Assignment is below at the bottom

Video 13.1 https://www.youtube.com/watch?v=kIGHE7Cfe1s

Video 13.2 https://www.youtube.com/watch?v=Rm9bJcDd1KU

Video 13.3 https://youtu.be/6HjZk-3LsjE

In [3]:
```
from keras.callbacks import TensorBoard
from keras.layers import Input, Dense
from keras.models import Model
from keras.datasets import mnist
import numpy as np

(xtrain, ytrain), (xtest, ytest) = mnist.load_data()

xtrain = xtrain.astype('float32') / 255.
xtest = xtest.astype('float32') / 255.
xtrain = xtrain.reshape((len(xtrain), np.prod(xtrain.shape[1:])))
xtest = xtest.reshape((len(xtest), np.prod(xtest.shape[1:])))
xtrain.shape, xtest.shape
```

Out[3]:
```
((60000, 784), (10000, 784))
```

In [18]:
```
# this is the size of our encoded representations
encoding_dim = 4  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

# this is our input placeholder
x = input_img = Input(shape=(784,))
# "encoded" is the encoded representation of the input
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
encoded = Dense(encoding_dim, activation='relu')(x)


# "decoded" is the lossy reconstruction of the input
x = Dense(128, activation='relu')(encoded)
x = Dense(256, activation='relu')(x)
decoded = Dense(784, activation='sigmoid')(x)

# this model maps an input to its reconstruction
autoencoder = Model(input_img, decoded)

encoder = Model(input_img, encoded)

# create a placeholder for an encoded (32-dimensional) input
encoded_input = Input(shape=(encoding_dim,))
# retrieve the last layer of the autoencoder model
dcd1 = autoencoder.layers[-1]
dcd2 = autoencoder.layers[-2]
dcd3 = autoencoder.layers[-3]

# create the decoder model
decoder = Model(encoded_input, dcd1(dcd2(dcd3(encoded_input))))
```

In [19]:
```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

In [21]:
```
autoencoder.fit(xtrain, xtrain,
                epochs=5,
                batch_size=256,
                shuffle=True,
                validation_data=(xtest, xtest))
                #callbacks=[TensorBoard(log_dir='/tmp/autoencoder')])
```

```
Epoch 1/5
235/235 [==============================] - 8s 20ms/step - loss: 0.2575 - val_loss: 0.1977
Epoch 2/5
235/235 [==============================] - 5s 20ms/step - loss: 0.1830 - val_loss: 0.1737
Epoch 3/5
235/235 [==============================] - 3s 13ms/step - loss: 0.1706 - val_loss: 0.1671
Epoch 4/5
235/235 [==============================] - 3s 13ms/step - loss: 0.1655 - val_loss: 0.1633
Epoch 5/5
235/235 [==============================] - 3s 13ms/step - loss: 0.1622 - val_loss: 0.1612
```

Out[21]:
```
<keras.callbacks.History at 0x1c19c99e0a0>
```

In [38]:
```
encoded_imgs
```

Out[38]:

```
array([[11.943697 ,  9.005527 , 12.027234 , 32.89881  ],
       [23.76052  , 13.926956 ,  5.6552634,  8.942506 ],
       [35.62965  , 34.729908 , 24.666973 , 41.5047   ],
       ...,
       [ 5.3135986, 11.108302 , 14.398285 , 17.106884 ],
       [ 4.376413 , 19.419018 , 15.854642 , 11.992302 ],
       [ 7.41167  , 18.699078 , 30.420742 , 11.0364065]], dtype=float32)
```

In [52]:

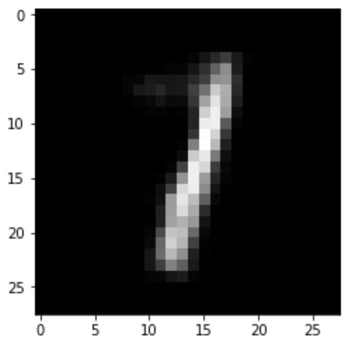
```
noise = np.random.normal(20,4, (4,4))
noise_preds = decoder.predict(noise)
```

In [55]:

