CS 545: Machine Learning Spring 2020

Name: *Sree Vandana Nadipalli*

Neural Network Structure

MNIST dataset is used for this task. The neural network used has 785 input units, one hidden layer with 'n' units and 10 output units, each corresponding to classes (0-9) and the network is fully connected. The bias values of both input and hidden layer is considered as '1'. The weights of each connection are randomly chosen between the intervals (-0.05, 0.05).

Here Back-Propagation with Stochastic Gradient Descent is used to train the network, with a learning rate 0.1 and momentum 0.9

Experiment 1: Vary number of hidden units

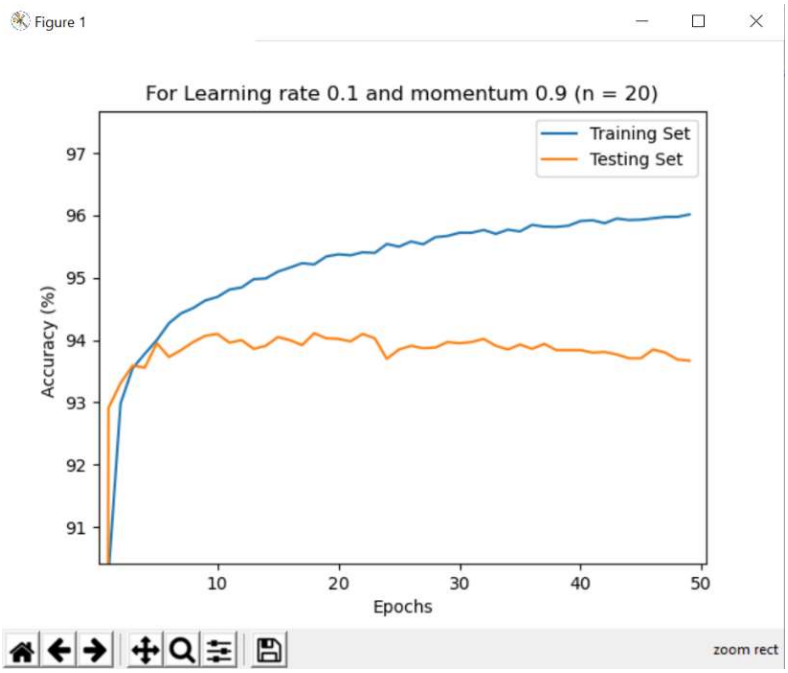
Here the parameter 'n', which indicates number of hidden units is varied. The experiment is done with n= 20, 50, 100. The network is trained and the weights are updated and accuracy is calculated after each epoch and plotted on an accuracy graph after 50 epochs.

***Case 1**: n = 20; Learning Rate 0.1; Momentum 0.9*
For training set

```
[[ 955    0    3    6    2    3    6    2    2    1]
 [   0 1114    5    4    0    1    5    1    5    0]
 [   6    5  953   15    9    0    7   10   22    5]
 [   2    0   16  959    0    2    1    7   20    3]
 [   1    1    3    3  929    0    8    0    5   32]
 [   2    2    2   38    7  785   25    1   18   12]
 [   8    2    7    2    9   15  908    0    7    0]
 [   1    7   22   16    9    3    2  945    5   18]
 [   8    9    5   14    7   10   16    2  899    4]
 [   3    9    1   22   32    2    0    5   15  920]]
```

```
learning rate = 0.1;  momentum = 0.9; hidden units = 20
Training Accuracy
[10.44166667,90.225     ,92.98333333,93.55      ,93.78      ,94.00166667,
 94.27166667,94.43      ,94.515     ,94.63666667,94.69333333,94.81166667,
 94.845     ,94.97833333,94.99      ,95.1       ,95.165     ,95.235     ,
 95.215     ,95.34333333,95.37666667,95.36166667,95.41      ,95.4       ,
 95.54333333,95.5       ,95.58333333,95.53833333,95.65166667,95.67166667,
 95.72333333,95.72333333,95.76833333,95.705     ,95.77333333,95.745     ,
 95.85      ,95.82166667,95.81666667,95.83666667,95.91166667,95.92333333,
 95.87666667,95.95166667,95.92666667,95.93333333,95.955     ,95.97666667,
 95.97833333,96.01666667]
```

```
Testing Accuracy
[10.28,92.91,93.31,93.59,93.56,93.95,93.73,93.84,93.97,94.07,94.1 ,93.96,
 94.  ,93.86,93.91,94.05,94.  ,93.92,94.11,94.03,94.02,93.98,94.1 ,94.03,
 93.7 ,93.85,93.91,93.87,93.88,93.97,93.95,93.97,94.02,93.91,93.85,93.93,
 93.86,93.94,93.84,93.84,93.84,93.8 ,93.81,93.77,93.71,93.71,93.85,93.8 ,
 93.69,93.67]
```

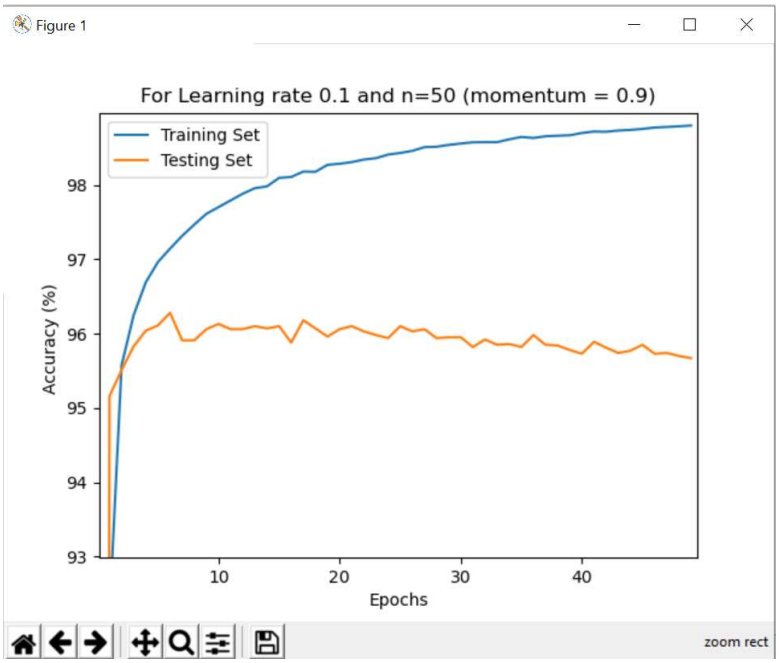


***Case 2**: n = 50; Learning Rate 0.1; Momentum 0.9*

For training set

```
[[ 961    0    4    0    1    1    5    2    5    1]
 [   0 1119    4    2    0    3    1    1    4    1]
 [   3    2  985    4    2    2    4    9   19    2]
 [   0    0   15  938    1   19    0    8   17   12]
 [   1    2    4    1  924    2   11    0    3   34]
 [   5    0    2   13    0  837    7    3   20    5]
 [   7    3    6    0    6   12  913    0   10    1]
 [   1    3   11    0    2    2    0  998    4    7]
 [   8    1    1    6    5    6    5    3  935    4]
 [   3    5    2    6    9    5    1   11   10  957]]
```

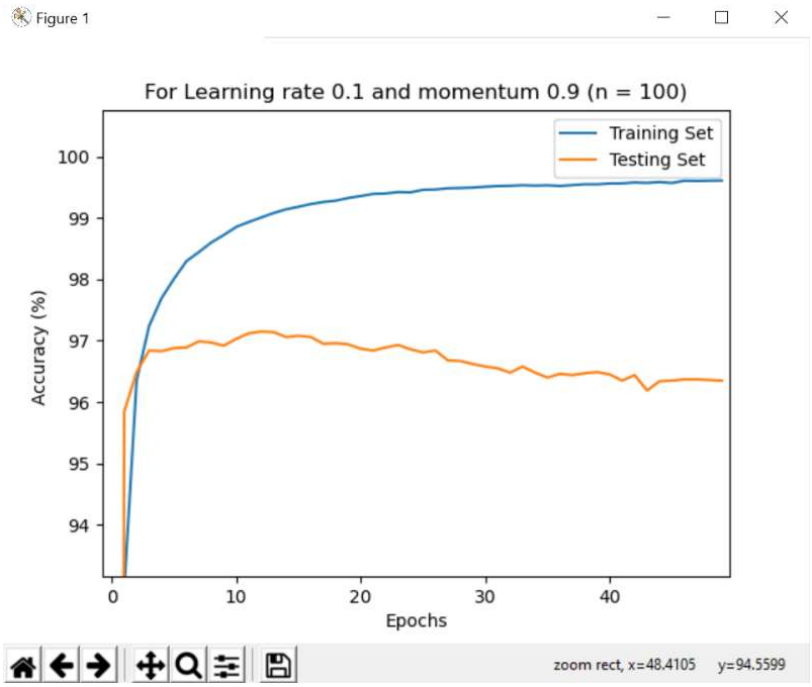
```
learning rate = 0.1;  momentum = 0.9; hidden units = 50
Training Accuracy
[ 9.73666667,92.22666667,95.57666667,96.25166667,96.69166667,96.96333333,
 97.14333333,97.315      ,97.46666667,97.61166667,97.70166667,97.79166667,
 97.88166667,97.95666667,97.98       ,98.095      ,98.10666667,98.18       ,
 98.17666667,98.27       ,98.285      ,98.30666667,98.34166667,98.36       ,
 98.40833333,98.43       ,98.45833333,98.50833333,98.51333333,98.53833333,
 98.55666667,98.57333333,98.575      ,98.575      ,98.61333333,98.645      ,
 98.63166667,98.655      ,98.66166667,98.66833333,98.69833333,98.71833333,
 98.715      ,98.73166667,98.74       ,98.75333333,98.77166667,98.78       ,
 98.78833333,98.79833333]
Testing Accuracy
[ 9.82,95.16,95.51,95.83,96.04,96.11,96.28,95.91,95.91,96.06,96.13,96.06,
 96.06,96.1 ,96.07,96.1 ,95.88,96.18,96.07,95.96,96.06,96.1 ,96.03,95.98,
 95.94,96.1 ,96.03,96.06,95.94,95.95,95.95,95.82,95.92,95.85,95.86,95.82,
 95.98,95.85,95.84,95.78,95.73,95.89,95.81,95.74,95.77,95.85,95.73,95.74,
 95.7 ,95.67]
```



***Case 3**: n = 100; Learning Rate 0.1; Momentum 0.9*
For training set

```
[[ 967    1    2    3    0    0    2    1    2    2]
 [   0 1117    2    5    1    1    1    1    7    0]
 [  12    2  972   16    3    1    4    3   18    1]
 [   0    0    9  972    0   10    0    4   11    4]
 [   4    0    2    1  945    0    3    0    1   26]
 [   4    0    1   14    0  850    5    0   12    6]
 [   5    3    5    1    3   13  919    0    9    0]
 [   0    3    7    3    7    1    0  985    6   16]
 [   2    0    2    5    4    5    1    5  947    3]
 [   5    3    0   14    9    3    0    5    9  961]]
```

```
learning rate = 0.1;  momentum = 0.9; hidden units = 100
Training Accuracy
[ 9.915     ,92.92666667,96.35166667,97.24      ,97.69333333,98.00833333,
 98.29666667,98.445     ,98.60166667,98.72333333,98.855     ,98.935     ,
 99.00833333,99.08166667,99.14166667,99.18333333,99.22833333,99.26166667,
 99.28333333,99.32666667,99.35833333,99.39333333,99.4       ,99.42333333,
 99.41833333,99.46      ,99.465     ,99.485     ,99.49      ,99.49666667,
 99.51      ,99.52166667,99.525     ,99.535     ,99.52833333,99.53333333,
 99.52333333,99.53666667,99.55      ,99.55      ,99.56333333,99.565     ,
 99.58      ,99.57333333,99.58666667,99.57166667,99.60833333,99.605     ,
 99.60833333,99.61166667]
Testing Accuracy
[10.09,95.84,96.49,96.84,96.83,96.88,96.89,96.99,96.97,96.92,97.03,97.12,
 97.15,97.14,97.06,97.08,97.06,96.95,96.96,96.94,96.87,96.84,96.89,96.93,
 96.86,96.81,96.84,96.68,96.67,96.62,96.58,96.55,96.48,96.58,96.48,96.4 ,
 96.46,96.44,96.47,96.49,96.45,96.35,96.44,96.19,96.34,96.35,96.37,96.37,
 96.36,96.35]
```



(1) How does the number of hidden units affect the final accuracy on the test data?

With the increase in hidden units, we see an increase in accuracy over training and testing set.

(2) How does it affect the number of epochs needed for training to converge?

Convergence chances reduced as the number of hidden units increase. Greater the hidden units, lesser the epochs needed where divergence occurs. That is chances of losing convergence occur at lower epoch value.

(3) Is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

We see the case of overfit in network after 25 epochs in n= 20, 50 and 100 where the model performed very well in training but not during testing phase.

(4) How do your results compare to the results obtained by your perceptron in HW 1?

In HW 1, single perceptron with learning rate 0.1, after 70 epochs has 89.08% accuracy on training set and 84.5%accuracy on testing set.
Now, with just 20 hidden units, with learning rate of 0.1, after 50 epochs we have 96% accuracy for training set and 93.6% accuracy over testing data set.
We can observe difference in accuracy of 9.1% during training .

Experiment 2: Vary the momentum value.

Here the number of hidden units is fixed to n= 100 and momentum value is varied to 0, 0.25, 0.5 and 0.9. the mode is trained and accuracy is plotted after 50 epochs.

Case 1*: n = 100; Learning Rate 0.1; Momentum 0*

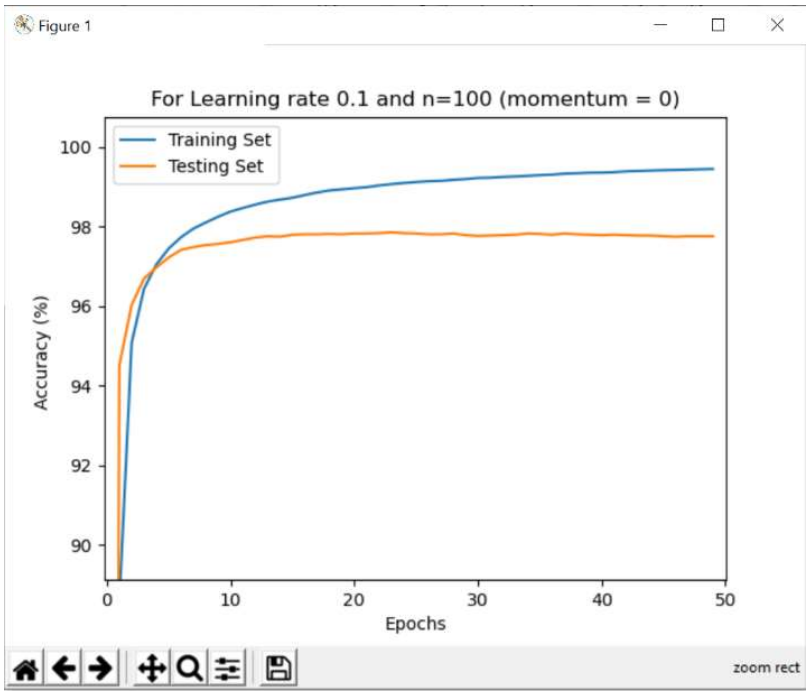
For testing

```
[[2955    0    0    0    0    0    0    0    1    1]
 [   1 3339    3    0    0    0    0    2    4    4]
 [   2    1 2954    0    0    0    0    2    0    0]
 [   0    0    2 3052    0    2    0    3    1    4]
 [   0    0    0    0 2920    0    1    1    1    4]
 [   3    0    1    2    0 2769    1    0    2    0]
 [   2    1    0    0    0    0 2945    0    1    0]
 [   0    3    2    2    1    0    0 3110    1    2]
 [   1    1    1    0    1    0    1    0 2930    0]
 [   2    1    1    0    0    0    0    3    2 2948]]
```

For Training

```
[[ 972    0    0    1    0    2    1    1    3    0]
 [   0 1117    3    6    0    1    2    2    4    0]
 [   5    2 1009    2    0    1    2    5    5    1]
 [   0    0    4  991    1    8    0    2    3    1]
 [   2    1    2    0  960    0    4    0    2   11]
 [   3    0    1   10    1  864    7    0    2    4]
 [   4    3    2    0    0    3  941    0    5    0]
 [   0    2   13    8    0    0    0  990    5   10]
 [   3    0    2    3    5    4    1    3  952    1]
 [   3    5    0    7    6    3    1    4    1  979]]
```

```
learning rate = 0.1;  momentum = 0; hidden units = 100
Training Accuracy
[10.44166667,88.50333333,95.07666667,96.43166667,97.045     ,97.445     ,
 97.72833333,97.945     ,98.09833333,98.24333333,98.37166667,98.46166667,
 98.54666667,98.62166667,98.67166667,98.71833333,98.78666667,98.85      ,
 98.90166667,98.93      ,98.95833333,98.98333333,99.02833333,99.06      ,
 99.08833333,99.11333333,99.13333333,99.14166667,99.16833333,99.18833333,
 99.215     ,99.22166667,99.24      ,99.25333333,99.26833333,99.28666667,
 99.29833333,99.32333333,99.33333333,99.34666667,99.35      ,99.36      ,
 99.38166667,99.39      ,99.4       ,99.41      ,99.41666667,99.425     ,
 99.43333333,99.44166667]
Testing Accuracy
[10.28,94.5 ,96.03,96.69,96.97,97.22,97.41,97.48,97.53,97.56,97.6 ,97.66,
 97.72,97.75,97.74,97.79,97.8 ,97.8 ,97.81,97.8 ,97.82,97.82,97.83,97.85,
 97.83,97.82,97.8 ,97.8 ,97.82,97.78,97.76,97.77,97.78,97.79,97.82,97.81,
 97.79,97.82,97.8 ,97.79,97.78,97.79,97.78,97.77,97.77,97.75,97.74,97.75,
 97.75,97.75]
```



