EE603A Assignment-1 : Audio Classification

Arya Bhatta
Roll Number:190184
Department of Electrical Engineering
Indian Institute Of Technology Kanpur
aryab@iitk.ac.in

1 Introduction

Audio classification or sound classification can be referred to as the process of analyzing audio recordings to classify sounds. This type of problem has many applications, such as chatbots, virtual assistants, music genre identification, text-to-speech applications, etc.

In this assignment, we were required to classify an audio sample as one of the 10 possible categories using deep learning models. The given training data consisted of 1000 Melspectrograms and their labels. The parameters used for generating the Melspectrograms from the audiofiles were, Nfft = 2048, nmels = 128, hop length = 512, windowing function = hann. Now, since our data now consists of Spectrogram images, we build a Convolutional Neural Network(CNN) classification architecture to process them. We also built a simple Neural Network model but we saw that the CNN model gave better results.

2 Literature Survey

Audio data is obtained by sampling the sound wave at regular time intervals and measuring the intensity or amplitude of the wave at each sample. The audio is then stored as a numpy array. But models don't generally take this raw audio directly as input, but this audio is often pre-processed and then feed into models. Mel-Spectograms, Short-Term Fourier Transform (STFT) and MFCC (Mel-Frequency Cepstral Coefficients) are all popular ways to process audio signals and generate features as input for machine learning algorithms such as Convolutional Neural Networks. We will be using Melspectrograms as inputs for our models.

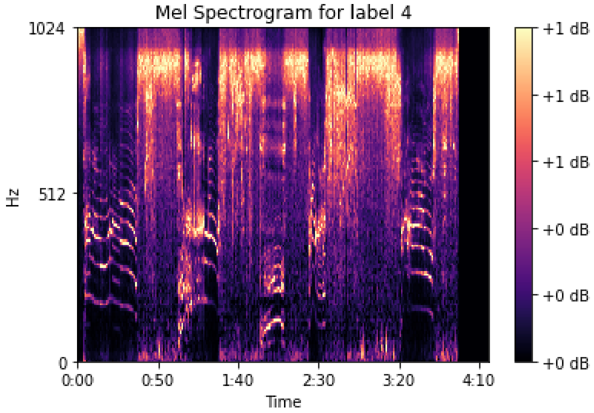


Figure 1: Melspectrogram

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2.1 Data Analysis

As training data we are provided with 1000 Melspectrograms which represent different samples of audio. The training data is equally balanced as it contains equal number of samples from all the 10 classes. The dimensions of the Spectrograms are (1,128,x) where x varies according to the time length of the audio. From the statistics of the third dimension we see that it has a mean of 590 and standard deviation of 600. So, only a few spectrograms have their third dimension which is very large(>1200). Therefore, we can choose limit for the third dimension and either clip or pad the spectrograms accordingly.

3 Methods

The problem at hand was largely similar to image classification with each sample's melspectrogram having 1 channel. The other two dimensions correspond to mel bands which are 128 and time which is different for each sample with the highest value being 2584. This means that the number of parameters can range upto $1 \times 128 \times 2584 = 330752$, which is quite high for a simple linear classifier model. Hence, we employed deep learning techniques.
After analysis of the variation in sizes of the third dimension of melspectrogram of different samples, it was observed that their mean and standard deviation both were close to 500. So we fixed the value to be 1500 for all the samples and used zero padding for the samples with value less than 1500.
The data has been normalised using $MinMaxScaler$ using a feature range of $(0, 1)$

3.1 Dense Neural Network

The network consists of different layers with the input layer being the flattened (linear array) version of the 2-d array corresponding to the melspectrogram of each sample. There are 3 hidden layers which are dense with the number of units being 512, 256 and 256 respectively. $ReLU$ activation has been used for all the hidden layers. The final layer has number of units equal to 10 (for 10 classes) and activation as $softmax$. The model has been trained using $CategoricalCrossEntropyLoss$.

3.2 Convolutional Neural Network

The spectrogram is a visual representation of an audio wave and since it is an image, it is well suited to being input to CNN-based architectures developed for handling images.
Our CNN model consists of 3 convolution layers consisting of 16, 32, 64 filters and having sizes (5×5), (3×3), (3×3) respectively. In between these convolution layers we have used pooling layers having filters of size (2×2). We have also used dropout layers with probability rate of 0.2. Then we used Flatten to convert our 3D feature maps to 1D feature vectors. Then similar to a Neural Network we use a couple of dense layers to train our data. Till now in all our $Conv2D$ and $Dense$ layers, we have used $ReLu$ for activation. Finally, we added a dense layer with $softmax$ to get out final output of probabilities of different classes. In our model we used $CategoricalCrossentropy$ loss function as it is a multi-class classification problem.

4 Observation

The first row corresponds to the NN Model.
The second row corresponds to the CNN Model.

4.1 Precision

Bark	Crying and sob-bing	Doorbell	Knock	Meow	Microwave oven	Shatter	Siren	Vehicle horn	Walk
0.53	0.68	0.93	0.43	0.59	0.57	0.7	0.79	0.46	0.38
0.7	0.71	0.75	0.64	0.54	0.84	0.73	0.84	0.8	0.64

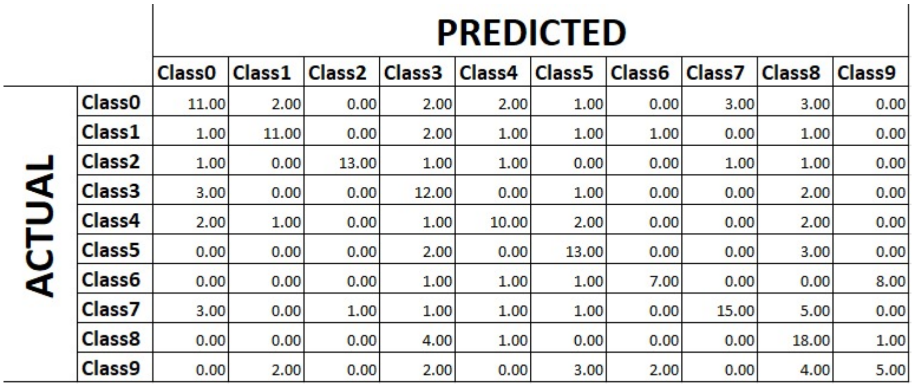
4.2 Recall

Bark	Crying and sobbing	Doorbell	Knock	Meow	Microwave oven	Shatter	Siren	Vehicle horn	Walk
0.46	0.61	0.72	0.66	0.55	0.72	0.38	0.55	0.75	0.27
0.75	0.67	0.83	0.88	0.77	0.88	0.62	0.6	0.67	0.5

4.3 F1 Score

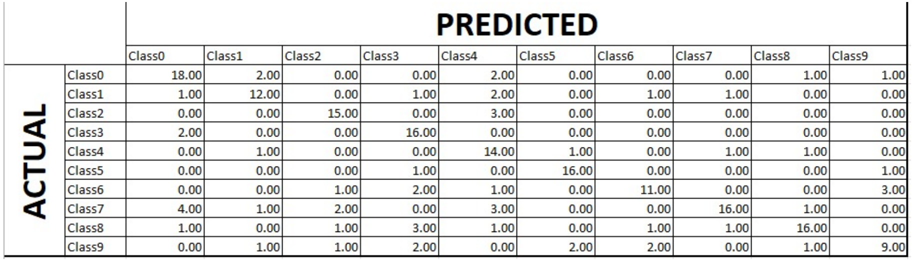
Bark	Crying and sobbing	Doorbell	Knock	Meow	Microwave oven	Shatter	Siren	Vehicle horn	Walk
0.49	0.65	0.81	0.52	0.57	0.63	0.5	0.65	0.57	0.31
0.72	0.68	0.79	0.74	0.64	0.86	0.67	0.7	0.72	0.56

4.4 Confusion Matrix



		PREDICTED									
		Class0	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9
ACTUAL	Class0	11.00	2.00	0.00	2.00	2.00	1.00	0.00	3.00	3.00	0.00
	Class1	1.00	11.00	0.00	2.00	1.00	1.00	1.00	0.00	1.00	0.00
	Class2	1.00	0.00	13.00	1.00	1.00	0.00	0.00	1.00	1.00	0.00
	Class3	3.00	0.00	0.00	12.00	0.00	1.00	0.00	0.00	2.00	0.00
	Class4	2.00	1.00	0.00	1.00	10.00	2.00	0.00	0.00	2.00	0.00
	Class5	0.00	0.00	0.00	2.00	0.00	13.00	0.00	0.00	3.00	0.00
	Class6	0.00	0.00	0.00	1.00	1.00	1.00	7.00	0.00	0.00	8.00
	Class7	3.00	0.00	1.00	1.00	1.00	1.00	0.00	15.00	5.00	0.00
	Class8	0.00	0.00	0.00	4.00	1.00	0.00	0.00	0.00	18.00	1.00
	Class9	0.00	2.00	0.00	2.00	0.00	3.00	2.00	0.00	4.00	5.00

Figure 2: Dense NN Model



		PREDICTED									
		Class0	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9
ACTUAL	Class0	18.00	2.00	0.00	0.00	2.00	0.00	0.00	0.00	1.00	1.00
	Class1	1.00	12.00	0.00	1.00	2.00	0.00	1.00	1.00	0.00	0.00
	Class2	0.00	0.00	15.00	0.00	3.00	0.00	0.00	0.00	0.00	0.00
	Class3	2.00	0.00	0.00	16.00	0.00	0.00	0.00	0.00	0.00	0.00
	Class4	0.00	1.00	0.00	0.00	14.00	1.00	0.00	1.00	1.00	0.00
	Class5	0.00	0.00	0.00	1.00	0.00	16.00	0.00	0.00	0.00	1.00
	Class6	0.00	0.00	1.00	2.00	1.00	0.00	11.00	0.00	0.00	3.00
	Class7	4.00	1.00	2.00	0.00	3.00	0.00	0.00	16.00	1.00	0.00
	Class8	1.00	0.00	1.00	3.00	1.00	0.00	1.00	1.00	16.00	0.00
	Class9	0.00	1.00	1.00	2.00	0.00	2.00	2.00	0.00	1.00	9.00

Figure 3: CNN Model

5 Discussion

The Melspectrograms are images representing the audio files. So, if we wish to use a neural network to train these spectrograms, then we will have to deal with a huge number of weights, i.e. a typical spectrogram from our data has dimensions (1,128,512) which is equal to $1 \times 128 \times 512 = 65536$ input features. Let our 1st hidden layer has 1024 nodes, then the total number of weights for the first layer alone is $65536 \times 1024 = 67+$ million.
Also, if the audio signal is displaced in time then we will get different spectrograms than before

and hence the neural network will not recognize the same audio signal. Therefore, CNN models are preferred as they mitigate these problems efficiently.

From the F1 scores, we can see that the CNN model performs better than the NN model.

Here are the plots for training and validation accuracy.

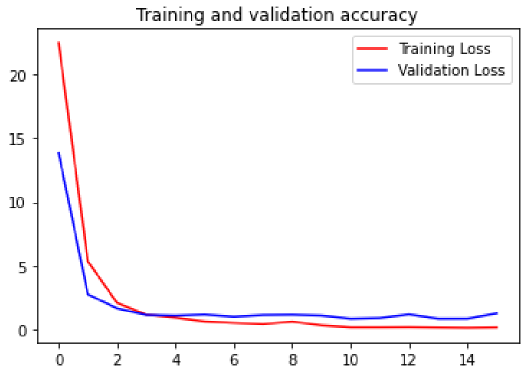


Figure 4: Dense NN training and validation loss

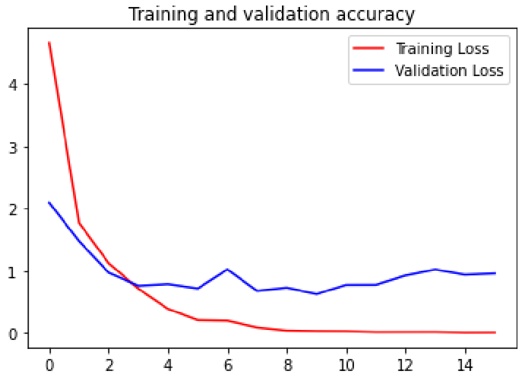


Figure 5: CNN training and validation loss