# 实时美国手语识别
# ____使用深层卷积神经网络和计算机视觉


# 重庆大学硕士学位论文

（专业学位）

学生姓名：Nouman Ahmad

指导教师：洪　沙　教授

专业学位类别：软件工程

研究方向：计算机视觉

答辩委员会主席：张　毅

授位时间：2020 年 6 月

# REAL-TIME AMERICAN SIGN LANGUAGE RECOGNITION USING DEEP CONVOLUTIONAL NEURAL NETWORK AND COMPUTER VISION


A Thesis Submitted to Chongqing University

in Partial Fulfillment of the Requirement for

Professional Degree

**By**

**Nouman Ahmad**

**Supervised By Dr. Hong Sha**

**June 2020**

# Abstract

Speech impairment is a disorder that affects the ability of a person to interact through speech and listening. Individuals affected by this use of other resources such as sign language. Recognition of sign language is still a problem for real practice, although it was discussed for many years. Complex context and lighting conditions impact hand tracking and make it very difficult to identify SL. As a result of recent advances in computer vision, promising progress has been made with deep learning and computer vision techniques in the fields of motion and gesture recognition. The goal of this thesis is to develop a vision-based application that offers text translation into sign language and therefore facilitates communication between signatories and non-signers.

The work of this paper is as follows:

① The basic theory and related technologies of sign language recognition are studied.

②A prototype model including a hybrid classifier is designed. The proposed prototype uses video sequences and extracts the required features, which are then verified by CNN.

③ According to the prototype model, four stages of preprocessing, segmentation, extraction and classification are completed.

④ The data set used is the American Sign Language data set, which is made by the author using the open source data set and OpenCV.

⑤ In addition, the database used for classification through CNN is keras, with a total of 36 gestures trained.

**Keywords:** Computer vision; deep learning; CNN; American Sign Language

# 摘 要

言语障碍是一种影响着人们通过言语和听力进行交互能力的障碍，尽管已经讨论了很多年，但手语的识别在实际应用中仍然存在着许多急需解决的问题。由于复杂的环境和光照条件会影响手部的跟踪识别效果，因此很难精准识别。随着时代的进步以及计算机视觉技术的最新进展，在运动和手势识别领域中，深度学习和计算机视觉技术已取得了可喜的进步，为解决上述问题提供了较好的解决方案。

论文主要完成的工作如下：

①深入研究了手语识别的基础理论和相关技术；

②设计了包括一个混合分类器的原型模型，提出的原型采用视频序列并提取所需的特征，然后由 CNN 进行验证。

③根据设计的原型模型完成了预处理、分割、提取和分类等四个阶段的工作；

④所使用的数据集是美国手语数据集，该数据集由作者采用开源数据集自己使用 OpenCV 制作而成。

⑤此外，通过 CNN 用于分类的库是 Keras，总共训练了 36 种手势。

**关键词：**计算机视觉；深度学习；CNN；美国手语

# Table of Contents

# Chart Catalog

# CHAPTER 1 INTRODUCTION

## 1.1 Introduction

Communicating is one of the unique practices of human interaction that has not only provided a crucial means of expression but also plays a key part in maintaining social coherence. It is a physical anomaly that impairs speech prohibit deaf and hard of hearing correspondences from talking for communication. To overcome this, a visual communication system called sign language was developed that uses gestures (mostly manual) to translate words into the oral language. American Sign Language (ASL) is such a common sign language, using hand gestures for the expression of letters in the English alphabet. The American Sign Language (ASL) is a complete and natural language with different syntax and language features from spoken languages. ASL is conveyed by hands and their different gestures. This is a primary language of many North Americans who are impaired and hard to understand, and it is also used by many listeners.

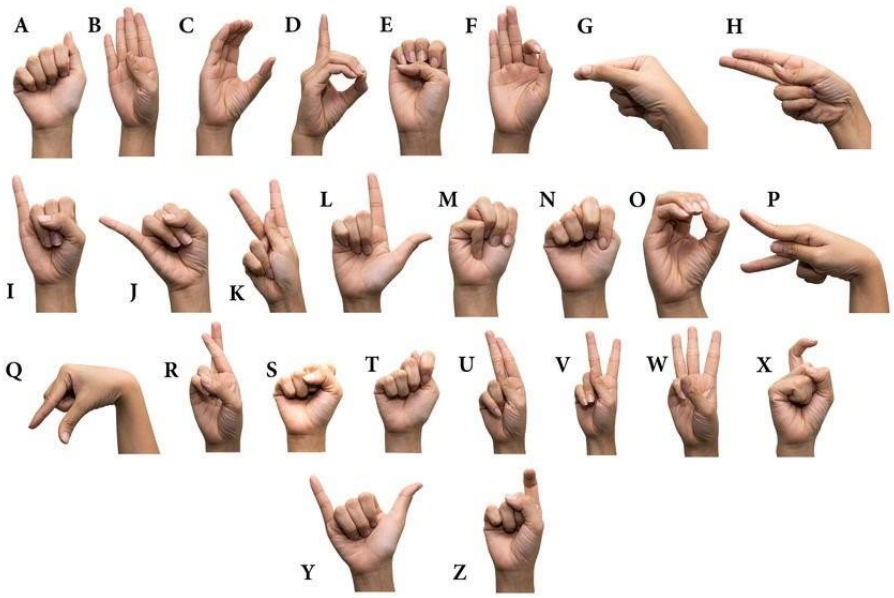



Fig 1. 1 Finger Spelling American Sign Language

## 1.2 Motivation

Despite this alternate form of communication, it is very challenging to have an effective conversation with the general public, because interpreting the American Sign Language needs a lot of training and practice [25]. As a result, this division creates a great

deal of social isolation between people with hearing and speech disabilities living together in society. A comprehensive support technology is needed to bridge this gap, which can interpret ASL hand gestures and translate them into natural language.

We developed an American Sign Language hand gesture recognition system using deep learning with a strong purpose to make personalizing help technology available to specially qualified groups to solve their challenges. In this project, we viewed the classification of hand gestures as an image recognition problem and have trained CNNs, using transfer learning to perform the recognition task We also incorporated American numbers estimation as an extra system with gesture recognition to enhance the correctness of a recognized behavior in a real-time.

## 1.3 Hand Gesture Recognition

The understanding of hand gestures is the ability of computers to recognize the gestures of a human by means of complex mathematical algorithms. Computer-based hand gesture recognition is a popular technique for human computing, as it is very similar to natural human interaction.

Computer hand gesture recognition is a well-explored research technology, featuring various implementation approaches, such as skin detection, hand segmentation, and hand motion tracking, using traditional image processing techniques. Nonetheless, this work is much less translated into a real application because it is very difficult to achieve stability under highly different conditions of lighting and widespread skin color based on ethnicity. Computer vision domain–especially image recognition–is advancing quite quickly after 50 years of its birth with recent advances in artificial intelligence and deep learning [21]. One such deep learning algorithm is implemented in our program to identify movements of the hand using Convolutional Neural Networks.

## 1.4 Deep Learning and Transfer Learning

Deep learning is a fast-growing, special class of machine learning, built into several layers by cascading nonlinear processing units that carry out transformation, extraction, and decision-making. Deep learning algorithms address challenging signal processing problems, such as speech recognition, image detection, processing of natural languages and automatic translation [30].

As this model has a wide variety of processing units, throughout training it requires a high level of computer resources. The use of GPU clusters to learn the training dataset

is adapted to parallel processing techniques. Transfer learning is an alternative to run these enormous models with minimal computational power.

Transfer Learning is a process by which hyperparameters of pre-trained networks can be distributed, that are trained on numerous GPUs for days. The boring task of building a deep network is minimized by this approach. So machine learning decision-making is a data-based activity. In case of limited training and a significantly wide range of class labels, transfer learning is flexible. A MobileNet CNN model for transfer training with 1.2 million pictures in 1000 classes was discussed in this project [11].

## 1.5 Convolutional Neural Networks

The Convolutional Neural Network is a revolutionary learning method built out of the visual cortex inspiration, which is the basic blocks of human vision. From the study, it is observed that on the base of this observation CNNs are produced and observed that they are above the influential classification techniques, in a large-scale convolution by the human brain that process visual signals.

The convolution and pooling are two main operations carried out at CNN and these blocks are designed in an extremely complicated way to mimic the human brain.

The neural network is developed into layers, with increased network complexity due to the number of layers and the consistency of the structure being tested. Three operational frames are connected to the CNN architecture as a complex architecture. Convolutional Neural Network Functional Blocks are

- Convolutional Layer
- Max Pooling layer
- Fully-Connected layer

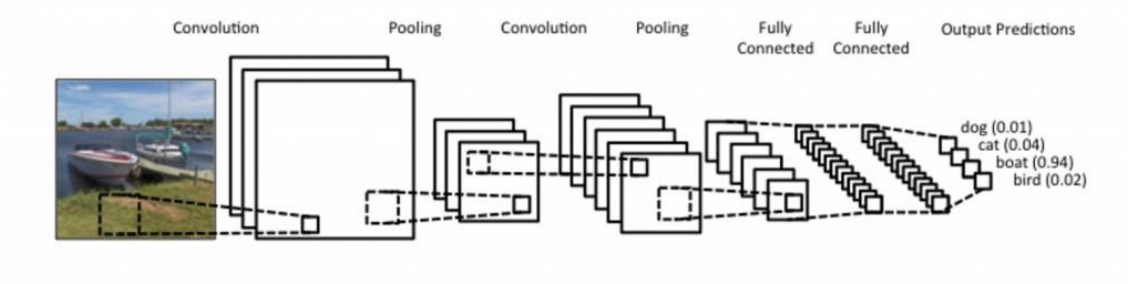



Fig 1. 2 Convolutional Neural Network

## 1.6. Thesis Outline

The organization of this thesis report is as follows,

Chapter 1 Introduction: In this chapter, we have introduced the problem statement we are addressing, our enthusiasm to choose this problem and a screening of our solution.

