Lip Reading Word Classification

Abiel Gutierrez
Stanford University
abielg@stanford.edu

Zoe-Alanah Robert
Stanford University
Zrobert7@stanford.edu

Abstract

We present a variety of models and methods for predicting words from video data without audio. Previous work exists in this subject area, but it is limited and very recent. In this paper we use the MIRACL-V1 dataset [0] containing videos of ten people speaking ten words. We pre-process the data by using existing facial recognition software to detect and crop around the subject's face in all frames of the video and then use the sequence of frames as input to the model. We explore a CNN + LSTM Baseline model, a Deep Layered CNN + LSTM model, an ImageNet Pretrained VGG-16 Features + LSTM model, and a Fine-Tuned VGG-16 + LSTM model. This paper discusses the effects of dropout, hyperparameter tuning, data augmentation, seen vs unseen validation splits, batch normalization, and other techniques for adjusting these models. We achieve a validation accuracy of 79% and a test accuracy of 59% on our best model.

1. Introduction

This paper investigates the task of speech recognition from video without audio. We present several neural network models with varying successes in this classification task. The input data to our algorithm is sequences of still images taken from frames of video footage. We use different neural network models to output one of 10 words that are spoken (or mouthed) by a face in the input images. We explore and combine a number of different models including CNNs, RNNs, and existing publically available pre-trained networks to assist in mouth recognition.

This is an interesting learning task given that video traffic is growing at a high rate throughout the web, and this model could help extract data and process it to gain insights into the action or topics occurring in a video. Applications of a visual audio classifier range from play prediction in sporting events to profanity detection on social media sites to a live action lip reading mobile application.

In the past, research efforts have been far more focused on gesture recognition rather than visual speech recognition, making this for a new and exciting field to explore. There are a few existing systems and applications for lip reading, although most do not use neural networks but instead other machine learning techniques. More advanced visual speech recognition models such as Google's DeepMind LipNet [1] network were published only a few months ago.

2. Related Work

In this section, we outline the existing work that has been done in the field. As previously mentioned, most approaches have involved machine learning methods that do not touch on deep learning. It has only been until very recently that deep learning methods have emerged and produced state-of-the-art results.

Pei et al [2] used Random Forest Manifold Alignment for this same task, extracting patched trajectories spatiotemporally and then mapping these features to motion patterns. Rekik et al [3] used Hidden Markov Models to solve this problem with color and depth representations of images. They extracted a 3D rendition of the speaker's mouth, and generated a variety of features from it. They obtained a 62.1% classification accuracy using HMMs on the MIRACL-V1, performing speaker independent testing.

One of the first works to use deep learning in speech recognition was Hinton et al.[4], where neural networks were used for acoustic processing. Other approaches include learning multimodal audio-visual representations [5, 6] and learning visual features to then apply to more traditional classifier structures like HMMs [7]. Some works have gone beyond word-level: Noda et al. [8] used CNNs to predict phonemes, and Shaikh et al. [9] trained an SVM to predict visemes. Koller et al. [10] also detected visemes using an image classifier CNN. More generally, the work by Graves et al. [11] has been considered critical for the development end-to-end deep speech recognition systems thanks to their development of the connectionist temporal classification loss (CTC), which allows for spatiotemporal CNNs.

Recently, Wand et al. [12] introduced LSTMs for lip reading at the word level, which we decided to use in this paper. Chung & Zisserman [13] made use of the work of Graves et al. by using spatiotemporal CNNs for word classification on the BBC TV dataset. Assael et al. [1] created LipNET, a phrase predictor that uses spatiotemporal convolutions and bidirectional GRUs and achieved a 11.4% WER on unseen speakers. Our model is primarily inspired by this work. We also took inspiration from Garg et al. [14], where a pre-trained VGG was used for transfer learning on the MIRACL-V1 dataset. A much more comprehensive list of lip reading works can be found in Zhou et al. [15].

