Spring 2024: CS5720
Neural Networks and Deep Learning - ICP-8
BHAVANA BILLA (700756590)

Github Link: https://github.com/BillaBhavana7/neuralN

```
from keras.layers import Input, Dense
from keras.models import Model
from keras.datasets import fashion_mnist
import numpy as np

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

input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])

(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))
```

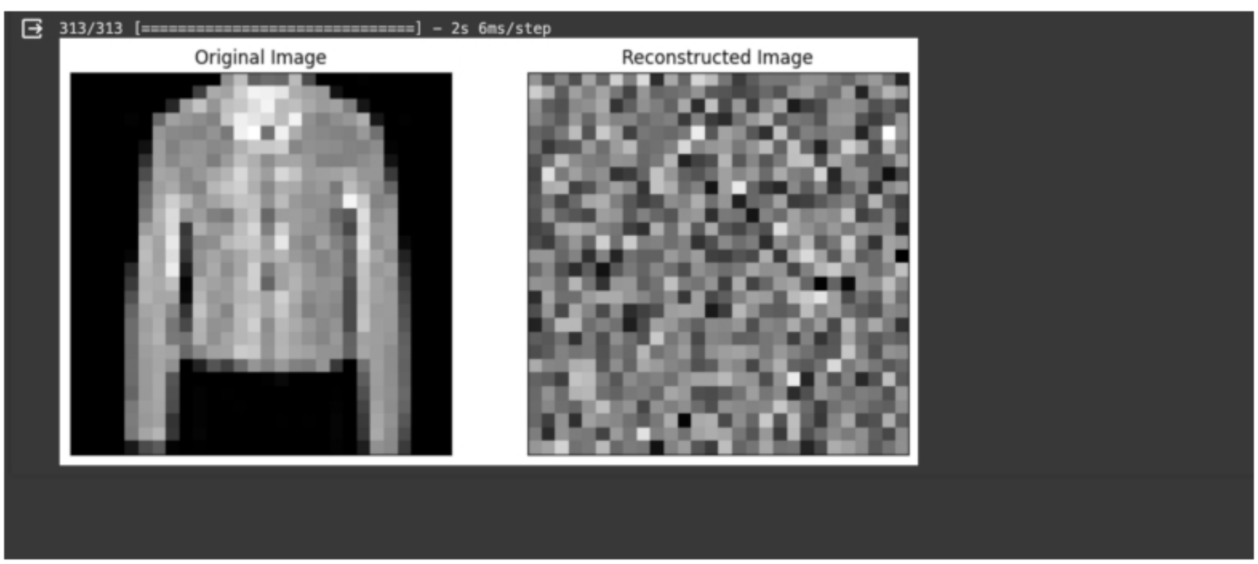
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 1s 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 1/5
235/235 [==============================] - 10s 33ms/step - loss: 0.6931 - accuracy: 0.0011 - val_loss: 0.6931 - val_accuracy: 0.0015
Epoch 2/5
235/235 [==============================] - 4s 17ms/step - loss: 0.6930 - accuracy: 0.0011 - val_loss: 0.6930 - val_accuracy: 0.0015
Epoch 3/5
235/235 [==============================] - 4s 16ms/step - loss: 0.6929 - accuracy: 0.0011 - val_loss: 0.6929 - val_accuracy: 0.0015
Epoch 4/5
235/235 [==============================] - 4s 18ms/step - loss: 0.6928 - accuracy: 0.0012 - val_loss: 0.6928 - val_accuracy: 0.0014
Epoch 5/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6927 - accuracy: 0.0011 - val_loss: 0.6927 - val_accuracy: 0.0014
<keras.src.callbacks.History at 0x7fae5d9e3f10>
```

```
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,))

encoded1 = Dense(128, activation='relu')(input_img)
encoded2 = Dense(encoding_dim, activation='relu')(encoded1)

decoded1 = Dense(128, activation='relu')(encoded2)
decoded2 = Dense(784, activation='sigmoid')(decoded1)

autoencoder = Model(input_img, decoded2)

encoder = Model(input_img, encoded2)

encoded_input = Input(shape=(encoding_dim,))
decoder_layer1 = autoencoder.layers[-2]
decoder_layer2 = autoencoder.layers[-1]

decoder = Model(encoded_input, decoder_layer2(decoder_layer1(encoded_input)))

autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy',metrics ='accuracy')

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))
```

```
Epoch 1/5
235/235 [==============================] - 5s 18ms/step - loss: 0.6944 - accuracy: 0.0018 - val_loss: 0.6943 - val_accuracy: 0.0021
Epoch 2/5
235/235 [==============================] - 4s 16ms/step - loss: 0.6942 - accuracy: 0.0018 - val_loss: 0.6942 - val_accuracy: 0.0021
Epoch 3/5
235/235 [==============================] - 4s 19ms/step - loss: 0.6941 - accuracy: 0.0018 - val_loss: 0.6940 - val_accuracy: 0.0020
Epoch 4/5
235/235 [==============================] - 4s 18ms/step - loss: 0.6939 - accuracy: 0.0019 - val_loss: 0.6939 - val_accuracy: 0.0020
Epoch 5/5
235/235 [==============================] - 4s 16ms/step - loss: 0.6938 - accuracy: 0.0019 - val_loss: 0.6937 - val_accuracy: 0.0020
<keras.src.callbacks.History at 0x7faee5b87f10>
```

```
import matplotlib.pyplot as plt
plt.showhe reconstructed images for the test set
reconstructed_imgs = autoencoder.predict(x_test)

# Choose a random image from the test set
n = 10 # index of the image to be plotted
plt.figure(figsize=(10, 5))

# Plot the original image
ax = plt.subplot(1, 2, 1)
plt.imshow(x_test[n].reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
ax.set_title("Original Image")

# Plot the reconstructed image
ax = plt.subplot(1, 2, 2)
plt.imshow(reconstructed_imgs[n].reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
ax.set_title("Reconstructed Image")

plt.show()
```

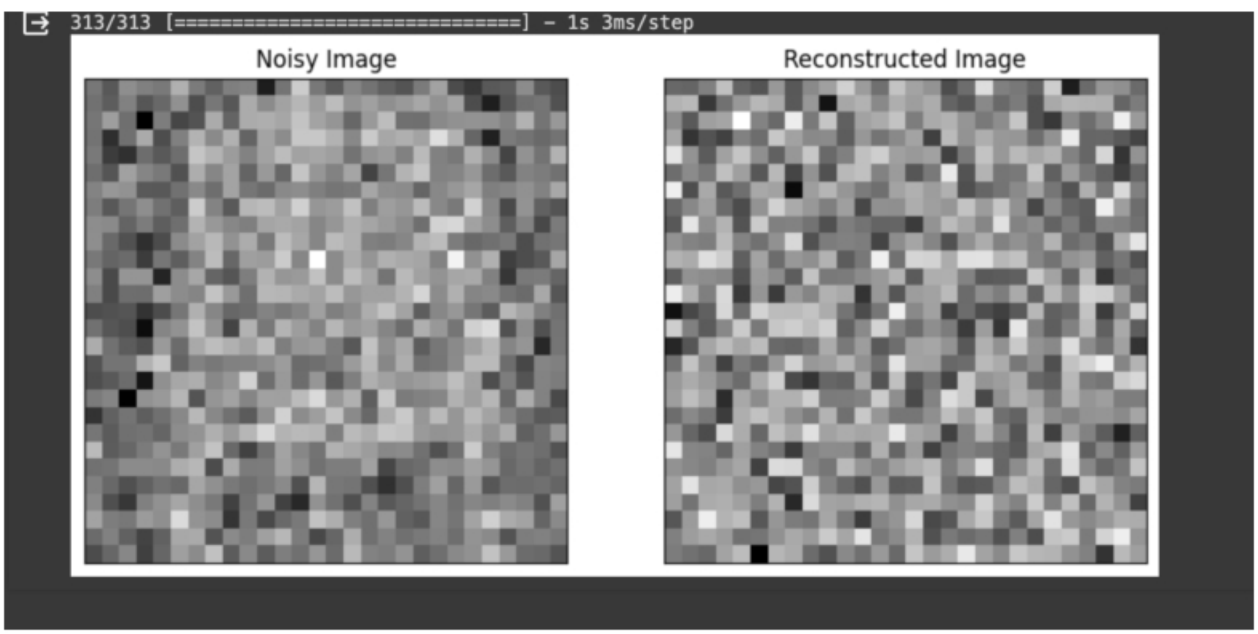
Output:



```
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 = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
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:])))

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,
            validation_data=(x_test_noisy, x_test_noisy))
```

```
Epoch 1/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6965 - accuracy: 0.0012 - val_loss: 0.6962 - val_accuracy: 0.0013
Epoch 2/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6962 - accuracy: 0.0012 - val_loss: 0.6960 - val_accuracy: 0.0014
Epoch 3/10
235/235 [==============================] - 3s 13ms/step - loss: 0.6959 - accuracy: 0.0013 - val_loss: 0.6957 - val_accuracy: 0.0013
Epoch 4/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6957 - accuracy: 0.0013 - val_loss: 0.6955 - val_accuracy: 0.0013
Epoch 5/10
235/235 [==============================] - 2s 11ms/step - loss: 0.6954 - accuracy: 0.0013 - val_loss: 0.6952 - val_accuracy: 0.0013
Epoch 6/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6952 - accuracy: 0.0013 - val_loss: 0.6950 - val_accuracy: 0.0013
Epoch 7/10
235/235 [==============================] - 3s 13ms/step - loss: 0.6949 - accuracy: 0.0014 - val_loss: 0.6947 - val_accuracy: 0.0013
Epoch 8/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6947 - accuracy: 0.0014 - val_loss: 0.6945 - val_accuracy: 0.0012
Epoch 9/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6945 - accuracy: 0.0014 - val_loss: 0.6943 - val_accuracy: 0.0012
Epoch 10/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6943 - accuracy: 0.0014 - val_loss: 0.6941 - val_accuracy: 0.0012
<keras.src.callbacks.History at 0x7fae5e0809a0>
```

```
import matplotlib.pyplot as plt

# Get the reconstructed images for the test set
reconstructed_imgs = autoencoder.predict(x_test_noisy)

# Choose a random image from the test set
n = 10 # index of the image to be plotted
plt.figure(figsize=(10, 5))

# Plot the original noisy image
ax = plt.subplot(1, 2, 1)
plt.imshow(x_test_noisy[n].reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
ax.set_title("Noisy Image")

# Plot the reconstructed image
ax = plt.subplot(1, 2, 2)
plt.imshow(reconstructed_imgs[n].reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
ax.set_title("Reconstructed Image")

plt.show()
```

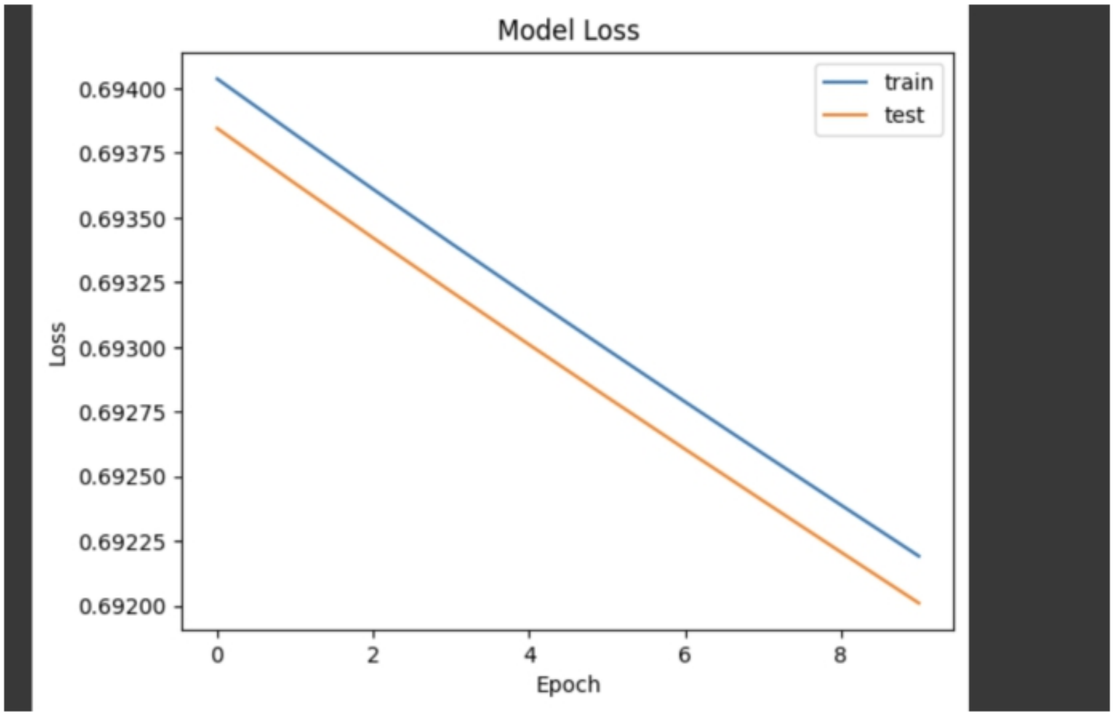


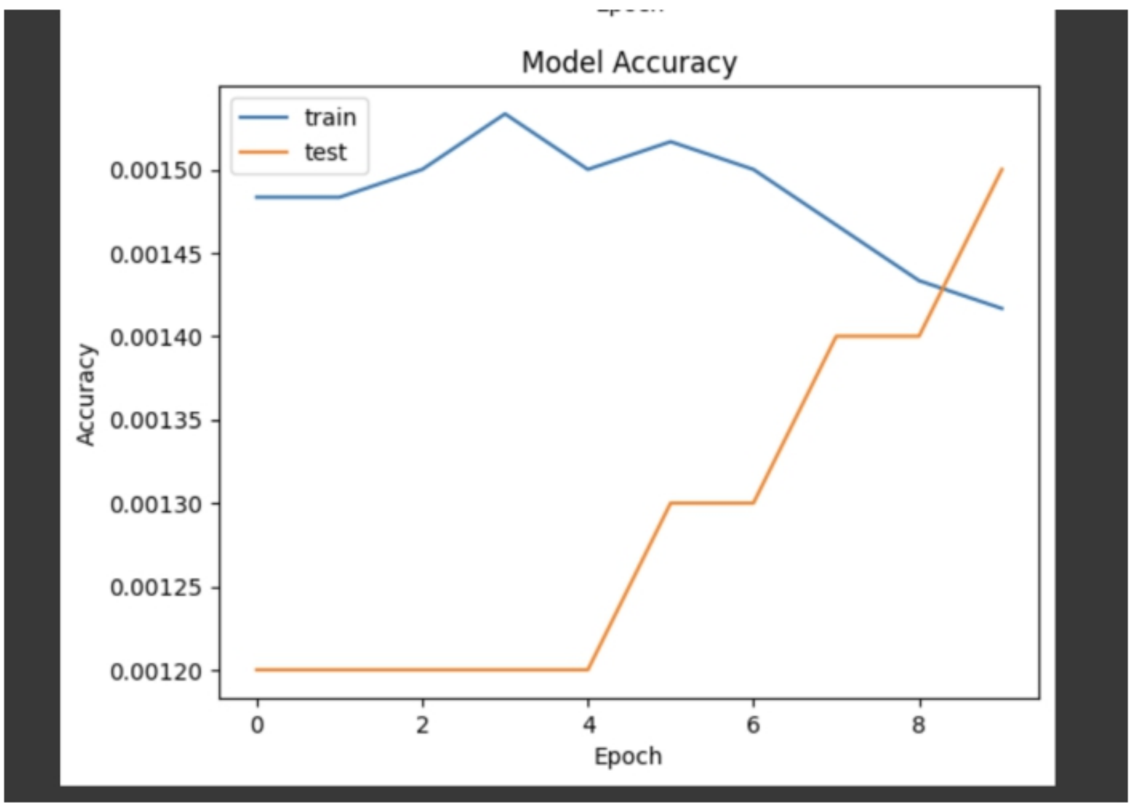
```
import matplotlib.pyplot as plt
plt.show the autoencoder
history = autoencoder.fit(x_train_noisy, x_train,
                epochs=10,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test_noisy, x_test_noisy))

plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```

Output:

```
Epoch 1/10
235/235 [==============================] - 4s 15ms/step - loss: 0.6940 - accuracy: 0.0015 - val_loss: 0.6938 - val_accuracy: 0.0012
Epoch 2/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6938 - accuracy: 0.0015 - val_loss: 0.6936 - val_accuracy: 0.0012
Epoch 3/10
235/235 [==============================] - 3s 15ms/step - loss: 0.6936 - accuracy: 0.0015 - val_loss: 0.6934 - val_accuracy: 0.0012
Epoch 4/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6934 - accuracy: 0.0015 - val_loss: 0.6932 - val_accuracy: 0.0012
Epoch 5/10
235/235 [==============================] - 3s 15ms/step - loss: 0.6932 - accuracy: 0.0015 - val_loss: 0.6930 - val_accuracy: 0.0012
Epoch 6/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6930 - accuracy: 0.0015 - val_loss: 0.6928 - val_accuracy: 0.0013
Epoch 7/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6928 - accuracy: 0.0015 - val_loss: 0.6926 - val_accuracy: 0.0013
Epoch 8/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6926 - accuracy: 0.0015 - val_loss: 0.6924 - val_accuracy: 0.0014
Epoch 9/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6924 - accuracy: 0.0014 - val_loss: 0.6922 - val_accuracy: 0.0014
Epoch 10/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6922 - accuracy: 0.0014 - val_loss: 0.6920 - val_accuracy: 0.0015
```

Test data using Matplotlib:


Model Loss
train
test
Loss
0.69400
0.69375
0.69350
0.69325
0.69300
0.69275
0.69250
0.69225
0.69200
0
2
4
6
8
Epoch


Model Accuracy
train
test
Accuracy
0.00150
0.00145
0.00140
0.00135
0.00130
0.00125
0.00120
0
2
4
6
8
Epoch