In [1]:

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

# 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 CN

# 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=
    return np.clip(noisy_images, 0., 1.)  # Ensure values are still in [0, 

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

# Step 4: Build the Autoencoder Model
def build_autoencoder():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (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.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'
    return model

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

# Step 5: Train the Autoencoder
autoencoder.fit(x_train_noisy, x_train,
                epochs=50,
                batch_size=128,
                validation_data=(x_test_noisy, x_test))

# Step 6: Denoise the Test Images
x_test_denoised = autoencoder.predict(x_test_noisy)

# 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)
```

```python
    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()
```

```
C:\Users\sd616\anaconda\lib\site-packages\keras\src\layers\convolutiona
l\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `
Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Epoch 1/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 30s 54ms/step - loss: 0.2813 - val_loss:
0.1247
Epoch 2/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 24s 51ms/step - loss: 0.1228 - val_loss:
0.1148
Epoch 3/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 48ms/step - loss: 0.1149 - val_loss:
0.1118
Epoch 4/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 48ms/step - loss: 0.1111 - val_loss:
0.1081
Epoch 5/50
469/469 ━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - loss: 0.1088 - val_loss:
```

In [ ]: