LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : Pengantar Kecerdasan Buatan

Kelas : 3IA08

Praktikum ke- : 1

Tanggal : 24 Oktober 2020

Materi : Membuat M Learning Menggunakan Collab

NPM : 56418554

Nama : Satrio Wibowo

Ketua Asisten : Siti Athifah

Paraf Asisten :

Nama Asisten : Siti Athifah

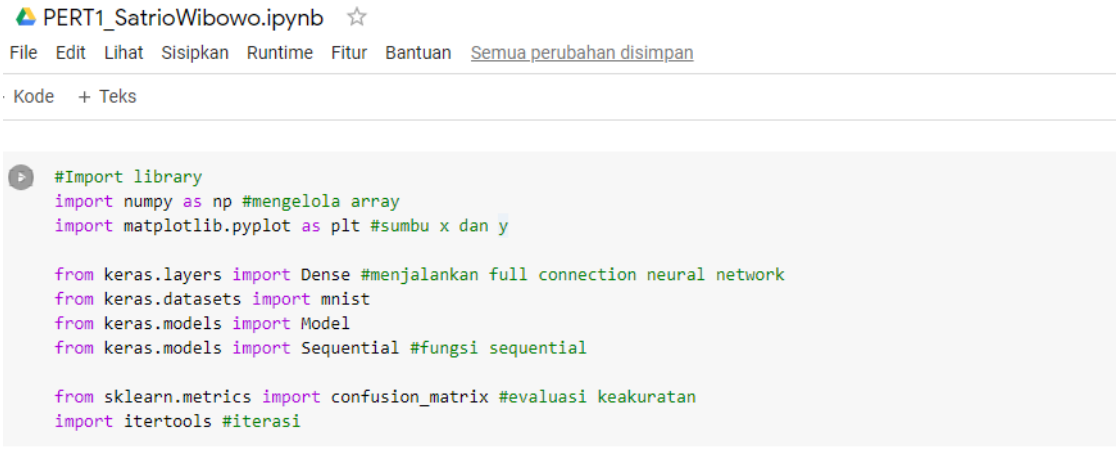
Jumlah Lembar : 6 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA 2020

Pada pertemuan pertama saya mempelajari tentang pembuatan model machine learning menggunakan Googl Colab. Kemudian pada laporan akhir ini saya akan membuat lagi model machine learning menggunakan Google Colab

\



2. Kemudian membuat variabel x_train, y_train, x_test, dan y_test dengan value mnist.load_data()

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

3. Cetak data shape dari variabel x_train.shape dan bentuk dari sisi data dari variabel x_train indeks ke-3

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

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

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,
```

4. Menggunakan reshape dari library np yang ditampung dalam variabel x_train dan x_test

```
[8]  x_train = np.reshape(x_train,(x_train.shape[0],784))
     x_test = np.reshape(x_test,(x_test.shape[0],784))
```

5. Menampilkan x_train dari indeks ke-1

```
print('Data yang diubah menjadi 2 dimensi: ')
x_train[1]
```

```
Data yang diubah menjadi 2 dimensi:
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,
```

6. Melakukan normalisasi pada x_train dan x_test dan menampilkan x_train indeks ke-0

```
[8]  from keras.utils import to_categorical
     y_train = to_categorical(y_train)
     y_test = to_categorical(y_test)
```

```
#normalisasi
x_train = x_train/255.0
x_test = x_test/255.0
```

```
x_train[0]
```

```
array([[0.        , 0.1372549 , 0.94509804, 0.88235294, 0.62745098,
        0.42352941, 0.00392157, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.31764706, 0.94117647, 0.99215686, 0.99215686, 0.46666667,
        0.09803922, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.17647059,
        0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.0627451 , 0.36470588,
        0.98823529, 0.99215686, 0.73333333, 0.        , 0.
```

7. Mengambil to_categorical dari library keras
8. Menampilkan y_train

```
 [[0., 1.],
  [1., 0.]]],


[[[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]],

 ...,

 [[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]]],


[[[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]],

 [[0., 1.],
  [1., 0.]],

 ...,

 [[0., 1.],
  [1., 0.]],

 [[1., 0.],
  [0., 1.]],

 [[0., 1.],
  [1., 0.]]], dtype=float32)
```

9. Menggunakan model dari library keras

```
num_input= 28*28 #jumlah input vektor

model=Sequential() #membuat model berurutan
model.add(Dense(500, input_dim = num_input,activation = 'relu')) #input data sebanyak num_input terhadap hidden layer
model.add(Dense(10, activation = 'softmax')) #untuk output sebanyak 10 secara kategori dan menggunakan softmax

model.compile(loss='categorical_crossentropy', #memperkirakan kesalahan model
              optimizer = 'adam', #optimasi data training
              metrics=['accuracy']) #mengevaluasi kemampuan model
```

10. Menggunakan variabel hist untuk menampung model.fit

```
his = model.fit(x_train,y_train, #melatih data latih x dan target y
                epochs = 15, #jumlah looping read data
                batch_size=200, #jumlah data yang diinput
                validation_data=(x_test,y_test)) #melatih data validasi untuk menguji data
```

```
Epoch 1/15
300/300 [==============================] - 3s 11ms/step - loss: 0.3040 - accuracy: 0.9143 - val_loss: 0.1564 - val_accuracy: 0.9559
Epoch 2/15
300/300 [==============================] - 3s 10ms/step - loss: 0.1278 - accuracy: 0.9640 - val_loss: 0.1060 - val_accuracy: 0.9678
Epoch 3/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0845 - accuracy: 0.9754 - val_loss: 0.0852 - val_accuracy: 0.9744
Epoch 4/15
300/300 [==============================] - 3s 11ms/step - loss: 0.0606 - accuracy: 0.9830 - val_loss: 0.0807 - val_accuracy: 0.9744
Epoch 5/15
300/300 [==============================] - 3s 11ms/step - loss: 0.0470 - accuracy: 0.9868 - val_loss: 0.0765 - val_accuracy: 0.9772
Epoch 6/15
300/300 [==============================] - 3s 11ms/step - loss: 0.0357 - accuracy: 0.9901 - val_loss: 0.0734 - val_accuracy: 0.9762
Epoch 7/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0277 - accuracy: 0.9927 - val_loss: 0.0658 - val_accuracy: 0.9803
Epoch 8/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0213 - accuracy: 0.9946 - val_loss: 0.0641 - val_accuracy: 0.9814
Epoch 9/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0160 - accuracy: 0.9966 - val_loss: 0.0681 - val_accuracy: 0.9794
Epoch 10/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0129 - accuracy: 0.9973 - val_loss: 0.0604 - val_accuracy: 0.9815
Epoch 11/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0100 - accuracy: 0.9982 - val_loss: 0.0664 - val_accuracy: 0.9798
Epoch 12/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0082 - accuracy: 0.9985 - val_loss: 0.0718 - val_accuracy: 0.9779
Epoch 13/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0063 - accuracy: 0.9989 - val_loss: 0.0623 - val_accuracy: 0.9818
Epoch 14/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0045 - accuracy: 0.9994 - val_loss: 0.0642 - val_accuracy: 0.9828
Epoch 15/15
300/300 [==============================] - 3s 10ms/step - loss: 0.0030 - accuracy: 0.9998 - val_loss: 0.0635 - val_accuracy: 0.9828
```

11. Menggunakan model.save_weights

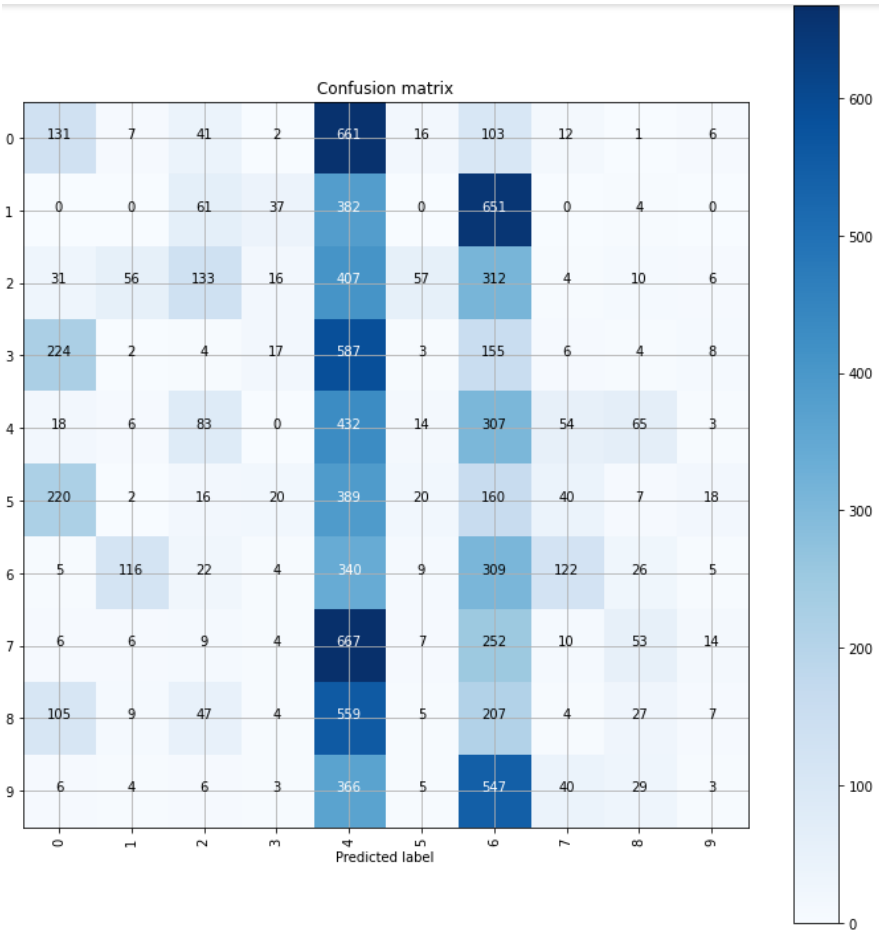
```
model.save_weights('model_weights.h5')
```

12. Membuatsebuah fungsi dengan nama plot_confusion_matrix serta membuat sebuah percabangan

```
def plot_confusion_matrix(cm, classes, normalize=False,
                          title='Confusion 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')
```

13. Membuat sebuah list dengan 10 elements serta variabel y_pred, y_pred_classes, Y_true, confusion_mtx dan memanggil fungsi plot_confusion_matrix kemudian dirun maka akan muncul grafik



14. Pada Variabel ini saya akan mempredik x_test pada gambar yang diatas

```
[21] img = x_test[1678] #index data, data ke 1678
     predicted = model.predict(np.reshape(img, (1,784)))
```

```
[22] predicted

     array([[0.04967107, 0.06024818, 0.09792227, 0.04133844, 0.09881792,
             0.3344078 , 0.08074284, 0.08275189, 0.09363806, 0.06046149]],
           dtype=float32)
```

15. Variabel predicted_number saya masukkan value np.argmax dengan parameter predicted kemudian dirun menghasilkan 6

```
[24] predictnumber = np.argmax(predicted)
```

```
predictnumber

5
```

16. Menampilkan gambar berdasarkan output nya yaitu 2

```
[ ] np.argmax(y_test[1678])
```

```
2
```

```
plt.imshow(np.reshape(x_test[1678], (28,28))) #menampilkan plot/sumbu gambar
plt.axis('off') #menghilangkan sumbu x dan y
plt.show()
```