**LAPORAN TUGAS KELOMPOK KE-2**

**EI7007 PEMBELAJARAN MESIN LANJUT**

Oleh:

Meza Silvana (NIM : 33221017)

Dani Ramdani (NIM: 33221049)

Dosen Pengampu : Prof. Dwi Hendratmo Widyantoro, Ph.D



**INSTITUT TEKNOLOGI BANDUNG**

**2022**

**1.** **PENDAHULUAN**

***Face Recognition***

Pengenalan wajah (face recognition) merupakan salah satu cara system dapat mengenali/membedakan seseorang dengan lainnya. Wajah merupakan representasi dari seseorang. Salah satu framework yang sedang berkembang belakangan ini adalah deepface. Deepface merupakan framework untuk pengenalan wajah yang dikembangkan dengan tetap memungkinkan tetap model-model sebelumnya seperti VGG-face, Google FaceNet, OpenFace, Facebook Deepface, Deep ArcFace dan Dlib.

Referensi wajah yang digunakan adalah database pas foto (area citra yang didominasi oleh area wajah) dari seluruh peserta kelas Pembelajaran Mesin Lanjut, sedangkan untuk test data, digunakan foto anggota kelas (bukan foto yang sama) dan foto bukan anggota kelas. Pada test data akan diberikan label sesuai antara nama dan wajah. Test data berisi 50% anggota kelas dan 50% non anggota kelas.

Buku 2011 tentang pengenalan wajah berjudul *"Handbook of Face Recognition"* menjelaskan dua mode utama untuk pengenalan wajah, sebagai berikut:

· Verifikasi Wajah. Pemetaan satu-ke-satu dari wajah yang diberikan terhadap identitas yang diketahui (misalnya apakah ini orangnya?).

· ID Wajah. Pemetaan satu-ke-banyak untuk wajah tertentu terhadap database wajah yang dikenal (misalnya siapa orang ini?).

***FaceNet* Model**

*FaceNet* adalah sistem pengenalan wajah yang dijelaskan oleh Florian Schroff, et al. di Google dalam makalah 2015 yang berjudul *“*[*FaceNet: A Unified Embedding for Face Recognition and Clustering*](https://arxiv.org/abs/1503.03832)*”*. *FaceNet* dikembangkan oleh para peneliti di Google yang mencapai hasil mutakhir pada berbagai setdata *benchmark* pengenalan wajah. Sistem *FaceNet* dapat digunakan secara luas berkat beberapa implementasi model sumber terbuka pihak ketiga dan ketersediaan model yang telah dilatih sebelumnya. Sistem *FaceNet* dapat digunakan untuk mengekstrak fitur berkualitas tinggi dari wajah, yang disebut penyematan wajah, yang kemudian dapat digunakan untuk melatih sistem identifikasi wajah.

*FaceNet* model merupakan *deep convolutional neural network* yang dilatih melalui fungsi *triplet loss* yang mendorong vektor untuk identitas yang sama menjadi lebih mirip (jarak lebih kecil), sedangkan vektor untuk identitas berbeda diharapkan menjadi kurang mirip (jarak lebih besar). Fokus pada pelatihan model untuk membuat *embeddings* secara langsung (daripada mengekstraknya dari lapisan perantara model) merupakan inovasi penting dalam model ini.

***Multi-Task Cascaded Convolutional Neural Network (MTCNN)***

MTCNN digunakan pada proses *face recognition* sebagai *face detection*. *Face detection* adalah proses mencari lokasi wajah secara otomatis dalam sebuah foto dan melokalisasinya dengan menggambar kotak pembatas di sekitar luasan wajahnya.

Rincian tugas:

· Menggunakan fungsi-fungsi yang disediakan library Deepface, buat prosedur (program) untuk dapat mengenali wajah image yang ada di Test Data.

· Melakukan eksplorasi untuk mendapatkan kombinasi model pengenalan wajah dan deteksi wajah dengan kinerja terbaik.

· Ukuran kinerja yang dipakai adalah akurasi, berdasarkan kemampuannya untuk mengenali wajah-wajah yang ada di Test Data.

· Pengenalan juga dihitung benar jika wajah yang tidak ada di referensi utama dikategorikan sebagai ‘unknown’.

· Selain menampilkan akurasi di bagian akhir, untuk setiap foto yang ada di Test Data, tampilkan juga hasil prediksinya dengan format sebagai berikut:

· Jika sistem berhasil mengenali wajah: <Foto+Label dari Test Data> <Foto+Label dari Database pas foto> Benar/Salah

· Jika sistem tidak berhasil mengenali wajah: <Foto+Label Test Data> Unknown Benar/Salah

· Benar/Salah pada format di atas ditulis berdasarkan penilaian hasil prediksi.

