A Novel Method for Handwritten Digit Recognition System

INTRODUCTION

- ***Project Overview :***

- Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically.Because of the progress in the field of science and technology, everything is being digitalize to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI.

- ***By the end of this project you will:***

- Know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks

- Gain a broad understanding of image data.
- Work with Sequential type of modeling
- Work with Kera capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

- ***Purpose:***

- The biggest challenge for natural language processing systems is to accurately identify and classify the hand–written characters. Accurate Handwritten Character recognition is a challenging task for humans too as the style, size and other handwriting parameters may vary from human to human. Though a relatively straightforward machine vision task but improved accuracy as compared to the existing implementations is still desirable. This manuscript aims to propose a novel neural network based framework for handwritten character recognition. The proposed neural network based framework, transforms the raw data set to a umPy array to achieve image flattening and feeds the same into a pixel vector before feeding it into the network. In the neural network, the activation function is applied to transfer the resultant value to the hidden layer where it is further minimized through the use of minimized mean square and back propagation algorithms before applying a stochastic gradient on the resultant mini–batches. After a detailed study, the optimal algorithm for effective handwritten character recognition was proposed. Initially, the framework has been simulated only on digits . This manuscript aims to give the reader an insight into how the proposed neural network based framework has been applied for

handwritten digit recognition. It highlights the successful applications of the same while laying down the directions for the enhancements possible.

➤ ***LITERATURE SURVEY:***

- PAPER - 1TITLE: Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition.
- AUTHOR: Saleh Aly , (Associate Member, IEEE), And Sultan Almotairi
- YEAR OF PUBLICATION: 2020
- JOURNAL NAME: IEEE Access
- DESCRIPTION:
- Deep Convolutional Neural Networks (DCNN) are currently the predominant technique commonly used to learn visual features from images. However, the complex structure of most recent DCNNs impose two major requirements namely, huge labeled dataset and high computational resources. In this paper, we develop a new efficient deep unsupervised network to learn invariant image representation from unlabeled visual data. The proposed Deep Convolution Self-organizing Maps (DCSOM) network comprises a cascade of convolutional SOM layers trained sequentially to represent multiple levels of features. The 2D SOM grid is commonly used for either data visualization or feature extraction. However, this work employs high dimensional map size to create a new deep

network. The N-Dimensional SOM (ND-SOM) grid is trained to extract abstract visual features using its classical competitive learning algorithm. The topological order of the features learned from ND-SOM helps to absorb local transformation and deformation variations exhibited in the visual data. The input image is divided into an overlapped local patches where each local patch is represented by the N-coordinates of the winner neuron in the ND-SOM grid. Each dimension of the NDSOM can be considered as a non-linear principal component and hence it can be exploited to represent the input image using N-Feature Index Image (FII) bank. Multiple convolutional SOM layers can be cascaded to create a deep network structure. The output layer of the DCSOM network computes local histograms of each FII bank in the final convolutional SOM layer. A set of experiments using MNIST handwritten digit database and all its variants are conducted to evaluate the robust representation of the proposed DCSOM network. Experimental results reveal that the performance of DCSOM outperforms state-of-the-art methods for noisy digits and achieve a comparable performance with other complex deep learning architecture for other image variations. Deblur GAN-CNN: Effective Image De-noising and Recognition for Noisy Handwritten Characters AUTHOR.

- PAPER - 2 TITLE: HDSR-Flor: A Robust End-to-End System to Solve the Handwritten Digit String Recognition Problem in Real Complex Scenarios.

- AUTHOR: Arthur Flor De Sousa Neto, Byron Leite Dantas Bezerra (Member, IEEE),Estanislau Baptista Lima And Alejandro Héctor Toselli
- YEAR OF PUBLICATION: 2020
- JOURNAL NAME: IEEE Access
- DESCRIPTION:
- Automatic handwriting recognition systems are of interest for academic research fields and for commercial applications. Recent advances in deep learning techniques have shown dramatic improvement in relation to classic computer vision problems, especially in Handwritten Text Recognition (HTR). However several approaches try to solve the problem of deep learning applied to Handwritten Digit String Recognition(HDSR), where it has to deal with the low number of trainable data, while learning to ignore any writing symbol around the digits (noise). In this context, we present a new optical model architecture (Gated-CNN-BGRU), based on HTR workflow, applied to HDSR. The International Conference on Frontiers of Handwriting Recognition (ICFHR) 2014 competition on HDSR were used as baselines to evaluate the effectiveness of our proposal, whose metrics, datasets and recognition methods were adopted for fair comparison. Furthermore, we also use a private dataset (Brazilian Bank Check - Courtesy Amount Recognition), and 11 different approaches from the state-of-the-art in HDSR, as well as 2 optical models from the state-of-the-art in HTR. Finally, the proposed optical model demonstrated robustness even with low data volume (126

trainable data, for example), surpassing the results of existing methods with an average precision of 96.50%, which is equivalent to an average percentage of improvement of 3.74 points compared to the state-of the-art in HDSR. In addition, the result stands out in the competition's CVL HDS set, where the proposed optical model achieved a precision of 93.54%, while the best result so far had been from Beijing group (from the competition itself).

- PAPER 3Title: SARAYUT GONWIRAT , AND OLARIK SURINTA
- YEAR OF PUBLICATION: 2022
- JOURNAL NAME: IEEE ACCESS
- DESCRIPTION:
- Many problems can reduce handwritten character recognition performance, such as image degradation, light conditions, low-resolution images, and even the quality of the capture devices. However,in this research, we have focused on the noise in the character images that could decrease the accuracy of handwritten character recognition. Many types of noise penalties influence the recognition performance, for example, low resolution, Gaussian noise, low contrast, and blur. First, this research proposes a method that learns from the noisy handwritten character images and synthesizes clean character images using the robust de-blur generative adversarial network (Deblur GAN). Second, we combine the De-blur GAN architecture with a convolutional neural network (CNN), called

De-blur GANCNN. Subsequently, two state-of-the-art CNN architectures are combined with Deblur GAN, namely DeblurGAN-DenseNet121 and DeblurGAN-MobileNetV2, to address many noise problems and enhance the recognition performance of the handwritten character images. Finally, the De-blur GAN-CNN could transform the noisy characters to the new clean characters and recognize clean characters simultaneously. We have evaluated and compared the experimental results of the proposed Deblur GAN-CNN architectures with the existing methods on four handwritten character datasets: n-THI-C68, n-MNIST, THI-C68, and THCC-67. For the n-THI-C68 dataset, the De-blur GAN-CNN achieved above 98% and outperformed the other existing methods. For the n-MNIST, the proposed Deblur GAN-CNN achieved an accuracy of 97.59% when the AWGN+Contrast noise method was applied to the handwritten digits. We have evaluated theDeblur GAN-CNN on the THCC-67 dataset. The result showed that the proposed Deblur GAN-CNN achieved an accuracy of 80.68%, which is significantly higher than the existing method, approximately 10% .

- PAPER 4 TITLE
- PAPER - 4 Title: A Novel Learning Algorithm to Optimize Deep Neural Networks: Evolved Gradient Direction Optimizer (EVGO).
- AUTHOR: Ibrahim Karabayir , Oguz Akbilgic , and Nihat Tas
- YEAR OF PUBLICATION: 2020
- JOURNAL NAME: IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEM

- DESCRIPTION:
- Gradient-based algorithms have been widely used in optimizing parameters of deep neural networks' (DNNs) architectures. However, the vanishing gradient remains as one of the common issues in the parameter optimization of such networks. To cope with the vanishing gradient problem, in this article, we propose a novel algorithm, evolved gradient direction optimizer (EVGO), updating the weights of DNNs based on the first-order gradient and a novel hyperplane we introduce. We compare the EVGO algorithm with other gradient-based algorithms, such as gradient descent, RMSProp, Adagrad, momentum, and Adam on the well known Modified National Institute of Standards and Technology (MNIST) data set for handwritten digit recognition by implementing deep convolutional neural networks. Furthermore, we present empirical evaluations of EVGO on the CIFAR-10 and CIFAR-100 datasets by using the well-known AlexNet and ResNet architectures. Finally, we implement an empirical analysis for EVGO and other algorithms to investigate the behavior of the loss functions. The results show that EVGO outperforms all the algorithms in comparison for all experiments. We conclude that EVGO can be used effectively in the optimization of DNNs, and also, the pro-posed hyperplane may provide a basis for future optimization algorithms.

➤ ***IDEATION & PROPOSED SOLUTION:***

- ***Major Issues in the Project :***

- The problem statement is to classify handwritten digits. The goal is to take an image of a handwritten digit and determine what that digit.

- The comparison between these algorithms is carried out on the base of their delicacy, crimes,andtesting- training time collaborated by plots and maps that have been constructed using matplotlib for visualization.

- The handwritten digit recognition is the capability of computer applications to recognize the human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different shapes and sizes. The handwritten digit recognition. Convolutional Neural Network model created using PyTorch library over the MNIST dataset to recognize handwritten digits .

- Handwritten digit recognition system is a way to tackle this problem which uses the image of a digit and recognizes the digit .

- ***Problem Statement with Proposed Solutions:***

- What is the issue?

☆ Some digits may not recognized properly due to some errors.

- What does the problem affect?

☆ Handwriting recognition tends to have problems when it comes to accuracy. People can struggle to read others' handwriting.

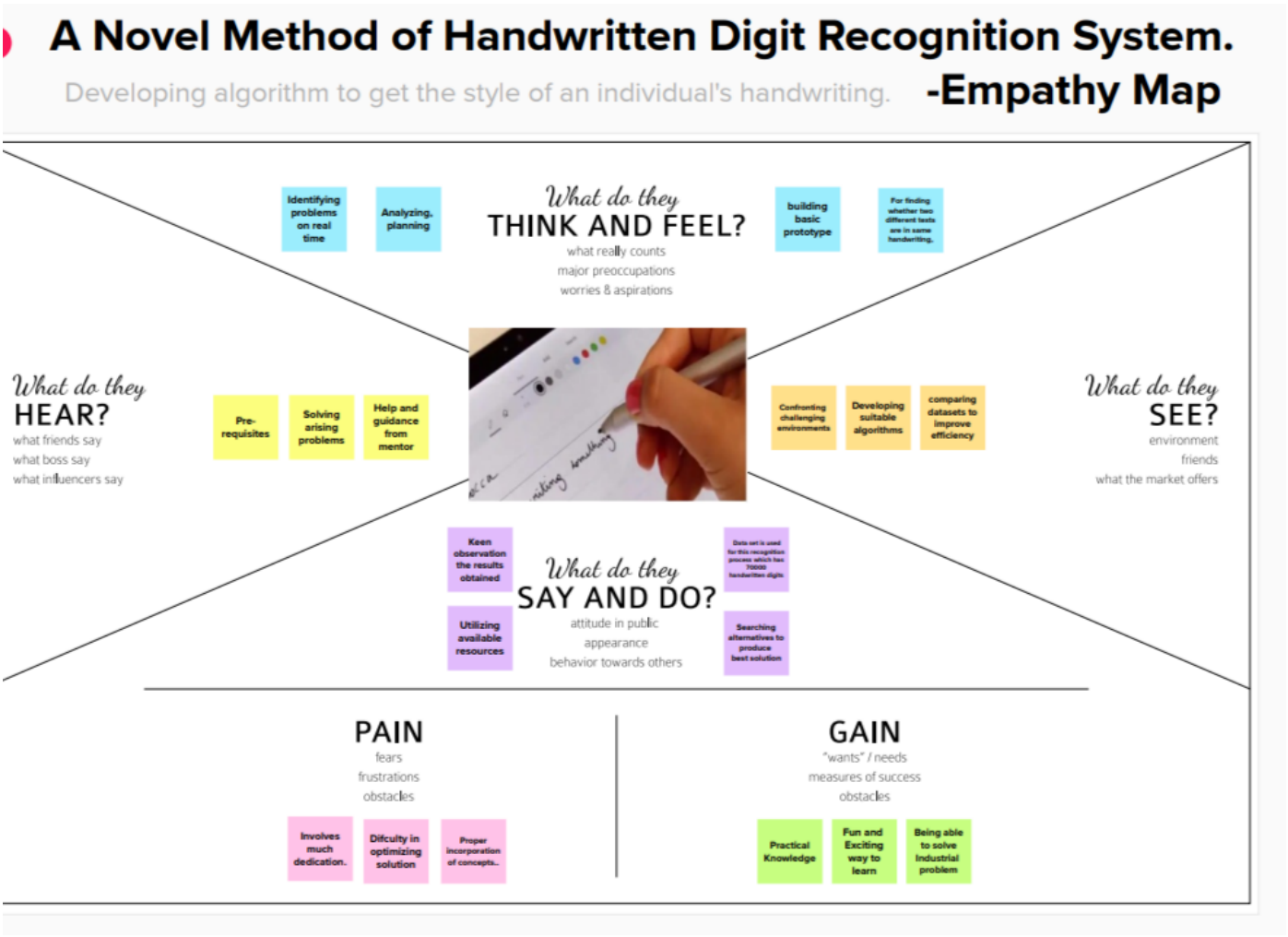
- How, then, is a computer going to do it?

☆ The issue is that there's a wide range of handwriting good and bad.

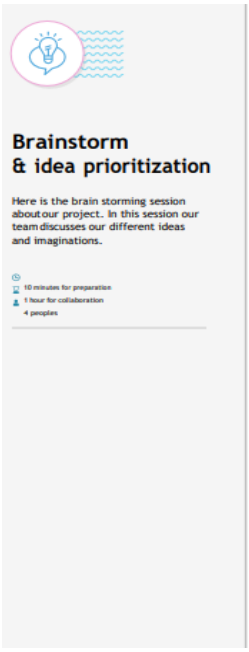
- Why is it important that we fix the problem?

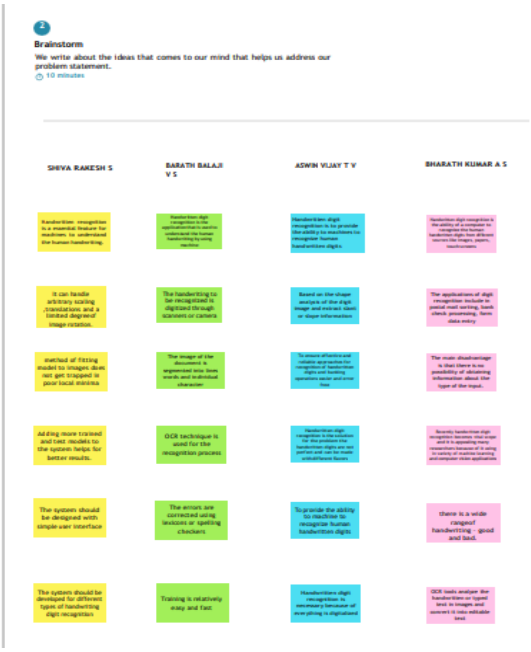
☆ The high variance in handwriting styles across people and poor quality of the handwritten text compared to printed text pose significant hurdles in converting it to machine readable text. The Complex problem to solve for multiple industries like healthcare, insurance and banking.

➤ ***Empathy Map:***



➤ ***Brain Stroming:***





➤ ***REQUIREMENT ANALYSIS:***

- We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for server side scripting.

- The model is built in the notebook Hand written recognition train.py We need the model which is saved and the saved model in this content is mnist The static folder will contain css and jquery files.

- The templates mainly used here are main.html and index6.html for showcasing the UI.

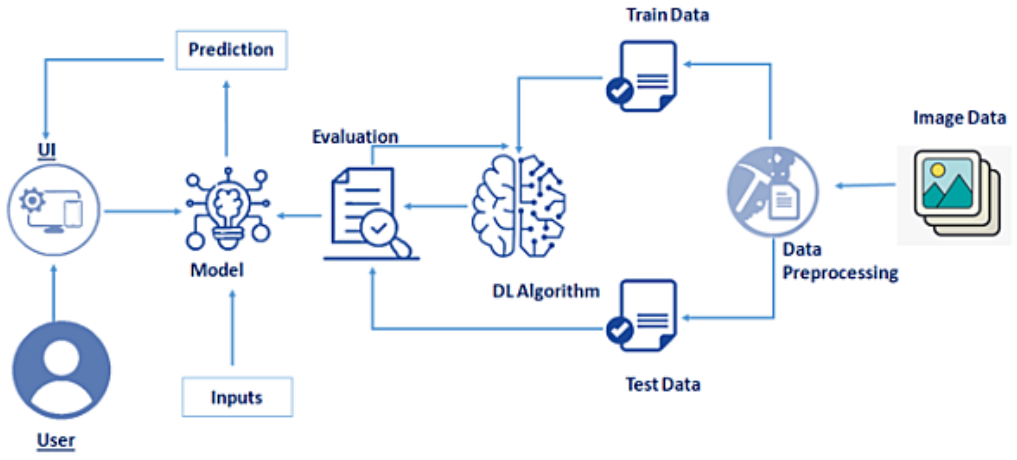
➤ ***PROJECT DESIGN :***

- Pattern recognition system consists of two-stage process. The first stage is feature extraction and the second stage is classification. Feature extraction is the measurement on a population of entities that will be classified. This assists the classification stage by looking for features that allows fairly easy to distinguish between the different classes. Several different features have to be used for classification. The set of features that are used makes up a feature vector, which represents each member of the population. Then, Pattern recognition system classifies each member of the population on the basis of information contained in the feature vector. The following is an example of feature vectors that have been plotted on a graph.

- Bayesian decision theory. The Bayesian decision theory is a system that minimizes the classification error. This theory plays a role of a prior. This is when there is priority information about something that we would like to classify. For example, suppose we do not know much about the fruits in the conveyer belt. The only information we

know is that 80% of the fruit in the conveyer belt are apples, and the rest of them are oranges. If this is the only information we have, then we can classify that a random fruit from the Conveyer belt is apple. In this case, the prior information is the probability of either an apple or an orange is in the conveyer belt. If we only have so little information, then we would have the following rule: Decide”apple‘if P (apple) > P (orange), otherwise decide”orange‘ Here, P (apple) is the probability of being an apple in the conveyer belt. This means that P(apple) = 0.8 (80%). This is probably strange, because if the above rule is used, then we are classifying a random fruit as an apple. But if we use this rule, we will be right 80% of the time. This is a simple example and can be used to understand the basic idea of pattern recognition. In real life, there will be a lot more information given about things that we are trying to classify. For example, we know that the color of the apples is red. Therefore if we can observe a red fruit, we should be able to classify it as an apple. We can have the probability distribution for the color of apples and oranges. Let wapp represent the state of nature where the fruit is an apple, let wora represent the state of nature where the fruit is an orange and let x be a continuous random variable that represents the color of a fruit. Then we can have the expression p(x|wapp) representing the density function for x given that the state of nature is an apple. In a typical problem, we would be able to calculate the conditional densities p (x|wj) for j so it will be either an apple or an orange. We would also know the prior probabilities P(wapp) and P(wora). These represent the total number of apples versus oranges in the conveyer belt. Here we are looking for a formula that will tell us about the probability of a fruit being an apple or an orange just by observing a certain color x. If we have the probability, then for the given color that we observed, we can classify the fruit by comparing it to the probability that an orange had such a color versus the probability that an apple had such a color. If we were more certain that an apple had such a color, then the fruit would be

classified as an apple. So, we can use Baye's formula, which states the following: P(wj |x) = p(x|wj) P(wj)/p(x) What the formula means is that using a prior information, we can calculate the a posterior probability of the state of nature being in state w hat we have given that the feature value x has been measured. So, if we observe a certain x for a random fruit in the conveyer belt, then by calculating P(wapp/x) and P(worg /x). we would decide that the fruit is apple if the first value is greater than the second one and if P(worg/x) is org greater, then we would decide that the fruit is orange. So, the Bayesian decision rule can be stated as: Decide worg if P(worg |x) > P(wapp |x), otherwise, decide wapp , Since p(x) occurs on both sides of the comparison, the rule can also be equivalent to the following rule: Decide worg if p(x|worg)P(worg) > p(x|wapp)P(wapp), otherwise decide wapp The following graph shows the a posterior probabilities for the two-class decision problem. For every x, the posteriors has to sum to 1. The red region on the x axes represents the values for x for which would decide as ”apple‘. The orange region represents values for x for which would decide as ”orange‘.



➤ ***PROJECT PLANNING & SCHEDULING:***

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	15 October 2022
Team ID	PNT2022TMID24826IBM
Project Name	Project - A novel Method for Handwritten Digit Recognition System
Maximum Marks	8 Marks

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	15 Oct 2022	20 Oct 2022	20	20 Oct 2022
Sprint-2	20	6 Days	21 Oct 2022	25 Oct 2022	20	25 Oct 2022
Sprint-3	20	4 Days	26 Nov 2022	30 Nov 2022	20	30 Oct 2022
Sprint-4	20	3 Days	01 Nov 2022	05 Nov 2022	20	05 Nov 2022

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Pattern Recognition	USN-1	As a user, I can recognize the pattern of the digits which I am entering by using two different methods.	2	High	Roshan Akthar, Sanjeev, Yogeswaran, Santhosh Kumar
Sprint-2	Handwritten Character Recognition	USN-2	As a user, I can recognize a handwritten character by using the concept of Neural Network	1	High	Roshan Akthar, Sanjeev, Yogeswaran, Santhosh Kumar
Sprint-3	Digit Recognition	USN-3	As a user, I can recognize the digits which are handwritten using the Neural Network	2	Low	Roshan Akthar, Sanjeev, Yogeswaran, Santhosh Kumar
Sprint-4	Simulation	USN-4	As a user, I can recognize the handwritten digits and stimulate it accordingly	2	Medium	Roshan Akthar, Sanjeev,

➤ ***CODING & SOLUTIONING (Explain the features added in the project along with code)***

➤ ***Code 1:***

```
 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
	<meta http-equiv="content-type" content="text/html; charset=windows-1252"/>
	<title></title>
	<meta name="generator" content="Neat Office 6.2.8.2 (Windows)"/>
	<meta name="created" content="00:00:00"/>
	<meta name="changed" content="00:00:00"/>
	<style type="text/css">
	 @page { size: 21cm 29.7cm; margin: 2cm }
	 p { margin-bottom: 0.25cm; line-height: 115%; background: transparent }
	 pre { background: transparent }
	 pre.western { font-family: "Liberation Mono", monospace; font-size: 10pt }
	 pre.cjk { font-family: "NSimSun", monospace; font-size: 10pt }
	 pre.ctl { font-family: "Liberation Mono", monospace; font-size: 10pt }
	</style>
</head>
<body lang="en-IN" link="#000080" vlink="#800000" dir="ltr"><pre class="western">&lt;!DOCTYPE
html&gt;
&lt;html lang=&quot;en&quot;&gt;

&lt;head&gt;
   &lt;meta charset=&quot;UTF-8&quot;&gt;
   &lt;meta http-equiv=&quot;X-UA-Compatible&quot; content=&quot;IE=edge&quot;&gt;
   &lt;meta name=&quot;viewport&quot; content=&quot;width=device-width, initial-
scale=1.0&quot;&gt;
   &lt;title&gt;Handwritten Recognition System&lt;/title&gt;
   &lt;link rel=&quot;stylesheet&quot; href=&quot;style.css&quot;&gt;
&lt;style&gt;
* {
   margin: 0;
   padding: 0;
```

```
    box-sizing: border-box;
}

body {
    background-image: url(db-bg.jpg);
    background-repeat: no-repeat;
    background-size: cover;
    height: 100vh;
}

.header {
    background-color: blue;
    opacity: 0.9;
    font-size: 30px;
    padding: 20px;
    position: sticky;
}

.navbar {
    text-decoration: none;
}

ul {
    display: flex;
    flex-direction: row;
    justify-content: flex-end;
    gap: 20px;
    list-style-type: none;
}

a {
    color: white;
    letter-spacing: 1px;
    text-decoration: none;
    padding: 10px;
    font-weight: 700;
}

a:hover {
    color: darkblue;
    cursor: pointer;
```

```
}

.main {
    margin: 50px;
}

.main-heading {
    color: whitesmoke;
    text-align: center;
    letter-spacing: 1.5;
    margin: 50px 0 0px;
}

.content {
    color: white;
    font-size: 30px;
    font-weight: 500;
    text-align: center;
    line-height: 1.5;
    margin-top: 150px;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
    &lt;header class=&quot;header&quot;&gt;
        &lt;nav class=&quot;navbar&quot;&gt;
            &lt;ul&gt;
                &lt;li&gt;
                    &lt;a href=&quot;#&quot;&gt;Home&lt;/a&gt;
                &lt;/li&gt;
                &lt;li&gt;
                    &lt;a href=&quot;second.html&quot;&gt;Recognize&lt;/a&gt;
                &lt;/li&gt;
            &lt;/ul&gt;
        &lt;/nav&gt;
    &lt;/header&gt;

    &lt;div class=&quot;bg-pic&quot;&gt;&lt;/div&gt;

    &lt;main class=&quot;main&quot;&gt;
```

<h1 class="main-heading">Handwritten Recognition System</h1>

<p class="content">

Handwritten Text Recognition is a technology that is much needed in this world as of today. This digit
Recognition system is used to recognize the digits from different sources like emails, bank cheque,
papers, images, etc. Before proper implementation of this technology we have relied on writing texts
with our own hands which can result in errors. It's difficult to store and access physical data with
efficiency. The project presents recognizing the handwritten digits (0 to 9) from the famous MNIST
dataset. Here we will be using artificial neural networks convolution neural network.

</p>
</main>
</body>

</html></pre>
</body>
</html>

- ***Code 2:***

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
	<meta http-equiv="content-type" content="text/html; charset=windows-1252"/>
	<title></title>
	<meta name="generator" content="Neat Office 6.2.8.2 (Windows)"/>
	<meta name="created" content="00:00:00"/>
	<meta name="changed" content="00:00:00"/>
	<style type="text/css">
	 @page { size: 21cm 29.7cm; margin: 2cm }
	 p { margin-bottom: 0.25cm; line-height: 115%; background: transparent }
	 pre { background: transparent }
	 pre.western { font-family: "Liberation Mono", monospace; font-size: 10pt }
	 pre.cjk { font-family: "NSimSun", monospace; font-size: 10pt }
	 pre.ctl { font-family: "Liberation Mono", monospace; font-size: 10pt }
```

```
        </style>
</head>
<body lang="en-IN" link="#000080" vlink="#800000" dir="ltr"><pre class="western">&lt;!DOCTYPE
html&gt;
&lt;html lang=&quot;en&quot;&gt;

&lt;head&gt;
   &lt;meta charset=&quot;UTF-8&quot;&gt;
   &lt;meta http-equiv=&quot;X-UA-Compatible&quot; content=&quot;IE=edge&quot;&gt;
   &lt;meta name=&quot;viewport&quot; content=&quot;width=device-width, initial-
scale=1.0&quot;&gt;
   &lt;title&gt;Digit Recognition&lt;/title&gt;
   &lt;link rel=&quot;stylesheet&quot; href=&quot;recognize.css&quot;&gt;

&lt;style&gt;
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}

body {
   background-image: url(&quot;bg-img.jpg&quot;);
   background-repeat: no-repeat;
   background-size: cover;
   width: 100%;
   height: 100vh;
}

.header {
   font-size: 30px;
   padding: 20px;
   background-color: white;
   width: 100%;
   opacity: 0.9;
}

.navbar {
   text-decoration: none;
}
```

```
ul {
    display: flex;
    flex-direction: row;
    justify-content: flex-end;
    gap: 20px;
    list-style-type: none;
}

a {
    color: black;
    letter-spacing: 1px;
    text-decoration: none;
    padding: 20px;
    font-size: 30px;
    font-weight: 600;
}

a:hover {
    color: darkcyan;
    cursor: pointer;
}

.main {
    margin: 70px;
}

.main-heading {
    color: darkcyan;
    letter-spacing: 1.5px;
    margin-bottom: 20px;
}

.flex-btn {
    display: flex;
    flex-direction: row;
    gap: 10px;
}

label {
    background-color: darkcyan;
    color: white;
```

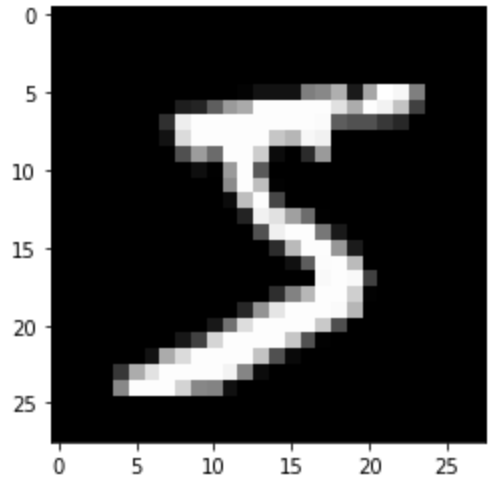
```
    padding: 10px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
}

.recognize-btn {
    border: none;
    padding: 10px;
    border-radius: 3px;
    background-color: darkcyan;
    color: white;
}

label:hover,
.recognize-btn:hover {
    cursor: pointer;
    background-color: lightblue;
    color: darkblue;
}

input {
    margin-top: 1rem;
}

input[type=&quot;file&quot;] {
    z-index: -1;
    position: absolute;
    opacity: 0;
}

input:focus+label {
    outline: 4px solid;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
    &lt;header class=&quot;header&quot;&gt;
        &lt;nav class=&quot;navbar&quot;&gt;
            &lt;ul&gt;
```

```
            &lt;li&gt;
              &lt;a href=&quot;first.html&quot;&gt;Home&lt;/a&gt;
            &lt;/li&gt;
            &lt;li&gt;
              &lt;a href=&quot;#&quot;&gt;Recognize&lt;/a&gt;
            &lt;/li&gt;
          &lt;/ul&gt;
        &lt;/nav&gt;
    &lt;/header&gt;

    &lt;main class=&quot;main&quot;&gt;
        &lt;h1 class=&quot;main-heading&quot;&gt;Digit Recognition&lt;/h1&gt;
        &lt;br&gt;
        &lt;div class=&quot;flex-btn&quot;&gt;
            &lt;input type=&quot;file&quot; id=&quot;file-upload&quot; multiple required /&gt;
            &lt;label for=&quot;file-upload&quot;&gt;Choose&lt;/label&gt;
            &lt;div id=&quot;file-upload-filename&quot;&gt;&lt;/div&gt;
            &lt;br&gt;&lt;br&gt;
            &lt;button class=&quot;recognize-btn&quot;&gt;Recognize&lt;/button&gt;
        &lt;/div&gt;
    &lt;/main&gt;

    &lt;script src=&quot;recognize.js&quot;&gt;&lt;/script&gt;

&lt;script&gt;
var input = document.getElementById('file-upload');
var infoArea = document.getElementById('file-upload-filename');

input.addEventListener('change', showFileName);

function showFileName(event) {
    var input = event.srcElement;
    var fileName = input.files[0].name;
    infoArea.textContent = 'File name: ' + fileName;
}
&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
</body>
</html>
```

➤ ***TESTING:***

➤ In Handwritten Digit Recognition System, the testing is done by the following code in the Jupiter notebook:

```
import cv2
import numpy as np
from keras.datasets import mnist
from keras.layers import Dense, Flatten, MaxPooling2D, Dropout
from keras.layers.convolutional import Conv2D
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

(X_train, y_train), (X_test, y_test) = mnist.load_data()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step

plt.imshow(X_train[0], cmap="gray")
plt.show()
print (y_train[0])
```




```
5
```

In [4]:

