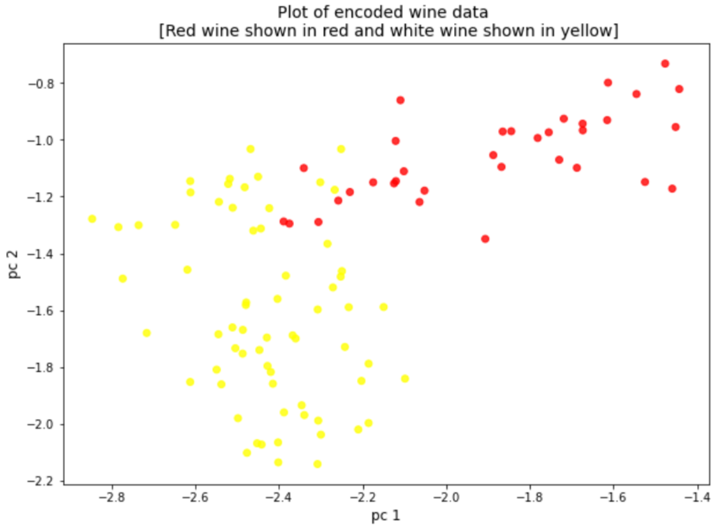
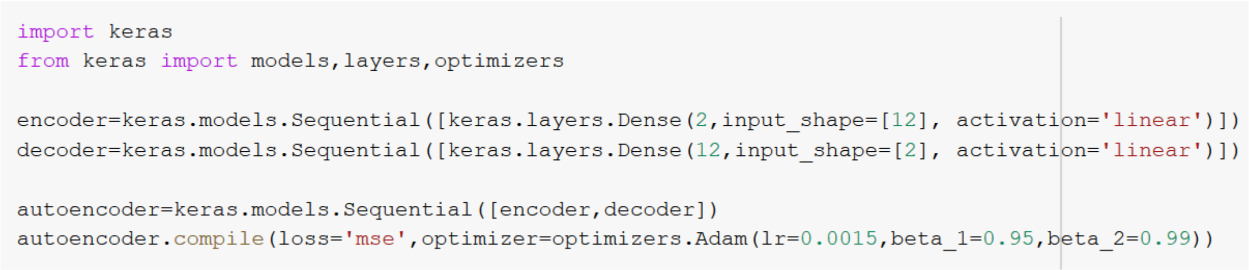
Name: Anand Krishnamoorthy
CSCI S-89 Introduction to Deep Learning
Assignment 5

Problem 1 (20 points)

Consider the winequality-red.csv and winequality-white.csv datasets of features (fixed acidity, volatile acidity, citric acid, residual sugar, etc.) that correspond to white and red wine, respectively. Load the datasets and merge them into one dataframe. Next, log-transform all entries (in case of citric acid you may want to add 1 prior to the transformation) and then scale the attributes. Using these scaled attributes, please build 1st and 2nd Principal Components using an autoencoder - a fully connected feedforward Neural Network with two units in the hidden layer and linear activation functions. Plot the observations in the plane of the two Principal Components you found. For each point, please indicate (using either shapes or colors) whether it corresponds to red or white wine.

SOLUTION:





```
import keras
from keras import models,layers,optimizers

encoder=keras.models.Sequential([keras.layers.Dense(2,input_shape=[12], activation='linear')])
decoder=keras.models.Sequential([keras.layers.Dense(12,input_shape=[2], activation='linear')])

autoencoder=keras.models.Sequential([encoder,decoder])
autoencoder.compile(loss='mse',optimizer=optimizers.Adam(lr=0.0015,beta_1=0.95,beta_2=0.99))
```

Problem 2 (20 points)

Please load the MNIST dataset of handwritten digits as follows:

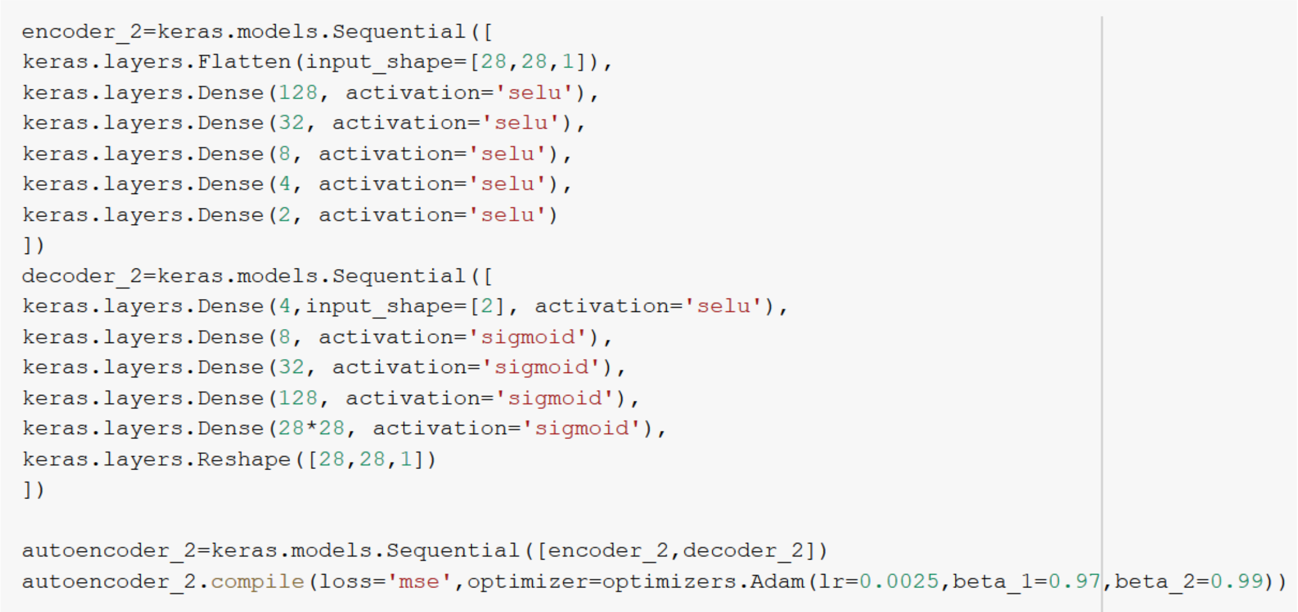
```
from keras.datasets import mnist
import numpy as np
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Build a fully connected feedforward undercomplete autoencoder of your choice using `X_train/255` (i.e. you need to scale the inputs/outputs). Try to make the number of codings as small as possible, given the reconstructed digits are still recognizable. Plot train/validation loss. Show 5 randomly selected test images (i.e. from `X_test/255`) along with their corresponding reconstructions. What is the smallest number of codings you had to use to get these results?

SOLUTION:

Smallest coding = 2

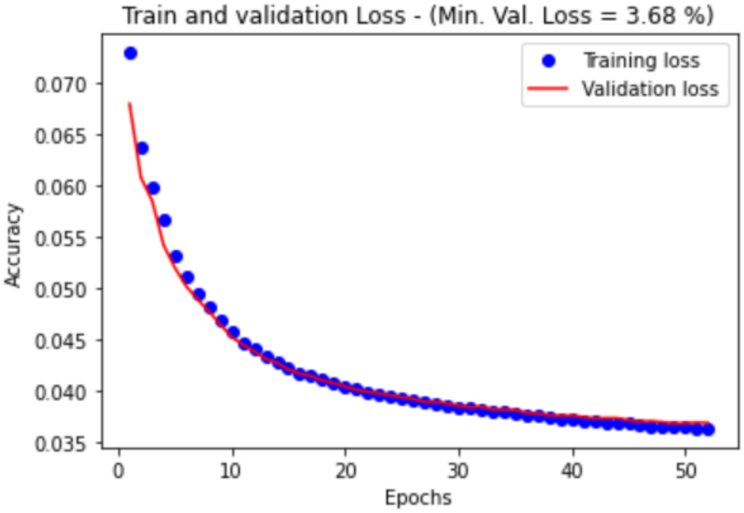
Model Architecture:



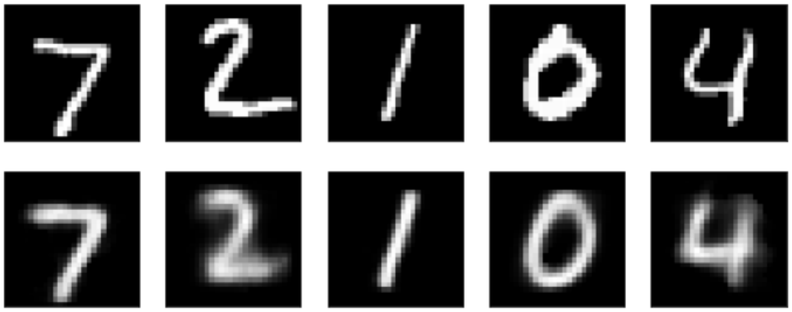
```
encoder_2=keras.models.Sequential([
keras.layers.Flatten(input_shape=[28,28,1]),
keras.layers.Dense(128, activation='selu'),
keras.layers.Dense(32, activation='selu'),
keras.layers.Dense(8, activation='selu'),
keras.layers.Dense(4, activation='selu'),
keras.layers.Dense(2, activation='selu')
])
decoder_2=keras.models.Sequential([
keras.layers.Dense(4,input_shape=[2], activation='selu'),
keras.layers.Dense(8, activation='sigmoid'),
keras.layers.Dense(32, activation='sigmoid'),
keras.layers.Dense(128, activation='sigmoid'),
keras.layers.Dense(28*28, activation='sigmoid'),
keras.layers.Reshape([28,28,1])
])

autoencoder_2=keras.models.Sequential([encoder_2,decoder_2])
autoencoder_2.compile(loss='mse',optimizer=optimizers.Adam(lr=0.0025,beta_1=0.97,beta_2=0.99))
```

Train/Validation loss plot:



Results:



Problem 3 (25 points)

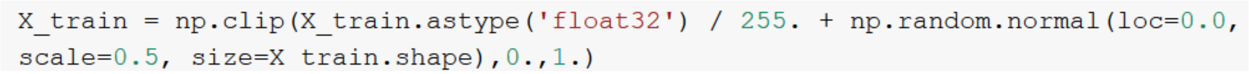
Please again consider the MNIST dataset. This time build a convolutional autoencoder of your choice. Make sure the autoencoder is undercomplete so that it is enforced to extract the most useful features. Use `X_train/255` as outputs and their noisy counterparts

```
np.clip(X_train/255 + np.random.normal(loc=0.0,
                                        scale=0.5,
                                        size=X_train.shape),0.,1.)
```

as inputs for training the autoencoder. Please use `'binary_crossentropy'` loss. Demonstrate that noise is minimized by showing 5 noisy images from the test set (you need to scale and add noise similarly to `X_train`) along with their reconstructions.

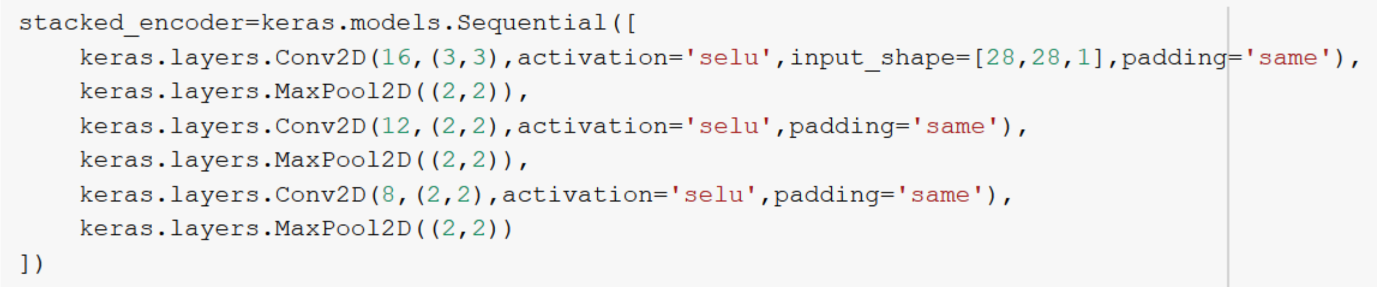
SOLUTION:

Adding Noise:

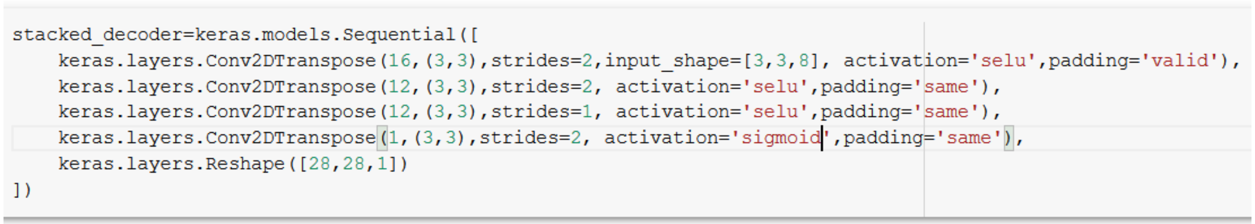


```
X_train = np.clip(X_train.astype('float32') / 255. + np.random.normal(loc=0.0,
scale=0.5, size=X train.shape),0.,1.)
```

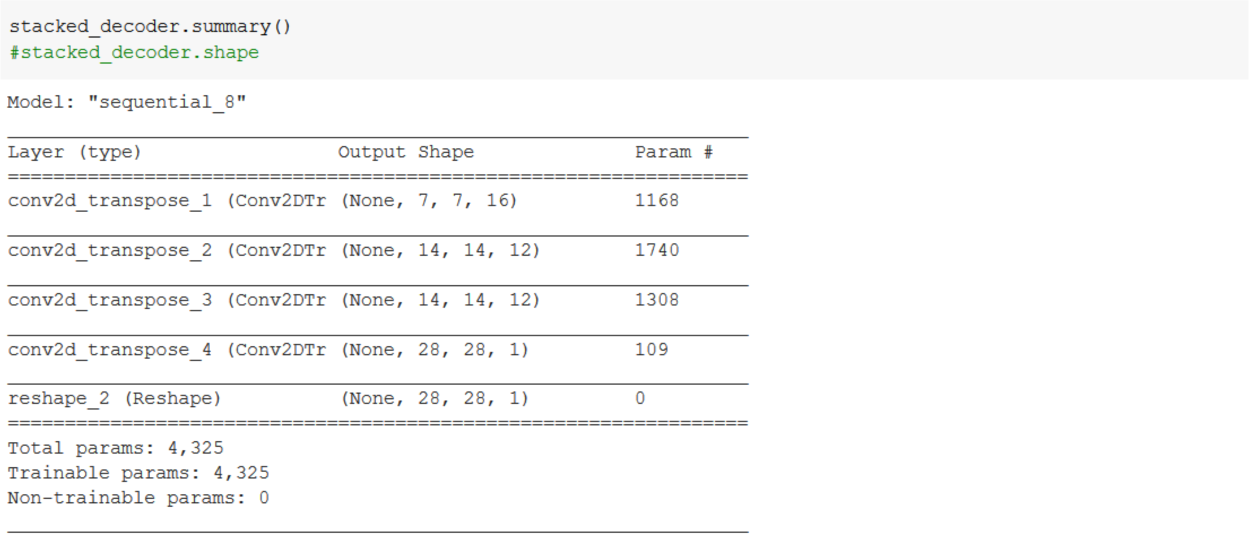
Architecture:



```
stacked_encoder=keras.models.Sequential([
    keras.layers.Conv2D(16,(3,3),activation='selu',input_shape=[28,28,1],padding='same'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Conv2D(12,(2,2),activation='selu',padding='same'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Conv2D(8,(2,2),activation='selu',padding='same'),
    keras.layers.MaxPool2D((2,2))
])
```

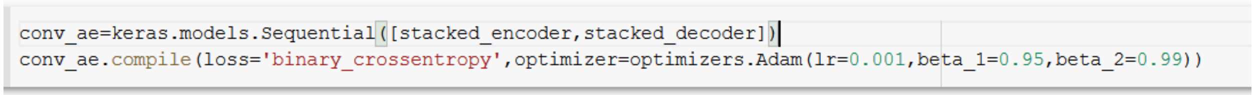


```
stacked_decoder=keras.models.Sequential([
    keras.layers.Conv2DTranspose(16,(3,3),strides=2,input_shape=[3,3,8], activation='selu',padding='valid'),
    keras.layers.Conv2DTranspose(12,(3,3),strides=2, activation='selu',padding='same'),
    keras.layers.Conv2DTranspose(12,(3,3),strides=1, activation='selu',padding='same'),
    keras.layers.Conv2DTranspose(1,(3,3),strides=2, activation='sigmoid',padding='same'),
    keras.layers.Reshape([28,28,1])
])
```



```
stacked_decoder.summary()
#stacked_decoder.shape
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_transpose_1 (Conv2DTr (None, 7, 7, 16)          1168
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 14, 14, 12)        1740
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 14, 14, 12)        1308
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 28, 28, 1)         109
_________________________________________________________________
reshape_2 (Reshape)          (None, 28, 28, 1)         0
=================================================================
Total params: 4,325
Trainable params: 4,325
Non-trainable params: 0
_________________________________________________________________
```



```
conv_ae=keras.models.Sequential([stacked_encoder,stacked_decoder])
conv_ae.compile(loss='binary_crossentropy',optimizer=optimizers.Adam(lr=0.001,beta_1=0.95,beta_2=0.99))
```

Results: