



VIT[®]
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

**SCHOOL OF INFORMATION TECHNOLOGY AND
ENGINEERING**

**FALL SEMESTER 2020-21
SOFT COMPUTING**

**BUILDING A HANDWRITTEN DIGIT RECOGNITION
SYSTEM**

M-TECH SOFTWARE ENGINEERING

By

CH.ASHOK – 18MIS0034

J.MRUNAL REDDY – 18MIS0337

SRIKAR KOTRA – 18MIS0369

Under the guidance of

PROF.SENTHIL KUMAR.P

SITE

ABSTRACT:

The problem of handwritten digit recognition has long been an open problem in the field of pattern classification. A ton of studies have shown that Neural networks, machine learning have great and efficient performance.

In data classification Deep learning and Neural Network algorithms are a branch of Machine learning that can identify patterns in the data, and use to uncovered patterns to predict future data or to perform alternative kinds of decision making under Deep Learning algorithms which are used to model high level abstractions in data. Digit Recognition is a combination of Deep Learning and Neural Network algorithms, which uses Tensor Flow tool as an interface to develop a model. **Deep learning** is a machine learning technique that lets computers learn by example. Deep Learning uses different types of neural network architectures like object recognition, image and sound classification, and object detection for different types of problems. The more data a Deep Learning algorithm is trained on, the more accurate it is.

INTRODUCTION:

Hand written text recognition is the process of detecting human written text from the online pictures that are uploaded by a scanner or a scanning app and converting the text to machine understandable language and giving the text in image. Apps like Adobe PDF reader and many other Adobe apps, Cam scanner, Google lens and many other apps and other online text reader use the same kind of software which is becoming an almost software now-a-days. The human brain recognizes words and reads text automatically, and software has now been developed that can recognize text as well.

This technology is becoming more accurate all the time, and will eventually be able to read text as well as our brains do. Hand written text recognition is a challenging task due to complexity and diversity of different hand writing styles and light insufficiency, inter-person hand style differences and the variation of performing text expression. Therefore, recognizing hand written expressions has attracted a great deal of interest by researchers in the last two decades. It has applications in the different fields such as Human Computer Interaction, virtual reality, video games, computer graphics, biometrics, psychology, detecting driver fatigue, analysing customer satisfaction, educational software, lie detection and pain assessment.

Many factors contribute in conveying hand written text of an individual. Pose, language, text expressions, writing styles and actions are some of them. Facial expressions have a higher importance since they are easily perceptible. In communicating with others humans can recognize text of another human with a considerable level of accuracy, if we can efficiently utilize heretofore found knowledge in computer science to find practical solutions for automatic recognition of hand written documents, we would be able to attain accuracy that is virtually comparable to the human perception. In the proposed system we are going to analyse various text expressions using CNN algorithms (Multi layer perceptron) and perform (training and testing) on the proposed system and obtain the accuracy of the system in recognizing the text expressions of the human written document.

REVIEW ON VARIOU PAPERS:

Paper 1:

Title: Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models

Authors:S. España-Boquera ; M.J. Castro-Bleda ; J. Gorbe-Moya ; F. Zamora-Martinez

Methodology:

This paper proposes the use of hybrid Hidden Markov Model (HMM)/Artificial Neural Network (ANN) models for recognizing unconstrained offline handwritten texts.

Advantages:

Experiments have been conducted on offline handwritten text lines from the IAM database, and the recognition rates achieved, in comparison to the ones reported in the literature, are among the best for the same task.

Drawbacks:

Since PHOG feature outperform others, we use this feature for full word recognition, For this purpose initially upper and lower zone components are recognized by PHOG features and SVM classifier.

Metrics used:

The system has been tested on a sufficiently large and variation-rich dataset consisting of 11,253 training and 3,856 testing data.

Literature review:

The structural part of the optical models has been modeled with Markov chains, and a Multilayer Perceptron is used to estimate the emission

probabilities. This paper also presents new techniques to remove slope and slant from handwritten text and to normalize the size of text images with supervised learning methods.

Paper 2:

Title: Handwritten Text Recognition using Deep Learning

Authors: BatuhanBalci, Dan Saadati, Dan Shiferaw

Methodology: In this paper, two main approaches are used for classifying words directly and character segmentation. For the former, we use Convolutional Neural Network (CNN) with various architectures to train a model that can accurately classify words. For the latter, we use Long Short Term Memory networks (LSTM) with convolution to construct bounding boxes for each character. We then pass the segmented characters to a CNN for classification, and then reconstruct each word according to the results of classification and segmentation.

Advantages: We can use a character/word-based language-based model to add a penalty/benefit score to each of the possible final beam search candidate paths, along with their combined individual probabilities, representing the probability of the sequence of characters/words.

Drawbacks: They have achieved a training accuracy of 35 percent and validation of accuracy of 27 percent and we guess it can be increased in terms of accuracy.

Metrics used: CNN, LSTM and concepts of Machine Learning are used in the paper.

Literature review: This project seeks to classify an individual handwritten word so that handwritten text can be translated to a digital form. Briefly, CTC (Connectionist Temporal Classification) is a method/loss function developed by Graves et. al. for training recurrent neural networks to label output sequences. Also used for such tasks as labelling speech signal data with word-level transcriptions, RNNs with CTC loss have been demonstrated to be more effective than more traditional approaches such as CRFs (Conditional Random Fields) and HMMs because of their automated learning process, only needing an input/output data representation as opposed to large hand-engineered features, and can more generally capture context over time and space in their hidden state.

Paper 3:

Title:Build a Handwritten Text Recognition System using TensorFlow

Authors: Harald Schiedl

Methodology: It consists of convolutional NN (CNN) layers, recurrent NN (RNN) layers and a final Connectionist Temporal Classification (CTC) layer. We discussed a NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC

decoding. An implementation using TF is provided and some important parts of the code were presented. Finally, hints to improve the recognition accuracy were given.