```
plt.imshow(noise_preds[1].reshape(28,28))
```

Out[55]:

```
<matplotlib.image.AxesImage at 0x13bf35780>
```



In [41]:

```
np.max(encoded_imgs)
```

Out[41]:

```
54.59457
```

In [32]:

```
encoded_imgs = encoder.predict(xtest)
decoded_imgs = decoder.predict(encoded_imgs)
import matplotlib.pyplot as plt

n = 20  # how many digits we will display
plt.figure(figsize=(40, 4))
for i in range(n):
    # display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(xtest[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

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

In [33]:

```
encoded_imgs
```

Out[33]:


```
array([[11.943697 ,  9.005527 , 12.027234 , 32.89881  ],
       [23.76052  , 13.926956 ,  5.6552634,  8.942506 ],
       [35.62965  , 34.729908 , 24.666973 , 41.5047   ],
       ...,
       [ 5.3135986, 11.108302 , 14.398285 , 17.106884 ],
       [ 4.376413 , 19.419018 , 15.854642 , 11.992302 ],
       [ 7.41167  , 18.699078 , 30.420742 , 11.0364065]], dtype=float32)
```


In [26]:

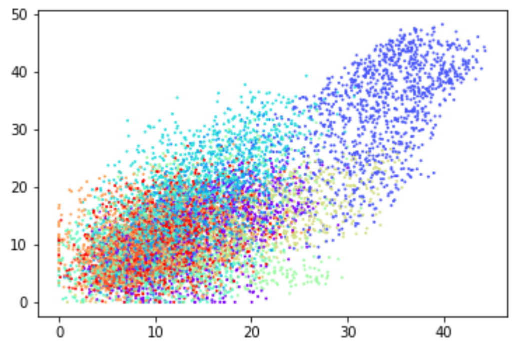
```
%matplotlib inline
```

In [34]:

```
plt.scatter(encoded_imgs[:,1], encoded_imgs[:,0], s=1, c=ytest, cmap='rainbow')
# plt.show()
```

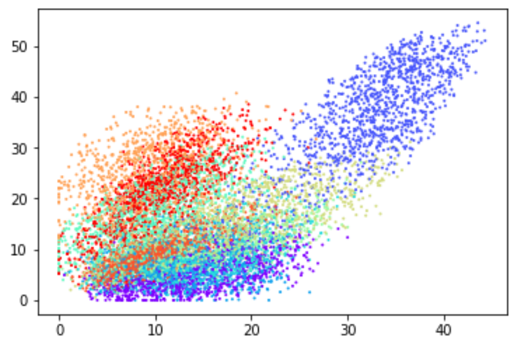
Out[34]:

```
<matplotlib.collections.PathCollection at 0x13c081978>
```



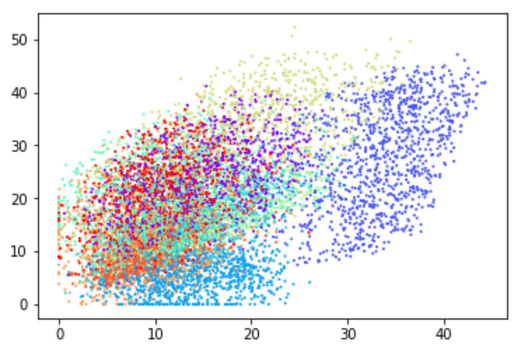
In [35]:
```
plt.scatter(encoded_imgs[:,1], encoded_imgs[:,3], s=1, c=ytest, cmap='rainbow')
# plt.show()
```

Out[35]:
```
<matplotlib.collections.PathCollection at 0x13b695e10>
```



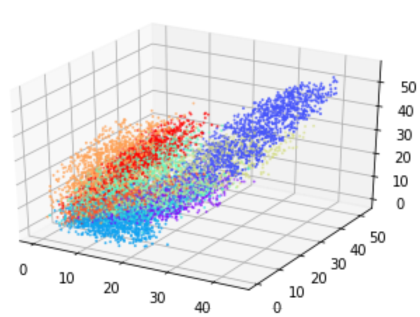
In [36]:
```
plt.scatter(encoded_imgs[:,1], encoded_imgs[:,2], s=1, c=ytest, cmap='rainbow')
# plt.show()
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x13b6eaf60>
```



In [37]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(encoded_imgs[:,1], encoded_imgs[:,2], encoded_imgs[:,3], c=ytest, cmap='rainbow', s=1)
```

Out[37]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x13c0e7da0>
```



Assignment

1. change the `encoding_dim` through various values (`range(2,18,2)` and save the loss you can get. Plot the 8 pairs of dimensions vs loss on a scatter plot

In [23]:
```
# dimensions list is out of order so that the second part of the assignment can be completed without recreating the model
dimensions = [2,4,6,10,12,14,16,8]
```

In [7]:
```
losses = []
for encoding_dim in dimensions:
    print(encoding_dim)
    x = input_img = Input(shape=(784,))
    # "encoded" is the encoded representation of the input
    x = Dense(256, activation='relu')(x)
    x = Dense(128, activation='relu')(x)
    encoded = Dense(encoding_dim, activation='relu')(x)


    # "decoded" is the lossy reconstruction of the input
    x = Dense(128, activation='relu')(encoded)
    x = Dense(256, activation='relu')(x)
    decoded = Dense(784, activation='sigmoid')(x)

    # this model maps an input to its reconstruction
    autoencoder = Model(input_img, decoded)

    encoder = Model(input_img, encoded)

    # create a placeholder for an encoded (32-dimensional) input
    encoded_input = Input(shape=(encoding_dim,))
    # retrieve the last layer of the autoencoder model
    dcd1 = autoencoder.layers[-1]
    dcd2 = autoencoder.layers[-2]
    dcd3 = autoencoder.layers[-3]

    # create the decoder model
    decoder = Model(encoded_input, dcd1(dcd2(dcd3(encoded_input))))

    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
    autoencoder.fit(xtrain, xtrain,
                    epochs=20,
                    batch_size=256,
                    shuffle=True,
                    validation_data=(xtest, xtest),
                    callbacks=[TensorBoard(log_dir='/tmp/autoencoder')]
                   )
    loss = autoencoder.evaluate(xtest,xtest,verbose=0)
    losses.append(loss)
```

```
2
Epoch 1/20
235/235 [==============================] - 3s 11ms/step - loss: 0.2838 - val_loss: 0.2491
Epoch 2/20
235/235 [==============================] - 2s 9ms/step - loss: 0.2271 - val_loss: 0.2152
Epoch 3/20
235/235 [==============================] - 2s 9ms/step - loss: 0.2116 - val_loss: 0.2060
Epoch 4/20
235/235 [==============================] - 2s 9ms/step - loss: 0.2029 - val_loss: 0.1987
Epoch 5/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1956 - val_loss: 0.1928
Epoch 6/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1917 - val_loss: 0.1902
Epoch 7/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1890 - val_loss: 0.1886
Epoch 8/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1872 - val_loss: 0.1866
Epoch 9/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1855 - val_loss: 0.1854
Epoch 10/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1845 - val_loss: 0.1840
Epoch 11/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1834 - val_loss: 0.1836
Epoch 12/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1823 - val_loss: 0.1823
Epoch 13/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1813 - val_loss: 0.1816
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1805 - val_loss: 0.1809
Epoch 15/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1799 - val_loss: 0.1806
Epoch 16/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1791 - val_loss: 0.1796
Epoch 17/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1783 - val_loss: 0.1796
Epoch 18/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1780 - val_loss: 0.1787
Epoch 19/20
235/235 [==============================] - 2s 8ms/step - loss: 0.1772 - val_loss: 0.1783
Epoch 20/20
235/235 [==============================] - 2s 8ms/step - loss: 0.1765 - val_loss: 0.1781
4
Epoch 1/20
235/235 [==============================] - 4s 11ms/step - loss: 0.2588 - val_loss: 0.1972
Epoch 2/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1842 - val_loss: 0.1742
Epoch 3/20
235/235 [==============================] - 2s 8ms/step - loss: 0.1706 - val_loss: 0.1676
Epoch 4/20
235/235 [==============================] - 2s 8ms/step - loss: 0.1654 - val_loss: 0.1635
Epoch 5/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1622 - val_loss: 0.1609
Epoch 6/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1598 - val_loss: 0.1593
Epoch 7/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1580 - val_loss: 0.1574
Epoch 8/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1564 - val_loss: 0.1564
Epoch 9/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1551 - val_loss: 0.1548
Epoch 10/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1540 - val_loss: 0.1540
Epoch 11/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1529 - val_loss: 0.1531
Epoch 12/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1520 - val_loss: 0.1524
Epoch 13/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1510 - val_loss: 0.1518
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1503 - val_loss: 0.1508
Epoch 15/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1496 - val_loss: 0.1503
Epoch 16/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1489 - val_loss: 0.1496
Epoch 17/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1482 - val_loss: 0.1492
Epoch 18/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1475 - val_loss: 0.1485
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1470 - val_loss: 0.1484
Epoch 20/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1465 - val_loss: 0.1479
6
Epoch 1/20
235/235 [==============================] - 4s 11ms/step - loss: 0.2616 - val_loss: 0.2074
Epoch 2/20
```

```
235/235 [==============================] - 2s 10ms/step - loss: 0.1779 - val_loss: 0.1633
Epoch 3/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1603 - val_loss: 0.1568
Epoch 4/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1549 - val_loss: 0.1526
Epoch 5/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1514 - val_loss: 0.1504
Epoch 6/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1491 - val_loss: 0.1483
Epoch 7/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1473 - val_loss: 0.1469
Epoch 8/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1459 - val_loss: 0.1454
Epoch 9/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1446 - val_loss: 0.1441
Epoch 10/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1436 - val_loss: 0.1432
Epoch 11/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1426 - val_loss: 0.1426
Epoch 12/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1418 - val_loss: 0.1419
Epoch 13/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1410 - val_loss: 0.1411
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1403 - val_loss: 0.1408
Epoch 15/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1397 - val_loss: 0.1401
Epoch 16/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1391 - val_loss: 0.1395
Epoch 17/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1385 - val_loss: 0.1393
Epoch 18/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1379 - val_loss: 0.1386
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1374 - val_loss: 0.1383
Epoch 20/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1369 - val_loss: 0.1381
10
Epoch 1/20
235/235 [==============================] - 3s 11ms/step - loss: 0.2491 - val_loss: 0.1786
Epoch 2/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1611 - val_loss: 0.1497
Epoch 3/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1464 - val_loss: 0.1415
Epoch 4/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1403 - val_loss: 0.1369
Epoch 5/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1365 - val_loss: 0.1340
Epoch 6/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1339 - val_loss: 0.1320
Epoch 7/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1320 - val_loss: 0.1305
Epoch 8/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1303 - val_loss: 0.1292
Epoch 9/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1289 - val_loss: 0.1282
Epoch 10/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1277 - val_loss: 0.1269
Epoch 11/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1264 - val_loss: 0.1251
Epoch 12/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1237 - val_loss: 0.1224
Epoch 13/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1218 - val_loss: 0.1213
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1207 - val_loss: 0.1204
Epoch 15/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1197 - val_loss: 0.1195
Epoch 16/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1189 - val_loss: 0.1188
Epoch 17/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1182 - val_loss: 0.1181
Epoch 18/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1175 - val_loss: 0.1180
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1170 - val_loss: 0.1173
Epoch 20/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1164 - val_loss: 0.1170
12
Epoch 1/20
235/235 [==============================] - 4s 11ms/step - loss: 0.2305 - val_loss: 0.1603
Epoch 2/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1508 - val_loss: 0.1424
Epoch 3/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1385 - val_loss: 0.1332
Epoch 4/20
```

```
235/235 [==============================] - 2s 10ms/step - loss: 0.1317 - val_loss: 0.1283
Epoch 5/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1269 - val_loss: 0.1237
Epoch 6/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1221 - val_loss: 0.1186
Epoch 7/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1181 - val_loss: 0.1159
Epoch 8/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1157 - val_loss: 0.1137
Epoch 9/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1138 - val_loss: 0.1121
Epoch 10/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1123 - val_loss: 0.1113
Epoch 11/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1109 - val_loss: 0.1101
Epoch 12/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1098 - val_loss: 0.1091
Epoch 13/20
235/235 [==============================] - 3s 12ms/step - loss: 0.1089 - val_loss: 0.1079
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1080 - val_loss: 0.1074
Epoch 15/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1072 - val_loss: 0.1065
Epoch 16/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1065 - val_loss: 0.1064
Epoch 17/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1059 - val_loss: 0.1058
Epoch 18/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1052 - val_loss: 0.1049
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1047 - val_loss: 0.1047
Epoch 20/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1042 - val_loss: 0.1042
14
Epoch 1/20
235/235 [==============================] - 3s 11ms/step - loss: 0.2317 - val_loss: 0.1588
Epoch 2/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1453 - val_loss: 0.1346
Epoch 3/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1310 - val_loss: 0.1240
Epoch 4/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1211 - val_loss: 0.1167
Epoch 5/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1153 - val_loss: 0.1120
Epoch 6/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1118 - val_loss: 0.1097
Epoch 7/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1094 - val_loss: 0.1076
Epoch 8/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1074 - val_loss: 0.1054
Epoch 9/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1059 - val_loss: 0.1041
Epoch 10/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1045 - val_loss: 0.1032
Epoch 11/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1033 - val_loss: 0.1020
Epoch 12/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1023 - val_loss: 0.1014
Epoch 13/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1015 - val_loss: 0.1006
Epoch 14/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1007 - val_loss: 0.0999
Epoch 15/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1001 - val_loss: 0.0995
Epoch 16/20
235/235 [==============================] - 2s 9ms/step - loss: 0.0995 - val_loss: 0.0989
Epoch 17/20
235/235 [==============================] - 2s 9ms/step - loss: 0.0989 - val_loss: 0.0985
Epoch 18/20
235/235 [==============================] - 2s 9ms/step - loss: 0.0984 - val_loss: 0.0979
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0980 - val_loss: 0.0976
Epoch 20/20
235/235 [==============================] - 2s 9ms/step - loss: 0.0976 - val_loss: 0.0972
16
Epoch 1/20
235/235 [==============================] - 3s 10ms/step - loss: 0.2264 - val_loss: 0.1510
Epoch 2/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1385 - val_loss: 0.1282
Epoch 3/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1245 - val_loss: 0.1184
Epoch 4/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1170 - val_loss: 0.1135
Epoch 5/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1130 - val_loss: 0.1105
Epoch 6/20
```

```
235/235 [==============================] - 2s 10ms/step - loss: 0.1104 - val_loss: 0.1080
Epoch 7/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1078 - val_loss: 0.1057
Epoch 8/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1056 - val_loss: 0.1039
Epoch 9/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1039 - val_loss: 0.1025
Epoch 10/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1026 - val_loss: 0.1013
Epoch 11/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1016 - val_loss: 0.1007
Epoch 12/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1007 - val_loss: 0.1000
Epoch 13/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0999 - val_loss: 0.0991
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0992 - val_loss: 0.0985
Epoch 15/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0985 - val_loss: 0.0981
Epoch 16/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0980 - val_loss: 0.0976
Epoch 17/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0975 - val_loss: 0.0972
Epoch 18/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0971 - val_loss: 0.0966
Epoch 19/20
235/235 [==============================] - 3s 11ms/step - loss: 0.0966 - val_loss: 0.0966
Epoch 20/20
235/235 [==============================] - 2s 10ms/step - loss: 0.0961 - val_loss: 0.0963
8
Epoch 1/20
235/235 [==============================] - 4s 12ms/step - loss: 0.2433 - val_loss: 0.1774
Epoch 2/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1596 - val_loss: 0.1441
Epoch 3/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1397 - val_loss: 0.1348
Epoch 4/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1333 - val_loss: 0.1306
Epoch 5/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1296 - val_loss: 0.1274
Epoch 6/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1271 - val_loss: 0.1255
Epoch 7/20
235/235 [==============================] - 2s 9ms/step - loss: 0.1251 - val_loss: 0.1236
Epoch 8/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1236 - val_loss: 0.1225
Epoch 9/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1224 - val_loss: 0.1214
Epoch 10/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1212 - val_loss: 0.1206
Epoch 11/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1203 - val_loss: 0.1195
Epoch 12/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1195 - val_loss: 0.1187
Epoch 13/20
235/235 [==============================] - 2s 11ms/step - loss: 0.1186 - val_loss: 0.1181
Epoch 14/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1180 - val_loss: 0.1177
Epoch 15/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1174 - val_loss: 0.1171
Epoch 16/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1167 - val_loss: 0.1166
Epoch 17/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1162 - val_loss: 0.1163
Epoch 18/20
235/235 [==============================] - 3s 11ms/step - loss: 0.1157 - val_loss: 0.1158
Epoch 19/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1152 - val_loss: 0.1155
Epoch 20/20
235/235 [==============================] - 2s 10ms/step - loss: 0.1148 - val_loss: 0.1149
```

In [8]:

```python
losses
```

Out[8]:

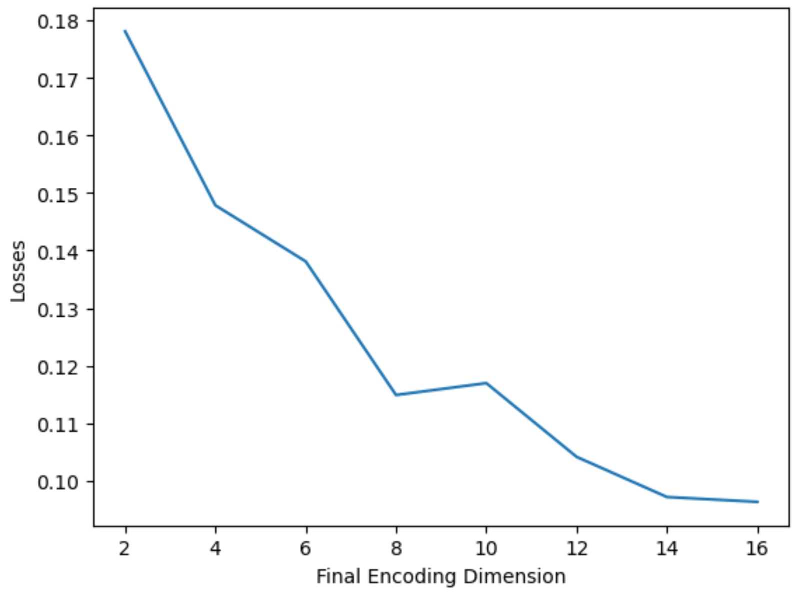
```
[0.17809730768203735,
 0.14785830676555634,
 0.1381126046180725,
 0.11696751415729523,
 0.10415196418762207,
 0.09717812389135361,
 0.09633705019950867,
 0.11490143090486526]
```

In [25]:

```python
import matplotlib.pyplot as  plt
%matplotlib inline

dimensions, losses = zip(*sorted(zip(dimensions, losses)))
```

```python
plt.figure()
plt.plot(dimensions,losses)
plt.xlabel('Final Encoding Dimension')
plt.ylabel('Losses')
plt.show()
```



1. ***After*** training an autoencoder with `encoding_dim=8`, apply noise (like the previous assignment) to *only* the input of the trained autoencoder (not the output). The output images should be without noise.

Print a few noisy images along with the output images to show they don't have noise.

In [10]:
```python
noise = xtest + np.random.normal(0, .1, size=(10000, 784))
```

In [11]:
```python
encoded_imgs = encoder.predict(noise)
decoded_imgs = decoder.predict(encoded_imgs)
import matplotlib.pyplot as plt

n = 20  # how many digits we will display
plt.figure(figsize=(40, 4))
for i in range(n):
    # display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(noise[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

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

```
313/313 [==============================] - 1s 3ms/step
313/313 [==============================] - 1s 1ms/step
```