Experiment: 3

Aim

To build and train an Autoencoder model that can learn to encode and decode images, particularly by denoising noisy MNIST images through reconstruction.

Theory:

An Autoencoder is a type of artificial neural network used to learn efficient representations of input data, typically for dimensionality reduction or noise removal. It consists of two main parts:

Encoder: Compresses the input into a lower-dimensional representation. Decoder: Reconstructs the input from this compressed representation. In this experiment, an Autoencoder is trained on the MNIST dataset, consisting of 28x28 grayscale images of handwritten digits. The data is preprocessed by normalizing pixel values to the range [0, 1] and reshaped into the required format. Additionally, random noise is added to the images, creating a noisy dataset that is used to train the model for denoising.

The Autoencoder uses convolutional layers to effectively learn spatial hierarchies in the images. The encoder compresses the input images using two convolutional layers followed by max-pooling layers. The decoder reconstructs the images using transposed convolutional layers, outputting images of the same shape as the input. The model is trained to minimize reconstruction error using binary cross-entropy as the loss function. After training, the Autoencoder can successfully remove noise from test images, producing a cleaner version of the original input.

In [ ]:

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matpl
otlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotli
b) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matp
lotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from matp
lotlib) (1.4.5)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.10/dist-packages (from matplotli
b) (1.24.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplo
tlib) (23.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-packages (from matplotlib)
(10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matpl
otlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from m
atplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateut
il>=2.7->matplotlib) (1.16.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour wit
h the system package manager. It is recommended to use a virtual environment instead: https://pip.pyp
a.io/warnings/venv
```

In [ ]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
```

Pre-processing

Normalizing the data into 0 to 1 and reshaping the size

In [ ]:

```
def preprocess(array):
    """
    Normalizes the supplied array and reshapes it into the appropriate format.
    """

    array = array.astype("float32") / 255.0
    array = np.reshape(array, (len(array), 28, 28, 1))
    return array
```

Adding noise to the original images

In [ ]:

```
def noise(array):
    """
    Adds random noise to each image in the supplied array.
    """

    noise_factor = 0.4
    noisy_array = array + noise_factor * np.random.normal(
        loc=0.0, scale=1.0, size=array.shape
    )

    return np.clip(noisy_array, 0.0, 1.0)
```

Visualizing the images

In [ ]:

```
def display(array1, array2):
    """
    Displays ten random images from each one of the supplied arrays.
    """

    n = 10

    indices = np.random.randint(len(array1), size=n)
    images1 = array1[indices, :]
    images2 = array2[indices, :]

    plt.figure(figsize=(20, 4))
    for i, (image1, image2) in enumerate(zip(images1, images2)):
        ax = plt.subplot(2, n, i + 1)
        plt.imshow(image1.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(image2.reshape(28, 28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

    plt.show()
```

Preparing the data

In [ ]:

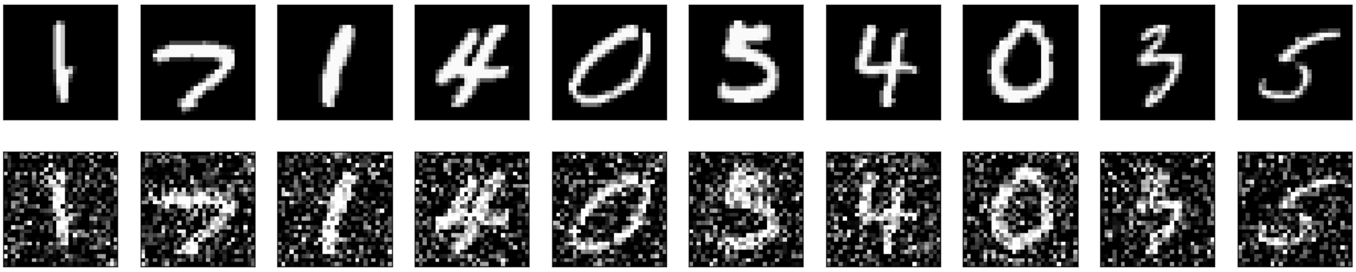
```
# Since we only need images from the dataset to encode and decode, we
# won't use the labels.
(train_data, _), (test_data, _) = mnist.load_data()

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

```python
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)
```



Building the Autoencoder

In [ ]:

```python
input = layers.Input(shape=(28, 28, 1))

# Encoder
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(input)
x = layers.MaxPooling2D((2, 2), padding="same")(x)
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(x)
x = layers.MaxPooling2D((2, 2), padding="same")(x)

# Decoder
x = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)

# Autoencoder
autoencoder = Model(input, x)
autoencoder.compile(optimizer="adam", loss="binary_crossentropy")
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 1)]       0         
                                                                 
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 14, 14, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 32)          0         
 g2D)                                                            
                                                                 
 conv2d_transpose (Conv2DTr  (None, 14, 14, 32)        9248      
 anspose)                                                        
                                                                 
 conv2d_transpose_1 (Conv2D  (None, 28, 28, 32)        9248      
 Transpose)                                                      
                                                                 
 conv2d_2 (Conv2D)           (None, 28, 28, 1)         289       
                                                                 
=================================================================
Total params: 28353 (110.75 KB)
Trainable params: 28353 (110.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
2024-07-11 10:14:49.345534: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1926] Created device /j
ob:localhost/replica:0/task:0/device:GPU:0 with 17947 MB memory:  -> device: 0, name: NVIDIA A100-SXM4
-40GB MIG 3g.20gb, pci bus id: 0000:b7:00.0, compute capability: 8.0
```

Training the model

In [ ]:

```
autoencoder.fit(
    x=train_data,
    y=train_data,
    epochs=50,
    batch_size=128,
    shuffle=True,
    validation_data=(test_data, test_data),
)
```

```
Epoch 1/50
```

```
2024-07-11 10:14:51.246797: I external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:467] Loaded cuDNN version 90100
2024-07-11 10:14:52.225470: I external/local_xla/xla/service/service.cc:168] XLA service 0x7fb160ae6fd0 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
2024-07-11 10:14:52.225522: I external/local_xla/xla/service/service.cc:176]   StreamExecutor device (0): NVIDIA A100-SXM4-40GB MIG 3g.20gb, Compute Capability 8.0
2024-07-11 10:14:52.232071: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1720692892.349650  994793 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
```

```
469/469 [==============================] - 6s 6ms/step - loss: 0.1317 - val_loss: 0.0737
Epoch 2/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0723 - val_loss: 0.0701
Epoch 3/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0699 - val_loss: 0.0686
Epoch 4/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0686 - val_loss: 0.0676
Epoch 5/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0677 - val_loss: 0.0669
Epoch 6/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0671 - val_loss: 0.0665
Epoch 7/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0666 - val_loss: 0.0660
Epoch 8/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0662 - val_loss: 0.0656
Epoch 9/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0659 - val_loss: 0.0653
Epoch 10/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0656 - val_loss: 0.0651
Epoch 11/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0653 - val_loss: 0.0648
Epoch 12/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0651 - val_loss: 0.0646
Epoch 13/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0649 - val_loss: 0.0644
Epoch 14/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0647 - val_loss: 0.0642
Epoch 15/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0645 - val_loss: 0.0641
Epoch 16/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0643 - val_loss: 0.0639
Epoch 17/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0642 - val_loss: 0.0638
Epoch 18/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0640 - val_loss: 0.0637
Epoch 19/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0639 - val_loss: 0.0635
Epoch 20/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0638 - val_loss: 0.0635
Epoch 21/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0637 - val_loss: 0.0634
Epoch 22/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0636 - val_loss: 0.0632
Epoch 23/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0635 - val_loss: 0.0631
Epoch 24/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0635 - val_loss: 0.0632
Epoch 25/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0634 - val_loss: 0.0630
Epoch 26/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0633 - val_loss: 0.0629
Epoch 27/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0633 - val_loss: 0.0629
Epoch 28/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0632 - val_loss: 0.0628
Epoch 29/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0631 - val_loss: 0.0628
Epoch 30/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0631 - val_loss: 0.0627
Epoch 31/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0631 - val_loss: 0.0627
Epoch 32/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0630 - val_loss: 0.0626
Epoch 33/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0630 - val_loss: 0.0626
Epoch 34/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 35/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 36/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0629 - val_loss: 0.0625
Epoch 37/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0628 - val_loss: 0.0625
Epoch 38/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0628 - val_loss: 0.0626
```

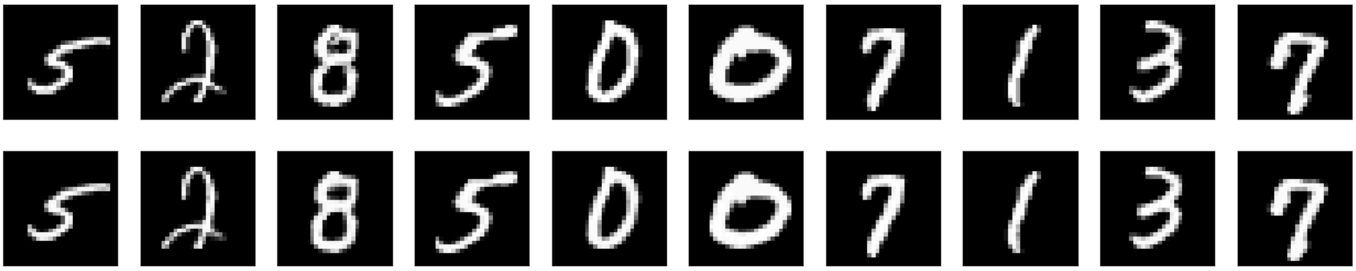
```
Epoch 39/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0628 - val_loss: 0.0624
Epoch 40/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0627 - val_loss: 0.0624
Epoch 41/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0627 - val_loss: 0.0623
Epoch 42/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0627 - val_loss: 0.0623
Epoch 43/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0627 - val_loss: 0.0624
Epoch 44/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 45/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 46/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 47/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0626 - val_loss: 0.0623
Epoch 48/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0625 - val_loss: 0.0622
Epoch 49/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0625 - val_loss: 0.0623
Epoch 50/50
469/469 [==============================] - 2s 4ms/step - loss: 0.0625 - val_loss: 0.0622
```

```
Out[ ]: <keras.src.callbacks.History at 0x7fbb3749d210>
```

Prediction

```
In [ ]: predictions = autoencoder.predict(test_data)
        display(test_data, predictions)
```

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



```
In [ ]: autoencoder.fit(
            x=noisy_train_data,
            y=train_data,
            epochs=100,
            batch_size=128,
            shuffle=True,
            validation_data=(noisy_test_data, test_data),
        )
```

```
Epoch 1/100
469/469 [==============================] - 2s 4ms/step - loss: 0.1017 - val_loss: 0.0943
Epoch 2/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0937 - val_loss: 0.0920
Epoch 3/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0920 - val_loss: 0.0908
Epoch 4/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0911 - val_loss: 0.0901
Epoch 5/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0904 - val_loss: 0.0895
Epoch 6/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0899 - val_loss: 0.0892
Epoch 7/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0895 - val_loss: 0.0888
Epoch 8/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0891 - val_loss: 0.0888
Epoch 9/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0888 - val_loss: 0.0882
Epoch 10/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0885 - val_loss: 0.0881
Epoch 11/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0883 - val_loss: 0.0875
Epoch 12/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0880 - val_loss: 0.0873
Epoch 13/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0878 - val_loss: 0.0872
Epoch 14/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0876 - val_loss: 0.0870
Epoch 15/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0875 - val_loss: 0.0876
Epoch 16/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0873 - val_loss: 0.0868
Epoch 17/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0871 - val_loss: 0.0866
Epoch 18/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0870 - val_loss: 0.0866
Epoch 19/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0869 - val_loss: 0.0863
Epoch 20/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0868 - val_loss: 0.0863
Epoch 21/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0867 - val_loss: 0.0862
Epoch 22/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0866 - val_loss: 0.0861
Epoch 23/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0865 - val_loss: 0.0860
Epoch 24/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0864 - val_loss: 0.0859
Epoch 25/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0863 - val_loss: 0.0858
Epoch 26/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0863 - val_loss: 0.0859
Epoch 27/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0862 - val_loss: 0.0857
Epoch 28/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0861 - val_loss: 0.0857
Epoch 29/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0860 - val_loss: 0.0856
Epoch 30/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0860 - val_loss: 0.0856
Epoch 31/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0859 - val_loss: 0.0859
Epoch 32/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0859 - val_loss: 0.0857
Epoch 33/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0858 - val_loss: 0.0856
Epoch 34/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0858 - val_loss: 0.0854
Epoch 35/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0858 - val_loss: 0.0856
Epoch 36/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0857 - val_loss: 0.0853
Epoch 37/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0857 - val_loss: 0.0853
Epoch 38/100
```

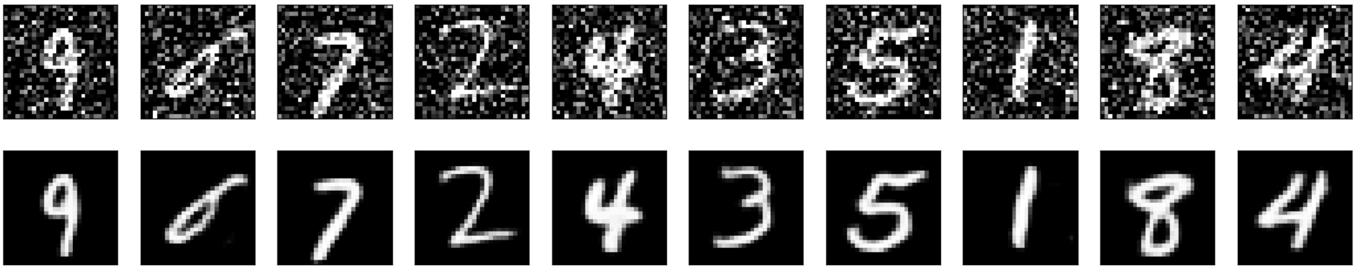
```
469/469 [==============================] - 2s 4ms/step - loss: 0.0856 - val_loss: 0.0853
Epoch 39/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0856 - val_loss: 0.0853
Epoch 40/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0855 - val_loss: 0.0852
Epoch 41/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0855 - val_loss: 0.0854
Epoch 42/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0855 - val_loss: 0.0853
Epoch 43/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0855 - val_loss: 0.0853
Epoch 44/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0854 - val_loss: 0.0851
Epoch 45/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0854 - val_loss: 0.0854
Epoch 46/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0853 - val_loss: 0.0851
Epoch 47/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0853 - val_loss: 0.0851
Epoch 48/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0853 - val_loss: 0.0850
Epoch 49/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0853 - val_loss: 0.0850
Epoch 50/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0853 - val_loss: 0.0851
Epoch 51/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0852 - val_loss: 0.0850
Epoch 52/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0852 - val_loss: 0.0850
Epoch 53/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0852 - val_loss: 0.0849
Epoch 54/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0852 - val_loss: 0.0849
Epoch 55/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0851
Epoch 56/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0849
Epoch 57/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0848
Epoch 58/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0849
Epoch 59/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0848
Epoch 60/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0848
Epoch 61/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0851 - val_loss: 0.0848
Epoch 62/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0850 - val_loss: 0.0848
Epoch 63/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0850 - val_loss: 0.0847
Epoch 64/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0850 - val_loss: 0.0847
Epoch 65/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0850 - val_loss: 0.0850
Epoch 66/100
469/469 [==============================] - 2s 3ms/step - loss: 0.0850 - val_loss: 0.0850
Epoch 67/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0850 - val_loss: 0.0847
Epoch 68/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0847
Epoch 69/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0847
Epoch 70/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0847
Epoch 71/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0848
Epoch 72/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0847
Epoch 73/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0847
Epoch 74/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0848
Epoch 75/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0849 - val_loss: 0.0846
```

```
Epoch 76/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 77/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 78/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 79/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0847
Epoch 80/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 81/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 82/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0845
Epoch 83/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0847
Epoch 84/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 85/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0848 - val_loss: 0.0846
Epoch 86/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 87/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 88/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0847
Epoch 89/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0849
Epoch 90/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 91/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 92/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 93/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0847
Epoch 94/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0846
Epoch 95/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0845
Epoch 96/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0847 - val_loss: 0.0844
Epoch 97/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0846 - val_loss: 0.0845
Epoch 98/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0846 - val_loss: 0.0845
Epoch 99/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0846 - val_loss: 0.0846
Epoch 100/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0846 - val_loss: 0.0845
```

```
Out[ ]: <keras.src.callbacks.History at 0x7fbb4049b070>
```

```
In [ ]: predictions = autoencoder.predict(noisy_test_data)
        display(noisy_test_data, predictions)
```

```
313/313 [==============================] - 0s 997us/step
```



```
In [ ]:
```