3. Dataset and Features

We used the MIRACL-VC1 data set [0] containing both depth and color images of fifteen speakers uttering ten words and ten phrases, ten times each. The sequence of images represents low quality video frames. The data set contains 3000 sequences of varying lengths of images of 640 x 480 pixels, in both color and depth representations, collected at 15 frames per second. The lengths of these sequences range from 4 to 27 image frames. The words and phrases are as follows:

Words: *begin, choose, connection, navigation, next, previous, start, stop, hello, web*
Phrases: *Stop navigation, Excuse me, I am sorry, Thank you, Good bye, I love this game, Nice to meet you, You are welcome, How are you, Have a good time*

For the sake of time and utilizing smaller data sizes, we focused on building a classifier that can identify which word is being uttered from a sequence of images of the speaker as input. We ignored the set of phrase data and also the depth images for the spoken word data. We built classifiers for both seen and unseen people. (Seen meaning that the model is trained on all people saying all words but saves certain trials for test and validation. Unseen removes people from training and adds them to exclusively to either testing or validation. The split is thirteen people for train, one for validation, and one for test.) The resulting datasets are (1200/150/150) (train/test/validation) examples for seen and (1300/100/100) (train/test/validation) examples for unseen. The class label distribution for the dataset is even as each person performs the same number of trials per word.

Preprocessing was an important part of working with this dataset. First, we utilized a python facial recognition library, dlib, in conjunction with OpenCV and a pre-trained model [2] to isolate the points of facial structure in each image and crop it to only include the face of the speaker, excluding any background that could interfere with the training of the model. We had to limit the size of every facial crop to a 90x90 pixel square in order to create uniform input data sequences for the model.

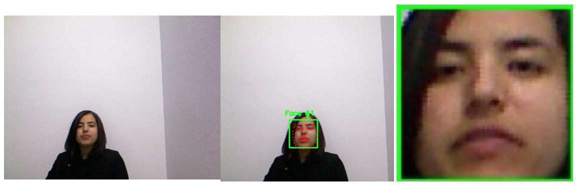


Figure 1: (left to right) Original Input image (part of a sequence) in the MIRACL-VC dataset; OpenCV and dlib facial recognition software labelling key points on around a detected face; final cropped image

One issue with this data set is its small size. To increase the number of training sequences, we performed data augmentation. We tripled the data set in size by adding a horizontally flipped version of each image and a randomly pixel-jittered version of each image.

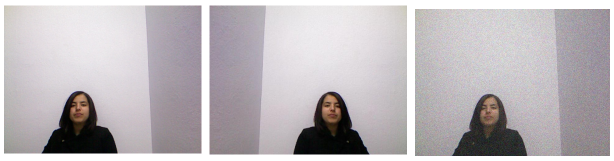


Figure 2: (left to right) Original Input image (part of a sequence) in the MIRACL-VC dataset; a horizontally flipped image; a jittered image.

In summary, each model receives a single image sequence as input – with anywhere from 4 to 27 images in the sequence – and produces a single word classification label as output.

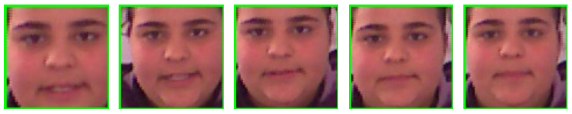


Figure 3: Example full input sequence of length 5. The subject is speaking "begin."

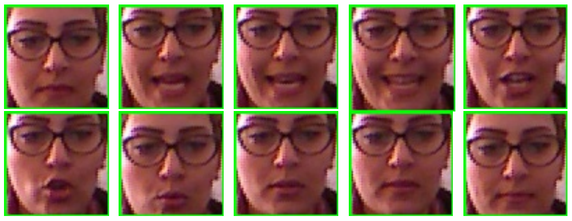


Figure 4: Example full input sequence of length 10. The subject is speaking "hello."

4. Methods

In this section we describe the different models that we created to solve the lip reading problem. We created four models: a Baseline CNN + LSTM network; a more robust and deep layered CNN + LSTM network inspired by Deep Mind's LipNET[1]; an LSTM network placed on top of bottleneck features developed by a VGG16 network pre-trained on ImageNet; and the same LSTM network on top of VGG16 with fine-tuning of the last convolutional block.

4.1 CNN + LSTM Baseline

Our first model ran every image of our sequenced input through a Convolutional Neural Network and then fed the flattened outputs as a sequence into a Long Short Term Memory Recurrent Neural Network, which produced a single output, making it a many-to-one RNN. We then added a Fully Connected layer that mapped to 10 units, and used a softmax activation layer to produce the probabilities of every word, of which we took the highest:

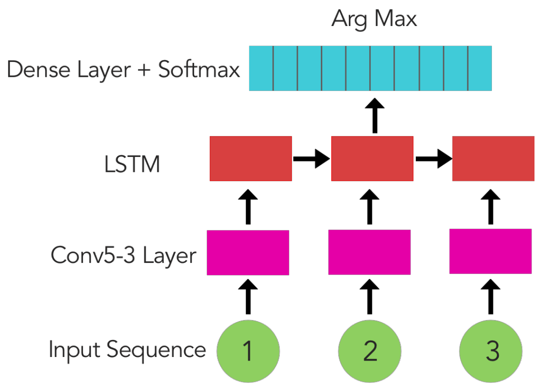


Figure 5: CNN + LSTM Baseline model layer architecture diagram

Our convolutional layer had a kernel size of 5x5 and depth of 3 filters -- inspired from LipNET's architecture [1] -- and was added to help the model make sense of the high-level features of the images. It achieves this by running the kernel across the image, mapping the dot products of the pixel overlaps to a new layer, and stacking together the layers produced by every filter:

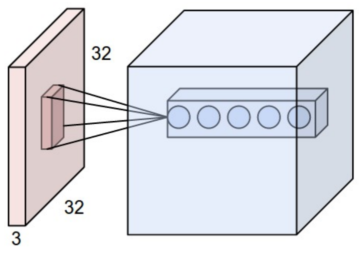


Figure 6: Structural diagram courtesy of CS231N at Stanford - http://cs231n.github.io/convolutional-networks/

The LSTM was added to package the entire sequence of CNN outputs into a single layer without losing the temporal understanding of the video frames. In particular, an LSTM fixes the vanishing gradient problem present in vanilla RNNs, which inhibits the backpropagation of gradients to occur [16]. It does so by adding 4 gates (input (i), forget (f), output (o), new memory (c)) whose activations can be learned, in order to control whether or not to hold on to information:

$$
\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) && \text{(Input gate)}\\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) && \text{(Forget gate)}\\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) && \text{(Output/Exposure gate)}\\
\tilde{c}_t &= \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) && \text{(New memory cell)}\\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t && \text{(Final memory cell)}\\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

Given that we use softmax as our last activation, our loss function is cross entropy loss:

$$loss = -\sum_i log\left(\frac{exp(Wx_i)}{\sum_j exp(Wx_j)}\right)$$

Finally, we used the Adam Optimizer to better navigate through the loss function.

4.2 Deep Layered CNN + LSTM

We expanded on our baseline by first adding 2 more layers of CNNs, in order to develop an understanding of more intricate features in our input images. We made our LSTM bidirectional, to avoid outweighing the output with frames in the latter parts of the sequence, and added dropout and batch normalization after every CNN layer. We kept our dropout probability at 0.2 given that we performed it several times across the model. We also interspersed 2x2 Max Pooling layers with strides of 2 between the CNNs. This model is even more similar to LipNET's [1]:

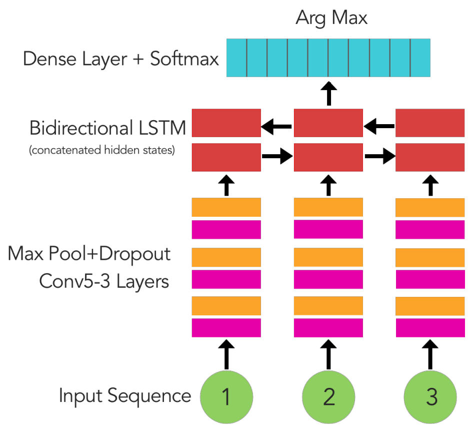


Figure 7: Deep Layered CNN + LSTM model layer architecture diagram

4.3 ImageNet Pretrained VGG-16 Features + LSTM

Given that we decided to focus only on words and not phrases, we were limited to 1500 data points (15 people uttering 10 words for 10 iterations). This is a really small dataset by deep learning standards; as a result, we decided to employ transfer learning by making use of a VGG-16 network pre-trained on ImageNet[17]. VGG networks [18] were developed under the premise that a smaller kernel size allows for deeper networks without increasing the number of parameters, thus increasing the number of non-linearities without the need of greater memory usage.

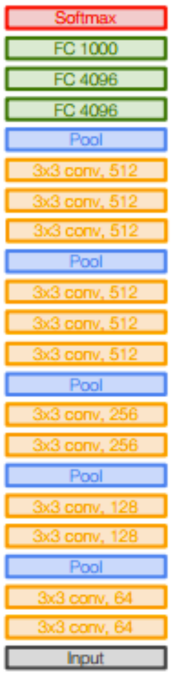


Figure 8: VGG-16 Structure

ImageNet is a dataset currently consisting of more than 14 million images, some of which contain annotations and object bounding boxes. 952,000 of those images are of humans, reason for which this network could be utilized for the lip reading problem [17].

We extracted the bottleneck features of our dataset up until the last convolutional layer of VGG. We then fed this result into the top portion of our second model -- the Bidirectional LSTM, and performed the training. This meant that we froze the VGG model weights, and only updated the weights of our LSTM and Dense layer. This sped up training considerably, given that our data was only processed by the VGG network once per training trial.

4.4 Fine-tuned VGG-16 + LSTM

Our last model consisted of unfreezing the last convolutional layer block of the pretrained VGG-16 model (3 CNN layers with 512 3x3 kernels) and training it alongside the Bidirectional LSTM and Dense Layers that we placed on top. This was done with the intention of capturing the more complex features of our input images, since later CNN layers tend to capture less obvious characteristics of an image. Given that the unfrozen convolutional layer was initialized with pretrained ImageNet weights, we also initialized the LSTM and Dense layers with the weights of the top portion of our third model (the LSTM and Dense layer training on top of the VGG-16 bottleneck features), to prevent a random weight initialization from recklessly modifying the VGG weights when backpropagating. We also prevented this by switching to a regular SGD optimizer rather than Adam, since SGD is subtler in updating weights.

5. Results & Discussion

We used accuracy as our primary metric, although we also looked at the confusion matrices of these models to better understand where the errors were occurring. With 10 classes, a random baseline for this classifier is 0.1. All four of our models outperformed this baseline, with the Fine-tuned VGG-16 + LSTM achieving top test accuracy. We tested on both seen and unseen subjects. Results for seen subjects were relatively good, but our accuracy for unseen subjects gravitated barely above the random choice metric of 10% for all ten models.

Model	**Training**	**Validation**	**Test**
Baseline	85%	64%	39%
Deep CNN + LSTM	52%	39%	25%
Frozen VGG + LSTM	100%	76%	55%
Fine-tuned VGG + LSTM	100%	79%	59%

Figure 9: Seen Subject Accuracy

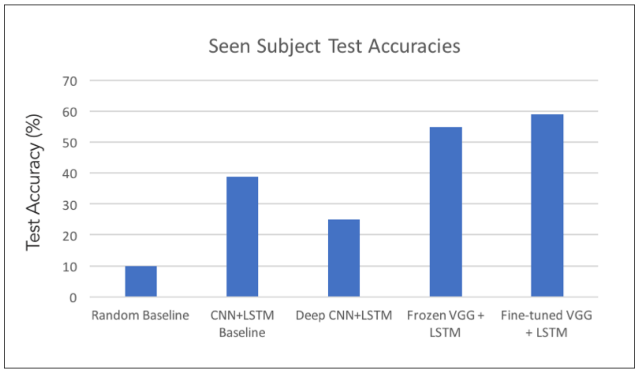


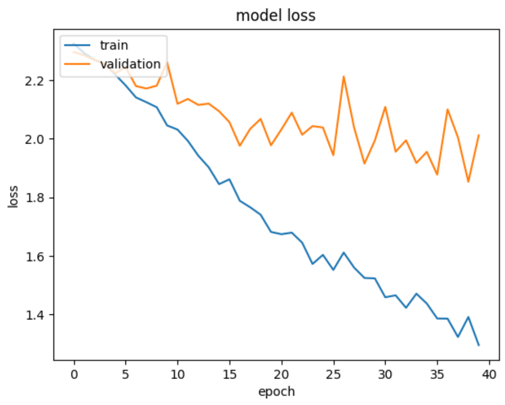
Figure 10: Seen Subject Accuracy Comparison Graph