Advantages: This implementation can be accessed anywhere on any computer using the code.

Drawbacks:

Data augmentation is poor.

Metrics used: They have taken look at Model.py, as the other source files are concerned with basic file IO (DataLoader.py) and image processing (SamplePreprocessor.py).

Literature review: They have built a Neural Network (NN) which is trained on word-images from the IAM dataset. As the input layer (and therefore also all the other layers) can be kept small for word-images, NN-training is feasible on the CPU (of course, a GPU would be better). This implementation is the bare minimum that is needed for HTR using TF.

Paper 4:

Title: The A2iA Arabic Handwritten Text Recognition System at the Open HaRT2013 Evaluation

Authors: Théodore Bluche ; Jérôme Louradour ; Maxime Knibbe ; Bastien Moyssset ; Mohamed FaouziBenzeghiba ;

Methodology: These systems were based on an optical model using Long Short-Term Memory (LSTM) recurrent neural networks, trained to recognize the different forms of the Arabic characters directly from the image, without

explicit feature extraction nor segmentation. Several recognition systems were also combined with the ROVER combination algorithm.

Advantages: The best system exceeded 80% of recognition rate. Hence the success rate is high.

Drawbacks: This paper has to remove cursive writing style in the input images

Metrics used:

Pattern recognition

Character/shape recognition

Moments

Moment invariants

Zernike moments

Literature review: This paper describes the Arabic handwriting recognition systems proposed by A2iA to the NIST OpenHaRT2013 evaluation. Large vocabulary selection techniques and n-gram language modelling were used to provide a full paragraph recognition, without explicit word segmentation.

Paper 5:

Title: A Scalable Handwritten Text Recognition System

Authors: R. Reeve Ingle ; Yasuhisa Fujii ; Thomas Deselaers ; Jonathan Baccash ; Ashok C. Popat

Methodology: LSTM, Data Synthesis, Rendering, and Degradation Pipeline

Advantages: Addition of HTR capability to a large scale multilingual OCR system

Drawbacks: They have to add more CNN layers

Metrics used: Two-step approach is possible with a small loss of accuracy and observed as per remaining approaches is 83.4 %

Literature review: This paper addresses three problems in building such systems: data, efficiency, and integration. Firstly, one of the biggest challenges is obtaining sufficient amounts of high quality training data. We address the problem by using online handwriting data collected for a large scale production online handwriting recognition system. We describe our image data generation pipeline and study how online data can be used to build HTR models. We show that the data improve the models significantly under the condition where only a small number of real images is available, which is usually the case for HTR models. It enables us to support a new script at substantially lower cost. Secondly, we propose a line recognition model based on neural networks without recurrent connections. The model achieves a comparable accuracy with LSTM-based models while allowing for better parallelism in training and inference. Finally, we present a simple way to integrate HTR models into an OCR system. These constitute a solution to bring HTR capability into a large scale OCR system.

Paper 6:

Title: An architecture for handwritten text recognition system

Authors: Gyeonghwan Kim, VenuGovindaraju & Sargur N. Srihari

Methodology: This paper used neural network concepts like KNN, ANN and also SVM. Key ideas employed in each functional module have been

developed for dealing with the diversity of handwriting in its various aspects with a goal of system reliability and robustness, are described in this paper.

Advantages: Preliminary experiments show promising results in terms of speed and accuracy.

Drawbacks: Low input size (if input of NN is large enough, complete text-lines has to be used or recommended)

Metrics used: Rate of recognition re 78.6% SVM, 81.8% with KNN,94.4% with ANN

Literature review: This paper presents an end-to-end system for reading handwritten page images. Five functional modules included in the system are introduced in this paper:

- (i) pre-processing, which concerns introducing an image representation for easy manipulation of large page images and image handling procedures using the image representation;
- (ii) line separation, concerning text line detection and extracting images of lines of text from a page image;
- (iii) word segmentation, which concerns locating word gaps and isolating words from a line of text image obtained efficiently and in an intelligent manner;
- (iv) word recognition, concerning handwritten word recognition algorithms; and
- (v) linguistic post-pro- cessing, which concerns the use of linguistic constraints to intelligently parse and recognize text.

Paper 7:

Title: Unsupervised writer adaptation applied to handwritten text recognition

Authors: Ali Nosary, Thierry Paquet

Methodology: This paper used neural network concepts like KNN, ANN and also SVM. Key ideas employed in each functional module have been developed for dealing with the diversity of handwriting in its various aspects with a goal of system reliability and robustness, are described in this paper.

Advantages: Rapid and Brief Communication

Drawbacks: They have used Replace LSTM instead of 2D-LSTM making the software old.

Metrics used: Rate of recognition are 78.6 % with SVM, 81.8 % with KNN, 94.4% with ANN

Literature review: This paper deals with the problem of offline hand written text recognition. It presents a system of text recognition that exploits an original principle of adaptation to the handwriting to be recognized. The adaptation principle is based on the automatic learning, during the recognition, of the graphical characteristics of the handwriting. This on-line adaptation of the recognition system relies on the iteration of two steps: a word recognition step that allows to label the writer's representations (allographs) on the whole text and a re-evaluation step of character models.

Paper 8:

Title: A data base for Arabic handwritten text recognition research

Authors: Somaya Al-ma'adeed, David Graham Elliman , Colin Anthony Higgins

Methodology: They have designed, collected and stored a database of Arabic handwriting (AHDB). This resulted in a unique databases dealing with handwritten information from Arabic text, both in terms of the size of the database as well as the number of different writers involved.

Advantages: Offline database is created and can be accessed for any number of applications for further use

Drawbacks: Decoder: word beam search decoding to constrain the output to dictionary words is lacking.

Metrics used: A standard database of images is needed to facilitate research in handwritten text recognition. Arabic, handwriting, recognition, database, pre-processing, cursive script are some of the metrics to be included in this paper.

Literature review:

In this paper they have presented a new database for off-line Arabic handwriting recognition which is very cool in our team point of perspective, together with several pre processing procedures. We designed, collected and stored a database of Arabic handwriting (AHDB). This resulted in a unique

databases dealing with handwritten information from Arabic text, both in terms of the size of the database as well as the number of different writers involved. We further designed an innovative, simple, yet powerful, in place tagging procedure for the database. It enables us to extract at will the bitmaps of words. We also built a pre processing class, which contains some useful pre processing operations. In this paper, the most popular words in Arabic writing were found for the first time using a specially designed program.

Paper 9:

Title: On the Modification of Binarization Algorithms to Retain Grayscale Information for Handwritten Text Recognition

Authors: Mauricio Villegas, Verónica Romero, Joan Andreu Sánchez

Methodology: The HTR system used in this paper followed the classical architecture composed of three main modules: document image pre-processing, line image feature extraction and HMM and language model training/decoding. Several approaches have been proposed in the literature for HTR based on hidden Markov models (HMM), recurrent neural networks, or hybrid systems using HMM and neural networks.

Advantages: A new technique to extract handwritten text from noisy backgrounds and improve the quality of the image is used. It is based on the well known Sauvola binarization method, with the main difference that it

produces a grayscale image, a feature that we show that is more adequate for an HMM-based HTR system.

Drawbacks: Text correction (If the recognized word is not contained in a dictionary, search for the most similar one) has to be included.

Metrics used: Available OCR technologies are not applicable to historical documents, since characters can not be isolated automatically in these images. Therefore, holistic, segmentation-free text recognition techniques are required like hidden Markov models (HMM) algorithms. Results are reported with the publicly available ESPOSALLES database.

Literature review:

This paper proposes to take existing binarization techniques, in order to retain their advantages, and modify them in such a way that some of the original gray scale information is preserved and be considered by the subsequent recognizer. These systems have proven to be useful in a restricted setting for simple tasks.

However, in the context of historical documents, their performance decreases dramatically, mainly due to paper degradation problems encountered in this kind of documents, such as presence of smear, significant background variation, uneven illumination, and dark spots, require specialized image-cleaning and enhancement algorithms.

Paper 10:

Title: A search method for on-line handwritten text employing writing-box-free handwriting recognition

Authors: Hideto Oda, Akihito Kitadai, Motoki Onuma and Masaki Nakagawa

Methodology:

- i) Segmentation of handwriting into text line elements
- ii) Segmentation of text line elements into character pattern elements
- iii) Generation of candidate lattice
- iv) Determination of the optimum text recognition candidate

Advantages: This paper includes a writing-box-free handwriting recognition system.

Drawbacks: Keyword should consist of three characters only.

Metrics used: When the keyword consists of three characters, we have achieved the recall rate 89.4%, the precision rate 93.2% and F measure 0.912.

Literature review: This paper presents a method for writing-box-free on-line handwritten text search. It searches for a target keyword in the lattice composed of candidate segmentations and candidate characters. By considering the accuracy of the recognition method and the length of the keyword, the method decreases noises to be output from the lattice effectively.

Paper 11:

Title: Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models

Authors: Yi-ChaoWu^aFeiYin^aCheng-LinLiu^{abc}

Methodology: Feedforward neural network language model, Recurrent neural network language model, Hybrid language model and Convolutional neural network (CNN) shape models are used in the methodology of the paper.

Advantages: The best performances can be provided with equal error rates 2.15% and 2.63%, respectively. Experimental results indicate the effectiveness and robustness of our proposed method.

Drawbacks: Signatures are aligned to reference template based on Gaussian Mixture Model before verification.

Metrics used: Experimental results on the Chinese handwriting database CASIA-HWDB validate that NNLMs and hybrid RNNLMs.

Literature review: Although back-off N-gram LMs (BLMs) have been used dominantly for decades, they suffer from the data sparseness problem, especially for high-order LMs. Recently, neural network LMs (NNLMs) have been applied to handwriting recognition with superiority to BLMs. With the aim of improving Chinese handwriting recognition, this paper evaluates the effects of two types of character-level NNLMs, namely, feedforward neural network LMs (FNNLMs) and recurrent neural network LMs (RNNLMs). Both FNNLMs and RNNLMs are also combined with BLMs to construct

hybrid LMs. For fair comparison with BLMs and a state-of-the-art system, we evaluate in a system with the same character over-segmentation and classification techniques as before, and compare various LMs using a small text corpus used before.

Paper 12:

Title: END-TO-END TEXT RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

Author: David J. Wu

Methodology: The feature-learning algorithms were able to derive a set of specialized features tuned particularly for the text recognition problem. These learned features were then integrated into a larger, discriminatively-trained convolutional neural network (CNN). CNNs are hierarchical neural networks that have immense representational capacity and have been successfully applied to many problems such as handwriting recognition, visual object recognition and character recognition.

Advantages: An alternative approach that combines the representational power of large, multilayer neural networks with recent developments in unsupervised feature learning.

Lexicon-Driven Recognition is also a big advantage over other papers as far as we observed.

Drawbacks: We observed a substantial gap between the performance of our system on lexicon-driven, cropped word recognition versus the full end-to-end text recognition task. Several reasons contributed to this performance gap.

Metrics used: There are three principal factors that distinguish the CNN from the simple feed- forward neural networks local receptive fields, weight sharing, and spatial pooling or subsampling layers. They consider each of these three properties individually in the context of a visual recognition problem. In particular, suppose that the input to the CNN consists of a single 32-by-32 image patch. For instance, this input can be a 32-by-32 grid of pixel intensity values.

Literature review: This particular approach enables us to train highly accurate text detection and character recognition modules.

Because of the high degree of accuracy and robustness of these detection and recognition modules, it becomes possible to integrate them into a full end-to-end, lexicon-driven, scene text recognition system using only simple off-the-shelf techniques. In doing so, we demonstrate state-of-the- art performance on standard benchmarks in both cropped-word recognition as well as full end-to-end text recognition.

Problems with handwritten digits:

- The handwritten digits are not always of the same size, width, orientation and justified to margins as they differ from writing of person to person, so the general problem would be while classifying the digits due to the similarity between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc.
- This problem is faced more when many people write a single digit with a variety of different handwritings. Lastly, the uniqueness and variety in the handwriting of different individuals also influence the formation and appearance of the digits. Now we introduce the concepts and algorithms of deep learning and machine learning.

Techniques used in Literature Survey:

- KNN (K Nearest Neighbours) : KNN is the non-parametric method or classifier used for classification as well as regression problems. This is the lazy or late learning classification algorithm where all of the computations are derived until the last stage of classification.
- SVM (Support Vector Machine) : Specific type of supervised ML method that intends to classify the data points by maximizing the margin among classes in a high-dimensional space
- NN (Neural Networks) (Multilayer Perceptions) : It's a neural network based classifier, called Multi-Layer perception (MLP), is used

to classify the handwritten digits. Multilayer perceptron consists of three different layers, input layer, hidden layer and output layer.

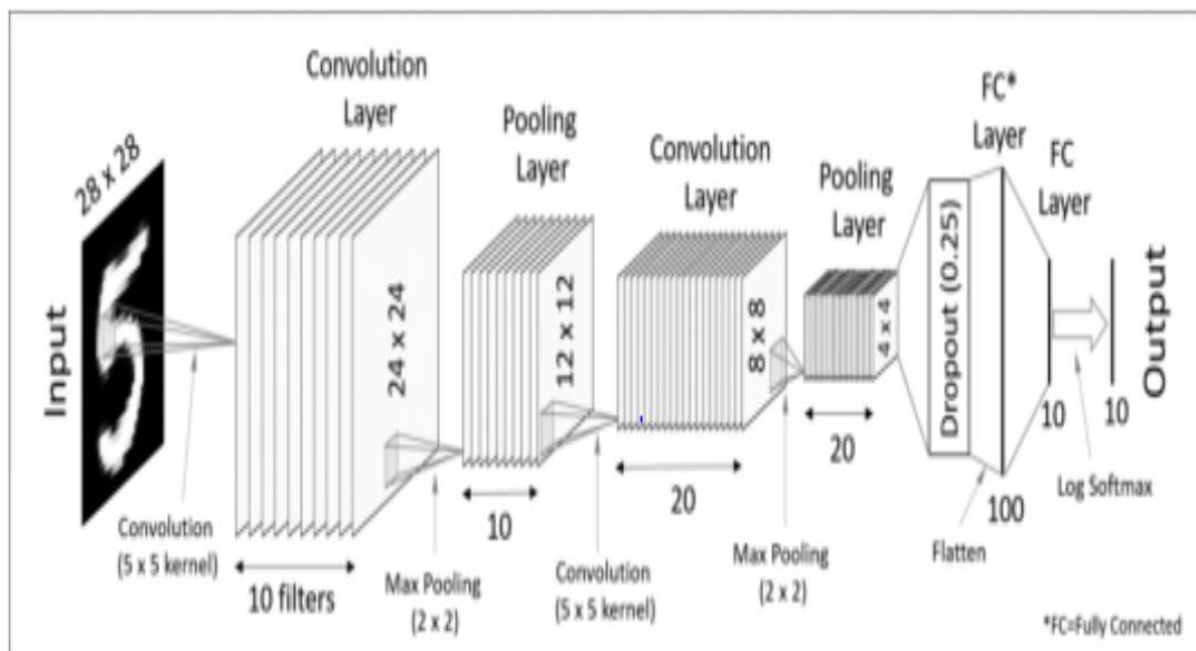
Proposed Methodology:

CNN (Convolutional Neural Network):

- CNN has become famous among the recent times. CNN is part of deep, feed forward artificial neural networks that can perform a variety of task with even better time and accuracy than other classifiers, in different applications of image and video recognition, recommender system and natural language processing.
- Break the image into small image tiles — Similar to sliding window, we can pass sliding window over the entire large image and each result is saved as separate, as a segment of large image as tiny picture tile.
- Feeding each tiny tile into the smaller size neural network — we rarely initialize the parameters with the same values and if not so, then we mark that tile as interesting.
- The first hidden layer is a convolutional layer called a Convolution2D. The layer has 32 feature maps, which with the size of 5×5 and a rectifier activation function.

- Next we define a pooling layer that takes the max called MaxPooling2D. It is configured with a pool size of 2×2 . The next layer is a regularization layer using dropout called Dropout.
- The output is to be processed by standard fully connected layers. Next a fully connected layer with 128 neurons and rectifier activation function. Finally, the output layer has 10 neurons for the 10 classes and a soft max activation function to output probability-like predictions for each class.

CNN architecture in MNIST dataset



The multiple occurring of these layers shows how deep our network is, and
 *Note: The number of filters in the convolution layers is chosen based on the number of filters in the previous layer.

Layers of Convolutional neural network:

- Input: raw pixel values are provided as input.
- Convolutional layer: Input layers translates the results of neuron layer. There is need to specify the filter to be used. Each filter can only be a 5*5 window that slider over input data and get pixels with maximum intensities.
- Rectified linear unit [ReLU] layer: provided activation function on the data taken as an image. In the case of back propagation, ReLU function is used which prevents the values of pixels form changing.
- Pooling layer: Performs a down-sampling operation in volume along the dimensions (width, height).
- Fully connected layer: score class is focused, and a maximum score of the input digits is found.

MNIST dataset:

MNIST (Modified National Institute of Standards and Technology database) is probably one of the most popular datasets among machine learning and deep learning enthusiasts.

The MNIST dataset contains 60,000 small square 28×28 pixel grayscale training images of handwritten digits from 0 to 9 and 10,000 images for testing. So, the MNIST dataset has 10 different classes.

Each image is represented as a 28×28 matrix where each cell contains grayscale pixel value.

Tensor flow:

Tensor Flow 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.

- Tensor: A tensor is any dimensional array which is not single dimensional.
- Node: A node is a mathematical computation that is being worked at the moment to give the desired result.

Activation Functions:

- The model is a simple neural network with one hidden layer with the same number of neurons as there are inputs (784). A rectifier activation function is used for the neurons in the hidden layer.
- A soft max activation function is used on the output layer to turn the outputs into probability-like values and allow one class of the 10 to be selected as the model's output prediction.

- Logarithmic loss is used as the loss function (called `categorical_crossentropy` in Keras) and the efficient ADAM gradient descent algorithm is used to learn the weights.

STEPS:

1. **Import the libraries and load the dataset:** Importing the necessary libraries, packages, and MNIST dataset
2. **Preprocess the data**
3. **Create the model**
4. **Train the Model**
5. **Evaluate the Model**
6. **Saving the model**
7. **Make Prediction**

Source Code:

Importing the necessary libraries:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
```



```
import Conv2D, MaxPooling2D from keras import backend as K # the data,
split between train and test sets (x_train, y_train), (x_test, y_test) =
mnist.load_data() print(x_train.shape, y_train.shape)
```

Preprocessing the data:

```
num_classes = 10 x_train = x_train.reshape(x_train.shape[0], 28, 28, 1) x_test
= x_test.reshape(x_test.shape[0], 28, 28, 1) input_shape = (28, 28, 1) # convert
class vectors to binary class matrices y_train =
keras.utils.to_categorical(y_train, num_classes) y_test =
keras.utils.to_categorical(y_test, num_classes) x_train =
x_train.astype('float32') x_test = x_test.astype('float32') x_train /= 255 x_test /=
255 print('x_train shape:', x_train.shape) print(x_train.shape[0], 'train
samples') print(x_test.shape[0], 'test samples')
```

Creating the model:

```
batch_size = 128 num_classes = 10 epochs = 120 model =
keras.Sequential() model.add(Conv2D(32, kernel_size=(3,
3), activation='relu', input_shape=input_shape)) model.add(Conv2D(64, (3, 3),
activation='relu')) model.add(MaxPooling2D(pool_size=(2,
```

```
2)))model.add(Dropout(0.25))model.add(Flatten())model.add(Dense(256,
activation='relu'))model.add(Dropout(0.5))model.add(Dense(num_classes,
activation='softmax'))model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])
```

Training the model:

```
hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_
test, y_test))print("The model has successfully
trained")model.save('mnist.h5')print("Saving the model as mnist.h5")
```

Evaluation of the model:

```
score = model.evaluate(x_test, y_test, verbose=0)print('Test loss:',
score[0])print('Test accuracy:', score[1])
```

Checking the results with user inputs

```
pip install pygame
```

Mounting drive to take the user inputs:

```
From google.colab import drive
```

```
drive.mount('/content/drive',force_remount=True)
```

Locating the directory containing the user test images

```
import osos.chdir("/content/drive/My Drive/Computing copy/images")! ls
```

Testing the results of user inputs:

```
! pip install pillow#for grabbing the image for test
```

```
from keras.models import load_modelfrom tkinter import *import tkinter as tkfrom PIL import Imageimport numpy as npimport randommodel = load_model('mnist.h5') def predict_Digit(img):  img = img.resize((28,28))  img = img.convert('L')  img = np.array(img)  img = img.reshape(1,28,28,1)  img = img/255.0  res = model.predict([img])[0]  return np.argmax(res), float(max(res)) def classify_handwriting():  img="zero.png"  im=Image.open(img)  display(im)  digit, acc = predict_digit(img)  print("Digit is: ", digit, " Accuracy: ",acc,"%") classify_handwriting()
```

Importing the necessary libraries:

```
[ ] import keras
    from keras.datasets import mnist
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, Flatten
    from keras.layers import Conv2D, MaxPooling2D
    from keras import backend as K
    # the data, split between train and test sets
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    print(x_train.shape, y_train.shape)

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

Pre processing the data:

```
[ ] num_classes = 10
    x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
    x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
    input_shape = (28, 28, 1)
    # convert class vectors to binary class matrices
    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)
    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255
    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Creating the model:

```
[ ] batch_size = 128
    num_classes = 10
    epochs = 120
    model = keras.Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
```

Training the model:

```
[ ] hist = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
print("The model has successfully trained")
model.save('mnist.h5')
print("Saving the model as mnist.h5")
```

Epoch 1/10

469/469 [=====] - 4s 9ms/step - loss: 2.2843 - accuracy: 0.1424 - val_loss: 2.2372 - val_accuracy: 0.3512

Epoch 2/10

469/469 [=====] - 4s 8ms/step - loss: 2.2079 - accuracy: 0.2939 - val_loss: 2.1441 - val_accuracy: 0.6257

Epoch 3/10

469/469 [=====] - 4s 8ms/step - loss: 2.1107 - accuracy: 0.4326 - val_loss: 2.0181 - val_accuracy: 0.7070

Epoch 4/10

469/469 [=====] - 4s 8ms/step - loss: 1.9767 - accuracy: 0.5297 - val_loss: 1.8444 - val_accuracy: 0.7416

Epoch 5/10

469/469 [=====] - 4s 8ms/step - loss: 1.7963 - accuracy: 0.5965 - val_loss: 1.6191 - val_accuracy: 0.7629

Epoch 6/10

469/469 [=====] - 4s 8ms/step - loss: 1.5793 - accuracy: 0.6398 - val_loss: 1.3630 - val_accuracy: 0.7849

Epoch 7/10

469/469 [=====] - 4s 8ms/step - loss: 1.3622 - accuracy: 0.6706 - val_loss: 1.1231 - val_accuracy: 0.8050

Epoch 8/10

469/469 [=====] - 4s 8ms/step - loss: 1.1687 - accuracy: 0.7026 - val_loss: 0.9333 - val_accuracy: 0.8210

Epoch 9/10

469/469 [=====] - 4s 8ms/step - loss: 1.0301 - accuracy: 0.7186 - val_loss: 0.7982 - val_accuracy: 0.8351

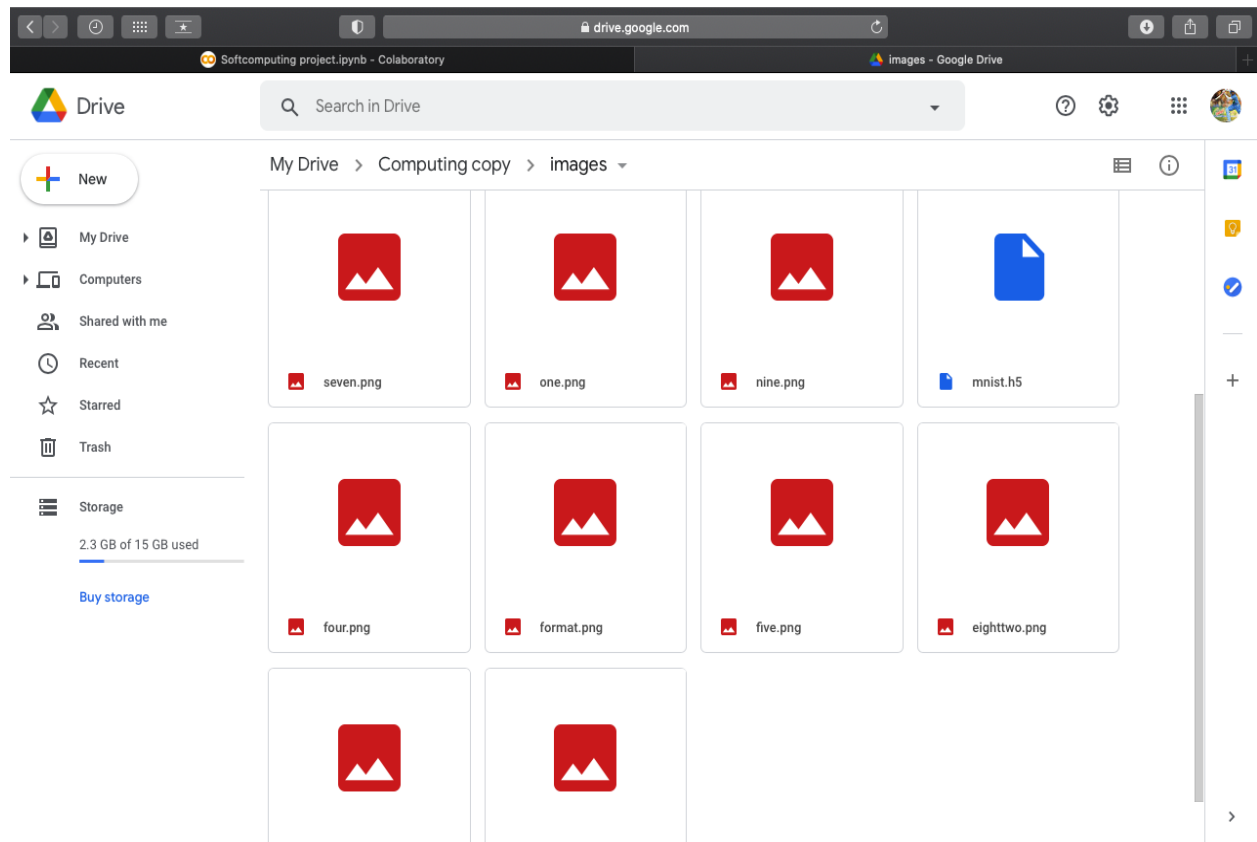
Epoch 10/10

469/469 [=====] - 4s 8ms/step - loss: 0.9290 - accuracy: 0.7378 - val_loss: 0.7018 - val_accuracy: 0.8455

The model has successfully trained

Saving the model as mnist.h5

Mounting the drive:



Results and Discussion:

- Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.
- The classification accuracy for the model on the test dataset is calculated and printed. In this case, we can see that the model achieved

an accuracy of 99.90%, or just less than 1%, which is not bad at all and reasonably close.

- We have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Convolutional neural network which is very effective for image classification purposes.
- We use three types of layers, in this model. They are the convolutional layer, pooling layer and fully connected layer. For this problem, I have defined the following network architecture.
- Note: The input is a sample set of 60,000 images, where each image is 28x28 pixels with 1 channel. The first layer is a Convolutional layer with 20 filters each of size (6,6) followed by another Convolutional layer with 20 filters each of size (3,3). Then, we have a Max Pooling layer of size (4,4). Similarly,
- We have defined same set of layers with Convolutional layer having only 10 filters this time. All the Convolutional layers defined above have ReLU as activation function.
- Then there are fully connected layers with 30 units in the first and 10 (no. of o/p units) in the second. The first layer uses Relu as activation function and the second one uses soft max. We also introduced dropouts in between the stacks of layers to avoid over fitting.
- The key idea behind dropout is to randomly drop units along with their connections to prevent units from co-adapting too much.

Comparative Study:

Convolutional Neural Network is capable of learning any nonlinear function. Hence, these networks are popularly known as Universal Function Approximators.

CNNs have the capacity to learn weights that map any input to the output. One of the main reasons behind universal approximation is the activation function. Activation functions introduce nonlinear properties to the network. This helps the network learn any complex relationship between input and output.

Why we choose CNN over RNN networks?

Advantages of CNN:

- > An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well.
- > Recurrent neural network are even used with convolutional layers extend the effective pixel neighborhood.

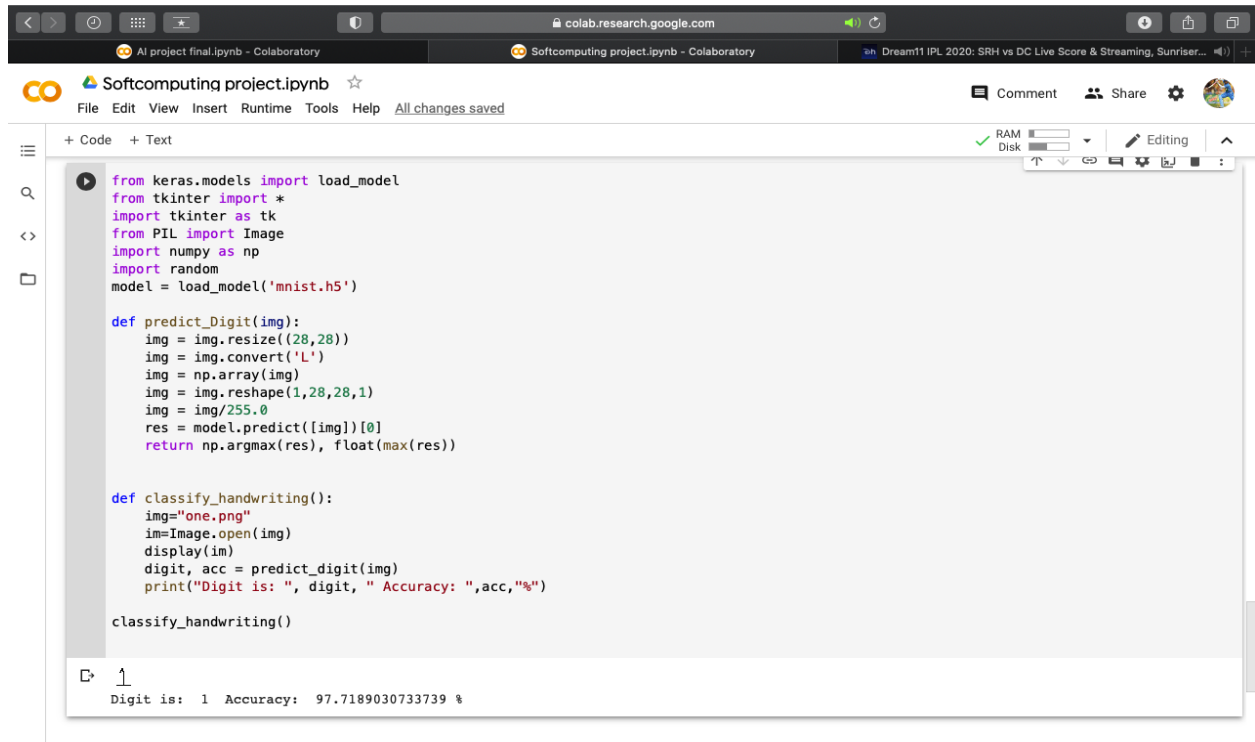
Drawbacks of RNN:

- > Gradient vanishing and exploding problems.
- > Training an RNN is a very difficult task.
- > It cannot process very long sequences if using or relu as an activation function.

The difference between CNN and RNN –

CNN	RNN
It is suitable for spatial data such as images.	RNN is suitable for temporal data, also called sequential data.
CNN is considered to be more powerful than RNN.	RNN includes less feature compatibility when compared to CNN.
This network takes fixed size inputs and generates fixed size outputs.	RNN can handle arbitrary input/output lengths.
CNN is a type of feed-forward artificial neural network with variations of multilayer perceptrons designed to use minimal amounts of preprocessing.	RNN unlike feed forward neural networks - can use their internal memory to process arbitrary sequences of inputs.
CNNs use connectivity pattern between the neurons. This is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.	Recurrent neural networks use time-series information - what a user spoke last will impact what he/she will speak next.
CNNs are ideal for images and video processing.	RNNs are ideal for text and speech analysis.

FINAL CODE OUTPUT:



The screenshot shows a Google Colab notebook titled 'Softcomputing project.ipynb'. The code defines a function `predict_Digit` that takes an image, resizes it to 28x28, converts it to grayscale, and uses a pre-trained Keras model to predict the digit. A `classify_handwriting` function is also defined, which opens and displays 'one.png' before calling `predict_Digit`. The output at the bottom shows the digit '1' with an accuracy of 97.7189030733739 %.

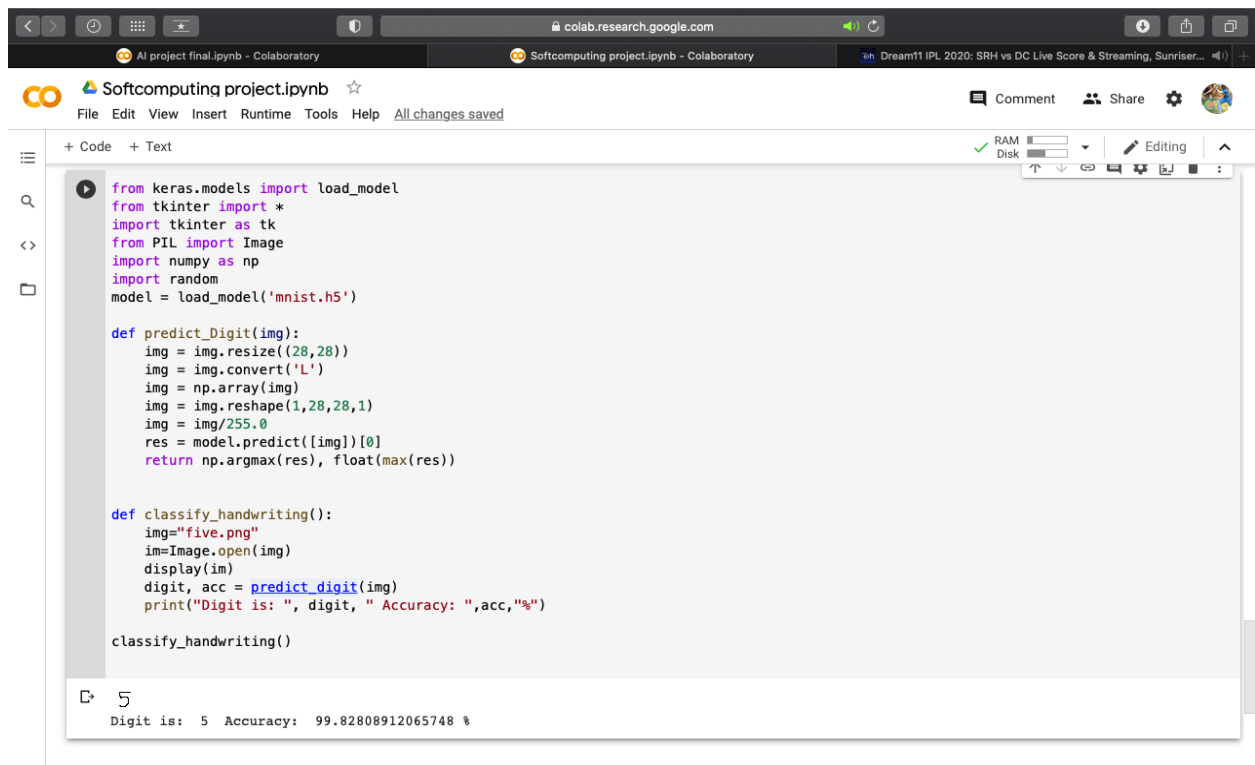
```
from keras.models import load_model
from tkinter import *
import tkinter as tk
from PIL import Image
import numpy as np
import random
model = load_model('mnist.h5')

def predict_Digit(img):
    img = img.resize((28,28))
    img = img.convert('L')
    img = np.array(img)
    img = img.reshape(1,28,28,1)
    img = img/255.0
    res = model.predict([img])[0]
    return np.argmax(res), float(max(res))

def classify_handwriting():
    img="one.png"
    im=Image.open(img)
    display(im)
    digit, acc = predict_digit(img)
    print("Digit is: ", digit, " Accuracy: ",acc,"%")

classify_handwriting()
```

Digit is: 1 Accuracy: 97.7189030733739 %



The screenshot shows the same Google Colab notebook, but the `classify_handwriting` function now opens and displays 'five.png'. The `predict_Digit` function remains unchanged. The output at the bottom shows the digit '5' with an accuracy of 99.82808912065748 %.

```
from keras.models import load_model
from tkinter import *
import tkinter as tk
from PIL import Image
import numpy as np
import random
model = load_model('mnist.h5')

def predict_Digit(img):
    img = img.resize((28,28))
    img = img.convert('L')
    img = np.array(img)
    img = img.reshape(1,28,28,1)
    img = img/255.0
    res = model.predict([img])[0]
    return np.argmax(res), float(max(res))

def classify_handwriting():
    img="five.png"
    im=Image.open(img)
    display(im)
    digit, acc = predict_digit(img)
    print("Digit is: ", digit, " Accuracy: ",acc,"%")

classify_handwriting()
```

Digit is: 5 Accuracy: 99.82808912065748 %



5

Digit is: 5 Accuracy: 99.82808912065748 %

Derived Accuracy for the digit 5



0

Digit is: 0 Accuracy: 97.32832431218921 %

Derived Accuracy for the digit 0

Conclusion:

Machine learning techniques including use of Tensor flow to obtain the appropriate digit recognition this study 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 1.25 and training accuracy is 98% and test accuracy 97% 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

- Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.
- The classification accuracy for the model on the test dataset is calculated and printed. In this case, we can see that the model achieved an accuracy of 99.90%, or just less than 1%, which is not bad at all and reasonably close.
- We have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Convolutional neural network which is very effective for image classification purposes.
- We use three types of layers, in this model. They are the convolutional layer, pooling layer and fully connected layer. For this problem, I have defined the following network architecture.

References:

- https://www.researchgate.net/profile/Christopher_Kermorvant/publication/264860340_The_A2iA_Arabic_Handwritten_Text_Recognition_System/links/544000000cf97c000191000000.pdf

[stem at the OpenHaRT2013 Evaluation/links/5bf28cb292851c6b27c9c123/The-A2iA-Arabic-Handwritten-Text-Recognition-System-at-the-OpenHaRT2013-Evaluation.pdf](#)

- [http://cs231n.stanford.edu/reports/2017/pdfs/810.pdf](#)
- [https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5](#)
- [https://ieeexplore.ieee.org/abstract/document/6830990](#)
- [https://link.springer.com/article/10.1007%2Fs100320050035](#)
- [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.8198&rep=rep1&type=pdf](#)
- [https://www.researchgate.net/profile/Somaya_Al-Maadeed/publication/3966320_A_data_base_for_Arabic_handwritten_text_recognition_research/links/5426fbfc0cf238c6ea7ab645.pdf](#)
- [https://riunet.upv.es/bitstream/handle/10251/64376/Villegas15_IbPria_Grayscale-Text-Enh_draft.pdf?sequence=3](#)
- [http://www.iapr-tc11.org/archive/iwfhr2004_proc/docs/091_x_oda-SearchMethod.pdf](#)
- [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.664.6212&rep=rep1&type=pdf](#)