplitude values) and every sample is the amplitude of the wave at a particular instance of time. Therefore, the data we will be using for every audio sample is basically a unidimensional vector of amplitude values.



Mel - Frequency Cepstral Coefficient (MFCC)

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice.

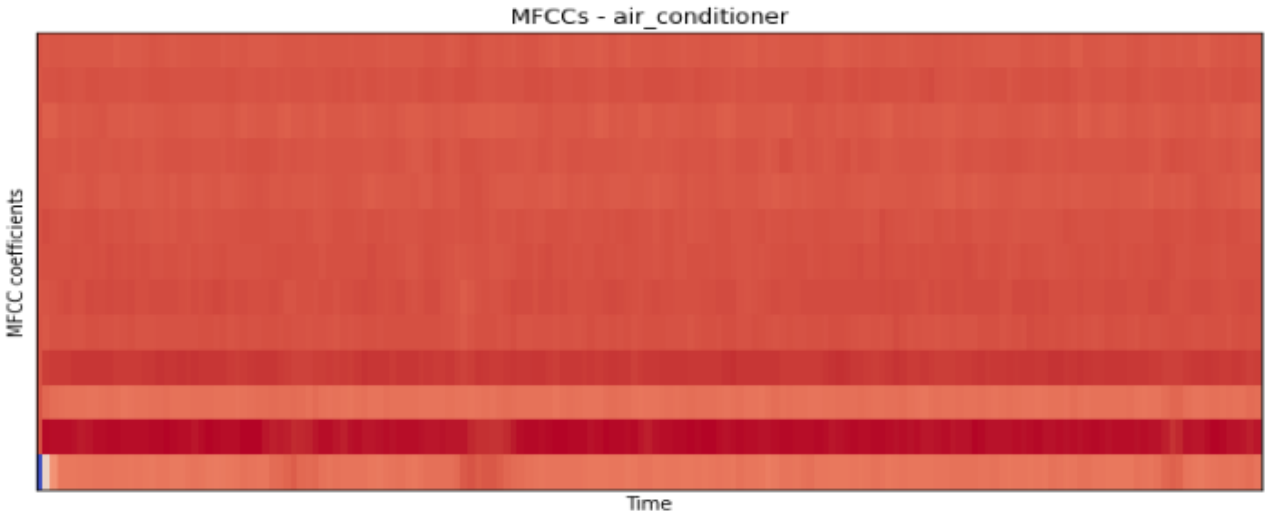
Pitch is one of the characteristics of a speech signal and is measured as the fre- quency of the signal. Mel scale is a scale that relates the perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear (humans are better at identi- fying small changes in speech at lower frequencies). A frequency measured in Hertz (f) is converted to the Mel scale using the following formula :

Any sound generated by humans is determined by the shape of their vocal tract. If this shape can be determined correctly, any sound produced can be accurately represented. The envelope of the time

power spectrum of the speech signal is representative of the vocal tract and MFCC (which is nothing but the coefficients that make up the Mel-frequency cepstrum) accurately represents this envelope. The following block diagram is a step-wise summary of how we arrived at MFCCs:

$$\mathrm{Mel}(f) = 2595 \log\left(1 + \frac{f}{700}\right)$$

After performing all these steps I got the extracted MFCC features. Following are the visual representation of audio file after extracting MFCC:



PADDING

Padding it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. Padding is to make sure all data are in the same shape so we can passs them to the model as input without a problem of inadequacy.

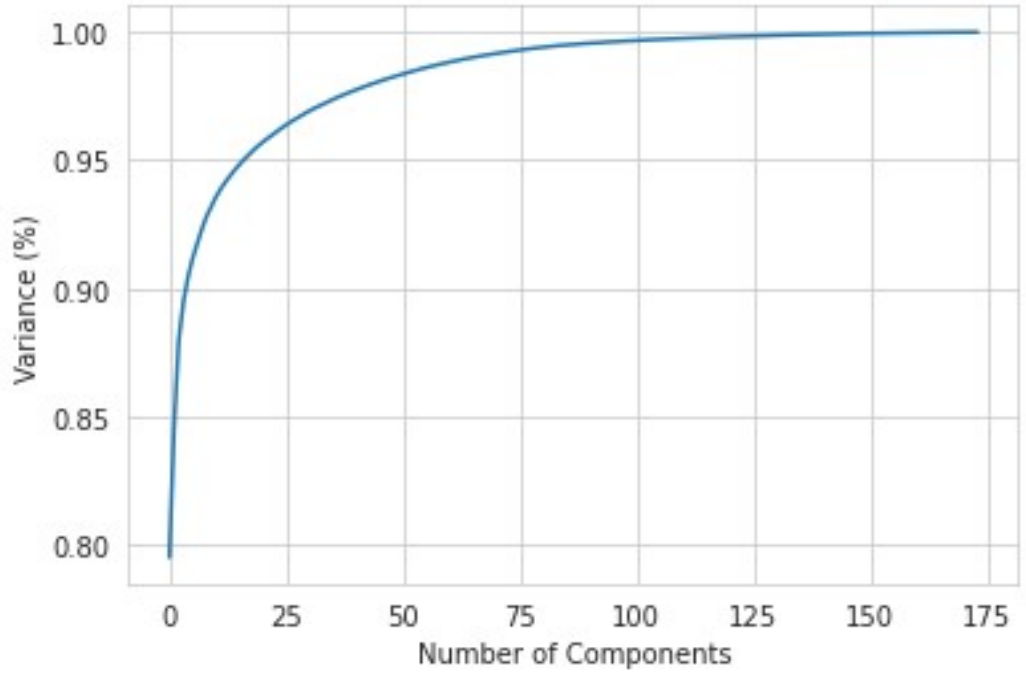
Normalizing the Data

the reason we normalize the images is to make the model converge faster. When the data is not normalized, the shared weights of the network have different calibrations for different features, which can make the cost function to converge very slowly and ineffectively. Normalizing the data makes the cost function much easier to train.

