DETEKSI DAN KLASIFIKASI SERANGAN JARINGAN *SMART CITY* MENGGUNAKAN *FEDERATED LEARNING*

Disusun Oleh:

N a m a : Rian Nur Ikhsan
NIM : 22523297

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

DETEKSI DAN KLASIFIKASI SERANGAN JARINGAN *SMART CITY* MENGGUNAKAN *FEDERATED LEARNING*

TUGAS AKHIR

Disusun Oleh:

N a m a : Rian Nur Ikhsan

NIM : 22523297

Yogyakarta, 12 Januari 2026

Pembimbing I,	Pembimbing II,
(Fayruz Rahma, S.T., M.Eng.)	(Kurniawan Dwi Irianto, S.T., M.Sc.)

HALAMAN PENGESAHAN DOSEN PENGUJI

DETEKSI DAN KLASIFIKASI SERANGAN JARINGAN *SMART CITY* MENGGUNAKAN *FEDERATED LEARNING*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 12 Januari 2026

Tim Penguji

Kurniawan Dwi Irianto, S.T., M.Sc. _____

Anggota 1

Moh. Idris, S.Kom., M.Kom. _____

Anggota 2

Dr. Mukhammad A Setiawan, S.T., M.Sc. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Rian Nur Ikhsan

NIM : 22523297

Tugas akhir dengan judul:

DETEKSI DAN KLASIFIKASI SERANGAN JARINGAN *SMART CITY* MENGGUNAKAN *FEDERATED LEARNING*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 12 Januari 2026

(Rian Nur Ikhsan)

HALAMAN PERSEMBAHAN

Alhamdulillahirobbil'alamin, puji syukur atas berkah dan nikmat dari Allah Swt. Tuhan dari semesta ala mini yang akbar. Tidak lupa Shalawat juga salam kita haturkan kepada junjungan kita Nabi Muhammad saw. Yang akan memberikan safa'atnya di hari akhir kelak.

Terima kasih telah menjadi penyambung saya ke dunia ini kepada orang tua saya, mengasuh, membimbing, menemani saya sejak di alam rahim. Mendukung saya untuk selalu semangat walaupun saya sering berkecil hati. Semoga amal-amal mereka diterima di sisi-Nya sepanjang masa.

Terima kasih kepada Dosen Pembimbing Tugas akhir saya, Ibu Fayruz Rahma dan Bapak Kurniawan Dwi Irianto yang selalu memberi nasihat dan membimbing saya dengan sabar selama ini.

Terima kasih kepada semua pihak yang tidak dapat saya sebutkan satu per satu atas bantuan, dukungan dan semangat baik secara langsung maupun tidak.

HALAMAN MOTO

“Maaf…”

“Terima kasih”

“Allah mengetahui, sedangkan kamu tidak mengetahui.“ (Q.S Al-Baqarah: 216)

“Nilai kehidupanku lemah, tapi diriku tetap menunggu nilainya terkena inflasi, cihuuy”

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh.

Segala puji dan syukur penulis panjatkan ke hadirat Allah Swt. atas limpahan rahmat, taufik, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik. Laporan ini berjudul "Deteksi dan Klasifikasi Serangan Jaringan *Smart City* dengan Federated Learning", yang disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana (S1) pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah memberikan dukungan, baik secara langsung maupun tidak langsung, selama proses penyusunan Tugas Akhir ini, yaitu:

1. Kedua orang tua penulis yang telah menuntun dari kecil hingga besar dengan nilai yang tidak bisa dihitungkan secara materi dan batin.
2. Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
3. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika – Program Sarjana UII.
4. Ibu Fayruz Rahma, S.T., M.Eng., yang selalu memberi nasihat dan membimbing dengan sabar selama ini.
5. Bapak Kurniawan Dwi Irianto, S.T., M.Sc., yang memberikan petunjuk serta arahan selama ini.
6. Paman-Paman yang memberikan ilmu apa itu sabar dan percaya diri.
7. Semua pihak yang tidak dapat disebutkan satu per satu, namun telah berkontribusi dalam bentuk apapun

Penulis secara sadah bahwa Tugas Akhir ini masih jauh dari kata cukup. Oleh karena itu, kritik, masukan, dan saran yang terbuka untuk membangun sangat diharapkan. Semoga karya ini kelak dapat memberikan manfaat serta dasar pengembangan penelitian yang lebih lanjut di masa yang akan datang.

Yogyakarta, 12 Januari 2026

(Rian Nur Ikhsan)

SARI

Perkembangan teknologi digital mendorong implementasi konsep *smart city* yang mengandalkan jaringan IoT untuk meningkatkan efisiensi layanan publik. Namun, kompleksitas jaringan ini meningkatkan risiko serangan siber yang dapat mengganggu layanan kritis. Penelitian ini mengkaji penerapan *Federated Learning* (FL) sebagai pendekatan terdistribusi untuk membangun Network Intrusion Detection System (NIDS) tanpa memusatkan data mentah, sehingga privasi tetap terjaga. Dataset yang digunakan adalah ToN-IoT, dengan simulasi distribusi data IID (Independent and Identically Distributed) dan Non-IID menggunakan Dirichlet untuk merepresentasikan heterogenitas antar *client*. Dua algoritma FL, yaitu Federated Averaging (FedAvg) dan Federated Proximal (FedProx), dibandingkan pada dua arsitektur deep learning: *Multi-Layer Perceptron* (MLP) dan *Convolutional Neural Network 1D* (CNN-1D), untuk skenario klasifikasi biner dan multi-kelas.

Hasil eksperimen menunjukkan bahwa FedProx lebih stabil pada kondisi Non-IID dibanding FedAvg, meskipun dengan biaya komputasi lebih tinggi. CNN-1D memberikan performa lebih baik daripada MLP, terutama pada skenario multi-kelas, dengan peningkatan akurasi dan F1-Score yang signifikan. Penelitian ini merekomendasikan penggunaan FedProx dengan CNN-1D untuk lingkungan *smart city* yang heterogen, karena mampu menjaga stabilitas pelatihan dan akurasi deteksi serangan.

Kata kunci: *smart city, federated learning*, ton-iot, nids, non-iid.

GLOSARIUM

Backdoor	Serangan yang memungkinkan akses tidak sah ke sistem melalui celah tersembunyi.
Batch Size	Jumlah sampel data yang diproses dalam satu iterasi pelatihan model.
Confusion Matrix	Matriks yang digunakan untuk mengevaluasi performa model klasifikasi dengan membandingkan prediksi dan label aktual.
Dirichlet	Distribusi probabilitas yang digunakan untuk mensimulasikan ketimpangan data antar *client* dalam skenario Non-IID.
Federated Learning	Paradigma pembelajaran mesin terdistribusi yang memungkinkan pelatihan model tanpa memindahkan data mentah ke server pusat.
Man in The Middle	Jenis serangan di mana penyerang menyusup di antara komunikasi dua pihak untuk mengintip atau memanipulasi data.
Micro (μ)	Simbol yang digunakan untuk merepresentasikan koefisien *proximal term* pada algoritma FedProx, yang mengontrol kekuatan regularisasi terhadap deviasi model lokal dari model global.
Non-IID	Kondisi distribusi data antar *client* yang tidak identik dan independen, umum terjadi pada lingkungan terdistribusi.
Proximal Term	Komponen tambahan pada fungsi loss FedProx yang membatasi penyimpangan model lokal dari model global untuk mengurangi *client*t yang mengalami *drift*.
Skew	Ketidakseimbangan distribusi data atau kelas yang menyebabkan bias dalam pelatihan model.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI viii
GLOSARIUM ix
DAFTAR ISI x
DAFTAR TABEL xiii
DAFTAR GAMBAR xv
BAB I PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 4
1.6 Metodologi Penelitian 4
1.7 Sistematika Penulisan 5
BAB II LANDASAN TEORI 7
2.1 *Network Intrusion Detection System* (NIDS) 7
2.2 CRISP-DM 8
2.3 Dataset ToN-IoT 9
2.4 Teknik Pre-Processing 14
2.5 *Federated Learning* 15
2.5.1 Arsitektur *Horizontal Federated Learning* 15
2.5.2 Karakteristik Distribusi Data (IID dan Non-IID) 16
2.5.3 Dirichlet 17
2.5.4 Algoritma Agregasi 17
2.6 Deep Learning 19
2.6.1 Multi-Layer Perceptron 20
2.6.2 Convolutional Neural Network 1 Dimensional (CNN-1D) 20
2.7 *Hyperparameter*: *Grid Search* 21
2.7.1 Mekanisme *Grid Search* 21
2.7.2 Parameter Kunci dalam Pembelajaran Terdistribusi 21
2.7.3 Strategi Pemilihan *Hyperparameter* Berbasis Multikriteria 22
2.8 Teknik Evaluasi 22
2.8.1 Confusion Matrix 22
2.8.2 Metrik Evaluasi Kinerja Model Klasifikasi 23
2.8.3 Evaluasi Kinerja Model *Federated Learning* 24
2.9 Penelitian Terkait 25
BAB III METODOLOGI PENELITIAN 30
3.1 Alur Pengerjaan 30
3.2 Lingkungan Penelitian 31
3.3 *Businesses Understanding* 33
3.3.1 Tujuan Bisnis (*Business Objectives*) 34

3.3.2 Kriteria Keberhasilan (*Success Criteria*) 34
3.4 Data Understanding 35
3.4.1 Analisis Distribusi Kelas 35
3.4.2 Analisis Integritas dan Kualitas Data 36
3.4.3 Analisis Korelasi Fitur 36
3.5 Data Preparation 37
3.5.1 Pembersihan Data 37
3.5.2 Transformasi Variabel Kategorikal (*Encoding*) 38
3.5.3 Seleksi Fitur Berbasis Variansi 38
3.5.4 Standardisasi Fitur (Feature Scaling) 38
3.5.5 Pembagian Data Terstratifikasi 39
3.5.6 Pembersihan Data 39
3.6 Modeling 39
3.6.1 Strategi *Horizontal Federated Learning* 40
3.6.2 Arsitektur *Deep Learning* 40
3.6.3 Strategi Distribusi Data Antar *Client* (Non-IID Simulation) 42
3.6.4 Optimasi *Hyperparameter* dan Reprodusibilitas 44
3.7 *Evaluation* 46
3.7.1 Evaluasi Kinerja Klasifikasi 47
3.7.2 Evaluasi pada Lingkungan Federated Learning 47
3.7.3 Keterkaitan Evaluasi dengan Tujuan Penelitian 48
BAB IV HASIL dan Pembahasan 49
4.1 Hasil Data Understanding 49
4.1.1 Distribusi Kelas dan Implikasi Non-IID 49
4.1.2 Kualitas dan Integritas Data 52
4.1.3 Karakteristik Statistik dan Relevansi Fitur 56
4.2 Hasil Data Preparation 59
4.2.1 Implementasi Pembersihan Data 59
4.2.2 Transformasi dan Seleksi Fitur 60
4.2.3 Hasil Pembagian Data (Data Splitting) 60
4.2.4 Implementasi *Pipeline* (*Core Code*) 61
4.2.5 Output Artifacts 64
4.3 Modeling dan Implementasi Sistem 66
4.3.1 Konfigurasi Lingkungan Eksperimen 66
4.3.2 Strategi Horizontal *Federated Learning* 67
4.3.3 Implementasi Arsitektur Deep Learning 67
4.3.4 Simulasi Distribusi Data Non-IID 69
4.3.5 Implementasi *Federated Learning* dengan Flower 73
4.3.6 Simulasi Distribusi Data Non-IID 74
4.4 Analisis *Federated Learning* dari Eksperimen 79
4.4.1 Gambaran Umum Hasil Eksperimen (*Global Metrics*) 80
4.4.2 Analisis Performa Arsitektur Model: MLP vs CNN Skenario Standar 82
4.4.3 Dampak Heterogenitas Data: IID vs Non-IID 83
4.4.4 Analisis Efektivitas Algoritma: FedAvg vs FedProx 84
4.4.5 Analisis Sensitivitas Parameter Proximal $\boldsymbol{\mu}$ 88
4.4.6 Analisis Pola Misklasifikasi dan Dampak Keamanan 90
4.4.7 Analisis Pola Misklasifikasi dan Dampak Keamanan 92
4.5 Dokumentasi Hasil Penelitian 92
BAB V KESIMPULAN 93
5.1 Kesimpulan 93

DAFTAR PUSTAKA.....97
LAMPIRAN 102

DAFTAR TABEL

Tabel 2.1 *Connection Features* 10
Tabel 2.2 *Statistical Features* 10
Tabel 2.3 DNS *Features* 10
Tabel 2.4 SSL *Features* 11
Tabel 2.5 HTTP *Features* 11
Tabel 2.6 *Violation Features* 12
Tabel 2.7 *Labelling Attributes* 12
Tabel 2.8 Distribusi Data dan Karakteristik Kelas 13
Tabel 2.9 Perbedaan FedAvg dengan FedProx 18
Tabel 2.10 Penelitian Terkait 26
Tabel 3.1 *Local Host Hardware Enviroment* 32
Tabel 3.2 Perangkat Lunak dan Pustaka Pendukung 32
Tabel 4.1 Kolom dengan *Missing Values* (Top 20) 55
Tabel 4.2 Statistik Deskriptif Fitur Utama per Tipe Serangan 58
Tabel 4.3 Strategi Implementasi Penanganan *Missing Values* 59
Tabel 4.4 Pemetaan Label *Encoding* (*Multiclass*) 60
Tabel 4.5 Dimensi Data Hasil *Preprocessing* 60
Tabel 4.6 Konfigurasi Lingkungan Tetap 66
Tabel 4.7 Distribusi Data *Client Binary Classification* (α=0.3 - Non-IID) 70
Tabel 4.8 Distribusi Data *Client Binary Classification* (α=5 - IID) 70
Tabel 4.9 Distribusi Data *Client Multiclass Classification* (α=0.3 - Non-IID) 71
Tabel 4.10 Distribusi Data *Client Multiclass Classification* (α=5 - IID) 71
Tabel 4.11 Hasil *Grid Search* (α=0.3 – Non-IID) 74
Tabel 4.12 Hasil *Grid Search* (α=5 – IID) 76
Tabel 4.13 Perbandingan Performa Algoritma (Global Metrics) pada Skenario IID 80
Tabel 4.14 Detail Per Kelas (F1-Score) untuk CNN *Multiclass* pada Skenario IID 81
Tabel 4.15 Lama Waktu Pelatihan (Training Cost) pada Skenario IID 81
Tabel 4.16 Perbandingan Performa Algoritma (Global Metrics) pada Skenario Non-IID (Dirichlet α=0.3) 83
Tabel 4.17 Detail Per Kelas (F1-Score) untuk CNN *Multiclass* pada Skenario Non-IID (Dirichlet α=0.3) 85
Tabel 4.18 Lama Waktu Pelatihan (*Training Cost*) pada Skenario Non-IID 87

Tabel 4.19 Perilaku FedProx terhadap Variasi Parameter Proximal (μ) pada Skenario IID dan Non-IID 88

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi *Horizontal Federated Learning* 15
Gambar 2.2 Ilustrasi *Confusion Matrix* 23
Gambar 3.1 Alur kerja penelitian..... 31
Gambar 4.1 *Plotting* Distribusi kategori serangan pada dataset ToN-IoT Network..... 51
Gambar 4.2 Distribusi kategori serangan pada dataset ToN-IoT Network 52
Gambar 4.3 Kode Alalisis *Missing Values* pada Fitur Spesifik Protokol 54
Gambar 4.4 Persentase *Missing Values* pada Fitur Spesifik Protokol 56
Gambar 4.5 Distribusi Fitur Numerik Utama 57
Gambar 4.6 *Scatter Plots* Persebaran Data per *Attack Type* 57
Gambar 4.7 Heatmap Korelasi Pearson Antar Fitur Numerik 58
Gambar 4.8 *Pipeline Core Code Cleaning Data* 62
Gambar 4.9 *Pipeline* Klasifikasi *Binary Class* 63
Gambar 4.10 *Pipeline* Klasifikasi *Multiclass* 64
Gambar 4.11 *Pipeline Eksekusi dan Serialisasi Artefak Preprocessing*..... 65
Gambar 4.12 *Seed Value*..... 67
Gambar 4.13 *Pipeline* MLP 68
Gambar 4.14 *Pipeline* CCN-1D..... 69
Gambar 4.15 *Pipeline Dirichlet*..... 70
Gambar 4.16 *Pipeline client*t 73
Gambar 4.17 Contoh inisiasi server *Federated Learning*..... 74
Gambar 4.18 Pemilihan *Hyperparameter* Berdasarkan Balance Score..... 78
Gambar 4.19 Confusion Matrix FedAvg pada Skenario IID (MLP Binary, CNN Binary, MLP *Multiclass*, CNN *Multiclass*) 82
Gambar 4.20 Confusion Matrix FedAvg pada Skenario Non-IID (MLP Binary, CNN Binary, MLP *Multiclass*, CNN *Multiclass*) 84
Gambar 4.21 Confusion Matrix FedProx (μ=0.01) pada Skenario Non-IID 86
Gambar 4.22 Confusion Matrix FedProx (μ=0.001) pada Skenario Non-IID 87
Gambar 4.23 *Confusion Matrix* FedProx (μ=0.01) pada Skenario IID 89
Gambar 4.24 *Confusion Matrix* FedProx (μ=0.001) pada Skenario IID 90

BAB I
PENDAHULUAN

1.1 Latar belakang

Seiring perkembangan teknologi digital, internet telah menjadi kebutuhan fundamental dalam kehidupan modern. Pada tahun 2025, lebih dari 6 miliar orang di seluruh dunia memiliki akses internet, dengan lebih dari 70% populasi global menggunakan perangkat yang terhubung ke jaringan digital (Statista, 2025). Pertumbuhan konektivitas ini mendorong berbagai kota untuk mengadopsi konsep *smart city*, yang memanfaatkan teknologi *Internet of Things* (IoT) dan interkoneksi jaringan untuk meningkatkan efisiensi layanan publik. Lebih dari 2,000 kota di dunia telah mengimplementasikan inisiatif *smart city*, mulai dari sistem transportasi, pengelolaan energi, hingga layanan kesehatan dan keamanan publik (Barker & Brooks, 2023). Namun berkembangnya infrastruktur jaringan digital juga meningkatkan permukaan serangan (*attack surface*) yang rentan terhadap ancaman siber.

Smart city menuntut pengelolaan jaringan yang kompleks dan terdistribusi. *smart city* umumnya memiliki puluhan hingga ratusan *gateway* dan router yang menghubungkan ribuan perangkat IoT, sensor, dan sistem kontrol. Setiap *gateway* berfungsi sebagai titik agregasi yang mengelola lalu lintas data dari perangkat di sekitarnya sebelum diteruskan ke sistem pusat atau *cloud*. *Gateway* tidak hanya sebagai jalur komunikasi, tetapi juga sebagai titik kritis yang memantau dan memfilter trafik untuk menjaga integritas dan keamanan data. Ketergantungan pada jaringan ini membuat pengelolaan *gateway* menjadi penting, karena gangguan atau serangan pada salah satu titik akses dapat berdampak pada layanan publik, mulai dari transportasi hingga sistem darurat.

Seiring meningkatnya konektivitas, ancaman keamanan siber terhadap *smart city* juga mengalami eskalasi. Kerugian karena serangan siber dapat proyeksikan secara global mencapai $10,5 triliun pada 2025, menggaris bawahi pentingnya investasi dalam infrastruktur keamanan siber yang kokoh (Morgan, 2025). Intrusion Detection System (IDS) menjadi mekanisme utama untuk mendeteksi aktivitas mencurigakan pada lalu lintas jaringan dengan memantau pola trafik dan mengidentifikasi anomali (Ahmad dkk., 2021). Volume data yang besar membuat pendekatan berbasis *machine learning* semakin relevan, karena model dapat mempelajari pola serangan dari data historis dan mendeteksi anomali secara otomatis (Prazeres dkk., 2023).

Pendekatan pembelajaran terdistribusi menawarkan solusi untuk mengatasi keterbatasan NIDS terpusat dimana data mentah dikirim ke server pusat. *Federated Learning* (FL) memungkinkan pelatihan model dilakukan secara lokal di masing-masing *gateway* tanpa memindahkan data mentah ke server pusat (McMahan dkk., 2017). Setiap *gateway* melatih model IDS secara lokal, kemudian hanya mengirim pembaruan model ke server pusat untuk digabung menjadi model global. Pendekatan ini sesuai untuk lingkungan dengan banyak *gateway*, karena mengurangi beban komunikasi dan memungkinkan pelatihan paralel tanpa mengorbankan data lokal (Thantharate & T, 2023).

Namun, penerapan *Federated Learning* pada *smart city* menghadapi tantangan. Data yang dikumpulkan tiap *gateway* berbeda berdasarkan lokasi, waktu, dan jenis layanan. Sebagai contoh, *gateway* untuk sistem transportasi publik memiliki pola trafik berbeda dengan *gateway* di fasilitas kesehatan atau area residensial. Kondisi ini menghasilkan distribusi data yang tidak seragam, atau *Non-Independent and Identically Distributed* (Non-IID) (Al-Huthaifi dkk., 2023). Non-IID dapat menyebabkan ketidakstabilan dalam pelatihan model karena model lokal pada satu *gateway* mungkin berbeda signifikan dari *gateway* lainnya, sehingga performa model global bisa menurun (Li dkk., 2020). Tantangan ini menjadi fokus penelitian agar IDS berbasis FL tetap efektif dan andal dalam mendeteksi serangan pada *smart city* yang kompleks (Li dkk., 2020).

Berdasarkan permasalahan heterogenitas data dan kebutuhan privasi di atas, penelitian ini mengajukan strategi penyelesaian teknis berupa kerangka kerja deteksi intrusi yang komprehensif menggunakan dataset ToN-IoT sebagai representasi trafik jaringan modern . Penelitian ini secara spesifik mengevaluasi dua algoritma agregasi, yaitu *Federated Averaging* (FedAvg) dan *Federated Proximal* (FedProx), di mana FedProx diuji kemampuannya dalam menangani data *Non-IID* melalui penambahan *proximal term* (McMahan dkk., 2017; Li dkk., 2020). Evaluasi dilakukan pada dua arsitektur model, yakni MLP yang efisien dan CNN-1D yang unggul dalam ekstraksi fitur, dengan skenario *binary classification* untuk deteksi cepat dan *multiclass classification* untuk identifikasi jenis serangan (Devi dkk., 2025). Strategi ini diimplementasikan secara sistematis melalui alur kerja yang dimulai dari identifikasi masalah, dilanjutkan dengan siklus teknis yang meliputi persiapan data menggunakan simulasi distribusi *dirichlet*, pemodelan melalui pelatihan lokal, serta evaluasi model global yang hasilnya dianalisis untuk menjawab tujuan penelitian.

1.2 Rumusan Masalah

Penelitian ini membahas beberapa permasalahan utama:

a. Bagaimana penerapan *Federated Learning* serta implikasi performanya untuk deteksi dan klasifikasi serangan pada lalu lintas jaringan *smart city*.
b. Bagaimana pengaruh perbedaan distribusi data antar *client* terhadap performa sistem IDS berbasis FL.
c. Bagaimana perbandingan performa beberapa pendekatan FL dan arsitektur model pada skenario tersebut.

1.3 Batasan Masalah

Penelitian ini berfokus pada penerapan Federated Learning untuk deteksi dan klasifikasi serangan pada lalu lintas jaringan *smart city*:

a. Dataset yang digunakan adalah ToN-IoT subset *network traffic*, yang merepresentasikan lalu lintas jaringan dengan beberapa jenis serangan pada lingkungan IoT dan jaringan konvensional
b. Lingkungan eksperimen berupa simulasi dengan beberapa *client* yang merepresentasikan *gateway* atau node jaringan terdistribusi.
c. Algoritma *Federated Learning* yang dibandingkan terbatas pada Federated Averaging (FedAvg) dan Federated Proximal (FedProx).
d. Arsitektur model yang dievaluasi terbatas pada Multi-Layer Perceptron (MLP) dan Convolutional Neural Network (CNN) 1D untuk dua skenario klasifikasi, yaitu *binary* dan *multiclass*.
e. Fokus penelitian adalah simulasi dan analisis performa model, tidak mencakup *deployment* pada infrastruktur fisik *smart city* atau optimisasi perangkat *embedded*.

1.4 Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan:

a. Mengkaji penerapan *Federated Learning* untuk sistem deteksi dan klasifikasi serangan pada lalu lintas jaringan *smart city* menggunakan dataset ToN-IoT
b. Menganalisis pengaruh heterogenitas data antar *client* terhadap performa sistem NIDS berbasis *Federated Learning*.
c. Membadingkan performa algoritma *Federated Averaging* (FedAvg) dan *Federated Proximal* (FedProx) dalam konteks deteksi intrusi.

d. Mengevaluasi performa dua arsitektur model, yaitu *MLP* dan CNN 1D pada skenario *binary* dan *multiclass classification*.
e. Memberikan rekomendasi praktis terkait pemilihan algoritma dan arsitektur model untuk implementasi NIDS berbasis *Federated Learning* pada lingkungan jaringan heterogen, disesuaikan kembali dengan konteks lapangan.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat bagi beberapa pihak, baik secara akademis maupun praktis:

a. Manfaat Akademis:

   Penelitian ini diharapkan dapat menambah referensi ilmiah terkait penerapan *Federated Learning* pada sistem deteksi intrusi jaringan, khususnya pada kondisi data heterogen (Non-IID).

b. Manfaat Praktis:

   Penelitian ini diharapkan dapat menjadi acuan awal bagi praktisi keamanan jaringan dalam memilih algoritma dan arsitektur model pada implementasi FL-based IDS tanpa memusatkan data sensitif.

1.6 Metodologi Penelitian

Penelitian ini menggunakan pendekatan *CRISP-DM* (*Cross-Industry Standard Process for Data Mining*) untuk memastikan proses penelitian sistematis dan terstruktur. Metodologi yang diterapkan mencakup lima tahap sebagai berikut:

a. *Business Understanding*

   Menelaah kebutuhan *smart city* terkait keamanan jaringan dan memahami karakteristik dataset lalu lintas IoT yang digunakan untuk *Network Intrusion Detection System* (NIDS). Tujuannya adalah untuk menetapkan fokus penelitian dan sasaran performa sistem yang akan dibangun.

b. *Data Understanding*

   Melakukan eksplorasi dan verifikasi terhadap data yang dikumpulkan, termasuk memahami distribusi trafik jaringan, jenis serangan, dan karakteristik heterogen antar *gateway* atau node. Tahap ini bertujuan memastikan kualitas dan relevansi data untuk analisis lebih lanjut.

c. *Data Preparation*

Memproses data mentah agar siap digunakan dalam pembelajaran mesin, termasuk tahap *filtering*, normalisasi, dan pembagian data ke subset yang merepresentasikan masing-masing *client*/*gateway*. Persiapan data ini penting untuk memastikan setiap node memiliki data yang sesuai untuk pelatihan lokal dalam skema *Federated Learning*.

d. Modeling

Mengonfigurasi skenario *Federated Learning horizontal*, di mana setiap *client* melatih model secara lokal menggunakan data masing-masing yang distribusikan dengan metode *dirichlet*. Model lokal kemudian digabungkan menjadi model global melalui algoritma *Federated Averaging* (FedAvg) *Federated Proximal* (FedProx). Tahap ini fokus pada pengembangan dan penerapan model machine learning sederhana untuk deteksi intrusi.

e. Evaluation

Menilai performa model global yang dihasilkan menggunakan metrik numerik seperti akurasi dan *F1-score*. Evaluasi juga mempertimbangkan aspek efisiensi, termasuk jumlah iterasi komunikasi dan waktu pelatihan. Tahap ini bertujuan untuk memastikan model dapat mendeteksi serangan dengan efektif di lingkungan *smart city* yang heterogen.

1.7 Sistematika Penulisan

Untuk memberikan gambaran yang singkat tentang penelitian ini, penulisan disusun dalam lima bab yang saling berhubungan. Penyusunan penelitian ini mengikuti kaidah penulisan ilmiah dengan sistematika sebagai berikut:

1. BAB I PENDAHULUAN

   Bab ini berisi latar belakang yang menjelaskan konteks *smart city*, tantangan keamanan jaringan, keterbatasan pendekatan terpusat, dan pengenalan *Federated Learning* sebagai solusi alternatif. Selain itu, bab ini juga memuat rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

2. BAB II LANDASAN TEORI

   Bab ini berisi teori-teori yang menjadi dasar penelitian, mencakup *Network Intrusion Detection System* (NIDS), dataset ToN-IoT, konsep *Federated Learning*, algoritma *Federated Averaging* (FedAvg) dan *Federated Proximal* (FedProx), arsitektur model *deep learning* (MLP dan CNN 1D), distribusi Non-IID dengan *Dirichlet*, metrik evaluasi, serta tinjauan penelitian terkait.

3. BAB III METODOLOGI PENELITIAN

Bab ini berisi kerangka kerja penelitian yang mengadaptasi *CRISP-DM* untuk konteks *Federated Learning*, meliputi tahapan *Business Understanding*, *Data Understanding*, *Data Preparation*, partisi data Non-IID, pemilihan arsitektur model dan algoritma, *hyperparameter tuning* dengan *Grid Search*, rencana implementasi FedAvg dan FedProx, serta metode evaluasi yang digunakan.

4. BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi hasil eksplorasi data, *preprocessing*, partisi data Non-IID, hasil *hyperparameter tuning*, perbandingan performa algoritma *Federated Averaging* dan *Federated Proximal* pada kondisi data heterogen, analisis pengaruh parameter α dan μ, perbandingan arsitektur model *MLP* dan CNN, serta diskusi implikasi praktis dari temuan penelitian.

5. BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diperoleh dari hasil penelitian terkait efektivitas *Federated Learning* untuk deteksi intrusi pada lalu lintas jaringan *smart city*, serta saran-saran untuk penelitian lanjutan dan implementasi praktis di masa depan.

BAB II
LANDASAN TEORI

Bab ini menyusun landasan teori yang menjadi dasar bagi perancangan dan analisis sistem dalam penelitian ini. Pembahasan diawali dengan *Network Intrusion Detection System* (NIDS) sebagai komponen fundamental dalam deteksi serangan jaringan. Selanjutnya, kerangka kerja *CRISP-DM* (*Cross-Industry Standard Process for Data Mining*) digunakan untuk menstrukturkan alur penelitian secara metodologis. Aspek data dibahas melalui karakteristik dataset yang digunakan serta tahapan *preprocessing* yang diterapkan guna memastikan kesiapan data sebelum pemodelan. Fokus kemudian diarahkan pada tahapan *modeling*, khususnya arsitektur dan algoritma yang digunakan dalam skema *Federated Learning*. Lalu terdapat fokus untuk mecari *hyperparameter* yang optimal secara komputasi. Terakhir, bab ini ditutup dengan pembahasan penelitian terkait yang menjadi landasan bagi metodologi pada bab berikutnya.

2.1 *Network Intrusion Detection System* (NIDS)

Network Intrusion Detection System (NIDS) merupakan sistem perangkat lunak yang berfungsi untuk memantau lalu lintas jaringan secara kontinu dengan tujuan mendeteksi aktivitas yang mencurigakan atau pelanggaran kebijakan keamanan. Dalam arsitektur keamanan siber, NIDS berperan sebagai mekanisme pertahanan awal yang menganalisis aliran paket data guna mengidentifikasi indikasi serangan sebelum berdampak langsung pada sistem yang dilindungi (García-Teodoro dkk., 2009). Berdasarkan mekanisme pendeteksiannya, NIDS secara umum diklasifikasikan ke dalam dua pendekatan utama, yaitu *signature-based detection* dan *anomaly-based detection*.

Pendekatan *signature-based detection* bekerja dengan melakukan pencocokan pola (*pattern matching*) antara lalu lintas jaringan yang diamati dan basis data tanda tangan (*signature*) serangan yang telah diketahui sebelumnya. Apabila karakteristik paket data yang dianalisis sesuai dengan salah satu *signature* dalam basis data, sistem akan menghasilkan peringatan atau mengambil tindakan mitigasi yang telah ditentukan (García-Teodoro dkk., 2009). Pendekatan ini memiliki tingkat akurasi yang tinggi serta tingkat *false positive* yang relatif rendah untuk serangan yang telah terdokumentasi. Namun, keterbatasan utamanya terletak pada ketergantungan terhadap pembaruan basis data, sehingga metode ini tidak efektif

dalam mendeteksi serangan baru atau *zero-day attacks* yang belum memiliki pola yang terdefinisi.

Berbeda dengan pendekatan signature-based yang bergantung pada pola serangan known untuk matching eksak, pendekatan *classification-based* pada *Network Intrusion Detection System* (NIDS) memanfaatkan teknik *supervised machine learning* untuk melatih model mengenali dan mengklasifikasikan pola trafik jaringan menjadi kelas normal atau berbagai jenis serangan secara langsung. Model ini dibangun menggunakan dataset berlabel yang mencakup contoh trafik normal dan serangan, sehingga memungkinkan pembelajaran fitur diskriminatif untuk klasifikasi *multiclass* yang akurat (Ferrag dkk., 2020). Keunggulan pendekatan ini terletak pada kemampuannya mendeteksi serangan known dengan presisi tinggi serta potensi generalisasi terhadap varian serangan baru melalui fitur yang dipelajari. Oleh karena itu, penelitian ini mengadopsi pendekatan klasifikasi supervised, di mana model dilatih menggunakan data trafik normal dan serangan untuk mengenali pola keduanya serta mengklasifikasikan jenis ancaman secara spesifik.

2.2 CRISP-DM

Cross-Industry Standard Process for Data Mining (CRISP-DM) merupakan kerangka kerja proses yang digunakan secara luas untuk merancang dan mengelola proyek berbasis data. Kerangka ini menyediakan struktur metodologis yang jelas dalam menghubungkan tujuan penelitian dengan tahapan teknis pengolahan dan analisis data (Schröer dkk., 2021). Dalam *CRISP-DM*, siklus penelitian dibagi ke dalam enam tahap utama yang saling berkaitan sebagai berikut.

a. *Business Understanding*

   Tahap *business understanding* dalam kerangka CRISP-DM berfokus pada perumusan tujuan bisnis dan identifikasi permasalahan utama yang ingin diselesaikan. Pada tahap ini, kebutuhan dan batasan sistem dianalisis untuk memastikan arah penelitian sesuai dengan konteks yang dihadapi (Schröer dkk., 2021).

b. *Data Understanding*

   Tahap *data understanding* mencakup eksplorasi awal terhadap dataset yang digunakan, serta potensi permasalahan kualitas data. Tahap ini bertujuan memastikan data yang digunakan relevan dan layak untuk proses analisis dan pemodelan lebih lanjut (Schröer dkk., 2021).

c. *Data Preparation*

Tahap *data preparation* mencakup seluruh proses pengolahan data mentah hingga siap digunakan dalam pemodelan. Kegiatan pada tahap ini meliputi pembersihan data, transformasi fitur (Schröer dkk., 2021).

d. *Modeling*

Pada tahap *modeling* dalam kerangka CRISP-DM berfokus pada pemilihan dan penerapan algoritma sesuai tujuan penelitian (Schröer dkk., 2021).

e. Evaluation

Digunakan untuk menilai performa model global yang dihasilkan berdasarkan metrik evaluasi yang telah ditetapkan, seperti akurasi dan *F1-score*. Selain performa prediktif, evaluasi juga mempertimbangkan aspek efisiensi proses pelatihan dan stabilitas model pada kondisi distribusi data yang heterogen (Schröer dkk., 2021).

f. *Deployment*

Pada *CRISP-DM* secara umum mencakup penerapan model ke lingkungan operasional. Namun, dalam penelitian ini, tahap deployment tidak implementasi pada penelitian in (Schröer et al., 2021).

2.3 Dataset ToN-IoT

Dataset ToN-IoT merupakan dataset lalu lintas jaringan heterogen yang dikembangkan oleh Cyber Range Lab, UNSW Canberra, untuk mendukung penelitian deteksi intrusi pada lingkungan *Internet of Things* (IoT) dan Industrial IoT (IIoT) (Moustafa, 2021). Dataset ini dikumpulkan dari sebuah *testbed* yang dirancang menyerupai lingkungan jaringan nyata, dengan arsitektur berlapis yang terdiri dari *edge*, *fog*, dan *cloud*, yang memungkinkan dataset merepresentasikan karakteristik lalu lintas jaringan yang kompleks.

Testbed ToN-IoT menghasilkan lalu lintas jaringan normal dan berbahaya dengan memanfaatkan berbagai skenario serangan siber, sehingga dataset ini banyak digunakan sebagai *benchmark* dalam penelitian *Network Intrusion Detection System* (NIDS). Dataset mencakup sembilan kategori serangan, yaitu *Distributed Denial of Service* (DDoS), *Denial of Service* (DoS), *Ransomware*, *Backdoor*, *Injection*, *Cross-Site Scripting* (XSS), *Password attacks*, *Scanning*, dan *Man-in-the-Middle* (MITM), serta lalu lintas normal (Moustafa, 2021).

Secara struktural, dataset ToN-IoT terdiri dari 46 atribut yang dikelompokkan ke dalam tujuh kategori fitur, mencerminkan informasi koneksi, statistik trafik, serta karakteristik protokol aplikasi. Pembagian ini memungkinkan analisis lalu lintas jaringan dilakukan dari berbagai lapisan, mulai dari transport hingga aplikasi.

a. *Connection Features*

Tabel 2.1 *Connection Features*

ID	Feature	Type	Description
1	ts	Time	Timestamp koneksi antar flow identifiers
2	src_ip	String	Alamat IP sumber
3	src_port	Number	Port TCP/UDP sumber
4	dst_ip	String	Alamat IP tujuan
5	dst_port	Number	Port TCP/UDP tujuan
6	proto	String	Protokol transport layer
7	service	String	Protokol yang terdeteksi dinamis (DNS, HTTP, SSL)
8	duration	Number	Durasi koneksi paket
9	src_bytes	Number	Bytes dari source payload
10	dst_bytes	Number	Bytes dari destination payload
11	conn_state	String	Status koneksi (S0, S1, REJ, dll)
12	missed_bytes	Number	Bytes yang hilang dalam content gaps

Sumber: (Moustafa, 2021)

Kelompok fitur Tabel 2.1 merepresentasikan informasi dasar koneksi jaringan, seperti alamat IP, *port*, protokol, durasi koneksi, serta status koneksi. Fitur-fitur ini memberikan gambaran umum mengenai aliran trafik antar node jaringan.

b. Statistical Features

Tabel 2.2 *Statistical Features*

ID	Feature	Type	Description
13	src_pkts	Number	Jumlah paket dari source
14	src_ip_bytes	Number	Total IP header bytes dari source
15	dst_pkts	Number	Jumlah paket dari destination
16	dst_ip_bytes	Number	Total IP header bytes dari destination

Sumber: (Moustafa, 2021)

Pada Tabel 2.2 menangkap karakteristik kuantitatif lalu lintas jaringan, seperti jumlah paket dan total *byte* yang ditransmisikan, yang sering digunakan untuk mengidentifikasi pola anomali berbasis volume trafik.

c. DNS *Features*

Tabel 2.3 DNS *Features*

ID	Feature	Type	Description
17	dns_query	String	Domain name dari DNS queries
18	dns_qclass	Number	DNS query class
19	dns_qtype	Number	DNS query type

20	dns_rcode	Number	Response code DNS
22	dns_RD	Boolean	Recursion desired (T/F)
23	dns_RA	Boolean	Recursion available (T/F)
24	dns_rejected	Boolean	DNS query rejected (T/F)

Sumber: (Moustafa, 2021)

Kelompok fitur Tabel 2.3 berisi informasi terkait permintaan dan respons *domain name system*, yang relevan untuk mendeteksi aktivitas mencurigakan seperti *tunneling* atau penyalahgunaan layanan DNS.

d. SSL *Features*

Tabel 2.4 SSL *Features*

ID	Feature	Type	Description
25	ssl_version	String	Versi SSL yang ditawarkan server
26	ssl_cipher	String	SSL cipher suite yang dipilih
27	ssl_resumed	Boolean	Session dapat digunakan untuk koneksi baru
28	ssl_established	Boolean	Koneksi SSL berhasil established
29	ssl_subject	String	Subject dari X.509 certificate
30	ssl_issuer	String	Certificate authority

Sumber: (Moustafa, 2021)

Fitur Tabel 2.4 merepresentasikan karakteristik koneksi terenkripsi, termasuk versi protokol, *cipher suite*, serta informasi sertifikat, yang dapat digunakan untuk menganalisis perilaku koneksi aman yang tidak wajar.

e. HTTP *Features*

Tabel 2.5 HTTP *Features*

ID	Feature	Type	Description
31	http_trans_depth	Number	Pipelined depth koneksi HTTP
32	http_method	String	HTTP method (GET, POST, HEAD)
33	http_uri	String	URI dalam request HTTP
34	http_referrer	String	Nilai header referer
35	http_version	String	Versi HTTP (1.1, dll)
36	http_request_body_len	Number	Ukuran content dari HTTP client
37	http_response_body_len	Number	Ukuran content dari HTTP server
38	http_status_code	Number	Status code HTTP
39	http_user_agent	Number	Nilai header User-Agent
40	http_orig_mime_types	String	MIME types dari source
41	http_resp_mime_types	String	MIME types dari destination

Sumber: (Moustafa, 2021)

Kelompok fitur Tabel 2.5 merepresentasikan aktivitas pada lapisan aplikasi, yang sangat penting untuk mendeteksi serangan berbasis web seperti *Injection* atau *Cross-Site Scripting* (XSS). Fitur-fitur ini memungkinkan sistem untuk menganalisis isi permintaan *client* dan respons server guna mengidentifikasi pola akses yang tidak wajar atau muatan paket yang mencurigakan.

f. Violation Features

Tabel 2.6 *Violation Features*

ID	Feature	Type	Description
42	weird_name	String	Nama-nama anomali/ violations yang terkait dengan protokol yang terjadi
43	weird_addl	String	Informasi tambahan yang terkait dengan anomali/ violations protokol
44	weird_notice	Boolean	Menandakan jika violations /anomali telah diubah menjadi peringatan

Sumber: (Moustafa, 2021)

Kelompok fitur Tabel 2.6 mencatat anomali atau pelanggaran protokol yang terdeteksi selama sesi komunikasi berlangsung. Fitur-fitur ini sangat berguna untuk mengidentifikasi aktivitas yang menyimpang dari standar protokol jaringan (RFC) yang sering kali menjadi indikasi awal adanya upaya eksploitasi. Atribut ini membantu model dalam menangkap perilaku paket yang tidak standar, seperti malformasi paket atau ketidaksesuaian struktur protokol, yang sulit dideteksi hanya melalui fitur statistik biasa.

g. Labelling Attributes

Tabel 2.7 *Labelling Attributes*

ID	Feature	Type	Description
45	label	Number	Tag normal (0) dan attack (1)
46	type	String	Kategori serangan (normal, DoS, DDoS, backdoor, dll)

Sumber: (Moustafa, 2021)

Kelompok fitur Tabel 2.7 merupakan atribut target yang digunakan untuk proses pembelajaran model. Atribut ini memberikan informasi mengenai status keamanan dari setiap baris data dalam dataset. Fitur *label* digunakan untuk skenario *binary classification*, sedangkan fitur *type* digunakan untuk skenario *multiclass classification* guna mengidentifikasi jenis serangan secara lebih mendetail.

h. Distribusi Data dan Karakteristik Kelas

Penelitian ini menggunakan subset *train_test_network.csv* dari dataset ToN-IoT sebagai data referensi dalam proses pemodelan. Dataset lengkap beserta subset yang digunakan dalam penelitian ini tersedia secara publik dan dapat diunduh melalui tautan berikut. Distribusi jumlah *record* untuk setiap kategori trafik dan serangan disajikan pada Tabel 2.8.

Tabel 2.8 Distribusi Data dan Karakteristik Kelas

Attack Type	**Total Record (Full Dataset)**	**Train Test Network Record**
backdoor	508,116	20,000
ddos	6,165,008	20,000
dos	3,375,328	20,000
injection	452,659	20,000
mitm	1,052	1,043
password	1,718,568	20,000
ransomware	72,805	20,000
scanning	7,140,161	20,000
xss	2,108,944	20,000
normal	796,380	50,000

Sumber: (Moustafa, 2021)

Karakteristik utama yang menonjol dari dataset ini adalah ketimpangan distribusi kelas (*class imbalance*). Sebagai contoh, kelas *Man-in-the-Middle* (MITM) hanya memiliki 1,043 *record* (0,49%), sementara kelas normal mendominasi dengan 50,000 *record* (23,69%). Ketimpangan ini sengaja dipertahankan karena merepresentasikan kondisi riil di lingkungan jaringan, di mana serangan tertentu memiliki frekuensi kemunculan yang jauh lebih rendah dibandingkan aktivitas normal atau serangan umum lainnya (Moustafa, 2021).

Secara keseluruhan, dataset ToN-IoT menyediakan representasi lalu lintas jaringan yang komprehensif untuk lingkungan *smart city* melalui keragaman fitur aplikasi dan variasi kategori serangan. Karakteristik dataset yang memiliki ketimpangan distribusi kelas (*class imbalance*) serta volume data yang besar menjadikannya instrumen yang relevan untuk menguji ketangguhan model deteksi intrusi. Dalam konteks penelitian ini, subset data tersebut akan didistribusikan ke beberapa *client* guna mensimulasikan kondisi *Non-Independent and Identically Distributed* (Non-IID), yang merupakan tantangan utama dalam implementasi *Federated Learning* di jaringan terdistribusi.

2.4 Teknik Pre-Processing

Pre-processing data adalah tahapan pengolahan data mentah untuk meningkatkan kualitas dan efisiensi proses pembelajaran mesin. Tahapan ini bertujuan untuk memastikan data memiliki konsistensi dan format yang sesuai dengan persyaratan algoritma yang digunakan (Qazi dkk., 2022a). Beberapa konsep utama dalam *pre-processing* meliputi:

a. Penanganan Data Tidak Lengkap (*Missing Values*)

Dalam pengumpulan data berskala besar, sering kali terdapat informasi yang hilang atau tidak tercatat. Penanganan data ini dapat dilakukan melalui teknik penghapusan (*deletion*) atau pengisian nilai pengganti (*imputation*). Pemilihan teknik pengisian harus mempertimbangkan karakteristik data agar tidak mengubah distribusi asli dari informasi tersebut secara signifikan (Moustafa, 2021).

b. Transformasi Fitur Kategorikal

Algoritma pembelajaran mesin, khususnya *deep learning*, bekerja atas dasar perhitungan matematis yang memerlukan input numerik. Oleh karena itu, fitur yang bertipe kategori harus ditransformasikan ke dalam representasi angka. Proses ini memastikan bahwa informasi kualitatif dapat diproses oleh model tanpa menghilangkan perbedaan antarkategori (Alsaedi dkk., 2020).

