

HAND SIGN LANGUAGE INTERPRETER

Submitted in partial fulfilment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

SANJAY.S (37110680)



DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119

MARCH – 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **SANJAY S (17SCS518)** who carried out the project entitled "**HAND SIGN LANGUAGE INTERPRETER**" under my supervision from **November 2020 to March 2021**.

Internal Guide

Dr. T. PREM JACOB, M.E.,PhD.,

Head of the Department

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **SANJAY S** hereby declare that the Project Report entitled “**HAND SIGN LANGUAGE INTERPRETER**” is done by me under the guidance of **Dr. T. Prem Jacob,M.E.,PhD.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide , **Dr. T. Prem Jacob ,M.E.,PhD., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Hand gestures are those gestures that are very expressive in nature. Hand gestures are the non-verbal method of communication. These hand gestures involve different movements or positioning of fingers which convey a message. These hand gestures may represent any short messages like numbers, greetings, or any other message. Here we are going to build and train a machine learning model which classifies images of various gestures. Here we implement this model in many types of convolution neural networks (CNN) and obtain an accuracy graph. We choose the best convolution neural network with the best results and deploy that model in the Django framework by converting it to a (.h5) file. Hence the output will be in a web application.

A human gesture is a non-verbal form of communication and is critical in human-robot interactions. Vision-based gesture recognition methods play a key role to detect hand motion and support such interactions. Hand gesture recognition allows a appropriate, and usable interface between users and devices. Hand gestures can be used for various fields which makes it be able to be implemented for communication and further. Hand gesture recognition is not only useful for people who are hearing-impaired or disabled but also for the people who have experienced a stroke, as they need to communicate with other people using different common essential gestures such as the sign of eating, drink, family and, more. In this paper, a system for recognizing hand gesture based on Convolutional Neural Network (CNN) is proposed. The developed method is evaluated and compared between training and testing modes based on several metrics such as execution time, accuracy, sensitivity, specificity, positive and negative predictive value, likelihood and root mean square. Results show that testing accuracy is 90% using CNN and is an effective technique in extracting distinct features and classifying data.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
1.	INTRODUCTION	1
	1.1 GENERAL INTRODUCTION	1
	1.2 AMERICAN SIGN LANGUAGE	5
2.	AIM, SCOPE AND LITERATURE SURVEY	9
	2.1 AIM	9
	2.2 SCOPE	9
	2.3 LITERATURE SURVEY	11
3.	METHODOLOGY	14
	3.1 ARCHITECTURE DIAGRAM	14
	3.2 EXISTING SYSTEM	14
	3.3 UML DIAGRAMS	15
	3.3.1 USE CASE DIAGRAM	15
	3.3.2 CLASS DIAGRAM	17
	3.3.3 ACTIVITY DIAGRAM	17
	3.3.4 SEQUENCE DIAGRAM	18
	3.3.5 COLLABORATION DIAGRAM	19
	3.3.6 ER DIAGRAM	19
	3.4 DESCRIPTION	20
	3.4.1 DATA PROCESSING	20
	3.4.2 TRAINING	21
	3.4.3 GESTURE CLASSIFICATION	21
	3.5 ALGORITHM	22
	3.6 CREATING A DEEP LEARNING MODEL	22

	3.7 MODULE-1 (MANUAL CNN)	25
	3.8 MODULE-2 (ALEXNET)	27
	3.9 MODULE-3 (LENET)	29
	3.10 DJANGO FRAMEWORK	30
	3.11 REQUIREMENT SPECIFICATIONS	31
4.	RESULTS AND DISCUSSIONS	33
5.	CONCLUSION AND FUTURE WORK	38
	REFERENCES	41
	APPENDIX	43
	A. PLAGARISM REPORT	43
	B. JOURNAL PAPER	44
	C. SOURCE CODE	49

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.1.1	BLOCK DIAGRAM	1
1.1.2	DEEP LEARNING	3
1.1.3	PROCESS OF DEEP LEARNING	4
1.2.1	AMERICAN SIGN LANGUAGE	7
1.2.2	CONVERSATION USING ASL	8
3.1	ARCHITECTURE DIAGRAM	14
3.3.1	USE CASE DIAGRAM	16
3.3.2	CLASS DIAGRAM	17
3.3.3	ACTIVITY DIAGRAM	18
3.3.4	SEQUENCE DIAGRAM	18
3.3.5	COLLOBORATION DIAGRAM	19
3.3.6	ER DIAGRAM	20
3.5	DEEP LEARNING	24
3.7.1	CONVOLUTION NEURAL NETWORK	25
3.7.2	MANUAL CNN ACCURACY & LOSS GRAPH	27
3.8.1	ALEXNET ARCHITECTURE	28
3.8.2	ALEXNET ACCURACY & LOSS GRAPH	29
3.9.1	LENET ARCHITETURE	29
3.9.2	LENET ACCURACY & LOSS GRAPH	30
3.10	DJANGO FRAMEWORK	31
4.1	JUPYTER NOTEBOOK	33
4.2	PYCHARM SCREEN	34
4.3	HOME SCREEN	35
4.4	INTERMEDIATE OUTPUT SREEN	36
4.5	FINAL OUTPUT SCREEN	37

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION:

Sign language is widely used by people who can't speak and hear or people who can hear but can't speak. A sign language is composed of various gestures formed by different hand shapes, movements and orientations of hands or body, or facial expressions. There are various sign languages across the world, each with its own vocabulary. These include American Sign Language (ASL) in Northern America, British Sign Language (BSL) in Great Britain, Japanese Sign Language (JSL) in Japan, South African Sign Language (SASL), Indian Sign language (ISL) etc. Gestures are used by the deaf people to express their thoughts. But the use of these gestures are always limited in the deaf-dumb community, normal people never try to learn the sign language. This causes a big gap in communication between the deaf-dumb people and the normal people. Usually, deaf people seek the help of sign language interpreters for translating their thoughts to normal people and vice versa. But this system is very costly and does not work throughout the life period of a deaf person. So, a system that automatically recognizes the sign language gestures is necessary. Such a system can minimize the gap between deaf people and normal people in the society.

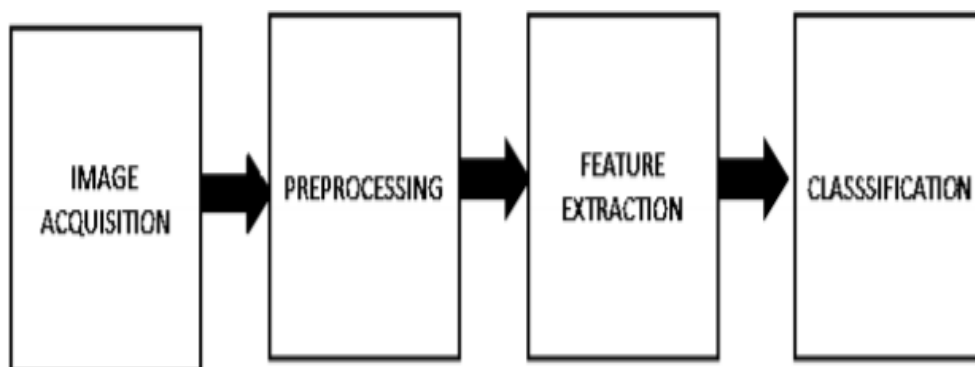


Fig.1.1.1: Block Diagram

The general block diagram of hand gesture recognition system shown in Fig 1.1 as above, it consists of four basic steps such as, Image acquisition, Pre-processing, Feature extraction and classification. Hand Gestures provide a natural and intuitive communication modality for human-computer interaction. Efficient human computer interfaces (HCIs) have to be developed to allow computers to visually recognize in hand gestures. However, vision-based hand tracking and gesture recognition is a challenging problem due to the complexity of hand gestures, which are rich in diversities due to high degrees of freedom (DOF) involved by the human hand. In order to successfully fulfill their role, the hand gesture HCIs have to meet the requirements in terms of realtime performance, recognition accuracy and robustness against transformations and cluttered background. To facilitate the communication between the Deaf and hearing persons, many gesture recognition systems used the help of color markers or data gloves to make the task easier. However, using of markers and gloves sacrifices the user's convenience. In the dissertation work, focus is given hand gesture recognition without help of any markers and gloves. Detecting and tracking hand gestures in a sequence of images help in extracting hand region. Thus, processing time will be reduced and accuracy will be increased as the features of that region will represent the hand gesture only.

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

Fig 1.1.2: Deep Learning

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. However, the data, which normally is unstructured, is so vast

that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

Fig 1.1.3: Process of Deep Learning

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other

classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

1.2 AMERICAN SIGN LANGUAGE:

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing, and is used by many hearing people as well. There is no universal sign language. Different sign languages are used

in different countries or regions. For example, British Sign Language (BSL) is a different language from ASL, and Americans who know ASL may not understand BSL. Some countries adopt features of ASL in their sign languages.

No person or committee invented ASL. The exact beginnings of ASL are not clear, but some suggest that it arose more than 200 years ago from the intermixing of local sign languages and French Sign Language (LSF, or *Lange des Signes Française*). Today's ASL includes some elements of LSF plus the original local sign languages; over time, these have melded and changed into a rich, complex, and mature language. Modern ASL and modern LSF are distinct languages. While they still contain some similar signs, they can no longer be understood by each other's users.

ASL is a language completely separate and distinct from English. It contains all the fundamental features of language, with its own rules for pronunciation, word formation, and word order. While every language has ways of signaling different functions, such as asking a question rather than making a statement, languages differ in how this is done. For example, English speakers may ask a question by raising the pitch of their voices and by adjusting word order; ASL users ask a question by raising their eyebrows, widening their eyes, and tilting their bodies forward.

Just as with other languages, specific ways of expressing ideas in ASL vary as much as ASL users themselves. In addition to individual differences in expression, ASL has regional accents and dialects; just as certain English words are spoken differently in different parts of the country, ASL has regional variations in the rhythm of signing, pronunciation, slang, and signs used. Other sociological factors, including age and gender, can affect ASL usage and contribute to its variety, just as with spoken languages. Fingerspelling is part of ASL and is used to spell out English words. In the fingerspelled alphabet, each letter corresponds to a distinct handshape. Fingerspelling is often used for proper names or to indicate the English word for something.



Fig 1.2.1: American Sign Language

Parents should expose a deaf or hard-of-hearing child to language as soon as possible. The earlier a child is exposed to and begins to acquire language, the better that child's language, cognitive, and social development will become. Research suggests that the first few years of life are the most crucial to a child's development of language skills, and even the early months of life can be important for establishing successful communication with caregivers. Thanks to screening programs in place at almost all hospitals in the United States and its territories, newborn babies are tested for hearing before they leave the hospital. If a baby has hearing loss, this screening gives parents an opportunity to learn about communication options. Parents can then start their child's language learning process during this important early stage of development.

The NIDCD supports research on ASL, including its acquisition and characterization. Funded research includes studies to understand sign language's grammar, acquisition, and development, and use of sign language when spoken language access is compromised by trauma or degenerative disease, or when speech is difficult to acquire due to early hearing loss or injury to the nervous system.



Fig 1.2.2: Conversation Using ASL

Study of sign language can also help scientists understand the neurobiology of language development. In one study, researchers reported that the building of complex phrases, whether signed or spoken, engaged the same brain areas. Better understanding of the neurobiology of language could provide a translational foundation for treating injury to the language system, for employing signs or gestures in therapy for children or adults, and for diagnosing language impairment in individuals who are deaf.

The NIDCD is also funding research on sign languages created among small communities of people with little to no outside influence. Emerging sign languages can be used to model the essential elements and organization of natural language and to learn about the complex interplay between natural human language abilities, language environment, and language learning outcomes.

CHAPTER 2

AIM, SCOPE & LITERATURE SURVEY

2.1 AIM:

The goal of this project was to build a neural network able to classify which letter of the American Sign Language (ASL) alphabet is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written and oral language. Such a translator would greatly lower the barrier for many deaf and mute individuals to be able to better communicate with others in day-to-day interactions. The framework provides a helping-hand for speech-impaired to communicate with the rest of the world using sign language. This leads to the elimination of the middle person who generally acts as a medium of translation. This would contain a user-friendly environment for the user by providing speech/text output for a sign gesture input. This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exist in higher rates among the deaf population, especially when they are immersed in a hearing world. Large barriers that profoundly affect life quality stem from the communication disconnect between the deaf and the hearing. Some examples are information deprivation, limitation of social connections, and difficulty integrating in society. Most research implementations for this task have used depth maps generated by depth camera and high-resolution images.

2.2 SCOPE:

There are many deaf people in the world lost their job due to discrimination. Some deaf people cannot understand the sign language. Enormous amount of deaf people does not get their education in sign language. So, this sign language interpreter act as a medium for deaf people. However, written communication is practically not possible in real time, the hand sign language interpreter is the alternate solution to this condition. The main idea of this project is to help and save time for those people with disabilities. This developed model can help in resolve the condition of unemployment among the deaf and hearing loss people. 70% of families are not able to communicate

with their deaf children. This situation can be solved by this model. If the deaf people hand sign is predicted by the system, then output in the form of word which is easily understood by the people. In future this application can also implement on video games, television, etc. to control. The sign language can be well developed and can be widely used among all the disability people and this communication issue can be resolved.

Humans possess the voice capability for interaction and communication among each other. Unfortunately, not everybody has the capability of speaking and hearing. Sign language is a gesture representation that involves simultaneously combining hand shapes, orientation and movement of the hands, arms or body, and facial expressions to express fluently with a speaker's thoughts. The people who cannot speak makes use of the sign languages to communicate with other fellow vocally impaired person and even with other normal people who knows the meanings of sign languages or an interpreter is needed to translate the meanings of sign languages to other people who can speak and do not know the meanings of sign languages. However, it is not always possible for an individual to be around all the time to interpret the sign languages Hand Gesture Vocalizer for Dumb and Deaf People - 23 - and not everybody can learn the sign languages. Thus, another alternative is that we can use a computer or a smart phone as a mediator.

The objective of this project was to see if neural networks are able to classify signed ASL letters using simple images of hands taken with a personal device. This is in alignment with the motivation as this would make a future implementation of a real time ASL-to-oral/written language translator practical in an everyday situation. Hand sign language is commonly used by deaf people and hearing loss people for communication. The hand gesture and posture are used to communicate, here the image is processed using Convolutional Neural Network (CNN) algorithm. Convolutional Neural Network is a class of Deep learning, this algorithm is used to analyse the images. The image is analysed by various process in CNN such as pooling, convolution, etc. Deep learning is a function that process data and creates pattern in order to make decision in real time. The hand posture image is processed and analysed, then based on the trained models the system predicts the result. The sign language used here is American sign language, since this sign language is popular and widely used. The main difference between gesture and posture are, the

shape is considered in the posture and whereas the movement is considered in the gesture. one of the hand gesture application such as gesture-based interface used television which can eliminate the use of remote in future. The model is trained using Kaggle datasets using CNN.

2.3 LITERATURE SURVEY:

In [1] “Hand Gesture Recognition for Sign Language Using 3DCNN” proposed by Muneer Al-Hammadi, Ghulam Muhammad, Wadood Abdul states automatic hand gesture recognition has gained increasing importance for two principal reasons: the growth of the deaf and hearing-impaired population, and the development of vision based applications and touchless control on ubiquitous devices. As hand gesture recognition is at the core of sign language analysis a robust hand gesture recognition system should consider both spatial and temporal features. In the pre-processing phase, linear sampling was used to normalize the temporal dimension of hand gesture samples. For spatial dimension normalization, they used the length of the detected face and human body part ratios. they proposed approach by performing a holistic search to optimize all the hyperparameters. They will test the proposed approach online while receiving a live video feed. In this aspect, we may utilize the edge-cloud computing to distribute the processing over edge devices and the core cloud.

In [2] “A Novel Detection and Recognition Method for Continuous Hand Gesture Using FMCW Radar” proposed by Yong Wang, Aihu Ren, Mu Zhou states that a novel method for continuous hand gesture detection and recognition is proposed based on a frequency modulated continuous wave (FMCW) radar. Firstly, we adopt the 2- Dimensional Fast Fourier Transform (2D-FFT) to estimate the range and Doppler parameters of the hand gesture raw data, and construct the range-time map (RTM) and Doppler-time map (DTM). Meanwhile, we apply the Multiple Signal Classification (MUSIC) algorithm to calculate the angle and construct the AngleTime Map (ATM). Secondly, a hand gesture detection method is proposed to segment the continuous hand gestures using a decision threshold. Thirdly, the central time-frequency trajectory of each hand gesture spectrogram is clustered using the k-means algorithm, and then the Fusion Dynamic Time Warping (FDTW) algorithm is

presented

to

recognize

the

hand gestures. hand gesture detection and recognition method were proposed. Firstly, we collected the raw data of the radar to obtain the IF signal, and used the IF signal to estimate the RTM, DTM and ATM of hand gestures. Then, we proposed an amplitude-based detection method, which applied the amplitude of the normalized hand gesture spectrogram in conjunction with a threshold. Finally, the k-means algorithm was adopted to cluster the center time-frequency trajectories of RTM, DTM and ATM, and FDTW algorithm was proposed to recognize the hand gestures. Experimental results showed that the segmentation accuracy of the proposed hand gesture detection method could reach 96.17%, and the average recognition accuracy of the proposed FDTW algorithm for six types of hand gestures was improved by more than 5% while saving half of the running time.

In [3] “Hand Gesture Recognition Based on Active Ultrasonic Sensing of Smartphone: A Survey” proposed by Zhengjie Wang, Yushan Hou, Kangkang Jiang, Wenwen Dou states as follows. Hand gesture recognition has drawn wide attention in the field of ubiquitous computing because it provides us with simple and natural human-computer interaction mode. Among these various implementations, hand gesture recognition using ultrasonic signals of smartphone has become a hot research topic due to its various advantages. In this paper, we consider the smartphone as an active sonar sensing system to identify hand movements. Specifically, the speakers emit ultrasonic signal and the microphone on the same phone receives the changed echo affected by hand movements. This paper investigates the state-of-the-art hand gesture applications and presents a comprehensive survey on the characteristics of studies using the active sonar sensing system. Rapid development of hardware and software technology, smartphone is becoming a strong communication and entertainment tool and has become an indispensable electric device in daily lives because it provides us with various functions to facilitate our work, lives, entertainment, and mutual communications. A detailed discussion about these systems from signal acquisition, signal processing, and performance evaluation. Finally, based on current study trends, we discuss the limitations and open issues involved in human hand gesture recognition based on the ultrasonic signal of smartphone and present some potential solution to these issues.

In [4] “Electromyography-Based Hand Gesture Recognition System for Upper Limb Amputees” proposed by Sidharth Pancholi, Amit M. Joshi states as follows. Electromyography (EMG) signals are emerging as one of the indispensable biological parameters, which have extensive variety of utilizations in human-machine interaction, prosthetic device development, and rehabilitation devices. EMG measures muscle potential and can provide useful investigatory or diagnostic information regarding neuromuscular activity and rich motor control information for prosthetic devices. EMG signals can be decoded in order to control smart prosthetic devices using pattern recognition (PR) methods. This can be achieved by applying EMG electrodes on the residual limb, and a classification technique is utilized in the embedded platform to determine the user intention. a real-time arm gesture recognition system is presented for the amputees. This system can be utilized for myoelectric prosthetic control in the real-time environment, with above 91% of accuracy. Four-time domain features and LDA classifier are incorporated within the DSP processor. With this system, grasping motion can decode easily in 75 ms completion time after training model generation. This system has light weight and battery power support up to 40 Hrs. This system presents admirable results for low-power applications in rehabilitation and myoelectric prosthetics. Power optimization will further enhance the life time of the proposed system.

CHAPTER 3

METHODOLOGY

3.1 ARCHITECTURE DIAGRAM:

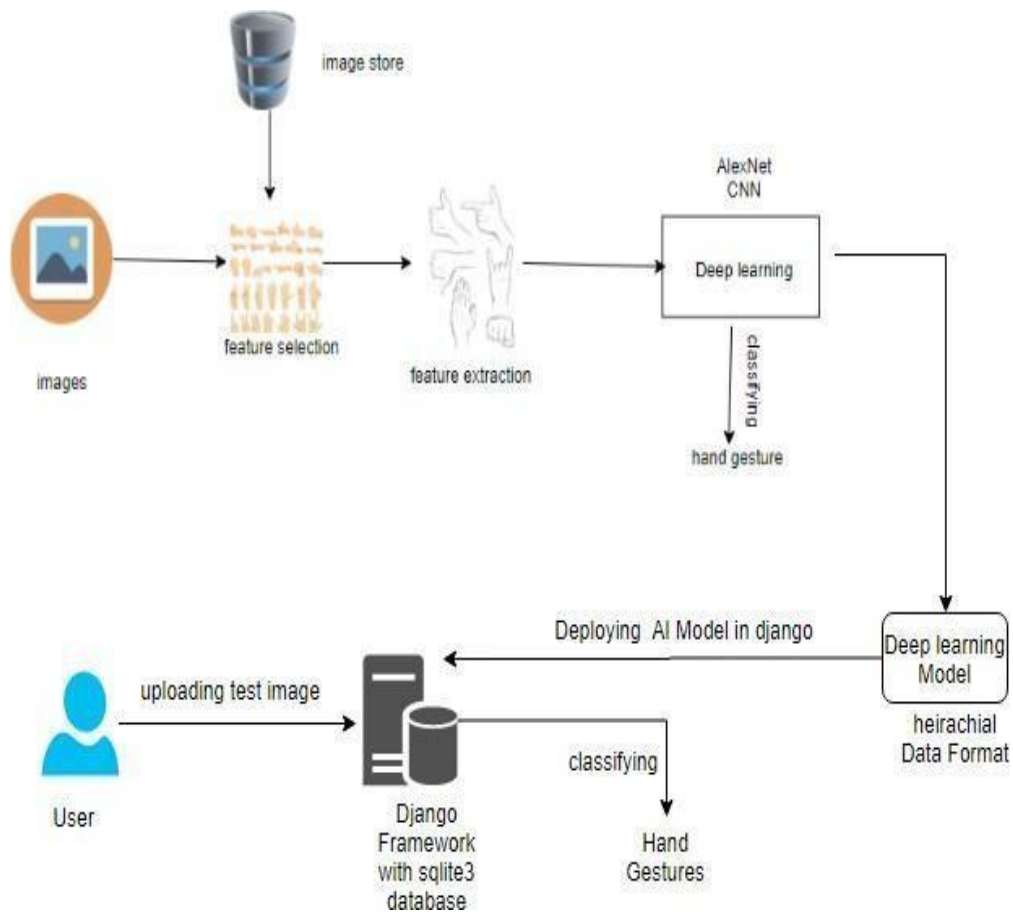


Fig 3.1: Architecture Diagram

3.2 EXISTING SYSTEM:

Understanding and classifying electromyogram (EMG) signals is of significance for dexterous prosthetic hand control, sign languages, grasp recognition, human-machine interaction, etc. The existing research of EMG-based hand gesture classification faces the challenges of unsatisfied classification accuracy, insufficient generalization ability, lack of training data and weak robustness. To address these problems, this paper combines unsupervised and supervised learning methods to

classify an EMG dataset consisting of 10 classes of hand gestures. To lessen the difficulty of classification, clustering methods including subtractive clustering and fuzzy c-means (FCM) clustering algorithms are employed first to obtain the initial partition of the inputs. The electromyogram (EMG) signals from human muscle are widely used to identify neuromuscular diseases. Apart from medical and diagnostic applications, the recognition of patterns of EMG signals generated by different movements has great values and potential in Biomechanics, Kinesiology, Mechanical bionics, etc. One of the most successful and significant applications is the EMG controlled prosthesis, which is beneficial for amputees and patients who have difficulties to make specific movements. Understanding and classifying EMG signals are also helpful for recognizing gestures, sign languages, human-machine interaction. They have at the first stage, a top classifier is used to classify the data into three clusters; at the second stage, sub-classifiers are trained accordingly to classify the data in each cluster. The proposed method achieves the strongest robustness compared to five widely used classifiers. They further The main contributions include the modification of conventional FCM algorithm to make it accustomed to multi-class classification, the design of twostep classification which classification accuracy and strong robustness, and the comprehensive comparison with five widely used machine learning models. Based on the obtained clusters, the classification procedure is divided into two steps. At the first stage, a top classifier is used to classify the data into three clusters; at the second stage, sub-classifiers are trained accordingly to classify the data in each cluster.

3.3 UML DIAGRAMS:

3.3.1 Use case diagram:

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

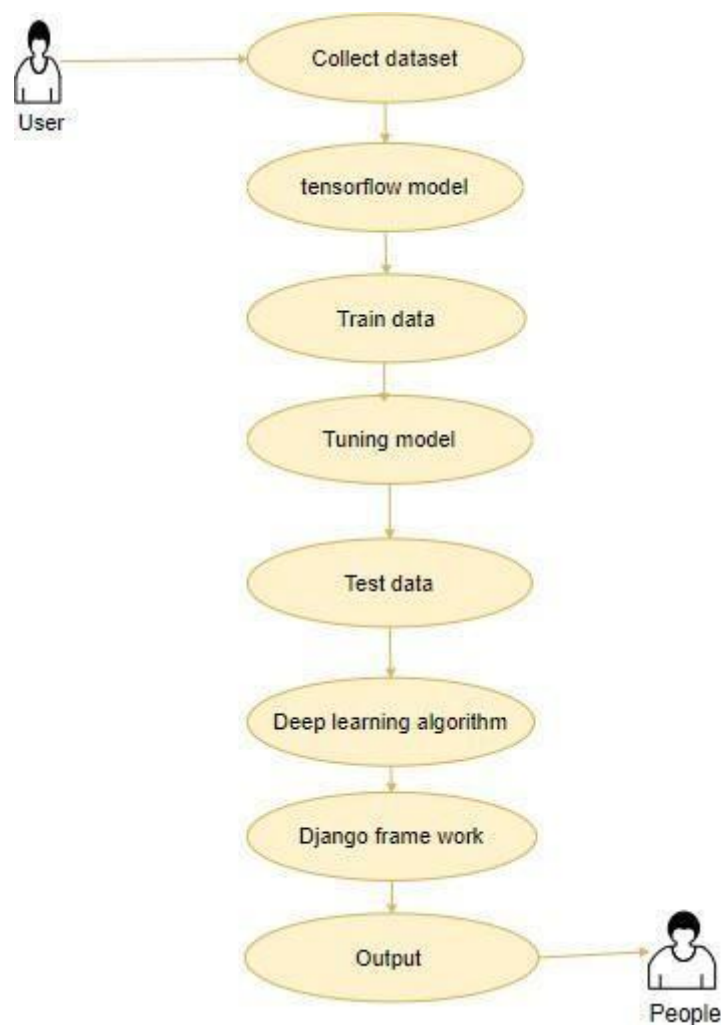


Fig 3.3.1: Use Case Diagram

3.3.2 Class Diagram:

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

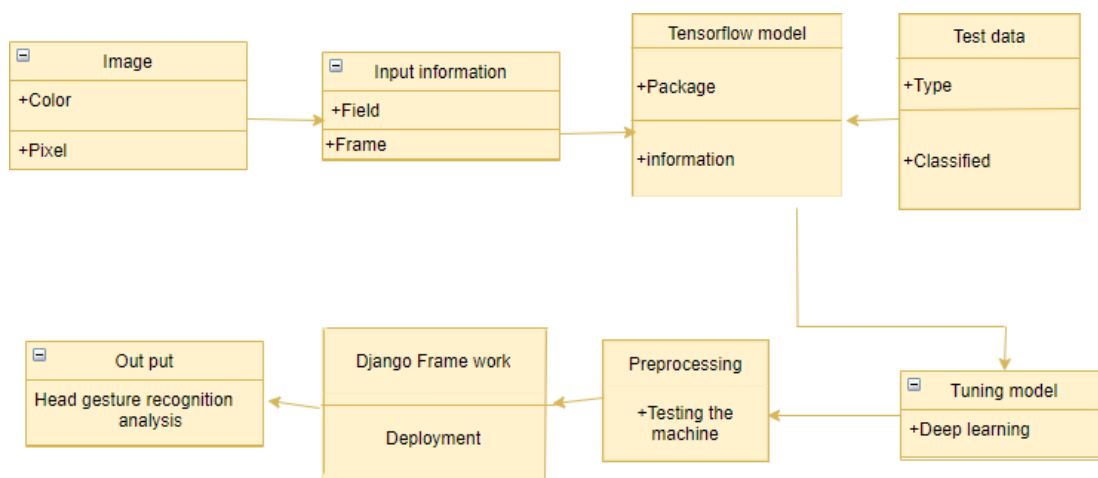


Fig 3.3.2: Class Diagram

3.3.3 Activity Diagram:

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

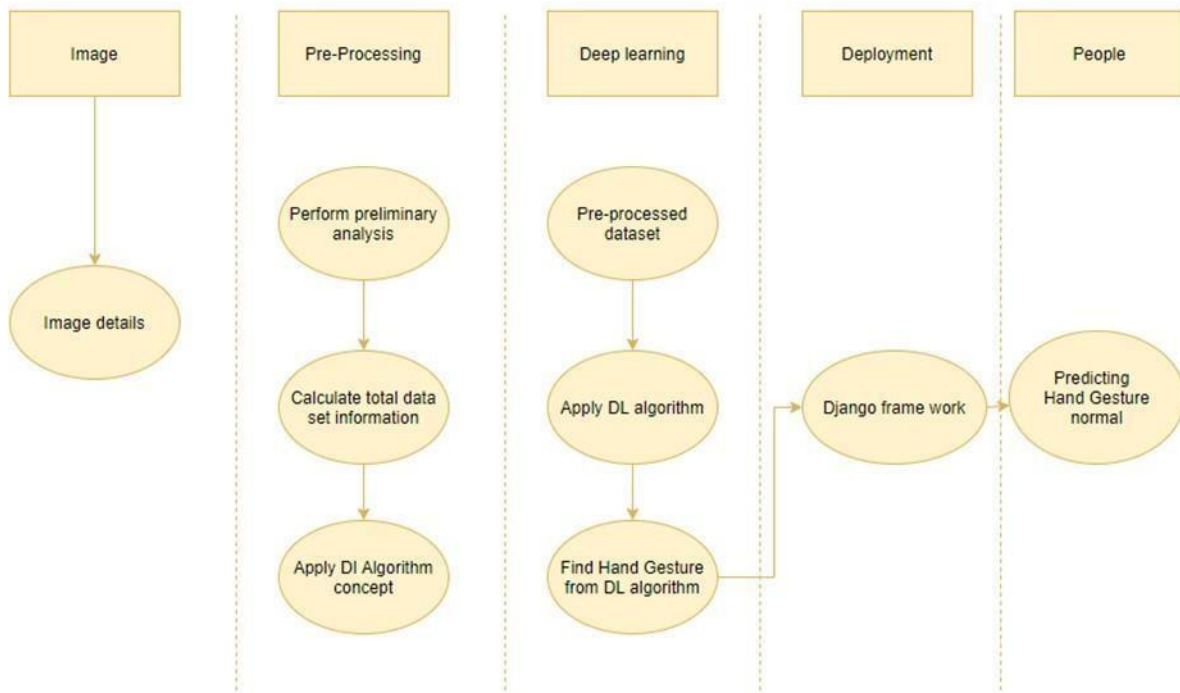


Fig 3.3.3: Activity Diagram

3.3.4 Sequence Diagram:

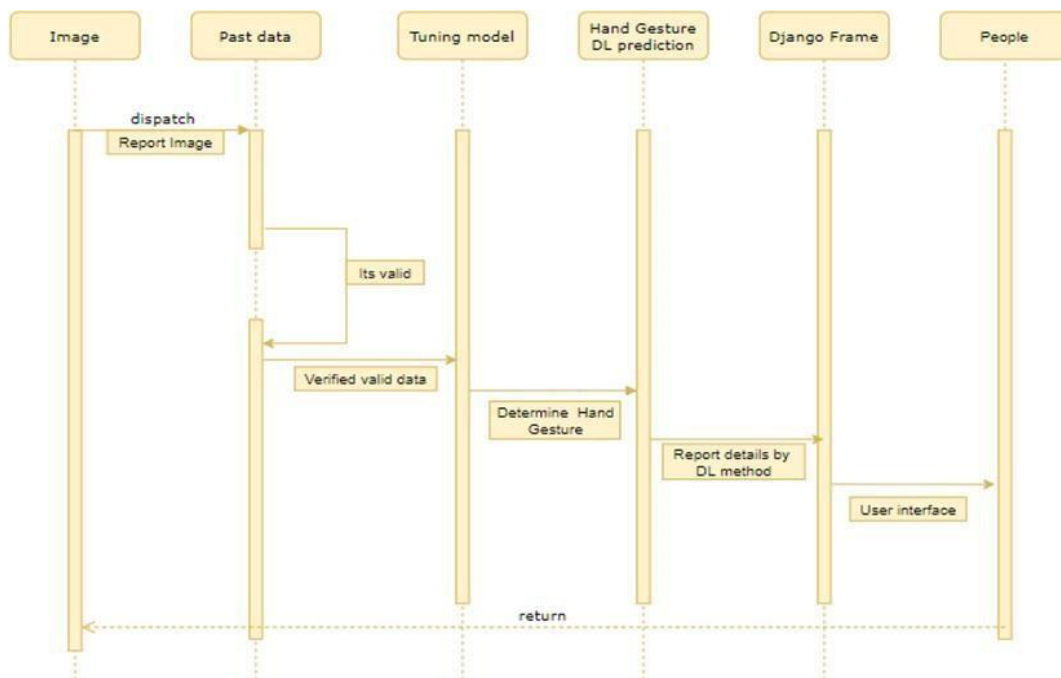


Fig 3.3.4: Sequence Diagram

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

3.3.5 Collaboration Diagram:

A collaboration diagram is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users (like doctor or patient) can benefit from this collaboration. A collaboration diagram often comes in the form of a visual chart that resembles a flow chart. It can show, at a glance, how a single piece of software complements other parts of a greater system.

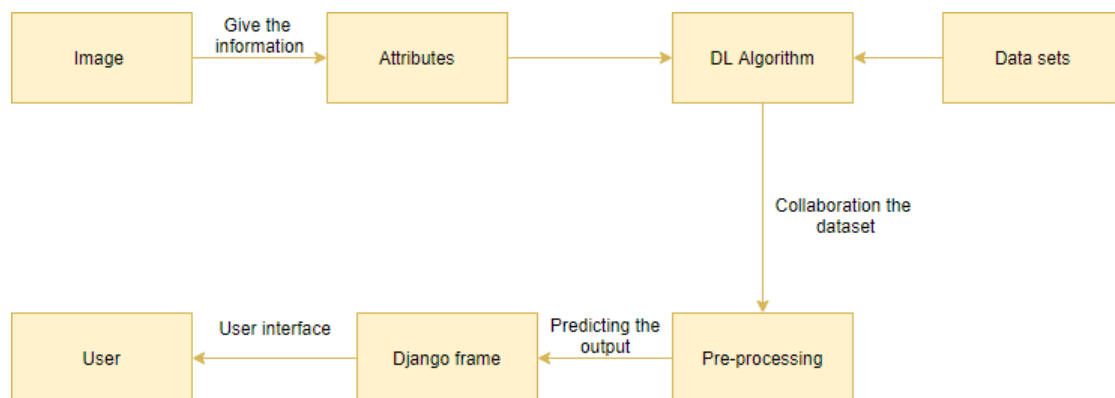


Fig 3.3.5: Collaboration Diagram

3.3.6 ER Diagram:

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modelling technique that can help define business processes and

be

used

as

the

foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

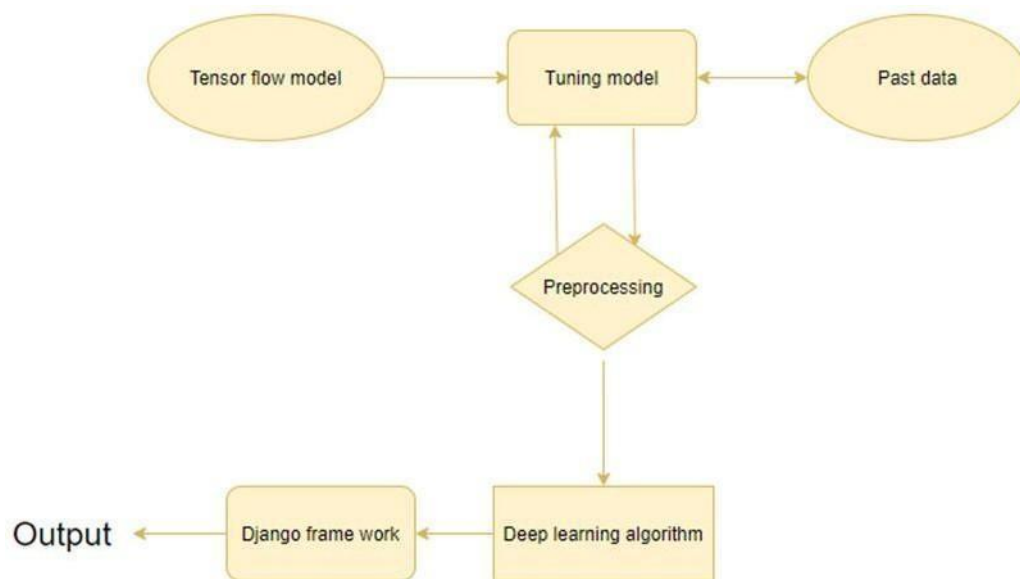


Fig 3.3.6: ER Diagram

3.4 DESCRIPTION:

The project will be structured into 3 distinct functional blocks, Data Processing, Training, Classify Gesture.

3.4.1 Data Processing:

The load data.py script contains functions to load the Raw Image Data and save the image data as NumPy arrays into file storage. The process data.py script will load the image data from data.npy and pre-process the image by resizing/rescaling the image, and applying filters and ZCA whitening to enhance features. During training the processed image data was split into training, validation, and testing data and written to storage. Training also involves a load dataset.py script that loads the relevant data split into a Dataset class. For use of the trained model in classifying gestures, an individual image is loaded and processed from the filesystem.

3.4.2 Training:

The training loop for the model is contained in `train model.py`. The model is trained with hyperparameters obtained from a config file that lists the learning rate, batch size, image filtering, and number of epochs. The configuration used to train the model is saved along with the model architecture for future evaluation and tweaking for improved results. Within the training loop, the training and validation datasets are loaded as Data loaders and the model is trained using Adam Optimizer with Cross Entropy Loss. The model is evaluated every epoch on the validation set and the model with best validation accuracy is saved to storage for further evaluation and use. Upon finishing training, the training and validation error and loss is saved to the disk, along with a plot of error and loss over training.

3.4.3 Classify Gesture:

After a model has been trained, it can be used to classify a new ASL gesture that is available as a file on the filesystem. The user inputs the file path of the gesture image and the `test data.py` script will pass the file path to `process data.py` to load and pre-process the file the same way as the model has been trained.

The primary source of data for this project was the compiled dataset of American Sign Language (ASL) called the ASL Alphabet from Kaggle. The dataset is comprised of 87,000 images which are 200x200 pixels. There are 29 total classes, each with 3000 images, 26 for the letters A-Z and 3 for space, delete and nothing.

A self-generated test set was created in order to investigate the neural network's ability to generalize. Five different test sets of images were taken with a webcam under different lighting conditions, backgrounds, and use of dominant/non-dominant hand. These images were then cropped and pre-processed.

The model used in this classification task is a fairly basic implementation of a Convolutional Neural Network (CNN). As the project requires classification of images, a CNN is the go-to architecture. The basis for our model design came from Using Deep Convolutional Networks for Gesture Recognition in American Sign Language paper that accomplished a similar ASL Gesture Classification task. This model consisted of convolutional blocks containing two 2D Convolutional Layers with ReLU activation, followed by Max Pooling and Dropout layers. These convolutional blocks are repeated

3 times and followed by Fully Connected layers that eventually classify into the required categories. The kernel sizes are maintained at 3 X 3 throughout the model. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage. A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions.

3.5 ALGORITHM:

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

- Deep learning requires large amounts of labelled data. Driverless car development requires millions of images and thousands of hours of video.
- Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks. The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150. Deep learning models are trained by using large sets of labelled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not retrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

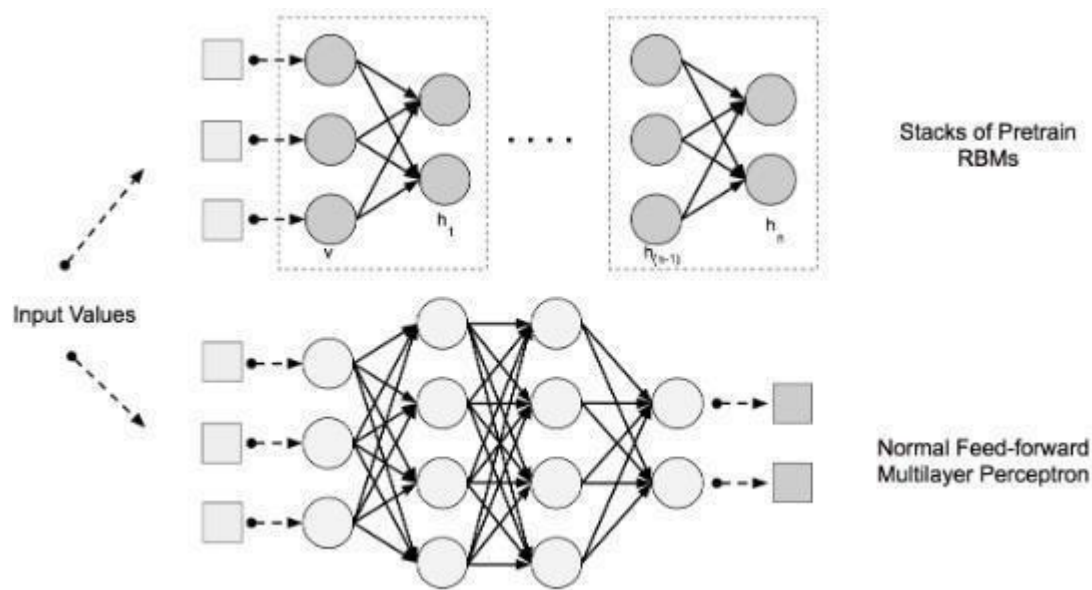


Fig 3.5: Deep Learning

3.6 CREATING A DEEP LEARNING MODEL:

To train a deep network from scratch, you gather a very large labelled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train.

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.

A slightly less common, more specialized approach to deep learning is to use the network as a feature extractor. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during

the training process. These features can then be used as input to a machine learning model such as support vector machines (SVM).

3.7 MODULE-1(MANUAL CNN):

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that scan the hidden layers and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters.

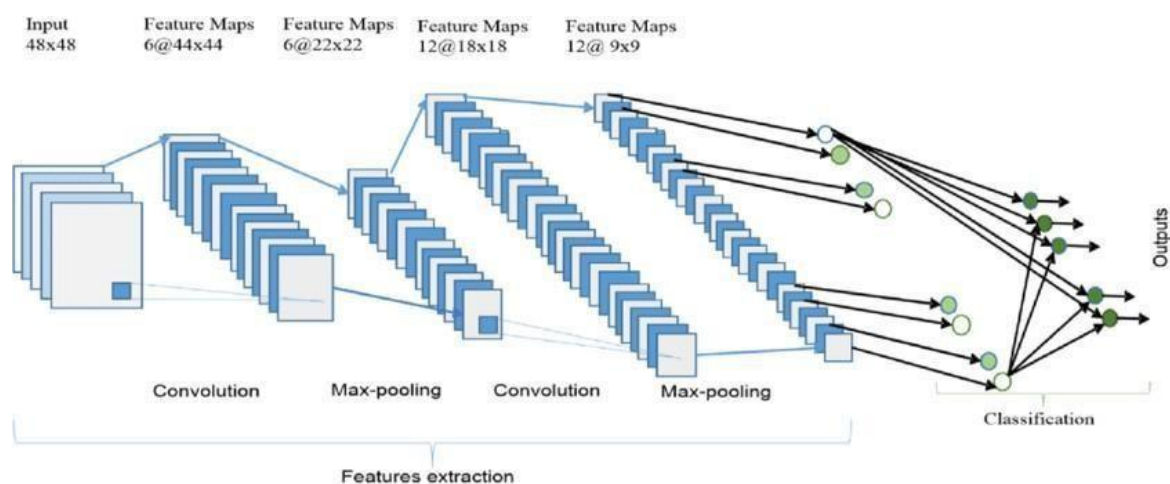


Fig 3.7.1: Convolution Neural Network

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive fields (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4×4 dimension input is reducing to 2×2 dimensions.

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

SoftMax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and SoftMax is for multi-classification.

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

Firstly, we are creating a manual convolution neural network. This manually created convolution neural network has no layer formats, unlike Alexnet and Lenet. Firstly, we are importing all the required packages and modules like OS, NumPy, Matplotlib, TensorFlow, Keras, and others. Now we are importing the dataset containing images for our model by locating our dataset folders in the directory. At

this

point, we are displaying the information of the images in the dataset for each hand gesture. The information displayed is image count, minimum width, maximum width, minimum height, and maximum height for every hand gesture. Now we are creating a sequential model with few parameters like convolution 2D, input shape, activation, dense, flatten, and max pooling. We use convolution 2D for filtering the pixels in images and Relu activation for better activation of neurons. To make our model understand the shape of the image which it should get and flatten for converting the data into a 1-dimensional array. And now we are displaying the classifier summary in table format. At this point, we are training our data for 50 epochs. Now we obtain our accuracy and loss percentage at the end of our training. Now, by using Matplotlib we are plotting a graph for accuracy and loss.

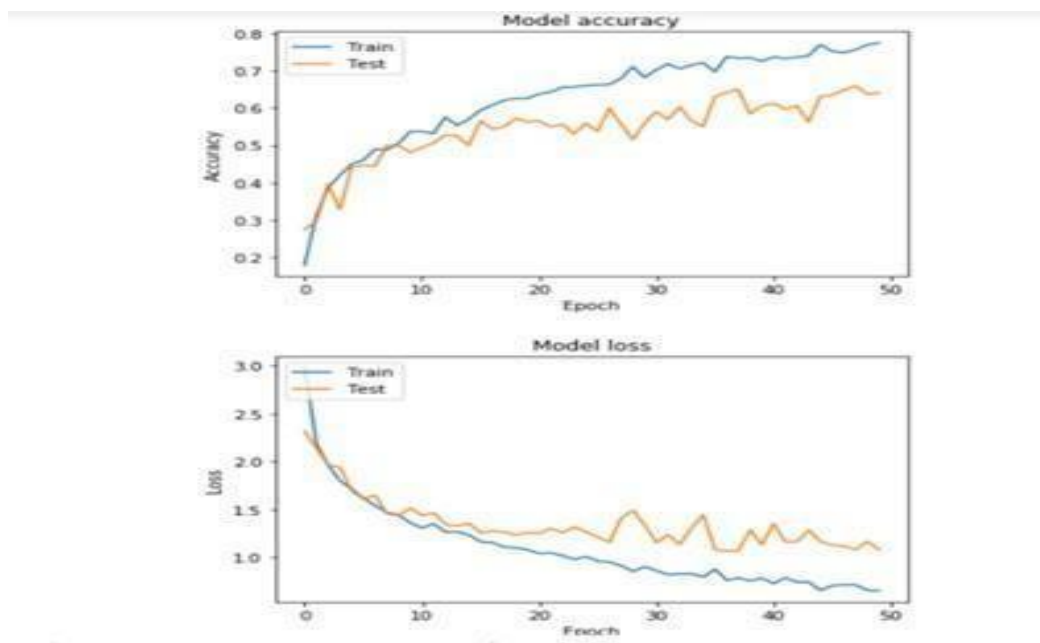


Fig 3.7.2 Manual CNN Accuracy & Loss Graph

3.8 MODULE-2(ALEXNET):

AlexNet is the name of a convolutional neural network (CNN), designed by Krizhevsky's Ph.D. advisor. AlexNet competed in the ImageNet Large Scale Visual Recognition Challenge on September 30, 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The

original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of graphics processing units (GPUs) during training.

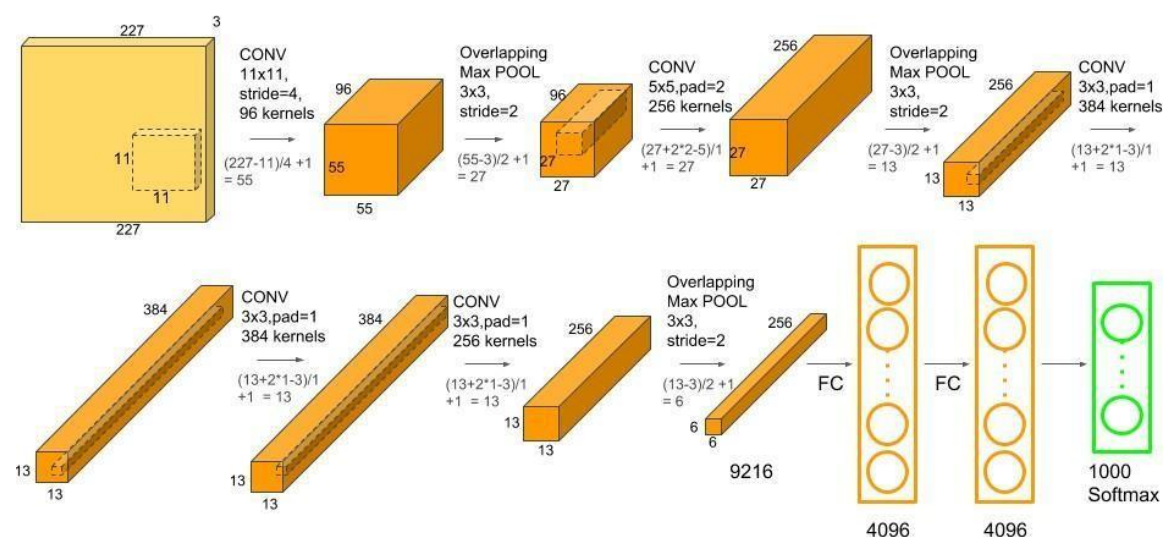


Fig 3.8.1: AlexNet Architecture

Unlike the manually created Convolution neural network, AlexNet has layer formats. Now we are going to implement our model in AlexNet. Implementing the model in Alexnet involves 11 layers. The first layer is Convolutional layer 1, where the filter is 96 and kernel size is (11 x 11). The second layer is the max-pooling layer where pool size is (3,3). The third layer is convolution layer 2, where the filter is 256. The fourth layer is again a max-pooling layer. The fifth layer is convolution layer 3, where the filter is 384. The Sixth layer is the convolution layer 4, where the filter is 384. The seventh layer is convolution layer 5, where the filter is 256. The eighth layer is again a max-pooling layer with (3 x 3). After this point, we are using flatten to convert it into a 1-dimensional array and pass it to a fully connected layer. The ninth layer is a fully connected layer 1 with a dense value 4096. Now, we are making a dropout of (0.4) to prevent the issues of overfitting. The tenth layer is a fully connected layer 2 with a dense value of 4096. And the final layer is fully connected layer 3 with a dense value of 1000. Finally, our output layer will with SoftMax activation and a dense value of 16. Now we are training our dataset for 30 epochs. Now, we obtain our accuracy and loss graph for our AlexNet CNN. Here we obtain an accuracy rate of 67.42%.

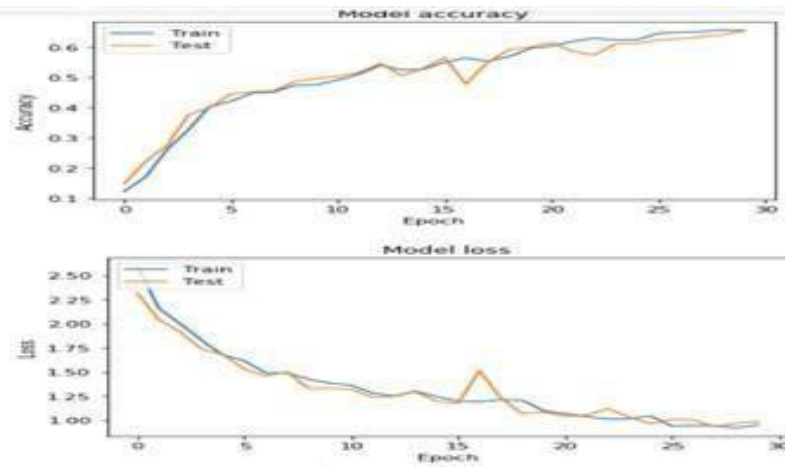


Fig 3.8.2: AlexNet Accuracy & Loss Graph

3.9 MODULE-3(LENET):

LeNet is a convolutional neural network structure proposed by Yann LeCun et al. in 1989. In general, LeNet refers to lenet-5 and is a simple convolutional neural network. Convolutional neural networks are a kind of feed-forward neural network whose artificial neurons can respond to a part of the surrounding cells in the coverage range and perform well in large-scale image processing.

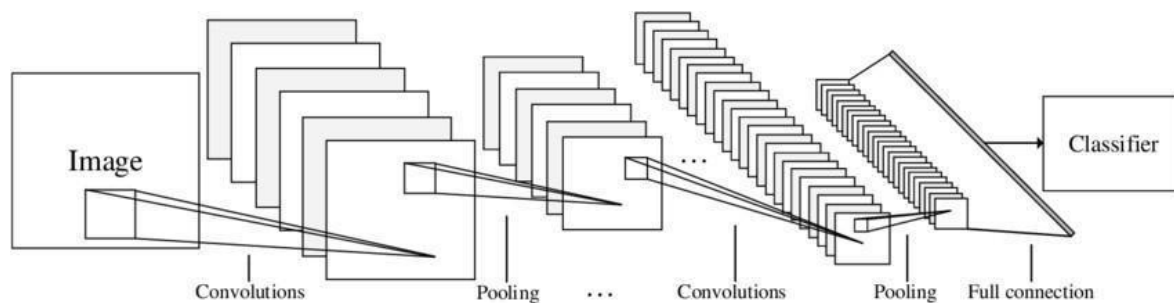


Fig 3.9.1: LeNet Architecture

Here in LeNet, we implement our model in 7 layers. The first layer is convolutional layer 1 with convolution 2D where we give filter with 6, kernel size with 5, Relu activation, and input shape of (225,225,3). The second layer is max-pooling layer 1 with a pool size of 2. The third layer is convolution layer 2 with convolution 2D where we give filter with 16, kernel size with 5, Relu activation, and input shape with (225,225,6). The fourth layer is max-pooling layer 2 with a pool size of 2. Now, we apply flatten to convert it to single dimension array. The fifth layer is fully connected layer 1 with a dense of 120 units and Relu activation. The sixth layer is fully connected layer 2 with a dense of 84 units and Relu activation. And the last layer is

the

output

layer with a dense of 16 units and softmax activation. At this point, we display the classifier summary. Now, we train the dataset for 30 epochs and produce the accuracy and loss graph. Here we obtain an accuracy rate of 90.72%.

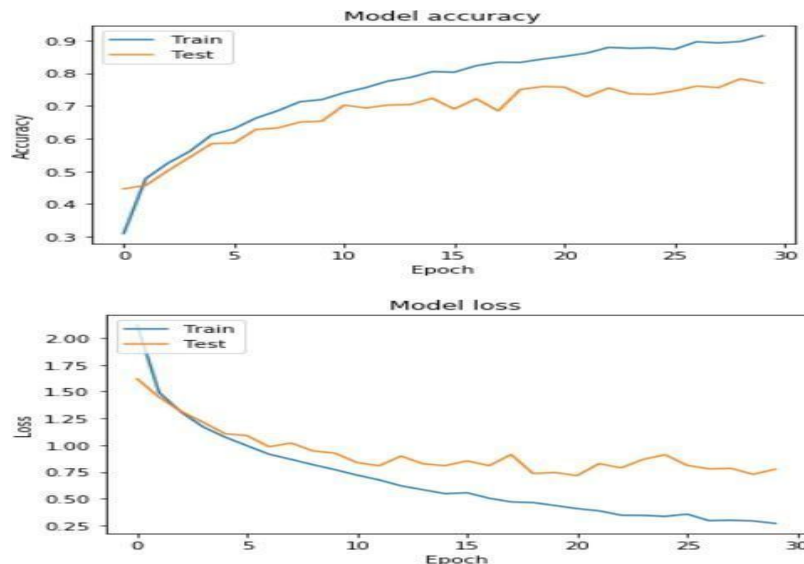


Fig 3.9.2: LeNet

3.10 DJANGO FRAMEWORK:

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005.

Django has continued to grow and improve, from its first milestone release (1.0) in September 2008 through to the recently-released version 3.1 (2020). Each release has added new functionality and bug fixes, ranging from support for new types of

databases, template engines, and caching, through to the addition of "generic" view functions and classes (which reduce the amount of code that developers have to write for a number of programming tasks).



Fig 3.10: Django Framework

At this point, we have implemented our model in a manually created convolution neural network, AlexNet CNN and LeNet CNN. Now, we compare the accuracy percentage of the model in all the convolution neural networks. As we observe, the accuracy percentage in LeNet is higher. So, the trained deep learning model is converted into a hierarchical data format file (.h5 file) which is then deployed in our Django framework for providing a better user interface and predicting the output which type of gesture is it.

3.11 REQUIREMENT SPECIFICATION:

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the customer's machines already. So, the main need is to introduce appropriate equipment for the product. The requirements are split into two categories, namely:

Software Requirements:

The basic software requirements to run the program are:

- i. Microsoft Windows XP, Windows 7, Windows 10
- ii. Anaconda Navigator
- iii. PyCharm
- iv. Browser e.g. Mozilla Firefox, Safari, chrome, etc.

Hardware Requirements:

The basic hardware required to run the program are:

- i. Hard disk of 40 GB
- ii. System memory (RAM) of 4 GB
- iii. I3 processor-based computer or higher.

CHAPTER 4

RESULTS AND DISCUSSIONS

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

The Jupyter Notebook is a living online notebook, letting faculty and students weave together computational information (code, data, statistics) with narrative, multimedia, and graphs. Faculty can use it to set up interactive textbooks, full of explanations and examples which students can test out right from their browsers. Students can use it to explain their reasoning, show their work, and draw connections between their classwork and the world outside. Scientists, journalists, and researchers can use it to open up their data, share the stories behind their computations, and enable future collaboration and innovation.

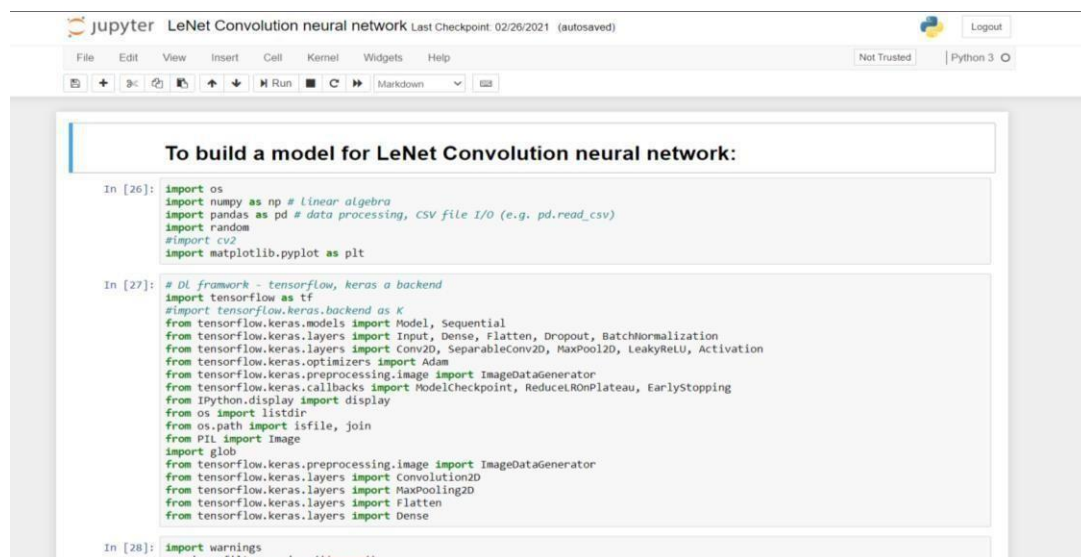


Fig 4.1: Jupyter Notebook

Jupyter notebook is the platform where we have developed and trained our model.

Here, is the screenshot of the LeNet model, which we have trained.

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda. PyCharm is cross-platform, with Windows, macOS and Linux versions.

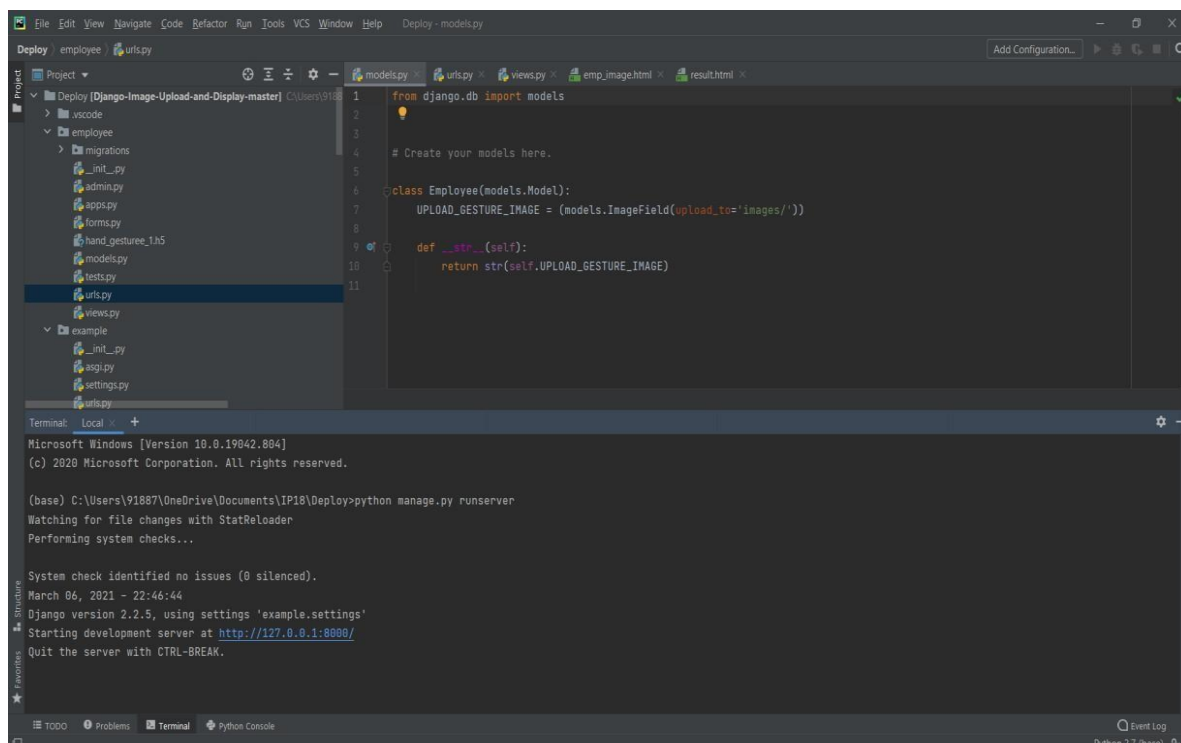


Fig 4.2: PyCharm Screen

As, we have implemented our LeNet Convolution neural network model in Django framework, we have used PyCharm to run the module. Here we locate your model. We type the command which is “python manage.py runserver” in the terminal window. This command you'll probably run the most of all commands. It means to run a emulated server on your local computer. So, after running it, you can go to [localhost:8000] (http://localhost:8000) or 127.0.0.1:8000.

Requirements: We must run this in the ROOT of our django project where manage.py lives. This, gives the local host link, where we can find our implemented model in a web application.

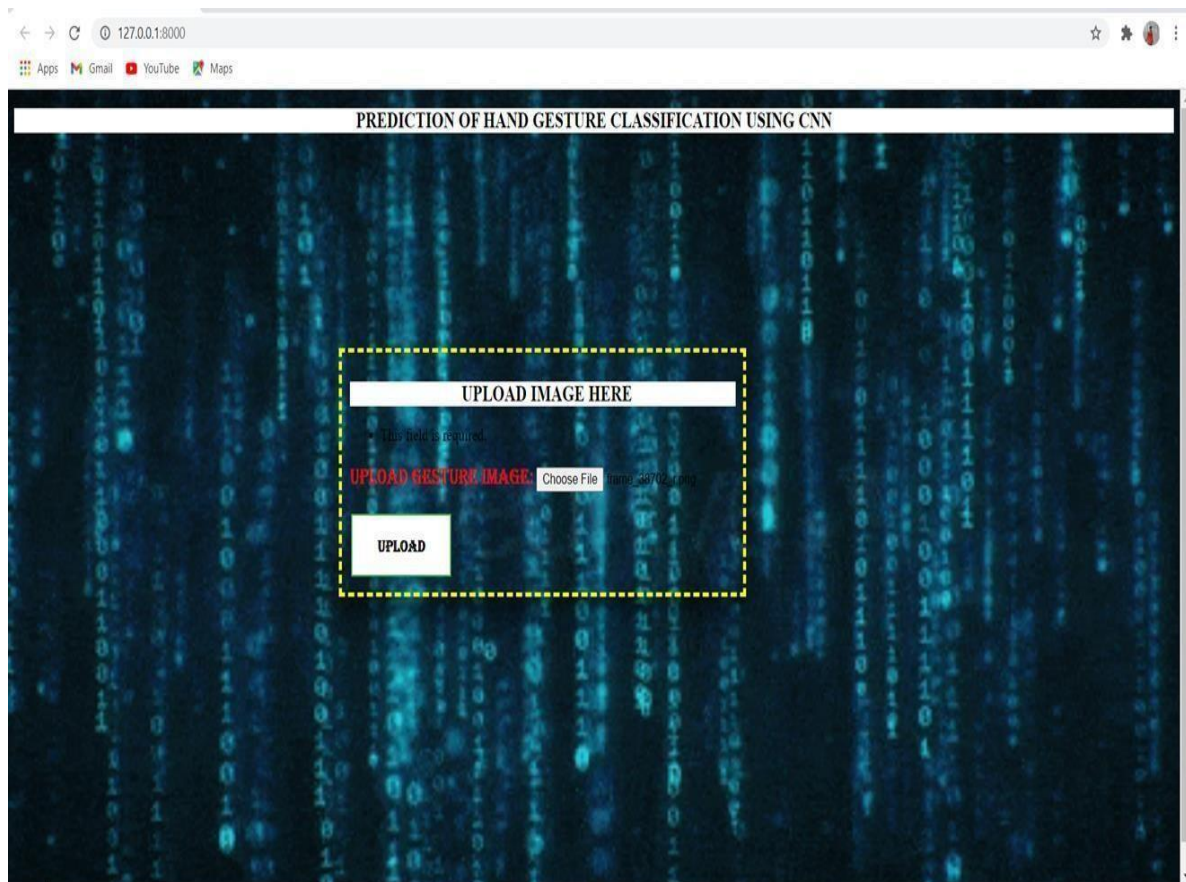


Fig 4.3: Home Screen

After clicking the link in the pycharm screen, the local host website opens. Here the user interface runs on a local host, which is created using Django framework. Here HTML and CSS is used as frontend and SQLite3 is used as backend. This user interface consists of a home page which displays the title as “PREDICTION OF HAND GESTURE CLASSIFICATION USING CNN”. Background image is used here to make the webpage more attractive. A small highlighted box is displayed with the title as “UPLOAD IMAGE HERE”.

Inside that box, we are displaying a message to the user like “UPLOAD GESTURE HERE” followed by a button. Users can click that button and browse for the gesture image in their systems.

There is a button which says “UPLOAD”. Users, after selecting the gesture image for which they are expecting the recognition, they can click the upload button. This upload

button accepts the image and consider that image as input and passes the image to the next output screen.

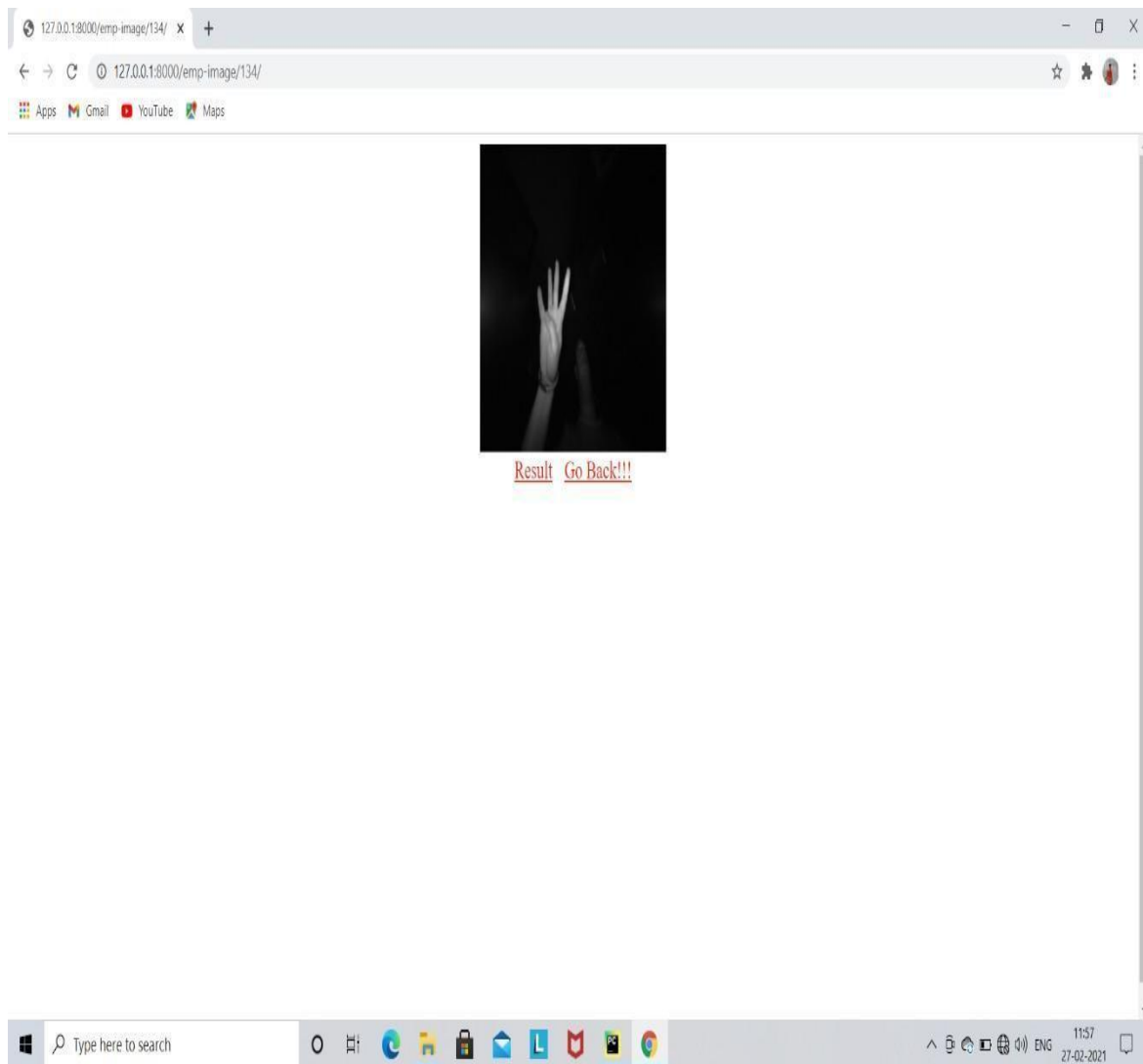


Fig 4.4 Intermediate output screen

This is the intermediate output screen between the home screen and the final output screen. After clicking the upload button in the home screen, the user gets redirected to this intermediate output page. Here in this intermediate output page the uploaded image is displayed.

Here users can confirm their uploaded image is correct or incorrect. Here, two buttons are placed for the users. The first button is result button. If users feel the displayed image is the correct image they hit the result button. This result button redirects the user to the final output screen.

If user feels the displayed image is not correct, they can hit the GoBack button. This goback button redirects the user to the home screen, where user can reselect the correct image for the prediction.

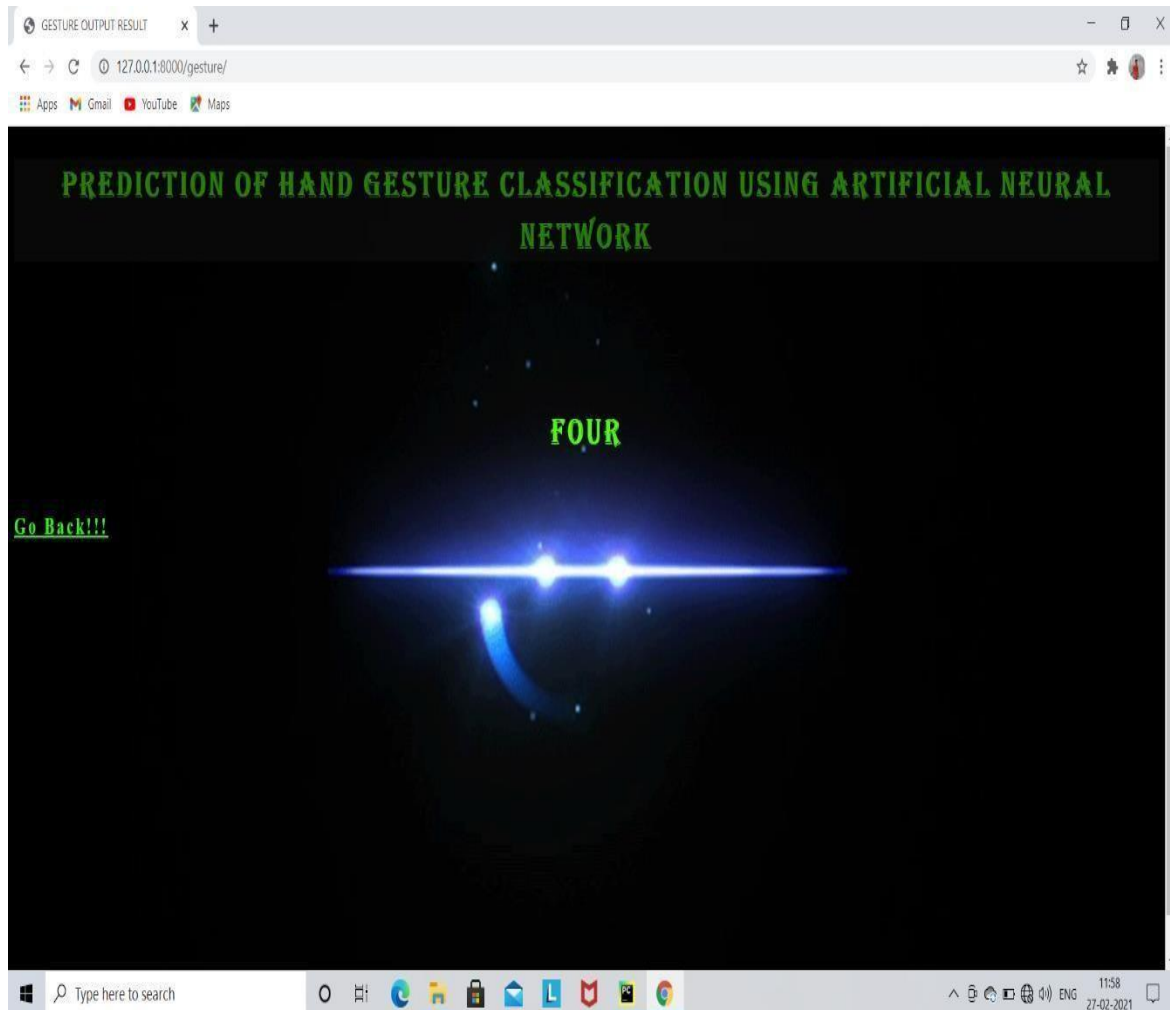


Fig 4.5: Final Output Screen

This is the final output screen. This is the final output webpage. This page displays the title as “PREDICTION OF HAND GESTURE CLASSIFICATION USING ARTIFICIAL NEURAL NETWORK”. Background image is used to make the webpage look more attractive. Here the predicted result is displayed in the text format. The LeNet CNN predicts the result from the uploaded image.

The predicted result in the text format is the final output. And also Go Back button is provided for the user to go back to the home. There user can upload the next image for the prediction.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Hand sign language interpreter addresses the fault in communication among people with disabilities. This proposed system resolves the common issues among deaf people. The trained model can process the continuous stream of inputs and predict results with more accuracy. The trained accuracy of the model is approximately 78%, whereas the prediction of the result is 90.72%. There are various techniques and methods used among the hand gesture projects. Each technique has their advantages and drawbacks, however, the methodology used in this project suits and perform well in most challenging cases. Since the model has better accuracy in both trained datasets and prediction, it is more suitable for real-time applications.

The problem of Sign Language (SL) recognition using images is still a challenge. Similarity of gestures, user's accent, context and signs with multiple meanings lead to ambiguity. These are some reasons why previous work used limited datasets. In this manuscript, I used Neural Networks (NN) to conduct a preliminary study about SL recognition using depth images. This study provided information about the constraints of the problem. I concluded that a dataset to train and evaluate the system must have sufficient gesture variations to generalize each symbol. I extended the average method to determine if a dataset has enough gesture variations and I implemented a Convolutional Neural Network (CNN) for recognizing hand gestures in images of American Sign Language letter and number symbols. The average method was based on, but in my case the focus was hand images and I parallelized the method using the GPU to increase the efficiency. Most of the SL datasets are color images or depth images with no preprocessing. These images had background and users' bodies. Considering there was not an "ideal" American Sign Language dataset, I gathered images to create a new dataset. SL can have a lot of variations for each gesture; accent of a person, signing speed, and the next symbol determines gesture shape. Therefore, a dataset has to have enough variations of each gesture to capture most of the possible shapes a gesture can have. Since the method creates a representation of each gesture, if it acquires high

recognition rate, the dataset has similar images, which is not good to generalize the problem of SL recognition. If the method cannot

recognize gestures, the dataset will have enough variance to provide information about diverse shapes of the gestures. Compared to the average method, the CNN achieved a higher recognition rate. Because American Sign Language has very similar symbols (m and n for example) for some of the letters and gesture shape variation in the dataset is high, the system had problems to classify correctly some of the images. Some of the images looked similar and the CNN labeled them incorrectly. Instead of taking the best classification (top-1), if in the top-3 possible labels was the correct one, it meant the system was close to recognizing the image, but due to similar symbols the CNN misclassified it. Looking further than top-5, the CNN did not provide meaningful information, suggesting top-5 classifications were enough to achieve high recognition rate. The collected dataset of American Sign Language letters and symbols is going to be shared with the scientific community. It should be extended to more fluent and proficient users because the aim of this work is for deaf persons who are fluent users. However, collected non-fluent user images was greater than the number of fluent users, and thus, images may not fulfill the requirements of those users. Due to the lack of dexterity in hands of non-fluent users, the captured images may not fit gesture shapes of fluent users and thus the created system would not achieve high accuracy recognition rate for deaf people.

At this moment, results showed high accuracy on the captured dataset with symbols for letters and numbers, but the CNN has been designed based on my own experience with some tests and tutorials I did before and following the architecture presented. It may be that adding more symbol classes to recognize, the model precision decreases and thus the system will not be as good as before. My idea to improve system's design is to start with a group of gestures and test diverse configurations of the CNN to observe which configurations are the best for that case. The group of gestures will be increased with additional classes and, using the evaluation methods, identify the best configuration. From the relation number of classes and best configurations, my goal is to find a relation between input number of classes and design of the CNN so the system can construct an effective model for each case. Once the letter recognition system has achieved high recognition rate, a logical step towards the SL transcription is the concatenation of predicted letters to construct words. Since the CNN for letter recognition is not perfectly precise, errors exist at the recognition step, consequently, errors can accumulate over the

transcription 70 process. My current knowledge about language processing is not high enough to define exact features of this future work but one idea is to use Hidden Markov Models for the creation of finger-spelled words and correct errors caused by the CNN. SL has symbols to represent words instead of finger-spelling them, therefore if a word gesture is detected the word creation step is skipped. Recognized words will then be used to construct sentences in the final step of the transcription process. SL has no direct translation to natural language due to the elusion of some of the words; as an example, stop words or prepositions. Techniques used in query suggestion systems may be useful to ensure correctness of sentence forming, but they have to be trained in already transcribed SL texts. As stated, the proposed system was trained only using static images; such a case excludes some of the letters in American Sign Language alphabet (J and Z). The inclusion of these letters and gestures done by movement is crucial for proper transcription, thus methods to recognize gestures will be required in the future. I mentioned the significance of face expressions. I would like to add this feature to the system because it may be helpful for understanding what the users are saying, and thus produce more precise transcriptions. The use of another CNN should be useful to detect face expressions, that is one area I would like to study to improve the transcription system.

To be more accurate on future works:

- To deployment real time this process by show the prediction results in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.
- To deploy this model to AI on web application.

REFERENCES

- [1] Nasser H. Dardas and Nicolas D. Georganas, "Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques," *IEEE Transaction on Instrumentation and Measurement*, volume 60, no. 11, pp.132- 137 November 2011.
- [2] El-Sawah, et.al "A prototype for 3-D hand tracking and gesture estimation," *IEEE Transaction Instrumentation Measurement*. volume 57, no. 8, pp. 1627-1636, August 2008.
- [3] Priyanka Mekala, et.al, Asad Davari "Real-time Sign Language Recognition based on Neural Network Architecture" 978-1-4244-9592- 4/11/ ©2011IEEE.
- [4] S. Pandita, and S. P. Narote "Hand Gesture Recognition using SIFT" *International Journal of Engineering Research and Technology (IJERT)* Volume 2, no.1, pp1760- 1764, January 2013.
- [5] Usha Gopalakrishnan, Vinod P. R, et.al "Artificial Neural Network Based Method for Indian Sign Language Recognition" *IEEE Conference on Information and Communication Technologies (ICT 2013)*.
- [6] D. G. Lowe, "Distinctive image features from scale-invariant key points," *International Journal of Computer Science*, volume 60, no. 2, pp. 91- 110, November 2004.
- [7] J. Singha and K. Das, "Indian Sign Language Recognition Using Eigen Value Weighted Euclidean Distance Based Classification Technique", *(IJACSA) International Journal of Advanced Computer Science and Applications*, Volume 4, no. 2 , pp. 188-195, July 2013.
- [8] Y. Wu, J. Lin, and et.al, "Capturing Natural Hand Articulation". In *IEEE International Conference on Computer Vision II*, pp.426–432, September 2001.
- [9] H. Zhou and T. S. Huang, "Tracking articulated hand motion with Eigen dynamics analysis," in *Processing Of International Conference on Computer Vision*, volume 2, pp. 1102–1109, November 2003.
- [10] T. Sterner and A. Pent Land, "Real-time American sign language recognition from video using hidden markov models", *Technical Report, M.I.T Media Laboratory Perceptual Computing Section*, Technical Report No. 375, 1995.
- [11] L. Bretzner, et.al "Hand gesture recognition uses multi-scale color features, hierarchical models and particle ltering", in *Processing of International Conference. Au-tom. hand Gesture Recognition*, Washington, DC, May 2002.

[12] A. Argyros and M. Lourakis, "Vision-based interpretation of hand gestures for remote control of a computer mouse", in *Processing of Workshop Computer Human Interaction*, pp.4051, July 2006.

[13] Qing Chen, N. D. Georganas, "Hand Gesture Recognition Using Haar- Like Features and a Stochastic Context-Free Grammar", *IEEE transactions on instrumentation and measurement*, volume 57, no. 8, pp.113-117, August 2008.

[14] G. R. S. Murthy and R. S. Jadon, "Hand Gesture Recognition using Neural Networks", *IEEE Transactions*, Volume 6, pp.427-431, May 2010.

[15] N.A. Ibrahim and R.Z. Khan, "Survey on Various Gesture Recognition Technologies and Techniques", *International Journal of Computer Applications*, Volume 50, No. 7, pp1230-1235, May2014.

[16] G. Deepa, et.al 'Emotion based Music Retrieval for Gesture Recognition' *International Journal of Computer Applications*, Volume 78, no.14, pp.0975 – 8887, September 2013.

APPENDIX

A. PLAGIARISM REPORT:

testing report			
ORIGINALITY REPORT			
7%	1%	6%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	"Intelligent Computing and Information and Communication", Springer Science and Business Media LLC, 2018		1%
	Publication		
2	Yuxiang Kuang, Qun Wu, Ying Wang, Nilanjan Dey, Fuqian Shi, Rubén González Crespo, R. Simon Sherratt. "Simplified inverse filter tracked affective acoustic signals classification incorporating deep convolutional neural networks", Applied Soft Computing, 2020		1%
	Publication		
3	Submitted to University of Southern California		1%
	Student Paper		
4	"Complex, Intelligent, and Software Intensive Systems", Springer Science and Business Media LLC, 2019		1%
	Publication		
5	Submitted to SASTRA University		1%
	Student Paper		
Submitted to Netaji Subhas Institute of			

6	Technology	1%
	Student Paper	
7	towardsdatascience.com	<1%
	Internet Source	
8	Bhavya Chawla, Shikhar Tyagi, Rupav Jain, Archit Talegaonkar, Smriti Srivastava. "Chapter 7 Finger Vein Recognition Using Deep Learning", Springer Science and Business Media LLC, 2021	<1%
	Publication	
9	www.freepatentsonline.com	<1%
	Internet Source	
10	"Computer Vision", Springer Science and Business Media LLC, 2015	<1%
	Publication	
Exclude quotes Off		
Exclude matches Off		
Exclude bibliography Off		

B. JOURNAL PAPER

Hand Sign Language Interpreter

Sanjay.S Computer
Science and
Engineering Sathyabama
Institute of Science and
Technology Chennai-119
sanju.aswm@gmail.com

Sachin Prassana.A
Computer Science and
Engineering Sathyabama
Institute of Science and
Technology Chennai-119
sachinprassana1999@gmail.co
m

Dr.T.Prem Jacob
M.E.,Ph.D. Computer
Science and
Engineering
Sathyabama
Institute of
Science and
Technology
Chennai-119

ABSTARCT:

Hand gestures are those gestures that are very expressive in nature. Hand gestures are the non- verbal method of communication. These hand gestures involve different movements or positioning of fingers which convey a message. These hand gestures may represent any short messages like numbers, greetings, or any other message. Here we are going to build and train a machine learning model which classifies images of various gestures. Here we implement this model in many types of convolution neural networks (CNN) and obtain an accuracy graph. We choose the best convolution neural network with the best results and deploy that model in the Django framework by converting it to a (.h5) file. Hence the output will be in a web application.

1. INTRODUCTION:

Hand sign language is commonly used by deaf people and hearing loss people for communication. The hand gesture and posture are used to communicate, here the image is processed using Convolutional Neural Network (CNN) algorithm. Convolutional Neural Network is a class of deep learning, this algorithm is used to analyse the images. The image is analysed by various processes in CNN such as pooling, convolution, etc. Deep learning is a function that processes data and creates a pattern to decide in real-time. The hand posture image is processed and analysed, then based on the trained models the system predicts the result. The sign language used here is American sign language, since this sign language is popular and widely used. The main difference between gesture and posture is, the shape is considered in the posture and whereas the movement is considered in the gesture. one of the hand gesture applications such as gesture-based interface used television which can eliminate the use of remote in the future. The model is trained using Kaggle datasets using CNN. AlexNet is used to predict and train the dataset. AlexNet is the name of the convolution neural network which is widely used in this deep learning field. LeNet is also the name of the Convolution neural network this is used to compare the prediction percentage among this two

CNN to get a better prediction rate. Django framework is used as a user interface in this project to provide input and the result is displayed. Once the user provided the image, the content required for prediction is extracted from it using the deep learning algorithm and then the predicted output is displayed in the form of a word that can be easily recognized by people. There are many deaf people in the world lost their job due to discrimination. Some deaf people cannot understand sign language. An enormous amount of deaf people does not get their education in sign language. So, this sign language interpreter act as a medium for deaf people. However, written communication is practically not possible in real-time, the hand sign language interpreter is the alternate solution to this condition. The main idea of this project is to help and save time for those people with disabilities. This developed model can help in resolve the condition of unemployment among deaf and hearing loss people. 70% of families are not able to communicate with their deaf children. This situation can be solved by this model. If the deaf people's hand sign is predicted by the system, then output in the form of a word that is easily understood by the people. In the

future, this application can also implement on video games, television, etc. to control. Sign language can be well developed and can be widely used among all

disabled people and this communication issue can be resolved.

2. LITERATURE SURVEY:

[1] In this paper Muneer Al-Hammadi, Wadood Abdul explores the utilize of 3DCNN for hand motion acknowledgment. Straight inspecting was used in the pre-processing stage to standardise the earthly calculation of hand signal tests. They have used size of the known face and human body segment proportions to stabilize dimensional measurement. At that point, we utilized 3DCNN for highlight learning in two approaches. Three instances of the 3-Dimensional CNN framework were prepared in the moment approach to extract the hand motion highlights from the video test's beginning, middle, and end. These area-based highlights had been entangled for some time and were being fed to the classifier at the time. MLP, LSTM, and an autoencoder were utilized for highlight combination. For sorting, they had used SoftMax tunable layer in both techniques. Several datasets were used to evaluate the proposed methods. In each of these signer-dependent as well as signer-independent modules, the three datasets performed admirably.

[2] In this article by Yong Wang, Aihu Ren a persistent hand signal detection and acknowledgment strategy were proposed. To begin, they gathered the actual data from the radar to obtain the If signals, which they use to test the ATM, DTM, and RTM of hand motions. They developed an amplitude-based discovery approach at that time, which linked the sufficiency of the stabilized hand motion spectrogram to a threshold. Finally, the k-means method was used to bundle the ATM, DTM, and RTM's centre time-frequency directions, and FDTW estimation was proposed to identify the hand signals. The proposed hand motion discovery strategy's division precision could hit 96.17 percent, and the planned FDTW calculation's normal acknowledgment exactness for 6 types of gestures was improved more by 5% while sparing much of the program lifecycle phase, according to test results.

[3] In this segment, Zhengjie Wang, Yushan Hou balanced the difference between the two sorts of hand signal ponders and highlight the properties of their works. They centralize on the likeness and distinction of these works and show the examination of the framework execution from the taking after perspectives counting signal sorts, feature presentation, exploratory situations, and acknowledgment exactness. The discoveries encourage the improvement of potential works and give a few experiences into this particular behavior acknowledgment. The energetic hand motion acknowledgment can be treated as a classification

issue since we ordinarily characterize some particular hand motions and classify the obscure motion into these predetermined names. Unexpectedly, hand tracking ordinarily comprises localization and tracking. The previous one finds the position of the hand whereas the last mentioned persistently decides the point position of the hand in 2D or 3D space. Hence, hand tracking is more complex since it must find the position of the hand more precise to identify a few letters or shapes. They have isolated the discourse into three segments counting signal procurement, signal handling, and performance assessment. After showing the common highlights of energetic motion acknowledgment and hand tracking, we outline the key features of them and make a comprehensive investigation of these works based on smartphone ultrasonic signals.

[4] In this paper Sidharth Pancholi, Amit M. Joshi presented a minimum-energy implanted framework that interprets surface electromyography (EMG) to detect arm movements. For the EMG design acknowledgment study, each subject completed six exercises. 2 test cases for real-time classification of information inside a DSP processor have been developed. The first case (extraction of features in a DSP processor and checking off-line) has the highest classification exactness 97.60% and also the mean grouping accuracy 95.40%. The second case (preparation and evaluation within a DSP processor) has the highest classification precision (97.75%) and also the mean classification precision (92) %. This demonstrates that this system is able to understand client purpose in a testing situation.

[5] In this article Hong Cheng, Lu Yang depicted an overview of a few later works on hand gesture acknowledgment utilizing 3D profundity sensors. They begin with a survey of the commercial profundity sensors and open information sets that are broadly utilized in this field. At that point, we audit the state-of-the-art investigation for 3D hand gesture acknowledgment in four viewpoints: 1) 3D hand modeling; 2) inactive hand signal recognition; 3) hand direction gesture acknowledgment; and 4) persistent hand signal acknowledgment. Whereas the accentuation is on 3D hand signal acknowledgment approaches, the related works and ordinary frameworks are moreover briefly summarized for specialists.

3. PROPOSED WORK:

We are going to implement this idea in four following steps to get our fully developed

hand sign language interpreter. The steps involved are Manual Convolution neural network, Alex net, LeNet, Django framework implementation.

A. Architecture:

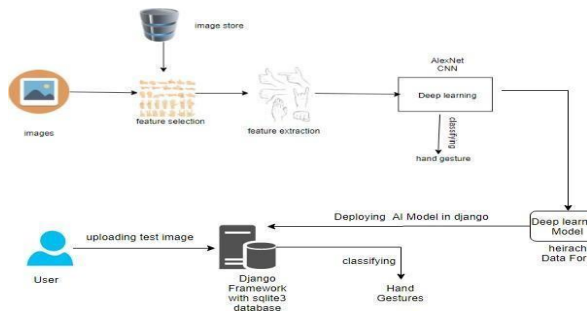


Fig 1. Architecture

B. Manual CNN:

Firstly, we are creating a manual convolution neural network. This manually created convolution neural network has no layer formats, unlike Alexnet and Lenet. Firstly, we are importing all the required packages and modules like OS, NumPy, Matplotlib, TensorFlow, Keras, and others. Now we are importing the dataset containing images for our model by locating our dataset folders in the directory. At this point, we are displaying the information of the images in the dataset for each hand gesture. The information displayed is image count, minimum width, maximum width, minimum height, and maximum height for every hand gesture. Now we are creating a sequential model with few parameters like convolution 2D, input shape, activation, dense, flatten, and max pooling. We use convolution 2D for filtering the pixels in images and Relu activation for better activation of neurons. To make our model understand the shape of the image which it should get and flatten for converting the data into a 1-dimensional array. And now we are displaying the classifier summary in table format. At this point, we are training our data for 50 epochs. Now we obtain our accuracy and loss percentage at the end of our training. Now, by using Matplotlib we are plotting a graph for accuracy and loss.

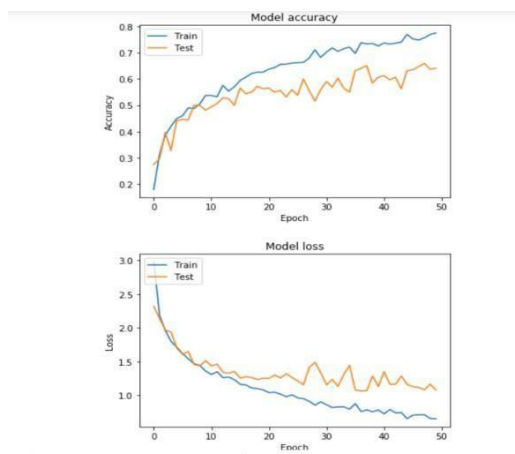


Fig 2. Manual CNN accuracy & loss graph

C. AlexNet:

Unlike the manually created Convolution neural network, AlexNet has layer formats. Now we are going to implement our model in AlexNet. Implementing the model in Alexnet involves 11 layers. The first layer is Convolutional layer 1, where the filter is 96 and kernel size is (11 x 11). The second layer is the max-pooling layer where pool size is (3,3). The third layer is convolution layer 2, where the filter is 256. The fourth layer is again a max-pooling layer. The fifth layer is convolution layer 3, where the filter is 384. The Sixth layer is the convolution layer 4, where the filter is 384. The seventh layer is convolution layer 5, where the filter is 256. The eighth layer is again a max-pooling layer with (3 x 3). After this point, we are using flatten to convert it into a 1-dimensional array and pass it to a fully connected layer. The ninth layer is a fully connected layer 1 with a dense value 4096. Now, we are making a dropout of (0.4) to prevent the issues of overfitting. The tenth layer is a fully connected layer 2 with a dense value of 4096. And the final layer is fully connected layer 3 with a dense value of 1000. Finally, our output layer will with SoftMax activation and a dense value of 16. Now we are training our dataset for 30 epochs. Now, we obtain our accuracy and loss graph for our AlexNet CNN. Here we obtain an accuracy rate of 67.42%.

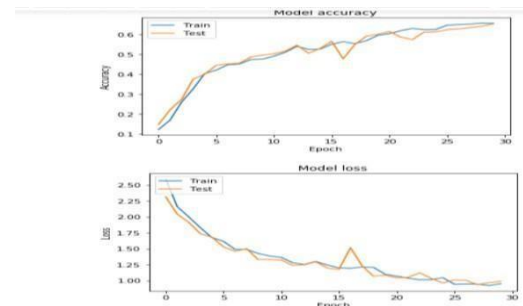


Fig 3. AlexNet accuracy & loss graph

D. LeNet:

Here in LeNet, we implement our model in 7 layers. The first layer is convolutional layer 1 with convolution 2D where we give filter with 6, kernel size with 5, Relu activation, and input shape of (225,225,3). The second layer is max-pooling layer 1 with a pool size of 2. The third layer is convolution layer 2 with convolution 2D where we give filter with 16, kernel size with 5, Relu activation, and input shape with (225,225,6). The fourth layer is

max-pooling layer 2 with a pool size of 2. Now, we apply flatten to convert it to single dimension array. The fifth layer is fully connected layer 1 with a

dense of 120 units and Relu activation. The sixth layer is fully connected layer 2 with a dense of 84 units and Relu activation. And the last layer is the output layer with a dense of 16 units and softmax activation. At this point, we display the classifier summary. Now, we train the dataset for 30 epochs and produce the accuracy and loss graph. Here we obtain an accuracy rate of 90.72%.

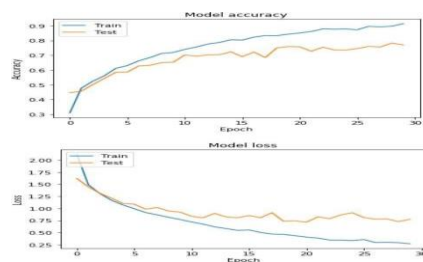


Fig 4. LeNet accuracy & loss graph

E. Deploying the model in Django framework:

At this point, we have implemented our model in a manually created convolution neural network, AlexNet CNN and LeNet CNN. Now, we compare the accuracy percentage of the model in all the convolution neural networks. As we observe, the accuracy percentage in LeNet is higher. So, the trained deep learning model is converted into a hierarchical data format file (.h5 file) which is then deployed in our Django framework for providing a better user interface and predicting the output which type of gesture is it.

4. RESULTS AND DISCUSSION

In this section, the result of the proposed model is explained and discussed. The step-by-step process of how the implementation takes place has been shown in this section.

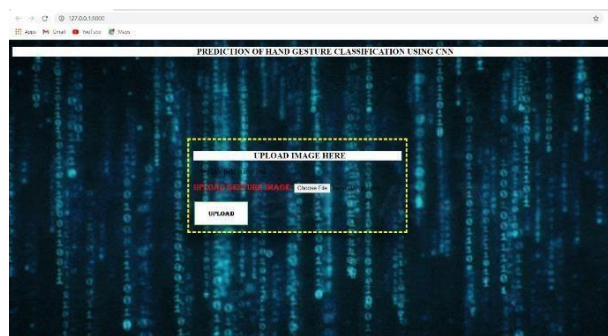


Fig 6. Home Screen

In this module Fig-6, the user interface runs on localhost, which is created using the Django

framework. Html and CSS are used in the frontend and in the backend, MySQL is used. The UI consist of the home page with the title "PREDICTION OF HAND GESTURE CLASSIFICATION USING CNN" and here the input image is uploaded.

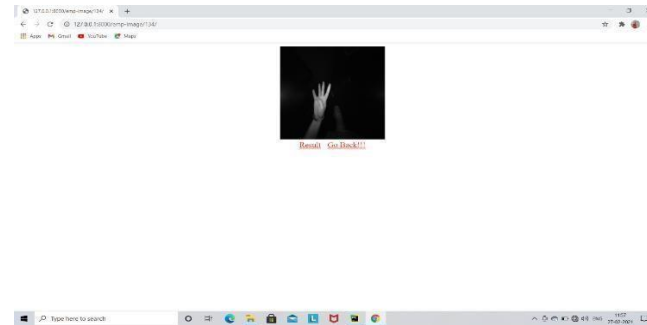


Fig 7. Intermediate Screen

In this module Fig-7, the uploaded image is shown. It provides two options; one is the result and another one is to go back. The go-back option navigates to the previous page and the result option navigate to the result page.



Fig 8. Final output screen

In this module Fig-8, the predicted result of the proposed model is provided and this is the final module of the hand sign language interpreter.

5. CONCLUSION:

Hand sign language interpreter addresses the fault in communication among people with disabilities. This proposed system resolves the common issues among deaf people. The trained model can process the continuous stream of inputs and predict results with more accuracy. The trained accuracy of the model is approximately 78%, whereas the prediction of the result is 90.72%. There are various techniques and methods used among the hand gesture projects. Each technique has their advantages and drawbacks, however, the methodology used in this project suits and perform well in most challenging cases. Since

the model has better accuracy in both trained

datasets and prediction, it is more suitable for real-time applications.

REFERENCES

- [1] Takuichi Nishimura, Hiroaki Yabe, Ryuichi Oka, Toshiro Mukai, "Recognition of Gestures Using Morphological Features of Networks Made of Gesture Motion Images and Word Sequences", vol. 00, no., pp. 39, 1999.
- [2] E. Stergiopoulou and N. A. Papamarkos, "Engineering Applications of Artificial Intelligence Hand gesture recognition using a neural network shape fitting technique," *Eng. Appl. Artif. Intell.*, vol. 22, no. 8, pp. 1141–1158, 2009.
- [3] N. H. Adnan, K. Wan, and A. B. Shahrman, "Measurement of the Flexible Bending Force of the Index and Middle Fingers for Virtual Interaction," vol. 41, no. *Iris*, pp. 388–394, 2012.
- [4] Z. Y. Meng, J. S. Pan, K. K. Tseng, and W. Zheng, "Dominant points based hand finger counting for recognition under skin color extraction in hand gesture control system," *Proc. - 2012 6th Int. Conf. Genet. Evol. Comput. ICGEC 2012*, pp. 364–367, 2012.
- [5] M. M. Hasan and P. K. Mishra, "Robust Gesture Recognition Using Gaussian Distribution for Features Fitting," vol. 2, no. 3, 2012.
- [6] M. M. Gharasuie and H. Seyedarabi, "Real-time Dynamic Hand Gesture Recognition using Hidden Markov Models," pp. 194–199, 2013.
- [7] M. Murugeswari and S. Veluchamy, "Hand gesture recognition system for real-time application," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, 2014, pp. 1220–1225.
- [8] G. Qh, N. Cpf, Q. Pikpggtpki, G. Qh, N. Cpf, Q. Pikpggtpki, O. Ucokv, C. T. K. Iockn, O. Fkrcmmwoct, and I. Iockn, "Static Hand Gesture Recognition Using Mixture of Features and SVM Classifier," pp. 1094–1099, 2015.
- [9] X. Sun, L. Liu, H. Wang, W. Song, and J. Lu, "Image Classification via Support Vector Machine," no. *Iccsnt*, pp. 485–489, 2015.
- [10] J. J. L. Jr., *A Survey of Hand Posture and Gesture Recognition Techniques and Technology*, vol. 102.
- [11] H. Zhou, D. J. Lin, and T. S. Huang, "Static Hand Gesture Recognition based on Local Orientation Histogram Feature Distribution Model,"
- [12] M. A. Rashid and X. Han, "Gesture Control of ZigBee Connected Smart Home Internet of Things," pp. 667–670, 2016.
- [13] A. A. Hassan and S. N. Basah, "Gesture-based remote-control system using coordinate features," vol. 11, no. 8, pp. 4979–4986, 2016.

C. SAMPLE CODE:

```
import os

import numpy as np # linear algebra

import matplotlib.pyplot as plt

# DL framework - tensorflow, keras a backend

import tensorflow as tf

import tensorflow.keras.backend as K

from tensorflow.keras.models import Model, Sequential

from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization

from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D,
LeakyReLU, Activation

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping

from IPython.display import display

from os import listdir

from os.path import isfile, join

from PIL import Image

import glob

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

dir_name_train_palm = 'dataset/train_pose/01_palm'
dir_name_train_l = 'dataset/train_pose/02_l'
dir_name_train_fist_moved = 'dataset/train_pose/04_fist_moved'
dir_name_train_down = 'dataset/train_pose/05_down'
dir_name_train_index = 'dataset/train_pose/06_index'
dir_name_train_ok = 'dataset/train_pose/07_ok'
dir_name_train_palm_m = 'dataset/train_pose/08_palm_m'
```

```

dir_name_train_c = 'dataset/train_pose/09_c'
dir_name_train_heavy = 'dataset/train_pose/11_heavy'
dir_name_train_hang = 'dataset/train_pose/12_hang'
dir_name_train_two = 'dataset/train_pose/13_two'
dir_name_train_three = 'dataset/train_pose/14_three'
dir_name_train_four = 'dataset/train_pose/15_four'
dir_name_train_five = 'dataset/train_pose/16_five'
dir_name_train_palm_u = 'dataset/train_pose/17_palm_u'
dir_name_train_up = 'dataset/train_pose/18_up'
def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)

    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(35, 10))
    for idx, img_path in enumerate(item_files):
        plt.subplot(2, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()
def Images_details_Print_data(data, path):
    print(" ===== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.\"", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100 # No image will be bigger than that

```

```
data['max_width'] = 0
data['min_height'] = 10**100 # No image will be bigger than that
data['max_height'] = 0
```

```
for f in files:
```

```
    im = Image.open(f)
    width, height = im.size
    data['min_width'] = min(width, data['min_width'])
    data['max_width'] = max(width, data['max_width'])
    data['min_height'] = min(height, data['min_height'])
    data['max_height'] = max(height, data['max_height'])
```

```
Images_details_Print_data(data, path)
print("")
print("Trained data for Palm:")
print("")
Images_details(dir_name_train_palm)
print("")
plot_images(dir_name_train_palm, 10)
Classifier=Sequential()
Classifier.add(Convolution2D(64,(3,3),input_shape=(225,225,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(32, activation='relu'))
Classifier.add(Dense(16, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
Classifier.summary()
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

```

training_set=train_datagen.flow_from_directory('dataset/train_pose',target_size=(2
25,225),batch_size=32,class_mode='categorical')

test_set=test_datagen.flow_from_directory('dataset/test_pose',target_size=(225,2
25),batch_size=32,class_mode='categorical')

img_dims = 150

epochs = 50

batch_size = 64

#### Fitting the model

history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set,validation_steps=test_set.samples // batch_size)

def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # Plot training & validation loss values
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

graph()

def Images_details_Print_data(data, path):
    print(" ===== Images in: ", path)

```

```

for k, v in data.items():
    print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.\"", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100 # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100 # No image will be bigger than that
    data['max_height'] = 0

    for f in files:
        im = Image.open(f)
        width, height = im.size
        data['min_width'] = min(width, data['min_width'])
        data['max_width'] = max(width, data['max_height'])
        data['min_height'] = min(height, data['min_height'])
        data['max_height'] = max(height, data['max_height'])

    Images_details_Print_data(data, path)

#Conv Layer 1
model.add(Conv2D(filters = 6,
                  kernel_size = 5,
                  strides = 1,
                  activation = 'relu',
                  input_shape = (225,225,3)))

#Pooling layer 1
model.add(MaxPooling2D(pool_size = 2, strides = 2))

#Layer 2
#Conv Layer 2

```

```

model.add(Conv2D(filters = 16,
                  kernel_size = 5,
                  strides = 1,
                  activation = 'relu',
                  input_shape = (225,225,6)))

#Pooling Layer 2
model.add(MaxPooling2D(pool_size = 2, strides = 2))

#Layer 3
#Fully connected layer 1
model.add(Dense(units = 120, activation = 'relu'))

#Layer 4
#Fully connected layer 2
model.add(Dense(units = 84, activation = 'relu'))

#Layer 5
#Output Layer
model.add(Dense(units = 16, activation = 'softmax'))

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

model.summary()

#### Fitting the model
history = model.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set,validation_steps=test_set.samples // batch_size)

```