In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
```

In [3]:
```
# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize the pixel values to be between 0 and 1
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Flatten the 28x28 images into vectors of size 784
x_train = x_train.reshape(-1, 28 * 28)
x_test = x_test.reshape(-1, 28 * 28)
```

In [4]:
```
# Input layer
input_img = Input(shape=(784,))

# Encoder: reduce the dimensionality
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)  # Bottleneck layer

# Decoder: reconstruct the input
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)  # Output layer

# Compile the autoencoder model
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='mse')
```

In [5]:
```
# Input layer
input_img = Input(shape=(784,))

# Encoder: reduce the dimensionality
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)  # Bottleneck layer

# Decoder: reconstruct the input
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)  # Output layer

# Compile the autoencoder model
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='mse')
```

In [6]:
```
# Train the model
history = autoencoder.fit(x_train, x_train,
                          epochs=50,
                          batch_size=256,
                          shuffle=True,
                          validation_data=(x_test, x_test))
```

```
Epoch 1/50
235/235 [==============================] - 5s 13ms/step - loss: 0.0657 - val_
loss: 0.0388
Epoch 2/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0321 - val_
loss: 0.0265
Epoch 3/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0243 - val_
loss: 0.0217
Epoch 4/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0209 - val_
loss: 0.0192
Epoch 5/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0189 - val_
loss: 0.0176
Epoch 6/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0175 - val_
loss: 0.0165
Epoch 7/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0165 - val_
loss: 0.0157
Epoch 8/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0157 - val_
loss: 0.0149
Epoch 9/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0149 - val_
loss: 0.0142
Epoch 10/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0142 - val_
loss: 0.0136
Epoch 11/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0136 - val_
loss: 0.0131
Epoch 12/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0132 - val_
loss: 0.0128
Epoch 13/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0128 - val_
loss: 0.0123
Epoch 14/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0125 - val_
loss: 0.0124
Epoch 15/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0122 - val_l
oss: 0.0118
Epoch 16/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0120 - val_l
oss: 0.0116
Epoch 17/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0118 - val_l
oss: 0.0114
Epoch 18/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0116 - val_
loss: 0.0112
Epoch 19/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0114 - val_l
oss: 0.0111
```

```
Epoch 20/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0112 - val_l
oss: 0.0108
Epoch 21/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0110 - val_l
oss: 0.0106
Epoch 22/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0108 - val_l
oss: 0.0106
Epoch 23/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0106 - val_l
oss: 0.0104
Epoch 24/50
235/235 [==============================] - 2s 8ms/step - loss: 0.0104 - val_l
oss: 0.0102
Epoch 25/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0103 - val_l
oss: 0.0100
Epoch 26/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0102 - val_l
oss: 0.0100
Epoch 27/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0100 - val_
loss: 0.0098
Epoch 28/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0099 - val_
loss: 0.0097
Epoch 29/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0098 - val_
loss: 0.0096
Epoch 30/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0097 - val_
loss: 0.0095
Epoch 31/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0096 - val_
loss: 0.0094
Epoch 32/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0095 - val_
loss: 0.0094
Epoch 33/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0094 - val_
loss: 0.0092
Epoch 34/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0093 - val_
loss: 0.0091
Epoch 35/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0092 - val_
loss: 0.0090
Epoch 36/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0092 - val_
loss: 0.0090
Epoch 37/50
235/235 [==============================] - 3s 12ms/step - loss: 0.0091 - val_
loss: 0.0089
Epoch 38/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0090 - val_
loss: 0.0088
```

```
Epoch 39/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0090 - val_
loss: 0.0089
Epoch 40/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0089 - val_
loss: 0.0087
Epoch 41/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0089 - val_
loss: 0.0089
Epoch 42/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0088 - val_
loss: 0.0087
Epoch 43/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0088 - val_
loss: 0.0087
Epoch 44/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0087 - val_
loss: 0.0086
Epoch 45/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0087 - val_
loss: 0.0085
Epoch 46/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0087 - val_
loss: 0.0086
Epoch 47/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0086 - val_
loss: 0.0086
Epoch 48/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0086 - val_
loss: 0.0085
Epoch 49/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0085 - val_
loss: 0.0085
Epoch 50/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0085 - val_
loss: 0.0084
```

In [8]:
```
# Function to display original and reconstructed images
def display_images(original, reconstructed, n=10):
    plt.figure(figsize=(20, 4))
    for i in range(n):
        # Display original images
        ax = plt.subplot(2, n, i + 1)
        plt.imshow(original[i].reshape(28, 28), cmap='gray')
        plt.title("Original")
        plt.axis('off')

        # Display reconstructed images
        ax = plt.subplot(2, n, i + 1 + n)
        plt.imshow(reconstructed[i].reshape(28, 28), cmap='gray')
        plt.title("Reconstructed")
        plt.axis('off')
    plt.show()
```

In [9]:
```
# Predict the reconstructed images from test data
reconstructed_imgs = autoencoder.predict(x_test)

# Show original and reconstructed images
display_images(x_test, reconstructed_imgs)
```

```
313/313 [==============================] - 1s 2ms/step
```