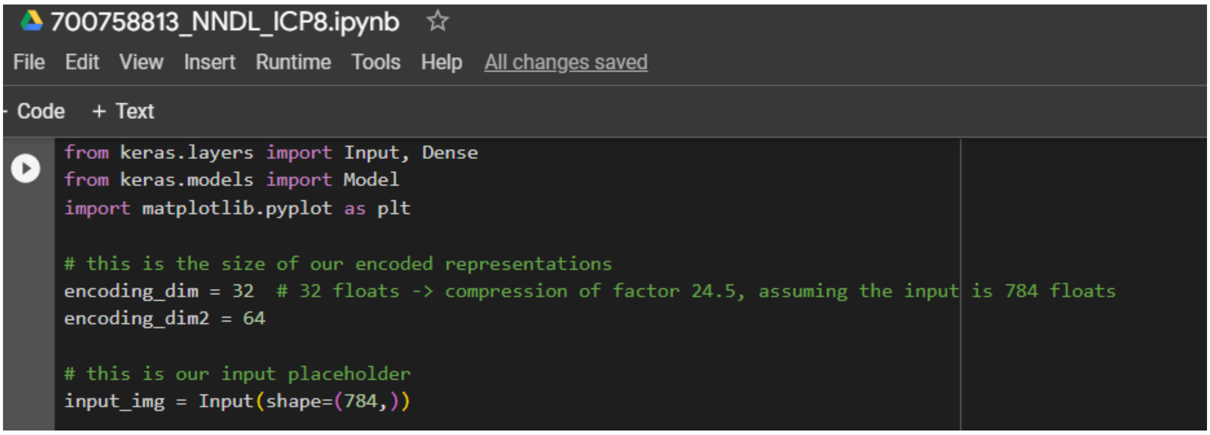
ICP8

Student Name: **Srinivas Musinuri**
Student id: 700758813

GitHub Link: https://github.com/srinivasmusinuri/700758813_NNDL_ICP8

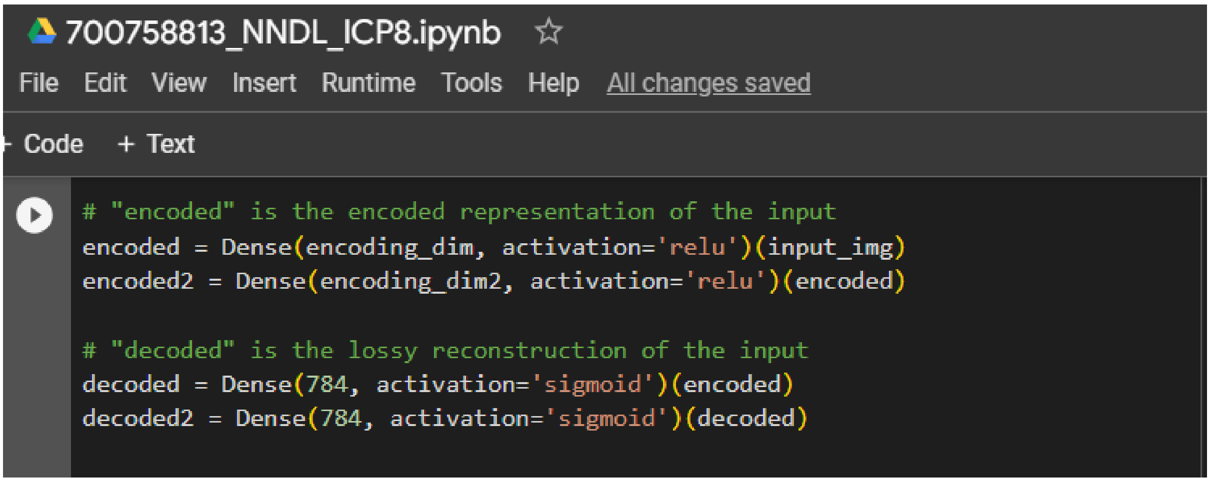
Video Link:
https://drive.google.com/file/d/1AyyltQ_XxIlI4SBCUTWRRR9m15xsnxk6/view?usp=sharing

1. 1. Add one more hidden layer to autoencoder




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

# this is the size of our encoded representations
encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
encoding_dim2 = 64

# this is our input placeholder
input_img = Input(shape=(784,))
```

Adding more hidden layer into the existing code




```python
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
encoded2 = Dense(encoding_dim2, activation='relu')(encoded)

# "decoded" is the lossy reconstruction of the input
decoded = Dense(784, activation='sigmoid')(encoded)
decoded2 = Dense(784, activation='sigmoid')(decoded)
```

working with a basic autoencoder model in Keras for image reconstruction using the Fashion MNIST dataset and code setup for unsupervised learning, where we are training an autoencoder to encode and then decode the input data

```python
# this model maps an input to its reconstruction
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 mnist, 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:])))

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

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

# Prediction on the test data
decoded_imgs = autoencoder.predict(x_test)

# Choosing an index to a test image for visualizing
idx = 10

# Reshaping the test image
test_image = x_test[idx].reshape(28, 28)

# Reshape the reconstructed image
reconstructed_image = decoded_imgs[idx].reshape(28, 28)

# Plotting the original and reconstructed images side by side
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(test_image, cmap='Blues_r')
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(reconstructed_image, cmap='Blues_r')
plt.title('Reconstructed Image')
plt.show()
```

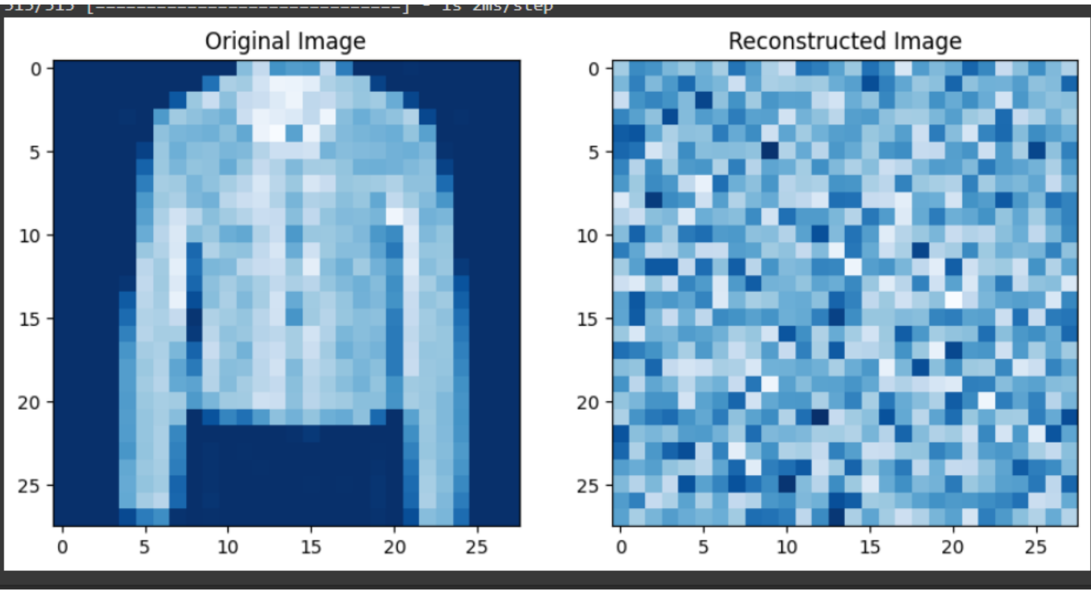
OUTPUT:

700758813_NNDL_ICP8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
Epoch 1/5
235/235 [==============================] - 5s 17ms/step - loss: 0.6930 - val_loss: 0.6930
Epoch 2/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6929 - val_loss: 0.6928
Epoch 3/5
235/235 [==============================] - 3s 14ms/step - loss: 0.6928 - val_loss: 0.6927
Epoch 4/5
235/235 [==============================] - 3s 11ms/step - loss: 0.6926 - val_loss: 0.6926
Epoch 5/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6925 - val_loss: 0.6925
313/313 [==============================] - 1s 2ms/step
```



3. Repeat the question 2 on the denoisening autoencoder

700758813_NNDL_ICP8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
[20] from keras.layers import Input, Dense
     from keras.models import Model

     # this is the size of our encoded representations
     encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

     # this is the size of our encoded representations
     encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

     # this is our input placeholder
```

700758813_NNDL_ICP8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[20] # this is our input placeholder
     input_img = Input(shape=(784,))
     # "encoded" is the encoded representation of the input
     encoded = Dense(encoding_dim, activation='relu')(input_img)
     # "decoded" is the lossy reconstruction of the input
     decoded = Dense(784, activation='sigmoid')(encoded)
     # this model maps an input to its reconstruction
     autoencoder = Model(input_img, decoded)
     # this model maps an input to its encoded representation
     autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])
     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:])))
```

```
#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)

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

OUTPUT:

700758813_NNDL_ICP8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

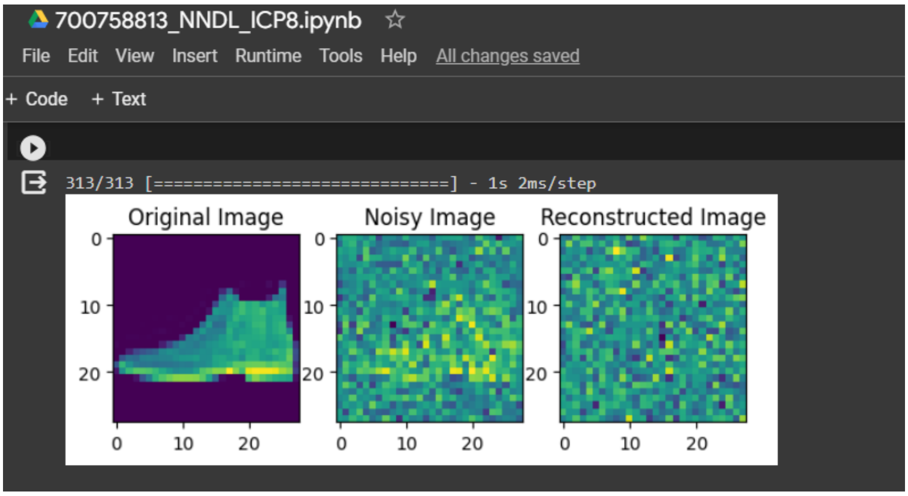
+ Code + Text

```
Epoch 1/10
235/235 [==============================] - 4s 15ms/step - loss: 0.6963 - accuracy: 9.8333e-04 - val_loss: 0.6960 - val_accuracy: 0.0019
Epoch 2/10
235/235 [==============================] - 3s 13ms/step - loss: 0.6959 - accuracy: 9.6667e-04 - val_loss: 0.6957 - val_accuracy: 0.0022
Epoch 3/10
235/235 [==============================] - 3s 13ms/step - loss: 0.6956 - accuracy: 0.0010 - val_loss: 0.6954 - val_accuracy: 0.0022
Epoch 4/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6953 - accuracy: 9.8333e-04 - val_loss: 0.6951 - val_accuracy: 0.0023
Epoch 5/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6950 - accuracy: 9.8333e-04 - val_loss: 0.6948 - val_accuracy: 0.0023
Epoch 6/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6947 - accuracy: 0.0010 - val_loss: 0.6946 - val_accuracy: 0.0023
Epoch 7/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6945 - accuracy: 0.0010 - val_loss: 0.6943 - val_accuracy: 0.0022
Epoch 8/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6942 - accuracy: 0.0010 - val_loss: 0.6940 - val_accuracy: 0.0022
Epoch 9/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6939 - accuracy: 0.0010 - val_loss: 0.6937 - val_accuracy: 0.0022
Epoch 10/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6937 - accuracy: 0.0011 - val_loss: 0.6935 - val_accuracy: 0.0022
```

Visualizing the test data using Matplotlib

```python
import matplotlib.pyplot as plt

# Get the reconstructed images
reconstructed_images = autoencoder.predict(x_test_noisy)

# Select one image to display
img_to_display = 0

# Display the original, noisy, and reconstructed images side by side
plt.subplot(1, 3, 1)
plt.imshow(x_test[img_to_display].reshape(28, 28))
plt.title('Original Image')

plt.subplot(1, 3, 2)
plt.imshow(x_test_noisy[img_to_display].reshape(28, 28))
plt.title('Noisy Image')

plt.subplot(1, 3, 3)
plt.imshow(reconstructed_images[img_to_display].reshape(28, 28))
plt.title('Reconstructed Image')

plt.show()
```

OUTPUT:



3. plot loss and accuracy using the history object

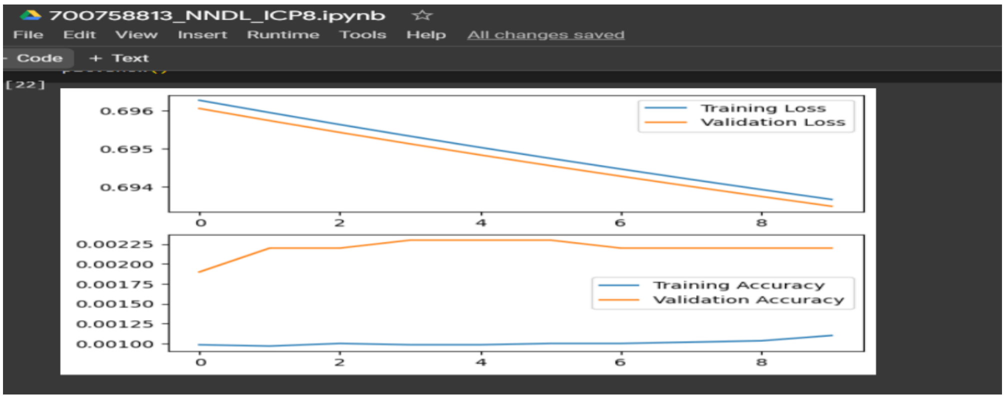
700758813_NNDL_ICP8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
# Plot the loss and accuracy over epochs
plt.subplot(2, 1, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()

plt.show()
```

OUTPUT:


700758813_NNDL_ICP8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Code + Text
[22]
0.696
0.695
0.694
Training Loss
Validation Loss
0 2 4 6 8
0.00225
0.00200
0.00175
0.00150
0.00125
0.00100
Training Accuracy
Validation Accuracy
0 2 4 6 8