```
print ("Shape of X_train: {}".format(X_train.shape))
print ("Shape of y_train: {}".format(y_train.shape))
print ("Shape of X_test: {}".format(X_test.shape))
print ("Shape of y_test: {}".format(y_test.shape))
```

```
Shape of X_train: (60000, 28, 28)
Shape of y_train: (60000,)
Shape of X_test: (10000, 28, 28)
Shape of y_test: (10000,)
```

In [5]:

```
# Reshaping so as to convert images for our model
X_train = X_train.reshape(60000, 28, 28, 1)
X_test = X_test.reshape(10000, 28, 28, 1)
```

In [6]:

```
print ("Shape of X_train: {}".format(X_train.shape))
print ("Shape of y_train: {}".format(y_train.shape))
print ("Shape of X_test: {}".format(X_test.shape))
print ("Shape of y_test: {}".format(y_test.shape))
```

```
Shape of X_train: (60000, 28, 28, 1)
Shape of y_train: (60000,)
Shape of X_test: (10000, 28, 28, 1)
Shape of y_test: (10000,)
```

In [7]:

```
#one hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [8]:

```
model = Sequential()

## Declare the layers
layer_1 = Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,
28, 1))
layer_2 = MaxPooling2D(pool_size=2)
layer_3 = Conv2D(32, kernel_size=3, activation='relu')
layer_4 = MaxPooling2D(pool_size=2)
layer_5 = Dropout(0.5)
layer_6 = Flatten()
layer_7 = Dense(128, activation="relu")
layer_8 = Dropout(0.5)
layer_9 = Dense(10, activation='softmax')

## Add the layers to the model
model.add(layer_1)
model.add(layer_2)
model.add(layer_3)
model.add(layer_4)
model.add(layer_5)
model.add(layer_6)
```

```
model.add(layer_7)
model.add(layer_8)
model.add(layer_9)
```

In [9]:

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

In [10]:

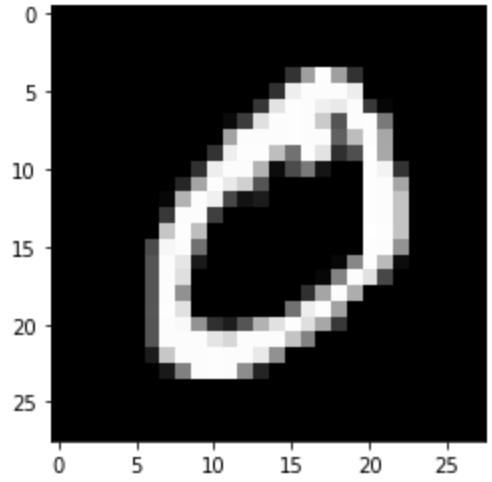
```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3)
Epoch 1/3
1875/1875 [==============================] - 103s 53ms/step - loss: 0.8556
- accuracy: 0.7747 - val_loss: 0.1181 - val_accuracy: 0.9639
Epoch 2/3
1875/1875 [==============================] - 88s 47ms/step - loss: 0.2763
- accuracy: 0.9168 - val_loss: 0.0739 - val_accuracy: 0.9771
Epoch 3/3
1875/1875 [==============================] - 88s 47ms/step - loss: 0.2119
- accuracy: 0.9375 - val_loss: 0.0653 - val_accuracy: 0.9819
```

Out[10]:

In [11]:

```
example = X_train[1]
prediction = model.predict(example.reshape(1, 28, 28, 1))
print ("Prediction (Softmax) from the neural network:\n\n
{}".format(prediction))
hard_maxed_prediction = np.zeros(prediction.shape)
hard_maxed_prediction[0][np.argmax(prediction)] = 1
print ("\n\nHard-maxed form of the prediction: \n\n
{}".format(hard_maxed_prediction))

print ("\n\n--------- Prediction --------- \n\n")
plt.imshow(example.reshape(28, 28), cmap="gray")
plt.show()
print("\n\nFinal Output: {}".format(np.argmax(prediction)))
1/1 [==============================] - 0s 102ms/step
Prediction (Softmax) from the neural network:

 [[1.0000000e+00 3.5452457e-12 4.2870735e-08 2.3118686e-12 1.1107838e-12
  1.7552315e-12 7.4592929e-11 3.0929332e-11 4.9751794e-08 3.9869605e-09]]


Hard-maxed form of the prediction:

 [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

```
--------- Prediction ---------
```




```
Final Output: 0
```

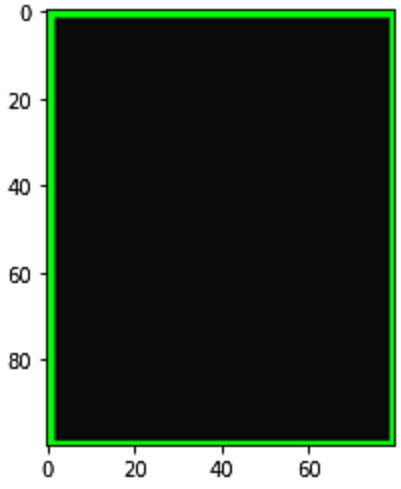
In [12]:

```
metrices=model.evaluate(X_test,y_test,verbose=0)
print("Metrices(test loss and Test Accuracy):")
print(metrices)
Metrices(test loss and Test Accuracy):
[0.06527048349380493, 0.9818999767303467]
```

In [13]:

