Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning

Miftakhurrokhmat

18917214

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer

Konsentrasi Sains Data

Program Studi Informatika Program Magister

Fakultas Teknologi Industri

Universitas Islam Indonesia

2023

Lembar Pengesahan Pembimbing

Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning

Miftakhurrokhmat
18917214

Yogyakarta, Januari, 2023

Pembimbing

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.

Lembar Pengesahan Penguji

Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning

Miftakhurrokhmat

18917214

Yogyakarta, Januari, 2023

Tim Penguji,

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.
Ketua

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.
Anggota I

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.
Anggota II

Mengetahui,
Ketua Program Studi Informatika Program Magister
Universitas Islam Indonesia

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.

Abstrak

Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning

Salah satu yang sering menjadi prasyarat dan tolak ukur dalam menilai mahasiswa di dunia pendidikan adalah kehadiran di pembelajaran pada suatu kelas. Masih sering dijumpai praktik curang mahasiswa dalam presensi. Selain itu juga, administrasi konvesional dalam presensi terutama berbasis kertas berpotensi pemborosan dan membutuhkan waktu yang tidak sebentar dalam proses rekapitulasi secara manual. Penelitian ini bermaksud untuk merancang bangun sistem presensi kelas berbasis pengenalan pola wajah dan tersenyum. Pengenalan wajah (*face recognition*) ini diimplementasikan menjadi suatu *service* yang dipasang pada *mini computer* atau IoT di tiap kelas, dan terhubung dengan IP CCTV dan jaringan kampus. Sistem terintegrasi ini diharapkan bisa menjadi alternatif solusi agar praktik "titip presensi" sulit dilakukan. Hanya mahasiswa yang hadir di kelas yang akan dipresensikan oleh *face recognition* ini secara otomatis. Selain itu juga, sebagai pemberitahuan ke mahasiswa, setiap presensi yang tercatat akan dikirim notifikasi ke mobile app mahasiswa bersangkutan, lalu dilakukan konfirmasi dengan *selfie* tersenyum. Implementasi presensi dengan tersenyum ini diharapkan mampu menjadi pengembangan budaya kampus yang bisa mendukung strategi untuk pendidikan karakter mahasiswa melalui pembelajaran. Metode yang digunakan di penelitian ini berupa *Deep Learning* dengan arsitektur dari FaceNet sebagai *feature extractor* dikombinasikan dengan SVM untuk klasifikasi pengenalan wajah, serta digunakan algoritma *Haar Cascade* untuk mengenali senyuman. Presensi kelas ini sudah terbentuk menjadi suatu *prototype* akan tetapi belum diimplementasikan secara nyata di lapangan mengingat saat penelitian sedang pandemi. Hasil akurasi yang didapat dari *testing* model sebesar 92,9%, dan akurasi hasil *testing live* sebesar 66,7%. Nilai *testing live* lebih kecil dan cukup jauh dari *testing model* menunjukkan hasil *training model* masih terlalu *overfitting* sehingga ke depan masih perlu dikembangkan lagi model pengenalan wajah dengan akurasi yang lebih tinggi.

Kata kunci

convolutional neural network, deep learning, pengenalan wajah, presensi, senyum

Abstract

Class Attendance Based on Face Patterns and Smiling Using Deep Learning

One that is often a prerequisite and benchmark in assessing students in the world of education is attendance in a class. There are still frequent fraudulent practices among students in attendance. In addition, conventional administration in attendance, especially paper-based, has the potential to be wasteful and requires a long time in the manual recapitulation process. This study intends to design a class attendance system based on facial and smile pattern recognition. This face recognition is implemented as a service that is installed on a minicomputer or IoT in each classroom and connected to IP CCTV and the campus network. This integrated system is expected to be an alternative solution because the practice of "entrusting attendance" is difficult. Only students present in class will be automatically recognized by face recognition. Apart from that, as a notification to students, every recorded presence will be sent as a notification to the student's mobile app, then confirmed with a smiling selfie. The implementation of attendance by smiling is expected to become a campus culture development that can support strategies for student character education through learning. The method used in this study is deep learning with the architecture of FaceNet as a feature extractor combined with SVM for facial recognition classification, and the Haar cascade algorithm is used to recognize smiles. This class presence has been formed into a prototype but has not been implemented in practice in the field, considering that during the research there was a pandemic. The accuracy results obtained from model testing are 92.9%, and the accuracy of the live testing results is 66.7%. The value of live testing is smaller and quite far from the testing model, indicating that the results of the training model are still too overfitting, and that, in the future, there will be a need to develop facial recognition models with higher accuracy.

Keywords

convolutional neural network, deep learning, facial recognition, presence, smile

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, Januari, 2023

Miftakhurrokhmat, S.Kom

Daftar Publikasi

Publikasi yang menjadi bagian dari tesis

Miftakhurrokhmat, Rajagede, R. A., & Rahmadi, R. (2021). Presensi Kelas Berbasis Pola Wajah, Senyum dan Wi-Fi Terdekat dengan Deep Learning. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 5(1), 31 - 38. https://doi.org/10.29207/resti.v5i1.2575

Kontributor	**Jenis Kontribusi**
Miftakhurrokhmat	Mendesain eksperimen (70%) Menulis *paper* (80%)
Rian Adam Rajagede	Menulis dan mengedit paper (20%) Review paper (20%) Memberikan saran teknis terkait *deep learning*
Dhomas Hatta Fudholi	Memberikan saran dan improvisasi terhadap penelitian
Ridho Rahmadi	Mendesain eksperimen (30%) Review paper (80%)

Halaman Kontribusi

Dosen Pembimbing, dan Dosen-dosen Penguji memberikan kontribusi berupa masukan perbaikan dalam penulisan Tesis dan memberikan masukan terkait hasil penelitian yang telah dilakukan.

Halaman Persembahan

Bismillahirrohmanirrohim

Saya persembahkan Tesis ini dengan segenap cinta dan kebanggaan teruntuk keluarga saya:

Istri saya Unun Tri Utari, kedua anak saya Raka Ahmad Habibi dan Nadim Ahmad Alfarizqi, dan ketua orang tua saya Bapak Abunawas dan Ibu Mahmudah
yang tanpa lelah selalu mendoakan saya dengan tulus,
selalu percaya terhadap apa yang saya upayakan,
dan tanpa henti memotivasi saya saat mulai jenuh.
Terimakasih semua.

Kata Pengantar

Bismillahirrohmanirrohim

Alhamdulillahirabbil'alamin, senantiasa penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga Tesis dengan judul "Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning" dapat diselesaikan dengan baik. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Informatika Program Magister Fakultas Teknologi Industri Universitas Islam Indonesia. Dalam penyelesaian Tesis ini, penulis banyak mendapatkan dukungan dari berbagai pihak, sehingga penulis perlu menyampaikan ucapan terima kasih sebanyak-banyaknya kepada:

1. Rektor Universitas Islam Indonesia saat ini, yth. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D.
2. Ketua Program Studi Informatika Program Magister Universitas Islam Indonesia sewaktu penulis masuk pertama kali, yth. Ibu Izzati Muhimmah, S.T., M.Sc., Ph.D.
3. Ketua Program Studi Informatika Program Magister Universitas Islam Indonesia saat ini, Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D.
4. Dosen Pembimbing yang telah memberikan arahan dan masukan dalam proses Tesis ini, yth. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D dan Bapak Dr. Ing. Ridho Rahmadi, S.Kom., M.Sc.
5. Dosen-Dosen Penguji yang telah memberikan arahan dan masukan dalam hasil Tesis ini, yth. Bapak Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D, dan Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D.
6. Bapak/Ibu Dosen Program Studi Informatika Program Magister yang telah berbagi ilmu yang bermanfaat bagi penulis, semoga dari bekal ilmu yang diberikan dapat menjadi amal jariyah Bapak/Ibu Dosen di dunia dan akhirat.
7. Bapak/Ibu Staff Akademik Program Studi Informatika Program Magister Universitas Islam Indonesia yang telah membantu segala urusan administrasi di kampus.
8. Sahabat Sains Data generasi ke-2 (*Putri, Yuan, Yohani, Aher, Vita, Nurdi, Firah*) yang selalu kompak dan selalu optimis bisa lulus semua (walaupun ternyata tidak bareng).

9. Sahabat Sains Data generasi ke-1 & ke-3 (*Windi, Fahmi, Atin, Rifai, Satya, Yopi, dll*) yang pernah kuliah bareng, dan yang berada di group Whatsapp "DataSains UII ++" (beberapa angkatan setelahnya), semoga silaturahmi tetap terjaga, saling berbagi informasi dan tegur sapa.
10. Sahabat Sharing Thesis (*Mas Pailus, Mas Dede, Pak Tofa, Yurio, Malik, Eko*) yang selalu berbagi cerita dan optimis insya Allah pasti lulus.
11. Sahabat Program Studi Informatika Program Magister Universitas Islam Indonesia Yogyakarta seangkatan dan beragam konsentrasi (DS, FD, IM, SIE).
12. Teman-teman riset di kampus lain : **Universitas Amikom Yogyakarta** (*Mba Achi, Mba Sal, Dito, Andhi, Ilyas, Hairul*), **STMIK AMIKOM Surakarta** (*Mba Ina, Adit, Pak Wawan, Bu Mita, Mba Tinuk, Pak Febri, Pak Rian, Indrawan*), **INSTIPER** (*Pak Erick, Pak Teddy, Mas Wawan*), **UNIMMA** (*Mas Resa*), dan **UNNES** (*Mas Jumanto*).
13. Teman-teman di industri baik di **Unisoft** (*Mas Ali, Mas Shaddiq, Mas Dimas*), **Indogriya** (*Pak Andi, Pak Teddy, Yuswan, Mba Rya, Mba Novi, Mas Hendhy, Rizki*), **Sobatechno** (*Ulul, Galih, Artha, Jamal*), **Akar Makna** (*Fuad, Mba Betty, Indra, Endo, As'ad Afaan gan, Satria, Ibnu*), **Inixindo** (*Mas Citra, Faizal, Mas Hani, Pak Topa, Pak Umar*), **Sinau Jogja** (*Mba Ivana & team, Mas Dimas Indojustice, Zidni GreatDay*), **Dolkode** (*Mba Dew, Lord Kemal*), **Dinamika Mediakom** (*Dika, Aldo, Pak Dedi, Pak Ahmad*), teman-teman freelancer (*Fajar Doni, Pak Yazid*), dan yang tidak bisa saya sebutkan satu persatu.

Penulis menyadari bahwa di penulisan Tesis ini kemungkinan masih banyak kekurangan, untuk itu kritik dan saran yang sifatnya membangun sangat penulis harapkan demi penelitian yang lebih baik.

Daftar Isi

Daftar Tabel

Daftar Gambar

Glosarium

API	- Application Programming Interface
CCTV	- Closed-Circuit Television
CNN	- Convolutional Neural Network
DL	- Deep Learning
FCN	- Fully Connected Network
FR	- Face Recognition
IoT	- Internet of Thing
IP	- Internet Protocol
JSON	- JavaScript Object Notation
KBM	- Kegiatan Belajar Mengajar
KTM	- Kartu Tanda Mahasiswa
ML	- Machine Learning
MTCNN	- Multi-task Cascaded Convolutional Neural Network
NIK	- Nomor Induk Karyawan
NIM	- Nomor Induk Mahasiswa
NIST	- National Institure of Standards and Technology
NN	- Neural Network
OpenCV	- Open Source Computer Vision Library
REST	- Representational State Transfer
SDK	- Software Development Tool
SSID	- Service Set Identifier
SVM	- Support Vector Machine
URL	- Uniform Resource Locator
XML	- Extensible Markup Language

BAB 1
Pendahuluan

1.1 Latar Belakang

Hingga saat ini, pencatatan presensi telah diimplementasikan dengan berbagai cara di bidang pendidikan, manajemen dan bidang lainnya. Ada yang menggunakan teknologi terkini seperti pengenalan sidik jari (Fakih et al., 2015), namun tidak sedikit institusi yang masih melakukan pencatatan manual seperti membubuhkan tanda tangan pada kertas atau buku yang lalu diagregasi menjadi laporan presensi definitif. Proses pencatatan partisipasi manual mempunyai beberapa kelemahan seperti boros kertas, rekapitulasi memakan waktu, integrasi dengan sistem lain sulit, dan rentan dari pemalsuan. Dalam sistem presensi dengan sidik jari, seringkali dibangun di atas platform hard-coded (tidak memungkinkan komunikasi dengan perangkat atau sistem lain), membuat integrasi dengan sistem utama fasilitas menjadi sulit. Bagian sistem ini juga rentan untuk dilakukan peretasan atau pemalsuan sidik jari (Matsumoto & Matsumoto, 2002; Samirso, n.d.) . Opsi lain, seperti presensi dengan kartu, memiliki masalah yang mirip: dibangun di sebuah platform yang tidak mudah diintegrasikan dengan sistem utama seperti sistem akademik. Belum lagi penggunaan kartu sangat rawan pemalsuan, dikarenakan kartu dapat dengan mudah dipindahkan. Lebih jauh lagi, dibutuhkan biaya yang relatif tidak sedikit dalam mencetak kartu. Oleh sebab itu, dibutuhkan suatu solusi untuk presensi yang menjawab isu-isu permasalahan diatas.

1.2 Rumusan Masalah

Bagaimana merancang purwarupa “Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning” ?

1.3 Tujuan Penelitian

1. Merancang purwarupa “Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning” sebagai alternatif solusi presensi kelas yang dapat mengurangi potensi kecurangan mahasiswa, mengurangi penggunaan kertas, dan memudahkan rekapitulasi.
2. Bagaimana mengimplementasikan *Deep Learning* pada proses pengenalan wajah mahasiswa, dan bagaimana mengimplementasikan model pengenalan senyuman.

1.4 Batasan Masalah

1. Menggunakan analisis kebutuhan layaknya mengembangkan sistem informasi dengan batasan pengembangan aplikasi web maupun android tidak disertakan.
2. Menggunakan *dataset* sampel mahasiswa dengan jumlah 7 orang dengan masing-masing 100 foto atau citra.
3. Menggunakan satu arsitektur *deep learning* untuk implementasi pengenalan wajah.
4. Menggunakan satu algoritma *machine learning* untuk implementasi pengenalan senyuman.
5. Sebagian besar *script* yang digunakan menggunakan Python dengan spesifikasi tertentu baik yang dikembangkan sendiri maupun yang diadopsi.
6. Tingkat keakurasian model belum menjadi prioritas utama karena untuk tahap awal ini lebih diprioritaskan untuk purwarupa dan konsep integrasi.
7. Kesalahan dan kegagalan dalam mendeteksi wajah dan pengenalan wajah dalam sistem belum menjadi cakupan penelitian saat ini.
8. Kesalahan dan kegagalan dalam mendeteksi senyuman dalam sistem belum menjadi cakupan penelitian saat ini.
9. Penanganan masalah apabila data presensi gagal dicatat dalam sistem belum menjadi cakupan penelitian saat ini.
10. Dalam purwarupa ini diperlukan peran mahasiswa menggunakan smartphone miliknya untuk konfirmasi kehadiran dengan senyuman.
11. Beberapa permasalahan yang muncul saat sistem presensi ini diterapkan secara real di lapangan belum dapat didefinisikan, dikarenakan implementasi purwarupa ini baru diujicoba melalui simulasi. Permasalahan-permasalahan yang ditemukan melalui simulasi menjadi dasar perbaikan ke depan.

1.5 Manfaat Penelitian

1. Bagi lembaga pendidikan, memberikan gambaran atau alternatif solusi sistem presensi kelas dengan menggunakan teknologi *deep learning* yang bertujuan untuk menghindari kecurangan
2. Bagi para peneliti, konsep yang dirancang dapat dikembangkan dan dapat menjadi referensi jika akan melakukan penelitian serupa dengan improvisasi di bagian saran dan perbaikan atau mengembangkan hal lain.

1.6 Sistematika Penulisan

Sistematika penulisan dalam tesis ini, disusun sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi gambaran penelitian terdahulu, dan konsep pengetahuan tentang *face processing*, *face recognition*, *ML*, *classification*, *DL*, CNN, MTCNN, inception, inception-resnet-v1, facenet, *face embeddings*, SVM, openCV, dan Dlib.

BAB III METODOLOGI

Bab ini berisi tentang metodologi penelitian yang terdiri dari analisis kebutuhan dan perancangan sistem presensi, pemilihan arsitektur *deep learning*, pengumpulan data citra mahasiswa, *preprocessing* data citra mahasiswa, *training dataset* citra mahasiswa, evaluasi model pengenalan pola wajah, implementasi model pengenalan wajah & model senyuman, dan simulasi & evaluasi sistem

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil dan pembahasan penelitian yang terdiri dari pengumpulan data citra mahasiswa, *training dataset* mahasiswa, *script* pengujian model, *script* implemetnasi *service* saver, *script* implementasi *service face recognition*, *script* REST API *smile detection*, pengujian model pengenalan wajah, pengujian model deteksi senyuman, proses presensi, simulasi & evaluasi sistem.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari hasil penelitian dan saran perbaikan untuk kelanjutan penelitian di masa yang akan datang.

BAB 2
Tinjauan Pustaka

2.1 Penelitian Terdahulu

Berdasarkan hasil penelitian dari Dianradika Prasti (Prastika, 2014), sistem presensi menggunakan *barcode* masih memiliki kelemahan jika dosen tidak mengawasi jalannya presensi dengan *barcode*, masih dimungkinkan mahasiswa melakukan titip presensi. Sistem presensi dengan menggunakan finger print sudah menjawab permasalahan presensi konvensional seperti titip presensi, namun masih memiliki kelemahan untuk dalam hal waktu, karena antrian untuk melakukan presensi (Fakih, Raharjana, & Zaman, 2015).

Pada beberapa penelitian sebelumnya (Fachmi et al., 2019; Suhery & Ruslianto, 2017), diberikan solusi sistem presensi berbasis mobile dengan pola pengenalan wajah. Alternatif ini cukup efektif diintegrasikan ke sistem utama di institusi, selain menghemat biaya penyediaan alat presensi. Akan tetapi, ada beberapa faktor lain ikut berpengaruh dalam tingkat akurasi pengenalan wajah (Derisma, 2016). Sehingga perlu dilakukan kelanjutan pengembangan untuk aplikasi pengenalan wajah berbasis mobile ini.

Penelitian ini bermaksud untuk merancang bangun purwarupa sistem presensi berbasis *deep learning* menggunakan *mini computer* dan IP CCTV. Terdiri dari 3 komponen yaitu *backend process* dengan menggunakan *mini computer* dan IP CCTV; sebuah aplikasi mobile yang digunakan untuk notifikasi dan validasi kehadiran dengan pola wajah yang tersenyum; sebuah aplikasi web yang digunakan dosen untuk menyatakan mulai dan selesai untuk perkuliahan sekaligus laporan rekapitulasi kehadiran mahasiswa.

Tabel 2.1 Rangkuman Tinjauan Pustaka

No.	**Nama**	**Tujuan**	**Pendekatan**	***Platform***
1	(Prasti, 2014)	Membantu aktivitas perkuliahan di kampus	*Barcode* dari KTM	*Desktop*
2	(Fakih, Raharjana, & Zaman, 2015)	Memudahkan otomatisasi pencatatan presensi dibandingkan dengan sistem	*Fingerprint Authentication*	*Desktop*

		konvensional (tanda tangan)		
3	(Suhery & Ruslianto, 2017)	Sistem *monitoring* presensi menggunakan deteksi wajah manusia diintegrasikan dengan basis data	Deteksi wajah menggunakan metode *Haar-Cascade Classifier*, ekstraksi fitur menggunakan metode PCA	Android *mobile* terintegrasi dengan server lokal
4	(Fachmi, Sudarma, & Jasa, 2019)	Mengimplementasikan sistem kehadiran perkuliahan yang terintegrasi dengan matakuliah yang diambil dan jadwal perkuliahan	Deteksi wajah dengan metode *Viola Jones*, proses ekstraksi ciri dengan metode *Sobel*, pengenalan wajah dengan metode KNN	Web dengan *webcam*
5	**Usulan**	Alternatif solusi presensi kelas yang untuk mengurangi potensi kecurangan mahasiswa, mengurangi penggunaan kertas, dan memudahkan rekapitulasi.	Deteksi wajah di CCTV menggunakan *library* DLib, pengenalan wajah menggunakan *deep learning* dengan arsitektur CNN (FaceNet) dan SVM untuk klasifikasi	Sistem integrasi *backend processing* di *mini computer*, aplikasi web untuk dosen, aplikasi *mobile* untuk mahasiswa
	Pertimbangan teknologi	Dlib merupakan *face detection* dengan waktu paling sedikit dalam memproses menurut penelitian (Amirgaliyev et al., 2021). FaceNet adalah *embedding* yang universal untuk *face*		

		recognition dan memungkinkan untuk menggunakan beragam jenis algoritma *clustering* (Schroff et al., 2015). Berdasarkan penelitian perbandingan akurasi CNN dan CNN + SVM menunjukkan hasil bahwa CNN + SVM memiliki tingkat akurasi lebih tinggi (Ahlawat & Choudhary, 2020).

Penelitian yang diusulkan ini menggunakan pendekatan *face recognition* sebagai solusi dari titip presensi yang sering dilakukan, dengan *face recognition* dilakukan di ruangan kelas dan pengambilan gambar suasana kelas dilakukan secara langsung menggunakan IP CCTV, peluang untuk berbuat kecurangan diminimalisir. Beberapa penelitian dengan konsep *face recognition* dengan metode lain tapi dengan tujuan yang sama dilakukan yaitu untuk menjawab permasalah fenomena "titip presensi" yang sering dilakukan (Maslihatin et al., 2020; Santoso & Kristianto, 2020).

Penerapan presensi dengan tersenyum ini secara tidak langsung merupakan bentuk pendidikan karakter melalui strategi pembelajaran. Pendidikan karakter di perguruan tinggi diperlukan untuk membentuk dan membangun mahasiswa menjadi pribadi berkarakter sesuai nilai-nilai luhur ideologi negara Indonesia, dan memperkuat karakter peserta didik pada jenjang pendidikan sebelumnya (Susanti, 2013).

2.2 Konsep Pengetahuan

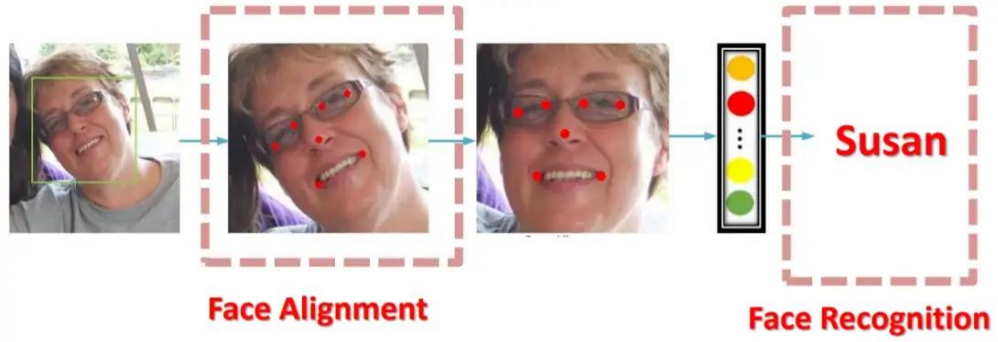
2.2.1 Face Processing

Pengenalan wajah merupakan suatu metode klasifikasi pola, dimana masukannya adalah gambar dan keluarannya adalah label dari gambar tersebut. Pendeteksian wajah (*face detection*) adalah langkah pertama yang sangat penting sebelum pengenalan wajah (*face recognition*). Bidang penelitian yang berkaitan dengan pemrosesan wajah yaitu: pengenalan wajah (*face recognition*), autentikasi wajah (*face authentication*), lokalisasi wajah (*face localization*), penjejakan wajah (*face tracking*), dan pengenalan ekspresi wajah (*facial expression recognition*) (Fachmi et al., 2019).

2.2.2 Face Recognition

Face recognition (FR) adalah teknik yang digunakan untuk verifikasi atau identifikasi identitas seseorang dengan menganalisis dan menghubungkan pola berdasarkan fitur wajah seseorang. Langkah pertama dalam *face recognition* adalah *face detection*, yaitu teknologi yang digunakan

untuk mendeteksi wajah dalam gambar atau video. Dalam suatu proses Deep FR ada *face detector* dan *alignment* yang ditunjukan secara *pipeline* di Gambar 2.1 bawah ini.



Gambar 2.1 Pipeline face recognition. [1]

Dalam *pipeline* FR, tahap pertama detektor wajah (*face detector*) digunakan untuk melokalkan wajah. Selanjutnya di tahap kedua wajah disejajarkan dengan koordinat *canonical* yang dinormalisasi, dan di tahap terakhir modul FR diimplementasikan. Pada modul FR, terdapat langkah yang disebut *face anti-spoofing*, yaitu mengenali apakah wajah tersebut langsung (*live*) atau dipalsukan (*spoofed*). Dalam modul lain yang disebut pemrosesan wajah (*face processing*), yaitu menangani kesulitan pengenalan sebelum pelatihan (*training*) dan pengujian (*testing*) dan saat proses pelatihan (*training*) ada ekstraksi *deep feature* yang diskriminatif, disini digunakan arsitektur dan fungsi *loss* yang berbeda. Dalam kasus metode pencocokan wajah untuk melakukan klasifikasi fitur apabila *deep feature* dari data pengujian diekstraksi.

2.2.3 Machine Learning

Machine Learning (ML) atau dalam Bahasa Indonesia yaitu Pembelajaran Mesin adalah suatu bidang ilmiah yang mempelajari algoritma dan model statistik yang digunakan oleh sistem komputer dengan menggunakan pola. *Machine learning* memberi kemampuan pada sistem untuk secara otomatis belajar dan meningkatkan pengalaman tanpa diprogram secara eksplisit. *Machine learning* fokus pada pengembangan program komputer yang mengakses data dan menggunakan data tersebut untuk belajar mandiri.

Peran pembelajaran mesin (ML) membantu orang di banyak bidang. Bahkan saat ini penerapan ML dapat dengan mudah ditemukan dalam kehidupan sehari-hari. Misalnya, saat menggunakan pengenalan wajah untuk membuka kunci ponsel cerdas atau menjelajahi web

[1] https://miro.medium.com/max/1100/1*WD2A4GIhfS89zJVvqtGgbA.webp

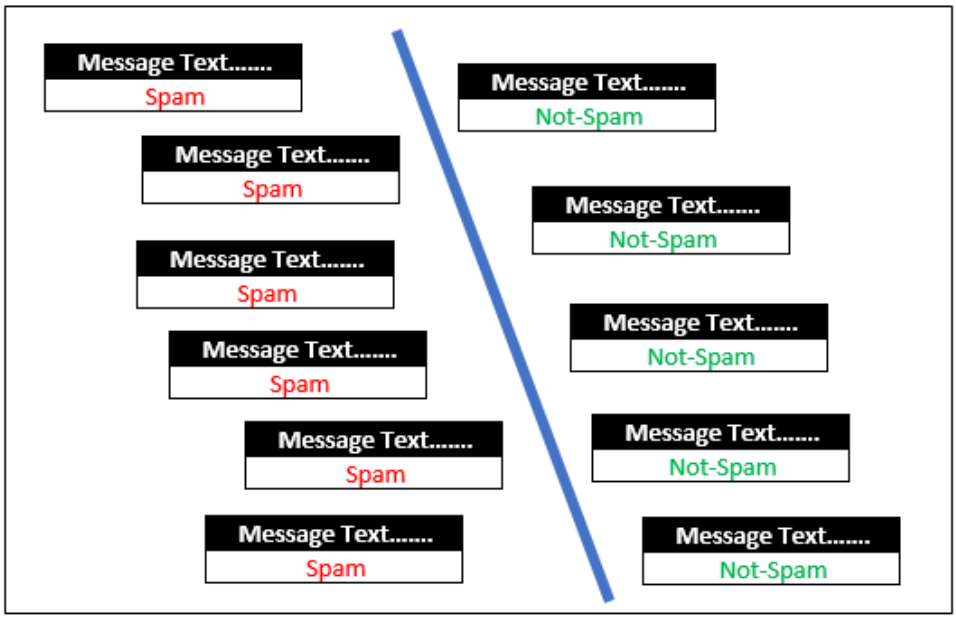
atau media sosial sering kita lihat banyak iklan. Iklan yang terlihat merupakan hasil pengolahan ML, yang menampilkan iklan sesuai kepribadian.

2.2.4 Classification

Salah satu tugas Data Mining maupun Pembelajaran Mesin (ML) adalah klasifikasi (*classification*). Klasifikasi termasuk dalam kategori pembelajaran supervisi (*supervised learning*), yaitu model pembelajaran yang "membutuhkan" bimbingan dari ahli atau pakar. Data *training* untuk klasifikasi sudah memiliki label kelas (*class*) yang dibuat oleh para ahli. Berdasarkan pembelajaran dari data *training* yang telah berlabel sebelumnya, model klasifikasi dapat memprediksi label untuk data baru.

1. **Binary Classification**

   Setiap item data dalam klasifikasi biner (*binary classification*) memiliki atribut kelas yang terdiri dari dua nilai. Nilai kelas bisa positif atau negatif; 0 atau 1; benar atau salah; dll. Contoh dari klasifikasi biner seperti Gambar 2.2 Spam filtering (model klasifikasi yang mengenali pesan dalam kategori *spam* atau *non-spam*). Contoh lainnya seperti analisis kredit (menentukan apakah pelanggan sehat secara finansial), penilaian medis apakah pasien memiliki penyakit atau tidak, dll.Tujuan dari model klasifikasi biner adalah untuk menemukan batasan dimana data dapat dipisahkan secara optimal berdasarkan kategorinya (positif atau negatif).

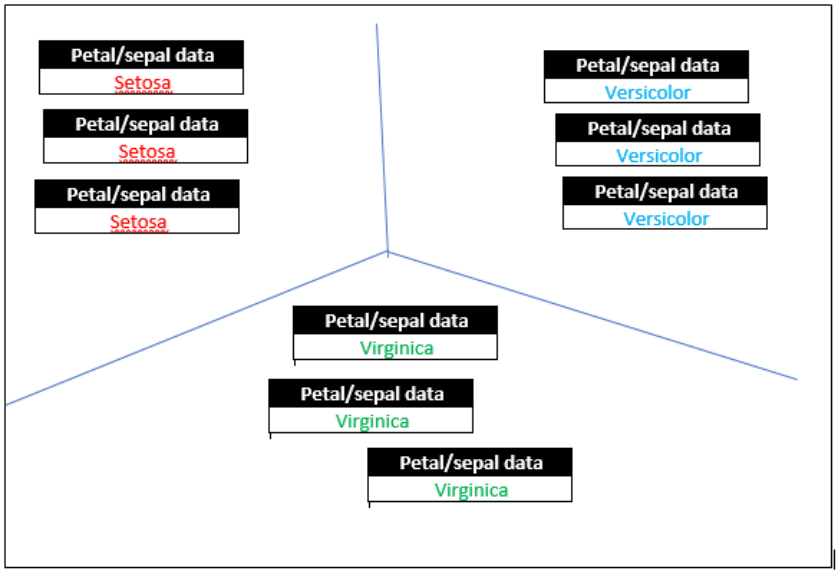


Gambar 2.2 Contoh binary classification. [2]

[2] https://zidny.dosen.ittelkom-pwt.ac.id/wp-content/uploads/sites/26/2019/04/Spam-NotSpam.png

2. Multiclass Classification

Klasifikasi multikelas (*multiclass classification*) memiliki atribut kelas yang terdiri dari beberapa nilai. Himpunan nilai suatu kelas adalah terbatas dan diskrit. Contoh klasifikasi *multi class* adalah pengenalan tipe iris, sebagai contoh Gambar 2.3 disini iris memiliki 3 kelas yaitu: Setosa, Versicolor dan Virginica. Strategi pembelajaran klasifikasi multikelas dapat diimplementasikan dengan menggunakan dua pendekatan, yaitu *One-VS-One Classification* (OVO) dan *One-VS-All Classification* (OVA).



Gambar 2.3 Contoh multiclass classification. [3]

2.2.5 Deep Learning

Definisi *Deep Learning* (DL) dapat diartikan sebagai teknik pembelajaran mesin yang mengarahkan sistem komputer atau mesin untuk bekerja layaknya manusia, yaitu dengan mempelajari situasi dari pembelajaran atau pemrograman khusus.

Deep Learning atau Pembelajaran Mendalam menjadi kunci pengembangan teknologi berbasis *Artificial Intelligence* (AI) atau kecerdasan buatan. Di DL, komputer mempelajari beragam model dan melakukan klasifikasi tugas-tugas berdasarkan data yang dikumpulkan. Data tersebut dapat berupa teks, gambar, atau suara. Dengan jumlah data yang besar, akurasinya dapat ditingkatkan menjadi lebih tinggi.

Lebih khusus lagi, membangun DL adalah tentang meningkatkan kinerja data yang tidak terstruktur (*unstructured data*) menjadi lebih optimal di suatu web atau aplikasi. Selain itu, secara langsung atau tidak langsung mempengaruhi biaya operasional dan pengembangan

[3] https://zidny.dosen.ittelkom-pwt.ac.id/wp-content/uploads/sites/26/2019/04/multiclass.png

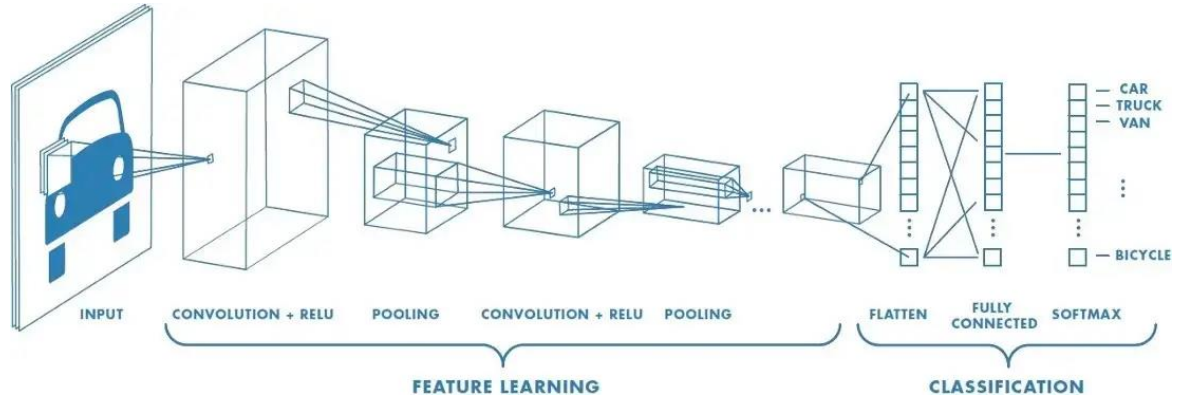
teknologi. Selanjutnya DL dapat membuat teknik desain manipulatif dan fitur menjadi lebih efektif. Sebagian besar DL bekerja dengan menggunakan metode jaringan saraf (*neural network*). Oleh karena itu, mereka juga dikenal sebagai jaringan saraf dalam, yang beroperasi pada berbagai level atau lapisan. Misalnya, jaringan saraf *traditional* hanya memiliki 2-3 lapisan (*layer*) yang tersedia. Pada saat yang sama, jaringan dalam memiliki lebih dari 150 lapisan (*layer*).

Salah satu cara kerja yang banyak diterapkan dari jenis neural networks adalah *Convolutional Neural Networks* (CNN). CNN bekerja dengan melakukan ekstraksi fitur secara langsung dari data gambar. Pemindaian data dan proses analisis menggunakan data gambar menjadi lebih akurat dalam klasifikasi objek.

2.2.6 Convolutional Neural Network

CNN dirancang khusus untuk mengolah data dalam bentuk larik. CNN menerapkan operasi konvolusional yang terinspirasi oleh sistem saraf biologis pada satu atau lebih lapisannya (Hu et al., 2015). Dengan CNN, setiap neuron direpresentasikan dalam dua dimensi, sehingga metode ini cocok untuk diolah dalam bentuk gambar (Maggiori et al., 2016).

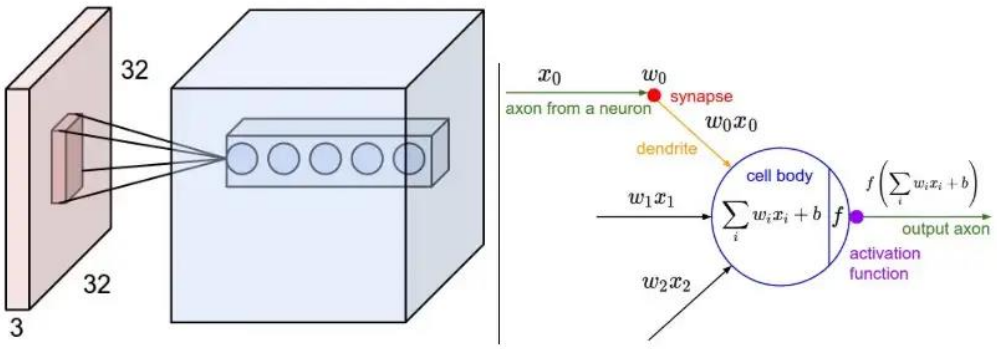
Convolutional Neural Network sangat mirip seperti jaringan Neural pada umumnya, memiliki neuron dengan bobot (weight) dan bias yang dapat dipelajari dan setiap neuron menerima beberapa input, melakukan dot product dan secara opsional mengikutinya dengan non-linearitas. Dalam arsitektur jaringan saraf Deep Convolutional seperti terlihat di Gambar 2.4, seluruh jaringan diekspresikan dari piksel gambar mentah (raw image) di satu ujung ke skor kelas (class) di ujung lainnya. Pada lapisan terakhir, mereka memiliki fungsi loss (mis. Softmax) yang terhubung sepenuhnya (fully connected).



Gambar 2.4 Arsitektur CNN. [4]

[4] https://miro.medium.com/max/1100/1*XbuW8WuRrAY5pC4t-9DZAQ.webp

Pada Gambar 2.5 di bawah ini, setiap neuron di CNN hanya terhubung ke region lokal dalam volume *input* secara spasial, tetapi dengan kedalaman penuh (*full depth*). Ada banyak neuron bersama dengan kedalamannya, semuanya melihat region yang sama di *input*. Di sisi kanan neuron dari bab *Neural Network* tetap tidak berubah: Mereka masih menghitung dot product dari bobotnya dengan *input* yang diikuti oleh non-linearitas, tetapi konektivitasnya sekarang dibatasi untuk lokal secara spasial.

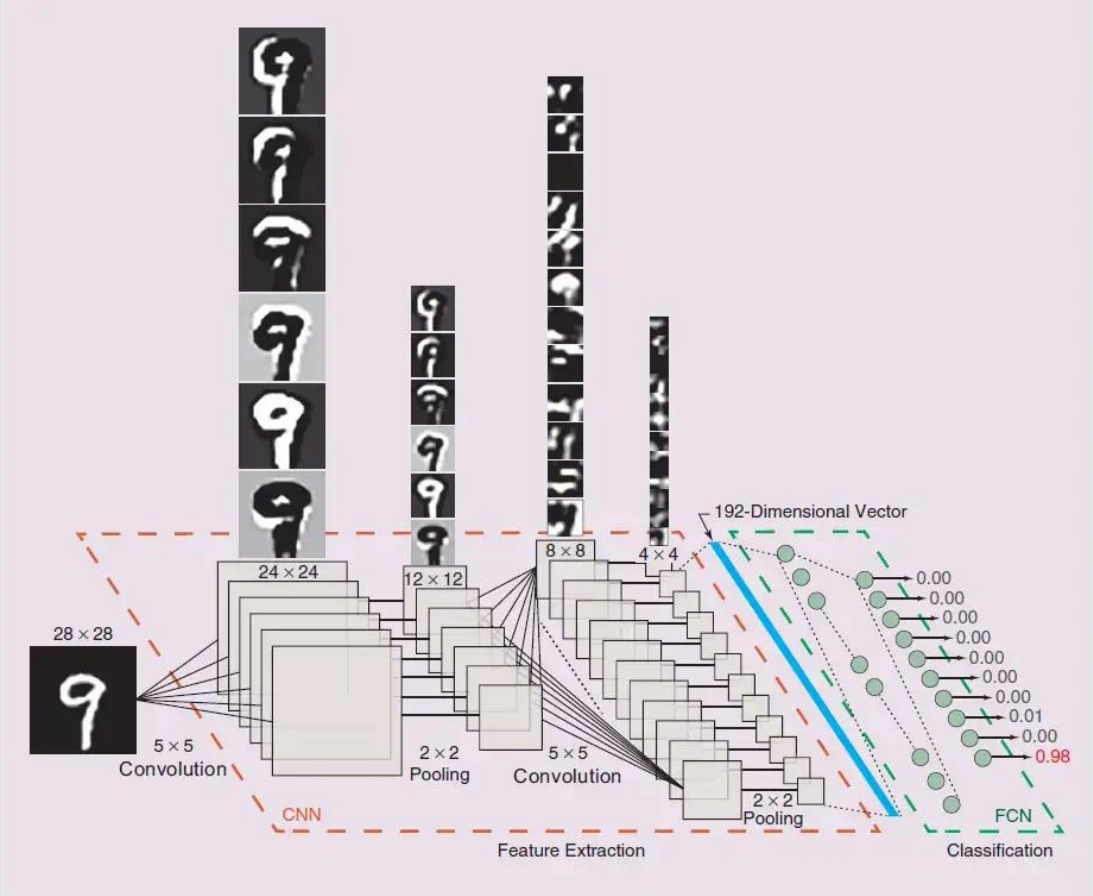


Gambar 2.5 Neural Network di CNN. [5]

Pada satu tahap, CNN secara umum terdiri dari tiga volume, masing-masing terdiri dari peta masukan (*input maps*), peta fitur (*feature maps*), dan peta fitur gabungan (*pooled maps*). *Pooled maps* tidak selalu digunakan di setiap tahap dan, di beberapa aplikasi.

Pada Gambar 2.6 di bawah, CNN dilatih untuk mengekstraksi fitur yang kemudian digunakan oleh *fully connected network* (FCN) untuk mengklasifikasikan angka tulisan tangan. Gambar *input* yang ditampilkan berasal dari *database* National Institute of Standards and Technology (NIST).

[5] https://miro.medium.com/max/1100/1*ZA39SpvHpVwc92cJXvzrGw.webp

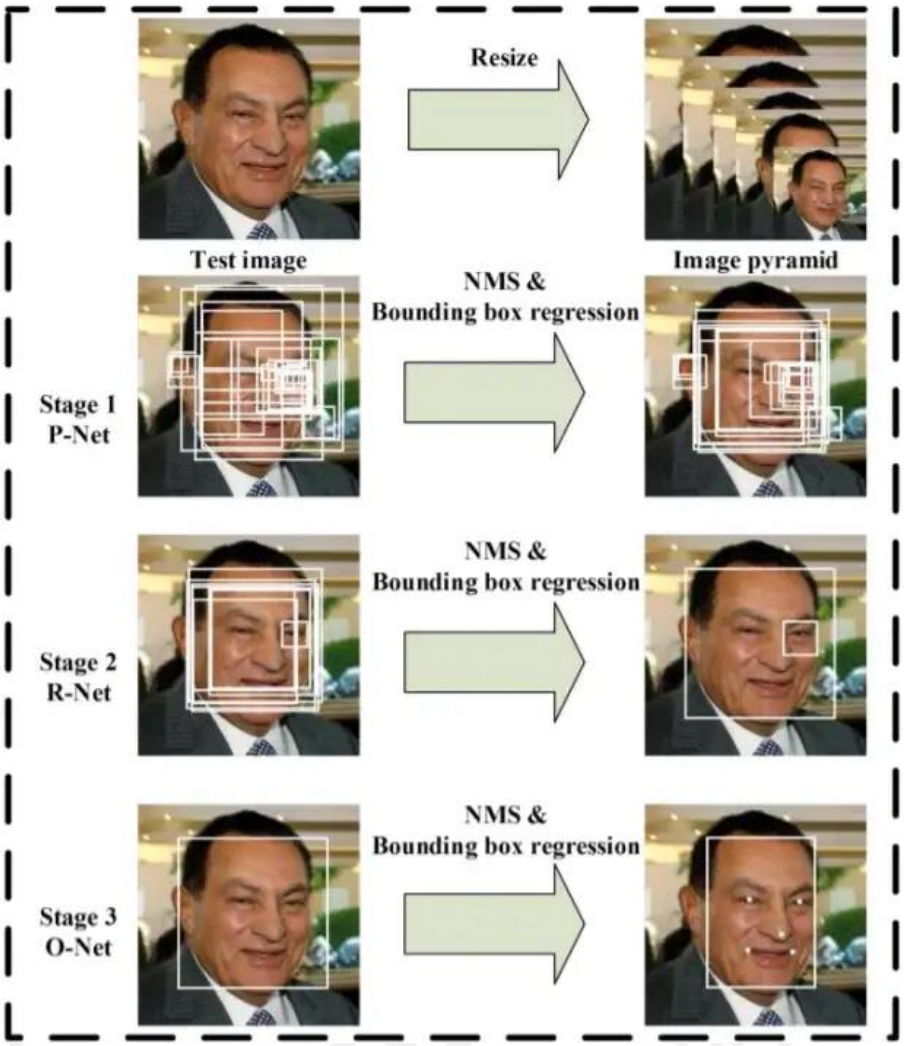


Gambar 2.6 Ilustrasi ekstraksi fitur di CNN. [6]

2.2.7 MTCNN

Dalam *Multi-task based convolution neural network* (MTCNN), digabungkan *multi-task learning* (MTL) dengan kerangka kerja CNN dengan berbagi beberapa lapisan di antara tugas-tugas yang berbeda (Yin & Liu, 2018). Pada dasarnya ada tiga model *deep face recognition*, Lightened CNN, CASIA-Net dan SphereFace, semuanya adalah model yang relatif ringan. Hal ini memungkinkan tidak hanya menyempurnakan model tetapi juga melatihnya dari awal dengan menggunakan *dataset facial depth images* yang relatif kecil. Di MTCNN digunakan CASIANet dengan tiga modifikasi. Pertama, *normalisasi batch* (BN) diterapkan untuk mempercepat proses pelatihan. Kedua, *contrastive loss* dikecualikan untuk menyederhanakan fungsi *loss*. Ketiga, dimensi *fully connected* layer diubah sesuai dengan tugas yang berbeda. *Pipeline framework* berikut mencakup jaringan konvolusional dalam tiga tahap *multi-task* ditunjukkan di Gambar 2.7 bawah ini,

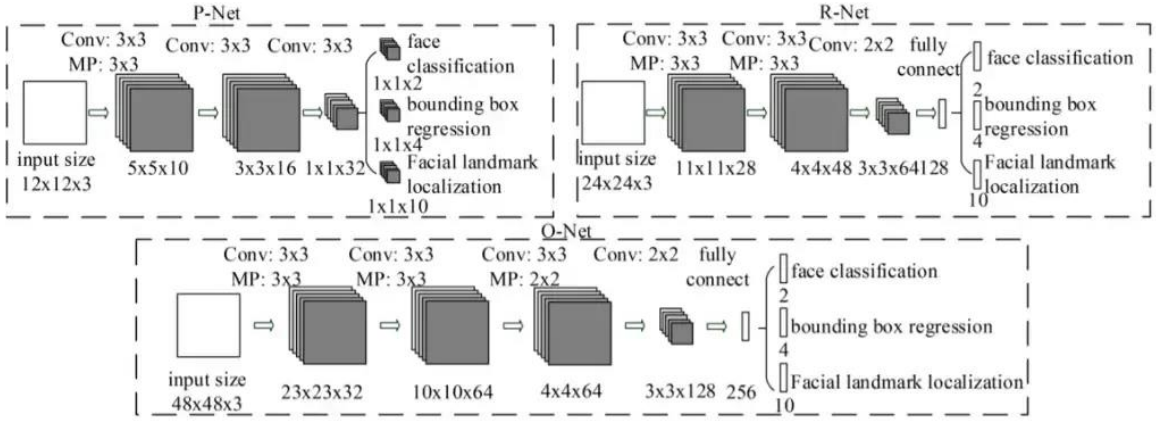
[6] https://miro.medium.com/max/1100/1*Xvugks1vUy4whF1Rdn3GEA.webp



Gambar 2.7 Pipeline MTCNN. [7]

Pada gambar di atas, Ada tiga tahap: Pada tahap pertama, jendela kandidat diproduksi melalui *fast Proposal Network* (P-Net), pada tahap kedua menyempurnakan kandidat ini melalui *Refinement Network* (R-Net) dan di tahap ketiga yang juga disebut *output network* (O-Net) yang menghasilkan *final bounding box* dan posisi *facial landmarks*. Arsitektur P-Net, R-Net, dan O-Net, di mana "MP" berarti *max pooling* dan "Conv" berarti konvolusi adalah seperti di Gambar 2.8 bawah ini,

[7] https://miro.medium.com/max/828/1*PTaoe90DKLVhFso4T57dDw.webp



Gambar 2.8 Arsitektur P-Net, R-Net, O-Net. [8]

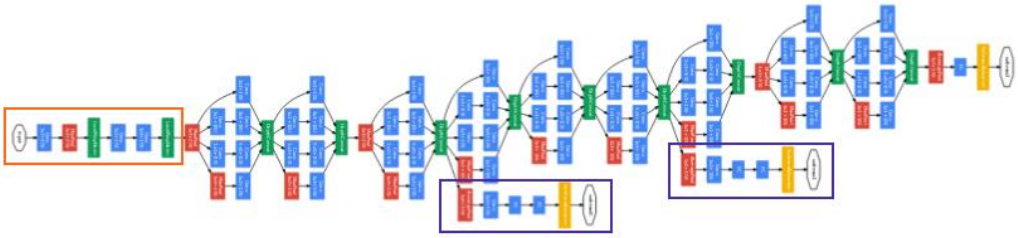
Dalam arsitektur jaringan di atas yang masing-masing terdiri dari lima blok termasuk dua lapisan *convolutional* dan satu lapisan *pooling*. BN dan ReLU digunakan setelah setiap lapisan *convolutional*, yang dihilangkan dari gambar untuk kejelasan gambar. Demikian pula, tidak ada ReLU yang digunakan setelah lapisan conv52 untuk mempelajari representasi fitur yang ringkas, dan lapisan *dropout* dengan rasio 0,4 diterapkan setelah lapisan pool5.

2.2.8 Inception

Inception merupakan pengembangan dari *Convolutional Neural Network* (CNN) yang diperkenalkan oleh Szegedy, dkk pertama kali pada artikel tahun 2014 dengan judul "Going Deeper with Convolutions". Jaringan konvolusional yang sangat dalam (*very deep convolutional networks*) baru-baru ini menjadi pusat pengembangan kinerja pengenalan gambar. Salah satu contohnya adalah arsitektur Inception, yang menawarkan performa sangat baik dengan daya komputasi yang relatif kecil (Szegedy et al., 2015).

Arsitektur *deep convolutional* Inception diperkenalkan sebagai GoogLeNet, di sini bernama Inception-v1. Kemudian arsitektur Inception disempurnakan dengan berbagai cara, pertama dengan pengenalan normalisasi *batch* (Inception-v2). Nanti dengan ide faktorisasi tambahan di iterasi ketiga yang akan disebut sebagai Inception-v3. Arsitektur Inception V1 ditunjukkan di Gambar 2.9 bawah ini,

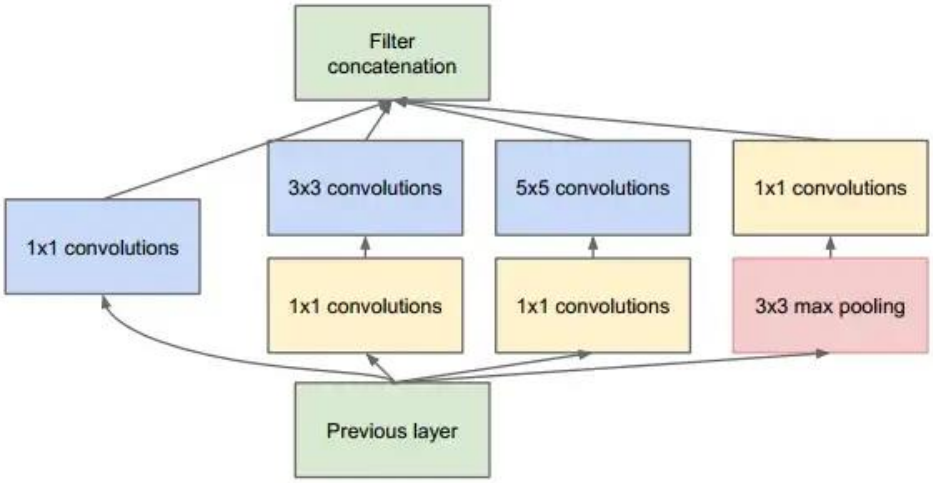
[8] https://miro.medium.com/max/1100/1*7tMjN2EQqoXXPTgv8haraQ.webp



Gambar 2.9 Arsitektur Google LeNet (Inception). [9]

Block Inception

Blok inception memiliki semuanya seperti terlihat pada Gambar 2.10. Ini memiliki konvolusi 1x1 diikuti oleh konvolusi 3x3, memiliki konvolusi 1x1 diikuti oleh konvolusi 5x5, ia memiliki lapisan *max pool* 3x3 diikuti oleh konvolusi 1x1 dan memiliki konvolusi 1x1 tunggal. Idenya adalah jika kita menggunakan semua konvolusi dan penyatuan dalam satu blok, beberapa di antaranya akan cukup efisien untuk mengekstrak beberapa informasi bermakna dari gambar. Untuk memastikan dimensi citra tetap terjaga, konvolusi 3x3 memiliki *padding* 1, dan layer 5x5 memiliki *padding* 2 sehingga citra masukan dan keluaran memiliki ukuran yang sama. Dan akhirnya, mereka semua ditumpuk bersama.

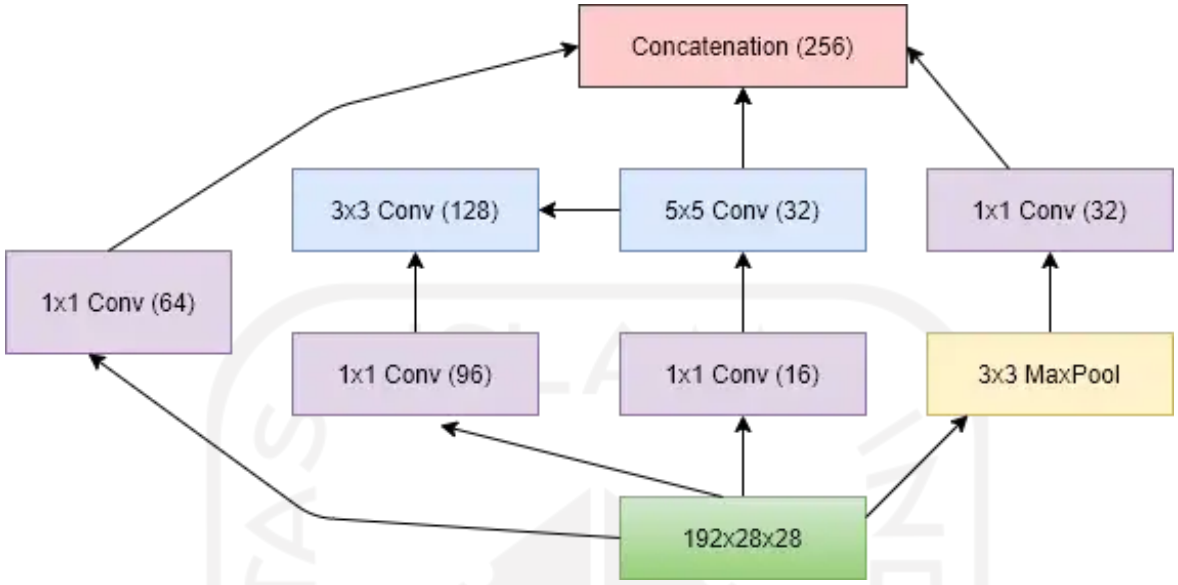


Gambar 2.10 Blok Inception. [10]

Jumlah filter di setiap lapisan di blok Inception dirancang sedemikian rupa sehingga didapatkan jumlah *channel* yang diinginkan sebagai keluaran untuk blok berikutnya. Misalnya,

[9] https://miro.medium.com/max/1100/1*G9ir2O2wEcSZtglhcWsnNA.webp
[10] https://miro.medium.com/max/828/1*H_ZYfZ52t6M4UmML6a6pAQ.webp

di blok inception pertama, seperti yang ditunjukkan pada Gambar 2.11, jumlah *channel* berjumlah 256.



Gambar 2.11 Blok Inception pertama. [11]

Blok Inception dirancang sedemikian rupa sehingga yang membutuhkan parameter yang lebih sedikit dan kompleksitas komputasi yang lebih sedikit daripada satu lapisan konvolusi 3x3 atau 5x5, seperti yang ditunjukkan pada Gambar 2.12. Jika kita memiliki 256 *channel* di lapisan *output*, Inception hanya membutuhkan 16.000 parameter dan biaya hanya 128 Mega FLOPS, sedangkan lapisan konvolusional 3x3 akan membutuhkan 44.000 parameter dan biaya 346 Mega FLOPS, dan lapisan konvolusional 5x5 akan membutuhkan 1.22.000 parameter dan biaya 963 Mega FLOPS. Jadi blok Inception, pada dasarnya menyelesaikan pekerjaan yang sama sebagai lapisan konvolusional tunggal, dengan memori dan efisiensi komputasi yang jauh lebih baik.

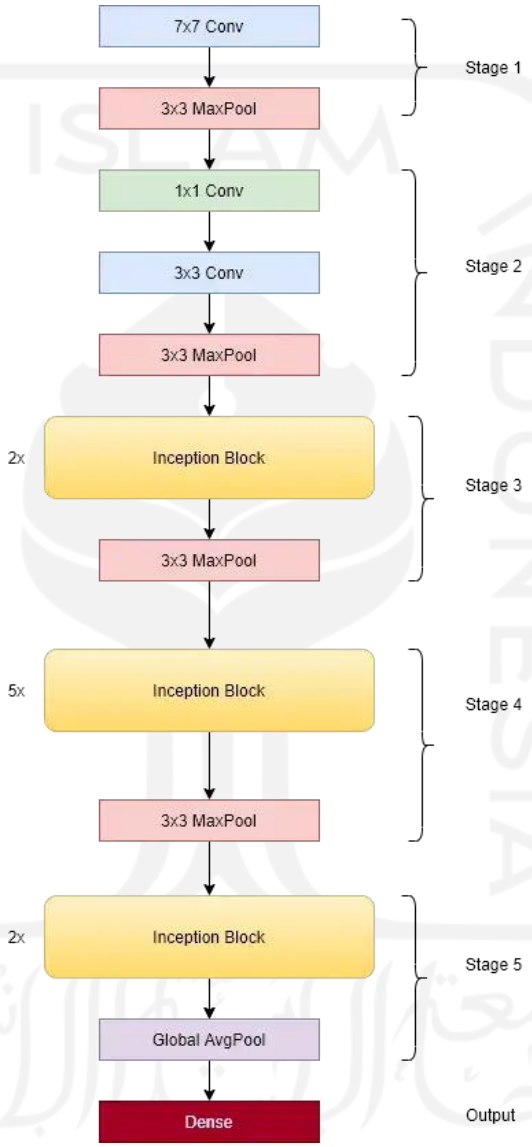
	# parameters	FLOPS
Inception	0.16 M	128 M
3x3 Conv	0.44 M	346 M
5x5 Conv	1.22 M	963 M

Gambar 2.12 Perbandingan blok Inception vs 3x3 dan 5x3 convolution. [12]

[11] https://miro.medium.com/max/828/1*rD-Zp3zQmKR91jRiv3WjIw.webp
[12] https://miro.medium.com/max/640/1*pm5CeyqeWFjrpkOOpjssBg.webp

Jaringan Inception Disederhanakan:

Jika kita melihat Gambar 2.9, maka jaringan Inception terlihat cukup menakutkan. Maka untuk mempermudah jaringan Gambar 2.13 dibuat. Gambar 2.13 persis sama dengan Gambar 2.9, tetapi seluruh arsitektur direpresentasikan sebagai blok. Gambar 2.13 menunjukkan bahwa jaringan awal tidak berbeda dengan jaringan vanilla biasa yang sudah kita kenal. Jaringan Inception memiliki 5 tahap.



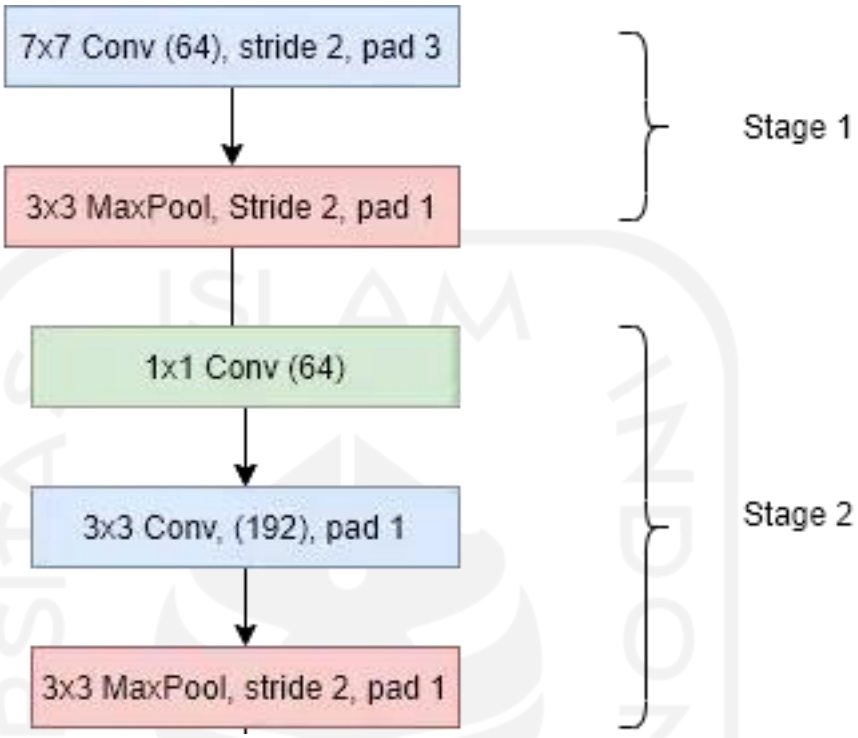
Gambar 2.13 Jaringan Inception disederhanakan. [13]

Tahap 1 & 2:

Jaringan dimulai dengan ukuran gambar 224x224x3. Kemudian melalui Conv 1x1, MaxPool 3x3, Conv 1x1, Conv 3x3, dan MaxPool 3x3, dan menghasilkan gambar berukuran

[13] https://miro.medium.com/max/640/1*VoTKM-H4OD3TfJewNVxbkg.webp

192x28x28. Bagian jaringan ini sangat mirip dengan AlexNet, hanya saja ukuran gambar di AlexNet dikurangi menjadi 256x12x12. Ilustrasi pada Gambar 2.14.

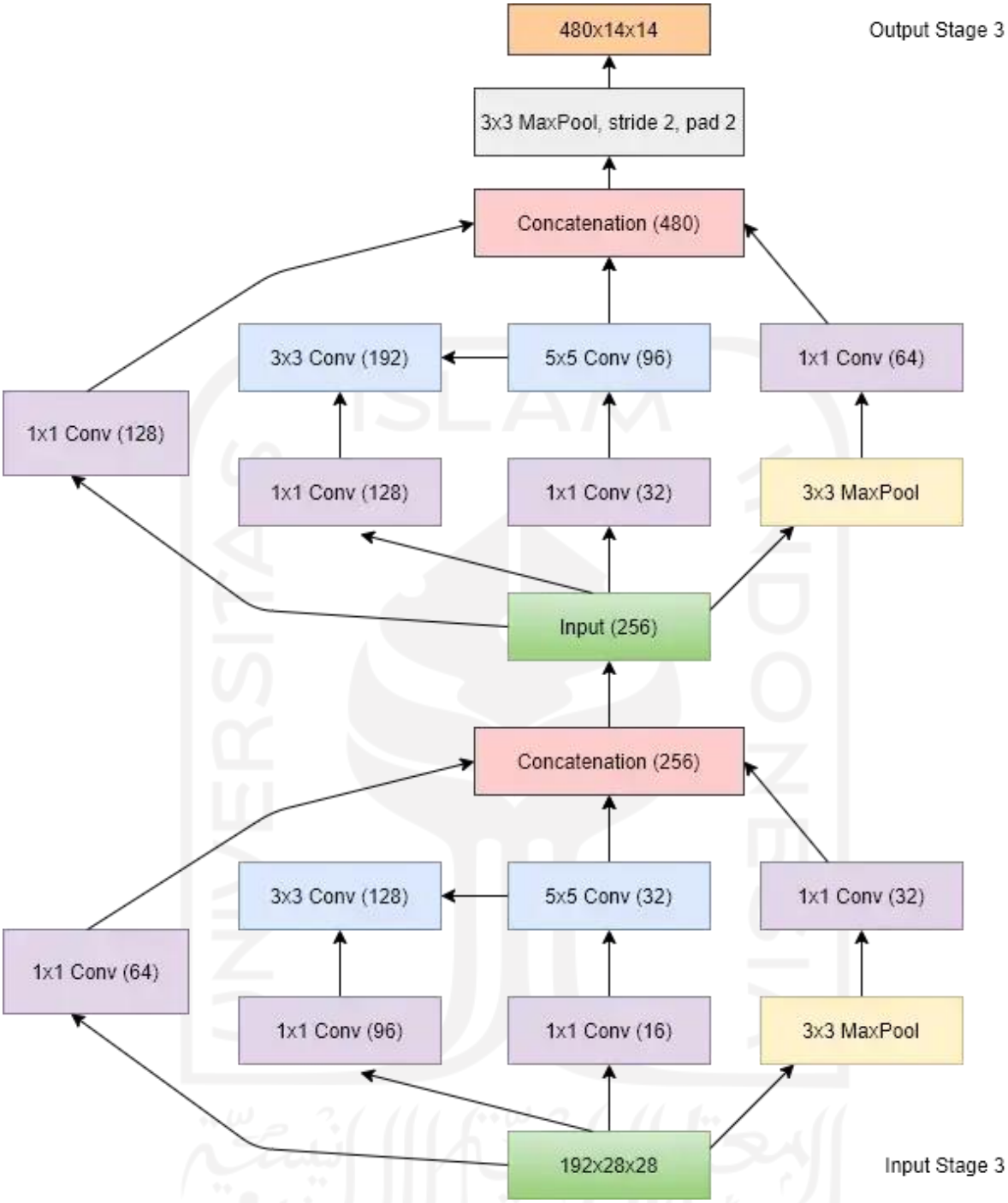


Gambar 2.14 Tahap 1 dan 2 jaringan Inception. [14]

Tahap 3 :

Seperti yang kita lihat pada Gambar 2.15, tahap 3 memiliki dua blok Inception dan pada akhirnya lapisan Max Pool. Namun blok inception tidak memiliki alokasi *channel* yang sama, seperti yang terlihat pada gambar. Blok 1 memiliki 256 saluran, sedangkan blok 2 memiliki 480 saluran. Jadi gambar *input* 192x28x28 sekarang menjadi 480x14x14, setelah dua blok awal dan lapisan MaxPooling

[14] https://miro.medium.com/max/640/1*0vskjgpEPlCbLfbv-hHTWw.webp



Gambar 2.15 Tahap 3 jaringan Inception. [15]

[15] https://miro.medium.com/max/828/1*cfsMjTISu6bAZ3a43--RiQ.webp

Tahap 4 dan 5:

Tahap 4 dan 5 sangat mirip dengan tahap 3. Tahap 4 memiliki 5 blok inception yang diikuti oleh MaxPool, dan tahap 5 memiliki 2 blok inception yang diikuti oleh GlobalAveragePool.

Tahap 4

Ukuran gambar masukan — 480x14x14

Inception Block 1–512 *channel* (*channel* keluaran yang ditingkatkan)

Inception Block 2–512 *channel*

Inception Block 3–512 *channel*

Inception Block 4–512 *channel*

Inception Block 5–832 *channel* (*channel* keluaran yang ditingkatkan)

Ukuran gambar keluaran setelah Max Pool — 832x7x7

Tahap 5

Ukuran gambar masukan — 832x7x7

Inception Block 1–832 *channel*

Inception Block 2–1024 *channel* (*channel* keluaran yang ditingkatkan)

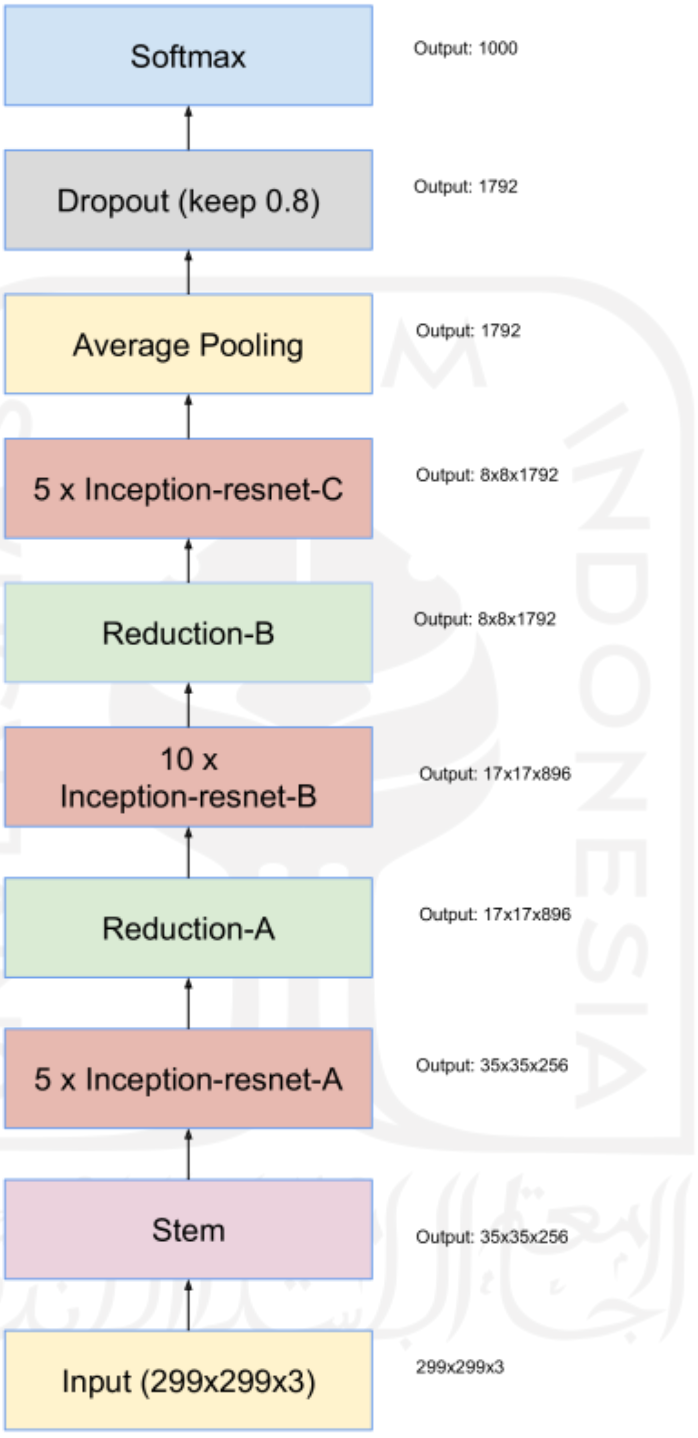
Ukuran gambar keluaran setelah GlobalAvgPool — 1024x1x1

2.2.9 Inception-ResNet-v1

Jaringan Inception ResNet adalah jaringan hybrid yang terinspirasi oleh kinerja Inception dan ResNet. Ada dua versi hibrida ini; Inception-ResNet v1 dan v2. Walaupun prinsip operasinya sama, Inception-ResNet v2 lebih akurat, namun biaya komputasinya lebih tinggi daripada jaringan Inception-ResNet v1 sebelumnya (Szegedy et al., 2017).

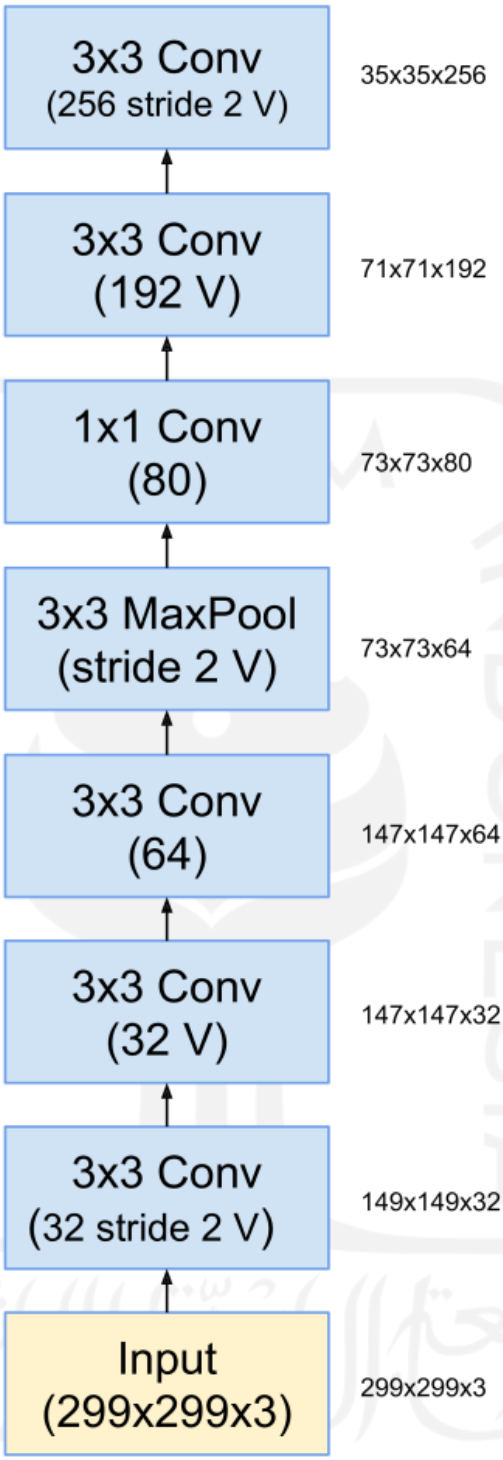
Pada Gambar 2.16 ditunjukkan skema jaringan Inception-ResNet-v1 dimana ada blok "Stem" yang masih dibagi lagi seperti pada Gambar 2.17. Seperti Inception v4, Inception-ResNet menggabungkan penggunaan 3 modul utama yang berbeda (Gambar 2.18) dan blok pengurangan (*reduction*) yang terlihat pada Gambar 2.19. Namun dikarenakan jaringan adalah gabungan dari Inception v4 dan ResNet, tugas utama Inception-ResNet adalah menghubungkan *output* modul Inception ke *input* (yaitu data dari lapisan sebelumnya). Supaya berfungsi, dimensi output modul Inception dan dimensi input lapisan sebelumnya harus sama. Oleh karena itu, di sini pembagian faktor (*factorization*) menjadi penting untuk menjawab dimensi tersebut. Namun, penelitian lebih lanjut menunjukkan bahwa ketika jumlah filter

konvolusi melebihi 1000, jaringan mati. Activation Scaling kemudian diperkenalkan, yang memecahkan masalah kematian jaringan.



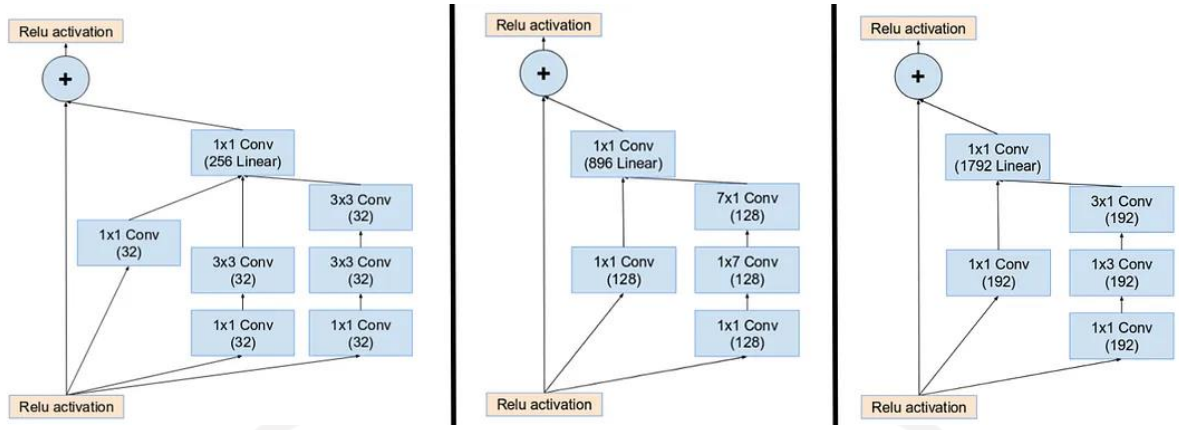
Gambar 2.16 Skema untuk jaringan Inception-ResNet-v1.[16]

[16] https://media.arxiv-vanity.com/render-output/6458002/x15.png

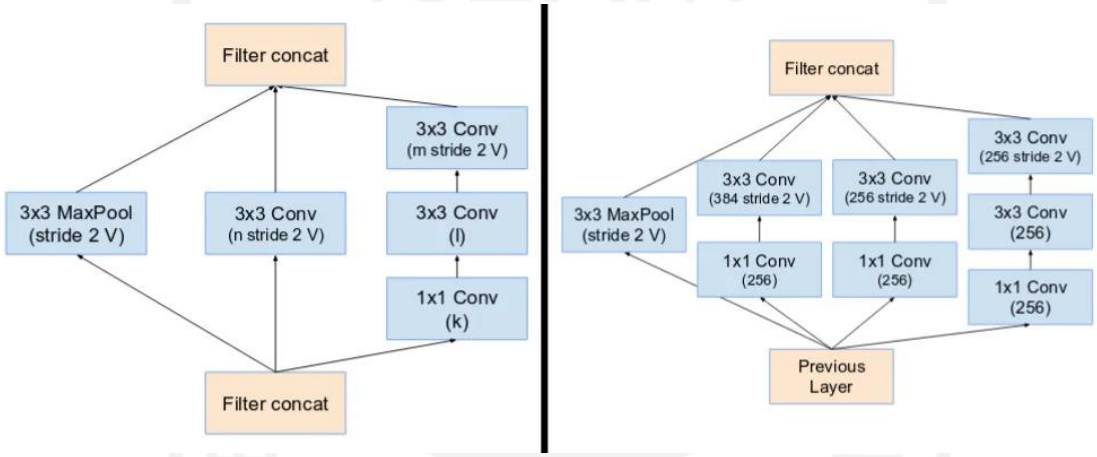


Gambar 2.17 Stem dari jaringan Inception-ResNet-v1. [17]

[17] https://media.arxiv-vanity.com/render-output/6458002/x14.png



Gambar 2.18 3 modul Inception-ResNet: A,B, dan C (dari kiri ke kanan).[18]



Gambar 2.19 blok reduction A dan B (kiri dan kanan). [19]

2.2.10 FaceNet

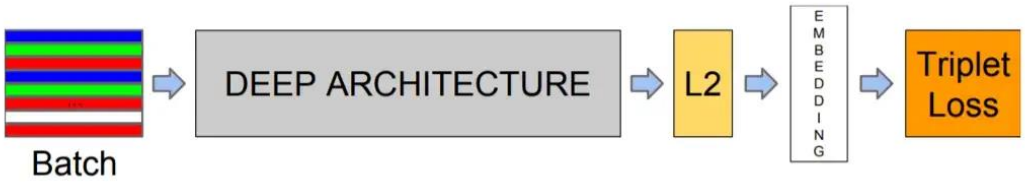
Pada tahun 2015, peneliti di Google telah mencapai hasil terbaik pada rangkaian *dataset* tolok ukur pengenalan wajah dan sistem yang disebut FaceNet (Schroff et al., 2015). Sistem FaceNet adalah implementasi model sumber terbuka pihak ketiga dan juga tersedia sebagai model *pre-trained*. Sistem FaceNet dapat digunakan untuk mengekstraksi fitur berkualitas tinggi dari wajah, yang disebut *face embeddings*, yang dapat digunakan untuk melatih sistem identifikasi wajah. Ini secara langsung mempelajari pemetaan dari gambar wajah ke ruang Euclidean yang padat di mana jarak secara langsung sesuai dengan ukuran kesamaan wajah, yaitu wajah orang yang sama memiliki jarak yang kecil dan wajah orang yang berbeda memiliki jarak yang jauh.

Menggunakan *FaceNet Embedding* sebagai vektor fitur kita dapat mengimplementasikan tugas-tugas seperti pengenalan wajah (siapa orang ini), verifikasi (apakah ini orang yang sama) atau pengelompokan (menemukan orang umum di antara wajah-wajah ini) juga yaitu verifikasi wajah hanya melibatkan ambang jarak antara keduanya

[18] https://miro.medium.com/v2/resize:fit:1100/format:webp/1*WyqyCKA4mP1jsl8H4eHrjg.jpeg
[19] https://miro.medium.com/v2/resize:fit:1100/format:webp/1*QY-g6oMF_6-v7N668HNvvA.jpeg

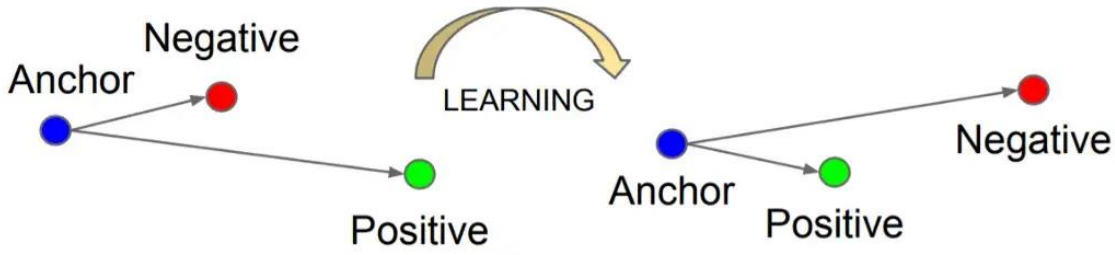
penyematan (*embedding*); *recognition* menjadi masalah klasifikasi k-NN, dan pengelompokan dapat dicapai dengan menggunakan teknik *off-the-shelf* seperti k-means atau agglomerative clustering.

Metode ini (FaceNet) menggunakan *deep convolutional network* yang dilatih untuk secara langsung mengoptimalkan penyematan (*embedding*) itu sendiri, bukan lapisan intermediate bottleneck seperti pada pendekatan *deep learning* sebelumnya. Untuk melatih gambar menggunakan FaceNet, digunakan triplet dari tambalan wajah yang serasi (*matching*) atau tidak cocok (*non-matching*) secara kasar yang dihasilkan menggunakan metode mining triplet online baru. Dalam struktur model, terlihat pada Gambar 2.20 jaringan terdiri dari lapisan *input batch* dan *deep CNN* diikuti dengan normalisasi L2, yang menghasilkan *face embedding*. Ini diikuti oleh triplet loss selama *training*.



Gambar 2.20 Pipeline triplet loss. [20]

Triplet loss adalah fungsi *loss* untuk jaringan saraf tiruan di mana input dasar (*anchor*) dibandingkan dengan *input* positif (*truthy*) dan input negatif (*falsy*). Jarak dari *input* baseline (*anchor*) ke *input* positif (*truthy*) diminimalkan, dan jarak dari input baseline (*anchor*) ke *input* negatif (*falsy*) dimaksimalkan diilustrasikan pada Gambar 2.21.



Gambar 2.21 Ilustrasi triplet loss. [21]

2.2.11 Face Embeddings

Penyematan wajah adalah representasi vektor dari fitur yang diekstraksi dari wajah. Sangat membantu untuk menemukan kesamaan antara dua vektor fitur. Misalnya, vektor lain yang

[20] https://miro.medium.com/max/1100/1*8gARcWmh84p_MMNiWBFmTQ.webp
[21] https://miro.medium.com/max/1100/1*U9dGNZnVQImM-fZTmJQ4Dw.webp

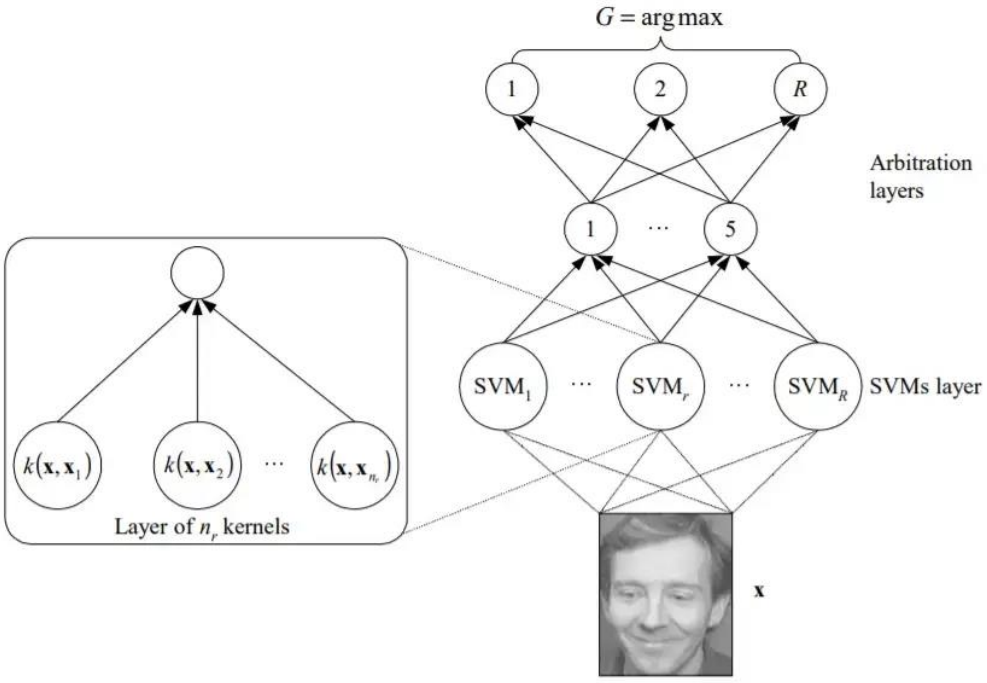
dekat (menurut beberapa ukuran dari vektor fitur) mungkin adalah orang yang sama, sedangkan vektor lain yang jauh (menurut beberapa ukuran dari vektor fitur) mungkin adalah orang yang berbeda. Model pengklasifikasi yang ingin kita kembangkan akan mengambil penyisipan wajah sebagai *input* dan memprediksi identitas wajah tersebut.

Model FaceNet akan menghasilkan penyematan (*embedding*) untuk gambar wajah tertentu. Model FaceNet dapat digunakan sebagai bagian dari pengklasifikasi sendiri, atau dapat digunakan model FaceNet untuk melakukan *pre-process* wajah untuk membuat *face embedding* yang dapat disimpan dan digunakan sebagai *input* untuk model pengklasifikasi. Pendekatan terakhir ini lebih disukai karena model FaceNet berukuran besar dan lambat untuk membuat *face embedding*.

2.2.12 Support Vector Machine Classifier

Support Vector Machine (SVM) adalah metode klasifikasi biner dan Pengenalan wajah adalah masalah kelas K di mana K adalah jumlah individu yang diketahui. Tujuan SVM adalah untuk mencari *hyper-plane* pemisah yang optimal yang meminimalkan risiko kesalahan klasifikasi. Setelah jumlah *training* yang layak, SVM dapat memprediksi apakah *input* termasuk dalam salah satu dari dua kategori. Ini dilakukan dengan membuat *hyperplan*e di ruang berdimensi tinggi dan kemudian memetakan *input* ke titik-titik di ruang tersebut (Burges, 1998).

