VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA" , BELAGAVI-590018, KARNATAKA

Project Report
On

"Sign Language Recognition"

Submitted in the partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING

Submitted By

Satwik Ram K	**1VA17CS047**
N Pavan Kumar	**1VA17CS028**

Under the Guidance of
SUNIL G L

Assistant Professor,

Dept. of CS&E, SVIT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)

RAJANUKUNTE, BENGALURU – 560 064

2020-21

SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)

Rajanukunte, Bengaluru- 560 064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Project work entitled *"**Sign Language Recognition**"* carried out by **Mr. Satwik Ram K (1VA17CS047), Mr. N Pavan Kumar (1VA17CS028),** a bonafide students of **SAI VIDYA INSTITUTE OF TECHNOLOGY**, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belagavi during the year **2020-21.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

--------------------	--------------------------	----------------------
Prof. Sunil G L	**Prof. Sreelatha P K**	**Dr. H S Ramesh Babu**
Assistant Professor,	HOD,	Principal,
Dept. of CS&E, SVIT	Dept. of CS&E, SVIT	SVIT

External Viva:

	Name	**Signature**
1.	______________________________	______________________________
2.	______________________________	______________________________

Table of Contents

ACKNOWLEDGEMENT

The completion of project brings with and sense of satisfaction, but it is never completed without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **Sai Vidya Institute of Technology**, for providing mean opportunity to do our education.

I would like to thank the **Management** and **Prof. M R Holla,** Director, Sai Vidya Institute of Technology for providing the facilities.

I extend my deep sense of sincere gratitude to **Dr. H S Ramesh Babu**, Principal, Sai Vidya Institute of Technology, Bengaluru, for having permitted to carry out the project work on **"Sign Language Recognition"** successfully**.**

I am thankful to **Prof. A M Padma Reddy**, Additional Director and Professor, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, for his constant support and motivation.

I express my heartfelt sincere gratitude to **Prof. Sreelatha P K,** HOD, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for his valuable suggestions and support.

I express my sincere gratitude to **Prof. Sunil G L**, Assistant Professor, Project Guide, Department of CSE, Sai Vidya Institute of Technology, Bengaluru, for his constant support in completing the project.

Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their support.

Satwik Ram K	1VA17CS047
N Pavan Kumar	1VA17CS028

ABSTRACT

Sign Language is mainly used by deaf (hard hearing) and dumb people to exchange information between their own community and with other people. It is a language where people use their hand gestures to communicate as they can't speak or hear. Sign Language Recognition (SLR) deals with recognizing the hand gestures acquisition and continues till text or speech is generated for corresponding hand gestures. Here hand gestures for sign language can be classified as static and dynamic. However, static hand gesture recognition is simpler than dynamic hand gesture recognition, but both recognition is important to the human community. We can use Deep Learning Computer Vision to recognize the hand gestures by building Deep Neural Network architectures (Convolution Neural Network Architectures) where the model will learn to recognize the hand gestures images over an epoch. Once the model Successfully recognizes the gesture the corresponding English text is generated and then text can be converted to speech. This model will be more efficient and hence communicate for the deaf (hard hearing) and dump people will be easier. In this paper, we will discuss how Sign Language Recognition is done using Deep Learning.

CHAPTER 1

INTRODUCTION

1.1 Overview

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal.

A subset of artificial intelligence is machine learning, which refers to the concept that computer programs can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images, or video.

Artificial intelligence is based on the principle that human intelligence can be defined in a way that a machine can easily mimic it and execute tasks, from the most simple to those that are even more complex. The goals of artificial intelligence include learning, reasoning, and perception. As technology advances, previous benchmarks that defined artificial intelligence become outdated. For example, machines that calculate basic functions or recognize text through optical character recognition are no longer considered to embody artificial intelligence, since this function is now taken for granted as an inherent computer function.

AI is continuously evolving to benefit many different industries. Machines are wired using a cross-disciplinary approach based on mathematics, computer science, linguistics, psychology, and more.

1.2 Problem Statement

Sign language uses lots of gestures so that it looks like movement language which consists of a series of hands and arms motion. For different countries, there are different sign languages and hand gestures. Also, it is noted that some unknown words are translated by simply showing gestures for each alphabet in the word. In addition, sign language also includes specific gestures to each alphabet in the English dictionary and for each number between 0 and 9. Based on these sign languages are made up of two groups, namely static gesture, and dynamic gesture. The static gesture is used for alphabet and number representation, whereas the dynamic gesture is used for specific concepts. Dynamic also include words, sentences, etc. The static gesture consists of hand gestures, whereas the latter includes the motion of hands, head, or both. Sign language is a visual language and consists of 3 major components, such as finger-spelling, word-level sign vocabulary, and non-manual features. Finger-spelling is used to spell words letter by letter and convey the message whereas the latter is keyword-based. But the design of a sign language translator is quite challenging despite many research efforts during the last few decades. Also, even the same signs have significantly different appearances for different signers and different viewpoints. This work focuses on the creation of a static sign language translator by using a Convolutional Neural Network. We created a lightweight network that can be used with embedded devices/standalone applications/web applications having fewer resources.

1.3 Motivation

The various advantages of building a Sign Language Recognition system includes:

- ✓ Sign Language hand gestures to text/speech translation system or dialog systems which are used in specific public domains such as airports, post offices.
- ✓ Sign Language Recognition (SLR) can help to translate the video to text or speech enables inter-communication between normal and deaf people.

1.4 Objectives

The main objectives of this project are to contribute to the field of automatic sign language recognition and translation to text or speech. In our project, we focus on static sign language hand gestures. This work focused on recognizing the hand gestures which includes 26 English alphabets (A-Z) and 10 digits (0-9) using Deep Neural Networks (DNN). We created a convolution neural networks classifier that can classify the hand gestures into English alphabets and digits.

We have trained the neural network under different configurations and architectures like LeNet-5, MobileNetV2, and our own architecture. We used the horizontal voting ensemble technique to achieve the maximum accuracy of the model. We have also created a web application using Django Rest Frameworks to test our results from a live camera.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey

2.1.1 Machine Learning

Machine Learning is a subset of Artificial Intelligence that uses statistical learning algorithms to build systems that can automatically learn and improve from experiences without being explicitly programmed. ML algorithms can be broadly classified into three categories:

Supervised Learning

In supervised learning we have input variables (x) and an output variable (Y) and we use an algorithm to learn the mapping from input to output. In other words, a supervised learning algorithm takes a known set of input dataset and its known responses to the data (output) to learn the regression/classification model. A learning algorithm then trains a model to generate a prediction for the response to new data or the test datasets.

Unsupervised Learning

Unsupervised Learning is used when we do not have labelled data. Its main focus is to learn more about the data by inferring patterns in the dataset without reference to the known outputs. It is called unsupervised because the algorithms are left on their own to group the unsorted information by finding similarities, differences and patterns in the data. Unsupervised learning is mostly performed as a part of exploratory data analysis. It is most commonly used to find clusters of data and for dimensionality reduction.

Reinforcement Learning

In simple terms, reinforcement learning can be explained as learning by continuously interacting with the environment. It is a type of machine learning algorithm in which an agent learns from an interactive environment in a trial and error way by continuously using feedback from its previous actions and experiences. The reinforcement learning uses rewards and punishments, the agents receive rewards for performing correct actions and penalties for doing it incorrectly.

2.1.2 Deep Learning

Deep learning is a machine learning technique that is inspired by the way a human brain filters information, it is basically learning from examples. It helps a computer model to filter the input data through layers to predict and classify information. Since deep learning processes information in a similar manner as a human brain does, it is mostly used in applications that people generally do. It is the key technology behind driver-less cars, that enables them to recognize a stop sign and to distinguish between a pedestrian and lamp post. Most of the deep learning methods use neural network architectures, so they are often referred to as deep neural networks. Deep Learning is basically mimicking the human brain, it can also be defined as a multi neural network architecture containing a large number of parameters and layers.

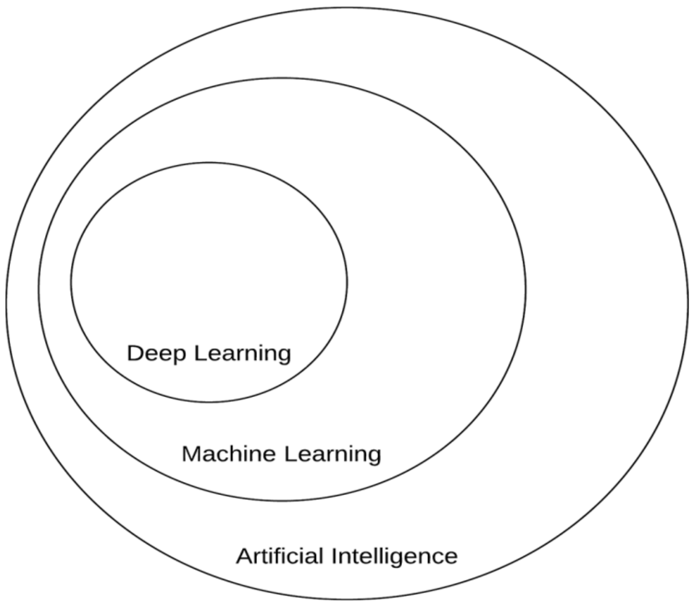


Fig 2.1 Relation between Artificial Intelligence, Machine and Deep Learnin

2.1.3. Computer vision

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do." Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding”.

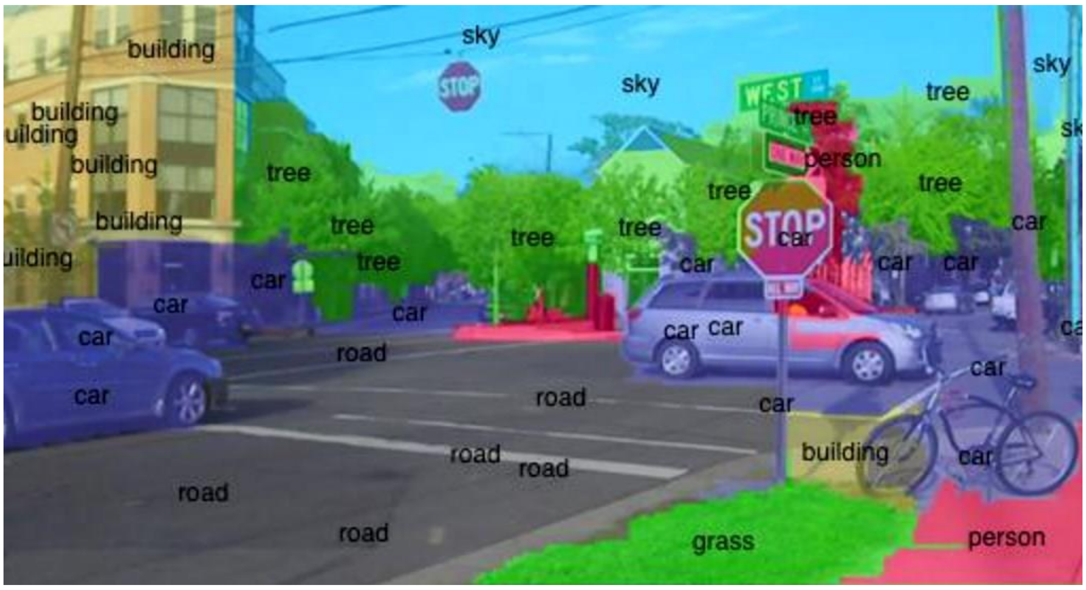


Fig 2.2 Computer Vision – Important manifestation of Artificial Intelligence

As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

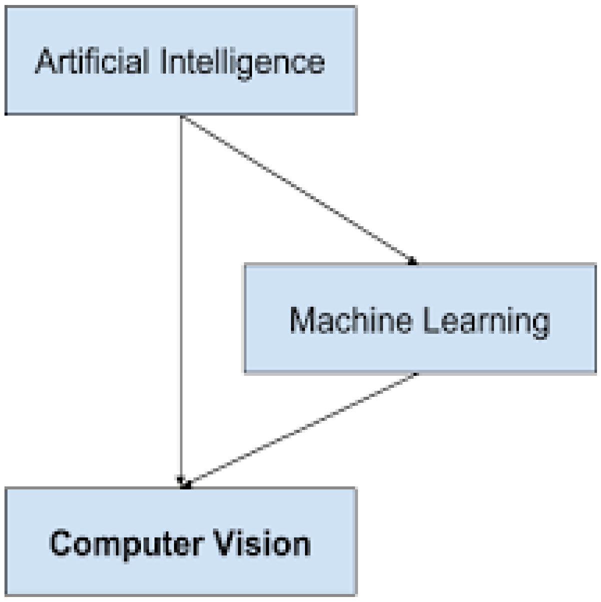


Fig 2.3 Relation between AI, Machine Learning and Computer Vision

2.1.4. Applications

The applications for artificial intelligence are endless. The technology can be applied to many different sectors and industries. AI is being tested and used in the healthcare industry for dosing drugs and different treatment in patients, and for surgical procedures in the operating room. Other examples of machines with artificial intelligence include computers that play chess and self-driving cars. Each of these machines must weigh the consequences of any action they take, as each action will impact the end result. In chess, the end result is winning the game. For self-driving cars, the computer system must account for all external data and compute it to act in a way that prevents a collision.

Artificial intelligence also has applications in the financial industry, where it is used to detect and flag activity in banking and finance such as unusual debit card usage and large account deposits—all of which help a bank's fraud department. Applications for AI are also being used to help streamline and make trading easier. This is done by making supply, demand, and pricing of securities easier to estimate. There are many systems developed in colleges and industries to keep track of attendance. But there are performance and stability problems.

[1] Kang, Byeongkeun, Subarna Tripathi, and Truong Q. Nguyen. ”Real- time sign language fingerspelling recognition using convolutional neural networks from depth map.” arXiv preprint arXiv: 1509.03001 (2015).

This works focuses on static fingerspelling in American Sign Language A method for implementing a sign language to text/voice conversion system without using handheld gloves and sensors, by capturing the gesture continuously and converting them to voice. In this method, only a few images were captured for recognition. The design of a communication aid for the physically challenged.

[2] Suganya, R., and T. Meeradevi. ”Design of a communication aid for physically challenged.” In Electronics and Communication Systems (ICECS), 2015 2nd International Conference on, pp. 818-822. IEEE, 2015.

The system was developed under the MATLAB environment. It consists of mainly two phases via training phase and the testing phase. In the training phase, the author used feed-forward neural networks. The problem here is MATLAB is not that efficient and also integrating the concurrent attributes as a whole is difficult.

[3] Sruthi Upendran, Thamizharasi. A," American Sign Language Interpreter System for Deaf and Dumb Individuals", 2014 International Conference on Control, Instrumentation, Communication and Computation

The discussed procedures could recognize 20 out of 24 static ASL alphabets. The alphabets A, M, N and S couldn't be recognized due to occlusion problem. They have used only a limited number of images.

CHAPTER 3

SYSTEM AND SOFTWARE REQUIREMENTS AND SPECIFICATIONS

The program works on Desktop PC and is executed using a Django Rest Frameworks interface which interacts with a PostgreSQL database running on localhost.

3.1 FUNCTIONAL REQUIREMENTS

A description of the facility or feature required. Functional requirements deal with what the system should do or provide for users. They include description of the required functions, outlines of associated reports or online queries, and details of data to be held in the system.

3.1.1 Interface Requirements:

- ✓ The system shall provide user to upload image via frontend.
- ✓ The system shall give the prediction output in the web page.
- ✓ The system shall provide option to convert the predicted text to speech and all the details will be stored in the database.
- ✓ The system shall provide option to make prediction on both web application and standalone application.
- ✓ The system shall provide option to capture live hand gesture via camera and detect the corresponding English alphabets and then store in the database.

3.2 NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements define the overall qualities or attributes of the resulting system.

3.2.1 Usability

Usability is the ease with which a user can learn to operate the Online Shopping Website system and get results.

3.2.2 Security

Security requirements are included in a system to ensure

- ✓ All inventory and supplier information are well secured
- ✓ SQL injection is prevented

3.2.3 Reliability

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time. Constraints on the run-time behaviour of the system can be considered under two separate headings:

- ✓ Availability: is the system available for service when requested by end-users.
- ✓ Failure rate: how often does the system fail to deliver the service as expected by end- users.

3.2.4. Efficiency

The comparison of what is actually produced or performed with what can be achieved with the same consumption of Clouds (money, time, labor, etc.). It is an important Factor in Determination of Productivity.

3.3 SOFTWARE REQUIREMENTS

Programming language	:	Python
Operating system	:	ANY OS
Application required	:	Web Application
Coding language	:	Core Python OOPS

3.4 HARDWARE REQUIREMENTS

CPU	:	Pentium IV 2.4 GHz
Memory (Primary)	:	512 MB, 1 GB or above
Hard Disk	:	40 GB, 80GB, 160GB or above
Monitor	:	15 VGA color

CHAPTER 4

SYSTEM DESIGN AND METHODOLOGY

4.1 Methodology

Image recognition is done using TensorFlow

Techniques used:

- ✓ Convolution Neural Network
- ✓ OpenCV
- ✓ TensorFlow
- ✓ Django Rest Frameworks

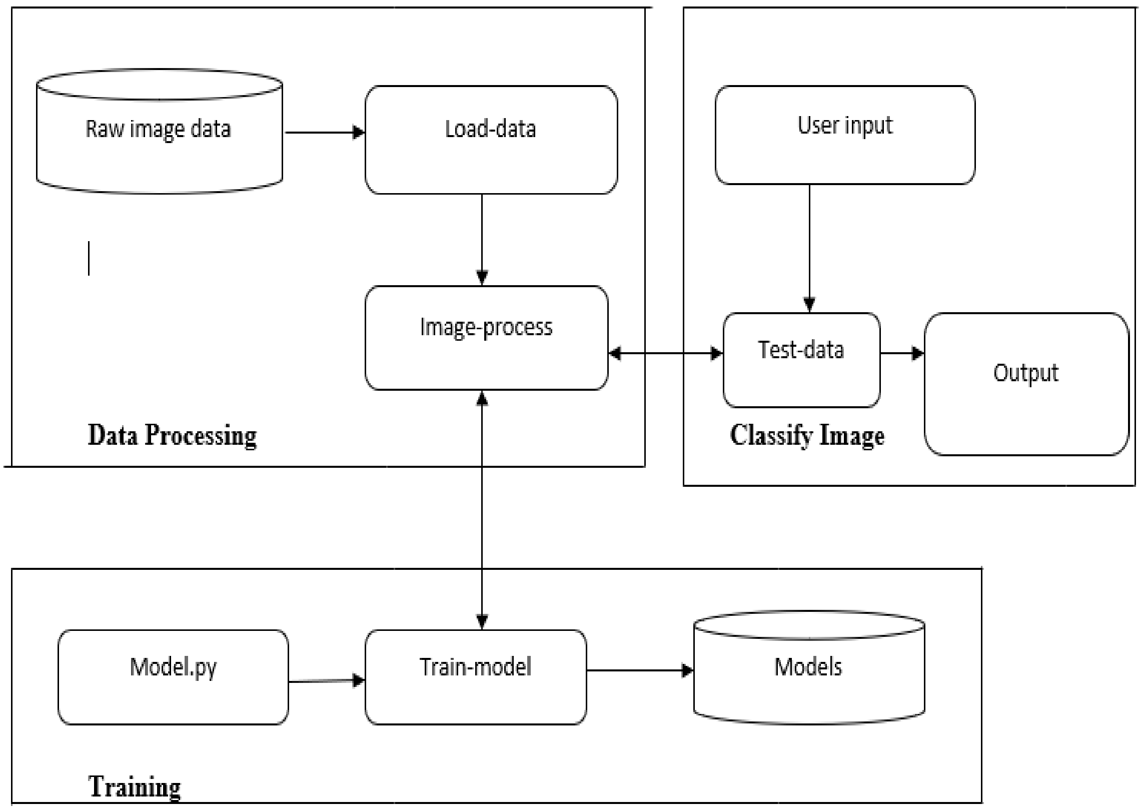


Fig 4.1: Methodology of Proposed Model

The methodology has 3 major steps as shown in the Fig 4.1:

1. Data preprocessing
2. Training
3. Image classification

Data preprocessing is the first step of the process. Here the raw data is obtained, loaded and then it is processed. Training is the second step of the process. Here, the preprocessed images are given to the YOLO model and the model is trained on the images. Once the model is trained, the input image is given through the camera, the input image is processed and the processed image is trained to model. The model classifies the image and the output is obtained. The model is deployed using Django Rest Frameworks.

