

AMERICAN SIGN LANGUAGE RECOGNITION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering (B.E) Degree in

Computer Science and Engineering

By

V.L.V. SUSEEL KUMAR (Reg. No. 37110817)

V.SAI TEJA (Reg. No. 37110844)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119. TAMIL NADU

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.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of VELAGAPUDY LEELA VENKATA SUSEEL KUMAR (Reg. No. 37110817) & VISHNUMOLAKA SAI TEJA (Reg. No. 37110844) who underwent the professional training in "**American Sign Language to Text Using PYTHON and CNN**" under our supervision from December 2020 to March 2021.

Internal Guide

Dr. R. SUBHASHINI, M.E., Ph.D.,

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We, VELAGAPUDY LEELA VENKATA SUSEEL KUMAR (Reg. No. 37110817) & VISHNUMOLAKA SAI TEJA (Reg. No. 37110844) here by declare that the Professional Training Report on “**American Sign Language to Text Using PYTHON and CNN**” done by us under the guidance of Dr. R. SUBHASHINI, ME., Ph.D., at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of technology degree in **Information Technology**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the Board of **Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sasikala, M.E., Ph.D, Dean**, School of Computing , **Dr. R. Subhashini M.E., Ph.D. , Head** of the Department of Information Technology for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. R. Subhashini M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of our project work.

We wish to express our 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

Sign Languages also popularly called Signed Languages. This is the only language by which deaf and dumb can communicate with each other and one of the most unexplored areas. So the proposed paper represents a model that can be used to translate given sign language to simple English text so that it could be used for dumb people. By late proceedings in the field of profound learning, there exist many applications that neural network systems can solve one such problem is this sign language. To meet the requirements, the proposed model uses deep learning techniques to solve the problem. One such technique is Convolutional Neural Networks for image depiction and classification. The model designed in the proposed paper has a dataset that consists of 87,002 images. The dataset has been split into training data and test data in the ratio of 9:1. The proposed model was trained on 78,300 images and the testing was performed on 8700 images which were classified in 29 classes. The model has produced a training accuracy of 98.67%.

CHAPTER	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	1
2	LITERATURE SURVEY	9
	2.1 Economic Feasibility	10
	2.2 Unsupervised Training	11
	2.3 Supervised Training	12
	2.4 The Structure of Feed Neural Network	13
	2.5 The Input and Output Layer	14
	2.6 Vision Based Detection with voice output	15
	2.6.1 Introduction	15
	2.6.2 Image Capturing and Preprocessing	16
	2.6.3 Automatic Text Extraction	16
	2.6.4 Text Region Localization	17
	2.6.5 Text Recognition and Output	18
3	SYSTEM DESIGN	22

4	DEEP LEARNING	25
	4.1 Overview	25
	4.2 Neural Network	26
	4.2.1 Artificial Neuron	26
	4.2.2 Artificial Neuron Network	27
	4.2.3 Forward Propagation	27
	4.2.4 Backward Propagation	28
	4.2.5 Activation Function	28
	4.3 Convolutional Neural Network	29
	4.3.1 Overview	29
	4.3.2 Architecture	29
	4.3.3 Convolutional Layer	30
	4.3.4 Pooling Layer	30
	4.4 U-Net Model	31
	4.4.1 Overview	31
	4.4.2 Architecture	31
	4.5 Improving Deep Neural Network	32
	4.5.1 Overview	32
	4.5.2 Regularization using Dropout	33

	4.5.3 Weight Initialization	34
	4.5.4 Gradient Descent	35
	4.5.5 Variant of Gradient Descent	36
	DATA SET	42
5	OCR ALGORITHM	44
6	PYTHON	52
7	RESULT AND DISCUSSION	59
8	SOURCE CODE	60
	REFERCE	63
	SCREENSHOTS (with outputs)	64

LIST OF FIGURES

FIGURES	TITLE	PAGE NO
6	OCR ARCHITECTURE	4
	FEED FORWARD NEURAL NETWORK	13
	BLOCK DIAGRAM	24
	OBJECT DETECTION	29
	BLOCK DIAGRAM FOR SIGN CONVENTION	41
	DATA SETS	42

LIST OF ABBREVIATIONS

SVM	Support Vector Machine
RNN	Recurrent Neural Network
OCR	Optical Character Recognition
Open CV	Open Source Computer Vision Library
CNN	Convolution Neural Network

CHAPTER 1

1. INTRODUCTION

In the running world, there is growing demand for the software systems to recognize characters in computer system when information is scanned through paper documents as we know that we have number of newspapers and books which are in printed format related to different subjects. These days there is a huge demand in “storing the information available in these paper documents in to a computer storage disk and then later reusing this information by searching process”. One simple way to store information in these paper documents in to computer system is to first scan the documents and then store them as IMAGES. But to reuse this information it is very difficult to read the individual contents and searching the contents form these documents line-by-line and word-by-word. The reason for this difficulty is the font characteristics of the characters in paper documents are different to font of the characters in computer system. As a result, computer is unable to recognize the characters while reading them. This concept of storing the contents of paper documents in computer storage place and then reading and searching the content is called DOCUMENT PROCESSING. Sometimes in this document processing we need to process the information that is related to languages other than the English in the world. For this document processing we need a software system called **CHARACTER RECOGNITION SYSTEM**. This process is also called DOCUMENT IMAGE ANALYSIS (DIA).

Thus our need is to develop character recognition software system to perform Document Image Analysis which transforms documents in paper format to electronic format. For this process there are various techniques in the world. Among all those techniques we have chosen Optical Character Recognition as main fundamental technique to recognize characters. The conversion of paper documents in to electronic format is an on-going task in many of the organizations particularly in Research and Development (R&D) area, in large business enterprises, in government institutions, so on. From our problem statement we can introduce the necessity of Optical Character Recognition in mobile electronic devices such as cell phones, digital cameras to acquire images and recognize them as a part of face recognition and validation.

To effectively use Optical Character Recognition for character recognition in-order to perform Document Image Analysis (DIA), we are using the information in Grid format. . This system is thus effective and useful in *Virtual Digital Library*'s design and construction.

PURPOSE

The main purpose of **Optical Character Recognition (OCR)** system based on a grid infrastructure is to perform Document Image Analysis, document processing of electronic document formats converted from paper formats more effectively and efficiently. This improves the accuracy of recognizing the characters during document processing compared to various existing available character recognition methods. Here OCR technique derives the meaning of the characters, their font properties from their bit-mapped images.

- The primary objective is to speed up the process of character recognition in document processing. As a result the system can process huge number of documents with-in less time and hence saves the time.
- Since our character recognition is based on a grid infrastructure, it aims to recognize multiple heterogeneous characters that belong to different universal languages with different font properties and alignments.

PROJECT SCOPE

The scope of our product Optical Character Recognition on a grid infrastructure is to provide an efficient and enhanced software tool for the users to perform Document Image Analysis, document processing by reading and recognizing the characters in research, academic, governmental and business organizations that are having large pool of documented, scanned images. Irrespective of the size of documents and the type of characters in documents, the product is recognizing them, searching them and processing them faster according to the needs of the environment.

EXISTING SYSTEM

In the running world there is a growing demand for the users to convert the printed documents in to electronic documents for maintaining the security of their data. Hence the basic OCR system was invented to convert the data available on papers in to computer process able documents, So that the documents can be editable and reusable. The existing

system/the previous system of OCR on a grid infrastructure is just OCR without grid functionality. That is the existing system deals with the homogeneous character recognition or character recognition of single languages.

DRAWBACK OF EXISTING SYSTEM

The drawback in the early OCR systems is that they only have the capability to convert and recognize only the documents of English or a specific language only. That is, the older OCR system is uni-lingual.

PROPOSED SYSTEM

Our proposed system is OCR on a grid infrastructure which is a character recognition system that supports recognition of the characters of multiple languages. This feature is what we call grid infrastructure which eliminates the problem of heterogeneous character recognition and supports multiple functionalities to be performed on the document. The multiple functionalities include editing and searching too where as the existing system support only editing of the document. In this context, Grid infrastructure means the infrastructure that supports group of specific set of languages. Thus OCR on a grid infrastructure is multi-lingual.

BENEFIT OF PROPOSED SYSTEM

The benefit of proposed system that overcomes the drawback of the existing system is that it supports multiple functionalities such as editing and searching. It also adds benefit by providing heterogeneous characters recognition.

ARCHITECTURE OF THE PROPOSED SYSTEM

The Architecture of the optical character recognition system on a grid infrastructure consists of the three main components. They are:-

- Scanner
- OCR Hardware or Software
- Output Interface

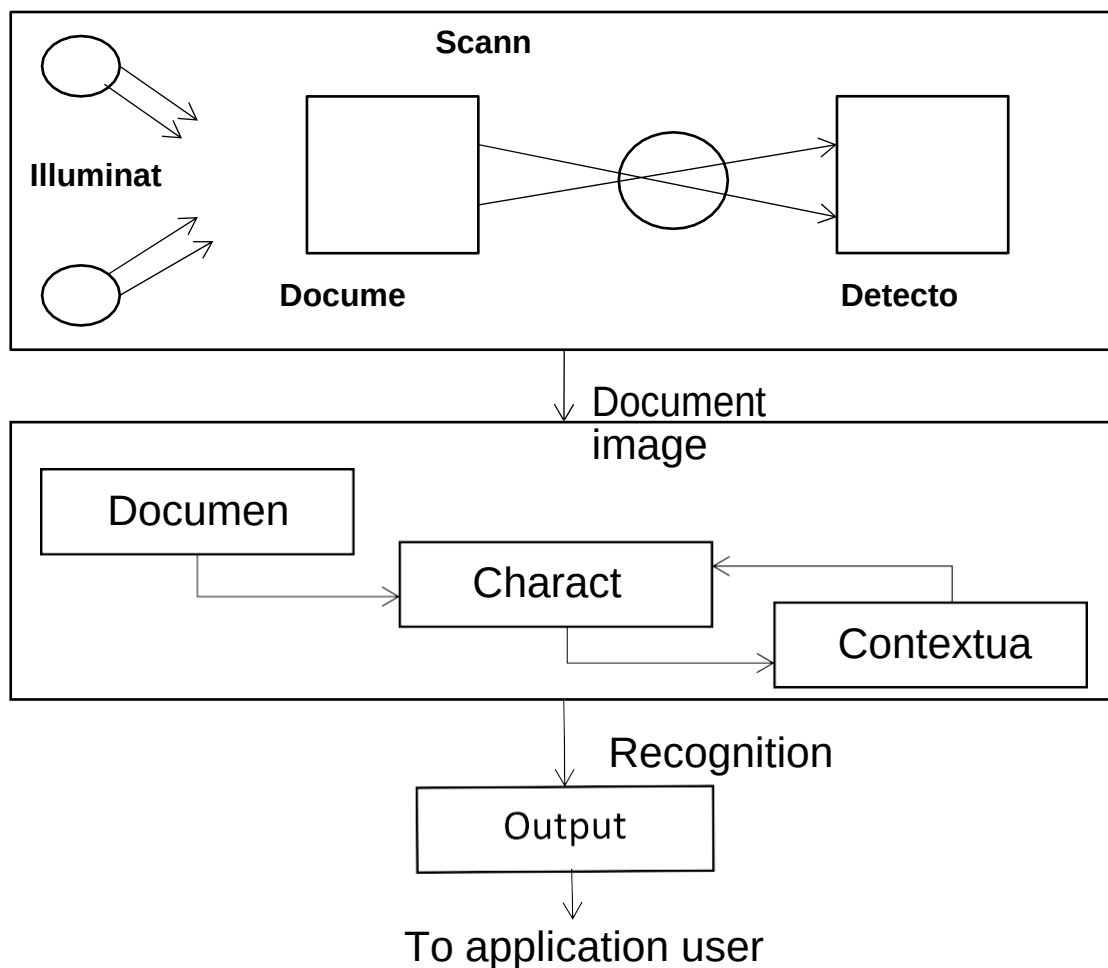


Figure.1: OCR Architecture

INTENDED AUDIENCE AND READING SUGGESTIONS

In this section, we identify the audience who are interested with the product and are involved in the implementation of the product either directly or indirectly. As from our research, the OCR system is mainly useful in R&D at various scientific organizations, in governmental institutes and in large business organizations, we identify the following as various interested audience in implementing OCR system:-

- The scientists, the research scholars and the research fellows in telecommunication institutions are interested in using OCR system for processing the word document that contains base paper for their research.
- The Librarian to manage the information contents of the older books in building virtual digital library requires use of OCR system.

- Various sites that vendor e-books have a huge requirement of this OCR system in-order to scan all the books in to electronic format and thus make money. The Amazon book world is largely using this concept to build their digital libraries.

Now we present the reading suggestions for the users or clients through which the user can better understand the various phases of the product. These suggestions may be effective and useful for the beginners of the product rather than the regular users such as research scholars, librarians and administrators of various web-sites. With these suggestions, the user need not waste his time in scrolling the documents up and down, browsing through the web, visiting libraries in search of different books and ... The following are the various reading suggestions that the user can follow in-order to completely understand about our product and to save time:-

- It would help you if you start with Wikipedia.com. It lets you know the basic concept of every keyword you require. First learn from it what is OCR? And how does it work based on a Grid infrastructure?
- Now you can proceed your further reading with the introduction of our product we provided in our documentation. From these two steps you completely get an in-depth idea of the use of our product and several processes involved in it.
- The more you need is the implementation of the product. For this you can visit FreeOCR.com where you can view how the sample OCR works and you can try it.

PROBLEM STATEMENT:

- Traditional methods like Braille exist using which the blind people have to trace and read text, which is very slow and not very practical.
- Existing OCR systems are not automatic and require full-fledged computers to run and hence are not effective.
- 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. This goal is further motivated by the isolation that is felt within the deaf

community. Loneliness and depression exists in higher rates among the deaf population, especially when they are immersed in a hearing world [1]. 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 [2]. Most research implementations for this task have used depth maps generated by depth camera and high resolution images. 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 such as a laptop webcam. This is in alignment with the motivation as this would make a

- **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 preprocess 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.
- **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 Dataloaders 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.
- **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 filepath of the gesture image and the `test data.py` script will pass the filepath to `process data.py` to load and preprocess the file the same way as the model has been trained.

Motivation and Overview The advances in Human Computer Interaction (HCI) in the recent times have led to the growth of man machine communication. With this thrust of development, the technology has contributed significantly to social applications like sign language recognition. Sign language is the prime means of communication between a signer and a non-signer. It is the only formal medium of communication by which mute persons and hearing impaired people express themselves using body movement or facial expressions. Sign language recognition systems make use of mainly hand gestures to communicate information. Hand gesture recognition is another widely growing area which has many applications in various areas such as entertainment industry including interactive virtual tourisms and computer games, controlling industry including 2D and 3D pointing, control of household appliances, objects grasping, command descriptions. Most of the gestures are performed with the hand but also with the face and the body. Hand shape together with its movement and position with respect to the other body parts, forms a hand gesture. Hand gestures are used in every aspect of human communication to accompany speech, or alone for communication in noisy environment and are the most important aspect of the aspect of sign language as most of the information is communicated through hand by the signers. Developing algorithms and techniques to correctly recognize a sequence of produced signs and understand their meaning is called sign language recognition (SLR). SLR is a hybrid research area involving pattern recognition, natural language processing, computer vision and linguistics [1]. Sign Language recognition systems can be used as an interface between human being and computer systems. Sign languages are complete natural language with their phonology, morphology, syntax Chapter 1 Introduction Dept. of CEA, GLAU, Mathura 2 and grammar. A sign language is a visual-gesture language that is developed to facilitate the differently abled persons by creating visual gestures using face, hand, body and arms [2]. SLR and spoken language are drastically different in many respects: 1. The communication medium for spoken language is dependent on sound while, the communication medium for spoken language depends upon the visual channels. 2. Because of the multi-channel of sign language, time is relatively less important for the meaning of a sign as compared to spoken word [3] 3. Phonemes are the smallest constructing unit of words in spoken language. Minimum one phoneme is required to differentiate two words. For sign language, cheremes are used to build up a sign and at least one chereme must be different among different signs [2]. The sign language being used in India is called Indian Sign Language (henceforth called ISL). Similarly the sign language of other parts of the world are American Sign Language (ASL), British Sign Language (BSL), Japanese Sign

Language (JSL), Arabic Sign Language (ArSL) and German Sign Language (GSL), etc. There are different dialects of ISL which are present in different parts of India. However, all the dialects possess the same grammatical structure despite having large lexical variation [4]. A sign is made by cheremes that can be either manual or non-manual [2]. Cheremes are the sub units of sign which are equivalent to the phonemes of spoken languages. Manual cheremes consist of several parameters like: hand shape, hand orientation, hand location and hand motion and non-manual cheremes are defined by facial expression, head/body posture. There are three categories of ISL signs. They are: one handed, two handed, and non-manual signs. However, there are certain signs which use only either the manual parameters or non-manual parameters. For example, a single waving hand representing the sign of “Hello” has no non-manual components and sign “Yes” is depicted by nodding the head vertically without any manual component. Figure 1.1 shows the overall Indian sign hierarchy

CHAPTER 2

2. LITERATURE REVIEW

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

TECHNICAL FEASIBILITY

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?.

OPERATIONAL FEASIBILITY

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

- Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
- Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
- Have the user been involved in the planning and development of the project?

- Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

2.1 ECONOMIC FEASIBILITY

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

TRAINING

Training is a very important process of working with a neural network. As seen from neural networks, there are two forms of training that can be employed with a neural network. They are namely:-

1. Un-Supervised Training
2. Supervised Training

Supervised training provides the neural network with training sets and the anticipated output. Unsupervised training supplies the neural network with training sets, but there is no anticipated output provided.

2.2 UNSUPERVISED TRAINING

Unsupervised training is a very common training technique for Kohonen neural networks. We will discuss how to construct a Kohonen neural network and the general process for training without supervision.

What is meant by training without supervision is that the neural network is provided with training sets, which are collections of defined input values. But the unsupervised neural network is not provided with anticipated outputs.

Unsupervised training is usually used in a classification neural network. A classification neural network takes input patterns, which are presented to the input neurons. These input patterns are then processed, and one single neuron on the output layer fires. This firing neuron can be thought of as the classification of which group the neural input pattern belonged to. Handwriting recognition is a good application of a classification neural network.

The input patterns presented to the Kohonen neural network are the dot image of the character that was hand written. We may then have 26 output neurons, which correspond to the 26 letters of the English alphabet. The Kohonen neural network should classify the input pattern into one of the 26 input patterns.

During the training process the Kohonen neural network in handwritten recognition is presented with 26 input patterns. The network is configured to also have 26 output patterns. As the Kohonen neural network is trained the weights should be adjusted so that the input patterns are classified into the 26 output neurons. This technique results in a relatively effective method for character recognition.

Another common application for unsupervised training is data mining. In this case you have a large amount of data, but you do not often know exactly what you are looking for. You want the neural network to classify this data into several groups. You do not want to dictate, ahead of time, to the neural network which input pattern should be classified to which group. As the neural network trains the input patterns will fall into similar groups. This will allow you to see which input patterns were in common groups.

2.3 SUPERVISED TRAINING

The supervised training method is similar to the unsupervised training method in that training sets are provided. Just as with unsupervised training these training sets specify input signals to the neural network.

The primary difference between supervised and unsupervised training is that in supervised training the expected outputs are provided. This allows the supervised training algorithm to adjust the weight matrix based on the difference between the anticipated output of the neural network, and the actual output.

There are several popular training algorithms that make use of supervised training. One of the most common is the back-propagation algorithm. It is also possible to use an algorithm such as simulated annealing or a genetic algorithm to implement supervised training

INTRODUCING KOHONEN NEURAL NETWORK

The Kohonen neural network differs considerably from the feed-forward back propagation neural network. The Kohonen neural network differs both in how it is trained and how it recalls a pattern. The Kohonen neural network does not use any sort of activation function. Further, the Kohonen neural network does not use any sort of a bias weight.

Output from the Kohonen neural network does not consist of the output of several neurons. When a pattern is presented to a Kohonen network one of the output neurons is selected as a "winner". This "winning" neuron is the output from the Kohonen network. Often these "winning" neurons represent groups in the data that is presented to the Kohonen network. For example, in an OCR program that uses 26 output neurons, the 26 output neurons map the input patterns into the 26 letters of the Latin alphabet.

The most significant difference between the Kohonen neural network and the feed forward back propagation neural network is that the Kohonen network is trained in an unsupervised mode. This means that the Kohonen network is presented with data, but the correct output that corresponds to that data is not specified. Using the Kohonen network this data can be classified into groups. We will begin our review of the Kohonen network by examining the training process.

It is also important to understand the limitations of the Kohonen neural network. Neural networks with only two layers can only be applied to linearly separable problems. This is the case with the Kohonen neural network. Kohonen neural networks are used because they are a relatively simple network to construct that can be trained very rapidly.

A "feed forward" neural network is similar to the types of neural networks that we are ready examined. Just like many other neural network types the feed forward neural network begins with an input layer. This input layer must be connected to a hidden layer. This hidden layer can then be connected to another hidden layer or directly to the output layer. There can be any number of hidden layers so long as at least one hidden layer is provided. In common use most neural networks will have only one hidden layer. It is very rare for a neural network to have more than two hidden layers. We will now examine, in detail, and the structure of a "feed forward neural network".

2.4 The Structure of a Feed Forward Neural Network

A "feed forward" neural network differs from the neural networks previously examined. **Figure 2.1 shows a typical feed forward neural network with a single hidden layer.**

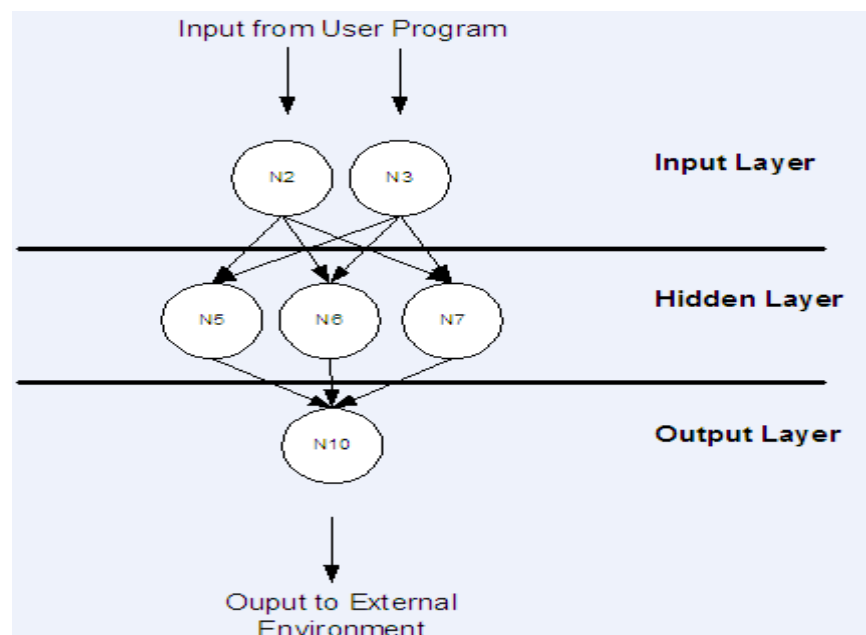


Figure 2 Feed Forward Neural Network

2.5 The Input Layer

The input layer to the neural network is the conduct through which the external environment presents a pattern to the neural network. Once a pattern is presented to the input layer of the neural network the output layer will produce another pattern. In essence this is all the neural network does. The input layer should represent the condition for which we are training the neural network for. Every input neuron should represent some independent variable that has an influence over the output of the neural network.

It is important to remember that the inputs to the neural network are floating point numbers. These values are expressed as the primitive Java data type "double". This is not to say that you can only process numeric data with the neural network. If you wish to process a form of data that is non-numeric you must develop a process that normalizes this data to a numeric representation.

The Output Layer

The output layer of the neural network is what actually presents a pattern to the external environment. Whatever pattern is presented by the output layer can be directly traced back to the input layer. The number of output neurons should directly related to the type of work that the neural network is to perform.

To consider the number of neurons to use in your output layer you must consider the intended use of the neural network. If the neural network is to be used to classify items into groups, then it is often preferable to have one output neurons for each groups that the item is to be assigned into. If the neural network is to perform noise reduction on a signal then it is likely that the number of input neurons will match the number of output neurons. In this sort of neural network you would one day he would want the patterns to leave the neural network in the same format as they entered.

For a specific example of how to choose the numbers of input and output neurons consider a program that is used for optical character recognition, or OCR. To determine the number of neurons used for the OCR example we will first consider the input layer. The number of input neurons that we will use is the number of pixels that might represent any given character. Characters processed by this program are normalized to universal size that is represented by a 5x7 grid. A 5x7 grid contains a total of 35 pixels. The optical character recognition program therefore has 35 input neurons.

The number of output neurons used by the OCR program will vary depending on how many characters the program has been trained for. The default training file that is provided with the optical character recognition program is trained to recognize 26 characters. As a result using this file the neural network would have 26 output neurons. Presenting a pattern to the input neurons will fire the appropriate output neuron that corresponds to the letter that the input pattern corresponds to.

2.6. VISION BASED ASSISTIVE SYSTEM FOR LABEL DETECTION WITH VOICE OUTPUT

A camera based assistive text reading framework to help blind persons read text labels and product packaging from hand-held object in their daily resides is proposed. To isolate the object from cluttered backgrounds or other surroundings objects in the camera view, we propose an efficient and effective motion based method to define a region of interest (ROI) in the video by asking the user to shake the object. In the extracted ROI, text localization and recognition are conducted to acquire text information. To automatically localize the text regions from the object ROI, we propose a novel text localization algorithm by learning gradient features of stroke orientations and distributions of edge pixels in an Adaboost model. Text characters in the localized text regions are then binarized and recognized by off-the-shelf optical character recognition software. The recognized text codes are output to blind users in speech

2.6.1 INTRODUCTION

Of the 314 million visually impaired people worldwide, 45 million are blind. Recent developments in computer vision, digital cameras and portable computers make it feasible to assist these individuals by developing camera based products that combine computer vision technology with other existing commercial products such optical character recognition (OCR) systems.

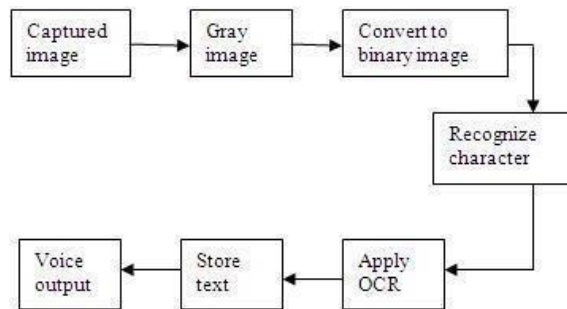
Reading is obviously essential in today's society. Printed text is everywhere in the form of reports, receipts, bank statements, restaurant menus, classroom handouts, product packages, instructions on medicine bottles, etc.

The ability of people who are blind or have significant visual impairments to read printed

labels and product packages will enhance independent living and foster economic and social self-sufficiency. Today, there are already a few systems that have some promise for portable use, but they cannot handle product labelling. Selectively extract the image of the object held by the blind user from the cluttered background or other neutral objects in the camera view; and 2) text localization to obtain image regions containing text, and text recognition to transform image-based text information into readable codes. We use a mini laptop as the processing device in our current prototype system. The audio output component is to inform the blind user of recognized text codes.

2.6.2 IMAGE CAPTURING AND PRE-PROCESSING

The video is captured by using web-cam and the frames from the video is segregated and undergone to the pre-processing. First, get the objects continuously from the camera and adapted to process. Once the object of interest is extracted from the camera image and it converted into gray image. Use haar cascade classifier for recognizing the character from the object. The work with a cascade classifier includes two major stages: training and detection. For training need a set of samples. There are two types of samples: positive and negative.



To extract the hand-held object of interest from other objects in the camera view, ask users to shake the hand-held objects containing the text they wish to identify and then employ a motion-based method to localize objects from cluttered background.

2.6.3 AUTOMATIC TEXT EXTRACTION

In order to handle complex backgrounds, two novel feature maps to extracts text features based on stroke orientations and edge distributions, respectively. Here, stroke is defined as a uniform region with bounded width and significant extent. These feature maps are

combined to build an Adaboost based text classifier.

2.6.4 TEXT REGION LOCALIZATION

Text localization is then performed on the camera based image. The Cascade-Adaboost classifier confirms the existence of text information in an image patch but it cannot the whole images, so heuristic layout analysis is performed to extract candidate image patches prepared for text classification. Text information in the image usually appears in the form of horizontal text strings containing no less than three character members.

2.6.5 TEXT RECOGNITION AND AUDIO OUTPUT

Text recognition is performed by off-the-shelf OCR prior to output of informative words from the localized text regions. A text region labels the minimum rectangular area for the accommodation of characters inside it, so the border of the text region contacts the edge boundary of the text characters. However, this experiment show that OCR generates better performance text regions are first assigned proper margin areas and binaries to segments text characters from background

The recognized text codes are recorded in script files. Then, employ the Microsoft Speech Software Development Kit to load these files and display the audio output of text information. Blind users can adjust speech rate, volume and tone according to their preferences. Are designed to easily interface with dedicated computer systems by using the same USB technology that is found on most computers. Static random-access memory (SRAM) is a type of a semiconductor memory that uses bi-stable latching circuitry to store each bit.

ARM11 features are, Supports 4-64k cache sizes, Powerful ARMV6 instruction set architecture, SIMD (Single Instruction Multiple Data) media processing extensions deliver up to 2x performance for video processing, and High-performance 64-bit memory system speeds data access for media processing and networking applications. LAN9512/LAN9512i contains an integrated USB 2.0 hub, two integrated downstream USB 2.0 PHYs, an integrated upstream USB 2.0 PHY, a 10/100 Ethernet PHY, a 10/100 Ethernet Controller, a TAP Controller and EEPROM Controller. Flash memory is an electronic non-volatile compute storage medium that can be an electrically erased and reprogrammed. Earphones either have wires for connection to a signal source such as an audio amplifier, radio, CD player, portable media player or have a wireless receiver, which is used to pick up signal

without using a cable

The proposed system ensures to read printed text on hand-held objects for assisting blind persons. In order to solve the common aiming problem for blind users, a motion-based method to detect the object of interest is projected, while the blind user simply shakes the object for a couple of seconds. This method can effectively distinguish the object of interest from background or other objects in the camera view. An Adaboost learning model is employed to localize text in camera-based images. Off the shelf OCR is used to perform word recognition on the localized text regions and transform into audio output for blind users.

Algorithms used:

Classification machine learning algorithms like SVM, k-NN are used for supervised learning, which involves labeling the dataset before feeding it into the algorithm for training. For this project, various classification algorithms are used: SVM, k-NN and CNN.

Feature extraction algorithms are used for dimensionality reduction to create a subset of the initial features such that only important data is passed to the algorithm. When the input to the algorithm is too large to be processed and is suspected to be redundant (like repetitiveness of images presented by pixels), then it can be converted into a reduced set of features. Feature extraction algorithms: PCA, LBP, and HoG, are used alongside classification algorithms for this purpose. This reduces the memory required and increases the efficiency of the model.

The algorithms used are as follow:

Support Vector Machine (SVM)

In SVM, each data point is plotted in an n-dimensional space (n is the number of features) with the value of each feature being the value of a particular coordinate. The classification is done by finding a hyper-plane that differentiates the classes the best.

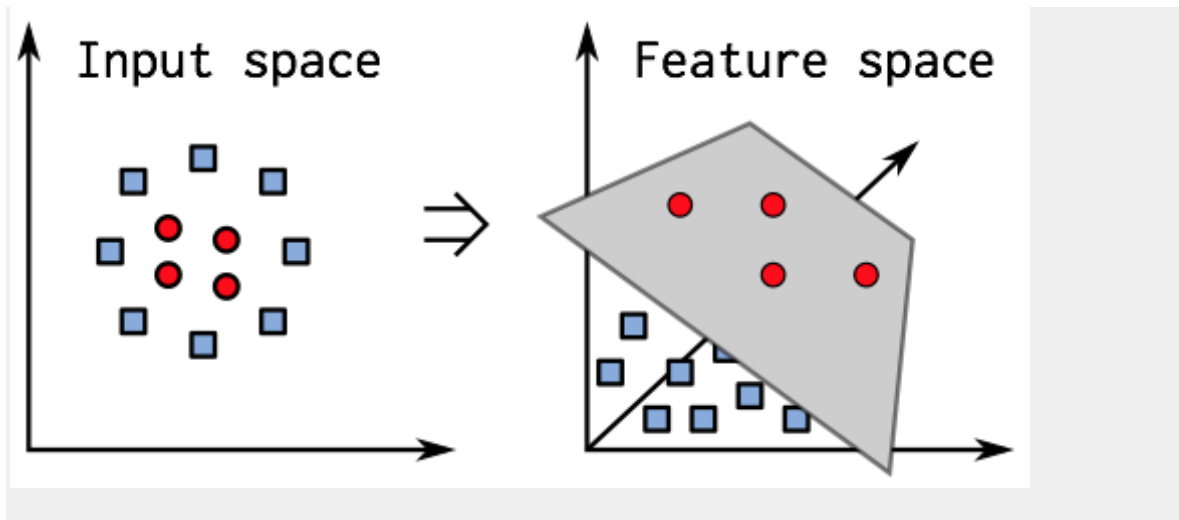


Fig 2 SVM classifier

k-NN (k-Nearest Neighbors)

In k-NN classification, an object is classified by a majority vote of its neighbours, with object assigned to the class that is the most common among its k-nearest neighbors, where k is a positive integer, typically small. The output of the algorithm is a class membership.

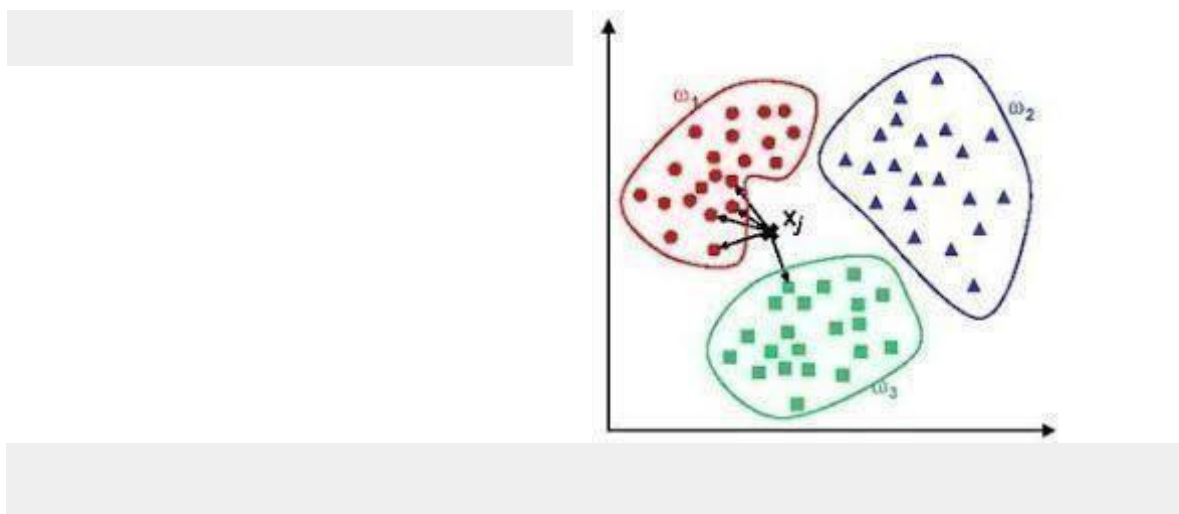


Fig 3 k-NN

CNN

Convolutional Neural Networks (CNN), are deep neural networks used to process data that have a grid-like topology, e.g images that can be represented as a 2-D array of pixels. A CNN model consists of four main operations: Convolution, Non-Linearity (Relu), Pooling and Classification (Fully-connected layer).

Convolution: The purpose of convolution is to extract features from the input image. It preserves the spatial relationship between pixels by learning image features using small squares of input data. It is usually followed by Relu.

Relu: It is an element-wise operation that replaces all negative pixel values in the feature map by zero. Its purpose is to introduce non-linearity in a convolution network.

Pooling: Pooling (also called downsampling) reduces the dimesionality of each feature map but retains important data.

Fully-connected layer: It is a multi layer perceptron that uses softmax function in the output layer. Its purpose is to use features from previous layers for classifying the input image into various classes based on training data.

The combination of these layers is used to create a CNN model. The last layer is a fully connected layer.

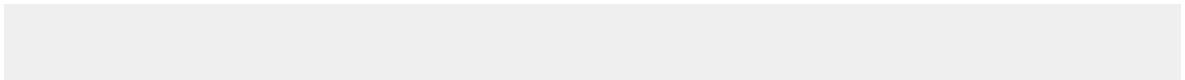


Fig 4 CNN

Pre-training a CNN model:

The concept of **Transfer learning** is used here, where the model is first pre-trained on a dataset that is different from the original. This way the model gains knowledge that can be transferred to other neural networks. The knowledge gained by the model, in the form of “weights” is saved and can be loaded into some other model. The pre-trained model can be used as a feature extractor by adding fully-connected layers on top of it. The model is trained with the original dataset after loading the saved weights.

PCA (Pricipal Component Analysis)

Using PCA, data is projected to a lower dimension for dimensionality reduction. The most important feature is the one with the largest variance or spread, as it corresponds to the largest entropy and thus encodes the most information. Thus the dimension with the largest variance is kept while others are reduced.

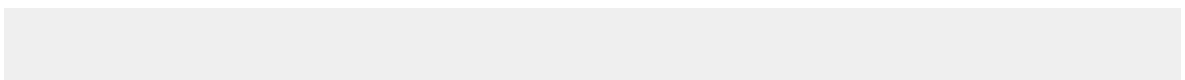


Fig 5 PCA feature reduction

LBP (Local Binary Patterns):

LBP computes a local representation of texture which is constructed by comparing each pixel by its surrounding or neighbourig pixels. The results of this are stored as an array which is then converted into decimal and stored as an LBP 2D array.

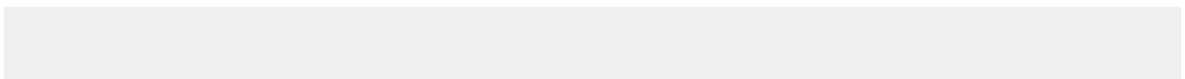
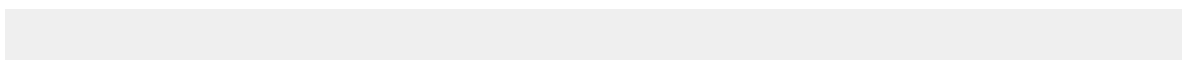


Fig 6 Coverision of pixel into LBP representation



HoG (Histogram of Gradients):

A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information.

Hog is a feature descriptor that calculates a histogram of gradient for the image pixels, which is a vector of 9 bins (numbers) corresponding to the angles: 0, 20, 40, 60, 160. The images are divided into cells, (usually, 8x8), and for each cell, gradient magnitude and gradient angle is calculated, using which a histogram is created for a cell. The histogram of a block of cells is normalized, and the final feature vector for the entire image is calculated.

CHAPTER 3:

SYSTEM DESIGN:

EXISTING SYSTEM:

- Traditional methods like Braille exist using which the blind people have to trace and read text, which is very slow and not very practical.
- Existing OCR systems are not automatic and require full-fledged computers to run and hence are not effective.
- KReader Mobile runs on a cell phone and allows the user to read mail, receipts, fliers, and many other documents

PROBLEM STATEMENT

- Understanding the exact context of symbolic expressions of deaf and dumb people is the challenging job in real life until unless it is properly specified.
- Due to brightness and contrast sometimes webcam can hardly detect the expected skin color.
- Because of the similarity of tracking environment background color and skin color the SLR gets unexpected pixels.
- Hard to implement in the hardware setup
- The output of the whole system is not a accuracy one.

PROPOSED SYSTEM:

- • Sign language is learned by deaf and dumb, and usually it is not known to normal people, so it becomes a challenge for communication between a normal and hearing-impaired person.
- Its strike to our mind to bridge the between hearing impaired and normal people to make the communication easier.
- Sign language recognition (SLR) system takes an input expression from the hearing-impaired person gives output to the normal person in the form text or voice.

- Sign language recognizer (SLR) is a tool for recognizing sign language of deaf and dumb people of the world.

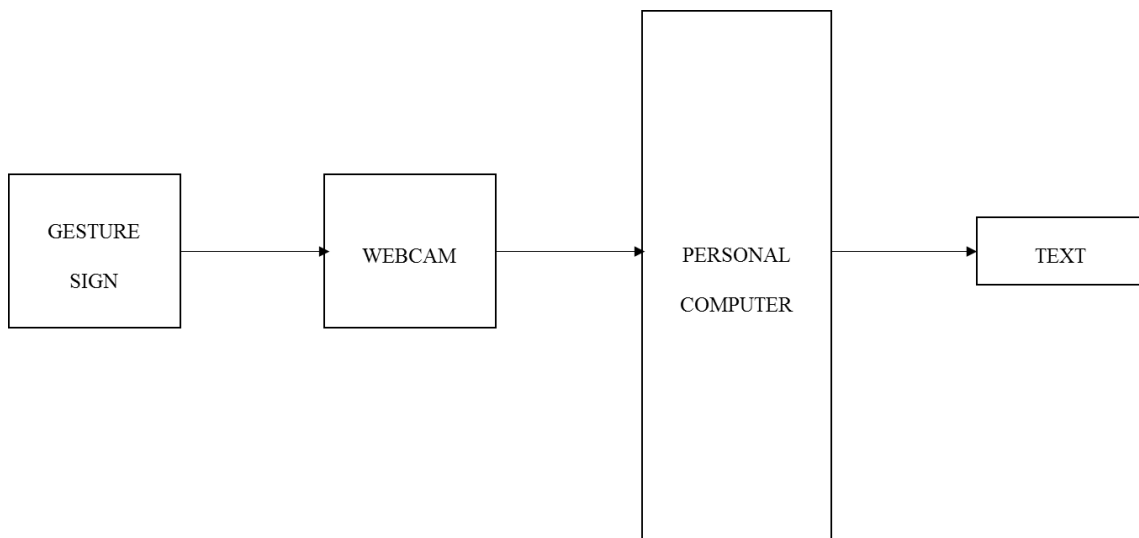
ADVANTAGES:

- It is Effective.
- It is Low cost.
- Automatic character recognition.
- A prototype system to read printed text on hand-held objects for assisting blind persons.
- We have proposed a motion-based method to detect the object of interest, while the blind user simply shakes the object for a couple of seconds.

METHODOLOGY

- Human beings interact with each other either using a natural language channel such as words, writing, or by body language (gestures) e.g. hand gestures, head gestures, facial expression, lip motion and so on.
- The sign language is the basic communication method within hearing disable people. People with hearing disabilities face problems in communicating with other hearing people without a translator.
- For this reason, the implementation of a system that recognize the sign language would have a significant benefit impact on deaf people social live. In this paper, we have proposed a marker-free, visual Indian Sign Language recognition system using image processing, computer vision and neural network methodologies,
- This approach will convert video of daily frequently used full sentences gestureinto a text and then convert it into audio. Identification of hand shape from continuous frames will be done by using series of image processing operations.

BLOCK DIAGRAM



SOFTWARE REQUIREMENTS

- Raspbian OS
- Python
- Open cv



CHAPTER 4

4. DEEP LEARNING

4.1 Overview

According to Andrew Ng[34], The theory of Deep Learning has been around for decades but it has only been in the past few years that it has risen and taken off. In our society we have gone from having a relatively small amount of data to huge amounts to the point that we don't know what to do with them.

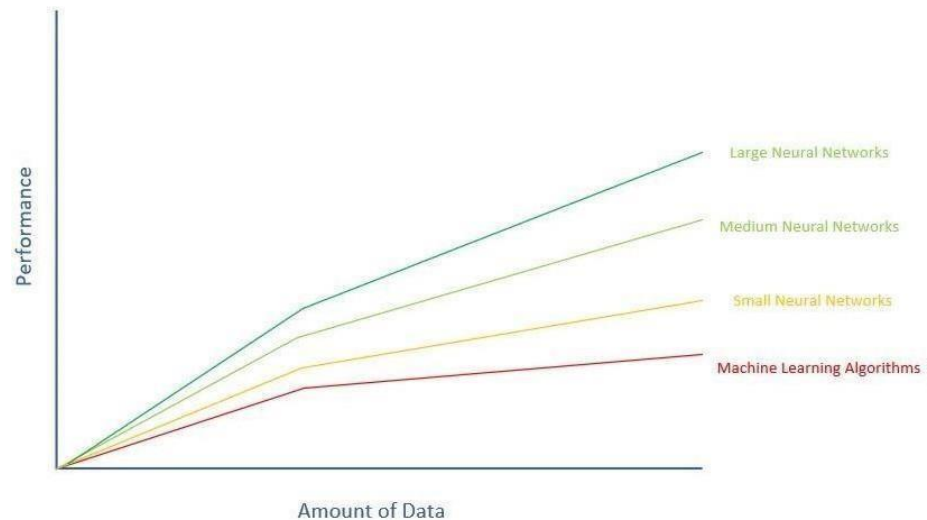


Figure 3.1: Performance of Neural Networks compared to Traditional Learning Algorithms by Andrew Ng[34]

According to Huang et al[26], Deep Learning has gained large popularity within different fields of interest ranging from a theoretical research in academia to practical application in industry. Newly emerging algorithms have expanded applications of Learning Machines to be able to compute large amounts of data with the help of hardware implementations and optimized parallel computational techniques available today. With this Deep Learning Architectures are able to be generalized, customized and configured to advance efficient and accurate models.

Compared to other techniques such as standard Machine Learning algorithms, Deep Learning is a preferred option as performance increases as the data scales[34]. Machine Learning algorithms such as Support Vector Machines, Random Forest and etc, performance plateaus when data increases in these algorithms. See Figure 3.1. Deep

Learning Neural Networks works well for my lung cancer detection project as our dataset is large and can iteratively grow larger as more people infer new data to it which will result in a boost in performance.

In this chapter, the author explores the intuitions made into creation deep learning models.

4.2 Neural Network

A neural network is a model that has been inspired by the brain, the brain consists of nearly 10 billion neurons with 60 trillion connections between each other. A neuron consists of a cell body called the soma where the nucleus is found, many dendrites where input signals are received and transmitted and a synapse which is basically connections between neuron to neuron.[13]

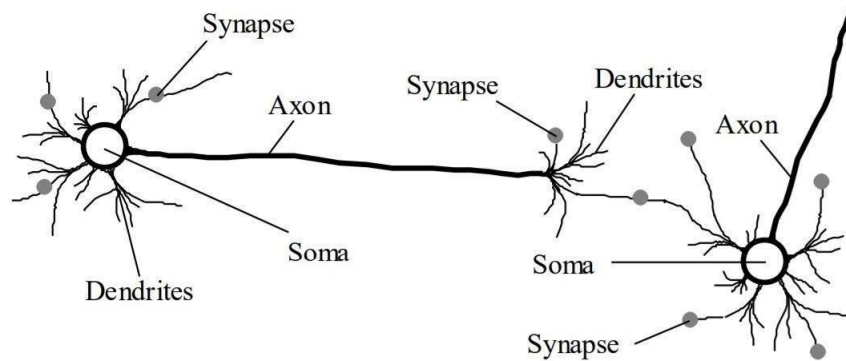


Figure 3.2: A Single Brain Neuron

4.2.1 Artificial Neuron

Like a brain neuron, neural networks consists of an artificial neuron. An artificial neuron consists of a weight to determine the strength of a connection, a linear function that needs to be computed and an activation function that computes the weighted sum of the linear function that is then compared to a threshold value.

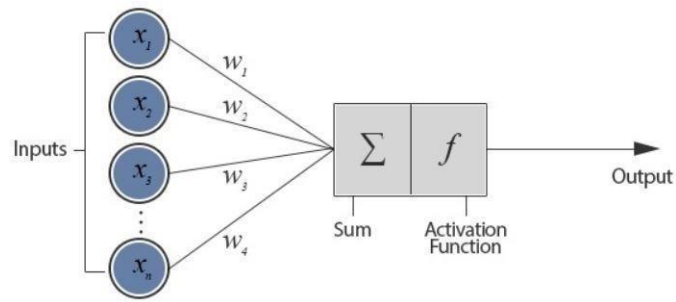


Figure 3.3: An Artificial Neuron also known as a Perceptron [27]

It is also worthy to note that artificial neurons like Figure 3.3 can be modified to contain different linear function algorithms such as Gradient Descent, Logistic Regression, etc and also different activation functions such as ReLU (Rectified Linear Unit) or TanH functions which depends on the problem to solve.

4.2.2 Artificial Neural Network

An Artificial Neural Network is an interconnected architecture where there exists an input layer where input data is placed, a hidden layer(s) where artificial neurons are stacked on on top of each other and an output layer where the prediction or classification is made.

4.2.3 Forward Propagation

Forward propagation is technique in which data moves through from the corresponding input layer, hidden layers and output layer sequentially.

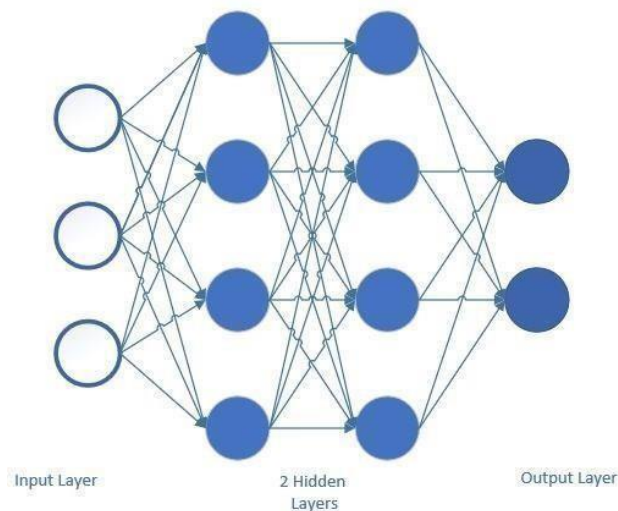


Figure 3.4: An Artificial Neural Network also known as a Multi Layer Perceptron

A hidden layer usually consists of an weight which is used in an optimization algorithm such as Gradient Descent, an activation function such as Sigmoid, TanH, LeakyReLU etc and a Loss Function to calculate the loss or error of the function which we use to back propagate to adjust the weights.

4.2.4 Back Propagation

Back propagation is the opposite of Forward propagation as it feeds the network backwards. Back propagation is used to adjust the weight of the neural network after the errors have been computed by forward propagation algorithms. In later sections of this chapter I detail how Gradient Descent is used in back propagation to adjust the weights of the neural network.

4.2.5 Activation Functions

Activation functions are an important part of a neural network. They allow neural networks to create non linear functions to solve problems. The 3 most used activation functions are Sigmoid, TanH and ReLU. Activation functions allow or stop neurons from firing into the next layer by comparing it into the activation function thresholds.

Activation functions are used both in the forward and backward propagation where in the forward propagation an activation function is used to calculate the loss where the output of a function is compared to the a real number and in backward propagation they are used to update the parameters of the neural network. Figure 3.5 shows commonly used activation functions in a neural network.

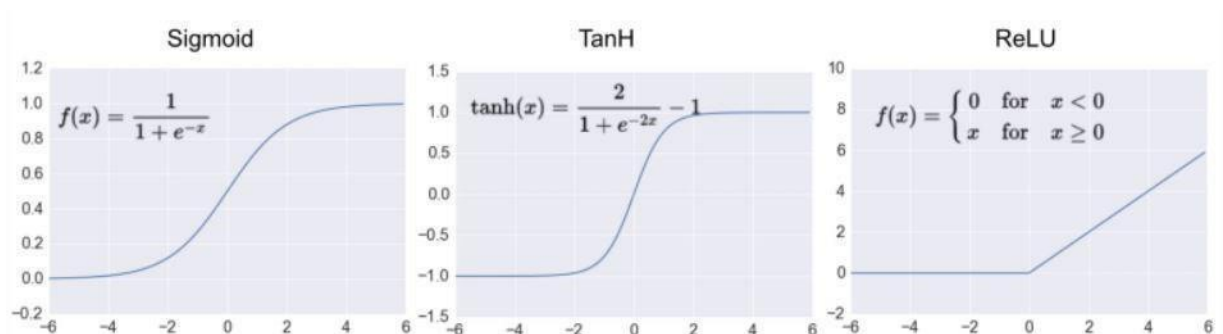
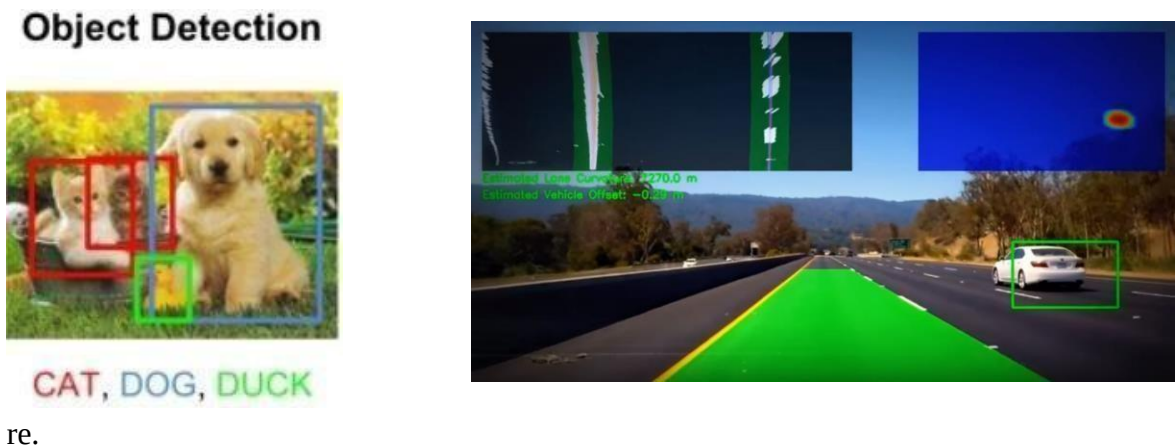


Figure 3.5: Common Activation Functions[11]

4.3 Convolutional Neural Networks

4.3.1 Overview

Convolutional Neural Networks is all about using Deep Learning with Computer Vision. A good way to gain intuition about this is to think about a Neural Network Architecture and how it is applied to visual tasks i.e. Images and Video. Convolutional Neural Networks have allowed us to create Facial Recognition, Object Recognition. Figure 3.6a, Self-Driving Cars Figure 3.6b and mo



(b) Self Driving Car: Lane Finding Detection [31]

(a) Object Detection [12]

Figure 3.6: Convolutional Neural Networks Examples

4.3.2 Architecture

Like a Neural Network, a typical Convolutional Neural Network consists of a multiple hidden layers called a Convolutional Layer where the linear function computes the strided convolutions over an image to extract features. It also consists of a pooling layer that computes another function such as Max Pool or Average Pool to reduce the size of the image in the neuron to speed up the computation. It does it by extracting the features of the neuron image and ignoring the rest, this makes the network more robust. There is also fully connected layer which is like a hidden layer in a neural network where the sum of the outputs of each layer are flattened and where each value is an input to the next layer followed by an activation function and an output[3]. See Figure 3.7.

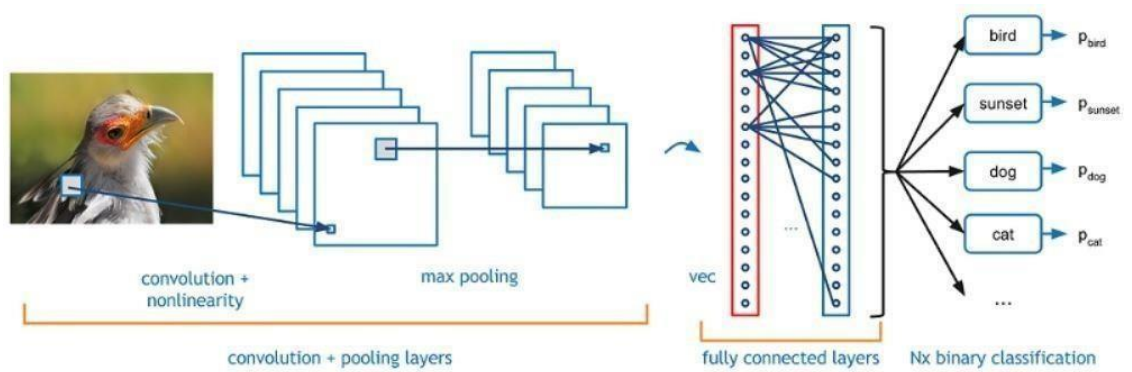


Figure 3.7: A Convolutional Neural Network Architecture Example [35]

4.3.3 Convolutional Layer

In a Convolutional Neural Network, the linear function that is used is called a convolutional layer. Each node in the hidden layer extracts different features by using image processing feature detectors. For example, in the first layer, the first node may extract the horizontal edges of an image, the second node may extract vertical edges and etc. These features are extracted using a kernel. Figure 3.11 shows an example of how stride convolutions work on an image. The bottom is the original image and the top is the output of the convolutions. It is also worth noting that the output of the convolutions reduces the dimension of the original image.

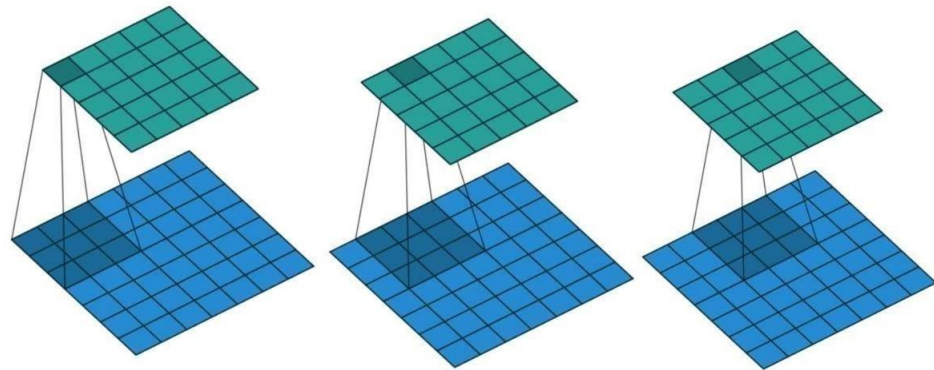


Figure 3.8: Stride Convolution Example [41]

4.3.4 Pooling Layer

The pooling layer happens tends to be computed after the convolutional layer. The reason why pooling is done is to further reduce the dimensions of the convolutional layer and just extract out the features to make the model more robust. There are two types of

pooling done: max pooling and average pooling. Max pooling extracts out the highest pixel value out of a feature while average pooling calculates the average pixel value that has to be extracted. Figure 3.9 shows how the max and average pooling operation works.

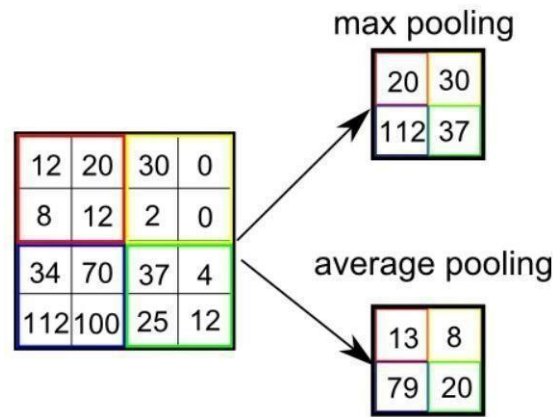
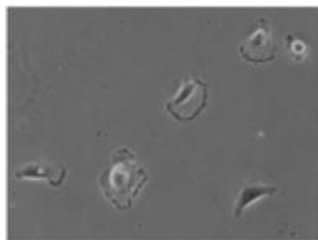


Figure 3.9: Max and Average Pooling Operation [2]

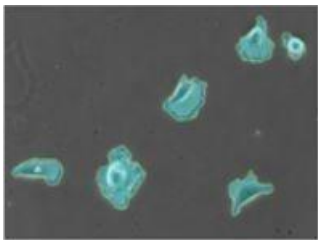
4.4 U-Net Model

4.4.1 Overview

A U-Net model is a different variation of a Convolutional Neural Network. It has been mainly used for biomedical image segmentation but I have also seen it being used in a variety of ways to segment regions of interest. See Figure 3.10. In this section the author details variant nature of U-Net models.



4.4.1.1



U-Net Input Example

(b) U-Net Output ExampleFigure 3.10: U-Net Segmentation

4.4.2 Architecture

Ranneberger et al 2015,[36] mentions that a U-Net model consists of a contracting path (left side). This is the same as standard Convolutional Networks outlined in the last section (3x3 Conv with ReLU and 2x2 Max Pooling). As earlier discussed, the great idea

about this is that the network is able to create features necessary to detect patterns that we are looking to predict(In this case its lung cancer). The problem with this on its own is that convolutional layers down sample the size of the image. Hence data is lost as it feed forwards.

The unique design of the U-Net model lies in its expanding path (right side) which consists of up-convolutions (size 2×2) and merge layers. The up-convolutions take a down sampled sized image and expands the borders to increase the size of the image. This is done before the merge happens, the merge layers then takes output from an earlier convolution layer and concatenate it with the previous upsampled image. This concatenated data or 'Merge' is then fed through another convolution layer. The network is able to send data from an earlier network to a later one through skip connections. This is how the network is able to segment the images after it has found the features. See Figure 3.11, Figure 3.21 and Figure 3.22.

The last layer is a 1×1 convolutional layer to map feature vectors to a desired number of classes.

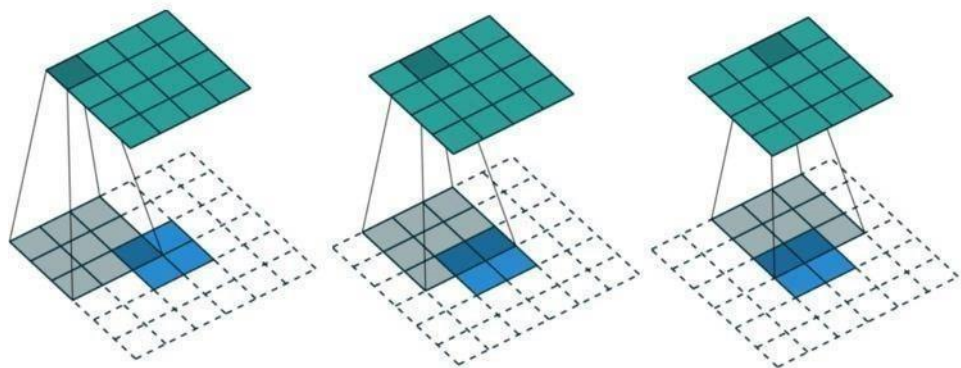


Figure 3.11: Up sampling aka Deconvolution [41]

4.5 Improving Deep Neural Networks

4.5.1 Overview:

This section discusses the decisions made into improving the U-Net model. Deep Learning is a highly iterative process and training is extremely slow and can get stuck on plateaus. i.e. the model is not getting better. I've had this experience many times while training the U-Net for hours and getting bad results. To avoid this, different techniques have been applied on the model that helped tremendously in training and reaching better

convergence. It's also worth mentioning that there are many ways to improve a neural network and that it is constantly evolving as new techniques are created. It is completely infeasible to explain each one. The author has decided to only introduce techniques that are effective and currently state of the art.

4.5.2 Regularization using Dropout

Neurons in a neural network have a high tendency to adapt co-dependency among each other during training time. This means that certain activations become more heavily weighted and results in a network becoming overfit to the training set. To stop the network from overfitting Dropout is used. Figure 3.12

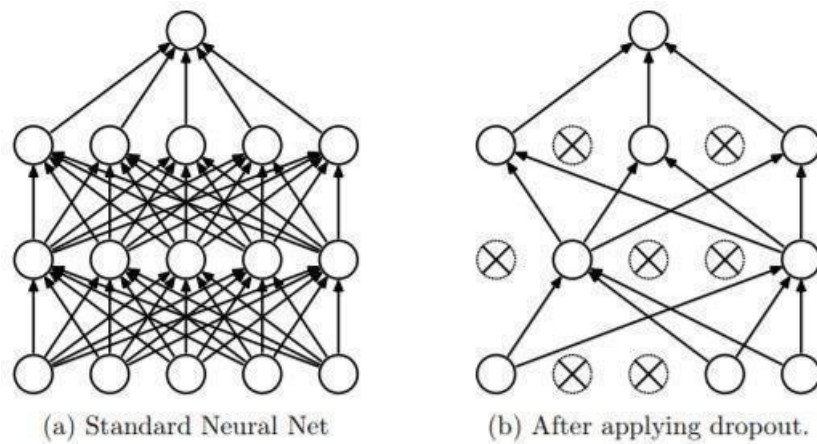


Figure 3.12: Dropout on a Neural Network [40]

According to Hinton et al[40], Dropout is a regularization technique that reduced neuron's ability to be interdependent among one another during training time. This is achieved by shutting off random neurons in the network during time. This forces the network to learn better features that are more robust relative to random subsets of other neurons.

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	1.05

Figure 3.13: Comparing Regularization Technniques [40]

Hinton et al also mentions that compared to the standard regularization techniques such as L2 with learning decay. Test classification error is greatly reduced which means that Dropout is currently state of the art (See

Figure 3.13). The effects of learned features on MNIST benchmark can be seen at Figure 3.14

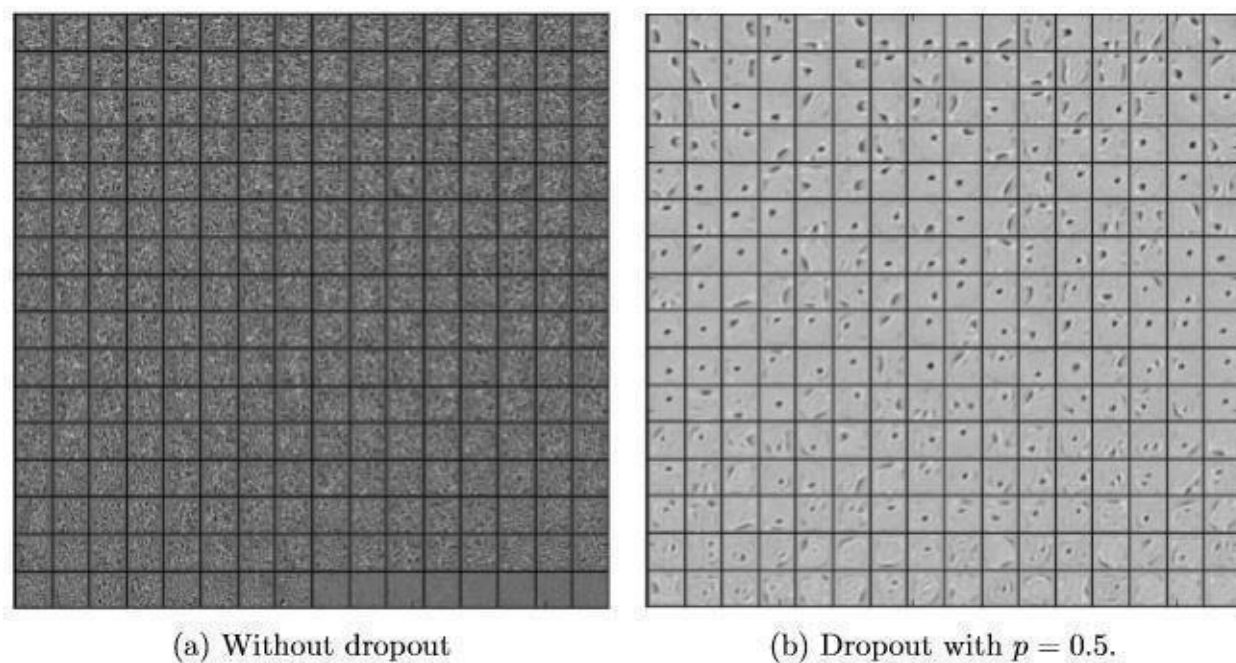


Figure 3.14: Dropout Effects on MNIST [40]

4.5.3 Weight Initializations

One of the main problems encountered in deep learning is vanishing / exploding gradients. This is when the activations become smaller or larger at each layer. To summarise briefly, if the weights in a network are too small, then the signals shrink as it passes through each layer until its too small to be useful conversely if the weights in the network start too large then the signals becomes too large to be useful.

This also has terrible consequence for the parameter updates on the weights as the network back propagates either giving very small updates or very large into the network. This has terrible consequence for large neural networks as it means the larger the network is the more likely it will be to be slow to train and inaccurate.

The two main sophisticated ways to initialize the weights is Xavier [42] and He[25]. Both of these initializations put neurons into a Gaussian or uniform distribution and draws weights from the distribution at a certain mean and variance.

After reading He et al's paper [25], the author decided to use this technique as apparently it performs better than Xavier.(See Figure 3.15)

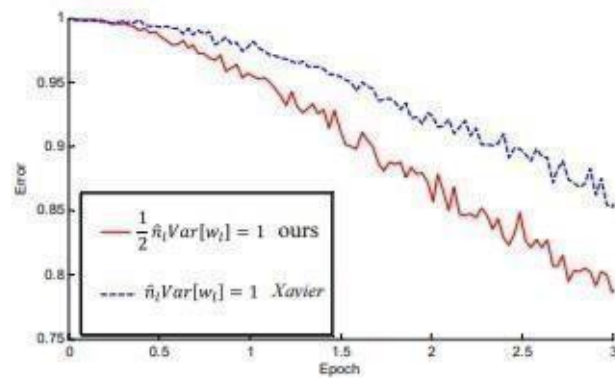


Figure 3.15: He vs Xavier Performance [25]

4.5.4 Gradient Descent

Gradient descent is the learning algorithm used in Deep Learning. It is mainly used in the back propagation step in the training of the neural network. In a neural network the cost function measures the inconsistency between a predicted value and the actual value. The idea behind gradient descent is that we increase the quality of the weights so that the weights minimize a cost function in the neural network.

Gradient descent takes the loss function and calculates a derivative. This is also known as calculating the gradient of a loss function. The idea behind calculating the gradient is that it computes the best direction which the weights must be updated to guarantee a best outcome for minimizing the cost function.

The learning rate parameter then is used to determine the size of the steps we take given the direction. The closer the network gets to convergence i.e. lowest point in the cost function, the more accurate the neural network becomes.

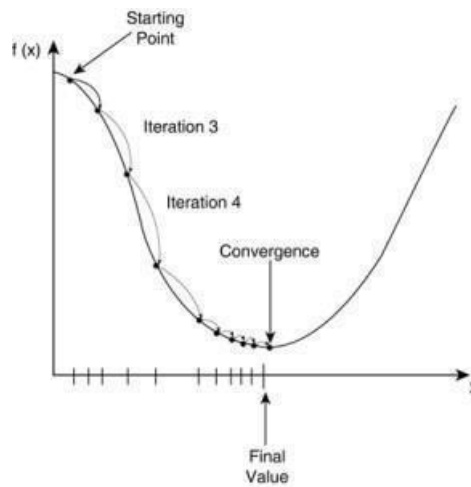


Figure 3.16: Gradient Descent Example

4.5.5 Variants of Gradient Descent

Over time there have been many variations of gradient descent that have improved the algorithm's capability. The main ones are batch gradient descent, stochastic gradient descent and mini-batch gradient descent. Sebastian Ruder wrote a great paper on gradient descent optimization algorithms[37] and according to Ruder the trade-off between the algorithms is the accuracy of the parameter update and the time it takes to perform the update.

3.5.5.1 Batch Gradient Descent

Batch gradient descent computes the gradient of a cost function relating to the parameters for the entire dataset. This basically means that batch gradient descent has to look at the entire dataset before an updating the parameters at a set number of epochs.

An epoch is a pre-defined number of times a neural network will a forward and backward pass during training. It also means predefines how many times the parameters gets updated. 1 epoch = 1 forward pass = 1 backward pass = 1 parameter update on the weights. Listing 3.1 shows the batch gradient descent algorithm.

Since batch gradient descent looks at an entire dataset before updating the parameters, it could mean that training could be really slow and inefficient as it recomputes gradients before updating a parameter. This could also mean that the algorithm could get stuck on a local minima and not improve per epoch. [37]

3.5.5.2 Stochastic Gradient Descent

Stochastic gradient descent on the other hand performs a parameter update on each training instance(x) and its corresponding label(y). Listing 3.2 shows the SGD algorithm.

SGD removes the redundancy problem that batch gradient descent performs which means it could be faster during training. It also means that stochastic gradient descent could jump out of a local minima during training and go towards a better local minima.

The downside to this is that, it could complicate convergence on a local minima as the direction fluctuates per training example.

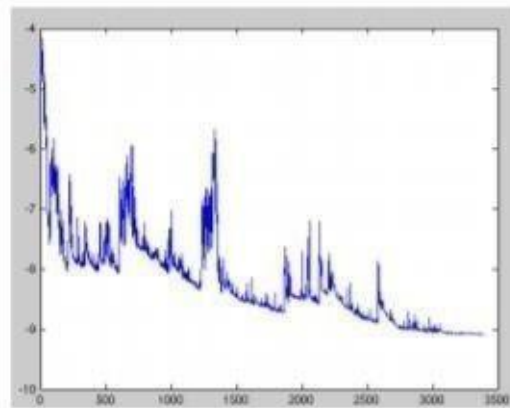


Figure 3.17: Stochastic Gradient Descent Training Fluctuations

3.5.5.3 Mini-Batch Gradient Descent

Mini-batch gradient descent takes both elements of batch gradient descent and stochastic gradient descent. Mini batch gradient descent splits training data into n mini-batches and performs an update for each one. Listing 3.3 shows the mini batch gradient descent algorithm.

Using mini batches means that there would be no redundancy in computing gradients and at the same time less variance in the gradients themselves which leads to faster training and less fluctuations of the cost function during training time.

Mini-batch gradient descent is usually the method of choice for training neural networks.

3.5.5.4 Conclusion

These learning algorithms help in training the neural networks however on their own they cannot guarantee good convergence. In the next section I introduce Optimization algorithms that help gradient descent learning algorithms perform better.

4.5.6 Gradient Descent Optimization Algorithms

3.5.6.1 Overview

Although there are different gradient descent algorithms out there with their own advantages and disadvantages, there are still some issues regarding training the network that has to be solved in a different way. In this section I introduce some ways we can optimize our gradient descent algorithm.

According to Ruder, choosing a learning rate can be difficult as statically predefining the learning rate can either mean slow convergence when its too small or fluctuate around the minimum when the learning rate is too large. To solve this problem there needs to be a dynamic way to choose the learning rate.

Choosing the same learning rate also means that an update applies to each parameter, this can be a problem if we don't want to update the weights that find less frequent features in our data. This means that its not a good idea to update weights that find rare occurring features compared to commonly occurring ones.

The last challenge is minimizing non-convex error functions aka saddle points. Bengio et al[22], mentions that minimizing over saddle points give great difficulty for a neural network to find the global minimum which means learning can be extremely slow and give an impression that a network is stuck on a local minimum. The reason for this is because a saddle point's gradient is extremely small which makes it difficult for the gradient descent algorithm to minimize the cost function. Figure 3.18 shows gradient descent traversing through a saddle point plateau.

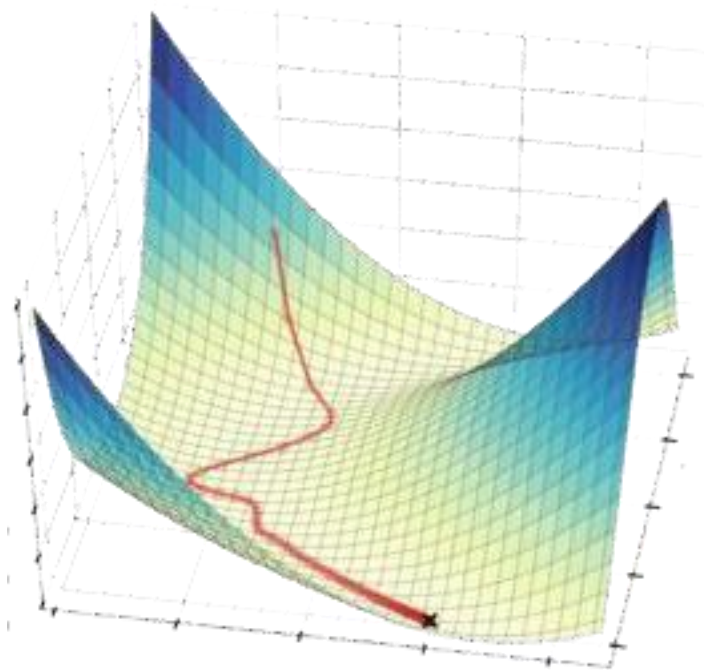


Figure 3.18: Saddle Point

3.5.6.2 Momentum

Due to Stochastic Gradient nature to oscillate between different directions, progressing towards a local / global minimum can be slow. To minimise these oscillations, Momentum is used. Momentum reduces the oscillations by updating the current step to made by adding a fraction of the past step.

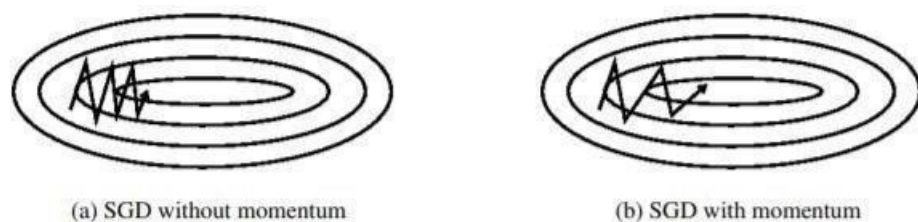


Figure 3.19: Momentum reduced oscillations during training

According to Ruder[37], the momentum term increases for dimensions whose gradients point in the same direction and reduces updates for dimensions whose gradients change directions. This means that a neural network can converge faster by reducing oscillations. The default value for momentum is 0.9.

Sources of Data 3.1

Data Collection :

The primary source of data for this project was the compiled dataset of American Sign Language (ASL) called the ASL Alphabet from Kaggle user Akash [3]. 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. This data is solely of the user Akash gesturing in ASL, with the images taken from his laptop's webcam. These photos were then cropped, rescaled, and labelled for use. 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 preprocessed.

3.2 Data Pre-processing The data preprocessing was done using the PILLOW library, an image processing library, and sklearn.decomposition library, which is useful for its matrix optimization and decomposition functionality.

2 MIE324 Fall 2018 Project Proposal (a) Sample from nonuniform background test set (b) Sample from plain white background test set (c) Sample from a darker test set (d) Sample from plain white background test set

Figure 3: Examples of the signed letter A T from two test sets with differing lighting and background

Image Enhancement: A combination of brightness, contrast, sharpness, and color enhancement was used on the images. For example, the contrast and brightness were changed such that fingers could be distinguished when the image was very dark.

Edge Enhancement: Edge enhancement is an image filtering techniques that makes edges more defined. This is achieved by the increase of contrast in a local region of the image that is detected as an edge. This has the effect of making the border of the hand and fingers, versus the background, much more clear and distinct. This can potentially help the neural network identify the hand and its boundaries.

Image Whitening: ZCA, or image whitening, is a technique that uses the singular value decomposition of a matrix. This algorithm decorrelates the data, and removes the redundant, or obvious, information out of the data. This allows for the neural network to look for more complex and sophisticated relationships, and to uncover the underlying structure of the patterns it is being trained on. The covariance matrix of the image is set to identity, and the mean to zero.

(a) (b) Preprocessed Image Figure 4: Examples of image preprocessing

Data collection and preprocessing

- To collect the dataset it was asked to four users to wear the vision system and perform every gesture for 10 seconds while both cameras were recording in a 640x480 pixel resolution.
- It was requested to the users to perform this process in three different scenarios: indoors, outdoors, and in a green background scenario. For the indoors and outdoors scenarios the users were requested to move around while performing the gestures in order to obtain images with different backgrounds, light sources, and positions. The green background scenario was intended for a data augmentation process, we'll describe later.
- After obtaining the videos, the frames were extracted and reduced to a 125x125 pixel resolution.

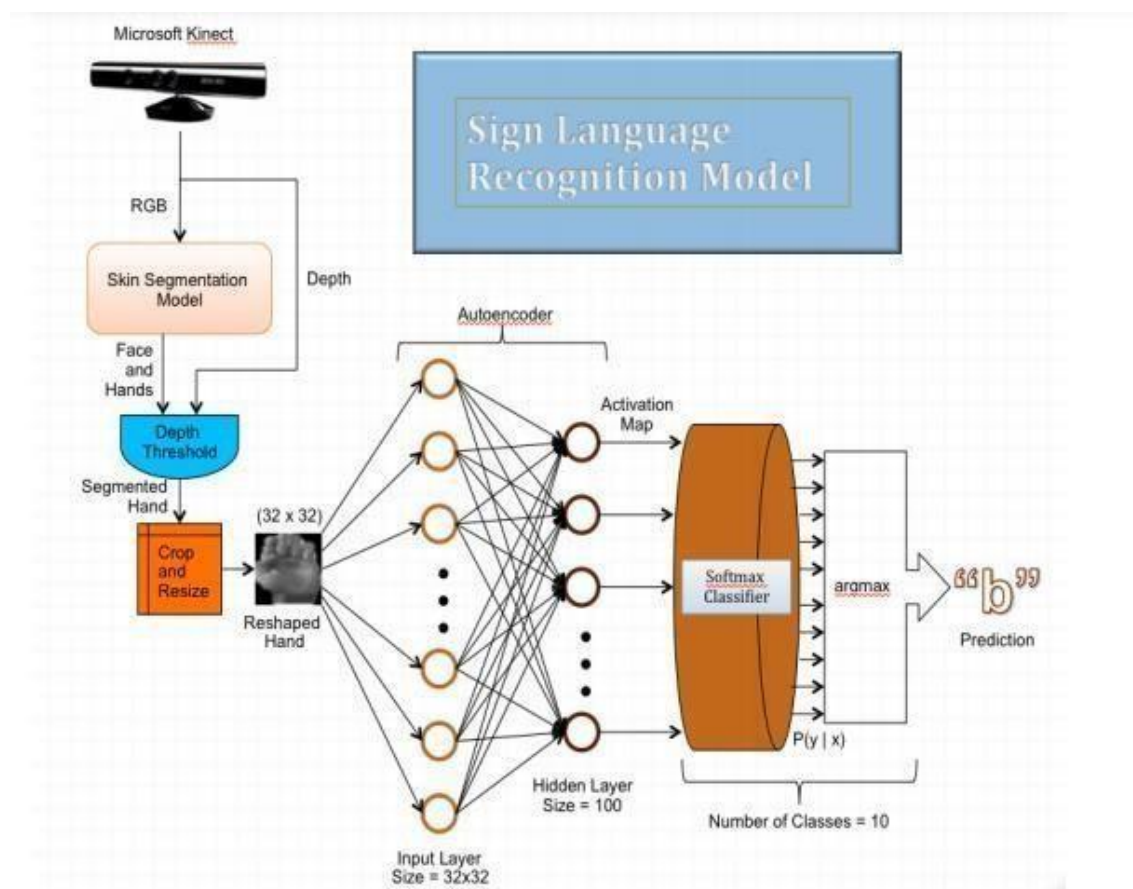
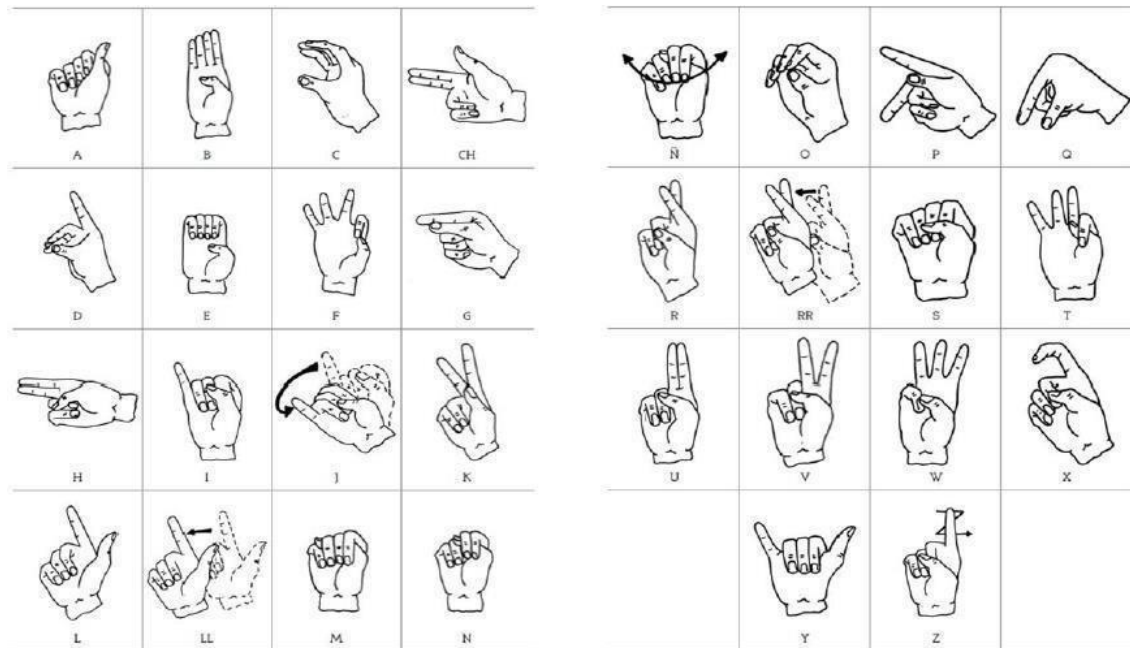


Figure 9: Block diagram summarizing our approach for sign language recognition

DATASET

To develop the first prototype of this system is was used a dataset of 24 static signs from the Panamanian Manual Alphabet.



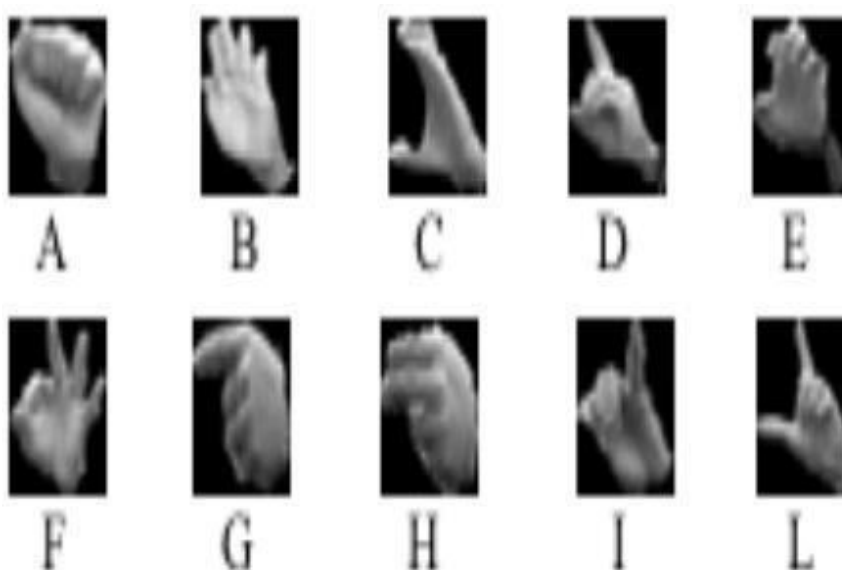
- To collect the dataset it was asked to four users to wear the vision system and perform every gesture for 10 seconds while both cameras were recording in a 640x480 pixel resolution.
- It was requested to the users to perform this process in three different scenarios: indoors, outdoors, and in a green background scenario. For the indoors and outdoors scenarios the users were requested to move around while performing the gestures in order to obtain images with different backgrounds, light sources, and positions. The green background scenario was intended for a data augmentation process, we'll describe later.
- After obtaining the videos, the frames were extracted and reduced to a 125x125 pixel resolution.

Machine Learning on Feature Vectors The accuracies we reported in the previous subsection corresponds to the best results we obtained for each of the feature vectors. But before we obtained those best results, we explored the following algorithms on the obtained feature vectors. Multiclass SVM with linear kernel was used with almost every feature vector. Overall the following approaches were tried :-

- Support Vector Machines :- Multiclass SVMs were tried on all the feature vectors. Results obtained with linear kernel and four fold Cross Validated accuracies are reported for all feature vectors. The confusion matrices shown in the results sections below correspond to the different techniques tried using linear kernel MultiClass SVMs. The best accuracies were observed for this algorithm. Our try with 'rbf' kernel failed miserably on HOG feature vectors as only 4.76% accuracy was observed.

- Random Forest :- Since this was a 26 class problem, we tried our luck at Random Forest with HOG feature vectors on the compressed images. It fell a little short of the Multiclass SVM with 46.45% 4 fold CV accuracy. Our interpretation is that implementation of decision trees in python is monothetic which may have led to formation of rectangular regions whereas the actual boundaries may not have been rectangular.

- Hierarchical Classification :- One of the major reasons for the comparatively lower accuracy is the large number of classes(26). So one approach we tried was breaking the classification problem into multiple levels or hierarchical classification . First we train a linear kernel SVM model to classify alphabets as one handed or two handed. 7 This model performed with 95% accuracy. Then we trained linear kernel Multiclass SVM models to classify the one handed alphabets(56% accuracy) and two handed alphabets(60% accuracy) and then put the system together. An alphabet is first classified as one handed or two handed, and then depending on the classification, is put into the corresponding model and given a label. Even though the individual models performed better than the direct multiclass SVM on HOG features, overall the performance was nearly the same and four fold CV accuracy of 3.23% was observed.



CHAPTER 5

OCR ALGORITHM

Optical character recognition is the translation of optically scanned bitmaps of printed or written text into digitally editable data files. OCRs developed for many world languages are already under efficient use.

1.1 Computer Vision

Human intelligence is described as the capability to make decisions based on information which is incomplete and noisy. This is the ability which makes human being the most superior amongst all living creatures in the world.

There are five senses that provide information to humans for making everyday decisions and out of these the senses of hearing and vision are the sharpest of all. The auditory sense helps us recognize sounds and classify them. It is this sense which tells us that the person on the phone is a friend because his voice is *recognizable*. We can differentiate between an endless variety of sounds, voices, utterances and put them in exactly the slots they belong to, animal sounds, musical notes, wind swishing, the footsteps of a family member, all are within recognition range of a person with a normal sense of hearing.

The other one and the one more profound is the human vision which allows us to identify a known person in a crowd of unknowns merely by casting a cursory glance at him. It allows us to pick an object that belongs to us from a number of those looking exactly as ours, and by being able to recognize a miss-spelt word in a sentence and unconsciously correct it. The fact is that the human mind is capable of identifying an image on features spontaneously determined and not predefined or predetermined.

With the development of technology these human processes are imitated to create intelligent machines, so much for the immense growth of robotics and intelligent decision making systems and yet the work done so far is not comparable to any natural involuntary human action or process. The hindrance however is that it is not practically possible to imitate all the functions of the human mind and make computer vision as efficient and accurate as the human eye but even though such a possibility may be remote, efforts are consistently being made to bring them as close to it as possible.

Artificial Intelligence is a broad field of computer science taking into account a number of other disciplines that form the bulk of its study. One of the most popular definitions of

Artificial Intelligence (AI) was given by Elaine Rich and Kevin Knight as, “Artificial Intelligence (AI) is the study of how to make computers do things which, at the moment, people do better” [20]. One important branch of Artificial intelligence is Computer vision, as mentioned in Figure 1.1 that shows few sub-branches of Computer Vision, the area of research which aims to imitate human vision and forms the basis of all image acquisition, its processing, document understanding and recognition. Computer vision relies on a solid understanding of the physical process of image formation to obtain simple inference from individual pixel values like shape of the object and to recognize objects using geometric information or probabilistic techniques

In its own turn document understanding is a vast and difficult area for the focus of research today lies in being able to make content based searches which hope to allow machines to look beyond the key words, headings or merely topics to find a piece of information. A far more streamlined field of Document Recognition and understanding is Optical Character Recognition which attempts to identify a single character from an optically read text image as a part of a word that can be then used to process further information on. The area gains rising significance as more and more information each day needs to stored processed and retrieved rather than being keyed in from an already present printed or handwritten source.



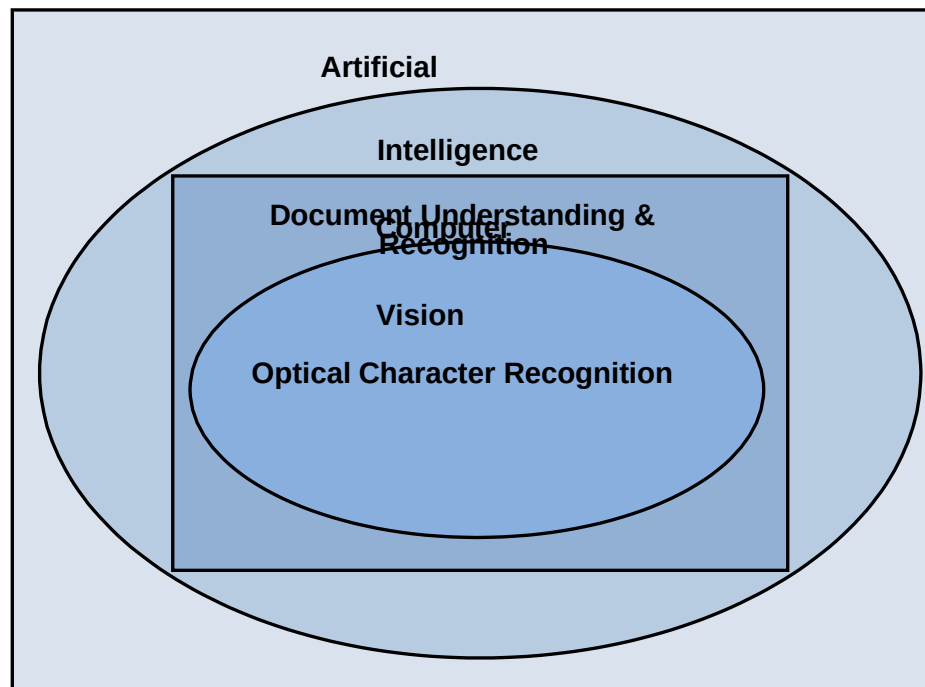


Figure 1.1: Sub-fields of Artificial Intelligence

1.2 Character Recognition

Character recognition is a sub-field of pattern recognition in which images of characters from a text image are recognized and as a result of recognition respective character codes are returned, these when rendered give the text in the image.

The problem of character recognition is the problem of automatic recognition of raster images as being letters, digits or some other symbol and it is like any other problem in computer vision

Character recognition is further classified into two types according to the manner in which input is provided to the recognition engine. Considering figure 1.2 which shows the classification hierarchy of character recognition.

The two types of character recognition are:

- a. On-line character recognition
- b. Off-line character Recognition

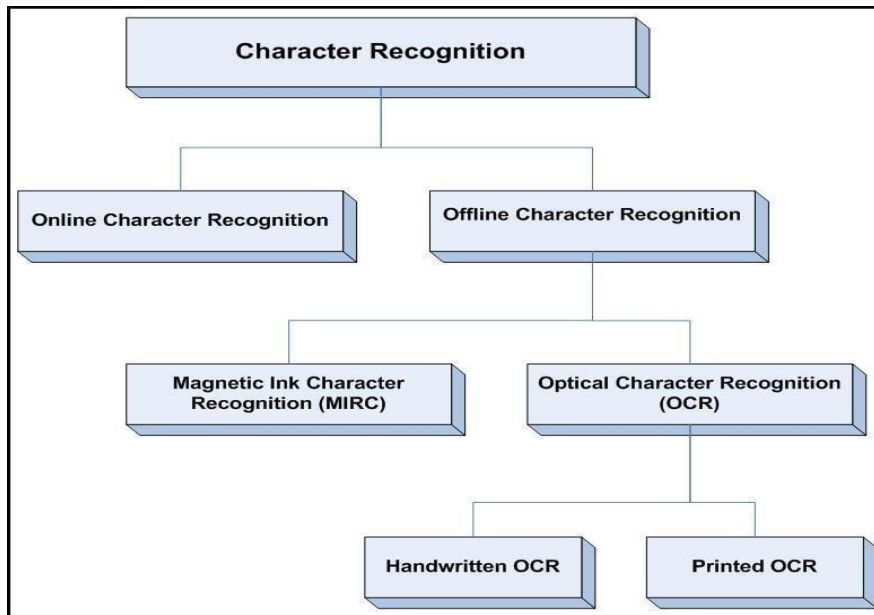


Figure 1.2: Classification of Character Recognition

1.2.1 Online Character Recognition

Online character recognition systems deal with character recognition in real time. The process involves a dynamic procedure using special sensor based equipment that can capture input from a transducer while text is being written on a pressure sensitive, electrostatic or electromagnetic digitizing tablet. The input text is automatically converted with the help of a recognition algorithm to a series of electronic signals which can be stored for further processing in the form of letter codes. The recognition system functions on the basis of the x and y coordinates generated in a temporal sequence by the pen tip movements as they create recognizable patterns on a special digitizer as the text is written.

1.2.2 Offline Character Recognition

There is a major difference in the input system of off-line and on-line character recognition which influences the design, architecture and methodologies employed to develop recognition systems for the two. In online recognition the input data is available in the form of a temporal sequence or real time text generated on a sensory device thus providing time sequence contextual information. On the contrary in an off line system the actual recognition begins after the data has been written down as it does not require real time contextual information.

Offline character recognition is further classified in two types according to the input provided to the system for recognition of characters. These are,

- i. Magnetic ink Character Recognition (MICR)
- ii. Optical Character Recognition (OCR)

1.2.3 Magnetic ink Character Recognition (MICR)

MICR is a unique technology that relies on recognizing text which has been printed in special fonts with magnetic ink usually containing iron oxide. As the machine prepares to read the code the printed characters become magnetized on the paper with the North Pole on the right of each MICR character creating recognizable waveforms and patterns that are captured and used for further processing. The reading device is comparable to a tape recorder head that recognizes the wave patterns of sound recorded on the magnetic tape. The system has been in efficient use for a long time in banks around the world to process checks as results give high accuracy rates with relatively low chances of error.

There are special fonts for MICR, the most common fonts being E-13B and CMC-7.

1.2.4 Optical Character Recognition (OCR)

Optical Character Recognition or OCR is the text recognition system that allows hard copies of written or printed text to be rendered into editable, soft copy versions. It is the translation of optically scanned bitmaps of printed or written text into digitally editable data files. An OCR facilitates the conversion of geometric source object into a digitally representable character in ASCII or Unicode scheme of digital character representation [37].

Many a times we want to have an editable copy of the text which we have in the form of a hard copy like a fax or pages from a book or a magazine. The system employs the use of an optical input device usually a digital camera or a scanner which pass the captured images to a recognition system that after passing it through a number of processes convert it to a soft copy like an MS Word document.

When we scan a sheet of paper we reformat it from hard copy to a soft copy, which we save as an image. The image can be handled as a whole but its text cannot be manipulated separately. In order to be able to do so, we need to ask the computer to recognize the text as such and to let us manipulate it as if it was a text in a word document. The OCR application does that; it recognizes the characters and makes the text editable and searchable, which is what we need. The technology has also enabled such materials to be stored using much less

storage space than the hard copy materials. OCR technology has made a huge impact on the way information is stored, shared and communicated.

OCRs are of two types,

- i. OCRs for recognizing printed characters
- ii. OCRs for recognizing hand-written text.

OCRs meant for printed text recognition are generally more accurate and reliable because the characters belong to standard font files and it is relatively easier to match images with the ones present in the existing library. As far as hand writing recognition is concerned the vast variety of human writing styles and customs make the recognition task more challenging. Today we have OCRs for printed text in Latin script as an everyday tool in offices while an OCR for hand writing is still in the research and development stage to have more result accuracy.

Optical Character Recognition (OCR) is one of the most common and useful applications of machine vision, which is a sub-class of artificial intelligence, and has long been a topic of research, recently gaining even more popularity with the development of prototype digital libraries which imply the electronic rendering of paper or film based documents through an imaging process.

1.3 History of OCR

The history of OCR dates back to the early 1950s with the invention of *Gismo* a machine that could translate printed messages into machine codes for computer processing. The product had been a combined effort of David Shepard, a cryptanalyst at (Armed Forces Security Agency) AFSA and Harvey Cook. The successful achievement was then followed by the construction of the world's first OCR system, also by David Shepard under his Intelligent Machines Research Corporation. Shepard's customers included The Readers' Digest, Standard Oil Company of California for making credit card imprints, Ohio Bell Telephone Company for a bill stub reader and The U.S. Air force for reading and transmitting teletype messages.

Another success came in the area of postal number recognition, which although crude in its inception stage became more refined with advancement in technology. By 1965 Jacob Rainbow an American inventor had patented an OCR for sorting mails which was then used by the U.S. Postal Service. The OCR has since become an interesting field of research posing numerous complexities and offering unique possibilities in the areas of Artificial Intelligence and Computer Vision. As advancements were made and new challenges were undertaken it became more and more clear that the scope of such an undertaking, though attractive would be daunting and adventurous.

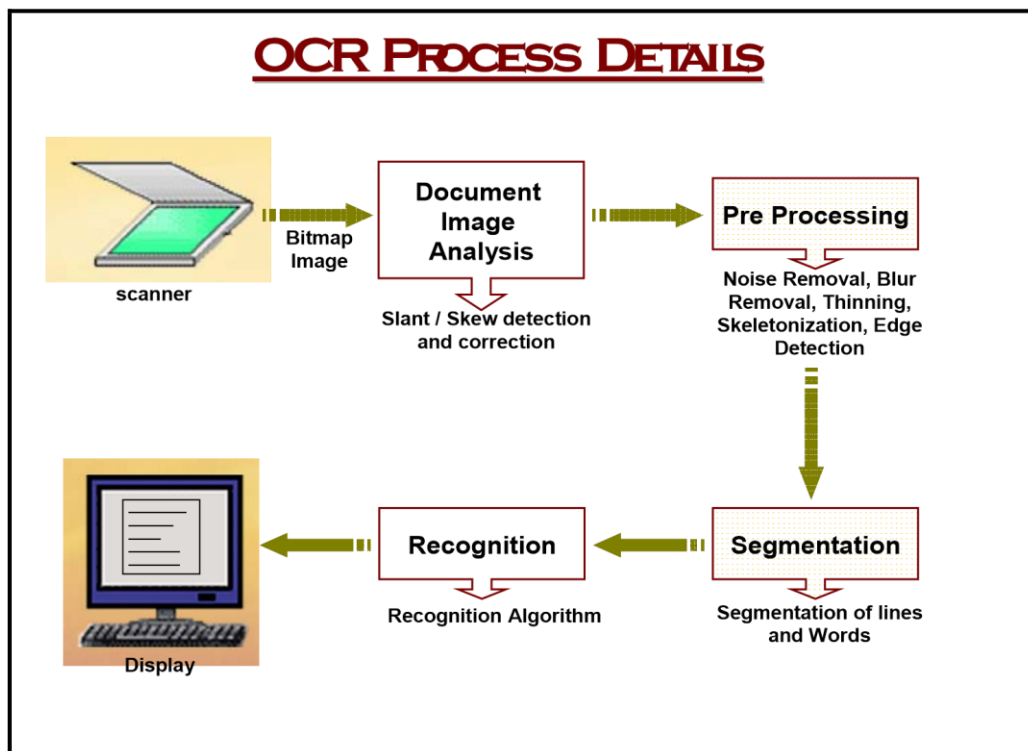


Figure 1.3: OCR Processes

1.4.1 Scanning

A flat-bed scanner is usually used at 300dpi which converts the printed material on the page being scanned into a bitmap image.

1.4.2 Document Image Analysis

The bitmap image of the text is analyzed for the presence of skew or slant and consequently these are removed. Quite a lot of printed literature has combinations of text and tables, graphs and other forms of illustrations. It is therefore important that

the text area is identified separately from the other images and could be localized and extracted.

1.4.3 Pre-processing

In this phase several processes are applied to the text image like noise and blur removal, binarization, thinning, skeletonization, edge detection and some morphological processes, so as to get an OCR ready image of the text region which is free from noise and blur.

1.4.4 Segmentation

If the whole image consists of text only, the image is first segmented into separate lines of text. These lines are then segmented into words and finally words into individual letters. Once the individual letters are identified, localized and segmented out in a text image it becomes a matter of choice of recognition algorithm to get the text in the image into a text processor.

1.4.5 Recognition

This is the most vital phase in which recognition algorithm is applied to the images present in the text image segmented at the character level. As a result of recognition character code corresponding to its image is returned by the system which is then passed to a word processor to be displayed on the screen where it can be edited, modified and saved in a new file format.

CHAPTER 6

PYTHON :

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.^[22]

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

Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Using third-party tools, such as Py2exe or Pyinstaller,^[24] Python code can be packaged into stand-alone executable programs for some of the most popular operating systems, allowing for the distribution of Python-based software for use on those environments without requiring the installation of a Python interpreter.

CPython, the reference implementation of Python, is free and open-source software and has a community-based development model, as do nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation.

Python expressions are similar to languages such as C and Java:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs (see Mathematics for details). Python also added the `**` operator for exponentiation.
- In Python, `==` compares by value, in contrast to Java, where it compares by reference. (Value comparisons in Java use the `equals ()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). Comparisons may be chained, for example `a <= b <= c`.
- Python uses the words `and`, `or`, `not` for its Boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.

- Python has a type of expression termed a *list comprehension*. Python 2.4 extended list comprehensions into a more general expression termed a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be a single expression.
- Conditional expressions in Python are written as `x if c else y`^[54] (different in order of operands from the `?:` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The parentheses around the tuple are optional in some contexts. Tuples can appear on the left side of an equal sign; hence a statement like `x, y = y, x` can be used to swap two variables.
- Python has a "string format" operator `%`. This functions analogous to printf format strings in C, e.g. `"foo=%s bar=%d" % ("blah", 2)` evaluates to `mfool=blah bar=2`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, `"foo={0} bar={1}".format("blah", 2)`.
- Python has various kinds of string literals:
 - Strings delimited by single or double quotation marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quotation marks and double quotation marks function identically. Both kinds of string use the backslash (`\`) as an escape character and there is no implicit string interpolation such as `"$foo"`.
 - Triple-quoted strings, which begin and end with a series of three single or double quotation marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
 - Raw string varieties, denoted by prefixing the string literal with an `r`. No escape sequences are interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare `@`-quoting" in C#.

- Python has index and slice expressions on lists, denoted as `a[key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to some duplication of functionality. For example:

- List comprehensions vs. `for`-loops
- Conditional expressions vs. `if` blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a = 1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c = 1) { ... }` is valid C code but `if c = 1: ...` causes a syntax error in Python.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes themselves are

instances of the `metaclass type` (itself an instance of itself), allowing metaprogramming and reflection.

Prior to version 3.0, Python had two kinds of classes: "old-style" and "new-style".^[56] Old-style classes were eliminated in Python 3.0, making all classes new-style. In versions between 2.2 and 3.0, both kinds of classes could be used. The syntax of both styles is the same, the difference being whether the class `object` is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of `type`).

Summary of Python 3's built-in types		
Type	Description	Syntax example
<code>str</code>	A <u>character string</u> : an <u>immutable</u> sequence of Unicode code points.	'Wikipedia' "Wikipedia" """Spanning multipl lines"""
<code>bytearray</code>	A mutable sequence of <u>bytes</u> .	bytearray(b'Some ASCII') bytearray(b"Some ASCII") bytearray([119, 105, 107, 105])
<code>bytes</code>	An immutable sequence of bytes.	b'Some ASCII' b"Some ASCII"

		<code>bytes([119, 105, 107, 105])</code>
<code>list</code>	Mutable <u>list</u> , can contain mixed types.	<code>[4.0, 'string', True]</code>
<code>tuple</code>	Immutable, can contain mixed types.	<code>(4.0, 'string', True)</code>
<code>set</code> , <code>frozenset</code>	Unordered <u>set</u> , contains no duplicates. A <code>frozenset</code> is immutable. Either can contain mixed types as long as they are hashable.	<code>{4.0, 'string', True}</code> <code>frozenset([4.0, 'string', True])</code>
<code>dict</code>	A mutable <u>associative array</u> (or dictionary) of key and value pairs. Can contain mixed types (keys and values). Keys must be a hashable type.	<code>{'key1': 1.0, 3: False}</code>
<code>int</code>	An immutable <u>integer</u> of unlimited magnitude. ^[57]	<code>42</code>
<code>float</code>	An immutable <u>floating point</u> number (system-defined precision).	<code>3.1415927</code>
<code>complex</code>	An immutable <u>complex number</u> with real and imaginary parts.	<code>3+2.7j</code>
<code>bool</code>	An immutable <u>Boolean value</u> .	<code>True</code> <code>False</code>

ellipsis	An <u>ellipsis</u> placeholder to be used as an index in <u>NumPy</u> arrays.	...
----------	---	-----

WEBCAM: -

A **webcam** is a video camera which feeds its images in real time to a computer or computer network, often via USB, Ethernet or Wi-Fi. Their most popular use is the establishment of video links, permitting computers to act as videophones or videoconference stations. This common use as a video camera for the World Wide Web gave the webcam its name. Other popular uses include security surveillance and computer vision. Webcams are known for their low manufacturing cost and flexibility, making them the lowest cost form of video telephony. They have also become a source of security and privacy issues, as some built-in webcams can be remotely activated via spyware.

LENS:

Webcams typically include a lens, an image sensor, support electronics, and may also include a microphone for sound. Various lenses are available, the most common in consumer-grade webcams being a plastic lens that can be screwed in and out to focus the camera. Fixed focus lenses, which have no provision for adjustment, are also available. As a camera system's depth of field is greater for small image formats and is greater for lenses with a large f-number (small aperture), the systems used in webcams have a sufficiently large depth of field that the use of a fixed focus lens does not impact image sharpness to a great extent.

Image sensors can be CMOS or CCD, the former being dominant for low-cost cameras, but CCD cameras do not necessarily outperform CMOS-based cameras in the low cost price range. Most consumer webcams are capable of providing VGA resolution video at a frame rate of 30 frames per second. Many newer devices can produce video in multi-megapixel resolutions, and a few can run at high frame rates such as the PlayStation Eye, which can produce 320×240 video at 120 frames per second.

Support electronics read the image from the sensor and transmit it to the host computer. The camera pictured to the right, for example, uses a Sonix SN9C101 to transmit its image over USB. Typically, each frame is transmitted uncompressed in RGB or YUV or compressed as JPEG. Some cameras, such as mobile phone cameras, use a CMOS sensor with supporting electronics "on die", i.e. the sensor and the support electronics are built on a single silicochip to save space and manufacturing costs.

PORTABILITY:

Most webcams feature built-in microphones to make video calling and videoconferencing more convenient. The USB video device class (UVC) specification allows for interconnectivity of webcams to computers without the need for proprietary device drivers. Microsoft Windows XP SP2, Linux and Mac OS X (since October 2005) have UVC support built in and do not require extra device drivers, although they are often installed to add additional features.

FEATURES (LOGITECH WEBCAM C100):-

- Plug-and-play setup (UVC)
- Video capture: Up to 640 x 480 pixels
- Photos: Up to 1.3 megapixels (software enhanced)
- Frame rate: Up to 30 frames per second (with recommended system)
- Hi-Speed USB 2.0 certified
- Fixed focus
- Universal clip fits notebooks, LCD or CRT moni



CHAPTER 7

RESULT AND DISCUSSION:

Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them. The system didn't perform too well but it was demonstrated that it can be built a first-person sign language translation system using only cameras and convolutional neural networks. It was observed that the model tends to confuse several signs with each other, as U and W. But thinking a bit about it, maybe it doesn't need to have a perfect performance since using an orthography corrector or a word predictor would increase the translation accuracy.

CONCLUSIONS AND FUTURE WORK:

In this project, we proposed an idea for feasible communication between hearing impaired and normal person with the help of deep learning and machine learning approach.

This proposed work ensures the accuracy of 91.22% using SVM, 95.71% using Random Forest, 87.54% using KNN, 98.74% using CNN and 99.79 using AlexNet.

There is ever the sounding challenge to develop a sign language system in data the collection remains invariant of the unconstraint environment. This project can be extended to the real time data.

SOURCE CODE:

```
import cv2
import numpy as np
import math
cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    cv2.imshow('Thresholded', thresh1)
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
        cv2.CHAIN_APPROX_NONE)
    max_area = -1
    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
        if(area>max_area):
            max_area=area
            ci=i
    cnt=contours[ci]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
```

```

cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
for i in range(defects.shape[0]):
    s,e,f,d = defects[i,0]
    start = tuple(cnt[s][0])
    end = tuple(cnt[e][0])
    far = tuple(cnt[f][0])
    a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
    b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
    c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
    angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
    if angle <= 90:
        count_defects += 1
        cv2.circle(crop_img,far,1,[0,0,255],-1)
        #dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        #cv2.circle(crop_img,far,5,[0,0,255],-1)
if count_defects == 1:
    cv2.putText(img,"I'M kavin", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
elif count_defects == 2:
    str = "i am fine"
    cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
elif count_defects == 3:
    cv2.putText(img,"how are you (:", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
elif count_defects == 4:
    cv2.putText(img,"whats the matter", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
else:
    cv2.putText(img,"hand gesture", (50,50),\
        cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
#cv2.imshow('drawing', drawing)
#cv2.imshow('end', crop_img)
cv2.imshow('Gesture', img)
all_img = np.hstack((drawing, crop_img))
cv2.imshow('Contours', all_img)
k = cv2.waitKey(10)

```

```
if k == 27:  
    break
```

REFERENCE PAPERS :

- [1] Farnaz D. Notash and Elahe Elhamki. “Comparing loneliness, depression and stress in students with hearingimpaired and normal students studying in secondary schools of Tabriz”. In: International Journal of Humanities and Cultural Studies February 2016 Special Issue (2016). issn: 2356-5926.
- [2] “The Cognitive, Psychological and Cultural Impact of Communication Barrier on Deaf Adults”. In: Journal of Communication Disorders, Deaf Studies Hearing Aids 4 (2 2016). doi: 10.4172/2375-4427.1000164.
- [3] Akash. ASL Alphabet. url: <https://www.kaggle.com/grassknotted/asl-alphabet>. (accessed: 24.10.2018).
- [4] Vivek Bheda and Dianna Radpour. “Using Deep Convolutional Networks for Gesture Recognition in American Sign Language”. In: CoRR abs/1710.06836 (2017). arXiv: 1710.06836. url: <http://arxiv.org/abs/1710.06836>.

gesture.py x main.py x

```
1 import cv2
2 import numpy as np
3 import math
4 cap = cv2.VideoCapture(0)
5 while(cap.isOpened()):
6     ret, img = cap.read()
7     cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
8     crop_img = img[100:300, 100:300]
9     grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
10    value = (35, 35)
11    blurred = cv2.GaussianBlur(grey, value, 0)
12    _, thresh1 = cv2.threshold(blurred, 127, 255,
13                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
14    cv2.imshow('Thresholded', thresh1)
15    contours, hierarchy = cv2.findContours(thresh1.copy(), cv2.RETR_TREE,
16                                           cv2.CHAIN_APPROX_NONE)
```

```
while (cap.isOpened()) > for i in range(defects.shape[0])
```

pytho

gesture.py x

main.py x

```
10 value = (35, 35)
11 blurred = cv2.GaussianBlur(grey, value, 0)
12 _, thresh1 = cv2.threshold(blurred, 127, 255,
13                             cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
14 cv2.imshow('Thresholded', thresh1)
15 contours, hierarchy = cv2.findContours(thresh1.copy(), cv2.RETR_TREE,
16                                         cv2.CHAIN_APPROX_NONE)
17 max_area = -1
18 for i in range(len(contours)):
19     cnt=contours[i]
20     area = cv2.contourArea(cnt)
21     if(area>max_area):
22         max_area=area
23         ci=i
24 cnt=contours[ci]
25 x,y,w,h = cv2.boundingRect(cnt)
```

```
while (cap.isOpened()) > for i in range(defects.shape[0])
```

ytho

gesture.py × main.py ×

```
25 x, y, w, h = cv2.boundingRect(cnt)
26 cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)
27 hull = cv2.convexHull(cnt)
28 drawing = np.zeros(crop_img.shape, np.uint8)
29 cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
30 cv2.drawContours(drawing, [hull], 0, (0, 0, 255), 0)
31 hull = cv2.convexHull(cnt, returnPoints=False)
32 defects = cv2.convexityDefects(cnt, hull)
33 count_defects = 0
34 cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 3)
35 for i in range(defects.shape[0]):
36     s, e, f, d = defects[i, 0]
37     start = tuple(cnt[s][0])
38     end = tuple(cnt[e][0])
39     far = tuple(cnt[f][0])
40     a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
```

```
while (cap.isOpened()) > for i in range(defects.shape[0])
```

gesture.py x main.py x

```
37 start = tuple(cnt[s][0])
38 end = tuple(cnt[e][0])
39 far = tuple(cnt[f][0])
40 a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
41 b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
42 c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
43 angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
44 if angle <= 90:
45     count_defects += 1
46     cv2.circle(crop_img, far, 1, [0, 0, 255], -1)
47     #dist = cv2.pointPolygonTest(cnt, far, True)
48     cv2.line(crop_img, start, end, [0, 255, 0], 2)
49     #cv2.circle(crop_img, far, 5, [0, 0, 255], -1)
50 if count_defects == 1:
51     cv2.putText(img, "I'M Teja", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
52 elif count_defects == 2:
```

while (cap.isOpened()) > for i in range(defects.shape[0])

gesture.py × main.py ×

```
46     cv2.circle(crop_img, far, 1, [0, 0, 255], -1)
47     #dist = cv2.pointPolygonTest(cnt, far, True)
48     cv2.line(crop_img, start, end, [0, 255, 0], 2)
49     #cv2.circle(crop_img, far, 5, [0, 0, 255], -1)
50     if count_defects == 1:
51         cv2.putText(img, "I'M Teja", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
52     elif count_defects == 2:
53         str = "i am fine"
54         cv2.putText(img, str, (5, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
55     elif count_defects == 3:
56         cv2.putText(img, "how are you (:)", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
57     elif count_defects == 4:
58         cv2.putText(img, "whats the matter", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
59     else:
60         cv2.putText(img, "hand gesture", (50, 50),
61                     cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
```

```
while (cap.isOpened()) > for i in range(defects.shape[0])
```

gesture.py × main.py ×

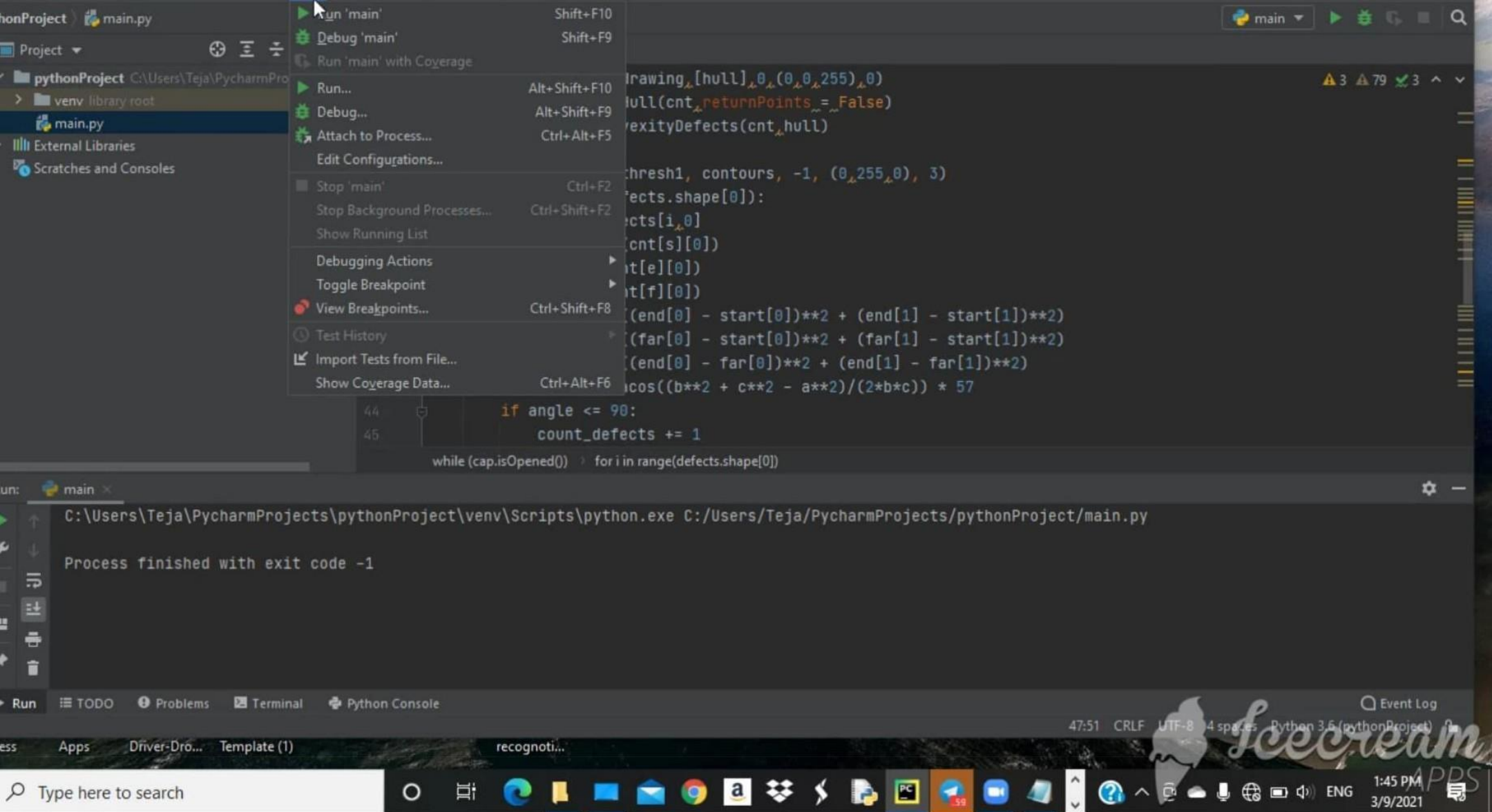
```
60         cv2.putText(img, "hand gesture", (50, 50),  
61                     cv2.FONT_HERSHEY_SIMPLEX, 2, 2)  
62         #cv2.imshow('drawing', drawing)  
63         #cv2.imshow('end', crop_img)  
64         cv2.imshow('Gesture', img)  
65         all_img = np.hstack((drawing, crop_img))  
66         cv2.imshow('Contours', all_img)  
67         k = cv2.waitKey(10)  
68         if k == 27:  
69             break  
70
```

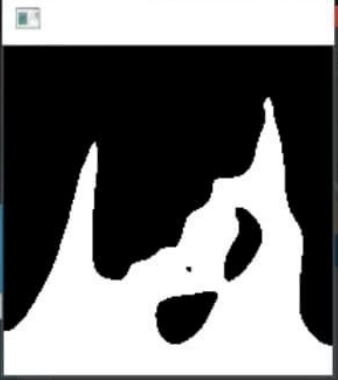
```
while (cap.isOpened()) → for i in range(defects.shape[0])
```

gesture.py × main.py ×

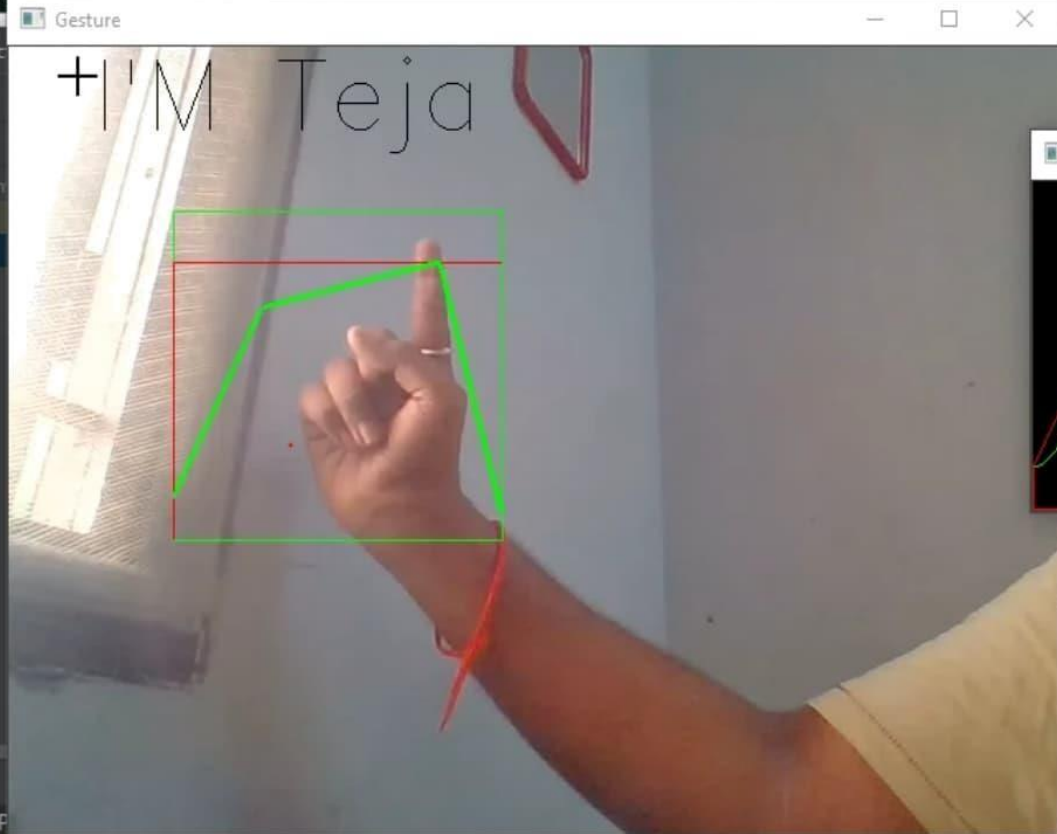
```
60         cv2.putText(img, "hand gesture", (50, 50),  
61                     cv2.FONT_HERSHEY_SIMPLEX, 2, 2)  
62         #cv2.imshow('drawing', drawing)  
63         #cv2.imshow('end', crop_img)  
64         cv2.imshow('Gesture', img)  
65         all_img = np.hstack((drawing, crop_img))  
66         cv2.imshow('Contours', all_img)  
67         k = cv2.waitKey(10)  
68         if k == 27:  
69             break  
70
```

```
while (cap.isOpened()) > for i in range(defects.shape[0])
```

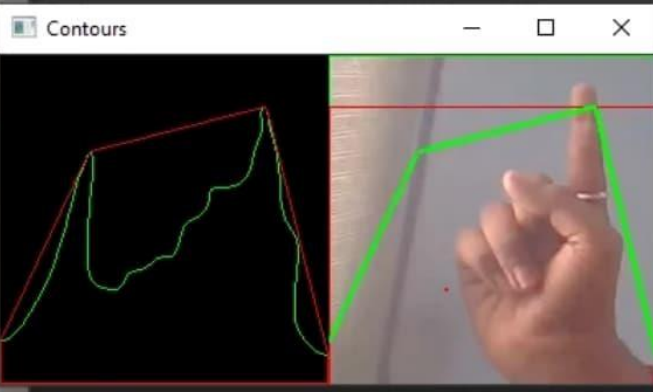




Gesture



Contours

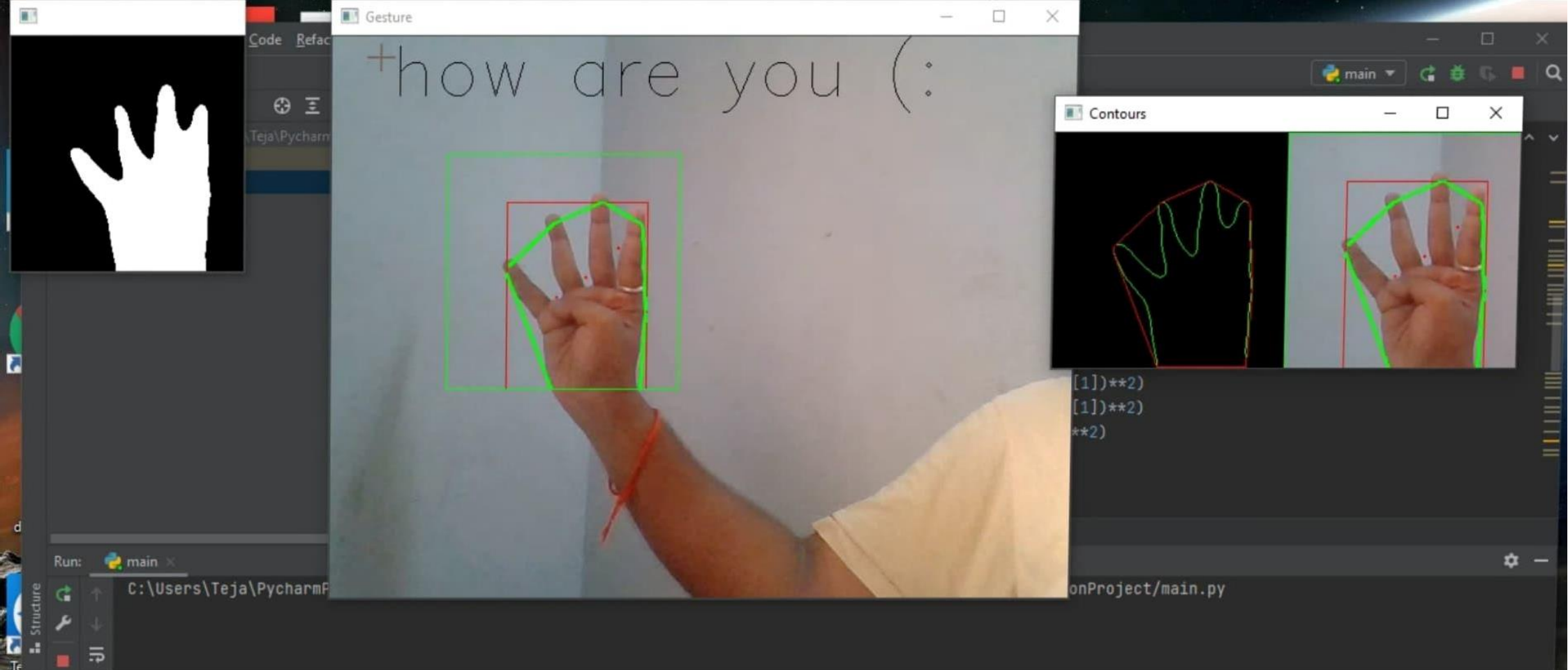


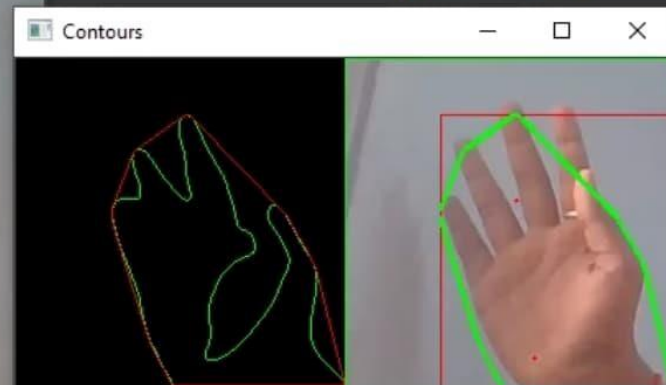
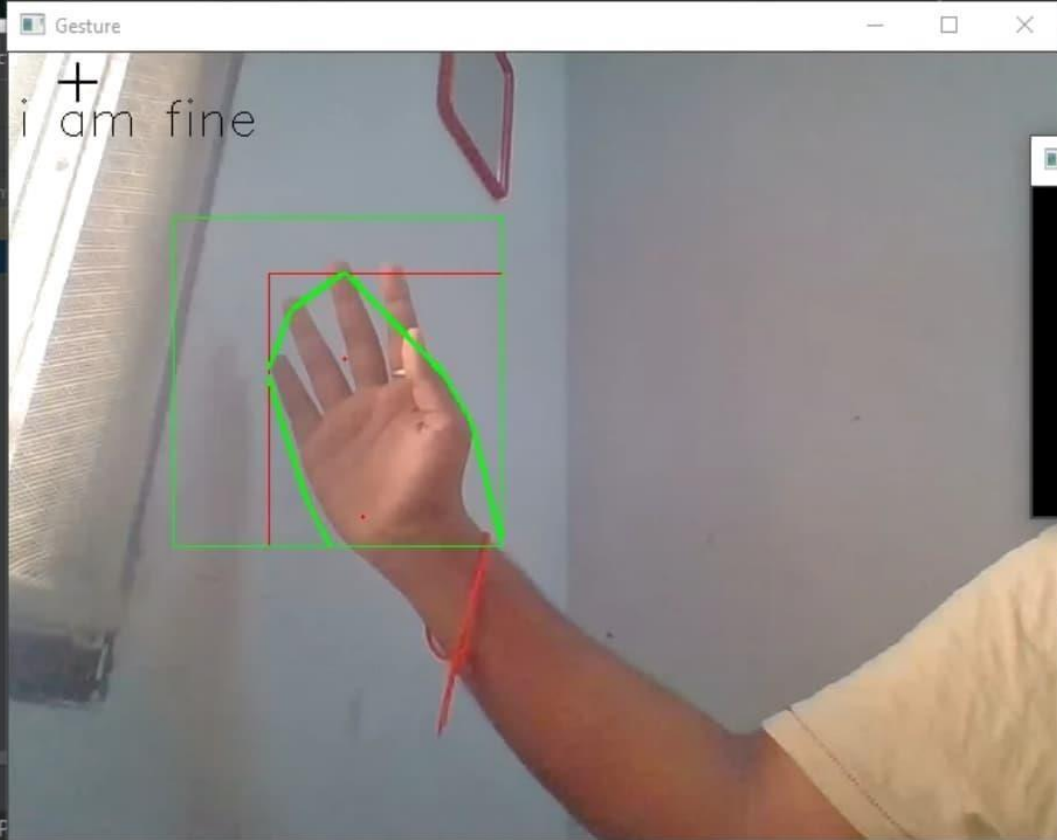
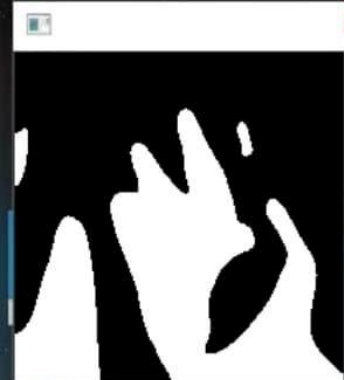
```
[1])**2)
[1])**2)
**2)
```

Run: main

C:\Users\Teja\PycharmF

onProject/main.py





```
[1])**2)  
[1])**2)  
**2)
```

onProject/main.py

P Run TODO 8 Problems M Terminal N Qt console

* " 51 CALf

drowsiness Apps 0 river- Dro ... Tempia ie t 1

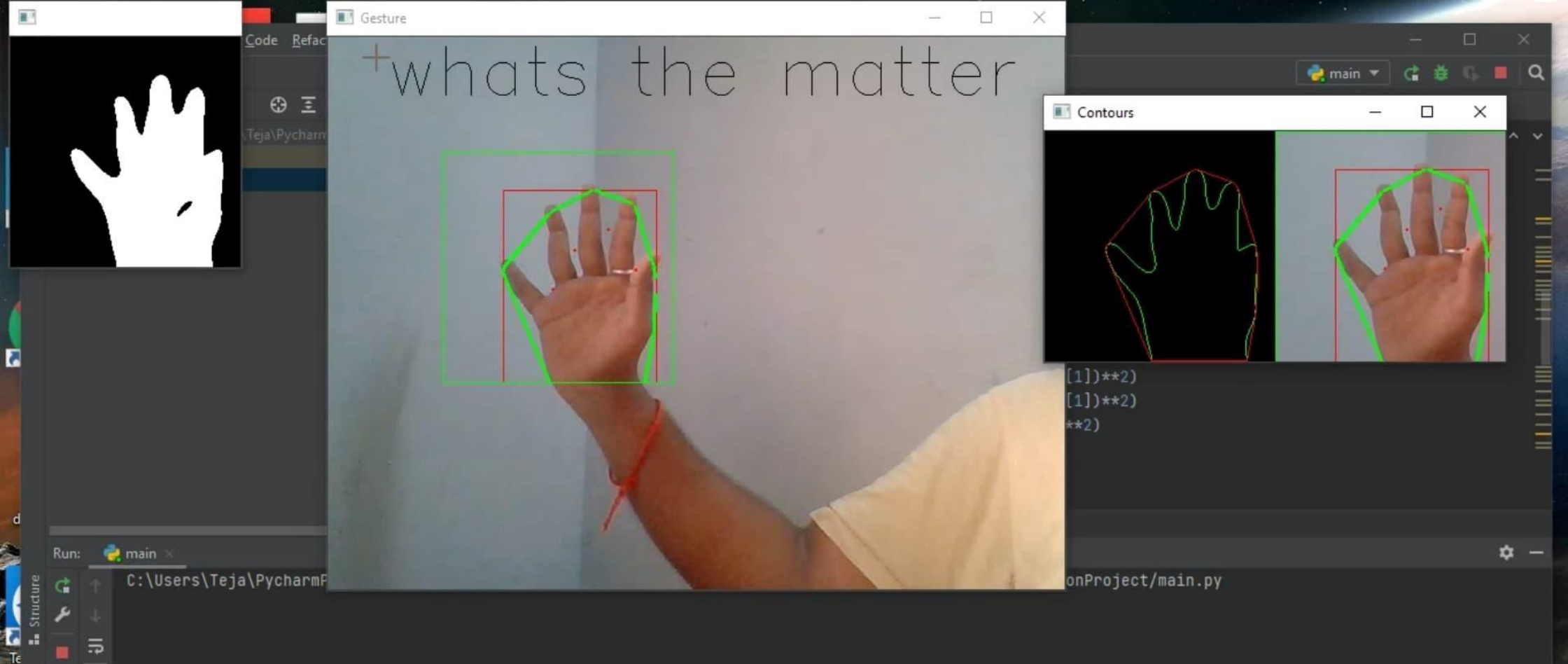
0

% :

Rwording: 0...

Threeholded

Gesture



P Run TODO B Problems M Ttrminal 4 Un on Con\oir

drowsiness Apps 0 river- Dro... Tempia ie t 1

j

Rwording: 0...

Threeholded

Gesture