***Case 2**: n = 100; Learning Rate 0.1; Momentum 0.25*

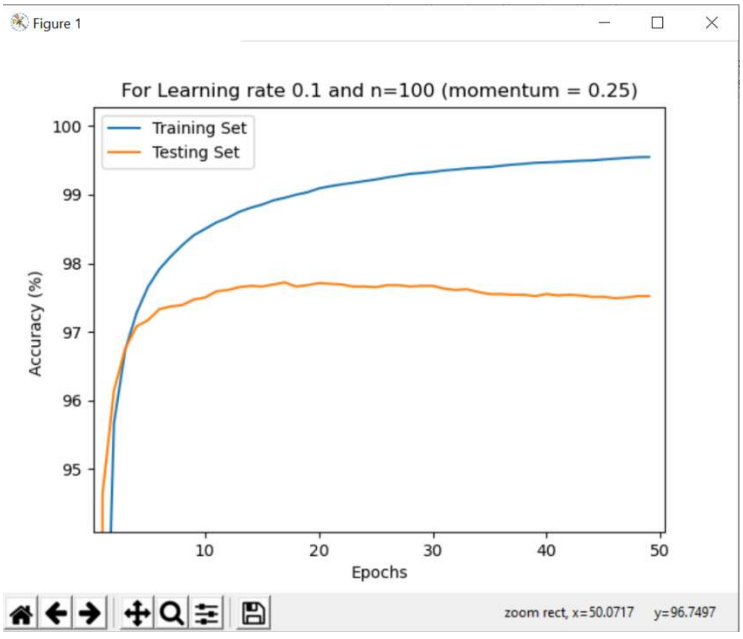
For training

```
[[2953    0    0    2    0    1    0    0    2    0]
 [   1 3365    1    1    1    0    1    1    3    1]
 [   0    1 2932    0    0    0    0    1    0    0]
 [   0    0    1 3025    0    0    0    1    4    1]
 [   1    2    0    0 2944    0    0    0    1    1]
 [   0    0    1    1    1 2711    1    0    2    0]
 [   1    1    0    0    0    0 2949    0    2    0]
 [   0    2    4    0    0    0    0 3188    0    4]
 [   1    1    1    0    0    1    0    0 2886    0]
 [   2    1    0    0    1    0    0    2    1 2987]]
```

For testing

```
[[ 970    0    1    1    0    1    3    1    3    0]
 [   0 1120    5    2    0    2    2    2    2    0]
 [   7    2 1002    2    1    0    3    6    9    0]
 [   0    1    5  987    1    5    0    6    3    2]
 [   1    0    4    0  961    0    3    0    1   12]
 [   4    0    1   13    0  853   11    1    7    2]
 [   7    3    0    1    2    5  938    0    2    0]
 [   0    3    8    2    2    0    0 1001    2   10]
 [   3    1    5    4    4    4    4    3  944    2]
 [   4    5    1    4    9    1    0    5    4  976]]
```

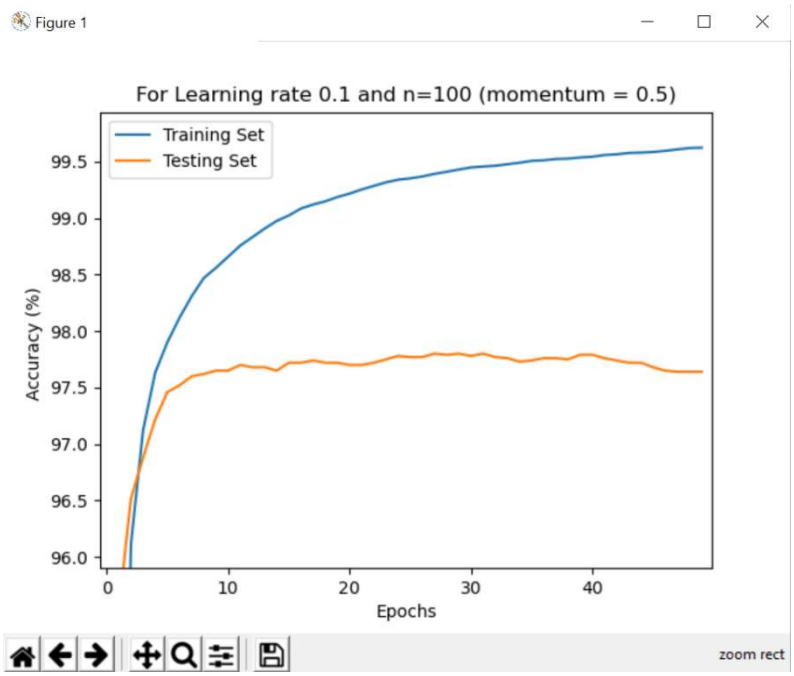
```
learning rate = 0.1;  momentum = 0.25; hidden units = 100
Training Accuracy
[14.565      ,89.76       ,95.66       ,96.74       ,97.28       ,97.65833333,
 97.915      ,98.1        ,98.26333333,98.40666667,98.5        ,98.595      ,
 98.66333333,98.75        ,98.80833333,98.855      ,98.91666667,98.955      ,
 99.         ,99.035      ,99.09166667,99.12333333,99.15       ,99.17333333,
 99.19833333,99.22166667,99.25166667,99.27666667,99.30333333,99.31666667,
 99.33166667,99.35333333,99.36666667,99.38333333,99.39333333,99.40333333,
 99.42166667,99.43833333,99.45       ,99.465      ,99.47166667,99.47833333,
 99.48666667,99.495      ,99.5        ,99.51333333,99.525      ,99.53666667,
 99.54666667,99.55       ]
Testing Accuracy
[14.52,94.63,96.14,96.76,97.08,97.17,97.33,97.37,97.39,97.47,97.5 ,97.59,
 97.61,97.65,97.67,97.66,97.69,97.72,97.66,97.68,97.71,97.7 ,97.69,97.66,
 97.66,97.65,97.68,97.68,97.66,97.67,97.67,97.63,97.61,97.62,97.58,97.55,
 97.55,97.54,97.54,97.52,97.55,97.53,97.54,97.53,97.51,97.51,97.49,97.5 ,
 97.52,97.52]
```



