Final Project Presentation

PYTHON LIBRARY FOR CHARACTER RECOGNITION

Bachelor's of Technology - Computer Science and Engineering

Group Details

Sl.No.	Reg. No.	Name of the student	Department
1	16ETCS002144	ASHISH KUMAR	CSE
2	16ETCS002115	SATYAM AGRAWAL	CSE
3	16ETCS002114	SARAH Z ANWAR	CSE
4	16ETCS002161	SAURAV KUMAR	CSE

Batch : FT-16

- **Title of the Project**

  PYTHON LIBRARY FOR CHARACTER RECOGNITION

- **Supervisors**

  Supervisor : : Ms. Pallavi R Kumar

- **Place of Work**

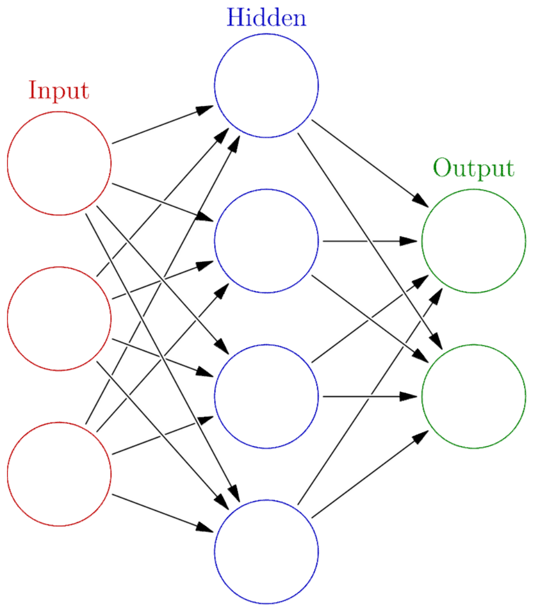
  RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Outline

- Introduction
- Motivation(Project Concept and its relevance)
- Aims and Objectives
  - Title, Aim, Objectives, Methods and Methodology
- Problem Solving
  - Project Concept, Design, Implementation
- Project Costing
- Conclusions
- References
- Demonstration
- Workload Allocation
- Team Experience
- Report Writing and Uploading Product video on YouTube

Introduction

- Supervised machine learning to convert hard copy of document to digital text document using Image processing and Neural Network.
- Artificial neural networks (ANN) are computing systems vaguely inspired by the biological neural networks that constitute animal brains.
- Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.



Motivation (Project Concept and its relevance)

- Earlier, in field of studies and documenting, if someone were presented with a large amount of handwritten text to edit, they would have to input it manually into the computer which is time consuming.
- The huge amount of data stored in form of papers in the government offices, case details in courts and police stations etc. become hard to search through and interpret.
- These handwritten or printed documents stay in a large pile of pages at workplace and makes the work place messy and put a psychological tension on the person trying to get the data out of those piles.

Motivation (Project Concept and its relevance)

- Handwritten notes and important documents are prone to getting lost or destroyed, saving a digital copy is much more efficient and helpful.
- This project holds great significance since it aims to assist in easing the conversion from manual to digital text type.
- With the help of our tool any amount of handwritten data or printed documented can be turned into digital document.
- Process of editing and searching in the documents get easier.
- Interpreting the data becomes very easy hence a data which need to very specific can also be obtained without any effort.

Aim

AIM: PYTHON LIBRARY FOR CHARACTER RECOGNITION

Objectives

1. To conduct literature survey on Character Recognition using neural network and textual image processing.
2. To design the methodology to be followed for developing the library.
3. To develop and implement the model that recognizes a hard text from the image and convert into a digital text document.
4. To test and validate the model for various textual images.
5. To document the report by unifying all the results and outcomes.

Methods and Methodologies

Objective 1: Literature Review

1. Machine Learning in Document Analysis and Recognition

→Summary: The objective of this paper is to understand principal components required for the implementation of a pattern recognition that are pre-processing, object-segmentation, object recognition and post processing in significant details. The paper explores the application of different pattern recognition outputs including handwriting and text recognition.

Literature Review

2. Neural Network based Handwritten Character Recognition system

→Summary- The effectiveness of their work emphasizes using feature extraction using character geometry and gradient technique from scanned images containing handwritten characters. Their proposed methodology has produced good results for images containing handwritten text written in different styles, size and alignment.

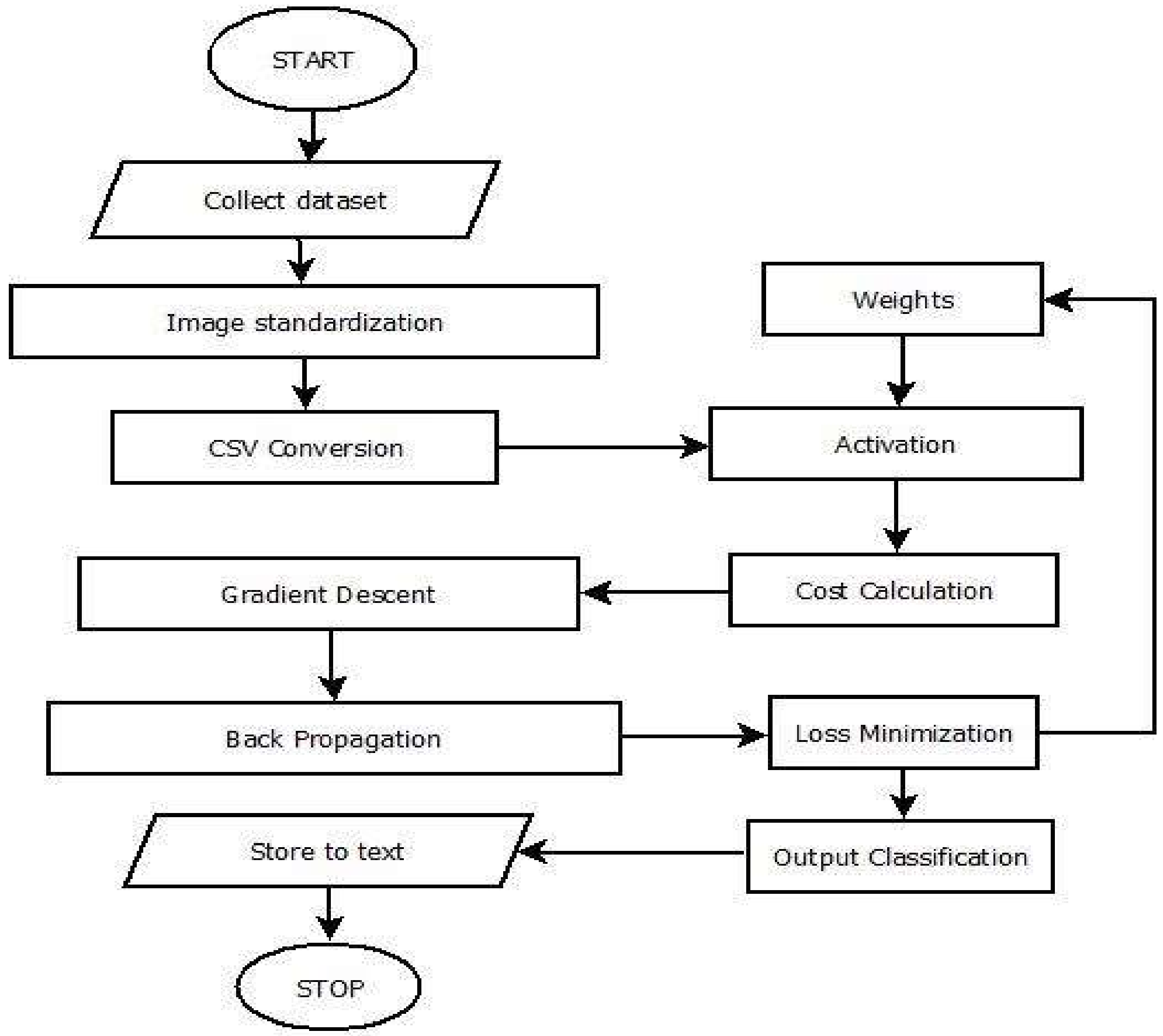
Method and Methodology Contd..

Objective 2: Design

Assumptions

- The input image is assumed to contain only English characters.
- The input image is assumed to be a text written on un-ruled white paper.
- The input image is assumed to contain text that are not cursive.
- The input image is assumed to contain text that does not contain noise e.g. ~~Ashish .~~

Design - Flowchart



Method and methodology contd..

Objective 3: *Implementation*

- We are using kaggle.com for the implementation which includes about 372,449 image data of characters.
- The assumptions are made to develop a better classifier as the we have dataset limited to English characters only.
- To process the computation we are taking the help of google colaboratory as they provide free GPU and TPU processing.

Method and methodology contd..

Objective 4: *Testing and validation*

- Collecting various test samples for the input.
- Conducting test on the test samples.
- Conducting validation check of the tool.
- Checking the performance of the tool for test inputs.

Method and methodology contd..

Objective 5: *Documentation*

- Based on the literature survey done, the requirements are reported.
- Demonstration of the developed algorithm with the test data will be reported.
- After the design, implementation, verification, testing and validation, results of the system are reported.
- Analyze performance of the tool and report the performance test’ results.

PROJECT CONCEPT

- Concept of the project is to construct a model which will take handwritten document(image) as input dataset, preprocess the image, pass through the trained the neural network model to recognize the characters and then store the recognized characters into a text document.

Development and Implementation

- For the development of the model we have used deep learning model.
- Deep learning is an aspect of artificial intelligence (AI) that is concerned with emulating the learning approach that human beings use to gain certain types of knowledge. At its simplest, deep learning can be thought of as a way to automate predictive analytics.
- While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

H E L L O

W O R L D

Implementation

- Training dataset

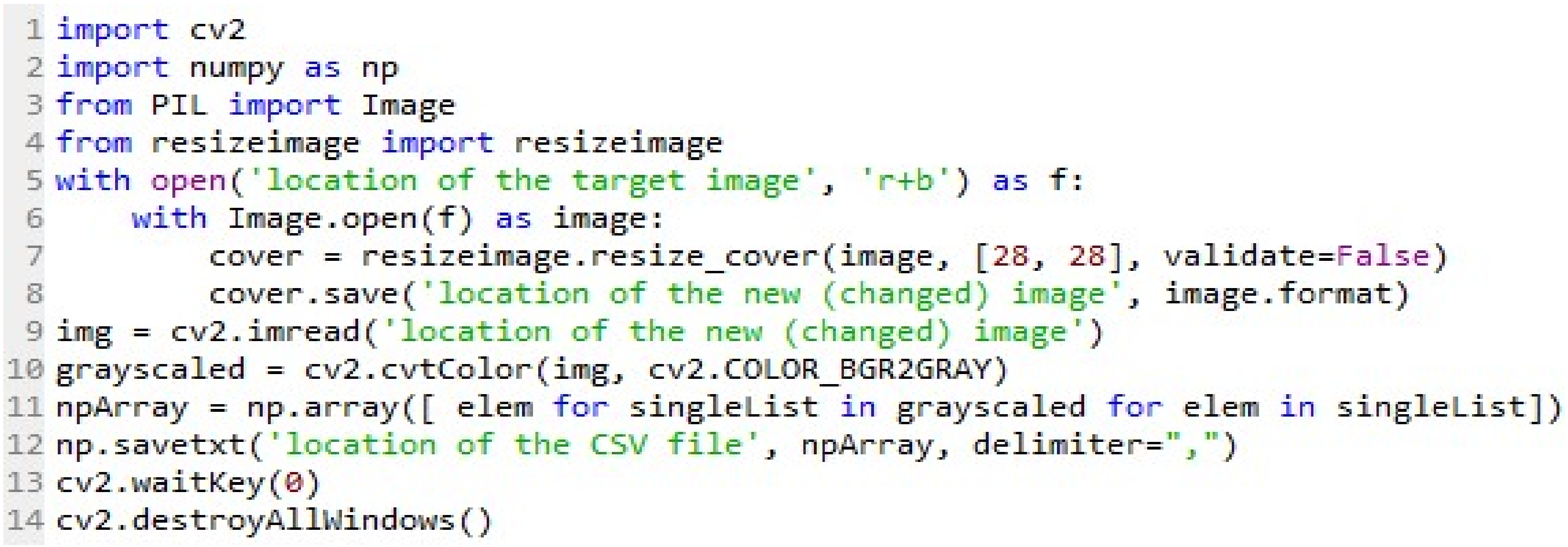
➢ The dataset contains the 372450 image of 28*28 pixels converted into csv form. The data contains 26 classes of the data that are the 26 capital letters of the English alphabet.

➢ Classifying such a big dataset into so many classes requires the neural network to be big and well trained. To accomplish this we have used deep learning consisting of 3 hidden layers where the 1st layer contains 100 neurons and the 2nd layer contains 50 neurons and the last layer of the output layer contains 26 neurons for the 26 classes. Where the input layer has 784(28*28) units.

Implementation

- Test dataset

For test dataset we have taken images of the letters and converted that image into 28*28 pixels and then into csv form for our model. To do that we used the following code:



```python
import cv2
import numpy as np
from PIL import Image
from resizeimage import resizeimage
with open('location of the target image', 'r+b') as f:
    with Image.open(f) as image:
        cover = resizeimage.resize_cover(image, [28, 28], validate=False)
        cover.save('location of the new (changed) image', image.format)
img = cv2.imread('location of the new (changed) image')
grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
npArray = np.array([ elem for singleList in grayscaled for elem in singleList])
np.savetxt('location of the CSV file', npArray, delimiter=",")
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Implementation

- Weights and biases

➢ The weights and biases are the variables of the model which are supposed to be initialized with certain value.

➢ The weights in the trained model were initialized with Xavier Initializer. The Xavier initializer work better than any other random initializer for the weights and biases specially, when coupled with ReLu activation function.

➢ Whereas the bases are always initialized with zeros hence it the model also it has been initialized with zeros.

➢ The weights and biases are the parameters in the model, which are trained when the model trains, they are trained by minimizing the cost of error.

Implementation

- Forward propagation

➢ Forward propagation is the propagation of values of input layer through each neuron of each layer in the network to the output of the output layer is forward propagation.

➢ Since our problem is a multiclass classification problem, hence we need specific set of activation functions.

➢ The activation function used for the hidden layers is ReLU, which classifies the values into a linear output. Both the hidden layers use ReLU function as the activation function but the third layer or the output layer uses softmax layer as the activation function as softmax function is used for multiclass classification problem.

Implementation

- Cost computation

➢ The cost computation of in the model is a very important part of the training. This the part where the error of prediction of the model is identified by subtracting the obtained output of the model with the actual output.

➢ In our model we have used mean of the Softmax cross entropy to calculate cost. Cross entropy indicates the distance between what the model believes the output distribution should be, and what the original distribution really is.

➢ It is used when node activations can be understood as representing the probability that each hypothesis might be true, i.e. when the output is a probability distribution. Thus, it is used as a loss function in neural networks, which have softmax activations in the output layer.

Implementation

- Backpropagation and optimization

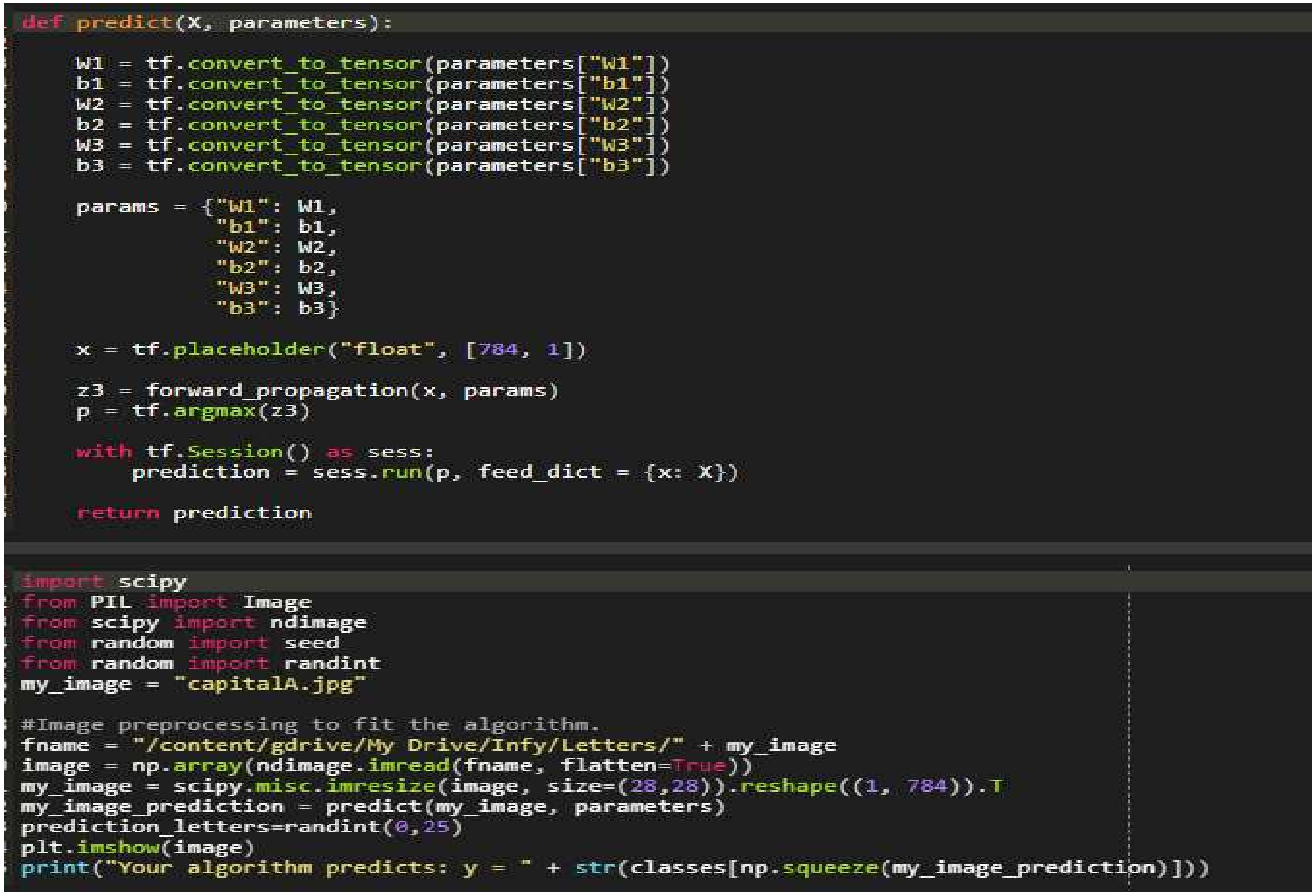
➢ Backpropagation is used to adjust the values of the weights and biases while training by calculating the cost and optimizing the cost.

➢ In the implemented model, we have used Adam Optimizer to optimize the model quickly.

➢ The learning rate for Adam optimizer used in the model is 0.0001. The learning rate is tuned and then decided to go with this value as it gives better performance than other values of the learning rate.

Outcomes

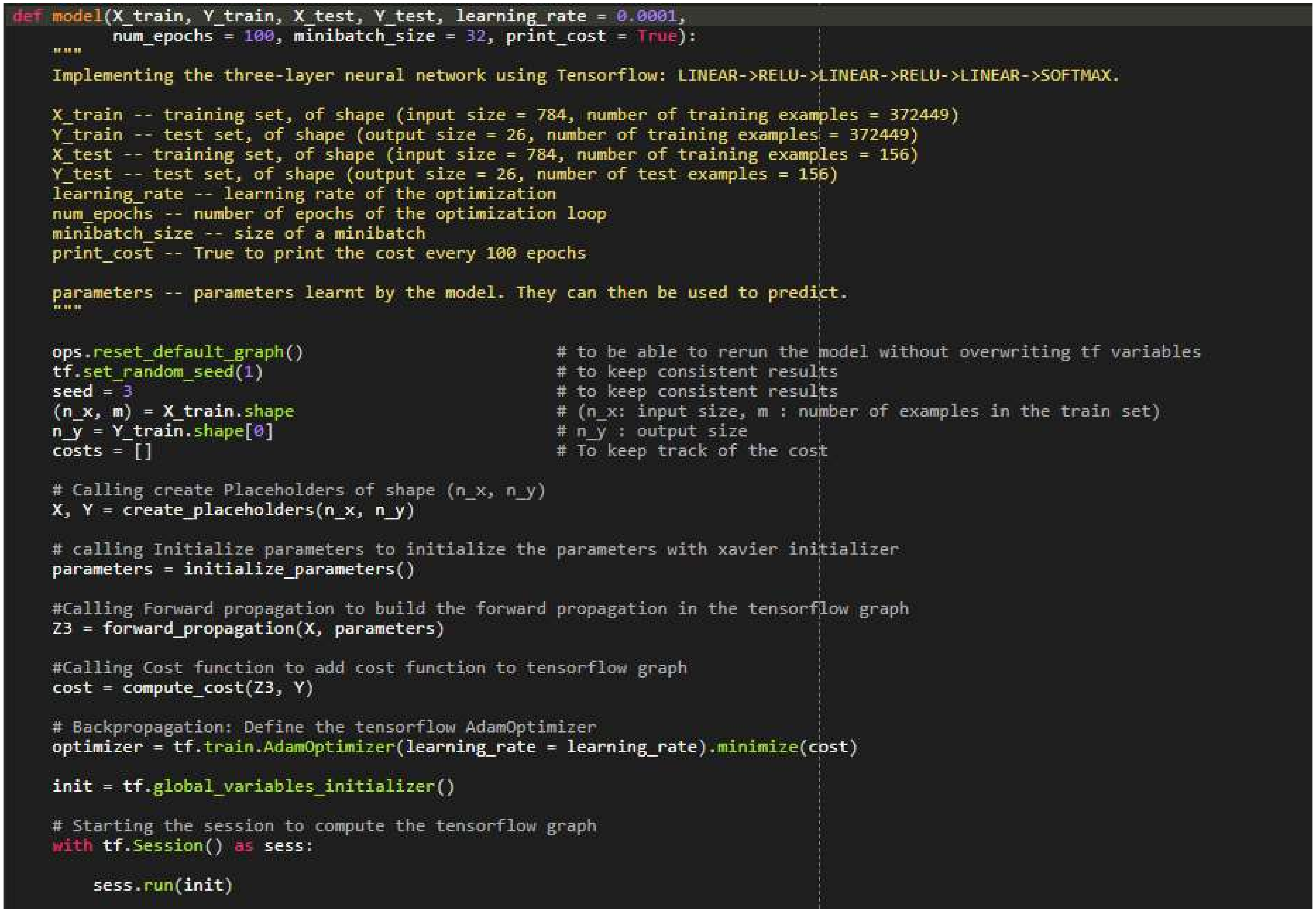
- Prototype



