SISTEM DETEKSI MULTI WAJAH SECARA REALTIME MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

by Erwin Erwin

Submission date: 26-Feb-2020 09:24AM (UTC+0700)
Submission ID: 1264274552
File name: manual_book.pdf (1.15M)
Word count: 3575
Character count: 22031

BUKU MANUAL

SISTEM DETEKSI MULTI WAJAH SECARA REALTIME MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

OLEH:

Dr. Erwin, S.Si., M.Si
Prof. Drs. Saparudin, M. T, Ph.D
Reza Firsandaya Malik, S.T., M.T.,Ph.D.
M. Fachrurrozi, S.Si.,M.T.

FAKULTAS ILMU KOMPUTER
UNIVERSITAS SRIWIJAYA

2019

Latar Belakang

Pengenalan objek biometrik salah satunya adalah wajah dapat diterapkan untuk berbagai keperluan seperti akses kontrol, keamanan bandara, dan sebagainya [1]. Pengenalan wajah dapat juga diterapkan pada bidang pendidikan yaitu sistem absensi otomatis yang menerapkan pengenalan wajah untuk mengenali peserta belajar dalam kelas. Penerapan pengenalan wajah pada sistem absensi otomatis bertujuan untuk memverifikasi apakah peserta belajar berada di dalam kelas, hal ini dinilai lebih aman dibandingkan menggunakan cara penandatanganan daftar hadir yang memungkinkan peserta belajar melakukan kecurangan dalam mengisi daftar hadir. Contoh lainnya dalam penerapan pengenalan wajah adalah untuk melakukan pencatatan secara otomatis terhadap siapa saja yang berada pada suatu ruangan dengan tujuan meningkatkan keamanan.

Metode dalam melakukan implementasi pengenalan wajah telah dilakukan pada beberapa penelitian antara lain pengenalan wajah menggunakan *Laplacianface* [2], pengenalan wajah menggunakan *Support Vector Machine* [3], menggunakan gabungan *Principal Component Analysis* dan *Backpropagation* [4], dan menggunakan *Convolutional Neural Network* [5]. Implementasi yang menunjukan performa paling baik dalam akurasi adalah menggunakan *Convolutional Neural Network* yaitu diatas 90% [5].

Program komputer pengenalan wajah secara realtime yang dibangun ini menggunakan implementasi dari metode *Convolutional Neural Network.* Program komputer pengenalan wajah ini dapat mengenali wajah secara *realtime*, dengan

artian masukan *(input)* citra melalui kamera dapat langsung diproses oleh sistem menghasilkan keluaran (*output*) berupa identitas dari wajah yang berhasil dikenali.

Deskripsi Program Komputer

Program komputer pengenalan wajah secara *realtime* menggunakan *convolutional neural network* merupakan program komputer berbasis web yang dibangun menggunakan bahasa pemrograman *Python* dan *Django Web Framework*. Tahapan pengembangan perangkat lunak meliputi fase pengumpulan kebutuhan sistem (*system requirements*), dan fase konstruksi yaitu implementasi kode program dan implementasi antar muka program.

a. Kebutuhan Sistem (*System Requirements*)

Suatu program komputer memiliki kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional merupakan kebutuhan yang mencakup keseluruhan proses maupun layanan yang akan ada dalam suatu sistem, mencakup bagaimana sistem harus bereaksi pada input tertentu dan bagaimana perilaku sistem pada situasi tertentu. Sedangkan kebutuhan non-fungsional adalah kebutuhan tambahan yang mendukung kinerja perangkat lunak yang berfokus kepada properti perilaku yang ada pada sistem. Tabel 1 menjelaskan mengenai kebutuhan fungsional dari perangkat lunak. Dilanjutkan pada tabel 2 menjelaskan mengenai kebutuhan non-fungsional dari perangkat lunak.

Tabel 1. Kebutuhan Fungsional

ID	Kebutuhan Fungsional	Penjelasan
KF-01	Sistem dapat menerima masukan berupa citra wajah dari kamera	User dapat memasukkan data berupa gambar wajah
KF-04	Sistem dapat melakukan proses pelatihan untuk membuat model pengenalan wajah	User dapat melakukan training untuk membuat model yang dapat mengenali wajah.
KF-05	Sistem dapat melakukan Pengenalan Wajah	User dapat melakukan proses pengenalan pada citra wajah yang telah dimasukkan.

Tabel 2. Kebutuhan Non Fungsional

ID	Kebutuhan Non Fungsional	Penjelasan
KNF-01	*Availability*	Perangkat lunak dapat beroperasi selama 24 jam
KNF-02	*Reliability*	Perangkat lunak bersifat *offline*
KNF-03	Ergonomi	Tampilan perangkat lunak dibuat *user friendly*
KNF-04	*Portability*	Dijalankan menggunakan sistem operasi *Windows* dan *Linux*
KNF-05	*Memory*	-
KNF-06	*Response time*	Dapat melakukan pengenalan wajah dalam waktu ≤ 1 detik

b. Konstruksi

Fase konstruksi berfokus pada implementasi desain perangkat lunak sesuai hasil anaisis kebutuhan. Hasil yang akan diperoleh pada proses ini ialah perangkat lunak yang siap digunakan sebagai alat penelitian oleh end user. Tahapan yang dilakukan pada fase konstruksi meliputi implementasi kode program dan

implementasi antar muka pengguna. Berikut ini merupakan deskripsi mengenai implementasi kode program yang telah dibangun, deskripsi dijelaskan dalam setiap file kode program.

1. simple_cnn.py

File *simple_cnn.py* berfungsi untuk membangun arsitektur convolutional neural network sederhana.

simple_cnn.py

```
1   from keras import (layers, backend, models, utils)
2
3   def build(input_shape,num_classes):
4       model = models.Sequential()
5       model.add(layers.Conv2D(32, kernel_size=(3, 3),
6                               activation='relu',
7                               input_shape=input_shape))
8       model.add(layers.Conv2D(64, (3, 3), activation='relu'))
9       model.add(layers.MaxPooling2D(pool_size=(2, 2)))
10      model.add(layers.Dropout(0.25))
11      model.add(layers.Flatten())
12      model.add(layers.Dense(128, activation='relu'))
13      model.add(layers.Dropout(0.5))
14      model.add(layers.Dense(num_classes, activation='softmax'))
15      return model
```

Gambar 1. Baris kode pada file *simple_cnn.py*

Baris 1 pada file simple_cnn.py melakukan import modul dari pustaka *(library*) yang dibutuhkan yaitu modul *layers, backend, models,* dan *utils* dari pustaka *keras*. Selanjutnya pada baris 3 sampai 15 adalah sebuah *method* bernama *build*. Method *build* berfungsi membangun struktur layer dari *convolutional neural network*. Struktur layer yang dibangun dapat dilihat pada tabel berikut:

Tabel 3. Arsitektur *Simple CNN*

No.	Layer	Output Size	Filter	Kernel /Pool Size	Activation
1	Convolution	94 x 94 x 32	32	3 x 3	ReLu
2	Convolution	92 x 92 x 64	64	3 x 3	ReLu
3	Pooling (Max Pooling)	46 x 46 x 64	-	2 x 2	-
4	Flatten	1 x 135.454	-	-	-
5	Fully Connected	1 x 128	-	-	ReLu
6	Fully Connected (prediction/ output layer)	1 x jumlah kelas	-	-	SoftMax

2. face_recognition.py

File face_recognition.py berfungsi untuk melakukan proses pelatihan dan membangun model *convolutional neural network*. Pada baris 1 sampai 14 dilakukan import modul dari *library* yang dibutuhkan *library* yang dibutuhkan pada proses pelatihan adalah *keras, split_folders, pyprind, numpy, pickle, argparse, dan os*. Pada baris 1 sampai 14 ini juga dilakukan import dari modul yang telah dibuat sebelumnya yaitu file simple_cnn (baris ke-9).

face_recognition.py

```
1   from keras.applications.vgg16 import VGG16
2   import split_folders
3   from keras import layers, models
4   from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
5   import numpy as np
6   from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
7   from keras.optimizers import SGD
8   from keras.regularizers import l2
9   import simple_cnn
10
11  import argparse
12  import pickle
13  import os
14  from pyprind import  ProgBar
```

Gambar 2. Baris kode 1 sampai 14 pada file *face_recognition.py*

Baris selanjutnya adalah baris kode ke-16 sampe baris ke-44. Pada baris ini terdapat tiga *method* yaitu *create_label_encoder*, *create_model_vgg16*, dan *create_simple_model.*

face_recognition.py

```
16  def create_label_encoder():
17      images_path = 'train_data/images'
18      cls_name = []
19      for i in sorted(os.listdir(images_path)):
20          cls_name.append(i)
21
22      pickle.dump(cls_name, open('face_ai/models/label_encoder', 'wb'))
23
24      return cls_name
25
26
27  def create_model_vgg16(num_classes):
28      conv_base = VGG16(weights='imagenet',
29                        include_top=False,
30                        input_shape=(96, 96, 3))
31
32      conv_base.trainable = False
33
34      model = models.Sequential()
35      model.add(conv_base)
36      model.add(layers.Flatten())
37      model.add(layers.Dense(1024, activation='relu'))
38      model.add(layers.Dropout(0.5))
39      model.add(layers.Dense(num_classes, kernel_regularizer=l2(.0005), activation='softmax'))
40
41      return model
42
43  def create_simple_model(num_classes):
44      return simple_cnn.build((96,96,3),num_classes)
```

Gambar 3. Baris kode 16 sampai 44 pada file *face_recognition.py*

Method *create_label_encoder* berfungsi untuk melakukan pengkodean pada label citra, setiap label citra dimasukan dalam *python array* lalu kode setiap label adalah index dari array itu sendiri. Kemudian menggunakan library pickle, array python disimpan ke dalam bentuk binary sehingga dapat dimuat ketika akan digunakan untuk melakukan pengenalan wajah. Method *create_model_vgg16* dan *create_simple_model* berguna untuk membangun arsitektur *convolutional neural network.* Method *create_model_vgg16* untuk membangun arsitektur *VGG16* sedangkan *create_simple_model* untuk membangun arsitektur yang lebih sederhana

yaitu dapat dilihat pada file simple_cnn.py, *create_simple_model* memanggil *method* build dari modul *simple_cnn*.

Baris selanjutnya pada file *face_recognition.py* adalah baris ke-47 sampai baris ke-73. Baris ini merupakan *method* yang bernama *create_data_generator*. Berfungsi sebagai objek yang memberikan data citra yang telah dibaca dari file ke dalam proses pelatihan, objek yang dibuat oleh method ini adalah *train_data* dan *validation_data*. Objek train_data memberikan data pada proses pelatihan untuk dilakukan *learning* sedangkan objek *validation_data* berfungsi memberikan data validasi pada model *convolutional neural network*, hal ini bertujuan untuk mengukur performa *learning process* oleh model.

face_recognition.py

```
47  def create_data_generator(train_dir, test_dir, batch_size):
48
49      data_gen = ImageDataGenerator(rotation_range=0,
50                                    featurewise_center=False,
51                                    featurewise_std_normalization=False,
52                                    horizontal_flip=True,
53                                    vertical_flip=False,
54                                    data_format='channels_last',
55                                    rescale=1./255
56                                    )
57
58      train_data = data_gen.flow_from_directory(
59          directory=train_dir,
60          target_size=(96, 96),
61          color_mode='rgb',
62          batch_size=batch_size,
63          seed=11
64      )
65      validation_data = data_gen.flow_from_directory(
66          directory=test_dir,
67          target_size=(96, 96),
68          color_mode='rgb',
69          batch_size=batch_size,
70          seed=11
71      )
72
73      return train_data, validation_data
```

Gambar 4. Baris kode 47 sampai 73 pada file *face_recognition.py*

Baris selanjutnya pada file *face_recognition.py* adalah baris ke-76 sampai baris ke-107. Baris ini memuat *method train_model* yang berfungsi melakukan proses *learning* untuk membangungun model pengenalan wajah. Pada baris ke-78 memanggil method ratio pada library split_folders, berfungsi untuk membagi data menjadi data *training* dan data *validation*, output hasil pemisahan tersimpan dalam folder "train_data/dataset" kemudian pada baris ke-81 dilakukan pemanggilan dari *method create_data_generator* yang terdapat pada baris sebelumnya. Pada baris ke-86 sampai baris ke-96 merupakan proses mendefinisikan fungsi *callback*, fungsi *callback* merupakan fungsi yang akan dijalankan ketika awal epoch, akhir setiap epoch, dan akhir dari seluruh proses epoch. Fungsi *callback* yang digunakan adalah *CSVLogger, ReduceLROnPlateau, EarlyStopping,* dan *ModelCheckpoint. CSVLogger* merupakan *callback* yang berfungsi untuk mencatat parameter *learning* yaitu nilai akurasi proses pelatihan dan validasi setiap epoch dan nilai loss dari setiap epoch pada pelatihan dan validasi di setiap epoch. *ReduceLROnPlateau* berfungsi untuk menurunkan nilai *learning rate* dari model pada proses pelatihan jika nilai *loss* tidak berkurang sebanyak epoch yang ditentukan. *EarlyStopping* berfungsi menghentikan proses pelatihan selagi epoch masih belum berakhir jika nilai loss tidak mengalami penurunan. *ModelCheckpoint* berfungsi untuk menyimpan model pada setiap epoch. Pengaturan model terdapat pada baris ke-98 dan baris ke-99 dimana menggunakan optimizer SGD (*stochastic gradient descent*) dengan *learning rate* awal yaitu sebesar 0.01 dan menggunakan *loss function categorical crossentropy*. Pada baris ke-101 merupakan baris kode yang menjalankan fungsi *learning*.

face_recognition.py

```
76  def train_model(model, epochs, batch_size):
77      create_label_encoder()
78      split_folders.ratio('train_data/images', output="train_data/dataset",
79                          seed=1337, ratio=(.8, .2))
80
81      train_data, validation_data = create_data_generator('train_data/dataset/train',
82                                                          'train_data/dataset/val',
83                                                          batch_size)
84
85      # callbacks
86      base_path = 'train_data/'
87      patience = 50
88      log_file_path = base_path + 'logs/training.log'
89      csv_logger = CSVLogger(log_file_path, append=False)
90      reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
91                                    patience=int(patience / 4), verbose=1)
92      early_stop = EarlyStopping('val_loss', patience=patience)
93      trained_models_path = base_path + 'models/_models_'
94      model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
95      model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1, save_best_only=True)
96      callbacks = [model_checkpoint, csv_logger, reduce_lr, early_stop]
97
98      opt = SGD(lr=.01, momentum=.9)
99      model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
100
101     model.fit_generator( generator=train_data, validation_data=validation_data,
102         steps_per_epoch= 128,
103         validation_steps= 128,
104         epochs=epochs,
105         verbose=1,
106         callbacks=callbacks
107     )
```

Gambar 5. Baris kode 76 sampai 107 pada file *face_recognition.py*

Pada baris terakhir yaitu baris ke-110 sampai baris ke-135 merupakan baris kode yang berjalan ketika file *face_recognition.py* dijalankan. Pada baris ke-111 sampai baris ke-122 berfungsi untuk mendefinisikan *command line parameter* yang dapat dimasukan oleh user. Pada baris ke-124 sampai baris ke-127 adalah proses pemilihan arsitektur *convolutional neural network* berdasarkan parameter yang dimasukan oleh user. Baris ke-129 sampai baris ke-135 berfungsi untuk menjalankan pilihan berdasarkan parameter input yang dimasukan oleh user.

face_recognition.py

```
110 if __name__ == '__main__':
111     parser = argparse.ArgumentParser(description='Train face recognition model')
112     parser.add_argument('--numclasses', '-n', type=int, default=11,
113                         help='Number of classes')
114     parser.add_argument('--batchsize', '-b', type=int, default=32,
115                         help='Number of images in each mini-batch')
116     parser.add_argument('--epochs', '-e', type=int, default=1000,
117                         help='Number of sweeps over the dataset to train')
118     parser.add_argument('--model', '-m', default='vgg16',
119                         help='model selection')
120     parser.add_argument('--option', '-o', default='start_train',
121                         help='Option for models')
122     args = parser.parse_args()
123 
124     if args.model == 'vgg16':
125         model = create_model_vgg16(args.numclasses)
126     else:
127         model = model = create_simple_model(args.numclasses)
128 
129     if args.option == 'model_details':
130         model.summary()
131     elif args.option == 'start_train':
132         train_model(model, args.epochs, args.batchsize)
133     else:
134         pass
135         print(f'Error: No option name: {args.option}')
```

Gambar 6. Baris kode 110 sampai 135 pada file *face_recognition.py*

3. face_ai/utils.py

File yang terdapat pada folder "*face_ai*" yang memiliki nama *"utils.py"* berfungsi sebagai inisiator model *convolutional neural network* yang akan digunakan saat program berjalan dan juga terdapat fungsi untuk mengenali data input. Pada baris pertama sampai baris ke-7 merupakan proses *import* dari library yang dibutuhkan oleh program saat berjalan. Pada baris ke-10 sampai baris ke-18 dilakukan proses memuat model dan encoder yang telah dibuat pada saat melakukan pelatihan, pada baris ini juga diinisiasi variable *graph* yang berfungsi untuk melakukan regulasi *session graph* yang digunakan oleh tensorflow.

face_ai ▸ utils.py

```
1   from keras.models import load_model
2   import keras.backend
3   import tensorflow as tf
4
5   import numpy as np
6   import pickle
7   import time
8
9   # initializing the graph
10  graph = tf.get_default_graph()
11
12  _backend = keras.backend.backend()
13
14  # loading trained model
15  print("Model loading.......")
16  model = load_model('face_ai/models/face_ai.hdf5', compile=False)
17  label_encoder = pickle.load(open('face_ai/models/label_encoder', 'rb'))
18  print("Model loaded!!")
```

Gambar 7. Baris kode 1 sampai 18 pada file *face_ai/utils.py*

Baris kode selanjutnya adalah baris ke-21 sampai baris ke-33 dimana terdapat *method* yaitu *predict* yang berfungsi untuk mengenali data input berupa citra wajah, lalu mengembalikan variable *label*, *prediction,* dan *respond time.* Variabel *label* adalah label dari kelas yang berhasil dikenali, *prediction* merupakan vektor yang merupakan hasil output dari proses *feed-forward* pada *convolutional neural network*, dan *respond_time* merupakan jumlah waktu yang dibutuhkan model untuk mengenali data dalam satuan detik.

face_ai ▸ utils.py

```
21  def predict(img):
22      start_time = time.time()
23
24      if _backend == 'tensorflow':
25          with graph.as_default():
26              prediction = model.predict(np.expand_dims(img, axis=0))[0]
27      else:
28          prediction = model.predict(np.expand_dims(img, axis=0))[0]
29
30      respond_time = time.time() - start_time
31      label = label_encoder[np.argmax(prediction)]
32
33      return label, prediction, respond_time
```

Gambar 8. Baris kode 21 sampai 33 pada file *face_ai/utils.py*

4. media_stream/camera.py

File "*media_stream/camera.py*" berfungsi untuk menangkap citra input dari kamera. Pada baris pertama sampai baris ke-6 berfungsi untuk melakukan import *library* yang dibutuhkan yaitu OpenCV(*cv2*), numpy, dlib, imutils, time, dan modul dari kode program yang dibuat pada file "face_ai/utils.py" yang telah dijelaskan sebelumnya.

media_stream ▸ camera.py

```
1  import cv2
2  import numpy as np
3  import dlib
4  from imutils import face_utils
5  from face_ai import utils
6  import time
```

Gambar 9. Baris kode 1 sampai 6 pada file *media_stream/camera.py*

media_stream ▸ camera.py

```
10  def recognize(face, image, id):
11      start_time = time.time()
12      (fX, fY, fW, fH) = face_utils.rect_to_bb(face)
13      roi = image[fY:fY + fH, fX:fX + fW]
14      roi = cv2.resize(roi, (96, 96))
15      label, prediction, respond_time = utils.predict(roi)
16      respond_time = time.time() - start_time
17
18      acc = prediction[np.argmax(prediction)] * 100
19
20      prob = "%.2f" % acc
21      respond_time = "%.2f" % respond_time
22
23      time_format = 'Respond Time: {0}s'.format(respond_time)
24      opt = '{0}. {1}'.format(id,label)
25
26      if acc > 98 :
27          cv2.putText(image, opt, (fX, fY - 10),
28                      cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)
29
30      cv2.rectangle(image, (fX, fY), (fX + fW, fY + fH),
31                    (244, 212, 66), 1)
32
33      cv2.putText(image, time_format, (0, 10),
34                  cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)
35      return image
```

Gambar 10. Baris kode 10 sampai 35 pada file *media_stream/camera.py*

3 Selanjutnya pada baris ke-10 sampai baris ke-35 terdapat *method* bernama *recognize* yang berfungsi untuk melakukan pemanggilan method *predict* pada modul *utils* lalu menggambar *bounding box* serta menuliskan label yang didapat dari hasil pengenalan pada wajah yang terdeteksi dari citra masukan kamera.

Pada baris ke-37 sampai baris ke-62 terdapat sebuah kelas bernama *VideoCamera* yang dapat diinisiasi sebagai objek. Pada kelas *VideoCamera* terdapat kelas yaitu *__init__*, *__del__*, dan *get_frame*. Method *__init__* dan *__del__*, merupakan method *default* yang terdapat pada python object yang bertujuan untuk memberi perintah ketika objek diinisiasi dan dan ketika objek *terminated*. Method *get_frame* berfungsi untuk menangkap citra perdetik dari

kamera. Pada baris ke-50 terdapat kode program yang berfungsi untuk mendeteksi seluruh citra wajah yang terdapat pada citra input dari kamera. Selanjutnya baris ke-52 sampai baris ke-59 merupakan sebuah kondisi dimana jika hasil deteksi terdapat setidaknya satu wajah, maka dilakukan proses pengenalan dengan memanggil method *recognize*.

media_stream ▸ camera.py

```
37  class VideoCamera(object):
38      def __init__(self):
39          self.video = cv2.VideoCapture(0)
40          self.detector = dlib.get_frontal_face_detector()
41
42      def __del__(self):
43          self.video.release()
44
45      def get_frame(self):
46          success, image = self.video.read()
47          image = cv2.flip(image, 1)
48          #image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
49
50          faces = self.detector(image)
51
52          if len(faces) > 0:
53              i = 1
54              for face in faces:
55                  try:
56                      image = recognize(face=face, image=image,id=i)
57                  except:
58                      print('error')
59                  i += 1
60
61          ret, jpeg = cv2.imencode('.jpg', image)
62          return jpeg.tobytes()
```

Gambar 11. Baris kode 37 sampai 62 pada file *media_stream/camera.py*

5. media_stream/views.py

File "*media_stream/views.py*" berfungsi untuk melakukan *rendering* terhadap tampilan antar muka ketika mengakses antar muka pengenalan wajah secara *realtime*.

media_stream ▸ views.py

```
1  from django.http import StreamingHttpResponse
2  from django.views.decorators.gzip import gzip_page
3
4  from media_stream.camera import VideoCamera
5
6
7  def gen(camera):
8      while True:
9          frame = camera.get_frame()
10         yield (b'--frame\r\n'
11                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
12
13
14 @gzip_page
15 def cam_live(request):
16     try:
17         return StreamingHttpResponse(gen(VideoCamera()),
18         content_type="multipart/x-mixed-replace;boundary=frame")
19     except:
20         pass
```

Gambar 12. Baris kode 1 sampai 20 pada file *media_stream/views.py*

6. media_stream/urls.py

File "*media_stream/url.py*" berfungsi untuk melakukan *routing* pada *URL* yang dapat diakses oleh pengguna.

media_stream ▸ urls.py

```
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      path('', views.cam_live, name='cam_live'),
7  ]
8
```

Gambar 13. Baris kode 1 sampai 8 pada file *media_stream/urls.py*

7. web_app/views.py

Selanjutnya adalah file *views*.py pada folder *web_app* berfungsi untuk melakukan penyajian laman melalui *request* yang dilakukan oleh pengguna. Laman yang ditampilkan berupa file *markup* yaitu HTML5 yang bernama "index.html" yang terdapat pada folder "*web_app/templates*".

web_app ▸ views.py

```
1  from django.shortcuts import render
2
3
4  def index(request):
5      return render(request, "index.html")
```

Gambar 14. Baris kode 1 sampai 5 pada file *web_app/views.py*

8. web_app/urls.py

File "*web_app/url.py*" berfungsi untuk melakukan *routing* pada *URL* yang dapat diakses oleh pengguna.

web_app ▸ urls.py

```
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      path('', views.index, name='index'),
7  ]
```

Gambar 15. Baris kode 1 sampai 7 pada file *web_app/urls.py*

9. facerec_web/urls.py

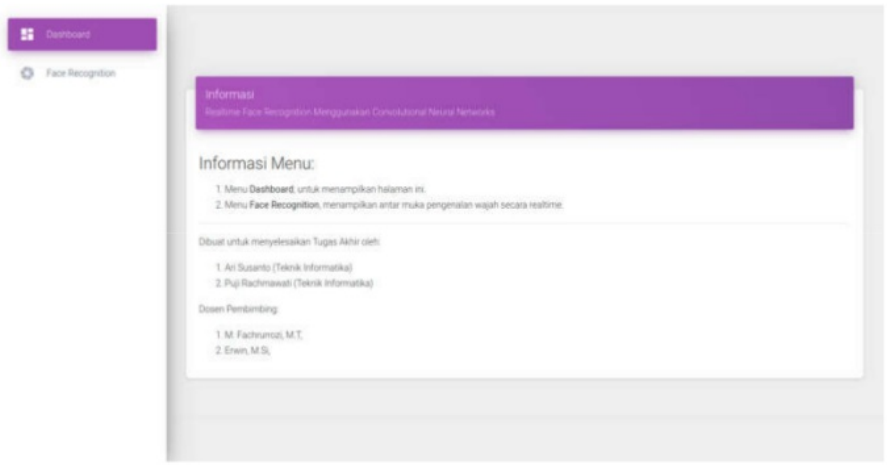
File "*web_app/url.py*" berfungsi untuk melakukan *routing* pada *URL* yang dapat diakses oleh pengguna. *Routing URL* pada file ini sedikit berbeda dengan *routing url* pada file *urls.py* sebelumnya. File "*web_app/url.py*" melakukan *routing URL* secara global dari keseluruhan project serta menambahkan *routing* untuk mengakses *url* media dan file *static* (gambar, css, dan javascript).

facerec_web ▸ urls.py

```
1   from django.urls import include, path
2   from django.conf import settings
3   from django.conf.urls.static import static
4
5   urlpatterns = [
6       path('', include('web_app.urls')),
7       path('stream/', include('media_stream.urls')),
8   ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) +
9   static(settings.STATIC_URL,
10  document_root=settings.STATIC_ROOT)
```

Gambar 16. Baris kode 1 sampai 10 pada file *facerec_web/urls.py*

Setelah mengetahui fungsi dari setiap kode program berikut ini adalah implementasi antar muka yang digunakan oleh program pengenalan wajah secara *realtime* menggunakan *convolutional neural network*. Antar muka laman awal merupakan hasil dari fungsi dalam file "*face_ai/views.py*" yang melakukan *rendering* file HTML untuk ditampilkan pada pengguna.



Gambar 17. Antar muka laman awal

Selanjutnya adalah implementasi antarmuka pengenalan wajah secara *realtime*. Antar muka pengenalan wajah secara *realtime* ini merupakan hasil dari proses *rendering* yang terdapat pada file "*media_stream/views.py*"

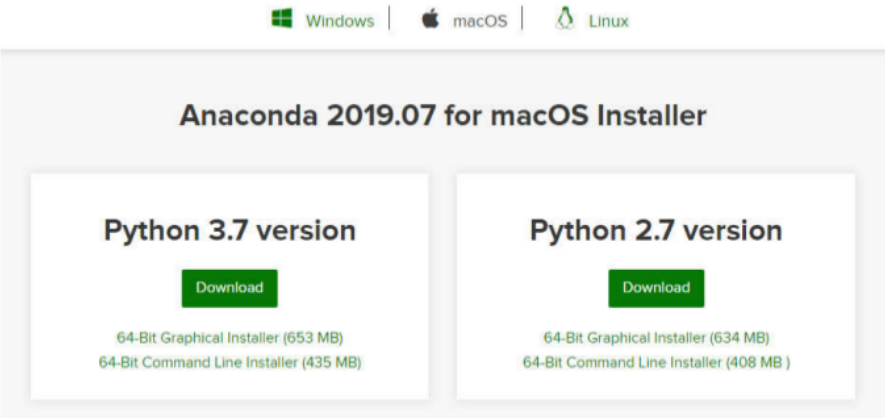
Gambar 18. Antar muka laman pengenalan wajah

Cara Kerja Program Komputer

Program komputer pengenalan wajah secara *realtime* ini berjalan dengan basis web menggunakan *web server* yang telah disediakan oleh *Django framework*. Sebelum menjalankan program, dilakukan persiapan sistem terlebih dahulu. Langkah-langkah dalam persiapan sistem adalah sebagai berikut.

1. Menginstall *anaconda*

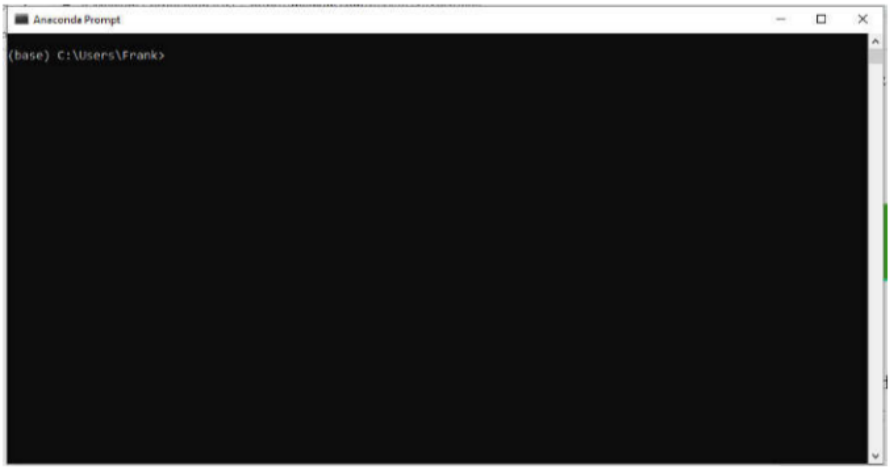
Anaconda merupakan sebuah distribusi platform yang berfungsi untuk memudahkan pengembang dalam membangun program komputer yang bersifat *scientific*. Untuk menginstall *anaconda* yaitu dengan mengunduh *anaconda installer* yang didapatkan dari laman https://www.anaconda.com/distribution/



Gambar 19. Laman download pada *anaconda installer*

Selanjutnya pilih sistem operasi misalnya *windows*, lalu klik *download*. Jalankan *anaconda installer* kemudian ikuti langkah yang disediakan oleh *installer*. Setelah

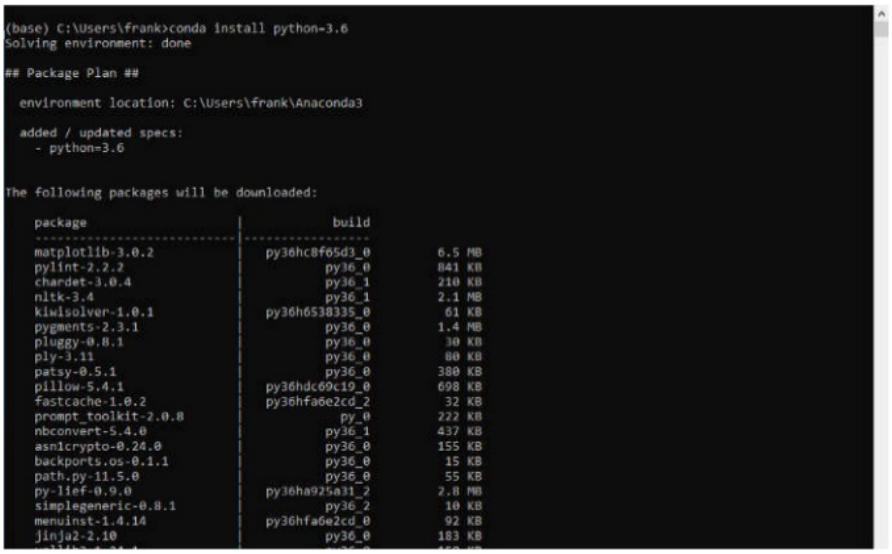
proses instalasi berhasil jalankan *anaconda promt* pada *windows.* Tampilan *anaconda promt* adalah sebagai berikut.



Gambar 20. Tampilan *anaconda prompt*

2. Menginstall *python*

Untuk menginstall *python* pada *anaconda* dilakukan dengan menggunakan perintah conda install python=3.6 proses instalasi akan berlangsung, dan *anaconda* akan membutuhkan konfirmasi pengguna untuk melakukan instalasi paket ketik ‘y’ pada dialog konfirmasi untuk menyetujui proses instalasi.



Gambar 21. Proses instalasi python

3. Menginstall *keras* & *tensorflow*

Tensorflow adalah perpustakaan perangkat lunak (*library*) yang bertujuan untuk melakukan pemrograman yang membutuhkan perhitungan matematika, dan juga digunakan untuk aplikasi pembelajaran mesin seperti *neural network*. *Keras* adalah sebuah *high level* API (*application programming interface*) yang digunakan untuk mempermudah pemanggilan fungsi dari *tensorflow*. Untuk melakukan instalasi *keras* dan *tensorflow* masukan perintah conda install -c anaconda keras pada *anaconda prompt*.

4. Menginstall *OpenCV*

OpenCV adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis, yang dibuat oleh Intel. Untuk melakukan instalasi *OpenCV* masukan perintah pip install opencv-contrib-python pada *anaconda prompt*.

5. Menginstall *Dlib*

Dlib merupakan pustaka perangkat lunak yang berguna untuk membangun program pembelajaran mesin, khususnya pada bidang *computer vision*. Untuk menginstall *Dlib* dilakukan dengan memasukan perintah pip install dlib pada *anaconda prompt.*

6. Menginstall *Pillow*

Pillow atau Python Imaging Library (PIL) adalah perpustakaan gratis untuk bahasa pemrograman Python yang menambahkan dukungan untuk membuka, memanipulasi, dan menyimpan berbagai format file gambar yang berbeda. Untuk melakukan instalasi pillow dilakukan dengan memasukan perintah pip install pillow pada *anaconda prompt.*

7. Menginstall *H5py*

H5py merupakan perpustakaan gratis untuk bahasa pemrograman *Python* yang berfungsi untuk melakukan operasi pada file berformat HDF5. Untuk menginstall *H5py* dilakukan dengan memasukan perintah pip install h5py pada *anaconda prompt.*

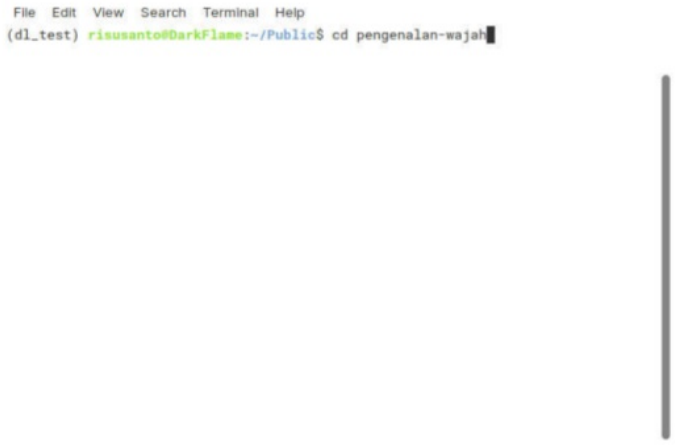
8. Menginstall *Imutils*

Imutils adalah pustaka perangkat lunak yang berguna untuk melakukan fungsi-fungsi dasar pengolahan citra seperti rotasi, *resize,* translasi, dan sebagainya. Untuk melakukan instalasi *Imutils* masukan perintah pip install imutils pada *anaconda prompt.*

Setelah melakukan persiapan sistem dengan melakukan langkah-langkah yang telah disebutkan program komputer pengenalan wajah secara *realtime*

menggunakan *convolutional neural network* siap untuk dijalankan. Arahkan *anaconda prompt* ke dalam direktori program dengan perintah *change directory (cd)* untuk menjalankan program komputer pengenalan wajah secara *realtime*.

Gambar 22. Folder *project* program



Gambar 23. Perintah *cd* untuk memasuki direktori *project*

Panduan dalam mengoperasikan program komputer pengenalan wajah secara *realtime* terbagi menjadi tiga tahapan yaitu persiapan data, proses pelatihan, dan menjalankan pengenalan wajah secara *realtime.*

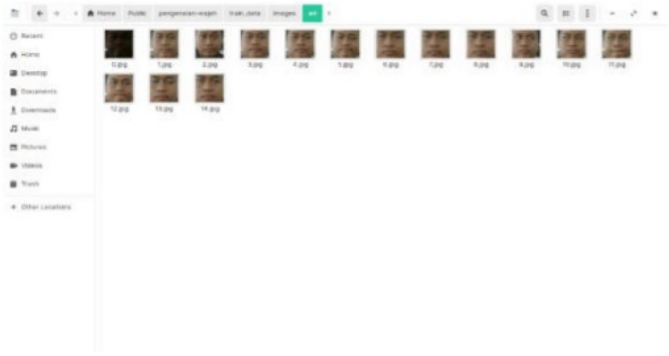
a. Persiapan Data

Tahap persiapan data berguna untuk mengumpulkan data citra wajah yang akan dikenali. Pengumpulan data dilakukan dengan menjalankan *python script* yang bernama *create_face.py* dengan cara memasukan perintah python create_face.py dialog program akan berjalan. Dialog pertama adalah untuk menentukan nama label dari citra yang akan dikumpulkan, misalnya 'ari'. Dialog kedua adalah jumlah data dari label 'ari' yang akan dikumpulkan, semakin banyak maka akan semakin baik, disini sebagai contoh input jumlah citra yang dikumpulkan sebanyak 15 citra.

Gambar 24. Menjalankan *script create_face.py*

Selanjutnya program akan mengambil data dengan cara membuka kamera yang terhubung pada perangkat komputer, posisikan wajah pengguna yang akan disimpan sampel citra wajahnya pada kamera perangkat komputer. Setelah proses pengumpulan data selesai, data hasil proses persiapan data dapat dilihat pada folder "train_data/images" seperti terlihat pada gambar.

Gambar 25. Data yang berhasil dikumpulkan citra dari setiap label dikumpulkan dalam satu folder



Gambar 26. Contoh citra yang dikumpulkan

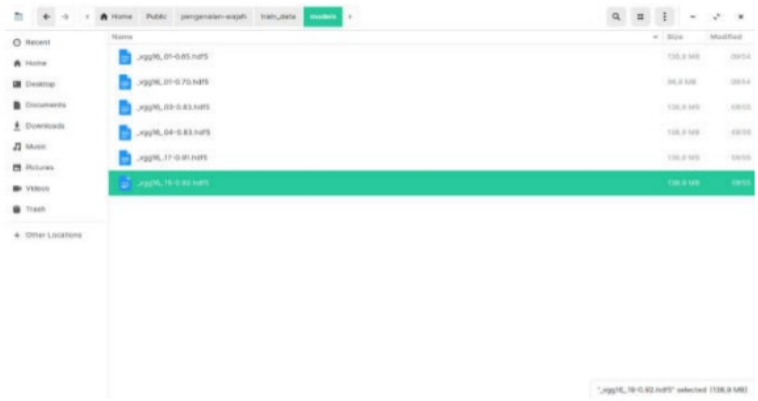
b. Pelatihan (*training*)

Proses pelatihan dilakukan untuk membangun model convolutional neural network yang berguna untuk melakukan klasifikasi terhadap data citra. Seluruh data yang dikumpulkan pada tahap persiapan data akan melalui proses learning menggunakan convolutional neural network. Untuk melakukan proses pelatihan masukan perintah python face_recognition.py -o start_train -m simple parameter "-m simple" berfungsi untuk memilih arsitektur *convolutional neural network* yang akan digunakan. Untuk memilih arsitektur *VGG16* ganti parameter dengan "-m vgg16" perlu diketahui untuk menggunakan arsitektur *VGG16* hanya jika data yang dikumpulkan untuk setiap label lebih dari 100 citra. Proses pelatihan akan berjalan, lama proses pelatihan bergantung pada spesifikasi perangkat komputer dan banyaknya data yang dikumpulkan pada tahap persiapan data.

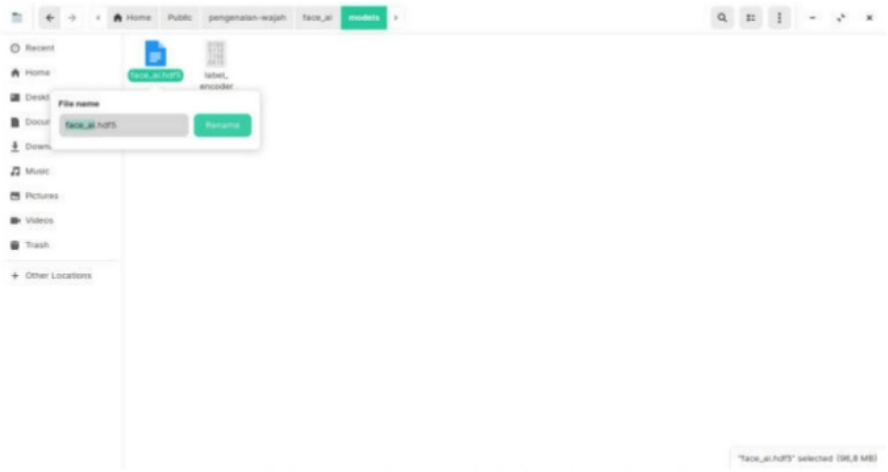
Gambar 27. Proses pelatihan

Setelah proses pelatihan selesai model yang dibangun selama proses pelatihan tersimpan sebagai file berformat HDF5 pada folder "train_data/models/". Aturan penamaan file model adalah "_<*model*>_.<*epoch*>-<*akurasi*>.*hdf5*"

sebagai contoh nama file model pada epoch ke 18 dengan akurasi 0.92 (92%) yaitu "*_vgg16_.18-0.92.hdf5"*. Selanjutnya adalah memindahkan file model dengan akurasi tertinggi misalnya file model "*_vgg16_.18-0.92.hdf5"* ke dalam folder "face_ai/models" lalu ubah nama file menjadi "*face_ai.hdf5*".



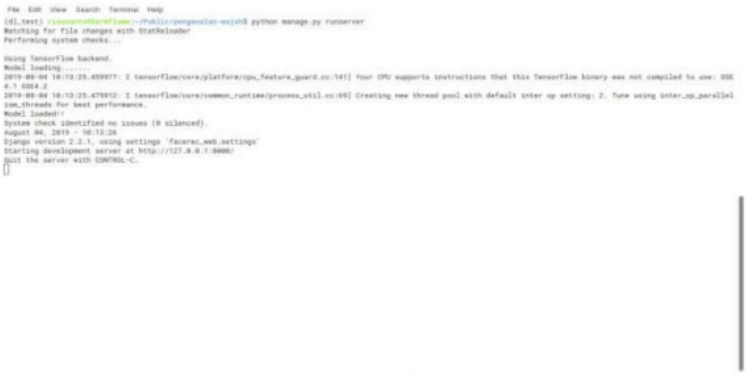
Gambar 27. File model yang dihasilkan dari proses pelatihan



Gambar 28. Pindahkan salah satu file folder ke dalam direktori "face_ai/models" dan ubah nama file menjadi *"face_ai.hdf5"*

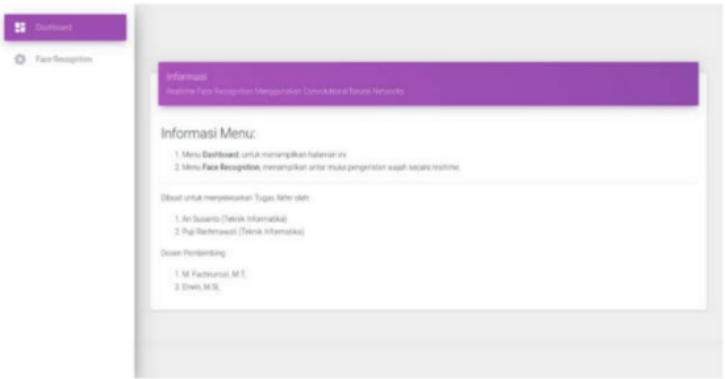
c. Pengenalan Wajah secara *Realtime*

Tahap selanjutnya adalah menjalankan server dan melakukan pengenalan wajah secara *realtime*. Menjalankan server dapat dilakukan dengan memasukan perintah python manage.py runserver alamat web untuk mengakses program akan ditampilkan pada *output* dari perintah yang dimasukan, jika program dijalankan secara lokal maka alamat web adalah http://127.0.0.1:8000/.



Gambar 29. Menjalankan *server*

Halaman antarmuka awal (*index page*) dari program adalah sebagai berikut.



Gambar 30. Laman awal (*index page*)

Pada *side bar* terdapat dua menu yaitu *dashboard* dan *face recognition.* Menu dashboard berguna untuk menampilkan laman awal. Menu *face recognition* untuk mengakses antar muka pengenalan wajah secara *realtime*. Klik menu *face recognition* maka kamera yang terhubung pada perangkat komputer akan aktif dan mulai menangkap citra melalui kamera pada antar muka pengenalan wajah secara *realtime*. Berikut merupakan antarmuka pengenalan wajah secara *realtime*.

Gambar 31. Antar muka pengenalan wajah

Menutup program pengenalan wajah secara *realtime* dapat dilakukan dengan mematikan server. Untuk mematikan server tekan tombol “ctrl” lalu diikuti dengan menekan tombol “c” tanpa melepas tombol “ctrl” pada keyboard. Proses pengenalan wajah akan terhenti ketika server dimatikan.

Keunggulan Program Komputer

Program komputer pengenalan wajah secara *realtime* menggunakan *convolutional neural network* memiliki keunggulan disamping menggunakan metode yang sudah terbukti menghasilkan performa akurasi yang baik. Keunggulan dari program komputer yang dibangun adalah sebagai berikut:

1. Dapat mengenali wajah secara *realtime* tanpa harus menunggu proses input citra secara menual sehingga hasil pengenalan dapat langsung terlihat melalui antar muka.
2. Dapat mengenali banyak wajah dalam sekali waktu.
3. *Respond time* dari sistem pada proses pengenalan wajah kurang dari satu detik.
4. Program komputer bersifat *cross platform*, menggunakan bahasa pemrograman python, dimana python dapat dijalankan pada berbagai sistem operasi. Program pengenalan wajah secara *realtime* ini dapat dijalankan pada sistem operasi yang mendukung instalasi bahasa pemrograman python seperti *Windows, Linux,* dan *MacOS*.

Kekurangan Program Komputer

Program komputer pengenalan wajah secara *realtime* menggunakan *convolutional neural network* yang telah dibangun masih memiliki beberapa kelemahan yaitu:

1. Membutuhkan performa komputasi yang tinggi bergantung pada spesifikasi komputer yang menjalankan program.

2. Pengenalan wajah hanya dapat dilakukan dengan baik dengan intensitas cahaya yang cukup. Lingkungan dengan cahaya terlalu redup atau terlalu terang dapat menurunkan performa akurasi dalam pengenalan wajah.

Kesimpulan

Program komputer pengenalan wajah secara *realtime* menggunakan *convolutional neural network* merupakan sebuah perangkat lunak berbasis web yang berfungsi untuk melakukan pengenalan wajah. Program dapat mengenali banyak wajah dalam sekali waktu secara langsung (*realtime*) melalui input dari kamera yang terhubung pada perangkat komputer.

DAFTAR PUSTAKA

[1] A. Juhong and C. Pintavirooj, "Face recognition based on facial landmark detection," *BMEiCON 2017 - 10th Biomed. Eng. Int. Conf.*, vol. 2017-Janua, no. 2, pp. 1–4, 2017.

[2] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face Recognition Using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, 2005.

[3] S. N. Sujay, H. S. M. Reddy, and J. Ravi, "Face recognition using extended LBP features and multilevel SVM classifier," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017, pp. 1–4.

[4] Yan Lei, "Fusion method of PCA and BP neural network for face recognition," in *2011 International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 3256–3259.

[5] K. Yan, S. Huang, Y. Song, W. Liu, and N. Fan, "Face Recognition Based on Convolution Neural Network," in *Proceedings of the 36th Chinese Control Conference*, 2017, pp. 4077–4081.

SISTEM DETEKSI MULTI WAJAH SECARA REALTIME MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

ORIGINALITY REPORT

13%	3%	2%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sriwijaya University Student Paper	8%
2	repository.unsri.ac.id Internet Source	1%
3	Submitted to Universitas Brawijaya Student Paper	1%
4	Submitted to University of Strathclyde Student Paper	1%
5	www.yongxu.org Internet Source	1%
6	Ridha Ilyas Bendjillali, Mohammed Beladgham, Khaled Merit, Abdelmalik Taleb-Ahmed. "Illumination-robust face recognition based on deep convolutional neural networks architectures", Indonesian Journal of Electrical Engineering and Computer Science, 2020 Publication	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		