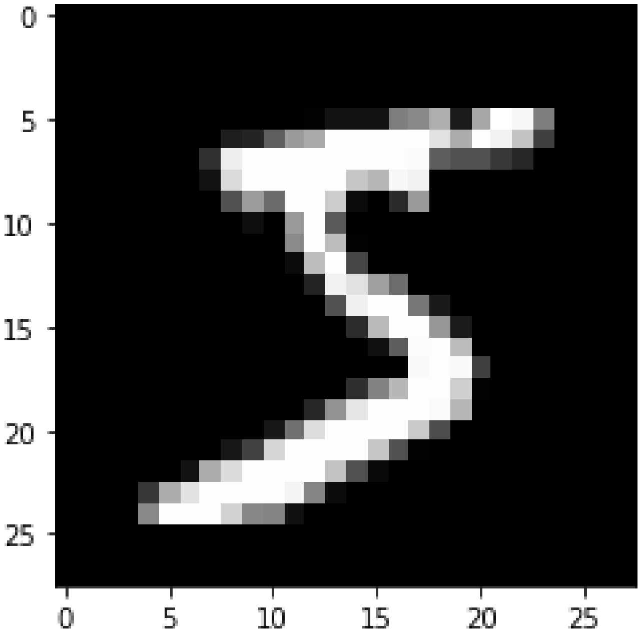
encoder

October 11, 2021

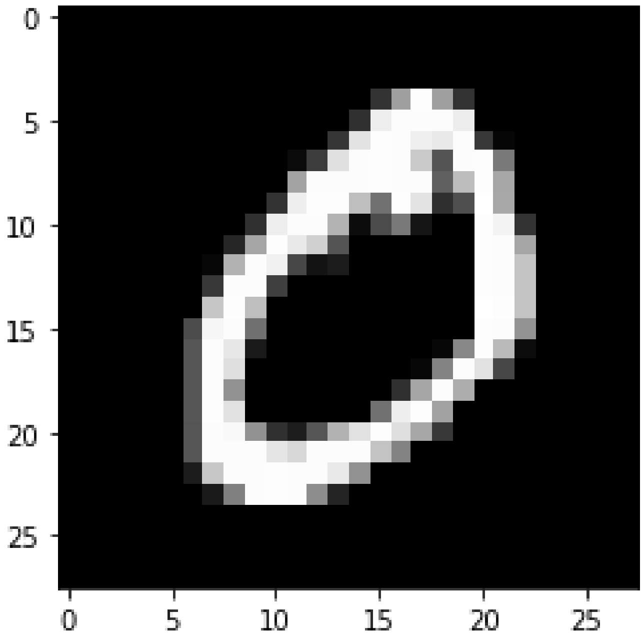
[2]:
```
# Github: https:github.com/stevemats

import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import cv2
import numpy as np

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()  #␣
 ↪loading "mnist" training dataset

plt.imshow(x_train[0], cmap="gray")
```

[2]:
```
<matplotlib.image.AxesImage at 0x1a896a39b80>
```



[3]:
```
plt.imshow(x_train[1], cmap="gray")
```

```
[3]: <matplotlib.image.AxesImage at 0x1a892ea4130>
```



```
[4]: #compression so that data is 28*28px
     x_train[0]
```

```
[4]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
             18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
            253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
            253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
            253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
            205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
             90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
            190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
            253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
            241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
            148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
            253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
            253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
            195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
             11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0]], dtype=uint8)
```

[5]:
```
x_train[0].shape
```

[5]:
```
(28, 28)
```

[6]:
```
28*28
# Below value is a result of the 28*28 px values equivalent to a total no. of␣
 ↪unique features
```

[6]:
```
784
```

[7]:
```
encoder_input = keras.Input(shape=(28, 28, 1), name='img') # Starts encoder
x = keras.layers.Flatten()(encoder_input) #flatten img so it can be used with␣
 ↪dense layers
encoder_output = keras.layers.Dense(64, activation="relu")(x) # compression␣
 ↪after flatten

encoder = keras.Model(encoder_input, encoder_output, name='encoder')


decoder_input = keras.layers.Dense(64, activation="relu")(encoder_output) #␣
 ↪starts decoder
decoder_output = keras.layers.Reshape((28, 28, 1))(x)
opt = tf.keras.optimizers.Adam(lr=0.001, decay=1e-6) # setting an optimizer
#now combining encoder with decoder into a singular "autoencoder" model
autoencoder = keras.Model(encoder_input, decoder_output, name='autoencoder')
autoencoder.summary() #making sure theirs no errors
```

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
img (InputLayer)             [(None, 28, 28, 1)]       0
_________________________________________________________________
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
reshape (Reshape)            (None, 28, 28, 1)         0
=================================================================
Total params: 0
Trainable params: 0
Non-trainable params: 0
_________________________________________________________________
```

[8]:

```python
autoencoder.compile(opt, loss='mse') #compiling our model with the optimizer and a loss metric
```

[9]:

```python
# Training & saving the model each time
epochs=3

for epoch in range(epochs):

    history = autoencoder.fit(
      x_train,
      x_train,
      epochs=1,
      batch_size=32, validation_split=0.10
        )
    autoencoder.save(f"models/AE-{epoch+1}.model")
```

```
1688/1688 [==============================] - 1s 625us/step - loss: 0.0000e+00 -
val_loss: 0.0000e+00
WARNING:tensorflow:From C:\Users\hyi\anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111:
Model.state_updates (from tensorflow.python.keras.engine.training) is deprecated
and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
WARNING:tensorflow:From C:\Users\hyi\anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111: Layer.updates
(from tensorflow.python.keras.engine.base_layer) is deprecated and will be
removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
INFO:tensorflow:Assets written to: models/AE-1.model\assets
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0000e+00 -
val_loss: 0.0000e+00
INFO:tensorflow:Assets written to: models/AE-2.model\assets
1688/1688 [==============================] - 1s 629us/step - loss: 0.0000e+00 -
```

```
val_loss: 0.0000e+00
INFO:tensorflow:Assets written to: models/AE-3.model\assets
```

[10]:

```
example = encoder.predict([ x_test[0].reshape(-1, 28, 28, 1) ])

print(example[0].shape)
```

```
(64,)
```

[11]:

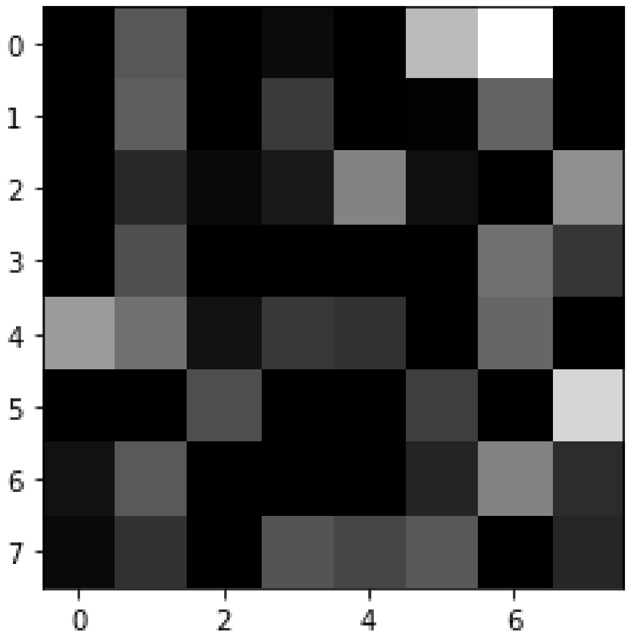
```
print(example[0])
```

```
[  0.          80.397766    0.          10.889713    0.         170.58217
 234.40257     0.          0.          86.214554    0.          53.194595
   0.           2.7461119  90.77078     0.          0.          37.25658
   9.770069    23.35424   120.705864   14.33755     0.         131.29291
   0.          73.125694    0.           0.          0.           0.
 102.702805    50.34705   143.04616   103.356865   15.875067    52.053226
  46.634453     0.         93.51317     0.          0.           0.
  71.6615       0.          0.          57.48574     0.         196.76404
  16.36266     83.08249     0.           0.          0.          33.257576
 120.1598      40.619038    9.068319   46.276497    0.          77.801956
  64.503365    81.07309     0.          35.08146  ]
```

[12]:

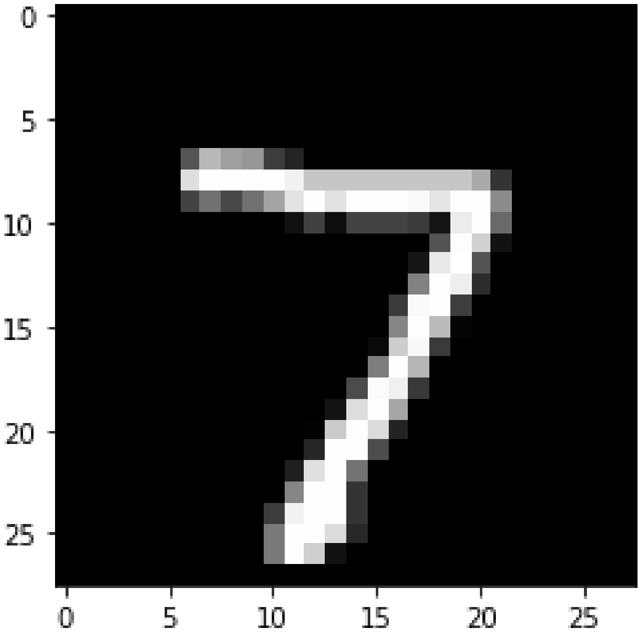
```
plt.imshow(example[0].reshape((8,8)), cmap="gray") # visualizing an 8*8 vector␣
 ↪of 64 values
```

[12]: <matplotlib.image.AxesImage at 0x1a892ecd9a0>



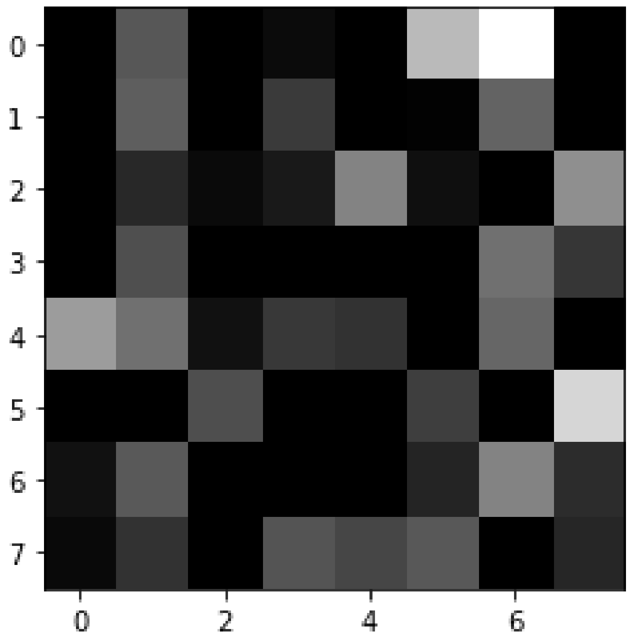
```
[13]: plt.imshow(x_test[0], cmap="gray")
```

```
[13]: <matplotlib.image.AxesImage at 0x1a89341c0d0>
```



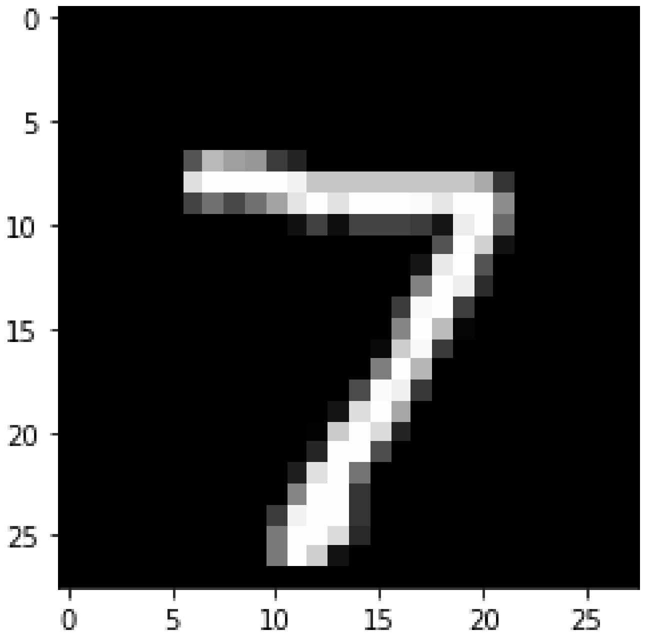
```
[14]: plt.imshow(example[0].reshape((8,8)), cmap="gray") # How the above looks after␣
      ↪going through our autoencoder
```

```
[14]: <matplotlib.image.AxesImage at 0x1a8944b3c70>
```



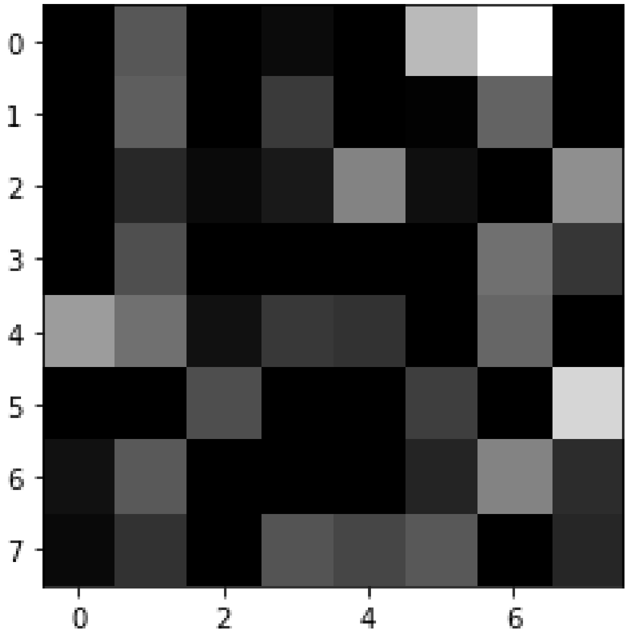
```
[15]: ae_out = autoencoder.predict([ x_test[0].reshape(-1, 28, 28, 1) ])
      img = ae_out[0]  # predict is done on a vector, and returns a vector, even if␣
       ↪its just 1 element, so we still need to grab the 0th
      plt.imshow(ae_out[0], cmap="gray")
```

```
[15]: <matplotlib.image.AxesImage at 0x1a89453f1c0>
```



[16]:
```
plt.imshow(example[0].reshape((8,8)), cmap="gray")
```

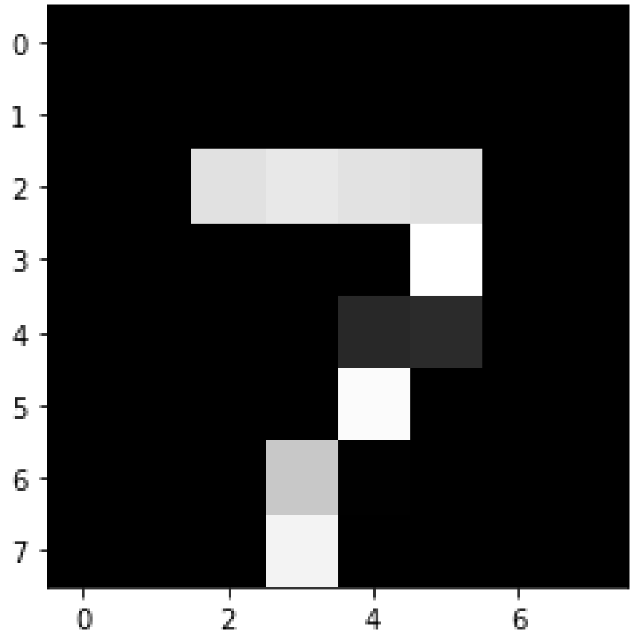
[16]:
```
<matplotlib.image.AxesImage at 0x1a894562e20>
```



[17]:
```
# The idea  behind autoencoders is in data simplification
for d in x_test[:5]:

    ae_out = autoencoder.predict([ d.reshape(-1, 28, 28, 1) ])
    img = ae_out[0]

    cv2.imshow("decoded",img)
    cv2.imshow("original",np.array(d))
    cv2.waitKey(1000)  # wait 1000ms = 1 sec, and then show the next
```

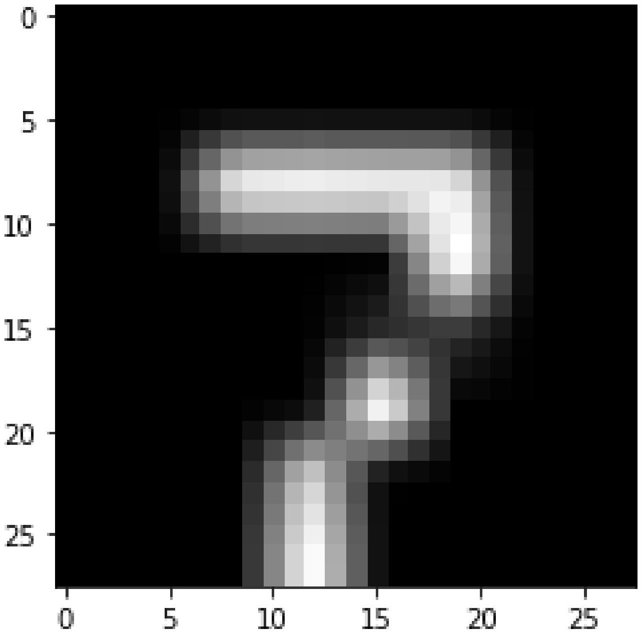
[18]:
```
smaller = cv2.resize(x_test[0], (8,8))
back_to_original = cv2.resize(smaller, (28,28))
plt.imshow(smaller, cmap="gray")
```

[18]:
```
<matplotlib.image.AxesImage at 0x1a8948dd490>
```



[19]:
```
plt.imshow(back_to_original, cmap="gray")
```

[19]:
```
<matplotlib.image.AxesImage at 0x1a894890df0>
```



[ ]:

```
```