

HANDWRITTEN DIGIT RECOGNITION

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

by

Abhishek Kumar Singh

(Reg.No.37110008)

Alok Raj

(Reg.No.37110034)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

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

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119**

MARCH - 2020



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **ABHISHEK KUMAR SINGH (Reg. No. 37110008)** and **ALOK RAJ (Reg. No.37110034)** who carried out the project entitled "**HANDWRITTEN DIGIT RECOGNISATION**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Mrs. Dr.L.Mary Gladence, M.E.,Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I ABHISHEK KUMAR SINGH hereby declare that the Project Report entitled **“HANDWRITTEN DIGIT RECOGNISATION”** is done by me under the guidance of **Mrs. Dr. L. Mary Gladence, M.E., Ph.d.,** Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. Dr. L. Mary Gladence, M.E., Ph.D., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Digit Recognition is a noteworthy and important issue. As the manually written digits are not of a similar size, thickness, position and direction, in this manner, various difficulties must be considered to determine the issue of handwritten digit recognition. The uniqueness and assortment in the composition styles of various individuals additionally influence the example and presence of the digits. It is the strategy for perceiving and arranging transcribed digits. It has a wide range of applications, for example, programmed bank checks, postal locations and tax documents and so on.

The aim of this project is to implement a classification algorithm to recognize the handwritten digits. The after effects of probably the most broadly utilized Machine Learning Algorithms like SVM, KNN and RFC and with Deep Learning calculation like multilayer CNN utilizing Keras with Theano and Tensorflow. Utilizing these, the accuracy of 98.70% utilizing CNN (Keras + Theano) when contrasted with 97.91% utilizing SVM, 96.67% utilizing KNN, 96.89% utilizing RFC was obtained.

TABLE OF CONTENTS

	ABSTRACT	v
	LIST OF FIGURES	viii
CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 RESEARCH AND SIGNIFICANCE	9
2.	AIM AND SCOPE	12
	2.1 AIM OF PROJECT	12
	2.1.1 OBJECTIVES	12
	2.2 SCOPES	12
	2.2.1 ADVANTAGES	12
3.	SYSTEM DESIGN & METHODOLOGY	17
	3.1 EXISTING SYSTEM	17
	3.2 PRAPOSED SYSTEM	19
	3.3 REQUIREMENT SPECIFICCATION	20
	3.4 MODULES	21
4.	NAIVE BAYES ALGORITHM	23
	4.1 Naive Bayes algorithm works	24
	4.2 Pros and Cons of Naive Bayes	24
	4.3 Applications of Naive Bayes Algorithms	24
5.	SVM LINEAR CLASSIFIER	25
	5.1 Examples of SVMboundaries	25
	5.2 Data FlowDiagram	

6.	UML DIAGRAMS	30
	6.1 ACTIVITYDIAGRAM	32
	6.2 USE CASEDIAGRAM	
7.	LANGUAGE OVERVIEW	
8	RESULTS AND DISCUSSION	
REFERENCES		44
APPENDIX		
	A. PAPER ACCEPTANCEMAIL	46
	B. PLAGIARISMREPORT	47
	C. JOURNALPAPER	48
	D. SOURCECODE	59

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	BLOCK DIAGRAM	24
4.2	ACTIVITY DIAGRAM	28
5.1	HOME PAGE	34
5.2	NEWS FEED	35
5.3	LOG-IN PAGE	36
5.4	REGISTRATION FORM	37
5.5	CRIME REPORT FORM	38
5.6	MISSING VEHICLE FORM	39
5.7	ALERT PAGE	40
5.8	APPLICATION FORM	41
5.9	DASHBOARD	42

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics. Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can

be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step. The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer

to improve the algorithm(s) it uses to determine correct answers. For example, the MNIST dataset of handwritten digits. Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its

problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

1.2 DEEP LEARNING

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, and then that a network with a nonpolynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part. Most modern deep learning models are based on artificial neural networks, specifically convolutional neural networks (CNN)s, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.

1.3 CONVOLUTIONAL NEURAL NETWORK

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that scan the hidden layers and translation invariance characteristics. They have applications in image and video 4 recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

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

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the

animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

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

The time delay neural network (TDNN) was introduced in 1987 by Alex Waibel et al. and was the first convolutional network, as it achieved shift invariance. It did so by utilizing weight sharing in combination with Backpropagation training. Thus, while also using a pyramidal structure as in the neocognitron, it performed a global optimization of the weights

instead of a local one. TDNNs are convolutional networks that share weights along the temporal dimension. They allow speech signals to be processed time-invariantly. In 1990 Hampshire and Waibel introduced a variant which performs a two dimensional convolution. Since these TDNNs operated on spectrograms, the resulting phoneme recognition system was invariant to both shifts in time and in frequency. This inspired translation invariance in image processing with CNNs. The tiling of neuron outputs can cover timed stages. TDNNs now achieve the best performance in far distance speech recognition. In 1990 Yamaguchi et al. introduced the concept of max pooling. They did so by combining TDNNs with max pooling in order to realize a speaker independent isolated word recognition system. In their system they used several TDNNs per word, one for each syllable. The results of each TDNN over the input signal were combined using max pooling and the outputs of the pooling layers were then passed on to networks performing the actual word classification. A system to recognize hand-written ZIP Code numbers[35] involved convolutions in which the kernel coefficients had been laboriously hand designed. Yann LeCun et al. (1989) used back-propagation to learn the convolution kernel coefficients directly from images of hand-written numbers. Learning was thus fully automatic, performed better than manual coefficient design, and was suited to a broader range of image recognition problems and image types. This approach became a foundation of modern computer vision.

1.4 SUPER RESOLUTION CONVOLUTIONAL NEURAL

NETWORK Image super-resolution (SR) is to convert a given low-resolution (LR) image into a corresponding high-resolution (HR) image by an algorithm [1, 2, 3]. In practice, SR has a wide range of application scenarios and application prospects, such as video, security and medical imaging where demand more image details. In early, traditional methods of SR, including interpolation, sparse coding and other methods, require rigorous algorithm to extract engineered features. With the development of CNN, using convolutional neural network. SRCNN is the first method to apply CNN to image superresolution (SR) and achieves the state-of-the-art performance. It fully demonstrates the effectiveness of CNN for SR and is an end-to-end method. End-to-end means inputting a LR image and then outputting a HR image, unnecessary to care about the middle of the conversion process [4,5]. The methods for SR based on CNN, is only needing to give images and the architecture of network for training, which makes these methods have a wider range of applicability. So more and more researchers have begun to use deep learning to do the image studies, not only for SR, and have made a lot of progress. In the field of image super-resolution (SR), the depth of convolutional networks has increased from SRCNN's 3 layers to currently 20-30 layers [7, 8, and 9]. Recently many methods using CNN for SR have been proposed. Among them, VDSR shows the best performance. VDSR is proposed by Kim et al. in 2016 and the depth of its network is 20 layers [9]. VDSR uses cascading the same convolutional layers and learning residuals to reduce the super-parameters of networks and speed-up the training. Although the promotion of VDSR for SR is noticeable, but the amount of the network parameters is still huge. If continuing to increase the depth of the

network, we will face the serious growth of computing complexity, which makes the training of the network difficult to converge and even effects the performance. bad. As known, huge growth of the amounts of parameters is accompanied by over-fitting. We would like to proposed a strategy to reduce the amount of the network parameters, while the performance is as good as before. In the field of classification and recognition, deep learning methods have also take place of the traditional methods. And some methods based on CNN have using topology strategy to improve the architecture of networks

CHAPTER 2

AIM AND SCOPE

2.1 AIM OF PROJECT

The aim of this project is to implement a classification algorithm to recognize the handwritten digits. The after effects of probably the most broadly utilized Machine Learning Algorithms like SVM, KNN and RFC and with Deep Learning calculation like multilayer CNN utilizing Keras with Theano and Tensorflow. Utilizing these, the accuracy of 98.70% utilizing CNN (Keras + Theano) when contrasted with 97.91% utilizing SVM, 96.67% utilizing KNN, 96.89% utilizing RFC was obtained.

2.1.1 OBJECTIVES

Hand writing recognition of characters has been around since the 1980s. The task of handwritten digit recognition, using a classifier, has great importance and use such as – online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example - tax forms) and so on. There are different challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins. Our goal was to implement a pattern classification method to recognize the handwritten digits provided in the MNIST data set of images of hand written digits (0-9). The data set used for our application is composed of 300 training images and 300 testing images, and is a subset of the MNIST data set [1] (originally

composed of 60,000 training images and 10,000 testing images). Each image is a 28 x 28 grayscale (0-255) labeled representation of an individual digit.

2.2 SCOPE

PCA can be applied to economically represent the input digit images by projecting them onto a low- dimensional space constituted by a small number of basis images [3]. These basis images or the “eigendigits” are derived by finding the most significant eigenvectors of the pixel wise covariance matrix, after mean centering the data for each attribute. After projection, we use the 1-NN classifier to classify the digit in the low dimensional space. PCA reduce the dimensions of the dataset from 784 to a lower value – for ease of computation.

2.2.1 ADVANTAGES

- Faster than VDSR, FRCNN and PSNR
- Better textures
- Easy to implement
- Huge advantage in many industry Applications

2.3 Benefitting Disaster management and Image Processing Software's

Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image

more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbor procedure) provided by imaging packages use no a priori model of the process that created the image.

With image enhancement noise can effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a fluorescence microscope, resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object. The objective of image restoration techniques is to reduce noise and recover resolution loss. Image processing techniques are performed either in the image domain or the frequency domain.

Most images and video are transmitted or stored in lossy compressed formats. Some of these compression artifacts (e.g., blocking artifacts in JPEG) are quite visible. Reducing them may be thought of as an image restoration problem. These techniques are also known as “postprocessing of compressed images.” Other types of impairment are due to data transmission. For instance, blocks of image data may be lost due to packet loss in ATM networks over which the image is transmitted. Assuming the receiver knows which packets have been lost, the degradation model mapping the original x to the received y is again deterministic and many-to-one.

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 EXISTING SYSTEM

Image restoration techniques has many approaches to solve the problem with image degradation. But, commonly each model follows the Convolution network to solve the problem. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

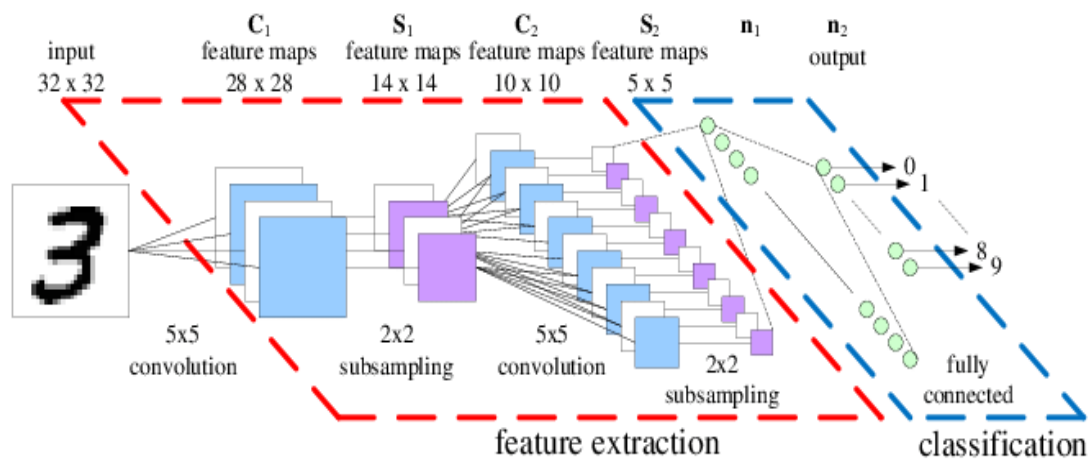
3.1.1 CONVOLUTIONAL LAYERS

In a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the convolution kernel/filter (the input channels) must equal the number channels (depth) of the input feature map.

The hyperparameters of the convolution operation, like padding size and stride. Convolutional layers convolve the input and pass its result to the next

layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is impractical for images. It would require a very high number of neurons, even in a shallow architecture, due to the very large input sizes associated 10 with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100×100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper. For example, regardless of image size, tiling 5×5 region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks

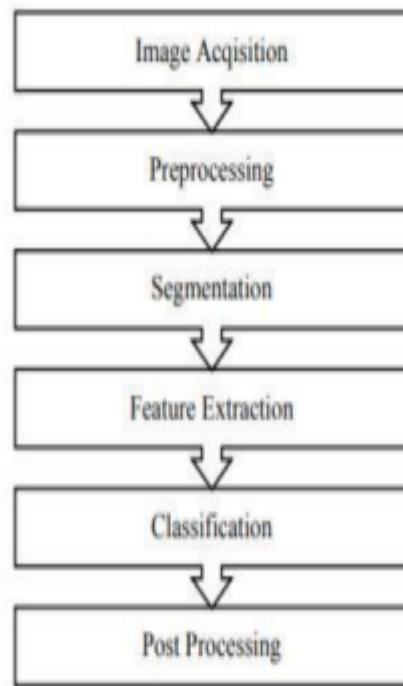


Layers of neural network (CNN)

3.1.2 Background

Generally Handwriting Character Recognition (HCR) is categorized into

six phases which are acquisition of image, pre-processing of enter image, segmentation, feature extraction, classification and put upprocessing.



Block Diagram of Handwriting Character Recognition

A. Image Acquisition

The input photo is supplied to the consciousness gadget at the Image Acquisition stage. The input can be either in an photograph layout such as JPEG, BMT, etc., or a scanned image, digital camera, or any other gorgeous digital input machine or can be taken from the canvas on the person interface.

B. Pre-Processing

The 2nd method, known as pre-processing, is the entry approach for personality cognizance and is very essential in finding out the focus quality. Preprocessing operates to normalize strokes and also to take away deviations that can minimize the accuracy rate. Preprocessing works usually on distinctive distortions such as irregular textual content size, missing points at some point of pen movement, jitters, left — proper bend and uneven spaces.

C. Segmentation

Segmentation is used to transform the enter representation of many characters to the individual characters. The methods used are the segmentation of words, strains and characters. Typically, it is carried out by way of isolating a single persona from a word picture. In addition, the contents are processed in a way that is like a tree. In the initial scenario, the line histogram is used to phase the lines. After that, each level, the characters are retrieved by a approach known as histogram, and subsequently they are retrieved.

D. Feature Extraction

The aim of the extraction characteristic is to permit the extraction of the sample that is most important for classification. Some of the Extraction Function techniques such as Principle Component Analysis (PCA), Scale Invariant Feature Extraction (SIFT), Linear Discriminant Analysis (LDA), Histogram, Chain Code (CC),

Zoning and Gradient-based applied sciences can also be used to get rid of the traits of character characters. All of these functions are used to educate the system. Each of the segmented pictures is taken with a pixel of dimension 28* 28

.By flattening the array into a vector of $28 * 28 = 784$ numbers, the photograph now converges to a minimal bunch of arrays in a quite high-quality structure 784-cell dimension. The photograph now turns into a n dimensional array tensor.

E. Classification

Decision-making takes vicinity at some point of the classification process. The extracted attributes are used to become aware of the characters. Different classifiers algorithms are used, such as SVM and Neural Networks. The classifiers sort the precise input function with reserved sample and find the best matching input classification for which Soft Max Regression is being used. Soft Max regression assigns each result with the probability so classification will become easy. This essentially incorporates all the proof this obtains through using components and then transforms it into the conceivable chances .

F. Post-Processing

The Post-processing is the last and ultimate phase of persona recognition. It is the procedure whereby herbal language is used to right the misclassified output. It procedures output by means of getting it after a recognition of the shape. If the shape is diagnosed basically then the accuracy can be increased in accordance to language knowledge. For exclusive handwriting inputs, shape recognizers behave differently.

Since 1998, researchers have been learning the hassle of handwritten digit focus with almost all the algorithms developed with the aid of then and even up till now. The rate of test blunders reduced from 12% in 1988 per linear classifier to 0.23% in 2012 through Convolutionary networks, and extra and extra facts scientists and computer studying experts are attempting to increase and validate unsupervised getting to know methods such as auto encoders and deep getting to know models.

The MNIST database consists of 70000 handwritten digitized numerals dispensed in ten different classes. For training purposes, the entire dataset is divided into 60,000 images, and the ultimate ten thousand is reserved for the check collection. In this work in the interval $[0,1]$, the gray level values of every pixel are coded the use of a price of 0 for white pixels and 1 for black pixel.

In the MNIST dataset, the records is already properly prepared: the pics have been founded in a 28x28 picture by computing the core of the pixel mass and translating the picture to role this point at the core of the 28x28 field. The coaching set consists of 30,000 patterns from SD-3 and 30,000 patterns from SD-1 and the take a look at set consisted of 5,000 SD-3 patterns, and 5,000 SD-1 patterns



Sample Dataset

3.1.3 REQUIREMENTS:

HARDWARE REQUIREMENTS:

Processor: i3 processor

RAM: 2 GB

Hard Disk: 1 TB

Monitor: 15" Color monitor

Key Board: 122 Keys

SOFTWARE REQUIREMENTS:

Operating System: Windows NT,

Windows 98,

Window 2000,

Window 2010,

Windows XP.

Language:Python

GUI:Tkinter

The system must be running Windows operating system and must meet the

following hardware requirements.

- For Windows 95 based computers, a 486 / 66 MHz or higher processor with 8MB
- For Windows 98 based computers, a 500/88MHz or higher processor with 32Mb of RAM.
- For Windows NT based computers, a 486 / 66 MHz or higher processor with 16 MB of RAM
- For Windows 200 based computers, a 700/850 MHz or higher processor with 512 MB of Ram

2.1.1 ROLE OF PYTHON

Python is a popular, general-purpose programming language with an emphasis on being readable and allowing programmers to use fewer lines of code to accomplish tasks than in older languages. Libraries such as NumPy, Matplotlib make it useful for scientific computing. Python is an excellent tool for anything for

four reasons:

- Open source
- Speed
- Support
- Scope

OPEN SOURCE

Python is free, open source, and is developed using a community-based model. It runs on Windows and Linux environments and can easily be ported to multiple platforms.

SPEED

Python is a high-level language, which means it has a number of

benefits that accelerate code development. The high-level character of Python makes prototyping ideas and code fast. This transparency eases both maintenance of the code (rewriting, finding bugs, etc.) and the process of adding to the code base in a multi-user development environment.

SUPPORT

Python is widely used for scientific computing in both academia and industry. As a consequence, a large number of useful analytics libraries are available (and well tested), including packages for numerical computing, data analysis, statistical analysis, visualization and machine learning.

SCOPE

Python supports object-oriented programming and advanced data structures such as lists, tuples, sets, dictionaries and so on. Also, matrix operations can be used with NumPy library and the package pandas supports data frames. Having these abilities within Python scope helps simplify and speed up data operations

2.TKINTER

Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter.tix and the tkinter.ttk modules.

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don't need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module `_tkinter` provides a thread safe mechanism which allows Python and Tcl to interact.

Tkinter's chief virtues are that it is fast, and that it usually comes bundled with Python. Although its standard documentation is weak, good material is available, which includes: references, tutorials, a book and others. tkinter is also famous for having an outdated look and feel, which has been vastly improved in Tk 8.5. Nevertheless, there are many other GUI libraries that you could be interested in. For more information about alternatives, see the Other Graphical User Interface Packages section. Python provides the standard library Tkinter for creating the graphical user interface for desktop-based applications.

Developing desktop-based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Importing tkinter is same as importing any other module in the python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x is 'tkinter'

Syntax: import tkinter

There are two main methods used you the user need to remember while creating the Python application with GUI.

1. Tk(screenName=None, baseName=None, className='Tk', useTk=1): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the

className to the desired one. The basic code used to create the main window of the application is:

Syntax:

m=tkinter.Tk() where m is the name of the main window object

2. **mainloop():** There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

Syntax:m.mainloop()

Example for creation of main window

```
from tkinter import *  
#creating the application main window  
top = Tk()  
#Entering the event main loop  
top.mainloop()
```

Tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:** It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method:** It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place()method:** It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application.

Some of the major widgets are explained below:

1. **Button:** To add a button in your application, this widget is used.

The general syntax is:

```
w=Button(master, option=value)
```

2. **Canvas:** It is used to draw pictures and other complex layout like graphics, text and widgets.

The general syntax is:

```
w = Canvas(master, option=value)
```

where, master is the parameter used to represent the parent window

3. **CheckButton:** To select any number of options by displaying a number of options to a user as toggle buttons.

The general syntax is:

```
w = CheckButton(master, option=value)
```

4. **Entry:** It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.

The general syntax is:

```
w=Entry(master, option=value)
```

5. **Label:** It refers to the display box where you can put any text or image which can be updated any time as per the code.

The general syntax is:

```
w=Label(master, option=value)
```

master is the parameter used to represent the parent window.

6. **Frame:** It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

```
w = Frame(master, option=value)
```

master is the parameter used to represent the parent window

7. **Text**: To edit a multi-line text and format the way it has to be displayed.

The general syntax is:

```
w =Text(master, option=value)
```

8. **MenuButton**: It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality.

The general syntax is:

```
w = MenuButton(master, option=value)
```

whereas, master is the parameter used to represent the parent win

9. **Message**: It refers to the multi-line and non-editable text. It works same as that of Label.

The general syntax is:

```
w = Message(master, option=value)
```

master is the parameter used to represent the parent window.

10. **Statusbar**: It is used to display the status of the window.

It is usually seen at the bottom of the root window.

The general syntax is:

```
status = Label(master, option=value)
```

```
status.Pack(fill=option=value)
```

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 Methodology (Using CNN Classifier)

Handwritten digit identification has come to be a subject of hobby among researchers. A giant variety of papers and articles about this situation are being posted in these days. It is proven in

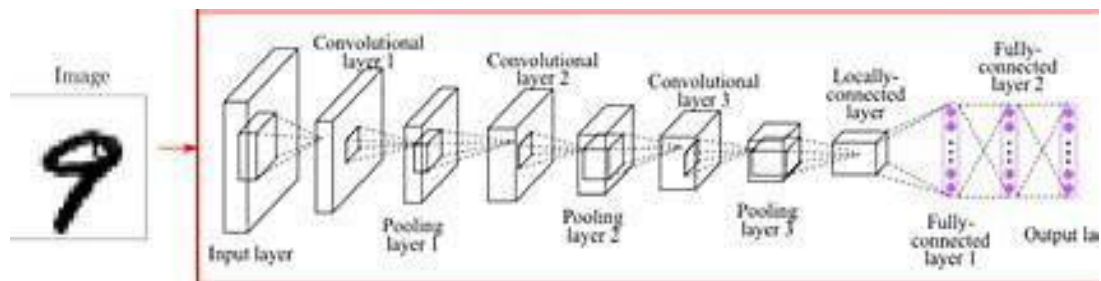
lookup that Deep Learning algorithms such as multilayer CNN the use of Keras with Tensorflow grant the absolute best accuracy compared to the most normally used machine getting to know algorithms such as SVM, KNN & RFC (Siddique et al., 2019). Because of its very best accuracy, Convolutional Neural Network (CNN) is being used for this project. Handwritten digit identification has come to be a subject of hobby among researchers. A giant variety of papers and articles about this situation are being posted in these days. It is proven in lookup that Deep Learning algorithms such as multilayer CNN the use of Keras with Tensorflow grant the absolute best accuracy compared to the most normally used machine getting to know algorithms such as SVM, KNN & RFC (Siddique et al., 2019). Because of its very best accuracy, Convolutional Neural Network (CNN) is being used for this project.

One of the earliest rankings was posted with the aid of (Lecun et al., 1998) themselves, which includes references up to 2012 (Baldominos et al., 2019). “It can be effortlessly validated that the early computer getting to know techniques used by way of (Lecun et al., 1998) protected linear classifiers (whose error rate stages from 7.6 to 12%), K-nearest neighbors procedures (K-NN, ranging from 1.1 to 5%), non-linear classifiers (about 3.5%), help vector machines (SVM, from 0.8 to 1.4%), neural networks (NN, from 1.6 to 4.7%) and convolutional neural networks (CNN, from 0.7 to 1.7%)”. It is notable that the additional facts results, are higher carried out through a Convolutionary neural network

besides distortion or preprocessing in (Lecun et al., 1998) used to be 0.95% (Baldominos et al., 2019). One of the earliest rankings was posted with the aid of (Lecun et al., 1998) themselves, which includes references up to 2012 (Baldominos et al., 2019). “It can be effortlessly validated that the early computer getting to know techniques used by way of (Lecun et al., 1998) protected linear classifiers (whose error rate stages from 7.6 to 12%), K-nearest neighbors procedures (K-NN, ranging from 1.1 to 5%), non-linear classifiers (about 3.5%), help vector machines (SVM, from 0.8 to 1.4%), neural networks (NN, from 1.6 to 4.7%) and convolutional neural networks (CNN, from 0.7 to 1.7%)”. It is notable that the additional facts results, are higher carried out through a Convolutionary neural network besides distortion or preprocessing in (Lecun et al., 1998) used to be 0.95% (Baldominos et al., 2019).

CNNs are beneficial mannequin category for both supervised and unsupervised gaining knowledge of paradigms, the place K-NN is used for unsupervised learning, and SVM is used for supervised learning. The supervised getting to know mechanism is the one the place the system enter and outputs (true labels) are identified and a mapping between the two is discovered. In the unsupervised gaining knowledge of mechanism, the proper labels for a given set of inputs are no longer known, and the mannequin ambitions to estimate the underlying distribution of data sample inputs (Khan et al., 2018).

The points of Convolutionary neural networks, together with the capability to extract information from multidimensional sources, provide them with a very useful option for solving laptop vision problems. Since the assignment is about extracting the elements from an photo and is associated to computer imaginative and prescient problems; hence, convolutional neural community is used(Baldominos et al., 2019)To recognize the handwritten digits, a seven-layered convolutional neural community with one input layer accompanied by hidden layers and two output layer is designed and illustrated below.



CNN Architecture

4.2 Algorithm

CNN object classification mannequin takes, analyzed and classifies an enter photo which in our case is digits under a positive category. Deep learning allows CNN fashions to be skilled and tested. photograph is transmitted via a collection of filters (kernels), pooling and Fully connected convolution layers and makes use of Softmax to classify an object with probabilistic values between zero and 1.

4.3 Layer's of CNN

CNN consists of a lot of layers. These layers when used repeatedly, lead to a formation of a Deep Neural Network. The fundamental types of layers used to build a CNN are:

1. Input

This layer holds the uncooked pixel values of photograph and convert it to grayscale pics using 28x28 matrix of pixels

.

2. Convolutional Layer

This layer gets the effects of the neuron layer that is linked to the enter regions. The wide variety of filters to be used in this layer is described here. Each filter may additionally be a 5x5 window that slider over the input records and receives the pixel with the most intensity as the output.

3. Rectified Linear Unit (ReLU) Layer

This layer applies an thing smart activation function on the picture

records and makes use of again propagation techniques. ReLU function is utilized in order to preserve the equal values of the pixels and not being changed by means of the returned propagation.

4. Pooling Layer

Down-sampling operation along the spatial dimensions (width, height), resulting in volume is utilized in this layer.

5. Fully Connected Layer

This layer is used to compute the score instructions that potential which class has the maximum score corresponding to the enter digits. The category label with the largest likelihood is chosen as the ultimate classification from the network and proven in the output.

4.4 Implementation

Tensorflow is used as backend in the application of this project. TensorFlow is an brilliant records circulation in the Machine Learning Library made by means of the Google Brain Team and made open supply in 2015. It is designed to ease the use and greatly relevant to each numeric and neural gadget troubles simply like different spaces. TensorFlow is essentially a low-level math-entangled tool that pursuits experts who apprehend what they're doing to construct exploratory studying structures, play around with them, and turn them into running programs. For the most part, it can be considered as a programming context in

which equations can be entitled as graphs. Math things to do are spoken by using nodes in the graph, and the edges include the multidimensional facts clusters (tensors) linked to them.

B. Python

Python is used for the duration of the implementation of assignment the place several traces of code had been brought in order to accomplish the assignment requirements. Python is typically used globally, and is a high-level programming language. It was once implemented in particular for application dominance, and its language shape allows software program engineers to carry thoughts in fewer traces of code. Python is a programming language which gives the chance to work shortly and more correctly organize frameworks.

C. Anaconda

Anaconda is used as IDE all through the implementation of the project. Anaconda is a free and open-source appropriation of the Python and R programming for logical figuring such as statistics science, AI applications, instruction of large-scale information, prescient investigation, etc. Anaconda accompanies over 1,400 programs just like the Conda package and digital surroundings director, Anaconda Navigator, so it takes the want to determine out how each library can be freely added.

D. Keras

Keras is used to build model to arrange the layers in the course of the

implementation of this project. Keras is a high-level neural community API written in Python that can run on top of TensorFlow, CNTK, or Theano. It used to be developed with a focal point on allowing for quickly experimentation. The key to doing true lookup is being capable to go from notion to result with the least delay viable.

Keras approves for handy and speedy prototyping (through person friendliness, modularity, and extensibility). Similarly, it supports each convolutional networks and recurrent networks, as properly as combinations of the two and runs seamlessly on CPU and GPU.

E. NumPy

NumPy is used for mathematical calculations to print out the predict records in this project. NumPy is the core bundle with Python for scientific computing. It is a versatile sophisticated (broadcasting) with N-dimensional array object characteristic software program for combining C / C++ and Fortran code, advantageous linear algebra, Fourier transform, and random number capabilities.

F. Matplotlib

Matplotlib is used to plot model accuracy and loss in a graphical view in this project. Matplotlib is a Python 2D plotting library that produces pleasant figures for the publication throughout platforms in a variety of hardcopy formats and interactive environments. Matplotlib can be used in Python scripts, Python and IPython shells, Jupyter notebook, Web software servers, and four

interface toolkits for graphical users.

CHAPTER 5

RESULTS AND DISCUSSION

Below are the steps to implement the handwritten digit recognition project:

1. Import the libraries and load the dataset

First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it. The `mnist.load_data()` method returns us the training data, its labels and also the testing data and its labels.

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape, y_train.shape)
```

2. Preprocess the data

The image data cannot be fed directly into the model so we need to perform some operations and process the data to make it ready for our neural network. The dimension of the training data is (60000,28,28). The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1).

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
```

convert class vectors to binary class matrices

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

3. Create the model

Now we will create our CNN model in Python data science project. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces over fitting of the model. We will then compile the model with the Adadelta optimizer.

```
batch_size = 128
num_classes = 10
epochs = 10
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.
optimizers.Adadelta(),metrics=['accuracy'])

```

4. Train the model

The `model.fit()` function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size.

It takes some time to train the model. After training, we save the weights and model definition in the 'mnist.h5' file.

```

hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test,
y_test))
print("The model has successfully trained")
model.save('mnist.h5')
print("Saving the model as mnist.h5")

```

5. Evaluate the model

We have 10,000 images in our dataset which will be used to evaluate how good our model works. The testing data was not involved in the training of the data therefore, it is new data for our model. The MNIST dataset is well balanced so we can get around 99% accuracy.

```

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

```

6. Create GUI to predict digits

Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function `predict_digit()` that

takes the image as input and then uses the trained model to predict the digit.

Then we create the App class which is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the `predict_digit()` function and display the result

Output:





REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [2] K. Fukushima, “Neocognitron for handwritten digit recognition,” *Neurocomputing*, vol. 51, pp. 161–180, Apr. 2003.
- [3] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
- [4] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
- [5] T. Kohonen, “Essentials of the self-organizing map,” *Neural Netw.*, vol. 37, pp. 52–65, Jan. 2013.
- [6] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.
- [7] M. Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” *Nature Neurosci.*, vol. 2, no. 11, pp. 1019–1025, Nov. 1999.
- [8] T. Serre, L. Wolf, and T. Poggio, “Object recognition with features inspired by visual cortex,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2005, pp. 994–1000.
- [9] J. Mutch and D. G. Lowe, “Object class recognition and localization using sparse features with limited receptive fields,” *Int. J. Comput. Vis.*, vol. 80, no. 1, pp. 45–57, Oct. 2008.
- [10] A. Coates and A. Y. Ng, “Learning feature representations with K-means,” in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 561–580.
- [11] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: A simple deep learning baseline for image classification?” *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.
- [12] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.
- [13] R. Hankins, Y. Peng, and H. Yin, “SOMNet: Unsupervised feature learning networks for image classification,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

- [14] S. Aly, "Learning invariant local image descriptor using convolutional Mahalanobis self-organising map," *Neurocomputing*, vol. 142, pp. 239–247, Oct. 2014.
- [15] E. Mohebi and A. Bagirov, "A convolutional recursive modified self organizing map for handwritten digits recognition," *Neural Netw.*, vol. 60, pp. 104–118, Dec. 2014.
- [16] H. Dozono, G. Niina, and S. Araki, "Convolutional self organizing map," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2016, pp. 767–771.
- [17] L. Elend and O. Kramer, "Self-organizing maps with convolutional layers," in *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization*, A. Vellido, K. Gibert, C. Angulo, and J. D. Martín Guerrero, Eds. Cham, Switzerland: Springer, 2020, pp. 23–32.
- [18] C. Ferles, Y. Papanikolaou, and K. J. Naidoo, "Denoising autoencoder self-organizing map (DASOM)," *Neural Netw.*, vol. 105, pp. 112–131, Sep. 2018.
- [19] M. Sakkari, R. Ejbali, and M. Zaied, "Deep SOMs for automated feature extraction and classification from big data streaming," in *Proc. 9th Int. Conf. Mach. Vis. (ICMV)*, Mar. 2017, Art. no. 103412L.
- [20] M. Sakkari and M. Zaied, "A convolutional deep self-organizing map feature extraction for machine learning," *Multimedia Tools Appl.*, early access, Mar. 6, 2020, doi: 10.1007/s11042-020-08822-9.
- [21] M. Wang, W. Zhou, Q. Tian, J. Pu, and H. Li, "Deep supervised quantization by self-organizing map," in *Proc. ACM Multimedia Conf. (MM)*, 2017, pp. 1707–1715.
- [22] M. Wang, W. Zhou, Q. Tian, and H. Li, "Deep scalable supervised quantization by self-organizing map," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 15, no. 3, pp. 1–18, Sep. 2019.
- [23] N. Liu, J. Wang, and Y. Gong, "Deep self-organizing map for visual classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–6.
- [24] C. S. Wickramasinghe, K. Amarasinghe, and M. Manic, "Deep selforganizing maps for unsupervised image classification," *IEEE Trans. Ind. Informat.*, vol. 15, no. 11, pp. 5837–5845, Nov. 2019.
- [25] S. Lawrence, C. L. Giles, A. Chung Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, 1997.
- [26] X. Tan, S. Chen, Z.-H. Zhou, and F. Zhang, "Recognizing partially occluded, expression

variant faces from single training image per person with SOM and softk-NN ensemble,” IEEE Trans. Neural Netw., vol. 16, no. 4, pp. 875–886, Jul. 2005.

[27] S. Aly, N. Tsuruta, R.-I. Taniguchi, and A. Shimada, “Visual feature extraction using variable map-dimension hypercolumn model,” in Proc. IEEE Int. Joint Conf. Neural Netw., IEEE World Congr. Comput. Intell., Jun. 2008, pp. 845–851.

APPENDIX

A. Paper AcceptanceMail

B. PlagiarismReport

HANDWRITTEN DIGIT RECOGNITION

ABSTRACT

- Introduce camera-based labels to assist daze individuals with learning the label names in items. The camera is a good thought to discover the item name picture or wood board. The picture is handled inside and the name is isolated from the picture utilizing the open CV library. At long last, the item is perceived and the brand name is articulated by voice.
- Now, it can perceive the utilization of tesseract library to change over the got picture of the name into text. At the point when the predetermined mark name is changed over to message and the changed over content is shown on the man-machine interface associated with the regulator. Presently, the changed over content ought to be changed over to discourse so the label name can be heard as discourse through earphones associated with the sound jack port utilizing the flite library.

OBJECTIVE

- We propose a camera-supported content refining asset to help discouraged individuals read text checks and heaps of hand-held articles in regular day to day existence.
- **Raspberry Pi** is an inserted PC comparable to a Visa used to make numerous undertakings unthinkable or incomprehensible previously.
- It will coordinate an assortment of advances like installed Linux, picture preparing and Optical character recognition (OCR).

EXISTING SYSTEM

- Blind individuals should utilize customary braille-like innovation to track and understand text, which is slow and powerful.
- The current OCR framework isn't computerized and requires the

utilization of a total PC, so it doesn't work.

- K Reader Mobile works on the telephone and permits clients to understand messages, receipts, handouts and numerous different archives.

PROPOSED SYSTEM

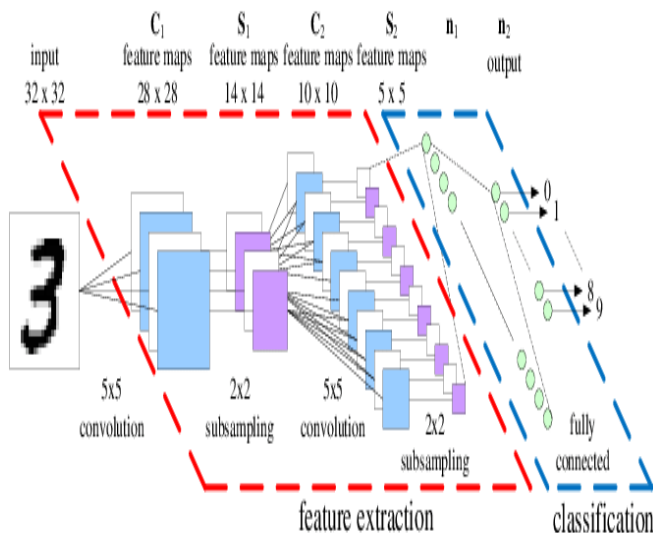
- A minimal effort robotized perusing framework will be accessible, which can change over printed course readings into computerized text, yet additionally read them as sound yield.
- Our proposed calculation can successfully oversee complex spaces and different examples and concentrate text data from close by items and images.

ADVANTAGES

- It is Effective.

- It is Low cost.
- Automatic character recognition.

SYSTEM ARCHITECTURE



LITERATURE SURVEY

- **A System-Prototype Representing 3D Space via Alternative-Sensing for Visually Impaired Navigation:** This article proposes how to speak with the closest 3-D space for the outwardly debilitated in free driving is a vital objective, including a few significant difficulties.
- **A System for Identification of Dangerous Spots and POIs for the Blind and Visually Impaired:** This article diagrams the way toward building up an arrangement that thinks about the territories of danger to the

outwardly debilitated and outwardly disabled worries in metropolitan zones.

- **A survey on wearable devices used to assist the visual impaired user navigation in outdoor environments:** This article leads a top to bottom investigation of wearable projects to help outwardly hindered clients use them in their day by day lives.

SOFTWARE REQUIREMENTS

- Linux Ubuntu
- Python
- Open cv

CONCLUSION

In this work, to improve the presentation of advanced penmanship acknowledgment, we investigated an assortment of convolutional neural organizations, which dodge complex pre-preparing, exorbitant component extraction, and complex joining techniques (classifier blend) of conventional acknowledgment frameworks. With broad investigation utilizing the MNIST information base, the current work has exhibited the part of different hyperparameters. We additionally affirmed

that appropriate hyperparameter change is fundamental to improve CNN's presentation. Utilizing the Adam enhancer of the MNIST information base, we acquired a 99.89% acknowledgment rate, which is superior to any recently revealed results. The examinations obviously show the impact of expanding the quantity of convolutional layers in the CNN structure in the exhibition of computerized acknowledgment physically. The advancement of this work is that it completely explores all the requirements of CNN development, which gives the best precision of MNIST information choice. Shared experts utilizing unadulterated CNN models can't coordinate this exactness. A few scientists are utilizing CNN's coordinated organization arrangements for a similar arrangement information to improve its exactness, yet because of extra PC costs and the intricacy of cutting-edge testing, the precision has arrived at this degree of exploration. Later on, we can examine CNN's various half breeds, to be specific the CNN mixture, specifically CNN-RNN and CNN-HMM models, just as mindfulness programs in a specific field. You can test transformative calculations to improve CNN learning boundaries, i.e., number of convolution channel layers, understanding rate, and portion size.

REFERENCES

- [1] E. Barnes, "Image recognition for shipping container tracking and I.D.," *Adv. Imag.*, vol. 10, no. 1, pp. 61–62, 1995.
- [2] M. S. Brown and W. B. Seales, "Document restoration using 3D shape: A general deskewing algorithm for arbitrarily warped documents," in *Proc. ICCV*, vol. 2, 2001, pp. 367–374.
- [3] Y. Cui and Q. Huang, "Character extraction of license plates from video," in *Proc. CVPR*, 1997, pp. 502–507.
- [4] D. Deng, K. P. Chan, and Y. Yu, "Handwritten Chinese character recognition using spatial Gabor filters and self-organizing feature maps," in *Proc. ICIP*, vol. 3, 1994, pp. 940–944.
- [5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1993.

Source code

1. Import the libraries and load the dataset

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

2. Preprocess the data

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

3. Create the model

```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(3,
3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

```

4. Train the model

```

hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
print("The model has successfully trained")

model.save('mnist.h5')
print("Saving the model as mnist.h5")

```

5. Evaluate the model

```

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

```

6. Create GUI to predict digits

```

from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np

model = load_model('mnist.h5')

def predict_digit(img):
    #resize image to 28x28 pixels
    img = img.resize((28,28))
    #convert rgb to grayscale
    img = img.convert('L')
    img = np.array(img)
    #reshaping to support our model input and normalizing
    img = img.reshape(1,28,28,1)
    img = img/255.0
    #predicting the class
    res = model.predict([img])[0]

```

```

return np.argmax(res), max(res)

class App(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)

self.x = self.y = 0

        # Creating elements
self.canvas = tk.Canvas(self, width=300, height=300, bg = "white",
cursor="cross")
self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
self.classify_btn = tk.Button(self, text = "Recognise", command =
self.classify_handwriting)
self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)

        # Grid structure
self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
self.label.grid(row=0, column=1, pady=2, padx=2)
self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
self.button_clear.grid(row=1, column=0, pady=2)

        #self.canvas.bind("<Motion>", self.start_pos)
self.canvas.bind("<B1-Motion>", self.draw_lines)

    def clear_all(self):
self.canvas.delete("all")

    def classify_handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas
rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas
im = ImageGrab.grab(rect)

        digit, acc = predict_digit(im)
self.label.configure(text= str(digit)+' ', '+ str(int(acc*100))+'%')

    def draw_lines(self, event):
self.x = event.x
self.y = event.y
        r=8
self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

app = App()
mainloop()

```