Q1.1

Q1.1.1 ReLU activation function

The first reason is that when we use sigmoid function as activation function, the gradient of sigmoid is among (0,1) so that the error will decrease significantly when back propagating. The learning speed of the network will be very low. The gradient will vanish. However, If we use ReLU as activation function, the problem would be resolved.

Q1.1.2

If the activation function g is linear,

The output will be

$$g(w_k(\cdots(g(w_3(g(w_2(g(w_1x+b_1))+b_2))+b_3))+\cdots)+b_k)$$

Since g is linear function, $g(w_ix+b_i)$ can be written as W_ix+B_i

The output will be

$$W_k(\cdots(W_3(W_2(W_1x+B_1)+B_2)+B_3)+\cdots)+B_k$$

Finally, we will get a linear combination of the input

$$W_{1-k}x+B_{1-k}$$

And any network with different number of hidden layers will give us the same representation capacity.

Q 2.1.1

If we initialize the network with all zero or all constant value, the update for each hidden layer would be the same when we do back propagation. So we can't start with all the weights to be the same constant.

Q 2.1.3

I initialize the weights and bias to be randomly distributed between -1 and 1 since I assume that all the weights and bias should have a mean of 0 and an initial value around 0. Also, the initial values can't be too close to zero, which would lead to a small update in the gradient. So randomly distributed from -1 and 1 should be a good choice.

Q 2.4.1

Stochastic gradient descent takes less time than batch gradient descent. But SGD doesn't always reduce the loss and move towards minimum because of the noise in the data.

Batch gradient descent takes longer time since it takes all the data to calculate updates at a time. And BGD always moves towards the minimum.

Q 2.5 Gradient Checker

The average error across layer 2 is 5.6706e-11.

Here is the sample output:

Error of W at layer 1 is
errorW =

1.5285e-12

Error of b(139) at layer 1 is

errorB =

1.0158e-14

total error =

1.5386e-12

3.1.2

0.01

Epoch 30 - accuracy: 0.89269, 0.75769 loss: 2770.66099, 2257.98440

Test accuracy: 0.7623

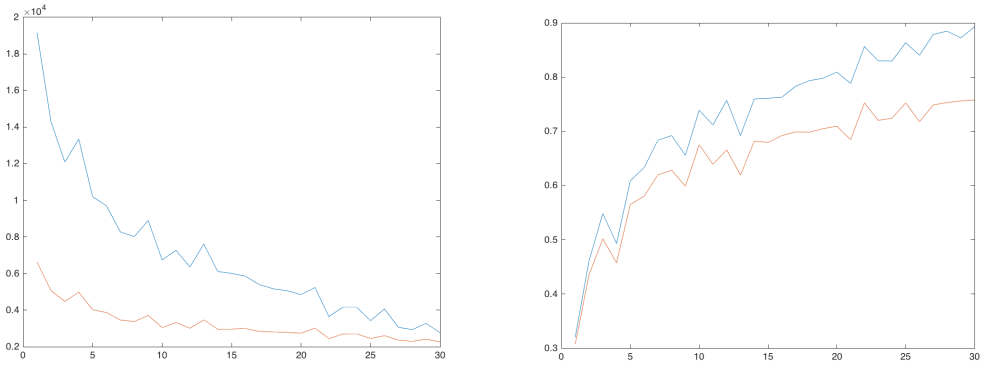

Figure X. loss and accuracy of 30 epochs with learning rate 0.01

0.001

Epoch 30 - accuracy: 0.70769, 0.61269 loss: 7834.28674, 3419.26142

Test accuracy: 0.6162

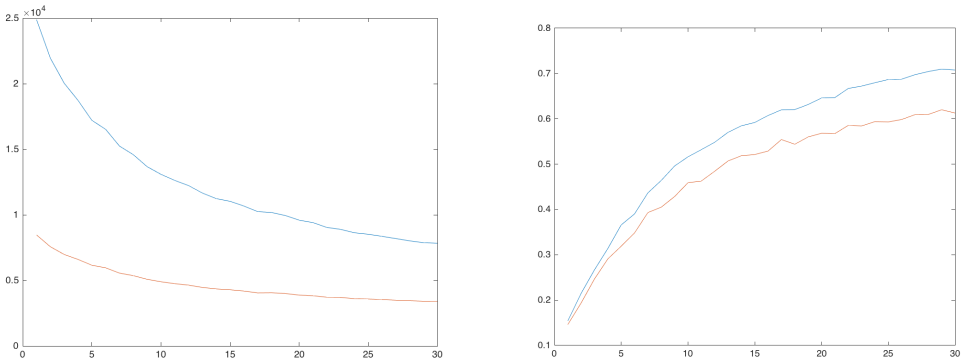

Figure X. loss and accuracy of 30 epochs with learning rate 0.001

A relatively large learning rate will speed up the learning process. As you can see in the figure, the validation accuracy increase to 0. 75769 after 30 epochs with learning rate 0.01 while the validation accuracy of learning rate 0.001 is 0. 61269.

But a relatively large learning rate will also result in a vibration in accuracy. The plot of 0.001 is much smoother than the plot of 0.01, which means that a relatively large learning rate may ‘over-step’ the gradient descent.

Q3.1.3
The best network is the one with learning rate 0.01.
Accuracy on test set is 76.23% and loss is 1143.8.

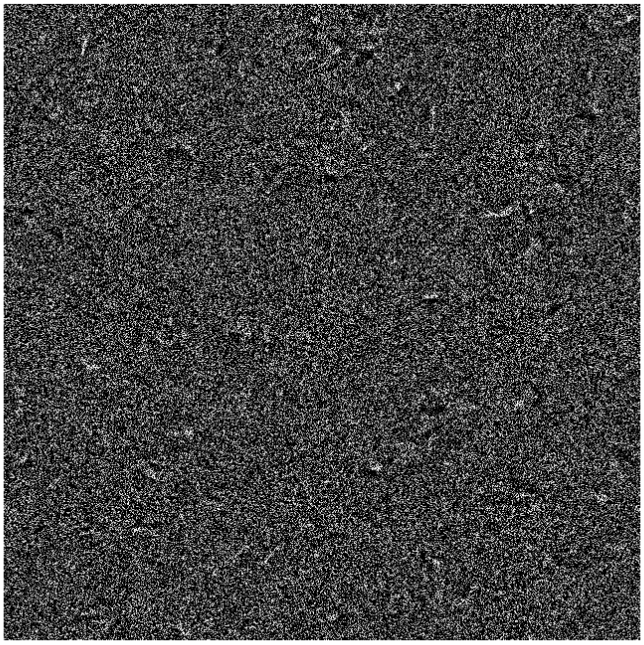

Figure X. First Layer Weights after Training

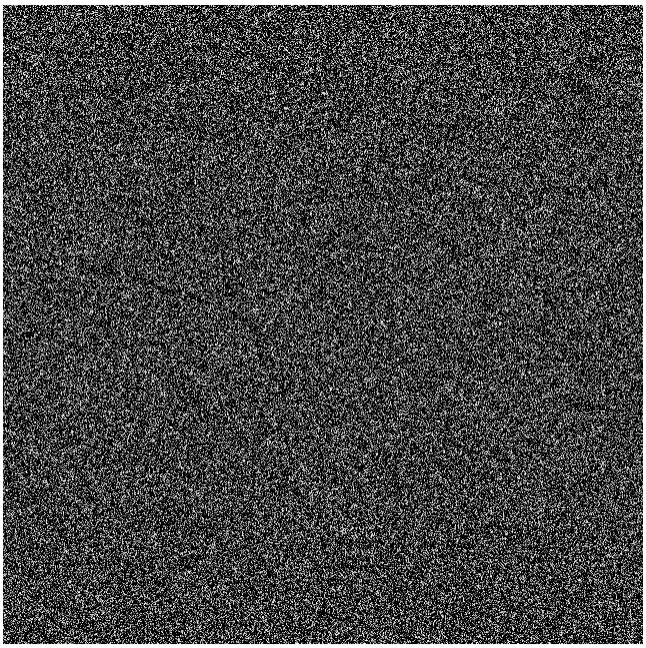

Figure X. First Layer Weights before Training

Comments:

I can't see any patterns on the weights before training since they were randomly generated. However, there are some 'filter-like' trained patterns on the weights after training.

Q 3.1.4

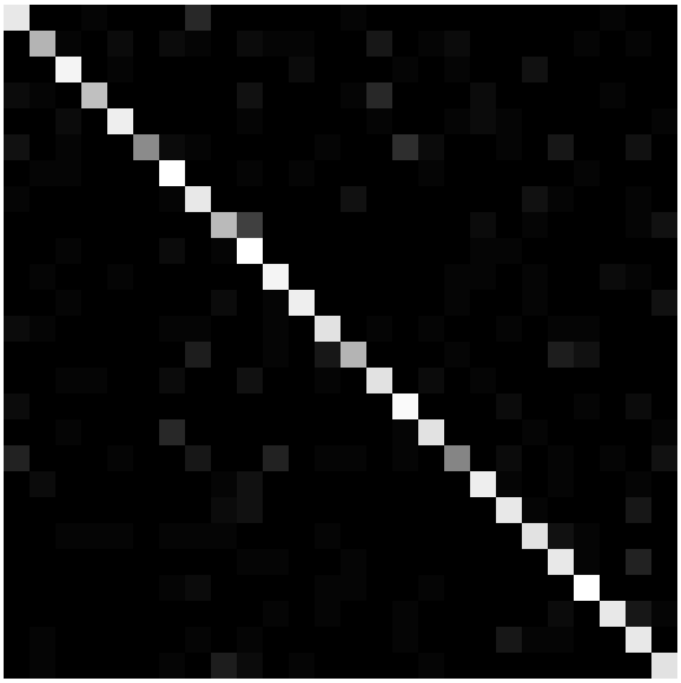

Figure X. Visualization of Confusion Matrix

The brighter the block, more accurate the estimate is. The 6^{th} and 18^{th} block is the least bright two.
For most confused elements, they are ‘R’ and ‘F’, the predicted accuracy is 23/50 and 24/50 respectively.

Q3.2.1

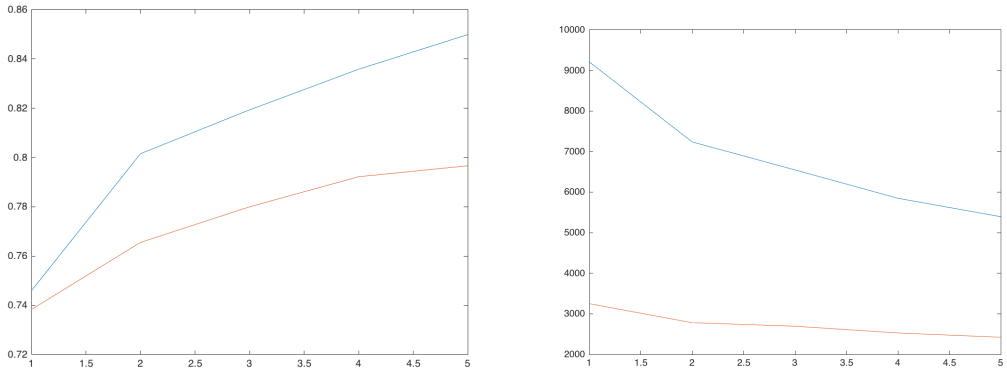

Figure X. Accuracy and Loss of 5 epochs fine-tuning with learning rate 0.01

Epoch 5 - accuracy: 0.84991, 0.79667 loss: 5393.61925, 2423.29743

Q3.2.2

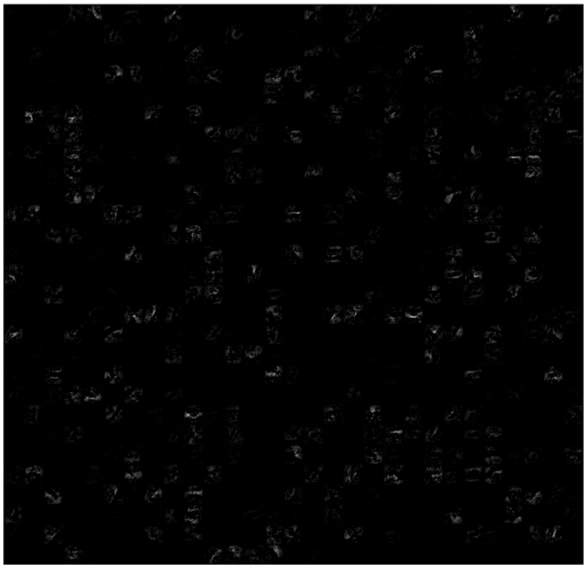

Figure X. First Layer Weights before Training

Figure X. First Layer Weights after Training

Comments:

There are existing patterns in the first layer of weights for pre-trained nist26 dataset. After we trained for 5 epochs, there is some gentle change of the weights to accommodate nist36 dataset.

Accuracy of test data is 80.17% and loss is 1137.9

3.2.3

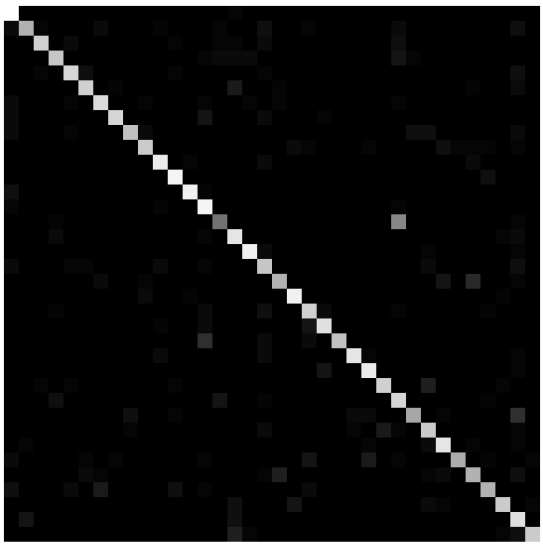

Figure X. Visualization of Confusion Matrix

The most confused entry is 15th and 28th entry, which represent ‘O’ and ‘1’ respectively. The original test accuracy of ‘nist26_model_60iters’ is 86%. However, after adding classes and fine-tuning, the accuracy decreases to 80.17%. Adding classes to fine-tuned network will decrease the accuracy.

Q4.1

Assumptions:

1. The letters are separate from each other.
2. The text is without rotation.

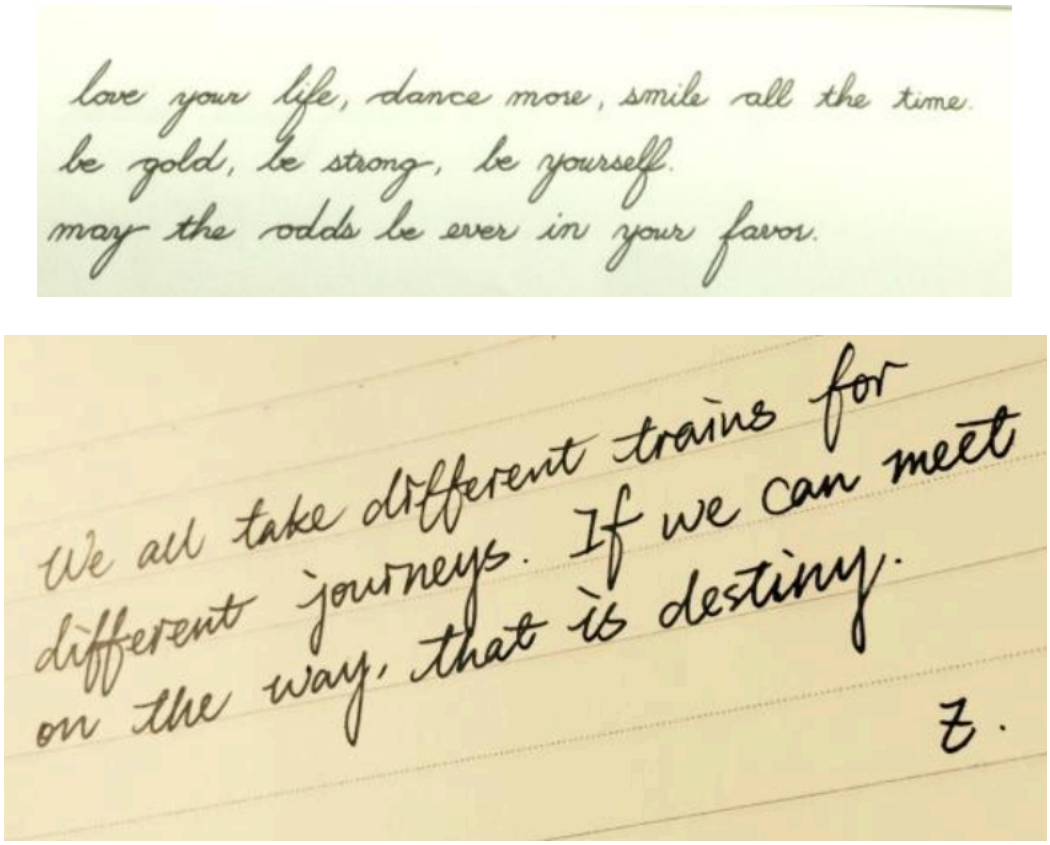

Figure X. Two Examples of Possible Failures

Q4.3

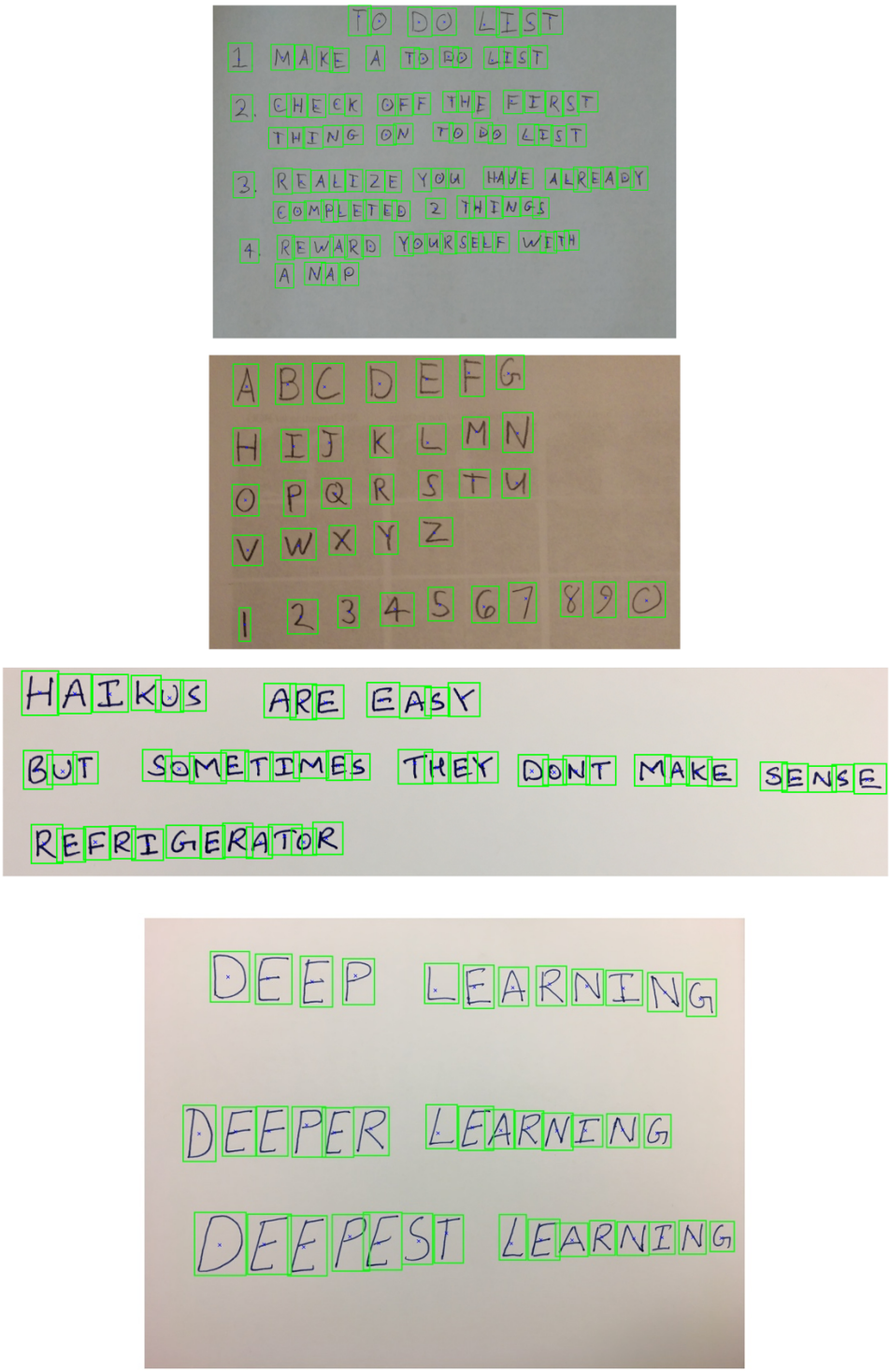

Figure X. Results for find_letters

Q4.5

I use hierarchical clustering to separate the lines.

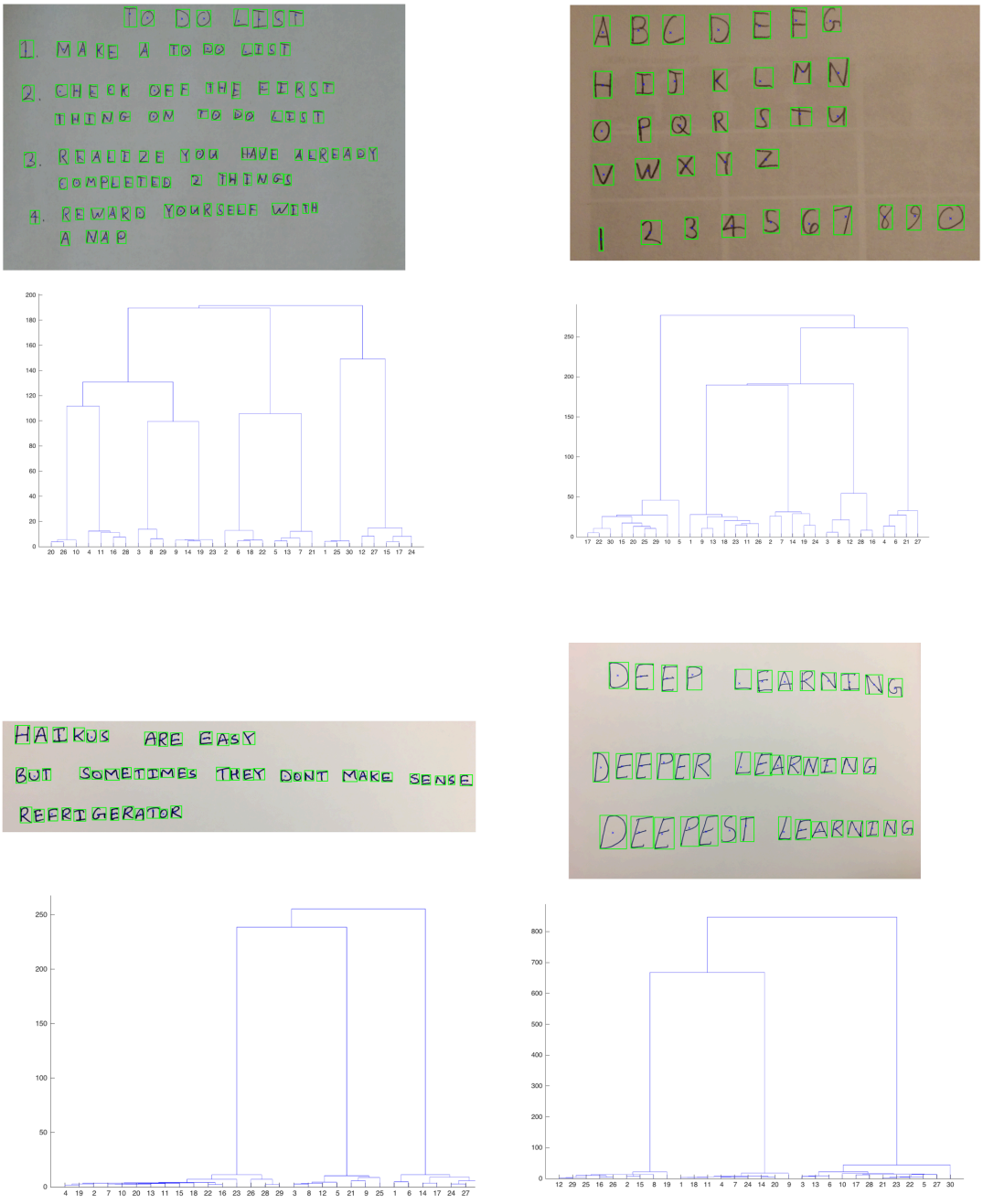

Figure X. Hierarchical Clustering

Letters:

01

'TT000L55'
'LUAKRAT0N0LJ5T'
'LRPRKKU88YNKFIR5T'
'YP8N50NT0D0LX5T'
'3RRAL8Z8Y0UAV8XKK8ADY'
'R0MPL8TLD2THXNF5'
'9R8WXRDY0UR58L8NITH'
'ANAP'

02

'ABL0EFG'
'HIJKLMN'
'0PQK5TU'
'VWXYZ'
'8X3FSQY8P0'

03

'HAIKUSAREEASX'
'BUTS0METIMESTHEYD0NTMAKESENGE'
'REFRIGERAT0R'

04

'P55PLRARN5NN'
'DFPP8RLEARNING'
'58FP8STL8ARNING'

Q5.1

The filter size, input and output channels and filters are wrong. Each layer's output should be the same as next layer's input and each layer's parameter should meet the handout's specification.

Q5.2.1

For the training process without and adjustment, the system seems not training as all. NaN(Not a Number) appears and no results show up.

Q5.2.2

Adjustment:

1.Learning Rate is too high. So I adjust the learning rate to 1e-3.

2. I set a new parameter 'L2Regularization',0.0005

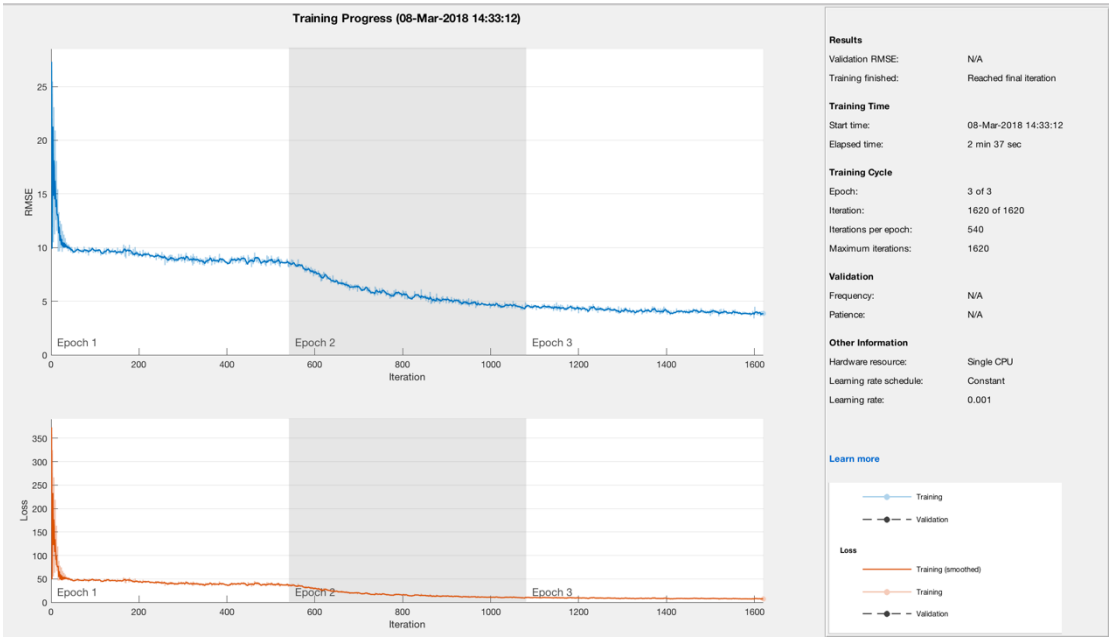

Figure X. Training Progress

Q5.3.1

Figure X. A

Figure X. B

Figure X. C

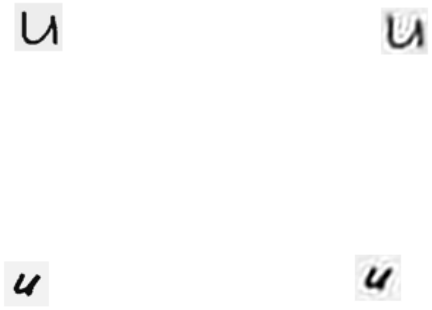

Figure X. U

Figure X. 5

Q5.3.2

1781
1788
1014
1011
1599
1600
130
110
59
51
10
2

For “A”
PSNR = 19.9635
PSNR = 17.0946
For “B”
PSNR = 15.1516
PSNR = 15.4522
For “C”
PSNR = 19.6373
PSNR = 18.9821
For “U”
PSNR = 20.1326
PSNR = 17.8575
For “5”
PSNR = 21.0260
PSNR = 17.8013

Q5.4.1
Size of projection matrix is 1024*(target reduced dimensions)
Here it is 1024 * 64.

Q5.4.2


PSNR: 20.4611 PSNR = 19.9635


PSNR: 16.6350 PSNR = 17.0946

Figure X. PCA (Left) and Network (Right)


PSNR: 15.8033 PSNR = 15.1516


PSNR: 15.4423 PSNR = 15.4522

Figure X. PCA (Left) and Network (Right)


PSNR: 20.2676 PSNR = 19.6373


PSNR: 19.6701 PSNR = 18.9821

Figure X. PCA (Left) and Network (Right)


PSNR: 21.0588 PSNR = 20.1326


PSNR: 18.0153 PSNR = 17.8575

Figure X. PCA (Left) and Network (Right)

PSNR: 18.7156 PSNR = 17.8013

PSNR: 21.8608 PSNR = 21.0260

Figure X. PCA (Left) and Network (Right)

Comment: Although the PSNR is close to each other, it seems that the reconstructed image from PCA is more clear than the reconstructed image from auto encoder.

Q5.4.3
See the PSNR below the figures.

The PSNR from PCA is mostly larger than PSNR from trained network, which means that PCA is better than auto encoder. The larger the PSNR, more similar the image to its original image.

Q5.4.5
The parameters for PCA is 1024*64.
The parameter for auto encoder is a net work weights and biases, which is smaller than parameter of PCAs.