Autoencoders are neural network models used for various tasks in machine learning, such as dimensionality reduction, anomaly detection, and image denoising. This program that demonstrates the application of autoencoders for image denoising using TensorFlow and Keras. This program will train an autoencoder to remove noise from images.

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist

# Load the MNIST dataset with added noise
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step

# Add Gaussian noise to the images
noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0,
scale=1.0, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0,
scale=1.0, size=x_test.shape)

# Clip pixel values to be in the range [0, 1]
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)

# Define the autoencoder model
input_img = Input(shape=(28, 28, 1))
x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Train the autoencoder
autoencoder.fit(x_train_noisy, x_train,
```

```
                epochs=10,
                batch_size=128,
                shuffle=True,
                validation_data=(x_test_noisy, x_test))
```

```
Epoch 1/10
469/469 [==============================] - 15s 9ms/step - loss: 0.1716
- val_loss: 0.1197
Epoch 2/10
469/469 [==============================] - 4s 8ms/step - loss: 0.1155
- val_loss: 0.1104
Epoch 3/10
469/469 [==============================] - 3s 7ms/step - loss: 0.1091
- val_loss: 0.1060
Epoch 4/10
469/469 [==============================] - 3s 7ms/step - loss: 0.1056
- val_loss: 0.1033
Epoch 5/10
469/469 [==============================] - 3s 7ms/step - loss: 0.1033
- val_loss: 0.1014
Epoch 6/10
469/469 [==============================] - 3s 7ms/step - loss: 0.1018
- val_loss: 0.1003
Epoch 7/10
469/469 [==============================] - 3s 6ms/step - loss: 0.1007
- val_loss: 0.0992
Epoch 8/10
469/469 [==============================] - 3s 7ms/step - loss: 0.0998
- val_loss: 0.0988
Epoch 9/10
469/469 [==============================] - 3s 7ms/step - loss: 0.0992
- val_loss: 0.0983
Epoch 10/10
469/469 [==============================] - 3s 7ms/step - loss: 0.0986
- val_loss: 0.0976
```

```
<keras.callbacks.History at 0x781450f52170>
```

```
# Denoise some test images
denoised_images = autoencoder.predict(x_test_noisy)
```

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

```
# Display noisy and denoised images
n = 10  # Number of images to display
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original image
    ax = plt.subplot(2, 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 denoised image
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(denoised_images[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```