PA4_EE21S056

November 4, 2022

Akash Sharma (EE21S056)

EE5179: Deep Learning for Imaging

1 Comparing PCA and Autoencoders

1.1 Importing packages

[1]:
```
import torch
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor, Lambda
from torch import nn
from torchvision.utils import make_grid
import matplotlib.pyplot as plt
import numpy as np
```

1.2 Loading MNIST dataset

[2]:
```
training_data = datasets.MNIST(root="data", train=True, download =␣
 ↪True,transform = ToTensor())
test_data = datasets.MNIST(root="data", train = False, download =␣
 ↪True,transform = ToTensor())

batch_size = 64

train_loader = DataLoader(dataset=training_data, batch_size=batch_size,␣
 ↪shuffle=True)
test_loader = DataLoader(dataset=test_data, batch_size=batch_size,␣
 ↪shuffle=False)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to
data/MNIST/raw/train-images-idx3-ubyte.gz

  0%|          | 0/9912422 [00:00<?, ?it/s]
```

```
Extracting data/MNIST/raw/train-images-idx3-ubyte.gz to data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to
data/MNIST/raw/train-labels-idx1-ubyte.gz

  0%|          | 0/28881 [00:00<?, ?it/s]

Extracting data/MNIST/raw/train-labels-idx1-ubyte.gz to data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to
data/MNIST/raw/t10k-images-idx3-ubyte.gz

  0%|          | 0/1648877 [00:00<?, ?it/s]

Extracting data/MNIST/raw/t10k-images-idx3-ubyte.gz to data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to
data/MNIST/raw/t10k-labels-idx1-ubyte.gz

  0%|          | 0/4542 [00:00<?, ?it/s]

Extracting data/MNIST/raw/t10k-labels-idx1-ubyte.gz to data/MNIST/raw
```

1.3 MNIST Image Reconstruction using PCA

Reference used: https://analyticsindiamag.com/guide-to-image-reconstruction-using-principal-component-analysis/

[3]:
```python
# Extracting just images from the dataset for PCA

images = training_data.data
print(images.shape)
```

```
torch.Size([60000, 28, 28])
```

1.3.1 Finding indices representing each class

[4]:
```python
# 114-0, 102-1, 117-2, 111-3, 115-4, 145-5, 106-6, 103-7, 125-8, 110-9

idx_PCA = [114, 102, 117, 111, 115, 145, 106, 103, 125, 110]

i=1 #just an iterator variable

for idx in idx_PCA:
  plt.subplot(2,5,i)
  plt.imshow(images[idx].reshape(28,28), cmap='gray')
  plt.axis('off')
```

```
  i+=1
plt.tight_layout()
plt.show()
```

1.3.2 Flattening PCA input images

[5]:
```
data = np.zeros((60000, 784))
k=0
for img in images:
  data[k] = img.squeeze().flatten()
  data[k] = data[k]/255
  k+=1
print(data.shape)
```

```
(60000, 784)
```

1.3.3 Performing PCA and checking reconstruction error

[6]:
```
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error
pca = PCA(n_components=30)
reconst_data = pca.inverse_transform(pca.fit_transform(data))

i = 1

for idx in idx_PCA:
  plt.subplot(2,5,i)
  plt.imshow(reconst_data[idx].reshape(28,28), cmap='gray')
  plt.axis('off')
```

```
  i+=1

plt.tight_layout()
plt.show()

print('Reconstruction error for PCA is ', mean_squared_error(data, reconst_data))
```

```
Reconstruction error for PCA is  0.018121815941217218
```

1.4 Using an Autoencoder on MNIST dataset

1.4.1 Defining stacked autoencoder class

[7]:
```
class stack_AE(nn.Module):
  def __init__(self):
    super(stack_AE, self).__init__()
    self.flatten = nn.Flatten()
    self.encoder = nn.Sequential(
        nn.Linear(784,512),
        nn.ReLU(),
        nn.Linear(512,256),
        nn.ReLU(),
        nn.Linear(256,128),
        nn.ReLU(),
        nn.Linear(128,30),
        nn.ReLU())
    self.decoder =nn.Sequential(
        nn.Linear(30,128),
```

```
        nn.ReLU(),
        nn.Linear(128,256),
        nn.ReLU(),
        nn.Linear(256,784),
        nn.ReLU())

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

1.4.2 Autoencoder Training

[8]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

learning_rate = 0.001
epochs = 10

criterion = nn.MSELoss() # Reconstruction error is defined as the MSE Loss

model_stackAE = stack_AE().to(device)
optimizer = torch.optim.Adam(model_stackAE.parameters(), learning_rate)

train_batch_loss = []
train_epoch_loss = []
val_loss = []
```

[9]:
```
for epoch in range(epochs):
  for batch_idx, (data, target) in enumerate(train_loader):
    data = data.to(device)
    target = target.to(device)

    model_stackAE.train()
    decoded, encoded = model_stackAE(data)
    loss = criterion(decoded, torch.flatten(data,1))
    train_batch_loss.append(loss.item())

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  train_epoch_loss.append(train_batch_loss[-1])

  model_stackAE.eval()
  with torch.no_grad():
    for (data, target) in test_loader:
```

```
        data = data.to(device)
        target = target.to(device)
        decoded, encoded = model_stackAE(data)
        val_iter_loss = criterion(decoded, torch.flatten(data,1))

    val_loss.append(val_iter_loss.item())
    print('Epoch Number: ',epoch+1)
    print('Training loss: ', train_epoch_loss[-1])
    print('Validation loss: ', val_loss[-1])
    print('-----------------------------------------')

print('Training done!')
```

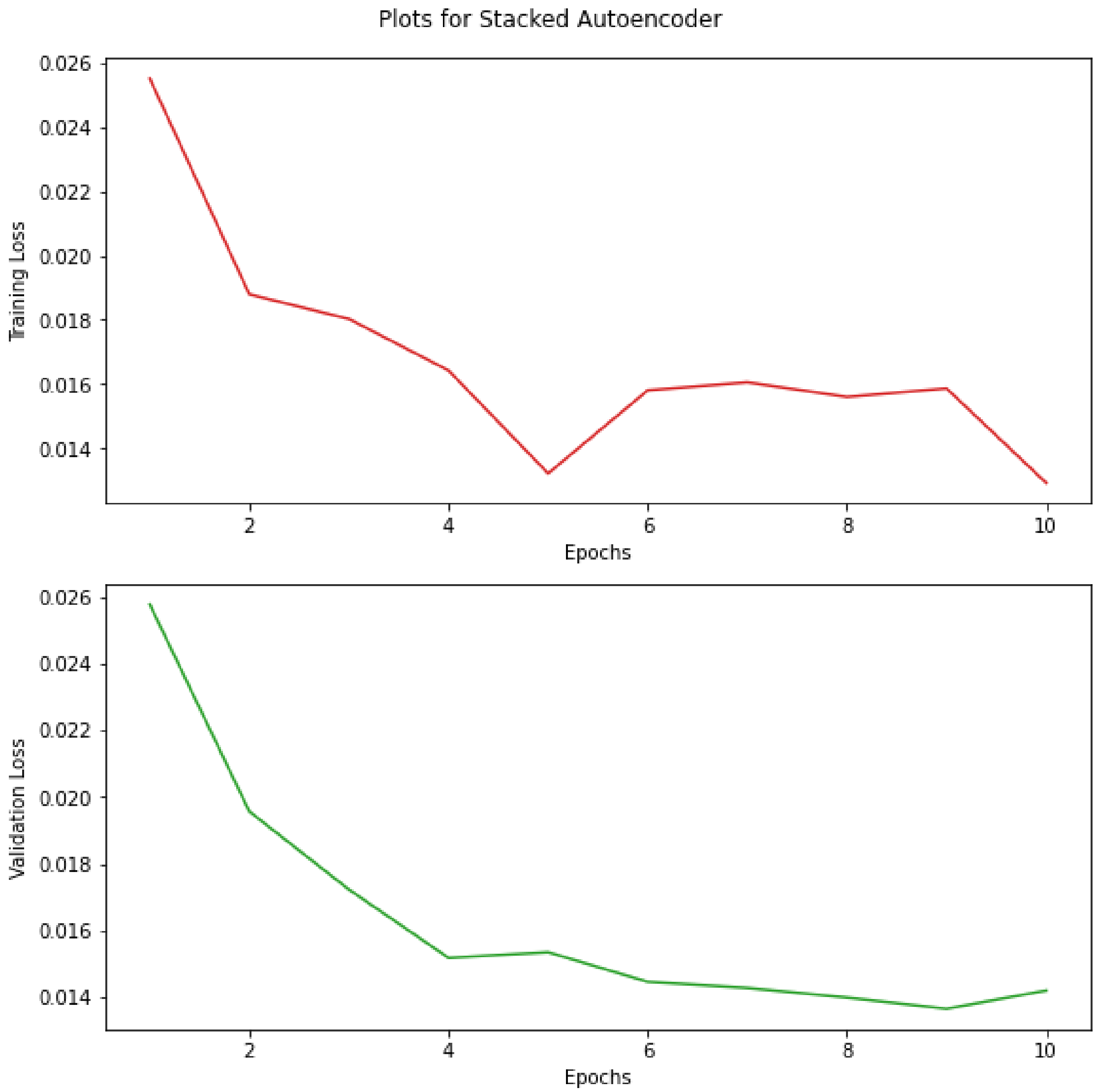
```
Epoch Number:  1
Training loss:  0.025514619424939156
Validation loss:  0.025768259540200233
-----------------------------------------
Epoch Number:  2
Training loss:  0.018792742863297462
Validation loss:  0.01956520602107048
-----------------------------------------
Epoch Number:  3
Training loss:  0.018027035519480705
Validation loss:  0.017227349802851677
-----------------------------------------
Epoch Number:  4
Training loss:  0.016422612592577934
Validation loss:  0.015180998481810093
-----------------------------------------
Epoch Number:  5
Training loss:  0.013217149302363396
Validation loss:  0.015349737368524075
-----------------------------------------
Epoch Number:  6
Training loss:  0.01580682583153248
Validation loss:  0.014463952742516994
-----------------------------------------
Epoch Number:  7
Training loss:  0.016053110361099243
Validation loss:  0.01427979115396738
-----------------------------------------
Epoch Number:  8
Training loss:  0.015601414255797863
Validation loss:  0.013991869054734707
-----------------------------------------
Epoch Number:  9
Training loss:  0.01586502604186535
Validation loss:  0.013657165691256523
```

```
-----------------------------------------
Epoch Number:  10
Training loss:  0.012922598049044609
Validation loss:  0.014202686958014965
-----------------------------------------
Training done!
```

1.4.3 Visualizing training and validation loss for autoencoder

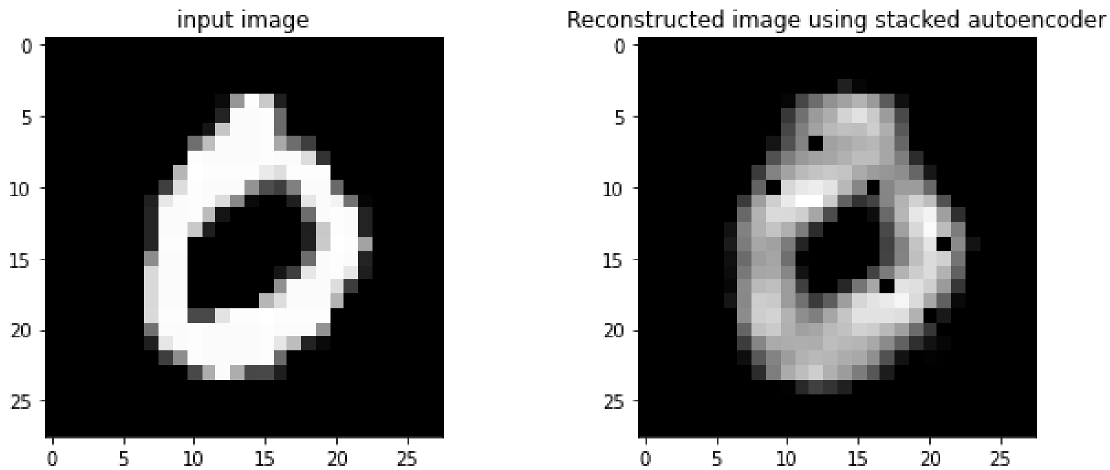
[10]:
```
fig, axs = plt.subplots(2, constrained_layout= True, figsize=(8,8))
fig.suptitle('Plots for Stacked Autoencoder')
axs[0].set(ylabel='Training Loss', xlabel='Epochs')
axs[0].plot(range(1,epochs+1),train_epoch_loss, 'tab:red')
axs[1].set(ylabel='Validation Loss', xlabel='Epochs')
axs[1].plot(range(1,epochs+1),val_loss, 'tab:green')
```

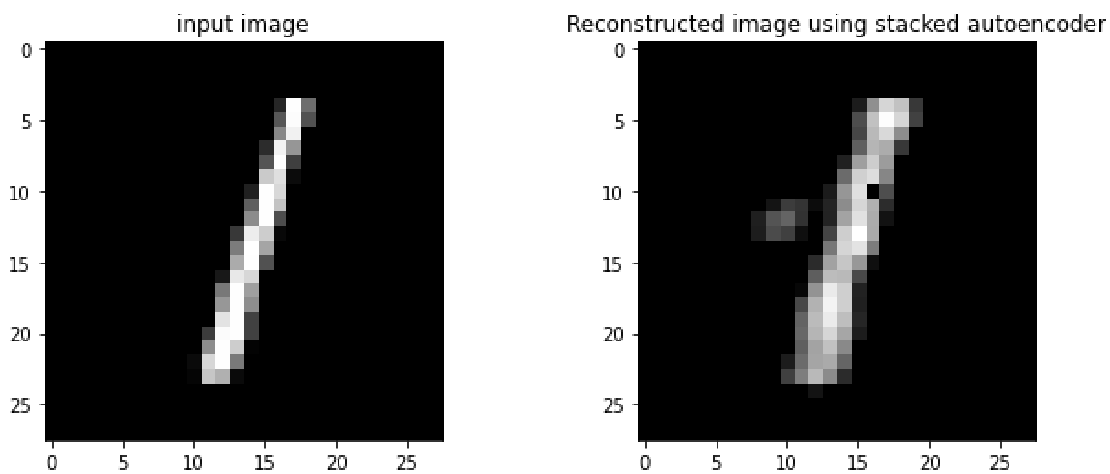
[10]:
```
[<matplotlib.lines.Line2D at 0x7fbd686d7a50>]
```

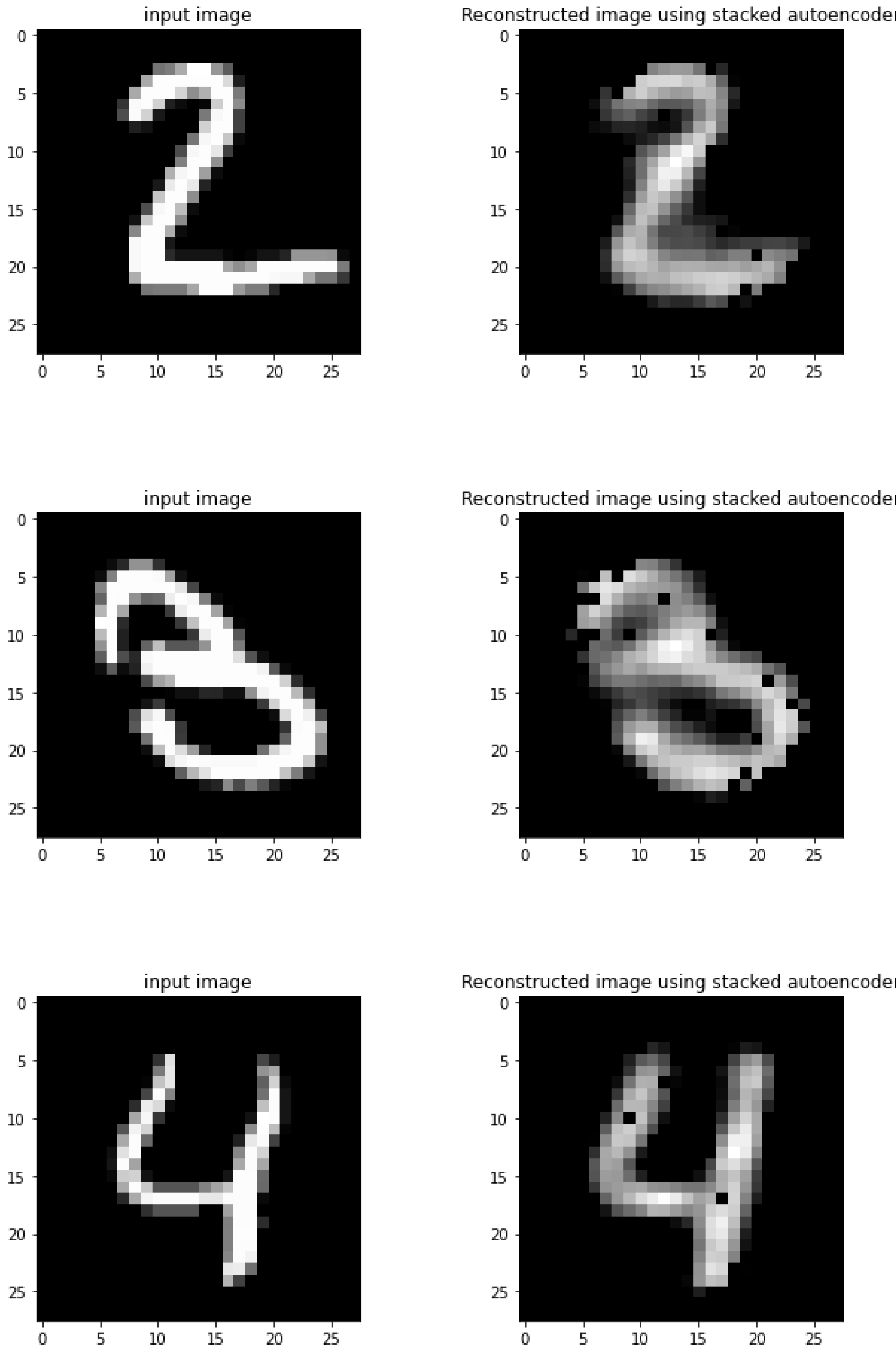


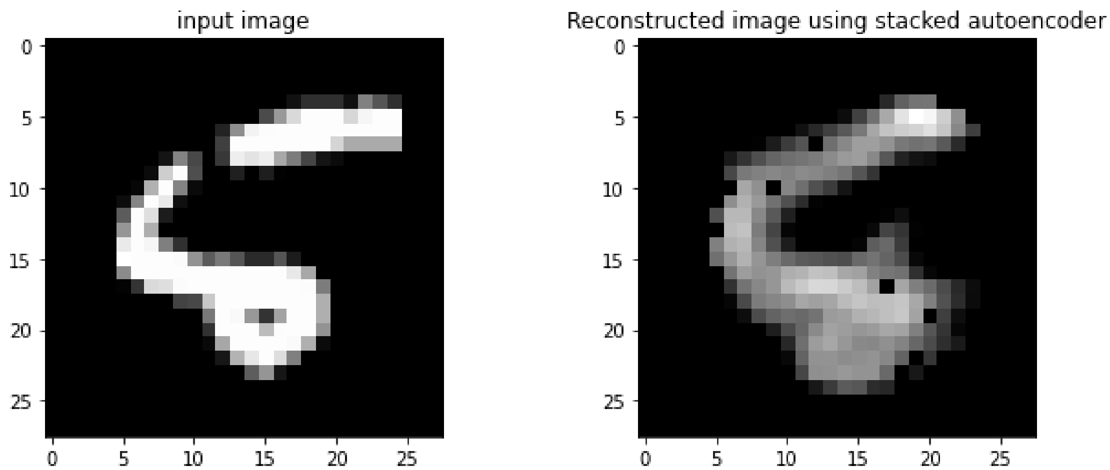
[11]:
```
idx_test=[3,2,1,18,4,8,11,0,84,7]

for idx in idx_test:
  image = test_loader.dataset.data[idx, :, :].clone()
  with torch.no_grad():
    image = image.view(-1,28,28).to(device=device).float()
    decoded, encoded = model_stackAE.forward(image)
    decoded = decoded.detach().cpu().numpy()
    image = image.reshape(28,28).detach().cpu().numpy()
    plt.subplot(1,2,1)
    plt.imshow(image , cmap='gray')
    plt.title('input image')
    plt.subplots_adjust(right=1.5)
```

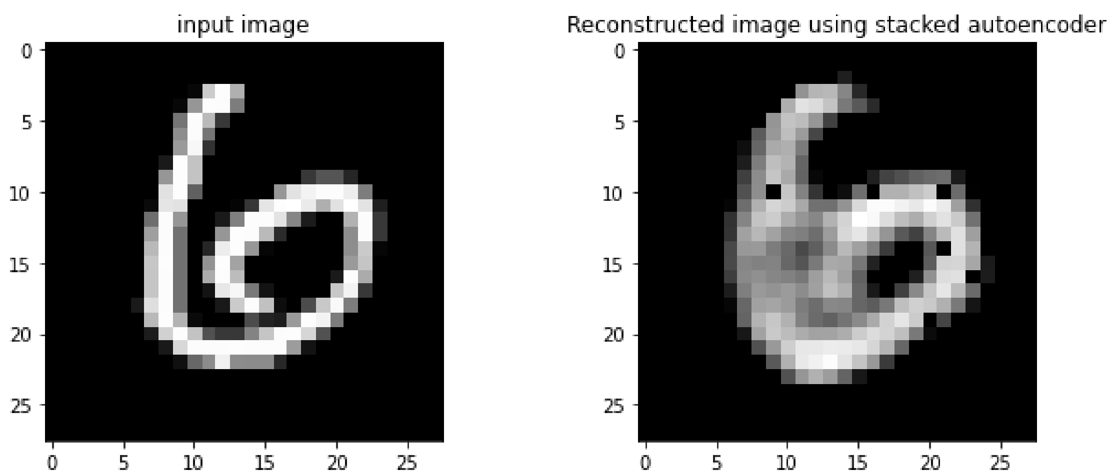
```
plt.subplot(1,2,2)
plt.imshow(decoded.reshape(28,28),cmap ='gray')
plt.title("Reconstructed image using stacked autoencoder")
plt.show()
```

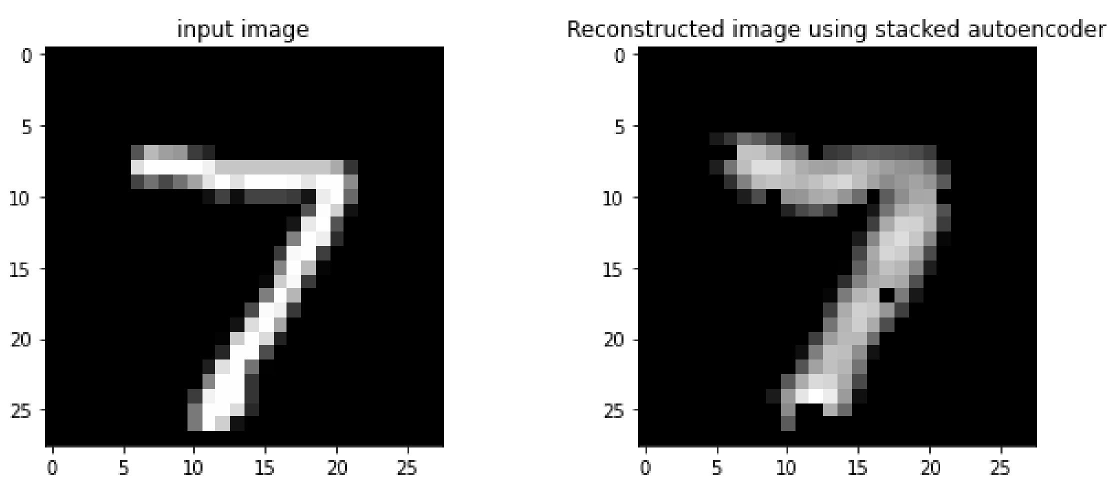


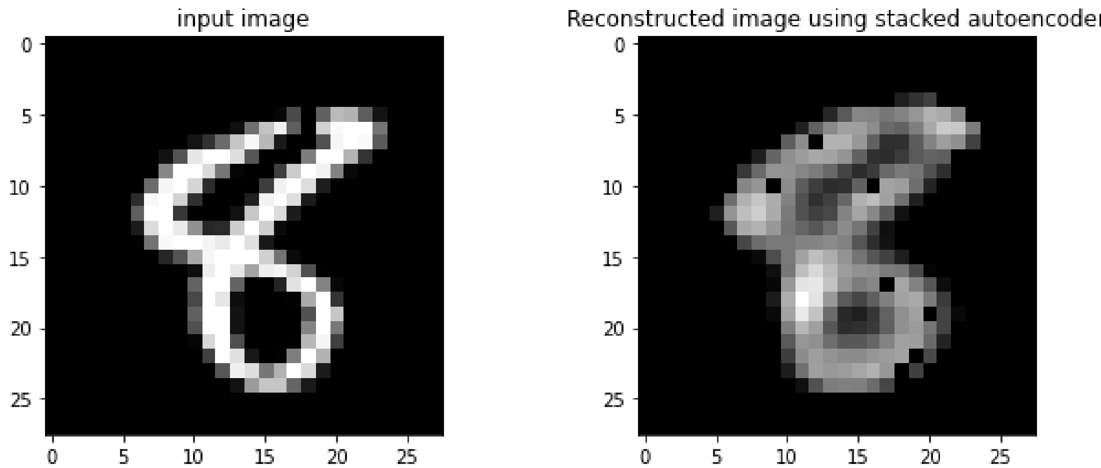


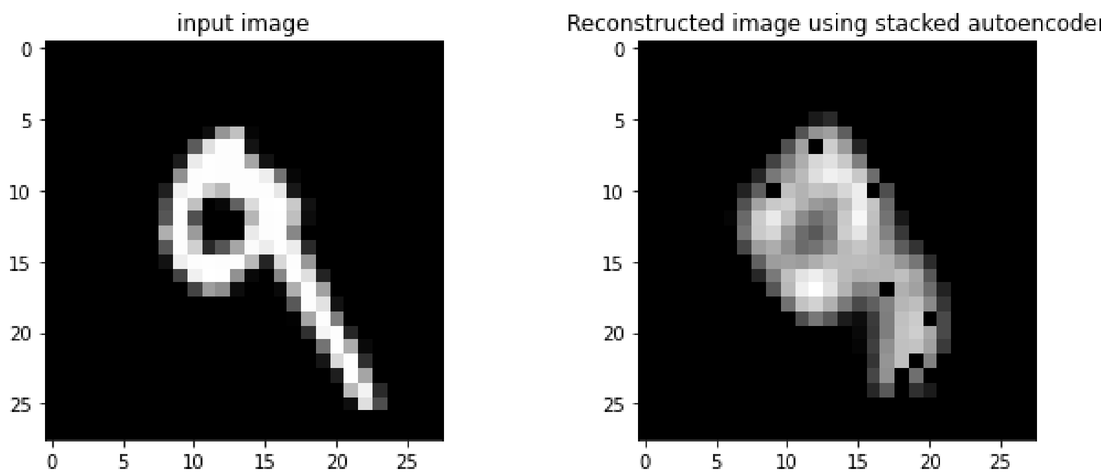

input image
Reconstructed image using stacked autoencoder
input image
Reconstructed image using stacked autoencoder
input image
Reconstructed image using stacked autoencoder


input image
Reconstructed image using stacked autoencoder


input image
Reconstructed image using stacked autoencoder


input image
Reconstructed image using stacked autoencoder





[12]:
```
model_stackAE.eval()
test_loss = 0
with torch.no_grad():
  for (data, target) in test_loader:
    data = data.to(device)
    target = target.to(device)
    decoded, encoded = model_stackAE(data)
    val_iter_loss = criterion(decoded, torch.flatten(data,1))
    test_loss += val_iter_loss/len(test_loader)
```

```
print('Reconstruction accuracy with the given autoencoder configuration is ',␣
↪test_loss.item())
```

```
Reconstruction accuracy with the given autoencoder configuration is
0.014440055005252361
```

2 Experimenting with hidden units of varying sizes

2.1 Defining standard autoencoder class

The following model takes the size of hidden layer as input as required by the experiment.

[13]:
```
class standard_AE(nn.Module):
  def __init__(self,hidden_layer):
    super(standard_AE, self).__init__()
    self.flatten = nn.Flatten()
    self.encoder = nn.Sequential(
        nn.Linear(784,hidden_layer),
        nn.ReLU())
    self.decoder =nn.Sequential(
        nn.Linear(hidden_layer,784),
        nn.ReLU())


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

2.2 Loading Fashion MNIST for experiment

[14]:
```
fash_test_data = datasets.FashionMNIST(root="data", train = False, download =␣
↪True,transform = ToTensor())
fash_test_loader = DataLoader(dataset=fash_test_data, batch_size=batch_size,␣
↪shuffle=False)
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz to data/FashionMNIST/raw/train-images-idx3-ubyte.gz

  0%|          | 0/26421880 [00:00<?, ?it/s]

Extracting data/FashionMNIST/raw/train-images-idx3-ubyte.gz to
data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz to data/FashionMNIST/raw/train-labels-idx1-ubyte.gz

  0%|          | 0/29515 [00:00<?, ?it/s]

Extracting data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to
data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to
data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz

  0%|          | 0/4422102 [00:00<?, ?it/s]

Extracting data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to
data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to
data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz

  0%|          | 0/5148 [00:00<?, ?it/s]

Extracting data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to
data/FashionMNIST/raw
```

2.3 Training and experiment results

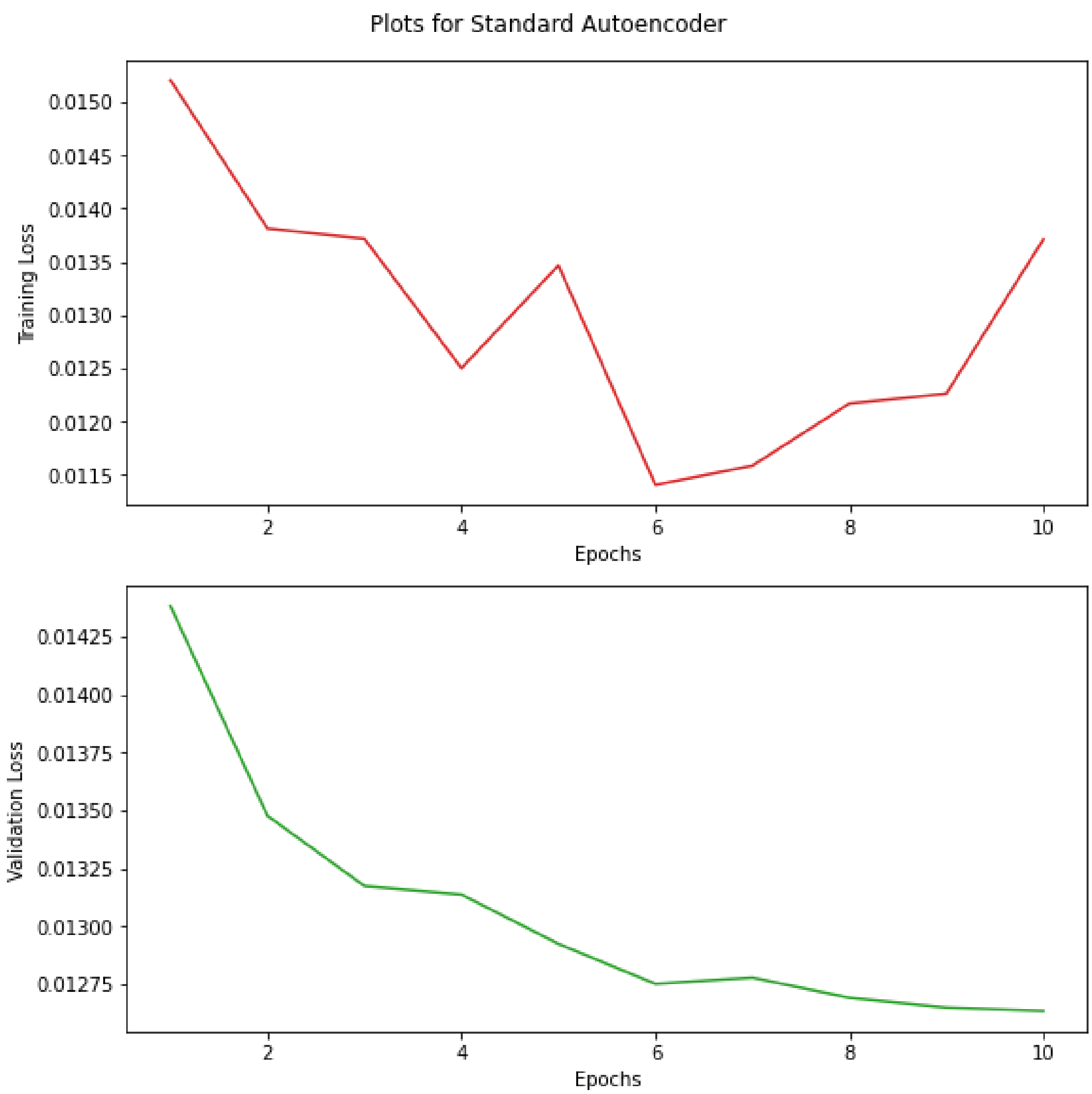
[15]:
```python
hid_size = [64, 128, 256]

for size in hid_size:
  model_standardAE = standard_AE(size).to(device)
  optimizer = torch.optim.Adam(model_standardAE.parameters(), learning_rate)

  train_batch_loss = []
  train_epoch_loss = []
  val_loss = []
  print('Training begins for hidden unit size ', size, '!')
  print('------------------------------------------------------------')
  for epoch in range(epochs):
    for batch_idx, (data, target) in enumerate(train_loader):
      data = data.to(device=device)
      target = target.to(device=device)

      model_standardAE.train()
```

```
    decoded, encoded = model_standardAE(data)
    loss = criterion(decoded, torch.flatten(data,1))
    train_batch_loss.append(loss.item())

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  train_epoch_loss.append(train_batch_loss[-1])

  model_standardAE.eval()
  with torch.no_grad():
    for (data, target) in test_loader:
      data = data.to(device)
      target = target.to(device)
      decoded, encoded = model_standardAE(data)
      val_iter_loss = criterion(decoded, torch.flatten(data,1))

  val_loss.append(val_iter_loss.item())
  print('Epoch Number: ',epoch+1)
  print('Training loss: ', train_epoch_loss[-1])
  print('Validation loss: ', val_loss[-1])
  print('-----------------------------------------')


fig, axs = plt.subplots(2, constrained_layout= True, figsize=(8,8))
fig.suptitle('Plots for Standard Autoencoder')
axs[0].set(ylabel='Training Loss', xlabel='Epochs')
axs[0].plot(range(1,epochs+1),train_epoch_loss, 'tab:red')
axs[1].set(ylabel='Validation Loss', xlabel='Epochs')
axs[1].plot(range(1,epochs+1),val_loss, 'tab:green')
plt.show()
print('Training done for hidden unit size ', size, '!')
print('-----------------------------------------------------------')
image = test_loader.dataset.data[1, :, :].clone() #take one image from test set
with torch.no_grad():
  image = image.view(-1,28,28).to(device=device).float()
  decoded, encoded = model_standardAE.forward(image)
  decoded = decoded.detach().cpu().numpy()
  image = image.reshape(28,28).detach().cpu().numpy()
  plt.subplot(1,2,1)
  plt.imshow(image , cmap='gray')
  plt.title('input image')
  plt.subplots_adjust(right=1.5)
  plt.subplot(1,2,2)
  plt.imshow(decoded.reshape(28,28),cmap ='gray')
```

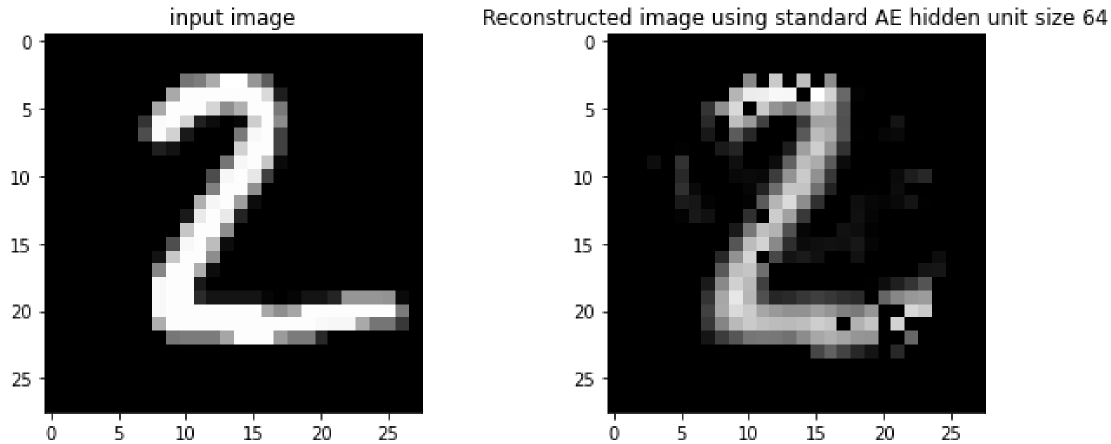
```
    plt.title("Reconstructed image using standard AE hidden unit size␣
↪"+str(size))
    plt.show()

    fash_image = fash_test_loader.dataset.data[12, :, :].clone()
    fash_image = fash_image.view(-1,28,28).to(device=device).float()
    decoded, encoded = model_standardAE.forward(fash_image)
    decoded = decoded.detach().cpu().numpy()
    fash_image = fash_image.reshape(28,28).detach().cpu().numpy()
    plt.subplot(1,2,1)
    plt.imshow(fash_image , cmap='gray')
    plt.title('input image')
    plt.subplots_adjust(right=1.5)
    plt.subplot(1,2,2)
    plt.imshow(decoded.reshape(28,28),cmap ='gray')
    plt.title("Reconstructed image using standard AE hidden unit size␣
↪"+str(size))
    plt.show()


    noisy_img = np.random.normal(loc=128,scale=10,size=(28,28))
    noisy_img = torch.from_numpy(noisy_img).reshape(1,1,28,28).
↪to(device=device).float()
    decoded, encoded = model_standardAE.forward(noisy_img)
    decoded = decoded.detach().cpu().numpy()
    noisy_img = noisy_img.reshape(28,28).detach().cpu().numpy()
    plt.subplot(1,2,1)
    plt.imshow(noisy_img , cmap='gray')
    plt.title('input image')
    plt.subplots_adjust(right=1.5)
    plt.subplot(1,2,2)
    plt.imshow(decoded.reshape(28,28),cmap ='gray')
    plt.title("Reconstructed image using standard AE hidden unit size␣
↪"+str(size))
    plt.show()
    print('-----------------------------------------------------------')
    print('-----------------------------------------------------------')
```

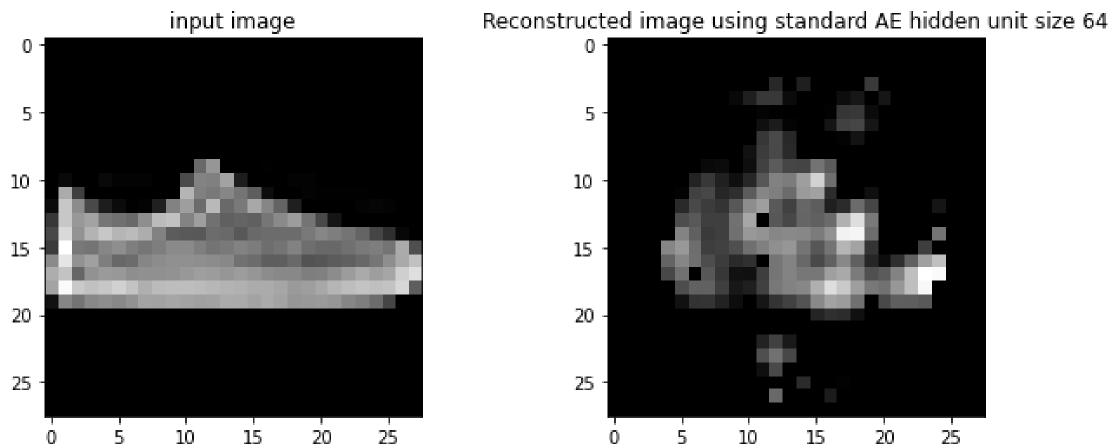
```
Training begins for hidden unit size  64 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  0.015205412171781063
Validation loss:  0.014379788190126419
-----------------------------------------
Epoch Number:  2
Training loss:  0.013814076781272888
Validation loss:  0.013476143591105938
```

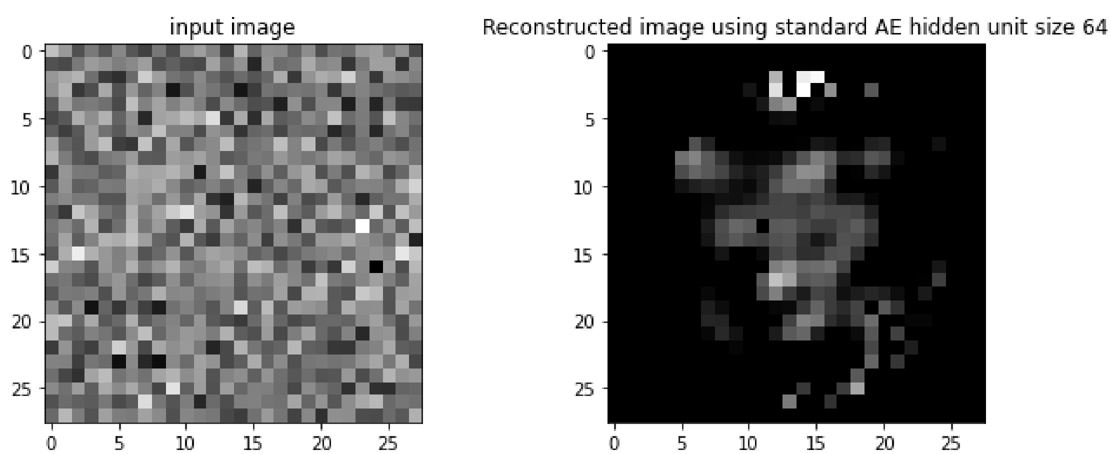
```
-----------------------------------------
Epoch Number:  3
Training loss:  0.013716896064579487
Validation loss:  0.013173110783100128
-----------------------------------------
Epoch Number:  4
Training loss:  0.012501811608672142
Validation loss:  0.013136602006852627
-----------------------------------------
Epoch Number:  5
Training loss:  0.013467206619679928
Validation loss:  0.012923912145197392
-----------------------------------------
Epoch Number:  6
Training loss:  0.011405030265450478
Validation loss:  0.012751072645187378
-----------------------------------------
Epoch Number:  7
Training loss:  0.011586284264922142
Validation loss:  0.012777873314917088
-----------------------------------------
Epoch Number:  8
Training loss:  0.012170090340077877
Validation loss:  0.012691356241703033
-----------------------------------------
Epoch Number:  9
Training loss:  0.012262396514415741
Validation loss:  0.012649083510041237
-----------------------------------------
Epoch Number:  10
Training loss:  0.013712306506931782
Validation loss:  0.012634700164198875
-----------------------------------------
```



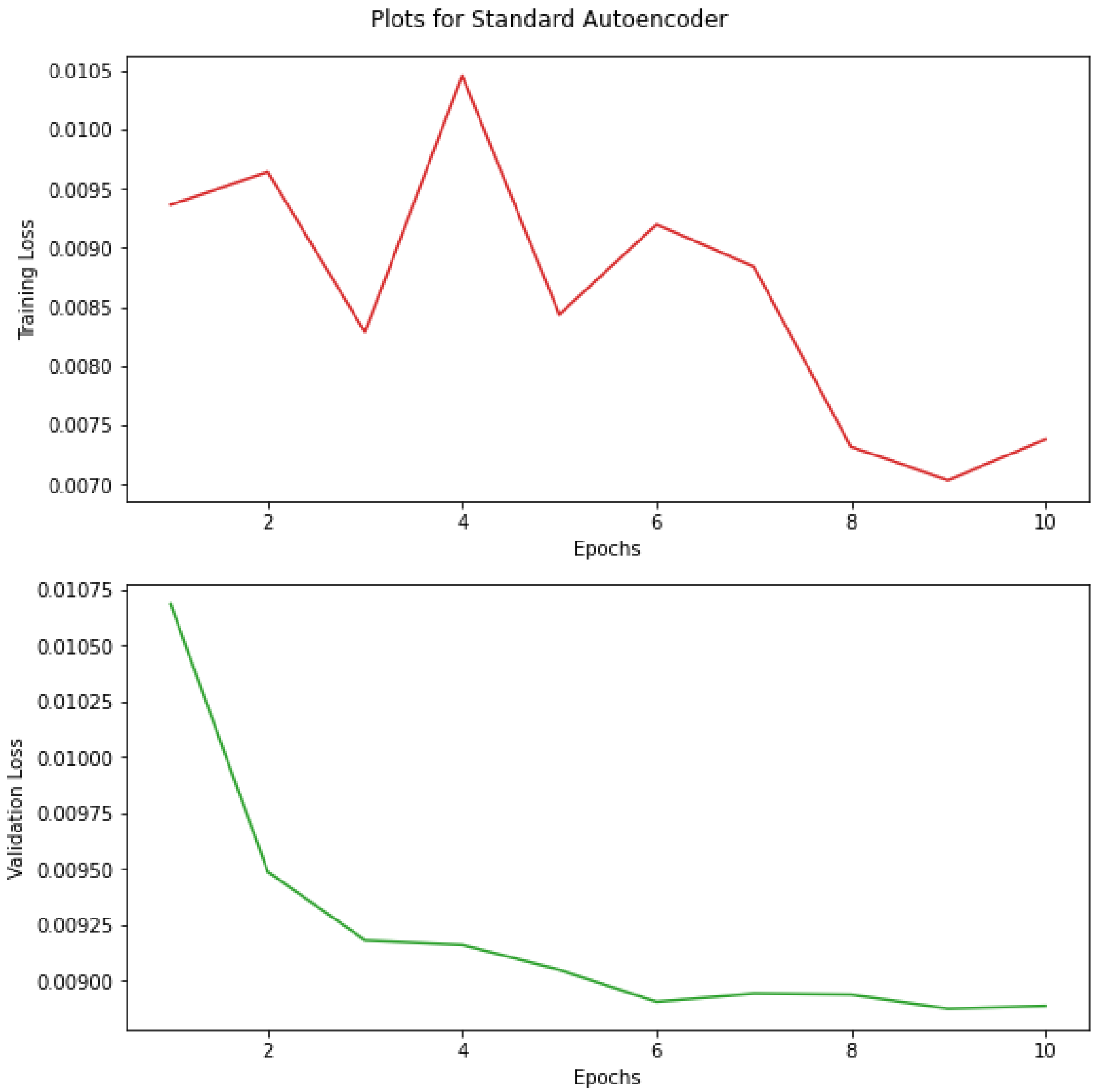

```
Training done for hidden unit size  64 !
-----------------------------------------------------------
```


input image
Reconstructed image using standard AE hidden unit size 64

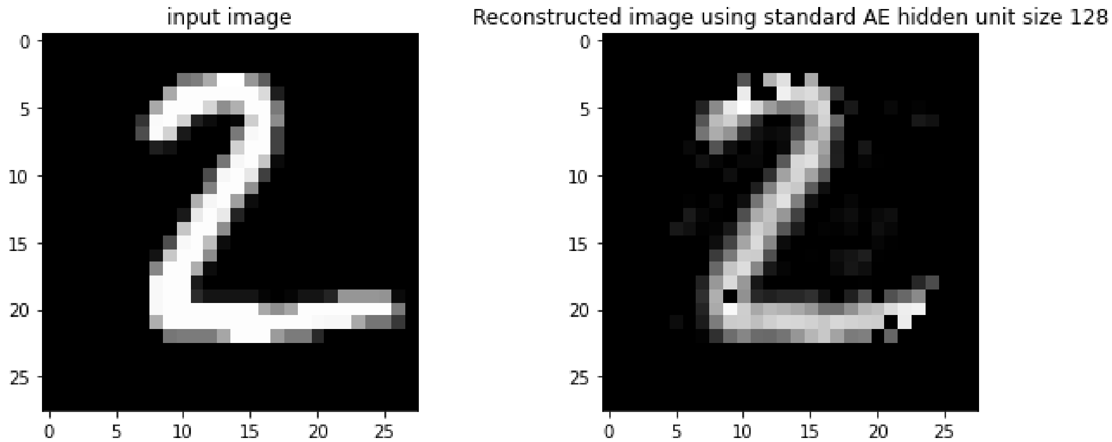

input image
Reconstructed image using standard AE hidden unit size 64


input image
Reconstructed image using standard AE hidden unit size 64

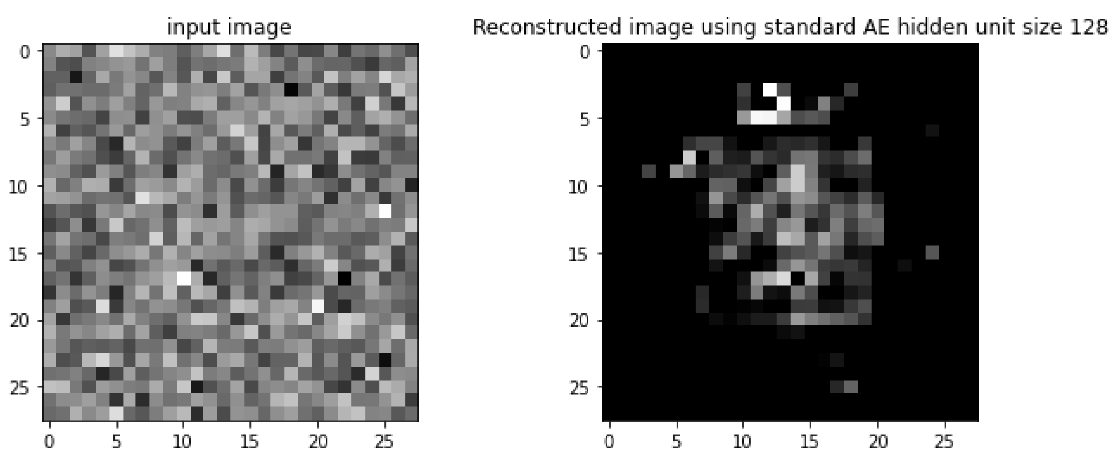
```
------------------------------------------------------------
------------------------------------------------------------
Training begins for hidden unit size  128 !
------------------------------------------------------------
Epoch Number:  1
Training loss:  0.009362838231027126
Validation loss:  0.010682893916964531
-----------------------------------------
Epoch Number:  2
Training loss:  0.009639010764658451
Validation loss:  0.009487487375736237
-----------------------------------------
Epoch Number:  3
Training loss:  0.008285951800644398
Validation loss:  0.009182061068713665
-----------------------------------------
Epoch Number:  4
Training loss:  0.010454070754349232
Validation loss:  0.009161614812910557
-----------------------------------------
Epoch Number:  5
Training loss:  0.00843499694019556
Validation loss:  0.00904966238886118
-----------------------------------------
Epoch Number:  6
Training loss:  0.009196995757520199
Validation loss:  0.008906994014978409
-----------------------------------------
Epoch Number:  7
Training loss:  0.008840101771056652
Validation loss:  0.00894408393651247
-----------------------------------------
Epoch Number:  8
Training loss:  0.0073185451328754425
Validation loss:  0.00893931183964014
-----------------------------------------
Epoch Number:  9
Training loss:  0.0070354267954826355
Validation loss:  0.008875870145857334
-----------------------------------------
Epoch Number:  10
Training loss:  0.007381132338196039
Validation loss:  0.008888255804777145
-----------------------------------------
```



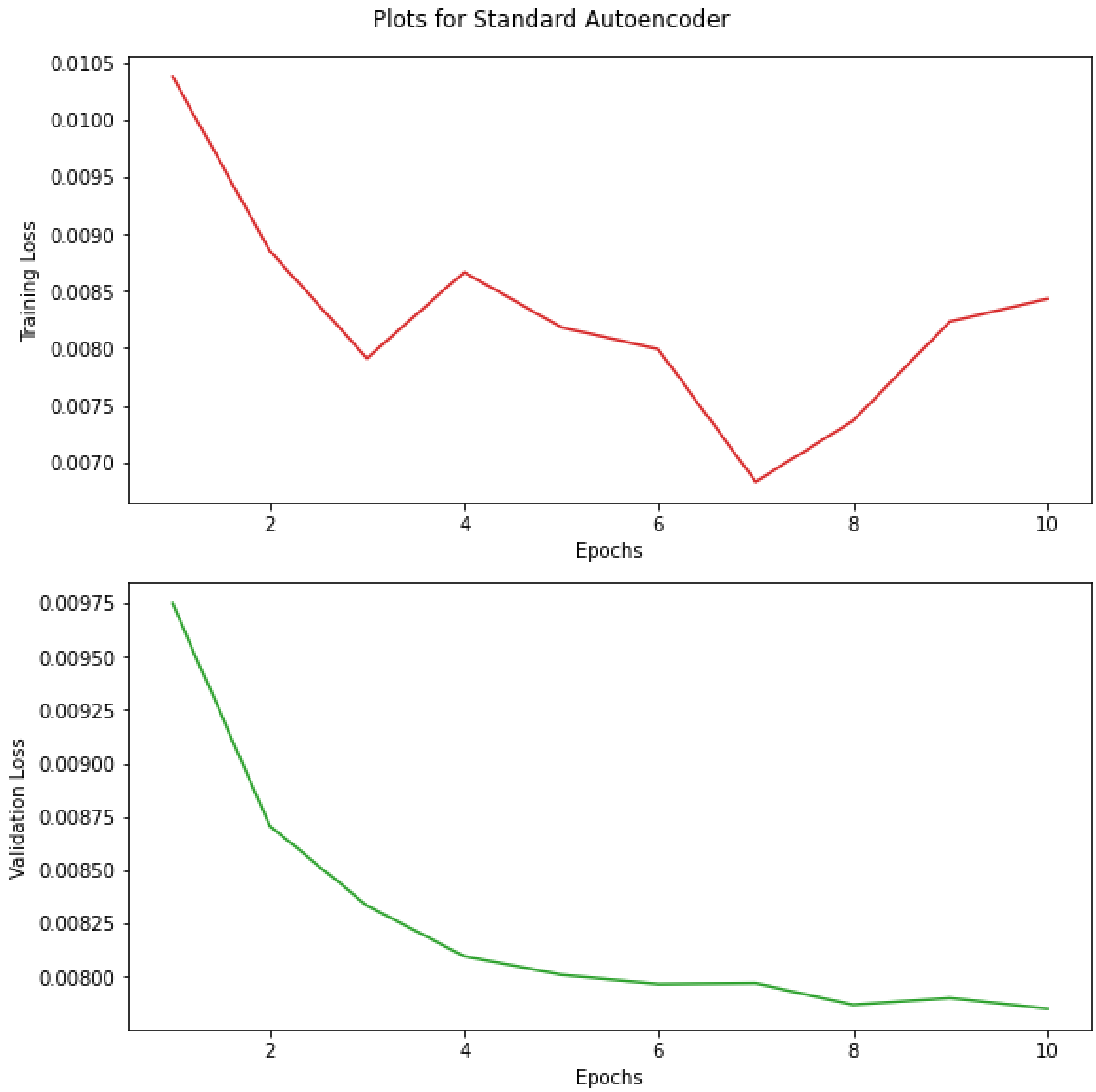

```
Training done for hidden unit size  128 !
-----------------------------------------------------------
```


input image
Reconstructed image using standard AE hidden unit size 128

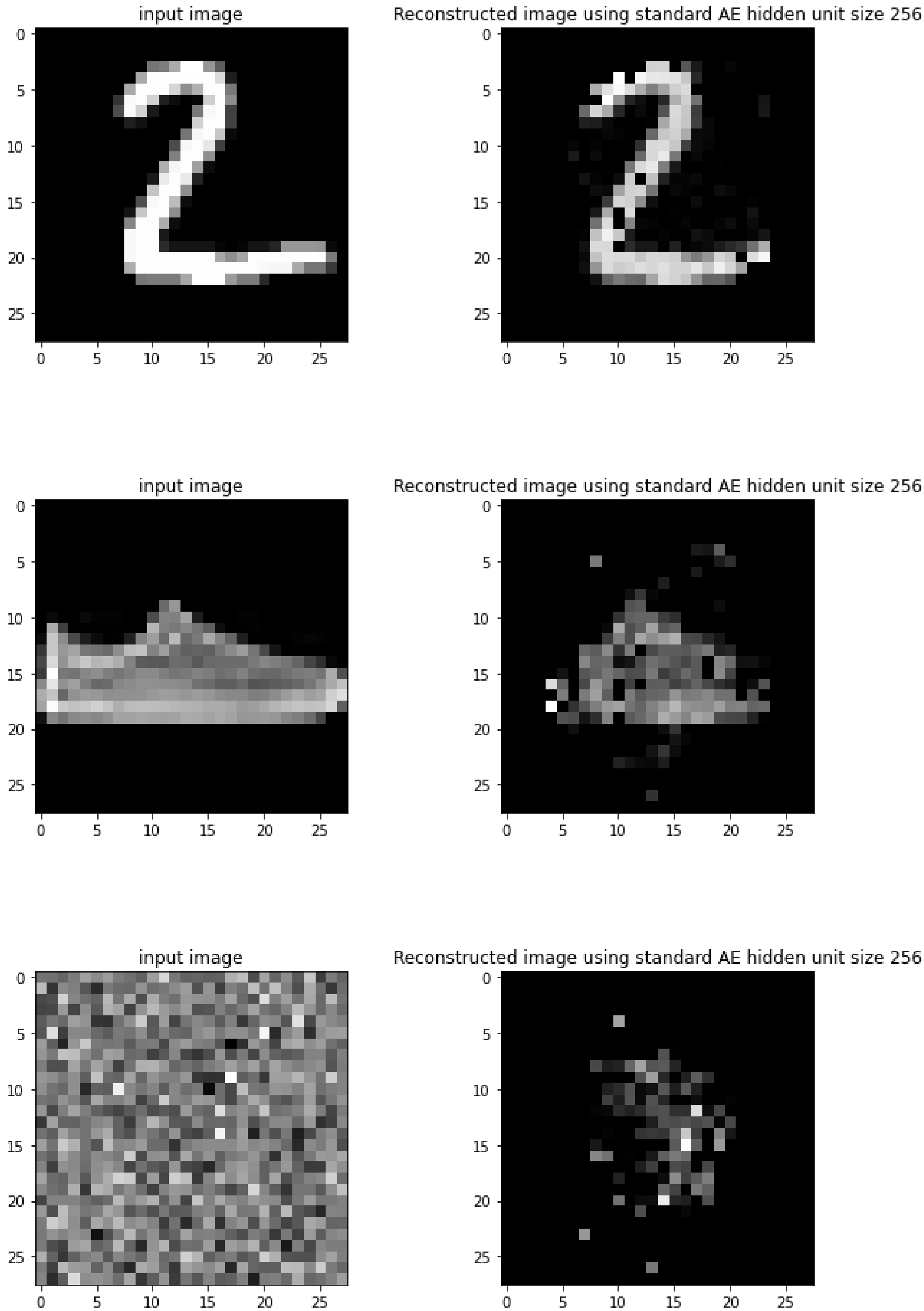

input image
Reconstructed image using standard AE hidden unit size 128


input image
Reconstructed image using standard AE hidden unit size 128

```
-----------------------------------------------------------
-----------------------------------------------------------
Training begins for hidden unit size  256 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  0.010379760526120663
Validation loss:  0.009749290533363819
-----------------------------------------
Epoch Number:  2
Training loss:  0.008854160085320473
Validation loss:  0.008706873282790184
-----------------------------------------
Epoch Number:  3
Training loss:  0.007910887710750103
Validation loss:  0.008333210833370686
-----------------------------------------
Epoch Number:  4
Training loss:  0.008666816167533398
Validation loss:  0.008095543831586838
-----------------------------------------
Epoch Number:  5
Training loss:  0.00818319246172905
Validation loss:  0.008007722906768322
-----------------------------------------
Epoch Number:  6
Training loss:  0.007989658042788506
Validation loss:  0.007964433170855045
-----------------------------------------
Epoch Number:  7
Training loss:  0.006828519515693188
Validation loss:  0.007969017140567303
-----------------------------------------
Epoch Number:  8
Training loss:  0.007364301010966301
Validation loss:  0.007867636159062386
-----------------------------------------
Epoch Number:  9
Training loss:  0.008234119042754173
Validation loss:  0.007899788208305836
-----------------------------------------
Epoch Number:  10
Training loss:  0.008432411588728428
Validation loss:  0.007849331013858318
-----------------------------------------
```




```
Training done for hidden unit size  256 !
-----------------------------------------------------------
```


input image
Reconstructed image using standard AE hidden unit size 256
input image
Reconstructed image using standard AE hidden unit size 256
input image
Reconstructed image using standard AE hidden unit size 256

----

3 Sparse Autoencoders

3.1 Defining sparse autoencoder class

[16]:
```
class overcomp_sparse_AE(nn.Module):
  def __init__(self):
    super(overcomp_sparse_AE, self).__init__()
    self.flatten = nn.Flatten()
    self.encoder = nn.Sequential(
        nn.Linear(784,1521),
        nn.ReLU())
    self.decoder =nn.Sequential(
        nn.Linear(1521,784),
        nn.ReLU())


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

3.2 Defining experimentation functions

- Average hidden layer activation `avg_hl_activations`
- Visualize activations `visualize_activations`
- Visualize learned filters `encoder_decoder_filters_plots`

[17]:
```
def avg_hl_activations(model,test_dataloader,model_name):
    model.eval()
    avg_act_val = 0
    with torch.no_grad():
        for (data,label) in test_dataloader:
            (data,label) = (data.to(device),label.to(device))
            decoded,encoded = model(data) #our prediction
            avg_act_val += float(torch.mean(encoded))
    avg_act_val /= len(test_dataloader)
    print("The average activation of "+ str(model_name)+" is",avg_act_val)

def encoder_decoder_filters_plots(model,model_name,device):
    with torch.no_grad():
        encoder_filters = model.encoder[0].weight.detach().cpu().numpy()
        decoder_filters = model.decoder[0].weight.detach().cpu().numpy()
        #plot the encoder and decoder weights as an image for 0th neuron
```

```
        plt.imshow(encoder_filters[0].reshape(28,28), cmap='gray')
        plt.colorbar()
        plt.title('Encoder Filters for '+str(0)+'th neuron of '+␣
 ↪str(model_name))
        plt.show()

        plt.imshow(decoder_filters[:,0].reshape(28,28), cmap='gray')
        plt.colorbar()
        plt.title('Decoder Filters for '+str(0)+'th neuron of '+␣
 ↪str(model_name))
        plt.show()

def visualize_activations(model,test_dataloader,model_name,device,hidden_layer):
 ↪ #visualize the activations
    for i,ind in enumerate(idx_test):
        test_image = test_dataloader.dataset.data[ind].clone()
        test_label = test_dataloader.dataset.targets[ind].clone()
        with torch.no_grad():
            test_image = test_image.reshape(1,1,28,28).to(device=device).float()
            decoded,encoded = model.forward(test_image)
            encoded = encoded.detach().cpu().numpy()
            plt.imshow(encoded.reshape(int(np.sqrt(hidden_layer)),int(np.
 ↪sqrt(hidden_layer))), cmap='gray')
            str_title = "Activation for digit "+str(test_label.item())
            plt.title(str_title)
            plt.show()
```

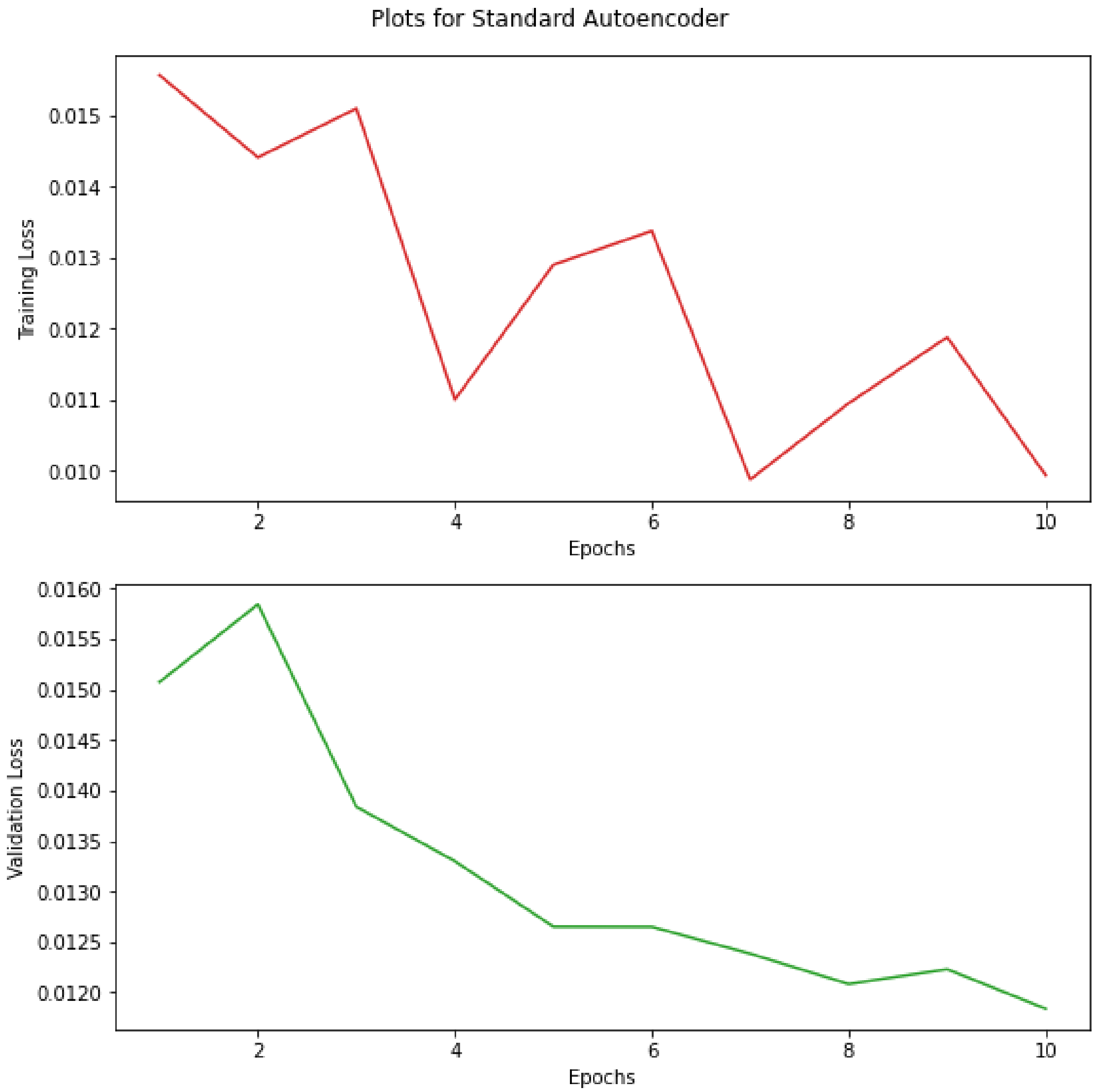
[18]:
```
sparsity_values = [0.0001, 0.001, 0.005, 0.1]

for lam in sparsity_values:
  model_over_sparseAE = overcomp_sparse_AE().to(device)
  optimizer = torch.optim.Adam(model_over_sparseAE.parameters(), learning_rate)

  train_batch_loss = []
  train_epoch_loss = []
  val_loss = []
  print('Training begins for regularization lambda value ', lam, '!')
  print('------------------------------------------------------------')
  for epoch in range(epochs):
    for batch_idx, (data, target) in enumerate(train_loader):
      data = data.to(device=device)
      target = target.to(device=device)

      model_over_sparseAE.train()
      decoded, encoded = model_over_sparseAE(data)
      loss = criterion(decoded, torch.flatten(data,1))
      loss+= lam*torch.linalg.norm(encoded, 1)
```

```
    train_batch_loss.append(loss.item())

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  train_epoch_loss.append(train_batch_loss[-1])

  model_over_sparseAE.eval()
  with torch.no_grad():
    for (data, target) in test_loader:
      data = data.to(device)
      target = target.to(device)

      decoded, encoded = model_over_sparseAE(data)
      val_iter_loss = criterion(decoded, torch.flatten(data,1))
      val_iter_loss += lam*torch.linalg.norm(encoded, 1)

  val_loss.append(val_iter_loss.item())
  print('Epoch Number: ',epoch+1)
  print('Training loss: ', train_epoch_loss[-1])
  print('Validation loss: ', val_loss[-1])
  print('-----------------------------------------')

fig, axs = plt.subplots(2, constrained_layout= True, figsize=(8,8))
fig.suptitle('Plots for Standard Autoencoder')
axs[0].set(ylabel='Training Loss', xlabel='Epochs')
axs[0].plot(range(1,epochs+1),train_epoch_loss, 'tab:red')
axs[1].set(ylabel='Validation Loss', xlabel='Epochs')
axs[1].plot(range(1,epochs+1),val_loss, 'tab:green')
plt.show()
print('Training done for regularization lambda value ', lam, '!')
print('------------------------------------------------------------')
image = test_loader.dataset.data[1, :, :].clone() #take one image from test set
with torch.no_grad():
  image = image.view(-1,28,28).to(device=device).float()
  decoded, encoded = model_over_sparseAE.forward(image)
  decoded = decoded.detach().cpu().numpy()
  image = image.reshape(28,28).detach().cpu().numpy()
  plt.subplot(1,2,1)
  plt.imshow(image , cmap='gray')
  plt.title('input image')
  plt.subplots_adjust(right=1.5)
  plt.subplot(1,2,2)
  plt.imshow(decoded.reshape(28,28),cmap ='gray')
  plt.title("Reconstructed image using Sparse AE with lambda value "+str(lam))
```

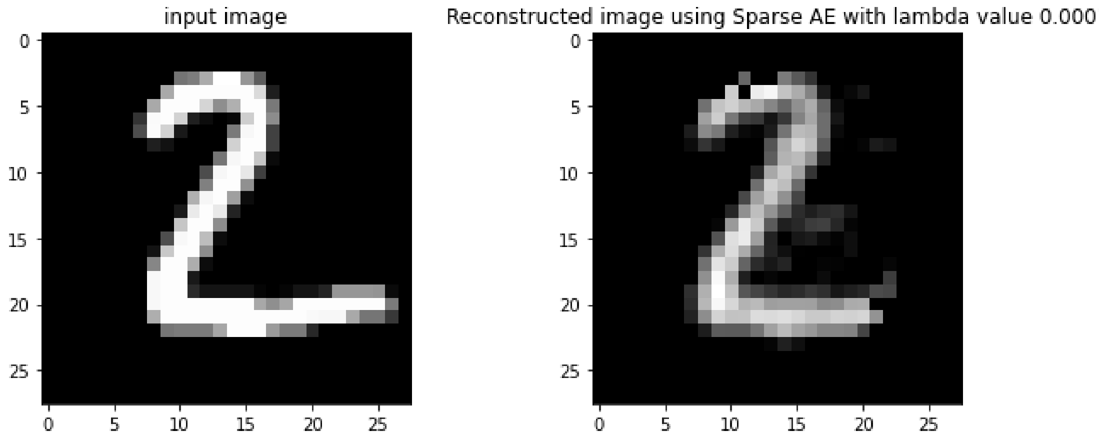
```
    plt.show()
    avg_hl_activations(model_over_sparseAE,test_loader,"Sparse AE with lambda␣
↪value "+str(lam))
    encoder_decoder_filters_plots(model_over_sparseAE,"Sparse AE with lambda␣
↪value"+str(lam),device)
    visualize_activations(model_over_sparseAE,test_loader,"Sparse AE with␣
↪lambda value"+str(lam),device,1521)
  print('-----------------------------------------------------------')
  print('-----------------------------------------------------------')
```

```
Training begins for regularization lambda value  0.0001 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  0.015572241507470608
Validation loss:  0.015071733854711056
-----------------------------------------
Epoch Number:  2
Training loss:  0.014412588439881802
Validation loss:  0.015841398388147354
-----------------------------------------
Epoch Number:  3
Training loss:  0.015103071928024292
Validation loss:  0.01383668091148138
-----------------------------------------
Epoch Number:  4
Training loss:  0.01100159715861082
Validation loss:  0.013298295438289642
-----------------------------------------
Epoch Number:  5
Training loss:  0.012901985086500645
Validation loss:  0.012650098651647568
-----------------------------------------
Epoch Number:  6
Training loss:  0.013376357965171337
Validation loss:  0.012648377567529678
-----------------------------------------
Epoch Number:  7
Training loss:  0.009877783246338367
Validation loss:  0.012382451444864273
-----------------------------------------
Epoch Number:  8
Training loss:  0.010951398871839046
Validation loss:  0.012081769295036793
-----------------------------------------
Epoch Number:  9
Training loss:  0.011881495825946331
Validation loss:  0.012228610925376415
```

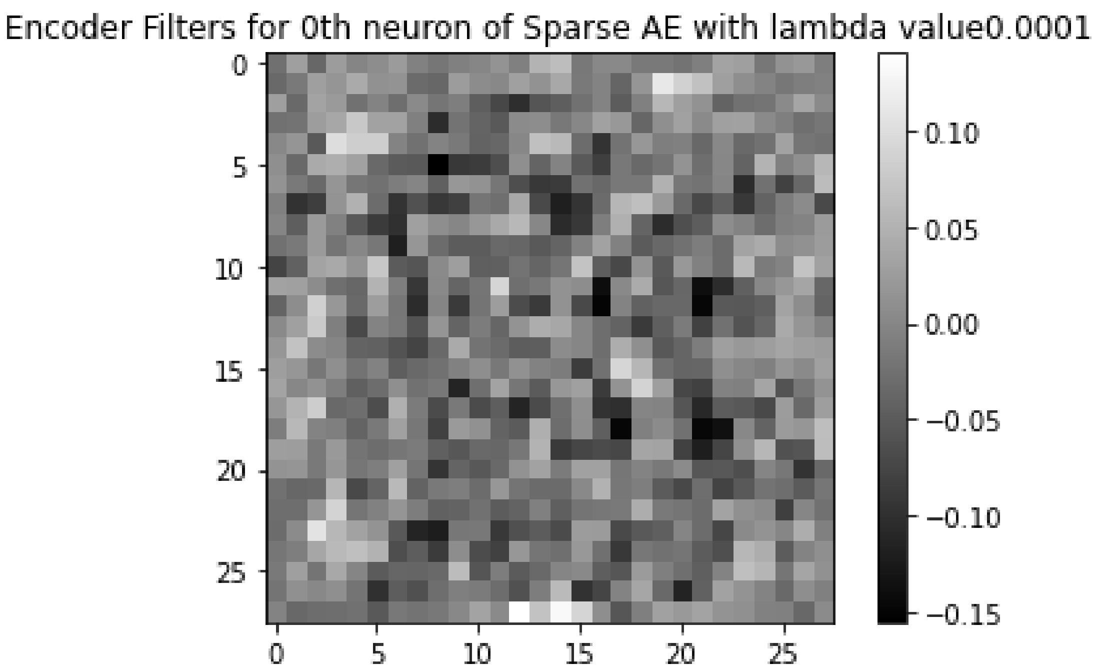
```
-----------------------------------------
Epoch Number:  10
Training loss:  0.009938022121787071
Validation loss:  0.011837400496006012
-----------------------------------------
```

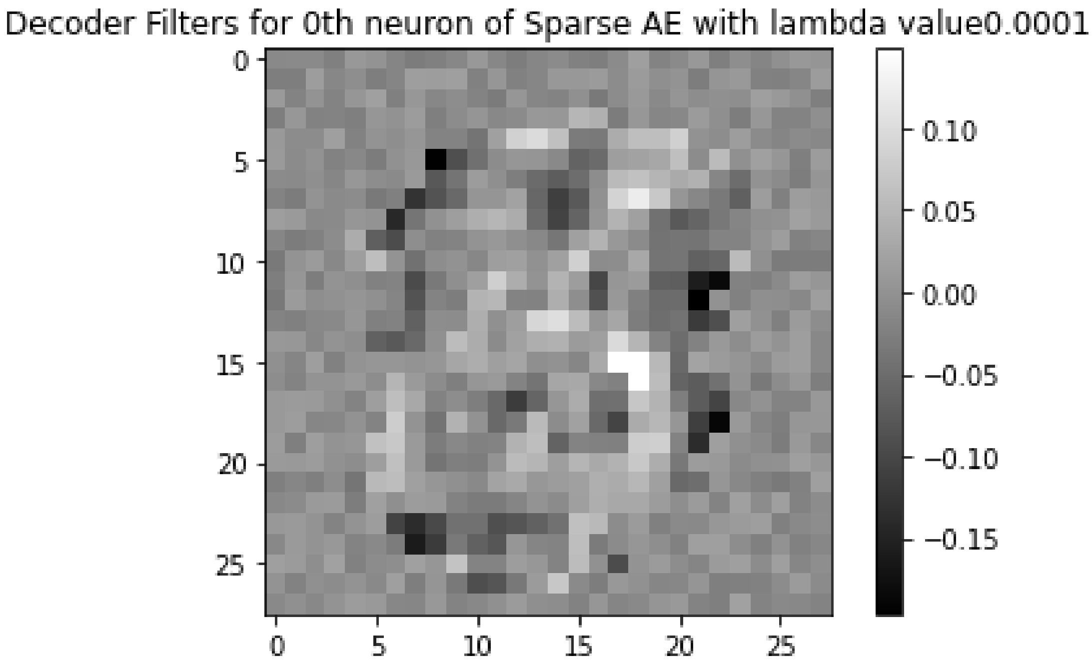


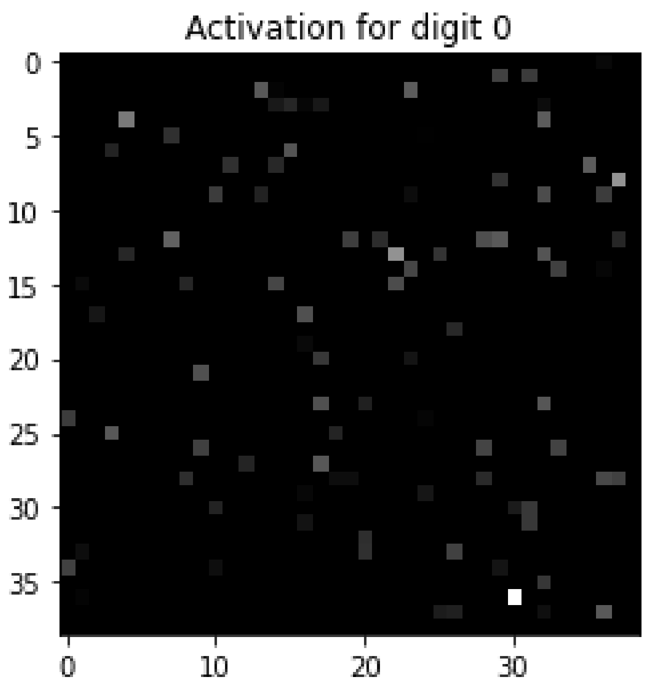

```
Training done for regularization lambda value  0.0001 !
-----------------------------------------------------------
```

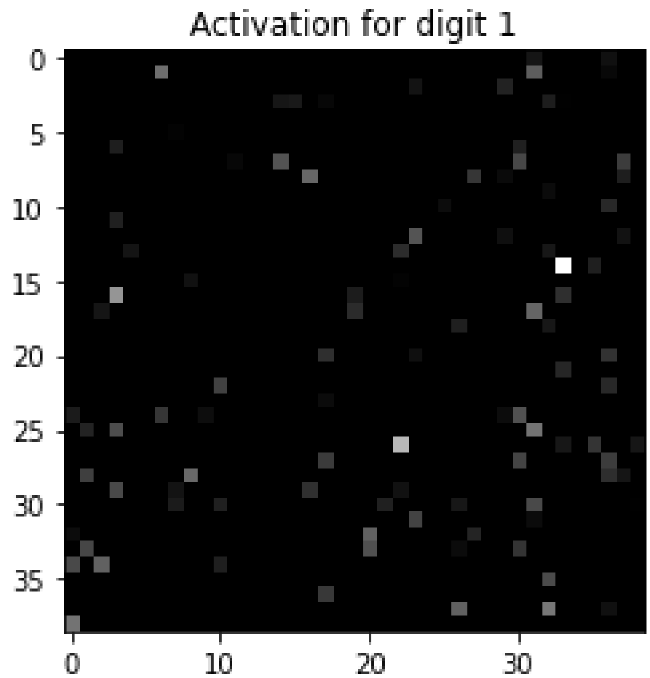


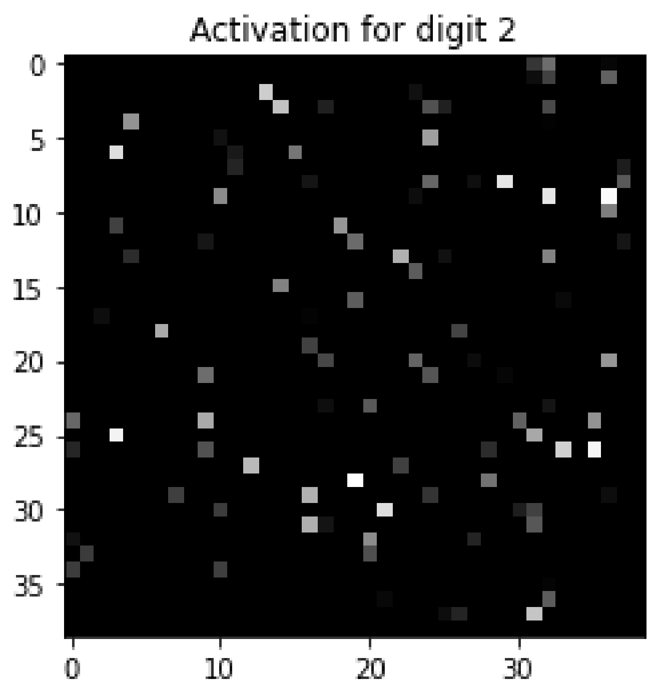
The average activation of Sparse AE with lambda value 0.0001 is
0.024317094676528768

