

LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : KA
Kelas : 3IA19
Praktikum : 3
Tanggal : 11/02/2023
Materi : Unsupervised Learning & Encoder
NPM : 50421859
Nama : Muhamad Ariel Dwi P
Ketua Asisten :
Nama Asisten : MUHAMMAD HAUZAN DINI FAKHRI
Paraf Asisten :
Jumlah Lembar : 10

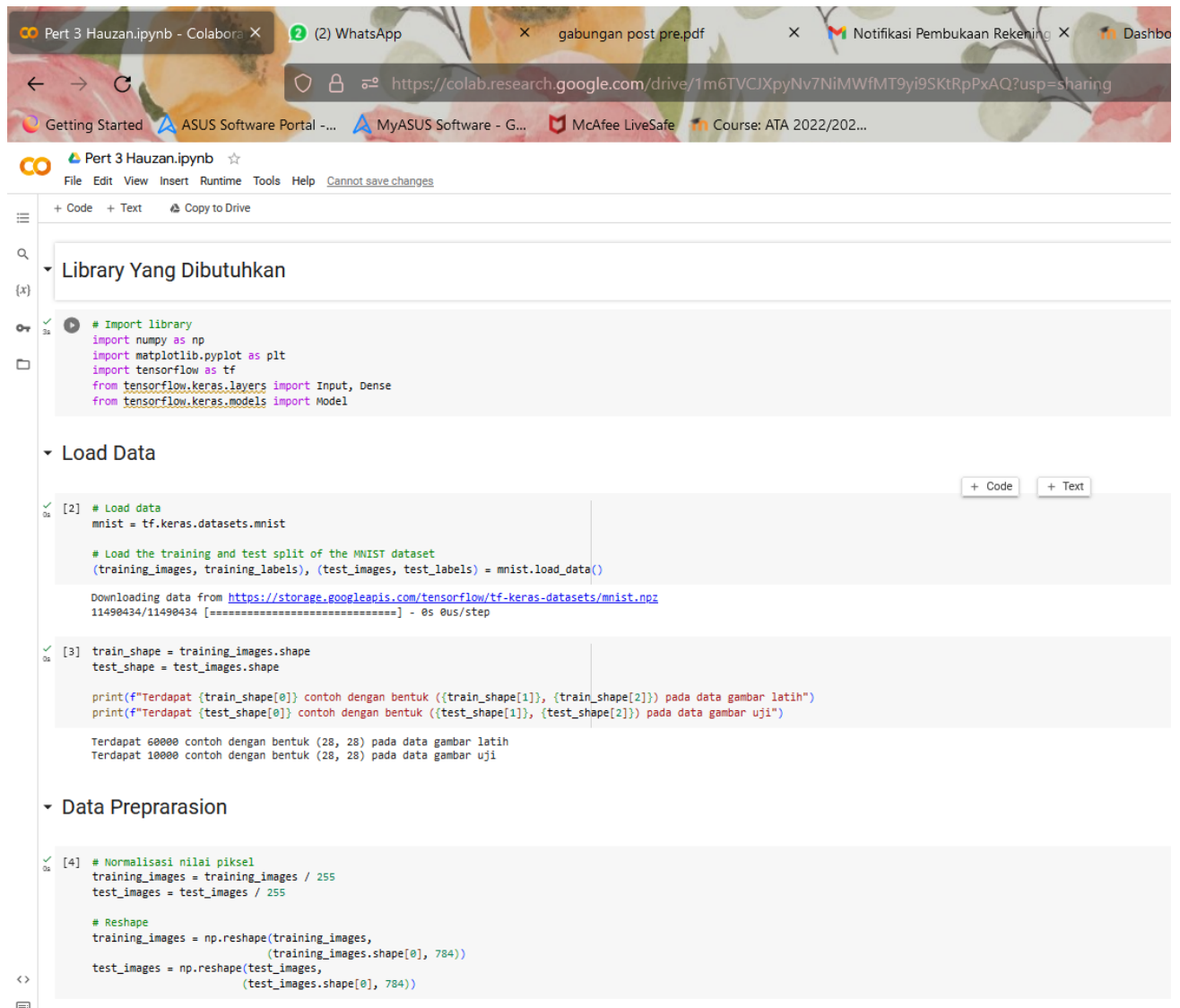


**LABORATORIUM INFORMATIKA
UNIVERSITAS GUNADARMA
2023**

LISTING

1. Screenshot saja semua kodingan notebook dan buat penjelasannya?

Jawab:



```
# Import library
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model

# Load data
mnist = tf.keras.datasets.mnist

# Load the training and test split of the MNIST dataset
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()

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

train_shape = training_images.shape
test_shape = test_images.shape

print(f"Terdapat {train_shape[0]} contoh dengan bentuk {(train_shape[1]), (train_shape[2])} pada data gambar latih")
print(f"Terdapat {test_shape[0]} contoh dengan bentuk {(test_shape[1]), (test_shape[2])} pada data gambar uji")

Terdapat 60000 contoh dengan bentuk (28, 28) pada data gambar latih
Terdapat 10000 contoh dengan bentuk (28, 28) pada data gambar uji

# Normalisasi nilai piksel
training_images = training_images / 255
test_images = test_images / 255

# Reshape
training_images = np.reshape(training_images,
                             (training_images.shape[0], 784))
test_images = np.reshape(test_images,
                          (test_images.shape[0], 784))
```

Kode ini digunakan untuk mengimpor beberapa library yang diperlukan dalam pemrograman Python, khususnya untuk pengolahan data dan pembuatan model machine learning:

1. `'import numpy as np'`: Mengimpor library ****NumPy**** dan memberinya alias `'np'`. NumPy adalah library yang digunakan untuk komputasi numerik di Python, seperti operasi matriks.
2. `'import matplotlib.pyplot as plt'`: Mengimpor modul `'pyplot'` dari library ****Matplotlib**** dan memberinya alias `'plt'`. Matplotlib adalah library yang digunakan untuk visualisasi data di Python, seperti membuat plot dan grafik.
3. `'import tensorflow as tf'`: Mengimpor library ****TensorFlow**** dan memberinya alias `'tf'`. TensorFlow adalah library yang digunakan untuk machine learning dan deep learning.

4. ``from tensorflow.keras.layers import Input, Dense``: Mengimpor modul ``Input`` dan ``Dense`` dari sub-library ``layers`` dalam TensorFlow Keras. Modul ini digunakan untuk membuat layer input dan dense (fully connected) dalam model neural network.

5. ``from tensorflow.keras.models import Model``: Mengimpor modul ``Model`` dari sub-library ``models`` dalam TensorFlow Keras. Modul ini digunakan untuk membuat model neural network.

Kode ini digunakan untuk memuat dataset MNIST (Modified National Institute of Standards and Technology) yang merupakan dataset angka tulisan tangan, dan membaginya menjadi data latihan (training) dan data uji (test):

1. ``mnist = tf.keras.datasets.mnist``: Mengimpor dataset MNIST dari TensorFlow Keras.
2. ``(training_images, training_labels), (test_images, test_labels) = mnist.load_data()``: Memuat dataset MNIST dan membaginya menjadi dua bagian:
 - ``training_images`` dan ``training_labels`` adalah gambar dan label untuk data latihan.
 - ``test_images`` dan ``test_labels`` adalah gambar dan label untuk data uji.

Setiap gambar dalam dataset adalah gambar grayscale 28x28 pixel yang mewakili angka dari 0 hingga 9, dan labelnya adalah angka yang sesuai.

Kode ini digunakan untuk mengetahui bentuk (shape) dari data gambar latihan dan data gambar uji, serta mencetak jumlah dan bentuk dari data tersebut:

1. ``train_shape = training_images.shape``: Mendapatkan bentuk dari ``training_images`` dan menyimpannya dalam variabel ``train_shape``. Bentuk ini mencakup jumlah gambar dan dimensi dari setiap gambar (misalnya, (60000, 28, 28) untuk 60000 gambar berukuran 28x28 pixel).
2. ``test_shape = test_images.shape``: Sama seperti di atas, tetapi untuk ``test_images``.
3. ``print(f'Terdapat {train_shape[0]} contoh dengan bentuk ({train_shape[1]}, {train_shape[2]}) pada data gambar latih')``: Mencetak jumlah dan bentuk dari data gambar latihan.

4. ``print(f'Terdapat {test_shape[0]} contoh dengan bentuk ({test_shape[1]}, {test_shape[2]}) pada data gambar uji')``: Mencetak jumlah dan bentuk dari data gambar uji.

Jadi, kode ini memberikan informasi tentang jumlah total gambar dan dimensi dari setiap gambar dalam dataset latihan dan uji.

Kode ini melakukan dua hal penting dalam persiapan data untuk model machine learning:

1. ****Normalisasi nilai piksel****: Nilai piksel dalam gambar biasanya berkisar antara 0 dan 255. Dalam banyak kasus, terutama saat bekerja dengan model deep learning, lebih baik untuk normalisasi nilai piksel menjadi rentang 0 dan 1. Ini dilakukan dengan membagi setiap nilai piksel dengan 255 (nilai piksel maksimum). Normalisasi ini membantu algoritma untuk berkonvergensi atau "belajar" lebih cepat.

```
```python
training_images = training_images / 255
test_images = test_images / 255
```
```

2. ****Reshape****: Gambar dalam dataset MNIST adalah gambar 2D berukuran 28x28 piksel. Namun, untuk memasukkan gambar ini ke dalam model neural network, kita perlu mengubah bentuknya menjadi vektor 1D. Fungsi ``np.reshape`` digunakan untuk mengubah bentuk gambar dari (28, 28) menjadi (784,), yang berarti gambar tersebut sekarang adalah vektor dengan panjang 784.

```
```python
training_images = np.reshape(training_images,
 (training_images.shape[0], 784))
test_images = np.reshape(test_images,
 (test_images.shape[0], 784))
```
```

Jadi, kode ini menyiapkan data gambar agar dapat diproses oleh model machine learning dengan lebih efisien.

```
[4] test_images = np.reshape(test_images,
                              (test_images.shape[0], 784))

Membangun Model

[5] inputs = Input(shape=(784,))
    h_encode = Dense(512, activation='relu')(inputs)
    h_encode = Dense(256, activation='relu')(h_encode)
    h_encode = Dense(128, activation='relu')(h_encode)
    h_encode = Dense(64, activation='relu')(h_encode)

    coded = Dense(32, activation='relu')(h_encode)

    h_decode = Dense(64, activation='relu')(coded)
    h_decode = Dense(128, activation='relu')(h_decode)
    h_decode = Dense(256, activation='relu')(h_decode)
    h_decode = Dense(512, activation='relu')(h_decode)

    outputs = Dense(784, activation='sigmoid')(h_decode)
    autoencoder = Model(inputs, outputs)

[6] autoencoder.compile(optimizer='adam',
                        loss='mean_squared_error',
                        metrics=['mse'])
    autoencoder.fit(training_images, training_images,
                    epochs=10,
                    batch_size=200,
                    shuffle=True,
                    validation_data=(test_images, test_images))

Epoch 1/10
300/300 [=====] - 13s 6ms/step - loss: 0.0547 - mse: 0.0547 - val_loss: 0.0319 - val_mse: 0.0319
Epoch 2/10
300/300 [=====] - 2s 5ms/step - loss: 0.0264 - mse: 0.0264 - val_loss: 0.0224 - val_mse: 0.0224
Epoch 3/10
300/300 [=====] - 2s 5ms/step - loss: 0.0205 - mse: 0.0205 - val_loss: 0.0183 - val_mse: 0.0183
Epoch 4/10
300/300 [=====] - 2s 5ms/step - loss: 0.0175 - mse: 0.0175 - val_loss: 0.0163 - val_mse: 0.0163
Epoch 5/10
300/300 [=====] - 2s 5ms/step - loss: 0.0158 - mse: 0.0158 - val_loss: 0.0151 - val_mse: 0.0151
Epoch 6/10
300/300 [=====] - 2s 5ms/step - loss: 0.0145 - mse: 0.0145 - val_loss: 0.0138 - val_mse: 0.0138
Epoch 7/10
300/300 [=====] - 2s 5ms/step - loss: 0.0135 - mse: 0.0135 - val_loss: 0.0129 - val_mse: 0.0129
Epoch 8/10
300/300 [=====] - 2s 5ms/step - loss: 0.0127 - mse: 0.0127 - val_loss: 0.0123 - val_mse: 0.0123
Epoch 9/10
300/300 [=====] - 2s 5ms/step - loss: 0.0121 - mse: 0.0121 - val_loss: 0.0118 - val_mse: 0.0118
Epoch 10/10
300/300 [=====] - 2s 5ms/step - loss: 0.0114 - mse: 0.0114 - val_loss: 0.0111 - val_mse: 0.0111
<keras.src.callbacks.History at 0x7ce5e90b9d0>
```

Kode ini digunakan untuk membuat model autoencoder dengan TensorFlow Keras. Autoencoder adalah jenis jaringan saraf tiruan yang digunakan untuk belajar representasi data yang efisien (encoding) dalam cara yang tidak diawasi, biasanya untuk tujuan reduksi dimensi. Berikut adalah penjelasan dari setiap bagian kode:

1. ``inputs = Input(shape=(784,))``: Ini mendefinisikan layer input model. Bentuk inputnya adalah (784,), yang sesuai dengan bentuk gambar yang telah diubah menjadi vektor 1D.
2. ``h_encode = Dense(512, activation='relu')(inputs)``: Ini mendefinisikan layer Dense pertama dengan 512 neuron dan fungsi aktivasi ReLU. Layer ini dihubungkan ke layer input.
3. ``h_encode = Dense(256, activation='relu')(h_encode)``: Ini mendefinisikan layer Dense kedua dengan 256 neuron. Layer ini dihubungkan ke layer sebelumnya.

4. ``h_encode = Dense(128, activation='relu')(h_encode)``: Ini mendefinisikan layer Dense ketiga dengan 128 neuron.

5. ``h_encode = Dense(64, activation='relu')(h_encode)``: Ini mendefinisikan layer Dense keempat dengan 64 neuron.

6. ``coded = Dense(32, activation='relu')(h_encode)``: Ini mendefinisikan layer tengah (atau "coded") dari autoencoder, dengan 32 neuron.

7. ``h_decode = Dense(64, activation='relu')(coded)``: Ini mendefinisikan layer Dense pertama dari decoder, dengan 64 neuron.

8. ``h_decode = Dense(128, activation='relu')(h_decode)``: Ini mendefinisikan layer Dense kedua dari decoder, dengan 128 neuron.

9. ``h_decode = Dense(256, activation='relu')(h_decode)``: Ini mendefinisikan layer Dense ketiga dari decoder, dengan 256 neuron.

10. ``h_decode = Dense(512, activation='relu')(h_decode)``: Ini mendefinisikan layer Dense keempat dari decoder, dengan 512 neuron.

11. ``outputs = Dense(784, activation='sigmoid')(h_decode)``: Ini mendefinisikan layer output model, dengan 784 neuron dan fungsi aktivasi sigmoid.

12. ``autoencoder = Model(inputs, outputs)``: Ini membuat model Keras dari layer input dan output yang telah didefinisikan.

Jadi, kode ini membuat model autoencoder yang dapat digunakan untuk belajar representasi data gambar yang efisien.

Kode ini digunakan untuk mengkompilasi dan melatih model autoencoder yang telah dibuat:

2. ``autoencoder.fit(training_images, training_images, epochs=10, batch_size=200, shuffle=True, validation_data=(test_images, test_images))``: Ini melatih model dengan data latihan. Parameter yang diberikan adalah sebagai berikut:

- `'training_images'`: Gambar yang akan digunakan sebagai input untuk pelatihan.
- `'training_images'`: Karena ini adalah autoencoder, gambar yang sama digunakan sebagai target output.
- `'epochs=10'`: Model akan dilatih selama 10 epoch, yang berarti seluruh dataset akan melewati model sebanyak 10 kali.
- `'batch_size=200'`: Selama setiap iterasi pelatihan, 200 sampel akan diproses secara bersamaan.
- `'shuffle=True'`: Data latihan akan diacak sebelum setiap epoch.
- `'validation_data=(test_images, test_images)'`: Data uji digunakan sebagai validation data untuk memantau kinerja model pada data yang belum pernah dilihat sebelumnya selama pelatihan.

[illegible]

```
predicted = autoencoder.predict(test_images)
```

Dalam konteks ini, `test_images` adalah gambar yang model belum pernah lihat sebelumnya selama pelatihan. Model akan mencoba untuk merekonstruksi gambar ini berdasarkan apa yang telah dipelajarinya selama pelatihan. Outputnya, yang disimpan dalam variabel `predicted`, adalah gambar yang telah direkonstruksi oleh model. Jadi, kode ini digunakan untuk menguji seberapa baik model dapat merekonstruksi gambar yang belum pernah dilihat sebelumnya.

Kode ini digunakan untuk memvisualisasikan gambar asli dari dataset uji dan gambar yang telah direkonstruksi oleh model autoencoder. Berikut adalah penjelasan dari setiap bagian kode:

1. ``n = 12``: Ini menentukan jumlah gambar yang akan ditampilkan.
2. ``plt.figure(figsize=(30,6))``: Ini membuat figure baru dengan ukuran yang ditentukan.
3. Loop ``for i in range(n)``: Ini melakukan iterasi sebanyak ``n`` kali. Untuk setiap iterasi:
 - ``ax = plt.subplot(2, n, i+1)``: Ini membuat subplot pada posisi ``i+1`` dalam grid subplot dengan 2 baris dan ``n`` kolom.
 - ``plt.imshow(test_images[i].reshape(28, 28))``: Ini menampilkan gambar asli dari dataset uji pada subplot tersebut.
 - ``plt.gray()``: Ini mengatur colormap menjadi grayscale.
 - ``ax.get_xaxis().set_visible(False)`` dan ``ax.get_yaxis().set_visible(False)``: Ini menyembunyikan sumbu x dan y pada subplot tersebut.
 - ``ax = plt.subplot(2, n, i + 1 + n)``: Ini membuat subplot lain pada posisi ``i + 1 + n`` dalam grid subplot.
 - ``plt.imshow(predicted[i].reshape(28, 28), cmap='inferno')``: Ini menampilkan gambar yang telah direkonstruksi oleh model pada subplot tersebut dengan colormap 'inferno'.
 - ``plt.inferno()``: Ini mengatur colormap menjadi 'inferno'.

- Sumbu x dan y juga disembunyikan pada subplot ini.

4. `plt.show()`: Ini menampilkan figure yang telah dibuat.

Jadi, kode ini digunakan untuk membandingkan gambar asli dengan gambar yang telah direkonstruksi oleh model autoencoder.

Kode yang Anda berikan digunakan untuk menghitung dan mencetak ukuran data gambar MNIST sebelum dan setelah kompresi.

Berikut adalah penjelasan lebih detail:

- `size_test_images = test_images.nbytes` : Baris ini menghitung ukuran total (dalam byte) dari array `test_images` yang mungkin berisi gambar MNIST sebelum kompresi. Fungsi `nbytes` mengembalikan jumlah byte yang digunakan oleh array numpy.

- `size_predicted = predicted.nbytes` : Sama seperti di atas, baris ini menghitung ukuran total (dalam byte) dari array `predicted` yang mungkin berisi gambar MNIST setelah kompresi.

- `print("Ukuran Data MNist yang belum di kompres : ",size_test_images / 1000000,"Mb")` : Baris ini mencetak ukuran data `test_images` dalam megabyte (Mb). Ukuran dalam byte dibagi dengan 1.000.000 untuk mengubahnya menjadi megabyte.

- `print("Ukuran Data MNist yang sudah di kompres : ",size_predicted / 1000000,"Mb")` : Sama seperti di atas, baris ini mencetak ukuran data `predicted` dalam megabyte (Mb).

Jadi, kode ini pada dasarnya digunakan untuk membandingkan ukuran data gambar MNIST sebelum dan setelah proses kompresi. Ini bisa sangat berguna untuk melihat seberapa efektif metode kompresi dalam mengurangi ukuran data.

2. Ubah warna plot hasil rekonstruksi pada bagian prediksi model?

Jawab:

Menggunakan Warna inferno

