# Hand written recognition using Machine Learning

## Name:-S.Jagruthi

**Abstract;**

- I presents a camera-based label reader to help blind persons to read names of labels on the products. Camera acts as main vision in detecting the label image of the product or board then image is processed internally and separates label from image by using open CV library and finally identifies the product and identified product name is pronounced through voice.
- Now it identifies received label image is converted to text by using tesseract library. Once the identified label name is converted to text and converted text is displayed on display unit connected to controller. Now converted text should be converted to voice to hear label name as voice through ear phones connected to audio jack port using flite library.

Objective

- I presents a camera-based label reader to help blind persons to read names of labels on the products. Camera acts as main vision in detecting the label image of the product or board then image is processed internally and separates label from image by using open CV library and finally identifies the product and identified product name is pronounced through voice.
- Now it identifies received label image is converted to text by using tesseract library. Once the identified label name is converted to text and converted text is displayed on display unit connected to controller. Now converted text

should be converted to voice to hear label name as voice through ear phones connected to audio jack port using flite.

# Chapter 1

## Introduction:

This project, 'Handwritten Character Recognition' is a software algorithm project to recognize any hand written character efficiently on computer with input is either an old optical image or currently provided through touch input, mouse or pen. Character recognition, usually abbreviated to optical character recognition or shortened OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. It is a field of research in pattern recognition, artificial intelligence and machine vision. Though academic research in the field continues, the focus on character recognition has shifted to implementation of proven techniques. Optical character recognition is a scheme which enables a computer to learn, understand, improvise and interpret the written or printed character in their own language, but present correspondingly as specified by the user. Optical Character Recognition uses the image processing technique to identify any character computer/typewriter printed or hand written. A lot of work has been done in this field. But a continuous improvisation of OCR techniques is being done based on the fact that algorithm must have higher accuracy of recognition, higher persistency in number of times of correct prediction and increased execution time. The idea is to device efficient algorithms which get input in digital image format. After that it processes the image for better comparison. Then after the

processed image is compared with already available set of font images. The last step gives a prediction of the character in percentage accuracy.

## Chapter 2

**Literature survey:**

1. **Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing**

**Author:** Sara Aqab1, Muhammad Usman Tariq2

Abstract:

Due to increased usage of digital technologies in all sectors and in almost all day to day activities to store and pass information, Handwriting character recognition has become a popular subject of research. Handwriting remains relevant, but people still want to have Handwriting copies converted into electronic copies that can be communicated and stored electronically. Handwriting character recognition refers to the computer's ability to detect and interpret intelligible Handwriting input from Handwriting sources such as touch screens, photographs, paper documents, and other sources. Handwriting characters remain complex since different individuals have different handwriting styles. This paper aims to report the development of a Handwriting character recognition system that will be used to read students and lectures Handwriting notes. The development is based on an artificial neural network, which is a field of study in artificial intelligence. Different techniques and methods are used to develop a Handwriting character recognition system. However, few of them focus on neural networks. The use of neural networks for

recognizing Handwriting characters is more efficient and robust compared with other computing techniques. The paper also outlines the methodology, design, and architecture of the Handwriting character recognition system and testing and results of the system development. The aim is to demonstrate the effectiveness of neural networks for Handwriting character recognition.

## 2. HAND-WRITTEN CHARCTER RECOGNITION

**Author:** CHANDAN KUMAR

**Abstract:** In todays' world advancement in sophisticated scientific techniques is pushing further the limits of human outreach in various fields of technology. One such field is the field of character recognition commonly known as OCR (Optical Character Recognition). In this fast paced world there is an immense urge for the digitalization of printed documents and documentation of information directly in digital form. And there is still some gap in this area even today. OCR techniques and their continuous improvisation from time to time is trying to fill this gap. This project is about devising an algorithm for recognition of hand written characters also known as HCR (Handwritten Character Recognition) leaving aside types of OCR that deals with recognition of computer or typewriter printed characters. A novel technique is proposed for recognition English language characters using Artificial Neural Network including the schemes of feature extraction of the characters and implemented. The persistency in recognition of characters by the AN network was found to be more than 90% of times.

## 3. DIAGONAL BASED FEATURE EXTRACTION FOR HANDWRITTEN ALPHABETS RECOGNITION SYSTEM USING NEURAL NETWORK

**Author:** J.Pradeep1 , E.Srinivasan2 and S.Himavathi3

Abstract: An off-line handwritten alphabetical character recognition system using multilayer feed forward neural network is described in the paper. A new method, called, diagonal based feature extraction is introduced for extracting the features of the handwritten alphabets. Fifty data sets, each containing 26 alphabets written by various people, are used for training the neural network and 570 different handwritten alphabetical characters are used for testing. The proposed recognition system performs quite well yielding higher levels of recognition accuracy compared to the systems employing the conventional horizontal and vertical methods of feature extraction. This system will be suitable for converting

handwritten documents into structural text form and recognizing handwritten names

## Chapter 3

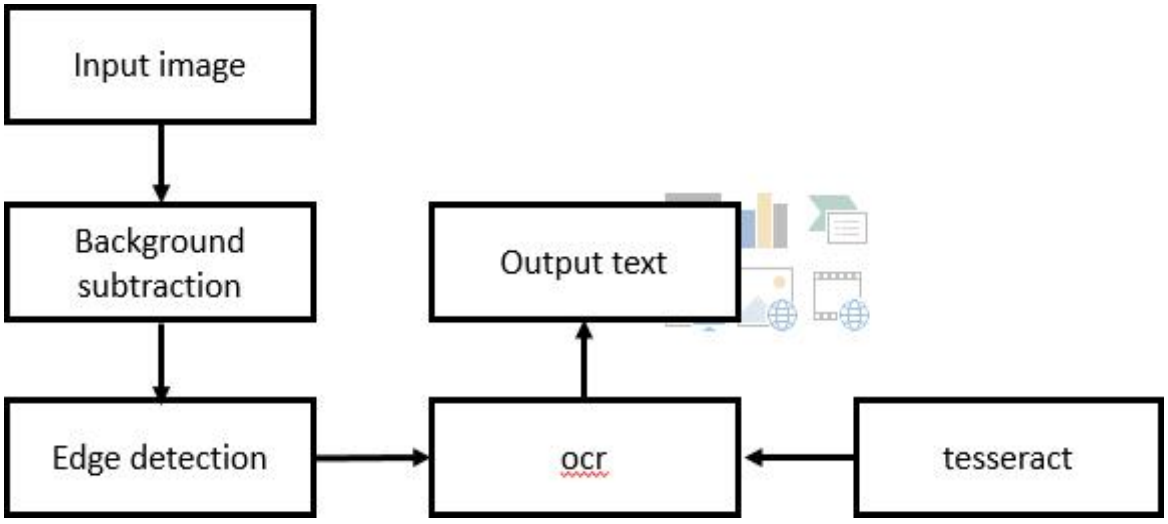
### Existing system

- 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.
- K Reader Mobile runs on a cell phone and allows the user to read mail, receipts, fliers, and many other documents

### Proposed system

- A low cost, automatic system for reading text books will be implemented that not only converts printed books to digital text, but also reads them as a audio output.
- Our proposed algorithm can effectively handle complex background and multiple patterns, and extract text information from both hand-held objects and nearby signage

**Block diagram**




**Software required:**

- Python
- Open cv

**OCR:**

**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

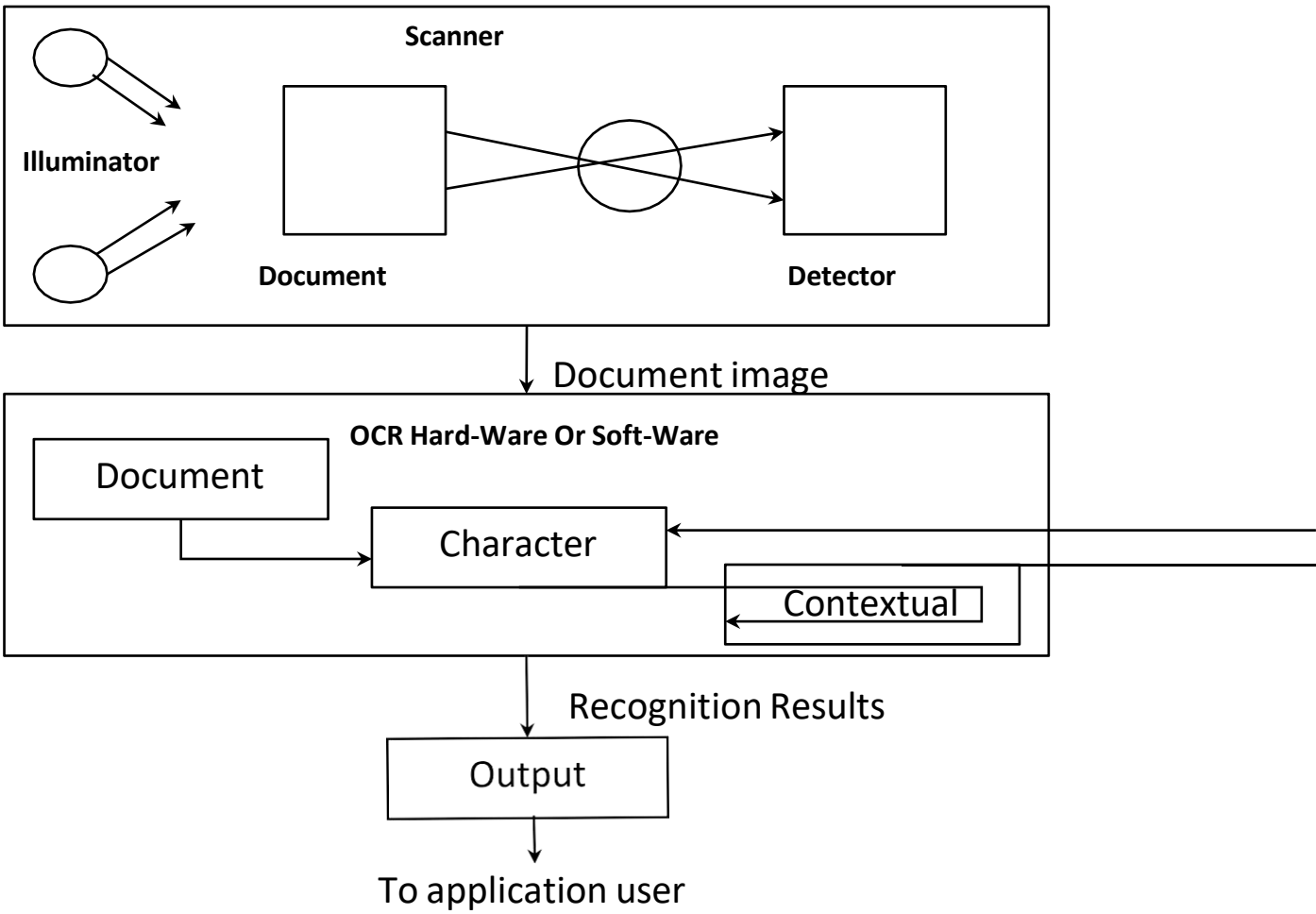



**Figure.1: OCR Architecture**

## 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.
- Reader Mobile runs on a cell phone and allows the user to read mail, receipts, fliers, and many other documents

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.

## 2.1 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?.

## 2.2 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.

## 2.3 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.

## 2.4 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.

### 2.4.1 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.

### 2.4.2 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

## 2.5 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 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".

## 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.**

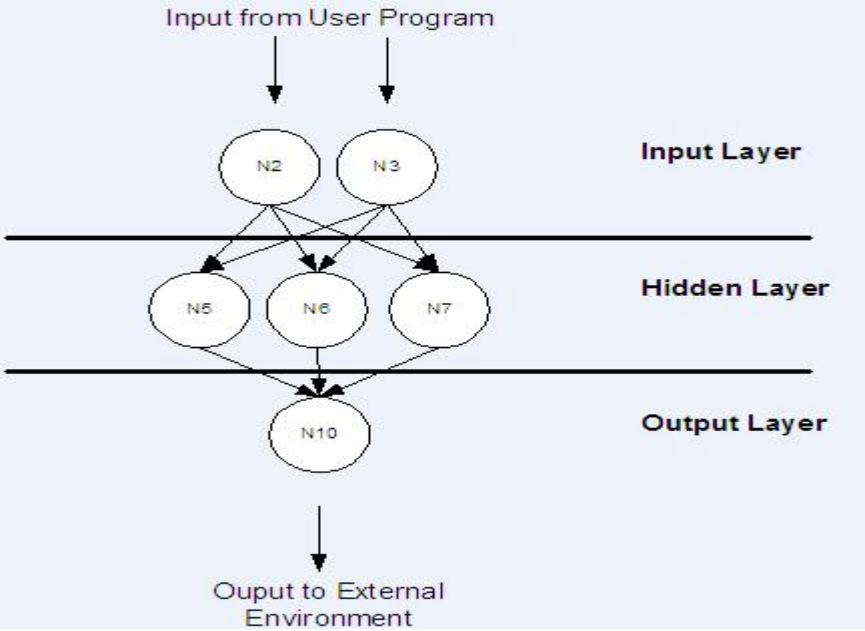



**Figure 2 Feed Forward Neural Network**

## 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 patter is presented by the output layer can be

directly traced back to the input layer. The number of a 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

## I. INTRODUCTION

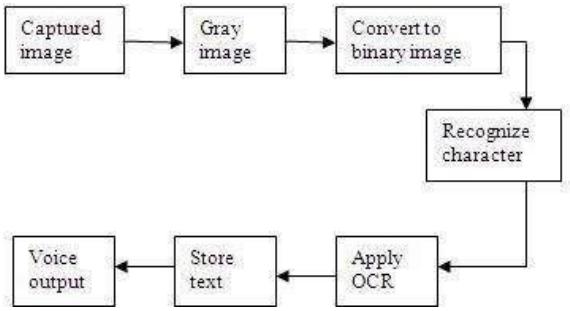
Of the 314 million visually impaired people worldwide, 45 million are blind. Recent developments in computer vision, digital cameras and portable computers make it feasible to assist these individuals by developing camera based products that combine computer vision technology with other existing commercial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

## III. 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.

## IV. 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.

## V. 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.

## VI. 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.

**OCR**

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.

### 4.7.1 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.

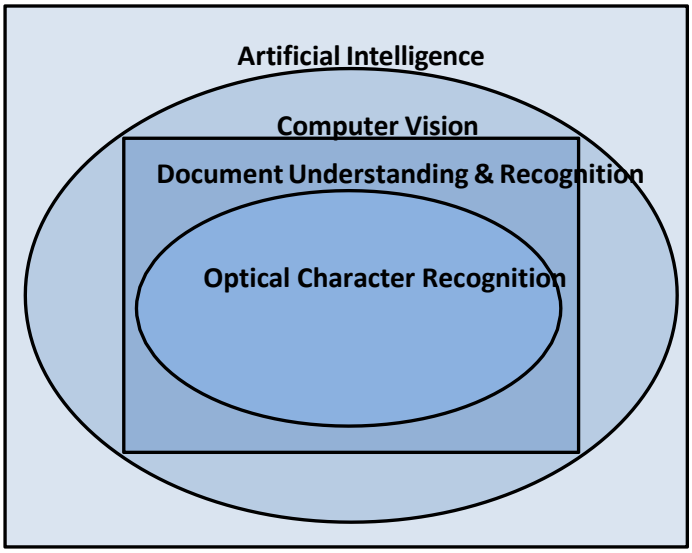



**Figure 1.1: Sub-fields of Artificial Intelligence**

a. **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:

b. On-line character recognition

c. Off-line character Recognition

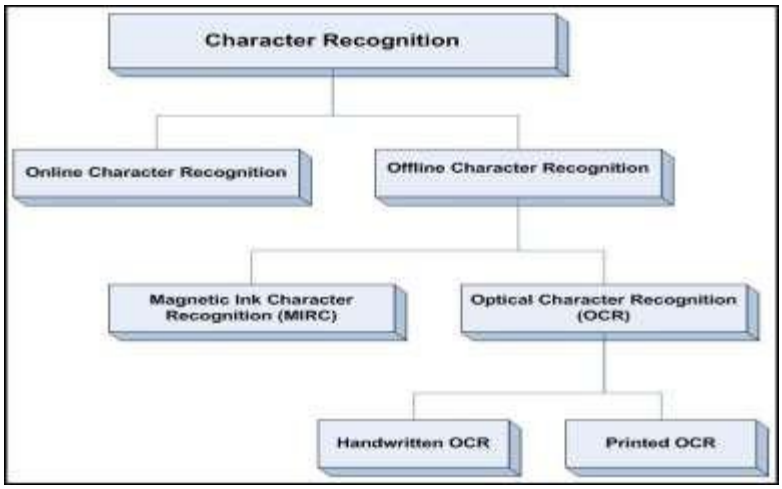



**Figure 1.2: Classification of Character Recognition**

d. **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.

4.7.2 **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)

### 4.7.3 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.

### 4.7.4 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.

### 4.7.5 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.

### 4.7.6 OCR Processes

The OCR process begins with the scanning and subsequent digital reproduction of the text in the image. It involves the following discrete sub-processes, as shown in figure 1.3.

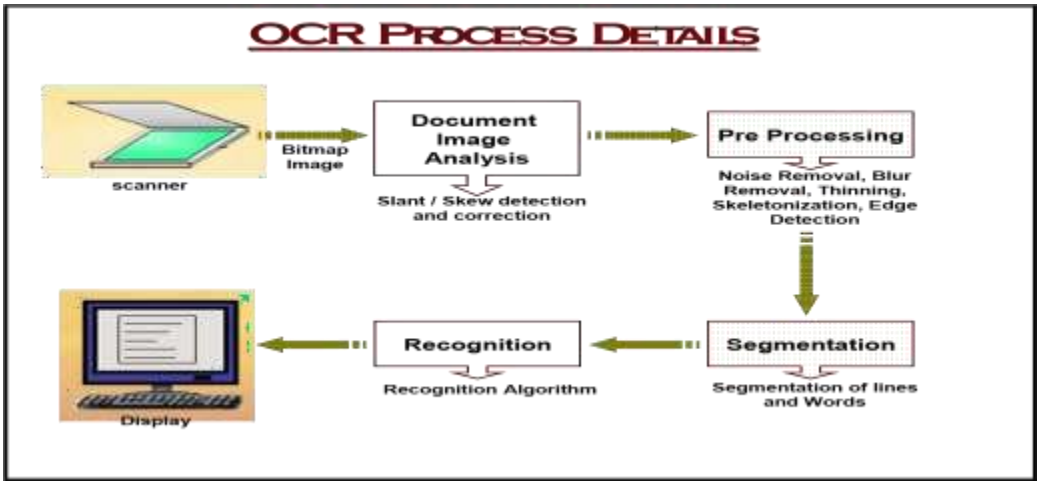



**Figure 1.3: OCR Processes**

### 4.7.7 Scanning

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

### 4.7.8 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.

### 4.7.9 Pre-processing

In this phase several processes are applied to the text image like noise and blur removal, banalization, 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.

### 4.7.10 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.

### 4.7.11 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. Offline Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text, an example is shown in Fig. 1. We will build a Neural Network (NN) which is trained on word-images from the IAM dataset. As the input layer (and therefore also all the other layers) can be kept small for word-images, NN-training is feasible on the CPU (of course, a GPU would be better). This implementation is the bare minimum that is needed for HTR using TF.

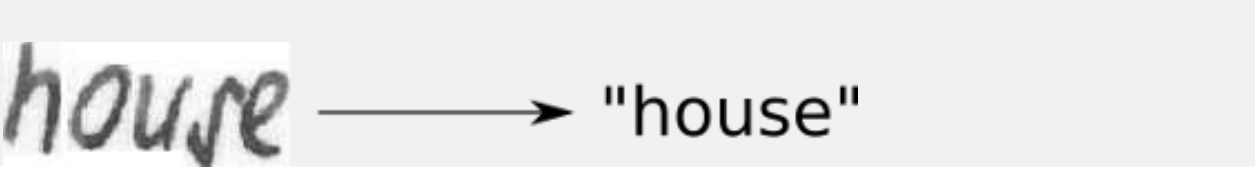



Fig. 1: Image of word (taken from IAM) and its transcription into digital text.

Get code and data

1. You need Python 3, TensorFlow 1.3, numpy and OpenCV installed
2. Get the implementation from GitHub: either take the code version this article is based on, or take the newest code version if you can accept some inconsistencies between article and code
3. Further instructions (how to get the IAM dataset, command line parameters, …) can be found in the README

Model Overview

We use a NN for our task. It consists of convolutional NN (CNN) layers, recurrent NN (RNN) layers and a final Connectionist Temporal Classification (CTC) layer.

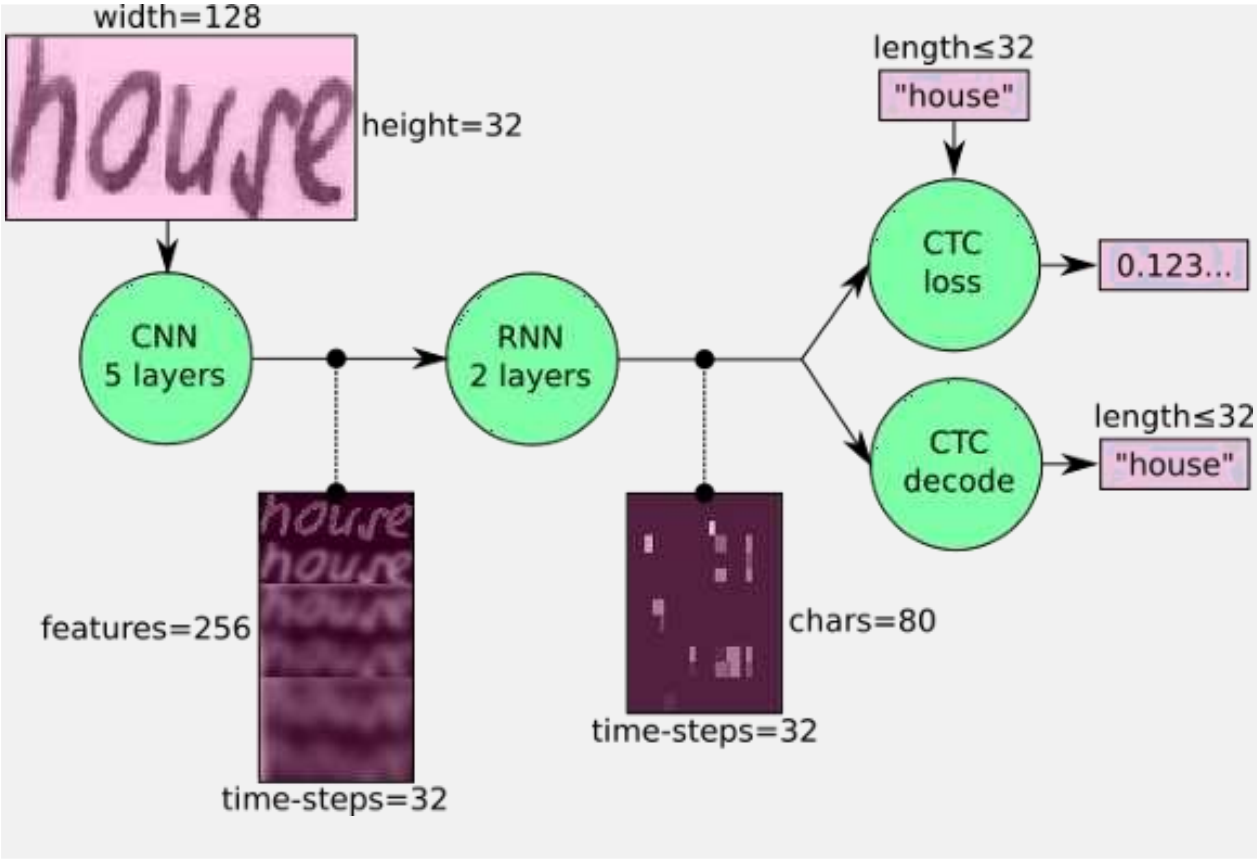



Fig. 2: Overview of the NN operations (green) and the data flow through the NN (pink).

We can also view the NN in a more formal way as a function (see Eq. 1) which maps an image (or matrix) M of size W×H to a character sequence (c1, c2, …) with a length between 0 and L. As you can see, the text is recognized on character-level, therefore words or texts not contained in the training data can be recognized too (as long as the individual characters get correctly classified).

$$\underset{W\times H}{M} \rightarrow \underset{0\leq n\leq L}{(c_1, c_2, \ldots, c_n)}$$

Eq. 1: The NN written as a mathematical function which maps an image M to a character sequence (c1, c2, …).

Operations

**CNN**: the input image is fed into the CNN layers. These layers are trained to extract relevant features from the image. Each layer consists of three operation. First, the convolution operation, which applies a filter kernel of size 5×5 in the first two layers and 3×3 in the last three layers to the input. Then, the non-linear RELU function is applied. Finally, a pooling layer summarizes image regions and outputs a downsized version of the input. While the image height is downsized by 2 in each layer, feature maps (channels) are added, so that the output feature map (or sequence) has a size of 32×256.

**RNN**: the feature sequence contains 256 features per time-step, the RNN propagates relevant information through this sequence. The popular Long Short-Term Memory (LSTM) implementation of RNNs is used, as it is able to propagate information through longer distances and provides more robust training-characteristics than vanilla RNN. The RNN output sequence is mapped to a matrix of size 32×80. The IAM dataset consists of 79 different characters, further one additional character is needed for the CTC operation (CTC blank label), therefore there are 80 entries for each of the 32 time-steps.

**CTC**: while training the NN, the CTC is given the RNN output matrix and the ground truth text and it computes the **loss value**. While inferring, the CTC is only given the matrix and it decodes it into the **final text**. Both the ground truth text and the recognized text can be at most 32 characters long.

Data

**Input**: it is a gray-value image of size 128×32. Usually, the images from the dataset do not have exactly this size, therefore we resize it (without distortion) until it either has a width of 128 or a height of 32. Then, we copy the image into a (white) target image of size 128×32. This process is shown in Fig. 3. Finally, we normalize the gray-values of the image which simplifies the task for the NN. Data augmentation can easily be integrated by copying the image to random positions instead of aligning it to the left or by randomly resizing the image.

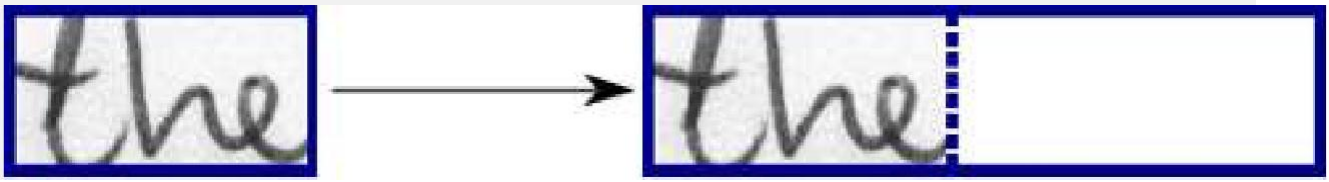


Fig. 3: Left: an image from the dataset with an arbitrary size. It is scaled to fit the target image of size 128×32, the empty part of the target image is filled with white color.

**CNN output**: Fig. 4 shows the output of the CNN layers which is a sequence of length 32. Each entry contains 256 features. Of course, these features are further processed by the RNN layers, however, some features already show a high correlation with certain high-level properties of the input image: there are features which have a high correlation with characters (e.g. —e‖), or with duplicate characters (e.g. —tt‖), or with character-properties such as loops (as contained in handwritten —l‖s or —e‖s).

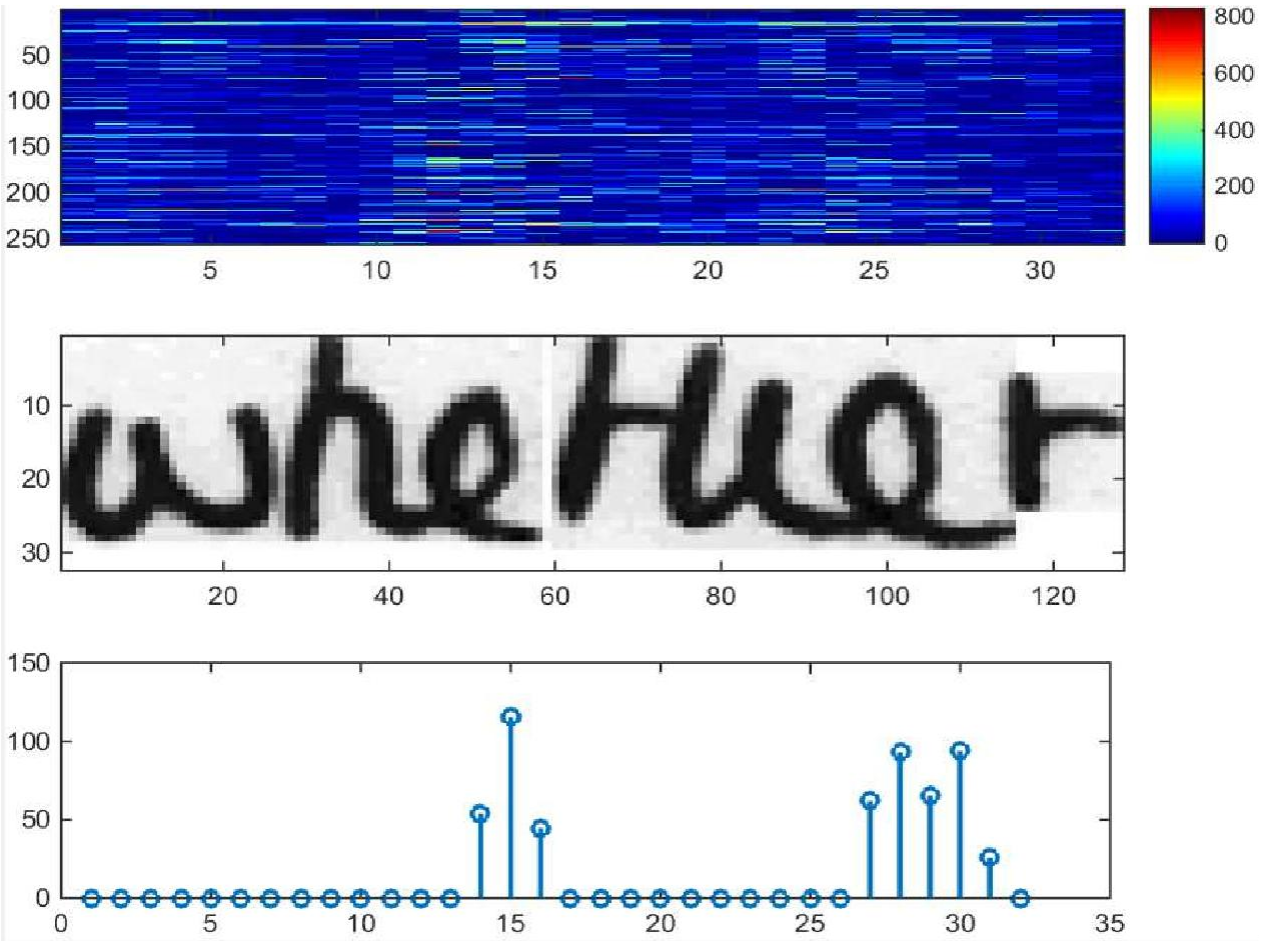



Fig. 4: Top: 256 feature per time-step are computed by the CNN layers. Middle: input image. Bottom: plot of the 32nd feature, which has a high correlation with the occurrence of the character ―e‖ in the image.

**RNN output**: Fig. 5 shows a visualization of the RNN output matrix for an image containing the text ―little‖. The matrix shown in the top-most graph contains the scores for the characters including the CTC blank label as its last (80th) entry. The other matrix-entries, from top to bottom, correspond to the following characters: ―!‖#&’()*+,-./0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz‖. It can be seen that most of the time, the characters are predicted exactly at the position they appear in the image (e.g. compare the position of the ―i‖ in the image and in the graph). Only the last character ―e‖ is not aligned. But this is OK, as the CTC operation is segmentation-free and does not care about absolute positions. From the bottom-most graph showing the scores for the characters ―l‖, ―i‖, ―t‖, ―e‖ and the CTC blank label, the text can easily be decoded: we just take

the most probable character from each time-step, this forms the so called best path, then we throw away repeated characters and finally all blanks: —l---ii--t-t--l-…-e‖ → —l---i--t-t--l-…-e‖ → —little‖.

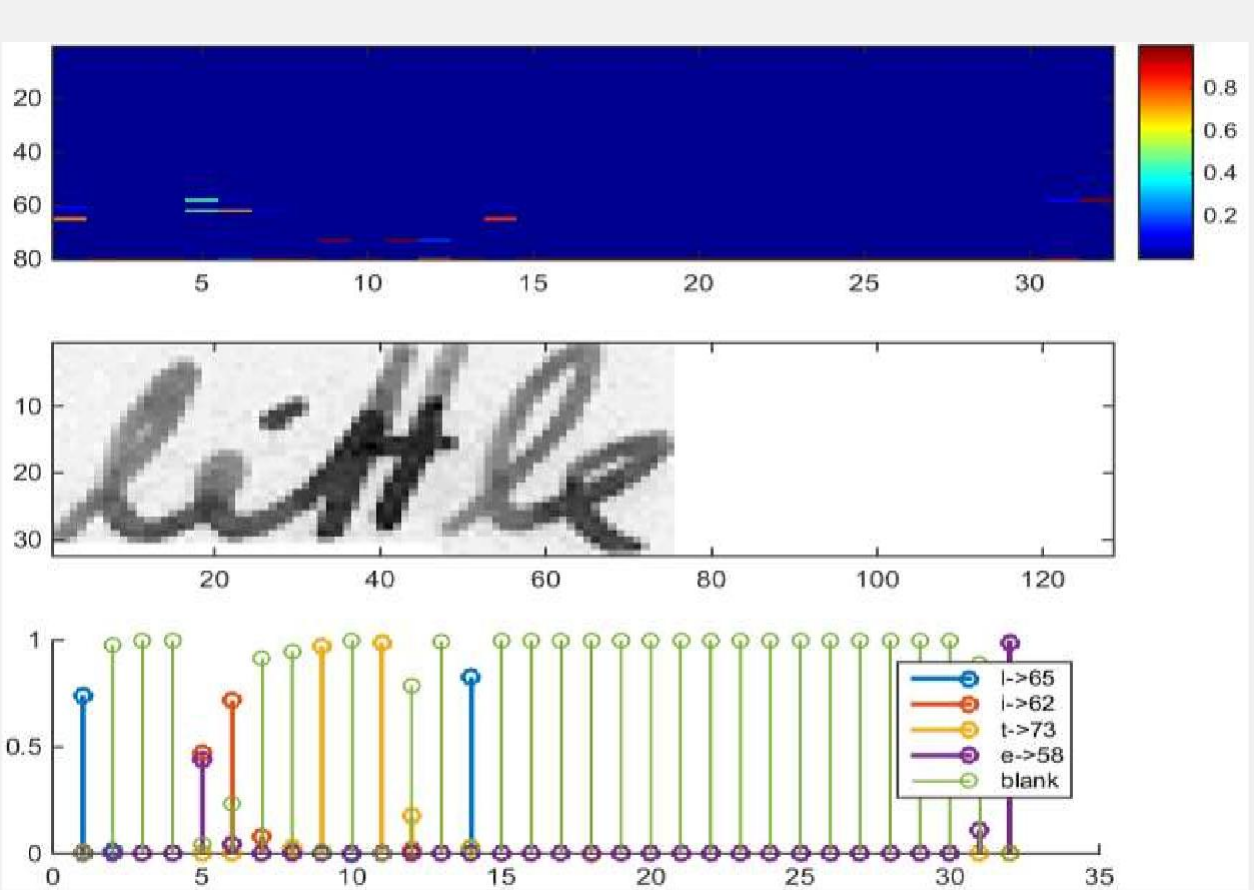



Fig. 5: Top: output matrix of the RNN layers. Middle: input image. Bottom: Probabilities for the characters —l‖, —i‖, —t‖, —e‖ and the CTC blank label.

Implementation using TF

The implementation consists of 4 modules:

1. SamplePreprocessor.py: prepares the images from the IAM dataset for the NN
2. DataLoader.py: reads samples, puts them into batches and provides an iterator-interface to go through the data

3. Model.py: creates the model as described above, loads and saves models, manages the TF sessions and provides an interface for training and inference

4. main.py: puts all previously mentioned modules together

We only look at Model.py, as the other source files are concerned with basic file IO (DataLoader.py) and image processing (SamplePreprocessor.py).

CNN

For each CNN layer, create a kernel of size k×k to be used in the convolution operation.

Then, feed the result of the convolution into the RELU operation and then again to the pooling layer with size px×py and step-size sx×sy.

These steps are repeated for all layers in a for-loop.

RNN

Create and stack two RNN layers with 256 units each.

Then, create a bidirectional RNN from it, such that the input sequence is traversed from front to back and the other way round. As a result, we get two output sequences fw and bw of size 32×256, which we later concatenate along the feature-axis to form a sequence of size 32×512. Finally, it is mapped to the output sequence (or matrix) of size 32×80 which is fed into the CTC layer.

CTC

For loss calculation, we feed both the ground truth text and the matrix to the operation. The ground truth text is encoded as a sparse tensor. The length of the input sequences must be passed to both CTC operations.

We now have all the input data to create the loss operation and the decoding operation.

Training

The mean of the loss values of the batch elements is used to train the NN: it is fed into an optimizer such as RMSProp.

Improving the model

In case you want to feed complete text-lines as shown in Fig. 6 instead of word-images, you have to increase the input size of the NN.

Fig. 6: A complete text-line can be fed into the NN if its input size is increased (image taken from IAM).

If you want to improve the recognition accuracy, you can follow one of these hints:

- Data augmentation: increase dataset-size by applying further (random) transformations to the input images
- Remove cursive writing style in the input images (see DeslantImg)
- Increase input size (if input of NN is large enough, complete text-lines can be used)
- Add more CNN layers
- Replace LSTM by 2D-LSTM
- Decoder: use token passing or word beam search decoding (see CTCWordBeamSearch) to constrain the output to dictionary words
- Text correction: if the recognized word is not contained in a dictionary, search for the most similar one

We discussed a NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding. An implementation using TF is provided and some important parts of the code were presented. Finally, hints to improve the recognition accuracy were given.

# Chapter 4

## Python (programming language)

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation

## History

Python was conceived in the late 1980s, and its implementation began in December 1989 [28] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the operating system Amoeba. Van Rossum is Python's principal author, and his continuing central role

in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (BDFL).
About the origin of Python, Van Rossum wrote in 1996:Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed. Python 3.0 (which early in its development was commonly referred to as Python 3000 or py3k), a major, backwards-incompatible release, was released on 3 December 2008 after a long period of testing. Many of its major features have been backported to the backwards-compatible Python 2.6.xand 2.7.x version series. The End Of Life date (EOL, sunset date) for Python 2.7 was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code cannot easily be forward-ported to Python 3.

In January 2017, Google announced work on a Python 2.7 to Go transcompiler, which The Register speculated was in response to Python 2.7's planned end-of-life but Google cited performance under concurrent workloads as their only motivation. Features and philosophy Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 3/19 paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management.

An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. The design of Python offers some support for functional programming in the Lisp tradition. The language has map(), reduce() and filter() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. The core philosophy of the language is summarized by the document The Zen of Python (PEP 20), which includes aphorisms such as: Beautiful is better than ugly Explicit is better than implicit Simple is better than complex Complex is better than complicated Readability counts Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface.

This design of a small core language with a large standard library and an easily extensible interpreter was intended by Van Rossum from the start because of his frustrations with ABC, which espoused the opposite mindset. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax, such as in Perl, in favor of a sparser, less-cluttered grammar. As Alex Martelli put it: "To describe something as clever is not considered a compliment in the Python culture." Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it". Python's developers strive to avoid premature optimization, and moreover, reject patches to non-critical parts of CPython that would offer a marginal increase in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or try using PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is making it fun to use. This is reflected in the origin of the name, which comes from Monty Python, and in an occasionally playful approach to tutorials and reference materials, such as using examples that refer to spam and eggs instead of the standard foo and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style.

To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers. Syntax and semantics 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 4/19 Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, often using English keywords where other languages use punctuation. Python doesn't have semicolons and curly brackets "{}" which is different compared to most of the programming language. Further, Python has fewer syntactic exceptions and special cases than C or Pascal. Indentation Python uses whitespace indentation to delimit blocks – rather than curly braces or keywords. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. This feature is also sometimes termed the off-side rule. Statements and control flow Python's statements include (among others): The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language.

Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating

storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it.

However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing. The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if). The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block. The while statement, which executes a block of code as long as its condition is true. The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits. The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming. The def statement, which defines a function or method. The with statement (from Python 2.5), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior.

The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block. The assert statement, used during debugging to check for conditions that ought to apply. The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines. The import statement, which is used to import modules whose functions or variables can be used in the current program. The print statement was changed to the print() function in Python 3. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 5/19 of the generator.

As of Python 2.5, it is possible to pass information back into a generator function, and as of Python 3.3, the information can be passed through multiple stack levels. Expressions Some Python expressions are similar to languages such as C and Java, while some are not: Addition, subtraction, and multiplication are the same, but the behavior of division differs. Python also added the ** operator for exponentiation. As of Python 3.5, it supports matrix multiplication directly with the @ operator, versus C and Java, which implement these as library functions. Earlier versions of Python also used methods instead of an infix operator. In Python, == compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, 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 one expression.

Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y 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. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python has various kinds of string literals: Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String

interpolation (done as "$spam" in Unix shells and Perl-influenced languages) became available in Python 3.6 as "formatted string literals". Triple-quoted strings, which begin and end with a series of three single or double quote 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. Escape sequences are not 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 array index and array slicing 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 duplicating some 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. 6/13/2017 Python (programming language) - 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 syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python. Methods Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Typing 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 objectoriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta programming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style. 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). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0. The long term plan is to support gradual typing and as of Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

Mathematics Python has the usual C arithmetic operators (+, –, *, /, %). It also has ** for exponentiation, e.g. 5**3 == 125 and 9**0.5 == 3.0, and a new matrix multiply @ operator is included in version 3.5. [71] Additionally, it has a unary operator (~), which essentially inverts all the bytes of its one argument. For integers, this means ~x=-x-1. Other operators include bitwise shift operators x << y, which shifts x to the left y places, the same as x*(2**y) , and x >> y, which shifts x to the right y places, the same as x/(2**y) . [73] The behavior of division has changed significantly over time: Python 2.1 and earlier use the C division behavior. The / operator is integer division if both operands are integers, and floating-point division otherwise. Integer division rounds towards 0, e.g. 7 / 3 == 2

and –7 / 3 == –2. Python 2.2 changes integer division to round towards negative infinity, e.g. 7 / 3 == 2 and –7 / 3 == –3. The floor division // operator is introduced. So 7 // 3 == 2, –7 // 3 == –3, 7.5 // 3 == 2.0 and 6/13/2017 Python (programming language) – Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 8/19 –7.5 // 3 == –3.0. Adding from __future__ import division causes a module to use Python 3.0 rules for division (see next). Python 3.0 changes / to be always floating-point division. In Python terms, the pre-3.0 / is classic division, the version-3.0 / is real division, and // is floor division. Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation (a+b) // b == a // b + 1 is always true. It also means that the equation b * (a // b) + a % b == a is valid for both positive and negative values of a. However, maintaining the validity of this equation means that while the result of a % b is, as expected, in the half-open interval [0, b), where b is a positive integer, it has to lie in the interval (b, 0] when b is negative. Python provides a round function for rounding a float to the nearest integer. For tie-breaking, versions before 3 use round-away-from-zero: round(0.5) is 1.0, round(-0.5) is −1.0. Python 3 uses round to even: round(1.5) is 2, round(2.5) is 2. Python allows boolean expressions with multiple equality relations in a manner that is consistent with general use in mathematics. For example, the expression a < b < c tests whether a is less than b and b is less than c. Cderived languages interpret this expression differently: in C, the expression would first evaluate a < b, resulting in 0 or 1, and that result would then be compared with c.

## Python has extensive built-in support for arbitrary precision arithmetic.

Integers are transparently switched from the machine-supported maximum fixed-precision (usually 32 or 64 bits), belonging to the python type int, to arbitrary precision, belonging to the python type long, where needed. The latter have an "L" suffix in their textual representation. (In Python 3, the distinction between the int and long types was eliminated; this behavior is now entirely contained by the int class.) The Decimal type/class in module decimal (since version 2.4) provides decimal floating point numbers to arbitrary precision and several rounding modes.The Fraction type in module fractions (since version 2.6) provides arbitrary precision for rational numbers. Due to Python's extensive mathematics library, and the third-party library NumPy which further extends the native capabilities, it is frequently used as a scientific scripting language to aid in problems such as numerical data processing and manipulation. Libraries Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks.

This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, many standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included. Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333 ), but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations. As of May, 2017, the Python Package Index, the official repository containing third-party software for Python, contains over 107,000 packages offering a wide range of functionality, including: graphical user interfaces, web frameworks, multimedia, databases, networking and communications test frameworks, automation and web scraping, documentation tools, system administration 6/13/2017 Python (programming language) - Wikipedia

https://en.wikipedia.org/wiki/Python_(programming_language) 9/19 scientific computing, text processing, image processing Development environments Most Python implementations (including CPython) include a read–eval–print loop (REPL), meaning they can function as a command line interpreter, for which the user enters statements sequentially and receives the results immediately.

Other shells add abilities beyond those in the basic interpreter, including IDLE and IPython. While generally following the visual style of the Python shell, they implement features like auto-completion, session state retention, and syntax highlighting. In addition to standard desktop integrated development environments (Python IDEs), there are also web browser-based IDEs, SageMath (intended for developing science and math-related Python programs), and a browser-based IDE and hosting environment, PythonAnywhere. Additionally, the Canopy IDE is also an option for writing Python programs. Implementations Reference implementation The main Python implementation, named CPython, is written in C meeting the C89 standard. It compiles Python programs into intermediate bytecode, which is executed by the virtual machine. CPython is distributed with a large standard library written in a mixture of C and Python. It is available in versions for many platforms, including Windows and most modern Unix-like systems. CPython was intended from almost its very conception to be cross- platform. Other implementations PyPy is a fast, compliant interpreter of Python
2.7 and 3.5. Its just-in-time compiler brings a significant speed improvement over CPython. A version taking advantage of multi-core processors using software transactional memory is being created. Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack, thus allowing massively concurrent programs.

PyPy also has a stackless version. MicroPython is a Python 3 variant that is optimised to run on microcontrollers. No longer supported implementations Other just-in-time compilers have been developed in the past, but are now unsupported: Google began a project named Unladen Swallow in 2009 with the aim of speeding up the Python interpreter fivefold by using the LLVM, and of improving its multithreading ability to scale to thousands of cores. syco is a just-in-time specialising compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialised for certain data types and

is faster than standard Python code. In 2005, Nokia released a Python interpreter for the Series 60 mobile phones named PyS60. It includes many of the modules from the CPython implementations and some additional modules to integrate with the Symbian operating system. This project has been kept up to date to run on all variants of the S60 platform and there are 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 10/19 several third party modules available.

The Nokia N900 also supports Python with GTK widget libraries, with the feature that programs can be both written and run on the target device. [96] Cross-compilers to other languages There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language: Jython compiles into Java byte code, which can then be executed by every Java virtual machine implementation. This also enables the use of Java class library functions from the Python program. IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime. The RPython language can be compiled to C, Java bytecode, or Common Intermediate Language, and is used to build the PyPy interpreter of Python. Pyjamas compiles Python to JavaScript. Cython compiles Python to C and C++. Pythran compiles Python to C++. Somewhat dated Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively. Google's Grumpy compiles Python to Go. Performance A performance comparison of various Python implementations on a non-numerical (combinatorial) workload was presented at EuroSciPy '13. Development Python's development is conducted largely through the Python Enhancement Proposal (PEP) process. The PEP process is the primary mechanism for proposing major new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python.

Outstanding PEPs are reviewed and commented upon by the Python community and by Van Rossum, the Python project's benevolent dictator for life. [98] Enhancement of the language goes along with development of the CPython reference implementation. The mailing list python-dev is the primary forum for discussion about the language's development; specific issues are discussed in the

Roundup bug tracker maintained at python.org. Development took place on a selfhosted source code repository running Mercurial, until Python moved to GitHub in January 2017. CPython's public releases come in three types, distinguished by which part of the version number is incremented: Backwards-incompatible versions, where code is expected to break and must be manually ported. The first part of the version number is incremented. These releases happen infrequently—for example, version 3.0 was released 8 years after 2.0. Major or "feature" releases, which are largely compatible but introduce new features. The second part of the version number is incremented. These releases are scheduled to occur roughly every 18 months, and each major version is supported by bugfixes for several years after its release. Bugfix releases, which introduce no new features but fix bugs. The third and final part of the version number is incremented. These releases are made whenever a sufficient number of bugs have been fixed upstream since the last release, or roughly every 3 months. Security vulnerabilities are also patched in bugfixreleases.6/13/2017 Python (programming language)-Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 11/19 Many alpha, beta, and release-candidates are also released as previews and for testing before final releases. Although there is a rough schedule for each release, this is often pushed back if the code is not ready.

The development team monitors the state of the code by running the large unit test suite during development, and using the BuildBot continuous integration system. The community of Python developers has also contributed over 86,000 software modules (as of 20 August 2016) to the Python Package Index (PyPI), the official repository of third-party libraries for Python. The major academic conference on Python is named PyCon. There are special mentoring programmes like the Pyladies. Naming Python's name is derived from the television series Monty Python's Flying Circus, and it is common to use Monty Python references in example code. For example, the metasyntactic variables often used in Python literature are spam and eggs, instead of the traditional foo and bar. Also, the official Python documentation and many code examples often contain various obscure Monty Python references. The prefix Py- is used to show that something is related to Python.

Examples of the use of this prefix in names of Python applications or libraries include Pygame, a binding of SDL to Python (commonly used to create games); PyS60, an implementation for the Symbian S60 operating system; PyQt and PyGTK, which bind Qt and GTK to Python respectively; and PyPy, a Python implementation originally written in Python. Uses Since 2003, Python has consistently ranked in the top ten most popular programming languages as measured by the TIOBE Programming Community Index. As of March 2017, it is the fifth most popular language. It was ranked as Programming Language of the Year for the year 2007 and 2010. It is the third most popular language whose grammatical syntax is not predominantly based on C, e.g. C++, Objective-C (note, C# and Java only have partial syntactic similarity to C, such as the use of curly braces, and are closer in similarity to each other than C). An empirical study found scripting languages (such as Python) more productive than conventional languages (such as C and Java) for a programming problem involving string manipulation and search in a dictionary. Memory consumption was often "better than Java and not much worse than C or C++". Large organizations that make use of Python include Wikipedia, Google, Yahoo!, CERN, NASA, and some smaller entities like ILM, and ITA.

The social news networking site Reddit is written entirely in Python. Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjamas and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox. Libraries like NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, [120][121] with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a "notebook" programmable in Python: its library covers many aspects of 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 12/19 mathematics, including algebra, combinatorics, numerical mathematics, number

theory, and calculus. The Python language re-implemented in Java platform is used for numeric and statistical calculations with 2D/3D visualization by the DMelt project. Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS.It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Python is also used in algorithmic trading and quantitative finance. Python can also be implemented in APIs of online brokerages that run on other languages by using wrappers. Python has been used in artificial intelligence tasks. As a scripting language with module architecture, simple syntax and rich text processing tools, Python is often used for natural language processing tasks. Many operating systems include Python as a standard component; the language ships with most Linux distributions, AmigaOS 4, FreeBSD, NetBSD, OpenBSD and macOS, and can be used from the terminal. Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage. Python has also seen extensive use in the information security industry, including in exploit development. Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language. LibreOffice includes Python and intends to replace Java with Python. Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

Languages influenced by Python Python's design and philosophy have influenced several programming languages, including: Boo uses indentation, a similar syntax, and a similar object model. However, Boo uses static typing (and optional duck

typing) and is closely integrated with the .NET Framework. Cobra uses indentation and a similar syntax. Cobra's "Acknowledgements" document lists Python first among languages that influenced it. However, Cobra directly supports design-by-contract, unit tests, and optional static typing. ECMAScript borrowed iterators, generators, and list comprehensions from Python. Go is described as incorporating the "development speed of working in a dynamic language like Python". Groovy was motivated by the desire to bring the Python design philosophy to Java. Julia was designed "with true macros [.. and to be] as usable for general programming as Python [and] should be as fast as C". Calling to or from Julia is possible; to with PyCall.jl (https://github.com/steve ngj/PyCall.jl) and a Python package pyjulia (https://github.com/JuliaLang/pyjulia) allows calling, in the other direction, from Python. 6/13/2017 Python (programming language) - Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 13/19 OCaml has an optional syntax, named twt (The Whitespace Thing), inspired by Python and Haskell. Ruby's creator, Yukihiro Matsumoto, has said: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language." Coffee Script is a programming language that cross-compiles to JavaScript; it has Python-inspired syntax. Swift is a programming language invented by Apple; it has some Python-inspired syntax. Python's development practices have also been emulated by other languages. The practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python's case, a PEP) is also used in Tcl and Erlang because of Python's influence. Python has been awarded a TIOBE Programming Language of the Year award twice (in 2007 and 2010), which is given to the language with the greatest growth in popularity over the course of a year, as measured by the TIOBE index.

**OpenCV:**

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license

History Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU intensive applications, part of a series of

projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as: Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel. Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable. Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. In mid-2008, OpenCV obtained corporate support from Willow Garage, and is now again under active development. A version 1.1 "pre-release" was released in October 2008. The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations. In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

Applications OpenCV's application areas include: 2D and 3D feature toolkits

- Emotion estimation
- Facial recognition system
- Gesture recognition Human–computer interaction (HCI)
- Mobile robotics Motion understanding
- Object identification Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras Structure from motion
- (SFM) Motion tracking Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting Decision tree learning

- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks Random forest Support vector machine (SVM)

Programming language OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Haskell and Ruby have been developed to encourage adoption by a wider audience. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

Hardware acceleration If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. A CUDA-based GPU interface has been in progress since September 2010. An OpenCL-based GPU interface has been in progress since October 2012, documentation for version 2.4.9.0 can be found at docs.opencv.org.

OS support OpenCV runs on a variety of platforms. Desktop: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD; Mobile: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.c

**SOURCECODE:**

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
```

In [2]:

```
digits= datasets.load_digits()
x=digits.data
y=digits.target
```

In [3]:

```
x[1]
```

Out[3]:

```
array([ 0.,  0.,  0., 12., 13.,  5.,  0.,  0.,  0.,  0.,  0., 11., 16.,
```

```
        9.,  0.,  0.,  0.,  0.,  3., 15., 16.,  6.,  0.,  0.,  0.,  7.,
       15., 16., 16.,  2.,  0.,  0.,  0.,  0.,  1., 16., 16.,  3.,  0.,
        0.,  0.,  0.,  1., 16., 16.,  6.,  0.,  0.,  0.,  0.,  1., 16.,
       16.,  6.,  0.,  0.,  0.,  0.,  0., 11., 16., 10.,  0.,  0.])
In [4]:
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
In [5]:
from sklearn import svm
clf=svm.SVC(kernel="poly",C=1,gamma=0.1)
clf.fit(x_train,y_train)
Out[5]:
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
In [6]:
pred=clf.predict(x_test)
In [8]:
from sklearn.metrics import accuracy_score
In [9]:
accuracy_score(pred,y_test)
Out[9]:
0.9888888888888889
In [10]:
clf.predict(digits.data[[100]])
Out[10]:
array([4])
In [11]:
clf.predict(digits.data[[50]])
Out[11]:
array([2])
In [12]:
clf.predict(digits.data[[500]])
Out[12]:
array([8])
In [13]:
plt.imshow(digits.images[100])
plt.show()

In [14]:
```

```
plt.imshow(digits.images[50])
plt.show()

In [15]:
plt.imshow(digits.images[500])
plt.show()

In [ ]:
```

**ExperimentalResults**:

**Test application Analysis:**

The test application accompanying the source code can perform the recognition of handwritten digits. To do so, open the application (preferably outside Visual Studio, for better performance). Click on the menu File and select Open. This will load some entries from the Optdigits dataset into the application. To perform the analysis, click the Run Analysis button. Please be aware that it may take some time. After the analysis is complete, the other tabs in the sample application will be populated with the analysis' information. The level of importance Experiments were performed on different samples having mixed scripting languages on numerals using single hidden layer.

| **Data set** | **Training Set Size** | **Testing Set Size** | **Validation Set Size** | **Training Set Accuracy** | **Test Set Accuracy** | **Validation Set Accuracy** |
|---|---|---|---|---|---|---|
| Digit | 1778 | 6270 | 5430 | 96 | 97 | 96 |

**Table:** Detail Recognition performance of SVM

It is observed that recognition rate using SVM is higher than other model, i.e. Hidden Markov Model. However, free parameter storage for SVM model is significantly higher. The memory space required for SVM will be the number of support vectors multiply by the number of feature values. This is significantly large compared to HMM which only need to store the weight. HMM needs less space due to the weight-sharing scheme. However, in SVM, space saving can be achieved by storing only the original online signals and the penup/ pen-down status

in a compact manner. During recognition, the model will be expanded dynamically as required. SVM clearly outperforms in all three isolated character cases. The result for the isolated character cases above indicates that the recognition rate for the hybrid word recognizer could be improved by using SVM instead of HMM.

**Source code:**

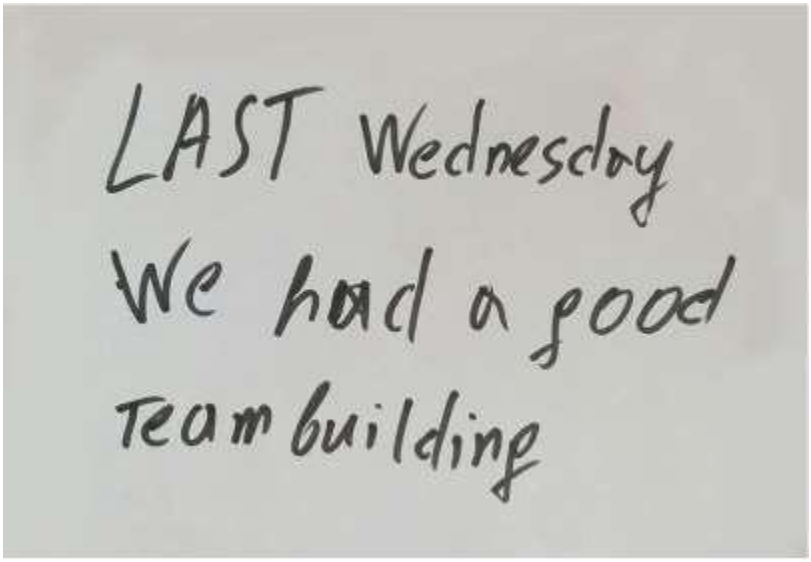
```
import cv2
from time import sleep
import os
import requests
import io
import json
import os
import random
import pyttsx3
import xlsxwriter


workbook = xlsxwriter.Workbook('test.xlsx')
worksheet = workbook.add_worksheet("My sheet")
resim = "hand1.jpg"
img = cv2.imread(resim)
print("Picture is Detected")


api = img


# Ocr
url_api = "https://api.ocr.space/parse/image"
compressedimage = cv2.imencode(".jpg", img)[1]
file_bytes = io.BytesIO(compressedimage)
```

```
result = requests.post(url_api,
                       files={resim: file_bytes},
                       data={"apikey": "helloworld",
                            "language": "eng"})

result = result.content.decode()
print(result)
result = json.loads(result)

parsed_results = result.get("ParsedResults")[0]
text_detected = parsed_results.get("ParsedText")
print(text_detected)
print("Text is writing to file...")
f = open("text_detected.txt", "a+") f.write(text_detected)
f.close()
text = text_detected
chunks = text.split('\n')

print("Given string : ", text)
#worksheet.write(text)

#res1 = ""
#for i in text:
#    if i.isalpha():
#        res1 = "".join([res1, i])
```

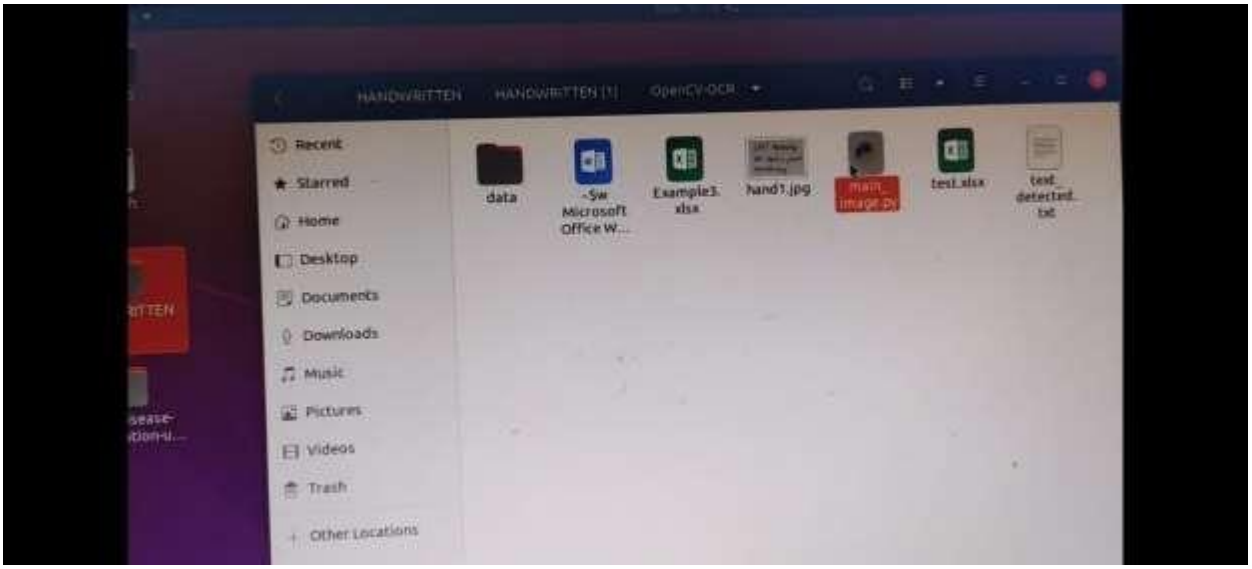
```
# Result
#print("Result: ", res1)

#chunks = res1.split('#')

#print(chunks)

#row = 0
#column = 0



#for item in res1 :

  #  worksheet.write(row, column, res1)
 #    row += 2

#res = [int(i) for i in text_detected.split() if i.isdigit()]
#print("The numbers list is : " + str(res))



row1 = 0
column1 = 0
```

```
for item in text :

    worksheet.write(row1, column1, text)
    row1 += 1
workbook.close()


print("Operation is successful")

#voice cv2.imshow("roi",
api) cv2.imshow("Img",
img) cv2.waitKey(0)
os.remove(resim)
```

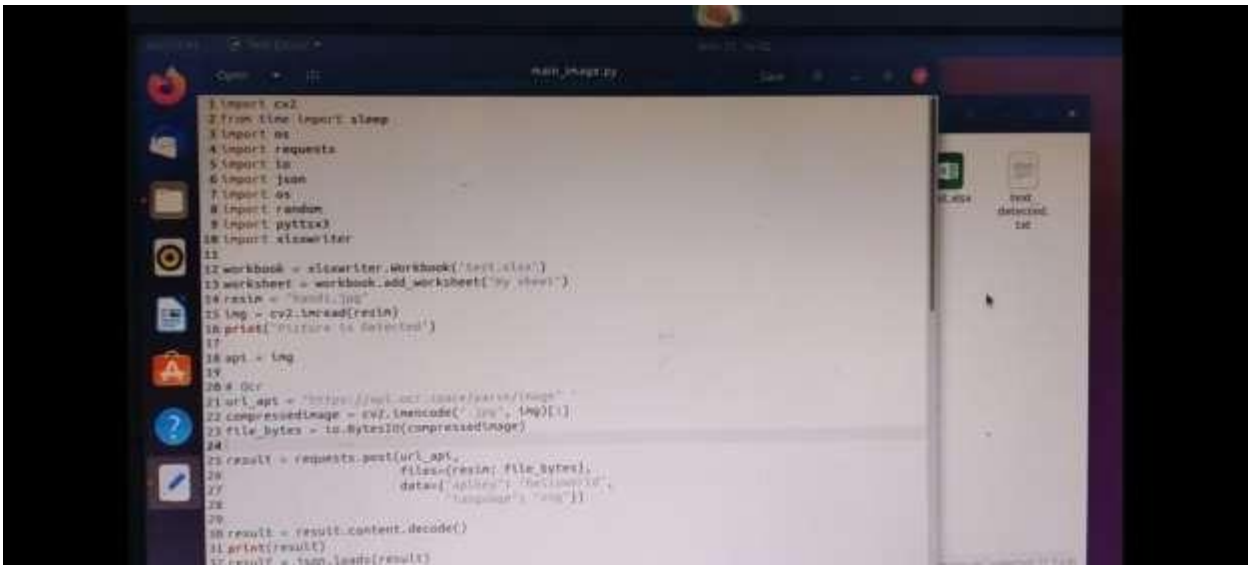
Input image:



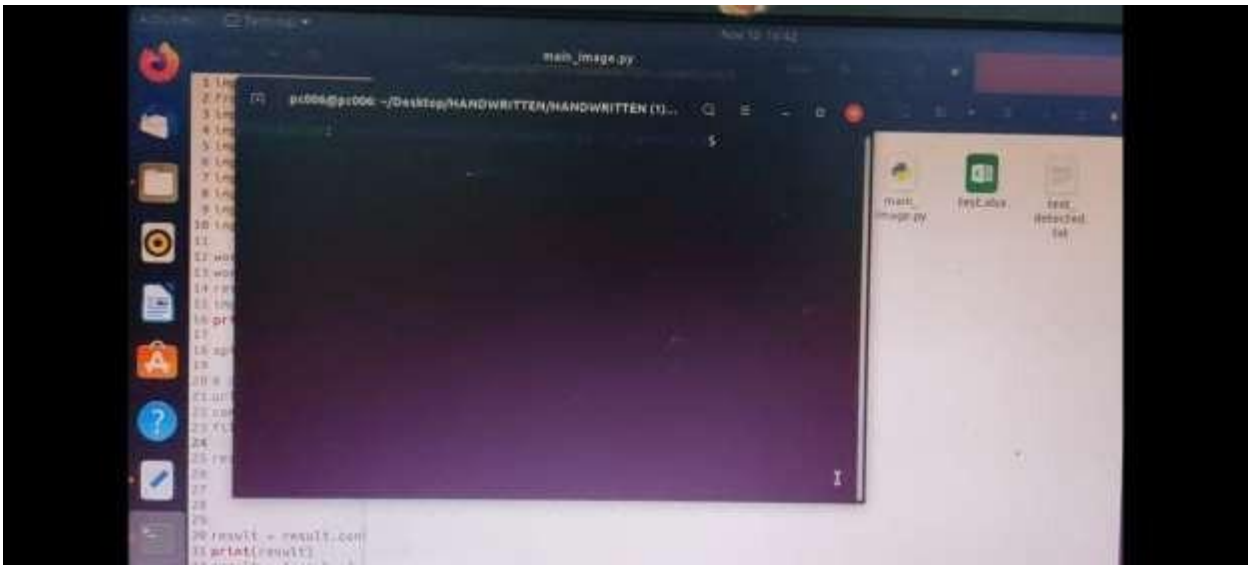
## Handwritten zip file:



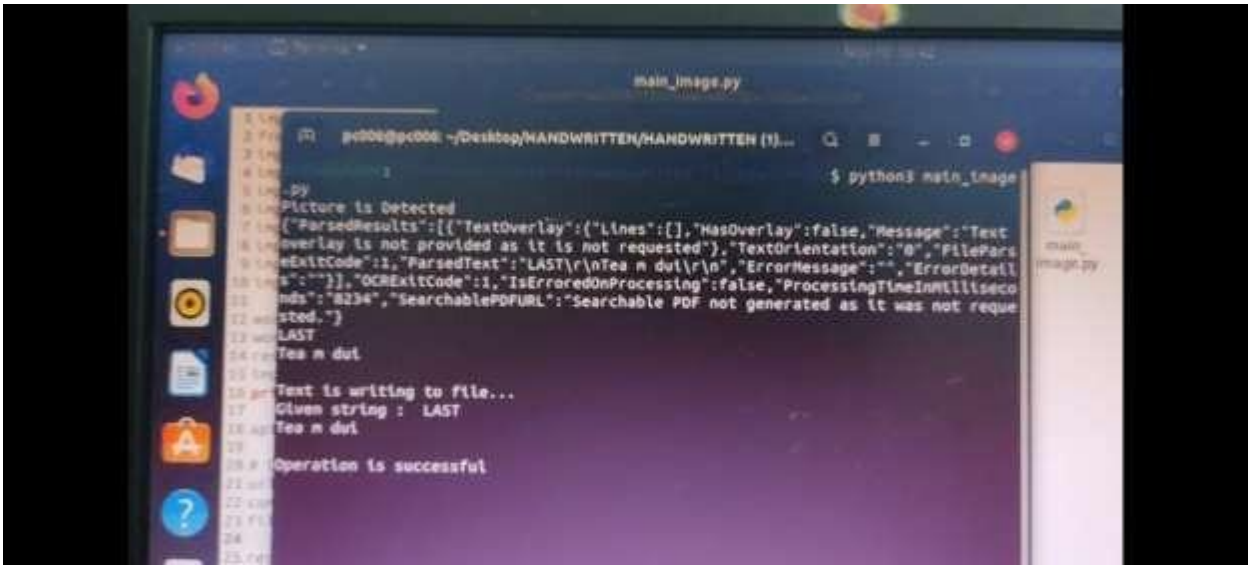

## Opened code in note pad:

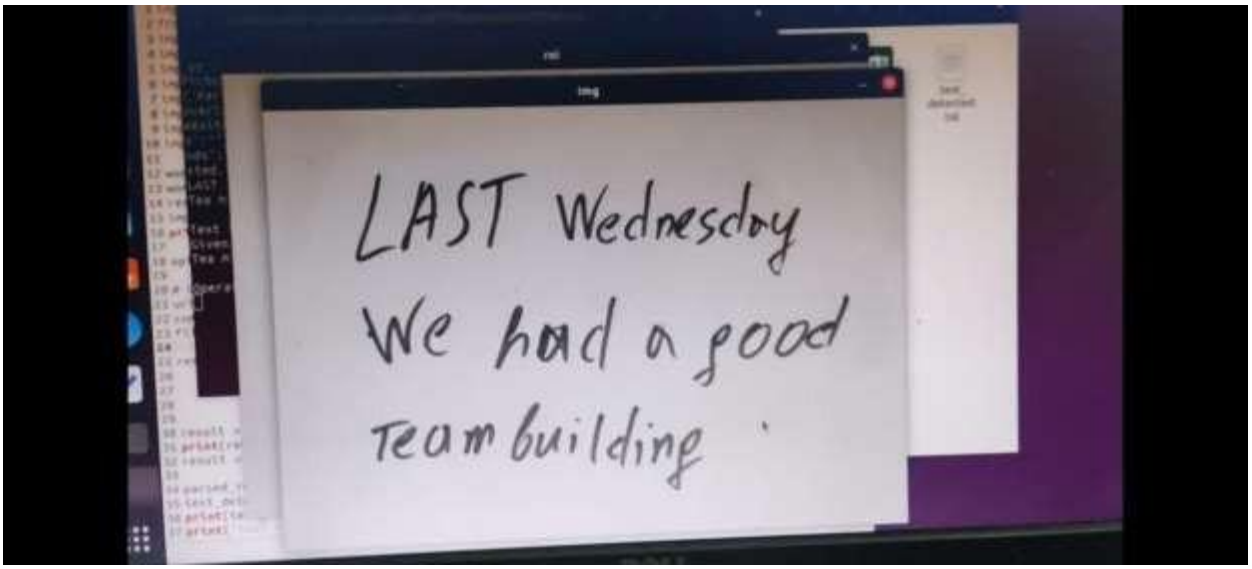



## Testing and run code using open CV




## Output screen shots:






**Results:**

After the analysis has been completed and validated, we can use it to classify the new digits drawn directly in the application. We can see the analysis also performs rather well on completely new and previously unseen data.

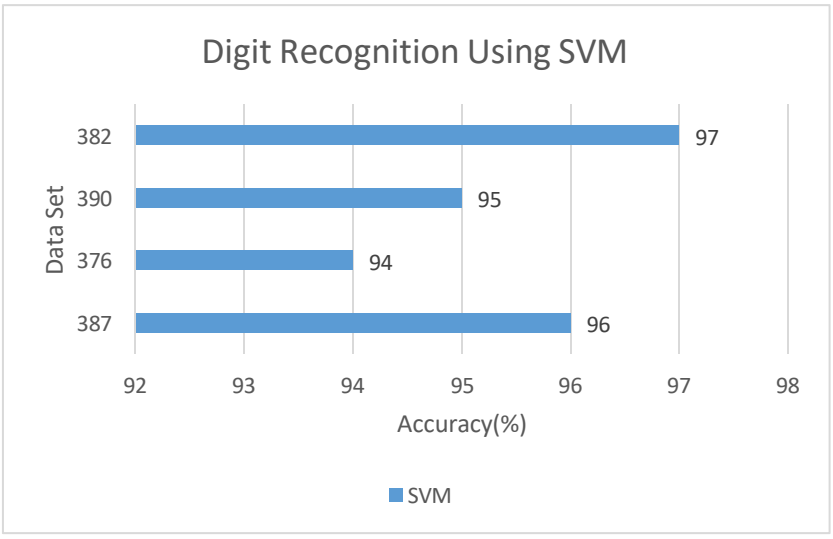



**Figure:** Graph representation of accuracy of SVM

## CONCLUSION

In this article, we detailed and explored how (Kernel) Support Vector Machines could be applied in the problem of handwritten digit recognition with satisfying results. The suggested approach does not suffer from the same limitations of Kernel Discriminant Analysis, and also achieves a better recognition rate. Unlike KDA, the SVM solutions are sparse, meaning only a generally small subset of the

training set will be needed during model evaluation. This also means the complexity during the evaluation phase will be greatly reduced since it will depend only on the number of vectors retained during training.

SVM model requires the most space since each support vector (SV) consist of many feature values . However, space saving can be achieved by storing only the original online signals and the pen-up/pen-down status corresponding to the SV in a compact manner. During recognition, the model will be expanded dynamically as required.Experiments using SVMs with probabilistic output were also performed on the same datasets for comparison. In many experiments, the results have shown that at character level, SVM recognition rates are significantly better due to structural risk minimization implemented by maximizing margin of separation in the decisionfunction. However, the increase in recognition rate isnot without some impact. The number of support vectors obtained in the training characterizes SVM model size.Storing these support vectors for recognition requires larger memory as compared to NN weights since each support vector is a multidimensional feature vector. The number of support vectors can be reduced by selecting better C and gamma parameter values through a finer grid search and by reduced set selection . The comparison of recognition results of SVM with probabilistic output and SVM distance output shows that both are comparable. In some datasets, SVM distance gives slightly higher while in some others the probabilistic output gives higher recognition rates.

## Future work

Future works on the database includes extending the character class collection by including all the presently used valid orthographic shapes for specific language script and creating word, line and page level collection of document images so that the researchers can focus on other stages of document recognition system as well.

It has been shown that Support Vector Machines (SVMs) can be applied to image and hand-written character recognition . However, SVMs don’t perform well in large datasets as the training time becomes cubic in the size of the dataset. This could be an issue as bigger datasets dataset containing thousand of samples which is quite large. To deal with this issue, a techniquecan be proposed ,which is to train a support vector machine on the collection of nearest neighbours in a solution they called ―SVM-KNN‖ . Training an SVM on the entire data set is slow and the extension of SVM to multiple classes is not as natural as Nearest Neighbor (NN).

However, in the neighbourhood of a small number of examples and a small number of classes, SVMs often perform better than other classification methods.

We can use NN as an initial pruning stage and perform SVM on the smaller but more relevant set of examples that require careful discrimination. This approach reflects the way humans perform coarse categorization: when presented with an image, human observers can answer coarse queries such as presence or absence of an animal in as little as 150ms, and of course, can tell what animal it is given enough time . This process of a quick categorization, followed by successive finer but slower discrimination was the inspiration behind the ―SVM-KNN‖ technique.

## References:


[1] X. Chen and A. L. Yuille, ―Detecting and reading text in natural scenes,‖ in Proc. Comput. Vision Pattern Rec-ognit., 2004, vol. 2, pp. II-366–II-373.

[2] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi, ―Text extraction and document image seg-mentation using matched wavelets and MRF model,‖ IEEE Trans Image Process., vol. 16, no. 8, pp. 2117– 2128,Aug. 2007.

[3] K. Kim, K. Jung, and J. Kim, ―Texture-based ap-proach for text detection in images using support vec-tor machines and continuously adaptive mean shift al-gorithm,‖ IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 12,pp. 1631–1639, Dec. 2003.

[4] N. Giudice and G. Legge, ―Blind navigation and the role of technology,‖in The Engineering Handbook of Smart Technology for Aging, Disability,and Indepen-dence, A. A. Helal, M. Mokhtari, and B. Abdulrazak, Eds. Hoboken, NJ, USA: Wiley, 2008.

[5] World Health Organization. (2009). 10 facts about blindness and visual impairment.

[6] Advance Data Reports from the National Health In-terview Survey (2008).

[7] International Workshop on Camera-Based Docu-ment Analysis and Recognition (CBDAR 2005, 2007, 2009, 2011).

[8] X. Chen, J. Yang, J. Zhang, and A. Waibel, ―Auto-matic detection and recognition of signs from natural scenes,‖ IEEE Trans. Image Process.,vol. 13, no. 1, pp. 87–99, Jan. 2004.

[9] D. Dakopoulos and N. G. Bourbakis, ―Wearable ob-stacle avoidance electronic travel aids for blind: A sur-vey,‖ IEEE Trans. Syst., Man, Cybern.,vol. 40, no. 1, pp. 25–35, Jan. 2010.

[10] B. Epshtein, E. Ofek, and Y. Wexler, ―Detecting text in natural scenes with stroke width transform,‖ in Proc. Comput. Vision Pattern Recognit.,2010, pp. 2963–2970.

[11] Y. Freund and R. Schapire, ―Experiments with a new boosting algorithm,‖in Proc. Int. Conf. Machine Learning, 1996, pp. 148–156.

[13] An overview of the Tesseract OCR (optical charac-ter recognition) engine, and its possible enhancement for use in Wales in a pre-competitive research stage Prepared by the Language Technologies Unit (Canol-fan Bedwyr), Bangor University April 2008.

[14] A. Shahab, F. Shafait, and A. Dengel, ―ICDAR 2011 robust reading competition:ICDAR Robust Reading Competition Challenge 2: Reading text in scene imag-es,‖ in Proc. Int. Conf. Document Anal. Recognit., 2011, pp. 1491–1496.

[15] KReader Mobile User Guide, knfb Reading Technol-ogy Inc. (2008).

[16] S. M. Lucas, ―ICDAR 2005 text locating competi-tion results,‖ in Proc.Int. Conf. Document Anal. Recog-nit., 2005, vol. 1, pp. 80–84.

THANK YOU..!