A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

PROJECT REPORT

SUBMITTED BY

NILAVANI K (312819205024)

YAMINI D (312819205049)

VASUNTHRA R (312819205047)

SNEHASRI (312819205039)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

AGNI COLLEGE OF TECHNOLOGY, THALAMBUR

ANNA UNIVERSITY: CHENNAI 600025

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project **A Novel Method for Handwritten Digit Recognition System** is the bonafide work **of Nilavani.K (312819205024) Yamini.D (312819205049) Vasunthra.R (312819205047) and Snehasri.S V (312819205039)** who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Dr. S. GEERTHIK M.E, Ph.D.	**Mrs. SUGANTHI P M.E**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Information Technology	Department of Information Technology
Agni College of Technology	Agni College of Technology
Thalambur, Chennai-600130	Thalambur, Chennai-600130

Submitted for the project work and viva voice held on ....

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ABSTRACT

The handwritten digit recognition is the capability of computer applications to recognize 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 system is a way to tackle this problem which uses the image of a digit and recognizes the digit present in the image. Convolutional Neural Network model created using the MNIST dataset to recognize handwritten digits.

Handwritten Digit Recognition is the capability of a computer to fete the mortal handwritten digits or characters from different sources and classify them into 10 predefined classes (0-9). This has been a content of bottomless exploration in the field of deep literacy. Number recognition has numerous operations like number plate recognition, postal correspondence sorting, bank check processing, etc. In Handwritten number recognition, we face numerous challenges because of different styles of jotting of different peoples as it is not an Optic character recognition.

This exploration provides a comprehensive comparison between different machine literacy and deep literacy algorithms for the purpose of handwritten number recognition. For this, we have used Convolutional Neural Network. The Neural network is used to train and identify written digits. 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 to UI.

ACKNOWLEDGEMENT

I would like to express our deepest gratitude to the management of **"AGNI COLLEGE OF TECHNOLOGY"** and would like to thank our respected Principal **Dr. SRINIVASAN ALAVANDAR** for his words of inspiration and for providing necessary facilities to carry out our project work successfully.

I immensely thankful to **Dr. GEERTHIK M.E, PhD**, Head of the Department, Information Technology for his words of wisdom and his constant source of inspiration.

I would like to offer our thanks to our guide **Mrs. P.SUGANTHI M.E** Professor, Department of Information Technology who molded us accordingly and gave valuable suggestions for completing our project work successfully.

We extended our warmest thanks to all the faculty members of our department for their assistance and we also thank all our friends who helped us in bringing out our project in good shape and form.

Finally, we express our sincere benevolence to our beloved parents for their perpetual encouragement and support in all endeavors.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Brief Overview of Work

In recent times, with the increase of Artificial Neural Network (ANN), deep learning has brought a dramatic twist in the field of machine learning by making it more artificially intelligent. Deep learning is remarkably used in vast ranges of fields because of its diverse range of applications such as surveillance, health, medicine, sports, robotics, drones, etc. In deep learning, Convolutional Neural Network (CNN) is at the center of spectacular advances that mixes Artificial Neural Network (ANN) and up to date deep learning strategies. It has been used broadly in pattern recognition, sentence classification, speech recognition, face recognition, text categorization, document analysis, scene, and handwritten digit recognition. The goal of this paper is to observe the variation of accuracies of CNN to classify handwritten digits using various numbers of hidden layers and epochs and to make the comparison between the accuracies. For this performance evaluation of CNN, we performed our experiment using Modified National Institute of Standards and Technology (MNIST) dataset. Further, the network is trained using stochastic gradient descent and the backpropagation algorithm.

1.2 Objective

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 Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

1.3 Scope of the Project

Recently Deep Convolutional Neural Networks (CNNs) becomes one of the most appealing approaches and has been a crucial factor in variety of recent success and challenging machine learning applications such as challenge ImageNet, object detection, image segmentation, and face recognition. Therefore, CNNs is considered our main model for our challenging tasks of image classification. Specifically, it is used for handwriting digits recognition which is one of high academic and business transactions. Handwriting digit recognition application is used in different tasks of our real life time purposes. Precisely, it is used in banks for reading checks, post offices for sorting letter, and many other related tasks.

1.4 Project Requirements

Hardware Requirements

- Hard Disk: 250GB(further increase that as per requirement)
- RAM: 2GB

Software Requirements

Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

Tensor flow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Flask

Web frame work used for building Web applications

CHAPTER 2

SYSTEM ANALYSIS

2.1 Literature Survey

Paper 1: Novel Deep Neural Network Model for Handwritten Digit Classification and Recognition

Year: 2021

Authors: Ayush Kumar Agrawal and Vineet Kumar Awasthi

An artificial neural network has one hidden layer between the input and output layers, whereas a deep neural network has numerous hidden layers with input and output layers. Deep neural networks use several hidden layers to increase model performance and achieve higher accuracy compared to accuracy of machine learning models. Most researchers do their research in the area of pattern recognition. In the field of pattern recognition, there are many patterns that can be used, including handwritten numbers, characters, pictures, faces, sounds, and speech.

This study focuses on the classification and recognition of handwritten digits.1000 were utilized as test samples and 1000 were training samples.10000 picture samples make up the USPS dataset, of which 7291 serve as training samples and 2007 serve as testing samples. We've used the proposed deep neural network technique in this paper to classify and identify data from the ARDIS and USPS datasets. The suggested model consists of six layers with softmax and relu activation functions. After model implementation, accuracy for ARDIS samples reached 98.70% testing and 99.76% training, which is greater than accuracy from prior research. Additionally, using the USPS samples dataset, 98.22% training accuracy and 93.01% testing accuracy were attained. When compared to earlier methodologies, the data show that deep neural networks perform incredibly well.

Paper 2: A Novel Handwritten Digit Classification System Based on Convolutional Neural Network Approach

Year: 2021

Authors: Ali Abdullah Yahya, Jieqing Tan, Min Hu

There have been a tons of CNN classification algorithms put forth in the literature. However, these algorithms do not take into account the proper filter size selection, data preparation, dataset restrictions, or noise. As a result, few algorithms have been able to significantly increase classification accuracy. The paper makes the following contributions to solve these methods' drawbacks: First, the size of the effective receptive field (ERF) is determined after taking domain knowledge into account. They choose a typical filter size with the aid of the ERF calculation, improving the classification accuracy of our CNN. Second, excessive data produces inaccurate results, which has a detrimental impact on classification accuracy.

Before carrying out the data classification task, data preparation is conducted to ensure that the dataset is devoid of any redundant or irrelevant variables to the goal variable. Thirdly, data augmentation has been suggested as a way to reduce training and validation errors and prevent dataset limitations. Fourthly, the paper suggests adding an additive white Gaussian noise with a threshold of 0.5 to the MNIST dataset in order to imitate the natural factors that can affect image quality in the real world. With a recognition accuracy of 99.98% and 99.40% with 50% noise, our CNN algorithm achieves state-of-the-art performance in handwritten digit recognition.

Paper 3: Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)

Year : 2020

Authors: Savita Ahlawat , Amit Choudhary , Anand Nayyar , Saurabh Singh and Byungun Yoon

Customized features and a vast quantity of past knowledge have been used in traditional handwriting recognition systems. It is difficult to train an optical character recognition (OCR) system based on these conditions. Deep learning approaches have enabled significant performance in the field of handwriting recognition research in recent years. Nonetheless, the increasing increase in the amount of handwritten data, along with the availability of vast computing capacity, necessitates improvements in recognition accuracy and warrants additional exploration. Convolutional neural networks (CNNs) are extremely excellent in perceiving the structure of handwritten characters/words in ways that aid in the automatic extraction of distinguishing features, making CNN the best solution for solving handwriting recognition challenges. The proposed work aims to investigate several design alternatives for CNN-based handwritten digit recognition, such as the number of layers, stride size, receptive field, kernel size, padding, and dilution. Furthermore, we intend to assess the effectiveness of several SGD optimization techniques in enhancing the performance of handwritten digit recognition. Using ensemble architecture improves the recognition accuracy of a network. In this case, we want to obtain equal accuracy by employing a pure CNN design without ensemble architecture, because ensemble structures increase computational overhead and testing complexity. As a result, a CNN design is developed in order to obtain higher accuracy than ensemble systems while reducing operational complexity and expense.

Furthermore, we demonstrate an appropriate combination of learning parameters in the design of a CNN that leads us to a new absolute record in categorising MNIST handwritten digits. We conducted extensive trials and achieved 99.87% recognition accuracy for an MNIST dataset.

Paper 4: Handwritten Character Recognition using Neural Network and TensorFlow

Year : 2019

Authors : Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta

The offline handwritten character recognition in this study will be carried out using Tensorflow and a convolutional neural network. a process known as using SoftMax Regression, one may assign probabilities to one of the many characters in the handwritten text that offers the range of values from 0 to 1, summed to 1.

The objective is to create software that is extremely accurate and that has a minimum level of spatial and temporal complexity. It was determined that strategies for feature extraction like diagonal and direction are significantly better at producing high accuracy. Outcomes in comparison to other conventional vertical and horizontal techniques moreover use the best Neural network tried layers provides the benefit of a higher accurate outcome by having a high noise tolerance.

The feed forward model in neural networks is the back-propagation algorithm that was primarily used to classify the characters, recognise them, and receive training continually more. In addition to these, normalizing along with feature extraction, the results were better and more effective.

Character recognition is the outcome of accuracy. The paper will describe the best approach to get more than 90% accuracy in the field of Handwritten Character Recognition (HCR).

S.NO	NAME OF THE PAPER	YEAR OF PUBLICATION	AUTHORS
1.	**Novel Deep Neural Network Model for Handwritten Digit Classification and Recognition**	2021	Ayush Kumar Agrawal and Vineet Kumar Awasthi
2.	**A Novel Handwritten Digit Classification System Based on Convolutional Neural Network Approach**	2021	Ali Abdullah Yahya, Jieqing Tan, Min Hu
3.	**Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)**	2020	Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon
4.	**Handwritten Character Recognition using Neural Network and TensorFlow**	2019	Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta

2.2 Project Feasibility Study

2.2.1 Technical Feasibility

Technical feasibility study is concerned with specifying equipment and software that will successfully satisfy the user requirement, the technical needs of the system may vary considerably. The facility to produce outputs in a given time.

2.2.2 Economical Feasibility

Economical feasibility is the measure to determine the cost and benefit of the proposed system. A project is economical feasible which is under the estimated cost for its development. These benefits and costs may be tangible or intangible. Handwritten digit recognition system is the cost-effective project in which there is less possibility of intangible cost so there is no difficulty to determine the cost of the project.

2.2.3 Operational Feasibility

Operation feasibility is used to check whether the project is operationally feasible or not. Our project is mainly different from the other system because of its web-support feature. So the measure for operational feasibility is something different from other system. Generally the operational feasibility is related to organization aspects.

2.3 Modules Description

Pytorch

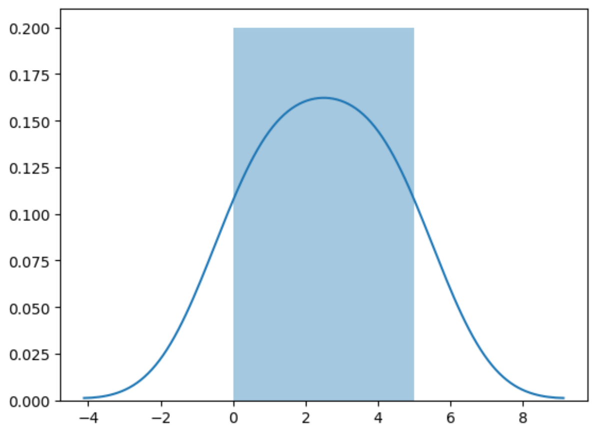
PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license.

Torchvision

Torchvision is **a library for Computer Vision** that goes hand in hand with PyTorch. It has utilities for efficient Image and Video transformations, some commonly used pre-trained models, and some datasets (torchvision does not come bundled with PyTorch , you will have to install it separately.).

Seaborn

Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.



Matplotlib

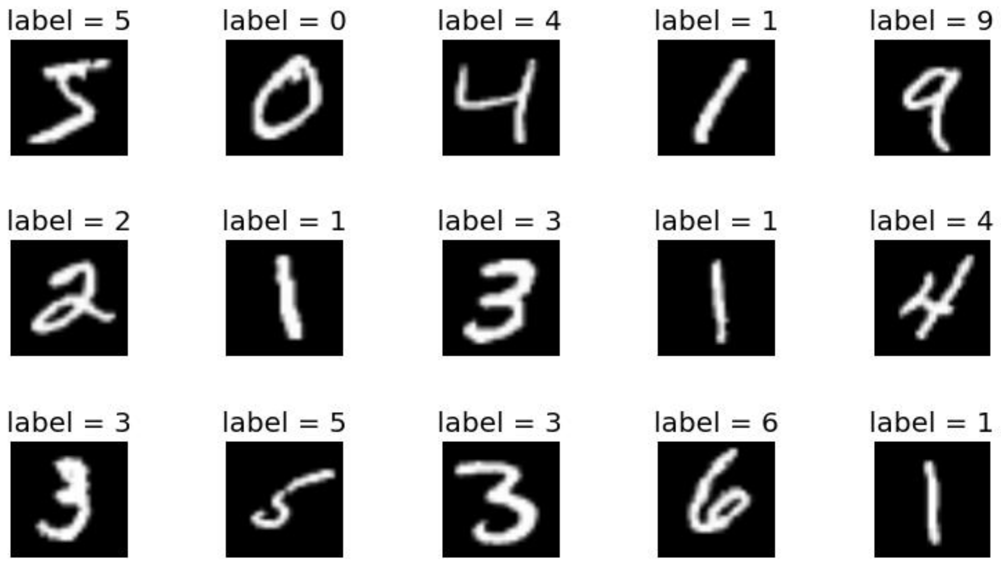
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Tensorflow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

MNIST dataset

Modified National Institute of Standards and Technology (MNIST) is a large set of computer vision dataset which is extensively used for training and testing different systems. It was created from the two special datasets of National Institute of Standards and Technology (NIST) which holds binary images of handwritten digits. The training set contains handwritten digits from 250 people, among them 50% training dataset was employees from the Census Bureau and the rest of it was from high school students [26]. However, it is often attributed as the first datasets among other datasets to prove the effectiveness of the neural networks**.**

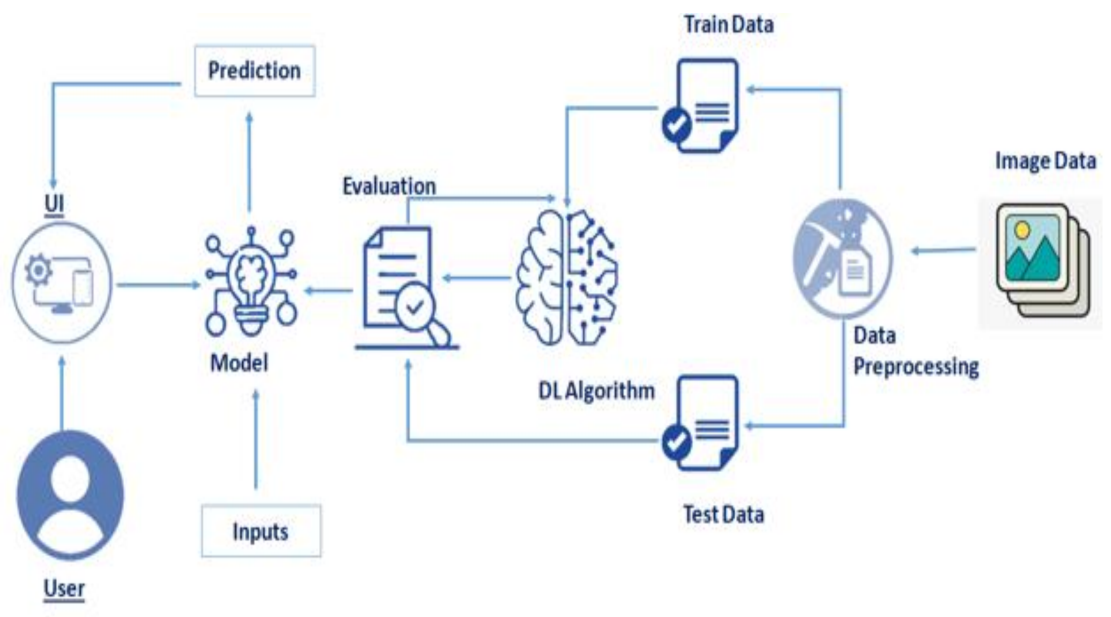


The database contains 60,000 images used for training as well as few of them can be used for cross-validation purposes and 10,000 images used for testing [27]. All the digits are grayscale and positioned in a fixed size where the intensity lies at the center of the image with 28×28 pixels. Since all the images are 28×28 pixels, it forms an array which can be flattened into 28*28=784 dimensional vector. Each component of the vector is a binary value which describes the intensity of the pixel.

CHAPTER 3

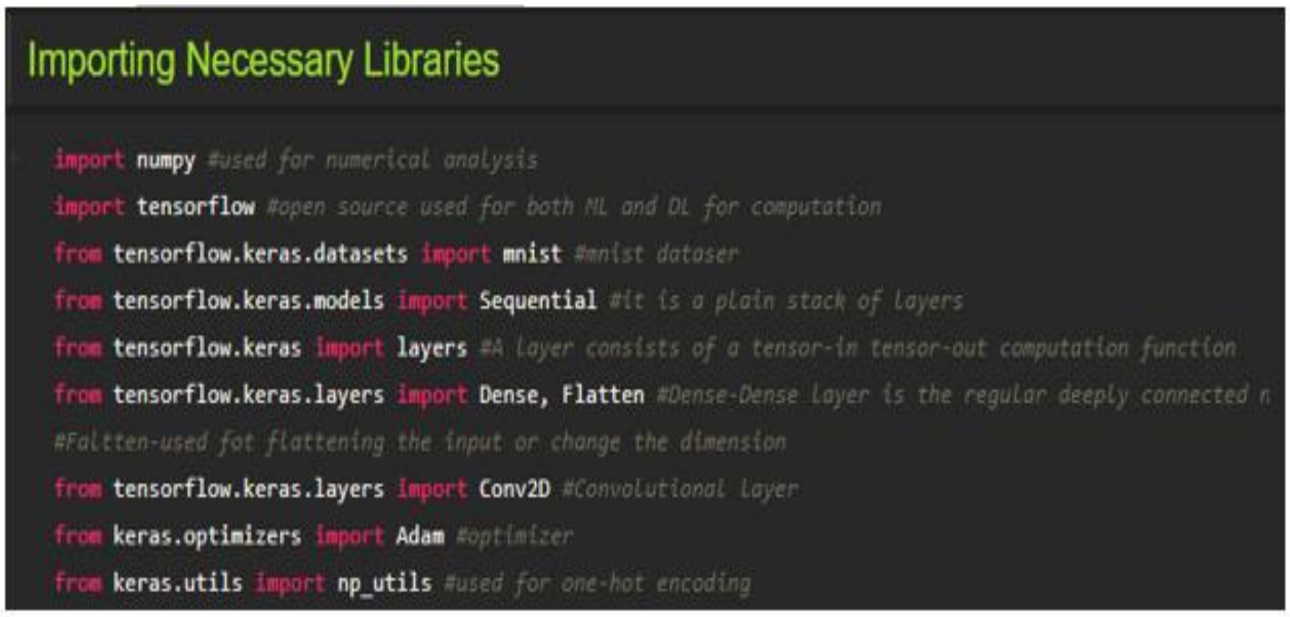
SYSTEM DESIGN

3.1 Architecture Design



3.2 Understanding the Data

Importing the required libraries

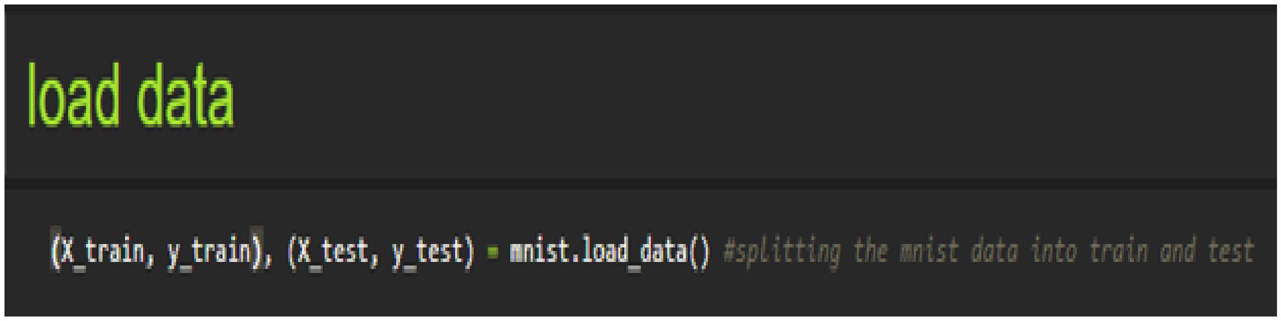


Importing Necessary Libraries

```
import numpy #used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataser
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deeply connected n
#Faltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #Convolutional Layer
from keras.optimizers import Adam #optimizer
from keras.utils import np_utils #used for one-hot encoding
```

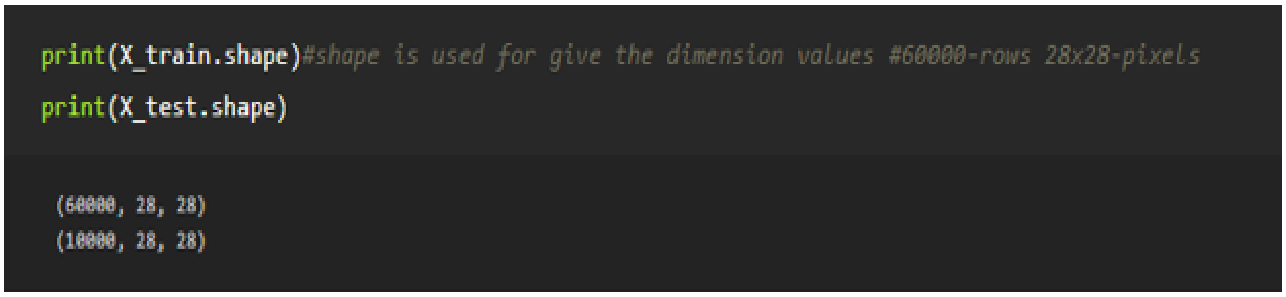
Loading the data

The dataset for this model is imported from the Keras module.



load data

```
(X_train, y_train), (X_test, y_test) = mnist.load_data() #splitting the mnist data into train and test
```



```
print(X_train.shape)#shape is used for give the dimension values #60000-rows 28x28-pixels
print(X_test.shape)
```

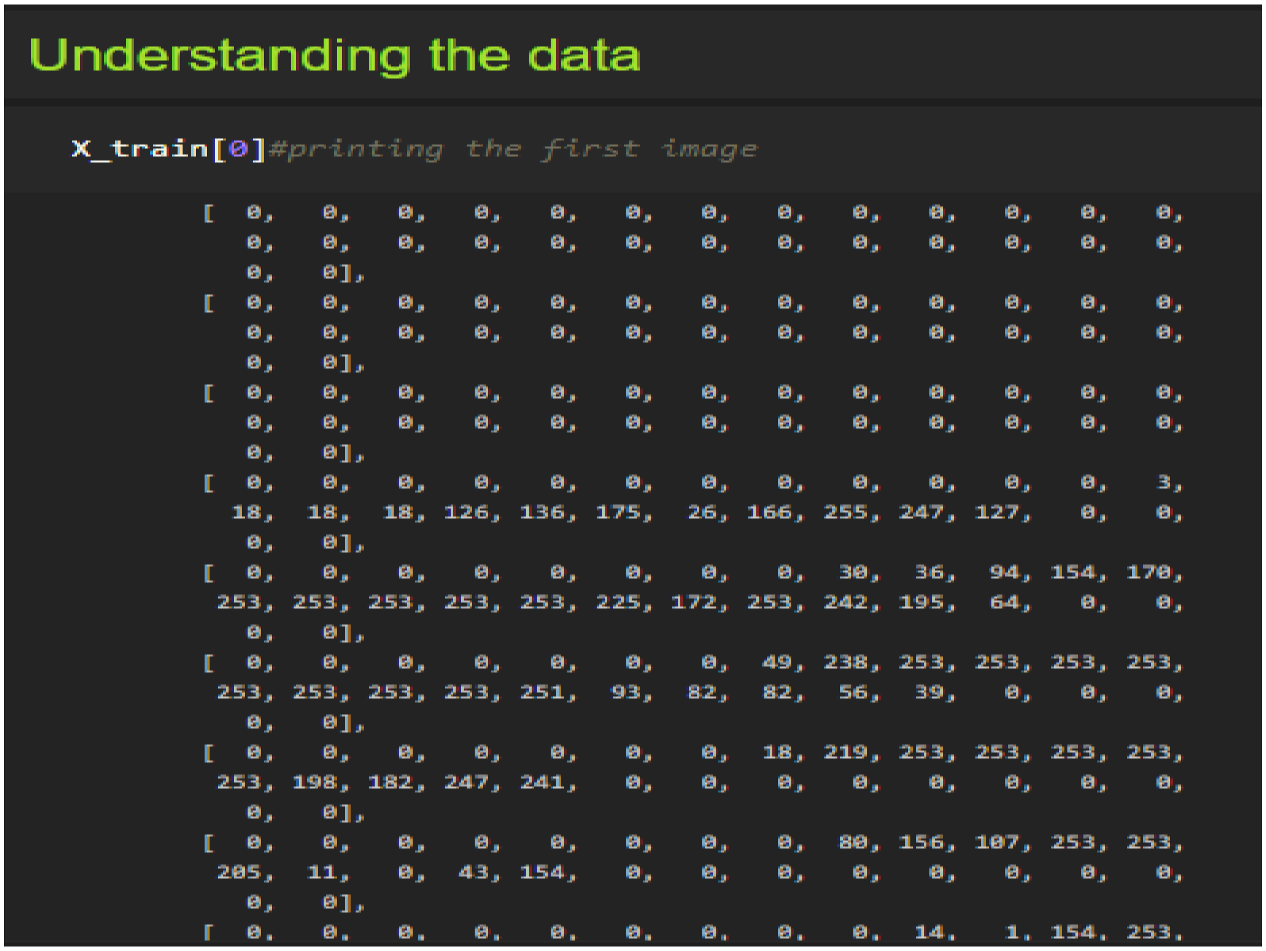
```
(60000, 28, 28)
(10000, 28, 28)
```

We are finding out the shape of X_train and x_test for better understanding. It lists out the dimensions of the data present in it.

In trainset, we have 60000 images, and in the test set we have 10000 images.

Analyzing the Data

Let's see the Information of an image lying inside the x_train variable



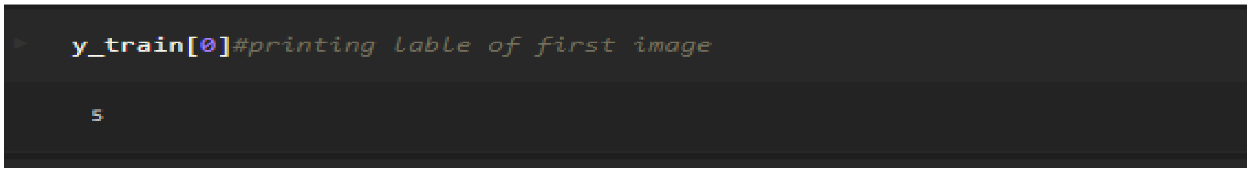
Understanding the data

```
X_train[0]#printing the first image
```

```
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
          18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
         253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
         253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
         253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
         205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
```

Basically, the pixel values range from 0-255. Here we are printing the first image pixel value which is index[0] of the training data. As you see it is displayed in the output.

With respect to this image, the label of this image will e stored in y_train let's see what is the label of this image by grabbing it from the y_train variable

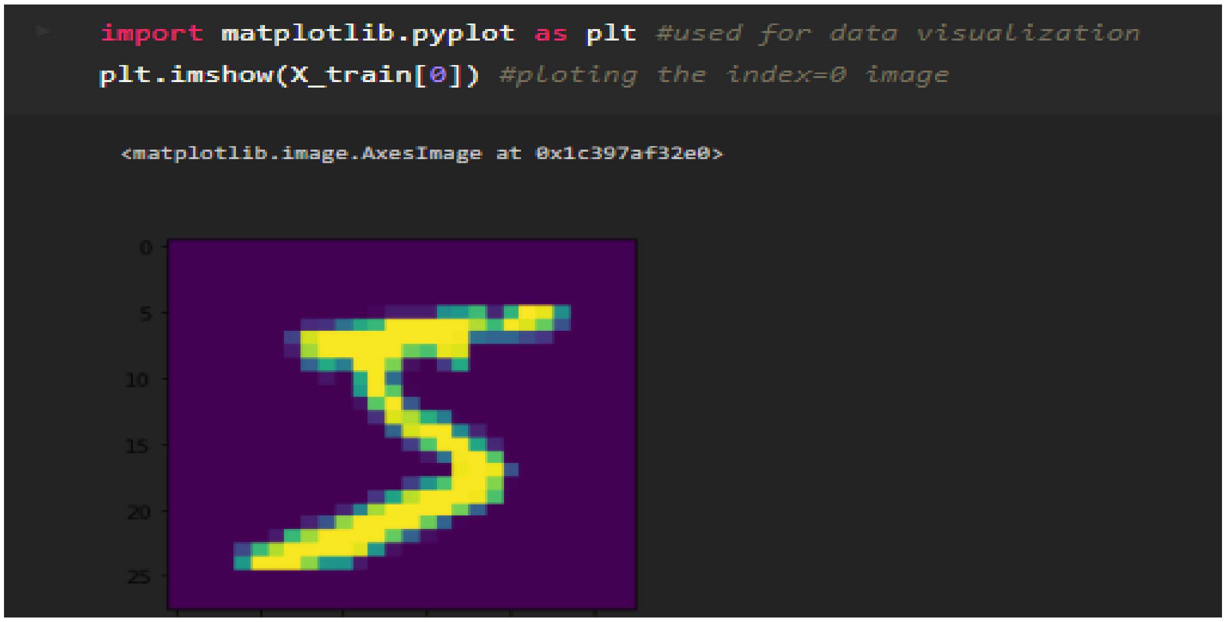


```
y_train[0]#printing lable of first image
```

```
5
```

As we saw in the previous screenshot, we get to know that the pixel values are printed. Now here we are finding to which image the pixel values belong to. From the output displayed we get to know that the image is ‘5’.

Lets Plot the image on a graph using the Matplot library

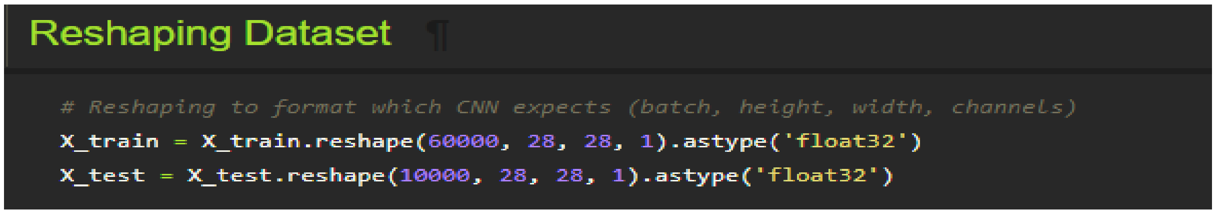


```
import matplotlib.pyplot as plt #used for data visualization
plt.imshow(X_train[0]) #ploting the index=0 image

 <matplotlib.image.AxesImage at 0x1c397af32e0>
```

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. By using the Matplotlib library we are displaying the number '5' in the form of an image for proper understanding.

Reshaping the Data

As we are using Deep learning neural network, the input for this network to get trained on should be of higher dimensional. Our dataset is having three-dimensional images so we have to reshape them too higher dimensions.



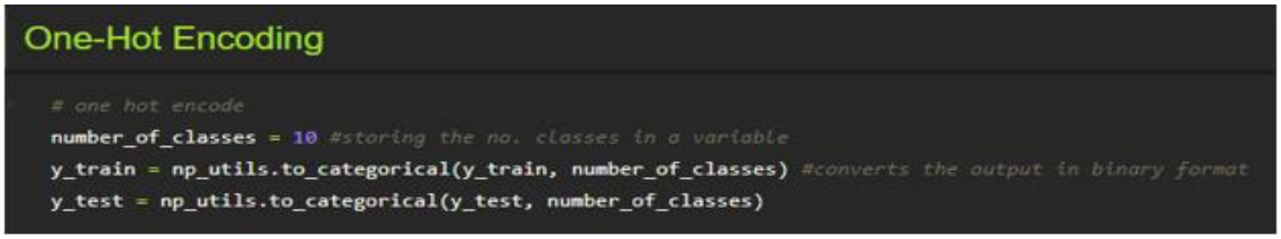
Reshaping Dataset ¶

```
# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

We are reshaping the dataset because we are building the model using CNN. As CNN needs four attributes batch, height, width, and channels we reshape the data.

Applying one hot Encoding

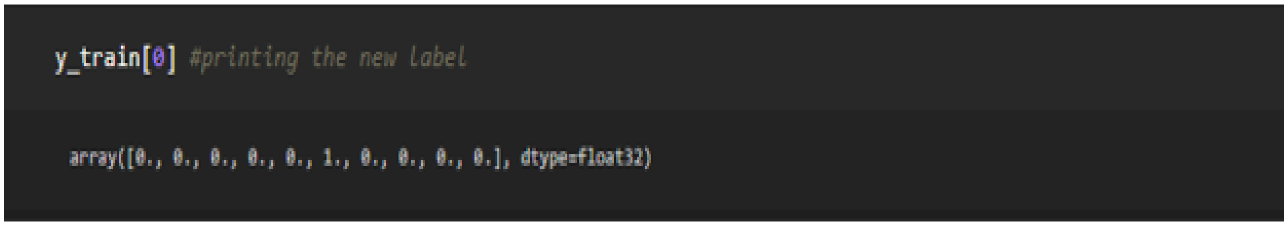
If you see our y_train variable contains Labels representing the images containing in x_train. AS these are numbers usually they can be considered as numerical or continuous data, but with respect to this project these Numbers are representing a set of class so these are to be represented as categorical data, and we need to binaries these categorical data that's why we are applying One Hot encoding for y_train set


One-Hot Encoding

```
# one hot encode
number_of_classes = 10 #storing the no. classes in a variable
y_train = np_utils.to_categorical(y_train, number_of_classes) #converts the output in binary format
y_test = np_utils.to_categorical(y_test, number_of_classes)
```

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. We apply One-Hot Encoding in order to convert the values into 0's and 1's. For a detailed point of view, look at this link

Now let's see how our label 5 is index 0 of y_train is converted



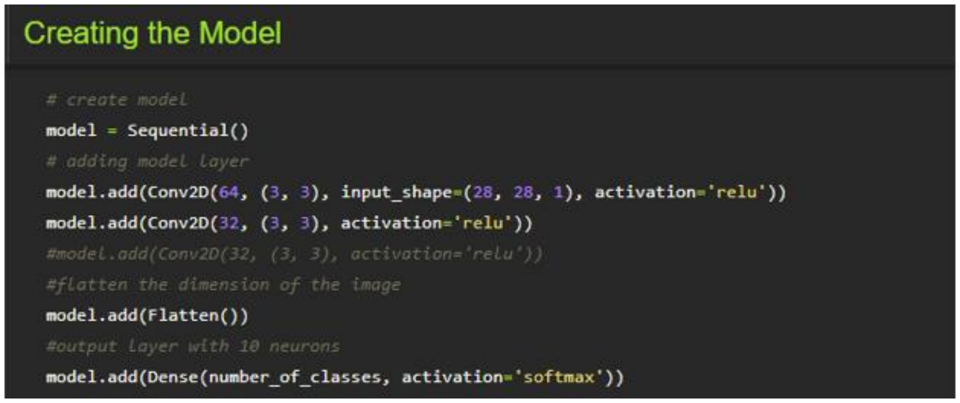
```
y_train[0] #printing the new label

array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

As we see the new the label is printed in the form of 0's and 1's and is of type float.

3.3 Model Building

Adding CNN layers

Creating the model and adding the input, hidden, and output layers to it.

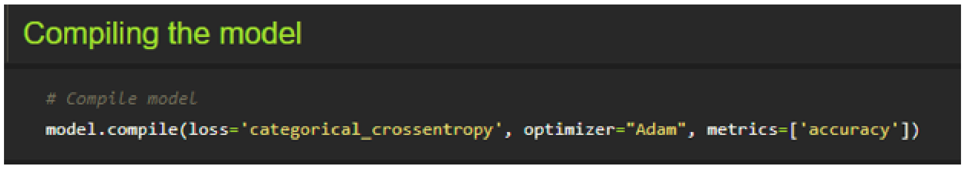

Creating the Model

```
# create model
model = Sequential()
# adding model layer
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu'))
#model.add(Conv2D(32, (3, 3), activation='relu'))
#flatten the dimension of the image
model.add(Flatten())
#output layer with 10 neurons
model.add(Dense(number_of_classes, activation='softmax'))
```

The Sequential model is a linear stack of layers.

Compiling the Model

With both the training data defined and model defined, it's time to configure the learning process. This is accomplished with a call to the compile () method of the Sequential model class. Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics.


Compiling the model

```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
```

3.4 Design Phase

3.4.1 Proposed Solution

Problem Statement: Handwritten Digit Recognition

MNIST (“Modified National Institute of Standards and Technology”) is considered an unofficial computer vision “hello-world” dataset. This is a collection of thousands of handwritten pictures used to train classification models using Machine Learning techniques.

As a part of this problem statement, we will train a multi-layer perceptron using TensorFlow to recognize the handwritten digits.

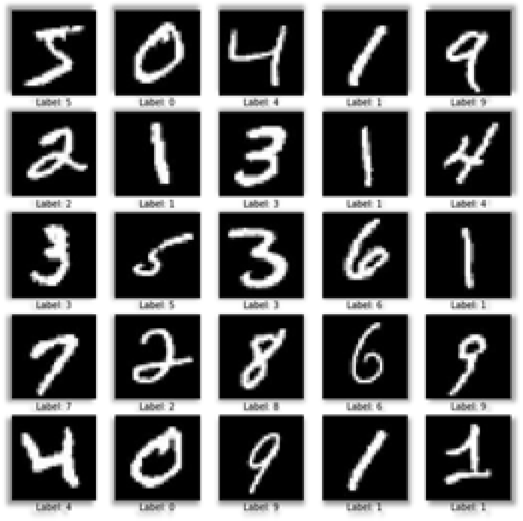
Solution

MNIST Dataset Description

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 digitalized 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 analysed by the model and the detected result is returned on to UI

The MNIST Handwritten Digit Recognition Dataset contains 60,000 training and 10,000 testing labelled handwritten digit pictures.

Each picture is 28 pixels in height and 28 pixels wide, for a total of 784 (28×28) pixels. Each pixel has a single pixel value associated with it. It indicates how bright or dark that pixel is (larger numbers indicates darker pixel). This pixel value is an integer ranging from 0 to 255.



Procedure

- Install the latest TensorFlow library.
- Prepare the dataset for the model.
- Develop Single Layer Perceptron model for classifying the handwritten digits.
- Plot the change in accuracy per epochs.

- Evaluate the model on the testing data.
- Analyse the model summary.
- Add hidden layer to the model to make it Multi-Layer Perceptron.
- Add Dropout to prevent overfitting and check its effect on accuracy.
- Increasing the number of Hidden Layer neuron and check its effect on accuracy.
- Use different optimizers and check its effect on accuracy.
- Increase the hidden layers and check its effect on accuracy.
- Manipulate the batch size and epochs and check its effect on accuracy.

MNIST is a dataset which is widely used for handwritten digit recognition. The dataset consists of 60,000 training images and 10,000 test images. The artificial neural networks can all most mimic the human brain and are a key ingredient in image processing field.

Handwritten digit recognition using MNIST dataset is a major project made with the help of Neural Network. It basically detects the scanned images of handwritten digits.

We have taken this a step further where our handwritten digit recognition system not only detects scanned images of handwritten digits but also allows writing digits on the screen with the help of

an integrated GUI for recognition.

Approach

We will approach this project by using a three-layered Neural Network.

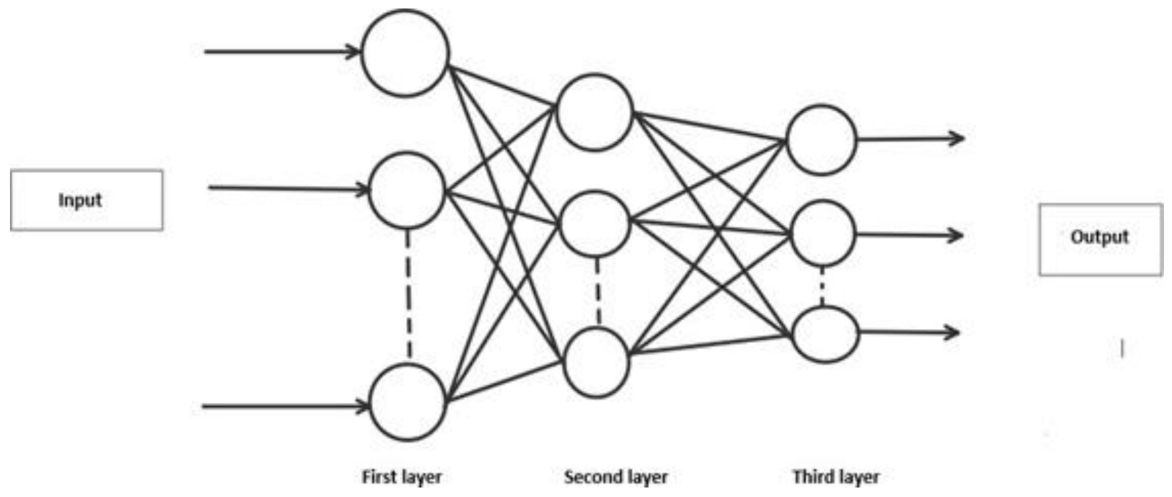
- **The input layer:** It distributes the features of our examples to the next layer for calculation of activations of the next layer.

- **The hidden layer:** They are made of hidden units called activations providing nonlinear ties for the network. A number of hidden layers can vary according to our requirements.
- **The output layer:** The nodes here are called output units. It provides us with the final prediction of the Neural Network on the basis of which final predictions can be made.

A neural network is a model inspired by how the brain works. It consists of multiple layers having many activations, this activation resembles neurons of our brain. A neural network tries to learn a set of parameters in a set of data which could help to recognize the underlying relationships. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

Methodology

We have implemented a Neural Network with 1 hidden layer having 100 activation units (excluding bias units). The data is loaded from a .mat file, features(X) and labels(y) were extracted. Then features are divided by 255 to rescale them into a range of [0,1] to avoid overflow during computation. Data is split up into 60,000 training and 10,000 testing examples. Feedforward is performed with the training set for calculating the hypothesis and then backpropagation is done in order to reduce the error between the layers. The regularization parameter lambda is set to 0.1 to address the problem of overfitting. Optimizer is run for 70 iterations to find the best fit model.



Algorithm

Forward Propagation Architecture

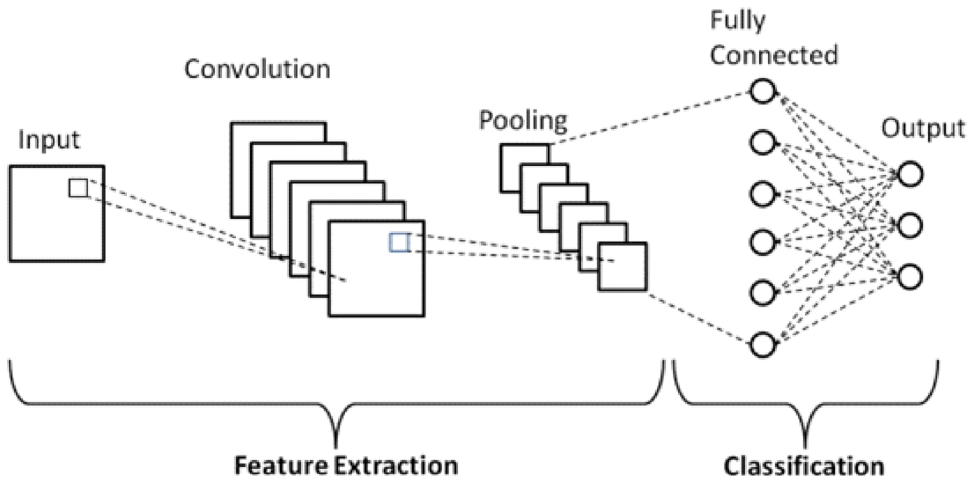
It is a small workflow of how CNN module will extract the features and classify the image based on it. The architecture shows the input layer, hidden layers and output layer of the network. There are many layers involved in the feature extraction phase of the network which involves convolution and subsampling .

Explanation of the Proposed System

- The first layer of the architecture is the User layer. User layer will comprise of the people who interacts with the app and for the required results.
- The next three layers is the frontend architecture of the application.

The application will be developed using which is the open-source platform for HTML, CSS and JavaScript.

The application is deployed in the localhost which is shown on the browser. Through the app, the user will be able to upload pictures of the handwritten digits and convert it into the digitalized form. • The one in between the database and view layer is the business layer which is the logical calculations on the basis of the request from the client side. It also has the service interface. • The backend layer consists of two datasets: Training Data and Test Data. The MNIST database has been used for that which is already divided into training set of 60,000 examples and test of 10,000 examples. • The training algorithm used is Convolution Neural Network. This will prepare the trained model which will be used to classify the digits present in the test data. Thus, we can classify the digits present in the images as: Class 0,1,2,3,4,5,6,7,8,9.



Working

- Neural Networks receive an input and transform it through a series of hidden layers.
- Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer.
- Neurons in a single layer function completely independently. • The last fully connected layer is called the "output layer“.

Convolution Layer

The Convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume.

During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter.

As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input..

Feature Extraction

All neurons in a feature share the same weights .In this way all neurons detect the same feature at different positions in the input image. Reduce the number of free parameters.

Subsampling Layer

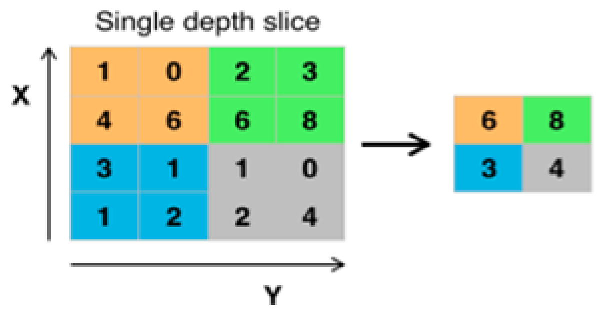
Subsampling, or down sampling, refers to reducing the overall size of a signal .The subsampling layers reduce the spatial resolution of each feature map. Reduce the effect of noises and shift or distortion invariance is achieved.

Pooling layer

It is common to periodically insert a Pooling layer in-between successive Conv layer in a Convent architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

TensorFlow

TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. See the sections below to get started. By scanning the numerical digit and convert into png format using python3 command in terminal we can get text output and sound output.



Pooling layer

RESULT

As with any work or project taken up in the field of machine learning and image processing, we are not considering our results to be perfect.

Machine learning is a constantly evolving field and there is always room for improvement in your methodology; there is always going to be another new approach that gives better results for the same problem. The application has been tested using three models: Multi-Layer Perceptron (MLP), Convolution Neural Network (CNN).

CHAPTER –4

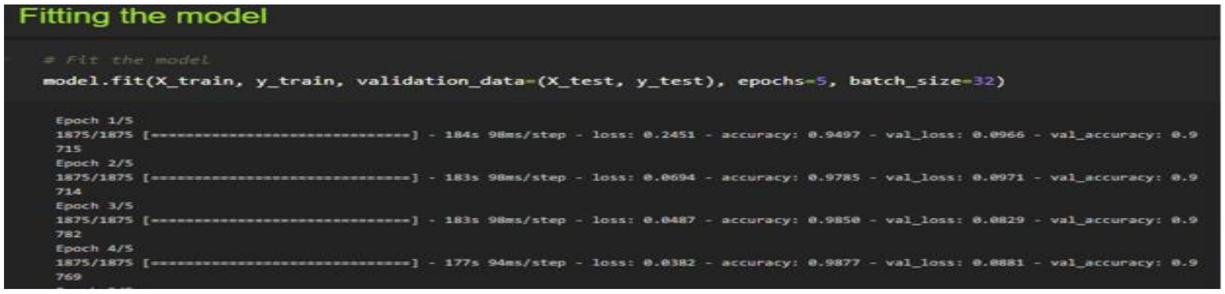
PLANNING, TRAINING AND TESTING

4.1 Planning

Sprint Delivery

Sprint	Functional Requirement	User Story/Task
Sprint-1	Pre-Processing & Image data	As a User, you must acquire Image Data of Hand Written Images in order to train the model.
Sprint-2	Dash board or Website	We created a dynamic Webpage to host our model using the Python Flask Framework (UI).
Sprint-3	Feature Extraction Model	Image Classification Using the CNN Model features can be identified.
Sprint-4	Classified Model	We prediction separate identification of the digital letter and predict letters
	Final Result	The image can be predict with the given data set Model

4.2 Training the Model



Fitting the model

```
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5, batch_size=32)

  Epoch 1/5
  1875/1875 [==============================] - 184s 98ms/step - loss: 0.2451 - accuracy: 0.9497 - val_loss: 0.0966 - val_accuracy: 0.9
  715
  Epoch 2/5
  1875/1875 [==============================] - 183s 98ms/step - loss: 0.0694 - accuracy: 0.9785 - val_loss: 0.0971 - val_accuracy: 0.9
  714
  Epoch 3/5
  1875/1875 [==============================] - 183s 98ms/step - loss: 0.0487 - accuracy: 0.9850 - val_loss: 0.0829 - val_accuracy: 0.9
  782
  Epoch 4/5
  1875/1875 [==============================] - 177s 94ms/step - loss: 0.0382 - accuracy: 0.9877 - val_loss: 0.0881 - val_accuracy: 0.9
  769
```

Arguments:

steps_per_epoch : it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

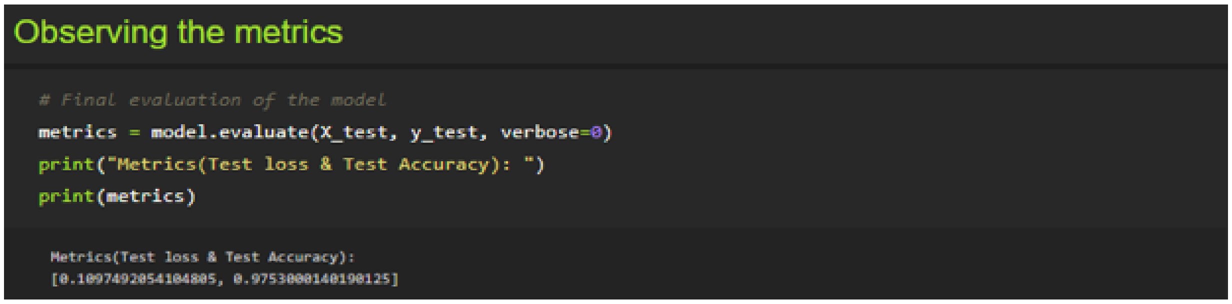
Epochs: an integer and number of epochs we want to train our model for.

Validation_data :

- an inputs and targets list
- a generator
- inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

Observing the Metrics



Observing the metrics

```
# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

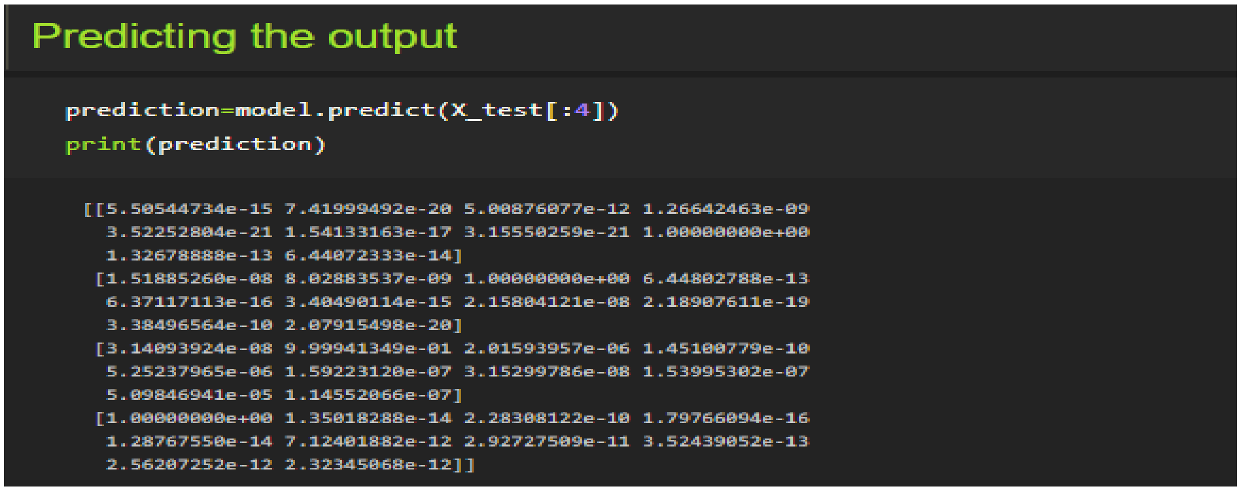
```
Metrics(Test loss & Test Accuracy):
[0.1097492054104805, 0.9753000140190125]
```

We here are printing the metrics which lists out the Test loss and Test accuracy

- Loss value implies how poorly or well a model behaves after each iteration of optimization.
- An accuracy metric is used to measure the algorithm's performance in an interpretable way.

4.3 Testing the Model

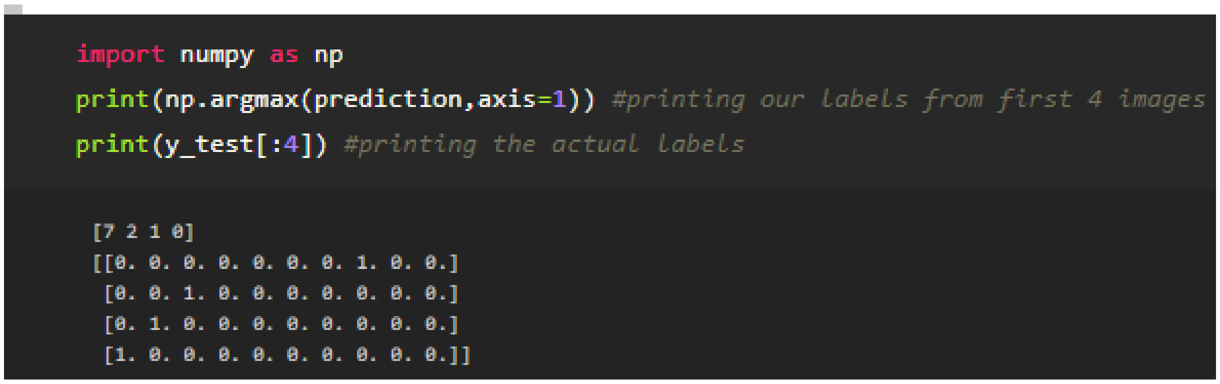
Firstly we are slicing the x_test data until the first four images. In the next step we the printing the predicted output.



Predicting the output

```
prediction=model.predict(X_test[:4])
print(prediction)
```

```
[[5.50544734e-15 7.41999492e-20 5.00876077e-12 1.26642463e-09
  3.52252804e-21 1.54133163e-17 3.15550259e-21 1.00000000e+00
  1.32678888e-13 6.44072333e-14]
 [1.51885260e-08 8.02883537e-09 1.00000000e+00 6.44802788e-13
  6.37117113e-16 3.40490114e-15 2.15804121e-08 2.18907611e-19
  3.38496564e-10 2.07915498e-20]
 [3.14093924e-08 9.99941349e-01 2.01593957e-06 1.45100779e-10
  5.25237965e-06 1.59223120e-07 3.15299786e-08 1.53995302e-07
  5.09846941e-05 1.14552066e-07]
 [1.00000000e+00 1.35018288e-14 2.28308122e-10 1.79766094e-16
  1.28767550e-14 7.12401882e-12 2.92727509e-11 3.52439052e-13
  2.56207252e-12 2.32345068e-12]]
```

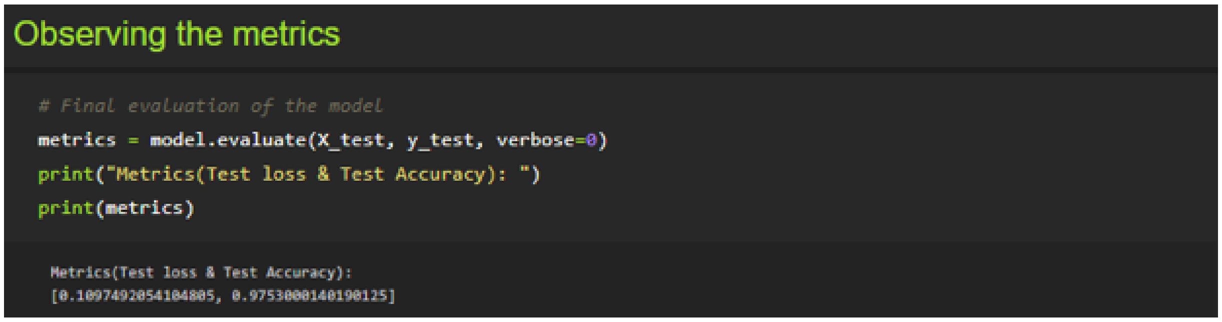


```
import numpy as np
print(np.argmax(prediction,axis=1)) #printing our labels from first 4 images
print(y_test[:4]) #printing the actual labels
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

As we already predicted the input from the x_test. According to that by using argmax function here we are printing the labels with high prediction values .

Observing the Metrics



Observing the metrics

```
# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy):
[0.1097492054104805, 0.9753000140190125]
```

We here are printing the metrics which lists out the Test loss and Test accuracy

- Loss value implies how poorly or well a model behaves after each iteration of optimization.
- An accuracy metric is used to measure the algorithm's performance in an interpretable way.

CHAPTER-5

CONCLUSION

An implementation of Handwritten Digit Recognition using Deep Learning has been implemented in this paper. Additionally, some of the most widely used Machine Learning algorithms i.e. CNN using Tensorflow have been trained and tested on the same data to draw a comparison as to why we require deep learning methods in critical applications like Handwritten Digit Recognition. In this paper, I have shown that that using Deep Learning techniques, a very high amount of accuracy can be achieved. Using the Convolutional Neural Network with Keras and Theano as backend, I am able to get an accuracy of 95.72%. Every tool has its own complexity and accuracy. Although, we see that the complexity of the code and the process is bit more as compared to normal Machine Learning algorithms but looking at the accuracy achieved, it can be said that it is worth it. Also, the current implementation is done only using the CPU Thus we settled on classifying a given handwritten digit image as the required digit using three different algorithms and consequently testing its accuracy.

In future we are planning to further explore the topic to recognize people’s handwriting.

APPENDIX-A(CODING)

Creating an HTML file:

```
<!DOCTYPE html>

<html>

<head>

<title>Home</title>

<style>

Body

{

background:url("https://dm0qx8t0i9gc9.cloudfront.net/thumbnails/video/BXJT8TRDeiymbx54w
/videoblocks-binary-code-black-and-green-background-with-digits-moving-on-
screen_bswlauoflw_thumbnail-1080_01.png");

background-attachment: fixed;

background-color: black;

}

. pd{

padding-bottom:100%;}

.navbar

{

margin: 0px;

padding:20px;

background-color:rgb(219, 213, 213);

opacity:0.6;
}

color:black;

font-family:'Roboto',sans-serif;

font-style: italic;

border-radius:20px;

font-size:25px;

}

a

{ss

color:grey;

float:right;

text-decoration:none;

font-style:normal;

padding-right:20px;

}

a:hover{

background-color:black;

color:white;

border-radius:15px;

font-size:30px;

padding-left:10px;

}

P

{
```

```
color:black;

font-style:italic;

font-size:30px;

font-style: normal;

font-weight: 700;

}

</style>

</head>

<body>

<div class="navbar">

<a href="/upload">Recognize</a>

<a href="/home">Home</a>

<br>

</div>

<br>

<center><b class="pd"><font color="black" size="15" font-family="Comic Sans MS"
>Handwritten Recognition System</font></b></center>

<div>

<br>

<center>

<p>
```

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 rising artificial neural networks/convolution neural network.

```
</center>

</div>

</body>

</html>
```

Web.html

```
<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Recognize</title>

<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>

<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

<script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">

<style>

body[

background:
url("https://dm0qx8t0i9gc9.cloudfront.net/thumbnails/video/BXJT8TRDeiymbx54w/videoblock
s-binary-code-black-and-green-background-with-digits-moving-on-
screen_bswlauoflw_thumbnail-1080_01.png");

background-attachment: fixed;

background-size: cover;

}

.bar

{

margin: 0px;

 padding:20px;

    background-color:rgb(205, 202, 202);

    opacity:0.6;

    color:black;

    font-family:'Roboto',sans-serif;

    font-style: italic;

    border-radius:20px;

    font-size:25px;

    }

    a

    {

    color:grey;

    float:right;

    text-decoration:none;

    font-style:normal;
```

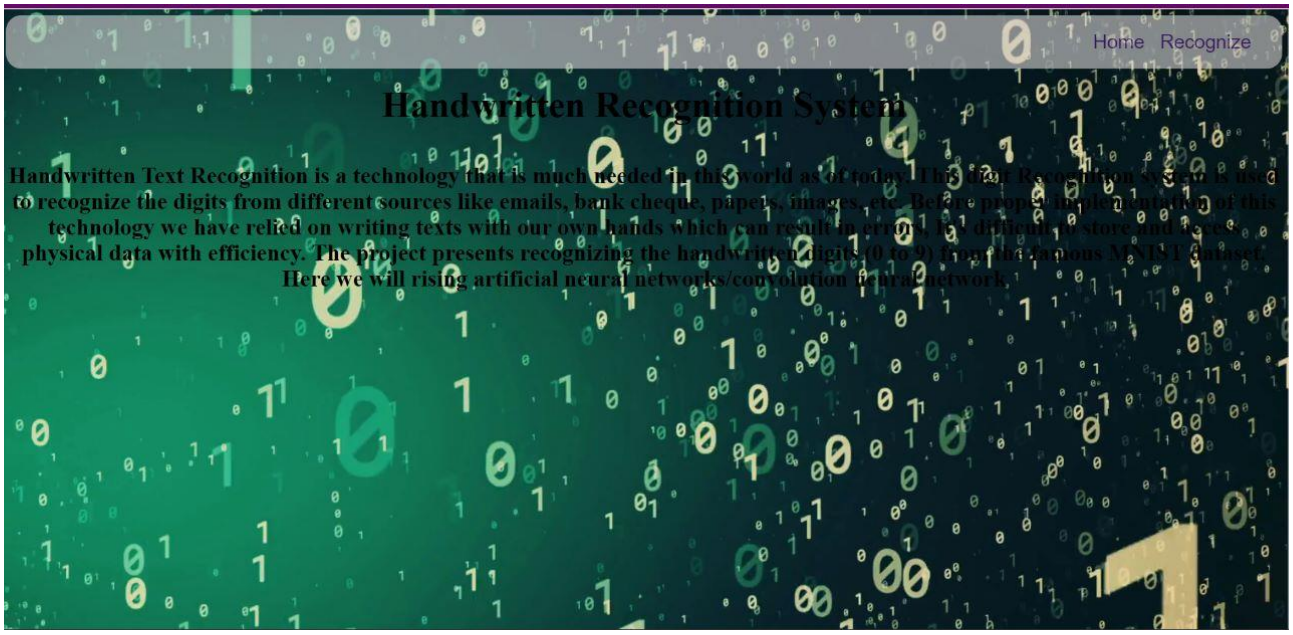
```
padding-right:20px;
}
a:hover{
background-color:black;
color:white;
border-radius:15px;
font-size:30px;
padding-left:10px;
}
body
{
    background-color: rgb(251, 251, 250);
}
</style>
</head>
<body>
<div class="bar">
<a href="/upload" >Recognize</a>
<a href="/home">Home</a>
<br>
</div>
    <div class="container">
```

```
        <center> <div id="content" style="margin-top:2em">{% block content %}{% endblock
%}</div></center>

      </div>

   </body>

   <footer>

      <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>

   </footer>

   </html>
```

App.py

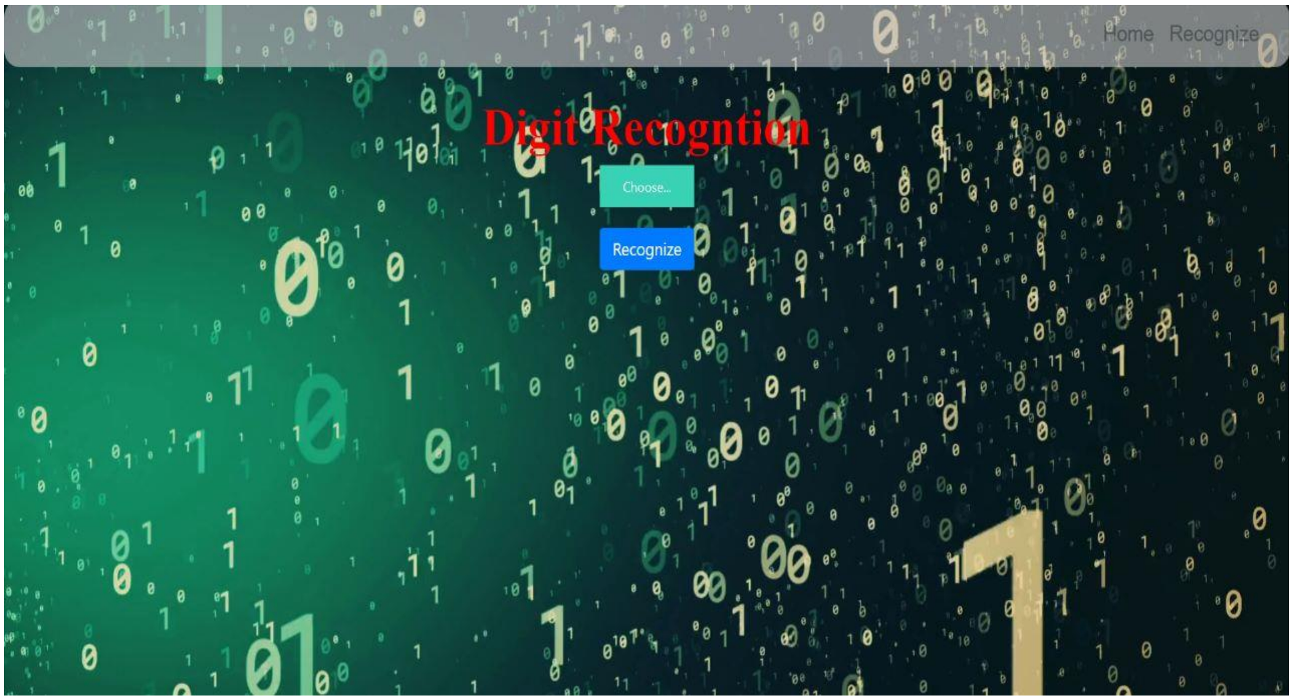
```
import os

import numpy as np #used for numerical analysis

from flask import Flask,request,render_template

# Flask-It is our framework which we are going to use to run/serve our application.

#request-for accessing file which was uploaded by the user on our application.

#render_template- used for rendering the html pages

import  tensorflow as tf

from tensorflow.keras.models import load_model #to load our trained model

from tensorflow.keras.preprocessing import image

from PIL import Image

app=Flask(__name__)#our flask app

model=load_model('mnistCNN.h5')#loading the model

@app.route("/") #default route

def about():
    return render_template("main.html")#rendering html page
@app.route("/home") #default route
def home():
    return render_template("main.html")#rendering html page
@app.route("/upload") #default route
def test():
    return render_template("index6.html")#rendering html page
@app.route("/predict",methods=["GET","POST"]) #route for our prediction
def upload_image_file():
   if request.method == 'POST':
      img = Image.open(request.files['file'].stream).convert("L")
      img = img.resize((28,28))
      im2arr = np.array(img)
      im2arr = im2arr.reshape(1,28,28,1)
      y_pred = model.predict(im2arr)
      predict = np.argmax(y_pred)
      print(predict)
      #return "Predicted Number: + str(pred) returning our output
      if(predict == 0):
          return render_template("0.html")
      elif(predict == 1):
          return render_template("1.html")
      elif(predict == 2):
          return render_template("2.html")
      elif(predict == 3):
          return render_template("3.html")
      elif(predict == 4):
          return render_template("4.html")
      elif(predict == 5):
          return render_template("5.html")
      elif(predict == 6):
          return render_template("6.html")
      elif(predict == 7):
          return render_template("7.html")
      elif(predict == 8):
          return render_template("8.html")
      elif(predict == 9):
          return render_template("9.html")
      else:
          return None
if __name__=="__main__":
    #app.run(debug=False)#running our app
    app.run(host='0.0.0.0', port=8000)
```

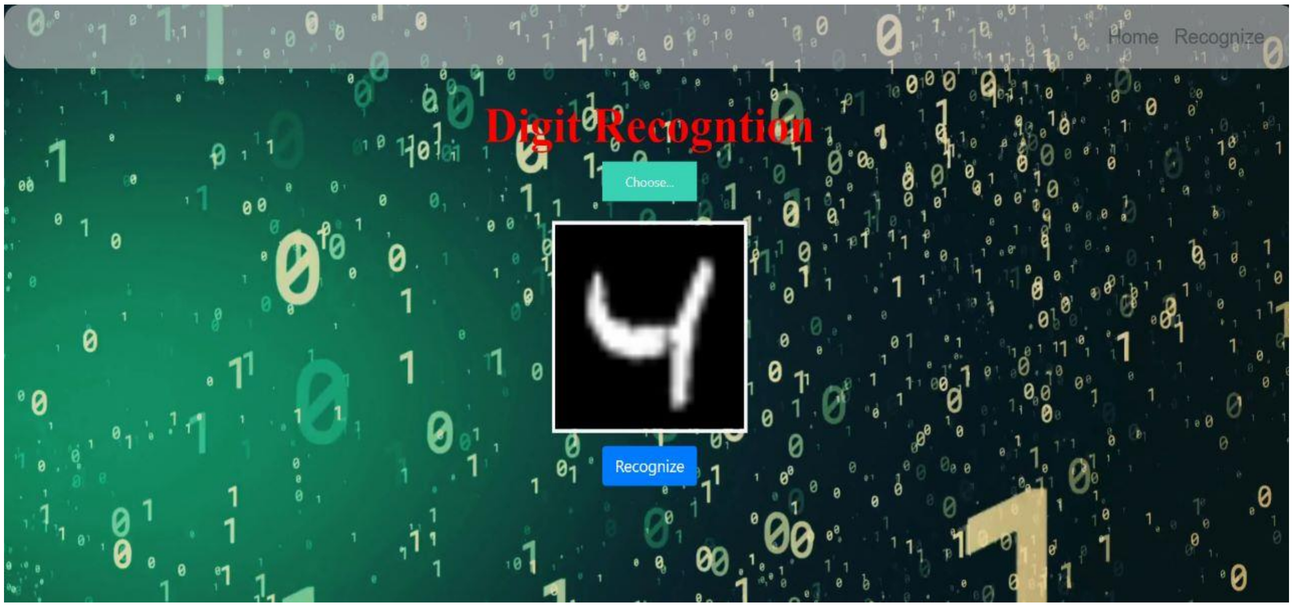
APPENDIX - B (OUTPUT)

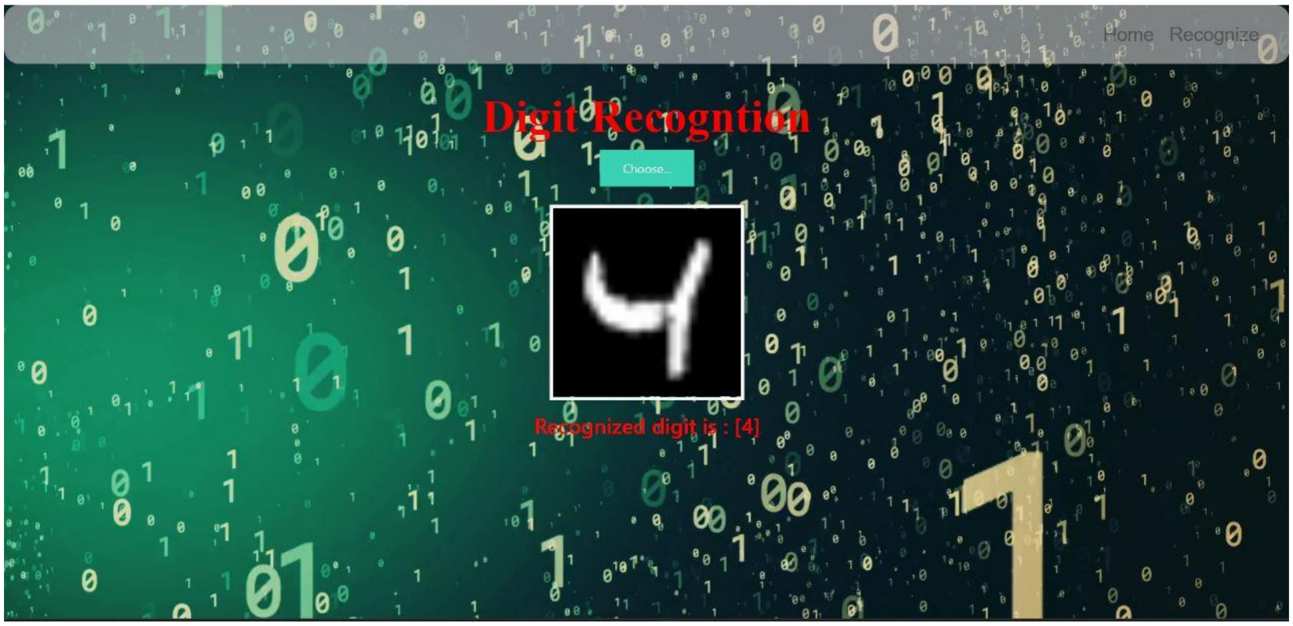


Handwritten Recognition System

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 rising artificial neural networks/convolution neural network.




Home
Recognize
Digit Recogntion
Choose...
Recognize


Home
Recognize
Digit Recogntion
Choose...
Recognized digit is : [4]

REFERENCES

[1] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, no. 4, pp. 541551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

[3] D. Hubel and T. Wiesel, "Aberrant visual projections in the Siamese cat," The Journal of physiology, vol. 218, no. 1, pp. 3362, 1971.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, p. 436, 2015.

[5] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," arXiv preprint arXiv:1202.2745, 2012.

[6] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in Competition and cooperation in neural nets: Springer, 1982, pp. 267-285.

[7] Y. LeCun et al., "Handwritten digit recognition with a backpropagation network," in Advances in neural information processing systems, 1990, pp. 396-404.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradientbased learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[9] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in Neural networks for perception: Elsevier, 1992, pp. 65-93.

[10] Y. LeCun, "LeNet-5, convolutional neural networks," URL: http://yann. lecun. com/exdb/lenet, vol. 20, 2015.

[11] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016.

[12] S. Haykin, "Neural networks: A comprehensive foundation: MacMillan College," New York, 1994.

[13] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," IEEE Transactions on Semiconductor Manufacturing, vol. 30, no. 2, pp. 135-142, 2017.

[14] K. G. Pasi and S. R. Naik, "Effect of parameter variations on accuracy of Convolutional Neural Network," in 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016, pp. 398-403: IEEE.

[15] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in Twenty-Second International Joint Conference on Artificial Intelligence, 2011.

[16] K. Isogawa, T. Ida, T. Shiodera, and T. Takeguchi, "Deep shrinkage convolutional neural network for adaptive noise reduction," IEEE Signal Processing Letters, vol. 25, no. 2, pp. 224-228, 2018.

[17] C. C. Park, Y. Kim, and G. Kim, "Retrieval of sentence sequences for an image stream via coherence recurrent convolutional networks," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 945-957, 2018.

[18] Y. Yin, J. Wu, and H. Zheng, "Ncfm: Accurate handwritten digits recognition using convolutional neural networks," in 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 525-531: IEEE.

[19] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing cnn on the loss layer," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4753-4762.

[20] A. Tavanaei and A. S. Maida, "Multi-layer unsupervised learning in a spiking convolutional neural network," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2023-2030: IEEE.