[49]: num rows = 40 num columns = 174 num channels = 1
[52]: le = LabelEncoder() y test encoded = to categorical(le.fit transform(y test))
y train encoded = to categorical(le.fit transform(y train))
[53]: *# Reshape to fit the network input (channel last)*
X train = X train.reshape(X train.shape[0], num rows, num columns, num channels) X test = X test.reshape(X test.shape[0], num rows, num columns, num channels)
[ ]: *# Total number of labels to predict (equal to the network output nodes)* num labels
= y train encoded.shape[1]

PCA NOISE REDUCTION

PCA, is a dimensionality-reduction method that is often **used** to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set, Before building our model we checked if we would benefit of some noise reduction using PCA.



And as we can see most of the variance is explained using all the features of the MFCC. This is expected since each feature gives information about the wave shape. Finally we can fit our model.

Model Compilation

We use the following parameters for compiling our model:
Loss Function : To compute loss, we use categorical_crossentropy, in which a lower score indicates that the model is performing better.

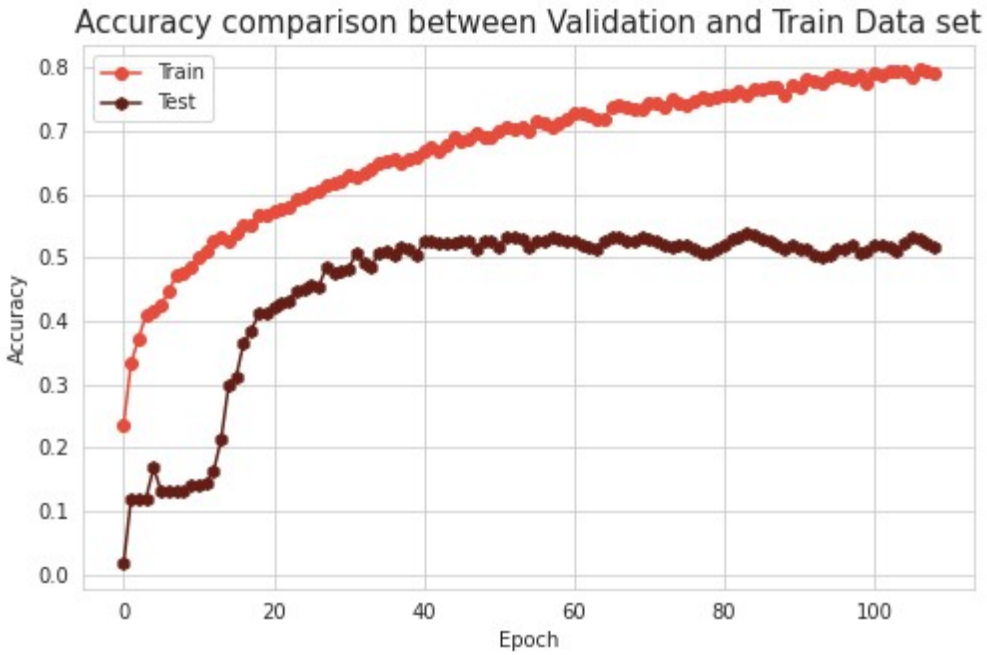
Metric : We use the *accuracy* metric which enables us to calculate the accuracy score on the validation dataset while training the model.

Rectified Linear Activation Function

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. Defined as the positive part of the argument where x is the input to a neuron, allows faster and effective training of deep neural architectures on large and complex datasets. We Trained all the models for 250 epochs.

Training and testing

In order to avoid both training and testing set having related samples, we can take data from 10 of 5 predefined folds as training set (1,2,3,4,6) and from the remaining 5 fold as testing set (5,7,8,9,10). We implement these 10-fold cross-validation in our CNN model and average test accuracy is about 65%. Below we can observe the accuracy in each epochs.



5. CONCLUSION

Convolutional neural networks can classify sound clips to a high degree of accuracy through the use of image representations with more filters in later layers outperform simpler ones when working with similar images Urban sound 8K classification dataset is a very large and extensive dataset that comes from a significantly more difficult problem where it contains audio files with noisy features. Where as CNN's are a good approach for Image Classification Tasks and Dropout works as a better in our case. Where we build a spectrogram to convert the audio files into images and train our model with corresponding folds as we tested with 5 of the folds

Adaptive Moment Estimation (Adam) computes adaptive learning rates for each parameter in storing an exponentially decaying average of past gradients. However, adaptive methods train the network faster and Batch Normalization is a good technique to improve the overall performance of the network Real time Data Augmentation can help the network achieve better performance. the proposed architecture was tested on five classes obtaining an accuracy of 65%. And we also implemented confusion matrix which Is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in the dataset. We have also seen the accuracy per class and Below is the classification report shown for our dataset.

	precision	recall	f1-score	support
Air Conditioner	0.67	0.66	0.66	517
Car Horn	0.78	0.68	0.73	209
Children Playing	0.69	0.69	0.69	528
Dog bark	0.73	0.68	0.70	475
Drilling	0.60	0.56	0.58	486
Engine Idling	0.85	0.81	0.83	485
Gun Shot	0.68	0.97	0.80	195
Jackhammer	0.65	0.75	0.70	431
Siren	0.85	0.72	0.78	383
Street Music	0.69	0.73	0.71	524
accuracy			0.71	4233
macro avg	0.72	0.73	0.72	4233
weighted avg	0.71	0.71	0.71	4233