HAND GESTURE RECOGNITION USING CNN AND CONVERSION TO SPEECH

A MAJOR PROJECT Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment for the requirement for the award of Degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

TANGELLA NAGARAJU	**19641A05D1**
HUMERA ANWAR	**19641A05G2**
RENUKUNTLA SOMESH	**19641A05C2**
YERRABELLI HARISH RAO	**19641A05C3**

Under the Guidance of

Mrs. A. Swetha

Asst. Prof

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VAAGDEVI COLLEGE OF ENGINEERING

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta, Khilla Warangal (Mandal), Warangal Urban – 506005 (T.S)

VAAGDEVI COLLEGE OF ENGINEERING

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta, Khilla Warangal (Mandal), Warangal Urban – 506005 (T.S)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the MAJOR PROJECT entitled **"HAND GESTURE RECOGNITION USING CNN AND CONVERSION TO SPEECH"** is submitted by **TANGELLA NAGARAJU(19641A05D1), HUMERA ANWAR(19641A05G2), RENUKUNTLA SOMESH (19641A05C2), YERRABELLI HARISHRAO(19641A05C3)** in partial fulfilment of the requirements for the award of the Degree in Bachelor of Technology in Computer Science and Engineering during the academic year 2022-2023.

Guide	**Head of the Department**
Mrs. A. Swetha	**Dr. N. SATHYAVATHI**

External Examiner

ACKNOWLEDGEMENT

The development of the project though it was an arduous task, it has been made by the help of many people. We are pleased to express our thanks to the people whose suggestions, comments, criticisms greatly encouraged us in betterment of the project.

We would like to express our sincere thanks and profound gratitude to **Dr. K. Prakash**, principal of **Vaagdevi College of Engineering,** for his support, guidance and encouragement in the course of our project.

We are also thankful to Head of the Department of CSE **Dr. N. Satyavathi**, Associate Professor for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

We would like to express our sincere gratitude and indebtedness to my project Guide **A. Swetha ,** for her/his valuable suggestions and interest throughout the course of this project.

We are also thankful to **CH. ARAVINDH KUMAR, REKHA DEVI** for their valuable suggestions, encouragement and motivations for completing this project successfully.

We are thankful to all other faculty members for their encouragement. We convey our heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support through the work. we sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

TEAM

TANGELLA NAGARAJU	**19641A05D1**
HUMERA ANWAR	**19641A05G2**
RENUKUNTLA SOMESH	**19641A05C2**
YERRABELLI HARISHRAO	**19641A05C3**

DECLARATION

We hereby declare that the project report entitled "**Hand Gesture Recognition using CNN and Conversion to Speech**" has been written by us and has not been submitted either in part of or whole for the award of any degree, diploma or any other similar title to this or any other university.

TEAM:

TANGELLA NAGARAJU	**19641A05D1**
HUMERA ANWAR	**19641A05G2**
RENUKUNTLA SOMESH	**19641A05C2**
YERRABELLI HARISH RAO	**19641A05C3**

ABSTRACT

Hand gestures are a nonverbal way of communication that we see all the time as we communicate with one another. A gesture is a physical movement of the hands, fingers, arms, and other body parts that enables people to express meaningful information to oneanother [1]. Two approaches for human–computer interactions are the data gloves methodand the vision-based approach. Hand gesture detection and classification were among the experiments in which the vision-based technique was put to the test. A Hand gestures areone of the most practical ways to make a user-friendly and adaptive interface between people and machines. Applications including virtual object manipulation, gaming, and gesture detection are used in Human Computer Interaction (HCI) systems. Hand trackingis concerned with three fundamental aspects of computer vision: hand segmentation, handcomponent detection via feature extraction, and hand tracking as a theoretical component.The most successful communication tool and the most widely used idea in a gesture recognition system is hand gestures. Hand gestures can be identified using one of two methods: posture (a static hand shape ratio with no movement) or gesture (movement of the hands) (a dynamic hand motion with or without hand motions). Any camera can detectany type of hand motion; however, different cameras have different resolution quality. Two-dimensional cameras in a continuous surface known as 2D can detect the majority of finger movement. To recognise dynamic movements in our research, we employed Deep Learning models such as Convolutional Neural Networks (CNN) and some pre- trained models and OpenCV Functions. With CNN, we obtained excellent results.

Keywords: Hand gesture**,** Convolutional Neural Networks (CNN), OpenCV, Deep Neural Networks (DNN), Pre trained models,

LIST OF ABBREVATIONS

AI	:	Artificial Intelligence
CNN (ConvNet)	:	Convolutional Neural Network
HCI	:	Human Computer Interaction
DL	:	Deep Learning
DNN	:	Deep Neural Network
TL	:	Transfer Learning
VGG	:	Visual Geometry Group

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Hand Gestures and its Types

Hand gesture recognition is a promising research field in computer vision. Its most appealing application is the development of more effective and friendly interfaces for human– machine interaction, since gestures are a natural and powerful way of communication. Moreover, it can be used for teleconferencing because it does not requireany special hardware. Last but not least, it can be applied to the interpretation and learningof sign languages [1]. Sign language is a language that employs signs made by moving the hands combined with facial expressions and postures of the body. It is one of severalcommunication options used by people who are deaf or hard-of-hearing. Gesture language identification is one of the areas being explored to help the deaf integrate into the community and has high applicability. The essential aim of building hand gesture recognition system is to create a natural interaction between human and computer where the recognized gestures can be used for controlling a robot or conveying meaningful information.

Hand gestures are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human–computer interaction (HCI), home automation and medical applications. Research papers based on hand gestures have adopted many different techniques, including those based on instrumented sensor technology and computer vision. In other words, the hand sign can be classified under many headings, such as posture and gesture, as well as dynamic and static, or a hybrid of the two.

The whole system consists of three components: hand detection, gesture recognition andhuman-computer interaction (HCI) based on recognition; and realizes the robust control of mouse and keyboard events with a higher accuracy of gesture recognition. Specifically, we use the convolutional neural network (CNN) to recognize gestures and makes it attainable to identify relatively complex gestures.

Human computer interaction (HCI) also named Man-Machine Interaction (MMI) [3][4] refers to the relation between the human and the computer or more precisely the machine,and since the machine is insignificant without suitable utilize by the human [3]. There are two main characteristics should be deemed when designing a HCI system as mentioned in [3]: functionality and usability. System functionality referred to the set of functions orservices that the system equips to the users [3], while system usability referred to the leveland scope that the system can operate and perform specific user purposes efficiently [3].The system that attains a suitable balance between these concepts considered as influential performance and powerful system [3]. Gestures used for communicating between human and machines as well as between people using sign language [5].

There are two kinds of Hand Gestures, they can be either static or dynamic gestures. Statichand gestures (Posture or certain pose) can be defined as the gestures where the position and orientation of hand in space does not change for an amount of time. If there are any changes within the given time, the gestures are called dynamic gestures. Dynamic hand gestures (A Sequence of Postures) include gestures like waving of hand while static handgestures include joining the thumb and the forefinger to form the "Ok" symbol.

1.2 Hand Gesture Detection and Recognition

1.2.1 Hand Detection

The process of hand detection involves locating the presence of a hand in a still image or series of moving photos. The hand in the scene can be tracked in the case of moving sequences, however this is more applicable to applications like sign language. The fundamental idea behind hand detection is that a human eye can accurately detect objects that a machine cannot. From the perspective of a machine,it seems exactly like a person using his senses to look for something. The factors, which make the hand detection task difficult to solve are variations in image plane and pose, skin color and other structure components, lighting condition and background.

1.2.2 Hand Recognition

Over the past 30 years, hands have been important research topics in the fields of computer vision and image processing. In these sectors, significant advancements have been made, and numerous strategies have been put forth. However, a fully automated hand gesture recognition system's normal operation can be demonstratedin the Figure 1.1 below.

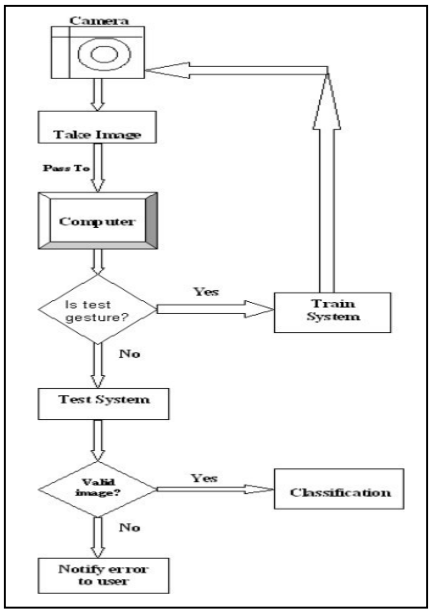


Fig 1.1 Flow Chart of Hand Gesture Recognition

CHAPTER - 2

LITERATURE REVIEW

2. Literature Review

2.1 Literature Review

In the beginning of this chapter, we present the conventional models available in the literature for Gesture classification. Then we present a detailed study of deep learning-based models for the same task.

Hand gesture recognition is a natural way of human computer interaction and an area of very active research in computer vision and machine learning. This is an areawith many different possible applications, giving users a simpler and more natural way to communicate with robots/systems interfaces, without the need for extra devices. So, the primary goal of gesture recognition research applied to Human- Computer Interaction (HCI) is to create systems, which can identify specific humangestures and use them to convey information or controlling devices. For that, vision-based hand gesture interfaces require fast and extremely robust hand detection, andgesture recognition in real time.

The design of any gesture recognition system essentially involves the following three aspects: (1) data acquisition and pre-processing; (2) data representation or feature extraction and (3) classification or decision-making.

2.1.1 Data Acquisition and Pre-Processing

Hand Gesture can be obtained through different ways using Data Gloves, 3D Cameras, Mechanical devices etc. Hand gesture recognition research is classified inthree categories. First "**Glove Based Analysis**" attaching sensor with gloves mechanical or optical to transduces flexion of fingers into electrical signals for handposture determination and additional sensor for position of the hand. This sensor is usually an acoustic or a magnetic that attached to the glove. The second approach is"**Vision Based Analysis**" that human beings get information from their surroundings, and this is probably most difficult approach to employ in satisfactory way. Several cameras attached to this model to determine parameters correspondingfor matching images of the hand, palm orientation and join angles to perform hand gesture classification. The Third implementation is "**Analysis of drawing gesture**"use stylus as an input device. These drawing analyses led to recognition of written text. Mechanical sensing work has used for hand gesture recognition at vast level for direct and virtual environment manipulation.

To achieve better results, the Gestures produced from Vision-based analysis must be preprocessed in order to fit those pictures to the model. To prepare the data, we utilize OpenCV methods to reshape, resize, and convert the photos from color to grayscale to integrate them to Convolutional Neural Networks.

Segmentation process is the first process for recognizing hand gestures. It is the process of dividing the input image (in this case hand gesture image) into regions separated by boundaries [12]. The segmentation process depends on the type of gesture, if it is dynamic gesture then the hand gesture needs to be located and tracked[12], if it is static gesture (posture) the input image has to be segmented only. The hand should be located firstly, generally a bounding box is used to specify the depending on the skin color [13] and secondly, the hand have to be tracked, for tracking the hand there are two main approaches; either the video is divided into frames and each frame have to be processed alone, in this case the hand frame is treated as a posture and segmented [12], or using some tracking information such asshape, skin color using some tools such as Kalman filter [12].

2.1.2 Data Representation and Feature Extraction

The objective of this first application is to identify the number of fingers active in ahand gesture. The strategy applied is to count the number of U turns appearing in the refined outline. From the example, the posture contains five active fingers and ten U-turns. The scale factor of two is the same for all other cases expect for zero that was not considered in this project. The average length of a human finger has been considered as threshold, the reason being to avoid counting ragged corners thatstill appear in the outline after the smoothing step. To detect the U-turns, four consecutive segment lines are considered at a time. By computing the azimuth, the algorithm looks for two segments with opposite directions within a threshold. The process is repeated throughout the outline.

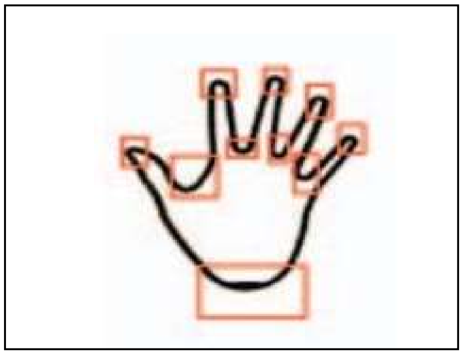


Figure 2.1 Gesture Recognition using U-turns.

Various methods have been applied for representing the features can be extracted. Some methods used the shape of the hand such as hand contour and silhouette [6] while others utilized fingertips position, palm center, etc. [6] created 13 parametersas a feature vector, the first parameters represent the ratio aspect of the bounding box of the hand and the rest 12 parameters are mean values of brightness pixels in the image. [14] used Self-Growing and Self-Organized Neural Gas (SGONG) neuralalgorithm to capture the shape of the hand, then three features are obtained; Palm region, Palm center, and Hand slope.

2.1.3 Classification and Decision Making

The gesture classification approach is used to recognize the gesture after modellingand analyzing the input hand picture. The recognition process is influenced by the right selection of feature parameters and an appropriate classification method [7]. For example, edge detection or contour operators [9] cannot be employed for gestureidentification since they create a large number of hand postures and may result in misclassification [9]. The Euclidean distance measure was used to categorize the gestures [19][5][17]. Among the statistical tools used for gesture classification, the HMM tool has demonstrated its capacity to detect dynamic movements [20][13], inaddition to the Finite State Machine (FSM) [21], Learning Vector Quantization [22], and Principal Component Analysis (PCA) [23]. The neural network has been widelyused in the field of hand contour extraction [14], as well as hand gesture recognition[24][25][26].

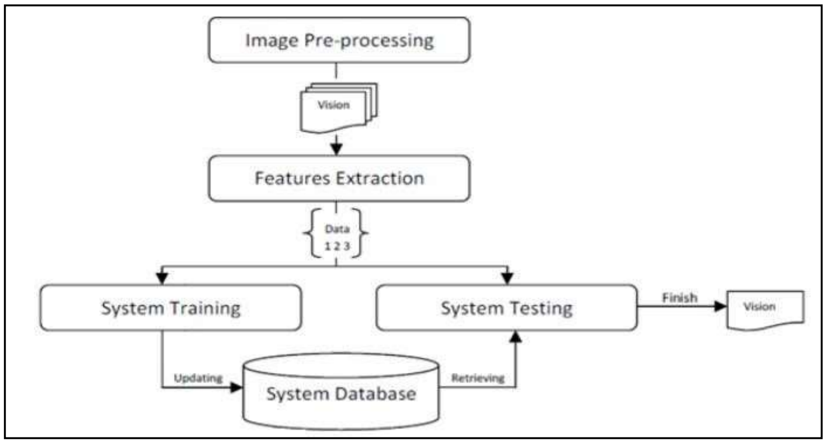


Figure 2.2 Architecture of Hand Gesture Recognition

2.2 Existing Methodologies

2.2.1 Hand gesture recognition using a neural network shape fitting technique

This method has been constructed using an Artificial Neural Network (ANN) basedon a form fitting methodology [4]. After filtering, a colour segmentation approach using the YbrCr colour space was utilised in this system to detect hand. The hand morphology then approached the form of the hand. Hand shape and finger orientationcharacteristics were retrieved and fed into an ANN. Using this strategy, they attained 94.05 percent accuracy. They have identified 5 distinct gestures named Little, Index, Middle, Ring, Thumb Fingers.

2.2.2 Static hand gesture recognition using artificial neural network

An ANN-based gesture detection system has also been created [5]. Images in thissystem were segregated depending on skin colour. The ANN characteristics used were pixel changes through cross sections, boundaries, and scalar descriptions such as aspect ratio and edge ratio. Following the creation of those feature vectors, they were put into the ANN for training. The accuracy was almost 98%. This study has used American Sign Language dataset and from that dataset they identified only 10Gestured named A, B, C, D, G, H, I, L, V, Y. These alphabets are represented in theAmerican Sign Language dataset.

2.2.3 Hand gesture recognition using haar-like features and a stochastic context-free grammar

To recognise gestures, a statistical technique based on haar-like features was presented [6]. The AdaBoost technique was used to train the model in this system. The entire projectwas separated into two tiers. A stochastic context-free grammar was employed at a higherlevel to recognise gestures. Postures were recognised at a lower level. The grammar wasused to construct a terminal string for each input. Each rule's probability was computed, and the rule with the highest probability for the provided string was chosen as the rule. The gesture linked with this rule was returned as the input's gesture. However, feature extraction by manual process has some drawbacks. The extraction process is tedious andevery possible feature may not be extracted. Moreover, the extraction may become human-biased.

Then comes the automated feature engineering which is not tedious and not human biased. Also, almost all the features can be captured using automated feature engineering.Useful features from structured data can be extracted by CNN. So, a move to automated feature engineering was made and deep learning i.e., CNN started to emerge. This studyhave recognized 4 kinds of gestures like Two Fingers, Palm, Fist and Little Finger.

2.2.4 Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation

A Hand Gesture Detection system based on Convolutional Neural Networks (CNN)was applied to two distinct datasets: one with 8000 training samples and 1600 testingsamples, and the other with 2000 samples. They pre-processed the photos, convertingthem from colour to grayscale, and then fed the result to the CNN Model, which contains two convolutional layers, two Max-Pooling layers, three dropout layers, andtwo fully connected layers, as well as Stochastic Gradient Descent as an optimizer function. They attained 92.87 percent accuracy without augmenting the dataset and

97.12 percent accuracy after augmenting the dataset [7]. This paper have implemented on 10 distinct gestures like Index, Peace, Three, Palm_Opened, Palm_closed, OK, Thumb, Fist, Swing and Smile.

2.2.5 Hand Gesture Recognition for Human Computer Interaction

The authors of this study used Human Computer Interaction to create a gesture recognition system that changed the apps based on the movements produced. They have employed both static and dynamic gestures to translate those gestures into actions that will launch power point, VLC Player, or Google Chrome, for example,based on the rules they have specified. The dynamic motions they utilize are employed to move the power point slides. They utilized OpenCV functions to transform the colored picture to grayscale, identify convexity flaws, and detect movements based on the convexity flaws. They received the greatest results when they applied plain backdrop photos for categorization, and vice versa when they applied non-plain backdrop photos. The accuracies varied by an average of 25-30 percent. This study has identified 7 gestures named 2, 3, 4, 5, Palm, Fist, Swipe (Dynamic)

CHAPTER – 3

PROPOSED WORK

3. PROPOSED WORK

This Chapter is broadly divided into 3 different sections. First part describes Computer Vision for Hand Gesture Recognition, second part illustrates various Deep Learning (DL) models for Hand Gesture Recognition. Finally, we describe ourproposed Methodology.

3.1 Computer Vision for Hand Gesture Recognition

Russell and Norvig [6] identified three broad approaches used in computer vision todistill useful information from the raw data provided by images. The first is the feature extraction approach, which focuses on simple computations applied directly to digital images to measure some useable characteristic, such as size. This relies ongenerally known image processing algorithms for noise reduction, filtering, object detection, edge detection, texture analysis, computation of optical flow, and segmentation, which techniques are commonly used to pre-process images for subsequent image analysis. This is also considered an “uninformed” approach. The second strategy is called the recognition approach, and it focuses on classifying anddifferentiating things based on knowledge of traits that groups of related items have,such as shape, appearance, or patterns of elements. Here, a "classifier" can match classes to objects based on the pattern of their features or structural descriptions thanks to the application of artificial intelligence in knowledge representation in computer vision. By being fed a training set of objects and their classes, a classifiermust "learn" the patterns in order to minimize errors and maximize successes through a methodical process of development. There are many techniques in artificial intelligence that can be used for object or pattern recognition, including statistical pattern recognition, neural nets, genetic algorithms and fuzzy systems. The third method is called reconstruction, and it focuses on creating a geometric representation of the world that is implied by the image or images and is then utilizedas a springboard for action. This relates to the picture interpretation step, which is the most advanced and difficult phase of computer vision processing. The goal hereis to make it possible for the computer vision system to build internal models basedon the information provided by the images and to update or discard these internal models as they are validated against the real world or other criteria. Image understanding occurs if the internal model is consistent with reality. Thus, image understanding requires the construction, manipulation and control of models and atthe moment relies heavily upon the science and technology of artificial intelligence.

3.1.1 OpenCV

OpenCv is a widely used tool in computer vision. It is a computer vision library forreal-time applications, written in C and C++, which works with the Windows, Linuxand Mac platforms. It is freely available as opensource software from http://sourceforge.net/projects/opencvlibrary/.

A digital image is generally understood as a discrete number of light intensities captured by a device such as a camera and organized into a two-dimentional matrixof picture elements or pixels, each of which may be represented by number and all of which may be stored in a particular file format (such as jpg or gif) [8].

OpenCV goes beyond representing an image as an array of pixels. It represents an image as a data structure called an Pil Image that makes immediately accessible useful image data or fields, such as:

- width – an integer showing the width of the image in pixels
- height – an integer showing the height of the image in pixels
- imageData – a pointer to an array of pixel values
- nChannels – an integer showing the number of colors per pixel
- depth – an integer showing the number of bits per pixel
- widthStep – an integer showing the number of bytes per image row
- imageSize – an integer showing the size of in bytes
- roi – a pointer to a structure that defines a region of interest within theimage

Functions used in Gesture Recognition

Cv2.VideoCapture()

It is used to get a video capture object for the camera, in an infinity loop use the read() method to read the frames using created Video capture object. Its argumentcan be either the device index or the name of the device to capture the frames. Bydefault it gives 640*480 pixels of image.

Cv2.rectangle()

The general syntax is *cv2.rectangle(image, start_point, end_point, color, thickness)*. Itis used to draw a rectangle on any image. It detects the ROI and draws a rectangle overthe detected hand gestures.

Cv2.cvtcolor()

It is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV. Syntax is *cv2.cvtcolor(src, code, dst, dstcn).* Src is the input image, code is the color space conversion code, dst is the output image with same size and depth, dstcn is the number of channels in output image.

Cv2.GaussianBlur()

It is used to smooth the image using Gaussian Blur. Using this method the sharp edges in an image are smoothed. Syntax for applying Gaussian Blur function is *cv2.GaussianBlur(Input_array, Output_array, size, borderType).*

Cv2.threshold()

This function is used to apply the thresholding. The first argument is the source image, which should be a grayscale image, Second argument is the threshold valuewhich is used to classify pixel values, third argument is the maximum value that is assigned to pixel values exceeding the threshold. We have implemented both BinaryInversion and Otsu thresholding techniques in our model. There are many types of thresholding techniques.

3.2 Deep Learning Models

In this Section First, we discuss about Convolutional Neural Network which performs both Feature Extraction and Classification followed by Transfer Learningscheme and some popular pre-training Models that we use for Feature extraction. Finally, weintroduce a Deep Neural Network Architecture that we used for Hand Gesture Recognition.

3.2.1 Convolutional Neural Network

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It’s also knownas a ConvNet. A convolutional neural network is used to detect and classify objects in animage. In CNN, every image is represented in the form of an array of pixel values.

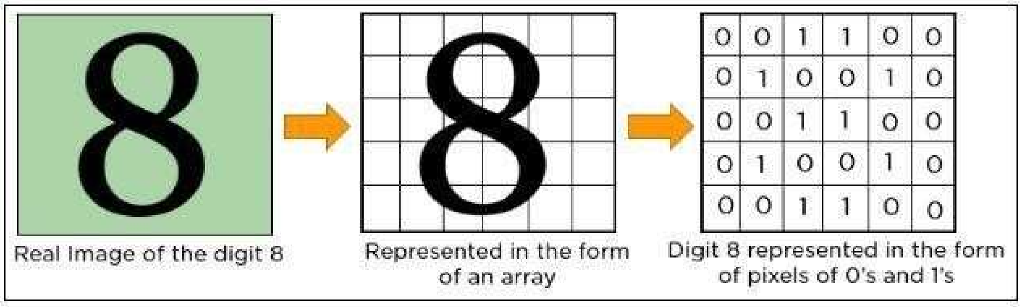

Real Image of the digit 8
Represented in the form of an array
Digit 8 represented in the form of pixels of 0's and 1's

Fig 3.1 Pixel representation of Image.

A convolution neural network has multiple hidden layers that help in extractinginformation from an image. The important layers in CNN are,

Convolution Layer: This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values. Consider the following5x5 image whose pixel values are either 0 or 1. There's alsoa filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.

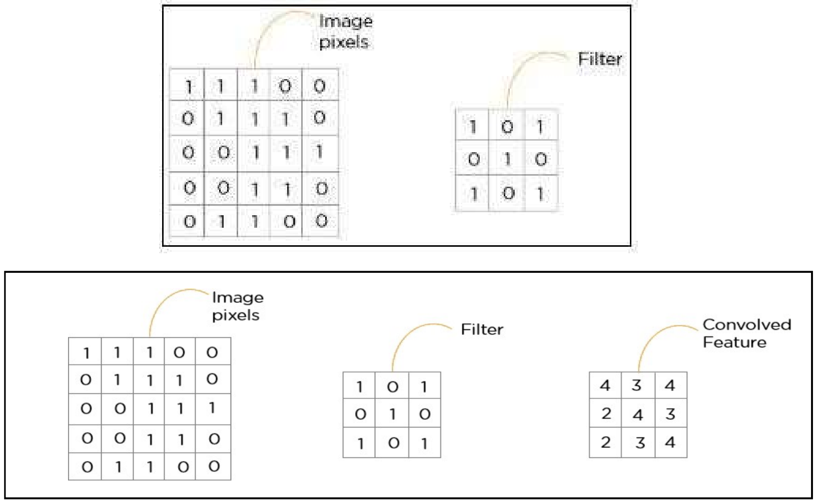


Fig 3.2 Convolution Operation.

ReLU Layer*:* ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer. ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to thenetwork, and the generated output is a rectified feature map. Below is the graph ofa ReLU function:

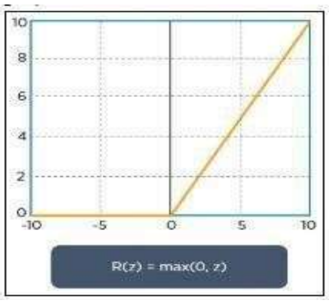


Fig 3.3 ReLU Operation

The original image is scanned with multiple convolution and ReLU layers for locating the features.

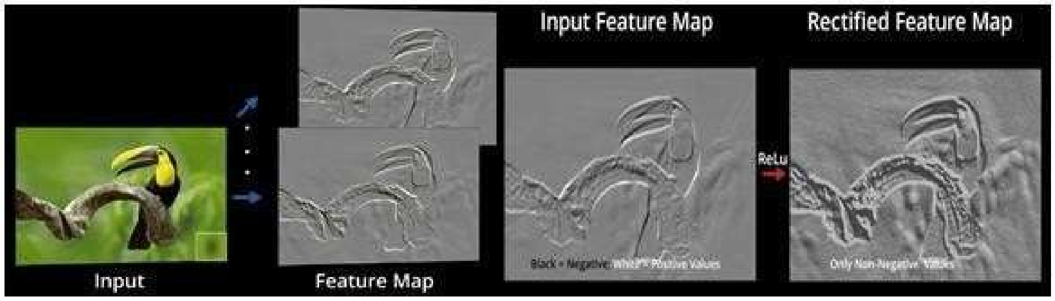


Fig 3.4 Feature Map Visualization after Convolution and ReLU Operations.

Pooling Layer: Pooling is a down-sampling operation that reduces the dimensionality ofthe feature map. The rectified feature map now goes through a pooling layer to generatea pooled feature map.

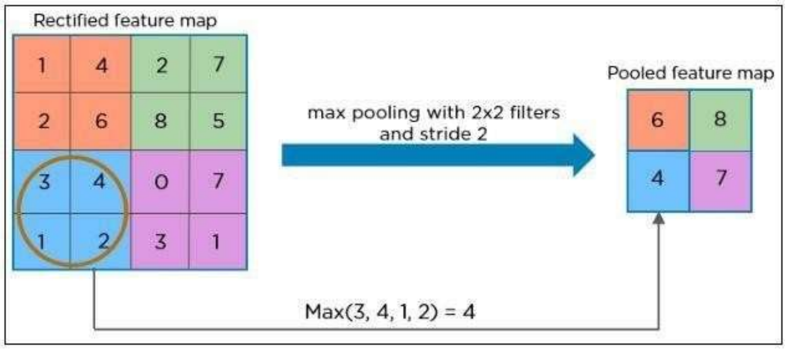


Fig 3.5 Pooling Operation.

The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.

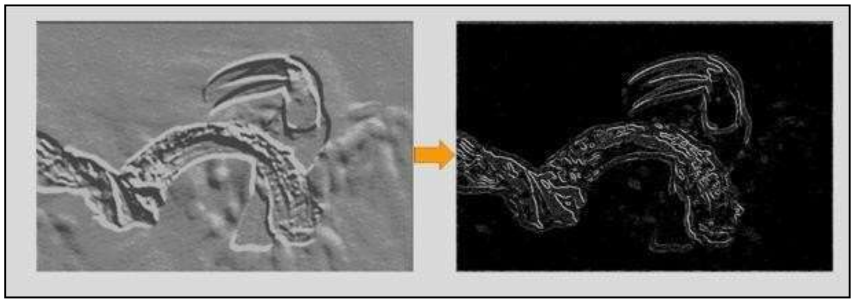


Fig 3. 6 Feature Map Visualization after Pooling Operation.

The next step in the process is called ***flattening***. Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuouslinear vector.

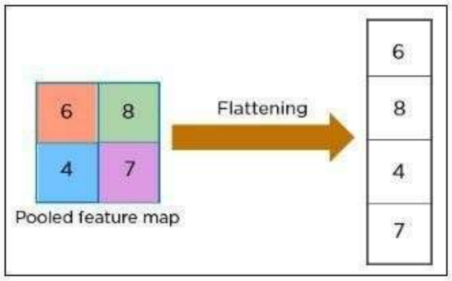


Fig 3.7 Flattening.

The flattened matrix is fed as input to the ***fully connected layer*** to classify the image.

The downside of the Convolutional Neural Networks (CNN) is that they need enormousamounts of data for training, which is usually scarce for most of the real- time applications. This problem can be addressed by the introduction of Transfer Learning where the knowledge gained by a deep learning model can be transferred toother modelswithout any training. This is achieved using pre-trained models.

3.2.2 Transfer Learning

Transfer learning is a popular method in computer vision because it allows to build accurate models in a timesaving way (Rawat & Wang 2017) [17]. With transfer learning, instead of starting the learning process from scratch, you start from patternsthat have been learned when solving a different problem. This way you leverage previous learnings and avoid starting from scratch.

In computer vision, transfer learning is usually expressed using **pre-trained models**.

A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem like the one that we want to solve. Accordingly, due to the computational cost of training such models, it is common practice to import and use models from publishedliterature (e.g. VGG, Inception, ResNet). A comprehensive review of pre-trained models' performance on computer vision problems using datafrom the ImageNet (Denget al. 2009) challenge is presented by Canziani et al. (2016). [18]

Several pre-trained models used in transfer learning are based on large **convolutional neural networks (CNN)** (Voulodimos et al. 2018) [19]. In general, CNN was shown toexcel in a wide range of computer vision tasks (Bengio 2009) [20]. Its high performanceand its easiness in training are two of the main factors driving the popularity of CNN over the last years. A typical CNN has two parts:

1. Convolutional base, which is composed by a stack of convolutional and poolinglayers. The main goal of the convolutional base is to generate features from the image. For an intuitive explanation of convolutional and pooling layers.

2. Classifier, which is usually composed by fully connected layers. The main goal of the classifier is to classify the image based on the detected features. A fully connected layer is a layer whose neurons have full connections to all activation in the previous layer.

One important aspect of these deep learning models is that they can automaticallylearn **hierarchical feature representations**. This means that featurescomputed by the first layer are general and can be reused in different problem domains, while features computed by the last layer are specific and depend on the chosen dataset and task. According to Yosinski et al. (2014) [21], '*if first-layer features are general and last-layerfeatures are specific, then there must be a transition from general to specific somewherein the network'*. As a result, the convolutional base of our CNN — especially its lower layers (those who are closer to the inputs) — refer to general features, whereas the classifier part, and some of the higher layers of the convolutional base, refer to specialized features.

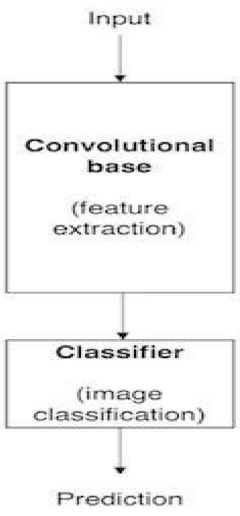


Fig 3.8 Architecture of a model based on CNN.

Repurposing a pre-trained model: When you're repurposing a pre-trained model for your own needs, you start by removing the original classifier, then you add a newclassifier that fits your purposes, and finally you must fine-tune your model according to one of three strategies,

1. Train the entire model. In this case, you use the architecture of the pre-trained model and train it according to your dataset. You're learning the model from scratch,so you'll need a large dataset (and a lot of computational power).

2. Train some layers and leave the others frozen. As you remember, lower layersrefer to general features (problem independent), while higher layers refer to specific features (problem dependent). Here, we play with that dichotomy by choosing howmuch we want to adjust the weights of the network (a frozen layerdoes not change during training).

3. Freeze the convolutional base. This case corresponds to an extreme situation ofthe train/freeze trade-off. The main idea is to keep the convolutional base in its original form and then use its outputs to feed the classifier. You're using the pre- trained model as a fixed feature extraction mechanism, which can be useful if you'reshort on computational power, your dataset is small, and/or pre-trained model solvesa problem very similar to the one you want to solve.

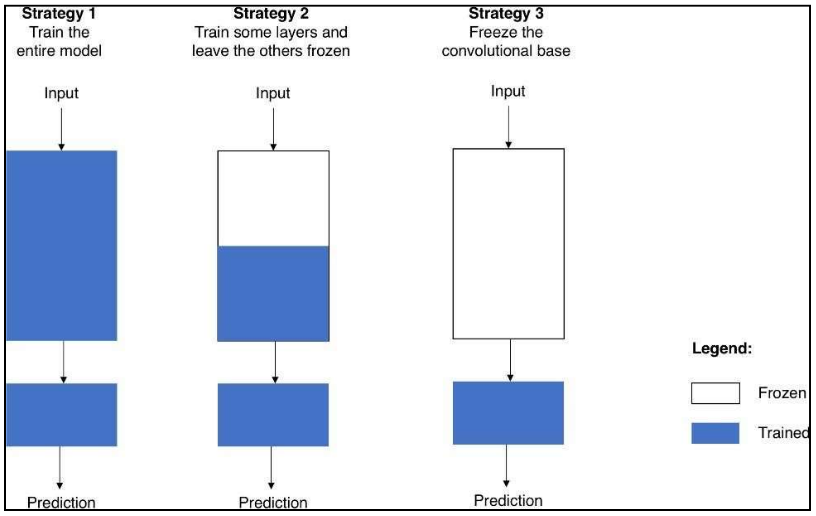


Fig 3.9 Fine-tuning strategies.

Transfer Learning Process: From a practical perspective, the entire transfer learning process can be summarised as follows,

1. Select a pre-trained model.
2. Classify your problem according to the Size-Similarity Matrix.
3. Fine-tune your model.

Classifiers on top of deep convolutional neural networks: As mentioned before, modelsfor image classification that result from a transfer learning approach based onpre-trained convolutional neural networks are usually composed of two parts**:**

1. **Convolutional base**, which performs feature extraction.
2. **Classifier**, which classifies the input image based on the features extracted by theconvolutional base.

Since in this section we focus on the classifier part, we must start by saying that differentapproaches can be followed to build the classifier. Some of the most popularare:

Fully connected layers: For image classification problems, the standard approach is to use a stack of fully connected layers followed by a softmax activated layer (Krizhevsky et al. 2012, Simonyan & Zisserman 2014, Zeiler & Fergus 2014) [24]

[15] [25]. The softmax layer outputs the probability distribution over each possible class label and thenwe just need to classify the image according to the most probableclass.

Global average pooling: A different approach, based on global average pooling, is proposed by Lin et al. (2013). In this approach, instead of adding fully connected layers on top of the convolutional base, we add a global average pooling layer and feed its output directly into the softmax activated layer. Lin et al. (2013) provides adetailed discussion on the advantages and disadvantages of this approach.

3.2.3 Pre-trained Models

VGG 16 [15] — VGG16 is a pre-trained model trained on 14 million image dataset belonging to 1000 different classes in ILSVR (ImageNet) challenge. In this architecture,2X2 filters with stride 1 are used for convolution operation and 2X2 filters with stride 2,same padding for max pooling operation across the network. Atthe end of architecture two fully connected dense layers of 4096 neurons are connected followed by softmax layer. We extracted features from two fully connected layers separately and used them for our experiments.

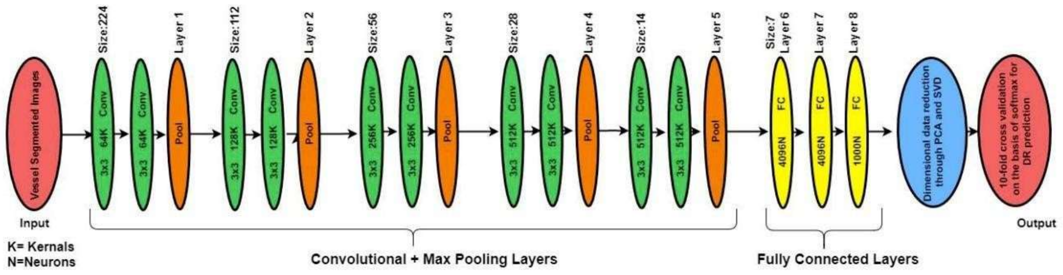


Figure 3.10 Architecture of VGG-16 Pre-trained model

3.3 Proposed Methodology

We propose deep learning-based approach to deep features of color fundus images obtained using different pre-trained models to detect and classify the severity level of retinopathy in diabetic patients. In the proposed method we do not need much pre- processing of the training data which reduces the time during testing. Instead of applying complex pre-processing steps we extract the features that are most descriptive. We traina Deep Neural Network with those features by introducing dropout at early layers of DNN.

Hand crafted features like GIST, HOG and SIFT (discussed in 3.1) gives global or localrepresentation of the image. Before the entry of deep learning models, the hand-crafted features were dominant and being widely used for feature extraction. Deep learning models do not require hand crafted features as the models themselves can

learn the important characteristics of the input images. This special capability of thedeep modelsmakes them representation models, as these models can represent the data efficiently. This reduces the conventional feature extraction phase where features are handcrafted. Deeper layers of these models can represent the entire given input in efficient way than the early layers. The downside of the deep learningmodels is that they need enormous amounts of data for training, which is usually scarce for most of the real-time applications. This problem can be addressed by theintroduction of transfer learning where the knowledge gained by a deep learning model can be transferred to other modelswithout any training. This is achieved usingpre-trained models. (discussed in 3.3)
The complete proposed model architecture is divided into two different modules namedfeature extraction, feature fusion and model training, evaluation.

