PENGENALAN WAJAH SECARA *REALTIME* MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* PADA CITRA *MULTI-FACE*

Diajukan Sebagai Syarat Untuk Menyelesaikan
Pendidikan Program Strata-1 Pada
Jurusan Teknik Informatika

Oleh:

Ari Susanto
NIM : 09021181520127

Jurusan Teknik Informatika
FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA
2019

LEMBAR PENGESAHAN TUGAS AKHIR

PENGENALAN WAJAH SECARA *REALTIME* MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK
PADA CITRA *MULTI-FACE*

Oleh :

ARI SUSANTO
NIM : 09021181520127

Indralaya, Agustus 2019

Pembimbing I,

M. Fachrurrozi, S.Si., M.T.
NIP. 198005222008121002

Pembimbing II,

Dr. Erwin, S.Si., M.Si.
NIP. 197101291994121001

Mengetahui,
Ketua Jurusan Teknik Informatika,

Rifkie Primartha, M.T.
NIP. 197706012009121004

TANDA LULUS UJIAN SIDANG TUGAS AKHIR

Pada hari Jumat tanggal 26 Juli 2019 telah dilaksanakan ujian sidang tugas akhir oleh Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.

Nama	:	Ari Susanto
N I M	:	09021181520127
Judul	:	Pengenalan Wajah secara *Realtime* Menggunakan *Convolutional Neural Network* Pada Citra *Multi-Face*

1. Pembimbing I

   M. Fachrurrozi, S.Si., M.T .....
   NIP. 198005222008121002

2. Pembimbing II

   Dr. Erwin, S.Si., M.Si. .....
   NIP. 197101291994121001

3. Penguji I

   Firdaus, M.Kom. .....
   NIP. 197801212008121003

4. Penguji II

   M. Naufal Rachmatullah, M.T. .....
   NIP.

Mengetahui,
Ketua Jurusan Teknik Informatika

Rifkie Primartha S.T. M.T.
NIP 197706012009121004

HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini :

Nama	:	Ari Susanto
N I M	:	09021181520127
Program Studi	:	Teknik Informatika
Judul	:	Pengenalan Wajah secara *Realtime* Menggunakan *Convolutional Neural Network* Pada Citra *Multi-Face*

Hasil Pengecekan Software *Turnitin* : 16 %

Menyatakan bahwa Laporan Projek saya merupakan hasil karya sendiri dan bukan hasil penjiplakan/plagiat. Apabila ditemukan unsur penjiplakan/plagiat dalam laporan projek ini, maka saya bersedia menerima sanksi akademik dari Universitas Sriwijaya sesuai dengan ketentuan yang berlaku.

Demikian, pernyataan ini saya buat dengan sebenarnya dan tidak ada paksaan oleh siapapun.

Palembang, 15 Agustus 2019

Ari Susanto
NIM. 09021181520127

"If you don't understand, don't worry about it"

-Andrew NG-

PENGENALAN WAJAH SECARA REALTIME MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK PADA CITRA MULTI-FACE

Oleh:
Ari Susanto
09021181520127

ABSTRAK

Penelitian ini membahas mengenai pengenalan wajah secara *realtime* menggunakan *convolutional neural network*. Pada penelitian ini menggunakan dua arsitektur *convolutional neural network* yaitu *VGG16* dan arsitektur *convolutional neural network* sederhana *(Simple Convolutional Neural Network)* yang terdiri dari dua *convolutional layer,* satu *pooling layer*, dan dua *fully connected layer*. Arsitektur *VGG16* terdiri dari 13 *convolutional layer, 5 pooling layer, 2 fully connected layer*. Pengujian secara *offline* dilakukan pada dataset sekunder yaitu *AT & T Face Database* mendapatkan nilai akurasi sebesar 95% pada arsitektur *Simple Convolutional Neural Network* dan akurasi yang didapat menggunakan arsitektur *VGG16* adalah 98%. Pengujian juga dilakukan secara *offline* dan *realtime* menggunakan data dari 11 mahasiswa Teknik Informatika Universitas Sriwijaya. Pengujian secara *offline* mendapatkan akurasi sebesar 99% menggunakan *Simple Convolutional Neural Network* dan akurasi sebesar 98% menggunakan arsitektur *VGG16*. Untuk pengujian secara *realtime* dan mendapatkan nilai akurasi sebesar 86% dengan rata-rata *respond time* selama 0.4 detik menggunakan arsitektur *VGG16* dan akurasi sebesar 70%. Dengan rata-rata *respond time* selama 0.02 detik menggunakan arsitektur *Simple Convolutional Neural Network.*

Kata Kunci: *Face recognition, deep learning, convolutional neural network*

Pembimbing I,

M. Fachrurrozi, S.Si., M.T.
NIP. 198005222008121002

Pembimbing II,

Dr. Erwin, S.Si., M.Si.
NIP. 197101291994121001

Mengetahui,
Ketua Jurusan Teknik Informatika,

Rifkie Primartha, M.T.
NIP. 197706012009121004

REALTIME FACE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK ON MULTI-FACE IMAGE

By:
Ari Susanto
09021181520127

ABSTRACT

This research is about realtime face recognition using convolutional neural network. In this research uses two convolutional neural network architectures, VGG16 and a simple convolutional neural network architecture consisting of two *convolutional layers*, one *pooling layer*, and two *fully connected layers*. The VGG16 architecture consists of 13 *convolutional layers*, 5 *pooling layers*, 2 *fully connected layers*. Offline testing is performed on AT & T Face Database and get an accuracy value of 95% on the *Simple Convolutional Neural Network architecture* and the accuracy obtained using VGG16 architecture is 98%. The test was also carried out offline and in realtime using data from 11 Informatics Engineering students at Sriwijaya University. Offline testing gets an accuracy of 99% using the Simple Convolutional Neural Network and an accuracy of 98% using the VGG16 architecture. For realtime testing accuracy value is 86% with an average respond time of 0.4 seconds using VGG16 architecture and 70% of accuracy with an average respond time of 0.02 seconds using the *Simple Convolutional Neural Network architecture*.

Keywords: *Face recognition, deep learning, convolutional neural network*

Pembimbing I,

M. Fachrurrozi, S.Si., M.T.
NIP. 198005222008121002

Pembimbing II,

Dr. Erwin, S.Si., M.Si.
NIP. 197101291994121001

Mengetahui,
Ketua Jurusan Teknik Informatika,

Rifkie Primartha, M.T.
NIP. 197706012009121004

KATA PENGANTAR

Syukur *Alhamdulillah* segala puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat limpahan Rahmat dan Ridho-Nya, hingga penulisan Laporan Tugas Akhir dengan judul "*Pengenalan Wajah secara Realtime Menggunakan Convolutional Neural Network pada Citra Multi Face*" ini dapat penulis selesaikan dengan baik.

Selama pembuatan Laporan Tugas Akhir ini, penulis banyak menemukan hambatan dan kesulitan, namun berkat bimbingan dan pengarahan serta bantuan dari berbagai pihak, maka penulis dapat selesaikan. Untuk itu pada kesempatan ini penulis ingin menyampaikan ucapan terimakasih kepada :

1. Keluarga tercinta yaitu Ayah, Ibu dan seluruh saudara yang selalu memberikan semangat, nasihat, dan do'a kepada penulis agar dapat sukses menjalani perkuliahan serta dapat menyelesaikan laporan tugas akhir ini.
2. Bapak Jaidan Jauhari,.M.IT selaku Dekan Fakultas Ilmu Komputer Universitas Sriwijaya.
3. Bapak Rifkie Primartha, M.T. selaku Ketua Jurusan Teknik Informatika.
4. Bapak M.Fachrurrozi, M.T. dan Erwin, M.Si. selaku pembimbing Tugas Akhir.
5. Segenap Dosen Fakultas Ilmu Komputer yang telah membekali ilmu kepada penulis sehingga penulis bisa menjalani dan menyelesaikan tugas akhir dengan baik.
6. Staff administrasi Fakultas Ilmu Komputer yang telah memberikan kemudahan dalam hal administrasi sehingga penulis dapat menjalani tugas akhir dengan lancar.
7. Penulis juga berterima kasih kepada semuanya yang tidak disebutkan di sini dan memohon maaf yang setulus-tulusnya dari mereka semua atas kesalahan penulis selama melaksanakan tugas akhir.

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kesempurnaan, baik teknis penulisan, bahasa maupun cara pemaparannya. Oleh karena itu saran dan tanggapan dari semua pihak sangat kami harapkan demi kesempurnaan laporan ini.

Penulis berharap semoga Laporan Tugas Akhir ini dapat bermanfaat bagi penulis khususnya, dan bagi mahasiswa Fakultas Ilmu Komputer Universitas Sriwijaya pada umumnya serta dapat memberikan masukan sebagai sumbangan pikiran dalam rangka peningkatan mutu dalam pembelajaran.

Palembang, 5 Juli 2019
Hormat Saya,

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN TUGAS AKHIR ii
TANDA LULUS UJIAN SIDANG TUGAS AKHIR iii
HALAMAN PERNYATAAN iv
MOTTO DAN PERSEMBAHAN v
ABSTRACT vi
ABSTRAK vii
KATA PENGANTAR viii
DAFTAR ISI x
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
DAFTAR LAMPIRAN xvi

BAB I PENDAHULUAN I-1
1.1 Pendahuluan I-1
1.2 Latar Belakang Masalah I-1
1.3 Rumusan Masalah I-3
1.4 Tujuan Penelitian I-4
1.5 Manfaat Penelitian I-4
1.6 Batasan Masalah I-4
1.7 Sistematika Penulisan I-4
1.8 Kesimpulan I-5

BAB II KAJIAN LITERATUR II-1
2.1 Pendahuluan II-1
2.1 Pengenalan Wajah II-1
2.4 Jaringan Syaraf Tiruan (*Neural Networks*) II-1
2.5 *Backpropagation* II-5
2.5.1 Feedfordward II-5
2.5.2 Menghitung Nilai *Error* pada *Layer* Terakhir II-6
2.5.3 Menghitung Nilai *Error* pada Setiap *Layer* II-6
2.5.4 Menghitung *Partial Derivative* untuk Fungsi *Cost* terhadap Bobot 7
2.5.5 Ubah Nilai Bobot dan Bias II-7
2.6 Convolutional Neural Networks II-8

2.6.1 Operasi Konvolusi pada *Convolutional Neural Networks*..... II-9
2.6.2 *Pooling*..... II-10
2.6.3 Arsitektur *Convolutional Neural Networks*..... II-12
2.7 *Confusion Matrix*..... II-12
2.8 *Rational Unified Process*..... II-13
2.9 Penelitian Lain yang Relevan..... II-14
2.9.1 Kewen Yan, Shaohui Huang, Yaoxian Song, Wei Liu, Neng Fan: *Face Recognition Based on Convolutional Neural Network*, Chinese Control Conference, 2017..... II-15
2.9.2 A. Lebedev, V. Khryashchev, A. Priorov, O. Stepanova: *Face Verification Based on Convolutional Neural Network and Deep Learning*, IEEE East-West Design and Test Symposium, 2017. ..... II-15
2.10Kesimpulan..... II-16

BAB III METODOLOGI PENELITIAN..... III-1
3.1 Pendahuluan III-1
3.2 Unit Penelitian..... III-1
3.3 Pengumpulan Data..... III-1
3.3.1 Jenis dan Sumber Data III-1
3.3.2 Metode Pengumpulan Data III-4
3.4 Tahapan Penelitian III-5
3.4.1 Menetapkan Kerangka Kerja / *Framework* III-5
3.4.2 Menetapkan Kriteria Pengujian..... III-12
3.4.3 Menetapkan Format Data Pengujian III-12
3.4.4 Menentukan Alat yang Digunakan dalam Pelaksanaan Penelitian III-12
3.4.5 Melakukan Pengujian Penelitian III-13
3.4.6 Melakukan Analisa Hasil Pengujian dan Membuat Kesimpulan III-13
3.5 Metode Pengembangan Perangkat Lunak III-13
3.5.1 Fase Insepsi..... III-13
3.5.2 Fase Elaborasi..... III-14
3.5.3 Fase Konstruksi..... III-14
3.5.4 Fase Transisi III-15
3.6 Manajemen Proyek Penelitian..... III-15

BAB IV PENGEMBANGAN PERANGKAT LUNAK IV-1
4.1 Pendahuluan IV-1
4.2 Rational Unified Process..... IV-1
4.2.1 Fase Insepsi..... IV-1

4.2.1.1 Pemodelan Bisnis IV-2
4.2.1.2 Kebutuhan Sistem IV-2
4.2.1.3 Analisis dan Desain IV-2
4.2.2 Fase Elaborasi IV-9
4.2.2.1 Pemodelan Bisnis IV-9
4.2.3 Fase Konstruksi IV-15
4.2.3.1 Kebutuhan Sistem IV-15
4.2.3.2 Implementasi IV-15
4.2.4 Fase Transisi IV-17
4.2.4.1 Pemodelan Bisnis IV-17
4.2.4.2 Kebutuhan Sistem IV-17
4.2.4.3 Pengujian Perangkat Lunak IV-18
4.3 Kesimpulan IV-19

BAB V HASIL DAN ANALISIS PENELITIAN V-1
5.1 Pendahuluan V-1
5.2 Hasil Percobaan Penelitian V-1
5.2.1. Skenario Percobaan Pertama V-7
5.2.2. Skenario Percobaan Kedua V-14
5.3 Analisis Hasil Percobaan Penelitian V-17
5.4 Kesimpulan V-18

BAB VI KESIMPULAN DAN SARAN VI-1
6.1 Pendahuluan VI-1
6.2 Kesimpulan VI-1
6.3 Saran VI-2

DAFTAR PUSTAKA xvii
LAMPIRAN xix

DAFTAR TABEL

Halaman

III-1. Arsitektur *Simple CNN* III-6
III-2. Arsitektur VGG16 ..... III-6
III-3. Rancangan Tabel Pengujian Performa Pengenalan Wajah III-12
III-4. Tabel Penjadwalan Penelitian dalam Bentuk Work Breakdown Structure (WBS)…. III-16
IV-1. Kebutuhan Fungsional IV-3
IV-2. Kebutuhan Non Fungsional. IV-4
IV-3. Definisi Aktor Use Case IV-5
IV-4. Definisi Use Case IV-6
IV-5. Spesifikasi dalam pengembangan perangkat lunak IV-11
IV-6. Kebutuhan sistem fase transisi IV-17
IV-7. Pengujian unit pada use case training IV-18
IV-8. Pengujian unit pada pengenalan wajah IV-19
V-1. Hasil Percobaan pada AT & T face database pada arsitektur VGG16 ..... V-8
V-2. Hasil Percobaan pada citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16 ..... V-9
V-3. Hasil Percobaan pada AT & T face database pada arsitektur Simple CNN V-10
V-4. Hasil Percobaan pada citra wajah mahasiswa fasilkom unsri pada arsitektur Simple CNN V-11
V-5. Performa precision, sensitifity, dan f1-score dataset citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16 ..... V-13
V-6. Performa precision, sensitifity, dan f1-score dataset citra wajah mahasiswa fasilkom unsri pada arsitektur Simple CNN V-13
V-7. Spesifikasi integrated webcam pada Dell Latitude E4310 ..... V-14
V-8. Pengujian Pada Device Dell Latitude E4310 arsitektur VGG16 ..... V-15
V-9. Pengujian Pada Device Dell Latitude E4310 arsitektur Simple CNN V-16
V-10. Perbandingan Performa arsitektur convolutional neural network saat pengujian secara realtime V-18

DAFTAR GAMBAR

Halaman

II-1. Cara kerja *perceptron* II-2
II-2. Bentuk dari jaringan syaraf tiruan II-3
II-3. Contoh operasi konvolusi tanpa membalik *kernel* II-10
II-4. Tahapan pada *convolution neural networks* II-11
II-5. Arsitektur RUP II-14
III-1. Sampel Dari AT&T Database III-2
III-2. Distribusi Data AT&T Database III-3
III-3. Sampel Citra wajah mahasiswa teknik informatika III-4
III-4. Distribusi Data Citra wajah mahasiswa teknik informatika III-4
III-5. Alur proses pelatihan (*training*) III-9
III-6. Alur proses pengujian Skenario Pertama III-10
III-7. Alur proses pengujian Skenario Kedua III-11
IV-1. *Use case diagram* IV-5
IV-2. Diagram aktifitas training. IV-7
IV-3. Diagram aktivitas pengenalan wajah IV-8
IV-4. Data citra beberapa mahasiswa Teknik Informatika IV-9
IV-5. Sampel Dari AT&T Database IV-9
IV-6. Rancangan antarmuka pengenalan wajah IV-11
IV-7. Sequence Diagram Training IV-13
IV-8. Sequence Diagram Pengenalan Wajah IV-14
IV-9. Implementasi antar muka IV-16
IV-10. Implementasi Antar muka IV-16
V-1. Grafik Distribusi Data Pelatihan Citra Wajah Teknik Informatika V-2
V-2. Grafik proses traning menampilkan nilai akurasi pada setiap epoch pada AT & T dataset arsitektur VGG16. ..... V-3
V-3. Grafik training menampilkan nilai *loss* pada setiap epoch pada AT & T dataset arsitektur VGG16 ..... V-3
V-4. Grafik training menampilkan nilai akurasi pada setiap epoch pada dataset wajah mahasiswa Teknik Informatika Universitas Sriwijaya arsitektur VGG16 ..... V-4
V-5. Grafik training menampilkan nilai *loss* setiap epoch pada dataset wajah mahasiswa Teknik Informatika Universitas Sriwijaya arsitektur VGG16 ..... V-4
V-6. Grafik proses traning menampilkan nilai akurasi pada setiap epoch pada AT & T dataset arsitektur *Simple CNN* V-5
V-7. Grafik training menampilkan nilai *loss* pada setiap epoch pada AT & T dataset arsitektur *Simple CNN* V-5
V-8. Grafik training menampilkan nilai akurasi pada setiap epoch pada dataset wajah mahasiswa Teknik Informatika

Universitas Sriwijaya arsitektur *Simple CNN* V-6
V-9. Grafik training menampilkan nilai *loss* setiap epoch pada dataset wajah mahasiswa Teknik Informatika UNSRI arsitektur *Simple CNN* V-6
V-10. *Confusion Matrix* citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16 ..... V-12
V-11. *Confusion Matrix* citra wajah mahasiswa fasilkom unsri *Simple CNN* V-12

DAFTAR LAMPIRAN

Halaman

LAMPIRAN I : Pengujian pada *AT & T Face Database*.....L-1
LAMPIRAN II : Pengujian pada Data Wajah Mahasiswa Teknik Informatika L-4
LAMPIRAN III : Kode Program L-74

BAB I

PENDAHULUAN

1.1 Pendahuluan

Pada bab ini membahas mengenai latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian serta batasan masalah yang menjadi gambaran umum mengenai penelitian yang akan dilakukan. Pendahuluan dimulai dengan penjelasan mengenai pengenalan wajah manusia serta masalah yang terdapat dalam pengenalan wajah. Penelitian yang berkaitan dengan pengenalan atau identifikasi wajah serta metode yang digunakan disertakan dalam latar belakang dari penelitian ini.

1.2 Latar Belakang Masalah

Pengenalan objek biometrik seperti wajah telah banyak diterapkan untuk berbagai keperluan seperti sistem absensi, akses kontrol, keamanan bandara, dan sebagainya (Juhong & Pintavirooj, 2017). Beberapa metode atau teknik mengenai pengenalan wajah telah dilakukan pada penelitian-penelitian terdahulu antara lain pengenalan wajah menggunakan *Laplacianface* (He, Yan, Hu, Niyogi, & Zhang, 2005), pengenalan wajah menggunakan *Support Vector Machine* (Sujay, Reddy, & Ravi, 2017), dan menggunakan gabungan *Principal Component Analysis* dan *Backpropagation* (Yan Lei, 2011). Ketiga penelitian ini menggunakan citra *single face* yaitu citra yang memiliki satu objek wajah dalam satu citra.

Metode terdahulu dalam pengenalan wajah yang telah disebutkan menggunakan ekstraksi ciri wajah yang dilakukan dengan metode tradisional (*handcrafted*) seperti *Local Binary Pattern* dan *Principal Component Analysis* menyebabkan ciri pada sebuah citra dalam kondisi tertentu tidak dapat terekstraksi dengan baik atau ciri yang tidak terlalu penting kemungkinan justru akan terekstraksi. Berdasarkan kekurangan tersebut, proses klasifikasi akan terganggu dikarenakan pengklasifikasian yang dilakukan sangat bergantung pada fitur wajah yang berhasil diekstraksi (Goodfellow, Bengio, & Courville, 2016).

Deep Learning merupakan bagian dari bidang ilmu *machine learning*. *Deep learning* tidak seperti metode klasik *machine learning* lainnya yang sangat bergantung pada representasi yang baik dari sebuah data melalui proses *feature extraction* ataupun *feature selection*, *deep learning* dapat mempelajari ciri atau representasi pada data itu sendiri (Goodfellow et al., 2016). *Convolutional Neural Network* merupakan salah satu metode dalam *deep learning. Convolutional Neural Network* dapat langsung menerima masukan citra wajah yang merupakan vektor *multi-dimensional* dimana dapat mengurangi kompleksitas pada proses rekonstruksi data untuk mendapatkan representasi yang baik pada proses klasifikasi. *Convolutional Neural Network* telah berhasil diterapkan pada pengenalan karakter, pengenalan wajah, dan pengenalan objek (Lebedev, Khryashchev, Priorov, & Stepanova, 2017).

Tahun 2017 pada penelitian yang berjudul *Face Recognition Based on Convolutional Neural Network* (Yan, Huang, Song, Liu, & Fan, 2017) dibahas mengenai pengenalan wajah atau *face recognition* menggunakan *convolutional*

neural network. Arsitektur yang digunakan terdiri dari tiga convolution layer, dua pooling layer, dua full-connected layer dan satu Softmax regression layer. Menggunakan dua dataset yaitu ORL dan AR penelitian ini mendapatkan hasil akurasi 99.78 % pada AR dataset dan 99.82% pada ORL dataset. Kemudian dalam topik penelitian yang sama, pada Pada tahun 2017, A. Lebedev, V. Khryashchev, A. Priorov, dan O. Stepanova melakukan studi verifikasi wajah menggunakan algoritma *Convolutional Neural Network* yang ditulis dalam jurnal berjudul *Face Verification Based on Convolutional Neural Network and Deep Learning*. Eksperimen menggunakan *Convolutional Neural Network* menghasilkan EER (*Equal Error Rate*) sebesar 9,4% yang menunjukan tingkat akurasi yang tinggi bila dibandingkan dengan algoritma pendekatan klasik (Lebedev et al., 2017).

Dalam penelitian ini akan dilakukan implementasi pada pengenalan wajah menggunakan citra *multi* wajah secara *realtime* menggunakan *Convolutional Neural network.* Citra *multi* wajah yaitu dimana dalam satu citra terdapat lebih dari satu wajah yang akan dikenali.

1.3 Rumusan Masalah

Rumusan masalah dalam penelitian ini dibagi menjadi beberapa pertanyaan penelitian yaitu:

1. Bagaimana akurasi pengenalan wajah menggunakan *Convolutional Neural Networks*?

2. Untuk setiap proses pengenalan wajah secara *realtime,* bagaimana performa sistem berdasarkan *respond time*?

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengimplementasikan *Convolutional Neural Networks* untuk mengenali wajah pada citra *multi* wajah. pada sistem pengenalan wajah secara *realtime*.

1.5 Manfaat Penelitian

Manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut :

1. Hasil penelitian dapat digunakan untuk mendukung bidang biometric, sistem absensi, dan berbagai bidang keamanan.
2. Hasil penelitian dapat digunakan sebagai landasan berpikir dalam penelitian mendatang.

1.6 Batasan Masalah

Batasan masalah dalam penelitian identifikasi wajah menggunakan *Convolutional Neural Network* adalah sebagai berikut:

1. Posisi wajah pada citra adalah menghadap kedepan (*frontal*).
2. Masukan citra yang diterima memiliki pencahayaan yang cukup.

1.7 Sistematika Penulisan

Sistematika penulisan laporan penelitian ini adalah sebagai berikut:

BAB I. PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian serta batasan masalah yang menjadi gambaran umum mengenai

penelitian yang akan dilakukan. Pendahuluan dimulai dengan penjelasan mengenai pengenalan wajah manusia serta masalah yang terdapat dalam pengenalan wajah. Penelitian yang berkaitan dengan pengenalan atau identifikasi wajah serta metode yang digunakan disertakan dalam latar belakang dari penelitian ini.

BAB II. KAJIAN PUSTAKA

Pada bab ini akan dibahas dasar-dasar teori yang digunakan dalam penelitian, seperti sistem pengenalan atau identifikasi wajah, tahapan-tahapan dalam identifikasi wajah, *Neural Networks, Multi-layer Perceptron, Convolutional Neural Networks,* desain model, dan metode pengujian. Pada bab ini juga akan membahas mengenai penelitian-penelitian lain yang relevan.

BAB III. METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai tahapan yang akan dilaksanakan pada penelitian ini. Masing-masing rencana tahapan penelitian dideskripsikan dengan rinci dengan mengacu pada suatu kerangka kerja. Di akhir bab ini berisi perancangan manajemen proyek pada pelaksanaan penelitian.

1.8 Kesimpulan

Penelitian mengenai identifikasi wajah wajah menggunakan *Convolutional Neural Network* pada citra *multi* wajah bertujuan mengimplementasikan

Convolutional Neural Network dalam membangun sistem pengenalan wajah secara *realtime* yang dapat mengindentifikasi wajah pada citra *multi* wajah.

BAB II

KAJIAN LITERATUR

2.1 Pendahuluan

Pada bab ini akan dibahas dasar-dasar teori yang digunakan dalam penelitian, seperti sistem pengenalan atau identifikasi wajah, tahapan-tahapan dalam identifikasi wajah, *Neural Networks, Multi-layer Perceptron, Convolutional Neural Networks,* desain model, dan metode pengujian. Pada bab ini juga akan membahas mengenai penelitian-penelitian lain yang relevan.

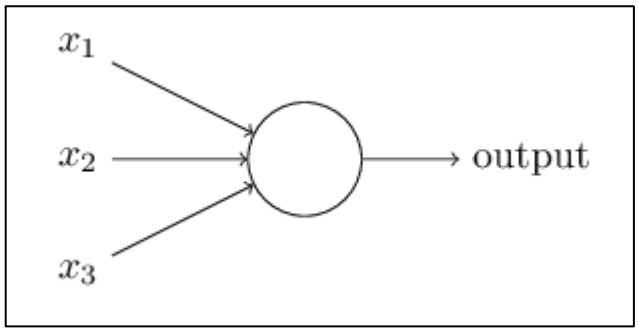
2.1 Pengenalan Wajah

Pengenalan wajah adalah teknologi komputer untuk mengukur dan mencocokan karakteristik unik pada wajah untuk tujuan identifikasi *personal* (Juhong & Pintavirooj, 2017). Sebagaimana diketahui pentingnya pengenalan biometrik pada tubuh manusia, pengenalan wajah telah banyak diimplementasikan dalam berbagai bidang khususnya bidang keamanan (Liu, 2015). Selain itu pengenalan wajah juga dapat diterapkan dalam berbagai bidang seperti sistem absensi, akses kontrol, keamanan bandara, dan sebagainya (Juhong & Pintavirooj, 2017).

2.4 Jaringan Syaraf Tiruan (*Neural Networks*)

Jaringan syaraf tiruan (*neural networks*) merupakan suatu metode dalam *machine learning* dimana bertujuan memodelkan hubungan kompleks antara *input* dan *output* untuk menemukan pola-pola data (Goodfellow et al., 2016).

Jaringan syaraf tiruan menggunakan istilah yang mirip dengan jaringan syaraf biologis. Neuron dalam jaringan syaraf tiruan disebut *perceptron. Perceptron* ditemukan oleh Frank Rosenblatt pada tahun 1950 dan 1960 (Nielsen, 2015). Cara kerja *perceptron* adalah dengan menerima beberapa input dan menghasilkan satu nilai output seperti ditunjukkan Gambar II-1.



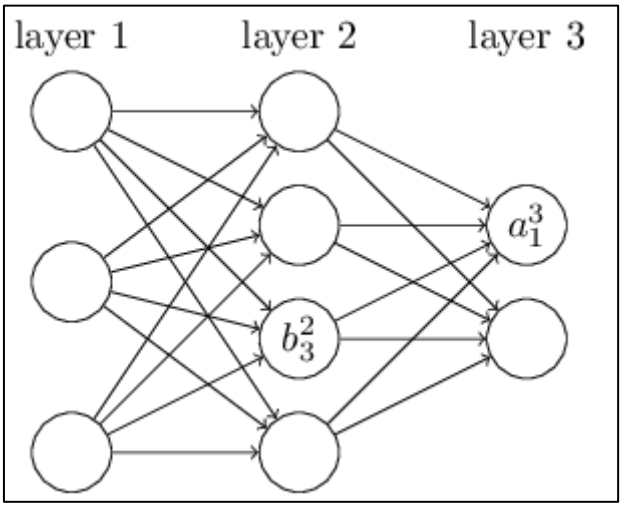
Gambar II-1. Cara kerja *perceptron* (Nielsen, 2015).

Pada Gambar II-1 *perceptron* menerima tiga input *x1*, *x2*, dan *x3*. Rosenblatt membuat sejumlah aturan dalam memproses *input* untuk menghasilkan nilai *output* yaitu menggunakan bobot (*weight*) dinotasikan sebagai *w*. Bobot adalah bilangan *real* yang mengekspresikan besarnya pengaruh *input* terhadap *output.* Selain bobot, terdapat juga nilai yang menentukan *output* suatu *perceptron* yaitu bias yang dinotasikan sebagai *b* (Nielsen, 2015).

Jaringan syaraf tiruan memiliki istilah yaitu kedalaman lapisan *layer* dari suatu jaringan. Kedalaman lapisan *layer* tersebut dapat diukur berdasarkan banyaknya lapisan *perceptron* yang terdapat pada jaringan (Goodfellow et al.,

2016). Jaringan syaraf tiruan dikatakan sebuah ***Multilayer Perceptron*** jika terdapat lebih dari dua lapisan *perceptron* pada jaringan tersebut.

Berdasarkan penjelasan diatas dapat digambarkan bentuk dari jaringan syaraf tiruan adalah seperti Gambar II-2 berikut ini.



Gambar II-2. Bentuk dari jaringan syaraf tiruan (Nielsen, 2015).

Pada Gambar II-2 dapat dilihat terdapat tiga lapisan *perceptron.* Lapisan *perceptron* pada *layer* pertama disebut *input layer* dan lapisan terakhir disebut *output layer.* Lapisan yang berada diantara *input layer* dan *output layer* disebut *hidden layer* atau *hidden unit,* jumlah *hidden layer* dapat disesuaikan sesuai kebutuhan dan seberapa dalam jaringan yang akan dimodelkan. Berdasarkan pada Gambar II-2 dapat dibuat persamaan sebagai berikut.

$$a_j^l = \sigma\left(\sum_k w_{jk}^l \, a_k^{l-1} + b\right) \qquad (\text{II} - 1)$$

Keterangan:

a_j^l = output perceptron ke-j pada layer l

σ = fungsi aktivasi

w_{jk}^l = bobot ke-k dari *perceptron* ke-j pada layer l

a_k^{l-1} = output ke-k pada layer (l-1)

b = bias ke-j pada layer l

Secara teknis operasi dalam jaringan syaraf tiruan menggunakan operasi matriks yang disebut *hadamart product* (Nielsen, 2015). Ekspresi pada persamaan II-1 tidak dalam bentuk operasi matriks. Untuk menuliskan persamaan II-1 menjadi bentuk ekspresi operasi matriks bobot pada setial *layer* l didefinisikan sebagai matriks w^l komponen pada matriks w^l adalah nilai dari baris ke-j dan kolom ke-k pada w_{jk}^l. Lalu untuk bias, direpresentasikan sebagai vektor bias b^l dimana tiap komponennya adalah nilai dari b_j^l, dimana j adalah perceptron ke-j pada setiap layer. Selanjutnya adalah *output* yang direpresentasikan sebagai vektor a^l yang mana tiap komponennya adalah adalah a_j^l. Langkah terakhir adalah menuliskan persamaan II-1 menjadi bentuk ekspresi operasi matriks seperti pada persamaan II-2 berikut.

$$a^l = \sigma(w^l a^{l-1} + b) \qquad \text{II} - 2$$

Keterangan:

a^l = vektor *output* pada *layer l*

σ = fungsi aktivasi

w^l = matriks bobot pada *layer l*

b^l = vektor bias pada *layer l*

2.5 *Backpropagation*

Algoritma *backpropagation* pertama kali diperkenalkan pada tahun 1970, namun belum terlalu menarik perhatian sampai terdapat penelitian populer pada tahun 1986 oleh David Rumelhart, Geoffrey Hinton, dan Ronald Williams. Penelitian tersebut mengemukakan bahwa *backpropagation* jauh lebih cepat dibanding algoritma-algoritma pembelajaran lainnya pada masa itu (Nielsen, 2015).

Tujuan utama dari algoritma *backpropagation* adalah mencari *partial derivative* dari fungsi *cost* C dinotasikan sebagai $\frac{\partial C}{\partial w}$, ekspresi tersebut menunjukan besarnya pengaruh perubahan bobot *w* (atau bias *b*) terhadap besaran nilai *cost.* Langkah-langkah dalam algoritma *backpropagation* dijelaskan pada sub bab 2.5.1 sampai dengan sub bab 2.5.5.

2.5.1 Feedfordward

Feedforward pada jaringan syaraf tiruan telah dijelaskan pada sub bab 2.4 dimana pada proses *feedforward* dilakukan perhitungan output dari jaringan syaraf tiruan dengan persamaan sebagai berikut.

$$a^l = \sigma(w^l a^{l-1} + b) \qquad \text{II} - 3$$

Keterangan:

a^l = vektor *output* pada *layer l*

σ = fungsi aktivasi

w^l = matriks bobot pada *layer l*

b^l = vektor bias pada *layer l*

2.5.2 Menghitung Nilai *Error* pada *Layer* Terakhir

Error pada layer terakhir dinotasikan sebagai δ^L yang mana diperoleh melalui persamaan berikut ini.

$$\delta^L = \nabla_a C \odot \sigma'(a^L) \qquad \text{II} - 4$$

Keterangan:

δ^L = nilai *error* pada *layer* terakhir

$\nabla_a C$ = vektor yang mana elemennya adalah setiap $\frac{\partial C}{\partial a_j^L}$

σ' = fungsi aktivasi

a^L = vektor output pada layer terakhir

2.5.3 Menghitung Nilai *Error* pada Setiap *Layer*

Setelah mendapatkan nilai *error* dari *layer* terakhir maka nilai *error* juga dihitung untuk setiap layer pada jaringan, proses inilah yang dinamakan dengan *backwardpass.* Nilai *error* pada setiap *layer* dinotasikan sebagai δ^l. Untuk menghitung nilai δ^l digunakan persamaan berikut.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(a^l) \qquad \text{II} - 5$$

Keterangan:

δ^l = nilai *error* pada setiap *layer l*

δ^{l+1} = nilai *error* pada setiap *layer l+1*

w^{l+1} = matriks bobot pada *layer l+1*

σ' = fungsi aktivasi

a^l = vektor output pada *layer l*

2.5.4 Menghitung *Partial Derivative* untuk Fungsi *Cost* terhadap Bobot

Proses utama dalam backpropagation untuk menghitung pengaruh perubahan bobot *w* terhadap perubahan nilai *cost*.

$$\frac{\partial C}{\partial w_{jk}} = a_k^{l-1} \delta_j^l \qquad \text{II} - 6$$

Keterangan:

$\frac{\partial C}{\partial w_{jk}}$ = *partial derivative* fungsi *cost* terhadap bobot

a_k^{l-1} = output perceptron ke-k pada *layer l-1*

δ_j^l = nilai error ke-j pada *layer l*

2.5.5 Ubah Nilai Bobot dan Bias

Sebuah aturan dalam mengubah nilai bobot *w* dan bias *b* menggunakan persamaan.

$$w^l = w^l - \frac{\eta}{m}\frac{\partial C}{\partial w} \qquad \text{II} - 7$$

$$b^l = b^l - \frac{\eta}{m}\frac{\partial C}{\partial b} \qquad \text{II} - 8$$

Keterangan:

w^l = matriks bobot pada *layer l*

η = learning rate

m = banyak data

$\frac{\partial C}{\partial w}$ = partial derivative fungsi cost terhadap bobot

$\frac{\partial C}{\partial b}$ = partial derivative fungsi cost terhadap bias

2.6 Convolutional Neural Networks

Convolutional neural networks, atau juga dikenal dengan *convolutional networks*, adalah salah satu jenis *neural networks* untuk memproses data yang bersifat bidang (*grid-like topology*). Contohnya termasuk data citra yang mana memiliki dua dimensi bidang piksel. Berdasarkan penamaannya, *Convolutional neural networks* mengindikasikan bahwa *network* (jaringan) mengimplementasikan operasi matematika yaitu **konvolusi** (*convolution)*. Konvolusi adalah bentuk dari operasi linear. *Convolutional neural networks* adalah sebuah *neural networks* yang menggunakan konvolusi untuk menggantikan perkalian matriks setidaknya satu dalam susunan *layer* (Goodfellow et al., 2016).

2.6.1 Operasi Konvolusi pada *Convolutional Neural Networks*

Pada *convolutional neural networks*, *input* adalah data *array* multi dimensi dan bobot (*w*) pada *neural networks* selanjutnya akan disebut *kernel*, *kernel* adalah *array* multidimensi dimana merupakan parameter yang akan diimplementasikan menggunakan algoritma pembelajaran sepeti *backpropagation*. Kedua *array* multidimensi yaitu *input* dan *kernel* secara umum disebut dengan istilah *tensor*.

Persamaan berikut menjelaskan operasi konvolusi dengan *I* sebagai input dan *K* adalah *kernel* dua dimensi:

$$(I * K)_{(i,j)} = \sum_{m=0}^{k1-1} \sum_{n=0}^{k2-1} I(i-m, j-n)K(m,n) \qquad \text{II-9}$$

Persamaan diatas juga berlaku komutatif sehingga dapat ditulis sebagai berikut:

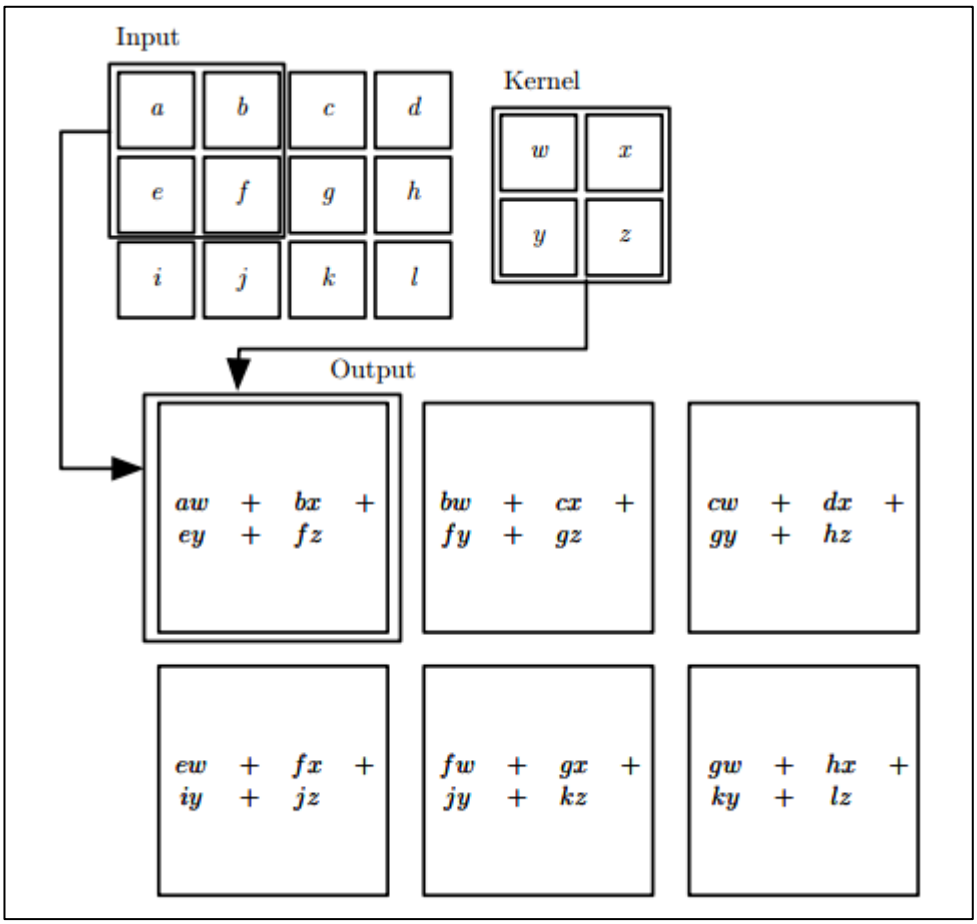
$$(I * K)_{(i,j)} = \sum_{m=0}^{k1-1} \sum_{n=0}^{k2-1} I(m,n)K(i-m, j-n) \qquad \text{II-10}$$

Bentuk komutatif didapat dikarenakan dalam operasi konvolusi *kernel K* akan dibalik (*flipped kernel*) berdasarkan *input I*, dimana jika *m* terus bertambah *index* pada *input* akan bertambah, tetapi *index* pada *kernel* terus berkurang.

Implementasi operasi konvolusi pada saat membangun kode program, atau secara teknis sering dilakukan dengan melakukan konvolusi tanpa membalik kernel. Operasi tersebut disebut *cross-correlation*, seperti dapat dilihat pada persamaan berikut:

$$(I * K)_{(i,j)} = \sum_{m=0}^{k1-1} \sum_{n=0}^{k2-1} I(i+m, j+n)K(m,n) \qquad \text{II-11}$$

Operasi konvolusi tanpa membalik *kernel* dapat dilihat pada Gambar II-3, diaplikasikan pada *tensor* dua dimensi:

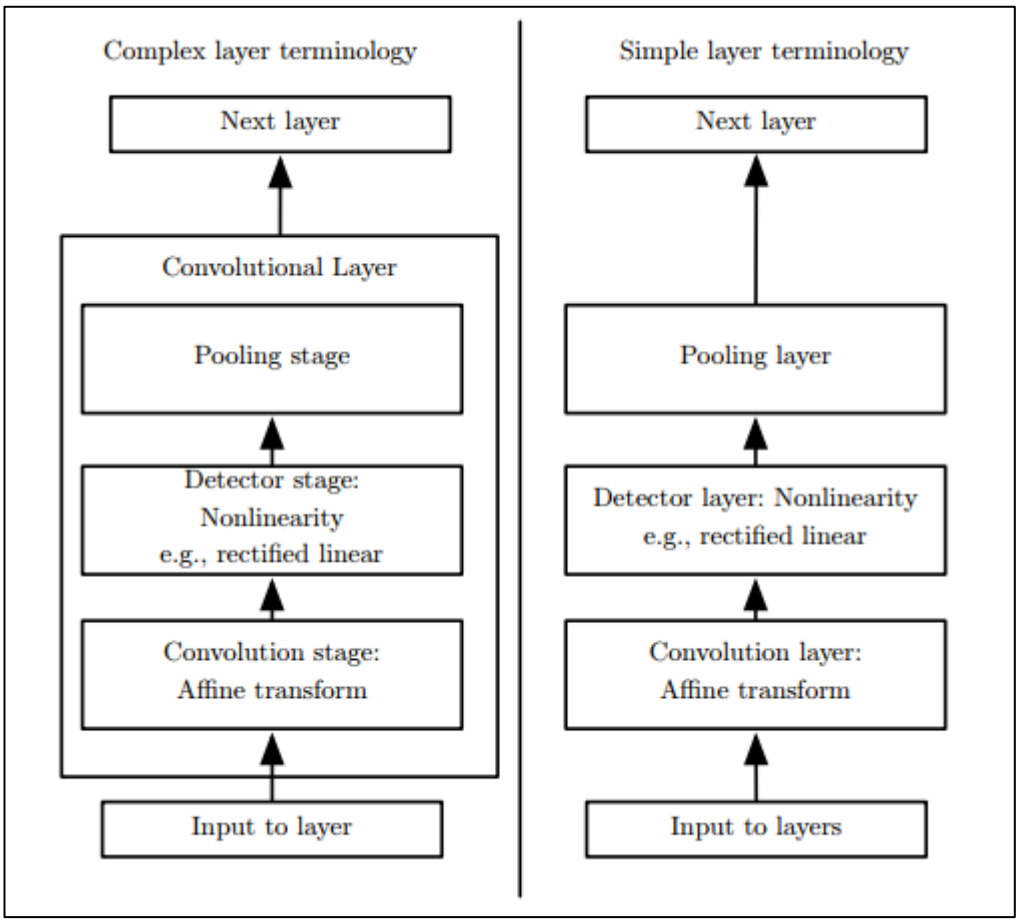


Gambar II-3. Contoh operasi konvolusi tanpa membalik *kernel* (Goodfellow et al., 2016).

2.6.2 *Pooling*

Sebuah ciri khusus *layer* pada *convolutional neural networks* terbagi dalam tiga tahapan. Pada tahap pertama dilakukan operasi konvolusi untuk menghasilkan himpunan nilai dari fungsi aktivasi linear . Pada tahap kedua setiap nilai dari himpunan aktivasi linear akan masuk dalam fungsi aktivasi nonlinear seperti fungsi *rectified linear activation*, tahapan ini sering disebut *detector stage*. Pada tahap

ketiga digunakan fungsi *pooling* untuk mengubah output pada layer (Goodfellow et al., 2016).



Gambar II-4. Tahapan pada *convolution neural networks* (Goodfellow et al., 2016).

Pooling berfungsi menggantikan *output* nilai pada beberapa lokasi dengan *summary statistic of the nearby outputs. Pooling* dibutuhkan untuk mengatasi jika terdapat perbedaan translasi pada input. Dengan mengatasi perbedaan translasi pada input berarti jika mengubah sedikit nilai pada input maka nilai yang berhasil didapat dari proses *pooling* tidak berubah.

2.6.3 Arsitektur *Convolutional Neural Networks*

Arsitektur *convolutional neural networks* adalah struktur layer pada *convolutional neural netowrks* meliputi kedalaman jaringan, ukuran dimensi tiap layer, dan susunan layer. Terdapat banyak arsitektur yang bisa digunakan dalam implementasi *convolutional neural networks* seperti VGG16, Lenet-5, ResNet, dan sebagainya.

2.7 *Confusion Matrix*

Confusion matrix adalah metode yang merangkum kinerja pengklasifikasian dari beberapa data uji. Metode ini terdiri dari matriks dua dimensi dimana indeks pertama untuk kelas sebenarnya dari suatu objek dan dilainnya untuk kelas dari pengklasifikasian. Contoh metode ini, jika pada *dataset* hanya terdapat dua kelas yaitu positif dan negatif, maka ada beberapa istilah yang biasa digunakan sebagai notasi pengukuran.

- *True Positive*(T+) : data dengan kelas positif yang terklasifikasi positif
- *True Negative*(T-) : data dengan kelas negatif yang terklasifikasi negatif
- *False Positive*(F+) : data dengan kelas negatif yang terklasifikasi positif
- *False Negative*(F-) : data dengan kelas positif yang terklasifikasi negatif

Keluaran dari *confusion matrix* adalah *accuracy, precision, recall,* dan *f-measure* dengan persamaan sebagai berikut

- *Accuracy*: seberapa akurat metode klasifikasi dalam menentukan kelas atau label pada data.

$$Accuracy = \frac{(T+) + (T-)}{Total\ Citra} \quad \text{(II-12)}$$

- *Precision*: nilai fraksi tiap kelas yang diklasifikasi dengan benar.

$$Precision = \frac{(T+)}{(T+) + (F+)} \quad \text{(II-13)}$$

$$Precision = \frac{(T-)}{(T-) + (F-)} \quad \text{(II-14)}$$

- *Recall*: fraksi jumlah kelas atau label yang berhasil didapatkan dibagi dengan jumlah kelas yang seharusnya didapatkan.

$$Recall = \frac{(T+)}{(T+) + (F-)} \quad \text{(II-15)}$$

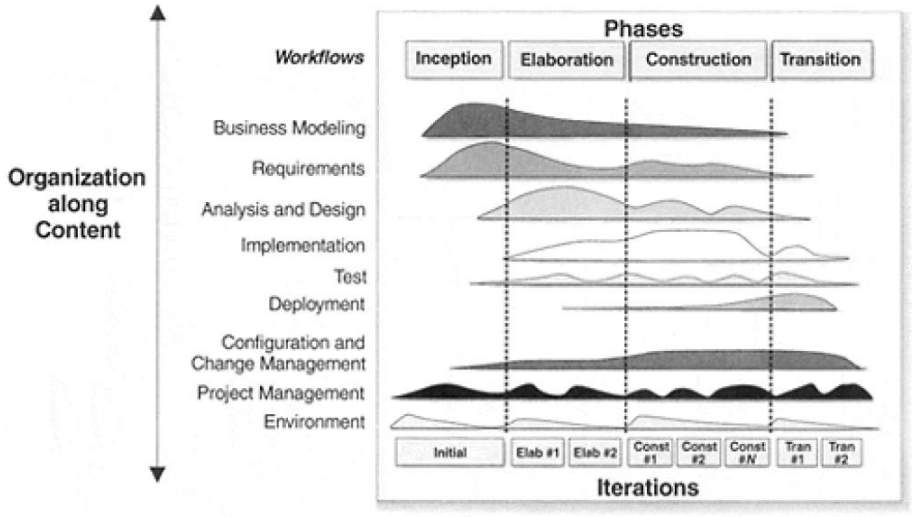
$$Recall = \frac{(T-)}{(T-) + (F+)} \quad \text{(II-16)}$$

- *F-Measure*: fungsi *harmonic mean* dari *precision* dan *recall.*

$$F - Measure = 2\frac{Precision\ .Recall}{Precision + Recall} \quad \text{(II-17)}$$

2.8 *Rational Unified Process*

Rational Unified Process (RUP) adalah proses rekayasa perangkat lunak. RUP memberikan pendekatan disiplin untuk menugaskan tugas dan tanggung jawab dalam organisasi pengembangan. Tujuannya adalah untuk memastikan produksi perangkat lunak berkualitas tinggi yang memenuhi kebutuhan pengguna akhir dalam jadwal dan anggaran yang dapat diprediksi (Kruchten, 2000). Gambar di bawah ini menunjukkan secara keseluruhan arsitektur RUP.



Gambar II-5. Arsitektur RUP (Kruchten, 2000).

Pada gambar di atas, diketahui bahwa RUP memiliki 2 dimensi struktur proses, yaitu:

1. Dimensi horizontal menggambarkan waktu dan menunjukkan aspek siklus hidup dari proses yang berlangsung. Fase-fase tersebut dapat bersifat iteratif sesuai dengan kebutuhan perangkat lunak yang akan dikembangkan. Fase-fase tersebut adalah fase insepsi, elaborasi, konstruksi, dan transisi.
2. Dimensi vertikal menggambarkan alur kerja proses inti, yang terdiri dari *Business Modelling, Requirements, Analysis & Design, Implementation, Test,* dan *Deployment*.

2.9 Penelitian Lain yang Relevan

Pada bagian ini memuat landasan teori dan beberapa penelitian yang telah dilakukan oleh peneliti sebelumnya. Hal ini dibuat untuk memperkuat penalaran

dan rasionalitas keterlibatan sejumlah variable pada penelitian ini. Selain itu juga difungsikan sebagai pendapat ilmiah yang dipadukan dengan hasil kajian pustaka untuk membangun kerangka berpikir peneliti dalam kaitannya dengan masalah yang sedang diteliti.

2.9.1 Kewen Yan, Shaohui Huang, Yaoxian Song, Wei Liu, Neng Fan: *Face Recognition Based on Convolutional Neural Network*, Chinese Control Conference, 2017.

Pada penelitian yang berjudul *Face Recognition Based on Convolutional Neural Network* (Yan et al., 2017) dibahas mengenai pengenalan wajah atau *face recognition* menggunakan *convolutional neural network.* Arsitektur yang digunakan terdiri dari tiga convolution layer, dua pooling layer, dua full-connected layer dan satu Softmax regression layer. Menggunakan dua dataset yaitu ORL dan AR penelitian ini mendapatkan hasil akurasi 99.78 % pada AR dataset dan 99.82% pada ORL dataset.

2.9.2 A. Lebedev, V. Khryashchev, A. Priorov, O. Stepanova: *Face Verification Based on Convolutional Neural Network and Deep Learning*, IEEE East-West Design and Test Symposium, 2017.

Pada penelitian yang berjudul *Face Verification Based on Convolutional Neural Network and Deep Learning* (Lebedev et al., 2017) dibahas mengenai verifikasi wajah menggunakan *convolutional neural network*. Arsitektur yang digunakan adalah VGG16 untuk memverifikasi wajah. Penelitian ini menggunakan dataset LFW dan memperoleh hasil akurasi sebesar 90.6%.

2.10 Kesimpulan

Bidang *computer vision* dapat membantu dalam memperkuat sistem keamanan, seperti penggunaan pola sidik jari, sensor retina, dan pengenalan wajah. Dalam penelitian ini, penulis mencoba mengembangkan sistem pengenalan wajah manusia dilakukan dengan menggunakan metode *convolutional neural network.*

BAB III

METODOLOGI PENELITIAN

3.1 Pendahuluan

Pada bab ini dibahas mengenai tahapan yang dilaksanakan pada penelitian ini. Masing-masing rangacangan tahapan penelitian dideskripsikan dengan rinci dengan mengacu pada suatu kerangka kerja. Di akhir bab ini berisi perancangan manajemen proyek pada pelaksanaan penelitian.

3.2 Unit Penelitian

Unit penelitian yang digunakan dalam penelitian ini adalah data wajah pada situs https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html yang disebut *The Database of Faces (AT&T database)*. Lalu untuk pengujian dilakukan pada data citra wajah mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.

3.3 Pengumpulan Data

Pada sub bab ini dijelaskan mengenai data yang akan digunakan pada penelitian.

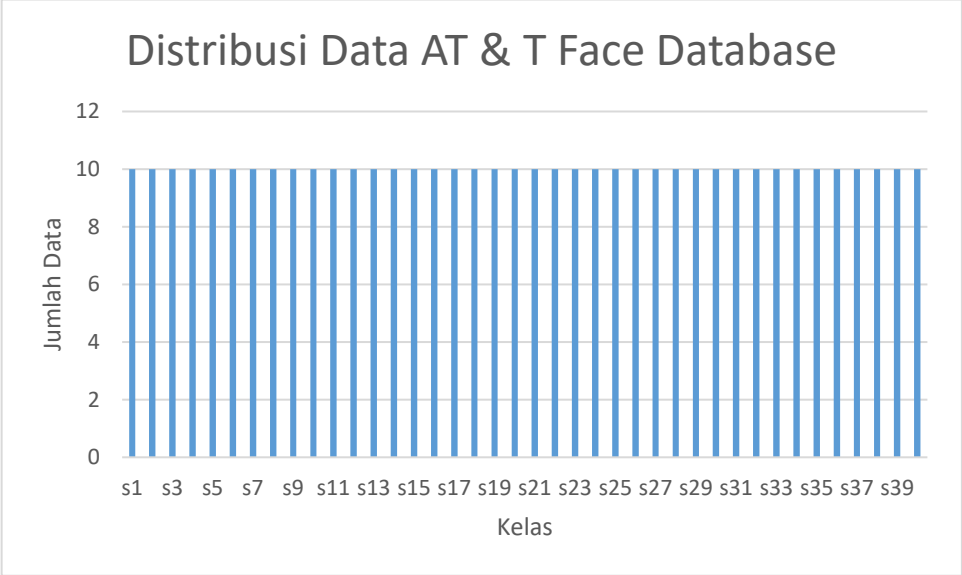
3.3.1 Jenis dan Sumber Data

Data yang digunakan sebagai obyek penelitian adalah jenis data sekunder untuk The Database of Faces (AT&T database) dan data primer pada data mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.

1. AT & T Face Database

The Database of Faces (AT&T database) berisi serangkaian gambar wajah yang diambil antara April 1992 dan April 1994 di lab. Ada 10 gambar berbeda dari masing-masing 40 kelas yang berbeda. Untuk beberapa kelas, gambar diambil pada waktu yang berbeda, memvariasikan pencahayaan, ekspresi wajah (mata terbuka/tertutup, tersenyum/tidak tersenyum) dan detail wajah (kacamata/tanpa kacamata). Gambar III-1 merupakan sampel dari AT & T Face Database dan Gambar III-2 merupakan grafik distribusi data dari AT & T Face Database

Gambar III-1. Sampel Dari AT&T Database

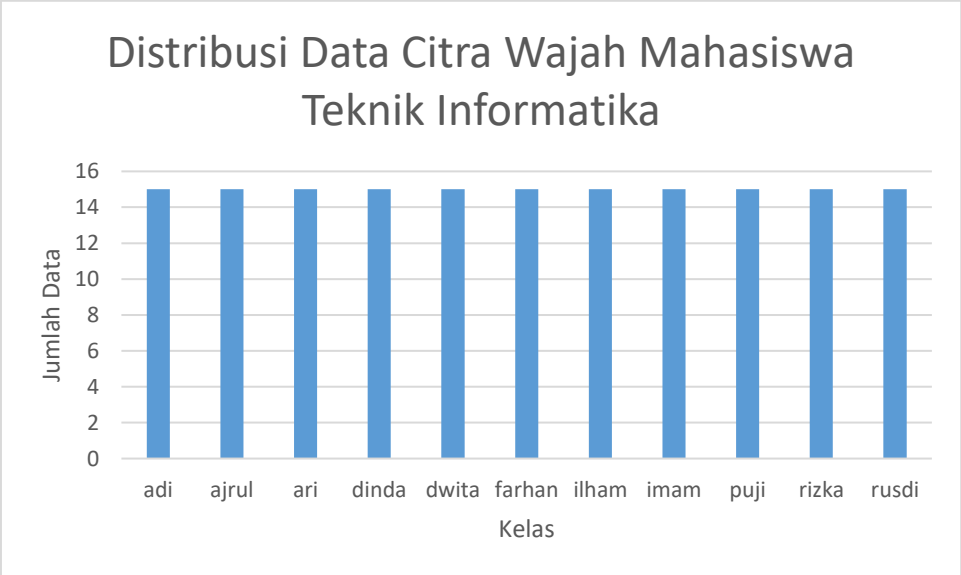


Gambar III-2. Distribusi Data AT&T Database

2. Citra Wajah Mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya

Jenis data primer yang digunakan bersumber dari 11 mahasiswa pada lingkungan kampus Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya. Setiap subjek mahasiswa terdapat 15 citra wajah. Berikut adalah contoh sampel data citra wajah dari beberapa mahasiswa teknik informatika fakultas ilmu komputer universitas sriwijaya dan distribusi data citra mahasiswa Teknik Informatika dapat dilihat pada gambar III-4.

Gambar III-3**.** Sampel Citra wajah mahasiswa teknik informatika



Gambar III-4**.** Distribusi data Citra wajah mahasiswa teknik informatika

3.3.2 Metode Pengumpulan Data

Pada sub bab ini akan dijelaskan mengenai metode yang dilakukan dalam pengumpulan data.

1. ***The Database of Faces (AT&T database)***

Data yang digunakan sebagai obyek penelitian adalah jenis data sekunder untuk *The Database of Faces (AT&T database)*. Data AT&T bebas digunakan dalam penelitian dan publikasi dengan catatan mensitasi beberapa publikasi dari pihak penyedia data.

2. **Citra Wajah Mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.**

Citra wajah dari 11 mahasiswa teknik informatika fakultas ilmu komputer universitas sriwijaya diambil secara langsung menggunakan kamera *webcam* dengan menjalankan salah satu program yang telah dibuat.

3.4 Tahapan Penelitian

Dalam melakukan penelitian pengenalan wajah pada citra *multi-face* menggunakan metode *Convolutional Neural Networks* dilakukan dengan tahapan-tahapan yang dijelaskan pada sub bab 3.4.1 sampai dengan 3.4.6.

3.4.1 Menetapkan Kerangka Kerja / *Framework*

Kerangka kerja pada penelitian ini dibagi menjadi dua tahapan yaitu proses pelatihan (*training*) dan proses pengujian (*testing*). Sebelum melakukan pelatihan dan pengujian terlebih dahulu dibangun arsitektur *Convolutional Neural Network* yang digunakan. Penelitian ini menggunakan dua arsitektur *Convolutional Neural Network.* Arsitektur pertama yaitu satu arsitektur sederhana (*Simple Convolutional*

Neural Network) yang terdiri dari dua *convolutional layer*, satu *pooling layer* menggunakan *maxpooling*, dan dua *fully connected layer*. Setiap *convolutional layer* pada arsitektur *simple convolutional neural network* terdapat fungsi aktivasi non linear yaitu *ReLu*, dan pada dua *fully connected layer* terakhir menggunakan fungsi aktivasi *softmax*. Arsitektur kedua adalah VGG16 yang terdiri dari 13 *convolutional layer, 5 pooling layer, 2 fully connected layer*. Setiap *convolutional layer* pada arsitektur VGG16 terdapat fungsi aktivasi non linear yaitu *ReLu*, dan pada dua *fully connected layer* terakhir menggunakan fungsi aktivasi *softmax.* Pada Tabel III-1 dan Tabel III-2 berikut adalah arsitektur *convolutional neural network* yang digunakan pada penelitian ini.

Tabel III-1. 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

Tabel III-2. Arsitektur VGG16

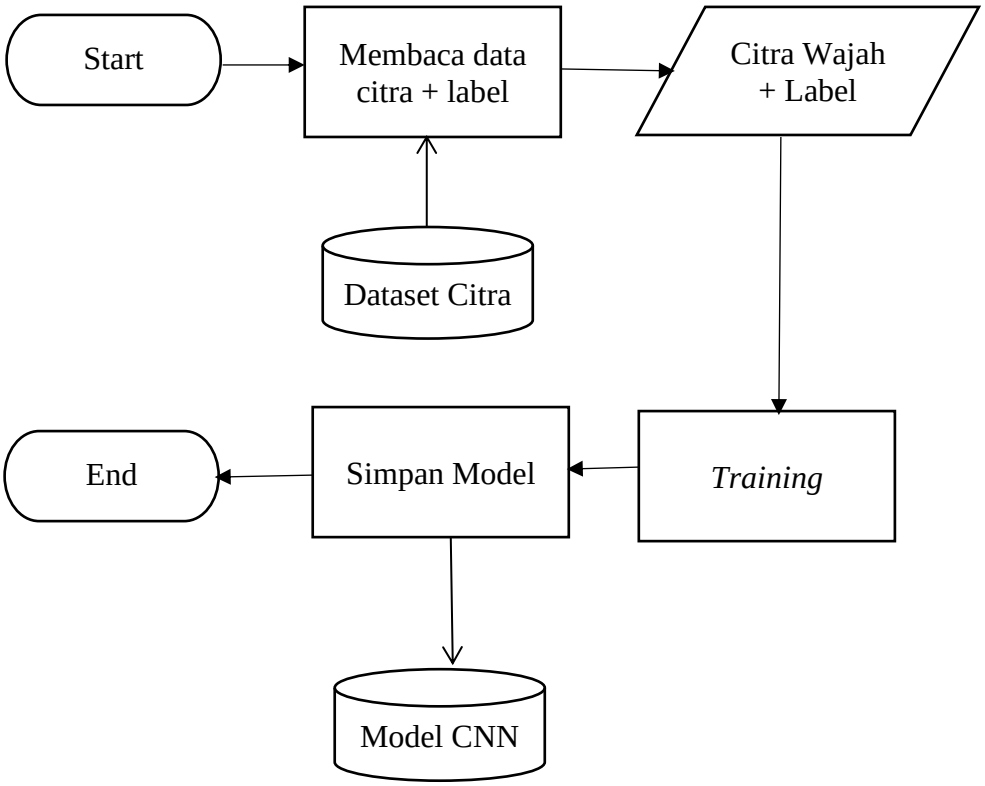
No	Layer	Size		
		Kernel Size	Input	Output
1	Name: - Input Layer		224 x 224 x 3	224 x 224 x 3

2	Name: - Conv2D Activation: ReLu	3 x 3	224 x 224 x 3	224 x 224 x 64
3	Name: - Conv2D Activation: ReLu	3 x 3	224 x 224 x 64	224 x 224 x 64
4	Max Pooling	2 x 2	224 x 224 x 64	112 x 112 x 64
5	Name: - Conv2D Activation: ReLu	3 x 3	112 x 112 x 64	112 x 112 x 128
6	Name: - Conv2D Activation: ReLu	3 x 3	112 x 112 x 128	112 x 112 x 128
7	Name: - Max Pooling	2 x 2	112 x 112 x 128	56 x 56 x 128
8	Name: - Conv2D Activation: ReLu	3 x 3	56 x 56 x 128	56 x 56 x 256
9	Name: - Conv2D Activation: ReLu	3 x 3	56 x 56 x 256	56 x 56 x 256
10	Name: - Conv2D Activation: ReLu	3 x 3	56 x 56 x 256	56 x 56 x 256
11	Name: - Max Pooling		56 x 56 x 256	28 x 28x 256
12	Name: - Conv2D Activation: ReLu	3 x 3	28 x 28x 256	28 x 28 x 512

13	Name: - Conv2D Activation: ReLu	3 x 3	28 x 28 x 512	28 x 28 x 512
14	Name: - Conv2D Activation: ReLu	3 x 3	28 x 28 x 512	28 x 28 x 512
15	Name: - Max Pooling	2 x 2	28 x 28 x 512	14 x 14 x 512
16	Name: - Conv2D Activation: ReLu	3 x 3	14 x 14 x 512	14 x 14 x 512
17	Name: - Conv2D Activation: ReLu	3 x 3	14 x 14 x 512	14 x 14 x 512
18	Name: - Conv2D Activation: ReLu	3 x 3	14 x 14 x 512	14 x 14 x 512
19	Name: - Max Pooling	2 x 2	14 x 14 x 512	7 x 7 x 512
20	Name: -Fully Connected Activation: ReLu		7 x 7 x 512	1 x 25088
21	Name: -Fully Connected Activation: ReLu		1 x 25088	1 x 256
22	Name: -Fully Connected Activation: Softmax		1 x 256	1 x jumlah kelas

1. Pelatihan (*Training*)

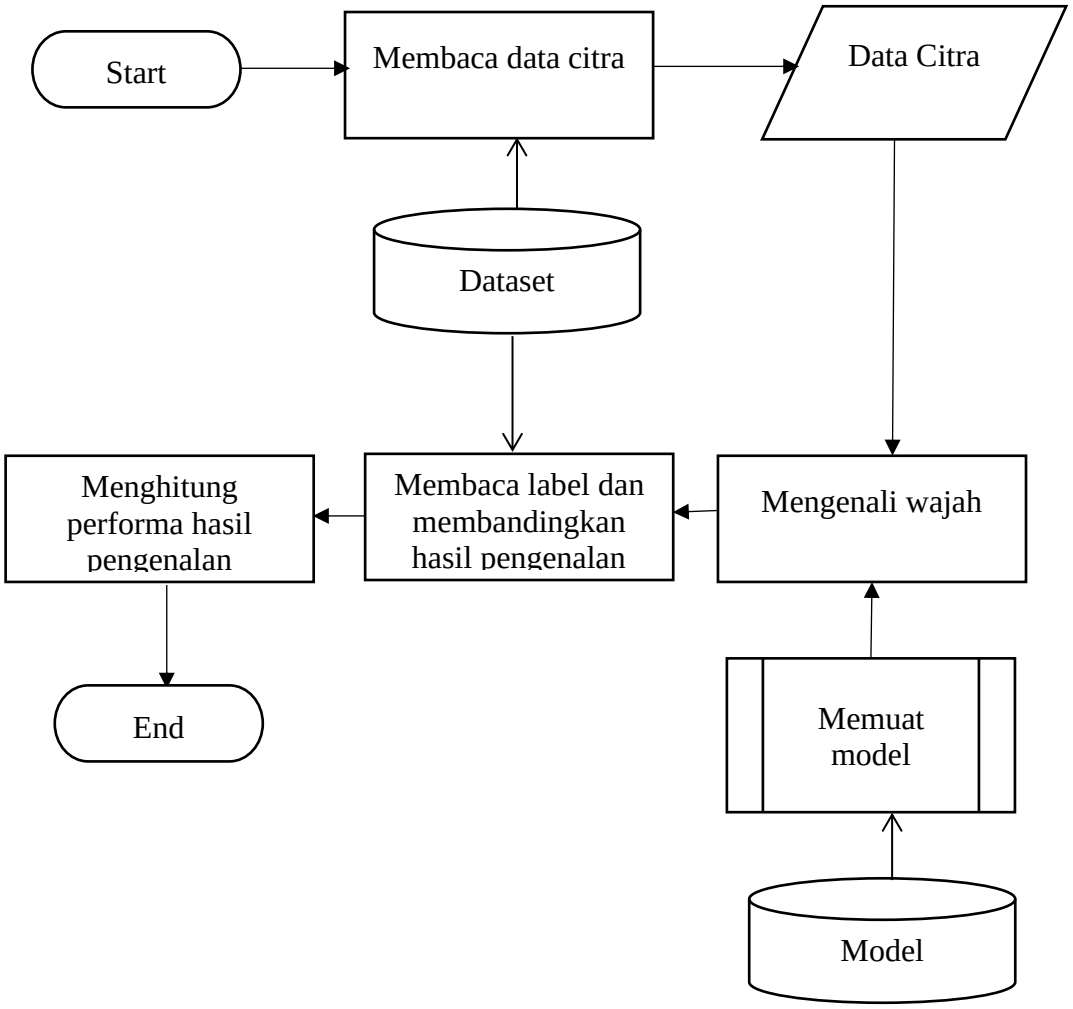
Proses pelatihan (*training*) dilakukan dengan tujuan membangun sebuah model *Convolutional Neural Networks* yang dapat memenuhi kebutuhan dalam masalah pengenalan wajah. Proses membangun sebuah model *Convolutional Neural Networks* pada tahap pelatihan meliputi proses optimisasi bobot dan penurunan *loss* atau nilai *error* pada model *neural network*. Pada proses pelatihan terlebih dahulu digunakan data *single face* dengan tujuan model dapat mengenali atau mengidentifikasi wajah secara individual. Alur proses pelatihan (*training*) secara keseluruhan dapat dilihat pada Gambar III-5.



Gambar III-5. Alur proses pelatihan (*training*)

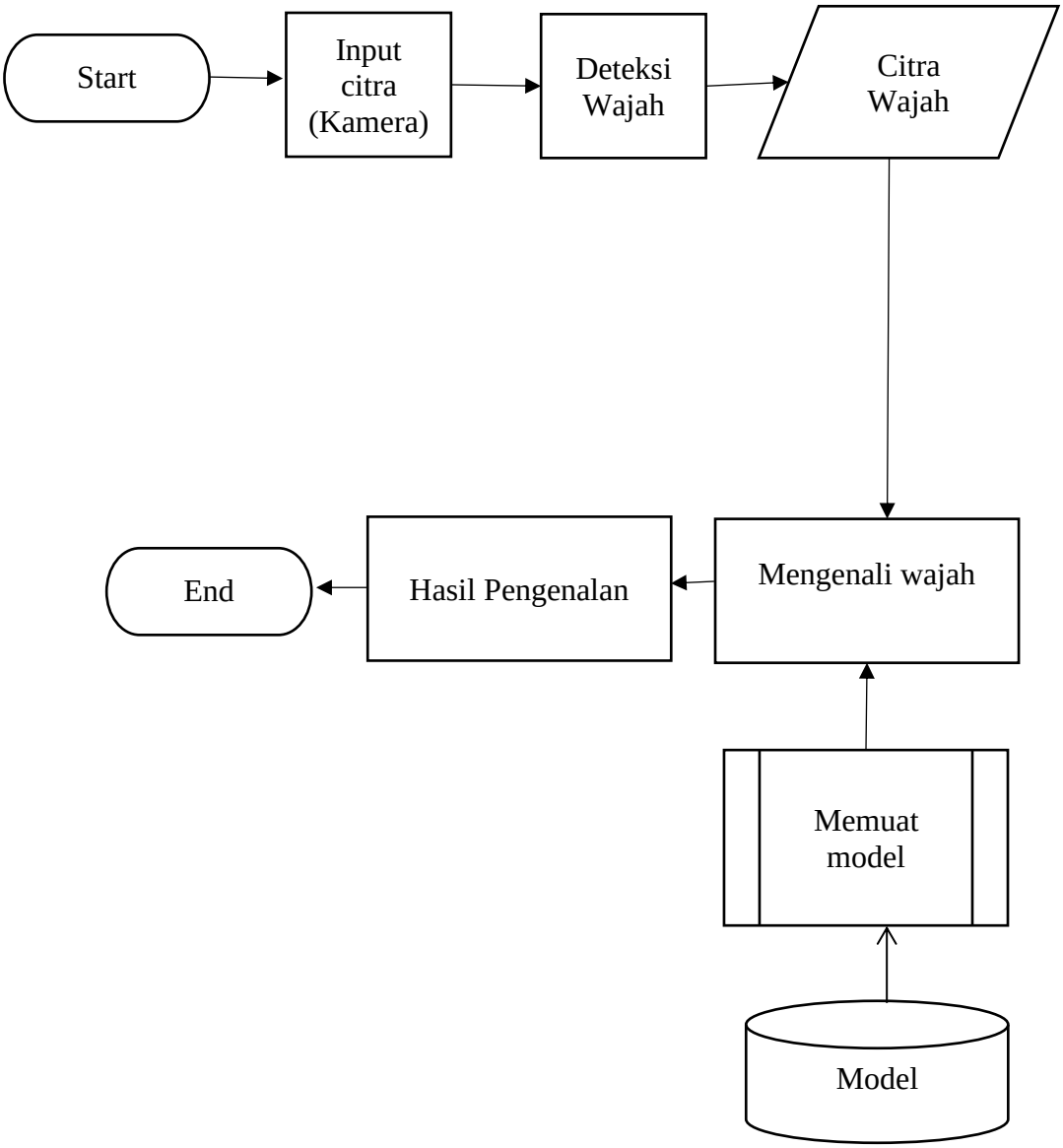
2. Pengujian (Testing)

Pengujian dilakukan dengan menggunakan citra masukan, citra tersebut akan diproses melalui tahapan-tahapan yang dilakukan dalam pengenalan wajah seperti yang dijelaskan pada bab II sub bab 2.1. Proses pengujian dilakukan dalam dua skenario pengujian. Skenario pertama dilakukan tidak secara *realtime* menggunakan citra input untuk mengetahui performa model *convolutional neural network* dalam melakukan pengenalan wajah. Berikut merupakan alur proses pengujian skenario pertama.



Gambar III-6. Alur proses pengujian skenario pertama

Skenario pengujian kedua yaitu secara *realtime*, untuk mengenali setiap individu di dalam citra *muti-face* terlebih dahulu dilakukan proses deteksi wajah untuk mendapatkan *region of interest* (ROI) yaitu citra wajah. Selanjutnya setiap citra wajah yang berhasil dideteksi dilakukan identifikasi untuk mengenali identitas dari wajah tersebut. Pada Gambar III-3 dijelaskan alur proses pengujian.



Gambar III-7. Alur proses pengujian

3.4.2 Menetapkan Kriteria Pengujian

Pembahasan mengenai tahapan ini dijelaskan pada bab IV. Model yang telah dilatih pada proses training selanjutkan diuji menggunakan data uji sebanyak satu bagian dan masing-masing pengujian diukur nilai akurasinya. Jika dataset yang digunakan bersifat tidak seimbang, maka pada tiap pengujian juga diukur nilai *precision, recall, dan f-measure* untuk setiap label.

3.4.3 Menetapkan Format Data Pengujian

Hasil pengujian performa pengenalan wajah digambarkan dalam tabel III-1.

Table III-3. Rancangan Tabel Pengujian Performa Pengenalan Wajah

Label	Precision	Recall	F-Measure	Jumlah
Nama 1				
Nama 2				
Nama 3				
…				
Nama n				
Rata-Rata / Total				

3.4.4 Menentukan Alat yang Digunakan dalam Pelaksanaan Penelitian

Alat yang digunakan dalam penelitian ini berupa kamera yang dapat menangkap citra dengan baik serta perangkat komputer yang mampu menjalanlan perangkat lunak yang dibuat dalam penelitian pengenalan wajah. Mengenai spesifikasi perangkat keras yang digunakan dijelaskan pada sub bab 3.5.3.

3.4.5 Melakukan Pengujian Penelitian

Proses pengujian dilakukan menggunakan kerangka yang telah ditentukan pada sub bab 3.4.1. Penjelasan proses pengujian terdapat pada bab V.

3.4.6 Melakukan Analisa Hasil Pengujian dan Membuat Kesimpulan

Hasil akurasi pengenalan wajah dengan *Convolutional Neural Network*, didapatkan dengan analisa hasil pengujian yang dapat dilihat pada tabel III-1. Jika dataset yang digunakan bersifat tidak seimbang, maka perlu juga melakukan perbandingan pada nilai *precision, recall, dan f-measure*. Setelah hasil analisa pengujian penelitian didapatkan, maka langkah selanjutnya adalah membuat kesimpulan penelitian yang akan dijelaskan pada bab VI.

3.5 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah metode Rational Unified Process (RUP). Fase-fase yang dilakukan terdiri dari fase insepsi, elaborasi, konstruksi, transisi.

3.5.1 Fase Insepsi

Pada tahap *business modelling*, kebutuhan pengguna dan fitur-fitur pada perangkat lunak yang dibutuhkan akan ditentukan. Pada tahap *requirements*, data penelitian berupa citra ekspresi wajah *single-face* dan *multiple face* dikumpulkan. Pada tahap *analysis & design, use-case diagram* akan dibuat. Pada tahap

implementation, kebutuhan pengguna didokumentasikan. Pada tahap *test*, akan dipastikan apakah kebutuhan pengguna dan fitur-fitur perangkat lunak sudah valid.

3.5.2 Fase Elaborasi

Pada tahapan *business modelling*, arsitektur perangkat lunak akan, desain basisdata, dan desain *user interface* akan ditentukan sesuai dengan kebutuhan pengguna. Pada tahap *requirements*, kebutuhan pengguna dilengkapi jika dirasa belum lengkap. Pada tahap *analysis & design*, *activity diagram* dan *sequence diagram* dibuat. Pada tahap *implementation*, dokumentasi yang memuat arsitektur perangkat lunak, desain basisdata, desain *user interface*, *activity diagram*, dan *sequence diagram* akan disusun. Lalu dipastikan seluruhnya sudah valid pada tahap *test*.

3.5.3 Fase Konstruksi

Pada tahap *business modelling*, kelas-kelas yang dibutuhkan oleh perangkat lunak akan ditentukan. Pada tahap *requirements*, ditentukan bahasa pemrograman yang digunakan untuk mengembangkan perangkat lunak. Kebutuhan lain dalam proses pengembangan perangkat lunak juga diidentifikasi, seperti perangkat keras dengan Processor Intel(R) Core(TM) i7-4720HQ CPU @ 2.60 GHz, RAM 4GB, dan Harddisk 1 TB, PlantUML, Apache, MariaDB, Flask, dan Sublime. Pada tahap *analysis & design*, *class diagram* dibuat. Pada tahap *implementation*, dikembangkan perangkat lunak dengan mengimplementasikan kelas-kelas yang telah ditentukan ke kode program dalam bahasa *python*. Lalu dilakukan *unit testing* tahap *test* untuk memastikan kelas-kelas telah terimplementasi dengan benar.

3.5.4 Fase Transisi

Pada tahap *business modelling*, rencana atau skenario pengujian perangkat lunak dibuat. Pada tahap *requirements*, ditentukan *tools* pengujian yang dibutuhkan. *Tools* pengujian adalah perangkat keras yang sama seperti yang dijelaskan pada subbab 3.5.3. Pada tahap *analysis & design*, dibuat tabel skenario. Pada tahap *implementation*, dilakukan pengujian terhadap perangkat lunak berdasarkan skenario atau rencana pengujian. Lalu ditinjau ulang skenario pengujian pada tahap *test*.

3.6 Manajemen Proyek Penelitian

Manajemen proyek penelitian adalah perencanaan aktivitas penelitian dari tahap perumusan masalah sampai dengan pada tahap kesimpulan dari penelitian. Adapun kegiatan-kegiatan yang berlangsung selama penelitian dapat dilihat dalam *Work Breakdown Structure* (WBS) yang tertera pada Tabel III-4.

Tabel III-4. Tabel Penjadwalan Penelitian dalam Bentuk *Work Breakdown Structure* (WBS)

ID	***Task Name***	***Duration***	***Start***	***Finish***	***Predecessors***
	Pengenalan Wajah Secara Realtime Menggunakan *Convolutional Neural Network* pada Citra *Multi Face*	**173 days**	**Tue 11/13/18 8:00 AM**	**Thu 7/11/19 5:00 PM**	
	Menentukan Ruang Lingkup dan Unit Penelitian	**24 days**	**Tue 11/13/18 8:00 AM**	**Fri 12/14/18 5:00 PM**	
T1	Menentukan masalah penelitian	0 days	Tue 11/13/18 8:00 AM	Tue 11/13/18 8:00 AM	
T2	Membuat latar belakang dan rumusan masalah	7 days	Sat 11/17/18 8:00 AM	Mon 11/26/18 5:00 PM	T1
T3	Menentukan tujuan dan manfaat penelitian	4 days	Tue 11/27/18 8:00 AM	Fri 11/30/18 5:00 PM	T2
T4	Menentukan batasan masalah	5 days	Mon 12/3/18 8:00 AM	Fri 12/7/18 5:00 PM	T3
T5	Menentukan unit penelitian	5 days	Mon 12/10/18 8:00 AM	Fri 12/14/18 5:00 PM	T4
M1	Tersedia dokumen hasil tahapan penelitian	1 day	Fri 12/14/18 8:00 AM	Fri 12/14/18 5:00 PM	T5
	Menentukan Dasar Teori yang Berkaitan dengan Penelitian	**18 days**	**Sat 12/15/18 8:00 AM**	**Tue 1/8/19 5:00 PM**	
T6	Mengumpulkan jurnal, paper, dan literatur ilmiah yang berkaitan dengan penelitian	9 days	Sat 12/15/18 8:00 AM	Wed 12/26/18 5:00 PM	T1
T7	Mempelajari metode *Convolutional Neural Networks*	17 days	Sat 12/15/18 8:00 AM	Tue 1/8/19 5:00 PM	T6
M2	Tersedia dokumen hasil tahapan penelitian	1 day	Tue 1/8/19 8:00 AM	Tue 1/8/19 5:00 PM	T7
	Menentukan Kriteria Pengujian	**27 days**	**Wed 1/9/19 8:00 AM**	**Thu 2/14/19 5:00 PM**	
T8	Menentukan proses yang akan menggunakan *Convolutional Neural Networks*	13 days	Wed 1/9/19 8:00 AM	Fri 1/25/19 5:00 PM	T6
T9	Menentukan kriteria pengujian dan penjadwalan	13 days	Sat 1/26/19 8:00 AM	Tue 2/12/19 5:00 PM	T6
M3	Tersedia dokumen hasil tahapan penelitian	2 days	Wed 2/13/19 8:00 AM	Thu 2/14/19 5:00 PM	T8, T9

	Menentukan Alat yang Digunakan untuk Pelaksanaan Penelitian	**92 days**	**Fri 2/15/19 8:00 AM**	**Sat 6/22/19 5:00 PM**	
	Inception	**15 days**	**Fri 2/15/19 8:00 AM**	**Thu 3/7/19 5:00 PM**	
	Business Modelling	**2 days**	**Fri 2/15/19 8:00 AM**	**Mon 2/18/19 5:00 PM**	
T10	Menentukan *user requirements* dan fungsionalitas perangkat lunak	2 days	Fri 2/15/19 8:00 AM	Mon 2/18/19 5:00 PM	T1, T4, T5
	Requirements	**4 days**	**Tue 2/19/19 8:00 AM**	**Fri 2/22/19 5:00 PM**	
T11	Menentukan dataset penelitian	4 days	Tue 2/19/19 8:00 AM	Fri 2/22/19 5:00 PM	T4, T5
	Analysis and Design	**4 days**	**Sat 2/23/19 8:00 AM**	**Wed 2/27/19 5:00 PM**	
T12	Membuat *use case diagram*	4 days	Sat 2/23/19 8:00 AM	Wed 2/27/19 5:00 PM	T10
	Implementation	**3 days**	**Thu 2/28/19 8:00 AM**	**Mon 3/4/19 5:00 PM**	
T13	Membuat dokumentasi	3 days	Thu 2/28/19 8:00 AM	Mon 3/4/19 5:00 PM	T10
	Testing	**3 days**	**Tue 3/5/19 8:00 AM**	**Thu 3/7/19 5:00 PM**	
T14	Memastikan *user requierements* dan fungsionalitas sudah valid	3 days	Tue 3/5/19 8:00 AM	Thu 3/7/19 5:00 PM	T13
	Elaboration	**12 days**	**Fri 3/8/19 8:00 AM**	**Mon 3/25/19 5:00 PM**	
	Business Modelling	**3 days**	**Fri 3/8/19 8:00 AM**	**Tue 3/12/19 5:00 PM**	
T15	Menentukan arsitektur perangkat lunak, desain basis data, dan desain antar muka	3 days	Fri 3/8/19 8:00 AM	Tue 3/12/19 5:00 PM	T14
	Requirements	**3 days**	**Wed 3/13/19 8:00 AM**	**Fri 3/15/19 5:00 PM**	
T16	Melengkapi *user requirements* yang telah didefinisikan di fase *inception*	3 days	Wed 3/13/19 8:00 AM	Fri 3/15/19 5:00 PM	T10
	Analysis & Design	**3 days**	**Sat 3/16/19 8:00 AM**	**Tue 3/19/19 5:00 PM**	
T17	Membuat *activity* dan *sequence diagram*	3 days	Sat 3/16/19 8:00 AM	Tue 3/19/19 5:00 PM	T12, T15
	Implementation	**2 days**	**Wed 3/20/19 8:00 AM**	**Thu 3/21/19 5:00 PM**	

T18	Membuat dokumentasi	2 days	Wed 3/20/19 8:00 AM	Thu 3/21/19 5:00 PM	T17
	Testing	**2 days**	**Fri 3/22/19 8:00 AM**	**Mon 3/25/19 5:00 PM**	
T19	Memastikan arsitektur perangkat lunak, desain basis data, dan desain antar muka sudah valid	2 days	Fri 3/22/19 8:00 AM	Mon 3/25/19 5:00 PM	T18
	Construction	**52 days**	**Tue 3/26/19 8:00 AM**	**Wed 6/5/19 5:00 PM**	
	Business Modelling	**4 days**	**Tue 3/26/19 8:00 AM**	**Fri 3/29/19 5:00 PM**	
T20	Menentukan kelas-kelas pada perangkat lunak	4 days	Tue 3/26/19 8:00 AM	Fri 3/29/19 5:00 PM	T19
	Requirements	**4 days**	**Sat 3/30/19 8:00 AM**	**Wed 4/3/19 5:00 PM**	
T21	Menentukan bahasa pemrograman yang digunakan untuk mengembangkan perangkat lunak	3 days	Sat 3/30/19 8:00 AM	Wed 4/3/19 5:00 PM	T7, T8, T9
T22	Menentukan kebutuhan perangkat keras yang digunakan	3 days	Sat 3/30/19 8:00 AM	Wed 4/3/19 5:00 PM	T21
	Analysis & Design	**3 days**	**Thu 4/4/19 8:00 AM**	**Sun 4/7/19 5:00 PM**	
T23	Membuat *class diagram*	2 days	Thu 4/4/19 8:00 AM	Sun 4/7/19 5:00 PM	T20, T21
	Implementation	**33 days**	**Mon 4/8/19 8:00 AM**	**Wed 5/22/19 5:00 PM**	
T24	Mengimplementasikan kelas-kelas ke dalam kode program	33 days	Mon 4/8/19 8:00 AM	Wed 5/22/19 5:00 PM	T23
	Testing	**10 days**	**Thu 5/23/19 8:00 AM**	**Wed 6/5/19 5:00 PM**	
T25	Melakukan *unit testing*	10 days	Thu 5/23/19 8:00 AM	Wed 6/5/19 5:00 PM	T23
	Transition	**13 days**	**Thu 6/6/19 8:00 AM**	**Sat 6/22/19 5:00 PM**	
	Business Modelling	**2 days**	**Thu 6/6/19 8:00 AM**	**Fri 6/7/19 5:00 PM**	
T26	Membuat rencana atau skenario pengujian	2 days	Thu 6/6/19 8:00 AM	Fri 6/7/19 5:00 PM	T25
	Requirements	**2 days**	**Sat 6/8/19 8:00 AM**	**Mon 6/10/19 5:00 PM**	
T27	Menentukan *tools* pengujian yang diperlukan	2 days	Sat 6/8/19 8:00 AM	Mon 6/10/19 5:00 PM	T25

	Analysis & Design	**3 days**	**Tue 6/11/19 8:00 AM**	**Thu 6/13/19 5:00 PM**	
T28	Membuat tabel skenario pengujian	3 days	Tue 6/11/19 8:00 AM	Thu 6/13/19 5:00 PM	T26, T27
	Implementation	**3 days**	**Fri 6/14/19 8:00 AM**	**Tue 6/18/19 5:00 PM**	
T29	Melakukan pengujian terhadap perangkat lunak berdasarkan skenario atau rencana pengujian	3 days	Fri 6/14/19 8:00 AM	Tue 6/18/19 5:00 PM	T26, T27
	Testing	**3 days**	**Wed 6/19/19 8:00 AM**	**Fri 6/21/19 5:00 PM**	
T30	Meninjau atau menguji skenario pengujian	3 days	Wed 6/19/19 8:00 AM	Fri 6/21/19 5:00 PM	T26, T27, T28
M4	Tersedia dokumen hasil tahapan penelitian	2 days	Thu 6/20/19 8:00 AM	Sat 6/22/19 5:00 PM	T30
	Melakukan Pengujian Penelitian	**9 days**	**Sun 6/23/19 8:00 AM**	**Wed 7/3/19 5:00 PM**	
T31	Menentukan rancangan hasil penelitian	5 days	Sun 6/23/19 8:00 AM	Thu 6/27/19 5:00 PM	T2, T4
T32	Melakukan pengujian penelitian berdasarkan hasil pengujian perangkat lunak	4 days	Fri 6/28/19 8:00 AM	Wed 7/3/19 5:00 PM	T29
M5	Tersedia dokumen hasil tahapan penelitian	1 day	Wed 7/3/19 8:00 AM	Wed 7/3/19 5:00 PM	T31, T32
	Melakukan Analisa Hasil Pengujian dan Membuat Kesimpulan	**6 days**	**Thu 7/4/19 8:00 AM**	**Thu 7/11/19 5:00 PM**	
T33	Melakukan analisa terhadap hasil pengujian penelitian	3 days	Thu 7/4/19 8:00 AM	Mon 7/8/19 5:00 PM	T32
T34	Membuat kesimpulan dan saran berdasarkan analisa terhadap hasil pengujian	2 days	Tue 7/9/19 8:00 AM	Wed 7/10/19 5:00 PM	T33
M6	Tersedia dokumen hasil tahapan penelitian	1 day	Thu 7/11/19 8:00 AM	Thu 7/11/19 5:00 PM	T34

BAB IV

PENGEMBANGAN PERANGKAT LUNAK

4.1 Pendahuluan

Pada bab ini dijlaskan mengenai pengembangan perangkat lunak menggunakan metode RUP yang memiliki empat fase yaitu insepsi, elaborasi, konstruksi, dan transisi.

4.2 Rational Unified Process

RUP merupakan metode pengembangan perangkat lunak yang digunakan pada pengembangan perangkat lunak ini, yang akan dimulai dengan analisis kebutuhan fungsional maupun non-fungsional sistem, perancangan sistem, implementasi sistem serta pengujian sistem. Semua tahapan pada metode ini akan diuraikan dalam sub bab 4.2.1 sampai 4.2.4

4.2.1 Fase Insepsi

Tahapan pertama dalam metode pengembangan perangkat lunak RUP adalah insepsi yang mana pada tahapan ini dilakukan identifikasi terhadap kebutuhan sistem yang dikembangkan. Aktivitas yang dilakukan antara lain yaitu analisis awal sistem, identifikasi dan spesifikasi kebutuhan sebelum adanya perangkat lunak, perumusan kebutuhan pengujian serta pemodelan diagram use case.

4.2.1.1 Pemodelan Bisnis

Dalam mengembangkan perangkat lunak pada Tugas Akhir ini digunakan bahasa pemrograman python yang berbasis *web application*. Perangkat lunak yang dibangun adalah perangkat lunak untuk mengenali wajah pada citra multi face secara *realtime*. Data masukkan dari perangkat lunak ini adalah citra wajah dari input berupa kamera. Output yang dihasilkan dari perangkat lunak ini yaitu wajah dengan label nama yang benar serta *respond time* antara proses input sampai proses pengenalan wajah.

4.2.1.2 Kebutuhan Sistem

Adapun fitur – fitur yang terdapat dalam pengembangan perangkat lunak ini adalah sebagai berikut:

1. Training

Berfungsi untuk melakukan melakukan proses pelatihan untuk membuat model pengenalan wajah menggunakan *convolutional neural networks*.

2. Face Recognition

Berfungsi untuk melakukan pengenalan wajah melalui citra input menggunakan kamera.

4.2.1.3 Analisis dan Desain

Dalam hal ini dilakukan analisis kebutuhan perangkat lunak dan desain dari perangkat lunak itu sendiri.

1. Analisis Kebutuhan Perangkat Lunak

Suatu perangkat lunak 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 IV-1 menjelaskan mengenai kebutuhan fungsional dari perangkat lunak. Dilanjutkan pada tabel IV-2 menjelaskan mengenai kebutuhan non-fungsional dari perangkat lunak.

Tabel IV-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 IV-2. Kebutuhan Non Fungsional

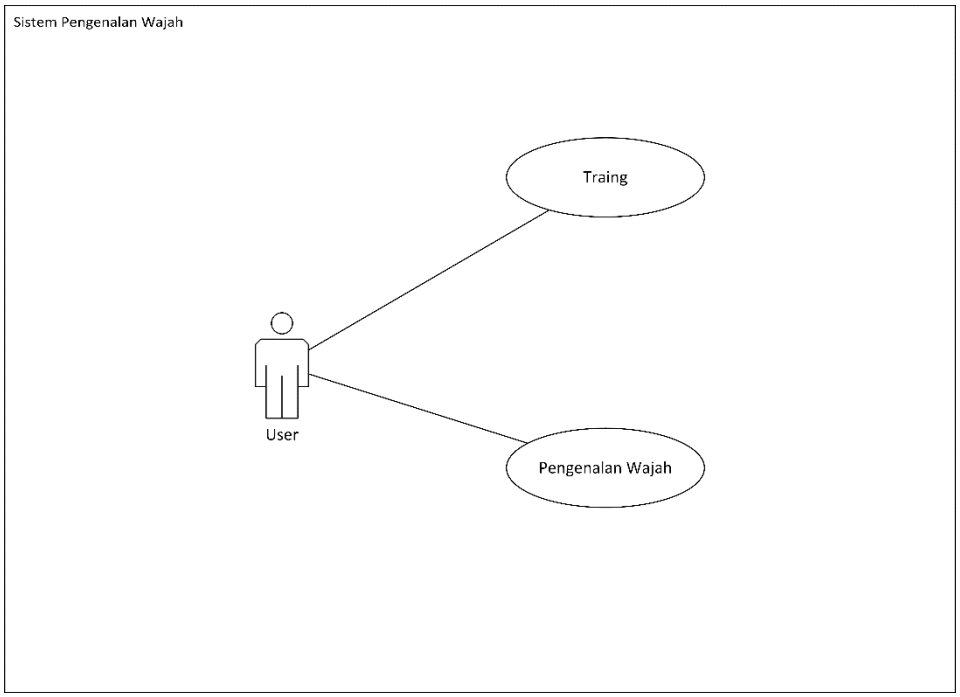
ID	Kebutuhan Non Fungsional	Penjelasan
KNF-01	*Availability*	Perangkat lunak dapat beroperasi selama 24 jam
KNF-02	*Reliability*	Perangkat lunak bersifat *offline*
KNF-03	Ergonomi	Tampilan perangkat lunak dibuat *user friendly*
KNF-04	*Portability*	Dijalankan menggunakan *OS Windows*
KNF-05	*Memory*	-
KNF-06	*Response time*	Dapat melakukan verifikasi wajah dalam waktu ≤ 1 detik
KNF-07	*Security*	Perangkat lunak tidak menggunakan *Passwords*
KNF-08	*Language*	*User interface* pada perangkat lunak menggunakan bahasa Inggris

2. Desain Perangkat Lunak

Desain perangkat lunak digambarkan dengan use case diagram dan juga diagram aktifitas.

1. Use case Diagram

Diagram Use Case merupakan suatu kegiatan yang dilakukan oleh actor terhadap perangkat lunak, use case dalam pengembangan perangkat lunak ini dapat dilihat dalam gambar IV–1.



Gambar IV-1. *Use case diagram*

b. Definisi Aktor

Definisi aktor dari use case yang telah didefinisikan di atas dijelaskan dalam tabel IV-3.

Tabel IV-3. Definisi Aktor *Use Case*

No.	Nama	Definisi
1.	User	User merupakan orang yang berinteraksi dengan perangkat lunak pengenalan wajah

c. Definisi *Use Case*

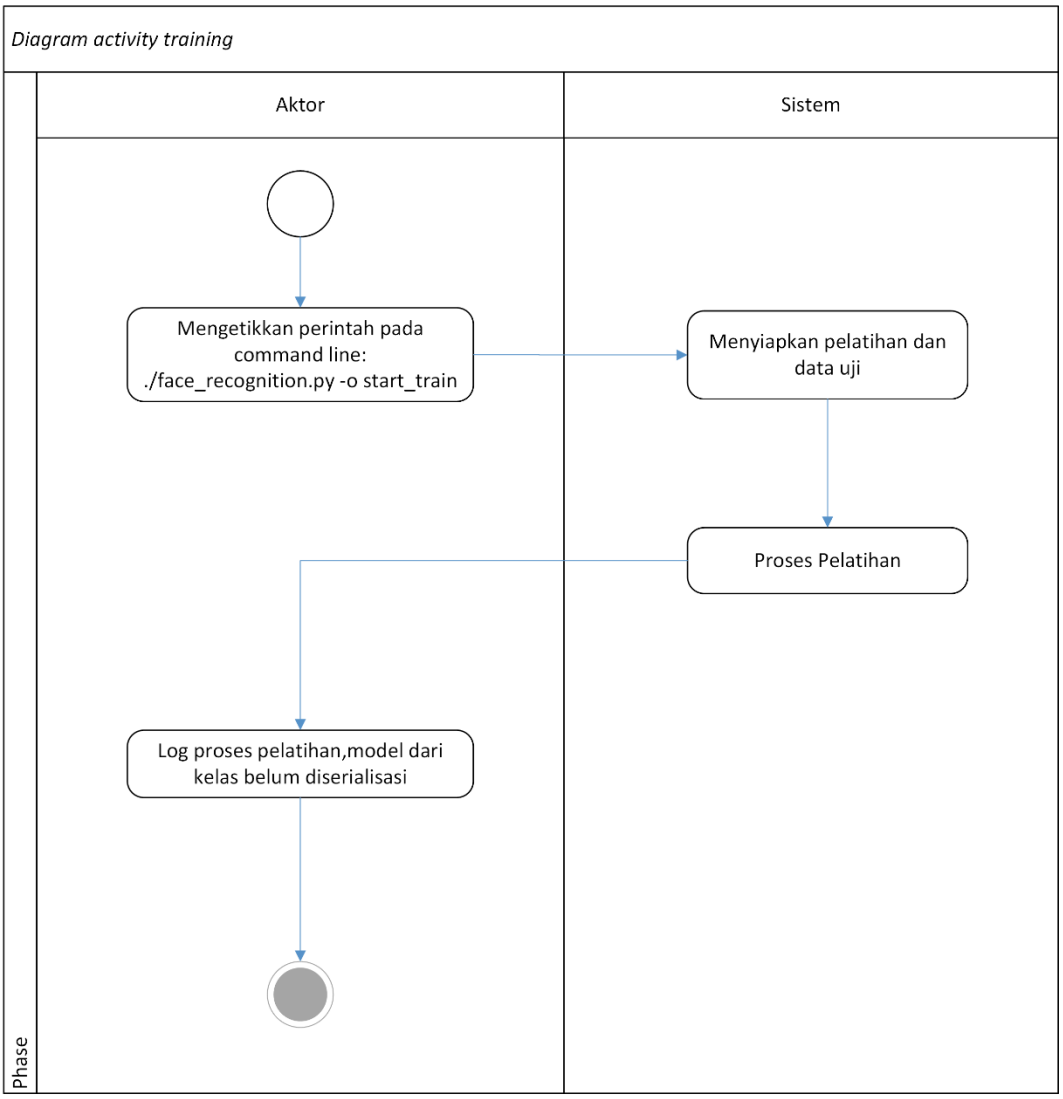
Definisi pada use case diagram di atas diuraikan dalam tabel IV-4.

Tabel IV-4. Definisi *use case*

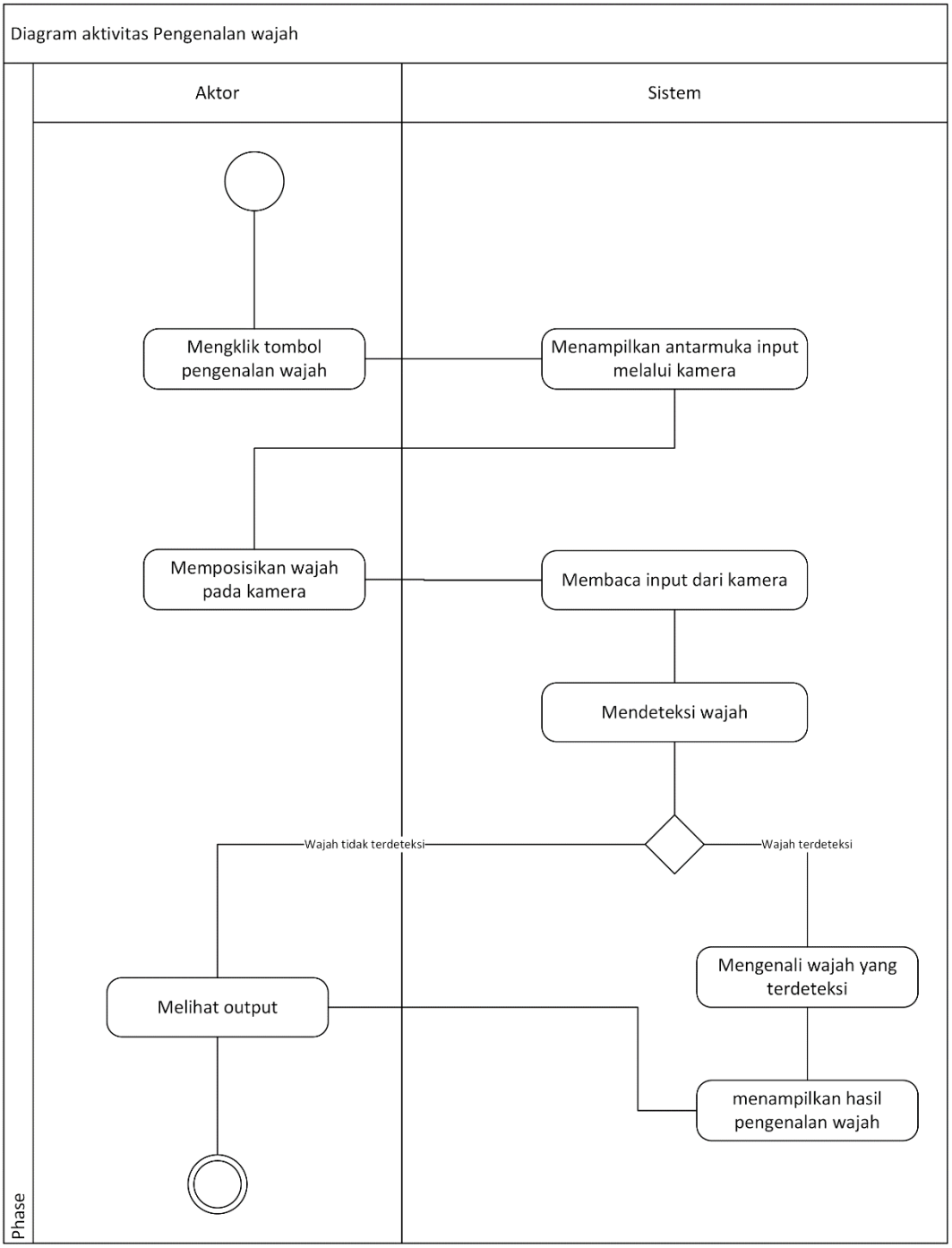
No.	Use Case	Keterangan
1.	Training	Use case ini menggambarkan proses pelatihan untuk membentuk model pengenalan wajah.
2.	Pengenalan wajah	Use case ini menggambarkan proses pengenalan wajah.

2. Diagram Aktivitas

Diagram Aktivitas adalah diagram yang menggambarkan sebuah alur kerja dari sebuah usecase atau model bisnis. Pada gambar IV-2 menunjukkan diagram aktivitas training dan gambar IV-3 menunjukkan diagram aktivitas pengenalan wajah.



Gambar IV-2. Diagram aktifitas training.



Gambar IV-3. Diagram aktivitas pengenalan wajah

4.2.2 Fase Elaborasi

Tahapan kedua dalam metode pengembangan perangkat lunak RUP adalah elaborasi dimana dilakukannya proses identifikasi dalam mengembangkan perangkat lunak melalui beberapa aktivitas, diantaranya perancangan data, perancangan antarmuka, identifikasi kebutuhan, perumusan kebutuhan pengujian, permodelan diagram sequence, dan pembuatan dokumentasi.

4.2.2.1 Pemodelan Bisnis

Pada subbab ini dibahas mengenai berbagai perancangan dari pengembangan perangkat lunak ini, baik perancangan data maupun perancangan antarmuka dari perangkat lunak yang dirancang berdasarkan hasil analisa pada fase insepsi.

1. Perancangan Data

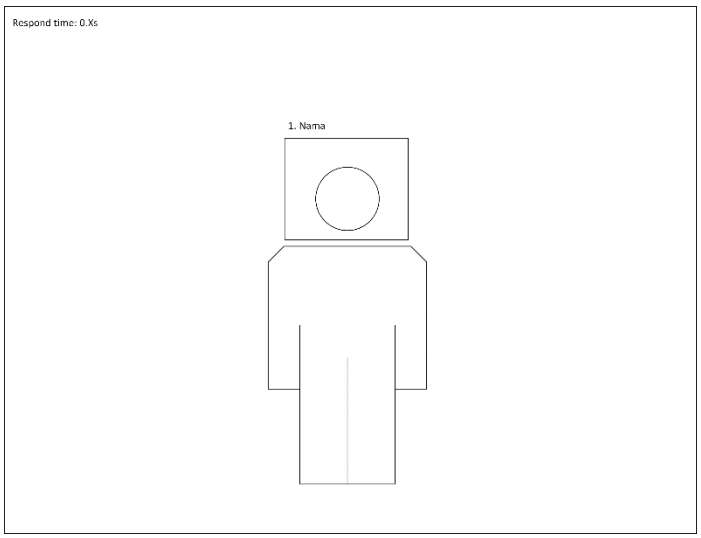
Data input yang digunakan dalam pengembangan perangkat lunak ini ialah data citra wajah mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya dan AT & T Face Database yang terdapat pada gambar IV-4 dan gambar IV-5.

Gambar IV-4. Data citra beberapa mahasiswa Teknik Informatika

Gambar IV-5. Sampel Dari AT&T Database

2. Perancangan Antar Muka

Pada subbab ini diperlihatkan mengenai perancangan antarmuka dari perangkat lunak yang akan dikembangkan. Rancangan antar muka untuk melakukan proses training dilakukan dengan menggukan command line interface. Sedangkan antar muka pada pengenalan wajah dapat dilihap pada gambar IV-6.



Gambar IV-6. Rancangan antarmuka pengenalan wajah

3. Kebutuhan Sistem

Dalam pengembangan perangkat lunak ini menggunakan software dan hardware serta bahasa pemrograman, dalam hal ini bahasa pemrograman yang digunakan ialah Python. Adapun spesifikasi dari hardware yang digunakan dalam pengembangan perangkat lunak ini ialah sebagai ditampilkan pada tabel IV- 5.

Tabel IV-5. Spesifikasi dalam pengembangan perangkat lunak

Merk & Tipe	Dell Latitude E4310
Prosessor	Intel® Core™ i5-5200U CPU @ 2.7GHz
RAM	4 GB
Media Penyimpanan	500 GB

Sedangkan untuk perangkat lunak yang digunakan dalam pengembangan beserta pengujian adalah:

a. Sistem Operasi Windows 10 Pro for Workstation 64bit

b. Pycharm Professional 2019.1.3

c. Anaconda 3

4. Sequence Diagram

Sequence diagram menggambarkan interaksi dan juga komunikasi yang dilakukan antar modul dengan tugasnya masing-masing. Pada perangkat lunak ini digambarkan sequence diagram pada gambar IV-7 sampai IV-8 berdasarkan *use case* yang telah didefinisikan di atas.

4.2.3 Fase Konstruksi

Tahapan ketiga dalam metode pengembangan perangkat lunak Rational Unified Process (RUP) adalah konstruksi, fase ini berfokus pada implementasi desain perangkat lunak sesuai hasil anaisis pada fase-fase sebelumnya. Hasil yang akan diperoleh pada proses ini ialah perangkat lunak yang siap digunakan sebagai alat penelitian oleh end user.

4.2.3.1 Kebutuhan Sistem

Pada pengembangan perangkat lunak ini digunakan Anaconda dengan bahasa python yang mendukung pengembangan perangkat lunak ini.

4.2.3.2 Implementasi

Dalam tahapan konstruksi terdapat fase implementasi yang mana dilakukannya pengembangan perangkat lunak berdasarkan diagram dan rancangan antarmuka yang telah dibuat pada fase elaborasi.

1. Implementasi Antar Muka

Dalam pengimplementasian antarmuka perangkat lunak ini disesuaikan dengan perancangan pada tahap elaborasi sebelumnya. Implementasi antarmuka dari perangkat lunak ini dapat dilihat dalam gambar IV- 9 dan IV- 10.

Gambar IV-9. Implementasi antar muka.

Gambar IV-10. Implementasi antar muka

4.2.4 Fase Transisi

Fase ini membahas tentang pengujian dari perangkat pengenalan wajah yang telah dibangun. Pengujian dilakukan berdasarkan hasil dari pengembangan perangkat lunak pada fase konstruksi.

4.2.4.1 Pemodelan Bisnis

Pengujian yang digunakan dalam menguji perangkat lunak yang dibangun ini ialah menggunakan *black box testing* yang mana pengujian ini diperuntukkan untuk memeriksa fungsionalitas dari perangkat lunak itu sendiri. Dalam hal ini diawali dengan rencana pengujian berdasarkan use case yang telah didefinisikan pada tahapan insepsi di atas.

4.2.4.2 Kebutuhan Sistem

Dalam pengembangan perangkat lunak ini menggunakan software dan hardware serta bahasa pemrograman, dalam hal ini bahasa pemrograman yang digunakan ialah Python. Adapun spesifikasi dari hardware yang digunakan dalam fase transisi perangkat lunak ini ialah sebagai ditampilkan pada tabel IV- 6.

Tabel IV-6. Kebutuhan system fase transisi.

Merk & Tipe	Dell Latitude E4310
Prosessor	Intel® Core™ i5-5200U CPU @ 2.7GHz
RAM	4 GB
Media Penyimpanan	500 GB

Sedangkan untuk perangkat lunak yang digunakan dalam pengujian adalah:

a. Sistem Operasi Ubuntu Linux 64bit.

b. Pycharm Professional 2019.1.3

c. Anaconda 3

4.2.4.3 Pengujian Perangkat Lunak

Pengujian perangkat lunak verifikasi wajah dengan oklusi ini dibuat berdasarkan use case yang telah dibuat sebelumnya. Pengujian ini akan dijelaskan pada tabel IV-7 sampai tabel IV-9.

1. Pengujian *Use Case* Training

Pengujian use case training dapat dilihat pada tabel IV-7.

Tabel IV-7. Pengujian unit pada use case training.

No.	Deskripsi	Jenis Pengujian	Tingkat Pengujian
1.	Menjalankan baris perintah: *./face_recognition.py -o start_train*	*Black Box*	Pengujian Unit

2. Pengujian *Use Case* Pengenalan Wajah

Pengujian untuk use case pengenalan dan menampilkan hasil pengenalan wajah disajikan pada tabel IV-8.

Tabel IV-8. Pengujian unit pada pengenalan wajah.

No.	**Deskripsi**	**Jenis Pengujian**	**Tingkat Pengujian**
1.	Memposisikan wajah pada kamera	*Black box*	Pengujian Unit
2.	Melihat hasil pengenalan wajah	*Black box*	Pengujian Unit

4.3 Kesimpulan

Penelitian ini menggunakan metode pengembangan perangkat lunak RUP yang mana terdiri dari empat fase yaitu fase insepsi, elaborasi, konstruksi dan transisi. Pada fase insepsi dilakukannya identifikasi terhadap perangkat lunak yang dikembangkan pada fase ini dibuat pemodelan diagram use case dan kelas analisis. Adapun use case pada penelitian ini yaitu memasukkan gambar wajah dan menampilkan hasil verifikasi wajah. Pada fase elaborasi dirancangnya sequence diagram yang dibuat berdasarkan diagram use case sebelumnya. Sedangkan pada fase konstrusi dilakukannya proses konstruksi yang menghasilkan kelas-kelas. Dan yang terakhir ialah fase transisi, pada fase ini dilakukan pengujian terhadap perangkat lunak yang dikembangkan dengan menggunakan black box testing.

BAB V

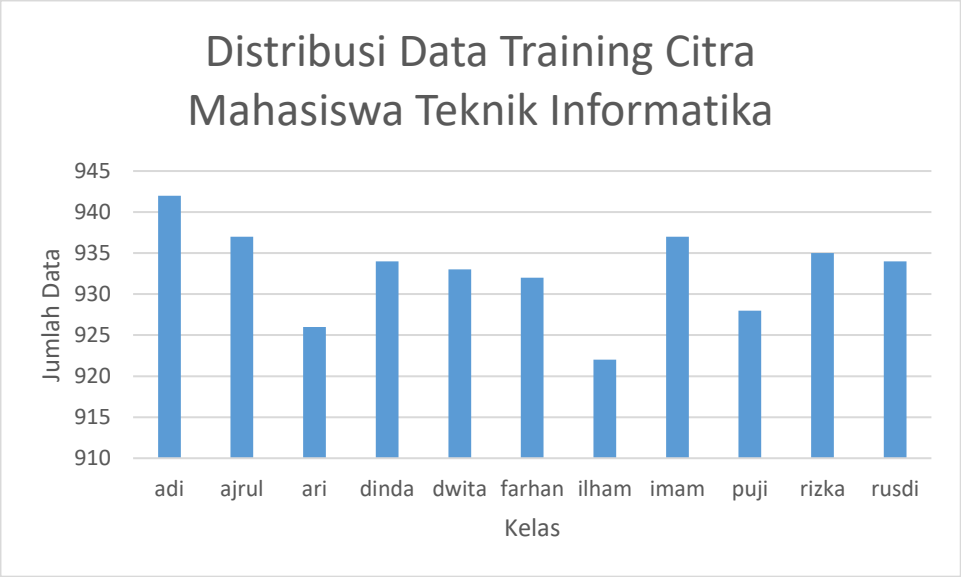
HASIL DAN ANALISIS PENELITIAN

5.1 Pendahuluan

Pada bab V ini dijelaskan mengenai hasil dari percobaan-percobaan yang dilakukan dari pengimplementasian perangkat lunak dan menganalisis hasil dari percobaan tersebut berdasarkan pengembangan perangkat lunak pengenalan wajah multiface secara *realtime* yang telah diterapkan pada bab IV.

5.2 Hasil Percobaan Penelitian

Proses pelatihan dilakukan pada dua dataset. Dataset pertama adalah citra wajah mahasiswa Teknik Informatika Universitas Sriwijaya yang berjumlah 15 citra pada setiap kelas terdiri dari 11 kelas. Setiap citra pada data citra wajah mahasiswa Teknik Informatika Universitas Sriwijaya diaugmentasi dengan cara merubah kecerahan dan kontras secara acak menjadi 10.260 citra. Pada gambar V-1 dapat dilihat grafik distribusi data citra Teknik informatika yang digunakan pada saat pelatihan setelah dilakukan augmentasi.



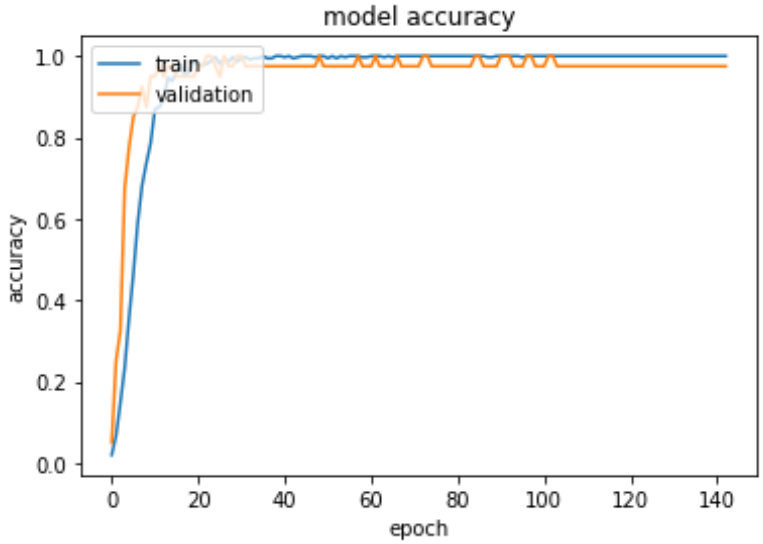
Gambar V-1. Grafik Distribusi Data Pelatihan Citra Wajah Teknik Informatika.

Dataset kedua adalah adalah ORL dataset atau sekarang telah berganti nama menjadi AT & T Dataset yang berju9mlah 400 citra terdiri dari 40 kelas atau label. Dalam proses training terdapat proses validasi yang berguna untuk memantau performa model pengenalan wajah pada data uji.

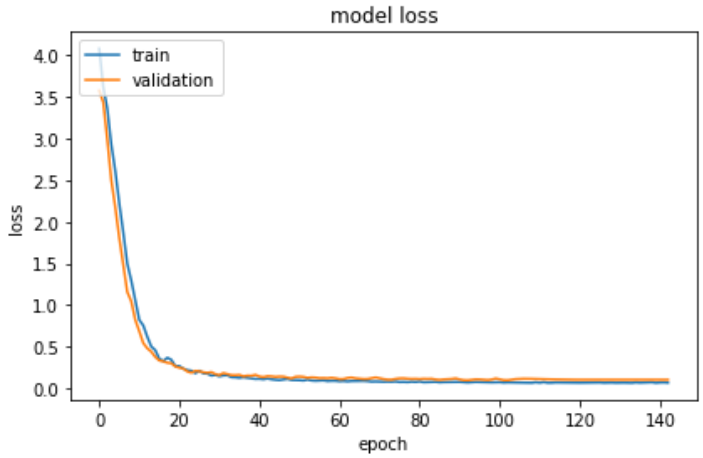
Pengujian dan pelatihan dilakukan menggunakan dua arsitektur *convolutional neural networks* yaitu arsitektur *convolutional neural network* sederhana (Simple CNN) dan VGG16 seperti yang dijelaskan pada sub bab 3.4.1. Pelatihan pada *Simple CNN* dan VGG16 dilakukan dengan epoch sebanyak 1000 kali. Dengan memantau nilai *loss* dan akurasi pada setiap epoch, maka dipilih model yang terbaik yaitu model yang memiliki nilai loss paling rendah dan akurasi paling tinggi. *Optimizer* yang digunakan adalah *Stockhastic Gradient Descent* dan *Loss* yang digunakan yaitu *Categorical Crossentropy*. Dalam teknisnya proses pelatihan pada kedua model diberlakukan aturan agar menghentikan proses

pelatihan pada epoch tertentu jika nilai loss tidak berkurang selama 50 epoch dari nilai loss terendah.

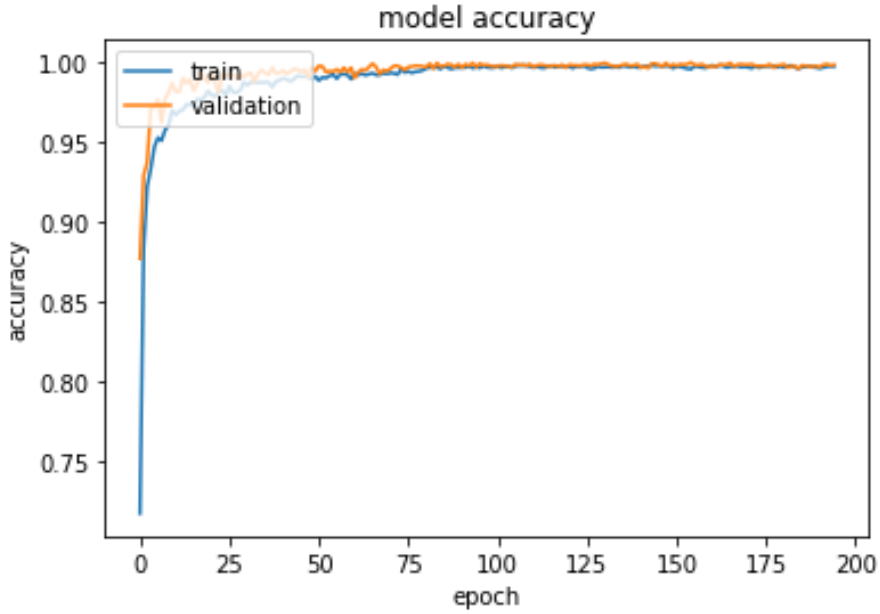
Hasil proses pelatihan ditampilkan dalam bentuk grafik perbandingan antara *nilai* akurasi dan *loss* dalam setiap *epoch* dari arsitektur *Simple CNN* dan VGG16 yang terdapat pada Gambar V-2 sampai dengan gambar V-9



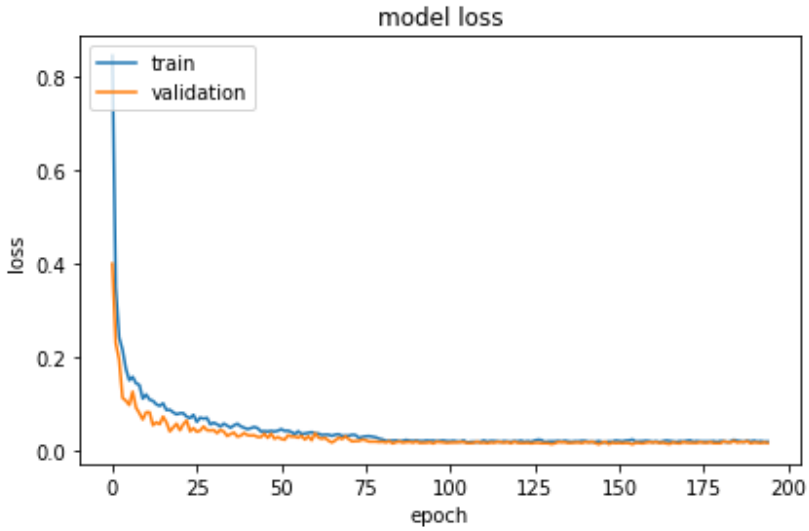
Gambar V-2. Grafik proses traning menampilkan nilai akurasi pada setiap epoch pada AT & T dataset arsitektur VGG16.



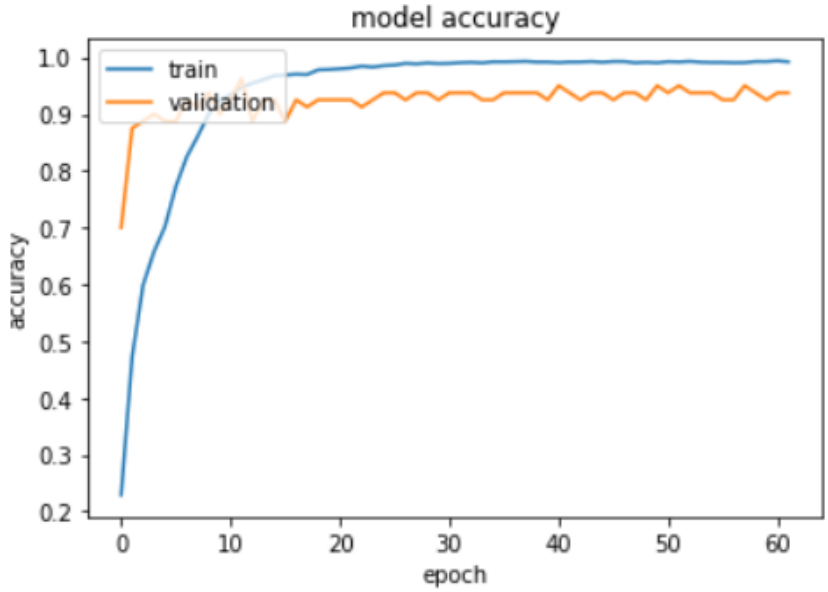
Gambar V-3. Grafik training menampilkan nilai *loss* pada setiap epoch pada AT & T dataset arsitektur VGG16.



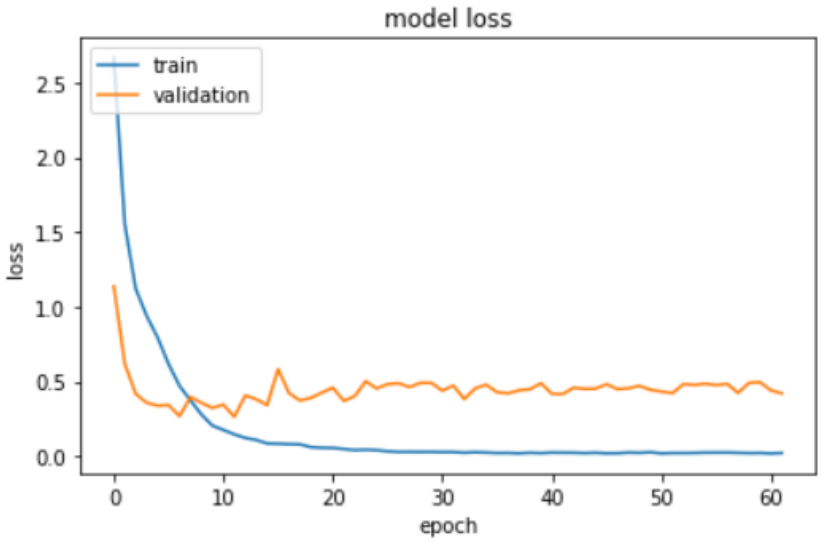
Gambar V-4. Grafik training menampilkan nilai akurasi pada setiap epoch pada dataset wajah mahasiswa Teknik Informatika Universitas Sriwijaya arsitektur VGG16.



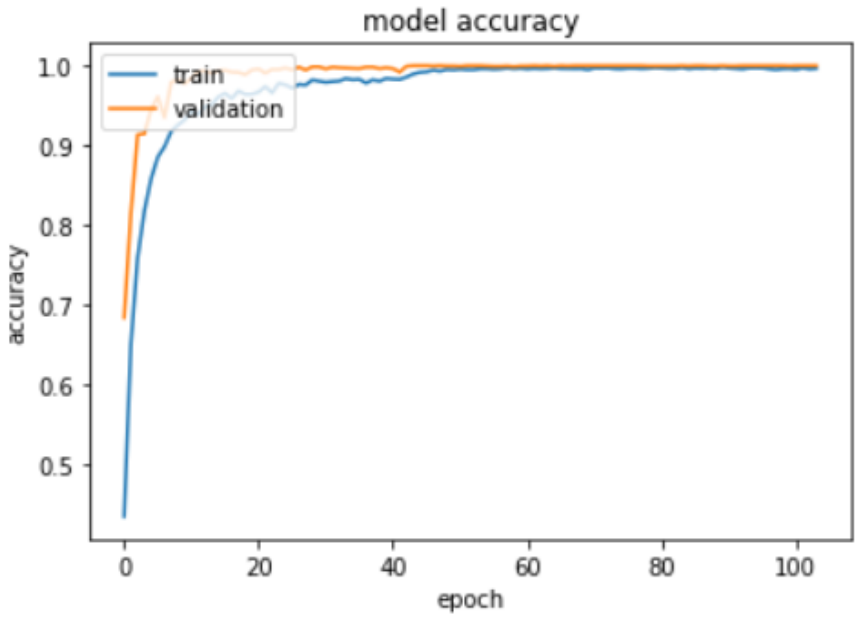
Gambar V-5. Grafik training menampilkan nilai *loss* setiap epoch pada dataset wajah mahasiswa Teknik Informatika Universitas Sriwijaya arsitektur VGG16.



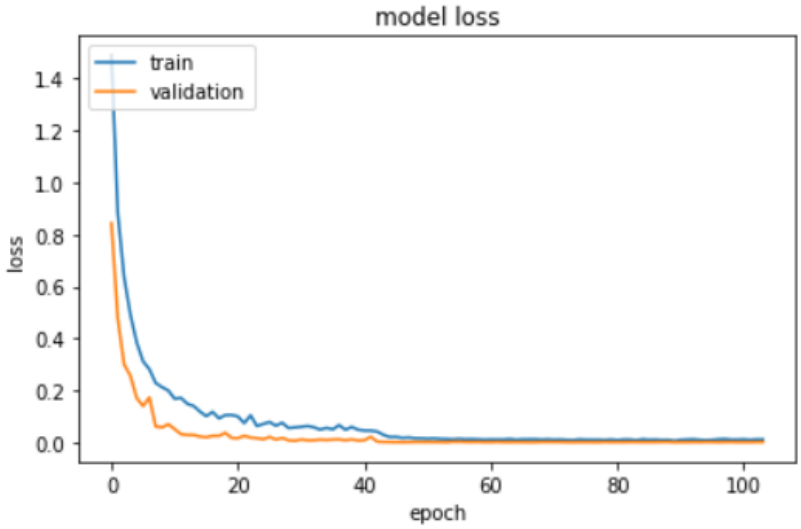
Gambar V-6. Grafik proses traning menampilkan nilai akurasi pada setiap epoch pada AT & T dataset arsitektur *Simple CNN*.



Gambar V-7. Grafik training menampilkan nilai *loss* pada setiap epoch pada AT & T dataset arsitektur *Simple CNN*.



Gambar V-8. Grafik training menampilkan nilai akurasi pada setiap epoch pada dataset wajah mahasiswa Teknik Informatika Universitas Sriwijaya arsitektur *Simple CNN*.



Gambar V-9. Grafik training menampilkan nilai *loss* setiap epoch pada dataset wajah mahasiswa Teknik Informatika UNSRI arsitektur *Simple CNN*.

Percobaan pengujian terhadap model hasil *training* dalam proses pengenalan wajah dibagi menjadi dua skenario. Skenario pertama dilakukan percobaan pengenalan pada citra single face terhadap citra wajah mahasiswa Teknik informatika yang berjumlah 2.769 citra dan AT & T dataset yang berjumlah 80 citra. hasil percobaan penelitian disajikan dalam format confusion matrix. Skenario kedua dilakukannya proses pengujian secara realtime pada citra multiface dengan input berupa citra yang ditangkap dari kamera.

5.2.1. Skenario Percobaan Pertama

Skenario percobaan pertama dilakukan pada data citra wajah mahasiswa fasilkom unsri sebanyak 2769 citra wajah yang terdiri dari dari 11 label dan AT & T dataset yang berjumlah 80 citra yang terdiri dari 40 label. Hasil pengujian pada citra wajah mahasiswa fasilkom unsri dan AT & T dataset ditampilkan dalam format *confusion matrix* yang dapat dilihat pada Gambar V-5 dan penghitungan performa pengenalan wajah dilakukan dengan menghitung empat nilai yaitu *accuracy, precision, sensitifity,* dan *f-measure* dengan persamaan seperti yang telah dijelaskan dalam subbab 2.6. Sebagian hasil dari proses pengenalan wajah pada arsitektur VGG16 dan *Simple CNN* dipaparkan pada tabel V-1 sampai tabel V-4. Hasil lengkap pengujian dari proses pengenalan wajah pada AT & T dataset dipaparkan pada lampiran.

Tabel V-1. Hasil Percobaan pada AT & T face database pada arsitektur VGG16

No	Image	Name	Predicted
1.		s1	s1
2.		s2	s2
3.		s3	s3
4.		s4	s4
5.		s5	s5

Tabel V-2. Hasil Percobaan *pada* citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16.

.

No	**Original Image**	**Name**	**Predicted**
1.		adi	adi
2.		ajrul	ajrul
3.		ari	Ari
4.		dinda	dinda
5.		dwita	dwita

Tabel V-3. Hasil Percobaan pada AT & T face database pada arsitektur *Simple CNN.*

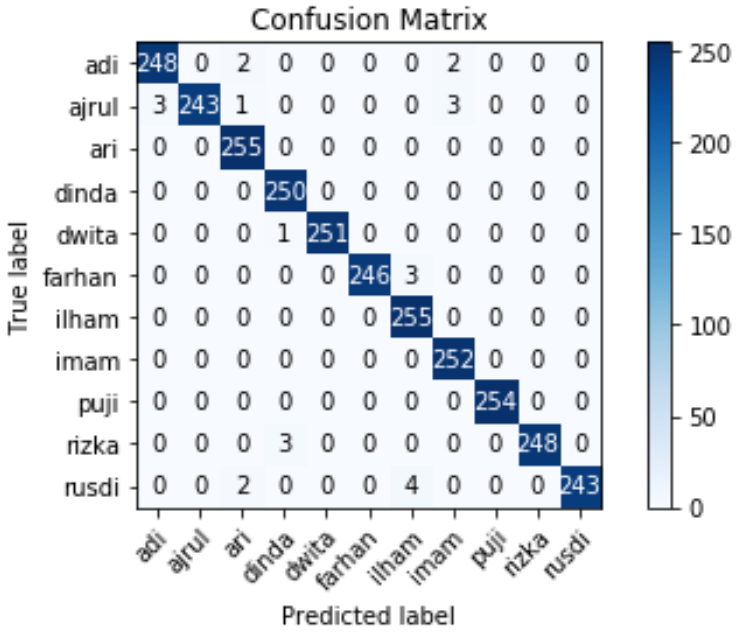
No	Image	Name	Predicted
1.		s1	s1
2.		s2	s2
3.		s3	s3
4.		s4	s4
5.		s5	s5

Tabel V-4. Hasil Percobaan *pada* citra wajah mahasiswa fasilkom unsri pada arsitektur *Simple CNN*.

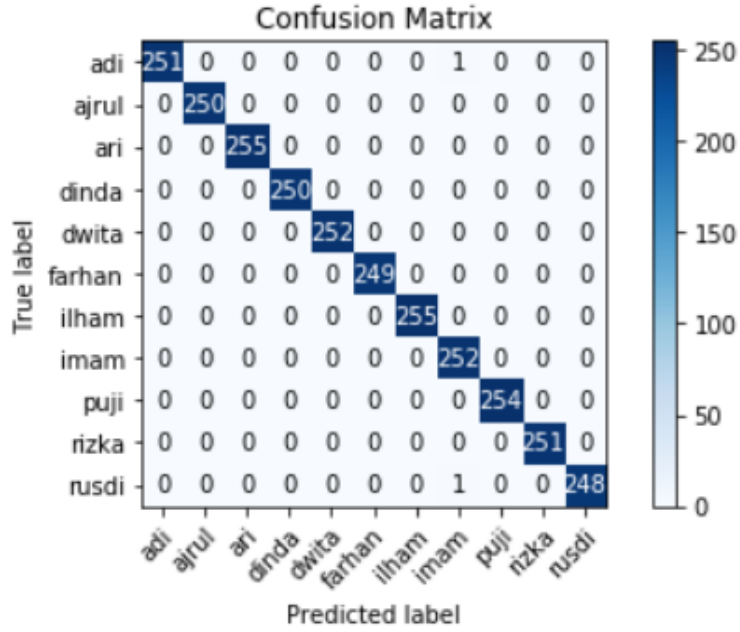
.

No	**Original Image**	**Name**	**Predicted**
1.		adi	adi
2.		ajrul	ajrul
3.		ari	Ari
4.		dinda	dinda
5.		dwita	dwita

Performa hasil penelitian dataset citra wajah mahasiswa fasilkom unsri dan dapat dilihat pada gambar dan V-10 sampai gambar V-11. Sedangkan untuk AT & T face database dapat dilihat di lampiran



Gambar V-10. *Confusion Matrix* citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16.



Gambar V-11. *Confusion Matrix* citra wajah mahasiswa fasilkom unsri *Simple CNN*.

Tabel V-5. Performa *precision, sensitifity,* dan *f1-score* dataset citra wajah mahasiswa fasilkom unsri pada arsitektur VGG16.

No	Name	T+	T-	F+	F-	Precision	Recall	F-Measure	Akurasi
1	adi	248	2493	3	4	0.99	0.98	0.99	0.99
2	ajrul	243	2498	0	7	1.00	0.97	0.99	0.99
3	ari	255	2486	5	0	0.98	1.00	0.99	0.99
4	dinda	246	2495	4	0	0.98	1.00	0.99	0.99
5	dwita	251	2490	0	1	1.00	1.00	1.00	0.99
6	farhan	246	2495	0	3	1.00	0.99	0.99	0.99
7	ilham	255	2486	7	0	0.97	1.00	0.99	0.99
8	imam	252	2489	5	0	0.98	1.00	0.99	0.99
9	puji	254	2487	0	0	1.00	1.00	1.00	0.99
10	rizka	248	2493	0	3	1.00	0.99	0.99	0.99
11	rusdi	243	2498	0	6	1.00	0.98	0.99	0.99

Tabel V-6. Performa *precision, sensitifity,* dan *f1-score* dataset citra wajah mahasiswa fasilkom unsri pada arsitektur *Simple CNN*.

Name	T+	T-	F+	F-	Precision	Recall	F-Measure	Akurasi
adi	251	2516	0	1	1.00	0.99603175	0.99801	0.99928
ajrul	250	2517	0	0	1.00	1.00	1.00	0.99928
ari	255	2512	0	0	1.00	1.00	1.00	0.99928
dinda	250	2517	0	0	1.00	1.00	1.00	0.99928
dwita	252	2515	0	0	1.00	1.00	1.00	0.99928
farhan	249	2518	0	0	1.00	1.00	1.00	0.99928
ilham	255	2512	0	0	1.00	1.00	1.00	0.99928
imam	252	2515	2	0	0.99213	1.00	0.99605	0.99928
puji	254	2513	0	0	1.00	1.00	1.00	0.99928
rizka	251	2516	0	0	1.00	1.00	1.00	0.99928
rusdi	248	2519	0	1	1.00	0.99598394	0.99799	0.99928

5.2.2. Skenario Percobaan Kedua

Percobaan kedua dilakukan pada citra *multiface* dan single face secara *realtime.* Skenario percobaan kedua dilakukan menggunakan kamera *integrated webcam* pada laptop Dell Latitude E4310. Laptop Dell latitude E4310 memiliki *integrated webcam* yang mempunyai spesifikasi seperti terlihat pada tabel V-7.

Tabel V-7. Spesifikasi *integrated webcam* pada Dell Latitude E4310

Frame rate:	4 FPS
Stream Type:	video
Image Mode:	rgb
Webcam MegaPixels:	1.92 MP
Webcam Resolution:	1600×1200
Video Standard:	UXGA
Aspect Ratio:	1.33
PNG File Size:	2.48 MB
JPEG File Size:	1.61 MB
Bitrate:	6.43 MB/s
Number of Colors:	98790
Lightness:	47.06%
Luminosity:	48.41%
Brightness:	47.58%
Hue:	60°
Saturation:	3.33%

Berdasarkan spesifikasi kamera pada tabel V-7 dilakukan pengujian pada device Dell Latitude E4310 untuk arsitektur VGG16 dan *Simple CNN* dimana sebagian hasil pengujian dapat dilihat pada table V-8 dan V-9.

Tabel V-8. Pengujian Pada Device Dell Latitude E4310 arsitektur VGG16

No.	Citra	Label	Output Sistem	Benar/Salah	Waktu (detik)
1.		Ari(1) Puji(2)	Ari(1) Puji(2)	Benar: 2 Salah: 0	0.44
2.		Ari (1)	Ari(1)	Benar: 1 Salah: 0	0.43
4.		Puji (1)	Puji (1)	Benar: 1 Salah: 0	0.48
5.		Ari (1)	Ari (1)	Benar: 1 Salah: 0	0.46
6.		Ari (1)	Ari (1)	Benar: 1 Salah: 0	0.44
7.		Puji (1) Ari (2)	Puji (1) Ilham (2)	Benar: 1 Salah: 1	0.43

Tabel V-9. Pengujian Pada Device Dell Latitude E4310 arsitektur *Simple CNN*.

No.	Citra	Label	Output Sistem	Benar/Salah	Waktu (detik)
1.		Ari(1) Puji(2)	Ari(1) Puji(2)	Benar: 2 Salah: 0	0.03
2.		Puji (1)	Puji(1)	Benar: 1 Salah: 0	0.02
4.		Ari(1) Puji(2)	Adi(1) Puji (2)	Benar: 1 Salah: 1	0.02
5.		Ari (1)	Ari (1)	Benar: 1 Salah: 0	0.03
6.		Ari (1)	Ari (1)	Benar: 1 Salah: 0	0.03
7.		Ari (1) Puji (2)	Ari (1) Puji (2)	Benar: 2 Salah: 0	0.02

5.3 Analisis Hasil Percobaan Penelitian

Dari pengujian di atas telah dilakukan dua skenario pengujian. Pada skenario pengujian pertama dilakukan proses pengenalan wajah dari AT&T dataset dan citra wajah Teknik Informatika Universitas Sriwijaya. yang sudah melalui proses pelatihan sebelumnya. Pengujian skenario kedua dilakukan pada model yang sudah dilatih menggunakan dataset citra mahasiswa Teknik Informatika Unsri menggunakan dua arsitektur yang berbeda.

Dataset citra wajah mahasiswa Teknik Informatika memperoleh performa lebih baik ketika dilakukan pengujian menggunakan VGG16 maupun *Simple CNN* dibanding dengan AT & T face database. Hal ini dikarenakan data pada citra wajah Teknik Informatika dilakukan augmentasi secara acak dan berjumlah lebih dari 10.000 berbeda dengan AT & T dataset untuk menjaga keaslian dataset tidak dilakukan augmentasi. Banyak data yang digunakan pada proses pelatihan mempengaruhi *robustness* dari model *convolutional neural network* sehingga dapat mengenali berbagai macam kondisi yang tidak menentu.

Pengujian skenario kedua dilakukan secara realtime dan mendapatkan hasil bahwa perbedaan arsitektur yang digunakan tidak terlalu besar mempengaruhi akurasi meskipun ukuran dari kedua arsitektur tersebut jauh berbeda dimana VGG16 lebih kompleks dibanding dengan *Simple CNN*. *Respond time* antara kedua arsitektur sangat terlihat dimana *Simple CNN* lebih unggul dibanding dengan VGG16 dikarenakan arsitektur VGG16 lebih kompleks.

Tabel V-10. Perbandingan Performa arsitektur *convolutional neural network* saat pengujian secara realtime

Arsitektur	Akurasi	*Respond Time* (Rata-Rata)
Simple CNN	70%	0.03 detik
VGG16	86%	0.46 detik

Nilai akurasi pada saat pengujian secara offline dan secara realtime berbeda dikarenakan kondisi saat pengujian secara realtime tidak menentu seperti pencahayaan, pose kepala, dan ekspresi yang beragam.

5.4 Kesimpulan

Skenario pengujian dibagi menjadi dua yaitu skenario pertama secara *offline* dan skenario kedua secara *realtime.* Hasil percobaan penelitian yang telah dilakukan pada skenario pertama menghasilkan performa rata-rata 0.99 untuk masing masing *precision, sensitifity,* dan *f1-score.* Pada proses pengujian secara *realtime* didapatkan nilai *respond time* sebesar rata-rata 0.46 detik pada arsitektur VGG16 dan 0.03 detik untuk arsitektur *simple CNN* untuk setiap proses pengenalan wajah oleh sistem. Arsitektur yang digunakan pada setiap skenario pengujianadalah *Simple CNN* dan VGG16. *Simple CNN* memiliki nilai akurasi lebih rendah daripada VGG16 tetapi memiliki *respond time* yang lebih baik dikarenakan *Simple CNN* tidak terlalu kompleks.

BAB VI

KESIMPULAN DAN SARAN

6.1 Pendahuluan

Pada bab VI ini dijelaskan mengenai kesimpulan dari hasil dan analisis pengembangan perangkat lunak yang telah dibuat dan kesimpulan dari seluruh bab sebelumnya serta membuat saran agar penelitian selanjutnya dapat lebih baik.

6.2 Kesimpulan

Kesimpulan yang diperoleh dari perangkat lunak yang telah dikembangkan adalah sebagai berikut:

1. Pengenalan wajah pada citra multi face dapat dilakukan menggunakan metode *Convolutional Neural Network*.
2. Setelah dilakukan pengujian perangkat lunak pengenalan wajah dengan metode *Convolutional Neural Network* menghasilkan nilai akurasi sebesar 99% menggunakan dataset dari *sample* mahasiswa Teknik Informatika Universitas Sriwijaya dan AT & T Face Database sebesar 95%.
3. Pengujian secara *realtime* mendapatkan nilai akurasi sebesar 80% dan *response time* untuk melakukan proses pengenalan wajah yaitu selama < 1 detik.

6.3 Saran

Saran yang dapat dipakai untuk pengembangan penelitian berikutnya agar penelitian selanjutnya dapat lebih baik adalah:

1. Penambahan pra-proses citra yang lebih baik untuk meningkatkan performa pengenalan wajah serta mengatasi permasalahan performa pengenalan wajah menurun ketika terjadi perubahan pencahayaan yang terlalu terang atau terlalu gelap.
2. Arsitektur *Convolutional Neural Network* selain *VGG16* dapat digunakan seperti *Siamese Network* dan *FaceNet.*

DAFTAR PUSTAKA

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. *MIT Press*, 1. https://doi.org/10.1038/nmeth.3707

He, X., Yan, S., Hu, Y., Niyogi, P., & Zhang, H.-J. (2005). Face Recognition Using Laplacianfaces. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, *27*(3). https://doi.org/10.1109/TPAMI.2005.55 PM - 15747789 M4 - Citavi

Juhong, A., & Pintavirooj, C. (2017). Face recognition based on facial landmark detection. *BMEiCON 2017 - 10th Biomedical Engineering International Conference*,*2017-Janua*(2),1–4. https://doi.org/10.1109/BMEiCON.2017.8229173

Kruchten, P. (2000). *The Rational Unified Process An Introduction Second Edition* (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Lebedev, A., Khryashchev, V., Priorov, A., & Stepanova, O. (2017). Face verification based on convolutional neural network and deep learning. *Proceedings of 2017 IEEE East-West Design and Test Symposium, EWDTS 2017*. https://doi.org/10.1109/EWDTS.2017.8110157

Liu, C. (2015). The development trend of evaluating face-recognition technology. *Proceedings - 2014 International Conference on Mechatronics and Control, ICMC2014*,(August1994),1540–1544. https://doi.org/10.1109/ICMC.2014.7231817

Nielsen, M. A. (2015). Neural Networks and Deeplearning.

Sujay, S. N., Reddy, H. S. M., & Ravi, J. (2017). Face recognition using extended LBP features and multilevel SVM classifier. *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques(ICEECCOT)*,1–4. https://doi.org/10.1109/ICEECCOT.2017.8284596

Yan, K., Huang, S., Song, Y., Liu, W., & Fan, N. (2017). Face Recognition Based on Convolution Neural Network. *Proceedings of the 36th Chinese Control Conference*, 4077–4081. https://doi.org/10.2991/iccia-17.2017.55

Yan Lei. (2011). Fusion method of PCA and BP neural network for face recognition. *2011 International Conference on Computer Science and Service System (CSSS)*, 3256–3259. https://doi.org/10.1109/CSSS.2011.5974680

Lampiran I. Pengujian pada *AT & T Face Database*

Pada tabel L-1 berikut merupakan hasil pengujian proses pengenalan wajah menggunakan *AT & T Face Database* serta perbandingan hasil pengenalan menggunakan arsitektur VGG16 dan *Simple CNN*. Data yang digunakan dalam pengujian dapat diperoleh dengan mengunjungi laman berikut ini:

1. https://bitbucket.org/arisusanto-ta/face_recognition/downloads/att_dataset.zip
2. https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Tabel L-1. Hasil Pengujian *AT & T Face Database*.

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1	1_8.png	s1	s1	Benar	s1	Benar
2	1_9.png	s1	s1	Benar	s1	Benar
3	10_8.png	s10	s10	Benar	s10	Benar
4	10_9.png	s10	s10	Benar	s10	Benar
5	11_8.png	s11	s11	Benar	s11	Benar
6	11_9.png	s11	s11	Benar	s11	Benar
7	12_8.png	s12	s12	Benar	s12	Benar
8	12_9.png	s12	s12	Benar	s12	Benar
9	13_8.png	s13	s13	Benar	s13	Benar
10	13_9.png	s13	s13	Benar	s13	Benar
11	14_8.png	s14	s14	Benar	s14	Benar
12	14_9.png	s14	s14	Benar	s14	Benar
13	15_8.png	s15	s15	Benar	s15	Benar
14	15_9.png	s15	s15	Benar	s15	Benar
15	16_8.png	s16	s16	Benar	s16	Benar
16	16_9.png	s16	s16	Benar	s16	Benar
17	17_8.png	s17	s17	Benar	s17	Benar
18	17_9.png	s17	s17	Benar	s17	Benar
19	18_8.png	s18	s18	Benar	s18	Benar
20	18_9.png	s18	s18	Benar	s18	Benar
21	19_8.png	s19	s19	Benar	s19	Benar
22	19_9.png	s19	s19	Benar	s16	Salah
23	2_8.png	s2	s2	Benar	s2	Benar
24	2_9.png	s2	s2	Benar	s2	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
25	20_8.png	s20	s20	Benar	s38	Salah
26	20_9.png	s20	s20	Benar	s20	Benar
27	21_8.png	s21	s21	Benar	s21	Benar
28	21_9.png	s21	s21	Benar	s21	Benar
29	22_8.png	s22	s22	Benar	s22	Benar
30	22_9.png	s22	s22	Benar	s22	Benar
31	23_8.png	s23	s23	Benar	s23	Benar
32	23_9.png	s23	s23	Benar	s23	Benar
33	24_8.png	s24	s24	Benar	s24	Benar
34	24_9.png	s24	s24	Benar	s24	Benar
35	25_8.png	s25	s25	Benar	s25	Benar
36	25_9.png	s25	s25	Benar	s25	Benar
37	26_8.png	s26	s26	Benar	s26	Benar
38	26_9.png	s26	s26	Benar	s26	Benar
39	27_8.png	s27	s27	Benar	s27	Benar
40	27_9.png	s27	s27	Benar	s27	Benar
41	28_8.png	s28	s28	Benar	s28	Benar
42	28_9.png	s28	s28	Benar	s37	Salah
43	29_8.png	s29	s29	Benar	s29	Benar
44	29_9.png	s29	s29	Benar	s29	Benar
45	3_8.png	s3	s3	Benar	s3	Benar
46	3_9.png	s3	s3	Benar	s3	Benar
47	30_8.png	s30	s30	Benar	s30	Benar
48	30_9.png	s30	s30	Benar	s30	Benar
49	31_8.png	s31	s31	Benar	s31	Benar
50	31_9.png	s31	s31	Benar	s31	Benar
51	32_8.png	s32	s32	Benar	s32	Benar
52	32_9.png	s32	s32	Benar	s32	Benar
53	33_8.png	s33	s33	Benar	s33	Benar
54	33_9.png	s33	s33	Benar	s33	Benar
55	34_8.png	s34	s34	Benar	s34	Benar
56	34_9.png	s34	s34	Benar	s34	Benar
57	35_8.png	s35	s35	Benar	s35	Benar
58	35_9.png	s35	s35	Benar	s35	Benar
59	36_8.png	s36	s36	Benar	s36	Benar
60	36_9.png	s36	s36	Benar	s36	Benar
61	37_8.png	s37	s37	Benar	s37	Benar
62	37_9.png	s37	s37	Benar	s37	Benar
63	38_8.png	s38	s38	Benar	s38	Benar
64	38_9.png	s38	s38	Benar	s38	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
65	39_8.png	s39	s39	Benar	s39	Benar
66	39_9.png	s39	s39	Benar	s39	Benar
67	4_8.png	s4	s4	Benar	s4	Benar
68	4_9.png	s4	s4	Benar	s4	Benar
69	40_8.png	s40	s40	Benar	s40	Benar
70	40_9.png	s40	s40	Benar	s18	Salah
71	5_8.png	s5	s5	Benar	s5	Benar
72	5_9.png	s5	s5	Benar	s5	Benar
73	6_8.png	s6	s6	Benar	s6	Benar
74	6_9.png	s6	s6	Benar	s6	Benar
75	7_8.png	s7	s7	Benar	s7	Benar
76	7_9.png	s7	s7	Benar	s7	Benar
77	8_8.png	s8	s8	Benar	s8	Benar
78	8_9.png	s8	s8	Benar	s8	Benar
79	9_8.png	s9	s9	Benar	s9	Benar
80	9_9.png	s9	s9	Benar	s9	Benar

Lampiran II. Pengujian pada Data Wajah Mahasiswa Teknik Informatika

Pada tabel L-2 berikut merupakan hasil pengujian proses pengenalan wajah menggunakan Citra Wajah Mahasiswa Teknik Informatika serta perbandingan hasil pengenalan menggunakan arsitektur VGG16 dan *Simple CNN*. Data yang digunakan dalam pengujian dapat diperoleh dengan mengunjungi laman berikut:

1. https://bitbucket.org/arisusanto-ta/face_recognition/downloads/data.zip

Tabel L-2. Hasil Pengujian Data Wajah Mahasiswa Teknik Informatika.

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1	adi_0_1061.jpeg	adi	adi	Benar	adi	Benar
2	adi_0_1072.jpeg	adi	adi	Benar	adi	Benar
3	adi_0_1109.jpeg	adi	adi	Benar	adi	Benar
4	adi_0_1120.jpeg	adi	ari	Salah	adi	Benar
5	adi_0_1138.jpeg	adi	adi	Benar	adi	Benar
6	adi_0_114.jpeg	adi	adi	Benar	adi	Benar
7	adi_0_1170.jpeg	adi	adi	Benar	adi	Benar
8	adi_0_1172.jpeg	adi	adi	Benar	adi	Benar
9	adi_0_1212.jpeg	adi	adi	Benar	adi	Benar
10	adi_0_1218.jpeg	adi	adi	Benar	adi	Benar
11	adi_0_1297.jpeg	adi	adi	Benar	adi	Benar
12	adi_0_1395.jpeg	adi	adi	Benar	adi	Benar
13	adi_0_1416.jpeg	adi	adi	Benar	adi	Benar
14	adi_0_1423.jpeg	adi	adi	Benar	adi	Benar
15	adi_0_1504.jpeg	adi	adi	Benar	adi	Benar
16	adi_0_1587.jpeg	adi	adi	Benar	adi	Benar
17	adi_0_171.jpeg	adi	adi	Benar	adi	Benar
18	adi_0_1757.jpeg	adi	adi	Benar	adi	Benar
19	adi_0_1798.jpeg	adi	adi	Benar	adi	Benar
20	adi_0_1835.jpeg	adi	adi	Benar	adi	Benar
21	adi_0_1857.jpeg	adi	adi	Benar	adi	Benar
22	adi_0_1865.jpeg	adi	adi	Benar	adi	Benar
23	adi_0_1871.jpeg	adi	adi	Benar	adi	Benar
24	adi_0_1890.jpeg	adi	adi	Benar	adi	Benar
25	adi_0_194.jpeg	adi	adi	Benar	adi	Benar
26	adi_0_1951.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
27	adi_0_199.jpeg	adi	adi	Benar	adi	Benar
28	adi_0_2023.jpeg	adi	adi	Benar	adi	Benar
29	adi_0_2025.jpeg	adi	adi	Benar	adi	Benar
30	adi_0_2090.jpeg	adi	adi	Benar	adi	Benar
31	adi_0_2155.jpeg	adi	adi	Benar	adi	Benar
32	adi_0_219.jpeg	adi	adi	Benar	adi	Benar
33	adi_0_2270.jpeg	adi	adi	Benar	adi	Benar
34	adi_0_2285.jpeg	adi	adi	Benar	adi	Benar
35	adi_0_2298.jpeg	adi	adi	Benar	adi	Benar
36	adi_0_2326.jpeg	adi	adi	Benar	adi	Benar
37	adi_0_2331.jpeg	adi	adi	Benar	adi	Benar
38	adi_0_2343.jpeg	adi	adi	Benar	adi	Benar
39	adi_0_2350.jpeg	adi	adi	Benar	adi	Benar
40	adi_0_2392.jpeg	adi	adi	Benar	adi	Benar
41	adi_0_2413.jpeg	adi	adi	Benar	adi	Benar
42	adi_0_2432.jpeg	adi	adi	Benar	adi	Benar
43	adi_0_2447.jpeg	adi	adi	Benar	adi	Benar
44	adi_0_2467.jpeg	adi	adi	Benar	adi	Benar
45	adi_0_2484.jpeg	adi	adi	Benar	adi	Benar
46	adi_0_2503.jpeg	adi	adi	Benar	adi	Benar
47	adi_0_2564.jpeg	adi	adi	Benar	adi	Benar
48	adi_0_2572.jpeg	adi	adi	Benar	adi	Benar
49	adi_0_2632.jpeg	adi	adi	Benar	adi	Benar
50	adi_0_2638.jpeg	adi	adi	Benar	adi	Benar
51	adi_0_2741.jpeg	adi	adi	Benar	adi	Benar
52	adi_0_29.jpeg	adi	adi	Benar	adi	Benar
53	adi_0_2904.jpeg	adi	adi	Benar	adi	Benar
54	adi_0_2919.jpeg	adi	adi	Benar	adi	Benar
55	adi_0_2921.jpeg	adi	adi	Benar	adi	Benar
56	adi_0_2955.jpeg	adi	adi	Benar	adi	Benar
57	adi_0_296.jpeg	adi	adi	Benar	adi	Benar
58	adi_0_2965.jpeg	adi	adi	Benar	adi	Benar
59	adi_0_2988.jpeg	adi	ari	Salah	adi	Benar
60	adi_0_3015.jpeg	adi	imam	Salah	adi	Benar
61	adi_0_3094.jpeg	adi	adi	Benar	adi	Benar
62	adi_0_3095.jpeg	adi	adi	Benar	adi	Benar
63	adi_0_318.jpeg	adi	adi	Benar	adi	Benar
64	adi_0_3241.jpeg	adi	adi	Benar	adi	Benar
65	adi_0_3246.jpeg	adi	adi	Benar	adi	Benar
66	adi_0_3272.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
67	adi_0_3295.jpeg	adi	adi	Benar	adi	Benar
68	adi_0_3334.jpeg	adi	adi	Benar	adi	Benar
69	adi_0_3398.jpeg	adi	adi	Benar	adi	Benar
70	adi_0_3403.jpeg	adi	adi	Benar	adi	Benar
71	adi_0_3424.jpeg	adi	adi	Benar	adi	Benar
72	adi_0_3425.jpeg	adi	adi	Benar	adi	Benar
73	adi_0_3485.jpeg	adi	adi	Benar	adi	Benar
74	adi_0_3543.jpeg	adi	adi	Benar	adi	Benar
75	adi_0_3608.jpeg	adi	adi	Benar	adi	Benar
76	adi_0_3678.jpeg	adi	adi	Benar	adi	Benar
77	adi_0_3689.jpeg	adi	adi	Benar	adi	Benar
78	adi_0_3754.jpeg	adi	adi	Benar	adi	Benar
79	adi_0_3776.jpeg	adi	adi	Benar	adi	Benar
80	adi_0_3793.jpeg	adi	adi	Benar	adi	Benar
81	adi_0_3821.jpeg	adi	adi	Benar	adi	Benar
82	adi_0_3896.jpeg	adi	adi	Benar	adi	Benar
83	adi_0_3913.jpeg	adi	adi	Benar	adi	Benar
84	adi_0_4027.jpeg	adi	adi	Benar	adi	Benar
85	adi_0_4065.jpeg	adi	adi	Benar	adi	Benar
86	adi_0_1710.jpeg	adi	adi	Benar	adi	Benar
87	adi_0_2296.jpeg	adi	adi	Benar	adi	Benar
88	adi_0_2898.jpeg	adi	adi	Benar	adi	Benar
89	adi_0_3369.jpeg	adi	adi	Benar	adi	Benar
90	adi_0_4088.jpeg	adi	adi	Benar	adi	Benar
91	adi_0_4674.jpeg	adi	adi	Benar	adi	Benar
92	adi_0_5321.jpeg	adi	adi	Benar	adi	Benar
93	adi_0_5928.jpeg	adi	adi	Benar	adi	Benar
94	adi_0_6406.jpeg	adi	adi	Benar	adi	Benar
95	adi_0_6950.jpeg	adi	adi	Benar	adi	Benar
96	adi_0_7658.jpeg	adi	adi	Benar	adi	Benar
97	adi_0_8221.jpeg	adi	adi	Benar	adi	Benar
98	adi_0_9027.jpeg	adi	adi	Benar	adi	Benar
99	adi_0_4097.jpeg	adi	adi	Benar	adi	Benar
100	adi_0_4105.jpeg	adi	adi	Benar	adi	Benar
101	adi_0_4126.jpeg	adi	adi	Benar	adi	Benar
102	adi_0_4173.jpeg	adi	adi	Benar	adi	Benar
103	adi_0_4240.jpeg	adi	adi	Benar	adi	Benar
104	adi_0_4244.jpeg	adi	adi	Benar	adi	Benar
105	adi_0_4253.jpeg	adi	adi	Benar	adi	Benar
106	adi_0_4292.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
107	adi_0_4336.jpeg	adi	adi	Benar	adi	Benar
108	adi_0_4337.jpeg	adi	adi	Benar	adi	Benar
109	adi_0_4344.jpeg	adi	adi	Benar	adi	Benar
110	adi_0_4412.jpeg	adi	adi	Benar	adi	Benar
111	adi_0_4508.jpeg	adi	adi	Benar	adi	Benar
112	adi_0_4562.jpeg	adi	adi	Benar	adi	Benar
113	adi_0_4613.jpeg	adi	adi	Benar	adi	Benar
114	adi_0_4614.jpeg	adi	adi	Benar	adi	Benar
115	adi_0_467.jpeg	adi	adi	Benar	adi	Benar
116	adi_0_4681.jpeg	adi	adi	Benar	adi	Benar
117	adi_0_4698.jpeg	adi	adi	Benar	adi	Benar
118	adi_0_4753.jpeg	adi	adi	Benar	adi	Benar
119	adi_0_4754.jpeg	adi	adi	Benar	adi	Benar
120	adi_0_4758.jpeg	adi	adi	Benar	adi	Benar
121	adi_0_4771.jpeg	adi	adi	Benar	adi	Benar
122	adi_0_481.jpeg	adi	adi	Benar	adi	Benar
123	adi_0_4847.jpeg	adi	adi	Benar	adi	Benar
124	adi_0_4886.jpeg	adi	adi	Benar	adi	Benar
125	adi_0_4969.jpeg	adi	adi	Benar	adi	Benar
126	adi_0_5055.jpeg	adi	adi	Benar	adi	Benar
127	adi_0_5077.jpeg	adi	adi	Benar	adi	Benar
128	adi_0_5122.jpeg	adi	adi	Benar	adi	Benar
129	adi_0_5125.jpeg	adi	adi	Benar	adi	Benar
130	adi_0_5174.jpeg	adi	adi	Benar	adi	Benar
131	adi_0_5185.jpeg	adi	adi	Benar	adi	Benar
132	adi_0_5188.jpeg	adi	adi	Benar	adi	Benar
133	adi_0_5333.jpeg	adi	adi	Benar	adi	Benar
134	adi_0_5397.jpeg	adi	adi	Benar	adi	Benar
135	adi_0_5492.jpeg	adi	adi	Benar	adi	Benar
136	adi_0_5531.jpeg	adi	adi	Benar	adi	Benar
137	adi_0_5548.jpeg	adi	adi	Benar	adi	Benar
138	adi_0_5568.jpeg	adi	adi	Benar	adi	Benar
139	adi_0_5574.jpeg	adi	adi	Benar	adi	Benar
140	adi_0_5634.jpeg	adi	adi	Benar	adi	Benar
141	adi_0_5642.jpeg	adi	adi	Benar	adi	Benar
142	adi_0_5644.jpeg	adi	adi	Benar	adi	Benar
143	adi_0_5661.jpeg	adi	adi	Benar	adi	Benar
144	adi_0_5671.jpeg	adi	adi	Benar	adi	Benar
145	adi_0_5677.jpeg	adi	adi	Benar	adi	Benar
146	adi_0_5738.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
147	adi_0_5777.jpeg	adi	adi	Benar	adi	Benar
148	adi_0_5784.jpeg	adi	adi	Benar	adi	Benar
149	adi_0_5880.jpeg	adi	adi	Benar	adi	Benar
150	adi_0_5946.jpeg	adi	adi	Benar	adi	Benar
151	adi_0_5964.jpeg	adi	adi	Benar	adi	Benar
152	adi_0_597.jpeg	adi	adi	Benar	adi	Benar
153	adi_0_5976.jpeg	adi	adi	Benar	adi	Benar
154	adi_0_6000.jpeg	adi	adi	Benar	adi	Benar
155	adi_0_6042.jpeg	adi	adi	Benar	adi	Benar
156	adi_0_608.jpeg	adi	adi	Benar	adi	Benar
157	adi_0_6082.jpeg	adi	adi	Benar	adi	Benar
158	adi_0_6086.jpeg	adi	adi	Benar	adi	Benar
159	adi_0_6113.jpeg	adi	adi	Benar	adi	Benar
160	adi_0_6161.jpeg	adi	adi	Benar	adi	Benar
161	adi_0_617.jpeg	adi	adi	Benar	adi	Benar
162	adi_0_6179.jpeg	adi	adi	Benar	adi	Benar
163	adi_0_631.jpeg	adi	adi	Benar	adi	Benar
164	adi_0_6329.jpeg	adi	adi	Benar	adi	Benar
165	adi_0_6367.jpeg	adi	adi	Benar	adi	Benar
166	adi_0_6376.jpeg	adi	adi	Benar	adi	Benar
167	adi_0_6408.jpeg	adi	adi	Benar	adi	Benar
168	adi_0_6423.jpeg	adi	adi	Benar	adi	Benar
169	adi_0_6434.jpeg	adi	adi	Benar	adi	Benar
170	adi_0_6448.jpeg	adi	adi	Benar	adi	Benar
171	adi_0_6528.jpeg	adi	adi	Benar	adi	Benar
172	adi_0_6576.jpeg	adi	adi	Benar	adi	Benar
173	adi_0_669.jpeg	adi	adi	Benar	adi	Benar
174	adi_0_6691.jpeg	adi	adi	Benar	adi	Benar
175	adi_0_6697.jpeg	adi	adi	Benar	adi	Benar
176	adi_0_6733.jpeg	adi	adi	Benar	adi	Benar
177	adi_0_6750.jpeg	adi	adi	Benar	adi	Benar
178	adi_0_6801.jpeg	adi	adi	Benar	adi	Benar
179	adi_0_6829.jpeg	adi	adi	Benar	adi	Benar
180	adi_0_6840.jpeg	adi	adi	Benar	adi	Benar
181	adi_0_689.jpeg	adi	adi	Benar	adi	Benar
182	adi_0_6897.jpeg	adi	adi	Benar	adi	Benar
183	adi_0_6949.jpeg	adi	adi	Benar	adi	Benar
184	adi_0_6960.jpeg	adi	adi	Benar	adi	Benar
185	adi_0_6973.jpeg	adi	adi	Benar	adi	Benar
186	adi_0_6981.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
187	adi_0_70.jpeg	adi	adi	Benar	adi	Benar
188	adi_0_7009.jpeg	adi	adi	Benar	adi	Benar
189	adi_0_7030.jpeg	adi	adi	Benar	adi	Benar
190	adi_0_7064.jpeg	adi	adi	Benar	adi	Benar
191	adi_0_7068.jpeg	adi	adi	Benar	adi	Benar
192	adi_0_7123.jpeg	adi	adi	Benar	adi	Benar
193	adi_0_7209.jpeg	adi	adi	Benar	adi	Benar
194	adi_0_7303.jpeg	adi	adi	Benar	adi	Benar
195	adi_0_7319.jpeg	adi	adi	Benar	adi	Benar
196	adi_0_7435.jpeg	adi	adi	Benar	adi	Benar
197	adi_0_7477.jpeg	adi	adi	Benar	adi	Benar
198	adi_0_7493.jpeg	adi	adi	Benar	adi	Benar
199	adi_0_7528.jpeg	adi	adi	Benar	adi	Benar
200	adi_0_7614.jpeg	adi	adi	Benar	adi	Benar
201	adi_0_7721.jpeg	adi	adi	Benar	adi	Benar
202	adi_0_7753.jpeg	adi	adi	Benar	adi	Benar
203	adi_0_7813.jpeg	adi	adi	Benar	adi	Benar
204	adi_0_7827.jpeg	adi	adi	Benar	adi	Benar
205	adi_0_7889.jpeg	adi	adi	Benar	adi	Benar
206	adi_0_7903.jpeg	adi	adi	Benar	adi	Benar
207	adi_0_7936.jpeg	adi	adi	Benar	adi	Benar
208	adi_0_7941.jpeg	adi	adi	Benar	adi	Benar
209	adi_0_7972.jpeg	adi	adi	Benar	adi	Benar
210	adi_0_7981.jpeg	adi	adi	Benar	adi	Benar
211	adi_0_7990.jpeg	adi	adi	Benar	adi	Benar
212	adi_0_8000.jpeg	adi	adi	Benar	adi	Benar
213	adi_0_8058.jpeg	adi	adi	Benar	adi	Benar
214	adi_0_8085.jpeg	adi	adi	Benar	adi	Benar
215	adi_0_8106.jpeg	adi	adi	Benar	adi	Benar
216	adi_0_8194.jpeg	adi	adi	Benar	adi	Benar
217	adi_0_8216.jpeg	adi	adi	Benar	adi	Benar
218	adi_0_8268.jpeg	adi	adi	Benar	adi	Benar
219	adi_0_8317.jpeg	adi	adi	Benar	adi	Benar
220	adi_0_8330.jpeg	adi	adi	Benar	adi	Benar
221	adi_0_8332.jpeg	adi	adi	Benar	imam	Salah
222	adi_0_8350.jpeg	adi	adi	Benar	adi	Benar
223	adi_0_8436.jpeg	adi	adi	Benar	adi	Benar
224	adi_0_8560.jpeg	adi	adi	Benar	adi	Benar
225	adi_0_8628.jpeg	adi	adi	Benar	adi	Benar
226	adi_0_8660.jpeg	adi	adi	Benar	adi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
227	adi_0_8663.jpeg	adi	adi	Benar	adi	Benar
228	adi_0_8691.jpeg	adi	adi	Benar	adi	Benar
229	adi_0_8709.jpeg	adi	adi	Benar	adi	Benar
230	adi_0_8713.jpeg	adi	adi	Benar	adi	Benar
231	adi_0_8721.jpeg	adi	adi	Benar	adi	Benar
232	adi_0_8799.jpeg	adi	adi	Benar	adi	Benar
233	adi_0_8836.jpeg	adi	adi	Benar	adi	Benar
234	adi_0_8989.jpeg	adi	adi	Benar	adi	Benar
235	adi_0_9229.jpeg	adi	adi	Benar	adi	Benar
236	adi_0_9241.jpeg	adi	adi	Benar	adi	Benar
237	adi_0_9316.jpeg	adi	adi	Benar	adi	Benar
238	adi_0_9332.jpeg	adi	adi	Benar	adi	Benar
239	adi_0_9351.jpeg	adi	adi	Benar	adi	Benar
240	adi_0_9380.jpeg	adi	adi	Benar	adi	Benar
241	adi_0_9411.jpeg	adi	adi	Benar	adi	Benar
242	adi_0_9433.jpeg	adi	adi	Benar	adi	Benar
243	adi_0_9460.jpeg	adi	adi	Benar	adi	Benar
244	adi_0_9499.jpeg	adi	adi	Benar	adi	Benar
245	adi_0_9556.jpeg	adi	adi	Benar	adi	Benar
246	adi_0_9636.jpeg	adi	adi	Benar	adi	Benar
247	adi_0_9756.jpeg	adi	adi	Benar	adi	Benar
248	adi_0_9824.jpeg	adi	imam	Salah	adi	Benar
249	adi_0_9829.jpeg	adi	adi	Benar	adi	Benar
250	adi_0_9831.jpeg	adi	adi	Benar	adi	Benar
251	adi_0_9885.jpeg	adi	adi	Benar	adi	Benar
252	adi_0_9938.jpeg	adi	adi	Benar	adi	Benar
253	ajrul_0_1038.jpeg	ajrul	ajrul	Benar	ajrul	Benar
254	ajrul_0_1049.jpeg	ajrul	ajrul	Benar	ajrul	Benar
255	ajrul_0_1146.jpeg	ajrul	ajrul	Benar	ajrul	Benar
256	ajrul_0_1150.jpeg	ajrul	ajrul	Benar	ajrul	Benar
257	ajrul_0_1313.jpeg	ajrul	ajrul	Benar	ajrul	Benar
258	ajrul_0_1344.jpeg	ajrul	ajrul	Benar	ajrul	Benar
259	ajrul_0_1372.jpeg	ajrul	ajrul	Benar	ajrul	Benar
260	ajrul_0_1388.jpeg	ajrul	ajrul	Benar	ajrul	Benar
261	ajrul_0_1399.jpeg	ajrul	ajrul	Benar	ajrul	Benar
262	ajrul_0_1404.jpeg	ajrul	ajrul	Benar	ajrul	Benar
263	ajrul_0_1418.jpeg	ajrul	ajrul	Benar	ajrul	Benar
264	ajrul_0_1421.jpeg	ajrul	imam	Salah	ajrul	Benar
265	ajrul_0_1439.jpeg	ajrul	ajrul	Benar	ajrul	Benar
266	ajrul_0_1484.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
267	ajrul_0_1526.jpeg	ajrul	ajrul	Benar	ajrul	Benar
268	ajrul_0_1613.jpeg	ajrul	ajrul	Benar	ajrul	Benar
269	ajrul_0_168.jpeg	ajrul	ajrul	Benar	ajrul	Benar
270	ajrul_0_1739.jpeg	ajrul	ajrul	Benar	ajrul	Benar
271	ajrul_0_1758.jpeg	ajrul	ajrul	Benar	ajrul	Benar
272	ajrul_0_1782.jpeg	ajrul	ajrul	Benar	ajrul	Benar
273	ajrul_0_1785.jpeg	ajrul	ajrul	Benar	ajrul	Benar
274	ajrul_0_1865.jpeg	ajrul	ajrul	Benar	ajrul	Benar
275	ajrul_0_1874.jpeg	ajrul	ajrul	Benar	ajrul	Benar
276	ajrul_0_1893.jpeg	ajrul	ajrul	Benar	ajrul	Benar
277	ajrul_0_1909.jpeg	ajrul	ajrul	Benar	ajrul	Benar
278	ajrul_0_1931.jpeg	ajrul	ajrul	Benar	ajrul	Benar
279	ajrul_0_2014.jpeg	ajrul	ajrul	Benar	ajrul	Benar
280	ajrul_0_2025.jpeg	ajrul	ajrul	Benar	ajrul	Benar
281	ajrul_0_2030.jpeg	ajrul	ajrul	Benar	ajrul	Benar
282	ajrul_0_2047.jpeg	ajrul	ajrul	Benar	ajrul	Benar
283	ajrul_0_2049.jpeg	ajrul	ajrul	Benar	ajrul	Benar
284	ajrul_0_2121.jpeg	ajrul	imam	Salah	ajrul	Benar
285	ajrul_0_2186.jpeg	ajrul	ajrul	Benar	ajrul	Benar
286	ajrul_0_219.jpeg	ajrul	ajrul	Benar	ajrul	Benar
287	ajrul_0_2200.jpeg	ajrul	ajrul	Benar	ajrul	Benar
288	ajrul_0_2306.jpeg	ajrul	ajrul	Benar	ajrul	Benar
289	ajrul_0_2326.jpeg	ajrul	ajrul	Benar	ajrul	Benar
290	ajrul_0_2357.jpeg	ajrul	ajrul	Benar	ajrul	Benar
291	ajrul_0_2393.jpeg	ajrul	ajrul	Benar	ajrul	Benar
292	ajrul_0_2484.jpeg	ajrul	ajrul	Benar	ajrul	Benar
293	ajrul_0_2535.jpeg	ajrul	ajrul	Benar	ajrul	Benar
294	ajrul_0_2675.jpeg	ajrul	ajrul	Benar	ajrul	Benar
295	ajrul_0_2678.jpeg	ajrul	ajrul	Benar	ajrul	Benar
296	ajrul_0_2688.jpeg	ajrul	ajrul	Benar	ajrul	Benar
297	ajrul_0_2726.jpeg	ajrul	ajrul	Benar	ajrul	Benar
298	ajrul_0_2741.jpeg	ajrul	ajrul	Benar	ajrul	Benar
299	ajrul_0_2743.jpeg	ajrul	ajrul	Benar	ajrul	Benar
300	ajrul_0_2867.jpeg	ajrul	ajrul	Benar	ajrul	Benar
301	ajrul_0_291.jpeg	ajrul	ajrul	Benar	ajrul	Benar
302	ajrul_0_2927.jpeg	ajrul	ajrul	Benar	ajrul	Benar
303	ajrul_0_2937.jpeg	ajrul	ajrul	Benar	ajrul	Benar
304	ajrul_0_3007.jpeg	ajrul	ajrul	Benar	ajrul	Benar
305	ajrul_0_3018.jpeg	ajrul	ajrul	Benar	ajrul	Benar
306	ajrul_0_3031.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
307	ajrul_0_306.jpeg	ajrul	ajrul	Benar	ajrul	Benar
308	ajrul_0_3062.jpeg	ajrul	ajrul	Benar	ajrul	Benar
309	ajrul_0_3081.jpeg	ajrul	ajrul	Benar	ajrul	Benar
310	ajrul_0_3092.jpeg	ajrul	ajrul	Benar	ajrul	Benar
311	ajrul_0_3149.jpeg	ajrul	ajrul	Benar	ajrul	Benar
312	ajrul_0_3154.jpeg	ajrul	ajrul	Benar	ajrul	Benar
313	ajrul_0_3160.jpeg	ajrul	ajrul	Benar	ajrul	Benar
314	ajrul_0_3183.jpeg	ajrul	ajrul	Benar	ajrul	Benar
315	ajrul_0_3194.jpeg	ajrul	ajrul	Benar	ajrul	Benar
316	ajrul_0_3196.jpeg	ajrul	ajrul	Benar	ajrul	Benar
317	ajrul_0_1645.jpeg	ajrul	ajrul	Benar	ajrul	Benar
318	ajrul_0_2172.jpeg	ajrul	ajrul	Benar	ajrul	Benar
319	ajrul_0_2909.jpeg	ajrul	ajrul	Benar	ajrul	Benar
320	ajrul_0_3211.jpeg	ajrul	ajrul	Benar	ajrul	Benar
321	ajrul_0_3881.jpeg	ajrul	ajrul	Benar	ajrul	Benar
322	ajrul_0_4432.jpeg	ajrul	ajrul	Benar	ajrul	Benar
323	ajrul_0_543.jpeg	ajrul	ajrul	Benar	ajrul	Benar
324	ajrul_0_5916.jpeg	ajrul	ajrul	Benar	ajrul	Benar
325	ajrul_0_6416.jpeg	ajrul	ajrul	Benar	ajrul	Benar
326	ajrul_0_7157.jpeg	ajrul	ajrul	Benar	ajrul	Benar
327	ajrul_0_7943.jpeg	ajrul	ajrul	Benar	ajrul	Benar
328	ajrul_0_8445.jpeg	ajrul	ajrul	Benar	ajrul	Benar
329	ajrul_0_8904.jpeg	ajrul	ajrul	Benar	ajrul	Benar
330	ajrul_0_323.jpeg	ajrul	ajrul	Benar	ajrul	Benar
331	ajrul_0_3251.jpeg	ajrul	ajrul	Benar	ajrul	Benar
332	ajrul_0_3259.jpeg	ajrul	ajrul	Benar	ajrul	Benar
333	ajrul_0_33.jpeg	ajrul	ajrul	Benar	ajrul	Benar
334	ajrul_0_3393.jpeg	ajrul	ajrul	Benar	ajrul	Benar
335	ajrul_0_3554.jpeg	ajrul	ajrul	Benar	ajrul	Benar
336	ajrul_0_3585.jpeg	ajrul	ajrul	Benar	ajrul	Benar
337	ajrul_0_3596.jpeg	ajrul	ajrul	Benar	ajrul	Benar
338	ajrul_0_369.jpeg	ajrul	adi	Salah	ajrul	Benar
339	ajrul_0_3715.jpeg	ajrul	ajrul	Benar	ajrul	Benar
340	ajrul_0_372.jpeg	ajrul	ajrul	Benar	ajrul	Benar
341	ajrul_0_3734.jpeg	ajrul	ajrul	Benar	ajrul	Benar
342	ajrul_0_3755.jpeg	ajrul	ajrul	Benar	ajrul	Benar
343	ajrul_0_3758.jpeg	ajrul	ajrul	Benar	ajrul	Benar
344	ajrul_0_3848.jpeg	ajrul	ajrul	Benar	ajrul	Benar
345	ajrul_0_3877.jpeg	ajrul	ajrul	Benar	ajrul	Benar
346	ajrul_0_3885.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
347	ajrul_0_3943.jpeg	ajrul	ajrul	Benar	ajrul	Benar
348	ajrul_0_4000.jpeg	ajrul	ajrul	Benar	ajrul	Benar
349	ajrul_0_4044.jpeg	ajrul	ajrul	Benar	ajrul	Benar
350	ajrul_0_4154.jpeg	ajrul	ajrul	Benar	ajrul	Benar
351	ajrul_0_4156.jpeg	ajrul	ajrul	Benar	ajrul	Benar
352	ajrul_0_4187.jpeg	ajrul	ajrul	Benar	ajrul	Benar
353	ajrul_0_4196.jpeg	ajrul	ajrul	Benar	ajrul	Benar
354	ajrul_0_4271.jpeg	ajrul	ajrul	Benar	ajrul	Benar
355	ajrul_0_4285.jpeg	ajrul	ajrul	Benar	ajrul	Benar
356	ajrul_0_4307.jpeg	ajrul	ajrul	Benar	ajrul	Benar
357	ajrul_0_4326.jpeg	ajrul	ajrul	Benar	ajrul	Benar
358	ajrul_0_434.jpeg	ajrul	ajrul	Benar	ajrul	Benar
359	ajrul_0_4346.jpeg	ajrul	ajrul	Benar	ajrul	Benar
360	ajrul_0_4404.jpeg	ajrul	ajrul	Benar	ajrul	Benar
361	ajrul_0_4411.jpeg	ajrul	ajrul	Benar	ajrul	Benar
362	ajrul_0_4436.jpeg	ajrul	ajrul	Benar	ajrul	Benar
363	ajrul_0_444.jpeg	ajrul	ajrul	Benar	ajrul	Benar
364	ajrul_0_4455.jpeg	ajrul	ajrul	Benar	ajrul	Benar
365	ajrul_0_4503.jpeg	ajrul	ajrul	Benar	ajrul	Benar
366	ajrul_0_4580.jpeg	ajrul	ajrul	Benar	ajrul	Benar
367	ajrul_0_4599.jpeg	ajrul	ajrul	Benar	ajrul	Benar
368	ajrul_0_4655.jpeg	ajrul	ajrul	Benar	ajrul	Benar
369	ajrul_0_4698.jpeg	ajrul	ajrul	Benar	ajrul	Benar
370	ajrul_0_4872.jpeg	ajrul	ajrul	Benar	ajrul	Benar
371	ajrul_0_4934.jpeg	ajrul	ajrul	Benar	ajrul	Benar
372	ajrul_0_5184.jpeg	ajrul	ajrul	Benar	ajrul	Benar
373	ajrul_0_5264.jpeg	ajrul	ajrul	Benar	ajrul	Benar
374	ajrul_0_5291.jpeg	ajrul	adi	Salah	ajrul	Benar
375	ajrul_0_5305.jpeg	ajrul	ajrul	Benar	ajrul	Benar
376	ajrul_0_533.jpeg	ajrul	ajrul	Benar	ajrul	Benar
377	ajrul_0_5381.jpeg	ajrul	ajrul	Benar	ajrul	Benar
378	ajrul_0_545.jpeg	ajrul	ajrul	Benar	ajrul	Benar
379	ajrul_0_5488.jpeg	ajrul	ajrul	Benar	ajrul	Benar
380	ajrul_0_5575.jpeg	ajrul	ajrul	Benar	ajrul	Benar
381	ajrul_0_5588.jpeg	ajrul	ajrul	Benar	ajrul	Benar
382	ajrul_0_562.jpeg	ajrul	ajrul	Benar	ajrul	Benar
383	ajrul_0_5620.jpeg	ajrul	ajrul	Benar	ajrul	Benar
384	ajrul_0_5672.jpeg	ajrul	ajrul	Benar	ajrul	Benar
385	ajrul_0_5683.jpeg	ajrul	ajrul	Benar	ajrul	Benar
386	ajrul_0_5691.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
387	ajrul_0_5769.jpeg	ajrul	ajrul	Benar	ajrul	Benar
388	ajrul_0_5793.jpeg	ajrul	ajrul	Benar	ajrul	Benar
389	ajrul_0_581.jpeg	ajrul	ajrul	Benar	ajrul	Benar
390	ajrul_0_5859.jpeg	ajrul	ajrul	Benar	ajrul	Benar
391	ajrul_0_5880.jpeg	ajrul	ajrul	Benar	ajrul	Benar
392	ajrul_0_5883.jpeg	ajrul	ajrul	Benar	ajrul	Benar
393	ajrul_0_5893.jpeg	ajrul	ajrul	Benar	ajrul	Benar
394	ajrul_0_5929.jpeg	ajrul	ajrul	Benar	ajrul	Benar
395	ajrul_0_5990.jpeg	ajrul	ajrul	Benar	ajrul	Benar
396	ajrul_0_6002.jpeg	ajrul	ajrul	Benar	ajrul	Benar
397	ajrul_0_6091.jpeg	ajrul	ajrul	Benar	ajrul	Benar
398	ajrul_0_615.jpeg	ajrul	ajrul	Benar	ajrul	Benar
399	ajrul_0_6156.jpeg	ajrul	ajrul	Benar	ajrul	Benar
400	ajrul_0_62.jpeg	ajrul	ajrul	Benar	ajrul	Benar
401	ajrul_0_6231.jpeg	ajrul	ajrul	Benar	ajrul	Benar
402	ajrul_0_6255.jpeg	ajrul	ajrul	Benar	ajrul	Benar
403	ajrul_0_6318.jpeg	ajrul	ajrul	Benar	ajrul	Benar
404	ajrul_0_6336.jpeg	ajrul	ajrul	Benar	ajrul	Benar
405	ajrul_0_6375.jpeg	ajrul	ajrul	Benar	ajrul	Benar
406	ajrul_0_6387.jpeg	ajrul	ajrul	Benar	ajrul	Benar
407	ajrul_0_640.jpeg	ajrul	ajrul	Benar	ajrul	Benar
408	ajrul_0_6406.jpeg	ajrul	ajrul	Benar	ajrul	Benar
409	ajrul_0_6407.jpeg	ajrul	ajrul	Benar	ajrul	Benar
410	ajrul_0_6440.jpeg	ajrul	ajrul	Benar	ajrul	Benar
411	ajrul_0_6455.jpeg	ajrul	ajrul	Benar	ajrul	Benar
412	ajrul_0_6462.jpeg	ajrul	ajrul	Benar	ajrul	Benar
413	ajrul_0_6492.jpeg	ajrul	ajrul	Benar	ajrul	Benar
414	ajrul_0_6527.jpeg	ajrul	ajrul	Benar	ajrul	Benar
415	ajrul_0_653.jpeg	ajrul	adi	Salah	ajrul	Benar
416	ajrul_0_6539.jpeg	ajrul	ajrul	Benar	ajrul	Benar
417	ajrul_0_6590.jpeg	ajrul	ajrul	Benar	ajrul	Benar
418	ajrul_0_6620.jpeg	ajrul	ajrul	Benar	ajrul	Benar
419	ajrul_0_675.jpeg	ajrul	ajrul	Benar	ajrul	Benar
420	ajrul_0_6757.jpeg	ajrul	ajrul	Benar	ajrul	Benar
421	ajrul_0_6910.jpeg	ajrul	ajrul	Benar	ajrul	Benar
422	ajrul_0_696.jpeg	ajrul	ajrul	Benar	ajrul	Benar
423	ajrul_0_6964.jpeg	ajrul	ajrul	Benar	ajrul	Benar
424	ajrul_0_6981.jpeg	ajrul	imam	Salah	ajrul	Benar
425	ajrul_0_7070.jpeg	ajrul	ajrul	Benar	ajrul	Benar
426	ajrul_0_7239.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
427	ajrul_0_7282.jpeg	ajrul	ajrul	Benar	ajrul	Benar
428	ajrul_0_7328.jpeg	ajrul	ajrul	Benar	ajrul	Benar
429	ajrul_0_7364.jpeg	ajrul	ajrul	Benar	ajrul	Benar
430	ajrul_0_744.jpeg	ajrul	ajrul	Benar	ajrul	Benar
431	ajrul_0_7471.jpeg	ajrul	ajrul	Benar	ajrul	Benar
432	ajrul_0_7516.jpeg	ajrul	ajrul	Benar	ajrul	Benar
433	ajrul_0_758.jpeg	ajrul	ajrul	Benar	ajrul	Benar
434	ajrul_0_7612.jpeg	ajrul	ajrul	Benar	ajrul	Benar
435	ajrul_0_7649.jpeg	ajrul	ajrul	Benar	ajrul	Benar
436	ajrul_0_7651.jpeg	ajrul	ajrul	Benar	ajrul	Benar
437	ajrul_0_7684.jpeg	ajrul	ajrul	Benar	ajrul	Benar
438	ajrul_0_7763.jpeg	ajrul	ajrul	Benar	ajrul	Benar
439	ajrul_0_7877.jpeg	ajrul	ajrul	Benar	ajrul	Benar
440	ajrul_0_7880.jpeg	ajrul	ajrul	Benar	ajrul	Benar
441	ajrul_0_7940.jpeg	ajrul	ajrul	Benar	ajrul	Benar
442	ajrul_0_8020.jpeg	ajrul	ajrul	Benar	ajrul	Benar
443	ajrul_0_8046.jpeg	ajrul	ajrul	Benar	ajrul	Benar
444	ajrul_0_8054.jpeg	ajrul	ajrul	Benar	ajrul	Benar
445	ajrul_0_8078.jpeg	ajrul	ajrul	Benar	ajrul	Benar
446	ajrul_0_8103.jpeg	ajrul	ajrul	Benar	ajrul	Benar
447	ajrul_0_8133.jpeg	ajrul	ajrul	Benar	ajrul	Benar
448	ajrul_0_8147.jpeg	ajrul	ajrul	Benar	ajrul	Benar
449	ajrul_0_8239.jpeg	ajrul	ajrul	Benar	ajrul	Benar
450	ajrul_0_8263.jpeg	ajrul	ajrul	Benar	ajrul	Benar
451	ajrul_0_8300.jpeg	ajrul	ajrul	Benar	ajrul	Benar
452	ajrul_0_8321.jpeg	ajrul	ajrul	Benar	ajrul	Benar
453	ajrul_0_8337.jpeg	ajrul	ajrul	Benar	ajrul	Benar
454	ajrul_0_8347.jpeg	ajrul	ajrul	Benar	ajrul	Benar
455	ajrul_0_8371.jpeg	ajrul	ajrul	Benar	ajrul	Benar
456	ajrul_0_842.jpeg	ajrul	ajrul	Benar	ajrul	Benar
457	ajrul_0_8430.jpeg	ajrul	ajrul	Benar	ajrul	Benar
458	ajrul_0_8447.jpeg	ajrul	ajrul	Benar	ajrul	Benar
459	ajrul_0_8492.jpeg	ajrul	ajrul	Benar	ajrul	Benar
460	ajrul_0_8494.jpeg	ajrul	ajrul	Benar	ajrul	Benar
461	ajrul_0_8511.jpeg	ajrul	ajrul	Benar	ajrul	Benar
462	ajrul_0_8554.jpeg	ajrul	ajrul	Benar	ajrul	Benar
463	ajrul_0_8566.jpeg	ajrul	ajrul	Benar	ajrul	Benar
464	ajrul_0_8567.jpeg	ajrul	ajrul	Benar	ajrul	Benar
465	ajrul_0_8603.jpeg	ajrul	ajrul	Benar	ajrul	Benar
466	ajrul_0_8611.jpeg	ajrul	ajrul	Benar	ajrul	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
467	ajrul_0_8626.jpeg	ajrul	ajrul	Benar	ajrul	Benar
468	ajrul_0_8676.jpeg	ajrul	ajrul	Benar	ajrul	Benar
469	ajrul_0_8727.jpeg	ajrul	ajrul	Benar	ajrul	Benar
470	ajrul_0_8744.jpeg	ajrul	ajrul	Benar	ajrul	Benar
471	ajrul_0_8818.jpeg	ajrul	ajrul	Benar	ajrul	Benar
472	ajrul_0_8870.jpeg	ajrul	ajrul	Benar	ajrul	Benar
473	ajrul_0_8882.jpeg	ajrul	ajrul	Benar	ajrul	Benar
474	ajrul_0_8916.jpeg	ajrul	ajrul	Benar	ajrul	Benar
475	ajrul_0_8953.jpeg	ajrul	ajrul	Benar	ajrul	Benar
476	ajrul_0_896.jpeg	ajrul	ajrul	Benar	ajrul	Benar
477	ajrul_0_8996.jpeg	ajrul	ajrul	Benar	ajrul	Benar
478	ajrul_0_9027.jpeg	ajrul	ajrul	Benar	ajrul	Benar
479	ajrul_0_9068.jpeg	ajrul	ajrul	Benar	ajrul	Benar
480	ajrul_0_9081.jpeg	ajrul	ajrul	Benar	ajrul	Benar
481	ajrul_0_9115.jpeg	ajrul	ajrul	Benar	ajrul	Benar
482	ajrul_0_9150.jpeg	ajrul	ajrul	Benar	ajrul	Benar
483	ajrul_0_9166.jpeg	ajrul	imam	Salah	ajrul	Benar
484	ajrul_0_9260.jpeg	ajrul	ajrul	Benar	ajrul	Benar
485	ajrul_0_9269.jpeg	ajrul	ajrul	Benar	ajrul	Benar
486	ajrul_0_9369.jpeg	ajrul	ajrul	Benar	ajrul	Benar
487	ajrul_0_9396.jpeg	ajrul	ajrul	Benar	ajrul	Benar
488	ajrul_0_9424.jpeg	ajrul	ajrul	Benar	ajrul	Benar
489	ajrul_0_9428.jpeg	ajrul	imam	Salah	ajrul	Benar
490	ajrul_0_9431.jpeg	ajrul	ajrul	Benar	ajrul	Benar
491	ajrul_0_9434.jpeg	ajrul	ajrul	Benar	ajrul	Benar
492	ajrul_0_9446.jpeg	ajrul	ajrul	Benar	ajrul	Benar
493	ajrul_0_9609.jpeg	ajrul	ajrul	Benar	ajrul	Benar
494	ajrul_0_9624.jpeg	ajrul	ajrul	Benar	ajrul	Benar
495	ajrul_0_9651.jpeg	ajrul	ajrul	Benar	ajrul	Benar
496	ajrul_0_9769.jpeg	ajrul	ajrul	Benar	ajrul	Benar
497	ajrul_0_9772.jpeg	ajrul	ajrul	Benar	ajrul	Benar
498	ajrul_0_982.jpeg	ajrul	ajrul	Benar	ajrul	Benar
499	ajrul_0_9820.jpeg	ajrul	ajrul	Benar	ajrul	Benar
500	ajrul_0_9853.jpeg	ajrul	ajrul	Benar	ajrul	Benar
501	ajrul_0_9874.jpeg	ajrul	ajrul	Benar	ajrul	Benar
502	ajrul_0_9919.jpeg	ajrul	ajrul	Benar	ajrul	Benar
503	ari_0_1035.jpeg	ari	ari	Benar	ari	Benar
504	ari_0_1055.jpeg	ari	ari	Benar	ari	Benar
505	ari_0_106.jpeg	ari	ari	Benar	ari	Benar
506	ari_0_1065.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
507	ari_0_1129.jpeg	ari	ari	Benar	ari	Benar
508	ari_0_1156.jpeg	ari	ari	Benar	ari	Benar
509	ari_0_1160.jpeg	ari	ari	Benar	ari	Benar
510	ari_0_1239.jpeg	ari	ari	Benar	ari	Benar
511	ari_0_1243.jpeg	ari	ari	Benar	ari	Benar
512	ari_0_1377.jpeg	ari	ari	Benar	ari	Benar
513	ari_0_1436.jpeg	ari	ari	Benar	ari	Benar
514	ari_0_1508.jpeg	ari	ari	Benar	ari	Benar
515	ari_0_1533.jpeg	ari	ari	Benar	ari	Benar
516	ari_0_1560.jpeg	ari	ari	Benar	ari	Benar
517	ari_0_1579.jpeg	ari	ari	Benar	ari	Benar
518	ari_0_1634.jpeg	ari	ari	Benar	ari	Benar
519	ari_0_1638.jpeg	ari	ari	Benar	ari	Benar
520	ari_0_1837.jpeg	ari	ari	Benar	ari	Benar
521	ari_0_1862.jpeg	ari	ari	Benar	ari	Benar
522	ari_0_1866.jpeg	ari	ari	Benar	ari	Benar
523	ari_0_1871.jpeg	ari	ari	Benar	ari	Benar
524	ari_0_1962.jpeg	ari	ari	Benar	ari	Benar
525	ari_0_1987.jpeg	ari	ari	Benar	ari	Benar
526	ari_0_2053.jpeg	ari	ari	Benar	ari	Benar
527	ari_0_2055.jpeg	ari	ari	Benar	ari	Benar
528	ari_0_2077.jpeg	ari	ari	Benar	ari	Benar
529	ari_0_2079.jpeg	ari	ari	Benar	ari	Benar
530	ari_0_2080.jpeg	ari	ari	Benar	ari	Benar
531	ari_0_2081.jpeg	ari	ari	Benar	ari	Benar
532	ari_0_2101.jpeg	ari	ari	Benar	ari	Benar
533	ari_0_2130.jpeg	ari	ari	Benar	ari	Benar
534	ari_0_2148.jpeg	ari	ari	Benar	ari	Benar
535	ari_0_2213.jpeg	ari	ari	Benar	ari	Benar
536	ari_0_2225.jpeg	ari	ari	Benar	ari	Benar
537	ari_0_2306.jpeg	ari	ari	Benar	ari	Benar
538	ari_0_2334.jpeg	ari	ari	Benar	ari	Benar
539	ari_0_2351.jpeg	ari	ari	Benar	ari	Benar
540	ari_0_2379.jpeg	ari	ari	Benar	ari	Benar
541	ari_0_245.jpeg	ari	ari	Benar	ari	Benar
542	ari_0_2450.jpeg	ari	ari	Benar	ari	Benar
543	ari_0_2476.jpeg	ari	ari	Benar	ari	Benar
544	ari_0_2510.jpeg	ari	ari	Benar	ari	Benar
545	ari_0_2530.jpeg	ari	ari	Benar	ari	Benar
546	ari_0_2572.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
547	ari_0_2593.jpeg	ari	ari	Benar	ari	Benar
548	ari_0_2607.jpeg	ari	ari	Benar	ari	Benar
549	ari_0_2673.jpeg	ari	ari	Benar	ari	Benar
550	ari_0_2734.jpeg	ari	ari	Benar	ari	Benar
551	ari_0_2750.jpeg	ari	ari	Benar	ari	Benar
552	ari_0_2802.jpeg	ari	ari	Benar	ari	Benar
553	ari_0_2819.jpeg	ari	ari	Benar	ari	Benar
554	ari_0_2915.jpeg	ari	ari	Benar	ari	Benar
555	ari_0_2956.jpeg	ari	ari	Benar	ari	Benar
556	ari_0_299.jpeg	ari	ari	Benar	ari	Benar
557	ari_0_3030.jpeg	ari	ari	Benar	ari	Benar
558	ari_0_3036.jpeg	ari	ari	Benar	ari	Benar
559	ari_0_304.jpeg	ari	ari	Benar	ari	Benar
560	ari_0_3071.jpeg	ari	ari	Benar	ari	Benar
561	ari_0_3084.jpeg	ari	ari	Benar	ari	Benar
562	ari_0_31.jpeg	ari	ari	Benar	ari	Benar
563	ari_0_3171.jpeg	ari	ari	Benar	ari	Benar
564	ari_0_3172.jpeg	ari	ari	Benar	ari	Benar
565	ari_0_3195.jpeg	ari	ari	Benar	ari	Benar
566	ari_0_3227.jpeg	ari	ari	Benar	ari	Benar
567	ari_0_328.jpeg	ari	ari	Benar	ari	Benar
568	ari_0_3285.jpeg	ari	ari	Benar	ari	Benar
569	ari_0_3374.jpeg	ari	ari	Benar	ari	Benar
570	ari_0_3404.jpeg	ari	ari	Benar	ari	Benar
571	ari_0_3485.jpeg	ari	ari	Benar	ari	Benar
572	ari_0_355.jpeg	ari	ari	Benar	ari	Benar
573	ari_0_3570.jpeg	ari	ari	Benar	ari	Benar
574	ari_0_3641.jpeg	ari	ari	Benar	ari	Benar
575	ari_0_3692.jpeg	ari	ari	Benar	ari	Benar
576	ari_0_3710.jpeg	ari	ari	Benar	ari	Benar
577	ari_0_3737.jpeg	ari	ari	Benar	ari	Benar
578	ari_0_3864.jpeg	ari	ari	Benar	ari	Benar
579	ari_0_3865.jpeg	ari	ari	Benar	ari	Benar
580	ari_0_3867.jpeg	ari	ari	Benar	ari	Benar
581	ari_0_3880.jpeg	ari	ari	Benar	ari	Benar
582	ari_0_390.jpeg	ari	ari	Benar	ari	Benar
583	ari_0_3932.jpeg	ari	ari	Benar	ari	Benar
584	ari_0_3968.jpeg	ari	ari	Benar	ari	Benar
585	ari_0_3980.jpeg	ari	ari	Benar	ari	Benar
586	ari_0_3998.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
587	ari_0_4031.jpeg	ari	ari	Benar	ari	Benar
588	ari_0_1812.jpeg	ari	ari	Benar	ari	Benar
589	ari_0_2299.jpeg	ari	ari	Benar	ari	Benar
590	ari_0_2855.jpeg	ari	ari	Benar	ari	Benar
591	ari_0_3471.jpeg	ari	ari	Benar	ari	Benar
592	ari_0_4094.jpeg	ari	ari	Benar	ari	Benar
593	ari_0_4750.jpeg	ari	ari	Benar	ari	Benar
594	ari_0_5264.jpeg	ari	ari	Benar	ari	Benar
595	ari_0_6079.jpeg	ari	ari	Benar	ari	Benar
596	ari_0_6555.jpeg	ari	ari	Benar	ari	Benar
597	ari_0_7375.jpeg	ari	ari	Benar	ari	Benar
598	ari_0_8017.jpeg	ari	ari	Benar	ari	Benar
599	ari_0_8603.jpeg	ari	ari	Benar	ari	Benar
600	ari_0_9230.jpeg	ari	ari	Benar	ari	Benar
601	ari_0_4102.jpeg	ari	ari	Benar	ari	Benar
602	ari_0_411.jpeg	ari	ari	Benar	ari	Benar
603	ari_0_412.jpeg	ari	ari	Benar	ari	Benar
604	ari_0_414.jpeg	ari	ari	Benar	ari	Benar
605	ari_0_4148.jpeg	ari	ari	Benar	ari	Benar
606	ari_0_4184.jpeg	ari	ari	Benar	ari	Benar
607	ari_0_4227.jpeg	ari	ari	Benar	ari	Benar
608	ari_0_4244.jpeg	ari	ari	Benar	ari	Benar
609	ari_0_4307.jpeg	ari	ari	Benar	ari	Benar
610	ari_0_4310.jpeg	ari	ari	Benar	ari	Benar
611	ari_0_4357.jpeg	ari	ari	Benar	ari	Benar
612	ari_0_4364.jpeg	ari	ari	Benar	ari	Benar
613	ari_0_4424.jpeg	ari	ari	Benar	ari	Benar
614	ari_0_4446.jpeg	ari	ari	Benar	ari	Benar
615	ari_0_4561.jpeg	ari	ari	Benar	ari	Benar
616	ari_0_4686.jpeg	ari	ari	Benar	ari	Benar
617	ari_0_4730.jpeg	ari	ari	Benar	ari	Benar
618	ari_0_478.jpeg	ari	ari	Benar	ari	Benar
619	ari_0_483.jpeg	ari	ari	Benar	ari	Benar
620	ari_0_4851.jpeg	ari	ari	Benar	ari	Benar
621	ari_0_4856.jpeg	ari	ari	Benar	ari	Benar
622	ari_0_4857.jpeg	ari	ari	Benar	ari	Benar
623	ari_0_4866.jpeg	ari	ari	Benar	ari	Benar
624	ari_0_488.jpeg	ari	ari	Benar	ari	Benar
625	ari_0_4999.jpeg	ari	ari	Benar	ari	Benar
626	ari_0_5020.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
627	ari_0_5051.jpeg	ari	ari	Benar	ari	Benar
628	ari_0_5080.jpeg	ari	ari	Benar	ari	Benar
629	ari_0_5109.jpeg	ari	ari	Benar	ari	Benar
630	ari_0_5116.jpeg	ari	ari	Benar	ari	Benar
631	ari_0_5146.jpeg	ari	ari	Benar	ari	Benar
632	ari_0_5166.jpeg	ari	ari	Benar	ari	Benar
633	ari_0_5229.jpeg	ari	ari	Benar	ari	Benar
634	ari_0_5235.jpeg	ari	ari	Benar	ari	Benar
635	ari_0_5269.jpeg	ari	ari	Benar	ari	Benar
636	ari_0_5320.jpeg	ari	ari	Benar	ari	Benar
637	ari_0_5412.jpeg	ari	ari	Benar	ari	Benar
638	ari_0_5455.jpeg	ari	ari	Benar	ari	Benar
639	ari_0_5503.jpeg	ari	ari	Benar	ari	Benar
640	ari_0_5515.jpeg	ari	ari	Benar	ari	Benar
641	ari_0_5521.jpeg	ari	ari	Benar	ari	Benar
642	ari_0_5540.jpeg	ari	ari	Benar	ari	Benar
643	ari_0_5623.jpeg	ari	ari	Benar	ari	Benar
644	ari_0_566.jpeg	ari	ari	Benar	ari	Benar
645	ari_0_5688.jpeg	ari	ari	Benar	ari	Benar
646	ari_0_572.jpeg	ari	ari	Benar	ari	Benar
647	ari_0_5747.jpeg	ari	ari	Benar	ari	Benar
648	ari_0_5925.jpeg	ari	ari	Benar	ari	Benar
649	ari_0_5999.jpeg	ari	ari	Benar	ari	Benar
650	ari_0_6027.jpeg	ari	ari	Benar	ari	Benar
651	ari_0_6053.jpeg	ari	ari	Benar	ari	Benar
652	ari_0_609.jpeg	ari	ari	Benar	ari	Benar
653	ari_0_6153.jpeg	ari	ari	Benar	ari	Benar
654	ari_0_6209.jpeg	ari	ari	Benar	ari	Benar
655	ari_0_6244.jpeg	ari	ari	Benar	ari	Benar
656	ari_0_6251.jpeg	ari	ari	Benar	ari	Benar
657	ari_0_6252.jpeg	ari	ari	Benar	ari	Benar
658	ari_0_6294.jpeg	ari	ari	Benar	ari	Benar
659	ari_0_6360.jpeg	ari	ari	Benar	ari	Benar
660	ari_0_6391.jpeg	ari	ari	Benar	ari	Benar
661	ari_0_6415.jpeg	ari	ari	Benar	ari	Benar
662	ari_0_6447.jpeg	ari	ari	Benar	ari	Benar
663	ari_0_6469.jpeg	ari	ari	Benar	ari	Benar
664	ari_0_6475.jpeg	ari	ari	Benar	ari	Benar
665	ari_0_648.jpeg	ari	ari	Benar	ari	Benar
666	ari_0_649.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
667	ari_0_6526.jpeg	ari	ari	Benar	ari	Benar
668	ari_0_6543.jpeg	ari	ari	Benar	ari	Benar
669	ari_0_6603.jpeg	ari	ari	Benar	ari	Benar
670	ari_0_6671.jpeg	ari	ari	Benar	ari	Benar
671	ari_0_6784.jpeg	ari	ari	Benar	ari	Benar
672	ari_0_6864.jpeg	ari	ari	Benar	ari	Benar
673	ari_0_6900.jpeg	ari	ari	Benar	ari	Benar
674	ari_0_6929.jpeg	ari	ari	Benar	ari	Benar
675	ari_0_6968.jpeg	ari	ari	Benar	ari	Benar
676	ari_0_6998.jpeg	ari	ari	Benar	ari	Benar
677	ari_0_7030.jpeg	ari	ari	Benar	ari	Benar
678	ari_0_7061.jpeg	ari	ari	Benar	ari	Benar
679	ari_0_7072.jpeg	ari	ari	Benar	ari	Benar
680	ari_0_7149.jpeg	ari	ari	Benar	ari	Benar
681	ari_0_7200.jpeg	ari	ari	Benar	ari	Benar
682	ari_0_7213.jpeg	ari	ari	Benar	ari	Benar
683	ari_0_7214.jpeg	ari	ari	Benar	ari	Benar
684	ari_0_7343.jpeg	ari	ari	Benar	ari	Benar
685	ari_0_7364.jpeg	ari	ari	Benar	ari	Benar
686	ari_0_7393.jpeg	ari	ari	Benar	ari	Benar
687	ari_0_7396.jpeg	ari	ari	Benar	ari	Benar
688	ari_0_7432.jpeg	ari	ari	Benar	ari	Benar
689	ari_0_7435.jpeg	ari	ari	Benar	ari	Benar
690	ari_0_7456.jpeg	ari	ari	Benar	ari	Benar
691	ari_0_7476.jpeg	ari	ari	Benar	ari	Benar
692	ari_0_7527.jpeg	ari	ari	Benar	ari	Benar
693	ari_0_7529.jpeg	ari	ari	Benar	ari	Benar
694	ari_0_7641.jpeg	ari	ari	Benar	ari	Benar
695	ari_0_7744.jpeg	ari	ari	Benar	ari	Benar
696	ari_0_7833.jpeg	ari	ari	Benar	ari	Benar
697	ari_0_7837.jpeg	ari	ari	Benar	ari	Benar
698	ari_0_7841.jpeg	ari	ari	Benar	ari	Benar
699	ari_0_7882.jpeg	ari	ari	Benar	ari	Benar
700	ari_0_7939.jpeg	ari	ari	Benar	ari	Benar
701	ari_0_7977.jpeg	ari	ari	Benar	ari	Benar
702	ari_0_8006.jpeg	ari	ari	Benar	ari	Benar
703	ari_0_804.jpeg	ari	ari	Benar	ari	Benar
704	ari_0_8061.jpeg	ari	ari	Benar	ari	Benar
705	ari_0_8075.jpeg	ari	ari	Benar	ari	Benar
706	ari_0_8077.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
707	ari_0_8108.jpeg	ari	ari	Benar	ari	Benar
708	ari_0_8197.jpeg	ari	ari	Benar	ari	Benar
709	ari_0_8254.jpeg	ari	ari	Benar	ari	Benar
710	ari_0_8276.jpeg	ari	ari	Benar	ari	Benar
711	ari_0_831.jpeg	ari	ari	Benar	ari	Benar
712	ari_0_8382.jpeg	ari	ari	Benar	ari	Benar
713	ari_0_8398.jpeg	ari	ari	Benar	ari	Benar
714	ari_0_8431.jpeg	ari	ari	Benar	ari	Benar
715	ari_0_8439.jpeg	ari	ari	Benar	ari	Benar
716	ari_0_8452.jpeg	ari	ari	Benar	ari	Benar
717	ari_0_8506.jpeg	ari	ari	Benar	ari	Benar
718	ari_0_8539.jpeg	ari	ari	Benar	ari	Benar
719	ari_0_8558.jpeg	ari	ari	Benar	ari	Benar
720	ari_0_861.jpeg	ari	ari	Benar	ari	Benar
721	ari_0_8626.jpeg	ari	ari	Benar	ari	Benar
722	ari_0_8675.jpeg	ari	ari	Benar	ari	Benar
723	ari_0_8713.jpeg	ari	ari	Benar	ari	Benar
724	ari_0_8794.jpeg	ari	ari	Benar	ari	Benar
725	ari_0_8810.jpeg	ari	ari	Benar	ari	Benar
726	ari_0_8834.jpeg	ari	ari	Benar	ari	Benar
727	ari_0_8835.jpeg	ari	ari	Benar	ari	Benar
728	ari_0_8911.jpeg	ari	ari	Benar	ari	Benar
729	ari_0_8922.jpeg	ari	ari	Benar	ari	Benar
730	ari_0_8970.jpeg	ari	ari	Benar	ari	Benar
731	ari_0_898.jpeg	ari	ari	Benar	ari	Benar
732	ari_0_9033.jpeg	ari	ari	Benar	ari	Benar
733	ari_0_914.jpeg	ari	ari	Benar	ari	Benar
734	ari_0_9186.jpeg	ari	ari	Benar	ari	Benar
735	ari_0_9191.jpeg	ari	ari	Benar	ari	Benar
736	ari_0_9224.jpeg	ari	ari	Benar	ari	Benar
737	ari_0_9317.jpeg	ari	ari	Benar	ari	Benar
738	ari_0_9330.jpeg	ari	ari	Benar	ari	Benar
739	ari_0_9368.jpeg	ari	ari	Benar	ari	Benar
740	ari_0_9396.jpeg	ari	ari	Benar	ari	Benar
741	ari_0_944.jpeg	ari	ari	Benar	ari	Benar
742	ari_0_9467.jpeg	ari	ari	Benar	ari	Benar
743	ari_0_9513.jpeg	ari	ari	Benar	ari	Benar
744	ari_0_9527.jpeg	ari	ari	Benar	ari	Benar
745	ari_0_9541.jpeg	ari	ari	Benar	ari	Benar
746	ari_0_9551.jpeg	ari	ari	Benar	ari	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
747	ari_0_9626.jpeg	ari	ari	Benar	ari	Benar
748	ari_0_9722.jpeg	ari	ari	Benar	ari	Benar
749	ari_0_9743.jpeg	ari	ari	Benar	ari	Benar
750	ari_0_9835.jpeg	ari	ari	Benar	ari	Benar
751	ari_0_9871.jpeg	ari	ari	Benar	ari	Benar
752	ari_0_9877.jpeg	ari	ari	Benar	ari	Benar
753	ari_0_9890.jpeg	ari	ari	Benar	ari	Benar
754	ari_0_990.jpeg	ari	ari	Benar	ari	Benar
755	ari_0_9952.jpeg	ari	ari	Benar	ari	Benar
756	ari_0_9978.jpeg	ari	ari	Benar	ari	Benar
757	ari_0_9988.jpeg	ari	ari	Benar	ari	Benar
758	dinda_0_1003.jpeg	dinda	dinda	Benar	dinda	Benar
759	dinda_0_1049.jpeg	dinda	dinda	Benar	dinda	Benar
760	dinda_0_1091.jpeg	dinda	dinda	Benar	dinda	Benar
761	dinda_0_1182.jpeg	dinda	dinda	Benar	dinda	Benar
762	dinda_0_1193.jpeg	dinda	dinda	Benar	dinda	Benar
763	dinda_0_1212.jpeg	dinda	dinda	Benar	dinda	Benar
764	dinda_0_1248.jpeg	dinda	dinda	Benar	dinda	Benar
765	dinda_0_1336.jpeg	dinda	dinda	Benar	dinda	Benar
766	dinda_0_1341.jpeg	dinda	dinda	Benar	dinda	Benar
767	dinda_0_135.jpeg	dinda	dinda	Benar	dinda	Benar
768	dinda_0_1371.jpeg	dinda	dinda	Benar	dinda	Benar
769	dinda_0_1403.jpeg	dinda	dinda	Benar	dinda	Benar
770	dinda_0_1406.jpeg	dinda	dinda	Benar	dinda	Benar
771	dinda_0_142.jpeg	dinda	dinda	Benar	dinda	Benar
772	dinda_0_1493.jpeg	dinda	dinda	Benar	dinda	Benar
773	dinda_0_1501.jpeg	dinda	dinda	Benar	dinda	Benar
774	dinda_0_1648.jpeg	dinda	dinda	Benar	dinda	Benar
775	dinda_0_1649.jpeg	dinda	dinda	Benar	dinda	Benar
776	dinda_0_1693.jpeg	dinda	dinda	Benar	dinda	Benar
777	dinda_0_1698.jpeg	dinda	dinda	Benar	dinda	Benar
778	dinda_0_1744.jpeg	dinda	dinda	Benar	dinda	Benar
779	dinda_0_1772.jpeg	dinda	dinda	Benar	dinda	Benar
780	dinda_0_1790.jpeg	dinda	dinda	Benar	dinda	Benar
781	dinda_0_187.jpeg	dinda	dinda	Benar	dinda	Benar
782	dinda_0_1875.jpeg	dinda	dinda	Benar	dinda	Benar
783	dinda_0_1881.jpeg	dinda	dinda	Benar	dinda	Benar
784	dinda_0_1909.jpeg	dinda	dinda	Benar	dinda	Benar
785	dinda_0_1925.jpeg	dinda	dinda	Benar	dinda	Benar
786	dinda_0_195.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
787	dinda_0_1970.jpeg	dinda	dinda	Benar	dinda	Benar
788	dinda_0_2044.jpeg	dinda	dinda	Benar	dinda	Benar
789	dinda_0_2117.jpeg	dinda	dinda	Benar	dinda	Benar
790	dinda_0_2155.jpeg	dinda	dinda	Benar	dinda	Benar
791	dinda_0_2175.jpeg	dinda	dinda	Benar	dinda	Benar
792	dinda_0_2185.jpeg	dinda	dinda	Benar	dinda	Benar
793	dinda_0_2298.jpeg	dinda	dinda	Benar	dinda	Benar
794	dinda_0_2303.jpeg	dinda	dinda	Benar	dinda	Benar
795	dinda_0_2368.jpeg	dinda	dinda	Benar	dinda	Benar
796	dinda_0_2489.jpeg	dinda	dinda	Benar	dinda	Benar
797	dinda_0_2533.jpeg	dinda	dinda	Benar	dinda	Benar
798	dinda_0_2544.jpeg	dinda	dinda	Benar	dinda	Benar
799	dinda_0_2548.jpeg	dinda	dinda	Benar	dinda	Benar
800	dinda_0_2582.jpeg	dinda	dinda	Benar	dinda	Benar
801	dinda_0_259.jpeg	dinda	dinda	Benar	dinda	Benar
802	dinda_0_2601.jpeg	dinda	dinda	Benar	dinda	Benar
803	dinda_0_2615.jpeg	dinda	dinda	Benar	dinda	Benar
804	dinda_0_271.jpeg	dinda	dinda	Benar	dinda	Benar
805	dinda_0_2732.jpeg	dinda	dinda	Benar	dinda	Benar
806	dinda_0_2943.jpeg	dinda	dinda	Benar	dinda	Benar
807	dinda_0_2948.jpeg	dinda	dinda	Benar	dinda	Benar
808	dinda_0_2976.jpeg	dinda	dinda	Benar	dinda	Benar
809	dinda_0_2992.jpeg	dinda	dinda	Benar	dinda	Benar
810	dinda_0_3048.jpeg	dinda	dinda	Benar	dinda	Benar
811	dinda_0_3064.jpeg	dinda	dinda	Benar	dinda	Benar
812	dinda_0_3083.jpeg	dinda	dinda	Benar	dinda	Benar
813	dinda_0_3144.jpeg	dinda	dinda	Benar	dinda	Benar
814	dinda_0_3181.jpeg	dinda	dinda	Benar	dinda	Benar
815	dinda_0_3257.jpeg	dinda	dinda	Benar	dinda	Benar
816	dinda_0_3273.jpeg	dinda	dinda	Benar	dinda	Benar
817	dinda_0_3274.jpeg	dinda	dinda	Benar	dinda	Benar
818	dinda_0_3288.jpeg	dinda	dinda	Benar	dinda	Benar
819	dinda_0_3341.jpeg	dinda	dinda	Benar	dinda	Benar
820	dinda_0_3514.jpeg	dinda	dinda	Benar	dinda	Benar
821	dinda_0_3516.jpeg	dinda	dinda	Benar	dinda	Benar
822	dinda_0_1647.jpeg	dinda	dinda	Benar	dinda	Benar
823	dinda_0_2144.jpeg	dinda	dinda	Benar	dinda	Benar
824	dinda_0_2930.jpeg	dinda	dinda	Benar	dinda	Benar
825	dinda_0_3630.jpeg	dinda	dinda	Benar	dinda	Benar
826	dinda_0_3943.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
827	dinda_0_4487.jpeg	dinda	dinda	Benar	dinda	Benar
828	dinda_0_5124.jpeg	dinda	dinda	Benar	dinda	Benar
829	dinda_0_5673.jpeg	dinda	dinda	Benar	dinda	Benar
830	dinda_0_6289.jpeg	dinda	dinda	Benar	dinda	Benar
831	dinda_0_6860.jpeg	dinda	dinda	Benar	dinda	Benar
832	dinda_0_7585.jpeg	dinda	dinda	Benar	dinda	Benar
833	dinda_0_8215.jpeg	dinda	dinda	Benar	dinda	Benar
834	dinda_0_8866.jpeg	dinda	dinda	Benar	dinda	Benar
835	dinda_0_3654.jpeg	dinda	dinda	Benar	dinda	Benar
836	dinda_0_3681.jpeg	dinda	dinda	Benar	dinda	Benar
837	dinda_0_3718.jpeg	dinda	dinda	Benar	dinda	Benar
838	dinda_0_3732.jpeg	dinda	dinda	Benar	dinda	Benar
839	dinda_0_3738.jpeg	dinda	dinda	Benar	dinda	Benar
840	dinda_0_3799.jpeg	dinda	dinda	Benar	dinda	Benar
841	dinda_0_3844.jpeg	dinda	dinda	Benar	dinda	Benar
842	dinda_0_3851.jpeg	dinda	dinda	Benar	dinda	Benar
843	dinda_0_3871.jpeg	dinda	dinda	Benar	dinda	Benar
844	dinda_0_3882.jpeg	dinda	dinda	Benar	dinda	Benar
845	dinda_0_3893.jpeg	dinda	dinda	Benar	dinda	Benar
846	dinda_0_3895.jpeg	dinda	dinda	Benar	dinda	Benar
847	dinda_0_3901.jpeg	dinda	dinda	Benar	dinda	Benar
848	dinda_0_3916.jpeg	dinda	dinda	Benar	dinda	Benar
849	dinda_0_3919.jpeg	dinda	dinda	Benar	dinda	Benar
850	dinda_0_3935.jpeg	dinda	dinda	Benar	dinda	Benar
851	dinda_0_3966.jpeg	dinda	dinda	Benar	dinda	Benar
852	dinda_0_4027.jpeg	dinda	dinda	Benar	dinda	Benar
853	dinda_0_4030.jpeg	dinda	dinda	Benar	dinda	Benar
854	dinda_0_4034.jpeg	dinda	dinda	Benar	dinda	Benar
855	dinda_0_4040.jpeg	dinda	dinda	Benar	dinda	Benar
856	dinda_0_4083.jpeg	dinda	dinda	Benar	dinda	Benar
857	dinda_0_4118.jpeg	dinda	dinda	Benar	dinda	Benar
858	dinda_0_4139.jpeg	dinda	dinda	Benar	dinda	Benar
859	dinda_0_4191.jpeg	dinda	dinda	Benar	dinda	Benar
860	dinda_0_4200.jpeg	dinda	dinda	Benar	dinda	Benar
861	dinda_0_421.jpeg	dinda	dinda	Benar	dinda	Benar
862	dinda_0_4241.jpeg	dinda	dinda	Benar	dinda	Benar
863	dinda_0_4357.jpeg	dinda	dinda	Benar	dinda	Benar
864	dinda_0_4385.jpeg	dinda	dinda	Benar	dinda	Benar
865	dinda_0_4390.jpeg	dinda	dinda	Benar	dinda	Benar
866	dinda_0_4464.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
867	dinda_0_4529.jpeg	dinda	dinda	Benar	dinda	Benar
868	dinda_0_4562.jpeg	dinda	dinda	Benar	dinda	Benar
869	dinda_0_4586.jpeg	dinda	dinda	Benar	dinda	Benar
870	dinda_0_465.jpeg	dinda	dinda	Benar	dinda	Benar
871	dinda_0_4655.jpeg	dinda	dinda	Benar	dinda	Benar
872	dinda_0_466.jpeg	dinda	dinda	Benar	dinda	Benar
873	dinda_0_4794.jpeg	dinda	dinda	Benar	dinda	Benar
874	dinda_0_4895.jpeg	dinda	dinda	Benar	dinda	Benar
875	dinda_0_4896.jpeg	dinda	dinda	Benar	dinda	Benar
876	dinda_0_4905.jpeg	dinda	dinda	Benar	dinda	Benar
877	dinda_0_4960.jpeg	dinda	dinda	Benar	dinda	Benar
878	dinda_0_500.jpeg	dinda	dinda	Benar	dinda	Benar
879	dinda_0_5009.jpeg	dinda	dinda	Benar	dinda	Benar
880	dinda_0_5038.jpeg	dinda	dinda	Benar	dinda	Benar
881	dinda_0_5074.jpeg	dinda	dinda	Benar	dinda	Benar
882	dinda_0_509.jpeg	dinda	dinda	Benar	dinda	Benar
883	dinda_0_5262.jpeg	dinda	dinda	Benar	dinda	Benar
884	dinda_0_5275.jpeg	dinda	dinda	Benar	dinda	Benar
885	dinda_0_5325.jpeg	dinda	dinda	Benar	dinda	Benar
886	dinda_0_5349.jpeg	dinda	dinda	Benar	dinda	Benar
887	dinda_0_536.jpeg	dinda	dinda	Benar	dinda	Benar
888	dinda_0_5364.jpeg	dinda	dinda	Benar	dinda	Benar
889	dinda_0_5366.jpeg	dinda	dinda	Benar	dinda	Benar
890	dinda_0_54.jpeg	dinda	dinda	Benar	dinda	Benar
891	dinda_0_5447.jpeg	dinda	dinda	Benar	dinda	Benar
892	dinda_0_5460.jpeg	dinda	dinda	Benar	dinda	Benar
893	dinda_0_5503.jpeg	dinda	dinda	Benar	dinda	Benar
894	dinda_0_5522.jpeg	dinda	dinda	Benar	dinda	Benar
895	dinda_0_553.jpeg	dinda	dinda	Benar	dinda	Benar
896	dinda_0_5594.jpeg	dinda	dinda	Benar	dinda	Benar
897	dinda_0_5609.jpeg	dinda	dinda	Benar	dinda	Benar
898	dinda_0_5664.jpeg	dinda	dinda	Benar	dinda	Benar
899	dinda_0_5713.jpeg	dinda	dinda	Benar	dinda	Benar
900	dinda_0_5758.jpeg	dinda	dinda	Benar	dinda	Benar
901	dinda_0_5814.jpeg	dinda	dinda	Benar	dinda	Benar
902	dinda_0_5863.jpeg	dinda	dinda	Benar	dinda	Benar
903	dinda_0_5910.jpeg	dinda	dinda	Benar	dinda	Benar
904	dinda_0_5917.jpeg	dinda	dinda	Benar	dinda	Benar
905	dinda_0_5943.jpeg	dinda	dinda	Benar	dinda	Benar
906	dinda_0_6066.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
907	dinda_0_6154.jpeg	dinda	dinda	Benar	dinda	Benar
908	dinda_0_6169.jpeg	dinda	dinda	Benar	dinda	Benar
909	dinda_0_6185.jpeg	dinda	dinda	Benar	dinda	Benar
910	dinda_0_6186.jpeg	dinda	dinda	Benar	dinda	Benar
911	dinda_0_6221.jpeg	dinda	dinda	Benar	dinda	Benar
912	dinda_0_6223.jpeg	dinda	dinda	Benar	dinda	Benar
913	dinda_0_6231.jpeg	dinda	dinda	Benar	dinda	Benar
914	dinda_0_6235.jpeg	dinda	dinda	Benar	dinda	Benar
915	dinda_0_6307.jpeg	dinda	dinda	Benar	dinda	Benar
916	dinda_0_6316.jpeg	dinda	dinda	Benar	dinda	Benar
917	dinda_0_6317.jpeg	dinda	dinda	Benar	dinda	Benar
918	dinda_0_6320.jpeg	dinda	dinda	Benar	dinda	Benar
919	dinda_0_6353.jpeg	dinda	dinda	Benar	dinda	Benar
920	dinda_0_6395.jpeg	dinda	dinda	Benar	dinda	Benar
921	dinda_0_640.jpeg	dinda	dinda	Benar	dinda	Benar
922	dinda_0_647.jpeg	dinda	dinda	Benar	dinda	Benar
923	dinda_0_6488.jpeg	dinda	dinda	Benar	dinda	Benar
924	dinda_0_6548.jpeg	dinda	dinda	Benar	dinda	Benar
925	dinda_0_6730.jpeg	dinda	dinda	Benar	dinda	Benar
926	dinda_0_6776.jpeg	dinda	dinda	Benar	dinda	Benar
927	dinda_0_68.jpeg	dinda	dinda	Benar	dinda	Benar
928	dinda_0_6811.jpeg	dinda	dinda	Benar	dinda	Benar
929	dinda_0_6838.jpeg	dinda	dinda	Benar	dinda	Benar
930	dinda_0_6851.jpeg	dinda	dinda	Benar	dinda	Benar
931	dinda_0_6928.jpeg	dinda	dinda	Benar	dinda	Benar
932	dinda_0_697.jpeg	dinda	dinda	Benar	dinda	Benar
933	dinda_0_7076.jpeg	dinda	dinda	Benar	dinda	Benar
934	dinda_0_720.jpeg	dinda	dinda	Benar	dinda	Benar
935	dinda_0_7202.jpeg	dinda	dinda	Benar	dinda	Benar
936	dinda_0_7253.jpeg	dinda	dinda	Benar	dinda	Benar
937	dinda_0_7316.jpeg	dinda	dinda	Benar	dinda	Benar
938	dinda_0_7423.jpeg	dinda	dinda	Benar	dinda	Benar
939	dinda_0_7441.jpeg	dinda	dinda	Benar	dinda	Benar
940	dinda_0_7443.jpeg	dinda	dinda	Benar	dinda	Benar
941	dinda_0_747.jpeg	dinda	dinda	Benar	dinda	Benar
942	dinda_0_7491.jpeg	dinda	dinda	Benar	dinda	Benar
943	dinda_0_7521.jpeg	dinda	dinda	Benar	dinda	Benar
944	dinda_0_7553.jpeg	dinda	dinda	Benar	dinda	Benar
945	dinda_0_7559.jpeg	dinda	dinda	Benar	dinda	Benar
946	dinda_0_7575.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
947	dinda_0_7586.jpeg	dinda	dinda	Benar	dinda	Benar
948	dinda_0_7595.jpeg	dinda	dinda	Benar	dinda	Benar
949	dinda_0_7634.jpeg	dinda	dinda	Benar	dinda	Benar
950	dinda_0_7672.jpeg	dinda	dinda	Benar	dinda	Benar
951	dinda_0_7713.jpeg	dinda	dinda	Benar	dinda	Benar
952	dinda_0_7760.jpeg	dinda	dinda	Benar	dinda	Benar
953	dinda_0_7787.jpeg	dinda	dinda	Benar	dinda	Benar
954	dinda_0_7843.jpeg	dinda	dinda	Benar	dinda	Benar
955	dinda_0_7854.jpeg	dinda	dinda	Benar	dinda	Benar
956	dinda_0_7858.jpeg	dinda	dinda	Benar	dinda	Benar
957	dinda_0_7923.jpeg	dinda	dinda	Benar	dinda	Benar
958	dinda_0_797.jpeg	dinda	dinda	Benar	dinda	Benar
959	dinda_0_8012.jpeg	dinda	dinda	Benar	dinda	Benar
960	dinda_0_8143.jpeg	dinda	dinda	Benar	dinda	Benar
961	dinda_0_817.jpeg	dinda	dinda	Benar	dinda	Benar
962	dinda_0_8201.jpeg	dinda	dinda	Benar	dinda	Benar
963	dinda_0_829.jpeg	dinda	dinda	Benar	dinda	Benar
964	dinda_0_8291.jpeg	dinda	dinda	Benar	dinda	Benar
965	dinda_0_83.jpeg	dinda	dinda	Benar	dinda	Benar
966	dinda_0_8351.jpeg	dinda	dinda	Benar	dinda	Benar
967	dinda_0_8363.jpeg	dinda	dinda	Benar	dinda	Benar
968	dinda_0_8364.jpeg	dinda	dinda	Benar	dinda	Benar
969	dinda_0_8388.jpeg	dinda	dinda	Benar	dinda	Benar
970	dinda_0_8435.jpeg	dinda	dinda	Benar	dinda	Benar
971	dinda_0_8469.jpeg	dinda	dinda	Benar	dinda	Benar
972	dinda_0_8522.jpeg	dinda	dinda	Benar	dinda	Benar
973	dinda_0_8527.jpeg	dinda	dinda	Benar	dinda	Benar
974	dinda_0_8548.jpeg	dinda	dinda	Benar	dinda	Benar
975	dinda_0_8582.jpeg	dinda	dinda	Benar	dinda	Benar
976	dinda_0_860.jpeg	dinda	dinda	Benar	dinda	Benar
977	dinda_0_8814.jpeg	dinda	dinda	Benar	dinda	Benar
978	dinda_0_8846.jpeg	dinda	dinda	Benar	dinda	Benar
979	dinda_0_8870.jpeg	dinda	dinda	Benar	dinda	Benar
980	dinda_0_8944.jpeg	dinda	dinda	Benar	dinda	Benar
981	dinda_0_8979.jpeg	dinda	dinda	Benar	dinda	Benar
982	dinda_0_899.jpeg	dinda	dinda	Benar	dinda	Benar
983	dinda_0_9127.jpeg	dinda	dinda	Benar	dinda	Benar
984	dinda_0_9146.jpeg	dinda	dinda	Benar	dinda	Benar
985	dinda_0_9157.jpeg	dinda	dinda	Benar	dinda	Benar
986	dinda_0_9198.jpeg	dinda	dinda	Benar	dinda	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
987	dinda_0_9231.jpeg	dinda	dinda	Benar	dinda	Benar
988	dinda_0_926.jpeg	dinda	dinda	Benar	dinda	Benar
989	dinda_0_9398.jpeg	dinda	dinda	Benar	dinda	Benar
990	dinda_0_9468.jpeg	dinda	dinda	Benar	dinda	Benar
991	dinda_0_9486.jpeg	dinda	dinda	Benar	dinda	Benar
992	dinda_0_9487.jpeg	dinda	dinda	Benar	dinda	Benar
993	dinda_0_9508.jpeg	dinda	dinda	Benar	dinda	Benar
994	dinda_0_9562.jpeg	dinda	dinda	Benar	dinda	Benar
995	dinda_0_960.jpeg	dinda	dinda	Benar	dinda	Benar
996	dinda_0_9649.jpeg	dinda	dinda	Benar	dinda	Benar
997	dinda_0_9658.jpeg	dinda	dinda	Benar	dinda	Benar
998	dinda_0_9720.jpeg	dinda	dinda	Benar	dinda	Benar
999	dinda_0_9728.jpeg	dinda	dinda	Benar	dinda	Benar
1000	dinda_0_9759.jpeg	dinda	dinda	Benar	dinda	Benar
1001	dinda_0_9771.jpeg	dinda	dinda	Benar	dinda	Benar
1002	dinda_0_9834.jpeg	dinda	dinda	Benar	dinda	Benar
1003	dinda_0_984.jpeg	dinda	dinda	Benar	dinda	Benar
1004	dinda_0_9854.jpeg	dinda	dinda	Benar	dinda	Benar
1005	dinda_0_9900.jpeg	dinda	dinda	Benar	dinda	Benar
1006	dinda_0_9984.jpeg	dinda	dinda	Benar	dinda	Benar
1007	dinda_0_9995.jpeg	dinda	dinda	Benar	dinda	Benar
1008	dwita_0_1004.jpeg	dwita	dwita	Benar	dwita	Benar
1009	dwita_0_1048.jpeg	dwita	dwita	Benar	dwita	Benar
1010	dwita_0_1073.jpeg	dwita	dwita	Benar	dwita	Benar
1011	dwita_0_1082.jpeg	dwita	dwita	Benar	dwita	Benar
1012	dwita_0_1114.jpeg	dwita	dwita	Benar	dwita	Benar
1013	dwita_0_114.jpeg	dwita	dwita	Benar	dwita	Benar
1014	dwita_0_1166.jpeg	dwita	dwita	Benar	dwita	Benar
1015	dwita_0_1174.jpeg	dwita	dwita	Benar	dwita	Benar
1016	dwita_0_1231.jpeg	dwita	dwita	Benar	dwita	Benar
1017	dwita_0_1242.jpeg	dwita	dwita	Benar	dwita	Benar
1018	dwita_0_1267.jpeg	dwita	dwita	Benar	dwita	Benar
1019	dwita_0_1268.jpeg	dwita	dwita	Benar	dwita	Benar
1020	dwita_0_1278.jpeg	dwita	dwita	Benar	dwita	Benar
1021	dwita_0_13.jpeg	dwita	dwita	Benar	dwita	Benar
1022	dwita_0_1314.jpeg	dwita	dwita	Benar	dwita	Benar
1023	dwita_0_1398.jpeg	dwita	dwita	Benar	dwita	Benar
1024	dwita_0_1415.jpeg	dwita	dwita	Benar	dwita	Benar
1025	dwita_0_1454.jpeg	dwita	dwita	Benar	dwita	Benar
1026	dwita_0_1474.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1027	dwita_0_1486.jpeg	dwita	dwita	Benar	dwita	Benar
1028	dwita_0_1506.jpeg	dwita	dwita	Benar	dwita	Benar
1029	dwita_0_1532.jpeg	dwita	dwita	Benar	dwita	Benar
1030	dwita_0_1561.jpeg	dwita	dwita	Benar	dwita	Benar
1031	dwita_0_1577.jpeg	dwita	dwita	Benar	dwita	Benar
1032	dwita_0_1623.jpeg	dwita	dwita	Benar	dwita	Benar
1033	dwita_0_1626.jpeg	dwita	dwita	Benar	dwita	Benar
1034	dwita_0_1632.jpeg	dwita	dwita	Benar	dwita	Benar
1035	dwita_0_1722.jpeg	dwita	dwita	Benar	dwita	Benar
1036	dwita_0_1730.jpeg	dwita	dwita	Benar	dwita	Benar
1037	dwita_0_1745.jpeg	dwita	dwita	Benar	dwita	Benar
1038	dwita_0_1747.jpeg	dwita	dwita	Benar	dwita	Benar
1039	dwita_0_1750.jpeg	dwita	dwita	Benar	dwita	Benar
1040	dwita_0_1802.jpeg	dwita	dwita	Benar	dwita	Benar
1041	dwita_0_1804.jpeg	dwita	dwita	Benar	dwita	Benar
1042	dwita_0_1836.jpeg	dwita	dwita	Benar	dwita	Benar
1043	dwita_0_190.jpeg	dwita	dwita	Benar	dwita	Benar
1044	dwita_0_1904.jpeg	dwita	dwita	Benar	dwita	Benar
1045	dwita_0_1911.jpeg	dwita	dwita	Benar	dwita	Benar
1046	dwita_0_2005.jpeg	dwita	dwita	Benar	dwita	Benar
1047	dwita_0_2006.jpeg	dwita	dwita	Benar	dwita	Benar
1048	dwita_0_2046.jpeg	dwita	dwita	Benar	dwita	Benar
1049	dwita_0_2124.jpeg	dwita	dwita	Benar	dwita	Benar
1050	dwita_0_2186.jpeg	dwita	dwita	Benar	dwita	Benar
1051	dwita_0_2222.jpeg	dwita	dwita	Benar	dwita	Benar
1052	dwita_0_227.jpeg	dwita	dwita	Benar	dwita	Benar
1053	dwita_0_2278.jpeg	dwita	dwita	Benar	dwita	Benar
1054	dwita_0_2287.jpeg	dwita	dwita	Benar	dwita	Benar
1055	dwita_0_2303.jpeg	dwita	dwita	Benar	dwita	Benar
1056	dwita_0_2365.jpeg	dwita	dwita	Benar	dwita	Benar
1057	dwita_0_2484.jpeg	dwita	dwita	Benar	dwita	Benar
1058	dwita_0_2488.jpeg	dwita	dwita	Benar	dwita	Benar
1059	dwita_0_2565.jpeg	dwita	dwita	Benar	dwita	Benar
1060	dwita_0_2634.jpeg	dwita	dwita	Benar	dwita	Benar
1061	dwita_0_269.jpeg	dwita	dwita	Benar	dwita	Benar
1062	dwita_0_2698.jpeg	dwita	dwita	Benar	dwita	Benar
1063	dwita_0_2706.jpeg	dwita	dwita	Benar	dwita	Benar
1064	dwita_0_2709.jpeg	dwita	dwita	Benar	dwita	Benar
1065	dwita_0_2730.jpeg	dwita	dwita	Benar	dwita	Benar
1066	dwita_0_274.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1067	dwita_0_2761.jpeg	dwita	dwita	Benar	dwita	Benar
1068	dwita_0_2775.jpeg	dwita	dwita	Benar	dwita	Benar
1069	dwita_0_2835.jpeg	dwita	dwita	Benar	dwita	Benar
1070	dwita_0_2912.jpeg	dwita	dwita	Benar	dwita	Benar
1071	dwita_0_2921.jpeg	dwita	dwita	Benar	dwita	Benar
1072	dwita_0_1410.jpeg	dwita	dwita	Benar	dwita	Benar
1073	dwita_0_1789.jpeg	dwita	dwita	Benar	dwita	Benar
1074	dwita_0_2359.jpeg	dwita	dwita	Benar	dwita	Benar
1075	dwita_0_2949.jpeg	dwita	dwita	Benar	dwita	Benar
1076	dwita_0_3417.jpeg	dwita	dwita	Benar	dwita	Benar
1077	dwita_0_4159.jpeg	dwita	dwita	Benar	dwita	Benar
1078	dwita_0_4872.jpeg	dwita	dwita	Benar	dwita	Benar
1079	dwita_0_5499.jpeg	dwita	dwita	Benar	dwita	Benar
1080	dwita_0_6142.jpeg	dwita	dwita	Benar	dwita	Benar
1081	dwita_0_6641.jpeg	dwita	dwita	Benar	dwita	Benar
1082	dwita_0_7132.jpeg	dwita	dwita	Benar	dwita	Benar
1083	dwita_0_7707.jpeg	dwita	dwita	Benar	dwita	Benar
1084	dwita_0_8731.jpeg	dwita	dwita	Benar	dwita	Benar
1085	dwita_0_2992.jpeg	dwita	dwita	Benar	dwita	Benar
1086	dwita_0_3013.jpeg	dwita	dwita	Benar	dwita	Benar
1087	dwita_0_3054.jpeg	dwita	dwita	Benar	dwita	Benar
1088	dwita_0_3101.jpeg	dwita	dwita	Benar	dwita	Benar
1089	dwita_0_3140.jpeg	dwita	dwita	Benar	dwita	Benar
1090	dwita_0_3159.jpeg	dwita	dwita	Benar	dwita	Benar
1091	dwita_0_3198.jpeg	dwita	dwita	Benar	dwita	Benar
1092	dwita_0_3204.jpeg	dwita	dwita	Benar	dwita	Benar
1093	dwita_0_3248.jpeg	dwita	dwita	Benar	dwita	Benar
1094	dwita_0_3270.jpeg	dwita	dwita	Benar	dwita	Benar
1095	dwita_0_3293.jpeg	dwita	dwita	Benar	dwita	Benar
1096	dwita_0_3295.jpeg	dwita	dwita	Benar	dwita	Benar
1097	dwita_0_3313.jpeg	dwita	dwita	Benar	dwita	Benar
1098	dwita_0_3347.jpeg	dwita	dwita	Benar	dwita	Benar
1099	dwita_0_337.jpeg	dwita	dwita	Benar	dwita	Benar
1100	dwita_0_338.jpeg	dwita	dwita	Benar	dwita	Benar
1101	dwita_0_3452.jpeg	dwita	dwita	Benar	dwita	Benar
1102	dwita_0_3459.jpeg	dwita	dwita	Benar	dwita	Benar
1103	dwita_0_3494.jpeg	dwita	dwita	Benar	dwita	Benar
1104	dwita_0_3563.jpeg	dwita	dwita	Benar	dwita	Benar
1105	dwita_0_3579.jpeg	dwita	dwita	Benar	dwita	Benar
1106	dwita_0_360.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1107	dwita_0_3621.jpeg	dwita	dwita	Benar	dwita	Benar
1108	dwita_0_3695.jpeg	dwita	dwita	Benar	dwita	Benar
1109	dwita_0_3707.jpeg	dwita	dwita	Benar	dwita	Benar
1110	dwita_0_3746.jpeg	dwita	dwita	Benar	dwita	Benar
1111	dwita_0_3754.jpeg	dwita	dwita	Benar	dwita	Benar
1112	dwita_0_3798.jpeg	dwita	dwita	Benar	dwita	Benar
1113	dwita_0_3872.jpeg	dwita	dwita	Benar	dwita	Benar
1114	dwita_0_3969.jpeg	dwita	dwita	Benar	dwita	Benar
1115	dwita_0_4027.jpeg	dwita	dwita	Benar	dwita	Benar
1116	dwita_0_4076.jpeg	dwita	dwita	Benar	dwita	Benar
1117	dwita_0_4207.jpeg	dwita	dwita	Benar	dwita	Benar
1118	dwita_0_4250.jpeg	dwita	dwita	Benar	dwita	Benar
1119	dwita_0_4329.jpeg	dwita	dwita	Benar	dwita	Benar
1120	dwita_0_4340.jpeg	dwita	dwita	Benar	dwita	Benar
1121	dwita_0_439.jpeg	dwita	dwita	Benar	dwita	Benar
1122	dwita_0_4410.jpeg	dwita	dwita	Benar	dwita	Benar
1123	dwita_0_4426.jpeg	dwita	dwita	Benar	dwita	Benar
1124	dwita_0_4450.jpeg	dwita	dwita	Benar	dwita	Benar
1125	dwita_0_4459.jpeg	dwita	dwita	Benar	dwita	Benar
1126	dwita_0_4463.jpeg	dwita	dwita	Benar	dwita	Benar
1127	dwita_0_4562.jpeg	dwita	dwita	Benar	dwita	Benar
1128	dwita_0_4585.jpeg	dwita	dwita	Benar	dwita	Benar
1129	dwita_0_4608.jpeg	dwita	dwita	Benar	dwita	Benar
1130	dwita_0_464.jpeg	dwita	dwita	Benar	dwita	Benar
1131	dwita_0_482.jpeg	dwita	dwita	Benar	dwita	Benar
1132	dwita_0_4849.jpeg	dwita	dwita	Benar	dwita	Benar
1133	dwita_0_4993.jpeg	dwita	dwita	Benar	dwita	Benar
1134	dwita_0_505.jpeg	dwita	dwita	Benar	dwita	Benar
1135	dwita_0_5097.jpeg	dwita	dwita	Benar	dwita	Benar
1136	dwita_0_51.jpeg	dwita	dwita	Benar	dwita	Benar
1137	dwita_0_5265.jpeg	dwita	dwita	Benar	dwita	Benar
1138	dwita_0_5283.jpeg	dwita	dwita	Benar	dwita	Benar
1139	dwita_0_5294.jpeg	dwita	dwita	Benar	dwita	Benar
1140	dwita_0_5333.jpeg	dwita	dwita	Benar	dwita	Benar
1141	dwita_0_536.jpeg	dwita	dwita	Benar	dwita	Benar
1142	dwita_0_5361.jpeg	dwita	dwita	Benar	dwita	Benar
1143	dwita_0_5451.jpeg	dwita	dwita	Benar	dwita	Benar
1144	dwita_0_546.jpeg	dwita	dwita	Benar	dwita	Benar
1145	dwita_0_5465.jpeg	dwita	dwita	Benar	dwita	Benar
1146	dwita_0_547.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1147	dwita_0_5470.jpeg	dwita	dwita	Benar	dwita	Benar
1148	dwita_0_5498.jpeg	dwita	dwita	Benar	dwita	Benar
1149	dwita_0_5592.jpeg	dwita	dwita	Benar	dwita	Benar
1150	dwita_0_5599.jpeg	dwita	dwita	Benar	dwita	Benar
1151	dwita_0_5608.jpeg	dwita	dwita	Benar	dwita	Benar
1152	dwita_0_567.jpeg	dwita	dwita	Benar	dwita	Benar
1153	dwita_0_5677.jpeg	dwita	dwita	Benar	dwita	Benar
1154	dwita_0_5680.jpeg	dwita	dwita	Benar	dwita	Benar
1155	dwita_0_5742.jpeg	dwita	dwita	Benar	dwita	Benar
1156	dwita_0_5827.jpeg	dwita	dwita	Benar	dwita	Benar
1157	dwita_0_5850.jpeg	dwita	dwita	Benar	dwita	Benar
1158	dwita_0_587.jpeg	dwita	dwita	Benar	dwita	Benar
1159	dwita_0_5871.jpeg	dwita	dwita	Benar	dwita	Benar
1160	dwita_0_5929.jpeg	dwita	dwita	Benar	dwita	Benar
1161	dwita_0_5955.jpeg	dwita	dwita	Benar	dwita	Benar
1162	dwita_0_5999.jpeg	dwita	dwita	Benar	dwita	Benar
1163	dwita_0_6027.jpeg	dwita	dwita	Benar	dwita	Benar
1164	dwita_0_6119.jpeg	dwita	dwita	Benar	dwita	Benar
1165	dwita_0_6172.jpeg	dwita	dwita	Benar	dwita	Benar
1166	dwita_0_6187.jpeg	dwita	dwita	Benar	dwita	Benar
1167	dwita_0_6260.jpeg	dwita	dwita	Benar	dwita	Benar
1168	dwita_0_6311.jpeg	dwita	dwita	Benar	dwita	Benar
1169	dwita_0_6346.jpeg	dwita	dwita	Benar	dwita	Benar
1170	dwita_0_6413.jpeg	dwita	dwita	Benar	dwita	Benar
1171	dwita_0_6440.jpeg	dwita	dwita	Benar	dwita	Benar
1172	dwita_0_6448.jpeg	dwita	dwita	Benar	dwita	Benar
1173	dwita_0_6486.jpeg	dwita	dwita	Benar	dwita	Benar
1174	dwita_0_6490.jpeg	dwita	dwita	Benar	dwita	Benar
1175	dwita_0_6546.jpeg	dwita	dwita	Benar	dwita	Benar
1176	dwita_0_6558.jpeg	dwita	dwita	Benar	dwita	Benar
1177	dwita_0_6569.jpeg	dwita	dwita	Benar	dwita	Benar
1178	dwita_0_6589.jpeg	dwita	dwita	Benar	dwita	Benar
1179	dwita_0_6599.jpeg	dwita	dwita	Benar	dwita	Benar
1180	dwita_0_6625.jpeg	dwita	dwita	Benar	dwita	Benar
1181	dwita_0_6659.jpeg	dwita	dwita	Benar	dwita	Benar
1182	dwita_0_6727.jpeg	dwita	dwita	Benar	dwita	Benar
1183	dwita_0_6767.jpeg	dwita	dwita	Benar	dwita	Benar
1184	dwita_0_680.jpeg	dwita	dwita	Benar	dwita	Benar
1185	dwita_0_685.jpeg	dwita	dwita	Benar	dwita	Benar
1186	dwita_0_6862.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1187	dwita_0_6878.jpeg	dwita	dwita	Benar	dwita	Benar
1188	dwita_0_6914.jpeg	dwita	dwita	Benar	dwita	Benar
1189	dwita_0_6917.jpeg	dwita	dwita	Benar	dwita	Benar
1190	dwita_0_6920.jpeg	dwita	dwita	Benar	dwita	Benar
1191	dwita_0_6925.jpeg	dwita	dwita	Benar	dwita	Benar
1192	dwita_0_6935.jpeg	dwita	dwita	Benar	dwita	Benar
1193	dwita_0_6951.jpeg	dwita	dwita	Benar	dwita	Benar
1194	dwita_0_6952.jpeg	dwita	dwita	Benar	dwita	Benar
1195	dwita_0_6962.jpeg	dwita	dwita	Benar	dwita	Benar
1196	dwita_0_6995.jpeg	dwita	dwita	Benar	dwita	Benar
1197	dwita_0_7227.jpeg	dwita	dwita	Benar	dwita	Benar
1198	dwita_0_724.jpeg	dwita	dwita	Benar	dwita	Benar
1199	dwita_0_7297.jpeg	dwita	dwita	Benar	dwita	Benar
1200	dwita_0_7323.jpeg	dwita	dwita	Benar	dwita	Benar
1201	dwita_0_7375.jpeg	dwita	dwita	Benar	dwita	Benar
1202	dwita_0_7432.jpeg	dwita	dwita	Benar	dwita	Benar
1203	dwita_0_7434.jpeg	dwita	dwita	Benar	dwita	Benar
1204	dwita_0_7481.jpeg	dwita	dwita	Benar	dwita	Benar
1205	dwita_0_7514.jpeg	dwita	dwita	Benar	dwita	Benar
1206	dwita_0_7580.jpeg	dwita	dwita	Benar	dwita	Benar
1207	dwita_0_7615.jpeg	dwita	dwita	Benar	dwita	Benar
1208	dwita_0_7616.jpeg	dwita	dwita	Benar	dwita	Benar
1209	dwita_0_7633.jpeg	dwita	dwita	Benar	dwita	Benar
1210	dwita_0_7653.jpeg	dwita	dwita	Benar	dwita	Benar
1211	dwita_0_7660.jpeg	dwita	dwita	Benar	dwita	Benar
1212	dwita_0_767.jpeg	dwita	dwita	Benar	dwita	Benar
1213	dwita_0_7891.jpeg	dwita	dwita	Benar	dwita	Benar
1214	dwita_0_7981.jpeg	dwita	dwita	Benar	dwita	Benar
1215	dwita_0_8024.jpeg	dwita	dwita	Benar	dwita	Benar
1216	dwita_0_8037.jpeg	dwita	dwita	Benar	dwita	Benar
1217	dwita_0_8165.jpeg	dwita	dwita	Benar	dwita	Benar
1218	dwita_0_8205.jpeg	dwita	dwita	Benar	dwita	Benar
1219	dwita_0_8206.jpeg	dwita	dwita	Benar	dwita	Benar
1220	dwita_0_8280.jpeg	dwita	dwita	Benar	dwita	Benar
1221	dwita_0_8299.jpeg	dwita	dwita	Benar	dwita	Benar
1222	dwita_0_8322.jpeg	dwita	dwita	Benar	dwita	Benar
1223	dwita_0_8386.jpeg	dwita	dwita	Benar	dwita	Benar
1224	dwita_0_8447.jpeg	dwita	dwita	Benar	dwita	Benar
1225	dwita_0_8482.jpeg	dwita	dwita	Benar	dwita	Benar
1226	dwita_0_852.jpeg	dwita	dwita	Benar	dwita	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1227	dwita_0_8687.jpeg	dwita	dwita	Benar	dwita	Benar
1228	dwita_0_8699.jpeg	dwita	dwita	Benar	dwita	Benar
1229	dwita_0_8752.jpeg	dwita	dwita	Benar	dwita	Benar
1230	dwita_0_8785.jpeg	dwita	dwita	Benar	dwita	Benar
1231	dwita_0_8869.jpeg	dwita	dwita	Benar	dwita	Benar
1232	dwita_0_887.jpeg	dwita	dwita	Benar	dwita	Benar
1233	dwita_0_8924.jpeg	dwita	dwita	Benar	dwita	Benar
1234	dwita_0_894.jpeg	dwita	dinda	Salah	dwita	Benar
1235	dwita_0_9056.jpeg	dwita	dwita	Benar	dwita	Benar
1236	dwita_0_9070.jpeg	dwita	dwita	Benar	dwita	Benar
1237	dwita_0_9123.jpeg	dwita	dwita	Benar	dwita	Benar
1238	dwita_0_9167.jpeg	dwita	dwita	Benar	dwita	Benar
1239	dwita_0_9214.jpeg	dwita	dwita	Benar	dwita	Benar
1240	dwita_0_9248.jpeg	dwita	dwita	Benar	dwita	Benar
1241	dwita_0_9278.jpeg	dwita	dwita	Benar	dwita	Benar
1242	dwita_0_9284.jpeg	dwita	dwita	Benar	dwita	Benar
1243	dwita_0_9285.jpeg	dwita	dwita	Benar	dwita	Benar
1244	dwita_0_9311.jpeg	dwita	dwita	Benar	dwita	Benar
1245	dwita_0_9319.jpeg	dwita	dwita	Benar	dwita	Benar
1246	dwita_0_9329.jpeg	dwita	dwita	Benar	dwita	Benar
1247	dwita_0_9395.jpeg	dwita	dwita	Benar	dwita	Benar
1248	dwita_0_9414.jpeg	dwita	dwita	Benar	dwita	Benar
1249	dwita_0_9420.jpeg	dwita	dwita	Benar	dwita	Benar
1250	dwita_0_9530.jpeg	dwita	dwita	Benar	dwita	Benar
1251	dwita_0_9603.jpeg	dwita	dwita	Benar	dwita	Benar
1252	dwita_0_9681.jpeg	dwita	dwita	Benar	dwita	Benar
1253	dwita_0_9709.jpeg	dwita	dwita	Benar	dwita	Benar
1254	dwita_0_9759.jpeg	dwita	dwita	Benar	dwita	Benar
1255	dwita_0_979.jpeg	dwita	dwita	Benar	dwita	Benar
1256	dwita_0_982.jpeg	dwita	dwita	Benar	dwita	Benar
1257	dwita_0_9909.jpeg	dwita	dwita	Benar	dwita	Benar
1258	dwita_0_9947.jpeg	dwita	dwita	Benar	dwita	Benar
1259	dwita_0_9969.jpeg	dwita	dwita	Benar	dwita	Benar
1260	farhan_0_0.jpeg	farhan	farhan	Benar	farhan	Benar
1261	farhan_0_1020.jpeg	farhan	farhan	Benar	farhan	Benar
1262	farhan_0_1047.jpeg	farhan	farhan	Benar	farhan	Benar
1263	farhan_0_1054.jpeg	farhan	farhan	Benar	farhan	Benar
1264	farhan_0_1080.jpeg	farhan	farhan	Benar	farhan	Benar
1265	farhan_0_1084.jpeg	farhan	farhan	Benar	farhan	Benar
1266	farhan_0_1118.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1267	farhan_0_134.jpeg	farhan	farhan	Benar	farhan	Benar
1268	farhan_0_1429.jpeg	farhan	farhan	Benar	farhan	Benar
1269	farhan_0_1481.jpeg	farhan	farhan	Benar	farhan	Benar
1270	farhan_0_1548.jpeg	farhan	farhan	Benar	farhan	Benar
1271	farhan_0_161.jpeg	farhan	farhan	Benar	farhan	Benar
1272	farhan_0_1617.jpeg	farhan	farhan	Benar	farhan	Benar
1273	farhan_0_1699.jpeg	farhan	farhan	Benar	farhan	Benar
1274	farhan_0_1713.jpeg	farhan	farhan	Benar	farhan	Benar
1275	farhan_0_1767.jpeg	farhan	farhan	Benar	farhan	Benar
1276	farhan_0_1797.jpeg	farhan	farhan	Benar	farhan	Benar
1277	farhan_0_1803.jpeg	farhan	farhan	Benar	farhan	Benar
1278	farhan_0_1849.jpeg	farhan	farhan	Benar	farhan	Benar
1279	farhan_0_1864.jpeg	farhan	farhan	Benar	farhan	Benar
1280	farhan_0_1889.jpeg	farhan	farhan	Benar	farhan	Benar
1281	farhan_0_1928.jpeg	farhan	farhan	Benar	farhan	Benar
1282	farhan_0_1973.jpeg	farhan	farhan	Benar	farhan	Benar
1283	farhan_0_1981.jpeg	farhan	farhan	Benar	farhan	Benar
1284	farhan_0_2005.jpeg	farhan	farhan	Benar	farhan	Benar
1285	farhan_0_2097.jpeg	farhan	farhan	Benar	farhan	Benar
1286	farhan_0_2169.jpeg	farhan	farhan	Benar	farhan	Benar
1287	farhan_0_2222.jpeg	farhan	farhan	Benar	farhan	Benar
1288	farhan_0_2232.jpeg	farhan	farhan	Benar	farhan	Benar
1289	farhan_0_228.jpeg	farhan	farhan	Benar	farhan	Benar
1290	farhan_0_2298.jpeg	farhan	farhan	Benar	farhan	Benar
1291	farhan_0_236.jpeg	farhan	farhan	Benar	farhan	Benar
1292	farhan_0_2411.jpeg	farhan	farhan	Benar	farhan	Benar
1293	farhan_0_2426.jpeg	farhan	farhan	Benar	farhan	Benar
1294	farhan_0_2464.jpeg	farhan	farhan	Benar	farhan	Benar
1295	farhan_0_2472.jpeg	farhan	farhan	Benar	farhan	Benar
1296	farhan_0_2486.jpeg	farhan	farhan	Benar	farhan	Benar
1297	farhan_0_2546.jpeg	farhan	farhan	Benar	farhan	Benar
1298	farhan_0_2627.jpeg	farhan	farhan	Benar	farhan	Benar
1299	farhan_0_2717.jpeg	farhan	farhan	Benar	farhan	Benar
1300	farhan_0_2743.jpeg	farhan	farhan	Benar	farhan	Benar
1301	farhan_0_2785.jpeg	farhan	farhan	Benar	farhan	Benar
1302	farhan_0_2795.jpeg	farhan	farhan	Benar	farhan	Benar
1303	farhan_0_2832.jpeg	farhan	farhan	Benar	farhan	Benar
1304	farhan_0_2873.jpeg	farhan	farhan	Benar	farhan	Benar
1305	farhan_0_288.jpeg	farhan	farhan	Benar	farhan	Benar
1306	farhan_0_2896.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1307	farhan_0_2976.jpeg	farhan	farhan	Benar	farhan	Benar
1308	farhan_0_3032.jpeg	farhan	farhan	Benar	farhan	Benar
1309	farhan_0_3117.jpeg	farhan	farhan	Benar	farhan	Benar
1310	farhan_0_3135.jpeg	farhan	farhan	Benar	farhan	Benar
1311	farhan_0_3171.jpeg	farhan	farhan	Benar	farhan	Benar
1312	farhan_0_3205.jpeg	farhan	farhan	Benar	farhan	Benar
1313	farhan_0_3433.jpeg	farhan	farhan	Benar	farhan	Benar
1314	farhan_0_3437.jpeg	farhan	farhan	Benar	farhan	Benar
1315	farhan_0_3464.jpeg	farhan	farhan	Benar	farhan	Benar
1316	farhan_0_3478.jpeg	farhan	farhan	Benar	farhan	Benar
1317	farhan_0_3483.jpeg	farhan	farhan	Benar	farhan	Benar
1318	farhan_0_350.jpeg	farhan	farhan	Benar	farhan	Benar
1319	farhan_0_3528.jpeg	farhan	farhan	Benar	farhan	Benar
1320	farhan_0_3555.jpeg	farhan	farhan	Benar	farhan	Benar
1321	farhan_0_3636.jpeg	farhan	farhan	Benar	farhan	Benar
1322	farhan_0_3793.jpeg	farhan	farhan	Benar	farhan	Benar
1323	farhan_0_3835.jpeg	farhan	farhan	Benar	farhan	Benar
1324	farhan_0_1777.jpeg	farhan	farhan	Benar	farhan	Benar
1325	farhan_0_2405.jpeg	farhan	ilham	Salah	farhan	Benar
1326	farhan_0_3003.jpeg	farhan	farhan	Benar	farhan	Benar
1327	farhan_0_3877.jpeg	farhan	farhan	Benar	farhan	Benar
1328	farhan_0_457.jpeg	farhan	farhan	Benar	farhan	Benar
1329	farhan_0_5371.jpeg	farhan	farhan	Benar	farhan	Benar
1330	farhan_0_6083.jpeg	farhan	farhan	Benar	farhan	Benar
1331	farhan_0_6691.jpeg	farhan	farhan	Benar	farhan	Benar
1332	farhan_0_7154.jpeg	farhan	farhan	Benar	farhan	Benar
1333	farhan_0_7610.jpeg	farhan	farhan	Benar	farhan	Benar
1334	farhan_0_8055.jpeg	farhan	farhan	Benar	farhan	Benar
1335	farhan_0_8735.jpeg	farhan	farhan	Benar	farhan	Benar
1336	farhan_0_9354.jpeg	farhan	farhan	Benar	farhan	Benar
1337	farhan_0_3888.jpeg	farhan	farhan	Benar	farhan	Benar
1338	farhan_0_391.jpeg	farhan	farhan	Benar	farhan	Benar
1339	farhan_0_3958.jpeg	farhan	farhan	Benar	farhan	Benar
1340	farhan_0_4053.jpeg	farhan	farhan	Benar	farhan	Benar
1341	farhan_0_4054.jpeg	farhan	farhan	Benar	farhan	Benar
1342	farhan_0_406.jpeg	farhan	farhan	Benar	farhan	Benar
1343	farhan_0_4150.jpeg	farhan	farhan	Benar	farhan	Benar
1344	farhan_0_4160.jpeg	farhan	farhan	Benar	farhan	Benar
1345	farhan_0_42.jpeg	farhan	farhan	Benar	farhan	Benar
1346	farhan_0_420.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1347	farhan_0_4229.jpeg	farhan	farhan	Benar	farhan	Benar
1348	farhan_0_4235.jpeg	farhan	farhan	Benar	farhan	Benar
1349	farhan_0_4337.jpeg	farhan	farhan	Benar	farhan	Benar
1350	farhan_0_4389.jpeg	farhan	farhan	Benar	farhan	Benar
1351	farhan_0_4475.jpeg	farhan	farhan	Benar	farhan	Benar
1352	farhan_0_4535.jpeg	farhan	farhan	Benar	farhan	Benar
1353	farhan_0_4666.jpeg	farhan	farhan	Benar	farhan	Benar
1354	farhan_0_4680.jpeg	farhan	farhan	Benar	farhan	Benar
1355	farhan_0_4694.jpeg	farhan	farhan	Benar	farhan	Benar
1356	farhan_0_4769.jpeg	farhan	farhan	Benar	farhan	Benar
1357	farhan_0_4871.jpeg	farhan	farhan	Benar	farhan	Benar
1358	farhan_0_4903.jpeg	farhan	farhan	Benar	farhan	Benar
1359	farhan_0_4959.jpeg	farhan	farhan	Benar	farhan	Benar
1360	farhan_0_496.jpeg	farhan	farhan	Benar	farhan	Benar
1361	farhan_0_5042.jpeg	farhan	farhan	Benar	farhan	Benar
1362	farhan_0_5098.jpeg	farhan	farhan	Benar	farhan	Benar
1363	farhan_0_5220.jpeg	farhan	farhan	Benar	farhan	Benar
1364	farhan_0_5284.jpeg	farhan	farhan	Benar	farhan	Benar
1365	farhan_0_5304.jpeg	farhan	farhan	Benar	farhan	Benar
1366	farhan_0_5323.jpeg	farhan	farhan	Benar	farhan	Benar
1367	farhan_0_5345.jpeg	farhan	farhan	Benar	farhan	Benar
1368	farhan_0_5350.jpeg	farhan	farhan	Benar	farhan	Benar
1369	farhan_0_5393.jpeg	farhan	farhan	Benar	farhan	Benar
1370	farhan_0_542.jpeg	farhan	farhan	Benar	farhan	Benar
1371	farhan_0_5451.jpeg	farhan	farhan	Benar	farhan	Benar
1372	farhan_0_5466.jpeg	farhan	farhan	Benar	farhan	Benar
1373	farhan_0_5623.jpeg	farhan	farhan	Benar	farhan	Benar
1374	farhan_0_5775.jpeg	farhan	farhan	Benar	farhan	Benar
1375	farhan_0_5830.jpeg	farhan	farhan	Benar	farhan	Benar
1376	farhan_0_5852.jpeg	farhan	farhan	Benar	farhan	Benar
1377	farhan_0_5917.jpeg	farhan	farhan	Benar	farhan	Benar
1378	farhan_0_5938.jpeg	farhan	farhan	Benar	farhan	Benar
1379	farhan_0_5947.jpeg	farhan	farhan	Benar	farhan	Benar
1380	farhan_0_6006.jpeg	farhan	farhan	Benar	farhan	Benar
1381	farhan_0_6010.jpeg	farhan	farhan	Benar	farhan	Benar
1382	farhan_0_602.jpeg	farhan	farhan	Benar	farhan	Benar
1383	farhan_0_6024.jpeg	farhan	farhan	Benar	farhan	Benar
1384	farhan_0_6032.jpeg	farhan	farhan	Benar	farhan	Benar
1385	farhan_0_6097.jpeg	farhan	farhan	Benar	farhan	Benar
1386	farhan_0_6098.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1387	farhan_0_6188.jpeg	farhan	farhan	Benar	farhan	Benar
1388	farhan_0_629.jpeg	farhan	farhan	Benar	farhan	Benar
1389	farhan_0_630.jpeg	farhan	farhan	Benar	farhan	Benar
1390	farhan_0_6310.jpeg	farhan	farhan	Benar	farhan	Benar
1391	farhan_0_6362.jpeg	farhan	farhan	Benar	farhan	Benar
1392	farhan_0_6384.jpeg	farhan	farhan	Benar	farhan	Benar
1393	farhan_0_639.jpeg	farhan	farhan	Benar	farhan	Benar
1394	farhan_0_6407.jpeg	farhan	farhan	Benar	farhan	Benar
1395	farhan_0_6455.jpeg	farhan	farhan	Benar	farhan	Benar
1396	farhan_0_6496.jpeg	farhan	farhan	Benar	farhan	Benar
1397	farhan_0_6498.jpeg	farhan	farhan	Benar	farhan	Benar
1398	farhan_0_6508.jpeg	farhan	farhan	Benar	farhan	Benar
1399	farhan_0_6596.jpeg	farhan	farhan	Benar	farhan	Benar
1400	farhan_0_6636.jpeg	farhan	farhan	Benar	farhan	Benar
1401	farhan_0_6700.jpeg	farhan	farhan	Benar	farhan	Benar
1402	farhan_0_6746.jpeg	farhan	farhan	Benar	farhan	Benar
1403	farhan_0_6766.jpeg	farhan	farhan	Benar	farhan	Benar
1404	farhan_0_6769.jpeg	farhan	farhan	Benar	farhan	Benar
1405	farhan_0_6785.jpeg	farhan	farhan	Benar	farhan	Benar
1406	farhan_0_6790.jpeg	farhan	farhan	Benar	farhan	Benar
1407	farhan_0_6880.jpeg	farhan	farhan	Benar	farhan	Benar
1408	farhan_0_6882.jpeg	farhan	farhan	Benar	farhan	Benar
1409	farhan_0_691.jpeg	farhan	farhan	Benar	farhan	Benar
1410	farhan_0_6915.jpeg	farhan	farhan	Benar	farhan	Benar
1411	farhan_0_6995.jpeg	farhan	farhan	Benar	farhan	Benar
1412	farhan_0_7013.jpeg	farhan	farhan	Benar	farhan	Benar
1413	farhan_0_7021.jpeg	farhan	farhan	Benar	farhan	Benar
1414	farhan_0_7076.jpeg	farhan	farhan	Benar	farhan	Benar
1415	farhan_0_7085.jpeg	farhan	farhan	Benar	farhan	Benar
1416	farhan_0_7137.jpeg	farhan	farhan	Benar	farhan	Benar
1417	farhan_0_717.jpeg	farhan	farhan	Benar	farhan	Benar
1418	farhan_0_7234.jpeg	farhan	farhan	Benar	farhan	Benar
1419	farhan_0_7237.jpeg	farhan	farhan	Benar	farhan	Benar
1420	farhan_0_7245.jpeg	farhan	farhan	Benar	farhan	Benar
1421	farhan_0_7296.jpeg	farhan	farhan	Benar	farhan	Benar
1422	farhan_0_7315.jpeg	farhan	farhan	Benar	farhan	Benar
1423	farhan_0_7323.jpeg	farhan	farhan	Benar	farhan	Benar
1424	farhan_0_7325.jpeg	farhan	farhan	Benar	farhan	Benar
1425	farhan_0_7350.jpeg	farhan	farhan	Benar	farhan	Benar
1426	farhan_0_7359.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1427	farhan_0_7367.jpeg	farhan	farhan	Benar	farhan	Benar
1428	farhan_0_7369.jpeg	farhan	farhan	Benar	farhan	Benar
1429	farhan_0_744.jpeg	farhan	farhan	Benar	farhan	Benar
1430	farhan_0_7502.jpeg	farhan	farhan	Benar	farhan	Benar
1431	farhan_0_7549.jpeg	farhan	farhan	Benar	farhan	Benar
1432	farhan_0_7592.jpeg	farhan	farhan	Benar	farhan	Benar
1433	farhan_0_7614.jpeg	farhan	farhan	Benar	farhan	Benar
1434	farhan_0_7617.jpeg	farhan	farhan	Benar	farhan	Benar
1435	farhan_0_7644.jpeg	farhan	farhan	Benar	farhan	Benar
1436	farhan_0_7715.jpeg	farhan	farhan	Benar	farhan	Benar
1437	farhan_0_7745.jpeg	farhan	farhan	Benar	farhan	Benar
1438	farhan_0_7756.jpeg	farhan	farhan	Benar	farhan	Benar
1439	farhan_0_7762.jpeg	farhan	farhan	Benar	farhan	Benar
1440	farhan_0_777.jpeg	farhan	farhan	Benar	farhan	Benar
1441	farhan_0_7770.jpeg	farhan	farhan	Benar	farhan	Benar
1442	farhan_0_7795.jpeg	farhan	farhan	Benar	farhan	Benar
1443	farhan_0_7798.jpeg	farhan	farhan	Benar	farhan	Benar
1444	farhan_0_784.jpeg	farhan	farhan	Benar	farhan	Benar
1445	farhan_0_7863.jpeg	farhan	farhan	Benar	farhan	Benar
1446	farhan_0_7881.jpeg	farhan	farhan	Benar	farhan	Benar
1447	farhan_0_7920.jpeg	farhan	farhan	Benar	farhan	Benar
1448	farhan_0_7955.jpeg	farhan	farhan	Benar	farhan	Benar
1449	farhan_0_8146.jpeg	farhan	farhan	Benar	farhan	Benar
1450	farhan_0_8186.jpeg	farhan	farhan	Benar	farhan	Benar
1451	farhan_0_8310.jpeg	farhan	farhan	Benar	farhan	Benar
1452	farhan_0_8312.jpeg	farhan	farhan	Benar	farhan	Benar
1453	farhan_0_8348.jpeg	farhan	farhan	Benar	farhan	Benar
1454	farhan_0_8363.jpeg	farhan	farhan	Benar	farhan	Benar
1455	farhan_0_8372.jpeg	farhan	farhan	Benar	farhan	Benar
1456	farhan_0_8398.jpeg	farhan	farhan	Benar	farhan	Benar
1457	farhan_0_8453.jpeg	farhan	farhan	Benar	farhan	Benar
1458	farhan_0_85.jpeg	farhan	farhan	Benar	farhan	Benar
1459	farhan_0_8522.jpeg	farhan	farhan	Benar	farhan	Benar
1460	farhan_0_8535.jpeg	farhan	farhan	Benar	farhan	Benar
1461	farhan_0_8542.jpeg	farhan	farhan	Benar	farhan	Benar
1462	farhan_0_8583.jpeg	farhan	farhan	Benar	farhan	Benar
1463	farhan_0_8641.jpeg	farhan	farhan	Benar	farhan	Benar
1464	farhan_0_8683.jpeg	farhan	farhan	Benar	farhan	Benar
1465	farhan_0_874.jpeg	farhan	ilham	Salah	farhan	Benar
1466	farhan_0_8740.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1467	farhan_0_8841.jpeg	farhan	farhan	Benar	farhan	Benar
1468	farhan_0_8881.jpeg	farhan	farhan	Benar	farhan	Benar
1469	farhan_0_8943.jpeg	farhan	farhan	Benar	farhan	Benar
1470	farhan_0_9012.jpeg	farhan	farhan	Benar	farhan	Benar
1471	farhan_0_9052.jpeg	farhan	farhan	Benar	farhan	Benar
1472	farhan_0_9058.jpeg	farhan	farhan	Benar	farhan	Benar
1473	farhan_0_9115.jpeg	farhan	farhan	Benar	farhan	Benar
1474	farhan_0_9178.jpeg	farhan	farhan	Benar	farhan	Benar
1475	farhan_0_9220.jpeg	farhan	farhan	Benar	farhan	Benar
1476	farhan_0_9240.jpeg	farhan	farhan	Benar	farhan	Benar
1477	farhan_0_9276.jpeg	farhan	farhan	Benar	farhan	Benar
1478	farhan_0_928.jpeg	farhan	farhan	Benar	farhan	Benar
1479	farhan_0_9323.jpeg	farhan	farhan	Benar	farhan	Benar
1480	farhan_0_935.jpeg	farhan	farhan	Benar	farhan	Benar
1481	farhan_0_9360.jpeg	farhan	farhan	Benar	farhan	Benar
1482	farhan_0_9362.jpeg	farhan	farhan	Benar	farhan	Benar
1483	farhan_0_9375.jpeg	farhan	farhan	Benar	farhan	Benar
1484	farhan_0_9395.jpeg	farhan	farhan	Benar	farhan	Benar
1485	farhan_0_9469.jpeg	farhan	farhan	Benar	farhan	Benar
1486	farhan_0_9518.jpeg	farhan	farhan	Benar	farhan	Benar
1487	farhan_0_9538.jpeg	farhan	farhan	Benar	farhan	Benar
1488	farhan_0_9544.jpeg	farhan	farhan	Benar	farhan	Benar
1489	farhan_0_9576.jpeg	farhan	farhan	Benar	farhan	Benar
1490	farhan_0_9632.jpeg	farhan	farhan	Benar	farhan	Benar
1491	farhan_0_9654.jpeg	farhan	farhan	Benar	farhan	Benar
1492	farhan_0_9655.jpeg	farhan	farhan	Benar	farhan	Benar
1493	farhan_0_9658.jpeg	farhan	farhan	Benar	farhan	Benar
1494	farhan_0_9683.jpeg	farhan	farhan	Benar	farhan	Benar
1495	farhan_0_9715.jpeg	farhan	farhan	Benar	farhan	Benar
1496	farhan_0_9770.jpeg	farhan	farhan	Benar	farhan	Benar
1497	farhan_0_9837.jpeg	farhan	farhan	Benar	farhan	Benar
1498	farhan_0_9852.jpeg	farhan	farhan	Benar	farhan	Benar
1499	farhan_0_9861.jpeg	farhan	farhan	Benar	farhan	Benar
1500	farhan_0_9895.jpeg	farhan	farhan	Benar	farhan	Benar
1501	farhan_0_9914.jpeg	farhan	farhan	Benar	farhan	Benar
1502	farhan_0_9925.jpeg	farhan	farhan	Benar	farhan	Benar
1503	farhan_0_9927.jpeg	farhan	farhan	Benar	farhan	Benar
1504	farhan_0_9955.jpeg	farhan	farhan	Benar	farhan	Benar
1505	farhan_0_9956.jpeg	farhan	farhan	Benar	farhan	Benar
1506	farhan_0_9958.jpeg	farhan	farhan	Benar	farhan	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1507	farhan_0_9979.jpeg	farhan	farhan	Benar	farhan	Benar
1508	farhan_0_9992.jpeg	farhan	ilham	Salah	farhan	Benar
1509	ilham_0_1009.jpeg	ilham	ilham	Benar	ilham	Benar
1510	ilham_0_1204.jpeg	ilham	ilham	Benar	ilham	Benar
1511	ilham_0_1305.jpeg	ilham	ilham	Benar	ilham	Benar
1512	ilham_0_1331.jpeg	ilham	ilham	Benar	ilham	Benar
1513	ilham_0_1345.jpeg	ilham	ilham	Benar	ilham	Benar
1514	ilham_0_14.jpeg	ilham	ilham	Benar	ilham	Benar
1515	ilham_0_1439.jpeg	ilham	ilham	Benar	ilham	Benar
1516	ilham_0_1473.jpeg	ilham	ilham	Benar	ilham	Benar
1517	ilham_0_1481.jpeg	ilham	ilham	Benar	ilham	Benar
1518	ilham_0_1531.jpeg	ilham	ilham	Benar	ilham	Benar
1519	ilham_0_1598.jpeg	ilham	ilham	Benar	ilham	Benar
1520	ilham_0_1620.jpeg	ilham	ilham	Benar	ilham	Benar
1521	ilham_0_1662.jpeg	ilham	ilham	Benar	ilham	Benar
1522	ilham_0_1719.jpeg	ilham	ilham	Benar	ilham	Benar
1523	ilham_0_1729.jpeg	ilham	ilham	Benar	ilham	Benar
1524	ilham_0_1759.jpeg	ilham	ilham	Benar	ilham	Benar
1525	ilham_0_1768.jpeg	ilham	ilham	Benar	ilham	Benar
1526	ilham_0_1837.jpeg	ilham	ilham	Benar	ilham	Benar
1527	ilham_0_184.jpeg	ilham	ilham	Benar	ilham	Benar
1528	ilham_0_1863.jpeg	ilham	ilham	Benar	ilham	Benar
1529	ilham_0_1875.jpeg	ilham	ilham	Benar	ilham	Benar
1530	ilham_0_1885.jpeg	ilham	ilham	Benar	ilham	Benar
1531	ilham_0_1895.jpeg	ilham	ilham	Benar	ilham	Benar
1532	ilham_0_1908.jpeg	ilham	ilham	Benar	ilham	Benar
1533	ilham_0_1959.jpeg	ilham	ilham	Benar	ilham	Benar
1534	ilham_0_2065.jpeg	ilham	ilham	Benar	ilham	Benar
1535	ilham_0_2078.jpeg	ilham	ilham	Benar	ilham	Benar
1536	ilham_0_2090.jpeg	ilham	ilham	Benar	ilham	Benar
1537	ilham_0_2097.jpeg	ilham	ilham	Benar	ilham	Benar
1538	ilham_0_2123.jpeg	ilham	ilham	Benar	ilham	Benar
1539	ilham_0_2126.jpeg	ilham	ilham	Benar	ilham	Benar
1540	ilham_0_2226.jpeg	ilham	ilham	Benar	ilham	Benar
1541	ilham_0_2238.jpeg	ilham	ilham	Benar	ilham	Benar
1542	ilham_0_2246.jpeg	ilham	ilham	Benar	ilham	Benar
1543	ilham_0_2248.jpeg	ilham	ilham	Benar	ilham	Benar
1544	ilham_0_2250.jpeg	ilham	ilham	Benar	ilham	Benar
1545	ilham_0_2258.jpeg	ilham	ilham	Benar	ilham	Benar
1546	ilham_0_2261.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1547	ilham_0_2289.jpeg	ilham	ilham	Benar	ilham	Benar
1548	ilham_0_2335.jpeg	ilham	ilham	Benar	ilham	Benar
1549	ilham_0_2368.jpeg	ilham	ilham	Benar	ilham	Benar
1550	ilham_0_2416.jpeg	ilham	ilham	Benar	ilham	Benar
1551	ilham_0_2419.jpeg	ilham	ilham	Benar	ilham	Benar
1552	ilham_0_2474.jpeg	ilham	ilham	Benar	ilham	Benar
1553	ilham_0_2514.jpeg	ilham	ilham	Benar	ilham	Benar
1554	ilham_0_252.jpeg	ilham	ilham	Benar	ilham	Benar
1555	ilham_0_2576.jpeg	ilham	ilham	Benar	ilham	Benar
1556	ilham_0_2640.jpeg	ilham	ilham	Benar	ilham	Benar
1557	ilham_0_2757.jpeg	ilham	ilham	Benar	ilham	Benar
1558	ilham_0_2777.jpeg	ilham	ilham	Benar	ilham	Benar
1559	ilham_0_2791.jpeg	ilham	ilham	Benar	ilham	Benar
1560	ilham_0_2809.jpeg	ilham	ilham	Benar	ilham	Benar
1561	ilham_0_2905.jpeg	ilham	ilham	Benar	ilham	Benar
1562	ilham_0_2949.jpeg	ilham	ilham	Benar	ilham	Benar
1563	ilham_0_3038.jpeg	ilham	ilham	Benar	ilham	Benar
1564	ilham_0_306.jpeg	ilham	ilham	Benar	ilham	Benar
1565	ilham_0_308.jpeg	ilham	ilham	Benar	ilham	Benar
1566	ilham_0_3138.jpeg	ilham	ilham	Benar	ilham	Benar
1567	ilham_0_317.jpeg	ilham	ilham	Benar	ilham	Benar
1568	ilham_0_3226.jpeg	ilham	ilham	Benar	ilham	Benar
1569	ilham_0_3239.jpeg	ilham	ilham	Benar	ilham	Benar
1570	ilham_0_3274.jpeg	ilham	ilham	Benar	ilham	Benar
1571	ilham_0_3336.jpeg	ilham	ilham	Benar	ilham	Benar
1572	ilham_0_3372.jpeg	ilham	ilham	Benar	ilham	Benar
1573	ilham_0_1760.jpeg	ilham	ilham	Benar	ilham	Benar
1574	ilham_0_223.jpeg	ilham	ilham	Benar	ilham	Benar
1575	ilham_0_2678.jpeg	ilham	ilham	Benar	ilham	Benar
1576	ilham_0_3380.jpeg	ilham	ilham	Benar	ilham	Benar
1577	ilham_0_4048.jpeg	ilham	ilham	Benar	ilham	Benar
1578	ilham_0_4637.jpeg	ilham	ilham	Benar	ilham	Benar
1579	ilham_0_515.jpeg	ilham	ilham	Benar	ilham	Benar
1580	ilham_0_5566.jpeg	ilham	ilham	Benar	ilham	Benar
1581	ilham_0_6081.jpeg	ilham	ilham	Benar	ilham	Benar
1582	ilham_0_6790.jpeg	ilham	ilham	Benar	ilham	Benar
1583	ilham_0_7459.jpeg	ilham	ilham	Benar	ilham	Benar
1584	ilham_0_7969.jpeg	ilham	ilham	Benar	ilham	Benar
1585	ilham_0_8710.jpeg	ilham	ilham	Benar	ilham	Benar
1586	ilham_0_9407.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1587	ilham_0_3388.jpeg	ilham	ilham	Benar	ilham	Benar
1588	ilham_0_3498.jpeg	ilham	ilham	Benar	ilham	Benar
1589	ilham_0_3503.jpeg	ilham	ilham	Benar	ilham	Benar
1590	ilham_0_3523.jpeg	ilham	ilham	Benar	ilham	Benar
1591	ilham_0_3551.jpeg	ilham	ilham	Benar	ilham	Benar
1592	ilham_0_3564.jpeg	ilham	ilham	Benar	ilham	Benar
1593	ilham_0_3620.jpeg	ilham	ilham	Benar	ilham	Benar
1594	ilham_0_3628.jpeg	ilham	ilham	Benar	ilham	Benar
1595	ilham_0_3660.jpeg	ilham	ilham	Benar	ilham	Benar
1596	ilham_0_3755.jpeg	ilham	ilham	Benar	ilham	Benar
1597	ilham_0_3801.jpeg	ilham	ilham	Benar	ilham	Benar
1598	ilham_0_3869.jpeg	ilham	ilham	Benar	ilham	Benar
1599	ilham_0_3913.jpeg	ilham	ilham	Benar	ilham	Benar
1600	ilham_0_3988.jpeg	ilham	ilham	Benar	ilham	Benar
1601	ilham_0_3994.jpeg	ilham	ilham	Benar	ilham	Benar
1602	ilham_0_4026.jpeg	ilham	ilham	Benar	ilham	Benar
1603	ilham_0_4182.jpeg	ilham	ilham	Benar	ilham	Benar
1604	ilham_0_4234.jpeg	ilham	ilham	Benar	ilham	Benar
1605	ilham_0_4265.jpeg	ilham	ilham	Benar	ilham	Benar
1606	ilham_0_4272.jpeg	ilham	ilham	Benar	ilham	Benar
1607	ilham_0_4310.jpeg	ilham	ilham	Benar	ilham	Benar
1608	ilham_0_4318.jpeg	ilham	ilham	Benar	ilham	Benar
1609	ilham_0_4326.jpeg	ilham	ilham	Benar	ilham	Benar
1610	ilham_0_4415.jpeg	ilham	ilham	Benar	ilham	Benar
1611	ilham_0_444.jpeg	ilham	ilham	Benar	ilham	Benar
1612	ilham_0_4448.jpeg	ilham	ilham	Benar	ilham	Benar
1613	ilham_0_4466.jpeg	ilham	ilham	Benar	ilham	Benar
1614	ilham_0_4494.jpeg	ilham	ilham	Benar	ilham	Benar
1615	ilham_0_4519.jpeg	ilham	ilham	Benar	ilham	Benar
1616	ilham_0_4553.jpeg	ilham	ilham	Benar	ilham	Benar
1617	ilham_0_4579.jpeg	ilham	ilham	Benar	ilham	Benar
1618	ilham_0_4618.jpeg	ilham	ilham	Benar	ilham	Benar
1619	ilham_0_4661.jpeg	ilham	ilham	Benar	ilham	Benar
1620	ilham_0_471.jpeg	ilham	ilham	Benar	ilham	Benar
1621	ilham_0_473.jpeg	ilham	ilham	Benar	ilham	Benar
1622	ilham_0_4758.jpeg	ilham	ilham	Benar	ilham	Benar
1623	ilham_0_4765.jpeg	ilham	ilham	Benar	ilham	Benar
1624	ilham_0_4803.jpeg	ilham	ilham	Benar	ilham	Benar
1625	ilham_0_4865.jpeg	ilham	ilham	Benar	ilham	Benar
1626	ilham_0_4873.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1627	ilham_0_4992.jpeg	ilham	ilham	Benar	ilham	Benar
1628	ilham_0_500.jpeg	ilham	ilham	Benar	ilham	Benar
1629	ilham_0_5043.jpeg	ilham	ilham	Benar	ilham	Benar
1630	ilham_0_5048.jpeg	ilham	ilham	Benar	ilham	Benar
1631	ilham_0_5079.jpeg	ilham	ilham	Benar	ilham	Benar
1632	ilham_0_5114.jpeg	ilham	ilham	Benar	ilham	Benar
1633	ilham_0_5126.jpeg	ilham	ilham	Benar	ilham	Benar
1634	ilham_0_514.jpeg	ilham	ilham	Benar	ilham	Benar
1635	ilham_0_5193.jpeg	ilham	ilham	Benar	ilham	Benar
1636	ilham_0_5210.jpeg	ilham	ilham	Benar	ilham	Benar
1637	ilham_0_5217.jpeg	ilham	ilham	Benar	ilham	Benar
1638	ilham_0_523.jpeg	ilham	ilham	Benar	ilham	Benar
1639	ilham_0_5260.jpeg	ilham	ilham	Benar	ilham	Benar
1640	ilham_0_5302.jpeg	ilham	ilham	Benar	ilham	Benar
1641	ilham_0_531.jpeg	ilham	ilham	Benar	ilham	Benar
1642	ilham_0_5341.jpeg	ilham	ilham	Benar	ilham	Benar
1643	ilham_0_5364.jpeg	ilham	ilham	Benar	ilham	Benar
1644	ilham_0_5368.jpeg	ilham	ilham	Benar	ilham	Benar
1645	ilham_0_5401.jpeg	ilham	ilham	Benar	ilham	Benar
1646	ilham_0_5410.jpeg	ilham	ilham	Benar	ilham	Benar
1647	ilham_0_5411.jpeg	ilham	ilham	Benar	ilham	Benar
1648	ilham_0_5443.jpeg	ilham	ilham	Benar	ilham	Benar
1649	ilham_0_5447.jpeg	ilham	ilham	Benar	ilham	Benar
1650	ilham_0_5494.jpeg	ilham	ilham	Benar	ilham	Benar
1651	ilham_0_557.jpeg	ilham	ilham	Benar	ilham	Benar
1652	ilham_0_5639.jpeg	ilham	ilham	Benar	ilham	Benar
1653	ilham_0_5673.jpeg	ilham	ilham	Benar	ilham	Benar
1654	ilham_0_5693.jpeg	ilham	ilham	Benar	ilham	Benar
1655	ilham_0_5711.jpeg	ilham	ilham	Benar	ilham	Benar
1656	ilham_0_5714.jpeg	ilham	ilham	Benar	ilham	Benar
1657	ilham_0_5719.jpeg	ilham	ilham	Benar	ilham	Benar
1658	ilham_0_5738.jpeg	ilham	ilham	Benar	ilham	Benar
1659	ilham_0_5876.jpeg	ilham	ilham	Benar	ilham	Benar
1660	ilham_0_5889.jpeg	ilham	ilham	Benar	ilham	Benar
1661	ilham_0_5897.jpeg	ilham	ilham	Benar	ilham	Benar
1662	ilham_0_5908.jpeg	ilham	ilham	Benar	ilham	Benar
1663	ilham_0_5958.jpeg	ilham	ilham	Benar	ilham	Benar
1664	ilham_0_603.jpeg	ilham	ilham	Benar	ilham	Benar
1665	ilham_0_6044.jpeg	ilham	ilham	Benar	ilham	Benar
1666	ilham_0_6050.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1667	ilham_0_6267.jpeg	ilham	ilham	Benar	ilham	Benar
1668	ilham_0_628.jpeg	ilham	ilham	Benar	ilham	Benar
1669	ilham_0_633.jpeg	ilham	ilham	Benar	ilham	Benar
1670	ilham_0_6344.jpeg	ilham	ilham	Benar	ilham	Benar
1671	ilham_0_6388.jpeg	ilham	ilham	Benar	ilham	Benar
1672	ilham_0_641.jpeg	ilham	ilham	Benar	ilham	Benar
1673	ilham_0_6483.jpeg	ilham	ilham	Benar	ilham	Benar
1674	ilham_0_6500.jpeg	ilham	ilham	Benar	ilham	Benar
1675	ilham_0_6543.jpeg	ilham	ilham	Benar	ilham	Benar
1676	ilham_0_6547.jpeg	ilham	ilham	Benar	ilham	Benar
1677	ilham_0_662.jpeg	ilham	ilham	Benar	ilham	Benar
1678	ilham_0_6624.jpeg	ilham	ilham	Benar	ilham	Benar
1679	ilham_0_6631.jpeg	ilham	ilham	Benar	ilham	Benar
1680	ilham_0_6676.jpeg	ilham	ilham	Benar	ilham	Benar
1681	ilham_0_6678.jpeg	ilham	ilham	Benar	ilham	Benar
1682	ilham_0_6768.jpeg	ilham	ilham	Benar	ilham	Benar
1683	ilham_0_6821.jpeg	ilham	ilham	Benar	ilham	Benar
1684	ilham_0_6847.jpeg	ilham	ilham	Benar	ilham	Benar
1685	ilham_0_6851.jpeg	ilham	ilham	Benar	ilham	Benar
1686	ilham_0_6892.jpeg	ilham	ilham	Benar	ilham	Benar
1687	ilham_0_69.jpeg	ilham	ilham	Benar	ilham	Benar
1688	ilham_0_6938.jpeg	ilham	ilham	Benar	ilham	Benar
1689	ilham_0_6994.jpeg	ilham	ilham	Benar	ilham	Benar
1690	ilham_0_7034.jpeg	ilham	ilham	Benar	ilham	Benar
1691	ilham_0_7102.jpeg	ilham	ilham	Benar	ilham	Benar
1692	ilham_0_7117.jpeg	ilham	ilham	Benar	ilham	Benar
1693	ilham_0_7148.jpeg	ilham	ilham	Benar	ilham	Benar
1694	ilham_0_7163.jpeg	ilham	ilham	Benar	ilham	Benar
1695	ilham_0_7206.jpeg	ilham	ilham	Benar	ilham	Benar
1696	ilham_0_727.jpeg	ilham	ilham	Benar	ilham	Benar
1697	ilham_0_7308.jpeg	ilham	ilham	Benar	ilham	Benar
1698	ilham_0_7354.jpeg	ilham	ilham	Benar	ilham	Benar
1699	ilham_0_7522.jpeg	ilham	ilham	Benar	ilham	Benar
1700	ilham_0_7577.jpeg	ilham	ilham	Benar	ilham	Benar
1701	ilham_0_7590.jpeg	ilham	ilham	Benar	ilham	Benar
1702	ilham_0_7596.jpeg	ilham	ilham	Benar	ilham	Benar
1703	ilham_0_7606.jpeg	ilham	ilham	Benar	ilham	Benar
1704	ilham_0_7635.jpeg	ilham	ilham	Benar	ilham	Benar
1705	ilham_0_7694.jpeg	ilham	ilham	Benar	ilham	Benar
1706	ilham_0_772.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1707	ilham_0_7729.jpeg	ilham	ilham	Benar	ilham	Benar
1708	ilham_0_773.jpeg	ilham	ilham	Benar	ilham	Benar
1709	ilham_0_7776.jpeg	ilham	ilham	Benar	ilham	Benar
1710	ilham_0_7812.jpeg	ilham	ilham	Benar	ilham	Benar
1711	ilham_0_7832.jpeg	ilham	ilham	Benar	ilham	Benar
1712	ilham_0_7865.jpeg	ilham	ilham	Benar	ilham	Benar
1713	ilham_0_7923.jpeg	ilham	ilham	Benar	ilham	Benar
1714	ilham_0_7935.jpeg	ilham	ilham	Benar	ilham	Benar
1715	ilham_0_7999.jpeg	ilham	ilham	Benar	ilham	Benar
1716	ilham_0_8001.jpeg	ilham	ilham	Benar	ilham	Benar
1717	ilham_0_8002.jpeg	ilham	ilham	Benar	ilham	Benar
1718	ilham_0_808.jpeg	ilham	ilham	Benar	ilham	Benar
1719	ilham_0_8085.jpeg	ilham	ilham	Benar	ilham	Benar
1720	ilham_0_8123.jpeg	ilham	ilham	Benar	ilham	Benar
1721	ilham_0_8161.jpeg	ilham	ilham	Benar	ilham	Benar
1722	ilham_0_8240.jpeg	ilham	ilham	Benar	ilham	Benar
1723	ilham_0_8243.jpeg	ilham	ilham	Benar	ilham	Benar
1724	ilham_0_8276.jpeg	ilham	ilham	Benar	ilham	Benar
1725	ilham_0_8363.jpeg	ilham	ilham	Benar	ilham	Benar
1726	ilham_0_8404.jpeg	ilham	ilham	Benar	ilham	Benar
1727	ilham_0_8462.jpeg	ilham	ilham	Benar	ilham	Benar
1728	ilham_0_8532.jpeg	ilham	ilham	Benar	ilham	Benar
1729	ilham_0_8594.jpeg	ilham	ilham	Benar	ilham	Benar
1730	ilham_0_8656.jpeg	ilham	ilham	Benar	ilham	Benar
1731	ilham_0_875.jpeg	ilham	ilham	Benar	ilham	Benar
1732	ilham_0_8769.jpeg	ilham	ilham	Benar	ilham	Benar
1733	ilham_0_8780.jpeg	ilham	ilham	Benar	ilham	Benar
1734	ilham_0_8793.jpeg	ilham	ilham	Benar	ilham	Benar
1735	ilham_0_9032.jpeg	ilham	ilham	Benar	ilham	Benar
1736	ilham_0_9035.jpeg	ilham	ilham	Benar	ilham	Benar
1737	ilham_0_9099.jpeg	ilham	ilham	Benar	ilham	Benar
1738	ilham_0_9121.jpeg	ilham	ilham	Benar	ilham	Benar
1739	ilham_0_9137.jpeg	ilham	ilham	Benar	ilham	Benar
1740	ilham_0_9187.jpeg	ilham	ilham	Benar	ilham	Benar
1741	ilham_0_9189.jpeg	ilham	ilham	Benar	ilham	Benar
1742	ilham_0_9275.jpeg	ilham	ilham	Benar	ilham	Benar
1743	ilham_0_9284.jpeg	ilham	ilham	Benar	ilham	Benar
1744	ilham_0_9303.jpeg	ilham	ilham	Benar	ilham	Benar
1745	ilham_0_931.jpeg	ilham	ilham	Benar	ilham	Benar
1746	ilham_0_9403.jpeg	ilham	ilham	Benar	ilham	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1747	ilham_0_9450.jpeg	ilham	ilham	Benar	ilham	Benar
1748	ilham_0_9510.jpeg	ilham	ilham	Benar	ilham	Benar
1749	ilham_0_9606.jpeg	ilham	ilham	Benar	ilham	Benar
1750	ilham_0_9665.jpeg	ilham	ilham	Benar	ilham	Benar
1751	ilham_0_9674.jpeg	ilham	ilham	Benar	ilham	Benar
1752	ilham_0_9689.jpeg	ilham	ilham	Benar	ilham	Benar
1753	ilham_0_9699.jpeg	ilham	ilham	Benar	ilham	Benar
1754	ilham_0_9719.jpeg	ilham	ilham	Benar	ilham	Benar
1755	ilham_0_973.jpeg	ilham	ilham	Benar	ilham	Benar
1756	ilham_0_9736.jpeg	ilham	ilham	Benar	ilham	Benar
1757	ilham_0_9763.jpeg	ilham	ilham	Benar	ilham	Benar
1758	ilham_0_9786.jpeg	ilham	ilham	Benar	ilham	Benar
1759	ilham_0_9837.jpeg	ilham	ilham	Benar	ilham	Benar
1760	ilham_0_9872.jpeg	ilham	ilham	Benar	ilham	Benar
1761	ilham_0_9889.jpeg	ilham	ilham	Benar	ilham	Benar
1762	ilham_0_9941.jpeg	ilham	ilham	Benar	ilham	Benar
1763	ilham_0_9970.jpeg	ilham	ilham	Benar	ilham	Benar
1764	imam_0_1012.jpeg	imam	imam	Benar	imam	Benar
1765	imam_0_1013.jpeg	imam	imam	Benar	imam	Benar
1766	imam_0_1076.jpeg	imam	imam	Benar	imam	Benar
1767	imam_0_1079.jpeg	imam	imam	Benar	imam	Benar
1768	imam_0_1097.jpeg	imam	imam	Benar	imam	Benar
1769	imam_0_1115.jpeg	imam	imam	Benar	imam	Benar
1770	imam_0_1161.jpeg	imam	imam	Benar	imam	Benar
1771	imam_0_1192.jpeg	imam	imam	Benar	imam	Benar
1772	imam_0_1204.jpeg	imam	imam	Benar	imam	Benar
1773	imam_0_1235.jpeg	imam	imam	Benar	imam	Benar
1774	imam_0_1292.jpeg	imam	imam	Benar	imam	Benar
1775	imam_0_1341.jpeg	imam	imam	Benar	imam	Benar
1776	imam_0_1343.jpeg	imam	imam	Benar	imam	Benar
1777	imam_0_1437.jpeg	imam	imam	Benar	imam	Benar
1778	imam_0_1490.jpeg	imam	imam	Benar	imam	Benar
1779	imam_0_1504.jpeg	imam	imam	Benar	imam	Benar
1780	imam_0_1573.jpeg	imam	imam	Benar	imam	Benar
1781	imam_0_1589.jpeg	imam	imam	Benar	imam	Benar
1782	imam_0_1603.jpeg	imam	imam	Benar	imam	Benar
1783	imam_0_1658.jpeg	imam	imam	Benar	imam	Benar
1784	imam_0_1664.jpeg	imam	imam	Benar	imam	Benar
1785	imam_0_1672.jpeg	imam	imam	Benar	imam	Benar
1786	imam_0_1721.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1787	imam_0_1756.jpeg	imam	imam	Benar	imam	Benar
1788	imam_0_1770.jpeg	imam	imam	Benar	imam	Benar
1789	imam_0_1826.jpeg	imam	imam	Benar	imam	Benar
1790	imam_0_1837.jpeg	imam	imam	Benar	imam	Benar
1791	imam_0_1842.jpeg	imam	imam	Benar	imam	Benar
1792	imam_0_1868.jpeg	imam	imam	Benar	imam	Benar
1793	imam_0_1874.jpeg	imam	imam	Benar	imam	Benar
1794	imam_0_1996.jpeg	imam	imam	Benar	imam	Benar
1795	imam_0_2005.jpeg	imam	imam	Benar	imam	Benar
1796	imam_0_2240.jpeg	imam	imam	Benar	imam	Benar
1797	imam_0_2257.jpeg	imam	imam	Benar	imam	Benar
1798	imam_0_2281.jpeg	imam	imam	Benar	imam	Benar
1799	imam_0_2337.jpeg	imam	imam	Benar	imam	Benar
1800	imam_0_235.jpeg	imam	imam	Benar	imam	Benar
1801	imam_0_2371.jpeg	imam	imam	Benar	imam	Benar
1802	imam_0_2375.jpeg	imam	imam	Benar	imam	Benar
1803	imam_0_2542.jpeg	imam	imam	Benar	imam	Benar
1804	imam_0_2597.jpeg	imam	imam	Benar	imam	Benar
1805	imam_0_2625.jpeg	imam	imam	Benar	imam	Benar
1806	imam_0_2701.jpeg	imam	imam	Benar	imam	Benar
1807	imam_0_2744.jpeg	imam	imam	Benar	imam	Benar
1808	imam_0_2789.jpeg	imam	imam	Benar	imam	Benar
1809	imam_0_2804.jpeg	imam	imam	Benar	imam	Benar
1810	imam_0_2825.jpeg	imam	imam	Benar	imam	Benar
1811	imam_0_2928.jpeg	imam	imam	Benar	imam	Benar
1812	imam_0_2964.jpeg	imam	imam	Benar	imam	Benar
1813	imam_0_2976.jpeg	imam	imam	Benar	imam	Benar
1814	imam_0_3177.jpeg	imam	imam	Benar	imam	Benar
1815	imam_0_3198.jpeg	imam	imam	Benar	imam	Benar
1816	imam_0_320.jpeg	imam	imam	Benar	imam	Benar
1817	imam_0_3233.jpeg	imam	imam	Benar	imam	Benar
1818	imam_0_3249.jpeg	imam	imam	Benar	imam	Benar
1819	imam_0_3268.jpeg	imam	imam	Benar	imam	Benar
1820	imam_0_330.jpeg	imam	imam	Benar	imam	Benar
1821	imam_0_3344.jpeg	imam	imam	Benar	imam	Benar
1822	imam_0_3349.jpeg	imam	imam	Benar	imam	Benar
1823	imam_0_3352.jpeg	imam	imam	Benar	imam	Benar
1824	imam_0_341.jpeg	imam	imam	Benar	imam	Benar
1825	imam_0_3449.jpeg	imam	imam	Benar	imam	Benar
1826	imam_0_3497.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1827	imam_0_351.jpeg	imam	imam	Benar	imam	Benar
1828	imam_0_1547.jpeg	imam	imam	Benar	imam	Benar
1829	imam_0_2115.jpeg	imam	imam	Benar	imam	Benar
1830	imam_0_2941.jpeg	imam	imam	Benar	imam	Benar
1831	imam_0_3520.jpeg	imam	imam	Benar	imam	Benar
1832	imam_0_4354.jpeg	imam	imam	Benar	imam	Benar
1833	imam_0_4952.jpeg	imam	imam	Benar	imam	Benar
1834	imam_0_5508.jpeg	imam	imam	Benar	imam	Benar
1835	imam_0_6171.jpeg	imam	imam	Benar	imam	Benar
1836	imam_0_7051.jpeg	imam	imam	Benar	imam	Benar
1837	imam_0_7493.jpeg	imam	imam	Benar	imam	Benar
1838	imam_0_7926.jpeg	imam	imam	Benar	imam	Benar
1839	imam_0_8422.jpeg	imam	imam	Benar	imam	Benar
1840	imam_0_9025.jpeg	imam	imam	Benar	imam	Benar
1841	imam_0_3548.jpeg	imam	imam	Benar	imam	Benar
1842	imam_0_3550.jpeg	imam	imam	Benar	imam	Benar
1843	imam_0_3682.jpeg	imam	imam	Benar	imam	Benar
1844	imam_0_3775.jpeg	imam	imam	Benar	imam	Benar
1845	imam_0_3793.jpeg	imam	imam	Benar	imam	Benar
1846	imam_0_3802.jpeg	imam	imam	Benar	imam	Benar
1847	imam_0_3838.jpeg	imam	imam	Benar	imam	Benar
1848	imam_0_402.jpeg	imam	imam	Benar	imam	Benar
1849	imam_0_4051.jpeg	imam	imam	Benar	imam	Benar
1850	imam_0_4125.jpeg	imam	imam	Benar	imam	Benar
1851	imam_0_4139.jpeg	imam	imam	Benar	imam	Benar
1852	imam_0_4227.jpeg	imam	imam	Benar	imam	Benar
1853	imam_0_4249.jpeg	imam	imam	Benar	imam	Benar
1854	imam_0_4255.jpeg	imam	imam	Benar	imam	Benar
1855	imam_0_4267.jpeg	imam	imam	Benar	imam	Benar
1856	imam_0_4327.jpeg	imam	imam	Benar	imam	Benar
1857	imam_0_4359.jpeg	imam	imam	Benar	imam	Benar
1858	imam_0_4398.jpeg	imam	imam	Benar	imam	Benar
1859	imam_0_4452.jpeg	imam	imam	Benar	imam	Benar
1860	imam_0_4481.jpeg	imam	imam	Benar	imam	Benar
1861	imam_0_45.jpeg	imam	imam	Benar	imam	Benar
1862	imam_0_4503.jpeg	imam	imam	Benar	imam	Benar
1863	imam_0_4509.jpeg	imam	imam	Benar	imam	Benar
1864	imam_0_4551.jpeg	imam	imam	Benar	imam	Benar
1865	imam_0_4557.jpeg	imam	imam	Benar	imam	Benar
1866	imam_0_4755.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1867	imam_0_4777.jpeg	imam	imam	Benar	imam	Benar
1868	imam_0_4780.jpeg	imam	imam	Benar	imam	Benar
1869	imam_0_4784.jpeg	imam	imam	Benar	imam	Benar
1870	imam_0_4817.jpeg	imam	imam	Benar	imam	Benar
1871	imam_0_4865.jpeg	imam	imam	Benar	imam	Benar
1872	imam_0_4947.jpeg	imam	imam	Benar	imam	Benar
1873	imam_0_4967.jpeg	imam	imam	Benar	imam	Benar
1874	imam_0_4995.jpeg	imam	imam	Benar	imam	Benar
1875	imam_0_5051.jpeg	imam	imam	Benar	imam	Benar
1876	imam_0_5058.jpeg	imam	imam	Benar	imam	Benar
1877	imam_0_5173.jpeg	imam	imam	Benar	imam	Benar
1878	imam_0_5248.jpeg	imam	imam	Benar	imam	Benar
1879	imam_0_5249.jpeg	imam	imam	Benar	imam	Benar
1880	imam_0_5255.jpeg	imam	imam	Benar	imam	Benar
1881	imam_0_5295.jpeg	imam	imam	Benar	imam	Benar
1882	imam_0_5352.jpeg	imam	imam	Benar	imam	Benar
1883	imam_0_5374.jpeg	imam	imam	Benar	imam	Benar
1884	imam_0_5383.jpeg	imam	imam	Benar	imam	Benar
1885	imam_0_5401.jpeg	imam	imam	Benar	imam	Benar
1886	imam_0_5415.jpeg	imam	imam	Benar	imam	Benar
1887	imam_0_5431.jpeg	imam	imam	Benar	imam	Benar
1888	imam_0_5463.jpeg	imam	imam	Benar	imam	Benar
1889	imam_0_551.jpeg	imam	imam	Benar	imam	Benar
1890	imam_0_5596.jpeg	imam	imam	Benar	imam	Benar
1891	imam_0_5601.jpeg	imam	imam	Benar	imam	Benar
1892	imam_0_5613.jpeg	imam	imam	Benar	imam	Benar
1893	imam_0_5621.jpeg	imam	imam	Benar	imam	Benar
1894	imam_0_5653.jpeg	imam	imam	Benar	imam	Benar
1895	imam_0_5674.jpeg	imam	imam	Benar	imam	Benar
1896	imam_0_5702.jpeg	imam	imam	Benar	imam	Benar
1897	imam_0_5752.jpeg	imam	imam	Benar	imam	Benar
1898	imam_0_5769.jpeg	imam	imam	Benar	imam	Benar
1899	imam_0_5911.jpeg	imam	imam	Benar	imam	Benar
1900	imam_0_592.jpeg	imam	imam	Benar	imam	Benar
1901	imam_0_596.jpeg	imam	imam	Benar	imam	Benar
1902	imam_0_5981.jpeg	imam	imam	Benar	imam	Benar
1903	imam_0_599.jpeg	imam	imam	Benar	imam	Benar
1904	imam_0_6136.jpeg	imam	imam	Benar	imam	Benar
1905	imam_0_6311.jpeg	imam	imam	Benar	imam	Benar
1906	imam_0_632.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1907	imam_0_6342.jpeg	imam	imam	Benar	imam	Benar
1908	imam_0_6352.jpeg	imam	imam	Benar	imam	Benar
1909	imam_0_6503.jpeg	imam	imam	Benar	imam	Benar
1910	imam_0_6536.jpeg	imam	imam	Benar	imam	Benar
1911	imam_0_6548.jpeg	imam	imam	Benar	imam	Benar
1912	imam_0_6577.jpeg	imam	imam	Benar	imam	Benar
1913	imam_0_6591.jpeg	imam	imam	Benar	imam	Benar
1914	imam_0_6620.jpeg	imam	imam	Benar	imam	Benar
1915	imam_0_6643.jpeg	imam	imam	Benar	imam	Benar
1916	imam_0_685.jpeg	imam	imam	Benar	imam	Benar
1917	imam_0_6898.jpeg	imam	imam	Benar	imam	Benar
1918	imam_0_6930.jpeg	imam	imam	Benar	imam	Benar
1919	imam_0_6954.jpeg	imam	imam	Benar	imam	Benar
1920	imam_0_6978.jpeg	imam	imam	Benar	imam	Benar
1921	imam_0_7086.jpeg	imam	imam	Benar	imam	Benar
1922	imam_0_7103.jpeg	imam	imam	Benar	imam	Benar
1923	imam_0_7156.jpeg	imam	imam	Benar	imam	Benar
1924	imam_0_7161.jpeg	imam	imam	Benar	imam	Benar
1925	imam_0_7209.jpeg	imam	imam	Benar	imam	Benar
1926	imam_0_7262.jpeg	imam	imam	Benar	imam	Benar
1927	imam_0_7265.jpeg	imam	imam	Benar	imam	Benar
1928	imam_0_7316.jpeg	imam	imam	Benar	imam	Benar
1929	imam_0_7350.jpeg	imam	imam	Benar	imam	Benar
1930	imam_0_7372.jpeg	imam	imam	Benar	imam	Benar
1931	imam_0_7404.jpeg	imam	imam	Benar	imam	Benar
1932	imam_0_7443.jpeg	imam	imam	Benar	imam	Benar
1933	imam_0_7454.jpeg	imam	imam	Benar	imam	Benar
1934	imam_0_746.jpeg	imam	imam	Benar	imam	Benar
1935	imam_0_7469.jpeg	imam	imam	Benar	imam	Benar
1936	imam_0_7490.jpeg	imam	imam	Benar	imam	Benar
1937	imam_0_7502.jpeg	imam	imam	Benar	imam	Benar
1938	imam_0_7529.jpeg	imam	imam	Benar	imam	Benar
1939	imam_0_7540.jpeg	imam	imam	Benar	imam	Benar
1940	imam_0_7549.jpeg	imam	imam	Benar	imam	Benar
1941	imam_0_7613.jpeg	imam	imam	Benar	imam	Benar
1942	imam_0_7634.jpeg	imam	imam	Benar	imam	Benar
1943	imam_0_7640.jpeg	imam	imam	Benar	imam	Benar
1944	imam_0_769.jpeg	imam	imam	Benar	imam	Benar
1945	imam_0_771.jpeg	imam	imam	Benar	imam	Benar
1946	imam_0_7743.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1947	imam_0_7765.jpeg	imam	imam	Benar	imam	Benar
1948	imam_0_7766.jpeg	imam	imam	Benar	imam	Benar
1949	imam_0_7791.jpeg	imam	imam	Benar	imam	Benar
1950	imam_0_7804.jpeg	imam	imam	Benar	imam	Benar
1951	imam_0_7859.jpeg	imam	imam	Benar	imam	Benar
1952	imam_0_7923.jpeg	imam	imam	Benar	imam	Benar
1953	imam_0_7945.jpeg	imam	imam	Benar	imam	Benar
1954	imam_0_8030.jpeg	imam	imam	Benar	imam	Benar
1955	imam_0_8054.jpeg	imam	imam	Benar	imam	Benar
1956	imam_0_8096.jpeg	imam	imam	Benar	imam	Benar
1957	imam_0_8107.jpeg	imam	imam	Benar	imam	Benar
1958	imam_0_8121.jpeg	imam	imam	Benar	imam	Benar
1959	imam_0_8169.jpeg	imam	imam	Benar	imam	Benar
1960	imam_0_8238.jpeg	imam	imam	Benar	imam	Benar
1961	imam_0_8305.jpeg	imam	imam	Benar	imam	Benar
1962	imam_0_8306.jpeg	imam	imam	Benar	imam	Benar
1963	imam_0_833.jpeg	imam	imam	Benar	imam	Benar
1964	imam_0_8367.jpeg	imam	imam	Benar	imam	Benar
1965	imam_0_8368.jpeg	imam	imam	Benar	imam	Benar
1966	imam_0_8386.jpeg	imam	imam	Benar	imam	Benar
1967	imam_0_8395.jpeg	imam	imam	Benar	imam	Benar
1968	imam_0_840.jpeg	imam	imam	Benar	imam	Benar
1969	imam_0_8456.jpeg	imam	imam	Benar	imam	Benar
1970	imam_0_849.jpeg	imam	imam	Benar	imam	Benar
1971	imam_0_8563.jpeg	imam	imam	Benar	imam	Benar
1972	imam_0_8583.jpeg	imam	imam	Benar	imam	Benar
1973	imam_0_8611.jpeg	imam	imam	Benar	imam	Benar
1974	imam_0_863.jpeg	imam	imam	Benar	imam	Benar
1975	imam_0_8636.jpeg	imam	imam	Benar	imam	Benar
1976	imam_0_8647.jpeg	imam	imam	Benar	imam	Benar
1977	imam_0_8652.jpeg	imam	imam	Benar	imam	Benar
1978	imam_0_8706.jpeg	imam	imam	Benar	imam	Benar
1979	imam_0_8719.jpeg	imam	imam	Benar	imam	Benar
1980	imam_0_8829.jpeg	imam	imam	Benar	imam	Benar
1981	imam_0_8885.jpeg	imam	imam	Benar	imam	Benar
1982	imam_0_8886.jpeg	imam	imam	Benar	imam	Benar
1983	imam_0_8918.jpeg	imam	imam	Benar	imam	Benar
1984	imam_0_8923.jpeg	imam	imam	Benar	imam	Benar
1985	imam_0_9076.jpeg	imam	imam	Benar	imam	Benar
1986	imam_0_9098.jpeg	imam	imam	Benar	imam	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
1987	imam_0_9105.jpeg	imam	imam	Benar	imam	Benar
1988	imam_0_9120.jpeg	imam	imam	Benar	imam	Benar
1989	imam_0_9159.jpeg	imam	imam	Benar	imam	Benar
1990	imam_0_920.jpeg	imam	imam	Benar	imam	Benar
1991	imam_0_922.jpeg	imam	imam	Benar	imam	Benar
1992	imam_0_9224.jpeg	imam	imam	Benar	imam	Benar
1993	imam_0_930.jpeg	imam	imam	Benar	imam	Benar
1994	imam_0_9308.jpeg	imam	imam	Benar	imam	Benar
1995	imam_0_9318.jpeg	imam	imam	Benar	imam	Benar
1996	imam_0_9411.jpeg	imam	imam	Benar	imam	Benar
1997	imam_0_9440.jpeg	imam	imam	Benar	imam	Benar
1998	imam_0_9495.jpeg	imam	imam	Benar	imam	Benar
1999	imam_0_955.jpeg	imam	imam	Benar	imam	Benar
2000	imam_0_9559.jpeg	imam	imam	Benar	imam	Benar
2001	imam_0_9583.jpeg	imam	imam	Benar	imam	Benar
2002	imam_0_9598.jpeg	imam	imam	Benar	imam	Benar
2003	imam_0_9643.jpeg	imam	imam	Benar	imam	Benar
2004	imam_0_9692.jpeg	imam	imam	Benar	imam	Benar
2005	imam_0_98.jpeg	imam	imam	Benar	imam	Benar
2006	imam_0_9833.jpeg	imam	imam	Benar	imam	Benar
2007	imam_0_9852.jpeg	imam	imam	Benar	imam	Benar
2008	imam_0_9860.jpeg	imam	imam	Benar	imam	Benar
2009	imam_0_9879.jpeg	imam	imam	Benar	imam	Benar
2010	imam_0_9888.jpeg	imam	imam	Benar	imam	Benar
2011	imam_0_9889.jpeg	imam	imam	Benar	imam	Benar
2012	imam_0_9941.jpeg	imam	imam	Benar	imam	Benar
2013	imam_0_9943.jpeg	imam	imam	Benar	imam	Benar
2014	imam_0_9965.jpeg	imam	imam	Benar	imam	Benar
2015	imam_0_9979.jpeg	imam	imam	Benar	imam	Benar
2016	puji_0_1003.jpeg	puji	puji	Benar	puji	Benar
2017	puji_0_1006.jpeg	puji	puji	Benar	puji	Benar
2018	puji_0_1012.jpeg	puji	puji	Benar	puji	Benar
2019	puji_0_1081.jpeg	puji	puji	Benar	puji	Benar
2020	puji_0_1090.jpeg	puji	puji	Benar	puji	Benar
2021	puji_0_1112.jpeg	puji	puji	Benar	puji	Benar
2022	puji_0_1120.jpeg	puji	puji	Benar	puji	Benar
2023	puji_0_1122.jpeg	puji	puji	Benar	puji	Benar
2024	puji_0_1205.jpeg	puji	puji	Benar	puji	Benar
2025	puji_0_1223.jpeg	puji	puji	Benar	puji	Benar
2026	puji_0_1235.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2027	puji_0_1240.jpeg	puji	puji	Benar	puji	Benar
2028	puji_0_1268.jpeg	puji	puji	Benar	puji	Benar
2029	puji_0_1301.jpeg	puji	puji	Benar	puji	Benar
2030	puji_0_1303.jpeg	puji	puji	Benar	puji	Benar
2031	puji_0_1320.jpeg	puji	puji	Benar	puji	Benar
2032	puji_0_1354.jpeg	puji	puji	Benar	puji	Benar
2033	puji_0_1378.jpeg	puji	puji	Benar	puji	Benar
2034	puji_0_144.jpeg	puji	puji	Benar	puji	Benar
2035	puji_0_1572.jpeg	puji	puji	Benar	puji	Benar
2036	puji_0_158.jpeg	puji	puji	Benar	puji	Benar
2037	puji_0_1607.jpeg	puji	puji	Benar	puji	Benar
2038	puji_0_1704.jpeg	puji	puji	Benar	puji	Benar
2039	puji_0_1719.jpeg	puji	puji	Benar	puji	Benar
2040	puji_0_1727.jpeg	puji	puji	Benar	puji	Benar
2041	puji_0_1753.jpeg	puji	puji	Benar	puji	Benar
2042	puji_0_185.jpeg	puji	puji	Benar	puji	Benar
2043	puji_0_1972.jpeg	puji	puji	Benar	puji	Benar
2044	puji_0_208.jpeg	puji	puji	Benar	puji	Benar
2045	puji_0_21.jpeg	puji	puji	Benar	puji	Benar
2046	puji_0_2114.jpeg	puji	puji	Benar	puji	Benar
2047	puji_0_2199.jpeg	puji	puji	Benar	puji	Benar
2048	puji_0_2256.jpeg	puji	puji	Benar	puji	Benar
2049	puji_0_2272.jpeg	puji	puji	Benar	puji	Benar
2050	puji_0_2299.jpeg	puji	puji	Benar	puji	Benar
2051	puji_0_2394.jpeg	puji	puji	Benar	puji	Benar
2052	puji_0_2460.jpeg	puji	puji	Benar	puji	Benar
2053	puji_0_2507.jpeg	puji	puji	Benar	puji	Benar
2054	puji_0_2520.jpeg	puji	puji	Benar	puji	Benar
2055	puji_0_2572.jpeg	puji	puji	Benar	puji	Benar
2056	puji_0_2579.jpeg	puji	puji	Benar	puji	Benar
2057	puji_0_2642.jpeg	puji	puji	Benar	puji	Benar
2058	puji_0_2652.jpeg	puji	puji	Benar	puji	Benar
2059	puji_0_2677.jpeg	puji	puji	Benar	puji	Benar
2060	puji_0_2689.jpeg	puji	puji	Benar	puji	Benar
2061	puji_0_2711.jpeg	puji	puji	Benar	puji	Benar
2062	puji_0_2731.jpeg	puji	puji	Benar	puji	Benar
2063	puji_0_2796.jpeg	puji	puji	Benar	puji	Benar
2064	puji_0_2856.jpeg	puji	puji	Benar	puji	Benar
2065	puji_0_2860.jpeg	puji	puji	Benar	puji	Benar
2066	puji_0_2901.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2067	puji_0_2943.jpeg	puji	puji	Benar	puji	Benar
2068	puji_0_2952.jpeg	puji	puji	Benar	puji	Benar
2069	puji_0_2976.jpeg	puji	puji	Benar	puji	Benar
2070	puji_0_3018.jpeg	puji	puji	Benar	puji	Benar
2071	puji_0_304.jpeg	puji	puji	Benar	puji	Benar
2072	puji_0_3150.jpeg	puji	puji	Benar	puji	Benar
2073	puji_0_3167.jpeg	puji	puji	Benar	puji	Benar
2074	puji_0_3240.jpeg	puji	puji	Benar	puji	Benar
2075	puji_0_331.jpeg	puji	puji	Benar	puji	Benar
2076	puji_0_3371.jpeg	puji	puji	Benar	puji	Benar
2077	puji_0_3405.jpeg	puji	puji	Benar	puji	Benar
2078	puji_0_3451.jpeg	puji	puji	Benar	puji	Benar
2079	puji_0_3475.jpeg	puji	puji	Benar	puji	Benar
2080	puji_0_1333.jpeg	puji	puji	Benar	puji	Benar
2081	puji_0_2248.jpeg	puji	puji	Benar	puji	Benar
2082	puji_0_2829.jpeg	puji	puji	Benar	puji	Benar
2083	puji_0_3500.jpeg	puji	puji	Benar	puji	Benar
2084	puji_0_4199.jpeg	puji	puji	Benar	puji	Benar
2085	puji_0_489.jpeg	puji	puji	Benar	puji	Benar
2086	puji_0_553.jpeg	puji	puji	Benar	puji	Benar
2087	puji_0_6042.jpeg	puji	puji	Benar	puji	Benar
2088	puji_0_6477.jpeg	puji	puji	Benar	puji	Benar
2089	puji_0_6962.jpeg	puji	puji	Benar	puji	Benar
2090	puji_0_7768.jpeg	puji	puji	Benar	puji	Benar
2091	puji_0_8094.jpeg	puji	puji	Benar	puji	Benar
2092	puji_0_8500.jpeg	puji	puji	Benar	puji	Benar
2093	puji_0_3544.jpeg	puji	puji	Benar	puji	Benar
2094	puji_0_3551.jpeg	puji	puji	Benar	puji	Benar
2095	puji_0_3577.jpeg	puji	puji	Benar	puji	Benar
2096	puji_0_3641.jpeg	puji	puji	Benar	puji	Benar
2097	puji_0_3685.jpeg	puji	puji	Benar	puji	Benar
2098	puji_0_3703.jpeg	puji	puji	Benar	puji	Benar
2099	puji_0_3744.jpeg	puji	puji	Benar	puji	Benar
2100	puji_0_3768.jpeg	puji	puji	Benar	puji	Benar
2101	puji_0_3792.jpeg	puji	puji	Benar	puji	Benar
2102	puji_0_3795.jpeg	puji	puji	Benar	puji	Benar
2103	puji_0_384.jpeg	puji	puji	Benar	puji	Benar
2104	puji_0_3897.jpeg	puji	puji	Benar	puji	Benar
2105	puji_0_3979.jpeg	puji	puji	Benar	puji	Benar
2106	puji_0_4141.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2107	puji_0_4144.jpeg	puji	puji	Benar	puji	Benar
2108	puji_0_4180.jpeg	puji	puji	Benar	puji	Benar
2109	puji_0_4258.jpeg	puji	puji	Benar	puji	Benar
2110	puji_0_4271.jpeg	puji	puji	Benar	puji	Benar
2111	puji_0_4282.jpeg	puji	puji	Benar	puji	Benar
2112	puji_0_433.jpeg	puji	puji	Benar	puji	Benar
2113	puji_0_4376.jpeg	puji	puji	Benar	puji	Benar
2114	puji_0_4382.jpeg	puji	puji	Benar	puji	Benar
2115	puji_0_4393.jpeg	puji	puji	Benar	puji	Benar
2116	puji_0_4399.jpeg	puji	puji	Benar	puji	Benar
2117	puji_0_4405.jpeg	puji	puji	Benar	puji	Benar
2118	puji_0_4427.jpeg	puji	puji	Benar	puji	Benar
2119	puji_0_4438.jpeg	puji	puji	Benar	puji	Benar
2120	puji_0_4495.jpeg	puji	puji	Benar	puji	Benar
2121	puji_0_4582.jpeg	puji	puji	Benar	puji	Benar
2122	puji_0_4679.jpeg	puji	puji	Benar	puji	Benar
2123	puji_0_4695.jpeg	puji	puji	Benar	puji	Benar
2124	puji_0_4786.jpeg	puji	puji	Benar	puji	Benar
2125	puji_0_4912.jpeg	puji	puji	Benar	puji	Benar
2126	puji_0_4957.jpeg	puji	puji	Benar	puji	Benar
2127	puji_0_501.jpeg	puji	puji	Benar	puji	Benar
2128	puji_0_5029.jpeg	puji	puji	Benar	puji	Benar
2129	puji_0_512.jpeg	puji	puji	Benar	puji	Benar
2130	puji_0_5146.jpeg	puji	puji	Benar	puji	Benar
2131	puji_0_5173.jpeg	puji	puji	Benar	puji	Benar
2132	puji_0_5191.jpeg	puji	puji	Benar	puji	Benar
2133	puji_0_522.jpeg	puji	puji	Benar	puji	Benar
2134	puji_0_5249.jpeg	puji	puji	Benar	puji	Benar
2135	puji_0_5368.jpeg	puji	puji	Benar	puji	Benar
2136	puji_0_5370.jpeg	puji	puji	Benar	puji	Benar
2137	puji_0_538.jpeg	puji	puji	Benar	puji	Benar
2138	puji_0_5452.jpeg	puji	puji	Benar	puji	Benar
2139	puji_0_5494.jpeg	puji	puji	Benar	puji	Benar
2140	puji_0_5520.jpeg	puji	puji	Benar	puji	Benar
2141	puji_0_5539.jpeg	puji	puji	Benar	puji	Benar
2142	puji_0_5555.jpeg	puji	puji	Benar	puji	Benar
2143	puji_0_5606.jpeg	puji	puji	Benar	puji	Benar
2144	puji_0_5642.jpeg	puji	puji	Benar	puji	Benar
2145	puji_0_5676.jpeg	puji	puji	Benar	puji	Benar
2146	puji_0_5684.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2147	puji_0_5698.jpeg	puji	puji	Benar	puji	Benar
2148	puji_0_5703.jpeg	puji	puji	Benar	puji	Benar
2149	puji_0_5764.jpeg	puji	puji	Benar	puji	Benar
2150	puji_0_5781.jpeg	puji	puji	Benar	puji	Benar
2151	puji_0_5812.jpeg	puji	puji	Benar	puji	Benar
2152	puji_0_5870.jpeg	puji	puji	Benar	puji	Benar
2153	puji_0_5896.jpeg	puji	puji	Benar	puji	Benar
2154	puji_0_5926.jpeg	puji	puji	Benar	puji	Benar
2155	puji_0_5937.jpeg	puji	puji	Benar	puji	Benar
2156	puji_0_5980.jpeg	puji	puji	Benar	puji	Benar
2157	puji_0_6057.jpeg	puji	puji	Benar	puji	Benar
2158	puji_0_6078.jpeg	puji	puji	Benar	puji	Benar
2159	puji_0_6085.jpeg	puji	puji	Benar	puji	Benar
2160	puji_0_6099.jpeg	puji	puji	Benar	puji	Benar
2161	puji_0_6103.jpeg	puji	puji	Benar	puji	Benar
2162	puji_0_612.jpeg	puji	puji	Benar	puji	Benar
2163	puji_0_6120.jpeg	puji	puji	Benar	puji	Benar
2164	puji_0_6130.jpeg	puji	puji	Benar	puji	Benar
2165	puji_0_6226.jpeg	puji	puji	Benar	puji	Benar
2166	puji_0_6238.jpeg	puji	puji	Benar	puji	Benar
2167	puji_0_6277.jpeg	puji	puji	Benar	puji	Benar
2168	puji_0_6298.jpeg	puji	puji	Benar	puji	Benar
2169	puji_0_6435.jpeg	puji	puji	Benar	puji	Benar
2170	puji_0_6451.jpeg	puji	puji	Benar	puji	Benar
2171	puji_0_6468.jpeg	puji	puji	Benar	puji	Benar
2172	puji_0_6472.jpeg	puji	puji	Benar	puji	Benar
2173	puji_0_654.jpeg	puji	puji	Benar	puji	Benar
2174	puji_0_6545.jpeg	puji	puji	Benar	puji	Benar
2175	puji_0_6562.jpeg	puji	puji	Benar	puji	Benar
2176	puji_0_6605.jpeg	puji	puji	Benar	puji	Benar
2177	puji_0_6619.jpeg	puji	puji	Benar	puji	Benar
2178	puji_0_6625.jpeg	puji	puji	Benar	puji	Benar
2179	puji_0_6634.jpeg	puji	puji	Benar	puji	Benar
2180	puji_0_6650.jpeg	puji	puji	Benar	puji	Benar
2181	puji_0_6715.jpeg	puji	puji	Benar	puji	Benar
2182	puji_0_6793.jpeg	puji	puji	Benar	puji	Benar
2183	puji_0_684.jpeg	puji	puji	Benar	puji	Benar
2184	puji_0_6869.jpeg	puji	puji	Benar	puji	Benar
2185	puji_0_6870.jpeg	puji	puji	Benar	puji	Benar
2186	puji_0_6901.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2187	puji_0_6928.jpeg	puji	puji	Benar	puji	Benar
2188	puji_0_6931.jpeg	puji	puji	Benar	puji	Benar
2189	puji_0_6983.jpeg	puji	puji	Benar	puji	Benar
2190	puji_0_7032.jpeg	puji	puji	Benar	puji	Benar
2191	puji_0_707.jpeg	puji	puji	Benar	puji	Benar
2192	puji_0_7077.jpeg	puji	puji	Benar	puji	Benar
2193	puji_0_7102.jpeg	puji	puji	Benar	puji	Benar
2194	puji_0_7108.jpeg	puji	puji	Benar	puji	Benar
2195	puji_0_7109.jpeg	puji	puji	Benar	puji	Benar
2196	puji_0_7256.jpeg	puji	puji	Benar	puji	Benar
2197	puji_0_7286.jpeg	puji	puji	Benar	puji	Benar
2198	puji_0_7365.jpeg	puji	puji	Benar	puji	Benar
2199	puji_0_7392.jpeg	puji	puji	Benar	puji	Benar
2200	puji_0_7467.jpeg	puji	puji	Benar	puji	Benar
2201	puji_0_7483.jpeg	puji	puji	Benar	puji	Benar
2202	puji_0_7501.jpeg	puji	puji	Benar	puji	Benar
2203	puji_0_7532.jpeg	puji	puji	Benar	puji	Benar
2204	puji_0_773.jpeg	puji	puji	Benar	puji	Benar
2205	puji_0_7823.jpeg	puji	puji	Benar	puji	Benar
2206	puji_0_7831.jpeg	puji	puji	Benar	puji	Benar
2207	puji_0_7845.jpeg	puji	puji	Benar	puji	Benar
2208	puji_0_7882.jpeg	puji	puji	Benar	puji	Benar
2209	puji_0_7908.jpeg	puji	puji	Benar	puji	Benar
2210	puji_0_7912.jpeg	puji	puji	Benar	puji	Benar
2211	puji_0_7918.jpeg	puji	puji	Benar	puji	Benar
2212	puji_0_7926.jpeg	puji	puji	Benar	puji	Benar
2213	puji_0_7947.jpeg	puji	puji	Benar	puji	Benar
2214	puji_0_7960.jpeg	puji	puji	Benar	puji	Benar
2215	puji_0_7976.jpeg	puji	puji	Benar	puji	Benar
2216	puji_0_8009.jpeg	puji	puji	Benar	puji	Benar
2217	puji_0_8035.jpeg	puji	puji	Benar	puji	Benar
2218	puji_0_8056.jpeg	puji	puji	Benar	puji	Benar
2219	puji_0_8063.jpeg	puji	puji	Benar	puji	Benar
2220	puji_0_8092.jpeg	puji	puji	Benar	puji	Benar
2221	puji_0_8109.jpeg	puji	puji	Benar	puji	Benar
2222	puji_0_8124.jpeg	puji	puji	Benar	puji	Benar
2223	puji_0_8131.jpeg	puji	puji	Benar	puji	Benar
2224	puji_0_8156.jpeg	puji	puji	Benar	puji	Benar
2225	puji_0_8193.jpeg	puji	puji	Benar	puji	Benar
2226	puji_0_8205.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2227	puji_0_827.jpeg	puji	puji	Benar	puji	Benar
2228	puji_0_8357.jpeg	puji	puji	Benar	puji	Benar
2229	puji_0_8369.jpeg	puji	puji	Benar	puji	Benar
2230	puji_0_8377.jpeg	puji	puji	Benar	puji	Benar
2231	puji_0_8395.jpeg	puji	puji	Benar	puji	Benar
2232	puji_0_8407.jpeg	puji	puji	Benar	puji	Benar
2233	puji_0_8415.jpeg	puji	puji	Benar	puji	Benar
2234	puji_0_8435.jpeg	puji	puji	Benar	puji	Benar
2235	puji_0_8446.jpeg	puji	puji	Benar	puji	Benar
2236	puji_0_8485.jpeg	puji	puji	Benar	puji	Benar
2237	puji_0_8506.jpeg	puji	puji	Benar	puji	Benar
2238	puji_0_8524.jpeg	puji	puji	Benar	puji	Benar
2239	puji_0_8633.jpeg	puji	puji	Benar	puji	Benar
2240	puji_0_8802.jpeg	puji	puji	Benar	puji	Benar
2241	puji_0_8872.jpeg	puji	puji	Benar	puji	Benar
2242	puji_0_8957.jpeg	puji	puji	Benar	puji	Benar
2243	puji_0_896.jpeg	puji	puji	Benar	puji	Benar
2244	puji_0_9011.jpeg	puji	puji	Benar	puji	Benar
2245	puji_0_9086.jpeg	puji	puji	Benar	puji	Benar
2246	puji_0_9105.jpeg	puji	puji	Benar	puji	Benar
2247	puji_0_912.jpeg	puji	puji	Benar	puji	Benar
2248	puji_0_9132.jpeg	puji	puji	Benar	puji	Benar
2249	puji_0_9152.jpeg	puji	puji	Benar	puji	Benar
2250	puji_0_9171.jpeg	puji	puji	Benar	puji	Benar
2251	puji_0_9338.jpeg	puji	puji	Benar	puji	Benar
2252	puji_0_9411.jpeg	puji	puji	Benar	puji	Benar
2253	puji_0_9428.jpeg	puji	puji	Benar	puji	Benar
2254	puji_0_9455.jpeg	puji	puji	Benar	puji	Benar
2255	puji_0_9480.jpeg	puji	puji	Benar	puji	Benar
2256	puji_0_9497.jpeg	puji	puji	Benar	puji	Benar
2257	puji_0_9520.jpeg	puji	puji	Benar	puji	Benar
2258	puji_0_9556.jpeg	puji	puji	Benar	puji	Benar
2259	puji_0_9560.jpeg	puji	puji	Benar	puji	Benar
2260	puji_0_9583.jpeg	puji	puji	Benar	puji	Benar
2261	puji_0_9690.jpeg	puji	puji	Benar	puji	Benar
2262	puji_0_97.jpeg	puji	puji	Benar	puji	Benar
2263	puji_0_9745.jpeg	puji	puji	Benar	puji	Benar
2264	puji_0_9764.jpeg	puji	puji	Benar	puji	Benar
2265	puji_0_9847.jpeg	puji	puji	Benar	puji	Benar
2266	puji_0_9848.jpeg	puji	puji	Benar	puji	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2267	puji_0_9904.jpeg	puji	puji	Benar	puji	Benar
2268	puji_0_9948.jpeg	puji	puji	Benar	puji	Benar
2269	puji_0_9989.jpeg	puji	puji	Benar	puji	Benar
2270	rizka_0_1015.jpeg	rizka	rizka	Benar	rizka	Benar
2271	rizka_0_1055.jpeg	rizka	rizka	Benar	rizka	Benar
2272	rizka_0_1074.jpeg	rizka	rizka	Benar	rizka	Benar
2273	rizka_0_1077.jpeg	rizka	rizka	Benar	rizka	Benar
2274	rizka_0_1101.jpeg	rizka	rizka	Benar	rizka	Benar
2275	rizka_0_1124.jpeg	rizka	rizka	Benar	rizka	Benar
2276	rizka_0_1212.jpeg	rizka	rizka	Benar	rizka	Benar
2277	rizka_0_1257.jpeg	rizka	rizka	Benar	rizka	Benar
2278	rizka_0_1347.jpeg	rizka	rizka	Benar	rizka	Benar
2279	rizka_0_1444.jpeg	rizka	rizka	Benar	rizka	Benar
2280	rizka_0_1464.jpeg	rizka	rizka	Benar	rizka	Benar
2281	rizka_0_1562.jpeg	rizka	rizka	Benar	rizka	Benar
2282	rizka_0_16.jpeg	rizka	rizka	Benar	rizka	Benar
2283	rizka_0_1659.jpeg	rizka	dinda	Salah	rizka	Benar
2284	rizka_0_1684.jpeg	rizka	rizka	Benar	rizka	Benar
2285	rizka_0_1708.jpeg	rizka	rizka	Benar	rizka	Benar
2286	rizka_0_1777.jpeg	rizka	rizka	Benar	rizka	Benar
2287	rizka_0_179.jpeg	rizka	rizka	Benar	rizka	Benar
2288	rizka_0_1818.jpeg	rizka	rizka	Benar	rizka	Benar
2289	rizka_0_1825.jpeg	rizka	rizka	Benar	rizka	Benar
2290	rizka_0_1933.jpeg	rizka	rizka	Benar	rizka	Benar
2291	rizka_0_1961.jpeg	rizka	rizka	Benar	rizka	Benar
2292	rizka_0_2132.jpeg	rizka	rizka	Benar	rizka	Benar
2293	rizka_0_219.jpeg	rizka	rizka	Benar	rizka	Benar
2294	rizka_0_2194.jpeg	rizka	rizka	Benar	rizka	Benar
2295	rizka_0_2227.jpeg	rizka	rizka	Benar	rizka	Benar
2296	rizka_0_2250.jpeg	rizka	rizka	Benar	rizka	Benar
2297	rizka_0_2262.jpeg	rizka	rizka	Benar	rizka	Benar
2298	rizka_0_233.jpeg	rizka	rizka	Benar	rizka	Benar
2299	rizka_0_2335.jpeg	rizka	rizka	Benar	rizka	Benar
2300	rizka_0_2343.jpeg	rizka	rizka	Benar	rizka	Benar
2301	rizka_0_2350.jpeg	rizka	rizka	Benar	rizka	Benar
2302	rizka_0_2448.jpeg	rizka	rizka	Benar	rizka	Benar
2303	rizka_0_2464.jpeg	rizka	rizka	Benar	rizka	Benar
2304	rizka_0_2468.jpeg	rizka	rizka	Benar	rizka	Benar
2305	rizka_0_2496.jpeg	rizka	rizka	Benar	rizka	Benar
2306	rizka_0_2522.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2307	rizka_0_2623.jpeg	rizka	rizka	Benar	rizka	Benar
2308	rizka_0_2636.jpeg	rizka	rizka	Benar	rizka	Benar
2309	rizka_0_2683.jpeg	rizka	rizka	Benar	rizka	Benar
2310	rizka_0_2711.jpeg	rizka	rizka	Benar	rizka	Benar
2311	rizka_0_273.jpeg	rizka	rizka	Benar	rizka	Benar
2312	rizka_0_2741.jpeg	rizka	rizka	Benar	rizka	Benar
2313	rizka_0_2866.jpeg	rizka	rizka	Benar	rizka	Benar
2314	rizka_0_288.jpeg	rizka	rizka	Benar	rizka	Benar
2315	rizka_0_3002.jpeg	rizka	rizka	Benar	rizka	Benar
2316	rizka_0_301.jpeg	rizka	rizka	Benar	rizka	Benar
2317	rizka_0_3018.jpeg	rizka	rizka	Benar	rizka	Benar
2318	rizka_0_3112.jpeg	rizka	rizka	Benar	rizka	Benar
2319	rizka_0_3134.jpeg	rizka	rizka	Benar	rizka	Benar
2320	rizka_0_3143.jpeg	rizka	rizka	Benar	rizka	Benar
2321	rizka_0_3157.jpeg	rizka	rizka	Benar	rizka	Benar
2322	rizka_0_3160.jpeg	rizka	rizka	Benar	rizka	Benar
2323	rizka_0_3175.jpeg	rizka	rizka	Benar	rizka	Benar
2324	rizka_0_3221.jpeg	rizka	rizka	Benar	rizka	Benar
2325	rizka_0_3231.jpeg	rizka	rizka	Benar	rizka	Benar
2326	rizka_0_3333.jpeg	rizka	rizka	Benar	rizka	Benar
2327	rizka_0_3373.jpeg	rizka	rizka	Benar	rizka	Benar
2328	rizka_0_3384.jpeg	rizka	rizka	Benar	rizka	Benar
2329	rizka_0_3390.jpeg	rizka	rizka	Benar	rizka	Benar
2330	rizka_0_3399.jpeg	rizka	rizka	Benar	rizka	Benar
2331	rizka_0_3423.jpeg	rizka	rizka	Benar	rizka	Benar
2332	rizka_0_3469.jpeg	rizka	rizka	Benar	rizka	Benar
2333	rizka_0_349.jpeg	rizka	rizka	Benar	rizka	Benar
2334	rizka_0_1712.jpeg	rizka	rizka	Benar	rizka	Benar
2335	rizka_0_2421.jpeg	rizka	rizka	Benar	rizka	Benar
2336	rizka_0_3110.jpeg	rizka	rizka	Benar	rizka	Benar
2337	rizka_0_3522.jpeg	rizka	rizka	Benar	rizka	Benar
2338	rizka_0_4081.jpeg	rizka	rizka	Benar	rizka	Benar
2339	rizka_0_4721.jpeg	rizka	rizka	Benar	rizka	Benar
2340	rizka_0_5144.jpeg	rizka	rizka	Benar	rizka	Benar
2341	rizka_0_5675.jpeg	rizka	rizka	Benar	rizka	Benar
2342	rizka_0_6672.jpeg	rizka	rizka	Benar	rizka	Benar
2343	rizka_0_7160.jpeg	rizka	rizka	Benar	rizka	Benar
2344	rizka_0_7711.jpeg	rizka	rizka	Benar	rizka	Benar
2345	rizka_0_8256.jpeg	rizka	rizka	Benar	rizka	Benar
2346	rizka_0_8785.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2347	rizka_0_357.jpeg	rizka	rizka	Benar	rizka	Benar
2348	rizka_0_3574.jpeg	rizka	rizka	Benar	rizka	Benar
2349	rizka_0_3631.jpeg	rizka	rizka	Benar	rizka	Benar
2350	rizka_0_3707.jpeg	rizka	rizka	Benar	rizka	Benar
2351	rizka_0_377.jpeg	rizka	rizka	Benar	rizka	Benar
2352	rizka_0_3780.jpeg	rizka	rizka	Benar	rizka	Benar
2353	rizka_0_3864.jpeg	rizka	rizka	Benar	rizka	Benar
2354	rizka_0_3883.jpeg	rizka	rizka	Benar	rizka	Benar
2355	rizka_0_3914.jpeg	rizka	rizka	Benar	rizka	Benar
2356	rizka_0_3923.jpeg	rizka	rizka	Benar	rizka	Benar
2357	rizka_0_3932.jpeg	rizka	rizka	Benar	rizka	Benar
2358	rizka_0_3956.jpeg	rizka	rizka	Benar	rizka	Benar
2359	rizka_0_3959.jpeg	rizka	rizka	Benar	rizka	Benar
2360	rizka_0_3960.jpeg	rizka	rizka	Benar	rizka	Benar
2361	rizka_0_3977.jpeg	rizka	rizka	Benar	rizka	Benar
2362	rizka_0_4014.jpeg	rizka	rizka	Benar	rizka	Benar
2363	rizka_0_4110.jpeg	rizka	rizka	Benar	rizka	Benar
2364	rizka_0_415.jpeg	rizka	rizka	Benar	rizka	Benar
2365	rizka_0_4168.jpeg	rizka	rizka	Benar	rizka	Benar
2366	rizka_0_4223.jpeg	rizka	rizka	Benar	rizka	Benar
2367	rizka_0_4227.jpeg	rizka	rizka	Benar	rizka	Benar
2368	rizka_0_4263.jpeg	rizka	rizka	Benar	rizka	Benar
2369	rizka_0_4428.jpeg	rizka	rizka	Benar	rizka	Benar
2370	rizka_0_4453.jpeg	rizka	rizka	Benar	rizka	Benar
2371	rizka_0_4491.jpeg	rizka	rizka	Benar	rizka	Benar
2372	rizka_0_4551.jpeg	rizka	rizka	Benar	rizka	Benar
2373	rizka_0_4558.jpeg	rizka	rizka	Benar	rizka	Benar
2374	rizka_0_4591.jpeg	rizka	rizka	Benar	rizka	Benar
2375	rizka_0_4672.jpeg	rizka	rizka	Benar	rizka	Benar
2376	rizka_0_4691.jpeg	rizka	rizka	Benar	rizka	Benar
2377	rizka_0_4714.jpeg	rizka	rizka	Benar	rizka	Benar
2378	rizka_0_4720.jpeg	rizka	rizka	Benar	rizka	Benar
2379	rizka_0_4727.jpeg	rizka	rizka	Benar	rizka	Benar
2380	rizka_0_4729.jpeg	rizka	rizka	Benar	rizka	Benar
2381	rizka_0_4730.jpeg	rizka	rizka	Benar	rizka	Benar
2382	rizka_0_4757.jpeg	rizka	rizka	Benar	rizka	Benar
2383	rizka_0_4765.jpeg	rizka	rizka	Benar	rizka	Benar
2384	rizka_0_4788.jpeg	rizka	rizka	Benar	rizka	Benar
2385	rizka_0_479.jpeg	rizka	rizka	Benar	rizka	Benar
2386	rizka_0_4797.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2387	rizka_0_4805.jpeg	rizka	rizka	Benar	rizka	Benar
2388	rizka_0_4831.jpeg	rizka	rizka	Benar	rizka	Benar
2389	rizka_0_4859.jpeg	rizka	rizka	Benar	rizka	Benar
2390	rizka_0_4940.jpeg	rizka	rizka	Benar	rizka	Benar
2391	rizka_0_4981.jpeg	rizka	rizka	Benar	rizka	Benar
2392	rizka_0_5003.jpeg	rizka	rizka	Benar	rizka	Benar
2393	rizka_0_5134.jpeg	rizka	rizka	Benar	rizka	Benar
2394	rizka_0_5141.jpeg	rizka	rizka	Benar	rizka	Benar
2395	rizka_0_5152.jpeg	rizka	rizka	Benar	rizka	Benar
2396	rizka_0_5159.jpeg	rizka	rizka	Benar	rizka	Benar
2397	rizka_0_5162.jpeg	rizka	rizka	Benar	rizka	Benar
2398	rizka_0_5170.jpeg	rizka	rizka	Benar	rizka	Benar
2399	rizka_0_5221.jpeg	rizka	rizka	Benar	rizka	Benar
2400	rizka_0_5226.jpeg	rizka	rizka	Benar	rizka	Benar
2401	rizka_0_5281.jpeg	rizka	rizka	Benar	rizka	Benar
2402	rizka_0_5297.jpeg	rizka	rizka	Benar	rizka	Benar
2403	rizka_0_5424.jpeg	rizka	rizka	Benar	rizka	Benar
2404	rizka_0_543.jpeg	rizka	rizka	Benar	rizka	Benar
2405	rizka_0_5522.jpeg	rizka	rizka	Benar	rizka	Benar
2406	rizka_0_5534.jpeg	rizka	rizka	Benar	rizka	Benar
2407	rizka_0_5607.jpeg	rizka	rizka	Benar	rizka	Benar
2408	rizka_0_565.jpeg	rizka	rizka	Benar	rizka	Benar
2409	rizka_0_5660.jpeg	rizka	rizka	Benar	rizka	Benar
2410	rizka_0_5667.jpeg	rizka	rizka	Benar	rizka	Benar
2411	rizka_0_5779.jpeg	rizka	rizka	Benar	rizka	Benar
2412	rizka_0_5814.jpeg	rizka	rizka	Benar	rizka	Benar
2413	rizka_0_5835.jpeg	rizka	rizka	Benar	rizka	Benar
2414	rizka_0_5843.jpeg	rizka	rizka	Benar	rizka	Benar
2415	rizka_0_5851.jpeg	rizka	rizka	Benar	rizka	Benar
2416	rizka_0_5873.jpeg	rizka	rizka	Benar	rizka	Benar
2417	rizka_0_5919.jpeg	rizka	rizka	Benar	rizka	Benar
2418	rizka_0_6027.jpeg	rizka	rizka	Benar	rizka	Benar
2419	rizka_0_6031.jpeg	rizka	rizka	Benar	rizka	Benar
2420	rizka_0_6123.jpeg	rizka	rizka	Benar	rizka	Benar
2421	rizka_0_6248.jpeg	rizka	rizka	Benar	rizka	Benar
2422	rizka_0_637.jpeg	rizka	rizka	Benar	rizka	Benar
2423	rizka_0_6475.jpeg	rizka	rizka	Benar	rizka	Benar
2424	rizka_0_6548.jpeg	rizka	rizka	Benar	rizka	Benar
2425	rizka_0_6593.jpeg	rizka	rizka	Benar	rizka	Benar
2426	rizka_0_6670.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2427	rizka_0_6702.jpeg	rizka	rizka	Benar	rizka	Benar
2428	rizka_0_6731.jpeg	rizka	rizka	Benar	rizka	Benar
2429	rizka_0_6782.jpeg	rizka	rizka	Benar	rizka	Benar
2430	rizka_0_6792.jpeg	rizka	rizka	Benar	rizka	Benar
2431	rizka_0_6798.jpeg	rizka	rizka	Benar	rizka	Benar
2432	rizka_0_685.jpeg	rizka	rizka	Benar	rizka	Benar
2433	rizka_0_6877.jpeg	rizka	rizka	Benar	rizka	Benar
2434	rizka_0_6906.jpeg	rizka	rizka	Benar	rizka	Benar
2435	rizka_0_6918.jpeg	rizka	rizka	Benar	rizka	Benar
2436	rizka_0_6968.jpeg	rizka	rizka	Benar	rizka	Benar
2437	rizka_0_7043.jpeg	rizka	rizka	Benar	rizka	Benar
2438	rizka_0_7047.jpeg	rizka	rizka	Benar	rizka	Benar
2439	rizka_0_7052.jpeg	rizka	rizka	Benar	rizka	Benar
2440	rizka_0_7095.jpeg	rizka	rizka	Benar	rizka	Benar
2441	rizka_0_7150.jpeg	rizka	rizka	Benar	rizka	Benar
2442	rizka_0_7153.jpeg	rizka	rizka	Benar	rizka	Benar
2443	rizka_0_7162.jpeg	rizka	rizka	Benar	rizka	Benar
2444	rizka_0_718.jpeg	rizka	rizka	Benar	rizka	Benar
2445	rizka_0_7222.jpeg	rizka	rizka	Benar	rizka	Benar
2446	rizka_0_7226.jpeg	rizka	rizka	Benar	rizka	Benar
2447	rizka_0_7294.jpeg	rizka	rizka	Benar	rizka	Benar
2448	rizka_0_738.jpeg	rizka	rizka	Benar	rizka	Benar
2449	rizka_0_7382.jpeg	rizka	rizka	Benar	rizka	Benar
2450	rizka_0_7410.jpeg	rizka	rizka	Benar	rizka	Benar
2451	rizka_0_7411.jpeg	rizka	rizka	Benar	rizka	Benar
2452	rizka_0_7417.jpeg	rizka	rizka	Benar	rizka	Benar
2453	rizka_0_7470.jpeg	rizka	rizka	Benar	rizka	Benar
2454	rizka_0_7535.jpeg	rizka	rizka	Benar	rizka	Benar
2455	rizka_0_7587.jpeg	rizka	rizka	Benar	rizka	Benar
2456	rizka_0_7606.jpeg	rizka	rizka	Benar	rizka	Benar
2457	rizka_0_761.jpeg	rizka	rizka	Benar	rizka	Benar
2458	rizka_0_7669.jpeg	rizka	rizka	Benar	rizka	Benar
2459	rizka_0_7723.jpeg	rizka	rizka	Benar	rizka	Benar
2460	rizka_0_7726.jpeg	rizka	rizka	Benar	rizka	Benar
2461	rizka_0_7757.jpeg	rizka	rizka	Benar	rizka	Benar
2462	rizka_0_787.jpeg	rizka	rizka	Benar	rizka	Benar
2463	rizka_0_7920.jpeg	rizka	rizka	Benar	rizka	Benar
2464	rizka_0_7932.jpeg	rizka	rizka	Benar	rizka	Benar
2465	rizka_0_7958.jpeg	rizka	rizka	Benar	rizka	Benar
2466	rizka_0_7993.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2467	rizka_0_8007.jpeg	rizka	rizka	Benar	rizka	Benar
2468	rizka_0_8050.jpeg	rizka	rizka	Benar	rizka	Benar
2469	rizka_0_8072.jpeg	rizka	dinda	Salah	rizka	Benar
2470	rizka_0_811.jpeg	rizka	rizka	Benar	rizka	Benar
2471	rizka_0_8158.jpeg	rizka	rizka	Benar	rizka	Benar
2472	rizka_0_8163.jpeg	rizka	rizka	Benar	rizka	Benar
2473	rizka_0_820.jpeg	rizka	rizka	Benar	rizka	Benar
2474	rizka_0_8249.jpeg	rizka	rizka	Benar	rizka	Benar
2475	rizka_0_8339.jpeg	rizka	rizka	Benar	rizka	Benar
2476	rizka_0_8377.jpeg	rizka	rizka	Benar	rizka	Benar
2477	rizka_0_8484.jpeg	rizka	rizka	Benar	rizka	Benar
2478	rizka_0_850.jpeg	rizka	rizka	Benar	rizka	Benar
2479	rizka_0_8524.jpeg	rizka	rizka	Benar	rizka	Benar
2480	rizka_0_8550.jpeg	rizka	rizka	Benar	rizka	Benar
2481	rizka_0_8569.jpeg	rizka	rizka	Benar	rizka	Benar
2482	rizka_0_8582.jpeg	rizka	rizka	Benar	rizka	Benar
2483	rizka_0_8648.jpeg	rizka	rizka	Benar	rizka	Benar
2484	rizka_0_8666.jpeg	rizka	rizka	Benar	rizka	Benar
2485	rizka_0_8731.jpeg	rizka	rizka	Benar	rizka	Benar
2486	rizka_0_8739.jpeg	rizka	rizka	Benar	rizka	Benar
2487	rizka_0_8741.jpeg	rizka	rizka	Benar	rizka	Benar
2488	rizka_0_8750.jpeg	rizka	rizka	Benar	rizka	Benar
2489	rizka_0_8755.jpeg	rizka	rizka	Benar	rizka	Benar
2490	rizka_0_8768.jpeg	rizka	rizka	Benar	rizka	Benar
2491	rizka_0_8882.jpeg	rizka	rizka	Benar	rizka	Benar
2492	rizka_0_9019.jpeg	rizka	rizka	Benar	rizka	Benar
2493	rizka_0_9059.jpeg	rizka	rizka	Benar	rizka	Benar
2494	rizka_0_9068.jpeg	rizka	rizka	Benar	rizka	Benar
2495	rizka_0_9100.jpeg	rizka	rizka	Benar	rizka	Benar
2496	rizka_0_9107.jpeg	rizka	rizka	Benar	rizka	Benar
2497	rizka_0_9116.jpeg	rizka	rizka	Benar	rizka	Benar
2498	rizka_0_9155.jpeg	rizka	rizka	Benar	rizka	Benar
2499	rizka_0_9183.jpeg	rizka	rizka	Benar	rizka	Benar
2500	rizka_0_9209.jpeg	rizka	rizka	Benar	rizka	Benar
2501	rizka_0_9306.jpeg	rizka	rizka	Benar	rizka	Benar
2502	rizka_0_9375.jpeg	rizka	rizka	Benar	rizka	Benar
2503	rizka_0_9376.jpeg	rizka	rizka	Benar	rizka	Benar
2504	rizka_0_9401.jpeg	rizka	rizka	Benar	rizka	Benar
2505	rizka_0_9453.jpeg	rizka	rizka	Benar	rizka	Benar
2506	rizka_0_9471.jpeg	rizka	rizka	Benar	rizka	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2507	rizka_0_9473.jpeg	rizka	rizka	Benar	rizka	Benar
2508	rizka_0_9551.jpeg	rizka	rizka	Benar	rizka	Benar
2509	rizka_0_9559.jpeg	rizka	rizka	Benar	rizka	Benar
2510	rizka_0_9577.jpeg	rizka	rizka	Benar	rizka	Benar
2511	rizka_0_9579.jpeg	rizka	rizka	Benar	rizka	Benar
2512	rizka_0_9633.jpeg	rizka	dinda	Salah	rizka	Benar
2513	rizka_0_9678.jpeg	rizka	rizka	Benar	rizka	Benar
2514	rizka_0_9750.jpeg	rizka	rizka	Benar	rizka	Benar
2515	rizka_0_9835.jpeg	rizka	rizka	Benar	rizka	Benar
2516	rizka_0_9862.jpeg	rizka	rizka	Benar	rizka	Benar
2517	rizka_0_9871.jpeg	rizka	rizka	Benar	rizka	Benar
2518	rizka_0_9908.jpeg	rizka	rizka	Benar	rizka	Benar
2519	rizka_0_9941.jpeg	rizka	rizka	Benar	rizka	Benar
2520	rizka_0_9991.jpeg	rizka	rizka	Benar	rizka	Benar
2521	rusdi_0_1032.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2522	rusdi_0_1048.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2523	rusdi_0_1127.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2524	rusdi_0_1135.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2525	rusdi_0_1141.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2526	rusdi_0_1246.jpeg	rusdi	rusdi	Benar	imam	Salah
2527	rusdi_0_1248.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2528	rusdi_0_1376.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2529	rusdi_0_140.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2530	rusdi_0_1415.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2531	rusdi_0_1422.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2532	rusdi_0_1426.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2533	rusdi_0_1473.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2534	rusdi_0_1483.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2535	rusdi_0_1488.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2536	rusdi_0_1494.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2537	rusdi_0_1558.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2538	rusdi_0_1569.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2539	rusdi_0_1614.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2540	rusdi_0_1627.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2541	rusdi_0_163.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2542	rusdi_0_1710.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2543	rusdi_0_1741.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2544	rusdi_0_1746.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2545	rusdi_0_1759.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2546	rusdi_0_1766.jpeg	rusdi	rusdi	Benar	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2547	rusdi_0_1886.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2548	rusdi_0_1913.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2549	rusdi_0_1934.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2550	rusdi_0_1976.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2551	rusdi_0_1983.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2552	rusdi_0_1999.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2553	rusdi_0_2035.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2554	rusdi_0_2120.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2555	rusdi_0_2165.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2556	rusdi_0_2179.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2557	rusdi_0_2218.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2558	rusdi_0_224.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2559	rusdi_0_2315.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2560	rusdi_0_2429.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2561	rusdi_0_247.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2562	rusdi_0_2476.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2563	rusdi_0_2509.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2564	rusdi_0_2517.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2565	rusdi_0_2518.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2566	rusdi_0_2557.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2567	rusdi_0_258.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2568	rusdi_0_2718.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2569	rusdi_0_2833.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2570	rusdi_0_2856.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2571	rusdi_0_2929.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2572	rusdi_0_3.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2573	rusdi_0_3049.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2574	rusdi_0_3053.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2575	rusdi_0_3058.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2576	rusdi_0_3103.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2577	rusdi_0_3132.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2578	rusdi_0_3142.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2579	rusdi_0_3162.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2580	rusdi_0_3167.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2581	rusdi_0_3194.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2582	rusdi_0_3204.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2583	rusdi_0_3216.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2584	rusdi_0_3251.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2585	rusdi_0_1538.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2586	rusdi_0_2030.jpeg	rusdi	rusdi	Benar	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2587	rusdi_0_2742.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2588	rusdi_0_3353.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2589	rusdi_0_3862.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2590	rusdi_0_4489.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2591	rusdi_0_532.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2592	rusdi_0_5925.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2593	rusdi_0_6415.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2594	rusdi_0_6997.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2595	rusdi_0_7756.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2596	rusdi_0_8333.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2597	rusdi_0_9032.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2598	rusdi_0_3356.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2599	rusdi_0_3450.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2600	rusdi_0_3607.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2601	rusdi_0_3629.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2602	rusdi_0_3638.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2603	rusdi_0_3667.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2604	rusdi_0_3668.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2605	rusdi_0_3674.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2606	rusdi_0_3746.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2607	rusdi_0_3763.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2608	rusdi_0_3768.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2609	rusdi_0_3785.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2610	rusdi_0_3802.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2611	rusdi_0_3823.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2612	rusdi_0_3849.jpeg	rusdi	ilham	Salah	rusdi	Benar
2613	rusdi_0_3861.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2614	rusdi_0_3885.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2615	rusdi_0_40.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2616	rusdi_0_4053.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2617	rusdi_0_4133.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2618	rusdi_0_4144.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2619	rusdi_0_4171.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2620	rusdi_0_4184.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2621	rusdi_0_4187.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2622	rusdi_0_4193.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2623	rusdi_0_4275.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2624	rusdi_0_4332.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2625	rusdi_0_4345.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2626	rusdi_0_4380.jpeg	rusdi	rusdi	Benar	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2627	rusdi_0_4387.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2628	rusdi_0_4414.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2629	rusdi_0_444.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2630	rusdi_0_4570.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2631	rusdi_0_4688.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2632	rusdi_0_4767.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2633	rusdi_0_4778.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2634	rusdi_0_4786.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2635	rusdi_0_4837.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2636	rusdi_0_4892.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2637	rusdi_0_4910.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2638	rusdi_0_4949.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2639	rusdi_0_5037.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2640	rusdi_0_516.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2641	rusdi_0_517.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2642	rusdi_0_5185.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2643	rusdi_0_5197.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2644	rusdi_0_5222.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2645	rusdi_0_5309.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2646	rusdi_0_5411.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2647	rusdi_0_5428.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2648	rusdi_0_5491.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2649	rusdi_0_5522.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2650	rusdi_0_5538.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2651	rusdi_0_5551.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2652	rusdi_0_5606.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2653	rusdi_0_5638.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2654	rusdi_0_5684.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2655	rusdi_0_5739.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2656	rusdi_0_5800.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2657	rusdi_0_584.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2658	rusdi_0_5869.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2659	rusdi_0_5881.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2660	rusdi_0_5888.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2661	rusdi_0_5908.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2662	rusdi_0_60.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2663	rusdi_0_6024.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2664	rusdi_0_6026.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2665	rusdi_0_6053.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2666	rusdi_0_6098.jpeg	rusdi	rusdi	Benar	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2667	rusdi_0_6131.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2668	rusdi_0_6146.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2669	rusdi_0_6160.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2670	rusdi_0_6247.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2671	rusdi_0_6252.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2672	rusdi_0_6266.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2673	rusdi_0_6285.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2674	rusdi_0_6314.jpeg	rusdi	ilham	Salah	rusdi	Benar
2675	rusdi_0_6316.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2676	rusdi_0_6329.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2677	rusdi_0_6349.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2678	rusdi_0_6429.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2679	rusdi_0_6455.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2680	rusdi_0_6458.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2681	rusdi_0_6507.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2682	rusdi_0_6519.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2683	rusdi_0_6577.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2684	rusdi_0_6658.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2685	rusdi_0_666.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2686	rusdi_0_6665.jpeg	rusdi	ari	Salah	rusdi	Benar
2687	rusdi_0_6692.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2688	rusdi_0_6698.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2689	rusdi_0_6818.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2690	rusdi_0_6822.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2691	rusdi_0_6890.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2692	rusdi_0_6957.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2693	rusdi_0_6991.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2694	rusdi_0_7030.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2695	rusdi_0_7046.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2696	rusdi_0_7059.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2697	rusdi_0_7083.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2698	rusdi_0_7125.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2699	rusdi_0_7239.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2700	rusdi_0_7256.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2701	rusdi_0_7284.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2702	rusdi_0_7292.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2703	rusdi_0_7421.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2704	rusdi_0_7548.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2705	rusdi_0_7560.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2706	rusdi_0_7661.jpeg	rusdi	ari	Salah	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2707	rusdi_0_769.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2708	rusdi_0_7729.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2709	rusdi_0_7735.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2710	rusdi_0_7762.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2711	rusdi_0_7778.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2712	rusdi_0_7809.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2713	rusdi_0_7896.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2714	rusdi_0_790.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2715	rusdi_0_7970.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2716	rusdi_0_8060.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2717	rusdi_0_8150.jpeg	rusdi	ilham	Salah	rusdi	Benar
2718	rusdi_0_8174.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2719	rusdi_0_8185.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2720	rusdi_0_8188.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2721	rusdi_0_8201.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2722	rusdi_0_8210.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2723	rusdi_0_8266.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2724	rusdi_0_827.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2725	rusdi_0_8293.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2726	rusdi_0_8485.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2727	rusdi_0_8490.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2728	rusdi_0_8596.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2729	rusdi_0_8597.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2730	rusdi_0_8652.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2731	rusdi_0_8718.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2732	rusdi_0_8731.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2733	rusdi_0_8793.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2734	rusdi_0_8807.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2735	rusdi_0_890.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2736	rusdi_0_8901.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2737	rusdi_0_8907.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2738	rusdi_0_8991.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2739	rusdi_0_8998.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2740	rusdi_0_9012.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2741	rusdi_0_9031.jpeg	rusdi	ilham	Salah	rusdi	Benar
2742	rusdi_0_9088.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2743	rusdi_0_9158.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2744	rusdi_0_9180.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2745	rusdi_0_9182.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2746	rusdi_0_9186.jpeg	rusdi	rusdi	Benar	rusdi	Benar

No.	Citra Uji	Label	vgg16	Ket.	simple_cnn	Ket.
2747	rusdi_0_9224.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2748	rusdi_0_9234.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2749	rusdi_0_9381.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2750	rusdi_0_9386.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2751	rusdi_0_9424.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2752	rusdi_0_9499.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2753	rusdi_0_9508.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2754	rusdi_0_9526.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2755	rusdi_0_9556.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2756	rusdi_0_9625.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2757	rusdi_0_9706.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2758	rusdi_0_971.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2759	rusdi_0_9733.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2760	rusdi_0_9746.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2761	rusdi_0_9767.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2762	rusdi_0_9792.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2763	rusdi_0_9806.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2764	rusdi_0_9818.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2765	rusdi_0_9838.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2766	rusdi_0_9866.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2767	rusdi_0_9872.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2768	rusdi_0_9897.jpeg	rusdi	rusdi	Benar	rusdi	Benar
2769	rusdi_0_997.jpeg	rusdi	rusdi	Benar	rusdi	Benar

Lampiran III. Kode Program

Berikut merupakan kode Program yang digunakan untuk membangun sistem pengenalan wajah secara realtime. Kode program berikut dapat diunduh pada laman https://bitbucket.org/arisusanto-ta/face_recognition.

1. face_recognition.py

```
from vgg16 import VGG16
import split_folders
from face_ai.deepnet import layers, models
from face_ai.deepnet.preprocessing.image import ImageDataGenerator, load_img, img_to_array
import numpy as np
from face_ai.deepnet.function.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from face_ai.deepnet.function.optimizers import SGD
from face_ai.deepnet.function.regularizers import l2
from face_ai.deepnet.function.callbacks import ReduceLROnPlateau
from face_ai.data import load_metadata
import simple_cnn


import argparse
import pickle
import os
from pyprind import  ProgBar
```

```
image_gen = ImageDataGenerator(rotation_range=0,
                               featurewise_center=True,
                               featurewise_std_normalization=True,
                               width_shift_range=0.1,
                               height_shift_range=0.1,
                               shear_range=0.01,
                               zoom_range=[0.9, 1.25],
                               horizontal_flip=True,
                               vertical_flip=False,
                               fill_mode='nearest',
                               data_format='channels_last',
                               brightness_range=[0.1, 2.0])


def load_images(metadata, height=96, width=96):
    img = np.empty((len(metadata), height, width, 3))
    i = 0
    for data in metadata:
        image = load_img(data.image_path())
        x = img_to_array(image)
        img[i] = x
        i += 1
    return img
```

```
    metadata = load_metadata('train_data/images')
    images = load_images(metadata)
    image_gen.fit(images)

    def create_test_data():
        output = 'test_data/images/'
        test_metadata = load_metadata('train_data/dataset/val')
        bar = ProgBar(len(test_metadata), bar_char='')
        for data in metadata:
            img = load_img(data.image_path())
            x = img_to_array(img)
            x = x.reshape((1,) + x.shape)

            output_path = output + data.name
            if not os.path.exists(output_path):
                os.makedirs(output_path)

            i = 0
            for batch in image_gen.flow(x, batch_size=1,
                                        save_to_dir=output_path,    save_prefix=data.name,
save_format='jpeg'):
                i += 1
```

```
            if i > 16:
                break  # otherwise the generator would loop indefinitely
        bar.update()


def create_label_encoder():
    images_path = 'train_data/images'
    cls_name = []
    for i in sorted(os.listdir(images_path)):
        cls_name.append(i)

    pickle.dump(cls_name, open('face_ai/models/label_encoder', 'wb'))

    return cls_name


def create_model_vgg16(num_classes):
    conv_base = VGG16(weights='imagenet',
                      include_top=False,
                      input_shape=(96, 96, 3))

    # for layer in conv_base.layers[:-4]:
    #     layer.trainable = False
```

```
        conv_base.trainable = False

        model = models.Sequential()
        model.add(conv_base)
        model.add(layers.Flatten())
        model.add(layers.Dense(1024, activation='relu'))
        model.add(layers.Dropout(0.5))
        model.add(layers.Dense(num_classes,  kernel_regularizer=l2(.0005),
activation='softmax'))

        return model

    def create_simple_model(num_classes):
        return simple_cnn.build((96,96,3),num_classes)


    def create_data_generator(train_dir, test_dir, batch_size):

        data_gen = ImageDataGenerator(rotation_range=0,
                                      featurewise_center=False,
                                      featurewise_std_normalization=False,
                                      horizontal_flip=True,
                                      vertical_flip=False,
```

```
                    data_format='channels_last',
                    rescale=1./255
                    )

    train_data = data_gen.flow_from_directory(
        directory=train_dir,
        target_size=(96, 96),
        color_mode='rgb',
        batch_size=batch_size,
        seed=11
    )
    validation_data = data_gen.flow_from_directory(
        directory=test_dir,
        target_size=(96, 96),
        color_mode='rgb',
        batch_size=batch_size,
        seed=11
    )

    return train_data, validation_data


def train_model(model, epochs, batch_size):
```

```
        train_data, validation_data = create_data_generator('test_data/train',
                                                            'test_data/val',
                                                            batch_size)

        # callbacks
        base_path = 'train_data/'
        patience = 50
        log_file_path = base_path + 'logs/training.log'
        csv_logger = CSVLogger(log_file_path, append=False)
        reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                                      patience=int(patience / 4), verbose=1)
        early_stop = EarlyStopping('val_loss', patience=patience)
        trained_models_path = base_path + 'models/_vgg16_'
        model_names = trained_models_path + '.{epoch:02d}-
{val_acc:.2f}.hdf5'
        model_checkpoint = ModelCheckpoint(model_names, 'val_loss',
verbose=1,
                                           save_best_only=True)
        callbacks = [model_checkpoint, csv_logger, reduce_lr, early_stop]

        opt = SGD(lr=.01, momentum=.9)
        model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
        model.fit_generator(
            generator=train_data,
            validation_data=validation_data,
            steps_per_epoch= 160,
            validation_steps= 160,
            epochs=epochs,
            verbose=1,
            callbacks=callbacks
        )


    if __name__ == '__main__':
        parser = argparse.ArgumentParser(description='Train face recognition
model')
        parser.add_argument('--numclasses', '-n', type=int, default=11,
                            help='Number of classes')
        parser.add_argument('--batchsize', '-b', type=int, default=64,
                            help='Number of images in each mini-batch')
        parser.add_argument('--epochs', '-e', type=int, default=1000,
                            help='Number of sweeps over the dataset to train')
        parser.add_argument('--model', '-m', default='vgg16',
                            help='model selection')
```

```
        parser.add_argument('--option', '-o', default='start_train',
                            help='Option for vgg16')
        args = parser.parse_args()

        if args.model == 'vgg16':
            model = create_model_vgg16(args.numclasses)
        else:
            model = model = create_simple_model(args.numclasses)

        if args.option == 'model_details':
            model.summary()
        elif args.option == 'split':
            split_folders.ratio('train_data/images',      output="train_data/dataset",
seed=1337, ratio=(.8, .2))
        elif args.option == 'start_train':
            train_model(model, args.epochs, args.batchsize)
        elif args.option == 'encode_label':
            cls_name = create_label_encoder()
            print(f'{len(cls_name)} Labels encoded')
            print(cls_name)
        elif args.option == 'create_test':
            create_test_data()
        else:
```

```
print(f'Error: No option name: {args.option}')
```

2. vgg16.py

```
from face_ai.deepnet import (layers, backend, models, utils)
import json
import warnings
import numpy as np
import os

WEIGHTS_PATH = ('https://github.com/fchollet/deep-learning-models/'
                'releases/download/v0.1/'
                'vgg16_weights_tf_dim_ordering_tf_kernels.h5')
WEIGHTS_PATH_NO_TOP = ('https://github.com/fchollet/deep-learning-models/'
                       'releases/download/v0.1/'
                       'vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5')

def _obtain_input_shape(input_shape,
                        default_size,
                        min_size,
                        data_format,
                        require_flatten,
                        weights=None):
```

```
    """Internal utility to compute/validate a model's input shape.

    # Arguments
        input_shape: Either None (will return the default network input
shape),
            or a user-provided shape to be validated.
        default_size: Default input width/height for the model.
        min_size: Minimum input width/height accepted by the model.
        data_format: Image data format to use.
        require_flatten: Whether the model is expected to
            be linked to a classifier via a Flatten layer.
        weights: One of `None` (random initialization)
            or 'imagenet' (pre-training on ImageNet).
            If weights='imagenet' input channels must be equal to 3.

    # Returns
        An integer shape tuple (may include None entries).

    # Raises
        ValueError: In case of invalid argument values.
    """
    if weights != 'imagenet' and input_shape and len(input_shape) == 3:
        if data_format == 'channels_first':
```

```
        if input_shape[0] not in {1, 3}:
            warnings.warn(
                'This model usually expects 1 or 3 input channels. '
                'However, it was passed an input_shape with ' +
                str(input_shape[0]) + ' input channels.')
        default_shape = (input_shape[0], default_size, default_size)
    else:
        if input_shape[-1] not in {1, 3}:
            warnings.warn(
                'This model usually expects 1 or 3 input channels. '
                'However, it was passed an input_shape with ' +
                str(input_shape[-1]) + ' input channels.')
        default_shape = (default_size, default_size, input_shape[-1])
else:
    if data_format == 'channels_first':
        default_shape = (3, default_size, default_size)
    else:
        default_shape = (default_size, default_size, 3)
if weights == 'imagenet' and require_flatten:
    if input_shape is not None:
        if input_shape != default_shape:
            raise ValueError('When setting`include_top=True` '
                             'and loading `imagenet` weights, '
```

```
                                '`input_shape` should be ' +
                                str(default_shape) + '.')
            return default_shape
        if input_shape:
            if data_format == 'channels_first':
                if input_shape is not None:
                    if len(input_shape) != 3:
                        raise ValueError(
                            '`input_shape` must be a tuple of three integers.')
                    if input_shape[0] != 3 and weights == 'imagenet':
                        raise ValueError('The input must have 3 channels; got '
                                         '`input_shape=' + str(input_shape) + '`')
                    if ((input_shape[1] is not None and input_shape[1] <
min_size) or
                            (input_shape[2] is not None and input_shape[2] <
min_size)):
                        raise ValueError('Input size must be at least ' +
                                         str(min_size) + 'x' + str(min_size) +
                                         '; got `input_shape=' +
                                         str(input_shape) + '`')
            else:
                if input_shape is not None:
                    if len(input_shape) != 3:
```

```
                raise ValueError(
                    '`input_shape` must be a tuple of three integers.')
            if input_shape[-1] != 3 and weights == 'imagenet':
                raise ValueError('The input must have 3 channels; got '
                                 '`input_shape=' + str(input_shape) + '`')
            if ((input_shape[0] is not None and input_shape[0] < min_size) or
                (input_shape[1] is not None and input_shape[1] < min_size)):
                raise ValueError('Input size must be at least ' +
                                 str(min_size) + 'x' + str(min_size) +
                                 '; got `input_shape=' +
                                 str(input_shape) + '`')
        else:
            if require_flatten:
                input_shape = default_shape
            else:
                if data_format == 'channels_first':
                    input_shape = (3, None, None)
                else:
                    input_shape = (None, None, 3)
        if require_flatten:
            if None in input_shape:
```

```
            raise ValueError('If `include_top` is True, '
                             'you should specify a static `input_shape`. '
                             'Got `input_shape=' + str(input_shape) + '`')
    return input_shape


def VGG16(include_top=True,
          weights='imagenet',
          input_tensor=None,
          input_shape=None,
          pooling=None,
          classes=1000):
    """Instantiates the VGG16 architecture.

        # Arguments
            include_top: whether to include the 3 fully-connected
                layers at the top of the network.
            weights: one of `None` (random initialization),
                  'imagenet' (pre-training on ImageNet),
                  or the path to the weights file to be loaded.
            input_tensor: optional Keras tensor
                (i.e. output of `layers.Input()`)
                to use as image input for the model.
            input_shape: optional shape tuple, only to be specified
```

```
    if `include_top` is False (otherwise the input shape
    has to be `(224, 224, 3)`
    (with `channels_last` data format)
    or `(3, 224, 224)` (with `channels_first` data format).
    It should have exactly 3 input channels,
    and width and height should be no smaller than 48.
    E.g. `(200, 200, 3)` would be one valid value.
  pooling: Optional pooling mode for feature extraction
    when `include_top` is `False`.
    - `None` means that the output of the model will be
        the 4D tensor output of the
        last convolutional layer.
    - `avg` means that global average pooling
        will be applied to the output of the
        last convolutional layer, and thus
        the output of the model will be a 2D tensor.
    - `max` means that global max pooling will
        be applied.
  classes: optional number of classes to classify images
    into, only to be specified if `include_top` is True, and
    if no `weights` argument is specified.

# Returns
```

```
        A model instance.

    # Raises
        ValueError: in case of invalid argument for `weights`,
            or invalid input shape.
    """
    if not (weights in {'imagenet', None} or os.path.exists(weights)):
        raise ValueError('The `weights` argument should be either '
                         '`None` (random initialization), `imagenet` '
                         '(pre-training on ImageNet), '
                         'or the path to the weights file to be loaded.')

    if weights == 'imagenet' and include_top and classes != 1000:
        raise ValueError('If using `weights` as `"imagenet"` with `include_top`'
                         ' as true, `classes` should be 1000')
    # Determine proper input shape
    input_shape = _obtain_input_shape(input_shape,
                                      default_size=224,
                                      min_size=32,
                                      data_format=backend.image_data_format(),
                                      require_flatten=include_top,
                                      weights=weights)
```

```
        if input_tensor is None:
            img_input = layers.Input(shape=input_shape)
        else:
            if not backend.is_deepnet_tensor(input_tensor):
                img_input = layers.Input(tensor=input_tensor,
shape=input_shape)
            else:
                img_input = input_tensor


        # Block 1
        x = layers.Conv2D(64, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block1_conv1')(img_input)
        x = layers.Conv2D(64, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block1_conv2')(x)
        x = layers.MaxPooling2D((2, 2), strides=(2, 2),
name='block1_pool')(x)


        # Block 2
```

```
        x = layers.Conv2D(128, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block2_conv1')(x)
        x = layers.Conv2D(128, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block2_conv2')(x)
        x = layers.MaxPooling2D((2, 2), strides=(2, 2),
name='block2_pool')(x)

        # Block 3
        x = layers.Conv2D(256, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block3_conv1')(x)
        x = layers.Conv2D(256, (3, 3),
                          activation='relu',
                          padding='same',
                          name='block3_conv2')(x)
        x = layers.Conv2D(256, (3, 3),
                          activation='relu',
                          padding='same',
```

```
                name='block3_conv3')(x)
        x = layers.MaxPooling2D((2, 2), strides=(2, 2),
name='block3_pool')(x)

        # Block 4
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
                padding='same',
                name='block4_conv1')(x)
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
                padding='same',
                name='block4_conv2')(x)
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
                padding='same',
                name='block4_conv3')(x)
        x = layers.MaxPooling2D((2, 2), strides=(2, 2),
name='block4_pool')(x)

        # Block 5
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
```

```
                padding='same',
                name='block5_conv1')(x)
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
                padding='same',
                name='block5_conv2')(x)
        x = layers.Conv2D(512, (3, 3),
                activation='relu',
                padding='same',
                name='block5_conv3')(x)
        x = layers.MaxPooling2D((2, 2), strides=(2, 2),
name='block5_pool')(x)

        if pooling == 'avg':
            x = layers.GlobalAveragePooling2D()(x)
        elif pooling == 'max':
            x = layers.GlobalMaxPooling2D()(x)

        # Ensure that the model takes into account
        # any potential predecessors of `input_tensor`.
        if input_tensor is not None:
            inputs = utils.get_source_inputs(input_tensor)
        else:
```

```
    inputs = img_input
# Create model.
model = models.Model(inputs, x, name='vgg16')


# Load weights.
if weights == 'imagenet':
    if include_top:
        weights_path = utils.get_file(
            'vgg16_weights_tf_dim_ordering_tf_kernels.h5',
            WEIGHTS_PATH,
            cache_subdir='models',
            file_hash='64373286793e3c8b2b4e3219cbf3544b')
    else:
        weights_path = utils.get_file(
            'vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5',
            WEIGHTS_PATH_NO_TOP,
            cache_subdir='models',
            file_hash='6d6bbae143d832006294945121d1f1fc')
    model.load_weights(weights_path)
    if backend.backend() == 'theano':
        utils.convert_all_kernels_in_model(model)
elif weights is not None:
    model.load_weights(weights)
```

```
    return model
```

3. image.py

```
"""Utilities for real-time data augmentation on image data.
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import re
from six.moves import range
import os
import threading
import warnings
import multiprocessing.pool
from functools import partial

from . import get_deepnet_submodule

backend = get_deepnet_submodule('backend')
deepnet_utils = get_deepnet_submodule('utils')
```

```
try:
    from PIL import ImageEnhance
    from PIL import Image as pil_image
except ImportError:
    pil_image = None
    ImageEnhance = None


try:
    import scipy
except ImportError:
    scipy = None


if pil_image is not None:
    _PIL_INTERPOLATION_METHODS = {
        'nearest': pil_image.NEAREST,
        'bilinear': pil_image.BILINEAR,
        'bicubic': pil_image.BICUBIC,
    }
    if hasattr(pil_image, 'HAMMING'):
        _PIL_INTERPOLATION_METHODS['hamming'] = pil_image.HAMMING
    if hasattr(pil_image, 'BOX'):
        _PIL_INTERPOLATION_METHODS['box'] = pil_image.BOX
    if hasattr(pil_image, 'LANCZOS'):
```

```
        _PIL_INTERPOLATION_METHODS['lanczos'] = pil_image.LANCZOS


def random_rotation(x, rg, row_axis=1, col_axis=2, channel_axis=0,
                    fill_mode='nearest', cval=0.):
    """Performs a random rotation of a Numpy image tensor.

    # Arguments
        x: Input tensor. Must be 3D.
        rg: Rotation range, in degrees.
        row_axis: Index of axis for rows in the input tensor.
        col_axis: Index of axis for columns in the input tensor.
        channel_axis: Index of axis for channels in the input tensor.
        fill_mode: Points outside the boundaries of the input
            are filled according to the given mode
            (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).
        cval: Value used for points outside the boundaries
            of the input if `mode='constant'`.

    # Returns
        Rotated Numpy image tensor.
    """
    theta = np.random.uniform(-rg, rg)
```

```
    x = apply_affine_transform(x, theta=theta, channel_axis=channel_axis,
                               fill_mode=fill_mode, cval=cval)
    return x


def random_shift(x, wrg, hrg, row_axis=1, col_axis=2, channel_axis=0,
                 fill_mode='nearest', cval=0.):
    """Performs a random spatial shift of a Numpy image tensor.

    # Arguments
        x: Input tensor. Must be 3D.
        wrg: Width shift range, as a float fraction of the width.
        hrg: Height shift range, as a float fraction of the height.
        row_axis: Index of axis for rows in the input tensor.
        col_axis: Index of axis for columns in the input tensor.
        channel_axis: Index of axis for channels in the input tensor.
        fill_mode: Points outside the boundaries of the input
            are filled according to the given mode
            (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).
        cval: Value used for points outside the boundaries
            of the input if `mode='constant'`.

    # Returns
```

```
    Shifted Numpy image tensor.
    """
    h, w = x.shape[row_axis], x.shape[col_axis]
    tx = np.random.uniform(-hrg, hrg) * h
    ty = np.random.uniform(-wrg, wrg) * w
    x = apply_affine_transform(x, tx=tx, ty=ty, channel_axis=channel_axis,
                               fill_mode=fill_mode, cval=cval)
    return x


def random_shear(x, intensity, row_axis=1, col_axis=2, channel_axis=0,
                 fill_mode='nearest', cval=0.):
    """Performs a random spatial shear of a Numpy image tensor.

    # Arguments
        x: Input tensor. Must be 3D.
        intensity: Transformation intensity in degrees.
        row_axis: Index of axis for rows in the input tensor.
        col_axis: Index of axis for columns in the input tensor.
        channel_axis: Index of axis for channels in the input tensor.
        fill_mode: Points outside the boundaries of the input
            are filled according to the given mode
            (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).
```

```
        cval: Value used for points outside the boundaries
            of the input if `mode='constant'`.

    # Returns
        Sheared Numpy image tensor.
    """
    shear = np.random.uniform(-intensity, intensity)
    x = apply_affine_transform(x, shear=shear, channel_axis=channel_axis,
                               fill_mode=fill_mode, cval=cval)
    return x


def random_zoom(x, zoom_range, row_axis=1, col_axis=2, channel_axis=0,
                fill_mode='nearest', cval=0.):
    """Performs a random spatial zoom of a Numpy image tensor.

    # Arguments
        x: Input tensor. Must be 3D.
        zoom_range: Tuple of floats; zoom range for width and height.
        row_axis: Index of axis for rows in the input tensor.
        col_axis: Index of axis for columns in the input tensor.
        channel_axis: Index of axis for channels in the input tensor.
        fill_mode: Points outside the boundaries of the input
```

```
        are filled according to the given mode
        (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).
    cval: Value used for points outside the boundaries
        of the input if `mode='constant'`.

# Returns
    Zoomed Numpy image tensor.

# Raises
    ValueError: if `zoom_range` isn't a tuple.
"""
if len(zoom_range) != 2:
    raise ValueError('`zoom_range` should be a tuple or list of two'
                     ' floats. Received: ', (zoom_range,))

if zoom_range[0] == 1 and zoom_range[1] == 1:
    zx, zy = 1, 1
else:
    zx, zy = np.random.uniform(zoom_range[0], zoom_range[1], 2)
x = apply_affine_transform(x, zx=zx, zy=zy, channel_axis=channel_axis,
                           fill_mode=fill_mode, cval=cval)
return x
```

```
def apply_channel_shift(x, intensity, channel_axis=0):
    """Performs a channel shift.

    # Arguments
        x: Input tensor. Must be 3D.
        intensity: Transformation intensity.
        channel_axis: Index of axis for channels in the input tensor.

    # Returns
        Numpy image tensor.

    """
    x = np.rollaxis(x, channel_axis, 0)
    min_x, max_x = np.min(x), np.max(x)
    channel_images = [
        np.clip(x_channel + intensity,
                min_x,
                max_x)
        for x_channel in x]
    x = np.stack(channel_images, axis=0)
    x = np.rollaxis(x, 0, channel_axis + 1)
    return x
```

```
def random_channel_shift(x, intensity_range, channel_axis=0):
    """Performs a random channel shift.

    # Arguments
        x: Input tensor. Must be 3D.
        intensity_range: Transformation intensity.
        channel_axis: Index of axis for channels in the input tensor.

    # Returns
        Numpy image tensor.
    """
    intensity = np.random.uniform(-intensity_range, intensity_range)
    return apply_channel_shift(x, intensity, channel_axis=channel_axis)


def apply_brightness_shift(x, brightness):
    """Performs a brightness shift.

    # Arguments
        x: Input tensor. Must be 3D.
        brightness: Float. The new brightness value.
```

```
        channel_axis: Index of axis for channels in the input tensor.


    # Returns
        Numpy image tensor.


    # Raises
        ValueError if `brightness_range` isn't a tuple.
    """
    if ImageEnhance is None:
        raise ImportError('Using brightness shifts requires PIL. '
                          'Install PIL or Pillow.')
    x = array_to_img(x)
    x = imgenhancer_Brightness = ImageEnhance.Brightness(x)
    x = imgenhancer_Brightness.enhance(brightness)
    x = img_to_array(x)
    return x



def random_brightness(x, brightness_range):
    """Performs a random brightness shift.


    # Arguments
        x: Input tensor. Must be 3D.
```

```
        brightness_range: Tuple of floats; brightness range.
        channel_axis: Index of axis for channels in the input tensor.

    # Returns
        Numpy image tensor.

    # Raises
        ValueError if `brightness_range` isn't a tuple.
    """
    if len(brightness_range) != 2:
        raise ValueError(
            '`brightness_range should be tuple or list of two floats. '
            'Received: %s' % (brightness_range,))

    u = np.random.uniform(brightness_range[0], brightness_range[1])
    return apply_brightness_shift(x, u)


def transform_matrix_offset_center(matrix, x, y):
    o_x = float(x) / 2 + 0.5
    o_y = float(y) / 2 + 0.5
    offset_matrix = np.array([[1, 0, o_x], [0, 1, o_y], [0, 0, 1]])
    reset_matrix = np.array([[1, 0, -o_x], [0, 1, -o_y], [0, 0, 1]])
```

```
    transform_matrix = np.dot(np.dot(offset_matrix, matrix), reset_matrix)
    return transform_matrix


def apply_affine_transform(x, theta=0, tx=0, ty=0, shear=0, zx=1, zy=1,
                           row_axis=0, col_axis=1, channel_axis=2,
                           fill_mode='nearest', cval=0.):
    """Applies an affine transformation specified by the parameters given.

    # Arguments
        x: 2D numpy array, single image.
        theta: Rotation angle in degrees.
        tx: Width shift.
        ty: Heigh shift.
        shear: Shear angle in degrees.
        zx: Zoom in x direction.
        zy: Zoom in y direction
        row_axis: Index of axis for rows in the input image.
        col_axis: Index of axis for columns in the input image.
        channel_axis: Index of axis for channels in the input image.
        fill_mode: Points outside the boundaries of the input
            are filled according to the given mode
            (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).
```

```
    cval: Value used for points outside the boundaries
        of the input if `mode='constant'`.


# Returns
    The transformed version of the input.
"""
if scipy is None:
    raise ImportError('Image transformations require SciPy. '
                      'Install SciPy.')
transform_matrix = None
if theta != 0:
    theta = np.deg2rad(theta)
    rotation_matrix = np.array([[np.cos(theta), -np.sin(theta), 0],
                                [np.sin(theta), np.cos(theta), 0],
                                [0, 0, 1]])
    transform_matrix = rotation_matrix


if tx != 0 or ty != 0:
    shift_matrix = np.array([[1, 0, tx],
                             [0, 1, ty],
                             [0, 0, 1]])
    if transform_matrix is None:
        transform_matrix = shift_matrix
```

```
    else:
        transform_matrix = np.dot(transform_matrix, shift_matrix)

if shear != 0:
    shear = np.deg2rad(shear)
    shear_matrix = np.array([[1, -np.sin(shear), 0],
                             [0, np.cos(shear), 0],
                             [0, 0, 1]])
    if transform_matrix is None:
        transform_matrix = shear_matrix
    else:
        transform_matrix = np.dot(transform_matrix, shear_matrix)

if zx != 1 or zy != 1:
    zoom_matrix = np.array([[zx, 0, 0],
                            [0, zy, 0],
                            [0, 0, 1]])
    if transform_matrix is None:
        transform_matrix = zoom_matrix
    else:
        transform_matrix = np.dot(transform_matrix, zoom_matrix)

if transform_matrix is not None:
```

```
        h, w = x.shape[row_axis], x.shape[col_axis]
        transform_matrix = transform_matrix_offset_center(
            transform_matrix, h, w)
        x = np.rollaxis(x, channel_axis, 0)
        final_affine_matrix = transform_matrix[:2, :2]
        final_offset = transform_matrix[:2, 2]

        channel_images = [scipy.ndimage.interpolation.affine_transform(
            x_channel,
            final_affine_matrix,
            final_offset,
            order=1,
            mode=fill_mode,
            cval=cval) for x_channel in x]
        x = np.stack(channel_images, axis=0)
        x = np.rollaxis(x, 0, channel_axis + 1)
    return x


def flip_axis(x, axis):
    x = np.asarray(x).swapaxes(axis, 0)
    x = x[::-1, ...]
    x = x.swapaxes(0, axis)
```

```
    return x


def array_to_img(x, data_format=None, scale=True):
    """Converts a 3D Numpy array to a PIL Image instance.

    # Arguments
        x: Input Numpy array.
        data_format: Image data format.
            either "channels_first" or "channels_last".
        scale: Whether to rescale image values
            to be within `[0, 255]`.

    # Returns
        A PIL Image instance.

    # Raises
        ImportError: if PIL is not available.
        ValueError: if invalid `x` or `data_format` is passed.
    """
    if pil_image is None:
        raise ImportError('Could not import PIL.Image. '
                          'The use of `array_to_img` requires PIL.')
```

```
x = np.asarray(x, dtype=backend.floatx())
if x.ndim != 3:
    raise ValueError('Expected image array to have rank 3 (single image). '
                     'Got array with shape:', x.shape)

if data_format is None:
    data_format = backend.image_data_format()
if data_format not in {'channels_first', 'channels_last'}:
    raise ValueError('Invalid data_format:', data_format)

# Original Numpy array x has format (height, width, channel)
# or (channel, height, width)
# but target PIL image has format (width, height, channel)
if data_format == 'channels_first':
    x = x.transpose(1, 2, 0)
if scale:
    x = x + max(-np.min(x), 0)
    x_max = np.max(x)
    if x_max != 0:
        x /= x_max
    x *= 255
if x.shape[2] == 4:
    # RGBA
```

```
            return pil_image.fromarray(x.astype('uint8'), 'RGBA')
        elif x.shape[2] == 3:
            # RGB
            return pil_image.fromarray(x.astype('uint8'), 'RGB')
        elif x.shape[2] == 1:
            # grayscale
            return pil_image.fromarray(x[:, :, 0].astype('uint8'), 'L')
        else:
            raise ValueError('Unsupported channel number: ', x.shape[2])


    def img_to_array(img, data_format=None):
        """Converts a PIL Image instance to a Numpy array.

        # Arguments
            img: PIL Image instance.
            data_format: Image data format,
                either "channels_first" or "channels_last".

        # Returns
            A 3D Numpy array.

        # Raises
```

```
    ValueError: if invalid `img` or `data_format` is passed.
"""
if data_format is None:
    data_format = backend.image_data_format()
if data_format not in {'channels_first', 'channels_last'}:
    raise ValueError('Unknown data_format: ', data_format)
# Numpy array x has format (height, width, channel)
# or (channel, height, width)
# but original PIL image has format (width, height, channel)
x = np.asarray(img, dtype=backend.floatx())
if len(x.shape) == 3:
    if data_format == 'channels_first':
        x = x.transpose(2, 0, 1)
elif len(x.shape) == 2:
    if data_format == 'channels_first':
        x = x.reshape((1, x.shape[0], x.shape[1]))
    else:
        x = x.reshape((x.shape[0], x.shape[1], 1))
else:
    raise ValueError('Unsupported image shape: ', x.shape)
return x
```

```
def save_img(path,
             x,
             data_format=None,
             file_format=None,
             scale=True, **kwargs):
    """Saves an image stored as a Numpy array to a path or file object.

    # Arguments
        path: Path or file object.
        x: Numpy array.
        data_format: Image data format,
            either "channels_first" or "channels_last".
        file_format: Optional file format override. If omitted, the
            format to use is determined from the filename extension.
            If a file object was used instead of a filename, this
            parameter should always be used.
        scale: Whether to rescale image values to be within `[0, 255]`.
        **kwargs: Additional keyword arguments passed to `PIL.Image.save()`.
    """
    img = array_to_img(x, data_format=data_format, scale=scale)
    if img.mode == 'RGBA' and (file_format == 'jpg' or file_format == 'jpeg'):
        warnings.warn('The JPG format does not support '
                      'RGBA images, converting to RGB.')
```

```
        img = img.convert('RGB')
    img.save(path, format=file_format, **kwargs)


def load_img(path, grayscale=False, color_mode='rgb', target_size=None,
             interpolation='nearest'):
    """Loads an image into PIL format.

    # Arguments
        path: Path to image file.
        color_mode: One of "grayscale", "rbg", "rgba". Default: "rgb".
            The desired image format.
        target_size: Either `None` (default to original size)
            or tuple of ints `(img_height, img_width)`.
        interpolation: Interpolation method used to resample the image if the
            target size is different from that of the loaded image.
            Supported methods are "nearest", "bilinear", and "bicubic".
            If PIL version 1.1.3 or newer is installed, "lanczos" is also
            supported. If PIL version 3.4.0 or newer is installed, "box" and
            "hamming" are also supported. By default, "nearest" is used.

    # Returns
        A PIL Image instance.
```

```
# Raises
    ImportError: if PIL is not available.
    ValueError: if interpolation method is not supported.
"""
if grayscale is True:
    warnings.warn('grayscale is deprecated. Please use '
                  'color_mode = "grayscale"')
    color_mode = 'grayscale'
if pil_image is None:
    raise ImportError('Could not import PIL.Image. '
                      'The use of `array_to_img` requires PIL.')
img = pil_image.open(path)
if color_mode == 'grayscale':
    if img.mode != 'L':
        img = img.convert('L')
elif color_mode == 'rgba':
    if img.mode != 'RGBA':
        img = img.convert('RGBA')
elif color_mode == 'rgb':
    if img.mode != 'RGB':
        img = img.convert('RGB')
else:
```

```
            raise ValueError('color_mode must be "grayscale", "rbg", or "rgba"')
        if target_size is not None:
            width_height_tuple = (target_size[1], target_size[0])
            if img.size != width_height_tuple:
                if interpolation not in _PIL_INTERPOLATION_METHODS:
                    raise ValueError(
                        'Invalid interpolation method {} specified. Supported '
                        'methods are {}'.format(
                            interpolation,
                            ", ".join(_PIL_INTERPOLATION_METHODS.keys())))
                resample = _PIL_INTERPOLATION_METHODS[interpolation]
                img = img.resize(width_height_tuple, resample)
        return img


def list_pictures(directory, ext='jpg|jpeg|bmp|png|ppm'):
    return [os.path.join(root, f)
            for root, _, files in os.walk(directory) for f in files
            if re.match(r'([\w]+\.(?:' + ext + '))', f.lower())]


class ImageDataGenerator(object):
    """Generate batches of tensor image data with real-time data augmentation.
```

The data will be looped over (in batches).

Arguments

featurewise_center: Boolean.

Set input mean to 0 over the dataset, feature-wise.

samplewise_center: Boolean. Set each sample mean to 0.

featurewise_std_normalization: Boolean.

Divide inputs by std of the dataset, feature-wise.

samplewise_std_normalization: Boolean. Divide each input by its std.

zca_epsilon: epsilon for ZCA whitening. Default is 1e-6.

zca_whitening: Boolean. Apply ZCA whitening.

rotation_range: Int. Degree range for random rotations.

width_shift_range: Float, 1-D array-like or int

- float: fraction of total width, if < 1, or pixels if >= 1.
- 1-D array-like: random elements from the array.
- int: integer number of pixels from interval
  `(-width_shift_range, +width_shift_range)`
- With `width_shift_range=2` possible values
  are integers `[-1, 0, +1]`,
  same as with `width_shift_range=[-1, 0, +1]`,
  while with `width_shift_range=1.0` possible values are floats
  in the interval [-1.0, +1.0).

height_shift_range: Float, 1-D array-like or int

- float: fraction of total height, if < 1, or pixels if >= 1.
- 1-D array-like: random elements from the array.
- int: integer number of pixels from interval
  `(-height_shift_range, +height_shift_range)`
- With `height_shift_range=2` possible values
  are integers `[-1, 0, +1]`,
  same as with `height_shift_range=[-1, 0, +1]`,
  while with `height_shift_range=1.0` possible values are floats
  in the interval [-1.0, +1.0).

brightness_range: Tuple or list of two floats. Range for picking
  a brightness shift value from.

shear_range: Float. Shear Intensity
  (Shear angle in counter-clockwise direction in degrees)

zoom_range: Float or [lower, upper]. Range for random zoom.
  If a float, `[lower, upper] = [1-zoom_range, 1+zoom_range]`.

channel_shift_range: Float. Range for random channel shifts.

fill_mode: One of {"constant", "nearest", "reflect" or "wrap"}.
  Default is 'nearest'.
  Points outside the boundaries of the input are filled
  according to the given mode:
  - 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)
  - 'nearest': aaaaaaaa|abcd|dddddddd
  - 'reflect': abcddcba|abcd|dcbaabcd

- 'wrap': abcdabcd|abcd|abcdabcd

cval: Float or Int.

Value used for points outside the boundaries

when `fill_mode = "constant"`.

horizontal_flip: Boolean. Randomly flip inputs horizontally.

vertical_flip: Boolean. Randomly flip inputs vertically.

rescale: rescaling factor. Defaults to None.

If None or 0, no rescaling is applied,

otherwise we multiply the data by the value provided

(after applying all other transformations).

preprocessing_function: function that will be implied on each input.

The function will run after the image is resized and augmented.

The function should take one argument:

one image (Numpy tensor with rank 3),

and should output a Numpy tensor with the same shape.

data_format: Image data format,

either "channels_first" or "channels_last".

"channels_last" mode means that the images should have shape

`(samples, height, width, channels)`,

"channels_first" mode means that the images should have shape

`(samples, channels, height, width)`.

It defaults to the `image_data_format` value found in your

Keras config file at `~/.keras/keras.json`.

```
        If you never set it, then it will be "channels_last".
    validation_split: Float. Fraction of images reserved for validation
        (strictly between 0 and 1).
"""

def __init__(self,
             featurewise_center=False,
             samplewise_center=False,
             featurewise_std_normalization=False,
             samplewise_std_normalization=False,
             zca_whitening=False,
             zca_epsilon=1e-6,
             rotation_range=0,
             width_shift_range=0.,
             height_shift_range=0.,
             brightness_range=None,
             shear_range=0.,
             zoom_range=0.,
             channel_shift_range=0.,
             fill_mode='nearest',
             cval=0.,
             horizontal_flip=False,
             vertical_flip=False,
```

```
        rescale=None,
        preprocessing_function=None,
        data_format=None,
        validation_split=0.0):
if data_format is None:
    data_format = backend.image_data_format()
self.featurewise_center = featurewise_center
self.samplewise_center = samplewise_center
self.featurewise_std_normalization = featurewise_std_normalization
self.samplewise_std_normalization = samplewise_std_normalization
self.zca_whitening = zca_whitening
self.zca_epsilon = zca_epsilon
self.rotation_range = rotation_range
self.width_shift_range = width_shift_range
self.height_shift_range = height_shift_range
self.brightness_range = brightness_range
self.shear_range = shear_range
self.zoom_range = zoom_range
self.channel_shift_range = channel_shift_range
self.fill_mode = fill_mode
self.cval = cval
self.horizontal_flip = horizontal_flip
self.vertical_flip = vertical_flip
```

```
self.rescale = rescale
self.preprocessing_function = preprocessing_function


if data_format not in {'channels_last', 'channels_first'}:
    raise ValueError(
        '`data_format` should be `"channels_last"` '
        '(channel after row and column) or '
        '`"channels_first"` (channel before row and column). '
        'Received: %s' % data_format)
self.data_format = data_format
if data_format == 'channels_first':
    self.channel_axis = 1
    self.row_axis = 2
    self.col_axis = 3
if data_format == 'channels_last':
    self.channel_axis = 3
    self.row_axis = 1
    self.col_axis = 2
if validation_split and not 0 < validation_split < 1:
    raise ValueError(
        '`validation_split` must be strictly between 0 and 1. '
        ' Received: %s' % validation_split)
self._validation_split = validation_split
```

```
self.mean = None
self.std = None
self.principal_components = None


if np.isscalar(zoom_range):
    self.zoom_range = [1 - zoom_range, 1 + zoom_range]
elif len(zoom_range) == 2:
    self.zoom_range = [zoom_range[0], zoom_range[1]]
else:
    raise ValueError('`zoom_range` should be a float or '
                     'a tuple or list of two floats. '
                     'Received: %s' % (zoom_range,))
if zca_whitening:
    if not featurewise_center:
        self.featurewise_center = True
        warnings.warn('This ImageDataGenerator specifies '
                      '`zca_whitening`, which overrides '
                      'setting of `featurewise_center`.')
    if featurewise_std_normalization:
        self.featurewise_std_normalization = False
        warnings.warn('This ImageDataGenerator specifies '
                      '`zca_whitening` '
```

```
                    'which overrides setting of'
                    '`featurewise_std_normalization`.')
    if featurewise_std_normalization:
        if not featurewise_center:
            self.featurewise_center = True
            warnings.warn('This ImageDataGenerator specifies '
                    '`featurewise_std_normalization`, '
                    'which overrides setting of '
                    '`featurewise_center`.')
    if samplewise_std_normalization:
        if not samplewise_center:
            self.samplewise_center = True
            warnings.warn('This ImageDataGenerator specifies '
                    '`samplewise_std_normalization`, '
                    'which overrides setting of '
                    '`samplewise_center`.')

def flow(self, x,
         y=None, batch_size=32, shuffle=True,
         sample_weight=None, seed=None,
         save_to_dir=None, save_prefix='', save_format='png', subset=None):
    """Takes data & label arrays, generates batches of augmented data.
```

```
# Arguments
    x: Input data. Numpy array of rank 4 or a tuple.
        If tuple, the first element
        should contain the images and the second element
        another numpy array or a list of numpy arrays
        that gets passed to the output
        without any modifications.
        Can be used to feed the model miscellaneous data
        along with the images.
        In case of grayscale data, the channels axis of the image array
        should have value 1, in case
        of RGB data, it should have value 3, and in case
        of RGBA data, it should have value 4.
    y: Labels.
    batch_size: Int (default: 32).
    shuffle: Boolean (default: True).
    sample_weight: Sample weights.
    seed: Int (default: None).
    save_to_dir: None or str (default: None).
        This allows you to optionally specify a directory
        to which to save the augmented pictures being generated
        (useful for visualizing what you are doing).
    save_prefix: Str (default: `''`).
```

```
        Prefix to use for filenames of saved pictures
        (only relevant if `save_to_dir` is set).
        save_format: one of "png", "jpeg"
        (only relevant if `save_to_dir` is set). Default: "png".
    subset: Subset of data (`"training"` or `"validation"`) if
        `validation_split` is set in `ImageDataGenerator`.

# Returns
    An `Iterator` yielding tuples of `(x, y)`
        where `x` is a numpy array of image data
        (in the case of a single image input) or a list
        of numpy arrays (in the case with
        additional inputs) and `y` is a numpy array
        of corresponding labels. If 'sample_weight' is not None,
        the yielded tuples are of the form `(x, y, sample_weight)`.
        If `y` is None, only the numpy array `x` is returned.
"""
return NumpyArrayIterator(
    x, y, self,
    batch_size=batch_size,
    shuffle=shuffle,
    sample_weight=sample_weight,
    seed=seed,
```

```
        data_format=self.data_format,
        save_to_dir=save_to_dir,
        save_prefix=save_prefix,
        save_format=save_format,
        subset=subset)

def flow_from_directory(self, directory,
                        target_size=(256, 256), color_mode='rgb',
                        classes=None, class_mode='categorical',
                        batch_size=32, shuffle=True, seed=None,
                        save_to_dir=None,
                        save_prefix='',
                        save_format='png',
                        follow_links=False,
                        subset=None,
                        interpolation='nearest'):
    """Takes the path to a directory & generates batches of augmented data.

    # Arguments
        directory: Path to the target directory.
            It should contain one subdirectory per class.
            Any PNG, JPG, BMP, PPM or TIF images
            inside each of the subdirectories directory tree
```

```
    will be included in the generator.
    See [this script](
    https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d)
    for more details.
target_size: Tuple of integers `(height, width)`,
    default: `(256, 256)`.
    The dimensions to which all images found will be resized.
color_mode: One of "grayscale", "rbg", "rgba". Default: "rgb".
    Whether the images will be converted to
    have 1, 3, or 4 channels.
classes: Optional list of class subdirectories
    (e.g. `['dogs', 'cats']`). Default: None.
    If not provided, the list of classes will be automatically
    inferred from the subdirectory names/structure
    under `directory`, where each subdirectory will
    be treated as a different class
    (and the order of the classes, which will map to the label
    indices, will be alphanumeric).
    The dictionary containing the mapping from class names to class
    indices can be obtained via the attribute `class_indices`.
class_mode: One of "categorical", "binary", "sparse",
    "input", or None. Default: "categorical".
    Determines the type of label arrays that are returned:
```

- "categorical" will be 2D one-hot encoded labels,
- "binary" will be 1D binary labels,
  "sparse" will be 1D integer labels,
- "input" will be images identical
  to input images (mainly used to work with autoencoders).
- If None, no labels are returned
  (the generator will only yield batches of image data,
  which is useful to use with `model.predict_generator()`,
  `model.evaluate_generator()`, etc.).
  Please note that in case of class_mode None,
  the data still needs to reside in a subdirectory
  of `directory` for it to work correctly.

batch_size: Size of the batches of data (default: 32).

shuffle: Whether to shuffle the data (default: True)

seed: Optional random seed for shuffling and transformations.

save_to_dir: None or str (default: None).
  This allows you to optionally specify
  a directory to which to save
  the augmented pictures being generated
  (useful for visualizing what you are doing).

save_prefix: Str. Prefix to use for filenames of saved pictures
  (only relevant if `save_to_dir` is set).

save_format: One of "png", "jpeg"

```
        (only relevant if `save_to_dir` is set). Default: "png".
    follow_links: Whether to follow symlinks inside
        class subdirectories (default: False).
    subset: Subset of data (`"training"` or `"validation"`) if
        `validation_split` is set in `ImageDataGenerator`.
    interpolation: Interpolation method used to
        resample the image if the
        target size is different from that of the loaded image.
        Supported methods are `"nearest"`, `"bilinear"`,
        and `"bicubic"`.
        If PIL version 1.1.3 or newer is installed, `"lanczos"` is also
        supported. If PIL version 3.4.0 or newer is installed,
        `"box"` and `"hamming"` are also supported.
        By default, `"nearest"` is used.

# Returns
    A `DirectoryIterator` yielding tuples of `(x, y)`
        where `x` is a numpy array containing a batch
        of images with shape `(batch_size, *target_size, channels)`
        and `y` is a numpy array of corresponding labels.
"""
return DirectoryIterator(
    directory, self,
```

```
        target_size=target_size, color_mode=color_mode,
        classes=classes, class_mode=class_mode,
        data_format=self.data_format,
        batch_size=batch_size, shuffle=shuffle, seed=seed,
        save_to_dir=save_to_dir,
        save_prefix=save_prefix,
        save_format=save_format,
        follow_links=follow_links,
        subset=subset,
        interpolation=interpolation)

    def standardize(self, x):
        """Applies the normalization configuration to a batch of inputs.

        # Arguments
            x: Batch of inputs to be normalized.

        # Returns
            The inputs, normalized.
        """
        if self.preprocessing_function:
            x = self.preprocessing_function(x)
        if self.rescale:
```

```
    x *= self.rescale
if self.samplewise_center:
    x -= np.mean(x, keepdims=True)
if self.samplewise_std_normalization:
    x /= (np.std(x, keepdims=True) + backend.epsilon())


if self.featurewise_center:
    if self.mean is not None:
        x -= self.mean
    else:
        warnings.warn('This ImageDataGenerator specifies '
                      '`featurewise_center`, but it hasn\'t '
                      'been fit on any training data. Fit it '
                      'first by calling `.fit(numpy_data)`.')
if self.featurewise_std_normalization:
    if self.std is not None:
        x /= (self.std + backend.epsilon())
    else:
        warnings.warn('This ImageDataGenerator specifies '
                      '`featurewise_std_normalization`, '
                      'but it hasn\'t '
                      'been fit on any training data. Fit it '
                      'first by calling `.fit(numpy_data)`.')
```

```
    if self.zca_whitening:
        if self.principal_components is not None:
            flatx = np.reshape(x, (-1, np.prod(x.shape[-3:])))
            whitex = np.dot(flatx, self.principal_components)
            x = np.reshape(whitex, x.shape)
        else:
            warnings.warn('This ImageDataGenerator specifies '
                          '`zca_whitening`, but it hasn\'t '
                          'been fit on any training data. Fit it '
                          'first by calling `.fit(numpy_data)`.')
    return x


def get_random_transform(self, img_shape, seed=None):
    """Generates random parameters for a transformation.

    # Arguments
        seed: Random seed.
        img_shape: Tuple of integers.
            Shape of the image that is transformed.

    # Returns
        A dictionary containing randomly chosen parameters describing the
        transformation.
```

```
"""
img_row_axis = self.row_axis - 1
img_col_axis = self.col_axis - 1

if seed is not None:
    np.random.seed(seed)

if self.rotation_range:
    theta = np.random.uniform(
        -self.rotation_range,
        self.rotation_range)
else:
    theta = 0

if self.height_shift_range:
    try:  # 1-D array-like or int
        tx = np.random.choice(self.height_shift_range)
        tx *= np.random.choice([-1, 1])
    except ValueError:  # floating point
        tx = np.random.uniform(-self.height_shift_range,
                               self.height_shift_range)
    if np.max(self.height_shift_range) < 1:
        tx *= img_shape[img_row_axis]
```

```
else:
    tx = 0

if self.width_shift_range:
    try:  # 1-D array-like or int
        ty = np.random.choice(self.width_shift_range)
        ty *= np.random.choice([-1, 1])
    except ValueError:  # floating point
        ty = np.random.uniform(-self.width_shift_range,
                               self.width_shift_range)
    if np.max(self.width_shift_range) < 1:
        ty *= img_shape[img_col_axis]
else:
    ty = 0

if self.shear_range:
    shear = np.random.uniform(
        -self.shear_range,
        self.shear_range)
else:
    shear = 0

if self.zoom_range[0] == 1 and self.zoom_range[1] == 1:
```

```
    zx, zy = 1, 1
else:
    zx, zy = np.random.uniform(
        self.zoom_range[0],
        self.zoom_range[1],
        2)

flip_horizontal = (np.random.random() < 0.5) * self.horizontal_flip
flip_vertical = (np.random.random() < 0.5) * self.vertical_flip

channel_shift_intensity = None
if self.channel_shift_range != 0:
    channel_shift_intensity = np.random.uniform(-self.channel_shift_range,
                                                self.channel_shift_range)

brightness = None
if self.brightness_range is not None:
    if len(self.brightness_range) != 2:
        raise ValueError(
            '`brightness_range should be tuple or list of two floats. '
            'Received: %s' % (self.brightness_range,))
    brightness = np.random.uniform(self.brightness_range[0],
                                   self.brightness_range[1])
```

```
    transform_parameters = {'theta': theta,
                            'tx': tx,
                            'ty': ty,
                            'shear': shear,
                            'zx': zx,
                            'zy': zy,
                            'flip_horizontal': flip_horizontal,
                            'flip_vertical': flip_vertical,
                            'channel_shift_intensity': channel_shift_intensity,
                            'brightness': brightness}

    return transform_parameters


def apply_transform(self, x, transform_parameters):
    """Applies a transformation to an image according to given parameters.

    # Arguments
        x: 3D tensor, single image.
        transform_parameters: Dictionary with string - parameter pairs
            describing the transformation.
            Currently, the following parameters
            from the dictionary are used:
```

```
        - `'theta'`: Float. Rotation angle in degrees.
        - `'tx'`: Float. Shift in the x direction.
        - `'ty'`: Float. Shift in the y direction.
        - `'shear'`: Float. Shear angle in degrees.
        - `'zx'`: Float. Zoom in the x direction.
        - `'zy'`: Float. Zoom in the y direction.
        - `'flip_horizontal'`: Boolean. Horizontal flip.
        - `'flip_vertical'`: Boolean. Vertical flip.
        - `'channel_shift_intencity'`: Float. Channel shift intensity.
        - `'brightness'`: Float. Brightness shift intensity.

    # Returns
        A transformed version of the input (same shape).
    """
    # x is a single image, so it doesn't have image number at index 0
    img_row_axis = self.row_axis - 1
    img_col_axis = self.col_axis - 1
    img_channel_axis = self.channel_axis - 1

    x = apply_affine_transform(x, transform_parameters.get('theta', 0),
                               transform_parameters.get('tx', 0),
                               transform_parameters.get('ty', 0),
                               transform_parameters.get('shear', 0),
```

```
                        transform_parameters.get('zx', 1),
                        transform_parameters.get('zy', 1),
                        row_axis=img_row_axis, col_axis=img_col_axis,
                        channel_axis=img_channel_axis,
                        fill_mode=self.fill_mode, cval=self.cval)

    if transform_parameters.get('channel_shift_intensity') is not None:
        x = apply_channel_shift(x,
                                transform_parameters['channel_shift_intensity'],
                                img_channel_axis)

    if transform_parameters.get('flip_horizontal', False):
        x = flip_axis(x, img_col_axis)

    if transform_parameters.get('flip_vertical', False):
        x = flip_axis(x, img_row_axis)

    if transform_parameters.get('brightness') is not None:
        x = apply_brightness_shift(x, transform_parameters['brightness'])

    return x

def random_transform(self, x, seed=None):
```

```
    """Applies a random transformation to an image.

    # Arguments
        x: 3D tensor, single image.
        seed: Random seed.

    # Returns
        A randomly transformed version of the input (same shape).
    """
    params = self.get_random_transform(x.shape, seed)
    return self.apply_transform(x, params)

def fit(self, x,
        augment=False,
        rounds=1,
        seed=None):
    """Fits the data generator to some sample data.

    This computes the internal data stats related to the
    data-dependent transformations, based on an array of sample data.

    Only required if `featurewise_center` or
    `featurewise_std_normalization` or `zca_whitening` are set to True.
```

```
# Arguments
    x: Sample data. Should have rank 4.
     In case of grayscale data,
     the channels axis should have value 1, in case
     of RGB data, it should have value 3, and in case
     of RGBA data, it should have value 4.
    augment: Boolean (default: False).
        Whether to fit on randomly augmented samples.
    rounds: Int (default: 1).
        If using data augmentation (`augment=True`),
        this is how many augmentation passes over the data to use.
    seed: Int (default: None). Random seed.
"""
 x = np.asarray(x, dtype=backend.floatx())
 if x.ndim != 4:
    raise ValueError('Input to `.fit()` should have rank 4. '
                     'Got array with shape: ' + str(x.shape))
 if x.shape[self.channel_axis] not in {1, 3, 4}:
    warnings.warn(
        'Expected input to be images (as Numpy array) '
        'following the data format convention "' +
        self.data_format + '" (channels on axis ' +
```

```
        str(self.channel_axis) + '), i.e. expected '
                             'either 1, 3 or 4 channels on axis ' +
        str(self.channel_axis) + '. '
                             'However, it was passed an array with shape ' +
        str(x.shape) + ' (' + str(x.shape[self.channel_axis]) +
        ' channels).')

if seed is not None:
    np.random.seed(seed)

x = np.copy(x)
if augment:
    ax = np.zeros(
        tuple([rounds * x.shape[0]] + list(x.shape)[1:]),
        dtype=backend.floatx())
    for r in range(rounds):
        for i in range(x.shape[0]):
            ax[i + r * x.shape[0]] = self.random_transform(x[i])
    x = ax

if self.featurewise_center:
    self.mean = np.mean(x, axis=(0, self.row_axis, self.col_axis))
    broadcast_shape = [1, 1, 1]
```

```
    broadcast_shape[self.channel_axis - 1] = x.shape[self.channel_axis]
    self.mean = np.reshape(self.mean, broadcast_shape)
    x -= self.mean

if self.featurewise_std_normalization:
    self.std = np.std(x, axis=(0, self.row_axis, self.col_axis))
    broadcast_shape = [1, 1, 1]
    broadcast_shape[self.channel_axis - 1] = x.shape[self.channel_axis]
    self.std = np.reshape(self.std, broadcast_shape)
    x /= (self.std + backend.epsilon())

if self.zca_whitening:
    if scipy is None:
        raise ImportError('Using zca_whitening requires SciPy. '
                          'Install SciPy.')
    flat_x = np.reshape(
        x, (x.shape[0], x.shape[1] * x.shape[2] * x.shape[3]))
    sigma = np.dot(flat_x.T, flat_x) / flat_x.shape[0]
    u, s, _ = scipy.linalg.svd(sigma)
    s_inv = 1. / np.sqrt(s[np.newaxis] + self.zca_epsilon)
    self.principal_components = (u * s_inv).dot(u.T)
```

```
class Iterator(deepnet_utils.Sequence):
    """Base class for image data iterators.

    Every `Iterator` must implement the `_get_batches_of_transformed_samples`
    method.

    # Arguments
        n: Integer, total number of samples in the dataset to loop over.
        batch_size: Integer, size of a batch.
        shuffle: Boolean, whether to shuffle the data between epochs.
        seed: Random seeding for data shuffling.
    """

    def __init__(self, n, batch_size, shuffle, seed):
        self.n = n
        self.batch_size = batch_size
        self.seed = seed
        self.shuffle = shuffle
        self.batch_index = 0
        self.total_batches_seen = 0
        self.lock = threading.Lock()
        self.index_array = None
        self.index_generator = self._flow_index()
```

```
def _set_index_array(self):
    self.index_array = np.arange(self.n)
    if self.shuffle:
        self.index_array = np.random.permutation(self.n)


def __getitem__(self, idx):
    if idx >= len(self):
        raise ValueError('Asked to retrieve element {idx}, '
                         'but the Sequence '
                         'has length {length}'.format(idx=idx,
                                                      length=len(self)))
    if self.seed is not None:
        np.random.seed(self.seed + self.total_batches_seen)
    self.total_batches_seen += 1
    if self.index_array is None:
        self._set_index_array()
    index_array = self.index_array[self.batch_size * idx:
                                   self.batch_size * (idx + 1)]
    return self._get_batches_of_transformed_samples(index_array)


def __len__(self):
    return (self.n + self.batch_size - 1) // self.batch_size  # round up
```

```
def on_epoch_end(self):
    self._set_index_array()

def reset(self):
    self.batch_index = 0

def _flow_index(self):
    # Ensure self.batch_index is 0.
    self.reset()
    while 1:
        if self.seed is not None:
            np.random.seed(self.seed + self.total_batches_seen)
        if self.batch_index == 0:
            self._set_index_array()

        current_index = (self.batch_index * self.batch_size) % self.n
        if self.n > current_index + self.batch_size:
            self.batch_index += 1
        else:
            self.batch_index = 0
        self.total_batches_seen += 1
        yield self.index_array[current_index:
```

```
                current_index + self.batch_size]

    def __iter__(self):
        # Needed if we want to do something like:
        # for x, y in data_gen.flow(...):
        return self

    def __next__(self, *args, **kwargs):
        return self.next(*args, **kwargs)

    def _get_batches_of_transformed_samples(self, index_array):
        """Gets a batch of transformed samples.

        # Arguments
            index_array: Array of sample indices to include in batch.

        # Returns
            A batch of transformed samples.
        """
        raise NotImplementedError


class NumpyArrayIterator(Iterator):
```

"""Iterator yielding data from a Numpy array.

Arguments
    x: Numpy array of input data or tuple.
        If tuple, the second elements is either
        another numpy array or a list of numpy arrays,
        each of which gets passed
        through as an output without any modifications.
    y: Numpy array of targets data.
    image_data_generator: Instance of `ImageDataGenerator`
        to use for random transformations and normalization.
    batch_size: Integer, size of a batch.
    shuffle: Boolean, whether to shuffle the data between epochs.
    sample_weight: Numpy array of sample weights.
    seed: Random seed for data shuffling.
    data_format: String, one of `channels_first`, `channels_last`.
    save_to_dir: Optional directory where to save the pictures
        being yielded, in a viewable format. This is useful
        for visualizing the random transformations being
        applied, for debugging purposes.
    save_prefix: String prefix to use for saving sample
        images (if `save_to_dir` is set).
    save_format: Format to use for saving sample images

```
        (if `save_to_dir` is set).
    subset: Subset of data (`"training"` or `"validation"`) if
        validation_split is set in ImageDataGenerator.
"""

def __init__(self, x, y, image_data_generator,
             batch_size=32, shuffle=False, sample_weight=None,
             seed=None, data_format=None,
             save_to_dir=None, save_prefix='', save_format='png',
             subset=None):
    if (type(x) is tuple) or (type(x) is list):
        if type(x[1]) is not list:
            x_misc = [np.asarray(x[1])]
        else:
            x_misc = [np.asarray(xx) for xx in x[1]]
        x = x[0]
        for xx in x_misc:
            if len(x) != len(xx):
                raise ValueError(
                    'All of the arrays in `x` '
                    'should have the same length. '
                    'Found a pair with: len(x[0]) = %s, len(x[?]) = %s' %
                    (len(x), len(xx)))
```

```
else:
    x_misc = []


if y is not None and len(x) != len(y):
    raise ValueError('`x` (images tensor) and `y` (labels) '
                     'should have the same length. '
                     'Found: x.shape = %s, y.shape = %s' %
                     (np.asarray(x).shape, np.asarray(y).shape))
if sample_weight is not None and len(x) != len(sample_weight):
    raise ValueError('`x` (images tensor) and `sample_weight` '
                     'should have the same length. '
                     'Found: x.shape = %s, sample_weight.shape = %s' %
                     (np.asarray(x).shape, np.asarray(sample_weight).shape))
if subset is not None:
    if subset not in {'training', 'validation'}:
        raise ValueError('Invalid subset name:', subset,
                         '; expected "training" or "validation".')
    split_idx = int(len(x) * image_data_generator._validation_split)
    if subset == 'validation':
        x = x[:split_idx]
        x_misc = [np.asarray(xx[:split_idx]) for xx in x_misc]
        if y is not None:
            y = y[:split_idx]
```

```
            else:
                x = x[split_idx:]
                x_misc = [np.asarray(xx[split_idx:]) for xx in x_misc]
                if y is not None:
                    y = y[split_idx:]
        if data_format is None:
            data_format = backend.image_data_format()
        self.x = np.asarray(x, dtype=backend.floatx())
        self.x_misc = x_misc
        if self.x.ndim != 4:
            raise ValueError('Input data in `NumpyArrayIterator` '
                             'should have rank 4. You passed an array '
                             'with shape', self.x.shape)
        channels_axis = 3 if data_format == 'channels_last' else 1
        if self.x.shape[channels_axis] not in {1, 3, 4}:
            warnings.warn('NumpyArrayIterator is set to use the '
                          'data format convention "' + data_format + '" '
                                                                    '(channels on axis ' +
str(channels_axis) +
                          '), i.e. expected either 1, 3, or 4 '
                          'channels on axis ' + str(channels_axis) + '. '
                                                                     'However, it was passed an array with
shape ' +
```

```
                str(self.x.shape) + ' (' +
                str(self.x.shape[channels_axis]) + ' channels).')
    if y is not None:
        self.y = np.asarray(y)
    else:
        self.y = None
    if sample_weight is not None:
        self.sample_weight = np.asarray(sample_weight)
    else:
        self.sample_weight = None
    self.image_data_generator = image_data_generator
    self.data_format = data_format
    self.save_to_dir = save_to_dir
    self.save_prefix = save_prefix
    self.save_format = save_format
    super(NumpyArrayIterator, self).__init__(x.shape[0],
                                             batch_size,
                                             shuffle,
                                             seed)

def _get_batches_of_transformed_samples(self, index_array):
    batch_x = np.zeros(tuple([len(index_array)] + list(self.x.shape)[1:]),
                       dtype=backend.floatx())
```

```
for i, j in enumerate(index_array):
    x = self.x[j]
    params = self.image_data_generator.get_random_transform(x.shape)
    x = self.image_data_generator.apply_transform(
        x.astype(backend.floatx()), params)
    x = self.image_data_generator.standardize(x)
    batch_x[i] = x

if self.save_to_dir:
    for i, j in enumerate(index_array):
        img = array_to_img(batch_x[i], self.data_format, scale=True)
        fname = '{prefix}_{index}_{hash}.{format}'.format(
            prefix=self.save_prefix,
            index=j,
            hash=np.random.randint(1e4),
            format=self.save_format)
        img.save(os.path.join(self.save_to_dir, fname))
batch_x_miscs = [xx[index_array] for xx in self.x_misc]
output = (batch_x if batch_x_miscs == []
          else [batch_x] + batch_x_miscs,)
if self.y is None:
    return output[0]
output += (self.y[index_array],)
```

```
        if self.sample_weight is not None:
            output += (self.sample_weight[index_array],)
        return output

    def next(self):
        """For python 2.x.

        # Returns
            The next batch.
        """
        # Keeps under lock only the mechanism which advances
        # the indexing of each batch.
        with self.lock:
            index_array = next(self.index_generator)
        # The transformation of images is not under thread lock
        # so it can be done in parallel
        return self._get_batches_of_transformed_samples(index_array)


def _iter_valid_files(directory, white_list_formats, follow_links):
    """Iterates on files with extension in `white_list_formats` contained in
`directory`.
```

```
# Arguments
    directory: Absolute path to the directory
        containing files to be counted
    white_list_formats: Set of strings containing allowed extensions for
        the files to be counted.
    follow_links: Boolean.

# Yields
    Tuple of (root, filename) with extension in `white_list_formats`.
"""

def _recursive_list(subpath):
    return sorted(os.walk(subpath, followlinks=follow_links),
                  key=lambda x: x[0])

for root, _, files in _recursive_list(directory):
    for fname in sorted(files):
        for extension in white_list_formats:
            if fname.lower().endswith('.tiff'):
                warnings.warn('Using \'.tiff\' files with multiple bands '
                              'will cause distortion. '
                              'Please verify your output.')
            if fname.lower().endswith('.' + extension):
```

```
                    yield root, fname


def _count_valid_files_in_directory(directory,
                                    white_list_formats,
                                    split,
                                    follow_links):
    """Counts files with extension in `white_list_formats` contained in `directory`.

    # Arguments
        directory: absolute path to the directory
            containing files to be counted
        white_list_formats: set of strings containing allowed extensions for
            the files to be counted.
        split: tuple of floats (e.g. `(0.2, 0.6)`) to only take into
            account a certain fraction of files in each directory.
            E.g.: `segment=(0.6, 1.0)` would only account for last 40 percent
            of images in each directory.
        follow_links: boolean.

    # Returns
        the count of files with extension in `white_list_formats` contained in
        the directory.
```

```
    """
    num_files = len(list(
        _iter_valid_files(directory, white_list_formats, follow_links)))
    if split:
        start, stop = int(split[0] * num_files), int(split[1] * num_files)
    else:
        start, stop = 0, num_files
    return stop - start


def _list_valid_filenames_in_directory(directory, white_list_formats, split,
                                       class_indices, follow_links):
    """Lists paths of files in `subdir` with extensions in `white_list_formats`.

    # Arguments
        directory: absolute path to a directory containing the files to list.
            The directory name is used as class label
            and must be a key of `class_indices`.
        white_list_formats: set of strings containing allowed extensions for
            the files to be counted.
        split: tuple of floats (e.g. `(0.2, 0.6)`) to only take into
            account a certain fraction of files in each directory.
            E.g.: `segment=(0.6, 1.0)` would only account for last 40 percent
```

```
        of images in each directory.
    class_indices: dictionary mapping a class name to its index.
    follow_links: boolean.

# Returns
    classes: a list of class indices
    filenames: the path of valid files in `directory`, relative from
        `directory`'s parent (e.g., if `directory` is "dataset/class1",
        the filenames will be
        `["class1/file1.jpg", "class1/file2.jpg", ...]`).
"""
dirname = os.path.basename(directory)
if split:
    num_files = len(list(
        _iter_valid_files(directory, white_list_formats, follow_links)))
    start, stop = int(split[0] * num_files), int(split[1] * num_files)
    valid_files = list(
        _iter_valid_files(
            directory, white_list_formats, follow_links))[start: stop]
else:
    valid_files = _iter_valid_files(
        directory, white_list_formats, follow_links)
```

```
    classes = []
    filenames = []
    for root, fname in valid_files:
        classes.append(class_indices[dirname])
        absolute_path = os.path.join(root, fname)
        relative_path = os.path.join(
            dirname, os.path.relpath(absolute_path, directory))
        filenames.append(relative_path)

    return classes, filenames


class DirectoryIterator(Iterator):
    """Iterator capable of reading images from a directory on disk.

    # Arguments
        directory: Path to the directory to read images from.
            Each subdirectory in this directory will be
            considered to contain images from one class,
            or alternatively you could specify class subdirectories
            via the `classes` argument.
        image_data_generator: Instance of `ImageDataGenerator`
            to use for random transformations and normalization.
```

target_size: tuple of integers, dimensions to resize input images to.

color_mode: One of `"rgb"`, `"rgba"`, `"grayscale"`.

Color mode to read images.

classes: Optional list of strings, names of subdirectories

containing images from each class (e.g. `["dogs", "cats"]`).

It will be computed automatically if not set.

class_mode: Mode for yielding the targets:

`"binary"`: binary targets (if there are only two classes),

`"categorical"`: categorical targets,

`"sparse"`: integer targets,

`"input"`: targets are images identical to input images (mainly

used to work with autoencoders),

`None`: no targets get yielded (only input images are yielded).

batch_size: Integer, size of a batch.

shuffle: Boolean, whether to shuffle the data between epochs.

seed: Random seed for data shuffling.

data_format: String, one of `channels_first`, `channels_last`.

save_to_dir: Optional directory where to save the pictures

being yielded, in a viewable format. This is useful

for visualizing the random transformations being

applied, for debugging purposes.

save_prefix: String prefix to use for saving sample

images (if `save_to_dir` is set).

```
    save_format: Format to use for saving sample images
        (if `save_to_dir` is set).
    subset: Subset of data (`"training"` or `"validation"`) if
        validation_split is set in ImageDataGenerator.
    interpolation: Interpolation method used to resample the image if the
        target size is different from that of the loaded image.
        Supported methods are "nearest", "bilinear", and "bicubic".
        If PIL version 1.1.3 or newer is installed, "lanczos" is also
        supported. If PIL version 3.4.0 or newer is installed, "box" and
        "hamming" are also supported. By default, "nearest" is used.
"""

def __init__(self, directory, image_data_generator,
             target_size=(256, 256), color_mode='rgb',
             classes=None, class_mode='categorical',
             batch_size=32, shuffle=True, seed=None,
             data_format=None,
             save_to_dir=None, save_prefix='', save_format='png',
             follow_links=False,
             subset=None,
             interpolation='nearest'):
    if data_format is None:
        data_format = backend.image_data_format()
```

```
self.directory = directory
self.image_data_generator = image_data_generator
self.target_size = tuple(target_size)
if color_mode not in {'rgb', 'rgba', 'grayscale'}:
    raise ValueError('Invalid color mode:', color_mode,
                     '; expected "rgb", "rgba", or "grayscale".')
self.color_mode = color_mode
self.data_format = data_format
if self.color_mode == 'rgba':
    if self.data_format == 'channels_last':
        self.image_shape = self.target_size + (4,)
    else:
        self.image_shape = (4,) + self.target_size
elif self.color_mode == 'rgb':
    if self.data_format == 'channels_last':
        self.image_shape = self.target_size + (3,)
    else:
        self.image_shape = (3,) + self.target_size
else:
    if self.data_format == 'channels_last':
        self.image_shape = self.target_size + (1,)
    else:
        self.image_shape = (1,) + self.target_size
```

```
self.classes = classes
if class_mode not in {'categorical', 'binary', 'sparse',
                      'input', None}:
    raise ValueError('Invalid class_mode:', class_mode,
                     '; expected one of "categorical", '
                     '"binary", "sparse", "input"'
                     ' or None.')
self.class_mode = class_mode
self.save_to_dir = save_to_dir
self.save_prefix = save_prefix
self.save_format = save_format
self.interpolation = interpolation

if subset is not None:
    validation_split = self.image_data_generator._validation_split
    if subset == 'validation':
        split = (0, validation_split)
    elif subset == 'training':
        split = (validation_split, 1)
    else:
        raise ValueError('Invalid subset name: ', subset,
                         '; expected "training" or "validation"')
else:
```

```
    split = None
self.subset = subset

white_list_formats = {'png', 'jpg', 'jpeg', 'bmp',
                      'ppm', 'tif', 'tiff'}
# First, count the number of samples and classes.
self.samples = 0

if not classes:
    classes = []
    for subdir in sorted(os.listdir(directory)):
        if os.path.isdir(os.path.join(directory, subdir)):
            classes.append(subdir)
self.num_classes = len(classes)
self.class_indices = dict(zip(classes, range(len(classes))))

pool = multiprocessing.pool.ThreadPool()
function_partial = partial(_count_valid_files_in_directory,
                           white_list_formats=white_list_formats,
                           follow_links=follow_links,
                           split=split)
self.samples = sum(pool.map(function_partial,
                            (os.path.join(directory, subdir)
```

```
                    for subdir in classes)))

print('Found %d images belonging to %d classes.' %
      (self.samples, self.num_classes))

# Second, build an index of the images
# in the different class subfolders.
results = []
self.filenames = []
self.classes = np.zeros((self.samples,), dtype='int32')
i = 0
for dirpath in (os.path.join(directory, subdir) for subdir in classes):
    results.append(
        pool.apply_async(_list_valid_filenames_in_directory,
                         (dirpath, white_list_formats, split,
                          self.class_indices, follow_links)))
for res in results:
    classes, filenames = res.get()
    self.classes[i:i + len(classes)] = classes
    self.filenames += filenames
    i += len(classes)

pool.close()
```

```
    pool.join()
    super(DirectoryIterator, self).__init__(self.samples,
                                            batch_size,
                                            shuffle,
                                            seed)

def _get_batches_of_transformed_samples(self, index_array):
    batch_x = np.zeros(
        (len(index_array),) + self.image_shape,
        dtype=backend.floatx())
    # build batch of image data
    for i, j in enumerate(index_array):
        fname = self.filenames[j]
        img = load_img(os.path.join(self.directory, fname),
                       color_mode=self.color_mode,
                       target_size=self.target_size,
                       interpolation=self.interpolation)
        x = img_to_array(img, data_format=self.data_format)
        # Pillow images should be closed after `load_img`,
        # but not PIL images.
        if hasattr(img, 'close'):
            img.close()
        params = self.image_data_generator.get_random_transform(x.shape)
```

```
    x = self.image_data_generator.apply_transform(x, params)
    x = self.image_data_generator.standardize(x)
    batch_x[i] = x
# optionally save augmented images to disk for debugging purposes
if self.save_to_dir:
    for i, j in enumerate(index_array):
        img = array_to_img(batch_x[i], self.data_format, scale=True)
        fname = '{prefix}_{index}_{hash}.{format}'.format(
            prefix=self.save_prefix,
            index=j,
            hash=np.random.randint(1e7),
            format=self.save_format)
        img.save(os.path.join(self.save_to_dir, fname))
# build batch of labels
if self.class_mode == 'input':
    batch_y = batch_x.copy()
elif self.class_mode == 'sparse':
    batch_y = self.classes[index_array]
elif self.class_mode == 'binary':
    batch_y = self.classes[index_array].astype(backend.floatx())
elif self.class_mode == 'categorical':
    batch_y = np.zeros(
        (len(batch_x), self.num_classes),
```

```
            dtype=backend.floatx())
        for i, label in enumerate(self.classes[index_array]):
            batch_y[i, label] = 1.
    else:
        return batch_x
    return batch_x, batch_y


def next(self):
    """


    # Returns
        The next batch.
    """
    with self.lock:
        index_array = next(self.index_generator)
    # The transformation of images is not under thread lock
    # so it can be done in parallel
    return self._get_batches_of_transformed_samples(index_array)
```

4. training.py

```
"""Training-related part of the Keras core.
"""
from __future__ import absolute_import
from __future__ import division
```

```
from __future__ import print_function

import warnings
import copy
import numpy as np

from .network import Network
from .base_layer import Layer
from .training_utils import collect_metrics
from .training_utils import check_array_length_consistency
from .training_utils import check_loss_and_target_compatibility
from .training_utils import standardize_class_weights
from .training_utils import standardize_input_data
from .training_utils import standardize_sample_weights
from .training_utils import standardize_weights
from .training_utils import weighted_masked_objective
from . import training_arrays
from . import training_generator
from .. import backend as K
from face_ai.deepnet.function import losses, metrics as metrics_module,
optimizers
from ..utils.generic_utils import slice_arrays
from ..utils.generic_utils import to_list
```

```
from ..utils.generic_utils import unpack_singleton
from ..legacy import interfaces


class Model(Network):
    """The `Model` class adds training & evaluation routines to a `Network`.
    """

    def compile(self, optimizer,
                loss=None,
                metrics=None,
                loss_weights=None,
                sample_weight_mode=None,
                weighted_metrics=None,
                target_tensors=None,
                **kwargs):
        """Configures the model for training.

        # Arguments
            optimizer: String (name of optimizer) or optimizer instance.
                See [optimizers](/optimizers).
            loss: String (name of objective function) or objective function.
                See [losses](/losses).
```

If the model has multiple outputs, you can use a different loss
on each output by passing a dictionary or a list of losses.
The loss value that will be minimized by the model
will then be the sum of all individual losses.

metrics: List of metrics to be evaluated by the model
during training and testing.
Typically you will use `metrics=['accuracy']`.
To specify different metrics for different outputs of a
multi-output model, you could also pass a dictionary,
such as `metrics={'output_a': 'accuracy'}`.

loss_weights: Optional list or dictionary specifying scalar
coefficients (Python floats) to weight the loss contributions
of different model outputs.
The loss value that will be minimized by the model
will then be the *weighted sum* of all individual losses,
weighted by the `loss_weights` coefficients.
If a list, it is expected to have a 1:1 mapping
to the model's outputs. If a tensor, it is expected to map
output names (strings) to scalar coefficients.

sample_weight_mode: If you need to do timestep-wise
sample weighting (2D weights), set this to `"temporal"`.
`None` defaults to sample-wise weights (1D).
If the model has multiple outputs, you can use a different

```
        `sample_weight_mode` on each output by passing a
        dictionary or a list of modes.
    weighted_metrics: List of metrics to be evaluated and weighted
        by sample_weight or class_weight during training and testing.
    target_tensors: By default, Keras will create placeholders for the
        model's target, which will be fed with the target data during
        training. If instead you would like to use your own
        target tensors (in turn, Keras will not expect external
        Numpy data for these targets at training time), you
        can specify them via the `target_tensors` argument. It can be
        a single tensor (for a single-output model), a list of tensors,
        or a dict mapping output names to target tensors.
    **kwargs: When using the Theano/CNTK backends, these arguments
        are passed into `K.function`.
        When using the TensorFlow backend,
        these arguments are passed into `tf.Session.run`.

# Raises
    ValueError: In case of invalid arguments for
        `optimizer`, `loss`, `metrics` or `sample_weight_mode`.
"""
self.optimizer = optimizers.get(optimizer)
self.loss = loss or []
```

```
self.metrics = metrics or []
self.loss_weights = loss_weights
self.sample_weight_mode = sample_weight_mode
self.weighted_metrics = weighted_metrics

if not self.built:
    # Model is not compilable because
    # it does not know its number of inputs
    # and outputs, nor their shapes and names.
    # We will compile after the first
    # time the model gets called on training data.
    return
self._is_compiled = True

# Prepare loss functions.
if isinstance(loss, dict):
    for name in loss:
        if name not in self.output_names:
            raise ValueError('Unknown entry in loss '
                             'dictionary: "' + name + '". '
                             'Only expected the following keys: ' +
                             str(self.output_names))
    loss_functions = []
```

```
    for name in self.output_names:
        if name not in loss:
            warnings.warn('Output "' + name +
                          '" missing from loss dictionary. '
                          'We assume this was done on purpose, '
                          'and we will not be expecting '
                          'any data to be passed to "' + name +
                          '" during training.', stacklevel=2)
        loss_functions.append(losses.get(loss.get(name)))
elif isinstance(loss, list):
    if len(loss) != len(self.outputs):
        raise ValueError('When passing a list as loss, '
                         'it should have one entry per model outputs. '
                         'The model has ' + str(len(self.outputs)) +
                         ' outputs, but you passed loss=' +
                         str(loss))
    loss_functions = [losses.get(l) for l in loss]
else:
    loss_function = losses.get(loss)
    loss_functions = [loss_function for _ in range(len(self.outputs))]
self.loss_functions = loss_functions
weighted_losses = [
    weighted_masked_objective(fn) for fn in loss_functions]
```

```
skip_target_indices = []
skip_target_weighing_indices = []
self._feed_outputs = []
self._feed_output_names = []
self._feed_output_shapes = []
self._feed_loss_fns = []
for i in range(len(weighted_losses)):
    if weighted_losses[i] is None:
        skip_target_indices.append(i)
        skip_target_weighing_indices.append(i)

# Prepare output masks.
masks = self.compute_mask(self.inputs, mask=None)
if masks is None:
    masks = [None for _ in self.outputs]
masks = to_list(masks)

# Prepare loss weights.
if loss_weights is None:
    loss_weights_list = [1. for _ in range(len(self.outputs))]
elif isinstance(loss_weights, dict):
    for name in loss_weights:
        if name not in self.output_names:
```

```
            raise ValueError('Unknown entry in loss_weights '
                             'dictionary: "' + name + '". '
                             'Only expected the following keys: ' +
                             str(self.output_names))
    loss_weights_list = []
    for name in self.output_names:
      loss_weights_list.append(loss_weights.get(name, 1.))
elif isinstance(loss_weights, list):
    if len(loss_weights) != len(self.outputs):
      raise ValueError('When passing a list as loss_weights, '
                       'it should have one entry per model output. '
                       'The model has ' + str(len(self.outputs)) +
                       ' outputs, but you passed loss_weights=' +
                       str(loss_weights))
    loss_weights_list = loss_weights
else:
    raise TypeError('Could not interpret loss_weights argument: ' +
                    str(loss_weights) +
                    ' - expected a list of dicts.')

# Prepare targets of model.
self.targets = []
self._feed_targets = []
```

```
if target_tensors is not None:
    if isinstance(target_tensors, list):
        if len(target_tensors) != len(self.outputs):
            raise ValueError(
                'When passing a list as `target_tensors`, '
                'it should have one entry per model output. '
                'The model has ' + str(len(self.outputs)) +
                ' outputs, but you passed target_tensors=' +
                str(target_tensors))
    elif isinstance(target_tensors, dict):
        for name in target_tensors:
            if name not in self.output_names:
                raise ValueError('Unknown entry in `target_tensors` '
                                 'dictionary: "' + name + '". '
                                 'Only expected the following keys: ' +
                                 str(self.output_names))
        tmp_target_tensors = []
        for name in self.output_names:
            tmp_target_tensors.append(target_tensors.get(name, None))
        target_tensors = tmp_target_tensors
    else:
        raise TypeError('Expected `target_tensors` to be '
                        'a list or dict, but got:', target_tensors)
```

```
for i in range(len(self.outputs)):
    if i in skip_target_indices:
        self.targets.append(None)
    else:
        shape = K.int_shape(self.outputs[i])
        name = self.output_names[i]
        if target_tensors is not None:
            target = target_tensors[i]
        else:
            target = None
        if target is None or K.is_placeholder(target):
            if target is None:
                target = K.placeholder(
                    ndim=len(shape),
                    name=name + '_target',
                    sparse=K.is_sparse(self.outputs[i]),
                    dtype=K.dtype(self.outputs[i]))
            self._feed_targets.append(target)
            self._feed_outputs.append(self.outputs[i])
            self._feed_output_names.append(name)
            self._feed_output_shapes.append(shape)
            self._feed_loss_fns.append(self.loss_functions[i])
        else:
```

```
            skip_target_weighing_indices.append(i)
        self.targets.append(target)


# Prepare sample weights.
sample_weights = []
sample_weight_modes = []
if isinstance(sample_weight_mode, dict):
    for name in sample_weight_mode:
        if name not in self.output_names:
            raise ValueError('Unknown entry in '
                             'sample_weight_mode dictionary: "' +
                             name + '". '
                             'Only expected the following keys: ' +
                             str(self.output_names))
    for i, name in enumerate(self.output_names):
        if i in skip_target_weighing_indices:
            weight = None
            sample_weight_modes.append(None)
        else:
            if name not in sample_weight_mode:
                raise ValueError('Output "' + name +
                                 '" missing from sample_weight_modes '
                                 'dictionary')
```

```
            if sample_weight_mode.get(name) == 'temporal':
                weight = K.placeholder(ndim=2,
                                       name=name + '_sample_weights')
                sample_weight_modes.append('temporal')
            else:
                weight = K.placeholder(ndim=1,
                                       name=name + '_sample_weights')
                sample_weight_modes.append(None)
        sample_weights.append(weight)
    elif isinstance(sample_weight_mode, list):
        if len(sample_weight_mode) != len(self.outputs):
            raise ValueError('When passing a list as sample_weight_mode, '
                             'it should have one entry per model output. '
                             'The model has ' + str(len(self.outputs)) +
                             ' outputs, but you passed '
                             'sample_weight_mode=' +
                             str(sample_weight_mode))
        for i in range(len(self.output_names)):
            if i in skip_target_weighing_indices:
                weight = None
                sample_weight_modes.append(None)
            else:
                mode = sample_weight_mode[i]
```

```
            name = self.output_names[i]
            if mode == 'temporal':
                weight = K.placeholder(ndim=2,
                                       name=name + '_sample_weights')
                sample_weight_modes.append('temporal')
            else:
                weight = K.placeholder(ndim=1,
                                       name=name + '_sample_weights')
                sample_weight_modes.append(None)
            sample_weights.append(weight)
    else:
        for i, name in enumerate(self.output_names):
            if i in skip_target_weighing_indices:
                sample_weight_modes.append(None)
                sample_weights.append(None)
            else:
                if sample_weight_mode == 'temporal':
                    sample_weights.append(
                        K.placeholder(ndim=2,
                                      name=name + '_sample_weights'))
                    sample_weight_modes.append('temporal')
                else:
                    sample_weights.append(
```

```
                    K.placeholder(ndim=1,
                                  name=name + '_sample_weights'))
                sample_weight_modes.append(None)
        self.sample_weight_modes = sample_weight_modes
        self._feed_sample_weight_modes = []
        for i in range(len(self.outputs)):
            if i not in skip_target_weighing_indices:
                self._feed_sample_weight_modes.append(
                    self.sample_weight_modes[i])

        # Prepare metrics.
        self.metrics_names = ['loss']
        self.metrics_tensors = []

        # Compute total loss.
        total_loss = None
        with K.name_scope('loss'):
            for i in range(len(self.outputs)):
                if i in skip_target_indices:
                    continue
                y_true = self.targets[i]
                y_pred = self.outputs[i]
                weighted_loss = weighted_losses[i]
```

```
    sample_weight = sample_weights[i]
    mask = masks[i]
    loss_weight = loss_weights_list[i]
    with K.name_scope(self.output_names[i] + '_loss'):
      output_loss = weighted_loss(y_true, y_pred,
                                  sample_weight, mask)
    if len(self.outputs) > 1:
      self.metrics_tensors.append(output_loss)
      self.metrics_names.append(self.output_names[i] + '_loss')
    if total_loss is None:
      total_loss = loss_weight * output_loss
    else:
      total_loss += loss_weight * output_loss
  if total_loss is None:
    if not self.losses:
      raise ValueError('The model cannot be compiled '
                       'because it has no loss to optimize.')
    else:
      total_loss = 0.

  # Add regularization penalties
  # and other layer-specific losses.
  for loss_tensor in self.losses:
```

```
        total_loss += loss_tensor

# List of same size as output_names.
# contains tuples (metrics for output, names of metrics).
nested_metrics = collect_metrics(metrics, self.output_names)
nested_weighted_metrics = collect_metrics(weighted_metrics,
                                          self.output_names)
self.metrics_updates = []
self.stateful_metric_names = []
self.stateful_metric_functions = []

def handle_metrics(metrics, weights=None):
    metric_name_prefix = 'weighted_' if weights is not None else ''

    for metric in metrics:
        if metric in ('accuracy', 'acc', 'crossentropy', 'ce'):
            # custom handling of accuracy/crossentropy
            # (because of class mode duality)
            output_shape = K.int_shape(self.outputs[i])
            if (output_shape[-1] == 1 or
               self.loss_functions[i] == losses.binary_crossentropy):
                # case: binary accuracy/crossentropy
                if metric in ('accuracy', 'acc'):
```

```
                metric_fn = metrics_module.binary_accuracy
            elif metric in ('crossentropy', 'ce'):
                metric_fn = metrics_module.binary_crossentropy
        elif self.loss_functions[i] == losses.sparse_categorical_crossentropy:
            # case: categorical accuracy/crossentropy
            # with sparse targets
            if metric in ('accuracy', 'acc'):
                metric_fn = metrics_module.sparse_categorical_accuracy
            elif metric in ('crossentropy', 'ce'):
                metric_fn = metrics_module.sparse_categorical_crossentropy
        else:
            # case: categorical accuracy/crossentropy
            if metric in ('accuracy', 'acc'):
                metric_fn = metrics_module.categorical_accuracy
            elif metric in ('crossentropy', 'ce'):
                metric_fn = metrics_module.categorical_crossentropy
        if metric in ('accuracy', 'acc'):
                suffix = 'acc'
        elif metric in ('crossentropy', 'ce'):
                suffix = 'ce'
        weighted_metric_fn = weighted_masked_objective(metric_fn)
        metric_name = metric_name_prefix + suffix
    else:
```

```
    metric_fn = metrics_module.get(metric)
    weighted_metric_fn = weighted_masked_objective(metric_fn)
    # Get metric name as string
    if hasattr(metric_fn, 'name'):
      metric_name = metric_fn.name
    else:
      metric_name = metric_fn.__name__
    metric_name = metric_name_prefix + metric_name

with K.name_scope(metric_name):
    metric_result = weighted_metric_fn(y_true, y_pred,
                                       weights=weights,
                                       mask=masks[i])

# Append to self.metrics_names, self.metric_tensors,
# self.stateful_metric_names
if len(self.output_names) > 1:
    metric_name = self.output_names[i] + '_' + metric_name
# Dedupe name
j = 1
base_metric_name = metric_name
while metric_name in self.metrics_names:
    metric_name = base_metric_name + '_' + str(j)
```

```
            j += 1
        self.metrics_names.append(metric_name)
        self.metrics_tensors.append(metric_result)

        # Keep track of state updates created by
        # stateful metrics (i.e. metrics layers).
        if isinstance(metric_fn, Layer) and metric_fn.stateful:
            self.stateful_metric_names.append(metric_name)
            self.stateful_metric_functions.append(metric_fn)
            self.metrics_updates += metric_fn.updates
with K.name_scope('metrics'):
    for i in range(len(self.outputs)):
        if i in skip_target_indices:
            continue

        y_true = self.targets[i]
        y_pred = self.outputs[i]
        weights = sample_weights[i]
        output_metrics = nested_metrics[i]
        output_weighted_metrics = nested_weighted_metrics[i]
        handle_metrics(output_metrics)
        handle_metrics(output_weighted_metrics, weights=weights)
```

```
    # Prepare gradient updates and state updates.
    self.total_loss = total_loss
    self.sample_weights = sample_weights
    self._feed_sample_weights = []
    for i in range(len(self.sample_weights)):
      if i not in skip_target_weighing_indices:
        self._feed_sample_weights.append(sample_weights[i])

    # Functions for train, test and predict will
    # be compiled lazily when required.
    # This saves time when the user is not using all functions.
    self._function_kwargs = kwargs

    self.train_function = None
    self.test_function = None
    self.predict_function = None

    # Collected trainable weights, sorted in topological order.
    trainable_weights = self.trainable_weights
    self._collected_trainable_weights = trainable_weights

def _check_trainable_weights_consistency(self):
    """Check trainable weights count consistency.
```

```
    This will raise a warning if `trainable_weights` and
    `_collected_trainable_weights` are inconsistent (i.e. have different
    number of parameters).
    Inconsistency will typically arise when one modifies `model.trainable`
    without calling `model.compile` again.
    """
    if not hasattr(self, '_collected_trainable_weights'):
      return

    if (len(self.trainable_weights) !=
        len(self._collected_trainable_weights)):
      warnings.warn(UserWarning(
        'Discrepancy between trainable weights and collected trainable'
        ' weights, did you set `model.trainable` without calling'
        ' `model.compile` after ?'))

def _make_train_function(self):
    if not hasattr(self, 'train_function'):
      raise RuntimeError('You must compile your model before using it.')
    self._check_trainable_weights_consistency()
    if self.train_function is None:
      inputs = (self._feed_inputs +
```

```
                self._feed_targets +
                self._feed_sample_weights)
        if self._uses_dynamic_learning_phase():
            inputs += [K.learning_phase()]

        with K.name_scope('training'):
            with K.name_scope(self.optimizer.__class__.__name__):
                training_updates = self.optimizer.get_updates(
                    params=self._collected_trainable_weights,
                    loss=self.total_loss)
            updates = (self.updates +
                       training_updates +
                       self.metrics_updates)
            # Gets loss and metrics. Updates weights at each call.
            self.train_function = K.function(
                inputs,
                [self.total_loss] + self.metrics_tensors,
                updates=updates,
                name='train_function',
                **self._function_kwargs)

    def _make_test_function(self):
        if not hasattr(self, 'test_function'):
```

```
        raise RuntimeError('You must compile your model before using it.')
    if self.test_function is None:
        inputs = (self._feed_inputs +
                  self._feed_targets +
                  self._feed_sample_weights)
        if self._uses_dynamic_learning_phase():
            inputs += [K.learning_phase()]
        # Return loss and metrics, no gradient updates.
        # Does update the network states.
        self.test_function = K.function(
            inputs,
            [self.total_loss] + self.metrics_tensors,
            updates=self.state_updates + self.metrics_updates,
            name='test_function',
            **self._function_kwargs)


def _make_predict_function(self):
    if not hasattr(self, 'predict_function'):
        self.predict_function = None
    if self.predict_function is None:
        if self._uses_dynamic_learning_phase():
            inputs = self._feed_inputs + [K.learning_phase()]
        else:
```

```
            inputs = self._feed_inputs
        # Gets network outputs. Does not update weights.
        # Does update the network states.
        kwargs = getattr(self, '_function_kwargs', {})
        self.predict_function = K.function(inputs,
                                           self.outputs,
                                           updates=self.state_updates,
                                           name='predict_function',
                                           **kwargs)

def _uses_dynamic_learning_phase(self):
    return (self.uses_learning_phase and
            not isinstance(K.learning_phase(), int))

def _set_inputs(self, inputs, outputs=None, training=None):
    """Set model's input and output specs based on the input data received.

    This is to be used for Model subclasses, which do not know at instantiation
    time what their inputs look like.

    # Arguments
      inputs: Single array, or list of arrays. The arrays could be placeholders,
        Numpy arrays, or data tensors.
```

```
            - if placeholders: the model is built on top of these placeholders,
              and we expect Numpy data to be fed for them when calling `fit`/etc.
            - if Numpy data: we create placeholders matching the shape of the Numpy
              arrays. We expect Numpy data to be fed for these placeholders
              when calling `fit`/etc.
            - if data tensors: the model is built on top of these tensors.
              We do not expect any Numpy data to be provided when calling `fit`/etc.
        outputs: Optional output tensors (if already computed by running the
model).
        training: Boolean or None. Only relevant in symbolic mode. Specifies
          whether to build the model's graph in inference mode (False), training
          mode (True), or using the Keras learning phase (None).
      """
      if self.__class__.__name__ == 'Sequential':
        # Note: we can't test whether the model
        # is `Sequential` via `isinstance`
        # since `Sequential` depends on `Model`.
        if isinstance(inputs, list):
            assert len(inputs) == 1
            inputs = inputs[0]
        self.build(input_shape=(None,) + inputs.shape[1:])
        return
```

```
if self.inputs:
    raise ValueError('Model inputs are already set.')

# On-the-fly setting of symbolic model inputs
# (either by using the tensor provided,
# or by creating a placeholder if Numpy data was provided).
self.inputs = []
self.input_names = []
self._feed_inputs = []
self._feed_input_names = []
self._feed_input_shapes = []
if isinstance(inputs, (list, tuple)):
    inputs = list(inputs)
else:
    inputs = [inputs]

for i, v in enumerate(inputs):
    name = 'input_%d' % (i + 1)
    self.input_names.append(name)
    if isinstance(v, list):
        v = np.asarray(v)
        if v.ndim == 1:
            v = np.expand_dims(v, 1)
```

```
    if isinstance(v, (np.ndarray)):
        # We fix the placeholder shape except the batch size.
        # This is suboptimal, but it is the best we can do with the info
        # we have. The user should call `model._set_inputs(placeholders)`
        # to specify custom placeholders if the need arises.
        shape = (None,) + v.shape[1:]
        placeholder = K.placeholder(shape=shape, name=name)
        self.inputs.append(placeholder)
        self._feed_inputs.append(placeholder)
        self._feed_input_names.append(name)
        self._feed_input_shapes.append(shape)
    else:
        # Assumed tensor - TODO(fchollet) additional type check?
        self.inputs.append(v)
        if K.is_placeholder(v):
            self._feed_inputs.append(v)
            self._feed_input_names.append(name)
            self._feed_input_shapes.append(K.int_shape(v))

if outputs is None:
    # Obtain symbolic outputs by calling the model.
    if self._expects_training_arg:
        outputs = self.call(unpack_singleton(self.inputs), training=training)
```

```
        else:
            outputs = self.call(unpack_singleton(self.inputs))
    if isinstance(outputs, (list, tuple)):
        outputs = list(outputs)
    else:
        outputs = [outputs]
    self.outputs = outputs
    self.output_names = [
        'output_%d' % (i + 1) for i in range(len(self.outputs))]
    self.built = True


def _standardize_user_data(self, x,
                           y=None,
                           sample_weight=None,
                           class_weight=None,
                           check_array_lengths=True,
                           batch_size=None):
    all_inputs = []
    if not self.built:
        # We need to use `x` to set the model inputs.
        # We type-check that `x` and `y` are either single arrays
        # or lists of arrays.
        if isinstance(x, (list, tuple)):
```

```
    if not all(isinstance(v, np.ndarray) or
               K.is_tensor(v) for v in x):
        raise ValueError('Please provide as model inputs '
                         'either a single '
                         'array or a list of arrays. '
                         'You passed: x=' + str(x))
    all_inputs += list(x)
elif isinstance(x, dict):
    raise ValueError('Please do not pass a dictionary '
                     'as model inputs.')
else:
    if not isinstance(x, np.ndarray) and not K.is_tensor(x):
        raise ValueError('Please provide as model inputs '
                         'either a single '
                         'array or a list of arrays. '
                         'You passed: x=' + str(x))
    all_inputs.append(x)

# Build the model using the retrieved inputs (value or symbolic).
# If values, then in symbolic-mode placeholders will be created
# to match the value shapes.
if not self.inputs:
    self._set_inputs(x)
```

```
if y is not None:
    if not self.optimizer:
        raise RuntimeError('You must compile a model before '
                           'training/testing. '
                           'Use `model.compile(optimizer, loss)`.')
    if not self._is_compiled:
        # On-the-fly compilation of the model.
        # We need to use `y` to set the model targets.
        if isinstance(y, (list, tuple)):
            if not all(isinstance(v, np.ndarray) or
                       K.is_tensor(v) for v in y):
                raise ValueError('Please provide as model targets '
                                 'either a single '
                                 'array or a list of arrays. '
                                 'You passed: y=' + str(y))
        elif isinstance(y, dict):
            raise ValueError('Please do not pass a dictionary '
                             'as model targets.')
        else:
            if not isinstance(y, np.ndarray) and not K.is_tensor(y):
                raise ValueError('Please provide as model targets '
                                 'either a single '
```

```
                    'array or a list of arrays. '
                    'You passed: y=' + str(y))
# Typecheck that all inputs are *either* value *or* symbolic.
if y is not None:
    if isinstance(y, (list, tuple)):
        all_inputs += list(y)
    else:
        all_inputs.append(y)
if any(K.is_tensor(v) for v in all_inputs):
    if not all(K.is_tensor(v) for v in all_inputs):
        raise ValueError('Do not pass inputs that mix Numpy '
                    'arrays and symbolic tensors. '
                    'You passed: x=' + str(x) +
                    '; y=' + str(y))

# Handle target tensors if any passed.
if not isinstance(y, (list, tuple)):
    y = [y]
target_tensors = [v for v in y if K.is_tensor(v)]
if not target_tensors:
    target_tensors = None
self.compile(optimizer=self.optimizer,
             loss=self.loss,
```

```
                metrics=self.metrics,
                loss_weights=self.loss_weights,
                target_tensors=target_tensors)

# If `x` and `y` were all symbolic,
# then the model should not be fed any inputs and targets.
# Note: in this case, `any` and `all` are equivalent since we disallow
# mixed symbolic/value inputs.
if any(K.is_tensor(v) for v in all_inputs):
    return [], [], []

# What follows is input validation and standardization to list format,
# in the case where all inputs are value arrays.

if not self._is_graph_network:
    # Case: symbolic-mode subclassed network.
    # Do not do shape validation.
    feed_input_names = self._feed_input_names
    feed_input_shapes = None
else:
    # Case: symbolic-mode graph network.
    # In this case, we run extensive shape validation checks.
    feed_input_names = self._feed_input_names
```

```
    feed_input_shapes = self._feed_input_shapes


# Standardize the inputs.
x = standardize_input_data(
    x,
    feed_input_names,
    feed_input_shapes,
    check_batch_axis=False,  # Don't enforce the batch size.
    exception_prefix='input')


if y is not None:
    if not self._is_graph_network:
        feed_output_names = self._feed_output_names
        feed_output_shapes = None
        # Sample weighting not supported in this case.
        # TODO: consider supporting it.
        feed_sample_weight_modes = [None for _ in self.outputs]
    else:
        feed_output_names = self._feed_output_names
        feed_sample_weight_modes = self._feed_sample_weight_modes
        feed_output_shapes = []
        for output_shape, loss_fn in zip(self._feed_output_shapes,
                                         self._feed_loss_fns):
```

```
        if loss_fn is losses.sparse_categorical_crossentropy:
            if K.image_data_format() == 'channels_first' and len(
                    output_shape) in [4, 5]:
                feed_output_shapes.append(
                    (output_shape[0], 1) + output_shape[2:])
            else:
                feed_output_shapes.append(output_shape[:-1] + (1,))
        elif (not hasattr(loss_fn, '__name__') or
              getattr(losses, loss_fn.__name__, None) is None):
            # If `loss_fn` is not a function (e.g. callable class)
            # or if it not in the `losses` module, then
            # it is a user-defined loss and we make no assumptions
            # about it.
            feed_output_shapes.append(None)
        else:
            feed_output_shapes.append(output_shape)

    # Standardize the outputs.
    y = standardize_input_data(
        y,
        feed_output_names,
        feed_output_shapes,
        check_batch_axis=False,  # Don't enforce the batch size.
```

```
        exception_prefix='target')

    # Generate sample-wise weight values given the `sample_weight` and
    # `class_weight` arguments.
    sample_weights = standardize_sample_weights(
        sample_weight, feed_output_names)
    class_weights = standardize_class_weights(
        class_weight, feed_output_names)
    sample_weights = [
        standardize_weights(ref, sw, cw, mode)
        for (ref, sw, cw, mode) in
        zip(y, sample_weights, class_weights,
            feed_sample_weight_modes)
    ]
    # Check that all arrays have the same length.
    check_array_length_consistency(x, y, sample_weights)
    if self._is_graph_network:
        # Additional checks to avoid users mistakenly
        # using improper loss fns.
        check_loss_and_target_compatibility(
            y, self._feed_loss_fns, feed_output_shapes)
else:
    y = []
```

```
        sample_weights = []

    if self.stateful and batch_size:
        # Check that for stateful networks, number of samples is a multiple
        # of the static batch size.
        if x[0].shape[0] % batch_size != 0:
            raise ValueError('In a stateful network, '
                             'you should only pass inputs with '
                             'a number of samples that can be '
                             'divided by the batch size. Found: ' +
                             str(x[0].shape[0]) + ' samples')
    return x, y, sample_weights


def fit(self,
        x=None,
        y=None,
        batch_size=None,
        epochs=1,
        verbose=1,
        callbacks=None,
        validation_split=0.,
        validation_data=None,
        shuffle=True,
```

```
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    **kwargs):
"""Trains the model for a given number of epochs (iterations on a dataset).

# Arguments
    x: Numpy array of training data (if the model has a single input),
        or list of Numpy arrays (if the model has multiple inputs).
        If input layers in the model are named, you can also pass a
        dictionary mapping input names to Numpy arrays.
        `x` can be `None` (default) if feeding from
        framework-native tensors (e.g. TensorFlow data tensors).
    y: Numpy array of target (label) data
        (if the model has a single output),
        or list of Numpy arrays (if the model has multiple outputs).
        If output layers in the model are named, you can also pass a
        dictionary mapping output names to Numpy arrays.
        `y` can be `None` (default) if feeding from
        framework-native tensors (e.g. TensorFlow data tensors).
    batch_size: Integer or `None`.
```

Number of samples per gradient update.

If unspecified, `batch_size` will default to 32.

epochs: Integer. Number of epochs to train the model.

An epoch is an iteration over the entire `x` and `y`

data provided.

Note that in conjunction with `initial_epoch`,

`epochs` is to be understood as "final epoch".

The model is not trained for a number of iterations

given by `epochs`, but merely until the epoch

of index `epochs` is reached.

verbose: Integer. 0, 1, or 2. Verbosity mode.

0 = silent, 1 = progress bar, 2 = one line per epoch.

callbacks: List of `keras.callbacks.Callback` instances.

List of callbacks to apply during training.

See [callbacks](/callbacks).

validation_split: Float between 0 and 1.

Fraction of the training data to be used as validation data.

The model will set apart this fraction of the training data,

will not train on it, and will evaluate

the loss and any model metrics

on this data at the end of each epoch.

The validation data is selected from the last samples

in the `x` and `y` data provided, before shuffling.

validation_data: tuple `(x_val, y_val)` or tuple
`(x_val, y_val, val_sample_weights)` on which to evaluate
the loss and any model metrics at the end of each epoch.
The model will not be trained on this data.
`validation_data` will override `validation_split`.
shuffle: Boolean (whether to shuffle the training data
before each epoch) or str (for 'batch').
'batch' is a special option for dealing with the
limitations of HDF5 data; it shuffles in batch-sized chunks.
Has no effect when `steps_per_epoch` is not `None`.
class_weight: Optional dictionary mapping class indices (integers)
to a weight (float) value, used for weighting the loss function
(during training only).
This can be useful to tell the model to
"pay more attention" to samples from
an under-represented class.
sample_weight: Optional Numpy array of weights for
the training samples, used for weighting the loss function
(during training only). You can either pass a flat (1D)
Numpy array with the same length as the input samples
(1:1 mapping between weights and samples),
or in the case of temporal data,
you can pass a 2D array with shape

```
        `(samples, sequence_length)`,
        to apply a different weight to every timestep of every sample.
        In this case you should make sure to specify
        `sample_weight_mode="temporal"` in `compile()`.
    initial_epoch: Integer.
        Epoch at which to start training
        (useful for resuming a previous training run).
    steps_per_epoch: Integer or `None`.
        Total number of steps (batches of samples)
        before declaring one epoch finished and starting the
        next epoch. When training with input tensors such as
        TensorFlow data tensors, the default `None` is equal to
        the number of samples in your dataset divided by
        the batch size, or 1 if that cannot be determined.
    validation_steps: Only relevant if `steps_per_epoch`
        is specified. Total number of steps (batches of samples)
        to validate before stopping.

# Returns
    A `History` object. Its `History.history` attribute is
    a record of training loss values and metrics values
    at successive epochs, as well as validation loss values
    and validation metrics values (if applicable).
```

```
# Raises
    RuntimeError: If the model was never compiled.
    ValueError: In case of mismatch between the provided input data
        and what the model expects.
"""
# Backwards compatibility
if batch_size is None and steps_per_epoch is None:
    batch_size = 32
# Legacy support
if 'nb_epoch' in kwargs:
    warnings.warn('The `nb_epoch` argument in `fit` '
                  'has been renamed `epochs`.', stacklevel=2)
    epochs = kwargs.pop('nb_epoch')
if kwargs:
    raise TypeError('Unrecognized keyword arguments: ' + str(kwargs))
if x is None and y is None and steps_per_epoch is None:
    raise ValueError('If fitting from data tensors, '
                     'you should specify the `steps_per_epoch` '
                     'argument.')
# Validate user data.
x, y, sample_weights = self._standardize_user_data(
    x, y,
```

```
    sample_weight=sample_weight,
    class_weight=class_weight,
    batch_size=batch_size)
# Prepare validation data.
do_validation = False
if validation_data:
    do_validation = True
    if len(validation_data) == 2:
        val_x, val_y = validation_data
        val_sample_weight = None
    elif len(validation_data) == 3:
        val_x, val_y, val_sample_weight = validation_data
    else:
        raise ValueError('When passing validation_data, '
                         'it must contain 2 (x_val, y_val) '
                         'or 3 (x_val, y_val, val_sample_weights) '
                         'items, however it contains %d items' %
                         len(validation_data))

    val_x, val_y, val_sample_weights = self._standardize_user_data(
        val_x, val_y,
        sample_weight=val_sample_weight,
        batch_size=batch_size)
```

```
    if self._uses_dynamic_learning_phase():
        val_ins = val_x + val_y + val_sample_weights + [0.]
    else:
        val_ins = val_x + val_y + val_sample_weights

elif validation_split and 0. < validation_split < 1.:
    if any(K.is_tensor(t) for t in x):
        raise ValueError(
            'If your data is in the form of symbolic tensors, '
            'you cannot use `validation_split`.')
    do_validation = True
    if hasattr(x[0], 'shape'):
        split_at = int(int(x[0].shape[0]) * (1. - validation_split))
    else:
        split_at = int(len(x[0]) * (1. - validation_split))
    x, val_x = (slice_arrays(x, 0, split_at),
                slice_arrays(x, split_at))
    y, val_y = (slice_arrays(y, 0, split_at),
                slice_arrays(y, split_at))
    sample_weights, val_sample_weights = (
        slice_arrays(sample_weights, 0, split_at),
        slice_arrays(sample_weights, split_at))
    if self._uses_dynamic_learning_phase():
```

```
        val_ins = val_x + val_y + val_sample_weights + [0.]
    else:
        val_ins = val_x + val_y + val_sample_weights

elif validation_steps:
    do_validation = True
    if self._uses_dynamic_learning_phase():
        val_ins = [0.]

# Prepare input arrays and training function.
if self._uses_dynamic_learning_phase():
    ins = x + y + sample_weights + [1.]
else:
    ins = x + y + sample_weights
self._make_train_function()
f = self.train_function

# Prepare display labels.
out_labels = self.metrics_names

if do_validation:
    self._make_test_function()
    val_f = self.test_function
```

```
    callback_metrics = copy.copy(out_labels) + [
        'val_' + n for n in out_labels]
  else:
    callback_metrics = copy.copy(out_labels)
    val_f = None
    val_ins = []

  # Delegate logic to `fit_loop`.
  return training_arrays.fit_loop(self, f, ins,
                                  out_labels=out_labels,
                                  batch_size=batch_size,
                                  epochs=epochs,
                                  verbose=verbose,
                                  callbacks=callbacks,
                                  val_f=val_f,
                                  val_ins=val_ins,
                                  shuffle=shuffle,
                                  callback_metrics=callback_metrics,
                                  initial_epoch=initial_epoch,
                                  steps_per_epoch=steps_per_epoch,
                                  validation_steps=validation_steps)

def evaluate(self, x=None, y=None,
```

```
        batch_size=None,
        verbose=1,
        sample_weight=None,
        steps=None):
"""Returns the loss value & metrics values for the model in test mode.

Computation is done in batches.

# Arguments
    x: Numpy array of test data (if the model has a single input),
        or list of Numpy arrays (if the model has multiple inputs).
        If input layers in the model are named, you can also pass a
        dictionary mapping input names to Numpy arrays.
        `x` can be `None` (default) if feeding from
        framework-native tensors (e.g. TensorFlow data tensors).
    y: Numpy array of target (label) data
        (if the model has a single output),
        or list of Numpy arrays (if the model has multiple outputs).
        If output layers in the model are named, you can also pass a
        dictionary mapping output names to Numpy arrays.
        `y` can be `None` (default) if feeding from
        framework-native tensors (e.g. TensorFlow data tensors).
    batch_size: Integer or `None`.
```

```
        Number of samples per evaluation step.
        If unspecified, `batch_size` will default to 32.
    verbose: 0 or 1. Verbosity mode.
        0 = silent, 1 = progress bar.
    sample_weight: Optional Numpy array of weights for
        the test samples, used for weighting the loss function.
        You can either pass a flat (1D)
        Numpy array with the same length as the input samples
        (1:1 mapping between weights and samples),
        or in the case of temporal data,
        you can pass a 2D array with shape
        `(samples, sequence_length)`,
        to apply a different weight to every timestep of every sample.
        In this case you should make sure to specify
        `sample_weight_mode="temporal"` in `compile()`.
    steps: Integer or `None`.
        Total number of steps (batches of samples)
        before declaring the evaluation round finished.
        Ignored with the default value of `None`.

# Returns
    Scalar test loss (if the model has a single output and no metrics)
    or list of scalars (if the model has multiple outputs
```

```
    and/or metrics). The attribute `model.metrics_names` will give you
    the display labels for the scalar outputs.
"""
# Backwards compatibility.
if batch_size is None and steps is None:
    batch_size = 32
if x is None and y is None and steps is None:
    raise ValueError('If evaluating from data tensors, '
                     'you should specify the `steps` '
                     'argument.')
# Validate user data.
x, y, sample_weights = self._standardize_user_data(
    x, y,
    sample_weight=sample_weight,
    batch_size=batch_size)
# Prepare inputs, delegate logic to `test_loop`.
if self._uses_dynamic_learning_phase():
    ins = x + y + sample_weights + [0.]
else:
    ins = x + y + sample_weights
self._make_test_function()
f = self.test_function
return training_arrays.test_loop(self, f, ins,
```

```
                    batch_size=batch_size,
                    verbose=verbose,
                    steps=steps)

def predict(self, x,
            batch_size=None,
            verbose=0,
            steps=None):
    """Generates output predictions for the input samples.

    Computation is done in batches.

    # Arguments
        x: The input data, as a Numpy array
            (or list of Numpy arrays if the model has multiple inputs).
        batch_size: Integer. If unspecified, it will default to 32.
        verbose: Verbosity mode, 0 or 1.
        steps: Total number of steps (batches of samples)
            before declaring the prediction round finished.
            Ignored with the default value of `None`.

    # Returns
        Numpy array(s) of predictions.
```

```
# Raises
    ValueError: In case of mismatch between the provided
        input data and the model's expectations,
        or in case a stateful model receives a number of samples
        that is not a multiple of the batch size.
"""
# Backwards compatibility.
if batch_size is None and steps is None:
    batch_size = 32
if x is None and steps is None:
    raise ValueError('If predicting from data tensors, '
                     'you should specify the `steps` '
                     'argument.')
# Validate user data.
x, _, _ = self._standardize_user_data(x)
if self.stateful:
    if x[0].shape[0] > batch_size and x[0].shape[0] % batch_size != 0:
        raise ValueError('In a stateful network, '
                         'you should only pass inputs with '
                         'a number of samples that can be '
                         'divided by the batch size. Found: ' +
                         str(x[0].shape[0]) + ' samples. '
```

```
                    'Batch size: ' + str(batch_size) + '.')

    # Prepare inputs, delegate logic to `predict_loop`.
    if self._uses_dynamic_learning_phase():
        ins = x + [0.]
    else:
        ins = x
    self._make_predict_function()
    f = self.predict_function
    return training_arrays.predict_loop(self, f, ins,
                                        batch_size=batch_size,
                                        verbose=verbose,
                                        steps=steps)

def train_on_batch(self, x, y,
                   sample_weight=None,
                   class_weight=None):
    """Runs a single gradient update on a single batch of data.

    # Arguments
        x: Numpy array of training data,
            or list of Numpy arrays if the model has multiple inputs.
            If all inputs in the model are named,
```

```
        you can also pass a dictionary
        mapping input names to Numpy arrays.
    y: Numpy array of target data,
        or list of Numpy arrays if the model has multiple outputs.
        If all outputs in the model are named,
        you can also pass a dictionary
        mapping output names to Numpy arrays.
    sample_weight: Optional array of the same length as x, containing
        weights to apply to the model's loss for each sample.
        In the case of temporal data, you can pass a 2D array
        with shape (samples, sequence_length),
        to apply a different weight to every timestep of every sample.
        In this case you should make sure to specify
        sample_weight_mode="temporal" in compile().
    class_weight: Optional dictionary mapping
        class indices (integers) to
        a weight (float) to apply to the model's loss for the samples
        from this class during training.
        This can be useful to tell the model to "pay more attention" to
        samples from an under-represented class.

# Returns
    Scalar training loss
```

```
        (if the model has a single output and no metrics)
        or list of scalars (if the model has multiple outputs
        and/or metrics). The attribute `model.metrics_names` will give you
        the display labels for the scalar outputs.
    """
    x, y, sample_weights = self._standardize_user_data(
        x, y,
        sample_weight=sample_weight,
        class_weight=class_weight)
    if self._uses_dynamic_learning_phase():
        ins = x + y + sample_weights + [1.]
    else:
        ins = x + y + sample_weights
    self._make_train_function()
    outputs = self.train_function(ins)
    return unpack_singleton(outputs)


def test_on_batch(self, x, y, sample_weight=None):
    """Test the model on a single batch of samples.

    # Arguments
        x: Numpy array of test data,
            or list of Numpy arrays if the model has multiple inputs.
```

```
        If all inputs in the model are named,
        you can also pass a dictionary
        mapping input names to Numpy arrays.
    y: Numpy array of target data,
        or list of Numpy arrays if the model has multiple outputs.
        If all outputs in the model are named,
        you can also pass a dictionary
        mapping output names to Numpy arrays.
    sample_weight: Optional array of the same length as x, containing
        weights to apply to the model's loss for each sample.
        In the case of temporal data, you can pass a 2D array
        with shape (samples, sequence_length),
        to apply a different weight to every timestep of every sample.
        In this case you should make sure to specify
        sample_weight_mode="temporal" in compile().

# Returns
    Scalar test loss (if the model has a single output and no metrics)
    or list of scalars (if the model has multiple outputs
    and/or metrics). The attribute `model.metrics_names` will give you
    the display labels for the scalar outputs.
"""
x, y, sample_weights = self._standardize_user_data(
```

```
        x, y,
        sample_weight=sample_weight)
    if self._uses_dynamic_learning_phase():
        ins = x + y + sample_weights + [0.]
    else:
        ins = x + y + sample_weights
    self._make_test_function()
    outputs = self.test_function(ins)
    return unpack_singleton(outputs)

def predict_on_batch(self, x):
    """Returns predictions for a single batch of samples.

    # Arguments
        x: Input samples, as a Numpy array.

    # Returns
        Numpy array(s) of predictions.
    """
    x, _, _ = self._standardize_user_data(x)
    if self._uses_dynamic_learning_phase():
        ins = x + [0.]
    else:
```

```
            ins = x
        self._make_predict_function()
        outputs = self.predict_function(ins)
        return unpack_singleton(outputs)

    @interfaces.legacy_generator_methods_support
    def fit_generator(self, generator,
                      steps_per_epoch=None,
                      epochs=1,
                      verbose=1,
                      callbacks=None,
                      validation_data=None,
                      validation_steps=None,
                      class_weight=None,
                      max_queue_size=10,
                      workers=1,
                      use_multiprocessing=False,
                      shuffle=True,
                      initial_epoch=0):
        """Trains the model on data generated batch-by-batch by a Python generator
(or an instance of `Sequence`).

        The generator is run in parallel to the model, for efficiency.
```

For instance, this allows you to do real-time data augmentation
on images on CPU in parallel to training your model on GPU.

The use of `keras.utils.Sequence` guarantees the ordering
and guarantees the single use of every input per epoch when
using `use_multiprocessing=True`.

Arguments
    generator: A generator or an instance of `Sequence`
      (`keras.utils.Sequence`) object in order to avoid
      duplicate data when using multiprocessing.
      The output of the generator must be either
      - a tuple `(inputs, targets)`
      - a tuple `(inputs, targets, sample_weights)`.
      This tuple (a single output of the generator) makes a single
      batch. Therefore, all arrays in this tuple must have the same
      length (equal to the size of this batch). Different batches may
      have different sizes. For example, the last batch of the epoch
      is commonly smaller than the others, if the size of the dataset
      is not divisible by the batch size.
      The generator is expected to loop over its data
      indefinitely. An epoch finishes when `steps_per_epoch`
      batches have been seen by the model.

steps_per_epoch: Integer.
    Total number of steps (batches of samples)
    to yield from `generator` before declaring one epoch
    finished and starting the next epoch. It should typically
    be equal to the number of samples of your dataset
    divided by the batch size.
    Optional for `Sequence`: if unspecified, will use
    the `len(generator)` as a number of steps.
epochs: Integer. Number of epochs to train the model.
    An epoch is an iteration over the entire data provided,
    as defined by `steps_per_epoch`.
    Note that in conjunction with `initial_epoch`,
    `epochs` is to be understood as "final epoch".
    The model is not trained for a number of iterations
    given by `epochs`, but merely until the epoch
    of index `epochs` is reached.
verbose: Integer. 0, 1, or 2. Verbosity mode.
    0 = silent, 1 = progress bar, 2 = one line per epoch.
callbacks: List of `keras.callbacks.Callback` instances.
    List of callbacks to apply during training.
    See [callbacks](/callbacks).
validation_data: This can be either
    - a generator or a `Sequence` object for the validation data

- tuple `(x_val, y_val)`
- tuple `(x_val, y_val, val_sample_weights)`
on which to evaluate
the loss and any model metrics at the end of each epoch.
The model will not be trained on this data.
validation_steps: Only relevant if `validation_data`
is a generator. Total number of steps (batches of samples)
to yield from `validation_data` generator before stopping
at the end of every epoch. It should typically
be equal to the number of samples of your
validation dataset divided by the batch size.
Optional for `Sequence`: if unspecified, will use
the `len(validation_data)` as a number of steps.
class_weight: Optional dictionary mapping class indices (integers)
to a weight (float) value, used for weighting the loss function
(during training only). This can be useful to tell the model to
"pay more attention" to samples
from an under-represented class.
max_queue_size: Integer. Maximum size for the generator queue.
If unspecified, `max_queue_size` will default to 10.
workers: Integer. Maximum number of processes to spin up
when using process-based threading.
If unspecified, `workers` will default to 1. If 0, will

execute the generator on the main thread.

use_multiprocessing: Boolean.

If `True`, use process-based threading.

If unspecified, `use_multiprocessing` will default to `False`.

Note that because this implementation

relies on multiprocessing,

you should not pass non-picklable arguments to the generator

as they can't be passed easily to children processes.

shuffle: Boolean. Whether to shuffle the order of the batches at

the beginning of each epoch. Only used with instances

of `Sequence` (`keras.utils.Sequence`).

Has no effect when `steps_per_epoch` is not `None`.

initial_epoch: Integer.

Epoch at which to start training

(useful for resuming a previous training run).

Returns

A `History` object. Its `History.history` attribute is

a record of training loss values and metrics values

at successive epochs, as well as validation loss values

and validation metrics values (if applicable).

Raises

    ValueError: In case the generator yields data in an invalid format.

Example

```python
def generate_arrays_from_file(path):
    while True:
        with open(path) as f:
            for line in f:
                # create numpy arrays of input data
                # and labels, from each line in the file
                x1, x2, y = process_line(line)
                yield ({'input_1': x1, 'input_2': x2}, {'output': y})

model.fit_generator(generate_arrays_from_file('/my_file.txt'),
                    steps_per_epoch=10000, epochs=10)
```
"""
return training_generator.fit_generator(
    self, generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    verbose=verbose,

```
        callbacks=callbacks,
        validation_data=validation_data,
        validation_steps=validation_steps,
        class_weight=class_weight,
        max_queue_size=max_queue_size,
        workers=workers,
        use_multiprocessing=use_multiprocessing,
        shuffle=shuffle,
        initial_epoch=initial_epoch)

@interfaces.legacy_generator_methods_support
def evaluate_generator(self, generator,
                       steps=None,
                       max_queue_size=10,
                       workers=1,
                       use_multiprocessing=False,
                       verbose=0):
    """Evaluates the model on a data generator.

    The generator should return the same kind of data
    as accepted by `test_on_batch`.

    # Arguments
```

generator: Generator yielding tuples (inputs, targets)
    or (inputs, targets, sample_weights)
    or an instance of Sequence (keras.utils.Sequence)
    object in order to avoid duplicate data
    when using multiprocessing.
steps: Total number of steps (batches of samples)
    to yield from `generator` before stopping.
    Optional for `Sequence`: if unspecified, will use
    the `len(generator)` as a number of steps.
max_queue_size: maximum size for the generator queue
workers: Integer. Maximum number of processes to spin up
    when using process based threading.
    If unspecified, `workers` will default to 1. If 0, will
    execute the generator on the main thread.
use_multiprocessing: if True, use process based threading.
    Note that because
    this implementation relies on multiprocessing,
    you should not pass
    non picklable arguments to the generator
    as they can't be passed
    easily to children processes.
verbose: verbosity mode, 0 or 1.

```
    # Returns
        Scalar test loss (if the model has a single output and no metrics)
        or list of scalars (if the model has multiple outputs
        and/or metrics). The attribute `model.metrics_names` will give you
        the display labels for the scalar outputs.

    # Raises
        ValueError: In case the generator yields
            data in an invalid format.
    """
    return training_generator.evaluate_generator(
        self, generator,
        steps=steps,
        max_queue_size=max_queue_size,
        workers=workers,
        use_multiprocessing=use_multiprocessing,
        verbose=verbose)

@interfaces.legacy_generator_methods_support
def predict_generator(self, generator,
                      steps=None,
                      max_queue_size=10,
                      workers=1,
```

```
                use_multiprocessing=False,
                verbose=0):
    """Generates predictions for the input samples from a data generator.

    The generator should return the same kind of data as accepted by
    `predict_on_batch`.

    # Arguments
        generator: Generator yielding batches of input samples
            or an instance of Sequence (keras.utils.Sequence)
            object in order to avoid duplicate data
            when using multiprocessing.
        steps: Total number of steps (batches of samples)
            to yield from `generator` before stopping.
            Optional for `Sequence`: if unspecified, will use
            the `len(generator)` as a number of steps.
        max_queue_size: Maximum size for the generator queue.
        workers: Integer. Maximum number of processes to spin up
            when using process based threading.
            If unspecified, `workers` will default to 1. If 0, will
            execute the generator on the main thread.
        use_multiprocessing: If `True`, use process based threading.
            Note that because
```

```
            this implementation relies on multiprocessing,
            you should not pass
            non picklable arguments to the generator
            as they can't be passed
            easily to children processes.
        verbose: verbosity mode, 0 or 1.


    # Returns
        Numpy array(s) of predictions.


    # Raises
        ValueError: In case the generator yields
            data in an invalid format.
    """
    return training_generator.predict_generator(
        self, generator,
        steps=steps,
        max_queue_size=max_queue_size,
        workers=workers,
        use_multiprocessing=use_multiprocessing,
        verbose=verbose)
```

5. train_generator.py

```
"""Part of the training core related to Python generators of array data.
```

```
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import warnings
import numpy as np

from .. import backend as K
from ..utils.data_utils import Sequence
from ..utils.data_utils import GeneratorEnqueuer
from ..utils.data_utils import OrderedEnqueuer
from ..utils.generic_utils import Progbar
from ..utils.generic_utils import to_list
from ..utils.generic_utils import unpack_singleton
from face_ai.deepnet.function import callbacks as cbks


def fit_generator(model,
                  generator,
                  steps_per_epoch=None,
                  epochs=1,
                  verbose=1,
```

```
            callbacks=None,
            validation_data=None,
            validation_steps=None,
            class_weight=None,
            max_queue_size=10,
            workers=1,
            use_multiprocessing=False,
            shuffle=True,
            initial_epoch=0):
"""See docstring for `Model.fit_generator`."""
wait_time = 0.01  # in seconds
epoch = initial_epoch

do_validation = bool(validation_data)
model._make_train_function()
if do_validation:
    model._make_test_function()

is_sequence = isinstance(generator, Sequence)
if not is_sequence and use_multiprocessing and workers > 1:
    warnings.warn(
        UserWarning('Using a generator with `use_multiprocessing=True`'
                    ' and multiple workers may duplicate your data.'
```

```
                ' Please consider using the`keras.utils.Sequence'
                ' class.'))
    if steps_per_epoch is None:
        if is_sequence:
            steps_per_epoch = len(generator)
        else:
            raise ValueError('`steps_per_epoch=None` is only valid for a'
                             ' generator based on the '
                             '`keras.utils.Sequence`'
                             ' class. Please specify `steps_per_epoch` '
                             'or use the `keras.utils.Sequence` class.')

    # python 2 has 'next', 3 has '__next__'
    # avoid any explicit version checks
    val_gen = (hasattr(validation_data, 'next') or
               hasattr(validation_data, '__next__') or
               isinstance(validation_data, Sequence))
    if (val_gen and not isinstance(validation_data, Sequence) and
            not validation_steps):
        raise ValueError('`validation_steps=None` is only valid for a'
                         ' generator based on the `keras.utils.Sequence`'
                         ' class. Please specify `validation_steps` or use'
                         ' the `keras.utils.Sequence` class.')
```

```
# Prepare display labels.
out_labels = model.metrics_names
callback_metrics = out_labels + ['val_' + n for n in out_labels]

# prepare callbacks
model.history = cbks.History()
_callbacks = [cbks.BaseLogger(
    stateful_metrics=model.stateful_metric_names)]
if verbose:
    _callbacks.append(
        cbks.ProgbarLogger(
            count_mode='steps',
            stateful_metrics=model.stateful_metric_names))
_callbacks += (callbacks or []) + [model.history]
callbacks = cbks.CallbackList(_callbacks)

# it's possible to callback a different model than self:
if hasattr(model, 'callback_model') and model.callback_model:
    callback_model = model.callback_model
else:
    callback_model = model
callbacks.set_model(callback_model)
```

```
callbacks.set_params({
    'epochs': epochs,
    'steps': steps_per_epoch,
    'verbose': verbose,
    'do_validation': do_validation,
    'metrics': callback_metrics,
})
callbacks.on_train_begin()

enqueuer = None
val_enqueuer = None

try:
    if do_validation:
        if val_gen and workers > 0:
            # Create an Enqueuer that can be reused
            val_data = validation_data
            if isinstance(val_data, Sequence):
                val_enqueuer = OrderedEnqueuer(val_data,
                                               use_multiprocessing=use_multiprocessing)
                validation_steps = len(val_data)
            else:
                val_enqueuer = GeneratorEnqueuer(val_data,
```

```
                                    use_multiprocessing=use_multiprocessing)
    val_enqueuer.start(workers=workers,
                       max_queue_size=max_queue_size)
    val_enqueuer_gen = val_enqueuer.get()
elif val_gen:
    val_data = validation_data
    if isinstance(val_data, Sequence):
        val_enqueuer_gen = iter(val_data)
    else:
        val_enqueuer_gen = val_data
else:
    # Prepare data for validation
    if len(validation_data) == 2:
        val_x, val_y = validation_data
        val_sample_weight = None
    elif len(validation_data) == 3:
        val_x, val_y, val_sample_weight = validation_data
    else:
        raise ValueError('`validation_data` should be a tuple '
                         '`(val_x, val_y, val_sample_weight)` '
                         'or `(val_x, val_y)`. Found: ' +
                         str(validation_data))
    val_x, val_y, val_sample_weights = model._standardize_user_data(
```

```
            val_x, val_y, val_sample_weight)
        val_data = val_x + val_y + val_sample_weights
        if model.uses_learning_phase and not isinstance(K.learning_phase(),
                                                        int):
            val_data += [0.]
        for cbk in callbacks:
            cbk.validation_data = val_data


if workers > 0:
    if is_sequence:
        enqueuer = OrderedEnqueuer(
            generator,
            use_multiprocessing=use_multiprocessing,
            shuffle=shuffle)
    else:
        enqueuer = GeneratorEnqueuer(
            generator,
            use_multiprocessing=use_multiprocessing,
            wait_time=wait_time)
    enqueuer.start(workers=workers, max_queue_size=max_queue_size)
    output_generator = enqueuer.get()
else:
    if is_sequence:
```

```
        output_generator = iter(generator)
    else:
        output_generator = generator

callback_model.stop_training = False
# Construct epoch logs.
epoch_logs = {}
while epoch < epochs:
    for m in model.stateful_metric_functions:
        m.reset_states()
    callbacks.on_epoch_begin(epoch)
    steps_done = 0
    batch_index = 0
    while steps_done < steps_per_epoch:
        generator_output = next(output_generator)

        if not hasattr(generator_output, '__len__'):
            raise ValueError('Output of generator should be '
                             'a tuple `(x, y, sample_weight)` '
                             'or `(x, y)`. Found: ' +
                             str(generator_output))

        if len(generator_output) == 2:
```

```
    x, y = generator_output
    sample_weight = None
elif len(generator_output) == 3:
    x, y, sample_weight = generator_output
else:
    raise ValueError('Output of generator should be '
                     'a tuple `(x, y, sample_weight)` '
                     'or `(x, y)`. Found: ' +
                     str(generator_output))
# build batch logs
batch_logs = {}
if x is None or len(x) == 0:
    # Handle data tensors support when no input given
    # step-size = 1 for data tensors
    batch_size = 1
elif isinstance(x, list):
    batch_size = x[0].shape[0]
elif isinstance(x, dict):
    batch_size = list(x.values())[0].shape[0]
else:
    batch_size = x.shape[0]
batch_logs['batch'] = batch_index
batch_logs['size'] = batch_size
```

```
callbacks.on_batch_begin(batch_index, batch_logs)

outs = model.train_on_batch(x, y,
                            sample_weight=sample_weight,
                            class_weight=class_weight)

outs = to_list(outs)
for l, o in zip(out_labels, outs):
    batch_logs[l] = o

callbacks.on_batch_end(batch_index, batch_logs)

batch_index += 1
steps_done += 1

# Epoch finished.
if steps_done >= steps_per_epoch and do_validation:
    if val_gen:
        val_outs = model.evaluate_generator(
            val_enqueuer_gen,
            validation_steps,
            workers=0)
    else:
```

```
                # No need for try/except because
                # data has already been validated.
                val_outs = model.evaluate(
                    val_x, val_y,
                    batch_size=batch_size,
                    sample_weight=val_sample_weights,
                    verbose=0)
                val_outs = to_list(val_outs)
                # Same labels assumed.
                for l, o in zip(out_labels, val_outs):
                    epoch_logs['val_' + l] = o

            if callback_model.stop_training:
                break

        callbacks.on_epoch_end(epoch, epoch_logs)
        epoch += 1
        if callback_model.stop_training:
            break

finally:
    try:
        if enqueuer is not None:
```

```
            enqueuer.stop()
        finally:
            if val_enqueuer is not None:
                val_enqueuer.stop()

    callbacks.on_train_end()
    return model.history


def evaluate_generator(model, generator,
                       steps=None,
                       max_queue_size=10,
                       workers=1,
                       use_multiprocessing=False,
                       verbose=0):
    """See docstring for `Model.evaluate_generator`."""
    model._make_test_function()

    stateful_metric_indices = []
    if hasattr(model, 'metrics'):
        for m in model.stateful_metric_functions:
            m.reset_states()
        stateful_metric_indices = [
```

```
        i for i, name in enumerate(model.metrics_names)
        if str(name) in model.stateful_metric_names]
else:
    stateful_metric_indices = []


steps_done = 0
wait_time = 0.01
outs_per_batch = []
batch_sizes = []
is_sequence = isinstance(generator, Sequence)
if not is_sequence and use_multiprocessing and workers > 1:
    warnings.warn(
        UserWarning('Using a generator with `use_multiprocessing=True`'
                    ' and multiple workers may duplicate your data.'
                    ' Please consider using the`keras.utils.Sequence'
                    ' class.'))
if steps is None:
    if is_sequence:
        steps = len(generator)
    else:
        raise ValueError('`steps=None` is only valid for a generator'
                         ' based on the `keras.utils.Sequence` class.'
                         ' Please specify `steps` or use the'
```

```
                    ' `keras.utils.Sequence` class.')
enqueuer = None


try:
    if workers > 0:
        if is_sequence:
            enqueuer = OrderedEnqueuer(
                generator,
                use_multiprocessing=use_multiprocessing)
        else:
            enqueuer = GeneratorEnqueuer(
                generator,
                use_multiprocessing=use_multiprocessing,
                wait_time=wait_time)
        enqueuer.start(workers=workers, max_queue_size=max_queue_size)
        output_generator = enqueuer.get()
    else:
        if is_sequence:
            output_generator = iter(generator)
        else:
            output_generator = generator


    if verbose == 1:
```

```
    progbar = Progbar(target=steps)


while steps_done < steps:
    generator_output = next(output_generator)
    if not hasattr(generator_output, '__len__'):
        raise ValueError('Output of generator should be a tuple '
                         '(x, y, sample_weight) '
                         'or (x, y). Found: ' +
                         str(generator_output))
    if len(generator_output) == 2:
        x, y = generator_output
        sample_weight = None
    elif len(generator_output) == 3:
        x, y, sample_weight = generator_output
    else:
        raise ValueError('Output of generator should be a tuple '
                         '(x, y, sample_weight) '
                         'or (x, y). Found: ' +
                         str(generator_output))
    outs = model.test_on_batch(x, y, sample_weight=sample_weight)
    outs = to_list(outs)
    outs_per_batch.append(outs)
```

```
        if x is None or len(x) == 0:
            # Handle data tensors support when no input given
            # step-size = 1 for data tensors
            batch_size = 1
        elif isinstance(x, list):
            batch_size = x[0].shape[0]
        elif isinstance(x, dict):
            batch_size = list(x.values())[0].shape[0]
        else:
            batch_size = x.shape[0]
        if batch_size == 0:
            raise ValueError('Received an empty batch. '
                             'Batches should contain '
                             'at least one item.')
        steps_done += 1
        batch_sizes.append(batch_size)
        if verbose == 1:
            progbar.update(steps_done)

finally:
    if enqueuer is not None:
        enqueuer.stop()
```

```
    averages = []
    for i in range(len(outs)):
        if i not in stateful_metric_indices:
            averages.append(np.average([out[i] for out in outs_per_batch],
                                       weights=batch_sizes))
        else:
            averages.append(np.float64(outs_per_batch[-1][i]))
    return unpack_singleton(averages)



def predict_generator(model, generator,
                      steps=None,
                      max_queue_size=10,
                      workers=1,
                      use_multiprocessing=False,
                      verbose=0):
    """See docstring for `Model.predict_generator`."""
    model._make_predict_function()

    steps_done = 0
    wait_time = 0.01
    all_outs = []
    is_sequence = isinstance(generator, Sequence)
```

```
if not is_sequence and use_multiprocessing and workers > 1:
    warnings.warn(
        UserWarning('Using a generator with `use_multiprocessing=True`'
                    ' and multiple workers may duplicate your data.'
                    ' Please consider using the`keras.utils.Sequence'
                    ' class.'))
if steps is None:
    if is_sequence:
        steps = len(generator)
    else:
        raise ValueError('`steps=None` is only valid for a generator'
                         ' based on the `keras.utils.Sequence` class.'
                         ' Please specify `steps` or use the'
                         ' `keras.utils.Sequence` class.')
enqueuer = None


try:
    if workers > 0:
        if is_sequence:
            enqueuer = OrderedEnqueuer(
                generator,
                use_multiprocessing=use_multiprocessing)
        else:
```

```
        enqueuer = GeneratorEnqueuer(
            generator,
            use_multiprocessing=use_multiprocessing,
            wait_time=wait_time)
    enqueuer.start(workers=workers, max_queue_size=max_queue_size)
    output_generator = enqueuer.get()
else:
    if is_sequence:
        output_generator = iter(generator)
    else:
        output_generator = generator

if verbose == 1:
    progbar = Progbar(target=steps)

while steps_done < steps:
    generator_output = next(output_generator)
    if isinstance(generator_output, tuple):
        # Compatibility with the generators
        # used for training.
        if len(generator_output) == 2:
            x, _ = generator_output
        elif len(generator_output) == 3:
```

```
            x, _, _ = generator_output
        else:
            raise ValueError('Output of generator should be '
                             'a tuple `(x, y, sample_weight)` '
                             'or `(x, y)`. Found: ' +
                             str(generator_output))
    else:
        # Assumes a generator that only
        # yields inputs (not targets and sample weights).
        x = generator_output

    outs = model.predict_on_batch(x)
    outs = to_list(outs)

    if not all_outs:
        for out in outs:
            all_outs.append([])

    for i, out in enumerate(outs):
        all_outs[i].append(out)
    steps_done += 1
    if verbose == 1:
        progbar.update(steps_done)
```

```
    finally:
        if enqueuer is not None:
            enqueuer.stop()

    if len(all_outs) == 1:
        if steps_done == 1:
            return all_outs[0][0]
        else:
            return np.concatenate(all_outs[0])
    if steps_done == 1:
        return [out[0] for out in all_outs]
    else:
        return [np.concatenate(out) for out in all_outs]
```

6. views.py

```
from django.http import StreamingHttpResponse
from django.views.decorators.gzip import gzip_page

from media_stream.camera import VideoCamera


def gen(camera):
    while True:
```

```
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')


@gzip_page
def cam_live(request):
    try:
        return StreamingHttpResponse(gen(VideoCamera()),
        content_type="multipart/x-mixed-replace;boundary=frame")
    except:
        pass
```

7. camera.py

```
import cv2
import numpy as np
import dlib
from imutils import face_utils
from face_ai import utils
import time
import random


def recognize(face, image, id):
```

```
start_time = time.time()
(fX, fY, fW, fH) = face_utils.rect_to_bb(face)
roi = image[fY:fY + fH, fX:fX + fW]
roi = cv2.resize(roi, (96, 96))
label, prediction, respond_time = utils.predict(roi)
respond_time = time.time() - start_time

acc = prediction[np.argmax(prediction)] * 100

prob = "%.2f" % acc
respond_time = "%.2f" % respond_time

time_format = 'Respond Time: {0}s'.format(respond_time)
opt = '{0}. {1}'.format(id,label)

if acc > 98 :
   cv2.putText(image, opt, (fX, fY - 10),
               cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)

cv2.rectangle(image, (fX, fY), (fX + fW, fY + fH),
              (244, 212, 66), 1)

cv2.putText(image, time_format, (0, 10),
```

```
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)
    return image



class VideoCamera(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.detector = dlib.get_frontal_face_detector()

    def __del__(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        image = cv2.flip(image, 1)
        #image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)

        faces = self.detector(image)

        if len(faces) > 0:
            i = 1
            for face in faces:
                try:
```

```
            image = recognize(face=face, image=image,id=i)
        except:
            print('error')
        i += 1

    ret, jpeg = cv2.imencode('.jpg', image)
    filename = str(random.randint(1,1000)) + '.jpg'
    cv2.imwrite('test_data/realtime/' + filename,image)

    return jpeg.tobytes()
```

8. utils.py

```
from face_ai.deepnet.models import load_model
import face_ai.deepnet.backend
import tensorflow as tf

import numpy as np
import pickle
import time

# initializing the graph
graph = tf.get_default_graph()

_backend = face_ai.deepnet.backend.backend()
```

```
# loading our trained model
print("Model loading.....")
model = load_model('face_ai/models/face_ai.hdf5', compile=False)
label_encoder = pickle.load(open('face_ai/models/label_encoder', 'rb'))
print("Model loaded!!")


def predict(img):
    start_time = time.time()

    if _backend == 'tensorflow':
        with graph.as_default():
            prediction = model.predict(np.expand_dims(img, axis=0))[0]
    else:
        prediction = model.predict(np.expand_dims(img, axis=0))[0]

    respond_time = time.time() - start_time
    label = label_encoder[np.argmax(prediction)]

    return label, prediction, respond_time
```

```
def predict_batch(x):
    start_time = time.time()

    for i in range(0, len(x)):
        x[i] = (x[i] / 225.)

    if _backend == 'tensorflow':
        with graph.as_default():
            predictions = model.predict(x)
    else:
        predictions = model.predict(x)

    labels = []
    for pred in predictions:
        label = label_encoder[np.argmax(pred)]
        labels.append(label)

    respond_time = time.time() - start_time

    return labels, predictions, respond_time
```