RECOGNIZING HAND GESTURES USING TEMPORALLY BLENDED IMAGE DATA AND DEEP LEARNING

A Thesis

by

SHUBHARAJ PRADEEP ARSEKAR

BE, Goa University, 2013

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

December 2018

December 2018

RECOGNIZING HAND GESTURES USING TEMPORALLY BLENDED IMAGE DATA AND DEEP LEARNING

A Thesis

by

SHUBHARAJ PRADEEP ARSEKAR

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Dr. Scott A. King, PhD
Chair

Dr. Alaa A. Sheta, PhD
Co-Chair

Dr. Byung Cheol Bruce Lee, PhD
Committee Member

December 2018

ABSTRACT

Hand gestures can allow for natural approach to human-computer interaction. A novel low computation Hand Gesture Recognition System (HGRS) using temporally blended image data with a convolutional neural network (CNN) is presented. The goal of HGRS is to recognize hand gestures in an optimized and efficient way. We created a dataset using Kinect depth and body data stream frames. The dataset comprised of eight different hand gestures, each gesture was performed with the right hand within a duration of three seconds. Data is first processed by segmenting the hand from the background using body data joints mapped onto depth data. Reduction in the computation of the HGRS was achieved by blending the temporal depth data frames into a single frame. The blending of temporal depth data frames is defined as the addition of the frames into a single frame by increasing the intensity of each consecutive frame. The resolution of the depth data frames was reduced to an empirically evaluated frame size of 50×50 which further improved the computational efficiency of HGRS. We trained and validated a CNN model for hand gesture classification which consists of three convolutional layers each followed by a max pooling layer, and two fully connected layers in the end. We tested the performance of the model and observed a test accuracy of 98.45%. We performed a quantitative analysis to measure the overall performance of the model.

ACKNOWLEDGEMENTS

First of all, I have to thank my thesis advisor, Dr. Scott A. King and comittee members Dr. Alaa Sheta and Dr. Byung Lee for their friendship, their valuable suggestions throughout my journey, their omnipresent criticism to my work which constantly kept me to strive harder to achieve my goals, their availability to clarify my queries and my inexperience, and their guidance during this difficult process of writing a dissertation. I will never have words to thank your persistence with me. A special thanks to Dr. Scott A. King for the wise words that allowed me to continue forward in difficult times during this work, his stay after office hours to discuss vital inputs on my work during the defense schedule. I would also like to thank Dr. Alaa Sheta and Dr. Byung Lee for all the ideas discussed during the scheduled meetings, for their ability to focus on issues that were very important for the proper development of this work. I would like to thank to all the friends that supported and encouraged me during this difficult journey. I would also like to thank all the people who volunteered to test all the developed models during this work and that also participated in the system testing phase, which was of utmost importance for the experiments carried out allowing to successfully complete this thesis. I would like to thanks my parents and my brother for life and for all the opportunities they were able to provide me during my life time, for their patience and dedication, for their love, for the constant words of encouragement and for being there for me when needed

DEDICATION

I would like to dedicate this thesis to my loving Mom, always supportive Dad and my helpful Bhaiya, it was through their persistence that I was able to achieve my dreams.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES PAGE

LIST OF TABLES

TABLES PAGE

CHAPTER I: INTRODUCTION

Hand gesture recognition is a growing field of interest due to advancements in computer vision. Gestures are a very important part of the human routine. According to the Oxford dictionary [36], the definition of gesture is the movement of a part of the body, especially a hand or head, to express an idea or meaning. Gestures include movement of the face, hands, or other parts of the body. They are part of both verbal communication such as presentations, lectures, conversations as well as non-verbal communication like sign language.

Gesture recognition is a technique that is used to design a system that can comprehend gestures to control computer systems. The beauty of a gesture recognition system is that the user is not required to understand the technical jargon when using hand gestures for controlling a system. Thus, a layman's knowledge is sufficient. In everyday communication, gestures play an important role not only to enforce the understanding that one would provide through speech but also drive home certain meaningful phrases, which otherwise would have been impossible to be conveyed through the speech alone. Therefore, to make human computer interaction as natural as possible, computers should be able to recognize gestures along with speech.

The most common human-computer interactions used by individuals are devices such as keyboards, joysticks, controllers, and mice to control systems and machines. Although these are carefully designed for simple and easy interaction with the users, they suffer from inherent difficulties when it comes to massive data inputs and reasonable speed. This limitation has become even more evident with the dramatic evolution of computers in the fields of storing capacity and processing speed. For these reasons, we need new input methodologies assimilating at a greater degree, the way people communicate with the system that is, speech and gestures. Hand gestures could nicely serve as an additional means of providing instructions to a computer or a robot. The Carnegie Mellon University robotics laboratory implemented a gesture based robot which was then given the task of cleaning through simple gestures [49].

Furthermore, gestures are a necessity in Virtual Reality (VR), where the users have to perform

hand movements to interact and manipulate the surroundings of a virtual environment.

1.1 Problem Description

We design and implement a hand gesture recognition system with the sole purpose of meeting the standard of natural interaction between humans and computers. The current basic human interaction with computers requires the use of keyboard, mouse, etc. Although, a hand gesture is not a replacement for these devices, addition of hand gesture with these devices provides a way to supplement the current interaction in a more natural way and intuitiveness. The improvements have been growing over the years with touch based interaction. The natural human-to-human interaction includes speech and expressions.

The main goal of this work is to make the human-computer interaction more natural and intuitive. The use of vision techniques do not require the use of color marker, hand gloves, or wearable devices. A simple use of the hand to create a gesture that can be recognized efficiently is a necessary criteria. With the rise of ubiquitous computing, the motivation to implement a hand gesture recognition system is given a boost. We implement a convolution neural network to recognize the hand gesture which is a state-of-the-art technique used in recognition.

Our contribution is the implementation of the temporally blended image frames which are inputs to a convolution neural network. We also contribute in the design of optimized and computationally efficient convolution neural network that provides an precise and accurate clasification and recognition. The major work is developing the entire system pipeline which can be used to control computers and robots.

1.2 Gesture Classification

Gestures can be divided based on the human anatomy, interpreted meaning, devices used, and methods used to create gestures. These can be classified as below.

Human Gestures Classification

Human gestures are related to head movement and facial expressions. A simple example would be nodding the head from the left to the right expressing refusal. Acknowledgement of an interruption with a nod of the head is a natural and intuitive communication gesture. Facial gestures involve eye movements, mouth expressions, eye lid, lip and nose movements. A smile is a very basic example used most frequently in daily activities. An example of an arm gesture would be conducting an orchestra which is defined as the art of directing the simultaneous performance of several players or singers using arm gestures by standing in front of them [9]. Hand gesture example would be raising a hand to greet someone. Another example of hand gesture would be to raise a specific number of fingers on either hand to indicate digits between zero and nine. Body gestures may involve one or more persons. Dance moves to describe a specific form of dance pattern is a good example of body gesture.

Temporal classification

Gestures are further conclusively divided into two types: Invariable (Static) and Variable (Dynamic). Static gesture is time independent whereas a dynamic gesture is dependent on time.

Invariable hand gesture involves posing a hand in a certain way or in a defined hand posture, whereas a variable hand gesture includes the entire hand trajectory motion and the movement. Our work considers a subset of both the hand gesture types.

Classification based on interpreted meaning

Emblems, illustrators, regulators, affect displays and adaptors [28, 18] are the typical classes to describe gestures. Emblems are defined as gestures that can be substituted for spoken words. An example is showing a thumbs up to show acceptance. Regulators are used in mass speeches as a means of interaction between active speakers and listeners. A perfect example would be raising hand to manage turn-taking to raise a question. Illustrators are used for illustrating spoken words. A way of providing directions by pointing to a specific location can be considered as an example of an illustrator. Af-

fect displays are expressions on a face combined with motions of the body to display the intensity of emotions. A good example would be when a person moves his or her body back during a sudden encounter with a snake is an instant and sudden display of the emotion of fear. Adaptors are gestures that are used in certain situations for personal satisfaction or comfort, which have converted to a habit. The act of shaking our legs while sitting idly or during a desk job is a classic example of adaptors.

Classification based on methods used

Two types of methods are used to record and detect gesture. The first method is to use a professional wearable electronic or electromagnetic devices like myo [27]. Myo armband reads the electrical activity of muscles and arm motion to control technology with hand gestures. Gloves like CyberGlove [19] are used to generate hand gestures. CyberGlove has 22-sensors which capture motion with up to 22 high-accuracy joint-angle measurements.

The other method utilizes computer vision and creates gestures using image processing techniques. This type is also known as non invasive as it does not require the use of any wearable devices. The former is expensive and only works in specific environment. On the other hand, with the evolution in the field of image processing and machine learning, the latter requires lesser hardware and is relatively less expensive and more robust.

1.3 Challenges in Hand Gesture Recognition

Recognition of hand gestures has various challenges associated with it. The following sections provide the details of each one of them.

Segmentation

Hand segmentation is a precursor for hand gesture recognition. It is a technique of image processing that involves separating the human hand or hands from the entire body in an image or a video. This plays a crucial role in the recognition of a hand gesture. A higher recognition rate is the direct result of better and properly-segmented regions of interest (hand area).

Dynamic Background

Dynamic background relates to movements in the background when a gesture action is being performed or posed in the foreground. The first case scenario is that it creates a lot of noise in the image which makes it difficult to segment the hand necessary for recognition. The second case scenario in a dynamic background is the presence of multiple persons in view. The issue with this scenario is the detection of multiple hands, which lead to an invalid output. Rick and John [21] proposed techniques to separate hand from a cluttered background in a gesture recognition. The techniques use target colors to generate histograms called color predicates. The performance described in the paper states that the system is flexible in different environments and it also performs well on cluttered crowd scenes involving muiltiple persons in view, extracting a smaller number of false regions.

Occlusion

The basic definition of occlusion is the limitation of a specific property of a device that causes deterrence in the output obtained. The first case scenario is when two or more fingers during a hand gesture appear to be connected due to noise or overlapping of shadows. The range of the camera also causes occlusion in the output. Another case of occlusion would be the tracking of hand gestures when there are other objects around causing the gesture to be partially covered. Koller et al. [22] addressed the problem of occlusion in object tracking. The authors employed a contour tracker based on intensity and motion. Occlusion was detected from the intersection of the depth ordered regions and then excluded. This helped in removal of occlusion on 3D object tracking.

Lighting Conditions

Lighting conditions like illumination and color brightness affect hand gestures. Also, low light conditions that affect output are a necessary consideration.

Noise

These contribute to the majority of errors in the output. Anything that causes the output or the input to go haywire is considered an error. This issue is caused in the first three modules of the proposed hand gesture recognition, and is resolved using image processing techniques like erosion and dilation. Removal of errors to 100% is practically unattainable. Inherent noise is impossible to be removed. These noises are related to hardware of the system. An attainable value of noise removal in any system would be to rely on the industry set standards and methods in noise reductions.

User-dependency

This is one of the most challenging issues. It is the ability to recognize the same gestures made by different people. Every person has distinctive ergonomic sizes of their hands. Therefore, chances are, that different data is generated for identical gestures. One solution would be calibration of the device used. The data augmentation (preprocessing steps) and the proposed convolution neural network handles this challenge in an agreeable manner.

1.4 Prior Work

Stern et al. [44] proposed a novel approach for design of hand gesture using both, human and technical design factors. Measuring human factors when designing gestures are represented in terms of intuitiveness and comfort. This paper focused on the development and collection of empirical matrices that performed quantitative analysis of the intuitiveness and comfort on hand gesture design. Three performance measures were defined intuitiveness, comfort and recognition accuracy. Measurement of intuitiveness and comfort used a bottom up approach, using a command to find matching gestures. This approach was followed by Coded Gesture Entry where the subject physically generates the gesture and enters the configuration information.

The subject selecting a level of belief for the command-gesture association followed each of method. Intuitiveness had eight commands and 59 distinct gesture set was generated. Based on popularity of use of gestures, the set was reduced to 22. Effort experiments were tested for the com-

fort factor. Several hard poses were added using the Borg scale of 0 to 10. Analysis showed that gestures are strongly associated with the command, an example was the "right command" with the gestures associated had a tilt or pointed to right side. The authors also observed a strong evidence of matching complementary gestures to complementary commands, which were defined as two commands with opposite connotations. Lower values were obtained in effort results showing the notion that the subjects inadvertently filtered out difficult gestures during intuitive experiments.

Wachs et al. [48] provided a review of vision based hand gesture applications. These application are divided into four main classes-medical systems and assistive technologies, crisis management and disaster relief, entertainment and human-robot interaction each illustrated through a set of examples. The authors discussed the three main advantages of these applications. Sterility is a necessity in medical and health care environments. Access to information maintaining sterility in medical environment is possible with these applications. Another advantage is the control of home appliances for impaired mobility and elderly users. The third and final advantage is exploration of large and complex data using a 3D interaction rather than 2D methods for intuitiveness benefit. The basic requirements with respect to benefits were also discussed. The first requirement is price, it is an important factor when the development budget is listed. Responsiveness of a system is necessary in gaming and entertainment applications.

User adaptability and feedback relates to ability to recognize gestures and the application it is used in. Learnability is based on how easy the hand gesture patterns are remembered or learned. Accuracy plays a role when the result is of prime importance, this is necessary in medical application like surgery procedure where human life is dependent on the application's accuracy. Intuitiveness provides a strong cognitive association between the commands and gestures performed. Comfort avoids gestures that have intense muscle tension over long periods referred to as "Gorilla arm". Lexicon size should be efficient enough as to allow robust classification between the gestures. "Come as you are" is a fundamental requirement in vision based application. This requirement suggests a non-invasive approach that avoids use of wearing additional aids or wired devices like gloves, markers that help in recognition.

Interaction space defines the virtual interaction envelope and suggests methods for calibration

when using multimodal cameras. Gesture spotting consists of distinction between useful and unintentional movements that appear as gestures relative to the immersion syndrome [4]. The paper described the four recommended guidelines for widespread commercial and social acceptance of the application. Validation using public and standard test sets like sensitivity/recall increase the robustness of the systems. User independence to promote customizability improves acceptability. Usability criteria to evaluate learning and ease of remembering gesture lexicons and likelihood of errors performance using subjective workload assessment is necessary for social acceptance. Qualitative and quantitative assessment provides way of testing the system using alternative modalities helps improving the robustness to use compared to state-of-the-art systems.

Murthy and Jadon [33] provided a survey on vision-based hand gestures recognition research papers. The main purpose of this survey was to introduce the field of gesture recognition as a mechanism for human machine interaction. The survey describes the application domains for gesture recognition such as Virtual Reality, Robotics and telepresence, desktops and tablet PC applications, games and sign language. Virtual Reality interactions use vision to enable realistic execution of virtual objects for 3D displays. Robotics domains use gesture to interact and control robots. Gestures provide an alternate interaction desktops and tablet PCs. Interactive games use motions and gestures to control movements of avatars in gaming domains. Sign languages are highly structural and therefore well suited for vision algorithms. The paper also gives requirements such as robustness, computational efficiency, user's tolerance and scalability required to make a successful working system.

The authors mentioned three approaches, Model based (kinematic model), view based, and low-level features in which hand features can be derived to classify it as a gesture. Hand gesture classification described are rule based and Machine learning approaches. The paper also presents the paper related to gesture taxonomies which vary from author to author. The taxonomy best appropriate according to this paper was developed by Quek [39, 40]. The paper presents problems hampering the solution to use of gesture recognition in commercial level. Most of the papers reviewed relied on several assumptions suited for controlled lab conditions and prevent the effect of generalization to arbitrary configuration. Assumptions such as high contrast stationary backgrounds and ambient lighting conditions. Recognition results presented in the surveyed papers are based on the author's own

collected data therefore, raising questions over its applicability in general real-world data. A solution proposed was to create a standard database for comparison techniques in analogy to FERET database [38].

There is a lot of research focused on hand gesture recognition. This research is divided into device based and computer vision based Hand Gesture Recognition. The former uses special hardware worn on the hand like data gloves, colored gloves or markers. The latter uses human body features like skin color and does not require any hardware to be worn.

Glove based detection

Glove based detection use gloves as physical devices to obtain the output. The gestures are produced by using sensors on the gloves.

Pawel et al. [35] worked to analyze gestures of the hand and body language using data from a specialized glove. The authors used data from ten sensors present on the glove. This data acts as an input to a machine learning algorithm. The results showed unique features of the classifiers achieving a sensitivity of 98.32%.

Devi and Deb [11] proposed a glove-based system connected to a mobile device to translate Hindi sign language to speech. The glove is build with flex sensors which gather the gesture information and an Arduino micro-controller to provide bluetooth communication with a mobile device. Eculidean distance is used to classify the gesture.

A proposed hand gesture recognition system implemented a charge-transfer capacitive touch sensor for translating gestures into American Sign Language [1]. The prototype thus developed is a set of capacitive touch sensors and Rasberry Pi single processor unit to recognize and translate hand gestures into sound. The results obtained through 1080 trials produced an accuracy of 92%.

Parvini et al. [34] provided a comparison of the major approaches for recognizing hand gestures. The advantages for each of the approaches were discussed. The single layer feed forward, back propagation neural networks were compared with GRUBC (Gesture Recognition by utilizing Bio-mechanical characteristics) over American sign language. They concluded that the GRUBC provided higher accuracy in detecting similar gestures without them having to be labeled. This approach

required the use of physical equipment to address the problem description, thus having wiring fixtures and minimizing the use to a specific range of access.

Kau, Su, Yu and Wei [17] proposed a glove to recognize dynamic gestures present in Taiwanese sign language. A gyroscope detected the motion of the trajectory movement of the hand. The authors used flex sensors to track and detect finger flexion combined with a gyroscope sensor to detect palm orientation. The proposed glove is connected via bluetooth to a mobile device that displays the final results. The architecture is structured on data from the sensors and gyroscope captured in cache of registers that are monitored per clock cycle. A total of four registers are used. Posture is stored in one, next monitors and records orientation of the palm, the third stores the trajectory and the final register stores the duration of the posture. A flag is set to check if a gesture has been recognized or not. Sensor data size is set to 20-bits. The accuracy of the glove was 94% on five dynamic gestures. This work does not relate any research to machine learning.

Finger based detection

The use of gloves in the detection of hand gestures requires the use of physical equipment as a source whereas vision based detection does not require any external equipment and thus has a higher advantage.

A research proposed using finger segmentation for hand gesture recognition as a way to recognize the gestures over SVM and CRF classifiers [6]. In their work, the hand region is extracted from the background with background subtraction and then the finger and palm are segmented so as to detect and recognize fingers. The proposed classifiers predict the label of the gesture.The performance of this method is based on hand detection and background subtraction. The background tends to be dynamic in nature and any object with similarity to skin color will tend to degrade the result obtained.

A significant part of the research work in gesture recognition is based on finger position and finger segmentation. The process of finger segmentation and positioning takes up most of the processing time. Finger emphasized multi-scale description [51] implemented a descriptor that used multiple scales for discriminative and complete representation of hand shapes, thus making the finger features emphasized resulting in detection of the hand.

Static hand gesture algorithm implemented by Yu et al. [52] is based on finger angle characteristics . A gesture is detected by the angle of the fingers relative to the palm center. The work concluded that true gesture recognition is obtained using the angle from the fingertip attachment to the center of the palm by calculating the size and quantity of the finger angles. The accuracy rate obtained was 96.8% from the 900 gesture images used.

Hand gesture recognition system to count the number of fingers using geometry of the hand was performed by D.K. Vishwakarma et al. [25]. The segmentation of hand region uses skin color likelihood method to extract skin color. Morphological and geometry functions are used to extract fingers and then the fingers are counted using rule based classification.

A lot of previous related work based on vision [31, 13, 33] for hand gesture recognition reviewed a lot to proposed methods. Survey and reviews were conducted about the use of gesture in real-world applications and environments. The factors effecting recognition were the nature of optical sensing and the quality, dynamic and variable backgrounds, lighting factors and color schemes. Overall detection and tracking of hand performance was lowered with effects of these factors.

Hand gesture recognition using Kinect

The development of inexpensive, reliable and robust Microsoft Kinect depth sensor has laid the path to improving the overall opportunities of hand gesture recognition. The absence of physical equipment in Kinect depth sensor is successfully used in body tracking [2], moving object detection [37] and face recongition [16]. In hand recognition there are few techniques that are successful but require large processing. These can be optimized further to increase the overall accuracy and robustness.

Liu, Zhou and Li [26] proposed a sign language recognition system with Long Term Short Memory (LSTM [15]) using the skeleton joint (left elbow, right elbow, right hand and left hand) trajectories provided by Kinect sensor camera (color image and depth information are discarded). The architecture consists of seven layers including the input layer. The first layer is the input layer which is fed with a 12-dimensional feature vector comprised of four 3D spatial point vectors which are the joints in the body. The next layer is the LSTM layer with 512 filters followed by a fully connected layer of 512 neurons and another consecutively of 100 neurons corresponding to 100 classes. The authors

used 100 isolated sign language words from the Chinese sign language and another with 500 sign words. The first dataset consisted of 5 performers that generated each gesture 5 times, thereby creating $25,000$ images. The second dataset had 50 performers each performing the gesture 5 times resulting in $125,000$ images. The accuracy of the system was 86% using the first dataset and 64% with the second dataset.

Sung, Ponce et al. [45] proposed a hierarchical maximum entropy Markov model (MEMM) that contemplated a subject's activity, collected as a set of sub-activites and concluded a two layered graph structure using the dynamic programming approach. The corpus of the dataset consisted of twelve different activity sets that are performed by four people in distinct environments-kitchen, living room, office, et cetera. The proposed model achieved an accuracy of 84.3% when the same person was present in the training data and 64.2% when the person was not seen before by the model (unbiased data test).

Convolution Neural Network based work

Javier et al. [3] discussed convolution neural network architectures for hand gesture recognition. This work used two classes of gestures: open and closed hand along with the unknown class and implemented six architectures varying the hyper parameters and depth. The results obtained showed how robust the network was during implementation, depending on the changes in the hyper parameters generating the model with the best performance. The sixth architecture achieved an improvement of more than 40% compared to the first architecture. The performance obtained above is based on only two gestures and required more classes to be added to make the overall system robust.

Pei Xu [50] proposed a hand gesture recognition with convolution neural network using a cheap monocular camera . The developed system runs with a fixed number of frames per second. When an image is captured by the camera, the system uses a hand detector to filter out the hand image or terminates when nothing is detected. The filtered out image is passed to a CNN classifier that recognizes the processed image and a Kalman estimator is employed to estimate the position of the mouse cursor based on points tracked by the hand detector. The recognition and estimation results are submitted to a control center which is a simple probabilistic model to decide the response the system

should make.

Recently, classification of hand gestures has been successful on the VIVA challenge dataset. Pavlo et al. [32] designed a hand gesture recognition system using 3D convolution neural network. The classifier used fused motion volume of normalized depth and image gradient values that improvised the spatial-temporal data augmentation to avoid over-fitting of the model. The model also used a combination of low and high resolution sub-networks to improve the classification accuracy. This system achieved an accuracy of 77.5%.

CHAPTER II: SYSTEM DESIGN

The system design has three modules namely frame preprocessing, convolution neural network model design (CNN) and validation of the CNN model. Figure 2.1 provides an overview of the system.

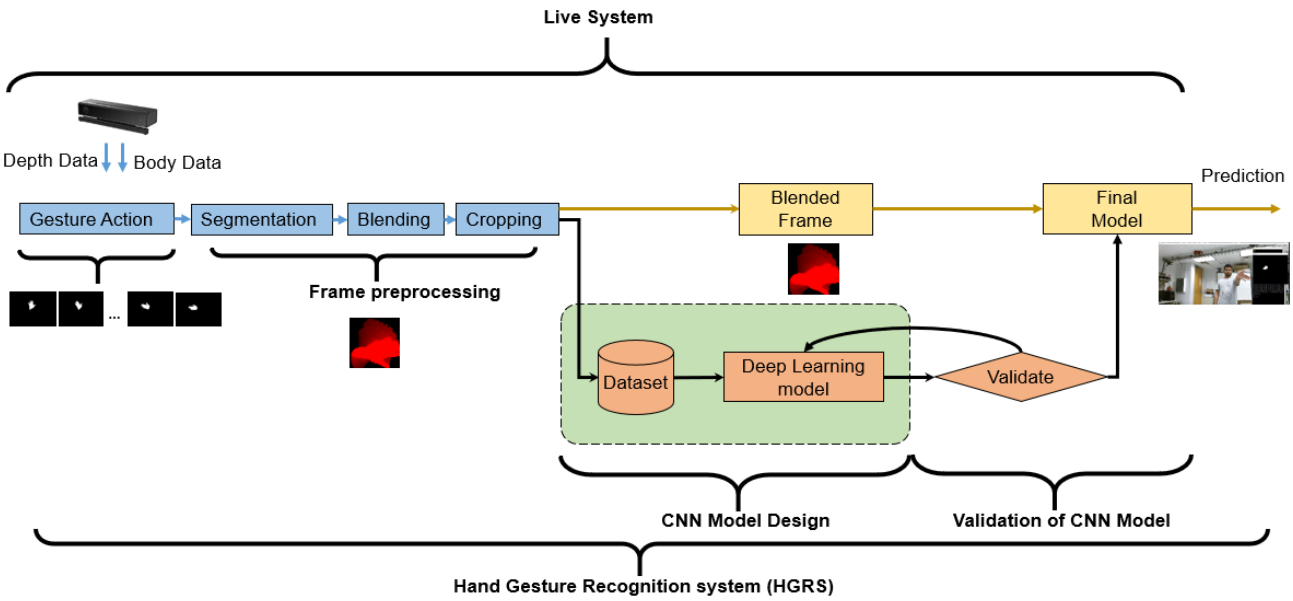


Figure 2.1: System Design work flow.

2.1 Frame Preprocessing

In this module, we perform image frame preprocessing using image processing techniques. Frame preprocessing is precursor for hand gesture recognition. This module uses three main techniques to perfrom frame preprocessing namely segmentation, blending and cropping. Each of these three techniques perform necessary steps in achieving the required frame for recognizing the hand gesture.

Segmentation

Segmentation is the separation of an area of interest from the rest of the environment. The area of interest, that is, the hand region up to the wrist needs to be split from the rest of the human body and the background environment.

This technique uses raw depth data and skeleton joints obtained from Microsoft Kinect [30]. Kinect allows tracking of 26 body joints. The right hand region has three joints defined as hand tip, hand center and wrist joint. Similarly, for the left hand.

Hand segmentation is performed by fetching the hand region joint types provided by Kinect, that is capable of tracking up to six bodies. We restrict the tracking to a single body that is tracked first when it comes into the view of the camera. Kinect integrates two different sensors, specifically a color with a resolution of 1920×1080, an infrared (IR) sensor with a resolution of 512×424. These sensors have different resolution and are not synchronized or aligned so as to map one on the other.

The projection of skeletal joints on depth data frame requires proper adjustment and mapping of the coordinates. Body tracking is performed by the depth sensor, so the coordinates of the axes are aligned with the depth data frame. Projection of the coordinates from the depth data frame onto color data results in mis-alignment. The main task of this step is to map the skeletal joints on the tracked depth data frame. We use the mapping functionality to map the joints on to the depth data frame, that is, finding the point in the depth data frame that matches the real world coordinates of the joints. This mapper identifies points from 3D space that corresponds to a coordinate in the depth 2D space and vice versa. The coordinates are 3D point values measured in meters. The dimensions of the visual elements are measured in pixels, so conversion of real-world 3D values into 2D screen pixels uses the depth coordinate points that stores the mapped value, thus, mapping 3D points on to 2D points. The final result obtained on this step sets the skeletal points of the hand region on the depth data frame.

Mapping is defined as setting the skeletal joints on the depth frame. The background and the region of interest (hand area) has to be segmented in the depth data frame. A point to be mentioned is when we discuss the term “frame”, we mean each individual frame obtained from the Kinect sensors. The above mapping is the first part of the segmentation. The previous paragraph described setting the body tracking to the first person that comes into the view. In effect, setting the tracking to the first person involved, partially solves the challenge of background removal, that is, when there are multiple persons in the field of view of the sensor. Figure 2.2 displays when a person is moving in the background whereas figures 2.3 and 2.4 show when a person is raising hands. The other part of the challenge is removal of the surrounding environment from the depth frame providing only the region

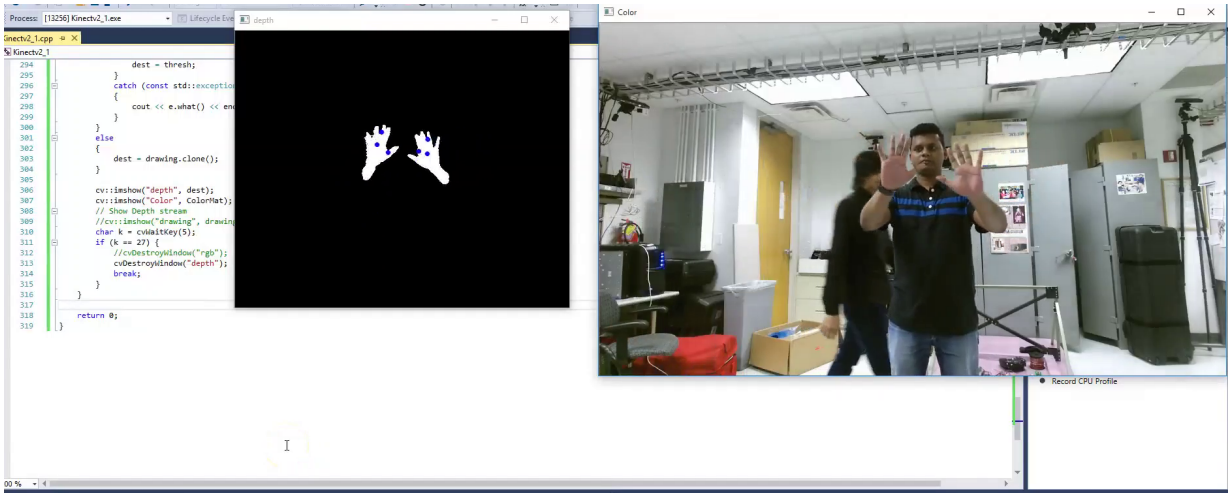


Figure 2.2: Dynamic background removal (human present).

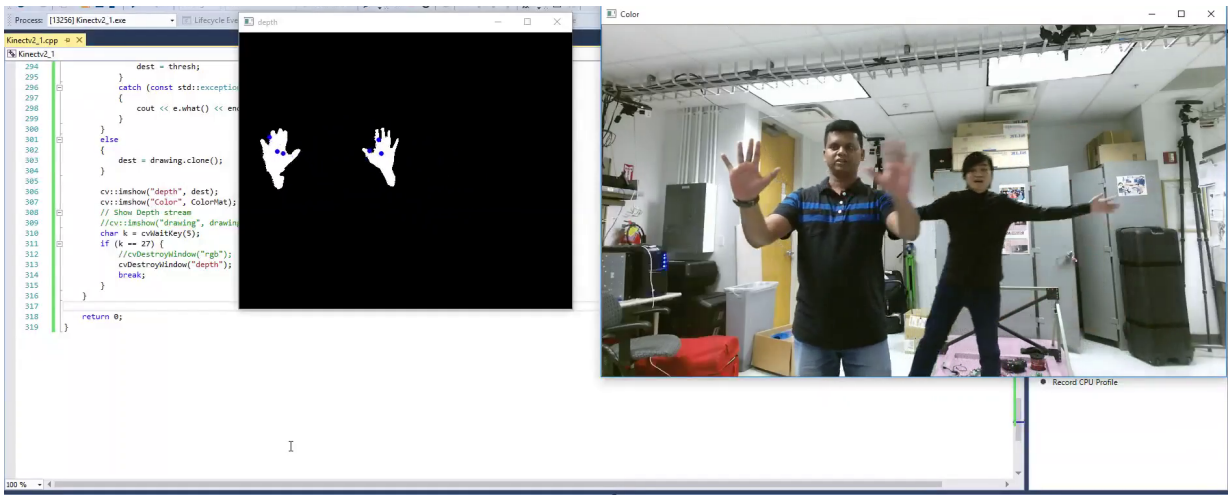


Figure 2.3: Dynamic background removal (human hand raised).

of interest (hand area).

The other part of the issue is solved by mapping another joint from the skeletal data. The right elbow joint is mapped using the above procedure. The use of this joint is to set a threshold in the depth frame. The distance (depth) of the depth data frame beyond this elbow joint depth value is set to null. The depth distance between the hand center and the right elbow joint is fixed. The difference allows us to check if the arm is bent, removing the hand out of depth frame completely. This differential value is necessary as we require only the hand region and not the forearm region.

The last part of the segmentation involves setting all the depth value pixels of the hand region to one specific value (white) and the remainder to a different value (black). This is analogous to binary

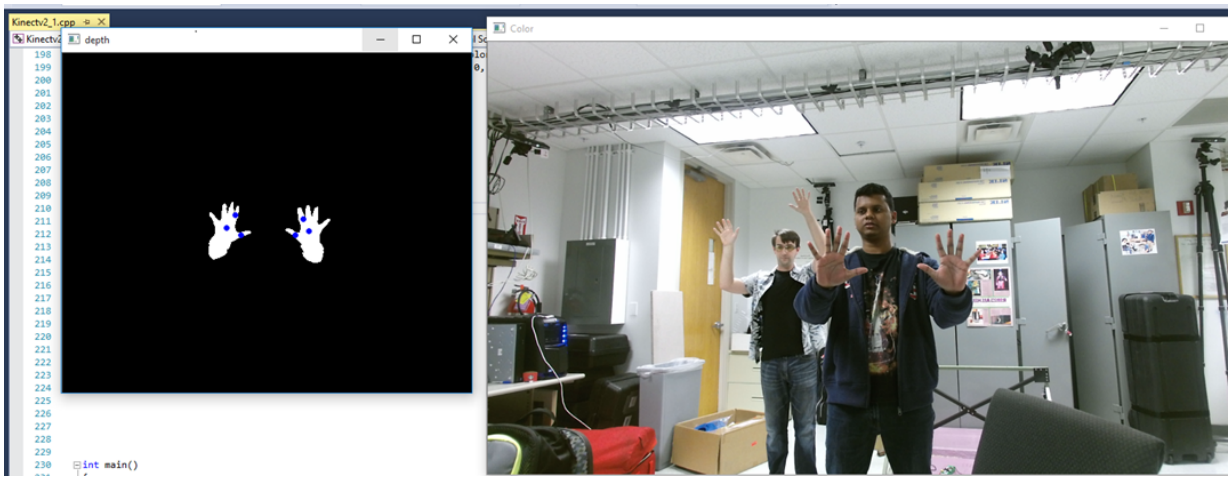


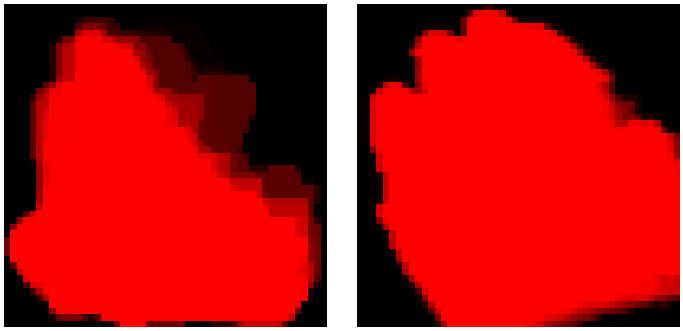
Figure 2.4: Dynamic background removal (Human hand raised).

frame which is a two-valued frame. The use of flood fill functionality fulfills the criteria. The step involves setting a seed. The seed uses connected components that spread to the entire hand region, providing the region of interest (hand area). The seed is the hand center joint mapped on the depth frame. This seed is then filled with a white value thereby generating white colored hand area. This is the segmented depth data frame which is a single channel 8-bit frame.

Since each gesture action is performed for a duration of three seconds, we obtain $85-90$ frames from the depth sensor.Theoretically, the number of frames is fixed to 90 frames with a frame rate of 30 frames per second. The range of $85-90$ frames indicates loss of negligible time due to computation in segmentation module. The conclusion is the frame rate is decreased by an insignificant value. Computation in segmentation module is not related to loss of frames during the gesture. The decrease in frame rate, results in less frames generated. Each frame fetched from the depth data stream, is passed into a frame buffer. Temporal features of a video are based on frames generated for each unit of time, each frame is individually fetched. This helps in keeping the temporal features of the frames intact. Consecutive frames in the frame buffer are temporally distinct. Depth data frames are passed to the frame buffer in original resolution of 512×424. Therefore, spatial features are not affected. At the end of the time limit, the entire buffer is passed to the blending module.

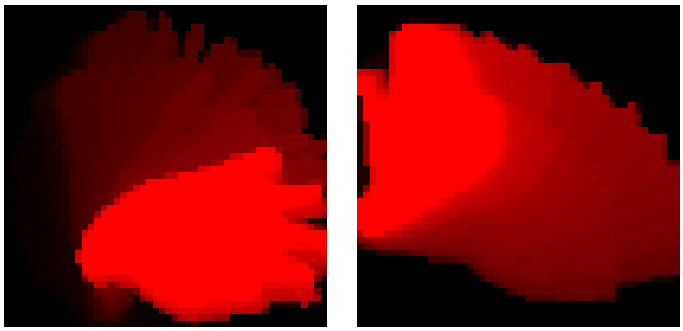
Blending

The blending technique performs blending of frames keeping the resolution of each frame in its original state. This is a necessary factor as temporal and spatial features are important in the construction of the frame. The blending of frames is defined as summing up the frames by increasing the intensity of consecutive frames into a single blended frame. Blending of frames without any change to pixel properties would result in an indistinguishable frame. An example to show distinguishable frames is to use swipe left and swipe left reverse gestures described in section 2.2.1. Figure 2.5 show the indistinguishable frames for each gesture shown in figure 2.6.



(a) (b)

Figure 2.5: Indistinguishable Frames: (a) 1 (b) 2.



(c) (d)

Figure 2.6: Gestures: (a) Swipe left (b) Swipe left Reverse.

We use the intensity of each frame in the frame buffer to distinguish between gestures having the same rotation axis as in the example above. Each frame intensity is increased sequentially. The

final frame has the maximum intensity or the original intensity. Equation 2.1 describes the process of intensity change for every i^{th} frame in N number of frames. In Equation 2.1, N is the number of frames in the frame buffer and i is the count of each frame in the frame buffer. I_i is intensity of the i^{th} frame and I_{r_i} is the resulting intensity of i^{th} frame.

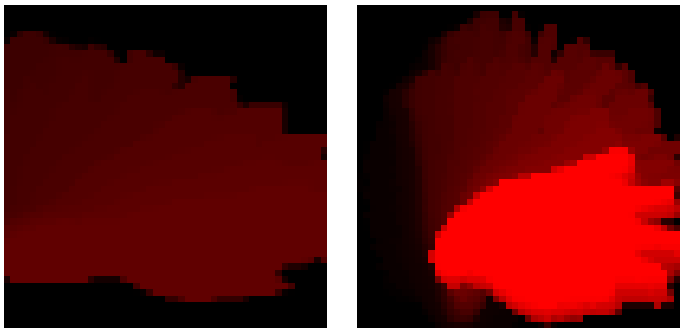
$$I_{r_i} = I_i \times \frac{i}{N} \tag{2.1}$$

An additional property used to provide distinctiveness for each frame is weight adding functionality [5]. Frame buffer stores 8-bit single channel frames, to use the weight adding functionality, it is necessary to convert each to 24-bit 3-channel frame. Equations 2.2 and 2.3 show the summing of frames [5]. First frame F_i and second frame F_{i+1} in the frame buffer is added by multiplying fixed scalar weights and stored in an interim frame $F_{interim}$. This interim frame $F_{interim}$ acts as the first frame with the sequential or next frame in the frame buffer. The final frame is the resulting blended frame. The scalar weights α, β and γ are set to 0.95, 1 and 0 . These values are not based on any specific rules and fixed by experimentation. Since intensity, is reduced for each preceding frame, it provides a perfect way to add up the two frames.

$$F_{interim} = \alpha F_i + \beta F_{i+1} + \gamma \quad when \quad i = 0 \tag{2.2}$$

$$F_{interim} = \alpha F_{interim} * + \beta F_{i+1} + \gamma \quad when \quad i = 1 \quad to \quad N-1 \tag{2.3}$$

Figure 2.7 shows the intensity of a single blended frame with and without adding weights to the frame. Each blended frame is further passed to the cropping module.



(a) (b)

Figure 2.7: Frames weights: (a) with weights (b) without weights.

Cropping

The cropping technique uses the region of interest to crop out the blended area in a frame. The first substep is to remove the blended frames that have no gesture present. No gesture implies that the blended frame is zero-valued pixels or empty frame having only black pixels. Frames with few pixels having non zero values are deleted as well. Figures 2.8 and 2.9 display the above cases.

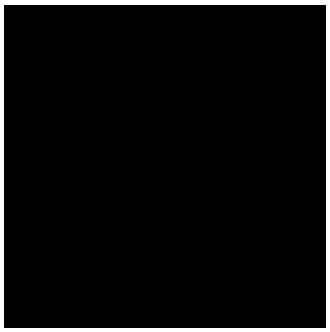


Figure 2.8: Zero pixel frame.

(a) (b)

Figure 2.9: Non zero pixel frames: (a) 10% (b) 14%.

This technique prevents faulty blended frame from being passed on the proposed convolution neural network. The second sub-step in the cropping module removes the blended frame if the skeletal tracking of the Kinect fails and the subject's entire body is displayed in the blended frame. Kinect failures occur due to faults in the hardware systems. A non zero count is set to check if the hand covers more than 30% of the actual frame area. Hand gesture should only cover a range of $18-30\%$

of actual frame area. This range is set based on the experimentation with various distances from the Kinect. These experiments were performed by sitting on chair which causes the detection of joints to go haywire and also by hidding the hand joints with an object or behind the back to register test faults in the system, thereby having entire subject's body in the frame. Figure 2.10 display the above case scenarios. Frame 1 and 2 in the figure 2.10 use all the 3-channels of frame for experimentation purpose. As a step of simple optimization, we use only a single channel of the 24-bit 3-channel frame. We use the red channel, setting the remaining channel intensity to zero.

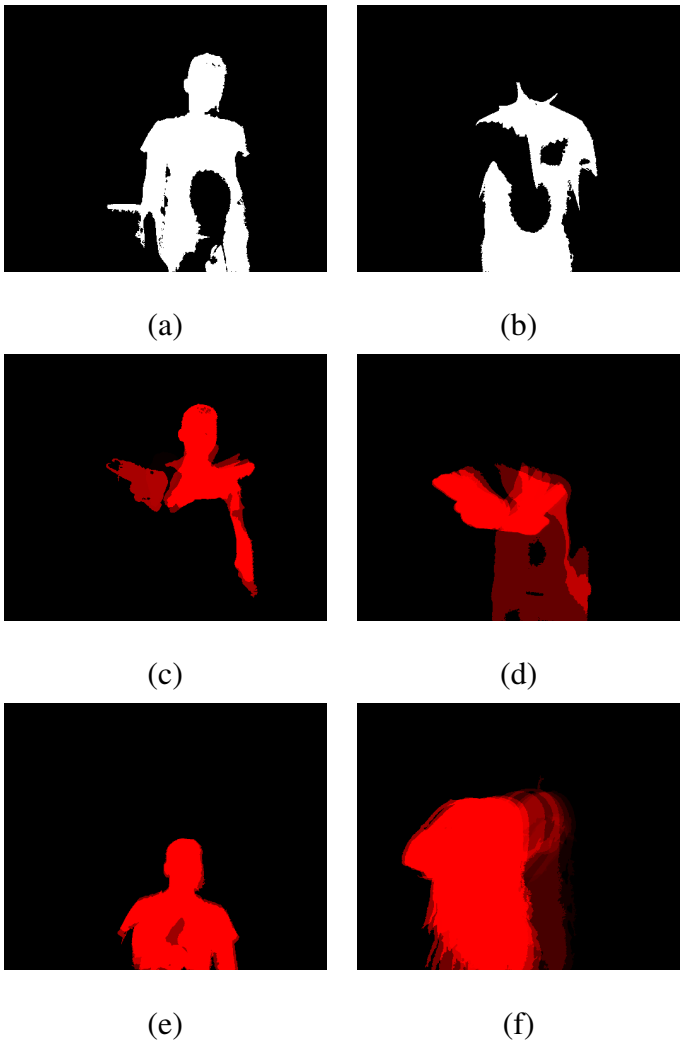


Figure 2.10: Kinect failures frames: (a) 1 (b) 2 (c) 3 (d) 4 (e) 5 (f) 6.

