NNTI Project Report

Harsh Agarwal - Akhil Juneja - Salih Talha Akgün
Saarland university
66123 Saarbrücken

Abstract

In this project we have worked on models for spoken digit recognition. In the first task, we defined the baseline model with Multi-Layer Perceptron and Stochastic Gradient Descent Linear Model. We built more complex RNN and CNN models in the second task. In the third and last task, we worked on developing a model that is trained with voices from only one person.

1 Exploring the Dataset

We investigated the data and saw that most recordings have close lengths. Also, most data does not have empty spaces after and before the actual voice. This means we don't have to do a clipping or trimming.

2 Task I

In this task, for baseline model we have implemented a SGD model with hinge loss (Weston et al., 1999) and L2 loss penalty and trained it on the downsampled dataset.

Also, in this task we have implemented an additional Multi Layer Perceptron (MLP) Classifier model to baseline the Neural Network model architecture. with layer sizes equal to (100,1000,100) and trained it with downsampled dataset.

2.1 Downsampling Spectogram

The function accepts N as a parameter for how many splits to create along the frequency axis, and using mean pooling for downsampling. Instead of simply using a iterative loop for computing and applying mean pooling along the frequency axis, we treat it as a strided pooling problem. This is done using 'block_reduce' from skimage library to downsample recordings.

For samples with number of spectral vectors T less than N, we have implemented both mean and zero padding to make number of spectral vectors the same as N.

2.2 Confusion Matrix Analysis

For identifying the best value of parameter N for the downsampling function we see that most values of N gave similar results and went ahead with a value of 21. Similarly, we also tried mean and zero padding for audio signals with less than 21 spectral vectors, there also we did not see any noticeable change in accuracy.

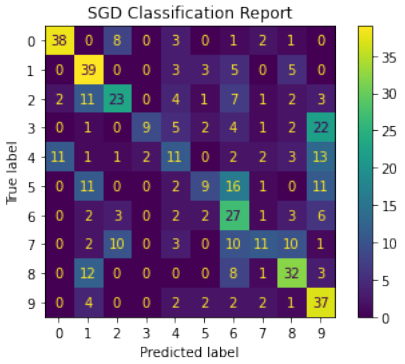


Figure 1: Confusion Matrix for SGD Classifier

We train the baseline **SGD Classifier** for 10 iterations. Using this we get around 43.7% accuracy on the test set. Also, looking at the train set accuracy we see it has already reached 88.6%.

Looking at individual class precision we see class 5 gets perfect precision. This is also visible from the confusion matrix in Figure 1 we see there isn't a case where we predict 5 and it is wrong. But this does not mean overall 5 is always predicted perfectly. As 5 has a rather bad recall value. This means samples with label 5 are often predicted badly. We don't see an abnormally high recall score with a very low precision value.

Overall, looking at the F1 score we see class 0 and 1 give the best performance.

As mentioned we also trained a simple **MLP Classifier**. Multiple configurations were tried and we got the best accuracy for 3 layer network with the number of neurons as (100, 1000, 100). Using

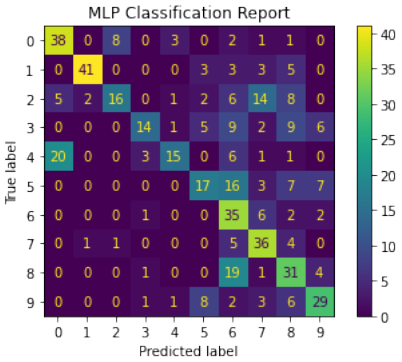


Figure 2: Confusion Matrix for MLP Classsfier

this MLP Classifier we get around 54% accuracy on the test set. Also, looking at the train set accuracy we see it has hugely overfitted and got 100%. overfitting was similar for all the configurations. Looking at the confusion matrix in Figure 2 we do not see any abnormally high precision or recall values.

Overall, looking at the F1 score we see classes 0, 1, and 9 give the best performance.