Chapter 2 Machine Learning using Convolutional Neural Networks (CNN): This chapter aims at introducing the basic concepts of CNN and the most essential terminologies that are important to figure out while performing experiments.

Chapter 3 significant challenges: This chapter delivers a brief summary of our study on related work in literature and the main challenges in gesture recognition.

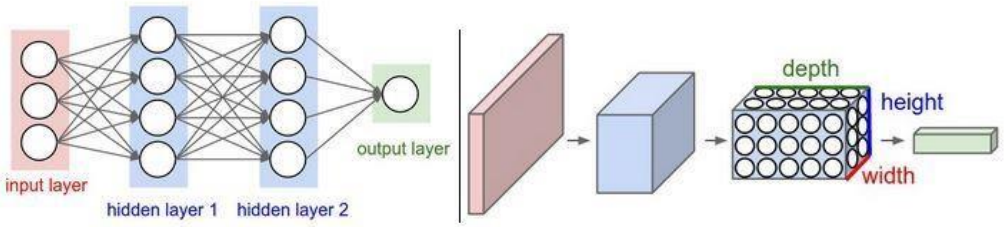
Chapter 4 system design: This chapter gives an overview of our project pipeline and a brief explanation of each module and its significance in our system.

Chapter 5 Sign Language Dataset: In this chapter, we discuss the datasets that we have generated and implementation of each module in our system. This chapter deals with algorithms, techniques, technologies and computational platforms incorporated and integrated into this project.

Chapter 6 Conclusion and future works: In this chapter, we discuss the conclusion with the graphs results and discuss the work that we are expected to do in our future.

# CHAPTER 2 MACHINE LEARNING USING CONVOLUTIONAL NEURAL NETWORKS (CNN)

In the field of image recognition, CNNs or ConvNets are categories of respectable neural systems. [9] CNN uses a multi-layer perceptron requiring minimal pre-training to train architecture to be highly effective in carrying out the task of recognition/classification. CNN has been designed as a biological process for the communication patterns in the animal's visual cortex between neurons [10]. In the fields of image classification, health image analysis, and natural language processing. CNN tends to perform better than other image and video recognition algorithms.




Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Fig 2. 1 A ConvNet Architecture Explained

## 2.1 LeNet Architecture

LeNet was one of the first CNN to become mainstream and laid the foundation for future research into the field of both CNN and multi-level perceptron. Yann LeCun [9] called this revolutionary work LeNet5 and was the result of many successful works since 1988. LeNet has mainly been developed for tasks like digits and zip codes. From this time, the MNIST dataset [11] has been created to test for accuracy each new neural network architecture.

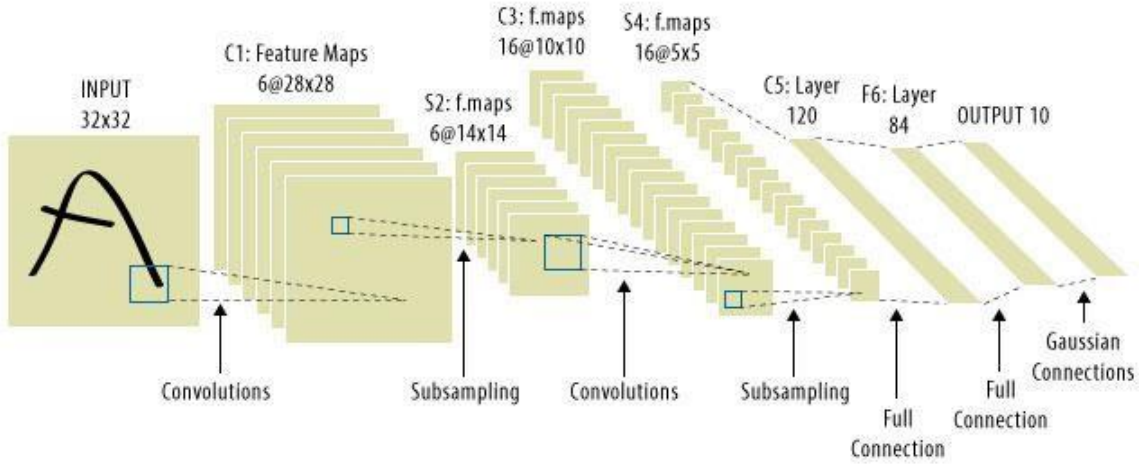



Fig 2. 2 A Simple ConvNet

The CNN in the above figure takes a 32x32 image from the MNIST handwritten image dataset and classifies it to a possible of 26 letters in the English alphabet.

The four key operations in the above image are as follow

1. Convolution
2. .Non-Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

These operations are the basic blocks of every CNN. Let's take a look to understand each in detail and how each operation works.

### 2.1.1 Convolution

Let take the handwritten MNIST numbers example. We have a 2D matrix with values of 0 to 255 in order to represent every pixel image as a matrix. ConvNets derive the word convolution from this step. The purpose of convolution is to extract features from the input data whether it be image, video or sequential data. Convolution works by preserving spatial relationships between pixels by learning image features using small ROI's. If we go back to considering each image as a matrix of pixels, convolution summarizes blocks of data from the matrix to a smaller dimension.

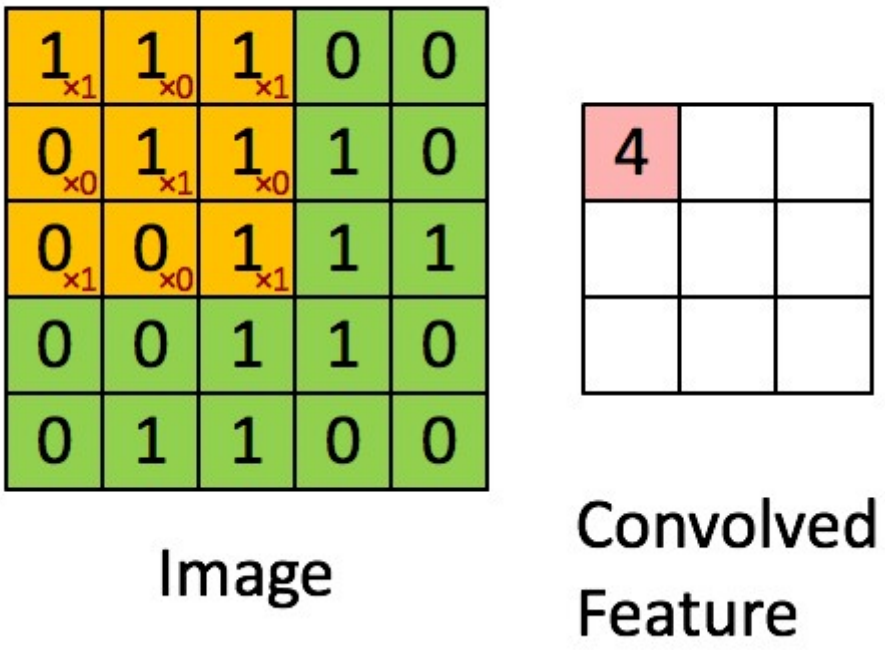



Fig 2. 3 Convolution Operation and Convolved Feature

The computation is achieved by computing element-wise multiplication and adding the outputs to get a representation of the ROI over the original image.

The 3x3 matrix is named in CNN terminology as a "filter" or "feature detector" and it is known as a Convolved Feature or Feature Map which is created by the sliding of a filter across the image. The process is repeated up to a series of feature maps for the input image.

Several options for convolution function can be used to generate a feature map which are

1. Edge Detection
2. Sharpen
3. Blur

All these are done simply by altering the filter matrix numerical values before the convolution operation [12] which means various filters obtain different results depending on the final achievement of the goal. All these are achieved just by changing the numeric values of the filter matrix before the convolution operation [12], this means that different filters achieve different results depending on the end goal.

The theory is that, through the backpropagation of gradients and loss between neurons' perceptions, CNN learns the values of the filters during the training process. Filters can be adjusted for different results in their scale, design and number of filters. The map size is a product of the following parameters

1. Depth: The volume depth determines the number of neurons attached to the stratum and the frequency of the input. Via backpropagation, we train these neurons to learn features. Let's say we take an image classification problem; the input layer takes an image and succeeding layers extract features from the image such as edges, blobs etc.

2. Stride: Stride refers to the number of pixels the filter passes across the image input. after a function map, it moves 4 pixels. There is a smaller feature map with a larger stride. With a smaller phase scale, receptive fields between columns are overlapping considerably. If the resulting fields are overlapping less, then spatial dimensions / feature maps are reduced[13].

3. Zero-padding: the input matrix is sometimes padded with zeroes to apply the filter on the image boundary elements. It is called broad convolution which, for non-padded images, is different from the narrow convolution. The size of the padding can be considered a hyperparameter and it provides control of the output volume spatial size.

Connectivity is an issue when dealing with high-dimensional entrances like pictures as it doesn't take spatial structure into account when linking all neurons to previous volumes. CNN benefits from the local connection between neurons of adjacent layers, the width of which is called a receptive field hyperparameter. The links in the space are always local but exceed the input volume range.

The principle of parameter sharing is used to monitor free parameters in convolutional layers. It is based on the assumption that a patch function is reusable and can be used in the neural network in several layers.

### 2.1.2 ReLU

The Non-Linearity operation is used after the convolution operation mentioned above. ReLU stands for the Rectified Linear Unit and is a non-linear operation [14]. It is applied to each element individually and it replaces all negative pixel values in the feature map to zero. The purpose of the ReLU is to introduce non-linearity since real-world training is non-linear and CNN should model to that.

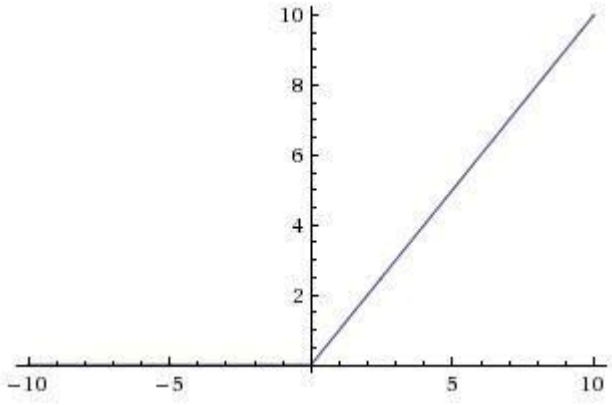



Fig 2. 4 ReLU Operation

The ReLU feature works by having a limit of input number and 0.

$$(x) = \max(0, x)$$

In comparison to other activation functions such as sigmoid and tanh, ReLU is also an extremely cheap activation function because it requires simpler mathematical operations and is a good idea when designing neural networks.

### 2.1.3 The Pooling Step

Pooling or subsampling is a layer that reduces the dimensionality of the characteristic maps created from the convolutional layer, with the main information kept [15]. There can be various types of swimming pools named as a max, average, total etc.

Max Pooling is when a certain window is specified and the largest part is taken from it. We can also take an average (average pooling) or summarize all the elements (Sum Pooling) instead of taking the greatest element. We keep moving the filter through the entire image as we progressed until we have a pooled layer of the kind defined in the architecture.

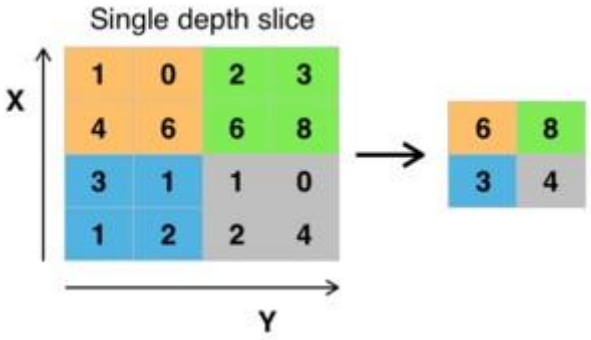



Fig 2. 5 Max Pooling Layer

The pooling layer further decreases the input image dimensionality and thus reduces the number of network parameters and calculations. It gives us a representation of the input image in a cleaner more concise form.

### 2.1.4 Fully Connected Layer

The fully connected layer is a multi-layer perception that uses activation functions in the output layer, for example, SoftMax. There are several functions such as SoftMax, but for thesis purposes, we will only discuss SoftMax. The term fully connected layer implies that every neuron in the layer is connected to every neuron in the previous layer. The convolutional layer along with the pulling layer generates a summarization of the original input image which is fed into the fully connected layer. The fully connected layer send gives an output which can be either classification or regression.

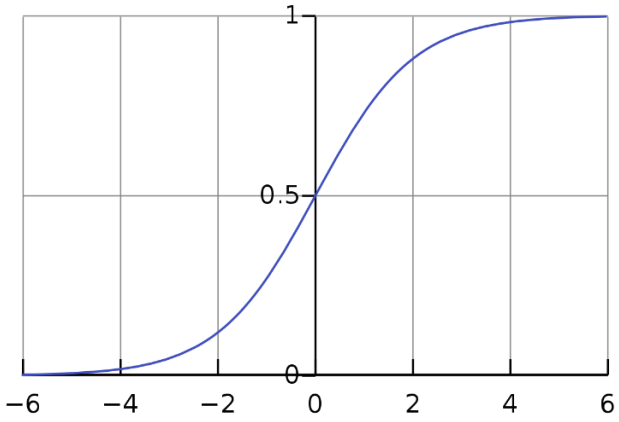



Fig 2. 6 SoftMax Function

The fully connected layer enables activity like backpacking, which are key features that allow a neural network to perform highly accurate classification. The SoftMax layer has a SoftMax function, which is used as the most used activation function in classification to squash a vector between zero and one.

## 2.2 The Complete Architecture

Now that different components of a CNN are covered, let's analyze how everything is integrated. The CNN preparation can be synthesized as follows:

1. To perform the convolution step on the input images start all the filters and parameters.

2. The design adopts an image of the input and goes in sequence through all of the

above steps and finds a result. The outputs are reversed via the network to "train" the network.

3. The architecture takes an image and goes sequentially across all of the above steps and finds an output. The outputs are reversed to "train" the network via the network.

4. Phase 2 is repeated until the expected effects are closed and unchanged.

The above-mentioned steps essentially train the neural network to perform a specific task. When a new image is introduced to the neural network after it has been trained it can predict a class of the image based on the training dataset. The training dataset plays a huge role in the accuracy and performance of the model.

## 2.3 Inception

First is a Google Net interface that aims to identify pictures in 2014 with incredible accuracy. It was developed by Yann LeCun and his Google team who developed an innovative solution to make the deep neural network more accurate than stacking layers. Initiated by a highly engineered and complex network, many techniques have been used to achieve high speed and accuracy.

Start v1 became Start v2, Start v3, Start v4, and Start ResNet since it was established... Consider the initiation module, which distinguishes the network from other models and ensures high levels of precision and performance.

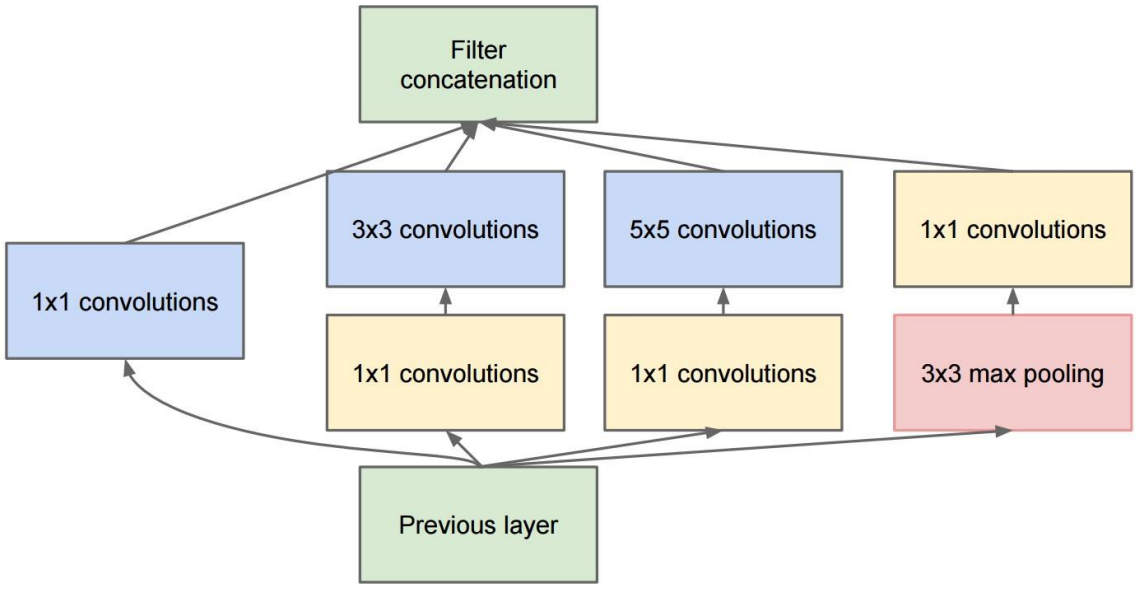



Fig 2. 7 Inception Module, Naive Version

Review the selection problem for the correct filter size and activation function type of the pooling layer when a neural network is constructed. It is not only time consuming but also trivial to test every combination of the trial and the error. The optimal combination may / should not be found. This solution is used in parallel before going on to the next layer with all possible parameters. If the next layer also consists of an Inception node, each characteristic map of the convolutions is transferred through the current layer mix. The theory is that if the network can do this for you, you don't have to think about the hyperparameter.

## 2.4 Regularization Methods

Regularization is an approach to additional information to solve a problem that is not present in order to avoid excess. Let's look at the various types of regularization provided by CNN.

1. Dropout: The fully connected layer is prone to overfitting due to the number of parameters it occupies [16]. It stops us from leaving. At each training stage, individual nodes are dropped out of the net with either probability 1-p or kept with probability. The removed nodes are then reinserted into the network with original weights. The probability that a hidden node would be dropped is 0.5 and during testing, we find an average of all possible 2n. Dropout decreases overfitting by avoiding training all nodes on all training data. Dropout also significantly improves training speed.

2. Drop Connect: DropConnect [17] is the generalizing drop-off that makes a 1-p likelihood to remove any connect. DropConnect is like dropouts, it adds instability, but weights instead of the output vectors are sparse.

# CHAPTER 3 SIGNIFICANT CHALLENGES

In this project, we developed a visual recognition system that uses the camera sensor to record the hand gesture in real-time and computer vision techniques for processing the frames captured. At each point of design and implementation, we have faced some important challenges that are addressed in this chapter.

## 3.1 Uncontrolled Environment

One of the key issues in developing any real-time computer vision projects is the uncontrolled environment. The dynamic changes in lighting, context and camera positions make it difficult to create a technology to isolate areas of interest from the whole environment. Our proposed program has great effects on the hand location of the environment lighting conditions and the camera position.

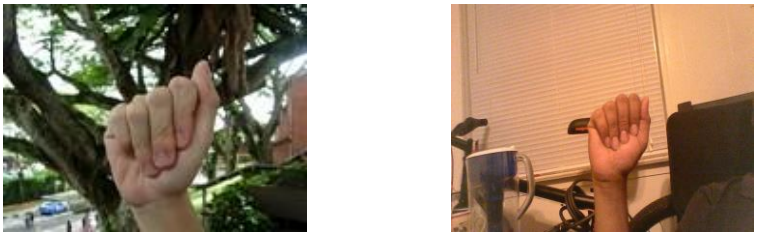


Fig 3. 1 Example Images of Hand Gestures with Variable Background and Lighting Conditions

## 3.2 Robust Hand Detection

It is a very complex task to detect and locate a hand field. Many common practices like motion or skin detection are not effective, as uncontrolled motion in the background and threshold-based extraction of skin area is impractical because of different, ethnic skin tones. An approach based on object detection is not a reliable technique as its push fingers can turn the hand into different sizes and shapes.

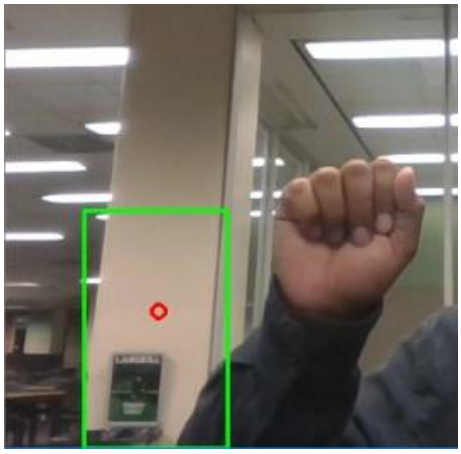


Fig 3. 2 False Detection of Hand Region in Skin Based Detection Technique

## 3.3 Occlusion

Because the image is a three-dimensional projection, occlusion is a very common obstacle in the accurate identification of the image. Some signs block the identification, in ASL recognition, of certain hand sections and figures that generate confusion in a function of image classification.

Fig 3. 3 Example of Occlusion: R Gesture can Look like D in 2D Projection because of Occlusion

## 3.4 Low Inter-Class Variability

The main technique in machine learning classification training as it decreases misclassification errors in the maintenance of high variability among interclass data and low variability among intraclass data. But few signs have very high visual similarity in American Sign Language, which can adversely affect system accuracy.


Fig 3. 4 Example of Low Inter-Class Variability: A gesture can be miss-classified as S because of Low Inter-Class Variability

## 3.5 Trigger to Recognition Process

In real-time systems, an initialization and termination trigger is critical. Hot-Words can be used in voice helpers to start the voice recognition process and, in a visual recognition system such as this, a vision-based trigger is required to activate the image recognition process.

## 3.6 LITERATURE REVIEW

Acknowledging the hand gesture is a well-developed field of research with many approaches to implementation; we study a variety of hand gesture recognition approaches in this chapter. The entire literature review can be divided into three groups: image processing / statistical modeling identification, deep learning-based recognition and classic machine-based acknowledgment.

### 3.6.1 Image Processing/ Statistical Modelling based approaches

Such solutions include either image processing with traditional image filters or gesture recognition by extracting spacetime features and developing models like Hidden Markov (HMM) and Finite State Machines (FSM) as an action recognition problem.

In 1996 Triesch and Malsburg [29] developed an ASL hand gesture recognition system to ensure that even images with complicated backgrounds are recognized accurately. In this method, the hand segmentation was removed by the input of hand pictures. The system uses Gabor filters to reflect the hand or extract features since Gabor filters are considered to mimic visual cortex receptive fields. When they obtained the Gabor features, they performed a technique called Elastic Bunch Graph Matching (EBMS), which matches a similarity to gesture recognition. For the research, they have chosen a subset of 10 sign language movements and the dataset consists of 657 photographs from 24 people, each with three different backgrounds. With a complex context condition and overall precision of 91 percent under all conditions, the system has

achieved a precision of 86.2 percent. Several experiments have followed the HMM modeling route to consider the hand gesture as a question of action detection.

Bergh et al.[8] suggested the use of hair wavelets and database search methods, using Haar classification to segment hand features used to extract hand gestures from a pool of gesture images that had previously been identified in the database. While the precision of both systems was substantially higher, only six gesture classes could be identified.

[3] Ashutosh et al. developed a robot control system based on HMM manual gesture recognition in 1998. They have got a form and optical-flow features in this system to build an HMM to recognize 12 different kinds of gestures. The authors claim that a 90% accuracy test with approximately 50 gesture inputs is achieved with the proposed system.

In 2001, in memory of numbers 0 to 9 Kapuscinski and Wysocki created a gesture identification system[14]. They have segmented skin-based hands using standardized color space and approximate hand positions with transforming Hit-miss. The hands are used to train two, three and four state HMMs with 600 gesture training data and provide 98 percent training accuracy on the training data and 97.2 percent test data accuracy.

In the case of recognition for temporal gestures, there were some methods using Bayesian networks which were very accurate. The analysis of Histogram of Gradients (HOG, Hidden Markov) and Hidden Markov (HMM), Starner et al.[ 28] has been developed by Liwicki et al. on a British Sign Language recognition system using 3-D gloves for hand motion and HMM for gesture recognition and achieved an accuracy of 99%. Suk et al. have proposed a model based on the Bayesian dynamic network for the recognition of temporal hand gestures, such as identification of circles and waving hands that exceeded 99 percent accuracy.

### 3.6.2 Classic Machine Learning-based approaches

In these methods, writers use classical machine learning techniques, such as Suppo rt Vector Machines SVM, Fuzzyc-mean, and neural network classifiers (ANN). Feature engineering plays an important role in system design in these classic machine-learning approaches. The category addresses a number of feature extraction and classification/ clustering techniques and their success in gesture recognition.

Amin and Yan [2] proposed in 2007 to use the Fuzzy-c-means clustering method for the identification of ASL function by Gabor. With the manual cropping of the hand area and a recorded dataset consisting of Handbilder with a consistent light background, they have eliminated hand segmentation in this system. Such images are assisted by Gabor filters accompanied by Principal Component Analysis (PCA), which is used to minimize

dimensionality as part of feature extraction. In the cross-validation test conducted under experimentation, this device performed 93.2% accurately.

In 2009 Huang et al. took a similar approach by using PCA Gabor for functional extraction and equipped a Vector Classification Support Machine[12]. They selected 11 hand gestures for this task and produced a dataset with 1320 images using 11 signatures. Through normalized RGB color space, skin color segmentation has been incorporated into the hand segmentation. When 660 images were tested, the machine achieved an accuracy of 95.2 percent.

By introducing IDSC-descriptors for functional extraction, Gopalan Gopalan and Dariush developed an SVM-based classification in 2009. In this process, they have segmented the skin in a color-based manner, where a Gaussian probabilistic model is constructed for filtering skin pixels within YCbCr's color space. The IDSC is used to calculate the similitude and the point match between different forms. Such features are used for training an 8 gesture level SVM classification. The accuracy of this technique was 85%.

In [26] 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. Signals are collected from the inputs for features such as Mean Absolute Value (MAV), Simple Square Integral (SSI), Average Change of Applications (AAC). The program has been trained to identify 26 ASL gestures and has achieved 95% training accuracy and 91.73% test accuracy.

In [24] ASL Recognition System with a Kinect Depth Camera has been developed by Otiniano-Rodriguez et al. The depth map for hand segmentation was used in this method, and the Invariant Feature Transformation Scale (SIFT) was used for hand images to draw features for SVM training. This program has an accuracy of 91 percent with a combination of RGB and depth map images.

The method used by Chuan et al is similar but instead of cyber or Kinect sensors, it is a leap motion sensor[6]. The data collected from the leap motion sensor is used to train both classifiers and SVMs in Knearest Neighborhoods in 7900 images. The exactness of this technology for KNN and SVM was 72.7 and 79.7 percent respectively.

Bag-of-features in SIFT has been included by Dardas et al [7] in the framework of SVM gesture reconnaissance. This device vectorizes the features obtained from SIFT with Vector Qualification (VQ) and extracts key points with the K-mean clustering

process. These key points, also known as code vectors, are used to construct SVM codebooks. The SVM has five fingers of the hand recognized and resulted in an average accuracy of 90%.

Lamari et al. [6] have presented the Artificial Neural Networks gesture recognition program. Colored gloves are used in this method for simple hand segmentation and morphological examination of the principal component of hand pictures for the extraction of features. A 3-layered feed-forward perceptron system with a Sigmoid activation feature, trained with backpropagation algorithm, is the Neural network used for classification. The network architecture consists of 20 or 24 neurons and the secret layer includes 42 neurons. For 26 ASL finger classification, the output layer consists of 26 neurons. This qualified neural network has provided 97.94 percent training precision and 89.06 percent test accuracy.

Mekala et al. [19]used the Gaussian average model of hand segmentation history subtraction for neural networks based ASL recognization.[ 19] Shape descriptors and motion vectors are extracted from segmented frames and a three-layered neural network is equipped. The authors claim to earn 100% appreciation for A-Z gestures.

The Ethiopian Sign Language (ESL) system of recognition has been developed by Admasu and Raimond using the neural networks which have achieved a 98.5% recognition rate[1]. In this method, the hand images were hand-processed and some basic hand segmentation was carried out for reaching the hand field. Gabor filters are used to extract functions and train a neural network of three stages. The network architecture comprises 20 neurons input, 100 neurons secret and 34 neurons output.

### 3.6.3 Deep learning-based approaches

The research has been conducted in different countries or regions, such as Pigou et al. [ 23 ] proposed the Italian sign-language recognition system using CNNs which recorded 95.6 percent precision for 20 training classes but poorly performed for the test set. Nagi et al. [20] proposed the HCI gesture recognition method with the use of CNN. [13] The US sign language recognition system Kang et al. have evolved using depth map pictures that have been trained in convolutional neural networks. In this method 3100 maps of depth for 31 gestures have been used that include all the alphabets and numbers except J, Z because they need time and 2/V, 6/W have a single class mark. They selected Alexnet architecture for training and achieved validation of 99 percent and an accuracy of 85 percent on the test dataset.

[5]A CNN based ASL gesture recognition program, which includes both ASLalphabets and digits gestures, has also been developed by Bheda and Radpour. In this method, both NZ ASL and its own dataset have been used for all actions. The program has achieved an alphabet gesture validation accuracy of 82.5% and an ASL digit gesture accuracy of 97%.

[9] A similar CNN-based ASL recognition system has been introduced by Garica and Viesca. The system uses googlenet with data sets of university Surrey and Massey using transfers. The model was trained to recognize A-Y except for J and the authors noted that the recognition of the A-E classification is the highest relative to other A-K and A-Y models (except J).

# Chapter 4 System Design

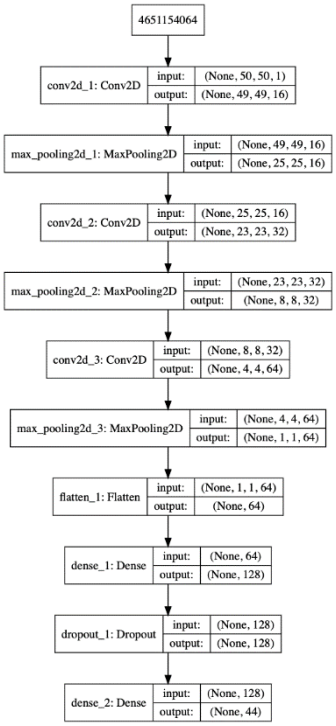



Fig. 4. 1 Our Hand Gesture Recognition System

## 4.1 Input

The input unit consists of a camera sensor that captures live frames containing movements of the front hand and transmits them to the cascading units. The objective of this project is to recognize gestures on native RGB frames from a web camera, rather than by applying increment techniques, such as depth cameras or specially designed hand gloves.

## 4.2 Model Training

In order to develop the model for this system, the web camera has been used to fetch the gesture first. After this, the CNN network has been defined with 8 layers. The time taken by this system was 9 hours on a system with the deep learning framework Keras and framework which was used for capturing gestures from the web camera is OpenCV.

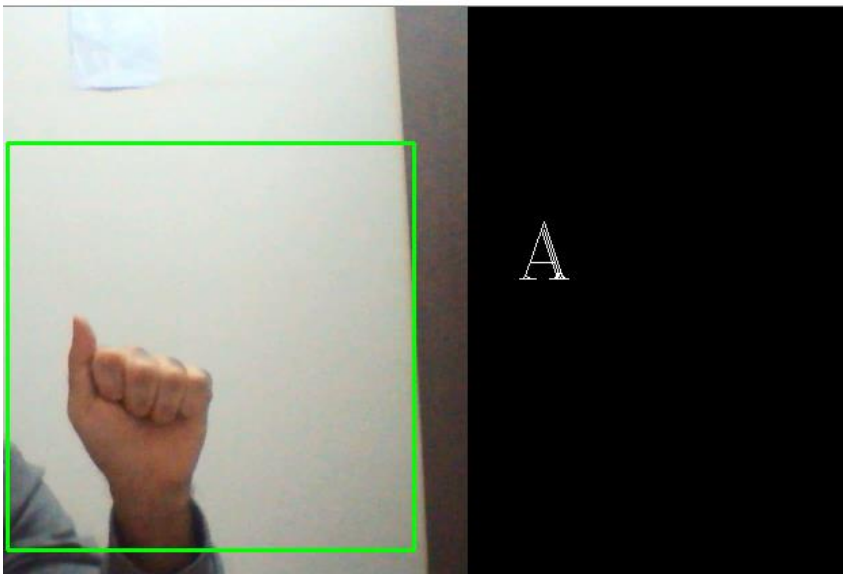



Fig. 4. 2 First Sign outcome

## 4.3 Hand Gesture Recognition

In our design, the recognition of gestures by hand is also a two-story process, the classification of hands/segments and gestures. We used a skin color filter for hand detection. This technique is used to achieve the desired precision and speed. We have equipped a bank of CNNs to understand the gesture and assembled its classification tests with a bagging approach. Ensemble learning is a revolutionary approach to traditional computer instruction. Following is a short description of ensemble learning.

**Ensemble Learning:** As the title describes, ensemble learning is a technique of merging the same task as a group of learning machines. This technique has been used to improve classification efficiency because it decreases the probability of model overfitting during training and mitigates the incidence of bias and variance factors that will adversely impact the classifier's performance on a new data set. Backaging is a common and easy approach to a group of classifiers ensemble. Bagging stands for Bootstrap aggregation. For each classifier, the bagging forecast scores are aggregated to produce a final score, which is used to assign the expected class Tag.

# Chapter 5 Sign Language Dataset

## 5.1 Sign Language dataset

The analysis was based on totally newly created datasets through webcam. This dataset was an American Sign Language dataset that includes both alphabets as well as numbers. The signers were presented with video stimulation, as they would naturally produce the sign. The video stimuli also contained changes of a sign in the dictionary. Nevertheless, the signatories did not always produce the same sign, the signatures were produced differently. The modifications were correctly listed and numbered.

Due to several drawbacks in the previous dataset, a new dataset was used. The emergence of several definitions of a single expression contributed to the model is less accurate because of the wrong traits. Due to the fact that many irrelevant features are trained in model accuracy, the presence of variation of clothing and hand expressions also led to a reduced level of accuracy.

We decided to use the most common 30 alphabets in American Sign Language as a starting point and ten numbers ranging from 0 to 9. Each alphabet was individually recorded 1200 times while keeping the same clothes and background. The variation between multiple gestures accounted for variation in practical scenarios.

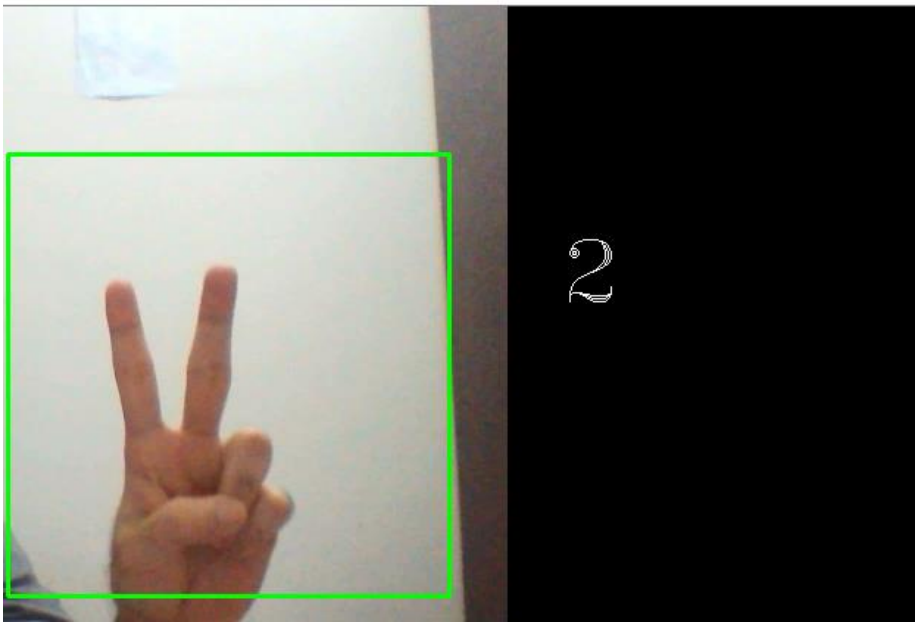



Fig. 5. 1 An Example from the Custom Dataset

The videos were recorded on a laptop camera at 60 frames per second. Both videos have been shot with a constant resolution of 720p. Each video was then preprocessed before feeding it to the model. Preprocessing the model involved breaking down each video to a size of 300 frames, which was used as the standard length of gestures. The dataset was then augmented to increase the size of training and test for better performance in training. Dataset image should be mentioned here.

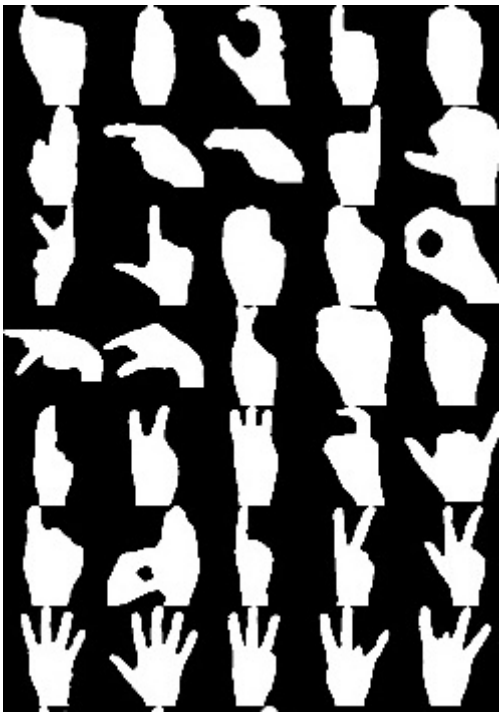


Fig. 5. 2 All datasets

## 5.2 IMPLEMENTATION

### 5.2.1 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.

### 5.2.2 Skin Color Extraction

We have adopted a 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 the hand region is extracted from real-time frames using hand 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 YCbCr channels respectively.

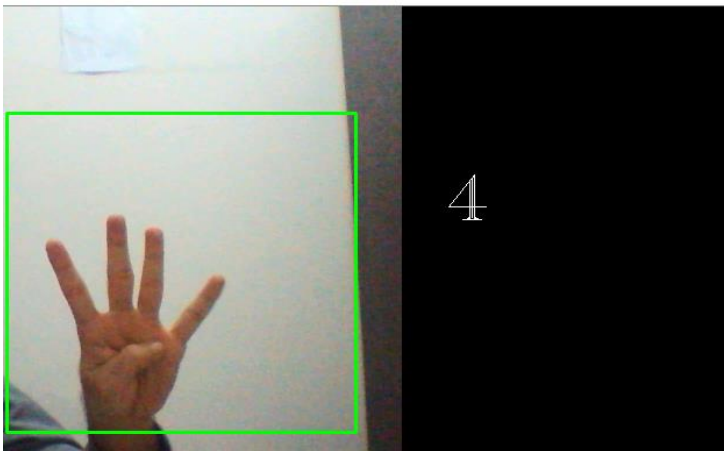



Fig. 5. 3 Hand Pixels Extracted

### 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 the Convex hull algorithm.

For a given set of Euclidian points, the 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, the convex hull also allows us to detect the tip of the fingers and center of the hand with few additional geometric computations which can be used for further validation of accuracy in hand region extraction.

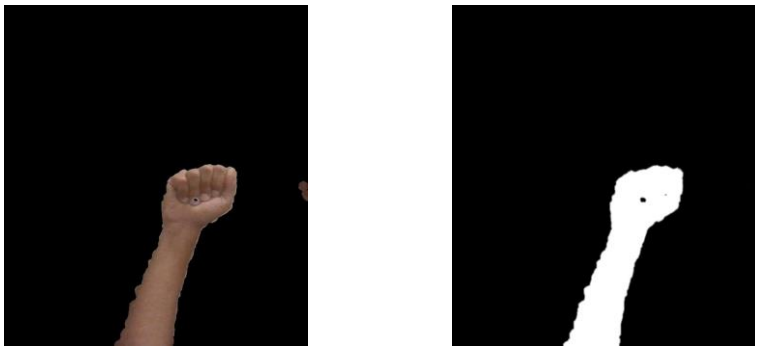


Fig. 5. 4 Extracted Hand Region

### 5.2.4 Hand Gesture Recognition

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

**Training**

In our system, we have trained four independent deep convolutional neural networks through the transfer learning process using Keras 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 fingerspelling 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 to 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 the 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,

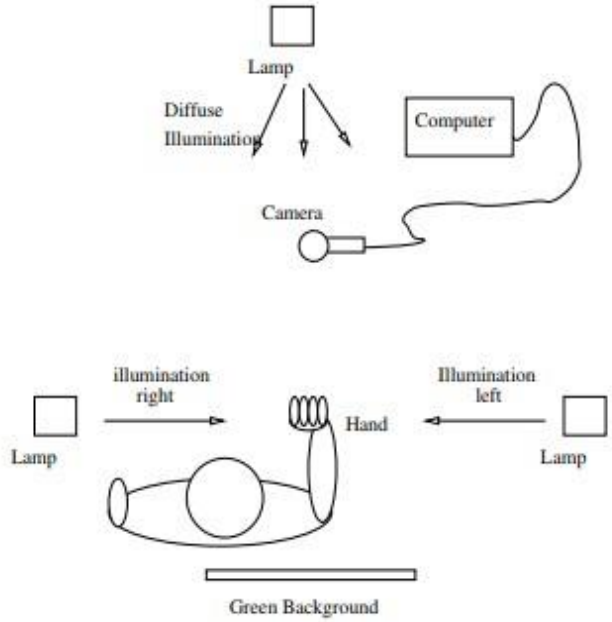



Fig. 5. 5 Massey University Experimental Set-Up for Capturing ASL Dataset (A Bird's Eye View) [4]

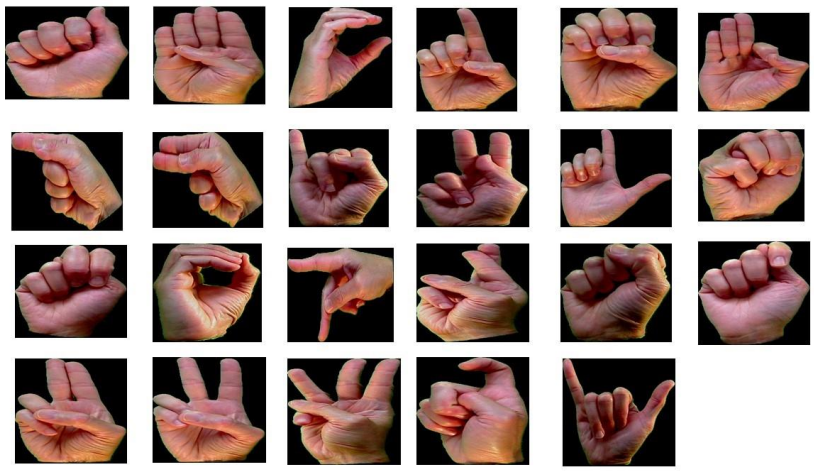


Fig. 5. 6 Sample Images from Massey University American Sign Language Dataset

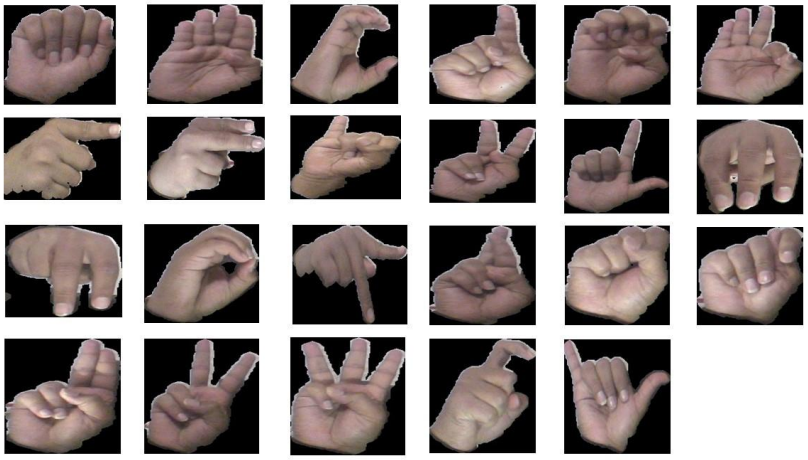


Fig. 5. 7 Sample Images from In-House Generated American Sign Language Dataset

The images with 2400 counting for each class are used as the dataset for this project

### 5.2.5. MobileNets

Since Alexnet, the winner of the 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, 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 resources. 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 depthwise separable convolution. The conventional convolution operation has an 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 are called depthwise separable convolution. This modification of a 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 a number of Multiplication and Addition Operations

(MACs) they perform is visualized in the below figure,

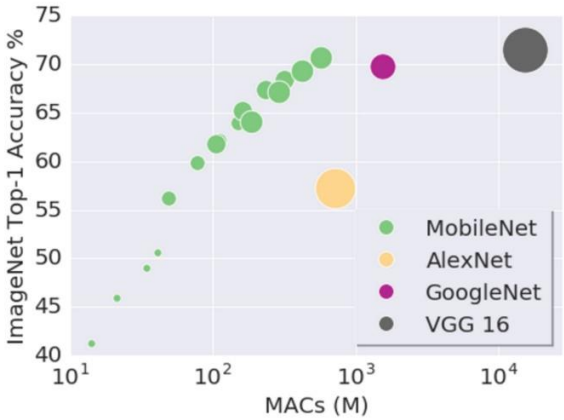



Fig. 5.8 Comparison of Popular CNN's Accuracies with Respect to their MACs

The model architecture of CNN 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 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| $5\times$ Conv dw / s1 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| $5\times$ 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 $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

Fig. 5. 8 CNN Architecture

**Training Configuration**

As discussed earlier, we have trained American Sign Language datasets using the transfer learning technique. In transfer learning, we have created datasets (2400 for one class) for both English alphabets as well as Numbers. 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

is 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.

**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 the bagging techniques 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 final prediction confidence. In real-time, while performing image classification there is a chance of inputting transition frames that are captured while a 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 a final prediction score.

**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,

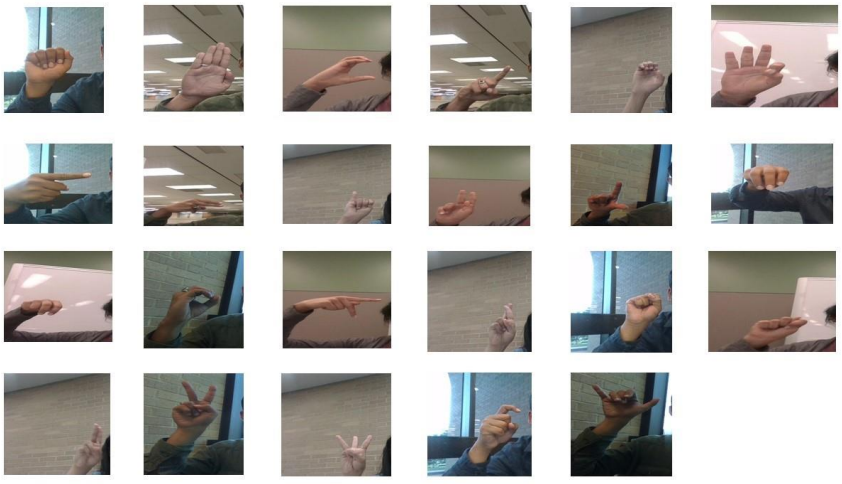


Fig. 5. 9 Sample Images from Test Set

# Chapter 6 conclusion and future works

To evaluate the performance of our system, we use training and validation accuracies, optimization 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 the training set that is assigned with the 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 the training set is used for validation and validation accuracy provides the percentage of images in a validation set that is assigned with a correct class label. The training and validation accuracy trends four classifiers over the configured learning iterations are visualized in the plots below,


Fig. 6. 1 Training/Validation Accuracy (CNN @ 10000 Iterations)

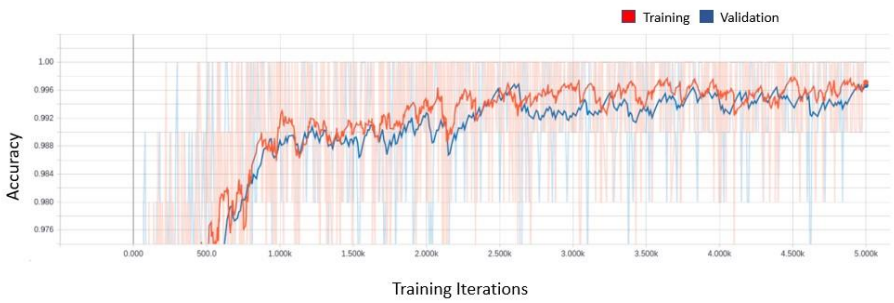



Fig. 6. 2 Training/Validation Accuracy (CNN @ 5000 Iterations)


Fig. 6. 3 Training/Validation Accuracy (CNN @ 15000 Iterations)

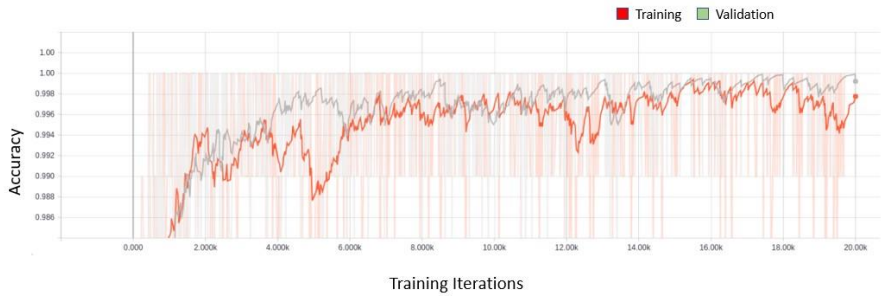



Fig. 6. 4 Training/Validation Accuracy (CNN @ 20000 Iterations)

## 6.2 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 on Top-1 accuracy and Top-2 accuracy. Top-1 accuracy is a regular accuracy measurement where the predicted class labels with the highest probability (top 1 probability) are compared with ground truths for correctness. Whereas in Top-2 accuracy we check for the expected label in the top 2 predicted class labels. Top-2 accuracy is used in our system as we use head pose estimation to validate the correctness of the first prediction and obtain the second-best prediction if the first label is wrong. This measure will help us to understand the significance of the head pose estimation system. The confusion matrices for all four classifiers and ensemble classifier is as follows,

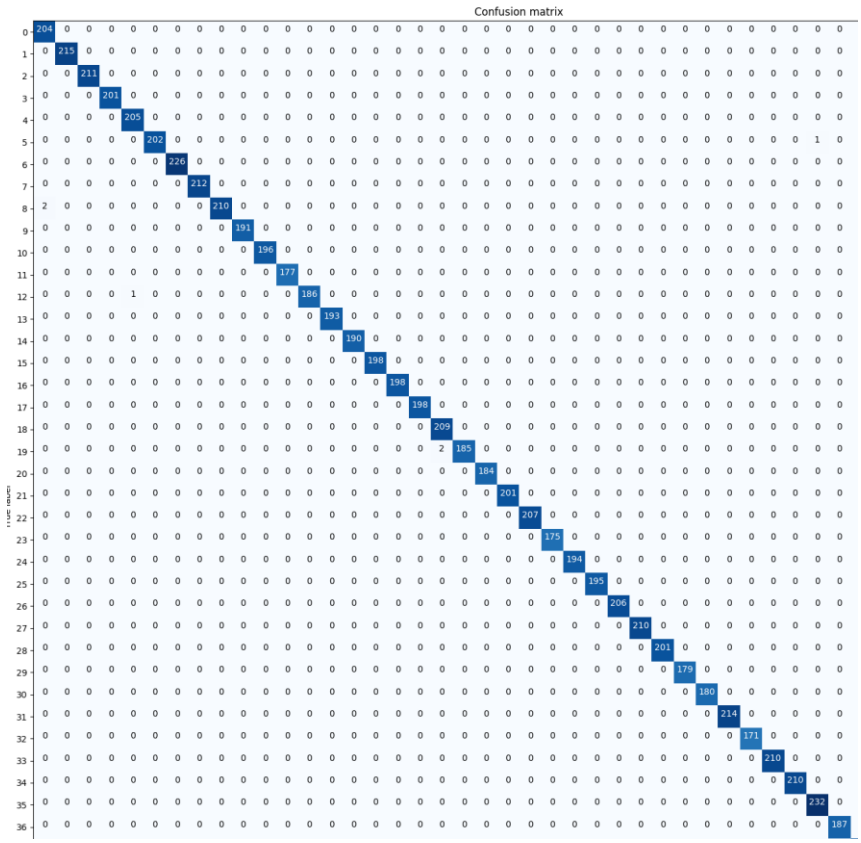



Fig. 6. 5 Confusion Matrix

## 6.3 Future works

From observation of above performance metrics, it is clearly noticed that hand 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 ensemble model was useful in attaining certain level of uniformity in terms of accuracy among all the gestures. Ensemble technology also had a minimum difference between Top 1 and Top 2, which means that the classifier shows consistent performance.

Convolutional neural networks ensemble has played a major role in training the precise CNN model with relatively small databases. It is one of the most unique research innovations in our system which has enhanced system performance and transfer learning.

The key location estimation is a very useful factor in a real-time system as the identification of movement based on image classification does not always result in 100%

classification. The technique used to identify hand gesture is also robust and cannot misidentify.

Our intention was to create an American Sign Language recognition device focused on a real-time computer vision, with no special equipment such as a Kinect 3D camera or a sensor glove. We also decided to use one of the normal measures in our daily lives to incorporate touchless feedback systems. This system performs gesture recognition using the embedded system compatible CNN. Along with a standard dataset from Massey University we have created our own ASL dataset for training independent models. These models through bagging approach has yielded an overall accuracy of 94% in classification and helped in obtaining near uniform accuracy across all the letters.

The whole system is built via python, OpenCV and Keras and is used as a desktop-based application. There are also several enhancements and changes to be made alongside existing apps and these will be done in future works.

# 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 color 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 the 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 visionbased 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, *Fingerspelling 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 Soukupov´a 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 complex 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].

# Appendix

## A Creating datasets through webcam

```
import cv2
import pickle, os, sqlite3, random

image_x, image_y = 50, 50

def get_hand_hist():
  with open("hist", "rb") as f:
    hist = pickle.load(f)
  return hist

def init_create_folder_database():
  # create the folder and database if not exist
  if not os.path.exists("gestures"):
    os.mkdir("gestures")
  if not os.path.exists("gesture_db.db"):
    conn = sqlite3.connect("gesture_db.db")
    create_table_cmd = "CREATE TABLE gesture ( g_id INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT UNIQUE, g_name TEXT NOT NULL )"
    conn.execute(create_table_cmd)
    conn.commit()

def create_folder(folder_name):
  if not os.path.exists(folder_name):
    os.mkdir(folder_name)

def store_in_db(g_id, g_name):
  conn = sqlite3.connect("gesture_db.db")
  cmd = "INSERT INTO gesture (g_id, g_name) VALUES (%s, \'%s\')" % (g_id, g_name)
  try:
```

```
        conn.execute(cmd)
    except sqlite3.IntegrityError:
        choice = input("g_id already exists. Want to change the record? (y/n): ")
        if choice.lower() == 'y':
            cmd = "UPDATE gesture SET g_name = \'%s\' WHERE g_id = %s" % (g_name, g_id)
            conn.execute(cmd)
        else:
            print("Doing nothing...")
            return
    conn.commit()

def store_images(g_id):
    total_pics = 1200
    hist = get_hand_hist()
    cam = cv2.VideoCapture(1)
    if cam.read()[0]==False:
        cam = cv2.VideoCapture(0)
    x, y, w, h = 300, 100, 300, 300

    create_folder("gestures/"+str(g_id))
    pic_no = 0
    flag_start_capturing = False
    frames = 0

    while True:
        img = cam.read()[1]
        img = cv2.flip(img, 1)
        imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([imgHSV], [0, 1], hist, [0, 180, 0, 256], 1)
        disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
        cv2.filter2D(dst,-1,disc,dst)
        blur = cv2.GaussianBlur(dst, (11,11), 0)
        blur = cv2.medianBlur(blur, 15)
        thresh = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
```

```
    thresh = cv2.merge((thresh,thresh,thresh))
    thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
    thresh = thresh[y:y+h, x:x+w]
    contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[1]

    if len(contours) > 0:
      contour = max(contours, key = cv2.contourArea)
      if cv2.contourArea(contour) > 10000 and frames > 50:
        x1, y1, w1, h1 = cv2.boundingRect(contour)
        pic_no += 1
        save_img = thresh[y1:y1+h1, x1:x1+w1]
        if w1 > h1:
          save_img = cv2.copyMakeBorder(save_img, int((w1-h1)/2) , int((w1-h1)/2) , 0, 0,
cv2.BORDER_CONSTANT, (0, 0, 0))
        elif h1 > w1:
          save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1-w1)/2) , int((h1-w1)/2) ,
cv2.BORDER_CONSTANT, (0, 0, 0))
        save_img = cv2.resize(save_img, (image_x, image_y))
        rand = random.randint(0, 10)
        if rand % 2 == 0:
          save_img = cv2.flip(save_img, 1)
        cv2.putText(img, "Capturing...", (30, 60), cv2.FONT_HERSHEY_TRIPLEX, 2, (127, 255,
255))
        cv2.imwrite("gestures/"+str(g_id)+"/"+str(pic_no)+".jpg", save_img)

    cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
    cv2.putText(img, str(pic_no), (30, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (127, 127, 255))
    cv2.imshow("Capturing gesture", img)
    cv2.imshow("thresh", thresh)
    keypress = cv2.waitKey(1)
    if keypress == ord('c'):
      if flag_start_capturing == False:
        flag_start_capturing = True
      else:
```

```
            flag_start_capturing = False
            frames = 0
        if flag_start_capturing == True:
            frames += 1
        if pic_no == total_pics:
            break

init_create_folder_database()
g_id = input("Enter gesture no.: ")
g_name = input("Enter gesture name/text: ")
store_in_db(g_id, g_name)
store_images(g_id)
```

# B Model training Source Code

```
import numpy as np
import pickle
import cv2, os
from glob import glob
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint
from keras import backend as K
from keras.utils import plot_model

K.set_image_dim_ordering('tf')

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

```
def get_image_size():
    img = cv2.imread('gestures/1/100.jpg', 0)
    return img.shape


def get_num_of_classes():
    return len(glob('gestures/*'))


image_x, image_y = get_image_size()


def cnn_model():
    num_of_classes = get_num_of_classes()
    model = Sequential()
    model.add(Conv2D(16, (2, 2), input_shape=(image_x, image_y, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=(3, 3), padding='same'))
    model.add(Conv2D(64, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(num_of_classes, activation='softmax'))
    sgd = optimizers.SGD(lr=1e-2)

    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
    filepath = "cnn_model_keras2.h5"
    checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True,
mode='max')
    callbacks_list = [checkpoint1]
```

```
    plot_model(model, to_file='model.png', show_shapes=True)
    return model, callbacks_list


def train():
    with open("train_images", "rb") as f:
        train_images = np.array(pickle.load(f))
    with open("train_labels", "rb") as f:
        train_labels = np.array(pickle.load(f), dtype=np.int32)

    with open("val_images", "rb") as f:
        val_images = np.array(pickle.load(f))
    with open("val_labels", "rb") as f:
        val_labels = np.array(pickle.load(f), dtype=np.int32)

    train_images = np.reshape(train_images, (train_images.shape[0], image_x, image_y, 1))
    val_images = np.reshape(val_images, (val_images.shape[0], image_x, image_y, 1))
    train_labels = np_utils.to_categorical(train_labels)
    val_labels = np_utils.to_categorical(val_labels)

    print(val_labels.shape)

    model, callbacks_list = cnn_model()
    model.summary()

    model.fit(train_images, train_labels, validation_data=(val_images, val_labels), epochs=20,
batch_size=500,
              callbacks=callbacks_list)
    scores = model.evaluate(val_images, val_labels, verbose=0)
    print("CNN Error: %.2f%%" % (100 - scores[1] * 100))


# model.save('cnn_model_keras2.h5')
```

```
train()
K.clear_session()
```

# C Model Testing Source Code

```
import cv2, pickle
import numpy as np
import sqlite3
from keras.models import load_model


def get_image_size():
    img = cv2.imread('gestures/0/100.jpg', 0)
    return img.shape


prediction = None
model = load_model('cnn_model_keras2.h5')
image_x, image_y = get_image_size()


def keras_process_image(img):
    img = cv2.resize(img, (image_x, image_y))
    img = np.array(img, dtype=np.float32)
    img = np.reshape(img, (1, image_x, image_y, 1))
    return img


def keras_predict(model, image):
    processed = keras_process_image(image)
    pred_probab = model.predict(processed)[0]
    pred_class = list(pred_probab).index(max(pred_probab))
    return max(pred_probab), pred_class
```

```
def get_pred_text_from_db(pred_class):
    conn = sqlite3.connect("gesture_db.db")
    cmd = "SELECT g_name FROM gesture WHERE g_id=" + str(pred_class)
    cursor = conn.execute(cmd)
    for row in cursor:
        return row[0]


def split_sentence(text, num_of_words):
    '''
    Splits a text into group of num_of_words
    '''
    list_words = text.split(" ")
    length = len(list_words)
    splitted_sentence = []
    b_index = 0
    e_index = num_of_words
    while length > 0:
        part = ""
        for word in list_words[b_index:e_index]:
            part = part + " " + word
        splitted_sentence.append(part)
        b_index += num_of_words
        e_index += num_of_words
        length -= num_of_words
    return splitted_sentence


def put_splitted_text_in_blackboard(blackboard, splitted_text):
    y = 200
    for text in splitted_text:
        cv2.putText(blackboard, text, (4, y), cv2.FONT_HERSHEY_TRIPLEX, 2, (255, 255, 255))
        y += 50
```

```
def get_hand_hist():
    with open("hist", "rb") as f:
        hist = pickle.load(f)
    return hist


def recognize():
    global prediction
    cam = cv2.VideoCapture(0)
    hist = get_hand_hist()
    x, y, w, h = 300, 100, 300, 300
    while True:
        text = ""
        img = cam.read()[1]
        img = cv2.flip(img, 1)
        img = cv2.resize(img, (640, 480))
        # imgCrop = img[y:y + h, x:x + w]
        imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([imgHSV], [0, 1], hist, [0, 180, 0, 256], 1)
        disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (10, 10))
        cv2.filter2D(dst, -1, disc, dst)
        blur = cv2.GaussianBlur(dst, (11, 11), 0)
        blur = cv2.medianBlur(blur, 15)
        thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
        thresh = cv2.merge((thresh, thresh, thresh))
        thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
        thresh = thresh[y:y + h, x:x + w]
        (openCV_ver, _, __) = cv2.__version__.split(".")
        contours = None
        if openCV_ver == '3':
            contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[1]
        elif openCV_ver == '4':
            contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[0]
        if len(contours) > 0:
```

```
            contour = max(contours, key=cv2.contourArea)
            # print(cv2.contourArea(contour))
            if cv2.contourArea(contour) > 10000:
                x1, y1, w1, h1 = cv2.boundingRect(contour)
                save_img = thresh[y1:y1 + h1, x1:x1 + w1]

                if w1 > h1:
                    save_img = cv2.copyMakeBorder(save_img, int((w1 - h1) / 2), int((w1 - h1) / 2), 0, 0,
                                                  cv2.BORDER_CONSTANT, (0, 0, 0))
                elif h1 > w1:
                    save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1 - w1) / 2), int((h1 - w1) / 2),
                                                  cv2.BORDER_CONSTANT, (0, 0, 0))

                pred_probab, pred_class = keras_predict(model, save_img)

                if pred_probab * 100 > 80:
                    text = get_pred_text_from_db(pred_class)
                    # print(text)
            blackboard = np.zeros((480, 640, 3), dtype=np.uint8)
            splitted_text = split_sentence(text, 2)
            put_splitted_text_in_blackboard(blackboard, splitted_text)
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            res = np.hstack((img, blackboard))
            cv2.imshow("Recognizing gesture", res)
            cv2.imshow("thresh", thresh)
            if cv2.waitKey(1) == ord('q'):
                break


if __name__ == '__main__':
    recognize()
```

# D 学位论文数据集

| 关键词 | 密 级 | 中图分类号 |
|---|---|---|
| Gesture Recognition; American Sign Language, Computer Vision, CNN | 公开 | TP |

| 学位授予单位名称 | 学位授予单位代码 | 学位类别 | 学位级别 |
|---|---|---|---|
| 重庆大学 | 10611 | 专业学位 | 硕士 |

| 论文题名 | 并列题名 | 论文语种 |
|---|---|---|
| 实时美国手语识别——使用深层卷积神经网络和计算机视觉 | 无 | 英文 |

| 作者姓名 | Nouman Ahmad | 学 号 | L1800103 |
|---|---|---|---|
| 培养单位名称 | | 培养单位代码 | |
| 重庆大学 | | 10611 | |

| 学科专业 | 研究方向 | 学制 | 学位授予年 |
|---|---|---|---|
| 软件工程 | 手势 | 2 年 | 2020 |
| 论文提交日期 | 2020.06 | 论文总页数 | 30 |
| 导师姓名 | Hong Sha | 职 称 | Prof. |
| 答辩委员会主席 | | Zhang Yi | |

电子版论文提交格式

文本（√） 图像（） 视频（） 音频（） 多媒体（） 其他（）

# Acknowledgements

I would like to thank all the people who have helped and inspired me during the pursuit of my Master's Degree. Initially, First, I would like to thank my supervisor, Dr. Hong Sha for his guidance during my research and study at Chongqing University. Dr. Hong Sha was constantly available and willing to help with my research making my exploration experience a smooth and remunerating one. I likewise might want to thank all my lab mates for making the lab a friendly work environment. My most profound appreciation and gratefulness to my family for their enduring affection and backing for a mind-blowing duration; this thesis would have basically been inconceivable without their assistance. I am totally obliged to my dad, for his mind-boggling care and support. Being the individual he will be, he strived from the start to help the family and saved no push to give the best life to me. With respect to my mom, I couldn't have requested more from her. No words can express my gratefulness to her for her everlasting adoration and her steady help at whatever point I experienced challenges. At long last, I can't forget my siblings who have dependably been there for me with their petitions and, guidance.

Most importantly, thanks to God for all the blessings throughout my life and for allowing me this chance to accomplish a higher level of education. My life has turned out to be increasingly plentiful.

***Nouman Ahmad***

二O二O年五月 于重庆