HANDWRITTEN DIGIT RECOGNITION

Mini Project Report

A Dissertation submitted to the RGUKT-AP in partial fulfillment of the degree of

Bachelor of Technology in Computer Science

By

Sunkara Bhavana Reddy (R170909)

Under the guidance of :

Ms.Hima Bindu

Assistant professor

Computer Science Department

Rajiv Gandhi University of Knowledge Technologies AP-IIIT, Rk Valley, Idupulapaya, Kadapa - 516 330 Andhra Pradesh, India

This is to certify that the dissertation entitled “Internship Report” submitted by Sunkara Bhavana Reddy, having Id.No.R170909, in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science is a bonafide work carried out by him under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Ms. Hima Bindu
Project Internal Guide
Computer Science Dept
RGUKT, RK VALLEY

DECLARATION

I Shaik Mohammad Yaseen hereby declare that this Dissertation entitled “Internship Report” submitted by me under the guidance and supervision of Asst Prof. Ms.Shabana is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date:-

Place:- RGUKT, RK VALLEY.

Sunkara Bhavana Reddy(R170909)

Acknowledgments

At the very outset of this report. I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor.

Without their active guidance, help, cooperation & encouragement, I wouldn't have made headway in the project.
I would like to thank Ms. Shabana for their valuable suggestions through out the project.

With Sincere Regards,
Sunkara Bhavana Reddy
(R170909)

HANDWRITTEN DIGIT RECOGNITION

ABSTRACT:

The identification of hand-written digits is amoung the most significant issue in the application for pattern detection. In many applications such as postal code, check online routing bank accounts, data form entry,etc. , the applications of digits recognition include the centre of issue is the need to construct an appropriate algorithm that can recognise written digits and that users upload through scanner and other digital device.

We take MNIST repositery from NIST results. These MNIST dataset accomadates the collection of handwrittenscanned images.
The primary objective of this project is to render hand-written digits recognition to be reliable and precise.

For the identification of digits using MNIST many machine learning algorithms have used including :
• Numpy
• pandas
• Tensorflow
• Convolution neural network 2D layer.

In CNN we use : Convolution, pooling and flattening. Here we use 2 layers of convolution and then compute the output layer to predict the hand-written digits.

Table of Contents

1. INTRODUCTION

Machine learning and deep learning plays an important role in computer sciences and artificial intelligence. The use of machine learning, deep learning and related principles have lowered the human efforts on industry.

Handwritten digit recognition has gained a good amount of popularity from the very beginning of machine learning and deep learning to an expert who has been practicing for years. Developing such a machine needs proper understanding of classification of digits and the difference between the minor and major points to properly differentiate between different digits which can be only possible with proper training and testing Handwritten recognition (HWR) is the ability of a computer to receive and understand intelligible handwritten input from sources such as paper documents, user input touch-screens and other devices. The image of the written text may be sensed from a piece of paper by optical scanning (optical character recognition) or intelligent word recognition or by user input.

Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, a generally easier task as there are more clues available This paper presents recognizing the handwritten digits (0 to 9) from the famous MNIST dataset using TensorFlow framework(library) and python as language and its libraries as user enters the respective digit the machine would recognize and show the results with accuracate percentage

2. METHODOLOGY

2.1 Tensorflow

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization

techniques for easy calculation of many mathematical expressions Tensorflow is an open source library created by the Google Brain Trust for heavy computational work, geared towards machine learning and deep learning tasks. Tensor Flow is built on C,C++ making it very fast while it is available for use via Python, C++, Haskell, Java and Go API depending upon the type of work.

It created data graph flows for each model, where a graph consists of two units – a tensor and a node.
a) Tensor: A tensor is any dimensional array which is not single dimensional.
b) Node: A node is a mathematical computation that is being worked at the moment to give the desired result.

A data graph flow essentially maps the flow of information via the interchange between these two
components. As the graph is completed, the model is executed for the output.

2.2 MNIST Dataset
MNIST (Modified National Institute of Standards and Technology) consists of samples of handwritten digits, they contain total 70,000 images us of which 60,000 are used in training set and 10,000 are used in testing set, both with appropriately labelled images 10 digits (0 to 9). Handwritten digits are images referring the form 28*28 gray scale intensities of images representing an image with the first column to be labelled as (0 to 9) for every image. Similarly, it has opted for the case of the testing set as 10,000 images with a label of 0 to 9 thus. MNIST is a computer science and vision database consisting of handwritten digits, with labels identifying the digits appropriately,every MNIST data point has two parts: an image of a handwritten digit and its corresponding label.

To start with Tensorflow, we will be using the MNIST database to create an image identifying model based on simple feedforward neural network with no hidden layers respectively.

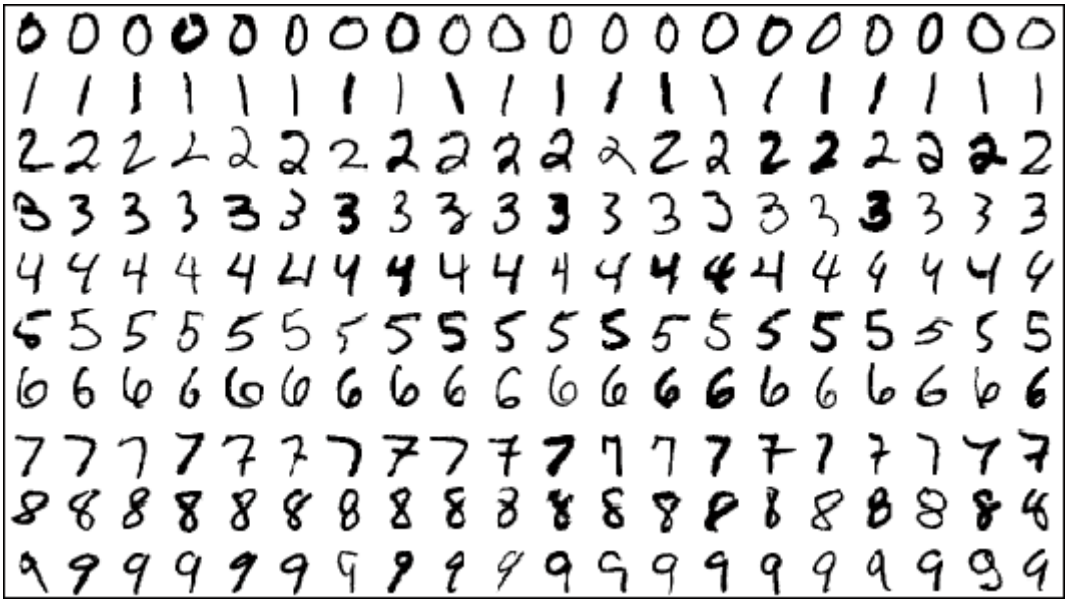
The following figure represents the example sample of the MNIST dataset which is to be used using which the system will be trained and then tested for respected output.

3. Convolutional Neural Network:

CNN is a deep learning neural network used for processing structured arrays of data.

CNN is a feed forward neural network with convolutional layers assembled on top of each other, each one recognises more sophisticated shapes. It has some group of layers and hidden layers followed by activation layers.

With 3 or 4 layers we can easily recognize handwritten written digits and with 25 layers we can easily differentiate human faces



MNIST dataset

3. IMPLEMENTING THE HANDWRITTEN DIGIT'S RECOGNITION MODEL

We will be building simple feedforward neural network using softmax to predict the number in each
image. We begin by calling in a Python environment.

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow.keras.utils as tku
```

 # Preprocessing the Training set

```
train_datagen = ImageDataGenerator(rescale = 1./255 ,shear_range = 0.2,
zoom_range = 0.2, horizontal_flip = True)
training_set
=train_datagen.flow_from_directory('/home/student/Desktop/Projects/datasets/
MNIST/training_set',target_size = (28, 28),batch_size = 32,
class_mode='categorical'
```

 # Preprocessing the Test set

```
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,
zoom_range = 0.2, horizontal_flip = True)
test_set=train_datagen.flow_from_directory('/home/student/Desktop/Projects/
datasets/MNIST/test_set', target_size = (28, 28), batch_size = 32,
class_mode='categorical’
```

#Initializing CNN

```
cnn=tf.keras.models.Sequential()
```

#step1:Convolution

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu',
input_shape=[28, 28, 3]))
```

#step-2 : Pooling

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

 # Adding a second convolutional layer

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

 # Step 3 – Flattening

```
cnn.add(tf.keras.layers.Flatten())
```

 # Step 4 - Full Connection

```
cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
```

 # Step 5 - Output Layer

```
cnn.add(tf.keras.layers.Dense(units=10, activation='softmax'))
```

 # Part 3 - Training the CNN
 # Compiling the CNN

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

 # Training the CNN on the Training set and evaluating it on the Test

```
trained_model = cnn.fit(x = training_set, validation_data = test_set, epochs =
20)
```

```
Epoch 1/20
1167/1167 [==============================] - 612s 521ms/step - loss: 0.3840 -
accuracy: 0.8765 - val_loss: 0.1780 - val_accuracy: 0.9509
Epoch 2/20
1167/1167 [==============================] - 441s 378ms/step - loss: 0.1436 -
accuracy: 0.9560 - val_loss: 0.1258 - val_accuracy: 0.9603
Epoch 3/20
1167/1167 [==============================] - 381s 327ms/step - loss: 0.1049 -
accuracy: 0.9659 - val_loss: 0.0823 - val_accuracy: 0.9742
Epoch 4/20
1167/1167 [==============================] - 382s 327ms/step - loss: 0.0877 -
accuracy: 0.9722 - val_loss: 0.0815 - val_accuracy: 0.9738
Epoch 5/20
1167/1167 [==============================] - 427s 366ms/step - loss: 0.0749 -
accuracy: 0.9754 - val_loss: 0.0751 - val_accuracy: 0.9753
Epoch 6/20
1167/1167 [==============================] - 411s 352ms/step - loss: 0.0656 -
accuracy: 0.9791 - val_loss: 0.0712 - val_accuracy: 0.9781
Epoch 7/20
1167/1167 [==============================] - 96s 82ms/step - loss: 0.0597 -
accuracy: 0.9810 - val_loss: 0.0631 - val_accuracy: 0.9822
Epoch 8/20
1167/1167 [==============================] - 100s 86ms/step - loss: 0.0550 -
accuracy: 0.9825 - val_loss: 0.0623 - val_accuracy: 0.9792
Epoch 9/20
1167/1167 [==============================] - 347s 298ms/step - loss: 0.0541 -
accuracy: 0.9826 - val_loss: 0.0683 - val_accuracy: 0.9788
Epoch 10/2
n1167/1167 [==============================] - 473s 405ms/step - loss: 0.0478 -
accuracy: 0.9854 - val_loss: 0.0572 - val_accuracy: 0.9800
Epoch 11/20
1167/1167 [==============================] - 567s 486ms/step - loss: 0.0440 -
accuracy: 0.9864 - val_loss: 0.0502 - val_accuracy: 0.9843
Epoch 12/20
1167/1167 [==============================] - 542s 465ms/step - loss: 0.0426 -
accuracy: 0.9865 - val_loss: 0.0542 - val_accuracy: 0.9833
Epoch 13/20
1167/1167 [==============================] - 211s 181ms/step - loss: 0.0385 -
accuracy: 0.9873 - val_loss: 0.0617 - val_accuracy: 0.9811
Epoch 14/20
1167/1167 [==============================] - 113s 97ms/step - loss: 0.0393 -
accuracy: 0.9875 - val_loss: 0.0545 - val_accuracy: 0.9830
Epoch 15/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0374 -
accuracy: 0.9880 - val_loss: 0.0508 - val_accuracy: 0.9822
Epoch 16/20
1167/1167 [==============================] - 95s 81ms/step - loss: 0.0335 -
accuracy: 0.9888 - val_loss: 0.0537 - val_accuracy: 0.9830
Epoch 17/20
1167/1167 [==============================] - 96s 82ms/step - loss: 0.0329 -
accuracy: 0.9890 - val_loss: 0.0552 - val_accuracy: 0.9815
Epoch 18/20
1167/1167 [==============================] - 94s 81ms/step - loss: 0.0306 -
accuracy: 0.9902 - val_loss: 0.0446 - val_accuracy: 0.9854
Epoch 19/20
1167/1167 [==============================] - 95s 82ms/step - loss: 0.0305 -
accuracy: 0.9904 - val_loss: 0.0533 - val_accuracy: 0.9848
Epoch 20/20
1167/1167 [==============================] - 123s 105ms/step - loss: 0.0328 -
accuracy: 0.9894 - val_loss: 0.0503 - val_accuracy: 0.9852"}]}
```

Part 4 - Making a single prediction

```
import numpy as np
from keras.preprocessing import image
```

```
test_image=image.load_img('/home/student/Desktop/Projects/datasets/MNIST/
single_prediction/img_52.jpg', target_size = (28, 28))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
results = (cnn.predict(test_image))
training_set.class_indices
if results[0][0] == 1:
  prediction = 'Zero'
elif results[0][1] == 1:
  prediction = 'One'
elif results[0][2] == 1:
  prediction = 'Two'
elif results[0][3] == 1:
  prediction = 'Three'
elif results[0][4] == 1:
  prediction = 'Four'
elif results[0][5] == 1:
  prediction = 'Five'
elif results[0][6] == 1:
  prediction = 'Six'
elif results[0][7] == 1:
  prediction = 'Seven'
elif results[0][8] == 1:
  prediction = 'Eight'
else:
  prediction = 'Nine'
```

```
print(results)
```

o/p: [[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]\n"}]},

```
print(prediction)
```

o/p: Eight

4.CONCLUSION

We practiced machine learning techniques including use of Tensorflow and Convolutional neural network to obtain the appropriate digit recognition .We built handwritten recognizers evaluated their performances on MNIST (Mixed National Institute of Standards and Technology) dataset and then improved the training speed and the recognition performance.

The error rate thus obtained is of 0.3% and training accuracy is 98.94% and test accuracy 98.52% demonstrating significant and promising performance.

Thus by practicing this we have achieved success in properly identifying the digits drawn at different angles and properly displaying the correct digit at a single turn.

Hence the system would be able to recognize the introduced digit according to the formations made and according to the values in the dataset.