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 [1]:

```
import numpy as np
import tensorflow as tf
```

In [10]:

```
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
import matplotlib.pyplot as plt

(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[10]:

```
((60000, 784), (10000, 784))
```

In [11]:

```
# this is the size of our encoded representations
encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the inpu

# 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]
```

```python

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

In [12]:

```python
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

In [13]:

```python
autoencoder.fit(xtrain, xtrain,
                epochs=10,
                batch_size=256,
                shuffle=True,
                validation_data=(xtest, xtest))
```

```
Epoch 1/10
  9/235 [>.............................] - ETA: 1s - loss: 0.6935
2022-04-20 19:05:16.565500: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
235/235 [==============================] - 2s 7ms/step - loss: 0.6934 - val_lo
ss: 0.6934
Epoch 2/10
  1/235 [..............................] - ETA: 1s - loss: 0.6934
2022-04-20 19:05:18.093062: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
235/235 [==============================] - 1s 6ms/step - loss: 0.6933 - val_lo
ss: 0.6933
Epoch 3/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6932 - val_lo
ss: 0.6932
Epoch 4/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6931 - val_lo
ss: 0.6931
Epoch 5/10
235/235 [==============================] - 2s 6ms/step - loss: 0.6930 - val_lo
ss: 0.6930
Epoch 6/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6929 - val_lo
ss: 0.6929
Epoch 7/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6928 - val_lo
ss: 0.6928
Epoch 8/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6927 - val_lo
ss: 0.6927
Epoch 9/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6926 - val_lo
ss: 0.6926
Epoch 10/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6925 - val_lo
ss: 0.6925
```

Out[13]:

```
<keras.callbacks.History at 0x2b8d6afa0>
```

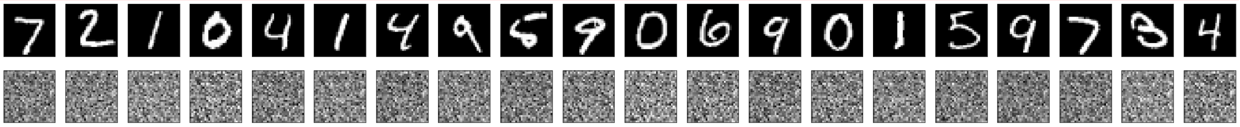
In [7]:

```python
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()
```

```
2022-04-20 19:03:50.025844: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:03:50.377605: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```



In [15]:

```
noise = np.random.normal(20,4, (4,32))
noise_preds = decoder.predict(noise)
```

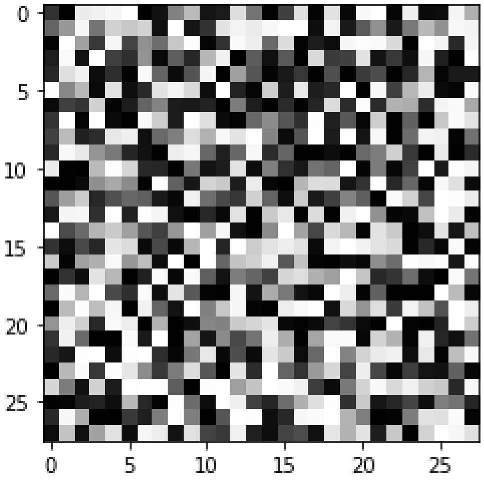
```
2022-04-20 19:05:58.056183: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [16]:

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

Out[16]:

```
<matplotlib.image.AxesImage at 0x2c44b9d00>
```



In [17]:

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

Out[17]:

```
1.3580321
```

In [18]:

```
encoded_imgs
```

Out[18]:

