LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Kecerdasan Buatan
Kelas	: 3IA13
Praktikum ke-	: 1
Tanggal	: 2 Desember 2022
Materi	:
NPM	: 51420249
Nama	: Tiara Puspita
Ketua Asisten	: David
Jumlah Lembar	:

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2022

Isi Laporan

1. Blok kode yang digunakan untuk mengimport library yang dibutuhkan

```
In [*]: import numpy as np #komputasi numerik
        import matplotlib.pyplot as plt #visualisasi data
        import tensorflow as tf #backend
        from tensorflow.keras.datasets import mnist #dataset
        from sklearn.metrics import confusion_matrix #evaluasi
        import itertools #iterator untuk iterasi data
```

2. Load dataset mnist dari tensorflow keras.

```
In [2]: (x_train, y_train), (x_test, y_test) = mnist.load_data()

        Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
        11490434/11490434 [==============================] - 18s 2us/step
```

3. Print dimensi data dan array x_train[3]

```
jupyter KB_LA1_Tiara Puspita_51420249 Last Checkpoint: 3 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Markdown

In [5]: print('Bentuk(shape) atau dimensi data : (jumlah data, tinggi piksel, lebar piksel)', x_train.shape)
        print('Bentuk dari sisi data: \n')
        x_train[3]

        Bentuk(shape) atau dimensi data : (jumlah data, tinggi piksel, lebar piksel) (60000, 28, 28)
        Bentuk dari sisi data:


Out[5]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0, 124, 253, 255,  63,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,  96, 244, 251, 253,  62,   0,   0,   0,   0,
                  0,   0],
```

4. Reshape x_train, ini hanya mengubah bentuk tanpa mengubah data

```
In [7]: x_train=np.reshape(x_train,(x_train.shape[0], x_train.shape[1] * x_train.shape[2]))
        x_test=np.reshape(x_test, (x_test.shape[0], 784))
```

5. Print data yang sudah di reshape

```
In [6]: print('Data yang diubah menjadi 2 dimensi : ')
        x_train[0]

        Data yang diubah menjadi 2 dimensi :

Out[6]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
                 18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
                253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
```

6. Normalisasi tingkat kecerahan

```
#normalisasi tingkat kecerahan . Diubah dari 0-255 menjadi 0-1
x_train = x_train/255.0
x_test = x_test/255.0
```

7. Array x_train[0]

```
In [7]: x_train[1]

Out[7]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,  51, 159, 253, 159,  50,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,  48, 238, 252, 252, 252, 237,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                 54, 227, 253, 252, 239, 233, 252,  57,   6,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  10,  60,
                224, 252, 253, 252, 202,  84, 252, 253, 122,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 163, 252,
                252, 252, 253, 252, 252,  96, 189, 253, 167,   0,   0,   0,   0,
                  0,   0],
```

8. Membuat label dataset menjadi kategori atau kelas-kelas dengan menggunakan tensorflow

```
In [ ]: from tensorflow.keras.utils import to_categorical
        y_train = to_categorical(y_train)
        y_test = to_categorical(y_test)
```

```
In [8]: y_train

Out[8]: array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

```
In [9]: y_train

Out[9]: array([[0., 0., 0., ..., 0., 0., 0.],
               [1., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 0., 0.],
               ...,
               [0., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 1., 0.]], dtype=float32)
```

9. Membuat model machine learning

```
In [ ]: num_input = 28 * 28 #jumlah input vektor

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(500, input_dim = num_input, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(loss='categorical_crossentropy', #mengukur informasi yang hilang
              optimizer='adam', #pengaturan learning rate
              metrics=['accuracy'])
```

10. Training Data (semakin banyak loop akurasi semakin bagus)

```
In [*]: hist = model.fit(x_train, y_train,
                epochs = 15,
                batch_size = 200,
                validation_data = (x_test, y_test))

Epoch 1/15
300/300 [==============================] - 4s 9ms/step - loss: 6.1705 - accuracy: 0.8959 - val_loss: 1.5220 - val_accuracy: 0.9
427
Epoch 2/15
300/300 [==============================] - 2s 8ms/step - loss: 0.8875 - accuracy: 0.9553 - val_loss: 0.8517 - val_accuracy: 0.9
496
Epoch 3/15
300/300 [==============================] - 2s 8ms/step - loss: 0.4660 - accuracy: 0.9680 - val_loss: 0.6045 - val_accuracy: 0.9
641
Epoch 4/15
300/300 [==============================] - 2s 8ms/step - loss: 0.2624 - accuracy: 0.9770 - val_loss: 0.7119 - val_accuracy: 0.9
610
Epoch 5/15
300/300 [==============================] - 3s 9ms/step - loss: 0.1872 - accuracy: 0.9817 - val_loss: 0.6227 - val_accuracy: 0.9
623
Epoch 6/15
300/300 [==============================] - 3s 9ms/step - loss: 0.1764 - accuracy: 0.9817 - val_loss: 0.6220 - val_accuracy: 0.9
655
Epoch 7/15
 98/300 [========>.....................] - ETA: 1s - loss: 0.1369 - accuracy: 0.9855
```

11. Menyimpan model dan visualisasi

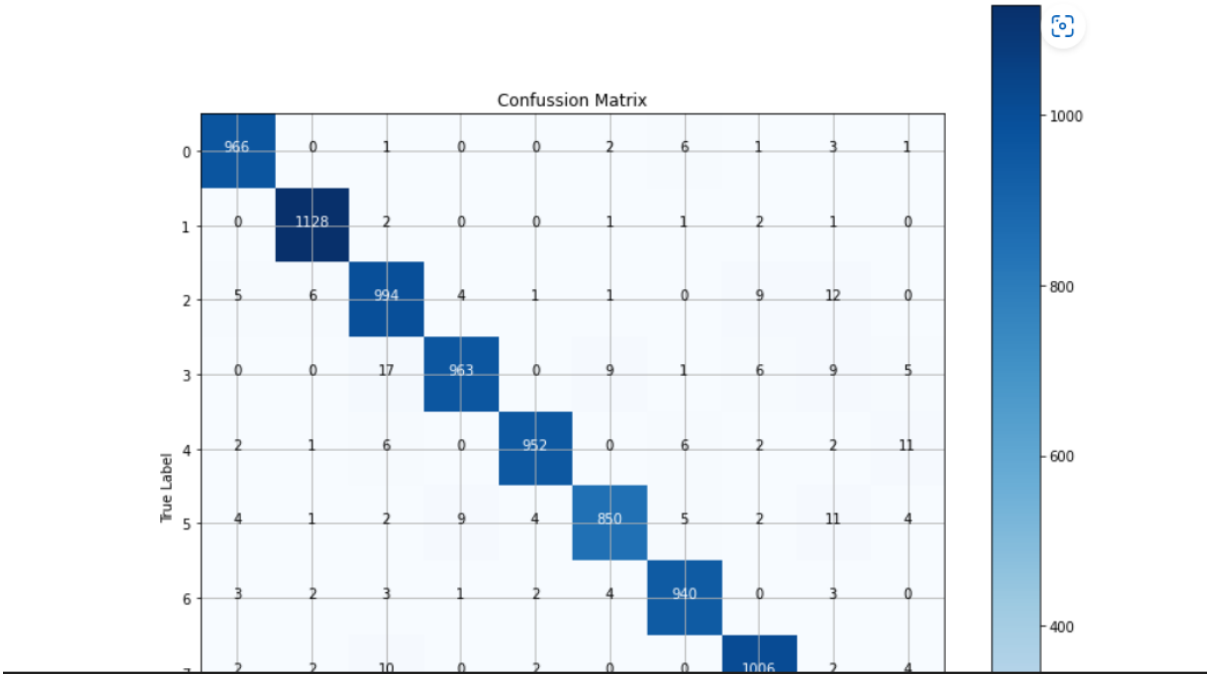
```
In [13]: model.save_weights('model_weights.h5') #menyimpan model
```

```
In [14]: from prompt_toolkit.layout.containers import HorizontalAlign
def plot_confusion_matrix(cm, classes, normalize = False,
                          title = 'Confussion Matrix',
                          cmap=plt.cm.Blues):
  plt.figure(figsize = (10,10))
  plt.imshow(cm, interpolation = 'nearest', cmap=cmap)
  plt.title(title)
  plt.colorbar()
  tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation = 90)
  plt.yticks(tick_marks, classes)
  if normalize:
    cm = cm.astype('float')/cm.sum(axis=1)[:, np.newaxis]

  thresh = cm.max() /2
  for i,j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j,i, cm[i,j],
             horizontalalignment = "center",
             color = "white" if cm[i,j] > thresh else "black")
    plt.grid('off')
    plt.tight_layout()
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
```

In [15]:
```
list_classes =[0,1,2,3,4,5,6,7,8,9]
Y_pred = model.predict(x_test)
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(y_test, axis=1)
confusion_mtx= confusion_matrix(Y_true, Y_pred_classes)
plot_confusion_matrix(confusion_mtx, classes= list_classes)
```

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



L

In [16]:
```
img = x_test[7777]
predicted = model.predict(np.reshape(img,(1,784)))
predicted
```

```
1/1 [==============================] - 0s 27ms/step
```

Out[16]:
```
array([[0.0000000e+00, 5.5085948e-24, 0.0000000e+00, 2.3357012e-23,
        0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 3.0657754e-37]], dtype=float32)
```

In [17]:
```
predictednumber= np.argmax(predicted)
predictednumber
```

Out[17]: 5

In [18]:
```
np.argmax(y_test[7777])
```

Out[18]: 5

In [ ]:
```
plt.imshow(np.reshape(x_test[7777],(28,28)))
plt.axis('off')
plt.sho
```

In [ ]: