Audio classification

Kapil singla
22104405
kapil22@iitk.ac.in

1 Introduction

Sound recognition problem consists of three different stages as pre-processing of signals, extraction of specific features and their classification. In the Signal pre-processing part it divide the input signal to different segments which is further used for extracting related features. After that for reduction of size of data we use feature extraction and represent the complex data as feature vectors. As we are given 1000 spectrograms along with the label so we will use them to train our model and then detect our test sample.

2 Literature Survey

There are various survey papers which have been published recently on the topic of audio classification. For example an article provide a description of the components of an automatic audio classification system, which contains pre-processing modules, feature extraction, training algorithm and evaluation module.The use of deep learning, particularly convolutional networks (CNNs), has lately been used successfully in computer vision and speech recognition.

3 Method

We are doing our code on google colab. So for that we have first store all the data as well as annotation file in the google drive and mount that drive onto the google colab notebook.

Now we have divide our problem statement into 5 parts:

3.1 Pre-processing of Input samples:

- We will read the name of the file name from the annotation.csv file and load the data from the train folder based on that file.

- We have 1000 spectrograms having size (1,128,x) where x is of variable length.
- First we will convert it into 2D format i.e (128,x).
- Now we will make every data of equal length for that we will iterate over all the samples once and find out the maximum of x.
- After that we will do zero padding in all the samples depending on the maximum value.
- We have store this data into a new folder called as train1 where each sample is in 2D form having fixed length
- Now we are finally ready with our sample data.

3.2 Feature extraction:

- Now we have given labels as strings so firstly we have to convert that into integer. For that we have extract the unique labels from the annotation.csv file and then make a dictionary to convert all that labels into an int from 0 to 9.

- After converting labels into int we will update our csv file with the updated values of labels.
- We will load our training data from train 1 folder where we have stored our processed samples and make a new input sample in which we will stack all our processed data using the concept of list.
- Now our input samples have a shape of (1000,128,max).
- Similar thing we will do with the labels we will stack all 1000 labels in a variable
- Now we are just saving this processed data as well as labels into our drive so that we don't have to run again this pre-processed part.

3.3 Defining our CNN Model:

3.3.1 First convolution layer

- Having Filters of 32 with a kernal size of 30X30 using activation layer as Relu with Max pooling=10

3.3.2 Second convolution layer

- Having Filters of 64 with a kernal size of 6X6 using activation layer as Relu with Max pooling=3

3.3.3 First hidden layer

- No. of neurons used are 256 with activation layer as Relu

3.3.4 Second hidden layer

- No. of neurons used are 64 with activation layer as Relu

3.3.5 Output layers

- Output neurons are 10 with activation layer as Softmax

Finally we have used sparse categorical cross entropy with optimizer as adam and a batch size of 32 having 10 epochs

3.4 Splitting training data for testing

- Here we have split our given data for testing.
- Out of 1000 samples we have used 800 samples for training and 200 for testing.

3.5 Evaluating model:

- Now we have use our model for actually testing the test data.
- After predicting the output we have find the argmax of the output and then convert that value into a string
- Now store that labels back into the test.csv file.
- Make classification report which includes precision, recall, F1 score.
- After than we have to find the confusion matrix.
- Then plot all the necessary graphs such as

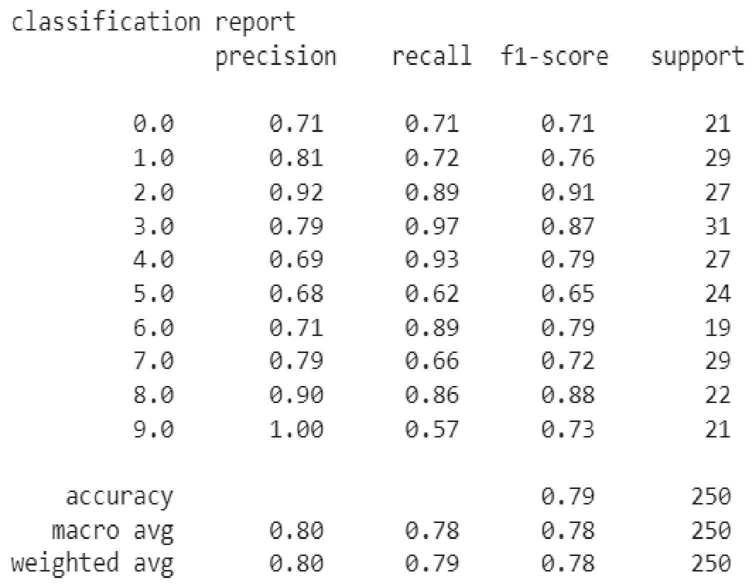
1. Loss vs epochs
2. accuracy vs epochs
3. validation loss
4. validation accuracy