Steps involved are:

- ✓ Initializing the model using Sequential class.
- ✓ Adding the Input layer that is images and applying the Convolution 2D method.
- ✓ Adding the Hidden layer and applying the activation function.
- ✓ Max pooling method is applied then.
- ✓ Flattening the image using the Flatten method.
- ✓ Adding the output layer and applying a suitable activation function. This is called full connection method.
- ✓ Compiling and training the object detection model
- ✓ Predicting the model and evaluating the accuracy.

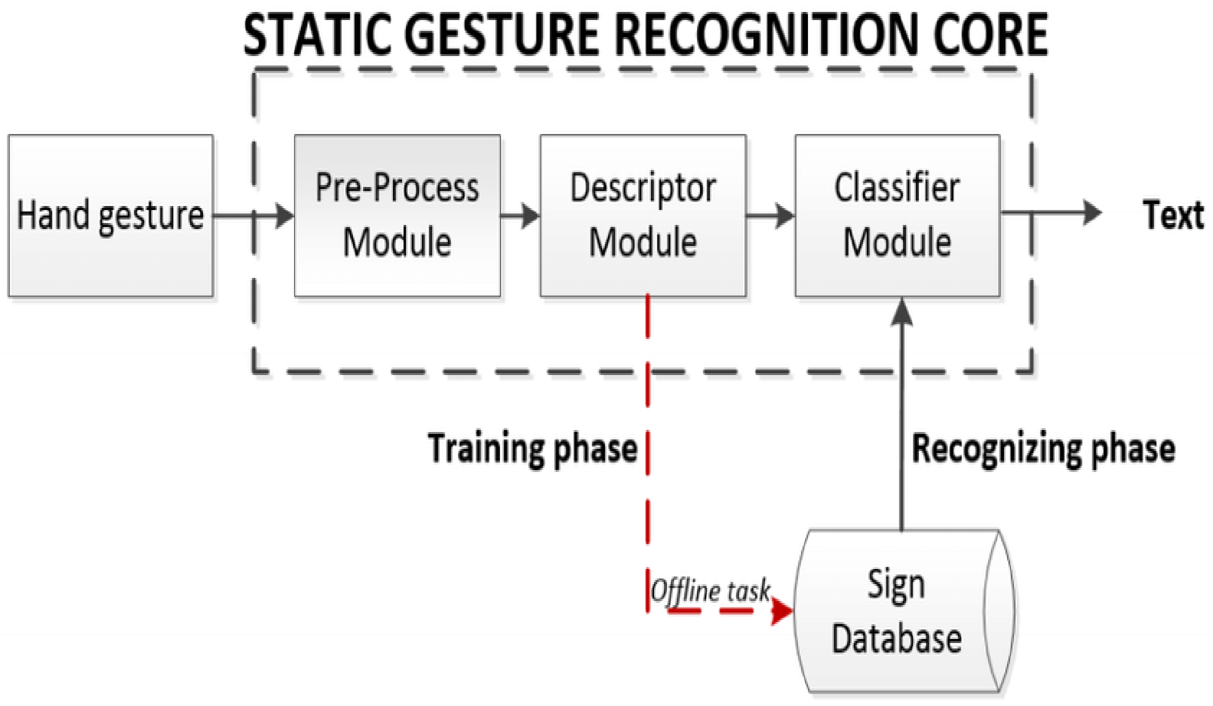


Fig. 4.2 Flow chart of the Proposed Model

Web Developing steps:

- ✓ The frontend is done using Html, CSS, and JavaScript.
- ✓ Backend is done using Django
- ✓ The database used PostgreSQL

Finally, to compound all these challenges, there is the issue of signer independence. While larger data sets are starting to appear, few allow true tests of signer independence over long continuous sequences. Maybe this is one of the most urgent problems in SLR that of creating data sets which are not only realistic, but also well annotated to facilitate machine learning. Despite these problems recent uses of SLR include translation to spoken language, or to another sign language when combined with avatar technology. SLR is also set to be used as an annotation aid, to automate annotation of sign video for linguistic research, currently a time-consuming and expensive task. Fig 4.2 shows the flow of the model proposed.

CHAPTER 5

IMPLEMENTATION

5.1 Dataset

We have used multiple datasets and trained multiple models to achieve good accuracy.

5.1.1 ASL Alphabet

The data is a collection of images of the alphabet from the American Sign Language, separated into 29 folders that represent the various classes. The training dataset consists of 87000 images which are 200x200 pixels. There are 29 classes of which 26 are English alphabets A-Z and the rest 3 classes are SPACE, DELETE, and, NOTHING. These 3 classes are very important and helpful in real-time applications.

5.1.2 Sign Language Gesture Images Dataset

The dataset consists of 37 different hand sign gestures which include A-Z alphabet gestures, 0-9 number gestures, and also a gesture for space which means how the deaf (hard hearing) and dumb people represent space between two letters or two words while communicating. Each gesture has 1500 images which are 50x50 pixels, so altogether there are 37 gestures which mean there 55,500 images for all gestures. Convolutional Neural Network (CNN) is well suited for this dataset for model training purposes and gesture prediction.

5.2 Data Pre-processing

An image is nothing more than a 2-dimensional array of numbers or pixels which are ranging from 0 to 255. Typically, 0 means black, and 255 means white. Image is defined by mathematical function f(x, y) where 'x' represents horizontal and 'y' represents vertical in a coordinate plane. The value of f(x, y) at any point is giving the pixel value at that point of an image. Image Pre-processing is the use of algorithms to perform operations on images. It is important to Pre-process the images before sending the images for model training. For example, all the images should have the same size of 200x200 pixels. If not, the model cannot be trained.

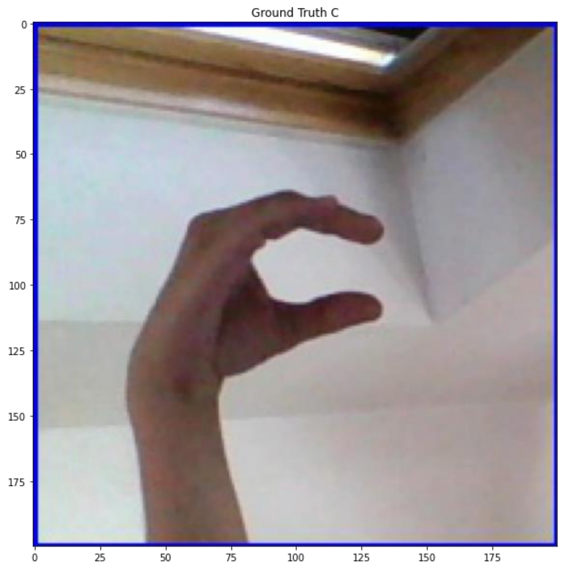


Fig. 5.1. Sample Image without Pre-processing

The steps we have taken for image Pre-processing are:

- ✓ Read Images.
- ✓ Resize or reshape all the images to the same
- ✓ Remove noise.
- ✓ All the image pixels array are converted to 0 to 255 by dividing the image array by 255.

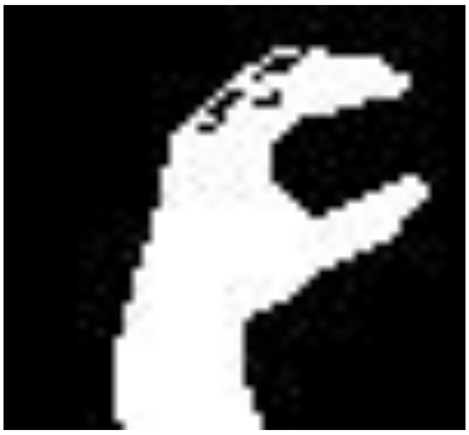


Fig. 5.2. Pre-Processed Image

5.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most popular architectures of deep learning which simulate biological nervous system like Artificial Neural Networks (ANNs). AlexNet, GoogleNet, Squeez Net and ResNet are the most common architectures of CNN. In comparison with ANNs, CNNs take the advantage of local connections instead of fully connections in all layers except the last layer.

CNNs structure contains series of most common layers as shown in Fig 4.3:

- ✓ Convolution Layer
- ✓ Pooling Layer
- ✓ Fully connected layer

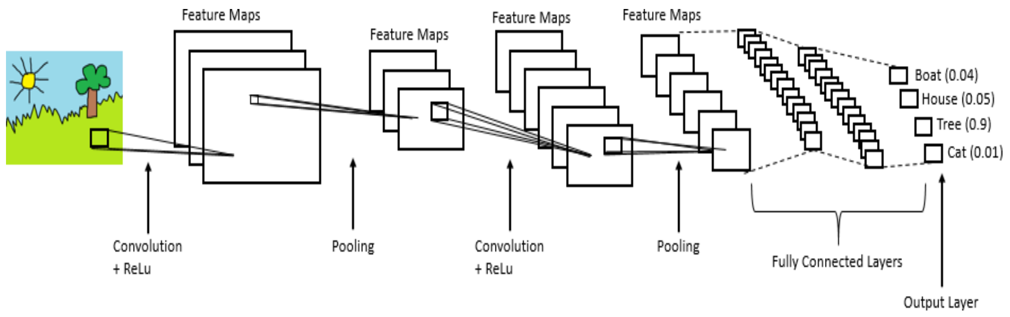


Fig. 5.3. CNN Layers

5.3.1 Convolution Layer

The first layer is convolutional layer. In this layer, each region that contains feature maps is connected to the feature maps of a local region in the previous layer by calculating weights that are known as kernels (filter banks). Sum of all local weights goes through a non- linearity function, e.g. Relu.

Input image, feature detector and feature maps are the three essential elements that enter into the convolution operation. The basic idea of the convolution is Input image x Feature detector = Feature Map. The main advantage of using feature maps is reducing the size of the input image, making it easier to read. A convolution matrix to adjust an image. It can be used to blur, edge detect and sharpen the image. The Eqn 1 represents the convolution function.

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \quad \text{(Eqn 1)}$$

The two main down sides of convolution are shrinking of outputs and the loss of information, especially in the corners of the image. In order to overcome this, padding is done. In simple words, it can described as an additional layer that can be added to the border of the image. This helps in increasing the accuracy.

$$n - f + 1 = 6$$
$$n = 6 - 1 + f$$
$$n = 5 + 3 = 8$$

The value of n obtained here is 8. If a padding p=1 is applied then, we get an image of size 8x8 is obtained. The following is the result obtained on multiplying the image with the filter.

$$\text{Result} = n + 2p - f + 1$$
$$\text{Result} = 6 + 2 - 3 + 1 = 6$$

The resultant matrix, thus obtained is 6x6 and there is no loss of data. The original size of the image can be retained if padding is applied. Any number of convolution and padding can be applied in order to retain the size of the image.

5.3.2 Pooling Layer

Pooling layer is the second common layer of CNNs which is used to decrease the spatial dimension of the output of convolutional layer without any change in depth. The advantage of using pooling layer is that by decreasing computational operations it prevents the overfitting in training process. Pooling layer contains min, max and average operations as shown in fig 4.4. However max pooling layer has achieved reasonable results in most fields.

The main objective of max pooling is to reduce dimensionality, down-sample an input representation and allowing the assumptions made about the features contained in the sub-regions. It can be described as a sample-based discretization process. In max pooling, only the high pixel values are being taken and hence, we are clearly able to recognize the face in the image.

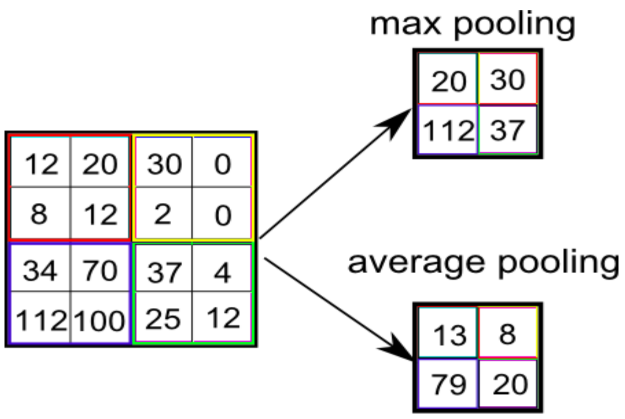


Fig. 5.4. Types of Pooling

5.3.3 Fully connected layer

The third common layer is fully connected layer. Before this, flattening must be done as shown in fig 4.5. Flattening can be defined as the process of conversion all the resultant two-dimensional arrays into a single long continuous linear vector. And it is connected to the final classification model, which is called a fully-connected layer as shown in fig 4.6. In other words, all the pixel data is put in one line and make connections with the final layer.

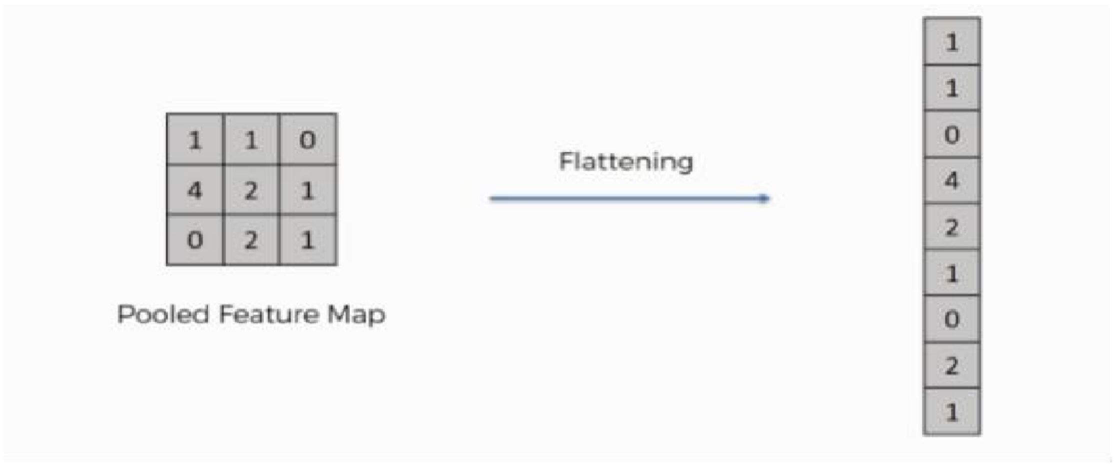


Fig. 5.5. Flattening

In the fully connected layer shown in fig 4.6 ,each neuron is not only connected to the all neurons of the previous layer but also calculated scores of dataset's classes are given in this layer. Moreover, generally in the last most convolutional layers softmax function is utilized to calculate the probable distribution through the labels of the class.

5.4 Convolution Neural Network (CNN) Architectures

5.4.1 LeNet-5 Architecture

Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters. The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification LeNet-5 is one of the simplest architectures. It has 2 convolutional and 3 fully-connected layers. The LeNet-5 architecture consists of two pairs of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully connected layers, and finally a SoftMax classifier.

The first layer is the input layer with feature map size 32X32X1. Then we have the first convolution layer with 6 filters of size 5X5 and stride is 1. The activation function used at his layer is tanh. The output feature map is 28X28X6. Next, we have an average pooling layer with filter size 2X2 and stride 1. The resulting feature map is 14X14X6. Since the pooling layer doesn't affect the number of channels. After this comes the second convolution layer with 16 filters of 5X5 and stride 1. Also, the activation function is tanh. Now the output size is 10X10X16. Again comes the other average pooling layer of 2X2 with stride 2. As a result, the size of the feature map reduced to 5X5X16. The final pooling layer has 120 filters of 5X5 with stride 1 and activation function tanh. Now the output size is 120. The next is a fully connected layer with 84 neurons that result in the output to 84 values and the activation function used here is again tanh. The last layer is the output layer with 10 neurons and Softmax function. The Softmax gives the probability that a data point belongs to a particular class. The highest value is then predicted.

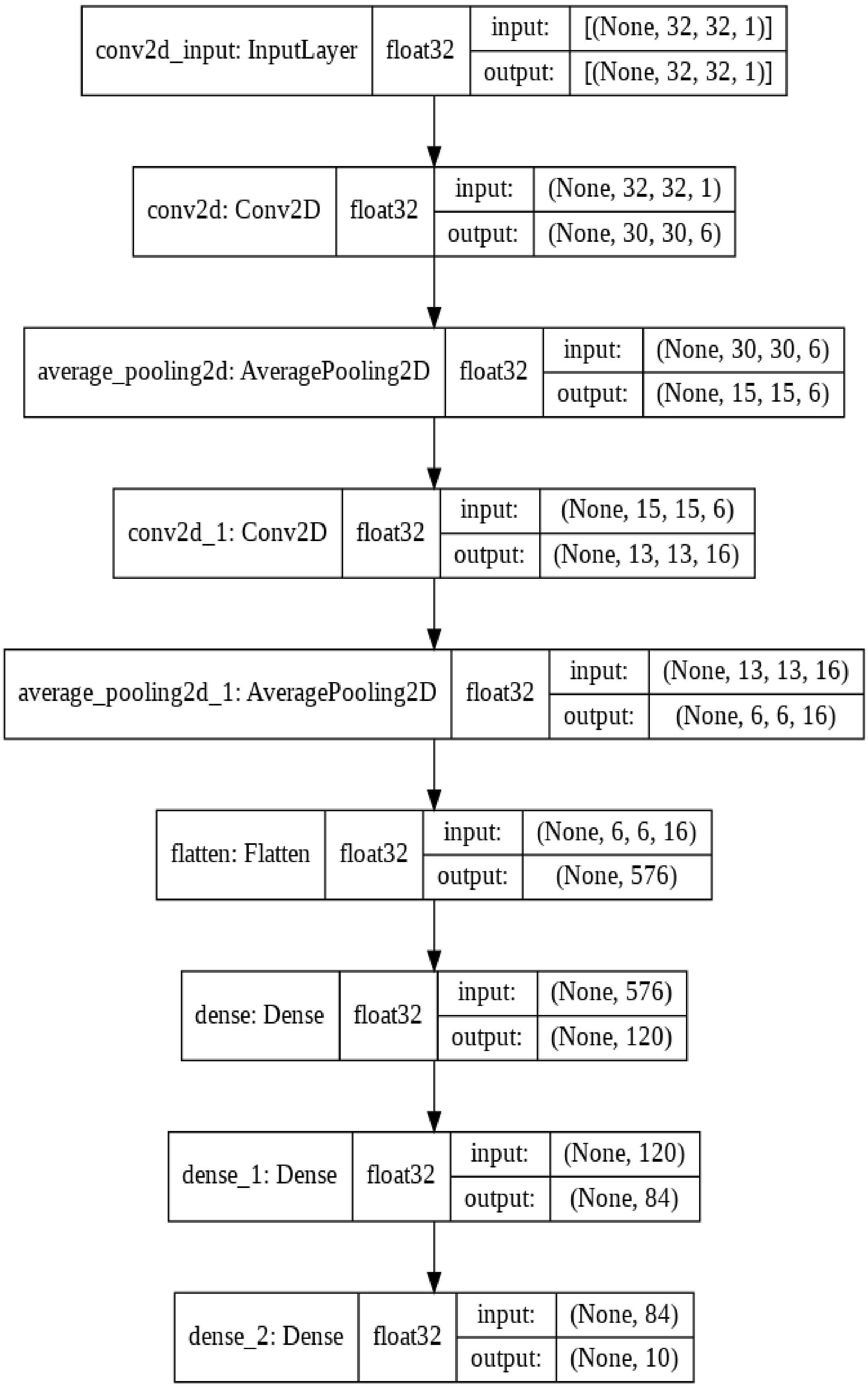

conv2d_input: InputLayer | float32 | input: [(None, 32, 32, 1)] | output: [(None, 32, 32, 1)]
conv2d: Conv2D | float32 | input: (None, 32, 32, 1) | output: (None, 30, 30, 6)
average_pooling2d: AveragePooling2D | float32 | input: (None, 30, 30, 6) | output: (None, 15, 15, 6)
conv2d_1: Conv2D | float32 | input: (None, 15, 15, 6) | output: (None, 13, 13, 16)
average_pooling2d_1: AveragePooling2D | float32 | input: (None, 13, 13, 16) | output: (None, 6, 6, 16)
flatten: Flatten | float32 | input: (None, 6, 6, 16) | output: (None, 576)
dense: Dense | float32 | input: (None, 576) | output: (None, 120)
dense_1: Dense | float32 | input: (None, 120) | output: (None, 84)
dense_2: Dense | float32 | input: (None, 84) | output: (None, 10)

Fig 5.6. LeNet Architecture

5.4.2 MobileNet V2 Architecture

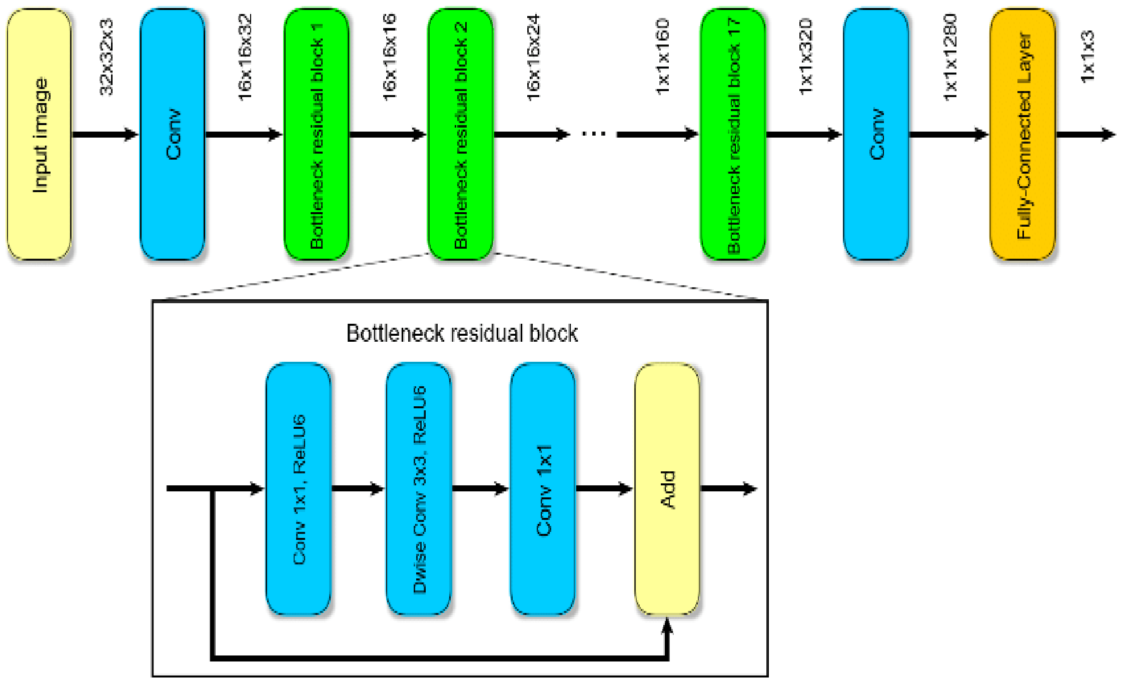


Fig. 5.7. MobileNet V2 Implementation

In the previous version MobileNetV1, Depthwise Separable Convolution is introduced which dramatically reduce the complexity cost and model size of the network, which is suitable to Mobile devices, or any devices with low computational power. In MobileNetV2, a better module is introduced with inverted residual structure. Non-linearities in narrow layers are removed this time. With MobileNetV2 as backbone for feature extraction, stateof-the-art performances are also achieved for object detection and semantic segmentation.

MobileNetV2 is a convolutional neural network architecture that performs well on mobile devices. The architecture of MobileNetV2 contains the fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. This network is lightweight and efficient.

5.4.3 Own Architecture

In our own architecture, we have implemented 3 convolution layers followed by batch normalization and max pooling, followed by global average pooling with dense layer and batch normalization, and a final dense layer for classification.

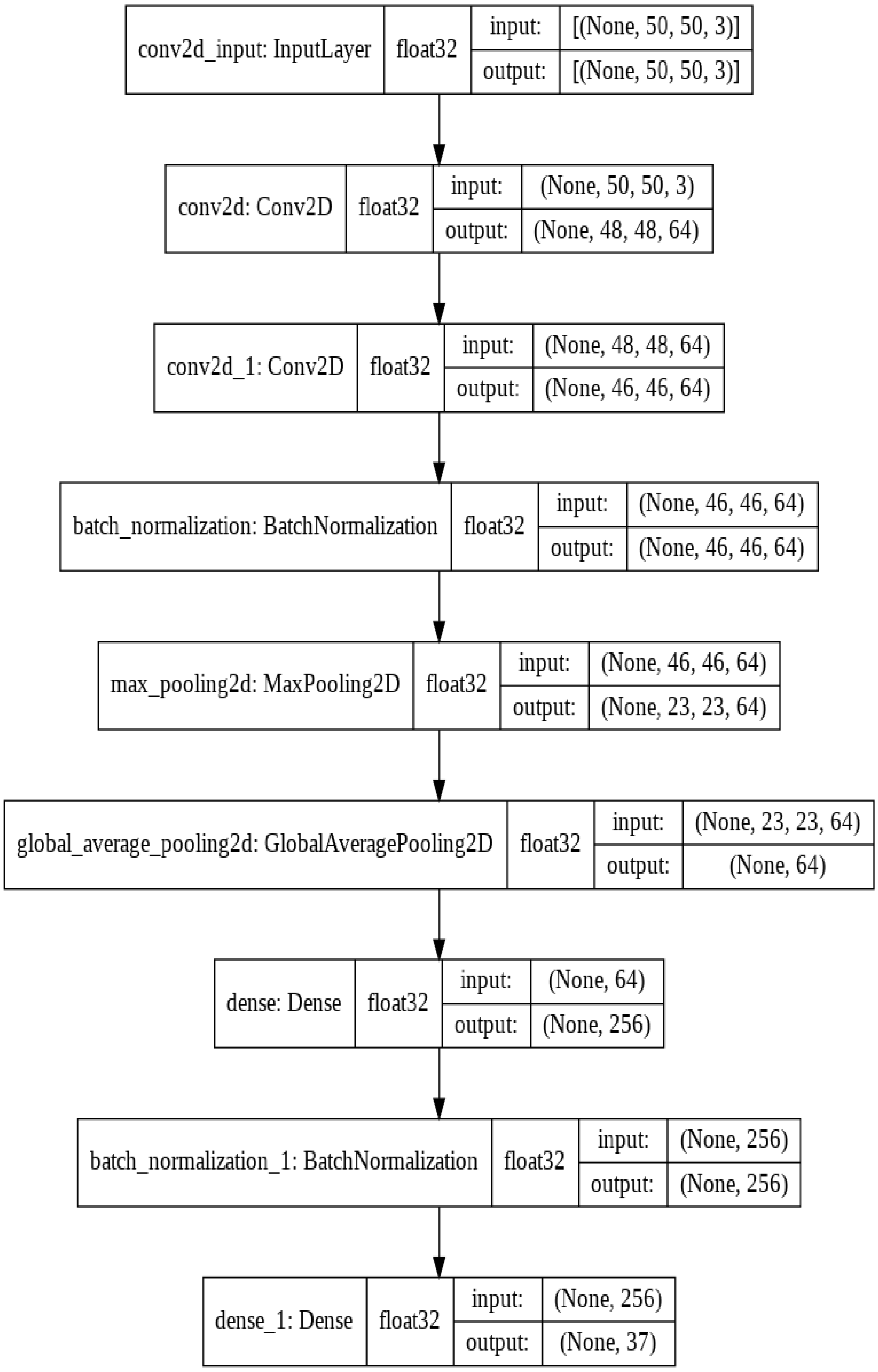


Fig.5.8 Architecture Implementation

5.5 Proposed Model

We have trained 3 models with 2 different datasets to perform well on unseen datasets. We have trained LeNet-5, MobineNetV2 and, our own architectures. We have not taken the best model out of 3 we have taken all 3 models and made a final model that will perform an ensemble of these 3 models.

5.5.1 Neural Network Ensemble Horizontal Voting

In Machine Learning we have an ensemble technique where we train multiple sub-models and average them. Random Forest algorithm is an example where it uses multiple Decision tree algorithms. Similarly, we can perform ensemble for Neural Networks [4] as well. There are a lot of ensemble techniques for Neural Network like Stacked generalization [8], Ensemble learning via negative correlation [9] and, Probabilistic Modelling with Neural Networks [10] [11].

We have implemented the Horizontal Voting Ensemble method to improve the performance of neural networks. Horizontal voting is an ensemble technique for neural networks where we train several sub-models and make predictions using these sub-models. For the final predictions, we make predictions from all the sub-models and see which class has got maximum votes.

The final prediction will be the class that has the maximum votes. For this, we have used 3 models that are an odd number of sub-models to avoid an even number of votes for two classes in worst cases.

For MobileNetV2 model we have taken Adam optimizer with learning rate = 0.001, eta_1=0.9, beta_2=0.999, and epsilon=1e-07 For all the models while training we have used ReduceLROnPlateau callback with factor = 0.2, patience = 2, min_lr = 0.001. By using this horizontal ensemble technique, we have achieved 99.8% accuracy.

We have deployed all the models in Django Web Frameworks and built a simple frontend to accept the image from users and send the response. We have also built an API that will pop up the live camera and detects the hand gestures and then converts them to the corresponding English alphabets.

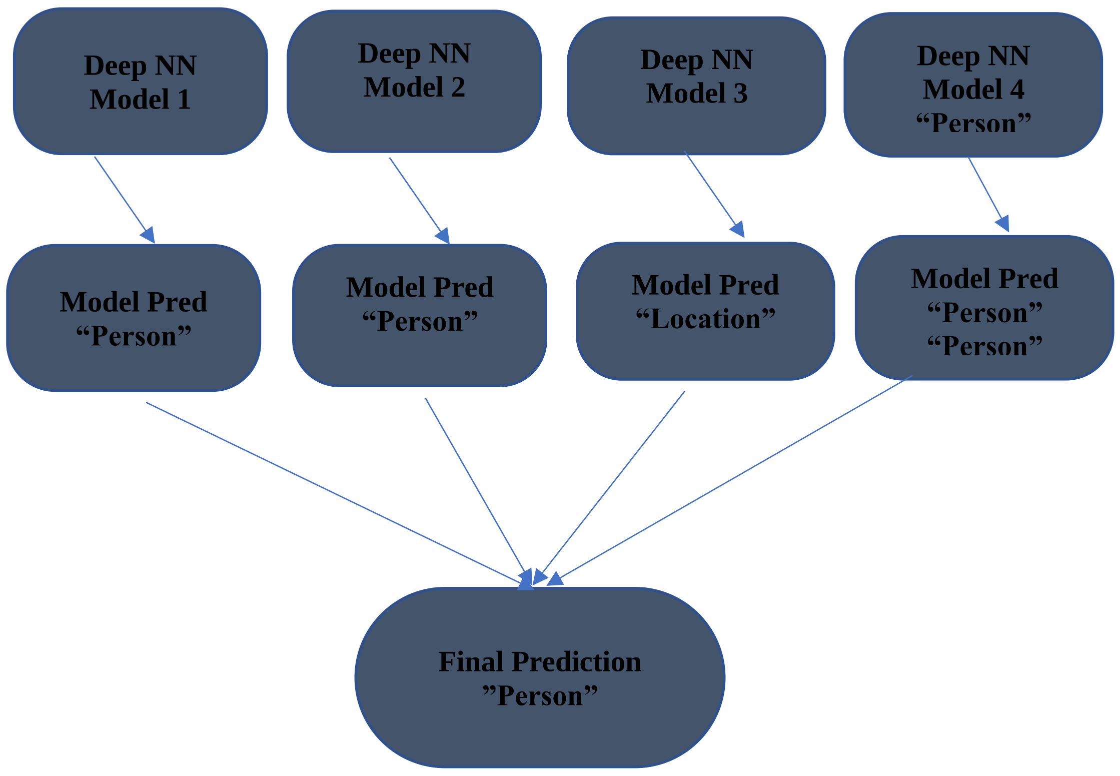


Fig. 5.9. Horizontal Voting

5.6 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions, but only that it does not function properly under specific conditions.

The scope of software testing often includes the examination of code as well as the execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

5.6.1 Testing Approaches

Static, dynamic and passive testing

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing.

Static testing is often implicit, like proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for these are either using stubs/drivers or execution from a debugger environment. Static testing involves verification, whereas dynamic testing also involves validation. Passive testing means verifying the system behavior without any interaction with the software product. Contrary to active testing, testers do not provide any test data but look at system logs and traces. They mine for patterns and specific behavior in order to make some kind of decisions. This is related to offline runtime verification and log analysis.

Exploratory approach

Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution. CemKaner, who coined the term in 1984, defines exploratory testing as "a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.

The "box" approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that the tester takes when designing test cases. A hybrid approach called grey-box testing may also be applied to software testing methodology. With the concept of grey-box testing—which develops tests from specific design elements - gaining prominence, this "arbitrary distinction" between black- and white-box testing has faded somewhat.

White-box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

Black-box testing

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Component interface testing

Component interface testing is a variation of black-box testing, with the focus on the data values beyond just the related actions of a subsystem component. The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units.

The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit.

Visual testing

The aim of visual testing is to provide developers with the ability to examine what was happening at the point of software failure by presenting the data in such a way that the developer can easily find the information she or he requires, and the information is expressed clearly. At the core of visual testing is the idea that showing someone a problem (or a test failure), rather than just describing it, greatly increases clarity and understanding. Visual testing, therefore, requires the recording of the entire test process – capturing everything that occurs on the test system in video format. Output videos are supplemented by real-time tester input via picture-in-a-picture webcam and audio commentary from microphones.

5.6.2 Testing Levels

Broadly speaking, there are at least three levels of testing: unit testing, integration testing, and system testing.

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other. Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development life cycle. Unit testing aims to eliminate construction errors before code is promoted to additional testing; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system. Integration tests usually involve a lot of code, and produce traces that are larger than those produced by unit tests. This has an impact on the ease of localizing the fault when an integration test fails. To overcome this issue, it has been proposed to automatically cut the large tests in smaller pieces to improve fault localization.

System testing

System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a login interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

5.6.3 Test Cases

A test case is a documentation which specifies input values, expected output and the preconditions for executing the test. Test case document is also a part of test deliverables, by reading test case document stakeholders get an idea about the quality of test cases written and the effectiveness of those test cases. Stakeholders can also provide inputs about the current set of test cases as well as suggest some more missing test cases. A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test cases for software helps to guide the tester through a sequence of steps to validate whether a software application is free of bugs and working as required by the end user.

The expected result tells the tester what they should experience as a result of the test performed. In this way we can determine if the test case is a “pass” or “fail”. In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and test is frequently done using a traceability matrix. Written test case includes a description of the functionality to be tested, and the preparation required to ensure that test can be conducted. For an applications or system without formal requirements, test case can be written based on accepted normal operation of programs of a similar class. A formal written test case is characterized by a known input and by an expected output, which worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition.

Test Case ID	**Test_01**
Test Input	**Image of letter ‘A’ in sign language**
Expected Output	**A**
Actual Output	**A**
Remarks	**Pass**

Table 6.1 Detection of letter A

In test case table 6.1, Image of letter ‘A’ in sign language is given as input. Model output

Test Case ID	Test_02
Test Input	Image of letter 'M' in sign language
Expected Output	M
Actual Output	M
Remarks	Pass

Table 6.2 Detection of Letter M

In test case table 6.2, Image of letter 'M' in sign language is given as input. Detection by the model is M. The actual output is M.

Test Case ID	Test_03
Test Input	Image of letter 'S in sign language
Expected Output	S
Actual Output	S
Remarks	Pass

Table 6.3 Detection of Letter S

In test case table 6.3 Image of letter 'S in sign language is given as input. Detectionby the model is S. The actual output is S.

Test Case ID	Test_04
Test Input	Image of number '1'
Expected Output	1
Actual Output	1
Remarks	Pass

Table 6.4 Detection of number 1

In test case table 6.4 a Image of number 1 in sign language is given as input Activity Detection by the model is 1. The actual output is 1.

CHAPTER 6

RESULTS

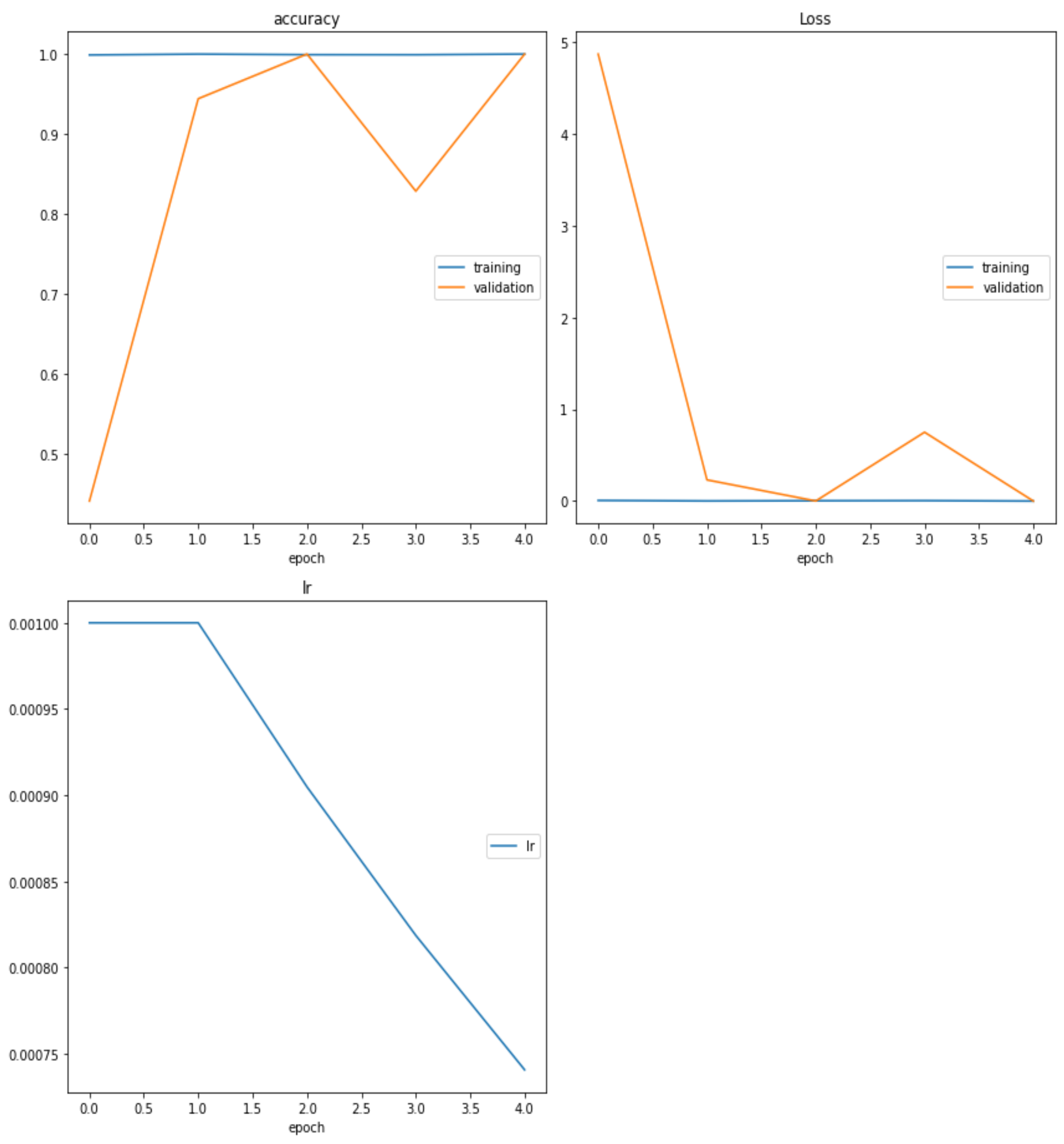


Fig 6.1 Trained Model Plots

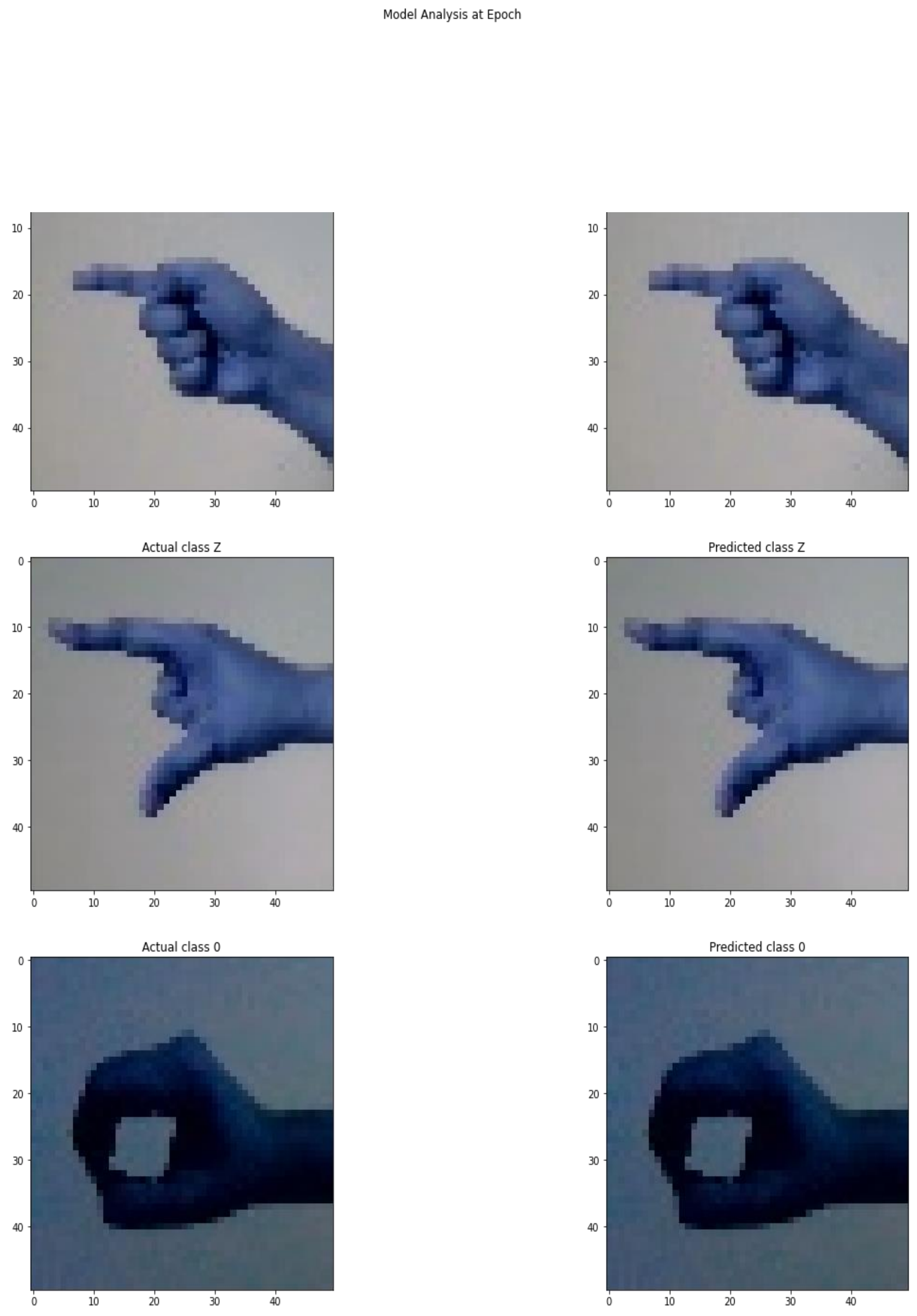
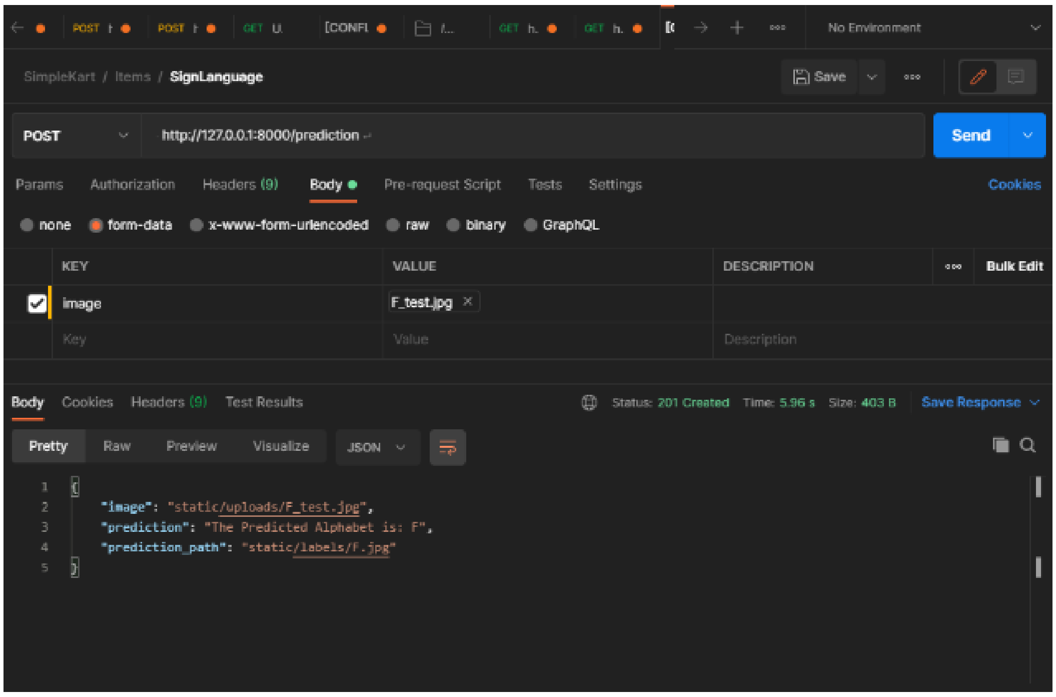


Fig 6.2 Model Prediction

Models	Accuracy
MobileNetV2	98.9%
LeNet-5	97%
Own Model	98%
Ensemble	**99.8%**

Table. 1. Performance Results

All the models performed well on the test cases. After applying the horizontal voting ensemble technique for these 3 models, we have achieved almost 100% accuracy.



SimpleKart / Items / SignLanguage

POST http://127.0.0.1:8000/prediction

Body: form-data

KEY	VALUE	DESCRIPTION
image	F_test.jpg	

Status: 201 Created Time: 5.96 s Size: 403 B

```
{
    "image": "static/uploads/F_test.jpg",
    "prediction": "The Predicted Alphabet is: F",
    "prediction_path": "static/labels/F.jpg"
}
```

Fig 6.3 JSON Response from the Application

We have used Django Rest Frameworks as a backend for our project. This is the sample JSON response that is sent to the frontend when a user inputs an Image.

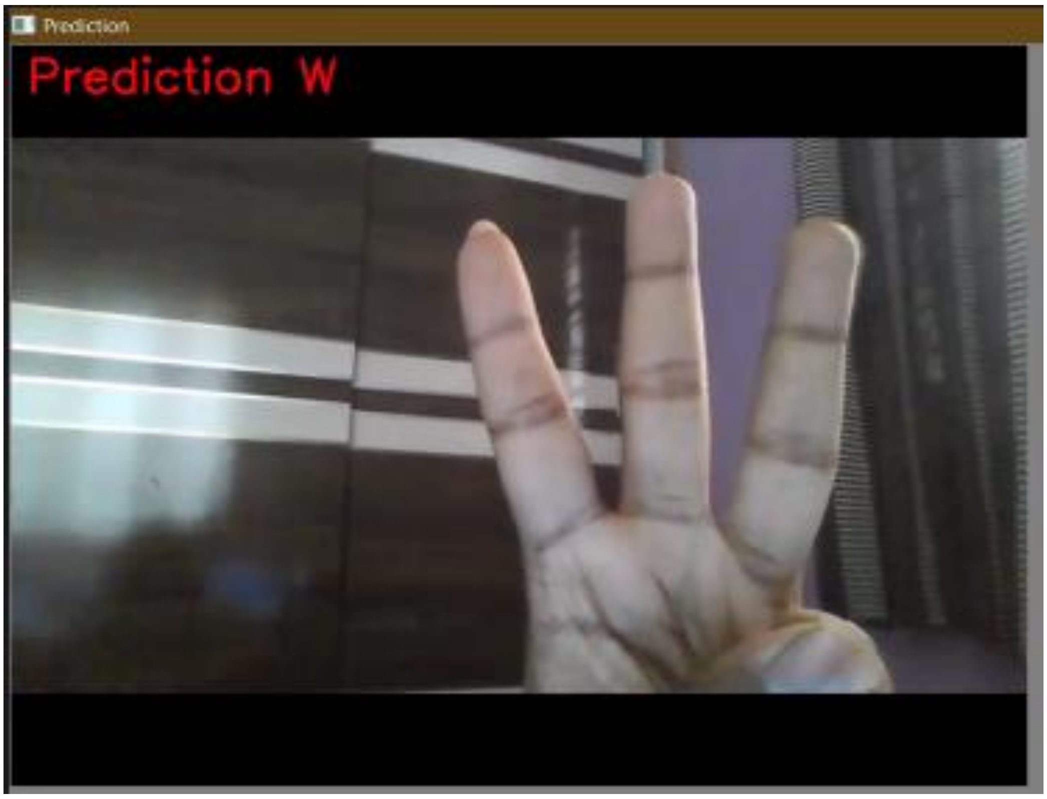


Fig 6.4 Prediction

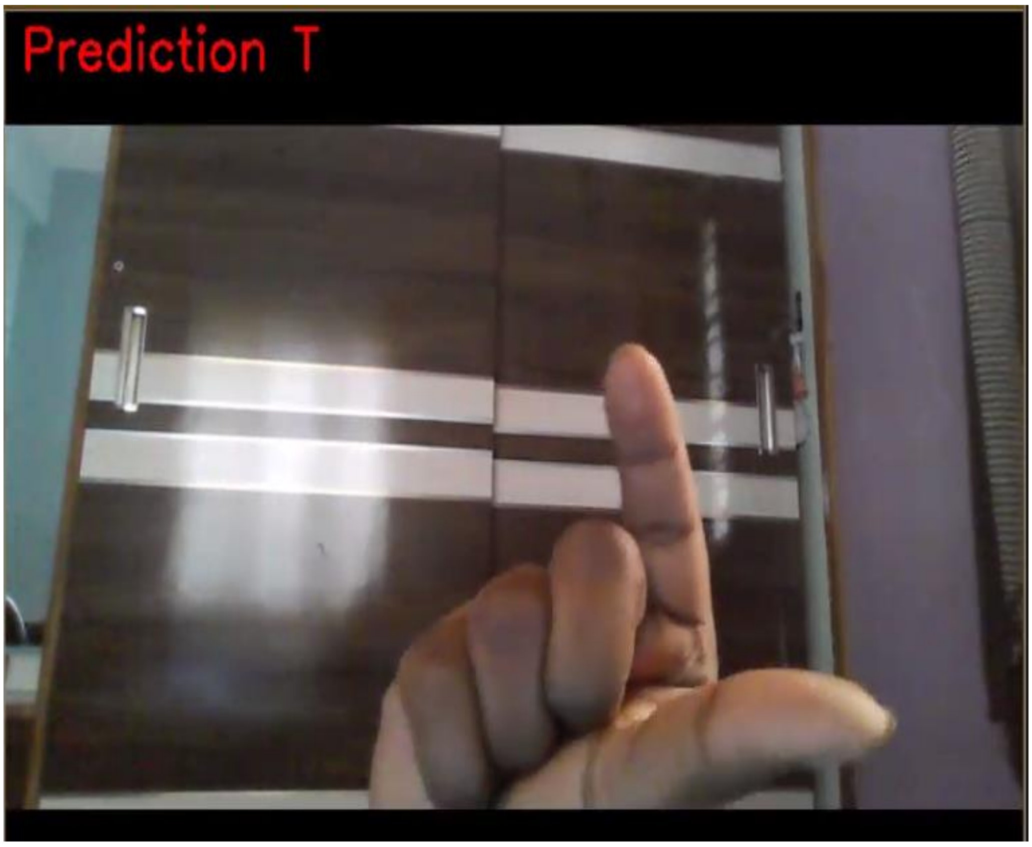


Fig 6.5 Prediction

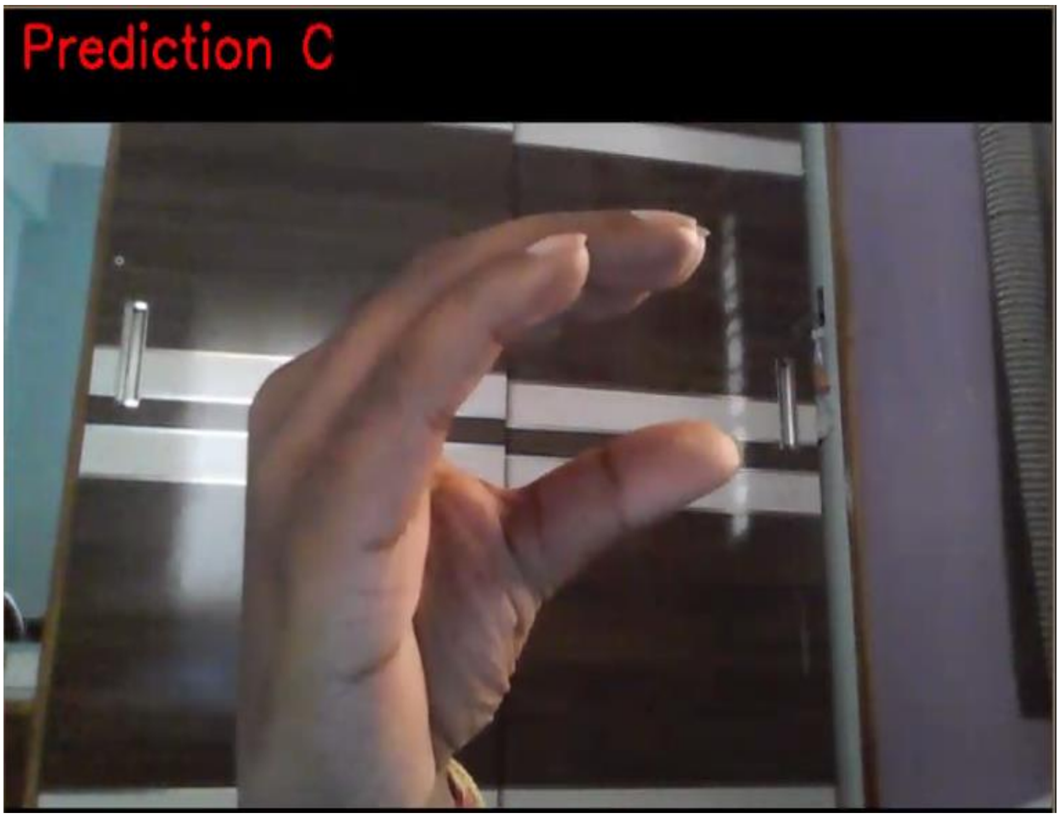


Fig 6.5 Prediction

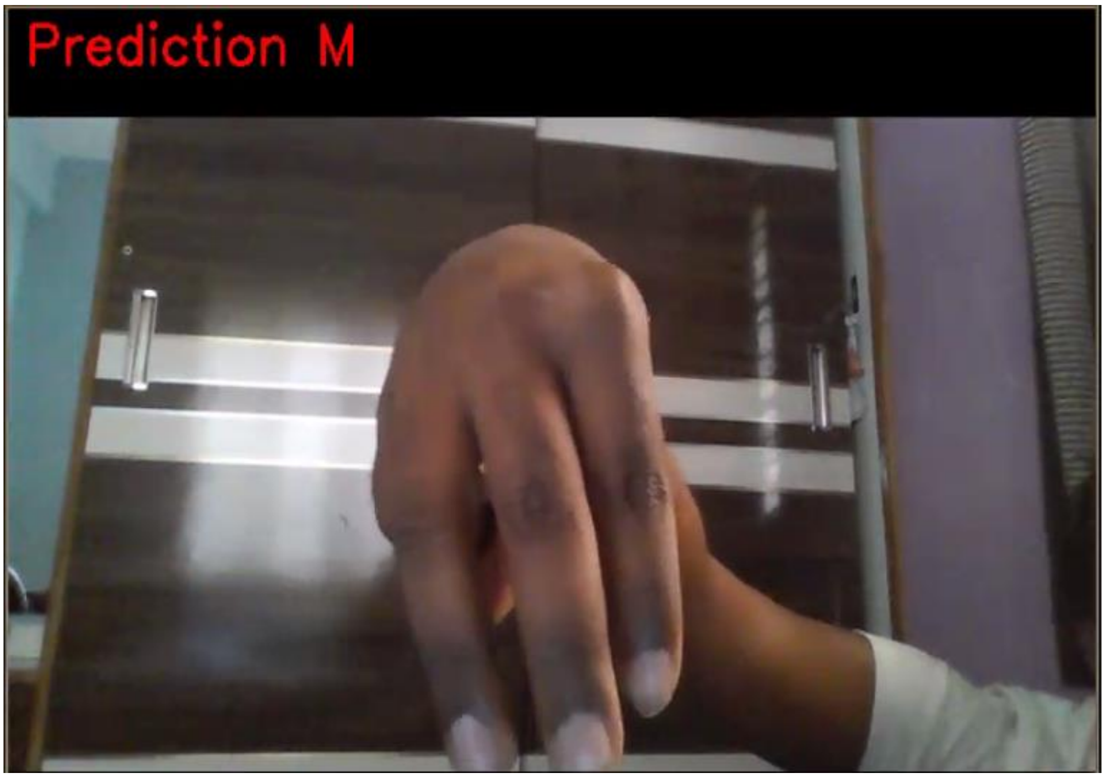


Fig 6.6 Prediction

CHAPTER 7

CONCLUSION

Communications between deaf-mute and a normal person have always been a challenging task. The goal of this project is to reduce the barrier of communication by contributing to the field of automatic sign language recognition. Through this work, a CNN classifier is constructed which is capable of recognizing static sign language gestures. A basic GUI application is created to test our classifier in this system. The application allows users to select the static sign gestures as input and it will speak out the words or sentences corresponding to the gesture. We have trained our model for 26 symbols which include alphabets. We were able to achieve an accuracy of 99% for our CNN classifier.

REFERENCES

[1] TensorFlow Documentation

[2] Django Rest Frameworks Documentation

[3] CNN Research paper

[4] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.

[5] David H. Wolpert, Stacked generalization, Neural Networks, Volume 5, Issue 2, 1992, Pages 241-259, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(05)80023-1.

[6] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, Volume 12, Issue 10,1999, Pages 1399-1404, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(99)00073-8.

[7] MacKay D.J.C. (1995) Developments in Probabilistic Modelling with Neural Networks — Ensemble Learning. In: Kappen B., Gielen S. (eds) Neural Networks: Artificial Intelligence and Industrial Applications. Springer, London. https://doi.org/10.1007/978-1-4471-3087-1_37

[8] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-9326-7_1