

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. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI.

1.2.PURPOSE

Handwritten character recognition is one of the practically important issues in pattern recognition applications. The applications of digit recognition include in postal mail sorting, bank check processing, form data entry, etc.

An enormous number of CNN classification algorithms have been proposed in the literature. Nevertheless, in these algorithms, appropriate filter size selection, data preparation, limitations in datasets, and noise have not been taken into consideration. As a consequence, most of the algorithms have failed to make a noticeable improvement in classification accuracy. To address the shortcomings of these algorithms, our paper presents the following contributions: Firstly, after taking the domain knowledge into consideration, the size of the effective receptive field (ERF) is calculated. Calculating the size of the ERF helps us to select a typical filter size which leads to enhancing the classification accuracy of our CNN. Secondly,

unnecessary data leads to misleading results and this, in turn, negatively affects classification accuracy. To guarantee the dataset is free from any redundant or irrelevant variables to the target variable, data preparation is applied before implementing the data classification mission.

2.LITERATURE SURVERY

2.1.EXISTING PROBLEM

[1] Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc, and classify them into 10 predefined classes (0-9). The models with low accuracy are not suitable for real-world applications. Methods like SVM, MLP, CNN and KNN are some machine learning and deep learning algorithms. In SVM, all the features are plotted and classification is performed using hyperplanes that separate the classes correctly. The MLP algorithm uses input layer, hidden layer and output layer. Each layer consists of several nodes that are also formally referred to as neurons and each node is interconnected to every other node of the next layer. The number of hidden layers may increase according to the problem with no restrictions to the number of nodes. The CNN is widely used for image processing where input image is given in small chunks rather than pixels at a time, to detect uncertain patterns more efficiently. CNN contains 3 layers namely, an input layer, an output layer, and multiple hidden layers which include Convolutional layers, Pooling layers (Max and Average pooling), Fully connected layers (FC), and normalization layers. After implementing all the three algorithms that are SVM, MLP and CNN, it is found that SVM has the highest accuracy while training data and CNN has the utmost accuracy while testing the data. SVM took the minimum time for execution while CNN took the maximum running time. Handwritten digit recognition has recently been of much interest among the researchers because of the evolution of various Machine Learning, Deep Learning and Computer Vision algorithms. They compare the results of some of the most widely used Machine Learning Algorithms like CNN-convolution neural networks and with Deep Learning algorithm like multilayer CNN using Keras with Theano and Tensorflow. 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 network can almost mimic the human brain and are a key ingredient in image processing field. For example, Convolution Neural network with back propagation for image processing. The applications where handwritten digit recognition can be used is banking sector where it can be used to maintain the security pin numbers, it can be also used for blind peoples by using sound output.

[2] Md. Anwar Hossain and Md. Mohan Ali state that CNN is an important tool when it comes to learning deep learning with a NN. Convolutional Neural Networks are a special kind of multi-layer neural networks designed to recognize visual patterns directly from pixel images with minimal pre-processing. Convolution is filtering the image with a smaller pixel filter to decrease the size of the image without losing the relationship between pixels. Max Pooling is one of the most common pooling techniques. A fully connected network is in any architecture where each parameter is linked to one another to determine the relation and effect of each parameter on the labels. All these convolution layers, pooling layers and fully connected layers are stacked. The pre-processed data is fed to the model. CNN transforms the original image layer by layer from the original pixel values to the final class scores.

[3] Alexander K. Seewald talks about a general weakness in present intelligent image analysis systems, calling it ‘brittleness’ in AI terminology. He has investigated two new aspects of such systems, they are essential training set size i.e., the relation between training set size and accuracy/error rate so as to determine the number of labelled training samples that are essential for a given performance level and dataset-independence i.e., how well models trained on one sample dataset for

handwritten digit recognition perform on other sample datasets for handwritten digit recognition after comprehensive normalization between the datasets. He says that small differences in the pre-processing methods which have not been documented in sufficient detail may be responsible for this effect. Another explanation might be that idiosyncrasies of the specific dataset used for training are learned as well and hamper the generalization ability of the underlying learning algorithm. This effect is observed independently of learning algorithm or feature representation.

2.2.REFERENCE

[1] Handwritten Digit Recognition using Machine and Deep Learning Algorithms by Ritik Dixit of Computer Science and Engineering, Rishika Kushwah of Computer Science and Engineering, Samay Pashine of Computer Science and Engineering, Acropolis Institute of Technology & Research, Indore, India, 23 June 2021

[2] Recognition of Handwritten Digit using Convolutional Neural Network (CNN), By Md. Anwar Hossain & Md. Mohon Ali, Pabna University of Science & Technology, 2019

[3] On the Brittleness of Handwritten Digit Recognition Models, Alexander K. Seewald, 30 Nov 2011

2.3.PROBLEM STATEMENT DEFINITION

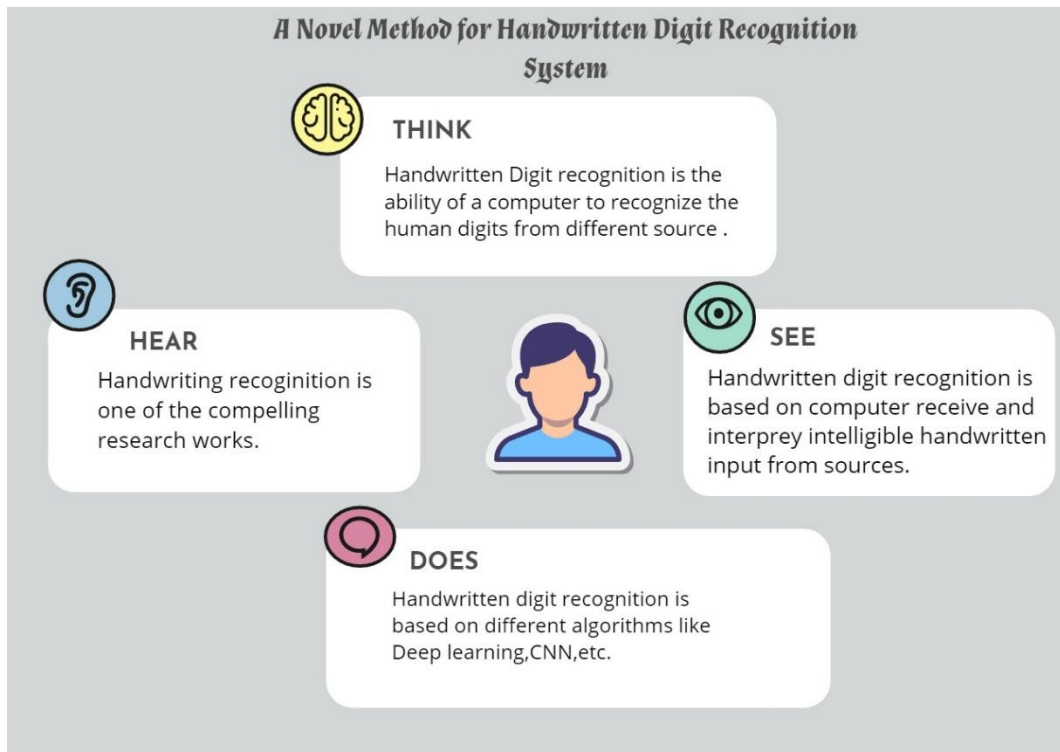
HANDWRITTEN digit recognition is the ability of a computer system to recognize the handwritten inputs like digits, characters etc. from a wide variety of sources like emails, papers, images, letters etc. This has been a topic of research for decades. Some of the research areas include signature verification, bank check processing, postal address interpretation from envelopes etc. Here comes the use of Deep Learning. In the past decade, deep learning has become the hot tool for Image Processing, object detection, handwritten digit and character recognition etc. A lot

of machine learning tools have been developed like scikit-learn, scipy-image etc. and pybrains, Keras, Theano, Tensorflow by Google, TFLearn etc. for Deep Learning. These tools make the applications robust and therefore more accurate. The Artificial Neural Networks can almost mimic the human brain and are a key ingredient in image processing field. For example, Convolutional Neural Networks with Back Propagation for Image Processing, Deep Mind by Google for creating Art by learning from existing artist styles etc.. Handwriting Recognition has an active community of academics studying it. The biggest conferences for handwriting recognition are the International Conference on Frontiers in Handwriting Recognition (ICFHR), held in even-numbered years, and the International Conference on Document Analysis and Recognition (ICDAR), held in odd-numbered years. Both of these conferences are endorsed by the IEEE. Active areas of research include: Online Recognition, OfflineRecognition, Signature Verification, Postal-Address Interpretation, Bank-Check Processing, Writer Recognition. Classification of images and patterns has been one of the major implementation of Machine Learning and Artificial Intelligence. People are continuously trying to make computers intelligent so that they can do almost all the work done by humans Handwriting recognition system is the most basic and an important step towards this huge and interesting area of Computer Vision.

3.IDEALTION AND PROPOSED SOLUTION

3.1.EMPATHY MAP CANVAS

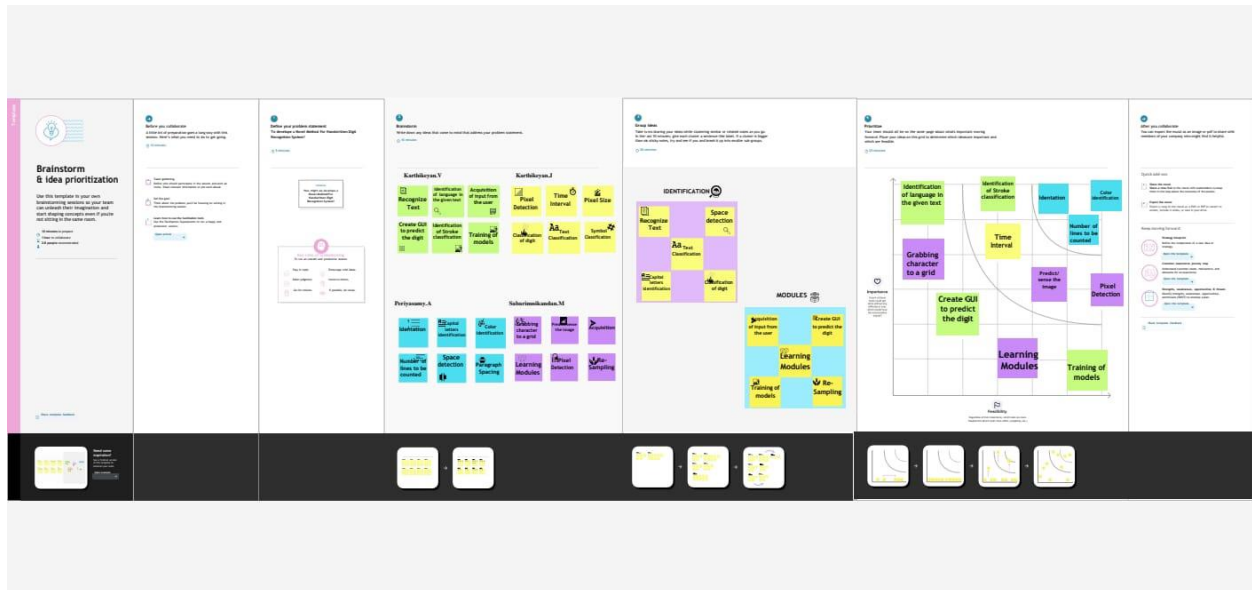
An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.



3.2.IDEATION AND BRAINSTROMING

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and

brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



3.3.PROPOSED SOLUTION

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 Bootstrap 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.

3.4.PROBLEM SOLUTION FIT

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.

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 .

Approach:

- **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.

4.REQUIREDMENT ANALYSIS

4.1.FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	The product essentially converts handwritten digits to digital form.	The user is first asked to draw a number on the canvas, and the model that is built is then utilised to compare the data and provide an output in digitalized form.
FR-2	Recognizing the handwritten digit and displaying.	Recognizing the handwritten digit and displaying.
FR-3	Import dataset file directly to the program from a command that will download the dataset from its website. Save the dataset file in the same directory as the program	Installing packages and applications.
FR-4	Build a Neural Network with a number of nodes in the input layer equal to the number of pixels in the arrays	Nil
FR-5	Activating the Neural Network	Packages – tensorflow

4.2.NON-FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	System design should be easily understood and user friendly to users. Furthermore, users of all skill levels of users should be able to navigate it without problems.
NFR-2	Security	The system should automatically be able to authenticate all users with their unique username and password
NFR-3	Performance	Should reduce the delay in information when hundreds of requests are given.
NFR-4	Availability	Information is restricted to each users limited access

5.PROJECT DESIGN

5.1.DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

DFD Level-0

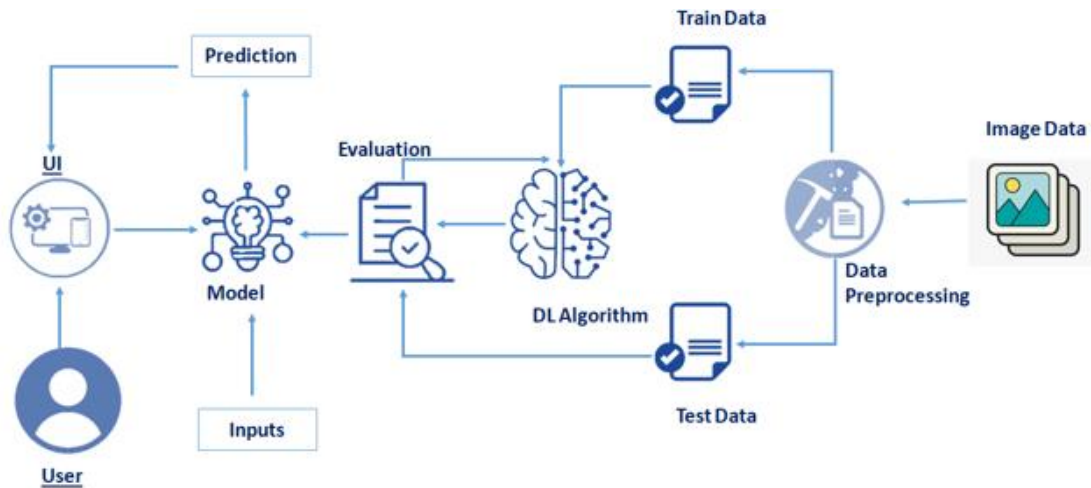
The DFD Level-0 consists of two external entities, the UI and the Output, along with a process, representing the CNN for Digit Recognition .Output is obtained after processing.

DFD Level-1

The DFD Level-1 consists of 2 external entities, the GUI and the Output, along with five process blocks and 2 data stores MNIST data and the Input image store, representing the internal workings of the CNN for Digit Recognition System. Process block imports MNIST data from library. Process block imports the image and process it and sends it to block where regression model is built. It sends objects with probabilities to CNN where weights are updated and multiple layers are built. Block trains and evaluates the model to generate output.

DFD Level-2

The DFD Level-2 for import data (figure 4) consists of two external data and one entity UI along with three process blocks, representing the three functionalities of the CNN for Digit Recognition System. It imports data from MNIST data store and stores on the system.



5.2.SOLUTION AND TECHNICAL ARCHITECTURE

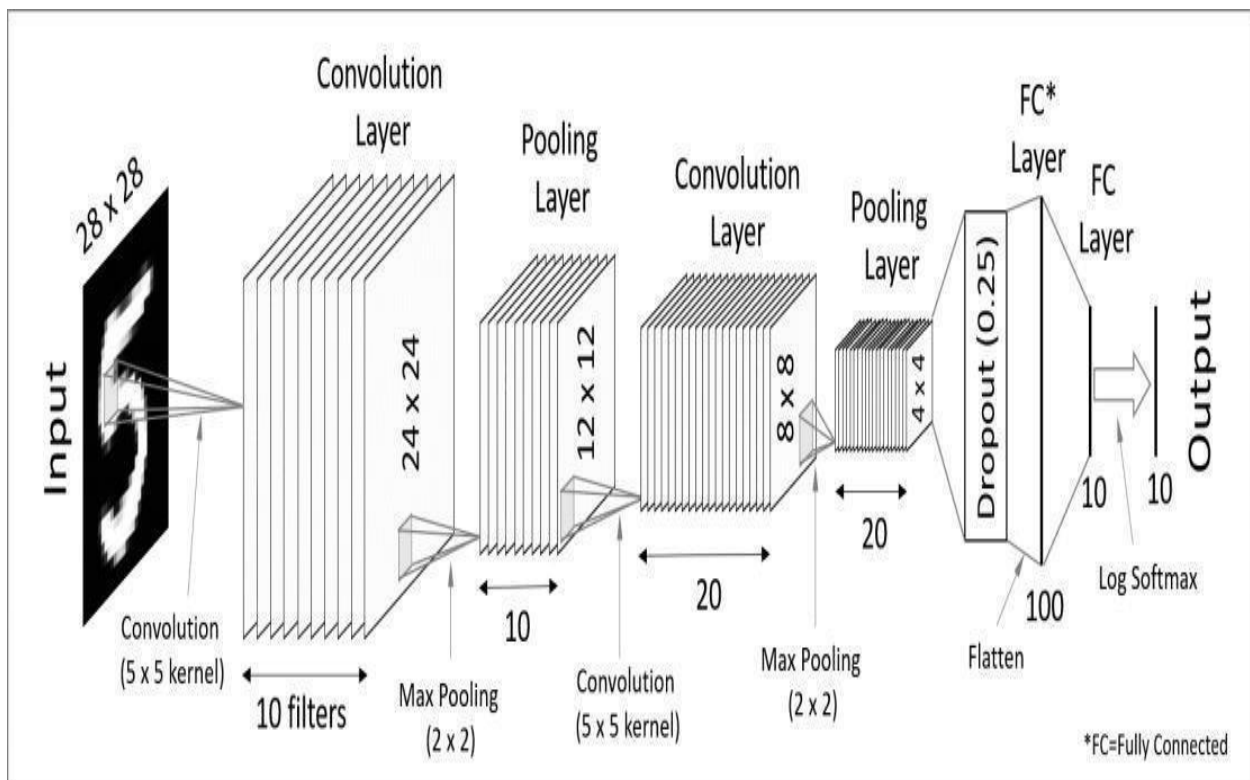
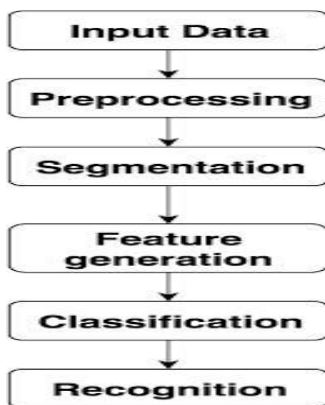


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant
7.	File Storage	File storage requirements	IBM Block Storage
8.	External API-1	Purpose of External API used in the application	IBM Weather API
9.	External API-2	Purpose of External API used in the application	Aadhar API
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access control implemented, use of firewalls etc.	SHA-256, Encryptions, IAM Controls, OWASP

3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	3 – tier, Micro-services
----	-----------------------	--	--------------------------

S.No	Characteristics	Description	Technology
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Distributed servers, IBM cloud
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Number of requests per sec, use of Cache, use of CDN's

5.3.USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can	I can access	High	Sprint-1

(Mobile user)			register for the application by entering my email, password, and confirming	my account / dashboard		
---------------	--	--	---	------------------------	--	--

			my password.			
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-2
		USN-3	As a user, I can register for the application through gmail or facebook	I can register & access the dashboard with Facebook Login	Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password	I can login to the application	High	Sprint-1
	Dashboard	USN-5	Go to dashboard and refer the content about our project	I can read instructions also and the home page is user-friendly.	Low	Sprint-1
	Upload Image	USN-6	As a user, I can able to input the images of digital documents	As a user, I can able to input the images of digital	High	Sprint-3

			to the application	documents to the application		
	Predict	USN-7	As a user I can able to get the recognised digit as output from the images of digital documents or images	I can access the recognized digits from digital document or images	High	Sprint-3
		USN-8	As a user, I will train and test the input to get the maximum accuracy of	I can able to train and test the application until it gets maximum accuracy of	Medium	Sprint-4
			output.	the result.		
Customer (Web user)	Login	USN-9	As a user, I can use the application by entering my email, password.	I can access my account	Medium	Sprint-4
Customer Care Executive	Dashboard	USN-10	upload the image	Recognize and get the output	High	Sprint-1
Administrator	Security	USN-11	updated the features	checking the security	Medium	Sprint-1

6.PROJECT PLANNING & SCHEDULING

6.1.SPRINT PLANNING & ESTIMATION

TITLE	DESCRIPTION	DATE
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	24 SEPTEMBER 2022
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	28 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	25 SEPTEMBER 2022
Proposed Solution	Creation of proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	23 SEPTEMBER 2022
Problem Solution Fit	Creation of problem solution fit document.	30 SEPTEMBER 2022
Solution Architecture	Creation of solution architecture document.	28 SEPTEMBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application.	20 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	20 OCTOBER 2022

Technology Architecture	Prepare the technology architecture diagram.	22 OCTOBER 2022
-------------------------	--	-----------------

6.2.SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement	Task
Sprint-1	Image Data	As a User need to collect the Image Data of Handly Written Images to train the model.
Sprint-2	Dash Board or Website	We using Python Flask Framework to create a dynamic Webpage to host our model (UI).
Sprint-3	Classifier Model	Using CNN Model for Image Classification.
Sprint-4	Cloud	Hosting the Organized appication in Cloud platform.

7.CODING AND SOLUTION

7.1.FEATURE 1

Importing the required libraries

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
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 r
#flatten -used for 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
import matplotlib.pyplot as plt #used for data visualization
```

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)
```

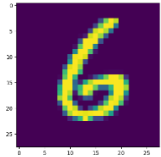
```
(10000, 28, 28)
```

```
x_train[0]
```

```
array([[ 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,  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, 0, 14, 1, 154, 253, 90, 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, 139, 253, 190, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11, 190, 253, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 35, 241, 225, 160, 108, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 0, 0, 0, 0, 55, 172, 226, 253, 253, 253, 253, 244, 133, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0],
[ 0, 0, 0, 0, 136, 253, 253, 253, 212, 135, 132, 16, 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]], dtype=uint8)
```

```
plt.imshow(x_train[6000]) #ploting the index=image
```



```
np.argmax(y_train[6000])
```

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')
```

Applying One Hot Encoding

number_of_classes = 10 #storing the no of 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)
```

7.2.FEATURE 2

Importing the required libraries

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
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
computat ion funct ion
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the
regular deeply connected r
#faltten -used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #onvoLutiona l Layer
from keras.optimizers import Adam #opt imizer
from keras. utils import np_utils #used for one-hot encoding
import matplotlib.pyplot as plt #used for data visualization
```

```
(x_train, y_train), (x_test, y_test)=mnist.load_data ()
x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')
x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')
number_of_classes = 10 #storing the no of 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)
```

Add CNN Layers

```
#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'))
```

#flatten the dimension of the image

```
model.add(Flatten())
```

#output layer with 10 neurons

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

Compiling the model

