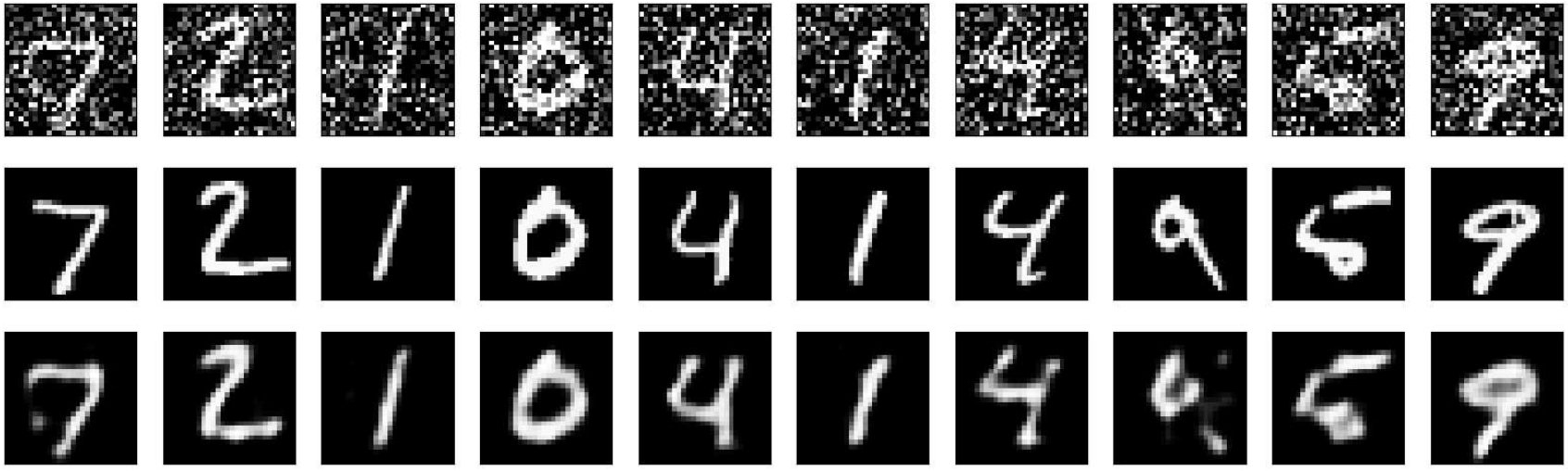
In [12]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset (MNIST in this case)
(x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()

# Normalize the data
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Reshape the data to include a channel dimension
x_train = np.reshape(x_train, (x_train.shape[0], 28, 28, 1))
x_test = np.reshape(x_test, (x_test.shape[0], 28, 28, 1))

# Function to add noise to images
def add_noise(images, noise_factor=0.5):
    noisy_images = images + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=images.shape)
    noisy_images = np.clip(noisy_images, 0., 1.)
    return noisy_images

# Add noise to the training and testing images
x_train_noisy = add_noise(x_train)
x_test_noisy = add_noise(x_test)

# Define the denoising autoencoder model
input_img = layers.Input(shape=(28, 28, 1))

# Encoder
x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = layers.MaxPooling2D((2, 2), padding='same')(x)

# Decoder
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
```

```
decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

# Autoencoder model
autoencoder = models.Model(input_img, decoded)

# Compile the autoencoder
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Train the denoising autoencoder
autoencoder.fit(x_train_noisy, x_train, epochs=10, batch_size=256, validation_data=(x_test_noisy, x_test))

# Denoise the test images
denoised_imgs = autoencoder.predict(x_test_noisy)

# Visualize the noisy, original, and denoised images
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))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display original images
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display denoised images
    ax = plt.subplot(3, n, i + 1 + 2 * n)
    plt.imshow(denoised_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```

```
Epoch 1/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 11s 43ms/step - loss: 0.3930 - val_loss: 0.1517
Epoch 2/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 0.1463 - val_loss: 0.1328
Epoch 3/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 29ms/step - loss: 0.1314 - val_loss: 0.1263
Epoch 4/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 13s 41ms/step - loss: 0.1260 - val_loss: 0.1227
Epoch 5/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 40ms/step - loss: 0.1229 - val_loss: 0.1201
Epoch 6/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 39ms/step - loss: 0.1209 - val_loss: 0.1183
Epoch 7/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 40ms/step - loss: 0.1189 - val_loss: 0.1169
Epoch 8/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 30ms/step - loss: 0.1178 - val_loss: 0.1159
Epoch 9/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - loss: 0.1167 - val_loss: 0.1148
Epoch 10/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 26ms/step - loss: 0.1154 - val_loss: 0.1140
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```



In [ ]: