Deep Learning Methods for Lipreading

Jing Hong, Daniel Avery Nisbet, Alex Vlissidis, Qingyang Zhao
University of California, Berkeley
Department of Electrical Engineering & Computer Sciences
{jing_hong, danisbet, alex_vlissidis, qingyang_zhao}@berkeley.edu

Abstract—**Lipreading, the task of determining spoken words from the movement of one's mouth, is performed with low accuracy by humans. Yet accurate lip reading has the potential to help the hearing-impaired community, assist in CCTV analysis and help driverless cars interact with pedestrians. Deep neural networks have made great strides in computer vision tasks, but have not yet achieved the same for performing lipreading from videos. We investigate the performance of three competing network architectures: a spatio-temporal convolutional neural network, a convolutional network with recurrent units, and a pretrained image classification network with recurrent units. We find that although each network can achieve relatively strong performance, a pretrained image classification network with recurrent units achieves the highest accuracy and is most tolerant to various hyperparameters, achieving an 82% accuracy on the GRID dataset.**

I. Problem Statement and Motivation

Lipreading is an essential skill for people suffering from hearing loss, allowing one to understand what others are speaking by reading their lips movement. However, lipreading is an extremely difficult task for humans. Hearing-impaired people can only achieve a word-level accuracy of 17%. As human lipreading performance is poor, we believe deep learning has great potential in this task, with applications ranging from hearing-impaired assistant systems, video analysis, and silent dictation in public spaces. Traditionally, lipreading systems consist of hand picked feature extraction and prediction, however deep learning has yielded superior results with end-to-end trained vision systems. Specifically we focus on speaker-dependent lipreading, which requires the training of the system on a specific speaker before it can predict words in unseen videos of the same speaker.

Spatial convolutions have been proven to successfully abstract features from images while previous research has shown limited success interpreting videos using this method. Naturally, convolutions over space and time (spatio-temporal convolutions) are able to capture time domain information that a spatial convolution would be unable to capture. Another method to capture arbitrary length time domain information are recurrent units. Using these building blocks, many network architectures can be produced to analyze videos.

We investigate the performance of different deep neural network architectures for single word lip reading classification. We use the GRID audiovisual sentence corpus and crop the videos to include single words and just the speaker's mouth. We study the accuracy and tractability of a purely convolutional model, a combination of convolution with recurrent units and a pretrained VGG16 with a recurrent unit.

II. Related Work

Lipreading has been approached via computer vision techniques such as using depth data for 3D pose tracking of the face, with Hidden Markov models (HMMs) [1]. Another approach has been to use unsupervised random forest manifold aligning to video data sets to find embeddings of the speaking video clips by a graph-based algorithm and perform lipreading [2]. Recently, neural networks have been used in speech recognition as feature extractors in HMM-based speech recognizers, and deep learning approaches have proved to achieve state-of-the-art performance. The first deep learning paper, on lip reading, was published by Hinton et al. [3], who used a deep CNN and achieved a top-one error rate of 37.5% in the AlexNet competition. An interesting approach was published by K. Noda et al. [4], who proposed an Audio-visual speed recognition system, using a CNN to extract visual features from raw mouth area images and a multi-stream HMM to integrate the acquired audio and visual respective features, achieving approximately 65% word recognition rate. Additionally, S. Petridis et al. [5] have proposed an LSTM to automatically extract features directly from pixels and perform classification, achieving state-of-the-art performance. M. Wand et al. [6] experimentally evaluated the performance of an LSTM model applied to raw images of the mouth regions, which achieved top-one 79.6% word accuracy, and showed the neural network based lipreading system could achieve better performance than a standard SVM classifier using conventional computer vision features, Eigenlips and Histograms of Oriented Gradients. Google DeepMind's LipNet [7] is the first end-to-end sentence-level lipreading model that simultaneously learns spatiotemporal visual features and a sequence model, achieving a 95.2% sentence level accuracy on the GRID dataset. The key difference though is that LipNet is focused on a slightly different task since it predicts whole sentences, while our system predicts individual words. Perhaps the most closely related work is from A. Gutierrez et Robert [8], who explored a CNN + LSTM combination for this task. They used the MIRACL-VC1 dataset, which is only limited to ten words from ten people. They used a pretrained VGG-16 model, achieving a validation accuracy of 79% and test accuracy of 59%.

III. Dataset

Our experiments were performed using GRID dataset, which is a large multi-speaker audiovisual sentence corpus. It contains video recordings of 360 $\times$ 288 pixel resolution

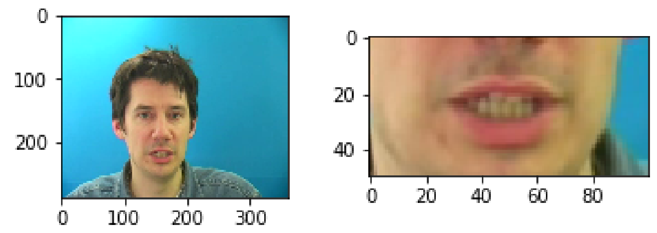

Fig. 1. Original face-centered frame from GRID (left) versus cropped mouth-centered frame (right) for speaker 1.

videos of 1000 sentences spoken by each of 34 speakers and corresponding transcripts. Each sentence consists of six words of fixed structure: command + color + preposition + letter + digit + adverb, i.e. "put red at G9 now", and contains 51 different words in total. The videos are centered on the speakers face. To preprocess the data, we used the DLib face detector to crop the videos to size of 100 $\times$ 50 pixels per frame that only includes the speakers mouth (Figure 1).

We extracted and stored the video frames of individual words spoken and corresponding word alignments (six words for each video). To standardize input sizes, each video was then zero padded so that the number of frames for each video was equal. Due to the limits of computation, we only used speaker 1 for hyper-parameter tuning and speakers 1-19 for validation. Speaker 18 was discarded because almost all of the videos were corrupted.

IV. Methods

For our methods we investigated three different types of models to approach this task. Lipreading through videos can be described as a non-linear function parameterized by θ, which learns the distribution of the probability of all the words of the GRID vocabulary, given the input video. Thus all of our models are trying to learn the following.

$$p(w_i|x_1,\ldots,x_t) = f(x_1,\ldots,x_t;\theta) \tag{1}$$

Where w_i is the i^{th} word in the vocabulary, $x_1 \ldots x_t$ is a video frame padded to t number of frames and θ are the parameters of the networks.

A. Baseline Model

There is a number of deep learning models that have been used for this task. Since our project goal is to compare convolutional versus recurrent models we consider two models as baselines. The first is an recurrent model proposed in *"Lip Reading with Long Short Term Memory"* by M. Wand et al., which is based on using just an LSTM for this task. For the preprocessing step it uses MATLAB's face detector to crop the images to 40 $\times$ 40 pixels, centered around the mouth. These image sequences are fed into a 128 unit LSTM neural network directly and the output is a fully connected layer with a softmax activation function. It achieves an accuracy of 79.4% on speakers 1-19. This paper inspired us for the design of our recurrent model.

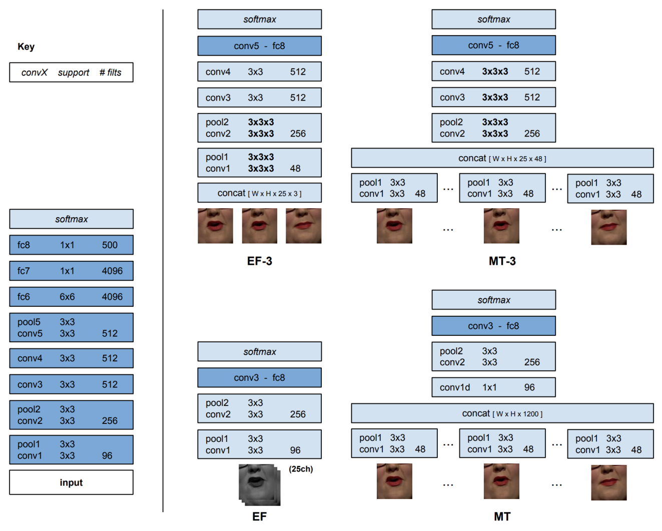

Fig. 2. The architecture for the convolutional baseline model

For the convolutional model we consider the system proposed by *"Lip Reading in the Wild"* by J. Son et al [9]. The best performing model is a multi-head spatial convolutional model. Each frame in the video is processed by a different convolutional head. So the number of heads is the same as the number of video frames. Then the output from each head is concatenated and fed into a spatio-temporal convolution, which predicts the probability of the vocabulary using softmax. The architectures of all the models proposed are shown in Figure 2.

The paper proposing this model also crops the video centered around the mouth. However, this paper uses the BBC dataset, which is much larger as it contains versions with 333 and 500 classes. It achieves accuracies of 65.4% and 61.1% respectively. This model served as our inspiration for our spatio-temporal convolutional model, although we cannot directly compare with its performance since we are using a different dataset.

B. Spatio-Temporal CNN

We propose a model based on spatio-temportal convolutions. This operation is the same as the spatial convolution operation, but extended in the time domain as well. For an input image $x \in \mathbb{R}^{C\times W\times H}$ and and kernel weights $w \in \mathbb{R}^{C'\times C\times k_w\times k_h}$ a spatio-temporal convolution is defined as shown in Equation 2.

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

We constructed a network from spatio-temporal convolutions and fully connected layers. The architecture, shown in Figure 3, consists of two convolutional heads followed by two linear layers. Each convolutional head consists of three iterations of spatio-temporal convolutions, followed by max

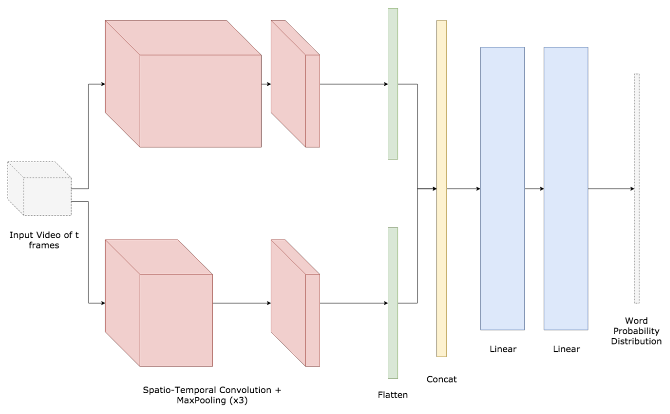
Input Video of t frames
Spatio-Temporal Convolution + MaxPooling (x3)
Flatten
Concat
Linear
Linear
Word Probability Distribution

Fig. 3. The architecture for the Spatio-Temporal Convolutional Network

pooling and dropout. One head convolves three frames of the video whereas the other head convolves seven frames. Although all convolutions have the same spatial dimensions, 3×3 kernels, the differing time frame for the convolutions allows the network to identify both short term and long term features and combining those to make more accurate predictions.

The intuition behind this architecture is to have the spatio-temporal convolutions capture temporal and spatial information from the video, thus learning feature extraction and temporal signals at the same time. However, unlike using an RNN network, which is flexible in its input, we implemented two heads in order to have different amounts of memory in each head. Ideally, we would like to have more granularity in this by having more heads. However, this would render the network too large and intractable for our computational resources.

C. CNN + GRU

For our recurrent model we proposed a network formed from three layers of spatial convolutions and maxpooling layers, in order to encode the images into the feature maps. Each convolutional + maxpooling block is applied to each of the video frames separately, using the same weights for all frames, producing essentially a video of features. This temporal signal is fed into a two layer bidirectional GRU. Finally the output is produced using a wide fully connected layer, which outputs a probability distribution over the GRID vocabulary, using a softmax activation function. The architecture is shown in Figure 4. Our convolutional layers use 3×3 kernels and 42, 96 and 128 filters for the three convolutional layers respectively. The GRUs have 256 hidden units each.

The intuition behind this model is to have the convolutional layers learn the important spatial features from the mouth-centered video frames, which will distinguish between different words or letters. This also reduces the dimensionality of the input, thus making the GRU more efficient. Then these feature maps are fed into the recurrent GRU, in order to capture the temporal information in the features of the video and thus the changes of the shape of the mouth over time.

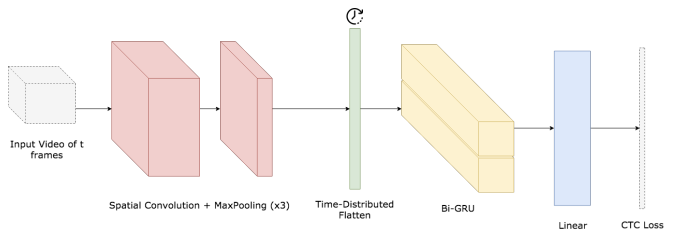

Fig. 4. The architecture for the CNN + GRU model

D. VGG16 + GRU

The last model we experimented with was inspired from the CNN + GRU model. We used the VGG16 model pretrained on ImageNet. In this implementation we replaced our custom CNN with VGG16, without the top fully connected layers, with frozen weights. The features extracted from it are fed into the same 2 layer GRU setup as before. This model is based on the assumption that VGG16 has the ability to generalize and thus extract the important features from the mouth of a human. A limitation for this project was that the preprocessed mouth videos are fairly small (100×50 pixels per frame), eliminating the use of any Inception network or ResNet, as these are very deep networks and require high resolution images.

E. Training

As mentioned in the Dataset section, all the models' hyperparameters were tuned using speaker 1 for computational reasons. Then the best model is trained on speakers 1-19 in order to compare our results with state-of-the-art. For all models we used the Adam optimizer with variable learning rate, along with various amounts of dropout minimizing the multi-class cross-entropy loss. In addition to that we used L2 regularization on the last fully connected layer of each network in order to boost generalization. Furthermore, we used early stopping in order to prevent overfitting of our models. We configured the optimizer to stop the training if the validation accuracy did not increase by 0.5% for 10 epochs. Due to the limited amount of data (approximately 5000-6000 words per speaker), we used a 60-20-20 train-validation-test split.

V. Results

For each of our models we explored several hyperparameters in order to achieve the optimal performance for each one. One limitation here is that due to time constraints and computational limitations, a full random search of hyperparameters was not performed, although it would be ideal. For each of the models we used out empirical intuition to perform experiments on different combination on the key hyperparameters of each model. Our results are shown in the tables below, also indicating the best performance for each model we are investigating.

The STCNN exhibited the most variation in performance. It proved to be very sensitive to dropout, as well as the size of the output fully connected layers. The best performing model achieved a 73% test accuracy for speaker 1.

Dropout	Learning Rate	FC Size	Accuracy
0.2	0.00001	1024	0.1589
0.0	0.00005	1024	0.7168
0.0	0.000005	1024	0.4189
0.0	**0.00001**	**4096**	**0.7337**
0.2	0.00001	1024	0.5849
0.7	0.00001	1024	0.1042

TABLE I
STCNN HYPERPARAMETER TUNING ON SPEAKER 1.

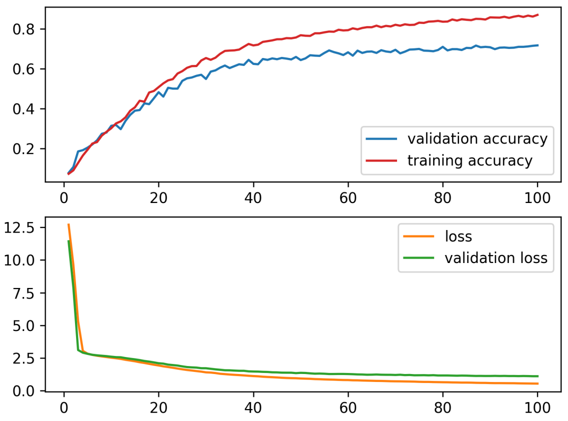

Fig. 5. Training curves of for the STCNN model, with 0.0 dropout, 0.00001 learninng rate and 4096 FC layer size. Early stopping was not used.

Dropout	Learning Rate	FC Size	Accuracy
0.2	0.00005	4096	0.7076
0.5	0.00005	4096	0.6908
0.7	0.00005	4096	0.6217
0.0	**0.00005**	**1024**	**0.7621**
0.0	0.0001	4096	0.5509

TABLE II
CNN + GRU HYPERPARAMETER TUNING ON SPEAKER 1.

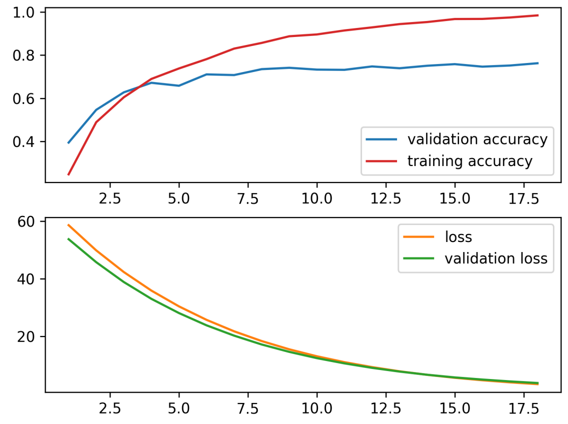

Fig. 6. Training curves of for the CNN+GRU model, with 0.0 dropout, 0.0001 learninng rate and 4096 FC layer size, with early stopping.

Dropout	Learning Rate	FC Size	RNN Type	Accuracy
0.2	0.00005	4096	GRU	0.7574
0.2	0.00005	4096	LSTM	0.6967
0.2	**0.0001**	**4096**	**GRU**	**0.7737**
0.2	0.00005	1024	GRU	0.6367
0.0	0.0001	4096	GRU	0.7650
0.5	0.0001	4096	GRU	0.7531
0.7	0.0001	4096	GRU	0.7675

TABLE III
VGG16 + GRU HYPERPARAMETER TUNING ON SPEAKER 1.

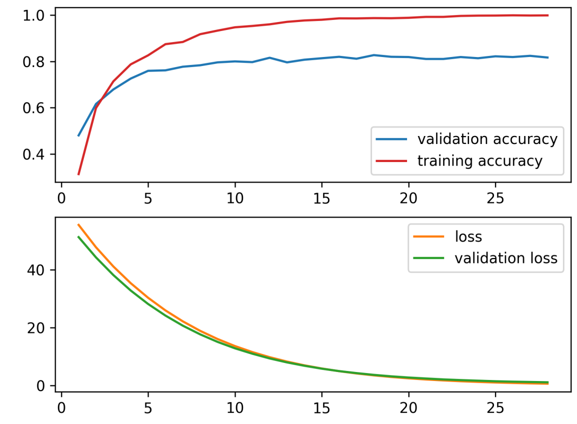

Fig. 7. Training curves of for the VGG16+GRU model, with 0.0 dropout, 0.0001 learninng rate and 4096 FC layer size, with early stopping.

The end-to-end trainable CNN + GRU solution was the second best model, also showing some sensitivity to the fully connected layer size as well as the learning rate. In addition, we tried to use an LSTM recurrent unit instead of a GRU, which not only increased the number of parameters to train, but also achieved worse performance. The best end-to-end trainable model achieved 76% test accuracy for speaker 1.

Finally using the pretrained VGG16 to encode the images to feature maps achieved the best performance at 77%. It was the most consistent to hyperparameter variations, which can be attributed to the fact that the majority of the parameters were frozen and only the GRU was learning during the training process. After we compared all the models on speaker 1, we trained the best one (VGG16 + GRU) on speakers 1-19 (Figure IV), in order to compare it with the system proposed in *"Lip Reading with Short Long Term Memory"*[6]. Ideally we would like to train all the models on all the speakers but we were limited by resources. The STCNN had 116,142,771 trainable parameters, and the CNN + GRU end-to-end trainable model had 85,911,915 parameters. Surprisingly the VGG16+GRU model can achieve better performance with only 1,388,544 trainable parameters, thus making it the most efficient model as well.

Our results indicate that our VGG16 + GRU model performed significantly better than the one proposed by M. Wand et al. in *"Lipreading with Long Short Term Memory"* [6],

Speaker	Accuracy
1	0.7737
2	0.8563
3	0.8652
4	0.8519
5	0.8324
6	0.8739
7	0.8754
8	0.8492
9	0.7875
10	0.7915
11	0.7906
12	0.9066
13	0.7006
14	0.7783
15	0.7614
16	0.8031
17	0.8808
19	0.8041
Mean Accuracy	**0.8213**

TABLE IV
VGG16 + GRU ACCURACY ON ALL SPEAKERS.

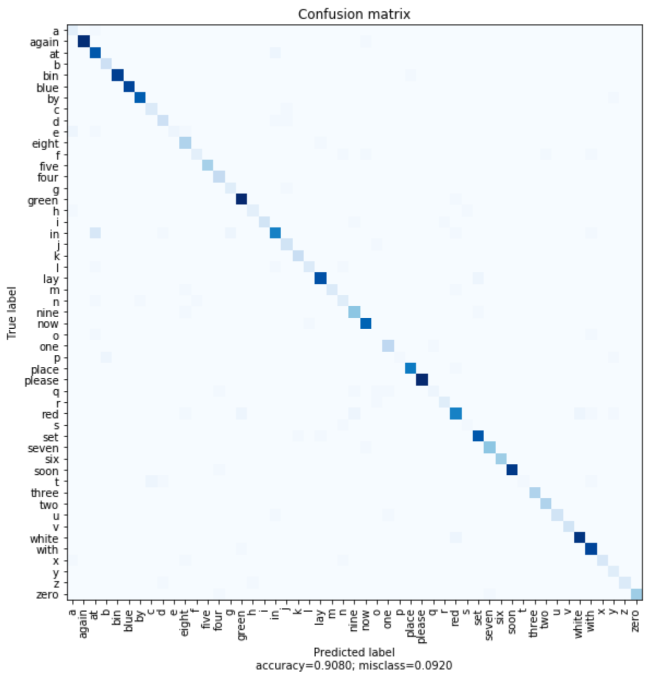

Fig. 8. Confusion Matrix for VGG16 + GRU.

which had an average accuracy of 79% for speakers 1-19. In fact, our model was not tuned for every speaker independently, but rather used the same setup for all speakers. This was not true for M. Wand et al. who tuned their model for each speaker.

VI. Analysis

A. Confusion Matrix

In order to assess the performance of our best performing model (VGG16 + GRU), we include a breakdown of the word predictions for speaker 1 that can be seen in its confusion matrix in Figure 8.

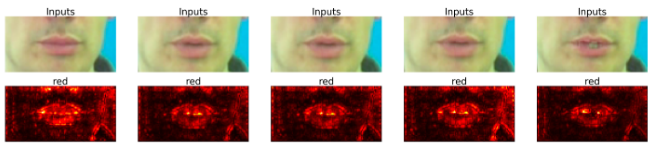

Fig. 9. Saliency Maps of VGG16 + GRU for word “red”. Only first 5 frames of video included.

The confusion matrix indicates that the classifier performs well in classifying the words uttered at each video sequence, but it also reveals that there are some words that are underrepresented in the dataset. This is one limitation of our project. Ideally we would like to re-balance all the classes in our dataset. Another limitation is that we did not analyze the performance of the model on words and letters separately. The intuition behind this is that words are harder to classify than letters, because they contain more temporal information in the video and require the model to understand deeper structure in the data.

B. Saliency Maps

We interpret our model’s learned behavior by applying saliency visualizations. These are obtained by using guided backpropagation by computing the gradient of output category with respect to input video. This should tell us how the output word prediction changes with respect to a small change in the inputs. We can use these gradients to highlight input regions that cause the most change in the output. Intuitively this should highlight salient image regions that most contribute towards the prediction of the output word. The saliency map for word “red” in Figure 9 shows that our model correctly localizes on the phonologically regions around speaker’s mouth instead of other regions in the video. In addition to that, this result proves that VGG16 can generalize well enough that it can extract face or mouth features without any additional training.

C. Class Visualization

One way to represent class visualization is to find an input image that best activates a specified class, namely, the corresponding neuron in “Softmax” layer. This method is referred as Activation Maximization. Literally, it maximizes the softmax score of the chosen “class” by performing gradient ascent onto the noisy input image. In practice, we also add L2 regularization to prevent the image intensity from blowing up. During our visualization, instead of initializing the input frames as random numbers, we perform gradient ascent directly onto 21 frames of identical silent images (extracted from silent moment, all closed mouth).

After considerable trials on fine tuning hyperparameters, we eventually achieve stable visualization on all the output classes. Figure 10 and Figure 11 are two representative results for class “set” and “red”, respectively. The first row shows the image representation of a word, and the second row represents our visualization, generated from silent images. As is shown in “set” class visualization (Figure 10), the mouths in first four images are forced opened after iterations, and occasionally

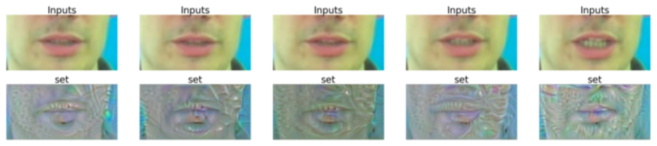

Fig. 10. Class Visualization for word “set” in VGG16 + GRU. Only the first 5 frames of video are included.

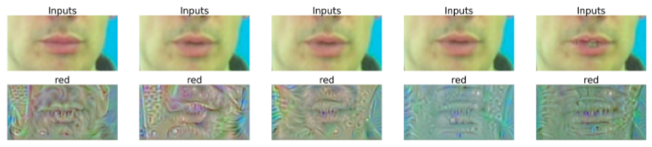

Fig. 11. Class Visualization for word “red” in VGG16 + GRU. Only the first 5 frames of video are included.

there is even ambiguity to interpret (the first image can be both opened and closed in different perspectives). This implies that our neural network indeed extracts complex but reasonable features from the input samples.

VII. Tools

We choose to use Keras to build our models. Keras offers simple APIs that are easy to use and provides clear and actionable feedback upon errors. In addition, Keras models can be trained on different hardware platforms beyond CPUs. Keras has built-in support for multi-GPU data parallelism and models can be trained on clusters of GPUs, so that we could train our models on UC Berkeley’s Savio cluster. This allowed us to train our models on large memory CPU nodes up to 512GB, and GPU nodes, which provided NVIDIA’s GTX 1080ti, significantly improving our experiments.

For our VGG16 + GRU model, we replaced our custom CNN with VGG16 model from Keras, with 3×3 convolution filters and 16 weight layers pre-trained on ImageNet. We believe the pre-trained VGG16 can achieve excellent performance in terms of feature extraction and provide valuable insights by comparing to our custom CNN.

VIII. Conclusions and Future Work

We investigate three competing architectures for the task of lipreading. The first, a spatio-temporal convolutional network had a top-one accuracy rate of 73.37%, but was sensitive to any dropout. A convolutional network with recurrent units had a top-one accuracy rate of 76.21%. Unlike other architectures the CNN+GRU performed better even with far fewer parameters because of a small fully connected head. The small fully connected head likely reduces the amount of overfitting during training. Finally, the VGG16+GRU model performed slightly better than the CNN+GRU model, with a top-one accuracy rate of 77.37%. The VGG16+GRU model performed well across all speakers, indicating that the model can successfully generalize the features for words spoken, rather than a single speaker’s mannerisms. From this the takeaway is that using a pretrained CNN as an encoder of images can be more efficient, since it has much less trainable parameters and perform better than all the other models investigated.

The performance of the STCNN was several points lower than both recurrent networks. The STCNN only had convolutions that considered three frames or seven frames of video. Since recurrent units can consider time series information of arbitrary length, they are likely more robust to words of various lengths and speakers with various cadences. Further research could investigate how well a similar STCNN architecture performs with words of wildly different lengths to further determine whether STCNNs can compete with the robustness of recurrent networks. Furthermore, STCNNs with more heads would be more robust to time domain information. It may be helpful to explore the tractability and performance of much larger STCNNs how well a scaled up model to determine the upper bound performance of models with this architecture. In general, this points out the weakness of CNNs versus recurrent networks in performing tasks in the time domain. The difference in performance can be attributed to both the flexibility of RNNs but also in the inherent structure of the GRU, which is highly optimized for recurrent tasks, unlike the spatio-temporal convolutional layer.

Due to the time and computation limitation, we only conducted experiments on seen people without unseen speakers, which means same speakers’ data are used in training, validation and testing. By employing a split for unseen speakers or overlapped speakers, we could further evaluate and compare our models’ performance.

The preprocessing step for our videos involved cropping the length of each video to include just a single word said, then zero-padding it to standardize input sizes. This means that the network may be identifying how much zero padding is in a video to help identify a word. This feature is not present in regular lip-reading environments. The GRID corpus has only three words that are two-syllables, and no words that are more than two syllables. These longer words can likely be identified from just the padding alone. Testing these network architectures with a different dataset, with both longer words, and videos of uniform length could further investigate how important these features are to the models tested.

Another suggestion on future work would be an exploration on even more sophisticated network, such as attention network, or training with Connectionist Temporal Classification (CTC) loss. On one hand, as mentioned above, our input frames are uneven and we use zero padding on short frames. Attention network might address this unbalanced issue by weighing out salient feature frames. On the other hand, instead of taking the word as a whole, we can decompose a single word into separate phonemes. The intuition of CTC loss is to classify each frame into a phoneme and then decode the combined phonemes into a real word in the dictionary. However, this approach is harder to train due to its model complexity and reliance on powerful computational resources.

IX. Team Contributions

Member Name	Worked on	Percentage
Jing	CNN+GRU, Preprocessing, Report	19%
Avery	STCNN, Poster, Report	16%
Alex	STCNN, CNN+GRU, VGG16+GRU, Results, Preprocessing, Poster, Presentation, Report	40%
Qingyang	CNN + GRU, Preprocessing, Visualizations (saliency maps, class visualizations, confusion matrix), Report	25%

TABLE V
Team Contibutions.

X. Link the the Github repository

https://github.com/danisbet/machine-lip-reading

References

[1] Rekik A., Ben-Hamadou A., Mahdi W. (2015) *Human Machine Interaction via Visual Speech Spotting.* In: Battiato S., Blanc-Talon J., Gallo G., Philips W., Popescu D., Scheunders P. (eds) Advanced Concepts for Intelligent Vision Systems. Lecture Notes in Computer Science, vol 9386. Springer, Cham

[2] Yuru Pei, Tae-Kyun Kim, and Hongbin Zha. *Unsupervised random forest manifold alignment for lipreading.* In The IEEE International Conference on Computer Vision (ICCV), December 2013.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 10971105. Curran Associates, Inc., 2012.

[4] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, *Audio-visual speech recognition using deep learning.* Applied Intelligence, vol. 42, no. 4, pp. 722 737, 2015.

[5] S. Petridis and M.Pantic, *Deep complementary bottleneck features for visual speech recognition.* In IEEE ICASSP, 2016, pp. 23042308.

[6] M. Wand, J. Koutnk and J. Schmidhuber. *Lipreading with Long Short-Term Memory.* arXiv preprint arXiv:1601.08188

[7] Assael, Yannis M., et al. *LipNet: End-To-End Sentence-Level Lipreading.* arXiv preprint arXiv:1611.01599 (2016).

[8] A. Gutierrez and Z. Robert. *Lip Reading Word Classification.* http://cs231n.stanford.edu/reports/2017/pdfs/227.pdf.

[9] Chung, Joon Son, et al. *Lip reading in the wild.* Asian Conference on Computer Vision. Springer, Cham, 2016.