5.1 CNN + LSTM Baseline

Our baseline got surprisingly good results for seen subjects, with a 39% accuracy for the test set. As a result, we suspect that it is relatively easy to distinguish between words uttered if the word and the subject have been seen multiple times before. This would mean that there's several high level features that make this distinction easy -- reason for which only one CNN layer was required to obtain decent results.

5.2 Deep Layered CNN + LSTM

Our deep layered model performed worse than our baseline, with a test accuracy of 25%, significantly lower than validation (39%) and training (52%).



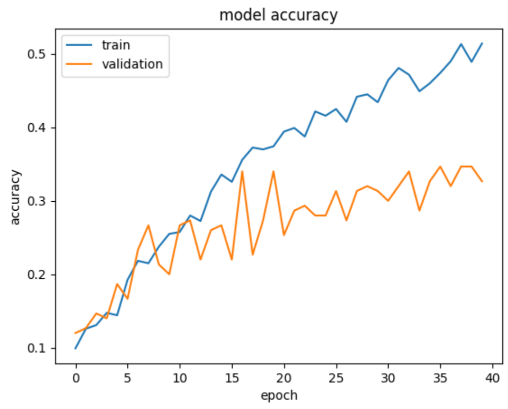


Figure 11: Loss and Accuracy Plots for Deep-Layered CNN + LSTM

Building up on the analysis made for the baseline, it seems like classifying on words spoken by seen subjects is not complex, and thus having the complex representations outputted by our three layered convolutional network was unnecessary. Another distinction between this model and the baseline was the addition of dropout and batch normalization. Again, it seems that regularization wasn't necessary for seen subjects, so it harmed this model's performance. The jagged loss plot below suggests that the model was having trouble navigating the validation loss space, which might suggest that it did not have a good intuition for what features to look for when making predictions.

4.3 ImageNet Pretrained VGG-16 Features + LSTM

This model achieved 100% training accuracy, which is probably due to the fact that we added the frozen VGG-16 model pre-trained on ImageNet. The learning rate used for this model was 0.00006, which is dramatically smaller than that used in the above two models (0.0001 and 0.001, respectively). Again, this was probably due to having pre-trained weights, although it's somewhat intriguing because the VGG weights were frozen, and the LSTM and Dense layer weights were randomly initialized. This might suggest that the representations outputted by the VGG were much more significant than the work done by the LSTM and Dense layer that we added on top, so the model worked best when we mitigated the effect of the top model by reducing the learning rate.

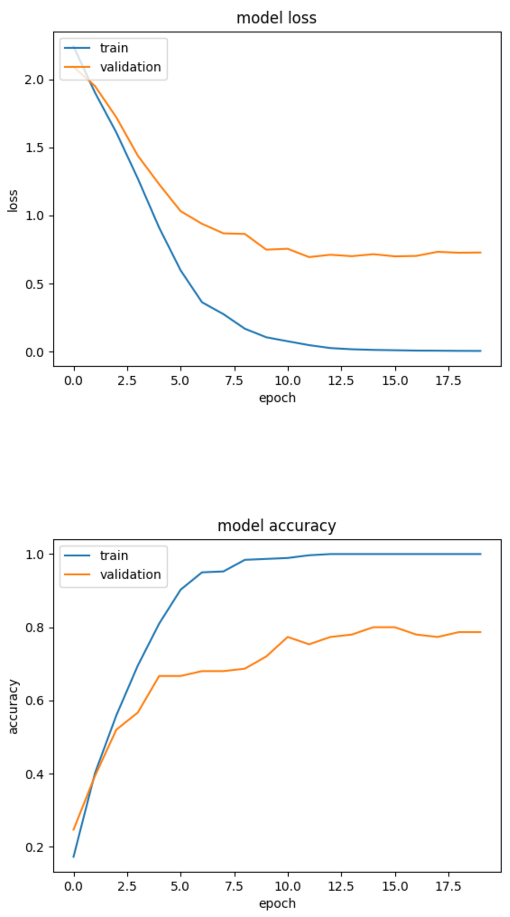


Figure 12: Loss and Accuracy Plots for ImageNet Pretrained VGG-16 Features + LSTM

The accuracy plot suggests that more regularization could have lifted our validation and test scores. However, we found that increasing the dropout rate made the model worse; we suspect that the short length of our dataset was what caused this behavior. We thus decided to stick to a 0.2 dropout probability.

5.4 Fine-tuned VGG-16 + LSTM

The fine-tuned VGG-16 + LSTM gave us the best test results, with 59% test accuracy. A breakdown of predictions can be seen in its confusion matrix in Figure 13.

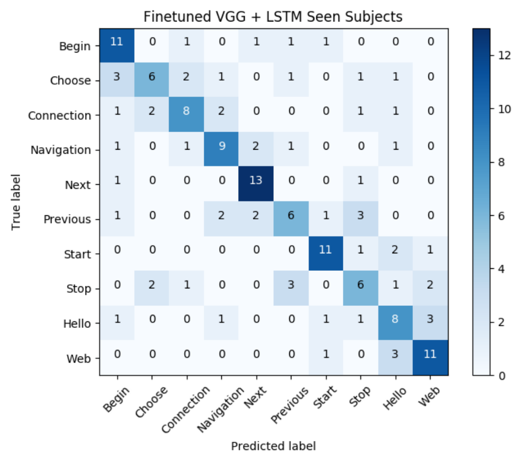


Figure 13: Confusion Matrix for VGG-16 + LSTM

This model was only trained for 5 epochs with a learning rate of 0.0001 -- it's validation accuracy shot up to 79% immediately and stayed there, with it's validation loss also staying constantly at slightly above 0.7. It's training loss was almost 0 from the start for training. This behavior is due to the pre-trained weights that were added to the LSTM and Dense layer, which were generated in the previous model. As a result, the only difference between both models was the fine-tuning that occurred in the last 3-layered convolutional block of the VGG-16 model, which helped us improve testing accuracy by 5%.

5.5 Unseen Data

Upon further analysis, we found some possible explanations for our bad results in unseen validation and test data (which again, wavered slightly above 10%). This is the confusion matrix for our second model, the Deep CNN + LSTM on unseen subjects:

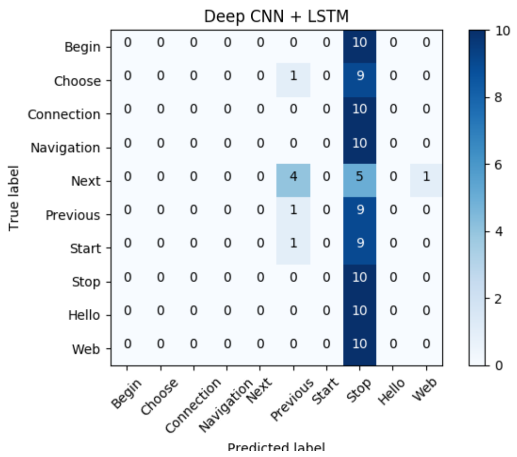


Figure 14: Confusion Matrix for Deep CNN + LSTM

Our model is predicting "Stop" for 92% of the words. We realized cross-validation could have helped mitigate this issue; a possible explanation for this result is that the person in the test set spoke faster than any other subject, and as a result, most of the words uttered by the subject are thought to be "stop", since "stop" has perhaps the shortest pronunciation within the dataset.

The confusion matrix of unseen subject for our fine-tuned VGG model also suggests a correlation between short words:

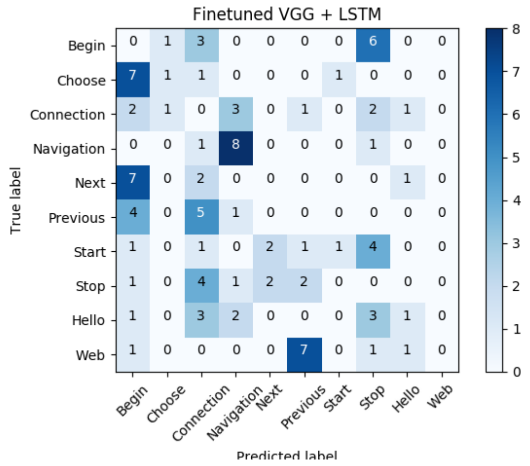


Figure 15: Confusion Matrix for VGG + LSTM

"Begin" is predicted for "Choose" and "Next" a total of fourteen times. But it's also interesting to note that "Web", "Choose", and "Start" were predicted a total of only five times; this might suggest that the model chooses only a few words to focus its predictions on -- again probably because of our limited data set. In the future, a bigger data set, cross validation, and data augmentation could help us improve our score for unseen subjects.

One way we attempted to improve results for unseen subjects was with data augmentation. We saw with the ImageNet Pretrained VGG-16 Features + LSTM, that although validation accuracy is still abysmal, the addition of randomly augmented data (flipped and jittered) improved validation accuracy by a factor of two. Unfortunately, time did not permit to run our other models with the augmented data, so we decided to exclude its use from our main analysis. It is, however, a vital addition that could help improve scores for this task when working with limited datasets in the future.

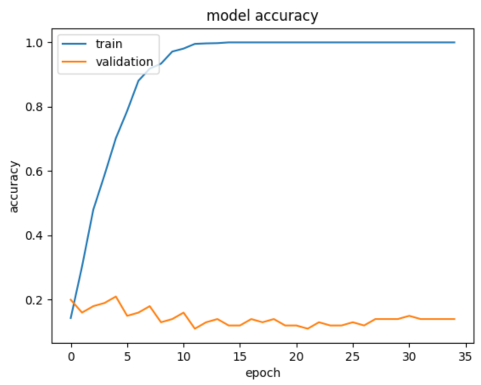


Figure 16: The ImageNet Pretrained VGG-16 Features + LSTM without data augmentation achieving an accuracy of 0.14

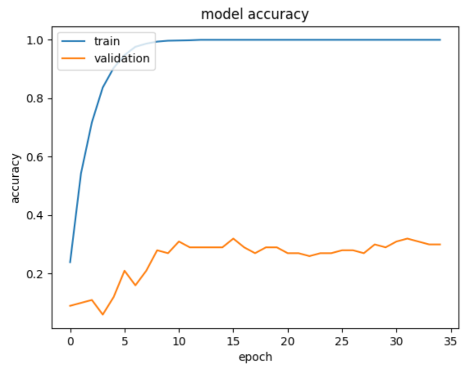


Figure 17: The ImageNet Pretrained VGG-16 Features + LSTM with data augmentation achieving an accuracy of 0.3

6. Conclusion & Future Work

Overall, we found that the inclusion of pre-trained facial recognition CNNs highly improved our models. The augmentation of our data proved helpful but only in the instance of unseen people. Our best model was the Fine-Tuned VGG + LSTM.

Regarding failures of specific models, the Deep Layered CNN + LSTM architecture was inspired by the LipNet architecture [1], which was designed to handle phrase inputs and is trained on a much larger corpus. Perhaps this explains why our baseline outperforms the Deep Layered CNN + LSTM architecture. Generally, in all models, we found it very difficult to avoid overfitting with unseen people. Thus, certain models and hyperparameters are a better fit depending on whether we are working with seen or unseen people for testing and validation. More work needs to be done to reduce overfitting even for seen people for the models that include pretrained networks. These reached training accuracies of 1 while validation accuracy remained close to .75.

Given more time and resources, the models outlined in this paper could be greatly improved. We think the addition of regularization would reduce the overfitting in our models even further. We also have yet to experiment with the number of filters in the fully connected layers. We only had 3 filters per layer, just like LipNET had, but other papers used anywhere from 64 to 512 filters per single CNN layer. Additionally, accuracy improvements could be found with further hyperparameter tuning and investigation of even more optimizer types. We also would have gotten value from saliency maps. Without them it is hard to know if the model is accurately focusing on mouth data or other aspects of the input sequences. Finally, performing analysis of confusion matrices earlier in our exploration process could have helped us mitigate the problems that we ran into with unseen subjects, given that we could have adjusted our models based on the patterns we perceived.

This project is easily extendible and raises the question of how to perform visual speech recognition on a much larger corpus (perhaps the entire English dictionary). How could the addition of audio data improve our ability to interpret the video as text? Is it easier to understand speech from video of a single word being spoken or entire phrases and sentences? This question could easily be investigated since the MIRACL-V1 dataset includes phrase inputs and would be an interesting area of exploration. Additionally, most speech recognition tasks in real life require phrase inputs over single words.

7. References

[0] Ahmed Rekik, Achraf Ben-Hamadou, and Walid Mahdi. A new visual speech recognition approach for RGB-D cameras. In Image Analysis and Recognition - 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part II, pages 21–28, 2014.

[1] Yannis M. Assael, Brendan Shillingford, Shimon White son, and Nando de Freitas. Lipnet: Sentence-level lipreading. CoRR, abs/1611.01599, 2016.

[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] 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

[4] 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 1097–1105. Curran Associates, Inc., 2012.

[5] C. Sui, R. Togneri, and M. Bennamoun, "Extracting deep bottleneck features for visual speech recognition," in ICASSP, 2015, pp. 1518–1522.

[6] S. Petridis and M. Pantic, "Deep complementary bottleneck features for visual speech recognition," in IEEE ICASSP, 2016, pp. 2304–2308.

[7] Y. Li, Y. Takashima, T. Takiguchi, and Y. Ariki, "Lip reading using a dynamic feature of lip images and convolutional neural networks," in IEEE/ACIS Intl. Conf. on Computer and Information Science, 2016, pp. 1–6.

[8] 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.

[9] Ayaz A. Shaikh, Dinesh K. Kumar, Wai C. Yau, M. Z. Che Azemin, and Jayavardhana Gubbi. Lip reading using optical flow and support vector machines. 2010 3rd International Congress on Image and Signal Processing, 1:327–330, 2010.

[10] Koller, O., Ney, H., Bowden, R.: Deep learning of mouth shapes for sign language. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 85–91 (2015)

[11] Graves, A., Fernandez, S., Gomez, F., and Schmidhuber, ´ J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In ICML, Pittsburgh, USA, 2006.

[12] M. Wand, J. Koutnik, and J. Schmidhuber. Lipreading with long short-term memory. In IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6115–6119, 2016.

[13] J. S. Chung and A. Zisserman. Lip reading in the wild. In Asian Conference on Computer Vision, 2016.

[14] Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. CoRR, abs/1603.04992, 2016.

[15] Zhou, Z., Hong, X., Zhao, G., Pietik¨ainen, M.: A compact representation of visual speech data using latent variables. IEEE transactions on pattern analysis and machine intelligence 36(1), 1–1 (2014)

[16] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009. http://image-net.org

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[19] JJ Allaire, Dirk Eddelbuettel, Nick Golding, and Yuan Tang (2016). tensorflow: R Interface to TensorFlow. https://github.com/rstudio/tensorflow

[20] G. Bradski. Opencv. Dr. Dobb's Journal of Software Tools, 2000.

[21] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

[22] Fran¸cois Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu

Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[24] Kotikalapudi, R. Keras Visualization Toolkit. MIT. https://github.com/raghakot/keras-vis

[25] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007),DOI:10.1109/MCSE.2007.55

[26] Davis E. King. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research 10, pp. 1755-1758, 2009

[27] "Building powerful image classification models using very little data" https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[28] "Real-time facial landmark detection with OpenCV, Python, and dlib"http://www.pyimagesearch.com/2017/04/17/real-time-facial-landmark-detection-opencv-python-dlib/