Spring 2024: CS5720 Neural Networks & Deep Learning –

Autoencoders

Assignment- Autoencoders

Name: Pushkara Naga Sai Sri Vyshnavi Chakka

STUDENT ID:700752861

GitHub link: https://github.com/PushkaraChakka/Assignment_8_autoencoders

Video link: https://drive.google.com/file/d/1uoiyuljsN2RWFn4wFN4B-C6S_Vzc7kXV/view?usp=sharing

Lesson Overview:

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

Programming elements:

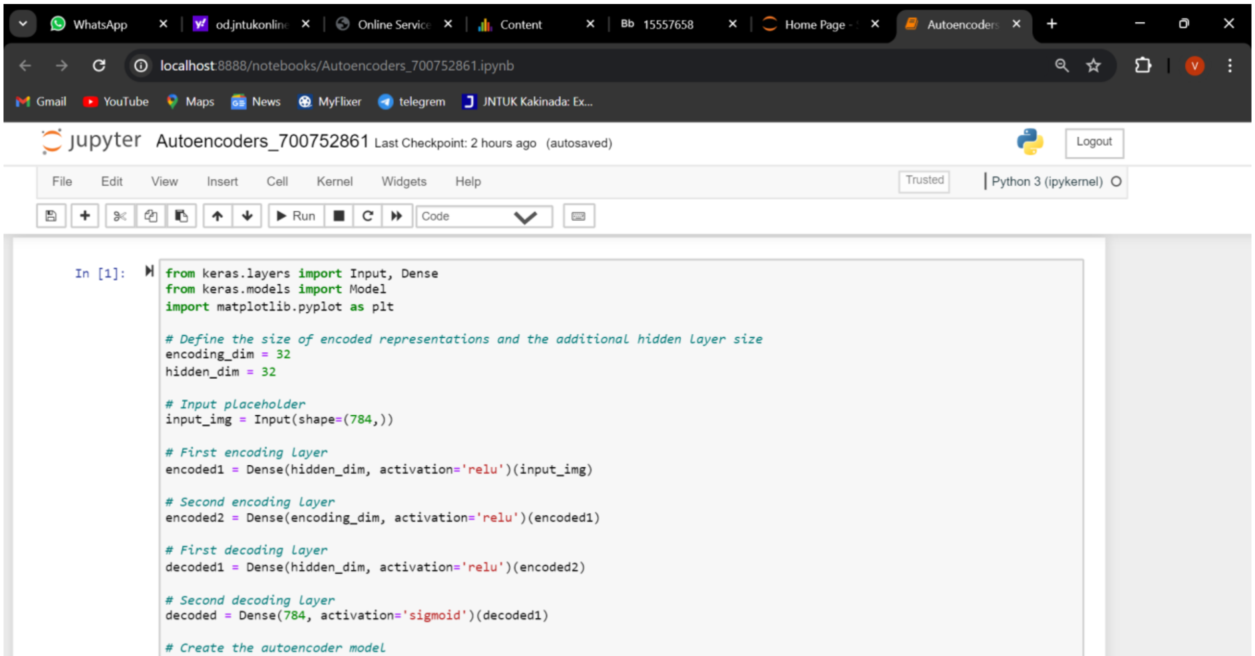
1. Basics of Autoencoders

2. Role of Autoencoders in unsupervised learning

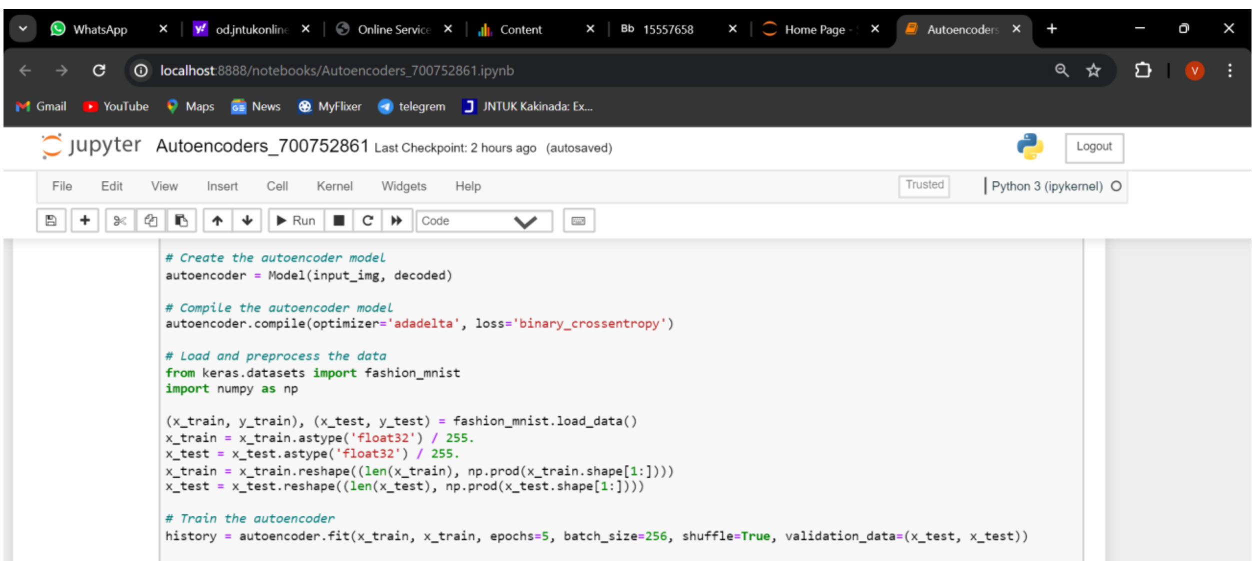
3. Types of Autoencoders

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

5. Use case: Stacked autoencoder



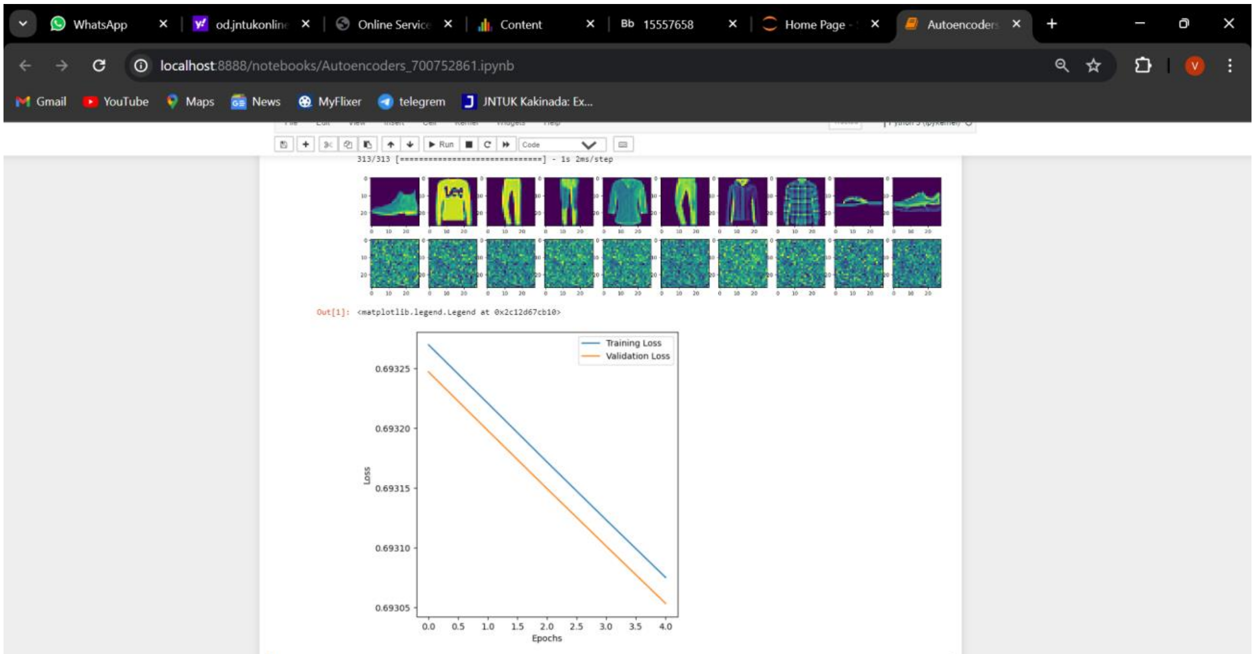

```python
from keras.layers import Input, Dense
from keras.models import Model
import matplotlib.pyplot as plt

# Define the size of encoded representations and the additional hidden layer size
encoding_dim = 32
hidden_dim = 32

# Input placeholder
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 autoencoder model
```




```python
# Create the autoencoder model
autoencoder = Model(input_img, decoded)

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

# Load and preprocess 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:])))

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

```python
# 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()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 3us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 1s 0us/step
Epoch 1/5
WARNING:tensorflow:From C:\Users\chakk\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.Ragg
edTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

235/235 [==============================] - 6s 17ms/step - loss: 0.6933 - val_loss: 0.6932
Epoch 2/5
235/235 [==============================] - 4s 17ms/step - loss: 0.6932 - val_loss: 0.6932
Epoch 3/5
235/235 [==============================] - 3s 11ms/step - loss: 0.6932 - val_loss: 0.6931
Epoch 4/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6931 - val_loss: 0.6931
Epoch 5/5
235/235 [==============================] - 4s 16ms/step - loss: 0.6931 - val_loss: 0.6931
313/313 [==============================] - 1s 2ms/step
```


WhatsApp
od.jntukonline
Online Service
Content
Bb 15557658
Home Page
Autoencoders
localhost:8888/notebooks/Autoencoders_700752861.ipynb
Gmail
YouTube
Maps
News
MyFlixer
telegrem
JNTUK Kakinada: Ex...
Run
Code
313/313 [==============================] - 1s 2ms/step
Out[1]: <matplotlib.legend.Legend at 0x2c12d67cb10>
Training Loss
Validation Loss
0.69325
0.69320
0.69315
0.69310
0.69305
Loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

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

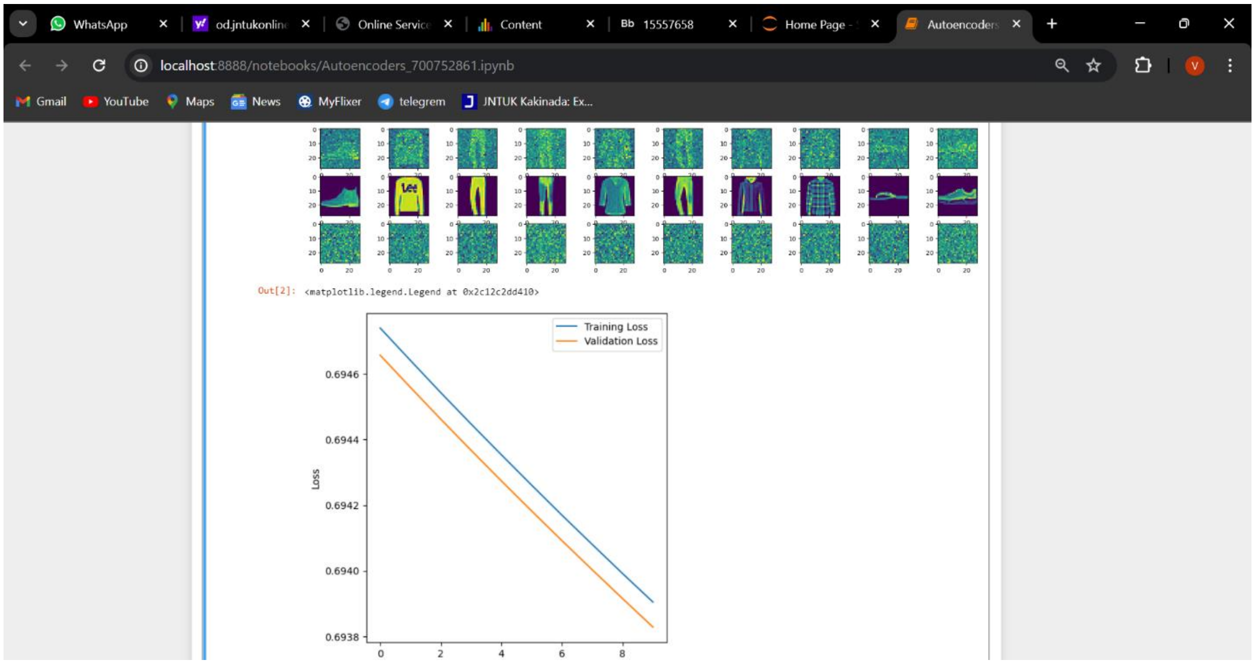
In [2]:

```
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:])))
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)
```

```python
# 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_

# 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()
```

```
Epoch 1/10
235/235 [==============================] - 8s 28ms/step - loss: 0.6947 - val_loss: 0.6947
Epoch 2/10
235/235 [==============================] - 4s 18ms/step - loss: 0.6946 - val_loss: 0.6946
Epoch 3/10
235/235 [==============================] - 6s 27ms/step - loss: 0.6945 - val_loss: 0.6945
Epoch 4/10
235/235 [==============================] - 6s 27ms/step - loss: 0.6944 - val_loss: 0.6944
Epoch 5/10
235/235 [==============================] - 7s 28ms/step - loss: 0.6944 - val_loss: 0.6943
Epoch 6/10
235/235 [==============================] - 6s 25ms/step - loss: 0.6943 - val_loss: 0.6942
Epoch 7/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6942 - val_loss: 0.6941
Epoch 8/10
235/235 [==============================] - 6s 27ms/step - loss: 0.6941 - val_loss: 0.6940
Epoch 9/10
235/235 [==============================] - 7s 28ms/step - loss: 0.6940 - val_loss: 0.6939
Epoch 10/10
235/235 [==============================] - 6s 26ms/step - loss: 0.6939 - val_loss: 0.6938
313/313 [==============================] - 1s 3ms/step
```


WhatsApp
Content
15557658
localhost:8888/notebooks/Autoencoders_700752861.ipynb
Gmail
YouTube
Maps
News
MyFlixer
telegrem
JNTUK Kakinada: Ex...
Out[2]: <matplotlib.legend.Legend at 0x2c12c2dd410>
Training Loss
Validation Loss
Loss
0.6946
0.6944
0.6942
0.6940
0.6938
0
2
4
6
8