A Mini Project Report

on

Handwritten Digit Recognition Using Deep Learning

In

Mini Project Lab

Bachelor of Technology

Department of Information Technology

by

Ashish Kumar Sharma

Roll. No. 1809713025

Ekansh Mishra

Roll. No. 1809713043

Mansi Bhalotia

Roll. No. 1809713058

Under the Supervision of

Dr. Sandhya Katiyar

Galgotias College of Engineering & Technology

Greater Noida 201306

Uttar Pradesh, INDIA

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University

Lucknow , Uttar Pradesh

ACKNOWLEDGEMENT

We want to give special thanks to our Mini Project coordinator Dr. Sandhya Katiyar for the timely advice and valuable guidance during designing and implementation of this mini project work.

We also want to express our sincere thanks and gratitude to Dr. Sanjeev Kumar Singh, Head of Department (HOD), Information Technology Department for providing us with the facilities and for all the encouragement and support.

Finally, We express our sincere thanks to all staff members in the department of Information Technology branch for all the support and cooperation.

Ekansh Mishra (1809713043)

Mansi Bhalotia (1809713058)

Ashish Kumar Sharma (1809713025)

ABSTRACT

The reliance of humans over machines has never been so high such that from object classification in photographs to adding sound to silent movies everything can be performed with the help of deep learning and machine learning algorithms. Likewise, Handwritten text recognition is one of the significant areas of research and development with a streaming number of possibilities that could be attained. Handwriting recognition (HWR), also known as Handwritten Text Recognition (HTR), is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. Apparently, in this paper, we have performed handwritten digit recognition with the help of MNIST datasets using Support Vector Machines (SVM), Multi-Layer Perceptron (MLP) and Convolution Neural Network (CNN) models. Our main objective is to compare the accuracy of the models stated above along with their execution time to get the best possible model for digit recognition. Keywords: Deep Learning, Machine Learning, Handwritten Digit Recognition, MNIST datasets, Support Vector Machines (SVM), Multi-Layered Perceptron (MLP), and Convolution Neural Network (CNN)

Content

1 INTRODUCTION

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). In the current age of digitization, handwriting recognition plays an important role in information processing. A lot of information is available on paper, and processing of digital files is cheaper than processing traditional paper files. The aim of a handwriting recognition system is to convert handwritten characters into machine readable formats. The main applications are vehicle license-plate recognition, postal letter-sorting services, Cheque truncation system (CTS) scanning and historical document preservation in archaeology departments, old documents automation in libraries and banks, etc. All these areas deal with large databases and hence demand high recognition accuracy, lesser computational complexity and consistent performance of the recognition system. It has been suggested that deep neural architectures are more advantageous than shallow neural architectures

This has been a topic of boundless-research in the field of deep learning. Digit recognition has many applications like number plate recognition, postal mail sorting, bank check processing, etc . In Handwritten digit recognition, we face many challenges because of different styles of writing of different peoples as it is not an Optical character recognition. The deep learning field is ever evolving, and some of its variants are autoencoders, CNNs, recurrent neural networks (RNNs), recursive neural networks, deep belief networks and deep Boltzmann machines. Here, we introduce a convolutional neural network, which is a specific type of deep neural network having wide applications in image classification, object recognition, recommendation systems, signal processing, natural language processing, computer vision, and face recognition. The ability 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 makes them (CNNs) more efficient

than their predecessors (Multi layer perceptron (MLP), etc.). The high capability of hierarchical feature learning results in a highly efficient CNN.

In this project, we are going to implement a handwritten digit recognition app using the MNIST dataset. We will be using a special type of deep neural network that is Convolutional Neural Networks. In the end, we are going to build a GUI in which you can draw the digit and recognize it straight away.

2 FEATURES AND DESIGN

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

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

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

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

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

2.1 Module Integration

1. DataSet Collection
2. Create And Train CNN Model
3. Create Python Application Using Pygames
4. Load CNN Model
5. Make Predictions

2.2 Technology Stack

Tensorflow

It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

Numpy, Pandas, Matplotlib

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.Keras is the high-level API of TensorFlow 2: an approachable,

highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

2.3 Dataset (MNIST)

It stands for Modified National Institute of Standards and Technology
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples.It is a subset of a larger set available from NIST.The digits have been size-normalized and centered in a fixed-size image.

2.4 System Design

In the model, Input is given as image of size 28*28*1 but the images in the dataset is of 3 channel so it has to be channelized to 1 colour channel. After pre-processing the input is then passed to the CNN layer of kernel size 3*3 and kernal is of 32 filters after passing to a rectified linear unit of activation layer and max pooling layer. The same convolution layer is repeated again with same 3*3 kernal size and 64 filter layer. After second layer of the convolution, the feature map is flattened and then passed to a dense layer learn subtle details.

The output class result has been softmaxed and optimized using Adam optimizer. For the loss per epochs is calculated by cross-entropy loss function.

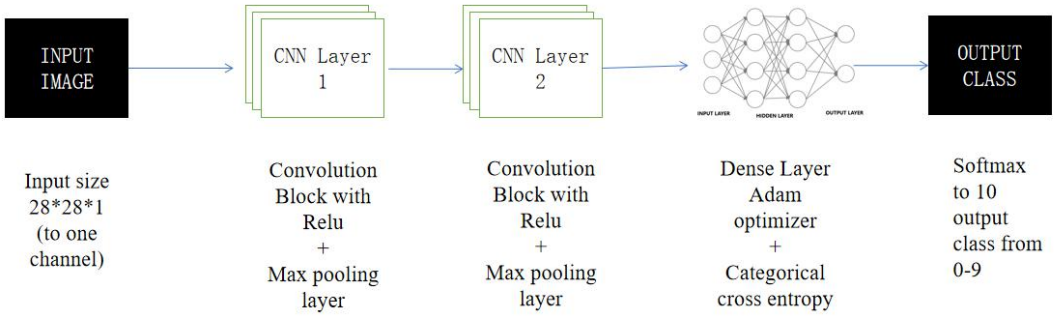

Fig- Block diagram of the CNN model

Fig- Loss-Epochs for the CNN model

3 Implemenation

```
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.layers import Dense,Conv2D,MaxPool2D,Flatten,Dropout
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train.shape, y_train.shape,x_test.shape, y_test.shape
def plot_input_img(i):
    plt.imshow(x_train[i],cmap='binary')
    plt.title(y_train[i])
    plt.show()
 for i in range(10):
    plot_input_img(i)
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)
from keras.utils import np_utils
y_train = keras.utils.np_utils.to_categorical(y_train)
y_test = keras.utils.np_utils.to_categorical(y_test)
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(0.25))
model.add(Dense(10,activation = "softmax"))
model.summary()
```

```
model.compile(optimizer='adam',loss = keras.losses.categorical_crossentropy ,
metrics =['accuracy'])
from keras.callbacks import EarlyStopping,ModelCheckpoint
es=EarlyStopping(monitor='val_acc',min_delta = 0.01,patience=4,verbose=1)
mc=ModelCheckpoint("./bestmodel.h5",monitor="val_acc",verbose =1,
save_best_only=True)
cb = [es,mc]
his = model.fit(x_train,y_train, epochs=5,validation_split = 0.3 , callbacks=cb)
model.save('C:/Users/ekans/otherMLmodels/hand new/bestmodel.h5')
model_S = keras.models.load_model('C:/Users/ekans/otherMLmodels/hand
new/bestmodel.h5')
#evaluate
score = model_S.evaluate(x_test,y_test)
print(f" the model accuracy is{score[1]}")
import matplotlib.pyplot as plt



his.history.keys()
plt.style.use('fivethirtyeight')
epochs = [i for i in range(5)]
fig , ax = plt.subplots(1,2)
train_acc = his.history['accuracy']
train_loss = his.history['loss']
test_acc = his.history['val_accuracy']
test_loss = his.history['val_loss']

fig.set_size_inches(20,6)
ax[0].plot(epochs , train_loss , label = 'Training Loss')
ax[0].plot(epochs , test_loss , label = 'Testing Loss')
ax[0].set_title('Training & Testing Loss')
ax[0].legend()
ax[0].set_xlabel("Epochs")

ax[1].plot(epochs , train_acc , label = 'Training Accuracy')
```

```
ax[1].plot(epochs , test_acc , label = 'Testing Accuracy')
ax[1].set_title('Training & Testing Accuracy')
ax[1].legend()
ax[1].set_xlabel("Epochs")
plt.show()
```

Python application

```
import pygame,sys
from pygame.locals import *
import numpy as np
from keras.models import load_model
import cv2

windowsizeX = 640
windowsizeY = 480
boundry=5
white = (255,255,255)
black = (0,0,0)
red = (255,0,0)

predict = True

image_cnt=1

Imagesave=False
Model = load_model("C:/Users/ekans/otherMLmodels/hand new/bestmodel.h5")

Label ={0:"ZERO",1:"ONE",
        2:"TWO",3:"THREE",
        4:"FOUR",5:"FIVE",
```

```
        6:"SIX",7:"SEVEN",
        8:"EIGHT",9:"NINE"}
pygame.init()
iswriting =False
number_xcord = []
number_ycord = []

Font = pygame.font.SysFont("Segoe UI",18)
DISPLAYSURF=pygame.display.set_mode((640,480))

pygame.display.set_caption("Digit Board")
while True:
    for event in pygame.event.get():
        if event.type ==QUIT:
            pygame.quit()
            sys.exit()
        if event.type== MOUSEMOTION and iswriting:
            xcord, ycord = event.pos
            pygame.draw.circle(DISPLAYSURF,white,(xcord,ycord), 4,0 )

            number_xcord.append(xcord)
            number_ycord.append(ycord)
        if event.type == MOUSEBUTTONDOWN:
            iswriting = True

        if event.type==MOUSEBUTTONUP:
            iswriting=False
            number_xcord = sorted(number_xcord)
            number_ycord = sorted(number_ycord)

            rect_min_x,rect_max_x  = max(number_xcord[0]-
boundry,0),min(windowsizeX,number_xcord[-1]+boundry)
            rect_min_y, rect_max_y = max(number_ycord[0] - boundry, 0),
min(number_ycord[-1] + boundry,windowsizeX)
```

```
        number_xcord =[]
        number_ycord = []

        img_arr =
np.array(pygame.PixelArray(DISPLAYSURF))[rect_min_x:rect_max_x,rect_min_y:r
ect_max_y].T.astype(np.float32)

        if Imagesave:
            cv2.imwrite("image.png")
            image_cnt+=1

        if predict:

            image = cv2.resize(img_arr,(28,28))
            image = np.pad(image,(10,10), 'constant', constant_values=0)
            image = cv2.resize(image,(28,28))/255

            label = str(Label[np.argmax(Model.predict(image.reshape(1,28,28,1)))])

            textSurface = Font.render(label,True,white,red)
            textRecObj = textSurface.get_rect()
            textRecObj.left , textRecObj.bottom = rect_min_x,rect_max_y

            DISPLAYSURF.blit(textSurface,textRecObj)

        if event.type == KEYDOWN:
            if event.unicode == "n":
                DISPLAYSURF.fill(black)

    pygame.display.update()
```

4 Result And Conclusion

In this work, with the aim of improving the performance of handwritten digit recognition, we evaluated a variant of convolutional neural network to avoid complex pre-processing, costly feature extraction and a complex ensemble (classifier combination) approach of a traditional recognition system. Through extensive evaluation using a MNIST dataset, the present work suggests the role of various hyper-parameters. We also verified that fine tuning of hyper-parameters is essential in improving the performance of CNN architecture.

We achieved a recognition rate of 99.08% with the Adam optimizer for the MNIST database. The effect of increasing the number of convolutional layers in CNN architecture on the performance of handwritten digit recognition is clearly presented through the experiment. The novelty of the present work is that it thoroughly investigates all the parameters of CNN architecture that deliver best recognition accuracy for a MNIST dataset.

In future, different architectures of CNN, namely, hybrid CNN, viz., CNN-RNN and CNN-HMM models, and domain-specific recognition systems, can be investigated. Evolutionary algorithms can be explored for optimizing CNN learning parameters, namely, the number of layers, learning rate and kernel sizes of convolutional filters.

