Spring 2024:CS5720 Neural Networks & Deep Learning-

ICP-9 Assignment-8

Name: Sai Sushma Sri Bireddy Student ID:700747557

GitHub Link: https://github.com/SaiSushmaSriBireddy/Assignment_8

Video Link:
https://drive.google.com/file/d/1RtWez4GtrKN29ZQqRAxdJ1DtgHKmYeOk/view?usp=sharing

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

Programming elements:

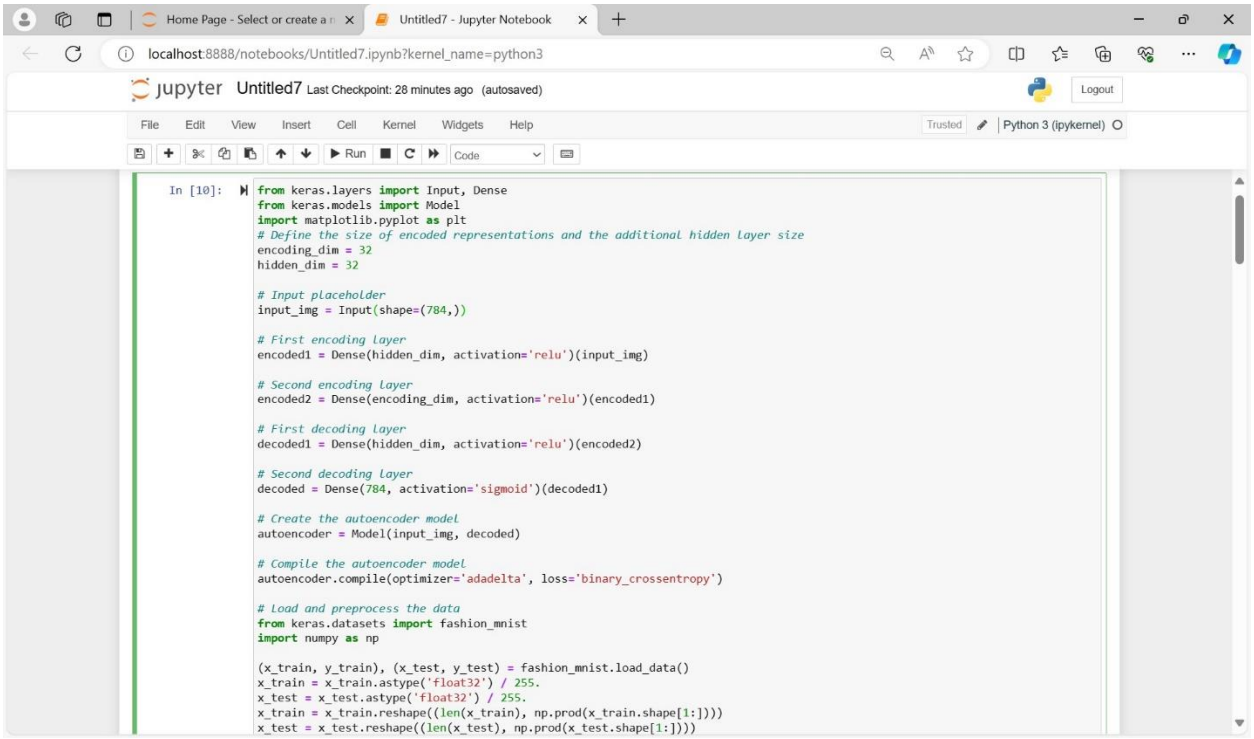
1. Basics of Autoencoders

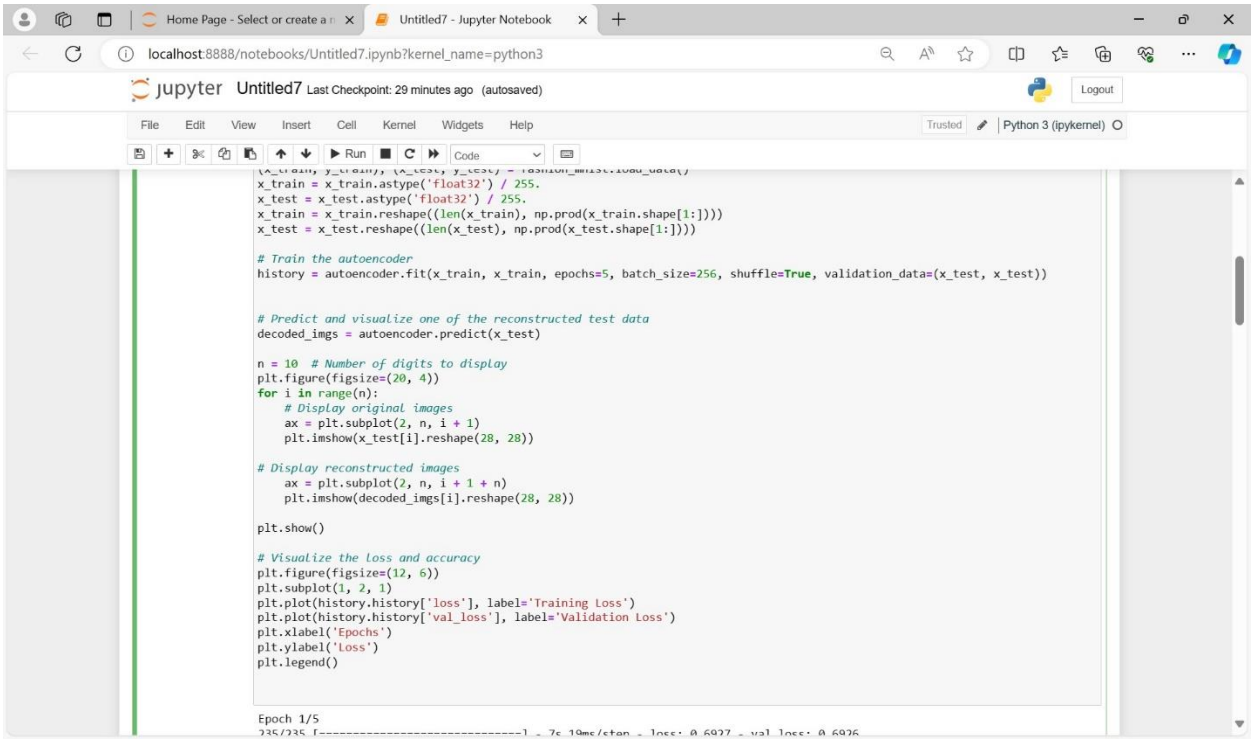
2. Role of Autoencoders in unsupervised learning

3. Types of Autoencoders

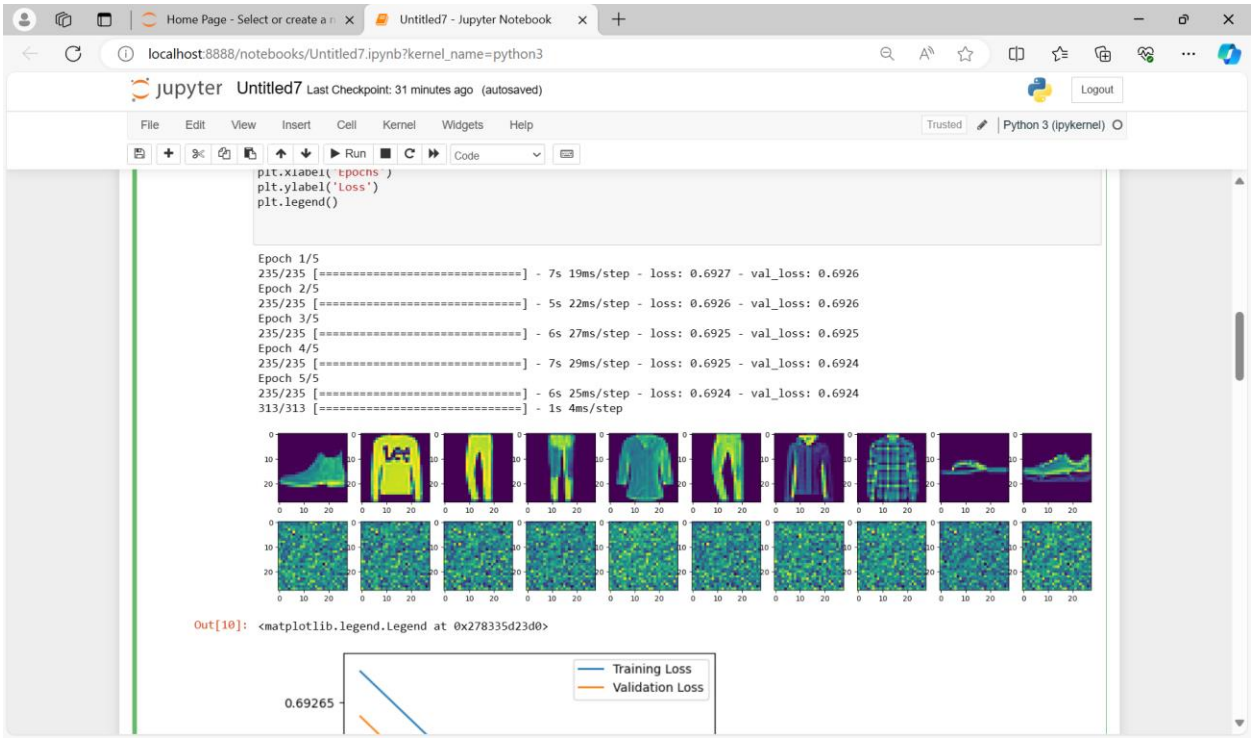
4. Use case: Simple autoencoder-Reconstructing the existing image, which will contain most important features of the image

