```
from keras.datasets import mnist
from keras.layers import Input, Dense
from keras.models import Model
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline


(X_train, _), (X_test, _) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
X_train = X_train.astype('float32')/255
X_test = X_test.astype('float32')/255


X_train.shape
```

```
(60000, 28, 28)
```

```
X_train.shape[1:]
```

```
(28, 28)
```

```
#data should always be of the format “(Number of data points, data point dimension)”. In t
X_train = X_train.reshape(len(X_train), np.prod(X_train.shape[1:]))
X_test = X_test.reshape(len(X_test), np.prod(X_test.shape[1:]))


print(X_train.shape)
print(X_test.shape)
```

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

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


encoded = Dense(units=32, activation='relu')(input_img)


decoded = Dense(units=784, activation='sigmoid')(encoded)


autoencoder=Model(input_img, decoded)


autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 32)                25120
_________________________________________________________________
dense_1 (Dense)              (None, 784)               25872
=================================================================
Total params: 50,992
Trainable params: 50,992
Non-trainable params: 0
_________________________________________________________________
```

```
encoder = Model(input_img, encoded)
```

```
encoder.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 32)                25120
=================================================================
Total params: 25,120
Trainable params: 25,120
Non-trainable params: 0
_________________________________________________________________
```

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
autoencoder.fit(X_train, X_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(X_test, X_test))
```

```
Epoch 1/50
235/235 [==============================] - 2s 7ms/step - loss: 0.2793 - accuracy: 0
Epoch 2/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1738 - accuracy: 0
Epoch 3/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1426 - accuracy: 0
Epoch 4/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1273 - accuracy: 0
Epoch 5/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1168 - accuracy: 0
Epoch 6/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1093 - accuracy: 0
Epoch 7/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1043 - accuracy: 0
Epoch 8/50
```

```
235/235 [==============================] - 1s 5ms/step - loss: 0.1009 - accuracy: 0
Epoch 9/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0985 - accuracy: 0
Epoch 10/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0970 - accuracy: 0
Epoch 11/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0959 - accuracy: 0
Epoch 12/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0953 - accuracy: 0
Epoch 13/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0948 - accuracy: 0
Epoch 14/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0945 - accuracy: 0
Epoch 15/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0942 - accuracy: 0
Epoch 16/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0940 - accuracy: 0
Epoch 17/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0938 - accuracy: 0
Epoch 18/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0937 - accuracy: 0
Epoch 19/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0936 - accuracy: 0
Epoch 20/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0935 - accuracy: 0
Epoch 21/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0934 - accuracy: 0
Epoch 22/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0934 - accuracy: 0
Epoch 23/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0933 - accuracy: 0
Epoch 24/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0933 - accuracy: 0
Epoch 25/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0932 - accuracy: 0
Epoch 26/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0932 - accuracy: 0
Epoch 27/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0931 - accuracy: 0
Epoch 28/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0931 - accuracy: 0
Epoch 29/50
```

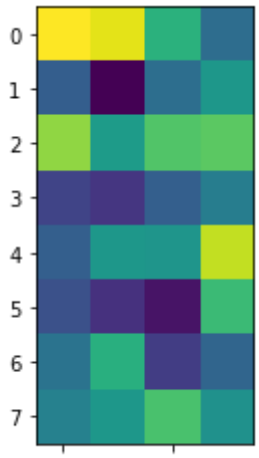
```
encoded_imgs = encoder.predict(X_test)
```

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

```
<matplotlib.image.AxesImage at 0x7fc0aa862d90>
```


```
plt.imshow(encoded_imgs[0].reshape(8,4))
```

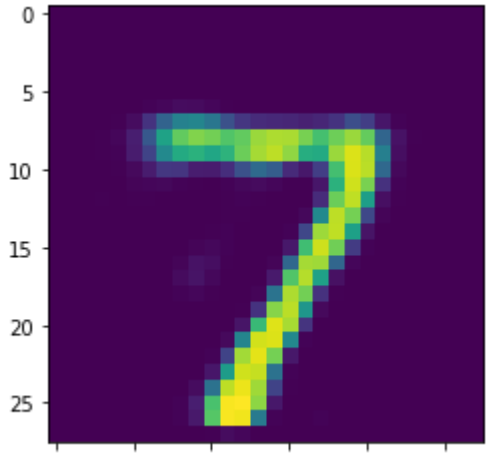
```
<matplotlib.image.AxesImage at 0x7fc0aeef8690>
```




```
predicted = autoencoder.predict(X_test)
```

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

```
<matplotlib.image.AxesImage at 0x7fc0aa8d8b50>
```



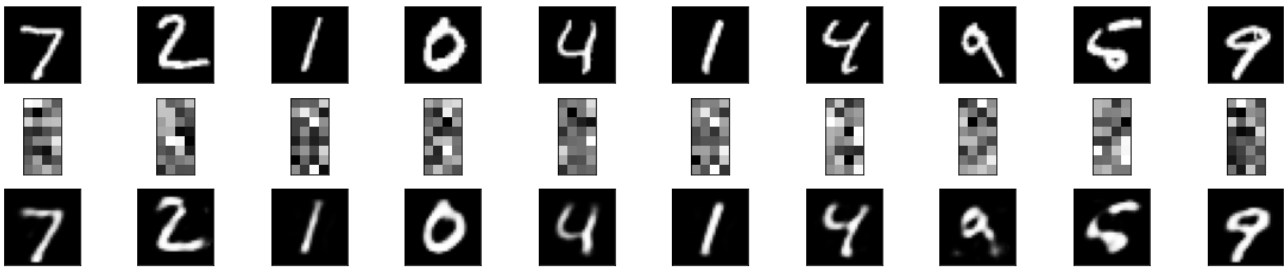

```
plt.figure(figsize=(40, 4))
for i in range(10):
    # display original
    ax = plt.subplot(3, 20, i + 1)
    plt.imshow(X_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # display encoded image
```

```
        ax = plt.subplot(3, 20, i + 1 + 20)
        plt.imshow(encoded_imgs[i].reshape(8,4))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
        # display reconstruction
        ax = plt.subplot(3, 20, 2*20 +i+ 1)
        plt.imshow(predicted[i].reshape(28, 28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)



    plt.show()
```



Colab paid products - Cancel contracts here