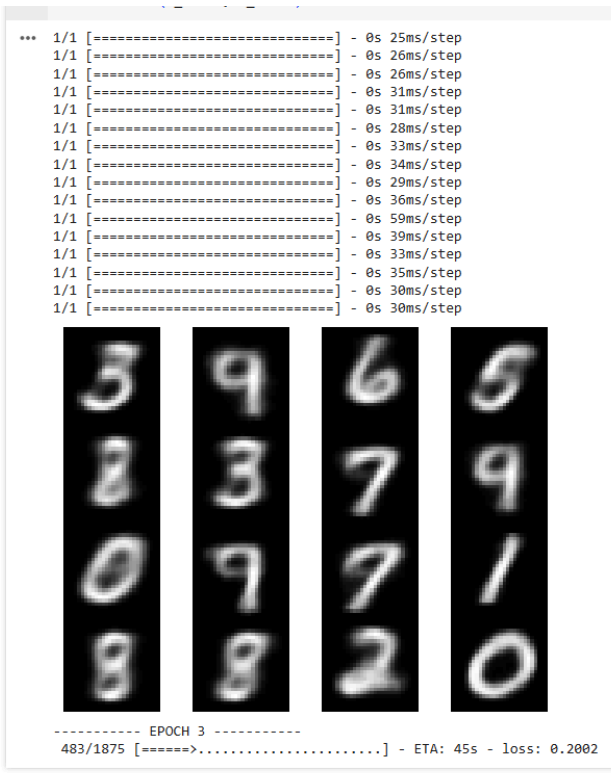
Tutorial

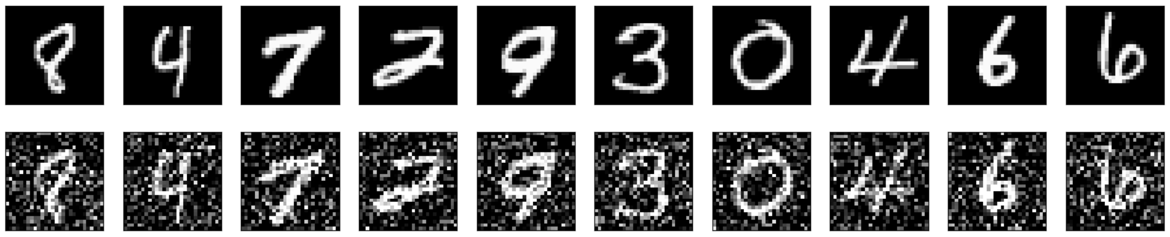


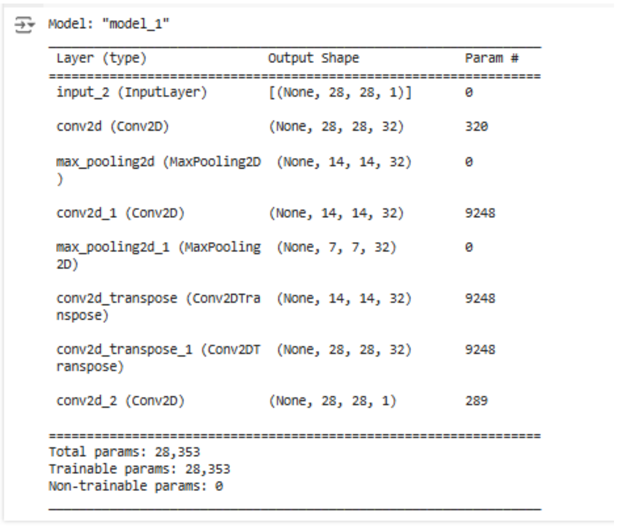
```
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
```

```
----------- EPOCH 3 -----------
 483/1875 [======>.......................] - ETA: 45s - loss: 0.2002
```

Image Denoising

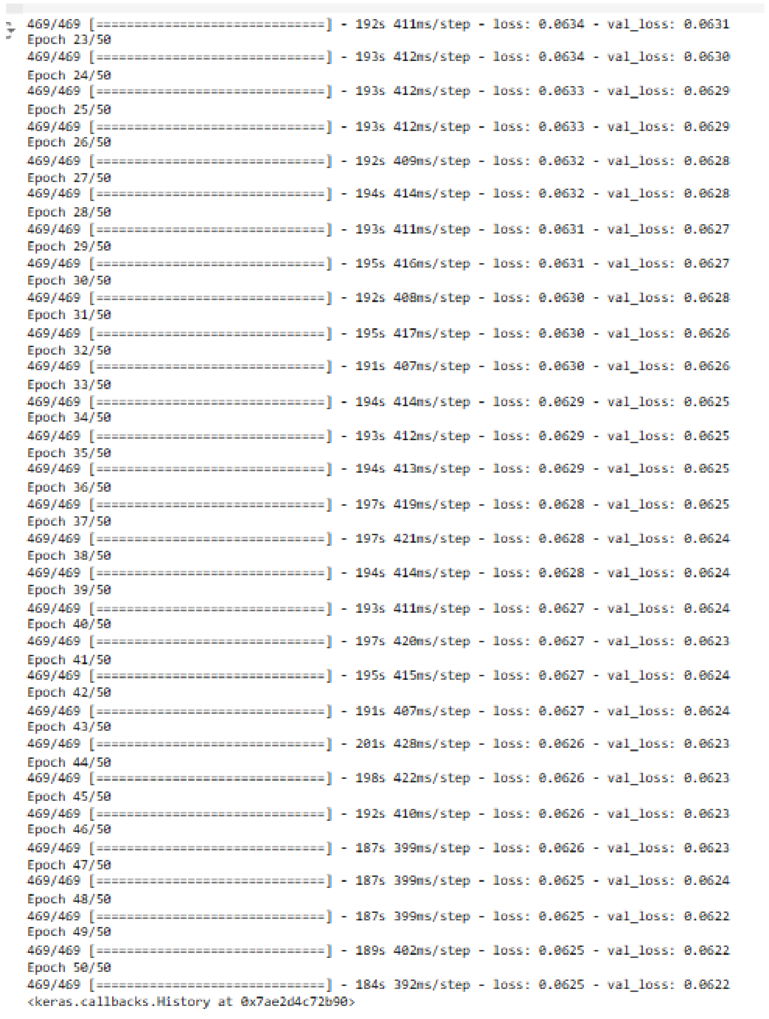
```
# Since we only need images from the dataset to encode and decode, we
# won't use the labels.
(train_data, _), (test_data, _) = mnist.load_data()

# Normalize and reshape the data
train_data = preprocess(train_data)
test_data = preprocess(test_data)

# Create a copy of the data with added noise
noisy_train_data = noise(train_data)
noisy_test_data = noise(test_data)

# Display the train data and a version of it with added noise
display(train_data, noisy_train_data)
```



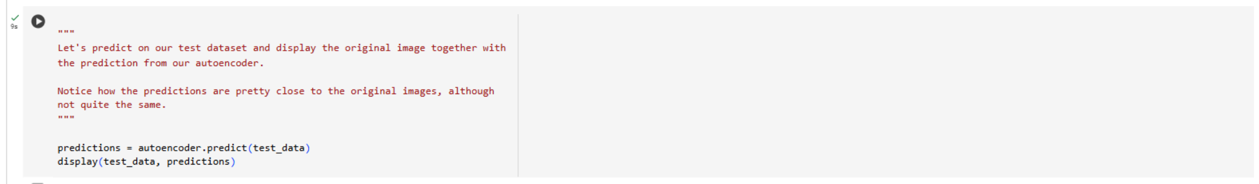


```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d (Conv2D)             (None, 28, 28, 32)        320

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 conv2d_transpose (Conv2DTra  (None, 14, 14, 32)       9248
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 28, 28, 32)       9248
 ranspose)

 conv2d_2 (Conv2D)           (None, 28, 28, 1)         289

=================================================================
Total params: 28,353
Trainable params: 28,353
Non-trainable params: 0
_________________________________________________________________
```

Training on Regular Images

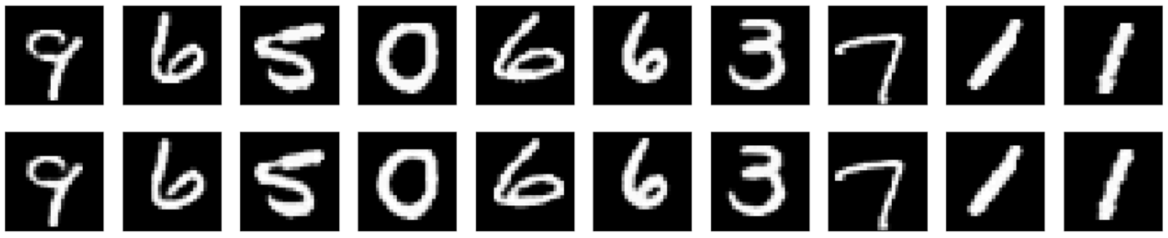


```
469/469 [==============================] - 192s 411ms/step - loss: 0.0634 - val_loss: 0.0631
Epoch 23/50
469/469 [==============================] - 193s 412ms/step - loss: 0.0634 - val_loss: 0.0630
Epoch 24/50
469/469 [==============================] - 193s 412ms/step - loss: 0.0633 - val_loss: 0.0629
Epoch 25/50
469/469 [==============================] - 193s 412ms/step - loss: 0.0633 - val_loss: 0.0629
Epoch 26/50
469/469 [==============================] - 192s 409ms/step - loss: 0.0632 - val_loss: 0.0628
Epoch 27/50
469/469 [==============================] - 194s 414ms/step - loss: 0.0632 - val_loss: 0.0628
Epoch 28/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0631 - val_loss: 0.0627
Epoch 29/50
469/469 [==============================] - 195s 416ms/step - loss: 0.0631 - val_loss: 0.0627
Epoch 30/50
469/469 [==============================] - 192s 408ms/step - loss: 0.0630 - val_loss: 0.0628
Epoch 31/50
469/469 [==============================] - 195s 417ms/step - loss: 0.0630 - val_loss: 0.0626
Epoch 32/50
469/469 [==============================] - 191s 407ms/step - loss: 0.0630 - val_loss: 0.0626
Epoch 33/50
469/469 [==============================] - 194s 414ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 34/50
469/469 [==============================] - 193s 412ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 35/50
469/469 [==============================] - 194s 413ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 36/50
469/469 [==============================] - 197s 419ms/step - loss: 0.0628 - val_loss: 0.0625
Epoch 37/50
469/469 [==============================] - 197s 421ms/step - loss: 0.0628 - val_loss: 0.0624
Epoch 38/50
469/469 [==============================] - 194s 414ms/step - loss: 0.0628 - val_loss: 0.0624
Epoch 39/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0627 - val_loss: 0.0624
Epoch 40/50
469/469 [==============================] - 197s 420ms/step - loss: 0.0627 - val_loss: 0.0623
Epoch 41/50
469/469 [==============================] - 195s 415ms/step - loss: 0.0627 - val_loss: 0.0624
Epoch 42/50
469/469 [==============================] - 191s 407ms/step - loss: 0.0627 - val_loss: 0.0624
Epoch 43/50
469/469 [==============================] - 201s 428ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 44/50
469/469 [==============================] - 198s 422ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 45/50
469/469 [==============================] - 192s 410ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 46/50
469/469 [==============================] - 187s 399ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 47/50
469/469 [==============================] - 187s 399ms/step - loss: 0.0625 - val_loss: 0.0624
Epoch 48/50
469/469 [==============================] - 187s 399ms/step - loss: 0.0625 - val_loss: 0.0622
Epoch 49/50
469/469 [==============================] - 189s 402ms/step - loss: 0.0625 - val_loss: 0.0622
Epoch 50/50
469/469 [==============================] - 184s 392ms/step - loss: 0.0625 - val_loss: 0.0622
<keras.callbacks.History at 0x7ae2d4c72b90>
```

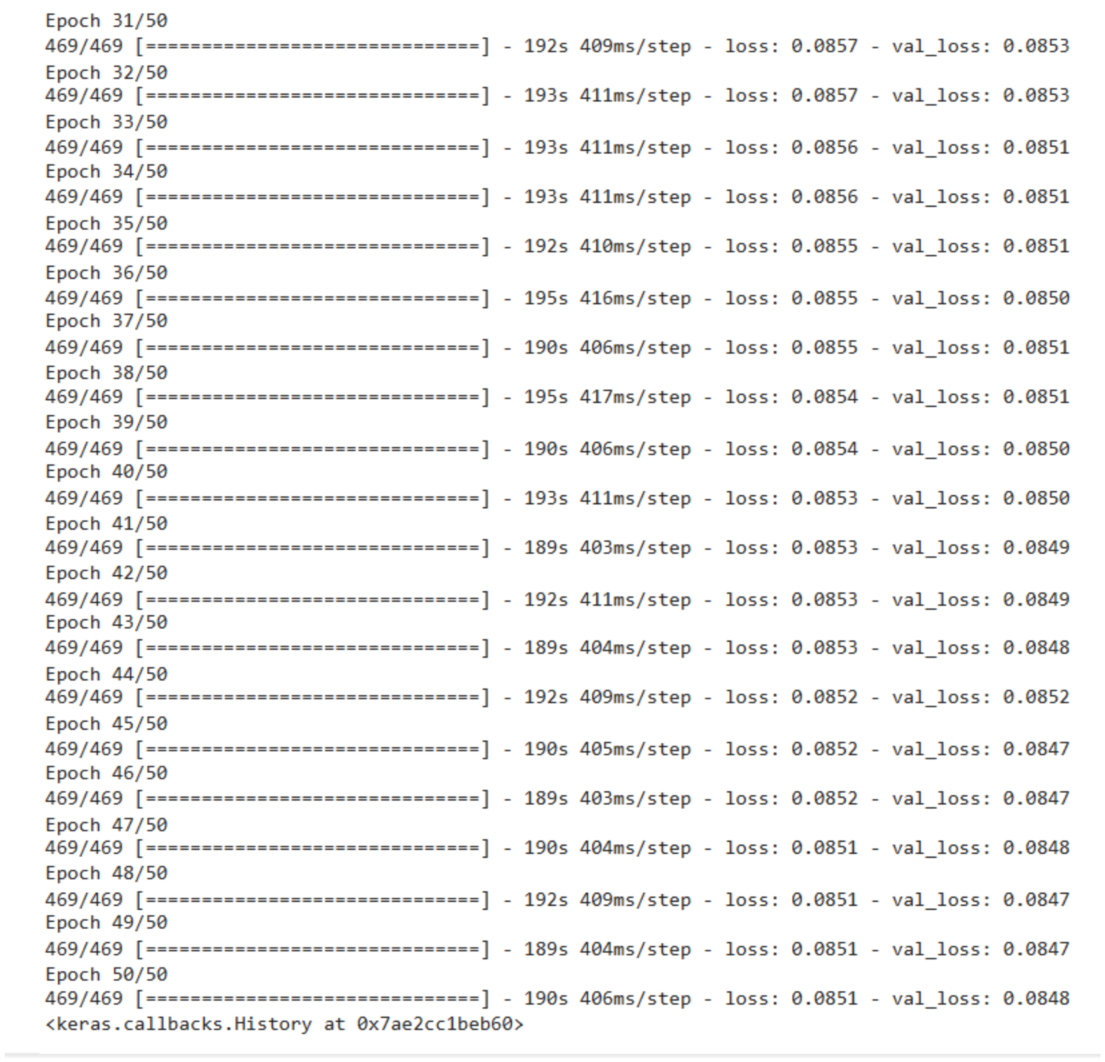


```
"""
Let's predict on our test dataset and display the original image together with
the prediction from our autoencoder.

Notice how the predictions are pretty close to the original images, although
not quite the same.
"""

predictions = autoencoder.predict(test_data)
display(test_data, predictions)
```

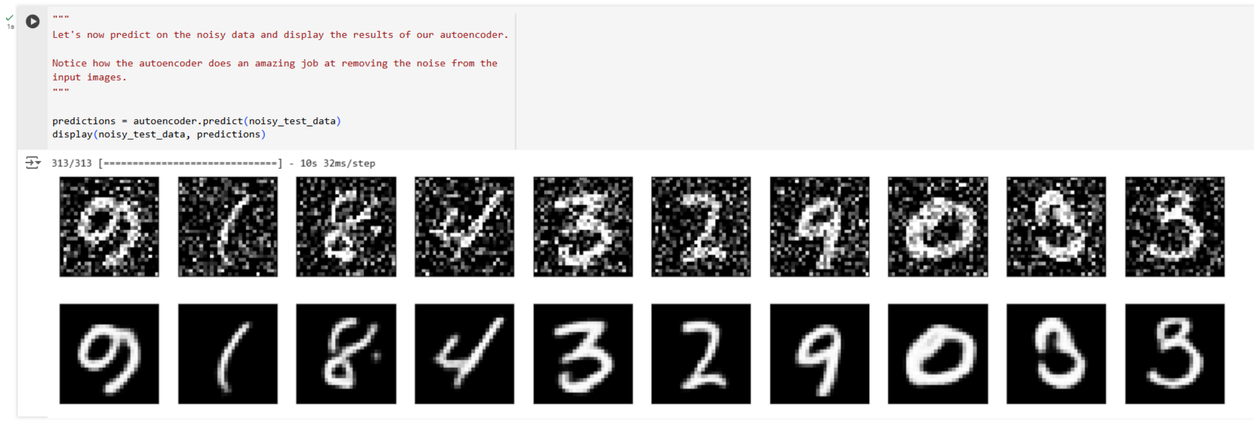
```
313/313 [==============================] - 8s 25ms/step
```



Training on Noisy Images



```
Epoch 31/50
469/469 [==============================] - 192s 409ms/step - loss: 0.0857 - val_loss: 0.0853
Epoch 32/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0857 - val_loss: 0.0853
Epoch 33/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0856 - val_loss: 0.0851
Epoch 34/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0856 - val_loss: 0.0851
Epoch 35/50
469/469 [==============================] - 192s 410ms/step - loss: 0.0855 - val_loss: 0.0851
Epoch 36/50
469/469 [==============================] - 195s 416ms/step - loss: 0.0855 - val_loss: 0.0850
Epoch 37/50
469/469 [==============================] - 190s 406ms/step - loss: 0.0855 - val_loss: 0.0851
Epoch 38/50
469/469 [==============================] - 195s 417ms/step - loss: 0.0854 - val_loss: 0.0851
Epoch 39/50
469/469 [==============================] - 190s 406ms/step - loss: 0.0854 - val_loss: 0.0850
Epoch 40/50
469/469 [==============================] - 193s 411ms/step - loss: 0.0853 - val_loss: 0.0850
Epoch 41/50
469/469 [==============================] - 189s 403ms/step - loss: 0.0853 - val_loss: 0.0849
Epoch 42/50
469/469 [==============================] - 192s 411ms/step - loss: 0.0853 - val_loss: 0.0849
Epoch 43/50
469/469 [==============================] - 189s 404ms/step - loss: 0.0853 - val_loss: 0.0848
Epoch 44/50
469/469 [==============================] - 192s 409ms/step - loss: 0.0852 - val_loss: 0.0852
Epoch 45/50
469/469 [==============================] - 190s 405ms/step - loss: 0.0852 - val_loss: 0.0847
Epoch 46/50
469/469 [==============================] - 189s 403ms/step - loss: 0.0852 - val_loss: 0.0847
Epoch 47/50
469/469 [==============================] - 190s 404ms/step - loss: 0.0851 - val_loss: 0.0848
Epoch 48/50
469/469 [==============================] - 192s 409ms/step - loss: 0.0851 - val_loss: 0.0847
Epoch 49/50
469/469 [==============================] - 189s 404ms/step - loss: 0.0851 - val_loss: 0.0847
Epoch 50/50
469/469 [==============================] - 190s 406ms/step - loss: 0.0851 - val_loss: 0.0848
<keras.callbacks.History at 0x7ae2cc1beb60>
```



```
"""
Let's now predict on the noisy data and display the results of our autoencoder.

Notice how the autoencoder does an amazing job at removing the noise from the
input images.
"""

predictions = autoencoder.predict(noisy_test_data)
display(noisy_test_data, predictions)
```

```
313/313 [==============================] - 10s 32ms/step
```