Lip2Text:lip reading from silent video

Wei Zheng, 52684610; Xiaoan Sun, 79879938

The University of British Columbia

April 17, 2021

Abstract

When speech is not present or corrupted by external noise, humans will involuntarily infer a partial transition from the movement of the lips. In this project, we refer to LipNet —— a video-to-text model based on the GRID dataset that consists of spatiotemporal convolutional network, recurrent network, and the connectionist temporal classification loss, and implement it to map variable-length sequences to texts for realizing speechreading. We also adjust the original model structure from GRU to LSTM and apply the residual connections, and the experiment results indicate that the adjustment from GRU to LSTM may provide better results. We then discover the appropriate hyperparameters by fine-tuning experiments, such as learning rate and batch size, to optimize the training process.

1. Introduction

Speechreading is a task to infer speech information from visually observed facial movements. It is beneficial for understanding voice-loss conversation, which is especially useful for helping the deaf and recovering voice-loss video. However, since a single viseme (visual unit of speech) usually has several phonemes (phonetic units of speech), it is difficult for humans to perform. Even if a person can accomplish speechreading, it takes years of training to reach high accuracy. Therefore, an easy tool for speechreading with high accuracy will be significant.

Understanding the significance of speechreading, many works have been done to realize automatic speechreading from silent video. Some of these works have achieved remarkable success such as [4], which can automatically create subtitles for silent facial videos (known as Lip2Text) with 95.2% accuracy in sentence-level and 86.4% accuracy in word-level. This accuracy is outstanding, but it is always profitable to explore methods for further improvement. Therefore, our project focuses on exploring possible methods to further improve the performance of the model in [4].

In this paper, we present our exploration of possible methods for improving lip2text model. We adjust and implement the original code provided by [4] in Google Colab and make it workable. Then we modify the model to replace the GRU network to LSTM network and add an extra residual connection to the model to improve its performance. We also try to fine tune the model to improve its behavior. Finally, we do experiments to test whether the performance can be improved through the methods we proposed. Compared with original paper [4], we attempt to include an extra metric to further test generated models' ability of generating understandable audio.

The outline of the paper is as follows. In the next section, we will introduce the related works of Lip2Text and other models and methods used in our project. Section 3 introduces the approach we used to accomplish the Lip2Text task and the adjustments we tried to improve the model. In Section 4, we will demonstrate the results of all adjustments and explain the experiments we did for adjustment evaluation. In the last section, we will discuss our experience of doing this project, sharing the challenges, difficulties, and possibilities we met during the project.

2. Related Work

LipNet The most related work to our project is [4] since we try to further improve the performance of their model "LipNet". With the help of spatiotemporal convolutional neural networks (STCNNs) and Recurrent neural networks (RNNs), they present LipNet that can map variable-length sequences of video frames to text rather than only realize word classification. The high accuracy of the model is impressive. Therefore, it

deserves further exploration on possible improvement on this model.

LSTM LSTM is the abbreviation of Long Short-Term Memory, which is an artificial recurrent neural network architecture first proposed by Sepp Hochreiter and Jürgen Schmidhuber in [12]. Compared with normal RNN or Hidden Markov Model, LSTM has better performance in long sequence training since it solves gradient vanishing and gradient explosion as a benefit of including forget gate in the design.

ResNet Residual neural network (ResNet) is an artificial neural network focused on solving the degradation of deep networks and accelerating the training process, which is proposed by Kainming He, et. al. in [11]. The core idea of ResNet is directly sending the features extracted in the shallow layers to the deeper layers as extra input (as a shortcut connection) so that deeper layers can at least perform as well as the shallow layers, which efficiently prevent degradation and improve the behavior of a model.

ForwardTacotron ForwardTacotron is a model created by Christian Schäfer in [2]. The model is a modified Tacotron model [15], which can generate natural and coherent speech if given the text as input. Inspired by FastSpeech [14], they adjust the Tacotron to a single forward pass model that can generate speech with the help of duration predictor to align text and generated mel-spectrograms, which allows faster but accurate text-to-speech generation.

3. Approach

This section will introduce the architecture of LipNet[4] in detail and what kinds of adjustments is done for improvement.

3.1. LipNet

3.1.1 Spatiotemporal Convolutions

Convolutional neural networks are helpful in computer vision tasks such as object recognition, which receives images as inputs, since it contains stacked convolutions operating spatially over images. For a basic 2D convolution layer from C channels to C', without a bias and with unit stride, it computes:

$$[conv(x,w)]_{c'ij} = \sum_{c=1}^{C} \sum_{i'=1}^{k_w} \sum_{j'=1}^{k_h} w_{c'ci'j'} x_{c,i+i',j+j'} \quad (1)$$

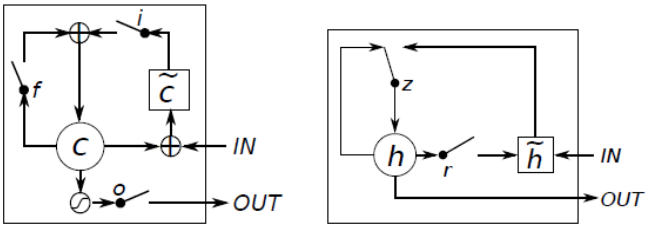
for input x and weights $w \in \mathbb{R}^{C' \times C \times k_w \times k_h}$ where $x_{(cij)} = 0$ for i, j out of bounds.

Different from the basic convolution layer, spatiotemporal convolution neural networks can take a series of images(which equals to a video clip) as input and convolving it both across time and spatial dimensions, which can be represented in formula expression as:

$$[stconv(x,w]_{c'tij} = \sum_{c=1}^{C} \sum_{t'=1}^{k_t} \sum_{i'=1}^{k_w} \sum_{j'=1}^{k_h} w_{c'ct'i'j'} x_{c,t+t',i+i',j+j'} \quad (2)$$

3.1.2 Bidirectional Gated Recurrent Unit (Bi-GRU)

Gated Recurrent Unit (GRU) [5] is a variant of RNN, which has a similar effect as LSTM on handling gradient vanishing and gradient explosion. The difference between GRU and LSTM is that GRU uses two gates (reset gate and update gate) rather than 3 gates in its structure as demonstrated in Fig.1. Bi-GRU[9] is a variant of GRU. Since a normal GRU can do prediction based on only one direction, if combining two GRU that does prediction forward and reversed respectively, it can utilize both prior and posterior information to do current prediction and therefore improve the performance. The structure of Bi-GRU is shown in Fig.2



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 1: *Structure of LSTM and GRU[5]. (a) i, f, o indicate input gate, forget gate and output gate. (b) z, r indicate reset gate and update gate.*

3.1.3 Connectionist Temporal Classification

Connectionist Temporal Classification[10] loss is a widely used loss function in modern speech recogni-

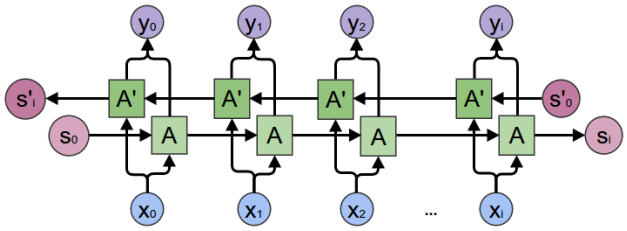


Figure 2: *Structure of Bi-GRU [1]. A and A′ represent a forward and a reversed GRU. S and S′ represent the same input sequence but in normal and reverse order.*

tion since it solves alignment problems between inputs and outputs, which is a common issue in speech recognition. This issue will cause the model to generate a text that contains duplicated letters such as recognizing "learn" to "llleeaarn". This issue can not be solved through easily removing duplicated letters since the model will fail to recognize words such as "happy". To solve this issue, CTC computes the probability of a sequence through marginalising over all sequences defined as equivalent to this sequence:

$$p(y|x) = \Sigma_{u \in \beta^{-1}(y) s.t. |u|=T} p(u_1, ..., u_T|x), y \in V^* \quad (3)$$

where x is the input and y is a label sequence. V^* is the set of all possible tokens. $\beta : \tilde{V}^* \rightarrow V^*$ is a function that deletes adjacent duplicate letters and removes blank tokens given a string over $\tilde{(V)}$. T is the number of time-steps, and u is all possible sequences combination equivalent to y. For example, if $T = 3$, then $p(am) = p(aam) + p(amm) + p(\sqcup am) + p(a \sqcup m) + p(am \sqcup)$.

