A Project Report
on

SIGN LANGUAGE TRANSLATOR FOR SPEECH-IMPAIRED

submitted in partial fulfillment of the requirements for the award of the degree
of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0516	Ms. S.SAMIKSHA
17WH1A0536	Ms. D.TURIYA
17WH1A0545	Ms. T.NEELIMA

under the esteemed guidance of
Mrs. Naga Kalyani.Ayyadevara
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

May,2021

DECLARATION

We hereby declare that the work presented in this project entitled **"SIGN LANGUAGE TRANSLATOR FOR SPEECH-IMPAIRED"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Mrs.Naga kalyani.Ayyadevara, Assistant Professor, Department of CSE.

Sign. with date:
Ms. S.SAMIKSHA
(17WH1A0516)

Sign. with date:
Ms. D.TURIYA
(17WH1A0536)

Sign. with date:
Ms. T.NEELIMA
(17WH1A0545)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**SIGN LANGUAGE TRANSLATOR FOR SPEECH-IMPAIRED**" is a bonafide work carried out by Ms. S.SAMIKSHA (17WH1A0516) ; Ms. D.TURIYA (17WH1A0536) ; Ms. T.NEELIMA (17WH1A0545) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. K.Srinivasa Reddy
Professor and HoD,
Department of CSE

Guide
Mrs.Naga Kalyani.Ayyadevara
Assistant Professor

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K.Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mrs. Naga Kalyani.Ayyadevara, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. S.SAMIKSHA

(17WH1A0516)

Ms. D.TURIYA

(17WH1A0536)

Ms. T.NEELIMA

(17WH1A0545)

TABLE OF CONTENTS

ABSTRACT

Inability to speak is considered to be true disability. People with this disability use different modes to communicate with others, there are number of methods available for their communication one such common method of communication is sign language.

Developing sign language application for deaf people can be very important, as they will be able to communicate easily with even those who don't understand sign language. This project aims at taking the basic step in bridging the communication gap between normal people, deaf and dumb people using sign language.

The main focus of this work is to create a vision based system to identify sign language gestures from the video sequences.

Sign language translation system which utilizes deep learning based convolutional neural network is proposed. The implemented sign language translator comprises of feature vector extraction using deep learning structure based on inception v3 learning technique. After feature vector extraction, the extracted vector is supplied to the classifier. Transfer learning on deep learning models (Inception v3) has been performed. Sign Language Translator can constructively translate detected hand gestures into text. From the implemented learning 88% accuracy is attained.

LIST OF FIGURES

1. INTRODUCTION

Motion of any body part like hand is a form of gesture. For gesture recognition image processing and computer vision is used. Gesture recognition enables computer to understand human actions and also acts as an interpreter between computer and human. This could provide potential to human to interact naturally with the computers without any physical contact of the mechanical devices. Gestures are performed by the deaf and dumb community to perform sign language. This community uses sign language for their communication as broadcasting audio is impossible.

Normally sign language is used by everyone when they do not want to speak, but this is the only way of communication for deaf and dumb community. Sign language is also serving the same meaning as spoken language does. This is used by deaf and dumb community all over the world but in their regional form like ISL, ASL. Sign language can be performed by using Hand gesture either by one hand or two hands. It is of two type Isolated sign language and continuous sign language. Isolated sign language consists of single gesture having single letter while continuous ISL or Continuous Sign language is a sequence of gestures that generate a meaningful sentence. This report contains isolated ASL gesture recognition technique.

1.1 Sign Language

Dumb people around the world communicate using sign language as distinct from spoken language in their every day. It is a visual language that uses a system of hand gestures as the means of communication. Sign language is not an universal language, and different sign languages are used in different countries, like the many spoken languages all over the world. Some countries such as Belgium, the UK, the USA or India may have more than one sign language. Hundreds of sign languages are in used around the world, for instance, Japanese Sign Language, British Sign Language (BSL), Spanish Sign Language, Turkish Sign Language.

Sign language is a visual language and consists of 3 major components:

Fingerspelling	**Word level sign vocabulary**	**Non-manual features**
Used to spell words letter by letter .	Used for the majority of communication.	Facial expressions and tongue, mouth and body position.

Fig 1.1:American sign language letters

1.2 Objectives

Although a certain fraction of the world suffers from speech and hearing disabilities, Sign Language is not a wide spread language across the world.

Therefore, to aid with a smoother communication between speaking and non-speaking world, technical development can be introduced. This project proposes an application for it.

1.3 Methodology

To detect the alphabet representing a specific gesture. Sign language we are using is American Sign Language. In this section, methodology followed is discussed in detail.

1.3.1 Dataset

The network was trained on kaggle dataset of ASL Alphabet.[3]

About

The data set is a collection of images of alphabets from the American Sign Language, separated in 29 folders which represent the various classes.

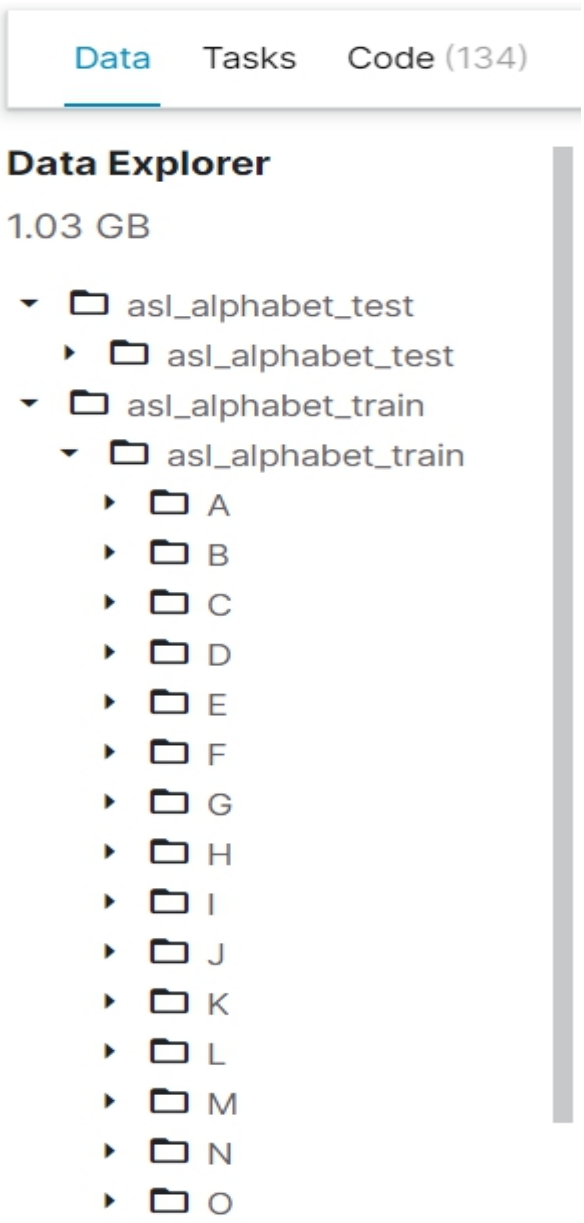


Fig 1.3.1.1: Classes of Dataset

Content

The dataset taken from Kaggle has asl_alphabet_train folder which is training data and asl_alphabet_test folder which is testing data.

Fig 1.3.1.2:Sign Gestures of A & O

1.3.2 CONVOLUTIONAL NEURAL NETWORK

ARCHITECTURE:

CNN image classifications takes an input image, process it and classify it under certain categories (Eg: A, B, C). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image. Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN[2] to process an input image and classifies the objects based on values.

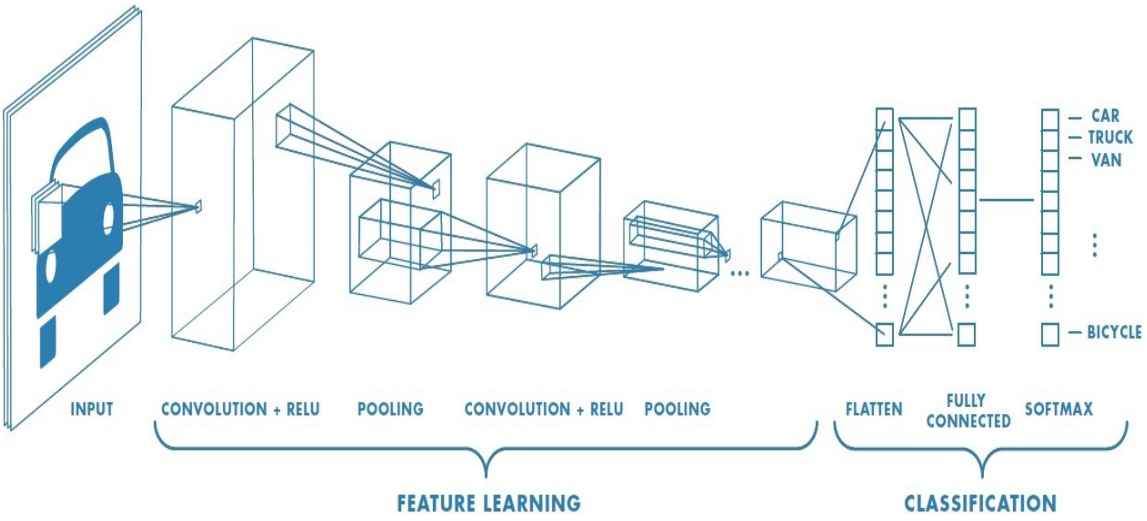


Fig 1.3.2.1: Convolution Neural Network

CONVOLUTIONAL LAYERS:

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. All neurons within a feature map have

weights that are constrained to be equal. However different feature maps within the same convolutional layer have different have different weights so that several features can be extracted at each location.

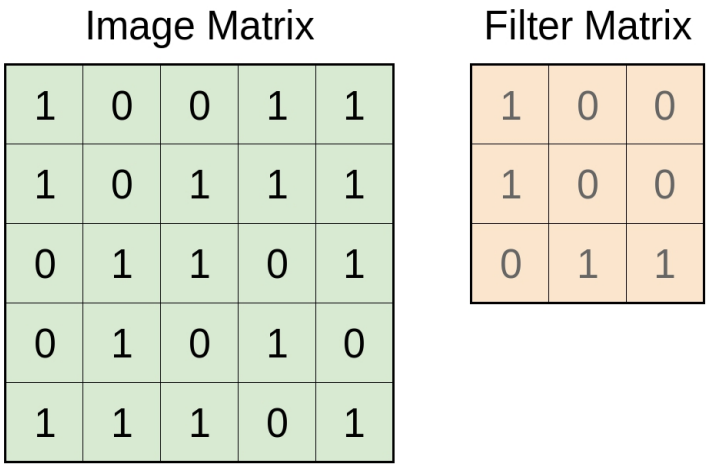


Fig 1.3.2.2: Image Matrix and Filter Matrix

STRIDES:

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

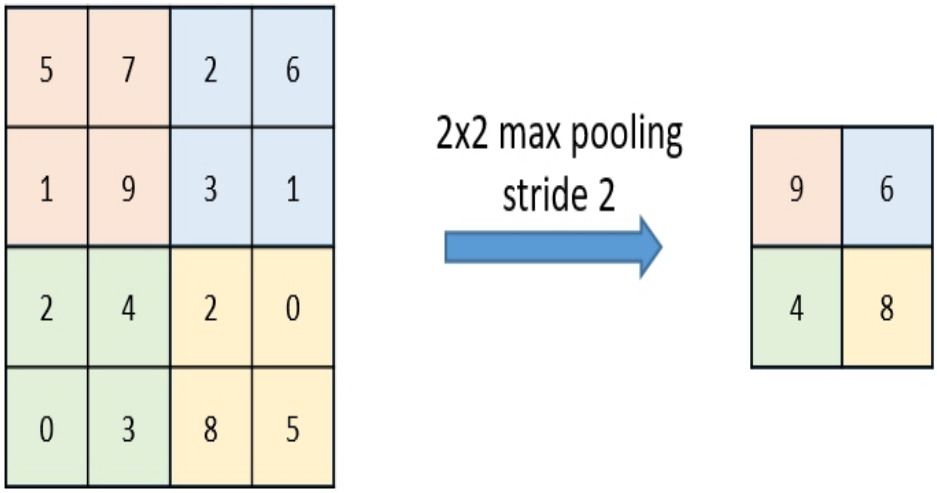


Fig 1.3.2.3:Strides

PADDING:

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

NON LINEARITY (ReLU):

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

POOLING LAYERS:

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. Initially, it was common practice to use average pooling aggregation layers to propagate the average of all the input values, of a small neighborhood of an image to the next layer. However, in more recent models, max pooling aggregation layers propagate the maximum value within a receptive field to the next layer.

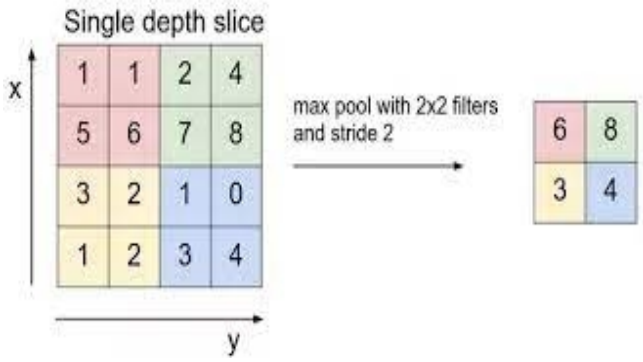


Fig 1.3.2.4: Max pool with 2x2 filters

FULLY CONNECTED LAYERS:

Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level reasoning. . For classification problems, it is standard to use the softmax operator on top of a CNN.

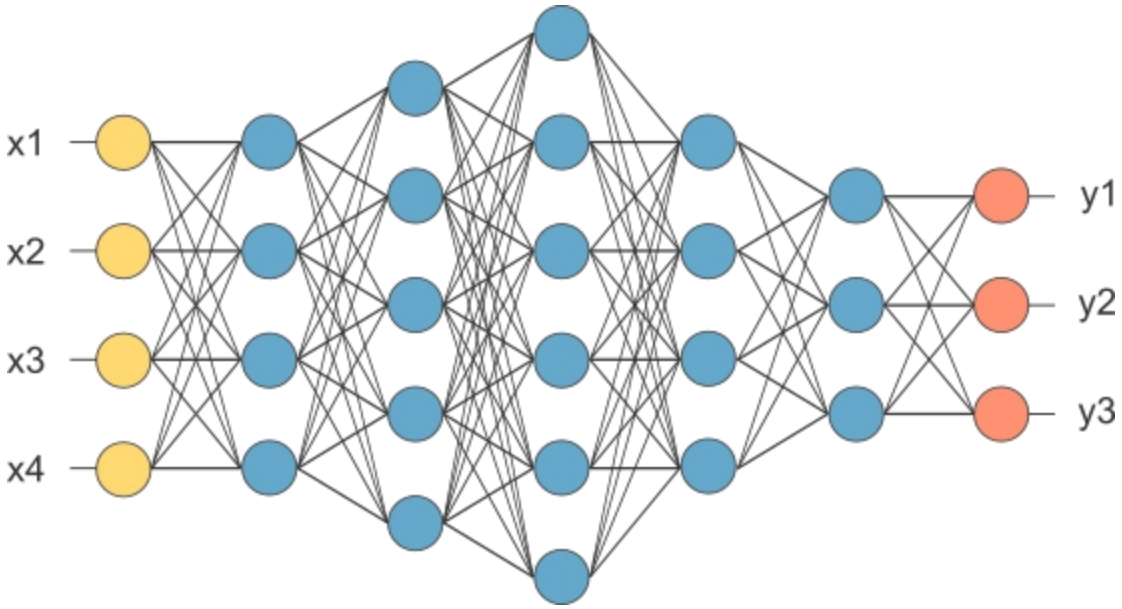


Fig 1.3.2.5: Fully Connected Layers

SOFTMAX FUNCTION:

Softmax function calculates the probabilities distribution of the event over n different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability

The formula computes the exponential (e-power) of the given input value and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function.

1.3.3 Inceptionv3

Inception network is considered a state-of-the-art deep learning architecture (or model) for solving image recognition and detection problems.

It put forward a breakthrough performance on the ImageNet Visual Recognition Challenge (in 2014), which is a reputed platform for benchmarking image recognition and detection algorithms. Along with this, it set off tons of research in the creation of new deep learning architectures with innovative and impactful ideas.

Objective

There's a simple but powerful way of creating better deep learning models. You can just make a bigger model, either in terms of deepness, i.e., number of layers, or the number of neurons in each layer. But as you can imagine, this can often create complications:

- Bigger the model, more prone it is to overfitting. This is particularly noticeable when the training data is small.
- Increasing the number of parameters means you need to increase your existing computational resources.

A solution for this, is to move on to sparsely connected network architectures which will replace fully connected network architectures, especially inside convolutional layers. This idea can be conceptualized in the below images:

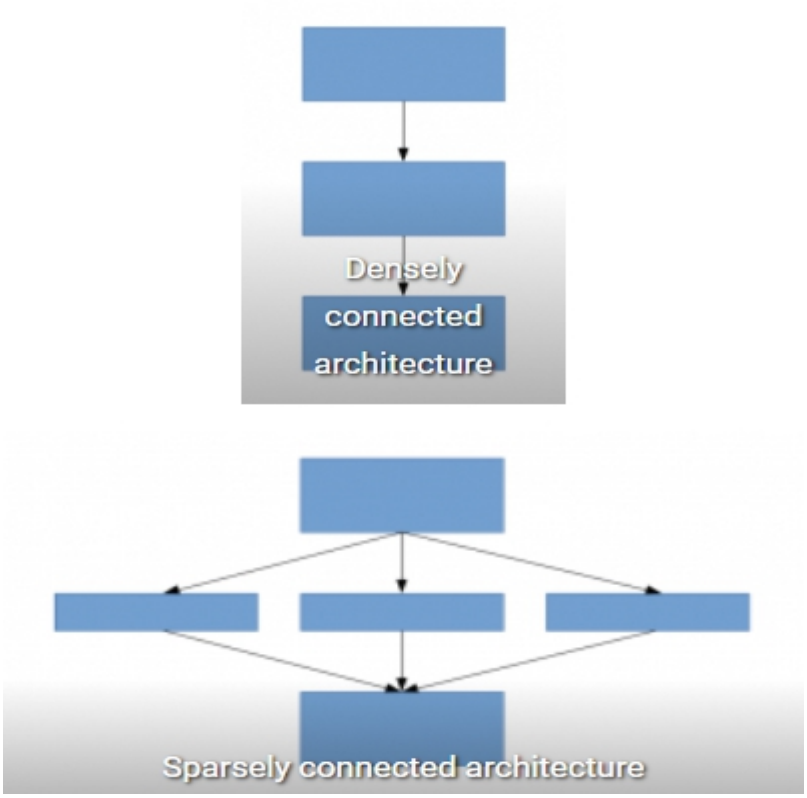


Fig 1.3.3.1: Densely and Sparsely Connected Architecture

This proposes a new idea of creating deep architectures. This approach lets you maintain the "computational budget", while increasing the depth and width of the network.

Inception v1:

It is basically a convolutional neural network (CNN) which is 27 layers deep. Below is the model

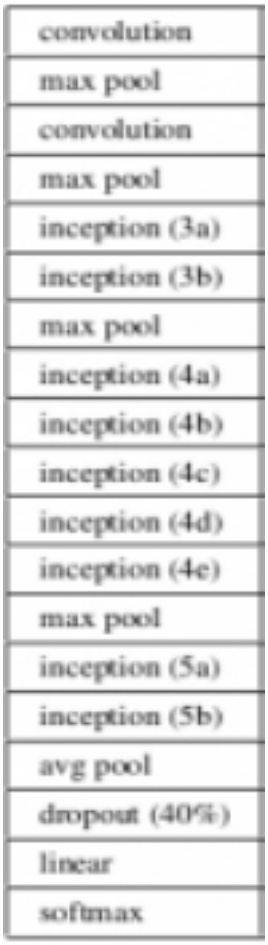


Fig 1.3.3.2: Layers

In the above image that there is a layer called inception layer. This is actually the main idea behind the paper's approach. The inception layer is the core concept of a sparsely connected architecture.

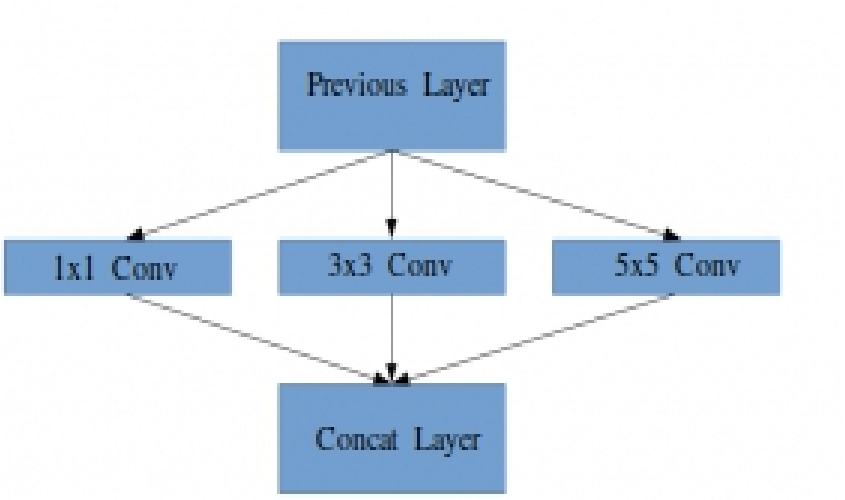


Fig 1.3.3.3: Inception Layer

Inception Layer is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.

Along with the above-mentioned layers, there are two major add-ons in the original inception layer:

- 1×1 Convolutional layer before applying another layer, which is mainly used for dimensionality reduction.
- A parallel Max Pooling layer, which provides another option to the inception layer.

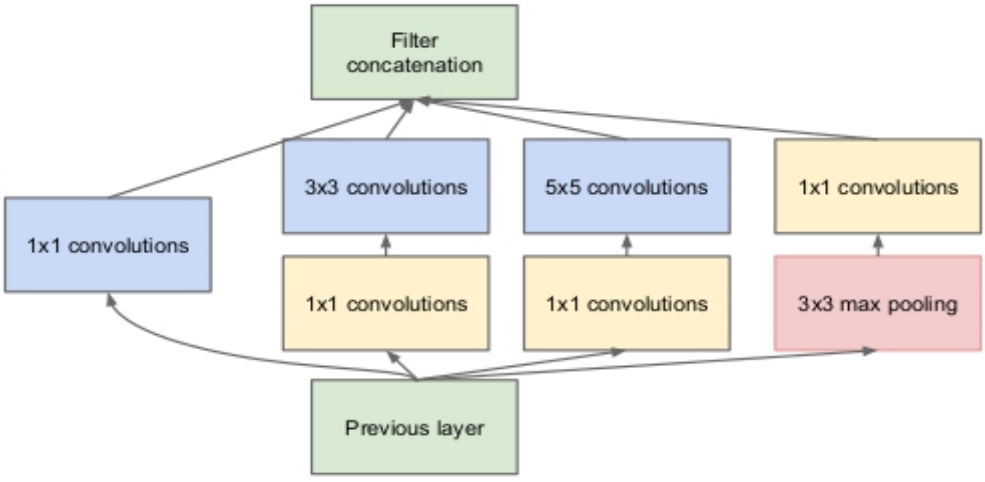


Fig 1.3.3.4: Layer Structure

When creating a subsequent layer in a deep learning model, one should pay attention to the learnings of the previous layer.

Suppose, for example, a layer in our deep learning model has learned to focus on individual parts of a face. The next layer of the network would probably focus on the overall face in the image to identify the different objects present there. Now to actually do this, the layer should have the appropriate filter sizes to detect different objects.

This is where the inception layer comes to the fore. It allows the internal layers to pick and choose which filter size will be relevant to learn the required information. So even if the size of the face in the image is different, the layer works accordingly to recognize the face.

The auxiliary training done: To prevent the middle part of the network from "dying out", the authors introduced two auxiliary classifiers. They essentially applied softmax to the outputs of two of the inception modules, and computed an auxiliary loss over the same labels. The total loss function is a weighted sum of the auxiliary loss and the real loss.

"The Inception deep convolutional architecture was introduced as GoogLeNet in (Szegedy et al. 2015a), here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Ioffe and Szegedy 2015) (Inception-v2). Later by additional factorization ideas in the third iteration (Szegedy et al. 2015b) which will be referred to as Inception-v3."[4]

Factorizing Convolutions

The aim of factorizing Convolutions is to reduce the number of connections/parameters without decreasing the network efficiency.

Factorization Into Smaller Convolutions

Two 3×3 convolutions replaces one 5×5 convolution as follows:

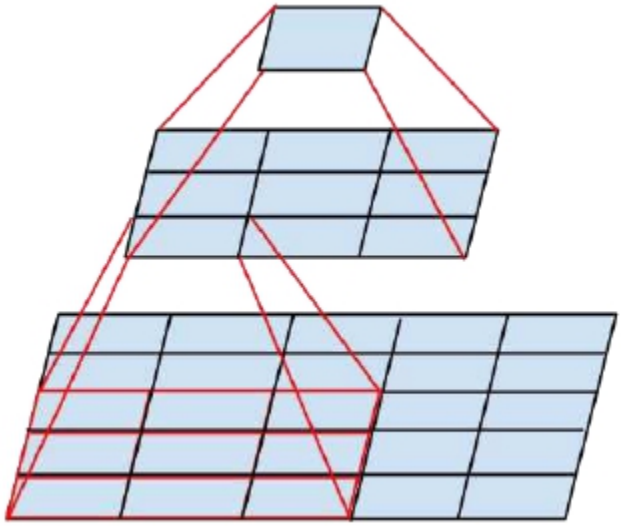


Fig 1.3.3.5: Two 3x3 convolutions replacing one 5x5

By using 1 layer of 5×5 filter, number of parameters = 5×5=25

By using 2 layers of 3×3 filters, number of parameters = 3×3+3×3=18

Number of parameters is reduced by 28%

With this technique, one Inception module (**Inception Module A**) becomes:

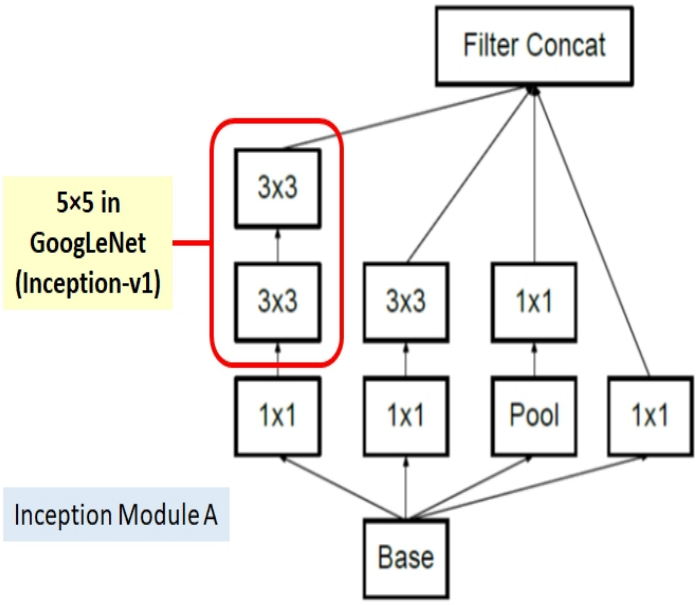


Fig 1.3.3.6: Inception Module A

Factorization Into Asymmetric Convolutions

One 3×1 convolution followed by one 1×3 convolution replaces one 3×3 convolution as follows:

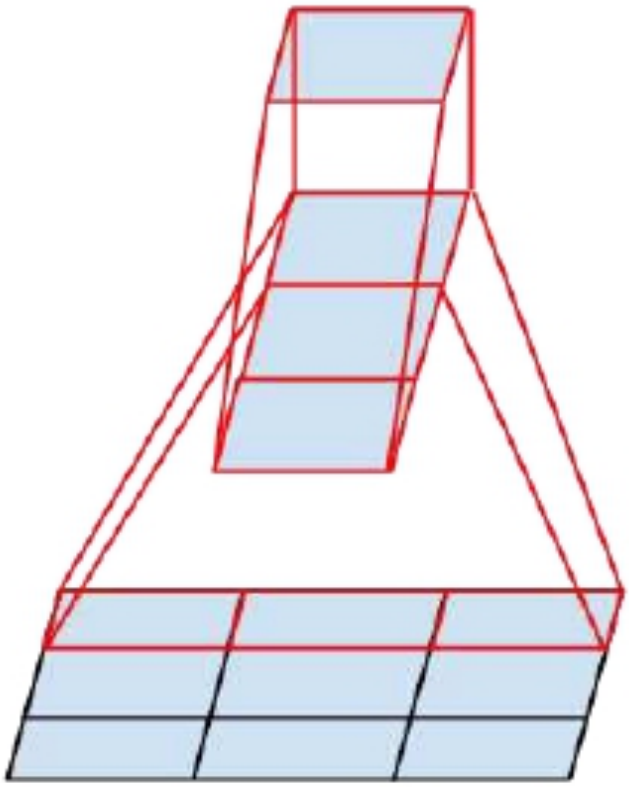


Fig 1.3.3.7:One 1x3 & one 3x1 convolutions replaces one 3x3

By using **3×3 filter**, number of parameters = **3×3=9**

By using 3×1 and 1×3 filters, number of parameters = **3×1+1×3=6**

Number of parameters is reduced by 33%

With this technique, another Inception module (**Inception Module B**) becomes:

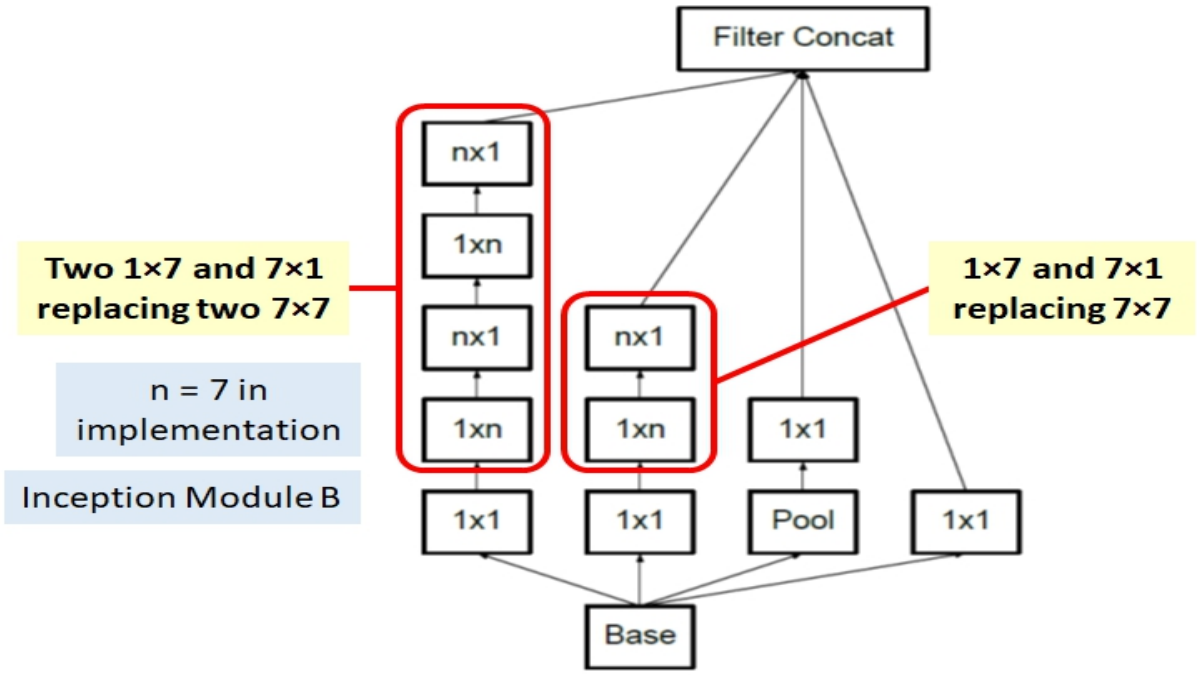


Fig 1.3.3.8: Inception Module B using asymmetric factorization

And **Inception module C** is also proposed for promoting high dimensional representations :

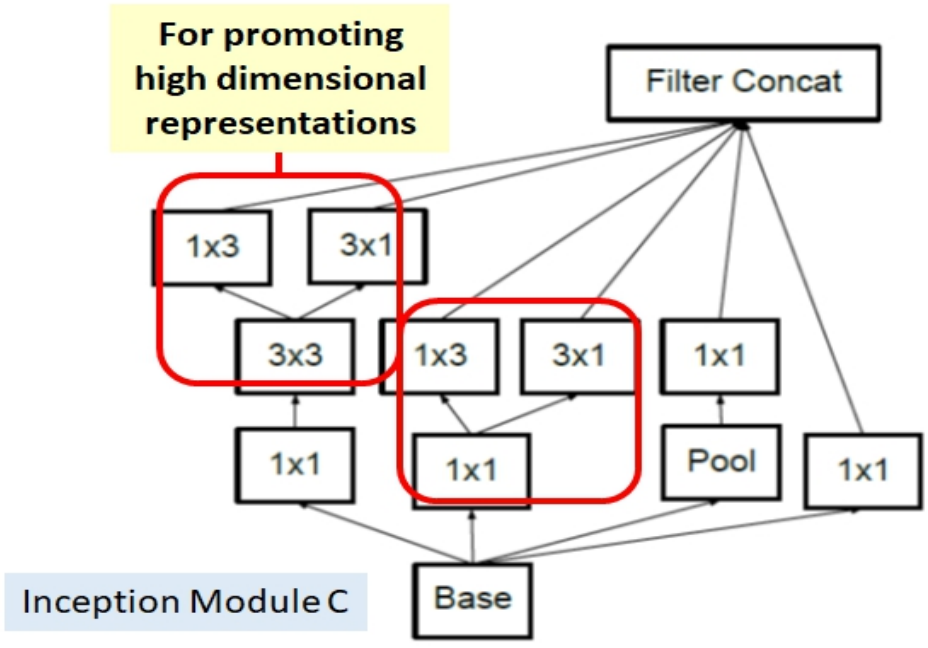


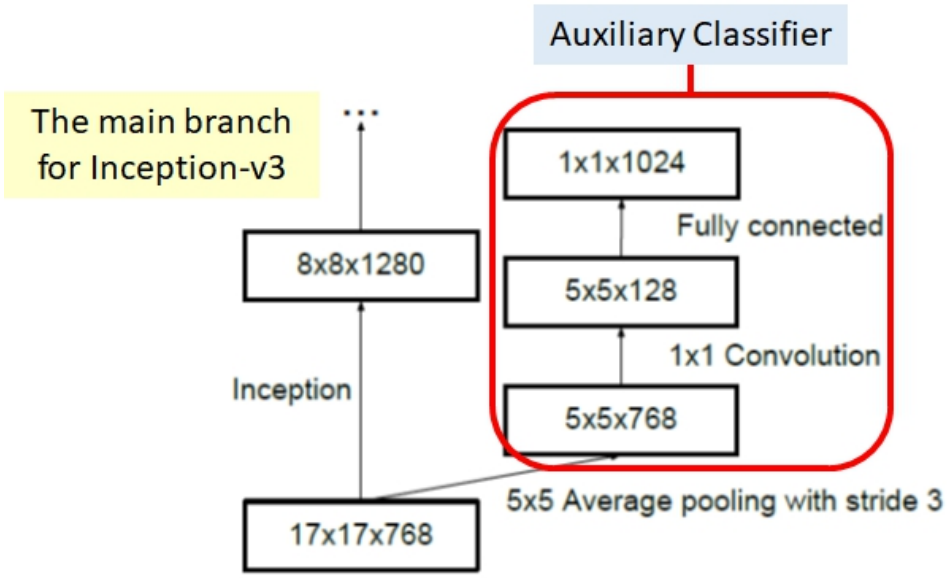
Fig 1.3.3.9: Inception Module c

With factorization, **number of parameters is reduced** for the whole network, it is **less likely to be overfitting**, and consequently, the **network can go wider and deeper also.**

Auxiliary Classifier

Auxiliary Classifiers were already suggested in GoogLeNet / Inception-v1. There are some modifications in Inception-v3.

Only 1 auxiliary classifier is used, instead of using 2 auxiliary classifiers.



1.3.3.10: Auxiliary Classifier act as a regularization

The purpose is also different. In GoogLeNet / Inception-v1, auxiliary classifiers are used for having deeper network. In Inception-v3, auxiliary classifier is used as **regularizer**. So, actually, in deep learning, the modules are still quite intuitive.

Batch normalization, suggested in Inception-v2, is also used in the auxiliary classifier.

Efficient Grid Size Reduction

Conventionally, such as AlexNet and VGGNet, the feature map downsizing is done by max pooling. But the drawback is either too greedy by max pooling followed by conv layer, or too expensive by conv layer followed by max pooling. Here, an efficient grid size reduction is proposed as follows:

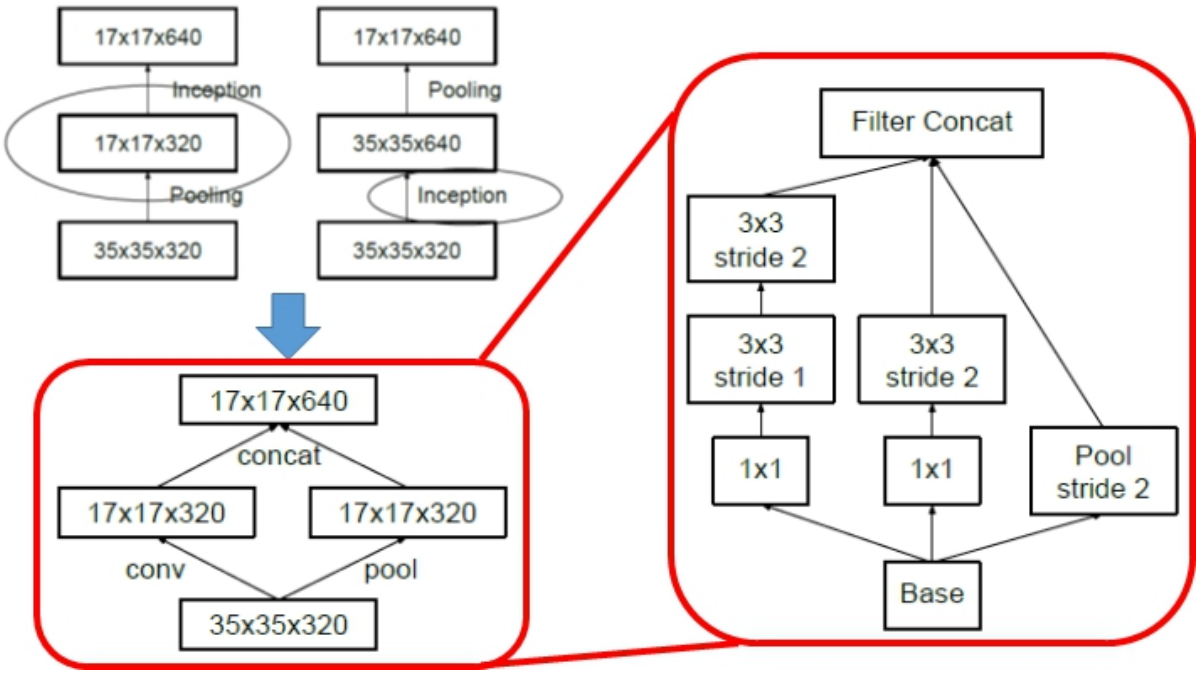


Fig 1.3.3.11: Architecture of Efficient Grid Size Reduction

With the efficient grid size reduction, 320 feature maps are done by conv with stride 2. 320 feature maps are obtained by max pooling. And these 2 sets of feature maps are concatenated as 640 feature maps and go to the next level of inception module.

Less expensive and still efficient network is achieved by this efficient grid size reduction.

Inception-v3 Architecture[5]

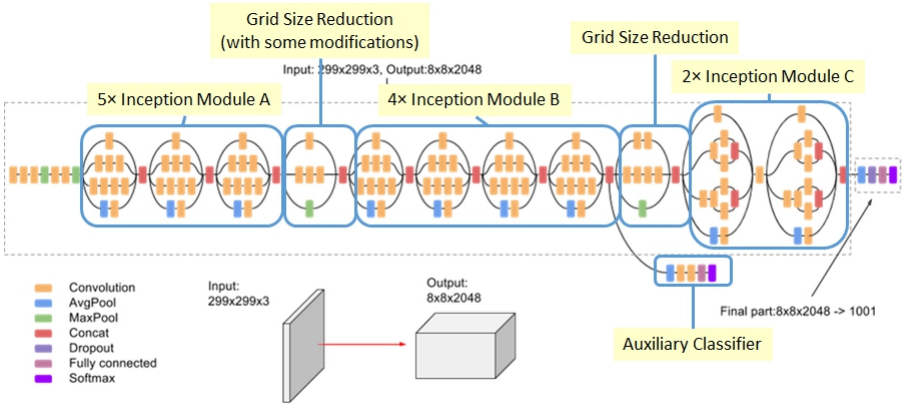


Fig 1.3.3.12: Inception V3

With 42 layers deep, Computational cost is very less and much more efficient.

1.4 Organization of Project

The technique which is developed is taking input as video. As input given is video, system divides the video into number of frames and processes each frame from the video sequence. Then apply Inceptionv3 algorithm on the frame captured and then alphabet representing the particular gesture is displayed on screen along with confidence of the object detected as result.

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements:

- **Software requirements** is a field within software engineering that deals with component to satisfy a contract, standard, specification, or other formally imposed document.
- A documented representation of a condition or capability as in 1 or 2. The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Programming Language : Python 3.6
- Packages : Opencv, numpy, Tensorflow, Keras, Matplotlib
- Tool : Google Colab, anaconda prompt

2.1.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- Operating System: Windows 10
- Processor : Intel Core i5
- CPU Speed : 2.30 GHz
- RAM : 8 GB

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools

have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Numpy

Numpy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object.
- Sophisticated (broadcasting) functions.
- Tools for integrating C/C++ and Fortran code.
- Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, Numpy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases

Tensorflow

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

Currently, the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, c++ or java.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

The core data structures of Keras are **layers** and **models**. The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Google Colab

If you have used Jupyter notebook previously, you would quickly learn to use Google Colab. To be precise, Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

As a programmer, we can

Write and execute code in Python
Document your code that supports mathematical equations
Create/Upload/Share notebooks
Import/Save notebooks from/to Google Drive
Import/Publish notebooks from GitHub
Import external datasets e.g. from Kaggle
Integrate PyTorch, TensorFlow, Keras, OpenCV
Free Cloud service with free GPU

Open CV

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python

API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

3. DESIGN

3.1 Introduction

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.Software design is the first step in SDLC(Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

Software design yields three levels of results:

- **Architectural Design:** It is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of the proposed solution domain.
- **High-Level Design:** It breaks the ‘single entity- multiple component ‘ concept of architectural design into a less-abstracted view of subsystems and modules and depicts their interaction with each other.
- **Detailed Design:** It deals with the implementation part of what is seen as a system and its subsystems in the previous two designs. It is more detailed towards modules and their implementations.

3.2 Architecture Diagram

The main purpose of architectural diagrams is to facilitate collaboration, to increase communication, and to provide vision and guidance.

Two ways architectural diagrams can help:

They help with comprehension: Architectural diagrams show systems. displaying information visually allows the viewer to see everything at a glance, including how things interact. This is especially useful when making changes.

They improve communication and collaboration: One of the main issues software engineers face is consistency. When you’re working on anything that involves multiple people, there's always a risk of miscommunication and discrepancies

between project teams and developers. It is crucial to standardize information which is where an architectural diagram comes in handy.

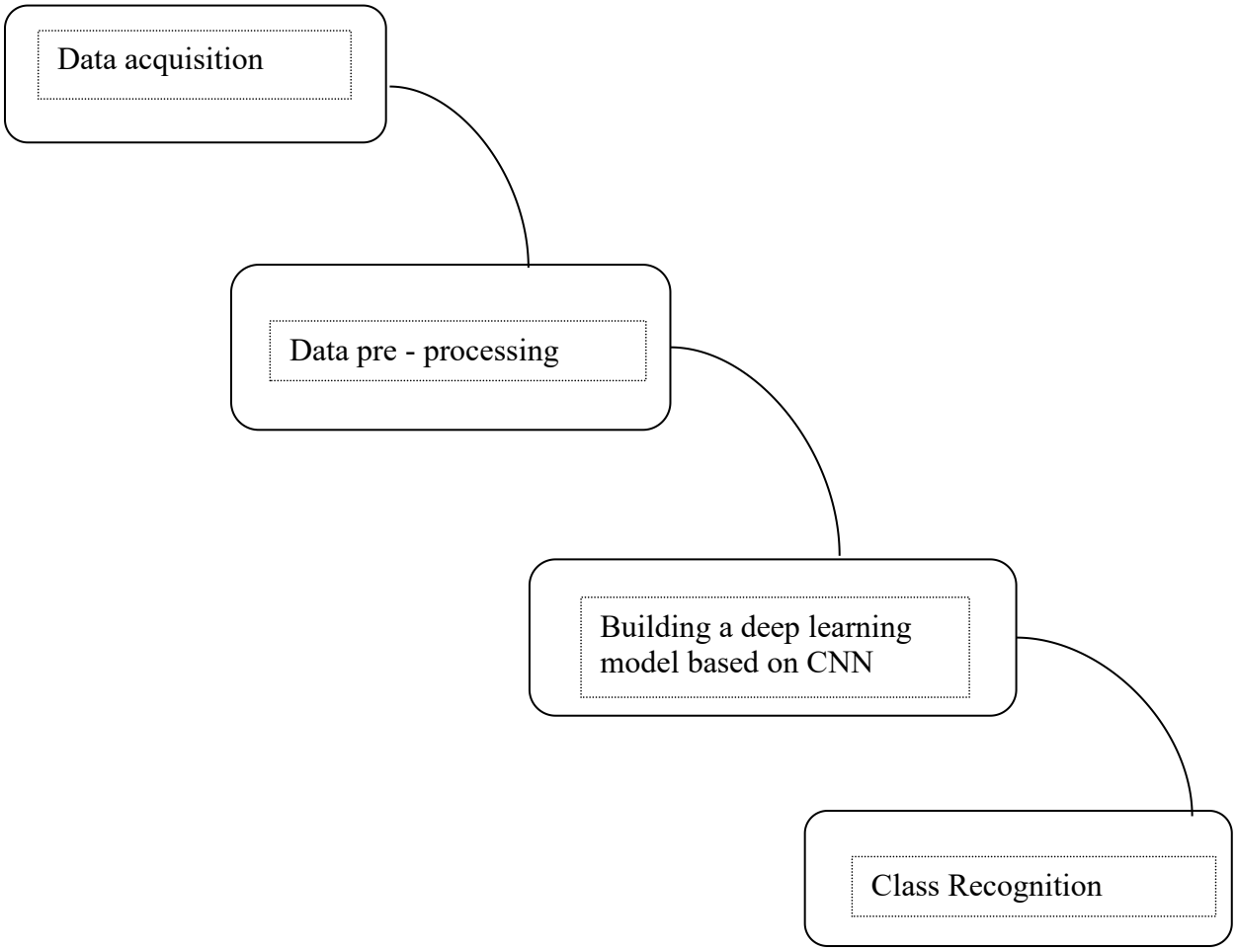


Fig 3.2: Architecture Diagram

3.3 UML Diagrams

3.3.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to

discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

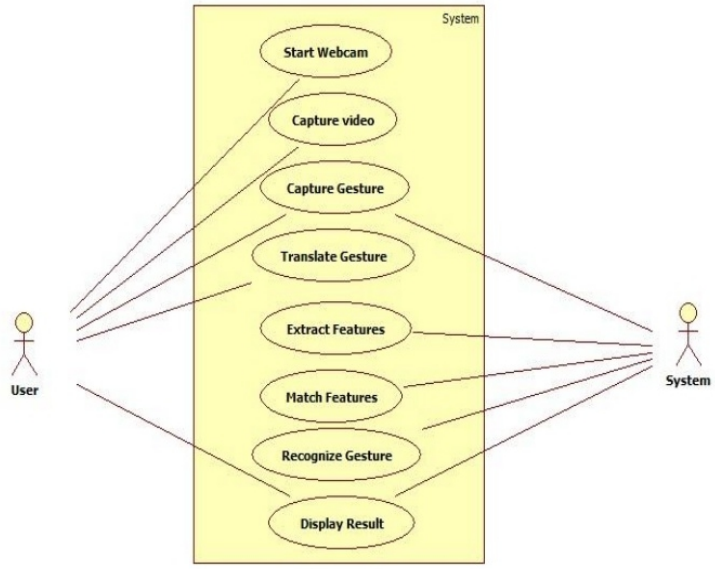


Fig 3.3.1: Use Case Diagram

3.3.2 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the

subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

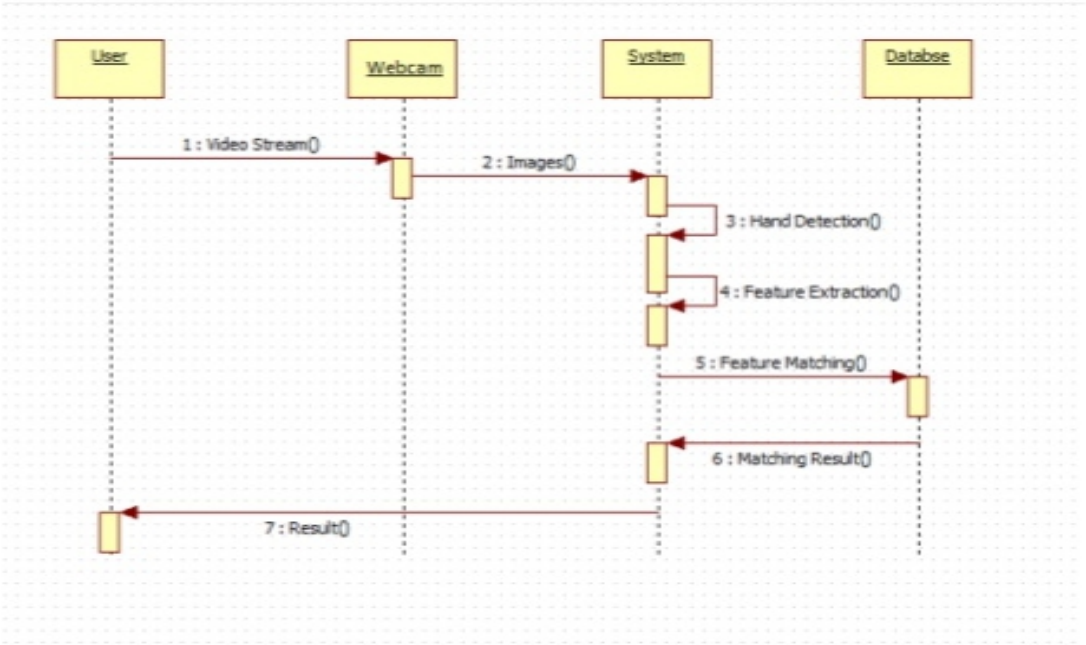


Fig 3.3.2: Sequence Diagram

3.3.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. It is also called object oriented flowchart.

It helps in envisioning the workflow from one activity to another. It puts emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent , and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

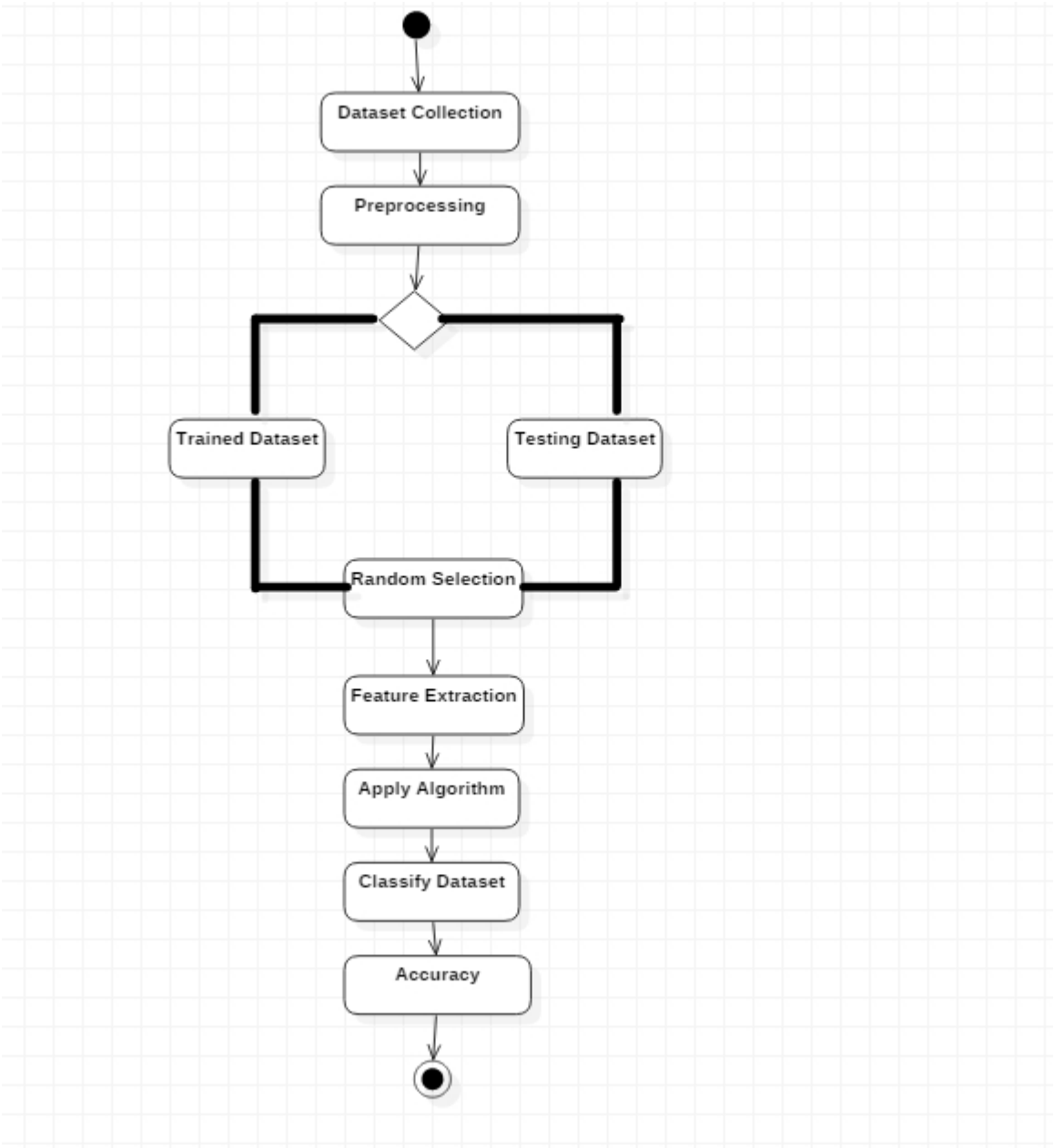


Fig3.3.3 Activity Diagram

3.3.4 Collaboration Diagram

A collaboration diagram also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the unified modelling language. These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object. It depicts the relationships and interactions among software objects.

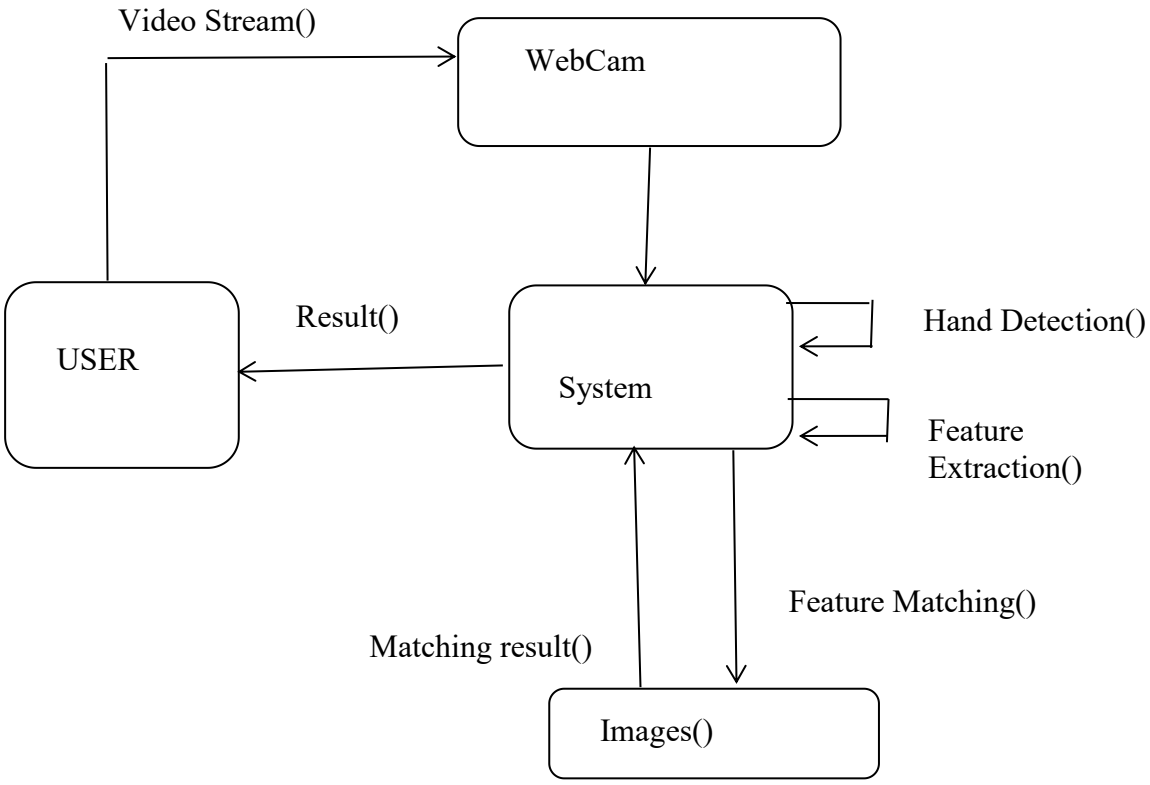


Fig3.3.4 Collaboration Diagram

3.3.5 Class Diagram

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's:

- Classes,
- their attributes,
- operations (or methods),
- and the relationships among objects.

Fig3.3.5 Class Diagram

4. IMPLEMENTATION

4.1 Coding

Importing tensorflow and checking tensorflow:

import tensorflow as tf

from google.colab import files

files.upload()

"""

Setting up the kaggle.json authentication file enabling me to download the dataset:"""

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

"""## **Downloading the grassknoted/asl-alphabet available**

Downloading the dataset using the API:

"""

!kaggle datasets download -d grassknoted/asl-alphabet

"""**Extracting the contents:**

```
"""

!unzip asl-alphabet.zip

"""## **Looking at the dataset**

**Specifying train and test directories:**
"""

# Specifying the training and test directories

TRAINING_DIR = './asl_alphabet_train/asl_alphabet_train/'
TEST_DIR = './asl_alphabet_test/asl_alphabet_test/'

"""**Looking at some random images from the dataset:**


"""

# Commented out IPython magic to ensure Python compatibility.

# Printing 5 random images from any training category or from a specified category
# %matplotlib inline

import cv2
import os
import random
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

number_of_rows = 1
number_of_columns = 5
```

```
categories = os.listdir(TRAINING_DIR)

random.seed(13)

category = categories[random.randint(1, 30)]
# category = 'A'

for i in range(number_of_columns):
  subplot = plt.subplot(number_of_rows, number_of_columns, i + 1)
  subplot.axis('Off')
  subplot.set_title(category)
  image_path = os.path.join(
      TRAINING_DIR,
      str(category),
      str(category) + str(random.randint(1, 1000)) + '.jpg'
  )
  image = mpimg.imread(image_path)
  plt.imshow(image)

plt.show()

"""## **Preparing the training set**

**Augmenting the data with brightness and zoom ranges:**
"""

# Preparing ImageDataGenerator object for training the model
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMAGE_SIZE = 200
BATCH_SIZE = 64

data_generator = ImageDataGenerator(
```

```
    samplewise_center=True,
    samplewise_std_normalization=True,
    brightness_range=[0.8, 1.0],
    zoom_range=[1.0, 1.2],
    validation_split=0.1
)

train_generator = data_generator.flow_from_directory(TRAINING_DIR,
target_size=(IMAGE_SIZE, IMAGE_SIZE), shuffle=True, seed=13,
                                        class_mode='categorical', batch_size=BATCH_SIZE,
subset="training")

validation_generator = data_generator.flow_from_directory(TRAINING_DIR,
target_size=(IMAGE_SIZE, IMAGE_SIZE), shuffle=True, seed=13,
                                        class_mode='categorical', batch_size=BATCH_SIZE,
subset="validation")

"""## **Preparing the model for training**

**Downloading custom weight file if required:**
"""

!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-
datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5 \
    -O /content/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

"""**Preparing Inception V3 Network for transfer learning:**


"""

# Loading inception v3 network for transfer learning
from tensorflow.keras import layers
```

```
from tensorflow.keras import Model

from tensorflow.keras.applications.inception_v3 import InceptionV3

WEIGHTS_FILE = './inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

inception_v3_model = InceptionV3(
    input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3),
    include_top = False,
    weights = 'imagenet'
)

# Not required --> inception_v3_model.load_weights(WEIGHTS_FILE)

# Enabling the top 2 inception blocks to train
for layer in inception_v3_model.layers[:249]:
    layer.trainable = False
for layer in inception_v3_model.layers[249:]:
    layer.trainable = True

# Checking model summary to pick a layer (if required)
inception_v3_model.summary()

"""**Choosing the inception output layer:**



"""

# Choosing the output layer to be merged with our FC layers (if required)
inception_output_layer = inception_v3_model.get_layer('mixed7')
print('Inception model output shape:', inception_output_layer.output_shape)

# Not required --> inception_output = inception_output_layer.output
```

```
inception_output = inception_v3_model.output

"""
**Adding our own set of fully connected layers at the end of Inception v3 network:**

"""

from tensorflow.keras.optimizers import RMSprop, Adam, SGD

x = layers.GlobalAveragePooling2D()(inception_output)
x = layers.Dense(1024, activation='relu')(x)
# Not required --> x = layers.Dropout(0.2)(x)
x = layers.Dense(29, activation='softmax')(x)

model = Model(inception_v3_model.input, x)

model.compile(
    optimizer=SGD(lr=0.0001, momentum=0.9),
    loss='categorical_crossentropy',
    metrics=['acc']
)

"""**Looking at the final model:**


"""

# Watch the new model summary
model.summary()

"""
**Setting up a callback funtion in order to stop training at a particular threshold:**
```

```
"""

# Creating a callback to stop model training after reaching a threshold accuracy

LOSS_THRESHOLD = 0.2
ACCURACY_THRESHOLD = 0.95

class ModelCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if logs.get('val_loss') <= LOSS_THRESHOLD and logs.get('val_acc') >=
ACCURACY_THRESHOLD:
      print("\nReached", ACCURACY_THRESHOLD * 100, "accuracy, Stopping!")
      self.model.stop_training = True

callback = ModelCallback()

"""## **Training the model generated using Inception v3 and our own set of Fully
Connected layers**

```
# This is formatted as code
```

Fitting the model to the training dataset:
"""

history = model.fit_generator(
    train_generator,
    validation_data=validation_generator,
    steps_per_epoch=100,
    validation_steps=50,
    epochs=50,
    callbacks=[callback]
)
```

Fitting the model to the training dataset:

```
[ ] history = model.fit_generator(
      train_generator,
      validation_data=validation_generator,
      steps_per_epoch=100,
      validation_steps=50,
      epochs=50,
      callbacks=[callback]
    )
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pleas
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/50
100/100 [==============================] - 142s 1s/step - loss: 3.4557 - acc: 0.0486 - val_loss: 3.3697 - val_acc: 0.0566
Epoch 2/50
100/100 [==============================] - 102s 1s/step - loss: 3.2251 - acc: 0.1208 - val_loss: 3.2160 - val_acc: 0.1047
Epoch 3/50
100/100 [==============================] - 103s 1s/step - loss: 3.0282 - acc: 0.2352 - val_loss: 3.1186 - val_acc: 0.1741
Epoch 4/50
100/100 [==============================] - 103s 1s/step - loss: 2.8590 - acc: 0.3336 - val_loss: 3.0217 - val_acc: 0.2428
Epoch 5/50
100/100 [==============================] - 103s 1s/step - loss: 2.6663 - acc: 0.4462 - val_loss: 2.9263 - val_acc: 0.2978
Epoch 6/50
100/100 [==============================] - 102s 1s/step - loss: 2.5074 - acc: 0.5153 - val_loss: 2.8264 - val_acc: 0.3400
Epoch 7/50
100/100 [==============================] - 102s 1s/step - loss: 2.3542 - acc: 0.5717 - val_loss: 2.7164 - val_acc: 0.3953
Epoch 8/50
100/100 [==============================] - 102s 1s/step - loss: 2.1829 - acc: 0.6074 - val_loss: 2.6422 - val_acc: 0.4159
Epoch 9/50
```

Fig.4.1.1. Fitting of the model

"""## **Plotting the results**

Training Accuracy vs Validation Accuracy:

"""

```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()


plt.show()
```

```
"""**Training Loss vs Validation Loss:**

"""

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure()

"""## **Saving the model**

**As we were satisfied with our results we save our model:**
"""

# Saving the model
MODEL_NAME = 'models/asl_alphabet_{}.h5'.format(9575)
model.save(MODEL_NAME)

"""## **Testing our model**

**Plotting images along with their respective actual and predicted classes:**
"""

import cv2
import numpy as np
import os
import matplotlib.pyplot as plt

classes = os.listdir(TRAINING_DIR)
classes.sort()
```

```
for i, test_image in enumerate(os.listdir(TEST_DIR)):
    image_location = TEST_DIR + test_image
    img = cv2.imread(image_location)
    img = cv2.resize(img, (IMAGE_SIZE, IMAGE_SIZE))
    plt.figure()
    plt.axis('Off')
    plt.imshow(img)
    img = np.array(img) / 255.
    img = img.reshape((1, IMAGE_SIZE, IMAGE_SIZE, 3))
    img = data_generator.standardize(img)
    prediction = np.array(model.predict(img))
    actual = test_image.split('_')[0]
    predicted = classes[prediction.argmax()]
    print('Actual class: {} \n Predicted class: {}'.format(actual, predicted))
    plt.show()

"""**Calculating test accuracy:**"""

test_images = os.listdir(TEST_DIR)
total_test_cases = len(test_images)
total_correctly_classified = 0
total_misclassified = 0
for i, test_image in enumerate(test_images):
    image_location = TEST_DIR + test_image
    img = cv2.imread(image_location)
    img = cv2.resize(img, (IMAGE_SIZE, IMAGE_SIZE))
    img = np.array(img) / 255.
    img = img.reshape((1, IMAGE_SIZE, IMAGE_SIZE, 3))
    img = data_generator.standardize(img)
    prediction = np.array(model.predict(img))
    actual = test_image.split('_')[0]
    predicted = classes[prediction.argmax()]
    print('Actual class: {} - Predicted class: {}'.format(
        actual, predicted), end=' ')
```

```
    if actual == predicted:
      print('PASS!')
      total_correctly_classified += 1
    else:
      print('FAIL!')
      total_misclassified += 1
print("=" * 20)
test_accuracy = (total_correctly_classified / total_test_cases) * 100
test_error_rate = (total_misclassified / total_test_cases) * 100

print('Test accuracy (%):', test_accuracy)
print('Test error rate (%):', test_error_rate)
print('Number of misclassified classes:', total_misclassified)
print('Number of correctly classified classes', total_correctly_classified)
```

Program_1.py

```
import cv2
import numpy as np

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Prepare data generator for standardizing frames before sending them into the model.
data_generator = ImageDataGenerator(samplewise_center=True,
samplewise_std_normalization=True)

# Loading the model.
MODEL_NAME = "C:/Users/Neelu/Downloads/asl_alphabet_9575.h5"
model = load_model(MODEL_NAME)

# Setting up the input image size and frame crop size.
IMAGE_SIZE = 200
CROP_SIZE = 400
```

```
# Creating list of available classes stored in classes.txt.
classes_file = open("classes.txt")
classes_string = classes_file.readline()
classes = classes_string.split()
classes.sort()  # The predict function sends out output in sorted order.

# Preparing cv2 for webcam feed
cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame.
    ret, frame = cap.read()

    # Target area where the hand gestures should be.
    cv2.rectangle(frame, (0, 0), (CROP_SIZE, CROP_SIZE), (0, 255, 0), 3)

    # Preprocessing the frame before input to the model.
    cropped_image = frame[0:CROP_SIZE, 0:CROP_SIZE]
    resized_frame = cv2.resize(cropped_image, (IMAGE_SIZE, IMAGE_SIZE))
    reshaped_frame = (np.array(resized_frame)).reshape((1, IMAGE_SIZE,
IMAGE_SIZE, 3))
    frame_for_model = data_generator.standardize(np.float64(reshaped_frame))

    # Predicting the frame.
    prediction = np.array(model.predict(frame_for_model))
    predicted_class = classes[prediction.argmax()]      # Selecting the max confidence
index.

    # Preparing output based on the model's confidence.
    prediction_probability = prediction[0, prediction.argmax()]
    if prediction_probability > 0.5:
        # High confidence.
        cv2.putText(frame, '{} - {:.2f}%'.format(predicted_class, prediction_probability
* 100),
```

```
                        (10, 450), 1, 2, (255, 255, 0), 2, cv2.LINE_AA)
    elif prediction_probability > 0.2 and prediction_probability <= 0.5:
        # Low confidence.
        cv2.putText(frame, 'Maybe {}... - {:.2f}%'.format(predicted_class,
prediction_probability * 100),
                        (10, 450), 1, 2, (0, 255, 255), 2, cv2.LINE_AA)
    else:
        # No confidence.
        cv2.putText(frame, classes[-2], (10, 450), 1, 2, (255, 255, 0), 2, cv2.LINE_AA)

    # Display the image with prediction.
    cv2.imshow('frame', frame)

    # Press q to quit
    k = cv2.waitKey(1) & 0xFF
    if k == ord('q'):
        break

# When everything done, release the capture.
cap.release()
cv2.destroyAllWindows()
```

4.2 Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There of basically two types of testing approaches.

One is Black-Box testing – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is White-Box testing – knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

4.2.1 Testing Strategies

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths through out the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an end less loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

REGRESSION TESTING

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

4.3 TEST CASES

Integrated and regression testing strategies are used in this application for testing.

TestCase ID	Test Scenario	Expected Result	Actual Result	Pass/Fail
TC01	Check if webcam is switched on	As soon as program is excuted, webcam switched on	As Expected	Pass
TC02	Check if video sequentlly are divided into frames for processing	To Process the given hand gesture, each frame is captured from video sequences	As Expected	Pass
TC03	Check if bouding box is displayed	As soon as program is excuted, bounding box(Green box) displayed	As Expected	Pass
TC04	Check if hand gesture is deteceted	As soon as hand gesture will be detected	As Expected	Pass
TC05	Check if alphabet along with its confidence is dipalyed or not	After prediction happens, alphabet along with its confidence for that respective gesture is dispalyed	As Expected	Pass

TC06	Check if nothing is displayed, when nothing is infront of camera	Nothing will be displayed, if no gesture is placed.	As Expected	Pass

Fig 4.3: Test Cases

4.4 DATASET TRAINING SCREENSHOTS

Fitting the model to the training dataset:

```
[ ] history = model.fit_generator(
        train_generator,
        validation_data=validation_generator,
        steps_per_epoch=100,
        validation_steps=50,
        epochs=50,
        callbacks=[callback]
    )
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pleas
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/50
100/100 [==============================] - 142s 1s/step - loss: 3.4557 - acc: 0.0486 - val_loss: 3.3697 - val_acc: 0.0566
Epoch 2/50
100/100 [==============================] - 102s 1s/step - loss: 3.2251 - acc: 0.1208 - val_loss: 3.2160 - val_acc: 0.1047
Epoch 3/50
100/100 [==============================] - 103s 1s/step - loss: 3.0282 - acc: 0.2352 - val_loss: 3.1186 - val_acc: 0.1741
Epoch 4/50
100/100 [==============================] - 103s 1s/step - loss: 2.8590 - acc: 0.3336 - val_loss: 3.0217 - val_acc: 0.2428
Epoch 5/50
100/100 [==============================] - 103s 1s/step - loss: 2.6663 - acc: 0.4462 - val_loss: 2.9263 - val_acc: 0.2978
Epoch 6/50
100/100 [==============================] - 102s 1s/step - loss: 2.5074 - acc: 0.5153 - val_loss: 2.8264 - val_acc: 0.3400
Epoch 7/50
100/100 [==============================] - 102s 1s/step - loss: 2.3542 - acc: 0.5717 - val_loss: 2.7164 - val_acc: 0.3953
Epoch 8/50
100/100 [==============================] - 102s 1s/step - loss: 2.1829 - acc: 0.6074 - val_loss: 2.6422 - val_acc: 0.4159
Epoch 9/50
```

Fig 4.4.1: Train the dataset

```
Epoch 35/50
100/100 [==============================] - 93s 929ms/step - loss: 0.7395 - acc: 0.8000 - val_loss: 0.9480 - val_acc: 0.7553
Epoch 36/50
100/100 [==============================] - 95s 951ms/step - loss: 0.7259 - acc: 0.8055 - val_loss: 0.9250 - val_acc: 0.7659
Epoch 37/50
100/100 [==============================] - 94s 944ms/step - loss: 0.6873 - acc: 0.8075 - val_loss: 0.8559 - val_acc: 0.7766
Epoch 38/50
100/100 [==============================] - 93s 934ms/step - loss: 0.6749 - acc: 0.8192 - val_loss: 0.8716 - val_acc: 0.7688
Epoch 39/50
100/100 [==============================] - 94s 937ms/step - loss: 0.6531 - acc: 0.8216 - val_loss: 0.8491 - val_acc: 0.7747
Epoch 40/50
100/100 [==============================] - 93s 931ms/step - loss: 0.6099 - acc: 0.8369 - val_loss: 0.7877 - val_acc: 0.7912
Epoch 41/50
100/100 [==============================] - 93s 927ms/step - loss: 0.5983 - acc: 0.8373 - val_loss: 0.8092 - val_acc: 0.7753
Epoch 42/50
100/100 [==============================] - 93s 933ms/step - loss: 0.5907 - acc: 0.8388 - val_loss: 0.7827 - val_acc: 0.7894
Epoch 43/50
100/100 [==============================] - 93s 935ms/step - loss: 0.5443 - acc: 0.8505 - val_loss: 0.7719 - val_acc: 0.7812
Epoch 44/50
100/100 [==============================] - 93s 936ms/step - loss: 0.5360 - acc: 0.8572 - val_loss: 0.7315 - val_acc: 0.8031
Epoch 45/50
100/100 [==============================] - 93s 935ms/step - loss: 0.5124 - acc: 0.8598 - val_loss: 0.7214 - val_acc: 0.7916
Epoch 46/50
100/100 [==============================] - 95s 954ms/step - loss: 0.4873 - acc: 0.8695 - val_loss: 0.6861 - val_acc: 0.8138
Epoch 47/50
100/100 [==============================] - 94s 940ms/step - loss: 0.4727 - acc: 0.8788 - val_loss: 0.6823 - val_acc: 0.8084
Epoch 48/50
100/100 [==============================] - 94s 943ms/step - loss: 0.4753 - acc: 0.8728 - val_loss: 0.6665 - val_acc: 0.8150
Epoch 49/50
100/100 [==============================] - 93s 933ms/step - loss: 0.4602 - acc: 0.8745 - val_loss: 0.6352 - val_acc: 0.8184
Epoch 50/50
100/100 [==============================] - 93s 932ms/step - loss: 0.4387 - acc: 0.8808 - val_loss: 0.6532 - val_acc: 0.8106
```

Fig 4.4.2: Training dataset completed

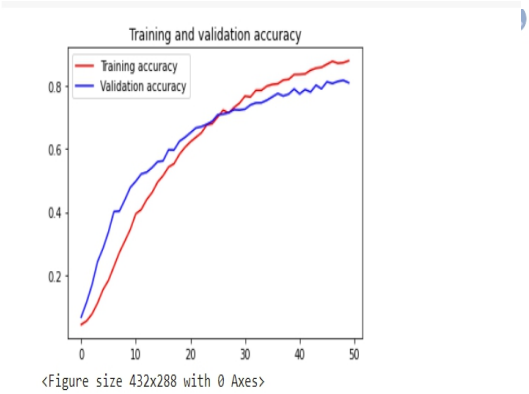


Fig 4.4.3: Training and validation accuracy

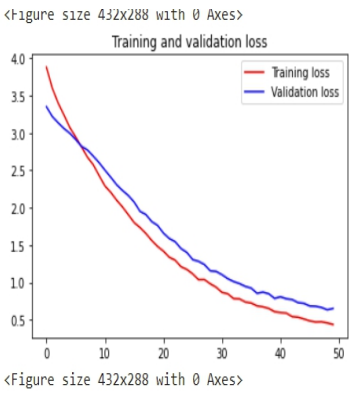


Fig 4.4.: Training and validation accuracy

4.5 INPUT SCREENSHOTS

Fig 4.5.1: Command to run project

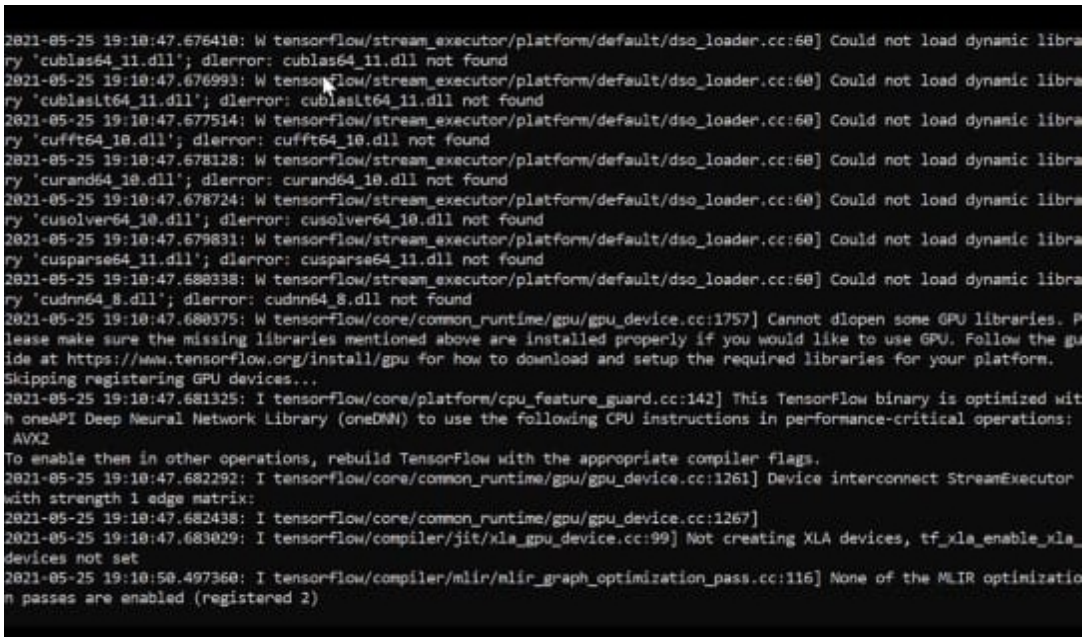


Fig 4.5.2: Executing

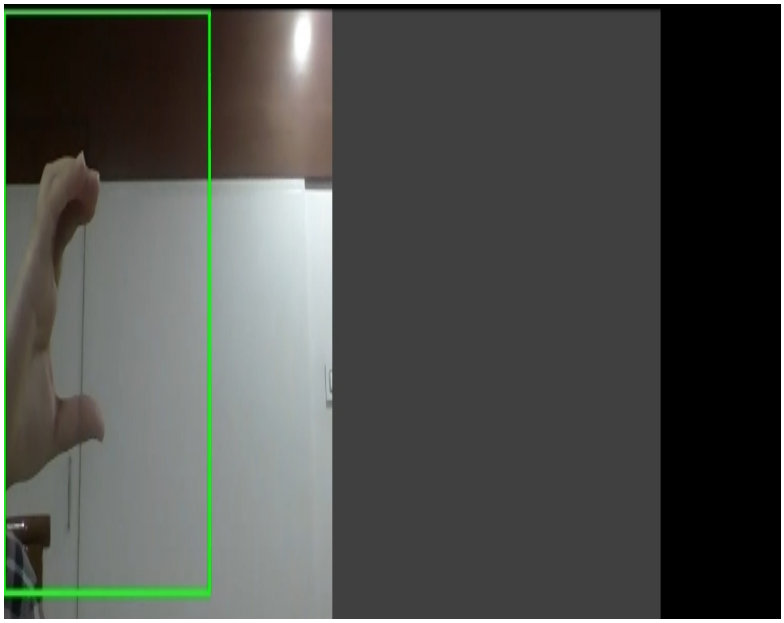


Fig 4.5.3 Live_Input-1

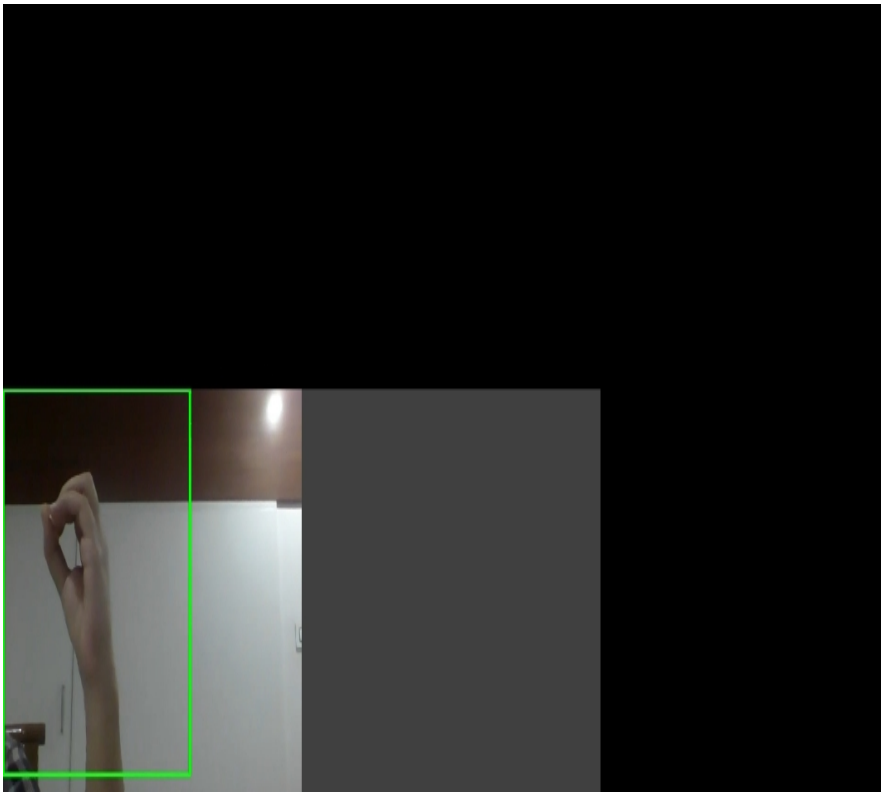


Fig 4.5.4 Live_Input-2

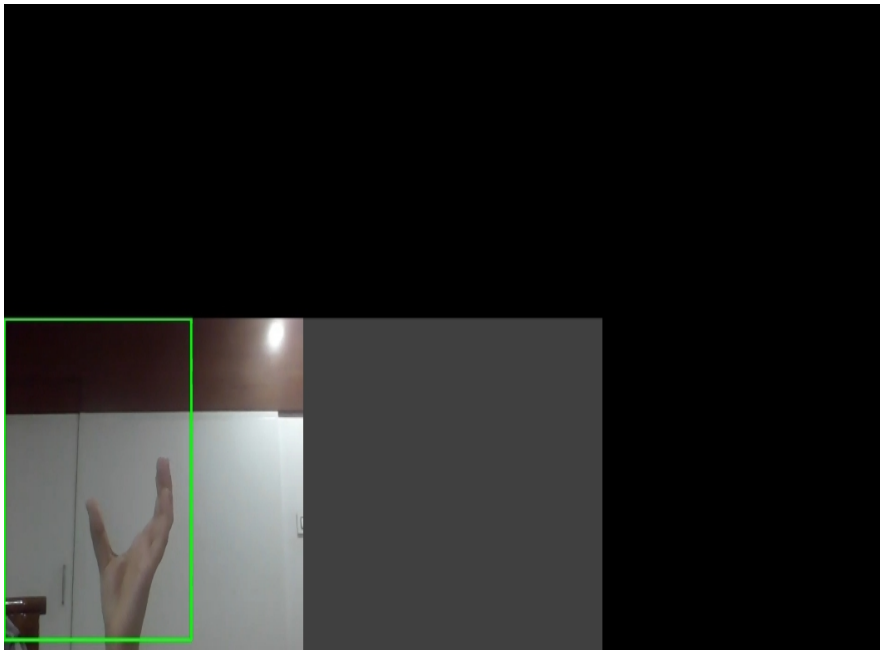


Fig 4.5.5 Live_Input-3

4.6 OUTPUT SCREENSHOTS

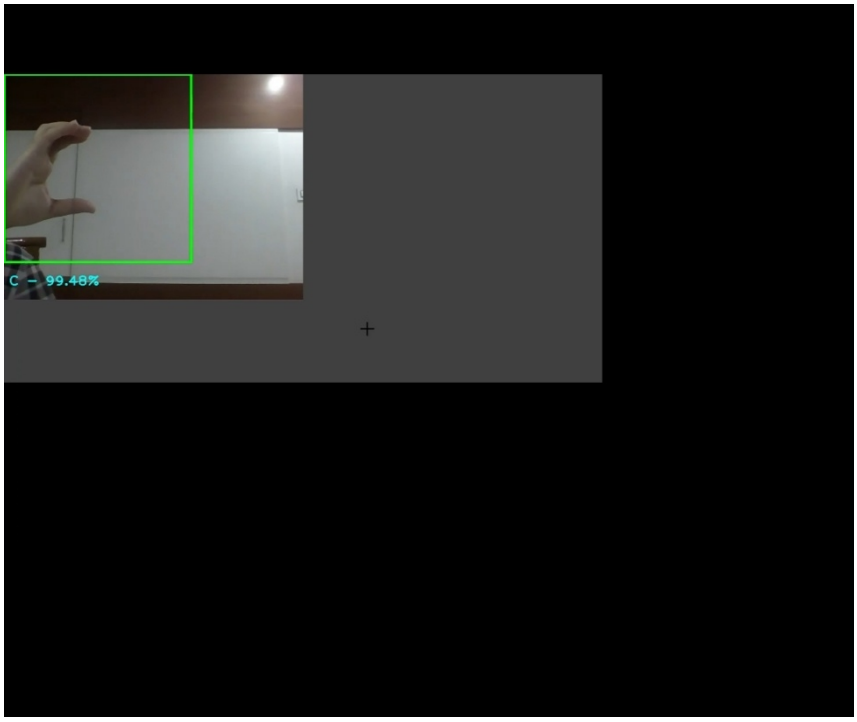


Fig 4.6.1: Live_Output-1

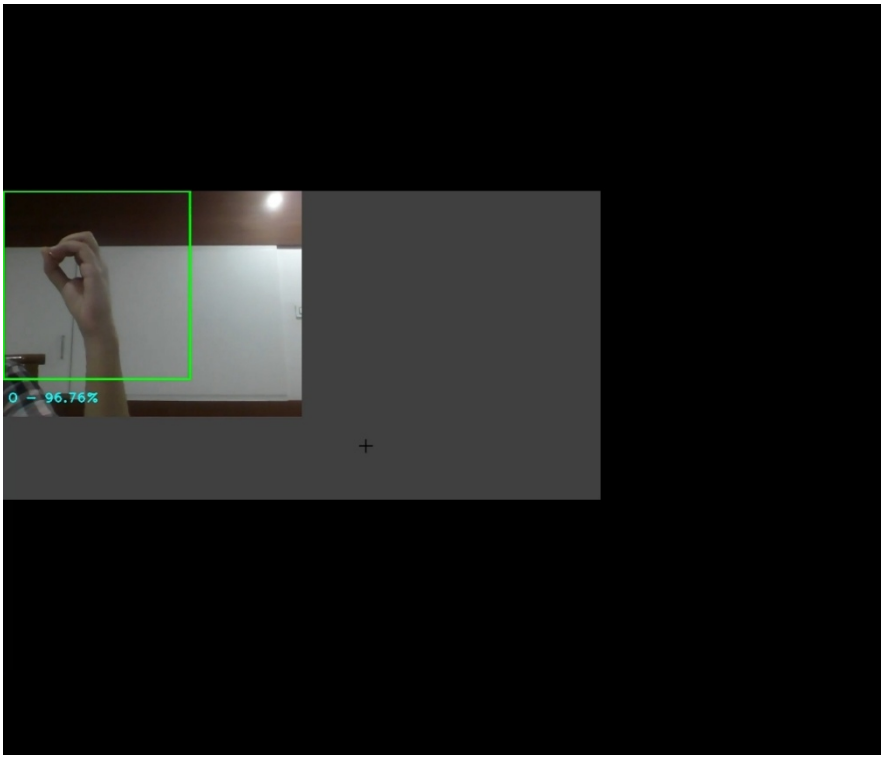


Fig 4.6.2: Live_Output-2

Fig 4.6.3: Live_Output-3

5.CONCLUSION AND FUTURE SCOPE

In this project, we have used anaconda prompt to run the project. During the execution, first webcam is switched on and the video capturing box is displayed . Area is specified to capture the hand gestures(Green box). Video sequence is split into number of frames. The data set we have considered for training is around 87,000 images . Then InceptionV3 algorithm is applied on each frame to detect the hand gesture. Then the alphabet that belongs to specific hand gesture is displayed along with confidence of the detected hand gesture. By confidence we can tell, whether we got accurate result or not. If confidence is greater than 0.20 and less than 0.50. Then displays the letter as may be. If confidence is greater than 0.50 it displays the exact letter. If confidence is less than 0.20 then nothing is printed .

Performance of the system reaches 88.08% accuracy. This project will help establishment of communication between speech impaired and normal people. Additionally, we can extend the program by includes numbers, some standard signs like how are you, reading etc… and also convert text into speech.

6. REFERENCES

[1]"Base Paper" https://www.ijcrt.org/papers/IJCRT2103503.pdf

[2]"Convolutional_Neural_Network"https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4

[3] "DataSet" https://www.kaggle.com/grassknoted/asl-alphabet

[4] https://www.geeksforgeeks.org/inception-v2-and-v3-inception-network-versions/

[5] https://arxiv.org/ftp/arxiv/papers/1805/1805.06618.pdf

[6]https://arxiv.org/pdf/1512.00567v3.pdf

[7]https://sh-tsang.medium.com/review-batch-normalization-inception-v2-bn-inception-the-2nd-to-surpass-human-level-18e2d0f56651

[8]https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/