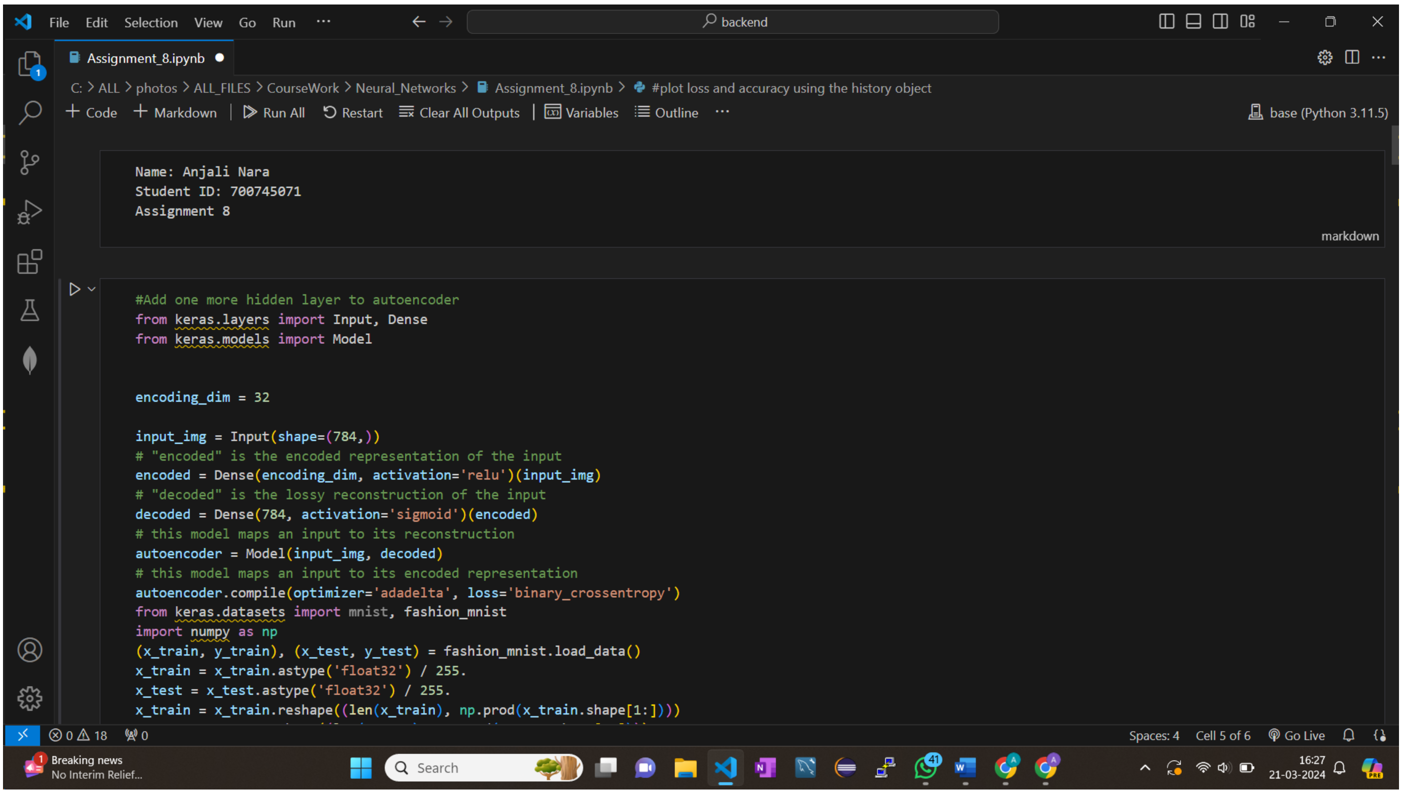
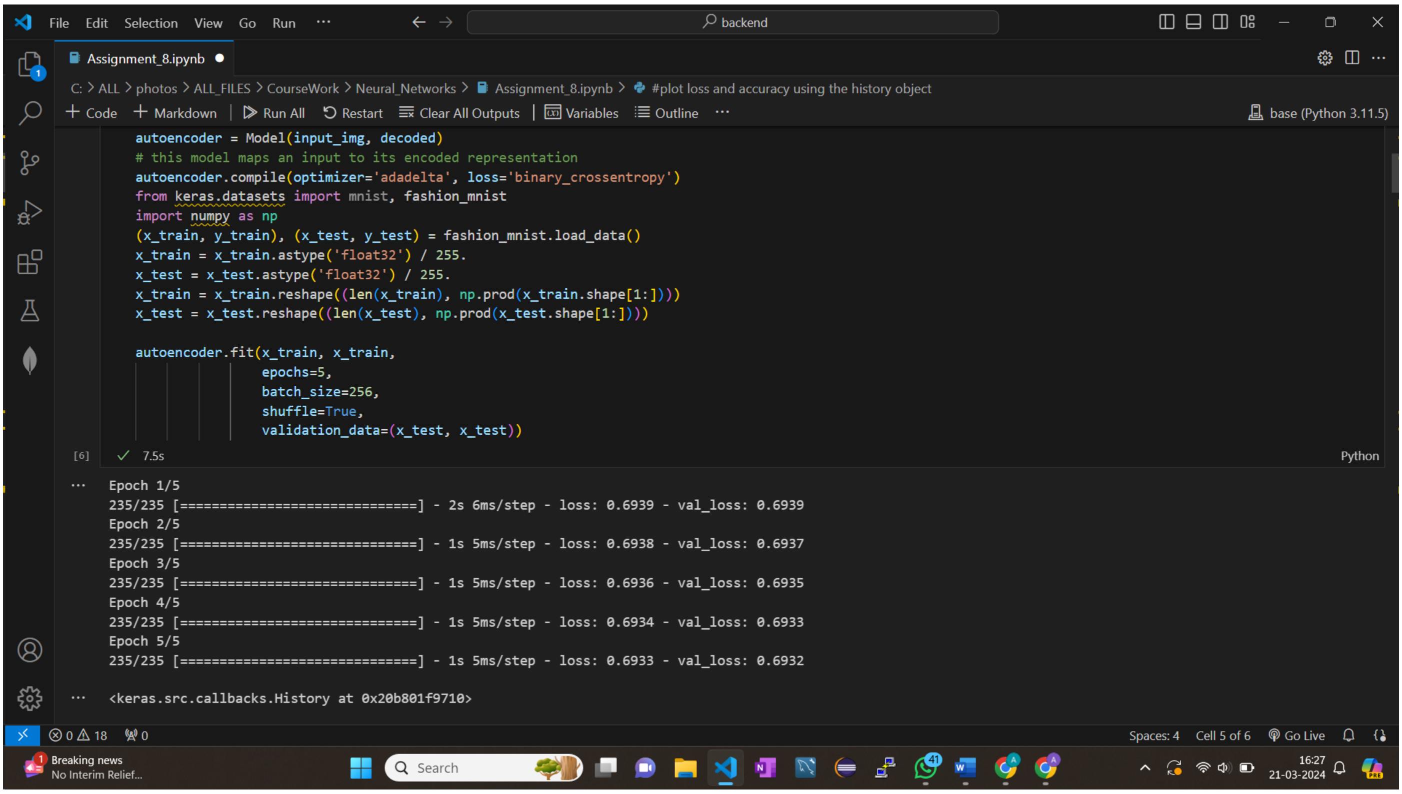
Anjali Nara | 700745071 | Assignment 8

