Programming Homework 04 **Usama Munir Sheikh**
Introduction to Deep Learning **661541154**

1) Give the dimension for each layer, and the number of weight parameters for each convolutional layer and the output layer

Dimensions as set in tensorflow:

Input: data_set_size x 32 x 32 x 3
Labels: data_set_size

Weights:
$\boldsymbol{W^1}$ = 5 x 5 x 3 x 32
$\boldsymbol{W^1_0}$ = data_set_size x 28 x 28 x 32
$\boldsymbol{W^2}$ = 5 x 5 x 32 x 32
$\boldsymbol{W^2_0}$ = data_set_size x 10 x 10 x 32
$\boldsymbol{W^3}$ = 3 x 3 x 32 x 64
$\boldsymbol{W^3_0}$ = data_set_size x 3 x 3 x 64
$\boldsymbol{W^o}$ = 576 x 10
$\boldsymbol{W^o_0}$ = data_set_size x 10

Convolutional Layer 1: data_set_size x 28 x 28 x 32
Activation Layer 1: data_set_size x 28 x 28 x 32
Pooling Layer 1: data_set_size x 14 x 14 x 32

Convolutional Layer 2: data_set_size x 10 x 10 x 32
Activation Layer 2: data_set_size x 10 x 10 x 32
Pooling Layer 2: data_set_size x 5 x 5 x 32

Convolutional Layer 3: data_set_size x 3 x 3 x 64
Activation Layer 3: data_set_size x 3 x 3 x 64

Fully Connected Layer: data_set_size x 576
Output: data_set_size x 10

2) Implement in Tensorflow the back-propagation method to train the CNN. Initialize weights for all filters for the convolutional layers to small values and iteratively update them with appropriate learning rate until convergence. Plot the training and testing errors for each epoch. Save weights for all layers, as detailed in the attached instructions.

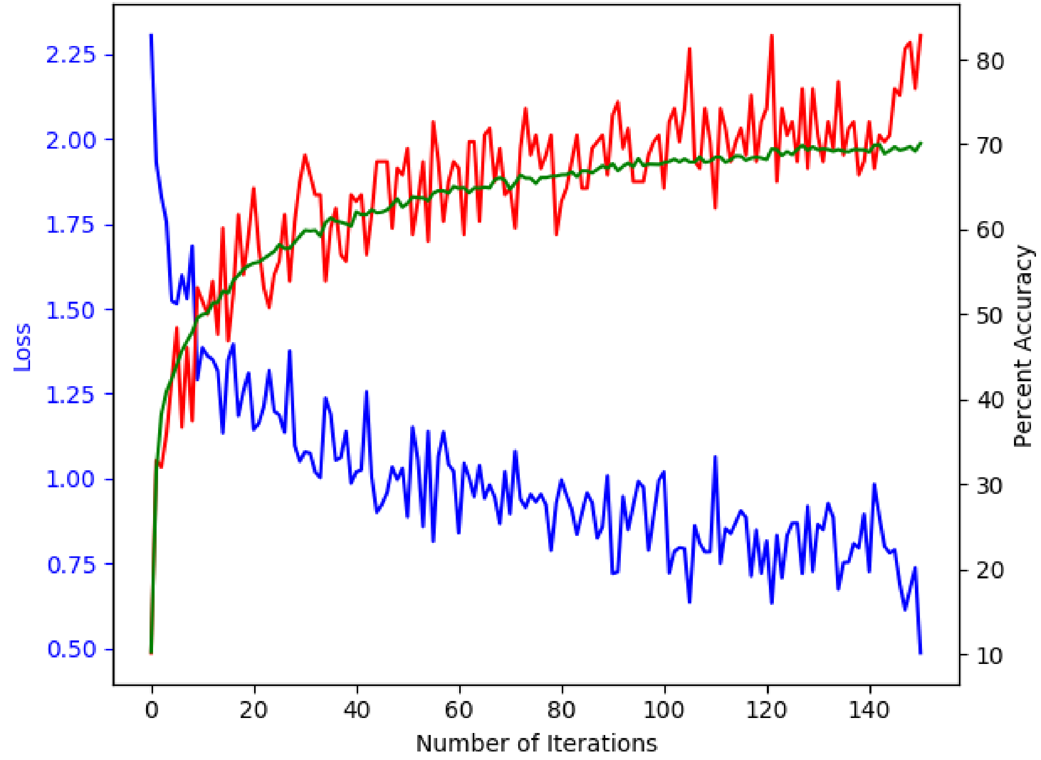
Implementation attached as
“pghw4_no_relu.py”