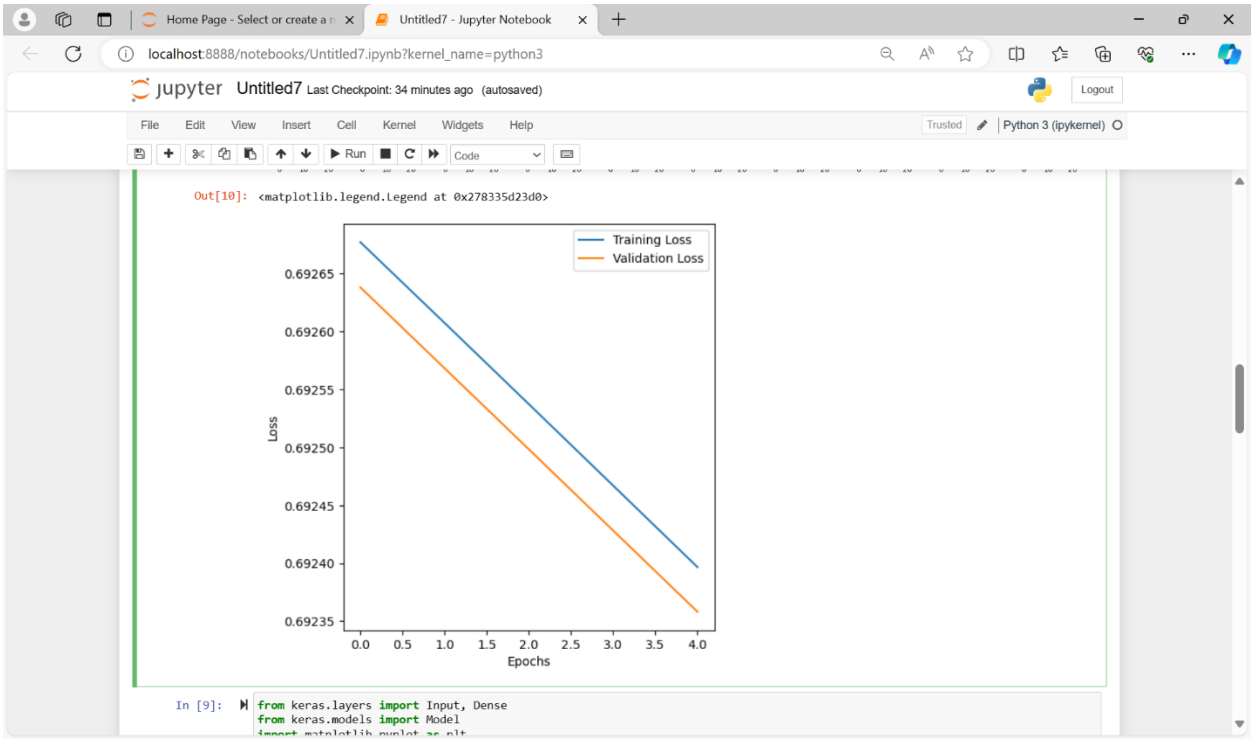
5. Use case: Stacked autoencoder




Home Page - Select or create a n
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 29 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
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:])))
# Train the autoencoder
history = autoencoder.fit(x_train, x_train, epochs=5, batch_size=256, shuffle=True, validation_data=(x_test, x_test))
# Predict and visualize one of the reconstructed test data
decoded_imgs = autoencoder.predict(x_test)
n = 10 # Number of digits to display
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))
# Display reconstructed images
ax = plt.subplot(2, n, i + 1 + n)
plt.imshow(decoded_imgs[i].reshape(28, 28))
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()
Epoch 1/5

Output:


Home Page - Select or create a n
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 31 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
Epoch 1/5
235/235 [==========] - 7s 19ms/step - loss: 0.6927 - val_loss: 0.6926
Epoch 2/5
235/235 [==========] - 5s 22ms/step - loss: 0.6926 - val_loss: 0.6926
Epoch 3/5
235/235 [==========] - 6s 27ms/step - loss: 0.6925 - val_loss: 0.6925
Epoch 4/5
235/235 [==========] - 7s 29ms/step - loss: 0.6925 - val_loss: 0.6924
Epoch 5/5
235/235 [==========] - 6s 25ms/step - loss: 0.6924 - val_loss: 0.6924
313/313 [==========] - 1s 4ms/step
Out[10]: <matplotlib.legend.Legend at 0x278335d23d0>
0.69265
Training Loss
Validation Loss



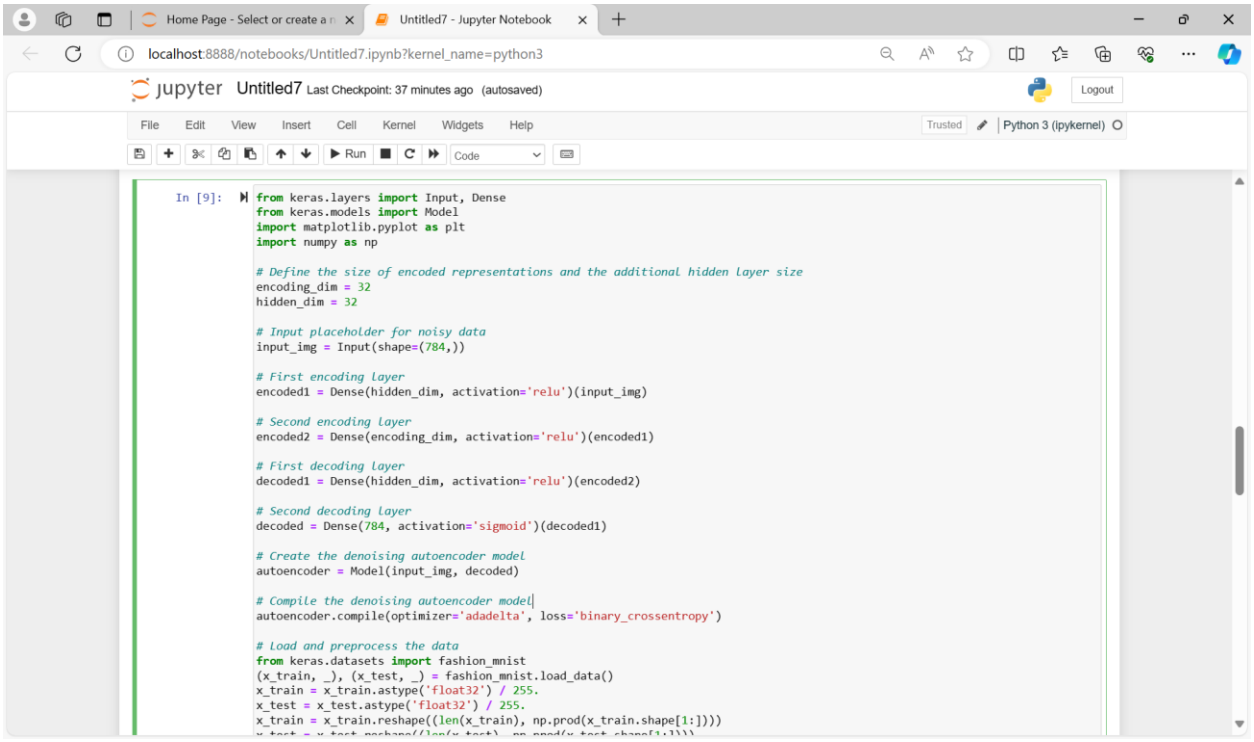
In class programming:

