Practical 4

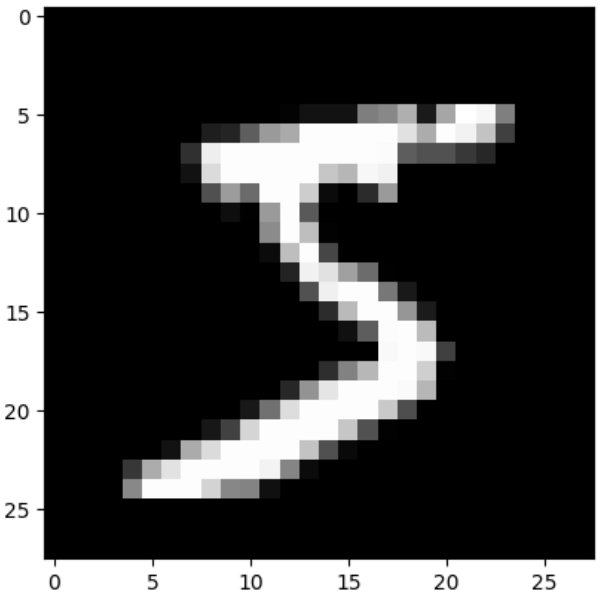
Aim: Implement deep learning for the Prediction of the autoencoder from the test data (e.g. MNIST data set).

Code:

```
import tensorflow.keras as keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Input, Flatten,Reshape,
LeakyReLU as LR,Activation, Dropout
from tensorflow.keras.models import Model, Sequential
from matplotlib import pyplot as plt
from IPython import display # If using IPython, Colab or Jupyter
import numpy as np

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train/255.0
x_test = x_test/255.0
# Plot image data from x_train
plt.imshow(x_train[0], cmap = "gray")
plt.show()
```

Output:

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step



Encoder Network Definition:

```
LATENT_SIZE = 32
```

```
encoder = Sequential([Flatten(input_shape = (28,
28)),Dense(512),LR(),Dropout(0.5),Dense(256),LR(),Dropout(0.5),Dense(12
8),

LR(),Dropout(0.5),Dense(64),LR(),Dropout(0.5),Dense(LATENT_SIZE),LR()])

decoder = Sequential([Dense(64, input_shape =
(LATENT_SIZE,)),LR(),Dropout(0.5),Dense(128),LR(),Dropout(0.5),Dense(25
6),
LR(),Dropout(0.5),Dense(512),LR(),Dropout(0.5),Dense(784),Activation("s
igmoid"),Reshape((28, 28))])
```

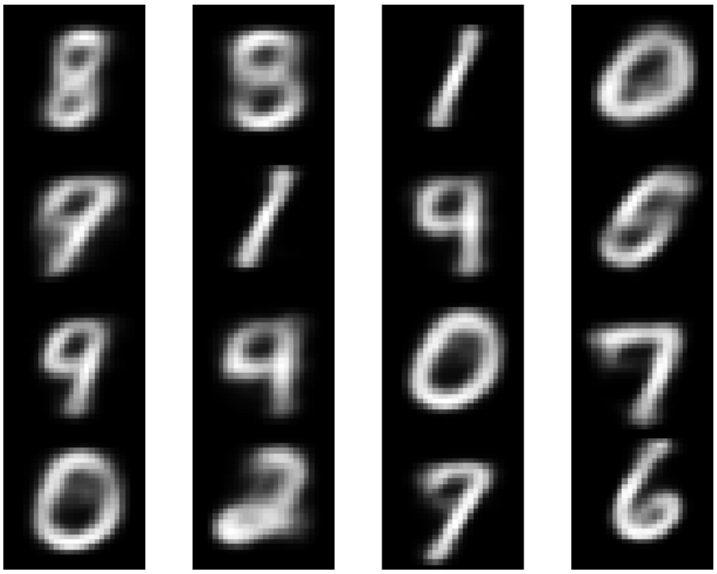
Training and Testing the model

```
img = Input(shape = (28, 28))
latent_vector = encoder(img)
output = decoder(latent_vector)
model = Model(inputs = img, outputs = output)
model.compile("nadam", loss = "binary_crossentropy")

EPOCHS = 5

for epoch in range(EPOCHS):
    fig, axs = plt.subplots(4, 4)
    rand = x_test[np.random.randint(0, 10000, 16)].reshape((4, 4, 1,
28, 28))
    display.clear_output() # If you imported display from IPython

    for i in range(4):
        for j in range(4):
            axs[i, j].imshow(model.predict(rand[i, j])[0], cmap =
"gray")
            axs[i, j].axis("off")

    plt.subplots_adjust(wspace = 0, hspace = 0)
    plt.show()
    print("-----------", "EPOCH", epoch, "-----------")
    model.fit(x_train, x_train)
```

Output:

```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
```

1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step



---------- EPOCH 4 ----------
1875/1875 [==============================] - 41s 22ms/step - loss: 0.1975

Evaluating

```
# Final evaluation of the model
test_reconstructions = model.predict(x_test)
test_loss = model.evaluate(x_test, x_test)
print("Final test loss:", test_loss)
```

Output:
313/313 [==============================] - 3s 8ms/step
313/313 [==============================] - 2s 6ms/step - loss: 0.1734
Final test loss: 0.1733781397342682