Automated Lip Reading using Deep Learning - Final Report

Kommana Vikas
CS19BTECH11045
cs19btech11022@iith.ac.in

Nistala Praneeth
CS19BTECH11054
cs19btech11054@iith.ac.in

Sujal
AI20BTECH11020
ai20btech11020@iith.ac.in

Vaibhav Chhabra
AI20BTECH11022
ai20btech11022@iith.ac.in

Vishwanath Hurakadli
AI20BTECH11023
ai20btech11023@iith.ac.in

Abstract

Lip reading is a technique to understand words or speech by visual interpretation of face, mouth, and lip movement without the involvement of audio. Here we present our model to predict words from only video(image sequence) without any audio signal. We employ a CNN-based classifier and compare the accuracy with ResNet model, pre-trained on the human faces of celebrities from IMDB and Google Images. We also explore different ways of handling these image sequences. Our model is a CNN-based model and it is trained on images concatenated from multiple frames in each sequence, to make an image grid. Though our model failed to outperform the ResNet model, we achieved very good training and testing accuracies of 92.19% *and* 86.22% *respectively.*

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 data needs to be pre-processed before training, which includes removing the informations which are not required and making it compatible with the model that has to be used.

2. Literature Review

Following are some of the papers, that helped us build our model on lip reading using deep learning.

2.1. Lip reading using CNN and LSTM

In this paper the authors tried 3 different models based on 2 main ideas to solve the AVR problem. They changed the problem to a classification problem by restricting the dataset which consists of a particular number of words and phrases. The input data is taken from MIRACL-VC1 dataset. As the data is low-rate videos and each video had a variable number of frames, the temporal data was needed to be captured to take all input features into account.

In the first model they transformed the temporal data of a given word or phrase into spatial by concatenating the k-images of the sequence. The output of this is a 2D array with equal width and height for all the data points. Each photo is cropped so that the only important features like lips are considered. Since the number of frames for each data point are different, the output is stretched so that it can fill the gaps and is done by using

```
stretch_seq[i] = original_seq[round((i *
        original_len)/25)]
```

An example of this is shown below:

Figure 1. 13 faces in 5x5 grid before and after stretching

After the input data was converted, the model was trained

using VGGNet which used a set of weights pre-trained on faces.

In the second method, they tried to use their own written neural network instead of VGGNet.

The third method they tried is by using 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 which makes these models a good choice to work with sequential data. In this model, they extracted some features of these images using VGGNet and then were provided as inputs for 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.

Out of all these methods, the first method performed the best with validation accuracy of about 75%.

2.2. Lip Reading Word Classification

Lip-Reading is a technique which includes a host of tasks, first of which is face detection. After face detection has been carried out, the Region of Interest is to be detected, which is the mouth of the Speaker. Mouth and lips segmentation could be an elementary problem in the detection of the face. This is an outline of the tasks the research paper followed. They first pre-processed 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 used the sequence of frames as the input to the model. The goal was to investigate the task of speech recognition from video without audio.

They presented several neural network models with varying successes in the classification task. The input data to the algorithm was sequences of still images taken from frames of video footage. They used different neural network models to output one of 10 words that were spoken (or mouthed) by a face in the input images. It included exploration and combination of a number of different models including CNNs, RNNs, and existing publically available pre-trained networks to assist in mouth recognition.

They used the MIRACL-VC1 data set containing both depth and color images of fifteen speakers uttering ten words and ten phrases, ten times each. Preprocessing was an important part of working with this dataset. The model ran every image of the 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.

They made the utilization of smaller data sizes, focused on building a classifier that can identify which word is being uttered from a sequence of images of the speaker as input, and ignored the set of phrase data and also the depth images for the spoken word data.

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

Here is an example of the word "Hello" with a sequence of 10 snapshots.

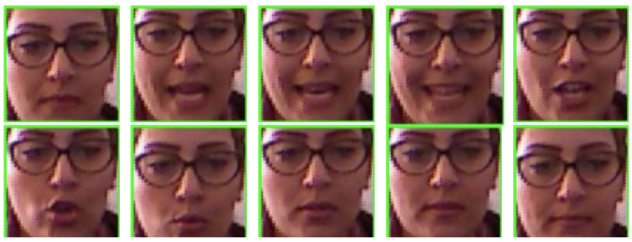

Figure 2. Image Frames saying "Hello"

Hence for some moments when we receive small data sets, we can solve the problem by performing data augmentation.

3. Dateset and Features

In this section we describe the dataset that we used for training, some of its properties and the pre-processing that we had to perform on it.

3.1. Dataset characteristics

We used the MIRACL-VC1 dataset in our project. It is used for diverse research fields like visual speech recognition, face detection, and biometrics. The dataset was created from 15 people(10 females, 5 males) who spoke ten instances of ten words and ten phrases, leading to a total of $15 \times 20 \times 10 = 3000$ instances. Each instance is a numbered sequence of color and depth images of 640×480 pixels. For example, the images for one instance is shown in Figure 3.

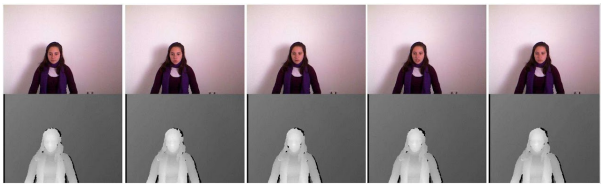

Figure 3. Example Instance

The words and phrases in the dataset are listed here

ID	Words	ID	Phrases
1	Begin	1	Stop navigation.
2	Choose	2	Excuse me.
3	Connection	3	I am sorry.
4	Navigation	4	Thank you.
5	Next	5	Good bye.
6	Previous	6	I love this game.
7	Start	7	Nice to meet you.
8	Stop	8	You are welcome.
9	Hello	9	How are you?
10	Web	10	Have a good time.

We only used the color images and discarded the depth images while implementing our model. Also, we used only words for the classification and discarded the phrases. The length of sequences in the dataset varies from a minimum of 4 images to a maximum of 22 images for words. On an average word sequences have 10.33 images. Out of the data of $15 \times 10 \times 10 = 1,500$ images, we used $1,050$(105 from each class of words) for training, and 450(45 from each class of words) for testing(validation).

3.2. Data Pre-processing

First, we modify the data since it has a lot of background information which is not useful in the lip reading task. We used the face-detector module in OpenCV and dlib to identify the faces from the image and cropped the lip region. This is crucial since our dataset is small, and we cannot afford the algorithm to waste computations on irrelevant parts of the image so we resize all image using CV2 library by add padding and make the size of all the lip regions equal of size 100×100. 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.

The next step is to combine the images of a particular instance into a single image. We were creating a single image that is 5×5 grid of the images of the instance. Since the number of images will not be 25, some of the images should be repeated. Thus we have to find a way to determine the number of times an image will be repeated. The way we did this is first we equally divided the 25 slots among all the images. This might leave some additional slots say k. Now we calculated the differences between every 2 adjacent images and gave an extra slot to the images that changed more. We filled the grid sequentially based on the number of slots. We later then resized the grid to the size 224*224, and saved these images in a directory, label-wise, which was later used by our CNN Model.

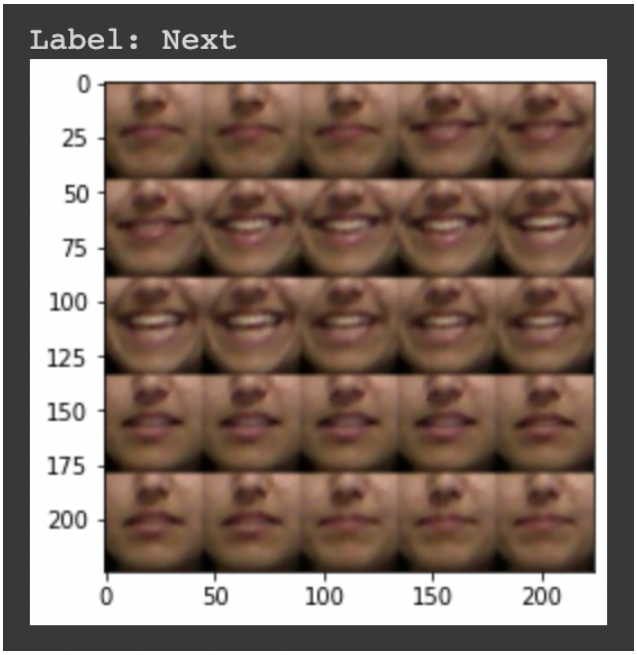

Figure 4. Example of an image stored after preprocessing

4. Model

In this section we describe the methods we used solve the lip reading problem. For both of these methods, we used the images that we pre-processed. We divided the images into train(70%) and test(30%) images randomly.

4.1. Model Implementation

We divided the images into batches of size 50, passed them into our CNN model. The model we made has a linear stack of CNN layers and fully connected layers. We added two Conv2D layers, with non-linear function as ReLu, then a maxpool layer, then two sets of a Conv2D layer with ReLu activation and maxpool layer and then flattened the output for the fully connected layers. We added dropouts($p = 0.5$ and $p = 0.3$) and batch-normalisation before & after the first fully connected layer, added ReLu activation and then added the second fully connected layer followed by the Softmax layer. We calculated the error with categorical cross entropy as the loss function and used RMSProp as optimizer, while backpropagating.

4.2. ResNet Model

We compared this model with the Residual Network (ResNet) model, which is one of the very famous deep learning model for image related data. The ResNet model is deeper than networks like VGG, but is still less complex. We used the ResNet18 model which was pre-trained with the human faces of celebrities from IMDB and Google Images.

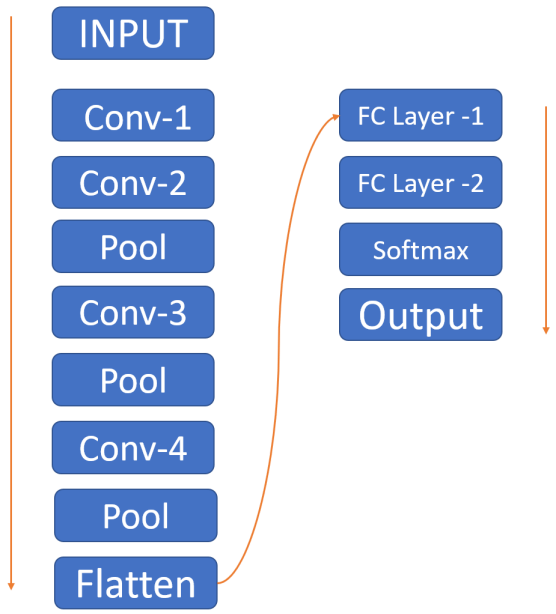

Figure 5. Illustration of our CNN model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 224, 224]             896
              ReLU-2         [-1, 32, 224, 224]               0
            Conv2d-3         [-1, 64, 224, 224]          18,496
              ReLU-4         [-1, 64, 224, 224]               0
         MaxPool2d-5         [-1, 64, 112, 112]               0
            Conv2d-6         [-1, 64, 112, 112]          36,928
              ReLU-7         [-1, 64, 112, 112]               0
         MaxPool2d-8           [-1, 64, 56, 56]               0
            Conv2d-9           [-1, 64, 56, 56]          36,928
             ReLU-10           [-1, 64, 56, 56]               0
        MaxPool2d-11           [-1, 64, 28, 28]               0
      BatchNorm1d-12                [-1, 50176]         100,352
        Dropout-13                [-1, 50176]               0
           Linear-14                 [-1, 4096]     205,524,992
      BatchNorm1d-15                 [-1, 4096]           8,192
        Dropout-16                 [-1, 4096]               0
             ReLU-17                 [-1, 4096]               0
           Linear-18                   [-1, 10]          40,970
          Softmax-19                   [-1, 10]               0
================================================================
Total params: 205,767,754
Trainable params: 205,767,754
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 97.74
Params size (MB): 784.94
Estimated Total Size (MB): 883.26
----------------------------------------------------------------
```

Figure 6. Summarization of CNN model

4.3. Parameters in our CNN Model

Here our input data of training set is image $224 \times 224 \times 3$ with a batch size 50, our CNN model consists of four convolution layers, first Conv2D layer have 32 filters of size $3 \times 3 \times 3$ and including bias term it will have 896 paramters . Similarly in the second convolution layer the number of learnable parameters are $18,496$ and for third convolution layer number of parameters is 36928, which is also same for fourth layer as both layer has same number and size of kernels. First batchnorm, after the flattening of output from fourth convolution layer, has $100,352$ parameters. First fully connected layer(hidden layer) has $205,524,992$ parameters. As the size of input and output is high in hidden layer which results in large number of parameters. Second batchnorm give $8,192$ learnable parameters. Output layer before Softmax have $40,970$ learnable parameters. Considering all parameters described, they total to $205,767,754$ learnable parameters. Complete description of the model and parameters are shown in Figure 6.

5. Result

We tried classification of augmented images using the above mentioned CNN model, which was inspired to some extent by a VGG model, but had lesser number of convolution layers, and we compared the accuracies of this model with ResNet18 model.

- Results from CNN model.
  The model had $205,767,754$ trainable paramters and 0 non-trainable parameters. Executing the code with 25 epochs, Train Accuracy in first epoch obtained is around 21.62% and that with test accuracy is around 25.33%. After execution of 25 epochs the final train and test accuracies obtained were 92.19% and 86.22% respectively. The Accuracy of both train dataset and test dataset with respect to epoch are shown in Figure 7. According to this graph, the accuracies are impoved over epochs and attain a decent stability by the end of 25 epochs.

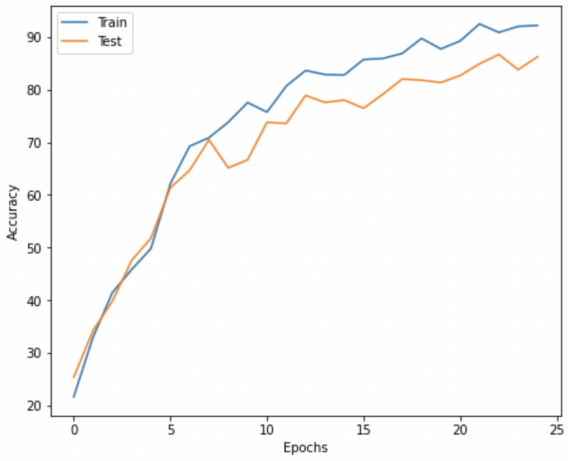

Figure 7. Accuracy of CNN model

- Results from ResNet model
  In ResNet number of trainable parameters is around $11,181,642$ and zero non-trainable parameter. As executed model for 25 epoch, initial accuracy of model is 14.09% as train accuracy and 16.44% as test accuracy by the end of first epoch. After execution of 25 epochs the final train and test accuracies obtained were 99.14% and 97.55% respectively. The Accuracy of both train dataset and test dataset with respect to epoch are shown in Figure 8. We can see the accuracy is somewhat stable by the end of 25 epochs.

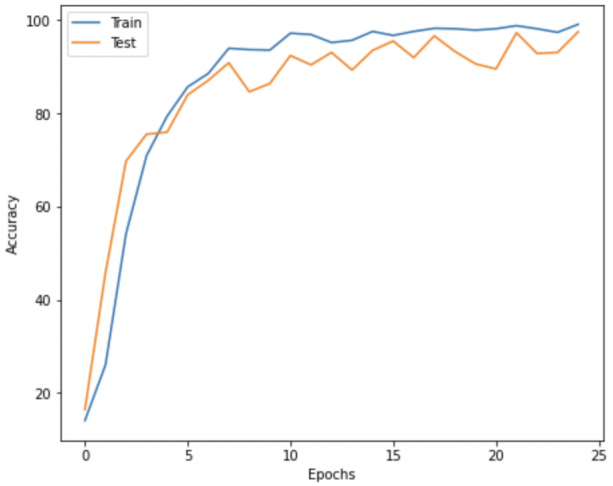

Figure 8. Accuracy of ResNet18 model

As we tried to train two different model on the same dataset, we can compare the accuracies obtained by both models. At the end of 25 epochs CNN model produced less accuracy on both training dataset and test dataset compared to that of ResNet model. ResNet is deeper than CNN and has more number of layers compared to CNN, which is one of the most important factor boosting the accuracy. ResNet has residual block which makes it easier for models having more number of layers to train.

6. Conclusion

This Lip-reading using CNN report compares the two approaches used and suggests that the ResNet model is better than our proposed CNN model. However, the CNN model also shows a decent accuracy, even though it had lesser number of layers, which shows how powerful deep networks like CNNs are. We learnt and applied the key concepts of deep learning models. We also found that preprocessing is an essential task in such deep learning models.

The LSTM models are known to be very efficient in handing classification of sequential data, but due to longer training times, and difficult tuning, we were unable to implement them for making a Lip Reading Model.

7. Contributions

- Kommana Vikas (cs19btech11045@iith.ac.in)
  - Literature Review
  - Coding part for Preprocessing
- Nistala Praneeth (cs19btech11054@iith.ac.in)
  - Coding part for Preprocessing
  - Helped in Report
- Vishwanath Hurakadli (ai20btech11023@iith.ac.in)
  - Coding part for the Models
  - Making the Report
- Vaibhav Chhabra (ai20btech11022@iith.ac.in)
  - Literature Review
  - Coding part for the Models
  - Making the Report
- Sujal (ai20btech11020@iith.ac.in)
  - Making the Report
  - Literature Review
  - Coding part for splitting test and train dataset

8. References

The links to the papers, datasets and libraries we used can be found below:

- Dataset Used MIRACL-VC
- Dlib
- OpenCV
- Lip reading using CNN and LSTM
- Lip Reading Word Classification