Model pengenalan wajah menemukan perbedaan antara dua citra wajah dalam ruang perbedaan. Dapat dirumuskan pengenalan wajah sebagai masalah dua kelas yang kelasnya adalah: perbedaan antara wajah orang yang sama dan orang yang berbeda. Dengan memodifikasi interpretasi permukaan keputusan yang dihasilkan oleh SVM, disini dihasilkan metrik kesamaan antara wajah yang dipelajari dari contoh perbedaan antar wajah. Gambar 2.22 di bawah menunjukkan Arsitektur jaringan SVM–NN untuk pengenalan wajah,



Gambar 2.22 Arsitektur SVM-NN. [22]

2.2.13 OpenCV

Open Source Computer Vision Library (OpenCV) adalah *library* perangkat lunak untuk pemrosesan gambar dinamis secara *realtime* (Zein, 2018). OpenCV dapat diimplementasikan dengan *script* Python untuk memproses gambar digital dari berbagai sumber seperti video, kamera atau protokol lain seperti *Real Time Streaming Protocol* (RTSP). Pada penelitian ini, OpenCV digunakan untuk mengumpulkan data foto, mengolah *frame* IP CCTV dan mengenali pola senyum.

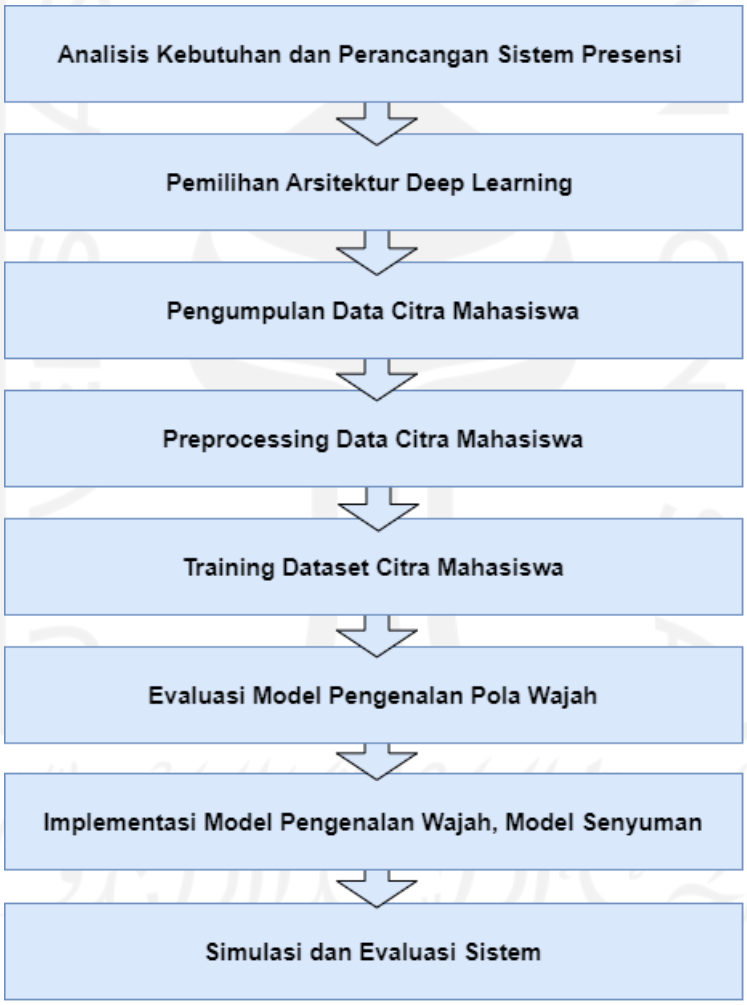
2.2.14 Dlib

Dlib adalah *library* pembelajaran mesin yang ditulis dalam bahasa pemrograman C++ (Wahyudiana & Budi, 2019). Pada penelitian ini, DLib digunakan untuk memotong (*cropping*) atau mendeteksi bagian wajah dari gambar yang ditangkap oleh IP CCTV.

[22] https://miro.medium.com/max/1100/1*9nrjgU7Yy50_fP20HY2iqA.webp

BAB 3
Metodologi

Penelitian ini menggunakan beberapa tahapan proses yang terdiri dari analisis kebutuhan dan perancangan sistem presensi, pemilihan arsitektur *deep learning*, pengumpulan data citra mahasiswa, *preprocessing* data citra mahasiswa, *training dataset* citra mahasiswa, evaluasi model pengenalan pola wajah, implementasi model pengenalan wajah & model senyuman, dan simulasi & evaluasi sistem. Gambar 3.1 menunjukkan alur metodologi penelitian.

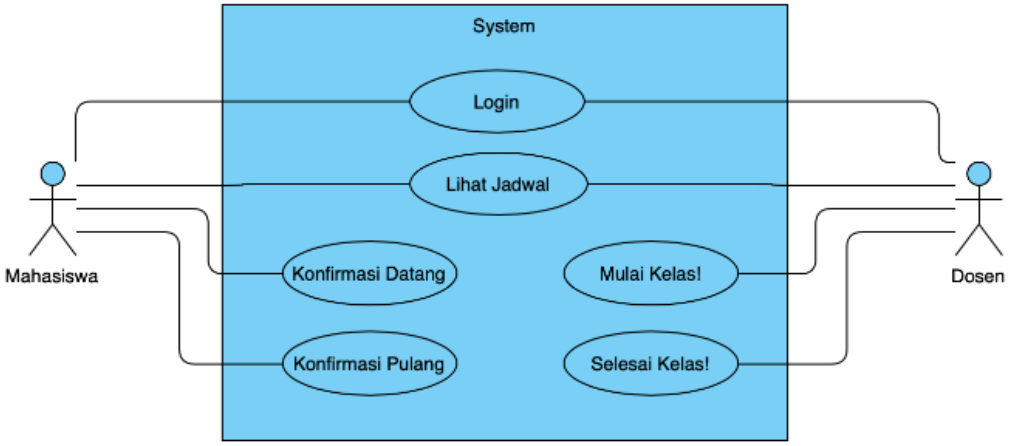


Gambar 3.1 Metodologi penelitian.

3.1 Analisis Kebutuhan dan Perancangan

3.1.1 Use Case Mahasiswa, Dosen

Untuk mengetahui fungsi/fitur apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut maka dibuatlah *use case* seperti Gambar 3.2. Aktor yang terlibat dalam sistem ini adalah Mahasiswa dan Dosen. Dosen mendapatkan fungsi seperti : Login, Lihat Jadwal, Mulai Kelas!, Selesai Kelas!; sedangkan mahasiswa memiliki fungsi seperti : Login, Lihat Jadwal, Konfirmasi Datang, Konfirmasi Pulang. Fungsi untuk mahasiswa dikembangkan dalam platform *mobile*, sedangkan fungsi untuk dosen dikembangkan dalam platform web.



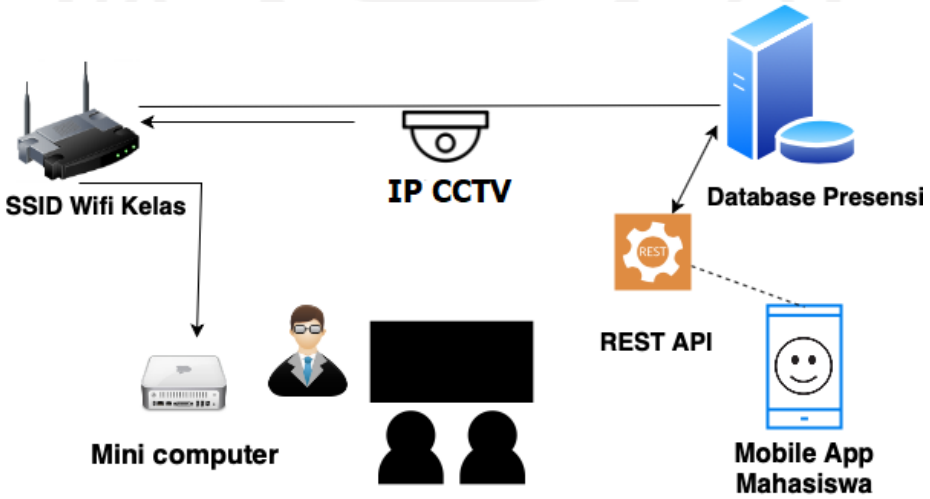
Gambar 3.2. Use case mahasiswa dan dosen.

3.1.2 Arsitektur Sistem Presensi Berbasis Pola Wajah dan Tersenyum

Pada Gambar 3.3 diperlihatkan ilustrasi arsitektur sistem presensi berbasis *deep learning*. Adapun untuk mempermudah Beberapa terminologi yang digunakan dalam arsitektur sistem:

- *Mini computer* : adalah seperangkat komputer yang berisi sistem operasi (Windows / Linux) dengan spesifikasi *middle* / tidak terlalu tinggi (minimal memori RAM 4GB, kapasitas *harddisk* 80GB), digunakan untuk mengakses aplikasi presensi berbasis web (melalui *browser*) yang diperuntukkan untuk dosen, dan untuk memasang *service* Python yang menerapkan *deep learning* untuk *face recognition*.
- SSID Wifi Kelas : Koneksi ke jaringan *local* kampus yang menghubungkan dengan beberapa layanan, umumnya disertai juga dengan koneksi internet.

- IP CCTV: Perangkat kamera untuk menangkap dan merekam kegiatan di dalam kelas, perangkat ini disertai IP Address sehingga dapat diakses melalui perangkat atau aplikasi lain.
- *Database* Presensi : Basis data yang menyimpan data jadwal perkuliahan, ruangan kelas, data dan akun dosen, data dan akun mahasiswa disertai NIM. Umumnya *database* diproteksi dan tidak sembarang dapat diakses sehingga hanya orang tertentu maupun aplikasi tertentu yang diijinkan.
- REST API : Sebuah layanan berbasis web service yang menjadi jembatan atau penghubung antara basis data dengan aplikasi di platform lain seperti *mobile* atau *desktop*.
- Mobile App Mahasiswa : Sebuah aplikasi yang diperuntukkan untuk presensi mahasiwa yang perlu di-*install* di *smartphone* masing-masing mahasiswa, tujuannya adalah untuk verifikasi kehadiran dan konfirmasi menggunakan senyuman.

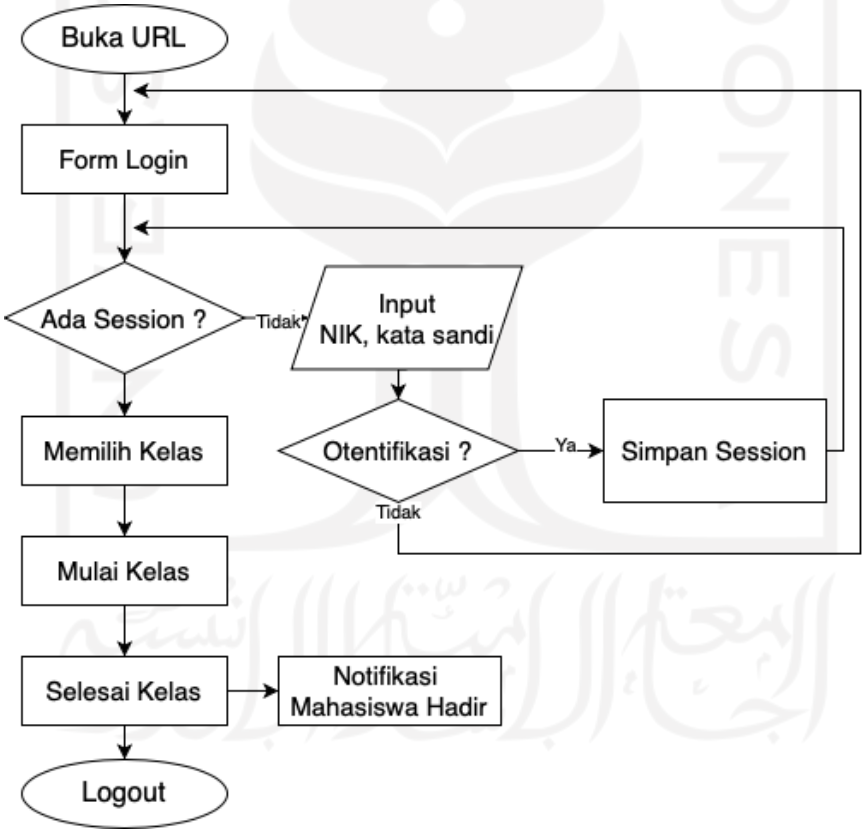


Gambar 3.3. Arsitektur sistem presensi berbasis pola wajah dan tersenyum.

Sistem ini diimplementasikan dalam ruangan dengan IP CCTV, *mini computer* dan *wireless base station* yang terhubung dengan jaringan kampus. Skrip Python yang mengimplementasikan model *deep learning* ditempatkan di komputer mini ini. Komputer mini tersebut terhubung dengan IP CCTV dan dalam jaringan kampus. Data kehadiran dikirim ke *database* pusat melalui Representative State Transfer Application Programming Interface (REST API). Aplikasi seluler siswa juga terhubung melalui REST API ini, memungkinkan untuk melihat informasi kehadiran dan konfirmasi kehadiran melalui jaringan kampus

3.1.3 Flowchart Aplikasi Untuk Dosen berbasis Web

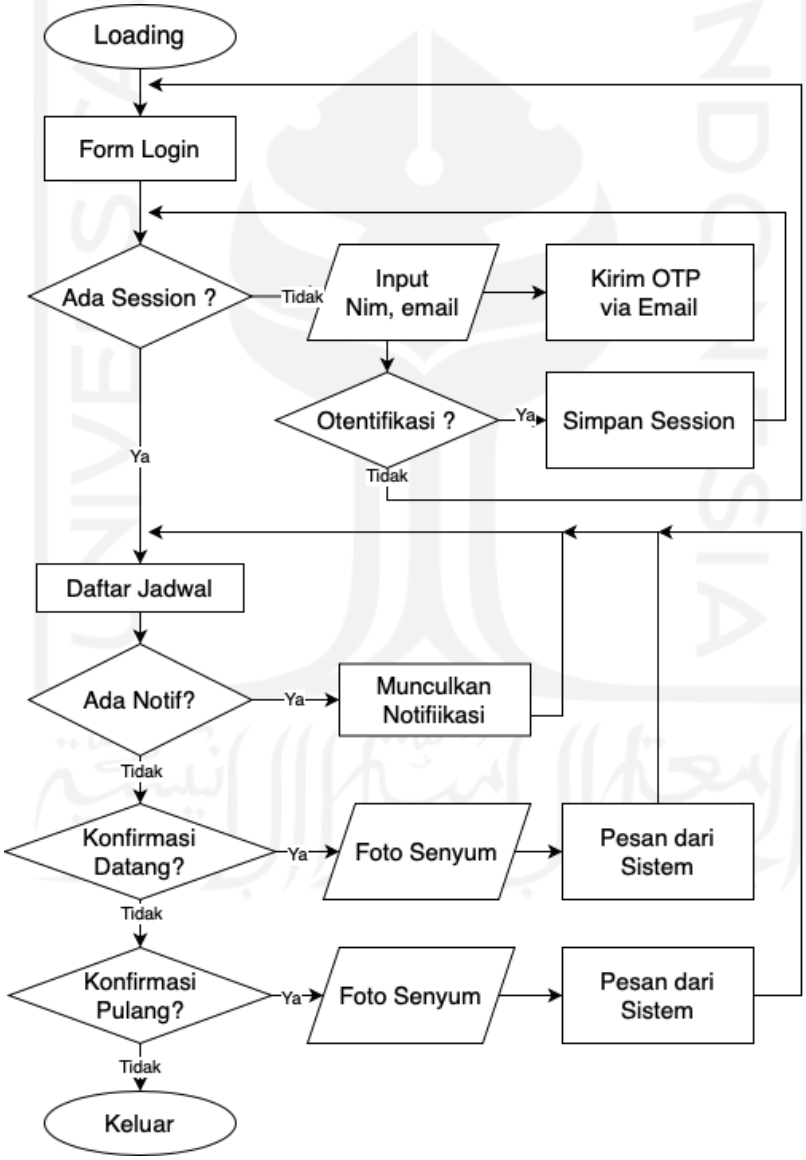
Selama proses kegiatan belajar mengajar (KBM), tugas dosen adalah membuka dan menutup KBM melalui aplikasi web. Aplikasi web ini terintegrasi dengan database akademik yang sudah ada. Aplikasi web ini menyediakan fitur untuk dosen melihat jadwal setiap hari. Setiap jadwal memiliki tombol mulai dan selesai. Tombol "Mulai" menunjukkan bahwa pelajaran telah dimulai dan kehadiran dicatat. *Input* dilakukan secara otomatis pada komputer mini menggunakan IP CCTV. Selain itu, nomor induk mahasiswa (NIM) diperoleh dari proses pengenalan wajah yang terekam oleh CCTV. Jika sesuai dengan *database* siswa yang terdaftar, maka informasi tersebut dimasukkan ke dalam tabel presensi. Saat pelajaran selesai, dosen menekan tombol "Selesai" untuk menunjukkan bahwa kelas telah berjalan. Flowchart bagaimana aplikasi web ini bekerja ditunjukkan pada Gambar 3.4.



Gambar 3.4. Flowchart aplikasi web untuk dosen.

3.1.4 Flowchart Aplikasi Untuk Mahasiswa berbasis Mobile

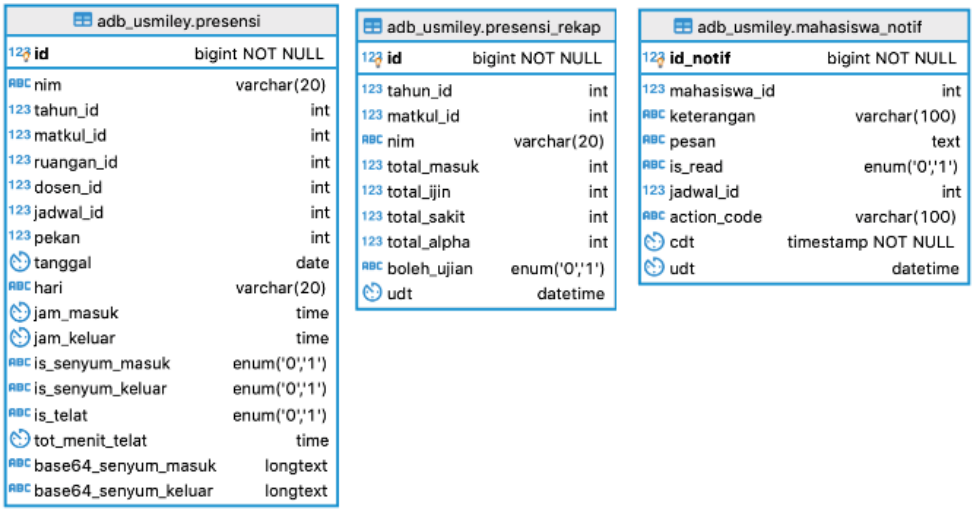
Penggunaan aplikasi mobile untuk mahasiswa ini mendukung pengoperasian sistem presensi ini. Melalui aplikasi mobile, mahasiswa akan menerima notifikasi seperti pesan telah masuk kelas atau nanti saat kelas selesai. Siswa harus memverifikasi dengan *selfie* tersenyum untuk kehadiran baik masuk maupun keluar. *Selfie* yang dikirim melalui aplikasi diverifikasi oleh aplikasi (menggunakan REST API identifikasi senyuman dengan pemanfaatan model klasifikasi senyuman). Jika "BENAR" diidentifikasi sebagai senyuman, setiap catatan kehadiran ditandai. Dua senyuman (presensi masuk dan presensi keluar) dihitung sebagai bukti kehadiran untuk setiap siswa. Flowchart bagaimana aplikasi *mobile* untuk mahasiswa ini bekerja ditunjukkan pada Gambar 3.5.



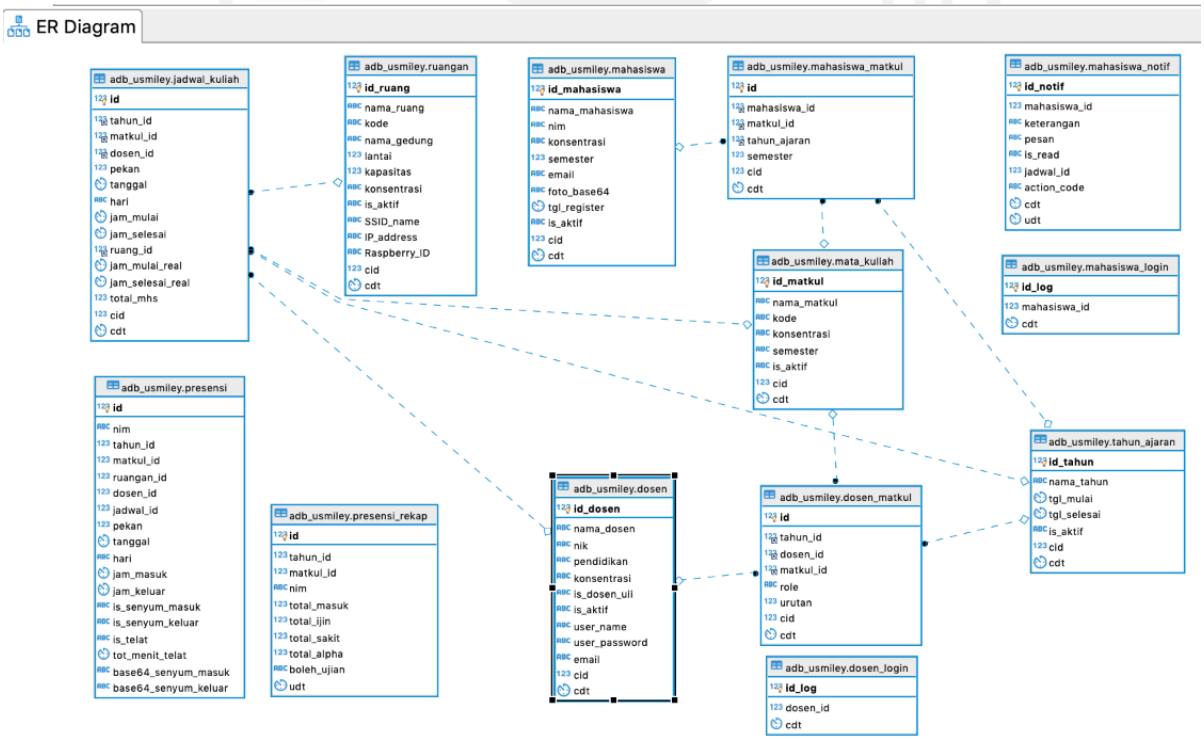
Gambar 3.5. Flowchart aplikasi mobile untuk mahasiswa.

3.1.5 Relasi Tabel Database

Gambar 3.6 menunjukkan perancangan tabel utama untuk presensi dan notifikasi. Total tabel yang dirancang sebagai penyimpanan master data maupun penyimpanan transaksi berjumlah 13 seperti terlihat pada Gambar 3.7, yaitu: dosen, dosen_login, dosen_matkul, jadwal_kuliah, mahasiswa, mahasiswa_login, mahasiswa_matkul, mahasiswa_notif, mata_kuliah, presensi, presensi_rekap, ruangan, dan tahun_ajaran.



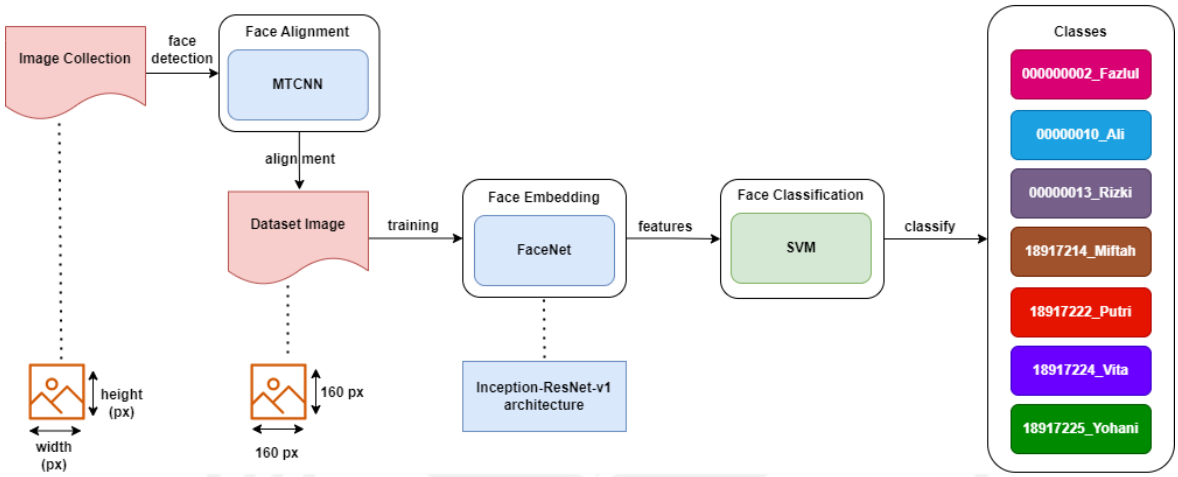
Gambar 3.6. Tabel utama presensi dan notifikasi.



Gambar 3.7 Keseluruhan tabel dalam ER diagram.

3.2 Pemilihan Arsitektur Deep Learning

Pada penelitian ini berdasarkan referensi atau studi pustaka dipilih beberapa teknologi yang akan digunakan. Teknologi *deep learning* yang digunakan untuk *face detection* dan *face alignment* menggunakan MTCNN, untuk *face embedding* menggunakan FaceNet (turunan CNN) seperti yang ditunjukkan pada Gambar 3.8. Arsitektur pemanfaatan deep learning dimulai dari blok "**Image Collection**" menuju "**Dataset Image**", lalu dari "**Dataset Image**" menuju "**Face Embedding**". Di bagian akhir untuk "**Face Classification**" dengan *multiclass* (7 mahasiswa) menggunakan algoritma *machine learning* SVM for Classifier (SVC).



Gambar 3.8 Arsitektur deep learning untuk training model pengenalan wajah.

Pemanfaatan FaceNet disini lebih digunakan untuk *feature extractor* atau ekstrasi fitur dari dataset citra yang dilatih. Pada penelitian tidak digunakan *triplet loss* sebagai layer terakhir untuk klasifikasi tapi digunakan *softmax*. FaceNet sendiri adalah konsep dimana untuk arsitektur *deep learning*-nya bisa beragam (Schroff et al., 2015). Pada penelitian ini digunakan arsitektur Inception_ResNet_v1 seperti terlihat pada Tabel 3.1 dengan detail per layer. Pemilihan arsitektur *deep learning* ini memiliki alasan karena akan digunakan *transfer learning* yang menggunakan arsitektur ini juga, dan Inception_ResNet_v1 memiliki hasil akurasi yang lebih baik dari Inception biasa (Szegedy et al., 2017). Sebagai input awal untuk arsitektur ini digunakan ukuran 149 x 149 pixel dimana untuk dataset citra akan disesuaikan ke ukuran tersebut.

Tabel 3.1 Inception_ResNet_v1 Network Structure

Layer	**Size-in**	**Size-out**	**Kernel**	**Stride, padding**	**ReLU _fn**	**Scale**
Conv2d_1a _3x3	149 x 149 x 32	147 x 147 x 32	3 x 3 x 32	2,V	True	-
Conv2d_2a _3x3	147 x 147 x 32	147 x 147 x 64	3 x 3 x 32	-	True	-
Conv2d_2b _3x3	147 x 147 x 64	73 x 73 x 64	3 x 3 x 64	-	True	-
MaxPool_3 a_3x3	73 x 73 x 64	73 x 73 x 80	3 x 3	2,V	True	-
Conv2d_3b _1x1	73 x 73 x 80	71 x 71 x 192	1 x 1 x 80	-	True	-
Conv2d_4a _3x3	71 x 71 x 192	35 x 35 x 256	3 x 3 x 192	-	True	-
Conv2d_4b _3x3	35 x 35 x 256	17 x 17 x 256	3 x 3 x 256	2,V	True	-
5 x Inception-resnet-A	17 x 17 x 256	17 x 17 x 256	Inception-resnet-A	-	True	0.17
Reduction-A	17 x 17 x 256	8 x 8 x 896	Reduction -A	-	True	-
10 x Inception-Resnet-B	8 x 8 x 896	8 x 8 x 896	Inception-Resnet-B	-	True	0.10
Reduction-B	8 x 8 x 896	3 x 3 x 1792	Reduction -B	-	True	-
5 x Inception-Resnet-C	3 x 3 x 1792	3 x 3 x 1792	Inception-Resnet-C	-	True	0.20
Inception-Resnet-C	3 x 3 x 1792	3 x 3 x 1792	Inception-Resnet-C	-	False	1.0
AvgPool2d	3 x 3 x 1792	1 x 1 x 1792	3 x 3	-	-	-

Sedangkan untuk pengenalan senyuman sendiri menggunakan model dari algoritma *Haar Cascade*. Algoritma *Haar cascade* (Lino et al., 2017) digunakan untuk mendeteksi wajah atau objek berupa citra digital. Algoritma ini menampilkan fungsi matematis dalam bentuk kotak yang menunjukkan nilai RGB dari setiap piksel. Kemudian Viola-Jones menemukan algoritma ini yang memproses setiap bidang dan menghasilkan banyak nilai dalam bentuk area gelap dan terang. Dan nilai-nilai tersebut digunakan sebagai dasar pengolahan citra, oleh karena itu dikenal dengan *Haar-Like Feature* (Sharma et al., 2015). OpenCV sendiri telah menyediakan beberapa model dari *Haar Like Feature* seperti deteksi wajah, mata, dan tersedia juga model untuk deteksi senyuman[23]. Pemanfaatan dari model ini adalah menggunakan XML dalam OpenCV sesuai bahasa atau *script* yang dipilih yaitu Python.

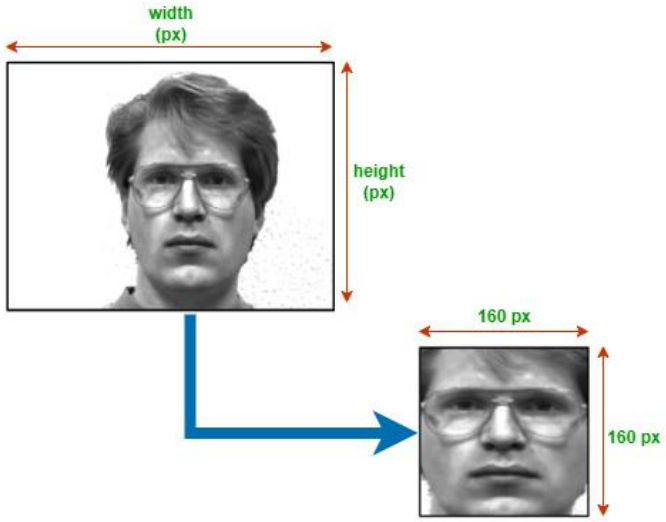
3.3 Pengumpulan Data Citra Mahasiswa

Dataset yang digunakan dalam penelitian ini berupa citra wajah yang dapat diperoleh menggunakan dengan dua cara. Cara pertama dilakukan dengan mengumpulkan foto mahasiswa dengan meminta secara langsung. Cara kedua adalah dengan mengumpulkan video mahasiswa yang berdurasi sekitar 1-2 menit dengan menggerakan posisi wajah. Video kemudian diproses dengan mengambil *frame* per detik, menghasilkan sekitar 60-120 *frame* setiap video.

3.4 Preprocessing Data Citra Mahasiswa

Sebelum membentuk *training set* dan *testing set*, foto-foto yang sebelumnya terkumpul diproses terlebih dahulu sesuai dengan arsitektur yang dipilih untuk *face detection*, yaitu MTCNN dengan dimensi 160x160 piksel. *Preprocessing* ini melakukan *cropping* citra terfokus pada area wajah atau disebut *face alignment*, dari kumpulan citra mahasiswa yang sudah dikumpulkan sebelumnya. (William et al., 2019). Proses ini mengubah dimensi citra semula dari *width* x *height* piksel menjadi 160 x 160 piksel seperti terlihat pada Gambar 3.9. Proses *preprocessing* ini menggunakan *script* Python yang telah tersedia dari peneliti FaceNet.[24]

[23] https://github.com/opencv/opencv/tree/master/data/haarcascades
[24] https://github.com/davidsandberg/facenet/blob/master/src/align/align_dataset_mtcnn.py



Gambar 3.9 Contoh cropping citra pada tahap preprocessing menggunakan MTCNN.

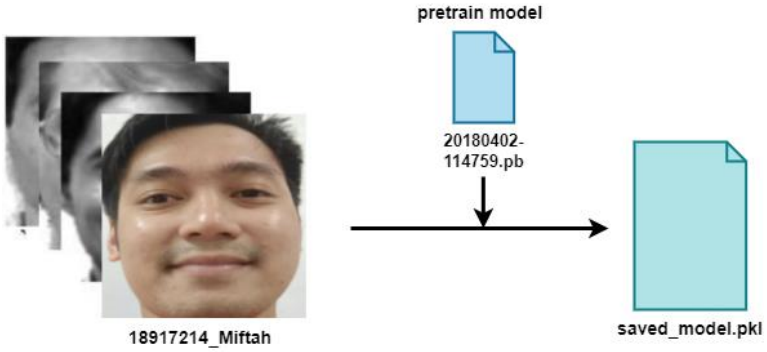
3.5 Training Dataset Citra Mahasiswa

Penelitian ini melibatkan 7 mahasiswa, masing-masing sejumlah 100 foto. 100 foto tersebut lalu dibagi menjadi tiga, yaitu : 10 foto digunakan untuk verifikasi data, 75 foto digunakan sebagai data latih dan 15 foto digunakan untuk validasi dengan tujuan agar model tidak terlalu *overfitting* (akurasi 100%) dan kurang bagus saat klasifikasi.

Proses training ini tidak dilakukan dari awal melainkan menggunakan model pretrain dari *transfer learning*. *Transfer learning* adalah teknik pembelajaran mesin di mana model dilatih dan dikembangkan untuk satu tugas dan kemudian digunakan kembali pada tugas kedua yang terkait. Ini mengacu pada situasi dimana apa yang telah dipelajari dalam satu setting dieksploitasi untuk meningkatkan optimasi di setting lain (Gao & Mosalam, 2018). Transfer learning biasanya diterapkan ketika ada *dataset* baru yang lebih kecil dari *dataset* asli yang digunakan untuk melatih model yang telah dilatih sebelumnya (Larsen-Freeman, 2013).

Model pretrain yang digunakan pada penelitian ini diambil dari eksperimen tim FaceNet sebelumnya[25] dengan hasil yang di-*share* di *link* Google Drive[26]. Model pretrain yang dipilih adalah model dengan akurasi LFW sebesar 99,65% dengan nama 20180402-114759.pb. Pada Gambar 3.10 ditunjukan proses *training* dimana dari pretrain model beserta *dataset* citra akan membentuk satu model untuk klasifikasi. Proses *training* ini menggunakan *script* Python yang telah tersedia dari peneliti FaceNet.[27]

[25] https://github.com/davidsandberg/facenet/wiki/Classifier-training-of-inception-resnet-v1
[26] https://drive.google.com/file/d/1EXPBSXwTaqrSC0OhUdXNmKSh9qJUQ55-/view
[27] https://github.com/davidsandberg/facenet/blob/master/src/classifier.py



Gambar 3.10 Ilustrasi output training.

3.6 Evaluasi Model Pengenalan Pola Wajah

Proses *training* lanjutan dari *transfer learning* akan menghasilkan model dengan ektensi .pkl (pickle). Selanjutnya untuk memastikan bahwa model sudah layak, dapat melakukan klasifikasi dengan menggunakan *script* Python yang tersedia dari peneliti FaceNet[27] dimana nanti akan dihasilkan *score accuracy*. *Accuracy* (Acc) adalah salah satu ukuran kinerja klasifikasi yang paling banyak digunakan, dan ini didefinisikan sebagai rasio sampel yang diklasifikasikan dengan benar terhadap jumlah total sampel (Tharwat, 2018). Untuk menghitung akurasi secara manual bisa memanfaatkan *confusion matrix* seperti terlihat pada Gambar 3.11. Selanjutnya untuk rumus perhitungan akurasi bisa menggunakan rumus berikut :

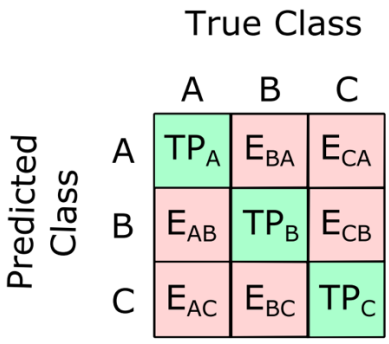
$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

TP = *True Positive*, jumlah data positif yang terklasifiksi dengan benar.

TN = *True Negative*, jumlah data negatif yang terklasifikasi dengan benar.

FP = *False Positive*, jumlah data negatif namun terklasifiksi salah.

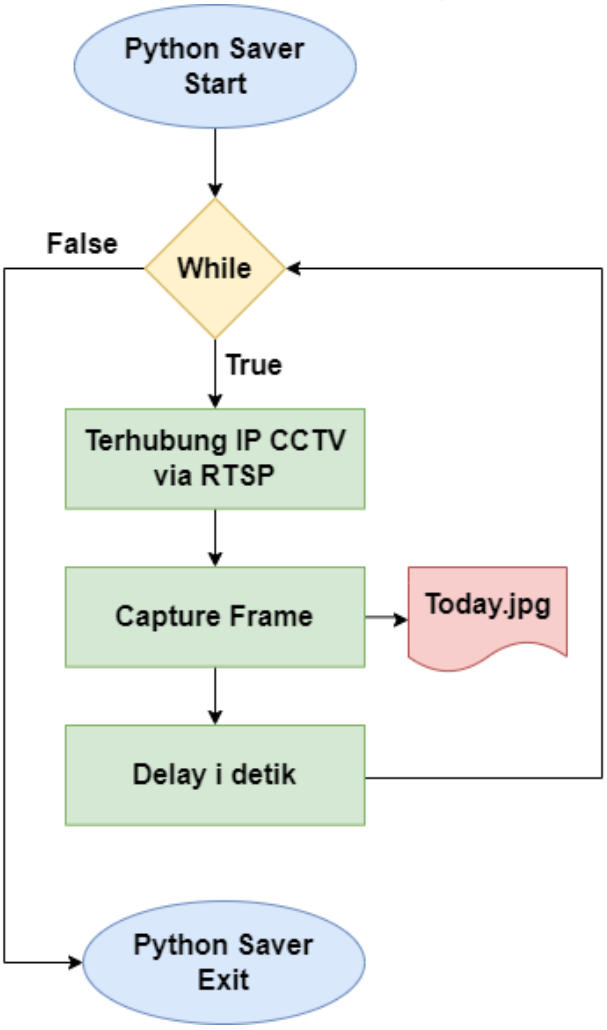
FN = *False Negative*, jumlah data positif namun terklasifikasi salah.



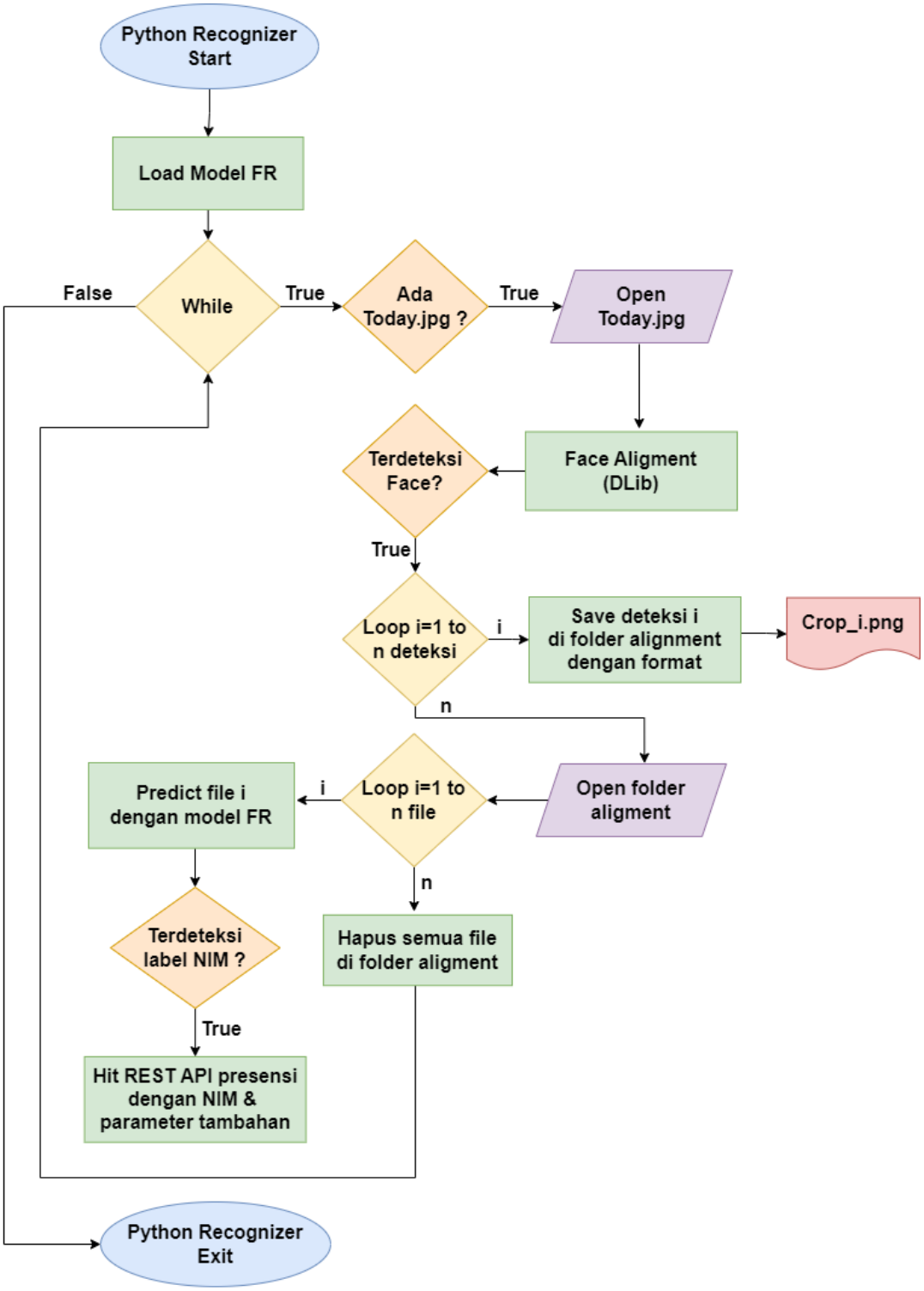
Gambar 3.11 Contoh ilustratif dari confusion matrix untuk tes klasifikasi multi-kelas.

3.7 Implementasi Model Pengenalan Wajah, Model Senyuman

Untuk pengenalan wajah mahasiswa melalui IP CCTV, peneliti membuat *pipeline* skrip Python yang mengakses IP CCTV melalui OpenCV dengan protokol RTSP. Foto diambil dalam hitungan detik dengan jeda. Setiap *frame* diambil lalu dideteksi menggunakan konsep *face-alignment* untuk deteksi wajah. Dalam pengenalan wajah ini dibutuhkan model FaceNet beserta model yang telah dilatih sebelumnya. Gambar 3.13 dan Gambar 3.13 menunjukkan bagaimana *pipeline* Python bekerja dalam mengimplementasikan model *deep learning*.



Gambar 3.12 Flowchart proses Python Saver.



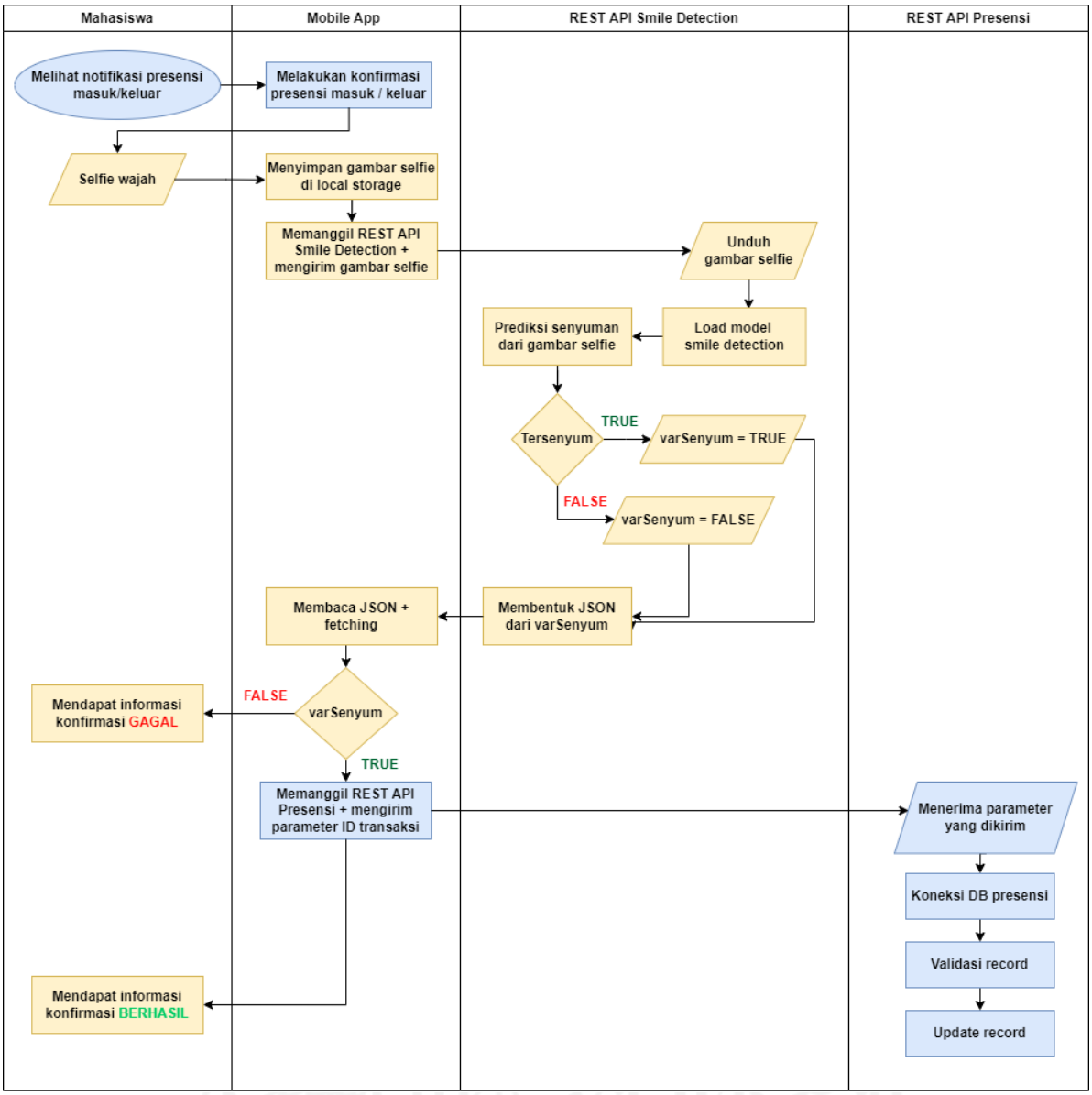
Gambar 3.13. Flowchart proses Python Recognizer.

Adapun urutan proses dari *pipeline* Python tersebut adalah sebagai berikut

1. RTSP IP CCTV adalah *stream* IP CCTV yang berjalan terus selama jam perkuliahan berlangsung.

2. Python Saver (usmiley_saver.py) dan Python Recognizer (usmiley_recognizer.py) dijalankan di *mini computer*, kedua *script* tersebut akan berjalan terus menerus selama tidak distop dan berperan sebagai *listener* (menangkap kejadian). Python Saver untuk pertama kali membaca RTSP IP CCTV lalu menjalankan *looping function*, sedangkan Python Recognizer pertama kali akan me-*load* model *deep learning* lalu *standby* untuk menunggu event selanjutnya.
3. Python Saver menangkap *frame* dari IP CCTV setiap detik (dengan jeda) lalu menyimpan *frame* tersebut menjadi sebut file dengan nama Today.jpg. Today.jpg hanya akan dibuat selama file tersebut tidak ada.
4. Python Recognizer sudah menemukan Today.jpg lalu dibaca dan diolah menggunakan *function* untuk *face alignment*. Apabila ditemukan adanya wajah dalam Today.jpg selanjutnya tiap ekstraksi wajah tadi akan disimpan menjadi Crop_1.jpg sampai dengan Crop_n.jpg (sejumlah wajah), dan masuk ke pemrosesan untuk *face recognition*. Setiap file *image* dengan prefix Crop_ akan dibaca lalu dimasukan ke model klasifikasi *deep learning* sehingga ditemukan identitas berupa NIM. NIM yang sudah dikenali tersebut selanjutnya menjadi parameter untuk proses pencatatan. Python Recognizer melalui *hit* atau pemanggilan API presensi dengan parameter NIM dan beberapa parameter yang sudah ditetapkan di script. Apabila proses berhasil selanjutnya file dengan ekstensi Crop_ akan dihapus sampai semua tidak tersisa.
5. Jika sudah tidak ada file dengan ekstensi Crop_ maka Today.jpg juga akan dihapus, dan proses dimulai lagi dari nomor 3.

Gambar 3.14 menunjukkan konsep dan bagaimana *smile detection* diimplementasikan dengan aktor utama mahasiswa dengan menggunakan aplikasi mobile. Aplikasi mobile terhubung dengan 2 REST API yaitu REST API Smile Detection dan REST API Presensi.



Gambar 3.14 Konsep smile detection.

Adapun urutan proses dari smile detection adalah sebagai sebagai berikut:

1. Mahasiswa melakukan *selfie* di mobile app sebagai konfirmasi dari presensi yang telah ia lakukan (masuk / keluar). Foto *selfie* tersebut oleh mobile app akan dikirim melalui REST API ke *service* REST API Smile Detection dengan parameter tambahan NIM mahasiswa, dan ID transaksi presensi.

2. REST API Smile Detection bekerja jika mendapatkan request berupa pengiriman file gambar (*selfie*), dan parameter lain. Dilakukan proses validasi inputan disini agar yang dimasukkan adalah parameter yang valid.
3. REST Smile Detection selanjutnya membaca *image* yang dikirimkan tadi, selanjutnya dengan menggunakan model smile detection yang sudah tersedia dicek apakah ini berupa senyuman atau tidak.
4. Jika berupa senyuman selanjutnya dilakukan update data melalui API presensi jika transaksi ID dari mahasiswa dengan NIM ini ditandai dengan senyuman.

3.8 Simulasi dan Evaluasi Sistem

Dalam melakukan simulasi sistem ini dapat dilakukan dengan 2 cara yaitu mensimulasikan di laboratorium khusus di kampus mengikuti arsitektur yang telah dirancang seperti Gambar 3.3 tentunya atas izin dari kampus, dan cara lain yaitu membuat simulasi sendiri di luar kampus sesuai dengan arsitektur yang dirancang juga.

BAB 4
Hasil dan Pembahasan

4.1 Pengumpulan Data Citra Mahasiswa

Kegiatan pengumpulan citra atau foto mahasiswa menjadi suatu *dataset* dimulai dari peneliti mendatangi mahasiswa yang akan dimintai pengambilan citra atau foto, selanjutnya melalui *script* Python yang telah dibuat dan dijalankan ini mahasiswa diminta untuk menggerakkan kepala seperti menghadap ke kanan, kiri, agak ke atas, dan agak ke bawah. Tujuan dari menggerakkan kepala ini agar pengambilan citra atau foto ini mendapatkan variasi pose wajah yang cukup beragam. Selama beberapa saat, semisal selama 2 menit *script* Python ini akan menangkap sebanyak 120 kali foto karena setiap detik akan diambil 1 kali foto.

Pada Gambar 4.1 diperlihatkan pada baris kode 8,9,10,13,dan 14 adalah variabel-variabel yang digunakan dalam *script*; **rstp_url** adalah URL *stream* dari IP CCTV yang digunakan untuk menangkap citra; **path** adalah lokasi tempat menyimpan file; **camera_mode** dapat diisi "*webcam*" jika akan menggunakan webcam dari komputer atau laptop, dan dapat diisi "*ipcam*" jika akan menggunakan **rstp_url; participant_name** dapat diisi dengan nama singkat mahasiswa.

usmiley_training_collector.py

```
1  import cv2
2  import numpy as np
3  import time
4  import os
5  import preprocess
6  from datetime import date
7
8  rtsp_url = "rtsp://admin:admin@192.168.0.100:554/live/0/MAIN"
9  path = '/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi' #folder path to save burst images
10 camera_mode = "webcam" # webcam | ipcam
11
12 preprocessor = preprocess.PreProcessor()
13 today = date.today()
14 participant_name = "miftahhh"
```

Gambar 4.1 Potongan script Python untuk pengambilan citra part 1.

Dalam script Python ini dibuatlah beberapa *function* terlihat pada Gambar 4.2 untuk mempermudah pemanggilan beberapa *method* dari OpenCV yaitu **rescale_frame()** dimulai baris kode 16 untuk mengubah ukuran *frame* sesuai prosentase yang diinginkan, dan

cam_capture() dimulai dari baris kode 26 yang digunakan untuk menyimpan *image* dari *frame* yang ditentukan dengan format filename tertentu.

```
16  def rescale_frame(frame, percent=75):
17      try:
18          width = int(frame.shape[1] * percent/ 100)
19          height = int(frame.shape[0] * percent/ 100)
20          dim = (width, height)
21
22          return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
23      except AttributeError as e:
24          print("Error rescale_frame = ", e)
25
26  def cam_capture(image):
27      try:
28          if (os.path.isfile("capture.txt")):
29              timestamp = int(time.time()*10030.0)
30              filename = participant_name + "_" + str(today) + "_" + str(timestamp) + ".jpg"
31
32              cv2.imwrite(filename,image)
33              bb = (preprocessor.align(filename))
34
35          if (camera_mode == "ipcam"):
36              cv2.imshow("frame40", image)
37          elif (camera_mode == "webcam"):
38              cv2.imshow("frame80", image)
39
40      except cv2.error as e:
41          # handle error: empty frame
42          if e.err == "!_src.empty()":
43              print("Error cam_capture = ", e)
```

Gambar 4.2 Potongan script Python untuk pengambilan citra part 2.

Selanjutnya pada Gambar 4.3 dimulai dari baris kode 45 dideklarasikan variabel **key** yang akan menangkap apa yang di-*input* melalui *keyboard*. Jika menekan huruf “s”, *script* Python akan mengarahkan pada blok kode 67-79 yaitu menyimpan *frame* yang telah diinisiasi pada baris kode 62 dengan format filename yang telah ditentukan secara manual. Jika menekan huruf “q”, *script* Python akan mengarahkan pada blok kode 81-84 yaitu keluar dari *capture* webcam atau IP CCTV lalu *script* akan terhenti. Selama belum ditekan huruf “q” maka *script* Python akan berjalan terus untuk menyimpan citra setiap detik secara otomatis.

```
key = cv2.waitKey(1)

try:
    if (camera_mode == "ipcam"):
        cap = cv2.VideoCapture(rtsp_url)
    elif (camera_mode == "webcam"):
        cap = cv2.VideoCapture(0)

    time.sleep(1)

except cv2.error as e:
    print("Error cv2.VideoCapture(rtsp_url) = ", e)
```

```
i = 0;
start = time.time()
while True:
    try:
        ret, frame = cap.read()

        key = cv2.waitKey(1)
        img_counter = 0

        if key & 0xFF == ord('s'): #press s to take images
            timestamp = int(time.time()*10030.0)
            filename = participant_name + "_manual_" + str(img_counter) + "_" + str(today) + "_" + str(timestamp) + ".jpg"

            if (camera_mode == "ipcam"):
                cv2.imwrite(os.path.join(path, filename), img=frame)
            elif (camera_mode == "webcam"):
                cv2.imwrite(os.path.join(path, filename), img=frame)

            img_ = cv2.imread(filename, cv2.IMREAD_ANYCOLOR) # save as RGB color format, size besar
            print("{} written!".format(filename))
            img_counter += 1
            time.sleep(0.2)

        elif key == ord('q'): #press q to quit without taking images
            cap.release()
            cv2.destroyAllWindows()
            break
```

```
        if (camera_mode == "ipcam"):
            frame40 = rescale_frame(frame, percent=40)
            cam_capture(frame40)
        elif (camera_mode == "webcam"):
            frame80 = rescale_frame(frame, percent=80)
            cam_capture(frame80)

    except(KeyboardInterrupt):
        print("Turning off camera.")
        cap.release()
        print("Camera off.")
        print("Program ended.")
        cv2.destroyAllWindows()
        break

end_time = time.time()
print(i/(end_time-start))
```

Gambar 4.3 Potongan script Python untuk pengambilan citra part 3.

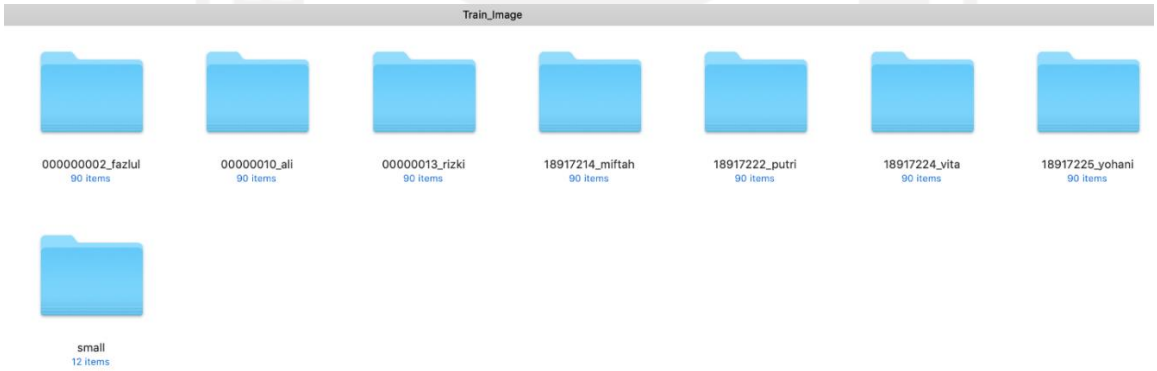
Hasil dari pengambilan foto menggunakan *script* Python akan menghasilkan file-file dengan nama yang disertai tanggal dan *timestamp* saat diambil. Apabila file-file tersebut akan dirapikan atau diurutkan berdasarkan nomor, maka dapat menggunakan *script* pada Gambar 4.4. Script ini adalah tambahan jika *dataset* yang akan dibentuk itu adalah gabungan dari beberapa kali pengambilan dan akan direname ulang.

usmiley_file_renamer.py

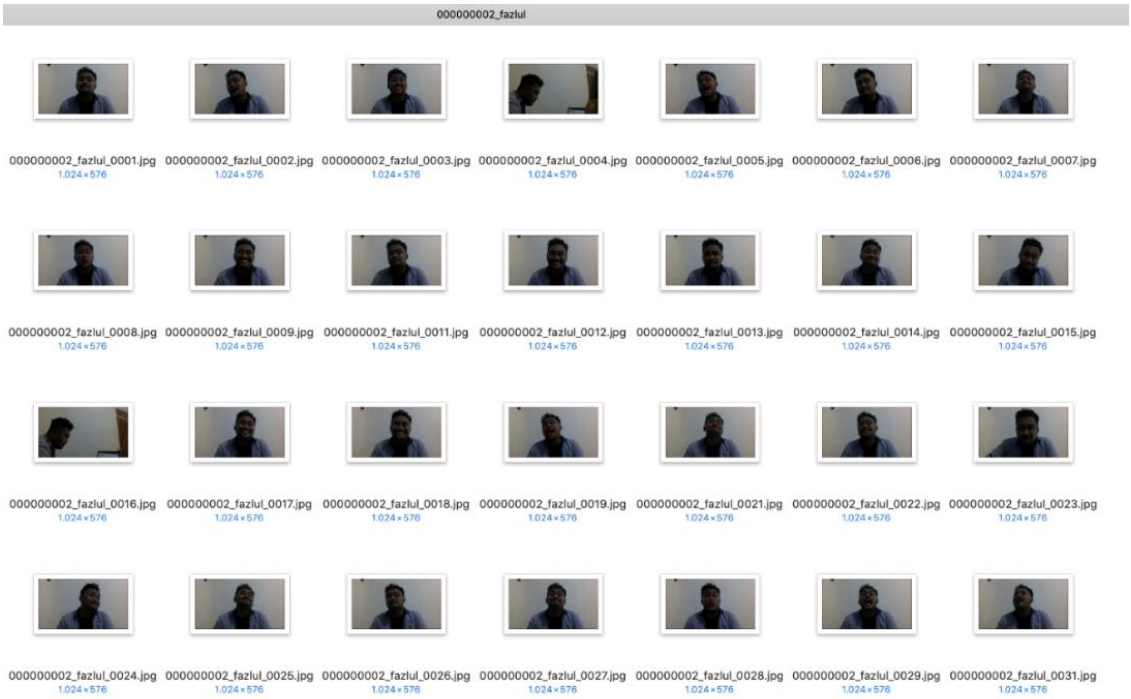
```python
import glob
import os

img_files = []
imf_files_new = []
source_dir = "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/alignment/18917225_yohani"
prefix_name = "18917225_yohani_"
i = 1
for filename in glob.glob(os.path.join(source_dir, '*.*')):
    img_files.append(os.path.basename(filename))
    filename_new = source_dir + "/" + prefix_name + str(i).zfill(4) + ".jpg"
    os.rename(filename, filename_new)
    i += 1
print(i)
```

Gambar 4.4 Script Python untuk rename file dengan nomor urut.

Pada penelitian ini *dataset* dibentuk dalam sebuah folder dengan nama **Train_Image**. Di dalamnya terdapat beberapa folder dengan format **NIK_NamaSingkat,** nama folder ini juga akan menjadi label dari klasifikasi yang akan dibuat nanti. Untuk melihat isi dari folder **Train_Image** dapat dilihat pada Gambar 4.5, secara detail isi didalamnya dapat dilihat pada Gambar 4.6 dari salah satu mahaasiswa.



Gambar 4.5 Struktur root directory tiap mahasiswa.



Gambar 4.6 Isi directory salah satu mahasiswa.

4.2 Training Dataset Mahasiswa

Setelah tahapan pengumpulan data foto atau citra mahasiswa, tahap selanjutnya adalah membentuk *dataset*. Jumlah foto yang diambil dari tiap mahasiswa pada penelitian dibatasi maksimal 100 foto dengan rencana pembagian 75% digunakan untuk *data training*, 15% digunakan untuk validasi, dan 10% digunakan untuk *testing* seperti terlihat pada Tabel 4.1 Pembagian Dataset. Adapun besaran % *dataset* yang digunakan untuk *training* mengikuti beberapa referensi dimulai dari 50-90%.

Tabel 4.1 Pembagian Dataset

No	Mahasiswa	NIM / Class	Training	Validation	Testing
1	Fazlul	000000002	75 foto	15 foto	10 foto
2	Ali	00000010	75 foto	15 foto	10 foto
3	Rizki	00000013	75 foto	15 foto	10 foto
4	Miftah	18917214	75 foto	15 foto	10 foto
5	Putri	18917222	75 foto	15 foto	10 foto
6	Vita	18917224	75 foto	15 foto	10 foto
7	Yohani	18917225	75 foto	15 foto	10 foto
Total			**526 foto**	**105 foto**	**70 foto**

Sebelum melakukan proses *training*, perlu disiapkan komputer atau laptop yang digunakan. Pada penelitian ini laptop yang digunakan menggunakan spesifikasi seperti pada Tabel 4.2, versi Python yang digunakan adalah 3.6. Selain Python diperlukan juga beberapa *package* tambahan yang diinstall menggunakan PIP terlihat pada Tabel 4.3 Requirement Package Python Yang Diinstall. Beberapa *package* yang dianggap penting ditandai dengan tanda *.

Tabel 4.2 Spesifikasi Laptop Yang Digunakan Untuk Training Dataset

No	**Komponen**	**Spesifikasi**
1	Processor	2,4 GHz Intel Core i7
2	Memory	8 GB 1600 MHz DDR3
3	Storage	SSD 250GB
4	Sistem Operasi	macOS Mojave
5	Graphics	Intel HD Graphics 4000 NVIDIA GeForce GT 650M

Tabel 4.3 Requirement Package Python Yang Diinstall

No	***Package***	***Version***	***Primary***
1	absl-py	0.10.0	
2	astor	0.8.1	
3	astunparse	1.6.3	
4	bleach	1.5.0	
5	cachetools	4.1.1	
6	certifi	2020.6.20	
7	chardet	3.0.4	
8	cycler	0.10.0	
9	flatbuffers	1.12	
10	gast	0.3.3	
11	google-auth	1.21.3	
12	google-auth-oauthlib	0.4.1	
13	google-pasta	0.2.0	
14	grpcio	1.32.0	*
15	h5py	2.10.0	*
16	html5lib	0.9999999	
17	idna	2.10	
18	imageio	2.9.0	*
19	importlib-metadata	2.0.0	
20	joblib	0.16.0	
21	Keras-Preprocessing	1.1.2	*

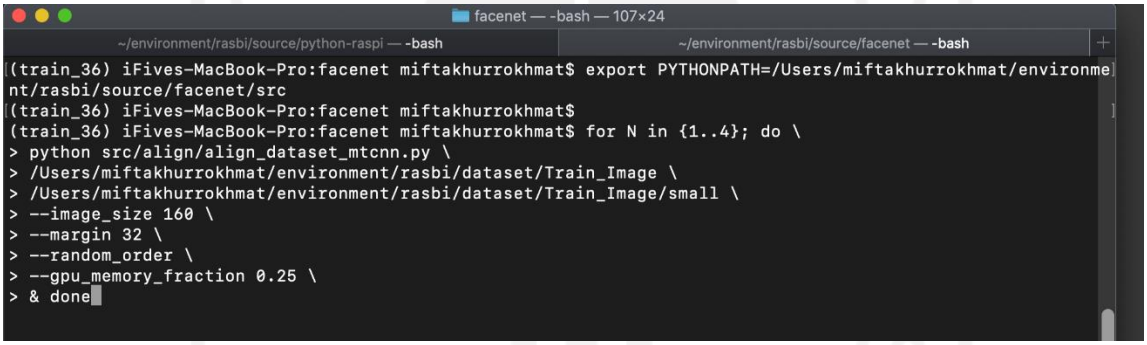
22	kiwisolver	1.2.0	
23	Markdown	3.2.2	
24	matplotlib	3.3.2	
25	numpy	1.16.0	*
26	oauthlib	3.1.0	
27	opencv-contrib-python	4.4.0.44	*
28	opt-einsum	3.3.0	
29	Pillow	7.2.0	*
30	protobuf	3.13.0	*
31	psutil	5.7.2	
32	pyasn1	0.4.8	
33	pyasn1-modules	0.2.8	
34	pyparsing	2.4.7	
35	python-dateutil	2.8.1	
36	requests	2.24.0	
37	requests-oauthlib	1.3.0	
38	rsa	4.6	
39	scikit-learn	0.23.2	*
40	scipy	1.2.2	*
41	six	1.15.0	
42	tensorboard	1.7.0	
43	tensorboard-plugin-wit	1.7.0	
44	tensorflow	1.7.0	*
45	termcolor	1.1.0	
46	tflite-runtime @ https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-macosx_10_14_x86_64.whl		
47	threadpoolctl	2.1.0	
48	typing-extensions	3.7.4.3	
49	urllib3	1.25.10	
50	Werkzeug	1.0.1	
51	wrapt	1.12.1	
52	zipp	3.2.0	

Langkah selanjutnya setelah melakukan instalasi *requirement* atau beberapa *package* yang dibutuhkan yaitu menambahkan *environment* ke dalam profil akun di Sistem Operasi terlihat pada Gambar 4.7. Hal ini dilakukan untuk memudahkan dalam mengakses *environment* di kemudian hari dan agar tidak salah lokasi.

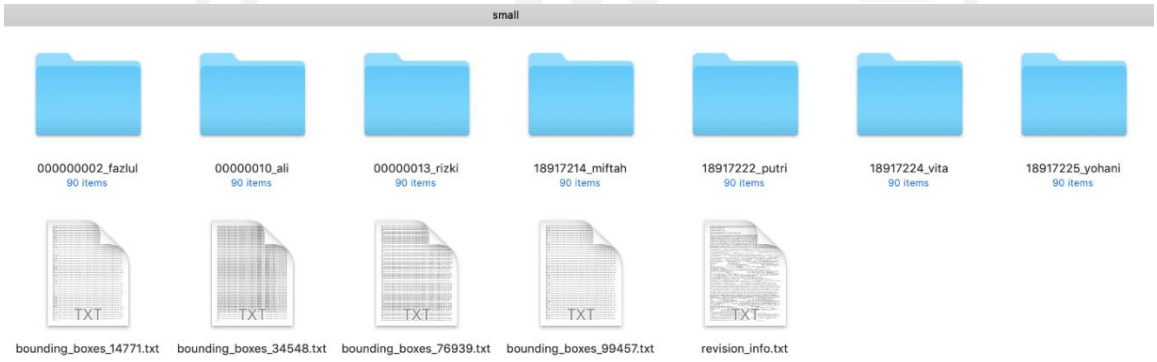
```
export THESIS_ENV=/Users/miftakhurrokhmat/environment/raspi/bin/activate
export THESIS_DATASET=/Users/miftakhurrokhmat/environment/rasbi/dataset/Train_Image
export THESIS_SOURCE=/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi

export TRAIN_ENV=/Users/miftakhurrokhmat/environment/train_36/bin/activate
export TRAIN_SOURCE=/Users/miftakhurrokhmat/environment/rasbi/source/facenet
```

Gambar 4.7 Environtment tambahan untuk Python.

Hal lain yang dilakukan sebelum *training* yaitu menstandarisasi ukuran dimensi foto dan menseleksi area wajah saja. Pada penelitian ini dilakukan proses *face-aligment* yaitu mengambil area wajah saja dengan bantuan *script* Python yang tersedia[28]. Proses *face-alignment* ini menggunakan arsitektur turunan dari CNN yaitu MTCNN sehingga mampu mengenali area wajah saja. Saat *script* Python ini dijalankan seperti pada Gambar 4.8 ini akan menghasilkan foto-foto baru dengan dimensi yang lebih kecil yaitu 160 x 160 dan dengan fokus di area wajah terlihat pada Gambar 4.10 dengan struktur folder masih sama seperti *dataset* semula terlihat pada Gambar 4.9.

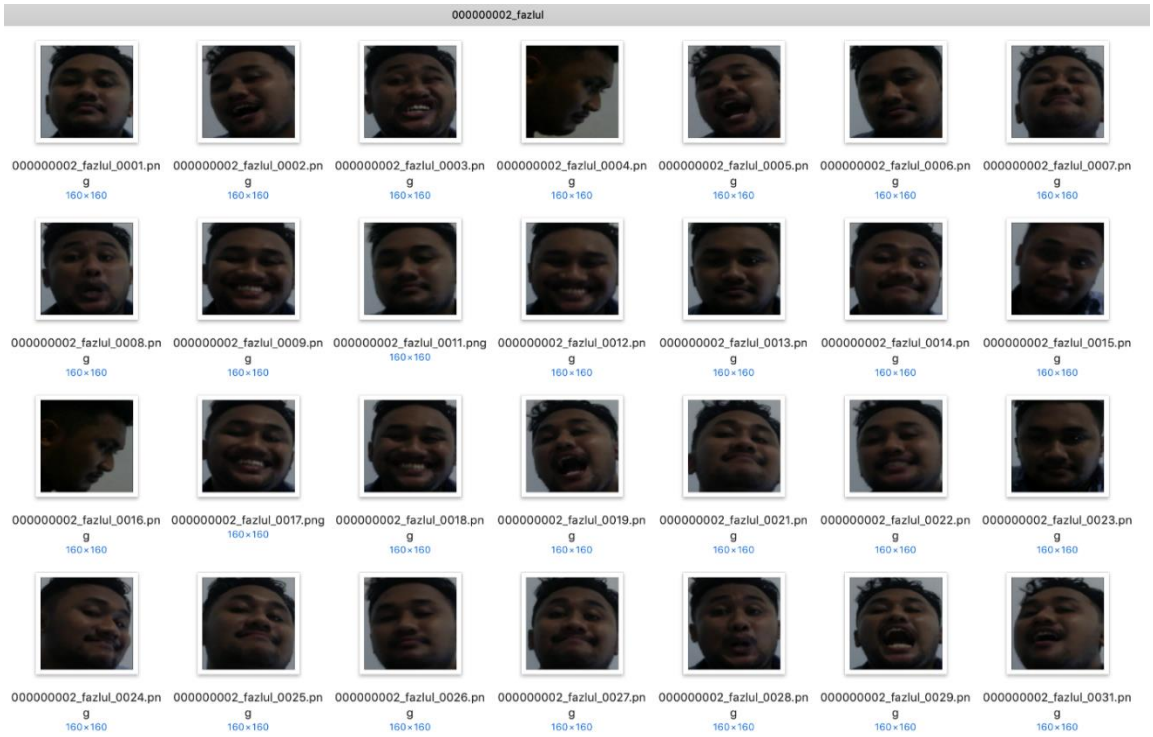


Gambar 4.8 Command untuk menjalankan alignment di dataset.



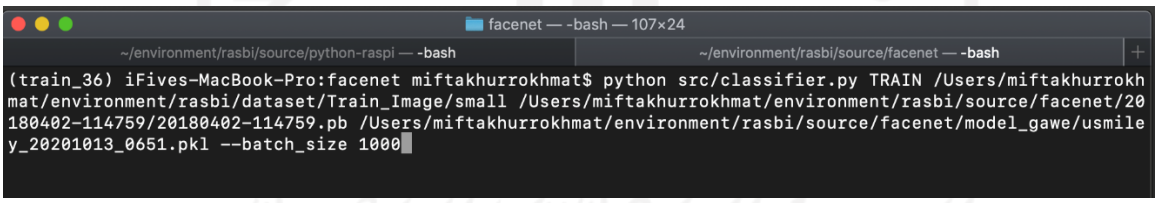
Gambar 4.9 Struktur root directory tiap mahasiswa setelah alignment.

[28] https://github.com/davidsandberg/facenet/blob/master/src/align/align_dataset_mtcnn.py



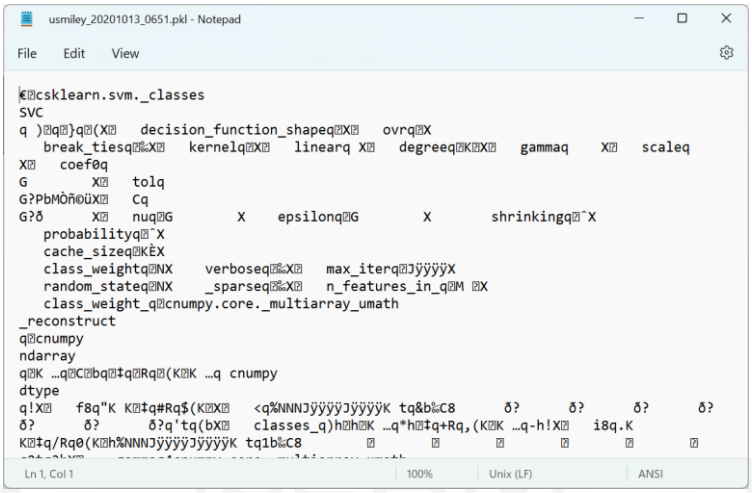
Gambar 4.10 Isi directory salah satu mahasiswa setelah alignment.

Pada langkah selanjutnya dilakukan *training* menggunakan *skrip* Python yang tersedia[29] terlihat pada Gambar 4.11. Hasil akhir dari proses *training* ini menghasilkan sebuah model berekstensi *pickle* (.pkl) sebesar 649 KB dan saat dibuka isinya dengan text editor akan terlihat seperti Gambar 4.12.



Gambar 4.11 Command untuk menjalankan training dataset.

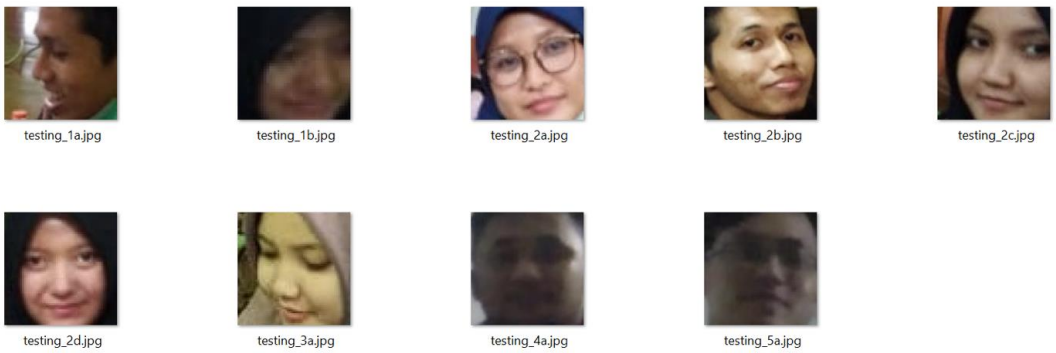
[29] https://github.com/davidsandberg/facenet/blob/master/src/classifier.py



Gambar 4.12 Bedah isi model hasil training.

4.3 Script Pengujian Model

Script yang terlihat pada Gambar 4.14, Gambar 4.15, Gambar 4.16 adalah keseluruhan *script* dari modifikasi atau pengembangan *script* yang ada di peneliti FaceNet[30]. Penjelasan dari Gambar 4.14 adalah sebagai berikut : Baris kode 1-7 adalah *import library* yang diperlukan seperti tensorflow, numpy, math untuk perhitungan matematis, facenet yang berisi *function* terkait pemrosesan di FaceNet, os untuk pembacaan file, dan pickle untuk menyimpan model. Langkah awal yang dilakukan adalah deklarasi tensorflow graph dengan *session* pada baris kode 9-10. Selanjutnya disiapkan 2 variabel yaitu **path_2** yang berisi lokasi gambar yang akan digunakan untuk *testing* (gambar yang digunakan terlihat Gambar 4.13), **labels** yang berisi *class* atau label dari gambar sesungguhnya (NIM mahasiswa), 2 variabel ini akan digunakan untuk pencocokan. Selanjutnya dilakukan *load* model FaceNet pada baris 28-29.



Gambar 4.13 Foto testing dari tangkapan layar.

[30] https://github.com/davidsandberg/facenet/blob/master/src/classifier.py

usmiley_image_classify.py

```
1  import tensorflow as tf
2  import numpy as np
3  import facenet
4  import os
5  import math
6  import pickle
7  import sys
8
9  with tf.Graph().as_default():
10     with tf.Session() as sess:
11         np.random.seed(seed=666)
12
13         paths_2 = [
14         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_1a.jpg",
15         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_1b.jpg",
16         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_2a.jpg",
17         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_2b.jpg",
18         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_2c.jpg",
19         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_2d.jpg",
20         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_3a.jpg",
21         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_4a.jpg",
22         "/Users/miftakhurrokhmat/environment/rasbi/dataset/testing_foto/baru/testing_5a.jpg"]
23
24         labels = [2,4,5,2,6,4,6,0,3]
25
26         # Load the model
27         print('Loading feature extraction model')
28         args_model="/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi/model/20180402-114759.pb"
29         facenet.load_model(args_model)
```

Gambar 4.14 Potongan script Python untuk pengujian part 1.

Penjelasan dari Gambar 4.15 adalah sebagai berikut : Setelah load model dari FaceNet, selanjutnya adalah pembacaan variable di tensorflow di baris kode 32-35, lalu disiapkan juga variabel-variabel yang digunakan di proses selanjutnya di baris kode 39-42.

```
31         # Get input and output tensors
32         images_placeholder = tf.get_default_graph().get_tensor_by_name("input:0")
33         embeddings = tf.get_default_graph().get_tensor_by_name("embeddings:0")
34         phase_train_placeholder = tf.get_default_graph().get_tensor_by_name("phase_train:0")
35         embedding_size = embeddings.get_shape()[1]
36
37         # Run forward pass to calculate embeddings
38         print('Calculating features for images')
39         nrof_images = len(paths_2)
40         args_batch_size=1000
41         nrof_batches_per_epoch = int(math.ceil(1.0*nrof_images / args_batch_size)) # int(ceil(1*23/1000))
42         emb_array = np.zeros((nrof_images, embedding_size))
```

Gambar 4.15 Potongan script Python untuk pengujian part 2.

Selanjutnya pada Gambar 4.16 dimulai dari baris kode 44 dilakukan *looping batch* dari variabel sebelumnya **nrof_batches_per_epoch**. Baris kode 45-51 memproses sehingga dihasilkan suatu variabel **emb_array** yang berisi *embedding* dari file-file gambar yang akan

dijadikan *testing* tadi. Lalu baris kode 55-56 akan membaca model hasil *training* dengan ekstensi .pkl untuk untuk ditampung dalam variabel **classifier_filename_exp**. Variabel ini dibaca dengan menggunakan *library* pickle lalu dijadikan 2 variabel yaitu **model** dan **class_name**. Selanjutnya **model** dapat menggunakan beberapa *function* untuk prediksi yaitu **predict_proba()** dengan memasukkan *embedding* **emb_array** yang sebelumnya telah diolah. Hasil akhirnya adalah dapat dihitung **accuracy** berdasarkan nilai yang didapat dari **best_class_indices** dicocokkan dengan **labels**.

```
44      for i in range(nrof_batches_per_epoch):
45          start_index = i*args_batch_size
46          end_index = min((i+1)*args_batch_size, nrof_images)
47          paths_batch = paths_2
48          args_image_size = 160
49          # image_paths, do_random_crop, do_random_flip, image_size, do_prewhiten=True
50          images = facenet.load_data(paths_batch, False, False, args_image_size)
51          feed_dict = { images_placeholder:images, phase_train_placeholder:False }
52
53          emb_array[start_index:end_index,:] = sess.run(embeddings, feed_dict=feed_dict)
54
55          args_classifier_filename="/Users/miftakhurrokhmat/environment/rasbi/source/facenet/model_gawe/usmiley_20201013_0651.pkl"
56          classifier_filename_exp = os.path.expanduser(args_classifier_filename)
57
58          # Classify images
59          print('Testing classifier')
60          with open(classifier_filename_exp, 'rb') as infile:
61              (model, class_names) = pickle.load(infile)
62
63          print('Loaded classifier model from file "%s"' % classifier_filename_exp)
64
65          predictions = model.predict_proba(emb_array)
66          best_class_indices = np.argmax(predictions, axis=1)
67          best_class_probabilities = predictions[np.arange(len(best_class_indices)), best_class_indices]
68
69          for i in range(len(best_class_indices)):
70              print('%4d  %s: %.3f %s' % (i, class_names[best_class_indices[i]], best_class_probabilities[i], paths_batch[i]))
71
72          accuracy = np.mean(np.equal(best_class_indices, labels))
73          print('Accuracy: %.3f' % accuracy)
```

Gambar 4.16 Potongan script Python untuk pengujian part 3.

4.4 Script Implementasi Service Saver

Dalam implementasi sistem pengenalan pola wajah ini perlu dilakukan pemasangan 2 *service* atau menjalankan 2 *script* Python di *mini computer* yang terhubung dengan IP CCTV. Adapun *service-service* ini disebut dengan "**Service Saver**" dan "**Service Recognizer**". Pada Gambar 4.17 terlihat potongan *script* "**Service Saver**" yang hampir mirip dengan Gambar 4.1 dan Gambar 4.2.

usmiley_image_process_save.py

```python
import cv2
import numpy as np
import time
import os
from datetime import date
from datetime import datetime
import gc

rtsp_url = "rtsp://admin:admin@192.168.0.100:554/live/0/MAIN"
path = '/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi' #folder path to save burst images
camera_mode = "ipcam" # webcam | ipcam
is_show = True

def rescale_frame(frame, percent=75):
    try:
        width = int(frame.shape[1] * percent/ 100)
        height = int(frame.shape[0] * percent/ 100)
        dim = (width, height)

        return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
    except AttributeError as e:
        print("Error rescale_frame = ", e)
```

Gambar 4.17 Potongan script Python service saver part 1.

Perbedaannya terletak pada Gambar 4.18 blok kode 27-43. Terdapat pengecekan file "capture.txt", jika file ini ada maka *script* Python akan menjalankan *capture image* setiap detik. Capture.txt adalah representasi dari waktu berlangsungnya perkuliahan di kelas atau ruangan yang ada IP CCTV. Apabila kelas telah usai maka "capture.txt" akan dihapus agar tidak mengambil *capture* secara terus menerus. Selain itu dibuat juga *function* **open_cam()** terlihat pada baris kode 46 dan **close_cam()** terlihat pada baris kode 58 dengan tujuan untuk menghemat *resource* (CPU / *memory*) saat mengakses IP CCTV. Selain itu untuk membersihkan *memory* yang sudah tidak dipakai di *mini computer* dengan melakuan pemanggilan method di Python yaitu **gc.collect()** terlihat pada Gambar 4.19 di baris kode 74, **gc.collect()** dilakukan jika jumlah perulangan mencapai hitungan maksimal atau **MAX_LOOP** dan counter (**CURRENT_LOOP**) akan diset ulang menjadi 0 kembali.

```python
def cam_capture(image):
    try:

        if (os.path.isfile("capture.txt")):
            today = date.today()
            file_today = str(today) + ".jpg"
            cv2.imwrite(path + "/process/" + file_today,image) # for process
            time.sleep(0.2)

        if (camera_mode == "ipcam"):
            if (is_show == True):
                cv2.imshow("frame40", image)
        elif (camera_mode == "webcam"):
            if (is_show == True):
                cv2.imshow("frame80", image)

    except cv2.error as e:
        # handle error: empty frame
        if e.err == "!_src.empty()":
            print("Error cam_capture = ", e)

def open_cam():
    try:
        if (camera_mode == "ipcam"):
            cap = cv2.VideoCapture(rtsp_url)
        elif (camera_mode == "webcam"):
            cap = cv2.VideoCapture(0)

        return cap

    except cv2.error as e:
        print("Error cv2.VideoCapture(rtsp_url) = ", e)

def close_cam(cap):
    cap.release()
```

Gambar 4.18 Potongan script Python service saver part 2.

```python
# ----------------------- MAIN ----------------------- #

key = cv2.waitKey(1)
cap = open_cam()

i = 0;
start = time.time()

MAX_LOOP = 10
CURRENT_LOOP = 0
while True:
    if (CURRENT_LOOP == MAX_LOOP):
        gc.collect()

    if not cap.isOpened():
        print('Unable to load camera.')
        time.sleep(0.2)
        close_cam(cap)
        cap = open_cam()
```

Gambar 4.19 Potongan script Python service saver part 3.

Pada Gambar 4.20 sebagian blok kode akhir dari *script* "**Service Saver**" ini secara umum hampir mirip dengan Gambar 4.3.

```
    try:
        ret, frame = cap.read()
        if ret:
            assert not isinstance(frame,type(None)), 'frame not found'

            key = cv2.waitKey(1)
            img_counter = 0

            if key & 0xFF == ord('s'): #press s to take images
                today_now = datetime.now().strftime("%d%m%Y %H%M%S")
                img_name = "manual_{}.png".format(today_now)

                if (camera_mode == "ipcam"):
                    cv2.imwrite(os.path.join(path, img_name), img=frame)
                elif (camera_mode == "webcam"):
                    cv2.imwrite(os.path.join(path, img_name), img=frame)

                img_ = cv2.imread(img_name, cv2.IMREAD_ANYCOLOR) # save as RGB color format, size besar
                print("{} written!".format(img_name))

            if key == ord('q'): #press q to quit without taking images
                cap.release()
                cv2.destroyAllWindows()
                break

            if (camera_mode == "ipcam"):
                frame40 = rescale_frame(frame, percent=40)
                cam_capture(frame40)
            elif (camera_mode == "webcam"):
                frame80 = rescale_frame(frame, percent=80)
                cam_capture(frame80)

            time.sleep(1)

        CURRENT_LOOP += 1

    except(KeyboardInterrupt):
        print("Turning off camera.")
        cap.release()
        print("Camera off.")
        print("Program ended.")
        cv2.destroyAllWindows()
        break

end_time = time.time()
print(i/(end_time-start))
```

Gambar 4.20 Potongan script Python service saver part 4.

4.5 Script Implementasi Service Recognizer

Service selain "**Service Saver**" yaitu "**Service Recognizer**", *service* ini secara umum akan melakukan pengenalan wajah terhadap *frame* yang sudah di-*capture* oleh "Service Saver". *Frame* yang sudah di-*capture* tadi disimpan dengan format "TanggalHariIni.jpg" dan dalam *frame* tersebut dapat terdapat lebih dari 1 *face* / wajah. Untuk melakukan ekstraksi ada berapa *face* / wajah dalam *frame image* tersebut diperlukan *library* Dlib. Dlib disini dapat melakukan ekstraksi wajah dari suatu gambar terlihat pada Gambar 4.21 di baris kode 21 yaitu Dlib membaca model deteksi wajah dengan *function* **cnn_face_detection_model_v1()** dengan sebelumnya di-*import* dulu di baris kode 9. Selain Dlib juga di-*import* beberapa *package* dimulai dari baris kode 1-10 diantaranya yaitu OpenCV, gc, glob, dan classify itu sendiri. Selain itu juga diperlukan deklarasi variabel seperti **raspberry_id** sebagai identitas *mini computer* di suatu kelas, **path_process** yang berisi 1 foto atau gambar hasil *capture* IP CCTV, dan **path_align** yang akan berisi foto-foto hasil ekstraksi wajah dari 1 gambar foto yang dapat terdiri dari banyak *face* / wajah.

usmiley_image_process_classify.py

```
1  import cv2
2  import numpy as np
3  import classify
4  import preprocess
5  from datetime import date
6  import time
7  import os
8  import gc
9  import dlib
10 import glob
11
12 raspberry_id = "RAPSI001"
13 path_process = '/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi/process' #folder path to save burst image
14 path_align = '/Users/miftakhurrokhmat/environment/rasbi/source/python-raspi/process/alignment'
15
16 # loading
17 today = date.today()
18 file_today = str(today) + ".jpg"
19 classifier = classify.Classify()
20
21 cnn_face_detector = dlib.cnn_face_detection_model_v1("mmod_human_face_detector.dat")
```

Gambar 4.21 Potongan script Python service recognizer 1.

Disediakan sebuah *function* **hit_api_presensi()** dengan parameter NIM hasil pengenalan wajah dari setiap *face* yang telah diekstrasi Dlib. Pada Gambar 4.22 di baris kode 25 terdapat URL sebagai alamat *web service* untuk melakuan pencatatan presensi, terdapat 2 parameter yang akan dikirim ke *web service* yaitu nim dan raspi_id, raspi_id disini sebagai representasi dari ruangan kelas.

```
23  def hit_api_presensi(nim):
24      try:
25          url = 'http://api-usmiley.yogyakarsa.com/raspi/public/presensi'
26          form_data = {'nim': nim, 'raspi_id': raspberry_id}
27          post = requests.post(url, data = form_data)
28          parse = json.loads(post.text)
29          print(parse['message'])
30      except requests.RequestException as e:
31          print(e)
```

Gambar 4.22 Potongan script Python service recognizer part 2.

Gambar 4.23 menunjukkan beberapa *function* yang digunakan dalam *script* Python ini yaitu **align_face()** di baris kode 33, **face_recognition()** di baris kode 59, dan **delete_alignment()** di baris kode 69.

Function **align_face()** berisi kode untuk membaca *image capture* (baris kode 35-37), lalu selanjutnya mengekstraksi pendeteksi wajah dengan method **cnn_face_detection()** di baris kode 38. Hasil dari ekstraksi ini akan ditampung sebagai *array*, selanjutnya *array* ini di-*looping* untuk disimpan menjadi file gambar (baris kode 43-57), file gambar ini nanti akan digunakan oleh *function* **face_recognition().**

Function **face_recognition()** berisi kode untuk membaca **path_align**, setiap *file image* yang dibaca akan dimasukkan ke *method* **classfier.predict()** untuk dilihat labelnya dan tingkat akurasinya. Label yang didapat berupa **NIM_NamaSingkat**. Selanjutnya dari **NIM_NamaSingkat** ini di-*explode* untuk mendapatkan NIM. Setelah NIM didapatkan selanjutnya dilakukan pemanggilan *function* **hit_api_presensi()** dengan parameter NIM sebagai *trigger* untuk mencatat presensi.

Function **delete_alignment()** berisi kode untuk menghapus semua file *alignment* dari *function* **align_face().** *Function* **delete_alignment()** ini hanya akan dijalankan setelah *function* **face_recognition()** telah dipanggil dengan maksud telah berhasil melakukan proses presensi.

```python
def align_face():
    # apply face detection (cnn)
    file_name = path_process + "/" + file_today
    image = cv2.imread(file_name)
    image_clean = cv2.imread(file_name)
    faces_cnn = cnn_face_detector(image, 1)

    # loop over detected faces
    idx_face = 1

    for face in faces_cnn:
        x = face.rect.left()
        y = face.rect.top()
        w = face.rect.right() - x
        h = face.rect.bottom() - y

        # ambil yang dideteksi
        slice = image_clean[y:y+h, x:x+w]
        # resize image
        resized = cv2.resize(slice, (160, 160), interpolation = cv2.INTER_AREA)

        file_saved = path_align + "/" + "align_" + str(today) + "_" + str(idx_face).zfill(3) + ".png"
        cv2.imwrite(file_saved, resized) # ekstensi save png

        idx_face += 1

def face_recognition():
    print("--- begin face_recognition ---")
    idx_recog = 1
    for filename in glob.glob(os.path.join(path_align, '*.png')):
        name, accuration = classifier.predict(filename)
        all_label = str(name) + " (" + str(accuration) + ")"
        print(idx_recog, all_label)
        idx_recog += 1
    print("--- end face_recognition ---")

def delete_alignment():
    print("delete alignment")
    for filename in glob.glob(os.path.join(path_align, '*.png')):
        if (os.path.isfile(filename)):
            os.remove(filename)
```

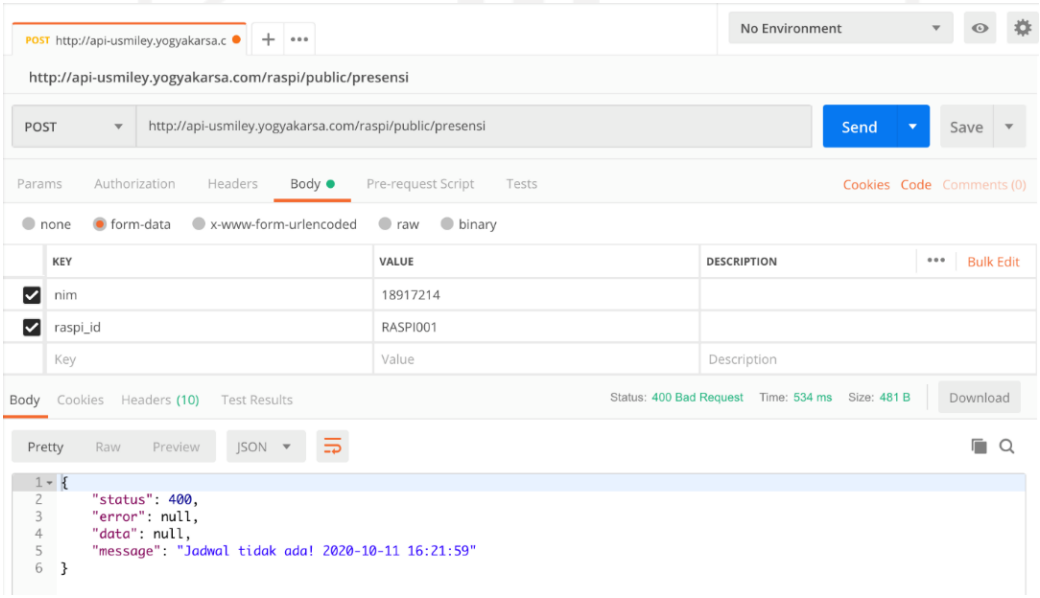
Gambar 4.23 Potongan script Python service recognizer part 3.

Gambar 4.24 merupakan blok akhir dari *script* “Service Recognizer”. Mulai dari baris kode 77-95 adalah serangkaian urutan proses yang berisi urutan-urutan pemanggilan *function* dan juga pembersihan *memory* saat *looping* sudah mencapai batas maksimal (**MAX_LOOP**).

```
75  # ----------------------- MAIN ----------------------- #
76
77  i = 0;
78  start = time.time()
79
80  MAX_LOOP = 10
81  CURRENT_LOOP = 0
82
83
84  while True:
85      if (CURRENT_LOOP == MAX_LOOP):
86          gc.collect()
87
88      align_face()
89      face_recognition()
90      delete_alignment()
91
92      if(CURRENT_LOOP == 1):
93          time.sleep(30)
94
95      CURRENT_LOOP += 1
```

Gambar 4.24 Potongan script Python service recognizer part 4.

Pada Gambar 4.22 di function **hit_api_presensi()** terdapat URL API di baris kode 25. Apabila disimulasikan menggunakan POSTMAN akan terlihat di Gambar 4.25. Terdapat 2 parameter yaitu **nim** dan **raspi_id**, keduanya harus benar dimasukkan, apabila ada salah satu yang tidak benar maka akan memunculkan pesan kesalahan karena ada pengecekan di *database*. Selain itu juga jika disimulasi presensi belum sesuai jadwal perkuliahan juga akan memunculkan pesan kesalahan.



Gambar 4.25 Hit API presensi melalui POSTMAN.

4.6 Script REST API Smile Detection

Implementasi pengenalan senyuman diterapkan pada aplikasi *mobile* untuk mahasiswa pada saat konfirmasi kehadiran untuk masuk kelas dan keluar kelas. Pada saat melakukan *selfie*, aplikasi mobile melakukan pemanggilan REST API Smile Detection yang di dalamnya dipasang model pengenalan senyuman. Pengenalan senyuman ini menggunakan model dari Haar Cascade Clasifier dengan nama file **haarcascade_smile.xml** yang di dalamnya berisi pola-pola (angka-angka) untuk mengenali senyuman. Model ini tidak dapat digunakan sendiri karena memerlukan model lain yaitu **haarcascade_frontalface_default.xml** sebagai deteksi awal untuk menangkap area wajah lalu dilanjutkan mendeteksi senyuman. Pada Gambar 4.26 dan Gambar 4.27 ditunjukkan baris-baris kode yang digunakan dalam penerapan model senyuman dengan menggunakan *script* Python di dalam *framework* Flask.

```python
import cv2
import os
import numpy as np

from flask import Blueprint, request
from flask_json import json_response
from datetime import datetime

uploadapi =  Blueprint('uploadapi', __name__, url_prefix='/api/v1/upload')
IMAGE_EXTENSIONS=[".png",".jpg","jpeg"]

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')

def isImage(ext):
    return ext in IMAGE_EXTENSIONS

def is_smile(gray, frame):
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    result={}
    if faces is None:
        return  False
    for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = frame[y:y + h, x:x + w]
        smiles = smile_cascade.detectMultiScale(roi_gray,1.2)
        print(smiles)
        if smiles is None:
            return False
        else:
            return True
    return False
```

Gambar 4.26 Potongan script Python REST API smile detection part 1.

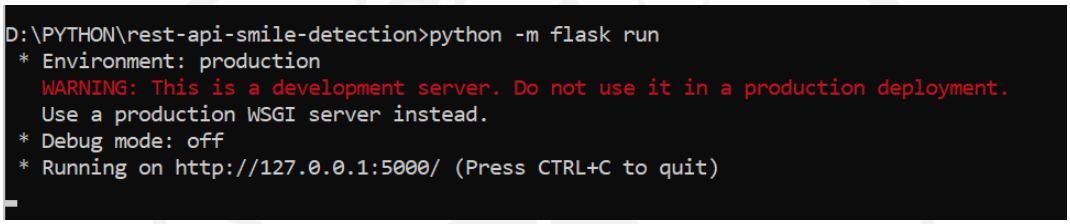
Pada baris kode 1-7 untuk pertama kali dilakukan *import* dari beberapa *package* yang diperlukan seperti **cv2** (openCV) yang nanti memangggil model Haar Cascade, **os** untuk beberapa operasi dengan file, *numpy* untuk mengambil string dari gambar menjadi object image, **flask** dan **flask_json** untuk inti dari *framework* Flask. Selanjutnya disiapkan beberapa variabel yang nanti diperlukan di baris kode 9-14 yaitu **upload_api** untuk *prefix endpoint*, **IMAGE_EXTENSIONS** untuk daftar ekstensi yang diijinkan, serta variabel untuk lokasi file model xml dari Haar Cascade yaitu **face_cascade**, **eye_cascade**, dan **smile_cascade**. Beberapa *function* penunjang disiapkan yaitu **isImage()** dimulai di baris kode 16 dan **is_smile()** yang didalamnya menggunakan model **haarcascade_smile.xml** dimulai di baris kode 19.

```
35  @uploadapi.route("/smile_detection",methods=["POST"])
36  def upload():
37      try:
38          obj = request.files.to_dict(flat=False)
39          files = obj["smile_file"]
40          file = files[0]
41          print(file)
42          input_type = request.form.get("input_type")
43          print(input_type)
44          now = datetime.now()
45          time_now = now.strftime("%H:%M")
46
47          name,ext = os.path.splitext(file.filename)
48          if(isImage(ext)):
49              filestr = file.read()
50              npimg = np.fromstring(filestr, np.uint8)
51              img = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
52              gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
53              gray = cv2.GaussianBlur(gray, (21, 21), 0)
54              print(is_smile(gray, img))
55              if is_smile(gray, img):
56                  if (input_type == "IN"):
57                      message="Senyuman anda manis sekali. Konfirmasi masuk diterima pukul " + time_now
58                  elif (input_type == "OUT"):
59                      message="Senyuman anda diterima. Konfirmasi keluar diterima pukul " + time_now
60                  return json_response(status_=200,varSenyum=True,msg=message)
61              else:
62                  return json_response(status_=200,varSenyum=False,msg="Anda belum tersenyum hari ini. Keep smile :)")
63      except Exception as e:
64          return json_response(status_=200,varSenyum=False,msg="Gagal menerima smile file. "+str(e))
65
```

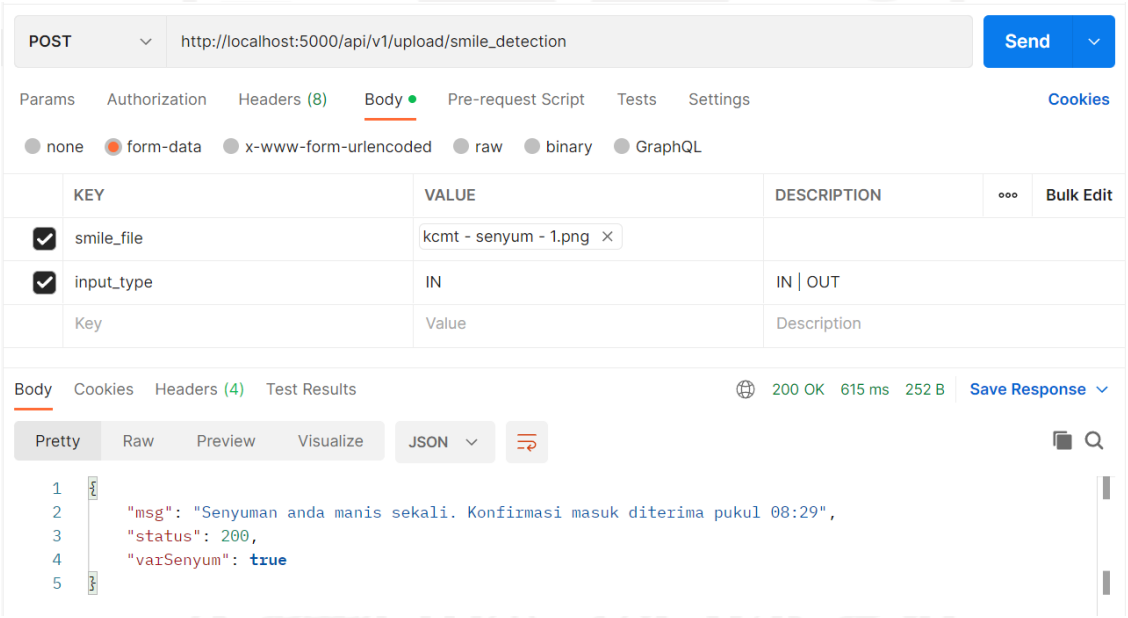
Gambar 4.27 Potongan script Python REST API smile detection part 2.

Function utama disini adalah **upload()** dimulai dari baris kode 36 yang sebelumnya didahului pendefinisian *route* untuk akses *endpoint*. Cara kerja dari *function* ini dimulai dari membaca *input* POST kiriman *mobile* yaitu **smile_type** dan **input_type** di baris kode 38-42, lalu dilakukan pengecekan ekstensi di baris kode 48, selanjutnya gambar *selfie* yang diupload tadi dibaca menggunakan *method* dari numpy dan openCV di baris kode 49-53. Setelah gambar *selfie* dapat dibaca, dilanjutkan dengan pengecekan dengan memanggil *function* **is_smile()** di baris kode 55. Jika hasil *function* **is_smile()** bernilai **true** dan **input_type "IN"** diberikan label

pesan sukses konfirmasi masuk (baris kode 56-57). Jika hasil *function* **is_smile()** bernilai **true** dan **input_type "OUT"** diberikan label pesan sukses konfirmasi keluar (baris kode 58-60). Akhir dari proses ini adalah dibentuk **JSON** sebagai format pesan untuk aplikasi mobile, jika sukses di baris kode 60, sedangkan jika gagal di baris kode 62 & 64. Pada Gambar 4.28 ditunjukkan perintah untuk menjalankan *service* REST API Smile Detection pada *development mode*, dan Gambar 4.29 menunjukkan parameter yang dikirim oleh aplikasi *mobile* ke REST API Smile Detection dan hasil JSON yang didapat.



Gambar 4.28 Menjalankan REST API smile detection di mode development.



Gambar 4.29 Hit REST API smile detection melalui POSTMAN.

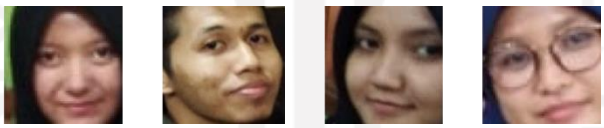
4.7 Pengujian Model Pengenalan Wajah

Berdasarkan hasil pengujian model atau *testing model* terhadap 70 foto dari 7 mahasiswa didapatkan nilai akurasi sebesar 92,9%. Nilai ini didapatkan dari menggunakan method **score()** pada klasifikasi SVM di *package* scikit learn dan melakukan perhitungan *confusion matrix* seperti terlihat pada Gambar 4.32.

Dilakukan juga pengujian atau *testing* secara *live* dari IP Camera dengan eksperimen pengambilan gambar suasana kelas sebanyak 5 foto atau 5 kali ambil dengan jumlah mahasiswa yang ada disana sekitar 1-5 orang. Gambar 4.30 menunjukan salah satu suasana kelas yang tertangkap dan Gambar 4.31 menunjukkan sejumlah mahasiswa yang terdeteksi.

Berdasarkan hasil pengujian atau *testing live* terhadap 9 foto dari simulasi seperti pada Tabel 4.4 didapatkan nilai akurasi sebesar 66,7%. Nilai ini didapatkan dari menggunakan method **score()** pada klasifikasi SVM di *package* scikit learn dan melakukan perhitungan *confusion matrix* seperti terlihat pada Gambar 4.33. Hasil dari pengujian live ini menghasilkan nilai akurasi yang lebih kecil jika dibandingkan dengan pengujian model.

Gambar 4.30. Tangkapan layar dari IP camera.



Gambar 4.31. Face detection dari tangkapan layar.

Tabel 4.4 Simulasi Testing Live Untuk Pengenalan Wajah

No	Skenario	Terdeteksi Seharusnya	Wajah Terdeteksi	Deteksi Sesuai	Dikenali Seharusnya	Wajah Dikenali	Dikenali Sesuai
1	Simulasi 1	2	2	Y	2	1	T
2	Simulasi 2	4	4	Y	4	3	T
3	Simulasi 3	2	1	T	1	1	Y
4	Simulasi 4	1	1	Y	1	1	Y
5	Simulasi 5	2	1	T	1	0	T
Total		11	9	Y = 3 T = 2	9	6	Y = 2 T = 3

Keterangan : Y = Ya, T= Tidak

	Prediksi						
Aktual	Fazlul	Ali	Rizki	Miftah	Putri	Vita	Yohani
Fazlul	10	0	0	0	0	0	0
Ali	0	10	0	0	0	0	0
Rizki	0	0	10	0	0	0	0
Miftah	0	0	0	10	0	0	0
Putri	0	0	0	0	10	0	0
Vita	0	0	0	0	2	5	3
Yohani	0	0	0	0	0	0	10

Gambar 4.32 Confusion matrix untuk testing model untuk pengenalan wajah.

	Prediksi						
Aktual	Fazlul	Ali	Rizki	Miftah	Putri	Vita	Yohani
Fazlul	1	0	0	0	0	0	0
Ali	0	0	0	0	0	0	0
Rizki	0	0	2	0	0	0	0
Miftah	0	0	0	0	0	0	1
Putri	0	0	0	1	0	0	1
Vita	0	0	0	0	0	1	0
Yohani	0	0	0	0	0	0	2

Gambar 4.33 Confusion matrix untuk testing live pengenalan wajah.

Merujuk pada Gambar 4.32 dilakukan perhitungan secara manual di bawah ini untuk mencocokkan dengan hasil yang didapat melalui *script* pada pengujian model yaitu :

Total data = 70

True Positive (TP) = TP (Fazlul) + TP (Ali) + TP (Rizki) + TP (Miftah) + TP (Putri) + TP (Vita) + TP (Yohani)

= 10 + 10 + 10 + 10 + 10 + 5 + 10

= 65

Accuracy = TP / Total data * 100%

= 65 / 70 * 100%

= 92,85 %

Merujuk pada Gambar 4.33 dilakukan perhitungan secara manual di bawah ini untuk mencocokkan dengan hasil yang didapat melalui *script* pada pengujian *live* yaitu :

Total data = 9

True Positive (TP) = TP (Fazlul) + TP (Ali) + TP (Rizki) + TP (Miftah) + TP (Putri) + TP (Vita) + TP (Yohani)

= 1 + 0 + 2 + 0 + 0 + 1 + 2

= 6

Accuracy = TP / Total data * 100%

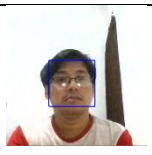
= 6 / 9 * 100%

= 66, 67 %

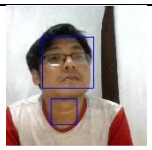
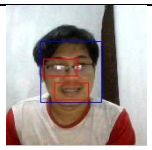
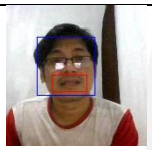
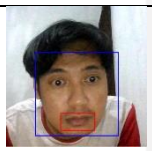
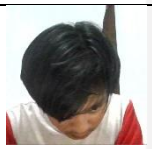
4.8 Pengujian Model Deteksi Senyuman

Model pengenalan senyuman yang digunakan pada sistem presensi ini adalah Haar Cascade. Model ini tidak perlu di-*training* tetapi langsung digunakan karena sudah disediakan oleh openCV [31]. Untuk menguji apakah model deteksi senyuman ini sudah sesuai atau belum dilakukan simulasi pengujian seperti pada Tabel 4.5. Pada simulasi ini dilakukan 10 kali percobaan dimana hanya 1 kali percobaan yang tidak sesuai atau 90% tingkat keberhasilannya. Walaupun tingkat keberhasilannya cukup tinggi tetapi ditemukan beberapa temuan yaitu di beberapa pose tertentu wajah tidak dapat dikenali yang nanti berpengaruh ke pengenalan senyum, beberapa pose tertentu mendapatkan deteksi wajah lebih dari 1, dan model ini belum dapat membedakan senyum yang terlihat gigi atau tanpa terlihat gigi.

Tabel 4.5 Simulasi Pengujian Model Deteksi Senyuman

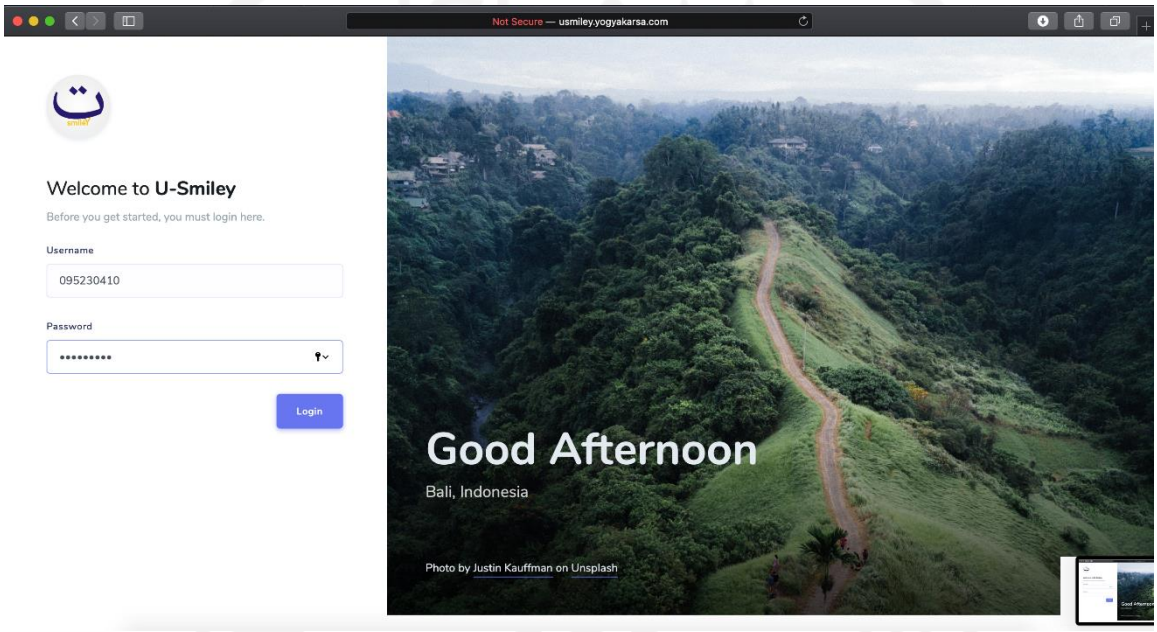
No	Picture	Attribut	Deteksi Wajah	Ekspresi	Label Senyum	Deteksi Senyum	Sesuai
1		Berkacamata	0, tidak terdeteksi	Datar	No Smile	No Smile	Ya
2		Berkacamata	1, terdeteksi	Datar	No Smile	No Smile	Ya

[31] https://github.com/opencv/opencv/tree/master/data/haarcascades

3		Berkacamata	2, terdeteksi area jakun	Datar, kepala menghadap ke atas	No Smile	No Smile	Ya
4		Berkacamata	1, terdeteksi	Senyum	Smile	Smile, mata kiri terdeteksi juga	Ya
5		Berkacamata	1, terdeteksi	Senyum	Smile	Smile	Ya
6		Tidak ada	0, tidak terdeteksi	Datar, wajah menghadap kanan	No Smile	No Smile	Ya
7		Tidak ada	1, terdeteksi	Bengong	No Smile	No Smile	Ya
8		Tidak ada	1, terdeteksi	Datar, mata melotot	No Smile	Smile	Tidak
9		Tidak ada	0, tidak terdeteksi	Data, kepala menunduk ke bawah	No Smile	No Smile	Ya
10		Tidak ada	1, terdeteksi	Sedikit senyum	Smile	Smile	Ya

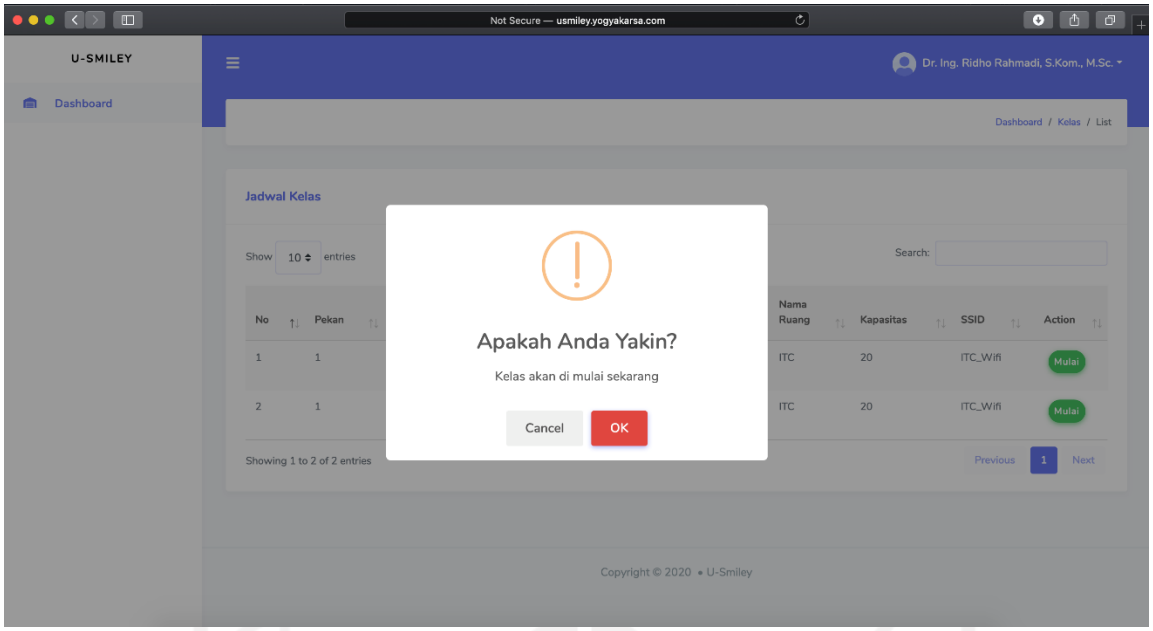
4.9 Bisnis Proses Presensi

Kegiatan Belajar Mengajar (KBM) diselenggarakan di ruangan kelas dari jadwal yang ditentukan dan diketahui baik oleh pengajar (dosen) maupun peserta didik (mahasiswa). Dalam memulai suatu pembelajaran dosen memasuki ruangan kelas, membuka kelas, lalu mengakses aplikasi web untuk presensi melalui komputer atau *mini computer* yang telah disediakan. Setiap dosen memiliki akun yang telah didaftarkan sebelumnya oleh admin, lalu mengakses URL aplikasi web seperti pada Gambar 4.34.

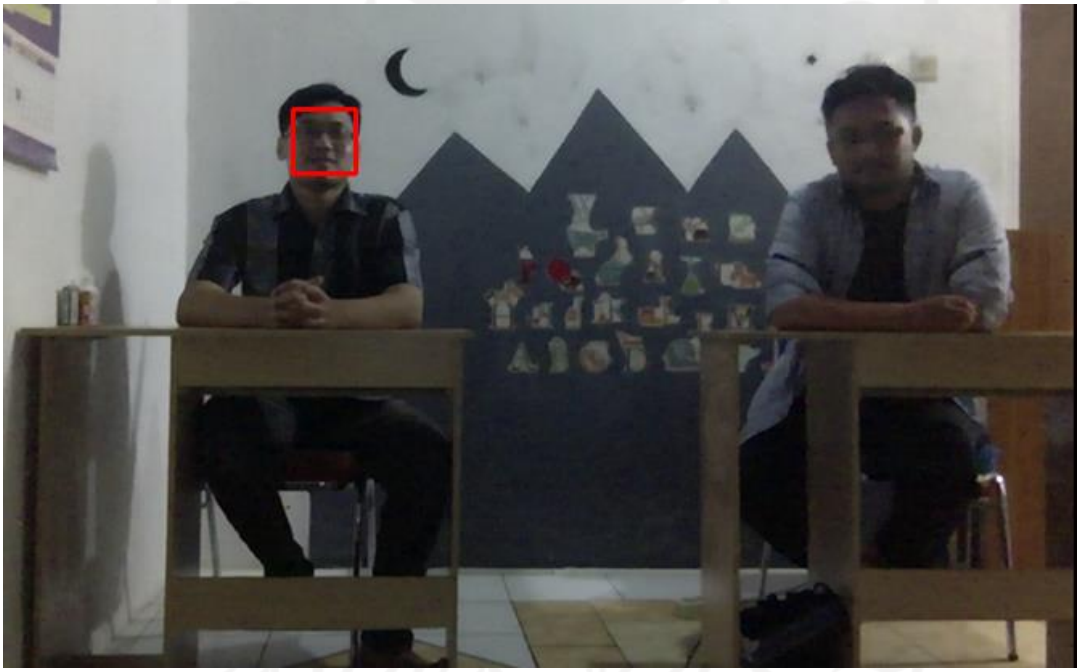


Gambar 4.34 Dosen melakukan login aplikasi web.

Sistem presensi ini hanya akan mulai berjalan saat dosen menekan tombol "**Mulai**" seperti pada Gambar 4.35. Hal yang perlu diperhatikan adalah tombol ini tersedia untuk semua jadwal, sehingga dosen perlu memastikan bahwa ini adalah jadwal yang tepat. Saat tombol "**Mulai**" ditekan, *trigger* untuk menjalankan penangkapan gambar suasana kelas dari IP CCTV dimulai. Service Python Saver & Python Recognizer akan bekerja dimulai dari menyimpan suasana kelas seperti pada Gambar 4.36. Selanjutnya dari gambar tersebut dilakukan pendeteksian wajah, sebagai contoh pada gambar dideteksi 1 wajah lalu dikenali yaitu sebagai mahasiswa dengan NIM 18917214 bernama Miftah. Saat NIM dikenali, saat itu juga dilakukan perekaman transaksi presensi ke database.

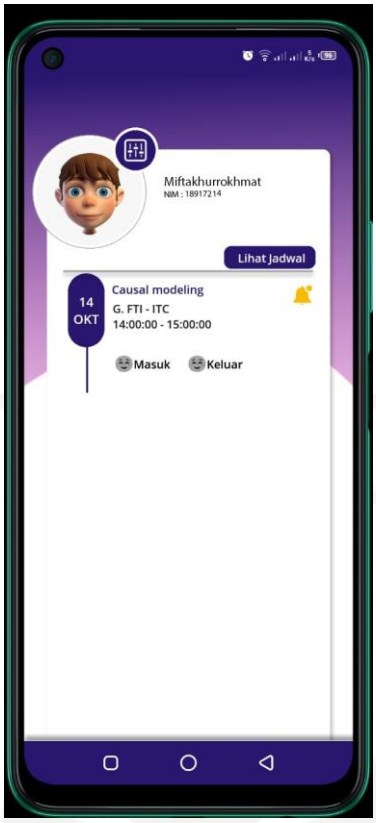


Gambar 4.35 Dosen memulai kelas dengan klik tombol Mulai.



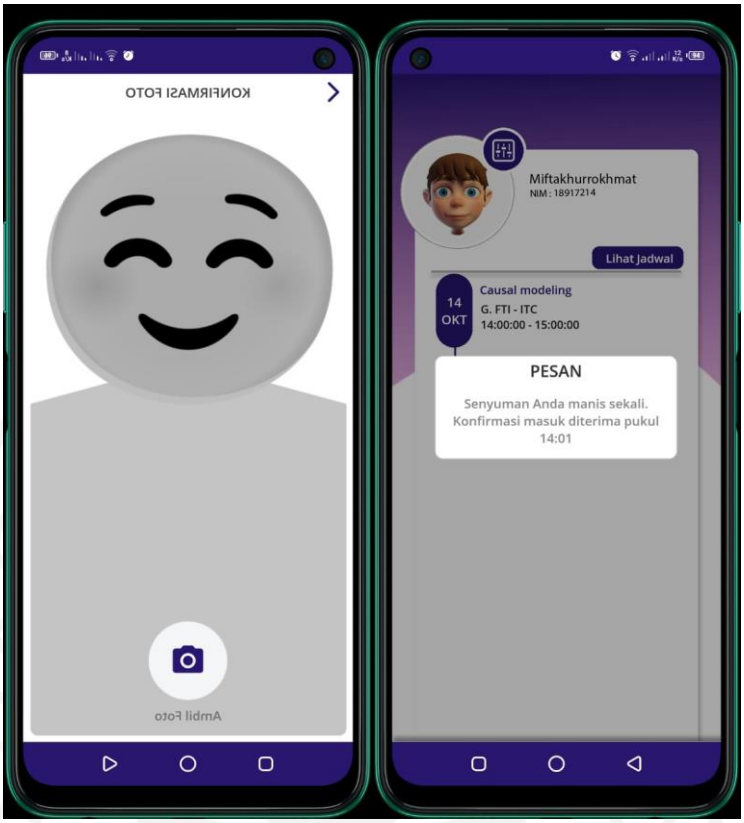
Gambar 4.36 Python Saver menangkap gambar dari IP CCTV/Camera.

Dikarenakan sistem ini telah terintegrasi dengan aplikasi *mobile* juga, saat suatu mahasiswa sudah tercatat di *database* presensi, selanjutnya apabila mahasiswa tersebut sudah meng-*install* lalu *login* ke aplikasi *mobile* akan mendapatkan notifikasi telah tercatat, saat ditekan akan muncul tampilan seperti Gambar 4.37.



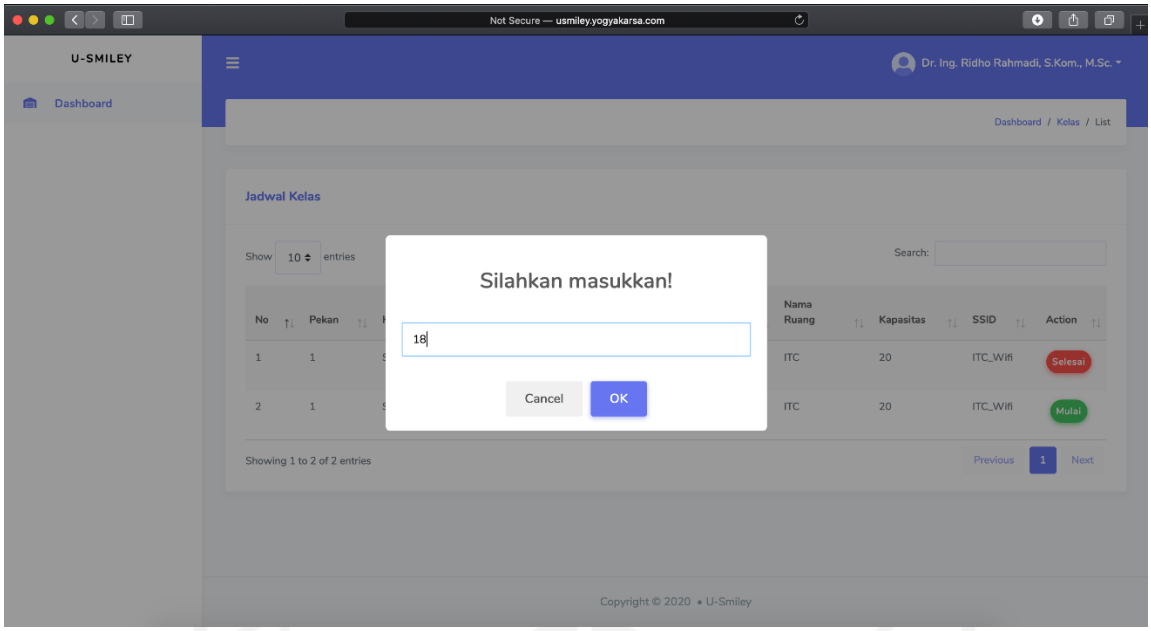
Gambar 4.37 Notifikasi Kehadiran di mobile app.

Berdasarkan perancangan yang telah dibuat sebelumnya, sistem presensi ini mengharuskan mahasiswa untuk melakukan konfirmasi kehadiran saat masuk kelas dengan melakukan *selfie* dengan keadaan tersenyum seperti pada Gambar 4.38. Hal ini dilakukan untuk memastikan bahwa *record* data yang tercatat memang benar miliknya, mengingat sistem ini masih sebatas purwarupa dan mode pengenalan wajah belum menghasilkan model yang benar-benar bagus.

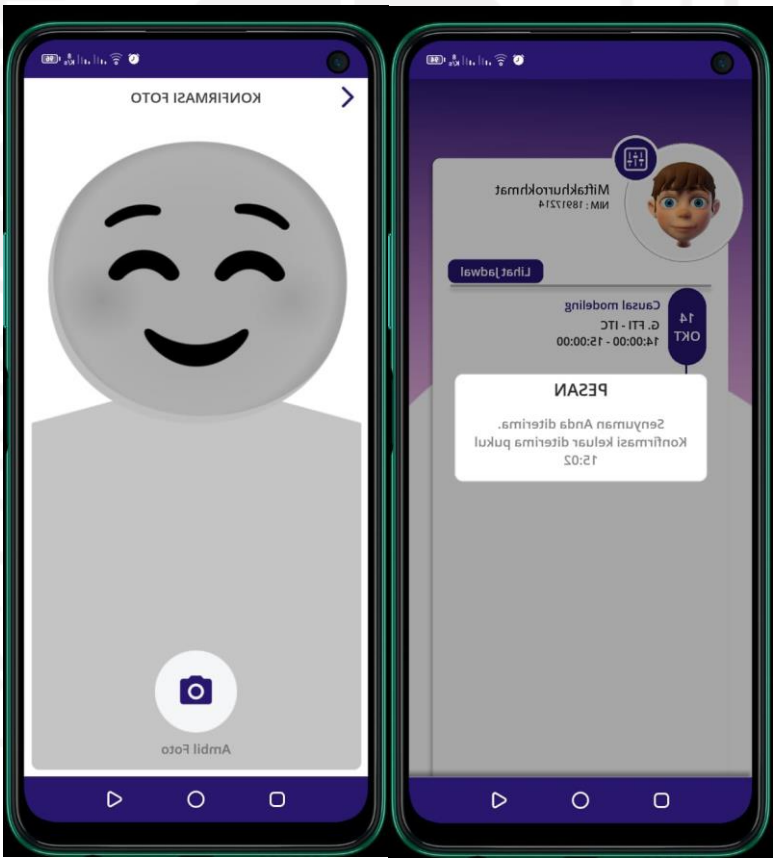


Gambar 4.38 Konfirmasi presensi masuk dan pesan sukses.

Proses yang dilakukan Python Saver & Python Recognizer akan berjalan terus menerus hingga kondisi kelas telah dinyatakan selesai, ditandai dengan dosen menutup kelas, lalu menekan tombol "**Selesai**" seperti pada Gambar 4.39, dilkonfirmasi dengan memasukkan jumlah mahasiswa yang hadir. Hal ini dilakukan sebagai *backup* jika di dalam sistem nantinya ada kesalahan perhitungan dan dapat digunakan sebagai data pembanding. Sama sepertinya proses pencatatan presensi masuk, di penutupan kelas ini mahasiswa akan mendapatkan notifikasi dan mahasiswa perlu melakukan konfirmasi untuk presensi keluar seperti terlihat pada Gambar 4.40.



Gambar 4.39 Dosen mengakhiri kelas dengan klik tombol Selesai.



Gambar 4.40 Konfirmasi presensi keluar dan pesan sukses.

4.10 Simulasi dan Evaluasi sistem

Dikarenakan pada saat penelitian sedang masa pandemi dan ruangan kelas selama perkuliahan tidak dapat digunakan, maka simulasi dilakukan dengan membuat suasana mirip kelas dengan peralatan seperti pada Tabel 4.6.

Tabel 4.6 Peralatan Simulasi

No	Nama Item	Qty	Spesifikasi
1	Raspberry	1	Raspberry Pi 3 Model B+ [32]
2	IP Cam	1	ESCAM PVR008
3	Switch AP	1	3G/4G Wireless N Router TL-MR3420

Dalam simulasi ini dilakukan *test case* terlihat di Tabel 4.7 dengan serangkaian kegiatan untuk memastikan alur sistem apakah sudah sesuai dengan perancangan yang telah dibuat. *Test case* ini dibuat dari alur pokok utama dalam sistem presensi, diawali dari dosen memulai kelas hingga dosen menutup kelas diikuti mahasiswa melakukan konfirmasi akhir dengan *selfie* senyuman. *Test case* ini adalah hasil akhir dari simulasi dan belum diterapkan dalam di kondisi *real* / lapangan sehingga kemungkinan nanti di kondisi real dapat berbeda. Dari keseluruhan (14 skenario) ini jumlah hasil yang diharapkan dan hasil pengujian sama, dengan beberapa kondisi prasyarat dan beberapa hal yang mempengaruhi dalam kondisi terbaik.

Tabel 4.7 Test Case Sistem Presensi Berbasis Pola Pengenalan Wajah dan Tersenyum

No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	Ujicoba dosen login aplikasi web	Dosen login aplikasi web dengan memasukkan username dan password.	Berhasil	Berhasil	*Prasyarat :* Akun login untuk dosen disediakan oleh admin
2	Ujicoba dosen lihat jadwal	Dosen melihat jadwal kelas yang sesuai dengan dirinya.	Berhasil	Berhasil	*Prasyarat :* Jadwal perkuliahan

[32] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

					diupdate olah admin
3	Ujicoba dosen mulai kelas	Dosen memulai kelas dengan klik tombol "Mulai".	Berhasil	Berhasil	*Dipengaruhi oleh : trigger* dosen agar tepat waktu
4	Ujicoba tangkap gambar mahasiswa di kelas	Mahasiswa yang berada di kelas berhasil ditangkap gambarnya melalui Python Saver.	Berhasil	Berhasil	*Dipengaruhi oleh* : kestabilan *service* untuk menangkap gambar, memungkinkan gambar tidak tertangkap; kualitas piksel IP CCTV/Camera memungkinkan gambar dengan piksel rendah berpengaruh ke *face detection.*
5	Ujicoba deteksi wajah	Gambar mahasiswa yang ditangkap, dideteksi wajahnya melalui Python Recognizer.	Berhasil	Berhasil	*Dipengaruhi oleh :* kemampuan *library* deteksi wajah, memungkinkan terjadinya kegagalan deteksi wajah

6	Ujicoba pengenalan wajah	Gambar mahasiswa yang ditangkap lalu dideteksi, dikenali melalui Python Recognizer.	Berhasil	Berhasil	D*ipengaruhi oleh :* hasil *training* model pengenalan wajah, memungkinkan terjadinya kegagalan pengenalan wajah
7	Ujicoba pencatatan presensi	Mahasiswa yang dikenali wajah didapatkan NIM lalu dicatat di database melalui hit REST API Presensi.	Berhasil	Berhasil	*Dipengaruhi oleh :* kestabilan REST API Presensi, memungkinkan *down* saat diakses
8	Ujicoba Mahasiswa login melalui mobile app	Mahasiswa login di mobile app untuk mendapatkan beberapa fitur.	Berhasil	Berhasil	*Prasyarat :* Akun login untuk mahasiswa disediakan oleh admin
9	Ujicoba mahasiswa dapat notif presensi masuk	Mahasiswa mendapatkan notif presensi masuk di mobile app.	Berhasil	Berhasil	*Dipengaruhi oleh* : kestabilan *service broadcast messaging* dan koneksi internet

					masing-masing mahasiswa.
10	Ujicoba mahasiswa konfirmasi senyum untuk presensi masuk	Mahasiswa melakukan konfirmasi senyum untuk presensi masuk di mobile app lalu mendapatkan informasi jika telah berhasil.	Berhasil	Berhasil	*Dipengaruhi : oleh :* keberhasilan *upload selfie* senyuman dan hasil deteksi senyuman
11	Ujicoba dosen mengakhiri kelas	Dosen mengakhiri kelas dan menekan tombol "Selesai" dengan mencatat jumlah mahasiswa yang masuk di aplikasi web.	Berhasil	Berhasil	*Dipengaruhi oleh : trigger* dosen agar tepat waktu
12	Ujicoba mahasiswa dapat notif presensi keluar	Mahasiswa mendapatkan notif presensi keluar di mobile app.	Berhasil	Berhasil	*Dipengaruhi oleh* : kestabilan *service broadcast messaging* dan koneksi internet masing-masing mahasiswa
13	Ujicoba mahasiswa konfirmasi senyum untuk presensi keluar	Mahasiswa melakukan konfirmasi senyum untuk presensi keluar di mobile app lalu mendapatkan informasi jika telah berhasil.	Berhasil	Berhasil	*Dipengaruhi : oleh :* keberhasilan *upload selfie* senyuman dan hasil deteksi senyuman

14	Ujicoba mahasiswa melihat status kehadiran	Mahasiswa melihat status kehadirannya di mobile app telah ditandai.	Berhasil	Berhasil	*Dipengaruhi oleh :* hasil konfirmasi masuk dan keluar dengan senyuman

Selain itu ada beberapa temuan juga saat melakukan simulasi yang nanti dapat dipergunakan untuk perbaikan sebagai berikut :

1. Python Saver atau *script* yang terhubung dengan IP Camera dapat *menyebabkan mini computer* terasa lambat, apalagi saat ditambah dengan melakukan load *model deep learning (transfer learning* + model klasifikasi*)* pada Python Recognizer, *memory* 1GB langsung habis sehingga dapat menyebabkan *mini computer* tidak bekerja secara maksimal
2. Saat simulasi kelas hanya terisi sekitar 2-5 orang, sehingga belum dapat mensimulasi suasana kelas dengan banyak orang.
3. Simulasi ini agak susah diterapin di tempat berbeda dikarenakan untuk sistem ini dapat bekerja harus melakukan banyak *setup* yaitu : instalasi *database*, instalasi web app, instalasi *web service* (API), pemasangan IP Camera, dan menghubungkan aplikasi dan alat ke dalam 1 *network*.

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dari penelitian yang dilakukan didapatkan hasil terbentuknya purwarupa sistem presensi berbasis pengenalan pola wajah dan tersenyum dengan menggunakan arsitektur *deep learning* berbasis Convolutional Neural Network (CNN) yaitu FaceNet. FaceNet digunakan sebagai *feature extractor* dan dikombinasikan dengan SVM for *Classifier (SVC)*, dan untuk pengenalan senyum digunakan algoritma *machine learning* Haar Cascade. Dalam simulasi sistem didapatkan hasil, sistem bisa berjalan baik dengan beberapa prasyarat dan faktor pendukung dalam kondisi terbaik. Selain itu juga, untuk model pengenalan wajah dan pengenalan senyuman sudah dilakukan pengujian. Hasil pengujian dari model pengenalan wajah menghasilkan akurasi test model sebesar 92,9%, sedangkan test *live* akurasi hanya sebesar 66,7%. Hal ini dipengaruhi oleh proses sebelum *face recognition* yaitu *face detection* menggunakan *library* Dlib. Hasil deteksi wajah yang terlalu kecil berpengaruh terhadap tingkat pengenalan wajah sehingga akurasinya berkurang. Model pengenalan senyuman dari hasil pengujian menunjukkan hasil 90% dari 10 percobaan sehingga pemanfaatan Haar Cascade untuk pengenalan senyuman cukup mendukung jalannya sistem.

5.2 Saran

Dari pembahasan dan kesimpulan yang diperoleh, ada beberapa point yang dapat dipergunakan untuk studi berkelanjutan yaitu:

1. Peningkatan jumlah *dataset* dan variasi serta lebih banyak eksperimen di *training* untuk mendapatkan hasil model yang lebih baik lagi.
2. Dalam menerapkan arsitektur yang digunakan (Inception-ResNet-v1) dalam *script* Python masih menggunakan tensorflow secara langsung. Hal ini berdampak apabila akan melakukan *upgrade* versi tensorflow akan berdampak langsung terhadap arsitektur yang telah disusun. Disarankan untuk penerapan arsitektur *deep learning* yang lebih baik menggunakan Keras API sehingga lebih memungkinkan untuk bereksperiman dan mengganti arsitektur lain.

3. Pengembangan model *deep learning* ini masih menggunakan tensorflow versi 1 sehingga belum dapat memanfaatkan fitur terbaru dari tensorflow 2, perlu ada penyesuaian kode agar dapat berjalan baik di tensorflow 2.
4. Arsitektur yang dipakai masih menggunakan Inception-ResNet-v1, untuk peningkatan hasil dari model dapat dilanjutkan ke Inception-ResNet-v2 atau menggunakan arsitektur lain.
5. Klasifikasi yang digunakan saat ini yaitu *SVM Classifier*, selanjutnya di penelitian selanjutnya dapat diganti dengan algoritma lain atau bereksperimen dengan memodifikasi layer akhir pada arsitektur FaceNet.
6. "Presensi Kelas Berbasis Pola Wajah dan Tersenyum Menggunakan Deep Learning" ini masih sebatas purwarupa atau *prototype*, dikarenakan saat penelitian berlangsung sedang masa pandemi dan pembelajaran berlangsung secara *online*, selanjutnya di kemudian hari dapat diujicobakan secara langsung di lapangan sehingga lebih mendapatkan beberapa *case* yang lebih spesifik.
7. Pemanfaatan algoritma *Haar Cascade Classifier* sebagai deteksi senyuman dapat diimplementasikan dalam bentuk REST API, akan tetapi hasil yang didapat belum terlalu bagus dikarenakan ada beberapa pose dan eksepresi yang tidak tepat dikenali, sehingga di kemudian hari dapat dikembangkan model pengenalan senyuman dengan teknik lain.
8. Efek presensi dengan senyuman memberikan efek psikologis yang baik dalam suatu pembelajaran, akan tetapi penerapan dalam purwarupa ini agak merepotkan mahasiswa, sehingga di kemudian hari dapat dikembangkan arsitektur yang lebih ringkas seperti menggabungkan deteksi senyuman bersamaan dengan pengenalan wajah saat di kelas, dan menghilangkan penggunaan aplikasi *mobile* oleh mahasiswa.

Daftar Pustaka

Ahlawat, S., & Choudhary, A. (2020). Hybrid CNN-SVM Classifier for Handwritten Digit Recognition. *Procedia Computer Science*, *167*. https://doi.org/10.1016/j.procs.2020.03.309

Amirgaliyev, Y., Sadykova, A., & Kenshimov, C. (2021). COMPARISION OF FACE DETECTION TOOLS. *BULLETIN Series of Physics & Mathematical Sciences*, *76*(4), 59–64. https://doi.org/10.51889/2021-4.1728-7901.08

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2). https://doi.org/10.1023/A:1009715923555

Derisma, D. (2016). Sistem Pengenalan Wajah Secara Realtime Berbasis Android Menggunakan Metode Eigenface Pada OpenCV. *Jurnal Komputer Terapan*, *2*(2), 127–136.

Fachmi, Z., Sudarma, M., & Jasa, L. (2019). Sistem Monitoring Kehadiran Perkuliahan Menggunakan Face Detection Dengan Algoritma Viola Jones. *Majalah Ilmiah Teknologi Elektro*. https://doi.org/10.24843/mite.2019.v18i01.p18

Fakih, A., Raharjana, I. K., & Zaman, B. (2015). Pemanfaatan Teknologi Fingerprint Authentication untuk Otomatisasi Presensi Perkuliahan. *Journal of Information Systems Engineering and Business Intelligence*, *1*(2), 41. https://doi.org/10.20473/jisebi.1.2.41-48

Gao, Y., & Mosalam, K. M. (2018). Deep Transfer Learning for Image-Based Structural Damage Recognition. *Computer-Aided Civil and Infrastructure Engineering*, *33*(9). https://doi.org/10.1111/mice.12363

Hu, F., Xia, G. S., Hu, J., & Zhang, L. (2015). Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*. https://doi.org/10.3390/rs71114680

Larsen-Freeman, D. (2013). Transfer of Learning Transformed. *Language Learning*, *63*(SUPPL. 1). https://doi.org/10.1111/j.1467-9922.2012.00740.x

Lino, A. F. S., Silva, B. C. R., Rocha, D. P. C., Furriel, G. P., & Calixto, W. P. (2017). Performance of haar and LBP features in cascade classifiers to whiteflies detection and counting. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2017 - Proceedings*, *2017-January*. https://doi.org/10.1109/CHILECON.2017.8229737

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2016). Fully convolutional neural networks for remote sensing image classification. *International Geoscience and Remote Sensing Symposium (IGARSS)*. https://doi.org/10.1109/IGARSS.2016.7730322

Maslihatin, T., Sulehu, M., & Darmansyah. (2020). Sistem Asosiasi Penyusunan Obat Pada Apotek Balai Rehabilitasi Badan Narkotika Nasional Baddoka Menggunakan Algoritma Apriori. *Celebes Computer Science Journal*, *2 No 2*(2020-10–30), 27–38. http://journal.lldikti9.id/ccsj/article/view/518

Matsumoto, T., & Matsumoto, H. (2002). Impact of artificial gummy fingers on fingerprint

systems. *Proceedings of …*.

Prasti, D. (2014). Sistem Presensi Perkuliahan Dengan Kartu Mahasiswa Menggunakan Barcode. *Jurnal Ilmiah d'ComPutarE*, *4*(Juni).

Samirso. (n.d.). *ZKTeco - How to Enter the Device Without Admin Affirming*. Retrieved August 17, 2019, from https://www.instructables.com/id/ZKTeco-How-to-Enter-the-Device-Without-Admin-Affir

Santoso, B., & Kristianto, R. P. (2020). IMPLEMENTASI PENGGUNAAN OPENCV PADA FACE RECOGNITION UNTUK SISTEM PRESENSI PERKULIAHAN MAHASISWA. *SISTEMASI*, *9*(2). https://doi.org/10.32520/stmsi.v9i2.822

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2015.7298682

Sharma, H., Saurav, S., Singh, S., Saini, A. K., & Saini, R. (2015). Analyzing impact of image scaling algorithms on viola-jones face detection framework. *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*. https://doi.org/10.1109/ICACCI.2015.7275860

Suhery, C., & Ruslianto, I. (2017). Identifikasi Wajah Manusia untuk Sistem Monitoring Kehadiran Perkuliahan menggunakan Ekstraksi Fitur Principal Component Analysis (PCA). *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *3*(1), 9. https://doi.org/10.26418/jp.v3i1.19792

Susanti, R. (2013). PENERAPAN PENDIDIKAN KARAKTER DI KALANGAN MAHASISWA. *Al-Ta Lim Journal*, *20*(3). https://doi.org/10.15548/jt.v20i3.46

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 4278–4284. https://doi.org/10.1609/aaai.v31i1.11231

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June-2015*. https://doi.org/10.1109/CVPR.2015.7298594

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*, *17*(1). https://doi.org/10.1016/j.aci.2018.08.003

Wahyudiana, N., & Budi, S. (2019). Perbandingan Performa Pre-Trained Classifier dLib dan HAAR Cascade (OpenCV) Untuk Mendeteksi Wajah. *Jurnal Strategi*, *1*, 376.

William, I., Ignatius Moses Setiadi, D. R., Rachmawanto, E. H., Santoso, H. A., & Sari, C. A. (2019). Face Recognition using FaceNet (Survey, Performance Test, and Comparison). *Proceedings of 2019 4th International Conference on Informatics and Computing, ICIC 2019*. https://doi.org/10.1109/ICIC47613.2019.8985786

Yin, X., & Liu, X. (2018). Multi-Task Convolutional Neural Network for Pose-Invariant Face Recognition. *IEEE Transactions on Image Processing*, *27*(2). https://doi.org/10.1109/TIP.2017.2765830

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka OpenCV dan DLIB Python. *Sainstech*.