Video Link: https://drive.google.com/file/d/1R_daHBknReThxbpqi78lfFrDac5k4RbD/view?usp=sharing

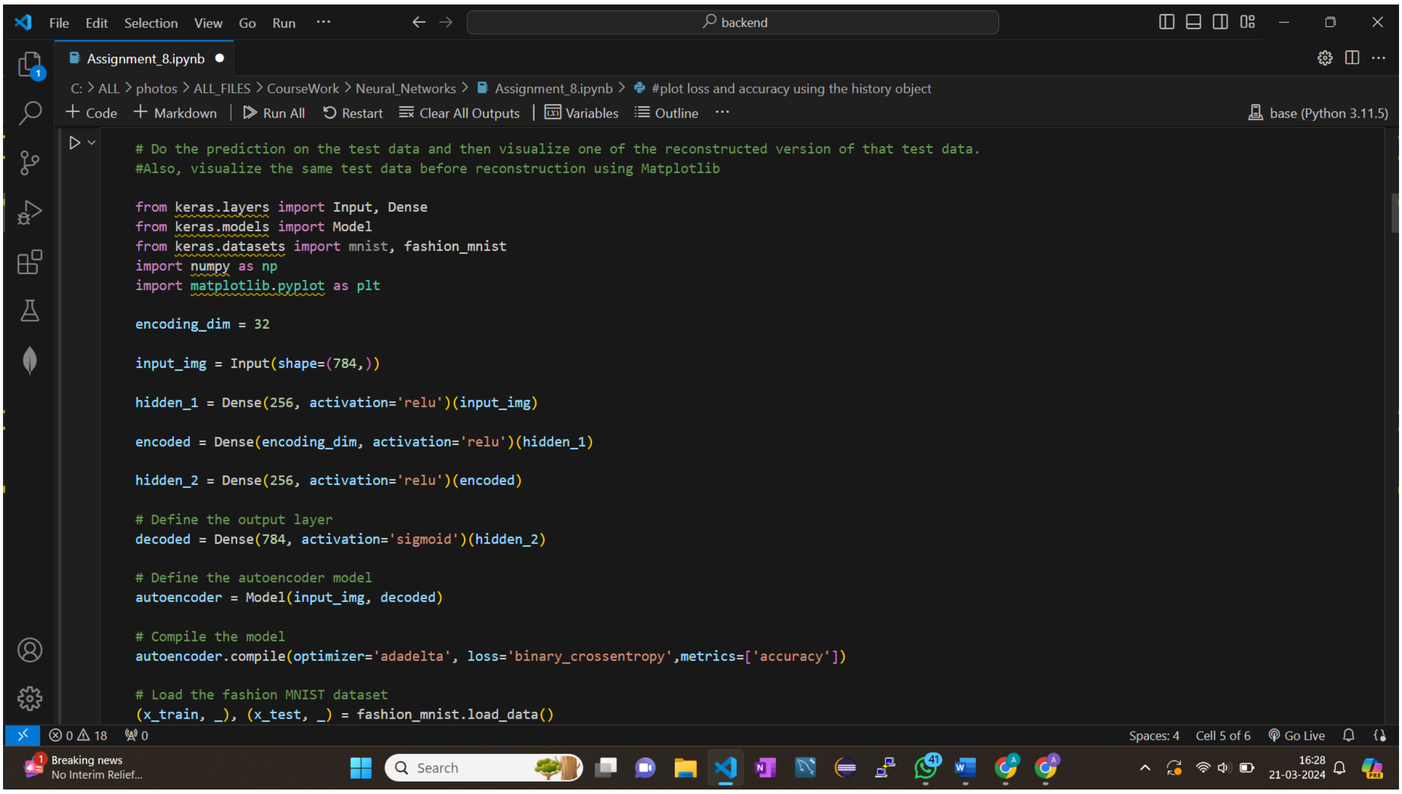
Github Link: https://github.com/naraanjali/Neural-assignment-8/tree/main

