MODUL 12 – Feature Matching, Face Detection, Face Tracking

A. TUJUAN

- Mahasiswa mampu memahami konsep Feature Matching
- Mahasiswa mampu memahami konsep face detection
- Mahasiswa dapat mengimplementasikan beberapa metode dalam proses Feature Matching dan Face detection menggunakan Python pada Google Colab

B. ALAT DAN BAHAN

1. PC/LAPTOP
2. Github
3. *Google Colaborator*

C. DASAR TEORI

C.1 Konsep Feature Matching

Feature matching melakukan ekstraksi ekstraksi fitur penting dari sebuah citra menggunakan ide dasar dari corner, edge, dan contour detection. Selanjutnya akan dilakukan perhitungan distance untuk mencari kecocokan antara image sumber dengan image template. Dengan menggunakan cara ini, anda tidak perlu lagi menggunakan template yang ada di image sumber. Jika objek pada citra template tidak berukuran sama dengan citra sumber, deteksi tetap bisa dilakukan dengan baik. Pada praktikum kali ini akan dicobakan 2 metode Feature Matching yaitu:

1. Brute-Force Matching menggunakan ORB Descriptor (Orient FAST and Rotated BRIEF)
2. Brute-Force Matching menggunakan SIFT Descriptor dan Test Ratio (Scale-Invariant Feature Transform)

ORB adalah berasal dari "OpenCV Labs". Algoritma ini diajukan oleh Ethan Rublee, Vincent Rabaud, Kurt Konolige dan Gary R. Bradski dalam artikel mereka berjudul "*ORB: An efficient alternative to SIFT or SURF*" pada tahun 2011. Seperti yang tertulis pada judulnya, ORB adalah alternatif pengganti dari SIFT dan SURF dalam hal biaya komputasi, kinerja matching dan terutama adalah paten. SIFT dan SURF dipatenkan dan kita harus membayar untuk menggunakannya, beda dengan ORB yang tidak perlu membayar untuk penggunaannya.

ORB pada dasarnya adalah perpaduan dari detektor keypoint FAST dan deskriptor BRIEF dengan banyak modifikasi untuk meningkatkan kinerjanya. Sebagai langkah awal, FAST digunakan untuk menemukan keypoint, lalu Harris Corner detection digunakan untuk mencari N-point tertinggi. Akan tetapi FAST tidak menghitung orientasinya. Modifikasi dilakukan agar algoritma ini invariant terhadap orientasi. Penjelasan lebih detil dapat dilihat pada dokumentasi OpenCV: https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

ORB Descriptor telah disediakan pada library Opencv cv.ORB(). Berikut adalah code untuk menunjukkan penggunakan ORB Descriptor. Pada praktikum ini anda diminta untuk menggunakan file reeses_puff dan cereals yang telah disediakan. Perhatikan pada 2 gambar tersebut, reeses puff memiliki resolusi yang lebih besar daripada reeses puff pada image cereals.

```
def display(img,cmap='gray'):
  fig = plt.figure(figsize=(12,10))
  ax = fig.add_subplot(111)
  ax.imshow(img,cmap='gray')
```

```
reeses = cv.imread('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/facedet/reeses_puffs.png',0)
display(reeses)
```


```
cereals = cv.imread('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/facedet/many_cereals.jpg',0)
display(cereals)
```

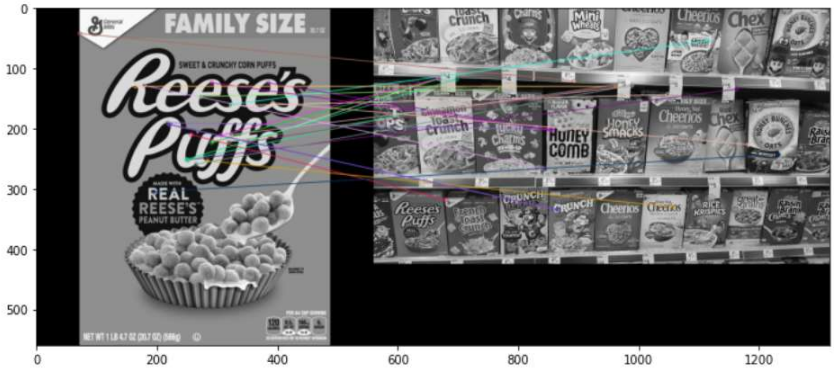

```
orb = cv.ORB_create()
kp1,des1 = orb.detectAndCompute(reeses,None)
kp2,des2 = orb.detectAndCompute(cereals,None)
bf = cv.BFMatcher(cv.NORM_HAMMING,crossCheck = True)
matches = bf.match(des1,des2)
single_match = matches[0]
single_match.distance
```

```
71.0
```

```
len(matches)
```

```
139
```

```
matches = sorted(matches,key=lambda x:x.distance)
reeses_match = cv.drawMatches(reeses,kp1,cereals,kp2,matches[:25],None,flags=2)
display(reeses_match)
```