Plots:

Cost/Loss (Blue)**, Training Accuracy** (Red) **& Test Accuracy** (Green)**:**

***x* axis:** Number of iterations ÷ 200

∴**Total Number of Iterations:** 30,000

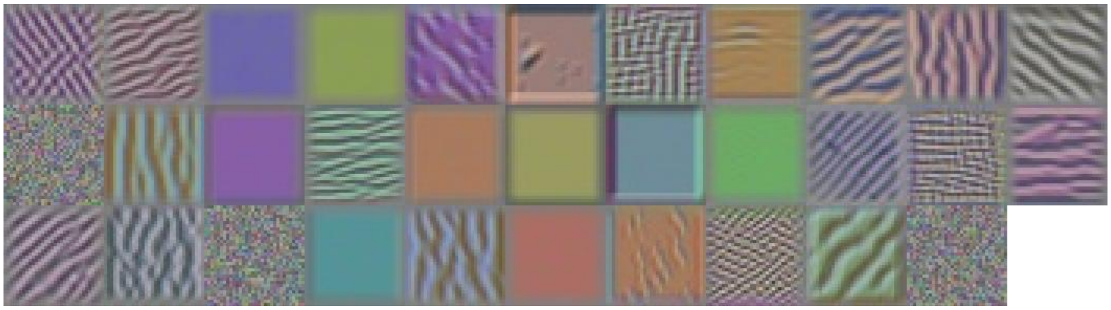


All four files required (**weights**, model etc.) are attached.

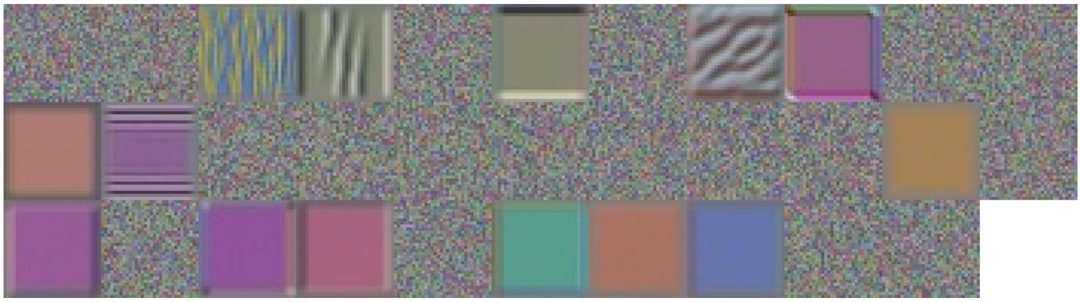
3) Visualize the learnt filters for the first convolution layer (see attached instructions).

Results with, both, before and after activation layers, respectively:

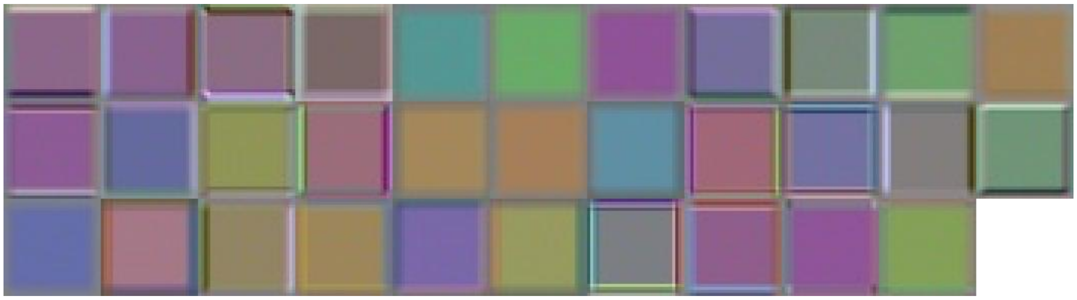
First Run, visualizing learnt filters by sending in the first convolutional layer after (ReLU) activation:



Second Run, visualizing learnt filters by sending in the first convolutional layer after (ReLU) activation but with noisy image + 100:



Last Run, visualizing learnt filters by sending in the first convolutional layer before activation:



4) Given the trained CNN, evaluate its performance on the testing dataset by computing its classification error for each class as well as the average classification errors for all 10 classes.

