Stevens Institute of Technology

Department of Electrical Engineering

EE628: Data Acquisition/Proc II

Homework Assignment 6

Submitted by:
Hadia Hameed
CWID: 10440803

Submitted to:
Prof. Rensheng Wang

March 28, 2019

1 Problem 1:

Use the posted Python Code file to test MNIST image data. Test if you can obtain the decoded images and show them properly.

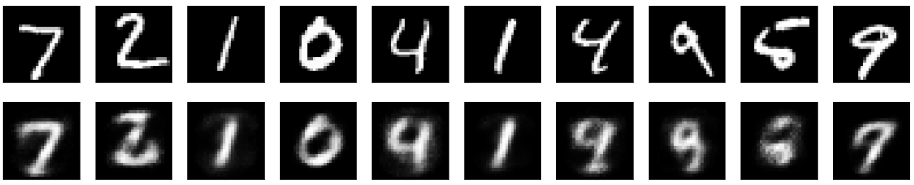


Figure 1: Output of auto-encoder for one layer and $epochs = 10$

2 Problem 2:

Use different epoch size from 5, 10, 20, 30, 40, 50, 100 and test the same code.

(a) $epochs = 5$

(b) $epochs = 20$

(c) $epochs = 30$

(d) $epochs = 40$

(e) $epochs = 50$

(f) $epochs = 100$

Figure 2: Results for digit recognition using single-layer auto-encoder for different values of epochs

3 Problem 3:

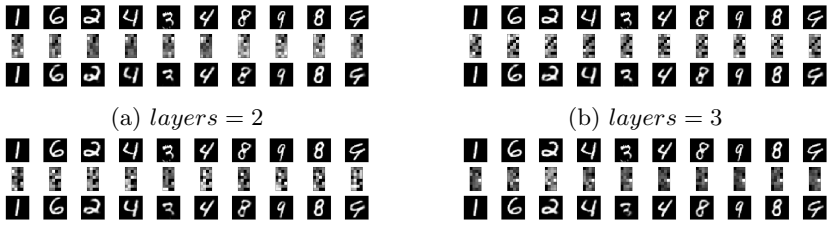
Use different number of layers of the deep networks (2, 3, 4, 5) and compare the performances.


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_5 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_6 (Dense)              (None, 784)               101136
=================================================================
Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Auto-encoder with 3 layers

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 192)               150720
_________________________________________________________________
dense_6 (Dense)              (None, 128)               24704
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_8 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_9 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_10 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_11 (Dense)             (None, 192)               24768
_________________________________________________________________
dense_12 (Dense)             (None, 784)               151312
=================================================================
Total params: 372,272
Trainable params: 372,272
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Auto-encoder with 4 layers



(a) $layers = 2$ (b) $layers = 3$

(c) $layers = 4$ (d) $layers = 5$

Figure 5: Results for digit recognition using multi-layer auto-encoder for different number of layers with $epochs = 50$

4 Problem 4: Use auto encoder to find anomaly images:

4.1 You randomly download a few images other than the MNIST image dataset. For example, animals, furnitures, automobiles, etc.

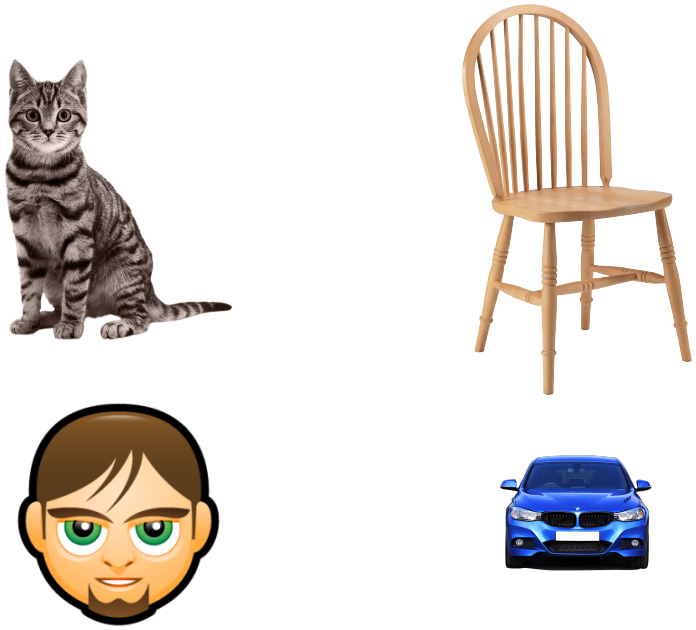


Figure 6: Chosen anomalous images

4.2 Reform these images to a grayscale images without color channels (similar to the minist dataset).

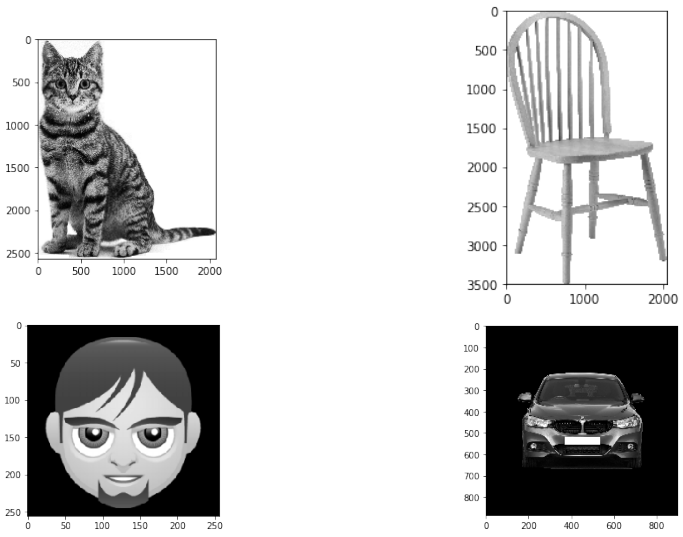


Figure 7: Reformed images

4.3 Use the above auto encoder to predict these download and reformed grayscale images. Show how much differences between the input images and the reconstructed after auto-encoder images.

Figure 8: Reconstruction by the auto-encoder. Detected as anomalies.

4.4 Code:

```
from PIL import Image
from resizeimage import resizeimage

x=Image.open('face.png','r')
x=x.convert('L') #makes it greyscale
x = resizeimage.resize_cover(x, [28, 28])
```



```
y=np.asarray(x.getdata(),dtype=np.float64).reshape((x.size[1],x.size[0]))

yy = y.reshape(1, 784)
encoded_imgs = encoder.predict(yy)
decoded_imgs = decoder.predict(encoded_imgs)

# display original
ax = plt.subplot(2, 1, 1)
plt.imshow(yy.reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

# display reconstruction
ax = plt.subplot(2, 1, 2)
plt.imshow(decoded_imgs.reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
plt.show()
```

