

```
temp.py ->/Downloads/MLP-MNIST/MLP-MNIST Save
import numpy as np
actual_y = np.argmax(y, axis=1)
accuracy_val = accuracy(output, actual_y)
return accuracy_val

many_hidden_layers = 2
params = init_params(input_size = 784, output_size=10, many_hidden_layers=many_hidden_layers)
train_acc_epoch = []
val_acc_epoch = []

lrs = [1e-2]
alphas = [0.005]
batch_sizes = [50]
epoches = [75]
# epoch = 60
best_accuracy = 0;
for lr in lrs:
    for alpha in alphas:
        for batch_size in batch_sizes:
            denmin@denmin:~/Downloads/MLP-MNIST/MLP-MNIST$ python temp.py
File Edit View Search Terminal Help
best acc = 96.61999999999999 @ 30
best acc = 96.64 @ 31
best acc = 96.67999999999999 @ 32
best acc = 96.7 @ 33
best acc = 96.76 @ 34
best acc = 96.8 @ 36
best acc = 96.94 @ 37
best acc = 96.96 @ 38
best acc = 96.999999999999 @ 39
best acc = 96.999999999999 @ 41
best acc = 96.99 @ 42
best acc = 96.99 @ 47
best acc = 96.96000000000001 @ 50
best acc = 97.04 @ 52
best acc = 97.1 @ 53
best acc = 97.11999999999999 @ 55
best acc = 97.14 @ 58
best acc = 97.17 @ 60
best acc = 97.18 @ 61
best acc = 97.22 @ 63
x
best acc = 97.24000000000001 @ 65
best acc = 97.26 @ 70
best acc = 97.28 @ 72
test accuracy = 97.46000000000001
p[]
```

TeamViewer  
Free license (non-commercial use)  
Session list  
hanks-MacBook-Pro...  
(150 858 525)

Chat  
www.teamview...

Figure 1

Epoch	Validation Accuracy (%)	Training Accuracy (%)
0	65	65
10	90	90
20	92	92
30	94	94
40	95	95
50	96	96
60	97	97
70	98	98
75	99	99

Nodes in layers = 75, 50

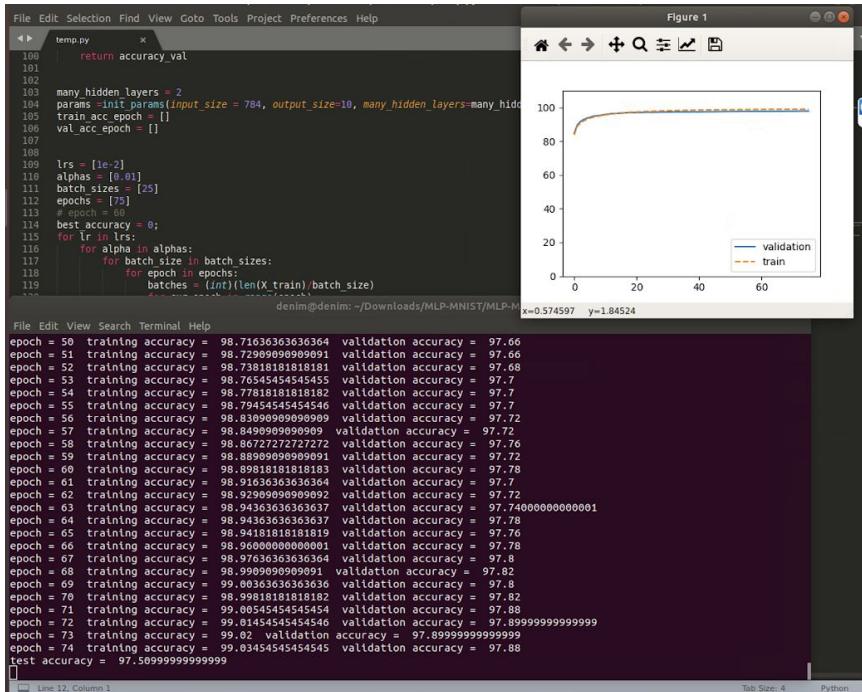
Learning rate:  $10^{-2}$

Alpha: 0.005

Batch size: 50

Epoch: 75

Test Accuracy = 97.46 %



Nodes in layers = 75, 50

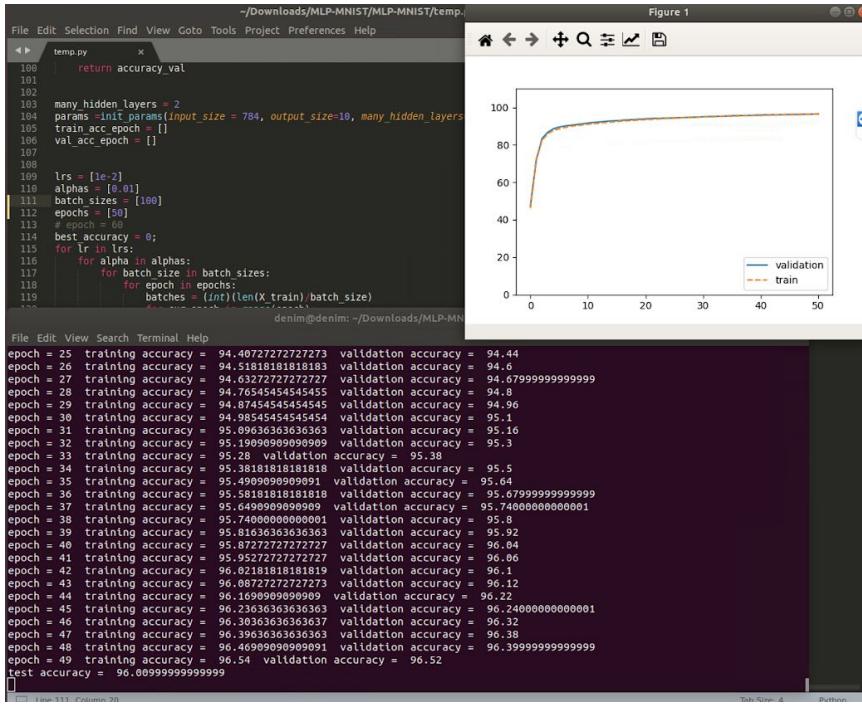
Learning rate:  $10^{-2}$

Alpha: 0.01

Batch size: 25

Epoch: 75

Test Accuracy = 97.50 %



Nodes in layers = 75, 50

Layers:2

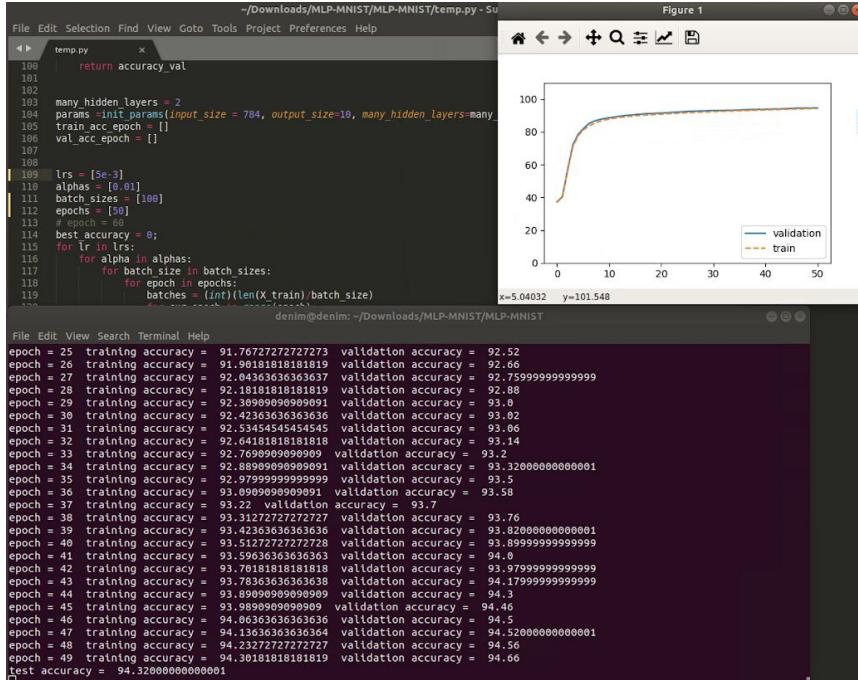
Learning rate:  $10^{-2}$

Alpha: 0.01

Batch size: 100

Epoch: 50

Test Accuracy = 96 %



Layers :2

Nodes in layers = 75, 50

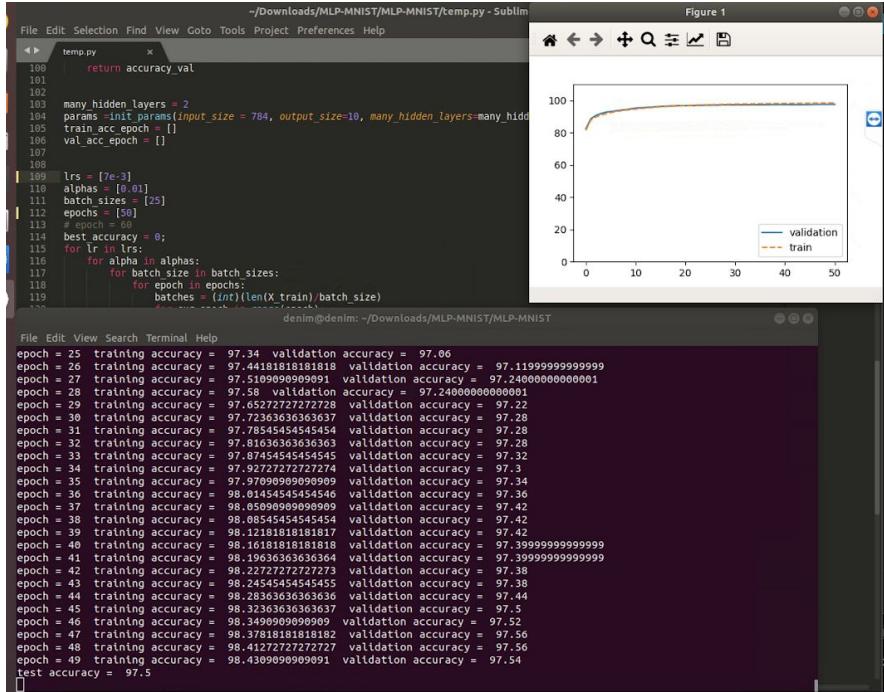
Learning rate:  $5 \times 10^{-3}$

Alpha: 0.01

Batch size: 100

Epoch: 50

Test Accuracy = 94.32 %



Layers: 2

Nodes in layers = 75, 50

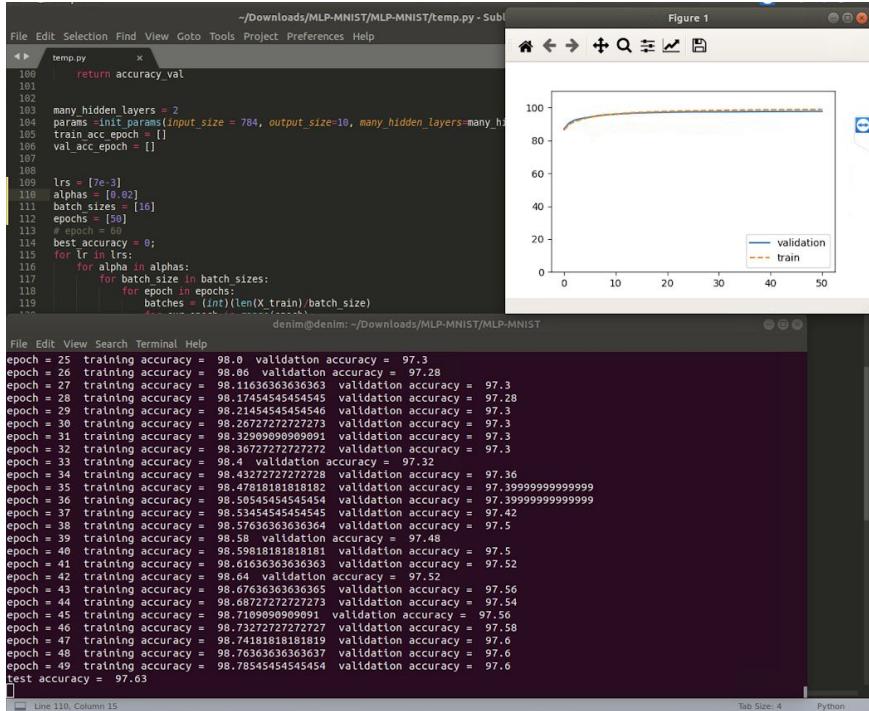
Learning rate:  $7 \times 10^{-2}$

Alpha: 0.01

Batch size: 25

Epoch: 50

Test Accuracy = 97.5 %



Layer: 2

Nodes in layers = 75, 50

Learning rate:  $7 \times 10^{-3}$

Alpha: 0.02

Batch size: 16

Epoch: 50

Test Accuracy = 97.63 %

Figure 1

```

temp.py
58     if(i == many_hidden_layers):
59         a_current = softmax(z_current).T
60     else:
61         a_current = relu(z_current)
62     # print('z mean = ', z_current.mean())
63     # print('a mean = ', a_current.mean())
64     layer_cache["z" + str(i)] = z_current
65     layer_cache["a" + str(i)] = a_current
66 return layer_cache
67
68 def calculate_derivatives(data_x, data_y, params, layer_cache, many_hidden_layer):
69     derivatives = {}
70     batch_size = len(data_x)
71     for i in reversed(range(many_hidden_layers+1)):
72         if(i==many_hidden_layers):
73             g = (np.subtract(layer_cache["a" + str(i)], data_y))
74         else:
75             g = np.multiply(np.dot(g,params["W"+str(i+1)].T),reluDerivative(i))
76             derivatives["b" + str(i)] = np.mean(g, axis = 0)
77     return derivatives
78
79
denim@denim:~/Downloads/MLP-MNIST/MLP-MNIST$ python temp.py
File Edit View Search Terminal Help
File Edit Selection Find View Goto Tools Project Preferences Help
temp.py
58     if(i == many_hidden_layers):
59         a_current = softmax(z_current).T
60     else:
61         a_current = relu(z_current)
62     # print('z mean = ', z_current.mean())
63     # print('a mean = ', a_current.mean())
64     layer_cache["z" + str(i)] = z_current
65     layer_cache["a" + str(i)] = a_current
66 return layer_cache
67
68 def calculate_derivatives(data_x, data_y, params, layer_cache, many_hidden_layer):
69     derivatives = {}
70     batch_size = len(data_x)
71     for i in reversed(range(many_hidden_layers+1)):
72         if(i==many_hidden_layers):
73             g = (np.subtract(layer_cache["a" + str(i)], data_y))
74         else:
75             g = np.multiply(np.dot(g,params["W"+str(i+1)].T),reluDerivative(i))
76             derivatives["b" + str(i)] = np.mean(g, axis = 0)
77     return derivatives
78
79
denim@denim:~/Downloads/MLP-MNIST/MLP-MNIST$ python temp.py
Epoch: 25 training accuracy = 98.22545454545455 validation accuracy = 97.54
Epoch: 26 training accuracy = 98.29454545454546 validation accuracy = 97.54
Epoch: 27 training accuracy = 98.37727272727273 validation accuracy = 97.5
Epoch: 28 training accuracy = 98.36727272727272 validation accuracy = 97.48
Epoch: 29 training accuracy = 98.409999999999 validation accuracy = 97.5
Epoch: 30 training accuracy = 98.46727272727273 validation accuracy = 97.5
Epoch: 31 training accuracy = 98.50363636363636 validation accuracy = 97.5
Epoch: 32 training accuracy = 98.53818181818181 validation accuracy = 97.52
Epoch: 33 training accuracy = 98.57818181818182 validation accuracy = 97.54
Epoch: 34 training accuracy = 98.60545454545453 validation accuracy = 97.56
Epoch: 35 training accuracy = 98.63154090909999 validation accuracy = 97.58
Epoch: 36 training accuracy = 98.65154090909999 validation accuracy = 97.6
Epoch: 37 training accuracy = 98.672727272727 validation accuracy = 97.65
Epoch: 38 training accuracy = 98.69000000000001 validation accuracy = 97.7
Epoch: 39 training accuracy = 98.70545454545454 validation accuracy = 97.72
Epoch: 40 training accuracy = 98.74545454545455 validation accuracy = 97.72
Epoch: 41 training accuracy = 98.752727272727 validation accuracy = 97.72
Epoch: 42 training accuracy = 98.756363636363 validation accuracy = 97.76
Epoch: 43 training accuracy = 98.78545454545454 validation accuracy = 97.72
Epoch: 44 training accuracy = 98.812727272727 validation accuracy = 97.76
Epoch: 45 training accuracy = 98.818181818181 validation accuracy = 97.76
Epoch: 46 training accuracy = 98.843636363636 validation accuracy = 97.76
Epoch: 47 training accuracy = 98.86 Validation accuracy = 97.78
Epoch: 48 training accuracy = 98.87090909090909 validation accuracy = 97.76
Epoch: 49 training accuracy = 98.907272727273 validation accuracy = 97.7400000000000001
Test accuracy = 97.0
[ 2 lines, 106 characters selected
Tab Size: 4 Python

```

No of nodes: 2

Nodes in layers = 75, 50

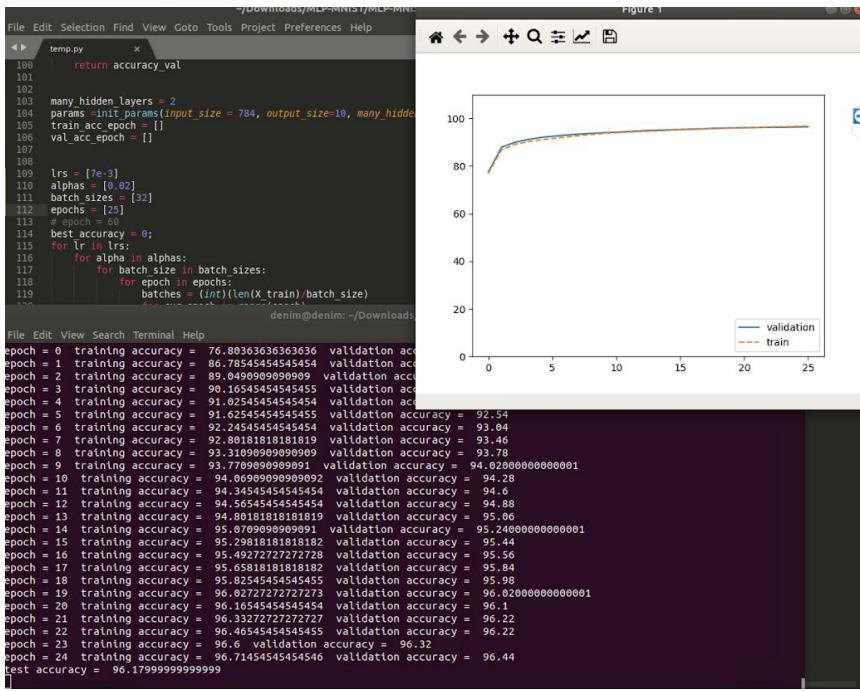
Learning rate:  $10^{-2}$

Alpha: 0.06

Batch size: 60

Epoch: 25

Test Accuracy = 97.6 %



No of layers: 2

Nodes in layers = 75, 50

Learning rate:  $7 \times 10^{-3}$

Alpha: 0.02

Batch size: 32

Epoch: 25

Test Accuracy = 96.17 %

File Edit Selection Find Goto Tools Project Preferences Help

```
temp.py x
100     return accuracy_val
101
102
103 many_hidden_layers = 2
104 params = init_params(input_size = 784, output_size=10, many_hidden_layers=many_hidden_layers)
105 train_acc_epoch = []
106 val_acc_epoch = []
107
108
109 lrs = [7e-3]
110 alphas = [0.02]
111 batch_sizes = [32]
112 epochs = [25]
113 # epoch = 0
114 best_accuracy = 0;
115
116 if __name__ == "__main__":
TeamViewer<  TeamViewer>
117     for alpha in alphas:
118         for batch_size in batch_sizes:
119             for epoch in epochs:
120                 batches = (int(len(X_train)/batch_size))
denim@denim: ~/Downloads/MLP-MNIST/MLP-MNIST/temp.py
```

Figure 1

Epoch	train accuracy	validation accuracy
0	~80	~80
5	~90	~90
10	~95	~95
25	~95	~95

```
File Edit View Search Terminal Help
epoch = 0 training accuracy = 76.80363636363636 validation accuracy = 77.34
epoch = 1 training accuracy = 86.78545454545454 validation accuracy = 87.88
epoch = 2 training accuracy = 89.0499999990909 validation accuracy = 89.92
epoch = 3 training accuracy = 90.16545454545455 validation accuracy = 91.10000000000001
epoch = 4 training accuracy = 91.02545454545454 validation accuracy = 92.04
epoch = 5 training accuracy = 91.62545454545455 validation accuracy = 92.54
epoch = 6 training accuracy = 92.24545454545454 validation accuracy = 93.04
epoch = 7 training accuracy = 92.72545454545454 validation accuracy = 93.54
epoch = 8 training accuracy = 93.18099999999999 validation accuracy = 93.78
epoch = 9 training accuracy = 93.77099999999991 validation accuracy = 94.02000000000001
epoch = 10 training accuracy = 94.06999999999992 validation accuracy = 94.28
epoch = 11 training accuracy = 94.34545454545454 validation accuracy = 94.6
epoch = 12 training accuracy = 94.56545454545454 validation accuracy = 94.88
epoch = 13 training accuracy = 94.80181818181819 validation accuracy = 95.06
epoch = 14 training accuracy = 95.07999999999991 validation accuracy = 95.24000000000001
epoch = 15 training accuracy = 95.29818181818182 validation accuracy = 95.44
epoch = 16 training accuracy = 95.49227727272728 validation accuracy = 95.56
epoch = 17 training accuracy = 95.65818181818182 validation accuracy = 95.84
epoch = 18 training accuracy = 95.82545454545454 validation accuracy = 95.98
epoch = 19 training accuracy = 96.02000000000001 validation accuracy = 96.02000000000001
epoch = 20 training accuracy = 96.18545454545454 validation accuracy = 96.1
epoch = 21 training accuracy = 96.33772727272727 validation accuracy = 96.22
epoch = 22 training accuracy = 96.46545454545454 validation accuracy = 96.22
epoch = 23 training accuracy = 96.6 validation accuracy = 96.32
epoch = 24 training accuracy = 96.71545454545454 validation accuracy = 96.44
test accuracy = 96.17999999999999
```

No of layers: 2

Nodes: 75, 50

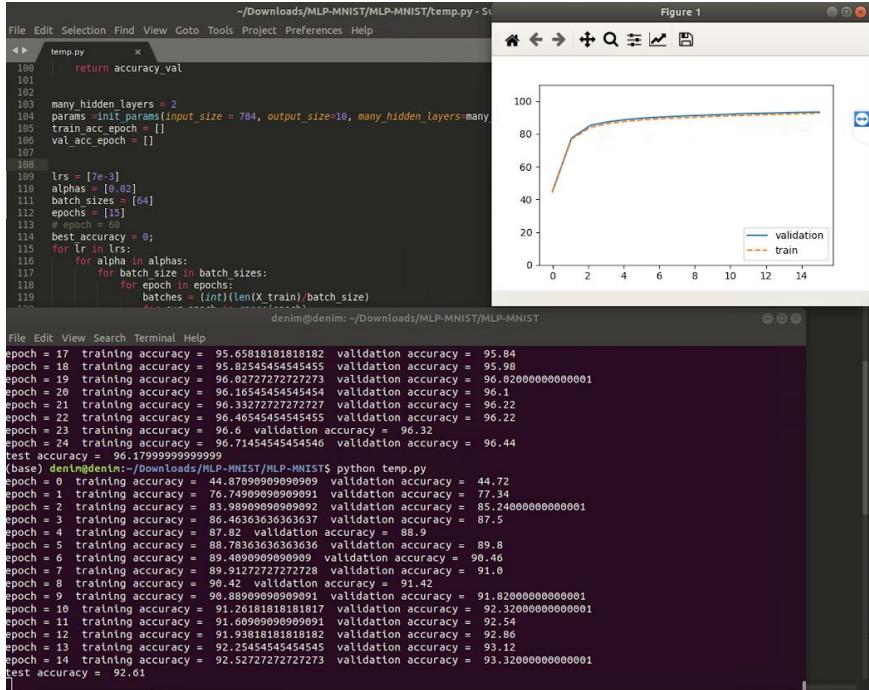
Learning rate:  $7 \times 10^{-2}$

Alpha: 0.02

Batch size: 32

Epoch: 25

Test Accuracy = 96.18 %



No of layers: 2

Nodes: 75, 50

Learning rate:  $7 \times 10^{-2}$

Alpha: 0.02

Batch size: 64

Epoch: 15

Test Accuracy = 92.61 %

The screenshot shows a terminal window with Python code for training a neural network. The code iterates over hidden layers (2), learning rates (3e-3), alphas (0.03), batch sizes (64), and epochs (20). It calculates training and validation accuracy for each combination. A plot is shown with validation accuracy (solid blue line) and train accuracy (dashed orange line) increasing from ~30% to ~90% over 20 epochs.

```

File Edit Selection Find View Goto Tools Project Preferences Help
temp.py x
100     return accuracy_val
101
102
103 many_hidden_layers = 2
104 params_init_params(input_size = 784, output_size=10, many_hidden_layers=many_h
105 train_acc_epoch = []
106 val_acc_epoch = []
107
108
109 lrs = [3e-3]
110 alphas = [0.03]
111 batch_sizes = [64]
112 epochs = [20]
113 # epoch = 60
114 best_accuracy = 0;
115 for lr in lrs:
116     for alpha in alphas:
117         for batch_size in batch_sizes:
118             for epoch in epochs:
119                 batches = (int)(len(X_train)/batch_size)
120                 ...
denim@denim:~/Downloads/MLP-MNIST/MLP-MNIST
```

```

File Edit View Search Terminal Help
epoch = 12 training accuracy = 91.93818181818182 validation accuracy = 92.86
epoch = 13 training accuracy = 92.2545454545454 validation accuracy = 93.12
epoch = 14 training accuracy = 92.52727272727273 validation accuracy = 93.320000000000001
test accuracy = 92.61
(base) denim@denim:~/Downloads/MLP-MNIST/MLP-MNISTS python temp.py
epoch = 0 training accuracy = 31.387272727272723 validation accuracy = 30.09999999999998
epoch = 1 training accuracy = 34.2165454545454 validation accuracy = 33.68
epoch = 2 training accuracy = 46.698181818181816 validation accuracy = 49.3
epoch = 3 training accuracy = 67.58181818181819 validation accuracy = 68.820000000000001
epoch = 4 training accuracy = 75.8999999999999 validation accuracy = 76.48
epoch = 5 training accuracy = 80.003636363636 validation accuracy = 80.84
epoch = 6 training accuracy = 82.74999999999991 validation accuracy = 83.58
epoch = 7 training accuracy = 84.58999999999991 validation accuracy = 85.7
epoch = 8 training accuracy = 85.96181818181819 validation accuracy = 87.44
epoch = 9 training accuracy = 86.92 validation accuracy = 88.2
epoch = 10 training accuracy = 87.91 validation accuracy = 89.16
epoch = 11 training accuracy = 88.26363636363637 validation accuracy = 89.32
epoch = 12 training accuracy = 88.79181818181818 validation accuracy = 89.28
epoch = 13 training accuracy = 89.08181818181818 validation accuracy = 89.75999999999999
epoch = 14 training accuracy = 89.40727272727273 validation accuracy = 89.94
epoch = 15 training accuracy = 89.7218181818181 validation accuracy = 90.22
epoch = 16 training accuracy = 89.956363636363 validation accuracy = 90.44
epoch = 17 training accuracy = 90.176363636363 validation accuracy = 90.58
epoch = 18 training accuracy = 90.3618181818181 validation accuracy = 90.84
epoch = 19 training accuracy = 90.485454545454 validation accuracy = 91.14
test accuracy = 90.89
```