Perhatikan hasil pencocokan diatas. Dengan memilih 25 titik yang paling cocok, hasil dari pencocokan pada kasus diatas tidak akurat sama sekali.

Pada pembahasan sebelumnya tentang Harris corner detection misal, metode-metodenya adalah rotation-invariant, artinya walaupun image dalam kondisi terrotasi, corner dapat dideteksi dengan baik. Hal ini jelas sekali, karena corner akan tetap berupa corner walaupun objek dirotasikan. Berbeda kasus dengan scaling. Corner bisa jadi bukan lagi corner jika citra discaling. Perhatikan gambar berikut. Corner pada citra kecil dengan window berukuran kecil akan berupa garis jika diperbesar dan menggunakan window berukuran sama. Hal ini menunjukkan bahwa Harris corner merupakan algoritma yang scale-invariant.

Pada tahun 2004, D. Lowe dari University of British Columbia, menemukan algoritma baru, Scale Invariant Feature Transform (SIFT) dalam artikelnya yang berjudul *"Distinctive Image Features from Scale-Invariant Keypoints",* yang mengekstrak keypoint dan menghitung deskriptornya. Artikel ini mudah dipahami dan dianggap sebagai materi terbaik yang tersedia di SIFT.

Terdapat 4 tahap yang terjadi pada algoritma SIFT. Penjelasan lebih detil dapat anda baca pada dokumentasi OpenCV: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html . Berikut akan ditunjukkan penggunaan algoritma SIFT pada image reese puffs dan cereal. Hasil yang didapatkan juga jauh lebih baik jika dibandingkan dengan ORB Descriptor. Untuk menggunakan SIFT descriptor, anda harus menggunakan OpenCV Versi 4.4.0.44 keatas. Lakukan instalasi menggunakan pip terlebih dahulu sebalum menjalankan code SIFT.

```
!pip install opencv-contrib-python==4.4.0.44
```

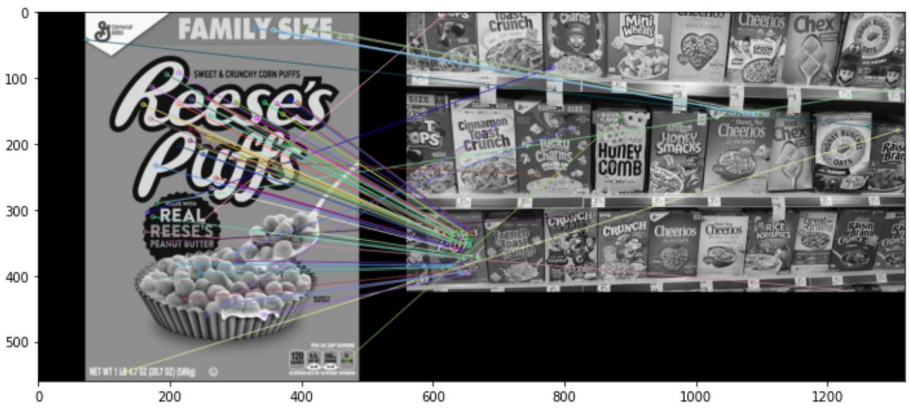
```
sift = cv.xfeatures2d.SIFT_create()
kp1,des1 = sift.detectAndCompute(reeses,None)
kp2,des2 = sift.detectAndCompute(cereals, None)
bf = cv.BFMatcher()
matches = bf.knnMatch(des1,des2,k=2)

good = []

#Makin kecil jarak makin cocok
#ratio match1 < 75% Match2
for match1,match2 in matches:
  #jika jarak match1 lebih kecil dari jarak 75% match2
  #descriptor disimpan
  if match1.distance < 0.75*match2.distance:
    good.append([match1])

len(good)
```

```
79
```

```
len(matches)
```

```
1501
```

```
sift_matches = cv.drawMatchesKnn(reeses,kp1,cereals,kp2,good,None,flags=2)
display(sift_matches)
```

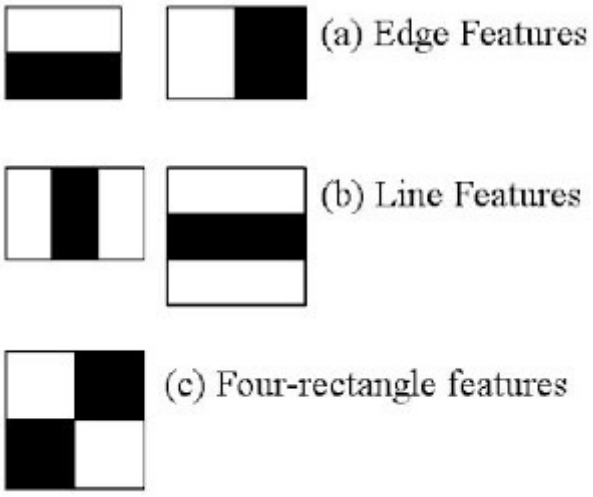


C.2 Konsep Face Detection

Pada materi ini akan dijelaskan cara kerja face detection menggunakan Haar Cascades, yaitu komponen kunci dari framework deteksi object Viola-Jones. Metode ini dapat mendeteksi wajah dengan cepat pada sebuah image dan mengetahui lokasinya dengan tepat. Metode ini digunakan sebagai tahap awal dalam mengenali wajah (Facial Recognition).

Pada tahun 2001, Paul Viola dan Michael Jones mempublikasikan metode face detection berdasarkan pada konsep sederhana dari beberapa fitur kunci. Ide lain yang digunakan adalah melakukan perhitungan awal dari integral image untuk mempercepat waktu komputasi.

Perhatikan tipe-tipe fitur utama yang diajukan oleh Viola dan Jones.



Tiap fitur adalah merupakan nilai tunggal yang didapatkan dengan mengurangkan rata-rata pixel pada area kotak putih dengan rata-rata pixel pada area kotak hitam.

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

0	0.1	0.8	1
0.3	0.1	0.7	0.8
0.1	0.2	0.8	0.8
0.2	0.2	0.8	0.8

mean(area gelap) – mean(area terang).

Sum([0.8, 1, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8]) = 6.5

Sum([0, 0.1, 0.3, 0.1, 0.1, 0.2, 0.2, 0.2]) = 1.2

Mean = 6.5 / 8 = 0.8125

Mean = 1.2 / 8 = 0.15

Delta = 0.8125 – 0.15 = 0.6625

Nilai delta tersebut kemudian di threshold sesuai kebutuhan kita. Semakin tinggi nilai delta (==1) maka akan semakin cocok fitur tersebut sebagai edge. Jika dimisalkan diberi nilai Threshold = 0.5, sedangkan nilai delta adalah 0.6625, maka fitur tersebut dianggap sebagai edge.

Menghitung nilai delta untuk keseluruhan citra akan membuat komputasi menjadi sangat mahal. Algoritma Viola-Jones mengajukan metode integral image.

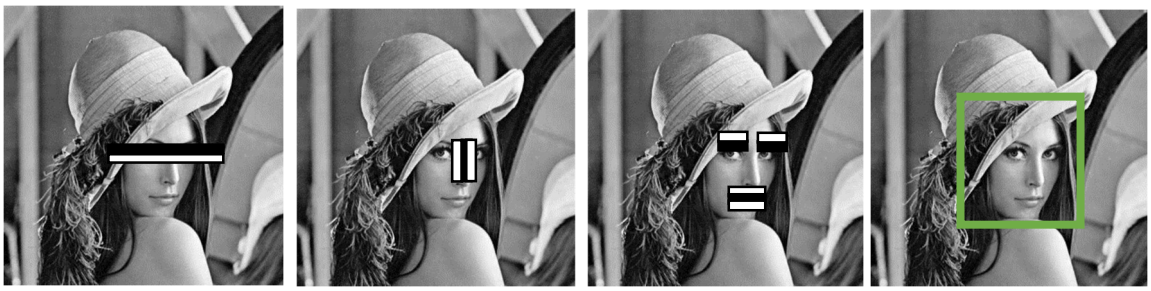
31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

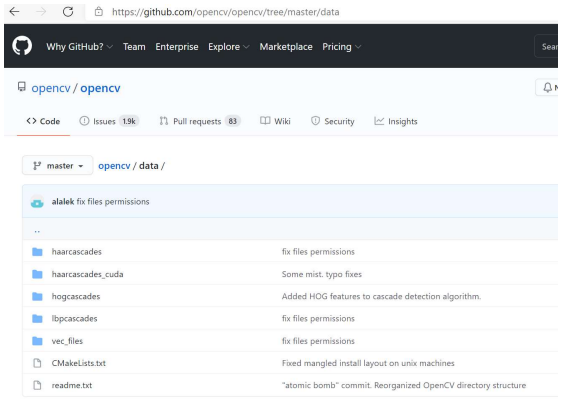
31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

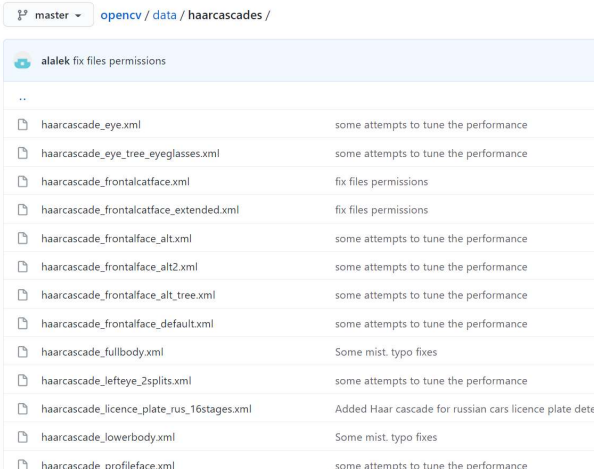
15 + 16 + 14 + 28 + 27 + 11 =
101 + 450 – 254 – 186 = 111

Algoritma ini juga dijalankan dengan cepat dengan menggunakan **cascade classifier**. **Cascade classifier** adalah menghitung nilai fitur image secara seri berdasarkan pada fitur sederhana yang telah ditunjukkan sebelumnya. Misalkan dilakukan ekstraksi fitur edge terlebih dahulu, dan jika tidak termasuk sebagai fitur edge, maka fitur line dan rectangle tidak dilakukan dan bergeser ke pencarian berikutnya.

Langkah awal yang dilakukan adalah dengan mengubah citra masukan ke dalam Grayscale. Fitur awal yang digunakan sebagai pencarian adalah edge feature yang menunjukkan adanya mata dan pipi. Jika diimage tidak menunjukkan adanya fitur ini, dianggap bahwa tidak ada wajah pada image tersebut. Jika terindikasi ada, maka dilanjutkan dengan mencari fitur berikutnya, seperti nose bridge (bagian tengah hidung), lips (bibir), alis. Pencarian terus dilakukan menggunakan cascading, yang berarti akan terdapat ribuan fitur yang terkumpul. Secara teori, pendekatan ini dapat digunakan pada berbagai objek dan deteksi. Sebagai contoh, dapat digunakan untuk mendeteksi mata, bibir, hidung dan lain sebagainya. Atau dapat juga digunakan untuk mendeteksi objek lain seperti mobil, motor, pesawat, dan lain sebagainya. Kelemahan dari algoritma ini adalah kebutuhan akan referensi dataset yang besar untuk mendapatkan fitur yang diinginkan. Saat ini fitur-fitur deteksi objek menggunakan Viola-Jones dapat ditemukan dengan mudah. Pada OpenCV juga disediakan fitur-fitur tertentu yang sering digunakan dan dipake oleh banyak peneliti. Fitur-fitur tersebut disediakan OpenCV dengan format XML.