The third substep in the cropping module is to find the region of interest (gesture area) in the blended frame to perform cropping. We use the region of interest (ROI) functionality [5] to perform

cropping. We get all the non zero points from the frame into a vector buffer. Using the minimum area of rectangle function gives us the bounding box on the ROI. We use this bounding box to crop out the blended frame. Figure 2.11 provide an example of cropping performed. The final substep in the cropping module is to resize the frame to an empirically evaluated size of 50×50.

(a) (b)

Figure 2.11: Cropping: (a) Cropped (b) Original.

2.2 Convolution Neural Network (CNN) Model Design

The Convolution Neural Network (CNN) Model Design module has two techniques named as dataset and deep learning model. Dataset technique defines the way the dataset was created and number of gesture classes used in the system. Deep learning model techniques displays the steps in achieving the optimized CNN model.

Dataset

Dataset technique performs following three steps, gesture description, dataset creation, and dataset preprocessing.

Gesture Description

We have defined eight gesture classes of which four are static and four are dynamic. We consider a dynamic gesture as one that involves the movement of the entire hand along a specific trajectory motion while static gestures involve the number of fingers being raised. Each gesture is performed

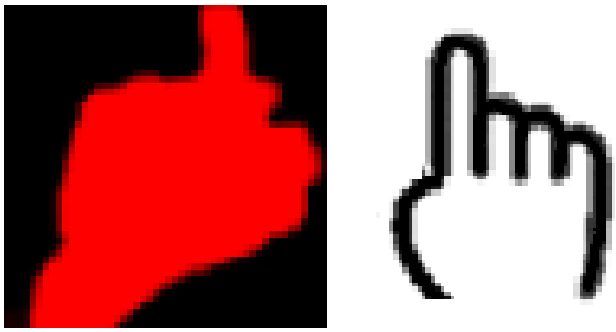
within three seconds. The arm should be positioned parallel to the line of sight or field of view of the depth sensor when performing the hand gesture action. This position is important, since we have set the elbow joint to be threshold limit for background removal and skipping this position would cut off the threshold resulting in faults from the segmentation module.

The first class in the gesture set is called the Open Palm gesture. This gesture is performed with all the five fingers raised and spread out, facing the depth sensor. Figure 2.12 display the blended frame and gives the gesture action or posture.

(a) (b)

Figure 2.12: Open Palm: (a) Blended (b) Action.

The second gesture class is defined as the Single Finger gesture. This gesture is performed by raising the single finger from the palm side facing the depth sensor with the prominent use is the index finger Figure 2.13 is an example of the blended frame and the posture of the gesture.



(a) (b)

Figure 2.13: Single Finger: (a) Blended (b) Action.

The third gesture class is the Two Fingers gesture. The gesture involves raising two fingers keeping the rest of the fingers closed into the palm, with the palm facing the depth sensor. The prominent fingers used are the index and the middle finger. Figure 2.14 display the blended frame and the posture of the hand gesture.

(a) (b)

Figure 2.14: Two Finger: (a) Blended (b) Action.

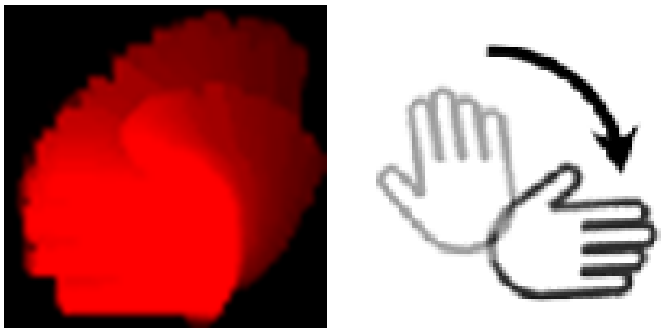
The fourth class is defined as the Three Fingers gesture. The gesture involves raising three fingers facing the depth sensor. The prominent fingers are the index, the middle and the ring finger. Figure 2.15 show the blended frame and gives an idea of the hand gesture posture.

(a) (b)

Figure 2.15: Three Finger: (a) Blended (b) Action.

The remaining four gesture classes are dynamic in nature that is, it involves movement of the hand along the wrist axis. The fifth class in the gesture set is called the Swipe Right gesture. The gesture involves an open palm facing the depth sensor with the fingers tightly fixed to one another and pointing up which is the start state. The action of the gesture is to rotate right, up to the point of making an

approximate angle of 90°(with the original start state) along the wrist rotation axis. Figure 2.16 provides an example of blended frame and defines the gesture action.



(a) (b)

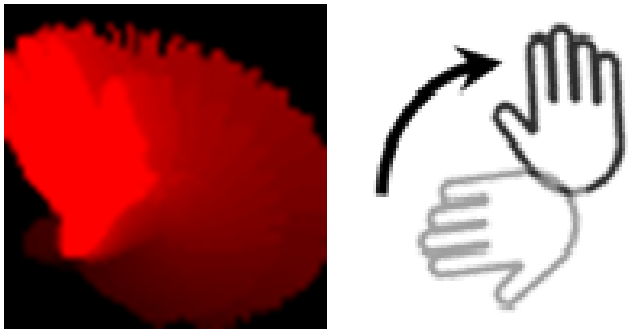
Figure 2.16: Swipe Right: (a) Blended (b) Action.

The sixth class in the gesture set is called the Swipe Left gesture. The gesture involves an open palm facing the depth sensor with the fingers tightly fixed to one another and pointing up which is the start state. The action of the gesture is to rotate left till it is making an an approximate angle of 90°(with the start state) along the wrist rotation axis. Figure 2.17 highlights an example of blended frame and the hand gesture action. The remaining gestures are the reverse of the above two hand gestures. We differentiate the temporal features and the below defined gestures will provide the proof of concept of this work. Temporal feature separation is a way to provide evidence that another gesture performed in reverse action of the original gesture does not generalize to the original gesture in the model.

(a) (b)

Figure 2.17: Swipe Left: (a) Blended (b) Action.

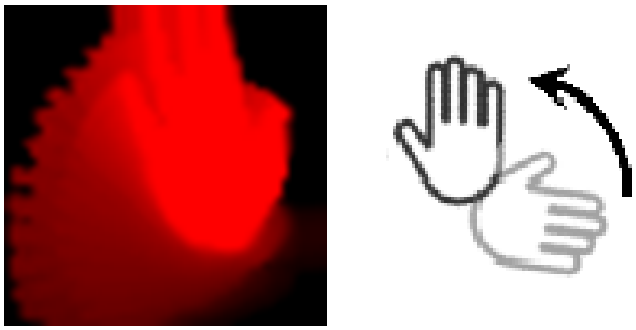
The next class in the gesture set is called the Swipe Left Reverse gesture. The gesture is the reverse of Swipe Left gesture defined above. The gesture is defined by an open palm facing the depth sensor with no gaps between the fingers.The start state has the fingers pointed out towards the left and rotating left making and angle of 90°(with the original start state) along the wrist rotation axis. Figure 2.18 highlights an example of this blended hand gesture frame and the gesture action.



(a) (b)

Figure 2.18: Swipe Left Reverse: (a) Blended (b) Action.

The final class in the gesture set is called the Swipe Right Reverse gesture. The gesture is the reverse of the defined Swipe Right gesture. The gesture is defined by an open palm facing the depth sensor with the fingers stacked side by side .The start state has the fingers pointed out towards the right and rotating right making and angle of 90°(with the original start state) along the wrist rotation axis. Figure 2.19 provide a visual example of this blended hand gesture frame and gesture action.



(a) (b)

Figure 2.19: Swipe Right Reverse: (a) Blended (b) Action.

Data Set Creation

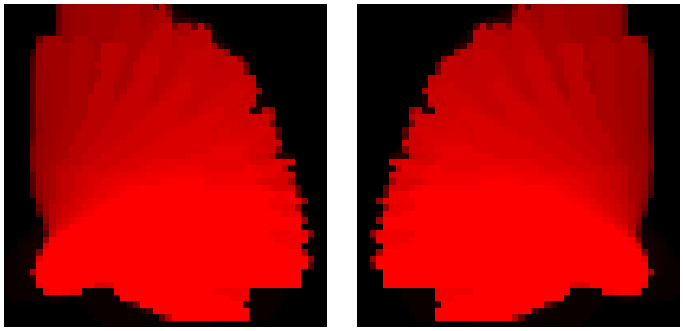
We create and use our own dataset to train the proposed Convolution Neural Network. The dataset consists of eight classes as described in the above section 2.2. Each class consists of a total of 1600 original images generated through the techniques mentioned in section 2.1. This is not the complete dataset as we require more data to train the proposed network. Section 2.2 provides the complete dataset. Therefore, based on these techniques, each image in the dataset is a blended frame of the hand gesture action (dynamic) or posture (static).

Dataset Preprocessing

This step describes the overall techniques used in preprocessing the dataset. There are eight classes each having 1600 blended frames. In practice, a good amount of data is required for training a model to be sure of its ability to generalize. The complexity of a neural network can be expressed through a number of parameters. Parameters of a neural network are defined as weights of the connections in a network. A general rule of thumb for generating dataset would be P^2 where P is the number of parameters used in the design of the CNN model [43]. Creation of more data is necessary for the model to generalize. We use image data generator functionality [7] which augments the blended frames. A number of random transformation performed generates frames that never have the same features. This functionality allows to configure random transformation and normalization operations to be performed on frame data before training.

Before the model is trained and tested, all the transformation must be accounted for in the blended frame. The types of transformation are reflection, rotation, dilation, translation and shear.

Reflection of a frame is not performed. Figure 2.20 display original and reflection (matches Swipe Right gesture present in the gesture classes) of the same gesture.



(a) (b)

Figure 2.20: Swipe Left Gesture: (a) Original (b) Reflection.

The rotation range of a frame is set between 0°- 15°. This range is set that any rotation above this would result in inconsistency in generalizing the model. A frame beyond the range specified is shown in Figure 2.21.

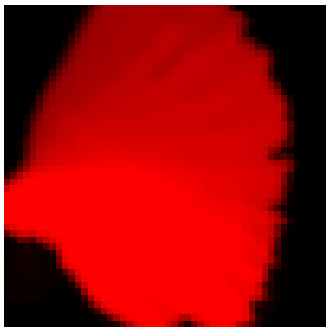
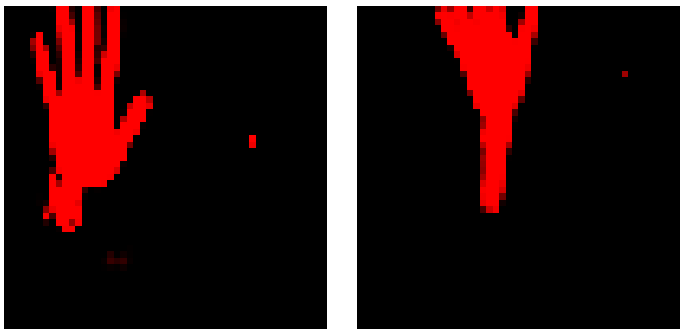


Figure 2.21: Frame rotated by 16°.

Dilation of frame is accounted by the cropping module described in section 2.2. Distance from the camera affects the frame in the original camera resolution. The cropping technique provides the region of interest and resizes the frame to an empirically evaluated size of 50×50.

Translation is set to 0.1 in width and height shift. This randomly translates frames vertically and horizontally. The value is set based on the experiments peformed and if increased, results in frames created as shown in figure 2.22. The figures show the gesture being cut off as a result of translation.



(a) (b)

Figure 2.22: Translation: (a) 0.2 (b) 0.5.

Shear is set to 0.1. This is also set based on experimentation. Any value outside the specified range results in the generation of images vertical to the human viewpoint. This effects the generalization that we can perform using model.

Setting these above parameters, we randomly selected 100 frames from each class. We created 4 frames for each blended frame generating a total of 3200 frames, that is, 400 for each class. The complete dataset had a total of 2000 $(400 + 1600$ original frames$)$ frames for each class. We did not select all the generated frames since the proposed model will not generalize. Training, would be an over fitting of the model. The reason is that for each frame, transformation is applied to the frame (original frame) but the pixel values of each generated frame remain the same. When features are created by the model, they will have a tight bound to the training data and the result would be over fitting. All factors of transformations and distance were taken into consideration. Using a $60 : 40$ train-to-test ratio, we separated the frames into train and test sets. This resulted in the division of 1200 training and 800 testing frames for each class. Thereby, the total number of frames was 16000 in the data set.

Deep Learning Model

Convolution Neural Networks (ConvNets or CNN) are a category of neural networks that are effective in computer vision areas such as image recognition and object classification. CNNs have recently proven to be very successful at image recognition [53, 23, 8]. CNNs are very useful in identifying

objects, faces and self-driving cars, etc. In a general design of a CNN model, three main operations (layers) are performed to detect and classify an object or an image. These three operations are convolution, pooling and fully connected layers. These are the fundamental basic building blocks for every convolution neural network. Each of these are stacked to form a full CNN architecture. Addition of layers to create appropriate architecture of CNNs require to focus on certain measures. These measures are number of efficient Convolution layers needed, the optimal number of hidden units, best pooling strategy, and the best input feature type for CNNs. The behviour of the neural network features extracted from the CNNs are also neccessary in the architecture design [42]. The following is the overview of our network.

$$INPUT - [[CONV - RELU - POOL] \times 3 - [FC - RELU] - FC]$$

INPUT stores the raw pixel values of the frames having a width of 50, a height of 50, and with 3 color channels.

CONV computes outputs of neurons that are connected to local regions in the input, each computing a dot product between their weights. This results in volumes as $[44 \times 44 \times 64]$ from the first Convolution layer generating a total trainable parameters of 3200. Similarly, the second layer results in volumes of $[18 \times 18 \times 64]$ with $102,464$ trainable parameters and final layer $[7 \times 7 \times 128]$ with $73,856$ trainable parameters.

RELU applies an element-wise activation function, max(0, x) thresholding at zero. It follows each of the convolution layer and the size of the output volume remains unchanged.

POOL performs down-sampling operation along the spatial dimensions (width, height), but the depth (feature sizes) remains fixed. It uses the max function to fetch the maximum value each of 2×2 with a stride of two. The resulting output volume is spatial reduced to half the dimensions but depth remains unchanged $[22 \times 22 \times 64]$, $[9 \times 9 \times 64]$ and $[3 \times 3 \times 128]$ are the resulting output for max pooling layers of our network. Each *CONV* is followed by max pooling layer.

FC (Fully Connected) layer computes the class score, resulting in volumes of sizes $[1 \times 1 \times 256]$ with an input of 1152 from a flatten layer [7] that generates a single dimensional vector, thereby generating $295,168$ trainable parameters and in the end of the network we use a fully connected layer of size$[1 \times 1 \times 8]$, generating 2056 trainable parameters where the output size is eight with each of the

eight numbers corresponding to a class score, as among the 8 categories. Similar to ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume obtained from pool.

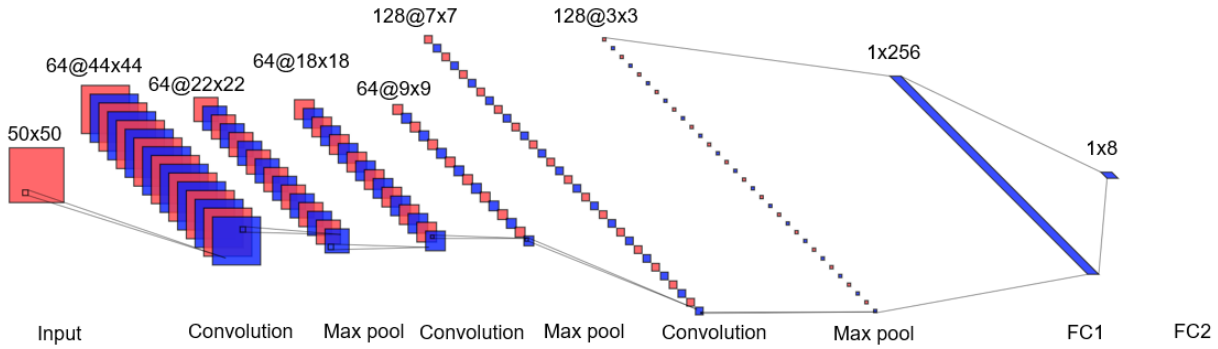


Figure 2.23: CNN Architecture Design.

Sequential model design with a linear stack of layers is used. Each layer is callable on at tensor and used to define a model. Our model consists of three convolution layers, max pooling layers, two dense (Fully connected) layer. A detailed description for each layer is provided below.

Convolution Layers

The primary purpose of a convolution layer is to obtain features from an input frame. The first convolution layer uses a set of 64 kernels or filters which are of size 7×7. Each of these filters scan over every pixel in the 50×50 frame taking a stride of 1. On each stride, a feature value is generated for that pixel. An intermediate output is computed as an element-wise multiplication of each pixel from the filter and the frame. These intermediate outputs are added up to get the final feature. This layer extracts higher level features of the input frame. The final frame obtained from the input frame is called convolved map. These filters act as trainable weights. The convolved map is also called a feature map because of the filters applied, and the activation functions of the neurons result in the extraction of features from the input frame.

Complex feature maps obtained from the first convolution layer are passed through a max pooling layer that reduces the spatial dimensions which then acts as input to the second convolution layer which uses a set of 64 filters of size 5×5. These filters perform the same operation as the above first

layer, but the detailed features are extracted from a complex feature map obtained from the first layer. The feature maps obtained from this layer are passed through a max pooling having a similar function as the above max pooling layer and then passed on to the final convolution layer.

The third and the final layer is composed of 128 filters of size 3×3. This extracts the intricate features from the feature map of the previous layer. The third layer passes the feature maps to the final max pooling layer.

Rectified Linear Units(ReLU)

Each of the above convolutional layer uses the ReLU activation function. This function computes max(0, x) that is a simple thresholding at zero. ReLu is used, since they accelerate the convergence of the gradient descent [23] (reduction of error rate) in comparison to other activation functions. Expensive computations like exponentials are not required and it involves only thresholding a matrix of activations at zero.

Max pooling Layer

Max pooling layers between successive Convolution layers or at the end of convolution layers is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in a network. These layers control overfitting. Layers of max pooling are used with filters of size 2×2 with a stride of 2, which removes $60 - 75\%$ of the feature maps. Therefore, every maximum operation performed produces 4 numbers.

In Figure 2.24, the input volume of size [$50 \times 50 \times 32$] is pooled with filter size 2, stride 2 into output volume of size [$25 \times 25 \times 32$]. An important conclusion is that the volume depth is preserved. Figure 2.25 is the most common downsampling operation called *max*, giving rise to max pooling, in this figure it is shown with a stride of 2. That is, each max is taken over 4 numbers (smaller 2×2 square is obtained).

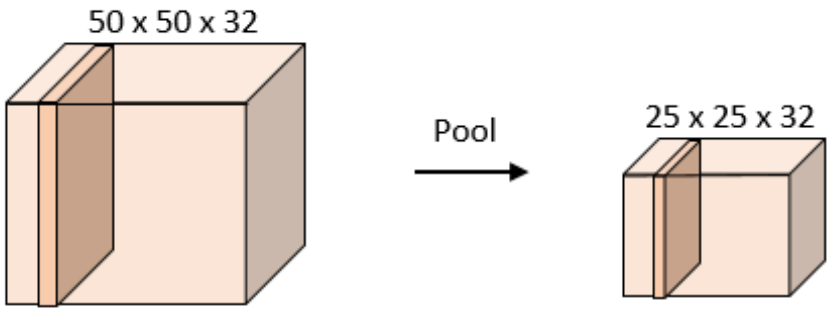


Figure 2.24: Pooling (showing depth preserved).

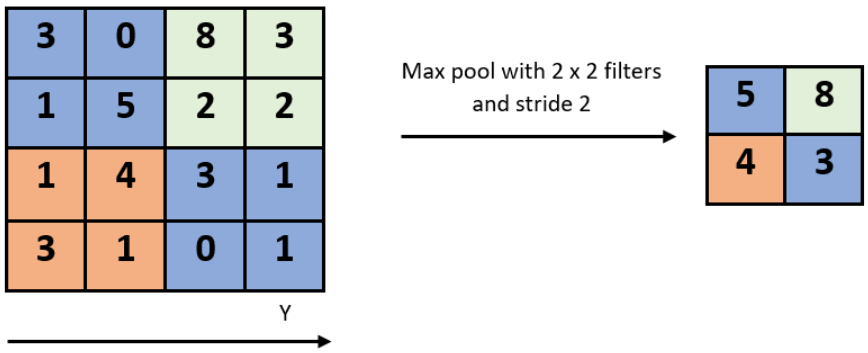
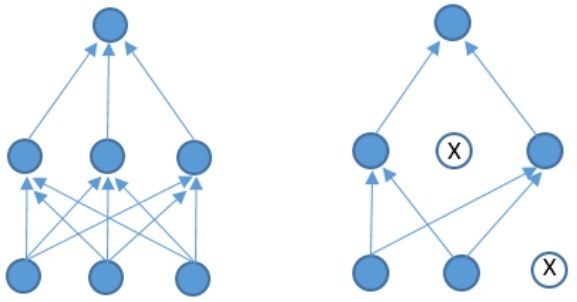


Figure 2.25: Classic max pooling example.

Dropout

Before we pass spatially down-sampled features to the fully-connected layer, we perform a dropout of 25% of the features after each max pooling layer to prevent over fitting. The dropout method is essentially a regularization technique to avoid the network from learning features fit only to training data. This effects generalization of the model and results failures in recognition during implementation on real world data. The role of the neurons is to approximate the features effectively from the training data. When the neurons try to approximate the features for the training data, they fit to a higher order approximation. This means that neurons will try to learn every details of the training data redundantly and in doing so, they over fit on training data. The dropout method in the Keras [7] API randomly mutes neurons creating a sparse set of features that reduces the possibility of over fitting. The features are then used to generalize on unseen data. Figure 2.26 provide an example of dropout.



(a) (b)

Figure 2.26: Dropout Layer: (a) Original (b) Dropout.

In our model design, a dropout of 25% is added after every max pool to reduce trainable parameters. Since most of the parameters are trained intrinsically on the training set, having a higher number of features learnt from the same data over again causes overfitting. Therefore, it is a necessary step to prevent overfitting of the model. Before we pass spatially down-sampled features to the fully-connected layer, we perform a dropout of 50% of the features to prevent over fitting which can be caused due to high activations generated from the neuron output.