3 Task II

In this task instead of using the downsampled audio signals as we did in Task 1, here we will use acoustic signals with arbitrary lengths. Here we build a model which accepts arbitrary length input and we saw how it compares to our baseline SGD and MLP Classifier model. We will evaluate these models against the baseline model in terms of accuracy and confusion matrix. We will also discuss the effect of hyperparameters on the model performance.

3.1 RNN

We first start by creating a RNN model to improve our classifier on arbitrary sized melspectrograms. We use LSTM model (Hochreiter and Schmidhuber, 1997) here with Fully Colvolutional Layer for final classification. This model is trained for 30 epochs with Adam optimizer.

Figure 3 shows Confusion Matrix for RNN Classifier. It gets around 59.2% accuracy on the test set. Also, looking at the train set accuracy we see it has hugely overfitted and got 98.2%. We see this an increase from the previous baseline and an increase from even the MLP baseline model we created.

Looking at individual class precision we see class 4 gets a perfect precision. There isn't a case where we predict 4 and it is wrong prediction. But this does not mean overall 4 is always predicted perfectly. As 4 has a rather bad recall value. This means samples with label 4 are often predicted badly. Further, looking at individual class recall we see class 7 gets the highest precision. We see samples belonging to class 7 are predicted correctly. But 7 has a rather bad precision value. This means many samples with different label values are often predicted as belonging to class 7. Overall, looking at the F1 score we see class 0, 1, 3, 4, 7, 8 and 9 give a good and comparable performance.

We tried tweaking the hyperparameters: learning rate and hidden layer dimensions in LSTM. Increasing learning rate from 0.001, makes the training unstable with lr=0.1 giving NaN loss values. Similarly decreasing the value further makes the training very slow. Similarly increasing or decreasing the hidden dimensions for LSTM layer it reduces the overall accuracy.

3.2 CNN

Next trying to improve the model with implement a 1D Convolutional Classifier (Kiranyaz et al., 2021). We stack 4 Convolutional - Relu layers together, followed by a Fully Connected layer. The model is

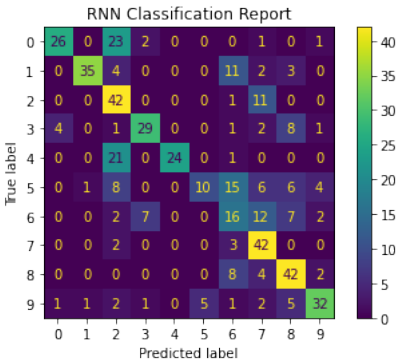


Figure 3: Confusion Matrix for RNN Classsfier

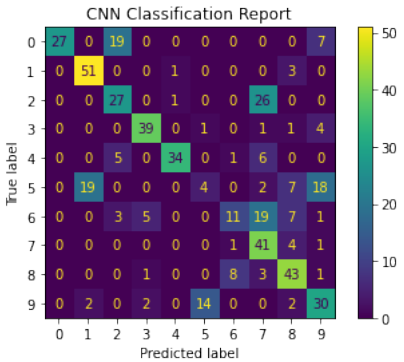
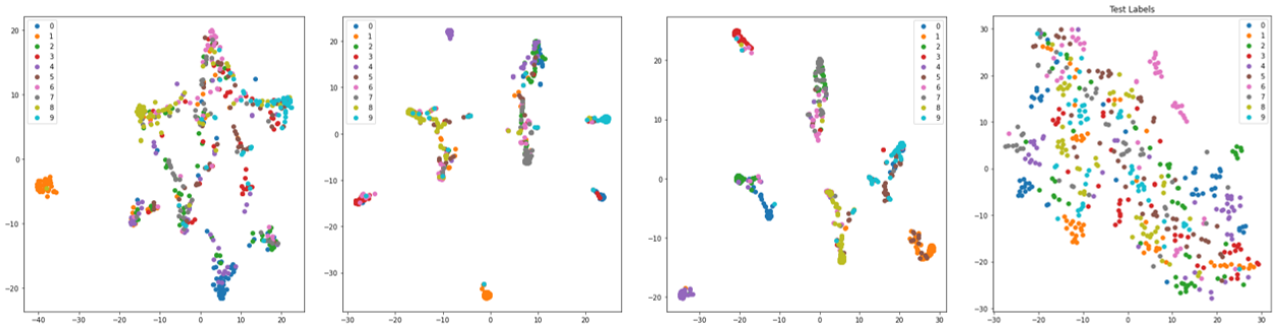