3.3.1. Feature Extraction

During this phase, we extracted most significant features of gesture images using various pre-trained models (VGG16).We reshaped the Images according to the fixed input dimensions, accepted by these models.After passing images through VGG16,we collected the output of first and secondfully connected layers (fc1 & fc2) as features. For each input image we got a feature vector of dimension 4096 from thosetwo layers. Inception ResNet V2 produced a feature vectors of length 2048 and 1536respectively, when we pass the images through them. Figure 3.23 represents featureextraction of images of hand gestures from deeper layers of pre-trained ConvNet. The features extracted from pre- trained deep models are proved to be better than thehand-crafted features as these model'scan represent the images efficiently.

3.3.2 Model Training and Evaluation

During this phase, we train the model with pre-trained features and hybrid features for comparative studies. We apply trained model on test data, evaluate it's performance. We use a Deep Neural Network of 8 hidden layers each with 8,16,32,64,256, 128 units respectively and ReLU activation. Categorical Crossentropy loss with adam optimizer is preferred for these experiments. We addedone morehidden layerwith 512 units at the beginning of network for thetask of HandGesture Recogniton. After input layer we applied 0.2 dropout to avoid model from over-fitting of model. This helped the model to become robust.We used 80% of available data for training and preserved 20% for validation. Figure 3.24 representsthe architecture of proposed approach for model training and evaluation.

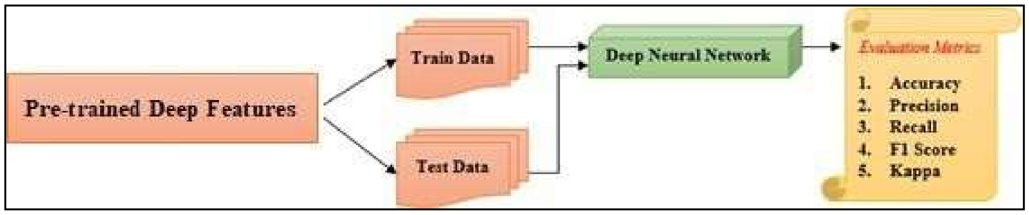


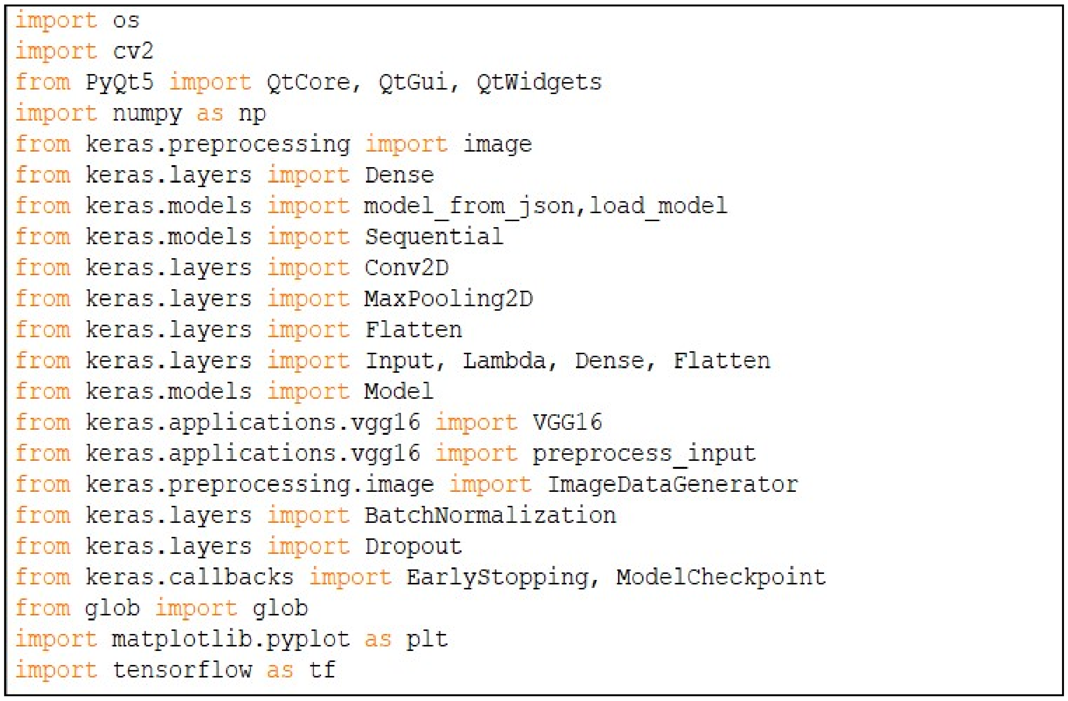
Fig 3. 11 Model Training and Evaluation

CHAPTER – 4
IMPLEMENTATION

4. IMPLEMENTATION

4.1 Implementation using CNN

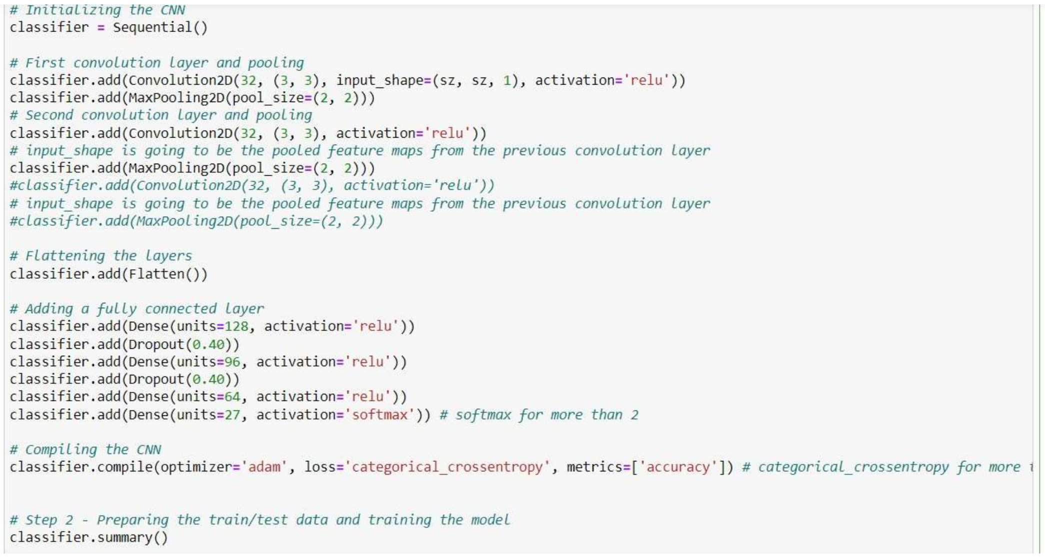
In this section we discuss how we have implemented the CNN Model to our datasetsand those functions and its uses are explained clearly in this section.



```
import os
import cv2
from PyQt5 import QtCore, QtGui, QtWidgets
import numpy as np
from keras.preprocessing import image
from keras.layers import Dense
from keras.models import model_from_json,load_model
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import BatchNormalization
from keras.layers import Dropout
from keras.callbacks import EarlyStopping, ModelCheckpoint
from glob import glob
import matplotlib.pyplot as plt
import tensorflow as tf
```

Fig 4.1 Import Statements

From Figure 4.1 we have imported all the necessary modules and layers fromkeras. The modules that we have imported are CV2, PyQt5, Numpy, Keras, Matplotlib, Tensorflow.



```
# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))
#classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
#classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2

# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) # categorical_crossentropy for more 


# Step 2 - Preparing the train/test data and training the model
classifier.summary()
```

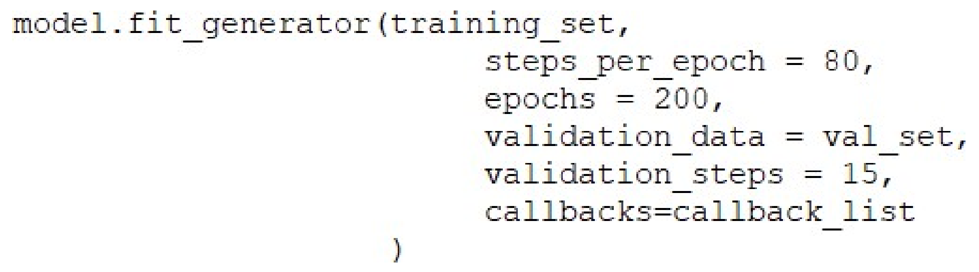
Fig 4.2 Implementation of CNN

In Figure 4.2 we have implemented CNN Model from Scratch by implementing sequential model. The sequential model creates an empty stack to add any number of convolutional layers we need. We have implemented 8 Convolutional layers and6 Max Pooling Layers and Fully connected layer and a dropout layer to help out incase of overfitting and finally an output layer consists of softmax function and number of classes as parameters.

The input size of an image is 256*256*1 as the image is Gray scale image and thuswe need only one channel, we have used ReLu as an activation function in all the hidden layers except output layer. The Max Pooling layer is used to reduce the sizeof the image so that we can have less number of computations and speedup the training process. The Dropout Layer is used inorder reduce the chances of overfitting.

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

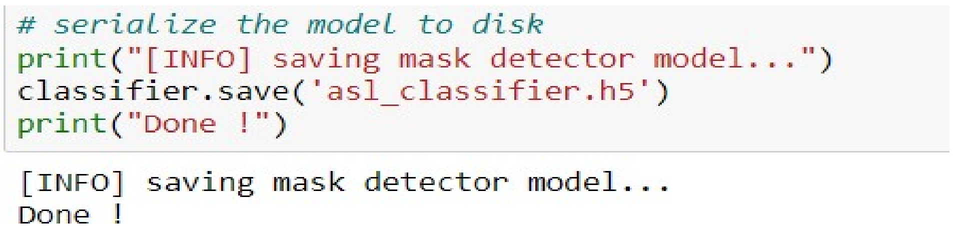
We have used Data Augmentation for CNN Model as the data which we have has limited number of datasets but these data cannot be sufficient for training and detecting the features from the gestures, if we have performed data augmentation tothe data then we can have different variations of data that can easily fit the model and easily train the model to obtain better features.



```
model.fit_generator(training_set,
                         steps_per_epoch = 80,
                         epochs = 200,
                         validation_data = val_set,
                         validation_steps = 15,
                         callbacks=callback_list
                    )
```

Fig 4.3 Fitting the Model

In Figure 4.3 we used fit_generator function to fit the model on the dataset we haveused and this function helps us to set the number of epoch we need to train and number of steps each epoch needs to take and the callbacks list will be taken as a parameter in this function to monitor the validation loss.



```
# serialize the model to disk
print("[INFO] saving mask detector model...")
classifier.save('asl_classifier.h5')
print("Done !")
```

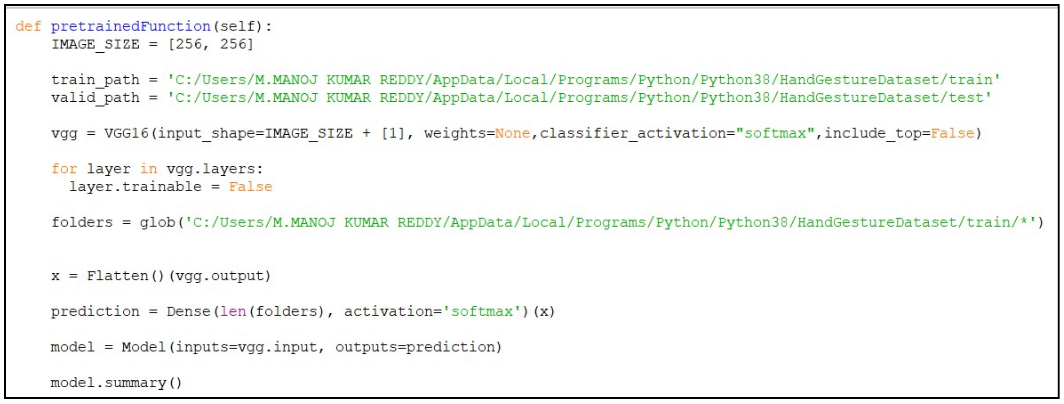
```
[INFO] saving mask detector model...
Done !
```

Fig 4.4 Saving the Model

In Figure 4.4 we have used the JSON Module to save the model and its weightsinto json file and h5 files respectively.

4.2 Implementation using Pre-Trained Model

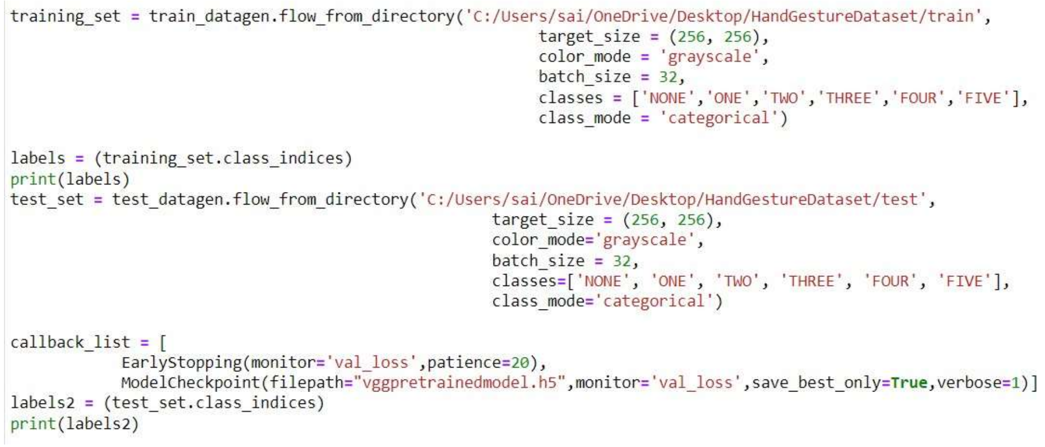
In this section we have discuss how we have implemented the pre-trained model andits summary.



```
def pretrainedFunction(self):
    IMAGE_SIZE = [256, 256]

    train_path = 'C:/Users/M.MANOJ KUMAR REDDY/AppData/Local/Programs/Python/Python38/HandGestureDataset/train'
    valid_path = 'C:/Users/M.MANOJ KUMAR REDDY/AppData/Local/Programs/Python/Python38/HandGestureDataset/test'

    vgg = VGG16(input_shape=IMAGE_SIZE + [1], weights=None,classifier_activation="softmax",include_top=False)

    for layer in vgg.layers:
      layer.trainable = False

    folders = glob('C:/Users/M.MANOJ KUMAR REDDY/AppData/Local/Programs/Python/Python38/HandGestureDataset/train/*')


    x = Flatten()(vgg.output)

    prediction = Dense(len(folders), activation='softmax')(x)

    model = Model(inputs=vgg.input, outputs=prediction)

    model.summary()
```

Fig 4.5 Implementation of VGG-16 (Pre-Trained Model)

Here we have used VGG-16 as a Pre-Trained Model and imported all the necessarymodules of VGG-16 from Keras.applications. The parameters for the VGG-16 are image size, weights,include_top. The image size for normal VGG-16 is 224*224*3but we have only Gray Scale images so we have resized them to 256*256*1 and fora normal VGG-16 Model the weights are assigned from the imagenet but we have different classes in our dataset to we have given to None and include_top attribute is set to False as we don’t need the output layers from VGG-16 and we want to change those layers according to our problem statement. We are not even training all the layers of the VGG-16 and we have set the layers.trainable to False and we will train only few layers which are required and set the remaining layers to freeze.



```
training_set = train_datagen.flow_from_directory('C:/Users/sai/OneDrive/Desktop/HandGestureDataset/train',
                                                 target_size = (256, 256),
                                                 color_mode = 'grayscale',
                                                 batch_size = 32,
                                                 classes = ['NONE','ONE','TWO','THREE','FOUR','FIVE'],
                                                 class_mode = 'categorical')

labels = (training_set.class_indices)
print(labels)
test_set = test_datagen.flow_from_directory('C:/Users/sai/OneDrive/Desktop/HandGestureDataset/test',
                                            target_size = (256, 256),
                                            color_mode='grayscale',
                                            batch_size = 32,
                                            classes=['NONE', 'ONE', 'TWO', 'THREE', 'FOUR', 'FIVE'],
                                            class_mode='categorical')

callback_list = [
            EarlyStopping(monitor='val_loss',patience=20),
            ModelCheckpoint(filepath="vggpretrainedmodel.h5",monitor='val_loss',save_best_only=True,verbose=1)]
labels2 = (test_set.class_indices)
print(labels2)
```

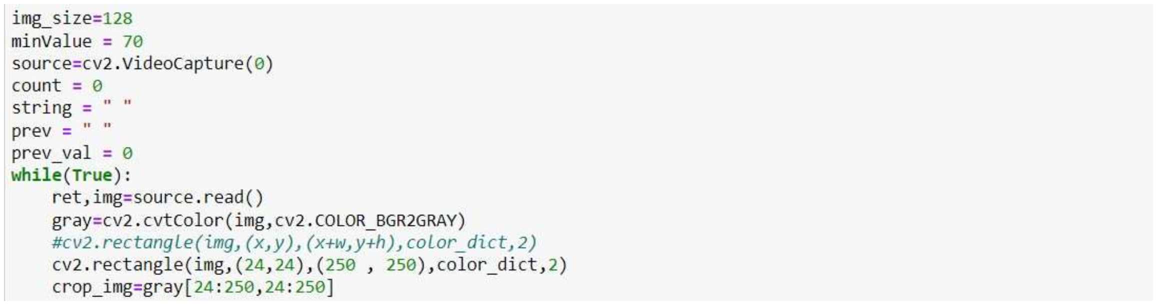
Fig 4.6 Training of Model

The rest of the code is same as like which we have done in implementation of CNNModel.

4.3 Implementation of OpenCV and GUI Functions

In this section we discuss about the implementation of OpenCV Functions and GUI .

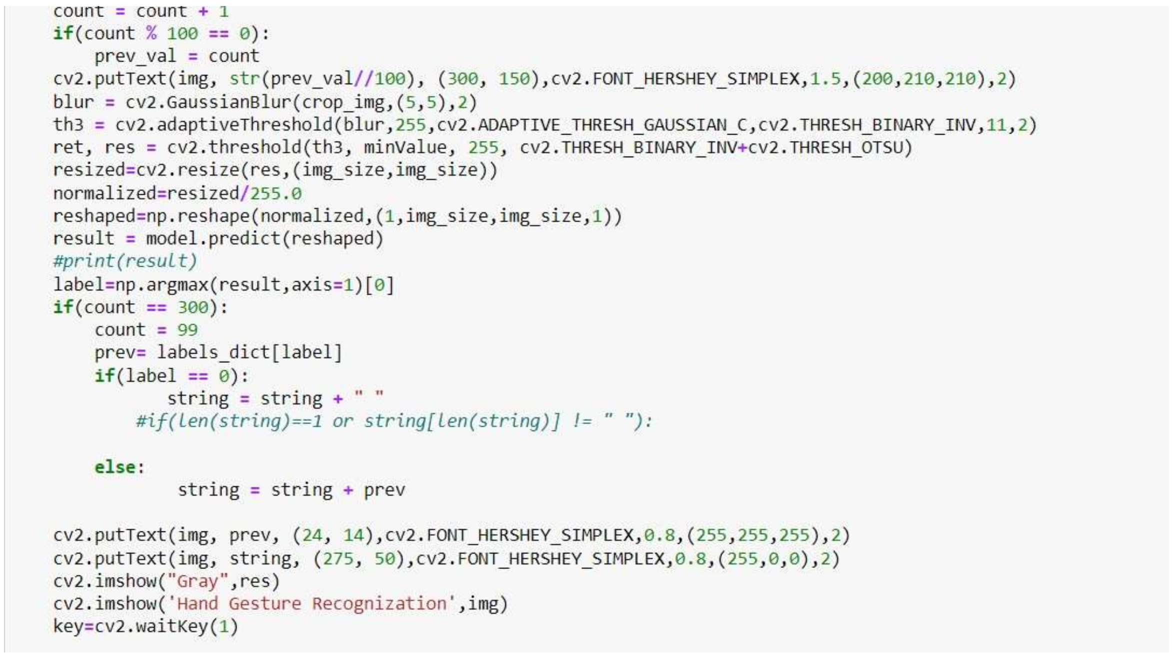
4.3.1 OpenCV Implementation



```
img_size=128
minValue = 70
source=cv2.VideoCapture(0)
count = 0
string = " "
prev = " "
prev_val = 0
while(True):
    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    #cv2.rectangle(img,(x,y),(x+w,y+h),color_dict,2)
    cv2.rectangle(img,(24,24),(250 , 250),color_dict,2)
    crop_img=gray[24:250,24:250]
```

Fig 4.7 Preprocessing using OpenCV

Here we have imported all the necessary modules from CV2. The VideoCapture()is used to start the camera to trace the hand and draw the rectangle over the detected range. In Figure 4.9 we have converted the Colored image to Gray scale image using cvtcolor() function and then we have applied different Blurring and Thresholding functions to preprocess the image to identify the gestures.