4 Observations

4.1 Testing model by splitting training data

- After splitting the training samples we get our loss, accuracy with a validation loss and accuracy as :

4.1.1 F1 Score



```
classification report
              precision    recall  f1-score   support

         0.0       0.71      0.71      0.71        21
         1.0       0.81      0.72      0.76        29
         2.0       0.92      0.89      0.91        27
         3.0       0.79      0.97      0.87        31
         4.0       0.69      0.93      0.79        27
         5.0       0.68      0.62      0.65        24
         6.0       0.71      0.89      0.79        19
         7.0       0.79      0.66      0.72        29
         8.0       0.90      0.86      0.88        22
         9.0       1.00      0.57      0.73        21

    accuracy                           0.79       250
   macro avg       0.80      0.78      0.78       250
weighted avg       0.80      0.79      0.78       250
```

Figure 1: F1 score

4.1.2 Confusion matrix

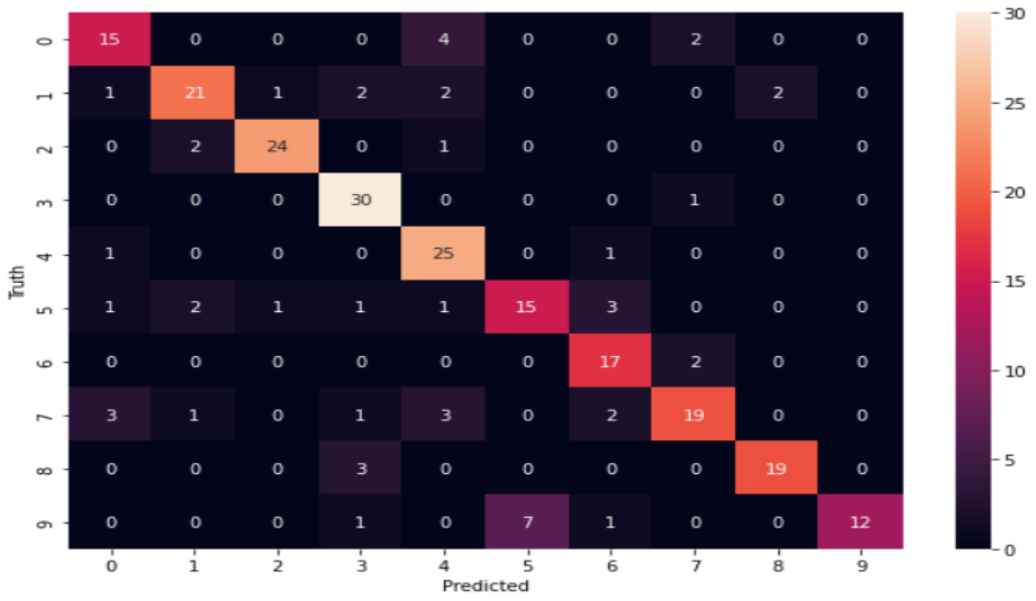


Figure 2: F1 score

4.1.3 Loss

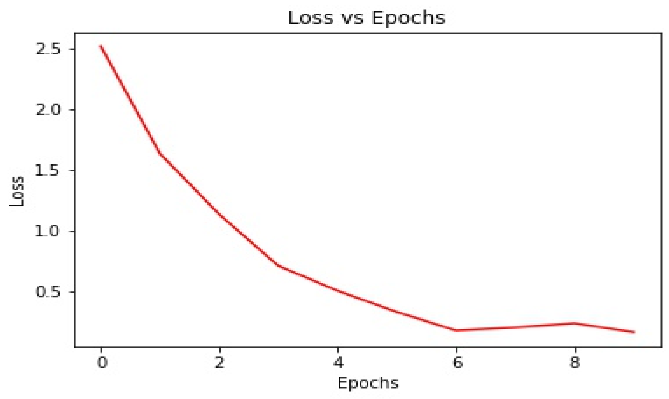


Figure 3: loss

4.1.4 Accuracy

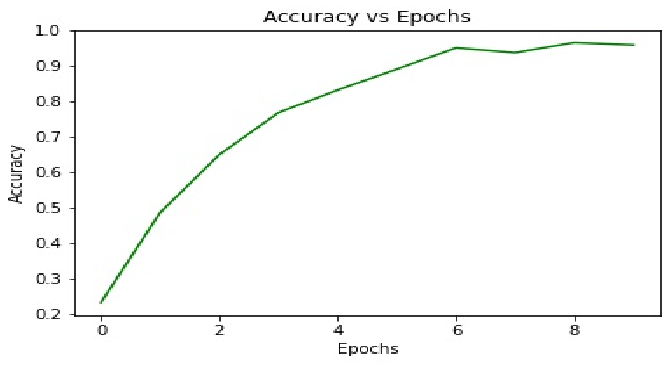


Figure 4: Accuracy

4.1.5 Validation accuracy

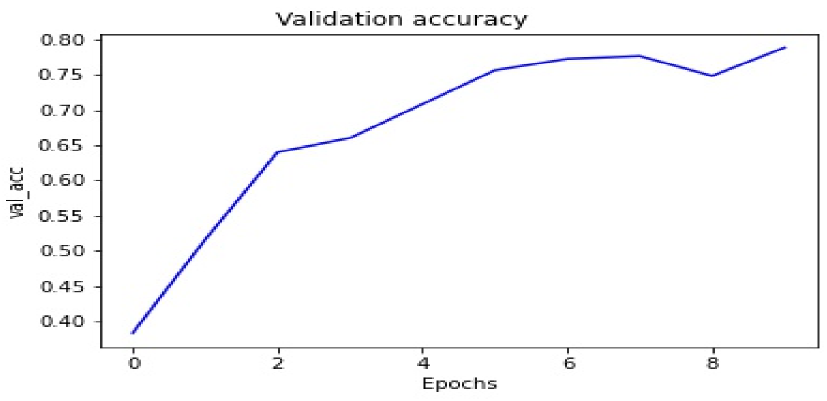


Figure 5: validation accuracy

4.2 Testing model based on *ACTUAL TEST* samples given

- Now based on given test data following are the observations

4.2.1 Loss

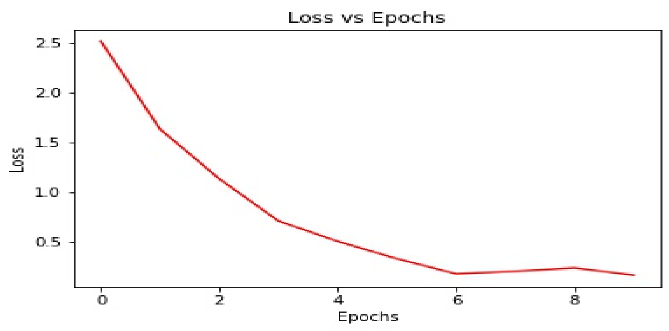


Figure 6: loss

4.2.2 Accuracy

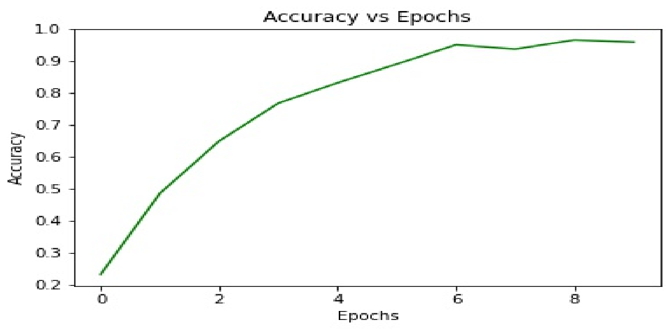
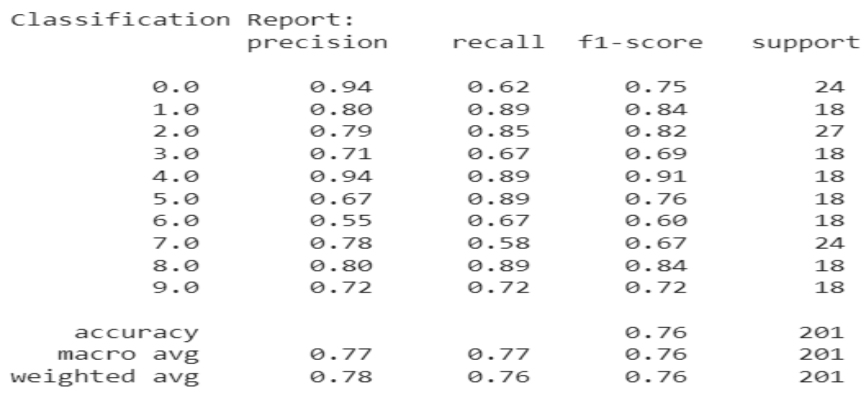