Percobaan berikut akan digunakan untuk melakukan face detection menggunakan OpenCV dan algoritma Viola Jones. Dokumentasi lengkap dapat dilihat pada OpenCV Documentation:

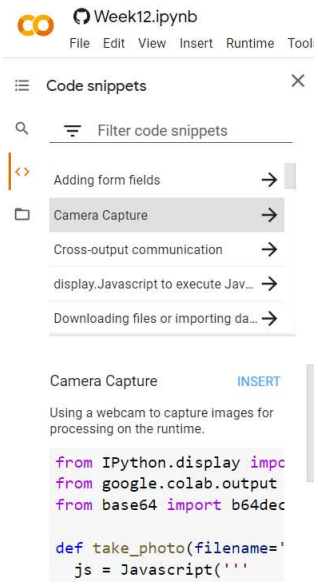
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

Beberapa images telah disediakan pada folder (/images/facedet). Untuk pretrained features juga telah disediakan pada folder (/images/haarcascades). Silahkan akses secara langsung pada folder tersebut. Fitur pretrained yang digunakan adalah "haarcascade_frontalface_alt.xml". fungsi OpenCV yang digunakan adalah menggunakan "detectMultiScale". Jika ditemukan wajah, maka akan digambarkan rectangle pada lokasi wajah tersebut menggunakan "rectangle".

```
cascade_wajah = cv.CascadeClassifier('/content/drive/MyDrive/Polinem
a/Kuliah/PCVK/Images/haarcascades/haarcascade_frontalface_alt.xml')


jokowi = cv.imread('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Imag
es/facedet/jokowi.jpg',0)
roi_wajah = cascade_wajah.detectMultiScale(jokowi)
for(x,y,w,h) in roi_wajah:
  cv.rectangle(jokowi,(x,y),(x+w,y+h),(255,255,255),3)
plt.figure(figsize = (10,10))
plt.imshow(jokowi, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f8acd8c6f50>
```

Face Tracking menggunakan live camera sebenarnya juga mudah sekali dilakukan pada Python dengan OpenCV, sayang sekali untuk Google Colab tidak bisa secara langsung menggunakan Camera yang ada pada komputer personal atau laptop, sehingga dibutuhkan code JavaScript untuk dapat mengakses kamera. code snippet untuk akses kamera telah tersedia juga pada Menu kiri Google Colab.



D. TUGAS PRAKTIKUM

Catatan: Untuk gambar pada praktikum ini menggunakan gambar pada link berikut:

https://drive.google.com/drive/folders/1d4U8FVnQ0Hq_K1Sy4XJvQsgq12ZjvmgK?usp=sharing

1. Buka https://colab.research.google.com/. Setelah dipastikan bahwa google Colab terhubung dengan Github Anda, buat notebook baru dan beri nama “Week12.ipynb”. Kemudian import beberapa library dan akses folder yang ada di Drive Anda dengan seperti biasa.

2. Lakukan Face Detection untuk image object lain yang tersedia pada (/images/facedet). Tampilkan seperti pada contoh berikut.







