Speaker-Independent Spoken Digit Recognition

Divya Mani Adhikari (7043090) and **Zurana Mehrin Ruhi (7023892)** and **Prashanth Pombala (7039928)**
Saarland University, Saarbruecken

Abstract

In this project, we developed an SDR system in a speaker-independent setting that can generalize to different speakers in real-world ASR systems. We performed a sequence classification task using two different neural networks, a simple RNN and a transformer model. Given a short audio clip as input, the respective model classifies the digit that was spoken. First, the given dataset was explored using MelSpectrograms and analyzed. Later, the raw data was fed to the neural network models to further improve inference results. We also explored two augmentation techniques SpecAugment and Wavaugment to emphasize on single-speaker generalization for global classification.

1 Introduction

In recent times there have been growing instances of numbers being used ubiquitously for identification purposes. Account numbers are used in banks in banks for identification purposes, Customer identification numbers are used in customer services to provide various services and social security numbers are used by various government services to name a few applications. Having people manually hear these numbers and verify them can be both time-consuming and prone to errors. Nowadays due to the recent advances in Machine learning, deep learning models can be used to recognize spoken digits. This is also aided by the rapid improvements in the field of Automatic speech recognition.

In this project, we have implemented a Spoken digit recognition system that takes as input a short audio clip of a digit being spoken and gives as output the digit. In the first part of the project, Downsampled Mel Spectrograms were created from the dataset and a linear baseline model was established. Next, two deep learning models: a simple RNN and a conformer model were trained both of which achieved considerably better results than the linear classifier. Furthermore, a dimensionality reduction algorithm, t-SNE was applied in order to visualize the correctness of the prediction decisions of all the models. To measure the statistical significance of the same, the bootstrap technique was used. Lastly, the viability of implementing these models for the case where training data is available from only a single speaker was thoroughly explored using augmentation techniques along with contrastive learning settings

2 Dataset

The dataset used for this project consists of short audio of people speaking digits. The speech samples were already divided into training, development, and test samples. The acoustic realization of a speech segment can be viewed as a time-variant waveform $S \in R^n$. Here, n depends on both the duration of the speech segment and the sampling rate of the continuous speech signal. In the given dataset, the temporal duration of the audio is 428.5 milliseconds. For digits in the range 0-9, the duration of the speech segment should be around 0.5 seconds with reasonable variation depending on speech rate.

2.1 The Speech Signal Representation - Mel Spectrograms

In the spectro-temporal representation of speech, a speech sample can be seen as a sequence of T spectral vectors as $X = x^1, x^2, .., x^T$. Each spectral vector $x^t \in R^k$ at time-step t is extracted from a short speech segment (25 milliseconds) with the assumption that the signal is time-invariant in this small time window. Here, k is the number of frequency bands in the spectrogram and this is a parameter of the feature extraction pipeline. The representation is based on the Fourier transform to convert the temporal signal into the frequency domain. In this work, we fixed k = 13.

3 Baseline model: SVM

Before implementing the Baseline model we first downsampled the spectrogram. As T varies in different speech samples, For our Baseline model we first preprocessed the data to create equal-sized sample representations where $X' \in R^{Nxk}$; N is the number of splits of S across the time-axis.

We used a Linear SGD-based Classifier with hinge loss, which is an SVM model authored by Hearst et al. (1998).

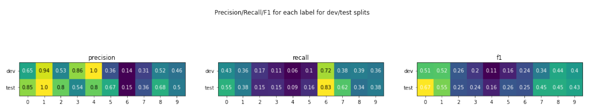

Figure 1: Precision/Recall/F1 for each label for dev/test splits

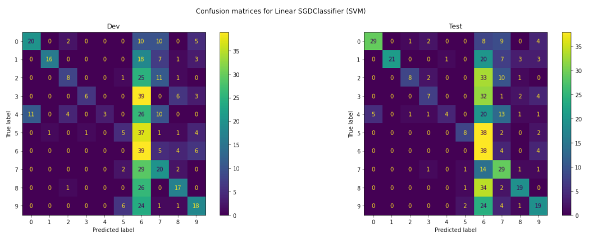

Figure 2: Confusion matrices for Linear SGDClassifier (SVM)

Using this model, we were able to obtain 94% accuracy on the training set. But, this didn't seem to generalize well as our dev set accuracy was only 31% and our test set accuracy was only 36%. The confusion matrix in figure 2 for the dev and test set shows that the linear SVM model seems to confuse most of the digits with the digits 6 and 7 as seen by the high number of predictions of 6 and 7.

4 Deep Learning models

To make the best use of all the features in our audio dataset, we applied Deep Learning models where we could feed the varied-length training samples without downsampling or adding any padding techniques that may introduce model bias. Out of the models we tried, we present RNN and the conformer model here.

4.1 RNN

We used a simple RNN architecture that uses a single LSTM layer and a linear layer followed by softmax-based output. In this case, each input wave sample was embedded to the size of 80. The model was trained for 30 epochs with early stopping (patience=5) to select the best model without exhausting resources. By just using a simple RNN model, the test accuracy is improved by 42.5% than that of the baseline as shown in table 1

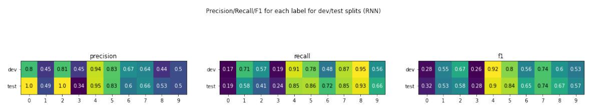

Figure 3: Precision/Recall/F1 for each label for RNN

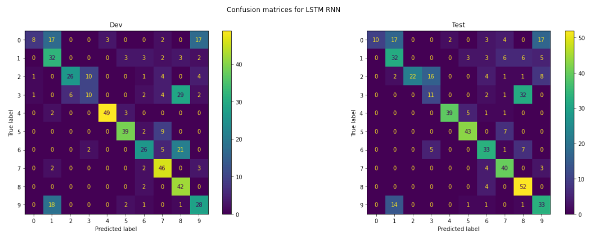

Figure 4: Confusion matrices for RNN

Figure 3 shows that the RNN model performs better than the baseline for each of the precision, recall, and F1 scores. The digits 0 and 3 have somewhat low precision and recall and consequently, low F1 scores.

The confusion matrix in figure 4 between the true labels and the predictions also reflects the same. The high confusion rate for digit 6 has been almost eliminated. However, the numbers of misclassified digits 3 and 8 for both the dev and test set show that the model still fails to generalize a considerable amount for these cases.

4.2 Conformer

4.2.1 Methodology

The Conformer model introduced by Gulati et al. (2020) is an amalgam of Convolution Neural networks and Transformers. This is based on the intuition that Transformer models can handle long-term dependencies well and Convolution models can handle local information well. Conformer was originally used in the case of speech recognition i.e. predicting the spoken text. The conformer block contains the self-attention module multi-headed self-attention module integrated with a relative sinusoidal positional encoding scheme; this module helps learn multiple long-term dependencies in a given sequence. In our implementation, we have adapted this model for the case of Classification.

4.2.2 Results

The conformer model was trained similarly to RNN with early stopping. The final test set accuracy stands at about 81% which is a great improvement over the baseline.

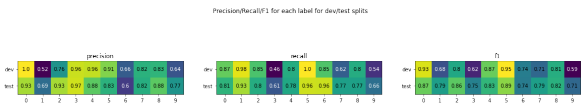

Figure 5: Precision/Recall/F1 for each label for Conformer

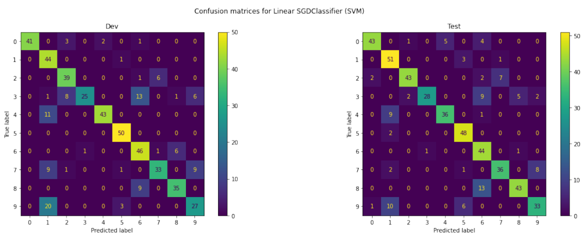

Figure 6: Confusion matrices for Conformer

With Conformer, the results seem much better with higher precision, recall and F1 scores in figure 5 Digits 3 and 9 show lower F1 scores due to their low recall scores.

However, the Confusion matrix for the dev set seems to be more chaotic than in the RNN model. Most of the examples have been classified well but there is quite a lot of confusion such as between digits 9 and 1. In contrast, the test set appears to perform better than the dev set. Still, 9 is mostly misclassified as 1 and 8 as 6 compared to misclassifications of other digits as seen in the right side of the fig 6

5 Dimensionality reduction

For dimensionality reduction, we used a t-SNE algorithm to reduce the dimensions of the spectrogram so that we can plot the data and see how our classification algorithms classify those points. We reduce the dimension of the spectrogram to 2 components of t-SNE which will act as the axes in our 2D plot. Figure 7 and 8 respectively show the t-SNE plots for the RNN and the Conformer model for both the dev and test set. As we can see in the plots, there don't seem to be definite clusters (based on the components generated by t-SNE) of each digit so it is a bit difficult to clearly see how accurate the classifications are from the plot. Nevertheless, we can still compare regions to get an idea of the classification accuracy or the decision boundaries.

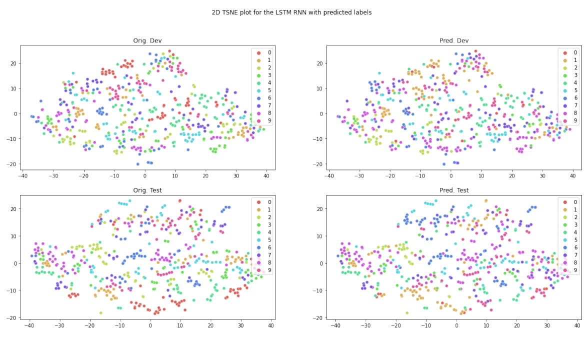

Figure 7: t-SNE analysis of RNN classifier

When plotted for RNN, we can see that the classification seems to be better with fewer misclassifications as we don't see any one digit dominating the predictions. Still, there seems to be confusion between the digits 0 and 1 particularly in the dev set. In the case of the test set, the misclassification seems to be more among 3 and 8.

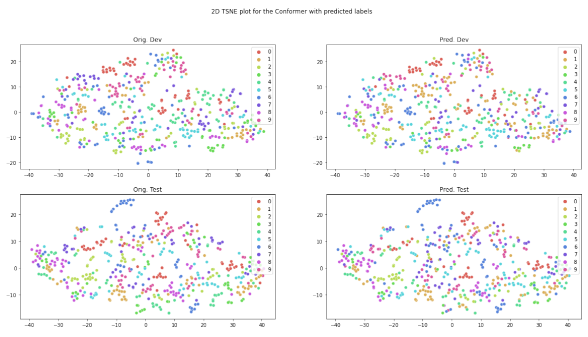

Figure 8: t-SNE analysis of Conformer

The plot is similar to that of the RNN with fewer misclassification errors. The Conformer model was much more accurate than the other two so both the original plot and the predicted model for Conformer seem similar with fewer errors. As we've discussed before in the Confusion matrix section, we can see the confusion between 9 and 1 and 8 and 6 in the plots for the dev and test set above.

6 Measuring Statistical Significance

To measure the statistical significance of our predictions and their accuracy on the test set the bootstrap approach was used. In Bootstrap we randomly sample n examples from the test set and repeatedly evaluate our model on this bootstrapped set. If the p-value is < 0.05, we can be 95 % sure that the improvement in accuracy due to our neural models is not due to chance. We calculated the average accuracy for 2500 iterations for the Conformer model. While this number of iterations is not enough to concretely say that the improvement is statistically significant, the standard error is zero and the average accuracy is around 80 % (close to the test accuracy of the Conformer model) which suggests that the improvement might be statistically significant.

7 Experimentation on Single-speaker Data

In this section, we discuss the results from training the conformer model using only the data of one speaker. Upon training the Conformer model on one of the speakers (George), the test accuracy dropped to 26% as the training set only had 500 examples to train from and the evaluations were made on the original dev and test set. This is expected as the model is trained on a very small dataset with fewer feature variations to learn from; thus the highly flexible creates a lot of bias and consecutively performs poorly on test sets. To tackle this issue and bring about more variation in the train dataset without introducing more speakers, we explored two different augmentation strategies.

7.1 Data Augmentation

The first Data Augmentation technique used was SpecAugment introduced by Park et al. (2019) as an approach to data augmentation for Audio. This approach augments the mel spectrogram directly. This is applied to George's audio data to generate additional 500 examples totalling 1000 training examples. We trained the conformer model with this augmented data using the same methodology mentioned as before. Secondly, We also used WavAugment introduced by Kharitonov et al. (2020) which directly augments the raw waveform. Thus, in this case, we can use the raw waveforms without extracting the Mel-frequency cepstral coefficients from them. The total number of training data and training process is the same as that of SpecAugment. The results are summarized in table 1.

7.2 Contrastive Loss Learning

In our case, SpecAugment was found to have performed better than WavAugment, hence contrastive loss learning is implemented with this augmentation technique. In a supervised learning setting, the contrastive loss is added to the cross-entropy loss and then backpropagated. Although Kharitonov et al. (2020) showed that contrastive learning improves the WavAugment technique's results, we did not find the same case for the SpecAugment technique. The previous model with simple cross-entropy loss performed better by 19.74% than the one combined with contrastive loss.

Whole Dataset	**Accuracy**
SVM	36%
RNN	62.62%
Conformer	**80.52%**
Single-speaker Dataset	
Conformer	27.63%
SpecAugment+Conformer	42.35%
WavAugment+Conformer	32.41%
SpecAugment+Conformer+CL	33.99%

Table 1: Test accuracy for different models

Limitations

Due to computational constraints on Google Colab, we couldn't run the bootstrap evaluation to a sufficiently large number of iterations. Although the recommended number of iterations was 10^6 iterations, we could only run up to 2500 iterations.

References

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. pages 5036–5040.

M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Eugene Kharitonov, Morgane Rivière, Gabriel Synnaeve, Lior Wolf, Pierre-Emmanuel Mazaré, Matthijs Douze, and Emmanuel Dupoux. 2020. Data augmenting contrastive learning of speech representations in the time domain.

Daniel Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Cubuk, and Quoc Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. pages 2613–2617.