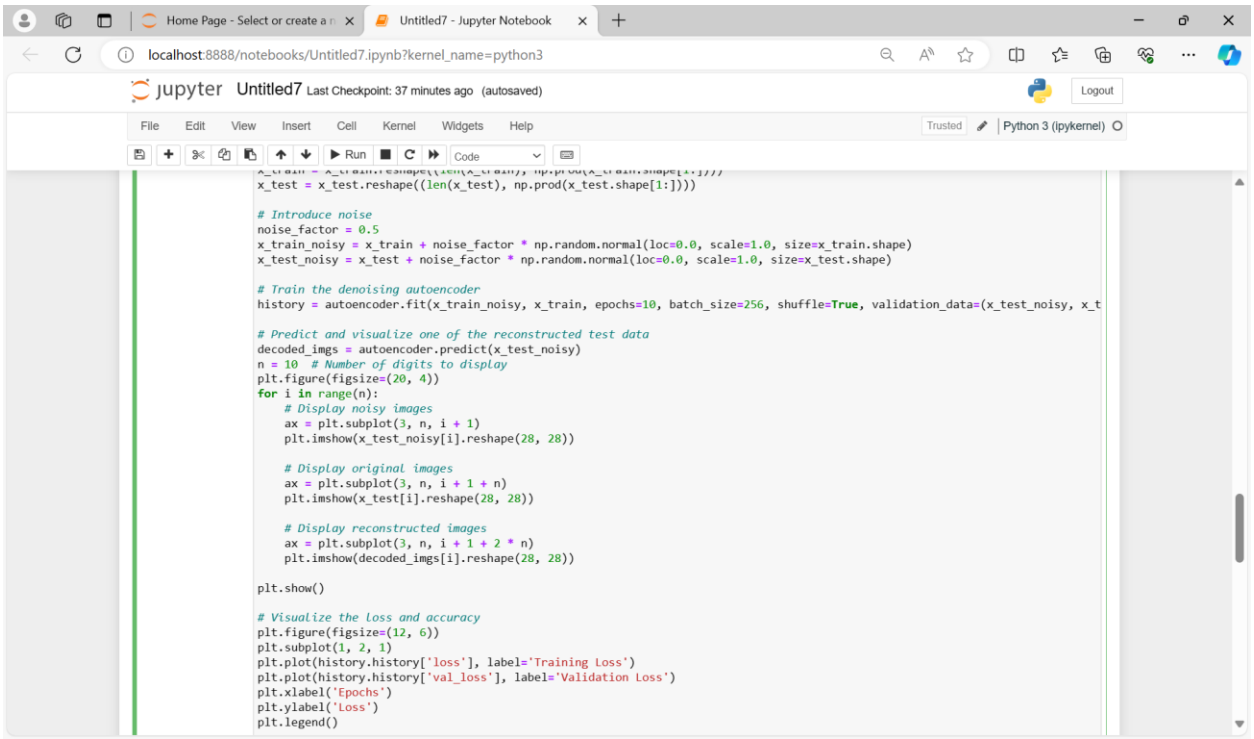
1. Add one more hidden layer to autoencoder

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


Home Page - Select or create a n
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 37 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [9]:
from keras.layers import Input, Dense
from keras.models import Model
import matplotlib.pyplot as plt
import numpy as np
# Define the size of encoded representations and the additional hidden layer size
encoding_dim = 32
hidden_dim = 32
# 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
decoded = Dense(784, activation='sigmoid')(decoded1)
# Create the denoising autoencoder model
autoencoder = Model(input_img, decoded)
# Compile the denoising autoencoder model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
# Load and preprocess 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:])))


Home Page - Select or create a n
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 37 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
# Introduce 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=10, batch_size=256, shuffle=True, validation_data=(x_test_noisy, x_t
# Predict and visualize one of the reconstructed test data
decoded_imgs = autoencoder.predict(x_test_noisy)
n = 10 # Number of digits to display
plt.figure(figsize=(20, 4))
for i in range(n):
# Display noisy images
ax = plt.subplot(3, n, i + 1)
plt.imshow(x_test_noisy[i].reshape(28, 28))
# Display original images
ax = plt.subplot(3, n, i + 1 + n)
plt.imshow(x_test[i].reshape(28, 28))
# Display reconstructed images
ax = plt.subplot(3, n, i + 1 + 2 * n)
plt.imshow(decoded_imgs[i].reshape(28, 28))
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()

Output:

Home Page - Select or create a n | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3

jupyter **Untitled7** Last Checkpoint: 39 minutes ago (autosaved)

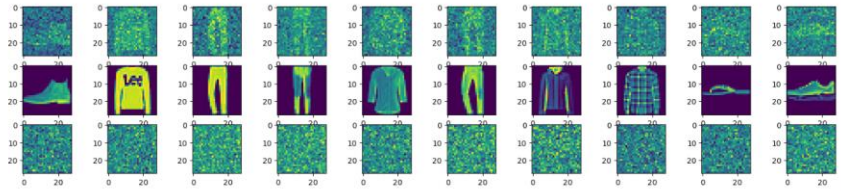
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Code

```
Epoch 1/10
235/235 [==============================] - 10s 34ms/step - loss: 0.6927 - val_loss: 0.6926
Epoch 2/10
235/235 [==============================] - 7s 29ms/step - loss: 0.6926 - val_loss: 0.6926
Epoch 3/10
235/235 [==============================] - 5s 22ms/step - loss: 0.6925 - val_loss: 0.6925
Epoch 4/10
235/235 [==============================] - 5s 21ms/step - loss: 0.6925 - val_loss: 0.6924
Epoch 5/10
235/235 [==============================] - 4s 18ms/step - loss: 0.6924 - val_loss: 0.6924
Epoch 6/10
235/235 [==============================] - 4s 15ms/step - loss: 0.6923 - val_loss: 0.6923
Epoch 7/10
235/235 [==============================] - 4s 18ms/step - loss: 0.6923 - val_loss: 0.6923
Epoch 8/10
235/235 [==============================] - 5s 22ms/step - loss: 0.6922 - val_loss: 0.6922
Epoch 9/10
235/235 [==============================] - 6s 25ms/step - loss: 0.6922 - val_loss: 0.6921
Epoch 10/10
235/235 [==============================] - 6s 26ms/step - loss: 0.6921 - val_loss: 0.6921
313/313 [==============================] - 2s 4ms/step
```



Out[9]: <matplotlib.legend.Legend at 0x27833594d90>

Home Page - Select or create a n | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/Untitled7.ipynb?kernel_name=python3

jupyter **Untitled7** Last Checkpoint: 39 minutes ago (autosaved)

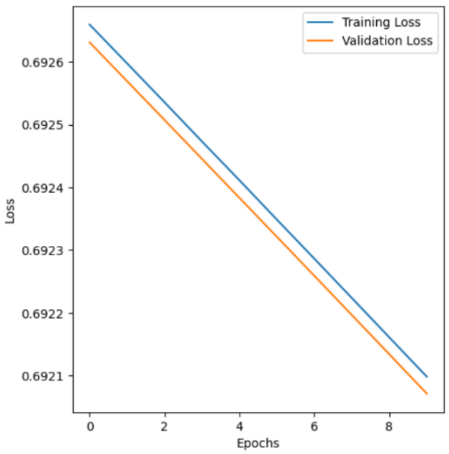
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Code

Out[9]: <matplotlib.legend.Legend at 0x27833594d90>



In [ ]: