Neural Network Deep Learning

Assignment – 8

Name: Kishor Kumar Andekar

Student ID: 700744713

Github Link : https://github.com/kishorreyansh/Neural-Network-Deep-Learning/tree/main/Assignment-8

Lesson Overview:

In this lesson, we are going to discuss types and applications of Autoencoder.

Programming elements:

1. Basics of Autoencoders

2. Role of Autoencoders in unsupervised learning

3. Types of Autoencoders

4. Use case: Simple autoencoder-Reconstructing the existing image, which will contain most important

features of the image

5. Use case: Stacked autoencoder

In class programming:

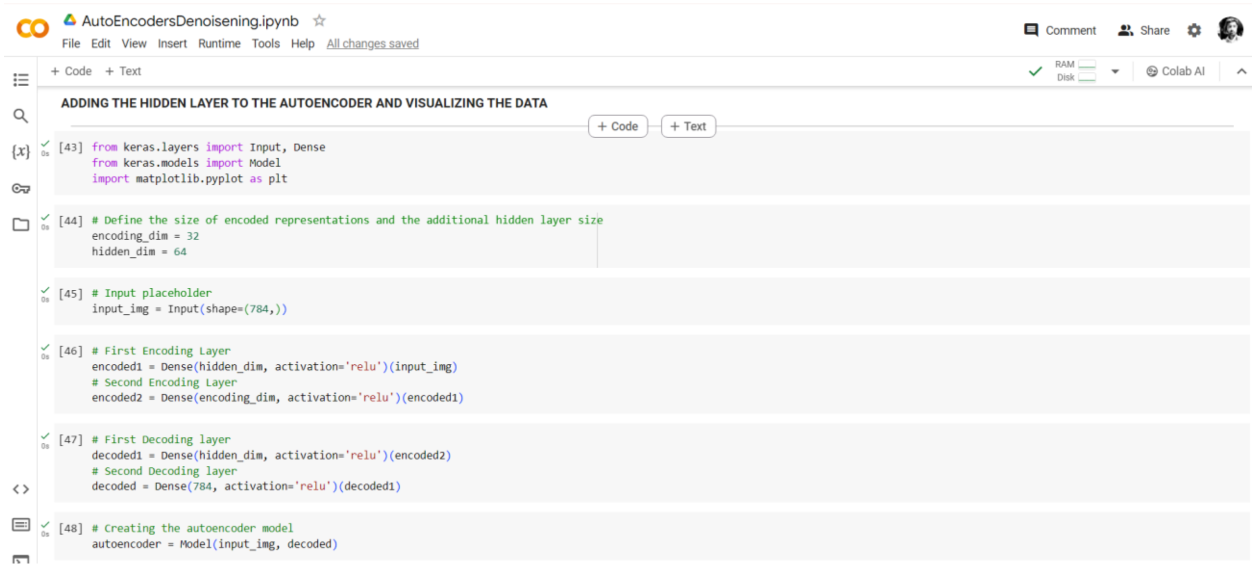
1. Add one more hidden layer to autoencoder

2. Do the prediction on the test data and then visualize one of the reconstructed version of that test data.

Also, visualize the same test data before reconstruction using Matplotlib

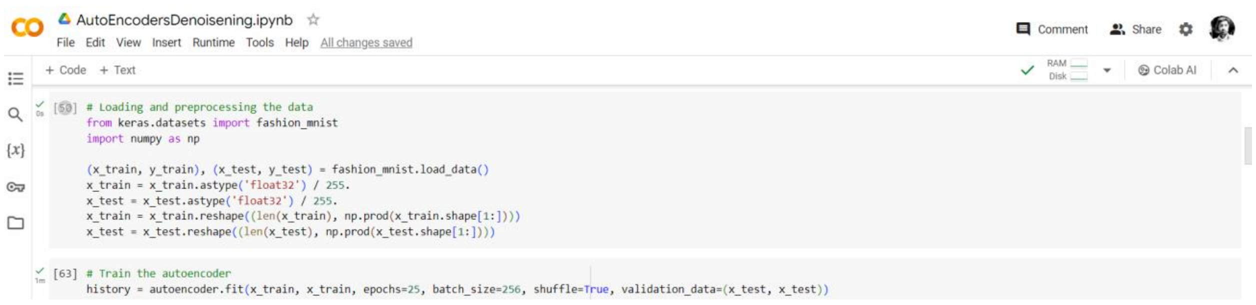
3. Repeat the question 2 on the denoisening autoencoder

4. plot loss and accuracy using the history object

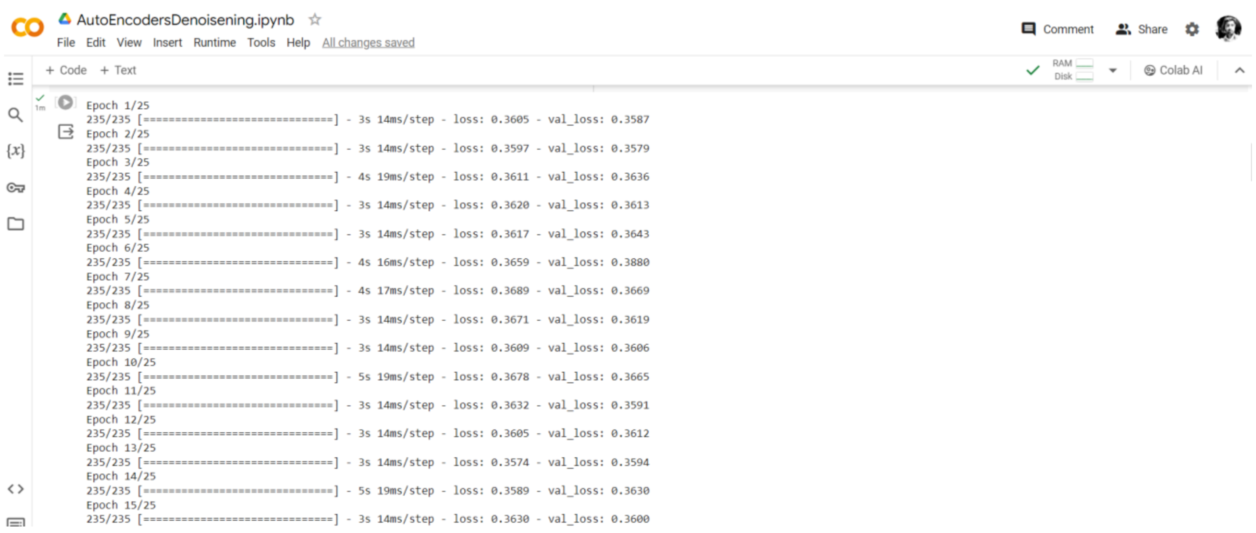


ADDING THE HIDDEN LAYER TO THE AUTOENCODER AND VISUALIZING THE DATA

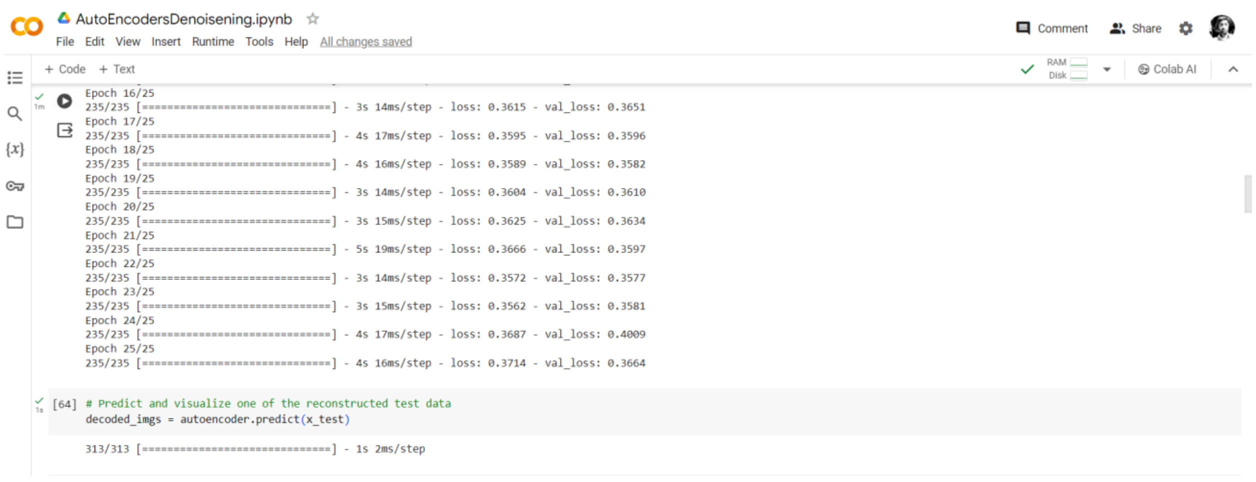
```
[43] from keras.layers import Input, Dense
     from keras.models import Model
     import matplotlib.pyplot as plt

[44] # Define the size of encoded representations and the additional hidden layer size
     encoding_dim = 32
     hidden_dim = 64

[45] # Input placeholder
     input_img = Input(shape=(784,))

[46] # First Encoding Layer
     encoded1 = Dense(hidden_dim, activation='relu')(input_img)
     # Second Encoding Layer
     encoded2 = Dense(encoding_dim, activation='relu')(encoded1)

[47] # First Decoding layer
     decoded1 = Dense(hidden_dim, activation='relu')(encoded2)
     # Second Decoding layer
     decoded = Dense(784, activation='relu')(decoded1)

[48] # Creating the autoencoder model
     autoencoder = Model(input_img, decoded)
```



```
[59] # Loading and preprocessing the data
     from keras.datasets import fashion_mnist
     import numpy as np

     (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
     x_train = x_train.astype('float32') / 255.
     x_test = x_test.astype('float32') / 255.
     x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
     x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

[63] # Train the autoencoder
     history = autoencoder.fit(x_train, x_train, epochs=25, batch_size=256, shuffle=True, validation_data=(x_test, x_test))
```



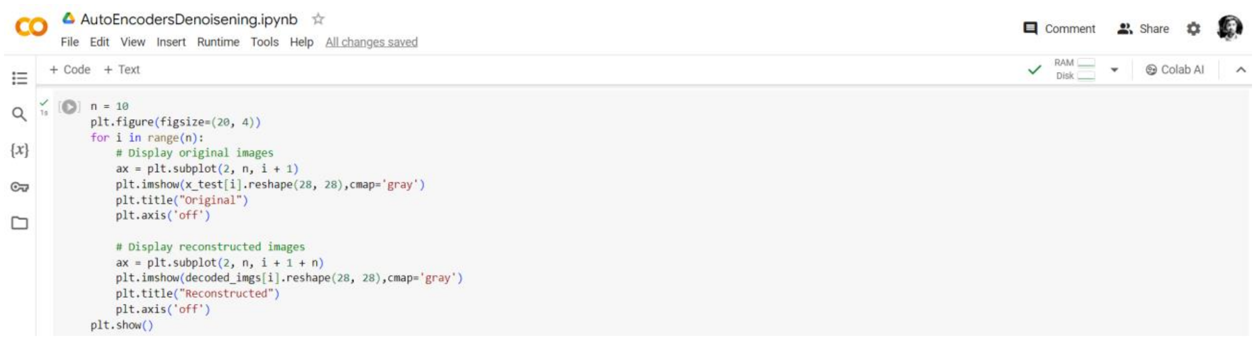
```
Epoch 1/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3605 - val_loss: 0.3587
Epoch 2/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3597 - val_loss: 0.3579
Epoch 3/25
235/235 [==============================] - 4s 19ms/step - loss: 0.3611 - val_loss: 0.3636
Epoch 4/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3620 - val_loss: 0.3613
Epoch 5/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3617 - val_loss: 0.3643
Epoch 6/25
235/235 [==============================] - 4s 16ms/step - loss: 0.3659 - val_loss: 0.3880
Epoch 7/25
235/235 [==============================] - 4s 17ms/step - loss: 0.3689 - val_loss: 0.3669
Epoch 8/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3671 - val_loss: 0.3619
Epoch 9/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3609 - val_loss: 0.3606
Epoch 10/25
235/235 [==============================] - 5s 19ms/step - loss: 0.3678 - val_loss: 0.3665
Epoch 11/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3632 - val_loss: 0.3591
Epoch 12/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3605 - val_loss: 0.3612
Epoch 13/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3574 - val_loss: 0.3594
Epoch 14/25
235/235 [==============================] - 5s 19ms/step - loss: 0.3589 - val_loss: 0.3630
Epoch 15/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3630 - val_loss: 0.3600
```


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Colab AI

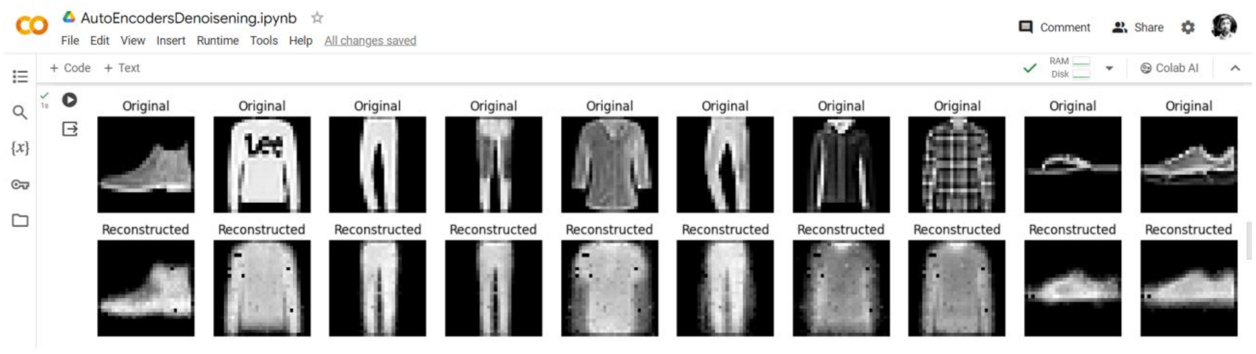
```
Epoch 16/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3615 - val_loss: 0.3651
Epoch 17/25
235/235 [==============================] - 4s 17ms/step - loss: 0.3595 - val_loss: 0.3596
Epoch 18/25
235/235 [==============================] - 4s 16ms/step - loss: 0.3589 - val_loss: 0.3582
Epoch 19/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3604 - val_loss: 0.3610
Epoch 20/25
235/235 [==============================] - 3s 15ms/step - loss: 0.3625 - val_loss: 0.3634
Epoch 21/25
235/235 [==============================] - 5s 19ms/step - loss: 0.3666 - val_loss: 0.3597
Epoch 22/25
235/235 [==============================] - 3s 14ms/step - loss: 0.3572 - val_loss: 0.3577
Epoch 23/25
235/235 [==============================] - 3s 15ms/step - loss: 0.3562 - val_loss: 0.3581
Epoch 24/25
235/235 [==============================] - 4s 17ms/step - loss: 0.3687 - val_loss: 0.4009
Epoch 25/25
235/235 [==============================] - 4s 16ms/step - loss: 0.3714 - val_loss: 0.3664
```

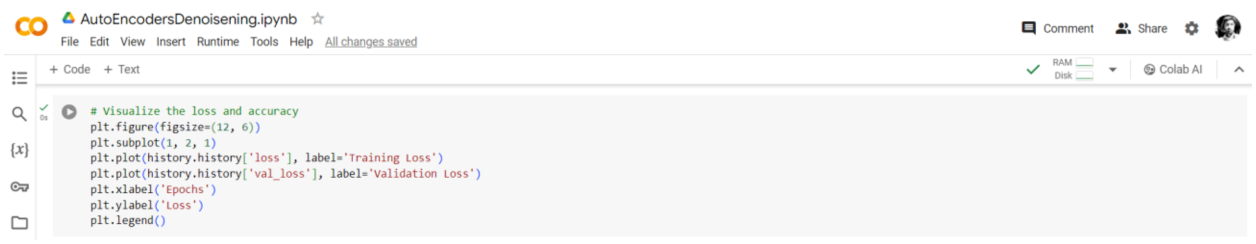
```
[64] # Predict and visualize one of the reconstructed test data
     decoded_imgs = autoencoder.predict(x_test)
```

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

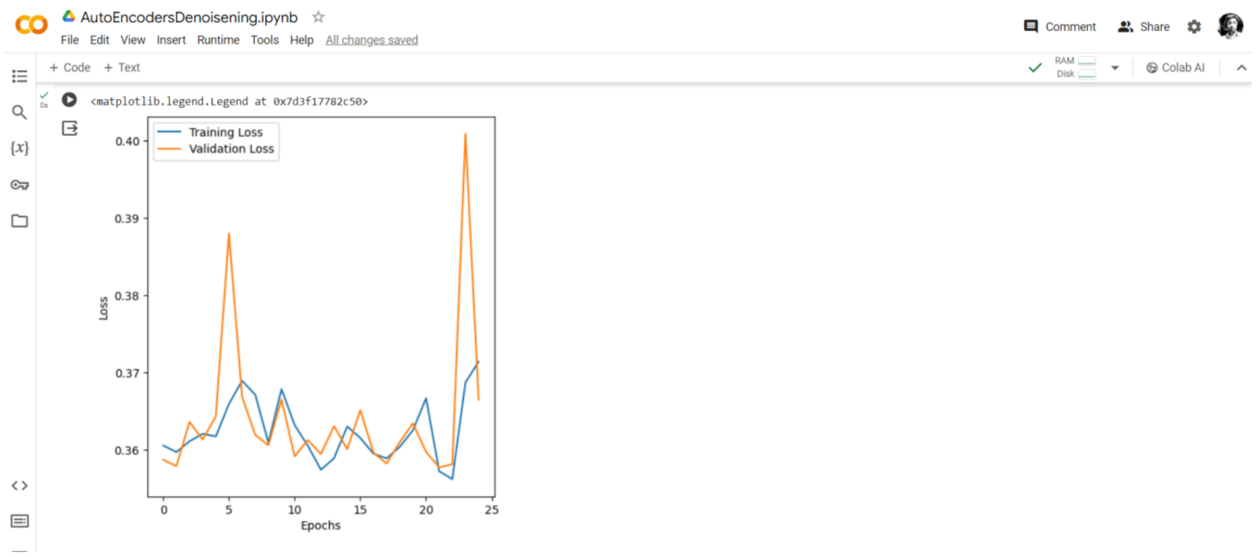

AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Colab AI

```
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[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(decoded_imgs[i].reshape(28, 28),cmap='gray')
    plt.title("Reconstructed")
    plt.axis('off')
plt.show()
```

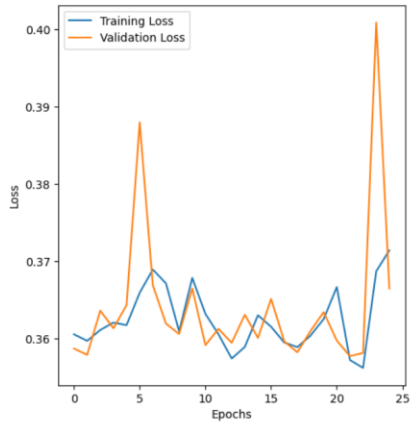


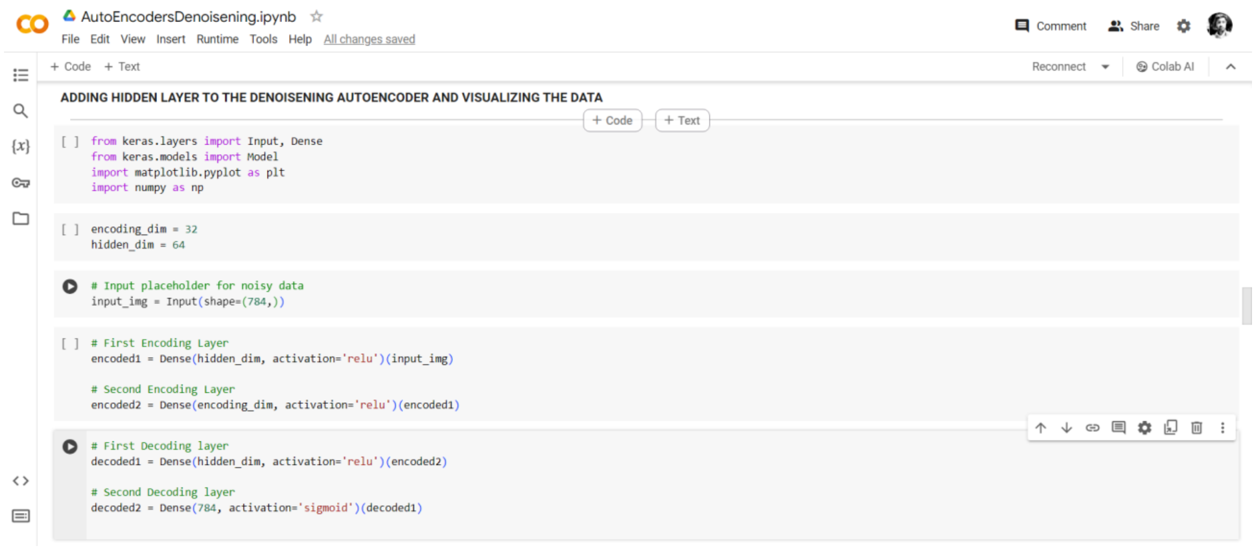

AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Colab AI

```
# Visualize the loss and accuracy
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Colab AI

```
<matplotlib.legend.Legend at 0x7d3f17782c50>
```




AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI

ADDING HIDDEN LAYER TO THE DENOISENING AUTOENCODER AND VISUALIZING THE DATA

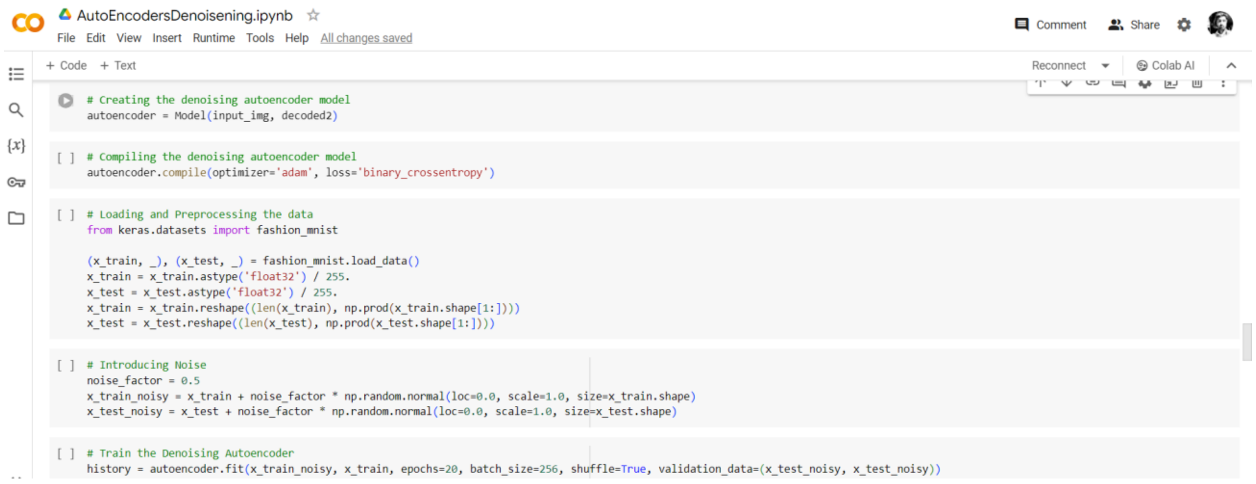
```
from keras.layers import Input, Dense
from keras.models import Model
import matplotlib.pyplot as plt
import numpy as np
```

```
encoding_dim = 32
hidden_dim = 64
```

```
# Input placeholder for noisy data
input_img = Input(shape=(784,))
```

```
# First Encoding Layer
encoded1 = Dense(hidden_dim, activation='relu')(input_img)

# Second Encoding Layer
encoded2 = Dense(encoding_dim, activation='relu')(encoded1)
```

```
# First Decoding layer
decoded1 = Dense(hidden_dim, activation='relu')(encoded2)

# Second Decoding layer
decoded2 = Dense(784, activation='sigmoid')(decoded1)
```


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI

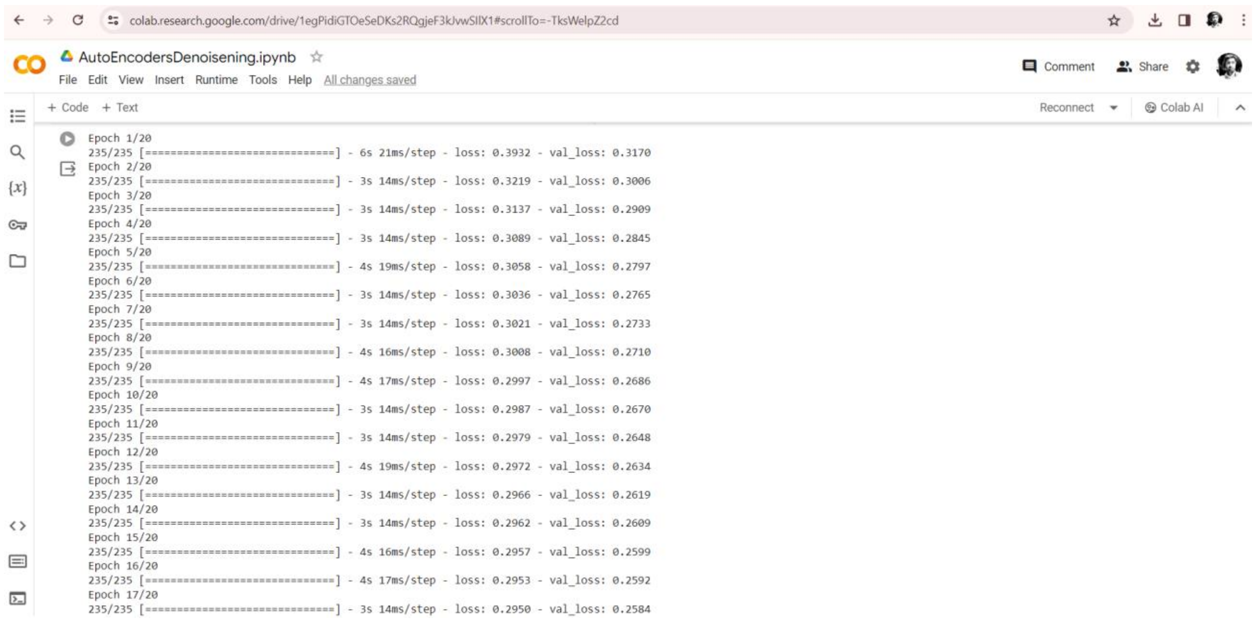
```
# Creating the denoising autoencoder model
autoencoder = Model(input_img, decoded2)
```

```
# Compiling the denoising autoencoder model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

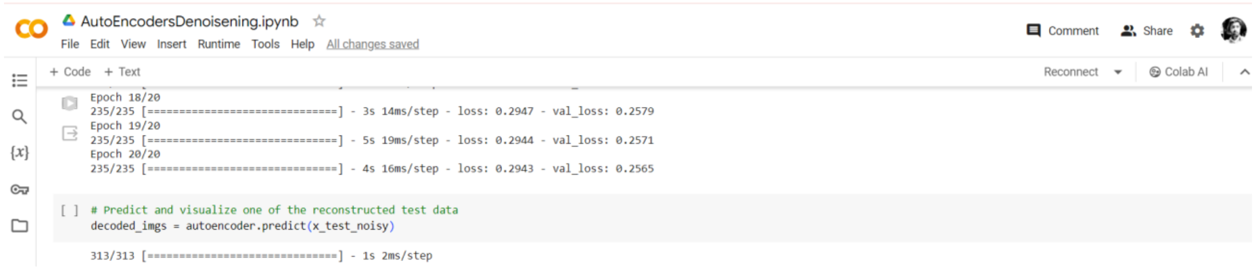
```
# Loading and Preprocessing the data
from keras.datasets import fashion_mnist

(x_train, _), (x_test, _) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
```

```
# Introducing Noise
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)
```

```
# Train the Denoising Autoencoder
history = autoencoder.fit(x_train_noisy, x_train, epochs=20, batch_size=256, shuffle=True, validation_data=(x_test_noisy, x_test_noisy))
```


colab.research.google.com/drive/1egPidiGTOeSeDKs2RQgjeF3kJvwSIlX1#scrollTo=-TksWelpZ2cd
AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI

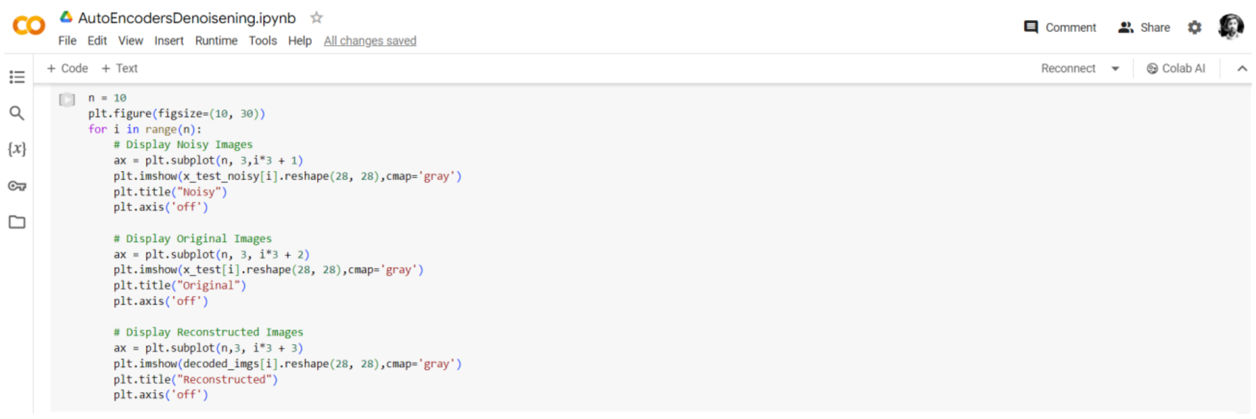
```
Epoch 1/20
235/235 [==============================] - 6s 21ms/step - loss: 0.3932 - val_loss: 0.3170
Epoch 2/20
235/235 [==============================] - 3s 14ms/step - loss: 0.3219 - val_loss: 0.3006
Epoch 3/20
235/235 [==============================] - 3s 14ms/step - loss: 0.3137 - val_loss: 0.2909
Epoch 4/20
235/235 [==============================] - 3s 14ms/step - loss: 0.3089 - val_loss: 0.2845
Epoch 5/20
235/235 [==============================] - 4s 19ms/step - loss: 0.3058 - val_loss: 0.2797
Epoch 6/20
235/235 [==============================] - 3s 14ms/step - loss: 0.3036 - val_loss: 0.2765
Epoch 7/20
235/235 [==============================] - 3s 14ms/step - loss: 0.3021 - val_loss: 0.2733
Epoch 8/20
235/235 [==============================] - 4s 16ms/step - loss: 0.3008 - val_loss: 0.2710
Epoch 9/20
235/235 [==============================] - 4s 17ms/step - loss: 0.2997 - val_loss: 0.2686
Epoch 10/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2987 - val_loss: 0.2670
Epoch 11/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2979 - val_loss: 0.2648
Epoch 12/20
235/235 [==============================] - 4s 19ms/step - loss: 0.2972 - val_loss: 0.2634
Epoch 13/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2966 - val_loss: 0.2619
Epoch 14/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2962 - val_loss: 0.2609
Epoch 15/20
235/235 [==============================] - 4s 16ms/step - loss: 0.2957 - val_loss: 0.2599
Epoch 16/20
235/235 [==============================] - 4s 17ms/step - loss: 0.2953 - val_loss: 0.2592
Epoch 17/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2950 - val_loss: 0.2584
```


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI

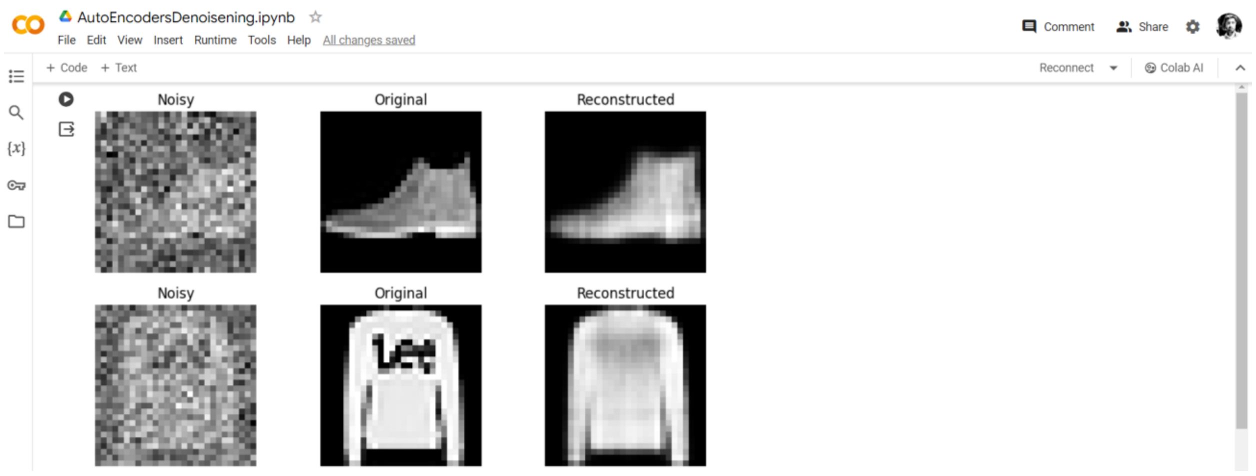
```
Epoch 18/20
235/235 [==============================] - 3s 14ms/step - loss: 0.2947 - val_loss: 0.2579
Epoch 19/20
235/235 [==============================] - 5s 19ms/step - loss: 0.2944 - val_loss: 0.2571
Epoch 20/20
235/235 [==============================] - 4s 16ms/step - loss: 0.2943 - val_loss: 0.2565
```

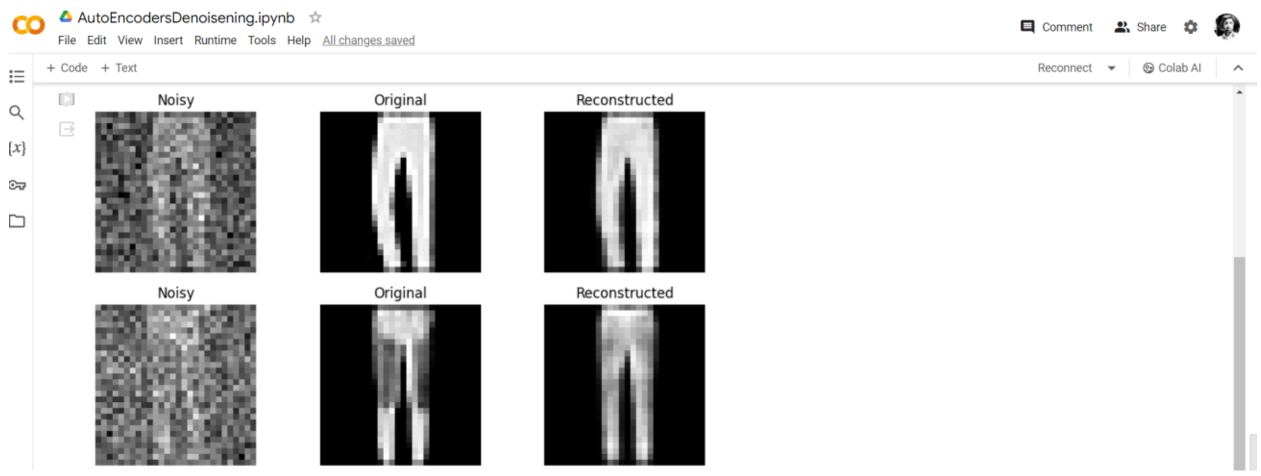
```
[ ] # Predict and visualize one of the reconstructed test data
    decoded_imgs = autoencoder.predict(x_test_noisy)
```

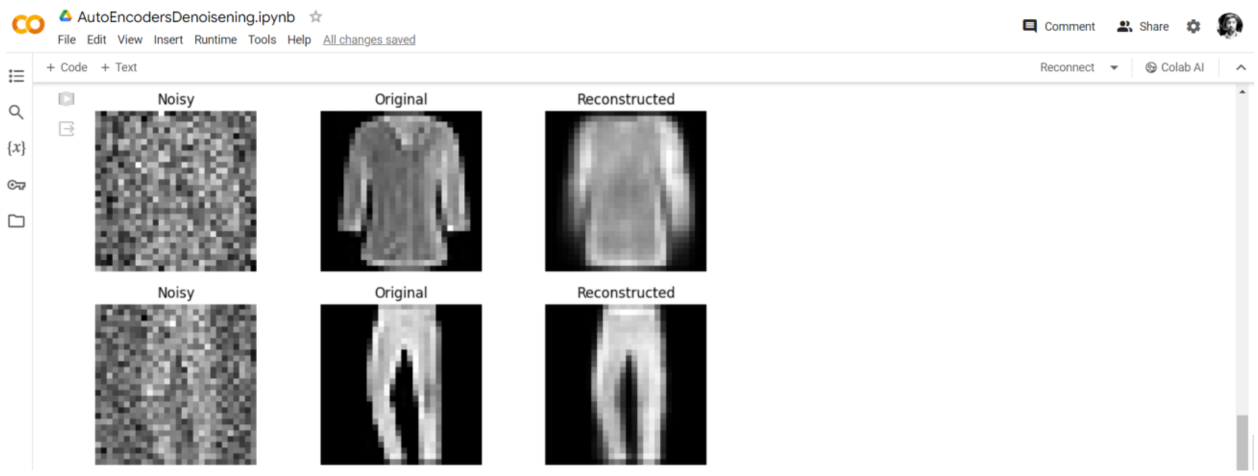
```
313/313 [==============================] - 1s 2ms/step
```

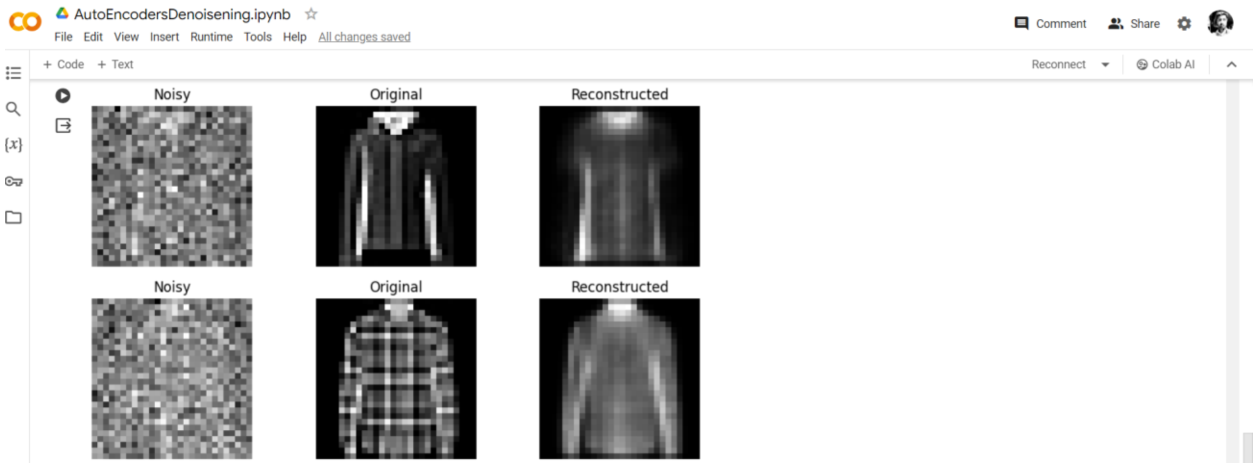

AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI

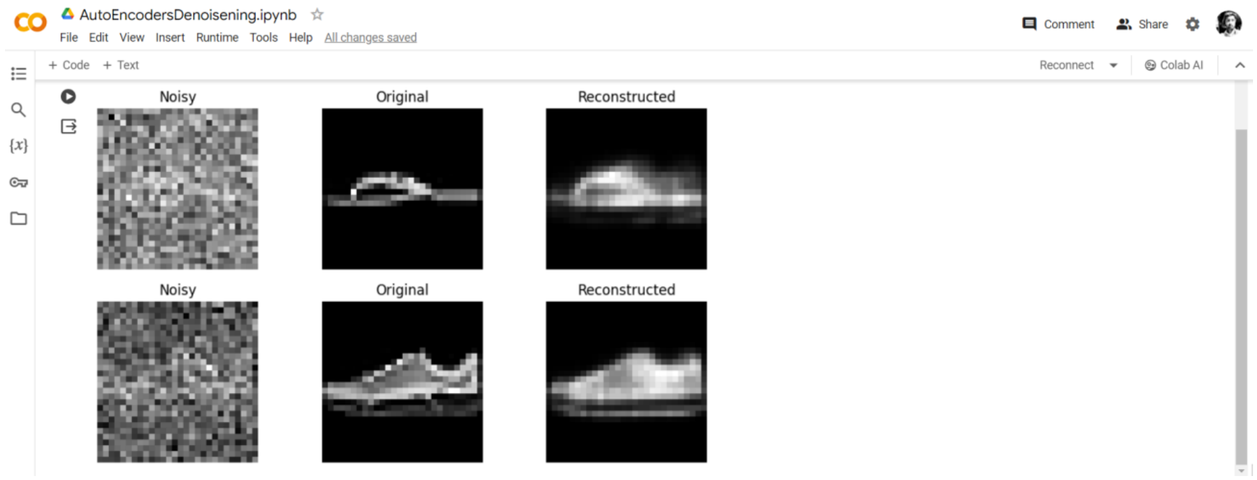
```
n = 10
plt.figure(figsize=(10, 30))
for i in range(n):
    # Display Noisy Images
    ax = plt.subplot(n, 3,i*3 + 1)
    plt.imshow(x_test_noisy[i].reshape(28, 28),cmap='gray')
    plt.title("Noisy")
    plt.axis('off')

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

    # Display Reconstructed Images
    ax = plt.subplot(n,3, i*3 + 3)
    plt.imshow(decoded_imgs[i].reshape(28, 28),cmap='gray')
    plt.title("Reconstructed")
    plt.axis('off')
```

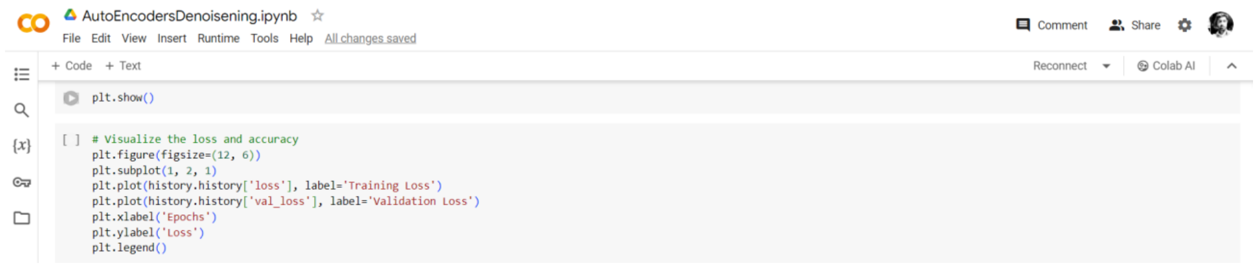

AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect Colab AI
Noisy
Original
Reconstructed
Noisy
Original
Reconstructed


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect Colab AI
Noisy
Original
Reconstructed
Noisy
Original
Reconstructed


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect Colab AI
Noisy
Original
Reconstructed
Noisy
Original
Reconstructed

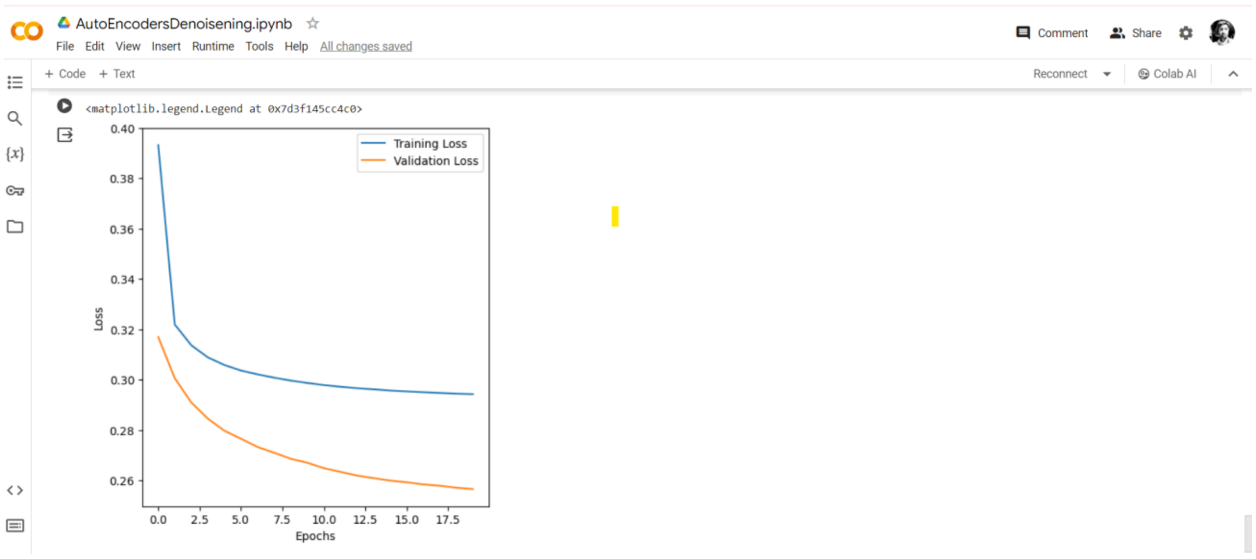




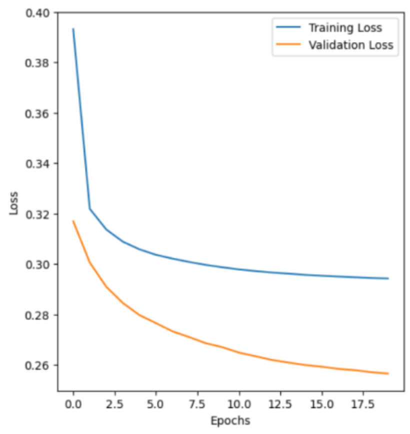

AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect Colab AI

```
plt.show()
```

```
# Visualize the loss and accuracy
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```


AutoEncodersDenoisening.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Colab AI

```
<matplotlib.legend.Legend at 0x7d3f145cc4c0>
```


Training Loss
Validation Loss
Loss
Epochs
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5