Lip reading using CNN and LSTM

Amit Garg
amit93@stanford.edu

Jonathan Noyola
jnoyola@stanford.edu

Sameep Bagadia
sameepb@stanford.edu

Abstract

Here we present various methods to predict words and phrases from only video without any audio signal. We employ a VGGNet pre-trained on human faces of celebrities from IMDB and Google Images [1], and explore different ways of using it to handle these image sequences. The VGGNet is trained on images concatenated from multiple frames in each sequence, as well as used in conjunction with LSTMs for extracting temporal information. While the LSTM models fail to outperform other methods for a variety of reasons, the concatenated image model that uses nearest-neighbor interpolation performed well, achieving a validation accuracy of 76%.

1. Introduction

Visual lip-reading plays an important role in human-computer interaction in noisy environments where audio speech recognition may be difficult. It can also be extremely useful as a hearing aid for the hearing-impaired. However, similar to speech recognition, lip-reading systems also face several challenges due to variances in the inputs, such as with facial features, skin colors, speaking speeds, and intensities. To simplify the problem, many systems are restricted to limited numbers of phrases and speakers. To further aid in lip-reading, more visual input data can be gathered in addition to color image sequences, such as depth image sequences.

The dataset used here is a set of image sequences (i.e. low-rate videos) that each show a person speaking a word or phrase. The goal is to classify these sequences. One of the main issues that prevents older methods is that sequence lengths, and hence number of features per sequence, vary widely. Therefore, various methods for capturing temporal information were used to take all input features into account.

The first method consisted of concatenating a fixed number of images from a sequence into one larger image which can then be passed through a VGGNet (see Figure 1), which used a set of weights pre-trained on faces [1]. This packed each sequence towards the front, leaving blank spaces at the ends of shorter sequences. The second method was similar, except we used nearest-neighbor interpolation to stretch and normalize the number of images per sequence.

The third method first passed each individual image through the VGGNet to extract a set of features, and then passed each sequence of features through several LSTM layers, retrieving the classification label from the final output. This method was attempted both with freezing the VGGNet, speeding up training time, and end-to-end, taking longer but allowing the VGGNet to be trained further on this particular dataset.

In order to make the problem tractable, we formulate it as a classification problem of detecting what words or phrases are being spoken out of a fixed set of known words and phrases. Each method received a single image sequence as input, and produced a single word or phrase classification label as output.

2. Related Work

In this section, we describe the related work done in this field. Most of the work done in lip reading uses non neural network approaches. They extract various features out of the image and then use machine learning approaches like SVMs to classify what was spoken.

Rekik et al in [2] propose HMMs as a way to perform lip reading using only image and depth information. The system consists of two main blocks – feature extraction and speech recognition. The first step estimates the speakers face pose using a 3D face model, including a 3D mouth patch to detect the mouth. This is followed by motion and appearance descriptors to generate features for the model – for instance HOG (histogram of gradients). The second step segments the speech video to look for frames corresponding to the utterance. The features on these frames are fed to the HMM for classification. The dataset is the same as what we

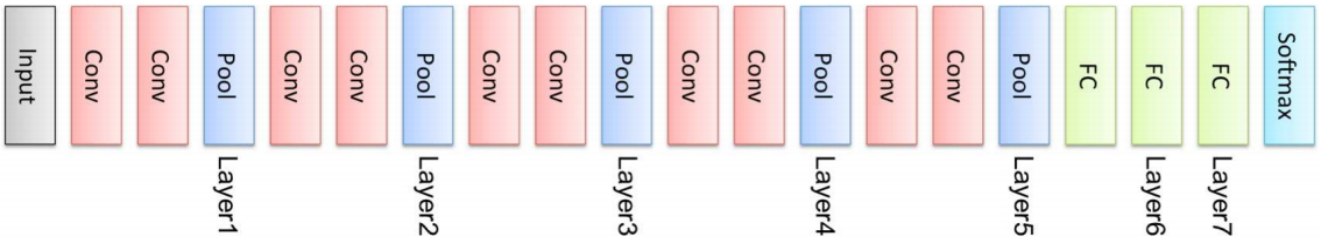

Figure 1: VGGNet

use – MIRACL-VC1 [3]. The final results for the speaker independent testing (where one speakers data is never used during training) gives 69.7% accuracy for phrases and 62.1% for words – giving an overall accuracy of 65.9%.

Rekik et al in [4] proposes a four step method for attempting the task of lip reading – 3D face pose tracking, mouth region extraction, feature computation and classification using SVM. In addition to the 3D images, they also make use of the depth information available in the dataset. The dataset comprises of MIRACL-VC1, OuluVS [5] and CUAVE [6]. An interesting detail in their implementation is the system that performs speaker identification thereby allowing the algorithm to learn different models based on different speakers. They achieve 79.2% accuracy on phrases and 63.1% accuracy on words, thereby giving an overall accuracy of 71.15% on the entire dataset – on the MIRACL-VC1 dataset.

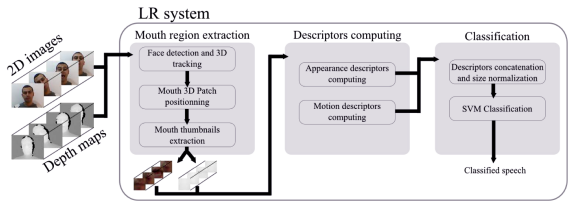

Figure 2: LR system using 3D face pose tracking and SVM

Rekik et al in [3] (see Figure 2) and [7] propose an adaptive lip-reading system using image and depth data. The algorithm is split into two main steps – first the mouth is extracted using 3D face pose tracking and then features are extracted and three different classifiers are used to get three different results – HMM, SVM and kNN. The algorithm is evaluated on three different datasets – MIRACL-VC1, OuluVS and CUAVE. SVMs are observed to be the best of the three giving an overall accuracy of 71.15%, with HMMs at 65.35% and kNNs at 50.85% when one of the speaker's entire data is held out for validation/testing. They employ a variety of features to input to their classifier – for instance, HOG (histogram of gradients) and MBH (motion boundary histograms).

Pei et al also use a non deep learning method for lip reading in [8]. They use Random Forest Manifold Alignment for training. They test their model on various lip reading data sets and compare their results to different approaches.

In [9], Ngiam et al use deep learning approaches to understand speech using both audio as well as video information. They show how using multiple modes assist each other in better prediction. This falls in line with one of our motivations for lip reading. Their architecture consists of using Restricted Boltzmann machines (RBMs) for each of audio and video modalities separately and then combining them. Their video only accuracy for two of the data sets they use are 64.4% and 68.7%.

Other methods take a different approach by instead recognizing phonemes and visemes, the smallest visually distinguishable facial movements when articulating a phoneme. Shaikh et al in [10] use vertical optical flow to train an SVM to predict visemes, a smaller set of classes than phonemes. Other features of their method include removing zero-energy frames and using interpolation to normalize the number of frames. While they achieve a high accuracy of 95.9%, a complex language model must be placed on top of this model to predict words from visemes in order to complete the process of lip reading. Noda et al try a different method that uses CNNs to predict phonemes, achieving an accuracy of 58% [11]. A GMM-HMM then uses these predicted phoneme labels to predict individual words, bringing their final lip reading accuracy to 37%.

Unlike the previous approaches, Wand et al tackle the issue of variance in sequence length by using LSTM layers on top of a neural network [12]. Their results were similar for NNs and CNNs, both showing improvement over an SVM with HOG or Eigenlips features. Although they used a relatively large dataset and all of their tests were speaker-dependent i.e. the train and test data were taken from the same speaker, this suggests that LSTMs could indeed show

improvement over dedicated sequence classifiers such as HMMs.

3. Dataset

In this section we describe the dataset we use, some of its properties and the pre-processing we performed on it.