Figure 4: Confusion Matrix for CNN Classsfier



(a) SGD Predictions (b) RNN Predictions (c) CNN Predictions (d) Input Data

Figure 5: TSNE plots for labels and predictions from various models

again trained with Adam opitmiser for 30 epochs. Here we get around 61% accuracy on the test set. Also, looking at the train set accuracy we see that the overfitting has reduces a lot and we got 83.4% accuracy. We see this further small increase from the previous RNN model we created.

Checking Figure 4 for individual class precision we see class 1 gets a perfect precision. We see there isn't a case where we predict 1 and it is wrong prediction. But this does not mean overall 1 is always predicted perfectly. As 1 has a bad recall value. This means samples with label 1 are often predicted badly. Further Looking at individual class recall we see class 7 again gets a very high precision. But again this does not mean overall 7 is always predicted perfectly. As 7 has a rather bad precision value. This means many samples with different label values are often predicted as belonging to class 7. Overall, looking at the F1 score we see class 0, 1, 3, 4, and 8 give a good and comparable performance.

Again we tried to further tweak the model. We tried introducing Dropout and Batch Normalisation as a regularizer, but that did not help either. Furthermore, we tried changing the number of filters for convolutional layers and the network with best configuration is chosen.

3.3 Dimensionality Reduction with TSNE

Now we try to visualize our results corresponding to ground truth. We use the outputs for the networks from the last hidden layer, for both CNN and RNN. These feature maps are then passed through the dimensionality reducer t-SNE (Van der Maaten and Hinton, 2008) for visualisation on a 2D graph.

Figure 5 shows t-SNE applied to various models.

- SGD predictions [Figure 5a]: We see the distribution is not very well separated.
- RNN predictions [Figure 5b]: The distribution is much better separated than SGD here.
- 1D CNN predictions [Figure 5b]: Again similar to RNN, even for CNNs the predictions are well separated.
- Input Data [Figure 5d]: Shows the trend for input test dataset.

We see from the plot in Figure 6 that around 50% samples are classified correctly by both the models, while around 30% samples are classified incorrectly by both the samples. This leaves around 20% samples which are misclassified by only one of these models.

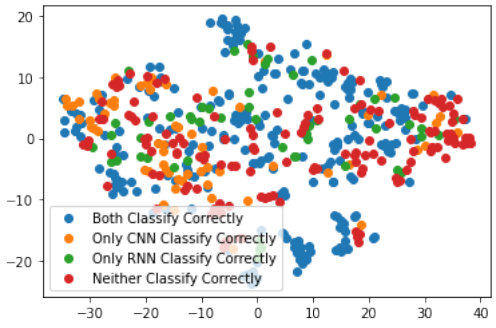


Figure 6: Confusion Matrix for RNN Classsfier

3.4 Statistical Significance

We ran 10,000 iterations, instead of 1,000,000 iterations as suggested for Bootstrap Statistics **??**, due to limited computational capabilities. Though we got a $p - value = 0.0$ but we still cannot be sure if this results are completely trustable. As we do not come across any combination of test dataset which has an accuracy of over 0.75 for RNNs (given accuracy of 0.43 for baseline model and 0.59 for RNN model)

4 Task III

In this task we aim at using training data from a single speaker, and using techniques to improve performance on complete test dataset.

4.1 RNN Model with Single Speaker

Using RNN Classifier we get around 46.9% accuracy on test set and a high overfitting with 97.6% accuracy on train set. This is reduced from the previous model we created in Task II, as same model is unable to learn the feature mapping and patterns with limited amount of training data. Hence it is also not able to generalize well to unseen data. Class 0, 1 and 4 give a good performance.

