

# Assignment9

December 11, 2017

## 1 Autoencoders

In this assignment, we will look at autoencoders and compare their performance with an autoencoder-based PCA model. For this, we will continue to use the MNIST dataset that we have seen in the previous assignments.

As usual, the structure of the code is given to you, and you need to fill in the parts corresponding to the questions below.

**PART A:** In this part we will build three models: an autoencoder-based PCA, a shallow and a deeper autoencoder. We will then train and test these models on the MNIST dataset and compare their performance visually as well as by computing the test set reconstruction error. - Question 1 Autoencoder-based PCA: implement an autoencoder-based PCA model formed of two linear layers, using a structure similar to the shallow autoencoder model. - Question 2 Shallow autoencoder: Use different number of nodes (e.g.[128,16]), in hidden layers and compare the results. - Question 3 Deep autoencoder: after reading the basic autoencoder provided as an example, build a deeper autoencoder. Use different number of nodes (e.g.[128,64,16]), in hidden layers and compare the results.

**PART B:** In this part we will interpolate between two images in the latent space. This is done by first obtaining the encoded images:

$$latent\_1 = encoder(image\_1) \tag{1}$$

$$latent\_2 = encoder(image\_2) \tag{2}$$

and then generating the new interpolated image using the following equation:

$$interpolated\_image\_{\lambda} = decoder[\lambda * latent\_1 + (1 - \lambda) * latent\_2] \tag{3}$$

We will do this using the PCA model and then the autoencoder model and look at the differences between the two interpolations. Then, we need to get the image space representation and to verify that the interpolation generated using the deep autoencoder is better. - Question 1 Interpolate between an image corresponding to digit 1 and an image corresponding to digit 7 using the latent space representation obtained using the PCA-based autoencoder. - Question 2 Interpolate between an image corresponding to digit 1 and an image corresponding to digit 7 using the latent space representation obtained using the deep autoencoder.

```

In [1]: # Import libraries

import os
import torch
import torchvision
from torch import nn
from torch.autograd import Variable
from torch.utils.data import DataLoader
from torchvision import transforms
import torchvision.datasets as dset
from torchvision.utils import save_image

In [2]: # Import dataset
batch_size = 128

# Convert the data to Tensor and normalise by the mean and std of the data set
transform = transforms.Compose([transforms.ToTensor(),
                                transforms.Normalize((0.5,), (0.5,))])

# Import and normalize the train set in mini-batches
train_set = torchvision.datasets.MNIST(root='./data', train=True,
                                       download=True, transform=transform)
train_set_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size,
                                              shuffle=True)

# Import and normalize the test set in mini-batches
test_set = torchvision.datasets.MNIST(root='./data', train=False,
                                       download=True, transform=transform)
test_set_loader = torch.utils.data.DataLoader(test_set, batch_size=batch_size,
                                              shuffle=True)

```

**Visualise** Visualise examples of the images from the dataset.

```

In [3]: # Import libraries for visualising
import matplotlib.pyplot as plt
import numpy as np

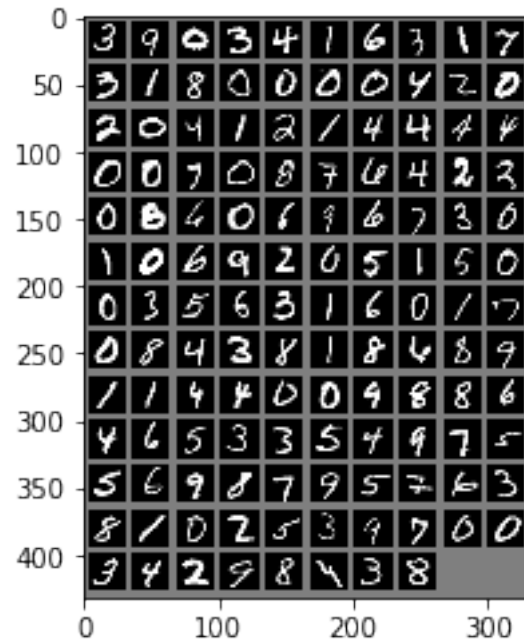
# Load a batch of training images for visualising
data_iterator = iter(train_set_loader)
images, labels = data_iterator.next()

# Create function for visualisation
def show_image(img):
    # revert the normalisation for displaying the images in their original form
    img = img * 0.5 + 0.5
    # Convert to numpy for visualisation
    npimg = img.numpy()
    # Plot each image

```

```
plt.imshow(np.transpose(npimg, (1, 2, 0)))
```

```
# Plot images as a grid using the 'make_grid' function
show_image(torchvision.utils.make_grid(images,10,5))
plt.show()
```



## 1.1 PART A:

### 1.2 Models

In this section we will implements three models as three different classes:

1. An autoencoder-based PCA
2. A shallow autoencoder
3. A deep autoencoder

**1. Autoencoder-based PCA model** Build an autoencoder-based PCA model. This will be made of two linear layers. The first linear layer is the encoder part. It receives as an input the image and has an output of shape `num_hidden_1`, where `num_hidden_1` the number of hidden units in layer `L1` . The second linear layer is the decoder and has an output of the same shape as the images in the dataset, since its role is to reconstruct the image.

```
In [10]: class autoencoder_PCA(nn.Module):
          def __init__(self,num_in,num_hidden_1,num_out):
              super(autoencoder_PCA, self).__init__()
```

```

##### Question 1 #####
# The layers of the encoder

# The layers of the decoder

##### End Question 1 #####

def forward(self, x):
    x = self.encoder(x)
    x = self.decoder(x)
    return x

```

**2. Shallow autoencoder model** This model is made of two parts:

- the encoder:** a layer that takes as an input the image and has an output of shape `num_hidden_1` and a **ReLU** activation function. A second layer that has an output of shape `num_hidden_2` and no activation function.
- the decoder:** this takes as an input the output of the encoder and has the same structure as a decoder, but in inverse order (the mirror image of the decoder). Additionally, the decoder applies a **tanh** activation function to the final layer.

This function is already given to you as an example of a basic autoencoder.

```

In [11]: class shallow_autoencoder(nn.Module):
    def __init__(self, num_in, num_hidden_1, num_hidden_2, num_out):
        super(shallow_autoencoder, self).__init__()
        # The layers of the encoder
        self.encoder = nn.Sequential(
            nn.Linear(num_in, num_hidden_1),
            nn.ReLU(True),
            nn.Linear(num_hidden_1, num_hidden_2))
        # The layers of the decoder
        self.decoder = nn.Sequential(
            nn.Linear(num_hidden_2, num_hidden_1),
            nn.ReLU(True),
            nn.Linear(num_hidden_1, num_out),
            nn.Tanh())

        # The forward pass
    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

```

**2. Deep autoencoder model** This model is similar to the previous one, but with a deeper structure:

- a. **the encoder:** a layer that takes as an input the image and has an output of shape `num_hidden_1` and a **ReLU** activation function. A second layer that has an output of shape `num_hidden_2` and a **ReLU** activation function. A final linear layer with output of shape `num_hidden_3` and no activation function
- b. **the decoder:** this takes as an input the output of the encoder and has the same structure as a decoder, but in inverse order (the mirror image of the decoder). Additionally, the decoder applies a **tanh** activation function to the final layer.

```
In [12]: class deep_autoencoder(nn.Module):
    def __init__(self, num_in, num_hidden_1, num_hidden_2, num_hidden_3, num_out):
        super(deep_autoencoder, self).__init__()
        ##### Question 3 #####
        # The layers of the encoder

        # The layers of the dencoder

        ##### End Question 3 #####

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

### 1.3 Train and test the models

**Train function** Now that we have build the models, let's define a function which performs the training. this function has the following parameters: the train set, the batch size, the number of epochs for which to perform the trainig, the learning rate, and the model to train.

```
In [13]: def train(train_set, batch_size, num_epochs, learning_rate, model):

    no_batches = int(np.round((len(train_set) / batch_size)))
    criterion = nn.MSELoss()
    optimizer = torch.optim.Adam(
        model.parameters(), lr=learning_rate, weight_decay=1e-5)

    losses = []
    for epoch in range(num_epochs):
        train_loss = 0
        for data in train_set_loader:
            img, _ = data
            img = img.view(img.size(0), -1)
            img = Variable(img)

            # ===== forward pass =====
            output = model(img)
            loss = criterion(output, img)
```

```

# ===== backward pass =====
optimizer.zero_grad()
loss.backward()
optimizer.step()
train_loss += loss.data[0]

# ===== log =====
print('====> Epoch: {} Average loss: {:.4f}'.format(
    epoch, train_loss / no_batches))
losses.append(train_loss / no_batches)
return losses

```

**Test function** Similarly to the train function, build a test function that evaluates the performance of the models on the test set

```

In [14]: def evaluate_test(test_set, test_set_loader, batch_size, model):

    test_loss = 0
    criterion = nn.MSELoss()
    test_no_batches = np.round(int(len(test_set)/batch_size))

    for data in test_set_loader:
        img, _ = data
        img = img.view(img.size(0), -1)
        img = Variable(img)

        # ===== forward pass =====
        output = model(img)
        loss = criterion(output, img)
        test_loss += loss.data[0]

    # ===== log =====
    test_loss = test_loss/test_no_batches
    print('====> Test loss: {:.4f}'.format(test_loss))
    return test_loss

```

Now that we have written the train and test functions, let's train and then compare the reconstruction error on the train and test set of the three models. For each model, plot the training error.

### 1.3.1 1. PCA based autoencoder model

```

In [18]: #define the model
num_input=28*28
num_hidden_1=16
num_output=28*28
model1 = autoencoder_PCA(num_input,num_hidden_1,num_output)

```

```

#train the model
model1_loss_train = train(train_set= train_set, batch_size = 128, num_epochs = 100, lea

# Plot the training error
fig = plt.figure()
plt.plot(model1_loss_train)
plt.xlabel('Epochs')
plt.ylabel('Reconstruction error')
plt.show()