```
array([[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 2.3225811e-01,
         0.0000000e+00, 2.5551948e-01],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.7010622e-01,
         4.1145870e-01, 9.5288701e-02],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 4.2720996e-02,
         0.0000000e+00, 3.2465896e-01],
        ...,
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
         4.1268975e-02, 3.0748990e-01],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 9.7543161e-05,
         9.6587375e-02, 2.7234137e-01],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 3.0143291e-01,
         3.2321402e-01, 5.5349982e-01]], dtype=float32)
```

In [19]:

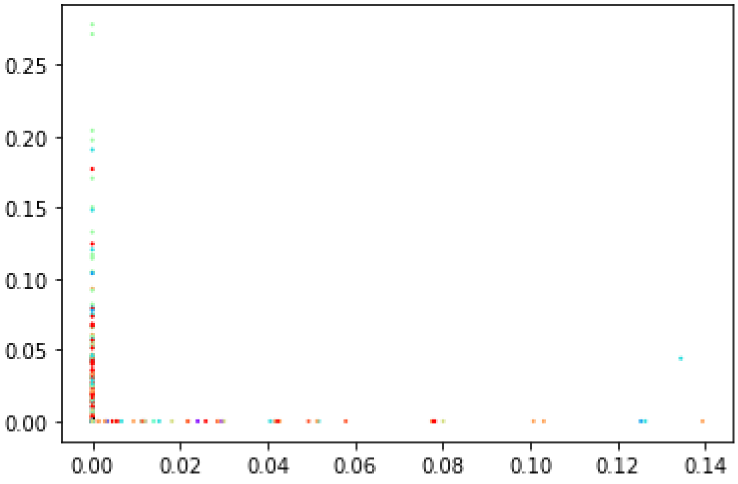
```
%matplotlib inline
```

In [20]:

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

Out[20]:

```
<matplotlib.collections.PathCollection at 0x2b8dfd2e0>
```

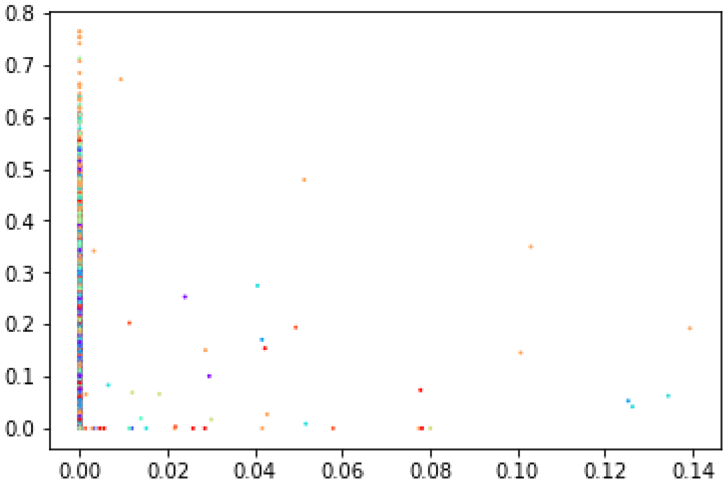


In [21]:

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

Out[21]:

```
<matplotlib.collections.PathCollection at 0x2cffb7f40>
```



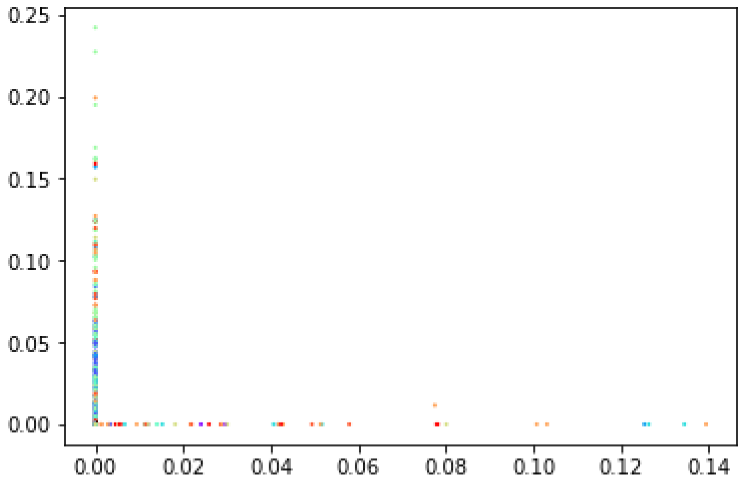
In [22]:

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

```
# plt.show()
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x2c4456070>
```

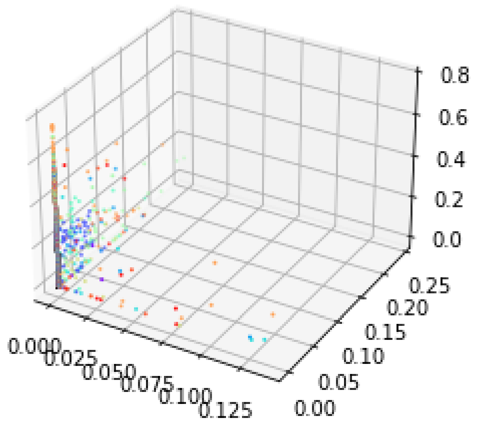


In [23]:

```
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, cm
```

Out[23]:

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x17f7a8130>
```



In [ ]:

```

```

Assignment

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

In [26]:

```
losses = []
encoding_dim = (range(2, 18, 2))

encoded_imgs = encoder.predict(xtest)
decoded_imgs = decoder.predict(encoded_imgs)
```

```
for encoding_dim in encoding_dim:

    x = input_img = Input(shape=(784,))

    x = Dense(256, activation='relu')(x)
    x = Dense(128, activation='relu')(x)
    encoded = Dense(encoding_dim, activation='relu')(x)

    x = Dense(128, activation='relu')(encoded)
    x = Dense(256, activation='relu')(x)
    decoded = Dense(784, activation='sigmoid')(x)

    autoencoder = Model(input_img, decoded)
    encoder = Model(input_img, encoded)

    encoded_input = Input(shape=(encoding_dim,))

    dcd1 = autoencoder.layers[-1]
    dcd2 = autoencoder.layers[-2]
    dcd3 = autoencoder.layers[-3]

    decoder = Model(encoded_input, dcd1(dcd2(dcd3(encoded_input))))

    autoencoder.compile(optimizer=tf.keras.optimizers.Adadelta(learning_rate=1)

    model = autoencoder.fit(xtrain, xtrain,
                epochs=100,
                batch_size=256,
                shuffle=True,
                verbose=0,
                validation_data=(xtest, xtest))



    n = 20
    plt.figure(figsize=(60, 4))
    for i in range(n):
        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)

        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()

    loss = np.min(model.history["loss"])
    losses.append(loss)
2022-04-20 19:10:32.098635: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:10:32.449981: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:10:33.144721: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:10:34.562938: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:12:42.101737: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:12:43.612989: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:14:52.686949: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:14:54.034494: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

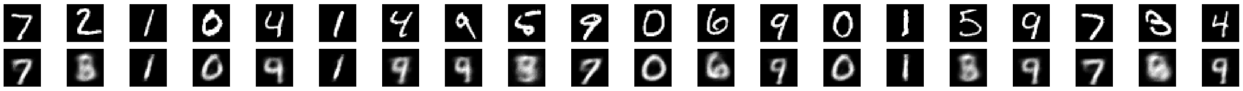
```
2022-04-20 19:17:04.097148: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:17:05.851535: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:19:16.498731: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:19:17.919002: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:21:26.398818: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:21:27.876831: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:23:33.971330: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:23:35.378756: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
2022-04-20 19:25:41.635916: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:25:43.075658: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```



In [27]:

```
encoding_dim = np.asarray(range(2, 18, 2))
losses = np.asarray(losses)
```

In [28]:

```
encoding_dim.shape, losses.shape
```

Out[28]:

```
((8,), (8,))
```

In [29]:

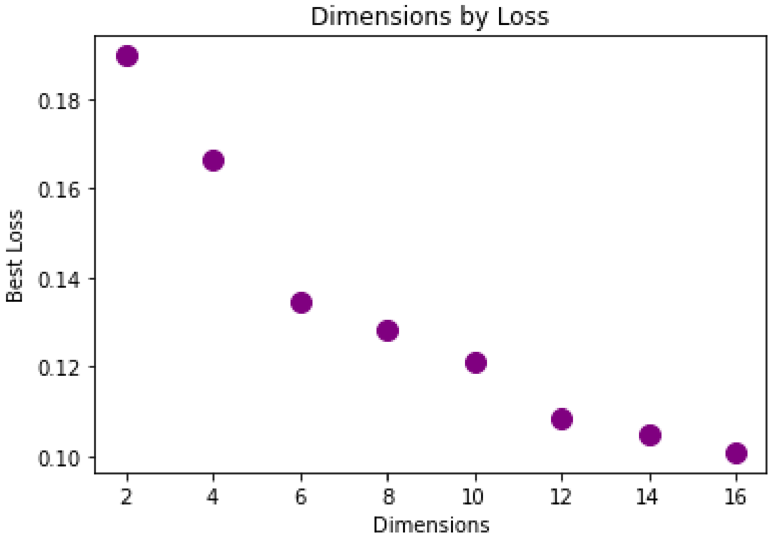
```
encoded_imgs
```

Out[29]:

```
array([[ 4.2861347, 22.637783 ,  0.       , 18.613836 ],
       [ 9.403811 ,  7.13039  ,  0.       ,  6.5799375],
       [24.402075 , 40.785385 ,  0.       , 13.966941 ],
       ...,
       [ 4.916314 , 16.677427 ,  0.       ,  8.452135 ],
       [ 6.9526405,  7.2563763,  0.       ,  0.82639  ],
       [20.648458 ,  9.083031 ,  0.       , 11.402394 ]], dtype=float32)
```

In [30]:

```
plt.scatter(encoding_dim, losses, s=100, c = "purple")
plt.title("Dimensions by Loss")
plt.xlabel("Dimensions")
plt.ylabel("Best Loss")
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.

In [31]:

```
(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[31]:
```
((60000, 784), (10000, 784))
```

In [32]:
```
encoding_dim = 8

x = input_img = Input(shape=(784,))

x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
encoded = Dense(encoding_dim, activation='relu')(x)

x = Dense(128, activation='relu')(encoded)
x = Dense(256, activation='relu')(x)
decoded = Dense(784, activation='sigmoid')(x)

autoencoder = Model(input_img, decoded)

encoder = Model(input_img, encoded)

encoded_input = Input(shape=(encoding_dim,))

dcd1 = autoencoder.layers[-1]
dcd2 = autoencoder.layers[-2]
dcd3 = autoencoder.layers[-3]

decoder = Model(encoded_input, dcd1(dcd2(dcd3(encoded_input))))
```

In [34]:
```
autoencoder.compile(optimizer=tf.keras.optimizers.Adadelta(learning_rate=1), lc
autoencoder.fit(xtrain, xtrain,
                epochs=100,
                batch_size=256,
                shuffle=True,
                verbose = 0,
                validation_data=(xtest, xtest))
```

```
2022-04-20 19:30:00.776915: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:30:02.122866: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

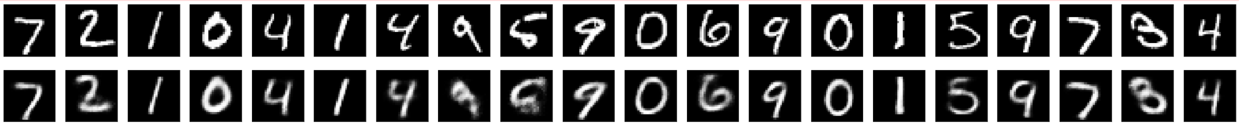
Out[34]:
```
<keras.callbacks.History at 0x2eb1c69d0>
```

In [35]:
```
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()
```

```
2022-04-20 19:33:05.093858: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-20 19:33:05.521201: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```



In [80]:

```
print(np.min(encoded_imgs))
print(np.max(encoded_imgs))
```

```
0.0
42.548054
```

In [134…

```
noise = np.random.normal(20,5, (10000,784))
noise_preds = autoencoder.predict(noise)
```

In [135…

```
noise_preds.shape
```

Out[135]:

```
(10000, 784)
```

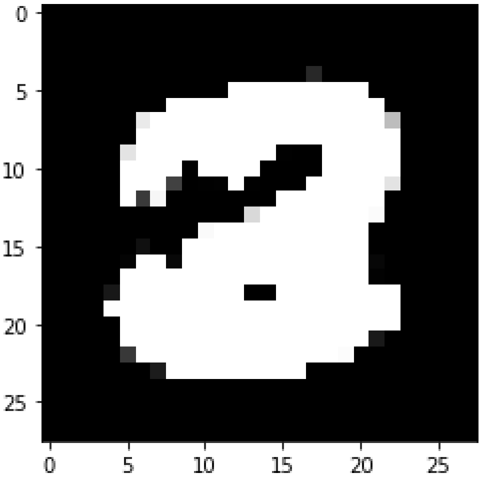
Print a few noisy images along with the output images to show they don't have noise.

In [136…

```
plt.imshow(noise_preds[0].reshape(28,28))
```

Out[136]:

```
<matplotlib.image.AxesImage at 0x162164880>
```



In [137…

```
decoded_imgs.shape
```

Out[137]:

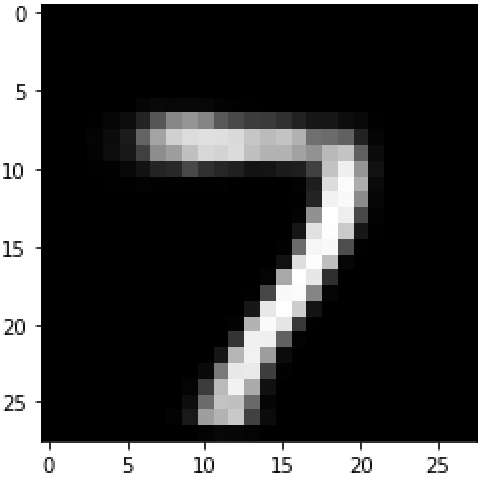
```
(10000, 784)
```

In [138…

```
plt.imshow(decoded_imgs[0].reshape(28,28))
```

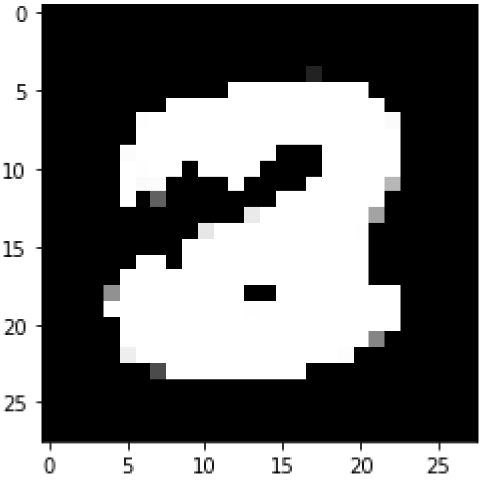
Out[138]:

```
<matplotlib.image.AxesImage at 0x161f26730>
```



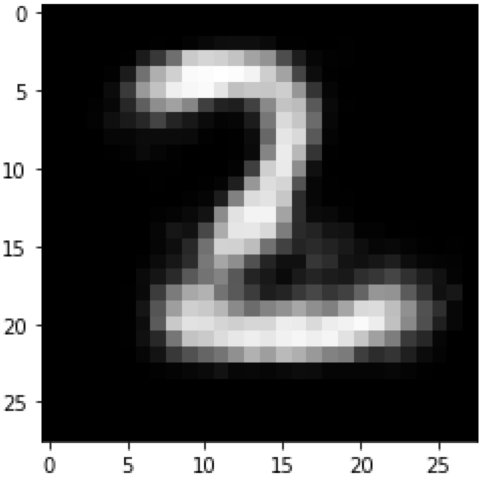
In [55]:
```
plt.imshow(noise_preds[1].reshape(28, 28))
```

Out[55]:
```
<matplotlib.image.AxesImage at 0x167405670>
```



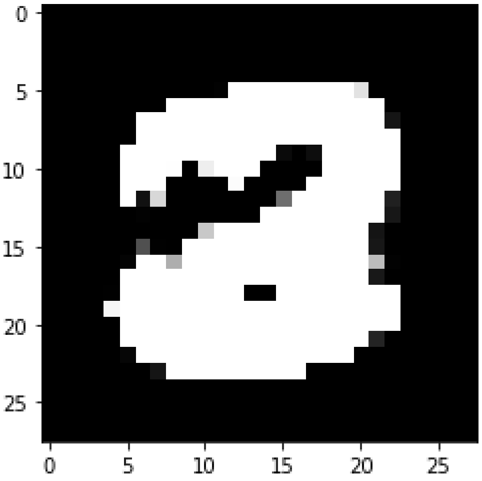
In [131…
```
plt.imshow(decoded_imgs[1].reshape(28,28))
```

Out[131]:
```
<matplotlib.image.AxesImage at 0x161eab850>
```



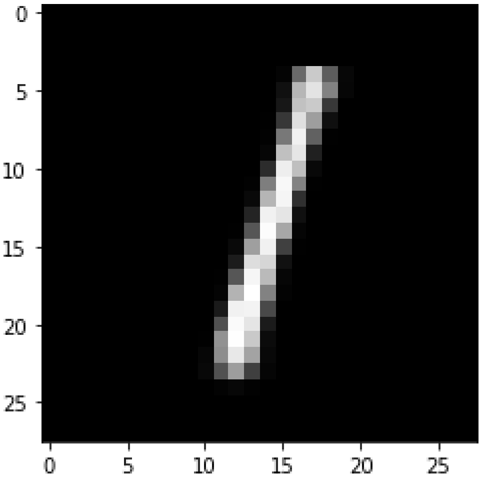
In [59]:
```
plt.imshow(noise_preds[2].reshape(28, 28))
```

Out[59]:
```
<matplotlib.image.AxesImage at 0x2cb277f70>
```



```
In [132… plt.imshow(decoded_imgs[2].reshape(28,28))
```

```
Out[132]: <matplotlib.image.AxesImage at 0x161ed7e20>
```



```
In [ ]:
```