Read My Lips

Team: Xiaotong Chen(06358484), Hao Mao(06349831)
Email: {xc2230, haomao}@stanford.com
Code: https://github.com/Xtchen/cs230

Introduction

Both of us are big NBA fans. From time to time, we see players talking on the field but what are they really talking about? To answer that question, we designed an end-to-end deep neural network that takes grayscale video that contains mouth area as input and outputs the corresponding sentence. The process of recognizing speech based on lips movements is also known as lip reading.

Related work

Lip Reading in the Wild[1] introduced a CNN architecture that focused on recognizing individual words. It also compared the performance of 3D CNN and VGG-M. VGG-M showed decent performance, which is why we decided to use VGG-M to ingest the video frames in our project. However, the concept of recognizing individual words may not work well in recognizing sentences. So we found Lip Reading Sentences in the Wild[2] is more feasible for our project. It talked about a deep neural network leveraging LSTM to match the two sequences, lip movements and characters. We were inspired by the idea and implemented something very similar. As we doing the error analysis, we realized that our model doesn't work well for the profile view- actually it only works for the frontal view, because there aren't many video from profile view in our dataset. As described in Lip Reading in Profile[3], with almost the same architecture, feeding the model more video from profile view could improve the accuracy a lot. In addition, we went through Deep Audio-Visual Speech Recognition[4] which talked about an idea that uses a spatio-temporal ResNet to process the video, while the audio features are the spectrograms obtained by applying Short Time Fourier Transform (STFT) to the audio signal. We didn't take the idea since we want to eventually make the prediction without any audio.
To gain more intuition about the attention layer, we found Listen, Attend and Spell[5] a good one to read. It also demonstrated that increasing beam width is almost always helpful in terms of the prediction accuracy.(WER, Word Error Rate)

Dataset and features

The LRS3[6] dataset is pre-processed with speaker's face recognized and correctly labeled sentences. We also downloaded some NBA games, prost-game interview videos from nba.com[7].
Based on our architecture, we need to process the data by cropping the mouth area to exclude noises like background, hair style, eyes and so on.

We used dlib with shape_predictor_68_face_landmarks[8] parameters to detect the mouth area. To make training faster, we also converted frames from RGB to grayscale as the color should have nothing to do with what the speaker is talking about. See Figure 1 for some examples of the dataset.
In total, our dataset has ~10K videos (1000 hours) and contains 3.9M word instances. We use 96% of the dataset as training set; 2% as dev set; 2% as test set. The NBA videos are evenly distributed. We reserved a small amount of them as a real NBA dataset to better evaluate the performance of our model.

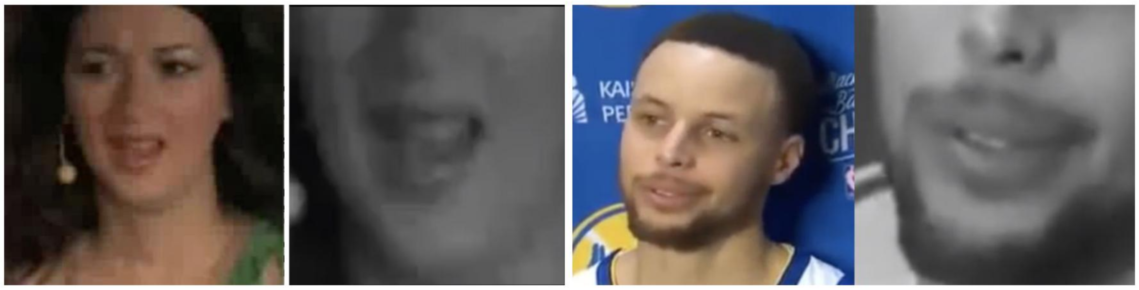


Figure 1 examples of training set and test set. ***Left*** *is the original video frame.* ***Right*** *is the same frame after our data processing with mouth detection, cropping and grayscale. Resolution after processing is 120 x 120.*

Methods

Overview of architecture

We designed and implemented the following architecture (see Figure 2) that includes three components: **Watcher** runs frames through VGG to understand the lip motion and then feed the class vector to LSTM; **Speller** takes output of Watcher from LSTM along with character sequence and runs them through LSTM; **Attention** figures out the right period of lip motions and aligns these two sequences. We used cross entropy as the loss function since predicting a character based on lip movements is classification.

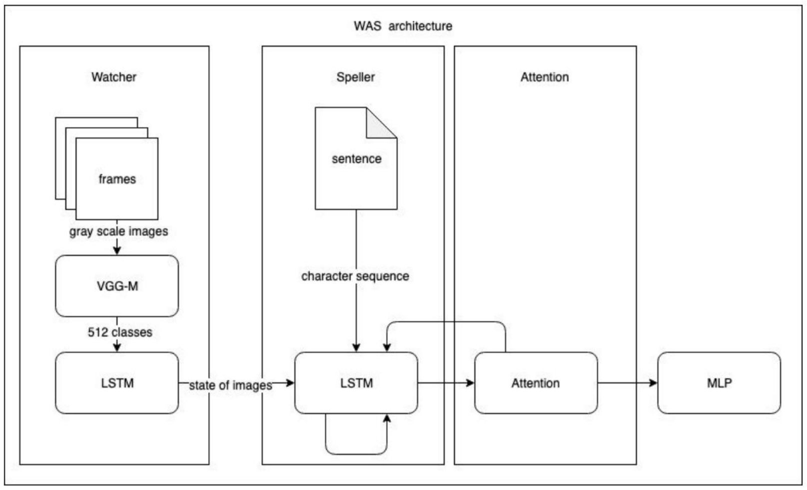
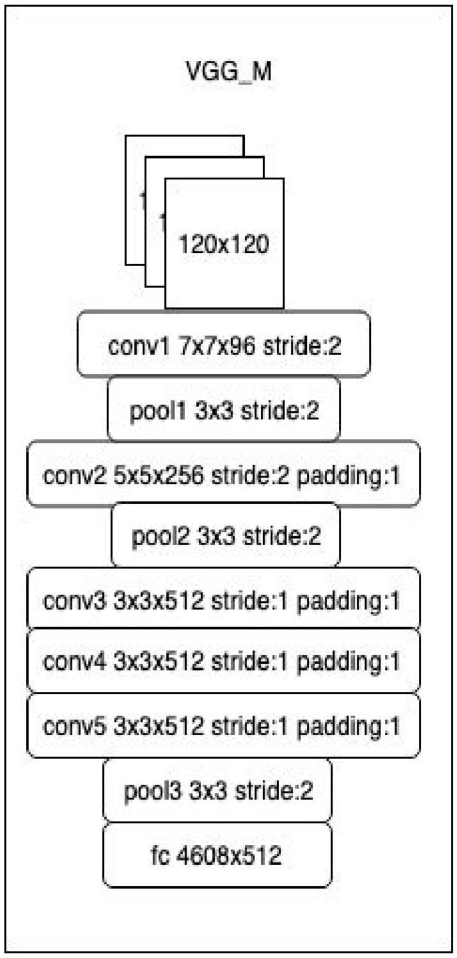


Figure 2 the overview of WAS architecture. The output of the MLP is a character, and it's a classification problem. So we use cross entropy as the loss function to measure whether or not the output character is the correct one.

Components

Using VGG-M to extract lip movement features

We pick VGG-M to process frames because it runs fast and has decent classification performance.
The input is a 120 x 120 grayscale image for each video frame.
The output is a 512 x 1 vector representing the posture of lips.


VGG_M
120x120
conv1 7x7x96 stride:2
pool1 3x3 stride:2
conv2 5x5x256 stride:2 padding:1
pool2 3x3 stride:2
conv3 3x3x512 stride:1 padding:1
conv4 3x3x512 stride:1 padding:1
conv5 3x3x512 stride:1 padding:1
pool3 3x3 stride:2
fc 4608x512

Feeding lip movement features into LSTM

We feed the output(512 x 1) of VGG-M to a single-directional LSTM with the input size of 512 and hidden size as 512 too. The LSTM has 3 stacked layers. The output are a sequence of features , hidden state and cell state. We drop the cell state here as we no longer care about it moving forwards.

Using LSTM to find out the mapping between character sequence and lip movements

We have another LSTM that takes the hidden state from the watcher net as the initial hidden state. It takes the concatenation of character embedding and context from the attention as the input. The output are, like the previous LSTM, a sequence of features , hidden state and cell state.

Using attention layer to align two sequences

Now we are facing a sequence to sequence problem with difference length. To help our model better align these two sequence, lip movements and characters, we introduce an attention layer. In the end, we feed the output of the attention layer to a MLP which is a 3-layer neural network to classify the lip movements into a character.

Implementation details

Our implementation is based on PyTorch[9] and trained on a NVIDIA GeForce GTX 2080 GPU.

Hyperparameters and tuning

Batch Size

We tried to apply batch training in our model. It requires the same data input length, however the training video and text data we have are not the same length. Some videos are about 10 seconds, some can be 30 seconds. We have two choices to solve this problem in order to apply batch training. The first one is to estimate the maximum video and text input length, and make sure all videos and text have the same shape and length by appending '<pad>'. We tried 1/1000 of the training dataset with batch size equals to one to experiment our idea. The result was poor. Even after 100 epoch, the loss was as high as 50. So we gave up this approach. The second one is to truncate all videos and input text to a fixed video and text length. We tried similar way to experiment as approach one. The result was also poor since it might cause video and text mismatch.

With above two attempts, we decided to do stochastic training.

Frame Size

Because of stochastic training, it will slow down our training. We wanted to try other ways to speed up our training. We noticed a 10 seconds video can have about 200 frames which we were sure a same word can represent in the multiple duplicated frames. With that in mind, instead of storing every frame to the training matrix, we tried to load one frame on every a few frames. We tried 1/1000 dataset to test our idea with loading one on every 3 or 5 frames. It turned out that loading every 5 frames didn't affect training result every much, and it saved around 20% of the training time compare to loading every frame. So for training data, we loaded one frame for every 5 frames.

Evaluation Metrics

While using cross entropy as the loss function during training, we select edit distance per character(EDPC) as the evaluation metric. The reason is that, the prediction usually has different length than the grand truth. E.g. CANT vs. CAT where the edit distance per character is 0.25.

Results and analysis

Training set EDPC: 0.34
Dev set EDPC: 0.36
Test set EDPC: 0.40
Real NBA video EDPC: 0.68

Figure 4 heap map by visualizing weights of attention layer

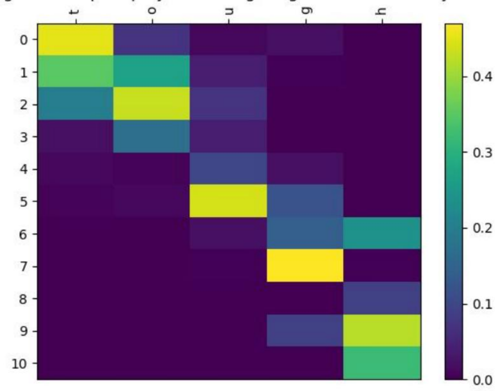


Figure 3 is the word “tough” spoke by Stephen Curry that our model got right. Figure 4 is the heat map which we get by plotting the weights of the attention layer. It shows what frames the model is looking when predicting each character of the word “tough”.

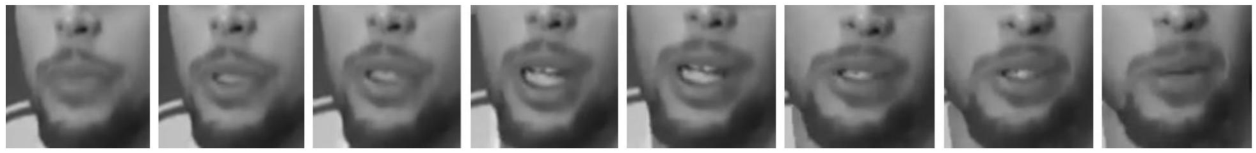
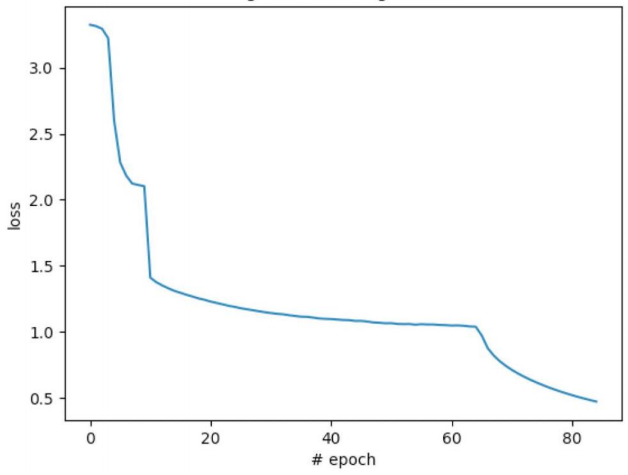


Figure 3 The word “tough” from the sentence “...it was a tough game...” by Stephen Curry.

With 150+ hours training on a NVIDIA GeForce GTX 2080 GPU, we were able to overfit the training set. EDPC of dev and test set are pretty close to that of training set. This is an evidence that our architecture makes sense.
When it comes to the real NBA dataset, which contains 8 real NBA clips, the result isn’t as good as we would like, though it could get some words or characters right.
We did some error analysis finding that our model only works for the frontal view, because there aren’t many video from profile view in our dataset. We believe the fact that the way how NBA players pronounce is not as clear as how speakers in the LRS3 dataset do might also have a big contribute to the poor performance.

Figure 5 Training Losses



Conclusion and future work

Within this semester, we designed and implemented a deep neural network that contains three components, watch, spell and attention, that could recognize sentences solely based on video. Also, we developed deep understanding on CNNs, like VGG-M, RNNs, like LSTM, and concepts like attention, embedding and so on.
By feeding 1000+ hours video to the model, we achieved decent performance on training, dev and test set. Due to the limit number of NBA videos we could get, the performance of real NBA videos is not great. Based on the analysis we mentioned in previous sections, we have future plans to improve the performance.

Future work

(1) Due to time limit, we only trained 85 epoch in about 150 hours. That’s not enough for 1000+ hours video training set. The next step for us is to train another 5 days to see the test performance.
(2) Based on the NBA video test result, we would put more NBA player videos/interviews to our training set.
(3) Given the current training set are all in front view, we could try Lip Reading in Profile[3] to help our model understand profile view video better.

Contributions

Xiaotong Chen: paper research, dataset processing, baseline model design, baseline model implementation, hyperparameter tuning, result/error analysis
Hao Mao: paper research, NBA video crawling, baseline model design, AWS environment setup, hyperparameter tuning, result/error analysis

Reference

[1] J. S. Chung and A. Zisserman. Lip reading in the wild. In Proc. ACCV, 2016
[2] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman. Lip reading sentences in the wild. In CVPR, 2017.
[3] J. S. Chung and A. Zisserman. Lip reading in profile. In Proc. BMVC., 2017.
[4] T. Afouras, J. Chung, A. Senior, O. Vinyals, and A. Zisserman. Deep audio-visual speech recognition. arXiv preprint arXiv:1809.02108, 2018
[5] Chan, W., Jaitly, N., Le, Q. V. & Vinyals, O. Listen, attend and spell. arXiv preprint 244 arXiv:1508.01211, 2015.
[6] Lip Reading Sentences 3 (LRS3) Dataset http://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrs3.html
[7] The official NBA site https://www.nba.com/#/
[8] TensorFace project on github https://github.com/AKSHAYUBHAT/TensorFace/blob/master/openface/models/dlib/shape_predictor_68_face_landmarks.dat
[9] PyTorch Python library https://pytorch.org/