# Calculate the loss in test set
model1_loss_test= evaluate_test(test_set = test_set, test_set_loader= test_set_loader,

====> Epoch: 0 Average loss: 0.2530
====> Epoch: 1 Average loss: 0.1292
====> Epoch: 2 Average loss: 0.1066
====> Epoch: 3 Average loss: 0.0967
====> Epoch: 4 Average loss: 0.0925
====> Epoch: 5 Average loss: 0.0904
====> Epoch: 6 Average loss: 0.0893
====> Epoch: 7 Average loss: 0.0884
====> Epoch: 8 Average loss: 0.0877
====> Epoch: 9 Average loss: 0.0871
====> Epoch: 10 Average loss: 0.0867
====> Epoch: 11 Average loss: 0.0864
====> Epoch: 12 Average loss: 0.0861
====> Epoch: 13 Average loss: 0.0859
====> Epoch: 14 Average loss: 0.0856
====> Epoch: 15 Average loss: 0.0855
====> Epoch: 16 Average loss: 0.0853
====> Epoch: 17 Average loss: 0.0852
====> Epoch: 18 Average loss: 0.0850
====> Epoch: 19 Average loss: 0.0849
====> Epoch: 20 Average loss: 0.0848
====> Epoch: 21 Average loss: 0.0847
====> Epoch: 22 Average loss: 0.0847
====> Epoch: 23 Average loss: 0.0846
====> Epoch: 24 Average loss: 0.0845
====> Epoch: 25 Average loss: 0.0844
====> Epoch: 26 Average loss: 0.0844
====> Epoch: 27 Average loss: 0.0844
====> Epoch: 28 Average loss: 0.0843
====> Epoch: 29 Average loss: 0.0843
====> Epoch: 30 Average loss: 0.0843
====> Epoch: 31 Average loss: 0.0843
====> Epoch: 32 Average loss: 0.0843
====> Epoch: 33 Average loss: 0.0842
====> Epoch: 34 Average loss: 0.0842

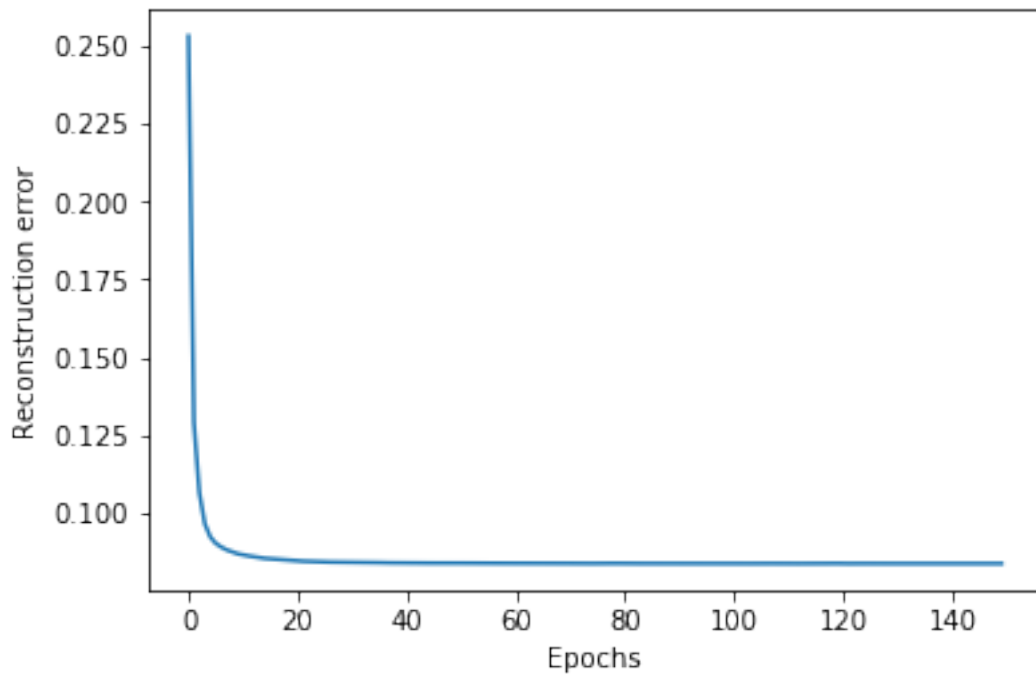
```

====> Epoch: 35 Average loss: 0.0843  
====> Epoch: 36 Average loss: 0.0842  
====> Epoch: 37 Average loss: 0.0842  
====> Epoch: 38 Average loss: 0.0842  
====> Epoch: 39 Average loss: 0.0842  
====> Epoch: 40 Average loss: 0.0841  
====> Epoch: 41 Average loss: 0.0841  
====> Epoch: 42 Average loss: 0.0841  
====> Epoch: 43 Average loss: 0.0841  
====> Epoch: 44 Average loss: 0.0841  
====> Epoch: 45 Average loss: 0.0841  
====> Epoch: 46 Average loss: 0.0841  
====> Epoch: 47 Average loss: 0.0841  
====> Epoch: 48 Average loss: 0.0841  
====> Epoch: 49 Average loss: 0.0841  
====> Epoch: 50 Average loss: 0.0841  
====> Epoch: 51 Average loss: 0.0841  
====> Epoch: 52 Average loss: 0.0841  
====> Epoch: 53 Average loss: 0.0841  
====> Epoch: 54 Average loss: 0.0840  
====> Epoch: 55 Average loss: 0.0841  
====> Epoch: 56 Average loss: 0.0841  
====> Epoch: 57 Average loss: 0.0841  
====> Epoch: 58 Average loss: 0.0841  
====> Epoch: 59 Average loss: 0.0841  
====> Epoch: 60 Average loss: 0.0840  
====> Epoch: 61 Average loss: 0.0841  
====> Epoch: 62 Average loss: 0.0841  
====> Epoch: 63 Average loss: 0.0840  
====> Epoch: 64 Average loss: 0.0840  
====> Epoch: 65 Average loss: 0.0841  
====> Epoch: 66 Average loss: 0.0840  
====> Epoch: 67 Average loss: 0.0840  
====> Epoch: 68 Average loss: 0.0840  
====> Epoch: 69 Average loss: 0.0840  
====> Epoch: 70 Average loss: 0.0840  
====> Epoch: 71 Average loss: 0.0840  
====> Epoch: 72 Average loss: 0.0840  
====> Epoch: 73 Average loss: 0.0840  
====> Epoch: 74 Average loss: 0.0840  
====> Epoch: 75 Average loss: 0.0840  
====> Epoch: 76 Average loss: 0.0840  
====> Epoch: 77 Average loss: 0.0840  
====> Epoch: 78 Average loss: 0.0840  
====> Epoch: 79 Average loss: 0.0840  
====> Epoch: 80 Average loss: 0.0840  
====> Epoch: 81 Average loss: 0.0840  
====> Epoch: 82 Average loss: 0.0840



====> Epoch: 83 Average loss: 0.0840  
====> Epoch: 84 Average loss: 0.0840  
====> Epoch: 85 Average loss: 0.0840  
====> Epoch: 86 Average loss: 0.0840  
====> Epoch: 87 Average loss: 0.0840  
====> Epoch: 88 Average loss: 0.0840  
====> Epoch: 89 Average loss: 0.0840  
====> Epoch: 90 Average loss: 0.0840  
====> Epoch: 91 Average loss: 0.0840  
====> Epoch: 92 Average loss: 0.0840  
====> Epoch: 93 Average loss: 0.0840  
====> Epoch: 94 Average loss: 0.0840  
====> Epoch: 95 Average loss: 0.0840  
====> Epoch: 96 Average loss: 0.0840  
====> Epoch: 97 Average loss: 0.0840  
====> Epoch: 98 Average loss: 0.0840  
====> Epoch: 99 Average loss: 0.0840  
====> Epoch: 100 Average loss: 0.0840  
====> Epoch: 101 Average loss: 0.0840  
====> Epoch: 102 Average loss: 0.0840  
====> Epoch: 103 Average loss: 0.0840  
====> Epoch: 104 Average loss: 0.0839  
====> Epoch: 105 Average loss: 0.0840  
====> Epoch: 106 Average loss: 0.0840  
====> Epoch: 107 Average loss: 0.0840  
====> Epoch: 108 Average loss: 0.0840  
====> Epoch: 109 Average loss: 0.0840  
====> Epoch: 110 Average loss: 0.0840  
====> Epoch: 111 Average loss: 0.0840  
====> Epoch: 112 Average loss: 0.0840  
====> Epoch: 113 Average loss: 0.0840  
====> Epoch: 114 Average loss: 0.0840  
====> Epoch: 115 Average loss: 0.0840  
====> Epoch: 116 Average loss: 0.0840  
====> Epoch: 117 Average loss: 0.0840  
====> Epoch: 118 Average loss: 0.0840  
====> Epoch: 119 Average loss: 0.0840  
====> Epoch: 120 Average loss: 0.0840  
====> Epoch: 121 Average loss: 0.0840  
====> Epoch: 122 Average loss: 0.0840  
====> Epoch: 123 Average loss: 0.0839  
====> Epoch: 124 Average loss: 0.0840  
====> Epoch: 125 Average loss: 0.0840  
====> Epoch: 126 Average loss: 0.0840  
====> Epoch: 127 Average loss: 0.0840  
====> Epoch: 128 Average loss: 0.0840  
====> Epoch: 129 Average loss: 0.0840  
====> Epoch: 130 Average loss: 0.0840

```
====> Epoch: 131 Average loss: 0.0839
====> Epoch: 132 Average loss: 0.0840
====> Epoch: 133 Average loss: 0.0839
====> Epoch: 134 Average loss: 0.0840
====> Epoch: 135 Average loss: 0.0840
====> Epoch: 136 Average loss: 0.0840
====> Epoch: 137 Average loss: 0.0840
====> Epoch: 138 Average loss: 0.0840
====> Epoch: 139 Average loss: 0.0839
====> Epoch: 140 Average loss: 0.0840
====> Epoch: 141 Average loss: 0.0839
====> Epoch: 142 Average loss: 0.0839
====> Epoch: 143 Average loss: 0.0839
====> Epoch: 144 Average loss: 0.0840
====> Epoch: 145 Average loss: 0.0840
====> Epoch: 146 Average loss: 0.0840
====> Epoch: 147 Average loss: 0.0840
====> Epoch: 148 Average loss: 0.0839
====> Epoch: 149 Average loss: 0.0840
```



```
====> Test loss: 0.0831
```

### 1.3.2 2. Shallow autoencoder

```
In [37]: #define the model
num_input=28*28
num_hidden_1=32
num_hidden_2=16
num_output=28*28

model2 = shallow_autoencoder(num_input,num_hidden_1,num_hidden_2,num_output)

#train the model
model2_loss_train = train(train_set= train_set, batch_size = 128, num_epochs = 100, lea

# Plot the error
fig = plt.figure()
plt.plot(model2_loss_train)
plt.xlabel('Epochs')
plt.ylabel('Reconstruction error')
plt.show()

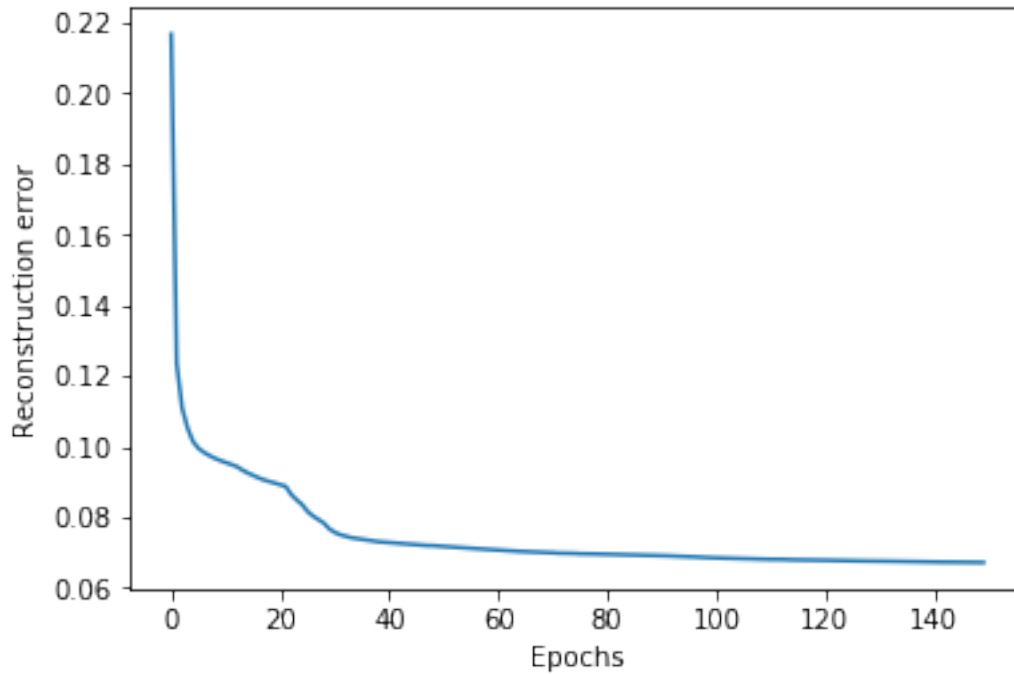
#calculate the loss in test set
model2_loss_test= evaluate_test(test_set = test_set, test_set_loader= test_set_loader,

====> Epoch: 0 Average loss: 0.2167
====> Epoch: 1 Average loss: 0.1236
====> Epoch: 2 Average loss: 0.1109
====> Epoch: 3 Average loss: 0.1053
====> Epoch: 4 Average loss: 0.1014
====> Epoch: 5 Average loss: 0.0995
====> Epoch: 6 Average loss: 0.0984
====> Epoch: 7 Average loss: 0.0975
====> Epoch: 8 Average loss: 0.0967
====> Epoch: 9 Average loss: 0.0961
====> Epoch: 10 Average loss: 0.0955
====> Epoch: 11 Average loss: 0.0950
====> Epoch: 12 Average loss: 0.0944
====> Epoch: 13 Average loss: 0.0934
====> Epoch: 14 Average loss: 0.0926
====> Epoch: 15 Average loss: 0.0919
====> Epoch: 16 Average loss: 0.0912
====> Epoch: 17 Average loss: 0.0906
====> Epoch: 18 Average loss: 0.0901
====> Epoch: 19 Average loss: 0.0897
====> Epoch: 20 Average loss: 0.0893
====> Epoch: 21 Average loss: 0.0888
====> Epoch: 22 Average loss: 0.0865
====> Epoch: 23 Average loss: 0.0849
====> Epoch: 24 Average loss: 0.0836
====> Epoch: 25 Average loss: 0.0817
```

====> Epoch: 26 Average loss: 0.0804  
====> Epoch: 27 Average loss: 0.0794  
====> Epoch: 28 Average loss: 0.0784  
====> Epoch: 29 Average loss: 0.0767  
====> Epoch: 30 Average loss: 0.0757  
====> Epoch: 31 Average loss: 0.0751  
====> Epoch: 32 Average loss: 0.0746  
====> Epoch: 33 Average loss: 0.0742  
====> Epoch: 34 Average loss: 0.0740  
====> Epoch: 35 Average loss: 0.0737  
====> Epoch: 36 Average loss: 0.0735  
====> Epoch: 37 Average loss: 0.0733  
====> Epoch: 38 Average loss: 0.0731  
====> Epoch: 39 Average loss: 0.0730  
====> Epoch: 40 Average loss: 0.0728  
====> Epoch: 41 Average loss: 0.0727  
====> Epoch: 42 Average loss: 0.0726  
====> Epoch: 43 Average loss: 0.0725  
====> Epoch: 44 Average loss: 0.0723  
====> Epoch: 45 Average loss: 0.0722  
====> Epoch: 46 Average loss: 0.0721  
====> Epoch: 47 Average loss: 0.0720  
====> Epoch: 48 Average loss: 0.0720  
====> Epoch: 49 Average loss: 0.0718  
====> Epoch: 50 Average loss: 0.0718  
====> Epoch: 51 Average loss: 0.0717  
====> Epoch: 52 Average loss: 0.0716  
====> Epoch: 53 Average loss: 0.0714  
====> Epoch: 54 Average loss: 0.0713  
====> Epoch: 55 Average loss: 0.0712  
====> Epoch: 56 Average loss: 0.0711  
====> Epoch: 57 Average loss: 0.0710  
====> Epoch: 58 Average loss: 0.0709  
====> Epoch: 59 Average loss: 0.0708  
====> Epoch: 60 Average loss: 0.0707  
====> Epoch: 61 Average loss: 0.0706  
====> Epoch: 62 Average loss: 0.0705  
====> Epoch: 63 Average loss: 0.0704  
====> Epoch: 64 Average loss: 0.0704  
====> Epoch: 65 Average loss: 0.0703  
====> Epoch: 66 Average loss: 0.0702  
====> Epoch: 67 Average loss: 0.0702  
====> Epoch: 68 Average loss: 0.0701  
====> Epoch: 69 Average loss: 0.0701  
====> Epoch: 70 Average loss: 0.0700  
====> Epoch: 71 Average loss: 0.0699  
====> Epoch: 72 Average loss: 0.0699  
====> Epoch: 73 Average loss: 0.0698

====> Epoch: 74 Average loss: 0.0698  
====> Epoch: 75 Average loss: 0.0697  
====> Epoch: 76 Average loss: 0.0697  
====> Epoch: 77 Average loss: 0.0696  
====> Epoch: 78 Average loss: 0.0696  
====> Epoch: 79 Average loss: 0.0696  
====> Epoch: 80 Average loss: 0.0695  
====> Epoch: 81 Average loss: 0.0695  
====> Epoch: 82 Average loss: 0.0695  
====> Epoch: 83 Average loss: 0.0694  
====> Epoch: 84 Average loss: 0.0694  
====> Epoch: 85 Average loss: 0.0693  
====> Epoch: 86 Average loss: 0.0693  
====> Epoch: 87 Average loss: 0.0692  
====> Epoch: 88 Average loss: 0.0692  
====> Epoch: 89 Average loss: 0.0692  
====> Epoch: 90 Average loss: 0.0691  
====> Epoch: 91 Average loss: 0.0691  
====> Epoch: 92 Average loss: 0.0690  
====> Epoch: 93 Average loss: 0.0690  
====> Epoch: 94 Average loss: 0.0689  
====> Epoch: 95 Average loss: 0.0689  
====> Epoch: 96 Average loss: 0.0688  
====> Epoch: 97 Average loss: 0.0687  
====> Epoch: 98 Average loss: 0.0686  
====> Epoch: 99 Average loss: 0.0686  
====> Epoch: 100 Average loss: 0.0685  
====> Epoch: 101 Average loss: 0.0684  
====> Epoch: 102 Average loss: 0.0684  
====> Epoch: 103 Average loss: 0.0684  
====> Epoch: 104 Average loss: 0.0684  
====> Epoch: 105 Average loss: 0.0683  
====> Epoch: 106 Average loss: 0.0683  
====> Epoch: 107 Average loss: 0.0682  
====> Epoch: 108 Average loss: 0.0682  
====> Epoch: 109 Average loss: 0.0682  
====> Epoch: 110 Average loss: 0.0681  
====> Epoch: 111 Average loss: 0.0681  
====> Epoch: 112 Average loss: 0.0681  
====> Epoch: 113 Average loss: 0.0680  
====> Epoch: 114 Average loss: 0.0680  
====> Epoch: 115 Average loss: 0.0680  
====> Epoch: 116 Average loss: 0.0679  
====> Epoch: 117 Average loss: 0.0679  
====> Epoch: 118 Average loss: 0.0679  
====> Epoch: 119 Average loss: 0.0679  
====> Epoch: 120 Average loss: 0.0678  
====> Epoch: 121 Average loss: 0.0678

====> Epoch: 122 Average loss: 0.0678  
====> Epoch: 123 Average loss: 0.0677  
====> Epoch: 124 Average loss: 0.0678  
====> Epoch: 125 Average loss: 0.0677  
====> Epoch: 126 Average loss: 0.0677  
====> Epoch: 127 Average loss: 0.0676  
====> Epoch: 128 Average loss: 0.0676  
====> Epoch: 129 Average loss: 0.0676  
====> Epoch: 130 Average loss: 0.0676  
====> Epoch: 131 Average loss: 0.0676  
====> Epoch: 132 Average loss: 0.0676  
====> Epoch: 133 Average loss: 0.0675  
====> Epoch: 134 Average loss: 0.0675  
====> Epoch: 135 Average loss: 0.0675  
====> Epoch: 136 Average loss: 0.0674  
====> Epoch: 137 Average loss: 0.0675  
====> Epoch: 138 Average loss: 0.0674  
====> Epoch: 139 Average loss: 0.0674  
====> Epoch: 140 Average loss: 0.0674  
====> Epoch: 141 Average loss: 0.0674  
====> Epoch: 142 Average loss: 0.0673  
====> Epoch: 143 Average loss: 0.0673  
====> Epoch: 144 Average loss: 0.0673  
====> Epoch: 145 Average loss: 0.0673  
====> Epoch: 146 Average loss: 0.0673  
====> Epoch: 147 Average loss: 0.0672  
====> Epoch: 148 Average loss: 0.0673  
====> Epoch: 149 Average loss: 0.0672



====> Test loss: 0.0666

```
In [38]: #define the model
          ##### Question 2 #####
          num_input=28*28
          num_hidden_1=
          num_hidden_2=
          num_output=28*28

          model2 = shallow_autoencoder(num_input,num_hidden_1,num_hidden_2,num_output)

          #train the model
          model2_loss_train = train(train_set= train_set, batch_size = 128, num_epochs = 100, lea

          # Plot the error
          fig = plt.figure()
          plt.plot(model2_loss_train)
          plt.xlabel('Epochs')
          plt.ylabel('Reconstruction error')
          plt.show()

          #calculate the loss in test set
          model2_loss_test= evaluate_test(test_set = test_set, test_set_loader= test_set_loader,
```

##### End of Question 2 #####

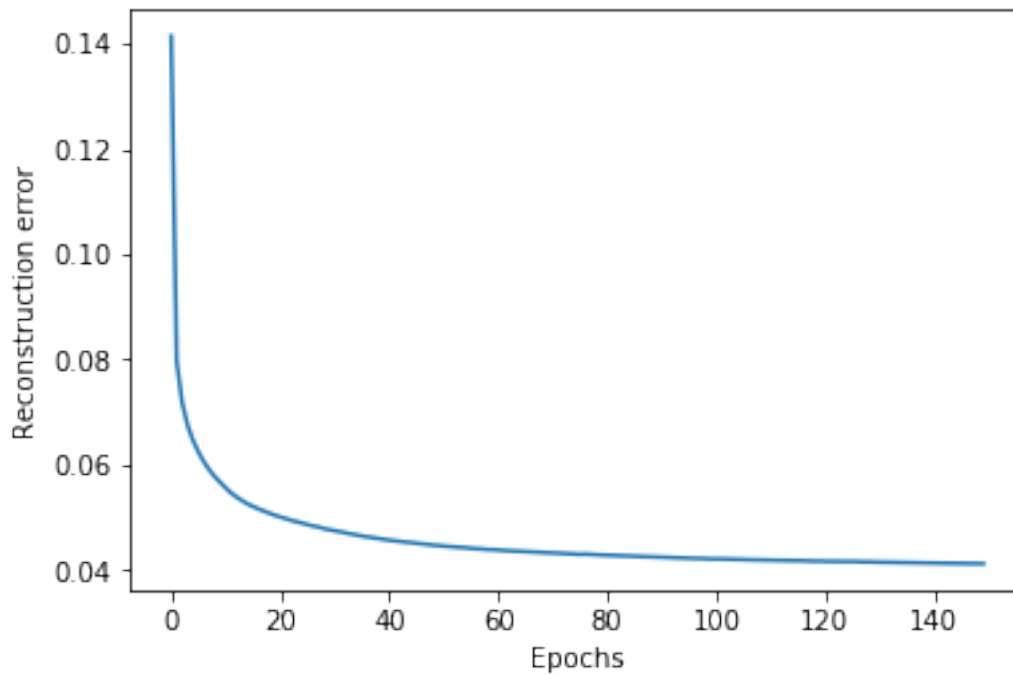
```
====> Epoch: 0 Average loss: 0.1415
====> Epoch: 1 Average loss: 0.0797
====> Epoch: 2 Average loss: 0.0719
====> Epoch: 3 Average loss: 0.0675
====> Epoch: 4 Average loss: 0.0647
====> Epoch: 5 Average loss: 0.0624
====> Epoch: 6 Average loss: 0.0606
====> Epoch: 7 Average loss: 0.0590
====> Epoch: 8 Average loss: 0.0577
====> Epoch: 9 Average loss: 0.0567
====> Epoch: 10 Average loss: 0.0556
====> Epoch: 11 Average loss: 0.0547
====> Epoch: 12 Average loss: 0.0539
====> Epoch: 13 Average loss: 0.0532
====> Epoch: 14 Average loss: 0.0526
====> Epoch: 15 Average loss: 0.0521
====> Epoch: 16 Average loss: 0.0516
====> Epoch: 17 Average loss: 0.0512
====> Epoch: 18 Average loss: 0.0508
====> Epoch: 19 Average loss: 0.0504
====> Epoch: 20 Average loss: 0.0501
====> Epoch: 21 Average loss: 0.0498
====> Epoch: 22 Average loss: 0.0495
====> Epoch: 23 Average loss: 0.0492
====> Epoch: 24 Average loss: 0.0490
====> Epoch: 25 Average loss: 0.0486
====> Epoch: 26 Average loss: 0.0484
====> Epoch: 27 Average loss: 0.0482
====> Epoch: 28 Average loss: 0.0479
====> Epoch: 29 Average loss: 0.0477
====> Epoch: 30 Average loss: 0.0474
====> Epoch: 31 Average loss: 0.0473
====> Epoch: 32 Average loss: 0.0471
====> Epoch: 33 Average loss: 0.0468
====> Epoch: 34 Average loss: 0.0466
====> Epoch: 35 Average loss: 0.0465
====> Epoch: 36 Average loss: 0.0463
====> Epoch: 37 Average loss: 0.0462
====> Epoch: 38 Average loss: 0.0460
====> Epoch: 39 Average loss: 0.0459
====> Epoch: 40 Average loss: 0.0457
====> Epoch: 41 Average loss: 0.0456
====> Epoch: 42 Average loss: 0.0455
====> Epoch: 43 Average loss: 0.0453
====> Epoch: 44 Average loss: 0.0452
```



====> Epoch: 45 Average loss: 0.0451  
====> Epoch: 46 Average loss: 0.0450  
====> Epoch: 47 Average loss: 0.0449  
====> Epoch: 48 Average loss: 0.0448  
====> Epoch: 49 Average loss: 0.0447  
====> Epoch: 50 Average loss: 0.0446  
====> Epoch: 51 Average loss: 0.0445  
====> Epoch: 52 Average loss: 0.0444  
====> Epoch: 53 Average loss: 0.0443  
====> Epoch: 54 Average loss: 0.0442  
====> Epoch: 55 Average loss: 0.0441  
====> Epoch: 56 Average loss: 0.0440  
====> Epoch: 57 Average loss: 0.0440  
====> Epoch: 58 Average loss: 0.0439  
====> Epoch: 59 Average loss: 0.0439  
====> Epoch: 60 Average loss: 0.0438  
====> Epoch: 61 Average loss: 0.0437  
====> Epoch: 62 Average loss: 0.0437  
====> Epoch: 63 Average loss: 0.0436  
====> Epoch: 64 Average loss: 0.0436  
====> Epoch: 65 Average loss: 0.0435  
====> Epoch: 66 Average loss: 0.0434  
====> Epoch: 67 Average loss: 0.0434  
====> Epoch: 68 Average loss: 0.0433  
====> Epoch: 69 Average loss: 0.0433  
====> Epoch: 70 Average loss: 0.0433  
====> Epoch: 71 Average loss: 0.0432  
====> Epoch: 72 Average loss: 0.0432  
====> Epoch: 73 Average loss: 0.0431  
====> Epoch: 74 Average loss: 0.0431  
====> Epoch: 75 Average loss: 0.0430  
====> Epoch: 76 Average loss: 0.0430  
====> Epoch: 77 Average loss: 0.0430  
====> Epoch: 78 Average loss: 0.0429  
====> Epoch: 79 Average loss: 0.0429  
====> Epoch: 80 Average loss: 0.0428  
====> Epoch: 81 Average loss: 0.0428  
====> Epoch: 82 Average loss: 0.0427  
====> Epoch: 83 Average loss: 0.0427  
====> Epoch: 84 Average loss: 0.0427  
====> Epoch: 85 Average loss: 0.0426  
====> Epoch: 86 Average loss: 0.0426  
====> Epoch: 87 Average loss: 0.0425  
====> Epoch: 88 Average loss: 0.0425  
====> Epoch: 89 Average loss: 0.0425  
====> Epoch: 90 Average loss: 0.0425  
====> Epoch: 91 Average loss: 0.0424  
====> Epoch: 92 Average loss: 0.0424

====> Epoch: 93 Average loss: 0.0423  
====> Epoch: 94 Average loss: 0.0423  
====> Epoch: 95 Average loss: 0.0423  
====> Epoch: 96 Average loss: 0.0422  
====> Epoch: 97 Average loss: 0.0422  
====> Epoch: 98 Average loss: 0.0422  
====> Epoch: 99 Average loss: 0.0421  
====> Epoch: 100 Average loss: 0.0421  
====> Epoch: 101 Average loss: 0.0421  
====> Epoch: 102 Average loss: 0.0421  
====> Epoch: 103 Average loss: 0.0421  
====> Epoch: 104 Average loss: 0.0420  
====> Epoch: 105 Average loss: 0.0420  
====> Epoch: 106 Average loss: 0.0420  
====> Epoch: 107 Average loss: 0.0420  
====> Epoch: 108 Average loss: 0.0419  
====> Epoch: 109 Average loss: 0.0419  
====> Epoch: 110 Average loss: 0.0419  
====> Epoch: 111 Average loss: 0.0418  
====> Epoch: 112 Average loss: 0.0418  
====> Epoch: 113 Average loss: 0.0418  
====> Epoch: 114 Average loss: 0.0418  
====> Epoch: 115 Average loss: 0.0418  
====> Epoch: 116 Average loss: 0.0418  
====> Epoch: 117 Average loss: 0.0417  
====> Epoch: 118 Average loss: 0.0417  
====> Epoch: 119 Average loss: 0.0417  
====> Epoch: 120 Average loss: 0.0417  
====> Epoch: 121 Average loss: 0.0417  
====> Epoch: 122 Average loss: 0.0417  
====> Epoch: 123 Average loss: 0.0416  
====> Epoch: 124 Average loss: 0.0416  
====> Epoch: 125 Average loss: 0.0417  
====> Epoch: 126 Average loss: 0.0416  
====> Epoch: 127 Average loss: 0.0416  
====> Epoch: 128 Average loss: 0.0416  
====> Epoch: 129 Average loss: 0.0415  
====> Epoch: 130 Average loss: 0.0415  
====> Epoch: 131 Average loss: 0.0415  
====> Epoch: 132 Average loss: 0.0415  
====> Epoch: 133 Average loss: 0.0415  
====> Epoch: 134 Average loss: 0.0414  
====> Epoch: 135 Average loss: 0.0414  
====> Epoch: 136 Average loss: 0.0414  
====> Epoch: 137 Average loss: 0.0414  
====> Epoch: 138 Average loss: 0.0414  
====> Epoch: 139 Average loss: 0.0414  
====> Epoch: 140 Average loss: 0.0414

```
====> Epoch: 141 Average loss: 0.0413
====> Epoch: 142 Average loss: 0.0413
====> Epoch: 143 Average loss: 0.0413
====> Epoch: 144 Average loss: 0.0412
====> Epoch: 145 Average loss: 0.0413
====> Epoch: 146 Average loss: 0.0412
====> Epoch: 147 Average loss: 0.0412
====> Epoch: 148 Average loss: 0.0412
====> Epoch: 149 Average loss: 0.0412
```



```
====> Test loss: 0.0415
```

### 1.3.3 3. Deep autoencoder

```
In [20]: #define the model
```

```
num_input=28*28
num_hidden_1=64
num_hidden_2=32
num_hidden_3=16
num_output=28*28
```

```
model3 = deep_autoencoder(num_input,num_hidden_1,num_hidden_2,num_hidden_3,num_output)
```

```

#train the model
model3_loss_train = train(train_set= train_set, batch_size = 128, num_epochs = 100, lea

# Plot the error
fig = plt.figure()
plt.plot(model3_loss_train)
plt.xlabel('Epochs')
plt.ylabel('Reconstruction error')
plt.show()

#calculate the loss in test set
model3_loss_test= evaluate_test(test_set = test_set, test_set_loader= test_set_loader,

====> Epoch: 0 Average loss: 0.2259
====> Epoch: 1 Average loss: 0.1283
====> Epoch: 2 Average loss: 0.1084
====> Epoch: 3 Average loss: 0.0989
====> Epoch: 4 Average loss: 0.0936
====> Epoch: 5 Average loss: 0.0907
====> Epoch: 6 Average loss: 0.0884
====> Epoch: 7 Average loss: 0.0856
====> Epoch: 8 Average loss: 0.0825
====> Epoch: 9 Average loss: 0.0795
====> Epoch: 10 Average loss: 0.0773
====> Epoch: 11 Average loss: 0.0757
====> Epoch: 12 Average loss: 0.0742
====> Epoch: 13 Average loss: 0.0729
====> Epoch: 14 Average loss: 0.0716
====> Epoch: 15 Average loss: 0.0705
====> Epoch: 16 Average loss: 0.0695
====> Epoch: 17 Average loss: 0.0686
====> Epoch: 18 Average loss: 0.0679
====> Epoch: 19 Average loss: 0.0672
====> Epoch: 20 Average loss: 0.0665
====> Epoch: 21 Average loss: 0.0658
====> Epoch: 22 Average loss: 0.0651
====> Epoch: 23 Average loss: 0.0645
====> Epoch: 24 Average loss: 0.0639
====> Epoch: 25 Average loss: 0.0633
====> Epoch: 26 Average loss: 0.0628
====> Epoch: 27 Average loss: 0.0622
====> Epoch: 28 Average loss: 0.0618
====> Epoch: 29 Average loss: 0.0614
====> Epoch: 30 Average loss: 0.0610
====> Epoch: 31 Average loss: 0.0606
====> Epoch: 32 Average loss: 0.0604
====> Epoch: 33 Average loss: 0.0600
====> Epoch: 34 Average loss: 0.0597

```

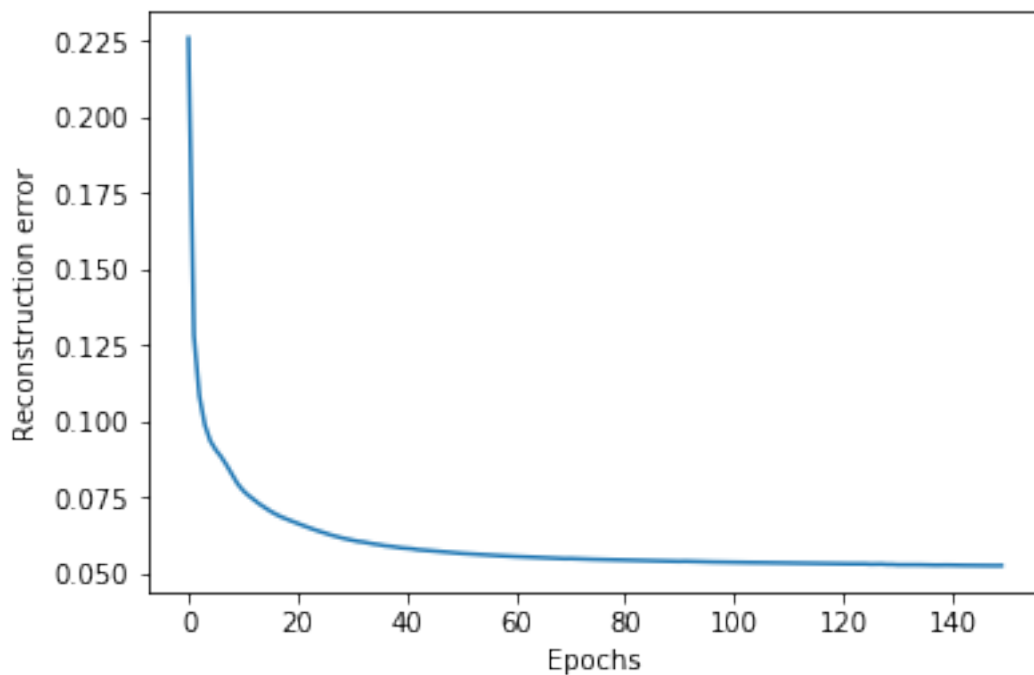
====> Epoch: 35 Average loss: 0.0595  
====> Epoch: 36 Average loss: 0.0592  
====> Epoch: 37 Average loss: 0.0590  
====> Epoch: 38 Average loss: 0.0587  
====> Epoch: 39 Average loss: 0.0585  
====> Epoch: 40 Average loss: 0.0583  
====> Epoch: 41 Average loss: 0.0582  
====> Epoch: 42 Average loss: 0.0579  
====> Epoch: 43 Average loss: 0.0578  
====> Epoch: 44 Average loss: 0.0576  
====> Epoch: 45 Average loss: 0.0574  
====> Epoch: 46 Average loss: 0.0572  
====> Epoch: 47 Average loss: 0.0571  
====> Epoch: 48 Average loss: 0.0569  
====> Epoch: 49 Average loss: 0.0569  
====> Epoch: 50 Average loss: 0.0567  
====> Epoch: 51 Average loss: 0.0566  
====> Epoch: 52 Average loss: 0.0564  
====> Epoch: 53 Average loss: 0.0563  
====> Epoch: 54 Average loss: 0.0562  
====> Epoch: 55 Average loss: 0.0561  
====> Epoch: 56 Average loss: 0.0560  
====> Epoch: 57 Average loss: 0.0559  
====> Epoch: 58 Average loss: 0.0558  
====> Epoch: 59 Average loss: 0.0557  
====> Epoch: 60 Average loss: 0.0556  
====> Epoch: 61 Average loss: 0.0556  
====> Epoch: 62 Average loss: 0.0555  
====> Epoch: 63 Average loss: 0.0554  
====> Epoch: 64 Average loss: 0.0553  
====> Epoch: 65 Average loss: 0.0552  
====> Epoch: 66 Average loss: 0.0552  
====> Epoch: 67 Average loss: 0.0551  
====> Epoch: 68 Average loss: 0.0550  
====> Epoch: 69 Average loss: 0.0549  
====> Epoch: 70 Average loss: 0.0550  
====> Epoch: 71 Average loss: 0.0549  
====> Epoch: 72 Average loss: 0.0548  
====> Epoch: 73 Average loss: 0.0547  
====> Epoch: 74 Average loss: 0.0547  
====> Epoch: 75 Average loss: 0.0547  
====> Epoch: 76 Average loss: 0.0546  
====> Epoch: 77 Average loss: 0.0546  
====> Epoch: 78 Average loss: 0.0545  
====> Epoch: 79 Average loss: 0.0545  
====> Epoch: 80 Average loss: 0.0544  
====> Epoch: 81 Average loss: 0.0545  
====> Epoch: 82 Average loss: 0.0543

====> Epoch: 83 Average loss: 0.0543  
====> Epoch: 84 Average loss: 0.0543  
====> Epoch: 85 Average loss: 0.0542  
====> Epoch: 86 Average loss: 0.0542  
====> Epoch: 87 Average loss: 0.0542  
====> Epoch: 88 Average loss: 0.0541  
====> Epoch: 89 Average loss: 0.0541  
====> Epoch: 90 Average loss: 0.0540  
====> Epoch: 91 Average loss: 0.0541  
====> Epoch: 92 Average loss: 0.0540  
====> Epoch: 93 Average loss: 0.0540  
====> Epoch: 94 Average loss: 0.0539  
====> Epoch: 95 Average loss: 0.0539  
====> Epoch: 96 Average loss: 0.0538  
====> Epoch: 97 Average loss: 0.0538  
====> Epoch: 98 Average loss: 0.0538  
====> Epoch: 99 Average loss: 0.0538  
====> Epoch: 100 Average loss: 0.0537  
====> Epoch: 101 Average loss: 0.0537  
====> Epoch: 102 Average loss: 0.0537  
====> Epoch: 103 Average loss: 0.0536  
====> Epoch: 104 Average loss: 0.0537  
====> Epoch: 105 Average loss: 0.0536  
====> Epoch: 106 Average loss: 0.0536  
====> Epoch: 107 Average loss: 0.0536  
====> Epoch: 108 Average loss: 0.0535  
====> Epoch: 109 Average loss: 0.0535  
====> Epoch: 110 Average loss: 0.0535  
====> Epoch: 111 Average loss: 0.0535  
====> Epoch: 112 Average loss: 0.0534  
====> Epoch: 113 Average loss: 0.0534  
====> Epoch: 114 Average loss: 0.0534  
====> Epoch: 115 Average loss: 0.0534  
====> Epoch: 116 Average loss: 0.0533  
====> Epoch: 117 Average loss: 0.0533  
====> Epoch: 118 Average loss: 0.0533  
====> Epoch: 119 Average loss: 0.0533  
====> Epoch: 120 Average loss: 0.0533  
====> Epoch: 121 Average loss: 0.0532  
====> Epoch: 122 Average loss: 0.0532  
====> Epoch: 123 Average loss: 0.0532  
====> Epoch: 124 Average loss: 0.0532  
====> Epoch: 125 Average loss: 0.0531  
====> Epoch: 126 Average loss: 0.0531  
====> Epoch: 127 Average loss: 0.0532  
====> Epoch: 128 Average loss: 0.0530  
====> Epoch: 129 Average loss: 0.0530  
====> Epoch: 130 Average loss: 0.0529

