In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
```

In [2]:
```
# Step 1: Load the MNIST Dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Step 2: Preprocess the Data
x_train = x_train.astype('float32') / 255.0  # Normalize to [0, 1]
x_test = x_test.astype('float32') / 255.0    # Normalize to [0, 1]
x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))  # Reshape for CNN
x_test = np.reshape(x_test, (len(x_test), 28, 28, 1))      # Reshape for CNN
```

In [3]:
```
# Step 3: Add Noise to the Data
def add_noise(images):
    noise_factor = 0.5  # You can adjust this to change the noise level
    noisy_images = images + noise_factor * np.random.normal(loc=0.0, scale=1.0, siz
    return np.clip(noisy_images, 0., 1.)  # Ensure values are still in [0, 1]

x_train_noisy = add_noise(x_train)
x_test_noisy = add_noise(x_test)
```

In [4]:
```
# Step 4: Build the Autoencoder Model
def build_autoencoder():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_sh
    model.add(layers.MaxPooling2D((2, 2), padding='same'))

    model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), padding='same'))

    model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
    model.add(layers.UpSampling2D((2, 2)))

    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(layers.UpSampling2D((2, 2)))

    model.add(layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same'))  # Ou
    return model

autoencoder = build_autoencoder()
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

```
C:\Users\bonde\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.
py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. W
hen using Sequential models, prefer using an `Input(shape)` object as the first laye
r in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [5]:
```
# Step 5: Train the Autoencoder
autoencoder.fit(x_train_noisy, x_train,
                epochs=50,
```

```
                batch_size=128,
                validation_data=(x_test_noisy, x_test))
```

```
Epoch 1/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 15s 29ms/step - loss: 0.2696 - val_loss: 0.1228
Epoch 2/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 37ms/step - loss: 0.1211 - val_loss: 0.1131
Epoch 3/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.1130 - val_loss: 0.1085
Epoch 4/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.1092 - val_loss: 0.1070
Epoch 5/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.1068 - val_loss: 0.1044
Epoch 6/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.1052 - val_loss: 0.1030
Epoch 7/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 41ms/step - loss: 0.1037 - val_loss: 0.1023
Epoch 8/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.1027 - val_loss: 0.1013
Epoch 9/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.1018 - val_loss: 0.1006
Epoch 10/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 40ms/step - loss: 0.1012 - val_loss: 0.1002
Epoch 11/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 40ms/step - loss: 0.1006 - val_loss: 0.0994
Epoch 12/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 39ms/step - loss: 0.1003 - val_loss: 0.0991
Epoch 13/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 37ms/step - loss: 0.0996 - val_loss: 0.0991
Epoch 14/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.0994 - val_loss: 0.0988
Epoch 15/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - loss: 0.0992 - val_loss: 0.0982
Epoch 16/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 37ms/step - loss: 0.0988 - val_loss: 0.0985
Epoch 17/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 38ms/step - loss: 0.0987 - val_loss: 0.0978
Epoch 18/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 37ms/step - loss: 0.0982 - val_loss: 0.0975
Epoch 19/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 37ms/step - loss: 0.0982 - val_loss: 0.0975
Epoch 20/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 37ms/step - loss: 0.0980 - val_loss: 0.0976
Epoch 21/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 36ms/step - loss: 0.0979 - val_loss: 0.0976
Epoch 22/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 38ms/step - loss: 0.0978 - val_loss: 0.0970
Epoch 23/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 39ms/step - loss: 0.0976 - val_loss: 0.0968
Epoch 24/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 43ms/step - loss: 0.0975 - val_loss: 0.0969
Epoch 25/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 43ms/step - loss: 0.0973 - val_loss: 0.0967
Epoch 26/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 43ms/step - loss: 0.0974 - val_loss: 0.0966
Epoch 27/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.0972 - val_loss: 0.0966
Epoch 28/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.0970 - val_loss: 0.0964
```

```
Epoch 29/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - loss: 0.0973 - val_loss: 0.0964
Epoch 30/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 45ms/step - loss: 0.0971 - val_loss: 0.0964
Epoch 31/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 23s 48ms/step - loss: 0.0968 - val_loss: 0.0965
Epoch 32/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.0969 - val_loss: 0.0963
Epoch 33/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 42ms/step - loss: 0.0971 - val_loss: 0.0961
Epoch 34/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 44ms/step - loss: 0.0967 - val_loss: 0.0962
Epoch 35/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 42ms/step - loss: 0.0967 - val_loss: 0.0961
Epoch 36/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 39ms/step - loss: 0.0965 - val_loss: 0.0960
Epoch 37/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 40ms/step - loss: 0.0963 - val_loss: 0.0961
Epoch 38/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 45ms/step - loss: 0.0966 - val_loss: 0.0960
Epoch 39/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 20s 42ms/step - loss: 0.0964 - val_loss: 0.0962
Epoch 40/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - loss: 0.0964 - val_loss: 0.0958
Epoch 41/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 23s 49ms/step - loss: 0.0965 - val_loss: 0.0961
Epoch 42/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 26s 55ms/step - loss: 0.0964 - val_loss: 0.0960
Epoch 43/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 23s 49ms/step - loss: 0.0963 - val_loss: 0.0960
Epoch 44/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - loss: 0.0963 - val_loss: 0.0957
Epoch 45/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 24s 50ms/step - loss: 0.0962 - val_loss: 0.0958
Epoch 46/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.0961 - val_loss: 0.0958
Epoch 47/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 44ms/step - loss: 0.0960 - val_loss: 0.0957
Epoch 48/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.0962 - val_loss: 0.0956
Epoch 49/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 45ms/step - loss: 0.0961 - val_loss: 0.0958
Epoch 50/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 21s 45ms/step - loss: 0.0962 - val_loss: 0.0956
```

Out[5]: `<keras.src.callbacks.history.History at 0x1ed09c2ee10>`

In [6]:
```python
# Step 6: Denoise the Test Images
x_test_denoised = autoencoder.predict(x_test_noisy)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step
```

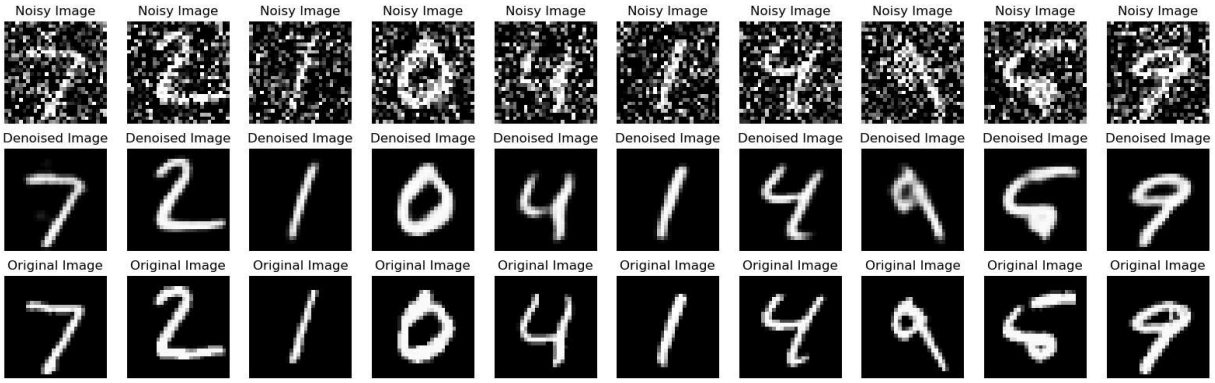
In [7]:
```python
# Step 7: Visualize the Results
n = 10  # Number of images to display
plt.figure(figsize=(20, 6))
for i in range(n):
    # Display noisy images
```

```
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(x_test_noisy[i].reshape(28, 28), cmap='gray')
    plt.title("Noisy Image")
    plt.axis('off')

    # Display denoised images
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(x_test_denoised[i].reshape(28, 28), cmap='gray')
    plt.title("Denoised Image")
    plt.axis('off')

    # Display original images
    ax = plt.subplot(3, n, i + 1 + 2 * n)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title("Original Image")
    plt.axis('off')

plt.show()
```



In [ ]: