CSEE 5590 0001
Special Topics SPRING 2018

DEEP LEARNING
LAB ASSIGNMENT - 3

Submitted On 4/6/2018

Name: Satya Sai Deepthi Katta

Class ID: 21

UMKC

Department of Computer Science and Electrical Engineering

DOCUMENT CONTENTS

Introduction

Logistic Regression is the predictive analysis of the data to explain the relationship between one dependent binary variable and one or more nominal, ordinal independent variables. The assignment is to display the Logistic Regression results and graphs to be plotted in Tensor Board.

Objective

The main aim of the lab work is to create an exposure to Deep Learning concepts.

- Implement Logistic using new dataset
- Show the workflow graph in tensor board
- Change the hyperparameter and comparing the results.

Approach

The approach for the assignment can be defined as simple steps given below:

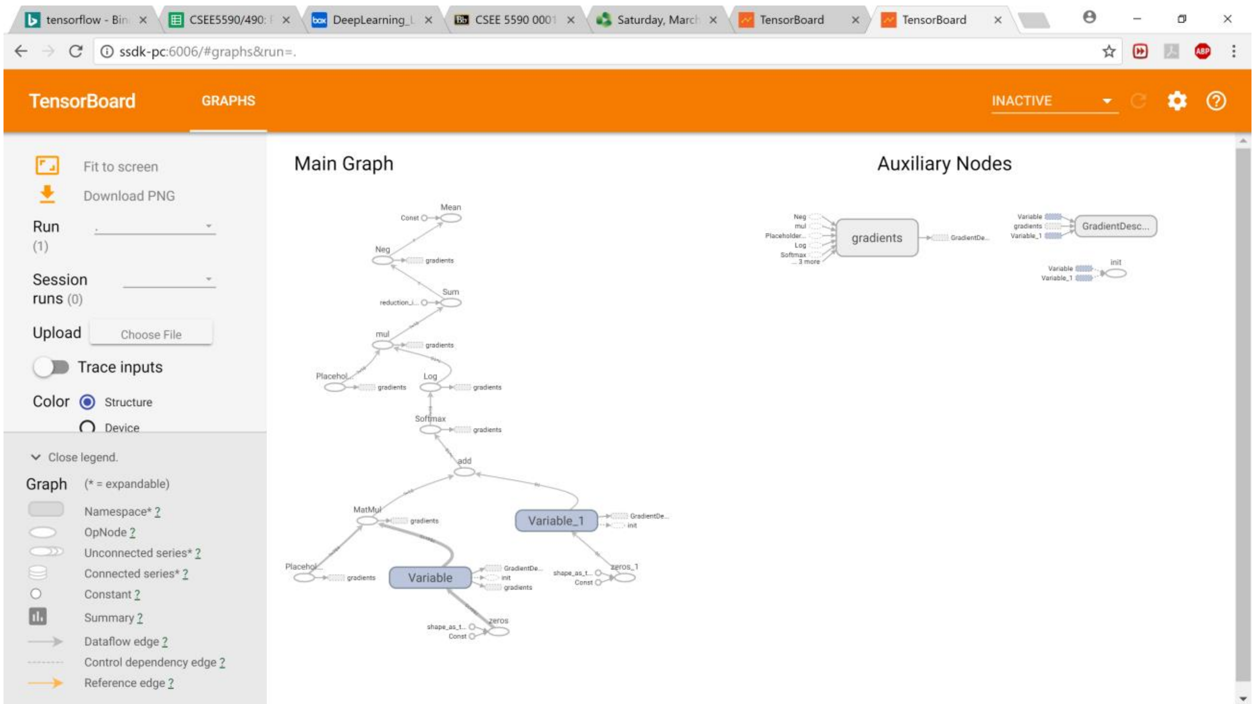
- Importing Data from Dataset
- Assigning X and Y Placeholders
- Variable Weights and Bias Collection
- Construction of prediction model
- Optimize model for less errors
- Train model for the training data
- Compare the prediction and actual model variables
- Compute the accuracy of the model

Parameters

Parameters set in the assignment are listed below:
Learning Rate = 0.01/0.1/1
Number of Training Epochs used = 150/100/50
Size of the batch = 10000/1000/100
Displaying step = 10

Workflow

The workflow of Logistic Regression on a Model performed on MNIST Dataset is shown in Tensor Board.

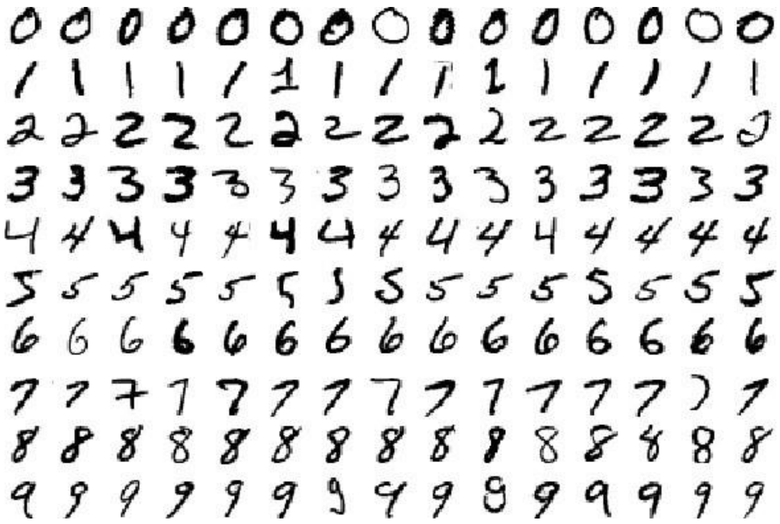


Dataset

The dataset used is ***MNIST Dataset***.

MNIST: Mixed National Institute of Standards and Technology Database.

- This dataset consists hand written digits ranging from 0 to 9.
- Every image is of 28 x 28 pixel flatten with one dimensional tensor size of 784 where label is provided for each.
- The datasets for certain values is given below, where images in first row are labeled 0, in second row labeled 1 and goes on.



Configuration

For executing the tasks given in the lab assignment, advanced version of **Python 3.6.4** is used and the code is built in **PYCHARM** Software.

Evaluation & Discussion

The code snippets are provided evaluating the logistic regression performed on MNIST dataset.

Firstly, tensor flow needs to be imported into the work space. The other import statements suppress the warnings raised while running tensor flow in unfavorable OS systems.

```
 import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as t_f
```

To read the data from datasets 'input_data' needs to be imported form tensor flow tutorials.

```
# Import new dataset not used in class : MNIST Dataset
from tensorflow.examples.tutorials.mnist import input_data
#saving the data
m = input_data.read_data_sets("", one_hot=True)
```

Now parameters are defined to define the model required for calculating the accuracy of the dataset. Learning rate, number of epochs, training batch size, displaying epoch size are the parameters set for the model.

```
# Parameters defined to train the data
l_r = 0.1  # learning rate for the dataset
```

```
t_e = 25     # number of training epochs used
b_s = 90     # batch size for training data
d_s = 1   # step size for displaying epochs
```

To display the output graph in Tensor Board, x and y variables are set with place holders for up to 784 data points and classifying 10 classes (0-9 digits).

```
# Tensor Flow Graph Input
# mnist data image of shape 28*28=784 in the database
x_var = t_f.placehlder(t_f.flat32, [None, 784])
# Variable needs to recognize 0-9 digits i.e, 10 classes
y_var = t_f.plceholder(t_f.flot32, [None, 10])
```

In the model, 'Weights' and 'Bias' are set as variable values which changes according to the data points taken from the set. The range is within the placeholder limits.

```
# Setting Model Weights for the Dataset
Wei = t_f.Variable(t_f.zros([784, 10]))
bi = t_f.Vriable(t_f.zeros([10]))
```

A prediction model is constructed using softmax. Cross entropy is used in reducing the mean error. An optimizer for Gradient Descent is used to reduce the error as much as possible which increases the accuracy of the model.

```
# Prediction Model Construction
prdt = t_f.nn.softmax(t_f.matmul(x_var, Wei) + bi)

# Cross entropy is used in reducing the error
cst = t_f.reduc_mean(-t_f.rduce_sum(y_var*t_f.log(prdt), reductin_indces=1))

# Gradient Descent
optm = t_f.train.GradintDescentOptimizer(l_r).minmize(cst)

# Initialize the variables (i.e. assign their default value)
st = t_f.globl_varibles_initilizer()
```

The dataset is trained iteratively within the session time for the number of epochs given specified during the parameters initialization.

```
# Training session begins here
with t_f.Sessin() as s:

    # Run the intializer
    s.run(st)
    writer = t_f.summry.FileWritr('./graphs/logistic_reg', s.graph)

   # Traning cycle for the specified range of epochs
    for ep in range(t_e):
        ag_ct = 0.
        t_b = int(m.trin.num_exampls/b_s) #total batch taken

        # Over all looping of the batch
        for i in range(t_b):
            batcch_xs, batchh_ys = m.trin.next_batch(b_s)

            #Running optimization
            _, c = s.run([optm, cst], fed_dict={x: batch_xs, y: batch_ys})
```

```
            # Average cost calculation
            ag_ct += c / t_b

        # Display logs per epoch step
        if (ep+1) % d_s == 0:

            print("Epoch:", '%04d' % (ep+1), "cost ", "{:.9f}".formt(ag_ct))

    print("Complete Optimization Done!")
```