3.2. LipNet Structure

Fig. 3 (a) demonstrates the original structure of LipNet. The input is a sequence of frames that focuses on the lip, which will be processed by three layers of STCNN followed by a spatial max-pooling layer. After the feature has been extracted, it will be processed by Bi-GRU layer for efficient aggregation. Then a linear transformation will be applied at each time-step, followed by a softmax over the vocabulary (that augmented with the blank symbol). Finally, the CTC loss will be calculated for backpropagation. Rectified linear unit (ReLU)[3]is used as activation functions and Adam optimizer[13] is used for training.

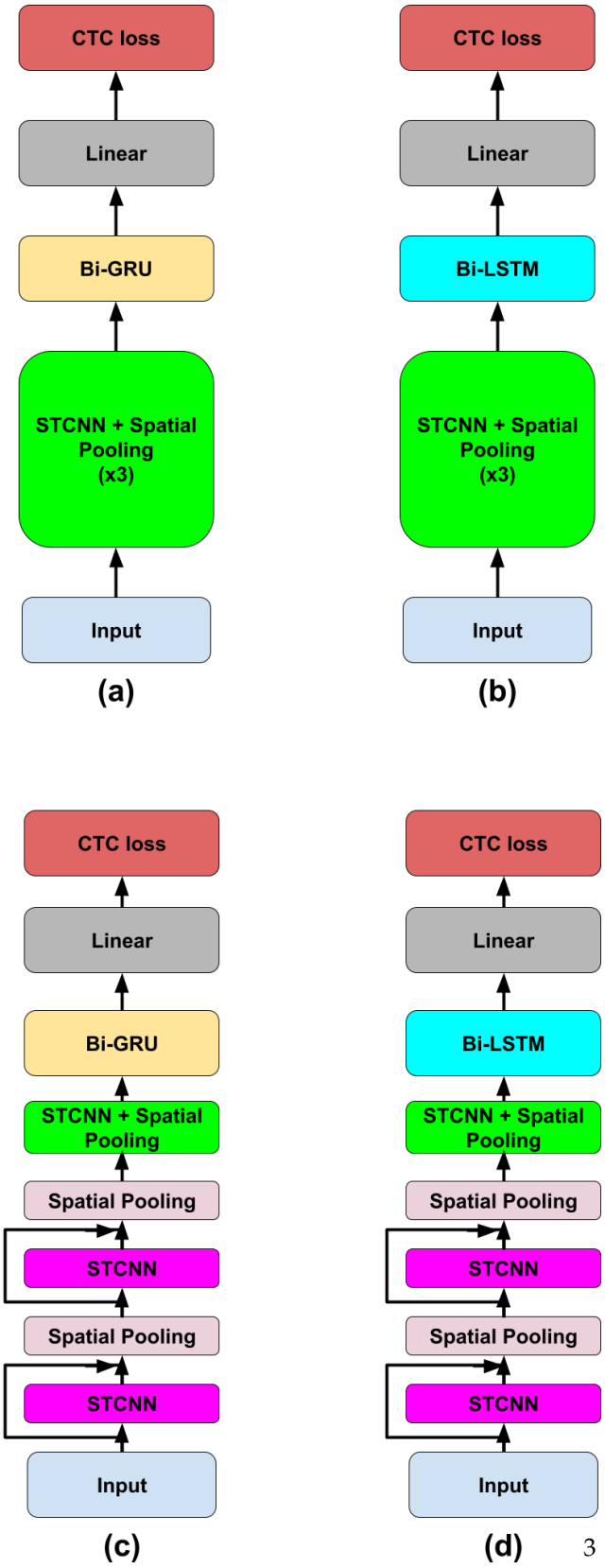


Figure 3: *LipNet structures tested in this project.*

3.3. Adjustment

To get better results, we adjust the original structure of LipNet in two parts. First, we change the RNN components of LipNet from GRU to LSTM, which is also an advanced structure for solving sequence problems. We decide to make this adjustment because LSTM has a more complicated structure, and theoretically, it may generate better results than GRU in some situation. Second, we apply residual connections between the convolutional layers to optimize the training process because residual connections can ensure that the deeper layers will not behave worse than the shallower layers. Considering these two directions of adjustment, we test the Lip2Text problem in four different structures: original LipNet, LipNet with LSTM, LipNet with GRU and residual connections, and LipNet with LSTM and residual connections. Fig. 3 (b) indicates the LipNet uses LSTM instead of GRU. Fig. 3 (c, d) shows the structures of adding residual connections to Fig. 3 (a) and Fig. 3 (b) respectively. Besides, we also fine tune the model on batch size and learning rate to explore possible improvements.

4. Results

4.1. Datasets

The dataset we use for this project is the GRID corpus[6], which has audio and video recordings of 34 speakers, each speaker generate 1,000 sentences, and a total of 28 hours in 34,000 sentences. The sentences follows a grammar: *command + color + preposition + letter + digit + adverb*.

4.2. Metrics

To measure the performance of the models, we calculated training loss, the word error rate (WER), and character error rate (CER). Among them, WER and CER are standard indicators of the automatic speech recognition (ASR) model's performance. We also take notes of the training duration(Time) of each experiment. By performing a CTC beam search, we generated an approximate maximum probability prediction from LipNet. WER or CER is defined as the minimum number of words or characters inserted, replaced, and deleted required to convert predictions into basic facts, divided by the number of words or characters in the basic facts. Note that when the predicted sentence has the same number of words as the basic facts, WER is usually equal to a classification error, especially in our case, because almost all errors are substitution errors.

We also plan to use the mean opinion score (MOS) as a metric since we can generate corresponding audio from the generated sentences with the help of ForwardTacotron[2] because how understandable the generated audio is can reflect the quality of the generated sentence to some extent if the text-to-speech model is reliable. However, due to the time limitation, we have not done this experiment yet, so that we leave it as future work.

4.3. Baselines

To evaluate our implementation of the model and our adjustments, we compare our results to the performance of the original model of LipNet. The original results from the paper are derived from two subsets of GRID datasets, one is unseen speakers, and the other subset is overlapped speakers. To limit the project's scope, we only acquire the results based on the former part of the dataset; therefore, we use the CER and WER of unseen speakers from the paper and trained after 4 epochs as our baseline.

We also introduce a qualitative baseline in this project. It is a video clip, and the speech of the speaker in it is '*Place Red In A Zero Now*'.

4.4. Experiments

The experiments are implemented in Google Colab, and two kinds of experiments, model adjustment and hyperparameters tuning, are listed in this section, and the purpose of these experiments is two-fold. First, we try to find if we can optimize the model structure from the LipNet paper. Second, we do the fine-tuning process to get optimized parameters for training the models. Here, we tune the parameters of learning rate and batch size. The following results are derived after training four epochs.

4.4.1 Model structure experiments

For model structure experiments, we apply a batch size of 4 and a learning rate of 2e-5. To speed up the models' training, we applied the authors' pre-trained

weights obtained after training over 10000 epochs for the model modules except for the RNN parts.

1) Original LipNet structure:

   We train the original LipNet model and get the result in Table 1. As mentioned before, we regard the result here as the baseline of later experiments. The qualitative result is shown in Fig. 4.

Model	CER	WER	Loss	Time(seconds)
Original LipNet	0.3477	0.5995	0.9654	13411

Table 1: *Results of experiment on the original LipNet*

Figure 4: *Lip to text result for original LipNet*

2) LipNet with LSTM:

   We then change the GRU components in the original structure to LSTM and redo the experiment to get the following results. The quantitative result is shown in Table. 2, and the qualitative result is shown in Fig. 5.

Model	CER	WER	Loss	Time(seconds)
LipNet & LSTM	0.3169	0.5672	0.9196	13312

Table 2: *Results of experiment on LipNet with LSTM*

3) LipNet with residual connections:

   We have tried applying residual connections in all three convolutional layers but found out the results are not acceptable. The result is listed in Table. 3 and the result of the next experiment are obtained from residual connections applied in the first two CNN layers.

4) LipNet with LSTM and residual connections This experiment is implemented with LipNet of LSTM

Figure 5: *Lip to text result for LipNet with LSTM*

Model	CER	WER	Loss	Time(seconds)
LipNet & GRU & residual	0.4720	0.8066	1.2133	13392

Table 3: *Results of experiment on LipNet with GRU and Residual connections*

and Residual connections, and the result is shown in Table 4.

Model	CER	WER	Loss	Time(seconds)
LipNet & LSTM & residual	0.3918	0.6709	1.0967	13211

Table 4: *Results of experiment on LipNet with LSTM and Residual connections*

From the results shown above, we conclude that the adjustment from GRU to LSTM provides better performance as its WER, CER, and loss are low, while we can not tell which one has a better qualitative result comparing Fig. 4 and Fig. 5. However, adding residual connections does not optimize the LipNet model.

4.4.2 Hyperparameters tuning

As mentioned before, we do not apply the authors' pre-trained weights for the RNN components for model structure experiments, but in the hyperparameters tuning experiments, we base on the original structure of LipNet and apply pre-trained weights for all layers because we believe that this will not affect the fairness of the selection of the hyperparameters.

1) Different learning rates: The results of tuning different learning rate is presented in Table. 5.

   The results indicate that a higher learning rate can positively influence the model's performance.

Learning rate	CER	WER	Loss	Time(seconds)
2e-4	0.0932	0.1832	0.9191	14409
2e-6	0.0743	0.1414	0.0003	10807

Table 5: *Learning rate tuning results*

2) Different batch size: The results of tuning different batch size is presented in Table. 6.

Batch size	CER	WER	Loss	Time(seconds)
8	0.0747	0.1429	0.4980	10433
2	0.0823	0.1525	0.0627	10825

Table 6: *Batch size tuning results*

From Table. 6, we can conclude that a larger batch size can contribute to obtaining better results.

4.5. Limitation and Future Work

Due to limited time and hardware constraints, all of the adjustments are tested after several training epochs, which may be insufficient to have a reliable performance for evaluation and comparison. Therefore, we plan to offer each adjustment enough training epochs before evaluation. Besides, we adjust only batch size and learning rate in a small range for fine-tuning, which leaves plenty of space for further exploration. This is also a possible aspect for future work since there are other hyperparameters, such as the number of frames for each input that deserves adjustment. Moreover, whether ForwardTacotron is a suitable model for generating test sets for MOS evaluation is also a topic that remains to be discussed.

5. Discussion

For us, this is a tough but very favorable experience. We met a lot of difficulties from the beginning of this project, which can be traced back to the time we accomplished the proposal. After we decided to do speechreading, we found a really good paper [7] for our project, which is also the one we mentioned in the proposal. We were excited about it and then started to contact the author for the source code since it's not published on the internet. Unfortunately, the author seemed not very willing to share the code so that we have to either develop the model by ourselves or switch to another paper. At that time, we thought that it would be meaningful and possible if we could first build the structure ourselves and then tried some adjustment on it since we have the paper and we found the paper[8] and the source code that the author based his work on. Therefore, we spent a lot of energy on this part but we failed finally. Then we had to choose option 2, which is to switch to a new paper for our project only 2 weeks before the deadline. We had to read plenty of paper again and find another one suitable but fortunately, we found it, based on which we did our current project.

One of the difficulties that surprise us most during the project is how hard it might be to use an existing code and make it workable. It should be very easy if someone wants to use the code offered in a recent paper. However, if a person wants to use a code developed several years ago (in our case the code is published in 2017), he/she might face plenty of issues due to updates of modules that the code is based on. This issue can be very deadly, which causes our failure in our first attempt and waste us a lot of time.

There are other predictable difficulties during the project. Such as how to find possible improvement methods, how to adjust the code to fit our expected assumptions, how to divide the work and work together so that one's adjustment on code will not affect the progress of another. Since we have prepared for these difficulties and we discuss and cooperate frequently during the project, they did not frustrate us a lot.

These are the major difficulties we met during the project, which frustrated us a lot but also brought us many benefits. First, we again realized the importance of cooperation and communication, which contributes significantly to the accomplishment of our project. Second, we learned some new meaningful skills that may benefit our research in the future, such as how to ffmpeg to generate data that we expect for training the model. Besides, since we now realize the possible issues caused by version differences, we will especially pay attention to this issue before we implement the code of other researches, and we will include the version information about the modules we used in our own released project to help anyone who wants to use it to prevent possible issues.

In brief, although this is a tough project for us, it is worthwhile and we enjoy it very much.

References

[1] Bidirectional rnn for digit classification. https://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/bidirectional-rnn-for-classification. Accessed: 2021-04-10.

[2] Forwardtacotron. https://github.com/as-ideas/ForwardTacotron. Accessed: 2021-04-10.

[3] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[4] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. Lipnet: End-to-end sentence-level lipreading, 2016.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[6] M Cooke, J Barker, S Cunningham, and X Shao. An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America*, 120(5):2421–2424, 2006.

[7] Ariel Ephrat, Tavi Halperin, and Shmuel Peleg. Improved speech reconstruction from silent video, 2017.

[8] Ariel Ephrat and Shmuel Peleg. Vid2speech: Speech reconstruction from silent video, 2017.

[9] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4, 2005.

[10] Alex Graves, Santiago Fernández, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[14] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech, 2019.

[15] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis, 2017.