```

====> Epoch: 131 Average loss: 0.0529
====> Epoch: 132 Average loss: 0.0529
====> Epoch: 133 Average loss: 0.0529
====> Epoch: 134 Average loss: 0.0529
====> Epoch: 135 Average loss: 0.0528
====> Epoch: 136 Average loss: 0.0528
====> Epoch: 137 Average loss: 0.0528
====> Epoch: 138 Average loss: 0.0528
====> Epoch: 139 Average loss: 0.0528
====> Epoch: 140 Average loss: 0.0528
====> Epoch: 141 Average loss: 0.0528
====> Epoch: 142 Average loss: 0.0527
====> Epoch: 143 Average loss: 0.0527
====> Epoch: 144 Average loss: 0.0527
====> Epoch: 145 Average loss: 0.0527
====> Epoch: 146 Average loss: 0.0527
====> Epoch: 147 Average loss: 0.0526
====> Epoch: 148 Average loss: 0.0526
====> Epoch: 149 Average loss: 0.0527

```



```

====> Test loss: 0.0525

```

```

In [36]: ##### Question 3 #####
         #define the model

```

```

num_input=28*28
num_hidden_1=
num_hidden_2=
num_hidden_3=
num_output=28*28

model3 = deep_autoencoder(num_input,num_hidden_1,num_hidden_2,num_hidden_3,num_output)

#train the model
model3_loss_train = train(train_set= train_set, batch_size = 128, num_epochs = 100, learning_rate=0.001)

# Plot the error
fig = plt.figure()
plt.plot(model3_loss_train)
plt.xlabel('Epochs')
plt.ylabel('Reconstruction error')
plt.show()

#calculate the loss in test set
model3_loss_test= evaluate_test(test_set = test_set, test_set_loader= test_set_loader, test_loader=test_loader)
##### End of Question 3 #####

====> Epoch: 0 Average loss: 0.1892
====> Epoch: 1 Average loss: 0.1061
====> Epoch: 2 Average loss: 0.0900
====> Epoch: 3 Average loss: 0.0802
====> Epoch: 4 Average loss: 0.0747
====> Epoch: 5 Average loss: 0.0704
====> Epoch: 6 Average loss: 0.0672
====> Epoch: 7 Average loss: 0.0644
====> Epoch: 8 Average loss: 0.0620
====> Epoch: 9 Average loss: 0.0603
====> Epoch: 10 Average loss: 0.0589
====> Epoch: 11 Average loss: 0.0577
====> Epoch: 12 Average loss: 0.0568
====> Epoch: 13 Average loss: 0.0559
====> Epoch: 14 Average loss: 0.0551
====> Epoch: 15 Average loss: 0.0543
====> Epoch: 16 Average loss: 0.0538
====> Epoch: 17 Average loss: 0.0532
====> Epoch: 18 Average loss: 0.0526
====> Epoch: 19 Average loss: 0.0521
====> Epoch: 20 Average loss: 0.0516
====> Epoch: 21 Average loss: 0.0513
====> Epoch: 22 Average loss: 0.0509
====> Epoch: 23 Average loss: 0.0504
====> Epoch: 24 Average loss: 0.0501
====> Epoch: 25 Average loss: 0.0498

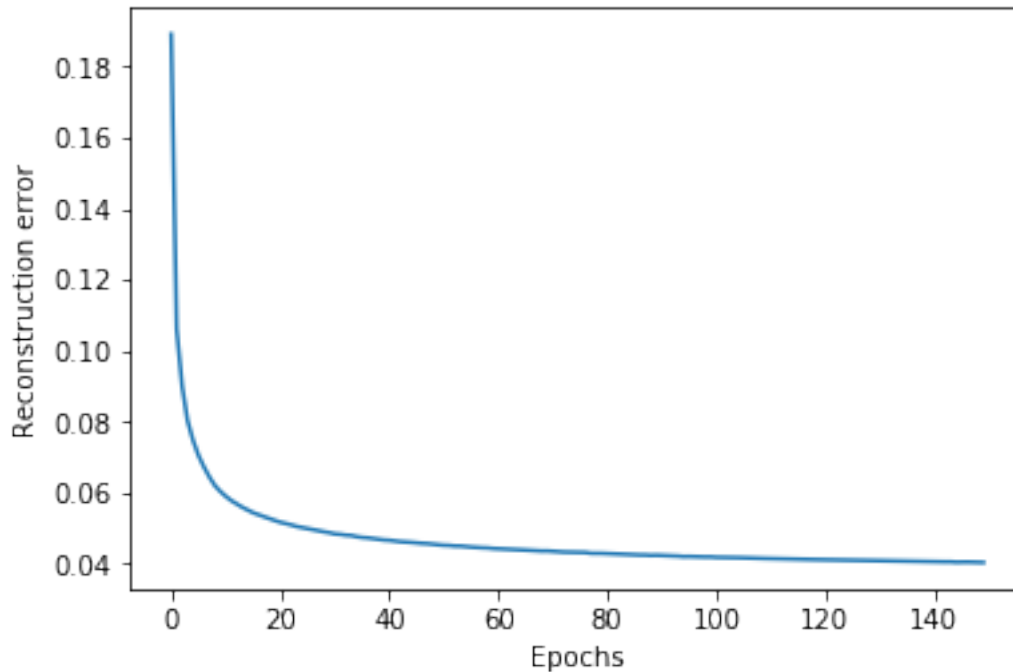
```



====> Epoch: 26 Average loss: 0.0495  
====> Epoch: 27 Average loss: 0.0492  
====> Epoch: 28 Average loss: 0.0489  
====> Epoch: 29 Average loss: 0.0487  
====> Epoch: 30 Average loss: 0.0483  
====> Epoch: 31 Average loss: 0.0482  
====> Epoch: 32 Average loss: 0.0480  
====> Epoch: 33 Average loss: 0.0479  
====> Epoch: 34 Average loss: 0.0475  
====> Epoch: 35 Average loss: 0.0473  
====> Epoch: 36 Average loss: 0.0472  
====> Epoch: 37 Average loss: 0.0470  
====> Epoch: 38 Average loss: 0.0469  
====> Epoch: 39 Average loss: 0.0466  
====> Epoch: 40 Average loss: 0.0465  
====> Epoch: 41 Average loss: 0.0463  
====> Epoch: 42 Average loss: 0.0461  
====> Epoch: 43 Average loss: 0.0460  
====> Epoch: 44 Average loss: 0.0459  
====> Epoch: 45 Average loss: 0.0458  
====> Epoch: 46 Average loss: 0.0456  
====> Epoch: 47 Average loss: 0.0456  
====> Epoch: 48 Average loss: 0.0454  
====> Epoch: 49 Average loss: 0.0453  
====> Epoch: 50 Average loss: 0.0451  
====> Epoch: 51 Average loss: 0.0451  
====> Epoch: 52 Average loss: 0.0449  
====> Epoch: 53 Average loss: 0.0449  
====> Epoch: 54 Average loss: 0.0447  
====> Epoch: 55 Average loss: 0.0446  
====> Epoch: 56 Average loss: 0.0445  
====> Epoch: 57 Average loss: 0.0444  
====> Epoch: 58 Average loss: 0.0444  
====> Epoch: 59 Average loss: 0.0442  
====> Epoch: 60 Average loss: 0.0441  
====> Epoch: 61 Average loss: 0.0440  
====> Epoch: 62 Average loss: 0.0440  
====> Epoch: 63 Average loss: 0.0439  
====> Epoch: 64 Average loss: 0.0438  
====> Epoch: 65 Average loss: 0.0438  
====> Epoch: 66 Average loss: 0.0437  
====> Epoch: 67 Average loss: 0.0436  
====> Epoch: 68 Average loss: 0.0435  
====> Epoch: 69 Average loss: 0.0435  
====> Epoch: 70 Average loss: 0.0434  
====> Epoch: 71 Average loss: 0.0433  
====> Epoch: 72 Average loss: 0.0432  
====> Epoch: 73 Average loss: 0.0432

====> Epoch: 74 Average loss: 0.0431  
====> Epoch: 75 Average loss: 0.0431  
====> Epoch: 76 Average loss: 0.0431  
====> Epoch: 77 Average loss: 0.0429  
====> Epoch: 78 Average loss: 0.0429  
====> Epoch: 79 Average loss: 0.0428  
====> Epoch: 80 Average loss: 0.0428  
====> Epoch: 81 Average loss: 0.0428  
====> Epoch: 82 Average loss: 0.0426  
====> Epoch: 83 Average loss: 0.0426  
====> Epoch: 84 Average loss: 0.0425  
====> Epoch: 85 Average loss: 0.0425  
====> Epoch: 86 Average loss: 0.0424  
====> Epoch: 87 Average loss: 0.0424  
====> Epoch: 88 Average loss: 0.0423  
====> Epoch: 89 Average loss: 0.0423  
====> Epoch: 90 Average loss: 0.0422  
====> Epoch: 91 Average loss: 0.0421  
====> Epoch: 92 Average loss: 0.0421  
====> Epoch: 93 Average loss: 0.0421  
====> Epoch: 94 Average loss: 0.0419  
====> Epoch: 95 Average loss: 0.0420  
====> Epoch: 96 Average loss: 0.0419  
====> Epoch: 97 Average loss: 0.0419  
====> Epoch: 98 Average loss: 0.0418  
====> Epoch: 99 Average loss: 0.0418  
====> Epoch: 100 Average loss: 0.0417  
====> Epoch: 101 Average loss: 0.0417  
====> Epoch: 102 Average loss: 0.0417  
====> Epoch: 103 Average loss: 0.0416  
====> Epoch: 104 Average loss: 0.0416  
====> Epoch: 105 Average loss: 0.0416  
====> Epoch: 106 Average loss: 0.0415  
====> Epoch: 107 Average loss: 0.0414  
====> Epoch: 108 Average loss: 0.0414  
====> Epoch: 109 Average loss: 0.0414  
====> Epoch: 110 Average loss: 0.0414  
====> Epoch: 111 Average loss: 0.0413  
====> Epoch: 112 Average loss: 0.0413  
====> Epoch: 113 Average loss: 0.0413  
====> Epoch: 114 Average loss: 0.0412  
====> Epoch: 115 Average loss: 0.0412  
====> Epoch: 116 Average loss: 0.0412  
====> Epoch: 117 Average loss: 0.0411  
====> Epoch: 118 Average loss: 0.0411  
====> Epoch: 119 Average loss: 0.0411  
====> Epoch: 120 Average loss: 0.0410  
====> Epoch: 121 Average loss: 0.0410

====> Epoch: 122 Average loss: 0.0410  
====> Epoch: 123 Average loss: 0.0409  
====> Epoch: 124 Average loss: 0.0409  
====> Epoch: 125 Average loss: 0.0409  
====> Epoch: 126 Average loss: 0.0409  
====> Epoch: 127 Average loss: 0.0408  
====> Epoch: 128 Average loss: 0.0408  
====> Epoch: 129 Average loss: 0.0408  
====> Epoch: 130 Average loss: 0.0407  
====> Epoch: 131 Average loss: 0.0407  
====> Epoch: 132 Average loss: 0.0407  
====> Epoch: 133 Average loss: 0.0407  
====> Epoch: 134 Average loss: 0.0406  
====> Epoch: 135 Average loss: 0.0406  
====> Epoch: 136 Average loss: 0.0406  
====> Epoch: 137 Average loss: 0.0405  
====> Epoch: 138 Average loss: 0.0405  
====> Epoch: 139 Average loss: 0.0405  
====> Epoch: 140 Average loss: 0.0405  
====> Epoch: 141 Average loss: 0.0404  
====> Epoch: 142 Average loss: 0.0404  
====> Epoch: 143 Average loss: 0.0405  
====> Epoch: 144 Average loss: 0.0403  
====> Epoch: 145 Average loss: 0.0404  
====> Epoch: 146 Average loss: 0.0404  
====> Epoch: 147 Average loss: 0.0403  
====> Epoch: 148 Average loss: 0.0403  
====> Epoch: 149 Average loss: 0.0403



====> Test loss: 0.0404

### 1.3.4 Visualise the reconstructions obtained from the three models

```
In [22]: data_iterator = iter(test_set_loader)
         images, labels = data_iterator.next()

# This function takes as an input the images to reconstruct
# and the name of the model with which the reconstructions
# are performed
def to_img(x):
    x = 0.5 * (x + 1)
    x = x.clamp(0, 1)
    x = x.view(x.size(0), 1, 28, 28)
    return x

def visualise_output(images, model):
    data = images
    img = data
    img = img.view(img.size(0), -1)
    img = Variable(img)
    out = model(img)
    pic = out.data.view(out.data.size(0), 1, 28, 28)
    pic = pic * 0.5 + 0.5
```

```

np_pic = torchvision.utils.make_grid(pic[1:50], 10, 5).numpy()
plt.imshow(np.transpose(np_pic, (1, 2, 0)))
plt.show()

```

```

In [23]: # First visualise the original images
print('Original images')
show_image(torchvision.utils.make_grid(images[1:50],10,5))
plt.show()

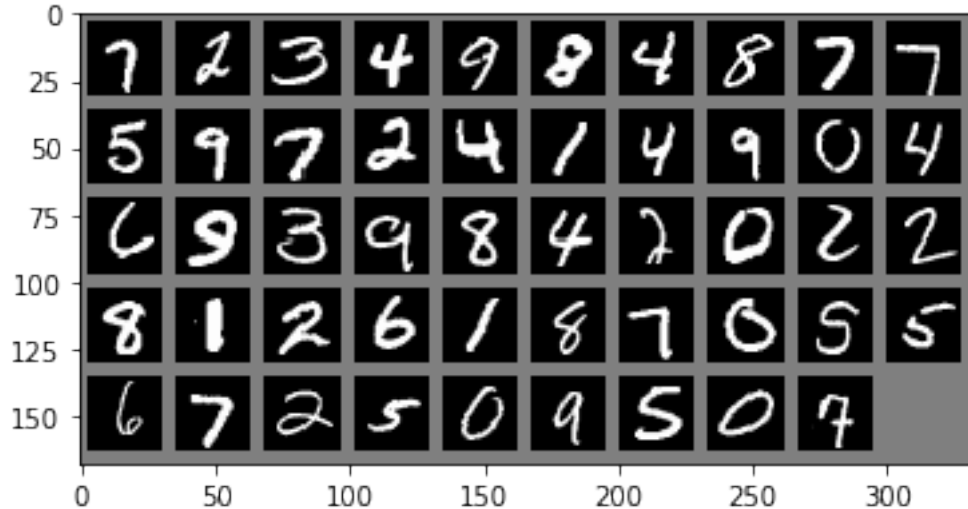
# Reconstruct and visualise the images using the first model
print('PCA based autoencoder reconstruction:')
visualise_output(images, model1)

# Reconstruct and visualise the images using the second model
print('Shallow autoencoder reconstruction:')
visualise_output(images, model2)

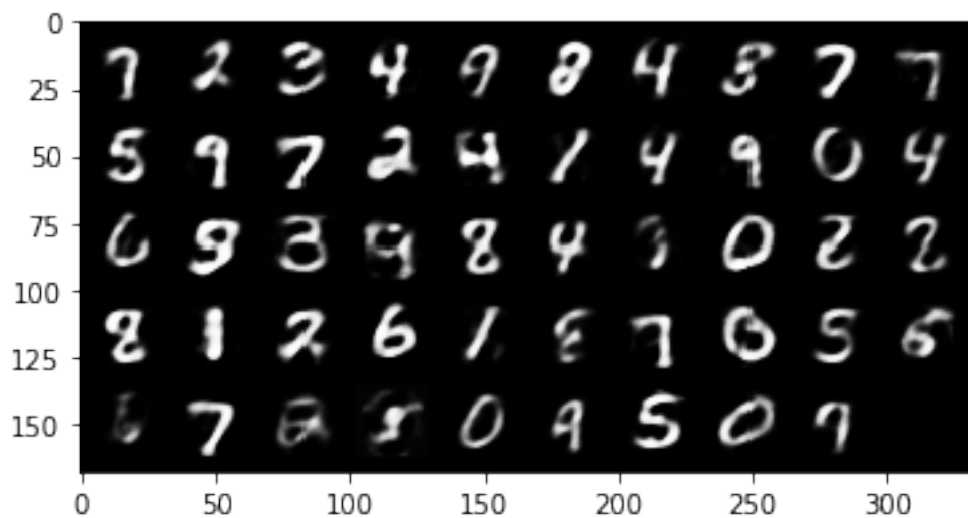
# Reconstruct and visualise the images using the third model
print('Deep autoencoder reconstruction:')
visualise_output(images, model3)

```

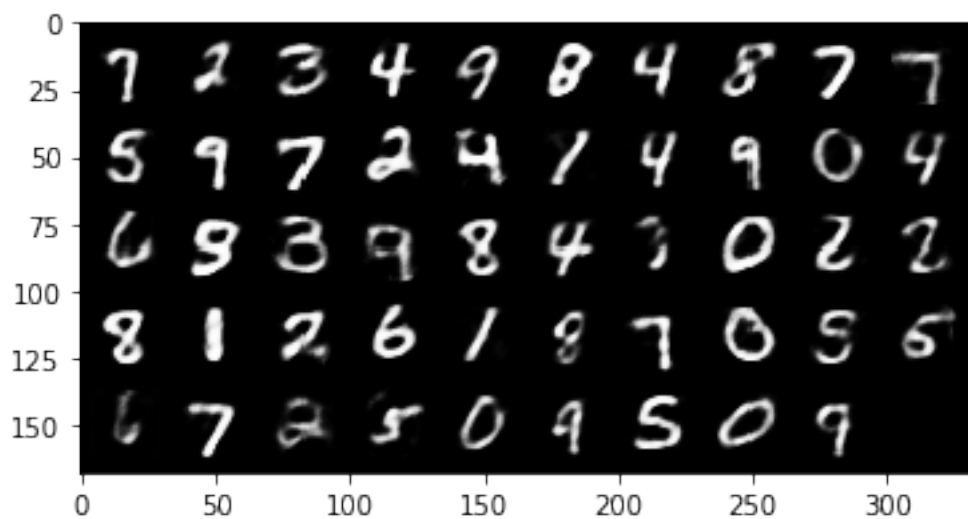
Original images



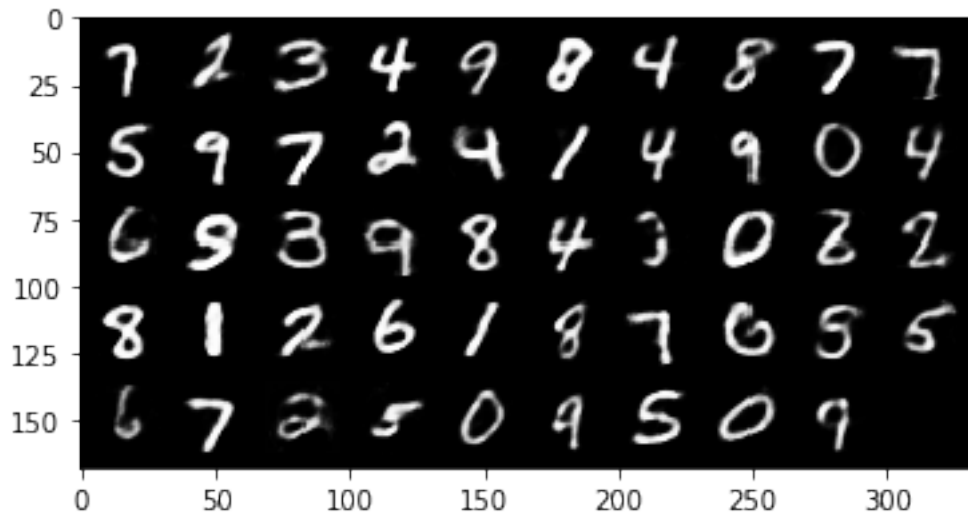
PCA based autoencoder reconstruction:



Shallow autoencoder reconstruction:



Deep autoencoder reconstruction:



## 1.4 Part B

In this part of the assignment we need to interpolate between two images using the latent space representation obtained by the PCA-based autoencoder and the deep autoencoder and subsequently to verify that the interpolation generated using the deep autoencoder is better.

In [24]: *#Extract images to apply interpolation*

```

one = []
four = []
seven = []
eight = []

for i in range(len(train_set)):
    if (train_set[i][1]==1):
        one.append(train_set[i])

    if (train_set[i][1]==4):
        four.append(train_set[i])

    if (train_set[i][1]==7):
        seven.append(train_set[i])

    if (train_set[i][1]==8):
        eight.append(train_set[i])

img1= seven[0][0]
npimg = img1.numpy()
im7 = npimg.reshape(28, 28)

```

```

img2 = one[0][0]
npimg = img2.numpy()
im1 = npimg.reshape(28, 28)

img3 = four[0][0]
npimg = img3.numpy()
im4 = npimg.reshape(28, 28)

img4 = eight[0][0]
npimg = img4.numpy()
im8 = npimg.reshape(28, 28)

```

```

In [25]: f, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4)
ax1.imshow(im1, cmap='gray')
ax1.axis('off')
ax2.imshow(im4, cmap='gray')
ax2.axis('off')
ax3.imshow(im7, cmap='gray')
ax3.axis('off')
ax4.imshow(im8, cmap='gray')
ax4.axis('off')
plt.show()

```



```

In [26]: sevens = torch.Tensor(seven[0][0])
ones = torch.Tensor(one[0][0])
fours = torch.Tensor(four[0][0])
eights = torch.Tensor(eight[0][0])

```

**interpolation()** function initially obtains the latent space representation of *image1* and *image2* and apply interpolation. Then, it obtains and returns the interpolated image space representation.

```

In [27]: def interpolation(lambda1, model, image1, image2):

    img1= image1
    img1 = img1.view(img1.size(0), -1)
    img1 = Variable(img1)

```



```

##### Question 1 and 2 #####
#Latent space interpretation of image1
latent_1 =

img2= image2
img2 = img2.view(img1.size(0), -1)
img2 = Variable(img2)
#Latent space interpretation of image2
latent_2 =

#Interpolation of the two images
inter_image_lat =

inter_image =

##### END of question 1 and 2 #####

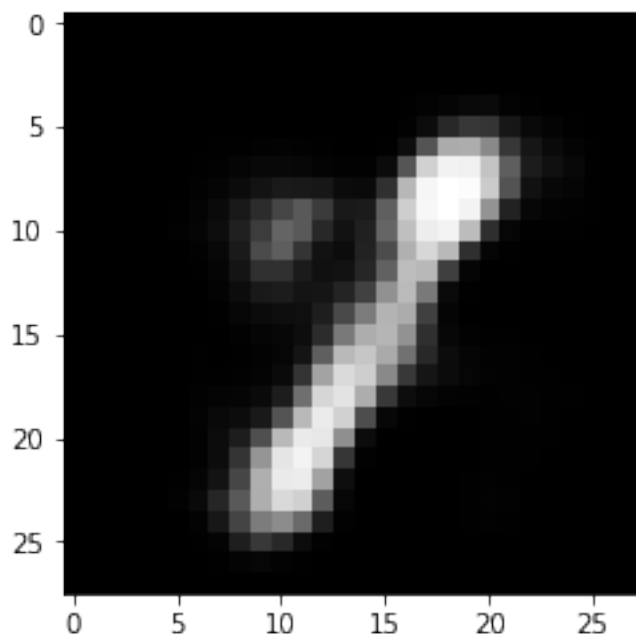
pic = to_img(image_lambda)
image = pic.data.numpy()
image = image.reshape(28,28)
plt.imshow(image, cmap='gray')
plt.show()

```

Interpolation between digits *seven* and *one* using the PCA based autoencoder.

In [32]: ##### Question 1 #####

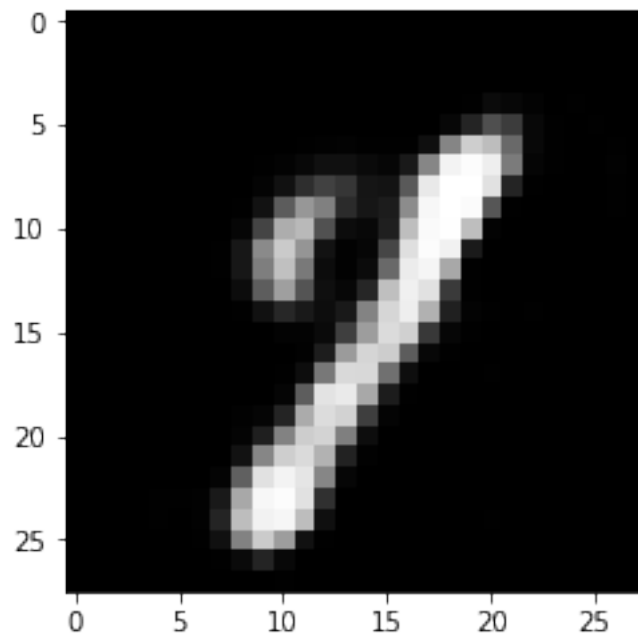
```
interpolation(0.4, model1, sevens, ones)
```



Interpolation between digits *seven* and *one* using the deep autoencoder.

```
In [33]: ##### Question 2 #####
```

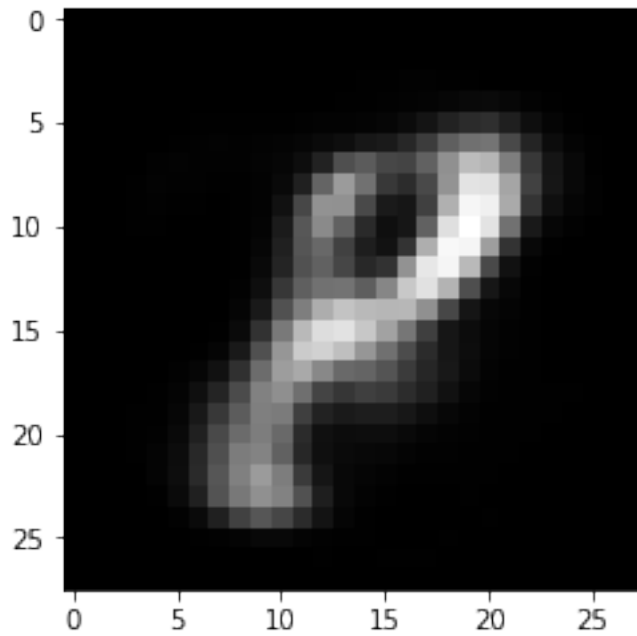
```
interpolation(0.4, model3, sevens, ones)
```



Interpolation between digits *four* and *eight* using the PCA based autoencoder.

```
In [34]: ##### Question 1 #####
```

```
interpolation(0.4, model1, fours, eights)
```



Interpolation between digits *four* and *eight* using the deep autoencoder.

In [35]: ##### Question 2 #####

```
interpolation(0.4, model3, fours, eights)
```

