

CpE 520: HW#9

West Virginia University

Ali Zafari

Table of Contents

0. MNIST Dataset	4
0.1 Downloading MNIST Dataset and Import into <i>Matlab</i>	4
0.2 Plotting a Sample of MNIST Dataset	4
1. 3-Layer Autoencoder with Different Hidden Units	6
1.1 Reshapinng the Images	6
1.2 Autoencoder with 5 Hidden Units	7
1.3 Autoencoder with 10 Hidden Units	9
1.4 Autoencoder with 30 Hidden Units	11
1.5 Autoencoder with 60 Hidden Units	13
2. Display Encoder Weights	15
2.1 Weights of Autoencoder with 5 Hidden Units	15
2.2 Weights of Autoencoder with 10 Hidden Units	16
2.3 Weights of Autoencoder with 30 Hidden Units	17
2.4 Weights of Autoencoder with 60 Hidden Units	18
3. Classifiaction Using Encoded Images	19
3.1 One Hot Labels	19
3.1 Encoded Images	19
3.2 Softmax Layer	19
3.3 Classifier Networks	19
3.3.1 Classifier with 5 Features	20
3.3.1 Classifier with 10 Features	20
3.3.1 Classifier with 30 Features	20
3.3.1 Classifier with 60 Features	20
3.4 Classification Performance Versus Number of Hidden Units	21
4. Confusion Matrices	23
4.1 Confusion Matrix for 5 Unit Classifier	23
4.2 Confusion Matrix for 10 Unit Classifier	24
4.3 Confusion Matrix for 30 Unit Classifier	25
4.4 Confusion Matrix for 60 Unit Classifier	26
5. Encoder Weights Comparing to PCA Eigenvectors and K-means Centroids	27
Appendix	30
A.1 Saving Workspace Variables for Future Use	30
A.2 Defiition of Auxiliary Functions	30

In this homework assignment, an autoencoder with different number of unit in its hidden layer is trained and evaluated on MNIST handwritten digit dataset.

```
1  clc; clear; close all;  
2  load('HW9_code_workspace.mat');
```

0. MNIST Dataset

0.1 Downloading MNIST Dataset and Import into *Matlab*

The code snippet below will download MNIST dataset from web and extract the downloaded file into Matlab matrices as test and train images and labels.

```
1 mnist_train_image = 'train-images-idx3-ubyte';
2 mnist_train_label = 'train-labels-idx1-ubyte';
3 mnist_test_image  = 't10k-images-idx3-ubyte';
4 mnist_test_label  = 't10k-labels-idx1-ubyte';
5 train_set_number  = 60000;
6 test_set_number   = 10000;
7
8 downloadMNIST(mnist_train_image, mnist_train_label, mnist_test_image,
               mnist_test_label);
```

```
MNIST dataset already downloaded.
```

```
1 [train_images, train_labels] = readMNIST(mnist_train_image, mnist_train_label,
      train_set_number);
2 [test_images,  test_labels] = readMNIST(mnist_test_image,  mnist_test_label,
      test_set_number);
3
4 clear mnist_train_image mnist_train_label mnist_test_image mnist_test_label
      train_set_number test_set_number
```

0.2 Plotting a Sample of MNIST Dataset

A set of 100 randomly chosen samples of the training dataset is plotted here.

```
1 random_indices = randi(60000, [1 100]);
2 montage(train_images(:, :, random_indices), 'BorderSize', [2 2], '
      BackgroundColor', 'white');
```

0	1	9	1	9	0	4	4	8	1
9	3	9	5	4	4	9	4	8	9
1	6	6	8	7	5	0	0	1	3
9	4	9	4	7	9	6	5	1	2
7	7	2	6	5	4	0	5	0	3
4	3	9	5	5	7	2	5	4	3
2	2	8	4	9	4	7	3	8	9
2	1	6	8	8	0	0	0	2	7
9	5	8	8	4	7	8	2	5	9
0	2	2	4	8	3	1	3	1	5

1. 3-Layer Autoencoder with Different Hidden Units

In this section we train an autoencoder with 5, 10, 30 and 60 units in its hidden layer.

1.1 Reshaping the Images

To train the network, we reshape the 28x28 images to vectors of 784 element, to feed them into the input layer with the same number of units (784).

```
1 train_images_reshaped = reshape(train_images, [size(train_images, 1)*size(
    train_images, 2) size(train_images, 3)]);
2 test_images_reshaped  = reshape(test_images,  [size(test_images, 1)*size(
    test_images, 2) size(test_images, 3)]);
```

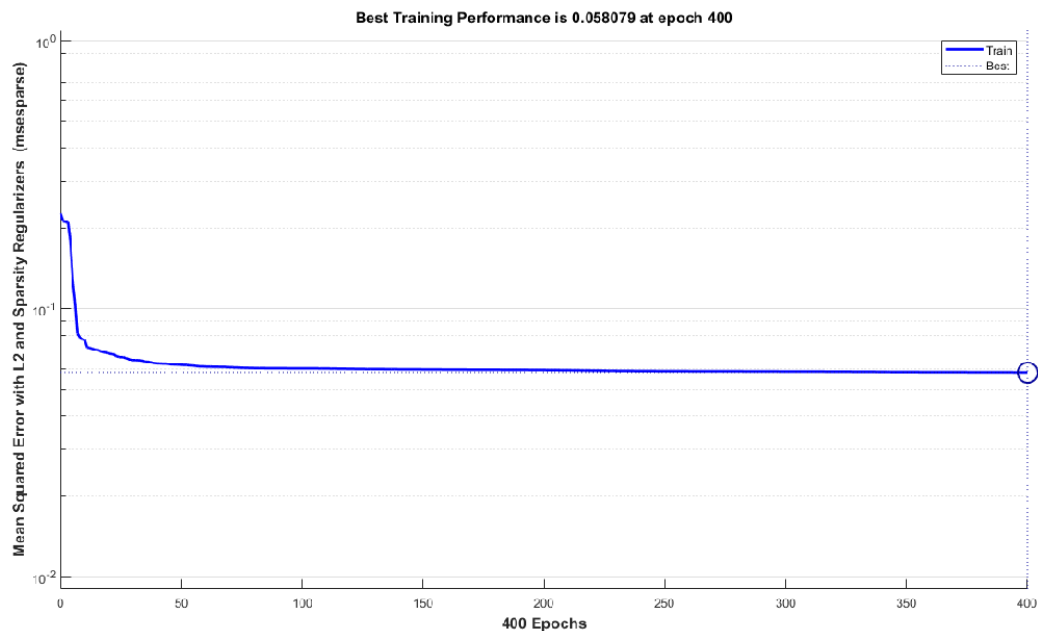
We also sampled 16 images of the test dataset to check the reconstruction performance of each autoencoder visually.

```
1 sample_original_images = train_images_reshaped(:, [11 3 2 33 5 24 22 1 85 13
    100:105]);
```

1.2 Autoencoder with 5 Hidden Units

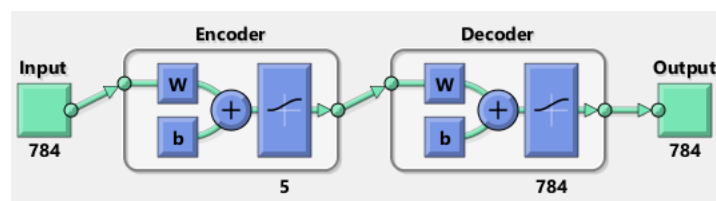
We trained a 3-layer autoencoder with a hidden layer consisting of 5 hidden units. The learning curve is plotted below.

```
1 autoenc_5 = trainAutoencoder(train_images_resaped , 5, ...  
2                               'MaxEpochs' , 400, ...  
3                               'L2WeightRegularization' , 0.004, ...  
4                               'SparsityRegularization' , 4, ...  
5                               'SparsityProportion' , 0.15);
```



To view the structure of the network we can get help from built-in functions of Matlab.

```
1 view(autoenc_5);
```

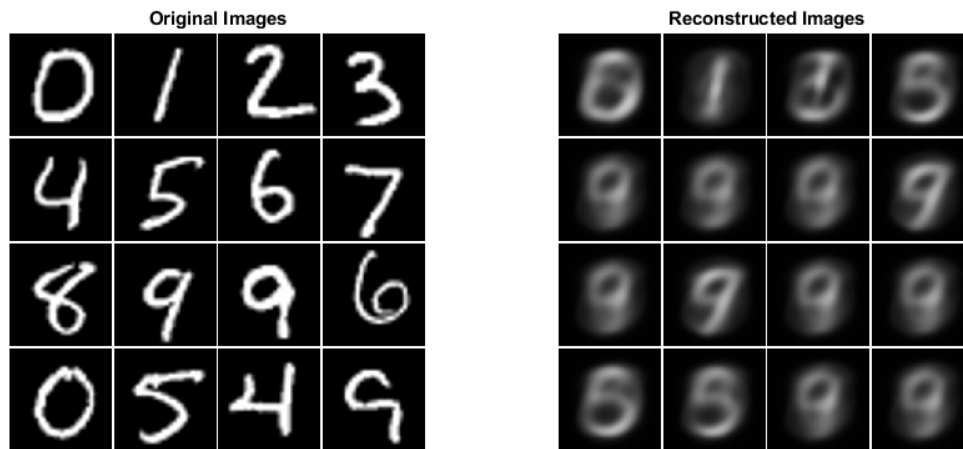


16 sample of the test images (which were never presented to the autoencoder in training process) are chosen to feed the autoencoder so we can visually evaluate the reconstruction performance of the trained autoencoder.

```

1 reconstructed_images_5 = predict(autoenc_5 , sample_original_images);
2 display_original_images_vs_reconstructed(sample_original_images ,
      reconstructed_images_5);

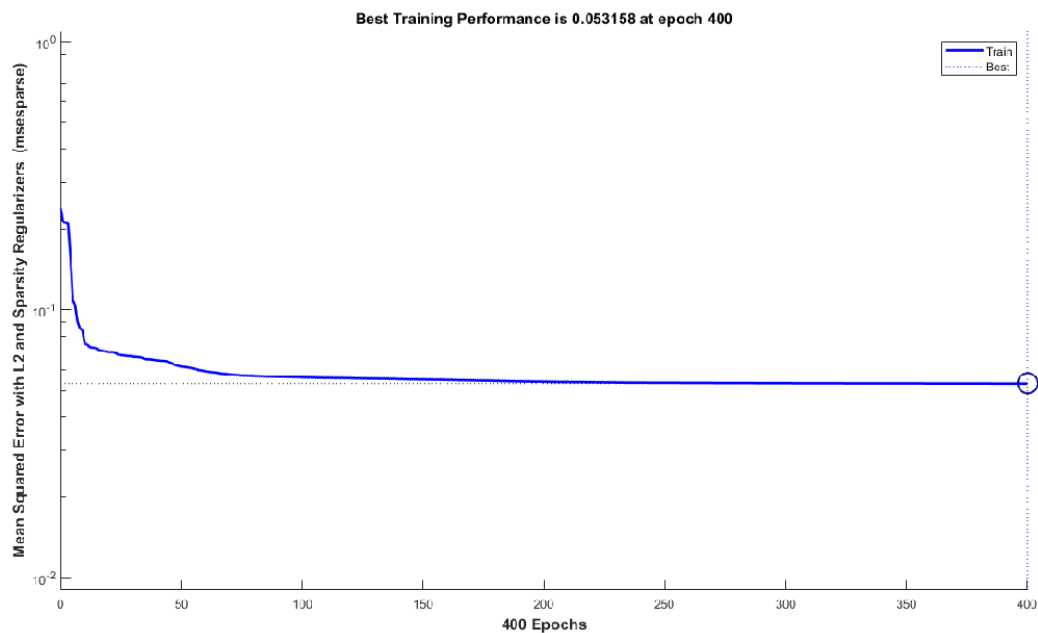
```



1.3 Autoencoder with 10 Hidden Units

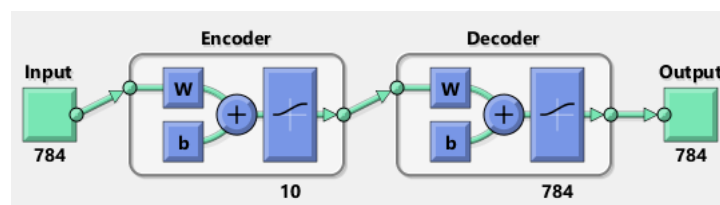
We trained a 3-layer autoencoder with a hidden layer consisting of 10 hidden units. The learning curve is plotted below.

```
1 autoenc_10 = trainAutoencoder(train_images_resaped , 10, ...  
2                               'MaxEpochs' , 400, ...  
3                               'L2WeightRegularization' , 0.004, ...  
4                               'SparsityRegularization' , 4, ...  
5                               'SparsityProportion' , 0.15);
```



To view the structure of the network we can get help from built-in functions of Matlab.

```
1 view(autoenc_10);
```

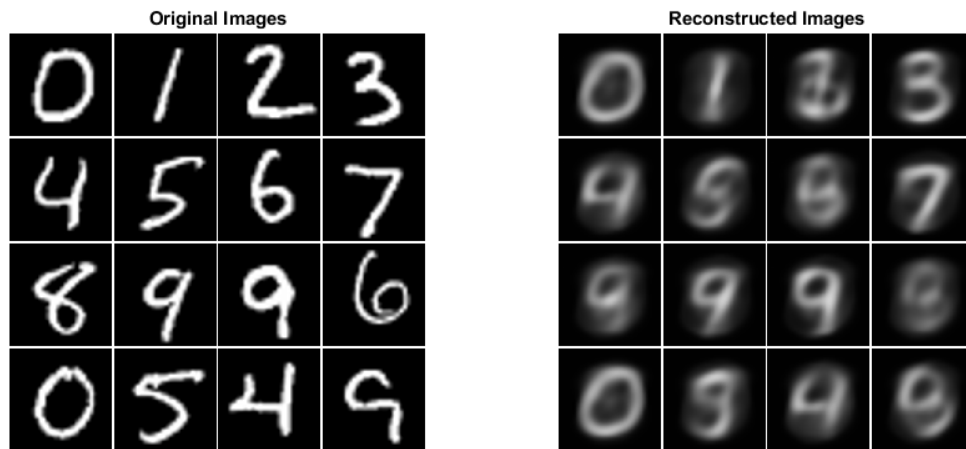


16 sample of the test images (which were never presented to the autoencoder in training process) are chosen to feed the autoencoder so we can visually evaluate the reconstruction performance of the trained autoencoder.

```

1 reconstructed_images_10 = predict(autoenc_10, sample_original_images);
2 display_original_images_vs_reconstructed(sample_original_images,
    reconstructed_images_10);

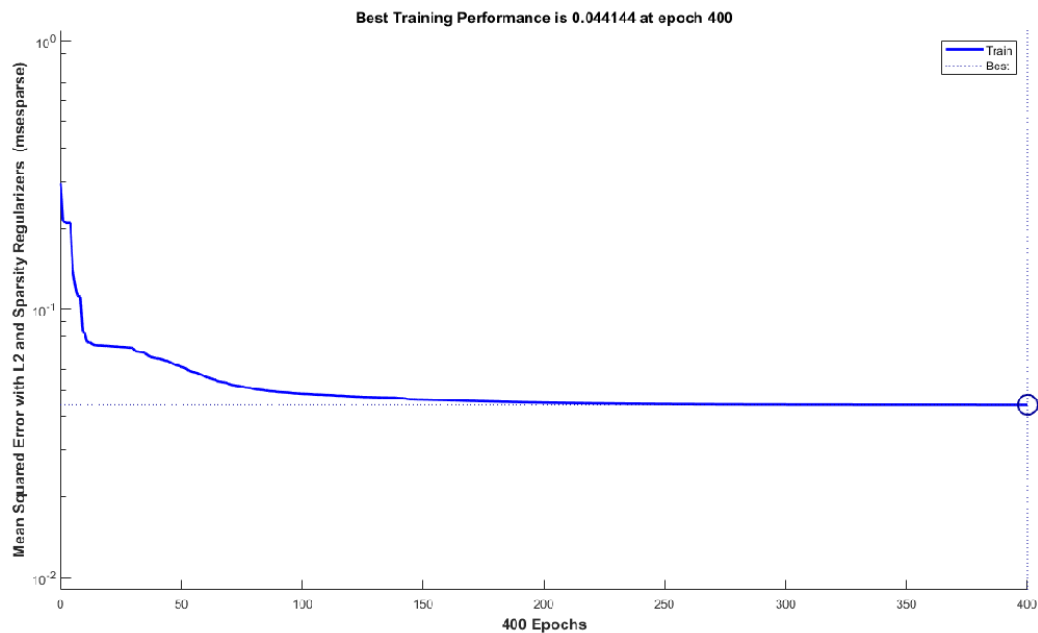
```



1.4 Autoencoder with 30 Hidden Units

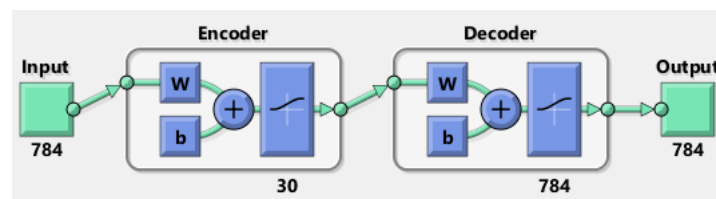
We trained a 3-layer autoencoder with a hidden layer consisting of 30 hidden units. The learning curve is plotted below.

```
1 autoenc_30 = trainAutoencoder(train_images_resaped , 30, ...  
2                               'MaxEpochs' , 400, ...  
3                               'L2WeightRegularization' , 0.004, ...  
4                               'SparsityRegularization' , 4, ...  
5                               'SparsityProportion' , 0.15);
```



To view the structure of the network we can get help from built-in functions of Matlab.

```
1 view(autoenc_30);
```

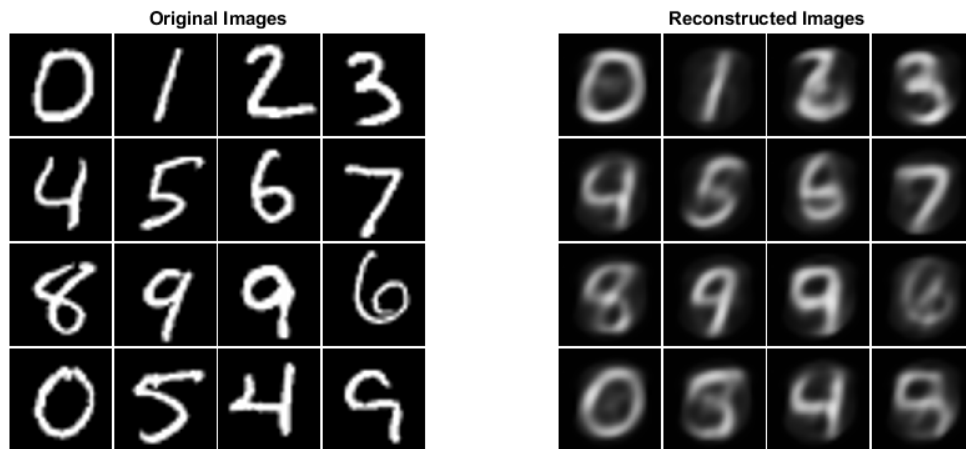


16 sample of the test images (which were never presented to the autoencoder in training process) are chosen to feed the autoencoder so we can visually evaluate the reconstruction performance of the trained autoencoder.

```

1 reconstructed_images_30 = predict(autoenc_30, sample_original_images);
2 display_original_images_vs_reconstructed(sample_original_images,
    reconstructed_images_30);

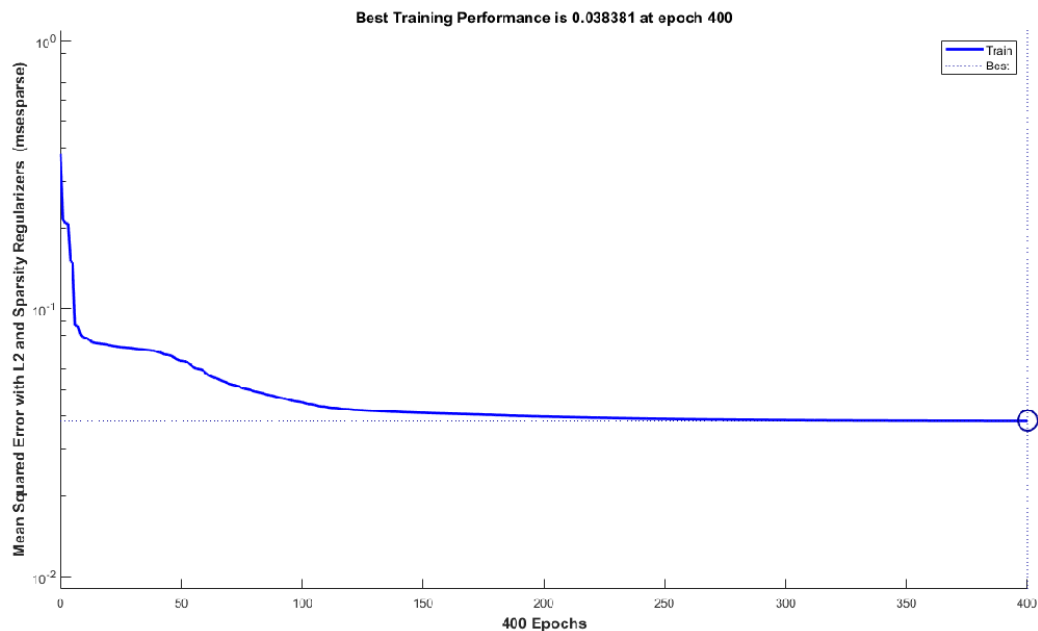
```



1.5 Autoencoder with 60 Hidden Units

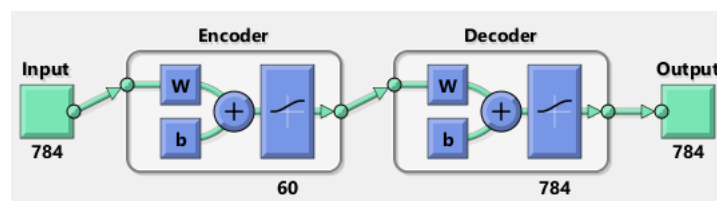
We trained a 3-layer autoencoder with a hidden layer consisting of 60 hidden units. The learning curve is plotted below.

```
1 autoenc_60 = trainAutoencoder(train_images_resaped, 60, ...
2                               'MaxEpochs', 400, ...
3                               'L2WeightRegularization', 0.004, ...
4                               'SparsityRegularization', 4, ...
5                               'SparsityProportion', 0.15);
```



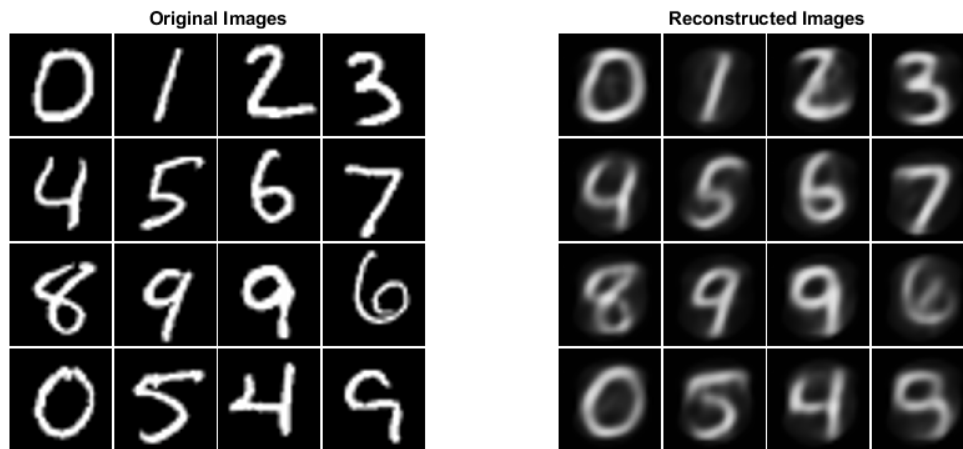
To view the structure of the network we can get help from built-in functions of Matlab.

```
1 view(autoenc_60);
```



16 sample of the test images (which were never presented to the autoencoder in training process) are chosen to feed the autoencoder so we can visually evaluate the reconstruction performance of the trained autoencoder.

```
1 reconstructed_images_60 = predict(autoenc_60, sample_original_images);
2 display_original_images_vs_reconstructed(sample_original_images ,
      reconstructed_images_60);
```

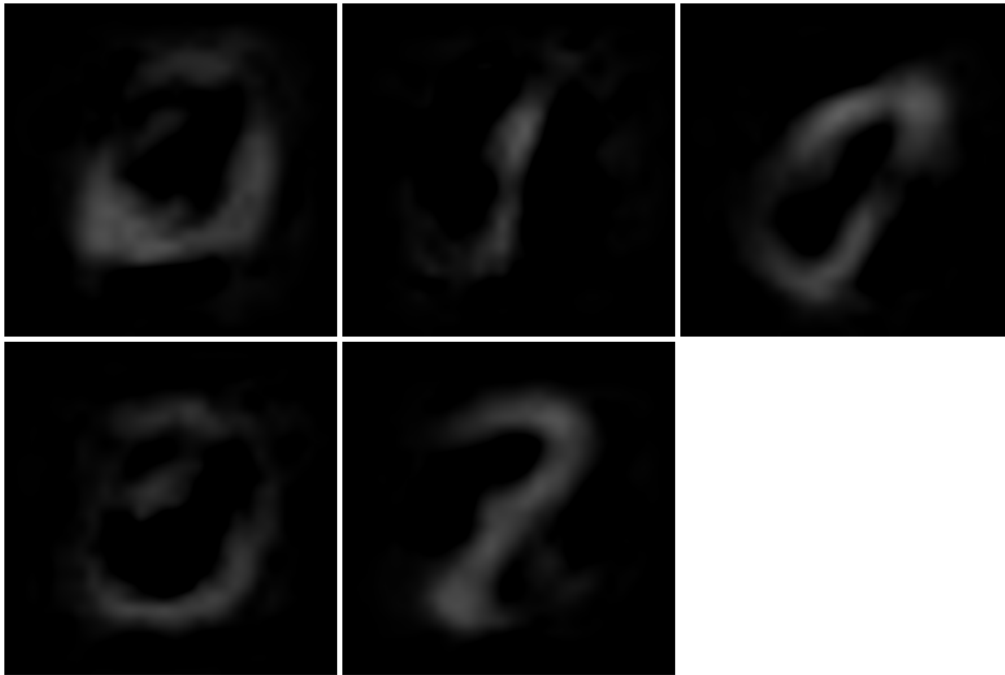


2. Display Encoder Weights

In this section, we have plotted the 784 weights of every node in the hidden layer as a 28x28 image.

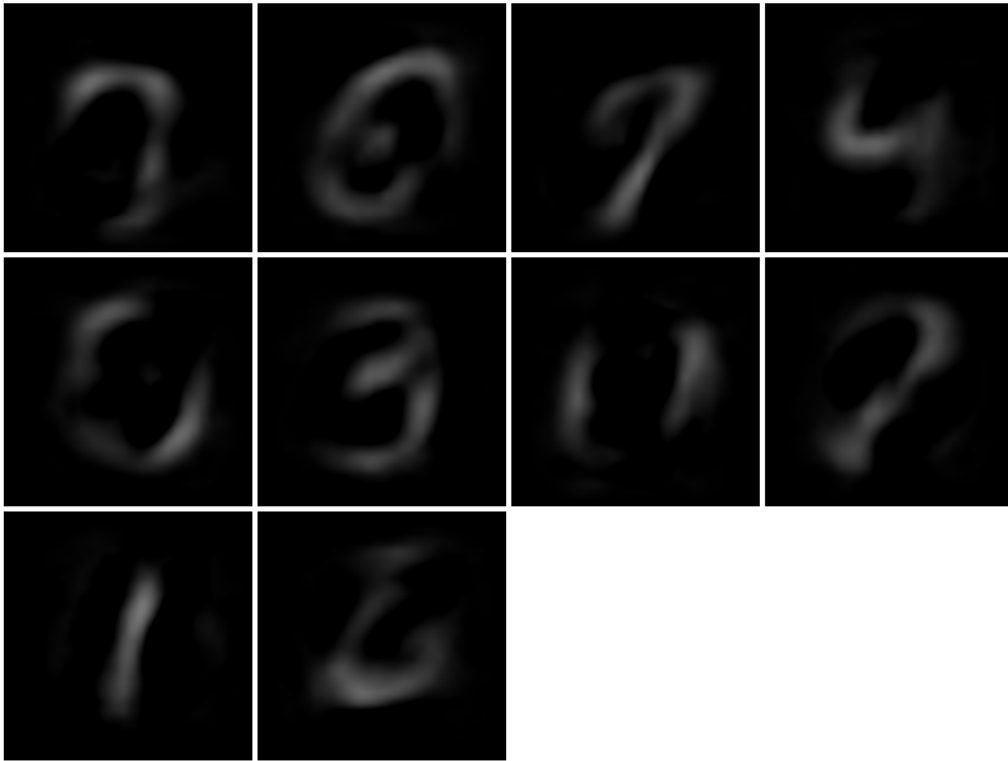
2.1 Weights of Autoencoder with 5 Hidden Units

```
1 iMontage(autoenc_5.EncoderWeights');
```



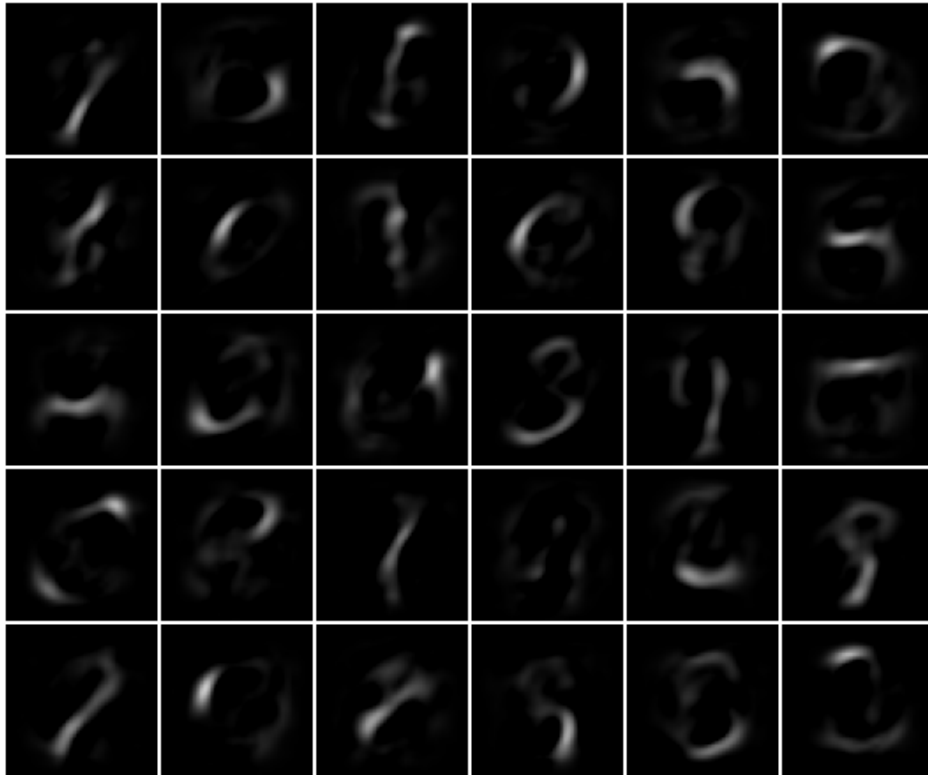
2.2 Weights of Autoencoder with 10 Hidden Units

```
1 iMontage(autoenc_10.EncoderWeights');
```



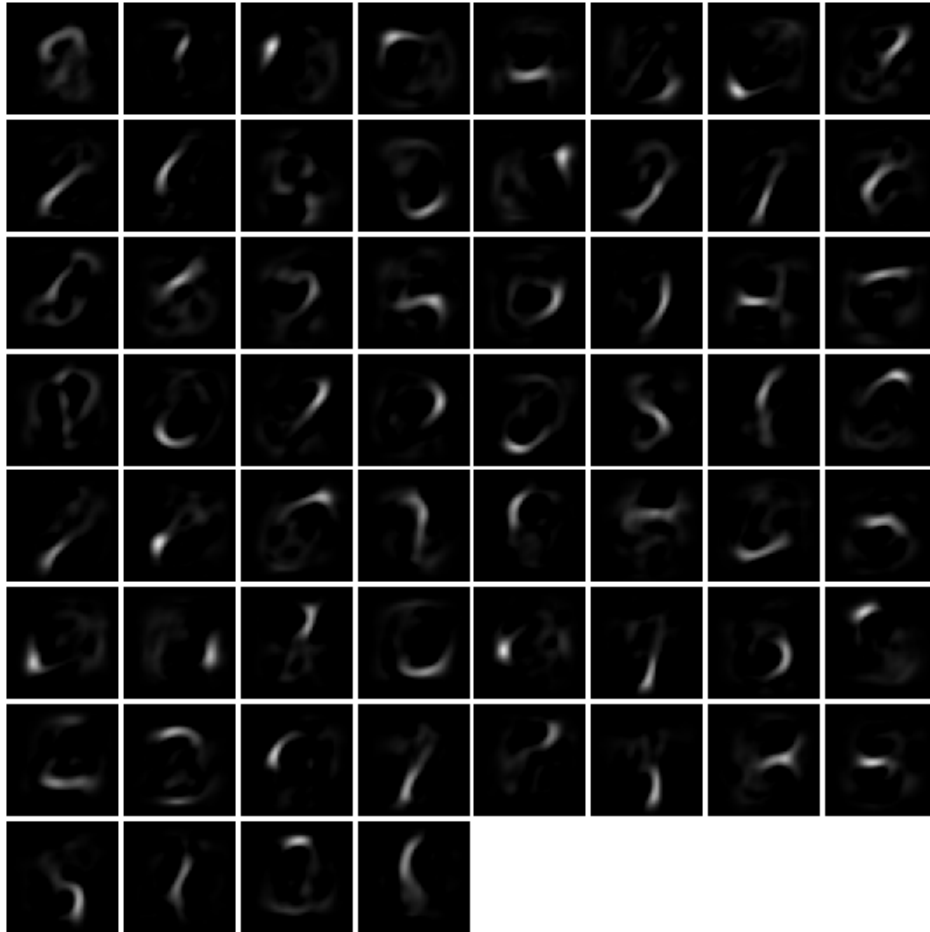
2.3 Weights of Autoencoder with 30 Hidden Units

```
1 iMontage(autoenc_30.EncoderWeights');
```



2.4 Weights of Autoencoder with 60 Hidden Units

```
1 iMontage(autoenc_60.EncoderWeights');
```



3. Classification Using Encoded Images

This section will use the encoded version of the images (which are derived from the hidden layer of the autoencoders in the previous section) to classify MNIST dataset.

3.1 One Hot Labels

The 2 line codes below will change the labels into a one-hot-encoded matrix.

```
1 train_labels_one_hot_encoded = full(ind2vec(train_labels '+1));
2 test_labels_one_hot_encoded  = full(ind2vec(test_labels '+1));
```

3.1 Encoded Images

Output of the hidden layer of every autoencoder are stored in the variables *features_5*, *features_10*, ..., *features_60*.

```
1 features_5  = encode(autoenc_5,  train_images_resaped);
2 features_10 = encode(autoenc_10, train_images_resaped);
3 features_30 = encode(autoenc_30, train_images_resaped);
4 features_60 = encode(autoenc_60, train_images_resaped);
```

3.2 Softmax Layer

Here we have trained only a softmax layer for classification. The inputs are the features extraced above and the outputs are the labels of evey image.

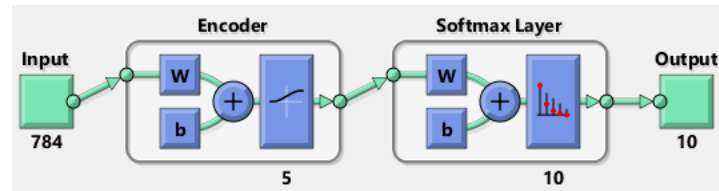
```
1 classifier_layer_5  = trainSoftmaxLayer(features_5 ,
      train_labels_one_hot_encoded , 'MaxEpochs' , 400);
2 classifier_layer_10 = trainSoftmaxLayer(features_10 ,
      train_labels_one_hot_encoded , 'MaxEpochs' , 400);
3 classifier_layer_30 = trainSoftmaxLayer(features_30 ,
      train_labels_one_hot_encoded , 'MaxEpochs' , 400);
4 classifier_layer_60 = trainSoftmaxLayer(features_60 ,
      train_labels_one_hot_encoded , 'MaxEpochs' , 400);
```

3.3 Classifier Networks

We now attach the trained softmax layer to the output of the hidden layer of the previously trained autoencoders.

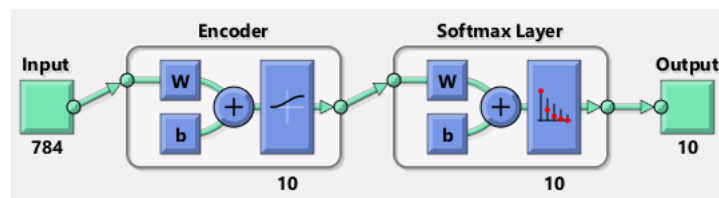
3.3.1 Classifier with 5 Features

```
1 stackednet_5 = stack(autoenc_5, classifier_layer_5);  
2 view(stackednet_5);
```



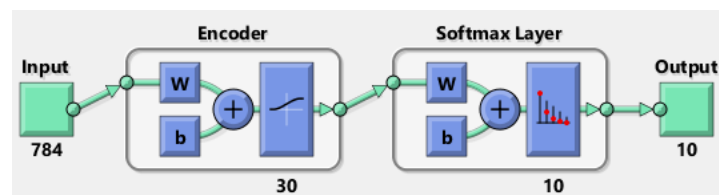
3.3.1 Classifier with 10 Features

```
1 stackednet_10 = stack(autoenc_10, classifier_layer_10);  
2 view(stackednet_10);
```



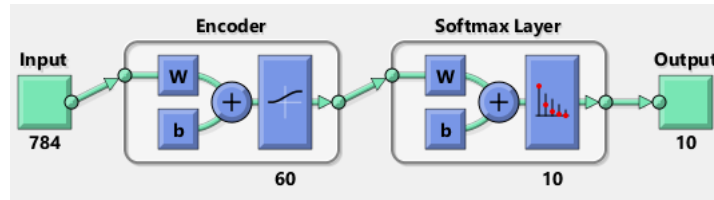
3.3.1 Classifier with 30 Features

```
1 stackednet_30 = stack(autoenc_30, classifier_layer_30);  
2 view(stackednet_30);
```



3.3.1 Classifier with 60 Features

```
1 stackednet_60 = stack(autoenc_60, classifier_layer_60);  
2 view(stackednet_60);
```



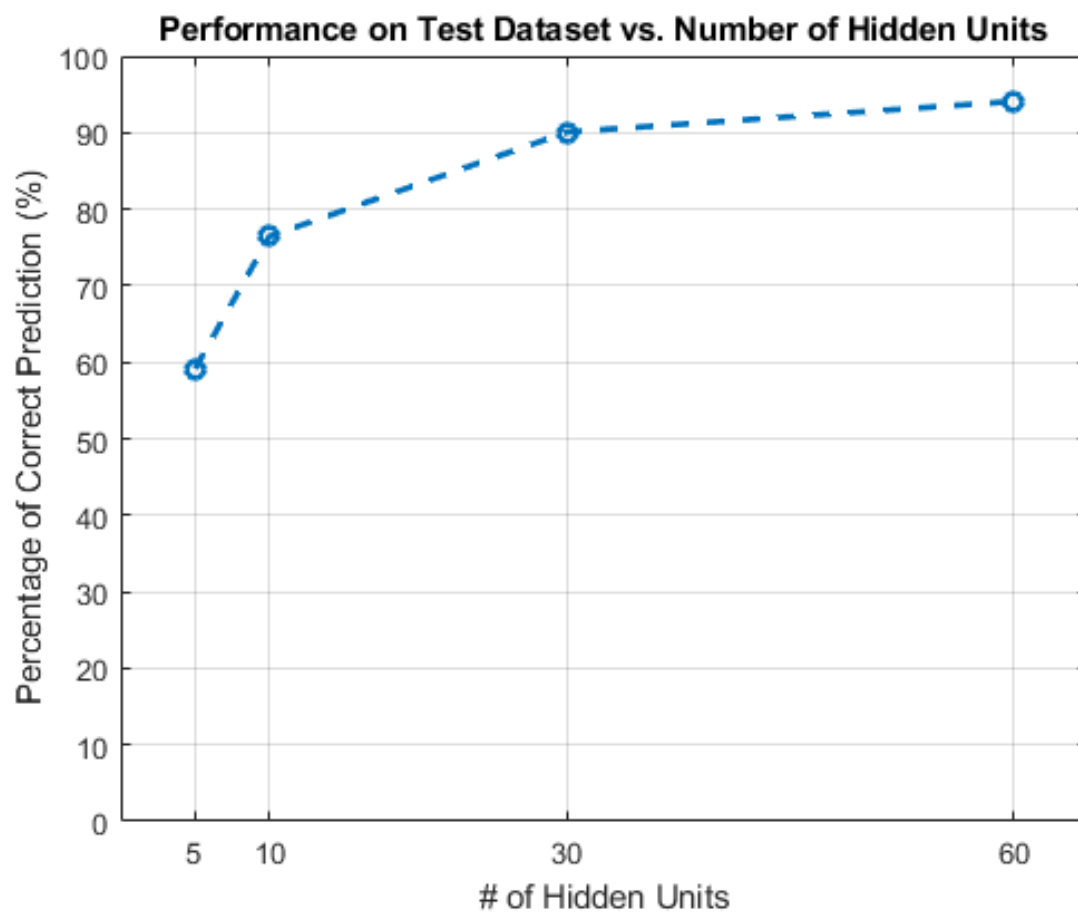
3.4 Classification Performance Versus Number of Hidden Units

Here we have investigated the performance of the classifiers (constructed above) on the whole test set of the MNIST dataset.

```

1 predicted_labels_5 = stackednet_5(train_images_resaped);
2 predicted_labels_10 = stackednet_10(train_images_resaped);
3 predicted_labels_30 = stackednet_30(train_images_resaped);
4 predicted_labels_60 = stackednet_60(train_images_resaped);
5
6 performance_5 = 1 - confusion(test_labels_one_hot_encoded, predicted_labels_5);
7 performance_10 = 1 - confusion(test_labels_one_hot_encoded, predicted_labels_10)
8     ;
9 performance_30 = 1 - confusion(test_labels_one_hot_encoded, predicted_labels_30)
10    ;
11 performance_60 = 1 - confusion(test_labels_one_hot_encoded, predicted_labels_60)
12    ;
13
14 performance = [performance_5 performance_10 performance_30 performance_60] *
15     100;
16 num_of_hidden_layers = [5 10 30 60];
17 plot(num_of_hidden_layers, performance, 'Marker', 'o', 'LineStyle', '--', '
18     LineWidth', 2);
19 xlim([0 65]);
20 ylim([0 100]);
21 grid on;
22 xticks(num_of_hidden_layers);
23 ylabel('Percentage of Correct Prediction (%)');
24 xlabel('# of Hidden Units');
25 title('Performance on Test Dataset vs. Number of Hidden Units');

```

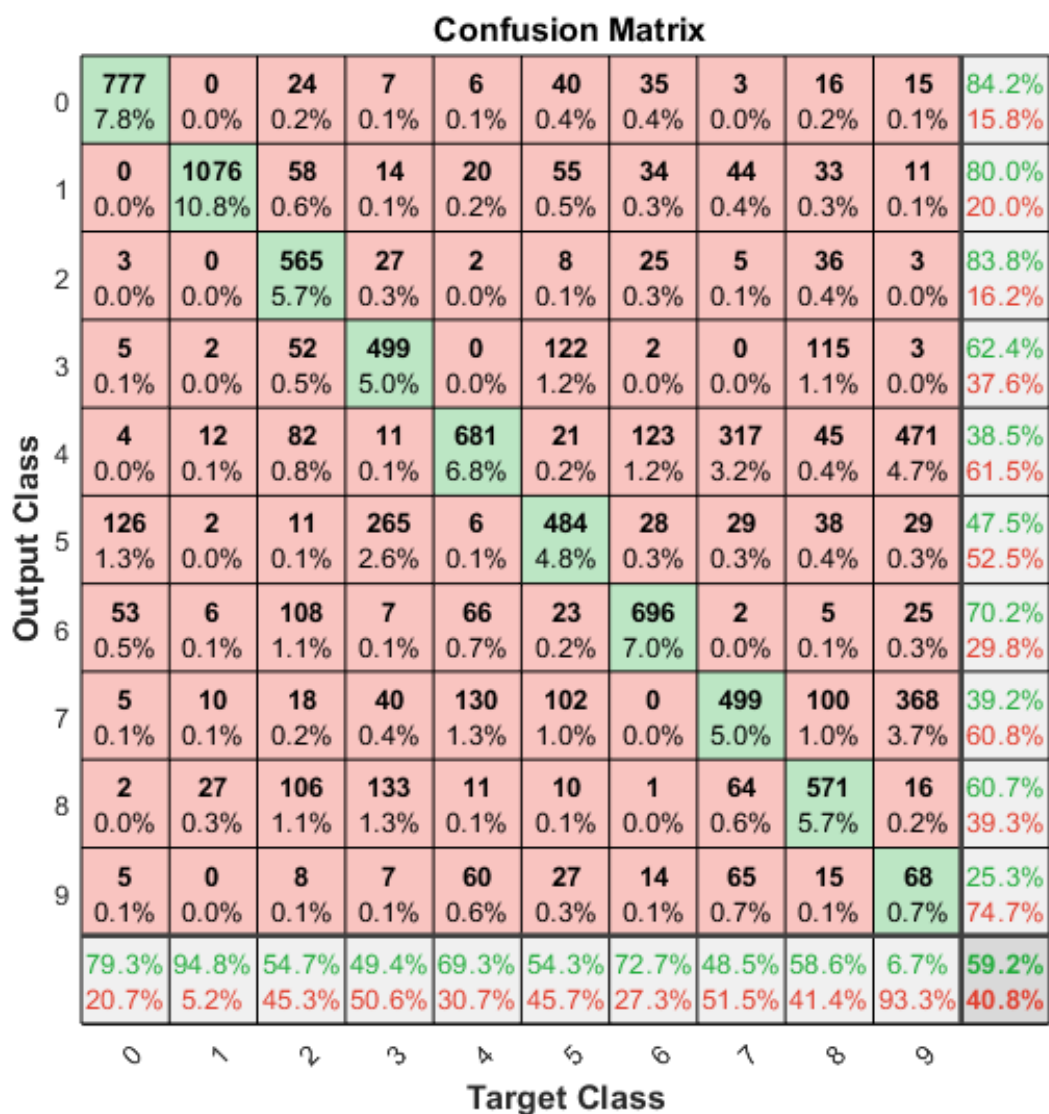


4. Confusion Matrices

Confusion matrices for every classifier of the previous part is plotted in this section.

4.1 Confusion Matrix for 5 Unit Classifier

```
1 plotconfusion(iCategorical(test_labels_one_hot_encoded), iCategorical(
    predicted_labels_5));
```



4.2 Confusion Matrix for 10 Unit Classifier

```
1 plotconfusion(iCategorical(test_labels_one_hot_encoded), iCategorical(
    predicted_labels_10));
```

Confusion Matrix

0	856 8.6%	0 0.0%	34 0.3%	10 0.1%	3 0.0%	26 0.3%	30 0.3%	6 0.1%	8 0.1%	16 0.2%	86.6% 13.4%
1	0 0.0%	1082 10.8%	26 0.3%	12 0.1%	10 0.1%	11 0.1%	19 0.2%	44 0.4%	32 0.3%	6 0.1%	87.1% 12.9%
2	12 0.1%	14 0.1%	772 7.7%	27 0.3%	6 0.1%	11 0.1%	24 0.2%	16 0.2%	26 0.3%	4 0.0%	84.6% 15.4%
3	19 0.2%	6 0.1%	47 0.5%	732 7.3%	2 0.0%	85 0.9%	3 0.0%	0 0.0%	182 1.8%	16 0.2%	67.0% 33.0%
4	0 0.0%	0 0.0%	34 0.3%	0 0.0%	737 7.4%	16 0.2%	28 0.3%	8 0.1%	12 0.1%	122 1.2%	77.0% 23.0%
5	67 0.7%	15 0.1%	9 0.1%	106 1.1%	31 0.3%	612 6.1%	48 0.5%	13 0.1%	105 1.1%	46 0.5%	58.2% 41.8%
6	17 0.2%	5 0.1%	46 0.5%	13 0.1%	31 0.3%	43 0.4%	799 8.0%	0 0.0%	8 0.1%	5 0.1%	82.6% 17.4%
7	8 0.1%	2 0.0%	24 0.2%	21 0.2%	9 0.1%	8 0.1%	5 0.1%	859 8.6%	24 0.2%	115 1.1%	79.9% 20.1%
8	1 0.0%	11 0.1%	35 0.4%	77 0.8%	15 0.1%	32 0.3%	2 0.0%	18 0.2%	526 5.3%	9 0.1%	72.5% 27.5%
9	0 0.0%	0 0.0%	5 0.1%	12 0.1%	138 1.4%	48 0.5%	0 0.0%	64 0.6%	51 0.5%	670 6.7%	67.8% 32.2%
	87.3% 12.7%	95.3% 4.7%	74.8% 25.2%	72.5% 27.5%	75.1% 24.9%	68.6% 31.4%	83.4% 16.6%	83.6% 16.4%	54.0% 46.0%	66.4% 33.6%	76.4% 23.6%
	0	1	2	3	4	5	6	7	8	9	
	Target Class										

4.3 Confusion Matrix for 30 Unit Classifier

```
1 plotconfusion(iCategorical(test_labels_one_hot_encoded), iCategorical(
    predicted_labels_30));
```

Confusion Matrix

0	943 9.4%	0 0.0%	14 0.1%	1 0.0%	1 0.0%	9 0.1%	17 0.2%	0 0.0%	9 0.1%	7 0.1%	94.2% 5.8%
1	0 0.0%	1115 11.2%	11 0.1%	8 0.1%	1 0.0%	4 0.0%	4 0.0%	14 0.1%	8 0.1%	14 0.1%	94.6% 5.4%
2	3 0.0%	2 0.0%	924 9.2%	23 0.2%	3 0.0%	7 0.1%	14 0.1%	24 0.2%	12 0.1%	3 0.0%	91.0% 9.0%
3	1 0.0%	2 0.0%	15 0.1%	909 9.1%	2 0.0%	41 0.4%	0 0.0%	5 0.1%	47 0.5%	4 0.0%	88.6% 11.4%
4	0 0.0%	1 0.0%	11 0.1%	0 0.0%	878 8.8%	9 0.1%	21 0.2%	5 0.1%	3 0.0%	54 0.5%	89.4% 10.6%
5	17 0.2%	0 0.0%	2 0.0%	28 0.3%	3 0.0%	771 7.7%	24 0.2%	4 0.0%	27 0.3%	11 0.1%	86.9% 13.1%
6	12 0.1%	4 0.0%	7 0.1%	2 0.0%	17 0.2%	16 0.2%	874 8.7%	0 0.0%	8 0.1%	4 0.0%	92.6% 7.4%
7	1 0.0%	0 0.0%	13 0.1%	14 0.1%	6 0.1%	3 0.0%	1 0.0%	908 9.1%	7 0.1%	39 0.4%	91.5% 8.5%
8	1 0.0%	11 0.1%	29 0.3%	23 0.2%	2 0.0%	27 0.3%	3 0.0%	7 0.1%	827 8.3%	14 0.1%	87.6% 12.4%
9	2 0.0%	0 0.0%	6 0.1%	2 0.0%	69 0.7%	5 0.1%	0 0.0%	61 0.6%	26 0.3%	859 8.6%	83.4% 16.6%
	96.2% 3.8%	98.2% 1.8%	89.5% 10.5%	90.0% 10.0%	89.4% 10.6%	86.4% 13.6%	91.2% 8.8%	88.3% 11.7%	84.9% 15.1%	85.1% 14.9%	90.1% 9.9%
	0	1	2	3	4	5	6	7	8	9	
	Target Class										

4.4 Confusion Matrix for 60 Unit Classifier

```
1 plotconfusion(iCategorical(test_labels_one_hot_encoded), iCategorical(
    predicted_labels_60));
```

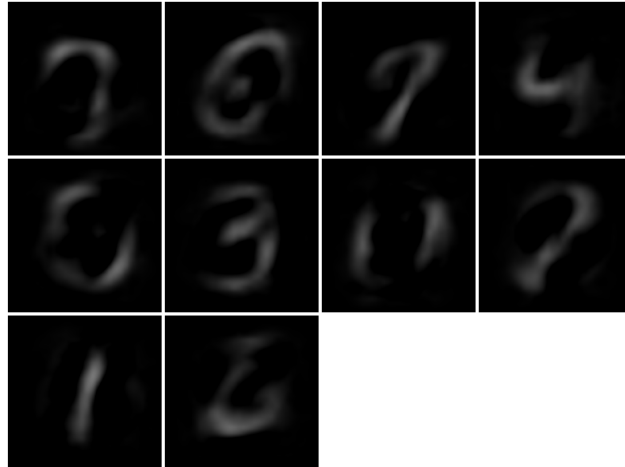
Confusion Matrix

0	965 9.7%	0 0.0%	7 0.1%	1 0.0%	2 0.0%	7 0.1%	15 0.1%	1 0.0%	5 0.1%	6 0.1%	95.6% 4.4%
1	0 0.0%	1121 11.2%	10 0.1%	5 0.1%	0 0.0%	0 0.0%	3 0.0%	8 0.1%	4 0.0%	5 0.1%	97.0% 3.0%
2	1 0.0%	1 0.0%	958 9.6%	13 0.1%	5 0.1%	5 0.1%	6 0.1%	17 0.2%	11 0.1%	1 0.0%	94.1% 5.9%
3	0 0.0%	3 0.0%	9 0.1%	946 9.5%	1 0.0%	17 0.2%	1 0.0%	7 0.1%	25 0.3%	10 0.1%	92.8% 7.2%
4	1 0.0%	0 0.0%	5 0.1%	0 0.0%	925 9.3%	1 0.0%	7 0.1%	6 0.1%	8 0.1%	25 0.3%	94.6% 5.4%
5	5 0.1%	1 0.0%	2 0.0%	17 0.2%	1 0.0%	836 8.4%	10 0.1%	1 0.0%	16 0.2%	8 0.1%	93.2% 6.8%
6	4 0.0%	3 0.0%	9 0.1%	0 0.0%	6 0.1%	5 0.1%	913 9.1%	0 0.0%	10 0.1%	0 0.0%	96.1% 3.9%
7	1 0.0%	0 0.0%	10 0.1%	13 0.1%	4 0.0%	1 0.0%	1 0.0%	955 9.6%	7 0.1%	24 0.2%	94.0% 6.0%
8	1 0.0%	6 0.1%	17 0.2%	10 0.1%	9 0.1%	14 0.1%	2 0.0%	3 0.0%	871 8.7%	12 0.1%	92.2% 7.8%
9	2 0.0%	0 0.0%	5 0.1%	5 0.1%	29 0.3%	6 0.1%	0 0.0%	30 0.3%	17 0.2%	918 9.2%	90.7% 9.3%
	98.5% 1.5%	98.8% 1.2%	92.8% 7.2%	93.7% 6.3%	94.2% 5.8%	93.7% 6.3%	95.3% 4.7%	92.9% 7.1%	89.4% 10.6%	91.0% 9.0%	94.1% 5.9%
	0	1	2	3	4	5	6	7	8	9	
	Target Class										