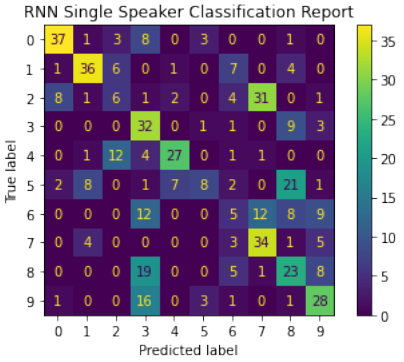


Figure 7: Confusion Matrix for Single Speaker

4.2 RNN Model with Single Speaker and Spec augmentation

4.2.1 Feature Masking

We used Spec augmentation (Park et al., 2019) technique called frequency masking on mel spectogram. In frequency masking, randomly chosen band of frequencies are masked with the value of zero. After using this augmentation, we can observe test error to be 44.5%. If we have to improve the performance then we have to make wider and deeper networks and train with longer schedules. By looking at confusion matrix, we can say that label 0 and label 1 perform better than others.

4.2.2 Time Masking

We used Spec augmentation technique called time masking on mel spectogram. In time masking, randomly chosen band of slice of time steps are masked with the value of zero. After using this augmentation technique, we can observe test error to be 45.9%. If we have to improve the performance then we have to make wider and deeper networks and train with longer schedules. By looking at confusion matrix, we can say that label 0, label 1 and label 7 perform better than others.

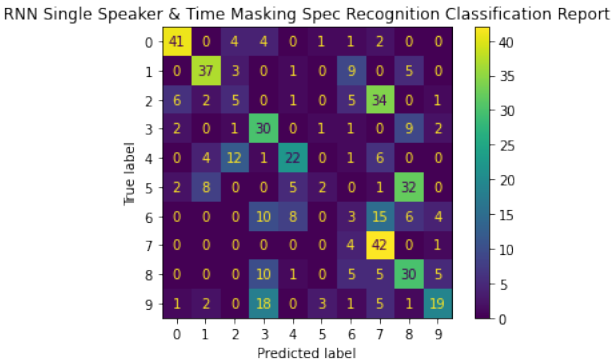


Figure 9: Confusion Matrix for Time Masking

4.2.3 Wave Augmentation

We use Wavaugment (Kharitonov et al., 2021) on raw waveform. In the dataloader we randomly apply one of the augmentation techniques which are: Pitch increase, Pitch reduction, adding Reverb and adding Additive Noise. Using this we get an accuracy of 46.7% on the test set. Looking at the confusion matrix in Figure 10, the performance on Label 5 is particularly bad.

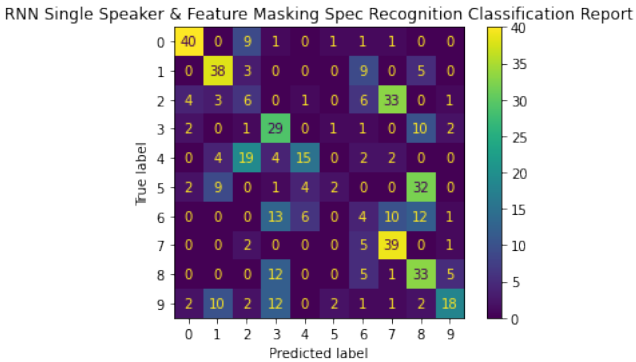


Figure 8: Confusion Matrix for Frequency Masking

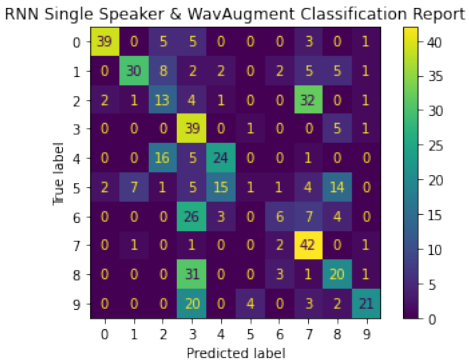


Figure 10: Confusion Matrix for WavAugment

References

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eugene Kharitonov, Morgane Rivière, Gabriel Synnaeve, Lior Wolf, Pierre-Emmanuel Mazaré, Matthijs Douze, and Emmanuel Dupoux. 2021. Data augmenting contrastive learning of speech representations in the time domain. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 215–222. IEEE.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 2021. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Jason Weston, Chris Watkins, et al. 1999. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224.