```
#Compile model
```

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

```
x_train = np.asarray(x_train)
```

```
y_train = np.asarray(y_train)
```

Train 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 [=====] - 120s 63ms/step - loss: 0.2  
808 - accuracy: 0.9487 - val_loss: 0.1263 - val_accuracy: 0.9649
```

Epoch 2/5

```
1875/1875 [=====] - 95s 50ms/step - loss: 0.07  
34 - accuracy: 0.9780 - val_loss: 0.0947 - val_accuracy: 0.9733
```

Epoch 3/5

```
1875/1875 [=====] - 88s 47ms/step - loss: 0.05  
24 - accuracy: 0.9839 - val_loss: 0.1133 - val_accuracy: 0.9701
```

Epoch 4/5

```
1875/1875 [=====] - 97s 52ms/step - loss: 0.03  
75 - accuracy: 0.9884 - val_loss: 0.1308 - val_accuracy: 0.9720
```

Epoch 5/5

```
1875/1875 [=====] - 96s 51ms/step - loss: 0.02  
75 - accuracy: 0.9912 - val_loss: 0.1233 - val_accuracy: 0.9781
```

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) :
```

```
[0.12333168834447861, 0.9781000018119812]
```

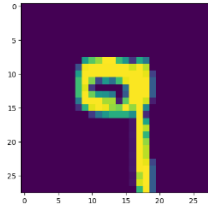


```

prediction=model.predict(x_test[6000:6001])
print(prediction)
1/1 [=====] - 0s 93ms/step
[[1.1050688e-15 9.2188809e-14 5.0320639e-15 5.0875104e-10 2.6267043e-01
  1.0428948e-08 5.2634429e-17 2.9271017e-09 6.4425700e-07 7.3732901e-01]]

```

```
plt.imshow(x_test[6000])
```



```

import numpy as np
print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images
[9]
np.argmax(y_test[6000:6001]) #printing the actual labels
9

```

Save The model

```
# Save the model
```

```
model.save('models/mnistCNN.h5')
```

8.TESTING

8.1.TEST CASE

Implementing to test case:

- User Acceptance Testing
- Performance Testing

8.2.USER ACCEPTANCE TESTING

User acceptance testing (UAT), also called *application testing* or *end-user testing*, is a phase of software development in which the software is tested in the real world by its intended audience. UAT is often the last phase of the software testing process and is performed before the tested software is released to its intended market. The goal of UAT is to ensure software can handle real-world tasks and perform up to development specifications.

In UAT, users are given the opportunity to interact with the software before its official release to see if any features have been overlooked or if it contains any bugs. UAT can be done in-house with volunteers, by paid test subjects using the software or by making the test version available for download as a free trial. The results from the early testers are forwarded to the developers, who make final changes before releasing the software commercially.

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9

Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULT

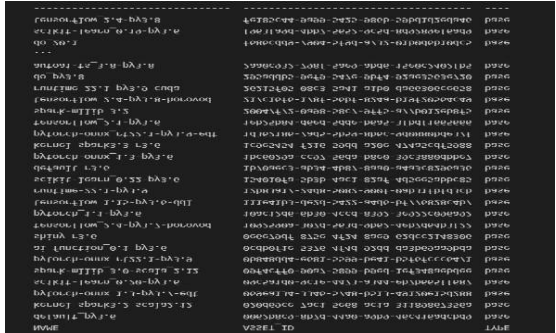
9.1.PERFORMANCE METRICS

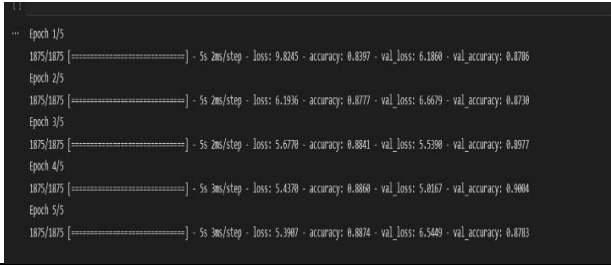
Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

1. Identify the Test Environment and Tools. Identify the production environment, testing environment, and testing tools at your disposal. ...
2. Define Acceptable Performance Criteria. ...
3. Plan and Design Tests. ...
4. Prepare Test Environment and Tools. ...
5. Run the Performance Tests. ...
6. Resolve and Retest.

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary		

2.	Accuracy	Training Accuracy – 88.74 Validation Accuracy – 87.83	 <pre>-- Epoch 1/5 1875/1875 [=====] - 5s 2ms/step - loss: 9.8245 - accuracy: 0.8397 - val_loss: 6.1860 - val_accuracy: 0.8786 Epoch 2/5 1875/1875 [=====] - 5s 2ms/step - loss: 6.1936 - accuracy: 0.8777 - val_loss: 6.6679 - val_accuracy: 0.8738 Epoch 3/5 1875/1875 [=====] - 5s 2ms/step - loss: 5.6770 - accuracy: 0.8841 - val_loss: 5.5390 - val_accuracy: 0.8977 Epoch 4/5 1875/1875 [=====] - 5s 3ms/step - loss: 5.4370 - accuracy: 0.8868 - val_loss: 5.8167 - val_accuracy: 0.9004 Epoch 5/5 1875/1875 [=====] - 5s 3ms/step - loss: 5.3907 - accuracy: 0.8874 - val_loss: 6.5449 - val_accuracy: 0.8783</pre>
----	----------	--	--

10.ADVANTAGES & DISADVANTAGES

Advantages

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

Disadvantages

- It is not done in real time as a person writes and therefore not appropriate for immediate text input.
- Applications of offline handwriting recognition are numerous: reading postal addresses, bank check amounts, and forms.

11.CONCLUSION

Handwritten digit recognition has immense applications in the field of medical, banking, student management, and taxation process etc. Many classifiers like KNN, SVM, CNN are used to identify the digit from the handwritten image. as per the review, CNN is providing better performance than others. Stages of HDR using CNN classifier is discussed in this paper. MNIST dataset consist of handwritten numbers from 0-9 and it is a standard dataset used to find performance of classifiers. HDR consists of three different stages. First is preprocessing where dataset is converted into binary form and image processing has been applied on it. Second stage is segmentation where the image is converted into multiple segments. Third stage is feature extraction where features of image are identified. Last stage is classification where classifiers like KNN, SVM, CNN are used. Results of HDR is improved a lot by using CNN classifier but it can be improved further in terms of complexity, duration of execution and accuracy of results by making combination of classifiers or using some additional algorithm with it.

12.FUTURE SCOPE

The task of handwritten digit recognition, using a classifier, has great importance and use such as online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example tax forms) and so on.

13.APPENDIX

Source code

```
import numpy as np
import os
from PIL import Image
from flask import Flask, request, render_template, url_for
from werkzeug.utils import secure_filename, redirect
from gevent.pywsgi import WSGIServer
from keras.models import load_model
from keras.preprocessing import image
from flask import send_from_directory
UPLOAD_FOLDER = 'C:\Users\rajes\Documents\flask_app\uploads'
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
model = load_model("mnistCNN.h5")
@app.route('/index',methods=['GET','POST'])
def index():
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == "POST":
        f = request.files["image"]
        filepath = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))
        upload_img = os.path.join(UPLOAD_FOLDER, filepath)
```

```

    img = Image.open(upload_img).convert("L") # convert image to
monochrome
    img = img.resize((28, 28)) # resizing of input image
    im2arr = np.array(img) # converting to image
    im2arr = im2arr.reshape(1, 28, 28, 1) # reshaping according to our
requirement
    pred = model.predict(im2arr)
    num = np.argmax(pred, axis=1) # printing our Label
    return render_template('predict.html', num=str(num[0]))
if __name__ == '__main__':
    app.run(debug=True, threaded=False)

```

GitHub Link:

<https://github.com/IBM-EPBL/IBM-Project-6918-1658843098>

Demo Video Link:

<https://drive.google.com/file/d/1Vlnr3zSoz0OXw8vSK3BPfK6cKiw3s6aM/view?usp=sharing>