Untuk tugas ini maka dibentuk desain model deep learning dengan beberapa kebutuhan yaitu membangun jenis model klasifikasi dengan model yang dipilih adalah google FaceNet. Kemudian menggunakan dataset dari 2 jenis dataset yaitu dataset citra anggota kelas dan non anggota kelas dan menjalankan coding environtmentnya menggunakan google colab.

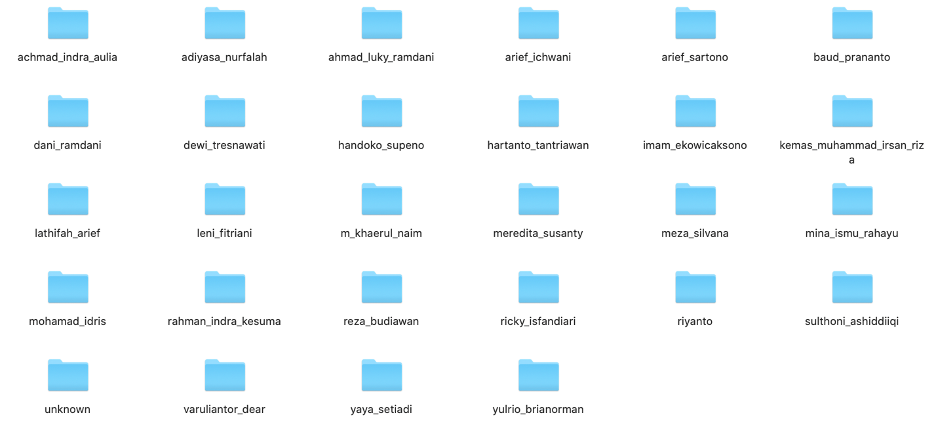
**2.** **LINK TUGAS**

URL GitHub :<https://github.com/dhanirmd1983/Facenet-FR>

**3.** **PROSEDUR PENGENALAN WAJAH DENGAN FACENET**

***Dataset***

Dataset yang digunakan terdiri dari foto wajah pengampu matakuliah Pemebelajaran Mesin Lanjut yang dibagi menjadi 2 folder yaitu data untuk train serta data untuk validasi.



***Alur Program***

Ada 5 langkah yang dilakukan untuk melakukan proses pengenalan wajah menggunakan deepface FaceNet yaitu :

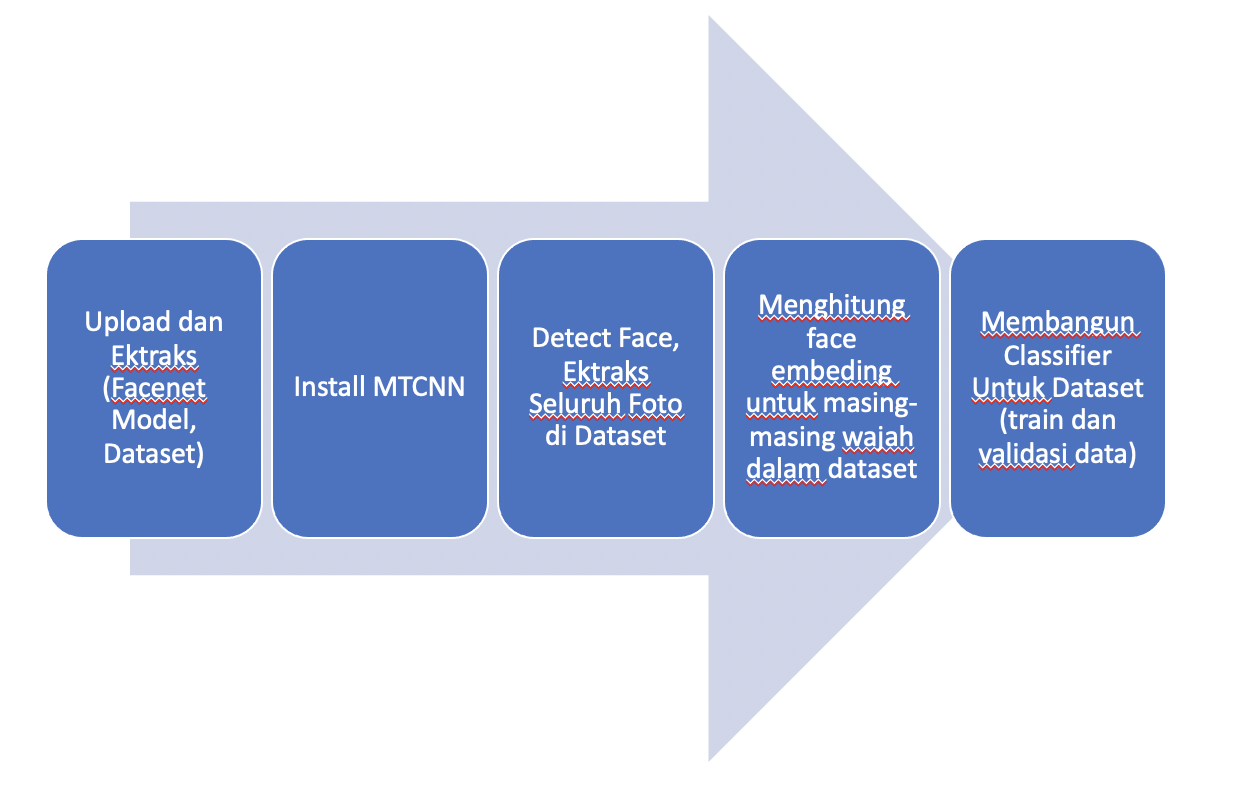
Langkah 1 : melakukan Ekstraksi facenet\_keras.h5 dan Dataset Foto Rekan Mahasiswa S3 Pembelajaran Mesin Lanjut

Langkah 2 : Instalasi MTCNN

Langkah 3 : Ekstrak Masing-masing Foto Dalam Dataset

Langkah 4 : Menghitung face embeding untuk masing-masing wajah dalam dataset

Langkah 5 : Membangun Classifier Untuk Dataset



Untuk penjelasan langkah tersebut dapat diuraikan dengan cuplikan list programming sebagai berikut:

Langkah 1 : Ekstrak facenet\_keras.h5 dan Dataset Foto Rekan Mahasiswa S3 Pembelajaran Mesin Lanjut

from zipfile import ZipFile

file\_name = "datakelas.zip"

with ZipFile(file\_name, 'r') as zip:

zip.extractall()

/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anymore to use a file ID.

category=FutureWarning,

Langkah 2 : Instalasi MTCNN

1. In [8]:

!pip install mtcnn

Langkah 3 : Ekstrak Masing-masing Foto Dalam Dataset

2. In [9]:

def extract\_face(filename, required\_size=(160, 160)):

image = Image.open(filename)

image = image.convert('RGB')

pixels = asarray(image)

detector = MTCNN()

results = detector.detect\_faces(pixels)

*# mengekstrak data face*

face = pixels[y1:y2, x1:x2]

*# me-load seluruh data pada direktori*

def load\_faces(directory):

faces = list()

*# me-load dataset pada tiap-tiap kelas*

def load\_dataset(directory):

X, y = list(), list()

*# me-load train dataset*

trainX, trainy = load\_dataset('/content/datakelas/train/')

print(trainX.shape, trainy.shape)

*# me-load test dataset*

testX, testy = load\_dataset('/content/datakelas/val/')

*# mensave data pada file terkompres*

*#savez\_compressed('kelas-dataset.npz', trainX, trainy, testX, testy)*

savez\_compressed('kelasPML-dataset.npz', trainX, trainy, testX, testy)

Ini adalah salah satu hasil loading dataset anggota kelas yang terdiri dari:

>loaded 1 examples for class: yulrio\_brianorman

>loaded 1 examples for class: mohamad\_idris

>loaded 1 examples for class: reza\_budiawan

>loaded 1 examples for class: leni\_fitriani

>loaded 7 examples for class: dani\_ramdani

Langkah 4 : Menghitung face embeding untuk masing-masing wajah dalam dataset

def get\_embedding(model, face\_pixels):

*# melakukan scalling pixel values*

face\_pixels = face\_pixels.astype('float32')

*# menyamakan nilai pixel*

mean, std = face\_pixels.mean(), face\_pixels.std()

face\_pixels = (face\_pixels - mean) / std

*# melakukan tranformasi pada satu sampel*

samples = expand\_dims(face\_pixels, axis=0)

*# membuat prediksi untuk mendapatkan nilai embedding*

yhat = model.predict(samples)

return yhat[0]

*# me-load face dataset*

data = load('kelasPML-dataset.npz')

trainX, trainy, testX, testy = data['arr\_0'], data['arr\_1'], data['arr\_2'], data['arr\_3']

print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)

*# me-load facenet model*

model = load\_model('facenet\_keras.h5')

print('Loaded Model')

*# mengkonversi tiap wajah pada train embedded*

newTrainX = list()

for face\_pixels in trainX:

embedding = get\_embedding(model, face\_pixels)

newTrainX.append(embedding)

newTrainX = asarray(newTrainX)

*# mengkonversi tiap wajah pada test embedded*

newTestX = list()

for face\_pixels in testX:

embedding = get\_embedding(model, face\_pixels)

newTestX.append(embedding)

newTestX = asarray(newTestX)

*# men-save hasil dalam file terkompres*

savez\_compressed('kelasPML-embeddings.npz', newTrainX, trainy,newTestX, testy)

Didapatkan hasil :

Loaded: (86, 160, 160, 3) (86,) (28, 160, 160, 3) (28,)

Loaded Model

(86, 128)

(28, 128)

Langkah 5 : Membangun Classifier Untuk Dataset

Proses Berikut aadalah Langkah yang digunakan untuk melakukan klasifikasi

*# me-load data face*

data = load('kelasPML-dataset.npz')

testX\_faces = data['arr\_2']

*# me-load face embeddings*

data = load('kelasPML-embeddings.npz')

trainX, trainy, testX, testy = data['arr\_0'], data['arr\_1'], data['arr\_2'], data['arr\_3']

*# me-normalize input vector*

in\_encoder = Normalizer(norm='l2')

trainX = in\_encoder.transform(trainX)

testX = in\_encoder.transform(testX)

*# memberikan label target*

out\_encoder = LabelEncoder()

out\_encoder.fit(trainy)

trainy = out\_encoder.transform(trainy)

testy = out\_encoder.transform(testy)

*# melakukan fit model*

model = SVC(kernel='linear', probability=True)

model.fit(trainX, trainy)

*# test model secara random dari dataset yang ada*

*#selection = choice([i for i in range(testX.shape[0])])*

*#memilih gambar untuk pengetesan*

for selection in range (testX.shape[0]) :

random\_face\_pixels = testX\_faces[selection]

random\_face\_emb = testX[selection]

random\_face\_class = testy[selection]

random\_face\_name = out\_encoder.inverse\_transform([random\_face\_class])

*# melakukan prediksi*

samples = expand\_dims(random\_face\_emb, axis=0)

yhat\_class = model.predict(samples)

yhat\_prob = model.predict\_proba(samples)

*# membuat nama*

class\_index = yhat\_class[0]

class\_probability = yhat\_prob[0,class\_index] \* 100

predict\_names = out\_encoder.inverse\_transform(yhat\_class)

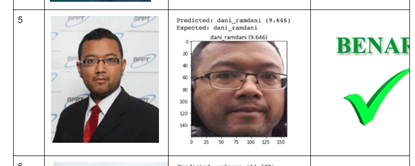
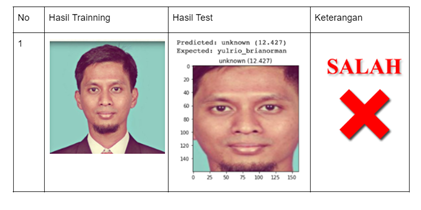
print('Predicted: %s (%.3f)' % (predict\_names[0], class\_probability))

print('Expected: %s' % random\_face\_name[0])

**4.** **HASIL PREDIKSI EKSPLORASI**

**a.** **Hasil pengetesan**

Pada percobaan eksplorasi ini didapatkan hasil sebagai contoh gambar berikut:

Ada dua hasil yang didapatkan yaitu benar atau salah, benar jika hasil ekspektasi sesuai dengan prediksi, dan salah jika tidak sesuai.

**b.** **Hasil prediksi eksplorasi**

Untuk testing , dilakukan 27 kali pengetesan pada dataset, dan mendapatkan hasil benar sebanyak 12 kali. Sehingga persentase akurasinya adalah 44,44% . (hasil dapat dilihat pada lampiran)

**c.** **Hasil kinerja**

Berdasarkan hasil percobaan, nilai akurasi masih berada di bawah 50%. Nilai ini dianggap masih rendah sehingga dibutuhkan perubahan arsitektur model ataupun penambahan proses training terhadap dataset.

**5.** **ANALISIS DAN KESIMPULAN**

Hasil percobaan yang dilakukan pada pelatihan model masih memberikan hasil yang kurang memuaskan. Beberapa kemungkinan yang terjadi adalah kurangnya data pelatihan dan jumlah data training. Kemudian Perlu adanya beberapa kali perubahan terhadap arsitektur model pada pelatihan ataupun pengetesan sehingga didapatkan model yang benar-benar memberikan hasil baik. Kemungkinan lain dapat juga memperbaiki dataset yang ada. Secara umum, sistem telah dapat memberikan hasil dengan dua kemungkinan, yaitu kelompok benar dan kelompok salah.

**6.** **REFERENSI**

1. G. Schroff, D. Kalenichenko, and J. Philb, “FaceNet: A Uniﬁed Embedding for Face Recognition and Clustering,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823

2. Y. Taigman, M. Yang, M.A. Ranzato, L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

**7.** **LAMPIRAN**

| No | Hasil Trainning | Hasil Test | Keterangan |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 |  |  |  |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 |  |  |  |
| 20 |  |  |  |
| 21 |  |  |  |
| 22 |  |  |  |
| 23 |  |  |  |
| 24 |  |  |  |
| 25 |  |  |  |
| 26 |  |  |  |
| 27 |  |  |  |