Multi-class Classification of Mel Spectrograms

Siddharth Mishra, 190841
Department of Electrical Engineering, IIT Kanpur
Machine Learning for Signal Processing, 2022-23 II

Abstract

To solve a multi-class classification of Mel Spectrograms, a patchwork of CNN and Fully connected Neural Networks was used to create an overall model, which achieved an accuracy and F-score of 0.71 and 0.72 respectively.

1 Introduction

The multi-class classification task provided is a subset of the overall task of audio classification. As part of the classification, audio signals are usually converted to Mel Spectrograms, which are Mel-scale representations of the various frequencies in a time-varying signal, while the Mel Scale is an adaptation of the logarithmic scale based on the human perception of frequency. As such, the samples contain significant amount of noise, as well as variations in multiple samples of the same class, all the while retaining the common features that are obvious to human ears. Hence, the task revolves around feature extraction from the Mel Spectrograms and subsequent classification into one of the following ten classes: "Bark", "Meow", "Siren", "Shatter", "Knock", "Crying and sobbing", "Microwave oven", "Vehicle horn and car horn and honking, "Doorbell", "Walk and footsteps".

2 Literature Review

Literature review indicates that various types of problems (such as bird signal [2], environmental noise [3], as well as urban sounds classifications [1]) involving audio classification commonly utilise a conversion of the audio signals to a Mel Spectrograms as an intermediate step. Hence, our task is a subset of a larger audio classification problem. In general, most of the papers using Mel Spectrograms tended to use a variation of a Convolutional Neural Network (CNN), due to their impressive success with regard to visual data. Some papers also combined deep learning (Dense NNs) or Recurrent Neural Networks and LSTMs [4]. Hence, these were the major techniques employed in this assignment.

3 Methodology

3.1 Preprocessing

It was observed that the training set provided consisted of mono-channel numpy arrays with 128 mel frequency bands and a time domain ranging from 80 to 2000 frames (i.e. of the form [1,128,x] where x is the number of frames in the time domain). Since all the Spectrograms need to be brought to a standardised shape, any samples with multiple channels were standardised to mono, and were also padded or cut so they were exactly 500 frames long. This number was chosen based on the distribution of the frame lengths (Fig. 1, such that not too much information was cut out from the training set, while at the same time remaining computationally fast.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

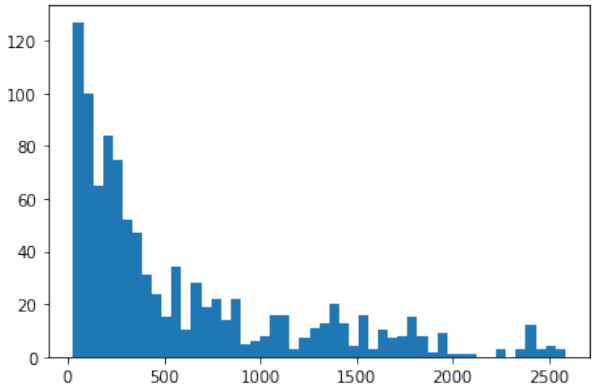

Figure 1: Frequency distribution of Mel Spectrogram lengths

3.2 Dense Neural Network

A vanilla Dense Neural Network was chosen as the first methodology, as the task is heavily based on nonlinear feature recognition, and in general, it is a good idea to try out a Dense NN before other NNs. The major drawback with using a FCNN was that the input of size (1x128x500) would have to be flattened into a single vector, which would make an extremely large network, which is slow, large, and extremely prone to overfitting. Hence, an average pooling along the time dimension was implemented, as well as dropout(p=0.25) in every layer. The hidden layers used were of 128 neurons (HL1) and 32 neurons (HL2) respectively, with ReLU applied after each one. The final output layer was of 10 neurons, and was followed by a softmax layer to complete the classification. Along with a minibatch size of 32, and a learning rate 0.001, the model achieved an accuracy of **0.868**.

3.3 Convolutional Neural Network

The generally accepted method for audio classification is the CNN, which was implemented next. The model summary is provided, with 4 convolutional blocks, followed by a fully connected network and a softmax for the final calculation. Max pooling (size = 2) was carried out in between to prevent the trainable parameters from getting too large. The channels were increased to 16,32, 64 and 128 respectively, while the kernel was of size 3 and stride 1, along with padding.Other hyperparameters were unchanged from the dense NN. The model achieved a significantly higher accuracy of **0.993**

```
if __name__ == "__main__":
    cnn = CNNNetwork()
    summary(cnn, (1, 128, 500))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 130, 502]             160
              ReLU-2         [-1, 16, 130, 502]               0
         MaxPool2d-3          [-1, 16, 65, 251]               0
            Conv2d-4          [-1, 32, 67, 253]           4,640
              ReLU-5          [-1, 32, 67, 253]               0
         MaxPool2d-6          [-1, 32, 33, 126]               0
            Conv2d-7          [-1, 64, 35, 128]          18,496
              ReLU-8          [-1, 64, 35, 128]               0
         MaxPool2d-9           [-1, 64, 17, 64]               0
           Conv2d-10          [-1, 128, 19, 66]          73,856
             ReLU-11          [-1, 128, 19, 66]               0
        MaxPool2d-12           [-1, 128, 9, 33]               0
================================================================
Total params: 97,152
Trainable params: 97,152
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.24
Forward/backward pass size (MB): 34.86
Params size (MB): 0.37
Estimated Total Size (MB): 35.48
----------------------------------------------------------------
```

Figure 2: CNN architecture implemented

4 Observations

The two models were combined to form a patchwork model, by accepting the prediction of the model which had higher confidence. The overall confusion matrics and performance metrics are detailed below.It may be noted that combining the models gave slightly higher values of all performance metrics (Accuracy, F-score, Precision and Recall), but not by a significant amount.

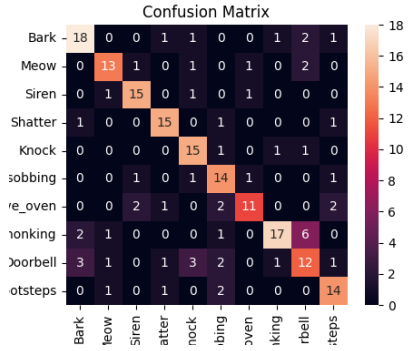

Figure 3: Confusion Matrix of the patchwork architecture

Table 1: Performance Metrics of combined model

Metric	Performance
Accuracy	0.7164179104477612
Precision	0.7241620500915158
Recall	0.7268518518518519
Recall	0.7205790252201705

5 Conclusions

The performance of the overall model was significantly lower than the accuracy achieved on the training set, suggesting strong evidence of overfitting. However, for a combination of rudimentary methods, the model still performed impressively, with above 0.7 in all key metrics. Further improvements could involve the factoring in of long term dependancies, by using LSTMs and RNNs, which is usually seen as the state-of-art solution for such classification problems.

References

[1] Jiuwen Cao, Min Cao, Jianzhong Wang, Chun Yin, Danping Wang, and Pierre-Paul Vidal. Urban noise recognition with convolutional neural network. *Multimedia Tools and Applications*, 78(20):29021–29041, 2019.

[2] Chih-Yuan Koh, Jaw-Yuan Chang, Chiang-Lin Tai, Da-Yo Huang, Han-Hsing Hsieh, and Yi-Wen Liu. Bird sound classification using convolutional neural networks. In *CLEF (Working Notes)*, 2019.

[3] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE, 2015.

[4] Tianhao Qiao, Shunqing Zhang, Zhichao Zhang, Shan Cao, and Shugong Xu. Sub-spectrogram segmentation for environmental sound classification via convolutional recurrent neural network and score level fusion. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 318–323. IEEE, 2019.