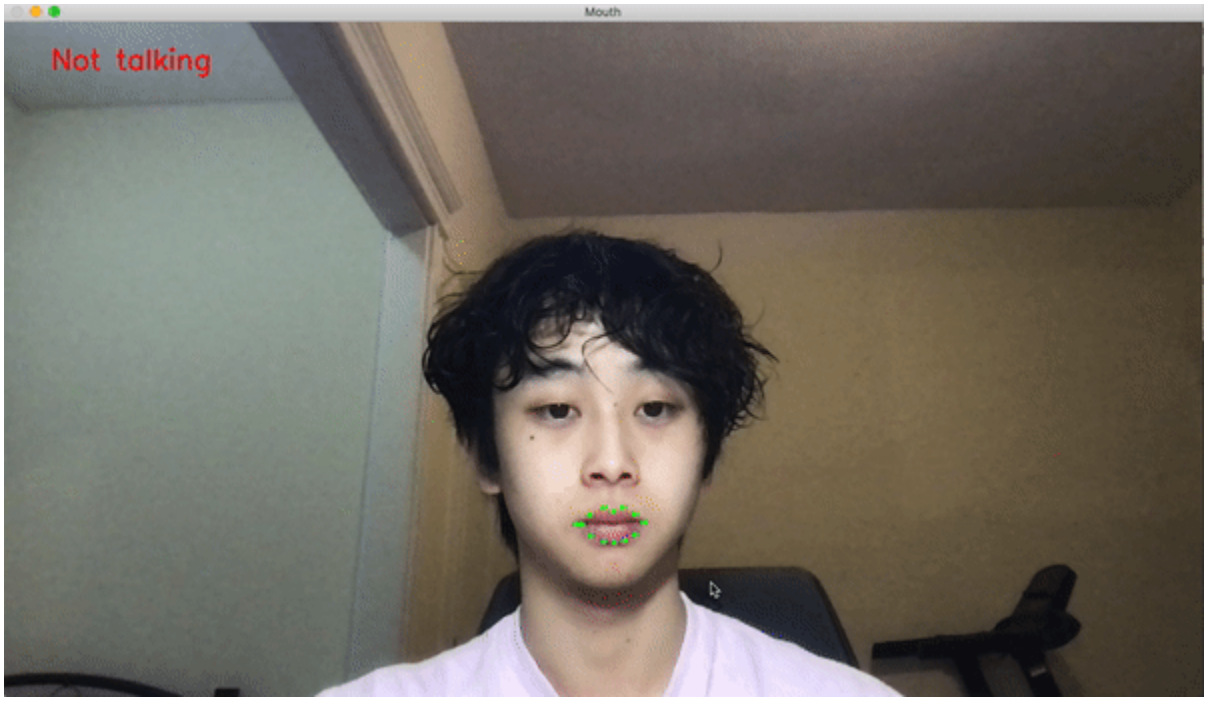
Computer Vision Lip Reading

By: Allen Ye

Introduction:

Abstract:

The goal of this project is to build a speech recognition system that can accurately recognize spoken words from a set of predefined words. The algorithm used to recognize words uses computer vision and deep learning, and is trained on a large dataset generated by myself and a friend. The dataset consists of around 700 individual video clips of subjects speaking individual words, totaling approximately 3 GB of data. The model architecture includes several convolutional and dense layers, and was trained using TensorFlow and Keras. The training process achieved a 95.7% training accuracy and a 98.5% validation accuracy, demonstrating strong classification performance. Once trained, the system can be used to recognize spoken words in a live setting.



Problem:

According to the WHO, more than 1.5 billion people or approximately 20% of the global population live with hearing loss.[1] Of those 1.5 billion people, 430 million of them have disabling hearing loss and suffer from deafness or hard-of-hearing.[2] In the United States alone, about 29 million or more than 11% of U.S. adults could benefit from using hearing

aids.[3] Hearing loss is mainly attributed to two causes: aging and exposure to loud noises. Due to the influx of technological developments in recent years (i.e. earbuds and headphones), hearing loss due to continual exposure to sound has become more common[4]. Given the increasing number of people suffering from hearing loss[5], I propose using computer vision and deep learning to help the deaf and hard-of-hearing by creating an algorithm that can transform lip movements into words.

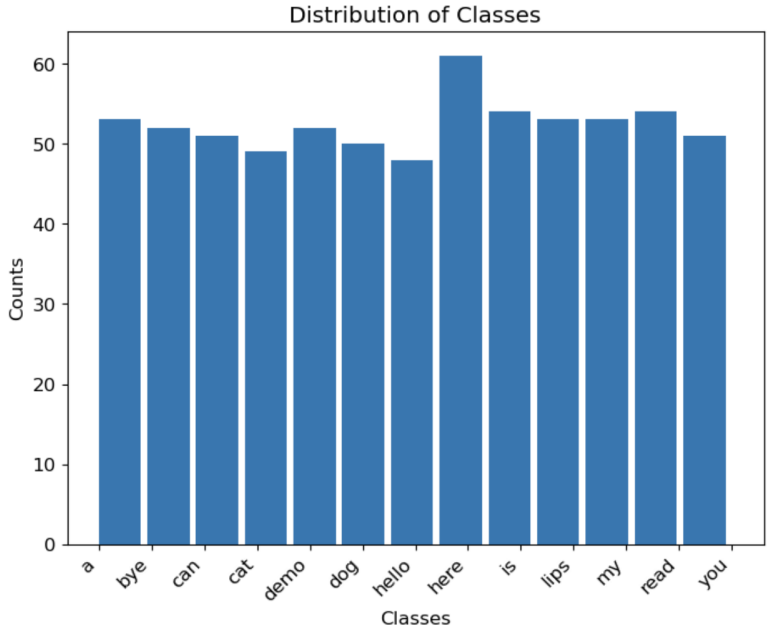
Methods:

Libraries:

The libraries I used to train my models include TensorFlow and Keras. I utilized the image libraries OpenCV and PIL for data collection and preprocessing. Other libraries used during the training process includes numpy and scikit-learn for preparing the data, as well as matplotlib and seaborn for graphing visualizations.

Dataset:

As this is a supervised learning project, I needed a dataset to train my model on. Since a suitable dataset was not available for this problem, I took the initiative to create my own dataset by collecting approximately 700 video clips of words being spoken. Each video clip was manually labeled with a word from a predefined set and in the end, I had around 3 gigabytes worth of total data. My dataset contains data for 13 different words, and to give an overview of the dataset's distribution, you can view the histogram chart on the left showing the number of video clips for each word. The x-axis represents the words in the set, and the y-axis represents the number of video clips. The chart confirms that the dataset has a roughly equal number of video clips for each word, with some minor variation.



This dataset is then split into training and testing, and we allocated 80% of it towards training and the rest towards testing. The final size of the training and testing dataset is (544, 22, 80, 112, 3) and (137, 22, 80, 112, 3) respectively.

Each video clip consists of 22 frames and has the dimensions 112x80 for width and height respectively. Below, I have included a few examples of the training data.

Environment:

I do not possess the necessary hardware (such as NVIDIA GPUs) for deep learning. Thus, I decided to train my models on the cloud using Kaggle, as this platform provides a variety of free accelerators (CPUs, GPUs, and TPUs). Kaggle satisfied my processing power needs, but the downside of using a free cloud service was that I had limited memory to work with. I had to constrict my dataset so that it would not exceed Kaggle's RAM limit.

Methods:

Creating My Dataset:

The steps to creating my dataset are as follows:

1. Use my computer's webcam and OpenCV to stream myself talking.
2. Apply transfer learning and run a pre-trained model (DLIB's Face Detector model) to segment out the mouth portion of the video feed.
3. Separate the video feed into frames and perform image processing on each frame (See below on how I processed each video frame for our model).
4. Save the final video clip as a numpy array (the final dataset is a numpy array of dimensions (681, 22, 80, 112, 3) as I have 681 video clips, 22 frames per video clip, 80x112 sized frames, and 3 channels for every pixel in the frame).

Image Processing:

Image processing is extremely useful for obtaining better model metrics and results. The right amount of processing can also speed up training time and reduce memory and CPU usage. Below are the techniques used to process each segmented frame. The steps to my frame processing are as follows:

1. Lip Segmentation
2. Gaussian Blurring
3. Contrast Stretching
4. Bilateral Filtering

5. Sharpening
6. Gaussian Blurring

Lip Segmentation:

Since the goal of this project aims to translate lip movements into words, I want to segment out just the lips and surrounding area. I created constant values for width and height, and after segmenting out the lips, I would add padding to the segmented image so that the final frame matches the predefined constant dimensions.

Gaussian Blurring:

Gaussian Blurring is a useful image pre-processing technique that passes a Gaussian filter through an image to blur edges and reduce contrast. By softening sharp curves previously present in the image, my model will have an easier time performing calculations in the hidden layers. Ultimately, Gaussian Blurring will reduce train time and overfitting.

Contrast Stretching:

Contrast Stretching is an image enhancement technique used to improve the contrast of an image by stretching the range of intensity values. By increasing the contrast between the darker and lighter pixels in an image, the details become more visible, which can help my models perform better. Contrast stretching is particularly helpful in this case since it enhances the contrast between different regions in the image, making the lips more distinguishable and prominent.

Bilateral Filtering:

Bilateral Filtering is a non-linear image filtering technique that smooths the image while preserving edges. This technique uses both the spatial and intensity distances between pixels to achieve this. By reducing noise in the image, Bilateral Filtering can improve image quality, which can improve the performance of my models. Bilateral Filtering can also help to remove small objects or noise in the image, which is particularly useful in this situation.

Sharpening:

Sharpening is an image enhancement technique that accentuates the edges of objects in an image, making them appear more defined and distinct. This technique can help to increase the contrast between objects and their surroundings, making it easier for my models to detect and recognize them. By improving the edges in an image, sharpening can improve the accuracy of object recognition tasks.

Deep Learning Models:

As I am classifying video clips or sequences of frames for this project, I will utilize 3D convolutional neural networks.

3D CNN:

After setting up the video dataset, I designed my model architecture with the intention of capturing the spatiotemporal features of the video frames. The model begins with three Conv3D layers with a kernel size of 3x3x3, each having 8, 32, and 256 filters respectively. Each Conv3D layer except the last one is followed by a MaxPooling3D layer with a pooling size of 2x2x2. Following the convolutional layers, I add a Flatten layer to prepare the input for the densely connected layers. Next, I have three Dense layers, with 1024, 256, and 64 neurons, respectively, and a Dropout layer with a rate of 0.5 in between each Dense layer. Finally, I have a Dense layer with 13 neurons, representing the 13 classes I want to classify the videos into. In the end, I have around 200 million training parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 20, 78, 110, 8)    656

 max_pooling3d (MaxPooling3D  (None, 10, 39, 55, 8)    0
 )

 conv3d_1 (Conv3D)           (None, 8, 37, 53, 32)     6944

 max_pooling3d_1 (MaxPooling  (None, 4, 18, 26, 32)    0
 3D)

 conv3d_2 (Conv3D)           (None, 2, 16, 24, 256)    221440

 flatten (Flatten)           (None, 196608)            0

 dense (Dense)               (None, 1024)              201327616

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 256)               262400

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 64)                16448

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 13)                845

=================================================================
Total params: 201,836,349
Trainable params: 201,836,349
Non-trainable params: 0
_________________________________________________________________
```

I prevent my model from overfitting by also adding L2 regularization to each Conv3D layer, as well as using multiple Dropout layers between my Dense layers.

For each L2 regularization in a Conv3D layer, I use an alpha value of 0.001. L2 regularization reduces the impact of the weights in the model during training, by adding a penalty term to the loss function. This encourages the model to have smaller weights and prevents the model from becoming overly sensitive to the training data.

Dropout drops out a fraction of the neurons in a layer during training. This helps to prevent over-reliance on any single neuron and encourages the model to learn more robust features that are useful for making accurate predictions on new data.

Results:

Before explaining the results of my model, I will first provide an overview of the metrics used to evaluate my model.

Metrics:

Accuracy: It is one of the most basic metrics. The accuracy score is determined by testing the model on "new" data or data the model has never been trained on. We compare the percentage of correct predictions made by the model to the actual labels.

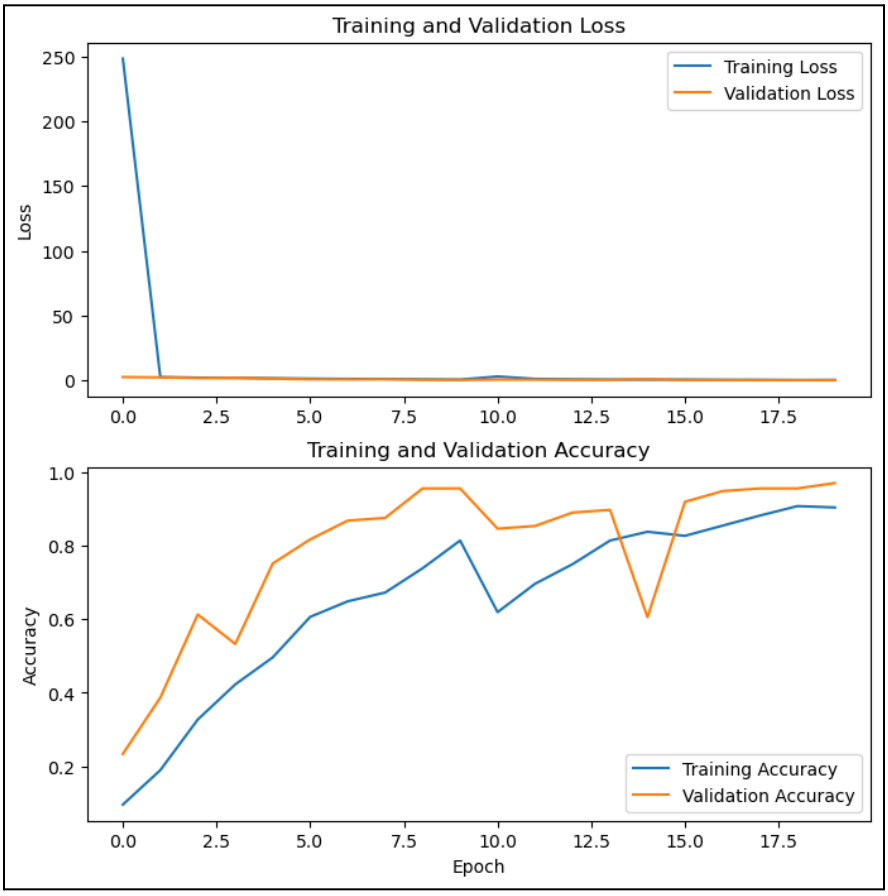
Balanced Accuracy: Similar to the accuracy metric, but in this case, this metric takes into account the different distributions of data counts. This metric takes into account discrepancies in unbalanced datasets and gives us a balanced accuracy value. It is to be noted that in my situation, the dataset is mostly balanced.

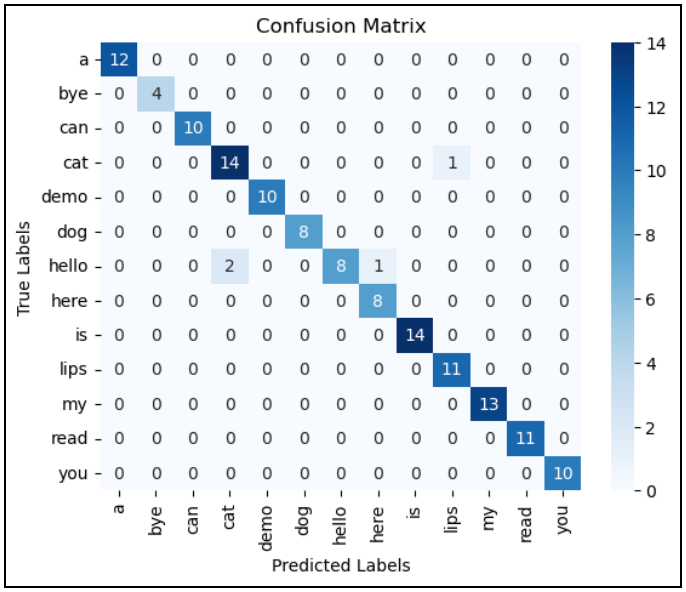
Precision: The precision metric is the ratio of True Positives to the sum of False Positives plus True Positives ($P = \frac{TP}{TP + FP}$). We can think of precision as a measure of the confidence of my model. The higher the precision, the more confident we are in the predictions made by the model.

Recall: Similar to precision, this metric is the ratio of True Positives to the sum of False Negatives plus True Positives($R = \frac{TP}{TP + FN}$).

F_1 Score: The F_1 Score is the weighted average of my precision and recall metrics, and is calculated with this formula:$\frac{TP}{TP + \frac{1}{2}(FP + FN)}$. We can think of the F_1 score as a "middle ground" between precision and recall. Precision and recall are specific to the situation: in some situations maximizing the precision over recall is optimal while vice versa in other situations. In general, having a high F_1 score equates to a good model.

I ran my 3D CNN for 20 epochs, and the graph to the right shows the model's progress. My final training accuracy was 95.7%, and my final testing accuracy was 98.5%. We use the testing accuracy as an important metric to evaluate my model's performance because it shows how the model performs on unseen data that it was not trained on. A test accuracy of 98.5% is very high and indicates that my model is capable of performing well on new data.

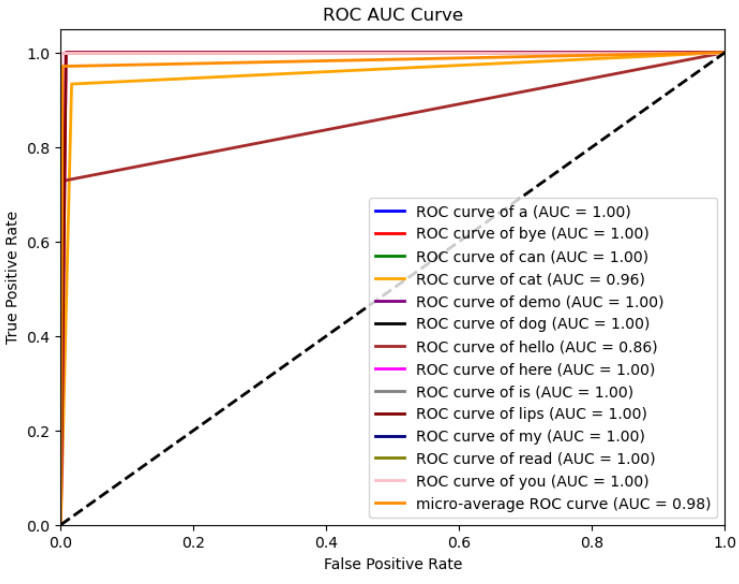




If we examine the confusion matrix generated using the testing data, we can see that a few instances where my model predicts wrong is when it confuses words “hello” with “cat”. When testing other models, a lot of the confusion matrices confused visually similar (in context of lip movements) words such as “hello” with “demo” or “my” with “bye”. I purposely tried including these visually similar words in my project in order to get a sense of how well the model can find and distinguish details between the lip movements.

To the right, we have a table containing more metrics regarding my model’s performance. My model achieved excellent results with a precision, recall, and F1 score of 1.0 for most classes, indicating that predictions were highly accurate and that the model correctly identified most of the positive samples. However, the recall for the "hello" class was relatively low at 0.73, which could be an area for improvement in future iterations. We also ended up with a balanced accuracy of 97.4%.

Class	Precision	Recall	F1-Score	Support
a	1.0	1.0	1.0	12
bye	1.0	1.0	1.0	4
can	1.0	1.0	1.0	10
cat	0.88	0.93	0.9	15
demo	1.0	1.0	1.0	10
dog	1.0	1.0	1.0	8
hello	1.0	0.73	0.84	11
here	0.89	1.0	0.94	8
is	1.0	1.0	1.0	14
lips	0.92	1.0	0.96	11
my	1.0	1.0	1.0	13
read	1.0	1.0	1.0	11
you	1.0	1.0	1.0	10



We can observe from the ROC AUC curve on the right that the AUC values are consistently high, indicating excellent discriminatory power of the model. The ROC curve shows a steep rise towards the top-left corner, indicating that the model has high true positive rates at low false positive rates, which indicates my model performs well.

Discussion:

Using 3D CNNs to classify video clips is not a novel idea, and has been done many times before for different tasks. One example of a 3D CNN classification problem is classifying video clips of humans into a specific type of action. This problem was documented by a research paper "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" written by João Carreira and Andrew Zisserman. Similar to my research, in their paper they were able to achieve 97.9% accuracy on a dataset which indicates that 3D CNNs are a good choice when the problem is a classification task and the dataset is composed of video clips.

Challenges and Future Improvements:

Gathering Data: I was unable to find a suitable dataset to meet my needs, so I resorted to generating my own dataset. This took a long time and a future improvement would be to gather even more training data from myself and many others.

Limited Processing Power: Because I had limited processing power, it was unrealistic to have a huge dataset or use higher quality images. I was also limited in the number of approaches I could take to perform hyperparameter optimization since with the limited processing power, an approach like grid search would be unfeasible. Future improvements would include having better hardware to train models so that I will no longer be limited by RAM and processing speeds.

Further Application:

This speechreading algorithm has numerous potential applications. First, it can help people suffering from hearing loss or deafness more easily communicate with others. Second, it can be applied to video surveillance systems where captured footage often doesn't have accompanying audio or has inadequate audio.

Conclusion:

This project uses computer vision and deep learning to create an algorithm that translates lip movements into spoken words. I use transfer learning to create a dataset of 700 video clips of words being spoken. Each clip is subjected to various image processing techniques, including lip segmentation, Gaussian blurring, contrast stretching, bilateral filtering, and sharpening. Libraries such as TensorFlow, Keras, OpenCV, PIL, numpy, and scikit-learn are used for data preparation and model training. The model architecture includes convolutional and dense layers and achieves a 95.7% training accuracy and a 98.5% validation accuracy. The model is lightweight enough that it can be used to recognize spoken words in a live setting once trained.

References:

1. https://www.who.int/health-topics/hearing-loss#tab=tab_2
2. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss#:~:text=Overview,will%20have%20disabling%20hearing%20loss
3. https://www.forbes.com/health/hearing-aids/deafness-statistics/#:~:text=Nearly%2029%20million%20adults%20in,men%20and%205.4%25%20of%20women
4. https://www.hopkinsallchildrens.org/Patients-Families/Health-Library/HealthDocNew/Earbuds#:~:text=Turning%20the%20volume%20up%20and,problem%20among%20kids%20and%20teens.
5. https://www.jhucochlearcenter.org/sites/default/files/2020-12/Fact%20Sheet%20-%20Hearing%20Loss%20Prevalence%20(Cochlear%20Center%20for%20Hearing%20and%20Public%20Health)v1.pdf