CSC413 Final Project - Lie Detection on CSC dataset

Yuan Dou
Department of Computer Science
University of Toronto
yuan.dou@mail.utoronto.ca

Yezheng Shao
Department of Computer Science
University of Toronto
yezheng.shao@mail.utoronto.ca

Yulin Wang
Department of Computer Science
University of Toronto
yulin.wang@mail.utoronto.ca

Abstract

In this project, we studied the problem of lie detection on the CSC deceptive speech dataset and improved an existing Complex Audio RNN model to conduct the automatic truth-and-lie detection on the labelled acoustic sequences from the CSC Deceptive Speech Dataset [2]. We leveraged U-Net and autoencoder architectures with LSTM. The experimental results show that we obtain a significant improvement of about 15% in test accuracy compared to the original model.

1 Introduction

Detecting lies is a challenging and necessary task with broad implications in many high-stakes situations, including criminal investigations, court decisions, and military situations. The paper Neural Lie Detection with the CSC Deceptive Speech Dataset proposed four different RNN algorithms to classify a speech segment as truthful or deceptive [3]. It shows that the RNN models with lexical features encoded have better performance than those taking only the speaker-dependent acoustic features as input. Therefore, we suspected there could be a breakthrough for the model with acoustic input only. Inspired by the U-Net architecture, we improved one of the RNN algorithms called the Complex Audio RNN model, feeding acoustic input to an LSTM followed by a dense layer, batchnorm, ReLU, and classification layer.

The CSC Deceptive Speech consists of 32 hours of audio interviews from 32 native speakers of Standard American English (16 male,16 female) recruited from the Columbia University student population and the community [2]. Our training examples consist of short speech recordings (less than 3 minutes) of the subject answering questions posed by the interviewer. In total, we spliced approximately transcribed audio for 32 speakers with roughly 10 recordings each, which resulted in about 320 training examples total. Each of the utterances has been labelled as either truthful or deceitful. And the Mel-frequency cepstral coefficients (MFCCs) were obtained using the python speech features module using default parameters, i.e., 25 ms windows, 13 cepstral coefficients, and 512 fast Fourier transform coefficients.

2 Related Work

Speech-based lie detection has been attempted since the 1970s, but several studies have investigated recording-based lie detection algorithms and found their efficacy to be questionable. More specifically for our dataset, a few studies have used traditional machine learning techniques on the CSC Deceptive Speech dataset. Hirschberg et al. [4] used a combination of lexical and acoustic features and a

Preprint. Under review.

speaker-dependent feature set, including gender, the ratio of laughter to a conversation, cue sentence frequency, and filler pause frequency. They achieved an accuracy level of 66.4% in identifying examples of deceptive speech. However, it is not clear from their error analysis which factors are most important to their model. Benus et al. [1] examined filled and silent pauses and their characteristics as cues to deception in a new corpus of deceptive and non-deceptive speech. Their training data shows that in general, the use of pauses correlates more with truthful than with deceptive speech. This was the case for both silent and vocalized pauses. They achieved an accuracy level of 67.8% in deceptive speech detection, which is slightly better than the previous performance, but there is still a potential improvement.

3 Methods

3.1 Intuition

The original Complex Audio RNN model takes the Speaker-dependent acoustic features as the input to an LSTM, then flatten the output and goes through two decreasing-size Fully Connected-BatchNorm-ReLU(FC-BN-ReLU) blocks with a dropout layer in between and a final classification layer [3].

Figure 1: Complex Audio RNN

Our method is inspired by the architecture of an autoencoder. Instead of two consecutive decreasing size blocks, we added an increasing size block after the first decreasing block. In this way, the model will be able to extract the most salient features and learn a non-linear generalization of PCA [5].

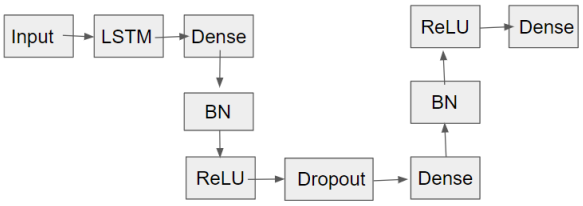

Figure 2: Improved Complex Audio RNN

3.2 Details

The input to the LSTM cell is an MFCC coefficient feature that has the dimension of $N \times C$, where N is the number of frames and C is the number of MFCC coefficients. The feature matrix is obtained by extracting features from audio files, and usually, the number of the coefficient is 13. We use the PyTorch single-direction LSTM model for our RNN cell. As The first dense layer works as an encoder. It has the dimension of hidden_size $\times$ hidden_size // 2. The second dense layer works as the decoder. It has the dimension of hidden_size // 2 $\times$ hidden_size. In the bottleneck between the encoder and decoder, we apply a dropout layer with a probability of 0.5. The final dense layer is the classification layer that has a dimension of hidden_size // 2 $\times$ 2. Finally, We used the cross-entropy loss function to train our model.

4 Experiments and discussions

4.1 Experiments

4.1.1 GRU or LSTM

In the beginning, we were thinking about using a GRU layer rather than an LSTM layer. Since we do not have access to the full dataset, and generally GRU performs better than LSTM on a small dataset, GRU also trains faster and uses less resources.

What we did was first replace the LSTM layer using the GRU layer, then we tried many different sets of parameters for both models and also tried to find the optimal sets. After we did some experiments and looked at the result for both models, we found that the original model using LSTM performs better than the model using GRU both in accuracy and consistency. Our explanation is that, for the problem of Lie Detection, we are dealing with a very large sequence of data and classify it at the end. Then long-term memory would be very important, which might explain why the LSTM layer outperformed GRU Layer.

4.1.2 Experiment on the parameters of LSTM

After we made a conclusion that the LSTM layer is a better choice for this particular problem, we were looking for the best set of parameters while investigating what each parameter means and how to adjust parameters based on different results.

- `input_size`: Fixed base on input data
- `hidden_size`: Anything larger than 13 doesn't make sense, generally 5-10 is a good size.
- `num_layers`: Stacked LSTM is not really needed for this problem and will take longer computation without increasing accuracy.
- `bias`: Makes the layer use bias weights. Default is True.
- `bidirectional`: Bidirectional LSTM is not needed here, also more computation same accuracy.

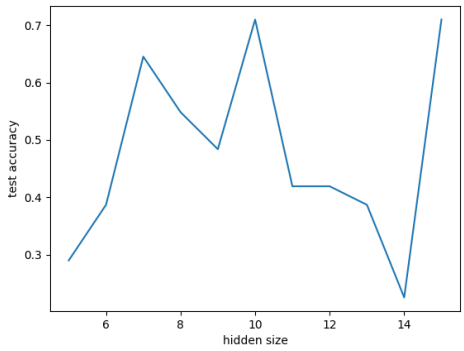

Figure 3: test accuracy vs hidden layer size

Figure 3 shows that when the hidden size is between 7 and 10, the test accuracy is above average. As the hidden size exceed 10, the test accuracy drops rapidly. Notice the surge at the end of the curve, we suspect this is caused by the curse of the autoencoder. It simply learns to perform the copying task and does not extract useful information about the distribution of the data [5].

4.1.3 Experiment on decoder layers

To evaluate the effect of decoder layers, we train our model with 1 layer decoder and 2 layer decoder respectively, and compare the test accuracy. Due to the limited access to the data set, we run the model with different seeds and the same hyper-parameters to create a more comprehensive result.

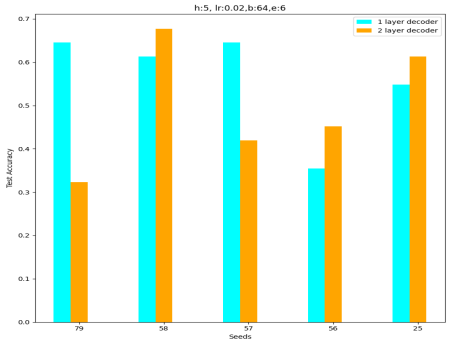

Figure 4: test accuracy vs different seeds

As Figure 4 shows, 2 of the 5 seeds have a significant drop in test accuracy and the other 3 have a slight improvement. Therefore, there is no clear indication that a deeper decoder would improve the model.

4.1.4 Experiment on general parameters

- `num_epochs`: We tried many different values, 4-6 is enough. Higher than that may cause over fitting, lower than that will make the model under trained.
- `learning_rate`: We tried many different values, 0.01-0.05 is enough.
- `batch_size`: Many values could achieve a really nice result, it is not that important.
- `optimizer`: We were using Adam for our model.

4.2 Discussion

After we optimized the parameter for the LSTM layer and fine-tune the hyper-parameters, we see a significant improvement in the test accuracy of 15% compared to the original Complex Audio RNN.

Model	Dataset	% Accuarcy
Complex Audio RNN	Validation	61.24
Complex Audio RNN	Test	62.22
Improved RNN	Validation	65.1
Improved RNN	Test	77.4

However, during our experiment, we found that our model is unstable. We suspect it is due to the randomness in data-splitting. Since we only have access to a limited amount of CSC deceptive speech data, the data can easily be not well-split and the training data can be not comprehensive. As a result, the test set could easily be biased in favor of the model. Therefore, the evidence of improvement is not concrete. However, we see the potential of the structure of LSTM followed by a decoder. We believe that with full access to the data, we can further improve our model and make a more generalized conclusion.

5 Summary

In this project, we investigated the Lie Detection problem on the CSC deceptive speech data set. We focused on the Complex Audio RNN model and experimented with many different architectures and different sets of parameters. In the end, due to the limited data size, we could not make a perfectly generalized model. However, we come to a conclusion that the structure of LSTM followed by a complex decoder layer has great potential for Lie Detection problems.

References

[1] Benus, Stefan, et al. (2006). ”Pauses in deceptive speech.” Speech Prosody. Vol. 18. 2006.

[2] Columbia University, SRI International, and University of Colorado Boulder. (2013). SC Deceptive Speech LDC2013S09. Web Download. Philadelphia: Linguistic Data Consortium.

[3] Desai, S., Siegelman, M., & Maurer, Z.D. (2017). Neural Lie Detection with the CSC Deceptive Speech Dataset.

[4] Hirschberg, Julia, et al. (2005). ”Distinguishing de-ceptive from non-deceptive speech.” Interspeech.

[5] Ian, G., Yoshua B., & Aaron, C. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.