TUGAS AKHIR - ET234801

IMPLEMENTASI *FEDERATED LEARNING* PADA SISTEM KEHADIRAN BERBASIS PENGENALAN WAJAH DI LINGKUNGAN *EDGE COMPUTING*

STEVEN FIGO

NRP 5027221021

Dosen Pembimbing

Fuad Dary Rosyadi, S.Kom., M.Kom.

NIP 19960910202406 1 003

Dr. Rizka Wakhidatus Sholikah, S.Kom.

NIP 1993202012054

Program Studi S1 Teknologi Informasi

Departemen Teknologi Informasi

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2026

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – ET234801

IMPLEMENTASI *FEDERATED LEARNING* PADA SISTEM KEHADIRAN BERBASIS PENGENALAN WAJAH DI LINGKUNGAN *EDGE COMPUTING*

STEVEN FIGO

NRP 5027221021

Dosen Pembimbing

Fuad Dary Rosyadi, S.Kom., M.Kom.

NIP 19960910202406 1 003

Dr. Rizka Wakhidatus Sholikah, S.Kom.

NIP 1993202012054

Program Studi S1 Teknologi Informasi

Departemen Teknologi Informasi

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2026

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – ET234801

IMPLEMENTATION OF FEDERATED LEARNING IN A FACE RECOGNITION-BASED ATTENDANCE SYSTEM IN AN EDGE COMPUTING ENVIRONMENT

STEVEN FIGO

NRP 5027221021

Advisor

Fuad Dary Rosyadi, S.Kom., M.Kom.

NIP 19960910202406 1 003

Dr. Rizka Wakhidatus Sholikah, S.Kom.

NIP 1993202012054

Study Program Information Technology

Department of Information Technology

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2026

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI *FEDERATED LEARNING* PADA SISTEM KEHADIRAN BERBASIS PENGENALAN WAJAH DI LINGKUNGAN *EDGE COMPUTING*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknologi Informasi
Departemen Teknologi Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **Steven Figo**
NRP. 5027221021

Disetujui oleh Tim Penguji Proposal Tugas Akhir :

1. Fuad Dary Rosyadi, S.Kom., M.Kom. — Pembimbing
2. Dr. Rizka Wakhidatus Sholikah, S.Kom. — Ko-pembimbing
3. Dr.techn. Ir. Raden Venantius Hari Ginardi, M.Sc. — Penguji 1
4. Gagatsatya Adiatmaja, S.T., M.Kom. — Penguji 2

SURABAYA
Juni, 2026

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

IMPLEMENTATION OF FEDERATED LEARNING IN A FACE RECOGNITION-BASED ATTENDANCE SYSTEM IN AN EDGE COMPUTING ENVIRONMENT

FINAL PROJECT

Submitted to fulfill one of the requirements

for obtaining a Bachelor of Computer Science degree in

Study Program Information Technology

Department of Information Technology

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

By : **Steven Figo**

NRP. 5027221021

Approved by Final Project Examiner Team :

1. Fuad Dary Rosyadi, S.Kom., M.Kom. — Advisor
2. Dr. Rizka Wakhidatus Sholikah, S.Kom. — Co-Advisor
3. Dr.techn. Ir. Raden Venantius Hari Ginardi, M.Sc. — Examiner
4. Gagatsatya Adiatmaja, S.T., M.Kom. — Examiner

SURABAYA

June, 2026

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	: Steven Figo / 5027221021
Program studi	: Teknologi Informasi
Dosen Pembimbing / NIP	: Fuad Dary Rosyadi, S.Kom., M.Kom. / 199609102024061003

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Implementasi *Federated Learning* pada Sistem Kehadiran Berbasis Pengenalan Wajah di Lingkungan *Edge Computing*" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, Juli 2026

Mengetahui Dosen Pembimbing 1	Mengetahui Dosen Pembimbing 2	Mahasiswa
Fuad Dary Rosyadi, S.Kom., M.Kom. NIP. 199609102024061003	Dr. Rizka Wakhidatus Sholikah, S.Kom. NIP. 1993202012054	Steven Figo NRP. 5027221021

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP	: Steven Figo / 5027221021
Department	: Information Technology
Advisor / NIP	: Fuad Dary Rosyadi, S.Kom., M.Kom. / 199609102024061003

Hereby declare that the Final Project with the title of "Implementation of Federated Learning in a Face Recognition-Based Attendance System in an Edge Computing Environment" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, July 2026

Acknowledged Advisor	Acknowledged Co-Advisor	Student
Fuad Dary Rosyadi, S.Kom., M.Kom. NIP. 199609102024061003	Dr. Rizka Wakhidatus Sholikah, S.Kom. NIP. 1993202012054	Steven Figo NRP. 5027221021

[Halaman ini sengaja dikosongkan]

ABSTRAK

IMPLEMENTASI *FEDERATED LEARNING* PADA SISTEM KEHADIRAN BERBASIS PENGENALAN WAJAH DI LINGKUNGAN *EDGE COMPUTING*

Nama Mahasiswa / NRP	**:**	**Steven Figo / 5027221021**
Departemen	**:**	**Teknologi Informasi / FTEIC - ITS**
Dosen Pembimbing	**:**	**1. Fuad Dary Rosyadi, S.Kom., M.Kom.**
		2. Dr. Rizka Wakhidatus Sholikah, S.Kom.

Abstrak

Sistem absensi berbasis pengenalan wajah saat ini telah banyak diterapkan untuk mengotomatiskan pencatatan kehadiran, namun metode konvensional berbasis *Centralized Learning* mengharuskan pengumpulan citra wajah mentah ke server pusat yang rentan terhadap risiko kebocoran data biometrik sensitif serta pemborosan *bandwidth* jaringan. Penelitian ini mengusulkan implementasi *Federated Learning* menggunakan algoritma FedProx dan model MobileFaceNet yang dikontainerisasi dengan Docker untuk membangun sistem kehadiran terdistribusi pada lingkungan *Edge Computing* menggunakan perangkat keras heterogen (Jetson Nano dan Raspberry Pi 3 Model B). Pendekatan ini memungkinkan pelatihan model dilakukan secara lokal pada masing-masing perangkat *edge* tanpa mengirimkan data biometrik mentah mahasiswa ke server, sehingga privasi data tetap terjaga. Pengujian dilakukan melalui evaluasi performa pelatihan model, efisiensi sumber daya, serta pengujian inferensi nyata dan video simulasi. Hasil penelitian menunjukkan bahwa model global hasil *Federated Learning* mencapai akurasi validasi hingga 99,69%, sebanding bahkan sedikit melampaui *Centralized Learning* yang konsisten mencatatkan akurasi validasi 99,65%. Dari sisi efisiensi sumber daya, *Federated Learning* jauh lebih hemat *bandwidth* dengan hanya mentransmisikan 167,28 MB, sekitar 22% dari kebutuhan *Centralized Learning* sebesar 744,99 MB, namun harus dibayar dengan durasi pelatihan dan konsumsi energi lebih besar akibat keterbatasan perangkat Raspberry Pi. Pada pengujian inferensi nyata, sistem mencatatkan keberhasilan fungsionalitas 100% pada jarak 0,5 hingga 3 meter di kondisi cahaya cukup maupun minim, dengan akurasi klasifikasi stabil di atas 90% menggunakan ambang batas kepercayaan 0,7. Latensi inferensi lokal pada Jetson Nano tercatat cepat sekitar 700-765 ms, sedangkan Raspberry Pi jauh lebih lambat sekitar 3100-3230 ms akibat keterbatasan RAM. Meskipun demikian, pengujian video simulasi menunjukkan *Federated Learning* masih lebih rentan terhadap kasus salah absen dibandingkan *Centralized Learning*. Penelitian ini menunjukkan bahwa integrasi *Federated Learning* dan *Edge Computing* efektif membangun sistem presensi yang hemat *bandwidth* dan menjaga privasi data biometrik, meski masih memerlukan peningkatan ketahanan terhadap kesalahan identifikasi.

Kata kunci: *Edge Computing*, *Face Recognition*, *Federated Learning*, MobileFaceNet, *Smart Attendance*.

[Halaman ini sengaja dikosongkan]

ABSTRACT

IMPLEMENTATION OF FEDERATED LEARNING IN A FACE RECOGNITION-BASED ATTENDANCE SYSTEM IN AN EDGE COMPUTING ENVIRONMENT

Student Name / NRP	**:**	**Steven Figo / 5027221021**
Department	**:**	**Information Technology / FTEIC - ITS**
Advisor	**:**	**1. Fuad Dary Rosyadi, S.Kom., M.Kom.**
		2. Dr. Rizka Wakhidatus Sholikah, S.Kom.

Abstract

Face recognition-based attendance systems have been widely adopted to automate presence recording; however, conventional Centralized Learning methods require collecting raw facial images at a central server, posing risks of biometric data leakage and excessive network bandwidth consumption. This study proposes implementing Federated Learning using the FedProx algorithm and MobileFaceNet model, containerized with Docker, to build a distributed attendance system in an Edge Computing environment using heterogeneous hardware (Jetson Nano and Raspberry Pi 3 Model B). This approach enables local model training on each edge device without transmitting raw student biometric data to the server, thereby preserving data privacy. Evaluation was conducted through training performance assessment, resource efficiency analysis, and real-life inference and video simulation testing. Results show the global model produced by Federated Learning achieved a validation accuracy of up to 99.69%, comparable to and slightly exceeding the consistent 99.65% accuracy of Centralized Learning. Federated Learning also demonstrated significantly greater bandwidth efficiency, transmitting only 167.28 MB, around 22% of the 744.99 MB required by Centralized Learning, though at the cost of longer training duration and higher energy consumption due to the limited capacity of the Raspberry Pi. In real-life inference testing, the system achieved 100% functional success at distances of 0.5 to 3 meters under both adequate and low-light conditions, with classification accuracy consistently above 90% at a confidence threshold of 0.7. Local inference latency on the Jetson Nano remained fast at approximately 700-765 ms, while the Raspberry Pi was considerably slower at approximately 3,100-3,230 ms due to RAM constraints. Nevertheless, video simulation testing revealed that Federated Learning remains more susceptible to false acceptance cases than Centralized Learning. These findings demonstrate that Federated Learning and Edge Computing effectively build a bandwidth-efficient, privacy-preserving attendance system, though further improvement in robustness against misidentification is still required.

Keywords: Edge Computing, Face Recognition, Federated Learning, MobileFaceNet, Smart Attendance.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan atas ke hadirat Tuhan Yang Maha Esa yang telah melimpahkan karunia, rezeki, dan rahmat-Nya sehingga penulis mampu menyelesaikan Tugas Akhir yang berjudul:

"IMPLEMENTASI *FEDERATED LEARNING* PADA SISTEM KEHADIRAN BERBASIS PENGENALAN WAJAH DI LINGKUNGAN *EDGE COMPUTING*"

Tugas Akhir ini dilaksanakan sebagai salah satu syarat kelulusan mahasiswa S-1 Departemen Teknologi Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Pengerjaan Tugas Akhir ini tidak lepas dari dukungan dan doa dari berbagai pihak yang membantu penulis. Oleh karena itu, penulis menyampaikan terima kasih sebesar-besarnya kepada:

1. Bapak Hardy Figo, Ibu Ling Cun, Owen Figo, dan Natasya Louise Figo selaku keluarga penulis yang selalu hadir memberikan dukungan tanpa henti yang membantu, mendoakan, dan memberikan yang terbaik dalam proses Tugas Akhir maupun selama proses perkuliahan.
2. Bapak Fuad Dary Rosyadi, S.Kom., M.Kom. dan Ibu Dr. Rizka Wakhidatus Sholikah, S.Kom., selaku dosen pembimbing penulis yang selalu memberikan masukan dan bantuan dalam proses penulisan Tugas Akhir ini.
3. Seluruh rekan-rekan Teknologi Informasi angkatan 2022 yang selalu mendukung satu sama lain.
4. Ellyzah, Zahra, Jody, Dwi, Bhisma, Hafiz yang membantu dan mendukung penulis dalam proses perkuliahan maupun penulisan Tugas Akhir.
5. Arya, Joel, Sherwin, Leowin, dan Marshal selaku rekan SMA penulis yang senantiasa mendukung penulis.
6. Tim BCFC selaku tim lomba paduan suara penulis yang selalu mendukung penulis.
7. Pihak lainnya yang telah membantu proses pengerjaan Tugas Akhir ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa masih terdapat kekurangan dalam pelaksanaan dan penulisan Tugas Akhir ini. Oleh karena itu, penulis menerima berbagai kritik dan saran dari pembaca agar penelitian ini dapat dikembangkan menjadi lebih baik. Besar harapan penulis supaya Tugas Akhir ini dapat bermanfaat bagi kita semua dan dapat dikembangkan menjadi penelitian yang lebih baik lagi.

Surabaya, Juni 2026

Steven Figo

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR LAMPIRAN

[Halaman ini sengaja dikosongkan]

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan kecerdasan buatan (*Artificial Intelligence*/AI) dalam beberapa tahun terakhir berlangsung sangat pesat dan memberikan dampak signifikan pada berbagai sektor, seperti industri, pendidikan, kesehatan, dan transportasi. Kemampuan AI dalam mengolah data berskala besar serta mengenali pola tertentu di dalamnya membuat proses pengambilan keputusan menjadi lebih cepat dan akurat. Di saat yang sama, penggunaan perangkat *Internet of Things* (IoT) juga semakin meluas sehingga menghasilkan lebih banyak data dari perangkat yang saling terhubung. Namun, perkembangan AI dan IoT ini turut menghadirkan tantangan baru, terutama terkait keamanan data, privasi pengguna, dan kompleksitas dalam pengelolaannya (Brecko et al., 2022).

Sebagian besar sistem AI saat ini masih menggunakan metode *Centralized Learning*, yaitu pendekatan yang mengharuskan seluruh data dikumpulkan ke satu server pusat sebelum dilatih. Pendekatan ini memang efektif, tetapi kurang ideal karena berpotensi menimbulkan kebocoran data pribadi serta memerlukan *bandwidth* yang besar, terutama ketika data mentah dikirim dari banyak perangkat IoT (Nandi et al., 2023). Permasalahan ini menjadi semakin menonjol pada aplikasi yang memanfaatkan data biometrik, seperti pengenalan wajah, karena jenis data tersebut bersifat sangat sensitif dan tidak dapat diganti apabila terjadi kebocoran.

Pengenalan wajah merupakan salah satu teknologi yang banyak digunakan dalam sistem kehadiran karena mampu mengotomatiskan proses identifikasi sekaligus mengurangi ketergantungan pada metode absensi manual. Namun, penggunaan citra wajah sebagai data biometrik menimbulkan persoalan penting terkait privasi, terutama ketika data tersebut perlu diproses atau disimpan di server pusat. Srinu et al. (2025) mengembangkan sistem kehadiran berbasis *convolutional neural network* (CNN) yang dijalankan pada perangkat *edge* seperti Raspberry Pi dan Jetson Nano, dengan proses inferensi yang dilakukan secara lokal untuk membatasi pergerakan data sensitif. Meskipun demikian, proses pelatihan model pada penelitian tersebut masih dilakukan secara terpusat, sehingga data wajah tetap harus dikumpulkan ke server setiap kali pembaruan model diperlukan. Hal ini menunjukkan bahwa isu privasi pada tahap pelatihan belum sepenuhnya teratasi.

Untuk mengurangi ketergantungan pada pelatihan terpusat, *Federated Learning* diperkenalkan sebagai pendekatan yang memungkinkan pelatihan model dilakukan langsung di perangkat pengguna. Dalam *Federated Learning*, data mentah tidak perlu dikirim ke server pusat; hanya parameter atau pembaruan model yang dikirim untuk digabungkan menjadi satu model global (Staňo et al., 2025) Dengan mekanisme tersebut, privasi data biometrik dapat lebih terjaga dan kebutuhan *bandwidth* menjadi lebih rendah. Penelitian Kim et al. (2021) turut menunjukkan bahwa *Federated Learning* efektif untuk melatih model pengenalan wajah karena citra wajah pengguna tetap berada di perangkat masing-masing. Meskipun temuan-temuan tersebut menunjukkan potensi besar *Federated Learning* untuk aplikasi biometrik, sejauh tinjauan pustaka yang telah dilakukan, belum ditemukan penelitian yang secara spesifik mengimplementasikan *Federated Learning* dengan algoritma FedProx pada sistem kehadiran berbasis pengenalan wajah menggunakan perangkat *edge* heterogen secara nyata. Kondisi ini menyisakan celah penelitian yang relevan untuk dieksplorasi lebih lanjut.

Berdasarkan permasalahan tersebut, penelitian tugas akhir ini mengusulkan penerapan *Federated Learning* pada sistem kehadiran berbasis pengenalan wajah di lingkungan *Edge Computing*. Tahap pengumpulan data dilakukan secara terpusat menggunakan laptop, yaitu dengan merekam video wajah mahasiswa yang kemudian dipotong menjadi kumpulan *frame* dan didistribusikan secara manual ke masing-masing perangkat *edge* sebelum pelatihan dimulai. Penerapan *Federated Learning* dalam penelitian ini mencakup keseluruhan proses, mulai dari tahap *preprocessing* lokal di setiap perangkat *client* hingga proses inferensi kehadiran, sementara model MobileFaceNet dilatih secara lokal pada setiap perangkat tanpa pernah mengirimkan citra wajah ke server pusat. Proses distribusi dan orkestrasi antara server dan *client* dilakukan menggunakan Docker agar sistem lebih mudah dijalankan dan direplikasi. Dengan demikian, kombinasi *Federated Learning*, MobileFaceNet, dan *Edge Computing* diharapkan mampu menghasilkan sistem kehadiran yang lebih aman, efisien, dan tetap menjaga privasi pengguna.

1.2 Rumusan Masalah

Berdasarkan permasalahan tersebut, rumusan masalah yang diangkat pada penelitian ini adalah sebagai berikut:

1. Bagaimana membangun sistem kehadiran berbasis pengenalan wajah yang memanfaatkan konsep *Federated Learning* agar pelatihan model dapat dilakukan secara lokal pada perangkat *edge*?
2. Bagaimana performa model MobileFaceNet yang dilatih menggunakan *Federated Learning* dibandingkan dengan *Centralized Learning* pada sistem kehadiran berbasis pengenalan wajah?

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini berdasarkan rumusan masalah adalah sebagai berikut:

1. Penelitian ini berfokus pada implementasi arsitektur *Federated Learning*, bukan pengembangan arsitektur model baru.
2. Model yang digunakan adalah MobileFaceNet dengan bobot *pre-trained* sebagai *backbone* pengenalan wajah, yang di-*fine-tuning* menggunakan skema *Federated Learning*, bukan dilatih dari awal.
3. Sistem tidak mencakup fitur *anti-spoofing* atau deteksi keaslian wajah.
4. Lingkungan pengujian menggunakan satu unit Raspberry Pi dan satu unit Jetson Nano sebagai perangkat *edge*, serta satu unit VPS sebagai server.
5. Implementasi sistem berbasis *containerization* menggunakan Docker.
6. Objek citra wajah yang digunakan dalam eksperimen dibatasi pada maksimal 30 subjek mahasiswa Departemen Teknologi Informasi.
7. Proses akuisisi data dilakukan menggunakan laptop dengan cara merekam video wajah mahasiswa menggunakan *webcam*, yang kemudian diekstraksi menjadi kumpulan *frame* dan didistribusikan secara manual ke setiap perangkat *edge*.

1.4 Tujuan

Merujuk pada rumusan masalah, tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut:

1. Membangun sistem kehadiran berbasis pengenalan wajah yang memanfaatkan konsep *Federated Learning* agar pelatihan model dapat dilakukan secara lokal pada perangkat *edge*.
2. Membandingkan performa model MobileFaceNet yang dilatih menggunakan *Federated Learning* dengan *Centralized Learning* pada sistem kehadiran berbasis pengenalan wajah.

1.5 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Mempermudah proses absensi dengan sistem pengenalan wajah.
2. Menjaga privasi pengguna karena citra wajah tidak dikirim ke server pusat.
3. Mengurangi kebutuhan *bandwidth* selama proses pelatihan model.

[Halaman ini sengaja dikosongkan]

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Pada penelitian ini digunakan sejumlah penelitian terdahulu sebagai acuan dalam perancangan dan pengembangan sistem yang diusulkan. Penelitian terdahulu dikelompokkan ke dalam dua kategori utama, yaitu penelitian terkait pengembangan *Smart Attendance System* dan penelitian terkait penerapan *Federated Learning* untuk pengenalan wajah. Ringkasan dari kedua kategori tersebut disajikan pada Tabel *2.1* dan Tabel *2.2*.

2.1.1 Penelitian Terkait Pengembangan *Smart Attendance System*

Tabel 2.1 Penelitian Terdahulu Terkait Pengembangan *Smart Attendance System*

No	Penulis (Tahun)	Pembahasan
1	*Smart Attendence Recording System Utilizing CNN and Edge Computing Techniques*	
	Srinu et al. (2025)	Penelitian ini mengimplementasikan sistem pencatatan kehadiran cerdas yang mengintegrasikan *Edge Computing* dengan CNN untuk pemrosesan data biometrik secara lokal. Sistem ini menggunakan model FaceNet yang dijalankan pada perangkat *edge* seperti Jetson dan Raspberry Pi untuk meminimalkan latensi serta mengurangi ketergantungan pada layanan *cloud*. Hasil evaluasi menunjukkan kinerja yang sangat baik dengan tingkat akurasi dan presisi mencapai 98,2%, serta aspek keamanan data yang diperkuat melalui enkripsi AES-256. Peneliti menyarankan penerapan *Federated Learning* di masa mendatang untuk mendesentralisasikan pelatihan model tanpa perlu mentransmisikan data mentah ke server pusat. Rekomendasi tersebut menjadi landasan utama bagi penelitian ini untuk melanjutkan pengembangan sistem menggunakan metode *Federated Learning* guna meningkatkan perlindungan privasi data biometrik mahasiswa.
2	*In-Browser Attendance System using Face Recognition and Serverless Edge Computing*	
	Yadav et al. (2021)	Penelitian ini mengimplementasikan arsitektur *Serverless Edge Computing* yang beroperasi sepenuhnya pada lingkungan *in-browser* guna memitigasi isu latensi dan privasi dalam pembelajaran daring. Sistem dirancang sebagai solusi presensi yang ringan dan *cost-effective* tanpa ketergantungan pada perangkat keras khusus, dengan memanfaatkan pustaka face-api.js dan TensorFlow.js. Pendekatan ini memungkinkan eksekusi model *deep learning*, seperti *Tiny Face Detector* dan *Face Recognition Net*, dilakukan langsung di sisi *client* dengan integrasi Google Sheets untuk manajemen basis data. Hasil evaluasi menunjukkan sistem memiliki performa responsif dengan waktu pemrosesan antara 2 hingga 3 detik dan mampu memvalidasi kehadiran menggunakan *confidence threshold* di atas 65%.

2.1.2 Penelitian Terkait *Federated Learning* untuk Pengenalan Wajah

Tabel 2.2 Penelitian Terdahulu Terkait *Federated Learning* untuk Pengenalan Wajah

No	Penulis (Tahun)	Pembahasan
1	pFedFace: *Personalized Federated Learning for Face Recognition*	
	Gao et al. (2025)	Penelitian ini memperkenalkan kerangka kerja *pFedFace* yang dirancang untuk menyeimbangkan antara generalisasi model global dengan kebutuhan personalisasi lokal pada sistem pengenalan wajah. Inovasi utama yang diusulkan adalah skema *Adaptive Batch Normalization* (BN) yang memisahkan setiap lapisan BN menjadi dua varian: varian generik untuk mencapai konsensus global dan varian spesifik *client* untuk adaptasi domain lokal. Hasil evaluasi pada *benchmark* skala besar menunjukkan bahwa pFedFace berhasil meningkatkan akurasi antara 1,0% hingga 2,68% dibandingkan metode *baseline* seperti FedAvg, serta mencapai akurasi 93,35% pada subset tertentu. Strategi ini diimplementasikan dalam penelitian ini melalui mekanisme pemisahan parameter *batch normalization* pada arsitektur MobileFaceNet. Dengan pendekatan tersebut, proses agregasi hanya dilakukan pada parameter *backbone* model, sementara lapisan BN dikelola secara lokal pada perangkat Raspberry Pi dan Jetson Nano untuk mengoptimalkan akurasi pengenalan identitas mahasiswa secara personal di setiap titik kehadiran.
2	*Federated Edge Learning with Hybrid CNN-Transformers for Secure, High-Accuracy Face Recognition in Resource-Constrained IoT Systems*	
	(Gupta & Singh, 2025)	Penelitian ini mengusulkan sebuah kerangka kerja sistem pengenalan wajah berorientasi IoT yang menggabungkan arsitektur hibrida CNN-Transformer dengan metode *Federated Edge Learning*. Metodologi ini memanfaatkan MobileNetV3 sebagai *backbone* CNN untuk ekstraksi fitur spasial lokal yang kemudian diintegrasikan dengan *transformer encoder* ringan untuk pemodelan konteks wajah secara global. Proses pelatihan dilakukan secara terdesentralisasi antar *node* IoT menggunakan fungsi *ArcFace margin loss* dan algoritma agregasi dengan teknik *gradient sparsification* untuk meminimalkan beban *bandwidth* komunikasi. Hasil evaluasi komprehensif pada *dataset* LFW menunjukkan tingkat akurasi yang tinggi mencapai 98,2% , serta mampu mempertahankan akurasi sebesar 96,7% di bawah kondisi wajah yang terhalang (*occlusion*). Selain itu, ketika diimplementasikan pada perangkat *edge* seperti Raspberry Pi 4 melalui *Quantization-Aware Training* (QAT) INT8, sistem ini berhasil memangkas latensi inferensi hingga menjadi 45 milidetik dan mengurangi konsumsi energi menjadi 290 mJ per inferensi.

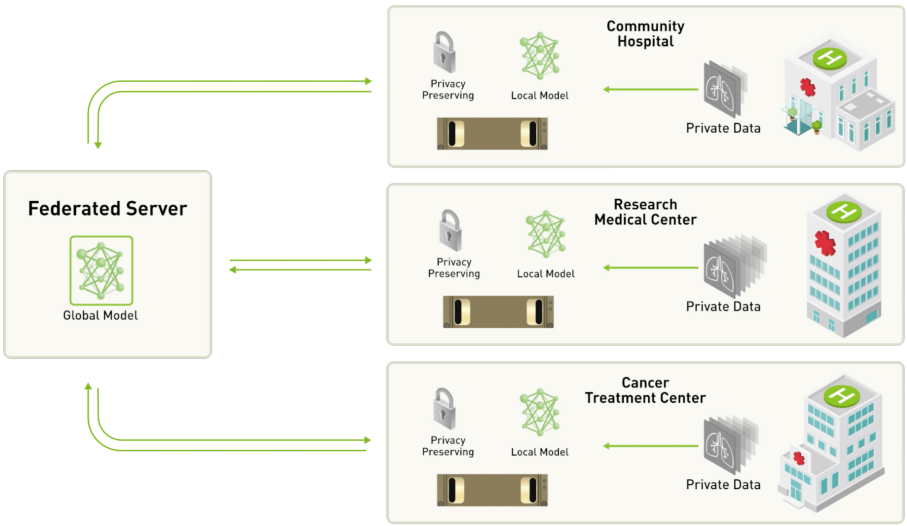
2.2 Dasar Teori

2.2.1 *Federated Learning*

Federated Learning merupakan pendekatan *machine learning* terdesentralisasi yang memungkinkan beberapa perangkat berkolaborasi untuk melatih satu model bersama tanpa mengirimkan data mentah ke server pusat. Setiap *client* melakukan pelatihan model secara lokal dan hanya mengirimkan pembaruan model, seperti parameter atau gradien untuk diagregasi secara terpusat. Pendekatan ini menjaga privasi data sekaligus mengurangi ketergantungan pada penyimpanan terpusat (Shaheen et al., 2022).

Arsitektur umum *Federated Learning* dapat dilihat pada Gambar 2.1. Prosesnya berjalan secara berulang sampai model mencapai konvergensi. Secara umum, terdapat empat tahapan utama dalam, yaitu (Cha & Chang, 2024):

1. Server mengirimkan model global ke seluruh *client*.
2. *Client* melakukan pelatihan model menggunakan data lokal.
3. *Client* mengirimkan pembaruan model ke server.
4. Server melakukan agregasi untuk menghasilkan model global yang diperbarui.



Gambar 2.1 *Federated Learning*
Sumber: (NVIDIA, 2019)

Dibandingkan dengan *Centralized Learning*, *Federated Learning* menawarkan beberapa keunggulan utama, yaitu:

1. Peningkatan privasi karena data tetap berada pada perangkat *edge* masing-masing.
2. Pengurangan beban komunikasi jaringan karena yang dikirim adalah pembaruan model.
3. Mitigasi masalah *data silos*, yaitu kondisi ketika data tersebar pada berbagai perangkat sehingga tidak dapat digabungkan secara langsung untuk pelatihan terpusat.

Federated Learning merupakan konsep utama yang diterapkan dalam penelitian ini untuk memungkinkan proses pelatihan model pengenalan wajah dilakukan secara terdistribusi pada perangkat *edge*. Melalui pendekatan ini, privasi data biometrik tetap terjaga di tingkat lokal, sekaligus mengatasi tantangan heterogenitas data (*Non-IID*) yang umum terjadi pada sistem kehadiran di lingkungan *edge computing*.

Perlu dicatat bahwa implementasi *Federated Learning* dalam penelitian ini berbeda dengan skenario ideal pada literatur, di mana data diasumsikan telah tersedia secara asli di masing-masing perangkat *client*. Pada penelitian ini, pengumpulan data awal dilakukan secara terpusat menggunakan laptop di mana video wajah mahasiswa direkam menggunakan *webcam*, diekstraksi menjadi kumpulan *frame*, lalu didistribusikan secara manual ke setiap perangkat *edge* sebelum pelatihan dimulai. Siklus *Federated Learning* yang sesungguhnya baru berjalan mulai dari tahap *preprocessing* di masing-masing perangkat client, kemudian dilanjutkan dengan pelatihan model lokal, agregasi parameter di server, dan inferensi kehadiran secara *real-time*. Dengan demikian, citra wajah asli tidak pernah meninggalkan perangkat edge selama siklus Federated Learning berlangsung.

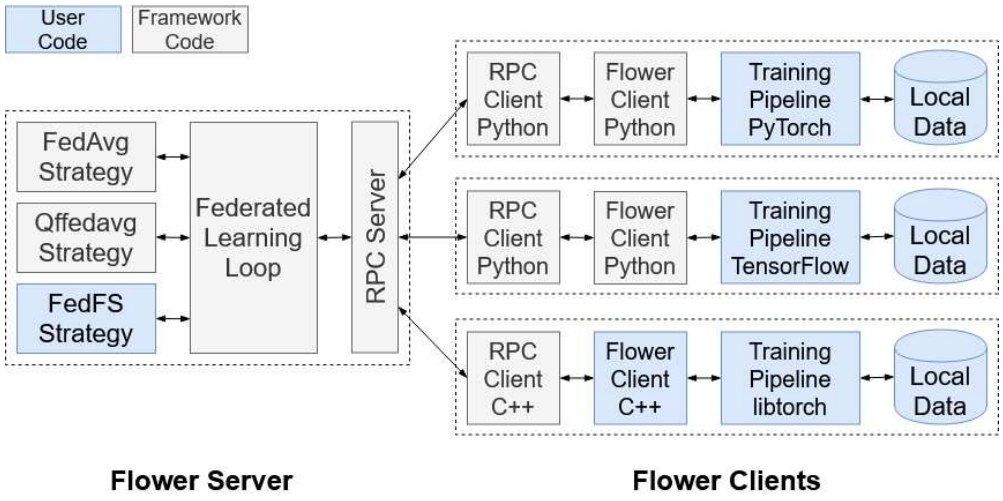
2.2.2 *Centralized Learning*

Centralized Learning merupakan pendekatan pembelajaran mesin konvensional di mana seluruh data yang dikumpulkan dari berbagai sumber harus ditransmisikan dan disimpan dalam satu server pusat untuk menjalani proses pelatihan model. Dalam arsitektur ini, unit pemrosesan terpusat memiliki kendali penuh atas *dataset*, yang memungkinkannya untuk melakukan pengoptimalan model secara global dengan visibilitas data yang menyeluruh (Tertulino, 2025).

Dalam penelitian ini, *Centralized Learning* diterapkan sebagai standar pembanding (*baseline*) untuk mengevaluasi performa model MobileFaceNet, guna mengukur sejauh mana efektivitas dan akurasi yang dihasilkan oleh *Federated Learning* dibandingkan dengan metode pelatihan tradisional tersebut

2.2.3 Flower

Flower adalah salah satu kerangka kerja *Federated Learning* yang dirancang untuk menghubungkan tahap simulasi riset dengan penerapan sistem di perangkat *edge*. Keunggulan utamanya adalah sifatnya yang agnostik, artinya kerangka kerja ini tidak terikat atau bergantung pada satu pustaka *machine learning* tertentu. Pengguna bebas menggunakan *tools* seperti TensorFlow, PyTorch, atau lainnya tanpa masalah kompatibilitas. Fleksibilitas ini sangat memudahkan proses pemindahan kode dari tahap eksperimen ke implementasi dunia nyata tanpa perlu melakukan perombakan besar pada struktur dasarnya (Beutel et al., 2022).



Gambar 2.2 Arsitektur Flower
Sumber: (Mathur et al., 2021)

Arsitektur Flower Seperti ditunjukkan pada Gambar 2.2, arsitektur Flower memisahkan fungsi server sebagai pengatur pusat dan *client* sebagai pelaksana pelatihan di perangkat. Komunikasi antar komponen menggunakan protokol gRPC (*gRPC Remote Procedure Calls*), yang mendukung pertukaran data pada perangkat dengan sumber daya terbatas. Dalam penelitian ini, Flower digunakan untuk menangani komunikasi data.

2.2.4 FedProx

Algoritma *Federated Proximal* (FedProx) hadir sebagai bentuk pengembangan dari metode konvensional *Federated Averaging* (FedAvg) untuk mengatasi kendala heterogenitas pada jaringan terdistribusi. Konsep utama dari FedProx adalah restrukturisasi pada fungsi objektif lokal di setiap perangkat *client* dengan menambahkan sebuah komponen penalti matematis yang disebut *proximal term* (μ). Mekanisme ini dirancang untuk membatasi arah pembaruan parameter selama pelatihan lokal agar tidak melenceng terlalu jauh dari arsitektur model global yang dikelola oleh server pusat. Melalui pendekatan ini, stabilitas dan konsistensi konvergensi model dapat tetap terjaga dengan baik meskipun data yang tersebar di lapangan bersifat tidak seragam (*Non-IID*). Fungsi objektif lokal utama FedProx dirumuskan melalui Persamaan (1) berikut:

$$\frac{min}{w_k}\left[F_k(w_k) + \frac{\mu}{2}\|w_k - w^g\|^2\right] \quad (1)$$

Keterangan:

- w_k: representasi matriks parameter model lokal pada *client* ke-k.
- w^g: bobot model global saat itu.
- $F_k(w_k)$: adalah nilai fungsi *loss* lokal yang dihitung berdasarkan data privat masing-masing perangkat.
- μ: Koefisien regularisasi proximal parameter yang mengontrol batas deviasi bobot model.
- $\|w_k - w^g\|^2$: menyatakan kuadrat jarak L_2 antara bobot lokal dan global.

Proses komputasi pada skema FedProx berjalan melalui tiga tahapan siklus iteratif yang ketat. Tahap pertama dimulai dari *Server-Side Broadcast*, di mana server membagikan parameter model global mutakhir (w^g) kepada seluruh perangkat *client* yang terpilih. Masuk ke tahap kedua, yaitu *Local Optimization*, setiap *client* menghitung nilai *error* dari fungsi *loss* lokal berbasis data privatnya ($F_k(w)$) menggunakan metode rata-rata sampel melalui Persamaan (2) berikut:

$$F_k(w_k) = \frac{1}{n_k}\sum_{xi,yi\in D_k} l(w; xi, yi) \quad (2)$$

Keterangan:

- n_k: Jumlah total sampel data latih yang dimiliki dan diisolasi oleh *client* ke-k.
- D_k: Kumpulan *dataset* lokal pada penyimpanan perangkat *client* ke-k.
- $l(w; xi, yi)$: Fungsi *loss* dasar yang dihitung untuk setiap satu sampel data tunggal.

Berbeda dengan FedAvg konvensional yang membiarkan *client* melakukan optimasi *loss* lokal secara bebas, komponen *proximal term* pada fungsi objektif utama FedProx bertindak sebagai pengekang matematis. Komponen ini memberikan nilai penalti yang semakin besar apabila arah pergeseran bobot lokal (w_k) menjauh dari jangkar model global (w^g).

Setelah iterasi pelatihan lokal selesai, proses memasuki tahap ketiga berupa *Weighted Aggregation*. Seluruh koordinasi parameter bobot yang dikirimkan oleh perangkat *edge* akan dikumpulkan dan diakumulasikan kembali oleh server pusat untuk memperbarui parameter model global pada ronde berikutnya ($w\frac{(t+1)}{global}$) melalui mekanisme rata-rata tertimbang (*weighted averaging*) menggunakan Persamaan (3) berikut:

$$w\frac{(t+1)}{global} = \sum_{K=1}^{K} \frac{n_k}{N} w_k^{(t)} \tag{3}$$

Keterangan:

- k: Jumlah total seluruh perangkat *client* yang berpartisipasi aktif dalam pelatihan.
- N: Akumulasi total volume data dari gabungan seluruh *client* yang terdaftar.
- $w_k^{(t)}$: Matriks parameter bobot lokal hasil latih dari *client* ke-k setelah menyelesaikan ronde ke-t *client*

Penentuan nilai besaran parameter hiperparameter μ (regularisasi *proximal term*) memiliki pengaruh yang sangat sensitif dan berbanding terbalik terhadap stabilitas serta karakteristik konvergensi model di lapangan. Jika nilai μ diatur terlalu besar, tali pengekang model akan menjadi sangat kaku; sistem dipaksa untuk terus menyerupai model global sehingga proses pembaruan bobot lokal menjadi lambat, menghambat model untuk mempelajari variasi fitur wajah baru, dan berisiko memperlama durasi pencapaian konvergensi sistem. Sebaliknya, apabila nilai μ diturunkan hingga mendekati angka nol, sifat bawaan FedProx akan luruh dan sistem kembali beroperasi layaknya algoritma FedAvg biasa. Dalam kondisi data lokal yang tidak seragam (*Non-IID*), nilai μ yang terlalu kecil akan membuat setiap perangkat *edge* memperbarui parameter secara agresif demi meminimalkan *loss* lokalnya sendiri, yang berujung pada fenomena *client drift* (pergeseran bobot lokal yang saling bertolak belakang), perlambatan akurasi global, hingga risiko kegagalan latih (*divergence*). Oleh karena itu, melalui proses *hyperparameter tuning* yang sekuensial, nilai μ harus dikunci pada angka optimal guna menyeimbangkan fleksibilitas adaptasi lokal di setiap *node edge* tanpa mengorbankan stabilitas akurasi model global (Xiong et al., 2025).

Dalam penelitian sistem kehadiran mahasiswa ini, FedProx diterapkan sebagai algoritma agregasi utama untuk mengorkestrasi pembaruan parameter model dari berbagai perangkat keras *edge* secara aman dan konsisten.

2.2.5 pFadFace

pFedFace adalah kerangka kerja *Personalized Federated Learning* yang dirancang khusus untuk mengatasi keterbatasan metode *Federated Learning* konvensional dalam pengenalan wajah, di mana metode konvensional hanya berfokus pada pembentukan representasi wajah secara global tanpa mempertimbangkan kebutuhan personalisasi lokal masing-masing *client* Gao et al. (2025). Inovasi utamanya adalah skema *Adaptive Batch Normalization* (BN) yang

menduplikasi setiap lapisan BN menjadi dua varian. Varian pertama bersifat generik dan berperan menyatukan pengetahuan antar *client* untuk generalisasi global, sedangkan varian kedua bersifat spesifik per *client* dan bertugas menjembatani perbedaan domain antar perangkat untuk personalisasi lokal. Selain itu, pFedFace memperkenalkan *personalized discriminative loss* yang mendorong model lokal untuk mengekstraksi fitur wajah secara lebih diskriminatif dibandingkan model global.

Dalam penelitian ini, strategi pFedFace diterapkan melalui mekanisme pemisahan parameter *Batch Normalization* pada arsitektur MobileFaceNet. Secara praktis, pada saat proses agregasi berlangsung, hanya bobot lapisan konvolusi (*backbone*) yang dikirim ke server dan diagregasi menjadi model global, sementara lapisan BN beserta statistiknya (*running mean* dan *running variance*) tidak ikut dikirim dan tetap dikelola secara lokal di masing-masing perangkat edge, yaitu Raspberry Pi dan Jetson Nano. Dengan cara ini, setiap perangkat dapat mempertahankan kalibrasi statistik normalisasinya terhadap kondisi kamera dan pencahayaan lokalnya masing-masing, sehingga akurasi inferensi identitas mahasiswa di setiap titik kehadiran dapat dioptimalkan secara personal tanpa mengorbankan pengetahuan global yang diperoleh melalui agregasi.

2.2.6 *Face Recognition*

Face Recognition atau pengenalan wajah adalah teknik biometrik yang digunakan untuk mengidentifikasi atau memverifikasi seseorang berdasarkan fitur wajah yang diproses melalui model *deep learning*. Kemajuan *deep learning* membuat akurasinya meningkat, tetapi pelatihan model biasanya membutuhkan banyak data wajah yang bersifat sensitif dan sulit dibagikan. Hal ini menjadikan pengenalan wajah sebagai bidang yang membutuhkan pendekatan pembelajaran yang aman dan terdistribusi (Kim et al., 2021)

Dalam penelitian ini, pengenalan wajah digunakan sebagai inti sistem absensi. Proses identifikasi dijalankan langsung pada perangkat *edge*. Dengan cara ini, sistem dapat bekerja lebih cepat dan tetap menjaga kerahasiaan data biometrik.

2.2.7 MobileFaceNet

MobileFaceNet merupakan arsitektur *deep learning* yang dirancang untuk pengenalan wajah pada perangkat dengan sumber daya terbatas. Model ini termasuk jaringan ringan (*lightweight CNN*) yang fokus pada efisiensi komputasi tanpa mengorbankan akurasi secara signifikan. MobileFaceNet menggunakan teknik seperti *depthwise separable convolution* yang membuat ukuran model kecil namun tetap mampu mengekstraksi fitur wajah dengan baik (Xiao et al., 2024).

Dalam penelitian ini, MobileFaceNet diimplementasikan sebagai model utama untuk pengenalan wajah pada sistem kehadiran. Penelitian ini menerapkan strategi *transfer learning* dengan memanfaatkan bobot awal (*pre-trained weights*) yang bersumber dari implementasi *open-source* MobileFaceNet berbasis PyTorch oleh Xiaoccer (2023) yang dilatih mengunakan *dataset* CASIA-WebFace dan dievaluasi menggunakan *dataset* LFW (*Labelled Faces in the Wild*). Penggunaan bobot awal ini berfungsi sebagai inisialisasi dasar untuk mempercepat konvergensi sebelum dilakukan proses *Federated Learning*. Model ini dipilih karena karakteristiknya yang ringan, sehingga mampu beroperasi secara optimal dan efisien pada perangkat *edge* yang memiliki keterbatasan sumber daya komputasi.

2.2.8 *Face Embedding*

Face Embedding adalah proses ekstraksi fitur biometrik di mana citra wajah mentah ditransformasikan oleh jaringan saraf tiruan mendalam menjadi sebuah vektor numerik berdimensi tinggi yang unik. Vektor ini merepresentasikan karakteristik geometris wajah individu secara abstrak, sehingga proses pengenalan identitas tidak lagi membandingkan gambar piksel demi piksel, melainkan cukup dengan menghitung kedekatan jarak matematis antar vektor embedding tersebut. Meskipun representasi data ini lebih ringkas dibandingkan citra asli (Shahreza et al., 2025).

Dalam penelitian ini, *face embedding* yang dihasilkan oleh arsitektur MobileFaceNet digunakan sebagai representasi data utama untuk melakukan verifikasi pada sistem kehadiran mahasiswa.

2.2.9 MTCNN

MTCNN atau *Multi-task Convolutional Neural Network* adalah model jaringan saraf *multi-task* yang dirancang untuk melakukan deteksi wajah dengan menyeimbangkan kinerja dan akurasi. Algoritma ini menggunakan struktur jaringan berjenjang yang terdiri dari tiga lapisan, yaitu P-Net (*Proposal Network*), R-Net (*Refine Network*), dan O-Net (*Output Network*). Mekanisme kerjanya dimulai dengan menghasilkan kandidat kotak area target menggunakan model kecil, yang kemudian disaring secara bertahap menggunakan model yang lebih kompleks untuk mendapatkan klasifikasi dan regresi kotak pembatas wajah yang presisi (Zhang et al., 2020).

Dalam penelitian ini, MTCNN digunakan sebagai metode utama pada tahap *preprocessing* untuk mendeteksi dan *cropping* area wajah dari citra asli. Kemampuan MTCNN dalam menyaring kandidat area wajah (*bounding box*) digunakan untuk memastikan bahwa input yang akan dimasukkan ke dalam model MobileFaceNet hanyalah bagian wajah yang bersih dan akurat.

2.2.10 *Edge Computing*

Edge Computing adalah arsitektur komputasi terdistribusi di mana pemrosesan data dilakukan secara lokal, dekat dengan sumber datanya. Karena tidak semua data dikirim ke *cloud*, pendekatan ini bisa mengurangi latensi, menghemat *bandwidth*, dan memungkinkan aplikasi yang butuh respon cepat berjalan lebih efisien (Brecko et al., 2022).

Penelitian ini memanfaatkan *edge computing* sebagai dasar perancangan sistem, Pemrosesan dilakukan langsung pada perangkat *edge*, sehingga identifikasi wajah bisa berlangsung secara lokal.

2.2.11 Docker

Docker adalah sebuah platform *containerization* yang memungkinkan aplikasi dijalankan dalam lingkungan terisolasi yang disebut container. Setiap container membawa aplikasi beserta dependensinya, sehingga aplikasi dapat berjalan konsisten di berbagai perangkat tanpa konflik konfigurasi (Nandi et al., 2023).

Dalam penelitian ini, Docker digunakan untuk menjalankan dan mengelola komponen aplikasi dalam lingkungan yang seragam. Penggunaan *container* membantu mengurangi perbedaan konfigurasi antarperangkat dan mempermudah proses pengembangan serta implementasi.

2.2.12 AES-256

AES-256 adalah varian dari standar enkripsi simetris *Advanced Encryption Standard* (AES) yang ditetapkan oleh NIST, yang beroperasi menggunakan panjang kunci 256-bit. Algoritma ini memproses data dalam blok 128-bit melalui 14 *rounds* enkripsi, jumlah yang lebih banyak dibandingkan varian standar 128-bit (Gonzalez et al., 2023).

Dalam penelitian ini, AES-256 diterapkan sebagai mekanisme keamanan untuk melindungi data yang tersimpan pada perangkat *edge*. Algoritma ini digunakan untuk mengenkripsi vektor *face embeddings* hasil ekstraksi sebelum disimpan ke dalam basis data lokal. Langkah ini memastikan bahwa data biometrik mahasiswa tetap aman dan tidak dapat dibaca oleh pihak yang tidak berwenang.

2.2.13 PostgreSQL

PostgreSQL adalah sistem manajemen basis data relasional objek (ORDBMS) bersifat *open-source* yang dikenal karena keandalan dan ketangguhannya dalam menangani data yang kompleks. Sistem ini menggunakan dan memperluas bahasa SQL standar untuk menyimpan serta mengelola beban kerja data dengan aman. Pengembangan PostgreSQL bermula dari proyek POSTGRES di University of California, Berkeley yang dimulai pada tahun 1986 (*About PostgreSQL.*, n.d.).

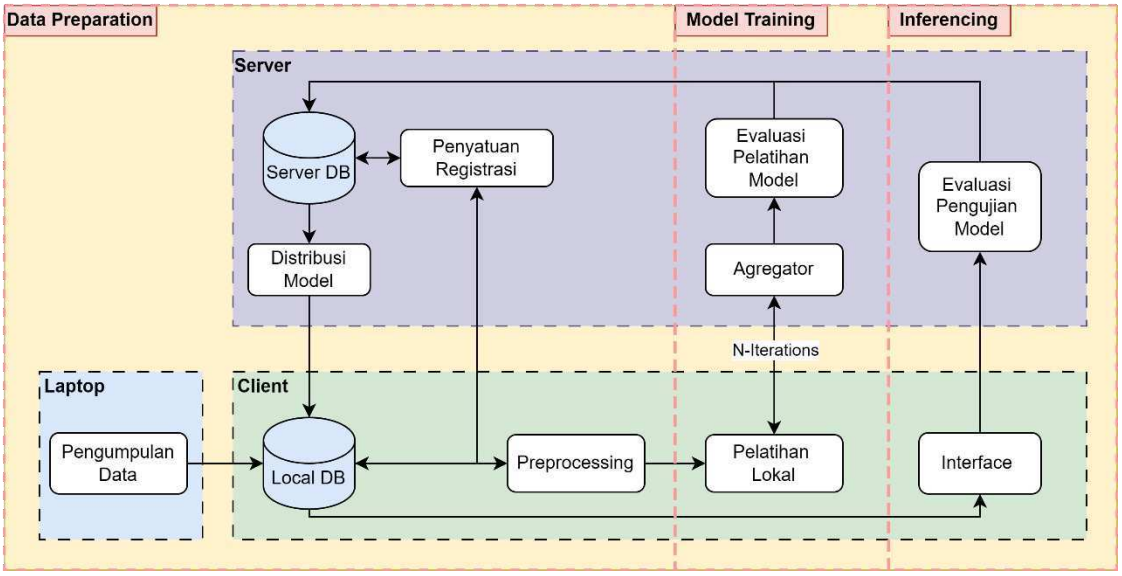
Dalam penelitian ini, PostgreSQL digunakan pada perangkat *edge* sebagai tempat penyimpanan data utama. Seperti menyimpan *embedding* milik pengguna, serta mencatat riwayat waktu kehadiran mereka.

[Halaman ini sengaja dikosongkan]

BAB 3 METODOLOGI

3.1 Metode yang digunakan

Penelitian ini menerapkan metode *Federated Learning* pada lingkungan *Edge Computing* sebagai pendekatan utama. Secara umum, alur penelitian dari awal hingga akhir ditunjukkan pada Gambar 3.1 berikut ini.

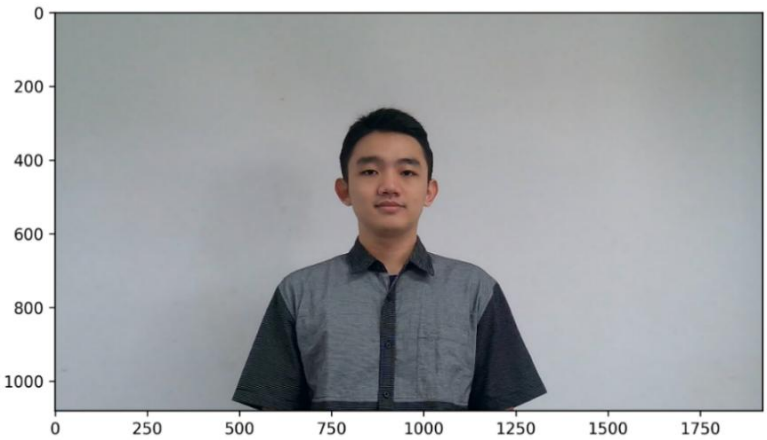


Gambar 3.1 Diagram Alur Penelitian

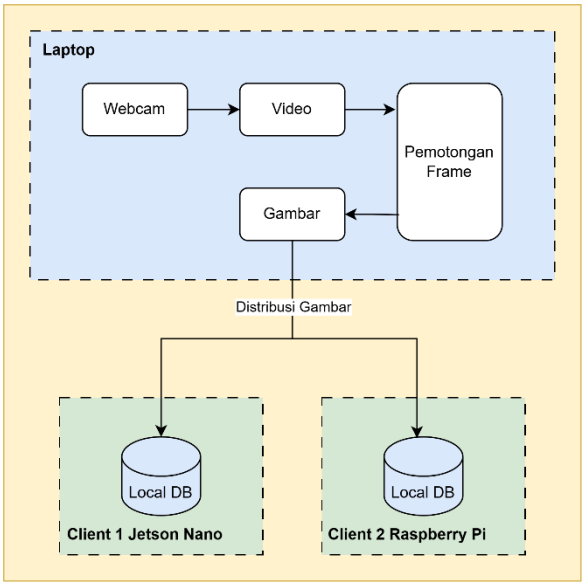
3.1.1 *Data Preparation*

1. Pengumpulan Data

Langkah pertama adalah merekam video wajah dari 30 mahasiswa menggunakan *webcam*. Video tersebut kemudian diekstraksi secara otomatis menjadi kumpulan *frame* dengan interval pengambilan setiap 0,17 detik, sebagaimana ditunjukkan pada contoh hasil citra mentah pada Gambar 3.2. Guna mensimulasikan skenario *Federated Learning* yang mendekati kondisi nyata, seluruh data tidak disimpan pada satu lokasi terpusat, melainkan didistribusikan ke beberapa perangkat *edge* sehingga masing-masing perangkat menyimpan data dari 15 mahasiswa yang berbeda, sebagaimana diilustrasikan pada Gambar 3.3.



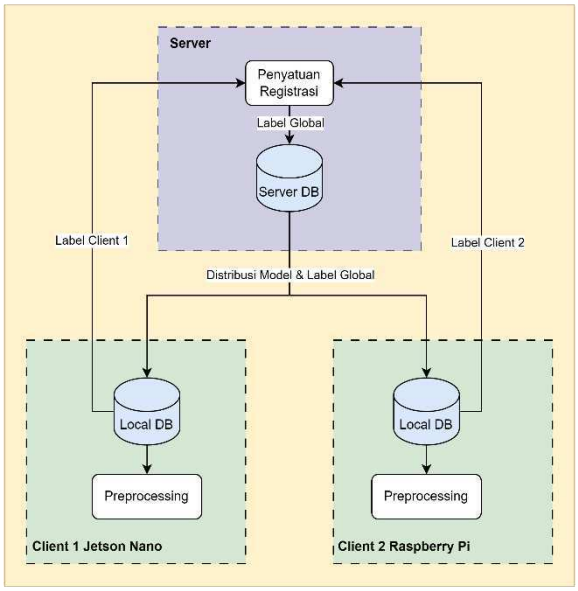
Gambar 3.2 Sampel Citra Mentah Hasil Ekstraksi *Frame*



Gambar 3.3 Alur Proses Pengumpulan dan Distribusi Data

2. Distribusi Model

Perlu ditegaskan bahwa proses pelatihan pada tahap Federated Learning ini bukan merupakan pelatihan model dari awal, melainkan proses *fine-tuning* di atas bobot yang telah dilatih sebelumnya (*pre-trained weights*), sebagaimana dijelaskan pada Subbab 2.2.7. Server mendistribusikan bobot awal ini ke setiap perangkat client untuk memastikan seluruh perangkat memiliki titik awal yang seragam, sebelum masing-masing menjalankan pelatihan lokal menggunakan data 15 mahasiswa yang dikelolanya. Dengan kata lain, proses Federated Learning pada penelitian ini berfungsi sebagai tahap adaptasi/personalisasi domain terhadap fitur wajah subjek penelitian, bukan pembentukan representasi fitur wajah dari nol. Proses distribusi model ini berjalan beriringan dengan proses penyatuan registrasi dan penempatan tahap preprocessing pada masing-masing *client*, sebagaimana diilustrasikan pada Gambar 3.4.



Gambar 3.4 Alur Distribusi Model, Penyatuan Registrasi, dan *Preprocessing* pada *Client*

3. Penyatuan Registrasi

Setiap perangkat *client* mendaftarkan identitas 15 mahasiswanya ke server pusat. Server kemudian menyusun satu daftar label terpadu (label global) yang mencakup seluruh 30 mahasiswa sehingga tidak terjadi tumpang tindih penomoran label antar-perangkat. Label global tersebut disimpan pada *database* server dan selanjutnya didistribusikan kembali ke masing-masing *client* bersamaan dengan bobot model, sebagaimana ditunjukkan pada Gambar 3.4. Daftar terpadu ini menjadi acuan utama dalam proses pengelompokan citra pada tahap *preprocessing*.

4. *Preprocessing*

Berdasarkan daftar label yang telah terbentuk, setiap citra wajah diproses secara sekuensial pada masing-masing perangkat *client* untuk memenuhi format masukan model, sebagaimana ditunjukkan pada blok *Preprocessing* di Gambar 3.4. Tahapannya adalah sebagai berikut:

a. Seleksi *Laplacian Variance*

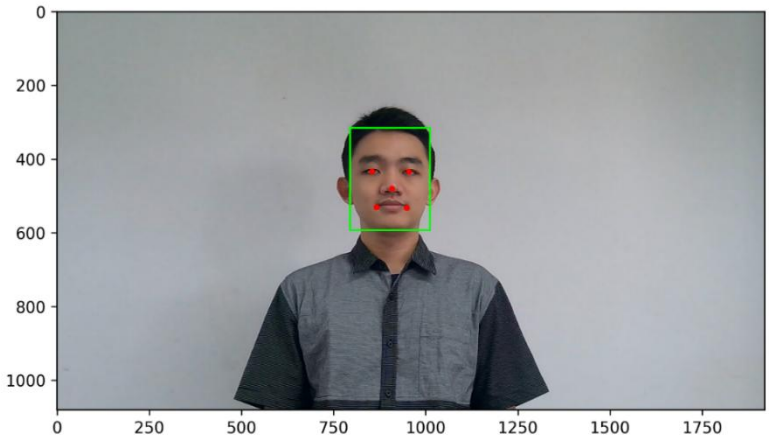
Sistem mengukur ketajaman setiap *frame* menggunakan operator Laplacian yang bekerja dengan mendeteksi gradien tepi objek dalam citra. *Frame* yang memiliki nilai ketajaman rendah akibat gerakan subjek maupun kondisi pencahayaan yang kurang optimal akan disaring secara otomatis. Dari seluruh *frame* yang tersedia, dipilih 50 *frame* dengan nilai ketajaman tertinggi untuk diteruskan ke tahap berikutnya. Perbandingan visual antara citra tajam dan citra buram dapat dilihat pada Gambar 3.5.

Gambar 3.5 Hasil Penyaringan Ketajaman Citra Menggunakan *Laplacian Variance*

b. *Face Detection* dan *Cropping*

Dari 50 *frame* terpilih, model MTCNN melokalisasi wajah secara otomatis dengan menghasilkan koordinat kotak pembatas (*bounding box*) di sekitar area wajah sekaligus mengekstraksi posisi 5 titik penanda wajah (*facial landmark*), yaitu mata kiri, mata kanan, hidung, sudut mulut kiri, dan sudut mulut kanan. MTCNN bekerja melalui tiga tahap jaringan konvolusi bertingkat, yaitu *Proposal Network* (P-Net) untuk menghasilkan kandidat area wajah secara cepat, *Refine Network* (R-Net) untuk menyaring kandidat yang kurang akurat, serta *Output Network* (O-Net) untuk menghasilkan koordinat kotak pembatas dan titik penanda

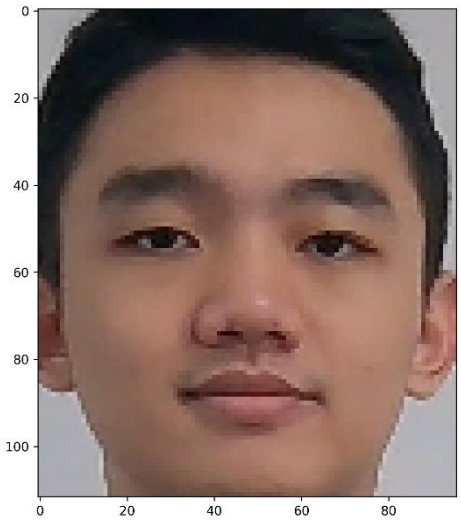
wajah secara presisi. Koordinat kotak pembatas yang dihasilkan digunakan untuk memotong (*cropping*) citra secara presisi sehingga hanya area wajah yang dipertahankan, sementara latar belakang dan bagian tubuh lain di luar wajah dihilangkan. Adapun kelima titik penanda wajah yang diperoleh pada tahap ini akan dimanfaatkan pada proses penyelarasan (*alignment*) wajah di tahap berikutnya, sebagaimana ditunjukkan pada Gambar 3.6.



Gambar 3.6 Deteksi *Bounding Box* dan Titik Landmarks Wajah Menggunakan MTCNN

c. *Resizing*

Citra wajah hasil pemotongan diubah ukurannya menjadi 96×112 piksel karena arsitektur MobileFaceNet hanya dapat menerima citra dengan dimensi tersebut sebagai masukan lapisan konvolusi pertamanya. Hasil normalisasi resolusi ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 Hasil Normalisasi Citra Wajah Menjadi Resolusi 96x112 Piksel

d. *Normalization*

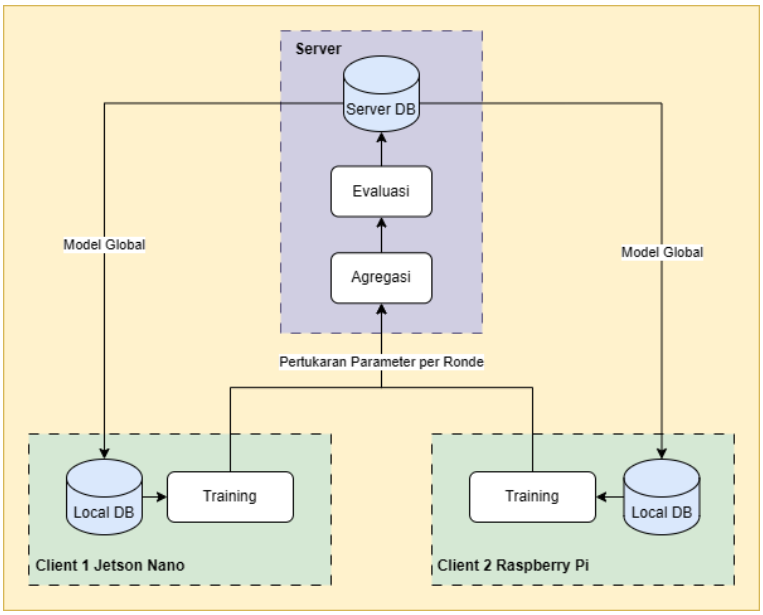
Nilai piksel citra dinormalisasi agar berada pada rentang yang konsisten. Citra dikonversi ke format *tensor* multidimensi, kemudian setiap nilai piksel dikurangi nilai rata-rata sebesar 0,5 dan dibagi dengan standar deviasi sebesar 0,5. Normalisasi ini bertujuan menstabilkan distribusi gradien sehingga proses pelatihan model berlangsung lebih stabil dan konvergen lebih cepat.

e. Enkripsi

Vektor *face embedding* yang dihasilkan model dienkripsi sebelum disimpan ke database lokal menggunakan algoritma AES-256 dalam mode CBC. Dengan demikian, meskipun perangkat *edge* berhasil diakses secara fisik oleh pihak tidak berwenang, data biometrik yang tersimpan tidak dapat dibaca maupun direkonstruksi kembali tanpa kunci enkripsi yang valid.

3.1.2 *Model Training*

Proses pelatihan dimulai setelah semua citra wajah di setiap perangkat *edge* selesai dibersihkan, dipotong, dan diamankan. Tujuan dari tahap ini adalah melatih model MobileFaceNet secara kolaboratif agar mampu mengenali wajah mahasiswa tanpa harus mengirimkan data ke server. Proses ini berjalan berulang dalam beberapa ronde menggunakan algoritma FedProx dengan alur sebagaimana ditunjukkan pada Gambar 3.8, dengan langkah-langkah sebagai berikut:



Gambar 3.8 Alur Pelatihan Model pada *Federated Learning*

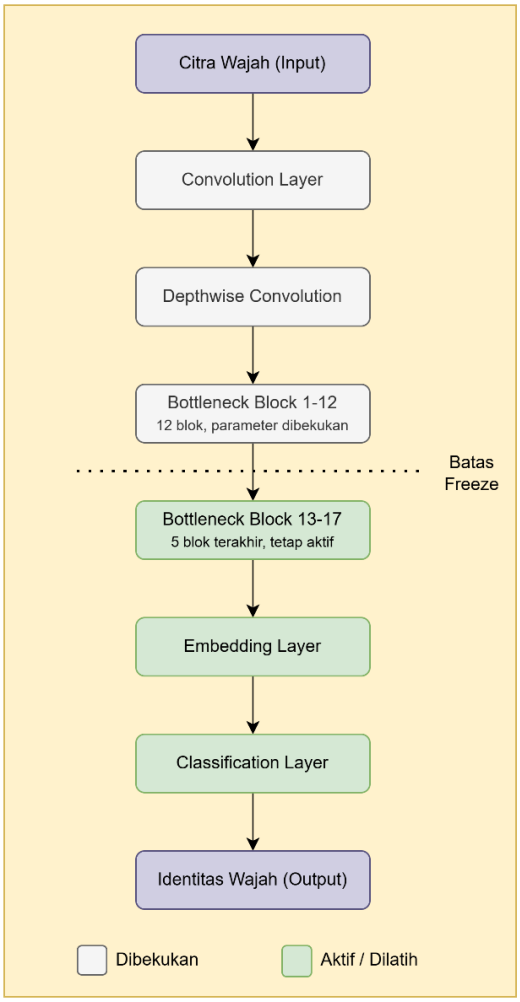
1. Pelatihan Lokal

Tahap ini merupakan fase eksekusi komputasi mandiri di sisi perangkat *client* menggunakan data lokal terisolasi dari 15 mahasiswa yang dikelolanya. Alur pengerjaan pada fase lokal ini dibagi ke dalam sub-proses berikut:

a. Konfigurasi *Layer Freezing*

Sebelum pelatihan dimulai, sebagian lapisan awal *backbone* MobileFaceNet dibekukan sehingga parameternya tidak ikut diperbarui selama pelatihan. Pembekuan ini menonaktifkan kalkulasi gradien pada lapisan-lapisan terkait, yang secara langsung mengurangi konsumsi memori komputasi pada perangkat *edge*. Pendekatan pembekuan sebagian lapisan (*partial layer freezing*) dengan rasio tertentu telah digunakan pada penelitian *transfer learning* berbasis CNN sebelumnya, di mana rasio pembekuan 25%, 50%, dan 75% diuji secara sistematis untuk menentukan konfigurasi *fine-tuning* yang optimal pada beberapa arsitektur CNN ringan (Yurdakul et al., n.d.). Efektivitas pembekuan pada penelitian tersebut ditemukan bergantung

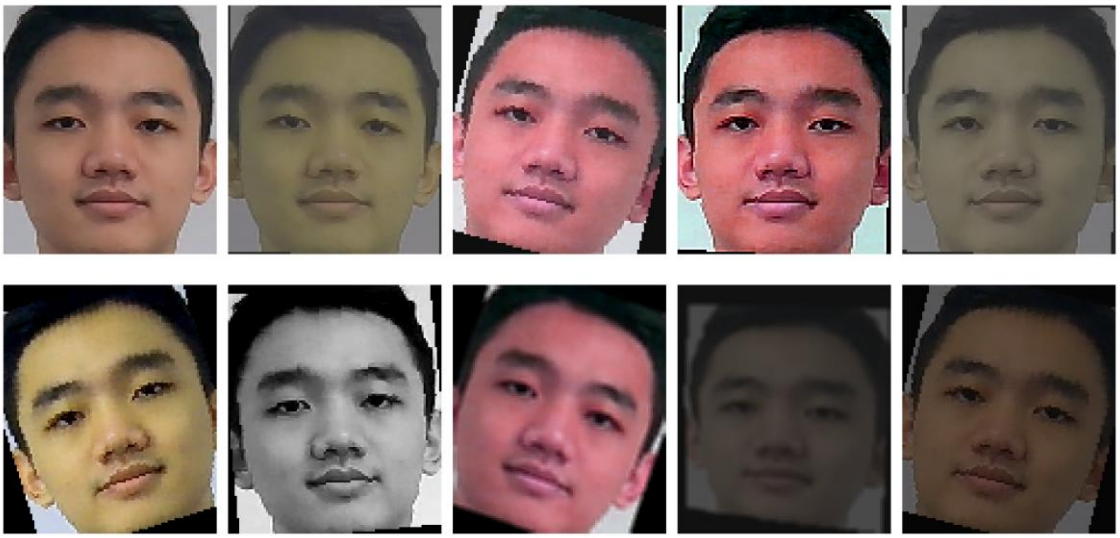
pada tingkat kemiripan antara domain data sumber dan domain data target. Semakin rendah kemiripannya, semakin besar kebutuhan untuk melatih ulang lapisan-lapisan awal. Karena MobileFaceNet dalam penelitian ini telah dilatih sebelumnya pada *dataset* wajah berskala besar dan diterapkan kembali pada tugas pengenalan wajah, sehingga memiliki tingkat kemiripan domain yang tinggi, pembekuan pada proporsi besar lapisan awal (14 dari 19 lapisan, sekitar 74%) dinilai sesuai untuk diterapkan. Dari total 19 lapisan utama MobileFaceNet yang terdiri dari 1 *convolution layer*, 1 *depthwise convolution*, dan 17 *bottleneck block*, sistem membekukan 14 lapisan pertama. Lapisan yang tetap aktif dan dilatih meliputi 5 *bottleneck block* terakhir, *embedding layer*, serta *classification layer*, diilustrasikan pada Gambar 3.9.



Gambar 3.9 Konfigurasi *layer freezing* pada *backbone* MobileFaceNet

b. Augmentasi Data *On-The-Fly*

Setiap citra wajah dimodifikasi secara acak pada saat pemuatan data tanpa mengubah berkas aslinya di penyimpanan fisik. Teknik augmentasi yang diterapkan meliputi pembalikan horizontal acak untuk mensimulasikan variasi sudut pandang wajah, perubahan acak kecerahan dan kontras untuk meniru fluktuasi intensitas cahaya ruangan, rotasi acak untuk mensimulasikan kemiringan kepala subjek, pemotongan dan penskalaan ulang acak untuk mensimulasikan variasi jarak terhadap kamera, serta konversi ke skala abu-abu secara acak untuk melatih ketahanan ekstraksi fitur pada kondisi minim warna. Variasi citra hasil augmentasi ini ditunjukkan pada Gambar 3.10.



Gambar 3.10 Variasi Sampel Hasil Transformasi *On-the-Fly Data Augmentation*

c. *Hiperparameter* dan Optimasi Lokal Dengan *FedProx Proximal Term*

Pelatihan lokal pada setiap perangkat *client* dikonfigurasi menggunakan algoritma optimasi *Stochastic Gradient Descent* (SGD) dengan *Nesterov momentum* untuk mempercepat konvergensi dan menstabilkan pembaruan bobot. Laju pembelajaran (*learning rate*) diturunkan secara bertahap mengikuti pola *Cosine Decay* agar pelatihan tetap stabil menjelang akhir proses. Fungsi kerugian yang diterapkan adalah *ArcFace Margin Loss*, yang dirancang khusus untuk pengenalan wajah dengan mempertegas separasi antar kelas dalam ruang fitur. Di samping itu, ditambahkan komponen penalti *proximal term* dari algoritma *FedProx* yang berfungsi membatasi deviasi parameter model lokal terhadap model global acuan. Penalti ini penting dalam skenario distribusi data yang tidak merata antar-perangkat karena tanpa mekanisme ini, model pada setiap *client* berpotensi berkembang ke arah yang divergen sehingga sulit diagregasikan. Rincian nilai parameter yang digunakan ditunjukkan pada Tabel 3.1.

Tabel 3.1 Parameter Pelatihan Lokal pada *Federated Learning*

Parameter	**Federated Learning**
Arsitektur *Backbone*	MobileFaceNet (128-dim)
Optimizer	SGD
Momentum / Nesterov	0,9 / Aktif
Batch size	32
Learning rate awal	0,05
Learning rate scheduler	Cosine Decay
Loss function	*ArcFace Margin Loss*
Proximal term (μ)	0,05
Partial Freezing	14 dari 19 lapisan dibekukan

d. Ekstraksi Parameter dan *Head Centroid*

Setelah satu ronde pelatihan lokal selesai, sistem mengambil dua komponen dari model, yaitu bobot lapisan konvolusi yang sudah diperbarui dan *head centroid weights* yang merepresentasikan profil rata-rata setiap identitas mahasiswa di lapisan klasifikasi. Kedua komponen ini dikemas menjadi satu paket data yang siap dikirim ke server.

e. Transmisi Parameter

Paket parameter tersebut dikirimkan ke server pusat melalui jaringan lokal. Yang ditransmisikan hanyalah nilai bobot model hasil pelatihan, bukan citra wajah mahasiswa dalam bentuk apapun. Mekanisme ini menjamin bahwa data biometrik asli tetap tersimpan secara eksklusif di dalam perangkat *edge* sepanjang siklus komunikasi berlangsung..

2. Agregator

Setelah semua *client* mengirimkan hasil pelatihan lokalnya, server bertugas menggabungkan seluruh parameter tersebut menjadi satu model global. Proses ini terdiri dari tiga langkah berikut:

a. Penyelarasan Pengumpulan Parameter

Server mengumpulkan semua bobot model dan *head centroid weights* dari setiap *client* yang aktif pada ronde tersebut, lalu menyelaraskan strukturnya berdasarkan indeks label yang sudah disepakati bersama.

b. Penggabungan (Agregasi)

Server menghitung rata-rata tertimbang dari seluruh bobot yang diterima. *Client* dengan volume data latih lebih besar memperoleh bobot kontribusi yang lebih besar dalam perhitungan. Hasilnya adalah satu model global baru yang memiliki representasi fitur biometrik dari seluruh 30 mahasiswa secara kolektif tanpa sekalipun menerima citra wajah dari perangkat manapun.

c. Redistribusi Model Global

Model lobal yang baru disebarkan kembali ke semua *client* sebagai titik awal pelatihan ronde berikutnya. Siklus ini terus berulang hingga jumlah ronde yang ditargetkan tercapai.

3. Evaluasi Pelatihan Model

Tahap evaluasi ini bertujuan untuk mengkomparasi model hasil pelatihan desentralisasi berbasis algoritma FedProx terhadap model tradisional *Centralized Learning* yang bertindak sebagai standar tolok ukur (*baseline*). Evaluasi dilakukan secara komprehensif dengan meninjau dua aspek utama sebagai berikut:

a. Evaluasi Performa Pelatihan

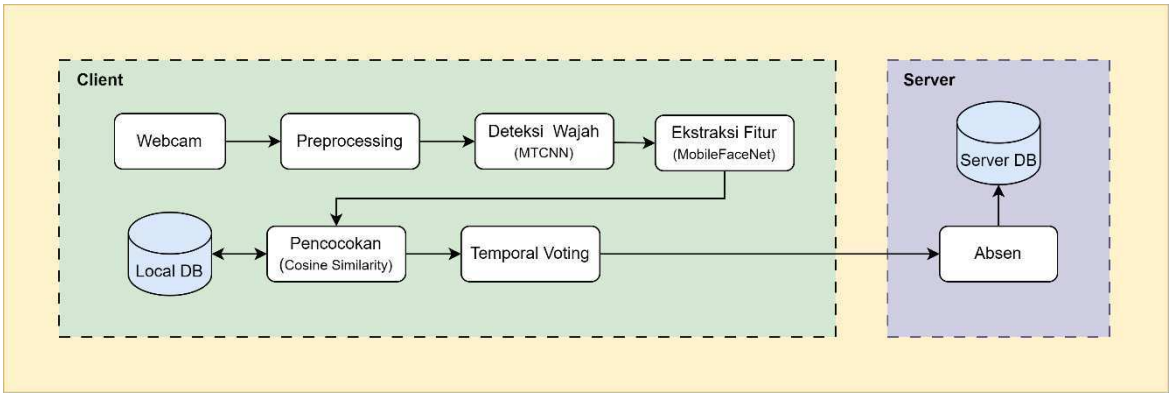
Aspek ini difokuskan untuk menilai karakteristik pembaruan model pada masing-masing arsitektur yang diuji. Pengujian dilakukan dengan cara mencatat seluruh pergerakan nilai *loss*, *accuracy*, *validation loss*, dan *validation accuracy* ke dalam tabel evaluasi pada setiap siklus putaran latih (ronde pada *Federated Learning* dan *epoch* pada *Centralized Learning*). Melalui representasi angka pada tabel tersebut, analisis dilakukan untuk melihat kestabilan model dalam mengenali fitur wajah mahasiswa.

b. Evaluasi Fisik dan Efisiensi Sumber Daya

Aspek ini ditujukan untuk menilai beban penggunaan infrastruktur komputasi riil dari kedua pendekatan sistem. Pengukuran dilakukan secara berurutan meliputi total durasi waktu latih sistem dalam satuan detik, volume transmisi data koordinasi parameter model yang dihabiskan melewati jaringan dalam satuan Megabyte (MB), serta akumulasi total konsumsi energi listrik perangkat keras yang dihabiskan selama operasional komputasi berlangsung dalam satuan kiloWatt-hour (kWh). Riset ini membatasi analisis objektif pada parameter fisik mutlak tersebut dan mencatatnya ke dalam tabel ringkasan.

3.1.3 *Inferencing*

Tahap *model integration* merupakan fase akhir dari perancangan sistem, di mana arsitektur model MobileFaceNet yang telah terlatih diintegrasikan ke dalam subsistem antarmuka aplikasi untuk mengeksekusi layanan verifikasi kehadiran mahasiswa secara langsung (*on-device*) pada perangkat *edge*, sebagaimana ditunjukkan pada Gambar 3.11. Rangkaian prosedur integrasi dan pengujian operasional ini dibagi ke dalam 2 poin utama sebagai berikut:



Gambar 3.11 Alur Inferensi pada Perangkat *Client* dan Server

1. *Interface*

Subsistem ini berfungsi sebagai komponen utama interaksi *real-time* untuk menjembatani proses akuisisi data visual pengguna dengan database lokal perangkat *edge*. Siklus eksekusi dimulai ketika aplikasi mengakses perangkat webcam secara lokal untuk menangkap aliran video wajah mahasiswa secara langsung. Aliran data visual tersebut diekstrak per *frame* untuk mendeteksi keberadaan objek wajah menggunakan MTCNN yang telah terpasang, sebelum akhirnya diumpankan menuju arsitektur MobileFaceNet global mutakhir hasil pelatihan desentralisasi. Model kemudian mengekstraksi karakteristik geometri wajah tersebut menjadi sebuah *face embeddings*, yang langsung dihitung tingkat jarak kedekatannya terhadap matriks *head centroid* referensi mahasiswa di database lokal menggunakan metode *Cosine Similarity*. Guna mengeliminasi fluktuasi salah deteksi akibat pergerakan dinamis subjek di lapangan, sistem menerapkan jaring pengaman *Temporal Voting* yang merata-ratakan akumulasi skor kemiripan dari rentang 3 *frame* terakhir yang tertangkap, sebelum akhirnya mengeluarkan keputusan identitas akhir berupa status dikenal (*KNOWN* beserta nomor registrasi mahasiswa) atau tidak dikenal (*UNKNOWN*).

2. Evaluasi Pengujian Model

Fase evaluasi ini ditujukan untuk menguji keandalan operasional sistem absensi secara biner dengan parameter keputusan berupa status Berhasil atau Gagal mendasarkan pada metodologi pengujian Tugas Akhir terdahulu (Nagamas, 2022). Pengujian fungsional pertama dilakukan melalui serangkaian skenario pengujian terkontrol yang memetakan performa pengenalan wajah berdasarkan variasi kondisi lingkungan fisik, yaitu jarak pemindaian (0,5 meter; 1 meter; 1,5 meter; 2 meter; 2,5 meter; dan 3 meter) serta tingkat intensitas pencahayaan ruangan (cahaya cukup dan pencahayaan minim). Status pengujian dinyatakan Berhasil apabila sistem antarmuka pada perangkat *edge* mampu mendeteksi wajah, melakukan ekstraksi vektor fitur, dan mengeluarkan label keputusan identitas secara tepat (*KNOWN* beserta nama/NRP mahasiswa) pada variasi jarak dan kondisi pencahayaan tersebut, serta memiliki kemampuan untuk mengenali identitas mahasiswa lintas *client* (mampu menebak wajah mahasiswa yang dilatih di perangkat *edge* lain berkat hasil agregasi model global *Federated Learning*). Sebaliknya, pengujian dikategorikan Gagal apabila sistem tidak mampu mendeteksi wajah, salah mengenali identitas subjek, mengeluarkan keputusan tidak dikenal (*UNKNOWN*) untuk mahasiswa yang seharusnya terdaftar, atau mengalami *timeout* dan kegagalan operasional pada database lokal.

Selanjutnya, pengujian performa sistem pada kondisi nyata dilakukan dengan memanfaatkan fungsi simulasi kelas otomatis yang telah disematkan pada pembaruan kode program terbaru. Prosedur ini dijalankan dengan mengumpankan berkas video rekaman langsung yang menangkap aktivitas sekelompok mahasiswa di dalam ruang kelas saat proses perkuliahan berlangsung. Skrip pengujian pada sisi *client* akan membaca aliran video tersebut secara sekuensial untuk menyimulasikan proses presensi dinamis di lapangan, di mana sistem dituntut untuk melakukan pelacakan multi-wajah dan inferensi *on-device* secara kontinu. Melalui evaluasi berbasis video kelas ini, efektivitas jaring pengaman *Temporal Voting* akan diuji secara riil dalam meredam *noise* visual dari pergerakan alami mahasiswa, seperti menoleh, menunduk, atau terhalang objek lain, guna memastikan akurasi pencatatan log kehadiran tetap stabil pada situasi operasional yang tidak ideal.

3.2 Bahan dan peralatan yang digunakan

Berikut merupakan alat dan bahan yang akan digunakan dalam penelitian ini:

a. *Dataset* wajah dikumpulkan secara mandiri dari mahasiswa Teknologi Informasi menggunakan *webcam*.
b. Perangkat lunak
    i. Python
    ii. Docker
    iii. Flower, PyTorch, OpenCV, CodeCarbon, dan pustaka pendukung lainnya.
c. Perangkat Keras

Penelitian ini menggunakan tiga jenis perangkat keras dengan spesifikasi sebagai berikut:

i. **VPS**

- Prosesor: Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
- Jumlah Core (vCPU): 4 core
- RAM: 6 GB
- Penyimpanan: 50 GB
- Sistem Operasi: Ubuntu 22.04 LTS

ii. **Raspberry Pi 3 Model B**

- Prosesor: Quad-core ARM Cortex-A53 @ 1.20 GHz
- Jumlah Core (vCPU): 4 core
- RAM: 1 GB
- Penyimpanan: 32 GB
- Sistem Operasi: Linux 6.12.75+rpt-rpi-v8 (aarch64)

iii. **NVIDIA Jetson Nano**

- Prosesor: Quad-core ARM Cortex-A57 @ 1.48 GHz
- Jumlah Core (vCPU): 4 core
- RAM: 4 GB
- Penyimpanan: 16 GB
- Sistem Operasi: Linux 4.9.337-tegra (aarch64)

3.3 Implementasi

Berikut merupakan tahapan implementasi pelaksanaan penelitian ini:

3.3.1 Penyiapan Lingkungan Sistem

Tahap awal pelaksanaan implementasi sistem dilakukan dengan mengonfigurasi seluruh lingkungan perangkat lunak, pustaka pemrograman, dan isolasi kontainerisasi pada sisi server maupun *client*. Langkah kontainerisasi ini krusial untuk menjamin kelancaran komunikasi data dan keselarasan dependensi sistem terdistribusi tanpa terikat oleh sistem operasi bawaan perangkat keras. Proses penyiapan lingkungan operasional ini dibagi menjadi dua poin utama sebagai berikut:

1. Client

Implementasi lingkungan pada sisi *client* diisolasi menggunakan layanan kontainerisasi untuk mendukung sistem komputasi berbasis prosesor ARM yang tertanam pada perangkat keras *edge*. Sistem menggunakan *image* dasar resmi *python:3.10-slim-bullseye* karena varian ini sangat ringan, stabil, dan memiliki kecocokan arsitektur kompilasi silang yang optimal.

Di dalam kontainer ini, sistem mengonfigurasi beberapa parameter lingkungan penting, meliputi tautan alamat basis data SQLite lokal untuk menyimpan data kehadiran sementara, serta variabel alamat server utama dan alamat server *Federated Learning* untuk mengarahkan jalur komunikasi *socket* menuju alamat IP server pusat. Guna menjembatani interaksi visual secara langsung, kontainer diberikan hak akses khusus dan pemetaan perangkat fisik untuk mengaktifkan modul kamera *webcam* eksternal secara langsung, lengkap dengan pengaturan format video MJPG serta batas resolusi tangkapan berdimensi 1280x720 piksel.

2. Server

Implementasi lingkungan pada sisi server pusat dirancang secara multi-kontainer menggunakan berkas konfigurasi yang mengorkestrasi dua layanan utama, yaitu mesin aplikasi utama *Federated Learning* dan sistem manajemen basis data terpusat. Sama halnya dengan sisi *client*, kontainer aplikasi utama server dibangun menggunakan *image* dasar *python:3.10-slim-bullseye* guna menjaga efisiensi konsumsi memori RAM dan keselarasan interpretasi tipe data saat proses serialisasi parameter bobot berlangsung.

Server membuka dua jalur pintu gerbang komunikasi internal, yaitu gerbang pertama untuk melayani penarikan data REST API dari antarmuka *web* Flask, serta gerbang kedua untuk mengelola lalu lintas sinkronisasi parameter bobot lokal algoritma FedProx menggunakan pustaka Flower. Sementara itu, untuk menjamin keamanan rekapitulasi log absensi dari seluruh *client*, server mengintegrasikan layanan basis data relasional berbasis *image* kontainer resmi *postgres:15* yang dikonfigurasi dengan kredensial otentikasi yang aman.

3.3.2 Ekstraksi Data Wajah

Tahap pengelolaan data wajah awal merupakan fase pra-pelatihan yang dieksekusi secara mandiri di luar lingkungan operasional kontainer server maupun *client* menggunakan perangkat laptop. Proses ini dilakukan dengan membaca setiap *frame* dari berkas rekaman video mentah secara berurutan melalui fungsi pemrosesan citra dari pustaka OpenCV. Sistem kemudian mengambil nilai laju *frame* asli dari properti video tersebut untuk dikalikan dengan konstanta *sampling rate* sebesar 0,17 detik, sehingga diperoleh angka interval lompatan bingkai komputasi yang konstan. Setiap kali indeks bingkai berjalan mencapai kelipatan dari nilai interval tersebut, sistem akan mengeksekusi pemotongan dan penyimpanan citra statis secara terformat ke dalam direktori luaran. Alur implementasi ekstraksi visual ini ditunjukkan pada Kode Sumber 3.1.

```
SAMPLING_RATE = 0.17

def run_extraction():
    # Cek folder output
    if not os.path.exists(OUTPUT_DIR):
        os.makedirs(OUTPUT_DIR)

    # Load video
    cap = cv2.VideoCapture(VIDEO_SOURCE)
    if not cap.isOpened():
        return

    # Ambil info FPS video asli
    fps = cap.get(cv2.CAP_PROP_FPS)
    interval = int(fps * SAMPLING_RATE)

    frame_idx = 0
    saved_idx = 0

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Simpan frame berdasarkan interval
        if frame_idx % interval == 0:
            filename = f"frame_{frame_idx:06d}.jpg"
            save_path = os.path.join(OUTPUT_DIR, filename)

            # Simpan
            cv2.imwrite(save_path, frame, [int(cv2.IMWRITE_JPEG_QUALITY), 95])
            saved_idx += 1

        frame_idx += 1

    cap.release()
```

Kode Sumber 3.1 Pemotongan Video Berdasarkan *Frame Rate*

3.3.3 Sinkronisasi Identitas

Tahap penyelarasan identitas merupakan fase krusial dalam arsitektur pembelajaran terdistribusi yang bertujuan untuk menstandarkan indeks label kelas mahasiswa dari seluruh *client* secara terpusat sebelum pelatihan model dimulai. Langkah ini dirancang untuk mengeliminasi risiko terjadinya tumpang tindih penomoran target matriks klasifikasi jala saraf MobileFaceNet antarperangkat *edge* yang mengoleksi subset data berbeda. Mekanisme ini dieksekusi melalui empat tahapan sinkronisasi asinkron antara komponen manajer *client* dan mesin server pusat sebagai berikut:

1. **Registrasi Identitas Pada *Client***

Proses diawali pada sisi *client* dengan memindai seluruh direktori data mentah mahasiswa pada penyimpanan internal untuk memisahkan format penamaan folder menjadi komponen nomor induk mahasiswa (NRP) dan nama subjek secara terpisah. Setiap kali entitas identitas baru ditemukan, manajer *client* secara otomatis mengemas parameter data tersebut ke dalam paket JSON dan mengirimkannya ke server pusat melalui protokol REST API, sekaligus mendaftarkan salinan data tersebut ke dalam pangkalan data SQLite lokal perangkat. Implementasi fungsi penemuan dan pengiriman identitas awal pada sisi *client* ini ditunjukkan pada Kode Sumber 3.2.

```
def run_discovery_phase(self):
    try:
        # Tentukan folder mahasiswa
        path = os.path.join(self.raw_data_path, "students")
        if not os.path.exists(path):
            path = self.raw_data_path

        # Membaca daftar sub-direktori mahasiswa
        folders = [
            f for f in os.listdir(path)
            if os.path.isdir(os.path.join(path, f))
        ]
        for folder in sorted(folders):
            # Memisah nama folder menjadi NRP dan Nama
            nrp = folder.split('_')[0] if "_" in folder else folder
            name = folder.split('_')[1] if "_" in folder else nrp

            # Kirim identitas ke server pusat menggunakan variabel
            url = f"{self.server_api_url}{api_training_get_label}"
            payload = {
                "nrp": nrp,
                "name": name,
                "client_id": self.client_id
            }
            self._safe_request("POST", url, json=payload)

            # Daftarkan ke SQLite lokal perangkat
            db = SessionLocal()
            try:
                user = db.query(UserLocal).filter_by(
                    user_id=nrp
                ).first()
                if not user:
                    db.add(UserLocal(user_id=nrp, name=name))
                    db.commit()
            finally:
                db.close()

```

```
        # Beritahu server bahwa discovery selesai
        done_url = (
            f"{self.server_api_url}"
            f"{api_clients_discovery_done}"
        )
        done_payload = {"client_id": self.client_id}
        self._safe_request("POST", done_url, json=done_payload)
    except Exception:
        pass
```

Kode Sumber 3.2 Registrasi Identitas Pada *Client*

2. Penanganan Registrasi Pada Server

Pada sisi server menangkap kiriman paket registrasi dari seluruh terminal *client* yang aktif melalui fungsi penerimaan *endpoint* khusus. Server memeriksa keberadaan data nomor induk mahasiswa di dalam *database* pusat. Jika data tersebut belum terdaftar, entri pengguna baru akan dibuat secara otomatis dengan pengenal unik menggunakan mekanisme *auto-increment*. Pendekatan terpusat ini memastikan setiap subjek memiliki satu representasi nomor label kelas yang mutlak dan terhindar dari risiko tabrakan identitas lintas perangkat *edge*. Alur penanganan registrasi dan penjaminan ID unik pada server ditunjukkan pada Kode Sumber 3.3.

```
@app.post(api_training_get_label)
async def get_label(data: dict, db: Session = Depends(get_db)):
    nrp = data.get("nrp")
    name = data.get("name", "Unknown")
    edge_id = data.get("client_id", "edge-1")
    emb_b64 = data.get("embedding")

    # Dekode embedding dari Base64
    emb_bytes = (
        base64.b64decode(emb_b64)
        if emb_b64 else None
    )

    # Periksa apakah NRP sudah terdaftar
    user = db.query(UserGlobal).filter_by(nrp=nrp).first()
    if not user:
        # Tambah entri baru (ID terbuat secara otomatis)
        user = UserGlobal(
            nrp=nrp,
            name=name,
            registered_edge_id=edge_id,
            embedding=emb_bytes
        )
        db.add(user)
        db.commit()
        db.refresh(user)
    else:
        # Perbarui nama atau embedding
        user.name = name
        if emb_bytes:
            user.embedding = emb_bytes
        db.commit()

    # Masukkan NRP ke daftar data aktif
    if nrp not in fl_manager.received_data:
        fl_manager.received_data.append(nrp)

    return {"nrp": nrp, "label": user.user_id}
```

Kode Sumber 3.3 Penanganan Registrasi Pada Server

3. Pembentukan *Global Label Map* Server

Setelah seluruh data dari *node client* terkumpul secara kolektif, server mengeksekusi fungsi penarikan data terpusat untuk membentuk struktur klaster matriks yang seragam. Sistem menarik seluruh baris identitas pengguna global dari *database*, mengurutkannya secara teratur berpatokan pada urutan string karakter nomor induk mahasiswa, lalu mengemas barisan data tersebut ke dalam sebuah *array* indeks terpadu. Larik data urutan yang terstandar inilah yang menjadi struktur acuan tunggal dimensi lapisan luaran (*output layer classifier*) pada arsitektur model. Logika pembentukan peta label terpadu di sisi server ditunjukkan pada Kode Sumber 3.4.

```
@app.get("/api/training/label_map")
async def get_label_map(db: Session = Depends(get_db)):
    # Ambil semua identitas terdaftar, urutkan berdasarkan NRP
    users = db.query(UserGlobal).order_by(UserGlobal.nrp).all()
    # Mengembalikan array NRP terurut sebagai Global Label Map
    return [u.nrp for u in users]
```

Kode Sumber 3.4 Pembentukan Peta Label Global Server

4. Sinkronisasi *Global* Label Map *Client*

Pada fase akhir sebelum proses pra-pemrosesan citra dijalankan, komponen manajer *client* memicu fungsi penarikan balik peta klasifikasi global. *Client* mengunduh *array* indeks pemetaan terpadu dari server melalui protokol HTTP GET, menghitung total dimensi kelas baru untuk melakukan ekspansi ukuran matriks bobot pada *classifier head* model lokal secara otomatis, lalu membekukan hasilnya ke dalam berkas konfigurasi JSON lokal. Berkas konfigurasi objek inilah yang menjadi acuan utama sistem *client* dalam memetakan target folder data latih secara seragam di seluruh ekosistem *Federated Learning*. Logika kode sinkronisasi umpan balik pada *client* ditunjukkan pada Kode Sumber 3.5.

```
def sync_label_map(self):
    try:
        self._ensure_models_loaded()

        # Request daftar label map dari server menggunakan variabel
        url = f"{self.server_api_url}{api_training_label_map}"
        res = self._safe_request("GET", url)

        if res and res.status_code == 200:
            self.client.label_map = res.json()
            self.num_classes = len(self.client.label_map)

            # Perluas classifier head jika kelas bertambah
            if self.head is not None:
                curr_classes = self.head.weight.shape[0]
                if curr_classes != self.num_classes:
                    new_map = {
                        nrp: idx for idx, nrp in
                        enumerate(self.client.label_map)
                    }
                    self.head = (
                        self.client.trainer
                        .update_head(self.num_classes, new_map)
                    )

            # Simpan peta label ke file JSON lokal
            path = os.path.join(
                self.data_path, "models", "label_map.json"
```

```
            )
            os.makedirs(os.path.dirname(path), exist_ok=True)
            with open(path, "w") as f:
                json.dump(self.client.label_map, f)
            return True
    except Exception:
        pass
    return False
```

Kode Sumber 3.5 Sinkronisasi Peta Label Global *Client*

3.3.4 *Image Preprocessing*

Tahap *image preprocessing* merupakan fase operasional terisolasi pada sisi *client* yang bertujuan untuk membersihkan, menyeimbangkan geometri posisi spasial, serta menstandarkan dimensi matriks piksel wajah mahasiswa sebelum diekstraksi menjadi vektor karakteristik. Langkah ini dirancang untuk mengondisikan data visual mentah agar memenuhi kriteria format masukan arsitektur jala saraf MobileFaceNet secara seragam. Seluruh siklus manipulasi data statis ini dijalankan menggunakan unit CPU pada perangkat keras *client* demi menjaga efisiensi konsumsi daya melalui mekanisme pemuatan model detektor secara tertunda (*lazy loading*). Mekanisme ini dibagi ke dalam empat tahapan proses operasional sebagai berikut:

1. Deteksi dan *Alignment* Wajah

Proses diawali dengan melokalisasi kotak pembatas wajah memanfaatkan arsitektur MTCNN sekaligus mengalkulasi matriks transformasi afinitas dua dimensi berdasarkan koordinat lima titik marka penanda biometrik meliputi mata, hidung, dan mulut. Citra wajah kemudian dirotasi secara matematis berpatokan pada koordinat acuan standar agar posisi kemiringan kepala mahasiswa tegak simetris, lalu dipotong secara presisi ke dalam dimensi potret wajib 96x112 piksel. Jika proses transformasi afinitas mengalami kegagalan, sistem menyediakan fungsi jala pengaman (*fallback*) untuk memotong gambar secara statis mengikuti koordinat kotak pembatas awal dengan margin tambahan. Implementasi kode untuk proses geometri spasial ini ditunjukkan pada Kode Sumber 3.6.

```
def detect_face(self, img, save_path=None):
    try:
        orig_w, orig_h = img.size
        max_size = 640
        # Downscaling gambar agar proses MTCNN lebih cepat
        if orig_w > max_size or orig_h > max_size:
            scale = max_size / max(orig_w, orig_h)
            new_size = (
                int(orig_w * scale),
                int(orig_h * scale)
            )
            img_detect = img.resize(new_size, Image.BILINEAR)
        else:
            img_detect = img
            scale = 1.0

        # Inferensi deteksi wajah menggunakan MTCNN
        boxes, probs, landmarks = self.mtcnn.detect(
            img_detect, landmarks=True
        )
        if boxes is None or len(boxes) == 0:
            return None, None, 0.0

        if scale != 1.0:
            boxes = boxes / scale
```

```
            if landmarks is not None:
                landmarks = landmarks / scale

        face_img = None

        # Affine Alignment berbasis koordinat 5 titik landmark
        if landmarks is not None and landmarks[0] is not None:
            try:
                src = np.array(landmarks[0], dtype=np.float32)
                M, _ = cv2.estimateAffinePartial2D(
                    src, CANONICAL_LANDMARKS, method=cv2.LMEDS
                )
                if M is not None:
                    img_cv = cv2.cvtColor(
                        np.array(img), cv2.COLOR_RGB2BGR
                    )
                    aligned = cv2.warpAffine(
                        img_cv, M, (96, 112),
                        flags=cv2.INTER_LINEAR,
                        borderMode=cv2.BORDER_REPLICATE
                    )
                    face_img = Image.fromarray(
                        cv2.cvtColor(aligned, cv2.COLOR_BGR2RGB)
                    )
            except Exception:
                pass

        # Fallback Bounding Box Crop jika alignment gagal
        if face_img is None:
            box = boxes[0]
            margin = 20
            x1 = max(0, int(box[0] - margin / 2))
            y1 = max(0, int(box[1] - margin / 2))
            x2 = min(img.width, int(box[2] + margin / 2))
            y2 = min(img.height, int(box[3] + margin / 2))

            face_img = img.crop((x1, y1, x2, y2)).resize(
                (96, 112), Image.BILINEAR
            )

        if save_path and face_img:
            face_img.save(save_path)

        return face_img, boxes[0], probs[0]
    except Exception:
        return None, None, 0.0
```

Kode Sumber 3.6 Deteksi dan *Alignment* Wajah

2. Persiapan Model Tensor

Proses kedua berfungsi untuk mengonversi format data citra hasil pemotongan wajah menjadi bentuk objek matriks bilangan desimal multidimensi (Tensor) PyTorch. Melalui fungsi penyiapan ini, nilai intensitas piksel dipetakan ulang lewat fungsi transformasi normalisasi dengan rentang nilai parameter rata-rata (*mean*) dan deviasi standar (*standard deviation*) yang disesuaikan pada angka 0,5 guna menstabilkan gradien komputasi. Sistem kemudian menyisipkan dimensi *batch* tambahan pada sumbu pertama matriks agar struktur data tensor siap diumpankan secara langsung menuju lapisan masukan model. Logika transformasi penyiapan data masukan ini ditunjukkan pada Kode Sumber 3.7.

```
def prepare_for_model(self, face_data):
        if face_data is None: return None

        # Konversi objek Tensor/Gambar ke format PIL Image
        if isinstance(face_data, torch.Tensor):
            if face_data.max() > 2.0:
                face_data = face_data / 255.0
            face_img = TF.to_pil_image(face_data.clamp(0, 1))
        else:
            face_img = face_data

        # Lakukan resize akhir ke ukuran wajib MobileFaceNet (96x112)
        if face_img.size != (96, 112):
            face_img = face_img.resize((96, 112), Image.BILINEAR)

        # Transformasi gambar ke Tensor dan normalisasi nilai L2
        face_tensor = TF.to_tensor(face_img)
        normalized_tensor = self.normalize(face_tensor)

        # Tambahkan dimensi batch (1, C, H, W)
        return normalized_tensor.unsqueeze(0).to(DEVICE)
```

Kode Sumber 3.7 Transformasi dan Normalisasi Nilai Tensor L2

3. Perhitungan Tingkat Kekaburan Gambar

Proses ketiga bertindak sebagai unit penilai kualitas fisik gambar untuk mendeteksi tingkat ketajaman fokus lensa kamera terhadap objek wajah mahasiswa. Langkah evaluasi dilakukan dengan mengubah ruang warna berkas gambar dari format warna asal menjadi skala keabu-abuan (*grayscale*) terisolasi. Sistem selanjutnya menerapkan operator kernel matematika Laplacian untuk menghitung nilai varians turunan kedua dari matriks gradien, di mana skor yang dihasilkan merepresentasikan tingkat ketajaman tepi objek visual secara kuantitatif. Logika perhitungan skor kekaburan citra ini ditunjukkan pada Kode Sumber 3.8.

```
def get_blur_score(self, image_path):
        try:
            img = cv2.imread(image_path)
            if img is None: return 0
            # Konversi gambar ke skala abu-abu (grayscale)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            # Hitung variansi Laplacian untuk mendapatkan nilai ketajaman
            return cv2.Laplacian(gray, cv2.CV_64F).var()
        except:
            return 0
```

Kode Sumber 3.8 Perhitungan Skor Ketajaman Piksel *Laplacian*

4. Seleksi Kualitas Citra Terbaik Berbasis Ranking

Proses keempat mengintegrasikan fungsi seleksi massal untuk menyaring dan memilih sejumlah N berkas foto wajah terbaik dari direktori penyimpanan sementara. Fungsi ini mendata seluruh *file* citra, mengurutkan daftar gambar berdasarkan pemetaan skor *Laplacian Variance* dari nilai tertinggi ke terendah, lalu mengekstrak sampel foto tertajam sesuai dengan jumlah batasan yang ditentukan. Guna mempercepat durasi eksekusi data, skrip memanfaatkan sistem pencatatan berkas riwayat berbasis objek dokumen konfigurasi agar kalkulasi matematika tidak perlu diproses ulang dari awal. Logika pengurutan kualitas citra terbaik ini ditunjukkan pada Kode Sumber 3.9.

```python
def select_best_faces(self, folder_path, n=50):
    if not os.path.exists(folder_path):
        return []

    final_cache_path = os.path.join(
        folder_path, ".selection_cache.json"
    )
    scores_cache_path = os.path.join(
        folder_path, ".laplacian_scores.json"
    )

    all_imgs = sorted([
        f for f in os.listdir(folder_path)
        if f.lower().endswith(('.jpg', '.jpeg', '.png'))
    ])

    if not all_imgs:
        return []

    scored_map = {}
    if os.path.exists(scores_cache_path):
        try:
            with open(scores_cache_path, "r") as f:
                scored_map = json.load(f)
        except Exception:
            pass

    needs_save = False
    for img_name in all_imgs:
        if img_name in scored_map:
            continue

        img_path = os.path.join(folder_path, img_name)
        scored_map[img_name] = self.get_blur_score(img_path)
        needs_save = True

    if needs_save:
        try:
            with open(scores_cache_path, "w") as f:
                json.dump(scored_map, f)
        except Exception:
            pass

    # Urutkan dan pilih N foto tertajam
    scored_list = [
        (name, score) for name, score in scored_map.items()
    ]
    scored_list.sort(key=lambda x: x[1], reverse=True)
    selected = [s[0] for s in scored_list[:n]]

    # Simpan hasil seleksi ke cache
    try:
        with open(final_cache_path, "w") as f:
            json.dump({"n": n, "filenames": selected}, f)
    except Exception:
        pass

    return selected

```

Kode Sumber 3.9 Seleksi Kualitas Citra Berbasis Ranking Ketajaman

3.3.5 Enkripsi Embedding

Tahap enkripsi kriptografi merupakan fase pengamanan data biometrik pada sisi *client* yang bertujuan untuk melindungi kerahasiaan vektor karakteristik (*face embedding*) mahasiswa sebelum disimpan ke dalam pangkalan data lokal maupun ditransmisikan di dalam jaringan. Langkah ini dirancang sebagai jaring pengaman privasi untuk memastikan bahwa jika pangkalan data lokal pada perangkat *edge* berhasil ditembus oleh pihak luar, data vektor wajah tersebut tetap tidak dapat disalahgunakan atau direkonstruksi kembali menjadi citra wajah asli tanpa kunci enkripsi yang sah. Proses pengamanan ini diimplementasikan menggunakan algoritma standar industri AES-256 bit dalam mode operasi *Cipher Block Chaining* (CBC). Fungsi operasional yang bertanggung jawab atas siklus pengamanan data biometrik ini ditunjukkan pada Kode Sumber 3.10.

```
def encrypt_embedding(self, embedding_numpy):
    # Serialisasi array numpy embedding wajah menjadi bytes
    data_bytes = pickle.dumps(embedding_numpy)

    # Buat Initialization Vector (IV) acak sepanjang 16 bytes
    iv = os.urandom(16)

    # Inisialisasi Cipher AES-256 dalam mode CBC
    cipher = Cipher(
        algorithms.AES(self.key),
        modes.CBC(iv),
        backend=default_backend()
    )
    encryptor = cipher.encryptor()

    # Lakukan padding data bytes menggunakan standar PKCS7
    padder = padding.PKCS7(128).padder()
    padded_data = (
        padder.update(data_bytes) +
        padder.finalize()
    )

    # Proses enkripsi data hasil padding
    encrypted_data = (
        encryptor.update(padded_data) +
        encryptor.finalize()
    )

    return encrypted_data, iv
```

Kode Sumber 3.10 Enkripsi Menggunakan AES-256

3.3.6 Pelatihan Model

Tahap pelatihan model merupakan fase pembelajaran utama dalam sistem absensi terdistribusi, di mana bobot model dioptimasi secara kolaboratif menggunakan algoritma FedProx. Sesuai strategi pFedFace, proses agregasi hanya dilakukan pada parameter *backbone*, sementara lapisan *Batch Normalization* dikelola secara lokal oleh masing-masing *client*. Fase ini terdiri dari empat tahap utama, yaitu pembekuan parsial lapisan model, augmentasi citra secara dinamis, pelatihan lokal dengan regularisasi FedProx, serta agregasi parameter terpusat di server.

1. Pembekuan Parsial (*Layer Freezing*)

Sebelum siklus pelatihan dimulai pada perangkat *edge*, sistem membekukan lapisan-lapisan awal *backbone MobileFaceNet*. Pembekuan lapisan awal (*early freezing*) ini menonaktifkan kalkulasi gradien pada lapisan ekstraksi fitur dasar, sehingga menghemat konsumsi memori komputasi perangkat *edge*. Pengondisian pembekuan lapisan ini ditunjukkan pada Kode Sumber 3.11.

```
def set_model_freeze(model, freeze_mode="early"):
    for param in model.parameters():
        param.requires_grad = True

    if freeze_mode == "none":
        return

    # Bekukan lapisan input awal MobileFaceNet
    if freeze_mode == "early":
        for param in model.conv1.parameters():
            param.requires_grad = False
        for param in model.dw_conv1.parameters():
            param.requires_grad = False

        # Bekukan 12 blok bottleneck awal pada backbone
        for i in range(12):
            for param in model.blocks[i].parameters():
                param.requires_grad = False

    # Bekukan seluruh parameter backbone
    elif freeze_mode == "backbone":
        for param in model.parameters():
            param.requires_grad = False
```

Kode Sumber 3.11 Pengaturan Pembekuan Lapisan Model

2. Augmentasi Data *On-The-Fly*

Selama proses pemuatan data, gambar wajah dimodifikasi secara dinamis di RAM menggunakan transformasi acak. Langkah ini menghasilkan variasi spasial dan visual baru di setiap *epoch* tanpa mengubah berkas citra asli pada penyimpanan fisik. Logika augmentasi data dan pemuatan dinamis ditunjukkan pada Kode Sumber 3.12.

```
# Konfigurasi pipa augmentasi citra
self.transform = transforms.Compose([
    transforms.Resize(
        (112, 96),
        interpolation=InterpolationMode.BILINEAR
    ),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(degrees=20),
    transforms.RandomPerspective(distortion_scale=0.2, p=0.3),
    transforms.ColorJitter(
        brightness=0.5, contrast=0.5,
        saturation=0.4, hue=0.1
    ),
    transforms.RandomGrayscale(p=0.2),
    transforms.RandomAutocontrast(p=0.2),
    transforms.RandomApply([
        transforms.GaussianBlur(
            kernel_size=(3, 3), sigma=(0.1, 2.0)
        )
    ], p=0.4),
    transforms.RandomAdjustSharpness(
```

```
        sharpness_factor=2, p=0.2
    ),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.5, 0.5, 0.5],
        std=[0.50196, 0.50196, 0.50196]
    ),
    transforms.RandomErasing(p=0.1)
])

# Pemuatan sampel data latih secara dinamis
def __getitem__(self, idx):
    sample = self.samples[idx]
    label = sample['label']

    # Baca gambar jika tipe data berupa file citra
    if sample['type'] == "image":
        image = Image.open(sample['path']).convert('RGB')
        if self.transform:
            image = self.transform(image)
        return image, label, False
    else:
        # Kembalikan data embedding memori global
        return sample['data'], label, True
```

Kode Sumber 3.12 Augmentasi dan Pemuatan Data Latih

3. Pelatihan Lokal

Pelatihan model lokal dijalankan pada *client* menggunakan optimasi *SGD* dan *ArcFace Margin Loss*. Guna mengatasi masalah ketidakseimbangan distribusi data *Non-IID*, ditambahkan penalti regulasi jarak L2 (*proximity loss*) antara parameter model lokal dengan model global acuan. Logika utama pelatihan lokal *client* ditunjukkan pada Kode Sumber 3.13.

```
def train(
    self, epochs, lr, round_num,
    global_embeddings=None, label_map=None, mu=0.05
):
    # Inisialisasi dataset dan loader
    dataset = FaceDataset(
        self.data_path,
        global_embeddings=global_embeddings,
        transform=self.transform
    )
    dataloader = DataLoader(
        dataset, batch_size=32,
        shuffle=True, collate_fn=hybrid_collate
    )

    # Bekukan lapisan awal model lokal
    set_model_freeze(self.backbone, freeze_mode="early")
    global_ref = copy.deepcopy(self.backbone.state_dict())

    # Pengaturan optimizer SGD dan kriteria loss
    optimizer = torch.optim.SGD(
        self.backbone.parameters(),
        lr=lr, momentum=0.9, nesterov=True
    )
    criterion = nn.CrossEntropyLoss()

    for epoch in range(epochs):
        self.backbone.train()
        self.head.train()

```

```
        for imgs, embs, labels, is_embedding in dataloader:
            imgs = imgs.to(self.device)
            embs = embs.to(self.device)
            labels = labels.to(self.device)
            optimizer.zero_grad()

            features_local = torch.zeros(
                (labels.size(0), 128), device=self.device
            )
            img_mask, emb_mask = ~is_embedding, is_embedding

            # Forward pass citra wajah ke backbone
            if img_mask.any():
                features_local[img_mask] = (
                    self.backbone(imgs[img_mask])
                )
            if emb_mask.any():
                features_local[emb_mask] = embs[emb_mask]

            # Hitung loss klasifikasi ArcFace
            outputs = self.head(features_local, labels)
            ce_loss = criterion(outputs, labels)

            # Hitung penalti regulasi jarak L2 (FedProx)
            prox_loss = torch.tensor(0.0, device=self.device)
            for name, param in self.backbone.named_parameters():
                if name in global_ref:
                    diff_norm = (
                        torch.norm(param - global_ref[name])**2
                    )
                    prox_loss += (mu / 2) * diff_norm

            # Total Loss = ArcFace Loss + Proximity Loss
            loss = ce_loss + prox_loss

            # Backward pass & update bobot
            loss.backward()
            optimizer.step()
```

Kode Sumber 3.13 *Loop* Pelatihan Lokal *Client*

4. Agregasi Parameter pada Server

Server mengagregasikan parameter *backbone* dari seluruh *client* aktif menggunakan pembobotan rata-rata berbasis Flower. Guna mempertahankan adaptasi personalisasi (*personalization*) tingkat *client*, statistik lapisan *Batch Normalization* (*BN*) tidak disertakan dalam proses agregasi global (*global aggregation*). Logika agregasi parameter konvolusi *backbone* ditunjukkan pada Kode Sumber 3.14.

```
def aggregate_fit(
    self, server_round: int, results: list, failures: list
):
    # Agregasi parameter terbobot Flower
    aggregated_parameters, aggregated_metrics = (
        super().aggregate_fit(
            server_round, results, failures
        )
    )
    if aggregated_parameters is None:
        return None, aggregated_metrics

    # Urutkan bobot hasil agregasi ke NumPy array
    params_np = (
```

```
        fl.common.parameters_to_ndarrays(
            aggregated_parameters
        )
    )
    sd = self.load_previous_global_weights()
    all_keys = list(sd.keys())

    # Filter lapisan konvolusi (tanpa BN) untuk pFedFace
    bn_patterns = ['bn', 'running_', 'num_batches_tracked']
    conv_keys = [
        k for k in all_keys
        if not any(x in k.lower() for x in bn_patterns)
    ]

    # Tentukan kunci target parameter yang akan diperbarui
    target_keys = (
        conv_keys if len(params_np) == len(conv_keys)
        else all_keys
    )

    # Rekonstruksi parameter ke Tensor PyTorch
    backbone_params = {}
    bn_params = {}
    for i, k in enumerate(target_keys):
        if i < len(params_np):
            val = torch.from_numpy(params_np[i].copy())
            if any(x in k.lower() for x in bn_patterns):
                bn_params[k] = val
            else:
                backbone_params[k] = val

    # Perbarui bobot model global baru
    sd.update(backbone_params)
    if bn_params:
        sd.update(bn_params)

    # Simpan bobot teragregasi ke database
    self.save_new_global_weights(sd)
    return aggregated_parameters, aggregated_metrics
```

Kode Sumber 3.14 Agregasi Model *Backbone* Global

3.3.7 Evaluasi Sistem

Tahap evaluasi sistem dibagi menjadi dua fokus pengukuran utama: evaluasi performa model untuk mengukur keandalan klasifikasi wajah, dan evaluasi konsumsi sumber daya (ekonomi) untuk mengukur dampak penggunaan daya komputasi serta transfer data jaringan (*bandwidth*) dalam skema pembelajaran terdistribusi.

1. Evaluasi Performa Model

Evaluasi performa model dilakukan baik di sisi *client* untuk memvalidasi model lokal, maupun di sisi server untuk mengukur performa agregasi model global secara menyeluruh:

a. Validasi Performa Lokal *Client*

Evaluasi performa model lokal dijalankan pada akhir setiap ronde pelatihan *client* menggunakan *dataset* validasi lokal yang terpisah. Pengukuran ini menerapkan teknik *Flip Trick* untuk mencerminkan akurasi sesungguhnya pada kondisi operasional. Implementasi *loop* evaluasi performa lokal ini ditunjukkan pada Kode Sumber 3.15.

```python
def evaluate(self, global_embeddings=None, label_map=None):
    # Memuat dataset validasi lokal
    dataset = FaceDataset(
        self.data_path,
        global_embeddings=global_embeddings,
        transform=self.val_transform,
        mode="val", label_map=label_map
    )
    if len(dataset) < 2:
        return 0.0, 0.0, len(dataset)

    dataloader = DataLoader(
        dataset, batch_size=16,
        shuffle=False, collate_fn=hybrid_collate
    )
    criterion = nn.CrossEntropyLoss()
    self.backbone.eval()
    self.head.eval()

    total_loss, correct, total = 0.0, 0, 0
    with torch.no_grad():
        for imgs, embs, labels, is_embedding in dataloader:
            imgs = imgs.to(self.device)
            embs = embs.to(self.device)
            labels = labels.to(self.device)

            features = torch.zeros(
                (labels.size(0), 128), device=self.device
            )
            img_mask, emb_mask = ~is_embedding, is_embedding

            if img_mask.any():
                img_input = imgs[img_mask]

                # Ekstraksi embedding asli & mirror (Flip Trick)
                f_orig = self.backbone(img_input)
                f_flip = self.backbone(torch.flip(img_input, [3]))

                features[img_mask] = (
                    torch.nn.functional.normalize(
                        f_orig + f_flip, p=2, dim=1
                    )
                )

            if emb_mask.any():
                features[emb_mask] = embs[emb_mask]

            logits = self.head.get_logits(features)
            outputs = self.head(features, labels)
            loss = criterion(outputs, labels)

            total_loss += loss.item()
            _, predicted = torch.max(logits.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    avg_loss = (
        total_loss / len(dataloader)
        if len(dataloader) > 0 else 0.0
    )
    accuracy = correct / total if total > 0 else 0.0
    return avg_loss, accuracy, total
```

Kode Sumber 3.15 Uji Coba Validasi Performa Model *Client*

b. Agregasi Metrik Global Server

Di sisi server, hasil evaluasi performa lokal dari seluruh *client* digabungkan menjadi metrik evaluasi global terpadu menggunakan metode rata-rata tertimbang (*weighted average*). Server mengagregasikan nilai *loss* dan *accuracy* baik dari data pelatihan lokal maupun data validasi lokal berdasarkan jumlah sampel data masing-masing *client*. Implementasinya ditunjukkan pada Kode Sumber 3.16.

```
def weighted_average(metrics: list) -> dict:
    # Mengambil jumlah sampel per client
    examples = [m[0] for m in metrics]
    total_examples = sum(examples)
    if total_examples == 0:
        return {}

    # Rata-rata tertimbang metrik pelatihan & validasi
    return {
        "accuracy": sum([
            m[1].get("accuracy", 0.0) * m[0] for m in metrics
        ]) / total_examples,
        "loss": sum([
            m[1].get("loss", 0.0) * m[0] for m in metrics
        ]) / total_examples,
        "val_accuracy": sum([
            m[1].get("val_accuracy", 0.0) * m[0] for m in metrics
        ]) / total_examples,
        "val_loss": sum([
            m[1].get("val_loss", 0.0) * m[0] for m in metrics
        ]) / total_examples,
    }
```

Kode Sumber 3.16 Fungsi Agregasi Metrik Performa Server

2. Evaluasi Konsumsi Sumber Daya (Ekonomi)

Evaluasi ekonomi mencakup pengukuran konsumsi sumber daya sistem terdistribusi yang dibagi secara berimbang menjadi dua fokus pengukuran:

a. Pengukuran Sisi *Client*

Client secara mandiri melacak durasi waktu pelatihan lokal (dalam detik) dan total konsumsi energi listrik CPU (dalam kWh) pada setiap putaran latihan lokal. Pengukuran durasi dilakukan dengan menghitung selisih waktu sistem operasi, sementara pengukuran energi listrik memanfaatkan pelacakan *offline* dari pustaka CodeCarbon. Penggabungan alur pelacakan durasi dan daya pada *client* ditunjukkan pada Kode Sumber 3.17.

```
# Inisialisasi awal waktu dan pencatat daya CodeCarbon
start_train_time = time.time()
tracker = OfflineEmissionsTracker(
    country_iso_code="IDN",
    measure_power_secs=15,
    log_level="error",
    save_to_file=False
)
tracker.start()

# Dapatkan total konsumsi kWh dan durasi detik pelatihan
energy_kwh = tracker.stop()
train_duration = time.time() - start_train_time
```

Kode Sumber 3.17 Pelacakan Durasi dan Energi *Client*

b. Pengukuran Sisi Server

Server melacak tiga parameter utama secara terpadu: total durasi eksekusi ronde global, estimasi transfer data jaringan (dalam MB) secara dinamis dari ukuran fisik berkas model di disk, serta total akumulasi konsumsi daya (server dan gabungan laporan energi *client*). Implementasi gabungan pelacakan sumber daya di server ditunjukkan pada Kode Sumber 3.18.

```
# Hitung total durasi waktu per ronde global
round_duration = round(
    datetime.now().timestamp() - round_start_time, 2
)

# Hitung transfer bandwidth nyata dari berkas model
bb_size = ECONOMICS["estimated_backbone_size_mb"]
if os.path.exists("data/backbone_pure.pth"):
    bb_size = (
        os.path.getsize("data/backbone_pure.pth")
        / (1024 * 1024)
    )
backbone_sync_mb = num_clients * bb_size * 2

# Akumulasi energi total server dan seluruh client
if has_real_server_energy:
    server_energy = max_server_energy
else:
    server_energy = (
        (round_duration / 3600)
        * server_power_kw
    )

client_energy_total = sum([
    c.get("energy_kwh", 0) for c in clients_data
])
total_energy = server_energy + client_energy_total
```

Kode Sumber 3.18 Pelacakan Durasi, *Bandwidth*, dan Akumulasi Energi Server

3.3.8 Proses Absensi

Tahap proses absensi merupakan fase operasional yang melibatkan interaksi dua arah antara *client* dan server untuk melakukan konsolidasi data kehadiran mahasiswa.

1. Pipa Inferensi *Real-Time Client*

Proses inferensi diawali dengan mendeteksi koordinat wajah menggunakan MTCNN, lalu menyelaraskan posisi wajah agar sudutnya lurus. Selanjutnya, sistem mengekstrak fitur wajah dua kali menggunakan teknik *Flip Trick*, dan mengurangi *noise* sesaat dengan *buffer Temporal Voting*. Jika wajah berhasil dikenali, data presensi disimpan dulu ke *database* lokal sebelum disinkronkan ke server. Alur proses inferensi presensi ini ditunjukkan pada Kode Sumber 3.19.

```
async def process_inference(self, image_b64: str, db: Session):
    img_bytes = base64.b64decode(image_b64)
    img_pil = Image.open(io.BytesIO(img_bytes)).convert('RGB')

    face_tensor, box, prob = image_processor.detect_face(img_pil)
    if face_tensor is None:
        img_pil.close()
        return {"matched": "Unknown", "confidence": 0}

    threshold = self.fl_manager.inference_threshold
    input_tensor = (
```

```
        image_processor
        .prepare_for_model(face_tensor)
        .to(self.fl_manager.device)
    )

    with torch.no_grad():
        self.fl_manager.backbone.eval()
        emb_orig = self.fl_manager.backbone(input_tensor)
        face_flipped = torch.flip(input_tensor, dims=[3])
        emb_mirror = self.fl_manager.backbone(face_flipped)
        query_emb = (
            torch.nn.functional.normalize(
                (emb_orig + emb_mirror) / 2, p=2, dim=1
            )
        )

    now = time.time()
    if now - self.fl_manager.last_face_time > 3.0:
        self.fl_manager.prediction_buffer.clear()

    self.fl_manager.prediction_buffer.append(query_emb)
    self.fl_manager.last_face_time = now

    mean_emb_tensor = (
        torch.stack(
            list(self.fl_manager.prediction_buffer)
        ).mean(0)
    )
    mean_emb_tensor = (
        torch.nn.functional.normalize(
            mean_emb_tensor, p=2, dim=1
        )
    )
    mean_emb = mean_emb_tensor.cpu().numpy()[0]

    local_refs = self._get_cached_identities(db)
    best_match, confidence = (
        identify_user_globally(
            mean_emb, local_refs, threshold=threshold
        )
    )

    user_id = best_match if confidence >= threshold else "Unknown"

    # Simpan hasil absensi jika wajah dikenali
    if user_id != "Unknown":
        try:
            new_attendance = AttendanceLocal(
                user_id=user_id,
                confidence=confidence,
                device_id=os.getenv("HOSTNAME", "terminal-1")
            )
            db.add(new_attendance)
            db.commit()
        except Exception:
            db.rollback()

    img_pil.close()
    return {
        "matched": user_id,
        "confidence": float(confidence),
        "box": box.tolist() if box is not None else None
    }
```

Kode Sumber 3.19 Alur Utama Inferensi Presensi Wajah *Client*

2. Sinkronisasi Presensi Server

Di sisi server, terdapat sebuah *endpoint* untuk menerima unggahan *batch log* presensi dari database lokal *client* ketika koneksi jaringan tersedia. Server mencocokkan NRP mahasiswa dan menyimpan *log* kehadiran global secara aman ke *database*. Implementasi sinkronisasi absensi pada server ditunjukkan pada Kode Sumber 3.20.

```python
@app.post("/api/attendance/sync")
async def sync_attendance(
    records: List[Dict[str, Any]] = Body(...),
    db: Session = Depends(get_db)
):
    new_records = 0
    for rec in records:
        try:
            nrp = rec.get('user_id')
            user = (
                db.query(UserGlobal)
                .filter_by(nrp=nrp).first()
            )
            if not user:
                continue

            ts_str = rec['timestamp']
            ts = datetime.fromisoformat(ts_str.replace('Z', '+00:00'))

            exists = (
                db.query(AttendanceRecap)
                .filter_by(user_id=user.user_id, timestamp=ts)
                .first()
            )
            if not exists:
                conf = rec.get('confidence', 0.0)
                client_id = rec.get('client_id', 'unknown-edge')

                # Simpan rekap absensi global mahasiswa
                new_item = AttendanceRecap(
                    user_id=user.user_id,
                    edge_id=client_id,
                    timestamp=ts,
                    confidence=conf
                )
                db.add(new_item)
                new_records += 1
        except Exception:
            continue

    db.commit()
    return {"status": "success", "new_records": new_records}
```

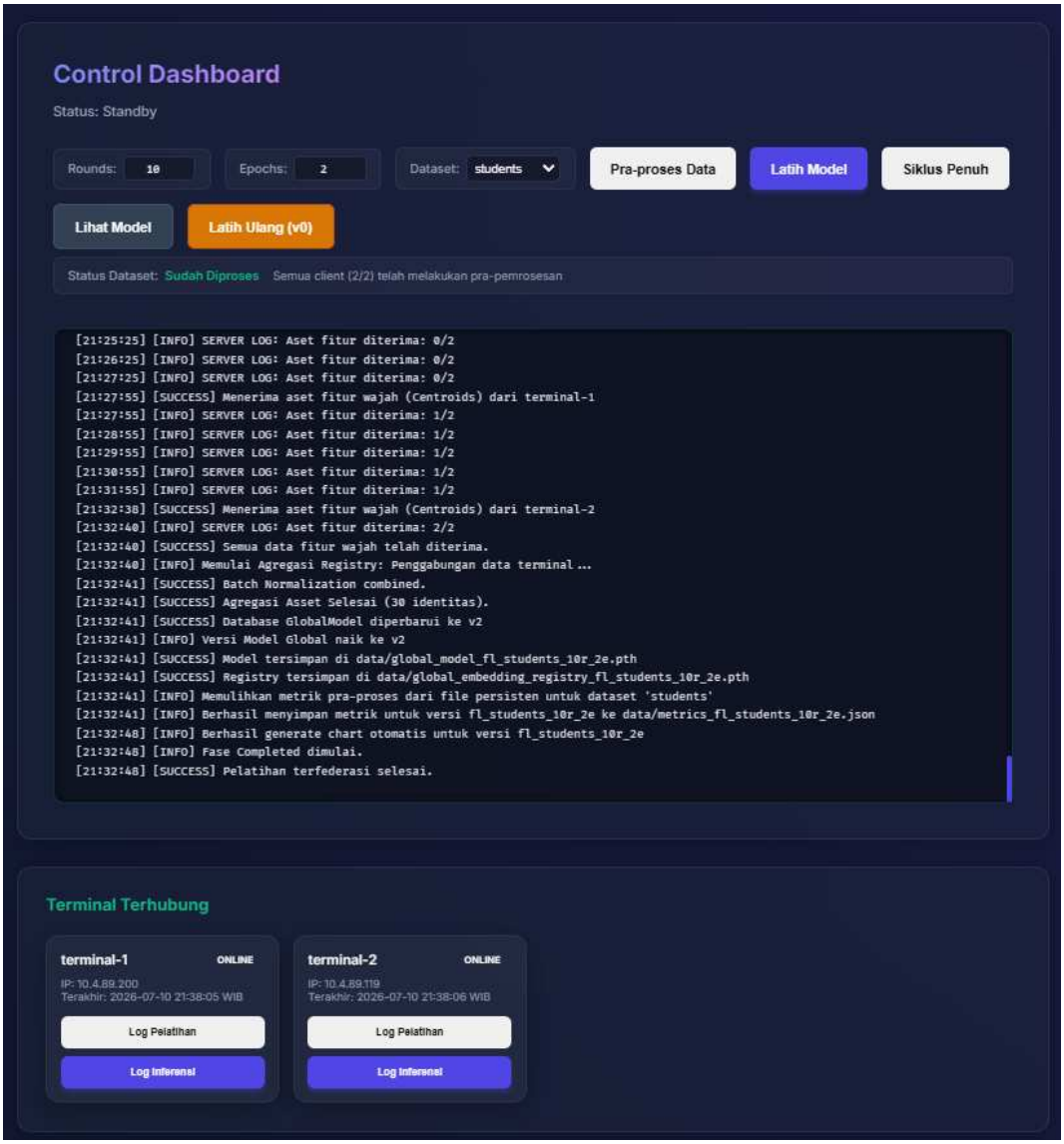
Kode Sumber 3.20 Sinkronisasi Kehadiran Global Server

3.3.9 Pengembangan Antarmuka Aplikasi

Tahap pengembangan antarmuka bertujuan untuk menyediakan visualisasi interaktif bagi administrator dan pengguna dalam memantau jalannya sistem absensi terdistribusi. Pengembangan ini dibagi menjadi dua bagian utama, yaitu antarmuka pusat pada sisi server dan antarmuka operasional pada sisi *client*.

1. Antarmuka Server

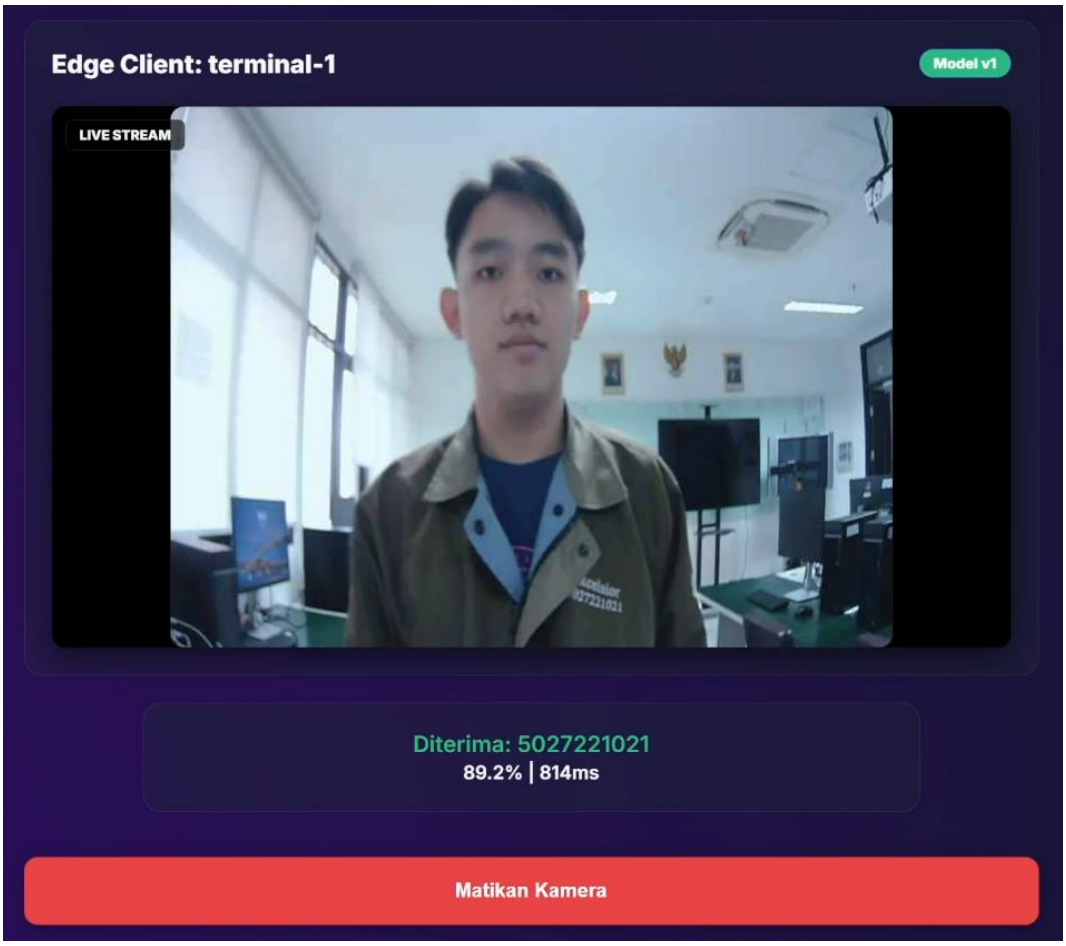
Antarmuka pada sisi server dirancang berbasis *web dashboard* terpusat yang memfasilitasi administrator untuk memantau status keaktifan masing-masing *edge node*, melihat visualisasi grafik metrik performa dari setiap ronde pelatihan *Federated Learning*, serta mengelola rekapitulasi data kehadiran mahasiswa global yang disimpan di *database* pusat. Tampilan antarmuka dasbor utama ini dapat dilihat pada Gambar 3.12.



Gambar 3.12 Tampilan Antarmuka Laman Utama Server

2. Antarmuka *Client*

Antarmuka pada sisi *client* dirancang untuk dioperasikan langsung di kelas menggunakan terminal *edge*. Halaman ini menyajikan tampilan *video stream* dari kamera secara *real-time*, melakukan identifikasi menggunakan algoritma pencocokan *embedding* dan *temporal voting*, serta menampilkan notifikasi status kehadiran mahasiswa secara langsung sebelum datanya disinkronisasikan ke server pusat. Tampilan antarmuka terminal sisi *client* ini dapat dilihat pada Gambar 3.13.



Gambar 3.13 Tampilan Antarmuka Terminal Presensi Sisi *Client*

[Halaman ini sengaja dikosongkan]

BAB 4 HASIL DAN PEMBAHASAN

4.1 Skenario Uji Coba

Pada penelitian ini, analisis performa sistem dibagi menjadi dua tahapan utama, yaitu skenario eksperimen pelatihan model untuk mengukur efisiensi sumber daya komputasi serta skenario uji coba inferensi untuk mengukur keandalan pengenalan identitas secara langsung pada perangkat *edge*.

4.1.1 Skenario Eksperimen Pelatihan Model

Skenario eksperimen pelatihan dilakukan untuk membandingkan proses pembentukan model serta beban ekosistem komputasi antara dua metode pembelajaran yang berbeda, yaitu *Federated Learning* dan *Centralized Learning*. Parameter pelatihan lokal pada *Federated Learning* telah dijelaskan sebelumnya pada Tabel 3.1. Guna memastikan perbedaan hasil yang teramati murni disebabkan oleh perbedaan pendekatan pembelajaran dan bukan oleh faktor konfigurasi lain, seluruh parameter tersebut, mulai dari arsitektur, optimasi, fungsi kerugian, strategi *freezing*, manajemen resolusi, hingga augmentasi data, juga diterapkan secara identik pada *Centralized Learning*, sebagaimana dirangkum pada Tabel 4.1.

Tabel 4.1 Parameter Eksperimen *Federated Learning* dan *Centralized Learning*

Parameter	***Federated Learning***	***Centralized Learning***
Arsitektur *Backbone*	MobileFaceNet (128-dim)	MobileFaceNet (128-dim)
Optimizer	SGD	SGD
Momentum / Nesterov	0,9 / Aktif	0,9 / Aktif
Batch size	32	32
Learning rate awal	0,05	0,05
Learning rate scheduler	Cosine Decay	Cosine Decay
Loss function	*ArcFace Margin Loss*	*ArcFace Margin Loss*
Partial Freezing	14 dari 19 lapisan dibekukan	14 dari 19 lapisan dibekukan
Manajemen Resolusi	*Downscale* 640px + *Vectorization*	*Downscale* 640px + *Vectorization*
Augmentasi	*Random Horizontal Flip, Random Rotation, Random Perspective, Color Jitter, Random Grayscale, Random Autocontrast, GaussianBlur, Random Adjust Sharpness, dan Random Erasing*	*Random Horizontal Flip, Random Rotation, Random Perspective, Color Jitter, Random Grayscale, Random Autocontrast, GaussianBlur, Random Adjust Sharpness, dan Random Erasing*

Berdasarkan parameter yang telah disetarakan pada Tabel 4.1, kedua metode pembelajaran selanjutnya dijalankan mengikuti skenario pengujiannya masing-masing. Skenario *Federated Learning* dirancang untuk mengevaluasi proses pelatihan desentralisasi menggunakan algoritma FedProx, sedangkan skenario *Centralized Learning* dirancang sebagai pembanding menggunakan pendekatan pelatihan terpusat konvensional. Rincian dari masing-masing skenario dijelaskan sebagai berikut:

1. ***Federated Learning***

Eksperimen ini bertujuan untuk mengevaluasi efisiensi sumber daya sekaligus kualitas performa model pada proses pelatihan desentralisasi menggunakan algoritma FedProx yang berjalan secara lokal pada perangkat *edge*. Proses pengujian dilakukan dengan cara menjalankan simulasi pelatihan desentralisasi sebanyak total 10 ronde, dengan variasi jumlah *epoch* lokal pada sisi *client* mulai dari 1 hingga 5 *epoch* per ronde untuk mengamati pengaruhnya terhadap performa dan efisiensi sumber daya. Pada setiap ronde pelatihan ini, sistem akan mencatat secara berkala metrik performa kecerdasan model yang meliputi nilai *loss*, *accuracy*, *validation loss*, dan *validation accuracy* pada sisi *client* maupun server. Bersamaan dengan itu, sistem mencatat secara berurutan parameter fisik komputasi riil yang meliputi total durasi waktu latih dalam satuan detik, volume transmisi data koordinasi parameter model dalam satuan Megabyte (MB), serta total konsumsi energi listrik perangkat dalam satuan kiloWatt-hour (kWh).

2. ***Centralized Learning***

Eksperimen ini bertujuan untuk mengukur performa komputasi dan kualitas model tradisional pada lingkungan server pusat sebagai standar pembanding (*baseline*) langsung terhadap pendekatan desentralisasi. Proses pengujian dilakukan dengan cara menjalankan simulasi pelatihan sentralisasi dalam satu siklus komputasi terpusat, dengan variasi jumlah *epoch* yaitu 10, 20, 30, 40, dan 50 *epoch* untuk mengamati pengaruhnya terhadap performa model. Pada setiap *epoch* pelatihan ini, sistem akan mencatat secara berkala metrik performa kecerdasan model yang meliputi nilai *loss*, *accuracy*, *validation loss*, dan *validation accuracy* murni pada sisi server pusat. Bersamaan dengan itu, sistem mencatat secara berurutan parameter fisik komputasi riil yang meliputi total durasi waktu latih dalam satuan detik, volume transmisi data akumulasi *bandwidth* jaringan yang dihabiskan untuk mentransmisikan seluruh data citra mentah dari tingkat lokal ke server pusat dalam satuan Megabyte (MB), serta total konsumsi energi listrik perangkat server dalam satuan kiloWatt-hour (kWh).

4.1.2 Skenario Pengujian Inferensi

1. ***Real-Life***

Skenario pengujian inferensi dirancang untuk memvalidasi keandalan operasional model global hasil pelatihan terdistribusi secara langsung pada perangkat *edge* saat mengeksekusi layanan verifikasi kehadiran mahasiswa. Pengujian pada tahap ini menggunakan model hasil pelatihan dari skenario 10 ronde dengan 1 *epoch* lokal untuk *Federated Learning*, serta 10 *epoch* untuk *Centralized Learning*, sebagai representasi konfigurasi dasar (*baseline*) dari masing-masing pendekatan. Pengujian pada tahap ini difokuskan pada penilaian fungsionalitas sistem secara biner dengan parameter keputusan berupa status Berhasil atau Gagal melalui serangkaian eksperimen terkontrol. Proses evaluasi terkontrol dilakukan dengan memetakan kemampuan pengenalan wajah pada variasi rentang jarak pemindaian subjek, mulai dari 0,5 meter hingga 3 meter, di bawah dua kondisi visual ruangan dengan tingkat intensitas cahaya yang diukur secara spesifik, yaitu kondisi pencahayaan cukup sebesar 341,01 lux dan pencahayaan minim sebesar 47,54 lux. Status pengujian dinyatakan Berhasil apabila sistem pada perangkat keras lokal mampu mendeteksi keberadaan objek wajah secara presisi serta memiliki kemampuan inferensi lintas perangkat untuk mengenali mahasiswa yang terdaftar di perangkat *edge* lain berkat transfer kecerdasan model global. Sebaliknya, pengujian

dikategorikan Gagal jika sistem salah mengenali identitas, memicu status tidak dikenal (*UNKNOWN*) pada data terdaftar, atau mengalami pemutusan waktu pemrosesan (*timeout*) komputasi.

Untuk membandingkan performa secara adil antara *Centralized Learning* dan terdistribusi *Federated Learning*, pengujian inferensi ini dilakukan pada empat skenario konfigurasi perangkat operasional yang dirangkum pada Tabel 4.2 sebagai berikut:

Tabel 4.2 Skenario Konfigurasi Perangkat

Skenario	**Keterangan**
Centralized Learning Client 1	Perangkat *edge* Jetson Nano yang menjalankan model hasil *Centralized Learning*.
Centralized Learning Client 2	Perangkat *edge* Raspberry Pi yang menjalankan model hasil *Centralized Learning*.
Federated Learning A	Perangkat *edge* Jetson Nano yang menjalankan model hasil *Federated Learning* dengan data dari Jetson Nano (registrasi dilakukan di *Client* 1 dan absensi dilakukan di *Client* 1).
Federated Learning B	Perangkat *edge* Raspberry Pi yang menjalankan model hasil *Federated Learning* untuk melakukan absensi (registrasi dilakukan di *Client* 1 dan absensi dilakukan di *Client* 2).

Evaluasi fungsionalitas biner (Berhasil/Gagal) dicatat untuk setiap variasi jarak pemindaian. Sementara itu, pengukuran tingkat keberhasilan klasifikasi (akurasi), nilai *Cosine Similarity* (rata-rata, maksimum, dan minimum), serta durasi latensi komputasi inferensi (rata-rata, maksimum, dan minimum) dihitung berdasarkan data pencatatan dari 30 log riwayat inferensi terakhir pada masing-masing skenario perangkat.

2. Video Simulasi

Selanjutnya, pengujian dilakukan menggunakan berkas rekaman video yang berisi aktivitas mahasiswa melewati pintu untuk mensimulasikan keadaan kondisi nyata. Pengujian ini juga menggunakan model hasil pelatihan dari skenario 10 ronde 1 *epoch* untuk *Federated Learning* dan 10 *epoch* untuk *Centralized Learning*, sejalan dengan skenario pengujian *Real-Life*. Pengujian ini menggunakan 5 berkas video simulasi (Video A, B, C, D, dan E) dengan rincian skenario yang dirangkum pada Tabel 4.3. Cuplikan hasil pengujian video simulasi dapat dilihat pada Lampiran 8. Untuk menjaga agar performa perangkat keras *edge* lokal tidak mengalami kelebihan beban komputasi, pemrosesan video dikonfigurasi dengan melakukan deteksi wajah setiap 5 *frame* sekali, serta membatasi pengenalan wajah maksimal 3 wajah secara bersamaan dalam satu *frame*. Evaluasi pada skenario pengujian video ini dinilai berdasarkan tiga indikator utama, yaitu mahasiswa terdeteksi yang merupakan rasio antara jumlah mahasiswa di dalam berkas video yang berhasil dicatat kehadirannya oleh sistem dengan total mahasiswa di dalam video yang terdaftar di dalam basis data, salah absen yang merupakan jumlah orang luar (tidak terdaftar) atau mahasiswa lain yang tidak terdaftar namun salah dikenali sehingga masuk ke dalam daftar kehadiran mahasiswa, serta durasi dalam satuan detik yang merupakan waktu yang dibutuhkan oleh sistem untuk memproses keseluruhan aliran berkas video absensi dari awal hingga selesai.

Tabel 4.3 Skenario Pengujian Video Simulasi

Sumber Video	Kode	Keterangan
rec_feed_20260507_1150_010.mp4	A	Terdapat 2 mahasiswa yang terdaftar tidak mengenakan kacamata.
rec_feed_20260519_0957_003.mp4	B	Terdapat 1 mahasiswa yang terdaftar mengenakan kacamata.
rec_feed_20260521_1006_008.mp4	C	Terdapat 2 mahasiswa yang terdaftar dengan 1 mahasiswa tidak mengenakan kacamata dan 1 mahasiswa mengenakan kacamata tapi saat pengambilan data tidak mengenakan kacamata.
rec_feed_20260526_0956_003.mp4	D	Terdapat 2 mahasiswa yang terdaftar mengenakan kacamata.
rec_feed_20260526_0956_023.mp4	E	Terdapat 0 mahasiswa yang terdaftar tujuan untuk menguji ketahanan sistem terhadap kasus salah absen

4.2 Hasil Uji Coba

4.2.1 Hasil Eksperimen Pelatihan Model

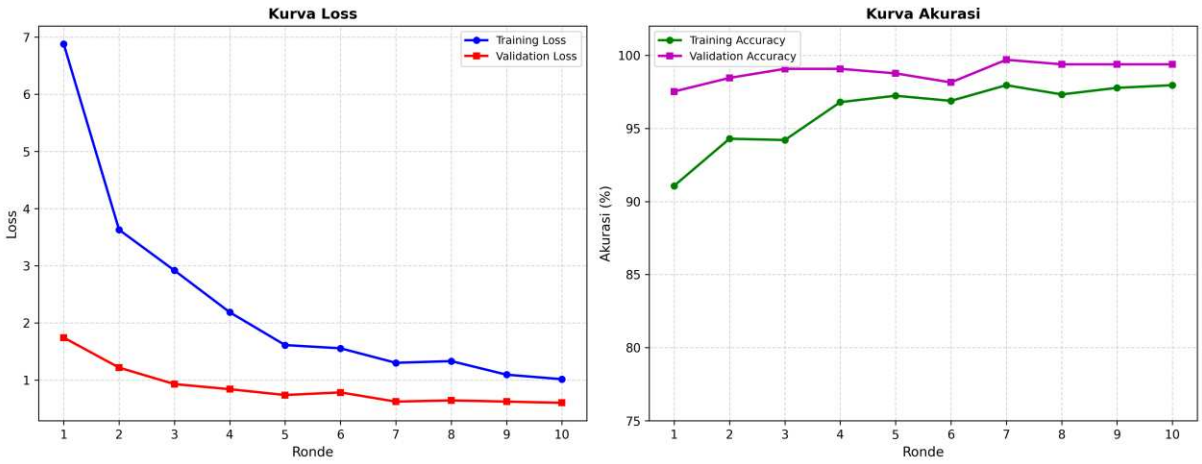
Pengujian performa komputasi dan karakteristik pembaruan bobot model dilakukan untuk membandingkan efisiensi sumber daya serta beban ekosistem antara dua metode pembelajaran yang berbeda. Arsitektur jaringan, parameter optimasi, serta batas atas eksekusi komputasi lokal disetarakan secara adekuat guna menjamin keadilan komparasi hasil performa latih.

1. *Federated Learning*

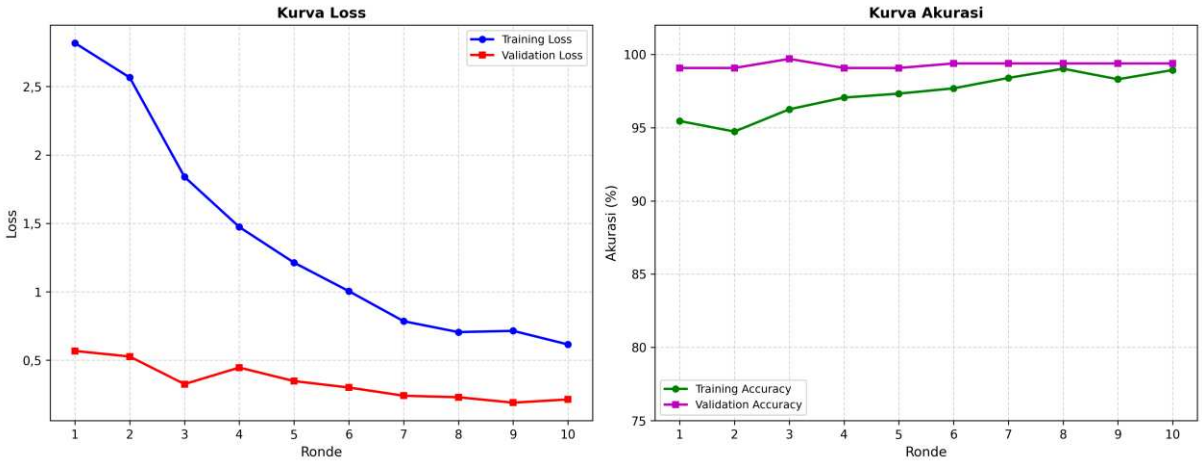
Hasil pencatatan metrik kecerdasan model pada skenario pelatihan desentralisasi terdistribusi disajikan secara terpisah berdasarkan peran masing-masing *node* infrastruktur. Nilai *loss* dan *accuracy* global pada sisi server untuk kelima variasi *epoch* lokal pada ronde ke-10 (terakhir) dirangkum pada Tabel 4.4. Kurva konvergensi masing-masing variasi ditunjukkan pada Gambar 4.1 untuk 1 *epoch*, Gambar 4.2 untuk 2 *epoch*, Gambar 4.3 untuk 3 *epoch*, Gambar 4.4 untuk 4 *epoch*, dan Gambar 4.5 untuk 5 *epoch*. Data lengkap performa global per ronde disajikan pada Lampiran 1.

Tabel 4.4 Perbandingan Performa Global server pada Ronde ke-10

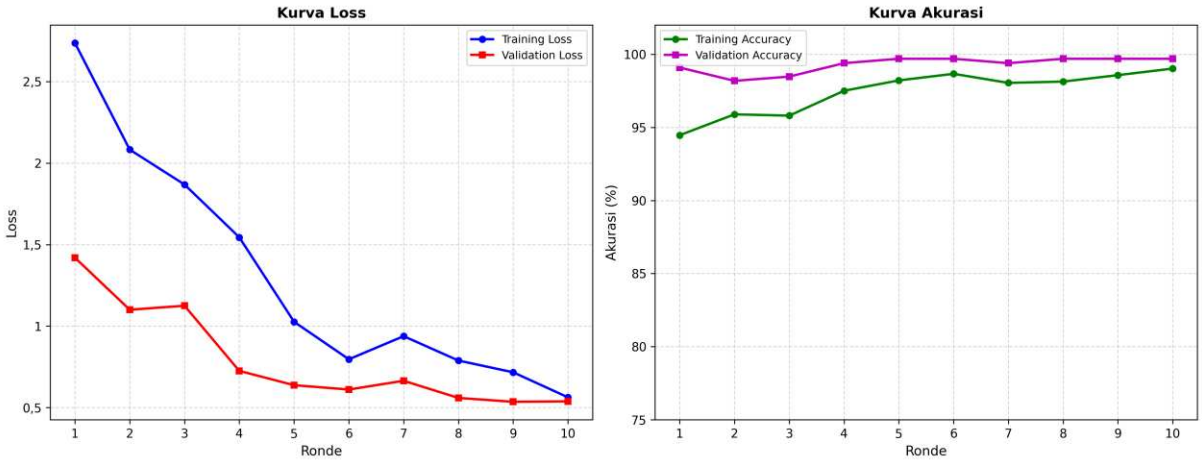
Variasi *Epoch*	*Loss*	*Accuracy* (%)	*Validation Loss*	*Validation Accuracy* (%)	Durasi (detik)
1	1,0141	97,95	0,6009	99,38	560,69
2	0,6148	98,93	0,2130	99,38	983,26
3	0,5630	99,02	0,5383	99,69	1436,80
4	0,6417	99,29	0,9174	99,40	2085,22
5	0,7399	98,81	0,3388	99,31	4061,56



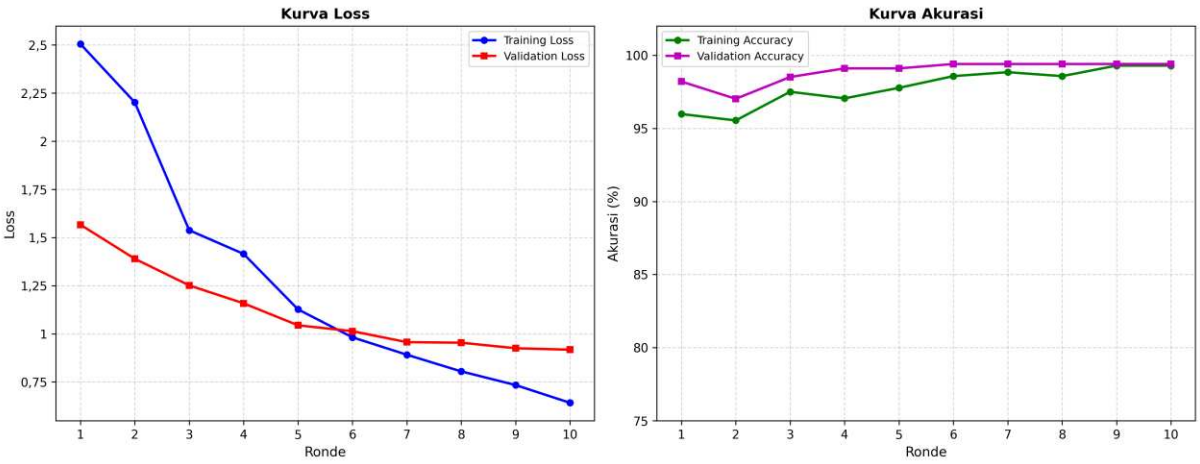
Gambar 4.1 Grafik Performa Pelatihan *Federated Learning* (1 *Epoch*)



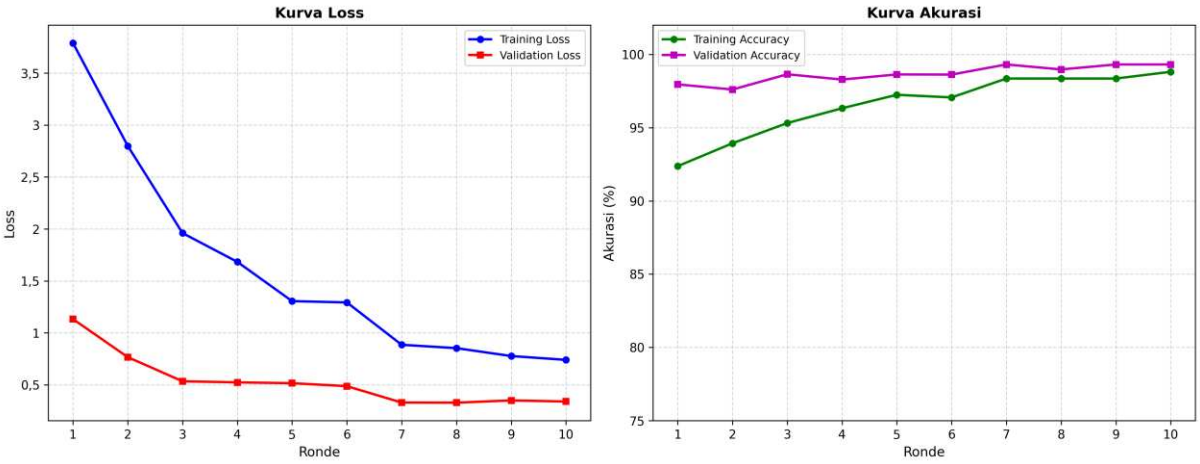
Gambar 4.2 Grafik Performa Pelatihan *Federated Learning* (2 *Epoch*)



Gambar 4.3 Grafik Performa Pelatihan *Federated Learning* (3 *Epoch*)



Gambar 4.4 Grafik Performa Pelatihan *Federated Learning* (4 *Epoch*)



Gambar 4.5 Grafik Performa Pelatihan *Federated Learning* (5 *Epoch*)

Selain performa pada sisi server, fluktuasi metrik *loss*, *accuracy*, *validation loss*, dan *validation accuracy* hasil komputasi lokal turut dicatat pada masing-masing perangkat *edge*. Hasil akhir performa lokal pada ronde ke-10 untuk perangkat Jetson Nano (*Client* 1) dirangkum pada Tabel 4.5, sedangkan hasil akhir performa lokal untuk perangkat Raspberry Pi (*Client* 2) dirangkum pada Tabel 4.6. Data lengkap performa lokal per ronde dan per *epoch* pada kedua perangkat dapat dilihat pada Lampiran 2 dan Lampiran 3.

Tabel 4.5 Perbandingan Performa lokal *Client* 1 pada Ronde ke-10 (Epoch Terakhir)

Variasi *Epoch*	***Loss***	***Accuracy* (%)**	***Validation Loss***	***Validation Accuracy* (%)**	**Durasi (detik)**
1	0,9472	98,35	0,5839	99,37	149,22
2	0,6516	99,08	0,2142	99,37	298,76
3	0,5587	99,08	0,6822	99,37	442,95
4	0,6284	99,45	0,7754	99,37	596,74
5	0,8421	98,35	0,2587	99,30	753,93

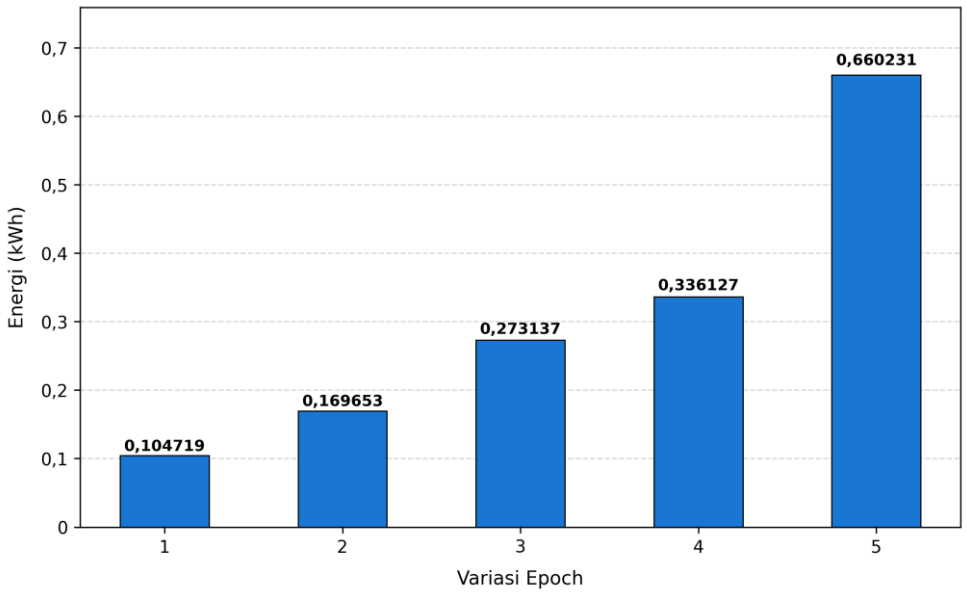
Tabel 4.6 Perbandingan Performa lokal *Client* 2 pada Ronde ke-10 (Epoch Terakhir)

Variasi *Epoch*	*Loss*	*Accuracy* (%)	*Validation Loss*	*Validation Accuracy* (%)	Durasi (detik)
1	1,0774	97,57	0,6170	99,39	444,35
2	0,5801	98,78	0,2118	99,39	868,20
3	0,5670	98,96	0,4025	100,00	1318,01
4	0,6541	99,13	1,0516	99,43	1952,53
5	0,6376	99,26	0,4189	99,32	3854,09

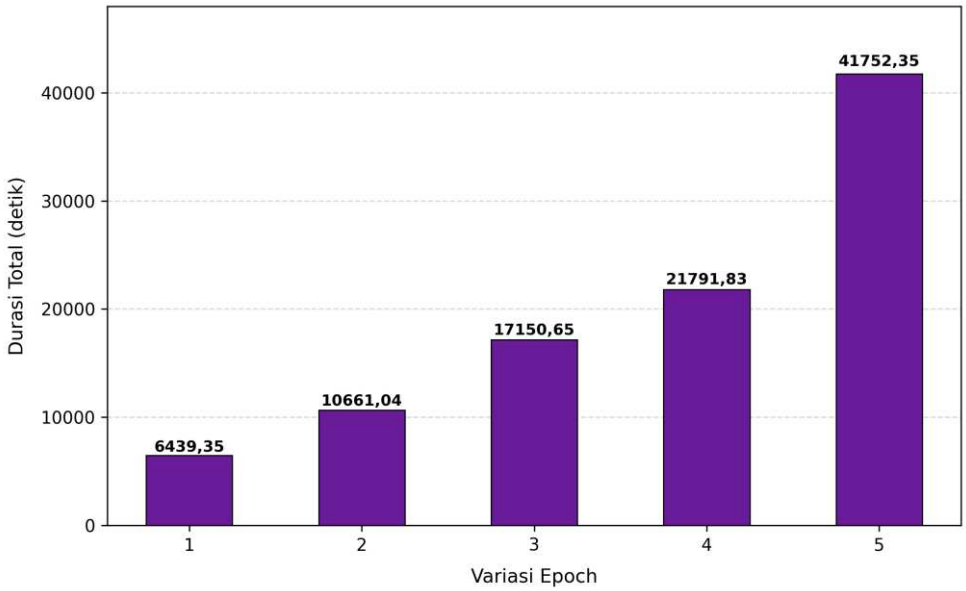
Selanjutnya, untuk mengevaluasi efisiensi sumber daya dari skema *Federated Learning*, parameter fisik riil terkait konsumsi sumber daya dari proses pelatihan, yang meliputi estimasi volume transmisi data (*bandwidth*), konsumsi energi listrik, serta durasi total pelatihan, dirangkum pada Tabel 4.7. Pola konsumsi energi dan durasi total terhadap variasi *epoch* lokal masing-masing diilustrasikan pada Gambar 4.6 dan Gambar 4.7.

Tabel 4.7 Perbandingan Nilai Ekonomi *Federated Learning*

Variasi *Epoch*	*Bandwidth* (MB)	Energi (kWh)	Durasi Total (detik)
1	167,28	0,104719	6439,35
2	167,28	0,169653	10661,04
3	167,28	0,273137	17150,65
4	167,28	0,336127	21791,83
5	167,28	0,660231	41752,35



Gambar 4.6 Grafik Konsumsi Energi *Federated Learning*



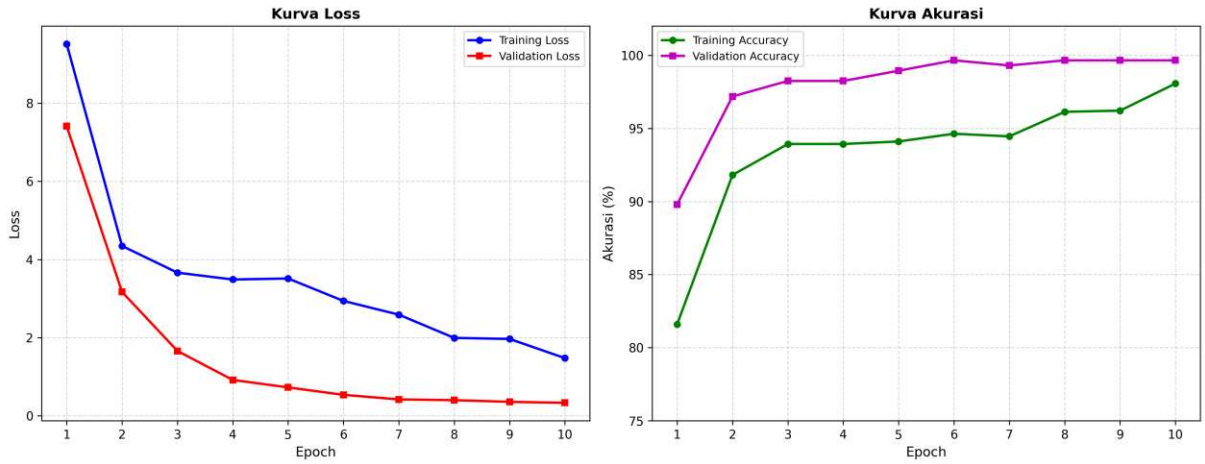
Gambar 4.7 Grafik Durasi Total Pelatihan *Federated Learning*

2. *Centralized Learning*

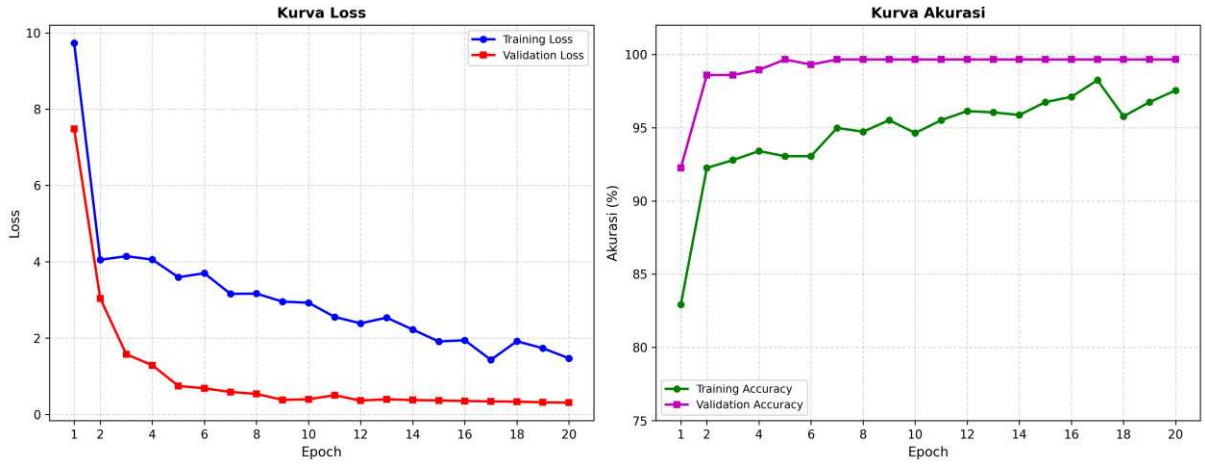
Sebagai standar tolok ukur (*baseline*), akumulasi data performa latih tradisional terpusat dirangkum ke dalam dua bagian tabel utama. Nilai *loss*, *accuracy*, *validation loss*, dan *validation accuracy* murni pada sisi server pusat untuk kelima variasi *epoch* pada *epoch* terakhir dirangkum pada Tabel 4.8. Kurva konvergensi masing-masing variasi ditunjukkan pada Gambar 4.8 untuk 10 *epoch*, Gambar 4.9 untuk 20 *epoch*, Gambar 4.10 untuk 30 *epoch*, Gambar 4.11 untuk 40 *epoch*, dan Gambar 4.12 untuk 50 *epoch*. Data lengkap performa global per *epoch* disajikan pada Lampiran 4.

Tabel 4.8 Perbandingan Performa *Centralized Learning* pada *Epoch* Terakhir

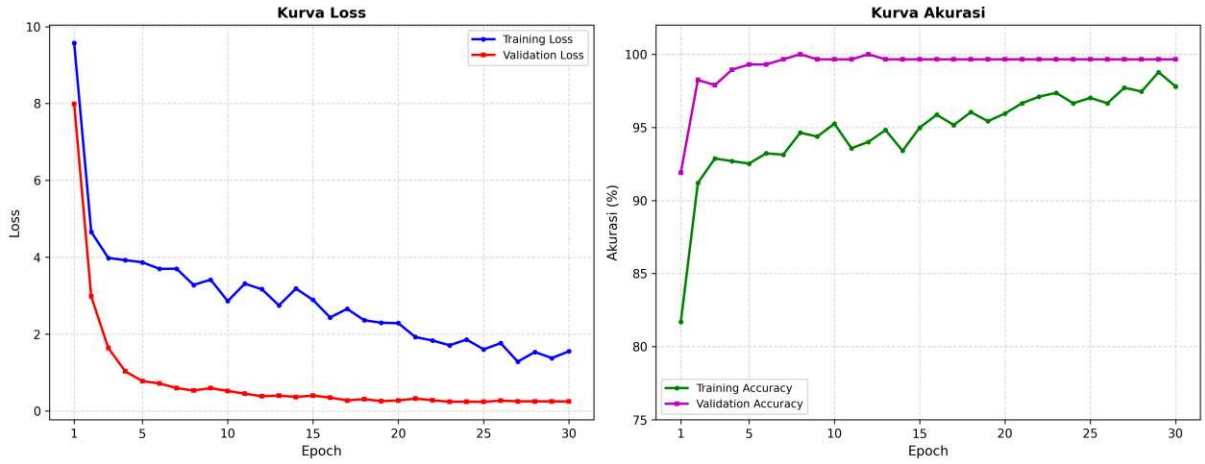
Variasi *Epoch*	***Loss***	***Accuracy* (%)**	***Validation Loss***	***Validation Accuracy* (%)**	**Durasi (detik)**
10	1,4730	98,06	0,3286	99,65	41,08
20	1,4697	97,54	0,3066	99,65	41,19
30	1,5526	97,80	0,2469	99,65	41,55
40	1,2321	97,89	0,2641	99,65	41,62
50	1,3416	97,71	0,2999	99,65	41,22



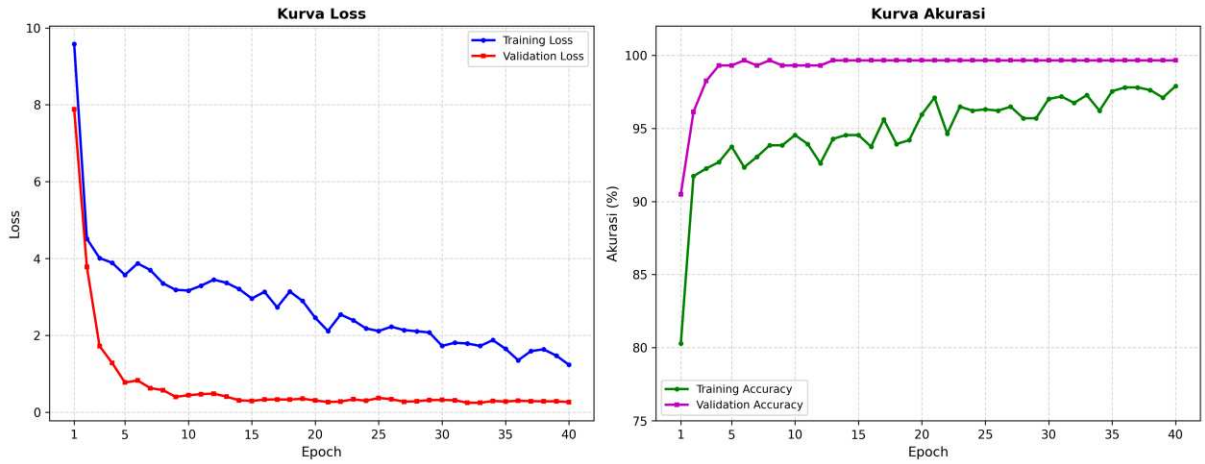
Gambar 4.8 Grafik Performa Pelatihan *Centralized Learning* (10 *Epoch*)



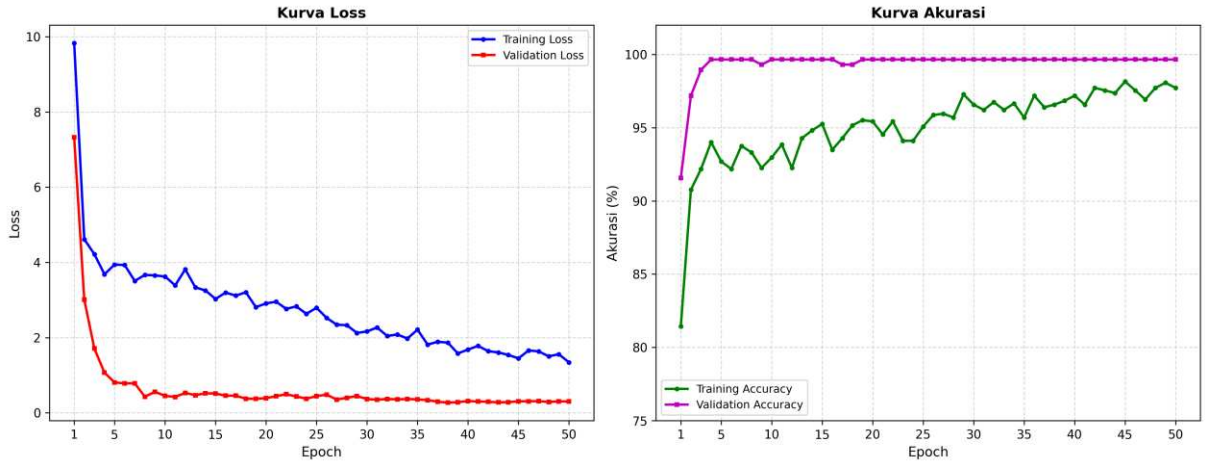
Gambar 4.9 Grafik Performa Pelatihan *Centralized Learning* (20 *Epoch*)



Gambar 4.10 Grafik Performa Pelatihan *Centralized Learning* (30 *Epoch*)



Gambar 4.11 Grafik Performa Pelatihan *Centralized Learning* (40 *Epoch*)

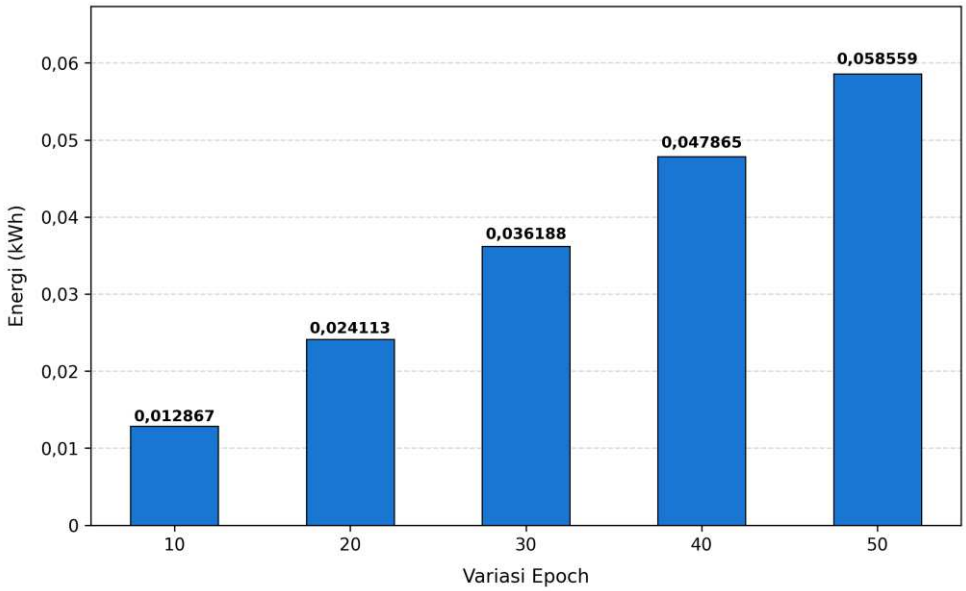


Gambar 4.12 Grafik Performa Pelatihan *Centralized Learning* (50 *Epoch*)

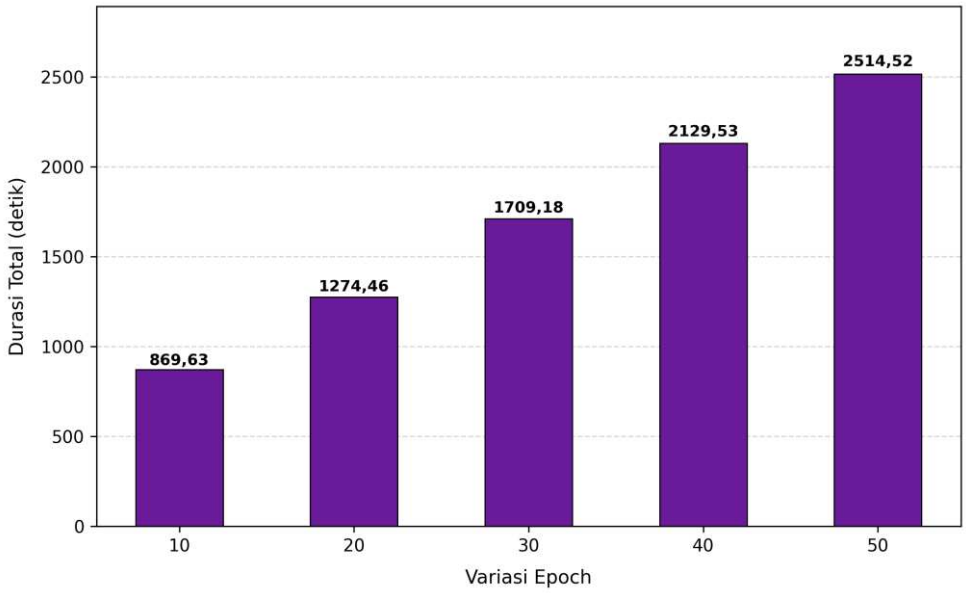
Selanjutnya, untuk mengevaluasi efisiensi sumber daya dari skema *Centralized Learning*, parameter fisik riil terkait konsumsi sumber daya dari proses pelatihan, yang meliputi estimasi volume transmisi data (*bandwidth*), konsumsi energi listrik, serta durasi total pelatihan, dirangkum pada Tabel 4.9. Pola konsumsi energi dan durasi total terhadap variasi *epoch* masing-masing diilustrasikan pada Gambar 4.13 dan Gambar 4.14.

Tabel 4.9 Perbandingan Nilai Ekonomi *Centralized Learning*

Variasi *Epoch*	*Bandwidth* (MB)	Energi (kWh)	Durasi Total (detik)
10	744,99	0,012867	869,63
20	744,99	0,024113	1274,46
30	744,99	0,036188	1709,18
40	744,99	0,047865	2129,53
50	744,99	0,058559	2514,52



Gambar 4.13 Grafik Konsumsi Energi *Centralized Learning*



Gambar 4.14 Grafik Konsumsi Energi *Centralized Learning*

4.2.2 Hasil Pengujian Inferensi

Pengujian inferensi dilakukan untuk memvalidasi keandalan operasional model hasil pelatihan secara langsung pada perangkat keras lokal. Pengujian ini dibagi menjadi pengujian fungsionalitas pada kondisi nyata serta pengujian berbasis berkas rekaman video.

1. *Real-Life*

Hasil pengujian fungsionalitas inferensi di bawah kondisi pencahayaan cukup disajikan pada Tabel 4.10, sedangkan hasil pada pencahayaan minim dirangkum pada Tabel 4.11. Evaluasi terhadap tingkat keberhasilan (akurasi) beserta durasi latensi inferensi pada kondisi pencahayaan cukup disajikan pada Tabel 4.12, dan hasil pengujian pada kondisi pencahayaan

minim disajikan pada Tabel 4.13. Data lengkap hasil pengujian pada kondisi pencahayaan cukup dan minim dapat dilihat pada Lampiran 5 dan Lampiran 6.

Tabel 4.10 Hasil Pengujian Inferensi Cukup Cahaya

Jarak (m)	***Centralized Learning***	***Federated Learning***	
		A	**B**
0,5	Berhasil	Berhasil	Berhasil
1	Berhasil	Berhasil	Berhasil
1,5	Berhasil	Berhasil	Berhasil
2	Berhasil	Berhasil	Berhasil
2,5	Berhasil	Berhasil	Berhasil
3	Berhasil	Berhasil	Berhasil

Tabel 4.11 Hasil Pengujian Inferensi Minim Cahaya

Jarak (m)	***Centralized Learning***	***Federated Learning***	
		A	**B**
0,5	Berhasil	Berhasil	Berhasil
1	Berhasil	Berhasil	Berhasil
1,5	Berhasil	Berhasil	Berhasil
2	Berhasil	Berhasil	Berhasil
2,5	Berhasil	Berhasil	Berhasil
3	Berhasil	Berhasil	Berhasil

Tabel 4.12 Hasil Keberhasilan, Akurasi, dan Latensi pada Kondisi Cukup Cahaya

Skenario	**Keberhasilan (%)**	***Cosine Similarity***			***Latency* (ms)**		
		Rata-rata	***Max***	***Min***	**Rata-rata**	***Max***	***Min***
***Centralized Learning Client* 1**	93	0,7817	0,8403	0,6635	724,97	793	679
***Centralized Learning Client* 2**	93	0,8161	0,8925	0,3749	3094,93	3344	2881
***Federated Learning* A**	97	0,8360	0,9414	0,6785	764,60	2190	684
***Federated Learning* B**	100	0,8681	0,9477	0,7868	3172,30	3361	2942

Tabel 4.13 Hasil Keberhasilan, Akurasi, dan Latensi pada Kondisi Minim Cahaya

Skenario	Keberhasilan (%)	*Cosine Similarity*			*Latency* (ms)		
		Rata-rata	*Max*	*Min*	Rata-rata	*Max*	*Min*
***Centralized Learning Client* 1**	97	0,8031	0,8653	0,5733	737,23	799	678
***Centralized Learning Client* 2**	93	0,8142	0,8898	0,2866	3231,83	3671	2411
***Federated Learning* A**	97	0,8248	0,9024	0,6799	719,13	766	680
***Federated Learning* B**	90	0,8607	0,9424	0,6070	3147,57	3328	2950

2. Video Simulasi

Hasil pengujian inferensi pelacakan multi-wajah secara kontinu menggunakan berkas rekaman video simulasi disajikan pada Tabel 4.14. Data lengkap hasil pengujian video simulasi dapat dilihat pada Lampiran 7.

Tabel 4.14 Hasil Pengujian Inferensi Video Simulasi

Skenario Video	Indikator	*Centralized Learning Client* 1	*Centralized Learning Client* 2	*Federated Learning* A	*Federated Learning* B
A	**Mahasiswa Terdeteksi**	2/2	2/2	2/2	2/2
	Salah Absen	0	0	2	2
	Durasi (detik)	835,83	1795,57	811,89	1790,22
B	**Mahasiswa Terdeteksi**	1/1	1/1	1/1	1/1
	Salah Absen	0	0	0	0
	Durasi (detik)	454,28	1044,80	486,58	1030,19
C	**Mahasiswa Terdeteksi**	1/2	1/2	1/2	1/2
	Salah Absen	0	0	0	0
	Durasi (detik)	364,06	850,78	392,94	843,67
D	**Mahasiswa Terdeteksi**	2/2	2/2	2/2	2/2
	Salah Absen	0	0	1	2
	Durasi (detik)	553,52	1216,38	562,58	1201,21
E	**Mahasiswa Terdeteksi**	0/0	0/0	0/0	0/0
	Salah Absen	0	0	0	0
	Durasi (detik)	388,20	888,18	390,55	878,88

4.3 Pembahasan

4.3.1 Pembahasan Eksperimen Pelatihan Model

Berdasarkan hasil eksperimen pelatihan model yang disajikan pada sub-bab 4.2.1, dengan variasi *local epoch* 1 hingga 5 pada skema *Federated Learning* dan variasi *epoch* 10 hingga 50 pada skema *Centralized Learning*, terdapat beberapa temuan yang dapat dianalisis, di antaranya pola *loss* dan akurasi data latih, *validation loss* dan *validation accuracy*, nilai ekonomi (*bandwidth*, energi, dan durasi pelatihan), serta perbedaan karakteristik antara kedua skema pembelajaran tersebut.

Dari segi kualitas model pada skema *Federated Learning*, ditemukan pola yang konsisten di seluruh variasi *local epoch*, yaitu nilai *loss* data latih selalu lebih tinggi dibandingkan *validation loss*, sementara akurasi data latih selalu sedikit lebih rendah dibandingkan *validation accuracy*. Sebagai contoh, pada *local epoch* 1 ronde ke-10, *loss* tercatat sebesar 1,0141 dengan akurasi 97,95%, sedangkan *validation loss* hanya 0,6009 dengan *validation accuracy* 99,38%. Pola serupa konsisten muncul pada seluruh variasi lainnya. Kesenjangan ini terjadi karena adanya perbedaan perlakuan data antara tahap pelatihan dan tahap validasi. Data latih diproses melalui jalur augmentasi acak *on the fly* yang cukup agresif, meliputi pembalikan horizontal, rotasi, distorsi perspektif, perubahan kecerahan dan kontras, konversi skala abu-abu, *gaussian blur*, penajaman acak, hingga *random erasing*, sehingga setiap *epoch* menghadapi variasi citra yang berbeda dan secara sengaja dibuat lebih sulit untuk mendorong generalisasi model. Sebaliknya, data validasi diproses melalui jalur transformasi terpisah yang jauh lebih bersih tanpa augmentasi acak, sehingga citra yang dievaluasi cenderung lebih representatif dan mudah dikenali. Selain itu, tahap validasi menerapkan teknik *Flip Trick*, yaitu merata-ratakan *embedding* dari citra asli dan citra hasil cerminan horizontal kemudian menormalisasikannya kembali, sehingga menghasilkan representasi fitur yang lebih stabil dibandingkan satu kali *forward pass* tunggal yang digunakan pada saat pelatihan. Faktor tambahan yang secara khusus memperbesar *loss* data latih pada *Federated Learning* adalah keberadaan komponen penalti *proximal term* dari algoritma FedProx, yang dijumlahkan langsung ke *loss* klasifikasi ArcFace pada setiap iterasi pelatihan lokal, namun tidak disertakan sama sekali pada perhitungan *validation loss*. Akibatnya, nilai *loss* pelatihan pada *Federated Learning* secara struktural memang dirancang untuk selalu tercatat lebih tinggi dibandingkan *validation loss* miliknya sendiri.

Pola fluktuasi nilai *validation loss* terhadap peningkatan *local epoch* juga perlu dicermati. *Validation loss* terendah justru dicapai pada *local epoch* 2 (0,2130), kemudian meningkat pada *local epoch* 3 dan 4 hingga mencapai puncaknya sebesar 0,9174, sebelum kembali turun pada *local epoch* 5. Kontradiksi antara *validation loss* yang naik namun *validation accuracy* yang tetap tinggi dan stabil pada rentang 99,31% hingga 99,69% mengindikasikan bahwa model tetap mampu mengklasifikasikan mayoritas data dengan tepat, tetapi tingkat keyakinan (*confidence*) prediksinya menjadi kurang terkalibrasi. Fenomena ini diduga kuat disebabkan oleh efek *client drift*, yaitu kondisi ketika setiap *client* dengan sebaran data *Non-IID* (masing-masing hanya memegang data 15 mahasiswa yang berbeda) melakukan pembaruan bobot lokal yang semakin menyimpang dari titik acuan global seiring bertambahnya jumlah *local epoch* sebelum agregasi dilakukan. Karena proses agregasi pada arsitektur ini hanya menggabungkan parameter *backbone* dan sengaja tidak menyertakan statistik *Batch Normalization* (yang dikelola secara personal per *client* sesuai strategi pFedFace), penggabungan arah pembaruan bobot yang saling berbeda antar *client* berpotensi saling meniadakan sebagian arah

diskriminatif yang telah dipelajari masing-masing *client*, sehingga model global hasil agregasi menjadi kurang stabil meskipun keputusan akhirnya tetap benar.

Pada skema *Centralized Learning*, pola kesenjangan antara *loss* data latih dan *validation loss* juga konsisten muncul, bahkan dengan selisih yang secara proporsional lebih besar dibandingkan *Federated Learning*. Pada *epoch* 10, *loss* tercatat 1,4730 berbanding *validation loss* 0,3286, dan pada *epoch* 50 tercatat 1,3416 berbanding 0,2999. Selisih ini tidak dapat dijelaskan oleh *proximal term*, karena komponen tersebut memang tidak diterapkan pada skema *Centralized Learning*. Penyebab utamanya adalah perbedaan jumlah kelas yang dihadapi model pada saat pelatihan. Setiap *client* pada *Federated Learning* hanya menyelesaikan masalah klasifikasi atas 15 mahasiswa, sedangkan *Centralized Learning* menyelesaikan masalah klasifikasi atas keseluruhan 30 mahasiswa sekaligus dalam satu ruang kelas gabungan. Karena fungsi *ArcFace Margin Loss* dihitung berdasarkan distribusi probabilitas yang penyebutnya menjumlahkan seluruh kelas negatif yang ada, semakin besar jumlah kelas yang dihadapi maka semakin besar pula nilai *loss* mentah yang dihasilkan untuk tingkat keyakinan yang setara, sehingga wajar apabila nilai *loss* data latih *Centralized Learning* secara konsisten berada pada rentang yang lebih tinggi (sekitar 1,2 hingga 1,5) dibandingkan *Federated Learning* (sekitar 0,6 hingga 1,0), tanpa hal ini berarti proses konvergensinya lebih buruk.

Perlu ditegaskan pula bahwa nilai *loss* pada kedua skema tidak pernah mendekati nol meskipun akurasi telah mencapai 97% hingga 99%. Hal ini merupakan karakteristik bawaan dari *ArcFace Margin Loss* itu sendiri, bukan indikasi kegagalan konvergensi. Berbeda dengan fungsi *cross entropy* konvensional yang nilainya dapat mendekati nol seiring meningkatnya keyakinan prediksi, ArcFace menerapkan margin sudut aditif serta faktor penskalaan tetap pada logit kelas target sebelum perhitungan *softmax*, sehingga secara struktural tetap menyisakan nilai *loss* dasar yang tidak dapat hilang sepenuhnya. Margin ini sengaja dipertahankan agar model terus didorong membentuk batas antar kelas yang lebih lebar dan tegas dalam ruang fitur, karakteristik yang memang dibutuhkan untuk tugas pengenalan wajah berbasis kemiripan *embedding*, bukan sekadar klasifikasi tertutup. Adapun kehalusan tren penurunan *loss* dari *epoch* ke *epoch*, khususnya penurunan tajam pada *epoch* awal yang kemudian melandai menjelang *epoch* akhir, merupakan hasil kerja dari penjadwal laju pembelajaran *Cosine Decay*, yang menurunkan nilai *learning rate* dari 0,05 secara bertahap mengikuti kurva kosinus sehingga pembaruan bobot berlangsung agresif pada tahap awal dan semakin halus mendekati akhir jadwal pelatihan, mencegah osilasi nilai *loss* yang tajam menjelang konvergensi.

Dari segi durasi waktu pelatihan, kenaikan jumlah *local epoch* pada *Federated Learning* menyebabkan pertambahan waktu yang sangat signifikan, yaitu dari 6439,35 detik pada *local epoch* 1 menjadi 41752,35 detik pada *local epoch* 5, atau meningkat sekitar 6,5 kali lipat, melebihi rasio penambahan *epoch* itu sendiri (5 kali). Hal ini terjadi karena durasi total pada skema terdistribusi tidak hanya dipengaruhi oleh waktu komputasi murni, tetapi juga oleh waktu tunggu sinkronisasi pada setiap ronde akibat heterogenitas perangkat keras. Perangkat Raspberry Pi (*Client* 2) yang hanya memiliki RAM 1 GB secara konsisten menjadi penghambat utama, dengan durasi pelatihan lokal yang terus memanjang seiring bertambahnya *local epoch* per ronde, bahkan mencapai lebih dari 3800 detik per ronde pada *local epoch* 5, jauh lebih lambat dibandingkan Jetson Nano (*Client* 1) yang berkapasitas RAM 4 GB dan hanya membutuhkan sekitar 750 detik per ronde pada pengaturan yang sama. Karena server harus menunggu *client* paling lambat menyelesaikan seluruh *epoch* lokalnya sebelum agregasi dapat dilakukan, penambahan *local epoch* berdampak berlipat ganda terhadap waktu tunggu

keseluruhan sistem. Sebaliknya, pada *Centralized Learning*, durasi total hanya meningkat secara sublinear, dari 869,63 detik pada *epoch* 10 menjadi 2514,52 detik pada *epoch* 50 (sekitar 2,9 kali untuk kenaikan 5 kali *epoch*), karena durasi per *epoch* relatif konstan di kisaran 41 detik dan sebagian besar durasi total berasal dari *overhead* tetap seperti pemuatan *dataset* yang hanya terjadi satu kali di awal proses.

Dari segi kebutuhan *bandwidth*, kedua skema menunjukkan nilai yang konstan pada seluruh variasi *epoch*, yaitu 167,28 MB untuk *Federated Learning* dan 744,99 MB untuk *Centralized Learning*. Hal ini terjadi karena volume transmisi data pada *Federated Learning* ditentukan oleh jumlah ronde komunikasi (tetap 10 ronde) dan ukuran parameter bobot model yang dikirim, bukan oleh banyaknya *local epoch* yang dijalankan secara internal pada masing-masing *client*. Demikian pula pada *Centralized Learning*, seluruh *dataset* citra mentah hanya diunggah satu kali ke server pusat sebelum pelatihan dimulai, sehingga penambahan jumlah *epoch* tidak menambah beban transmisi jaringan sama sekali. *Federated Learning* tetap unggul signifikan dalam efisiensi *bandwidth*, hanya menggunakan sekitar 22% dari kebutuhan *bandwidth Centralized Learning*, karena hanya parameter bobot yang dipertukarkan, bukan data citra mentah yang berukuran jauh lebih besar.

Dari segi konsumsi energi, pola kenaikan konsumsi daya sejalan dengan pola durasi pada masing-masing skema. *Federated Learning* mengalami kenaikan energi dari 0,104719 kWh pada *local epoch* 1 menjadi 0,660231 kWh pada *local epoch* 5, meningkat sekitar 6,3 kali lipat, sebanding dengan kenaikan durasinya yang juga superlinear, karena seluruh perangkat *edge* tetap aktif melakukan komputasi selama durasi yang memanjang tersebut. Sebaliknya, *Centralized Learning* hanya mengalami kenaikan energi dari 0,012867 kWh pada *epoch* 10 menjadi 0,058559 kWh pada *epoch* 50, meningkat sekitar 4,6 kali lipat, karena komputasi hanya dibebankan pada satu perangkat server berkinerja tinggi dalam durasi yang jauh lebih singkat, sementara perangkat *client* lokal berada dalam kondisi diam setelah proses pengunggahan *dataset* selesai. Secara keseluruhan, meskipun *Federated Learning* unggul dalam efisiensi *bandwidth*, metode ini harus dibayar dengan konsumsi energi dan durasi pelatihan yang jauh lebih besar dibandingkan *Centralized Learning*, terutama akibat adanya perangkat *edge* dengan spesifikasi rendah seperti Raspberry Pi yang menjadi titik hambatan utama dalam skema pelatihan terdistribusi ini.

4.3.2 Pembahasan Pengujian Inferensi

Berdasarkan hasil pengujian inferensi yang disajikan pada sub-bab 4.2.2, terdapat beberapa temuan penting yang perlu dianalisis secara mendalam, tidak hanya terkait keandalan operasional model di lapangan, tetapi juga akar penyebab teknis di balik pola hasil yang muncul. Pengujian inferensi ini secara spesifik mengevaluasi model hasil pelatihan skenario dasar (*baseline*) dari kedua skema, yaitu model hasil pelatihan 10 ronde dengan 1 *local epoch* untuk *Federated Learning*, serta model hasil pelatihan 10 *epoch* untuk *Centralized Learning*. Aspek analisis difokuskan pada keandalan pengenalan di bawah variasi intensitas cahaya, analisis kasus salah absen (*false positive*), pengaruh heterogenitas perangkat keras terhadap latensi komputasi inferensi, serta sensitivitas model terhadap perubahan atribut visual pengguna.

1. *Real-Life*

Pengujian inferensi pada skenario dunia nyata (*real-life*) menggunakan batas ambang kepercayaan (*confidence threshold*) sebesar 0,7. Hasil pengujian menunjukkan bahwa model global hasil pelatihan terdistribusi memiliki tingkat keandalan klasifikasi biner (Berhasil/Gagal) yang setara dengan model terpusat. Pada kondisi pencahayaan cukup (341,01 lux) maupun minim cahaya (47,54 lux), seluruh skenario pengujian verifikasi kehadiran pada jarak 0,5 meter hingga 3 meter dinyatakan berhasil mendeteksi dan mengenali wajah mahasiswa secara tepat.

Evaluasi lebih mendalam berdasarkan pencatatan dari 30 log riwayat inferensi terakhir pada kondisi pencahayaan cukup menunjukkan tingkat keberhasilan klasifikasi yang tinggi untuk seluruh skenario, yakni berada di atas 90% (93% untuk *Centralized Learning Client* 1 dan 2, 97% untuk *Federated Learning* A, serta 100% untuk *Federated Learning* B). Nilai *Cosine Similarity* rata-rata yang dicapai oleh skenario *Federated Learning* (0,8360 untuk *Federated Learning* A dan 0,8681 untuk *Federated Learning* B) bahkan sedikit lebih tinggi dibandingkan hasil pencapaian *Centralized Learning* (0,7817 untuk *Centralized Learning Client* 1 dan 0,8161 untuk *Centralized Learning Client* 2). Perlu dicermati bahwa perbedaan ini bukan disebabkan oleh ukuran *database* lokal, karena baik pada skenario FL maupun CL, setiap perangkat *client* sama-sama hanya menampung *database* registrasi 15 mahasiswa yang dikumpulkan pada perangkat tersebut. Penyebab yang lebih mendasar terletak pada strategi personalisasi lapisan *Batch Normalization* yang diterapkan pada *Federated Learning*. Karena statistik BN tidak diikutsertakan dalam agregasi global dan tetap dikelola secara lokal di setiap *client* (mengikuti strategi pFedFace), lapisan ini secara bertahap terkalibrasi secara spesifik terhadap karakteristik kamera dan kondisi pencahayaan perangkat itu sendiri, sehingga *embedding* wajah yang dihasilkan menjadi lebih rapat dan bersih secara statistik untuk domain lokal tersebut, menghasilkan skor kemiripan sudut kosinus yang lebih tinggi dan tegas ketika mencocokkan wajah dengan referensi lokalnya sendiri. Sebaliknya, model *Centralized Learning* menggunakan satu set statistik BN tunggal yang dioptimasi sebagai kompromi rata-rata dari kondisi kedua kamera perangkat sekaligus, sehingga kalibrasinya tidak sepenuhnya spesifik terhadap domain visual satu perangkat tertentu.

Daya tahan model terhadap penurunan intensitas cahaya yang drastis (dari 341,01 lux menjadi 47,54 lux) terbukti sangat tangguh. Tingkat keberhasilan klasifikasi tetap stabil di atas 90% untuk semua skenario (97% untuk *Centralized Learning Client* 1 dan *Federated Learning* A, 93% untuk *Centralized Learning Client* 2, serta 90% untuk *Federated Learning* B), dengan nilai rata-rata *Cosine Similarity* yang tetap tinggi dan kompetitif (0,8248 untuk *Federated Learning* A dan 0,8607 untuk *Federated Learning* B). Ketahanan terhadap variasi pencahayaan ini merupakan hasil langsung dari desain pipa pemrosesan data latih. Pertama, penerapan augmentasi *Color Jitter* dan *Random Grayscale* secara acak saat pelatihan melatih model MobileFaceNet untuk mengabaikan variasi warna dan intensitas piksel absolut, sehingga model lebih berfokus pada fitur geometris wajah yang cenderung invarian terhadap cahaya. Kedua, deteksi titik penanda wajah (*facial landmarks*) oleh MTCNN melakukan penyelarasan sudut kemiringan wajah secara konsisten sebelum diekstraksi oleh MobileFaceNet, sehingga meminimalkan distorsi bayangan cahaya pada struktur wajah. Ketiga, penggunaan metode *Cosine Similarity* pada vektor yang telah dinormalisasi secara inheren mengukur arah (sudut) vektor fitur, bukan panjang magnitudonya, sehingga perubahan intensitas cahaya yang sebagian besar memengaruhi magnitudo nilai respons fitur tidak banyak berpengaruh terhadap hasil perbandingan.

Meskipun kualitas pengenalannya setara, terdapat perbedaan latensi komputasi inferensi yang sangat kontras di antara kedua perangkat *edge*. Perangkat Jetson Nano (*Centralized Learning Client* 1 dan *Federated Learning* A) memproses inferensi secara cepat dengan rata-rata waktu masing-masing sebesar 724,97 ms dan 764,60 ms pada kondisi cahaya cukup, serta 737,23 ms dan 719,13 ms pada kondisi minim cahaya. Sebaliknya, Raspberry Pi (*Centralized Learning Client* 2 dan *Federated Learning* B) membutuhkan waktu rata-rata yang jauh lebih lambat, yaitu 3094,93 ms dan 3172,30 ms pada cahaya cukup, serta 3231,83 ms dan 3147,57 ms pada minim cahaya. Kesenjangan latensi ini murni disebabkan oleh perbedaan spesifikasi perangkat keras, bukan oleh metode pelatihan. *Pipeline* inferensi yang berjalan pada perangkat lokal harus memuat sistem operasi, dependensi PyTorch, serta model MTCNN dan MobileFaceNet sekaligus ke dalam memori. Kapasitas RAM sebesar 1 GB pada Raspberry Pi 3 Model B tidak memadai untuk menampung seluruh alokasi memori tersebut, sehingga sistem operasi terpaksa melakukan *swapping* memori virtual ke MicroSD eksternal yang memiliki kecepatan baca tulis jauh lebih lambat, menciptakan *bottleneck* masukan/keluaran (*I/O*) yang melipatgandakan waktu inferensi hingga mencapai kisaran 3 detik. Sebaliknya, Jetson Nano dengan kapasitas RAM 4 GB mampu menampung seluruh *pipeline* inferensi di dalam memori fisik tanpa perlu melakukan pertukaran data ke penyimpanan sekunder, sehingga proses inferensi dapat diselesaikan dalam waktu kurang dari 1 detik.

2. Video Simulasi

Hasil pengujian menggunakan video simulasi pada Tabel 4.14 memperkuat temuan mengenai latensi perangkat keras. Durasi total pemrosesan video pada perangkat Jetson Nano (*Centralized Learning Client* 1 dan *Federated Learning* A) berkisar antara 364 hingga 835 detik untuk Video A, sedangkan Raspberry Pi (*Centralized Learning Client* 2 dan *Federated Learning* B) membutuhkan waktu berkisar antara 843 hingga 1795 detik. Meskipun kedua perangkat dikondisikan sama-sama hanya mengeksekusi inferensi pada CPU untuk menjaga keadilan uji coba, keterbatasan kapasitas RAM 1 GB pada Raspberry Pi yang memicu *memory swapping* tetap menjadi faktor penentu lambatnya eksekusi pengolahan video per *frame*.

Analisis terhadap kasus salah absen menunjukkan perbedaan karakteristik ketahanan yang penting antara kedua skema pembelajaran. Kasus salah absen terdeteksi pada Video A untuk skenario *Federated Learning* A dan B yang mencatatkan 2 kasus salah absen, serta Video D yang mencatatkan 1 kasus pada *Federated Learning* A dan 2 kasus pada *Federated Learning* B. Sebaliknya, metode *Centralized Learning* (baik *Client* 1 maupun *Client* 2) berhasil mencatatkan 0 kasus salah absen pada seluruh berkas video uji. Pada Video A, kesalahan absensi pada metode *Federated Learning* dipicu oleh keberadaan subjek mahasiswi berkerudung (*hijab*) yang salah dikenali sebagai mahasiswa terdaftar lain yang juga berkerudung, sementara model *Centralized Learning* mampu membedakan identitas subjek tersebut secara tepat.

Akar penyebab perbedaan ini bersifat algoritmik dan berkaitan langsung dengan cara data dipartisi selama pelatihan. Model *Centralized Learning* dilatih dengan mengakses seluruh 30 kelas mahasiswa secara bersamaan dalam setiap *batch*, sehingga fungsi *ArcFace Margin Loss* secara langsung mendorong pembentukan batas keputusan (*decision boundary*) yang mempertegas jarak sudut antara seluruh pasangan kelas yang ada, termasuk pasangan kelas dengan atribut visual yang mirip seperti sesama pengguna kerudung. Sebaliknya, pada *Federated Learning*, setiap *client* hanya pernah melatih model menggunakan citra dari 15

mahasiswa lokalnya sendiri di setiap ronde, meskipun dimensi lapisan klasifikasi (*head*) telah diperluas mencakup 30 kelas global untuk keperluan agregasi. Konsekuensinya, dua individu yang secara visual mirip namun kebetulan terdaftar pada dua *client* yang berbeda tidak pernah benar-benar dikontraskan secara langsung dalam satu proses optimisasi yang sama. Pemisahan antara keduanya di ruang *embedding* global hanya terbentuk secara tidak langsung melalui proses agregasi rata-rata bobot *backbone* antar-*client* (FedProx), yang secara inheren merupakan mekanisme pemisahan kelas yang jauh lebih lemah dibandingkan pelatihan kontrastif langsung seperti pada *Centralized Learning*. Hal ini menjelaskan mengapa *Federated Learning* lebih rentan menghasilkan kesalahan klasifikasi pada pasangan wajah dengan kemiripan atribut visual yang dominan, meskipun secara umum performa akurasinya tetap tinggi.

Mengenai indikator mahasiswa terdeteksi, pada Video C, seluruh skenario pengujian hanya berhasil mencatatkan rasio deteksi sebesar 1/2 mahasiswa terdaftar. Kegagalan sistem dalam mengenali salah satu mahasiswa tersebut disebabkan oleh perubahan atribut visual subjek pada saat pengujian. Mahasiswa yang bersangkutan tidak mengenakan kacamata selama proses pengambilan data latih awal (*data gathering*), namun mengenakan kacamata saat proses perekaman video uji coba presensi dilakukan. Penambahan atribut kacamata ini menghasilkan perubahan representasi spasial dan oklusi pada area sekitar mata, salah satu wilayah kontur geometris terpenting bagi ekstraksi fitur wajah. Karena model belum pernah dilatih untuk mengaitkan identitas subjek tersebut dengan variasi atribut kacamata, vektor *embedding* yang dihasilkan mengalami pergeseran posisi yang cukup jauh di dalam ruang representasi 128 dimensi. Pergeseran ini menyebabkan nilai *Cosine Similarity* yang dihitung berada di bawah ambang batas (*threshold*) kepercayaan 0,7, sehingga sistem mengklasifikasikan wajah subjek sebagai tidak dikenal (*UNKNOWN*). Temuan ini konsisten dengan kasus di Video A, dan sama-sama menegaskan bahwa keandalan sistem sangat bergantung pada seberapa representatif dan bervariasinya sampel data pendaftaran wajah (*enrollment*) yang mencakup atribut visual dinamis seperti kacamata, ekspresi, dan aksesoris lainnya, terlepas dari metode pelatihan yang digunakan.

[Halaman ini sengaja dikosongkan]

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, berikut merupakan kesimpulan yang didapatkan dari seluruh tahapan yang telah dilakukan:

1. Sistem kehadiran berbasis pengenalan wajah menggunakan konsep *Federated Learning* pada lingkungan *Edge Computing* telah berhasil dibangun. Sistem ini diimplementasikan menggunakan model *MobileFaceNet* di dalam *container Docker* pada perangkat Jetson Nano dan Raspberry Pi, dengan server agregator.
2. Perbandingan performa menunjukkan bahwa *Federated Learning* mampu menyeimbangi akurasi model *Centralized Learning* sekaligus menawarkan efisiensi *bandwidth* jaringan yang jauh lebih baik, namun harus dibayar dengan durasi pelatihan dan konsumsi energi yang lebih besar akibat keterbatasan perangkat keras *client*. Dari sisi keandalan inferensi, kedua metode sama-sama stabil pada kondisi ideal, tetapi *Centralized Learning* masih lebih unggul dalam mencegah kesalahan klasifikasi absen.

5.2 Saran

Berdasarkan keterbatasan sistem serta hasil evaluasi yang diperoleh sepanjang pelaksanaan penelitian ini, beberapa saran yang dapat diajukan untuk pengembangan penelitian selanjutnya adalah sebagai berikut:

1. Menggunakan arsitektur model pengenalan wajah alternatif yang lebih modern selain *MobileFaceNet* guna menganalisis perbandingan akurasi dan latensi inferensinya.
2. Menerapkan metode deteksi keaktifan wajah (*anti-spoofing*) untuk memperkuat keamanan sistem dari ancaman pemalsuan identitas menggunakan foto atau video.
3. Menggunakan perangkat keras *edge* dengan kapasitas RAM yang lebih tinggi sebagai prasyarat sebelum memperluas skala *dataset*, mengingat keterbatasan RAM pada Raspberry Pi menjadi faktor penghambat utama durasi pelatihan dan kemampuan sistem dalam menampung lebih banyak subjek mahasiswa.
4. Mengeksplorasi skenario pengumpulan data yang lebih mendekati asumsi *Federated Learning* pada literatur, yaitu dengan merekam data wajah langsung melalui kamera yang terpasang pada masing-masing perangkat *edge*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

About PostgreSQL. (n.d.). Retrieved November 21, 2025, from https://www.postgresql.org/about

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2022). *Flower: A Friendly Federated Learning Research Framework*. http://arxiv.org/abs/2007.14390

Brecko, A., Kajati, E., Koziorek, J., & Zolotova, I. (2022). Federated Learning for Edge Computing: A Survey. In *Applied Sciences (Switzerland)* (Vol. 12, Number 18). MDPI. https://doi.org/10.3390/app12189124

Cha, N., & Chang, L. (2024). *Distributed client selection with multi-objective in federated learning assisted Internet of Vehicles*. http://arxiv.org/abs/2401.03159

Gao, Y., Su, A., Zhao, C., & Song, J. (2025). pFedFace: Personalized Federated Learning for Face Recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP49660.2025.10889072

Gonzalez, V. Z., Tena-Sanchez, E., & Acosta, A. J. (2023). A Security Comparison between AES-128 and AES-256 FPGA implementations against DPA attacks. *2023 38th Conference on Design of Circuits and Integrated Systems, DCIS 2023*. https://doi.org/10.1109/DCIS58620.2023.10336003

Gupta, S., & Singh, P. (2025). Federated Edge Learning with Hybrid CNN-Transformers for Secure, High-Accuracy Face Recognition in Resource-Constrained IoT Systems. *International Conference on Computing, Intelligence, and Application, CIACON 2025*. https://doi.org/10.1109/CIACON65473.2025.11189632

Kim, J., Park, T., Kim, H., & Kim, S. (2021). Federated Learning for Face Recognition. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, *2021-January*. https://doi.org/10.1109/ICCE50685.2021.9427748

Mathur, A., Beutel, D. J., de Gusmão, P. P. B., Fernandez-Marques, J., Topal, T., Qiu, X., Parcollet, T., Gao, Y., & Lane, N. D. (2021). *On-device Federated Learning with Flower*. http://arxiv.org/abs/2104.03042

Nagamas, A. P. (2022). *Pengembangan sistem kehadiran menggunakan pengenalan wajah dengan atau tanpa masker untuk mencatat kehadiran siswa secara otomatis*. http://repository.its.ac.id/132870/

Nandi, A., Xhafa, F., & Kumar, R. (2023). A Docker-based federated learning framework design and deployment for multi-modal data stream classification. *Computing*, *105*(10), 2195–2229. https://doi.org/10.1007/s00607-023-01179-5

NVIDIA. (2019, October 13). *What Is Federated Learning?* https://blogs.nvidia.com/blog/what-is-federated-learning

Shaheen, M., Farooq, M. S., Umer, T., & Kim, B. S. (2022). Applications of Federated Learning; Taxonomy, Challenges, and Research Trends. *Electronics (Switzerland)*, *11*(4). https://doi.org/10.3390/electronics11040670

Shahreza, H. O., George, A., & Marcel, S. (2025). Face Reconstruction from Face Embeddings Using Adapter to a Face Foundation Model. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 5584–5593. https://doi.org/10.1109/CVPRW67362.2025.00555

Srinu, N., Kumar, C. S. S., Senthil, M., & Babu, D. B. (2025). Smart Attendence Recording System Utilizing CNN and Edge Computing Techniques. *2025 5th International Conference on Soft Computing for Security Applications (ICSCSA)*, 551–558. https://doi.org/10.1109/ICSCSA66339.2025.11170847

Staňo, M., Hluchý, L., Krammer, P., & Hucko, M. (2025). Docker Survey for FLOps Efficiency. *Proceedings of the International Conference on Cybernetics and Informatics, K and I*, (2025). https://doi.org/10.1109/KI64036.2025.10916451

Tertulino, R. (2025). *Centralized vs. Federated Learning for Educational Data Mining: A Comparative Study on Student Performance Prediction with SAEB Microdata*. http://arxiv.org/abs/2509.00086

Xiao, J., Wang, W., Zhang, L., & Liu, H. (2024). A MobileFaceNet-Based Face Anti-Spoofing Algorithm for Low-Quality Images. *Electronics (Switzerland)*, *13*(14). https://doi.org/10.3390/electronics13142801

Xiaoccer. (2023). *MobileFaceNet*. https://github.com/Xiaoccer/MobileFaceNet_Pytorch.git

Xiong, Y., Cao, J., & Chen, G. (2025). Federated Learning Spam Detection Based on FedProx and Multi-Level Multi-Feature Fusion. *Informatics*, *12*(3). https://doi.org/10.3390/informatics12030093

Yadav, D., Maniar, S., Sukhani, K., & Devadkar, K. (2021). In-Browser Attendance System using Face Recognition and Serverless Edge Computing. *2021 12th International Conference on Computing Communication and Networking Technologies, ICCCNT 2021*. https://doi.org/10.1109/ICCCNT51525.2021.9580042

Yurdakul, M., Ayan, E., Horasan, F., & Taşdemir, Ş. (n.d.). *A Mobile Application for Flower Recognition System Based on Convolutional Neural Networks*.

Zhang, N., Luo, J., & Gao, W. (2020). Research on face detection technology based on MTCNN. *Proceedings - 2020 International Conference on Computer Network, Electronic and Automation, ICCNEA 2020*, 154–158. https://doi.org/10.1109/ICCNEA50255.2020.00040

LAMPIRAN

Lampiran 1 Hasil Pelatihan *Federated Learning* Server

Server (10 Ronde, 1 *Epoch*)					
Ronde	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	6,8812	91,07	1,7420	97,52	556,76
2	3,6297	94,29	1,2168	98,45	560,02
3	2,9173	94,20	0,9280	99,07	561,84

Server (10 Ronde, 2 *Epoch*)					
Ronde	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	3,2567	94,11	1,7161	97,01	977,57
2	2,3287	95,09	1,3659	97,32	969,30
3	2,0968	95,80	1,2829	97,61	969,21

Server (10 Ronde, 3 *Epoch*)					
Ronde	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	2,7368	94,46	1,4197	99,09	1827,19
2	2,0825	95,89	1,1002	98,18	1847,47
3	1,8682	95,80	1,1249	98,47	1738,68

Server (10 Ronde, 4 *Epoch*)					
Ronde	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	2,5054	95,98	1,5665	98,20	2683,25
2	2,2019	95,54	1,3896	97,02	2127,63
3	1,5379	97,50	1,2515	98,51	1972,36

Server (10 Ronde, 5 *Epoch*)					
Ronde	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	3,7895	92,37	1,1325	97,95	4092,35
2	2,7992	93,93	0,7663	97,6	4108,74
3	1,9601	95,31	0,5337	98,64	4094,96

Log selangkapnya: https://its.id/m/HasilPelatihanFLServer

Lampiran 2 Hasil Pelatihan *Federated Learning Client* 1 (Jetson Nano)

***Client* 1 Jetson Nano (10 Ronde, 1 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	6,4918	92,28	1,4735	97,47	157,43
2	1	3,7779	94,12	1,2566	99,37	151,52
3	1	3,0864	93,57	0,8937	99,37	148,60

***Client* 1 Jetson Nano (10 Ronde, 2 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	6,5227	94,49	1,4476	98,10	306,79
	2	2,7627	95,40	0,5870	98,73	
2	1	2,7399	94,67	0,6684	99,37	297,81
	2	2,5003	95,04	0,5063	98,73	
3	1	2,2544	94,49	0,5156	98,73	297,82
	2	2,0599	95,22	0,3669	99,37	

***Client* 1 Jetson Nano (10 Ronde, 3 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	8,1352	89,52	1,9833	98,10	450,72
	2	3,3349	94,30	1,4098	96,84	
	3	2,6132	95,40	1,1581	99,37	
2	1	2,5908	95,59	1,1635	98,73	445,70
	2	2,3271	94,85	1,1813	98,73	
	3	1,8161	97,06	0,9615	98,73	
3	1	1,7223	96,69	0,9991	98,73	444,99
	2	1,5564	97,79	1,0066	98,10	
	3	1,7497	96,14	1,2771	97,47	

Client 1 Jetson Nano (10 Ronde, 4 *Epoch*)						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	7,1500	91,36	2,1529	98,10	615,77
	2	3,4307	93,01	1,2592	98,10	
	3	2,8966	95,22	1,4819	98,73	
	4	2,4041	96,32	1,3671	98,73	
2	1	2,0957	97,79	1,0921	99,37	595,37
	2	2,6240	95,22	1,0962	98,73	
	3	2,2228	95,59	1,3759	96,84	
	4	2,0159	96,14	1,0963	98,73	
3	1	1,8546	97,06	1,1041	99,37	594,50
	2	1,9938	95,77	1,0828	99,37	
	3	1,7132	96,51	1,0030	98,73	
	4	1,4580	98,16	0,9774	99,37	

Client 1 Jetson Nano (10 Ronde, 5 *Epoch*)						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	13,8197	85,11	5,4787	90,21	762,11
	2	6,7791	85,11	2,8785	95,80	
	3	3,7918	93,01	0,7352	98,60	
	4	3,3425	92,83	0,5811	98,60	
	5	3,1496	93,38	0,8509	99,30	
2	1	4,0886	91,36	0,6152	99,30	753,86
	2	3,1225	93,38	0,5931	98,60	
	3	3,0807	93,38	0,6954	98,60	
	4	3,0858	94,12	0,8071	98,60	
	5	2,9818	93,75	0,4862	98,60	
3	1	2,7053	95,40	0,3316	99,30	754,00
	2	2,2615	96,14	0,2330	99,30	
	3	2,0367	95,96	0,3535	99,30	
	4	1,6142	96,32	0,2382	99,30	
	5	1,9269	95,96	0,1822	100,00	

Log selangkapnya: https://its.id/m/HasilPelatihanFLClient1

Lampiran 3 Hasil Pelatihan *Federated Learning Client* 2 (Raspberry Pi)

***Client* 2 Raspberry Pi (10 Ronde, 1 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	7,2491	89,93	1,9956	97,58	440,22
2	1	3,4898	94,44	1,1792	97,58	443,42
3	1	2,7577	94,79	0,9603	98,79	445,21

***Client* 2 Raspberry Pi (10 Ronde, 2 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	6,3666	93,06	1,6277	96,36	872,49
	2	2,8704	95,49	0,5488	99,39	
2	1	2,8973	95,14	1,0053	98,18	865,73
	2	2,6282	94,44	0,5458	99,39	
3	1	2,2143	96,53	0,4844	98,18	868,14
	2	1,6319	97,22	0,2857	100,00	

***Client* 2 Raspberry Pi (10 Ronde, 3 *Epoch*)**						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	8,3000	89,93	2,2198	96,47	1644,58
	2	3,2851	93,23	1,8333	95,29	
	3	2,8536	93,58	1,6667	98,82	
2	1	2,4538	96,35	1,6180	95,29	1677,79
	2	2,4143	94,97	1,2303	96,47	
	3	2,3340	94,79	1,2312	97,65	
3	1	1,6448	97,40	1,0072	98,24	1560,22
	2	1,8731	96,18	1,0658	98,82	
	3	1,9802	95,49	0,9812	99,41	

Client 2 Raspberry Pi (10 Ronde, 4 *Epoch*)						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	8,9520	92,36	3,0732	94,83	2538,92
	2	3,4664	91,67	2,3040	94,83	
	3	3,0922	93,06	2,1725	94,25	
	4	2,6011	95,66	1,7548	97,70	
2	1	3,2106	92,71	2,0961	96,55	2004,39
	2	2,7436	94,97	1,8773	95,98	
	3	1,7894	96,53	1,5279	97,13	
	4	2,3775	94,97	1,6666	95,40	
3	1	2,0327	96,35	1,7031	97,13	1848,42
	2	1,9146	96,35	1,5111	98,28	
	3	1,5486	96,53	1,5046	98,28	
	4	1,6135	96,88	1,5104	97,70	

Client 2 Raspberry Pi (10 Ronde, 5 *Epoch*)						
Ronde	***Epoch***	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	1	12,3014	87,68	8,0561	84,35	3883,07
	2	5,8396	85,85	2,3510	96,60	
	3	4,0206	91,54	1,1741	96,60	
	4	3,8206	92,46	1,7479	94,56	
	5	4,4294	91,36	1,4141	96,60	
2	1	4,2872	92,28	2,5431	95,24	3902,46
	2	2,9134	94,12	1,1235	97,28	
	3	2,4710	94,67	1,0333	97,96	
	4	2,7406	93,57	0,8632	95,24	
	5	2,6167	94,12	1,0465	96,60	
3	1	2,6474	94,30	1,1130	97,28	3889,54
	2	2,3745	95,77	1,2087	96,60	
	3	2,1526	96,69	0,8740	97,96	
	4	2,0206	95,40	0,8280	98,64	
	5	1,9933	94,67	0,8851	97,28	

Log selangkapnya: https://its.id/m/HasilPelatihanFLClient2

Lampiran 4 Hasil Pelatihan *Centralized Learning*

Server (10 *Epoch*)					
Epoch	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	9,5215	81,60	7,4162	89,79	40,90
2	4,3464	91,81	3,1681	97,18	41,29
3	3,6619	93,93	1,6573	98,24	41,49

Server (20 *Epoch*)					
Epoch	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	9,7325	82,92	7,4789	92,25	40,96
2	4,0510	92,25	3,0411	98,59	40,69
3	4,1462	92,78	1,5765	98,59	40,80

Server (30 *Epoch*)					
Epoch	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	9,5751	81,69	7,9887	91,90	41,77
2	4,6603	91,20	2,9790	98,24	41,52
3	3,9810	92,87	1,6381	97,89	41,54

Server (40 *Epoch*)					
Epoch	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	9,5836	80,28	7,8844	90,49	41,53
2	4,5145	91,73	3,7806	96,13	41,54
3	4,0111	92,25	1,7204	98,24	41,66

Server (50 *Epoch*)					
Epoch	***Loss***	***Accuracy (%)***	***Validation Loss***	***Validation Accuracy (%)***	**Durasi (detik)**
1	9,8319	81,43	7,3217	91,55	41,69
2	4,6132	90,76	3,0098	97,18	41,49
3	4,2194	92,17	1,7078	98,94	41,19

Log selangkapnya: https://its.id/m/HasilPelatihanCL

Lampiran 5 Hasil Uji Cukup Cahaya Federated Learning dan Centralized Learning

***Centralized Learning Client* 1 Cukup Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
14:51:44	5027221021	*KNOWN*	0,8101	778
14:51:42	5027221021	*KNOWN*	0,8037	692
14:51:41	5027221021	*KNOWN*	0,8053	685
14:51:39	5027221021	*KNOWN*	0,8053	679
14:51:37	5027221021	*KNOWN*	0,8403	712

***Centralized Learning Client* 2 Cukup Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
15:10:53	5027221021	*KNOWN*	0,8148	2999
15:10:49	5027221021	*KNOWN*	0,8828	3080
15:10:44	5027221021	*KNOWN*	0,8788	2951
15:10:40	5027221021	*KNOWN*	0,8648	2881
15:10:35	5027221021	*KNOWN*	0,8434	3090

***Federated Learning* A Cukup Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
15:54:44	5027221021	*KNOWN*	0,7429	726
15:54:43	5027221021	*KNOWN*	0,9212	749
15:54:41	5027221021	*KNOWN*	0,9414	688
15:54:39	5027221021	*KNOWN*	0,9228	684
15:54:39	5027221021	*KNOWN*	0,9261	690

***Federated Learning* B Cukup Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
16:10:46	5027221021	*KNOWN*	0,8446	3151
16:10:41	5027221021	*KNOWN*	0,8266	2990
16:10:36	5027221021	*KNOWN*	0,7868	2942
16:10:31	5027221021	*KNOWN*	0,8905	3076
16:10:26	5027221021	*KNOWN*	0,9073	2997

Log selangkapnya: https://its.id/m/HasilUjiCukupCahayaFLdanCL

Lampiran 6 Hasil Uji Cukup Cahaya Federated Learning dan Centralized Learning

***Centralized Learning Client* 1 Minim Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
15:29:29	5027221021	*KNOWN*	0,7932	678
15:29:27	5027221021	*KNOWN*	0,7969	710
15:29:26	5027221021	*KNOWN*	0,7977	724
15:29:24	5027221021	*KNOWN*	0,8054	744
15:29:22	5027221021	*KNOWN*	0,8072	712

***Centralized Learning Client* 2 Minim Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
15:38:34	5027221021	*KNOWN*	0,8226	2411
15:38:30	5027221021	*KNOWN*	0,8382	3336
15:38:25	5027221021	*KNOWN*	0,7414	3180
15:38:20	5027221021	*KNOWN*	0,8158	3433
15:38:15	5027221021	*KNOWN*	0,7856	3321

***Federated Learning* A Minim Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
16:30:35	5027221021	*KNOWN*	0,7925	711
16:30:33	5027221021	*KNOWN*	0,8155	705
16:30:32	5027221021	*KNOWN*	0,7900	697
16:30:30	5027221021	*KNOWN*	0,7832	691
16:30:28	5027221021	*KNOWN*	0,7849	714

***Federated Learning* B Minim Cahaya**				
Waktu	**Identitas (NRP)**	**Status**	***Cosine Similarity***	***Latency* (ms)**
16:20:30	5027221021	*KNOWN*	0,7727	3061
16:20:25	5027221021	*KNOWN*	0,7252	2975
16:20:20	5027221021	*UNKNOWN*	0,6911	3211
16:20:13	5027221021	*KNOWN*	0,8544	2997
16:20:08	5027221021	*KNOWN*	0,8844	2950

Log selengkapnya: https://its.id/m/HasilUjiMinimCahayaFLdanCL

Lampiran 7 Hasil Uji Video *Federated Learning* dan *Centralized Learning*

***Centralized Learning Client* 1**				
Waktu	***Frame***	**Identitas (NRP)**	**Status**	***Cosine Similarity***
0:00	*5*	5027241094	*UNKNOWN*	0,4792
0:00	*10*	5027241094	*UNKNOWN*	0,4825
0:00	*15*	5027241094	*UNKNOWN*	0,4949
0:00	*20*	5027231086	*UNKNOWN*	0,4962
0:00	*25*	5027231086	*UNKNOWN*	0,4851

***Centralized Learning Client* 2**				
Waktu	***Frame***	**Identitas (NRP)**	**Status**	***Cosine Similarity***
0:00	*5*	5027241094	*UNKNOWN*	0,4792
0:00	*10*	5027241094	*UNKNOWN*	0,4825
0:00	*15*	5027241094	*UNKNOWN*	0,4949
0:00	*20*	5027231086	*UNKNOWN*	0,4962
0:00	*25*	5027231086	*UNKNOWN*	0,4851

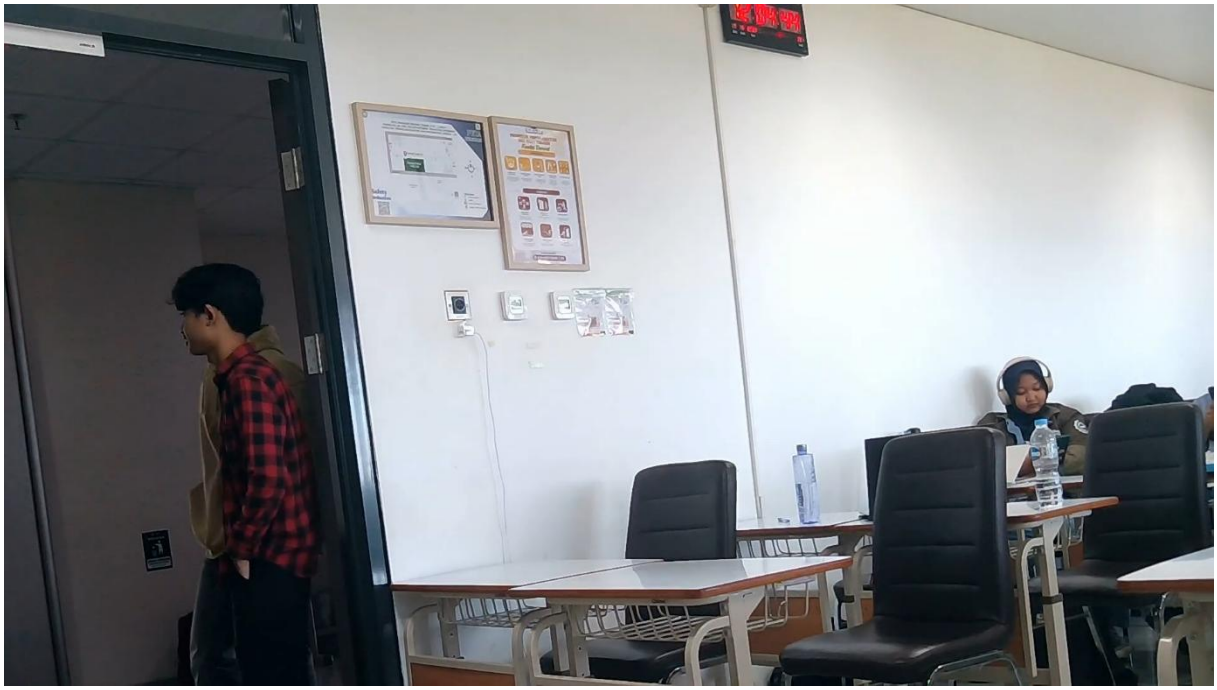
***Federated Learning* A**				
Waktu	***Frame***	**Identitas (NRP)**	**Status**	***Cosine Similarity***
0:00	*5*	5027231086	*UNKNOWN*	0,6167
0:00	*10*	5027241094	*UNKNOWN*	0,6409
0:00	*15*	5027241094	*UNKNOWN*	0,6414
0:00	*20*	5027241094	*UNKNOWN*	0,6478
0:00	*25*	5027241094	*UNKNOWN*	0,6525

***Federated Learning* B**				
Waktu	***Frame***	**Identitas (NRP)**	**Status**	***Cosine Similarity***
0:00	*5*	5027241094	*UNKNOWN*	0,6482
0:00	*10*	5027241094	*UNKNOWN*	0,6844
0:00	*15*	5027241094	*UNKNOWN*	0,6890
0:00	*20*	5027241094	*UNKNOWN*	0,6981
0:00	*25*	5027241094	*KNOWN*	0,7031

Log Selengkapnya: https://its.id/m/HasilUjiVideoFLdanCL

Lampiran 8 Video Simulasi Uji *Federated Learning* dan *Centralized Learning*

Cuplikan Uji Video Simulasi:



Video Selengkapnya: https://its.id/m/VideoUjiFLdanCL

BIOGRAFI PENULIS

Steven Figo lahir di Sibolga, 30 November 2004, merupakan anak pertama dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Tri Ratna Sibolga, SD Tri Ratna Sibolga, SMP Tri Ratna Sibolga, dan SMAS Tri Ratna Sibolga. Setelah lulus dari SMAS Tri Ratna Sibolga pada tahun 2022 penulis mengikuti SBMPTN dan diterima di Departemen Teknologi Informasi FTEIC - ITS pada tahun 2022 dan terdaftar dengan NRP 5027221021.

Di lingkungan Departemen Teknologi Informasi, penulis merupakan pengurus Himpunan Mahasiswa Teknologi Informasi (HMIT) dan pernah berkontribusi sebagai panitia dalam ajang tahunan ARA 4.0, 5.0, dan 6.0. Pada bidang kompetisi, penulis pernah menjadi finalis Gemastik 2024 pada kategori Smart City. Selain itu, penulis juga aktif sebagai asisten mata kuliah untuk subjek Sistem Operasi, Komunikasi Data dan Jaringan Komputer, *Internet of Things* (IoT), Kecerdasan Artifisial dan *Machine Learning*, Struktur Data & Pemrograman Berorientasi Objek, Strategi Optimasi Komputasi Awan, serta Smart City.

Di luar aktivitas akademik, penulis juga aktif mengembangkan diri melalui UKM Paduan Suara Mahasiswa ITS. Sebagai anggota, penulis telah mengikuti berbagai kompetisi tingkat nasional maupun internasional, termasuk meraih medali emas pada ajang Karangturi International Choir Competition, Fespa UBAYA, dan Malaysian Choral Eisteddfod.