Final Cost: 0.4859

Final Average Training Accuracy: 82.8125
Final Average Test Accuracy: 70.1200

Final Average Training Error: 0.1788
Final Average Test Error: 0.2988

For classes $[0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9]$

Total Number of Images for Each Class in Test Data:

$\boldsymbol{T} = [488, \quad 505, \quad 512, \quad 497, \quad 507, \quad 488, \quad 491, \quad 495, \quad 504, \quad 513]$

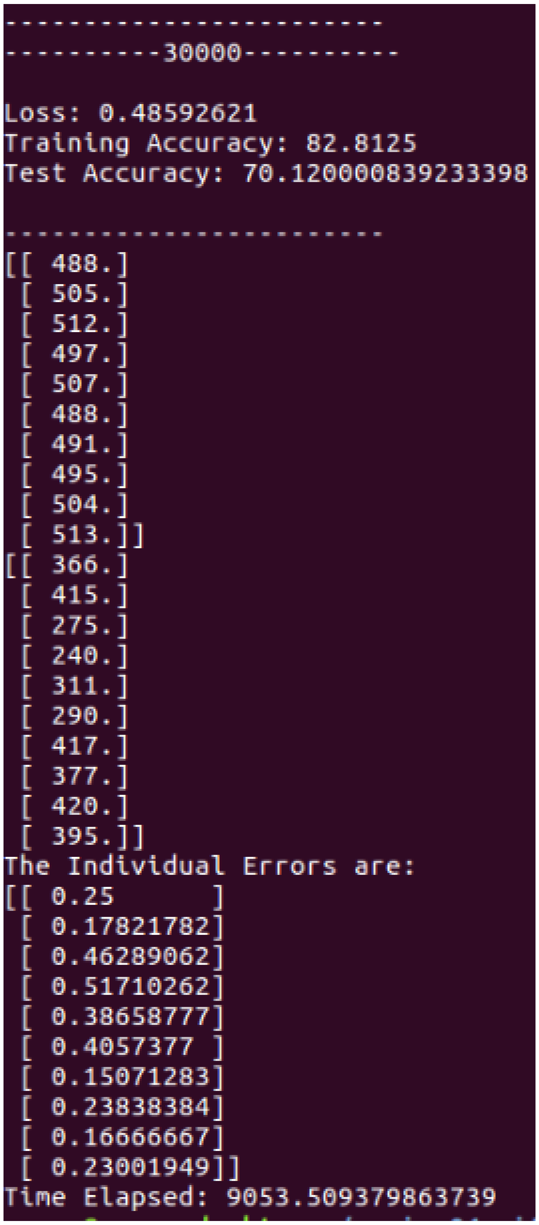
Number of Classification Images for Each Class in Test Data:

$\boldsymbol{I} = [488, \quad 505, \quad 512, \quad 497, \quad 507, \quad 488, \quad 491, \quad 495, \quad 504, \quad 513]$

Individual Classification Error for Each Class in Test Data:

Individual Error $= I \div T$

Individual Error: $[0.250,\ 0.178,\ 0.463,\ 0.517,\ 0.387,\ 0.406,\ 0.151,\ 0.238,\ 0.167,\ 0.230]$



```
------------------------
----------30000----------

Loss: 0.48592621
Training Accuracy: 82.8125
Test Accuracy: 70.120000839233398

------------------------
[[ 488.]
 [ 505.]
 [ 512.]
 [ 497.]
 [ 507.]
 [ 488.]
 [ 491.]
 [ 495.]
 [ 504.]
 [ 513.]]
[[ 366.]
 [ 415.]
 [ 275.]
 [ 240.]
 [ 311.]
 [ 290.]
 [ 417.]
 [ 377.]
 [ 420.]
 [ 395.]]
The Individual Errors are:
[[ 0.25      ]
 [ 0.17821782]
 [ 0.46289062]
 [ 0.51710262]
 [ 0.38658777]
 [ 0.4057377 ]
 [ 0.15071283]
 [ 0.23838384]
 [ 0.16666667]
 [ 0.23001949]]
Time Elapsed: 9053.509379863739
```