***Case 3**: n = 100; Learning Rate 0.1; Momentum 0.5*

```
[[ 971    1    2    1    1    1    0    1    2    0]
 [   0 1124    4    1    0    2    2    1    1    0]
 [   3    1 1012    2    0    0    3    6    4    1]
 [   1    1    6  984    0    6    2    3    5    2]
 [   1    0    2    1  962    0    4    0    0   12]
 [   3    0    1    7    1  866    5    2    5    2]
 [   6    3    2    1    2    5  935    0    4    0]
 [   0    2    8    6    3    0    0  997    3    9]
 [   4    1    5    2    6    6    4    7  937    2]
 [   2    3    0    6   11    2    1    3    5  976]]
```

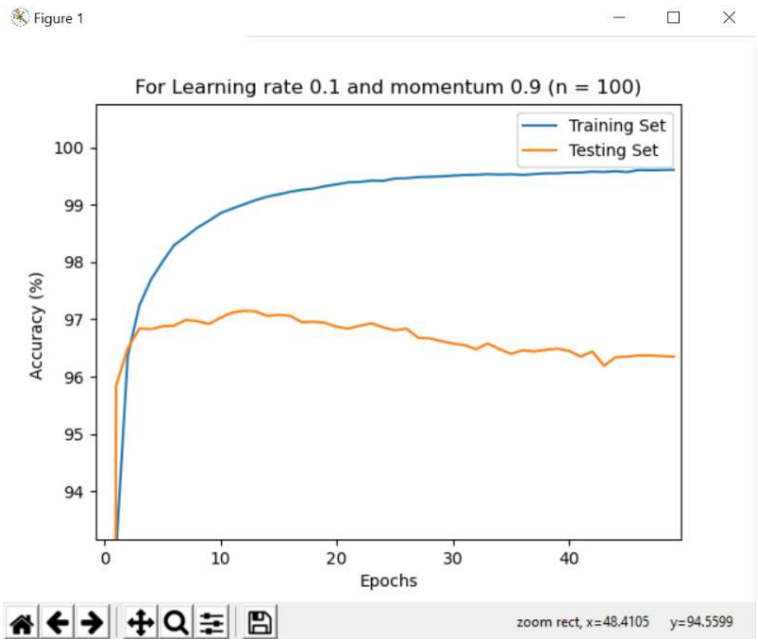
```
learning rate = 0.1;  momentum = 0.5; hidden units = 100
Training Accuracy
[ 9.86333333,91.07666667,96.11666667,97.12333333,97.63333333,97.90166667,
 98.11833333,98.30833333,98.47       ,98.56       ,98.65833333,98.75666667,
 98.83166667,98.90666667,98.975      ,99.02333333,99.085      ,99.12       ,
 99.14833333,99.18666667,99.21666667,99.25333333,99.285      ,99.31666667,
 99.34       ,99.35166667,99.36833333,99.39166667,99.41       ,99.43       ,
 99.44833333,99.45666667,99.46333333,99.47666667,99.49       ,99.50666667,
 99.51166667,99.52333333,99.52666667,99.53666667,99.54333333,99.55833333,
 99.565      ,99.57666667,99.58       ,99.585      ,99.595      ,99.60833333,
 99.62       ,99.62333333]
Testing Accuracy
[ 9.58,95.54,96.52,96.88,97.22,97.46,97.52,97.6 ,97.62,97.65,97.65,97.7 ,
 97.68,97.68,97.65,97.72,97.72,97.74,97.72,97.72,97.7 ,97.7 ,97.72,97.75,
 97.78,97.77,97.77,97.8 ,97.79,97.8 ,97.78,97.8 ,97.77,97.76,97.73,97.74,
 97.76,97.76,97.75,97.79,97.79,97.76,97.74,97.72,97.72,97.68,97.65,97.64,
 97.64,97.64]
```



***Case 4**: n = 100; Learning Rate 0.1; Momentum 0.9*

```
[[ 967    1    2    3    0    0    2    1    2    2]
 [   0 1117    2    5    1    1    1    1    7    0]
 [  12    2  972   16    3    1    4    3   18    1]
 [   0    0    9  972    0   10    0    4   11    4]
 [   4    0    2    1  945    0    3    0    1   26]
 [   4    0    1   14    0  850    5    0   12    6]
 [   5    3    5    1    3   13  919    0    9    0]
 [   0    3    7    3    7    1    0  985    6   16]
 [   2    0    2    5    4    5    1    5  947    3]
 [   5    3    0   14    9    3    0    5    9  961]]
```

```
learning rate = 0.1;  momentum = 0.9; hidden units = 100
Training Accuracy
[ 9.915     ,92.92666667,96.35166667,97.24      ,97.69333333,98.00833333,
 98.29666667,98.445     ,98.60166667,98.72333333,98.855     ,98.935     ,
 99.00833333,99.08166667,99.14166667,99.18333333,99.22833333,99.26166667,
 99.28333333,99.32666667,99.35833333,99.39333333,99.4       ,99.42333333,
 99.41833333,99.46      ,99.465     ,99.485     ,99.49      ,99.49666667,
 99.51      ,99.52166667,99.525     ,99.535     ,99.52833333,99.53333333,
 99.52333333,99.53666667,99.55      ,99.55      ,99.56333333,99.565     ,
 99.58      ,99.57333333,99.58666667,99.57166667,99.60833333,99.605     ,
 99.60833333,99.61166667]
Testing Accuracy
[10.09,95.84,96.49,96.84,96.83,96.88,96.89,96.99,96.97,96.92,97.03,97.12,
 97.15,97.14,97.06,97.08,97.06,96.95,96.96,96.94,96.87,96.84,96.89,96.93,
 96.86,96.81,96.84,96.68,96.67,96.62,96.58,96.55,96.48,96.58,96.48,96.4 ,
 96.46,96.44,96.47,96.49,96.45,96.35,96.44,96.19,96.34,96.35,96.37,96.37,
 96.36,96.35]
```



(1) How does the momentum value affect the final accuracy on the test data?

As the momentum increases the certain accuracy point for testing is reached more quickly.

(2) How does it affect the number of epochs needed for training to converge?

It needs lees epochs to converge.

(3) Again, is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

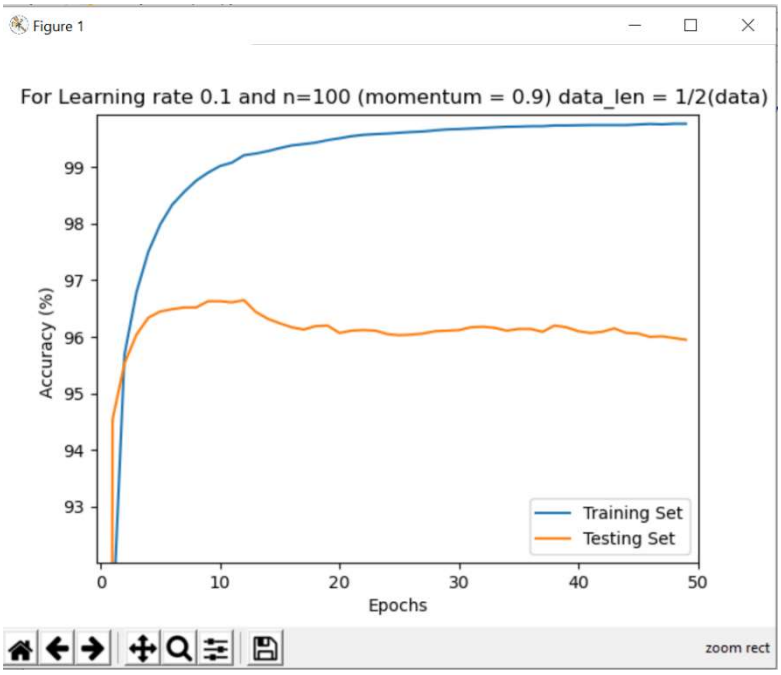
Yes, there is a deviation between training and testing data for n=100 and momentum=0.9

Experiment 3: Vary the number of training examples.

Case 1*: n = 100; Learning Rate 0.1; Momentum 0.9; data = ½(half).*

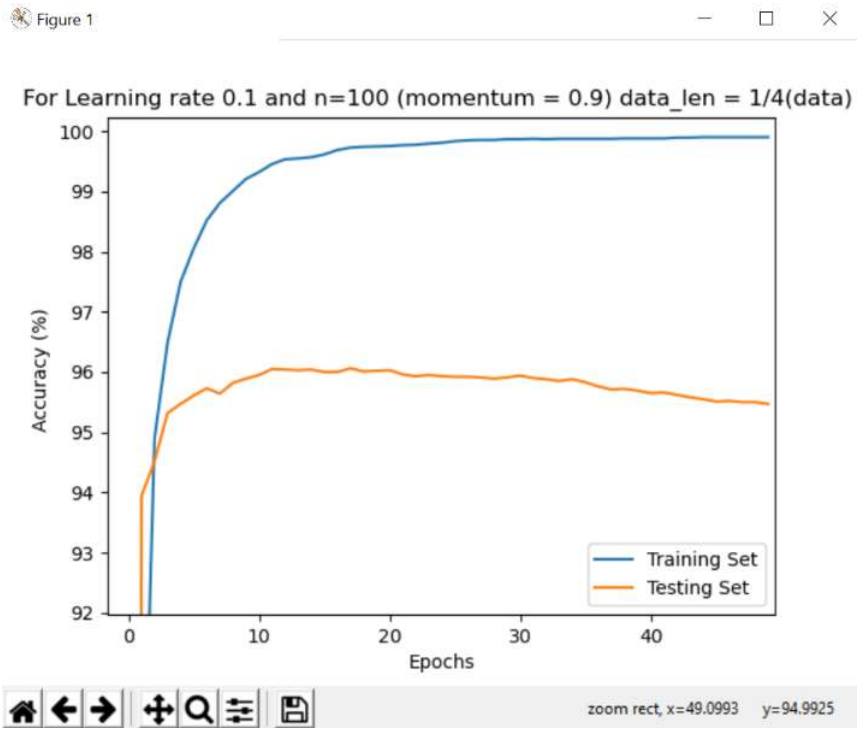
```
[[ 962    0    1    1    0    1    4    3    7    1]
 [   1 1119    1    4    0    3    1    1    5    0]
 [   6    0  984    7    1    1    5    6   19    3]
 [   1    0    6  966    0   10    1    4   16    6]
 [   1    0    2    0  936    0    5    2    4   32]
 [   4    1    0   17    1  832   10    3   16    8]
 [   7    3    3    1    2    6  920    0   15    1]
 [   1    3   13    5    3    1    0  978   11   13]
 [   4    0    1    7    3    1    3    6  945    4]
 [   4    5    1    4    9    0    1    5   27  953]]
```

```
learning rate = 0.1;  momentum = 0.9; hidden units = 100; data=30k
Training Accuracy
[11.70666667,90.81      ,95.68666667,96.78666667,97.50666667,97.99      ,
 98.33666667,98.56333333,98.76      ,98.90333333,99.02      ,99.08      ,
 99.21      ,99.24      ,99.28333333,99.33666667,99.38333333,99.40666667,
 99.43333333,99.47666667,99.51      ,99.54666667,99.57      ,99.58333333,
 99.59333333,99.60666667,99.62      ,99.63      ,99.65      ,99.66666667,
 99.67333333,99.68333333,99.69333333,99.70333333,99.71333333,99.71666667,
 99.72333333,99.72333333,99.73666667,99.73666667,99.74      ,99.74333333,
 99.74333333,99.74333333,99.74333333,99.75333333,99.76333333,99.75666667,
 99.76666667,99.76666667]
Testing Accuracy
[11.88,94.54,95.53,96.04,96.34,96.45,96.49,96.52,96.52,96.63,96.63,96.61,
 96.65,96.44,96.32,96.24,96.17,96.13,96.19,96.2 ,96.07,96.11,96.12,96.11,
 96.05,96.03,96.04,96.06,96.1 ,96.11,96.12,96.17,96.18,96.16,96.11,96.14,
 96.14,96.09,96.2 ,96.17,96.1 ,96.07,96.09,96.15,96.07,96.06,96.  ,96.01,
 95.98,95.95]
```



***Case 2**: n = 100; Learning Rate 0.1; Momentum 0.9; data =1/4(Quarter)*

```
[[ 955    0    3    1    0    1   10    2    5    3]
 [   0 1114    3    3    2    1    3    2    7    0]
 [   2    1  984    7    5    1    3   10   17    2]
 [   1    1   16  951    1   20    0    3   13    4]
 [   1    1    3    0  928    0    7    1    4   37]
 [   2    1    2   19    5  829    9    3   15    7]
 [   5    4    2    2    2   13  918    0   12    0]
 [   1    5   15    2    1    0    0  975   11   18]
 [   6    1    4    2    4    6    1    2  942    6]
 [   1    4    1    8   20    7    1    4   12  951]]
```



(1) How does the size of the training data affect the final accuracy on the test data?

As the size of the data reduced the accuracy over training data also reduced. When complete data is used for training the accuracy was % and when the data is reduce to half then the accuracy is % and when we took quarter of it its accuracy is %.The network need more data to train and improve its accuracy.

(2) How does it affect the number of epochs needed for training to converge?

As the training data of the model is high, it needed lees epochs to reach a maximum testing accuracy. And as the training data reduced it needed more epochs to reach that point and when the training fdata reduced a quarter it did not reach the maximum accuracy even when it reached 50 epochs.

(3) Again, is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

We see the case of overfit in network after 10 epochs in case 1 with 50% of data and in case 2 with 25% of data, where the model performed very well in training but not during testing phase.

```
[[2898    1    1    1    0    0    0    0    6    3]
 [   0 3401    1    0    1    0    0    2    1    0]
 [   1    0 2959    1    0    0    0    0    0    0]
 [   1    0    0 3037    1    0    0    3    8    4]
 [   0    0    1    0 2913    0    0    0    1    3]
 [   0    0    1    1    0 2758    2    0    3    1]
 [   1    0    1    0    0    0 2958    0    3    0]
 [   0    0    0    0    0    0    0 3124    1    4]
 [   0    0    0    1    0    0    0    0 2948    1]
 [   2    0    1    2    2    0    0    0    1 2935]]
```