Lip Reading using Neural Network and Deep learning

Karan Shrestha
Department of Computer Science
Earlham College
Richmond, Indiana 47374
kshres15@earlham.edu

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. This task is difficult as people use different dictions and various ways to articulate a speech. This project verifies the use of machine learning by applying deep learning and neural networks to devise an automated lip-reading system. A subset of the dataset was trained on two separate CNN architectures. The trained lip reading models were evaluated based on their accuracy to predict words. The best performing model was implemented in a web application for real-time word prediction.

KEYWORDS

lip reading, computer vision, deep learning, convolutional neural network, web application, object detection

1 INTRODUCTION

Lip reading is a recent topic which has been a problematic concern to even expert lip readers. There is a scope for lip reading to be resolved using various methods of machine learning. Lip reading is a skill with salient benefits. Enhancement in lip reading technology increases the possibility to allow better speech recognition in noisy or loud environments. A prominent benefit would be developments in hearing aid systems for people with hearing disabilities. Similarly, for security purposes, a lip reading system can be applied for speech analysis to determine and predict information from the speaker when the audio is corrupted or absent in the video.

With the variety of languages spoken around the world, the difference in diction and relative articulation of words and phrases. It becomes substantially challenging to create a computer program that automatically and accurately reads the spoken words solely based on the visual lip movement of the speaker. Even the expert lip readers are only able to estimate about every second word [7]. Thus, utilizing the capabilities of neural networks and deep learning algorithms two architectures were trained and evaluated. Based on the evaluation, the better performing model was further customized to enhanced accuracy. The model architecture with an overall better accuracy was implemented in a web application to devise a real-time lip-reading system.

2 BACKGROUND

This section provides the necessary background information to understand this research project better. It identifies Haar Feature-Based Cascade Classifier and Neural Network from the prominent researches and explains the reason for usage in my project for feature classification and training the lip reading model.

2.1 Haar Feature-Based Cascade Classifier

Paul Voila and Michael Jones introduced an efficient and robust system for object detection using Haar Cascades classifiers [23]. The Haar feature-based classifier is a machine learning based approach which is trained with cascade function using a set of positive and negative images. The Haar feature-based classifier is used to detect objects in the images. In my project, I have used Haar feature-based classifier to detect and track region of interest (ROI) - mouth region from the input speaker videos to preprocess the data before training the model with neural network architecture.

Initially, the algorithm required a set of positive images (frontal face with mouth visible) and negative images (non-frontal face) to train the classifier. Then, the specific feature (ROI) was extracted from the image. A feature was obtained in a single value by subtracting the sum of white pixels from the black pixels in the rectangle. Haar classifier detects the particular features (ROI) in an image based on three feature calculations - Edge features (Two rectangles), Line features (Three rectangles) and Four-rectangle features. The types of Haar cascade rectangle features are shown in Figure 1 [25].

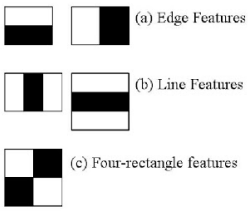

Figure 1: Haar cascade rectangle features

Haar feature-based classifier has three main characteristics - Integrated image, AdaBoost inspired learning algorithm and Attentional Cascade method.

Integral Image is an image obtained by the cumulative addition of pixels from the required region. The integral image at a location x,y contains the sum of the pixels to left and above the x,y region [23]. When detecting an object (ROI) in images an Attentional Cascade method is applied in each part of the image to provide an order for evaluation and checking features. It prevents computing on other features of the images. AdaBoost learning algorithm upgrades the classification performance of a simple learning algorithm by selecting a small set of features to train the classifier. Attentional Cascade method with AdaBoost ensures for redundancy computing to at a minimum level. Thus, these characteristics of Haar feature-based classifier allow it for object detection with a higher precision speed.

Wang et al. [24] proposed an approach for lip detection and tracking in real-time using a Haar-like feature classifier. Their method combined the primitive Haar-Like feature and variance value to construct a Variance based Haar-Like feature. This allowed the human face and lip to be represented with a small number of features. A Support Vector Machine (SVM) was used for training and classification, combined with the Kalman filter for real-time lip detection and tracking of the lip movement. As mentioned in the paper, the results of using Haar-like feature classifier was significantly better than other machine learning approaches to detect lip movements.

To accurately detect and track the lip movement for preprocessing the video data, I used Haar feature-based cascade classifier. Similarly, Haar classifier benefited my project from its high computation speed and precision in detecting and tracking of mouth region (ROI) of speakers from video inputs of Lip Reading in the Wild (LRW) dataset.

2.2 Neural Network

Neural Network (NN) is a computing system with roughly similar properties as the nervous system in human brains. The neural network is a framework of algorithms working together to identify the underlying relations in a dataset to provide the results in the best way possible. There are so-called hidden layers between input and output layers which have their separate functionality combined to perform specific tasks [8]. Specified by Hammerstrom, Neural Network was used in the various tasks of computer systems like image recognition, computer vision, character recognition, stock market prediction, medical applications, and image compression [6]. For my lip reading system, I use a specific type of deep learning Neural Network called Convolutional Neural Network.

Convolutional Neural Network

Convolutional Neural Network (CNN) is a class of Neural Network system in a standard multi-layered network. The layers comprise of a single or more layer connected in a multiple connection series. CNN is capable of utilizing the local-connectivity in high dimensional data such as datasets composed of images and videos. This feature permits the applicability of CNN in the field of computer vision and speech recognition. [13].

A basic form of CNN has four significant parts: convolutional layer, activation function, pooling layer, and fully connected layer. Convolutional layer uses a set of learning filters to learn the parameters from the input data. The activation function is a non-linear transformation function that defines the output of one node which is then the input for the next layer of neurons. In the pooling layer, the amount of parameters and computation in the network is reduced to control overfitting by decreasing the spatial size of the network. Fully connected layer takes the input volume from the convolutional layer or pooling layer to transform the result from the feature learning part to output.

Chung and Zisserman [5] provide two novel contributions in their paper. Firstly, Chung and Zisserman develop a pipeline of the automated system to collect data of videos for lip reading from TV broadcasts. Secondly, the paper uses a CNN architecture to learn and train the model from the collected dataset. The use of CNN in the model was to recognize individual words through the sequences of lip movements. The paper also discusses different data input techniques and the use of temporal fusion architectures to analyze and compare the efficiency of the CNN implementation. In this paper, 3D Convolution with Early Fusion (EF-3) architecture is introduced which I incorporate in my project. Likewise, this paper provides an analysis of the employment of the temporal sequences using models like Hidden Markov Models and Recurrent Neural Networks (RNN), which is less capable of adjusting with the motion of the image. This lessens the predicted accuracy of the trained models. However, the debate for using CNN is its applicability of use in the moving subject with higher precision. The result of the CNN model in their implementation showed a top-1 accuracy of 65.4%, which means the model accurately predicted the correct label of the word 65.4 times.

Similarly, Li, Yiting, et al. [15], significant research has been conducted for the dynamic feature extraction. Their research also integrates the deep learning model. The primary extent of their research was to use CNN to delimit the negative effects caused by unstable subjects in the video and images, and face alignment blurred during feature extraction. Their research provided a modification on the perspective for my project as their findings are significant, in terms of accounting for shaking or unstable objects in the dataset that I chose to train the model. A major constituent in their project was the use of a novel dynamic feature calculated using a collection of lip images and effectiveness of CNN architecture for word recognition. Their results showed 71.76% accuracy in recognizing words using a dynamic feature, which was 12.82% higher than the result obtained through conventional static features (composed through static images).

Noda et al. [17] applied CNN in their system of visual speech recognition, using the visual feature extraction mechanism. The CNN was trained by feeding the images of speakerâĂŹs mouth area combined with phoneme labels. A phoneme is a distinct unit of sound that collectively distinguishes a word in a language. The model acquired multiple convolutional filters extracting essential visual features that recognized phonemes. Further, their model accounted for temporal dependencies for the phonemes generated by label sequences which were handled by a hidden Markov model that recognized multiple isolated words. Their proposed system implemented CNN and a hidden Markov model for visual feature extraction, which significantly outperformed the models that acquired "conventional dimensionality compression approaches." The results demonstrated 58% recognition accuracy of phenomes achieved through image sequences of raw mouth area [17].

Hong et al. [10] proposed a novel network structure capable of reducing the computation cost and improving the accuracy for practical usage. This paper introduced lightweight deep neural network architecture which minimized the redundancy by adopting innovations like C.ReLU [20] and Inception structure [22]. I utilized their proposed lightweight model architecture as it achieved a reliable result compared to the current state-of-art in object detection.

From these existing research work, the use of deep learning CNN model showed a significant result and improvement in the automatic speech recognition. Thus, I integrated the CNN model in my project.

3 DATASET

This section describes the Oxford-BBC Lip Reading in the Wild (LRW) dataset [5] used for training and evaluation of the deep learning model in this project. LRW dataset is a large publicly available (upon request) dataset for non-commercial and academic researches. The dataset consists of short video clip segments of 1.16 seconds (approximately 29 frames). It comprises of hundreds of speakers videos from BBC programs, primarily from talk shows and news. Each video segment is clipped to record only the part where the speaker utters the word. Metadata is provided in the dataset to determine the start and end frames for each word duration in the video. A large volume of speakers complemented with variation in head poses, viewpoints (frontal to profile) and variant lighting conditions sets the dataset to be challenging and requires to be preprocessed to fit the training model.

This dataset contains 500 different word instances with up to 1000 utterances for each word spoken by different speakers. This dataset was generated using a multistage pipeline for a specific purpose of audio-visual speech recognition as described in [2]. The processing pipeline initially involved in video preparation by extracting the facial landmark using a linear Support Vector Machine (SVM) classifier, followed by audio and text preparation to force-align the subtitles to get a time-stamp of spoken word from each video. The unaligned audio and video streams were synchronized using Penn Phonetics Lab Forced Aligner [9, 27] due to the significant performance benefits over speech recognition methods.

The dataset was compiled into a training set, validation, set and test set based on broadcast dates of videos corresponding individual videos to one of the three sets. The training set consists of 500 words with more than 800 utterances of each word. Similarly, in both the validation set and test set, there is at least 40 video occurrence of each word.

Due to the limited computational resource and considerable easy usage of the web application, I only trained a part of the LRW dataset. Training the whole dataset within my computer architecture would take several days or weeks. I selected 19 words from the LRW dataset to train the lip reading model. The selected words have a noticeable difference in articulation and lip movement. I neglected the shorter words to include in the training dataset because of ambiguities produced by homophenes (for example, 'cell' and 'sell' have similar lip movement). The selected words from the LRW dataset are listed below:

ABOUT, ACCESS, ALLOW, BANKS, BLACK, CALLED, CONCERNS, CRISIS, DEGREES, DIFFERENT, DOING, EDITOR, ELECTION, EVERY, FOCUS, GROUP, HUMAN, IMPACT, JUSTICE

4 DATA PRE-PROCESSING

In the pre-processing stage, the speaker videos from LRW was initially used to detect the mouth - Region of Interest (ROI). An OpenCV python framework with Haar Feature-Based Cascade classifier was used to detect face and mouth region from each input videos. OpenCV is an open-source multi-platform library of programming functions focused on computer vision [2]. It provides many useful functions like video and camera input streams, object tracking, drawing tools, image cropping, and algorithm implementation of Haar Feature-Based Cascade classifier.

Solely focusing on the mouth area paced up the training process. Frontal and profile face Haar classifier detected face within each video frame, then, a mouth classier detected the mouth region. As seen in the first frame of Figure 2 the frontal face of the speaker is defined with a blue rectangle and the mouth region is recognized with a red rectangle. More information on feature detection with Haar Feature-Based Cascade classifier is provided in section 2.1.

Figure 2: Haar classifier detecting face and mouth, Median-Flow tracker tracking the mouth region

Median Flow tracker was initialized on the detected mouth area with slightly enlarged tracking area for better and more features to track. It prevented cropping parts of the mouth when the speaker moved in the video. It is an OpenCV tracking API (Application Programming Interface) that tracks objects in both backward and forward directions in time and measures the discrepancies between the two trajectories. This tracker enables to detect selected trajectories in the video and allows for tracking failures [16]. There are multiple tracking algorithms provided in the OpenCV library. From comparison and description provided in [16], I found about two robust algorithms for lip tracking - Median Flow Tracker and Kernelized Correlation Factor (KCF). However, using KCF and test processing on 200 sample videos ended up losing tracker in 83 samples running at an average speed of 115 ms/sample. Similarly, with the Median Flow tracker, the test processing concluded losing tracker on 67 videos running at an average rate of 84 ms/sample. Thus, I proceeded with Median Flow tracker due to relatively higher average processing speed. In Figure 3, the tracker was initialized from the second frame in the ROI with an enlarged tracking area,. It is shown by outlined with a yellow rectangle.

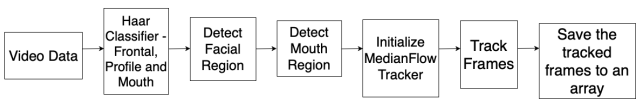

Figure 3: Lip Detection Process

Pre-processed Data Storage. The pre-processed data was stored to save time and not repeat the whole process. The frames of the tracked mouth region were cropped and converted to a grayscale image. Each frame was resized to a dimension of 24 x 32 pixels. A tensor with 28 frames produced size of 28 x 24 x 32 (depth x height x width). The tensor was converted and saved in a 3-dimensional(3D) NumPy array.

5 TRAINING

The lip reading model was trained with a CNN architecture. A 3-Dimensional (3D) CNN was applied to train the pre-processed lip samples and compare various parameters. Due to the 3D CNN architecture's capability of training with high dimensional data like image sequences. The saved pre-processed image data from training set could be loaded in 3D CNN to train the model. Two separate 3D CNN model architectures (EF-3 architecture and Lightweight Model architecture) as mentioned in Section 2.2, was initially trained for evaluation using the pre-processed data from the LRW dataset.

Both the architectures were implemented using Keras [4], an open-source neural network library written in Python. Keras combines high-performance computational functions with a relatively simple implementation of neural networks. Similarly, I used the Keras library to run on top of TensorFlow framework. TensorFlow is a python-friendly open-source framework providing numerical computation and dataflow programming for deep learning [1]. It allows for deployment of computation across multiple platforms - Graphics Processing Unit (GPU), Central Processing Unit (CPU) and Tensor Processing Unit (TPU). Keras with the TensorFlow backend was used to build the CNN models [4].

I constructed the EF-3 architecture based on VGG-M model as suggested by Chung et al. in 'Lip Reading in The Wild' [5]. Table 5.1 highlights the details of the parameters and various layers of EF-3 architecture.

Layer (type)	Output Shape	Param
conv3d	(None, 28, 24, 32, 48)	1344
max pooling3d	(None, 9, 8, 10, 48)	0
conv3d	(None, 9, 8, 10, 256)	332032
max pooling3d	(None, 3, 2, 3, 256)	0
conv3d	(None, 3, 2, 3, 512)	3539456
conv3d	(None, 3, 2, 3, 512)	7078400
flatten	(None, 9216)	0
dense	(None, 19)	175123
Total params: 11,126,355		
Trainable params: 11,126,355		
Non-trainable params: 0		

Table 1: EF-3 architecture

Initially, the input layer reads the images then, the convolutional layer (conv3d) learns the parameters. The term "parameter" refers to the learnable elements like weight matrices in a layer. The trained parameters from each convolutional layer are calculated using: ((kernel size x stride + 1) x filters, 1 is added to account for the bias term for each filter. As seen in table 1, in the first convolutional layer, the total trained parameters are ((3 x 3 x 3) x 1 + 1) x 48 = 6048. Since, the first convolutional layer has already learned 48 filters, the trainable parameters in the second convolutional layer are ((5 x 5 x 5) x 48 + 1) x 256 = 1536256 and so on. Supervised learning in CNN takes place as the parameters were trained and passed through each of the four convolutional layers. Total parameters learned during the training using this model architecture were 50,870,451.

To tune the hyperparameter, the number of epochs was set to 30, Adam optimizer algorithm [12] was used with the learning rate set to 0.0001, to speed up the training process. Training the model in EF-3 architecture ended after 16 epochs with a resulting test accuracy of 70.92%. The validation accuracy did not improve after 12 epochs providing a validation accuracy of 73.34%.

Similarly, I also experimented with a customized Lightweight model architecture /citehong2016pvanet to train the lip reading model using the pre-processed data. Using the lightweight model architecture the lip reading model trained significantly faster compared to the EF-3 architecture due to about 16 times less trainable parameters while maintaining the test accuracy at 66.56%. The details on the various layers and parameters are provided in Table 2. The structure of the lightweight model architecture is shown in Figure 4.

Layer (type)	Output Shape	Param
conv3d	(None, 28, 24, 32, 64)	1792
max pooling3d	(None, 9, 8, 10, 128)	0
conv3d	(None, 28, 24, 32, 128)	221312
conv3d	(None, 9, 8, 10, 128)	442496
max pooling3d	(None, 3, 2, 3, 128)	0
flatten	(None, 1152)	0
dense	(None, 19)	43759
Total params: 709,395		
Trainable params: 709,395		
Non-trainable params: 0		

Table 2: Model A1: Lightweight Model architecture

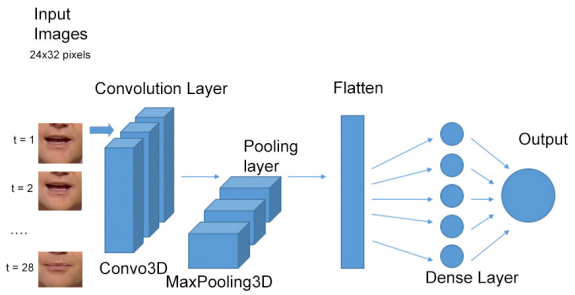

Figure 4: Lightweight model architecture

The total trainable parameters in the Model A1 were 709,395. In the first convolutional layer, 64 convolutional kernel filters of size (3x3x3) were used on input images which trained 1792 parameters. Kernel filters were set to 128 on other convolutional layers and max-pooling layers. Lesser kernel filter made a smaller convolutional layer which reduced the number of parameters in the network and

provided for regularization. As the Lightweight model architecture (Model A1) showcased a well-performing result with faster training neural networks, thus, I used this model as the base architecture for training the lip reading model to assigned the trained model weights in the web application.

In the base architecture (referred to as Model D) I added batch normalization layers to normalize each input channel across a mini-batch. Additionally, I also included dropout regularization [26] to prevent CNN to overfit the training data. The evaluation of the batch normalization and dropout regularization method is provided in section 6. The number of kernel filters used was the same as in the Model A1, but the kernel size was increased to (5x5x5). The trained parameters in Model D was only slightly higher than in Model A1. Training the pre-processed data with Model D provided a test accuracy of 77.14% and validation accuracy of 80.27%. Information on architecture differentiation, parameters, and various layers is shown in table 3.

Layer (type)	Output Shape	Param
conv3d	(None, 28, 24, 32, 64)	8064
batch normalization	(None, 28, 24, 32, 64)	256
max pooling3d	(None, 9, 8, 10, 64)	0
dropout	(None, 9, 8, 10, 64)	0
conv3d	(None, 9, 8, 10, 128)	1024128
conv3d	(None, 9, 8, 10, 128)	2048128
batch normalization	(None, 9, 8, 10, 128)	512
max pooling3d	(None, 3, 2, 3, 128)	0
flatten	(None, 2304)	0
dense	(None, 2304)	43759
Total params: 3,124,883		
Trainable params: 3,124,499		
Non-trainable params: 384		

Table 3: Model D: Lightweight architecture model with Batch Normalization and Dropout

6 EVALUATION

In the field of machine learning, evaluation of the model is a significant task. It is important to know if the trained model has learned patterns to generalize the prediction in unseen data to avoid overfitting on the lip reading model. For measuring the predictive accuracy of the model I performed a Top-1 accuracy, [14] on the test set of LRW dataset, so the word with the highest probability is the expected answer.

The evaluation of training each model is listed in Table 1; the testing accuracy was measured on a test set containing 877 samples of pre-processed data.

Kernel Size

In Model A2, I enlarged the kernel size in 3D convolutional layers to (5x5x5), comparing this to results of Model A1 with kernel size (3x3x3). The increased the kernel size considerably improved the validation accuracy from 66.56% to 70.29%. The comparison of Kernel sizes between Model A1 and A2 is provided in the Figure 5.

Model	Test Accuracy	Kernel Size	Epochs	Speed
EF-3	70.92%	3x3x3	16	190 ms
A1	66.56%	3x3x3	14	104 ms
A2	70.29%	5x5x5	14	110 ms
B	71.25%	5x5x5	17	115 ms
C	74.37%	5x5x5	12	120 ms
D	77.14%	5x5x5	25	117 ms

Table 4: Evaluation of model architectures

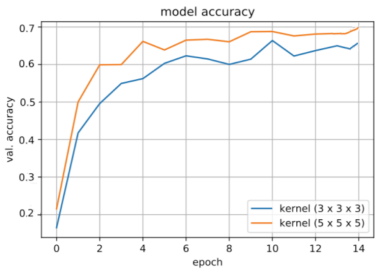

Figure 5: Comparison of Kernel Size between Model A1 and A2

Batch Normalization

Batch normalization is a method to normalize the internal representation of data to improve speed, performance and stability of the training in neural networks [19]. A few advantages of using batch normalization were stochastic optimization in training neural networks, regularized the model, mini-batches were jointly normalized, and parameters were learned per feature map, and an internal covariant shift is reduced [11]. Internal covariant shift is the covariant shift occurring within the neural network as the network learns and updates the weights with distribution of training output passing from one layer to another. The distribution of output takes more time in the higher layers which slows down the learning process. Thus, a reduction in the internal covariant shift along with a higher learning rate speeds up the learning.

Normalization was performed for each mini-batch before feeding the data into each neural network. A batch size of 16 was set for each mini-batch. Data was converted from the Numpy array of values between 0 and 255 to type float16 between the range -1 and 1. Model A2 does not include batch normalization, whereas Model B (built on top of Model A2) has a batch normalization layer. The comparison between the two models shows around 1% improvement in validation accuracy du toe to batch normalization. There is only a slight increase in accuracy as a subset of the LRW dataset is used to train the models. However, a significant impact can be measured using the whole LRW dataset. Table 5 compares the accuracy of the two models.

Dropout

Dropout is a regularization technique which helps to reduce overfitting and improve the generalization of neural networks. I included

Model	Val. Accuracy	Kernel Size	Batch Normalization
A2	70.29%	5x5x5	No
B	71.25%	5x5x5	Yes

Table 5: Evaluation of Batch Normalization with Model A2 and Model B

dropout regularization in my training model considering the suggestion provided by Srivastava et al. [21].

Model C and Model D were trained with dropout regularization. As seen in Table 6, in Model C with a dropout rate of 30%, the validation accuracy improved by more than 4% compared to Model B (without dropout). Similarly, using even a higher dropout rate of 40% in Model D, the validation accuracy increased by around 3%. In both of the Models C and D, the validation accuracy was higher than the testing accuracy which shows the overfitting got successfully prevented. Thus, dropout regularization helped in improving the overall model accuracy.

Model	Val. Accuracy	Test Accuracy	Dropout rate
C	74.37%	69.17%	30%
D	77.14%	70.35%	40%

Table 6: Evaluation of Dropout with Model C and Model D

Confusion Matrix

I generated a confusion matrix for the model with the best accuracy - Model D. The confusion matrix plot of the trained words is shown in Figure 5. A confusion matrix is a tool to summarize the performance of a classification algorithm. I implemented the confusion matrix using sci-kit-learn machine learning library [18]. Confusion matrix provided a robust evaluation of the trained model. The model was evaluated based on the classes that model predicts incorrectly comparing it the words that are being confused. The percentage number in each cell represents how much a particular word is being confused with another word. A higher confusion rate is shown with a darker background.

7 WEBSITE APPLICATION

A website (web) application was built to demonstrate the performance of the trained model. It allowed the user to view the predicted results of the lip articulated words. Web camera took the video input of the speaker similarly as when preprocessing the training data. Head and mouth were detected and tracked on each frame; then, the cropped area of lips was held in an array. A specific number of frames were counted from the array then passed to the model for prediction. The results were plotted in a bar plot to show the prediction of the words.

Web application allowed the model to run on the client side, thus utilizing the user's graphic card. This was beneficial as there was an improvement in speed for predicting the words, user's data was protected, and the server load got minimized. The user and server had to exchange the data only when the website was accessed. The model architecture and the trained weights would be

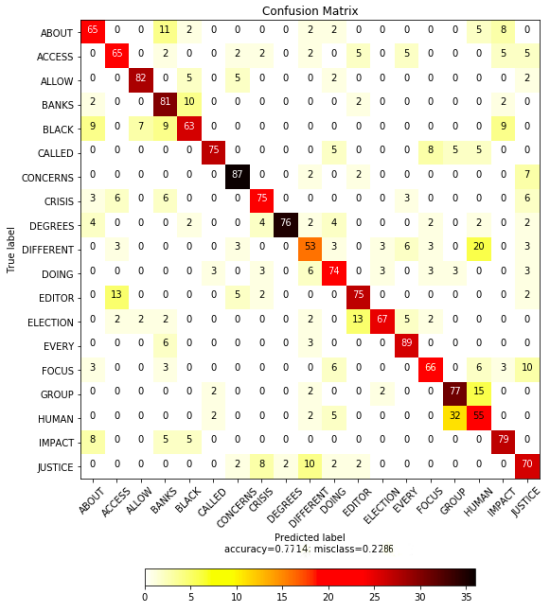

Figure 6: Confusion Matrix of Trained Words

downloaded on the client side and run in the user's architecture. Figure 6 provides an illustration of this process.

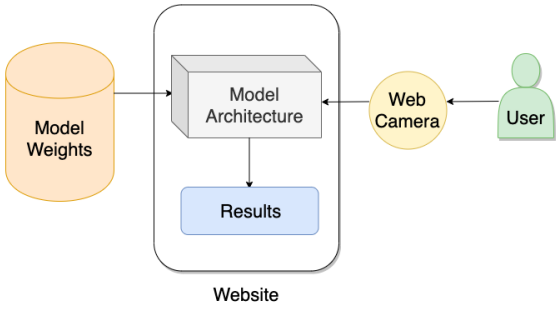

Figure 7: Design of Web Application

I choose Keras.js [3] library to deploy the model in the web application as the model was trained in Keras library. Keras.js used a custom protocol buffer format binary file which was easily converted from the original weight files. Similarly, I used clmtrackr.js [cite] to track the face from the web camera video input, and the face coordinates were mapped as closely as possible. The resulting tracked array of lip frames were normalized, and the resulting vector was fed into the model for word prediction. After the model completed prediction, an output vector probability of each word was displayed in a bar plot for user review. I used Chart.js [cite] to construct the chart for a bar plot and display the prediction results. The front-end of the web application is shown in Figure 7.