Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

# PENGENALAN WAJAH MENGGUNAKAN *DEEP LEARNING* UNTUK SISTEM MONITORING KEHADIRAN BERBASIS CCTV PADA KEGIATAN PERKULIAHAN

MUHAJIR BIN ABD LATIF
NRP 05111640000104

Dosen Pembimbing
Dr. Eng. Nanik Suciati S.Kom., M.Kom.
Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

**TUGAS AKHIR – IF184802**

# PENGENALAN WAJAH MENGGUNAKAN *DEEP LEARNING* UNTUK SISTEM MONITORING KEHADIRAN BERBASIS CCTV PADA KEGIATAN PERKULIAHAN

MUHAJIR BIN ABD LATIF
NRP 05111640000104

Dosen Pembimbing
Dr. Eng. Nanik Suciati S.Kom., M.Kom.
Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

*[Halaman ini sengaja dikosongkan]*

**FINAL PROJECT – IF184802**

# FACE RECOGNITION USING DEEP LEARNING FOR CCTV BASED ATTENDANCE MONITORING SYSTEM ON LECTURE ACTIVITIES

MUHAJIR BIN ABD LATIF
NRP 05111640000104

Advisors
Dr. Eng. Nanik Suciati S.Kom., M.Kom.
Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

## PENGENALAN WAJAH MENGGUNAKAN *DEEP LEARNING* UNTUK SISTEM MONITORING KEHADIRAN BERBASIS CCTV PADA KEGIATAN PERKULIAHAN

## TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
**MUHAJIR BIN ABD LATIF**
**NRP. 05111640000104**

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr. Eng. Nanik Suciati S.Kom., M.Kom. ……
NIP. 19710428 199412 2 001 (Pembimbing 1)

Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D ……
NID. 0009018705 (Pembimbing 2)

**SURABAYA**
**JUNI 2020**

*[Halaman ini sengaja dikosongkan]*

**PENGENALAN WAJAH MENGGUNAKAN *DEEP LEARNING* UNTUK SISTEM MONITORING KEHADIRAN BERBASIS CCTV PADA KEGIATAN PERKULIAHAN**


| | | |
|---|---|---|
| Nama Mahasiswa | : | Muhajir Bin Abd Latif |
| NRP | : | 05111640000104 |
| Departemen | : | Teknik Informatika FTEIC-ITS |
| Dosen Pembimbing 1 | : | Dr. Eng. Nanik Suciati S.Kom., M.Kom. |
| Dosen Pembimbing 2 | : | Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D |


## ABSTRAK


*Kehadiran mahasiswa merupakan salah satu hal yang dianggap penting dalam dunia perkuliahan. Terbukti dengan banyak universitas di Indonesia yang menjadikan kehadiran mahasiswa sebagai syarat untuk lulus dari mata kuliah yang diambil. Adanya peraturan tersebut membuat banyak mahasiswa yang tidak memenuhi kriteria kehadiran akhirnya melakukan kecurangan dengan memanipulasi kehadirannya. Oleh karena itu, dibutuhkan sistem yang dapat mengurangi kecurangan yang terjadi.*

*Dalam tugas akhir ini, akan dikembangkan model pengenalan wajah untuk sistem monitoring kehadiran berbasis CCTV pada kegiatan perkuliahan. Sistem menerima input berupa data video. Setiap frame dari video diproses untuk dilakukan deteksi wajah. Wajah-wajah yang berhasil dideteksi kemudian dilakukan proses pengenalan dan hasilnya disimpan pada database untuk penyusunan rekapitulasi kehadiran.*

*Berdasarkan hasil pengujian yang dilakukan, deteksi wajah menggunakan YOLO menghasilkan performa terbaik dengan mean average precision (mAP) sebesar 80,2%, sedangkan model pengenalan wajah dengan performa terbaik didapatkan saat*

*menggunakan ekstraksi fitur berbasis CNN dengan arsitektur VGG16 dan klasifikasi menggunakan SVM dengan akurasi yang didapatkan sebesar 99,28%. Gabungan dari kedua model deteksi wajah dan model pengenalan wajah tersebut menghasilkan akurasi sebesar 71,32% pada keseluruhan sistem.*

**Kata Kunci:** ***Convolutional Neural Network, Deteksi Wajah, Pengenalan Wajah, Sistem Monitoring Kehadiran***

**FACE RECOGNITION USING DEEP LEARNING FOR CCTV BASED ATTENDANCE MONITORING SYSTEM ON LECTURE ACTIVITIES**


| | | |
|---|---|---|
| Student Name | : | Muhajir Bin Abd Latif |
| NRP | : | 05111640000104 |
| Department | : | Teknik Informatika FTEIC-ITS |
| First Advisor | : | Dr. Eng. Nanik Suciati S.Kom., M.Kom. |
| Second Advisor | : | Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D |


## ABSTRACT


*Student's attendance has become one of the most important things in college. This evidenced by many of universities in Indonesia used it for prerequisite of the certain course that should be passed. By this fact there are so many students skipped and manipulated their attendance in class. There should be a system that can reduce the manipulation that occurs.*

*In this study, face recognition in order to monitor the student's attendance in college is developed. The system accepts video as an input. Face detection is performed on every frame of the video. Face identification is then performed on each detected face and the results are stored in a database for making attendance records.*

*Based on the test results, face detection using YOLO shows the best performance with mean average precision (mAP) of 80,2%, while the model of face recognition with the best performance is obtained when using the CNN-based feature extraction and classification VGG16 architecture using SVM with accuracy obtained at 99.28%. The combination of the face detection model and facial recognition model produces an accuracy of 71.32% for the entire system.*

**Keywords:** ***Attendance Monitoring System, Convolutional Neural Network, Face Detection, Face Recognition***

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT. karena atas karunian dan rahmat-Nya saya dapat menyelesaikan tugas akhir yang berjudul:

**PENGENALAN WAJAH MENGGUNAKAN *DEEP LEARNING* UNTUK SISTEM MONITORING KEHADIRAN BERBASIS CCTV PADA KEGIATAN PERKULIAHAN**

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Orang tua dan keluarga penulis, yang telah senantiasa membimbing, memberikan dukungan doa, moral, dan material sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Bu Dr. Eng. Nanik Suciati S.Kom., M.Kom. dan Bu Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D selaku pembimbing 1 dan 2 yang telah membimbing dan memberikan banyak masukan dalam pengerjaan tugas akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Informatika ITS dan segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
4. Marde Fasma dan Rasyid Fajar sebagai teman seperjuangan yang saling membantu selama tugas akhir.
5. Fino, Zevi, Yuda, Andre, Naja, Shania, Anggar, Hisam, Syubban, Kana, Ella, Mbak Pede, Mbak Purin, Mas Rafi, Mas Ariya, Mas Fajar, Mas adam, Mas Irfan, Mas Ovan,

Mas Hanif, Mas Ikhsan, Mbak Uli, dan Mas Fany sebagai keluarga admin lab. Mobile Innovation Studio.

6. Rizqi, Ajis, Anas, Lelly dan Hanim sebagai sahabat *dolor* yang selalu menemani penulis.
7. Rezi Ulya Fauziah yang selalu memberikan dukungan dan semangat kepada penulis.
8. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari banyaknya kekurangan dalam tugas akhir ini. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun dari pembaca untuk perbaikan kedepannya.

Surabaya, Juni 2020

Muhajir Bin Abd Latif

# DAFTAR ISI

**LEMBAR PENGESAHAN** ..... **VII**
**ABSTRAK** ..... **IX**
**ABSTRACT** ..... **XI**
**KATA PENGANTAR** ..... **XIII**
**DAFTAR ISI** ..... **XV**
**DAFTAR GAMBAR** ..... **XIX**
**DAFTAR TABEL** ..... **XXI**
**DAFTAR KODE SUMBER** ..... **XXIII**
**BAB 1 PENDAHULUAN** ..... **1**
1.1 Latar Belakang ..... 1
1.2 Rumusan Masalah ..... 2
1.3 Batasan Masalah ..... 2
1.4 Tujuan ..... 2
1.5 Manfaat ..... 3
1.6 Metodologi ..... 3
1.7 Sistematika Penulisan ..... 4
**BAB 2 TINJAUAN PUSTAKA** ..... **7**
2.1 *Deep Learning* ..... 7
2.2 *Convolutional Neural Network* ..... 7
2.3 *Transfer Learning* ..... 9
2.4 *Multi-task Cascaded Convolutional Networks* ..... 9
2.5 *Dlib Face Detector* ..... 11
2.6 *YOLO: You Only Look Once* ..... 11
2.7 *Keras VGGFace* ..... 13

2.8 *Confusion Matrix* ..... 13
2.9 *Mean Average Precision (mAP)* ..... 14
**BAB 3 ANALISIS DAN PERANCANGAN SISTEM ..... 19**
3.1 Perancangan Sistem ..... 19
3.2 Pengumpulan Data ..... 20
3.3 Perancangan Proses ..... 20
3.3.1 Proses Pemilihan *Frame* ..... 21
3.3.2 Proses Deteksi Wajah ..... 21
3.3.3 Proses Pengenalan Wajah ..... 23
3.3.4 Proses Pengolahan Hasil Pengenalan ..... 26
3.4 Perancangan Antarmuka ..... 28
**BAB 4 IMPLEMENTASI ..... 31**
4.1 Lingkungan Implementasi ..... 31
4.2 Implementasi Model Deteksi Wajah ..... 31
4.2.1 Metode *MTCNN* ..... 31
4.2.2 Metode *Dlib HOG based* ..... 32
*4.2.3* Metode *Dlib CNN based* ..... 33
4.2.4 Metode *YOLO* ..... 34
4.3 Implementasi Model Pengenalan Wajah ..... 37
4.3.1 Ekstraksi Fitur ..... 37
4.3.2 Pemisahan Dataset ..... 39
4.3.3 Klasifikasi *Naïve Bayes* ..... 39
4.3.4 Klasifikasi *Support Vector Machine* ..... 39
4.3.5 Klasifikasi *Random Forest* ..... 40
4.3.6 Proses *Training* dan *Testing* ..... 40
4.4 Implementasi Pengolahan Hasil Pengenalan ..... 41

4.4.1 Penyimpanan Data ..... 41
4.4.2 Penyusunan Rekapitulasi ..... 44
4.5 Implementasi Antarmuka ..... 46
**BAB 5 UJI COBA DAN EVALUASI ..... 49**
5.1 Lingkungan Uji Coba ..... 49
5.2 Data Uji Coba ..... 49
5.3 Skenario Uji Coba ..... 51
5.3.1 Skenario Uji Coba pada Model Deteksi Wajah ..... 51
5.3.2 Skenario Uji Coba pada Model Pengenalan Wajah ..... 59
5.3.3 Skenario Uji Coba pada Keseluruhan Sistem ..... 64
5.4 Evaluasi ..... 66
**BAB 6 KESIMPULAN DAN SARAN ..... 71**
6.1 Kesimpulan ..... 71
6.2 Saran ..... 72
**DAFTAR PUSTAKA ..... 73**
**BIODATA PENULIS ..... 75**

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

Gambar 2.1. Operasi *Covolution* [5]. ..... 8
Gambar 2.2. Operasi *Max Pooling* [5]. ..... 8
Gambar 2.3. Tiga tahapan pada *MTCNN* [6]. ..... 10
Gambar 2.4. Arsitektur *P-Net, R-Net,* dan *O-Net* pada *MTCNN* [6]. ..... 11
Gambar 2.5. Proses deteksi objek pada *YOLO* [9]. ..... 12
Gambar 2.6. Arsitektur *YOLO* [9]. ..... 13
Gambar 2.7. *Confusion matrix* [12]. ..... 14
Gambar 2.8. Grafik *precision-recall* [13]. ..... 16
Gambar 2.9. Grafik *interpolated precision* [13]. ..... 17
Gambar 3.1. Diagram alir sistem. ..... 20
Gambar 3.2. Contoh *frame* pada data video. ..... 21
Gambar 3.3. Contoh *ground truth* untuk evaluasi model deteksi wajah. ..... 22
Gambar 3.4. Diagram alir proses pelatihan dan pengujian pada model pengenalan wajah. ..... 23
Gambar 3.5. Contoh dataset untuk model pengenalan wajah. ..... 25
Gambar 3.6. Rancangan halaman utama. ..... 29
Gambar 3.7. Rancangan halaman detail. ..... 29
Gambar 3.8. Rancangan halaman *frame*. ..... 30
Gambar 5.1. Kurva *precision-recall* model deteksi wajah menggunakan MTCNN. ..... 53
Gambar 5.2. Contoh hasil deteksi wajah menggunakan *MTCNN*. ..... 53

Gambar 5.3. Kurva *precision-recall* model deteksi wajah menggunakan *Dlib HOG based.* .....54

Gambar 5.4. Contoh hasil deteksi wajah menggunakan *Dlib HOG based*. .....55

Gambar 5.5. Kurva *precision-recall* model deteksi wajah menggunakan *Dlib CNN based.* .....56

Gambar 5.6. Contoh hasil deteksi wajah menggunakan *Dlib CNN*. .....57

Gambar 5.7. Kurva precision-recall model deteksi wajah menggunakan *YOLO.* .....58

Gambar 5.8. Contoh hasil deteksi wajah menggunakan *YOLO*...58

Gambar 5.9. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *VGG16.* .....60

Gambar 5.10. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *Resnet50.* .....62

Gambar 5.11. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *Senet50.* .....63

# DAFTAR TABEL

Tabel 2.1. Tabel *precision* dan *recall.* ..... 16

Tabel 3.1. Jumlah gambar per label pada dataset pengenalan wajah. ..... 24

Tabel 3.2. Struktur tabel *files*. ..... 26

Tabel 3.3. Struktur tabel *frames*. ..... 27

Tabel 3.4. Struktur tabel *results*. ..... 27

Tabel 4.1. Daftar *endpoint* pada aplikasi web. ..... 46

Tabel 5.1. Spesifikasi data uji model deteksi wajah. ..... 50

Tabel 5.2. Spesifikasi data uji model pengenalan wajah. ..... 50

Tabel 5.3. Spesifikasi data uji keseluruhan sistem. ..... 51

Tabel 5.4. Hasil pengujian model deteksi wajah menggunakan metode MTCNN. ..... 52

Tabel 5.5. Hasil pengujian model deteksi wajah menggunakan metode *Dlib HOG based.* ..... 54

Tabel 5.6. Hasil pengujian model deteksi wajah menggunakan metode *Dlib CNN based.* ..... 55

Tabel 5.7. Hasil pengujian model deteksi wajah menggunakan metode *YOLO.* ..... 57

Tabel 5.8. Hasil pengujian model pengenalan wajah dengan arsitektur *VGG16*. ..... 60

Tabel 5.9. Hasil pengujian model pengenalan wajah menggunakan arsitektur *Resnet50.* ..... 61

Tabel 5.10. Hasil pengujian model pengenalan wajah menggunakan arsitektur *Senet50.* ..... 63

Tabel 5.11. Hasil pengujian waktu proses pada keseluruhan sistem. ..... 64

Tabel 5.12. Hasil uji coba keseluruhan sistem.....65
Tabel 5.13. Hasil deteksi per label pada uji coba keseluruhan sistem. .....65

# DAFTAR KODE SUMBER

Kode Sumber 4.1. Implementasi metode *MTCNN*. .....32

Kode Sumber 4.2. Implementasi metode *Dlib HOG based*. .....33

Kode Sumber 4.3. Implementasi metode *Dlib CNN based*. .....34

Kode Sumber 4.4. Implementasi metode *YOLO*. .....35

Kode Sumber 4.5. Fungsi *post-processing* pada metode *YOLO*. 36

Kode Sumber 4.6. Pembuatan area wajah berdasarkan *bounding box*.....37

Kode Sumber 4.7. *Preprocess* input pada ekstraksi fitur. .....37

Kode Sumber 4.8. Ekstraksi fitur dengan *VGG16*.....38

Kode Sumber 4.9. Ekstraksi fitur dengan *Resnet50*. .....38

Kode Sumber 4.10. Ekstraksi fitur dengan *Senet50*. .....38

Kode Sumber 4.11. Pemisahan dataset dengan *Cross Validation*. .....39

Kode Sumber 4.12. Implementasi metode *Naive Bayes*.....39

Kode Sumber 4.13. Implementasi metode *SVM*. .....40

Kode Sumber 4.14. Implementasi metode *Random Forest*. .....40

Kode Sumber 4.15. Proses *training* dan *testing* menggunakan metode *cross validation*. .....41

Kode Sumber 4.16. Implementasi pembuatan tabel *files*. .....42

Kode Sumber 4.17. Implementasi pembuatan tabel *frames*. .....42

Kode Sumber 4.18. Implementasi pembuatan tabel *results*. .....42

Kode Sumber 4.19. Implementasi proses penyimpanan data ke tabel *files*. .....43

Kode Sumber 4.20. Implementasi proses penyimpanan data ke tabel *frames*. ..... 43

Kode Sumber 4.21. Implementasi proses penyimpanan data ke tabel *results*. ..... 44

Kode Sumber 4.22. Implementasi perhitungan persentase kehadiran. ..... 45

Kode Sumber 4.23. Implementasi kode untuk menampilkan lama waktu pemrosesan. ..... 45

Kode Sumber 4.24. Implementasi kode untuk menampilkan hasil pengenalan pada suatu *frame*. ..... 46

Kode Sumber 4.25. Implementasi *endpoint* untuk mengunggah *file* dan menampilkan halaman utama. ..... 47

Kode Sumber 4.26. Implementasi *endpoint* untuk menampilkan hasil rekapitulasi. ..... 48

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Kehadiran mahasiswa merupakan salah satu hal yang dianggap penting dalam dunia perkuliahan. Terbukti dengan banyak universitas di Indonesia yang menjadikan kehadiran mahasiswa sebagai syarat untuk lulus dari mata kuliah yang diambil, tidak terkecuali di Institut Teknologi Sepuluh Nopember (ITS). Berdasarkan Peraturan Rektor Institut Teknologi Sepuluh Nopember Nomor 15 tahun 2018 tentang Peraturan Akademik Institut Teknologi Sepuluh Nopember Tahun 2018, mahasiswa wajib mengikuti proses pembelajaran setiap mata kuliah minimal 80% dari jumlah yang diselenggarakan dalam satu semester, bila tidak terpenuhi maka keikutsertaannya tidak diakui serta mendapat nilai E [1]. Adanya peraturan tersebut membuat banyak mahasiswa memutar otak untuk bisa lulus meskipun kehadirannya tidak memenuhi kriteria.

Banyak mahasiswa yang akhirnya melakukan kecurangan dengan memanipulasi kehadirannya, metode yang paling umum adalah "titip absen". Titip absen adalah salah satu metode memanipulasi kehadiran dengan meminta orang lain untuk mencatatkan kehadiran atas nama dirinya. Ditambah lagi, dengan sistem pencatatan kehadiran yang saat ini diterapkan, yaitu dengan kertas dan tanda tangan, kegiatan titip absen ini sangat mudah dilakukan. Selain titip absen, ada juga mahasiswa yang tetap masuk ke kelas, namun langsung meninggalkan ruangan setelah melakukan absensi meskipun kegiatan perkuliahan belum selesai. Hal seperti ini juga sangat sulit untuk dicegah dengan sistem pencatatan kehadiran yang konvensional.

Untuk mengatasi kecurangan-kecurangan yang ada pada metode pencatatan kehadiran yang konvensional, dalam tugas akhir ini akan dikembangkan pengenalan wajah untuk sistem monitoring kehadiran mahasiswa pada kegiatan perkuliahan

menggunakan metode berbasis pembelajaran mendalam (deep learning). Untuk implementasi tugas akhir ini, akan dibandingkan metode deteksi wajah *MTCNN, Dlib HOG based, Dlib CNN based, dan YOLO*. Selain itu akan dibandingkan juga model pengenalan wajah dengan ekstraksi fitur berbasis *Convolutional Neural Network* menggunakan arsitektur *VGG16, Resnet50, dan Senet50*, yang diikuti dengan klasifikasi menggunakan metode *Naïve Bayes, Support Vector Machine,* dan *Random Forest*.

## 1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mendeteksi area wajah pada data video?
2. Bagaimana membangun suatu *classifier* berbasis *deep learning* untuk mengidentifikasi wajah yang sudah dideteksi?
3. Bagaimana mengevaluasi performa dari sistem yang telah diimplementasikan?

## 1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini antara lain:

1. Dalam implementasi tugas akhir ini, hanya digunakan satu kamera dalam satu ruangan sehingga tidak semua orang dalam ruangan dapat terlihat atau terdeteksi.
2. Dataset yang digunakan untuk pengenalan wajah adalah wajah mahasiswa aktif di Departemen Informatika ITS yang mengikuti mata kuliah tertentu.
3. Implementasi program menggunakan bahasa python.

## 1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk mengembangkan pengenalan wajah menggunakan *deep learning* pada sistem monitoring kehadiran mahasiswa berbasis CCTV.

## 1.5 Manfaat

Manfaat diimplementasikannya tugas akhir ini antara lain:

1. Mempermudah proses monitoring kehadiran mahasiswa.
2. Mengurangi tingkat kecurangan yang terdapat pada metode pencatatan kehadiran konvensional.

## 1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

**1. Penyusunan proposal tugas akhir**

Proposal tugas akhir ini berisi tentang penjelasan mengenai pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri dari hal yang melatarbelakangi tugas akhir, rumusan masalah yang diangkat, batasan masalah yang ada, tujuan dan manfaat dari tugas akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung dalam pembuatan tugas akhir.

**2. Studi literatur**

Pada tahap ini akan dilakukan pencarian dan pembelajaran sejumlah literatur yang relevan dengan tugas akhir yang akan dikerjakan. Literatur dapat berupa buku, paper, jurnal ilmiah, atau artikel dari internet mengenai *Face Detection*, *Face Recognition, CNN,* dan *YOLO*.

**3. Analisis dan desain metode**

Pada tahap ini akan dilakukan eksplorasi terhadap metode-metode yang akan diimplementasikan. Metode yang akan diimplementasikan antara lain: *Deep Learning, YOLO*, dan *Transfer Learning*.

**4. Implementasi**

Tugas akhir ini akan diimplementasikan dalam bahasa python menggunakan IDE PyCharm. Selain itu akan digunakan juga library OpenCV, TensorFlow, dan Keras untuk mendukung proses *deep learning*.

**5. Pengujian dan evaluasi**

Pada tahap ini dilakukan pengujian dan evaluasi terhadap performa deteksi wajah dan model *classifier* yang telah dibuat. Kedua proses pengujian tersebut akan dilakukan menggunakan pengujian akurasi, presisi, *recall* dan *F- measure.*

**6. Penyusunan buku tugas akhir**

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, tinjauan pustaka, metode, implementasi, proses yang telah dilakukan, pengujian, evaluasi dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

## 1.7 Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

**BAB I. Pendahuluan**

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, Batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

**BAB II. Tinjauan Pustaka**

Bab ini menjelaskan beberapa teori yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

**BAB III. Analisis dan Perancangan Sistem**

Bab ini membahas mengenai perancangan sistem yang akan dibangun. Perancangan sistem meliputi perancangan data dan alur proses dari sistem itu sendiri.

**BAB IV. Implementasi**

Bab ini berisi implementasi dari perancangan sistem yang telah ditentukan sebelumnya.

**BAB V. Pengujian dan Evaluasi**

Bab ini membahas pengujian dari metode yang ditawarkan dalam tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

**BAB VI. Kesimpulan dan Saran**

Bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga membahas saran-saran untuk pengembangan sistem lebih lanjut.

**Daftar Pustaka**

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

**Lampiran**

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

*[Halaman ini sengaja dikosongkan]*

# BAB 2
# TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar tugas akhir ini.

## 2.1 *Deep Learning*

Pembelajaran mendalam atau *deep learning* merupakan bagian dari *machine learning* dan memanfaatkan *artificial neural network* yang terinspirasi oleh struktur dan fungsi otak manusia. *Deep learning* menggunakan beberapa lapisan (*multiple layer*) untuk mengekstrak fitur tingkat tinggi dari masukan yang diproses. *Deep learning* dapat digunakan untuk pembelajaran yang bersifat *supervised*, *semi-supervised*, atau bahkan *unsupervised*. Beberapa contoh model *deep learning* antara lain: *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), *Boltzmann Machine*, dan lain-lain [2].

## 2.2 *Convolutional Neural Network*

*Convolutional Neural Network* (CNN) [3] merupakan salah satu algoritma *Deep Learning* yang dirancang untuk mengolah data berbentuk dua dimensi seperti gambar. CNN dapat menerima input berupa gambar, menetapkan bobot dan bias pada setiap aspek atau objek pada gambar, dan dapat membedakan yang satu dengan yang lainnya [4].

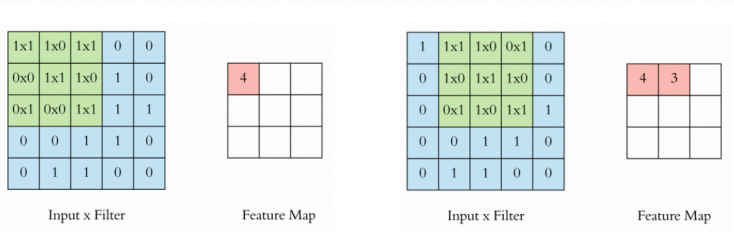
Setiap arsitektur CNN terdiri dari tiga layer utama, yaitu: *Convolution Layer*, *Pooling Layer*, dan *Fully Connected Layer*.

1. *Convolution Layer*

*Convolution Layer* merupakan lapisan pertama dan menjadi inti pada arsitektur CNN. Lapisan ini bertujuan untuk mengekstrak fitur seperti garis, sudut, atau yang lain dari sebuah gambar. Pada lapisan ini dilakukan operasi matematika terhadap matriks dari

gambar dan matriks filter, atau biasa disebut kernel, untuk menghasilkan *feature map*.

Operasi *convolution* dilakukan dengan menggeser matriks filter pada matriks input. Pada setiap pergeseran dilakukan operasi *element-wise matrix multiplication*, kemudian menjumlahkan hasilnya dan meletakkannya pada matriks *feature map* seperti pada Gambar 2.1 dibawah.

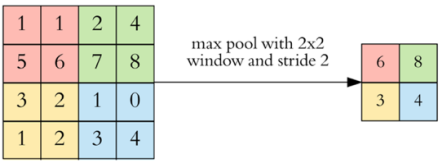



Gambar 2.1. Operasi *Covolution* [5].

2. *Pooling Layer*

Pada lapisan ini dilakukan operasi *pooling* untuk mereduksi dimensi dan jumlah parameter dimana keduanya dapat memperpendek waktu *training* dan mencegah adanya *overfitting*.

Jenis *pooling* yang paling umum adalah *max pooling*. *Max pooling* mengambil nilai tertinggi pada *pooling window*. Mirip dengan operasi konvolusi, operasi *max pooling* menggeser *pooling window* pada matriks input seperti pada Gambar 2.2.




Gambar 2.2. Operasi *Max Pooling* [5].

3. *Fully Connected Layer*

Pada lapisan inilah dilakukan proses klasifikasi menggunakan fungsi aktivasi tertentu. Input dari *fully connected layer* ini merupakan output dari lapisan konvolusi dan lapisan *pooling*. Perlu diingat bahwa output dari dua *layer* sebelumnya masih multidimensi, sedangkan *fully connected layer* memerlukan input berupa vektor satu dimensi, sehingga perlu dilakukan proses *flatten* agar bisa menjadi input untuk *fully connected layer*.

## 2.3 *Transfer Learning*

*Transfer Learning* merupakan sebuah teknik dalam *machine learning* dimana sebuah model di-*train* dan dikembangkan untuk suatu tugas tertentu kemudian digunakan kembali untuk tugas lain yang berhubungan. Model yang digunakan pada metode *transfer learning* biasa disebut dengan *pre-trained model*.

## 2.4 *Multi-task Cascaded Convolutional Networks*

*Multi-task Cascaded Convolutional Networks (MTCNN)* merupakan salah satu metode deteksi wajah berbasis *deep learning*. *MTCNN* diusulkan oleh Kaipeng Zhang dan yang lain dalam sebuah paper berjudul *"Join Face Detection and Alignment using Multi-task Cascaded Convolutional Networks"* [6]. Metode yang diusulkan memiliki tiga tahap utama untuk melakukan deteksi wajah dan deteksi *facial landmark*.

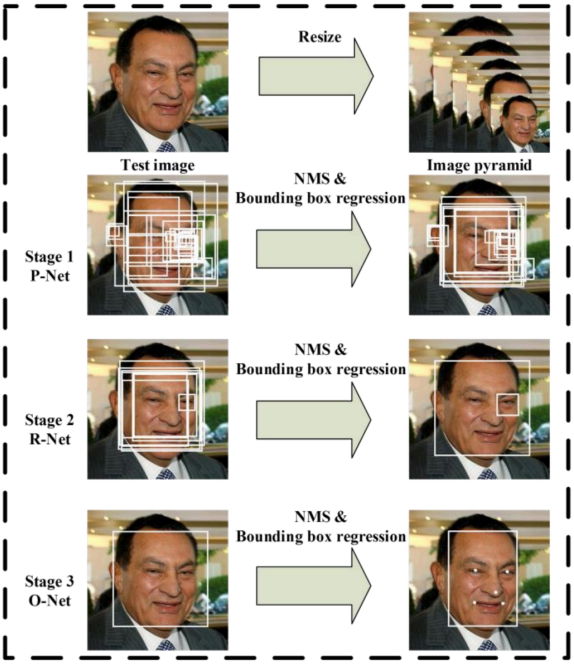
Seperti yang ditunjukkan pada Gambar 2.3, pertama input gambar diubah ke berbagai ukuran berbeda yang disebut *image pyramid* untuk kemudian menjadi input dari tiga tahapan sebagai berikut.

1. Pada tahap pertama, digunakan *fully convolutional network* yang disebut *Proposal Network (P-Net)* untuk mendapatkan area-area yang diusulkan dan vektor regresi *bounding box* dari masing-masing area. Vektor regresi yang diperoleh digunakan untuk melakukan kalibrasi pada area yang diusulkan. Setelah itu, dilakukan metode

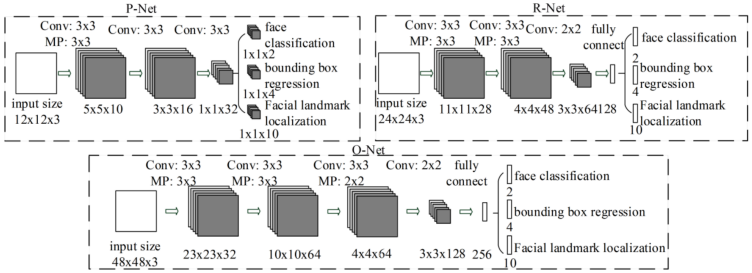
*non-maximum suppression (NMS)* untuk menggabungkan area-area yang sangat tumpeng tindih.

2. Semua area yang diusulkan kemudian dimasukkan ke *CNN* lain bernama *Refine Network (R-Net)* yang akan menolak banyak usulan area yang dinilai salah, melakukan kalibrasi dengan regresi *bounding box*, dan menggabungkan usulan area dengan *NMS*.
3. Tahap ketiga mirip dengan tahap kedua, namun *CNN* yang digunakan bernama *Output Network (O-Net)*. Tahap ini bertujuan untuk menggambarkan wajah secara lebih rinci dan juga menampilkan lima posisi *facial landmark*.

Arsitektur dari *P-Net, R-Net,* dan *O-Net* secara keseluruhan ditunjukkan pada Gambar 2.4.




Gambar 2.3. Tiga tahapan pada *MTCNN* [6].




Gambar 2.4. Arsitektur *P-Net, R-Net,* dan *O-Net* pada *MTCNN* [6].

## 2.5 *Dlib Face Detector*

*Dlib* merupakan *library* yang berisi berbagai algoritma *machine learning* dan berbagai macam alat untuk mengembangkan perangkat lunak yang kompleks. *Library dlib* menyediakan dua jenis model deteksi wajah, yaitu model deteksi wajah berbasis *Histogram of Oriented Gradients (HOG)* [7] dan model deteksi wajah berbasis *Convolutional Neural Network (CNN)*.

Pada model deteksi wajah berbasis *HOG*, deteksi wajah dilakukan dengan memakai fitur *HOG* yang diklasifikasi menggunakan *linear SVM*. Sedangkan pada model deteksi wajah berbasis *CNN*, deteksi wajah dilakukan dengan metode *CNN* dimana metode tersebut menghasilkan model yang lebih akurat dibandingkan dengan model deteksi wajah berbasis *HOG*. Meskipun lebih akurat, model deteksi wajah berbasis *CNN* membutuhkan lebih banyak daya komputasi sehingga perlu dijalankan pada *GPU* untuk mendapatkan kecepatan proses yang lebih cepat [8].

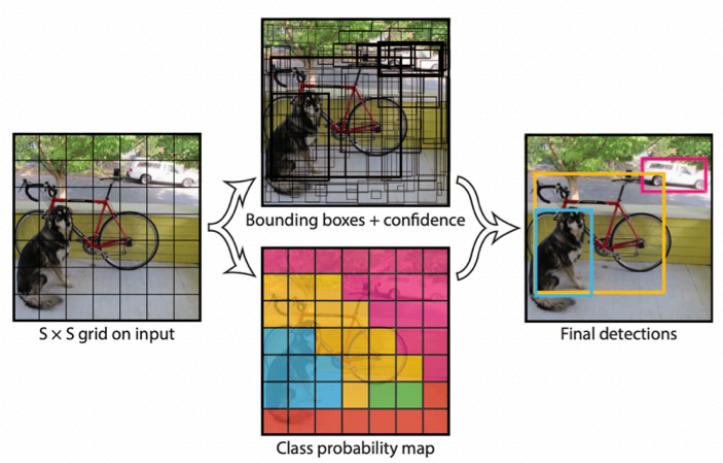
## 2.6 *YOLO: You Only Look Once*

*YOLO* [9] merupakan model deteksi objek secara *real time* berbasis *Convolutional Neural Network. YOLO* dapat mengenali objek dengan sangat cepat dan dapat berjalan pada 45 fps, bahkan versi cepat dari *YOLO* dapat berjalan pada 150 fps.

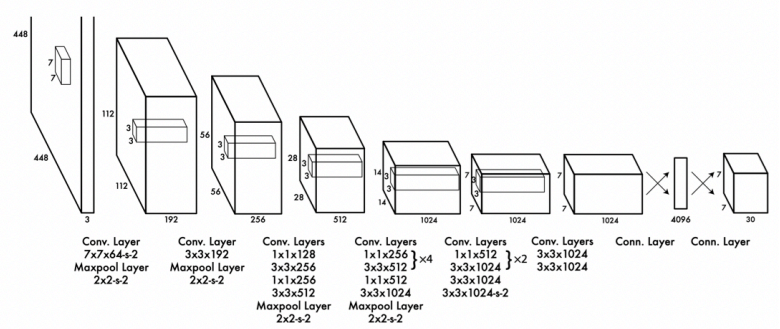
Berbeda dengan sistem deteksi objek yang lain, *YOLO* memiliki pendekatan yang berbeda pada sistem deteksi objek. *YOLO* menyatukan komponen-komponen pada model deteksi objek menjadi sebuah *single neural network*. Fitur-fitur yang ada pada seluruh gambar digunakan untuk memprediksi setiap *bounding box* dan mengklasifikasikannya ke seluruh kelas secara simultan.

Lebih detail, seperti pada Gambar 2.5, YOLO membagi input gambar menjadi *S x S grid*. Setiap *grid* akan memprediksi *B bounding box* beserta *confidence score* dari masing-masing *bounding box* dan *C* probabilitas kelas.

Arsitektur yang diadopsi pada *YOLO* terinspirasi dari model *GoogLeNet* untuk klasifikasi gambar. *YOLO* memiliki 24 *convolutional layer* yang diikuti dengan 2 *fully connected layer*. Berbeda dengan modul *inception* yang ada pada *GoogLeNet*, *YOLO* menggunakan 1 x 1 *reduction layer* dan diikuti dengan 3 x 3 *convolutional layer*. Untuk lebih detailnya dapat dilihat pada Gambar 2.6.




Gambar 2.5. Proses deteksi objek pada *YOLO* [9].




Gambar 2.6. Arsitektur *YOLO* [9].

## 2.7 *Keras VGGFace*

*Keras VGGFace* merupakan *library* untuk pengenalan wajah yang diimplementasikan dari model pengenalan wajah berbasis *CNN* bernama yang dikembangkan oleh Visual Geometry Group dari University of Oxford. Kode sumber dari *Keras VGGFace* dapat dilihat di https://github.com/rcmalli/keras-vggface.

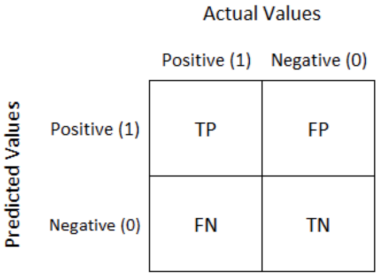
Pada *library Keras VGGFace* terdapat tiga model arsitektur yang dapat digunakan, diantaranya: *VGG16* yang diimplementasikan dari paper lama [10], *Resnet50* dan *Senet50* yang diimplementasikan dari paper baru [11].

## 2.8 *Confusion Matrix*

*Confusion Matrix* merupakan suatu ukuran yang dapat menunjukkan performa dari sebuah model klasifikasi pada serangkaian data uji yang diketahui nilai sebenarnya. Pada Gambar 2.7 ditunjukkan empat kombinasi nilai prediksi dan aktual yang berbeda.

*True Positive (TP)* menunjukkan jumlah nilai yang diprediksi positif dan nilai aktualnya juga positif. *False Positive (FP)* menunjukkan jumlah nilai yang diprediksi positif namun nilai aktualnya negatif. *True Negative (TN)* menunjukkan jumlah nilai yang diprediksi negative dan nilai aktualnya juga negatif. *False*

*Negative (FN)* menunjukkan jumlah nilai yang diprediksi negatif namun nilai aktualnya positif.




Gambar 2.7. *Confusion matrix* [12].

Nilai *TP, FP, FN,* dan *TN* dapat digunakan untuk menghitung beberapa metriks sebagai berikut.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

## 2.9 *Mean Average Precision (mAP)*

*Mean average precision (mAP)* merupakan metrik yang sering digunakan untuk mengukur performa dari sebuah model deteksi

objek. *Mean average precision* didapatkan dengan menghitung rata-rata nilai presisi untuk masing-masing nilai recall 0 sampai nilai recall 1.

Untuk menghitung *mAP* pada model deteksi objek, harus dihitung *intersection over union (IoU)* terlebih dahulu. *IoU* sendiri merupakan nilai yang menunjukkan tingkat kemiripan antara *bounding box* yang diprediksi dengan *ground truth bounding box* untuk mengevaluasi seberapa bagus hasil *bounding box* yang diprediksi. Nilai *IoU* berkisar dari 0 sampai 1, dimana semakin tinggi nilai *IoU* maka semakin bagus hasil deteksinya.

Setelah didapatkan nilai *IoU*, kemudian ditentukan nilai *IoU threshold*. Jika nilai *IoU* lebih besar atau sama dengan *IoU threshold*, maka *bounding box* yang diprediksi termasuk *TP (True Positive)*. Jika nilai *IoU* lebih kecil dari *IoU threshold*, maka *bounding box* yang diprediksi termasuk *FP (False Positive)*. Jika *ground truth bounding box* ada di dalam gambar tetapi model gagal untuk mendeteksi objek tersebut, maka termasuk *FN (False Negative)*. Sedangkan nilai *TN (True Negative)* tidak digunakan dalam evaluasi model deteksi objek, sehingga nilai *TN* dihiraukan.

Setelah itu, dihitung nilai *average precision (AP)*. Sebagai contoh perhitungan *AP*, terdapat 5 buah apel di dalam sebuah dataset. Diambil semua prediksi yang dibuat untuk apel pada semua gambar dan diurutkan menurun berdasarkan *confidence level* untuk kemudian disusun seperti Tabel 2.1. Setelah itu hasilnya diubah menjadi grafik *precision-recall* seperti Gambar 2.8. Kemudian dihitung *interpolated precision* dengan menghitung *precision* maksimum untuk setiap level *recall*, sehingga grafik berubah menjadi seperti Gambar 2.9.

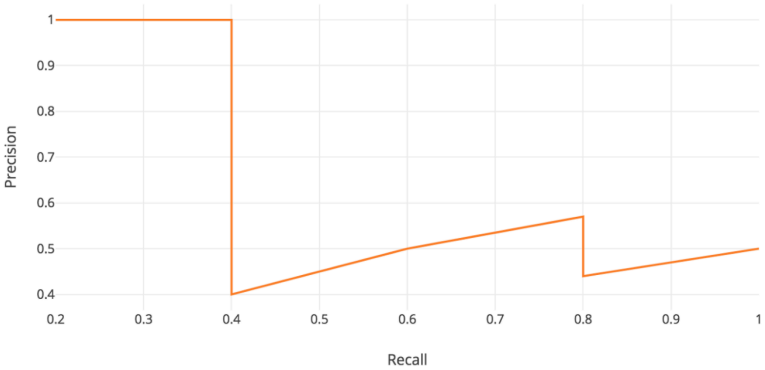
Nilai *AP* didapatkan dengan menghitung luas dibawah kurva *precision-recall*. Hal ini dilakukan dengan membagi nilai *recall* dari 0 sampai 1.0 menjadi 11 bagian: 0.1, 0.2, …, 0.9, 1.0. Kemudian dihitung rata-rata nilai *interpolated precision* untuk

setiap nilai *recall* tersebut seperti pada Persamaan (5). Nilai *mAP* merupakan rata-rata *AP* yang dihitung pada semua kelas [13].

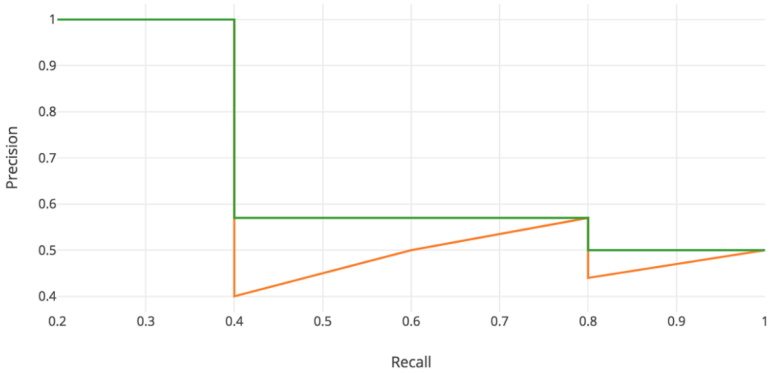
$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,0.2,\ldots,0.9,1.0\}} Pinter(r) \tag{5}$$

Tabel 2.1. Tabel *precision* dan *recall.*

| Rank | TP/FP | Precision | Recall |
|---|---|---|---|
| 1 | TP | 1.0 | 0.2 |
| 2 | TP | 1.0 | 0.4 |
| 3 | FP | 0.67 | 0.4 |
| 4 | FP | 0.5 | 0.4 |
| 5 | FP | 0.4 | 0.4 |
| 6 | TP | 0.5 | 0.6 |
| 7 | TP | 0.57 | 0.8 |
| 8 | FP | 0.5 | 0.8 |
| 9 | FP | 0.44 | 0.8 |
| 10 | TP | 0.5 | 1.0 |




Gambar 2.8. Grafik *precision-recall* [13].




Gambar 2.9. Grafik *interpolated precision* [13].

*[Halaman ini sengaja dikosongkan]*

# BAB 3
# ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijabarkan analisis dan desain sistem yang digunakan selama proses penyelesaian tugas akhir. Bab ini juga menjelaskan tentang analisis metode yang digunakan secara umum pada sistem.

## 3.1 Perancangan Sistem

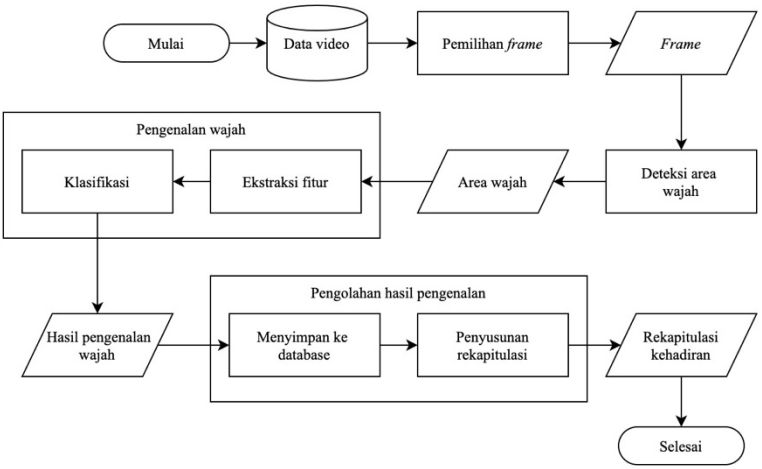
Pada tugas akhir ini akan dibangun sebuah model pengenalan wajah pada data video yang akan diimplementasikan pada sistem monitoring kehadiran berbasis CCTV. Sistem monitoring kehadiran yang akan dibangun memiliki tiga proses utama: deteksi area wajah, pengenalan wajah yang terdeteksi, dan pengolahan data hasil pengenalan. Gambaran dari semua proses ditunjukkan pada Gambar 3.1.

Tahap pertama yang dilakukan adalah deteksi area wajah. Pada tahap ini setiap wajah yang ada di dalam video dideteksi dan ditentukan masing-masing posisinya di dalam video. Tahap ini menerima input berupa *frame* dari data video dan mengeluarkan output berupa koordinat *bounding box* dari masing-masing area wajah yang terdeteksi.

Tahap selanjutnya adalah pengenalan wajah. Tahap ini terdiri dari dua proses: ekstraksi fitur dan klasifikasi. Proses yang pertama dilakukan adalah proses ekstraksi fitur. Pada proses ini dilakukan pengambilan fitur pada masing-masing wajah yang terdeteksi di tahap sebelumnya. Kemudian pada proses selanjutnya dilakukan klasifikasi pada masing-masing wajah berdasarkan fitur-fitur yang telah didapatkan pada proses sebelumnya. Output dari tahap ini adalah hasil pengenalan dari masing-masing wajah.

Setelah dilakukan pengenalan pada masing-masing wajah, tahap selanjutnya adalah pengolahan hasil pengenalan. Pada tahap ini hasil pengenalan dari masing-masing wajah disimpan ke

*database* untuk selanjutnya dilakukan penyusunan rekapitulasi. Tahap ini mengeluarkan output berupa rekapitulasi kehadiran.




Gambar 3.1. Diagram alir sistem.

## 3.2 Pengumpulan Data

Tahap pertama dilakukan proses pengumpulan data. Data yang dikumpulkan merupakan data video kegiatan perkuliahan di dalam kelas di Departemen Informatika ITS dan diambil menggunakan kamera CCTV. Data video yang diambil berjumlah 6 video dengan total durasi 40 menit 9 detik. Semua video memiliki resolusi 1920×1088 dan memiliki *frame rate* sebesar 20 fps. Contoh frame pada data video dapat dilihat pada Gambar 3.2.

## 3.3 Perancangan Proses

Tahapan-tahapan proses yang akan dilakukan untuk mengimplementasikan sistem monitoring kehadiran pada tugas akhir ini meliputi proses pemilihan *frame*, proses deteksi wajah, proses pengenalan wajah, dan proses pengolahan hasil pengenalan.


Gambar 3.2. Contoh *frame* pada data video.

### 3.3.1 Proses Pemilihan *Frame*

Untuk mengurangi komputasi, tidak semua *frame* dari data video diproses. Oleh karena itu, perlu dilakukan pemilihan *frame* mana yang akan diproses. Pada tugas akhir ini, *frame* yang akan diproses adalah *frame* dengan kelipatan *k*, dimana nilai *k* bergantung pada waktu yang dibutuhkan untuk memproses satu *frame*. Sebagai contoh, apabila waktu yang dibutuhkan untuk memproses satu *frame* adalah 1 detik dan video yang diproses memiliki 20 *frame* per detik, maka *frame* yang akan diproses adalah *frame* ke-1, ke-21, ke-41, ke-61, dan seterusnya.

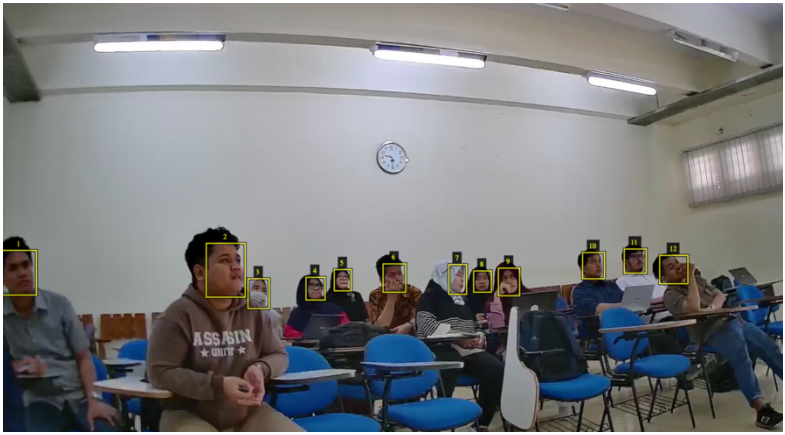
### 3.3.2 Proses Deteksi Wajah

Tahap selanjutnya adalah proses perancangan model deteksi wajah. Dalam tugas akhir ini, akan digunakan empat metode deteksi wajah: *Multi-task Cascaded Convolutional Networks* (MTCNN), *Dlib HOG based, Dlib CNN based, dan YOLO.* Keempat metode tersebut diimplementasikan terhadap dataset yang dihasilkan pada tahap pertama. Hasil deteksi dari masing-masing metode akan dibandingkan dengan *ground truth* dari dataset yang telah dibuat untuk kemudian dibandingkan menggunakan metode *Mean Average Precission (mAP)*. Selain itu

akan dibandingkan pula kecepatan *processing* yang dilakukan masing-masing metode.

#### 3.3.2.1 Penyusunan *Ground Truth* untuk Deteksi Wajah

Untuk evaluasi performa masing-masing metode, penulis membuat dataset berisi 60 gambar yang merupakan potongan-potongan *frame* dari dataset video yang telah disebutkan pada subbab 3.2. Pada masing-masing gambar, ditentukan area-area wajah yang seharusnya dideteksi, yaitu semua area wajah yang terlihat pada gambar, baik terlihat dari sisi depan ataupun dari sisi samping. Kemudian dilakukan proses pemberian anotasi pada area-area wajah tersebut. Proses pemberian anotasi dibantu menggunakan *tools* bernama *VGG Image Annotator* [14]. Contoh salah satu gambar pada dataset yang telah dilakukan proses anotasi dapat dilihat pada Gambar 3.3.



Gambar 3.3. Contoh *ground truth* untuk evaluasi model deteksi wajah.

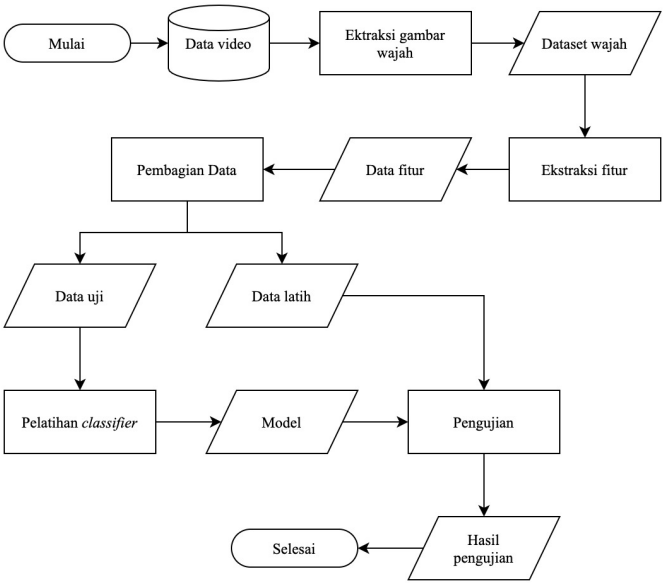
#### 3.3.2.2 *Pre-trained Model* untuk Deteksi Wajah

Pada tugas akhir ini, metode deteksi wajah MTCNN diimplementasikan menggunakan model yang disediakan pada *library* mtcnn. *Library* mtcnn merupakan implementasi dari metode MTCNN yang dibangun menggunakan *Keras* dan *Python*

*3.4*. Untuk metode *Dlib HOG based dan Dlib CNN based* diimplementasikan menggunakan model yang disediakan pada *library* dlib. Sedangkan untuk metode *YOLO* diimplementasikan menggunakan *OpenCV DNN module* dan memakai *pre-trained model* dari *YOLOFace* [15] yang dilatih pada dataset "*WIDER FACE: A Face Detection Benchmark"*.

### 3.3.3 Proses Pengenalan Wajah

Setelah model deteksi wajah dibuat, dilakukan proses perancangan model pengenalan wajah. Model pengenalan wajah dalam tugas akhir ini dibagi menjadi dua proses utama, yaitu proses ekstraksi fitur dan proses klasifikasi. Pada proses ekstraksi fitur dilakukan pengambilan fitur yang ada pada setiap wajah. Kemudian fitur-fitur yang telah didapatkan dilakukan klasifikasi untuk memperoleh hasil pengenalan pada masing-masing wajah.




Gambar 3.4. Diagram alir proses pelatihan dan pengujian pada model pengenalan wajah.

Proses pelatihan dan pengujian model pengenalan wajah dilakukan pada data fitur yang dihasilkan dari proses ekstraksi fitur pada dataset wajah, dimana dataset wajah merupakan kumpulan gambar wajah yang diekstrak dari dataset video. Diagram alir dari proses pelatihan dan pengujian model pengenalan secara lengkap ditunjukkan pada Gambar 3.4.

#### 3.3.3.1 Penyusunan *Ground Truth* untuk Pengenalan Wajah

Dataset yang akan digunakan pada tahap ini berupa gambar wajah dari hasil ekstraksi dataset video. Proses ekstraksi dilakukan menggunakan metode deteksi wajah yang memiliki performa paling baik dari hasil evaluasi pada tahap perancangan model deteksi wajah. Kemudian dilakukan proses pemberian label pada setiap data wajah. Label yang terdapat pada dataset berjumlah 13. Jumlah gambar pada masing-masing label ditunjukkan pada Tabel 3.1. Contoh gambar pada salah satu label dapat dilihat pada Gambar 3.5.

Tabel 3.1. Jumlah gambar per label pada dataset pengenalan wajah.

| Label | Jumlah Gambar |
|---|---|
| Bu Nanik | 228 |
| Face 1 | 319 |
| Face 2 | 413 |
| Face 3 | 319 |
| Face 4 | 355 |
| Face 5 | 220 |
| Face 6 | 60 |
| Face 7 | 454 |
| Face 8 | 353 |
| Face 9 | 254 |

| | |
|---|---|
| Face 10 | 210 |
| Face 11 | 217 |
| Face 12 | 106 |

Gambar 3.5. Contoh dataset untuk model pengenalan wajah.

Dataset dipisah menjadi dua: data *training* dan data *testing*. Pemisahan dataset dilakukan agar model yang dihasilkan memiliki generalisasi yang baik dalam melakukan klasifikasi data. Dataset dipisah menggunakan metode *k-fold cross validation* dan dibantu dengan fungsi `StratifiedKfold()` dari *sklearn*. Parameter n_splits bernilai 10 menunjukkan bahwa dataset dibagi menjadi 10 bagian dengan pembagian data *testing* sebesar 1/10 bagian dan data *training* sebesar 9/10 bagian, kemudian dilakukan perulangan sebanyak 10 kali.

#### 3.3.3.2 *Pre-trained Model* untuk Ekstraksi Fitur

Setelah dataset sudah siap, dilakukan proses ekstraksi fitur pada setiap label dari dataset wajah. Proses ekstraksi fitur dilakukan menggunakan metode berbasis CNN. Sedangkan arsitektur yang akan digunakan adalah *VGG16, Resnet50,* dan *Senet50*. Untuk implementasi dari metode tersebut, digunakan *pre-trained model* yang disediakan pada pustaka *Keras VGGFace*.

#### 3.3.3.3 Pembuatan Model untuk Pengenalan Wajah

Fitur yang dihasilkan dari proses ekstraksi fitur kemudian dilakukan proses klasifikasi. Metode klasifikasi yang akan digunakan antara lain *Naïve Bayes, Support Vector Machine (SVM),* dan *Random Forest*. Hasil model dari ketiga metode klasifikasi tersebut selanjutnya akan dibandingkan untuk menentukan metode klasifikasi yang paling baik digunakan pada tugas akhir ini.

### 3.3.4 Proses Pengolahan Hasil Pengenalan

Pada subbab ini akan dijelaskan mengenai proses pengolahan hasil pengenalan. Dalam tugas akhir ini, proses pengolahan hasil pengenalan dibagi menjadi dua: proses penyimpanan data dan proses penyusunan rekapitulasi.

#### 3.3.4.1 Penyimpanan Data

Tahap pertama pada proses pengolahan hasil pengenalan adalah penyimpanan data. Semua hasil pengenalan dari setiap *frame* disimpan pada sebuah *database*. Pada tugas akhir ini, *database* yang digunakan adalah *MySQL*. Adapun data yang akan disimpan antara lain: nama *file* video yang diproses, nomor *frame* yang diproses, *bounding box* dari area wajah yang dikenali, dan hasil pengenalan. Untuk itu, dibuat *database* dengan 3 tabel, yaitu tabel *files* untuk menyimpan data *file* apa saja yang sudah diproses, tabel *frames* untuk menyimpan daftar *frame* yang sudah diproses, dan tabel *results* untuk menyimpan data hasil pengenalan pada setiap *frame*. Struktur masing-masing tabel dapat dilihat pada Tabel 3.2, Tabel 3.3, dan Tabel 3.4.

Tabel 3.2. Struktur tabel *files*.

| Nama Kolom | Tipe Data | Keterangan |
|---|---|---|
| filename | VARCHAR(255) | Menyimpan nama file yang diproses |

| | | |
|---|---|---|
| started_at | TIMESTAMP | Menyimpan waktu proses dimulai |
| finished_at | DATETIME | Menyimpan waktu proses selesai |

Tabel 3.3. Struktur tabel *frames*.

| Nama Kolom | Tipe Data | Keterangan |
|---|---|---|
| filename | VARCHAR(255) | Menyimpan nama file yang diproses |
| frame_number | INT | Menyimpan nomor *frame* yang diproses |
| path | VARCHAR(255) | Menyimpan *filepath* dari *frame* |

Tabel 3.4. Struktur tabel *results*.

| Nama Kolom | Tipe Data | Keterangan |
|---|---|---|
| filename | VARCHAR(255) | Menyimpan nama file yang diproses |
| frame_number | INT | Menyimpan nomor *frame* yang diproses |
| left_side | INT | Menyimpan nilai sisi kiri dari *bounding box* |
| top_side | INT | Menyimpan nilai sisi atas dari *bounding box* |
| right_side | INT | Menyimpan nilai sisi kanan dari *bounding box* |
| bottom_side | INT | Menyimpan nilai sisi bawah dari *bounding box* |

| | | |
|---|---|---|
| label | VARCHAR(50) | Menyimpan label hasil pengenalan |
| confidence | DOUBLE | Menyimpan *confidence score* dari hasil pengenalan |

#### 3.3.4.2 Penyusunan Rekapitulasi

Tahap kedua setelah proses penyimpanan data adalah penyusunan rekapitulasi. Pada tahap ini, akan dilakukan perhitungan untuk mendapatkan persentase dari setiap label yang dikenali pada video yang diproses. Selain itu, pada hasil rekapitulasi akan ditampilkan juga lama waktu pemrosesan video dan hasil pengenalan pada masing-masing *frame* yang diproses.

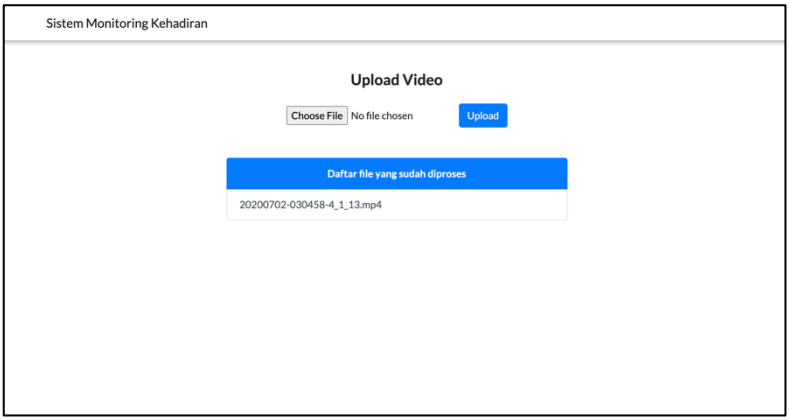
## 3.4 Perancangan Antarmuka

Pada tugas akhir ini, dibuat antarmuka untuk memudahkan penggunaan aplikasi. Antarmuka yang dibuat berupa aplikasi web yang terdiri dari 3 halaman, yaitu: halaman utama, halaman detail, dan halaman *frame*.

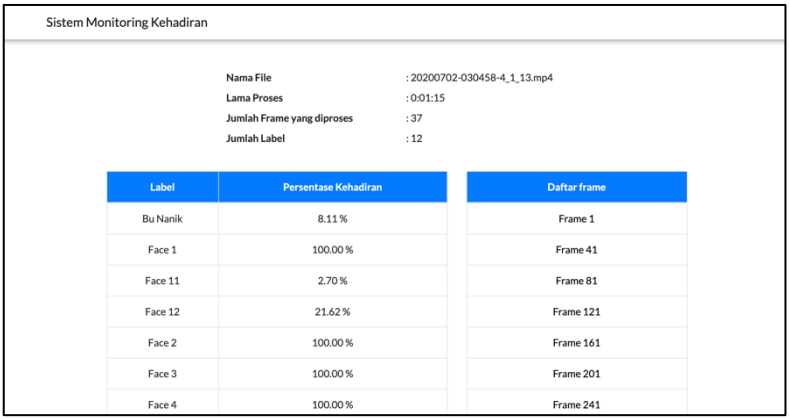
Pada halaman utama, disediakan *form* untuk mengunggah *file* video yang akan diproses. Selain itu, halaman ini juga menampilkan daftar nama *file* yang sudah diunggah dan sudah diproses. Ketika pengguna melakukan klik pada nama *file* yang tersedia, maka pengguna akan diarahkan menuju halaman detail. Rancangan halaman utama dapat dilihat pada Gambar 3.6.

Pada halaman detail, ditampilkan detail-detail hasil proses dari *file* yang dipilih. Detail yang ditampilkan berupa nama *file*, lama proses, jumlah *frame* yang diproses, dan jumlah label yang dikenali. Selain itu, ditampilkan juga persentase kehadiran dari masing-masing label dan daftar *frame* yang diproses. Ketika pengguna melakukan klik pada salah satu *frame*, maka pengguna

akan diarahkan menuju halaman *frame*. Rancangan halaman detail ditunjukkan pada Gambar 3.7.



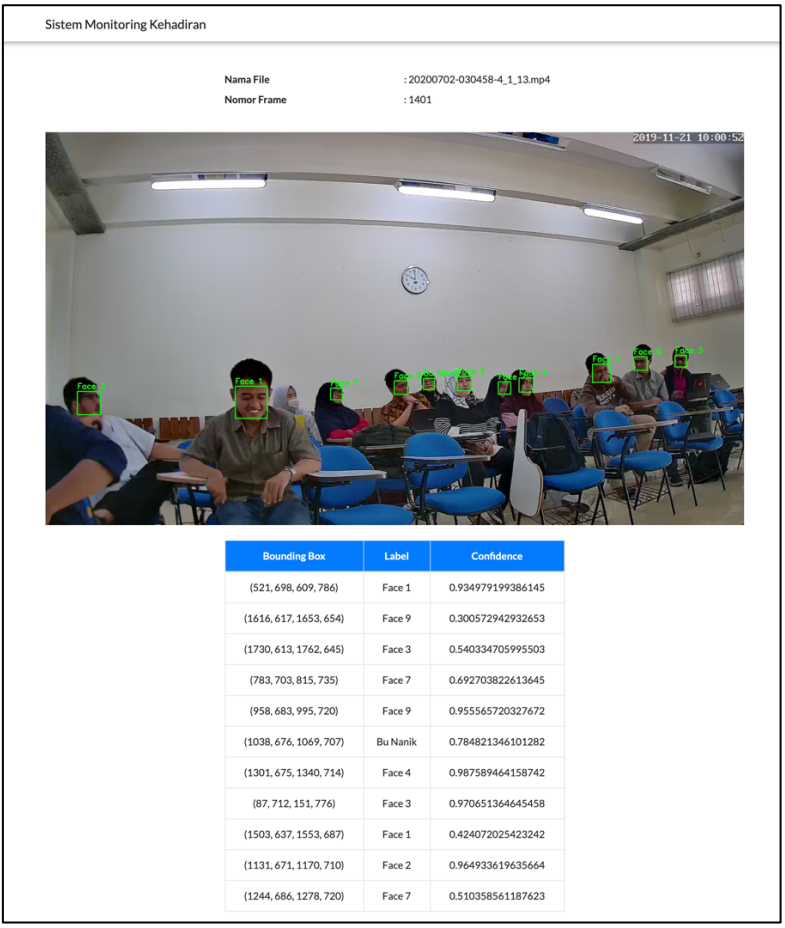

Gambar 3.6. Rancangan halaman utama.




Gambar 3.7. Rancangan halaman detail.

Pada halaman *frame*, ditampilkan hasil pengenalan secara lebih rinci dari *frame* yang dipilih. Pada halaman ini ditampilkan gambar *frame* yang telah diproses dan juga tabel hasil pengenalan

yang berisi posisi *bounding box*, label, dan *confidence score*. Rancangan halaman *frame* dapat dilihat pada Gambar 3.8.




Gambar 3.8. Rancangan halaman *frame*.

# BAB 4
# IMPLEMENTASI

Pada bab ini akan dijelaskan proses implementasi pada tugas akhir ini. Pada bab ini juga akan dirinci perangkat dan *tools* yang digunakan selama proses pengerjaan tugas akhir.

## 4.1 Lingkungan Implementasi

Implementasi sistem pada tugas akhir ini dilakukan pada perangkat keras dan sistem operasi sebagai berikut.

- Processor: 1.4 GHz Quad-Core Intel Core i5
- RAM: 8 GB 2133 MHz LPDDR3
- GPU: Intel Iris Plus Graphics 645 1536 MB
- Sistem Operasi: macOS Catalina 10.15.5

## 4.2 Implementasi Model Deteksi Wajah

Pada subbab ini akan dijelaskan proses implementasi model deteksi wajah berdasarkan perancangan yang dijelaskan pada subbab 3.3.2.

### 4.2.1 Metode *MTCNN*

Pada tugas akhir ini, implementasi metode deteksi wajah *MTCNN* dibantu menggunakan pustaka *mtcnn* yang dibangun menggunakan *Keras* dan *Python 3.4*. Untuk memudahkan implementasi, dibuat fungsi `detect_faces()` seperti yang ditunjukkan baris 5-16 pada Kode Sumber 4.1. Fungsi tersebut menerima input berupa *image array*. Pada baris ke-6, dijalankan fungsi `detect_faces` dari *library mtcnn* untuk melakukan proses deteksi wajah dan mendapatkan daftar *bounding box* dari area wajah yang terdeteksi beserta *confidence score* dari masing-masing *bounding box*. Kemudian pada baris 8-14 hasil deteksi wajah di-*convert* sesuai dengan kebutuhan.

```
1.  from mtcnn.mtcnn import MTCNN
2.
3.  face_detector = MTCNN()
4.
5.  def detect_faces(frame):
6.      faces = face_detector.detect_faces(frame)
7.
8.      boxes = []
9.      for face in faces:
10.         x, y, width, height = face['box']
11.         boxes.append({
12.             'box': [x, y, x + width, y + height],
13.             'confidence': face['confidence']
14.         })
15.
16.     return boxes
```

Kode Sumber 4.1. Implementasi metode *MTCNN*.

### 4.2.2 Metode *Dlib HOG based*

Implementasi metode *Dlib HOG based* pada tugas akhir ini dibantu menggunakan *library dlib*. Implementasi metode ini ditunjukkan pada Kode Sumber 4.2. Pada baris ke-3, dilakukan inisialisasi objek *face detector* dari *library dlib*. Sama seperti implementasi metode *MTCNN*, dibuat juga fungsi `detect_faces()` yang menerima input berupa *image array*. Pada baris ke-6, dijalankan fungsi run dari *face detector* dengan parameter pertama berupa *image array* yang akan diproses, parameter kedua menunjukkan berapa kali *upsample* dilakukan pada *image* yang diproses, dan parameter ketiga merupakan *detection threshold* dimana nilai negatif akan menghasilkan lebih banyak hasil deteksi dan nilai positif akan menghasilkan lebih sedikit hasi deteksi. Pada tugas akhir ini, nilai 1 digunakan untuk parameter kedua dan nilai -1 digunakan untuk parameter ketiga. Kemudian hasil deteksi berupa daftar *bounding box* beserta masing-masing *confidence score*-nya dilakukan proses *converting* sesuai dengan format yang dibutuhkan seperti pada baris ke 8-14.

```
1.  import dlib
2.
3.  face_detector = dlib.get_frontal_face_detector()
4.
5.  def detect_faces(frame):
6.      dets, scores, idx = face_detector.run(frame, 1,
    -1)
7.
8.      boxes = []
9.      for i, face in enumerate(dets):
10.         left, top, right, bottom = face.left(), face
    .top(), face.right(), face.bottom()
11.         boxes.append({
12.             'box': [left, top, right, bottom],
13.             'confidence': scores[i]
14.         })
15.
16.     return boxes
```

Kode Sumber 4.2. Implementasi metode *Dlib HOG based.*

### *4.2.3* Metode *Dlib CNN based*

Sama seperti implementasi metode *Dlib HOG based*, implementasi metode *Dlib CNN based* juga dibantu dengan *library dlib*. Implementasi dari metode ini dapat dilihat pada Kode Sumber 4.3. Pada baris ke-3 dan ke-4, dilakukan *load* pada *pre-trained model* yang disediakan *library dlib*. Kemudian dibuat fungsi `detect_faces()` yang menerima parameter berupa *image array* dari gambar yang akan dideteksi. Pada baris ke-7, dijalankan fungsi untuk deteksi wajah dengan parameter pertama berupa *image array* yang akan diproses dan parameter kedua berupa jumlah iterasi *upsample* yang akan dilakukan pada *image* yang diproses. Pada tugas akhir ini, *upsample* pada *image* dilakukan sebanyak satu kali. Sama seperti pada metode yang lain, hasil deteksi berupa daftar *bounding box* beserta masing-masing *confidence score*-nya dilakukan proses *converting* sesuai dengan format yang dibutuhkan, proses ini ditujukkan pada baris 9-16.

```
1.  import dlib
2.
3.  pretrained_model = 'mmod_human_face_detector.dat'
4.  face_detector = dlib.cnn_face_detection_model_v1(pretr
    ained_model)
5.
6.  def detect_faces(frame):
7.      dets = face_detector(frame, 1)
8.
9.      boxes = []
10.     for face in dets:
11.         left, top, right, bottom = face.rect.left(
12.         ), face.rect.top(), face.rect.right(), face.re
    ct.bottom()
13.         boxes.append({
14.             'box': [left, top, right, bottom],
15.             'confidence': face.confidence
16.         })
17.
18.     return boxes
```

Kode Sumber 4.3. Implementasi metode *Dlib CNN based.*

### 4.2.4 Metode *YOLO*

Pada tugas akhir ini, metode *YOLO* diimplementasikan menggunakan *OpenvCV DNN module* dan *pre-trained* model yang dilatih pada dataset "*WIDER FACE: A Face Detection Benchmark"*. Baris ke-12 pada Kode Sumber 4.4 menunjukkan kode untuk *load pre-trained* model. Sama seperti metode lain, dibuat fungsi `detect_faces()` yang menerima masukan berupa *image array* seperti yang ditunjukkan pada baris 20-28. Pada baris ke-21, masukan *frame* yang berupa *image array* di ubah menjadi *blob* untuk kemudian dijadikan input pada *network* seperti pada baris ke-23. Kemudian pada baris ke-24 dijalankan fungsi forward untuk mendapatkan keluaran dari *output layer* berupa daftar *bounding box* beserta *confidence score*-nya dan pada baris ke-26 dijalankan *post-processing* dengan memanggil fungsi `post_process()`.

```
1.  import cv2
2.  import numpy as np
3.
4.  config_file = 'yolov3-face.dat'
5.  model_weights = 'yolov3-wider_16000.weights'
6.  CONF_THRESHOLD = 0.5
7.  NMS_THRESHOLD = 0.4
8.  IMG_WIDTH = 416
9.  IMG_HEIGHT = 416
10.
11. # Load model
12. net = cv2.dnn.readNetFromDarknet(config_file, model_w
    eights)
13. net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)

14. net.setPreferableTarget(cv2.dnn.DNN_TARGET_OPENCL_FP1
    6)
15.
16. # Get the names of the output layers
17. layers_names = net.getLayerNames()
18. output_layers = [layers_names[i[0] - 1] for i in net.
    getUnconnectedOutLayers()]
19.
20. def detect_faces(frame):
21.     blob = cv2.dnn.blobFromImage(frame, 1 / 255, (IMG
    _WIDTH, IMG_HEIGHT), [0, 0, 0], 1, crop=False)
22.
23.     net.setInput(blob)
24.     outs = net.forward(output_layers)
25.
26.     boxes = post_process(frame, outs, CONF_THRESHOLD,
     NMS_THRESHOLD)
27.
28.     return boxes
```

Kode Sumber 4.4. Implementasi metode *YOLO*.

Pada fungsi post_process, *bounding box* yang memiliki *confidence score* dibawah *threshold* akan dihilangkan dan juga akan dihilangkan *bounding box* yang *redundant* dengan metode

*Non Maximum Suppression*. Pada tugas akhir ini, nilai *confidence threshold* yang dipakai adalah 0,5 dan *NMS threshold*-nya 0,4.

```
1.  def post_process(frame, outs, conf_thr, nms_thr):
2.    frame_height = frame.shape[0]
3.    frame_width = frame.shape[1]
4.
5.    confidences = []
6.    boxes = []
7.    for out in outs:
8.      for det in out:
9.        scores = det[5:]
10.       class_id = np.argmax(scores)
11.       confidence = scores[class_id]
12.       if confidence > conf_thr:
13.         center_x = int(det[0] * frame_width)
14.         center_y = int(det[1] * frame_height)
15.         width = int(det[2] * frame_width)
16.         height = int(det[3] * frame_height)
17.         left = int(center_x - width / 2)
18.         top = int(center_y - height / 2)
19.         confidences.append(float(confidence))
20.         boxes.append([left, top, width, height])
21.
22.   indices = cv2.dnn.NMSBoxes(boxes, confidences,
    conf_thr, nms_thr)
23.
24.   result_boxes = []
25.   for i in indices:
26.     i = i[0]
27.     box = boxes[i]
28.     left, top, width, height = box
29.
30.     result_boxes.append({
31.       'box': [left, top, left+width, top+height],
32.       'confidence': confidences[i]
33.     })
34.
35.   return result_boxes
```

Kode Sumber 4.5. Fungsi *post-processing* pada metode *YOLO*.

## 4.3 Implementasi Model Pengenalan Wajah

Pada subbab ini akan dijelaskan proses implementasi model pengenalan wajah berdasarkan perancangan yang dijelaskan pada subbab 3.3.3.

### 4.3.1 Ekstraksi Fitur

Sebelum dilakukan ekstraksi fitur, dilakukan proses untuk membuat *image array* untuk masing-masing area wajah berdasarkan *bounding box* yang didapatkan dari proses deteksi wajah. Untuk itu dilakukan pemotongan pada *frame* berdasarkan masing-masing *bounding box* seperti yang ditunjukkan pada Kode Sumber 4.6.

```
1.  faces = []
2.  for box in boxes:
3.      left, top, right, bottom = box
4.      faces.append(frame[top:top+(bottom-
    top), left:left+(right-left)])
```

Kode Sumber 4.6. Pembuatan area wajah berdasarkan *bounding box*.

Kemudian dilakukan *pre-processing* pada masing-masing area wajah. Setiap area wajah di-*resize* menjadi 224 × 224. Fungsi *pre-processing* ditunjukkan pada Kode Sumber 4.7.

```
1.  def preprocess(image, image_size=(224, 224)):
2.      x = cv2.resize(image, image_size, interpolation=c
    v2.INTER_AREA)
3.
4.      return x
5.
6.  faces = [preprocess(face) for face in faces]
```

Kode Sumber 4.7. *Preprocess* input pada ekstraksi fitur.

Setelah dilakukan *pre-precessing*, dilakukan proses ekstraksi fitur yang dibantu dengan pustaka *VGGFace*. Parameter yang

diatur pada metode ini yaitu `model`, `include_top`, `input_shape`, dan `pooling`. Parameter `model` menunjukkan arsitektur atau model yang akan digunakan, pada tugas akhir ini akan digunakan *vgg16*, *resnet50*, dan *senet50*. Parameter `include_false`, yang menujukkan apakah *fully-connected layers* diikutkan pada model, diatur bernilai *False* karena model hanya digunakan untuk ekstraksi fitur. Kemudian `input_shape` diatur bernilai (224, 224, 3) agar input memiliki ukuran 224 × 224 dengan kanal warna 3 dan parameter pooling diatur bernilai avg untuk menggunakan *global average pooling* pada output *convolutional layer* terakhir. Implementasi ekstraksi fitur dengan *VGG16, Resnet50, dan Senet50* ditunjukkan pada Kode Sumber 4.8, Kode Sumber 4.9, dan Kode Sumber 4.10.

```
1. from keras_vggface.vggface import VGGFace
2.
3. ftr_vgg16 = VGGFace(model='vgg16', include_top=False,
    input_shape=(224, 224, 3), pooling='avg')
4. ftr_vgg16.predict(faces)
```

Kode Sumber 4.8. Ekstraksi fitur dengan *VGG16*.

```
1. from keras_vggface.vggface import VGGFace
2.
3. ftr_resnet50 = VGGFace(model='resnet50', include_top=
   False, input_shape=(224, 224, 3), pooling='avg')
4. ftr_resnet50.predict(faces)
```

Kode Sumber 4.9. Ekstraksi fitur dengan *Resnet50*.

```
1. from keras_vggface.vggface import VGGFace
2.
3. ftr_senet50 = VGGFace(model='senet50', include_top=Fa
   lse, input_shape=(224, 224, 3), pooling='avg')
4. ftr_senet50.predict(faces)
```

Kode Sumber 4.10. Ekstraksi fitur dengan *Senet50*.

### 4.3.2 Pemisahan Dataset

Pemisahan dataset untuk *training* dan *testing* pada klasifikasi dilakukan dengan menggunakan metode *k-fold cross validation* dan dibantu dengan fungsi `StratifiedKfold()` dari *sklearn*. Parameter n_splits bernilai 10 menunjukkan bahwa dataset dibagi menjadi 10 bagian dengan pembagian data *testing* sebesar 1/10 bagian dan data *training* sebesar 9/10 bagian, kemudian dilakukan perulangan sebanyak 10 kali. Implementasi metode ini dapat dilihat pada Kode Sumber 4.11.

```
1.  from sklearn.model_selection import StratifiedKFold
2.
3.  skf = StratifiedKFold(n_splits=10)
```

Kode Sumber 4.11. Pemisahan dataset dengan *Cross Validation*.

### 4.3.3 Klasifikasi *Naïve Bayes*

Pada tugas akhir ini, metode klasifikasi *Naïve Bayes* diimplementasikan menggunakan fungsi `GaussianNB()` dari pustaka *sklearn*. *Gaussian Naïve Bayes* dipilih karena dataset fitur yang akan dilatih dan diprediksi memiliki distribusi normal atau distribusi *gaus*.

```
1.  from sklearn.naive_bayes import GaussianNB
2.
3.  gnbclassifier = GaussianNB()
```

Kode Sumber 4.12. Implementasi metode *Naive Bayes*.

### 4.3.4 Klasifikasi *Support Vector Machine*

Klasifikasi *Support Vector Machine (SVM)* pada tugas akhir ini diimplementasikan menggunakan fungsi `SVC()` dari pustaka *sklearn*. Pada metode ini digunakan kernel *Radial Basis Function (RBF)*. Implementasi metode ini ditunjukkan pada Kode Sumber 4.13.

```
1.  from sklearn.svm import SVC
2.
3.  svclassifier = SVC(kernel='rbf')
```

Kode Sumber 4.13. Implementasi metode *SVM*.

### 4.3.5 Klasifikasi *Random Forest*

Seperti metode *Naïve Bayes* dan *SVM*, pada tugas akhir ini implementasi dari metode *Random Forest* dibantu dengan fungsi `RandomForestClassifier()` dari pustaka *sklearn*. Implementasi metode ini dapat dilihat pada Kode Sumber 4.14.

```
1.  from sklearn.ensemble import RandomForestClassifier
2.
3.  rfclassifier = RandomForestClassifier()
```

Kode Sumber 4.14. Implementasi metode *Random Forest*.

### 4.3.6 Proses *Training* dan *Testing*

Proses *training* dan *testing* pada masing-masing metode klasifikasi *Naïve Bayes, SVM,* dan *Random Forest* dilakukan dengan metode *K-fold Cross Validation* dan dibantu menggunakan fungsi `cross_validate()` yang ada pada pustaka *sklearn*. Fungsi `cross_validate()` akan melakukan *training* dan evaluasi pada *classifier* berdasarkan metode *cross validation* dan kemudian mengembalikan beberapa metriks hasil evaluasi. Metriks yang dikembalikan berdasarkan parameter `scoring` yang ditentukan pada fungsi `cross_validate()`. Pada tugas akhir ini, metriks yang disertakan sebagai parameter `scoring` antara lain akurasi, presisi, recall, dan *f-1 score*. Parameter lain yang ditentukan pada fungsi `cross_validate()` adalah `cv`, yang menunjukkan strategi pembagian *cross validation* yang digunakan. Pada tugas akhir ini digunakan strategi *k-fold cross validation* seperti yang ditunjukkan pada Kode Sumber 4.11. Implementasi proses ini ditunjukkan pada Kode Sumber 4.15.

```
1.  from sklearn.model_selection import cross_validate
2.
3.  scoring = ['accuracy', 'precision_weighted', 'recall_
    weighted', 'f1_weighted']
4.  scores = cross_validate(classifier, features, labels,
     cv=skf, scoring=scoring)
5.
6.  print('fit time: %.04f' % scores['fit_time'].mean())

7.  print('score time: %.04f' % scores['score_time'].mean
    ())
8.  print('accuracy: %.04f' % scores['test_accuracy'].mea
    n())
9.  print('precision: %.04f' % scores['test_precision_wei
    ghted'].mean())
10. print('recall: %.04f' % scores['test_recall_weighted'
    ].mean())
11. print('f1-
    score: %.04f' % scores['test_f1_weighted'].mean())
```

Kode Sumber 4.15. Proses *training* dan *testing* menggunakan metode *cross validation*.

## 4.4 Implementasi Pengolahan Hasil Pengenalan

Pada subbab ini akan dijelaskan mengenai implementasi proses pengolahan hasil pengenalan berdasarkan perancangan yang dijelaskan pada subbab 3.3.4.

### 4.4.1 Penyimpanan Data

Sebelum dilakukan implementasi proses penyimpanan data, terlebih dahulu dilakukan pembuatan tabel-tabel yang dibutuhkan pada database. Seperti yang dijelakan pada subbab 3.3.4.1, akan dibuat tiga tabel: *files, frames,* dan *results*. Implementasi pembuatan ketiga tabel tersebut dapat dilihat pada Kode Sumber 4.16, Kode Sumber 4.17, dan Kode Sumber 4.18.

```
1.  CREATE TABLE `files` (
2.    `id` int(11) NOT NULL AUTO_INCREMENT,
3.    `filename` varchar(255) NOT NULL,
4.    `started_at` timestamp NOT NULL DEFAULT CURRENT_TIME
    STAMP,
5.    `finished_at` datetime DEFAULT NULL,
6.    PRIMARY KEY (`id`)
7.  ) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=utf8
    ;
```

Kode Sumber 4.16. Implementasi pembuatan tabel *files*.

```
1.  CREATE TABLE `frames` (
2.    `filename` varchar(255) NOT NULL,
3.    `frame_number` int(11) NOT NULL,
4.    `path` varchar(255) NOT NULL,
5.    PRIMARY KEY (`filename`,`frame_number`)
6.  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Kode Sumber 4.17. Implementasi pembuatan tabel *frames*.

```
1.  CREATE TABLE `results` (
2.    `id` int(11) NOT NULL AUTO_INCREMENT,
3.    `filename` varchar(255) NOT NULL,
4.    `frame_number` int(11) NOT NULL,
5.    `left_side` int(11) NOT NULL,
6.    `top_side` int(11) NOT NULL,
7.    `right_side` int(11) NOT NULL,
8.    `bottom_side` int(11) NOT NULL,
9.    `label` varchar(50) NOT NULL DEFAULT '',
10.   `confidence` double NOT NULL,
11.   PRIMARY KEY (`id`)
12. ) ENGINE=InnoDB AUTO_INCREMENT=2887 DEFAULT CHARSET=ut
    f8;
```

Kode Sumber 4.18. Implementasi pembuatan tabel *results*.

Setelah dibuat semua tabel yang dibutuhkan, kemudian dibuat implementasi proses penyimpanan data ke masing-masing tabel

yang sudah dibuat. Proses penyimpanan data diimplementasikan menggunakan bahasa *python* dan dibantu menggunakan *library flask-mysql*. Implementasi penyimpanan data ke tabel *files* dapat dilihat pada Kode Sumber 4.19, implementasi penyimpanan data ke tabel *frames* ditunjukkan pada Kode Sumber 4.20, dan implementasi penyimpanan data ke tabel *results* ditunjukkan pada Kode Sumber 4.21.

```
1.  def save_file(self, filename):
2.      conn = self.mysql.connect()
3.      cursor = conn.cursor()
4.      sql = "insert into files (filename) values (%s)"
5.      params = (filename)
6.
7.      cursor.execute(sql, params)
8.      conn.commit()
9.      cursor.close()
```

Kode Sumber 4.19. Implementasi proses penyimpanan data ke tabel *files*.

```
1.  def save_frame(self, filename, frame, path):
2.      sql = "insert into frames (filename, frame_number,
    path) values (%s, %s, %s)"
3.      params = (filename, frame, path)
4.
5.      conn = self.mysql.connect()
6.      cursor = conn.cursor()
7.      cursor.execute(sql, params)
8.      conn.commit()
9.      cursor.close()
```

Kode Sumber 4.20. Implementasi proses penyimpanan data ke tabel *frames*.

```
1.  def save_result(self, filename, frame, box, label, con
    fidence):
2.      conn = self.mysql.connect()
3.      cursor = conn.cursor()
4.      sql = "insert into results \
5.          (filename, frame_number, left_side, top_side,
    right_side, bottom_side, label, confidence) \
6.          values (%s, %s, %s, %s, %s, %s, %s, %s)"
7.
8.      left, top, right, bottom = box
9.      params = (filename, frame, left, top, right, botto
    m, label, confidence)
10.
11.     cursor.execute(sql, params)
12.     conn.commit()
13.     cursor.close()
```

Kode Sumber 4.21. Implementasi proses penyimpanan data ke tabel *results*.

### 4.4.2 Penyusunan Rekapitulasi

Seperti yang telah dijelaskan pada subbab 3.3.4.2, pada tahap penyusunan rekapitulasi dilakukan perhitungan untuk mendapatkan persentase kehadiran dari masing-masing label. Perhitungan dilakukan menggunaakn *query SQL*. Implementasi dari perhitungan persentase kehadiran dapat dilihat pada Kode Sumber 4.22.

Selain dilakukan perhitungan persentase kehadiran, juga ditampilkan lama waktu pemrosesan video dan hasil pengenalan pada masing-masing *frame*. Untuk menampilkan keduanya, dilakukan *query SQL* ke *database*. Implementasinya dapat dilihat pada Kode Sumber 4.23 dan Kode Sumber 4.24.

```
1.  def get_percentage_per_label(self, filename):
2.      sql = "select label, count(*) * 100 / (select coun
    t(*) from frames where filename=%s)\
3.          from results where filename=%s group by label"

4.      params = (filename, filename)
5.
6.      conn = self.mysql.connect()
7.      cursor = conn.cursor()
8.      cursor.execute(sql, params)
9.      results = cursor.fetchall()
10.     cursor.close()
11.
12.     return results
```

Kode Sumber 4.22. Implementasi perhitungan persentase kehadiran.

```
1.  def get_file(self, filename):
2.      sql = "select * from files where filename = %s"
3.      params = (filename)
4.
5.      conn = self.mysql.connect()
6.      cursor = conn.cursor()
7.      cursor.execute(sql, params)
8.      file = cursor.fetchone()
9.      cursor.close()
10.
11.     return file
12.
13. file = db.get_file(filepath)
14. process_time = file[3] - file[2]
```

Kode Sumber 4.23. Implementasi kode untuk menampilkan lama waktu pemrosesan.

```
def get_results(self, filename, frame):
    sql = "select * from results where filename=%s and
 frame_number = %s"
    params = (filename, frame)

    conn = self.mysql.connect()
    cursor = conn.cursor()
    cursor.execute(sql, params)
    results = cursor.fetchall()
    cursor.close()

    return results
```

Kode Sumber 4.24. Implementasi kode untuk menampilkan hasil pengenalan pada suatu *frame*.

## 4.5 Implementasi Antarmuka

Pada tugas akhir ini, dibuat antarmuka berupa aplikasi web. Pembuatan antarmuka diimplementasikan menggunakan bahasa *phyton* dan dibantu dengan *framework flask*. Secara garis besar, aplikasi web yang dibangun terdiri dari 3 *endpoint* seperti yang ditunjukkan pada Tabel 4.1. Implementasi *endpoint* untuk menampilkan halaman utama dan mengunggah *file* video dapat dilihat pada Kode Sumber 4.25. Sedangkan implementasi *endpoint* untuk menampilkan detail rekapitulasi dari suatu *file* dapat dilihat pada Kode Sumber 4.26.

Tabel 4.1. Daftar *endpoint* pada aplikasi web.

| Endpoint | Keterangan |
|---|---|
| GET / | Menampilkan halaman utama |
| POST / | Mengunggah *file* video |
| GET /<filename> | Menampilkan detail rekapitulasi dari suatu *file* |

```
@app.route('/', methods = ['GET', 'POST'])
def home():
    if request.method == 'POST':
        file = request.files['file']

        if not file:
            return 'upload failed.'

        now = datetime.now()
        filename = now.strftime('%Y%m%d-%H%M%S') + '-' + secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        process(filepath, db)

        return redirect(url_for('home'))
    else:
        files = db.get_files()
        files = [file[1] for file in files]
        files = [file.split('/')[1] for file in files]

        return render_template('index.html', files=files)
```

Kode Sumber 4.25. Implementasi *endpoint* untuk mengunggah *file* dan menampilkan halaman utama.

```
1.  @app.route('/<filename>')
2.  def result(filename):
3.      frame = request.args.get('frame')
4.      filepath = os.path.join(app.config['UPLOAD_FOLDER'
    ], filename)
5.
6.      if not frame:
7.          file = db.get_file(filepath)
8.          frames = db.get_frames(filepath)
9.          diff_time = file[3] - file[2]
10.         total_frame = len(frames)
11.         percentage_per_label = db.get_percentage_per_la
    bel(filepath)
12.         percentage_per_label = [(percent[0], "{:.2f}".f
    ormat(100) if percent[1] > 100 else "{:.2f}".format(p
    ercent[1])) for percent in percentage_per_label]
13.         total_label = len(percentage_per_label)
14.
15.         return render_template('result.html', filename=
    filename, frames=frames,
16.             process_time=diff_time, total_frame=total_fr
    ame, percentage_per_label=percentage_per_label,
17.             total_label=total_label)
18.     else:
19.         frame_path = db.get_frame_path(filepath, frame)
    [0]
20.         results = db.get_results(filepath, frame)
21.         frame_number = frame
22.
23.         return render_template('result_frame.html', fil
    ename=filename, frame_number=frame_number,
24.             frame_path=frame_path, results=results)
```

Kode Sumber 4.26. Implementasi *endpoint* untuk menampilkan hasil rekapitulasi.

# BAB 5
# UJI COBA DAN EVALUASI

Pada bab ini akan dijabarkan hasil uji coba dan evaluasi dari sistem yang telah dibuat. Pengujian dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

## 5.1 Lingkungan Uji Coba

Uji coba sistem pada pengerjaan tugas akhir ini dilakukan pada perangkat keras dan sistem operasi sebagai berikut.

- Processor: 1.4 GHz Quad-Core Intel Core i5
- RAM: 8 GB 2133 MHz LPDDR3
- GPU: Intel Iris Plus Graphics 645 1536 MB
- Sistem Operasi: macOS Catalina 10.15.5

## 5.2 Data Uji Coba

Seperti yang dijelaskan pada subbab **Error! Reference source not found.**, data yang digunakan untuk pengujian tugas akhir ini berasal dari video kegiatan perkuliahan di Departemen Informatika ITS. Data untuk pengujian dibagi menjadi dua, yaitu data untuk pengujian model deteksi wajah dan data untuk pengujian model pengenalan wajah.

Untuk pengujian model deteksi wajah, data yang digunakan berupa gambar yang merupakan potongan *frame* dari data video. Data gambar berjumlah 60 dengan masing-masing gambar terdiri dari 9 sampai 13 wajah. Pada masing-masing gambar dibuat *ground truth* yang menunjukkan posisi area wajah yang sebenarnya. Spesifikasi lengkap data untuk pengujian model deteksi wajah data ditujukkan pada Tabel 5.1.

Data untuk pengujian model pengenalan wajah berupa gambar wajah yang sudah dilakukan proses pemberian label. Gambar wajah berasal dari proses ekstraksi wajah yang dilakukan pada data video seperti yang dijelaskan pada subbab **Error! Reference**

**source not found.**. Adapun spesifikasi lengkap data untuk pengujian model pengenalan wajah ditunjukkan pada Tabel 5.2.

Tabel 5.1. Spesifikasi data uji model deteksi wajah.

| **Keterangan** | **Spesifikasi** |
|---|---|
| Ekstensi | .jpg |
| Resolusi | 1920×1080 |
| Jumlah gambar | 60 |
| Jumlah wajah per gambar | 9 – 13 |
| Kanal warna | 3 (RGB) |

Tabel 5.2. Spesifikasi data uji model pengenalan wajah.

| **Keterangan** | **Spesifikasi** |
|---|---|
| Ekstensi | .jpg |
| Resolusi | bervariasi |
| Jumlah gambar | 3508 |
| Jumlah kelas | 13 |
| Jumlah gambar per kelas | 60 – 454 |
| Kanal warna | 3 (RGB) |

Untuk pengujian keseluruhan sistem, data yang digunakan berupa data video. Karena keterbatasan lingkungan uji coba, data video yang akan digunakan memiliki durasi hanya 1 menit 13 detik. Spesifikasi lengkap dari data uji keseluruhan sistem dapat dilihat pada Tabel 5.3.

Tabel 5.3. Spesifikasi data uji keseluruhan sistem.

| **Keterangan** | **Spesifikasi** |
|---|---|
| Ekstensi | .mp4 |
| Resolusi | 1920×1080 |
| Durasi | 1 menit 13 detik |
| Jumlah *frame* | 1460 |
| Jumlah wajah per *frame* | 9 – 13 |
| Kanal warna | 3 (RGB) |

## 5.3 Skenario Uji Coba

Pada subbab ini akan dijelaskan skenario uji coba yang telah dilakukan. Skenario uji coba dibagi menjadi dua: uji coba pada model deteksi wajah dan uji coba pada model pengenalan wajah. Hasil terbaik dari skenario uji coba pada model deteksi wajah akan digunakan untuk scenario uji coba pada model pengenalan wajah.

### 5.3.1 Skenario Uji Coba pada Model Deteksi Wajah

Pada model deteksi wajah, terdapat empat skenario uji coba yang dilakukan. Keempat skenario dilakukan pada data uji yang ditunjukkan pada Tabel 5.1. Adapun skenario yang dilakukan adalah sebagai berikut.

1. Skenario Pengujian 1: dalam skenario ini dilakukan pengujian berupa nilai *mean average precision (mAP)* dan kecepatan proses pada metode deteksi wajah menggunakan *Multi-task Cascaded Convolutional Networks* (MTCNN).
2. Skenario Pengujian 2: dalam skenario ini dilakukan pengujian berupa nilai *mean average precision (mAP)*

kecepatan proses pada metode deteksi wajah menggunakan *Dlib HOG based*.

3. Skenario Pengujian 3: dalam skenario ini dilakukan pengujian berupa nilai *mean average precision (mAP)* dan kecepatan proses pada metode deteksi wajah menggunakan *Dlib CNN based*.
4. Skenario Pengujian 4: dalam skenario ini dilakukan pengujian berupa nilai *mean average precision (mAP)* kecepatan proses pada metode deteksi wajah menggunakan *YOLO*.

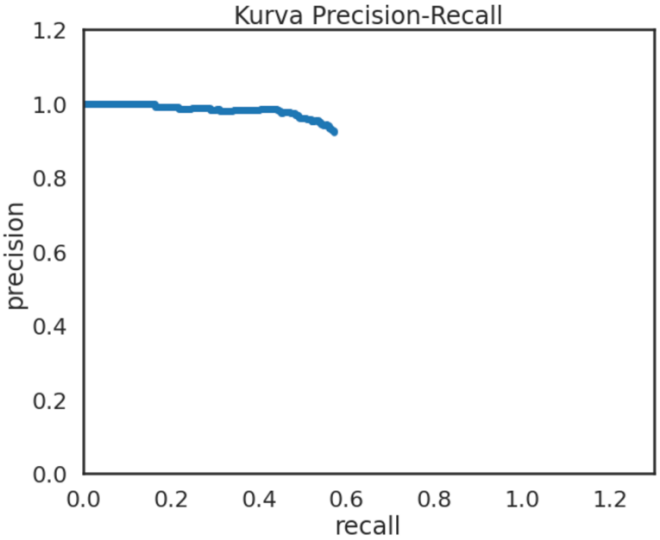
#### 5.3.1.1 Skenario Pengujian 1

Pada skenario ini dilakukan uji deteksi wajah menggunakan metode *Multi-task Cascaded Convoultional Networks (MTCNN)*. Percobaan pengujian dilakukan dengan mendeteksi wajah pada data uji untuk kemudian dibandingkan dengan *ground truth* dan dihitung nilai *mAp* serta kecepatan proses dari metode yang digunakan. Hasil pengujian dari skenario 1 ditunjukkan pada Tabel 5.4.

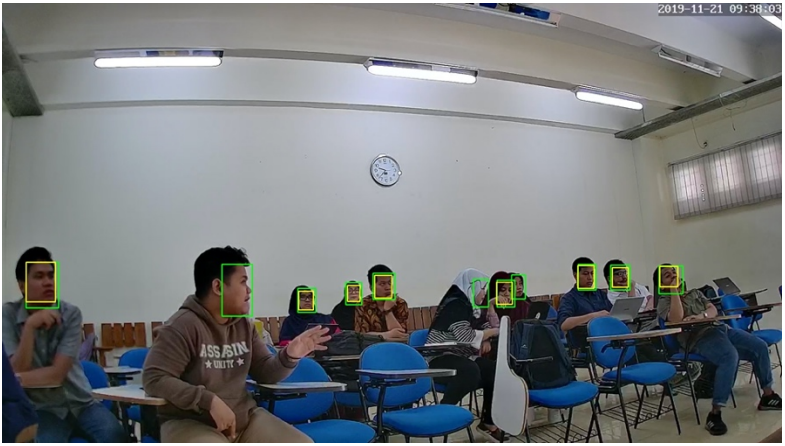
Tabel 5.4. Hasil pengujian model deteksi wajah menggunakan metode MTCNN.

| Ukuran | Nilai |
|---|---|
| mAP | 0.5387 |
| Kecepatan proses | 1.75 – 1.87 fps |

Kurva *precision-recall* dari metode *MTCNN* ditunjukkan pada Gambar 5.1. Semakin besar *AUC (area under curve)* atau luas dibawah kurva, maka semakin bagus metode yang diimplementasikan. Contoh deteksi wajah dari metode *MTCNN* dapat dilihat pada Gambar 5.2.




Gambar 5.1. Kurva *precision-recall* model deteksi wajah menggunakan MTCNN.



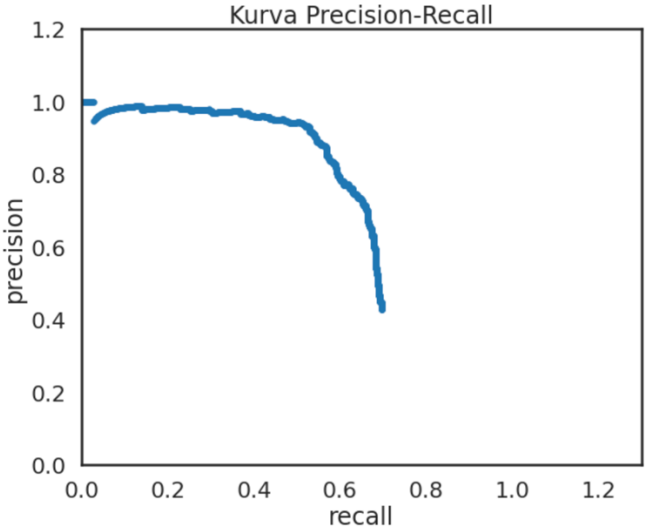

Gambar 5.2. Contoh hasil deteksi wajah menggunakan *MTCNN*.

### 5.3.1.2 Skenario Pengujian 2

Pada skenario pengujian 2 dilakukan uji deteksi wajah menggunakan metode *Dlib HOG based.* Percobaan pengujian dilakukan dengan mendeteksi wajah pada data uji untuk kemudian dibandingkan dengan *ground truth* dan dihitung nilai *mAP* serta kecepatan proses dari metode yang digunakan. Hasil pengujian dari skenario 2 ditunjukkan pada Tabel 5.5.

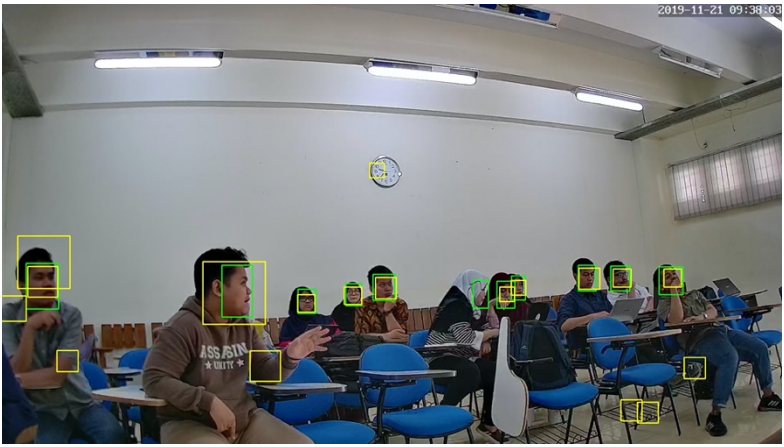
Tabel 5.5. Hasil pengujian model deteksi wajah menggunakan metode *Dlib HOG based*.

| Ukuran | Nilai |
|---|---|
| mAP | 0.6039 |
| Kecepatan proses | 1.09 – 1.11 fps |




Gambar 5.3. Kurva *precision-recall* model deteksi wajah menggunakan *Dlib HOG based.*

Kurva *precision-recall* dari metode *Dlib HOG based* ditunjukkan pada Gambar 5.3. Semakin besar *AUC (area under curve)* atau luas dibawah kurva, maka semakin bagus metode yang diimplementasikan. Contoh deteksi wajah dari metode *Dlib HOG based* dapat dilihat pada Gambar 5.4.



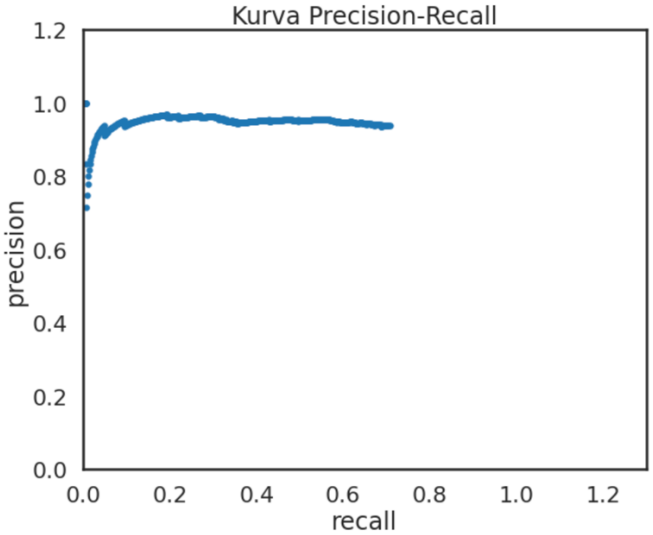

Gambar 5.4. Contoh hasil deteksi wajah menggunakan *Dlib HOG based*.

### 5.3.1.3 Skenario Pengujian 3

Pada skenario pengujian ini dilakukan uji deteksi wajah menggunakan metode *Dlib CNN based*. Percobaan pengujian dilakukan dengan mendeteksi wajah pada data uji untuk kemudian dibandingkan dengan *ground truth* dan dihitung nilai *mAP* serta kecepatan proses dari metode yang digunakan. Hasil pengujian dari skenario 3 ditunjukkan pada Tabel 5.6.

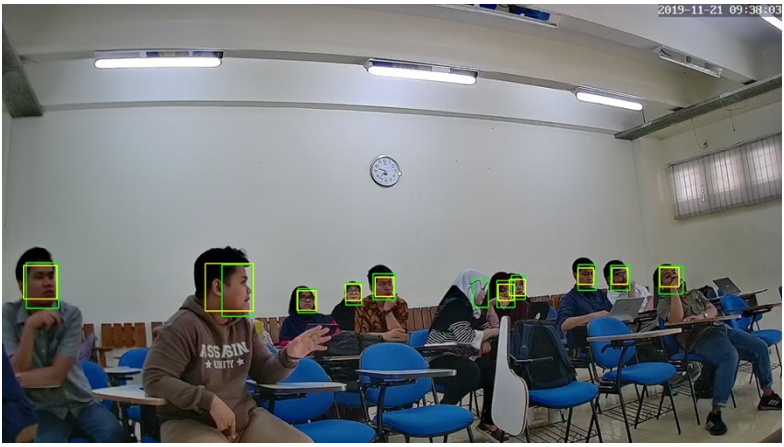
Tabel 5.6. Hasil pengujian model deteksi wajah menggunakan metode *Dlib CNN based.*

| Ukuran | Nilai |
|---|---|
| mAP | 0.7001 |
| Kecepatan proses | 0.035 – 0.042 fps |




Gambar 5.5. Kurva *precision-recall* model deteksi wajah menggunakan *Dlib CNN based.*

Kurva *precision-recall* dari metode *Dlib CNN based* ditunjukkan pada Gambar 5.5. Semakin besar *AUC (area under curve)* atau luas dibawah kurva, maka semakin bagus metode yang diimplementasikan. Contoh deteksi wajah dari metode *Dlib CNN based* dapat dilihat pada Gambar 5.6.




Gambar 5.6. Contoh hasil deteksi wajah menggunakan *Dlib CNN*.

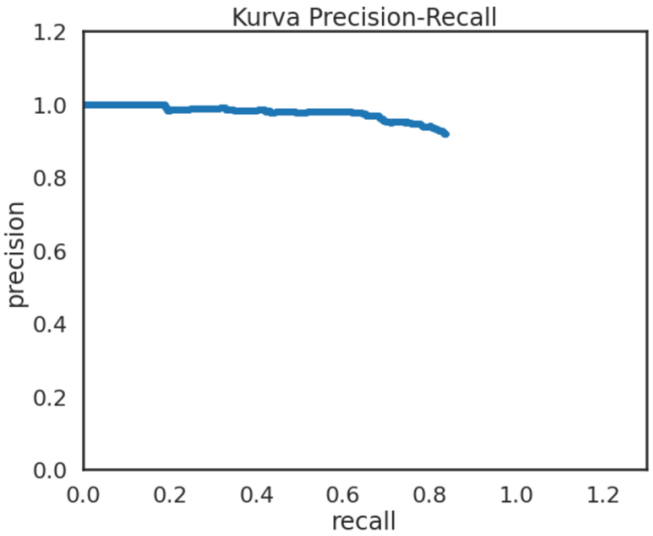
### 5.3.1.4 Skenario Pengujian 4

Pada skenario pengujian 4 dilakukan uji deteksi wajah menggunakan metode *YOLO*. Percobaan pengujian dilakukan dengan mendeteksi wajah pada data uji untuk kemudian dibandingkan dengan *ground truth* dan dihitung nilai *mAP* serta kecepatan proses dari metode yang digunakan. Hasil pengujian dari skenario 4 ditunjukkan pada Tabel 5.7.

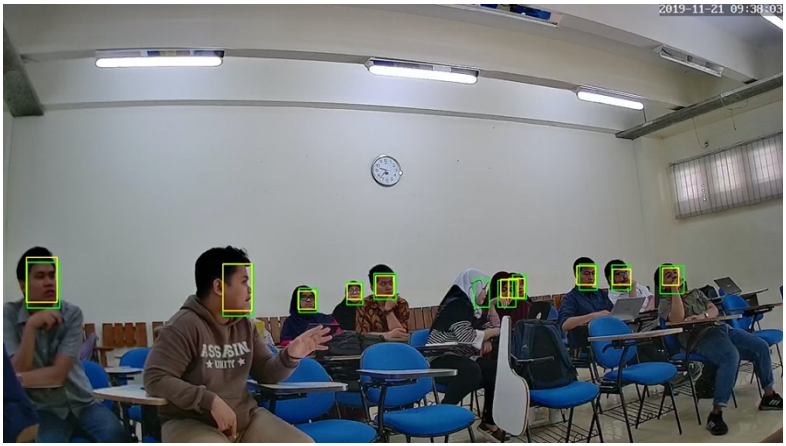
Tabel 5.7. Hasil pengujian model deteksi wajah menggunakan metode *YOLO*.

| Ukuran | Nilai |
|---|---|
| mAP | 0.802 |
| Kecepatan proses | 2.58 – 2.79 fps |

Kurva *precision-recall* dari metode *YOLO* ditunjukkan pada Gambar 5.7. Semakin besar *AUC (area under curve)* atau luas dibawah kurva, maka semakin bagus metode yang diimplementasikan. Contoh deteksi wajah dari metode *YOLO* dapat dilihat pada Gambar 5.8.




Gambar 5.7. Kurva precision-recall model deteksi wajah menggunakan *YOLO*.




Gambar 5.8. Contoh hasil deteksi wajah menggunakan *YOLO*.

### 5.3.2 Skenario Uji Coba pada Model Pengenalan Wajah

Pada model pengenalan wajah, terdapat tiga skenario uji coba yang dilakukan. Ketiga skenario dilakukan pada data uji yang ditunjukkan pada Tabel 5.2. Adapun skenario yang dilakukan adalah sebagai berikut.

1. Skenario Pengujian 1: dalam skenario ini dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* pada model pengenalan wajah dengan ekstraksi fitur menggunakan arsitektur *VGG16* dan klasifikasi menggunakan metode *Naïve Bayes, Support Vector Machine, dan Random Forest.*
2. Skenario Pengujian 2: dalam skenario ini dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* pada model pengenalan wajah dengan ekstraksi fitur menggunakan arsitektur *Resnet50* dan klasifikasi menggunakan metode *Naïve Bayes, Support Vector Machine, dan Random Forest.*
3. Skenario Pengujian 3: dalam skenario ini dilakukan pengujian berupa nilai akurasi, presisi, *recall*, dan *f-measure* pada model pengenalan wajah dengan ekstraksi fitur menggunakan arsitektur *Senet50* dan klasifikasi menggunakan metode *Naïve Bayes, Support Vector Machine, dan Random Forest.*

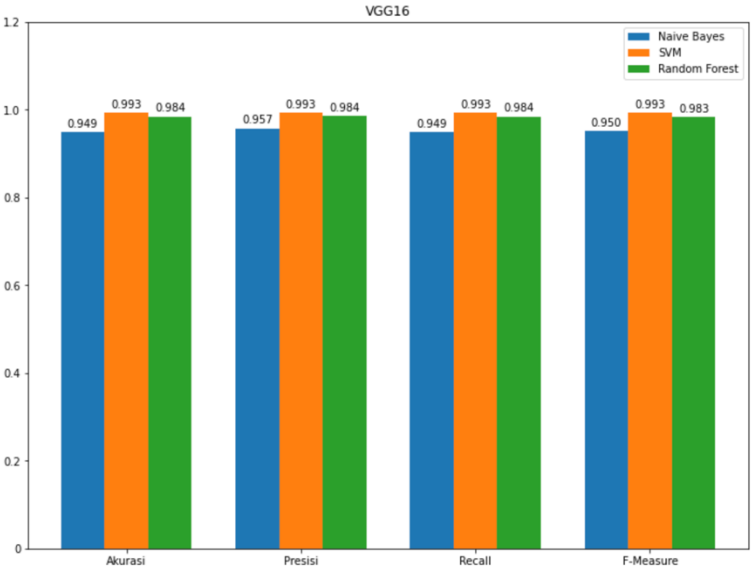
#### 5.3.2.1 Skenario Pengujian 1

Pada skenario ini dilakukan uji pengenalan wajah menggunakan metode ekstraksi fitur berbasis CNN dengan arsitektur *VGG16.* Percobaan pengujian dilakukan dengan memprediksi hasil proses ekstraksi fitur menggunakan metode klasifikasi *Naïve Bayes, Support Vector Machine (SVM),* dan *Random Forest*. Pemisahan dataset untuk pengujian dilakukan menggunakan metode *k-Fold Cross Validation* dengan nilai k = 10. Sedangkan skor presisi, *recall* dan *f-measure* yang ditampilkan pada hasil pengujian adalah skor presisi, *recall*, dan *f-measure*

dengan *average weighted* pada hasil iterasi *cross validation*. Hasil pengujian ditunjukkan pada Tabel 5.8 dan Gambar 5.9.

Tabel 5.8. Hasil pengujian model pengenalan wajah dengan arsitektur *VGG16*.

| Metode | Akurasi | Presisi | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 0.9490 | 0.9570 | 0.9490 | 0.9503 |
| SVM | **0.9928** | **0.9931** | **0.9928** | **0.9927** |
| Random Forest | 0.9837 | 0.9843 | 0.9837 | 0.9831 |




Gambar 5.9. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *VGG16.*

Berdasarkan hasil pengujian, model dengan akurasi tertinggi dengan skor 0.9928 adalah fitur ekstraksi menggunakan arsitektur *VGG16* yang kemudian diklasifikasi menggunakan metode *SVM*.

Sedangkan model dengan akurasi terendah dengan skor 0.949 adalah fitur ekstraksi menggunakan arsitektur *VGG16* yang kemudian diklasifikasi menggunakan metode *Naïve Bayes*.

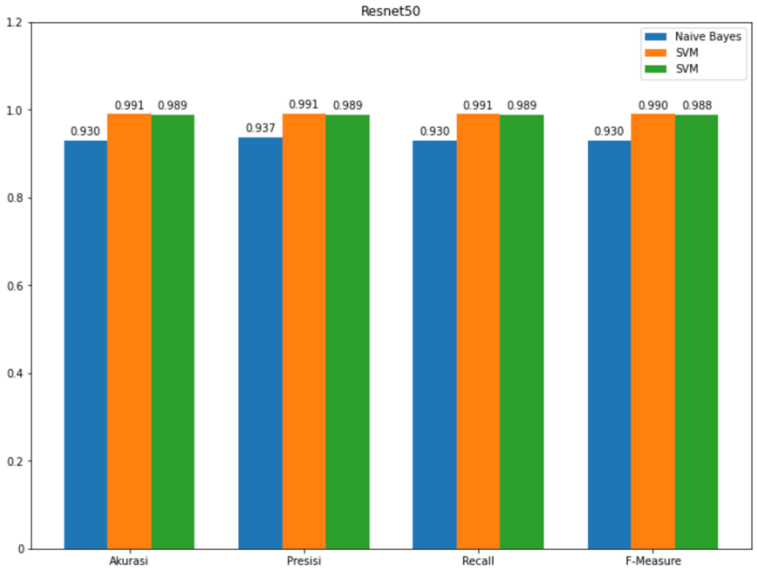
#### 5.3.2.2 Skenario Pengujian 2

Pada skenario ini dilakukan uji pengenalan wajah menggunakan metode ekstraksi fitur berbasis CNN dengan arsitektur *Resnet50.* Percobaan pengujian dilakukan dengan memprediksi hasil proses ekstraksi fitur menggunakan metode klasifikasi *Naïve Bayes, Support Vector Machine (SVM),* dan *Random Forest*. Pemisahan dataset untuk pengujian dilakukan menggunakan metode *k-Fold Cross Validation* dengan nilai k = 10. Sedangkan skor presisi, *recall* dan *f-measure* yang ditampilkan pada hasil pengujian adalah skor presisi, *recall*, dan *f-measure* dengan *average weighted* pada hasil iterasi *cross validation*. Hasil pengujian ditunjukkan pada Tabel 5.9 dan Gambar 5.10.

Tabel 5.9. Hasil pengujian model pengenalan wajah menggunakan arsitektur *Resnet50.*

| Metode | Akurasi | Presisi | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 0.9300 | 0.9371 | 0.9300 | 0.9302 |
| SVM | **0.9905** | **0.9908** | **0.9905** | **0.9904** |
| Random Forest | 0.9888 | 0.9892 | 0.9888 | 0.9885 |

Berdasarkan hasil pengujian, model dengan akurasi tertinggi dengan skor 0.9905 adalah fitur ekstraksi menggunakan arsitektur *Resnet50* yang kemudian diklasifikasi menggunakan metode *SVM*. Sedangkan model dengan akurasi terendah dengan skor 0.9300 adalah fitur ekstraksi menggunakan arsitektur *Resnet50* yang kemudian diklasifikasi menggunakan metode *Naïve Bayes*.



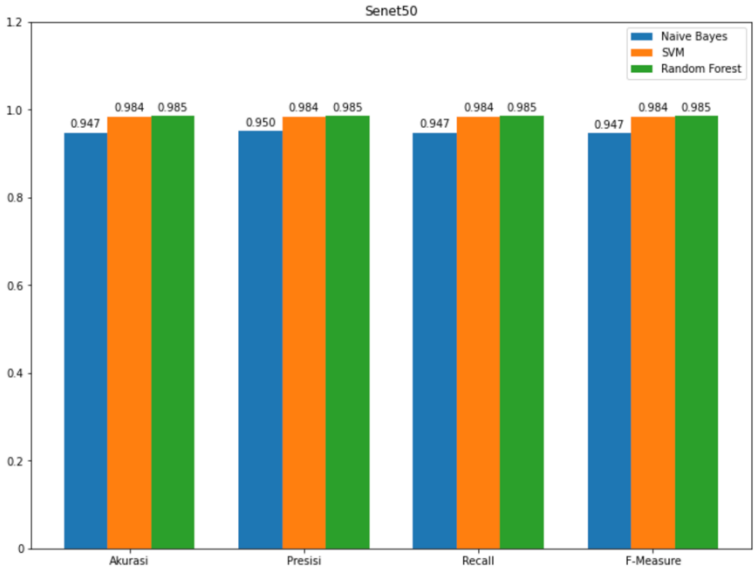

Gambar 5.10. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *Resnet50.*

### 5.3.2.3 Skenario Pengujian 3

Pada skenario ini dilakukan uji pengenalan wajah menggunakan metode ekstraksi fitur berbasis CNN dengan arsitektur *Senet50.* Percobaan pengujian dilakukan dengan memprediksi hasil proses ekstraksi fitur menggunakan metode klasifikasi *Naïve Bayes, Support Vector Machine (SVM),* dan *Random Forest*. Pemisahan dataset untuk pengujian dilakukan menggunakan metode *k-Fold Cross Validation* dengan nilai k = 10. Sedangkan skor presisi, *recall* dan *f-measure* yang ditampilkan pada hasil pengujian adalah skor presisi, *recall*, dan *f-measure* dengan *average weighted* pada hasil iterasi *cross validation*. Hasil pengujian ditunjukkan pada Tabel 5.10 dan Gambar 5.11.

Tabel 5.10. Hasil pengujian model pengenalan wajah menggunakan arsitektur *Senet50.*

| Metode | Akurasi | Presisi | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 0.9470 | 0.9504 | 0.9470 | 0.9466 |
| SVM | 0.9837 | 0.9843 | 0.9837 | 0.9835 |
| Random Forest | **0.9848** | **0.9854** | **0.9848** | **0.9846** |




Gambar 5.11. Grafik hasil pengujian model pengenalan wajah menggunakan arsitektur *Senet50.*

Berdasarkan hasil pengujian, model dengan akurasi tertinggi dengan skor 0.9848 adalah fitur ekstraksi menggunakan arsitektur *Senet50* yang kemudian diklasifikasi menggunakan metode *Random Forest*. Sedangkan model dengan akurasi terendah dengan

skor 0.947 adalah fitur ekstraksi menggunakan arsitektur *Senet50* yang kemudian diklasifikasi menggunakan metode *Naïve Bayes*.

### 5.3.3 Skenario Uji Coba pada Keseluruhan Sistem

Untuk pengujian pada keseluruhan sistem, terdapat dua skenario uji coba yang dilakukan. Kedua skenario dilakukan pada data uji yang ditunjukkan pada Tabel 5.3. Keseluruhan sistem dibangun berdasarkan hasil terbaik dari sekario-skenario pengujian sebelumnya. Adapun skenario yang dilakukan adalah sebagai berikut.

1. Skenario Pengujian 1: dalam skenario ini dilakukan pengujian untuk menghitung waktu yang dibutuhkan untuk memproses satu *frame* pada keseluruhan sistem.
2. Skenario Pengujian 2: dalam skenario ini akan dilakukan pengujian untuk mengukur akurasi dan tingkat keberhasilan pada keseluruhan sistem.

#### 5.3.3.1 Skenario Pengujian 1

Pada skenario ini, dilakukan pengujian untuk menghitung waktu yang dibutuhkan oleh sistem untuk memproses satu *frame* dari data video yang diproses. Hasil dari pengujian ini diperlukan untuk proses pemilihan *frame* seperti yang dijelaskan pada subbab 3.3.1. Hasil pengujian ditunjukkan pada Tabel 5.11.

Tabel 5.11. Hasil pengujian waktu proses pada keseluruhan sistem.

| Iterasi ke | Rata-rata Waktu |
|---|---|
| 1 | 2.03 detik |
| 2 | 2.32 detik |
| 3 | 1.84 detik |
| 4 | 1.83 detik |
| 5 | 1.99 detik |

Dari lima iterasi pengujian yang dijalankan, dihasilkan rata-rata sebesar 2.002 detik. Oleh karena itu, untuk proses pemilihan *frame* pada input video yang memiliki *fps* sebesar 20, dilakukan *skip* pada 40 *frame* sehingga *frame* yang diproses adalah *frame* ke-1, ke-41, ke-81, dan seterusnya.

#### 5.3.3.2 Skenario Pengujian 2

Pada skenario ini, dilakukan pengujian untuk mengukur akurasi dan tingkat keberhasilan pada keseluruhan sistem. Pengujian dilakukan menggunakan antarmuka berupa aplikasi web yang sudah dibangun. Dilakukan 5 iterasi uji coba menggunakan data uji yang ditunjukkan pada Tabel 5.3. Hasil pengujian ditunjukkan pada Tabel 5.12 dan Tabel 5.13.

Tabel 5.12. Hasil uji coba keseluruhan sistem.

| Nama Kolom | Iterasi ke | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** |
| Durasi video | 1:13 | 1:13 | 1:13 | 1:13 | 1:13 |
| Lama proses | 1:15 | 1:26 | 1:08 | 1:08 | 1:11 |
| Penyimpanan data | Sukses | Sukses | Sukses | Sukses | Sukses |
| Penyusunan Rekapitulasi | Sukses | Sukses | Sukses | Sukses | Sukses |

Tabel 5.13. Hasil deteksi per label pada uji coba keseluruhan sistem.

| Label | Jumlah terdeteksi | Jumlah aktual |
|---|---|---|
| Bu Nanik | 3 | 5 |
| Face 1 | 35 | 36 |
| Face 2 | 35 | 37 |

| | | |
|---|---|---|
| Face 3 | 30 | 31 |
| Face 4 | 37 | 37 |
| Face 5 | 36 | 37 |
| Face 6 | 16 | 31 |
| Face 7 | 37 | 37 |
| Face 8 | 23 | 37 |
| Face 9 | 31 | 37 |
| Face 10 | 0 | 34 |
| Face 11 | 1 | 26 |
| Face 12 | 7 | 23 |

Dari Tabel 5.13 didapatkan nilai akurasi yang didapatkan pada uji coba keseluruhan sistem sebesar 71,32%.

## 5.4 Evaluasi

Pada pengujian model deteksi wajah, hasil terbaik didapatkan pada saat melakukan deteksi wajah menggunakan metode *YOLO*. Seperti yang ditunjukkan pada subbab 5.3.1.4, *YOLO* mendapatkan nilai *mean average precision (mAP)* sebesar 80,2% dengan kecepatan proses mencapai 2.79 *frame* tiap detiknya. Dimana metode lain hanya mendapatkan nilai *mAP* sebesar 53,87% untuk metode *MTCNN*, 60,39% untuk metode *Dlib HOG based,* dan 70,01% untuk metode *Dlib CNN based*. Sedangkan untuk kecepatan prosesnya, metode *MTCNN* dapat memproses maksimal 1,87 *frame* per detik, metode *Dlib HOG based* dapat memproses maksimal 1,11 *frame* per detik, dan metode *Dlib CNN based* dapat memproses maksimal 0.042 *frame* per detik.

Untuk pengujian model pengenalan wajah, nilai terbaik untuk akurasi, presisi, *recall*, dan *f-measure* didapatkan pada saat menggunakan ekstraksi fitur dengan arsitektur *VGG16* dan

klasifikasi dengan metode *Support Vector Machine*. Nilai akurasi yang diperoleh sebesar 99,28%. Rincian hasil pengujian ditunjukkan pada subbab 5.3.2.

Dari kedua hasil pengujian diatas, dapat disimpulkan bahwa pada tugas akhir ini sistem pengenalan wajah dengan hasil terbaik didapatkan pada saat dilakukan deteksi wajah menggunakan *YOLO*. Kemudian wajah yang terdeteksi dilakukan proses ekstraksi fitur menggunakan *CNN* dengan arsitektur *VGG16*. Fitur-fitur yang telah didapatkan kemudian dilakukan klasifikasi menggunakan metode *Support Vector Machine (SVM)*.

Kemudian pada uji coba keseluruhan sistem, didapatkan kesimpulan bahwa keseluruhan sistem dapat berjalan dengan baik. Proses penyimpanan data dan penyusunan rekapitulasi berhasil dilakukan. Perbandingan durasi video yang diproses dan lama waktu pemrosesan menunjukkan bahwa sistem juga dapat berjalan secara *real-time*. Dari uji coba tersebut juga didapatkan akurasi sebesar 71,32%.

Berdasarkan contoh hasil uji coba pada Gambar 5.12, terdapat beberapa wajah pada *frame* yang tidak berhasil dideteksi oleh sistem. Hal tersebut dipengaruhi oleh posisi wajah yang menyebabkan tidak semua bagian wajah terlihat pada *frame*, baik karena sedang menoleh, posisinya terlalu jauh dari kamera, atau tertutup sesuatu seperti masker, tangan, dan sebagainya.

Selain kegagalan deteksi pada beberapa wajah, terjadi juga misklasifikasi yang dilakukan sistem. Pada Gambar 5.12, terlihat bahwa terdapat dua wajah yang dikenali sebagai “Face 7” dimana salah satu wajah seharusnya memiliki label “Bu Nanik”. Jika dilihat pada Gambar 5.13 dan Gambar 5.14, dataset dari kedua label tersebut memang terlihat mirip karena sama-sama menggunakan kacamata dan resolusi dari gambar yang cukup rendah.


Gambar 5.12. Contoh hasil uji coba keseluruhan sistem.

Gambar 5.13. Contoh dataset dengan label Face 7.

Gambar 5.14. Contoh dataset dengan label Bu Nanik.

*[Halaman ini sengaja dikosongkan]*

# BAB 6
# KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperolah selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

## 6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Cara mendeteksi area wajah pada data video di tugas akhir ini dilakukan dengan menggunakan metode *MTCNN, Dlib HOG based, Dlib CNN based,* dan *YOLO*.
2. Untuk mengidentifikasi wajah, pada tugas akhir ini dilakukan proses ekstraksi fitur pada wajah yang akan diidentifikasi untuk kemudian dilakukan klasifikasi berdasarkan fitur yang telah didapatkan. Metode ekstraksi fitur diimplementasikan menggunakan *CNN* dengan arsitektur *VGG16, Resnet50,* dan *Senet50*. Sedangkan untuk klasifikasi digunakan metode *Naïve Bayes, Support Vector Machine (SVM)*, dan *Random Forest*.
3. Evaluasi performa dari model deteksi wajah yang telah dibuat dilakukan dengan membandingkan hasil deteksi dengan *ground truth* dan kemudian dihitung nilai *mean average precision (mAP)*. Selain menghitung nilai *mAP*, juga dilakukan pengukuran terhadap kecepatan proses dari model deteksi wajah yang dibuat. Dari empat metode deteksi wajah yang diimplementasikan, *YOLO* mendapatkan nilai *mean average precision (mAP)* terbaik, yaitu 80,2%. Selain mendapatkan *mAP* terbaik, kecepatan proses dari metode *YOLO* juga menjadi yang tercepat dengan mencapai 2.79 *frame* per detik. Sedangkan untuk mengevaluasi performa dari

model pengenalan wajah, dilakukan perhitungan nilai akurasi, presisi, *recall*, dan *f-measure* dari model. Pemisahan dataset yang digunakan untuk *training* dan *testing* pada model pengenalan wajah dilakukan dengan metode *k-Fold Cross Validation* dengan nilai $k$ = 10. Nilai akurasi terbaik untuk model pengenalan wajah didapatkan dengan menggabungkan ekstraksi fitur menggunakan arsitektur *VGG16* dan metode klasifikasi *SVM*. Gabungan keduanya dapat menghasilkan akurasi sebesar 99,28%.

## 6.2 Saran

Berikut adalah beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan pengujian yang telah dilakukan.

1. Mengembangkan model deteksi wajah yang lebih akurat.
2. Menggunakan metode *face tracking* sehingga tidak perlu melakukan pengenalan wajah setiap *frame*.
3. Menerapkan metode *keyframe selection* untuk menguragi *redundant frame* yang diproses.

# DAFTAR PUSTAKA


[1] "Peraturan Akademik ITS," 26 March 2018. [Online]. Available: https://www.its.ac.id/informatika/akademik/peraturan-dan-informasi/peraturan-akademik-its/. [Accessed 23 December 2019].

[2] "Deep learning," Wikipedia, 6 October 2019. [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Accessed 8 January 2020].

[3] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[4] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks," 16 December 2016. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 23 December 2019].

[5] A. Dertat, "Applied Deep Learning," 8 November 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. [Accessed 23 December 2019].

[6] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters,* vol. 23, no. 10, pp. 1499-1503, Oct 2016.

[7] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005.

[8] A. Ponnusamy, "CNN based face detector from dlib," 18 April 2018. [Online]. Available: https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c. [Accessed 28 June 2020].

[9] J. Redmon, . S. Divvala, R. Girshick and A. Farhad, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, IEEE, 2016, pp. 779-788.

[10] O. M. Parkhi, A. Vedaldi and A. Zisserman, "Deep Face Recognition," in *British Machine Vision Conference 2015*, 2015.

[11] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," 2017.

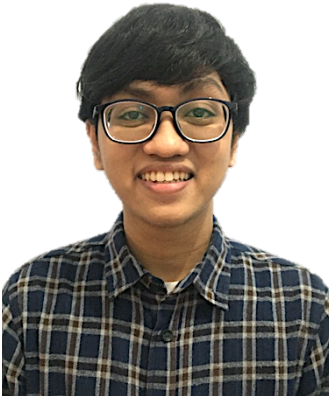
[12] S. Narkhede, "Understanding Confusion Matrix," 9 May 2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed 29 June 2020].

[13] J. Hui, "mAP (mean Average Precision) for Object Detection," 7 March 2018. [Online]. Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. [Accessed 30 June 2020].

[14] A. Dutta, G. Ankush and A. Zisserman, "VGG Image Annotator (VIA)," [Online]. Available: http://www.robots.ox.ac.uk/~vgg/software/via/. [Accessed 30 June 2020].

[15] T. Nguyen, "YOLOFace," [Online]. Available: https://github.com/sthanhng/yoloface. [Accessed 30 June 2020].

# BIODATA PENULIS



Muhajir bin Abd. Latif, lahir di Sidoarjo pada tanggal 28 April 1998. Penulis merupakan mahasiswa Teknik Informatika ITS 2016. Penulis aktif dalam berbagai organisasi mahasiswa, diantaranya menjadi staff Departemen Teknologi Himpunan Mahasiswa Teknik Computer Informatika dan staff *Organization and Social Responsibility* BEM Fakultas Teknologi Informasi. Selain itu penulis juga sempat menjadi panitia di berbagai *event* salah satunya Schematics. Penulis juga berpengalaman sebagai asisten dosen pada matakuliah jaringan computer dan tercatat sebagai administrator di lab. Mobile Innovation Studio. Dalam menyelesaikan pendidikan sarjana, penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV). Penulis dapat dihubungin melalui email muhajir.al28@gmail.com.