No of layers = 2

Nodes = 75, 50

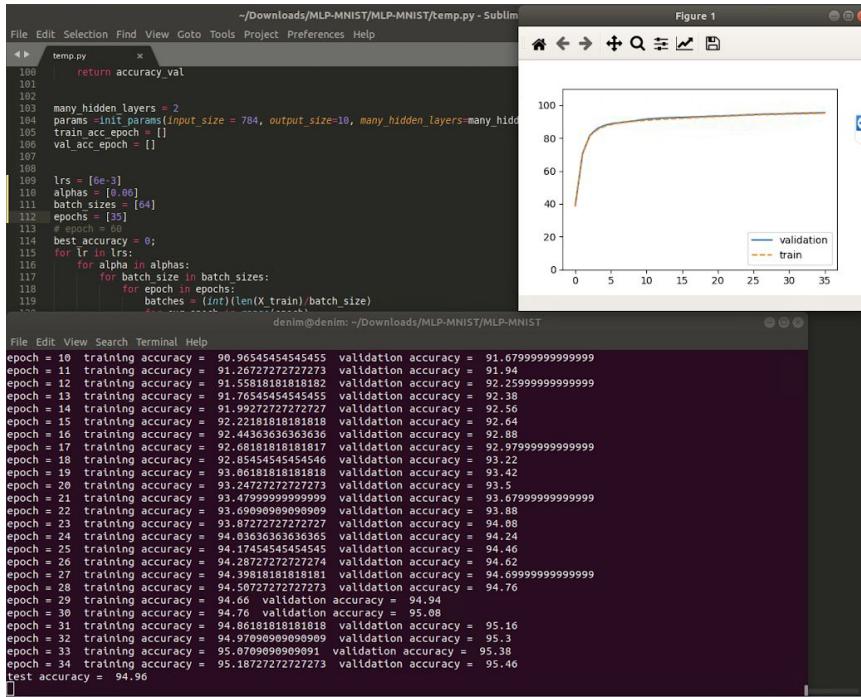
Learning rate:  $3 \times 10^{-3}$

Alpha: 0.03

Batch size: 64

Epoch: 20

Test Accuracy = 90.89 %



No of layers = 2

Nodes = 75, 50

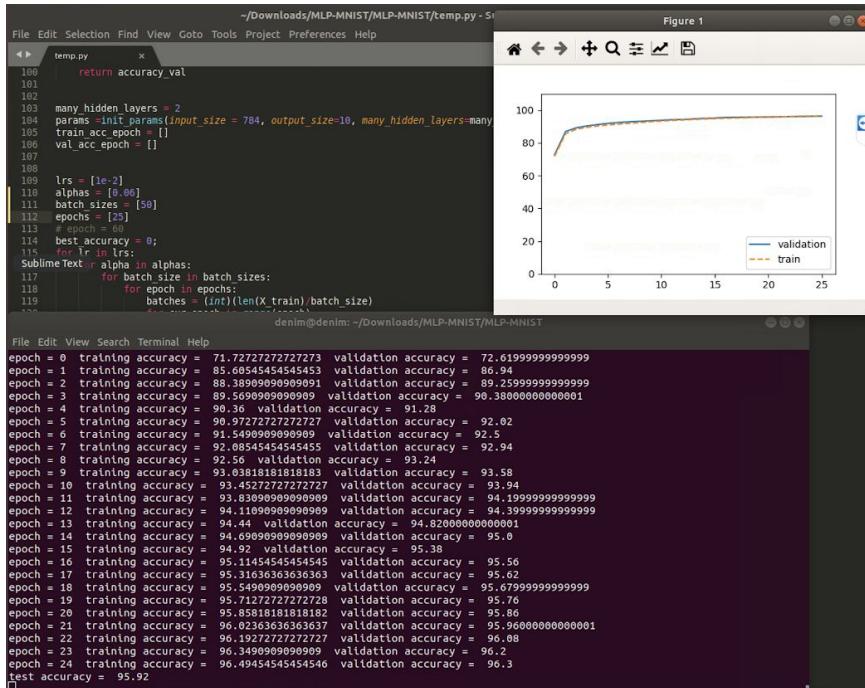
Learning rate:  $6 \times 10^{-2}$

Alpha: 0.06

Batch size: 64

Epoch: 35

Test Accuracy = 94.96 %



No of layers = 2

Learning rate:  $10^{-2}$

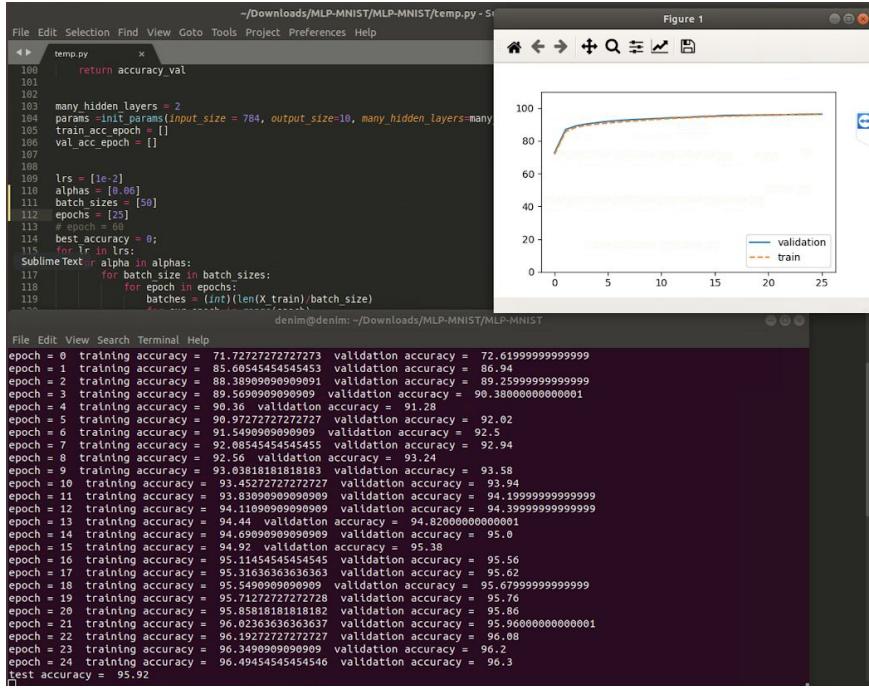
Nodes: 75, 50

Alpha: 0.06

Batch size: 60

Epoch: 25

Test Accuracy = 95.92 %



No of layers: 2

Nodes: 75, 50

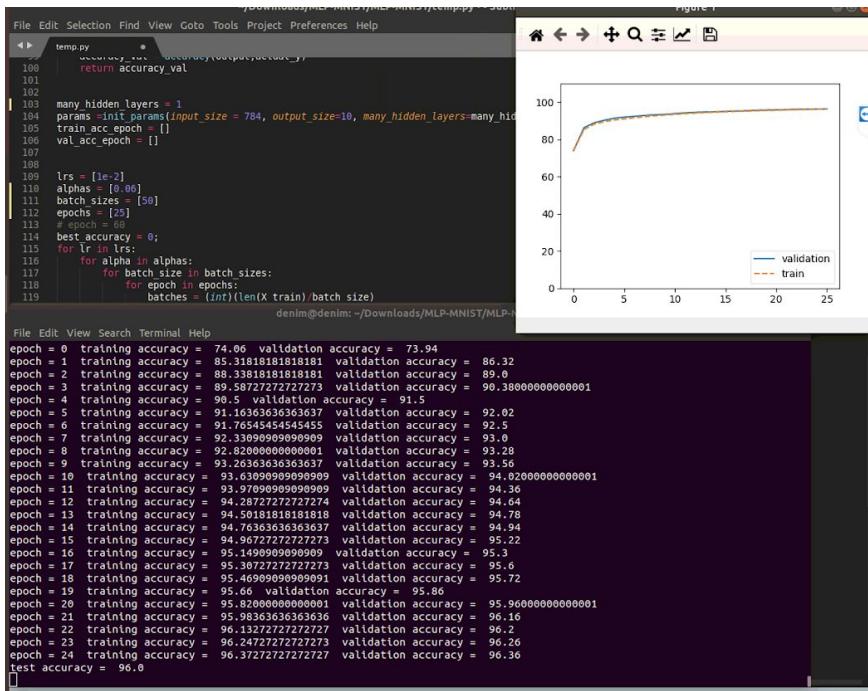
Learning rate: 10^-2

Alpha: 0.06

Batch size: 60

Epoch: 25

Test Accuracy = 95.92 %



No of layers: 1

Nodes: 75

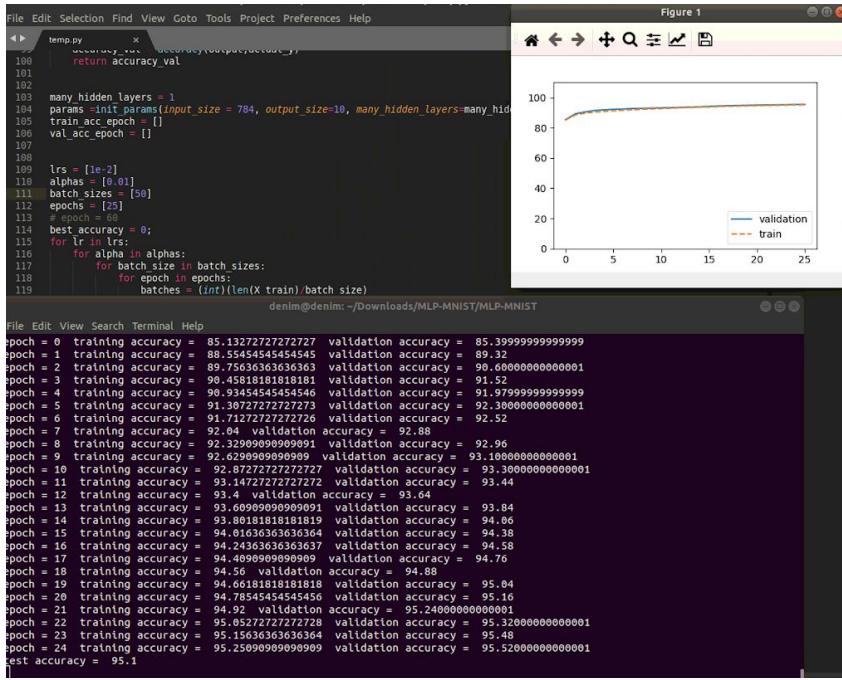
Learning rate:  $10^{-2}$

Alpha: 0.06

Batch size: 50

Epoch: 25

Test Accuracy = 96 %



No of layers: 1

Nodes: 75

Learning rate:  $10^{-2}$

Alpha: 0.01

Batch size: 50

Epoch: 25

Test Accuracy = 95.1 %

File Edit Selection Find View Goto Tools Project Preferences Help

temp.py    accuracy\_val = accuracy(output,actual)

```

100     return accuracy_val
101
102
103 many_hidden_layers = 1
104 params = init_params(input_size = 784, output_size=10, many_hidden_layers=many_hidden_
105 train_acc_epoch = []
106 val_acc_epoch = []
107
108 lrs = [3,3e-3]
109 alphas = [0.05]
110 batch_sizes = [50]
111 epochs = [25]
112 # epoch = 66
113 best_accuracy = 0;
114
115 for lr in lrs:
116     for alpha in alphas:
117         for batch_size in batch_sizes:
118             for epoch in epochs:
119                 batches = (int)(len(X_train)/batch_size)

```

denim@denim:~/Downloads/MLP-MNIST/MLP-MNIST\$

Figure 1

File Edit View Search Terminal Help

epoch = 0 training accuracy = 75.07545454545454 validation accuracy = 75.72  
epoch = 1 training accuracy = 82.98545454545454 validation accuracy = 83.78  
epoch = 2 training accuracy = 85.8 validation accuracy = 86.48  
epoch = 3 training accuracy = 87.21090909090909 validation accuracy = 88.08  
epoch = 4 training accuracy = 88.03272727272727 validation accuracy = 89.1  
epoch = 5 training accuracy = 88.5690909090909 validation accuracy = 89.48  
epoch = 6 training accuracy = 89.0290909090909 validation accuracy = 89.98  
epoch = 7 training accuracy = 89.456909090909 validation accuracy = 90.08  
epoch = 8 training accuracy = 89.76181818181819 validation accuracy = 90.380000000000001  
epoch = 9 training accuracy = 90.069990999999 validation accuracy = 90.84  
epoch = 10 training accuracy = 90.2945454545454 validation accuracy = 91.0  
epoch = 11 training accuracy = 90.52727272727272 validation accuracy = 91.58  
epoch = 12 training accuracy = 90.75090909090909 validation accuracy = 91.58  
epoch = 13 training accuracy = 90.930009090909 validation accuracy = 91.67999999999999  
epoch = 14 training accuracy = 91.09818181818183 validation accuracy = 91.74  
epoch = 15 training accuracy = 91.28545454545454 validation accuracy = 91.9  
epoch = 16 training accuracy = 91.43636363636364 validation accuracy = 92.08  
epoch = 17 training accuracy = 91.57272727272728 validation accuracy = 92.24  
epoch = 18 training accuracy = 91.7236363636363 validation accuracy = 92.34  
epoch = 19 training accuracy = 91.84727272727272 validation accuracy = 92.5  
epoch = 20 training accuracy = 91.99636363636363 validation accuracy = 92.7  
epoch = 21 training accuracy = 92.15636363636364 validation accuracy = 92.78  
epoch = 22 training accuracy = 92.265454545455 validation accuracy = 92.88  
epoch = 23 training accuracy = 92.372727272727 validation accuracy = 92.94  
epoch = 24 training accuracy = 92.478181818181 validation accuracy = 92.94  
test accuracy = 93.02

No of layers: 1

Nodes = 75

Learning rate: 3.3<sup>-3</sup>

Alpha: 0.05

Batch size: 50

Epoch: 25

Test Accuracy = 93.02 %

```

temp.py
Open ▾ Save ▾
compute_accuracy(actual_y, output)
actual_y = np.argmax(y, axis=1)
accuracy_val = accuracy(output, actual_y)
return accuracy_val

many_hidden_layers = 2
params = init_params(input_size = 784, output_size=10, many_hidden_layers=many_hidden_layers)
train_acc_epoch = []
val_acc_epoch = []

lrs = [1e-2]
alphas = [0.01]
batch_sizes = [25]
epochs = [75]
# epoch = 68
best_accuracy = 0;
for lr in lrs:
    for alpha in alphas:
        for batch_size in batch_sizes:
            for epoch in epochs:
                print("denim@denim:~/Downloads/MLP-MNIST/MLP-PYTHON$ python temp.py", lr, alpha, batch_size, epoch)
                best acc = 96.67999999999999 @ 18
                best acc = 96.78 @ 19
                best acc = 96.89999999999999 @ 20
                best acc = 97.0 @ 24
                best acc = 97.08 @ 26
                best acc = 97.11999999999999 @ 28
                best acc = 97.16 @ 29
                best acc = 97.18 @ 31
                best acc = 97.2 @ 32
                best acc = 97.28 @ 33
                best acc = 97.3 @ 34
                best acc = 97.32 @ 35
                best acc = 97.34 @ 37
                best acc = 97.39999999999999 @ 39
                best acc = 97.44 @ 42
                best acc = 97.46000000000001 @ 44
                best acc = 97.48 @ 45
                best acc = 97.5 @ 50
                best acc = 97.58 @ 51
                best acc = 97.61999999999999 @ 52
                best acc = 97.66 @ 54
                best acc = 97.68 @ 59
                test accuracy = 97.7
                print("denim@denim:~/Downloads/MLP-MNIST/MLP-PYTHON$")

```

The terminal shows the execution of a Python script named `temp.py`. The script performs a grid search over learning rate (lrs), alpha, batch size, and epoch count. It prints the accuracy at each step. The final test accuracy is 97.7%.

Figure 1 shows a plot of accuracy versus epoch number (0 to 70). The plot displays two curves: 'validation' (solid blue line) and 'train' (dashed orange line). Both curves start around 85% accuracy and quickly rise to stabilize near 100% accuracy by epoch 20.

**No of layers: 2**

**Nodes = 75, 50**

**Learning rate:  $10^{-2}$**

**Alpha: 0.01**

**Batch size: 25**

**Epoch: 75**

**Test Accuracy = 97.7 % <- Hyper Parameter**