3.1. Dataset characteristics

We use the MIRACL-VC1 [3] dataset in our project. The dataset was created from 15 people who spoke each of ten words and ten phrases ten times leading to a total of $15 \times 20 \times 10 = 3000$ instances. Each instance is a sequence of color and depth images of 640×480 pixels. For example, the images for one instance is shown in Figure 3. We only use the color image and discard the depth part since to conform to the pre-trained VGGNet model. The words and phrases in the dataset are listed in Table 1

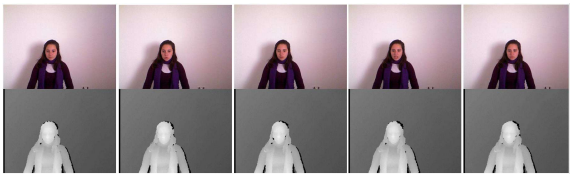

Figure 3: Example instance

Words	Phrases
Begin	Stop navigation.
Choose	Excuse me.
Connection	I am sorry.
Navigation	Thank you.
Next	Good bye.
Previous	I love this game.
Start	Nice to meet you.
Stop	You are welcome.
Hello	How are you?
Web	Have a good time.

Table 1: Words and phrases in the MIRACL-VC1 dataset

The length of sequences varies from a minimum of 4 images to a maximum of 22 images for words and the range is from 6 to 27 images for phrases. On an average word sequences have 10.33 images whereas phrase sequences have 12.85 images. Overall, there are 11.59 images per sequence. The distribution of sequence length for words and phrases is shown in Figure 4.

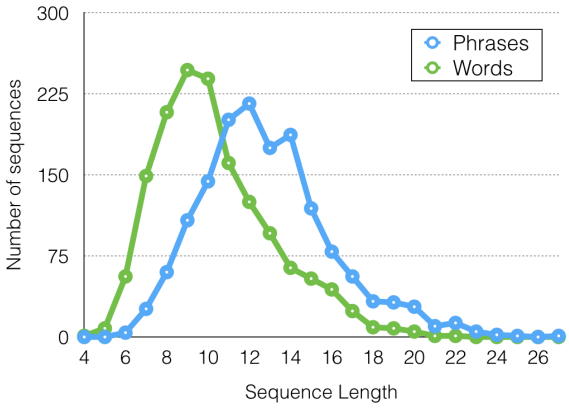

Figure 4: Sequence length distribution for words and phrases

Out of the data of 15 people, we use 13 of them for training, 1 for validation and 1 for test. Thus, we have 2600 instances in training data, 200 instances in validation set and 200 instances in test set. The objective is to achieve as high classification accuracy as possible in the test set. We would also like to analyze the accuracy separately for words and phrases.

3.2. Data Pre-processing

The data has a lot of background information which is not useful in the lip reading task. We use the face-detector module in OpenCV [13] to detect and extract faces from the images. This is crucial since our dataset is small, and we cannot afford the algorithm to waste computations on irrelevant parts of the image. After this step the size of each image becomes 128×128. Note that this is not the final size of image passed for training, since different methods use different size by cropping it further as required.

3.3. Data Augmentation

Our dataset contains a total of 3000 instances only. Our dataset is small especially for deep learning tasks. In order to tackle this problem we use data augmentation to artificially increase the data size. Our data augmentation includes the following two modifications to the original image.

- While cropping, slightly move around the crop region by random number of pixels horizontally and vertically

- Jitter the image by randomly increasing or decreasing the pixel values of the image by a small amount.

4. Methods

In this section we describe the various methods that we used to work on solving the lip reading problem using CNNs and LSTMs. Our model is primarily based on using convolutional neural networks, since they are excellent at deriving the correct features from images. A small caveat is that the dataset contains multiple images per data instance, i.e., the X in an (X, Y) pair in the dataset, is not a single image but a list of images with variable length. We have two different schools of thought to resolve this issue:

1. Our first idea was to simply append each image of a sequence such that we get a larger image.
2. Our second idea was to use LSTM layers to handle a sequence of features. This was expected to perform better because now the CNN layers would not need to infer the temporal information.

4.1. Concatenation Model

The first model first transforms the temporal information per datapoint into spatial, i.e.,

1. the first k images of the sequence are concatenated

(a) 4 faces (b) 9 faces

(c) 25 faces (d) 25 faces stretched

Figure 5: Example concatenated images

2. the new list is a 2D array with equal width and height. This essentially reshapes the sequence into a row-major 2D array. We tried 5 different structures here, with the first four setting missing images to 0:
   (a) Using the first $k = 4$ images (Figure 5a)
   (b) Using the first $k = 9$ images (Figure 5b)
   (c) Using the first $k = 16$ images
   (d) Using the first $k = 25$ images (Figure 5c)
   (e) Using the first $k = 25$ images stretched (Figure 5d)

Figure 5d shows the stretched version of figure 5c. The intuition behind this is that different speakers have different speed of speaking. By stretching the sequence to fit a length of 25, we are normalizing over the speaking speed. We expect this to help improve our validation accuracy. The sequence is stretched by filling in the missing images with nearest image rather than append zeros at the end. Equation 1 gives the equation used to convert original sequence to stretched sequence

$$\text{stretch\_seq}[i] = \text{orig\_seq}\left[\text{round}\left(\frac{i * \text{orig\_len}}{25}\right)\right] \quad (1)$$

Due to scarcity of data, we are required to use pre-trained models for convergence without over-fitting. Therefore, the image size we feed to the model must be identical to the ones used during pre-training, i.e., we crop out sections of the images before we join them such that the size of the final image is identical - 224 X 224 X 3. However, we ensure that no part of the lips is cropped out, since they are the most valuable section of the images. Thus, all of the input images are of the same size.

For training we retain most of the pre-trained VGGNet. We only replace the final fully connected layer, which now classifies among 20 classes using softmax (equation 2).

$$\text{loss} = -\sum_i \log\left[\frac{\exp\left(Wx_i\right)}{\sum_j \exp\left(Wx_j\right)}\right] \quad (2)$$

Thus, the learning rates we finally converged to were quite lower than usual, as most of the model already had the correct weights. In addition, we use dropout and l2 regularization to prevent overfitting. This model was implemented in python and trained using Keras [14] supported by Theano [15], [16] in the backend. The updates are done using Adam shown in equation 3

$$\begin{aligned} m &= \beta_1 m + (1-\beta_1)\, dx \\ v &= \beta_2 v + (1-\beta_2)\, dx^2 \\ m_b &= \frac{m}{1-\beta_1^t} \\ v_b &= \frac{v}{1-\beta_2^t} \\ x &= x - r * \frac{m_b}{\sqrt{v_b + \epsilon}} \end{aligned} \quad (3)$$

where t is used to indicate the iteration number, r is the learning rate and ϵ is a small positive number for numerical stability.

4.2. Model trained from scratch

In addition, we tried training a smaller model (Figure 6) from scratch, but the results were not very motivating. While the model did perform better than the baseline (random predictions), the results were not satisfactory. The model we used comprised of 4 CNN layers with sizes 64, 128, 256 and 512, respectively, all with kernel sizes 3×3. These were followed by two fully connected layers of size 512. The final layer was a softmax layer with output size 20.

Figure 6: Model trained from scratch

4.3. Recurrent Models

Our next model used recurrent neural networks (RNNs) to capture the temporal information from the data. Recurrent models have state information associated with them which gets updated with every point in the sequence. This makes recurrent models a good choice to work with sequential data. We use Long-Short Term Memory (LSTM) layers since they do not have the vanishing gradient problem faced by vanilla RNNs.

A state in LSTM is represented by hidden state (h) and cell state (c) variables. These variables are updated at each point in the sequence. LSTM update consists of using input (i), forget (f), output (o) and gate (g) variables. Equations 4 show the update equations for state variables of an LSTM.

$$
\begin{aligned}
i_t &= \sigma\left(W_x^{(i)} * x + W_h^{(i)} * h_{t-1}\right) \\
f_t &= \sigma\left(W_x^{(f)} * x + W_h^{(f)} * h_{t-1}\right) \\
o_t &= \sigma\left(W_x^{(o)} * x + W_h^{(o)} * h_{t-1}\right) \\
g_t &= \tanh\left(W_x^{(g)} * x + W_h^{(g)} * h_{t-1}\right) \\
c_t &= c_{t-1} \circ f_t + g_t \circ i_t \\
h_t &= \tanh\left(c_t\right) * o_t
\end{aligned}
\tag{4}
$$

In this model, we feed the images through the VGGNet pre-trained on human faces. The features extracted by the last CNN layer are then used by LSTM layers to extract temporal information from the sequence. The final hidden vector of the last LSTM layer is used by a softmax classifier to generate the label which is compared to the ground truth. Figure 7 gives a pictorial depiction of this model. Note that the number of LSTM layers and length of sequence shown in this figure is just for representation and varies in our actual model. This model was implemented in Lua using the Torch [17] module. We had to switch to Torch from theano due to missing support for multiple gpus in Theano and easier integration of LSTMs with CNNs in Torch.

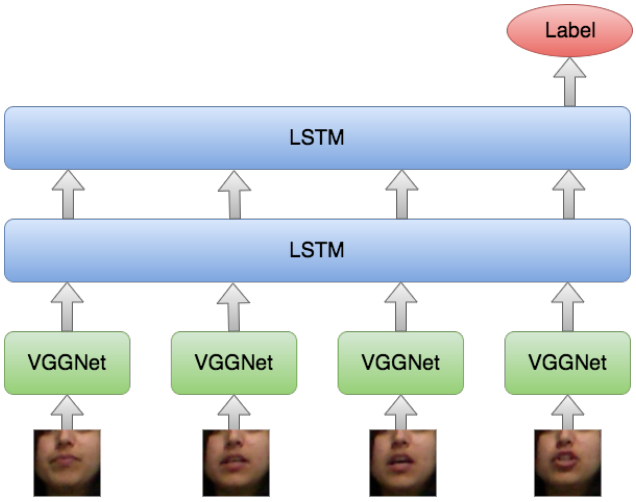

Figure 7: VGG and LSTM model

Since the VGGNet is pre-trained on faces, we tried two different modes of update:

1. Decrease the learning rate of VGGNet and train the complete model. We tried downscaling the learning rate by 100 and 1000. Both seemed to give equivalent results.

2. Freeze the VGGNet and only train the LSTM. This essentially involves doing a single forward pass through the VGGNet, and then reusing the extracted features as inputs to the LSTM for each epoch without updating the VGGNet.

   Further, we employed batch normalization (between different LSTM layers) for quicker convergence and dropout for regularization. We also used gradient clipping to prevent gradients from exploding.

5. Results

The single-validation accuracies of each model can be seen in Figure 8, along with the cross validation score of the state of the art SVM model. With 20 classes, a random baseline is 5%, while predicting from a single image of the sequence performs slightly higher. The concatenated image model performs progressively better as more frames

from each sequence are used, with the interpolation model surpassing the cross-validation score of the state of the art SVM model, achieving an accuracy of 76%. The LSTM models have lower accuracies, explained in more detail below.

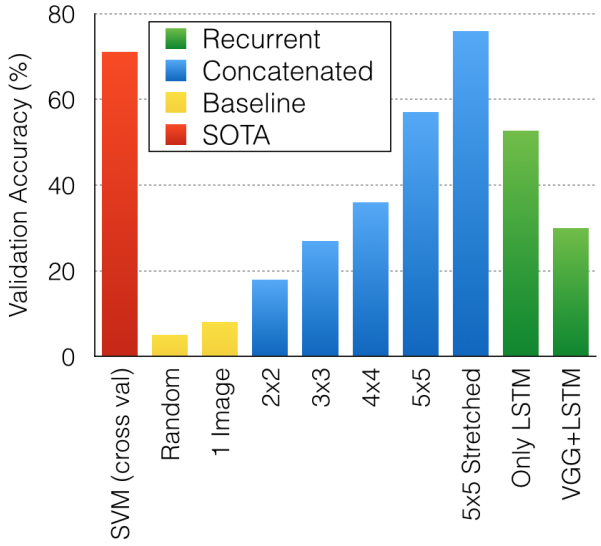

Figure 8: Accuracy of various models

The results from the best concatenated image model with interpolation are shown in Table 2. We have achieved best validation accuracy of 76%, and the test accuracy of our best model is 44.5%.

	Training	**Validation**	**Test**
Only words	63.1%	73.00%	56.00%
Only phrases	79.2%	79.00%	33.00%
Both	66.15%	76.00%	44.50%

Table 2: Best training, validation and test accuracies

Further testing produces the cross validation scores shown in Table 3. The unstretched model achieves 47.57%, while the stretched model achieves 59.73%, which both fail to surpass that of the state of the art SVM model. Even though these values are quite good as compared to random baseline of 5%, there is room for improvement. Figure 9 shows the training and validation accuracy during training which shows that our models are not able to generalize well. Additionally, the declining accuracies in later epochs shows that the model is regularizing too heavily and that early stopping is ideal.

Figure 10 shows the training loss with epochs. Although loss has not yet converged after training completes, we already see a peak accuracy (in Figure 9) and have no need to wait for convergence.

	5×5	5×5 stretched
Only words	41.60%	53.27%
Only phrases	53.53%	66.20%
Both	47.57%	59.73%

Table 3: Cross validation accuracies

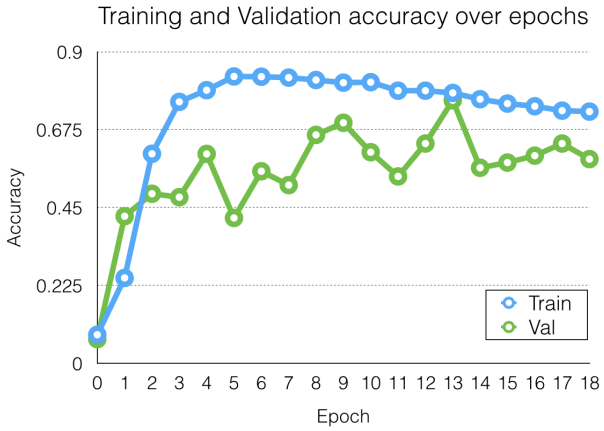

Figure 9: Accuracies over 18 epochs

Figure 10: Training loss over 18 epochs

Figures 11 and 12 give the confusion matrices for the best model evaluated on validation and test set. The matrices are quite informative in the kind of errors made. For instance, the phrase "How are you" and the word "previous" are getting mixed up in the test set. This is because the test set speaker's mouth movement for both of them is quite similar, which is not true for the speaker in validation

set. This discrepancy among various speakers is also evident from Figures 14 and 15.

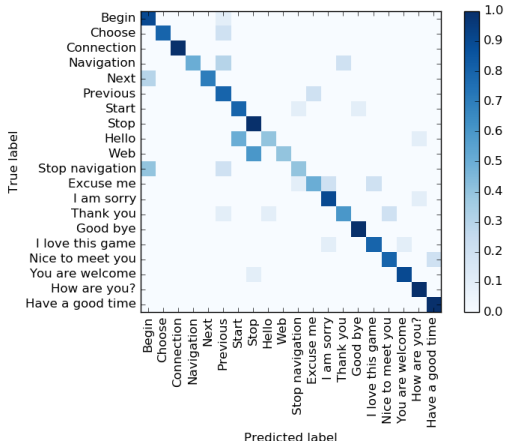

Figure 11: Confusion matrix on dev set for stretched model

This reveals another problem – difference in accents. People from different parts of the world have different ways of pronouncing the same word. This creates problems for a model that uses lip orientation to figure out the spoken word. We believe this to be the main cause behind the sharp valleys in our cross validation experiment.

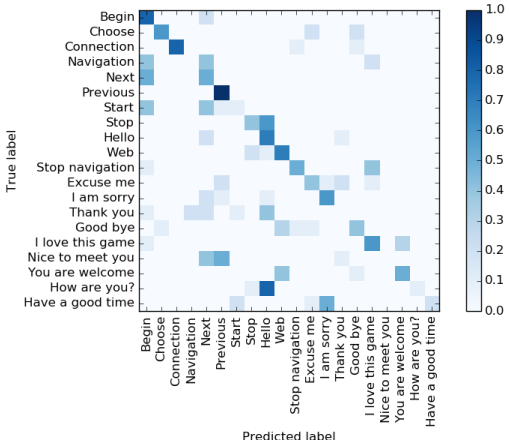

Figure 12: Confusion matrix on test set for stretched model

Further, it can be seen from the saliency map – see Figure 13 – (generated using guided backpropagation [18]) that the model is able to track the movement of lips and teeth to figure out the correct word / phrase. Also, it chooses to emphasize some images more than the others in the sequence which means that it is able to leverage the temporal information from the concatenated image.