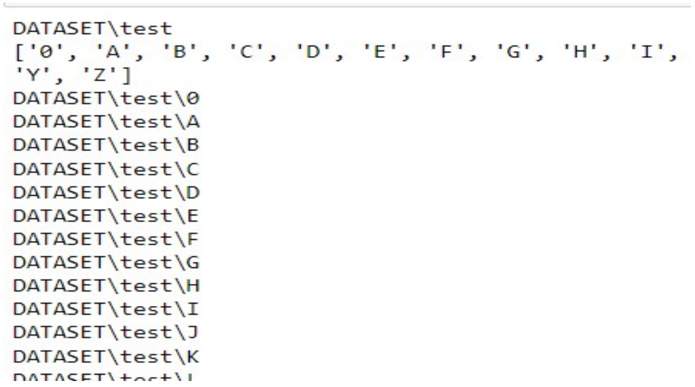


```
count = count + 1
if(count % 100 == 0):
    prev_val = count
cv2.putText(img, str(prev_val//100), (300, 150),cv2.FONT_HERSHEY_SIMPLEX,1.5,(200,210,210),2)
blur = cv2.GaussianBlur(crop_img,(5,5),2)
th3 = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
ret, res = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
resized=cv2.resize(res,(img_size,img_size))
normalized=resized/255.0
reshaped=np.reshape(normalized,(1,img_size,img_size,1))
result = model.predict(reshaped)
#print(result)
label=np.argmax(result,axis=1)[0]
if(count == 300):
    count = 99
    prev= labels_dict[label]
    if(label == 0):
           string = string + " "
        #if(len(string)==1 or string[len(string)] != " "):

    else:
            string = string + prev

cv2.putText(img, prev, (24, 14),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
cv2.putText(img, string, (275, 50),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,0,0),2)
cv2.imshow("Gray",res)
cv2.imshow('Hand Gesture Recognization',img)
key=cv2.waitKey(1)
```

Fig 4.8 Finding Contours and detecting Convexity Defects

Here we have used findContours function to identify the contours and to connect those contours. The ConvexHull function is used to identify the hull in the gesture and the defects in the convex Hull are known as convexity Defects that are used toclassify the type of gesture. We have used some mathematical functions to identifythe angle between the defects.

4.3.2 Dataset Creation Code

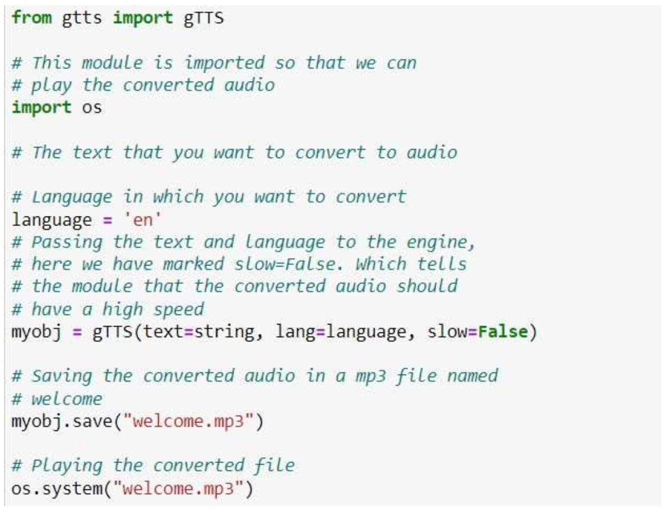


```
DATASET\test
['0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'Y', 'Z']
DATASET\test\0
DATASET\test\A
DATASET\test\B
DATASET\test\C
DATASET\test\D
DATASET\test\E
DATASET\test\F
DATASET\test\G
DATASET\test\H
DATASET\test\I
DATASET\test\J
DATASET\test\K
```

Fig 4.9 Dataset Creation

4.4 Conversion of Text to Speech Using gTTS Module

gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Writes spoken mp3 data to a file, a file- like object (bytestring) for further audio manipulation



```
from gtts import gTTS

# This module is imported so that we can
# play the converted audio
import os

# The text that you want to convert to audio

# Language in which you want to convert
language = 'en'
# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=string, lang=language, slow=False)

# Saving the converted audio in a mp3 file named
# welcome
myobj.save("welcome.mp3")

# Playing the converted file
os.system("welcome.mp3")
```

Fig 4.10 Converting Text to Speech

CHAPTER - 5

EXPERIMENTAL RESULTS

5. EXPERIMENTAL RESULTS

In this Chapter, we give the details of the experimental studies we have performed to understand the efficiency of the proposed method. First, we introduce the datasetused for our studies followed by the performance of different Deep Learning modelsand Pre-Trained Models. Later we present the results of Computer Vision Functionswhich we have used to classify the Hand Gestures.

5.1 Dataset Summary

To prove the efficiency of the proposed CNN features, we have used the benchmarkdataset available at Kaggle. We collected the data from Hand Gesture Recognition Database with CNN Challenge in Kaggle website. The dataset is a large set near Infrared images of distinct hand gestures under variety of imaging conditions. The dataset consists of nearly 13000 samples.

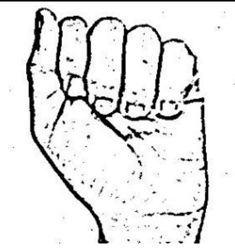
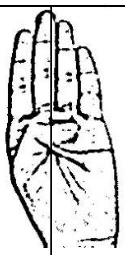
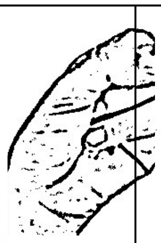
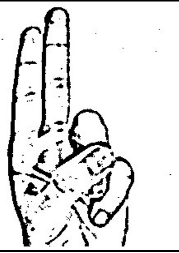
			
A	**B**	**J**	**U**

Fig 5.1 Sample images from dataset

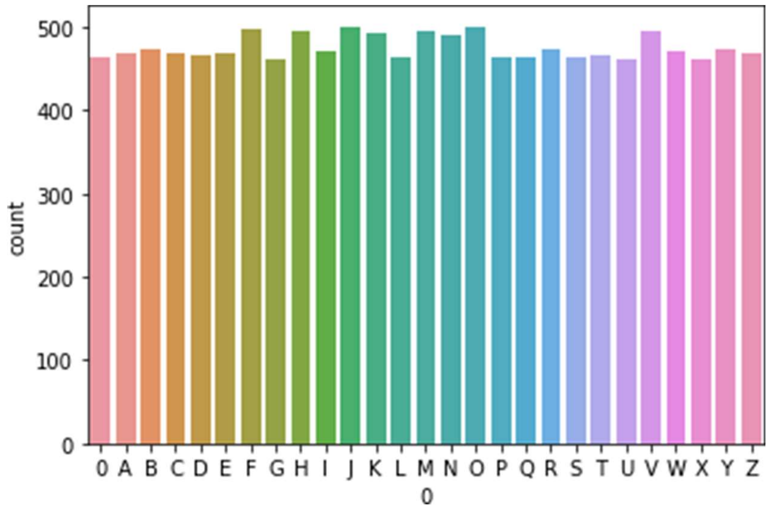


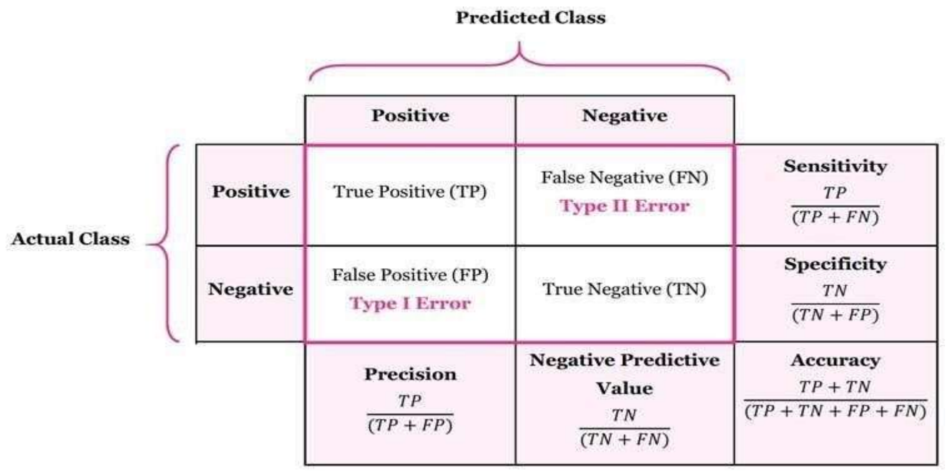
Fig 5.2 Dataset Summary

5.2 Results Analysis

In this section first, we discuss some classification metrics that we used to evaluatethe performance of Deep Neural Network. Next, we present the performance of Deep Neural Network when trained using Pre-Trained Features for Hand Gesture Recognition. Finally, we presented the necessary requirements to fulfil the successful completion of the project.

5.2.1 Evaluation Metrics

Confusion Matrix: A confusion matrix is an N X N matrix, where N is the numberof classes being predicted. The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It contains number of True Positive (TP), True Negative (TN), False Positive (FP) andFalse Negative (FN).



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Fig 5.3 Confusion Matrix.

Using this Confusion Matrix, we can calculate the following matrices:

- **Accuracy**: the proportion of the total number of predictions that were correct.
- **Positive Predictive Value (or) Precision**: the proportion of positive casesthat werecorrectly identified.
- **Negative Predictive Value:** the proportion of negative cases that were correctlyidentified.
- **Sensitivity (or) Recall:** the proportion of actual positive cases which are correctlyidentified.

- **Specificity:** the proportion of actual negative cases which are correctly identified.
- **F1 Score:** the harmonic mean of precision and recall values for a classificationproblem.

$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Kappa Statistic: It compares an observed accuracy with an expected accuracy.Equationfor computing Kappa Statistic is given by:

$$KappaStatistic = \frac{(Observed\ Accuracy - Expected\ Accuracy)}{(1 - Expected\ Accuracy)}$$

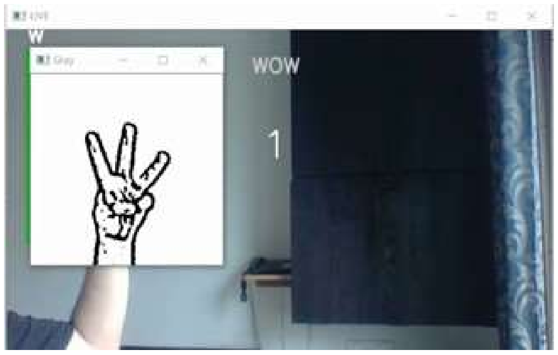
Experiments with different Deep Learning Models over Hand Gesture

EXPERIMENTAL RESULTS

Model	**Conv Layers**	**Training**		**Testing**		**Epochs**
		Accuracy	Loss	Accuracy	Loss	
CNN	8	99.50	0.0232	97.08	0.1149	50
CNN	5	98.35	0.0941	96.88	0.1066	50
VGG- 16	3	92.51	0.2389	84.79	0.6969	33

Performance of different models

Fig 5.4 Model Recognition of word ‘WOW’



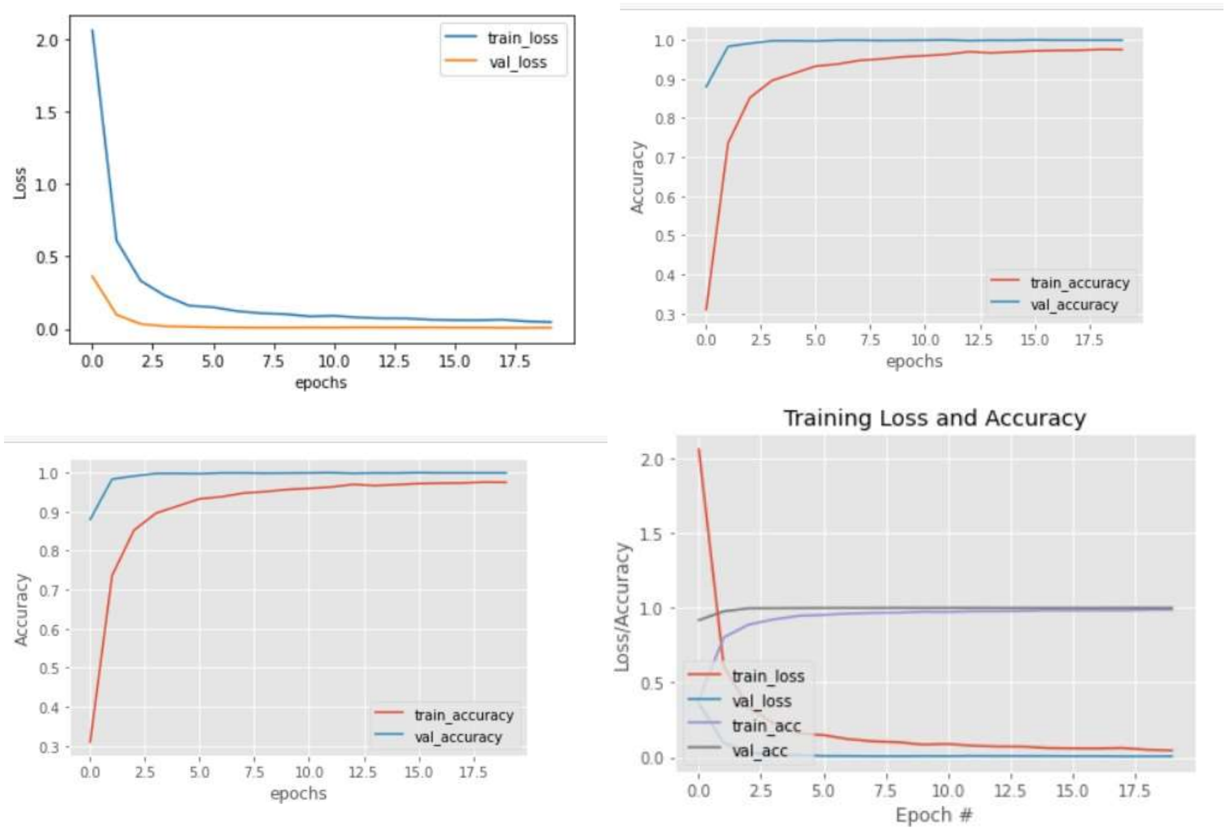


Fig 5.5 Different Plots between Accuracy/Loss and Epochs

CHAPTER – 6
CONCLUSION

6. CONCLUSION

The objective of this work is to develop a Hand Gesture Recognition model from thegiven distinct gestures. Major features of the proposed model are simple and robust.The model we designed extracts the deep features from pre- trained models like VGG16 and CNN to represent the more prominent features from the Gesture images and these are used for training a neural network. To avoid over-fitting of the modelwe introduced dropout at the ending of the neural network. The power of deep learning and simplicity of the models make the model more robust with minimum number of misclassifications.

6.1 Future Works

Recognition of gestures made with both hands is not possible by this system. Therefore, another future work can be the recognition of gestures made with both hands. We can implement the same algorithm with different kinds of datasets that are available to have Robust system and to obtain good results. We can implement more Deep Learning Algorithms so that we can compare which algorithm works good for these kinds of applications.

REFERENCES

1. E. Stergiopoulou and N. Papamarkos, "Hand gesture recognition using a neural network shape fitting technique," Engineering Applications of Artificial Intelligence, vol. 22, no. 8, pp. 1141–1158, 2009.
2. Breuer, P., Eckes, C. and Muller, S., 2007. Hand gesture recognition with a novel IR time-of-flight range camera - a pilot study, Proceedings, 28-30 March2007 2007, Springer-Verlag pp247-60.
3. Fakhreddine Karray, Milad Alemzadeh, Jamil Abou Saleh, Mo Nours Arab, (2008). "Human Computer Interaction: Overview on State of the Art", International Journal on Smart Sensing and Intelligent Systems, Vol. 1(1).
4. Hervé Lahamy and Derek Litchi, "REAL-TIME HAND GESTURE RECOGNITION USING RANGE CAMERAS".
5. Mokhtar M. Hasan, Pramoud K. Misra, (2011). "Brightness Factor Matching for Gesture Recognition System Using Scaled Normalization", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3(2).
6. Pei Xu, "A Real-time Hand Gesture Recognition and Human-Computer Interaction System".
7. T.-N. Nguyen, H.-H. Huynh, and J. Meunier, "Static hand gesture recognitionusing artificial neural network," Journal of Image and Graphics, vol. 1, no. 1,pp. 34–38, 2013.
8. Q. Chen, N. D. Georganas, and E. M. Petriu, "Hand gesture recognition usinghaar-like features and a stochastic context-free grammar," IEEE transactions on instrumentation and measurement, vol. 57, no. 8, pp. 1562–1571, 2008.
9. Md. Zahirul Islam, Mohammad Shahadat Hossain, Raihan Ul Islam, Karl Andersson, "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation", FULLTEXT01 (diva-portal.org).
10. Paulo Trigueiros, Fernando Ribeiro and Luís Paulo Reis, "Hand Gesture Recognition System based in Computer Vision and Machine Learning".
11. Rafiqul Zaman Khan and Noor Adnan Ibraheem, "HAND GESTURE RECOGNITION: A LITERATURE REVIEW", International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, July 2012.
12. Munir Oudah, Ali Al-Naji and Javaan Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques", J. Imaging 2020, 6, 73;doi:10.3390/jimaging6080073.
13. Aashni Haria, Archanasri Subramanian, Nivedhitha Asokkumar, Shristi Poddar, Jyothi S Nayak, "Hand Gesture Recognition for Human Computer Interaction", 7th International Conference on Advances in Computing & Communications, ICACC-2017, 22- 24 August 2017, Cochin, India.
14. Mais Yasen and Shaidah Jusoh, "A systematic review on hand gesture recognition techniques, challenges and applications", PeerJ Comput. Sci. 5:e218 http://doi.org/10.7717/peerj-cs.218.
15. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
16. F. Chollet, "Xception: Deep learning with depthwise separable convolutions,"in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251– 1258, 2017

17. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for imageclassification: A comprehensive review. *Neural computation*, *29*(9), 2352- 2449.
18. Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neuralnetworkmodels for practical applications. *arXiv preprint arXiv:1605.07678*.
19. Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.
20. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41-48).
21. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable arefeatures in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of theIEEE conference on computer vision and pattern recognition* (pp. 1-9).
23. Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.
24. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deepconvolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
25. Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understandingconvolutional networks. In *European conference on computer vision* (pp. 818-833). Springe