LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Kecerdasan Buatan
Kelas	: 3IA13
Praktikum ke-	: 3
Tanggal	: 13 Desember 2022
Materi	: Autoencoder
NPM	: 51420249
Nama	: Tiara Puspita
Ketua Asisten	: David
Jumlah Lembar	: 4

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2022

Isi Laporan

1. Blok kode yang digunakan untuk mengimport library yang dibutuhkan

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from keras.layers import Input, Dense
        from keras.models import Model
        from keras.datasets import mnist
```

2. Load data dari tensorflow keras. Ini digunakan untuk menormalisasikan data agar raise matrix berada pada range 0-1 guna untuk mengurangi waktu perhitungan.

```
In [6]: (x_train, y_train), (x_test, y_test) = mnist.load_data()

        x_train = x_train / 255.0
        x_test = x_test / 255.0

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

3. Reshaping data X_train(matrix) menjadi vector dengan library numpy dan fungsi reshape.

```
: x_train = np.reshape(x_train,
                       (x_train.shape[0], 784) )
  x_test = np.reshape (x_test,
                       (x_test.shape[0], 784) )
   #reshape menjadi vektor
```

4. Proses encode dan decode. Autoencode ini digunakan untuk mengurangi dimensi dari objek. Pada kasus ini digunakan untuk denoising objek.

```
input_img = Input(shape= (784,))
encoded = Dense(512, activation= 'relu')(input_img)
encoded = Dense(256, activation= 'relu')(encoded)
encoded = Dense(128, activation= 'relu')(encoded)
encoded = Dense(64, activation= 'relu')(encoded)
encoded = Dense(32, activation= 'relu')(encoded)

decoded = Dense(64, activation= 'relu')(encoded)
decoded = Dense(128, activation= 'relu')(decoded)
decoded = Dense(256, activation= 'relu')(decoded)
decoded = Dense(512, activation= 'relu')(decoded)
decoded = Dense(784, activation= 'sigmoid') (decoded)
autoencoder = Model(input_img, decoded)
```

Membandingkan input dan output. Data dikompile dengan menggunakan optimizer adam dan loss mean_squared_error, kamudian dilakukan training.

```
In [13]: autoencoder.compile(optimizer='adam', loss='mean_squared_error') #membandingkan input dan output
         autoencoder.fit(x_train, x_train,
         epochs=10,
         batch_size=100,
         shuffle=True,
         validation_data=(x_test, x_test))

         Epoch 1/10
         600/600 [==============================] - 15s 23ms/step - loss: 0.0441 - val_loss: 0.0256
         Epoch 2/10
         600/600 [==============================] - 14s 24ms/step - loss: 0.0217 - val_loss: 0.0187
         Epoch 3/10
         600/600 [==============================] - 14s 23ms/step - loss: 0.0177 - val_loss: 0.0161
         Epoch 4/10
         600/600 [==============================] - 14s 22ms/step - loss: 0.0153 - val_loss: 0.0142
         Epoch 5/10
         600/600 [==============================] - 13s 22ms/step - loss: 0.0138 - val_loss: 0.0134
         Epoch 6/10
         600/600 [==============================] - 14s 24ms/step - loss: 0.0128 - val_loss: 0.0126
         Epoch 7/10
         600/600 [==============================] - 14s 23ms/step - loss: 0.0120 - val_loss: 0.0118
         Epoch 8/10
         600/600 [==============================] - 13s 22ms/step - loss: 0.0113 - val_loss: 0.0111
         Epoch 9/10
         600/600 [==============================] - 13s 22ms/step - loss: 0.0106 - val_loss: 0.0105
         Epoch 10/10
         600/600 [==============================] - 14s 23ms/step - loss: 0.0100 - val_loss: 0.0104

Out[13]: <keras.callbacks.History at 0x7f845e920910>
```

5. Lakukan prediksi data.

```
In [14]: predicted = autoencoder.predict(x_test)

         313/313 [==============================] - 2s 5ms/step
```

6. Visualisasi hasil denoising dengan menggunakan autoencoder. Objek gambar bagian atas merupakan input dan bagian bawah merupakan output.

```
In [16]: n= 10

         plt.figure (figsize=(20,4))

         for i in range (n):
                 ax = plt.subplot(2, n, i+1)
                 plt.imshow(x_test[i].reshape(28, 28))
                 plt.gray()
                 ax.get_xaxis().set_visible(False)
                 ax.get_yaxis().set_visible(False)

                 ax = plt.subplot(2, n, i+ 1+ n)
                 plt.imshow(predicted[i].reshape(28, 28))
                 plt.gray()
                 ax.get_xaxis().set_visible(False)
                 ax.get_yaxis().set_visible(False)

         plt.show()
```