1. Add one more hidden layer to autoencoder

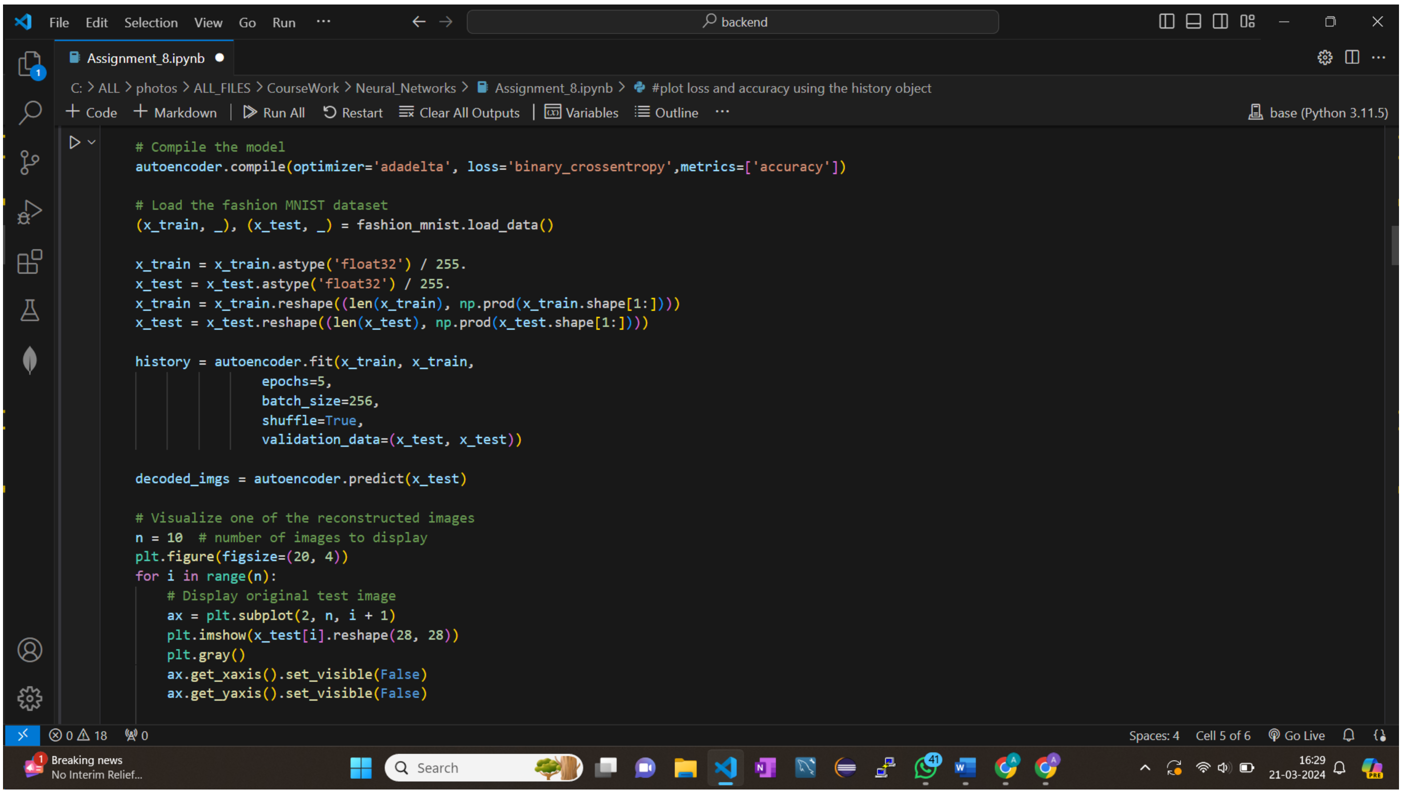




2. Do the prediction on the test data and then visualize one of the reconstructed version of that test data. Also, visualize the same test data before reconstruction using Matplotlib



```python
# 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

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

encoding_dim = 32

input_img = Input(shape=(784,))

hidden_1 = Dense(256, activation='relu')(input_img)

encoded = Dense(encoding_dim, activation='relu')(hidden_1)

hidden_2 = Dense(256, activation='relu')(encoded)

# Define the output layer
decoded = Dense(784, activation='sigmoid')(hidden_2)

# Define the autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy',metrics=['accuracy'])

# Load the fashion MNIST dataset
(x_train, _), (x_test, _) = fashion_mnist.load_data()
```

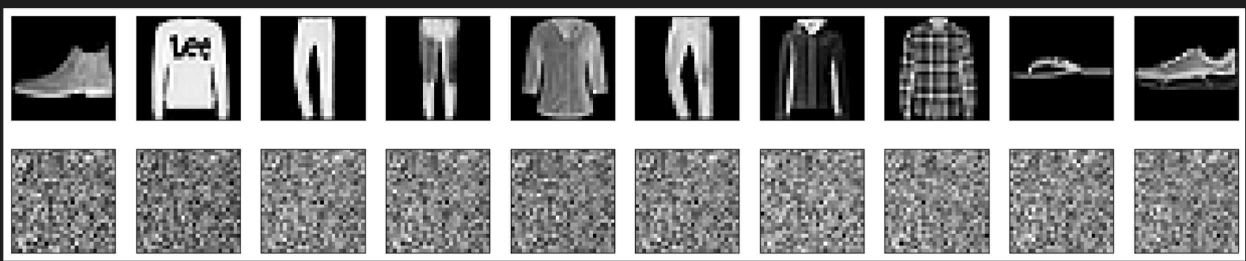


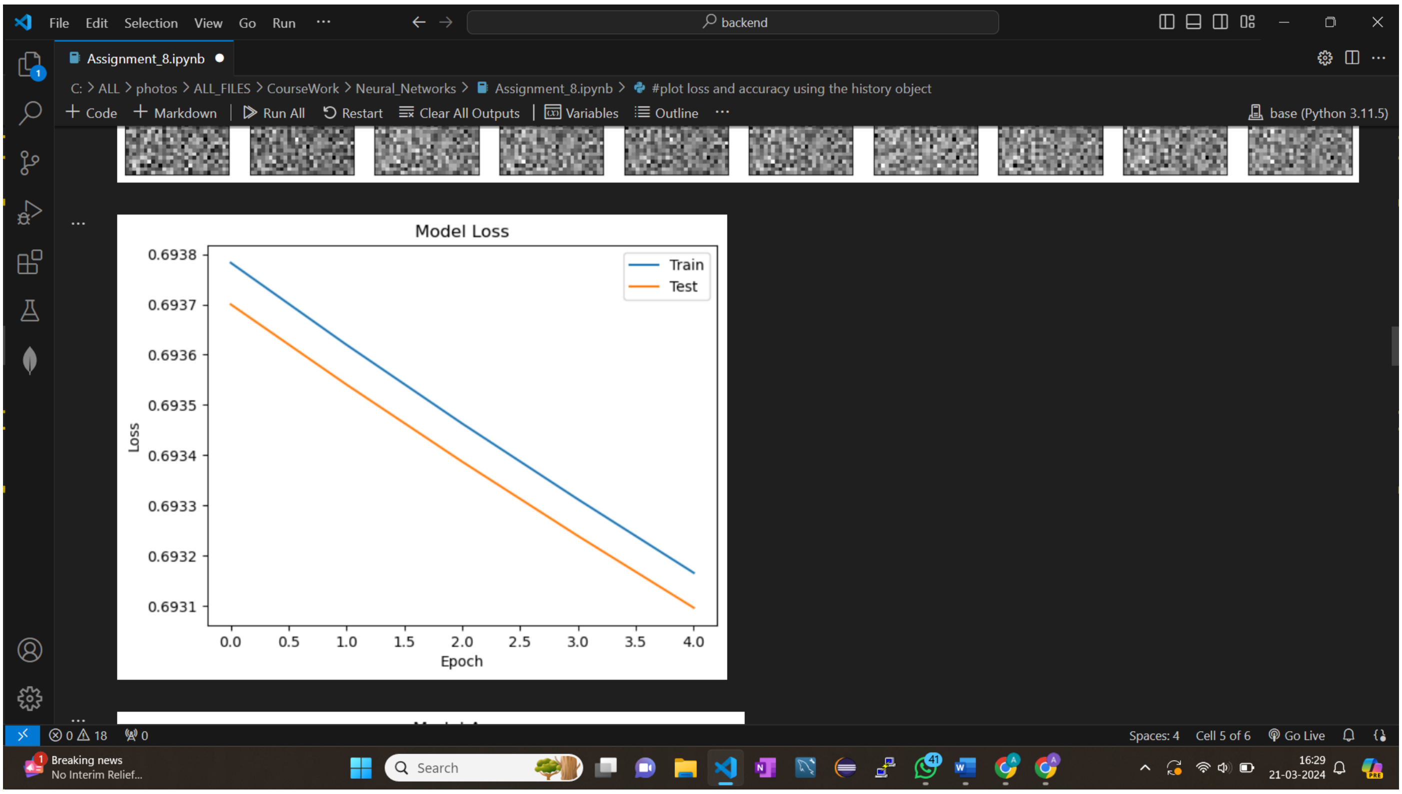
```python
# Compile the model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy',metrics=['accuracy'])

# Load the fashion MNIST dataset
(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:])))

history = autoencoder.fit(x_train, x_train,
                epochs=5,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

decoded_imgs = autoencoder.predict(x_test)

# Visualize one of the reconstructed images
n = 10  # number of images to display
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original test image
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

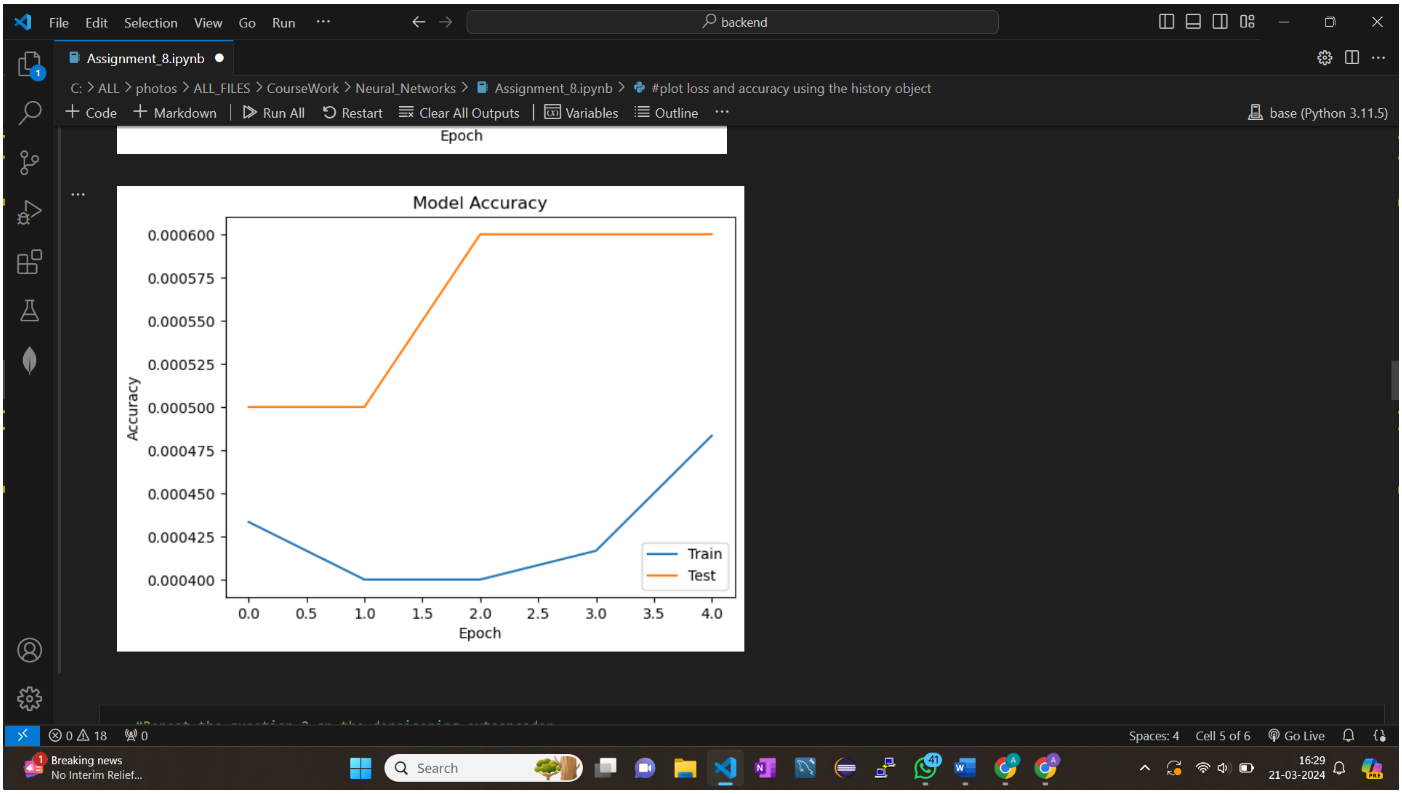
```python
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

        # Display reconstructed test image
        ax = plt.subplot(2, n, i + 1 + n)
        plt.imshow(decoded_imgs[i].reshape(28, 28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
    plt.show()

    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper right')
    plt.show()

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='lower right')
    plt.show()
```

[7] ✓ 14.2s

```
Epoch 1/5
235/235 [==============================] - 3s 12ms/step - loss: 0.6938 - accuracy: 4.3333e-04 - val_loss: 0.6937 - val_accuracy: 5.0000e-04
Epoch 2/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6936 - accuracy: 4.0000e-04 - val_loss: 0.6935 - val_accuracy: 5.0000e-04
Epoch 3/5
235/235 [==============================] - 2s 9ms/step - loss: 0.6935 - accuracy: 4.0000e-04 - val_loss: 0.6934 - val_accuracy: 6.0000e-04
Epoch 4/5
235/235 [==============================] - 2s 9ms/step - loss: 0.6933 - accuracy: 4.1667e-04 - val_loss: 0.6932 - val_accuracy: 6.0000e-04
Epoch 5/5
235/235 [==============================] - 2s 9ms/step - loss: 0.6932 - accuracy: 4.8333e-04 - val_loss: 0.6931 - val_accuracy: 6.0000e-04
313/313 [==============================] - 1s 1ms/step
```




File Edit Selection View Go Run
backend
Assignment_8.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_8.ipynb > #plot loss and accuracy using the history object
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.5)
Model Loss
Train
Test
0.6938
0.6937
0.6936
0.6935
0.6934
0.6933
0.6932
0.6931
Loss
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Epoch
Spaces: 4 Cell 5 of 6 Go Live
Breaking news
No Interim Relief...
Search
16:29
21-03-2024


File Edit Selection View Go Run
backend
Assignment_8.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_8.ipynb > #plot loss and accuracy using the history object
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.5)
Epoch
Model Accuracy
0.000600
0.000575
0.000550
0.000525
0.000500
0.000475
0.000450
0.000425
0.000400
Accuracy
Train
Test
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Epoch
Spaces: 4 Cell 5 of 6 Go Live
Breaking news
No Interim Relief...
Search
16:29
21-03-2024

3. Repeat the question 2 on the denoisening autoencoder

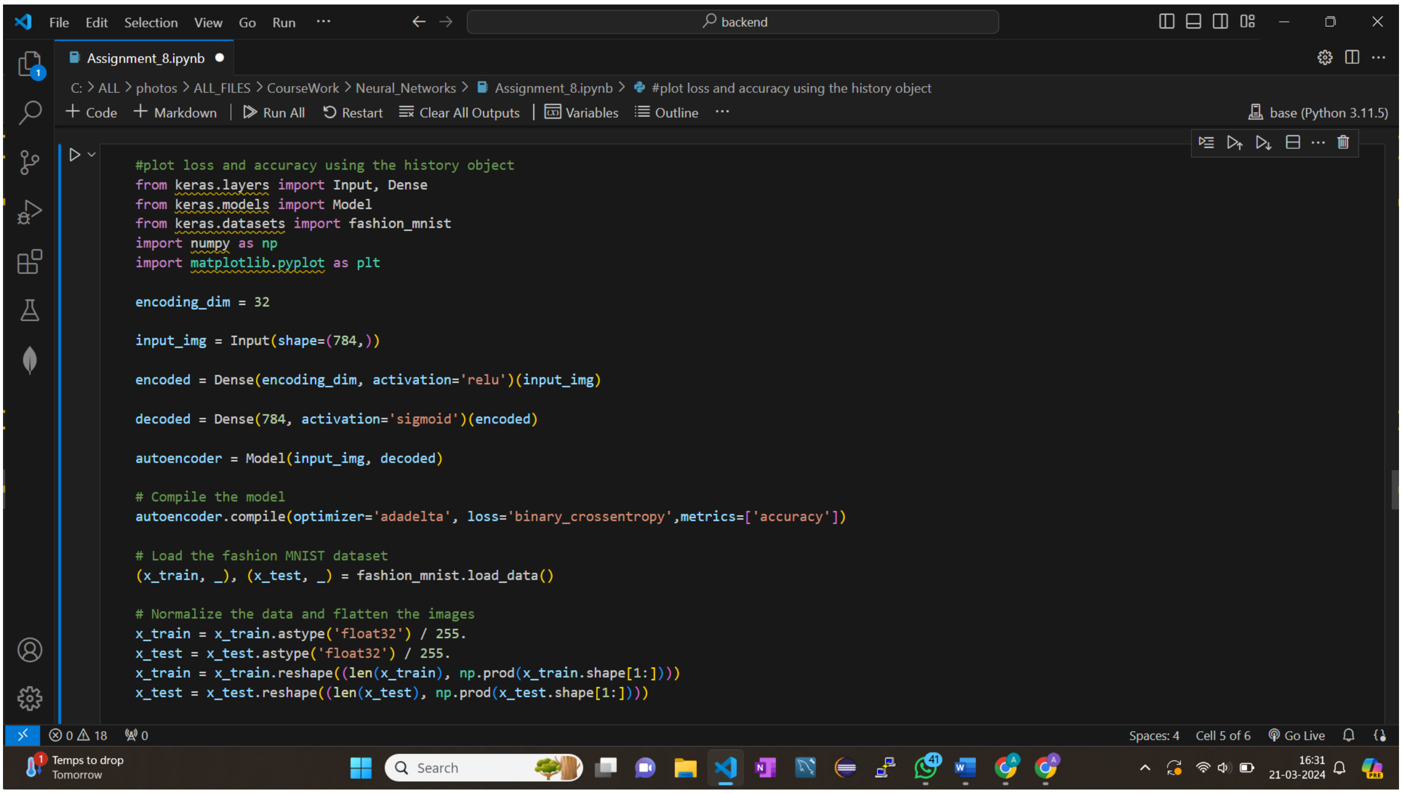
```python
#Repeat the question 2 on the denoisening autoencoder
from keras.layers import Input, Dense
from keras.models import Model

encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
# this model maps an input to its encoded representation
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
from keras.datasets import fashion_mnist
import numpy as np
(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:])))

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)

autoencoder.fit(x_train_noisy, x_train,
                epochs=10,
                batch_size=256,
                shuffle=True,
```

```python
                epochs=10,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test_noisy, x_test_noisy))
```

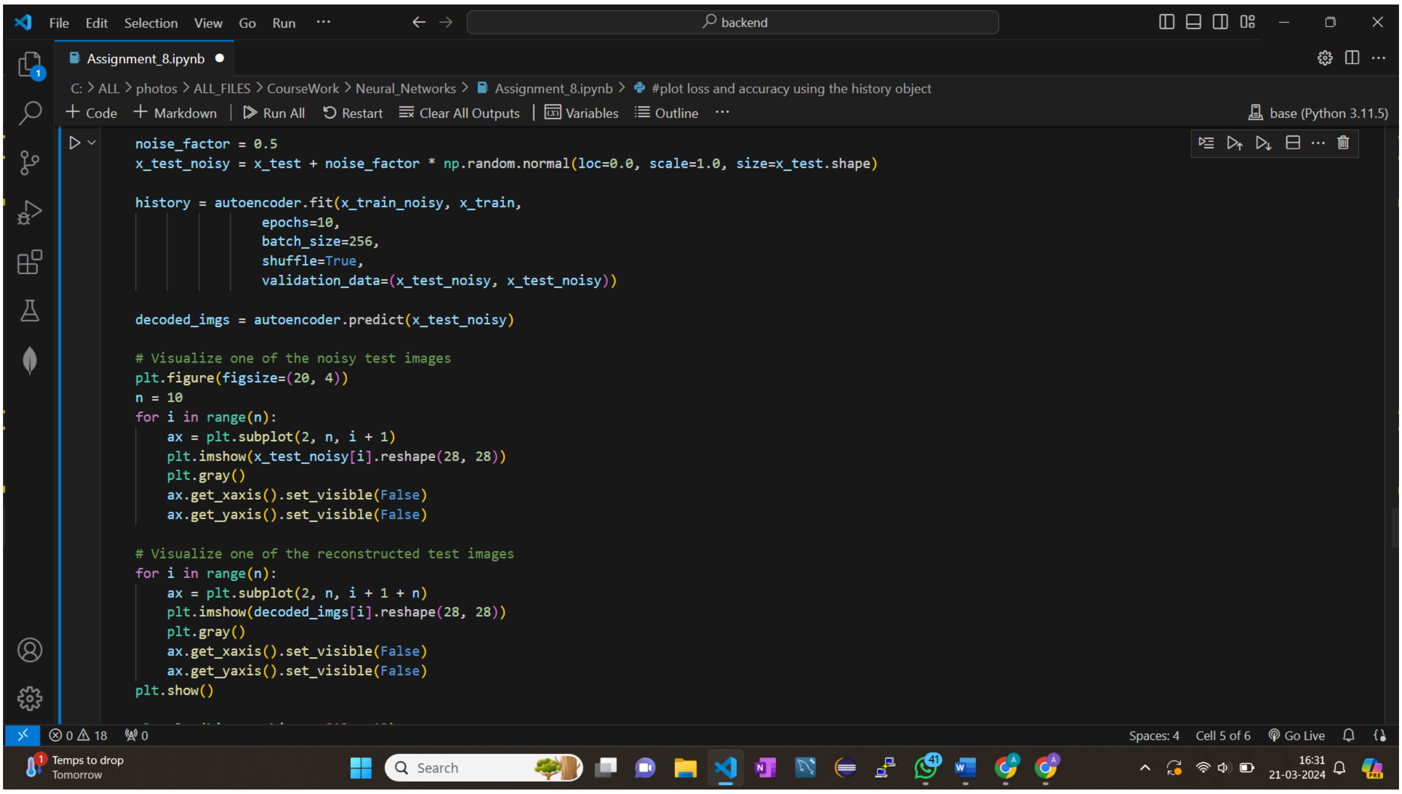
[8] 14.5s

```
Epoch 1/10
235/235 [==============================] - 2s 6ms/step - loss: 0.6977 - val_loss: 0.6977
Epoch 2/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6973 - val_loss: 0.6973
Epoch 3/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6970 - val_loss: 0.6970
Epoch 4/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6967 - val_loss: 0.6967
Epoch 5/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6965 - val_loss: 0.6965
Epoch 6/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6962 - val_loss: 0.6962
Epoch 7/10
235/235 [==============================] - 1s 5ms/step - loss: 0.6959 - val_loss: 0.6959
Epoch 8/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6956 - val_loss: 0.6957
Epoch 9/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6954 - val_loss: 0.6954
Epoch 10/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6951 - val_loss: 0.6952

<keras.src.callbacks.History at 0x20bc8b19ed0>
```

4. plot loss and accuracy using the history object



```python
#plot loss and accuracy using the history object
from keras.layers import Input, Dense
from keras.models import Model
from keras.datasets import fashion_mnist
import numpy as np
import matplotlib.pyplot as plt

encoding_dim = 32

input_img = Input(shape=(784,))

encoded = Dense(encoding_dim, activation='relu')(input_img)

decoded = Dense(784, activation='sigmoid')(encoded)

autoencoder = Model(input_img, decoded)

# Compile the model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy',metrics=['accuracy'])

# Load the fashion MNIST dataset
(x_train, _), (x_test, _) = fashion_mnist.load_data()

# Normalize the data and flatten the images
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:])))
```



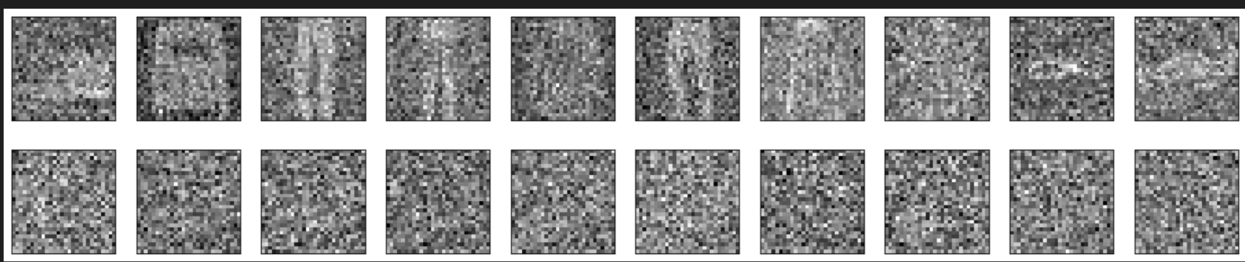
```python
noise_factor = 0.5
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_test.shape)

history = autoencoder.fit(x_train_noisy, x_train,
                epochs=10,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test_noisy, x_test_noisy))

decoded_imgs = autoencoder.predict(x_test_noisy)

# Visualize one of the noisy test images
plt.figure(figsize=(20, 4))
n = 10
for i in range(n):
    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)

# Visualize one of the reconstructed test images
for i in range(n):
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

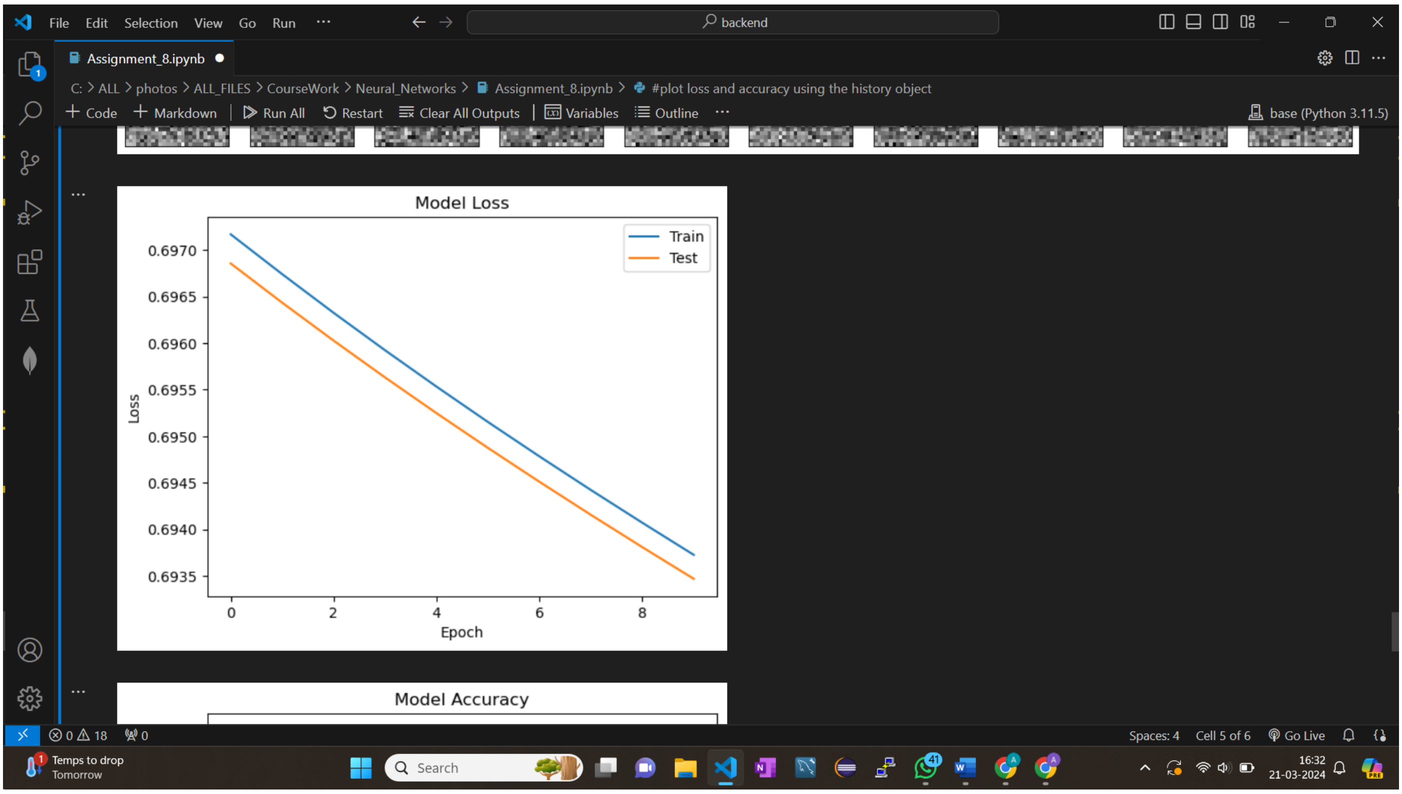
```python
    ax.get_yaxis().set_visible(False)
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper right')
plt.show()

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower right')
plt.show()
```

[9] ✓ 16.2s Python

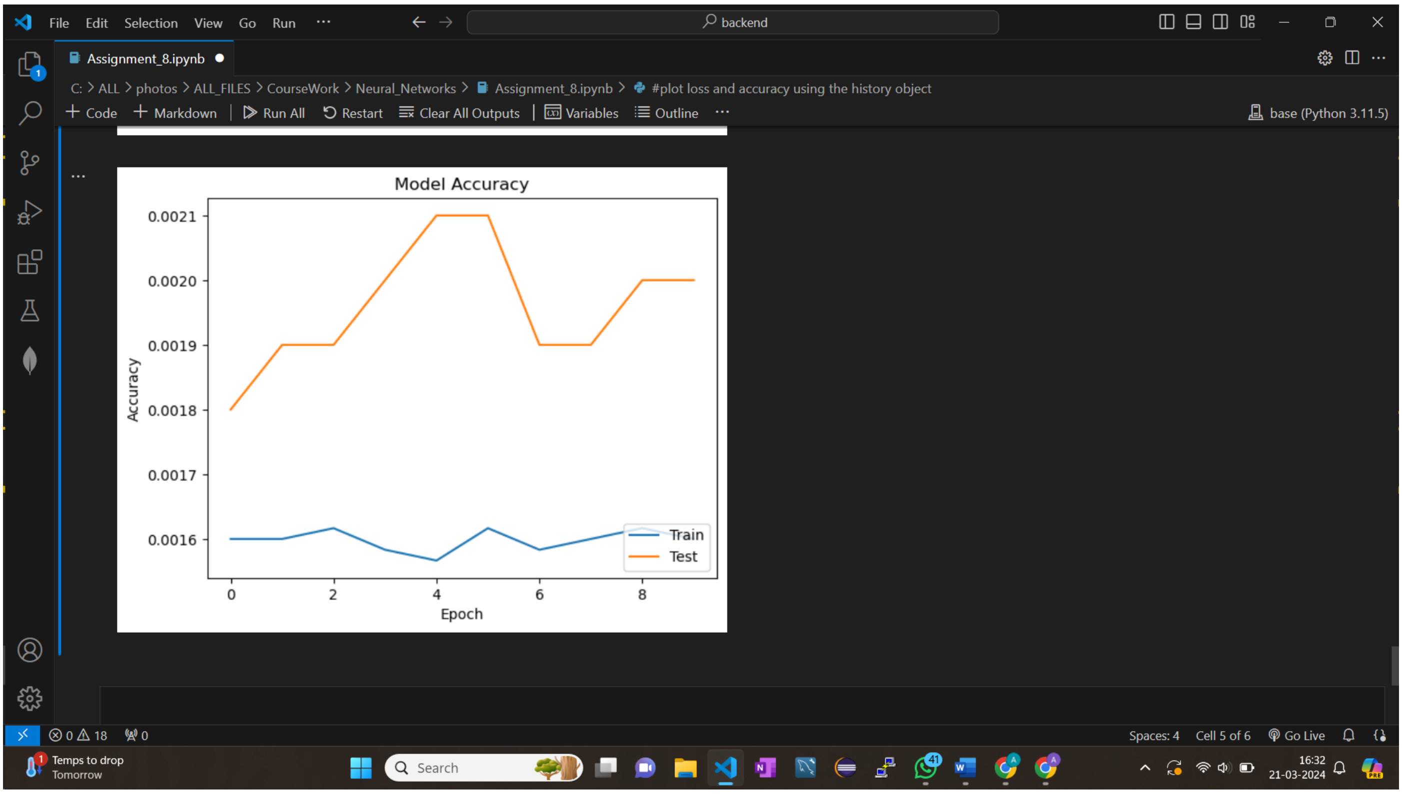
```
Epoch 1/10
235/235 [==============================] - 2s 7ms/step - loss: 0.6972 - accuracy: 0.0016 - val_loss: 0.6969 - val_accuracy: 0.0018
Epoch 2/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6967 - accuracy: 0.0016 - val_loss: 0.6964 - val_accuracy: 0.0019
Epoch 3/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6963 - accuracy: 0.0016 - val_loss: 0.6960 - val_accuracy: 0.0019
Epoch 4/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6959 - accuracy: 0.0016 - val_loss: 0.6956 - val_accuracy: 0.0020
Epoch 5/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6955 - accuracy: 0.0016 - val_loss: 0.6952 - val_accuracy: 0.0021
Epoch 6/10
```

```
235/235 [==============================] - 1s 6ms/step - loss: 0.6963 - accuracy: 0.0016 - val_loss: 0.6960 - val_accuracy: 0.0019
Epoch 4/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6959 - accuracy: 0.0016 - val_loss: 0.6956 - val_accuracy: 0.0020
Epoch 5/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6955 - accuracy: 0.0016 - val_loss: 0.6952 - val_accuracy: 0.0021
Epoch 6/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6952 - accuracy: 0.0016 - val_loss: 0.6949 - val_accuracy: 0.0021
Epoch 7/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6948 - accuracy: 0.0016 - val_loss: 0.6945 - val_accuracy: 0.0019
Epoch 8/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6944 - accuracy: 0.0016 - val_loss: 0.6942 - val_accuracy: 0.0019
Epoch 9/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6941 - accuracy: 0.0016 - val_loss: 0.6938 - val_accuracy: 0.0020
Epoch 10/10
235/235 [==============================] - 1s 6ms/step - loss: 0.6937 - accuracy: 0.0016 - val_loss: 0.6935 - val_accuracy: 0.0020
313/313 [==============================] - 0s 1ms/step
```





File Edit Selection View Go Run
backend
Assignment_8.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_8.ipynb > #plot loss and accuracy using the history object
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.5)
Model Loss
Loss
Epoch
Train
Test
Model Accuracy
Spaces: 4 Cell 5 of 6 Go Live
Temps to drop
Tomorrow
Search
16:32
21-03-2024



File Edit Selection View Go Run
backend
Assignment_8.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment_8.ipynb > #plot loss and accuracy using the history object
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.5)
Model Accuracy
Accuracy
Epoch
Train
Test
Spaces: 4 Cell 5 of 6 Go Live
Temps to drop
Tomorrow
Search
16:32
21-03-2024