Solution 1

(a) Finding cluster centers:

Figure 1: Cluster centers 25, 49, 64, 81, 100

(b) Finding superpixels with 25, 49, 64, 81, 100

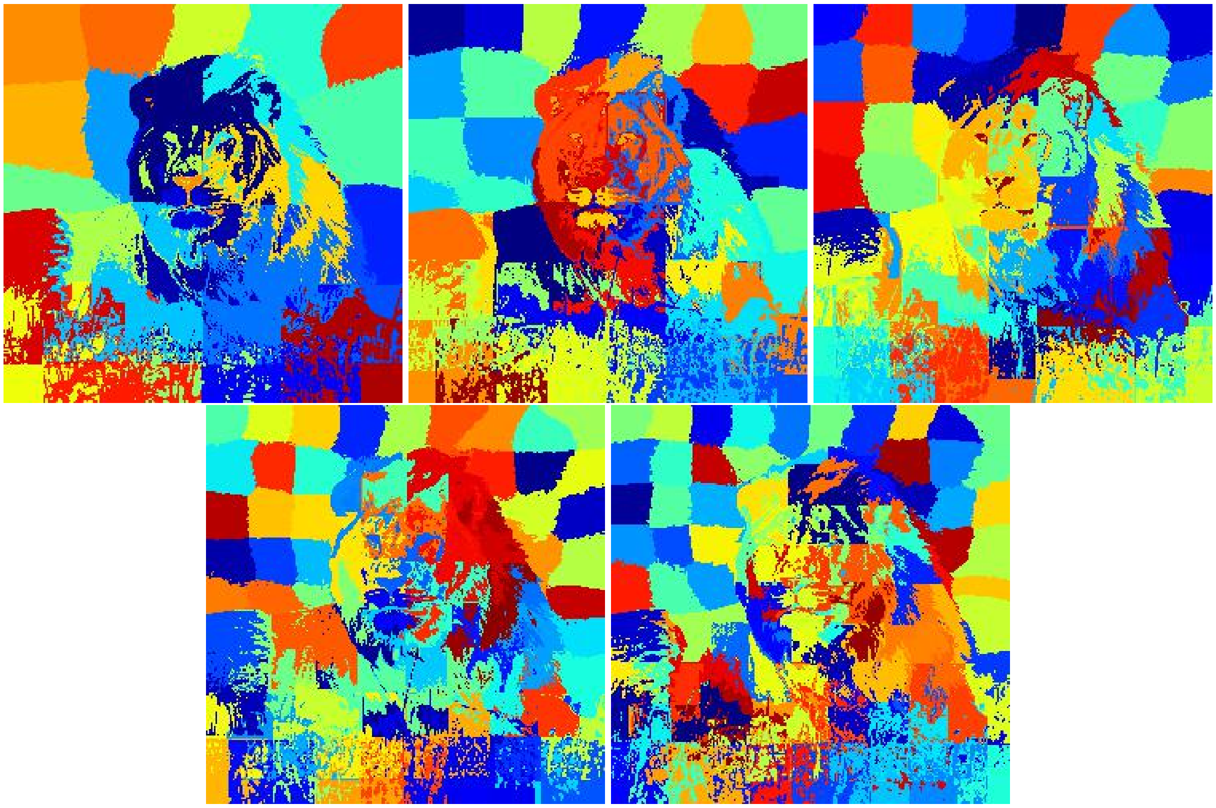


Figure 2: Superpixels 25, 49, 64, 81, 100 with spatial weight 0.7

The spatial weight I choose is 0.7. If the spatial weight is approaching 1.0, the superpixel will become more like a square; if the spatial weight is approaching 0.0, the superpixel will become more irregular, because under this circumstance, the cluster is more determined by the intensities of R, G and B. As shown in the following figures.

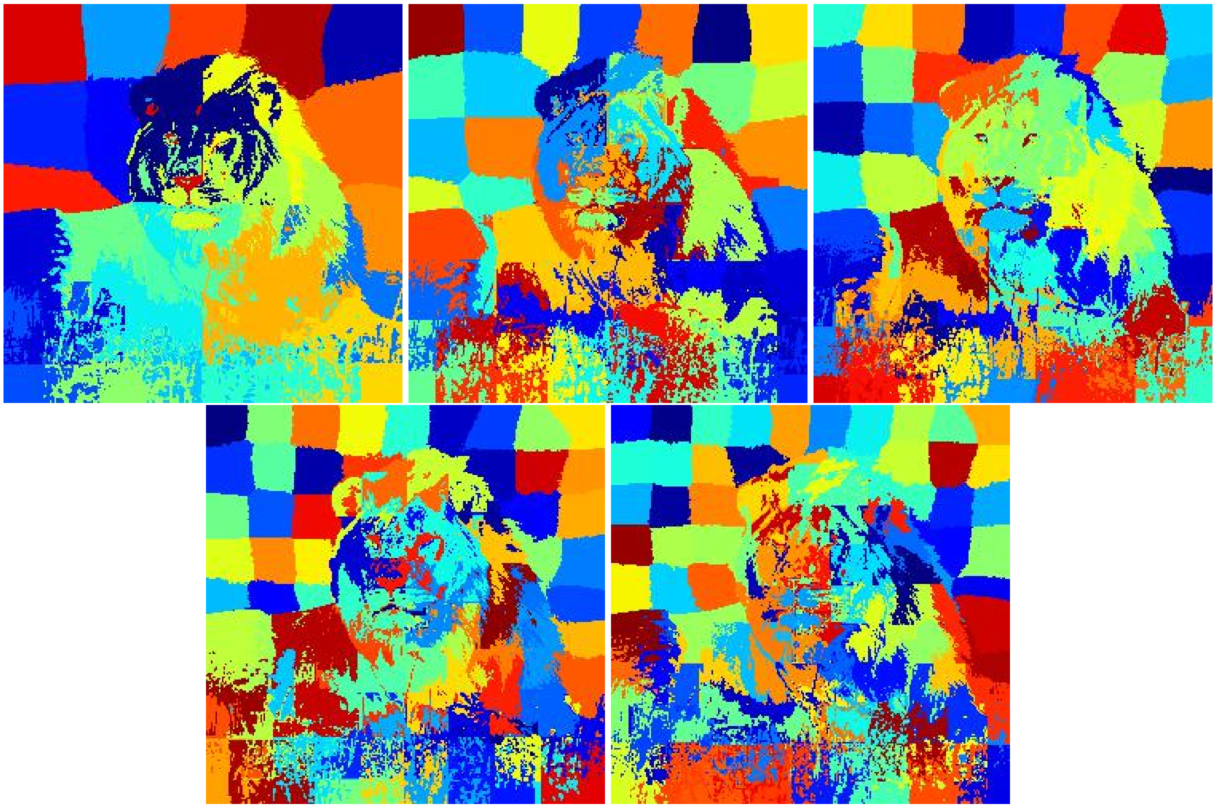


Figure 3: Superpixels 25, 49, 64, 81, 100 with spatial weight 1.0

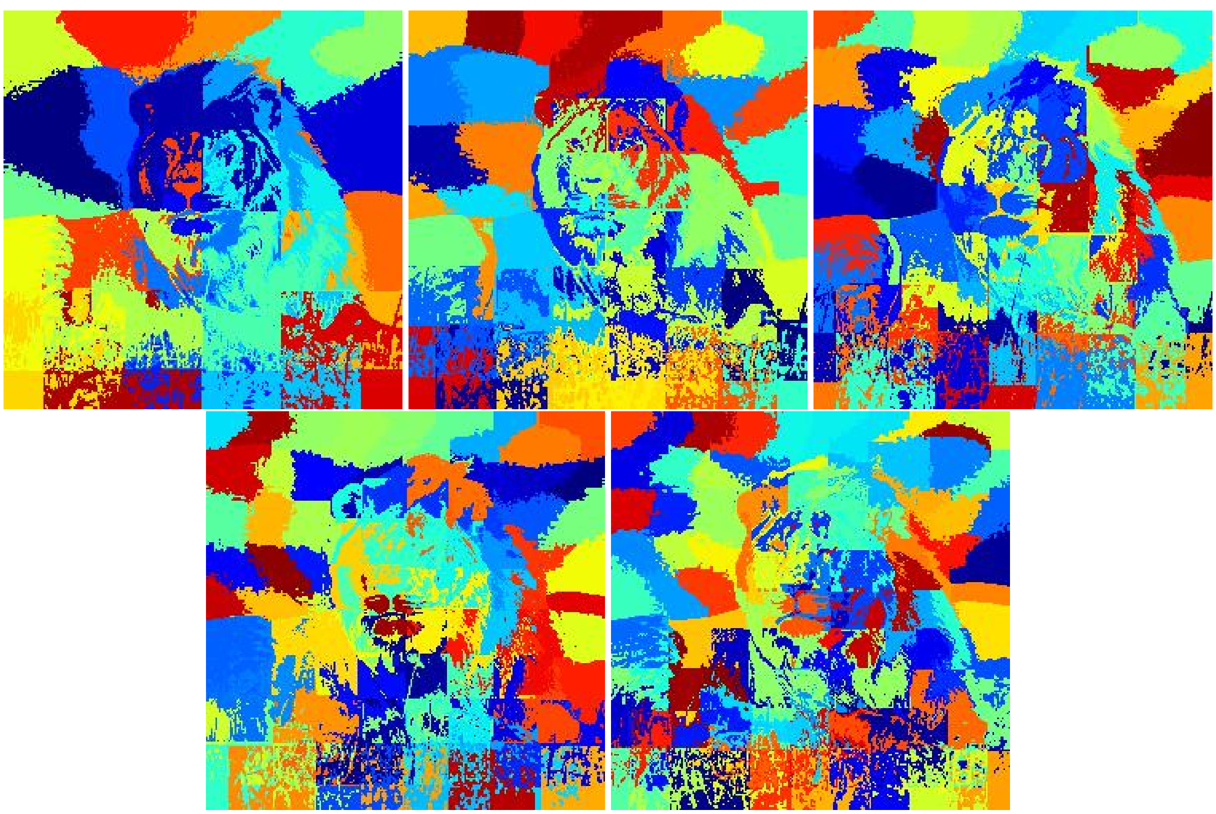


Figure 4: Superpixels 25, 49, 64, 81, 100 with spatial weight 0.3

Solution 2

(a) When I tried BSZ=50, lr = 0.075 and nHidden = 1024, I got the best performance, the training accuracy reached 100% and Val accuracy reached 97.70% after 50 epoches. As shown in the following screenshot.

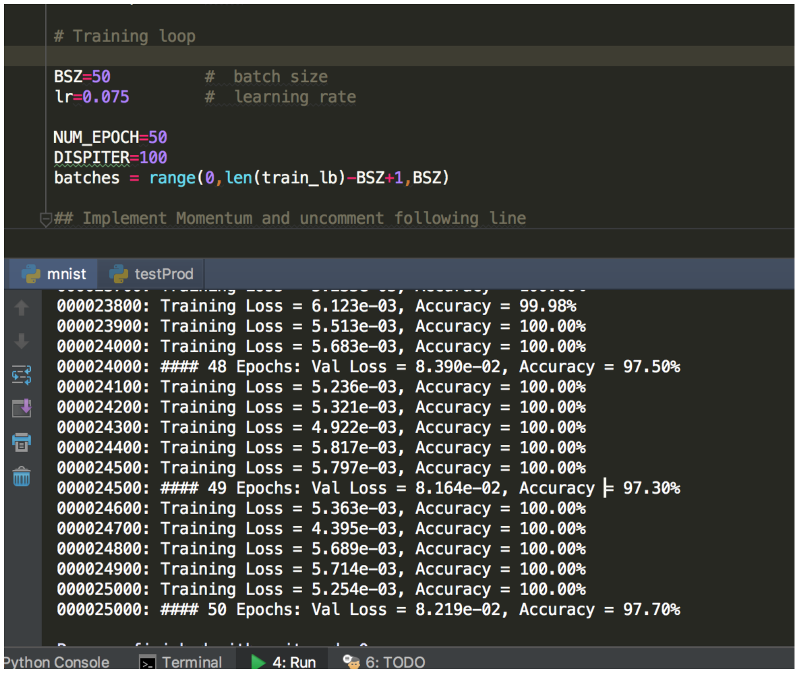


Figure 5: Training under BSZ=50, lr = 0.075 and nHidden = 1024

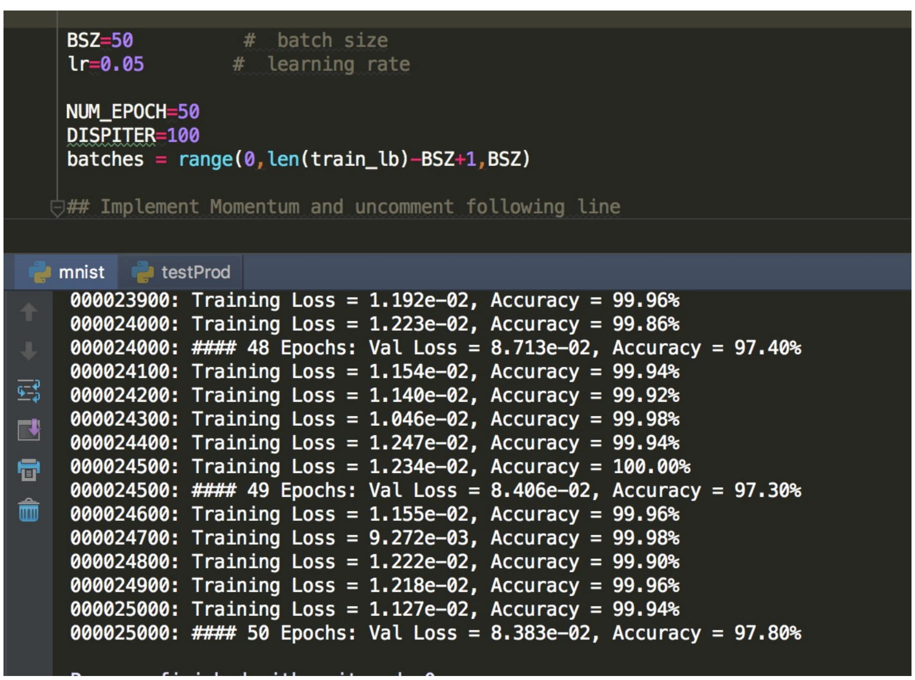


Figure 6: Training under BSZ=50, lr = 0.05 and nHidden = 1024

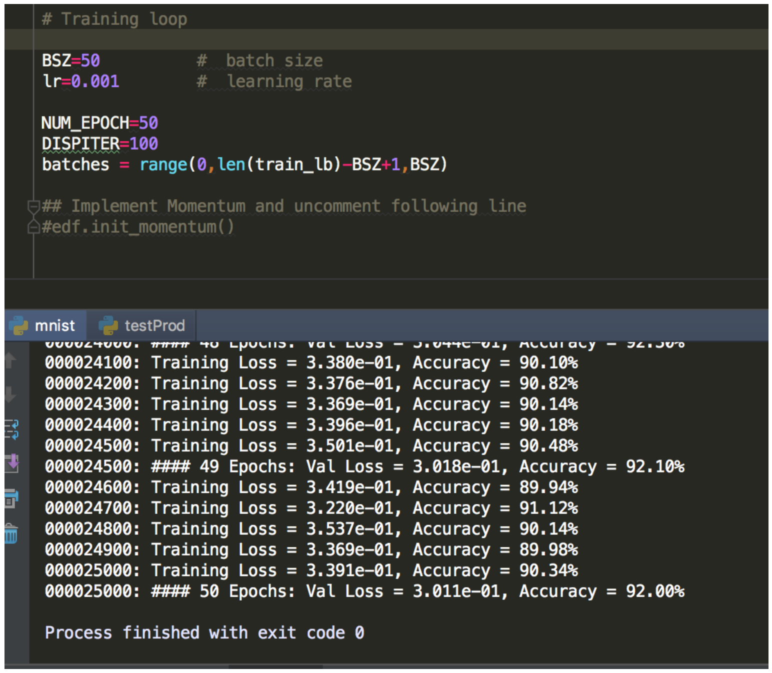


Figure 7: Training under BSZ=50, lr = 0.001 and nHidden = 1024

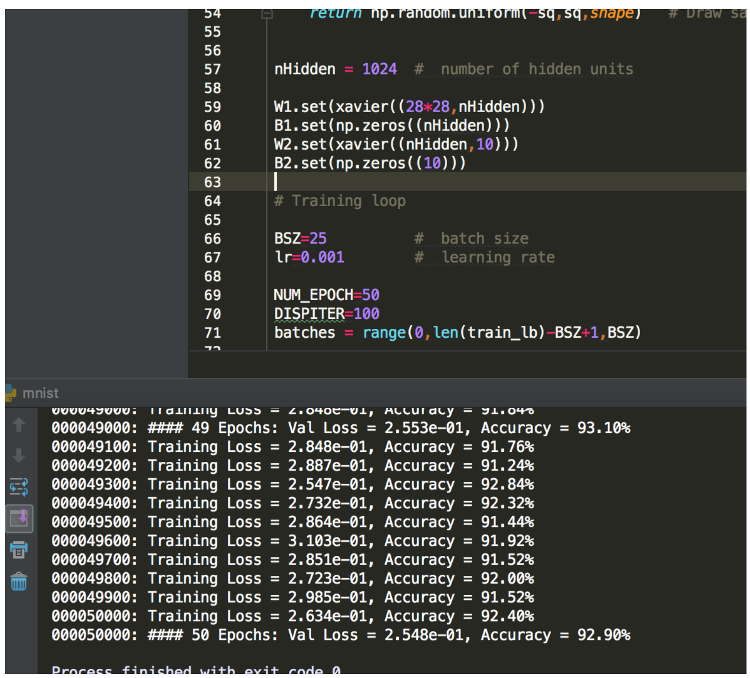


Figure 8: Training under BSZ=25, lr = 0.001 and nHidden = 1024

After several times of experiments, I verified the relationship between batch size and learning rate, that is smaller learning rate with smaller batch size, bigger learning rate with bigger batch size. Because with smaller batch size, after one batch training, program is less sure about the gradient, so "step size" (learning rate) should also be smaller. The number of hidden units represents the num of input features, the relation between classifier performance and number of hidden units is: less hidden units -> running faster -> lower accuracy; more hidden units -> running slower -> higher accuracy.

The limit of the uniform distribution being chosen in the way as function $xavier$ does is because: we have to make sure that weights are not too small and not too big to propagate accurately the signals.

(b) Implementing batched stochastic gradient descent with using momentum makes classifier training run faster and have higher accuracy. As showing in the following figures:

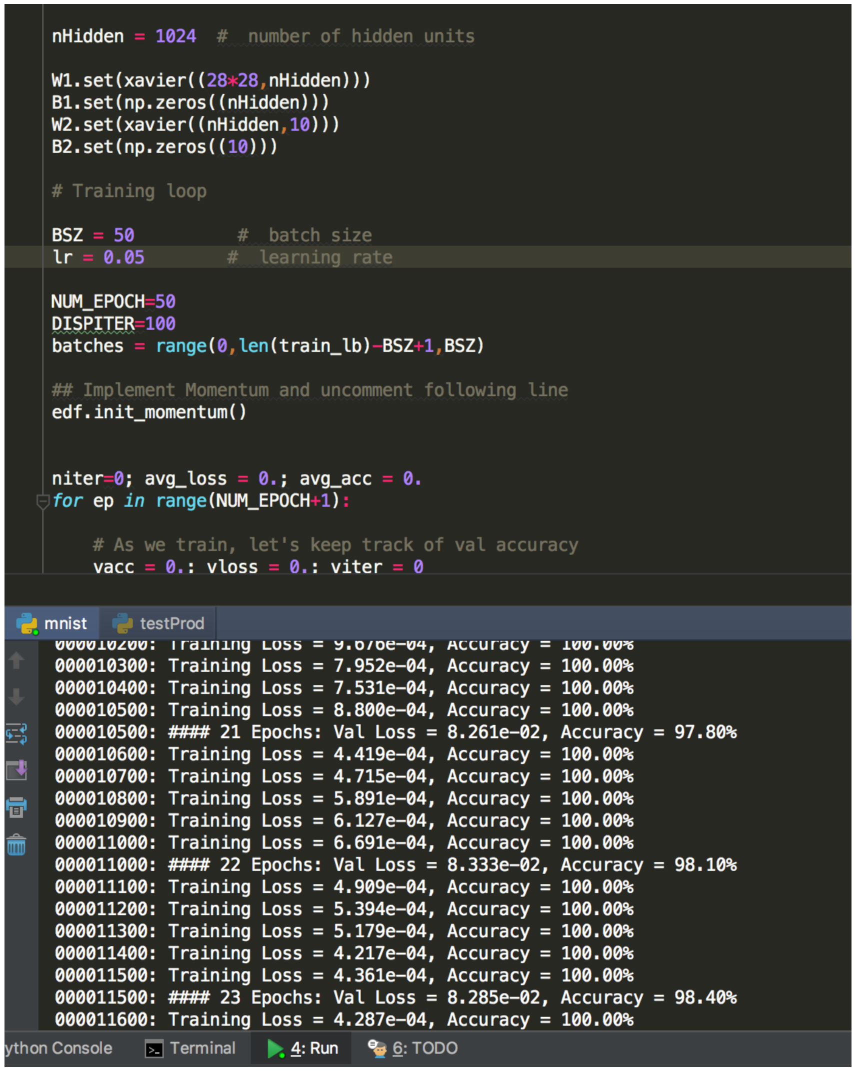
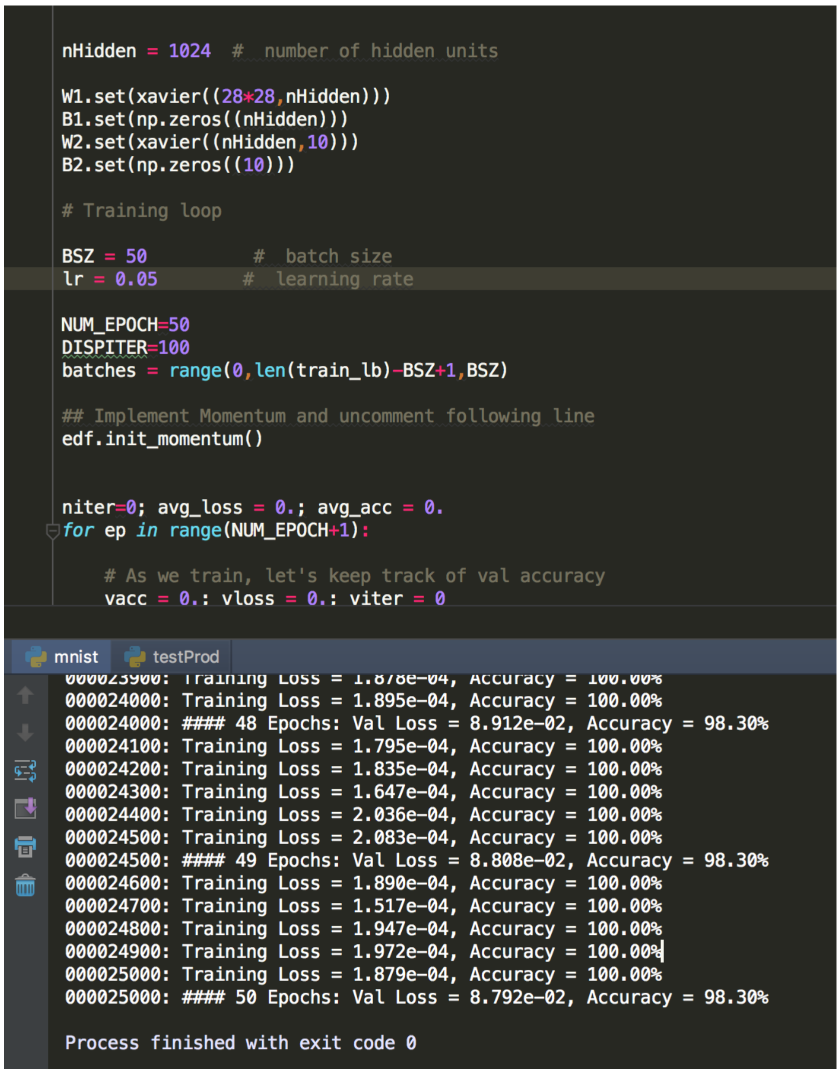


Figure 9: Processing: Training with momentum under BSZ=50, lr = 0.05 and nHidden = 1024



```python
nHidden = 1024  #  number of hidden units

W1.set(xavier((28*28,nHidden)))
B1.set(np.zeros((nHidden)))
W2.set(xavier((nHidden,10)))
B2.set(np.zeros((10)))

# Training loop

BSZ = 50           #  batch size
lr = 0.05         #  learning rate

NUM_EPOCH=50
DISPITER=100
batches = range(0,len(train_lb)-BSZ+1,BSZ)

## Implement Momentum and uncomment following line
edf.init_momentum()


niter=0; avg_loss = 0.; avg_acc = 0.
for ep in range(NUM_EPOCH+1):

    # As we train, let's keep track of val accuracy
    vacc = 0.; vloss = 0.; viter = 0
```



```
000024000: Training Loss = 1.895e-04, Accuracy = 100.00%
000024000: #### 48 Epochs: Val Loss = 8.912e-02, Accuracy = 98.30%
000024100: Training Loss = 1.795e-04, Accuracy = 100.00%
000024200: Training Loss = 1.835e-04, Accuracy = 100.00%
000024300: Training Loss = 1.647e-04, Accuracy = 100.00%
000024400: Training Loss = 2.036e-04, Accuracy = 100.00%
000024500: Training Loss = 2.083e-04, Accuracy = 100.00%
000024500: #### 49 Epochs: Val Loss = 8.808e-02, Accuracy = 98.30%
000024600: Training Loss = 1.890e-04, Accuracy = 100.00%
000024700: Training Loss = 1.517e-04, Accuracy = 100.00%
000024800: Training Loss = 1.947e-04, Accuracy = 100.00%
000024900: Training Loss = 1.972e-04, Accuracy = 100.00%
000025000: Training Loss = 1.879e-04, Accuracy = 100.00%
000025000: #### 50 Epochs: Val Loss = 8.792e-02, Accuracy = 98.30%

Process finished with exit code 0
```

Figure 10: Finished: Training with momentum under BSZ=50, lr = 0.05 and nHidden = 1024

Solution 3

The training results with one convolution layer:

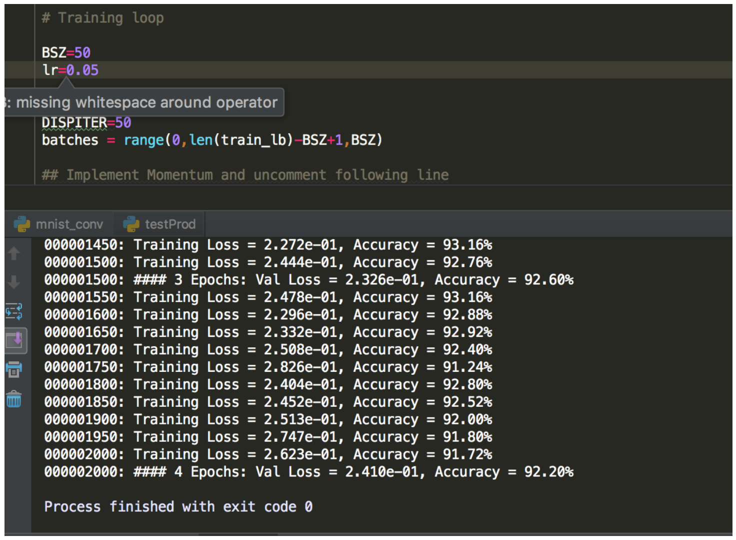


Figure 11: Processing: Training with momentum under BSZ=50, lr = 0.05 and C1 = 64

The training results with two convolution layers:

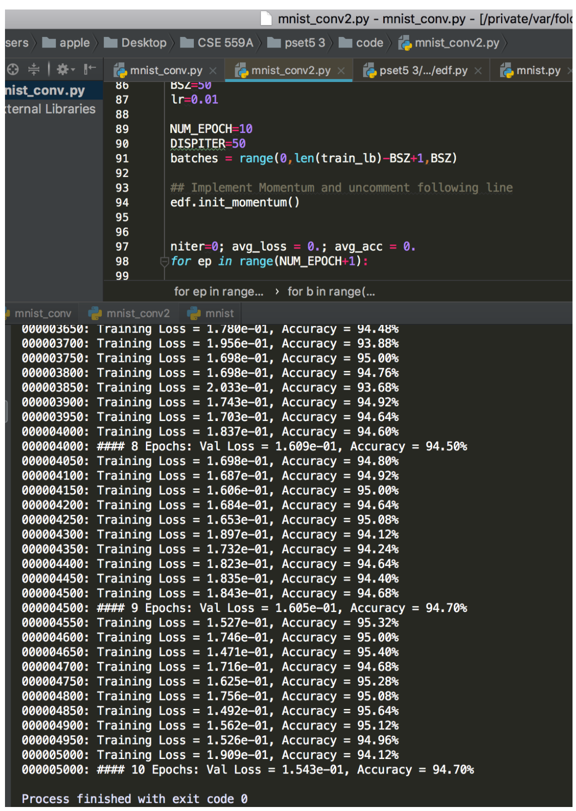


Figure 12: Processing: Training with momentum under BSZ=50, lr = 0.05

Information

This problem set took approximately 50 hours of effort.

I discussed this problem set with:

- Sijia Wang
- Jiarui Xing
- Ruxin Zhang
- Chunyuan Li

I also got hints from the following sources:

- lecture ppts
- http://jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/
- http://blog.csdn.net/zhongkejingwang/article/details/44514073