NEURAL NETWORKS & DEEP LEARNING

ASSIGNMENT – 5

NAME : SRAVANI MANNURU

STUDENT ID : 700766162

GITHUB LINK : https://github.com/sravs2031/Neural-Networks-ICP5.git

VIDEO LINK : https://drive.google.com/file/d/1iNo-Gu6vSCgZDB8xNvPIrV6gVni_kVD4/view?usp=drive_link

SCREENSHOTS :

```
(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:])))
# Training the autoencoder
autoencoder.fit(x_train, x_train,
                epochs=8,  # Increased epochs
                batch_size=128,  # Smaller batch size
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
Epoch 1/8
469/469 [==============================] - 5s 10ms/step - loss: 0.3435 - val_loss: 0.3018
Epoch 2/8
469/469 [==============================] - 6s 12ms/step - loss: 0.2936 - val_loss: 0.2917
Epoch 3/8
469/469 [==============================] - 5s 10ms/step - loss: 0.2856 - val_loss: 0.2852
```

```
Epoch 4/8
469/469 [==============================] - 7s 15ms/step - loss: 0.2814 - val_loss: 0.2819
Epoch 5/8
469/469 [==============================] - 8s 17ms/step - loss: 0.2786 - val_loss: 0.2805
Epoch 6/8
469/469 [==============================] - 8s 17ms/step - loss: 0.2767 - val_loss: 0.2779
Epoch 7/8
469/469 [==============================] - 4s 9ms/step - loss: 0.2751 - val_loss: 0.2769
Epoch 8/8
469/469 [==============================] - 6s 12ms/step - loss: 0.2740 - val_loss: 0.2756
<keras.src.callbacks.History at 0x77fc113c4c10>
```

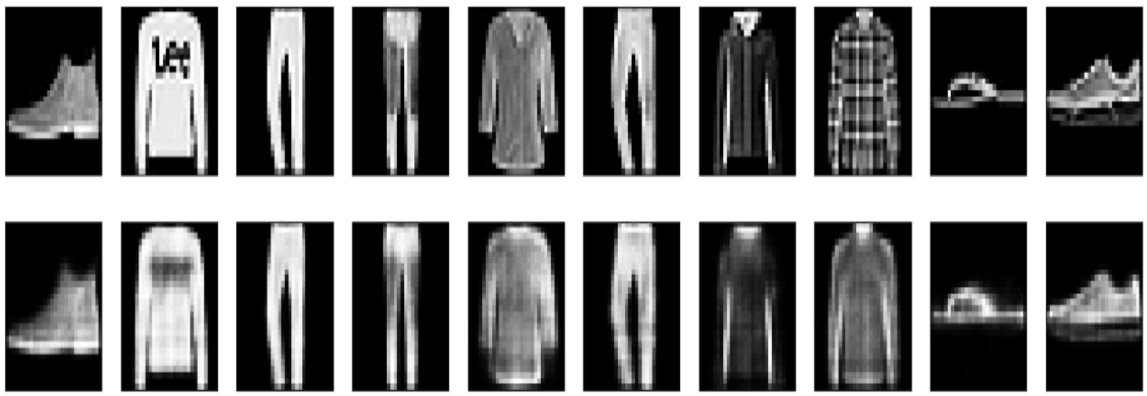
Neural ICP 5.ipynb

```
import matplotlib.pyplot as plt

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

# Function to plot images
def plot_images(original_images, decoded_images, num_images=10):
    plt.figure(figsize=(20, 4))
    for i in range(num_images):
        # Original images
        ax = plt.subplot(2, num_images, i + 1)
        plt.imshow(original_images[i].reshape(28, 28), cmap='gray')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

        # Reconstructed images
        ax = plt.subplot(2, num_images, i + 1 + num_images)
        plt.imshow(decoded_images[i].reshape(28, 28), cmap='gray')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
    plt.show()

# Visualize original and reconstructed images
plot_images(x_test, decoded_imgs)
```

OUTPUT:

+ Code + Text

313/313 [==============================] - 1s 2ms/step



Neural ICP 5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

[3]

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

# Define the size of our encoded representation
encoding_dim = 32  # 32 floats -> compression factor of 24.5 assuming the input is 784 floats

# Input placeholder with added noise
input_img = Input(shape=(784,))
noisy_img = GaussianNoise(0.5)(input_img)  # Adding Gaussian noise with stddev=0.5
# Encoded representation
encoded = Dense(128, activation='relu')(noisy_img)  # Increased size of hidden layer
# Additional hidden layer
hidden_layer = Dense(64, activation='relu')(encoded)  # Adding a hidden layer with 64 neurons
# Decoded representation
decoded = Dense(784, activation='sigmoid')(hidden_layer)
# Denoising Autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the autoencoder (defaults to adam optimizer with default learning rate)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Loading and preprocessing the Fashion MNIST data
from keras.datasets import fashion_mnist
(x_train, _), (x_test, _) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.
```

```python
(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:])))

# Add Gaussian noise to the training and test data
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)

# Clip the values to be between 0 and 1
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)

# Training the denoising autoencoder
autoencoder.fit(x_train_noisy, x_train,
                epochs=25,  # Increased epochs for better training
                batch_size=100,
                shuffle=True,
                validation_data=(x_test_noisy, x_test))

# Predict on test data
decoded_imgs = autoencoder.predict(x_test_noisy)
# Function to plot images
def plot_images(original_images, noisy_images, decoded_images, num_images=10):
    plt.figure(figsize=(20, 6))
    for i in range(num_images):
```

```python
        ax = plt.subplot(3, num_images, i + 1)
        plt.imshow(original_images[i].reshape(28, 28), cmap='gray')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
        if i == 0:
            ax.set_title('Original Images')

        # Noisy images
        ax = plt.subplot(3, num_images, i + 1 + num_images)
        plt.imshow(noisy_images[i].reshape(28, 28), cmap='gray')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
        if i == 0:
            ax.set_title('Noisy Input')

        # Reconstructed images
        ax = plt.subplot(3, num_images, i + 1 + 2 * num_images)
        plt.imshow(decoded_images[i].reshape(28, 28), cmap='gray')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
        if i == 0:
            ax.set_title('Denoised Output')
    plt.tight_layout()
    plt.show()

# Visualize original, noisy, and denoised images
plot_images(x_test, x_test_noisy, decoded_imgs)
```

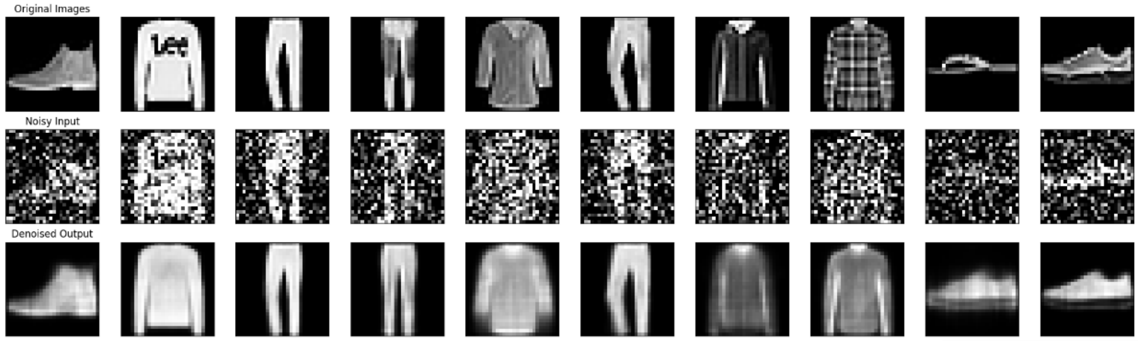
OUTPUT:

```
Epoch 1/25
600/600 [==============================] - 6s 9ms/step - loss: 0.3712 - val_loss: 0.3542
Epoch 2/25
600/600 [==============================] - 7s 12ms/step - loss: 0.3303 - val_loss: 0.3322
Epoch 3/25
600/600 [==============================] - 6s 9ms/step - loss: 0.3241 - val_loss: 0.3349
Epoch 4/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3202 - val_loss: 0.3221
Epoch 5/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3177 - val_loss: 0.3177
Epoch 6/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3160 - val_loss: 0.3153
Epoch 7/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3149 - val_loss: 0.3075
Epoch 8/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3137 - val_loss: 0.3077
Epoch 9/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3129 - val_loss: 0.3060
Epoch 10/25
600/600 [==============================] - 5s 9ms/step - loss: 0.3122 - val_loss: 0.3038
Epoch 11/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3116 - val_loss: 0.3026
Epoch 12/25
600/600 [==============================] - 6s 11ms/step - loss: 0.3111 - val_loss: 0.3015
Epoch 13/25
600/600 [==============================] - 7s 12ms/step - loss: 0.3107 - val_loss: 0.3007
Epoch 14/25
```

\+ Code \+ Text

```
Epoch 14/25
600/600 [==============================] - 7s 12ms/step - loss: 0.3103 - val_loss: 0.3007
Epoch 15/25
600/600 [==============================] - 7s 12ms/step - loss: 0.3101 - val_loss: 0.3004
Epoch 16/25
600/600 [==============================] - 5s 9ms/step - loss: 0.3097 - val_loss: 0.2999
Epoch 17/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3095 - val_loss: 0.2994
Epoch 18/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3093 - val_loss: 0.2989
Epoch 19/25
600/600 [==============================] - 7s 11ms/step - loss: 0.3090 - val_loss: 0.2990
Epoch 20/25
600/600 [==============================] - 5s 9ms/step - loss: 0.3088 - val_loss: 0.2988
Epoch 21/25
600/600 [==============================] - 6s 11ms/step - loss: 0.3086 - val_loss: 0.2984
Epoch 22/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3085 - val_loss: 0.2981
Epoch 23/25
600/600 [==============================] - 6s 9ms/step - loss: 0.3084 - val_loss: 0.2983
Epoch 24/25
600/600 [==============================] - 6s 11ms/step - loss: 0.3083 - val_loss: 0.2983
Epoch 25/25
600/600 [==============================] - 6s 10ms/step - loss: 0.3080 - val_loss: 0.2982
313/313 [==============================] - 1s 2ms/step
```

```
[3] 313/313 [==============================] - 1s 2ms/step
```



Neural ICP 5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

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

# Load and preprocess Fashion MNIST data
(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:])))

# Add Gaussian noise to the data
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)

# Clip values to be between 0 and 1
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)

# Define the denoising autoencoder model
input_img = Input(shape=(784,))
noisy_img = GaussianNoise(0.5)(input_img)
encoded = Dense(128, activation='relu')(noisy_img)
hidden_layer = Dense(64, activation='relu')(encoded)
```

```python
decoded = Dense(784, activation='sigmoid')(hidden_layer)
autoencoder = Model(input_img, decoded)
# Compile the model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Train the denoising autoencoder
history = autoencoder.fit(x_train_noisy, x_train,
                          epochs=20,
                          batch_size=128,
                          shuffle=True,
                          validation_data=(x_test_noisy, x_test))

# Plot training history (loss)
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

OUTPUT:

Neural ICP 5.ipynb

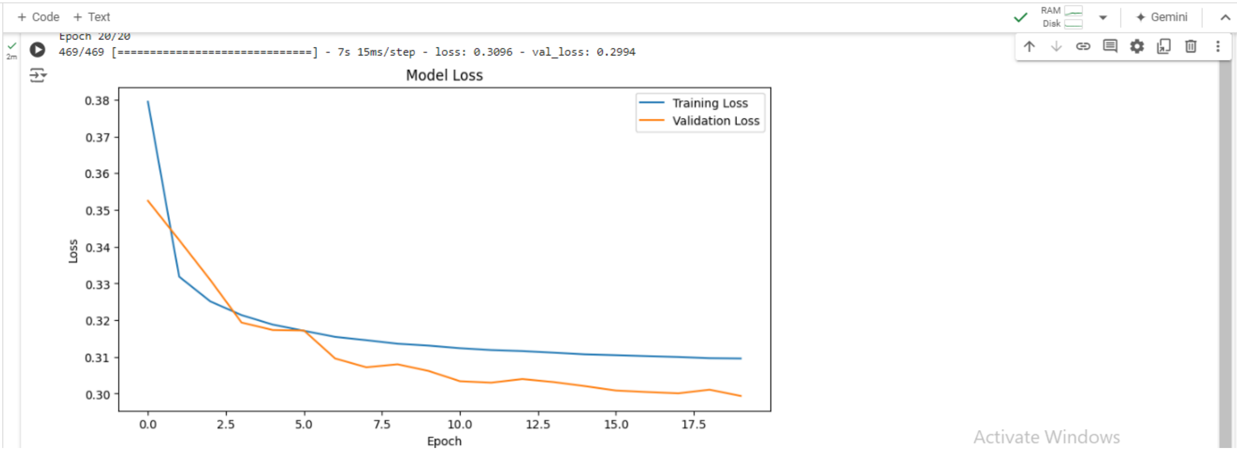
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
Epoch 1/20
469/469 [==============================] - 7s 14ms/step - loss: 0.3795 - val_loss: 0.3525
Epoch 2/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3319 - val_loss: 0.3418
Epoch 3/20
469/469 [==============================] - 6s 12ms/step - loss: 0.3251 - val_loss: 0.3309
Epoch 4/20
469/469 [==============================] - 6s 12ms/step - loss: 0.3214 - val_loss: 0.3194
Epoch 5/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3188 - val_loss: 0.3173
Epoch 6/20
469/469 [==============================] - 6s 14ms/step - loss: 0.3171 - val_loss: 0.3172
Epoch 7/20
469/469 [==============================] - 5s 12ms/step - loss: 0.3154 - val_loss: 0.3096
Epoch 8/20
469/469 [==============================] - 6s 14ms/step - loss: 0.3146 - val_loss: 0.3072
Epoch 9/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3136 - val_loss: 0.3080
Epoch 10/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3131 - val_loss: 0.3062
Epoch 11/20
469/469 [==============================] - 6s 13ms/step - loss: 0.3124 - val_loss: 0.3034
Epoch 12/20
469/469 [==============================] - 6s 12ms/step - loss: 0.3119 - val_loss: 0.3030
Epoch 13/20
469/469 [==============================] - 7s 14ms/step - loss: 0.3116 - val_loss: 0.3040
Epoch 14/20
469/469 [==============================] - 5s 12ms/step - loss: 0.3111 - val_loss: 0.3032
```


```
469/469 [==============================] - 7s 14ms/step - loss: 0.3116 - val_loss: 0.3040
Epoch 14/20
469/469 [==============================] - 5s 12ms/step - loss: 0.3111 - val_loss: 0.3032
```


```
Epoch 15/20
469/469 [==============================] - 6s 14ms/step - loss: 0.3107 - val_loss: 0.3020
Epoch 16/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3105 - val_loss: 0.3008
Epoch 17/20
469/469 [==============================] - 6s 12ms/step - loss: 0.3102 - val_loss: 0.3004
Epoch 18/20
469/469 [==============================] - 6s 13ms/step - loss: 0.3100 - val_loss: 0.3001
Epoch 19/20
469/469 [==============================] - 5s 11ms/step - loss: 0.3097 - val_loss: 0.3011
Epoch 20/20
469/469 [==============================] - 7s 15ms/step - loss: 0.3096 - val_loss: 0.2994
```


+ Code + Text
RAM
Disk
Gemini
Epoch 20/20
469/469 [==============================] - 7s 15ms/step - loss: 0.3096 - val_loss: 0.2994
Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.38
0.37
0.36
0.35
0.34
0.33
0.32
0.31
0.30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Activate Windows