A prediction correction is obtained comparing the predicted value and the y variable taken from the dataset. The accuracy is calculated by taking reduced mean between them.

```
 # Designed Test Model

    c_p = t_f.equal(t_f.argmax(prdt, 1), t_f.argmax(y, 1))
 #Accuracy Calculation for the developed epochs
    acc = t_f.reduce_mean(t_f.cast(c_p, t_f.float32))
    print("Accuracy of the MNIST Dataset:", acc.eval({x: m.test.images, y:
m.test.labels}))
```

Results

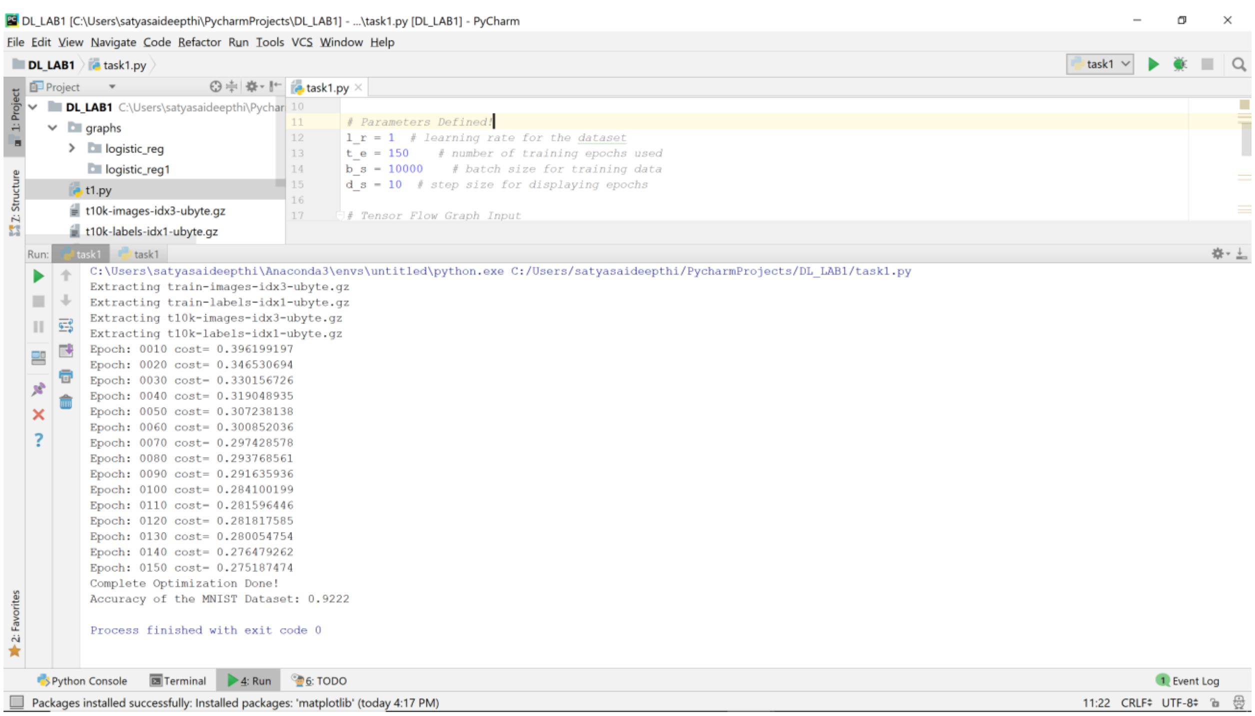
Case 1:

Changing Learning Rate

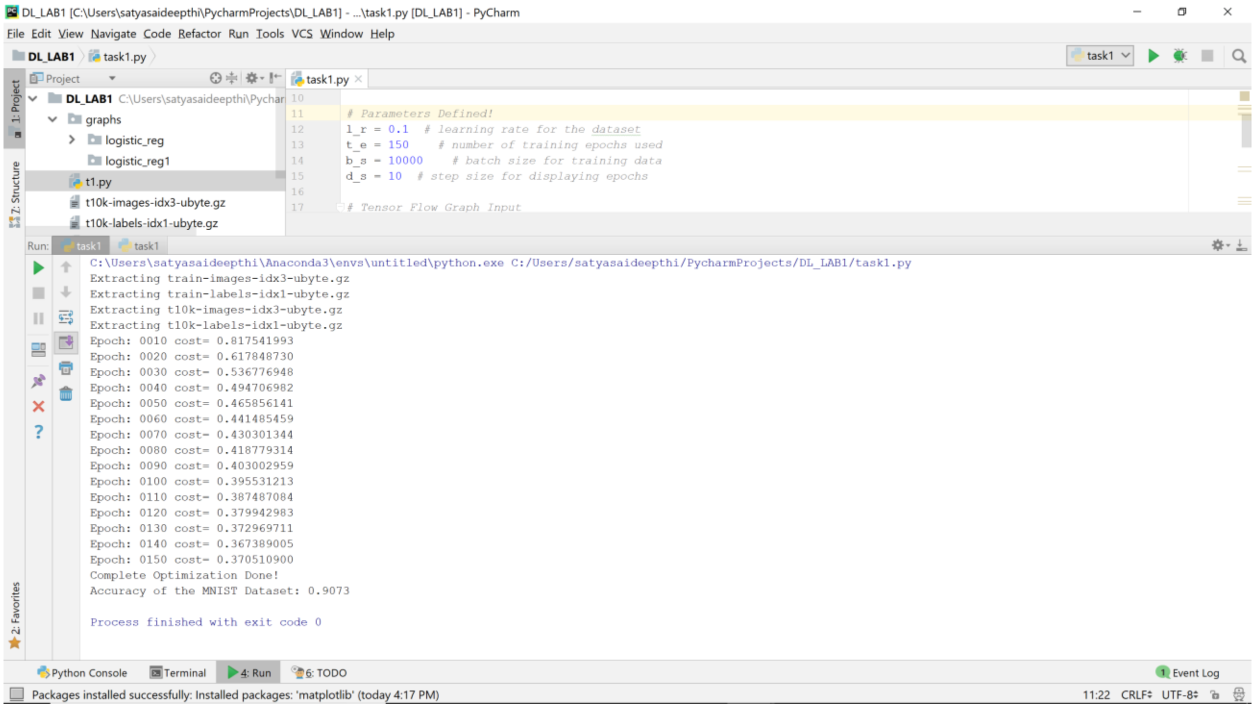
Logistic Regression of the MNIST Dataset considering parameters below

Training Epochs = 150; Batch size = 10000; constant display size

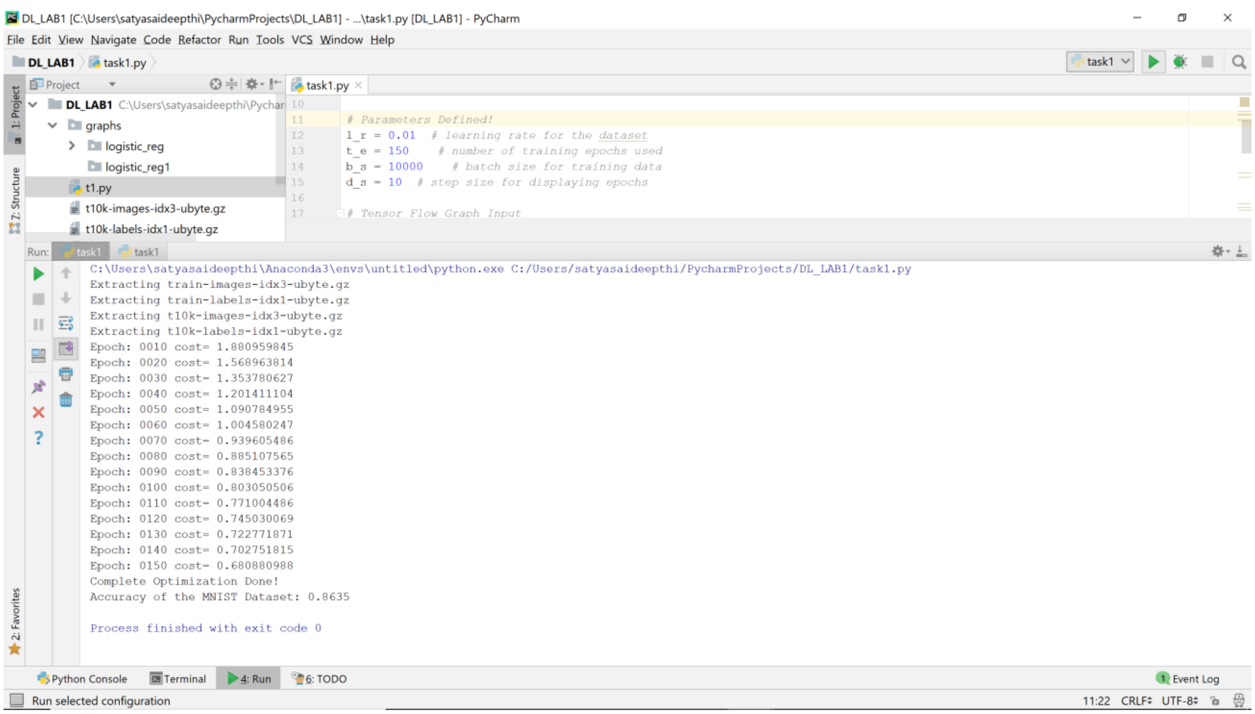
Learning Rate= 1



Learning Rate = 0.1



Learning Rate = 0.01



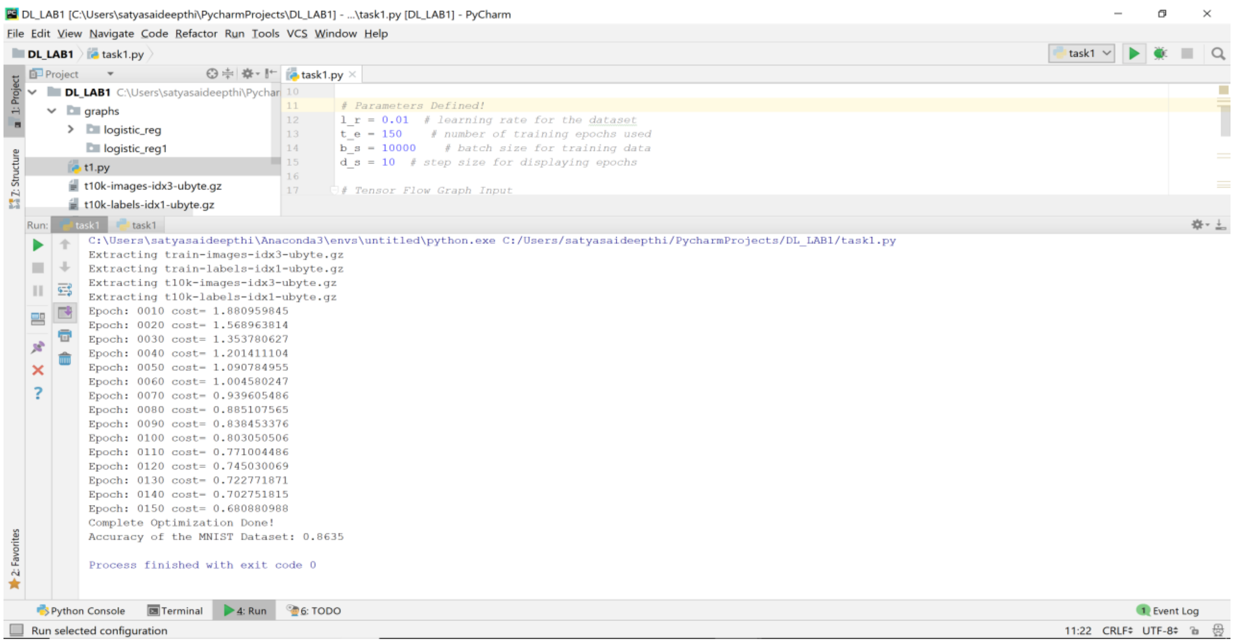
Case 2:

Changing Batch Size

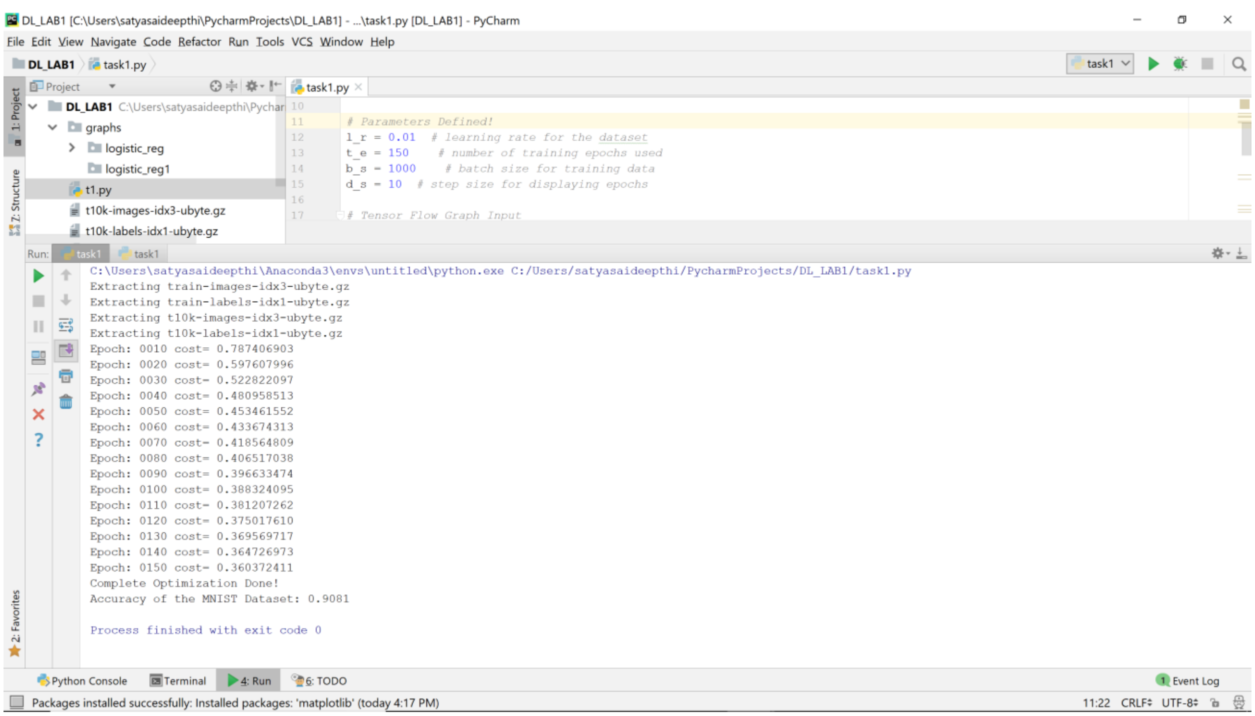
Logistic Regression of the MNIST Dataset considering parameters below

Training Epochs = 150; Learning Rate = 0.01; constant display size

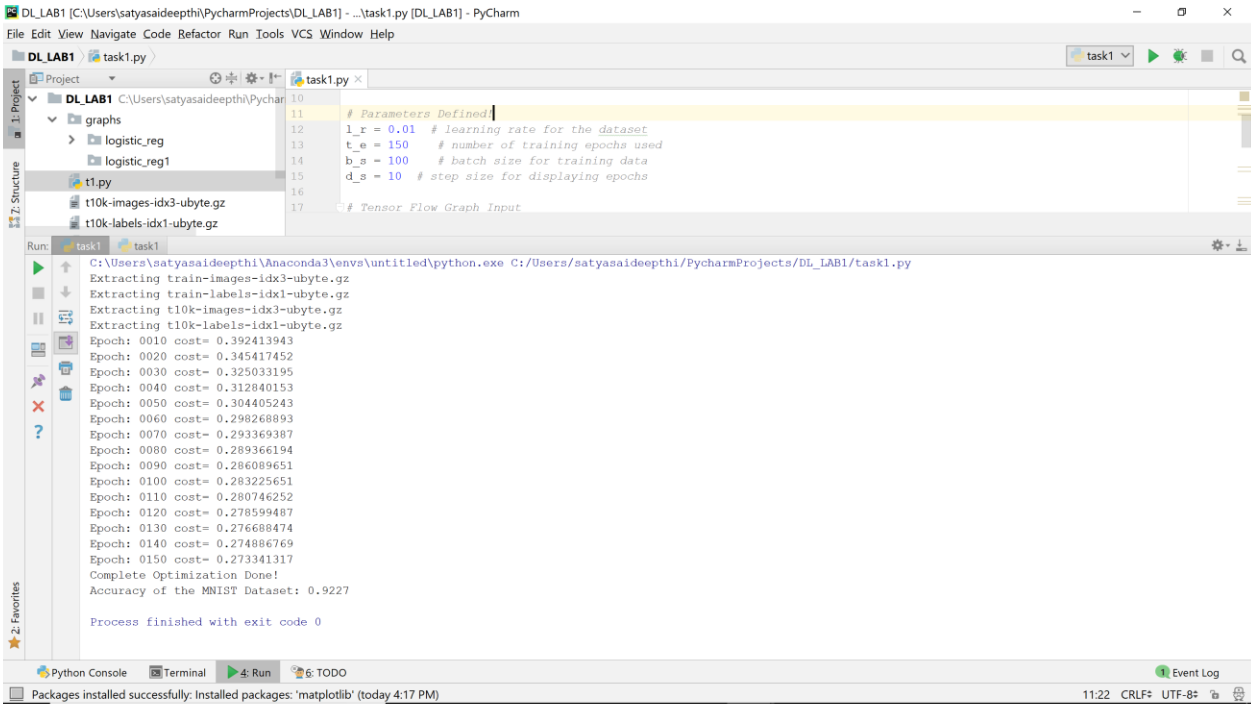
Batch Size = 10000



Batch size = 1000



Batch size = 100



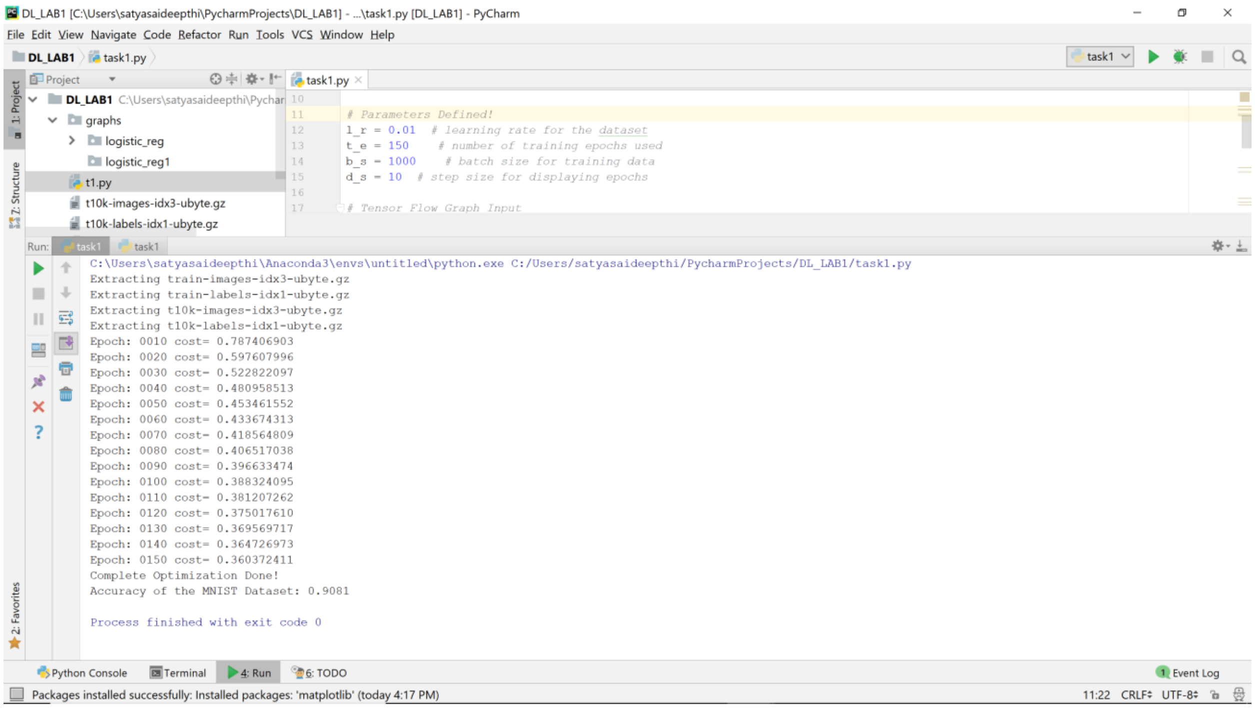
Case 3:

Changing Epoch Size

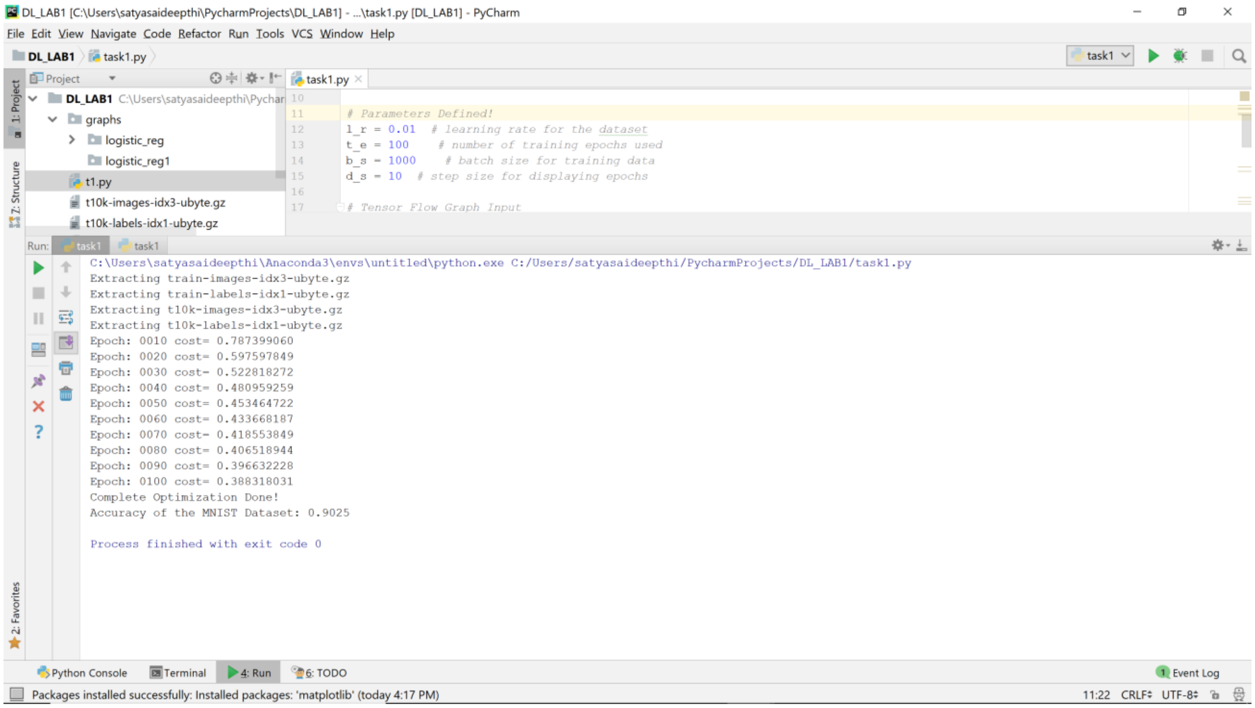
Logistic Regression of the MNIST Dataset considering parameters below

Learning Rate = 0.01; constant display size; Batch Size = 10000

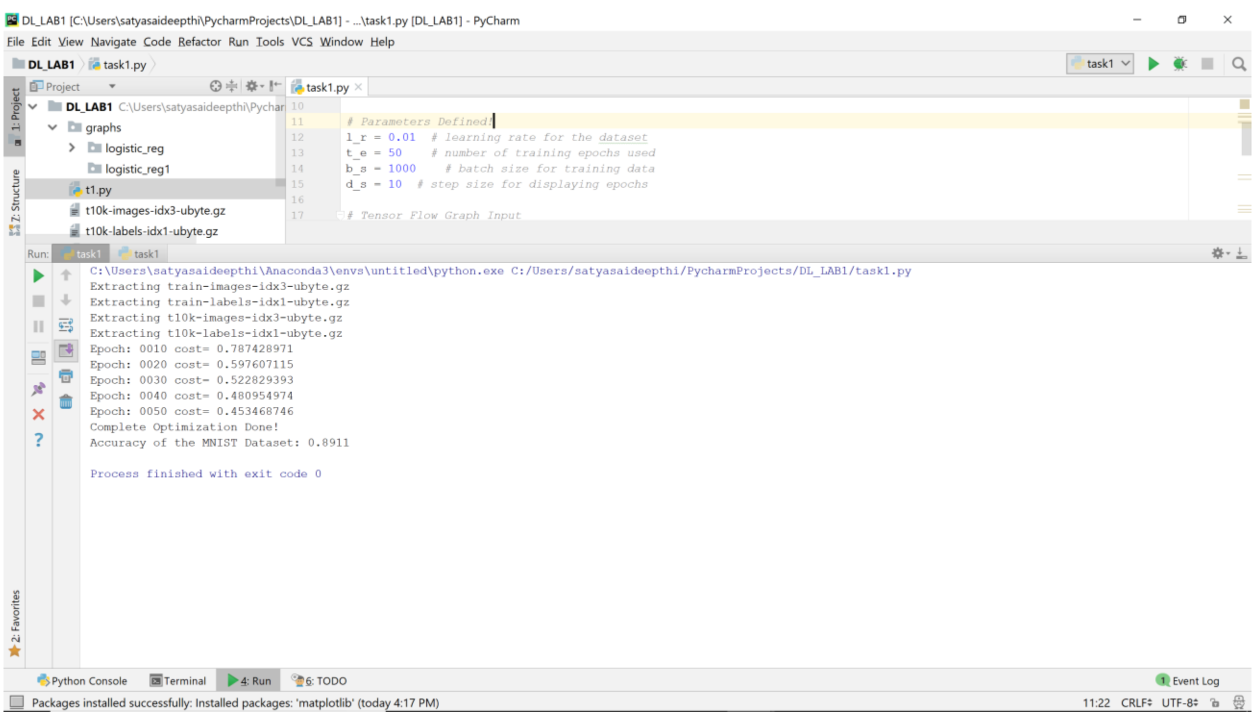
Epochs = 150



Epochs = 100



Epochs = 50



Conclusion

After performing Logistic Regression on MNIST Dataset, by changing hyper parameters the following conclusions are drawn from the results:

- With constant batch size and epoch number, as the learning rate increases the accuracy of the model increases.
- With constant learning rate and epoch number, as the batch size increases the accuracy of the model decreases.
- With constant learning rate and batch size, as the epoch number increases accuracy of the model increases.