Lip Reading Number Recognition

W251 Final Project

Marcus Chen, Marcelo Scatolin Queiroz, Sylvia Yang, Wei Wang

INTRODUCTION

Lipreading is defined as identifying text based on the movement of the speaker's mouth. It is a challenging task due to the fact that the degree of mouth movement and lip sequence vary as one pronounces words, causing ambiguities and increasing the difficulties of guessing. However, its potential practical applications in assisting hearing aids, recognizing speech in noisy environments, resolving multi-talker simultaneous speech and many other fields have put it in a crucial role for human communication, thus motivated many studies in this topic.

Lipreading has been considered as a talent for humans who can perform this task, even with poor accuracy. With the hope of widely and consistently apply this technique, machine lipreading models have been developed to assist with the task. Past studies present two types of lipreading models: recognize on word level, such as classification methods with deep learning; or to conduct sentence level learning, with or without deep learning techniques. Among the two, we consider the sentence level modeling a superior method since it provides more contextual understanding which helps in an ambiguous communication channel.

In this paper, we will show how to adapt an end-to-end sentence level lipreading model called LipNet to perform a simpler task: predicting spoken digits (zero to nine) without the help of audio or other contextual clues. We decide to choose LipNet because it has been demonstrated high accuracy from the initial development (95.2% correct word predictions) and the richness and well documented code, available in github. LipNet was originally trained on GRID corpus that leveraged deep learning architectures, including Spatio Temporal Convolutional Neural Networks (STCNNs), Recurrent Neural Networks (RNNs), Bi-directional Gated Recurrent Unit (Bi-GRUs) and Connectionist Temporal Classification (CTC) algorithms to map video only frames to text sequences. The mathematical details of LipNet architecture is beyond the scope of this paper. We refer readers to Assael, Shillingford, Whiteson, & de Freitas (2016) for an extensive review.

Since the original LipNet model were trained in a sentences-based dataset with limited vocabulary, we decided to use a different dataset were we could extract spoken digits and train LipNet. This way, we will use the MODALITY corpus, a 35 speakers dataset with commands

that contains digits and other words collected in 2016 at the Gdansk University of Technology in Poland. Therefore, our task focuses on training LipNet model on MODALITY using IBM Cloud Virtual Machines.

Finally, we hope such technique can be used in commercial environment soon. Potential uses are:

- Improve robustness in speech recognition, particularly in noisy environments;
- Combination with facial recognition techniques where one can speak silently of his/her password towards camera so that businesses and banks can first confirm the identity of the person using facial recognition then confirm the password as well;

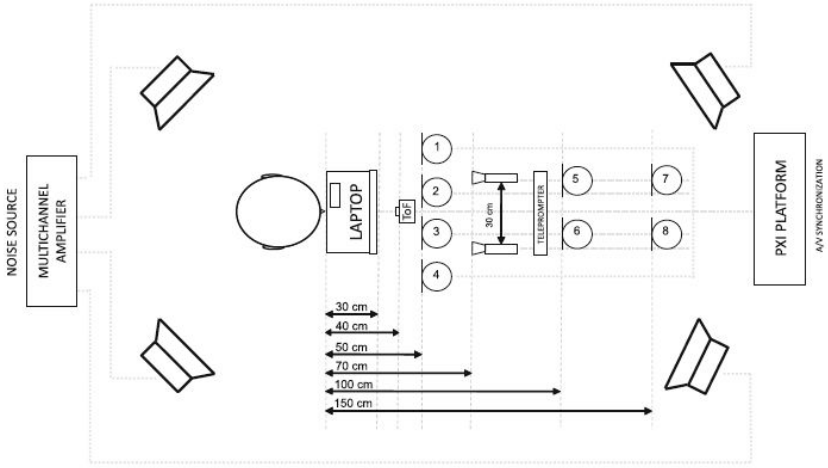
DATASET

With the popularity of lipreading studies increase, many datasets have been put together for researchers' convenience. LipNet model was originally developed using GRID corpus, which consist of both audio and video recordings of 34 speakers who produced 1000 sentences each. Each sentence consisted of 6 words categories: a command, a color, a preposition, a letter, a digit and an adverb, yielding 64,000 possible sentences. For example, two sentences in the data are "set blue by A four please" and "place red at C zero again". The initial idea was to build scripts to separate the digits and retrain the network specifically to those categories, however, we decided that adding an unseen set of speakers would benefit our results. Below there is an example of the GRID dataset videos.

Our study re-trained LipNet model using MODALITY corpus. It contains high-resolution (1080x1920 pixels at 100 fps) video streams and audio signals of 35 speakers, including both native and non-native speakers. There are two types of files for each speaker: “commands” files have words spaced by pauses whereas “sentence” files contains sentences in sequence. In order to identify numbers ranging from 0 to 9, “commands” videos were extracted and separated into short clips for each number that is pronounced.

The specific data processing steps are as follows:

1. MODALITY corpus is available here. Each “commands” file contains one .lab file with timestamps for words or sequences that were said, audio files, and `.mkv` files with images from cameras positioned at both left and right of the face. We used both camera files and audio files from microphone 2, as seen on the picture below.

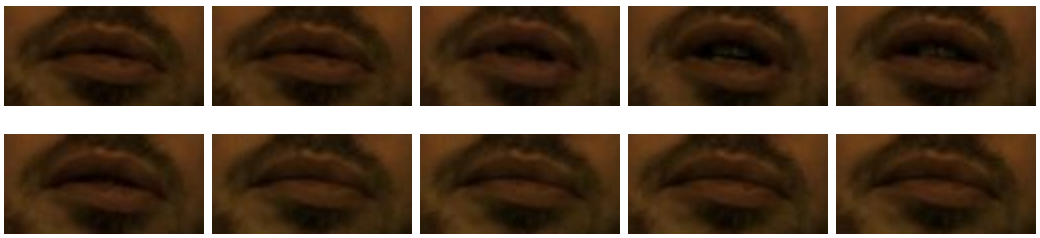


2. To improve the time to download, instead of manually downloading the videos, we defined a script to download all command videos and save the ones with transcriptions available (necessary for later clipping step) to a cloud based machine (see raw_downloader.py on our GitHub repo).
3. The original MODALITY commands videos have 12 videos per speaker (6 from the left camera, 6 from the right) in 1920x1080 pixels with 100 frames per second. As we are interested only on the spoken digits, we created another script, file_clipper.py, to clip each video where digits are spoken. Additionally we reduced the resolution to640x360 pixel to improve storage and processing time, and finally we saved all in .mpg format (as required by LipNet training scripts). The final dataset has 790 MB of videos which are 5548 files, and was saved to IBM Cloud bucket. The audio files, although not used in this

work, were preserved as well, case we decide to build on this project on a later date. Examples of processed videos are shown below:

4. In order to adapt the file to the format accepted by LipNet, we used the MODALITY_to_GRID_Converter.py script. This program sets each file to 75 frames, using frozen frames and the beginning and the end of each video to fill in missing frame count. It also saves the video in the needed directory structure, with matching names for de videos and the transcriptions.

5. Finally, in a 2xV100 enabled IBM Cloud Virtual Machine that we will use for training, we used the extract_mouth_batch.py script to crop all videos again with a 100x50 pixels crop of the mouth of each speaker and saved frame by frame. The final files were split between train (13 speakers) and validation (4 speakers) and were ready for training. Below there is an example of the result:



LipNet Model

Model Architecture

The building block of this model are, as mentioned before, the Spatio Temporal Convolutional Neural Networks, or STCNNs, which is similar to a regular CNN but has the capability to convolve across time due to an extra step in the convolution calculations. So if a regular CNN with no bias and with stride one can be represented by

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

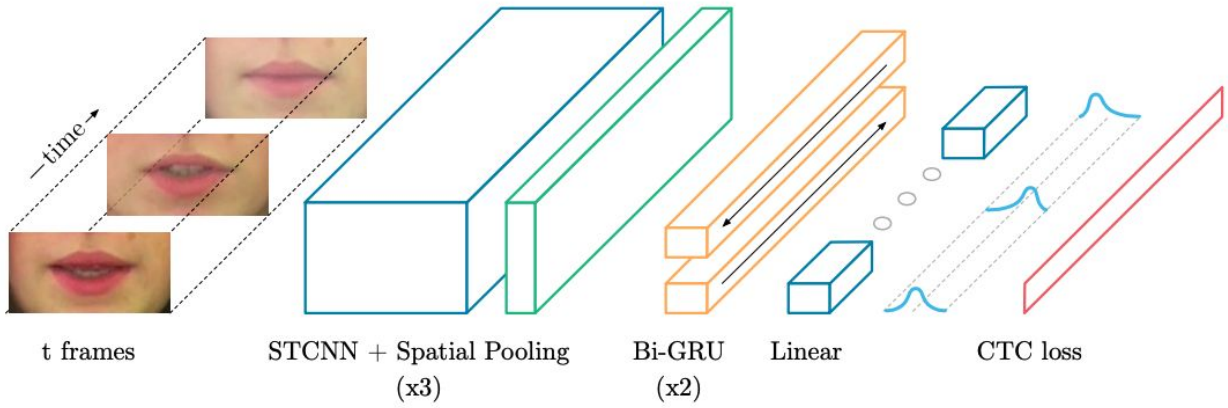
a STCNN will be represented by:

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

The next block in this model are Bidirectional Gated Recurrent Network, which improves the performance of regular CNNs by implementing and learning how to control gates for propagating information over more time-steps.

Finally they use a Connectionist Temporal Classification loss model that is widely used in speech recognition. CTC works computing the probability of a sequence by marginalising over all sequences that are defined as equivalents to this sequence.

LipNet architecture consists of 3x (STCNN, channel-wise dropout, spatial max-pooling), followed by 2x Bi-GRUs, a linear transformation for each time step, a soft-max layer over the vocabulary and finally the CTC loss. All layers are activated using ReLu activation function. For more details, please refer to the original paper. A schematic of this architecture can be found below.



Implementation

We levered a public repository for LipNet built by Muhammad Rizki (https://github.com/rizkiarm/LipNet). The code was written in 2017 using Tensorflow-gpu 1.0.1 (cuda 8), Keras 2.0.2 and python 2.7. Our initial plan was to deploy the pre-trained model to process live video stream on TX2, however due to multiple compatibility issues we ended up deploying the model on a 2x V100 VM.

In our first attempt we built a docker image with tensorflow-gpu 1.13.1 (which is the only version officially supported by Nvidia for Jetson), Keras 2.2.4 and python 3.6 on TX2. We converted the original py2 code to py3, but eventually came to a point where the prediction would break. We spent a long time on debugging the code but couldn't resolve the issue.

The second attempt was to deploy LipNet on a P100 VM with an x86 CPU. We leveraged the latest docker 19.03 and was able to directly run the official Tensorflow-gpu 1.01 container. This time we built the container as close to the original environment as possible. We installed Tensorflow-gpu 1.0.1, Keras 2.2.4 and python 2.7 and was able to run the prediction.

In the third attempt we tried to deploy the image built on the cloud VM to the tx2, as we expect the nvidia-docker (included in Jetpack 4.2.1) would work similar to docker 19.03 on the P100. Unfortunately the image was not compatible.

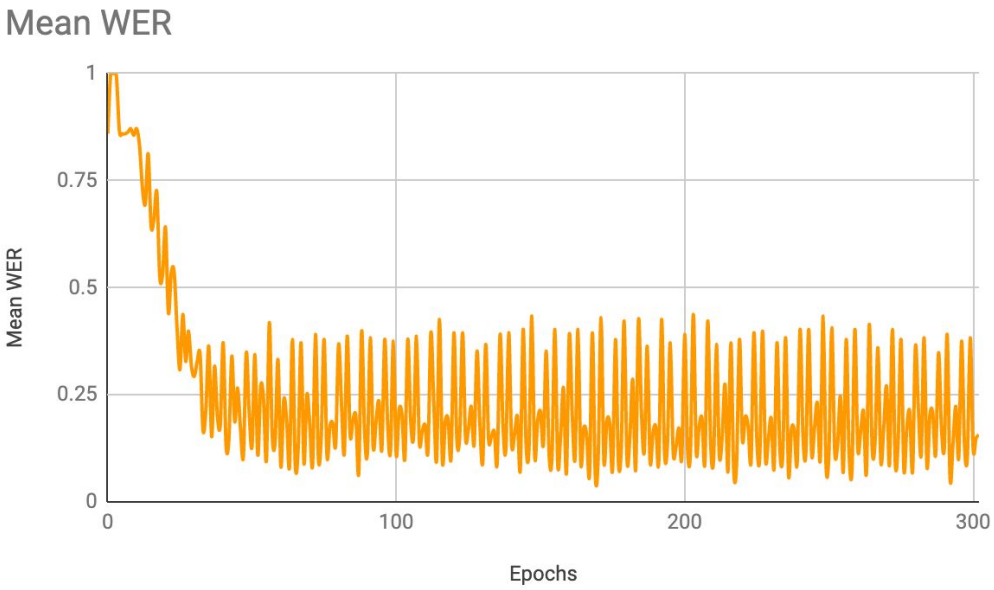
We figured out the root cause. The original LipNet code was designed for TF1.01 and does not work on TF1.13, and TX2 only have official support for TF1.13 at this moment. In order for the code to work, we have to either migrate the LipNet code to TF1.13, or compile TF1.01 on TX2 from the source code. As we were close to the deadline, we had to move on without the TX2 deployment. Eventually we ended up with a working implementation of the LipNet model on the ibmcloud VM.

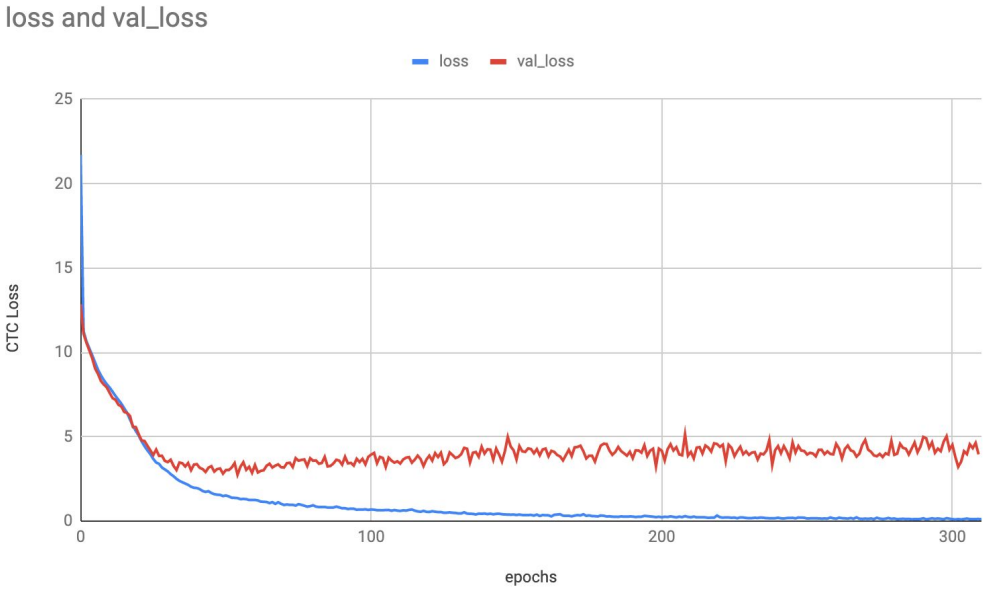
PERFORMANCE

The original LipNet model was trained with 5000 epochs, reaching Word Error Rate (WER) of 11.4% for unseen speakers and 4.8% for overlapped speakers (when a random set of 255 sentences from each speaker was used to evaluations, the rest for training). In our case, due to the limited dictionary, [zero, …, nine], we decided to go with less epochs: 1000. However, after 100 epochs we observed a stabilization of the mean WER and the training/validation losses. For that reason, we decided to stop training on epoch 300 and verify the results. Each epoch took approximately 3 minutes to compute, reaching a total training time of 15 hours and 27 minutes. The final WER rate was 12.5% which places us behind works like Pass et. al., 2010, with 2% WER, but better than some recent works like Stewart et. al. 2014, with 30% WER. For other comparisons, see table 4 in Fernandez-Lopez et. al. in our references section.

It is important to note here that our model were highly overfitted for the training speakers and the camera setups in MODALITY dataset as noted in the loss and the WER charts below. After epoch 20, approximately, we start to see an increasing variability on mean WER calculation and separation on the train/evaluation losses. This can be due to the light conditions of the MODALITY and the sequential nature of the STCNNs and RNNs that are implemented in LipNet and are not being used for this work. As ways to improve this performance we could work on using the augmenting techniques already implemented in LipNet code (which we decided not to use in this first work to assess the “pure” results). Another idea is to paste different videos together (as well as their transcripts), to take advantage of the sequential features. The latter solution could be even more valuable for proposed uses like silent password spelling, where inferencing a sequence of digits makes more sense than running the model for one digit at a time.

The overfitting behavior became more evident when we ran inference using videos of the group components. In our tests, we reached a WER score of 10%, which is not better than randomly guessing in a single digits vocabulary.





FUTURE DEVELOPMENT

Since this project only uses video frames to identify numbers, there are potential to leverage the audio files as well to increase the accuracy and broaden the application potential. Such data is available in MODALITY corpus as well and we have prepared the script of handling audio tracks in a similar manner to the video processing to benefit future researchers.

As mentioned in the previous section, to avoid the overfitting problem, we plan to work on using the data augmentation tools in LipNet (video horizontal mirroring, jittering the image, color channel manipulation, video speed manipulation). Related to this solution, is the use of a “ground truth” generator script, where random multiple digits number can be used to paste videos of the same speaker together. Finally we can work on adapting other datasets to the same input format, being able to avoid biases based on the environmental conditions the videos were shot.

Should we successfully deployed the code to run inference on TX2 on live stream, an eminent improvement could be achieved by replacing the video processing module from scikit-video (using FFmpeg backend) to opencv to leverage hardware acceleration.
With our current status (TX2 partially functional, fully working on P100) we could use TX2 to detect and retrieve the lips movement, send the data to cloud and perform the inference there.

REFERENCE

Assael, Y. M., Shillingford, B., Whiteson, S., & De Freitas, N. (2016). LIPNET: END-TO-END SENTENCE-LEVEL LIPREADING. Retrieved August 4, 2019, from https://arxiv.org/pdf/1611.01599v2.pdf.

Adriana Fernandez-Lopez, Federico M. Sukno. (2018). Survey on automatic lip-reading in the era of deep learning. Retrieved July 27, 2019, from https://www.sciencedirect.com/science/article/pii/S0262885618301276

LipNet github