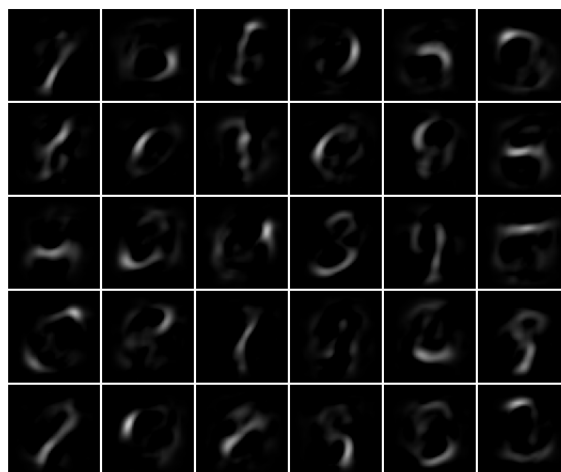
5. Encoder Weights Comparing to PCA Eigenvectors and K-means Centroids

First we repeat plotting the weights of autoencoders with 10 and 30 hidden units.

```
1 iMontage(autoenc_10.EncoderWeights');
```



```
1 iMontage(autoenc_30.EncoderWeights');
```



It can be seen from the figures above that:

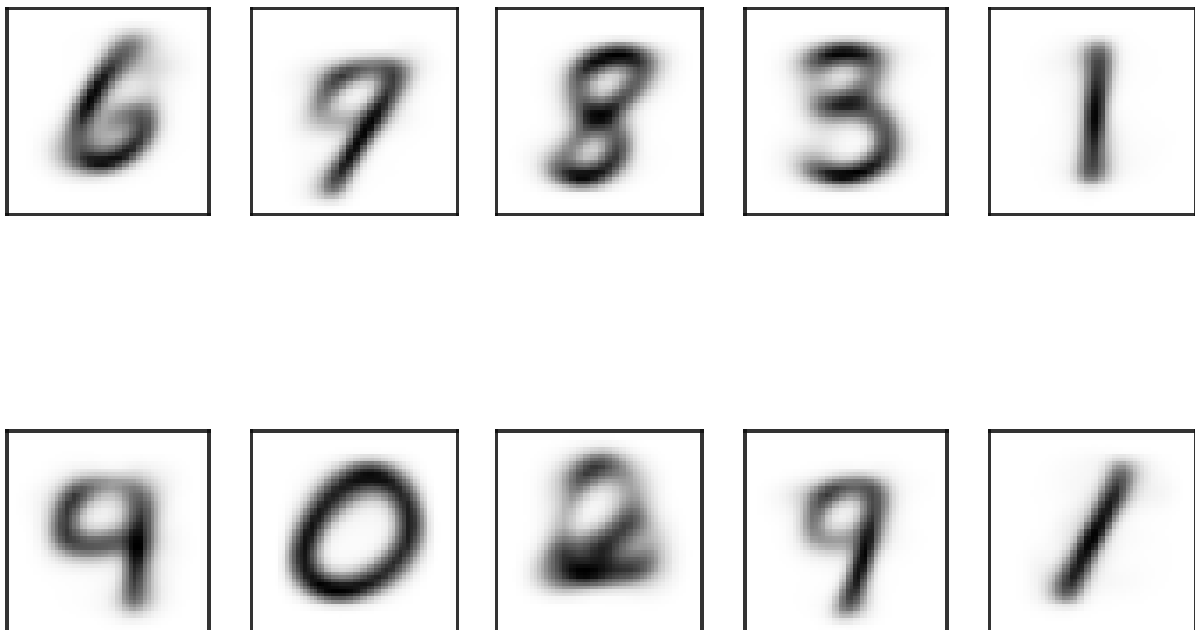
- Some weights of the autoencoder with 10 hidden units are actually a replica of a handwritten image, for example digits 0, 1, 3, 4, 6 and 7 could be seen in them.
- Autoencoder with 30 hidden units will be more precise in classifying images because every hidden node will be fired if a small pattern be found in the input image. The weights are not similar to a whole digit (as it was for autoencoder with 10 hidden units), but a portion of them are able to detect and reconstruct an image more precisely.

The first 10 PCA eigenvectors from HW#3 is plotted in figure below:



Here we can see a trace of digits 0, 1, 3 and 6 in the PCA eigenvectors.

10 K-means centroids from HW#4 are plotted below:



These centroids are much more like handwritten digits because they are constructed actually from averaging over the clusters of digits.

It is worth mentioning that PCA and K-means are linear dimensionality reduction algorithms however, autoencoders try to encode the images with neurons which have nonlinear activation functions.

Appendix

A.1 Saving Workspace Variables for Future Use

```
1 save('HW9_code_workspace.mat')
```

A.2 Definition of Auxiliary Functions

```
1 function downloadMNIST(mnist_train_image, mnist_train_label, mnist_test_image,
2   mnist_test_label)
3 if exist('train-images-idx3-ubyte','file') ~= 2
4     disp('Downloading MNIST dataset ... ');
5     websave([mnist_train_image, '.gz'], ...
6         ['http://yann.lecun.com/exdb/mnist/', ...
7         mnist_train_image, '.gz']);
8     websave([mnist_train_label, '.gz'], ...
9         ['http://yann.lecun.com/exdb/mnist/', ...
10        mnist_train_label, '.gz']);
11    websave([mnist_test_image, '.gz'], ...
12        ['http://yann.lecun.com/exdb/mnist/', ...
13        mnist_test_image, '.gz']);
14    websave([mnist_test_label, '.gz'], ...
15        ['http://yann.lecun.com/exdb/mnist/', ...
16        mnist_test_label, '.gz']);
17    disp('MNIST dataset downloaded. ');
18
19    disp('Unzipping started ... ');
20    gunzip([mnist_train_image, '.gz'])
21    gunzip([mnist_train_label, '.gz'])
22    gunzip([mnist_test_image, '.gz'])
23    gunzip([mnist_test_label, '.gz'])
24    delete([mnist_train_image, '.gz'])
25    delete([mnist_train_label, '.gz'])
26    delete([mnist_test_image, '.gz'])
27    delete([mnist_test_label, '.gz'])
28    disp('Unzipping completed. ');
29 else
30     disp('MNIST dataset already downloaded. ')
31 end
32
33 end
34
35 function [imgs, labels] = readMNIST(imgFile, labelFile, num_of_digits_to_read)
36
```

```

37 fileID = fopen(imgFile , 'r' , 'b');
38 header = fread(fileID , 1, 'int32');
39
40 if header ~= 2051
41     error('Invalid image file header');
42 end
43
44 count = fread(fileID , 1, 'int32');
45
46 if count < num_of_digits_to_read
47     error('Trying to read too many digits');
48 end
49
50 rows_num = fread(fileID , 1, 'int32');
51 cols_num = fread(fileID , 1, 'int32');
52
53 imgs = zeros([rows_num cols_num num_of_digits_to_read]);
54
55 for i = 1:num_of_digits_to_read
56     for row = 1:rows_num
57         imgs(row, :, i) = fread(fileID , cols_num, 'uint8');
58     end
59 end
60
61 fclose(fileID);
62
63 fileID = fopen(labelFile , 'r' , 'b');
64 header = fread(fileID , 1, 'int32');
65
66 if header ~= 2049
67     error('Invalid label file header');
68 end
69
70 count = fread(fileID , 1, 'int32');
71
72 if count < num_of_digits_to_read
73     error('Trying to read too many digits');
74 end
75
76 labels = fread(fileID , num_of_digits_to_read, 'uint8');
77
78 fclose(fileID);
79
80 imgs = double(imgs)./255.0;
81
82 end
83
84 function iMontage(images)

```

```

85  montage(reshape(images, [28 28 size(images, 2)]), 'BackgroundColor', 'white', '
      BorderSize', [2 2]);
86  end
87
88  function display_original_images_vs_reconstructed(original_images,
      reconstructed_images)
89  figure('Position', [100, 100, 1000, 500]);
90  subplot(1, 2, 1);
91  iMontage(original_images);
92  title('Original Images');
93  hold on;
94  subplot(1, 2, 2);
95  iMontage(reconstructed_images);
96  title('Reconstructed Images')
97  hold off
98  end
99
100 function categorized_label = iCategorical(on_hot_encoded_label)
101 [ind, ~]= vec2ind(on_hot_encoded_label);
102 categorized_label = categorical(ind', 1:10, {'0' '1' '2' '3' '4' '5' '6' '7' '8'
      '9'});
103 end

```