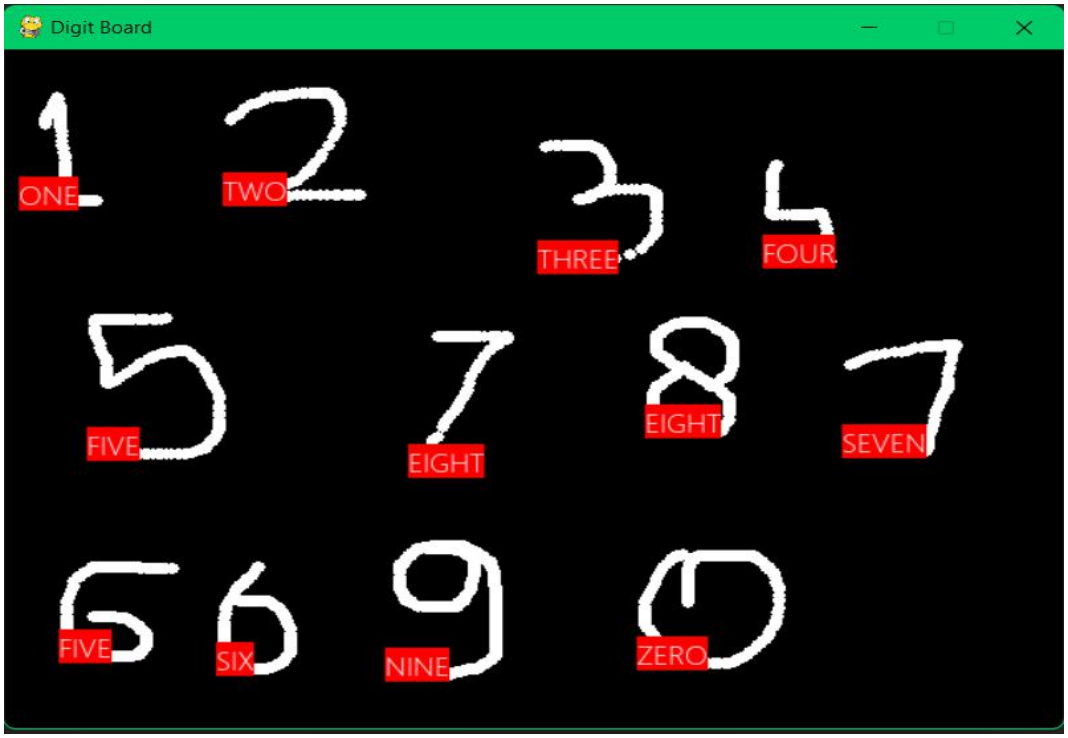

Fig (a) - Prediction made by the model

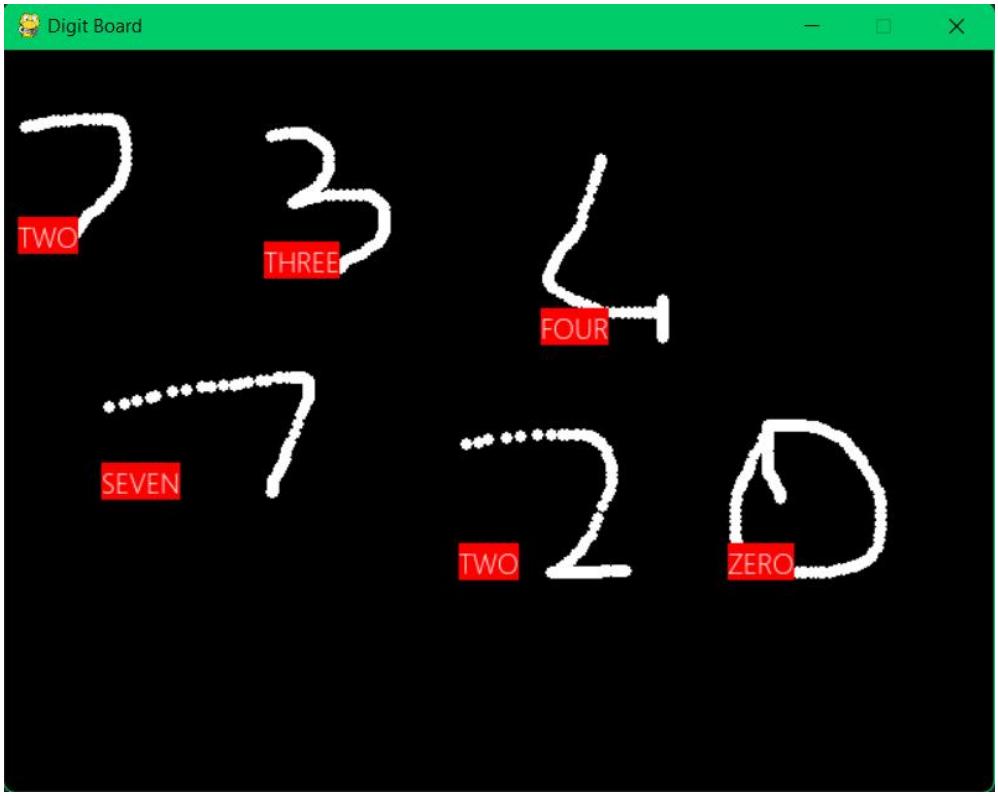

Fig (b) - Prediction made by the model

REFERENCES

[1] **Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.** [MNIST] "*Gradient-based learning applied to document recognition.*" Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[2] **Edureka Youtube India** *https://youtu.be/L2cAjgc1-bo.*

[3] **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**, 2nd Edition by Aurélien éron Released September 2019 Publisher(s): O'Reilly Media, Inc. ISBN: 9781492032649

[4] **M. Jain, G. Kaur, M. P. Quamar and H. Gupta (2021)**, "*Handwritten Digit Recognition Using CNN*," 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), 2021, pp. 211-215, doi: 10.1109/ICIPTM52218.2021.9388351.