ICP5

Tejendra konduru

700756561

Program1:

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

from keras.optimizers import Adadelta

# Load Fashion MNIST data

(x\_train, \_), (x\_test, \_) = fashion\_mnist.load\_data()

# Normalize and reshape the 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:])))

# Define hyperparameters

encoding\_dim = 32

input\_dim = x\_train.shape[1]

noise\_factor = 0.5

learning\_rate = 1.0

batch\_size = 128

epochs = 10

# Define the denoising autoencoder model

input\_img = Input(shape=(input\_dim,))

encoded = Dense(128, activation='relu')(input\_img)

encoded = Dense(encoding\_dim, activation='relu')(encoded)

decoded = Dense(128, activation='relu')(encoded)

decoded = Dense(input\_dim, activation='sigmoid')(decoded)

denoising\_autoencoder = Model(input\_img, decoded)

optimizer = Adadelta(learning\_rate=learning\_rate)

denoising\_autoencoder.compile(optimizer=optimizer, loss='binary\_crossentropy')

# Introducing noise to the input data

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)

x\_train\_noisy = np.clip(x\_train\_noisy, 0., 1.)

x\_test\_noisy = np.clip(x\_test\_noisy, 0., 1.)

# Train the denoising autoencoder

history = denoising\_autoencoder.fit(x\_train\_noisy, x\_train,

                                    epochs=epochs,

                                    batch\_size=batch\_size,

                                    shuffle=True,

                                    validation\_data=(x\_test\_noisy, x\_test\_noisy))

# Predict on test data

decoded\_imgs = denoising\_autoencoder.predict(x\_test\_noisy)

# Visualize one original and reconstructed image

plt.figure(figsize=(10, 4))

# Original image

plt.subplot(1, 2, 1)

plt.imshow(x\_test\_noisy[0].reshape(28, 28), cmap='gray')

plt.title('Original Noisy Image')

plt.axis('off')

# Reconstructed image

plt.subplot(1, 2, 2)

plt.imshow(decoded\_imgs[0].reshape(28, 28), cmap='gray')

plt.title('Reconstructed Image')

plt.axis('off')

plt.show()

# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model Loss')

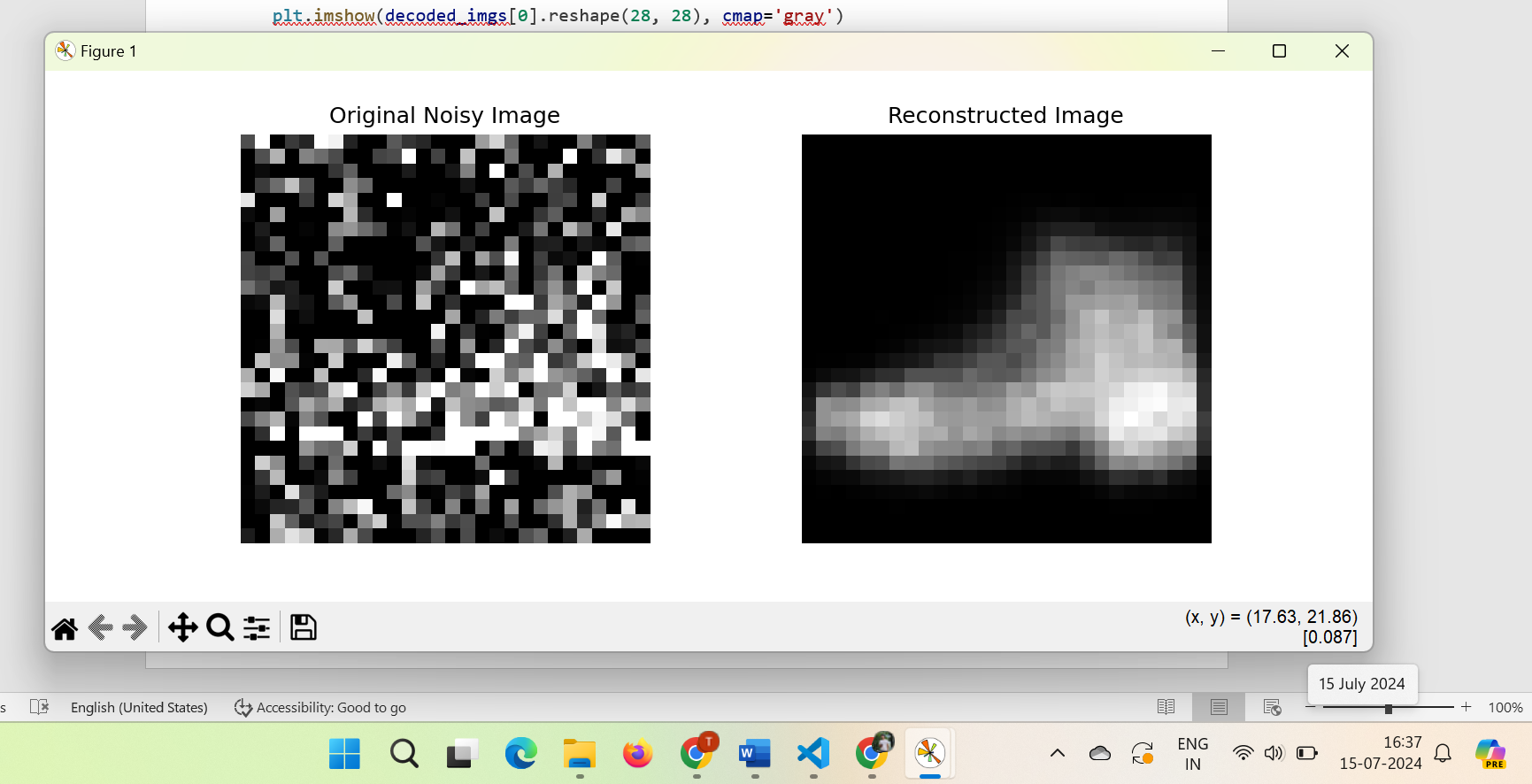
plt.ylabel('Loss')

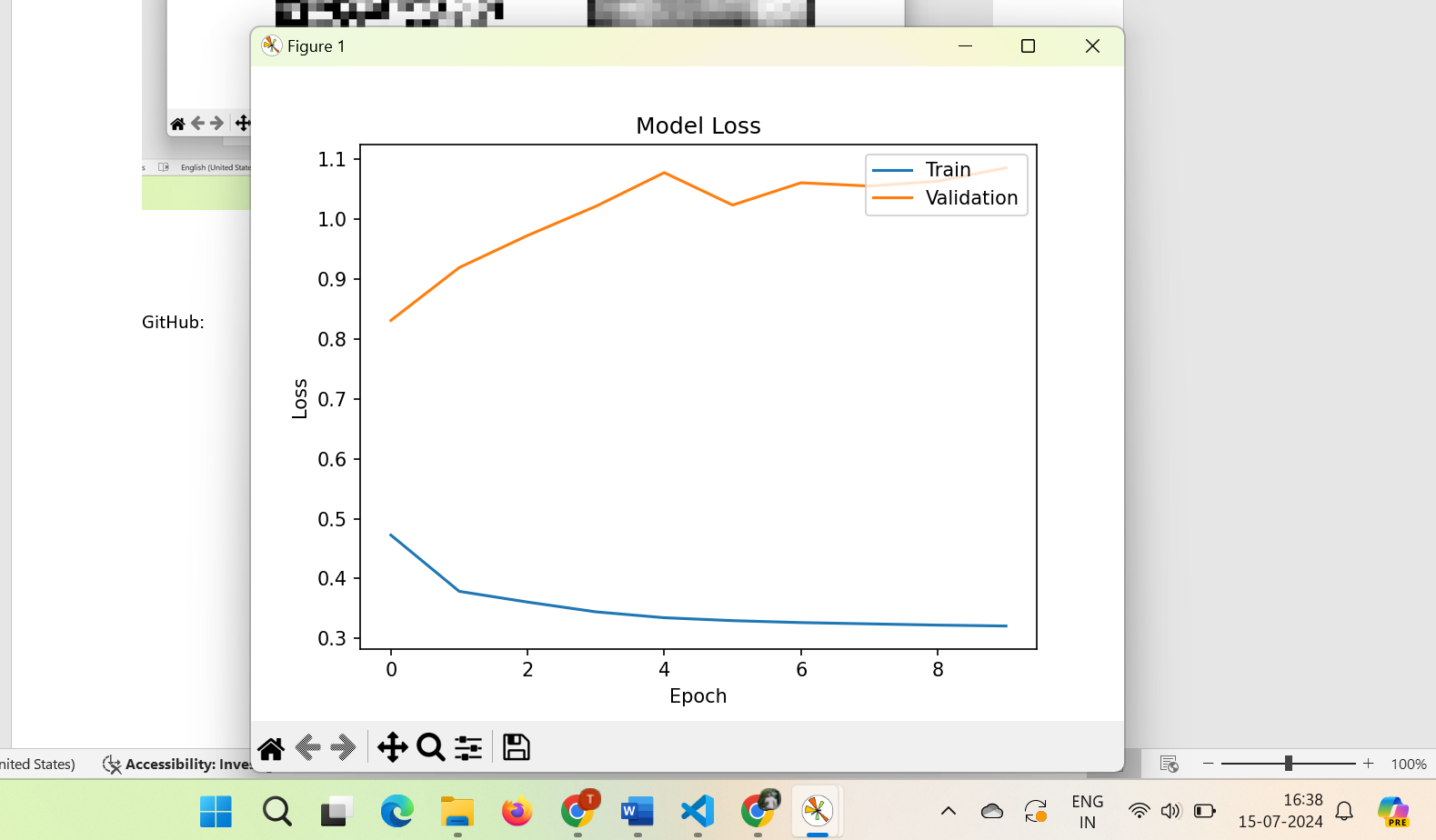
plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()

Output:





GitHub:

<https://github.com/teja375/DNN/tree/main/ICP5>