REAL-TIME FINGER SPELLING AMERICAN SIGN LANGUAGE RECOGNITION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Lokesh Kumar Viswavarapu

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2018

Viswavarapu, Lokesh Kumar. *Real-Time Finger Spelling American Sign Language Recognition Using Deep Convolutional Neural Networks*. Master of Science (Electrical Engineering), December 2018, 50 pp., 45 figures, 30 numbered references.

This thesis presents design and development of a gesture recognition system to recognize finger spelling American Sign Language hand gestures. We developed this solution using the latest deep learning technique called convolutional neural networks. This system uses blink detection to initiate the recognition process, Convex Hull-based hand segmentation with adaptive skin color filtering to segment hand region, and a convolutional neural network to perform gesture recognition. An ensemble of four convolutional neural networks are trained with a dataset of 25254 images for gesture recognition and a feedback unit called head pose estimation is implemented to validate the correctness of predicted gestures. This entire system was developed using Python programming language and other supporting libraries like OpenCV, Tensor flow and Dlib to perform various image processing and machine learning tasks. This entire application can be deployed as a web application using Flask to make it operating system independent.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my graduate advisor Dr. Kamesh Namuduri and Dr. Bill P. Buckles for all their guidance and support. I would also like to thank my committee members Dr. Xinrong Li and Dr. Parthasarathy Gutur for their support to my thesis. And a very big thanks to my parents Shri. Venkata Subba Rao Viswavarapu and Shri. Siva Durga Rani Viswavarapu for all your support to pursue my passion and thank my friends for standing with me during all my hurdles.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Introduction

Spoken language is one of the unique communication techniques of humans, which has not just enabled a vital medium of expression but also plays a crucial role in maintaining a social coherence. Any physical abnormality causing hindrance to speaking makes it impossible for them to use vocal language for communication. To overcome this, a visual communications system called sign language which uses body (mostly hand) gestures to translate expressions into verbal language was developed. Finger-spelling American sign language (ASL) is one such standard sign language which uses hand gestures to represent letters in English alphabet.

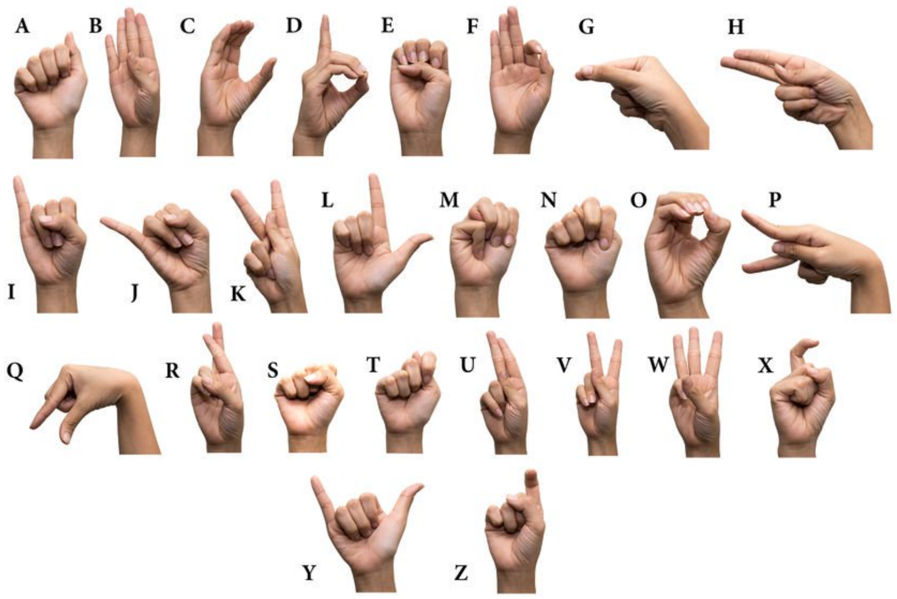


Figure 1.1. Finger Spelling American Sign Language

Source: www.daytranslations.com

1.2. Motivation

Even with this alternative form of communication, it is very challenging to have an ease of conversation with the general public, as understanding American Sign Language needs a lot of training and practice and is pursued as a special skill by a very small set of people on a need basis [25]. In consequence, this gap is creating a lot of social isolation among people with disability of hearing and speaking in the society they co-exist. To bridge this gap there is need for a robust assistive technology which can recognize ASL hand gestures and translate them into natural language.

With a strong motive of making artificially intelligent personal assistant technology available to specially-abled community for overcoming their challenges, we have developed an American Sign Language hand gesture recognition system using deep learning. In this project we have perceived hand gesture recognition as an image recognition problem and trained convolutional neural networks (CNNs), using transfer learning to perform the recognition task. Along with the gesture recognition we integrated head pose estimation as a feedback system to validate the correctness of the gesture recognized to mitigate the misclassification errors in the real-time.

1.3. Hand Gesture Recognition

Hand gesture recognition is an ability of computing devices to understand human hand gestures with help of sophisticated mathematical algorithms. Computer vision based hand gesture recognition is a popular human computer interaction technique, as it is very near to natural human interaction.

The computer vision based hand gesture recognition is a well explored technique in the research community with numerous implementations using conventional image processing techniques like skin detection, hand segmentation and hand motion tracking. But, there is very less transformation of this research into a real-world application as it is very challenging to attain robustness in highly varying lighting conditions and wide variation in the skin color based on the ethnicity. With the recent advancements in the artificial intelligence and deep learning, Computer vision domain (especially image recognition) is progressing very rapidly after 50 years of its birth[21]. One such deep learning algorithm called Convolutional Neural Networks is incorporated in our system to perform hand gesture recognition.

1.4. Deep Learning and Transfer Learning

Deep learning is a rapidly growing special class of machine learning which is developed by cascading non-linear processing units in the form of several layers which perform operations like transformation, feature extraction and decision making. The algorithms developed using deep learning can solve complex signal processing problems like Speech recognition, Image classification, Natural Language processing and Machine Translation [30].

As this model has a large number of processing units, it requires a very high computational resource while performing training. In practice, parallel processing techniques are adapted by using GPU clusters for learning the training dataset. An alternative for running such massive models with a limited computational power is using Transfer Learning.

Transfer Learning is a method of transferring the hyperparameters of pretrained networks which are trained for weeks on multiple GPUs and has the ability to classify the input data over 1000 classes. This method mitigates the tedious job of building a deep network from the scratch. And, the decision making in machine learning is a data driven task, transfer learning can be adapted in the case of having a limited amount of training data and considerably large number of class labels. In this project, we used MobileNet CNN Model for transfer learning which is trained with 1.2 million images with 1000 classes[11].

1.5. Convolutional Neural Networks

Convolutional Neural Network is a deep learning technique which is developed from the inspiration of visual cortex which are the fundamental blocks of human vision. It is observed from the research that, the human brain performs a large-scale convolutions to process the visual signals received by eyes, based on this observation CNNs are constructed and observed to be outperforming all the prominent classification techniques.

Two major operations performed in CNN are convolution ($w^T * X$) and pooling ($\max()$) and these blocks are wired in a highly complex fashion to mimic the human brain.

The neural network is constructed in layers, where the increase in the number of layers increases the network complexity and is observed to improve the system accuracy. The CNN architecture consists of three operational blocks which are connected as a complex architecture. The functional blocks of Convolutional Neural Network:

Convolutional Layer

Max Pooling layer

Fully-Connected layer

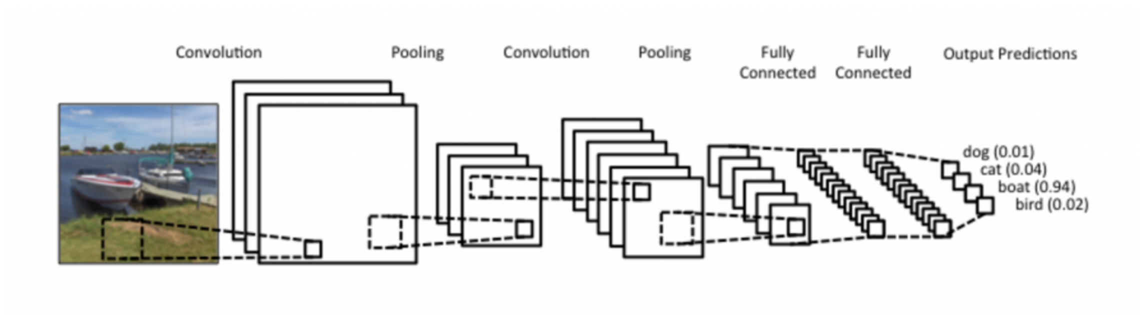


FIGURE 1.2. Convolutional Neural Network

Source: www.clarifai.com

1.6. Thesis Outline

The organization of this thesis report is as follows,

Chapter 1 Introduction: In this chapter we introduce the problem statement we are addressing, our motivation to choose this problem and a preview of our solution.

Chapter 2 Significant Challenges: This chapter aims at introducing the challenges involved in developing a robust hand gesture recognition system.

Chapter 3 Literature Review: This chapter provides a brief summary of our study on related work in literature.

Chapter 4 System Design: This chapter gives an overview of our project pipeline and brief explanation of each module and their significance in our system.

Chapter 5 Implementation: In this chapter, we discuss about implementation of each module in our system. This chapter deals with algorithms, techniques, technologies and computational platforms incorporated and integrated in this project.

Chapter 6 Results: In this chapter, we discuss about the performance of our system which is measured using various standard metrics like accuracy, cross entropy and confusion matrix.

Chapter 7 Discussion: In this chapter, we discuss about how this solution address our problem statement, our research innovations in developing this solution and limitations of our system.

Chapter 8 Future Work: The scope for improving the performance and adding more features in our system is discussed in this chapter.

Chapter 9 Conclusion: This chapter provides a synopsis of our work.

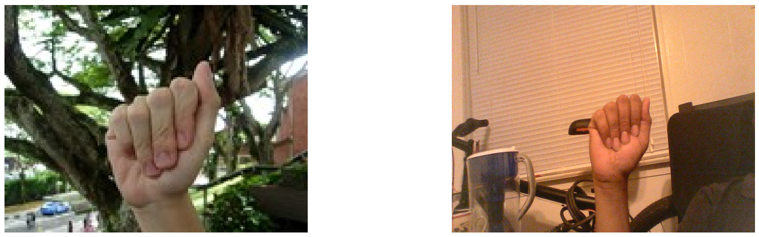
CHAPTER 2

SIGNIFICANT CHALLENGES

In this project, we have developed a visual recognition system which uses camera sensor to capture the real-time hand gesture, computer vision techniques to process the captured frames and a trained deep learning model to recognize the gesture. We have experienced some significant challenges at each stage of design and implementation, which will be discussed in this chapter.

2.1. Uncontrolled Environment

The uncontrolled environment is one of the major concerns of developing any real-time computer vision project. The dynamic changes in illumination, background and camera positions will make it complex to develop a technique to extract the regions of interest from the entire frame. In our system, the ambient lighting conditions and camera position have a great impact on face and hand localization.



(A) Hand gesture with fair skin-tone and variable background

(B) Hand gesture with a brown skin-tone and poor lighting conditions

Figure 2.1. Example Images of Hand Gestures with Variable Background and Lighting Conditions

2.2. Robust Hand Detection

Detection and localization of hand region is a very complex task. Incorporation of most common practices like motion or skin based detection is not an accurate approach as the motion in the background is uncontrolled and extracting skin region based on a threshold is impractical because of various skin tones based on the ethnicity. An object detection based approach is also not robust technique as the hand can transform into various sizes and shapes because of its moving fingers.

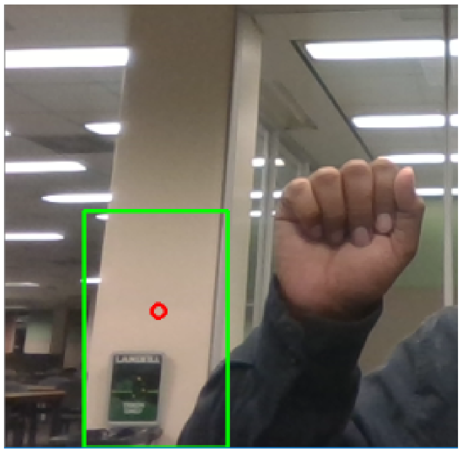
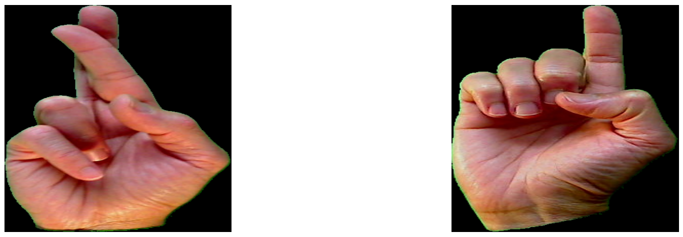


FIGURE 2.2. False Detection of Hand Region in Skin Based Detection Technique

2.3. Occlusion

As image is a two-dimensional projection of three-dimensional world, occlusion is observed to be a very common obstruction in accurate image recognition. In ASL recognition, few signs obstruct the visibility of certain parts of hand and figures which will create ambiguity in image classification task.

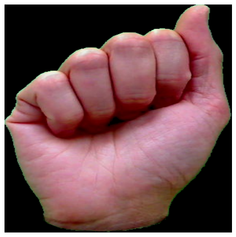


(A) ASL gesture for 'R'   (B) ASL gesture for 'D'

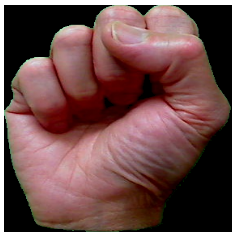
FIGURE 2.3. Example of Occlusion: R Gesture can Look like D in 2D Projection because of Occlusion

2.4. Low Inter-Class Variability

Maintaining a high variability among inter-class data and low variability among intra-class data is a general practice in machine learning classifier training as it reduces misclassification errors. But few signs in American sign language has very high visual similarity which can negatively impact the accuracy of the system.



(a) ASL Gesture for 'A'



(b) ASL Gesture for 'S'

Figure 2.4. Example of Low Inter-Class Variability: A gesture can be Mis-Classified as S because of Low Inter-Class Variability

2.5. Trigger to Recognition Process

In real-time systems, it is important to have a trigger for initialization and termination of processes. There is practice of using hot-words in voice assistants to start the speech recognition process and in a visual recognition system like this, a vision-based trigger to initiate the image recognition process is required, whereas the process can be terminated based on the idle time.

CHAPTER 3

LITERATURE REVIEW

The hand gesture recognition is a well contributed research area with a lot of different approaches in implementing it, in this chapter, we review a variety of such approaches for hand gesture recognition. Our entire literature review on hand gesture recognition can be categorized into three groups, image processing/statistical modelling based recognition, classic machine learning based recognition and deep learning based recognition.

3.1. Image Processing/ Statistical Modelling based approaches

These approaches either involve image processing using conventional image filters or pursued gesture recognition as action recognition problem and solved through extracting spacio-temporal features and developing models like Hidden Markov Models (HMM) or Finite State Machines (FSM).

[29]Triesch and Malsburg in 1996 developed an ASL hand gesture recognition system, aiming at performing accurate gesture recognition even for images captured with complex background. In this system, they have eliminated hand segmentation assuming hand images input. For hand representation or feature extraction, the system uses Gabor filters as the Gabor filters are known to resemble the receptive fields of visual cortex. Upon obtaining the Gabor features, they performed a similarity matching technique for gesture recognition called Elastic Bunch- Graph Matching (EBMS) technique. For experimentation they have considered a subset of 10 gestures from American sign language and the dataset consists of 657 images captured from 24 people in three different backgrounds (Complex, uniform light and uniform dark). The system has achieved an accuracy of 86.2% under complex background condition and overall accuracy of 91% under all background conditions. Many researches have taken the path of HMM based modeling for hand gesture recognition pursuing it as action recognition problem.

Bergh et al. [8] proposed a hand gesture recognition system using Haar wavelets and database searching where Haar classifier is used to segment the hand features which are used to retrieve the hand gestures from the pool of pre-labeled gesture images in the database. Even though the accuracy of both the systems were considerably good, these systems could classify only six gesture classes.

[3]Ashutosh et al. in 1998, developed a HMM based hand gesture recognition system to control a robot. In this system they have obtained shape and optical flow based features to build a HMM to recognize twelve different type of gestures. The authors claim, the proposed system achieves 90% accuracy on testing with around 50 gesture inputs.

[14]Kapuscinski and Wysocki in 2001 have developed a gesture recognition system to recognize digits 0-9. They have performed skin color based hand segmentation using normalized color space and estimated hand postures using Hit-miss transform. These hand postures are used to train two, three and four state HMMs with a training data of 600 gestures and has yield a training accuracy of 98% on training data and 97.2 % accuracy on test data.

There were a few approaches using Bayesian networks which achieved high accuracies in case of temporal gesture recognition. Liwicki et al. [17] work on British Sign Language recognition system using Histogram of Gradients (HOG) descriptors and Hidden Markov Models (HMM), Starner et al. [28] developed a gesture recognition system using 3-D glove for hand motion tracking and HMM for gesture recognition and achieved an accuracy of 99%. A Dynamic Bayesian Network (DBN) based model for temporal hand gesture recognition like identifying circles or waving hands has been proposed by Suk et a. which achieved an accuracy of 99%.

3.2. Classic Machine Learning based approaches

In these approaches, authors used classical machine learning techniques like Support Vector Machines (SVM) classifiers, Artificial Neural Networks (ANN) classifiers and Fuzzy-c-mean clustering. In these classic machine learning based approaches, feature engineering plays a major role in the system design. A variety of feature extraction and classification/clustering techniques and their performance in gesture recognition are briefly discussed in the category.

[2]Amin and Yan in 2007 proposed a gesture recognition system using Gabor features with Fuzzy-c-mean clustering for ASL recognition. In this system, they have eliminated hand segmentation by cropping the hand region manually and the captured dataset consisting of hand images with uniform light background. Gabor filters are applied to these images followed by Principal Component Analysis (PCA) for dimensionality reduction as a part of feature extraction and extracted features are used in training Fuzzy-c-mean clustering algorithm. This system has performed 93.2% accurate in cross-validation test conducted under experimentation.

[12] Huang et al. in 2009 has taken a similar approach of using Gabor with PCA for feature extraction and trained a Support Vector Machine for classification. For this task they have chosen a set of 11 hand gestures and generated a dataset of 1320 images using 11 signers. For hand segmentation they have incorporated skin color based segmentation in normalized RGB color space. The system has yielded an accuracy of 95.2% when tested with 660 set of images.

Gopalan Gopalan and Dariush in 2009 has developed an SVM based classifier by introducing Inner Distance Shape Context (IDSC) descriptors for feature extraction. In this technique, they have performed skin color based segmentation where a gaussian probabilistic model are built for filtering skin pixels in YCbCr color space and IDSC is applied to the segmented hands for feature extraction. Inner Distance Shape Context (IDSC) is used to compute the similarity and point correspondence between different shapes. These features are used to train a SVM classifier with 8 gesture classes. This technique has attained an accuracy of 85%.

[26] In this system Savur and Sahin use Surface Electromyography(sEMG) signals with SVM classifier for American sign language recognition. The surface EMG signals are obtained by obtaining signals by placing external sensors on the hand. Features like Mean Absolute Value (MAV), Simple Square Integral (SSI), Average Amplitude Change (AAC) are extracted from the signal and used for training a SVM classifier. The system is trained to recognize 26 ASL gestures and resulted a training accuracy of 95% and test accuracy of

91.73%.

[24] Otiniano-Rodriguez et al. has developed a ASL recognition system using Kinect depth camera. In this system, depth map is used for hand segmentation and Scale Invariant Feature Transformation (SIFT) is applied to hand images to extract features for training SVM. This system with a combination of RGB and depth map images has resulted an accuracy of 91%.

[6]Chuan et al. had a similar approach but used leap motion sensor instead of cyber glove or Kinect sensor. The leap motion sensor data collected data is used to train both K-Nearest Neighborhood classifier and SVM with round 7900 images. This technique achieved an accuracy of 72.7 and 79.7% of accuracies for KNN and SVM respectively.

[7] Dardas et al. have incorporated Bag-of-features in SIFT with SVM gesture recognition system. In this system, the features obtained from SIFT are vectorized using vector quatization(VQ) and key points are extracted using k-means clustering. These key points also known as code vectors are used for building codebooks which are used for training SVMs. The SVM is trained to recognize five fingers of the hand and has resulted an average accuracy of 90%.

[16] Lamari et al. has proposed a gesture recognition system using Artifical Neural Networks. In this system, colored glove is used for easy hand segmentation and morphological Principal Component analysis is applied on the hand images for feature extraction. The Neural network used for classification is a 3 layered feed forward perceptron network with sigmoid activation function, trained using back propagation algorithm. The network architecture consists of 20 or 24 neurons (depending in dimensionality) and hidden layer contains 42 neurons and output layer contains 26 neurons for 26 ASL finger spelling gesture classification. This trained neural network has yielded a training accuracy of 97.94% and test accuracy of 89.06%.

[19] Mekala et al. used a neural networks based ASL recognition which used Gaussian average model based background subtraction for hand segmentation. Shape descriptors and motion vectors are extracted from segmented frames and are used for training a three layered

neural network. The authors claim to obtain 100% recognition rate on recognizing gestures from A-Z.

[1]Admasu and Raimond have developed an Ethiopian Sign Language (ESL) recognition system using neural networks which has produced a recognition rate of 98.5%. In this system, they used manually preprocessed hand images and performed some basic hand segmentation for obtaining hand region. Gabor filters are applied to extract features and train a three layered neural network. The network architecture consists of 20 input neurons, 100 hidden layer neurons and 34 output neurons.

3.3. Deep learning based approaches

Nagi et al. [20] proposed a gesture recognition system for HCI using CNNs in 2011.There was research on recognizing sign languages are used in different countries or regions, like Pigou et al. [23] proposed an Italian sign language recognition system using CNNs which reported 95.6% accuracy for 20 classes on the training set but performed poor for test set.

[13] Kang et al. have developed a American sign language recognition system using depth map images trained on convolutional neural networks. In this system they have used 3100 depth maps for 31 gestures which includes all the alphabet and digits except for J, Z as they require temporal detail and 2/V, 6/W were assigned with single class label. For training they have chosen Alexnet architecture and achieved 99% validation accuracy and 85% accuracy on test dataset.

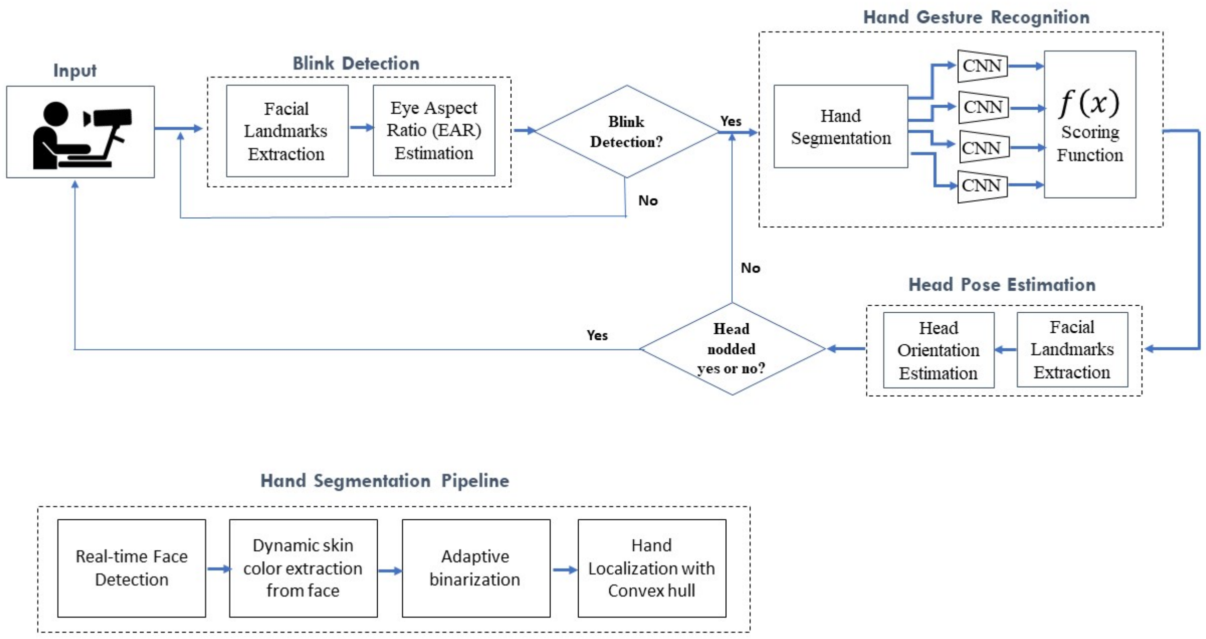
[5]Bheda and Radpour also developed a CNN based ASL gesture recognition system which classifies all the ASL alphabet and digit gestures. In this system they have used both existing NZ ASL dataset and their own dataset of all the gestures. The system has resulted a validation accuracy of 82.5% on alphabet gestures and 97% accuracy on ASL digit gestures.

[9]Garica and Viesca has implemented a similar CNN based ASL recognition system. The system uses googlenet trained with Surrey nad Massey university datasets using transfer learning technique. The model was trained to recognize A-Y excluding J and the authors

observed that recognition for A-E classification has performed the best when compared to other their models which are trained to recognized A-K and A-Y (excluding J).

CHAPTER 4

SYSTEM DESIGN



Input
Blink Detection
Facial Landmarks Extraction
Eye Aspect Ratio (EAR) Estimation
Blink Detection?
Yes
No
Hand Gesture Recognition
Hand Segmentation
CNN
CNN
CNN
CNN
$f(x)$ Scoring Function
Head Pose Estimation
Head Orientation Estimation
Facial Landmarks Extraction
Head nodded yes or no?
Yes
No
Hand Segmentation Pipeline
Real-time Face Detection
Dynamic skin color extraction from face
Adaptive binarization
Hand Localization with Convex hull

Figure 4.1. Our Hand Gesture Recognition System

4.1. Input

The input unit consists of a camera sensor which captures the live frames containing, frontal face and hand performing gestures and feeds them to the cascaded units for processing. This project aims at performing gesture recognition on the native RGB frames captured from a web camera instead of using augmentation techniques like depth cameras or specially designed gloves for hand detection.

4.2. Blink Detection

As it is very essential for a real-time recognition system to have trigger for initiation and termination of recognition process. We have developed a blink detection module to trigger the hand gesture recognition pipeline. Blink detection is a two-stage process, Facial landmark detection: Facial landmark detection is a technique obtain key points from a facial image which are referred as landmarks. Using these facial landmarks, we can localize different parts of a face like eyes, nose and lips and use their shape information for various facial analysis. In this project we have extracted eyes from the face using landmarks and performed blink detection.

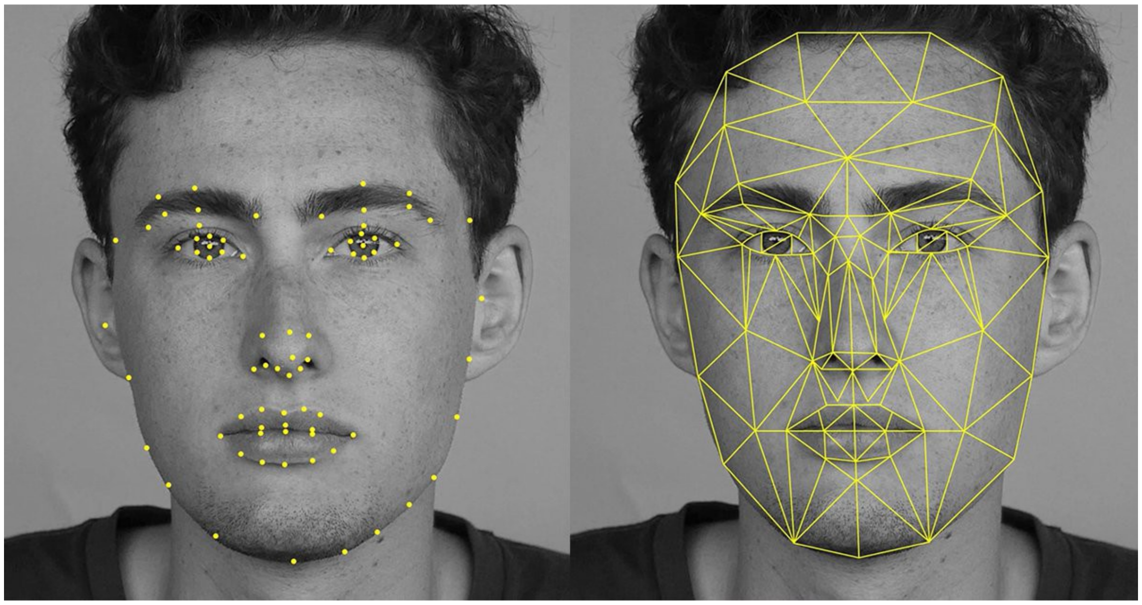


Figure 4.2. Facial Landmarks

Source: http://blog.murraycole.com

Eye Aspect Ratio estimation: Eye Aspect Ratio (EAR) is a coefficient obtained by computing the ratio of height and width of the eye (pixels) in the given image frame. This coefficient is used as a threshold for identification of a blink.

4.3. Hand Gesture Recognition

In our design, hand gesture recognition is also a two stage process, hand detection/segmentation and hand gesture classification. For hand detection we have employed a skin color filter with the thresholds obtained from face pixels. This technique is observed to achieve desired accuracy along with fastness. For gesture recognition we have trained a bank of CNNs and ensembled their classification output in a bagging approach. This is called ensemble learning, an innovative adaption from classic machine learning practices. A brief description of ensemble learning is provided below.

Ensemble Learning: As the name describes, ensemble learning is a technique of combining a group of machine learning models which are trained to perform same task. This technique proved to boost the performance of classification, as it reduces the chance of model overfitting during training and mitigates the occurrence of bias and variance conditions which will have a negative impact on the performance of the classifier on a new distribution of data. Bagging (Bootstrap Aggregation) is one of the popular and simple approaches to ensemble a group of classifiers. In bagging prediction scores of each classifier are aggregated to obtain a final score and this score is used to assign the predicted class label.

Even though ensemble learning is a proved technique to improve the performance of classification, it is not much explored in the deep learning domain. In our system we have employed this technique as a research innovation and its implementation and significance in improving the gesture recognition are discussed in the future sections.

4.4. Head Pose Estimation

In this system, users can validate the correctness of predicted gesture by nodding their head vertically or horizontally as a sign of ”yes” or ”no” respectively. These head poses are estimated using computer vision and geometric technique. The process typically involves face detection, facial landmark extraction and landmark orientation approximation to horizontal and vertical axis.

CHAPTER 5

IMPLEMENTATION

5.1. Blink Detection

Blink detections acts as a trigger to initiate the gesture recognition process followed by head pose estimation. Implementation of blink detection in this system involves two crucial step which will briefly be discussed in this section.

5.1.1. Blink Detection Flow

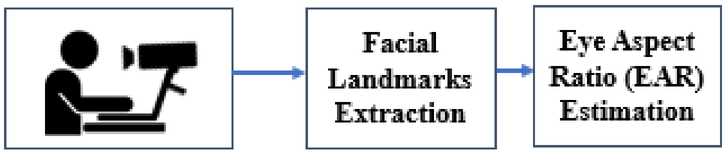

Facial Landmarks Extraction
Eye Aspect Ratio (EAR) Estimation

Figure 5.1. Blink Detection Pipeline

5.1.2. Facial Landmark Extraction

In our system, Facial landmark extraction is implemented using an opensource object detection library called Dlib[15]. It provides a robust trained model for frontal face detection and facial landmarking. This dlib model is trained over a large set of Histogram of Oriented Gradients (HoG) of faces using Support Vector Machine (SVM). Compared with any sophisticated deep neural network based model, this trained model is very light-weighted and highly compatible for our real-time system as it adds negligible latency while processing input frames. In this blink detection process, we detect all the faces in a given frame and lock the closest face to the camera and perform facial landmarking for the face using dlib. Facial landmarking of detected frontal face yeilds 68 landmark coordinates on the face as shown in the figure below.

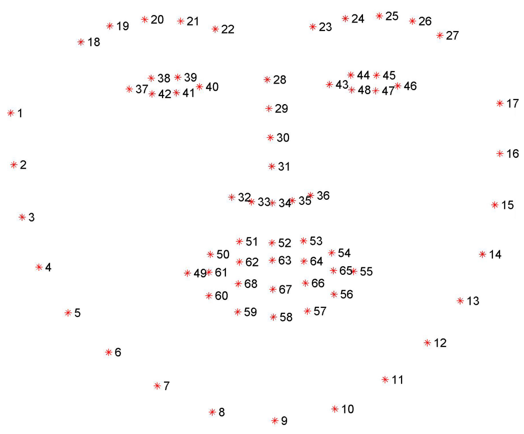


FIGURE 5.2. 68 Facial Landmark Points

Source: www.pyimagesearch.com

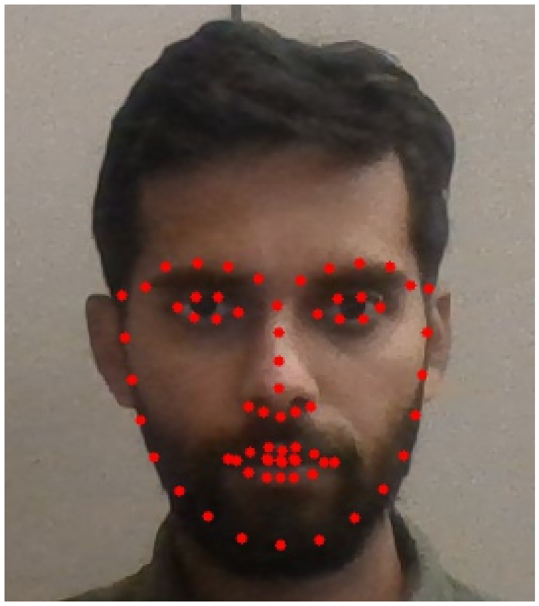


FIGURE 5.3. Landmark Points Depicting Different Regions of a Face

From 68 facial landmark coordinates, 6 points which can be used to obtain the width and height of each eye are obtained to calculate Eye Aspect Ratio (EAR) which will be discussed in the following section.

5.1.3. Eye Aspect Ratio Estimation and Thresholding

Eye Aspect Ratio (EAR) is a single scalar quantity proposed by Soukupov and ech in 2016 for eye blink detection. EAR is the ratio of horizontal and vertical distances calculated using six cartesian coordinates from 68 facial landmarks which represent the eye region. The EAR is formulated as follows,

$$\mathbf{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Where $P_1, P_2, P_3, P_4, P_5, P6$ are Facial Landmark Points Depicting Eye Region [27].

In the above formula, the numerator computes the Euclidian distance between the vertical landmarks where as denominator computes the distance between horizontal eye landmarks. EAR significantly falls and reaches zero when the eyes get closed and this feature helps identifying blink detection.

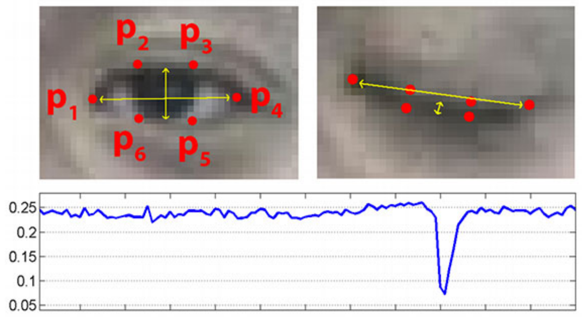


FIGURE 5.4. Open and Closed Eyes with Landmarks P_i . The Eye Aspect Ratio (EAR) Plotted for Several Frames of a Video Sequence [27]

5.2. Hand Segmentation

The hand gesture recognition triggered by blink detection is a two stage process, hand segmentation and gesture classification. Hand segmentation is the most crucial part of the system as the inaccuracies in hand segmentation effects the overall performance of the system, the segmentation process involves, localizing the hand region that is enacting the American sign language and extracting it for gesture recognition. In this section, the steps involved in implementing a robust hand segmentation technique is briefly discussed.

5.2.1. Hand Segmentation Flow

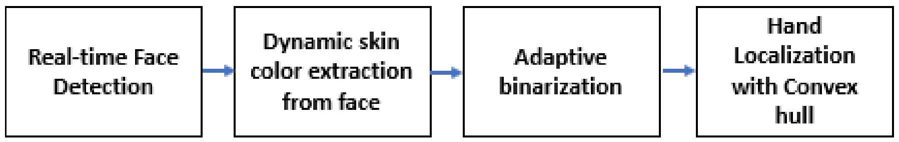

Real-time Face Detection
Dynamic skin color extraction from face
Adaptive binarization
Hand Localization with Convex hull

Figure 5.5. Hand Segmentation Pipeline

5.2.2. Skin Color Extraction from Face

We have adopted color based hand segmentation technique in our system. It is experimentally observed that transforming image frames from RGB to YCbCr color space and applying skin color thresholds for each channel has yielded better results compared to HSV or any other color space. To filter the skin region we have implemented an adaptive skin color thresholding technique. In this technique, a patch of pixels from nose region are extracted from real-time frames using facial landmarks detection and a mean image is computed from the patch and transformed into YCbCr color space. We obtain the upper threshold and lower threshold by adding a permissible offset of ± 100, ± 15, ± 15 for mean image's Y,Cb,Cr channels respectively.

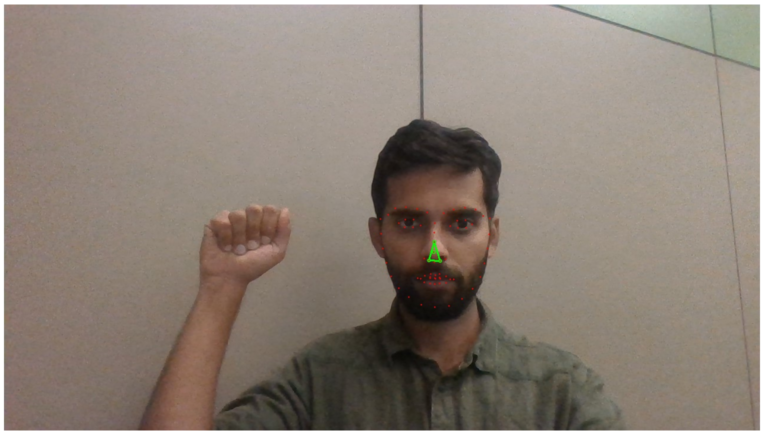
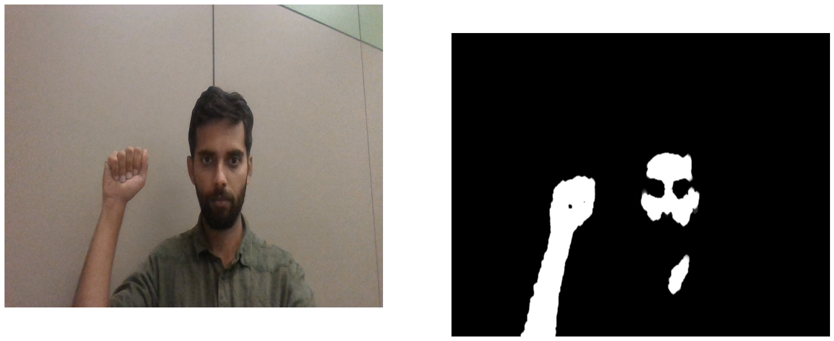


Figure 5.6. Nose Pixels Extracted Using Facial Landmarks for Computing Skin Color



(a) Original Image Captured from General Purpose Camera

(b) Mask for Skin Region Extraction

Figure 5.7. Skin Region Extraction using Skin Color Offsets from Nose Patch

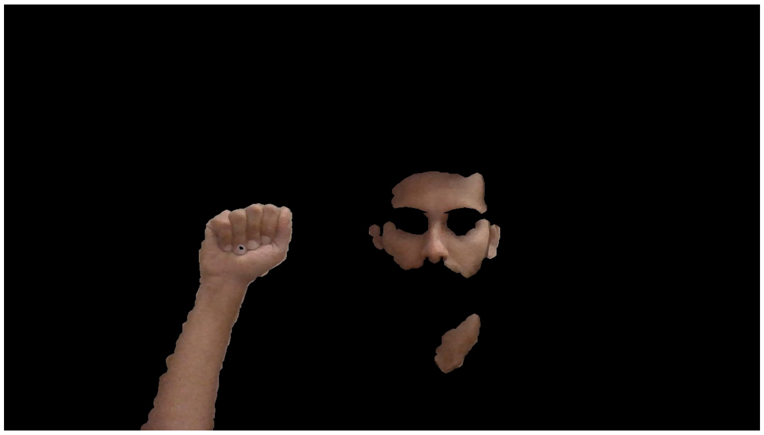
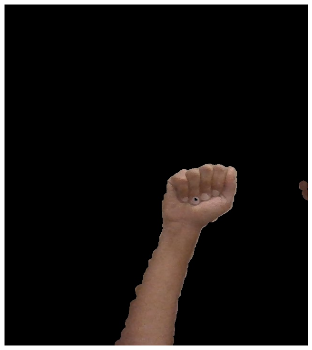


Figure 5.8. Image after Filtering Non-Skin Pixels

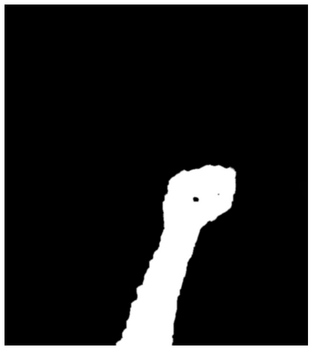
5.2.3. Convex Hull for Hand Segmentation

On filtering skin color pixels from the entire frame, we binarize the image using the popular Otsu's binarization technique and perform a series of morphological operations to obtain a perfect contour of hand region. After contouring all the skin regions, hand segmentation is performed using Convex hull algorithm.

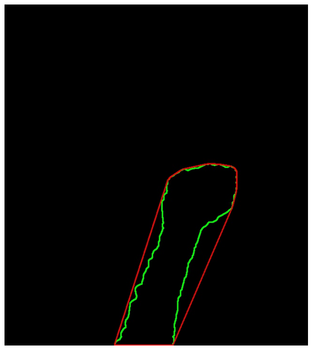
For a given set of Euclidian points, convex hull algorithm computes the smallest convex polygon that circumscribes all the given points. In our system, the convex hull algorithm is used to obtain the smallest convex polygon which tightly circumscribes the hand region. Along with isolating the hand region, convex hull also allows us to detect tip of the fingers and center of the hand with few additional geometric computations which can used for further validation of accuracy in hand region extraction.



(A) Extracted Hand Region from Skin Masking



(B) Binarized Extracted Hand Region



(C) Hand Region after Applying Convex Hull

FIGURE 5.9. Hand Segmentation using Convex Hull

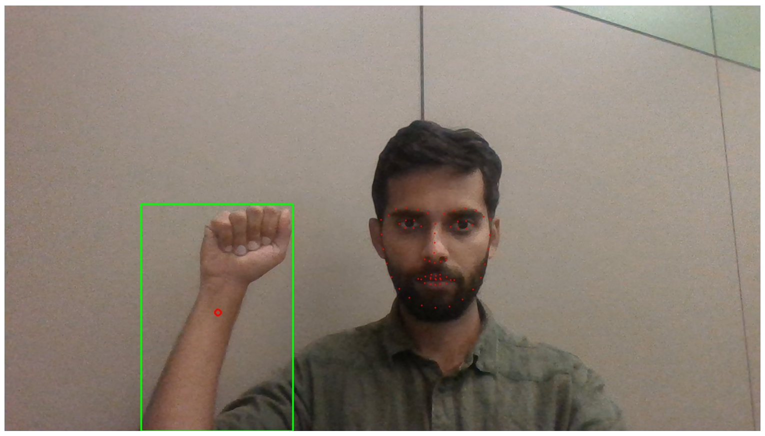


FIGURE 5.10. Detected Hand Region using Convex Hull

5.3. Hand Gesture Recognition

This module performs the core functionality of our system. Hand gesture recognition module performs image classification of segmented hand images and assigns a class label which is a letter from English alphabet. In our system we have an ensemble of image classifiers through transfer learning technique and developed a scoring function to combine confidences ensembled classifiers. Implementation of each of these techniques is discussed in this section.

5.3.1. Training

In our system we have trained four independent deep convolutional neural networks through transfer learning process using Tensorflow and ensembled together for predicting the hand gestures. The datasets used for training, the convolutional neural network architecture and training configuration are discussed in detail in this section.

Dataset

Machine learning is a data driven technique. The dataset used for training the machine learning model has a significant influence on the prediction confidence of the classifier. For this system, we have used two types of datasets for training; one is a standard finger-spelling American sign language dataset curated by Massey University.

This dataset consists of 2425 images captured from 5 individuals under various lighting conditions and variable hand postures and is also subjected to primary image processing in order reduce the presence of noise and extract important features from the image. The original dataset consists of 36 classes which are 26 letters from A-Z and digits from 0-9 but for our system we have taken a subset of 23 classes which are letters A-Y excluding J, Q and Z as these gestures need additional temporal features for recognition.

As the number of images in Massey dataset is very less to train a deep-convolutional neural network, we have generated an in-house finger-spelling ASL dataset by almost emulating the experimental set-up of Massey dataset. In our set-up instead of using a green or any other uniform background, we have used the hand segmentation technique discussed in the earlier sections for obtaining hand gesture images for training.

The experimental setup for capturing standard American sign language dataset by Massey university is as follows,

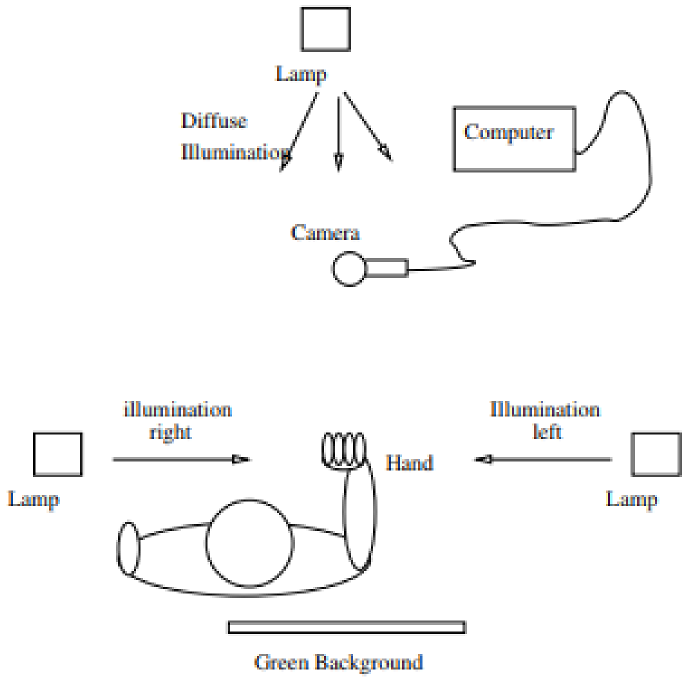


Figure 5.11. Massey University Experimental Set-Up for Capturing ASL Dataset (A Bird's Eye View) [4]

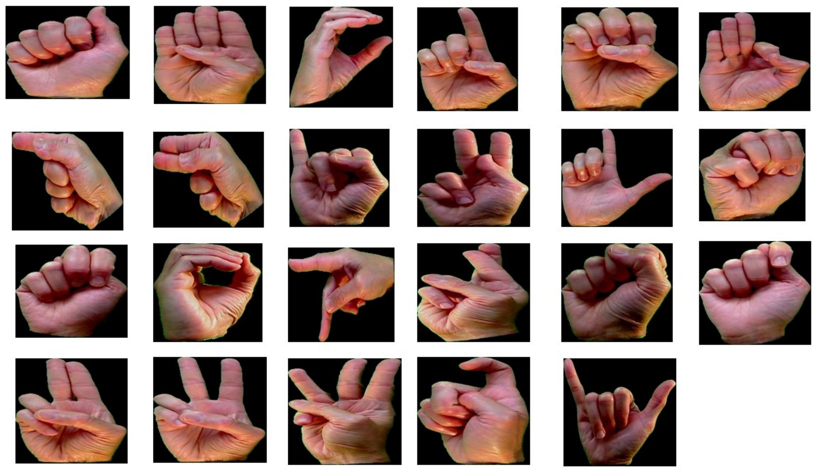


Figure 5.12. Sample Images from Massey University American Sign Language Dataset

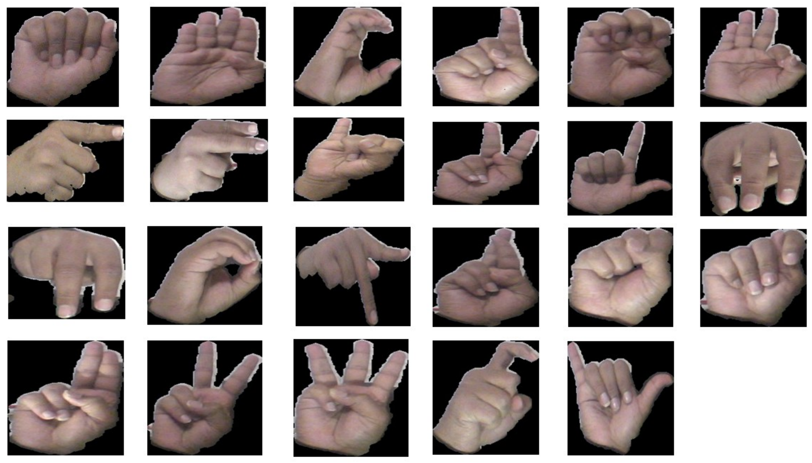


FIGURE 5.13. Sample Images from In-House Generated American Sign Language Dataset

In total, 25,254 images with 1,098 images for each class are used as dataset for this project

MobileNets

Since Alexnet, the winner of imageNet challenge: ILSVRC 2012 various deep convolutional neural networks like VGGNet, GoogleNet and ResNet gained a lot of popularity by achieving high accuracies in the coming years ImageNet Large Scale Visual Recognition Challenges(ILSVRC). These network architectures have become deeper and more complicated over the years to achieve high accuracies over large image datasets. In contrast, the real-world applications like our system like gesture recognition, robotics or self-driving cars require a neural network that runs on a computational device with limited computational resource. MobileNets is a deep-convolutional neural network developed by Google targeting the mobile and embedded vision application like ours. MobilNets are developed based on the concept of depth wise separable convolution. The conventional convolution operation has effect of filtering features based on the convolutional kernels and combining features in order to produce a new representation. These filtering and combination steps can be split into two steps using factorized convolutions which is called depth wise separable convolution. This modification of convolutional neural network has enabled MobileNets to achieve equivalent accuracy as any high performing neural network with better efficiency in terms of fastness and consumption of computational resources. A comparison of various Deep Convolutional neural network accuracies with respect to number of Multiplication and Addition Operations (MACs) they perform is visualized in the below figure,

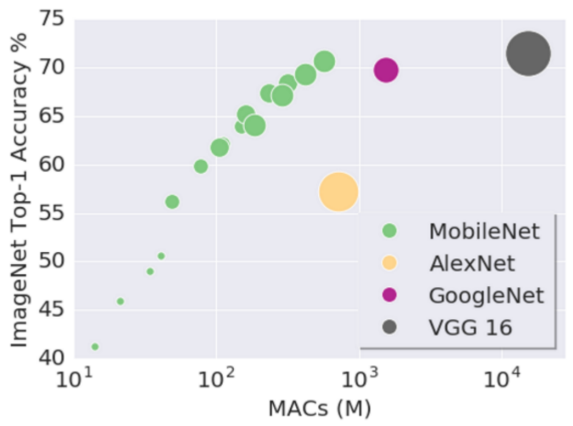
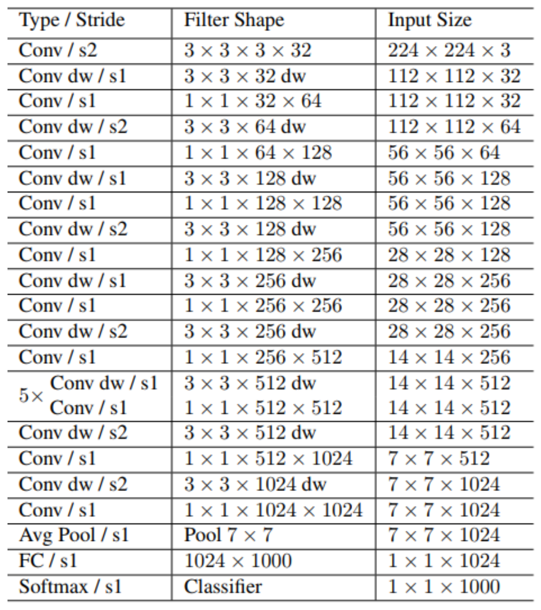


FIGURE 5.14. Comparison of Popular CNNs Accuracies with Respect to their MACs

The model architecture of MobileNets with filter and input dimensions is as follows,



Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

FIGURE 5.15. MobileNet Architecture [10]

Four MobileNets are ensembled together to perform gesture recognition.

Training Configuration

As discussed earlier, we have trained four independent MobileNets at different learning iterations using transfer learning technique. In transfer learning, we import a pre-trained MobileNet whose weights for each layer obtained through training the ILSVRC dataset and re-initiate the final classification layer to optimize for the new target dataset, in this case ASL dataset. As the CNNs seeks to optimize the cost function or loss function in our architecture we use SoftMax activation function in the final layer and use cross-entropy as loss or objective function.

Four Mobilenets are trained at 5000, 10000, 15000 and 20000 learning iterations with a learning rate of 0.001. A dataset of 25,254 images with 1,098 images of each class are used for training out of which 10% of data is used for cross-validation. The MobileNet architecture used for training accepts input images of size 224x224 and the batch size is set to 100 for both training and validation.

5.3.2. Scoring Function

The scoring function ensembles the independently trained MobileNets by aggregating their individual confidence scores. In ensemble learning, there are two popular techniques bagging and boosting and in our system, we use bagging technique to obtain final prediction confidence. This bagging based scoring makes the system highly stable as it reduces the prediction variance.

We take 10 consecutive frames of single gesture and aggregate their confidence scores for each classifier to obtain a final prediction confidence. In real-time, while performing image classification there is a chance of inputting transition frames which are captured while hand is transitioning between gestures. To avoid misclassification or low confidence scores, we capture 10 consecutive frames and aggregate their scores for each of the four CNNs. Later, a second level aggregation of scores obtained for all four learners is performed to obtain final prediction score.

5.3.3. Validation and Testing

As discussed in the previous section, 10% of the input dataset is used for validation i.e, 2525 images. For testing, we have captured gesture images from 10 subjects under variable environmental conditions like complex background, variable lighting conditions, inclining the gesture posture, posing gesture close and far-way to the camera etc. 100 test images for each letter are generated and given to the testing module. A sample of test images is as follows,

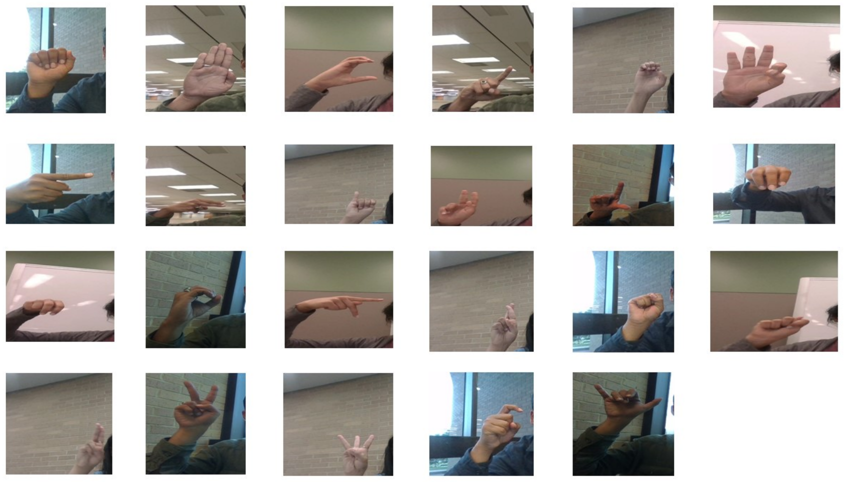


FIGURE 5.16. Sample Images from Test Set

5.4. Head Pose Estimation

To validate the correctness of the gesture recognition, we have incorporated a head pose estimation module as a feedback system where on a final decision of the gesture recognition system the user nods his head vertically or horizontally to convey his agreement or disagreement with the decision. For implementing this module, we have again made use of facial landmarks, estimated their 3D locations using 3D models from literature and computed translation and rotation of the head which will yield us the head pose. A detailed description of these techniques is discussed in this section.

5.4.1. Head Pose Estimation Flow

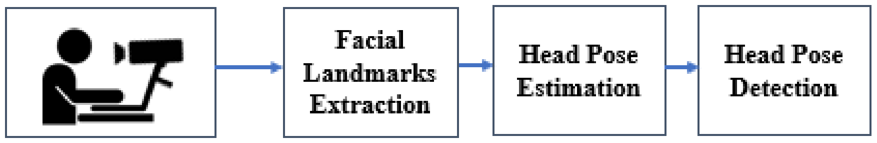


FIGURE 5.17. Head Pose Estimation Pipeline

5.4.2. Facial Landmarks for Head Pose Estimation

For head pose estimation, we require few 2D image coordinates of the face and their estimated 3D coordinates to compute translation and rotation of the head. To obtain the 2D image coordinates on the face we extract 6 crucial points from 68 facial landmarks obtained from dlib. The crucial coordinates are the landmarks that depict, tip of the nose, chin, left corner of the left eye, right corner of the right eye, left corner of the mouth, right corner of the mouth. We obtain the estimated 3D coordinates of these 6 feature points from world coordinates of OpenCV docs. The 2D feature points and their 3D locations are illustrated in the figure below [18],

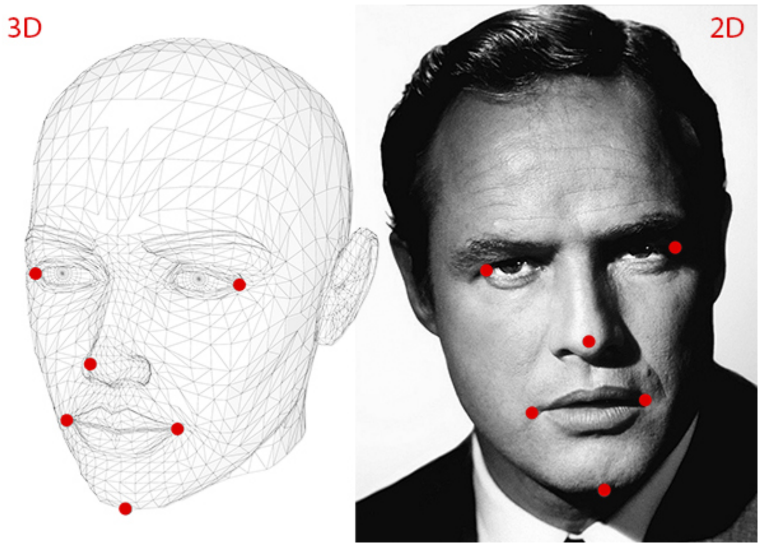


FIGURE 5.18. 2D and 3D Locations of the Feature Points for Head Pose Estimation [18]

5.4.3. Head Pose Detection

The head pose detection problem is solved by approaching it as a Perspective-n-Point problem. The Perspective-n-Points problem (PnP) deals with estimating the camera's pose in 3D when we have information about 3D coordinates and 2D image coordinates of the object it is capturing along with camera calibration [22].

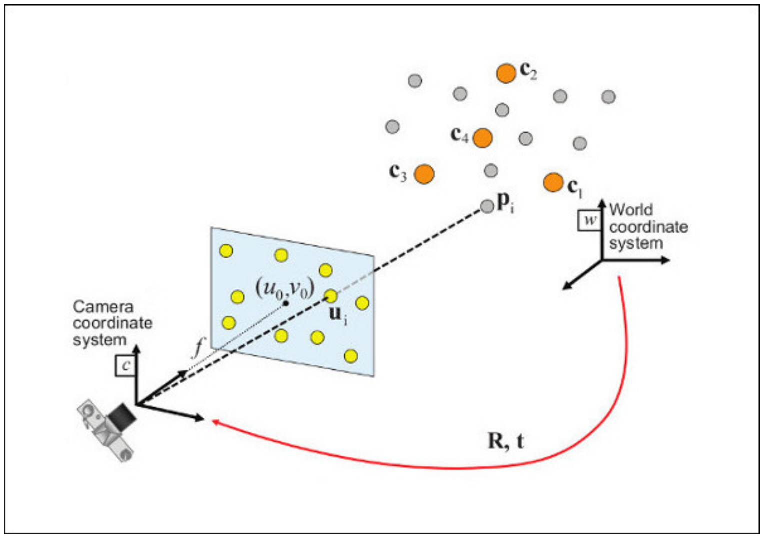


FIGURE 5.19. Illustration of Perspective n Point Problem [22]

Given a point $\boldsymbol{P_i}$ with coordinates (u,v,w) in real world coordinate system termed as 'world coordinate system' and $\boldsymbol{u_i}$ it's projection on a image plane with coordinates $(\boldsymbol{u_0}, \boldsymbol{v_0})$, head pose estimation involves two computational steps. First, computing (x,y,z) 3D coordinates that are representing the point in camera coordinate system and then computing $(\boldsymbol{u_0}, \boldsymbol{v_0})$, the projection of (u,v,w) on 2D image using camera calibration information. The first is computed using the following equation below,

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} = \boldsymbol{R} * \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{w} \end{bmatrix} + \boldsymbol{t}$$

Where R and t are Rotation and translation matrices which deal with rotation and translation of camera in it's coordinate system ,

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r_{00}} & \boldsymbol{r_{01}} & \boldsymbol{r_{02}} \\ \boldsymbol{r_{10}} & \boldsymbol{r_{11}} & \boldsymbol{r_{12}} \\ \boldsymbol{r_{20}} & \boldsymbol{r_{21}} & \boldsymbol{r_{22}} \end{bmatrix}, \boldsymbol{t} = \begin{bmatrix} \boldsymbol{t_x} \\ \boldsymbol{t_y} \\ \boldsymbol{t_z} \end{bmatrix}$$

The camera calibration matrix can be written as follows,

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{f_x} & \gamma & \boldsymbol{c_x} \\ 0 & \boldsymbol{f_y} & \boldsymbol{c_y} \\ 0 & 0 & 1 \end{bmatrix}$$

Here $\boldsymbol{f_x}$ and $\boldsymbol{f_y}$ are scaled focul lengths, $(\boldsymbol{c_x}, \boldsymbol{c_y})$ is principal point and $\boldsymbol{\gamma}$ is skew parameter assumed as 0.

For a given camera calibration information, we can compute $(\boldsymbol{u_0}, \boldsymbol{v_0})$ using the following equation,

$$\begin{bmatrix} \boldsymbol{u_0} \\ \boldsymbol{v_0} \\ 1 \end{bmatrix} = \boldsymbol{s} * \begin{bmatrix} \boldsymbol{f_x} & 0 & \boldsymbol{c_x} \\ 0 & \boldsymbol{f_y} & \boldsymbol{c_y} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$$

Where $\boldsymbol{s}$ is an unknown scaling factor. For simplicity consider focal lenghts to be 1 and principal point is (0,0). Therefore,

$$\begin{bmatrix} \boldsymbol{u_0} \\ \boldsymbol{v_0} \\ 1 \end{bmatrix} = \boldsymbol{s} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u_0} \\ \boldsymbol{v_0} \\ 1 \end{bmatrix} = \boldsymbol{s} * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$$

From the above equation we can derive a relation between world coordinates and their projection coordinates as,

$$\begin{bmatrix} \boldsymbol{u_0} \\ \boldsymbol{v_0} \\ \boldsymbol{1} \end{bmatrix} = \boldsymbol{s} * \begin{bmatrix} \boldsymbol{r_{00}} & \boldsymbol{r_{01}} & \boldsymbol{r_{02}} & \boldsymbol{t_x} \\ \boldsymbol{r_{10}} & \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{t_y} \\ \boldsymbol{r_{20}} & \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{t_z} \end{bmatrix} * \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{w} \\ \boldsymbol{1} \end{bmatrix}$$

In our case, $(\boldsymbol{u_0}, \boldsymbol{v_0})$ is a facial landmark point in image plane and (u,v,w) is a point in 3D coordinate system obtained by real-world 3D coordinates estimation model. For example if we know a facial landmark $(\boldsymbol{u_0}, \boldsymbol{v_0})$ depicting tip of the nose in the image, and estimated 3D coordinates of nose tip from estimation model, we can obtain the rotation and translation information by solving the above equation which will yield us the head pose. As the equation is nonlinear, the solution is not straightforward.

To solve for Rotation and translation matrices, we use iterative solvepnp technique implemented in OpenCV. This technique performs Direct Linear Transform (DLT) which is a technique used to solve equations like ours, which are almost linear equations but behave non-linear because of a scalar multiplication of an unknown factor. This is followed by Levenberg-Marquardt optimization technique which will minimize the inaccuracies introduced by DLT called reprojection errors. Therefore, the 2D projection obtained through rotation and translation vectors is used for head pose estimation.

To detect the estimated pose as 'Yes' or 'No we use Euler angles which are used to describe the orientation of rigid objects. The Euler angles for a human head are as follows,

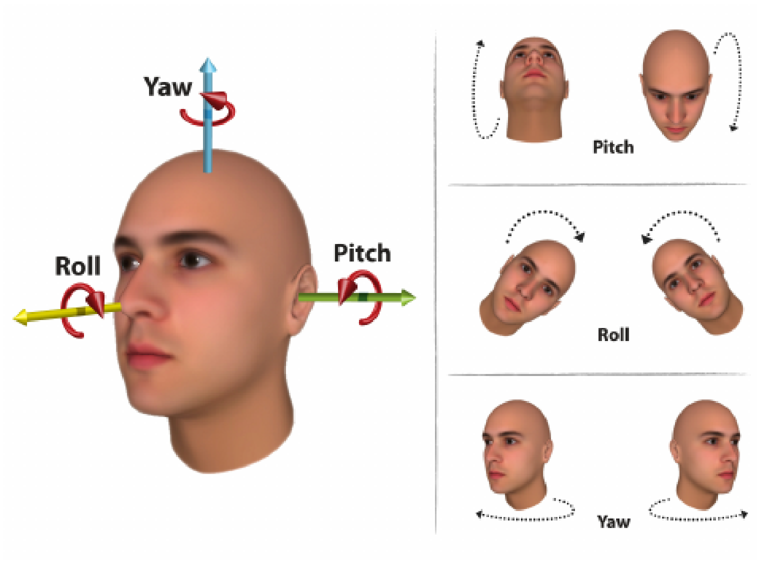


FIGURE 5.20. Euler Angles

Among the five quadrants, top left and right quadrants act as transitional buffers between 'Yes' and 'No' pose and the bottom left and right quadrants are used to detect Yaw i.e. 'No' and quadrant above the head is used to detect Pitch i.e. 'Yes'. The rotation and translation matrices obtained through solvePnP are used draw a line segment with certain magnitude from tip of the nose which changes it's orientation based on head pose. Using this illustration 'Yes' or 'No' can be decided based on the quadrant the other end of the line segment is lying in. The above detection process is clearly visualized in the below images,

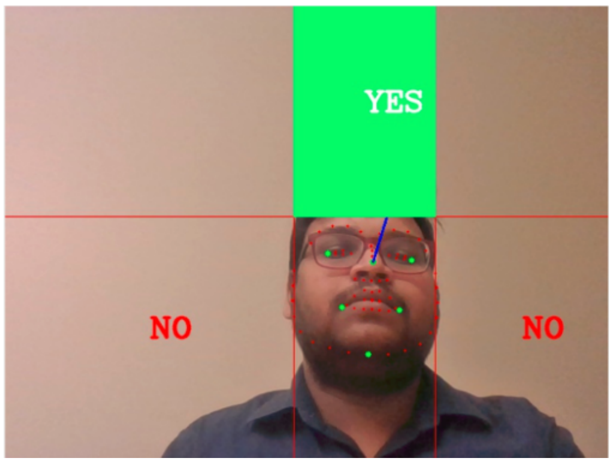


FIGURE 5.21. 'Yes' Detection using Head Pose Estimation

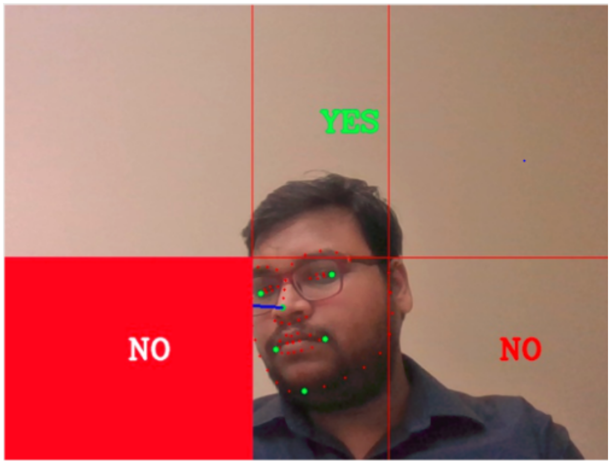


Figure 5.22. 'No' Detection using Head Pose Estimation

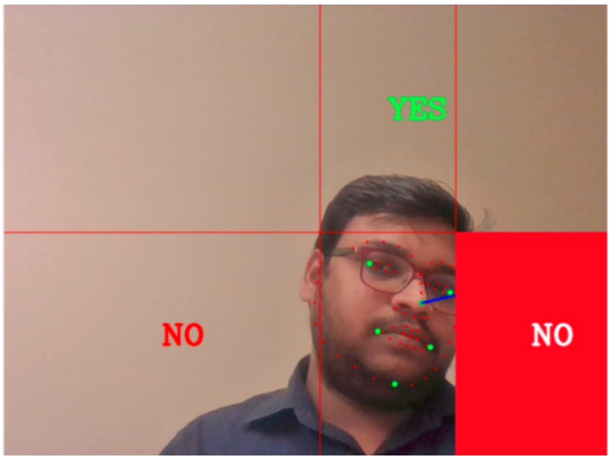


Figure 5.23. 'No' Detection using Head Pose Estimation

CHAPTER 6

RESULTS

To evaluate the performance of our system, we use training and validation accuracies, optimization of cross entropy and confusion matrix as metrics. The evaluation results for each of these parameters are presented in this chapter.

6.1. Training and Validation Accuracies

Training and validation accuracies are computed for each iteration during the training process. The training accuracy is the percentage of images in training set that are assigned with correct class label, this accuracy is generally observed to increase as the model converges over a number of iterations. To compute validation accuracy, 10% of training set is used for validation and validation accuracy provides the percentage of images in validation set that are assigned with correct class label. The training and validation accuracy trends four classifiers over the configured learning iterations are visualized in the plots below,

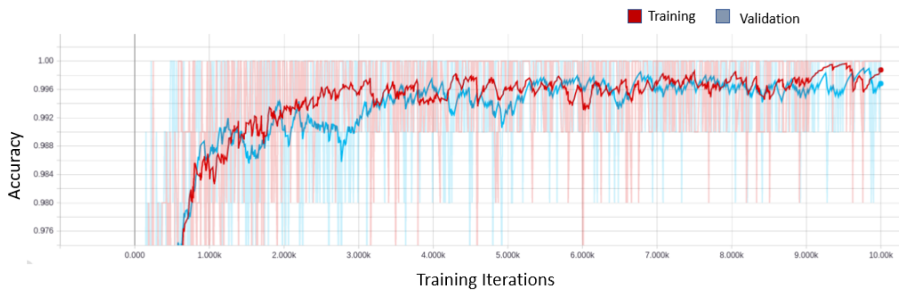


FIGURE 6.2. Training/Validation Accuracy (MobileNet @ 10000 Iterations)

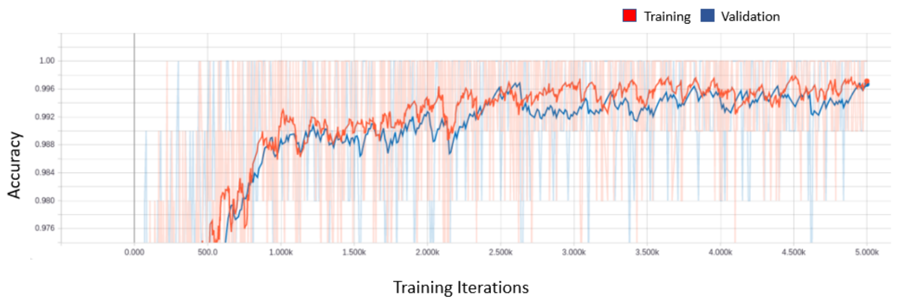


FIGURE 6.1. Training/Validation Accuracy (MobileNet @ 5000 Iterations)

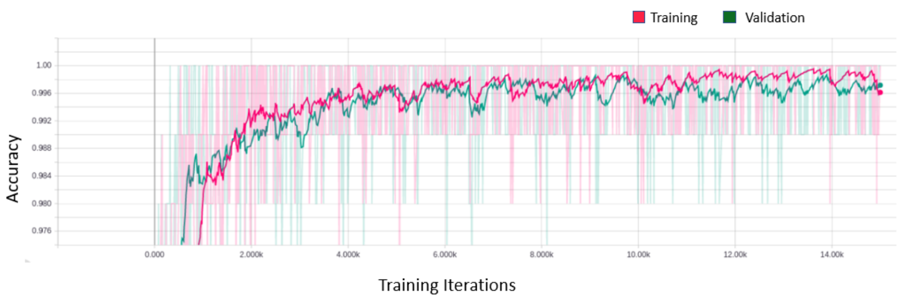


FIGURE 6.3. Training/Validation Accuracy (MobileNet @ 15000 Iterations)

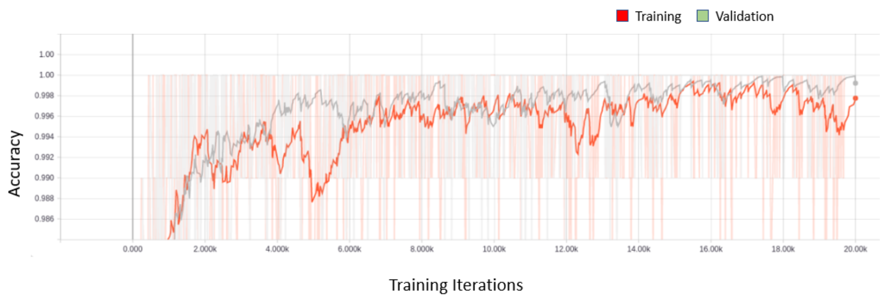


FIGURE 6.4. Training/Validation Accuracy (MobileNet @ 20000 Iterations)

6.2. Cross-Entropy Loss

Cross entropy is the loss function or objective function of our CNN. The cross entropy between the actual class distribution $\boldsymbol{p}$ and estimated class distribution $\boldsymbol{q}$ is defined as:

$$\boldsymbol{H(p,q) = -\sum_x p(x) \log q(x)}$$

As we are using SoftMax function as activation function the final layer, The cross entropy loss can be written as follows,

$$\boldsymbol{L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)} \qquad \text{or equivalently} \qquad \boldsymbol{L_i = -f_{y_i} + \log \sum_j e^{f_j}}$$

Visualization of cross entropy decay over defined learning iterations for four of our classifiers is as follows,

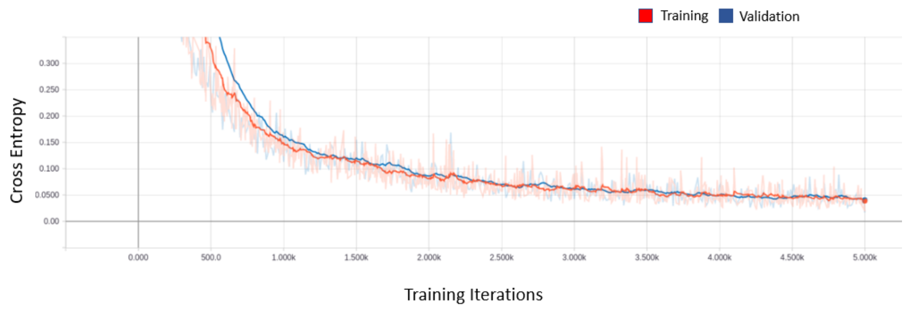


FIGURE 6.5. Cross Entropy (MobileNet @ 5000 Iterations)

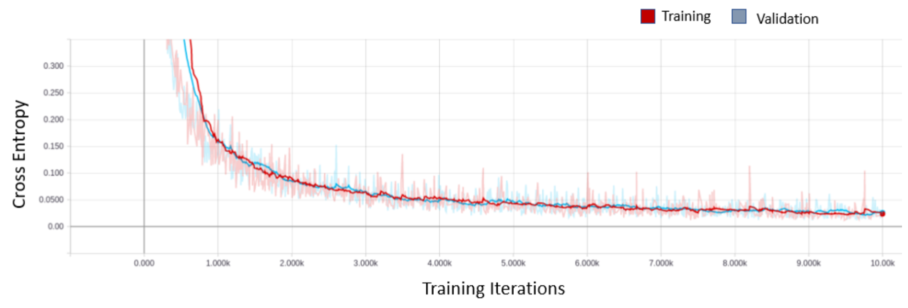


FIGURE 6.6. Cross Entropy (MobileNet @ 10000 Iterations)

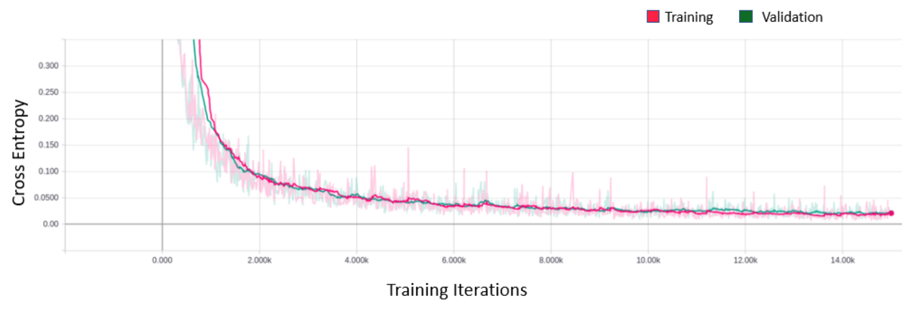


FIGURE 6.7. Cross Entropy (MobileNet @ 15000 Iterations)

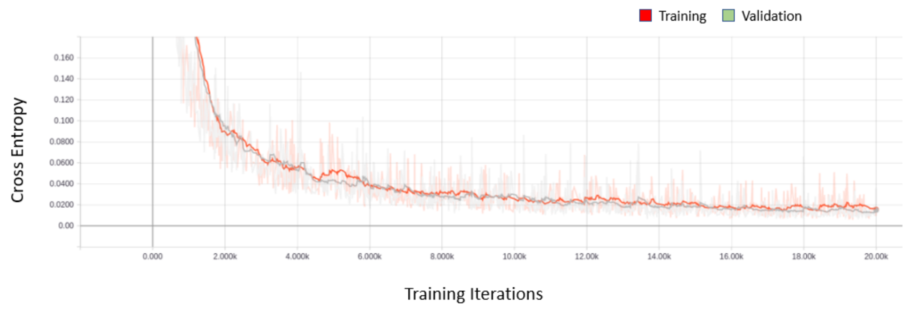
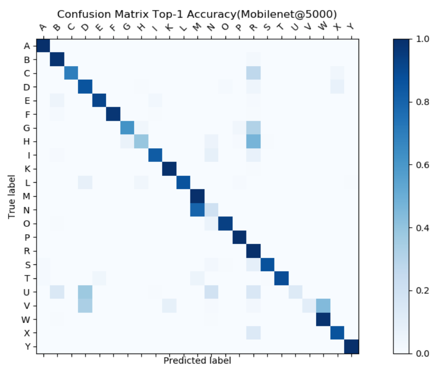


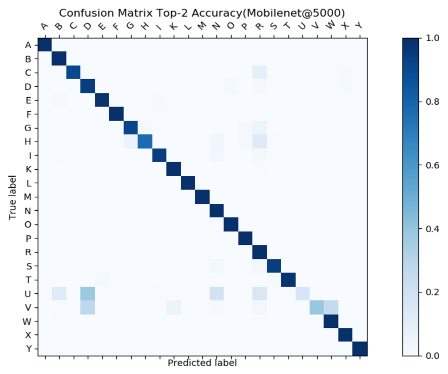
FIGURE 6.8. Cross Entropy (MobileNet @ 20000 Iterations)

6.3. Confusion Matrix

The confusion matrix provides a visualization of classification accuracy over the test-set. To evaluate the system performance, we construct two confusion matrices of each classifier based in Top-1 accuracy and Top-2 accuracy. Top-1 accuracy is a regular accuracy measurement where the predicted class labels with highest probability (top 1 probability) are compared with ground truths for correctness. Where as in Top-2 accuracy we check for expected label in top 2 predicted class labels. Top-2 accuracy is used in our system as we use head pose estimation to validate the correctness of first prediction and obtain the second-best prediction if the first label is wrong. This measure will help us to understand the significance of head pose estimation system. The confusion matrices for all four classifiers and ensemble classifier is as follows,



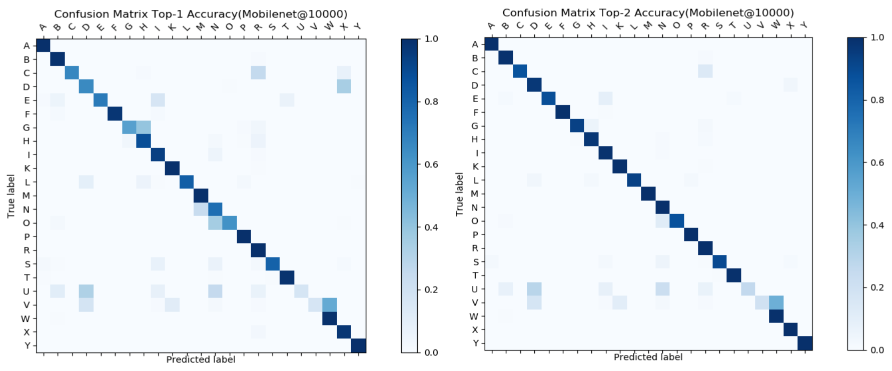
(A) Confusion Matrix Top-1 Accuracy



(B) Confusion Matrix Top-2 Accuracy

Figure 6.9. Confusion Matrices Comparison for MobileNet@5000

An average Top-1 and Top-2 accuracies of MobileNet@5000 classifier are 78% and 90% respectively. By observation, it is evident that head pose estimation will boost the system performance and for this classifier, there is a lot of confusion between M & N, D,U & V.

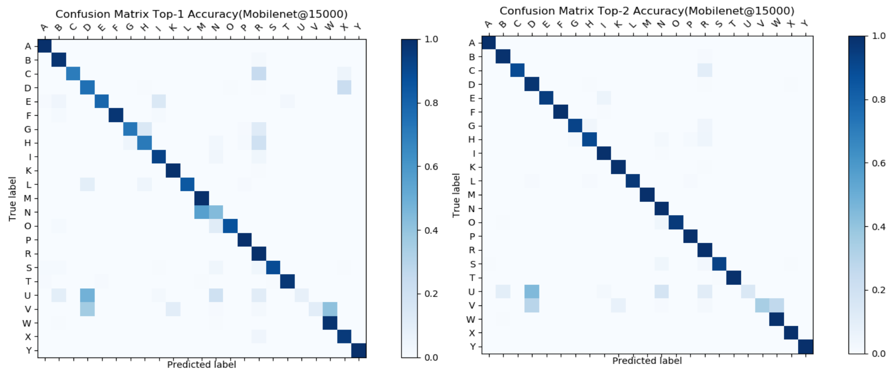


(A) Confusion Matrix Top-1 Accuracy

(B) Confusion Matrix Top-2 Accuracy

FIGURE 6.10. Confusion Matrices Comparison for MobileNet@10000

For MobileNet@10000, average Top-1 and Top-2 accuracies are observed as 81% and 90% with a similar confusion between D, U & V.

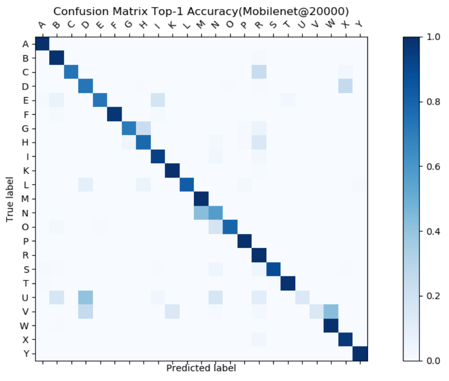


(A) Confusion Matrix Top-1 Accuracy

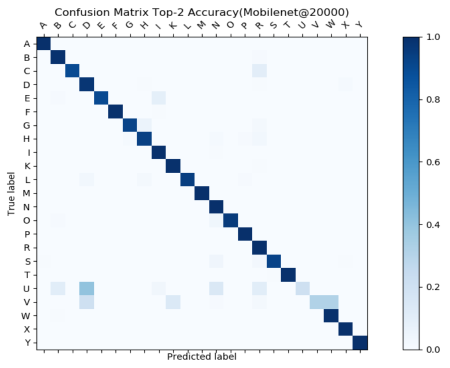
(B) Confusion Matrix Top-2 Accuracy

FIGURE 6.11. Confusion Matrices Comparison for MobileNet@15000

For MobileNet@15000, average Top-1 and Top-2 accuracies are 81% and 91% respectively.



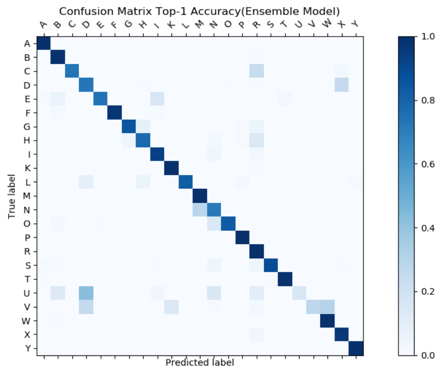
(A) Confusion Matrix Top-1 Accuracy



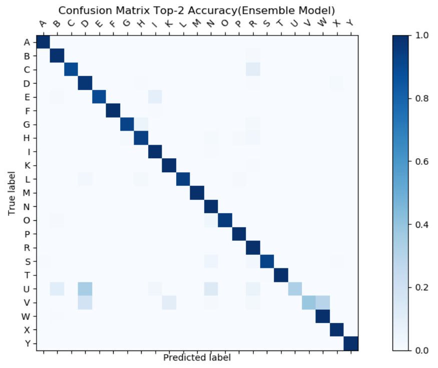
(B) Confusion Matrix Top-2 Accuracy

FIGURE 6.12. Confusion Matrices Comparison for MobileNet@20000

An average Top-1 and Top-2 accuracies of MobileNet@20000 classifier are 82% and 91% respectively



(A) Confusion Matrix Top-1 Accuracy

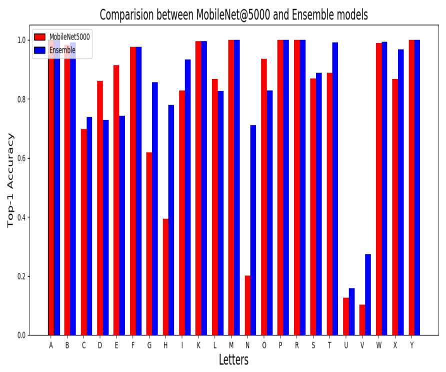


(B) Confusion Matrix Top-2 Accuracy

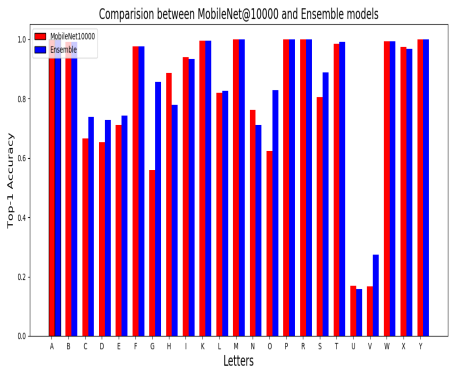
FIGURE 6.13. Confusion Matrices Comparison for Ensemble Model

For ensembled model the Top-1 and Top-2 accuracies are 85% and 92% respectively. From observation it is evident that misclassification of 'U' and 'D' is significantly reduced and confusion between 'U' and 'V' also considerably mitigated.

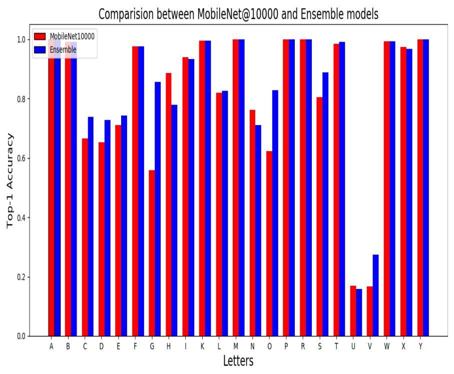
Comparison between individual models and ensembled model



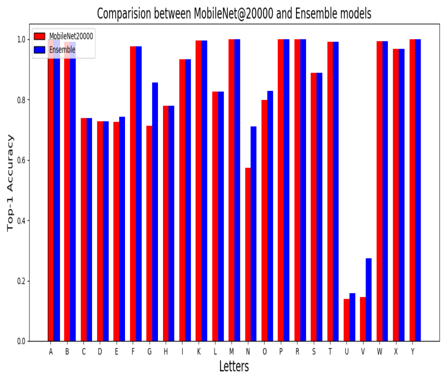
(A) Comparison between MobileNet@5000 and ensembled model



(B) Comparison between MobileNet@10000 and ensembled model



(C) Comparison between MobileNet@15000 and ensembled model



(D) Comparison between MobileNet@20000 and ensembled model

FIGURE 6.14. Comparison between Individual Models and Ensembled Model

CHAPTER 7

DISCUSSION

From observation of above performance metrics, it is clearly noticed that head pose estimation to validate the correctness of highest predicted letter will definitely boost the performance and it is observed that each model has shown high accuracy in recognizing certain gestures and the ensembled model was useful in attaining certain level of uniformity in terms of accuracy among all the gestures. Also, ensemble technique had the minimum difference between Top-1 and Top-2 accuracies which signifies that the classifier is showing a consistent performance.

Ensemble of Convolutional neural networks is one of the unique research innovations of our system which boosted the performance of the system and transfer learning has played a significant role in training an accurate CNN model with comparatively small set of data points.

Head pose estimation as a feedback unit is a very helpful feature in a real-time system as the image classification based gesture recognition doesn't always produce 100% classification results. The technique used to identify 'Yes' and 'No' head pose also results to be a robust technique with no scope for misidentification of 'Yes' as 'No' and vice versa.

Blink detection is another engineering innovation in our system as it is essential in the real-time because continuous up-time of recognition unit is computationally expensive and adds a lot of latency to the system.

Limitations

Along with the above features there are limitations of this system which shall be addressed in the coming versions. The current recognition system is trained to work only for right hand gestures and hand detection technique using skin color based segmentation has mitigated majority of false positives but may not work efficiently incase of a background very similar to skin color. Also the system is not trained to recognize gestures for letters J, Q, Z and digits.

CHAPTER 8

FUTURE WORK

This hand gesture recognition system we developed performs a fundamental task of mapping gestures with English letters. There is a high scope in extending this project and making it more assistive to specially-abled community. few of such advancements that has immediate scope to incorporate in our system are discussed in this section.

Including letters 'J','Q' and 'Z' for gesture recognition

The current system is not trained to recognize gestures for letters 'J', 'Q' and 'Z' as the letters 'J' and 'Z' involves spatio-temporal details for gesture classification, where as our system deals with static image classification which involves only spatial details. These gestures can be recognized by implementing hand motion tracking and memory based deep learning algorithms like LSTMs. Training gestures for 'Q' are exempted because the specific user for whom this system is designed cannot pose gesture 'Q' because of his physical challenges.

Deep learning based hand segmentation

Adapting advanced deep learning algorithms for hand detection may nullify false positives in hand detection and segmentation which will boost the performance of our system.

Integrating auto word/sentence completion techniques

With the recent advancements in Natural Language Processing (NLP), a lot of personalized predictive models for auto-completion of words and sentences are being developed. Integrating one such technique to our recognition system shall make it redundant for the user to enact each letter to compose a sentence.

Implementing Reinforcement Learning based gesture recognition model

Reinforcement learning is one of the latest deep learning techniques which performs its current actions by drawing inferences from previous failures. It learns based on a technique called Markov Decision Process. Incorporating such model into our system can use our feedback system to learn from its mistakes.

CHAPTER 9

CONCLUSION

We have aimed at developing a real-time computer vision based finger spelling American sign language recognition system without any special equipment like Kinect 3D camera or data glove. Also we wanted to implement a touch-less feedback unit using any of the natural gestures used in our daily life. This system performs gesture recognition using MobileNet, a mobile and embedded system compatible CNN architecture. Along with a standard dataset from Massey university we have created our own ASL dataset for training four independent mobileNet models. Ensembling of these four models through bagging approach has yielded an overall accuracy of 92% in classification and helped in obtaining near uniform accuracy across all the letters. Along with the core system, head pose estimation based feedback unit which is inspired from natural way of expressing acceptance or denial through head nodding is successfully implemented and observed to boost the overall classification accuracy.

The entire system is developed using python, OpenCV and tensorflow and deployed as a web application using Flask to make it operating system independent. Along with current features, there is a scope for making lot of improvements and enhancements. Deep learning based hand detection, re-enforcement learning based gesture recognition and LSTM based word/sentence auto completion enables this system to be more flexible for specially abled community.

REFERENCES

[1] Y. F. Admasu and K. Raimond, *Ethiopian sign language recognition using artificial neural network*, 2010 10th International Conference on Intelligent Systems Design and Applications, Nov 2010, pp. 995–1000.

[2] M. A. Amin and H. Yan, *Sign language finger alphabet recognition from gabor-pca representation of hand gestures*, 2007 International Conference on Machine Learning and Cybernetics, vol. 4, Aug 2007, pp. 2218–2223.

[3] Ashutosh, A. Singh, S. Banerjee, and S. Chaudhury, *A gesture based interface for remote robot control*, Proceedings of IEEE TENCON '98. IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control (Cat. No.98CH36229), vol. 1, Dec 1998, pp. 158–161 vol.1.

[4] Andre L. C. Barczak, Napoleon H. Reyes, M. Abastillas, A. Piccio, and Teo Susnjak, *A new 2 d static hand gesture colour image dataset for asl gestures*, 2012.

[5] Vivek Bheda and Dianna Radpour, *Using deep convolutional networks for gesture recognition in american sign language*, CoRR abs/1710.06836 (2017).

[6] C. Chuan, E. Regina, and C. Guardino, *American sign language recognition using leap motion sensor*, 2014 13th International Conference on Machine Learning and Applications, Dec 2014, pp. 541–544.

[7] N. Dardas, Q. Chen, N. D. Georganas, and E. M. Petriu, *Hand gesture recognition using bag-of-features and multi-class support vector machine*, 2010 IEEE International Symposium on Haptic Audio Visual Environments and Games, Oct 2010, pp. 1–5.

[8] M. Van den Bergh and L. Van Gool, *Combining rgb and tof cameras for real-time 3d hand gesture interaction*, 2011 IEEE Workshop on Applications of Computer Vision (WACV), Jan 2011, pp. 66–72.

[9] Brandon Garcia and Sigberto Viesca, *Real-time american sign language recognition with convolutional neural networks*, Convolutional Neural Networks for Visual Recognition (2016).

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, CoRR abs/1704.04861 (2017).

[11] ______, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, CoRR abs/1704.04861 (2017).

[12] D. Huang, W. Hu, and S. Chang, *Vision-based hand gesture recognition using pca+gabor filters and svm*, 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Sept 2009, pp. 1–4.

[13] Byeongkeun Kang, Subarna Tripathi, and Truong Q. Nguyen, *Real-time sign language fingerspelling recognition using convolutional neural networks from depth map*, CoRR abs/1509.03001 (2015).

[14] T. Kapuscinski and M. Wysocki, *Hand gesture recognition for man-machine interaction*, Proceedings of the Second International Workshop on Robot Motion and Control. RoMoCo'01 (IEEE Cat. No.01EX535), Oct 2001, pp. 91–96.

[15] Davis E. King, *Dlib-ml: A machine learning toolkit*, Journal of Machine Learning Research 10 (2009), 1755–1758.

[16] M. V. Lamari, M. S. Bhuiyan, and A. Iwata, *Hand alphabet recognition using morphological pca and neural networks*, IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), vol. 4, July 1999, pp. 2839–2844 vol.4.

[17] S. Liwicki and M. Everingham, *Automatic recognition of fingerspelled words in british sign language*, 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, June 2009, pp. 50–57.

[18] Satya Mallick, *Head pose estimation using opencv and dlib*, [Available at https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib; accessed 10-March-2017].

[19] P. Mekala, Y. Gao, J. Fan, and A. Davari, *Real-time sign language recognition based on neural network architecture*, 2011 IEEE 43rd Southeastern Symposium on System Theory, March 2011, pp. 195–199.

[20] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cirean, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, *Max-pooling convolutional neural networks for vision-based hand gesture recognition*, 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Nov 2011, pp. 342–347.

[21] Nat and Friends, *How computer vision is finally taking off, after 50 years*, [Available at "`https://www.youtube.com/watch?v=eQLcDmfmGB0`"; accessed 19-June-2018].

[22] Opencv.org, *Real time pose estimation of a textured object*, [Online; accessed 15-March-2017].

[23] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen, *Sign language recognition using convolutional neural networks*, Computer Vision - ECCV 2014 Workshops (Cham) (Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, eds.), Springer International Publishing, 2015, pp. 572–578.

[24] K. O. Rodrguez and G. C. Chvez, *Finger spelling recognition from rgb-d information using kernel descriptor*, 2013 XXVI Conference on Graphics, Patterns and Images, Aug 2013, pp. 1–7.

[25] Mitchell Ross, Young Travas, Bachleda Bellamie, and Karchmer Michael, *How many people use asl in the united states?: Why estimates need updating*, [Available at "`https://www.youtube.com/watch?v=eQLcDmfmGB0`"; accessed 19-June-2018].

[26] C. Savur and F. Sahin, *Real-time american sign language recognition system using surface emg signal*, 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Dec 2015, pp. 497–502.

[27] Tereza Soukupová and Jan Cech, *Real-time eye blink detection using facial landmarks*, 2016.

[28] T. Starner and A. Pentland, *Real-time american sign language recognition from video using hidden markov models*, Proceedings of International Symposium on Computer Vision - ISCV, Nov 1995, pp. 265–270.

[29] J. Triesch and C. von der Malsburg, *Robust classification of hand postures against com-*

plex backgrounds, Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, Quarterly 1996, pp. 170–175.

[30] Wikipedia, *Deep learning*, [Available at `https://en.wikipedia.org/w/index.php?title=Deep_learning`; accessed 19-June-2018].