```
image = cv2.imread('test_image.jpg')
image = np.full((100,80,3), 12, dtype = np.uint8)
grey = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(grey.copy(), 75, 255, cv2.THRESH_BINARY_INV)
contours,hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
preprocessed_digits = []

for c in contours:
    x,y,w,h = cv2.boundingRect(c)

    # Creating a rectangle around the digit in the original image (for
displaying the digits fetched via contours)
    cv2.rectangle(image, (x,y), (x+w, y+h), color=(0, 255, 0),
thickness=2)

    # Cropping out the digit from the image corresponding to the current
contours in the for loop
```

```
    digit = thresh[y:y+h, x:x+w]

    # Resizing that digit to (18, 18)
    resized_digit = cv2.resize(digit, (18,18))

    # Padding the digit with 5 pixels of black color (zeros) on each side
to finally produce the image of (28, 28)
    padded_digit = np.pad(resized_digit, ((5,5),(5,5)), "constant",
constant_values=0)

    # Adding the preprocessed digit to the list of preprocessed digits
    preprocessed_digits.append(padded_digit)

print("\n\n\n----------------Contoured Image--------------------")
import os, types
import pandas as pd

def __iter__(self): return 0

print=("\n\n\n----------------Contoured Image--------------------")
plt.imshow(image, cmap="gray")
plt.show()

inp = np.array(preprocessed_digits)
----------------Contoured Image--------------------
```



➢ ***ADVANTAGES:***

This approach has many advantages:

- The system not only produces a classification of the digit but also a rich description of the installation parameters which can yield information such as the writing style.
- The generative models can perform recognition driven segmentation.
- The method involves a relatively small number of parameters and hence training is relatively easy and fast.
- Unlike many other recognition schemes, it does not rely on some form of pre-normalization of input images, but can handle arbitrary scaling, translations and a limited degree of image rotation.

➢ ***DISADVANTAGES:***

The disadvantage is that,

- 1It is not done in real time as a person writes and therefore not appropriate for immediate text input.

➢ ***Result:***

- The core idea behind applying data augmentation is to avert the overfitting problemby artificially increasing the size of our datasets without collecting new data. We cando that by making tiny transformations to the existing training data, while keeping the labels without change. These small transformations act as a regularizer, which assists usin reducing overfitting. In our experiments, we randomly rotate the images in a rangeof 10 degrees. Zoom range, random vertical, and horizontal shifts of the data traininghave been set to 0.1. The highest accuracy that was achieved with data augmentation was99.98%, and only 99.50% without data augmentation.Thus it will work properly.

➢ ***Conclusion:***

- We presented a novel convolutional neural network architecture based on data preparation, receptive field, data augmentation, optimization, normalization, and regularization techniques for handwritten digit recognition. To guarantee the dataset does not contain any unnecessary details and that it is fit for applying in our CNN model, data preparation is conducted as an essential first step in our proposed model. Without applying data preparation to the raw data, it is highly possible that unnecessary data leads to misleading results. In our work, filter sizes are determined by calculating the size of the ERF. Calculating this size can help in enhancing the performance of our CNN. In ERF, the process is started with a proposed filter to convolve the input image and gain its feature map. Then, this process is repeated with the next layers to gain deeper feature maps until getting the output image with an effective receptive field with a size of 22 × 22. Maxpooling with a 2 × 2 filter and stride = 2 is used to increase the size of the receptive field. Proposed CNN architecture has achieved recognition accuracy of 99.98% on the MNIST handwritten digit dataset, and 99.40% with the same dataset contaminated with 50% noise. Our experimental results show that using data augmentation with CNN gives better recognition accuracy compared to CNN without data augmentation. Utilizing the data augmentation technique has helped to expand our training dataset, resulting in improving the performance and classification capability of our model. We also presented the RMSprop optimizer to restrict the oscillations in the vertical direction and take larger steps in the horizontal direction in order to converge substantially faster to the global minimum point. On the very competitive MNIST handwritten digits benchmark, our proposed CNN model has achieved superiority over state-of-the-art methods for handwritten digits recognition. In our experiments, batch normalization has been used to improve the training performance and enhance the stability of our model. With the usage of batch normalization, we can speed up

the training, reduce training and testing time, in addition to lowering the sensitivity initialization. In order to avoid overfitting and underfitting, an early stopping technique has used to determine the optimal number of training epochs.

➤ ***Future Scope:***

- We believe that our proposed model can further be applied to other datasets. In contrast, as a future work, we find that it is worth taking further actions to improve our model performance in terms of how to perfectly learn and extract the local features in the hidden layers, and how to enhance the recognition ability in the fully connected layers to avoid mislabeling problems.
- 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.
- Handwritten Digit Recognition has various real-life time uses. It is used in the detection of vehicle number, banks for reading cheques, post offices for arranging letter, and many other tasks.
- The handwriting recognition is highly applicable in writer identification, where it is used inforensics and biometrics. So one of the applications of handwriting recognition is solving theancient manuscript disputes, where the actual writer of the manuscript is identified based oncertain features of writers' handwriting. In this way, it assists to avoid false claims of handwriting or manuscript.

➤ ***Appendix:***

➤ ***Project Demo Link:***

https://drive.google.com/drive/folders/1HSdv1WyMraKM0QolcGKhJN9cnQDJRoEs

➤ ***Github:***

IBM-EPBL/IBM-Project-46145-1660739661: A Novel Method for Handwritten Digit Recognition System (github.com)