Fully-Connected Layers

A Keras [7] API flatten layer unrolls the features or reshapes it into a 1-dimensional vector which can then be passed to a fully-connected layer. The first fully connected layer output is $[1 \times 1 \times 256]$. The Keras API provides a dense layer which is just a regular fully connected layer of neurons. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. The final fully connected layer uses the softmax activation that compresses the outputs of each unit from the previous layer to be between 0 and 1. The output of the softmax function is equivalent to a categorical probability distribution, it relays the probability that any of the classes are true.

Optimization

Adam optimizer is used to optimize the network weights and reduce the cost function to a minimum. The choice of optimization algorithm of the deep learning model acts like a thresholded boundary between good results in few minutes and days. Adam optimization [20] is an extension to stochastic gradient descent. It is adopted for deep learning applications in computer vision instead of the classical stochastic gradient descent procedure to update weights iteratively on training data. Adam is an acroynm for adaptive moment estimation. Since Adam is an adaptation of stochastic gradient descent, we briefly define the stochastic gradient descent (SGD), as an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization [41]. SGD maintains a single learning rate (alpha) for all the weight updates and learning rate does not change during training.

In Adam, the learning rate is maintained for each network weight and separately adapted as learning unfolds. Adam combines the advantages of different solutions for optimizing. They are from AdaGrad that monitors a per-parameter learning rate that improves performance on problems with sparse gradients [12] and RMSPROP [46] maintains per-parameter learning rates that are adapted based on the average of magnitude of the garadients for the weight. The Adam algorithm, instead of adpating the parameter learning rates based on the average first moment (mean value) as in RMSPROP, uses the average of second moments of the gradients that is the uncentered variance. The algorithm generates the exponential moving average of the gradient and squared gradient and the parameters of β_1 and β_2 control the negative rates of the averages.

Loss Function

Cross entropy loss [10] or log loss is defined as the measurement of the performance of a classification model, whose output is a probability value between 0 and 1 Cross entropy increases as the prediction probablity diverges from the actual label. An example would be predicition of a probability of 0.09 when the actual ground truth is one is bad and results in high loss value. Figure 2.27 shows a graph of possible loss values given a actual observation. As the prediciton approaches one, log loss slowly

decreases. When the predicted probabilty decreases, loss is exponential. The probabilty of each class

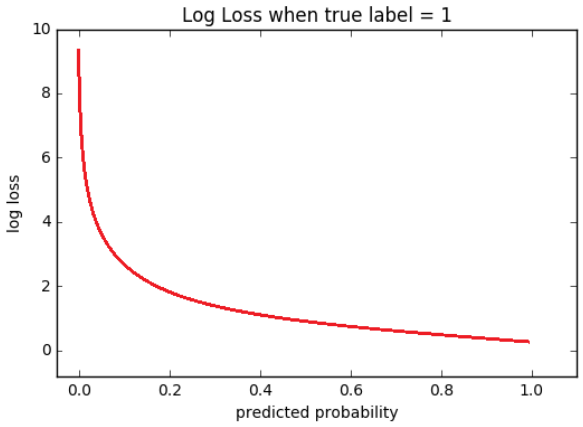


Figure 2.27: Cross Entropy Example

is obtained and compared the value to the actual observation label. The softmax fully connected layer (the last output layer in the model design) converts this highest probability to one and the rest to zero. This is compared to the actual label or ground truth value to determine the prediction correctness.

2.3 Validation of the CNN Model

This module describes validation of the proposed network using test data. This module performs training and testing validations. This module is of utmost importance for generalization of the model.

Training Validation

We plotted the training accuracy and training loss of the model. When training a model, the accuracy and loss of the model is displayed in the console. The plot helps to better visualize the training progress.

Testing Validation

We perform testing using the remaining 40% of the dataset. We generated a confusion matrix for the test data. It is used to describe the performance of the model. A confusion matrix is an $N \times N$ matrix, where N is the number of classes being predicted. $N = 8$ is used, and hence we get a 8×8 matrix. The

following definitions [14] will help us understand the result and evaluation of the confusion matrix and Area under the Receiver Operating Characteristics (AUROC) described in section 2.3.

True positives (TP) are cases which are classified correctly. For example, a blended frame showing an Open Palm gesture when predicted belong to a Open Palm gesture class is a true positive. True Negatives (TN) are cases that are classified correctly not to belong to a class. False positive (FP) is defined as cases that are classified incorrectly. For the above example of blended frame showing Single Finger gesture when predicted belongs to an Open Palm gesture class. True negative (TN) are cases that are classified incorrectly not to belong to a class. Referring to the above example if a gesture is predicted to belong to an Open Palm gesture class, but true label puts the class to belong to Single finger raised gesture. Figure 2.28 is an example of 2-class confusion matrix providing explanation.

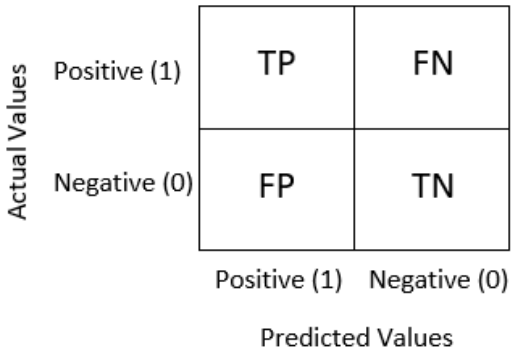


Figure 2.28: 2×2 Confusion Matrix

The percentage of the total number of predictions that were correct is defined as accuracy of the system. Equation 2.4 shows the formula of accuracy of the system. The accuracy over eight classes (one class against all) will the sum of TP for class 0 and TNs for all other classes. Similarly, the accuracy of other classes as well.

$$Accuracy = \frac{TP + TN}{Total \quad number \quad of \quad examples} \tag{2.4}$$

The proportion of positive cases that were correctly identified as positive case is known as Sensitivity, Recall, hit rate or true positive rate (TPR). It is given in equation 2.5.

$$Recall = TPR = \frac{TP}{TP + FN} \tag{2.5}$$

The proportion of actual negative cases which are correctly identified is defined as Specificity, selectivity or true negative rate (TNR). Equation 2.6 shows the formula for specificity.

$$TNR = \frac{TN}{TN + FP} = 1 - FPR \tag{2.6}$$

The proportion of positive cases that were correctly identified is called as positive predicted value (PPV) or Precision. Equation 2.7 defines the same.

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

The proportion of negative cases that were correctly identified is Negative Predictive Value (NPV).

$$NPV = \frac{TN}{TN + FN} \tag{2.8}$$

We also define the ROC curve to validate the model. Receiver Operating Characteristics represents the degree or measure of separability between classes. It visualizes the model's capability of distinguishing between classes. Higher the AUC, the better is the model in predicting the true positives. The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis. An ideal case of the ROC curve is when the classifier is perfectly able to distinguish between positive and negative classes, this means that there are no false positive and false negatives. In this situation, the top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one. This is not very realistic, but it does mean that a larger area under the curve (AUC) is usually better.

CHAPTER III: EVALUATION AND RESULTS

3.1 Model Training Analysis

The CNN model is trained on the dataset and an analysis is performed to determine how well the model was trained to generalize on real world data. The analysis is described in the below sections.

Training Dataset Cross Validation

We created a total of 8 models using the training data. The training data is split into a validation set. This set is a sample of data separated from the training data to give an estimate of the model outputs and tuning the model's hyper parameters. The validation dataset is different from the test dataset that is used to to give an unbiased estimate of the model output of the final tuned and trained model. We performed K-fold cross validation of the training data. K-fold cross validation is used to assess the results of the statistical analysis on how the overall generalization of the model is performed to an independent dataset. Tables 3.1 and 3.2 show the summary of the cross validation performed.

Table 3.1: Cross Validation Training Data Summary

$(K-1)$ **Folds**	**Training Accuracy**	**Training Loss**
Fold 1	98.31%	0.0505
Fold 2	98.50%	0.0431
Fold 3	96.93%	0.0506
Fold 4	98.43%	0.0482
Fold 5	98.74%	0.0686

In K-fold cross validation, we split the training set into k-parts or folds . We build the model based on data from $k-1$ folds and test the model on the remaining fold also called the validation set. This procedure is repeated $k-1$ times excluding a different fold (creating a different validation dataset) every time. We take the average of the values that provide the best accuracy on each of the k validation sets. A theoretical model is a good replica of reality if it has an acceptable error in prediction. When we perform cross validation, the model is tested on data that has not used in training

the model. If the prediction and thus the error rate is good, it is considered a good fit or replica of real world. Therefore, to generalize the model it should have the best fit optimization. This is possible

Table 3.2: Cross Validation Validation Data Summary

K^{th} **Fold**	**Validation Accuracy**	**Validation Loss**
Fold 1	98.44%	0.0578
Fold 2	98.02%	0.0475
Fold 3	98.21%	0.0591
Fold 4	97.08%	0.0620
Fold 5	97.59%	0.0721

by reduction or minimization of the error rate. The model is validated using k-fold cross validation over a split of $k = 5$. The training data set is split in 5 folds and the model is trained on 4 folds iteratively over all the folds. The tables show that when we perform cross validation on the optimized model obtained from section below, there is consistency in the accuracy and loss for the training and validation. This evaluates to the fact that the model is generalized and will efficiently work on real world data

Model Evaluation

Model evaluation is divided into four categories based on how they fit in generalization to validation data. The first category is underfitting. This occurs when a model is not powerful enough or has not been trained enough, that is, it has not learned the relevant patterns in the training data. Underfitting is often not discussed as it is easy to detect a given good performance metric. An underfit model will have poor performance on the training data. When validating the data, the validation and the training error will be high and in the order of $1.0 - 2.0$.

The second category is overfitting. This occurs when the model is trained for too long, resulting in the model learning pattern from the training that would not generalize to test data or real world data. When a model learns the detail and noise in the training data to an extent that it negatively impacts the performance of the model with new data. Noise in training data would mean random fluctuations in the training data are learned as concepts by the model. The issue is these learned concepts and features

do not apply to new data and negatively impact the model's ability to generalize. The validation error is high and the training error is low for an overfit model. This shows that that the model is generalizing only on the training data and when provided with an unbiased data that the model has never seen before it fails resulting in a large error rate.

The third category is a good fit model, that is, well learned on the training data and ready for real world data generalization. The validation error is and training error are low, but the validation error may be slightly higher than training error in the order of $0.001 - 0.01$. The reason for a slight higher rate is that the model is generalized and may tend to not read a very minute quantity of the real world data which is considered acceptable, since a model in practical can never be perfect to detect every real world data, but theoretically it is possible.

The last category is the unknown fit category where the validation rate is low but the training error is very high. This is known as an "unknown" fit because the conclusion drawn from the model are counter-intuitive to the concepts of machine learning.

Usually, training error in general, underestimates the validation error. Therefore, for the validation error to be lower than the training, it may be the result of the two ways the model is trained and tested. The first is that the training set may have "hard" examples to train the model on and the validation set may have the most "easy" examples to predict on. In this context, hard examples would be defined as those which are false positives in the system which have features different than those obtained to generalize a particular class. Similarly, an easy example would be those have matching features to the feature maps stored by the model work on the new data. This is reason that cross-validation is of most importance when working on such smaller dataset.

We designed various versions before obtaining a good fit for the model. We modified the CNN to build an optimal design based on the validation set separated from the training data. Table 3.3 provides the optimal design of the CNN model. It provides model summary obtained using the summary functionality from Keras API [7]. The total trainable parameters in this model is $476,744$ as displayed in the table.

The first step was the ratio of division between train, validate and test data. The split ratio was set to $50 : 10 : 40$ (that is, 60:40 train-to-test ratio). This was not based on any thumb rule, and was chosen

Table 3.3: Proposed Model Summary

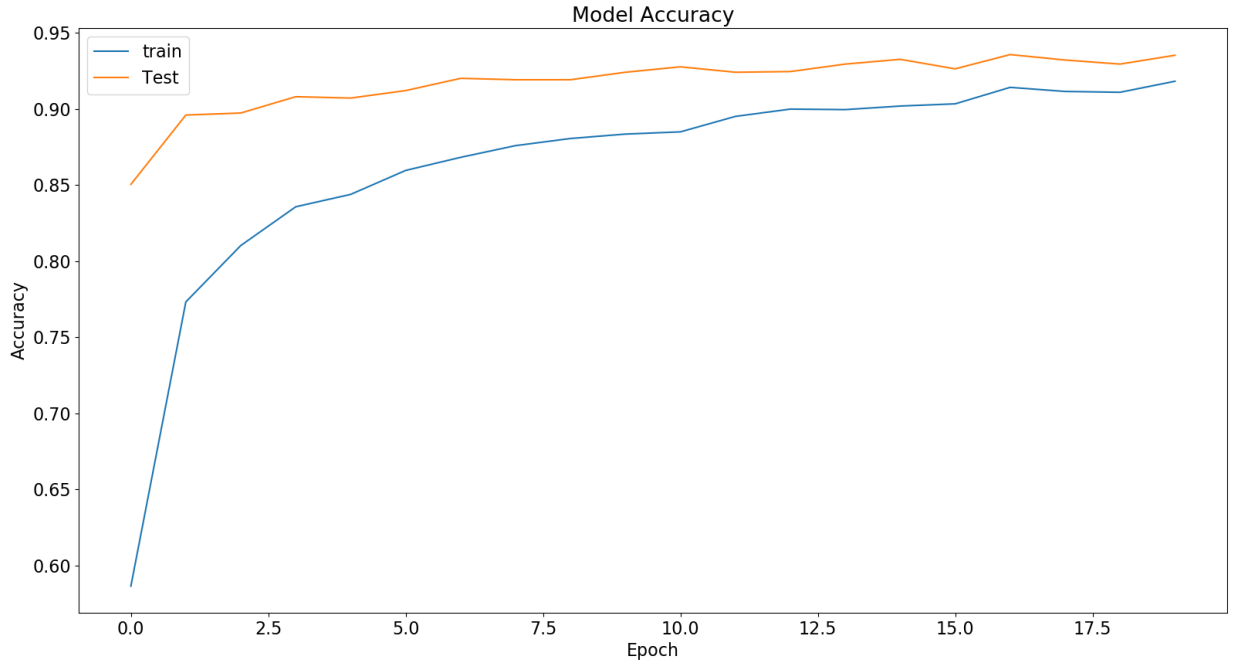
Layer (type)	**Output Shape**	**Parameters**
Conv2d_1 (Convolution2D)	(None,44, 44, 64)	3200
max_pooling2d_1 (Max Pooling)	(None,22, 22, 64)	0
Dropout_1 (Dropout)	(None,22, 22, 64)	0
Conv2d_2 (Convolution2D)	(None,18, 18, 64)	102464
max_pooling2d_2 (Max Pooling)	(None,9, 9, 64)	0
Dropout_2 (Dropout)	(None,9, 9, 64)	0
Conv2d_3 (Convolution2D)	(None,7, 7, 128)	73856
max_pooling2d_3 (Max Pooling)	(None,3, 3, 128)	0
Dropout_3 (Dropout)	(None,3, 3, 128)	0
Flatten_1 (Flatten)	(None,1152)	0
Dense_1 (Dense)	(None,256)	295168
Dropout_4 (Dropout)	(None,256)	0
Dense_2 (Dense)	(None,8)	2056
Total Parameters	-	476, 744
Trainable Parameters	-	476, 744
Non-trainable Parameters	-	0

arbitrarily. The result are shown in table 3.4. We selected the model V4 (60 : 40) as the efficient and optimized model. Each model design is described below and the analysis for selection of V4 (60 : 40) as the most generalized model.

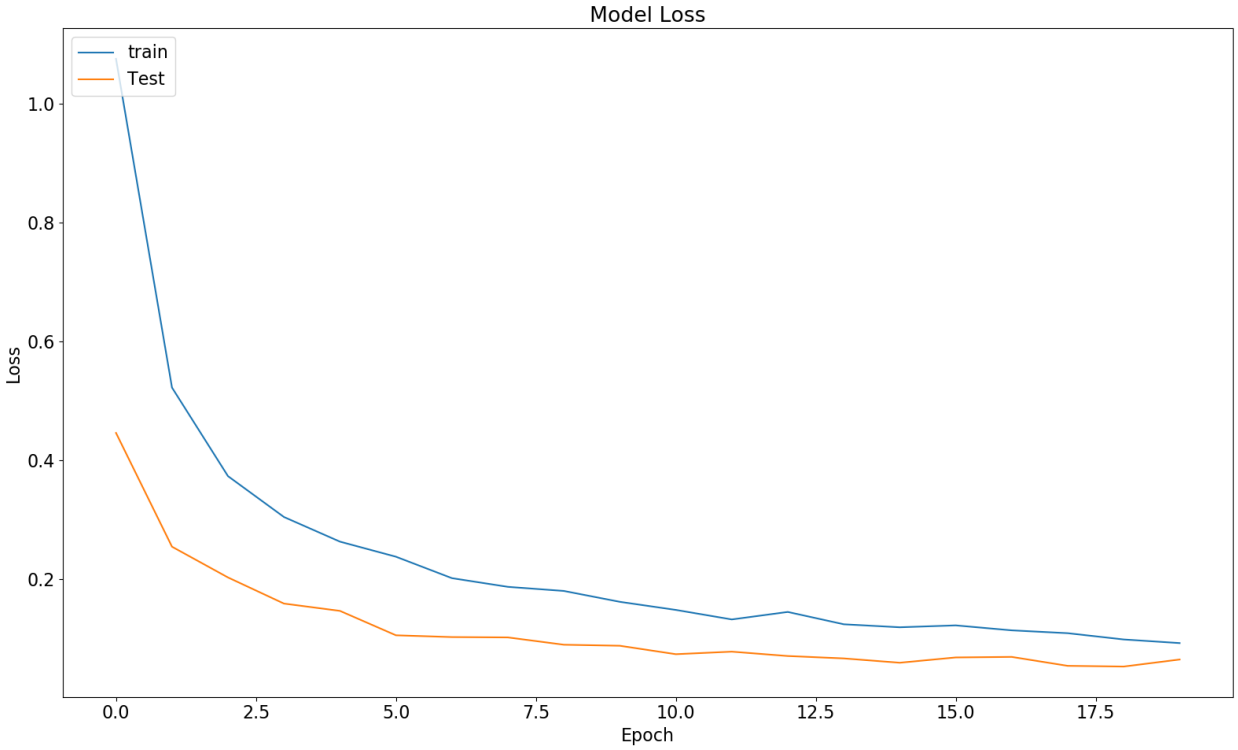
Table 3.4: Model Accuracy and Loss for various designs

		Validation		**Training**		**Testing**	
Model	**Ratio**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
V1	60 : 40	93.44%	0.2047	88.41%	0.2921	92.97%	0.1989
V1	70 : 30	95.84%	0.1735	95.00%	0.1735	95.29%	0.1532
V2	60 : 40	94.02%	0.1518	94.37%	0.2000	94.45%	0.1850
V2	70 : 30	95.02%	0.1299	94.46%	0.1716	94.54%	0.1775
V3	60 : 40	98.12%	0.0760	97.96%	0.0601	98.22%	0.0581
V3	70 : 30	98.87%	0.0358	98.30%	0.0623	98.29%	0.0647
V4	60 : 40	98.80%	0.0594	98.32%	0.0425	98.45%	0.0515
V4	70 : 30	98.49%	0.0783	99.06%	0.0314	98.09%	0.0619

We start with a simple model V1 having a single convolution layer having 32 filters of size 3. We obtained very high error rates thus, concluding that the model was underfitting. This shows the overall structure of V1. Figure 3.29 shows the accuracy convergence curve and the loss convergence curve.



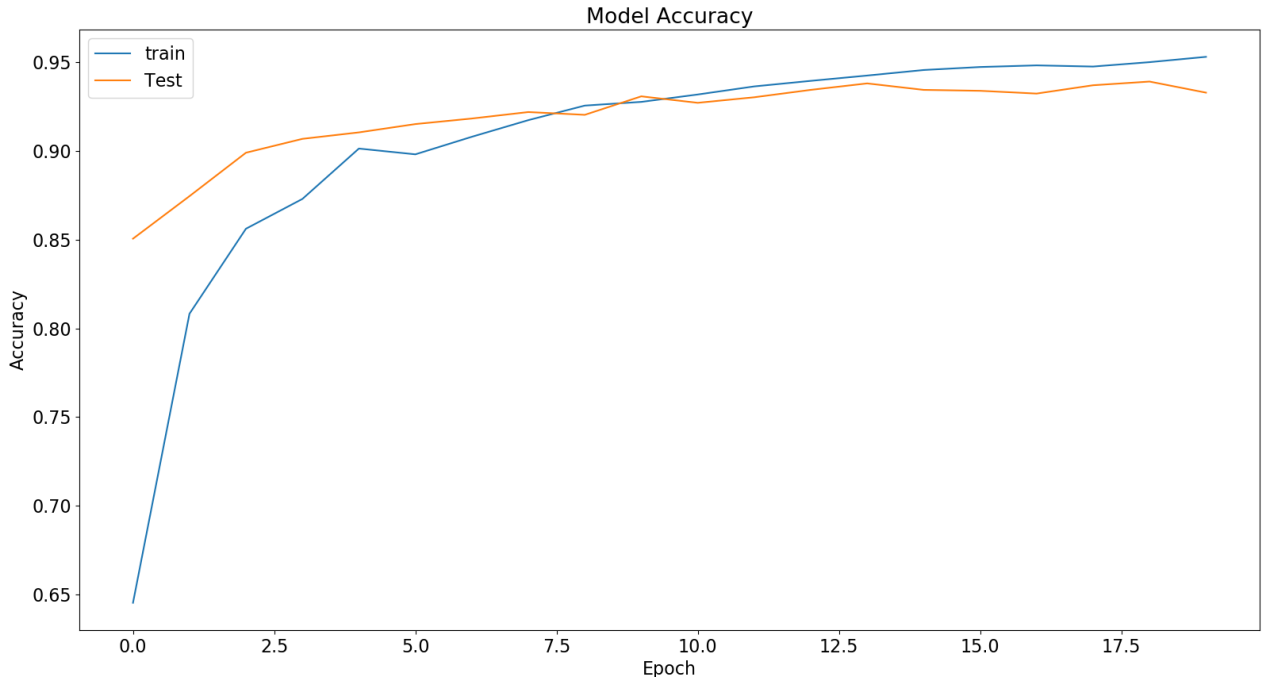
(a)



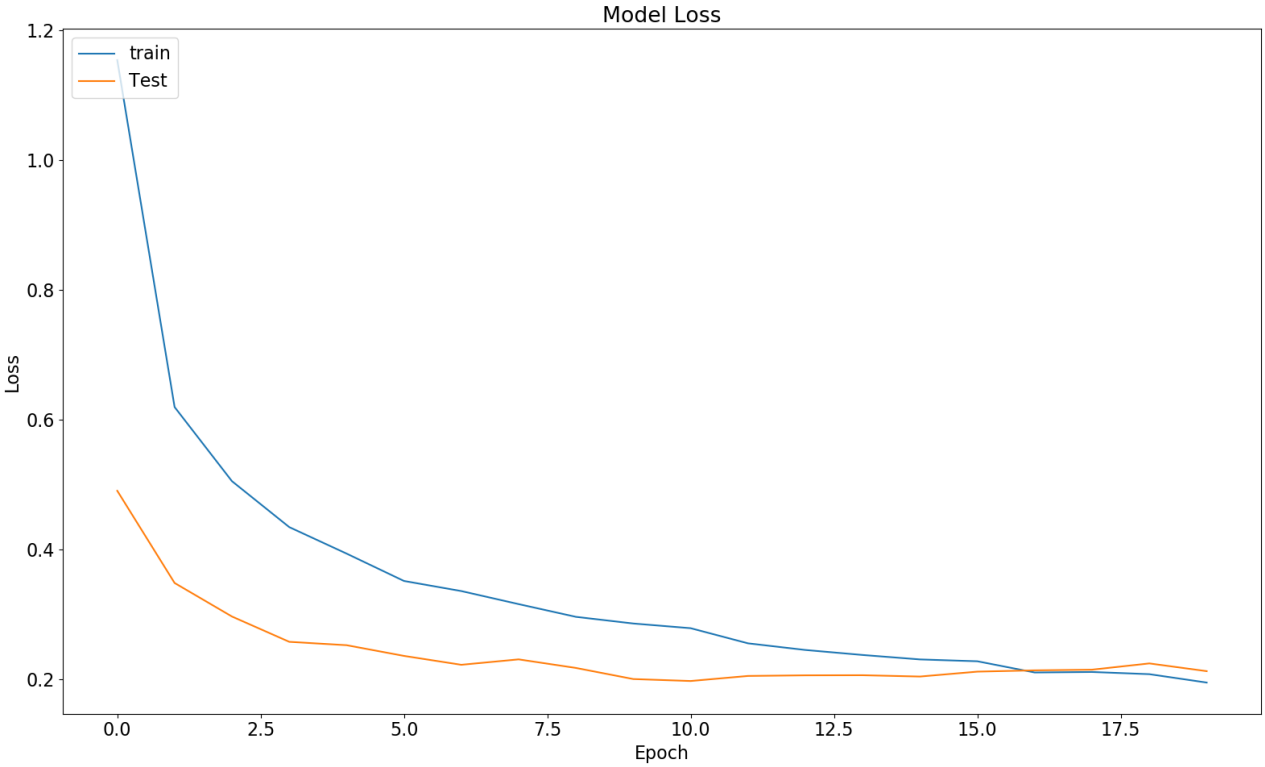
(b)

Figure 3.29: V1 Curves (60:40): (a) Accuracy (b) Loss.

We changed the ratio to 70 : 30. Figure 3.30 provide the results of this model.

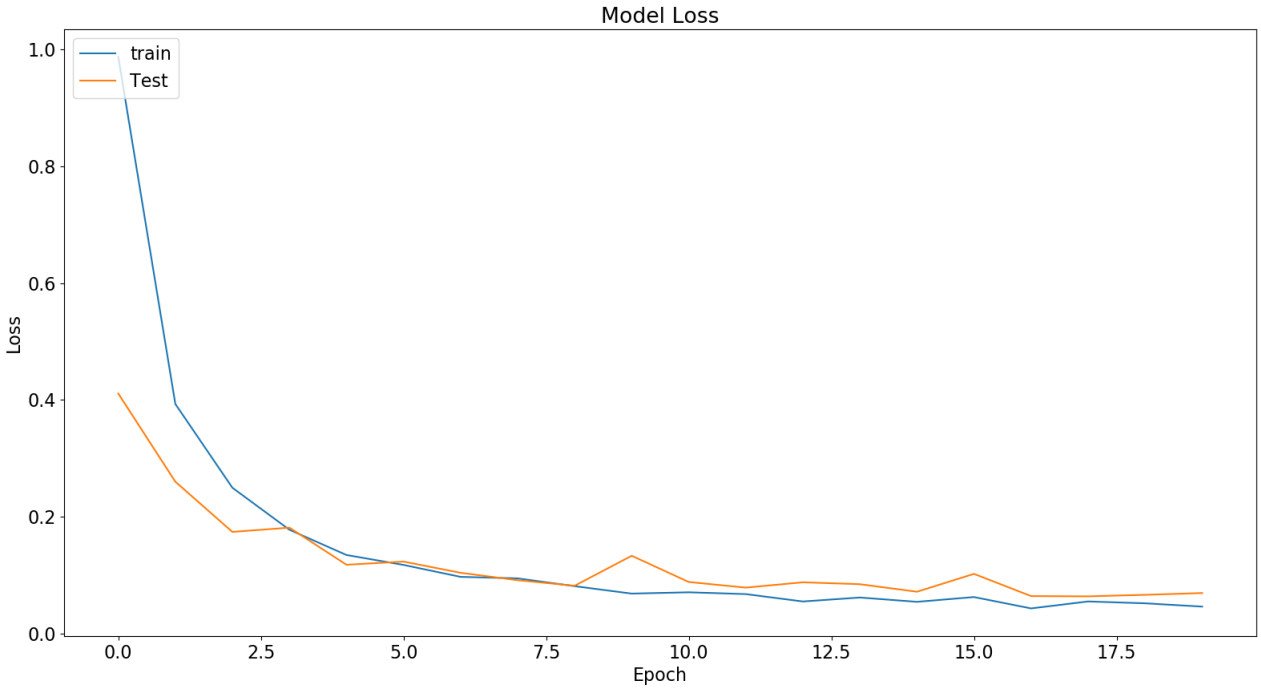


(a)

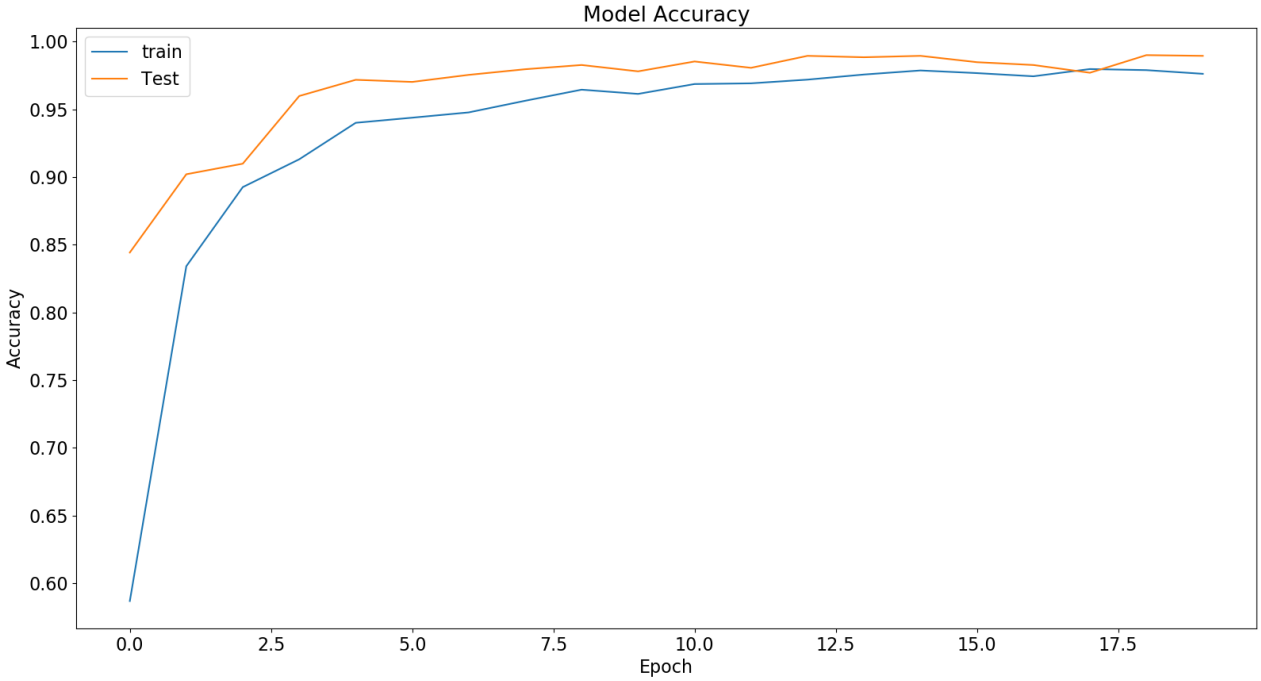


(b)

Figure 3.30: V1 Curves (70:30): (a) Accuracy (b) Loss.



(a)



(b)

Figure 3.31: V2 Curves (60:40): (a) Accuracy (b) Loss.

The model is still under fitting as it has a very high rate of error.

$INPUT - [[32CONV - RELU] - [2 \times 2POOL - 0.25DROPOUT] - [32FC - RELU] - [8FC]]$

In the second design V2, we used 64 filters of size 3 and increased the fully connected layer to have 64 neurons keeping the remaining architecture the same. The model validation loss is reduced by a minuscule amount and therefore, the model is still underfitting. This model also has a higher error rate, therefore, we cannot use this design for our optimized design Figure 3.31 display the convergence and accuracy of this model.

Similarly, as V1, we changed the ratio to $70:30$ and train the model. The result of high error rate has small change and therefore we consider the next version of the model. Figure 3.32 and figure 3.33 provide this example.

$INPUT - [[64@3 \times 3CONV - RELU] - [2 \times 2POOL - 0.25DROPOUT] - [64FC - RELU] - [8FC]]$

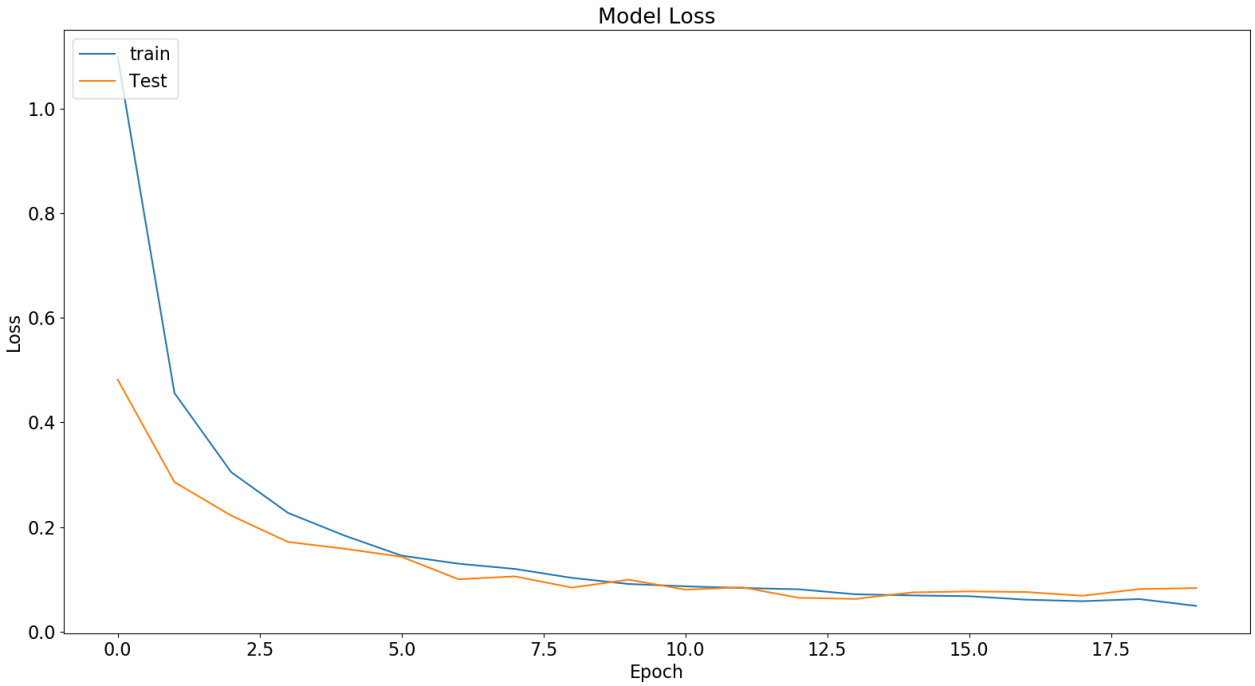


Figure 3.32: V2 Convergence curve: Accuracy (Train-to-Test 70:30).

The addition of another convolution layer does not reduce the error rate significantly and therefore we excluded the results, therefore, we implement a 3-layered convolution model.

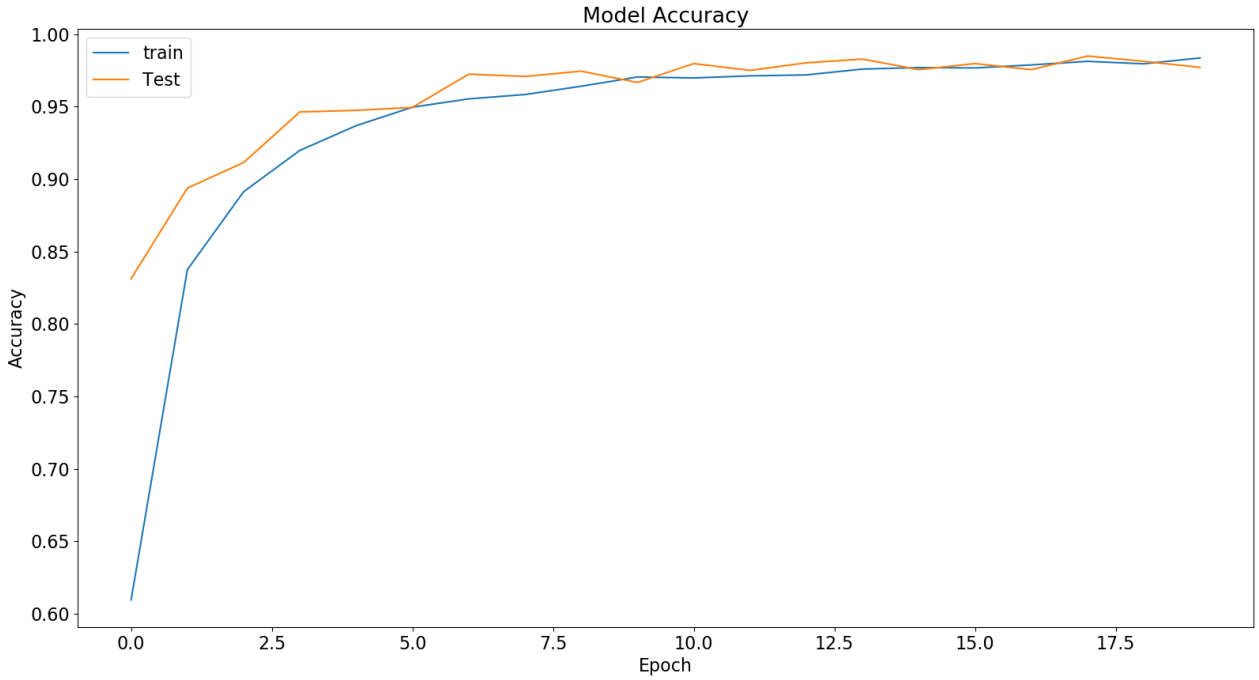
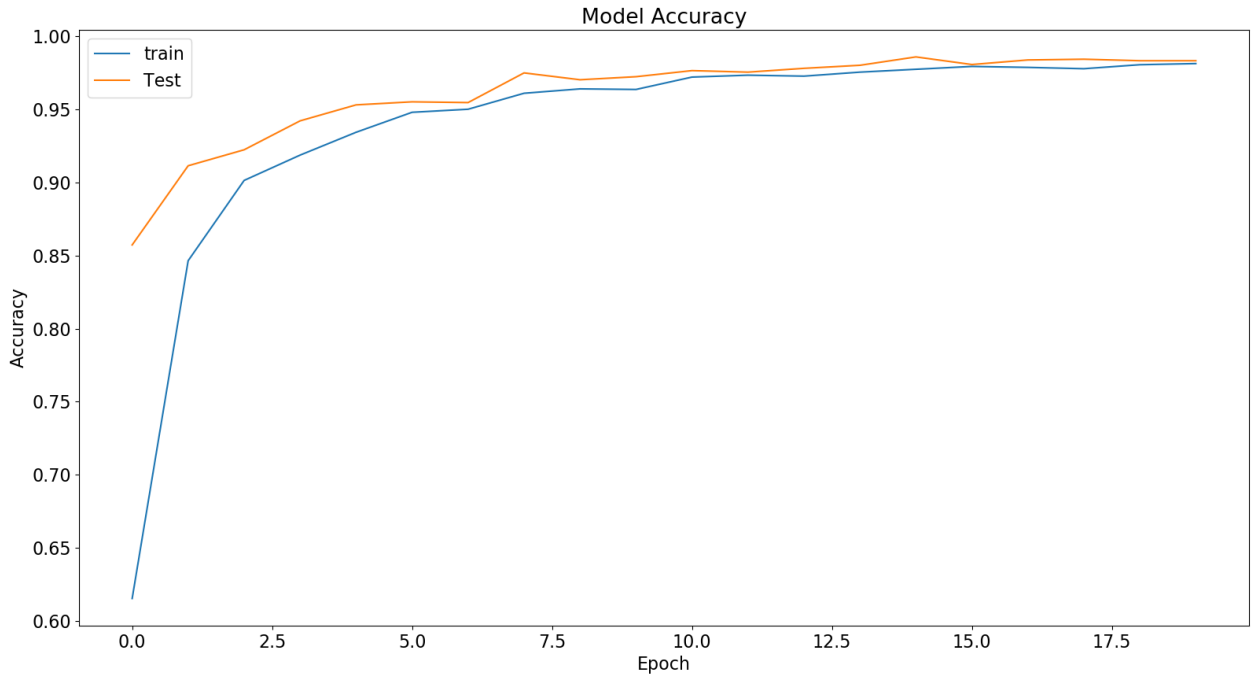


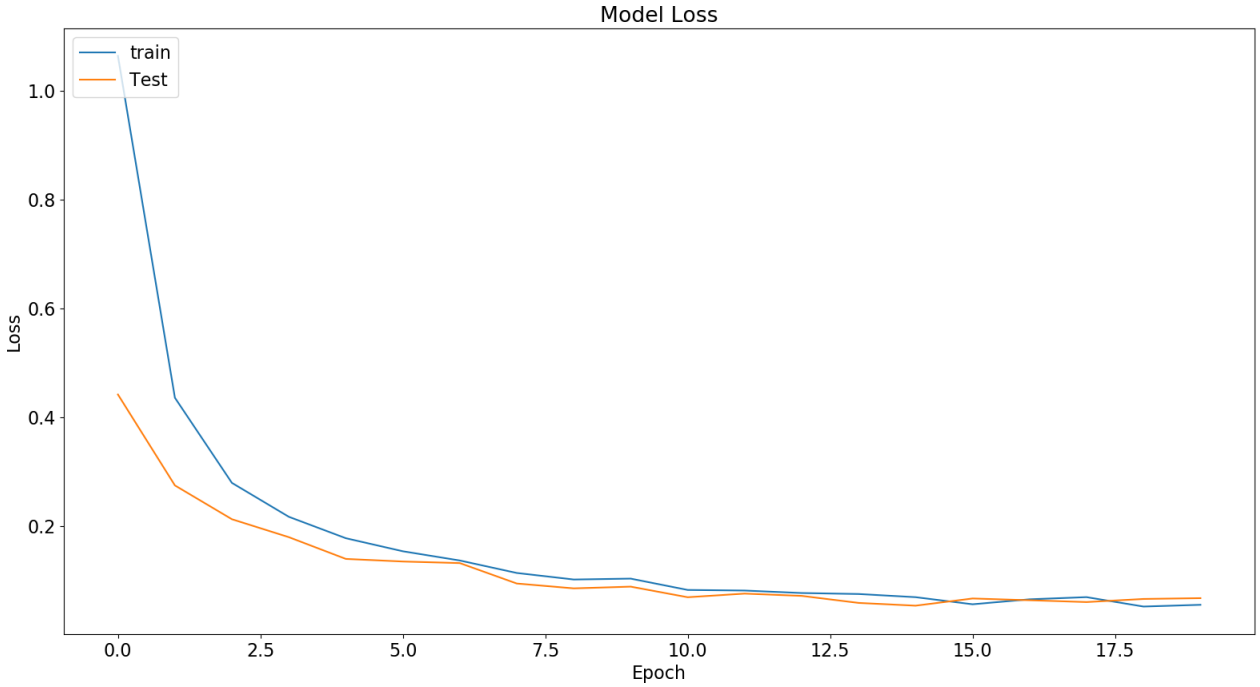
Figure 3.33: V2 Convergence curve: Loss (Train-to-Test 70:30).

The third version design model V3 is implemented with three convolution layers. The first convolution layer has 64 filters of size 7. Second layer has 64 filters of size 5 and the final convolution layer 64 of size 3. Each of the layer are followed by a max pooling layer of stride 2×2. We added a dropout of 25% after each layer to drop neurons and reduce overfitting. Figure 3.34 show the results of this model.

The result shows a validation error of is slightly over training error, thus this model of V2 is good fit. The only problem with this model is it has a significantly higher error in both validation and training. This prompted to reduce the error a bit further by upgrading the filters on the final convolution to reduce the error rate a bit further. We tested the same model V3 with train-to-test ratio of $70 : 30$, the result was the model was overfitting.



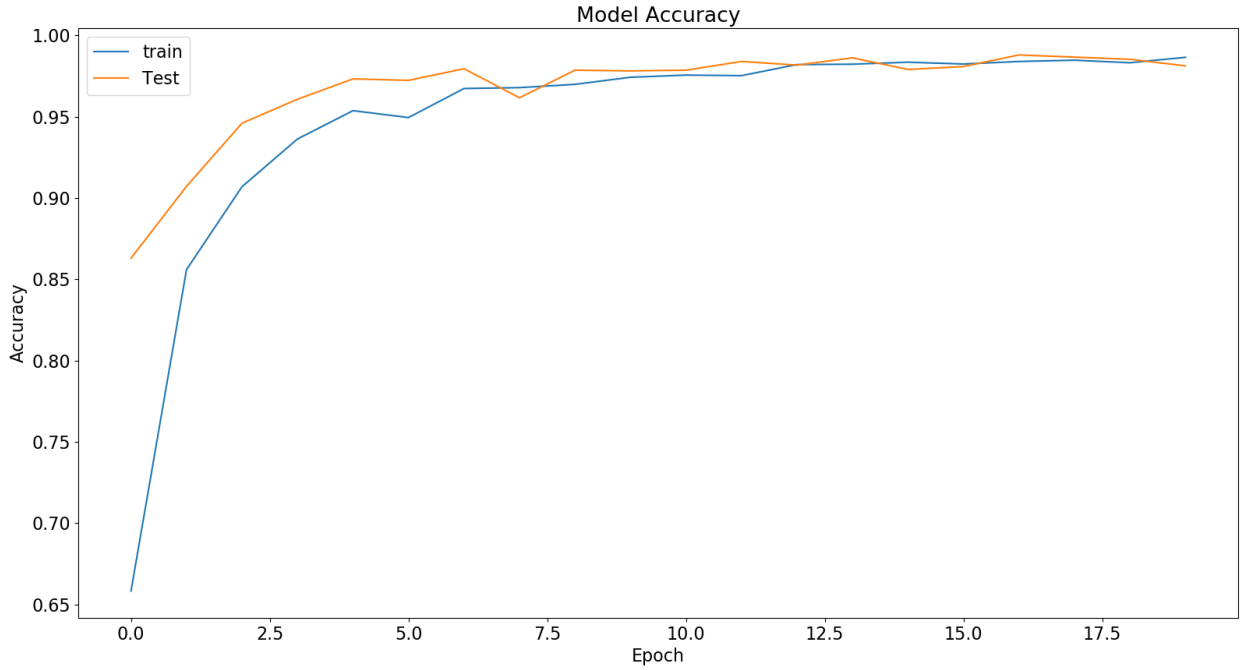
(a)



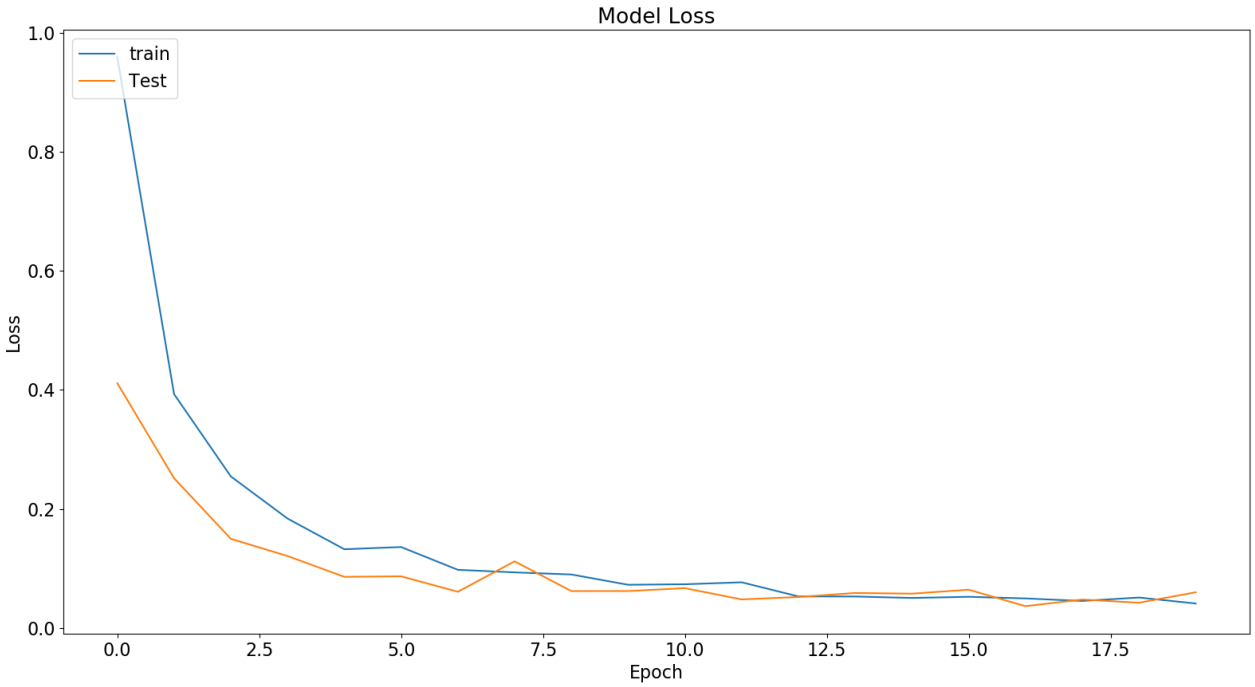
(b)

Figure 3.34: V3 Curves (60:40): (a) Accuracy (b) Loss.

To reduce this overfitting, we introduced a dropout rate of 75% after the dense



(a)



(b)

Figure 3.35: V3 Curves (70:30): (a) Accuracy (b) Loss.

layer, but it negatively impacted the result giving an uknown fit. Figure 3.35 shows the overfitting

results. $INPUT - [[CONV - RELU - POOL - DROPOUT] * 3 - [64FC - RELU] - [8FC]]$

Model V3 for the train-to-test ratio is a good fit, but we tried to improve on the error value on model V4. The final convolution was replaced with 128 filter and same size of 3. The dense layer was improved to 128 neuron but there was no significant improvement. Therefore, we increased the neurons to 256. The rest of the architecture remained the same as previous model. This model reduced the error rates by approximately 0.02 values. Figures 3.36 and 3.37 provide an details of the accuracy and training loss. We also implemented the same for the ratio of 70 : 30 and the result was

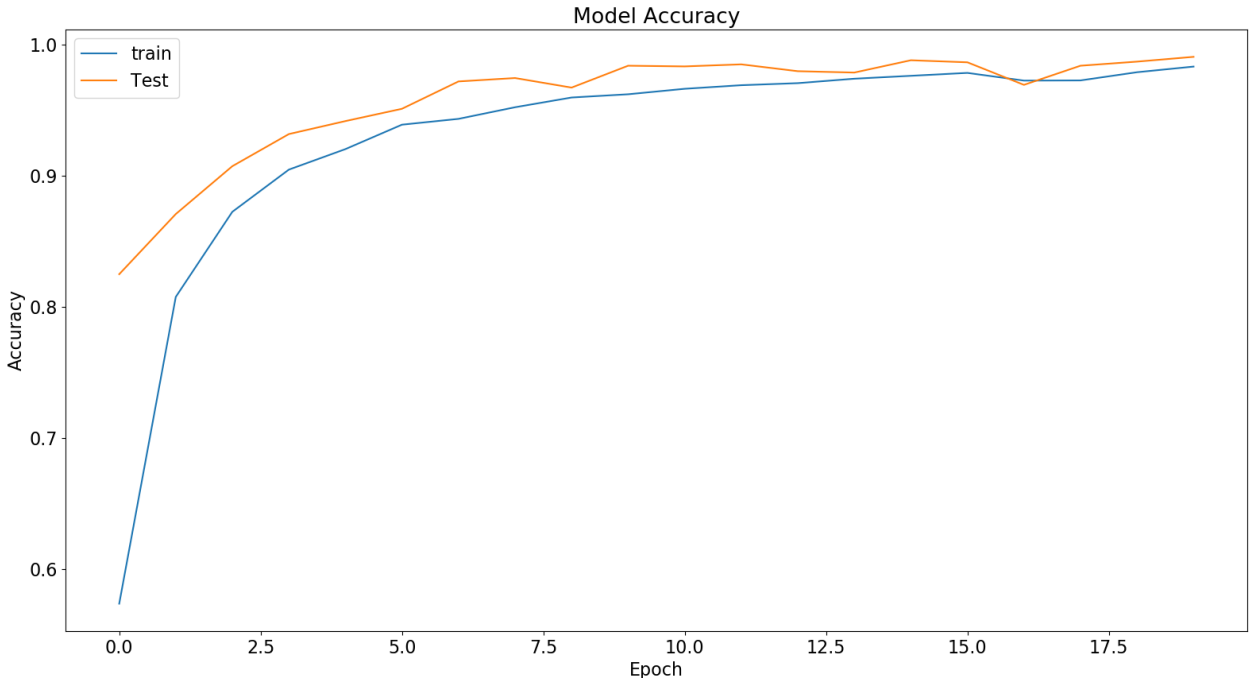


Figure 3.36: V4 Convergence curve: Accuracy (Train-to-Test 60:40).

overfitting of the model V4. Therefore, the reason of overfitting when using 70 : 30 ratio was that since there is additional 10% increase in the training dataset. The feature and learning of the model increased. Thereby, the model was trending to closely fit on the training data and the noise associated with it. Figure 3.38 gives the detail of the accuracy and the loss functions of the model. These noise were added up as features and result in overfitting the model. Usually, the train-to-test ratio is set to 70 : 30 or 80 : 20 when there are a large number of classes in the design of the dataset. The additional features are obtained from the added up $10 - 20\%$ of data in the training. This does not point to the fact that the dataset size should be less. The higher number of classes the higher the data to optimize

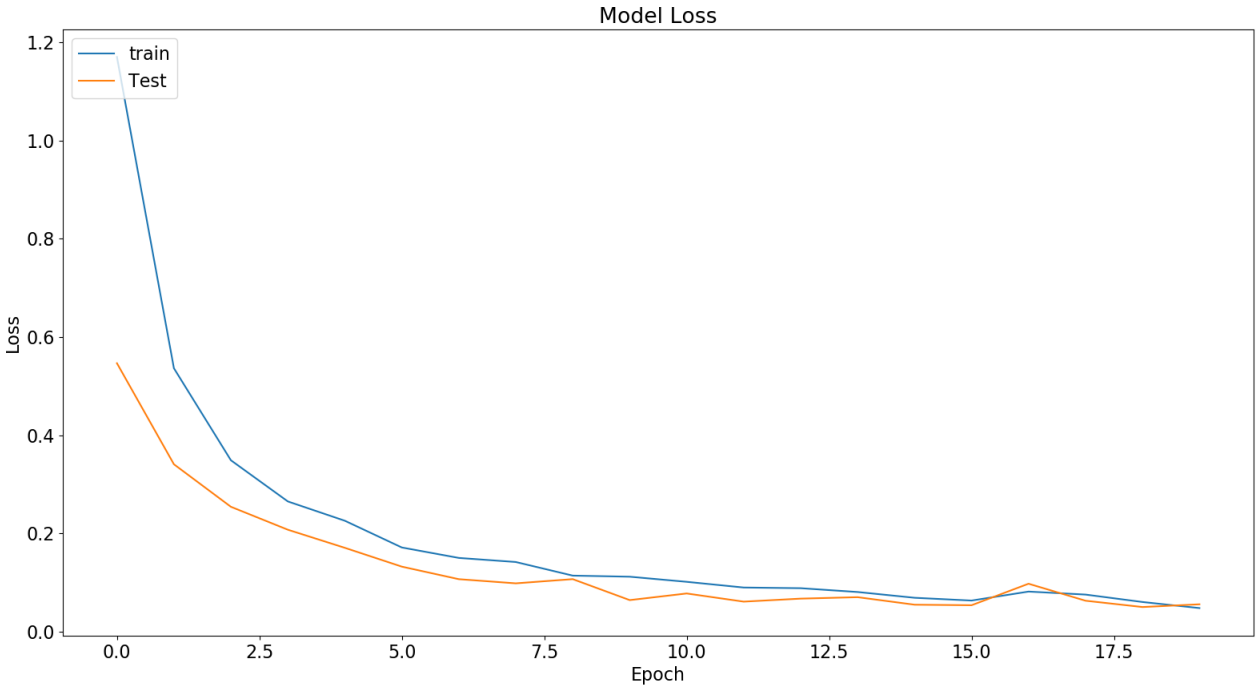
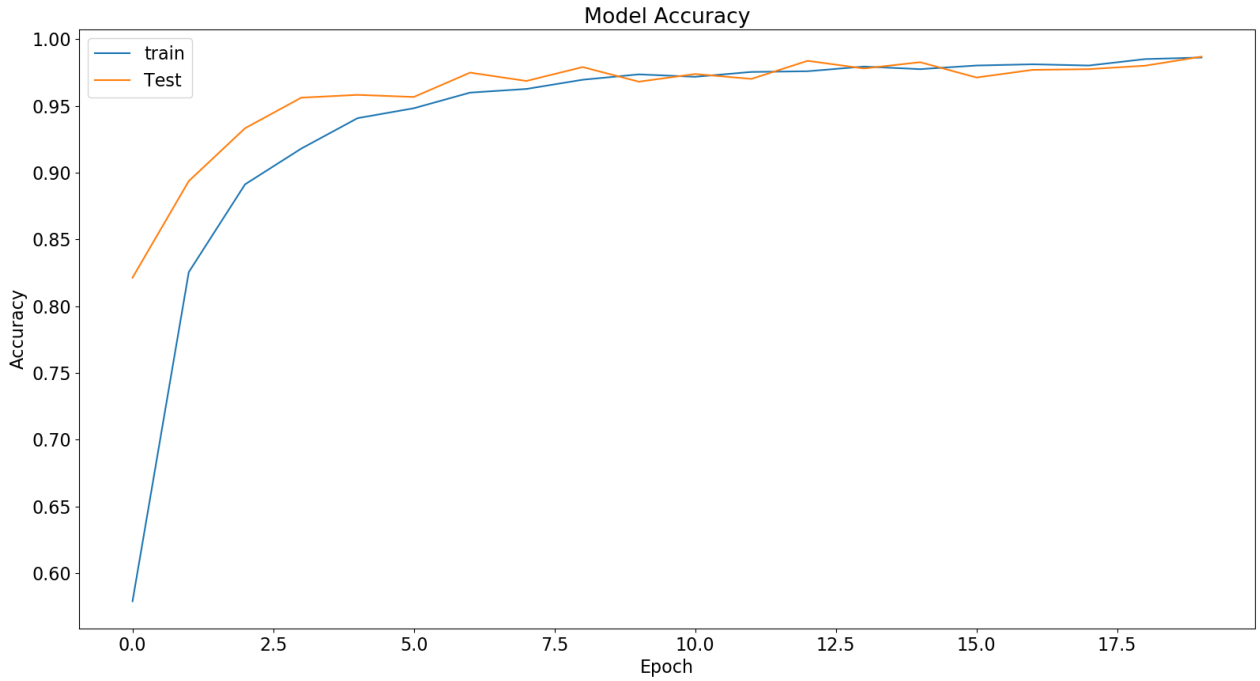


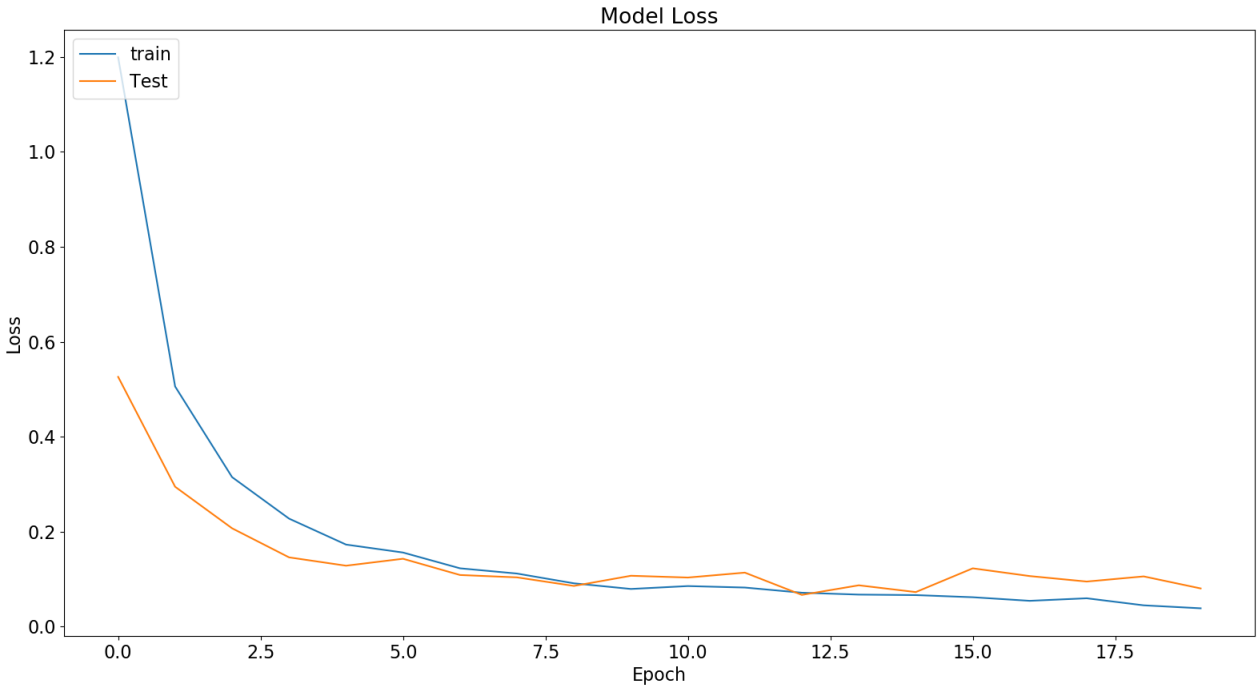
Figure 3.37: V4 Convergence curve: Loss (Train-to-Test 60:40).

the generalization of the model training.

$INPUT - [[CONV - RELU - POOL - DROPOUT] * 3 - [256FC - RELU] - [8FC]]$ Table 3.4 shows that we obtained a fine tuned accuracy of 98.45 % on model V4 for train-to-test ratio of $60:40$. The model is optimized to its efficiency to provide a good fit. We used this optimized and computationally inexpensive model in our HGRS.



(a)



(b)

Figure 3.38: V4 Curves (70:30): (a) Accuracy (b) Loss.

The k-fold cross validation is performed on this model to provide a mean accuracy and loss. There were a multiple attempts that were not significant enough to put our results. The results mentioned

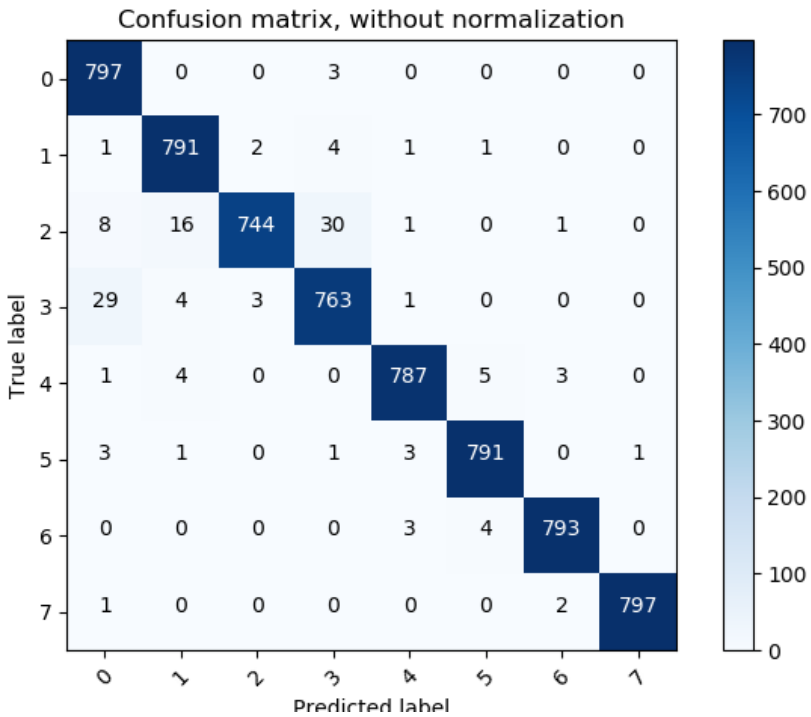
in this table are significant enough that provide fine tuning of the model to generate an optimized and computationally efficient.

3.2 Quantitative Analysis

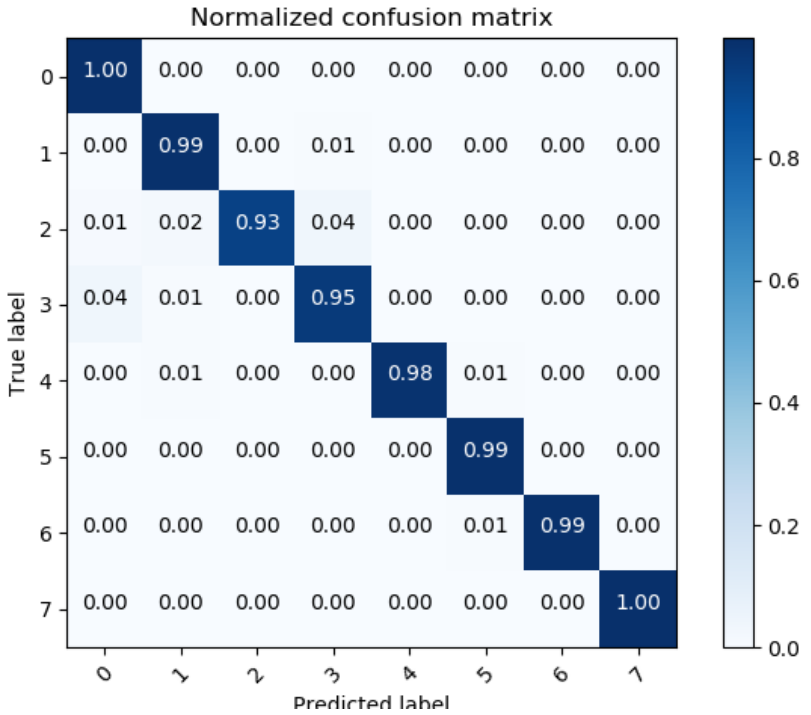
Quantitative Analysis of a model is performed using standard metrics that are obtained when we train the model. This metrics is more related to performance analysis of the curve based on numerical values and graphical visualization. Metrics such as confusion matrix, Area under the Receiver Operating Characteristics help understand how well the model would perform on real world examples that may be unbiased. These metrics ascertain the model generalization without placing the system in real world. This means that the model can be again fined tuned to improve the factors of the model to obtain a better matrix.

Confusion Matrix

The above section showed the convergence curve and the cross validation to obtain a good fit of the proposed model. We now design the confusion matrix on the unbiased test data which acts as new data to the trained model. Table 3.4 shows the testing results. We use the model V4 as a optimized model. Similarly, figures 3.36 and 3.37 show the accuracy and loss of the model. We designed the confusion matrix without normalization (raw examples) alongwith normalized matrix. Figures 3.39 display the confusion matrices.



(a)



(b)

Figure 3.39: Confusion Matrices: (a) Original (b) Normalized.

Area under the ROC Curve

A Receiver Operating Characteristic (ROC) [29] curve demonstrates the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity). ROC analysis is a standard tool used in evaluation of two class classification problems. ROC dimensions for a multiclass can be simplified as some dimensions are independent of each other [24]. Figure 3.40

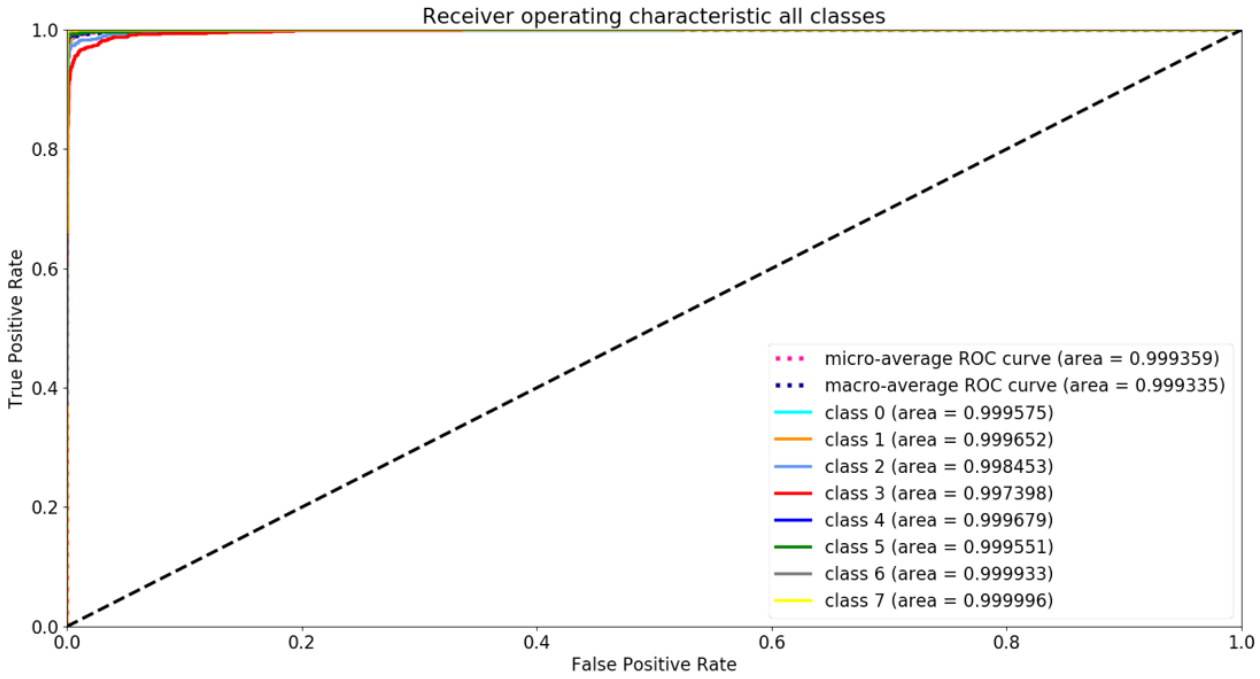


Figure 3.40: Receiver Operating Characteristics for all classes.

is the actual fit size graph and figure 3.41 is the zoom-in graph to analyze the class characteristics. The graph in the figure 3.41 show eight ROC curves representing each gesture class. The accuracy of the ROC depends on how well the curve separates the group being tested into those that belong to the class and those which do not. The ROC for a multi class system is defined by comparing the gesture class in question with the remaining classes, that means it is a one class versus all the rest curve. The figure shows the graph for each class and we noticed that each class has an area under the curve slight above 99% approximately proving it is an excellent test with each of the class. Theoretically, if the area under the curve is 1, it represent a prefect test accuracy. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate is the class test. In figure 3.41, we see that the ROC curve for classes $5-7$ are closer to the left border and top ROC space

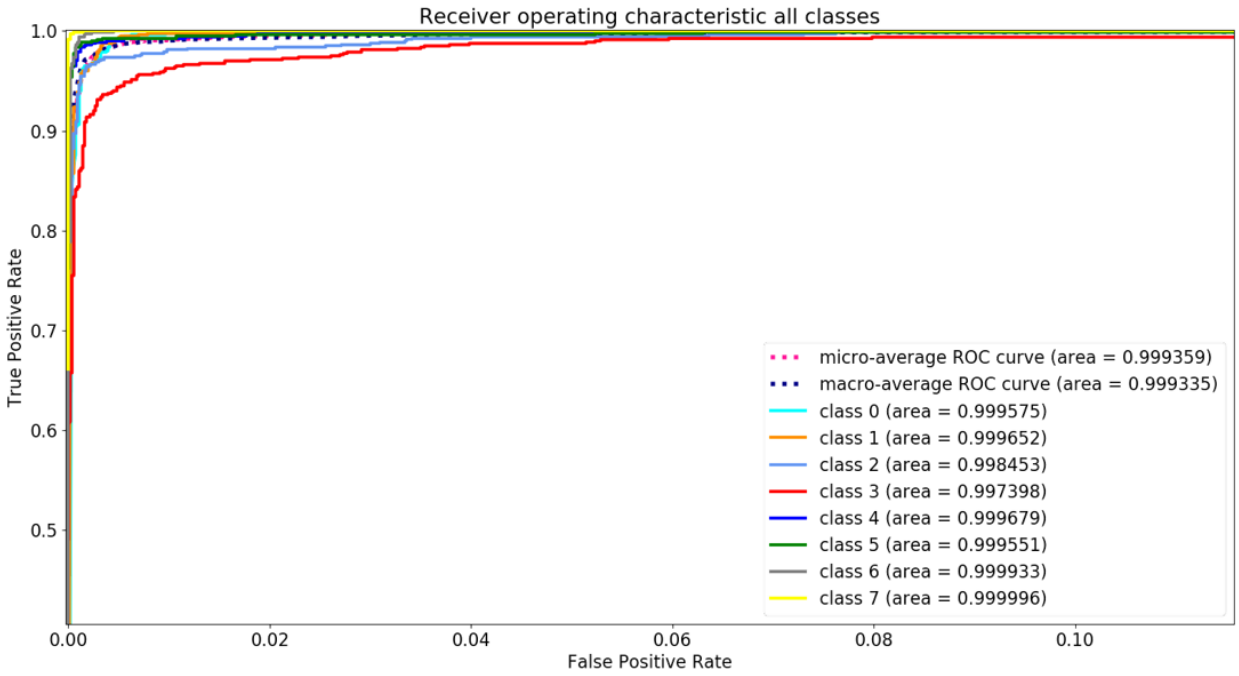


Figure 3.41: Receiver Operating Characteristics for all classes.

region, thus providing a higher accuracy for these classes. This also meant that each of these had a better performance for recognition with real world examples. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate is the test performed. In figure 3.41 that classes $0-4$ are closer to 45-degree diagonal, thus showing that these classes have slightly lesser accuracy in comparison to the remaining four classes in correspondance to real world examples. Classes 2 and 3 have a slightly lesser accuracy which corresponds to the values displayed by the confusion matrix

Precision-Recall curve

Precision-Recall is a useful measure of the success of prediction when the classes are very imbalanced [47]. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. The precision-recall curve shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall). A system with

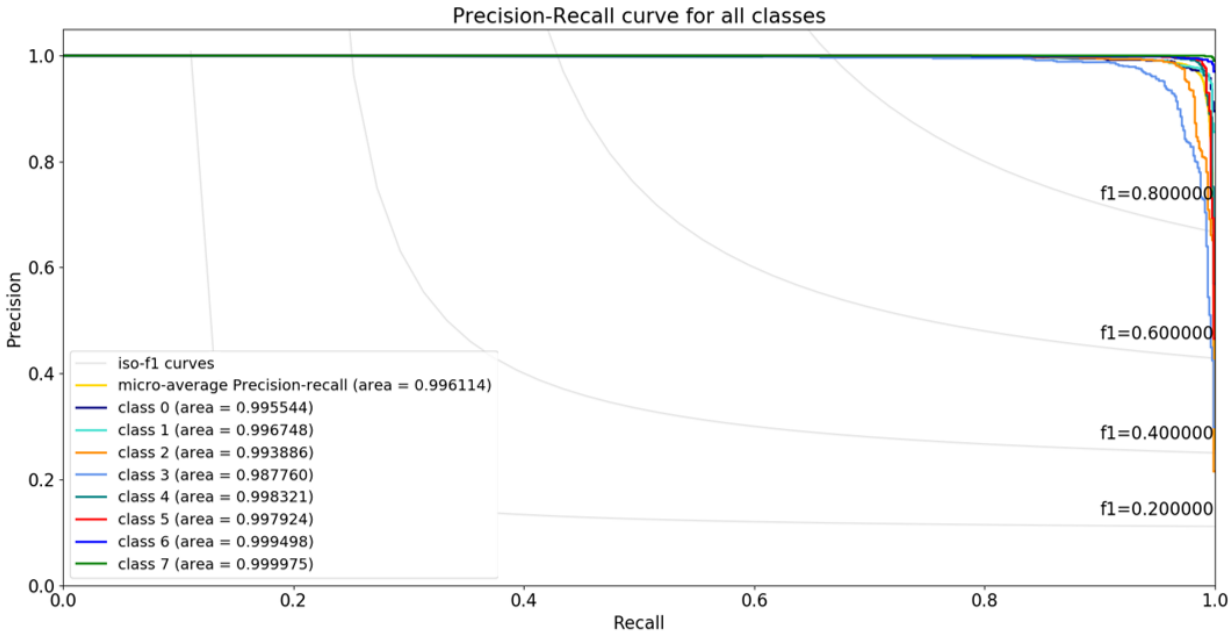


Figure 3.42: Precision-Recall curve for all classes.

high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly. Figure 3.42 shows the precision-recall curve and the figure 3.43 provides a zoom-in view for visual analysis. We can visualize in the graph that the system has a high precision and low recall thereby infering that the predicition results will have higher chances of being a true positive in the system with unbiased data.

3.3 Qualitative Analysis

Qualitative analysis is defined as the use of the system in real time with unbiased and unknown environment, checking the performance of the system. We performed a simple qualitative analysis by controlling the system music player. The results were accurate when gestures were performed decisively, with this we mean that the we used performed the gesture within the three second duration and also had the arm raised full to shoulder level to perform the gesture accurately. Figures 3.44 - 3.51 display the results for all the eight gesture classes during a live system test.

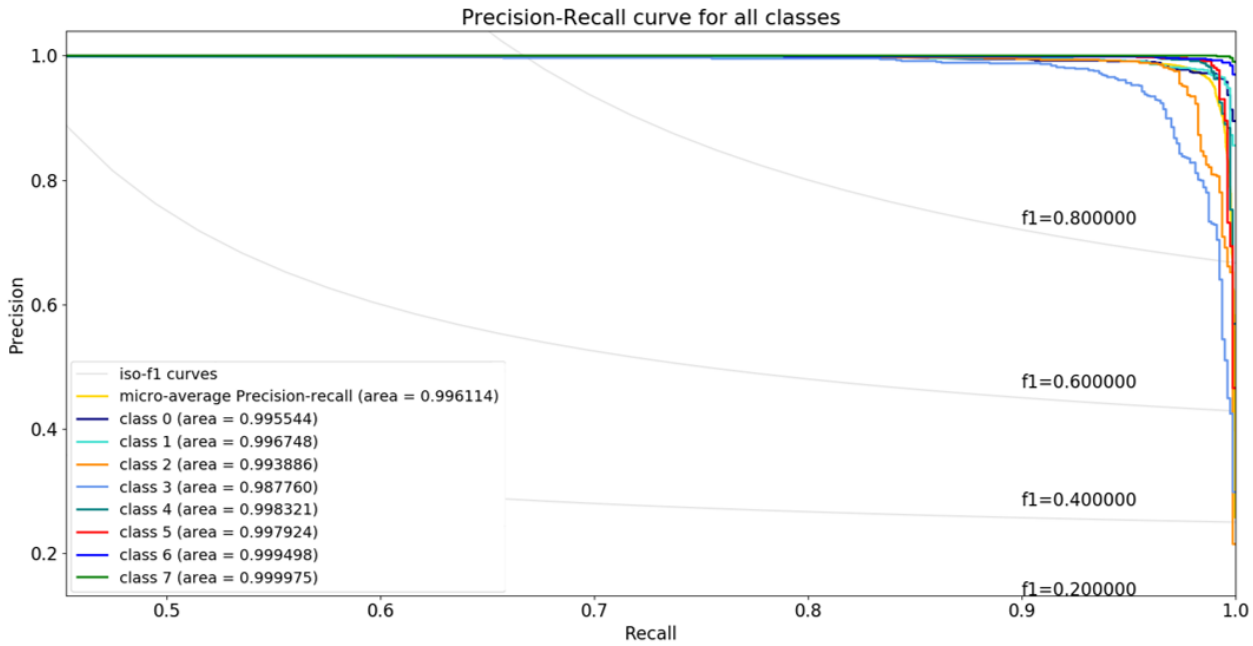


Figure 3.43: Precision-Recall curve for all classes.

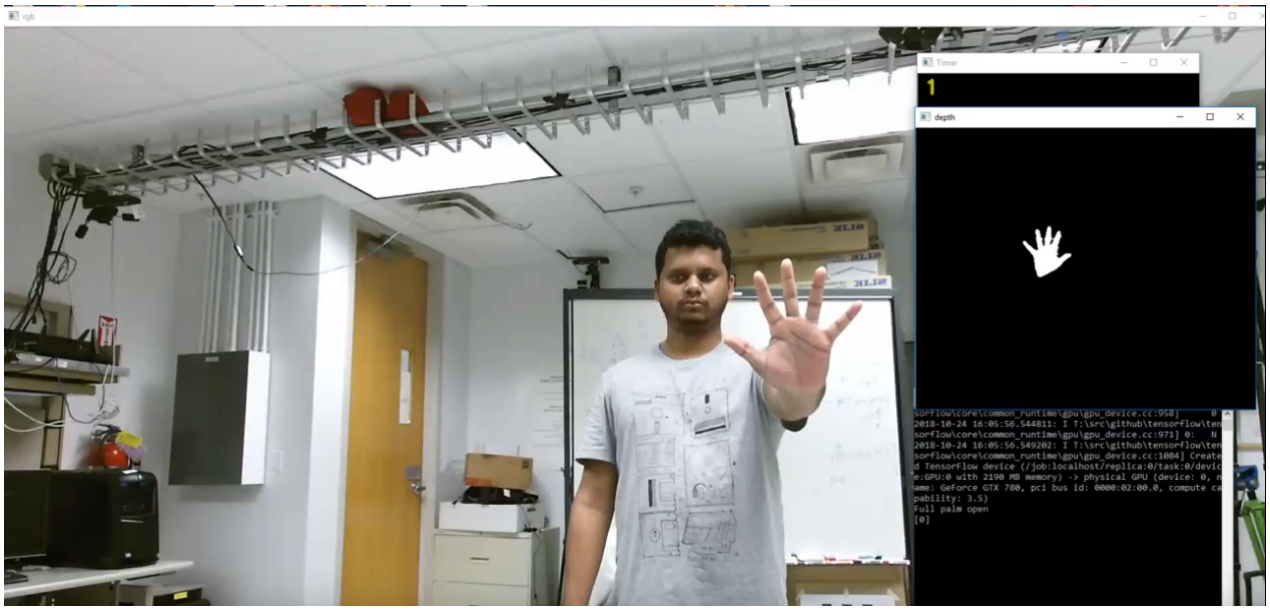


Figure 3.44: Live system test class 0.

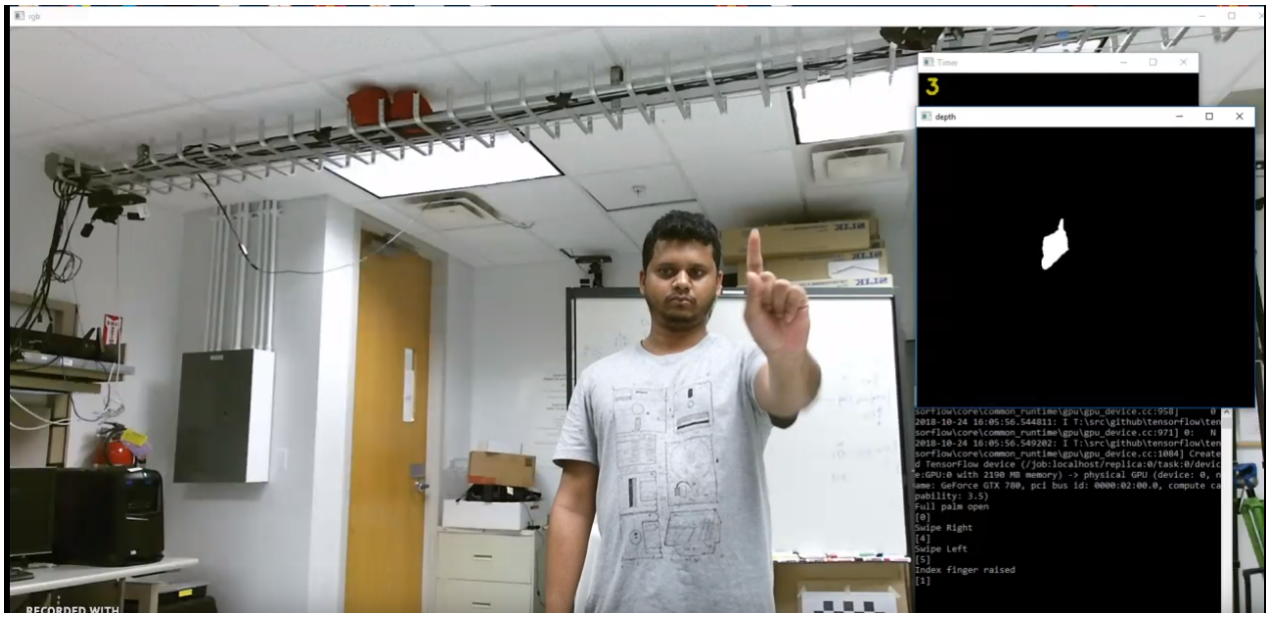


Figure 3.45: Live system test class 1.

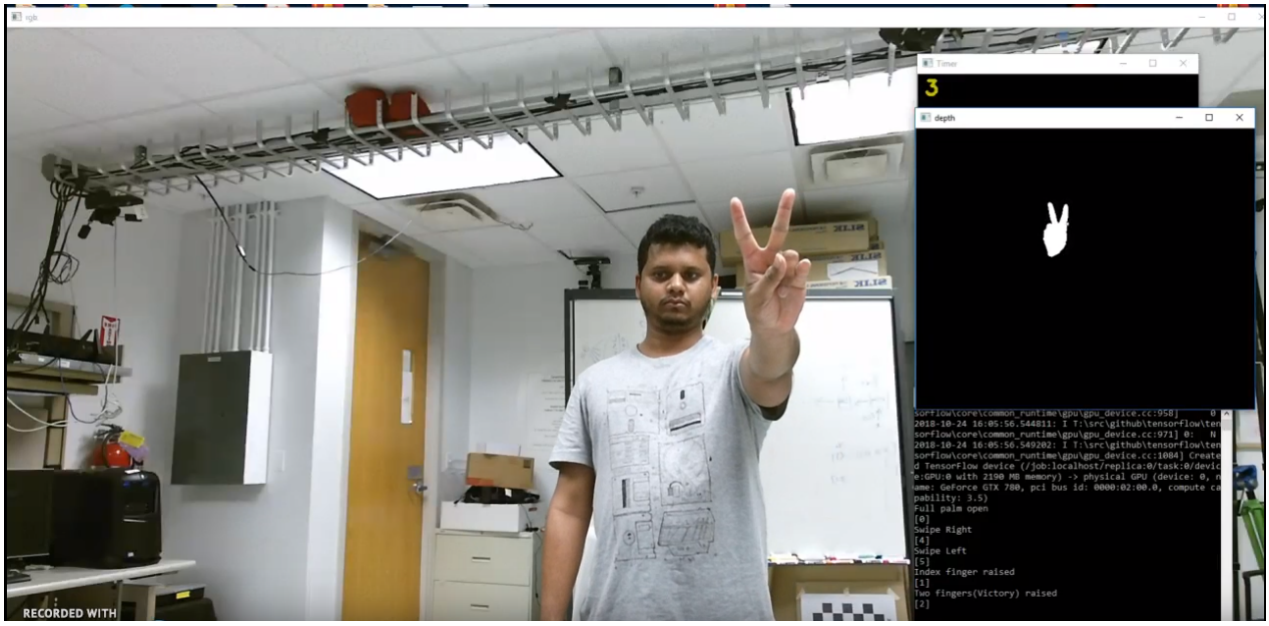


Figure 3.46: Live system test class 2.

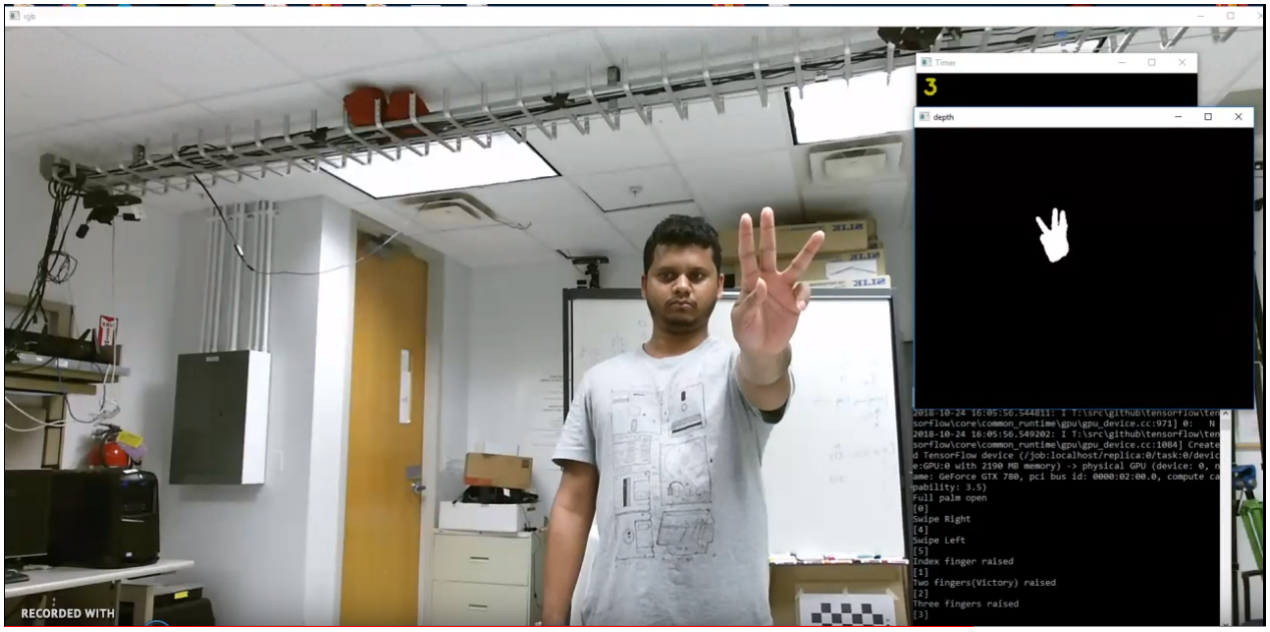


Figure 3.47: Live system test class 3.

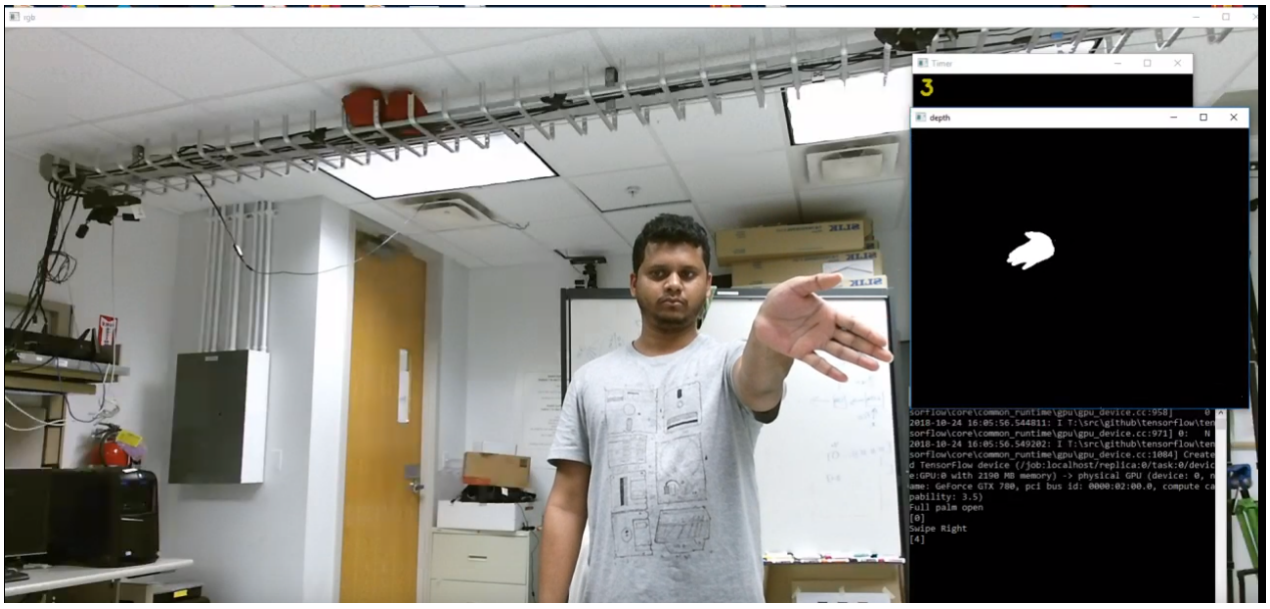


Figure 3.48: Live system test class 4.

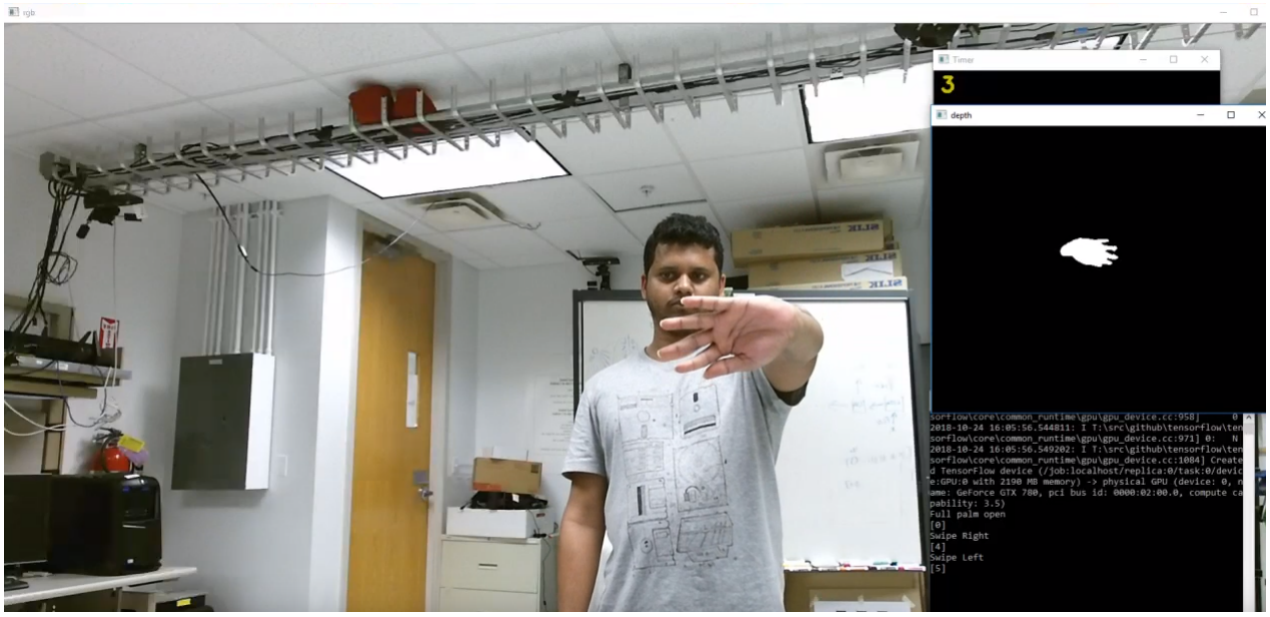


Figure 3.49: Live system test class 5.

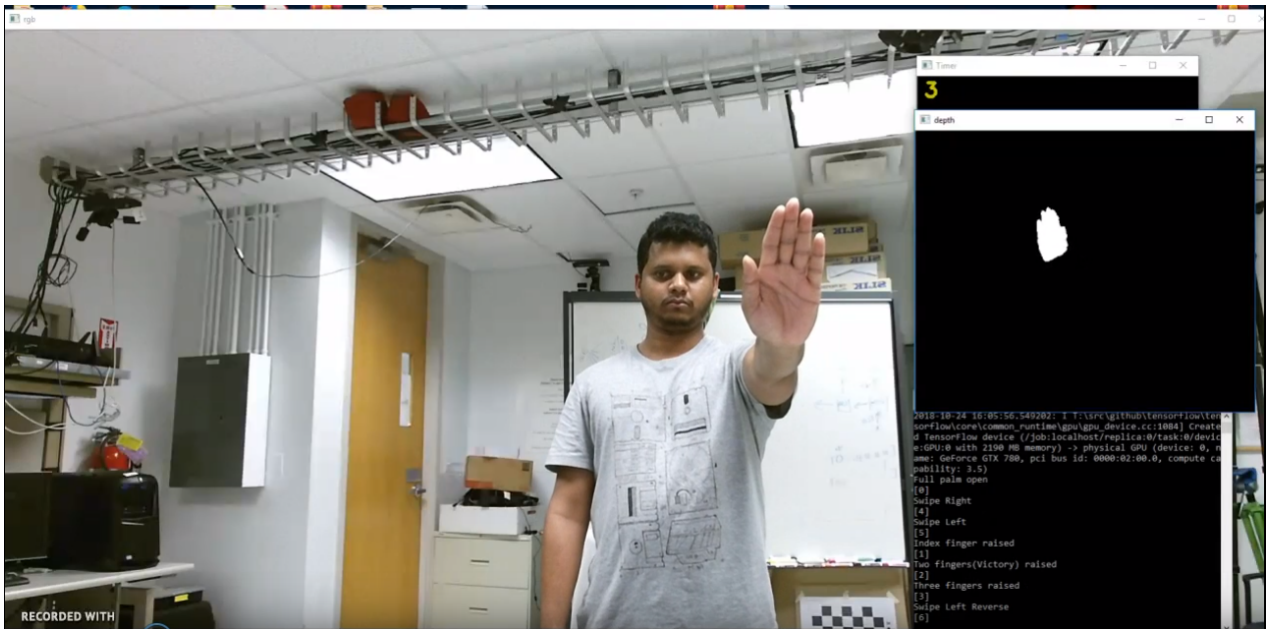


Figure 3.50: Live system test class 6.

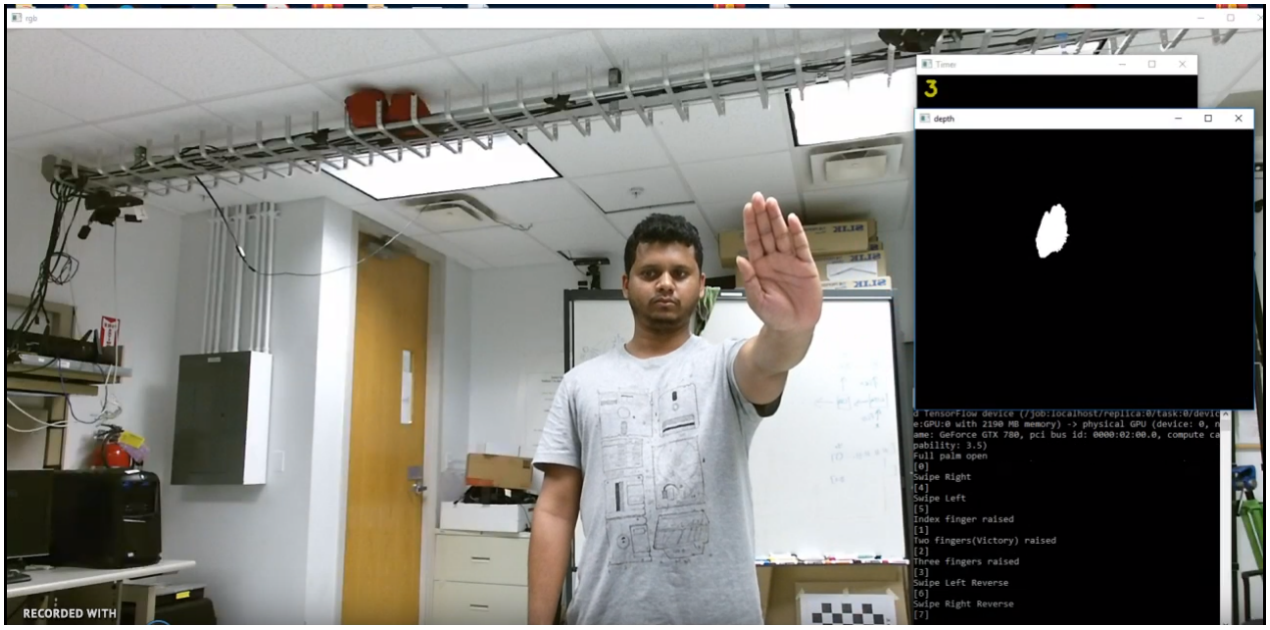


Figure 3.51: Live system test class 7.

CHAPTER IV: CONCLUSION

We presented a novel, computationally efficient hand gesture recognition system using temporally blended image data. A Kinect depth sensor was used as a source for the depth and the body data raw input frames. A major contribution was blending the frames using the temporal feature. This contributed to reduction of the layers used in the design of CNN. We also designed an optimal and efficient CNN creating eight different model versions and observed the behaviour of each model to obtain and estimate the most generalized model. The computational efficiency of the model was observed to increase with an empirically evaluated size of 50×50. Live system testing was performed and the system achieved precise classification of the gestures. System was further tested to control the music player of the computer using the hand gestures. It was observed to accurately control the music player with a good accuracy. System failure was observed when tracking of the Kinect suffered inherent hardware failures.

Recognizing gesture was an integral part which was performed with a convolutional neural network (CNN). We generated a dataset using the Kinect depth sensor. A total of 2000 blended frames were generated per class, which summed to $16,000$ frames. An appropriate, efficient and optimized CNN model was designed to have eight layers. This model was achieved after generating eight models having different configurations and measuring the performance and behaviour of the accuracy and the loss of each model. It consisted of three convolution layer of sizes $64 \times 7 \times 7$, $64 \times 5 \times 5$ and $128 \times 3 \times 3$, each followed by a max pooling layer with a stride of two and a dropout of 25%. The last two layer were the fully connected layer having 256 and 8 units of neurons. The first fully connected layer was followed by a dropout of 50% to reduce overfitting in the model. The dataset was split randomly into train-to-test ratio of $60 : 40$ and the accuracy and loss was observed for each model design. The split of $70 : 30$ was also used to test the accuracy and loss for each model design. The models using $70 : 30$ ratio had overfitting issues monitored using the accuracy and loss hyperparameters. We observed that the CNN model version V4 was the most efficient and optimized, therefore, it was selected for classification. Cross-validation was performed using 5-fold cross validation on

the V4 CNN model. The performance after cross validation displayed an overall generalization of the model. The optimized model achieved a mean training accuracy of 97.92%, a mean validation accuracy of 97.87%. We obtained test data accuracy 98.45%. This helped us conclude that the model was generalized well to work on real world data. Since this is just one of the criteria to observe if the model is generalized, we performed more analysis.

An additional and necessary analysis of the model was perfomed using metric analysis in quantitative measures of confusion matrix, Receiver Operating Characteristics (ROC) and the precision-recall graph for all the eight classes. We observed that the model was a good fit and generalized to predict on a real unbiased data.

The final process in the system pipeline design was to merge the system to make it online as HGRS to read gestures from users to control computer system functionality. HGRS has a lot of scope in terms of usage and control of systems.

CHAPTER V: FUTURE WORK

This research was implemented as a proof of concept with the image processing techniques and deep learning as main foundations in the recognition of hand gestures.

Blending of frames was performed using the intensity changes in the ascending order for each consecutive frames. This work considers temporal aspect of the frame that is, each frame is different than the previous frame based on time. A future work would be implementing the spatial feature to detect gestures using the position of the hand in the frame independent of the temporal feature. When considering a spatial feature it may not be relevant to use cropping so as to maintain the aspects of spatial features. Another area of research would be upsampling of the frame rate when the number of frames in the gesture is less. Up sampling may improve the features in the images by adding more frames and therby increasing the image details.The process of upsampling would increase the frame rate when less frames are generated.

We implemented a CNN model to classify eight gesture classes. A first direction of future work would be to train a CNN model that can distinguish more than eight classes. The fact that the CNN model uses only eight classes does by no means imply that it is not generalizable to more number of classes. Another direction of future work would be creating more complex hand gestures using both hand and testing the accuracy of the system. Another approach for future research is to train our model on a much larger dataset hoping that the benefits of generating a start-of-the-art model.

We used blended frames to detect gesture, a field of future work would be to combine the solutions with speech as a form combined interaction with the systems. Since speech is also another form of communication for simple control of systems.This would make the system a multi-task classification. Future research may also focus on three-dimensional gestures to improve the accuracy of the system. These three-dimensional gestures may a point cloud of generation of three dimensional hand models. The other way around would be to combine the hand gesture with wearable devices such as arm bands and vision-based device such as Leap Motion to improve the accuracy with real world problems. Another plan would be to evaluate the system on a larger set of gestures, collected from multiple

users. Such a dataset will also enable quantifying how good a system is trained for one user versus multiple users. A practical deployment of our approach would also benefit from allowing the user to label a misclassified gesture and provide it as an online training example, in a never-ending learning version of the system.

We use offline augmentation technique to improve the size of the dataset, this augmentation technique is balanced when the data set is small,it would be very interesting to use a larger dataset and implement online data augmentation techniques which is performing data augmentation on the mini batches of data that we input to the model during training of the network.

REFERENCES

[1] K. S. Abhishek, L. C. F. Qubeley, and D. Ho. Glove-based hand gesture recognition sign language translator using capacitive touch sensor. In *2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pages 334–337, Aug 2016.

[2] H. Alabbasi, A. Gradinaru, F. Moldoveanu, and A. Moldoveanu. Human motion tracking amp; evaluation using kinect v2 sensor. In *2015 E-Health and Bioengineering Conference (EHB)*, pages 1–4, Nov 2015.

[3] Javier Orlando Pinzón Arenas, Paula Catalina Useche Murillo, and Robinson Jiménez Moreno. Convolutional neural network architecture for hand gesture recognition. In *Electronics, Electrical Engineering and Computing (INTERCON), 2017 IEEE XXIV International Conference on*, pages 1–4. IEEE, 2017.

[4] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Communications of the ACM*, 36(7):28–35, 1993.

[5] Gary Bradski and Adrian Kaehler. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 3:1– 81, 2000.

[6] Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang, and Yu-Bo Yuan. Real-time hand gesture recognition using finger segmentation. *The Scientific World Journal*, 2014, 2014.

[7] François Chollet et al. Keras, 2015. [Online; accessed 16-October-2017].

[8] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[9] Wikipedia contributors. Conducting — Wikipedia, the free encyclopedia, 2018. [Online; accessed 11-January-2018].

[10] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[11] Suraksha Devi and Suman Deb. Low cost tangible glove for translating sign gestures to speech and text in hindi language. In *Computational Intelligence & Communication Technology (CICT), 2017 3rd International Conference on*, pages 1–5. IEEE, 2017.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[13] Ali Erol, George Bebis, Mircea Nicolescu, Richard D Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007.

[14] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] Y. Hu and Z. Chen. Kinect-based face recognition and its application in moocs production. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 298–301, Aug 2015.

[17] Lih-Jen Kau, Wan-Lin Su, Pei-Ju Yu, and Sin-Jhan Wei. A real-time portable sign language translation system. In *Circuits and Systems (MWSCAS), 2015 IEEE 58th International Midwest Symposium on*, pages 1–4. IEEE, 2015.

[18] Adam Kendon. Gesture and speech: How they interact. *Nonverbal interaction*, 11:13–45, 1983.

[19] G. Drew Kessler, Larry F. Hodges, and Neff Walker. Evaluation of the cyberglove as a whole-hand input device. *ACM Trans. Comput.-Hum. Interact.*, 2(4):263–283, December 1995.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Rick Kjeldsen and John Kender. Finding skin in color images. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 312–317. IEEE, 1996.

[22] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *European Conference on Computer Vision*, pages 189–196. Springer, 1994.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] T. C. W. Landgrebe and R. P. W. Duin. Efficient multiclass roc approximation by decomposition via confusion matrix perturbation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):810–822, May 2008.

[25] Hyeon-Kyu Lee and J. H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, Oct 1999.

[26] T. Liu, W. Zhou, and H. Li. Sign language recognition with long short-term memory. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2871–2875, Sept 2016.

[27] Myo Systems LLC. Myo systems home page, 2017. [Online; accessed 02-August-2017].

[28] Loretta A Malandro and Larry Lee Barker. *Nonverbal communication*. Addison Wesley Publishing Company, 1983.

[29] Charles E. Metz. Basic principles of roc analysis. *Seminars in Nuclear Medicine*, 8(4):283 – 298, 1978.

[30] Microsoft. Microsoft kinect online api reference, 2017. [Online; accessed 20-March-2017].

[31] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[32] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[33] GRS Murthy and RS Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[34] Farid Parvini, Dennis McLeod, Cyrus Shahabi, Bahareh Navai, Baharak Zali, and Shahram Ghandeharizadeh. An approach to glove-based gesture recognition. In *International Conference on Human-Computer Interaction*, pages 236–245. Springer, 2009.

[35] P. Pławiak, T. Sośnicki, M. Niedźwiecki, Z. Tabor, and K. Rzecki. Hand body language gesture recognition based on signals from specialized glove and machine learning algorithms. *IEEE Transactions on Industrial Informatics*, 12(3):1104–1113, June 2016.

[36] Judy Pearsall. *The oxford encyclopedic english dictionary*. Oxford University Press, USA, 1995.

[37] T. T. D. Pham, H. T. Nguyen, S. Lee, and C. S. Won. Moving object detection with kinect v2. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4, Oct 2016.

[38] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 22(10):1090–1104, 2000.

[39] Francis KH Quek. Toward a vision-based hand gesture interface. In *Virtual reality software and technology*, pages 17–31. World Scientific, 1994.

[40] Francis KH Quek. Eyes in the interface. *Image and vision computing*, 13(6):511–526, 1995.

[41] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

[42] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, pages 8614–8618. IEEE, 2013.

[43] Eduardo D Sontag. Vc dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, 168:69–96, 1998.

[44] Helman I Stern, Juan P Wachs, and Yael Edan. Human factors for design of hand gesture human-machine interaction. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 5, pages 4052–4056. IEEE, 2006.

[45] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from rgbd images. *plan, activity, and intent recognition*, 64, 2011.

[46] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

[47] Kai Ming Ting. *Precision and Recall*, pages 781–781. Springer US, Boston, MA, 2010.

[48] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[49] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[50] Pei Xu. A real-time hand gesture recognition and human-computer interaction system. *arXiv preprint arXiv:1704.07296*, 2017.

[51] Jianyu Yang, Chen Zhu, and Junsong Yuan. Real time hand gesture recognition via finger-emphasized multi-scale description. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 631–636. IEEE, 2017.

[52] B. Yu, Y. Chen, Y. Huang, and C. Xia. Static hand gesture recognition algorithm based on finger angle characteristics. In *Proceedings of the 33rd Chinese Control Conference*, pages 8239–8242, July 2014.

[53] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.