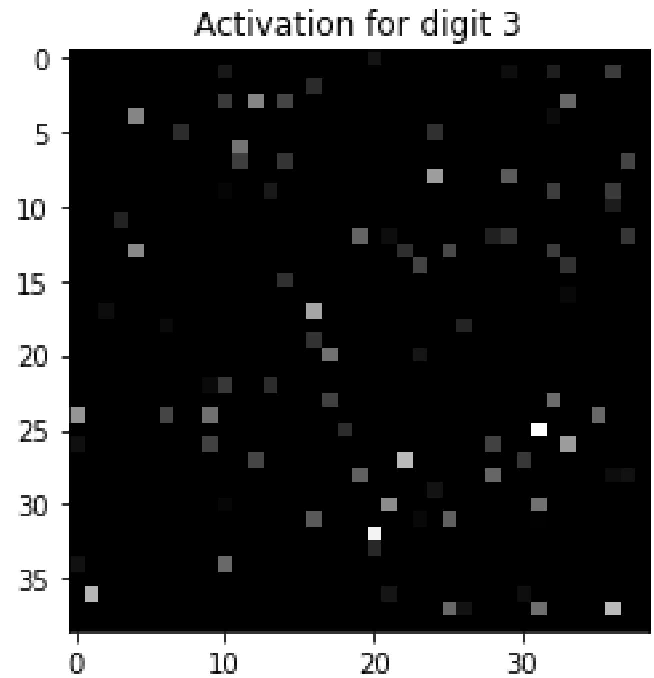


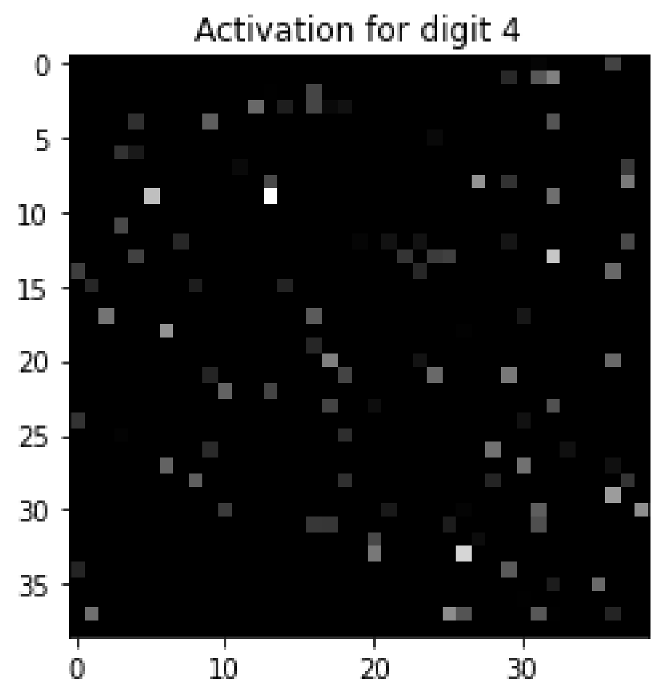

Decoder Filters for 0th neuron of Sparse AE with lambda value0.0001
0
5
10
15
20
25
0.10
0.05
0.00
−0.05
−0.10
−0.15

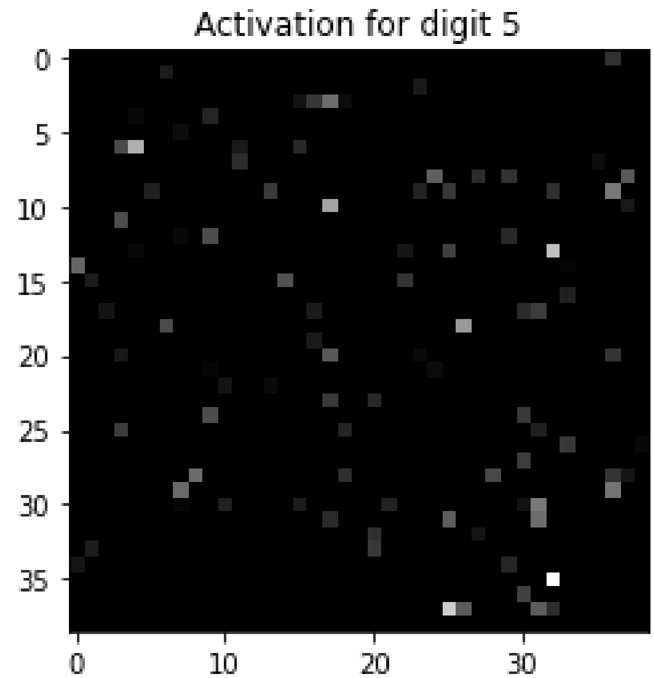

Activation for digit 0
0
5
10
15
20
25
30
35
0
10
20
30

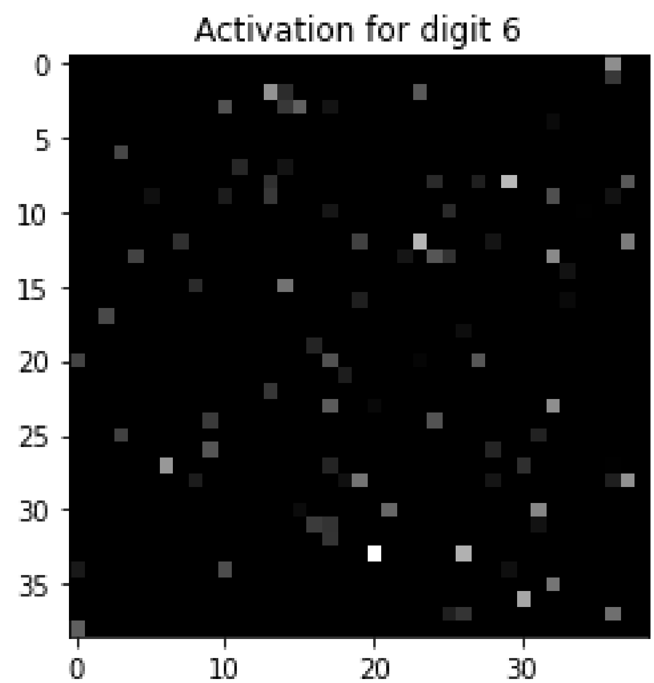

Activation for digit 1
0
5
10
15
20
25
30
35
0
10
20
30

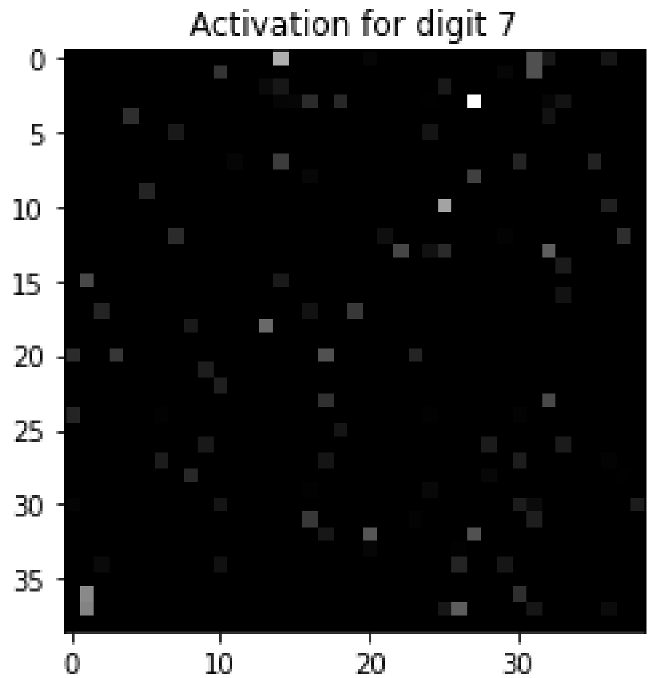

Activation for digit 2
0
5
10
15
20
25
30
35
0
10
20
30

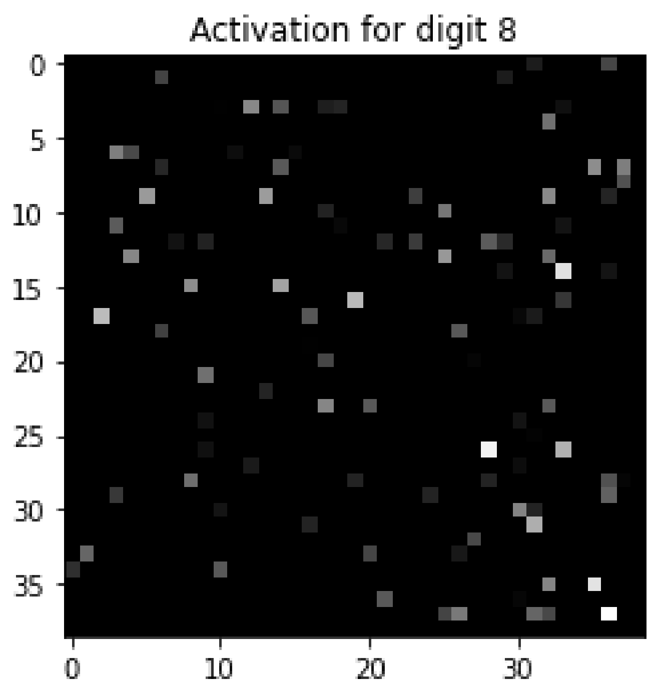

Activation for digit 3
0
5
10
15
20
25
30
35
0
10
20
30

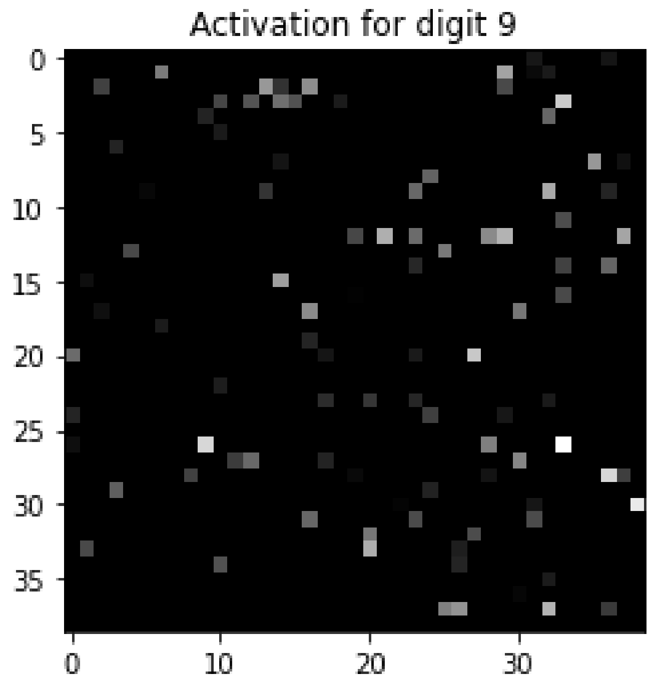

Activation for digit 4
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 5
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 6
0
5
10
15
20
25
30
35
0
10
20
30

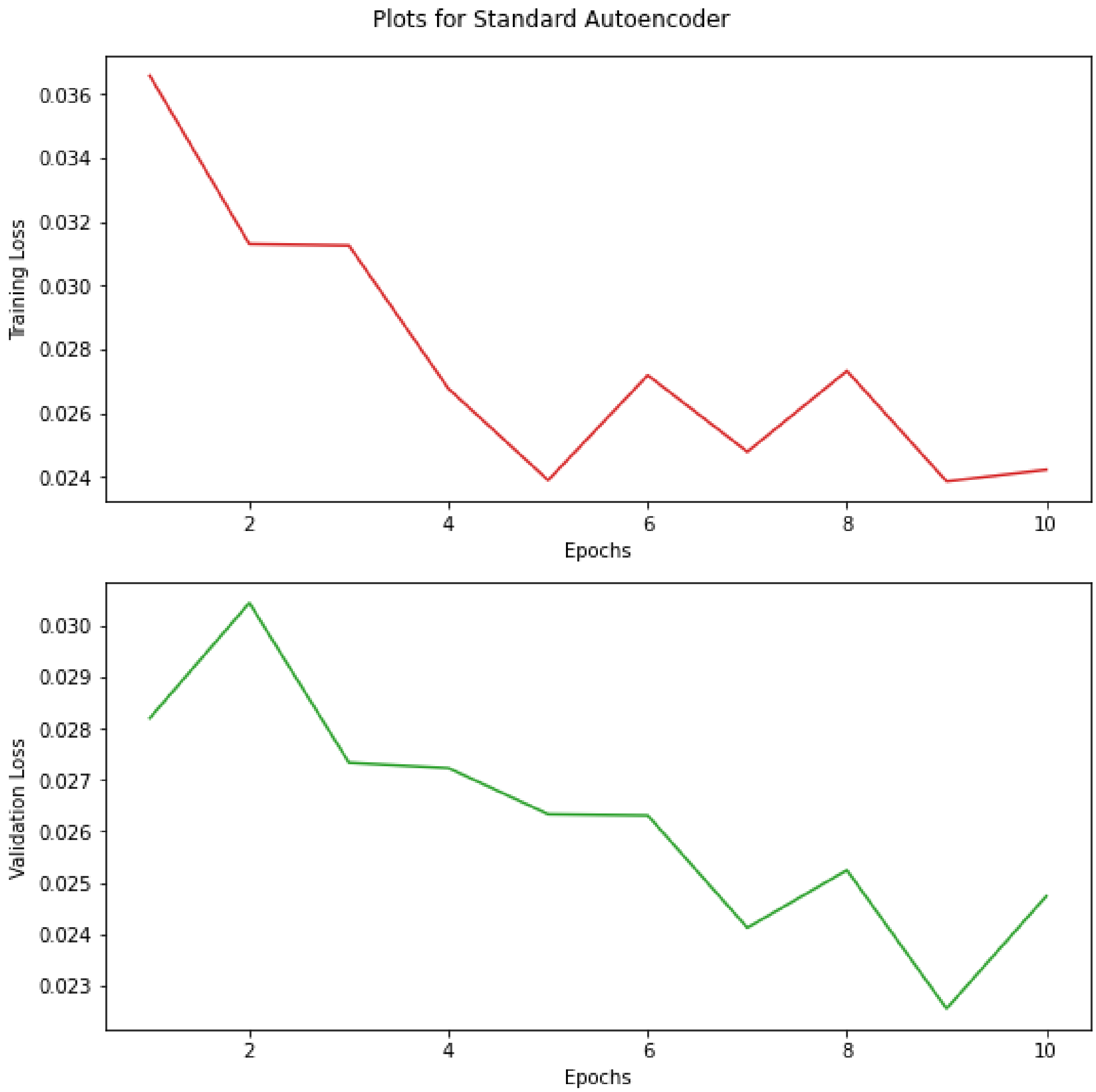

Activation for digit 7
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 8
0
5
10
15
20
25
30
35
0
10
20
30



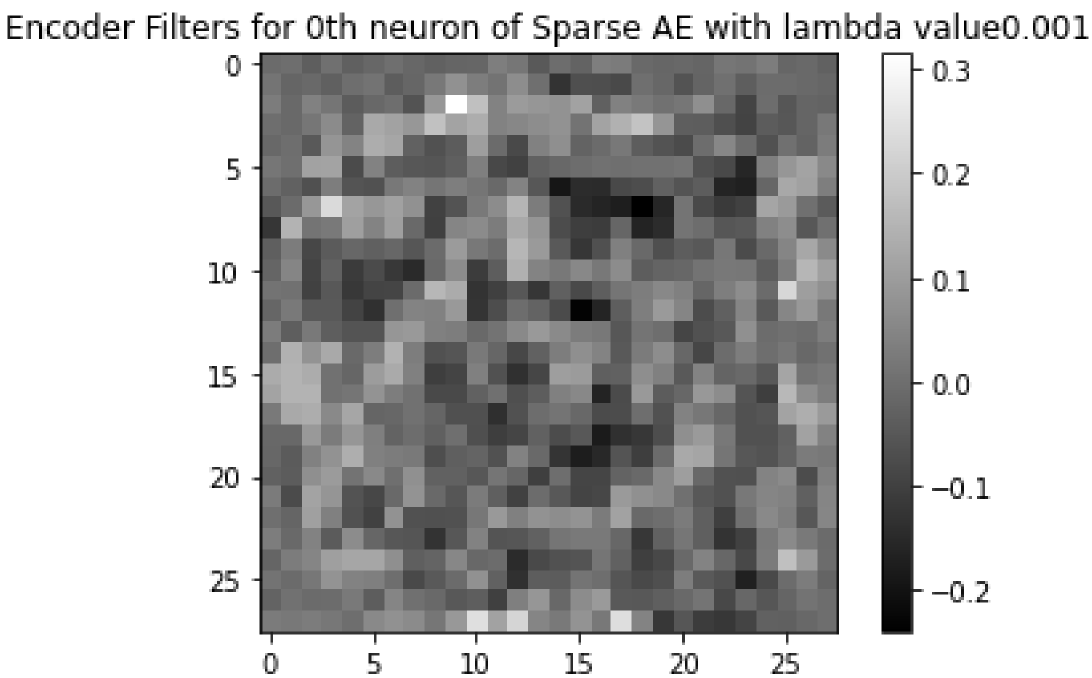

```
------------------------------------------------------------
------------------------------------------------------------
Training begins for regularization lambda value  0.001 !
------------------------------------------------------------
Epoch Number:  1
Training loss:  0.03657766059041023
Validation loss:  0.02819947898387909
-----------------------------------------
Epoch Number:  2
Training loss:  0.03130034729838371
Validation loss:  0.030439140275120735
-----------------------------------------
Epoch Number:  3
Training loss:  0.03125472366809845
Validation loss:  0.027336187660694122
-----------------------------------------
Epoch Number:  4
Training loss:  0.026766687631607056
Validation loss:  0.02722999081015587
-----------------------------------------
Epoch Number:  5
Training loss:  0.023902306333184242
Validation loss:  0.0263354554772377
-----------------------------------------
Epoch Number:  6
```

```
Training loss:  0.02719135954976082
Validation loss:  0.026311347261071205
-----------------------------------------
Epoch Number:  7
Training loss:  0.02479413151741028
Validation loss:  0.024127982556819916
-----------------------------------------
Epoch Number:  8
Training loss:  0.027327170595526695
Validation loss:  0.02524668537080288
-----------------------------------------
Epoch Number:  9
Training loss:  0.023866549134254456
Validation loss:  0.022555096074938774
-----------------------------------------
Epoch Number:  10
Training loss:  0.024228475987911224
Validation loss:  0.024739695712924004
-----------------------------------------
```

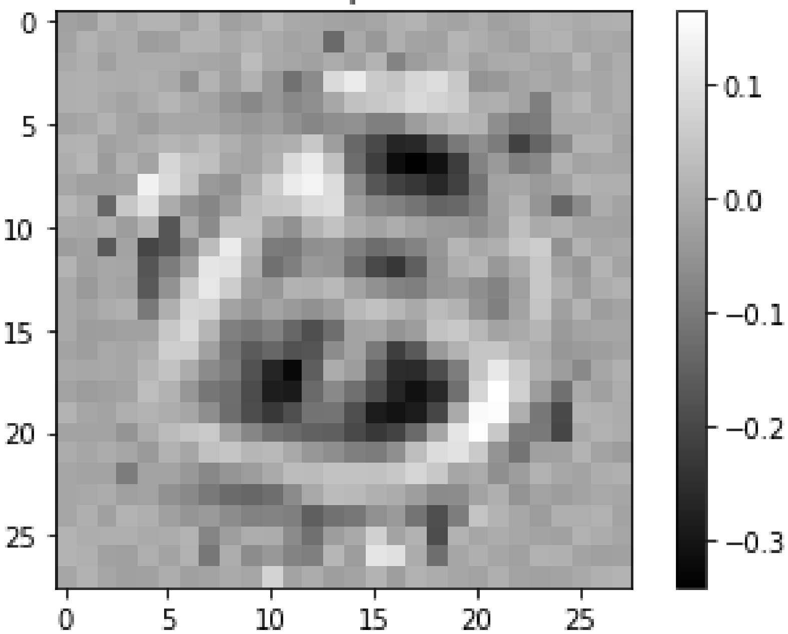



```
Training done for regularization lambda value  0.001 !
-----------------------------------------------------------
```

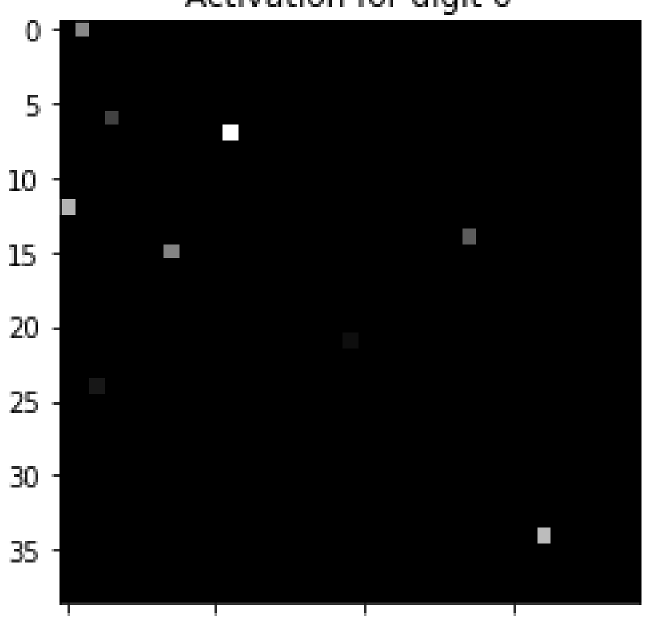

```
The average activation of Sparse AE with lambda value 0.001 is
0.005339490537454558
```

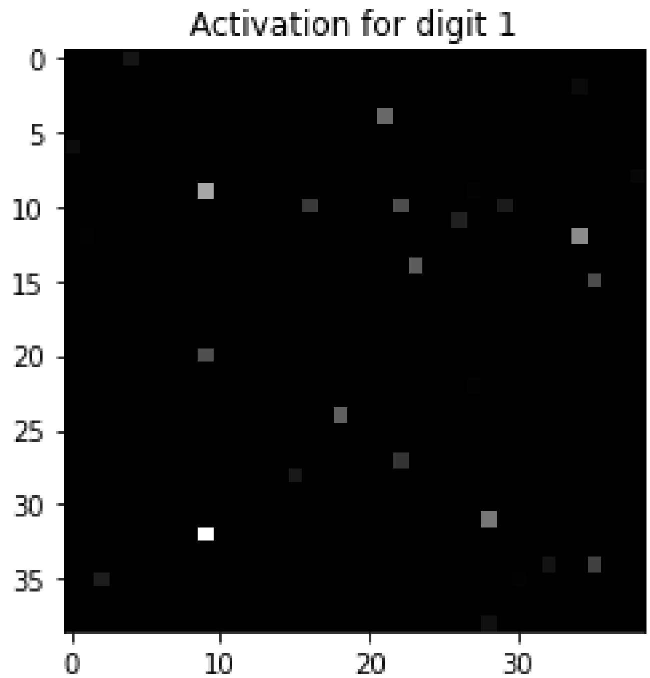


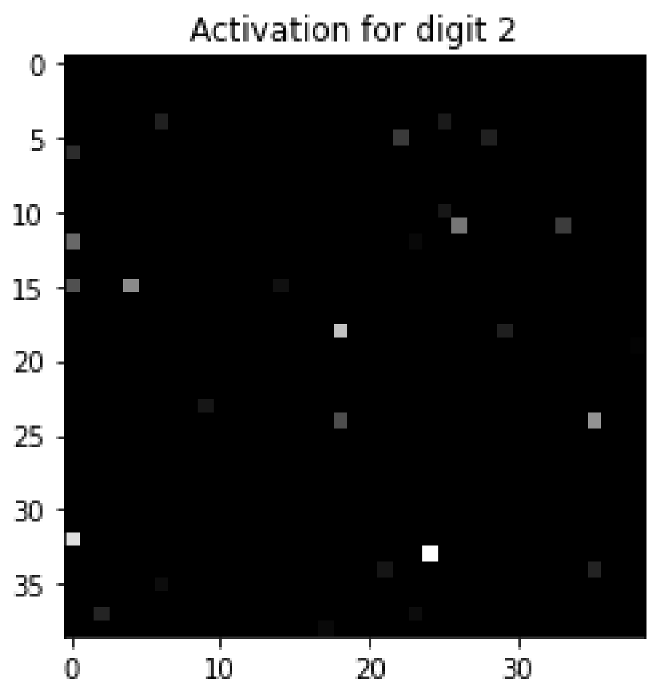
Decoder Filters for 0th neuron of Sparse AE with lambda value0.001

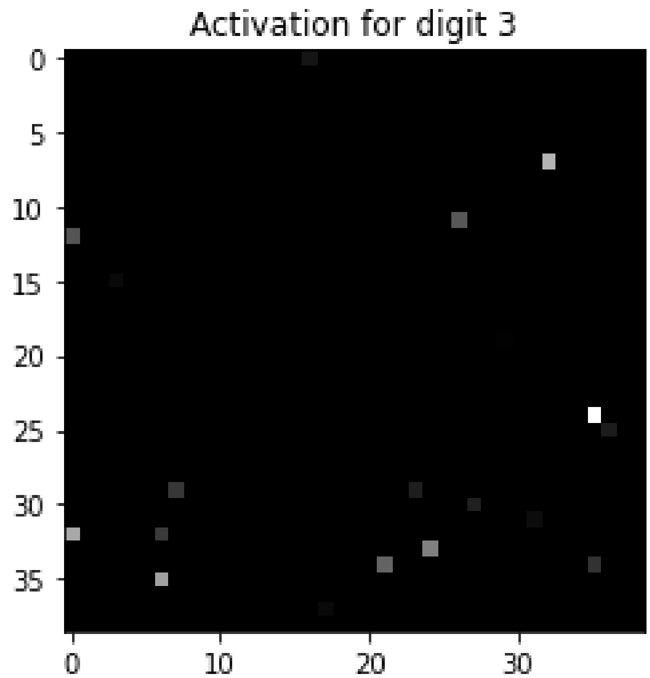


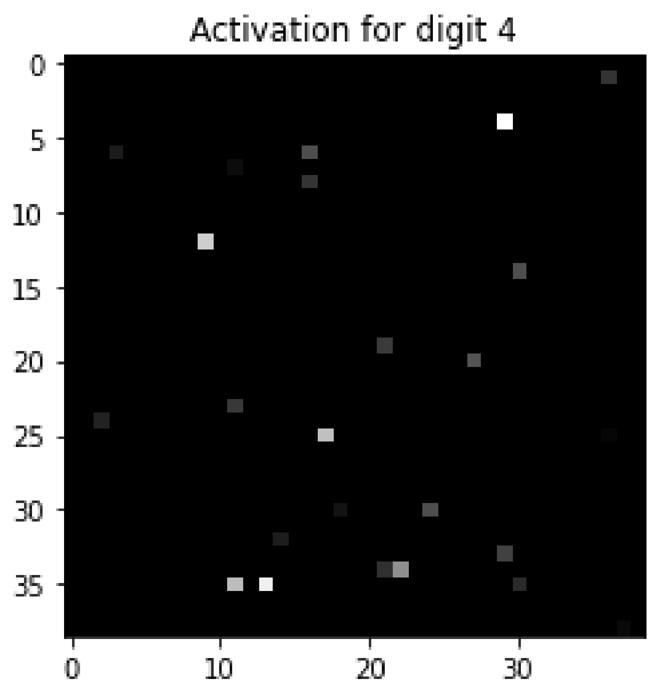
Activation for digit 0

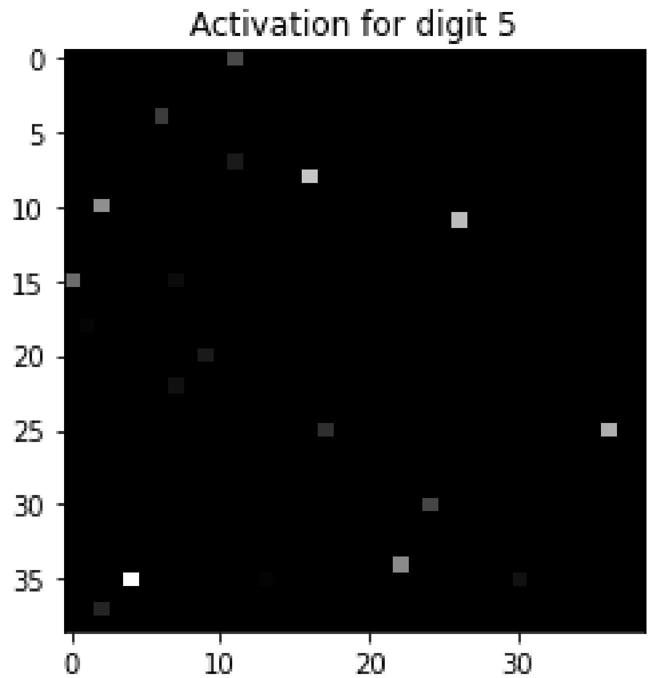



Activation for digit 1
0
5
10
15
20
25
30
35
0
10
20
30

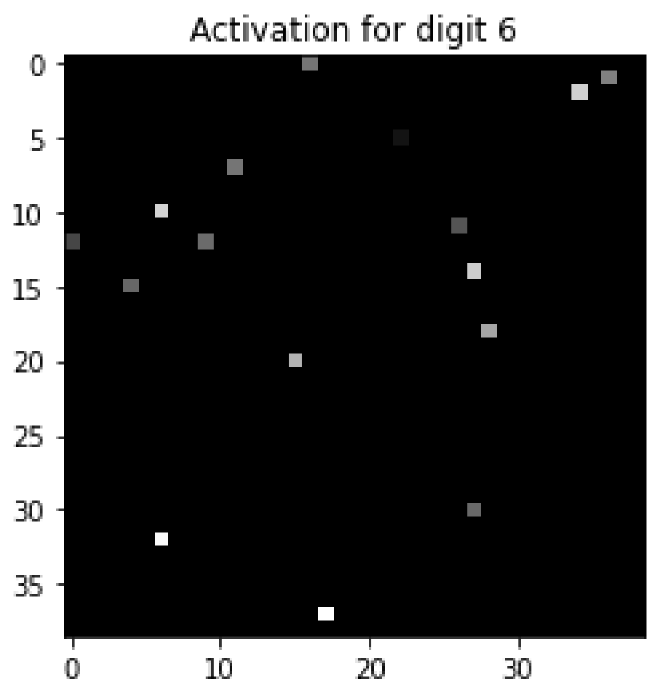

Activation for digit 2
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 3
0
5
10
15
20
25
30
35
0
10
20
30

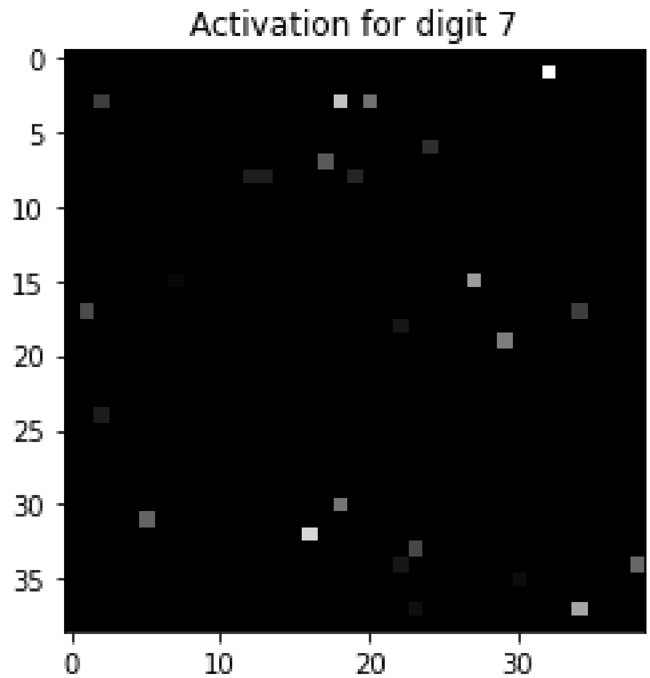

Activation for digit 4
0
5
10
15
20
25
30
35
0
10
20
30

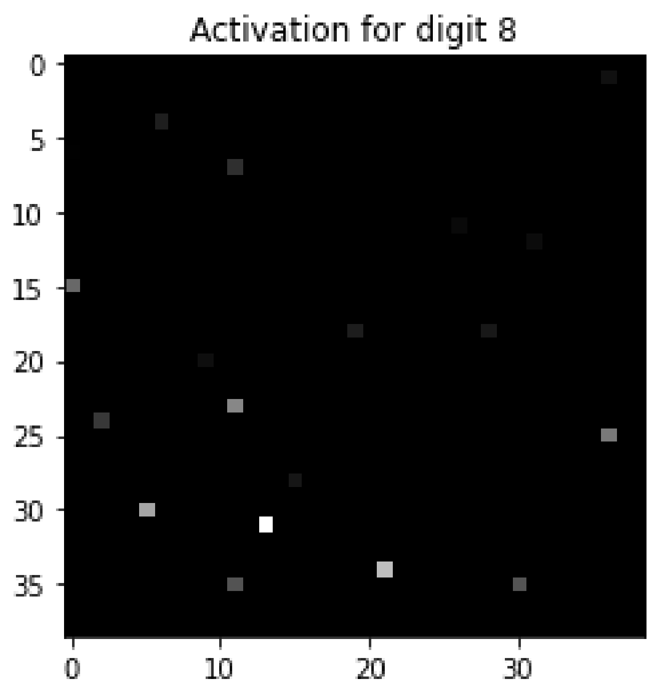


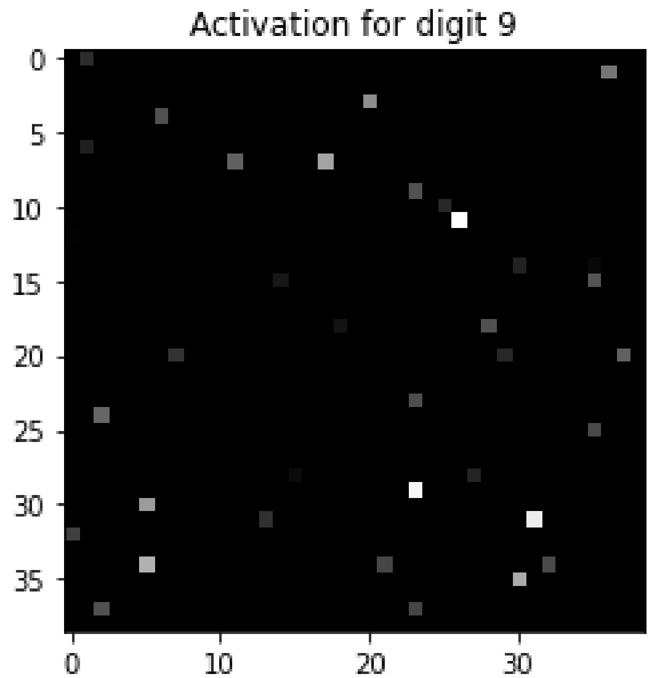
Activation for digit 5
0
5
10
15
20
25
30
35
0
10
20
30



Activation for digit 6
0
5
10
15
20
25
30
35
0
10
20
30

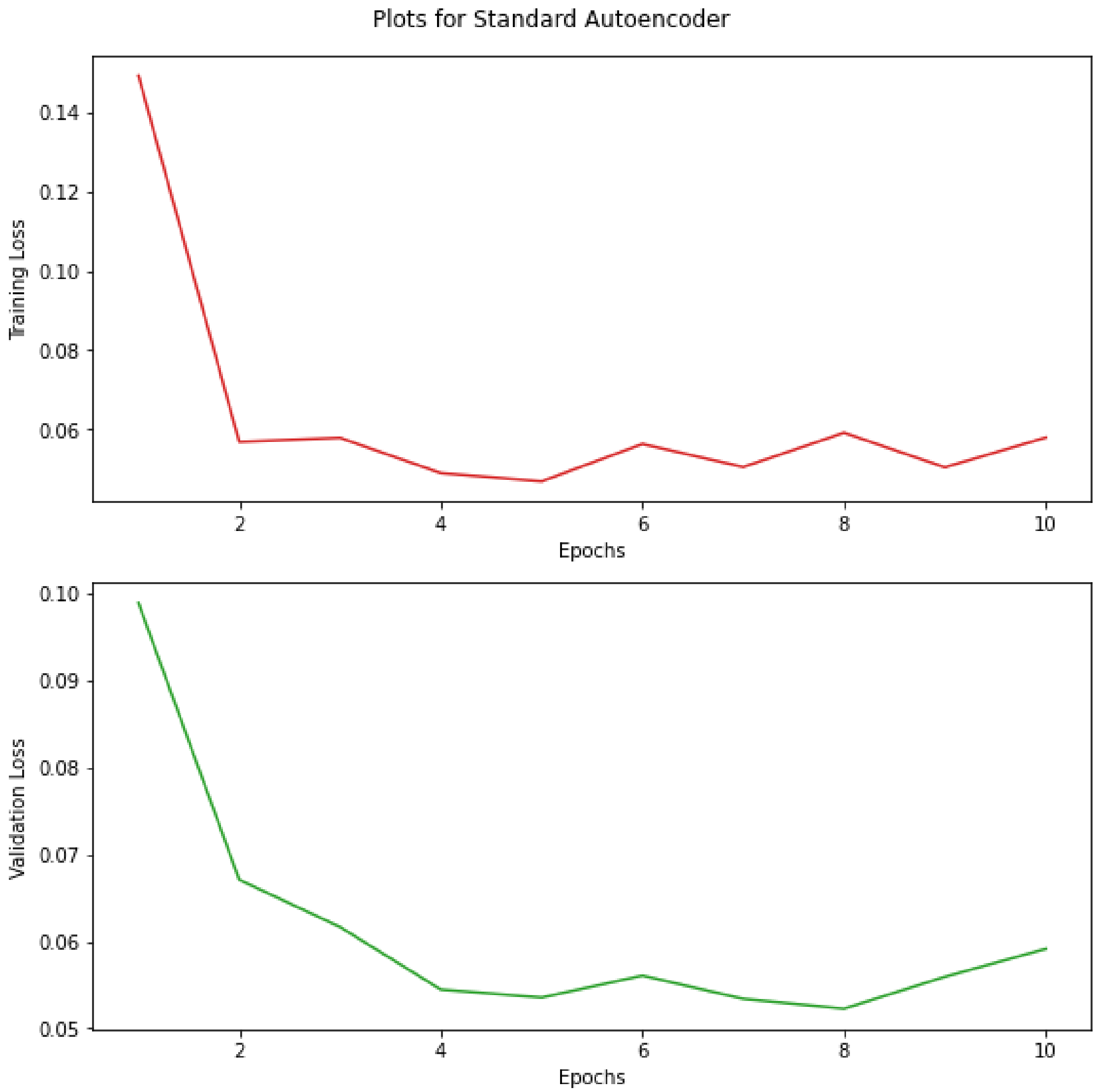

Activation for digit 7
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 8
0
5
10
15
20
25
30
35
0
10
20
30

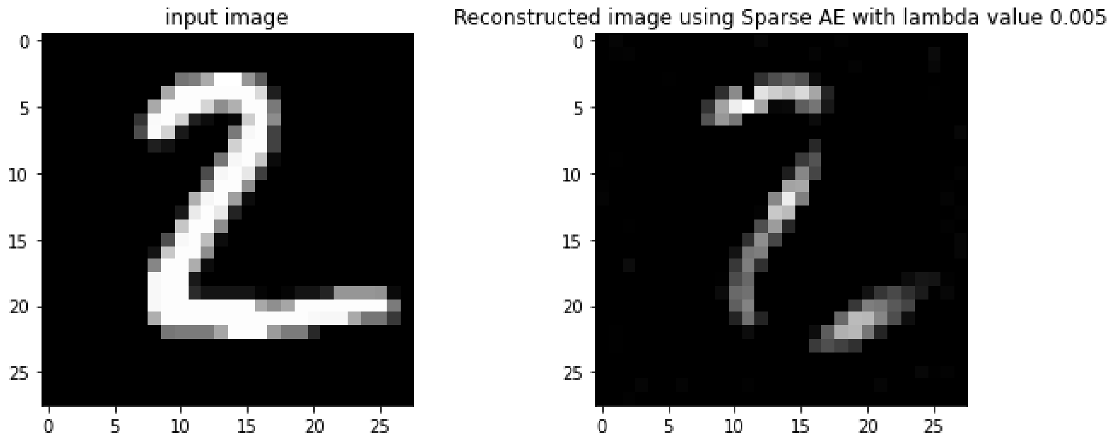



```
------------------------------------------------------------
------------------------------------------------------------
Training begins for regularization lambda value  0.005 !
------------------------------------------------------------
Epoch Number:  1
Training loss:  0.14927056431770325
Validation loss:  0.09885752201080322
-----------------------------------------
Epoch Number:  2
Training loss:  0.05682261288166046
Validation loss:  0.06706684827804565
-----------------------------------------
Epoch Number:  3
Training loss:  0.05783994868397713
Validation loss:  0.0616452656686306
-----------------------------------------
Epoch Number:  4
Training loss:  0.04894670844078064
Validation loss:  0.05445782095193863
-----------------------------------------
Epoch Number:  5
Training loss:  0.0468580424785614
Validation loss:  0.053544532507658005
-----------------------------------------
Epoch Number:  6
```

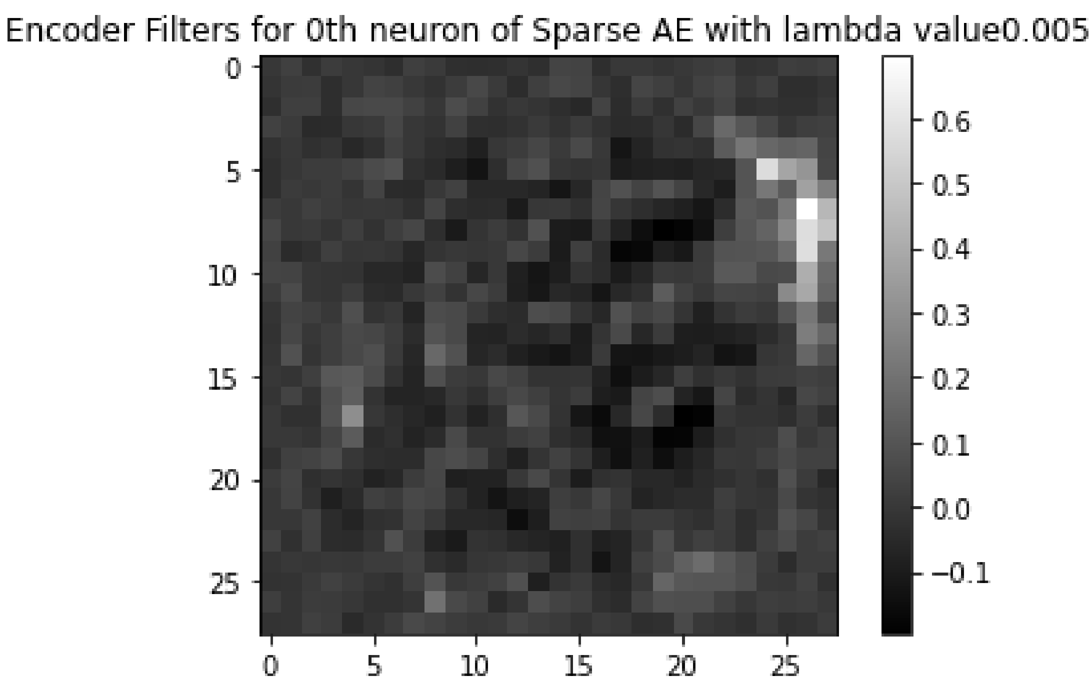
```
Training loss:  0.05629307766895294
Validation loss:  0.056057292968034744
-----------------------------------------
Epoch Number:  7
Training loss:  0.05046890303492546
Validation loss:  0.053395211696624756
-----------------------------------------
Epoch Number:  8
Training loss:  0.05911600589752197
Validation loss:  0.05226348713040352
-----------------------------------------
Epoch Number:  9
Training loss:  0.050400275737047195
Validation loss:  0.05592313036322594
-----------------------------------------
Epoch Number:  10
Training loss:  0.05790144205093384
Validation loss:  0.05912882089614868
-----------------------------------------
```

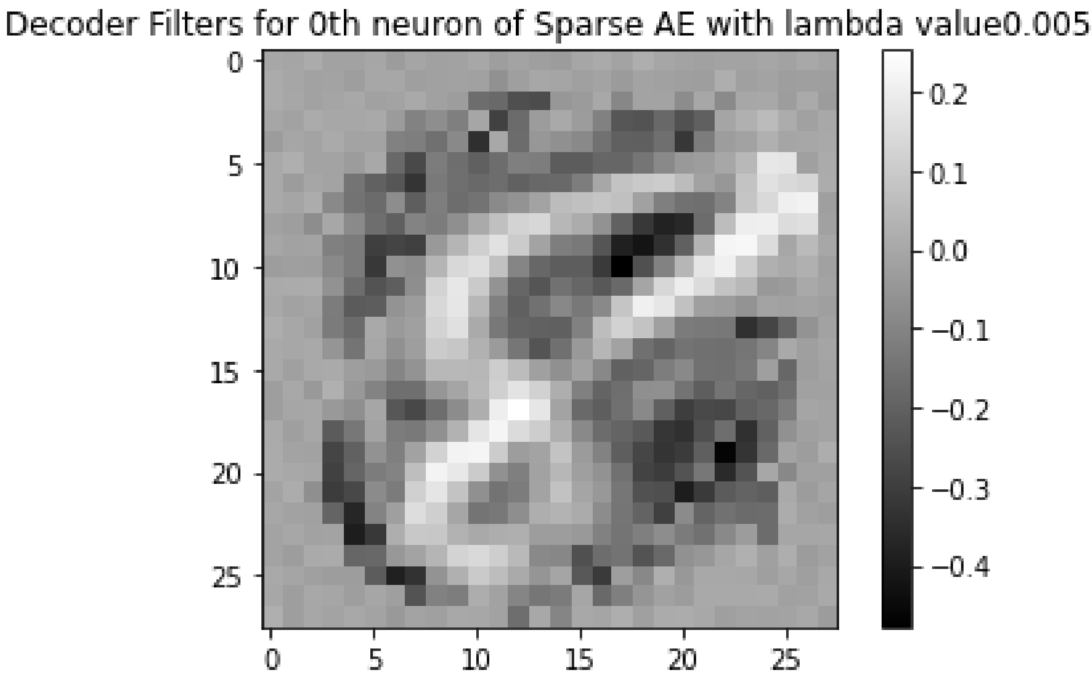


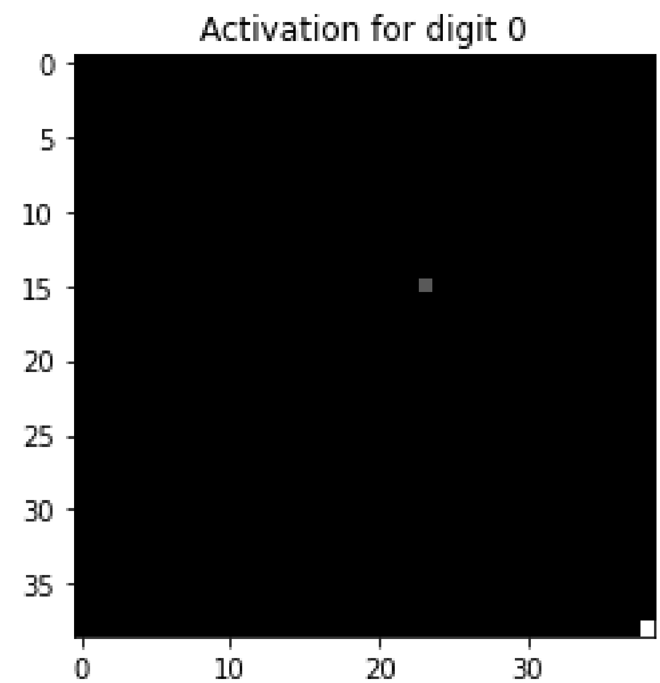

```
Training done for regularization lambda value  0.005 !
-----------------------------------------------------------
```

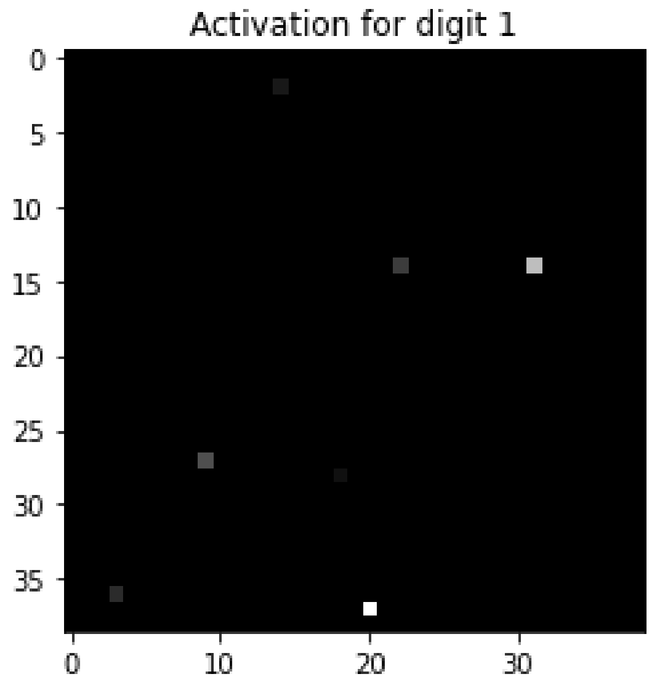


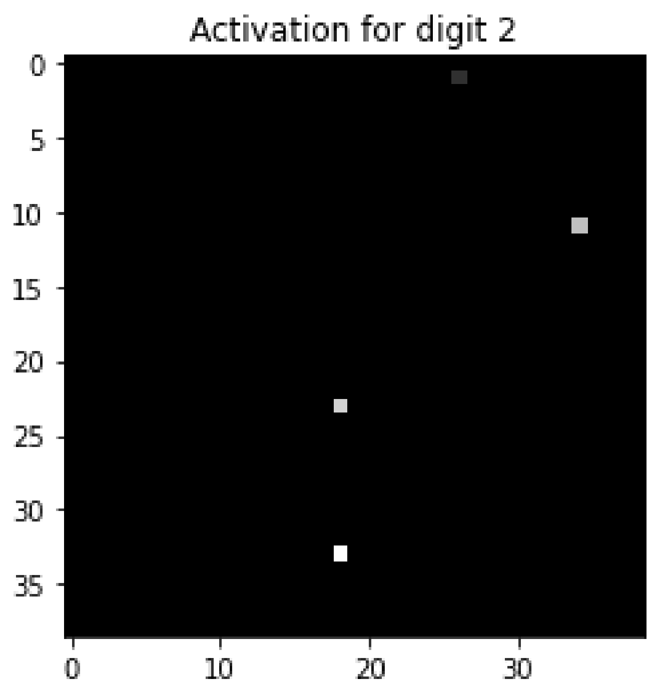

```
The average activation of Sparse AE with lambda value 0.005 is
0.0010910846091284874
```

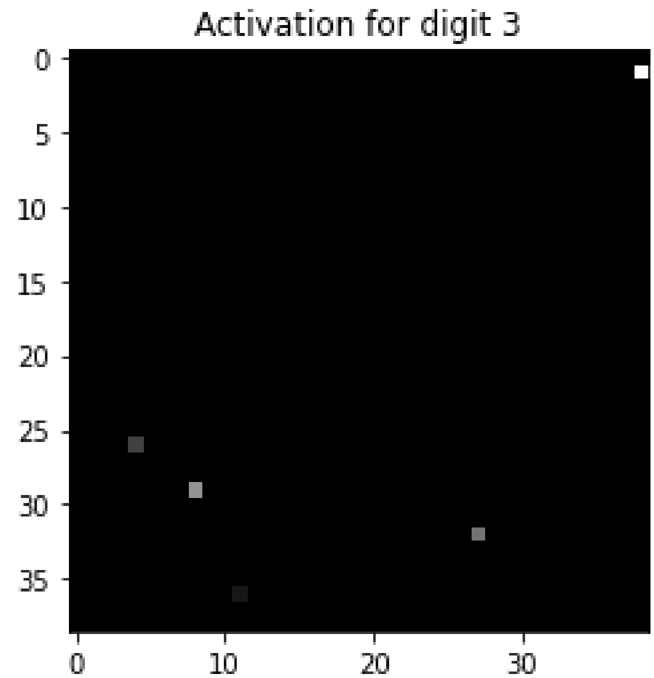


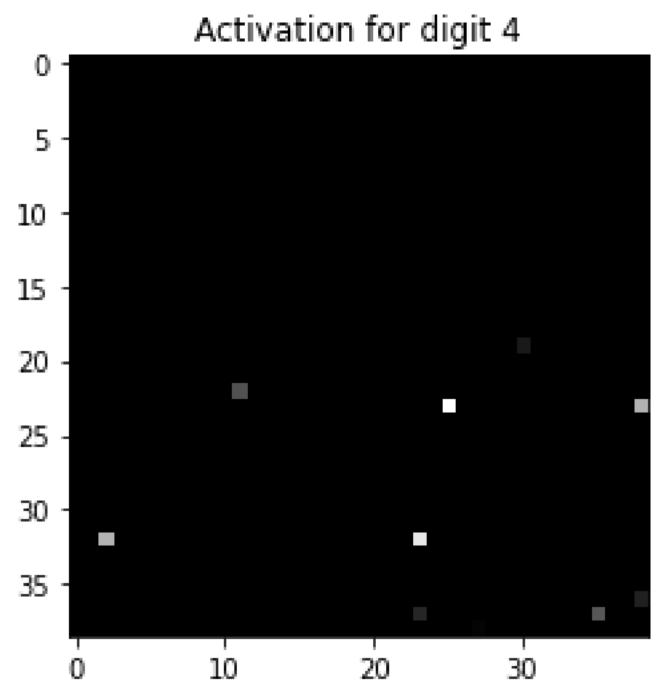

Decoder Filters for 0th neuron of Sparse AE with lambda value0.005
0
5
10
15
20
25
0
5
10
15
20
25
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4

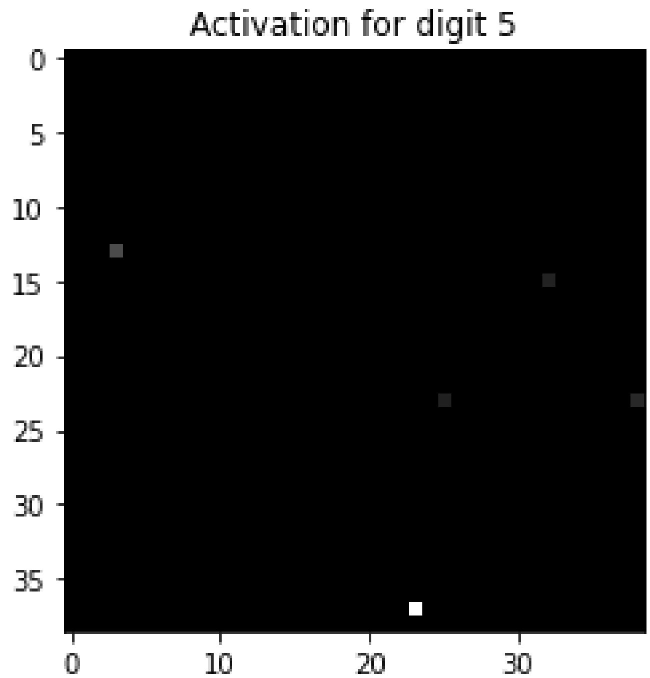

Activation for digit 0
0
5
10
15
20
25
30
35
0
10
20
30

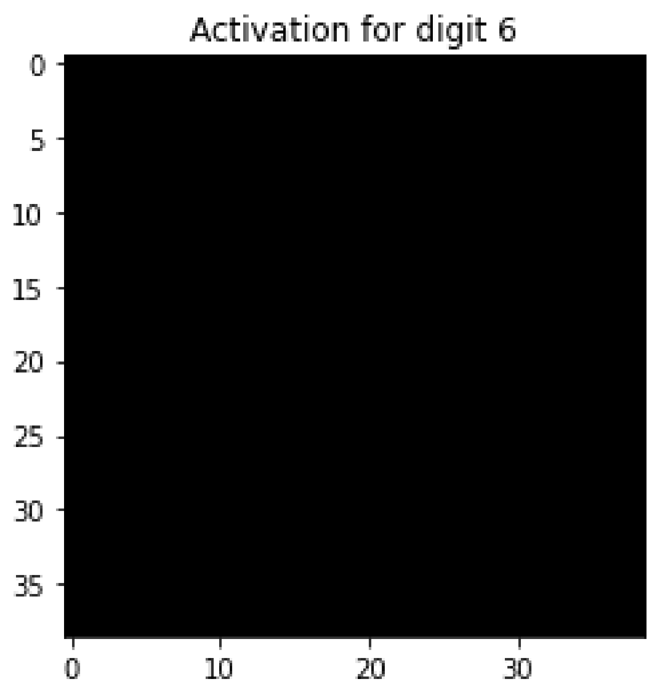

Activation for digit 1
0
5
10
15
20
25
30
35
0
10
20
30

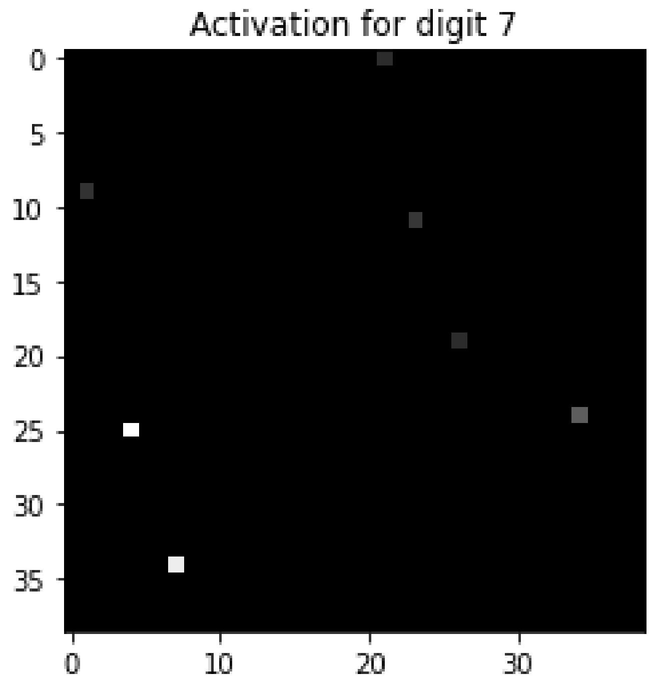

Activation for digit 2
0
5
10
15
20
25
30
35
0
10
20
30

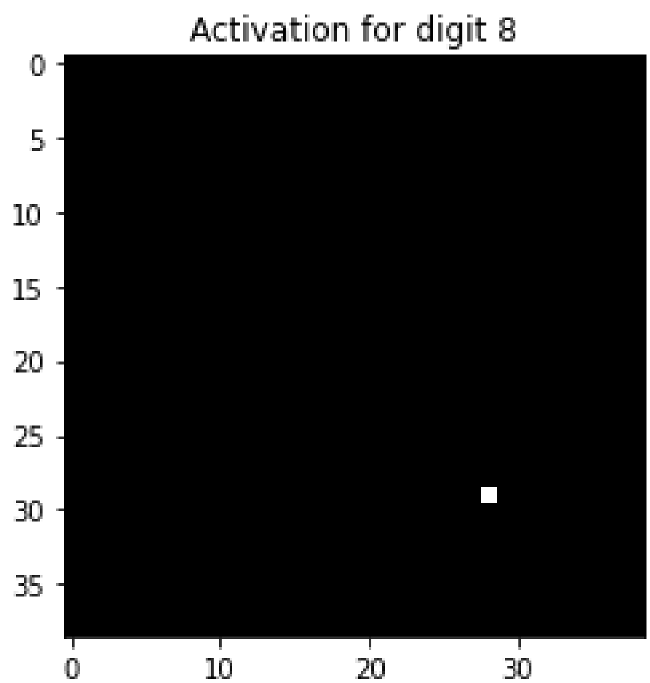

Activation for digit 3
0
5
10
15
20
25
30
35
0
10
20
30

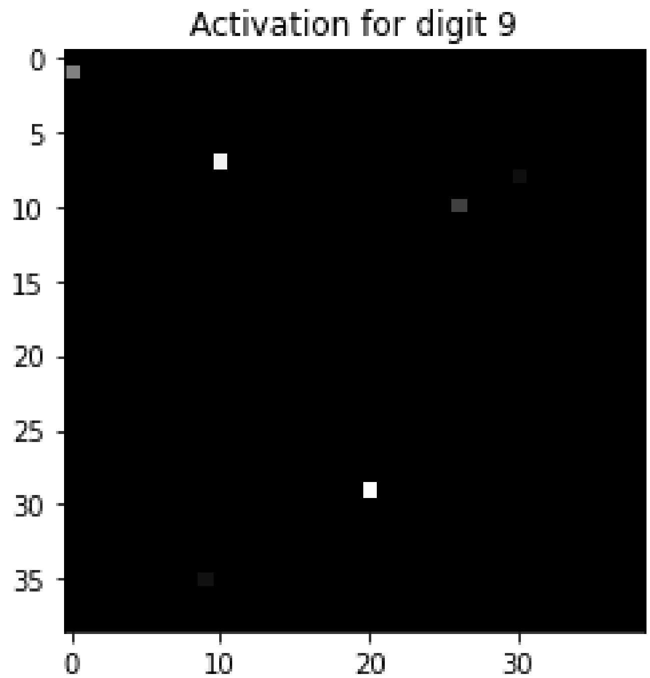

Activation for digit 4
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 5
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 6
0
5
10
15
20
25
30
35
0
10
20
30

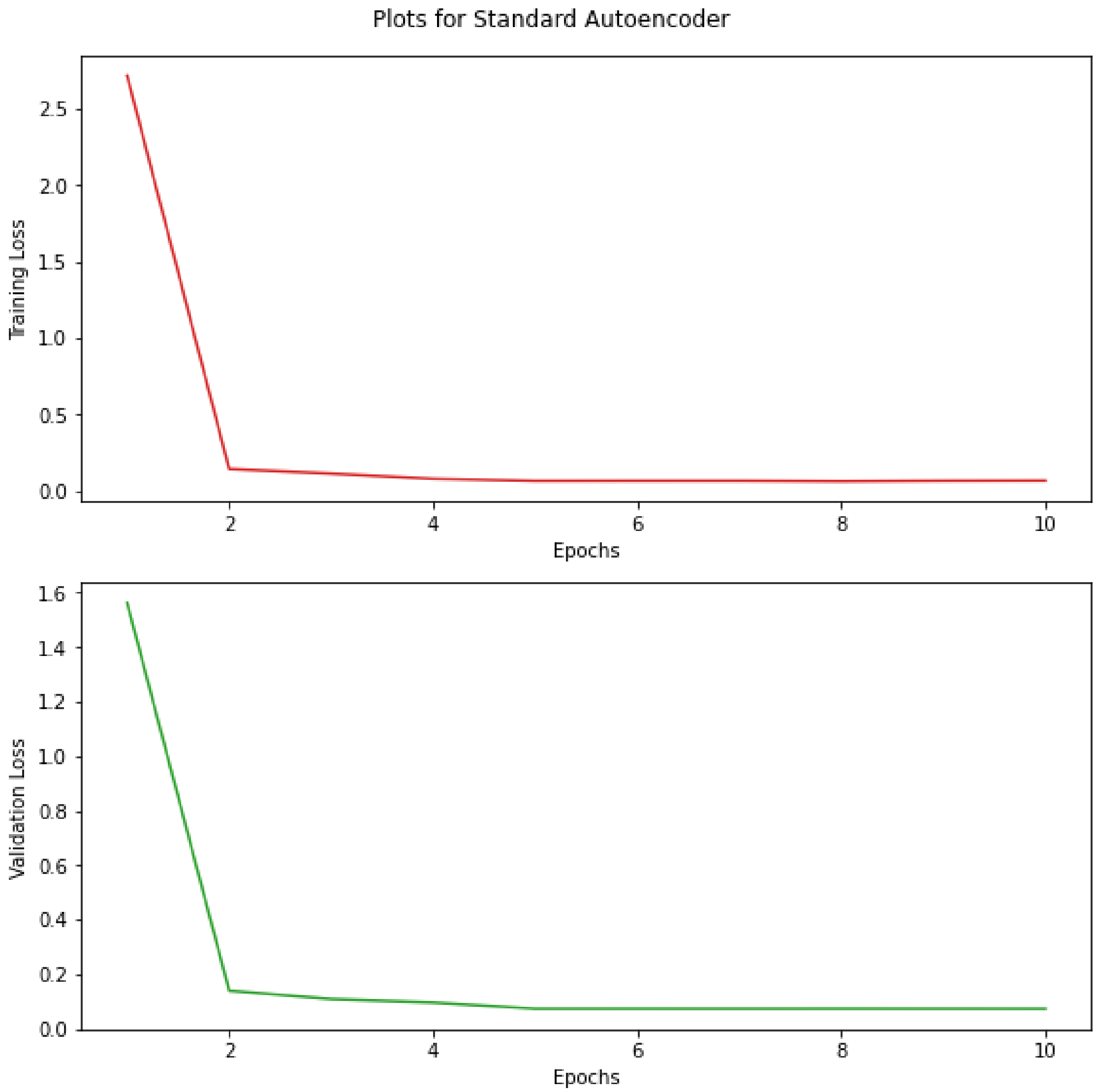

Activation for digit 7
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 8
0
5
10
15
20
25
30
35
0
10
20
30

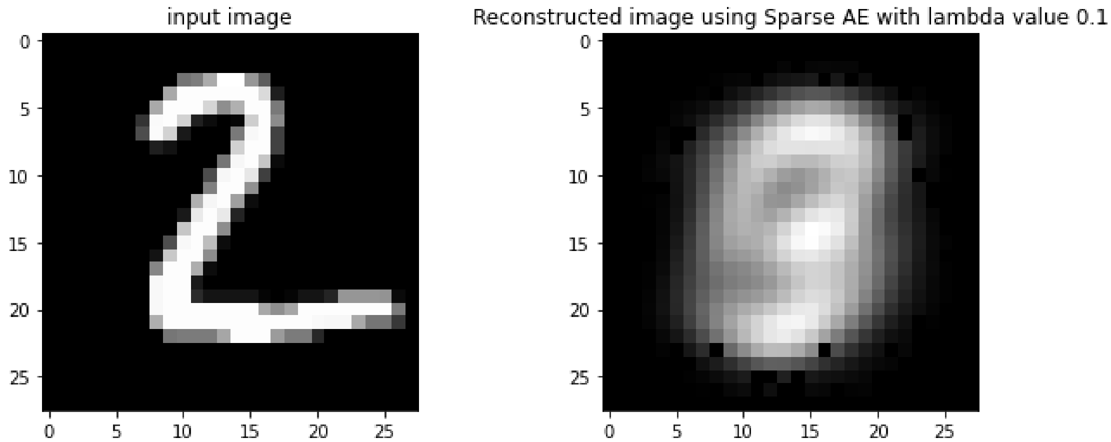



```
-----------------------------------------------------------
-----------------------------------------------------------
Training begins for regularization lambda value  0.1 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  2.71380352973938
Validation loss:  1.5620596408843994
-----------------------------------------
Epoch Number:  2
Training loss:  0.14296624064445496
Validation loss:  0.1392284780740738
-----------------------------------------
Epoch Number:  3
Training loss:  0.11376488953828812
Validation loss:  0.1082632914185524
-----------------------------------------
Epoch Number:  4
Training loss:  0.07981755584478378
Validation loss:  0.09571825712919235
-----------------------------------------
Epoch Number:  5
Training loss:  0.0655784159898758
Validation loss:  0.07309792935848236
-----------------------------------------
Epoch Number:  6
```

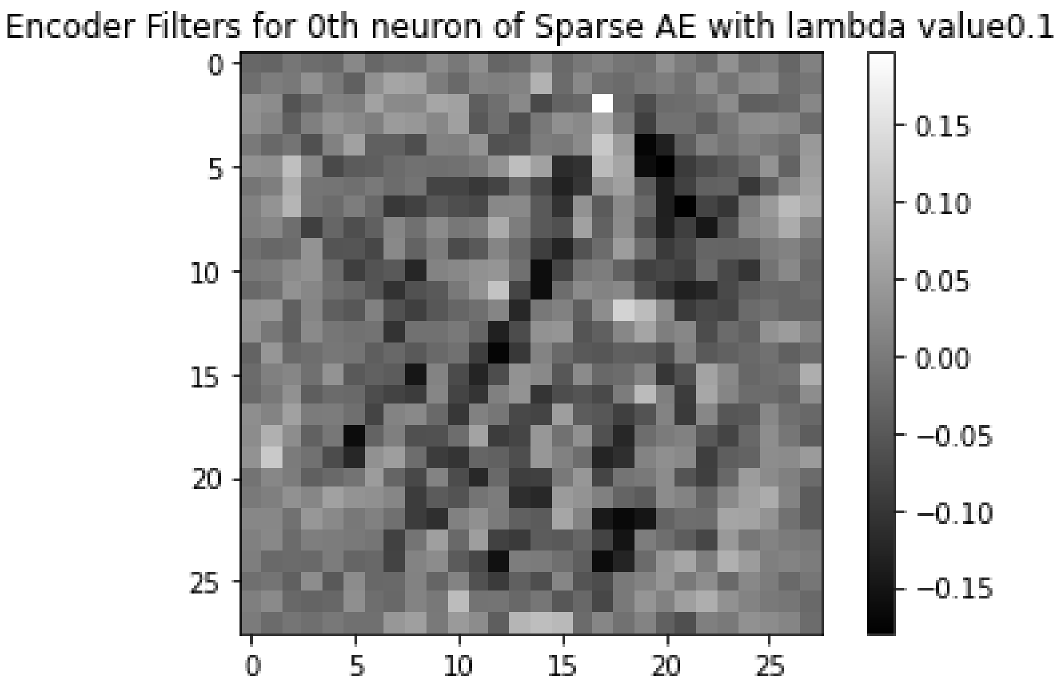
```
Training loss:  0.06599514186382294
Validation loss:  0.07321781665086746
-----------------------------------------
Epoch Number:  7
Training loss:  0.06614267081022263
Validation loss:  0.07310178875923157
-----------------------------------------
Epoch Number:  8
Training loss:  0.06285998225212097
Validation loss:  0.07327315956354141
-----------------------------------------
Epoch Number:  9
Training loss:  0.06616947799921036
Validation loss:  0.07315822690725327
-----------------------------------------
Epoch Number:  10
Training loss:  0.06715003401041031
Validation loss:  0.07301023602485657
-----------------------------------------
```



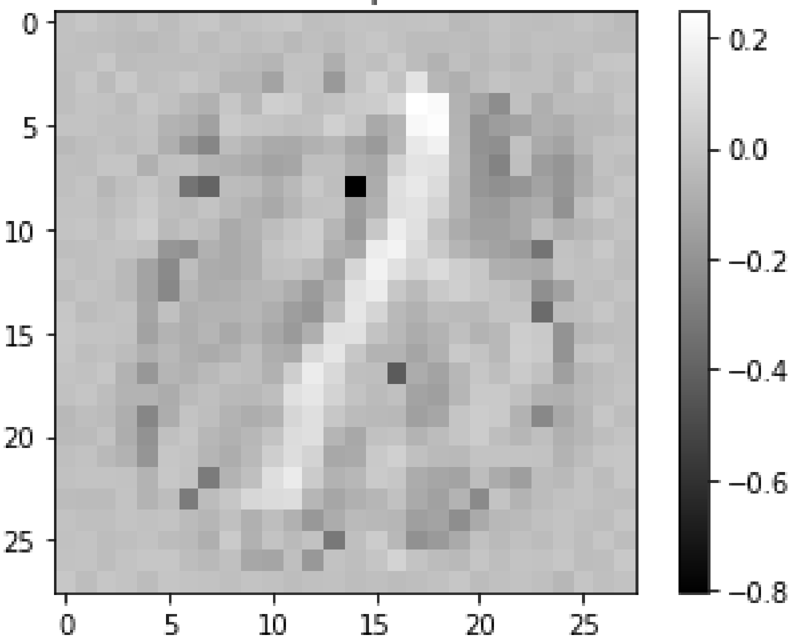

```
Training done for regularization lambda value  0.1 !
-----------------------------------------------------------
```



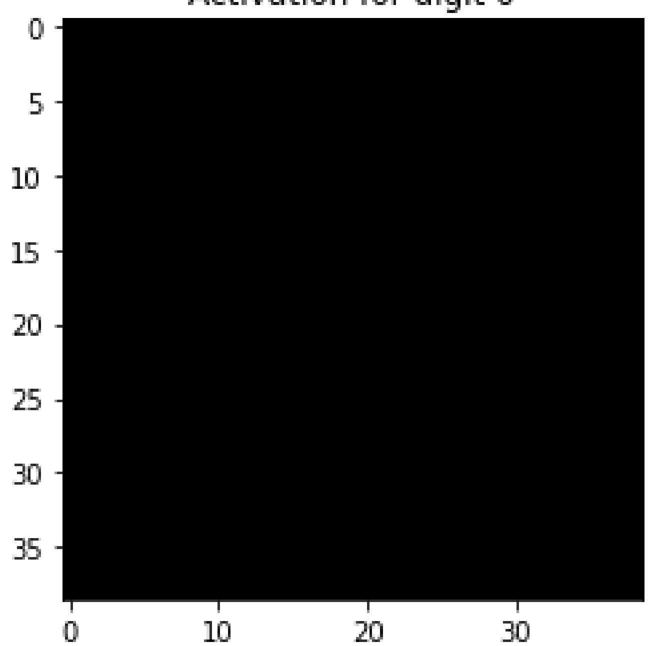
The average activation of Sparse AE with lambda value 0.1 is
4.6114608687622034e-07

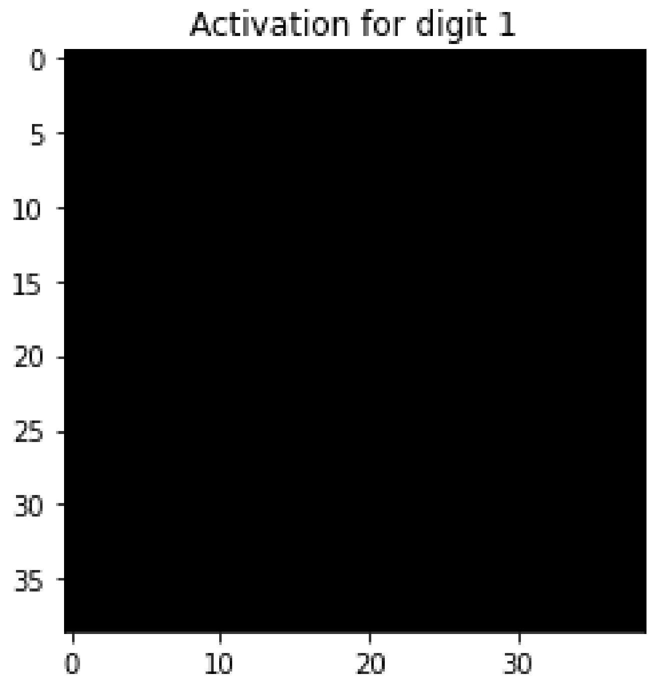


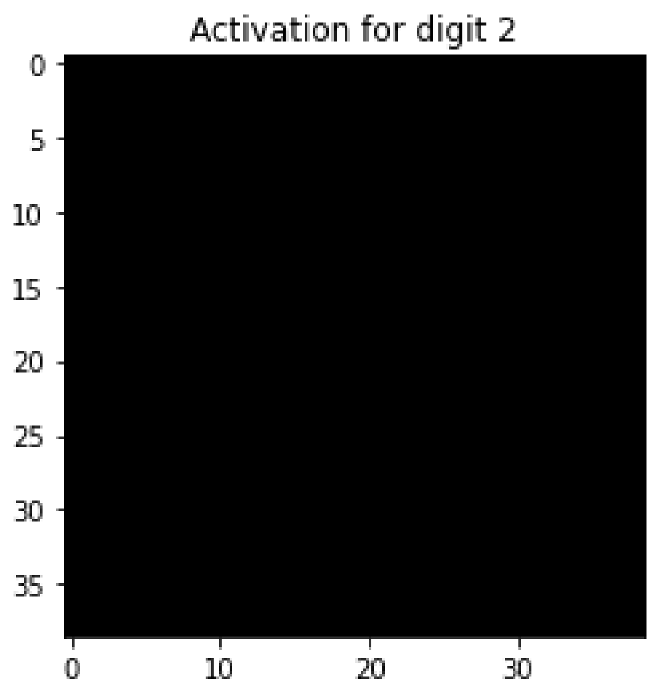
Decoder Filters for 0th neuron of Sparse AE with lambda value0.1



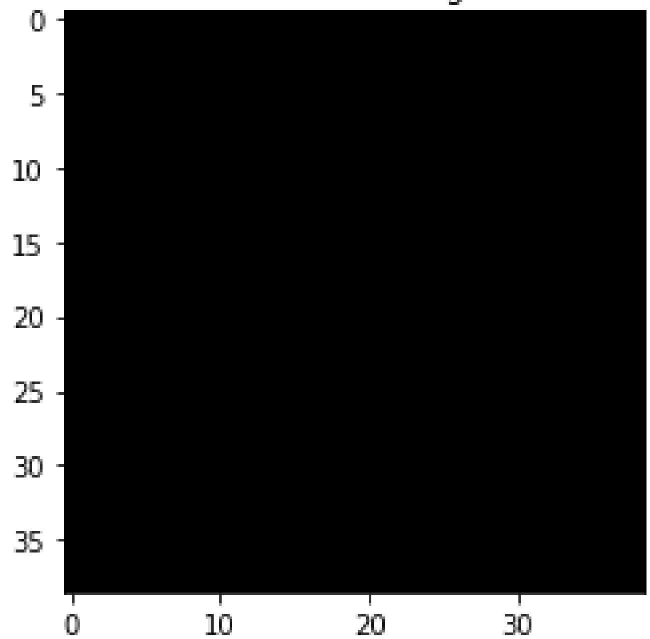
Activation for digit 0



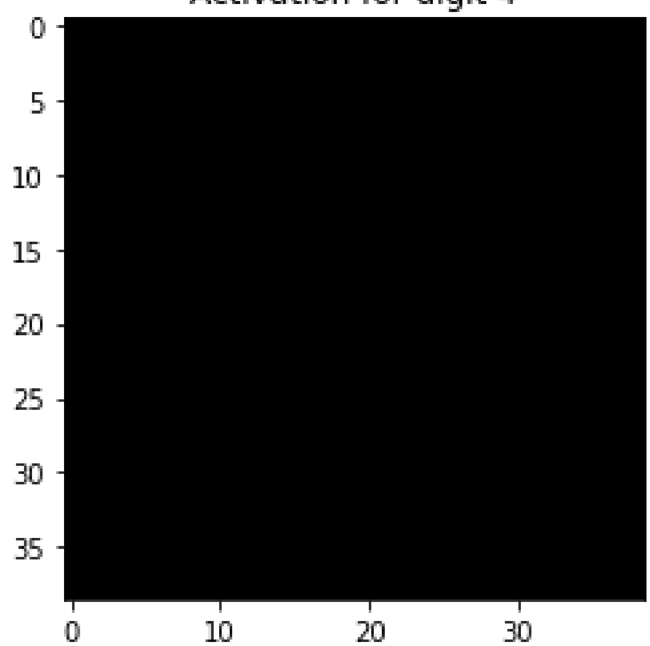

Activation for digit 1

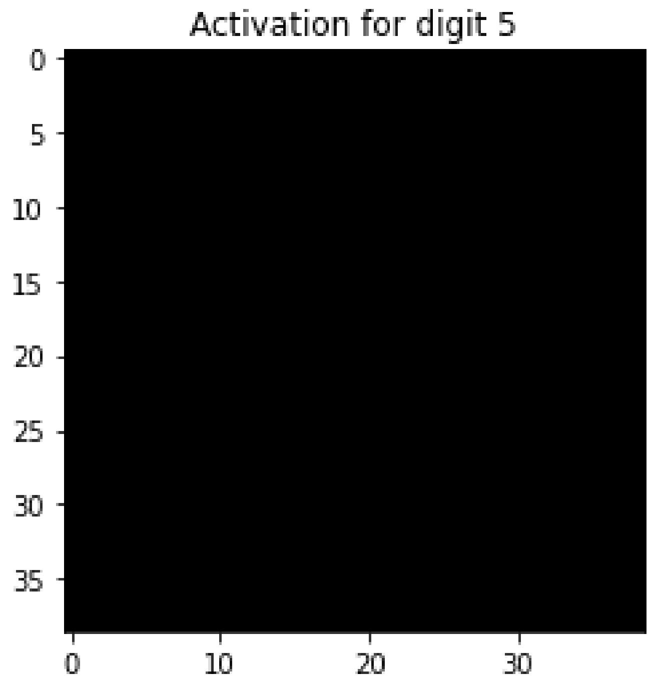

Activation for digit 2

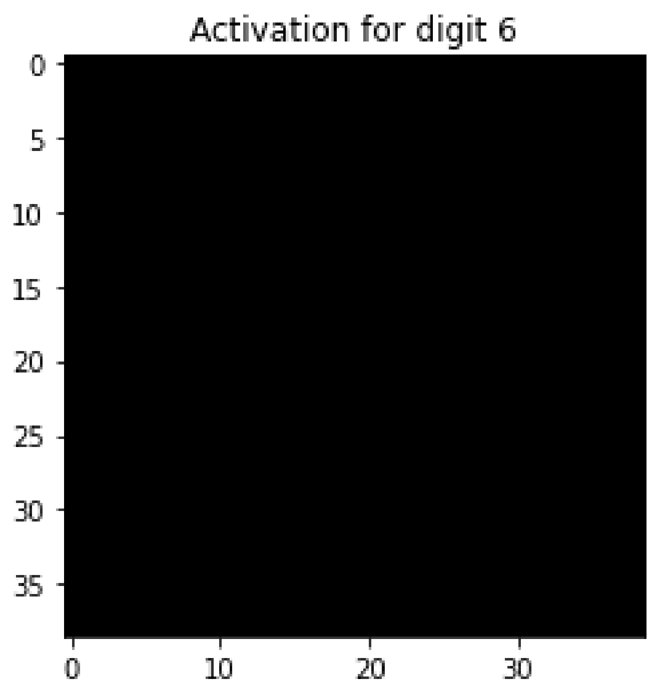
Activation for digit 3

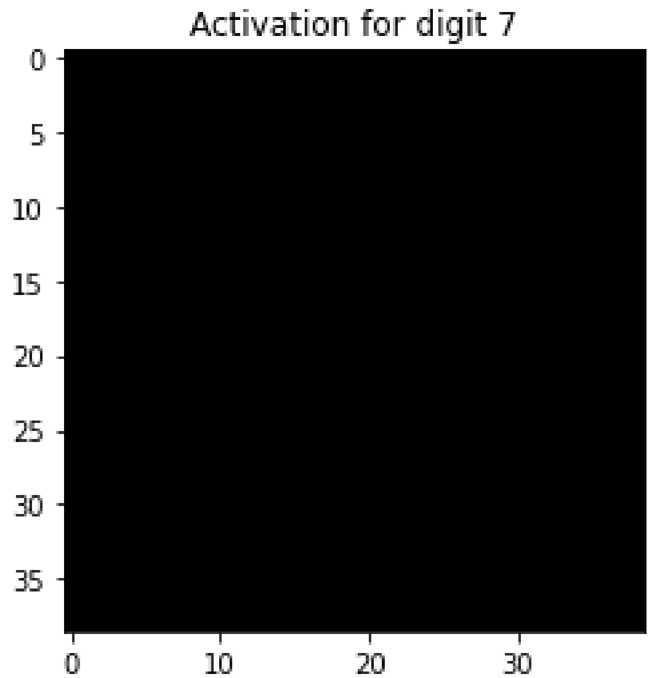


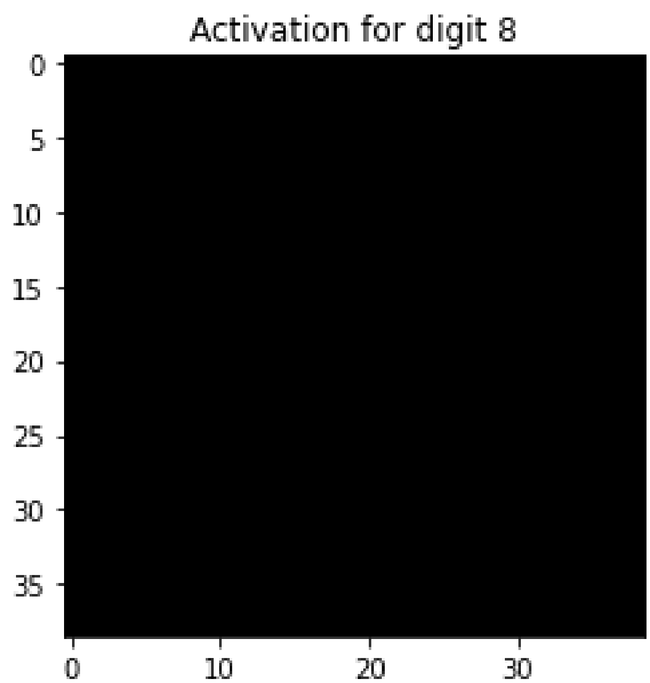
Activation for digit 4

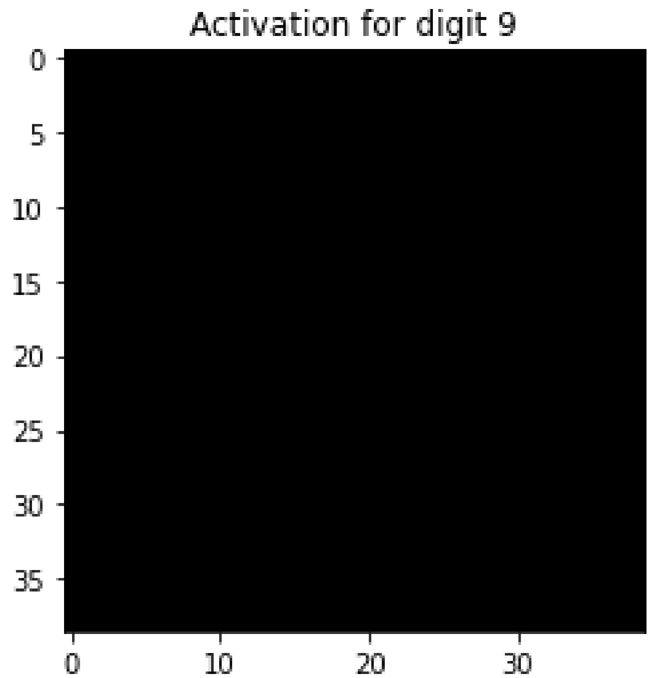



Activation for digit 5
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 6
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 7
0
5
10
15
20
25
30
35
0
10
20
30


Activation for digit 8
0
5
10
15
20
25
30
35
0
10
20
30



Observations

- For different sparsity values, it can be noted that for higher sparsity values the reconstruction is recognizable but not perfect in the sense that the strokes of digits are not continuous. However, for lower sparsity values, the reconstruction includes more pixels as part of the reconstructions making it more noisy.
- This can also be interpreted using the activation filters shown above, as the sparsity value is increased, the activation for each digit also becomes sparse.
- For the highest sparsity value (`lambda` value) chosen, it can be noticed that the activation for any digit is almost blank, it also reflects in the reconstructed image as it does not resemble the chosen digit at all.

4 Denoising Autoencoders

4.1 Experimentation with different noise values

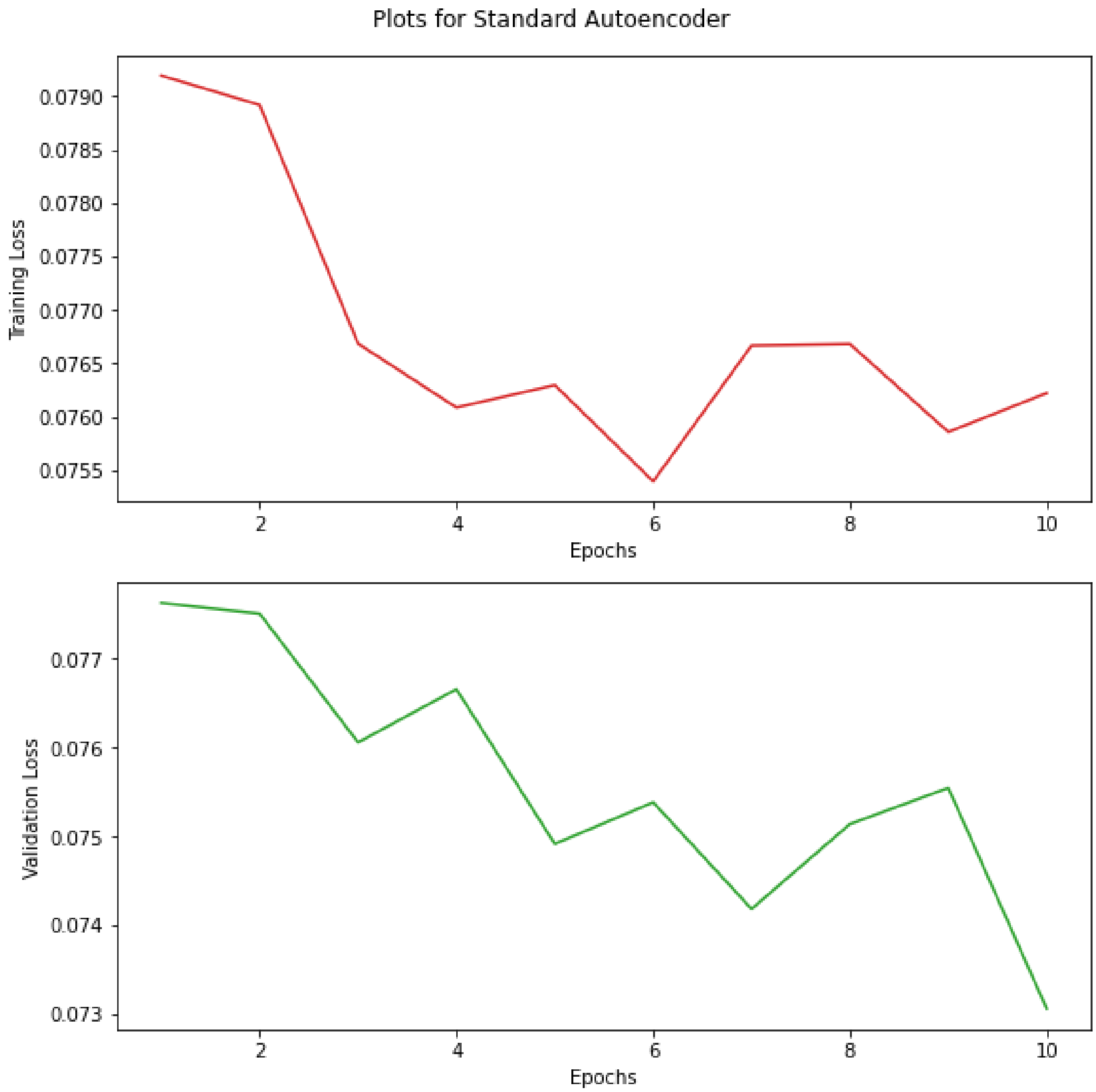
[21]:
```
noise_values = [0.3, 0.5, 0.8, 0.9]

for noise_val in noise_values:
  model_denoiseAE = standard_AE(256).to(device)
  optimizer = torch.optim.Adam(model_denoiseAE.parameters(), learning_rate)
```

```python
train_batch_loss = []
train_epoch_loss = []
val_loss = []
print('Training begins for noise value ', noise_val, '!')
print('------------------------------------------------------------')
for epoch in range(epochs):
  for batch_idx, (data, target) in enumerate(train_loader):

    noise = torch.randn(data.size())*noise_val
    data = data + noise
    data = data.to(device=device)
    target = target.to(device=device)

    model_denoiseAE.train()
    decoded, encoded = model_denoiseAE(data)
    loss = criterion(decoded, torch.flatten(data,1))
    train_batch_loss.append(loss.item())

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  train_epoch_loss.append(train_batch_loss[-1])

  model_denoiseAE.eval()
  with torch.no_grad():
    for (data, target) in test_loader:
      noise = torch.randn(data.size())*noise_val
      data = data + noise
      data = data.to(device=device)
      target = target.to(device)
      decoded, encoded = model_denoiseAE(data)
      val_iter_loss = criterion(decoded, torch.flatten(data,1))

  val_loss.append(val_iter_loss.item())
  print('Epoch Number: ',epoch+1)
  print('Training loss: ', train_epoch_loss[-1])
  print('Validation loss: ', val_loss[-1])
  print('-----------------------------------------')


fig, axs = plt.subplots(2, constrained_layout= True, figsize=(8,8))
fig.suptitle('Plots for Standard Autoencoder')
axs[0].set(ylabel='Training Loss', xlabel='Epochs')
axs[0].plot(range(1,epochs+1),train_epoch_loss, 'tab:red')
axs[1].set(ylabel='Validation Loss', xlabel='Epochs')
```

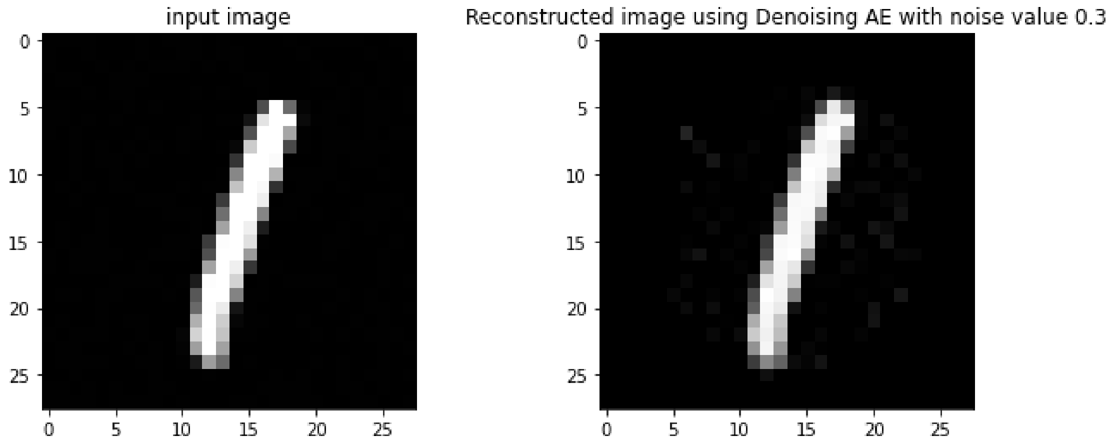
```
  axs[1].plot(range(1,epochs+1),val_loss, 'tab:green')
  plt.show()
  print('Training done for denoising AE with noise value',noise_val,'!')
  print('-----------------------------------------------------------')
  test_image = test_loader.dataset.data[5, :, :].clone() #RANDOM TEST IMAGE
  noise = torch.randn(test_image.size())*noise_val
  image = test_image + noise
  with torch.no_grad():
    image = image.view(-1,28,28).to(device=device).float()
    decoded, encoded = model_denoiseAE.forward(image)
    decoded = decoded.detach().cpu().numpy()
    image = image.reshape(28,28).detach().cpu().numpy()
    plt.subplot(1,2,1)
    plt.imshow(image , cmap='gray')
    plt.title('input image')
    plt.subplots_adjust(right=1.5)
    plt.subplot(1,2,2)
    plt.imshow(decoded.reshape(28,28),cmap ='gray')
    plt.title("Reconstructed image using Denoising AE with noise value␣
↪"+str(noise_val))
    plt.show()
  encoder_decoder_filters_plots( model_denoiseAE,"Denoising AE with noise␣
↪value"+str(noise_val),device)
  print('-----------------------------------------------------------')
  print('-----------------------------------------------------------')
```

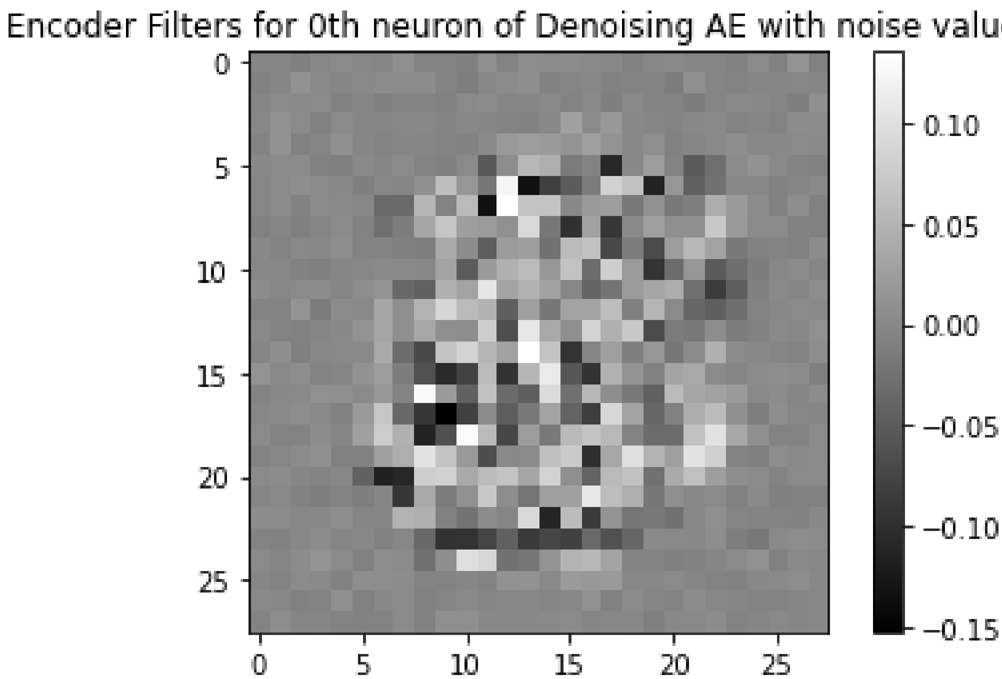
```
Training begins for noise value  0.3 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  0.07919100672006607
Validation loss:  0.07763116806745529
-----------------------------------------
Epoch Number:  2
Training loss:  0.07891659438610077
Validation loss:  0.07751056551933289
-----------------------------------------
Epoch Number:  3
Training loss:  0.07668483257293701
Validation loss:  0.07605963945388794
-----------------------------------------
Epoch Number:  4
Training loss:  0.07608699053525925
Validation loss:  0.0766594409942627
-----------------------------------------
Epoch Number:  5
Training loss:  0.07629714161157608
Validation loss:  0.07491587102413177
```

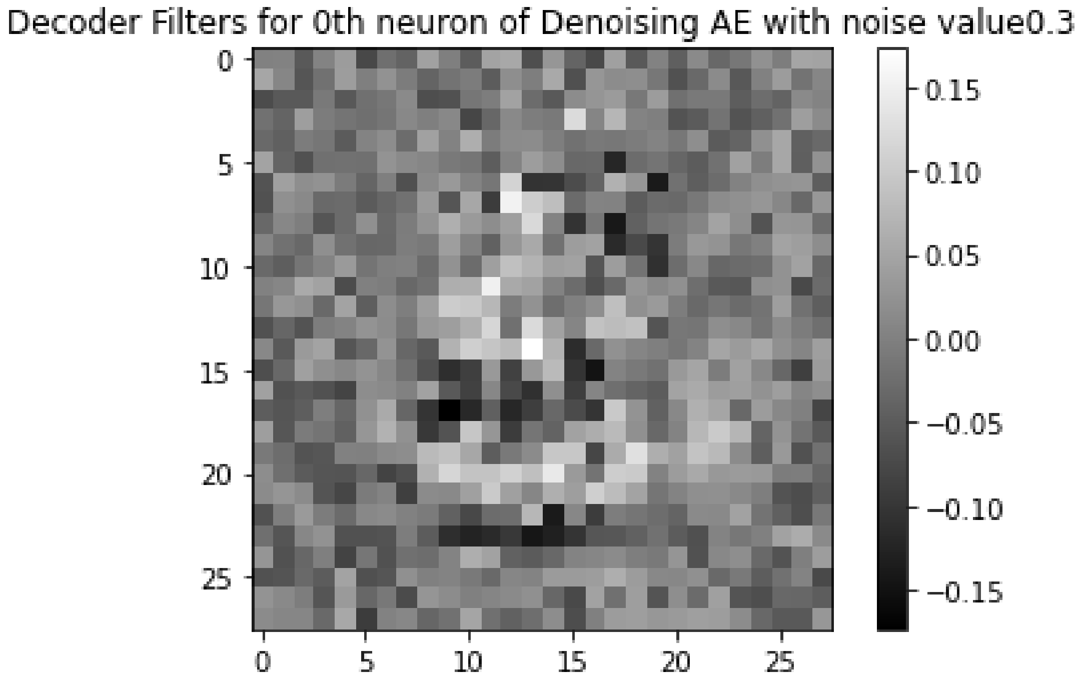
```
-----------------------------------------
Epoch Number:  6
Training loss:  0.07539673149585724
Validation loss:  0.0753844678401947
-----------------------------------------
Epoch Number:  7
Training loss:  0.07666618376970291
Validation loss:  0.07418350130319595
-----------------------------------------
Epoch Number:  8
Training loss:  0.07668063789606094
Validation loss:  0.0751395970582962
-----------------------------------------
Epoch Number:  9
Training loss:  0.07585837692022324
Validation loss:  0.07554743438959122
-----------------------------------------
Epoch Number:  10
Training loss:  0.07622240483760834
Validation loss:  0.07305971533060074
-----------------------------------------
```




```
Training done for denoising AE with noise value 0.3 !
------------------------------------------------------------
```

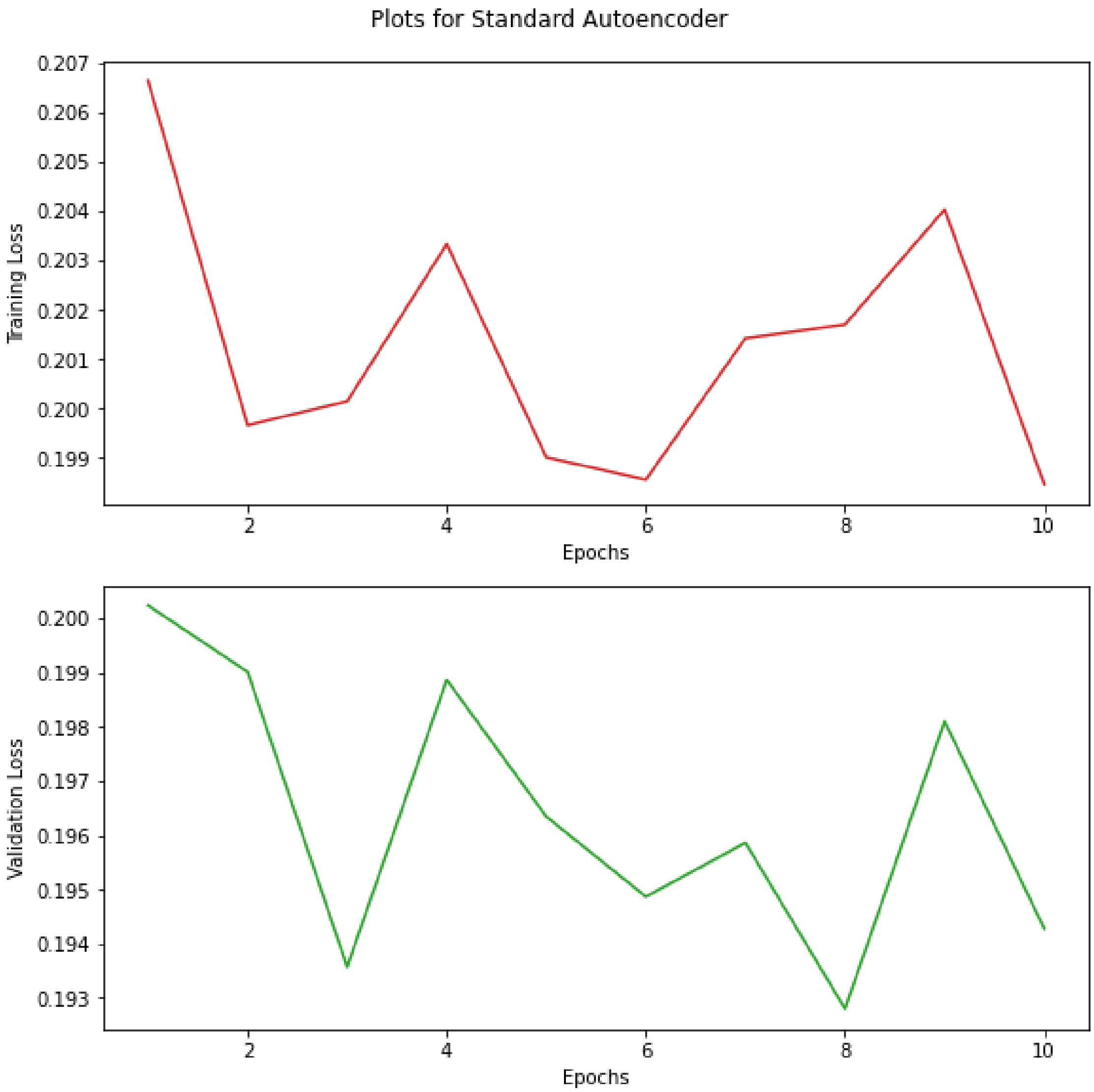

input image
Reconstructed image using Denoising AE with noise value 0.3
0
5
10
15
20
25


Encoder Filters for 0th neuron of Denoising AE with noise value0.3
0
5
10
15
20
25
0.10
0.05
0.00
−0.05
−0.10
−0.15

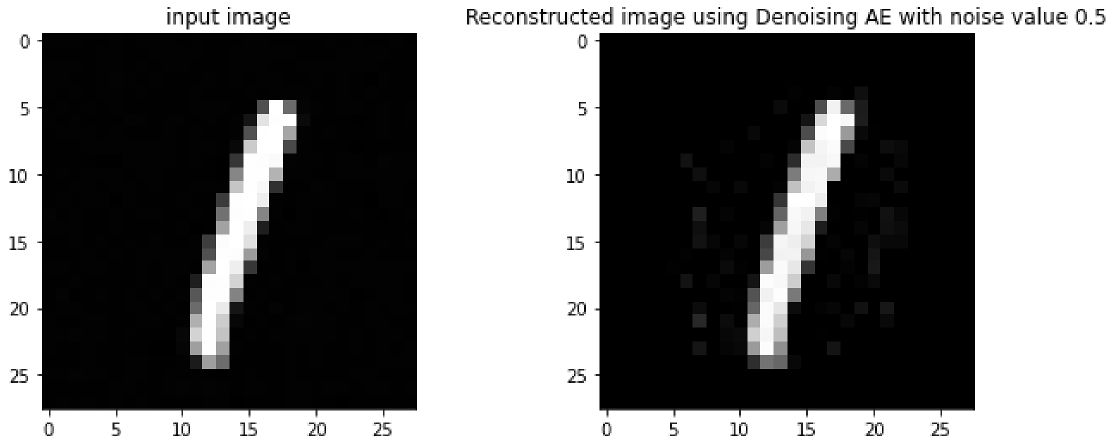


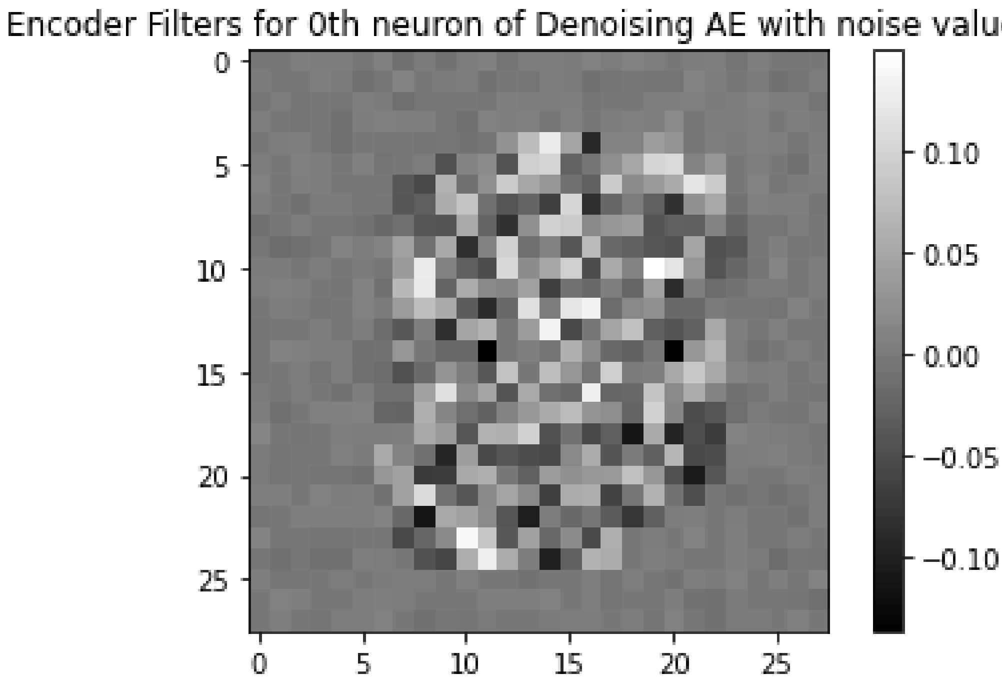

```
------------------------------------------------------------
------------------------------------------------------------
Training begins for noise value  0.5 !
------------------------------------------------------------
Epoch Number:  1
Training loss:  0.2066396325826645
Validation loss:  0.20023320615291595
-----------------------------------------
Epoch Number:  2
Training loss:  0.19966499507427216
Validation loss:  0.19900991022586823
-----------------------------------------
Epoch Number:  3
Training loss:  0.2001471221446991
Validation loss:  0.1935756355524063
-----------------------------------------
Epoch Number:  4
Training loss:  0.20333196222782135
Validation loss:  0.19886252284049988
-----------------------------------------
Epoch Number:  5
Training loss:  0.1990119367837906
Validation loss:  0.1963425874710083
-----------------------------------------
Epoch Number:  6
```

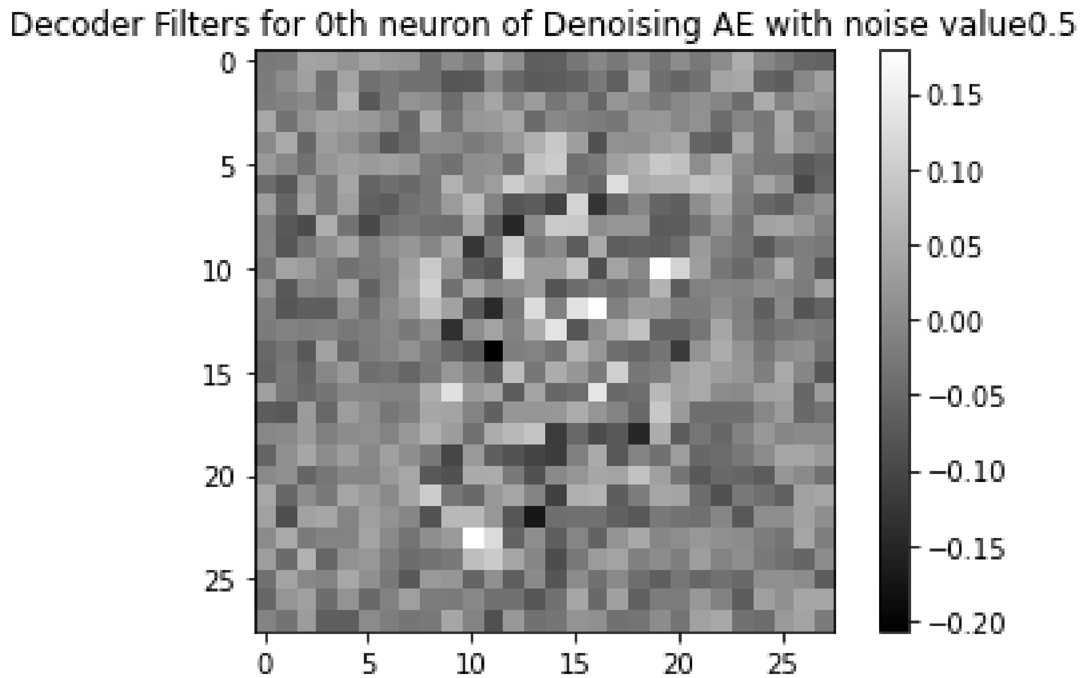
```
Training loss:  0.19856153428554535
Validation loss:  0.19486773014068604
-----------------------------------------
Epoch Number:  7
Training loss:  0.2014264464378357
Validation loss:  0.19586099684238434
-----------------------------------------
Epoch Number:  8
Training loss:  0.20170161128044128
Validation loss:  0.1928018033504486
-----------------------------------------
Epoch Number:  9
Training loss:  0.20403216779232025
Validation loss:  0.1980990767478943
-----------------------------------------
Epoch Number:  10
Training loss:  0.19847135245800018
Validation loss:  0.19428159296512604
-----------------------------------------
```




```
Training done for denoising AE with noise value 0.5 !
------------------------------------------------------------
```

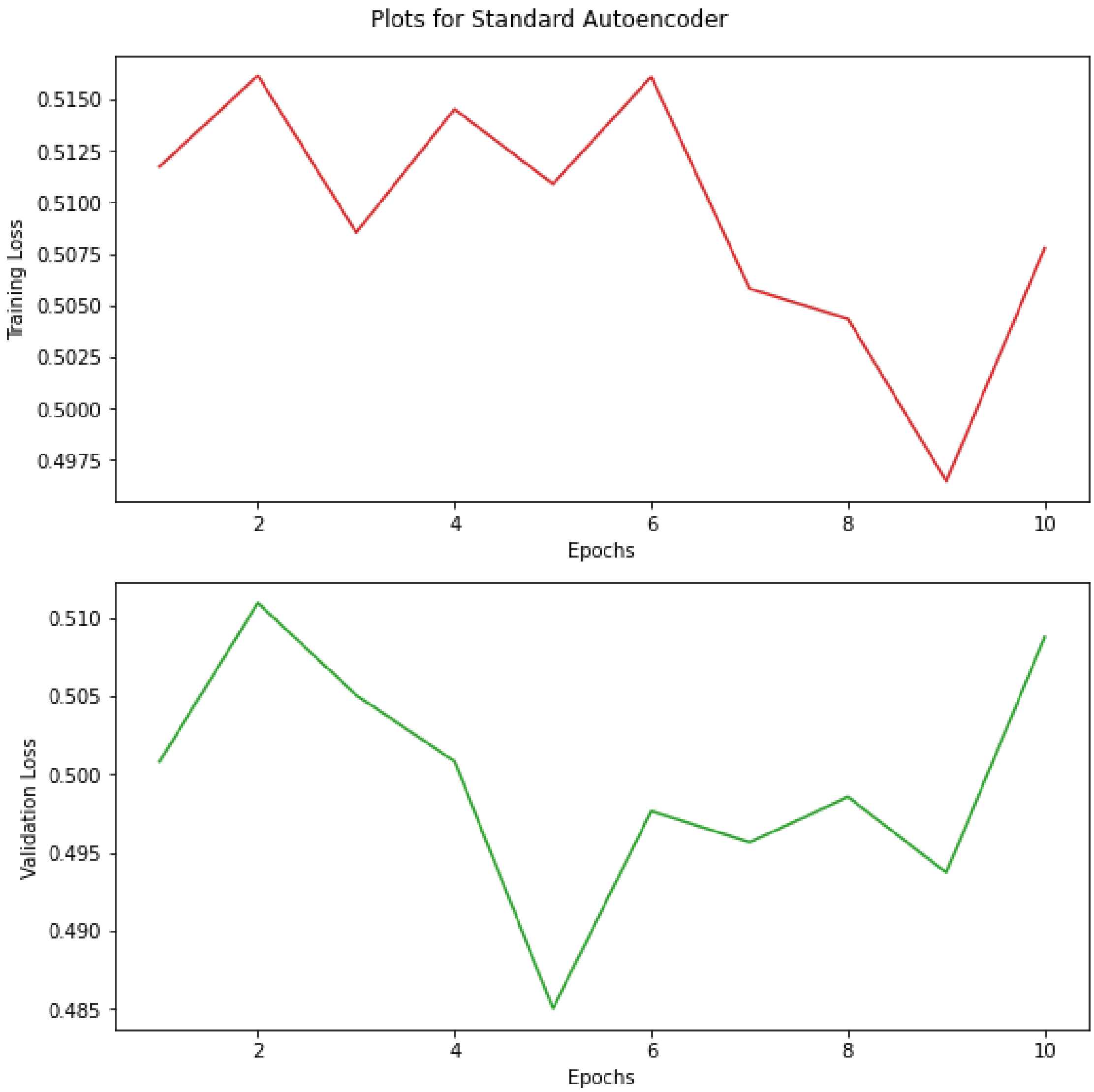

input image
Reconstructed image using Denoising AE with noise value 0.5
0
5
10
15
20
25


Encoder Filters for 0th neuron of Denoising AE with noise value0.5
0
5
10
15
20
25
0.10
0.05
0.00
−0.05
−0.10

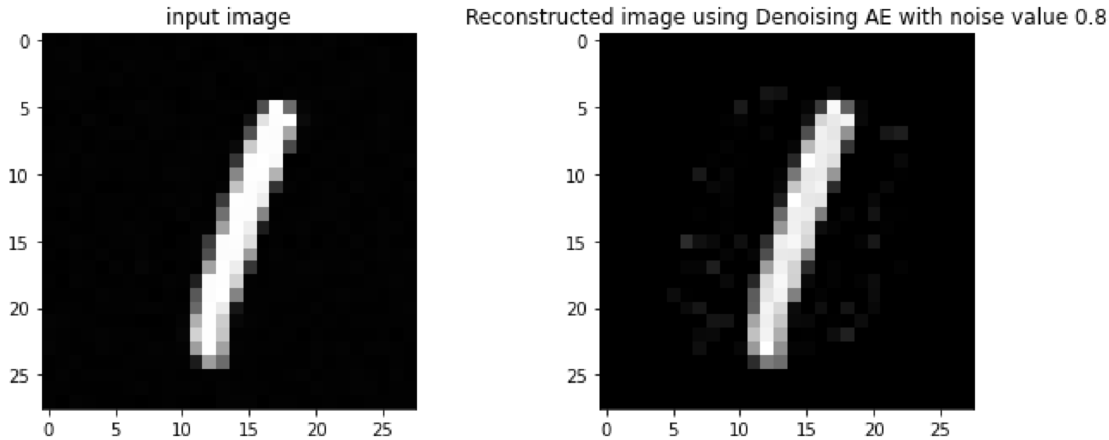


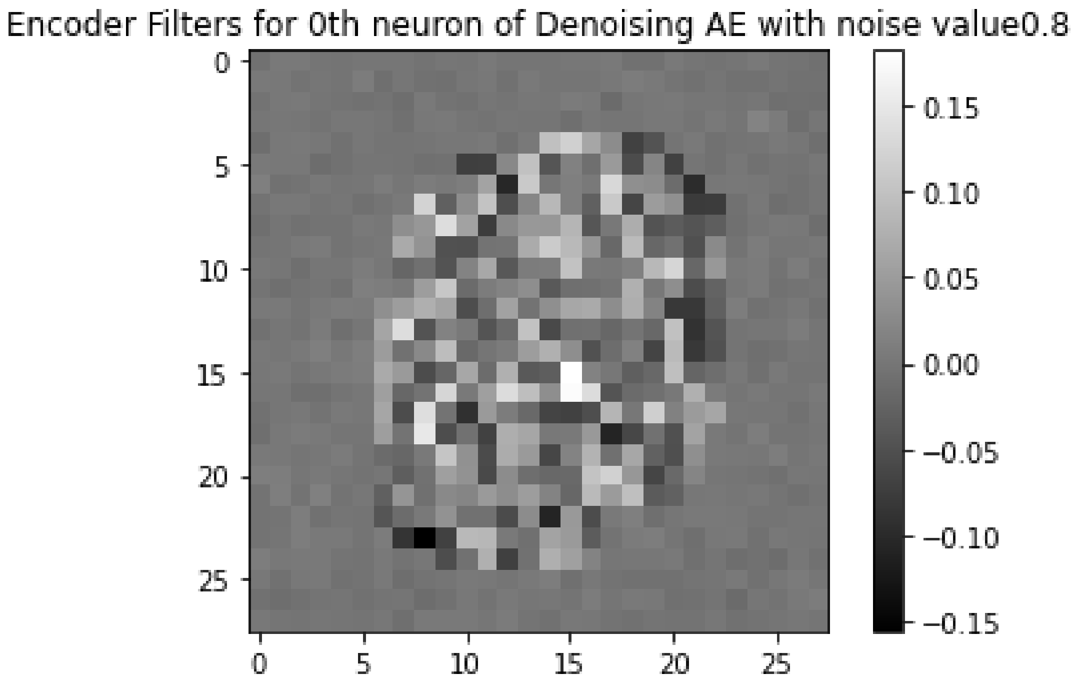

```
-----------------------------------------------------------
-----------------------------------------------------------
Training begins for noise value  0.8 !
-----------------------------------------------------------
Epoch Number:  1
Training loss:  0.5117299556732178
Validation loss:  0.5008126497268677
-----------------------------------------
Epoch Number:  2
Training loss:  0.516144871711731
Validation loss:  0.5109405517578125
-----------------------------------------
Epoch Number:  3
Training loss:  0.5085366368293762
Validation loss:  0.5050548315048218
-----------------------------------------
Epoch Number:  4
Training loss:  0.5145164132118225
Validation loss:  0.5008255839347839
-----------------------------------------
Epoch Number:  5
Training loss:  0.5108858942985535
Validation loss:  0.4850398302078247
-----------------------------------------
Epoch Number:  6
```

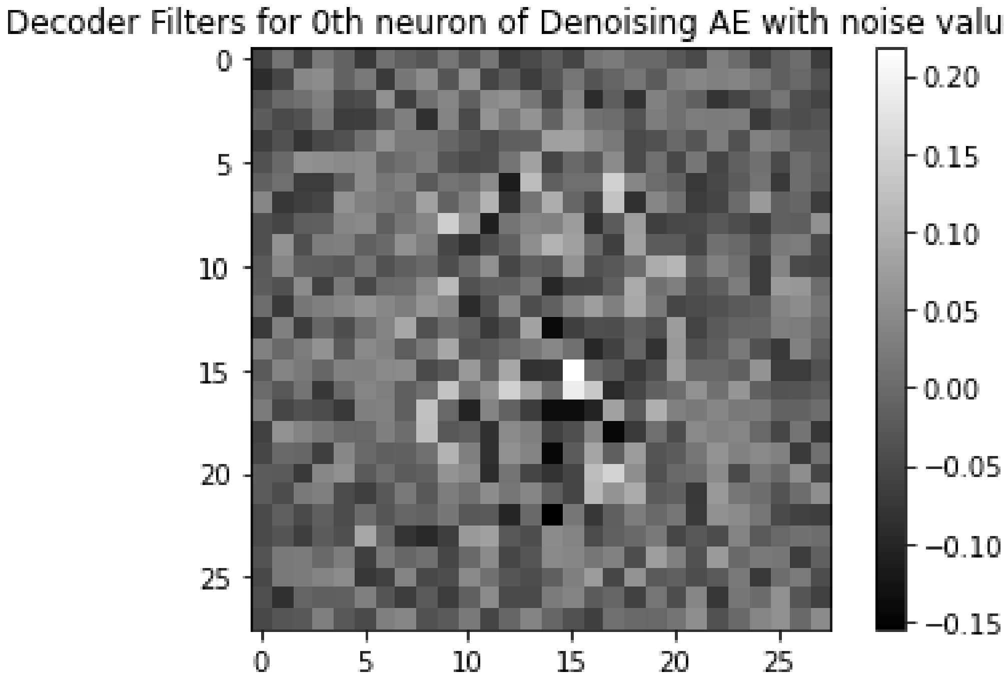
```
Training loss:  0.516089677810669
Validation loss:  0.49766290187835693
-----------------------------------------
Epoch Number:  7
Training loss:  0.505807101726532
Validation loss:  0.49566373229026794
-----------------------------------------
Epoch Number:  8
Training loss:  0.5043478012084961
Validation loss:  0.49855220317840576
-----------------------------------------
Epoch Number:  9
Training loss:  0.496466726064682
Validation loss:  0.4937262535095215
-----------------------------------------
Epoch Number:  10
Training loss:  0.507779598236084
Validation loss:  0.5087576508522034
-----------------------------------------
```




```
Training done for denoising AE with noise value 0.8 !
------------------------------------------------------------
```

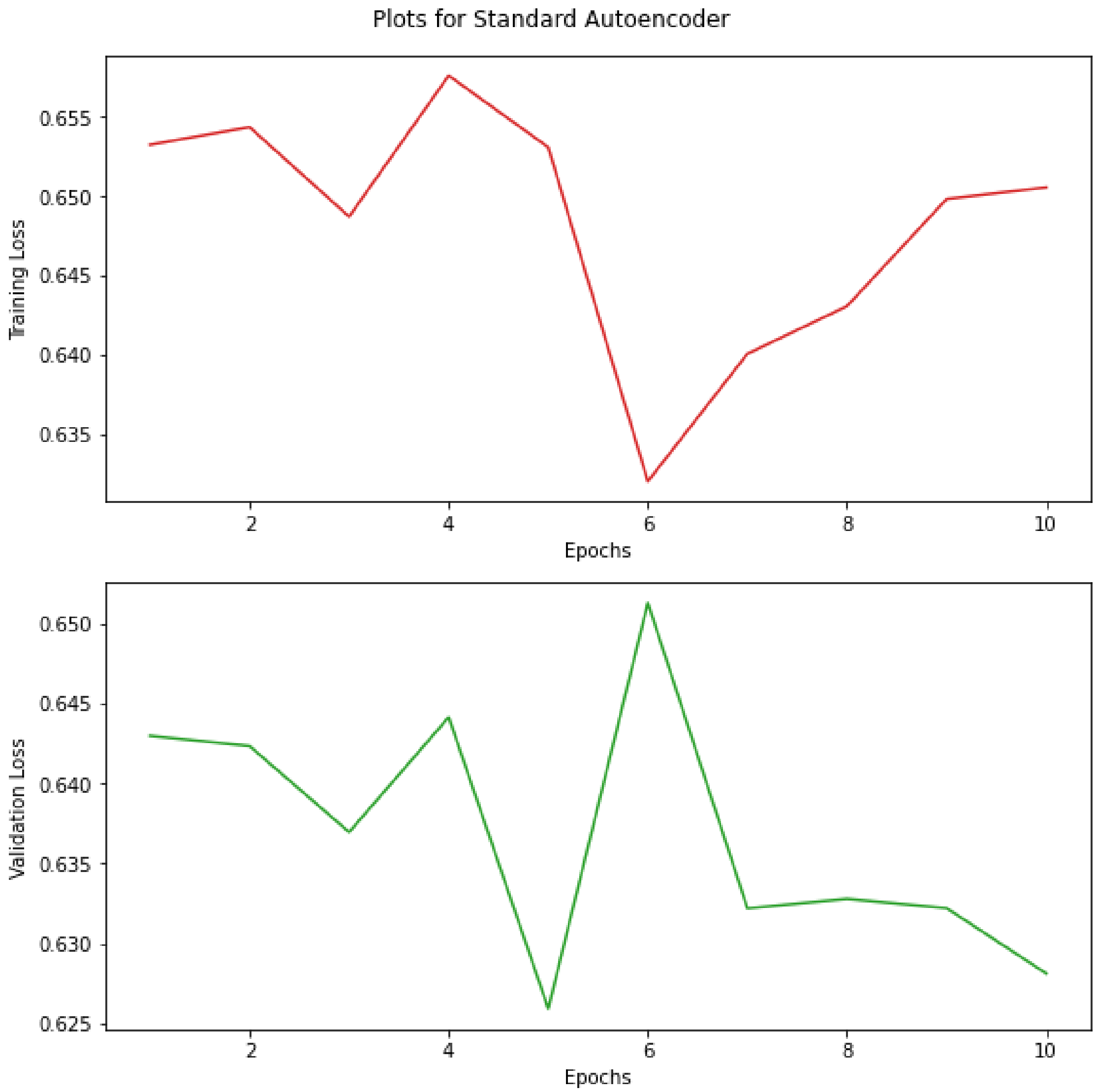

input image
Reconstructed image using Denoising AE with noise value 0.8
0
5
10
15
20
25


Encoder Filters for 0th neuron of Denoising AE with noise value0.8
0
5
10
15
20
25
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15

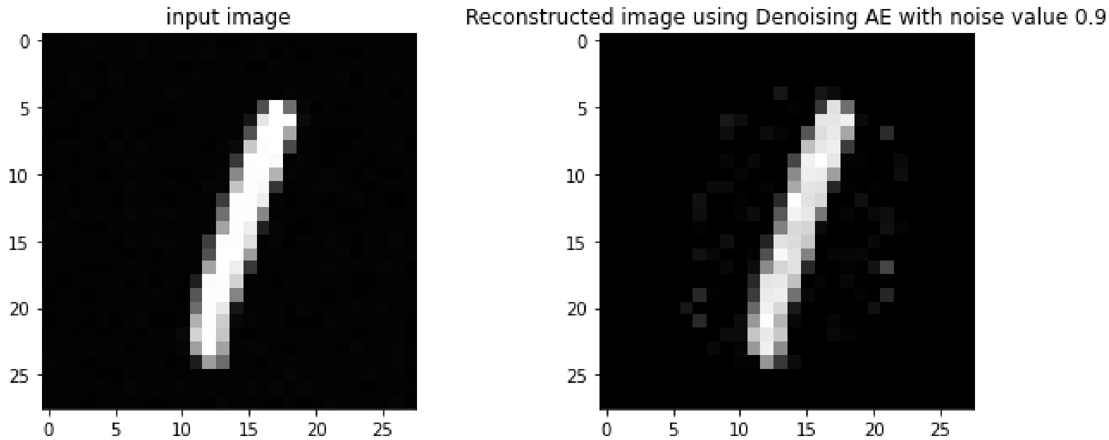


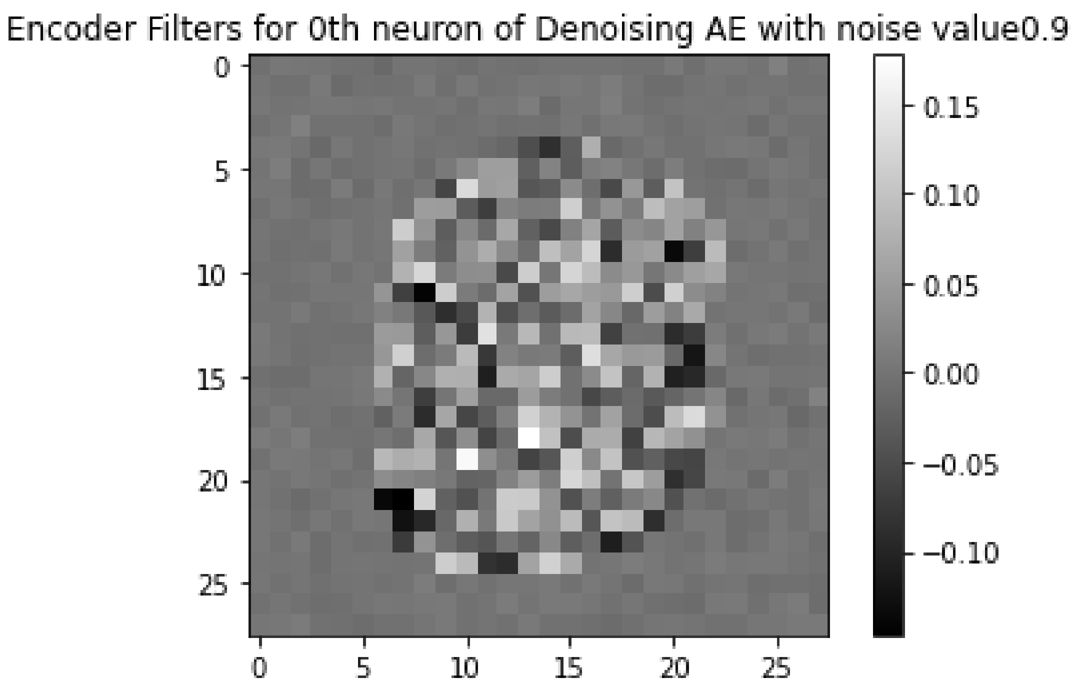

```
------------------------------------------------------------
------------------------------------------------------------
Training begins for noise value  0.9 !
------------------------------------------------------------
Epoch Number:  1
Training loss:  0.6532461047172546
Validation loss:  0.6429724097251892
------------------------------------------
Epoch Number:  2
Training loss:  0.6543552279472351
Validation loss:  0.6423315405845642
------------------------------------------
Epoch Number:  3
Training loss:  0.6487111449241638
Validation loss:  0.6369641423225403
------------------------------------------
Epoch Number:  4
Training loss:  0.657607913017273
Validation loss:  0.6441336274147034
------------------------------------------
Epoch Number:  5
Training loss:  0.6530838012695312
Validation loss:  0.625943124294281
------------------------------------------
Epoch Number:  6
```

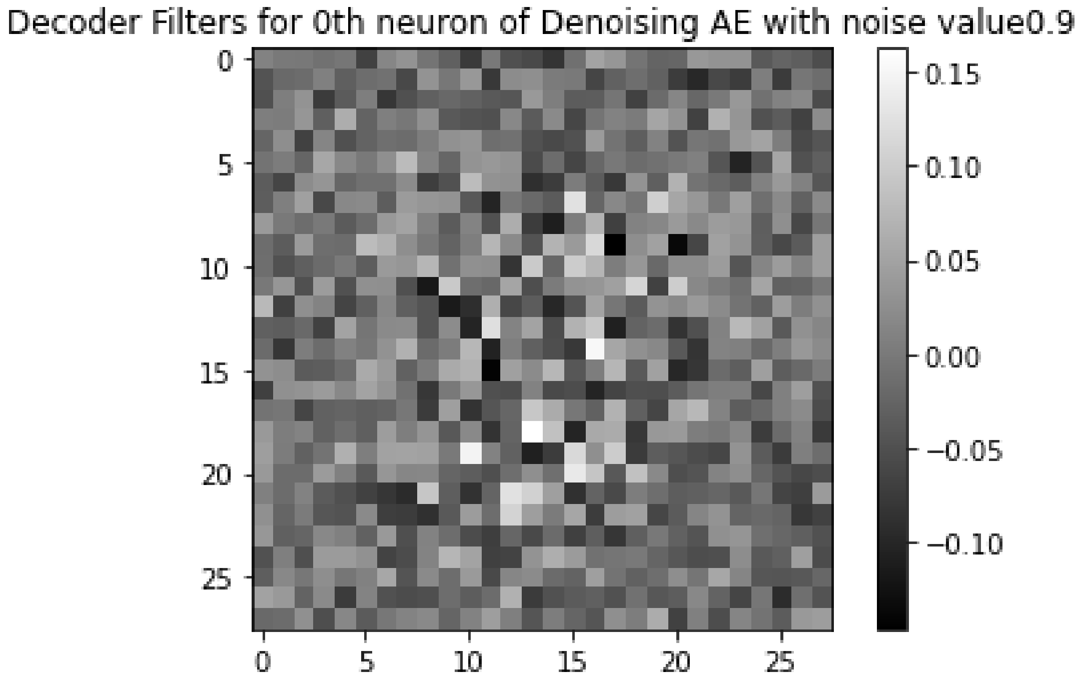
```
Training loss:  0.6320071816444397
Validation loss:  0.6512637138366699
-----------------------------------------
Epoch Number:  7
Training loss:  0.640059232711792
Validation loss:  0.6322088241577148
-----------------------------------------
Epoch Number:  8
Training loss:  0.6430586576461792
Validation loss:  0.6327941417694092
-----------------------------------------
Epoch Number:  9
Training loss:  0.6498229503631592
Validation loss:  0.6322207450866699
-----------------------------------------
Epoch Number:  10
Training loss:  0.650553286075592
Validation loss:  0.6281419992446899
-----------------------------------------
```




```
Training done for denoising AE with noise value 0.9 !
------------------------------------------------------------
```


input image
Reconstructed image using Denoising AE with noise value 0.9
0
5
10
15
20
25


Encoder Filters for 0th neuron of Denoising AE with noise value0.9
0
5
10
15
20
25
0.15
0.10
0.05
0.00
−0.05
−0.10



Observations

- As the noise value is increased, the reconstruction becomes worse but the shape of the digits are still recognizable.
- The learned filters are more focused and have higher variation in pixel values as the noise value increases.