KULLIYYAH OF INFORMATION

AND COMMUNICATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

FINAL YEAR PROJECT REPORT

REAL-TIME SIGN LANGUAGE DETECTION MODEL BASED ON HAND GESTURE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

HADIL ABDULHAKIM QASEM
1414620

SUPERVISED BY

DR. AKEEM OLOWOLAYEMO
ASSISTANT PROFESSOR

JUNE 2021
SEMESTER 2 2020/2021

FINAL YEAR PROJECT REPORT

REAL-TIME SIGN LANGUAGE DETECTION MODEL BASED ON HAND GESTURE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

by

HADIL ABDULHAKIM QASEM

1414620

SUPERVISED BY

DR. AKEEM OLOWOLAYEMO
ASSISTANT PROFESSOR

In partial fulfillment of the requirement for the

Bachelor of Computer Science

Department of Computer Science

Kulliyyah of Information and Communication Technology

International Islamic University Malaysia

JUNE 2021

Semester 2 2020/2021

ACKNOWLEDGEMENTS

Praise and thanks to Allah first and foremost whose blessing enabled me to accomplish this project.

In the name of Allah, The Most Gracious and The Most Merciful, First and foremost, I would like to present my heartiest gratitude to Allah S.W.T for the strength, health and perseverance that HE gave me to complete my final year project and report, Real-Time Sign Language Detection Model Based on Hand Gesture Recognition Using Convolutional Neural Network.

I would like to express my most profound gratitude and appreciation towards my supervisor, Dr. Akeem Olowolayemo for accepting me as his student, the patience, support and his continuous guidance and constructive encouragement throughout the project. His approach in always using positive and encouraging words are truly inspirational.

I would like to thank to all my dear great lecturers who have taught me since my first year in IIUM. The knowledge I gained since the first year really helped me a lot to accomplish my project. I would like to dedicate my special thanks to my FYP coordinator of the Final Year Project, Dr. Norzariyah Binti Yahya for her help, support and assistance.

A special thanks to my mother and father, family, fellow friends who supported me morally and also to solve various problems during this project development, for their cooperation and assistance during my studies. Our sincerely thanks to all those whom directly or indirectly help us and gave courage and support throughout our studies in IIUM and complete this project your kindness means a lot to me. May Allah bless and reward all of you for your uncountable support and kindness.

ABSTRACT

A hand gesture recognition system is proposed to help the normal people communicate better with deaf community. The mission of this system is to make American Sign Language available for everyone. I will work with computer vision and deep learning using CNN for hand gesture recognition that can help to detect sign language. For this project I will be classifying each hand gesture into one of the numbers from 0 to 9 in American Sign Language (ASL). To address this problem, a deep learning model is proposed that is based on CNN that uses the weights of samples to train multiple labels. Background subtraction, thresholding and contours will be used to help in the detection of the hand gesture.

Index Terms—Convolutional Neural Network, Computer Vision, Machine Learning, Hand Gesture Recognition, Deep Learning, Sign Language, American Sign Language, OpenCV-Python, Segmentation, Thresholding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

HCI	Human Computer Interaction
HGR	Hand Gesture Recognition
ASL	American Sign Language
CO	Convolutional Operation
CNN	Convolutional Neural Networks
FC	Fully Connected *layers*
ReLU	Rectified Linear Unit
RGB	Red Green Blue (3-channeled) images
DNN	Deep Neural Networks
AI	Artificial Intelligence
RBM	Restricted Boltzmann Machines
2D	Two Dimensional
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ROI	Region of Interest

CHAPTER ONE

INTRODUCTION

1.1 Overview

Humans use gestures to communicate non-verbally by moving their hands and faces. As hands are considered an important part of the body, therefore important information can be delivered using hands movements only, such us marshallers whom their task requires visual communication with the aircraft pilot and signaling the pilot to follow specific orders. Hand gesture recognition (HGR) works on classifying specific features and gestures of the hands. HGR is a way to interpret human hand gestures by using specific mathematical and machine learning algorithms. HGR is used widely in the field of Human Computer Interaction (HCI). It facilitates the communication of humans with machines without the intervention of mechanical physical input devices. Variety of different commands can be given to the machine to perform specific tasks by using HGR. It can be conducted using computer vision and image processing. The problem of detecting hands gestures and movements and interpreting the meaning of them using machines is becoming huge demand in applications such as sign language translation, remote controlling devices, and more. Gestures of the hand can be conveyed through several means that include the center of the palm, position of the finger and the hand shape. Those gestures are categorized into static gestures which indicates stable shapes of the hand not moving, and dynamic gestures including a sequence of hand movements. How to approach the problem of detecting the gestures of the hand is by glove-based and camera vision-based sensor approaches which are two important approaches that are being used in the field of HGR. The glove-based approach was the first approach to be used in HGR in its early days that it requires a wire glove to be connected physically to the machine and this glove contains sensors to provide the coordinates and motion of the hand, however this approach has many drawbacks due to the ease of interaction between human and machine as well is these wire gloves are quite pricy (O. Munir, 2020) whereas the camera vision-based sensor approach is becoming more commonly used because it doesn’t require a direct physical contact with the machine however this approach presented challenging aspects such as cluttered background, rotation, and occlusion. In this paper background subtraction method will be used to differentiate the background from the foreground which is the hand.

Sign Language is one of the oldest ways of communication between the deaf and blind community. It is a language where two hands are being. Sign language relies on the usage of hand where each sign indicates a letter or a number and by moving the hand a sequence of sign language letters form a sentence. Since there is a big gap between the deaf and hearing communities as well as deaf people are participating in the society more than ever, a solution is needed to help those people engage in the society comfortable. The computer vision plays an important role in solving this problem through the algorithms of HGR. In this paper we will be working on detecting American Sign Language (ASL) numbers from 0 to 9 as shown in figure 1.

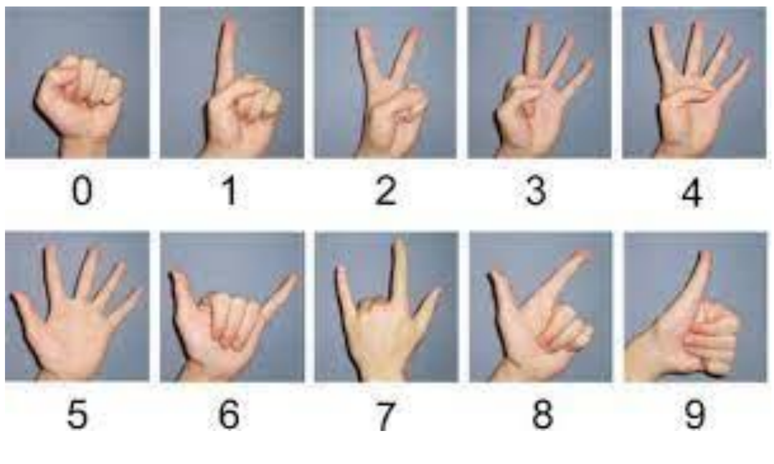


Figure 1: American Sign Language Digits from 0 to 9

Disabled people may lost an ability like hearing, talking or walking, but many of them are smart and excel in life if they are given opportunities and are involved more in the society with hearing people. According to Rebecca Carol (2015), the vast majority of humans receive information auditorily, however this is not available for deaf disabled people. Since deaf people cannot hear, so when they communicate with hearing people, they reply on reading hearing people's lips or body language. However, lips and body posture reading may not be accurate in many cases. Therefore, deaf people find it tremendously difficult to communicate with hearing people. On the other hands, hearing people are not willing to learn sign language as they might need to use it quite often and as a result of that they find it useless and a waste of time and efforts to learn it. This is because deaf people have not been effectively involved and treated normally in the hearing society. Nowadays, many restaurants and companies realized that those specially-abled people are intelligent and hardworking despite their disabilities and they even perform better than the hearing people.

1.2 Problem Statement

Many restaurants and public places are involving deaf people and treating them similar to hearing people. An example for this, Kentucky Fried Chicken restaurants are hiring specially-abled staff as shown in Figure 2.

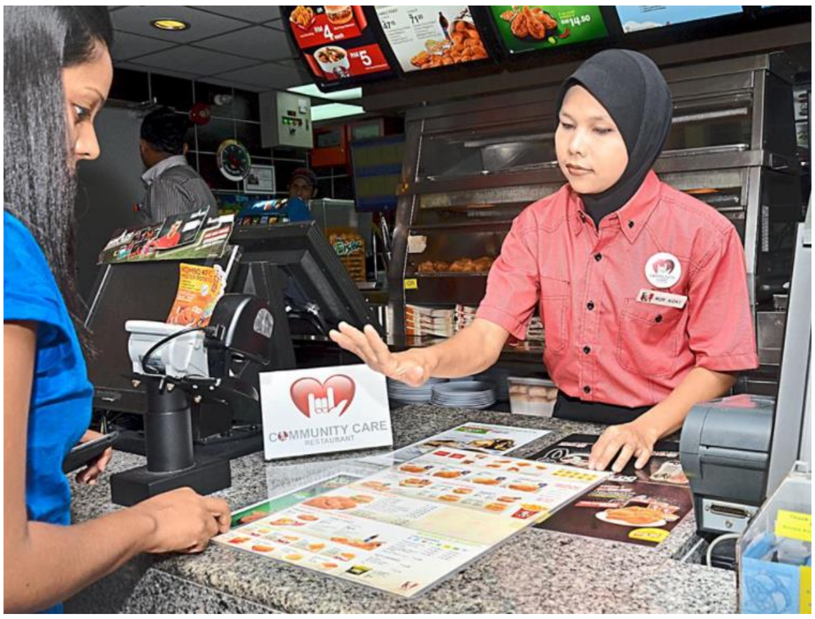


Figure 2: Deaf and mute staff working at Kentucky Fried Chicken
("Disabled shine at work", 2015)

Technology exists to ease life for humans. However, learning sign language for hearing people requires tremendous efforts as it requires knowledge in grammar and constructing meaningful visual sentences. Not everyone can learn another language easily since people have different learning styles and learning difficulties. Therefore, with the help of intelligent technology and smart systems, the communication between deaf community and hearing community can be eased by using machine learning and intelligent systems.

1.3 Project Objectives

- To translate sign language to hearing people without their need to learn sign language.
- To make sign language available for everyone.
- To use computer vision newest technologies for object detection
- To apply machine learning algorithms to classify different sign language numbers.

1.4 Project Scope

In this project, a one hand will be detected based on HGR model to determine the ASL number corresponds to the hand gesture. The algorithm that will be used is Convolutional Neural Network (CNN) for prediction. Advanced computer vision libraries like OpenCV python and Keras will be used also to ease this project progression.

1.5 Significance of the Project

This project will change the way people can communicate with deaf people and vice versa. It can be used in restaurants, customer service centers, transport stations, emergency cases, and instant message apps. This project can also be used with different sign languages like the Arabic, Malay, Chinese and more.

CHAPTER TWO

2. REVIEW OF PREVIOUS WORKS

2.1 Deep Learning

Machine learning is considered the base of nowadays advanced intelligent computers and other electronic devices. It uses prediction methods and techniques that can learn from existing training datasets and predict the future behaviours, outcomes and trends. A sub-field of machine learning is deep learning that uses methods inspired by the brain of the human being and these methods are expressed mathematically. Deep Neural Networks (DNN) is an emerging approach that has been widely applied in artificial intelligence (AI) approaches such as computer vision, natural language processing and many more. Five important categories of DNN include the following: Convolutional Neural Networks (CNN), Restricted Boltzmann Machines (RBM), Autoencoders, Sparse Codes, and Recurrent Neural Networks (RNN). CNN considered the most used types of DNN. (Gholamalinezhad and Khosravi, 2020). Due to the prevalence of touchless applications and rapid growth of hearing-impaired populations, it is particularly important to have sufficient hand gesture recognition systems. Translating sing language is one of the most significant applications of hand gesture recognitions based on computer vision. An efficient recognition system should be able to consider the global configuration of the hand including hand’s orientation and relative position to the body as well as the local fingers’ configuration (Muneer et al., 2020). The main aim of real-time HGR is to classify and recognize the gestures of the hands. Hand recognition can be defined as a technique in which different algorithms and concepts of various techniques, such as image processing and neural networks, in order for us to understand the movements and gestures of the hand (Abdullah M. et al., 2021).

2.2 Convolutional Neural Networks

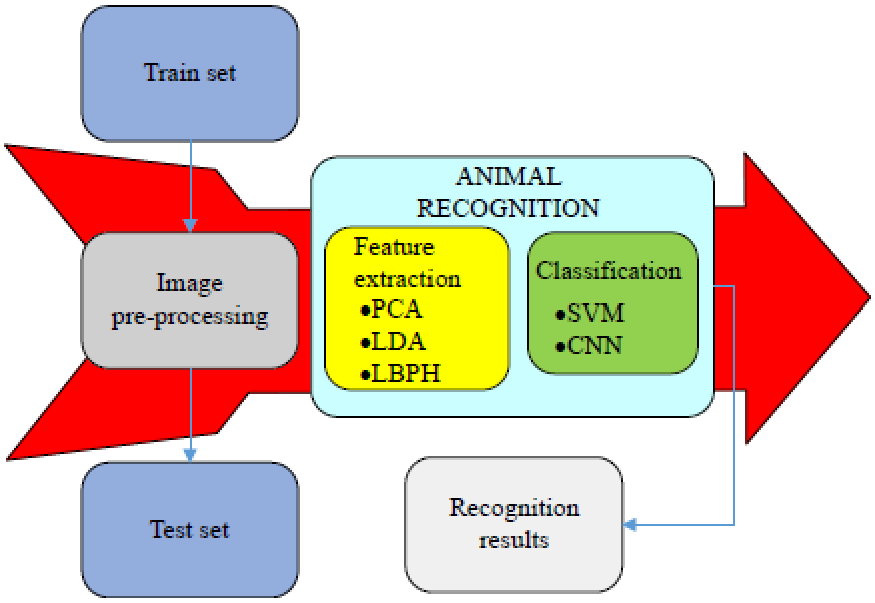


Figure 3: Animal recognition and classification system (Trnovszky T et al., 2017)

Convolutional neural networks (CNN) foundation evolved from the discovery of how mammals' visual cortex cells recognize the light in the small receptive field. Figure 3 shows that the animals recognition system is divided into 3 main sections: image pre-processing section, feature extraction and classification section where each section is responsible for certain tasks. In pre-processing section, the training and or testing dataset will be received and pre-processed. These data will be received as either in RBG or Gray-scale format and some pre-processing techniques will be applied to minimize the effects of factors that can influence the recognition algorithm of the animal eyes. The feature that will be used in the recognition phase will be computed based on some techniques. On the other hand, classification section is the trainable section and will include learnable algorithms such as CNN that will use the features in the training dataset to predict the features in the test dataset to estimate their class labels. (Trnovszky et al., 2017). CNN architecture developed through time with massive and intensive rework and advancement starting with the early version lenet-5, followed by alexnet, vgg-16, inception-v1, inception-v3, resnet-50, xception, inception-v4, inception resnets, and to the latest version which is resnext-50 that was released in 2019. CNN is widely used in deep learning complex tasks from image process, object detection to pattern recognition. It is being observed that the systems using CNN have shown astonishing performance in HCI. This is because CNN architecture consists of multiple layers, this type of architecture was designed for automating feature learning, hence data are represented through a hierarchy of features arranged from low to high levels (Tran et al., 2020). CNN structure generally consists of two general layers that are feature extraction layer and feature map layer. CNN is composed of multiple layers that are input layers, convolution layers, pooling layers followed by fully connected (FC) layers and lastly output layer. The convolution and pooling layers are considered the feature extraction layers whereas the FC layer is the classification layer, therefore with CNN those two layers can learn together concurrently. The most important and leading processes in CNN are convolution operation and sampling (Liu T. et al., 2015).

2.2.1 Convolution Layer and Convolutional Operation

Starting with convolution layers are designed to extract the important features of the input image. It consists of learnable filters (kernels); these filters are simply a set of matrices of values or weights that are trained to detect edges that are present in an image. The weights of the filters are assigned to random values at first before the training of CNN starts, then these weights are updated after each training epoch. Filters carry out a Convolution Operation (CO) which is element-wise product and summation between two matrices: the filter matrix and matrix a portion of input image. The result of the CO is stored in matrix to generate an output feature map. If input image feature matches and is part of the image (training image in the dataset); then CO between filter and input image matrices will equal to a real number with high value; otherwise, the value is a low real number. The result of CO is stored in a convolved feature matrix. CO will stop until the filter matrix can no longer slide further. Filters moves over the input image by a stride value to check if the feature is present. This stride value dictates by how much the filter should move at each step of the CO process. Before we do CO, zero-padding is applied to get an efficient feature map results. Zero-padding will protect the features that are located near to the border side of the input image from being washed away through the process of filter matrix scanning the input image and performing the OC. By applying the zero-padding to the input image, this will increase the size of the input image and therefore both the size of kernel matrix and feature map matrix will increase. The following Figure 4 summarize the padding process and how the feature map output was affected:

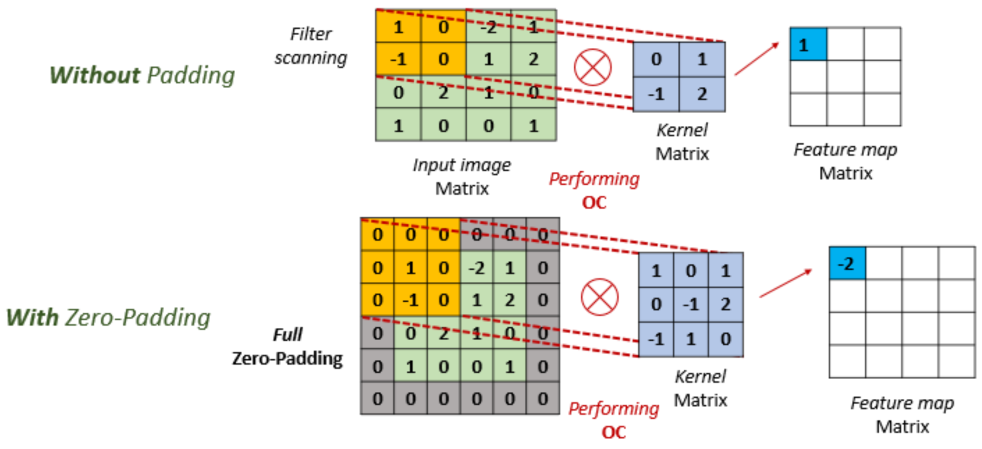


Figure 4: Zero-padding and Convolution Operation

With full zero-padding that covers all the borders of input image, the kernel will scan the input image from end-to-end during the CO which will result in a more sufficient feature map.

But before moving to Pooling layer, a non-linear activation function will be applied. This activation function will take the summation of CO convoluted matrix (output feature map) and bias term. The result of summation will be passed to the activation function to introduce non-linearity to the CNN network. Activation functions most applied to the learnable layers which are convolutional layers and fully connected layers. The activation function will decide whether a neuron will be activated or not, depends on input to define the output neuron. The bias term can be considered as a vector which is an additional set of weights that does not require an input. Each feature map has its own bias term, the bias will be added to the feature map to produce the final output neuron. The following graph simply explains the different types of activation functions commonly used in CNN:

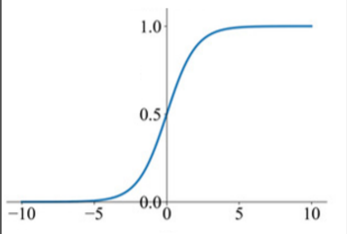
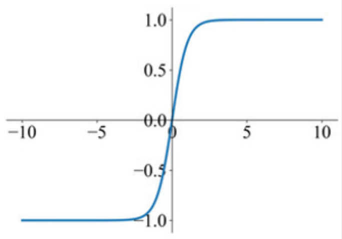
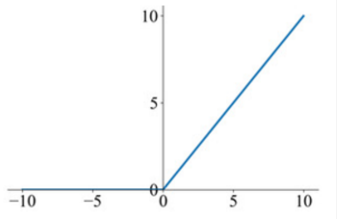
Sigmoid - Input (x-axis): real numbers - Output (y-axis): [0,1] - Perfect to be used with models that their output is the prediction of probability.	**Mathematical Representation** $f(x)_{sigm} = \frac{1}{1+e^{-x}}$	
Tanh - Input (x-axis): real numbers - Output (y-axis): [1,-1] - Perfect for classification between two classes.	$f(x)_{tanh} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU **Rectified Linear Unit** - Input (x-axis): real numbers - Output (y-axis): [0,+∞] - Commonly used in CNN, less expensive and simpler mathematically	$f(x)_{ReLU} = \max(0,x)$	

Table 1: Non-Linear Activation Functions Types

The ReLU activation function is most used in CNN compared to other functions as it requires less computational power. Most of the activation functions are differentiable, however the ReLU is not differentiable at x=0 even though it is widely used in CNN. This differentiable feature allows the model to be trained by error backpropagation. Backpropagation or backward propagation of errors can be defined as a standard method to train the CNN using gradient descent, it computes the gradient of the error function with respect to the weights of the neural network. In nowadays neural networks, backpropagation is an initial component of all neural networks. Through convolutional and max pooling layers, CNN networks weights need to be updated and optimized by certain backpropagation gradients (Ghosh A. et al., 2020).

2.2.2 Pooling Layer and Pooling Operation

Once the OC is completed, there will be multiple feature map matrices as a result of OC completion, therefore each feature map matrix will be sub-sampled to smaller-sized matrices. Pooling layer is inserted after a convolutional layer, and it works by reducing the size of feature maps and network parameters. This layer is considered a key-step in convolutional based systems as it reduces the dimensionality of the feature maps. It works by combining a set of values into a smaller number of values, therefore by reducing these values the dimensionality of the feature map reduces as well. (Gholamalinezhad AND KhosravI, 2020). By performing pooling on an image, a portion of the input image will be looked at to perform either one of the following pooling operations: average pooling which is getting the average value or max pooling which will take the maximum value. Max pooling region will be scanned over the input image based on a stride value; at each step, the maximum value is pooled into an output matrix. Max pooling is widely used compared to average pooling because it preserves the detected features. As it can be seen in Figure 3, the Max pooling is performing well, and the output matrix seems better than the average max pooling one.

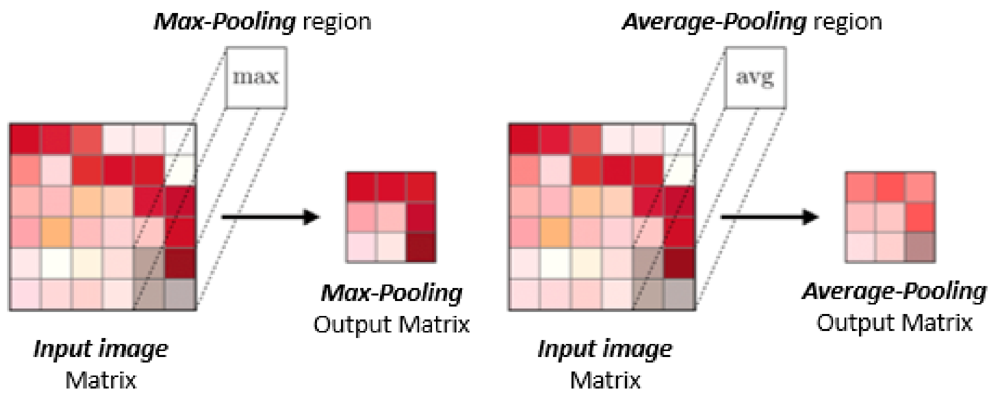


Figure 5: Comparison between Max & Average Pooling Operations

Max pooling reduces the representation size; hence this will reduce the amount of memory needed by CNN and the number of operations that will be done later in the CNN's other layers. The main purpose of pooling or in other words downsampling is to accelerate the training process and reduce the amount of memory consumed by the network. Max-pooling can lead to faster convergence rate as it selects the superior invariant features and this will result in improving the generalization performance (Nagi J. et al., 2014). Max pooling output matrix consists of the most important dominant features that will be needed in the next neurons of the CNN network, however this may create a disadvantage of the pooling process as it may decrease the CNN overall performance because pooling layer does not care whether the feature that was presented is in the correct position or not, it focus whether the feature is present Sampling process happens here through pooling steps by using activation function. Sub-sampling by time or space to reduce size of training parameters. CNN are designed to be spatially invariant (they are not sensitive to the position of object in the picture). Max pooling suppress and conceal the noise activation at the same time as it reduces the dimensions of the feature map matrices. The depth of the convolution and pooling layers depends on the complexities presented in the input image features creating an increase in the number of low-level layers, however this will require more computational power. At this stage of the neural network, the CNN understand the features of the input image.

2.2.3 Fully Connected Layer

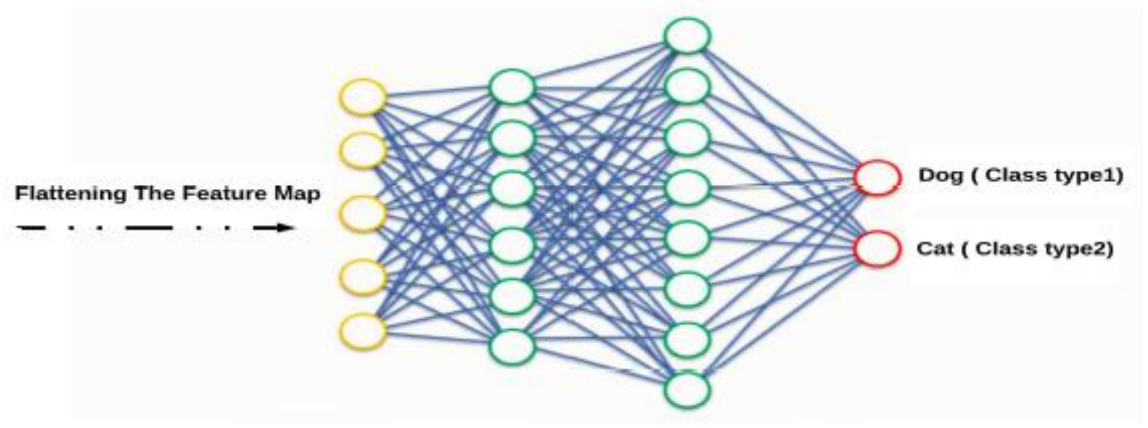


Figure 6: Fully Connected layers architecture

The last layer of the CNN is responsible of the classification process, it consists of the Fully Connected (FC) layers, these layers receive the input of flattened previous feature maps. The rows of the feature map matrix will be concatenated to form a flattened feature vector. Multiple dense layers will receive the feature vector, the feature vector will be multiplied by the dense layer weights, the result will be summed with a bias term, and all together will be passed to a non-linear activation function. The purpose of FC is to collect information from final feature maps (after convolution and pooling) and generate final classification output layers. There can be many layers for the FC, and each layer is responsible for certain tasks. If the task is to aggregate information, then ReLU activation function is used. The dimensions of the previous layer will be gathered to be used in the process of flattening the previous layer into FC layer. Then these FC will be passed to multiple dense layers of neurons to produce raw predictions. Otherwise, to produce final classification output layers, Softmax activation function will be used to generate output probability. The final output of the dense layers is passed to the Softmax activation function that will result a probability vector that if summed will be equal to one.

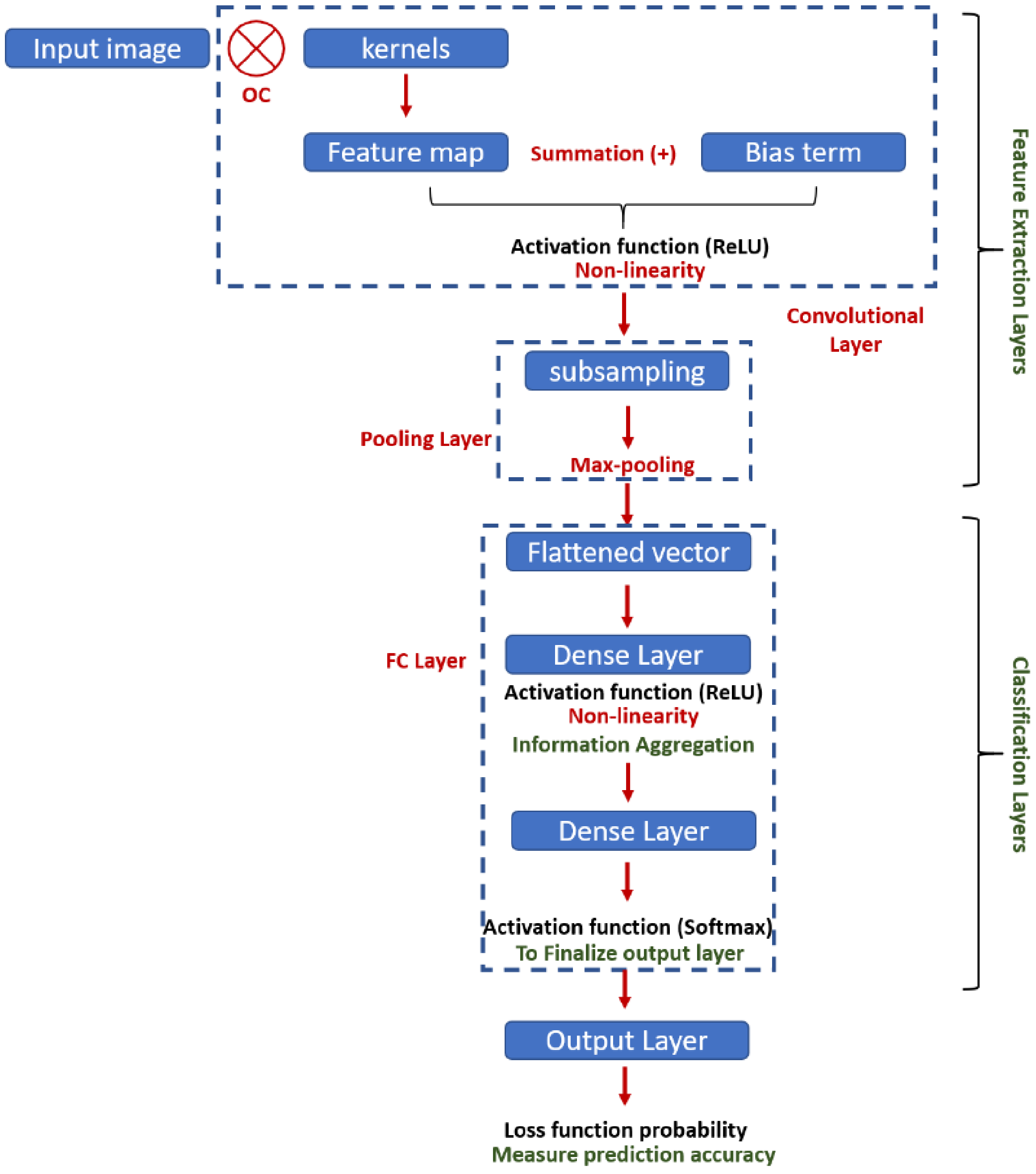


Figure 7: Summary of CNN layers and operations

CHAPTER THREE

3. METHODOLOGY

The objective of this proposed system is to recognize in real time the gesture of the hand and determine the corresponding ASL number from 0 to 9 with the help of computer vision and deep learning using OpenCV-Python which is an open-source computer vision library.

3.1 Background Subtraction

Segmenting foreground from background is called background subtraction. In our project the hand is the foreground and anything else is considered the background. The camera is the eye to the computer, and computers are not smart enough compared to human to differentiate easily different objects in a frame. A video is a sequence of frames, therefore by computing accumulated weighted average of current frame with the previous frame the computer can figure out the background model from the hand object. In our project, the background was static not moving to ease the process of differentiating it from foreground. Once the running average is computed over the background model and current frame, the weighted average is computed and accumulated and then update the background based on the following formula dst(x,y) = (1-a).dst(x,y) + a.src(x,y) where src(x,y) is the input image RGB image that contains 3 channels, dst(x,y) is the output image and a is the weight of the input image. Therefore, the running average of the 30 frames is the background. After getting the accumulated average of the background model, then the absolute difference will be computed to subtract the background from every frame to find any object that covers the background model. Now, once the foreground is being differentiated from the background, we will create a Region of Interest (ROI) which is a square to help the computer focus only on a specific region where we will place the hand.

3.2 Gray-scaling and Gaussian Blurring

After detecting and differentiating the background model from the foreground which is our hand. Now the RGB hand foreground frame needs to be converted into grayscale image using the function:

cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

The reason to grayscale the image is to help with edge extraction which will be later used in finding contours of the hand. Gray scale images are single channel images which can lead to better and faster performance in detection since they provide less information per each pixel and they are considered less expensive computationally. Once the image of the hand is in grey scale, then a gaussian blur will be applied to reduce the noise and smooth the image to ease the process of thresholding (M. Siddharth and W. Yaokun, 2020).

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 8: Gaussian Blur formula

The function that was used to perform the gaussian blurring:

cv2.GaussianBlur(gray_frame, (9, 9), 0)

3.3 Thresholding

Thresholding can be defined as the assignment of pixel intensity to 0 or 1 based on a particular threshold level so that the object of our interest which the hand is captured alone from the frame and everything else is marked with a 0. Thresholding involves masking all pixels of the image that we not interested in with 0 which is equivalent to black and all the other regions with white. But before applying thresholding, the image will be taken from the webcam in RBG (Red, Blue, Green) and needed to be converted into greyscale, then a simple threshold must be applied to convert the image into binary image consisting of 0 and 1 where 1 is the object of interest and 0 is the background.

$$J(x,y) = \begin{cases} 0 & \text{if } I(x,y) < T \\ 1 & \text{otherwise.} \end{cases}$$

Figure 9: Thresholding or Image binarization formula

The formula in figure 2 returns the value of the thresholded binary image if the input image is less than the threshold value then it is mapped by 0 = black, otherwise if it greater than the threshold value then it is mapped by 1=white, as it can be seen in the following figure 3.

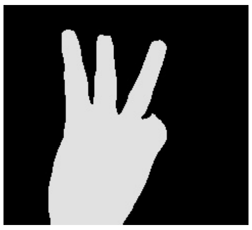


Figure 10: Thresholded binary image for ASL digit 6

The following are all the digits from 0 to 9 represented as thresholded binary images.

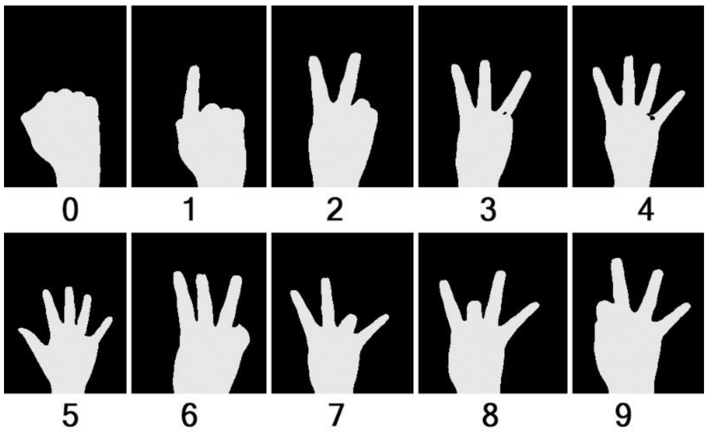


Figure 11: ASL digits from 0 to 9 as binary images

3.4 Contours and Motion Detection

Within the Region of Interest (ROI) we can detect the hand edges which are the contours. Contours can be explained simple as a curve joining all the continuous points along the boundary having same colour or intensity. The contours are useful tool for shape analysis and object detection and recognition. For better accuracy, using binary image is the best way to get efficient contours. The contours in our project represents the hand. In OpenCV finding contours is like finding white object from black object that is why the thresholding process is crucial for contour finding. After performing the thresholding.

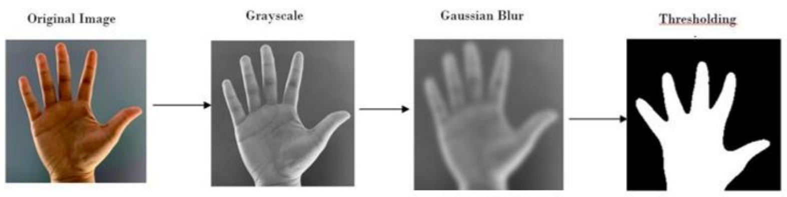


Figure 12: Hand Segmentation Process

3.5 Dataset

In this proposed system, the dataset was collected through the live feed of the video camera in real time with the help of OpenCV. Each frame that is captured by the webcam is stored in a specific directory that matches the ASL digit. For each ASL digit 300 frames where collected, resulting in total of 2400 dataset for the ASL numbers from 0 to 9.

3.6 A Brief Review of CNN

Even though CNN accepts input images of multi-channelled such as RGB (3-channeled image), however with a gray-scale which is a single channelled image, CNN will be faster in detecting the main features and achieving better results. Since gray-scale images represents less information compared to RGB.

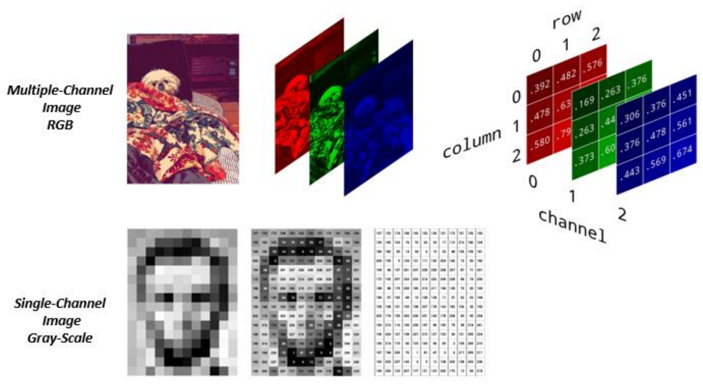


Figure 13: RGB and Gray-Scale Image Analysis

Therefore, we used gray-scale images and we thresholded them to be used in training the CNN. The CNN algorithm in general, consists of 3 main layers which are convolutional, pooling and fully connected layers. The first two layers will work on extracting the features from the input imagen and the FC layer will produce the output classification result. In both convolutional and FC layers, the weights will learn and therefore applying activation functions is very necessary at these two layers. Convolutional layer will perform convolution operation and sum the result with bias term to be passed as an argument to a non-linear activation function. Then, downsampling will be performed on the feature map. Max-pooling will be used to get the maximum value of the sub-sampled feature map. The feature map is then flattened and passed to dense neurons in the FC. Based on the hand gesture input image that was detected, the system will predict whether it matches an ASL number in the dataset or not.

The general model of hand gesture detection combined with CNN algorithm can be summarized in the following figure.

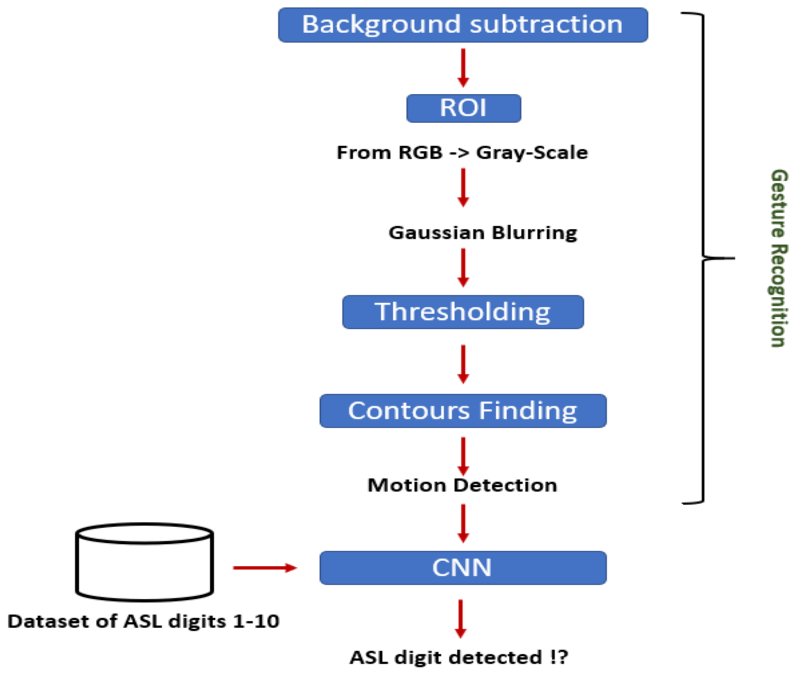


Figure 14: General representation of ASL detection system

CHAPTER FOUR

4. ANALYSIS AND RESULTS DISCUSSION

In our CNN, we used 11 layers: 3 convolutional layers followed by 3 pooling layers, flattened layer, and finally 4 dense layers. the input image was with the size 64x64, and we used 3 levels of convolutional layers, each was followed with a max-pooling layer. The input was the thresholded image of ASL digits from 0 to 9. At each convolutional layer we used ReLU as the activation function. The first convolutional layer had 32 filters with size 3x3 and max-pooling size 2x2 and stride value equal to 2. The following conv layers we added the padding operation. However, in the second level of the convolutional layer we increased the number of filters to 64 with 3x3 kernel size. Lastly the third conv layer had 128 filters with same kernel size as to the previous levels. Each max-pooling size and stride value were the same at all the three levels. The last feature map after the third level of convolutional and pooling layers, is flattened. Then we had three fully connected layers (FC). In the first FC layer we used ReLU to aggregate information, and the same was applied to the second FC layer, however for the last FC layer we used Softmax to finalize the output layer and to measure the accuracy of the prediction. We used 10 training epochs. The Keras sequential model is demonstrated in Appendices B.