Figure 7: Accuracy

4.2.3 F1 Score



```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.62      0.75        24
         1.0       0.80      0.89      0.84        18
         2.0       0.79      0.85      0.82        27
         3.0       0.71      0.67      0.69        18
         4.0       0.94      0.89      0.91        18
         5.0       0.67      0.89      0.76        18
         6.0       0.55      0.67      0.60        18
         7.0       0.78      0.58      0.67        24
         8.0       0.80      0.89      0.84        18
         9.0       0.72      0.72      0.72        18

    accuracy                           0.76       201
   macro avg       0.77      0.77      0.76       201
weighted avg       0.78      0.76      0.76       201
```

Figure 8: F1 score

4.2.4 Confusion Matrix

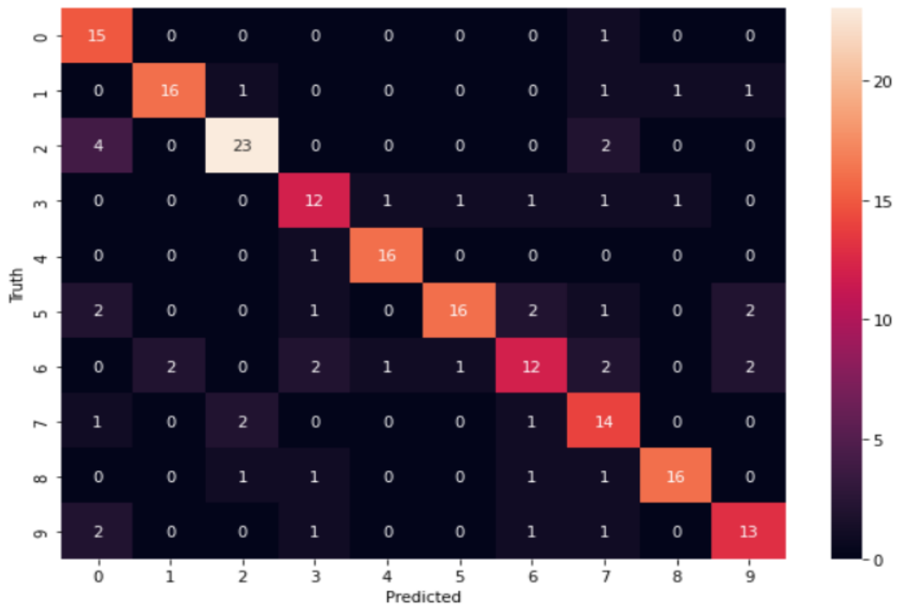


Figure 9: F1 score

5 Discussion

- We can see as epochs are increasing loss is decreasing and accuracy is increasing.
- our model is learning better.
- Some of the observations i have seen as i am changing my kernal size , max polling are:
  1. Validation loss starts increasing, validation accuracy starts decreasing. This means model is cramming values not learning.
  2. Validation loss starts increasing, validation accuracy also increases.This could be case of overfitting or diverse probability values in our cases as we have used softmax in the output layer.
  3. val loss starts decreasing, val acc starts increasing. This is fine as that means model built is learning and working fine.
- Now to decrease overfitting i have used some of the techniques which are as follows:
  1. I have added a dropout layer after the last convolution layer but it didn't give much better result. Reason for this may be our dataset is not so large.
  2. I have reduce the no. of epochs.
  3. I have used early stopping in which I have specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on the validation dataset.

6 References

[1] https://www.ijmsi.org/Papers/Volume.8.Issue.6/C08062428.pdf

[2] https://www.youtube.com/watch?v=7HPwo4wnJeAlist=PLeo1K3hjS3uu7CxAacxVndI4bE$_o 3BDtOindex = 24t = 61s$

[3] https://www.youtube.com/watch?v=zfiSAzpy9NMlist=PLeo1K3hjS3uu7CxAacxVndI4bE$_o 3BDtOindex = 23$