```python
def predict(X, parameters):

    W1 = tf.convert_to_tensor(parameters["W1"])
    b1 = tf.convert_to_tensor(parameters["b1"])
    W2 = tf.convert_to_tensor(parameters["W2"])
    b2 = tf.convert_to_tensor(parameters["b2"])
    W3 = tf.convert_to_tensor(parameters["W3"])
    b3 = tf.convert_to_tensor(parameters["b3"])

    params = {"W1": W1,
              "b1": b1,
              "W2": W2,
              "b2": b2,
              "W3": W3,
              "b3": b3}

    x = tf.placeholder("float", [784, 1])

    z3 = forward_propagation(x, params)
    p = tf.argmax(z3)

    with tf.Session() as sess:
        prediction = sess.run(p, feed_dict = {x: X})

    return prediction
```

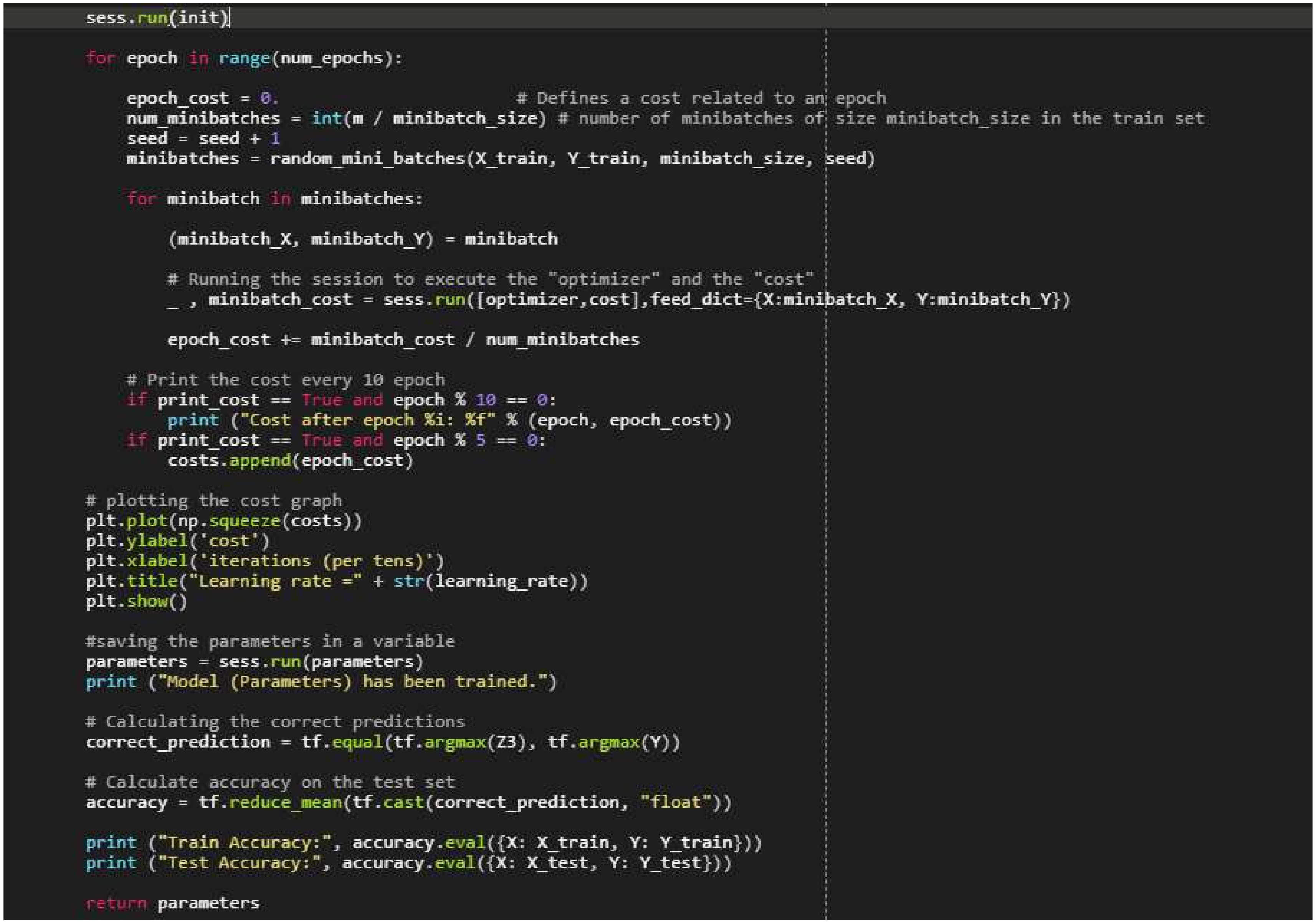
```python
import scipy
from PIL import Image
from scipy import ndimage
from random import seed
from random import randint
my_image = "capitalA.jpg"

#Image preprocessing to fit the algorithm.
fname = "/content/gdrive/My Drive/Infy/Letters/" + my_image
image = np.array(ndimage.imread(fname, flatten=True))
my_image = scipy.misc.imresize(image, size=(28,28)).reshape((1, 784)).T
my_image_prediction = predict(my_image, parameters)
prediction_letters=randint(0,25)
plt.imshow(image)
print("Your algorithm predicts: y = " + str(classes[np.squeeze(my_image_prediction)]))
```

Working model Procedure



```
def model(X_train, Y_train, X_test, Y_test, learning_rate = 0.0001,
          num_epochs = 100, minibatch_size = 32, print_cost = True):
    """
    Implementing the three-layer neural network using Tensorflow: LINEAR->RELU->LINEAR->RELU->LINEAR->SOFTMAX.

    X_train -- training set, of shape (input size = 784, number of training examples = 372449)
    Y_train -- test set, of shape (output size = 26, number of training examples = 372449)
    X_test -- training set, of shape (input size = 784, number of training examples = 156)
    Y_test -- test set, of shape (output size = 26, number of test examples = 156)
    learning_rate -- learning rate of the optimization
    num_epochs -- number of epochs of the optimization loop
    minibatch_size -- size of a minibatch
    print_cost -- True to print the cost every 100 epochs

    parameters -- parameters learnt by the model. They can then be used to predict.
    """

    ops.reset_default_graph()                         # to be able to rerun the model without overwriting tf variables
    tf.set_random_seed(1)                             # to keep consistent results
    seed = 3                                          # to keep consistent results
    (n_x, m) = X_train.shape                          # (n_x: input size, m : number of examples in the train set)
    n_y = Y_train.shape[0]                            # n_y : output size
    costs = []                                        # To keep track of the cost

    # Calling create Placeholders of shape (n_x, n_y)
    X, Y = create_placeholders(n_x, n_y)

    # calling Initialize parameters to initialize the parameters with xavier initializer
    parameters = initialize_parameters()

    #Calling Forward propagation to build the forward propagation in the tensorflow graph
    Z3 = forward_propagation(X, parameters)

    #Calling Cost function to add cost function to tensorflow graph
    cost = compute_cost(Z3, Y)

    # Backpropagation: Define the tensorflow AdamOptimizer
    optimizer = tf.train.AdamOptimizer(learning_rate = learning_rate).minimize(cost)

    init = tf.global_variables_initializer()

    # Starting the session to compute the tensorflow graph
    with tf.Session() as sess:

        sess.run(init)
```

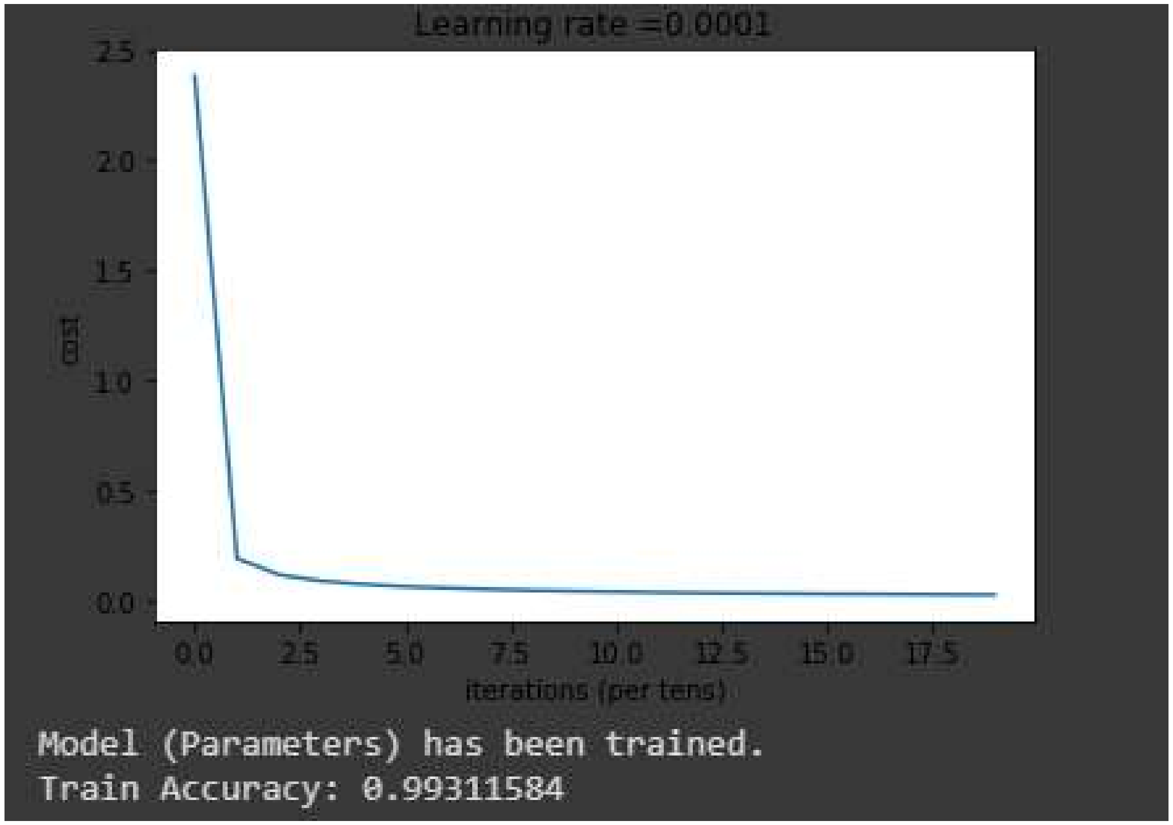
Working model procedure



```
    sess.run(init)

    for epoch in range(num_epochs):

        epoch_cost = 0.                       # Defines a cost related to an epoch
        num_minibatches = int(m / minibatch_size) # number of minibatches of size minibatch_size in the train set
        seed = seed + 1
        minibatches = random_mini_batches(X_train, Y_train, minibatch_size, seed)

        for minibatch in minibatches:

            (minibatch_X, minibatch_Y) = minibatch

            # Running the session to execute the "optimizer" and the "cost"
            _ , minibatch_cost = sess.run([optimizer,cost],feed_dict={X:minibatch_X, Y:minibatch_Y})

            epoch_cost += minibatch_cost / num_minibatches

        # Print the cost every 10 epoch
        if print_cost == True and epoch % 10 == 0:
            print ("Cost after epoch %i: %f" % (epoch, epoch_cost))
        if print_cost == True and epoch % 5 == 0:
            costs.append(epoch_cost)

    # plotting the cost graph
    plt.plot(np.squeeze(costs))
    plt.ylabel('cost')
    plt.xlabel('iterations (per tens)')
    plt.title("Learning rate =" + str(learning_rate))
    plt.show()

    #saving the parameters in a variable
    parameters = sess.run(parameters)
    print ("Model (Parameters) has been trained.")

    # Calculating the correct predictions
    correct_prediction = tf.equal(tf.argmax(Z3), tf.argmax(Y))

    # Calculate accuracy on the test set
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))

    print ("Train Accuracy:", accuracy.eval({X: X_train, Y: Y_train}))
    print ("Test Accuracy:", accuracy.eval({X: X_test, Y: Y_test}))

    return parameters
```

Accuracy of the trained model

- The accuracy of the trained model on the training dataset is more that 99%.



```
Model (Parameters) has been trained.
Train Accuracy: 0.99311584
```

Prediction

While the model works very great with the training data it does not work so well with the test data.

The reasons for the model not working well with the test data are:

- The test data is very small as compared to the training data.
- The model uses only deep learning and not the convolutional Neural network.
- The train data and the test data are from different dataset.
- The model is overfit for the train data so doesn't work well with the test data.
- The model doesn't use dropout and regularization to improve the model (Which we intend to use but for the lack of time we are sticking to the model we have right now).

Project Costing

Component	Units	Cost
Man power	16weeks*4people	₹4,00,0000
Laptop (Hardware and software)	1 Units	₹40,000
Internet and Electricity	4	₹5,000
Total		₹4,45,000

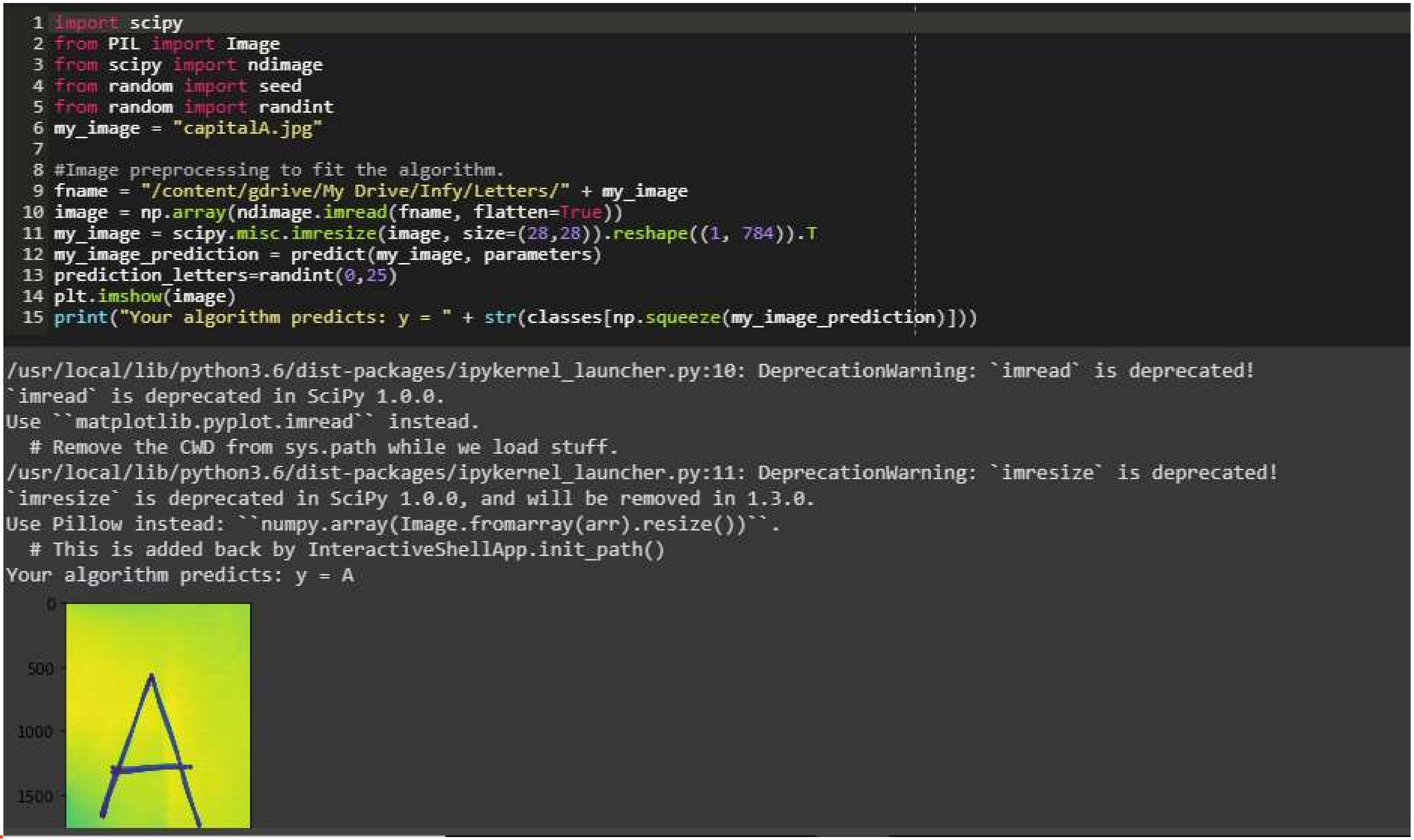
Conclusions

- The character recognition methods have developed remarkably in the last decade. A variety of techniques have emerged, influenced by developments in related fields such as image recognition and face recognition.
- In this paper, we have proposed an organization of these methods under two basic strategies. It is hoped that this comprehensive discussion will provide insight into the concepts involved, and perhaps provoke further advances in the area.
- The difficulty of performing accurate recognition is determined by the nature of the text to be read and by its quality. Generally, improper segmentation rates for unconstrained material increase progressively from machine print to handprint to cursive writing.

References

- Attigeri, S. (2018). Neural Network based Handwritten Character Recognition system. *International Journal of Engineering and Computer Science*, *7*(03), 23761-23768.
- LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P. and Vapnik, V., 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, *261*, p.276.
- Simone Marinai, Hiromichi *Fujisawa.Machine Learning in Document Analysis and Recognition*, 2008 Springer-Verlag Berlin Heidelberg.

Demonstration



```python
import scipy
from PIL import Image
from scipy import ndimage
from random import seed
from random import randint
my_image = "capitalA.jpg"

#Image preprocessing to fit the algorithm.
fname = "/content/gdrive/My Drive/Infy/Letters/" + my_image
image = np.array(ndimage.imread(fname, flatten=True))
my_image = scipy.misc.imresize(image, size=(28,28)).reshape((1, 784)).T
my_image_prediction = predict(my_image, parameters)
prediction_letters=randint(0,25)
plt.imshow(image)
print("Your algorithm predicts: y = " + str(classes[np.squeeze(my_image_prediction)]))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0.
Use ``matplotlib.pyplot.imread`` instead.
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: DeprecationWarning: `imresize` is deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.3.0.
Use Pillow instead: ``numpy.array(Image.fromarray(arr).resize())``.
  # This is added back by InteractiveShellApp.init_path()
Your algorithm predicts: y = A
```

Work Load Allocation

	Saurav K	Satyam A	Ashish K	Sarah Anwar
Literature Survey	🟧	🟨	🟨	🟧
Documentation	🟨	🟧	🟨	🟧
Requirements analysis	🟨	🟨	🟧	🟨
Designing	🟧	🟨	🟨	🟨
Implementation	🟨	🟨	🟧	🟨
Testing	🟨	🟧	🟨	🟨

Team Experience

- We have learnt and explored our experience on this group project which includes analyzing, researching, documenting and implementing.
- Sharing and discussing different ideas and amount of knowledge amongst the group gave us a brief insight on how to deal with problems and come up with an applicable solution.
- We experienced how to confront difficulties in a team work and how to resolve any sort of argument or problem by coming up with something which is agreeable by everyone.

Team Experience

- While working on project such as this, we exercised a great opportunity of developing and enhancing our technical skills effectively and efficiently.
- We have also acquired different set of skills from each other that is beneficial to each one of us in the long run.
- In conclusion, We’ve learnt a lot of things about ourselves while working with our group members in undertaking this project.

FET

Thank You