Oregon State University

CS 535 Deep Learning
Assignment 2

Mohamad Hosein Danesh
ID: 933619683

February 13, 2019

1) The function that evaluates the trained model is implemented in the *MLP* class, defined as below:

```
def evaluate(self, x, y, batch_size, l2_penalty)
```

In this function, *x* is the input to the network. *y* is the output of the network. *batch_size* is the batch size of data given to the network at once. *l2_penalty* is the l2 regularization term.

The functions that compute the subgradients using backpropagation are implemented in *SigmoidCrossEntropy*, *ReLU*, and *LinearTransformation* classes. Each backward function has its own definition with respect to the class they are in. Below, every one of them will be shown:

```
class SigmoidCrossEntropy(object):
  def backward(self, y, grad_output)

class ReLU(object):
  def backward(self, grad_output)

class LinearTransform(object):
  def backward(self, grad_output)
```

2) In the function *train()* which is defined in the *MLP* class, the weight and biases will be updated after each iteration of mini-batches. Also, momentum has been used during weights and biases update.

3) The best performance I got was as following:

```
[EPOCH 100/100]     Train Accuracy: 96.228 Train Loss: 0.375      Validation Accuracy: 81.232
Validation Loss: 2.267
Test Accuracy: 82.448   Test Loss: 2.128
```

To achieve this, I used the following hyperparameters:

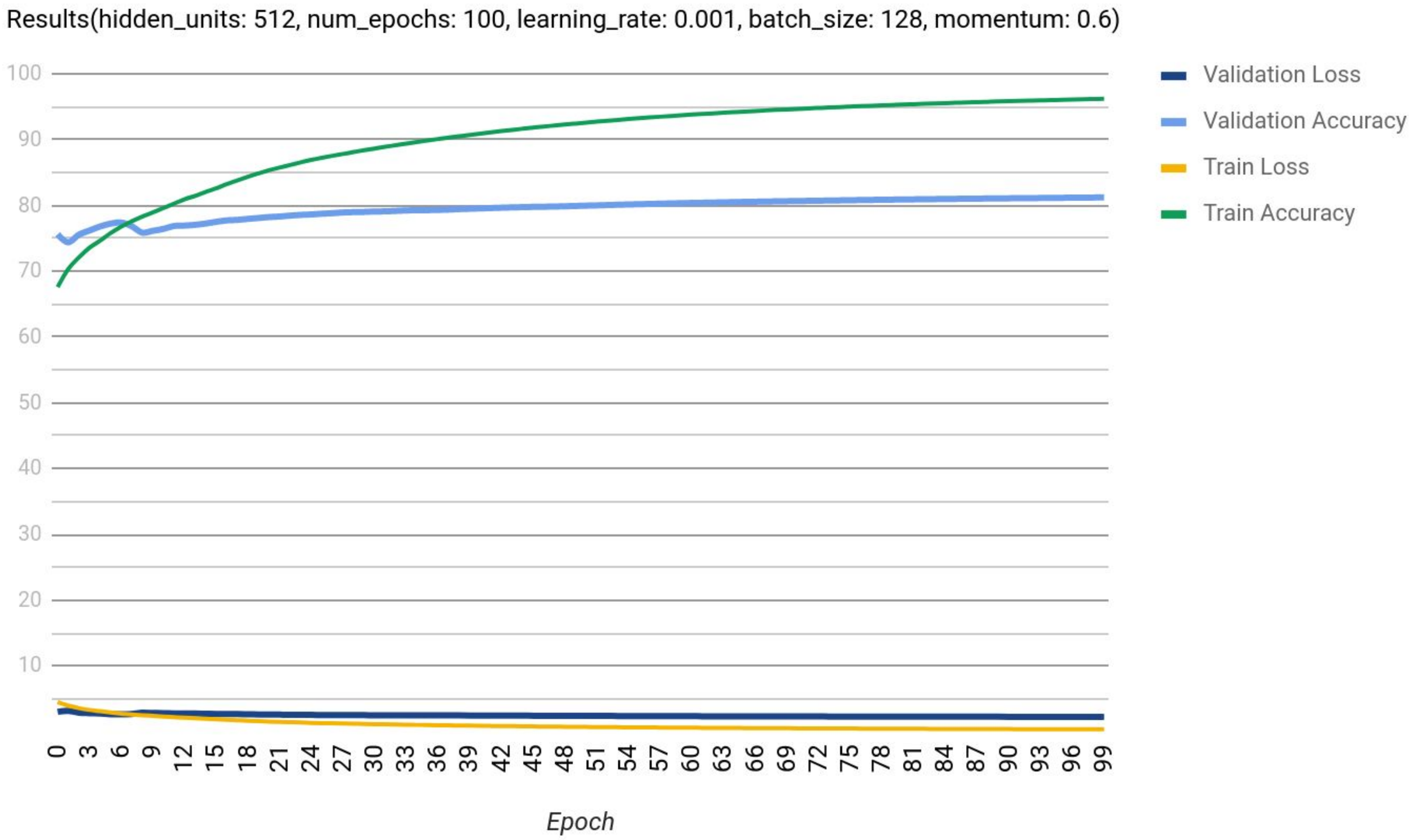
hidden_units: 512
num_epochs: 100
learning_rate: 0.001
batch_size: 128
momentum: 0.6

4) With respect to the hyperparameters in part 3, the requested values are printed as follows(outputs from epoch 11 to 90 are omitted):

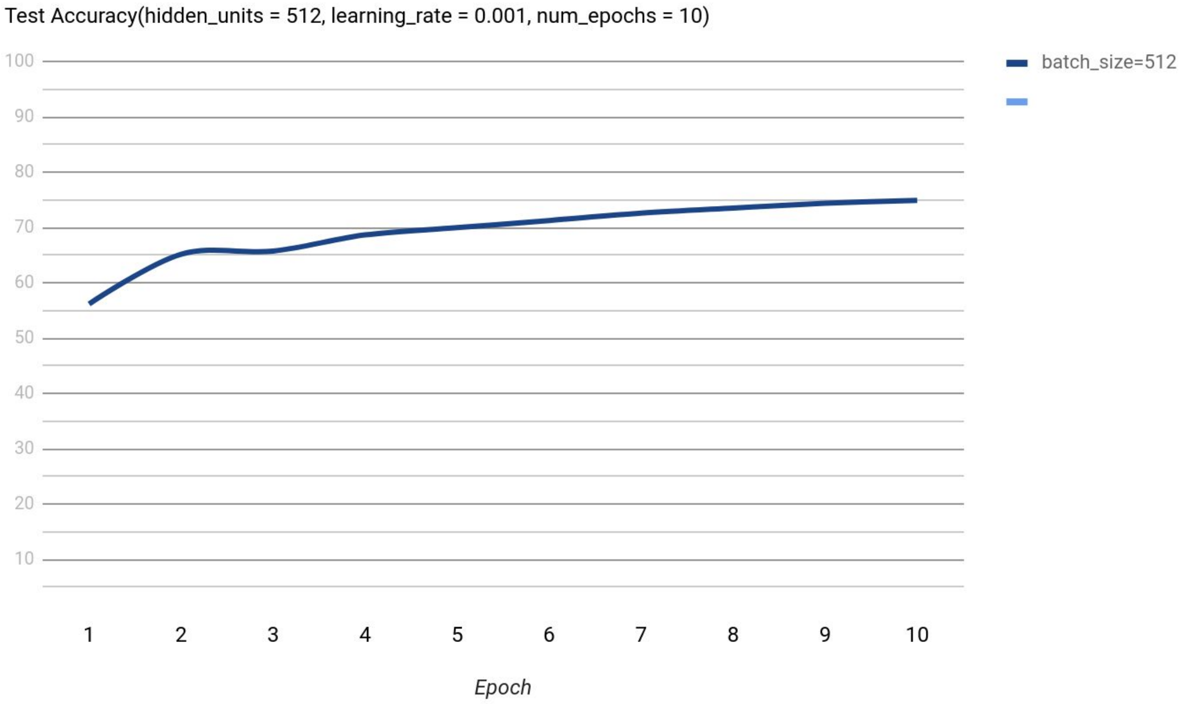
```
[EPOCH 1/100]        Train Accuracy: 67.598 Train Loss: 4.526        Validation Accuracy: 75.625
Validation Loss: 3.036
[EPOCH 2/100]        Train Accuracy: 70.333 Train Loss: 3.98         Validation Accuracy: 74.401
Validation Loss: 3.196
[EPOCH 3/100]        Train Accuracy: 72.102 Train Loss: 3.604        Validation Accuracy: 75.556
Validation Loss: 2.9
[EPOCH 4/100]        Train Accuracy: 74.639 Train Loss: 3.129        Validation Accuracy: 76.802
Validation Loss: 2.767
[EPOCH 5/100]        Train Accuracy: 75.796 Train Loss: 2.935        Validation Accuracy: 77.24
Validation Loss: 2.679
[EPOCH 6/100]        Train Accuracy: 76.787 Train Loss: 2.76         Validation Accuracy: 77.418
Validation Loss: 2.644
[EPOCH 7/100]        Train Accuracy: 77.587 Train Loss: 2.631        Validation Accuracy: 76.855
Validation Loss: 2.719
[EPOCH 8/100]        Train Accuracy: 78.312 Train Loss: 2.516        Validation Accuracy: 75.874
Validation Loss: 2.886
[EPOCH 9/100]        Train Accuracy: 78.933 Train Loss: 2.419        Validation Accuracy: 76.151
Validation Loss: 2.853
[EPOCH 10/100]       Train Accuracy: 79.598 Train Loss: 2.322        Validation Accuracy: 76.444
Validation Loss: 2.825
.
. (omitted outputs)
.
[EPOCH 90/100]       Train Accuracy: 95.854 Train Loss: 0.412        Validation Accuracy: 81.096
Validation Loss: 2.282
[EPOCH 91/100]       Train Accuracy: 95.9   Train Loss: 0.408        Validation Accuracy: 81.112
Validation Loss: 2.28
[EPOCH 92/100]       Train Accuracy: 95.944 Train Loss: 0.403        Validation Accuracy: 81.125
Validation Loss: 2.279
[EPOCH 93/100]       Train Accuracy: 95.987 Train Loss: 0.399        Validation Accuracy: 81.14
Validation Loss: 2.277
[EPOCH 94/100]       Train Accuracy: 96.029 Train Loss: 0.395        Validation Accuracy: 81.157
Validation Loss: 2.275
[EPOCH 95/100]       Train Accuracy: 96.07  Train Loss: 0.391        Validation Accuracy: 81.173
Validation Loss: 2.274
[EPOCH 96/100]       Train Accuracy: 96.111 Train Loss: 0.387        Validation Accuracy: 81.186
Validation Loss: 2.272
[EPOCH 97/100]       Train Accuracy: 96.151 Train Loss: 0.383        Validation Accuracy: 81.2
Validation Loss: 2.27
```

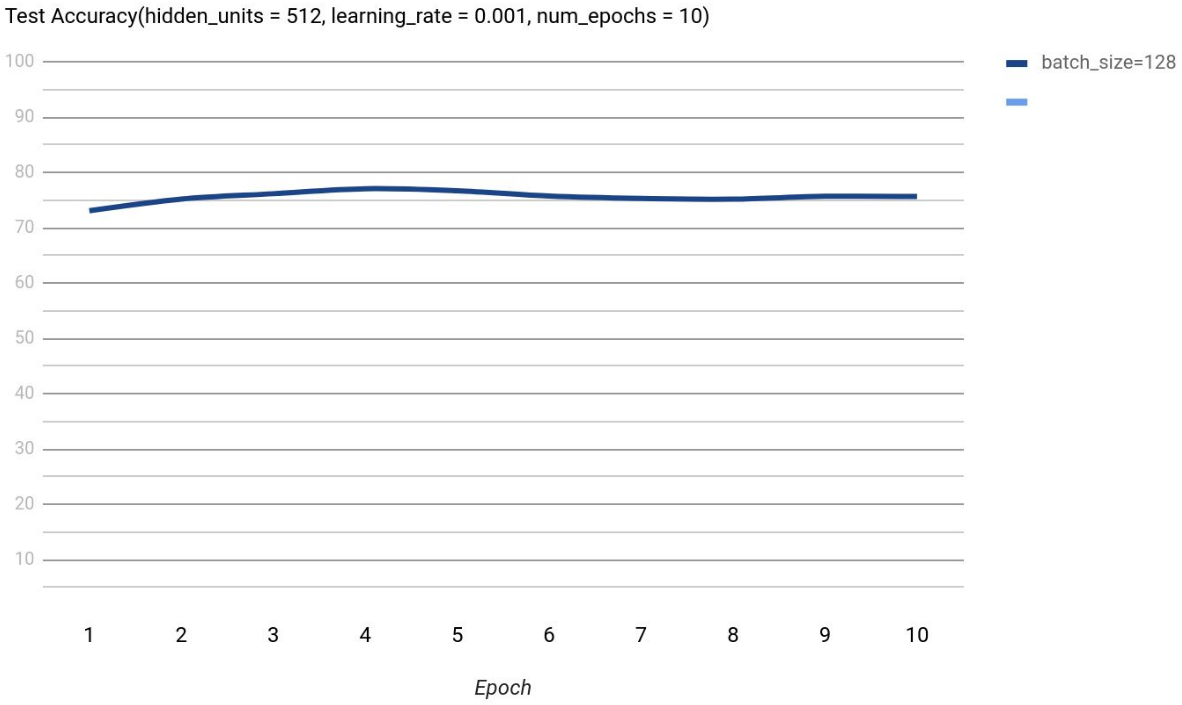
```
[EPOCH 98/100]      Train Accuracy: 96.189 Train Loss: 0.379      Validation Accuracy: 81.215
Validation Loss: 2.269
[EPOCH 99/100]      Train Accuracy: 96.228 Train Loss: 0.375      Validation Accuracy: 81.232
Validation Loss: 2.267
Test Accuracy: 82.448   Test Loss: 2.128
```

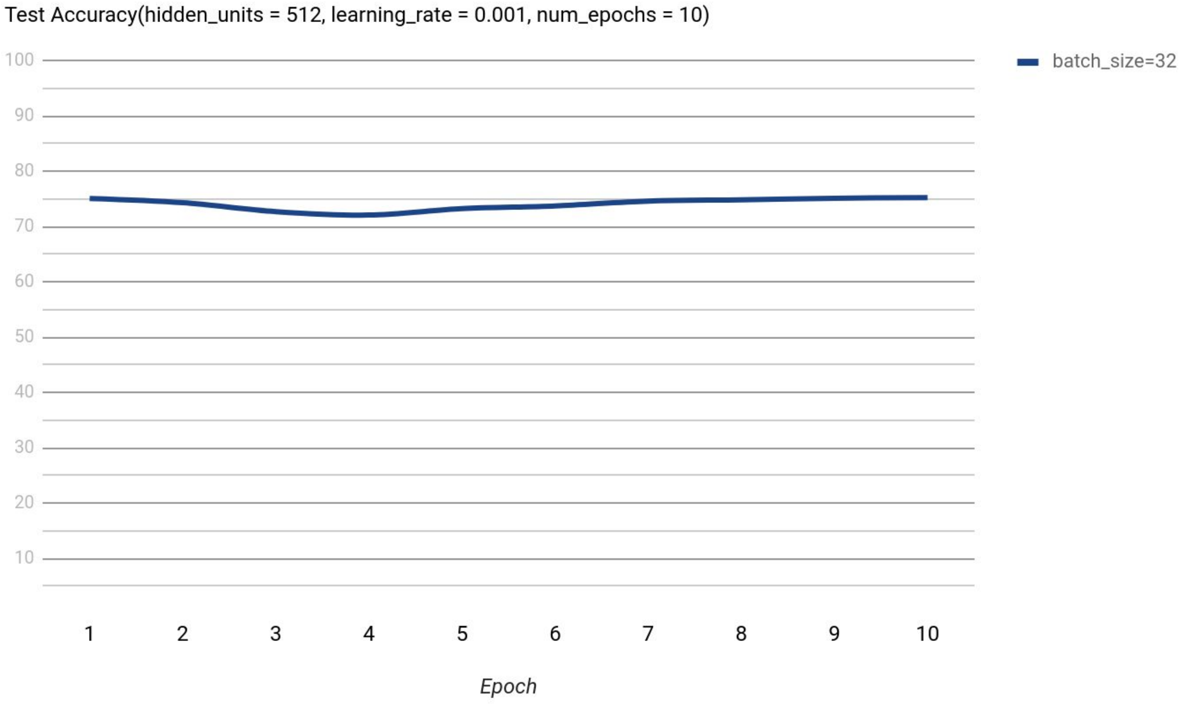
Also there are diagrams of the output values(all the outputs are shown in diagrams):



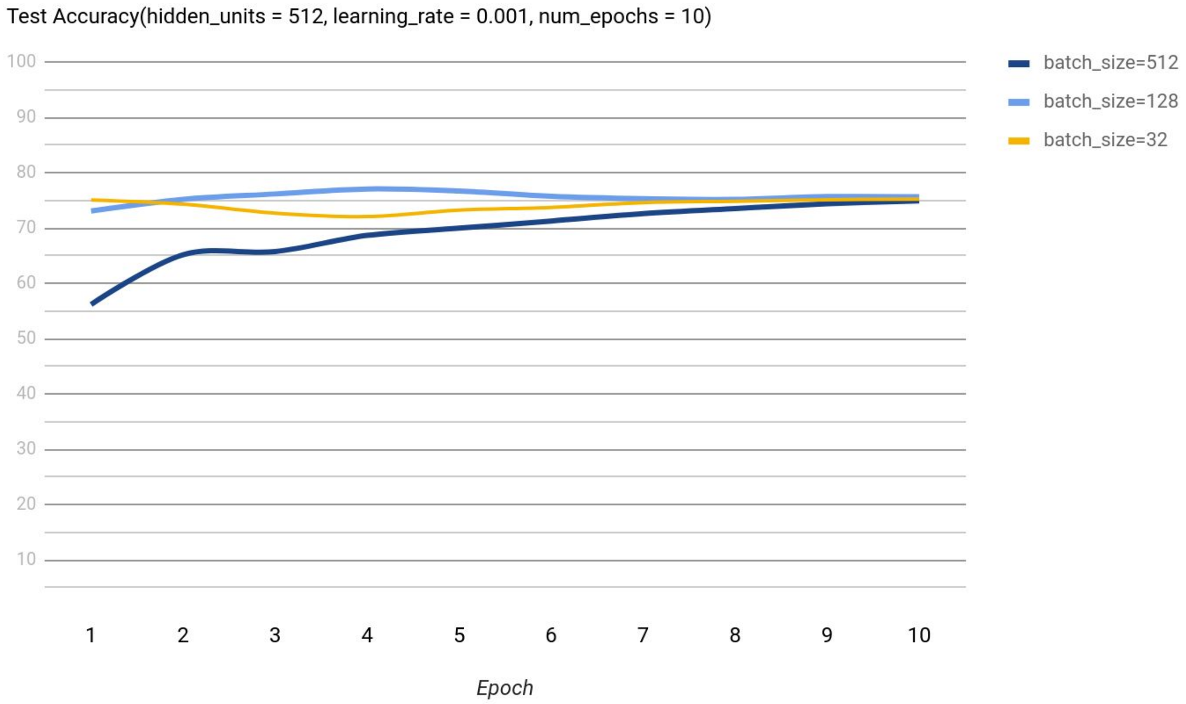
5) Test accuracy for trying different batch sizes:



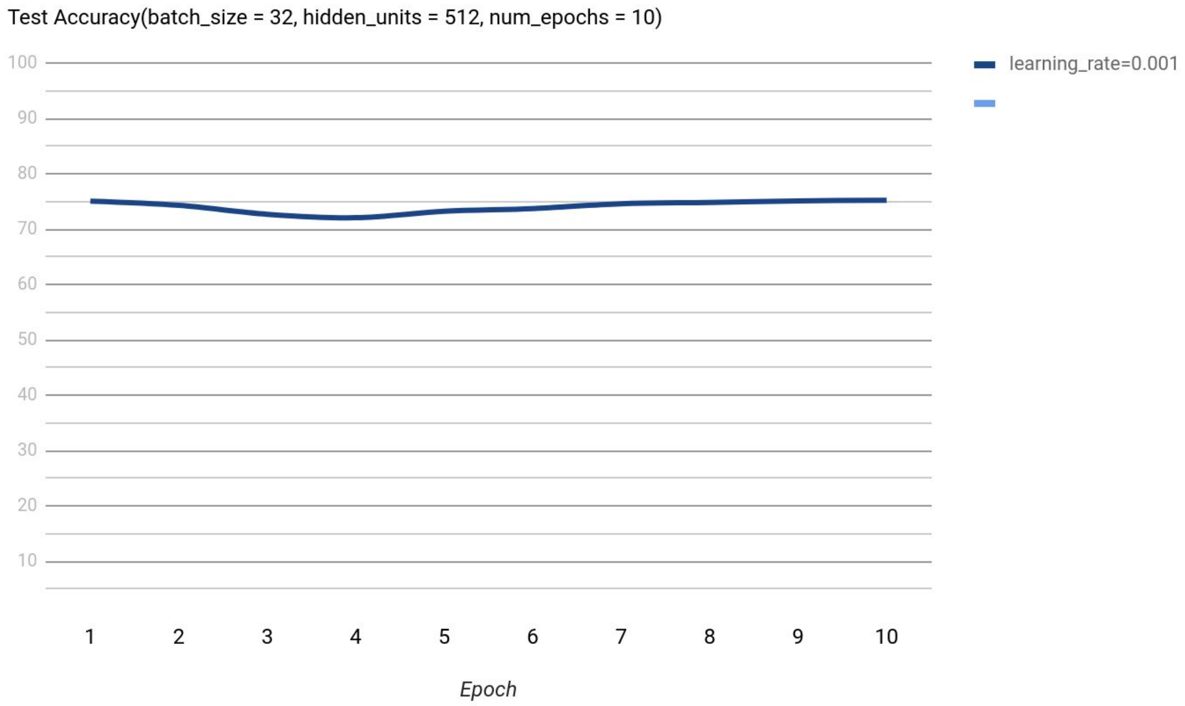


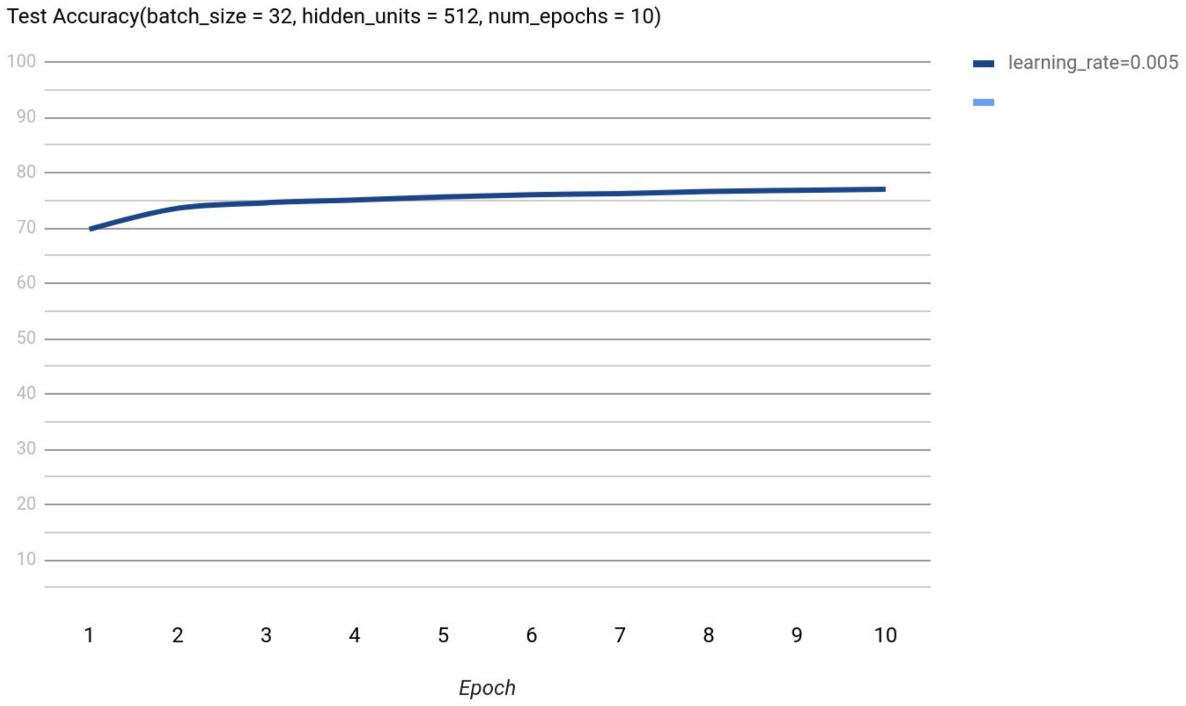


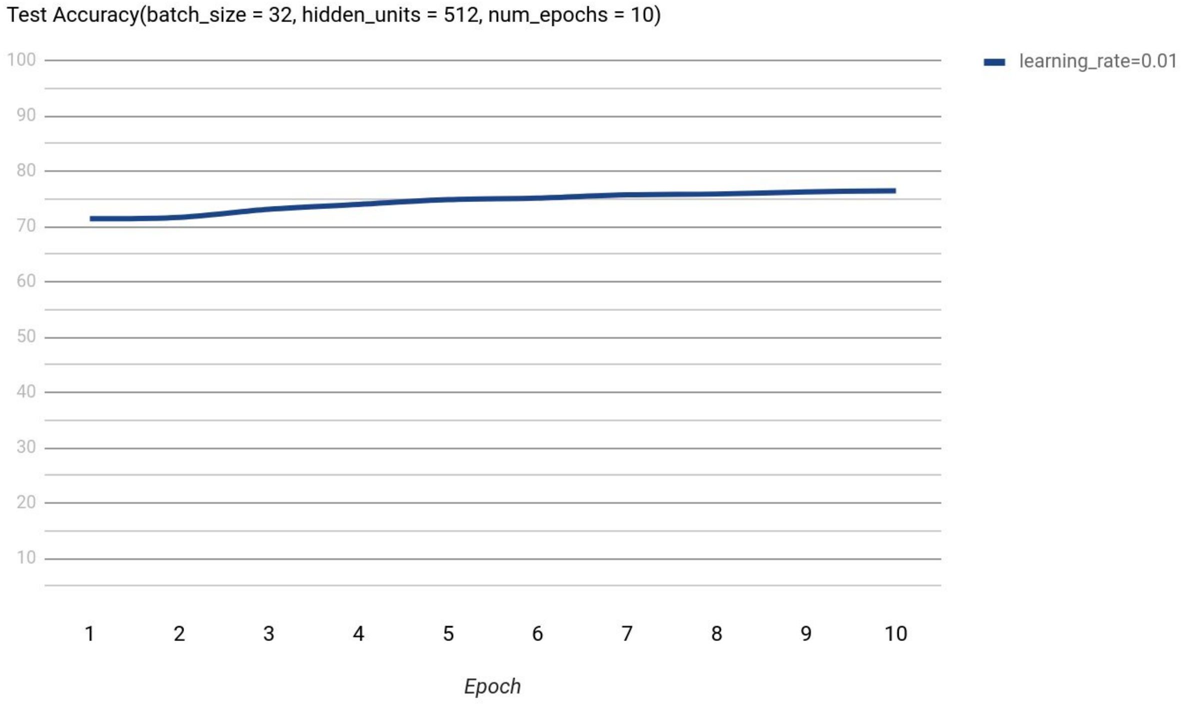
And by putting these three diagram into one we will have the following diagram:


Test Accuracy(hidden_units = 512, learning_rate = 0.001, num_epochs = 10)
batch_size=512
batch_size=128
batch_size=32
100
90
80
70
60
50
40
30
20
10
1
2
3
4
5
6
7
8
9
10
Epoch

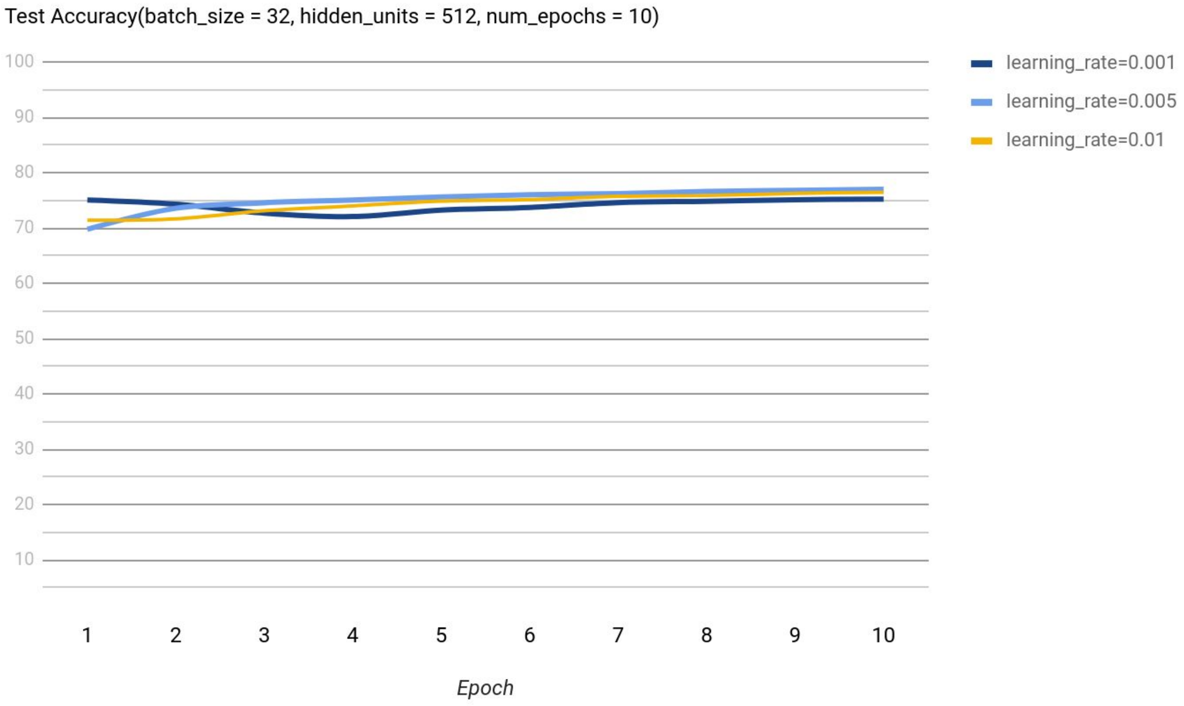
Test accuracy for trying different learning rates:





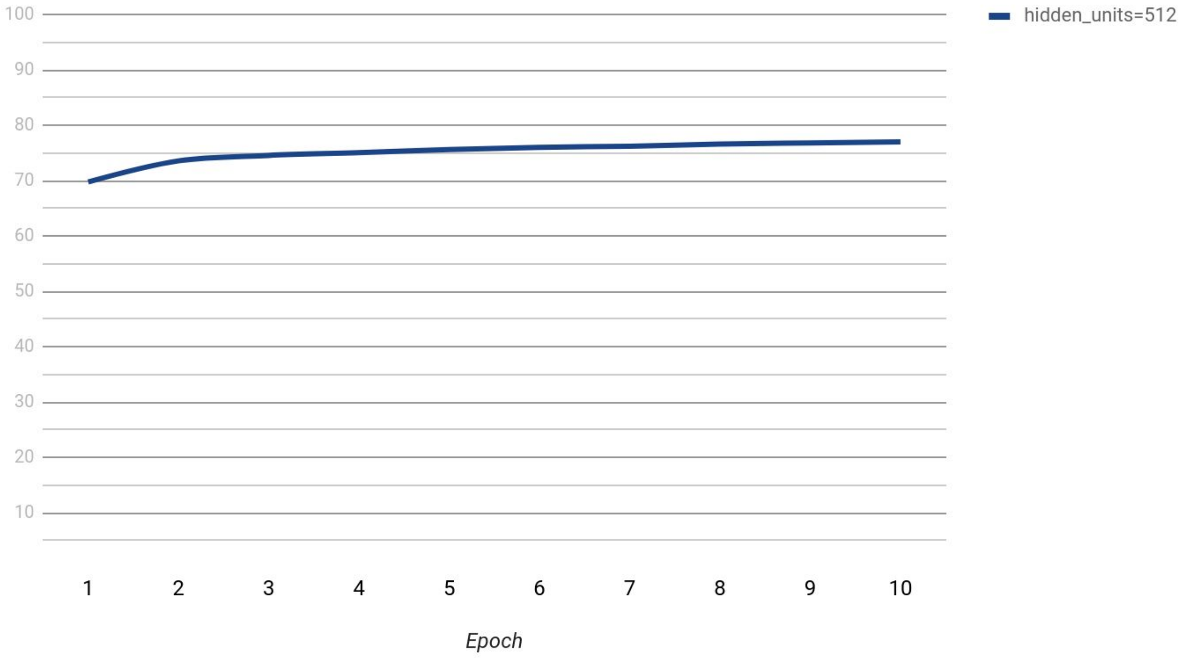


And by putting these three diagram into one we will have the following diagram:

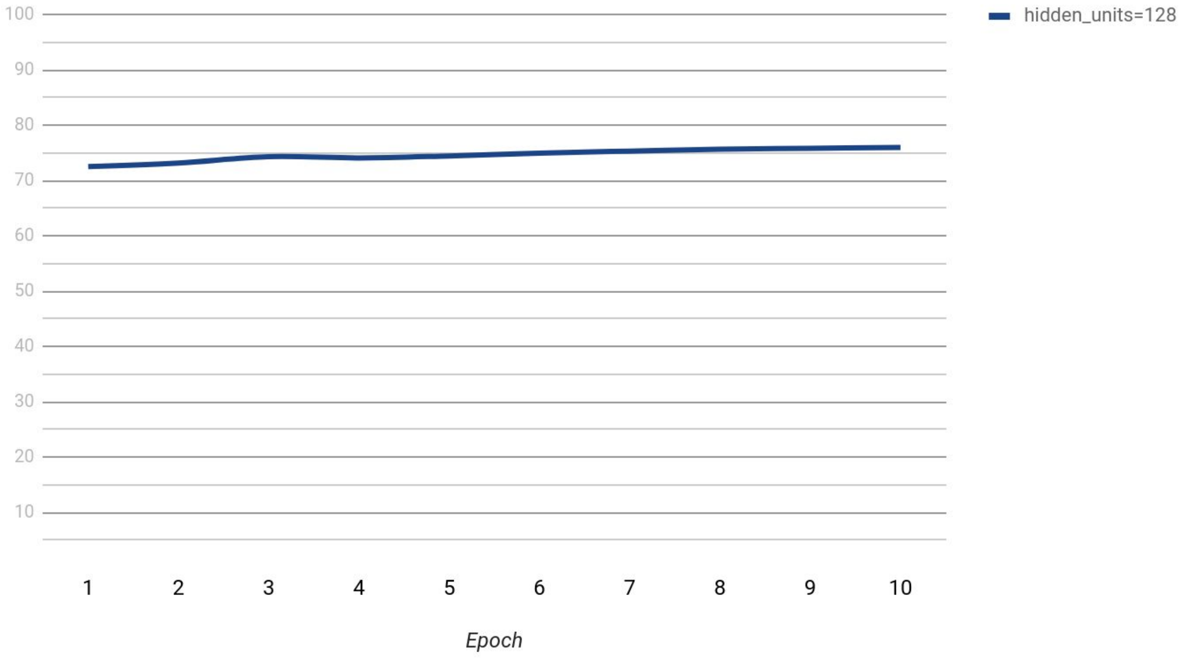


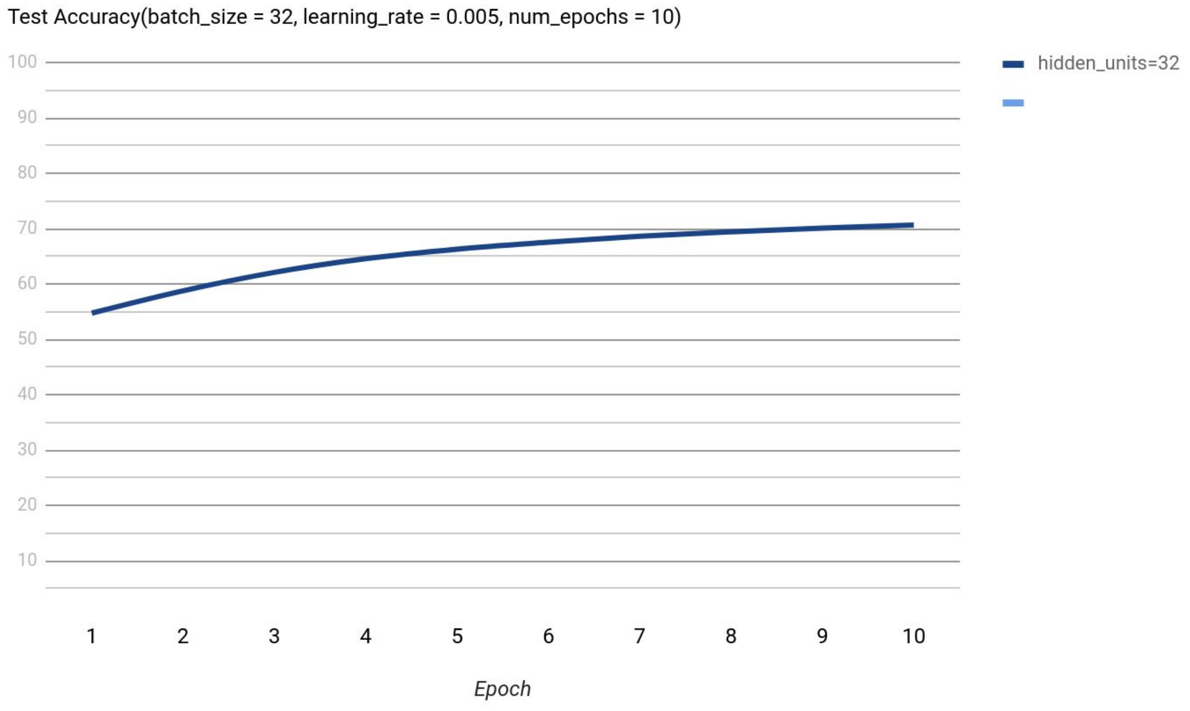
Test accuracy for trying different hidden units:

Test Accuracy(batch_size = 32, learning_rate = 0.005, num_epochs = 10)

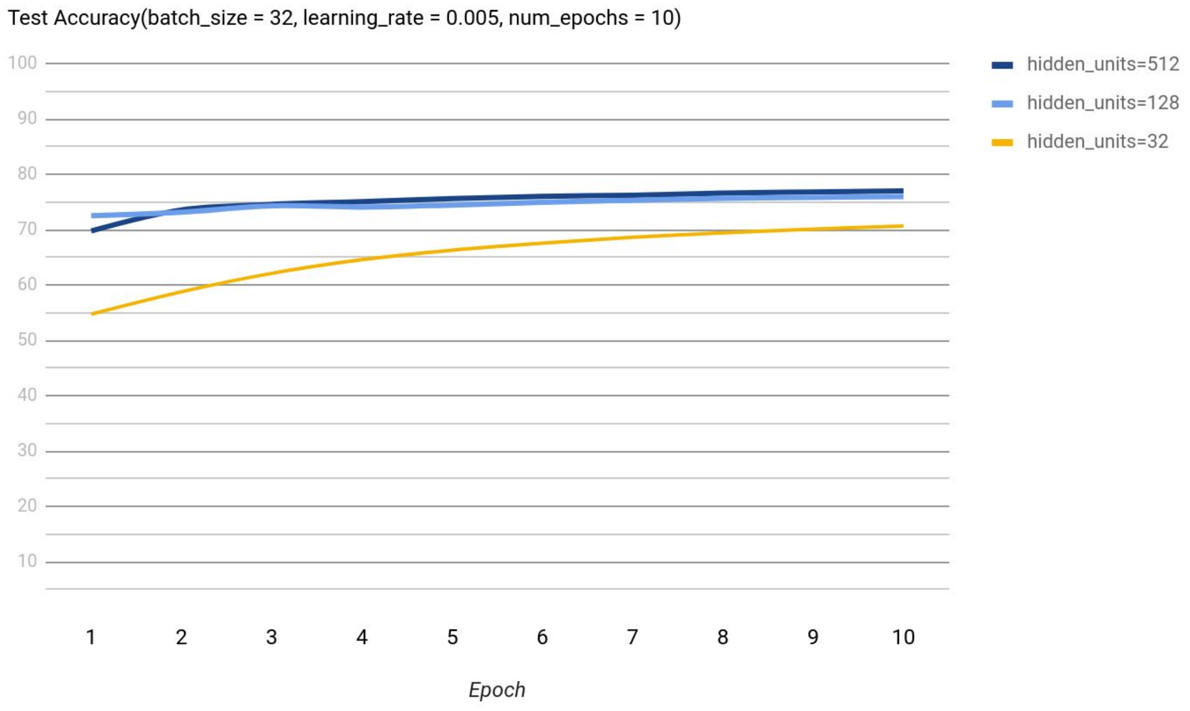


Test Accuracy(batch_size = 32, learning_rate = 0.005, num_epochs = 10)





And by putting these three diagram into one we will have the following diagram:



6) Although it is not a convolutional neural network and has only one hidden layer, it performs reasonably well. At final epochs, it gets around 96% accuracy in the training set. But it only gets around 82% accuracy in the test set. In my opinion that is because the model is overfitted over the training data.

According to the experiments done in the part 5 of the assignment, different batch size does not have much impact on the test accuracy, eventually. Nevertheless, the batch size equal to 128 got the most test accuracy among different values. They all got closed to the 75% accuracy.

Different learning rates also have little impact on the final accuracy. As it is obvious from the diagrams of part 5, the three different settings all got to about 76% accuracy. Although learning rate equals to 0.005 got the most accuracy among others. Regarding the number of hidden units, according to experiments run on part 5, it has the most impact on the model's accuracy. The bigger the hidden layer, the more accurate it gets. It got to around 77% accuracy with 512 hidden neurons.