In this project we used the webcam to capture the ASL digits and hand gestures. We made sure that the background is clear and white and the lighting is normal so that the system can track the hand gesture movements. For the first time running the system, we need to wait for some seconds for the system to detect the background model then we can show our hand in front of the webcam. During the hand gesture recognition process, the background should be static and there should not be any move or change as it is shown in Figure 15, otherwise a noise will be generated affecting the detection of the hand.

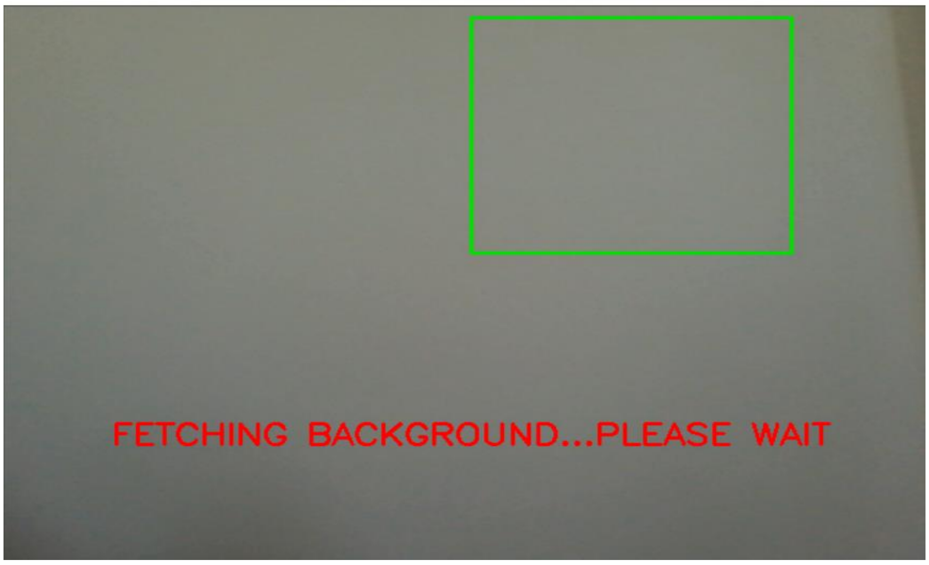
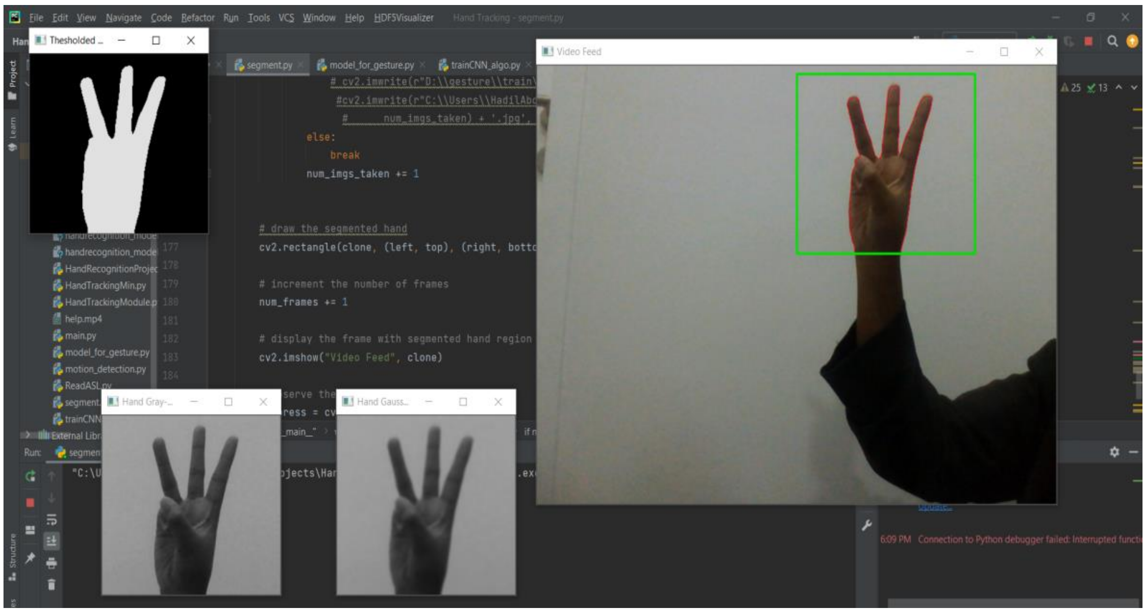


Figure 15: OpenCV webcam fetching background model.

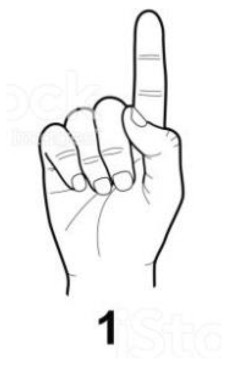
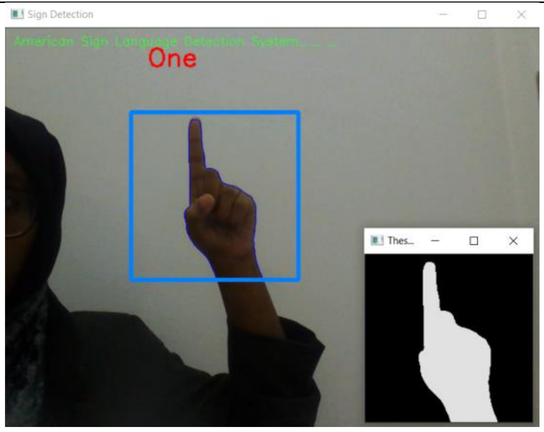
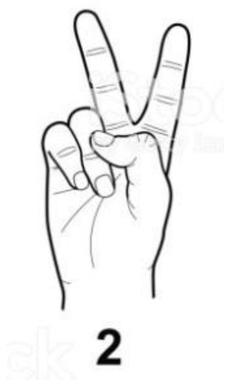
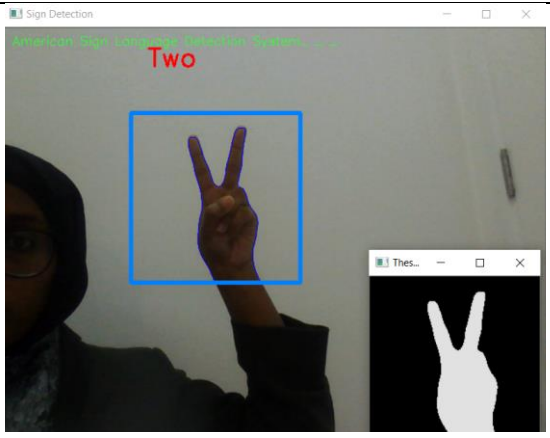
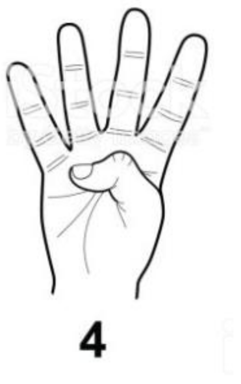
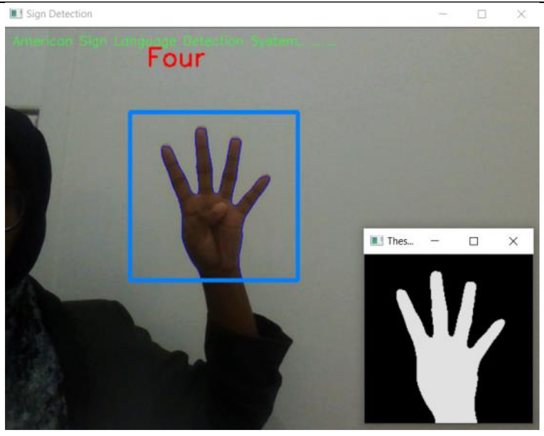
The following Figure 16 shows a successful detection of the hand. The system will add input frames to the running average function until 30 frames. In our system we have 3 python files, one for segmenting and extracting the dataset, one for training the CNN, and the last one contains the recognition model. The Region of Interest (ROI) is shown in green square, and the hand is detected.



```
        # cv2.imwrite(r"D:\\gesture\\train\
         #cv2.imwrite(r"C:\\Users\\HadilAbc
          #      num_imgs_taken) + '.jpg',
    else:
        break
    num_imgs_taken += 1


# draw the segmented hand
cv2.rectangle(clone, (left, top), (right, bottc

# increment the number of frames
num_frames += 1

# display the frame with segmented hand region
cv2.imshow("Video Feed", clone)
```

```
6:09 PM  Connection to Python debugger failed: Interrupted functi
```

Figure 16: Hand detected in ROI.

ASL Number	ASL Number Image	Result
One		 Sign Detection American Sign Language Detection System....... One Thes...
Two		 Sign Detection American Sign Language Detection System....... Two Thes...
Four		 Sign Detection American Sign Language Detection System....... Four Thes...

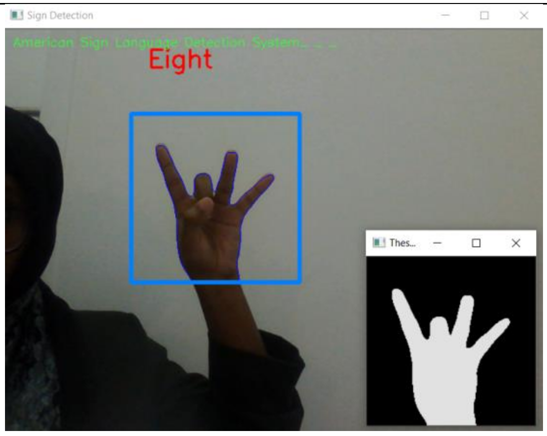
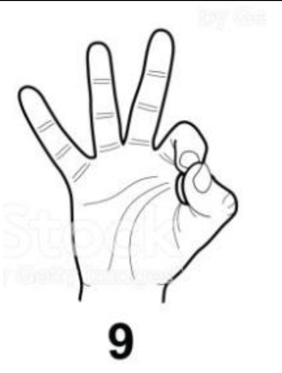
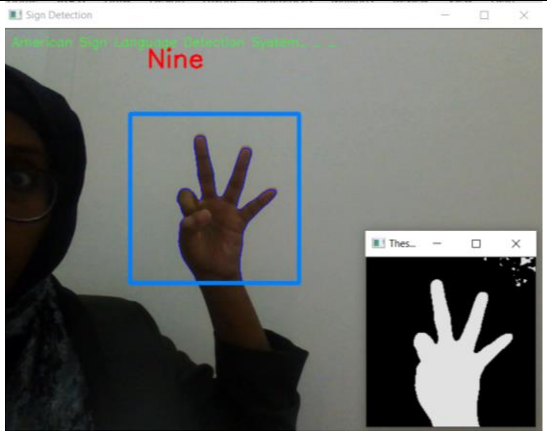
Eight		 Sign Detection American Sign Language Detection System... Eight Thes...
Nine		 Sign Detection American Sign Language Detection System... Nine Thes...

Table 2: ASL prediction system results

As shown in table 2, the system could successfully predict the ASL numbers. Figure 17 demonstrates the accuracy results report over 10 epochs.

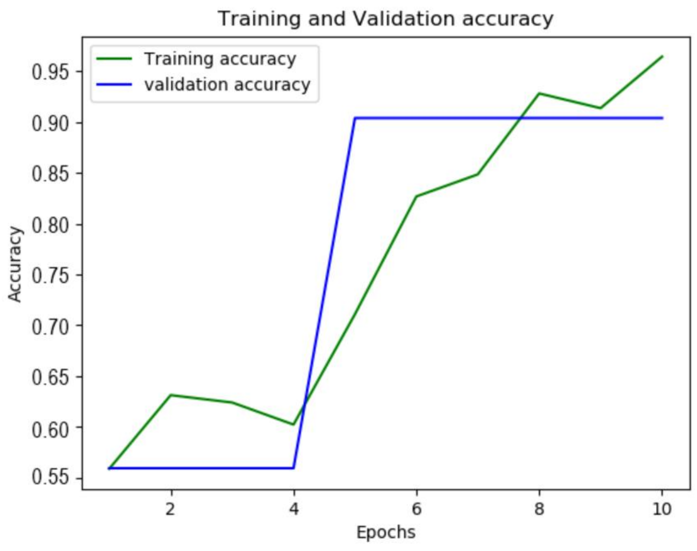


Figure 17: Accuracy result through 10 epochs

CHAPTER FIVE

5. CONCLUSION

5.1 CONCLUSION

In this project, I have proposed a lightweight CNN model and I accomplished creating a hand gesture recognition model to detect ASL numbers from 0 to 9 by using deep learning algorithm CNN with the help of computer vision OpenCV-Python open-source library. The developed hand gesture recognition system detects one hand in real-time from video frames. The system achieved accuracy result of more than 98% with the available manually collected dataset. Despite the accuracy that the system achieved for now, this model needs improvements, as for now the model can detect static gestures with static backgrounds. However, the model can be improved for detecting two hands at the same time and can be improved by detecting more than one gesture at a time. Lastly, despite my laptop could not support the implementation of OpenPose and Tensorflow advanced object detection method as they heavily depend on GPU, because it lacks the latest version of an upgraded GPU and CPU, I could manage to create a lightweight detection system with good accuracy result.

5.2 FUTURE WORK

For the future work, I would like to train the model to detect two hands at the same time to solve complex ASL words and sentences. I would also add the feature of facial landmark detection combined with pose estimation. Once the system can read the body movement combined with face landmarks and hand movements, the system will be able to read complex ASL sentences. I would also use the feature of converting the result detected sentence into a voice by using text-to-speech feature. This feature will allow the system to be an efficient translator. I would also improve the hand gesture recognition in the system and apply the OpenPose algorithm which require advanced GPU. The OpenPose will be faster and precise at detecting the movements. The system will also be deployed into a mobile app, so it can be a portable asl translator system that can be used everywhere. In this system I used OpenCV-python, however in the future I would like to work with Tensorflow and Mediapipe advanced computer-vision libraries.

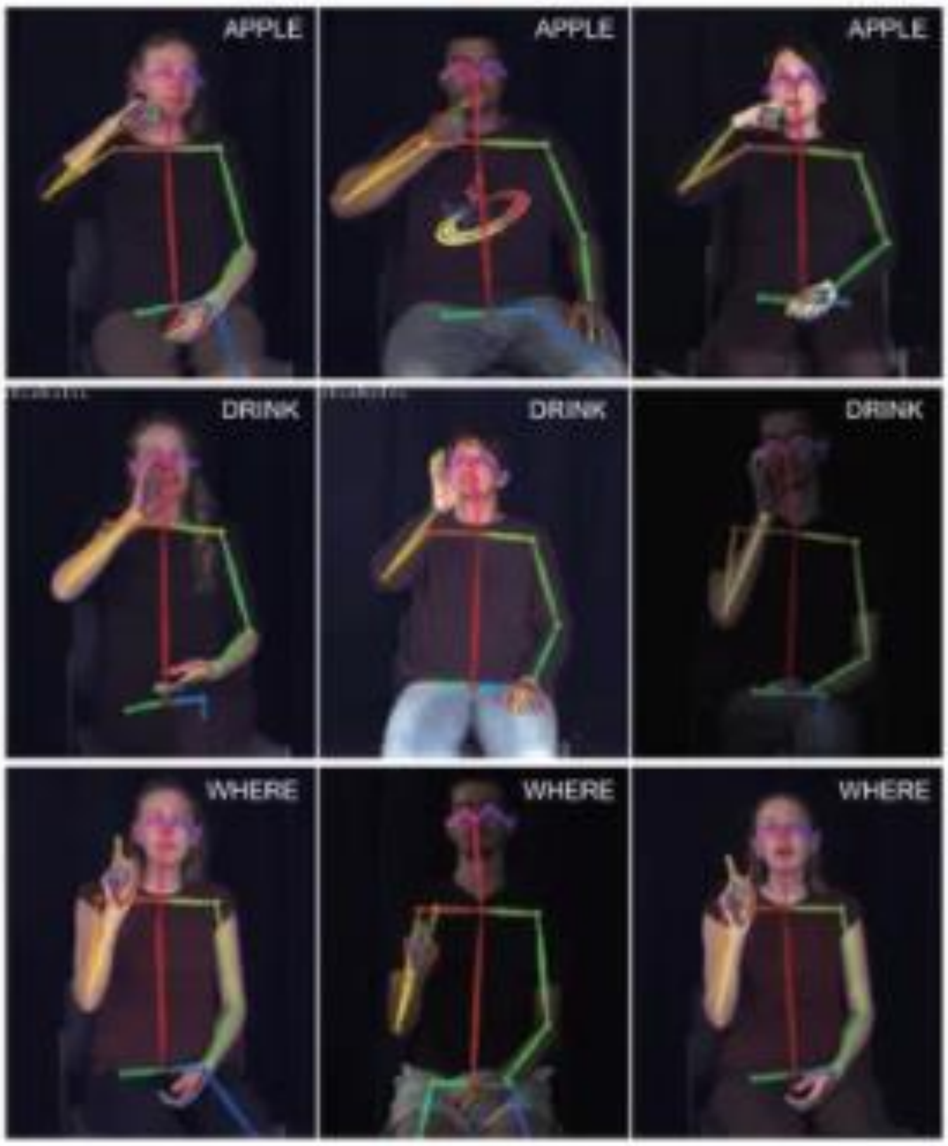


Figure 18: Sign Language Detection using OpenPose

REFERENCES

Gholamalinezhad H. and Khosravi H., (2020). Pooling Methods in Deep Neural Networks, a Review. Image Processing, Faculty of Electrical & Robotics Engineering, Shahrood University of Technology, Daneshgah Blvd., Shahrood, Iran.

Hassan M., and Mohamed A.M. (2020). Deep Learning-Based Approach for Sign Language Gesture Recognition with Efficient Hand Gesture Representation. King Saud University. doi: 10.1109/ACCESS.2020.3032140

Rebecca C. H. (2015). Social interaction between deaf and hearing people. Oxford.

Muneer A., Ghulam M, Wadood A., Mansour A., Mohammed A. B., Tareq S. A.,

L. Zheng, B. Liang and A. Jiang. "Recent Advances of Deep Learning for Sign Language Recognition". (2017). International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2017, pp. 1-7, doi: 10.1109/DICTA.2017.8227483.

Liu T., Fang S., Zhao Y., Wang P. and Zhang J., (2015). Implementation of Training Convolutional Neural Networks. University of Chinese Academy of Sciences, Beijing, China.

Mujahid A., Awan M.J., Yasin A., Mohammed M.A., Damaševiˇcius R., Maskeliunas, R., Abdulkareem, K.H. (2021). Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model. Apple Science. doi: 10.3390/app11094164

M. Siddharth and W. Yaokun. (2020) Chapter 10 - Machine Learning for Subsurface Characterization. Gulf Professional Publishing.

Nagi J., Ducatelle F., Caro G.A.D., Cires¸an D., Meier U., Giusti A., Nagi F., Schmidhuber J., Gambardella L.M., (2011). Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition. 011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011. 342-347. 10.1109/ICSIPA.2011.6144164.

O. Munir, A. Ali, and C. Javaan. (2020). Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. Jornal of Imaging, Trnovszky T., Kamencay P., Orjesek R., Benco M. and Sykora P., (2020). Animal Recognition System Based on Convolutional Neural Network. Department of multimedia and information-communication technologies, Faculty of Electrical Engineering, University of Zilina. Digital image processing and computer graphics. doi: 10.15598/aeee.v15i3.2202

Tran D.S., Ho N.H, Yang H.J., Baek E.T., Kim S.H. and Lee G. (2020). Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network. Applied Sciences, MDPI. doi: 10.3390/app10020722

APPENDICES

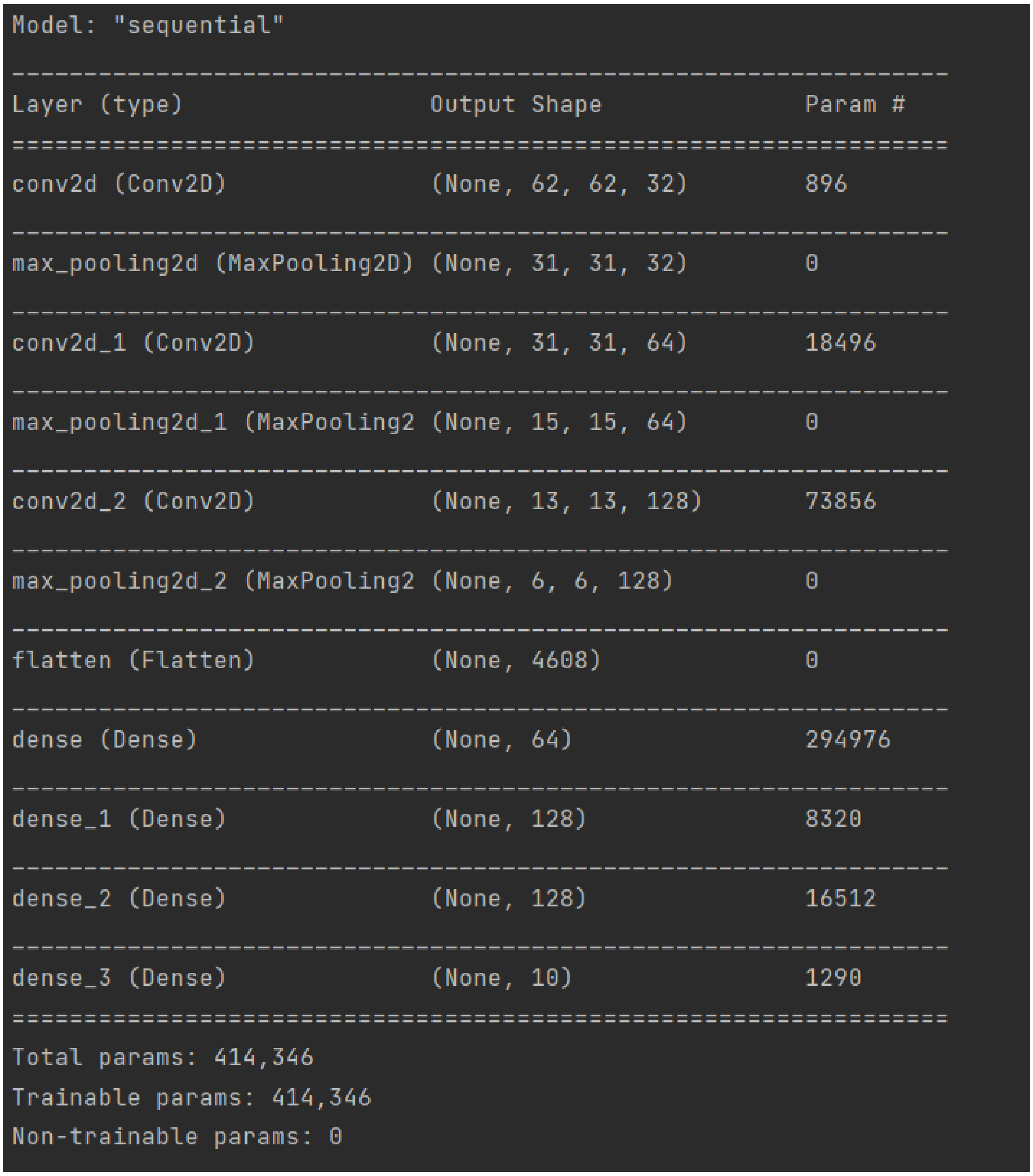
APPENDIX A

Gantt Chart

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	completed	**Real-time sign language detection model based on hand gesture recognition using convolutional neural network**	95 days	1/3/2021	25/6/2021	
2	completed	**Finding supervisor**	5 days	1/3/2021	5/3/2021	
3	completed	Search for supervisor	5 days	1/3/2021	5/3/2021	2
4	completed	Title discussion	1 day	4/3/2021	4/3/2021	2
5	completed	Supervisor consent	1 day	4/3/2021	4/3/2021	2
6	completed	**Proposal**	5 days	1/3/2021	4/3/2021	
7	completed	Discussion with the supervisor	1 day	5/3/2021	5/3/2021	6
8	completed	Submitting Proposal	1 day	5/3/2021	5/3/2021	6
9	completed	Supervisor approval of topic	1 day	5/3/2021	5/3/2021	6
10	completed	**Progress**	95 days	6/3/2021	25/6/2021	
11	completed	Installing OpenCV	21 days	10/3/2021	1/4/2021	10
12	completed	Collecting Data	7 days	2/4/2021	9/4/2021	10
13	completed	Experimenting and Testing data	7 days	10/4/2021	17/4/2021	10
14	completed	**Create algorithm**	14 days	18/4/2021	2/5/2021	
15	completed	**Testing and Training**	14 days	3/5/2021	24/5/2021	
16	completed	**Writing final report**	27 days	25/5/2021	22/5/2021	
17	completed	**Poster**	1 days	23/5/2021	24/5/2021	
18	completed	**Final Submission [Report + Poster]**	1 day	25/6/2021	26/6/2021	
19	Ongoing	**Presentation Q & A**	1 day	28/5/2021	2/7/2021	

APPENDIX B

CNN Keras “Sequential” model representation:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 31, 31, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 64)                294976
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 414,346
Trainable params: 414,346
Non-trainable params: 0
```