Project Report Format

1. **INTRODUCTION**
   1.1 Project Overview
   1.2 Purpose
2. **LITERATURE SURVEY**
   2.1 Existing problem
   2.2 References
   2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   3.1 Empathy Map Canvas
   3.2 Ideation & Brainstorming
   3.3 Proposed Solution
   3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   4.1 Functional requirement
   4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   5.1 Data Flow Diagrams
   5.2 Solution & Technical Architecture
   5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   6.1 Sprint Planning & Estimation
   6.2 Sprint Delivery Schedule
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   7.1 Understanding the Data
   7.2 Building the model
   7.3 Application Building
   7.4 Train the model on IBM
8. **TESTING**
   8.1 Test Cases
   8.2 User Acceptance Testing
9. **RESULTS**
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
    Source Code
    GitHub & Project Demo Link

Project Report

1. INTRODUCTION

1.1. 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 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. The 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

1.2. Purpose

Our purpose of the application is to create a Web application where the user can upload a handwritten digit image as input and give the correct predicted output.

2. LITERATURE SURVEY

2.1. Existing problem

Character recognition is becoming increasingly important in today's society. It makes human work easier and aids in the resolution of more difficult issues. One example is handwritten character recognition, which is widely used around the world. This strategy was developed in order to recognize zip codes or postal codes for use in mail sorting. This can help individuals involved in the difficult-to-read postal code mail sorting process. Researchers have been working for over thirty years on handwriting recognition. The number of businesses participating in handwriting recognition research has steadily increased over the last few years.In a number of years, Handwriting processing has advanced as a result of a combination of factors such as increased recognition rates and the use of complex systems.

2.2. References

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 a single hidden layer between the input and output layers, whereas a deep neural network has many hidden layers between the input and output layers. Deep neural networks employ multiple hidden layers to improve model performance and achieve greater accuracy than machine learning models. The majority of researchers work in the field of pattern recognition. Many patterns can be used in pattern recognition, such as handwritten numbers, characters, pictures, faces, sounds, and speech. This research focuses on the classification and recognition of handwritten digits. 1000 were used as test samples and 1000 as training samples. The USPS dataset contains 10000 image samples, 7291 of which are training samples and 2007 of which are 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 the accuracy from prior research. Additionally, using the USPS samples dataset, 98.22% training accuracy and 93.01% testing accuracy was 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**

A plethora of CNN classification algorithms has been proposed in the literature. These algorithms, however, do not account for proper filter size selection, data preparation, dataset constraints, or noise. As a result, few algorithms have significantly improved classification accuracy. The paper makes the following contributions to overcoming the drawbacks of these methods: First, the size of the effective receptive field (ERF) is calculated using domain knowledge. They use the ERF calculation to select a typical filter size, which improves the classification accuracy of our CNN. Second, excessive data produces inaccurate results, which reduces classification accuracy.Data preparation is performed prior to performing the data classification task to ensure that the dataset is free of any redundant or irrelevant variables to the goal variable. Third, data augmentation has been proposed as a method of reducing training and validation errors while avoiding dataset limitations. Fourth, the paper proposes that an additive white Gaussian noise with a threshold of 0.5 be added to the MNIST dataset to mimic natural factors that can affect image quality in the real world. Our CNN algorithm achieves state-of-the-art performance in handwritten digit recognition, with a recognition accuracy of 99.98% and 99.40% with 50% noise.

Paper 3: Handwritten Character Recognition using Neural Network and TensorFlow
Year: **2019**
Authors: **Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta**

In this study, offline handwritten character recognition will be performed using Tensorflow and a convolutional neural network. Using SoftMax Regression, one can assign probabilities to one of the many characters in the handwritten text that have values ranging from 0 to 1, summed to 1. The goal is to develop software that is extremely accurate and has a low level of spatial and temporal complexity. It was discovered that feature extraction strategies such as diagonal and direction produce significantly higher accuracy. Outcomes in comparison to other traditional vertical and horizontal techniques are also used. Because of their high noise tolerance, the neural network tried layers provide a more accurate outcome. In neural networks, the feed-forward model is a back-propagation algorithm that was primarily used to classify characters, recognize them, and receive ongoing training. In addition to these, normalizing and feature extraction produced better and more effective results. Character recognition is the result of precision. The paper will describe the best method for achieving more than 90% accuracy in Handwritten Character Recognition (HCR).

Paper 4: Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)
Year: **2020**
Authors: **Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh, and Byungun Yoon**

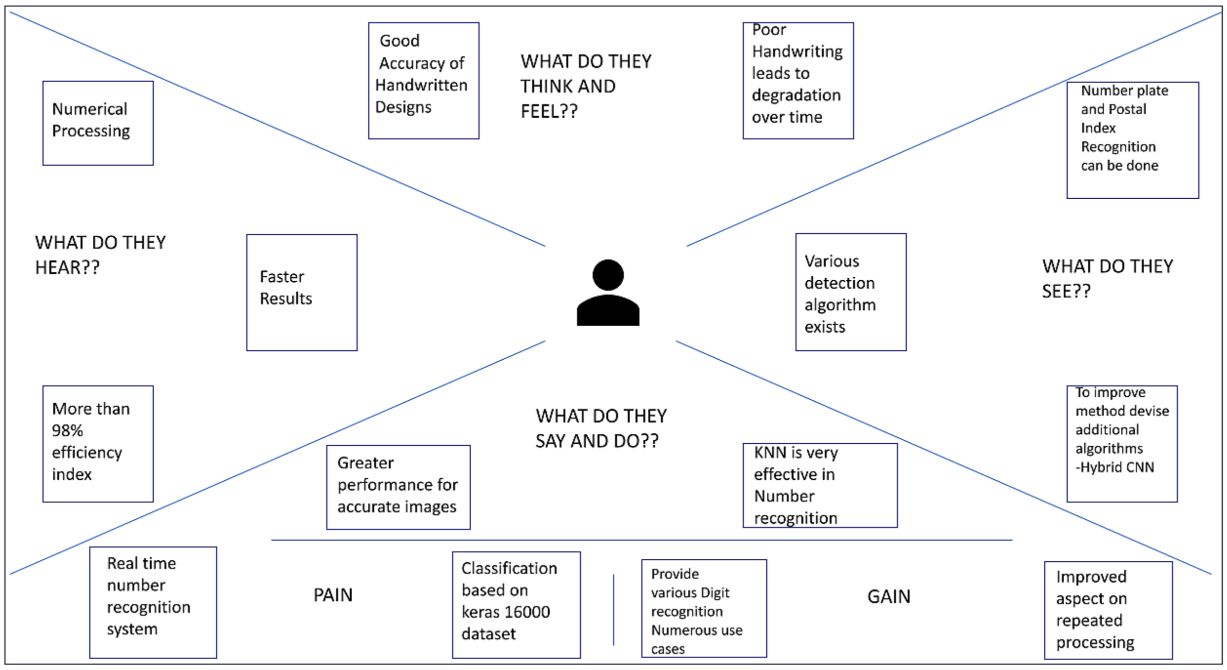
Traditional handwriting recognition systems have relied on customized features and a large amount of prior knowledge. Based on these conditions, training an optical character recognition (OCR) system is difficult. In recent years, deep learning approaches have enabled significant performance in the field of handwriting recognition research. Nonetheless, the growing amount of handwritten data, combined with the availability of vast computing capacity, necessitates improvements in recognition accuracy and warrants further investigation. CNNs are the best solution for solving handwriting recognition challenges because they are extremely good at perceiving the structure of handwritten characters/words in ways that aid in the automatic extraction of distinguishing features.The proposed work will look into various design options for CNN-based handwritten digit recognition, such as the number of layers, stride size, receptive field, kernel size, padding, and dilution. Furthermore, we plan to evaluate the efficacy of various SGD optimization techniques in improving the performance of handwritten digit recognition. Using ensemble architecture improves a network's recognition accuracy. Because ensemble structures increase computational overhead and testing complexity, we want to achieve equal accuracy by using a pure CNN design without ensemble architecture in this case. As a result, a CNN design is created to achieve higher accuracy than ensemble systems while reducing operational complexity and expense.As a result, a CNN design is created to achieve higher accuracy than ensemble systems while reducing operational complexity and cost. In addition, we show how an appropriate combination of learning parameters in the design of a CNN leads to a new absolute record in categorizing MNIST handwritten digits. We ran numerous tests and achieved 99.87% recognition accuracy for an MNIST dataset.

2.3. Problem Statement Definition

The problem statement tells us to create a handwritten digit recognition-based website, with good accuracy providing novel and vital aspects of prediction.

3. **IDEATION & PROPOSED SOLUTION**

3.1. Empathy Map Canvas


Good Accuracy of Handwritten Designs
WHAT DO THEY THINK AND FEEL??
Poor Handwriting leads to degradation over time
Numerical Processing
Number plate and Postal Index Recognition can be done
WHAT DO THEY HEAR??
Faster Results
Various detection algorithm exists
WHAT DO THEY SEE??
More than 98% efficiency index
WHAT DO THEY SAY AND DO??
To improve method devise additional algorithms -Hybrid CNN
Greater performance for accurate images
KNN is very effective in Number recognition
Real time number recognition system
PAIN
Classification based on keras 16000 dataset
Provide various Digit recognition Numerous use cases
GAIN
Improved aspect on repeated processing

3.2. Ideation and Brainstorming

As far as brainstorming and ideation are concerned, the team has discussed a lot of points covering

- High efficient algorithm
- Use in real-time prediction
- Deployment in Cloud server

- Deployment in Flask or Heroku server
- User-friendly interface
- More detailing about the algorithm
- Scanner integration
- Postal mail sorting
- Bank cheque processing
- Form data entry
- Reading postal address
- Bank cheque amounts
- Bank forms

Among these ideas, the top ideas we finalized were

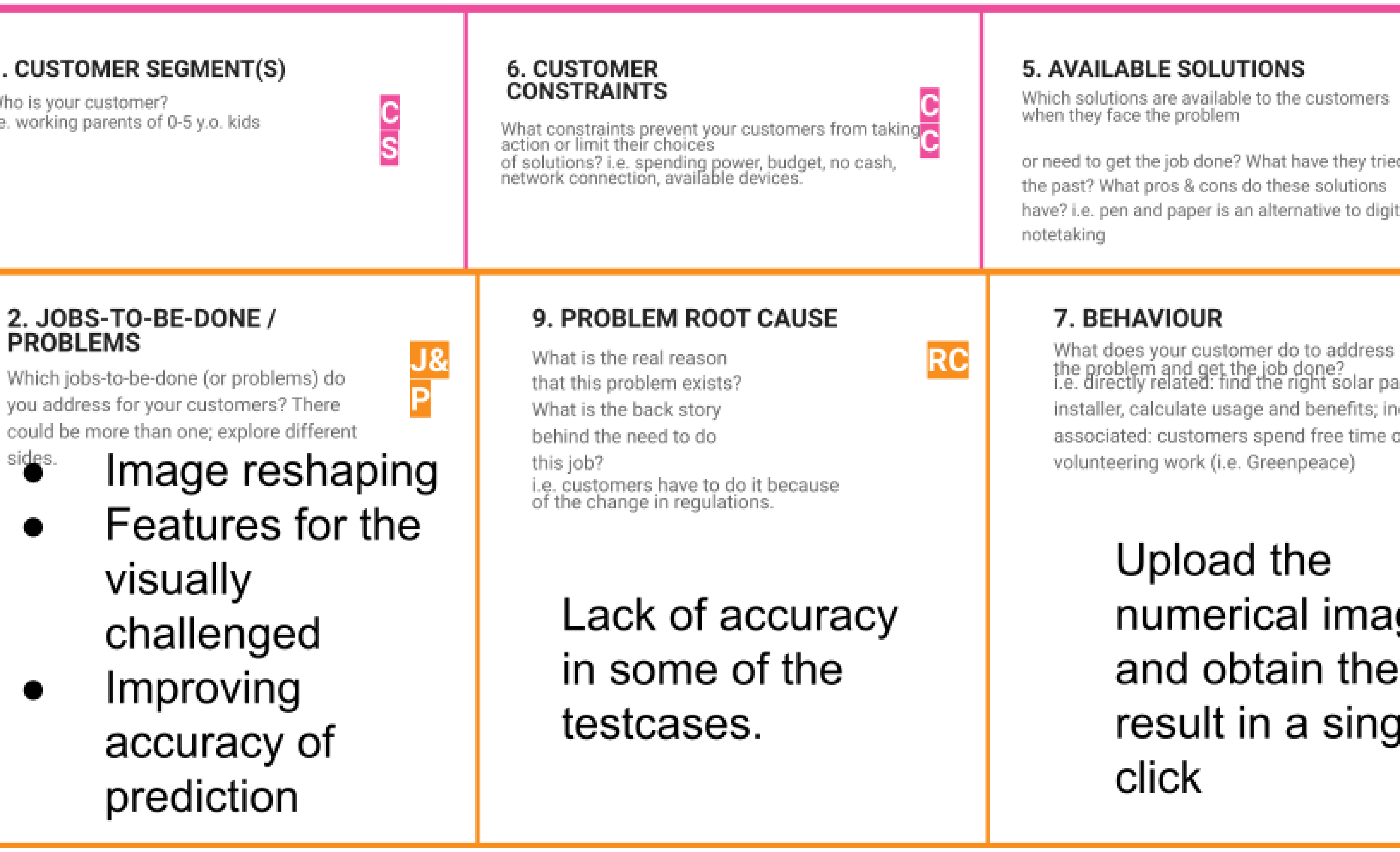
- Deployment in Flask or Heroku server
- Scanner integration
- Reading postal address

3.3. Proposed Solution

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	A solution for the classification of numerical images of different handwritings given using Convolutional Neural Networks.
2.	Idea / Solution description	An application that is used to detect the numerical value of the given image by uploading the image to a website. Once the image is uploaded, click on the “Predict” button. It will tell you the result of the number.
3.	Novelty / Uniqueness	There are similar solutions on the internet.

4.	Social Impact / Customer Satisfaction	This solution can help visually challenged people by uploading the image and getting an output response. Also, this can be used in a real-time image-to-text converter for faster documentation of data.
5.	Business Model (Revenue Model)	This application holds good potential in the market and this could be distributed to all people around the globe since it can be useful to everyone, especially as a visual aid.
6.	Scalability of the Solution	In the case of a server we can upgrade the server to handle more requests at a time which ensures scalability from the server side. In the case of application, the machine learning model can be converted to an API which could be further upgraded for advanced features.

3.4. Problem Solution Fit



. CUSTOMER SEGMENT(S)

CS

/ho is your customer?
e. working parents of 0-5 y.o. kids

6. CUSTOMER CONSTRAINTS

CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

5. AVAILABLE SOLUTIONS

Which solutions are available to the customers when they face the problem

or need to get the job done? What have they tried the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digit notetaking

2. JOBS-TO-BE-DONE / PROBLEMS

J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Image reshaping
- Features for the visually challenged
- Improving accuracy of prediction

9. PROBLEM ROOT CAUSE

RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Lack of accuracy in some of the testcases.

7. BEHAVIOUR

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar pa installer, calculate usage and benefits; in associated: customers spend free time o volunteering work (i.e. Greenpeace)

Upload the numerical imag and obtain the result in a sing click

3. TRIGGERS

TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Need for more accurate and faster prediction of data.

4. EMOTIONS: BEFORE / AFTER

EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Inaccurate > precision, speed, accuracy

10. YOUR SOLUTION

SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

An application to provide faster and more accurate predictions by uploading images and adding more preprocessing to obtain a clear output.

8. CHANNELS of BEHAVIOUR

CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

All the uploading and prediction of results will be done in online mode.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

No feature is available in offline mode.

4. **REQUIREMENT ANALYSIS**

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Experience	A responsive user interface and user experience is needed to attract more users and also to have good user feedback.
FR-2	Deployment in cloud platforms	The system has the potential to deploy the trained CNN model in the cloud.
FR-3	Image Output Prediction	The trained model should predict the number in a more accurate and precise manner. Incorrect predictions can cause a degradation in the reliability in the minds of users.
FR-4	Pertinence in the application domain	The CNN model which is used must be relevant to the application domain and trained for maximum accuracy.

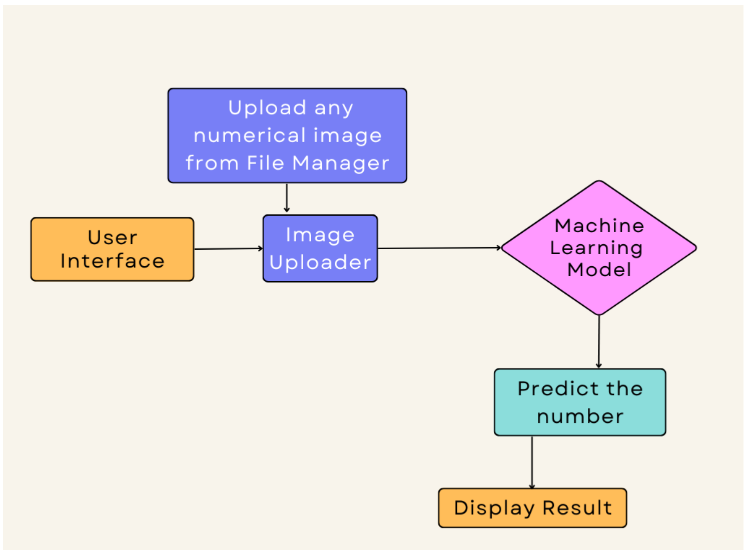
4.2 Non - Functional Requirements

Following are the non-functional requirements of the proposed solution.

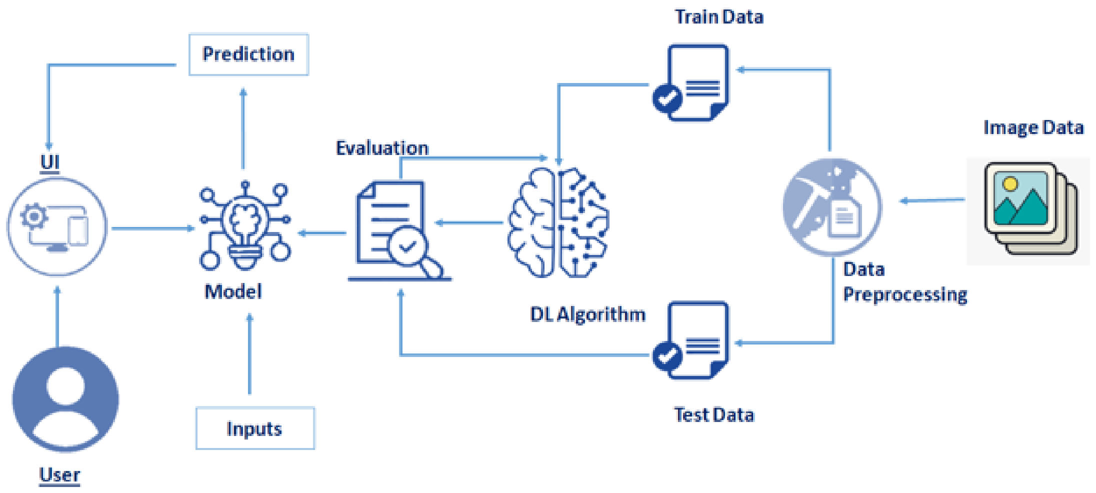
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	The website should act as an interface between users and the neural network-based model.
NFR-2	**Security**	Actually, no security is needed to be provided as of now, since it involves only the prediction of numerical images.
NFR-3	**Reliability**	The system will be able to operate for at least 99% of the year, and also it can give more than 98% accurate results which makes it more reliable.
NFR-4	**Performance**	The system should be able to predict and display the output in 3-5 seconds.
NFR-5	**Availability**	The model shall be available for public usage until it is in operational mode.
NFR-6	**Scalability**	The system can be able to accommodate one million concurrent users without any degradation in performance. For more accommodation, cloud-based hosting can be enabled.

5. **PROJECT DESIGN**

5.1. Data Flow Diagrams



5.2. Solution & Technical Architecture



5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Image Input	USN-1	As a user, I can upload the image containing numbers from the file manager	Image is successfully uploaded	High	Sprint-1
	Project Overview Interface	USN-2	As a user, I will be able to get the relevant details about the website and the steps involved in the process	Text and image-wise details are displayed successfully.	Medium	Sprint-1
	Preprocessing for data and model building	USN-3	As a user, I have to wait for a few seconds based on my internet connectivity to obtain the required output.	Display “Output: Number” is successfully executed	High	Sprint-2
	Obtaining accurate results	USN-3	As a user, I will be able to get a maximum accurate output	Output matches with the numerical image given	High	Sprint-2

6. **PROJECT PLANNING AND SCHEDULING**

6.1. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Launching of Web Application	USN-1	As a user, I can download and launch the application and perform the prediction of numerical images.	1	Medium	**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**
Sprint-2	Local File Upload interface	USN-2	As a user, I need a local storage interface to get the image data from my storage and use it for prediction.	1	Medium	**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	CNN Based AI Model Training	USN-3	As a user, I need the CNN based AI model to be ready so that I can do a good accurate result-giving prediction with high precision scores.	2	High	**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**
Sprint-3	Cloud Deployment	USN-4	As a user, I need to access the application anywhere, so I need a cloud deployment so that everyone can perform predictions anytime	1	Medium	**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**
Sprint-3	Prediction and Result checking	USN-5	As a user, I need the prediction to be done and to check whether the prediction is correct or not through visual outputs.	2	High	**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**
Sprint-4	Image Processing	USN-6	As a user, I need the image to be altered so that we can do prediction based on the mode built earlier.			**Kishore M** **Pradeep S** **Nissanth NA** **Kishore K** **Karthikeyan A**

6.2. Sprint Delivery Schedule

S.No	MILESTONE	ACTIVITIES	DATE
1	**Preparation phase**	Pre-requisites	22 Aug -28 Aug 2022
		Prior knowledge	
		Project structure	

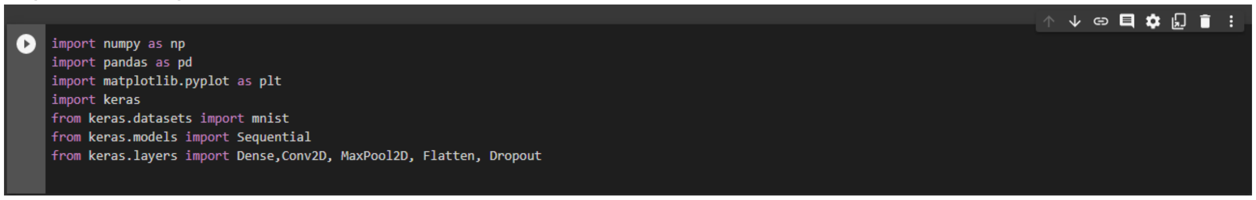
		Project Flow	
		Project Objectives	
		Registrations	
		Environment Set-up	
2	**Ideation phase**	Literature Survey	30 Aug - 04 Sept 2022
		Empathy Map	05 Sept - 07 Sept 2022
		Problem Statement	09 Sept-11 Sept 2022
		Ideation Phase	13 Sept - 16 Sept 2022

3	**Project Design Phase-I**	Proposed Solution	19 Sept - 24 Sept 2022
		Problem Solution Fit	25 Sept -27 Sept 2022
		Solution Architecture	29 Sept-01 Oct 2022
4	**Project Design Phase-II**	Customer Journey	03 Oct -07 Oct 2022
		Requirement Analysis	08 Oct -11 Oct 2022
		Data Flow Diagrams	12 Oct -14 Oct 2022
		Technology Architecture	15 Oct -17 Oct 2022
5	**Project Planning**	Milestones and Tasks	18 Oct -19 Oct 2022
		Sprint Schedules	20 Oct -23 Oct 2022
	Project	Sprint-1	25 Oct-30 Oct 2022
		Sprint-2	02 Oct -07 Nov 2022
		Sprint-3	08 Nov - 13 Nov 2022
		Sprint-4	15 Nov -19 Nov 2022

7. **CODING AND SOLUTIONING**

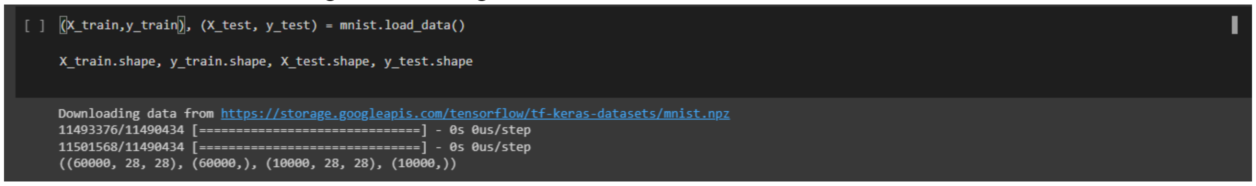
7.1. Understanding data

1. Import the required libraries and datasets



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense,Conv2D, MaxPool2D, Flatten, Dropout
```

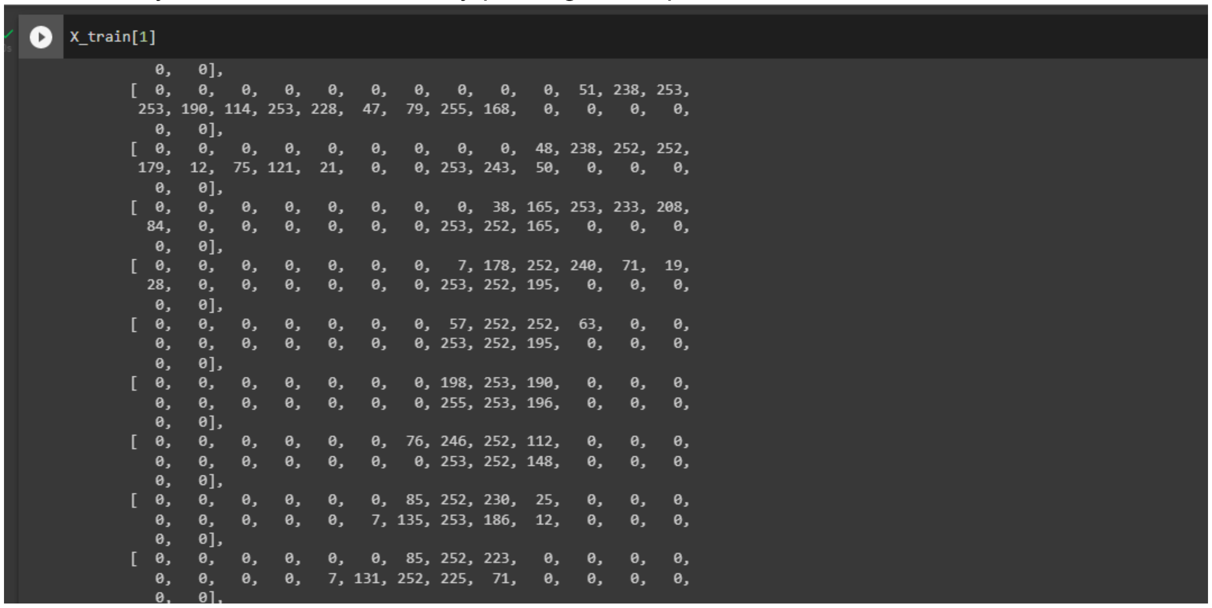
2. Load the data into training and testing datasets



```
[ ] (X_train,y_train), (X_test, y_test) = mnist.load_data()

    X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

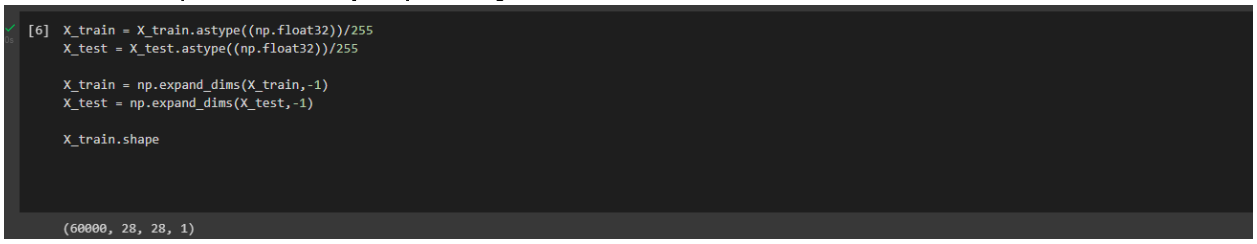
3. Analyze the obtained data by printing a sample value.



```
X_train[1]
```

```
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  51, 238, 253,
        253, 190, 114, 253, 228,  47,  79, 255, 168,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  48, 238, 252, 252,
        179,  12,  75, 121,  21,   0,   0, 253, 243,  50,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  38, 165, 253, 233, 208,
         84,   0,   0,   0,   0,   0,   0, 253, 252, 165,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   7, 178, 252, 240,  71,  19,
         28,   0,   0,   0,   0,   0,   0, 253, 252, 195,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  57, 252, 252,  63,   0,   0,
          0,   0,   0,   0,   0,   0,   0, 253, 252, 195,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0, 198, 253, 190,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0, 255, 253, 196,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  76, 246, 252, 112,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0, 253, 252, 148,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  85, 252, 230,  25,   0,   0,   0,
          0,   0,   0,   0,   0,   7, 135, 253, 186,  12,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  85, 252, 223,   0,   0,   0,   0,
          0,   0,   0,   0,   7, 131, 252, 225,  71,   0,   0,   0,   0,
          0,   0],
```

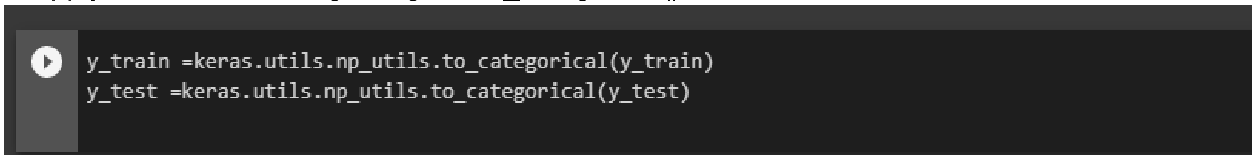
4. Now, reshape the data by expanding the dimensions of dataset



```
[6] X_train = X_train.astype((np.float32))/255
    X_test = X_test.astype((np.float32))/255

    X_train = np.expand_dims(X_train,-1)
    X_test = np.expand_dims(X_test,-1)

    X_train.shape
```

```
(60000, 28, 28, 1)
```

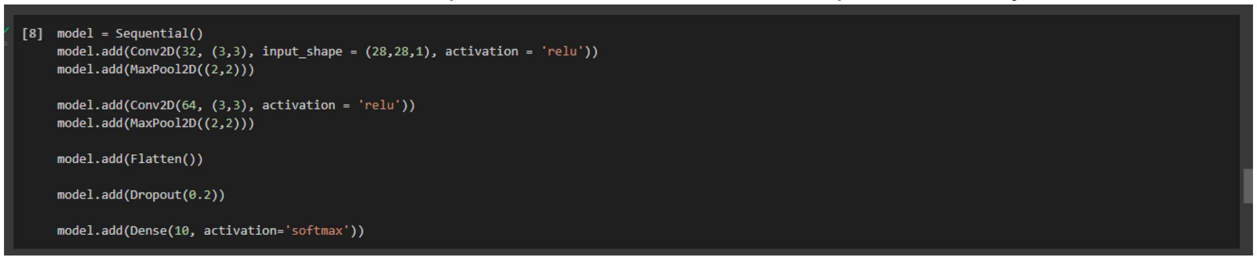
5. Apply One hot encoding using the to_categorical() function



```
y_train =keras.utils.np_utils.to_categorical(y_train)
y_test =keras.utils.np_utils.to_categorical(y_test)
```

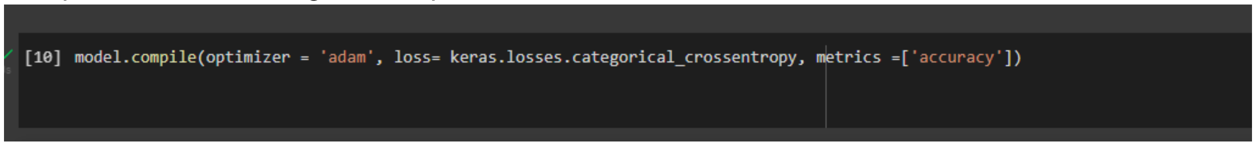
7.2. Model Building

1. Create a new variable with the Sequential model and add the required CNN layers to it.



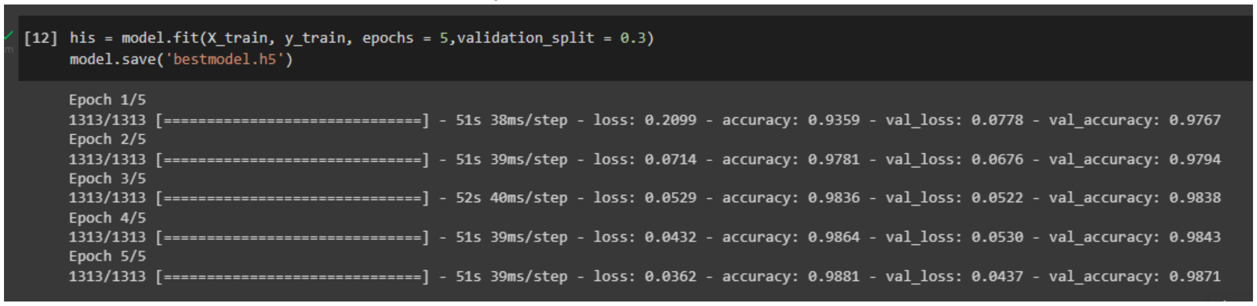
```
[8]  model = Sequential()
     model.add(Conv2D(32, (3,3), input_shape = (28,28,1), activation = 'relu'))
     model.add(MaxPool2D((2,2)))

     model.add(Conv2D(64, (3,3), activation = 'relu'))
     model.add(MaxPool2D((2,2)))

     model.add(Flatten())

     model.add(Dropout(0.2))

     model.add(Dense(10, activation='softmax'))
```

2. Compile the model using Adam optimizer.



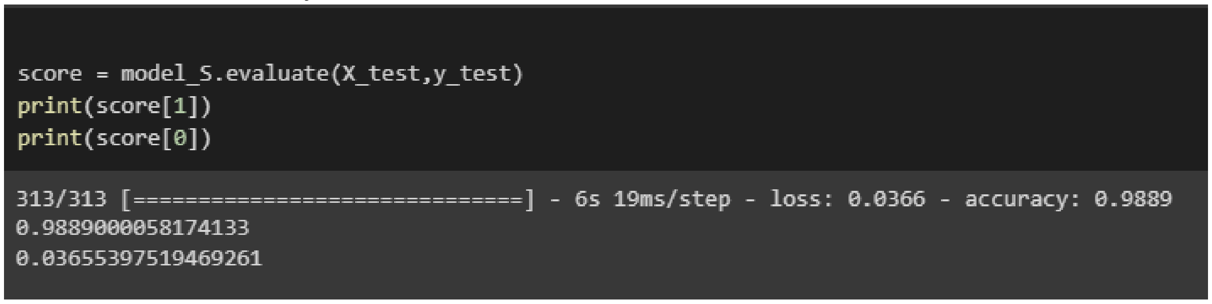
```
[10] model.compile(optimizer = 'adam', loss= keras.losses.categorical_crossentropy, metrics =['accuracy'])
```

3. Train the model for at least 5 epochs.



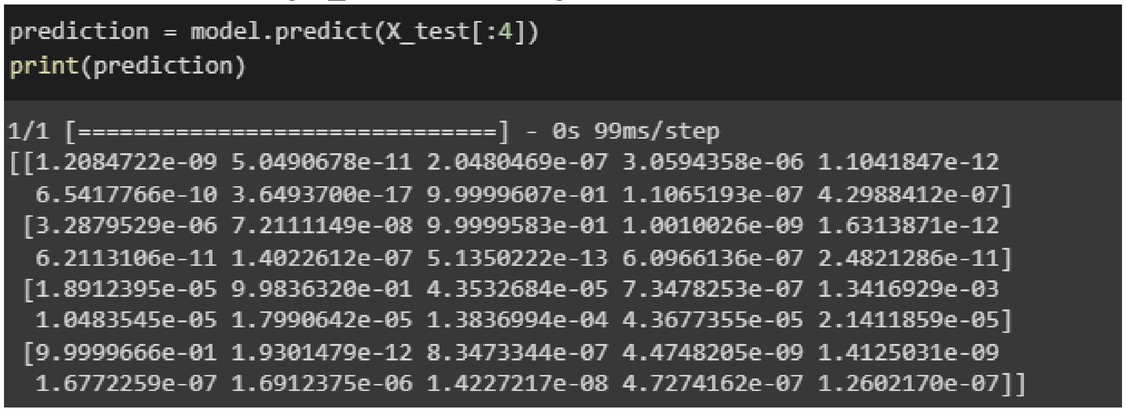
```
[12] his = model.fit(X_train, y_train, epochs = 5,validation_split = 0.3)
     model.save('bestmodel.h5')

     Epoch 1/5
     1313/1313 [==============================] - 51s 38ms/step - loss: 0.2099 - accuracy: 0.9359 - val_loss: 0.0778 - val_accuracy: 0.9767
     Epoch 2/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0714 - accuracy: 0.9781 - val_loss: 0.0676 - val_accuracy: 0.9794
     Epoch 3/5
     1313/1313 [==============================] - 52s 40ms/step - loss: 0.0529 - accuracy: 0.9836 - val_loss: 0.0522 - val_accuracy: 0.9838
     Epoch 4/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0432 - accuracy: 0.9864 - val_loss: 0.0530 - val_accuracy: 0.9843
     Epoch 5/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0362 - accuracy: 0.9881 - val_loss: 0.0437 - val_accuracy: 0.9871
```

4. Observe the accuracy and losses incurred in the model



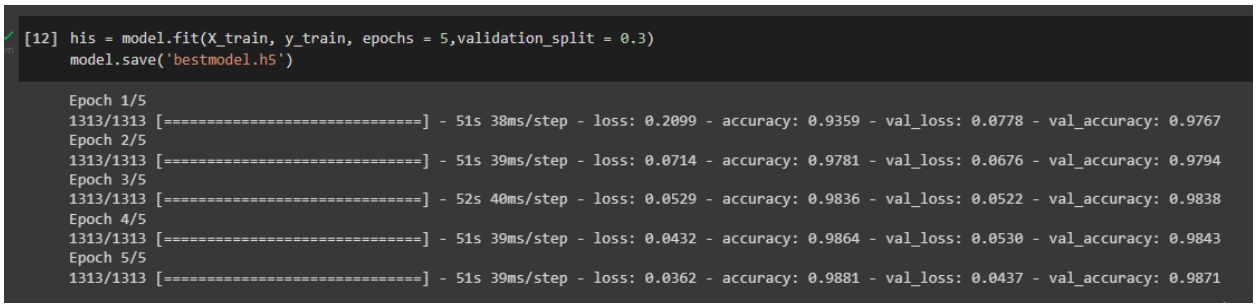
```
score = model_S.evaluate(X_test,y_test)
print(score[1])
print(score[0])

313/313 [==============================] - 6s 19ms/step - loss: 0.0366 - accuracy: 0.9889
0.9889000058174133
0.03655397519469261
```

5. Test the model using X_test dataset images



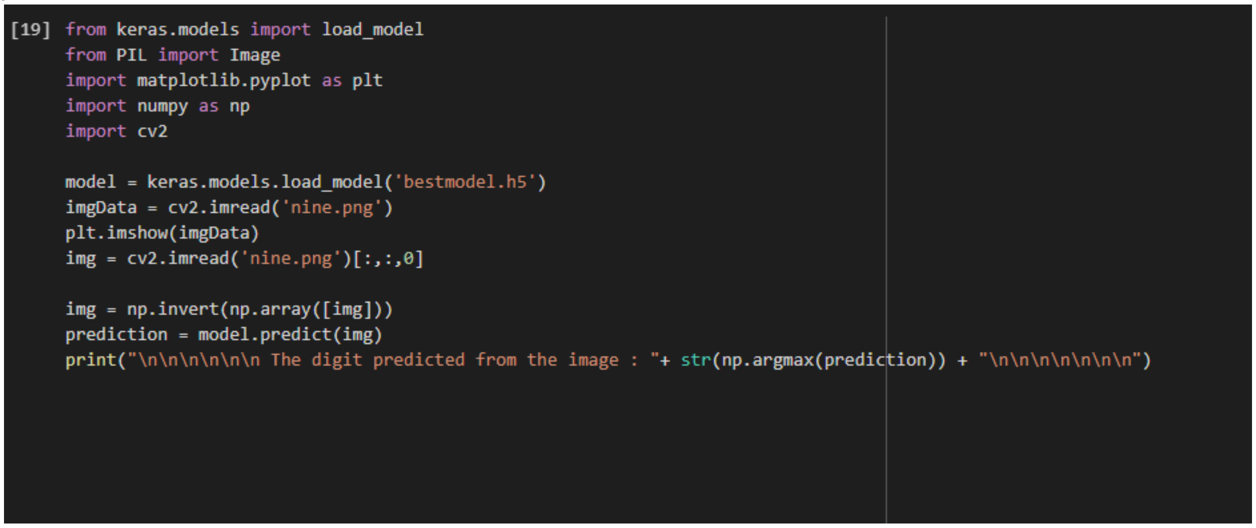
```
prediction = model.predict(X_test[:4])
print(prediction)

1/1 [==============================] - 0s 99ms/step
[[1.2084722e-09 5.0490678e-11 2.0480469e-07 3.0594358e-06 1.1041847e-12
  6.5417766e-10 3.6493700e-17 9.9999607e-01 1.1065193e-07 4.2988412e-07]
 [3.2879529e-06 7.2111149e-08 9.9999583e-01 1.0010026e-09 1.6313871e-12
  6.2113106e-11 1.4022612e-07 5.1350222e-13 6.0966136e-07 2.4821286e-11]
 [1.8912395e-05 9.9836320e-01 4.3532684e-05 7.3478253e-07 1.3416929e-03
  1.0483545e-05 1.7990642e-05 1.3836994e-04 4.3677355e-05 2.1411859e-05]
 [9.9999666e-01 1.9301479e-12 8.3473344e-07 4.4748205e-09 1.4125031e-09
  1.6772259e-07 1.6912375e-06 1.4227217e-08 4.7274162e-07 1.2602170e-07]]
```

6. Save the model in the name of “bestmodel.h5” file.

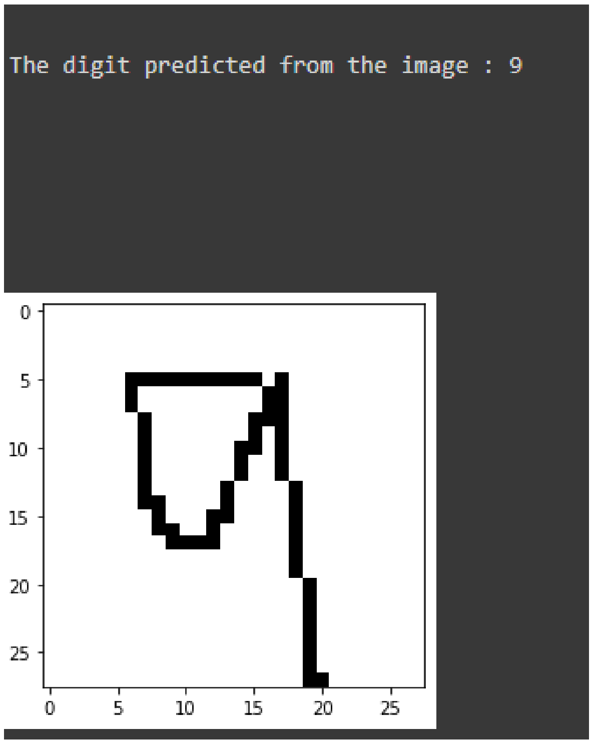


```
[12] his = model.fit(X_train, y_train, epochs = 5,validation_split = 0.3)
     model.save('bestmodel.h5')

     Epoch 1/5
     1313/1313 [==============================] - 51s 38ms/step - loss: 0.2099 - accuracy: 0.9359 - val_loss: 0.0778 - val_accuracy: 0.9767
     Epoch 2/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0714 - accuracy: 0.9781 - val_loss: 0.0676 - val_accuracy: 0.9794
     Epoch 3/5
     1313/1313 [==============================] - 52s 40ms/step - loss: 0.0529 - accuracy: 0.9836 - val_loss: 0.0522 - val_accuracy: 0.9838
     Epoch 4/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0432 - accuracy: 0.9864 - val_loss: 0.0530 - val_accuracy: 0.9843
     Epoch 5/5
     1313/1313 [==============================] - 51s 39ms/step - loss: 0.0362 - accuracy: 0.9881 - val_loss: 0.0437 - val_accuracy: 0.9871
```

7. Test the saved model by adding an image from google drive or from your local drive and predict the result.



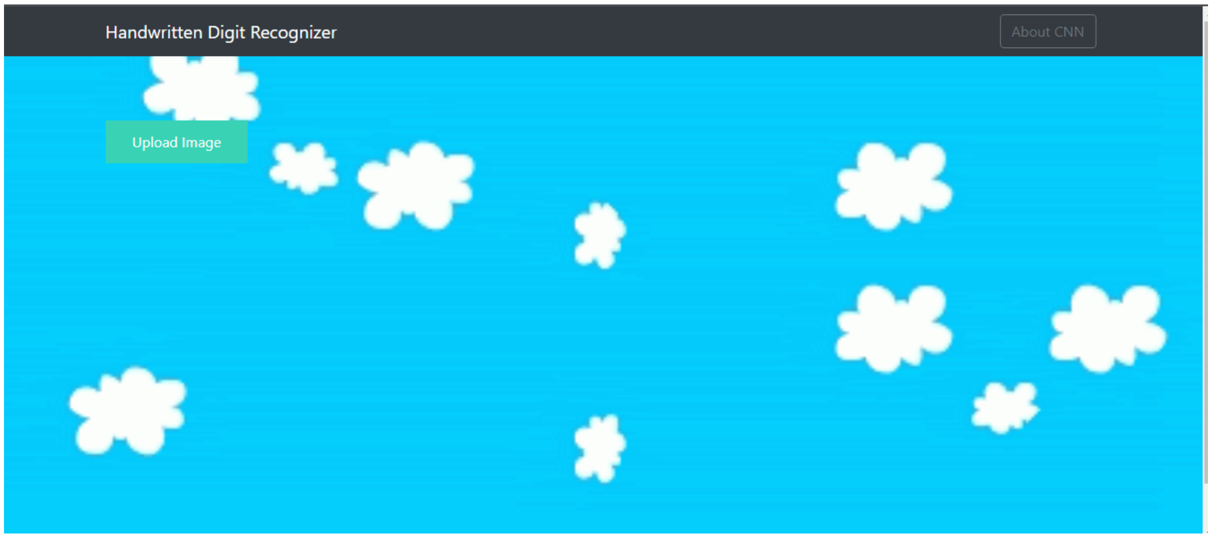
```
[19] from keras.models import load_model
     from PIL import Image
     import matplotlib.pyplot as plt
     import numpy as np
     import cv2

     model = keras.models.load_model('bestmodel.h5')
     imgData = cv2.imread('nine.png')
     plt.imshow(imgData)
     img = cv2.imread('nine.png')[:,:,0]

     img = np.invert(np.array([img]))
     prediction = model.predict(img)
     print("\n\n\n\n\n\n The digit predicted from the image : "+ str(np.argmax(prediction)) + "\n\n\n\n\n\n\n")
```

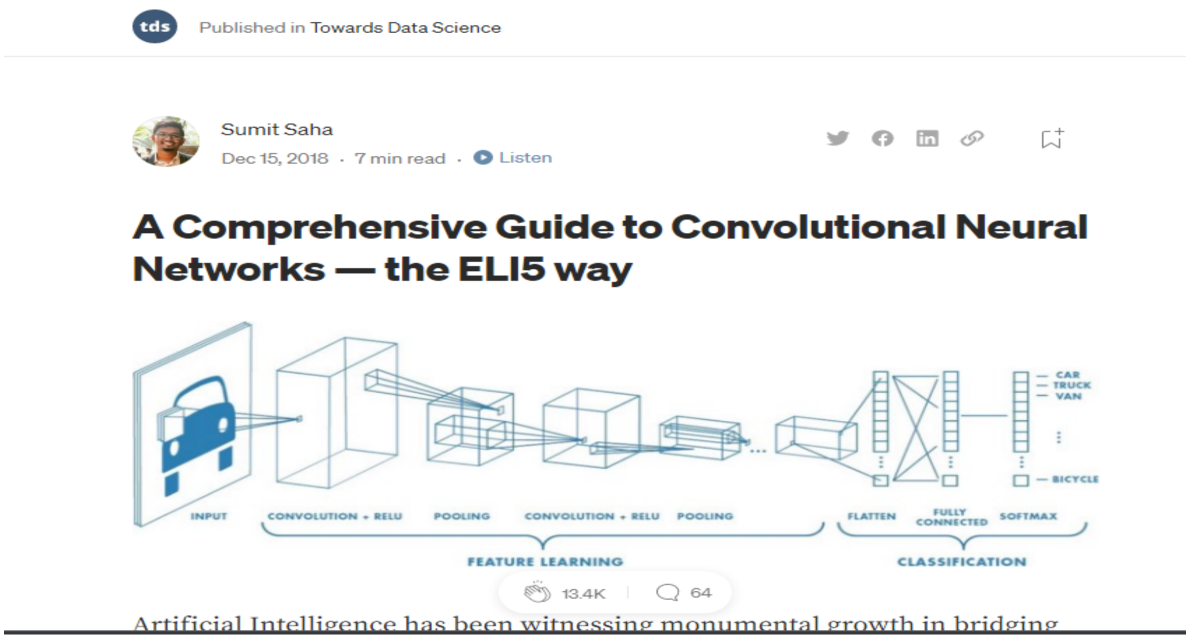


```
The digit predicted from the image : 9
```

7.3. Application Building

1. Create a new HTML File with a neat UI and reference material for Convolutional Neural Networks





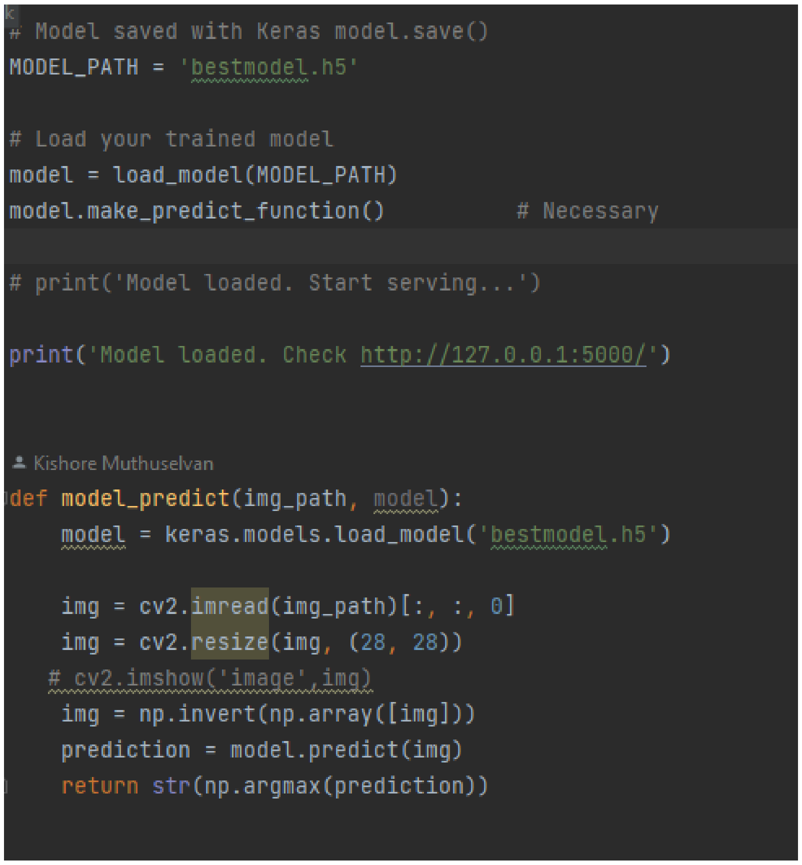
Published in Towards Data Science

Sumit Saha
Dec 15, 2018 · 7 min read · Listen

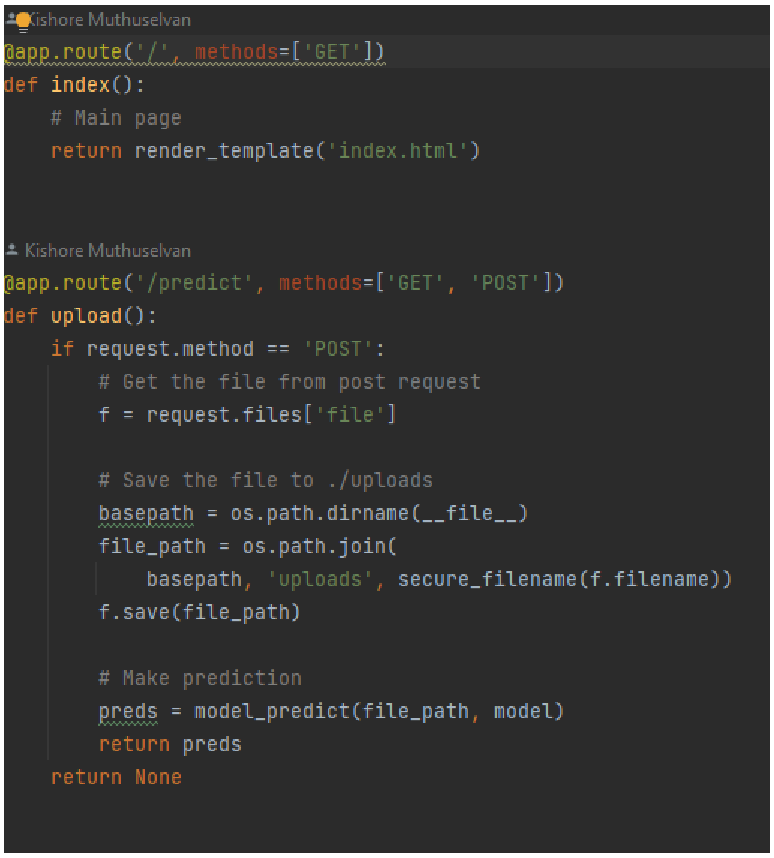
A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way

Artificial Intelligence has been witnessing monumental growth in bridging

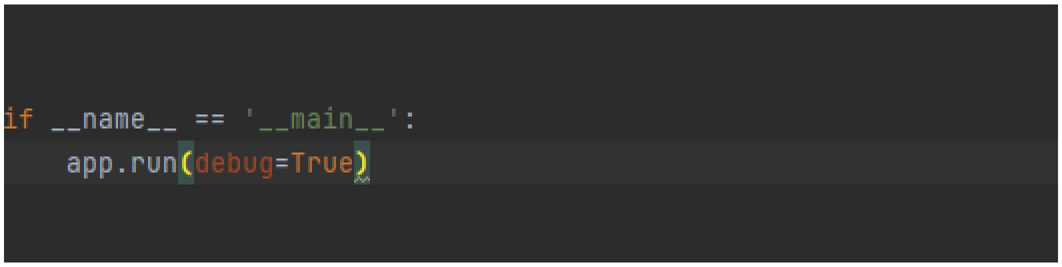
2. Build the python code inside the app.py file



```
# Model saved with Keras model.save()
MODEL_PATH = 'bestmodel.h5'

# Load your trained model
model = load_model(MODEL_PATH)
model.make_predict_function()          # Necessary

# print('Model loaded. Start serving...')

print('Model loaded. Check http://127.0.0.1:5000/')


Kishore Muthuselvan
def model_predict(img_path, model):
    model = keras.models.load_model('bestmodel.h5')

    img = cv2.imread(img_path)[:, :, 0]
    img = cv2.resize(img, (28, 28))
   # cv2.imshow('image',img)
    img = np.invert(np.array([img]))
    prediction = model.predict(img)
    return str(np.argmax(prediction))
```

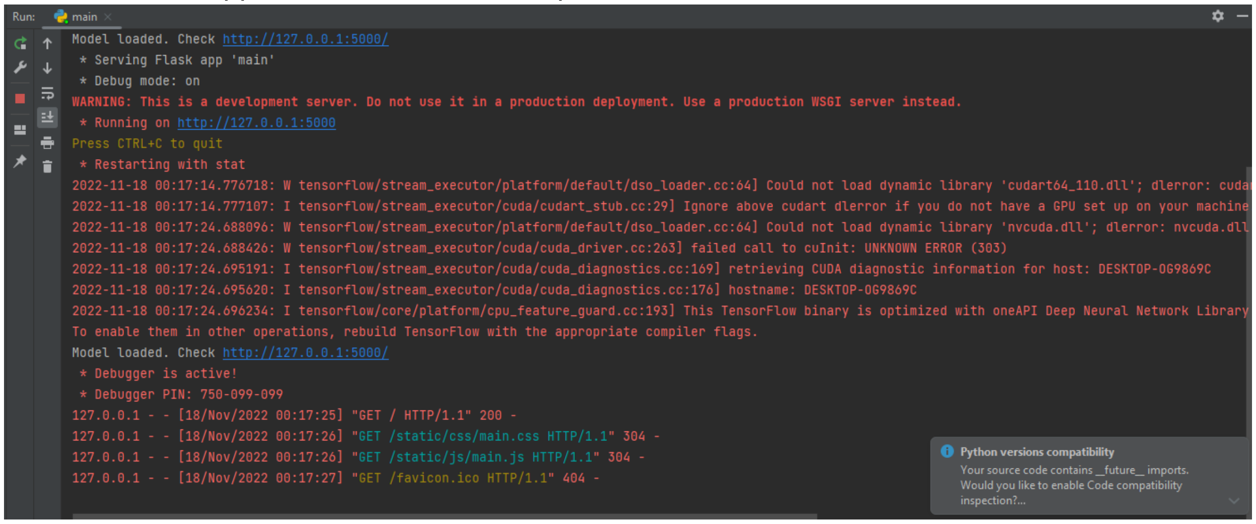


```
Kishore Muthuselvan
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')


Kishore Muthuselvan
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        preds = model_predict(file_path, model)
        return preds
    return None
```



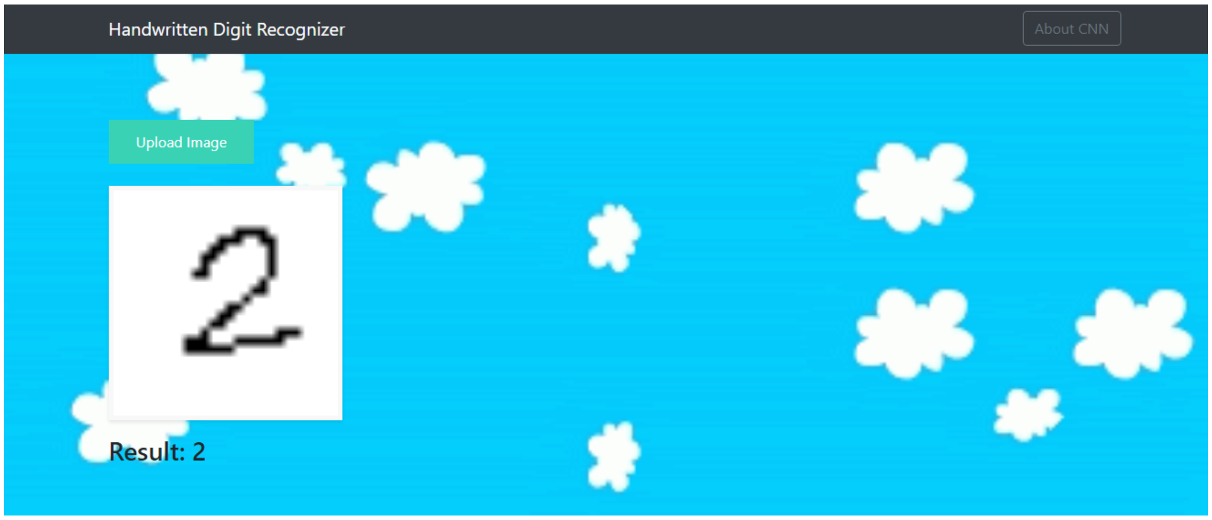
```python
if __name__ == '__main__':
    app.run(debug=True)
```

3. Run the application and see the output in localhost



```
Model loaded. Check http://127.0.0.1:5000/
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2022-11-18 00:17:14.776718: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cuda
2022-11-18 00:17:14.777107: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine
2022-11-18 00:17:24.688096: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll
2022-11-18 00:17:24.688426: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] failed call to cuInit: UNKNOWN ERROR (303)
2022-11-18 00:17:24.695191: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-0G9869C
2022-11-18 00:17:24.695620: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-0G9869C
2022-11-18 00:17:24.696234: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model loaded. Check http://127.0.0.1:5000/
 * Debugger is active!
 * Debugger PIN: 750-099-099
127.0.0.1 - - [18/Nov/2022 00:17:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 00:17:26] "GET /static/css/main.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 00:17:26] "GET /static/js/main.js HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 00:17:27] "GET /favicon.ico HTTP/1.1" 404 -
```

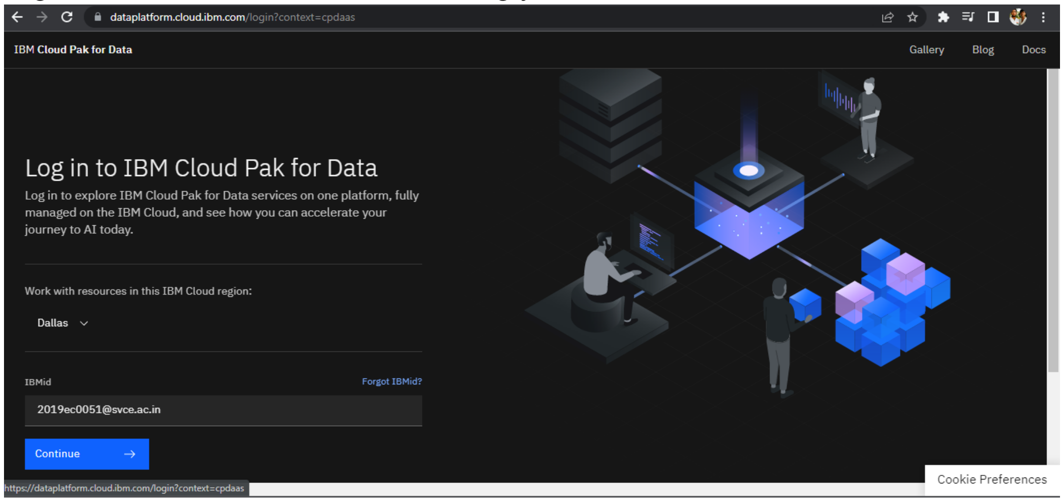
Python versions compatibility
Your source code contains __future__ imports. Would you like to enable Code compatibility inspection?...



Result: 2

7.4. Train the model on IBM

1. Login into the IBM Cloud Account using your credentials.


dataplatform.cloud.ibm.com/login?context=cpdaas
IBM Cloud Pak for Data
Gallery
Blog
Docs
Continue
Cookie Preferences
https://dataplatform.cloud.ibm.com/login?context=cpdaas

Log in to IBM Cloud Pak for Data

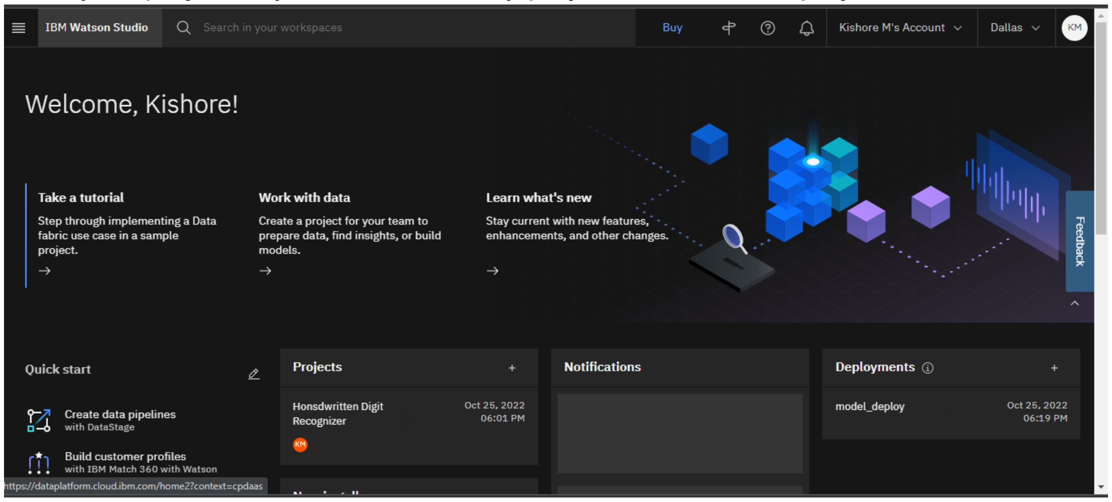
Log in to explore IBM Cloud Pak for Data services on one platform, fully managed on the IBM Cloud, and see how you can accelerate your journey to AI today.

Work with resources in this IBM Cloud region: Dallas

IBMid: 2019ec0051@svce.ac.in

Forgot IBMid?

2. Go to your project. If you don't have any project create a new project



Welcome, Kishore!

Take a tutorial
Step through implementing a Data fabric use case in a sample project.

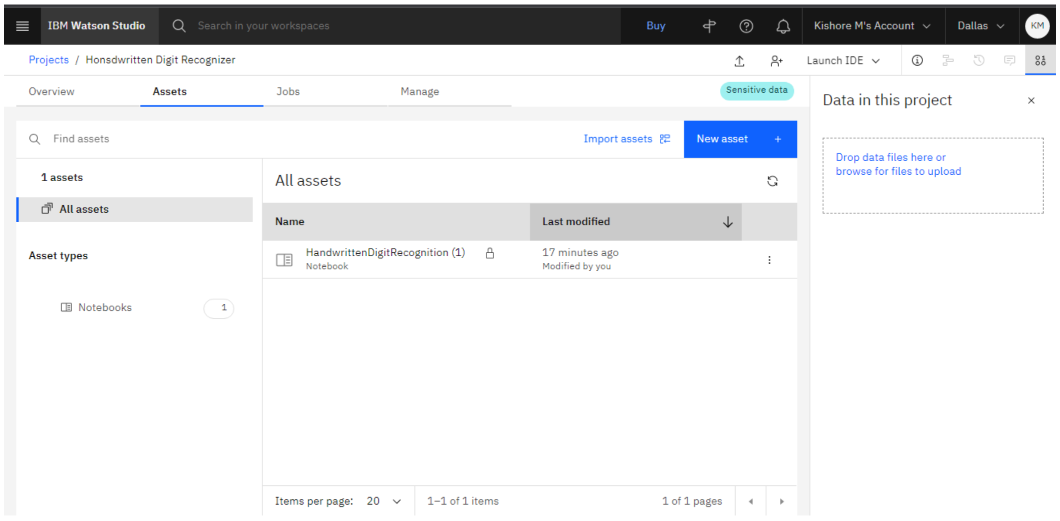
Work with data
Create a project for your team to prepare data, find insights, or build models.

Learn what's new
Stay current with new features, enhancements, and other changes.

Projects	
Honsdwritten Digit Recognizer	Oct 25, 2022 06:01 PM

Deployments	
model_deploy	Oct 25, 2022 06:19 PM

3. Inside the project upload a new **Jupyter Notebook** file as an asset and click the file.

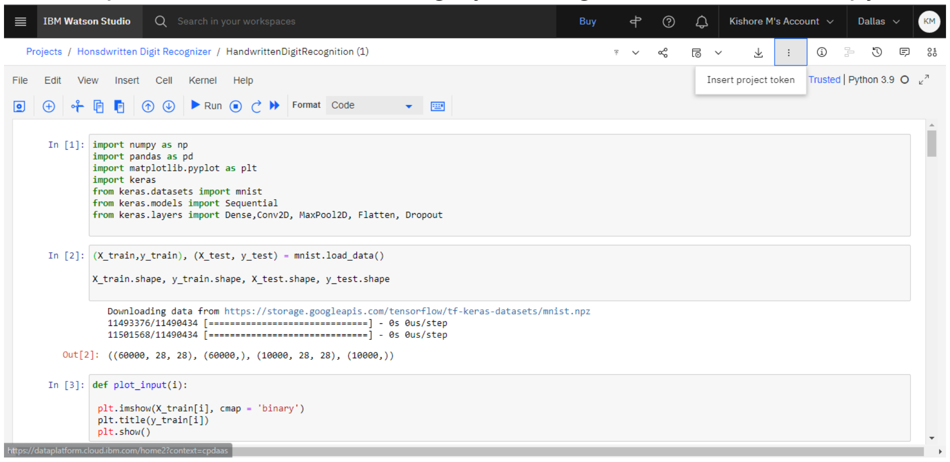


All assets

Name	Last modified
HandwrittenDigitRecognition (1) Notebook	17 minutes ago Modified by you

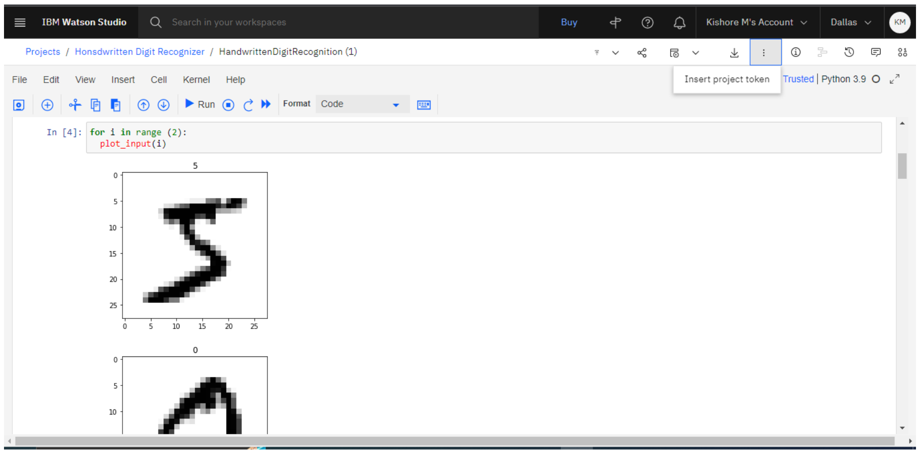
Items per page: 20 | 1–1 of 1 items | 1 of 1 pages

4. Now, perform the model training by running the codes of the Jupyter notebook file



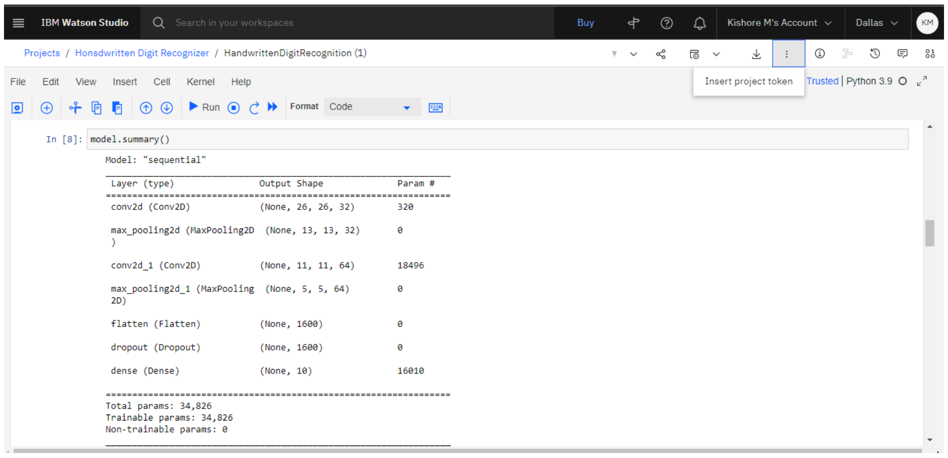
Projects / Honsdwritten Digit Recognizer / HandwrittenDigitRecognition (1)

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import keras
        from keras.datasets import mnist
        from keras.models import Sequential
        from keras.layers import Dense,Conv2D, MaxPool2D, Flatten, Dropout

In [2]: (X_train,y_train), (X_test, y_test) = mnist.load_data()

        X_train.shape, y_train.shape, X_test.shape, y_test.shape

          Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
          11493376/11490434 [==============================] - 0s 0us/step
          11501568/11490434 [==============================] - 0s 0us/step

Out[2]: ((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))

In [3]: def plot_input(i):

          plt.imshow(X_train[i], cmap = 'binary')
          plt.title(y_train[i])
          plt.show()
```



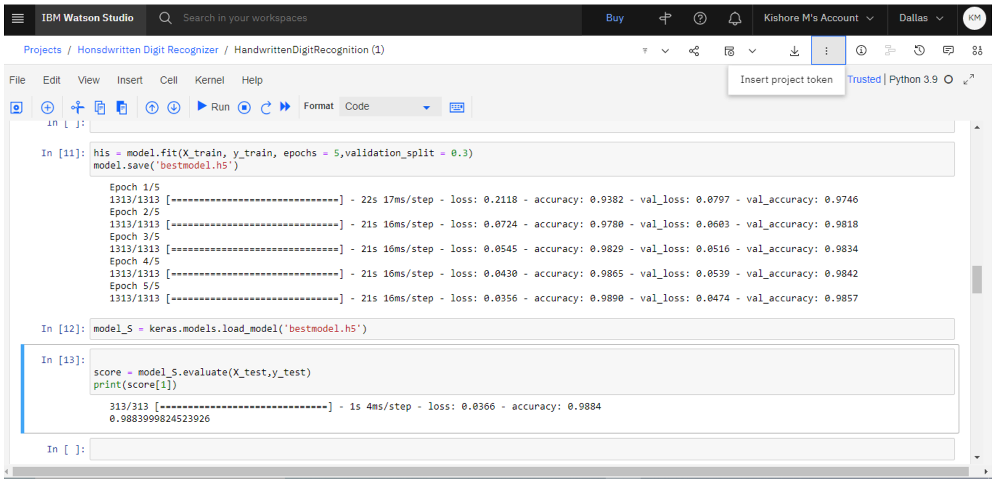
Projects / Honsdwritten Digit Recognizer / HandwrittenDigitRecognition (1)

```
In [4]: for i in range (2):
          plot_input(i)
```



Projects / Honsdwritten Digit Recognizer / HandwrittenDigitRecognition (1)

```
In [8]: model.summary()

          Model: "sequential"
          _________________________________________________________________
           Layer (type)                Output Shape              Param #
          =================================================================
           conv2d (Conv2D)             (None, 26, 26, 32)        320

           max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
           )

           conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

           max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
           2D)

           flatten (Flatten)           (None, 1600)              0

           dropout (Dropout)           (None, 1600)              0

           dense (Dense)               (None, 10)                16010

          =================================================================
          Total params: 34,826
          Trainable params: 34,826
          Non-trainable params: 0
          _________________________________________________________________
```



```
In [11]: his = model.fit(X_train, y_train, epochs = 5,validation_split = 0.3)
         model.save('bestmodel.h5')

           Epoch 1/5
           1313/1313 [==============================] - 22s 17ms/step - loss: 0.2118 - accuracy: 0.9382 - val_loss: 0.0797 - val_accuracy: 0.9746
           Epoch 2/5
           1313/1313 [==============================] - 21s 16ms/step - loss: 0.0724 - accuracy: 0.9780 - val_loss: 0.0603 - val_accuracy: 0.9818
           Epoch 3/5
           1313/1313 [==============================] - 21s 16ms/step - loss: 0.0545 - accuracy: 0.9829 - val_loss: 0.0516 - val_accuracy: 0.9834
           Epoch 4/5
           1313/1313 [==============================] - 21s 16ms/step - loss: 0.0430 - accuracy: 0.9865 - val_loss: 0.0539 - val_accuracy: 0.9842
           Epoch 5/5
           1313/1313 [==============================] - 21s 16ms/step - loss: 0.0356 - accuracy: 0.9890 - val_loss: 0.0474 - val_accuracy: 0.9857

In [12]: model_S = keras.models.load_model('bestmodel.h5')

In [13]:
         score = model_S.evaluate(X_test,y_test)
         print(score[1])

           313/313 [==============================] - 1s 4ms/step - loss: 0.0366 - accuracy: 0.9884
           0.9883999824523926

In [ ]:
```

8. TESTING

8.1. Testcases

The test cases are available in the following link:
https://docs.google.com/spreadsheets/d/1uau-KFBat_XIWZ8nqxaUsPZcU1_hXhvX/edit#gid=1760590381

8.2. User Acceptance Testing

The test cases are available at the following link:
https://github.com/IBM-EPBL/IBM-Project-4236-1658725558/blob/main/Project%20Development%20Phase/Performance%20Testing/UAT%20Report%20Template%20-%20KNPKK.docx.pdf

9. RESULTS

The above project is executed and output is obtained and verified with the expected outcomes successfully.

10. ADVANTAGES AND DISADVANTAGES

Advantages:

This strategy offers a number of benefits:

1) In addition to classifying the digits, the algorithm also generates a thorough description of the instantiation parameters, which might reveal details like the writing style.

2) The generative models are capable of segmentation driven by recognition.

3) The approach only requires a limited number of parameters, making training simple and quick.

4) It can withstand variable scaling, translation, and a little amount of picture rotation, unlike many other recognition techniques that rely on some kind of pre-normalization of input images.

5) Offline handwriting recognition has a wide range of uses, including reading postal addresses, bank check amounts, and forms. Additionally, OCR is crucial for digital libraries because it enables the entry of image textual data into computers using methods for digitization, picture restoration, and recognition.

Disadvantages:

1) The drawback is that it cannot be used for immediate text input because it is not done in real-time as a person writes.
2) The four processes of offline handwriting systems are acquisition, segmentation, recognition, and postprocessing. The handwriting that needs to be read is first digitized using scanners or cameras. The document's picture is divided into lines, words, and individual characters in the second step. Third, OCR methods are used to identify each character. Lexicon or spell checkers are then used to fix mistakes.
3) Because only geographical information is accessible for offline systems, but both spatial and temporal information is available for online systems, offline handwriting recognition systems are less accurate than online systems.

11. **CONCLUSION**

Using MNIST datasets, research on handwritten digit recognition uses deep and machine learning methods. To determine which model was the most accurate, we compared them based on their individual properties. In our research, CNN consistently produces the most precise outcomes for handwritten digit recognition. This leads us to the conclusion that CNN is the most effective solution for any prediction issue that uses picture data as input. Next, by comparing the algorithms' execution times, we have come to the conclusion that, due to a particular model's limitation, increasing the number of epochs without changing the configuration of the algorithm is useless. We have also noticed that, after a certain number of epochs, the model starts overfitting the dataset and giving us biased results.

12. **FUTURE SCOPE**

The uniqueness of the current study is that it carefully examines each CNN architectural parameter to determine which one provides the greatest recognition accuracy for the MNIST dataset. A pure CNN model could not equal this accuracy according to peer researchers. Some of the researchers made use of an ensemble CNN network. architectures for the same dataset that would boost recognition accuracy while incurring higher processing costs and testing complexity while maintaining accuracy equivalent to that of the current study. Future CNN architectures will be accessible, including hybrid CNN, CNN-RNN, and CNN-HMM. It is possible to look at models and domain-specific recognition systems. In order to optimize the CNN learning parameters, especially the number of layers, learning rate, and Kernel sizes, evolutionary techniques might be investigated. In the future, we intend to experiment with the model's other parameters as well as the number of hidden layers to see how the classification accuracy varies overall. Right now, we're using an image data augmentation method that can handle photos that have been rotated by 10 degrees. In order to enable our model to detect the rotated 6 and 9 pictures, we intend to further improve it in this direction (s). Since colored pictures may have different border conditions, we also want to improve accuracy outcomes for colored images. We'll also try to apply our models to databases with letters and double digits so that we may broaden and diversify our application.

13. **APPENDIX**

SUBSET OF MACHINE LEARNING

Deep Learning is a kind of machine learning that uses artificial neural networks and many processing layers to extract more complex characteristics from the input. The CNN model has been widely used in recent years to recognize handwritten digits from the MNIST benchmark collection. In terms of handwritten digit recognition accuracy, some researchers have claimed values as high as 98% or 99%. The table demonstrates the differences between shadow neural networks and deep neural networks.

Table 1

Shallow neural network vs deep neural network.

Factors	**Shallow Neural Network (SNN)**	**Deep Neural Network (DNN)**
Number of hidden layers	- single hidden layer (need to be fully connected).	- multiple hidden layers (not necessarily fully connected).
Feature Engineering	- requires a separate feature extraction process. - some of the famous features used in the literature including local binary patterns (LBPs), histogram of oriented gradients (HOGs), speeded-up robust features (SURFs), and scale-invariant feature transform (SIFT).	- supersede the handcrafted features and works on the whole image. - useful in computing complex pattern recognition problems. - can capture complexities inherent in the data.

Requirements	- emphasizes the quality of features and their extraction process. - networks are more dependent on the expert skills of researchers.	- able to automatically detect the important features of an object (here an object can be an image, a handwritten character, a face, etc.) without any human supervision or intervention.
Dependency on data volume	- requires a small amount of data.	- requires a large amount of data.

CNN ARCHITECTURE METHODS

Using the same MNIST benchmark dataset, it can be seen that our CNN model performs better than a number of other CNN models that have been developed by other academics. For the same dataset, several researchers employed ensemble CNN architectures to raise recognition accuracy, albeit at the expense of higher processing costs and testing complexity. Even without using ensemble architecture, the suggested CNN model obtained a recognition accuracy of 99.89% for the MNIST dataset.

Table 2

Comparison of proposed CNN architecture for numeral recognition with other techniques.

Handwritten Numeral Recognition

Reference	Approach	Database	Features	Accuracy (%)/Error Rate
[75]	CNN	MNIST	Pixel based	0.23%
[76]	CNN	MNIST	Pixel based	0.19%
[8]	CNN	MNIST	Pixel based	0.53%
[77]	CNN	MNIST	Pixel based	0.21%
[78]	CNN	MNIST	Pixel based	0.17%
[79]	Deep Learning	The Chars74K	Pixel based	88.89% (GoogleNet) 77.77% (Alexnet)
[43]	CNN	Urdu Nasta'liq handwritten dataset (UNHD)	Pixel and geometrical based	98.3%
Proposed approach	CNN	MNIST	Pixel and geometrical based	99.89%

Source Code:

https://github.com/IBM-EPBL/IBM-Project-4236-1658725558/tree/main/Final%20Deliverables

GitHub & Project Demo Link:

https://github.com/IBM-EPBL/IBM-Project-4236-1658725558

https://drive.google.com/file/d/1bg5XMqB5wf6Mhha9OTPkCU7bu6u_FXlX/view?usp=sharing