When comparing the accuracy of words and phrases, we notice that phrases have a higher accuracy as compared to

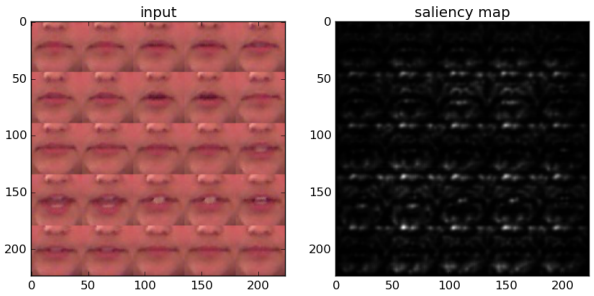

Figure 13: Saliency map for an image in dev set for normal training, where the stretched model works

words. We can attribute this to phrases being long and thus having a more data to differentiate between them whereas words are short and it would be more difficult to differentiate between them.

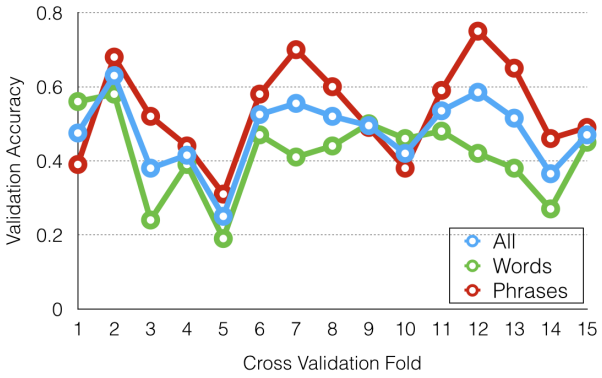

Figure 14: Cross Validation for 5×5 unstretched

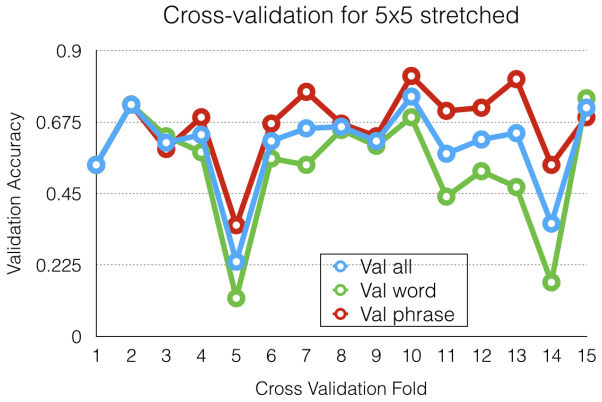

Figure 15: Cross Validation for 5×5 stretched

It can be seen that the stretched model in general performed better. To investigate the poor performance of the stretch model during the 5^{th} fold of cross validation, we looked at the saliency map (Figure 16) for a test image. It

can be seen that the model in this case has focussed on the wrong parts of the images – the white regions in the saliency map do not include the mouth but the nose and some part of cheek. This is most likely due to scarcity of data, due to which the model is not robust enough to work in all cases.

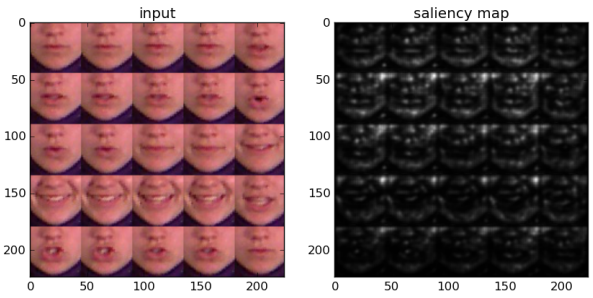

Figure 16: Saliency map for an image in dev set while cross validating, where the stretched model fails

We also experimented with different parameter update strategies. We noticed that SGD itself was incapable of training the model in reasonable time. It gave no significant improvements even after 20 epochs. In comparison, Adam showed improvements right from the first epoch.

6. Conclusion

We proposed several new methods for performing visual speech recognition on sequences of color images with variable length. The initial methods concatenated the first k images of each sequence into a 2D grid, which was then classified by a VGGNet pre-trained on faces. One method attempted to train a smaller model on these concatenated images from scratch. The final model attempted to handle variable-length sequences with multiple LSTM layers which were given the feature vectors output from the VGGNet as input.

Our best-performing model was the concatenated model that used interpolation. This likely performed better than the other concatenated models because it used nearly all available input data, and it was invariant to speaking speed, as opposed to the original 5×5 concatenated model which only compared each frame to corresponding frames in other sequences.

The model we trained from scratch did not perform well because the dataset we used is relatively small. Our data augmentation did not provide any independent data, so this could not have provided a significant boost. Additionally, although we created custom concatenated images, the features we hoped to extract from each were the same as those that can be found in a photograph of a single face. Therefore the pre-trained VGGNet was still able to perform well on these concatenated images.

While we originally expected the LSTM model to perform the best, we expect that it failed because it does not handle the sequence until after feature extraction. It may perform better if we instead use recurrent networks throughout the model to include temporal information while extracting features. Additionally, the LSTM model took a long time to train, especially while updating the VGGNet as well, and with more time it may have produced better results.

7. Future Work

Our models leave much to be experimented, especially the LSTM models which are difficult to tune due to long training times. Due to the time constraint, we were unable to train these models to their full potential, evident from their dismal performance. In addition there are other architectures that seem appropriate for this problem setting:

1. Using volumetric convolutions rather than 2D convolutions followed by recurrent networks. This has the benefit of learning temporal information in conjunction with the spatial features, rather than learning them separately. One key challenge would be training the new model from scratch since it would be incompatible with the pre-trained VGGNet.

2. The dataset has rich depth information that could be useful in better estimating the texture of the mouth. Since we were primarily working with pre-trained networks, we were unable to determine a reasonable way to include it in the model.

3. Replacing the kernels in the CNN with recurrent kernels similar to the method suggested in [19]. This not only has the advantage that the volumetric convolutions enjoy, but it can also learn longer temporal relations. Again the key issue would be training the model from scratch.

4. Optical flow is an important feature used to train models on videos. This can be another feature besides those learned through the CNN and directly used by the LSTM.

References

[1] A. V. Omkar M Parkhi and A. Zisserman, "Deep face recognition," *Proceedings of the British Machine Vision*, vol. 1, no. 3, p. 6, 2015.

[2] A. Rekik, A. Ben-Hamadou, and W. Mahdi, "Human machine interaction via visual speech spotting," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2015, pp. 566–574.

[3] A. Rekik, A. Ben-Hamadou, and W. 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*, 2014, pp. 21–28.

[4] A. Rekik, A. Ben-Hamadou, and W. Mahdi, "Unified system for visual speech recognition and speaker identification," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2015, pp. 381–390.

[5] G. Zhao, M. Barnard, and M. Pietikäinen, "Lipreading with local spatiotemporal descriptors," *Multimedia, IEEE Transactions on*, vol. 11, no. 7, pp. 1254–1265, 2009.

[6] E. K. Patterson, S. Gurbuz, Z. Tufekci, and J. N. Gowdy, "Cuave: A new audio-visual database for multimodal human-computer interface research," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 2. IEEE, 2002, pp. II–2017.

[7] A. Rekik, A. Ben-Hamadou, and W. Mahdi, "An adaptive approach for lip-reading using image and depth data," *Multimedia Tools and Applications*, pp. 1–28.

[8] Y. Pei, T.-K. Kim, and H. Zha, "Unsupervised random forest manifold alignment for lipreading," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 129–136.

[9] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[10] A. A. Shaikh, D. K. Kumar, W. C. Yau, M. C. Azemin, and J. Gubbi, "Lip reading using optical flow and support vector machines," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 1. IEEE, 2010, pp. 327–330.

[11] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Lipreading using convolutional neural network." 2014.

[12] M. Wand, J. Koutnk, and J. Schmidhuber, "Lipreading with long short-term memory," 2016.

[13] G. Bradski, *Dr. Dobb's Journal of Software Tools*.

[14] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[15] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[16] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.

[17] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," IDIAP, Tech. Rep., 2002.

[18] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[19] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," 2015.