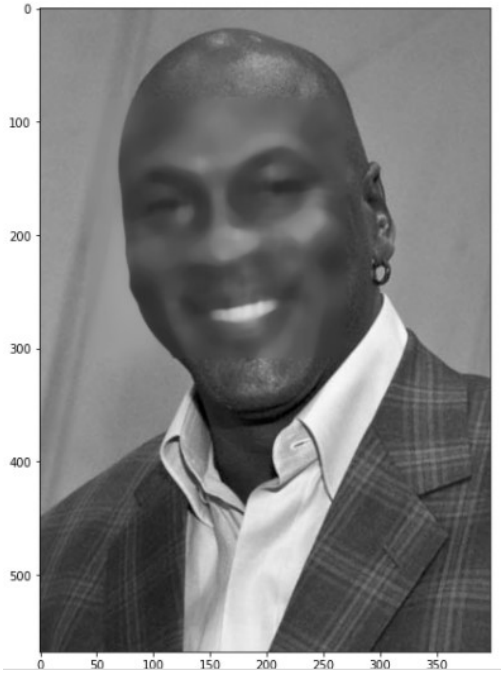




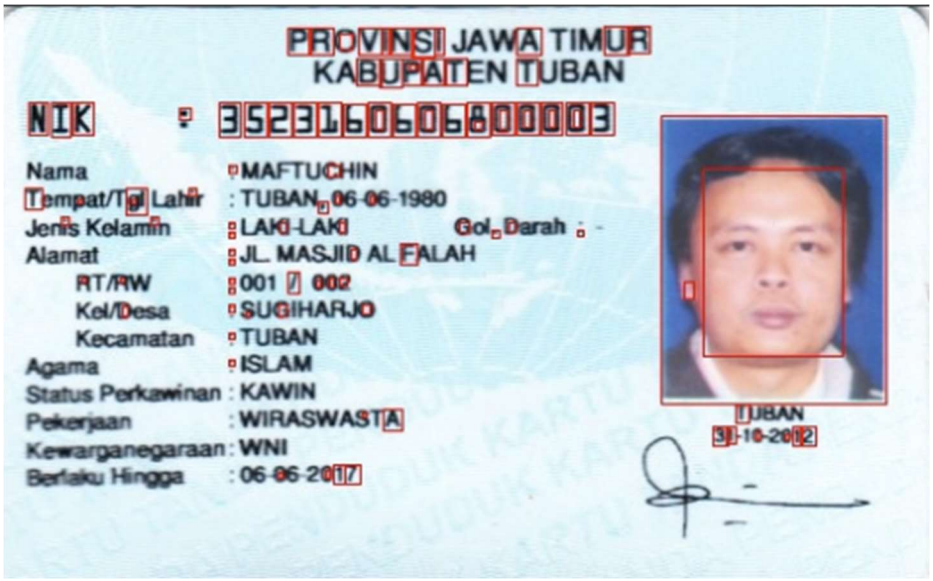

Perhatikan pada hasil face detection diatas. Secara keseluruhan, face detection dapat dilakukan dengan baik, bahkan untuk image berupa gambar bukan foto, wajah bermasker, atau wajah yang berukuran kecil (solvay).

3. Pada Soal No.2. wajah kucing tidak bisa dideteksi dengan baik. Lakukan deteksi wajah kucing hingga muncul rectangle pada bagian wajahnya. Petunjuk pada soal ini, perhatikan pretrained features yang telah disediakan OpenCV. Gunakan xml yang ada jika memang telah disediakan. Jika belum ada, coba cari dengan searching melalui search engines.
4. Cobakan juga untuk eyes detection.

5. Lakukan Face Tracking menggunakan Google Colab. Petunjuk, Tutorial selengkapnya tentang akses kamera dan FaceDetection pada google colab dapat dilihat di link berikut: https://www.youtube.com/watch?v=YjWh7QvVH60
6. Lakukan Blurring pada bagian wajah yang terdeteksi. Berikut contoh keluarannya. Petunjuk: anda dapat menggunakan cv.medianBlur untuk melakukan Blurring



7. Lakukan segmentasi karakter pada KTP seperti gambar berikut



8. Lakukan cropping pada bagian NIK saja sehingga muncul hasil sebagai berikut

9. Kita akan menggunakan Deep Learning untuk melakukan pengenalan karakter Sebelum melakukan training data, terlebih dahulu di siapkan data yang akan dilakukan training yaitu data image nomor angka 0-9. Untuk membuat data tersebut bisa gunakan image editor untuk dilakukan croping satu persatu. Kemudian selanjutnya gunakan source code di bawah ini untuk proses persiapan untuk training.

```
import os
import tqdm
import cv2
import random
import numpy as np
import pickle

# Direktori data training
DATADIR = "dataset/training"
dirs = []

training_data = []
width, height = 100, 100

# Looping direktori data training untuk diambil nama karakternya
for char_name in sorted(os.listdir(DATADIR)):
    dirs.append(char_name)

# Looping semua image data training untuk diubah menjadi array
for char_name in dirs:
    path = os.path.join(DATADIR, char_name)
    class_number = dirs.index(char_name)
    for img in tqdm(os.listdir(path)):
        try:
            img_array = cv2.imread(os.path.join(data_dir_testing, car,
char_image), cv2.IMREAD_ANYCOLOR)
            new_array = cv2.resize(img_array, (width, height))
            training_data.append([new_array, class_number])
        except Exception as e:
            pass

random.shuffle(training_data)
X = []
Y = []

for feature, label in training_data:
    X.append(feature)
    Y.append(label)

X = np.array(X).reshape(-1, width, height, 1)

# Tulis ke file pickle
pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("Y.pickle", "wb")
pickle.dump(Y, pickle_out)
pickle_out.close()
```

Penjelasan kode di atas adalah sebagai berikut ini

a. Import terlebih dahulu beberapa yang paket-paket yang dibutuhkan, ada beberapa paket yang baru misalkan tqdm digunakan untuk meload data diikuti dengan progress bar, numpy merupakan sebuah paket yang digunakan untuk melakukan operasi-operasi matriks atau array serta pickle adalah depedensi untuk menyimpan file untuk model data training.

b. Looping data training yang kelak digunakan untuk melabeli hasil pengenalan. Looping semua file training untuk diubah ke dalam sebuah image array.
c. Variabel X dan variabel Y digunakan untuk menyimpan label dan feature, label berisi karakter A-Z dan 0-9 sedangkan feture berisi data image array masing-masing label tersebut.
d. Terakhir tulis isi variabel X dan variabel Y ke dalam sebuah file pickle. File pickle tersebut nanti akan diload ketika proses training data.

10. Lakukan Training Data

Code berikut digunakan untuk melakukan training data

```
import pickle
from keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense,
ZeroPadding2D
from keras.models import Model
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical

# Load file pickle
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_in = open("Y.pickle", "rb")
Y = pickle.load(pickle_in)

Y = to_categorical(Y)
X = X / 255.0
width, height = 100, 100

# Input layer
inputs = Input(shape=(width, height, 1))
conv_layer = ZeroPadding2D(padding=(2, 2))(inputs)
conv_layer = Conv2D(16, (5, 5), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = MaxPooling2D((2, 2))(conv_layer)
conv_layer = Conv2D(32, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = Conv2D(32, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = MaxPooling2D((2, 2))(conv_layer)
conv_layer = Conv2D(64, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)

flaten = Flatten()(conv_layer)

fc_layer = Dense(256, activation='relu')(flaten)
fc_layer = Dense(64, activation='relu')(fc_layer)

# Output layer
outputs = Dense(34, activation='softmax')(fc_layer)

adam = Adam(lr=0.0001)
model = Model(inputs=inputs, outputs=outputs)
model.compile(optimizer=adam, loss='categorical_crossentropy',
metrics=['accuracy'])

model.fit(X, Y, epochs=20, verbose=1)

model.save('anpr.model')
```

Dengan kode di atas kita akan membuat sebuah model, kode di atas menggunakan deep learning dengan arsitektur CNN(Convolutional Neural Network). Sebenarnya yang saya ketahui perbedaan mendasar arsitektur ini dengan neural network biasa adalah masalah feature extraction, feature merupakan sebuah ciri yang khas yang membedakan antara objek satu dengan objek lainnya. Sebagai contoh karakter A dan karakter B pada pelat kendaraan tentunya memiliki ciri dengan bentuk yang berbeda. Feature extraction yang dimiliki CNN(Convolutional Neural Network) sudah

disediakan, kita tinggal mengutak-utik arsitekturnya/parameter yang ada di dalamnya. Sedangkan neural network biasa kita harus mencari sendiri atau dilakukan secara manual untuk mencari feature tersebut. Di bawah ini adalah bagian kode yang digunakan untuk membuat atau membangun sebuah feature tersebut.

```
conv_layer = ZeroPadding2D(padding=(2, 2))(inputs)
conv_layer = Conv2D(16, (5, 5), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = MaxPooling2D((2, 2))(conv_layer)
conv_layer = Conv2D(32, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = Conv2D(32, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)
conv_layer = MaxPooling2D((2, 2))(conv_layer)
conv_layer = Conv2D(64, (3, 3), strides=(1, 1),
activation='relu')(conv_layer)
```

11. Testing Data

Contoh kode berikut digunakan untuk melakukan testing pengenalan karakter

```
import os
import cv2
import tensorflow as tf
import numpy as np

data_dir_training = "dataset/training-bak"
data_dir_testing = "dataset/testing"
dirs = []
width, height = 100, 100

model = tf.keras.models.load_model("anpr.model")

for char_name in sorted(os.listdir(data_dir_training)):
    dirs.append(char_name)

for car in sorted(os.listdir(data_dir_testing)):
    temp = ""
    for char_img in sorted(os.listdir(os.path.join(data_dir_testing,
car))):
        img_array = cv2.imread(os.path.join(data_dir_testing, car,
char_img), cv2.IMREAD_ANYCOLOR)
        new_array = cv2.resize(img_array, (width, height))
        new_array = np.array(new_array).reshape(-1, width, height,
1)

        new_array = new_array / 255.0

        prediction = model.predict(new_array)
        temp += dirs[np.argmax(prediction[0])]

    print("folder name: {} no: {}".format(car, temp))
```

Tugas

Silakan melakukan pengenalan NIK pada e-KTP yang terdapat pada tugas modul 11!

--- SELAMAT BELAJAR ---