c. Reduksi Dimensi dan Seleksi Fitur

Reduksi dimensi dan seleksi fitur merupakan tahap penting dalam prapemrosesan data untuk meningkatkan kinerja model. Seleksi fitur berfokus pada pemilihan atribut yang paling relevan dan informatif, sedangkan reduksi dimensi bertujuan untuk menurunkan jumlah variabel dengan cara mentransformasikan data ke ruang berdimensi lebih rendah tanpa kehilangan informasi yang signifikan. Dengan mengurangi atribut yang tidak berkontribusi secara signifikan, kompleksitas komputasi dapat ditekan dan risiko *overfitting* dapat diminimalkan. Selain itu, proses ini membantu model untuk lebih fokus pada pola-pola utama yang benar-benar membedakan setiap kelas data, sehingga meningkatkan efisiensi dan akurasi pembelajaran (Al-Ghadi dkk., 2025).

d. Penskalaan Fitur (*Feature Scaling*)

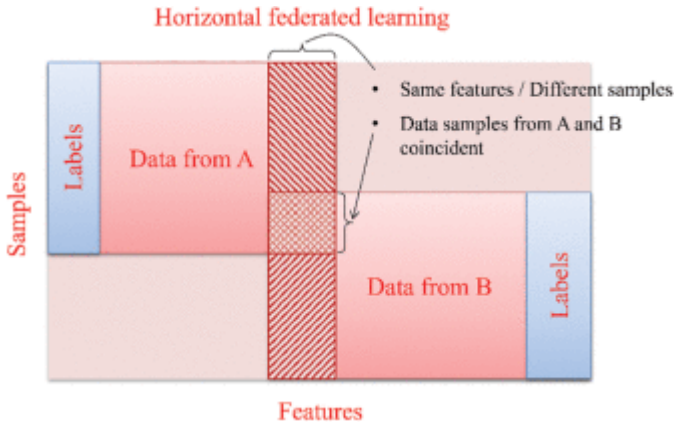
Penskalaan fitur bertujuan untuk menyamakan rentang nilai antar atribut yang berbeda. Tanpa penskalaan, fitur dengan rentang nilai yang besar akan mendominasi perhitungan fungsi biaya dan menghambat efisiensi pembaruan bobot dalam model. Teknik seperti standardisasi memastikan seluruh fitur berada pada skala yang sebanding, sehingga proses pelatihan model menjadi lebih stabil (Javeed dkk., 2024).

2.5 *Federated Learning*

Federated Learning (FL) adalah paradigma pembelajaran mesin terdistribusi yang memungkinkan pelatihan model dilakukan secara kolaboratif pada perangkat *client* yang berbeda tanpa perlu memindahkan data mentah ke server pusat (McMahan dkk., 2017). Prinsip utama dari pendekatan ini adalah melakukan proses komputasi di tempat data tersebut berada, yang secara fundamental berbeda dari pendekatan pembelajaran mesin tradisional yang bersifat tersentralisasi.

Dalam arsitektur *Federated Learning* (FL), server pusat bertugas mengelola model global dan mendistribusikannya ke sejumlah *client* atau node. Setiap *client* melatih model tersebut menggunakan data lokal masing-masing, kemudian mengirimkan kembali pembaruan parameter model, seperti bobot atau gradien, ke server pusat untuk proses agregasi. Mekanisme ini memberikan keunggulan dalam menjaga privasi data sensitif, mengurangi kebutuhan *bandwidth* akibat transmisi data berukuran besar, serta membantu pemenuhan regulasi perlindungan data (Upreti dkk., 2024).

2.5.1 Arsitektur *Horizontal Federated Learning*



Gambar 2.1 Ilustrasi *Horizontal Federated Learning*

Sumber: (Upreti dkk., 2024).

Pada Gambar 2.1 menggambarkan konsep *Horizontal Federated Learning* (HFL) berdasarkan hubungan antara fitur, sampel, dan label pada beberapa sumber data. Pada ilustrasi tersebut, sumbu horizontal merepresentasikan fitur, sedangkan sumbu vertikal menunjukkan sampel data. Dataset yang berasal dari *client* A dan *client* B memiliki struktur fitur dan label

yang identik, dengan perbedaan pada jumlah serta distribusi sampel data yang dimiliki. Kondisi ini mencerminkan karakteristik utama HFL, yaitu *same features, different samples*, di mana setiap *client* melakukan pelatihan model secara lokal tanpa membagikan data mentah. Pendekatan ini sejalan dengan penelitian ini yang menggunakan skenario HFL, di mana setiap *client* atau gateway memiliki struktur fitur yang sama dan mengumpulkan data dari lokasi atau populasi pengguna yang berbeda. Skema tersebut sangat relevan untuk ekosistem *smart city*, karena berbagai gateway IoT umumnya memantau parameter jaringan yang serupa dan berasal dari lingkungan operasional yang berbeda, sehingga memungkinkan kolaborasi pembelajaran model secara terdistribusi dengan tetap menjaga privasi data.

2.5.2 Karakteristik Distribusi Data (IID dan Non-IID)

Dalam implementasi FL, distribusi data antar *client* dibagi menjadi dua kategori utama yang sangat menentukan performa model global:

a. *Independent and Identically Distributed* (IID)

Independent and Identically Distributed (IID) merupakan kondisi ideal dalam *Federated Learning*, di mana setiap *client* memiliki distribusi data yang serupa dengan distribusi populasi global, baik dari segi proporsi kelas maupun karakteristik fitur. Pada skenario ini, data pada masing-masing *client* dapat dianggap sebagai sampel acak yang merepresentasikan keseluruhan dataset.

Dalam kondisi IID, algoritma agregasi standar seperti *Federated Averaging* (FedAvg) cenderung mencapai konvergensi dengan cepat dan stabil, karena gradien lokal yang dihasilkan oleh setiap *client* relatif konsisten satu sama lain. Oleh sebab itu, skenario IID sering digunakan sebagai *baseline* untuk mengevaluasi performa Federated Learning sebelum diuji pada kondisi yang lebih realistis dan kompleks (Zhu dkk., 2021).

b. *Non-Independent and Identically Distributed* (Non-IID)

Non-Independent and Identically Distributed (Non-IID) menggambarkan kondisi di mana distribusi data antar *client* berbeda secara signifikan, baik dalam hal jumlah sampel, proporsi kelas, maupun pola fitur. Pada lingkungan *smart city*, kondisi ini muncul secara alami karena setiap *gateway* melayani area dengan karakteristik trafik yang berbeda, seperti kawasan residensial, industri, atau fasilitas publik, yang menghasilkan pola serangan dan lalu lintas jaringan yang tidak seragam (Zhu dkk., 2021).

2.5.3 Dirichlet

Untuk memodelkan dan menganalisis kondisi Non-IID secara sistematis, sejumlah penelitian mengadopsi Distribusi Dirichlet sebagai mekanisme matematis dalam mensimulasikan ketimpangan distribusi kelas antar klien. Distribusi ini memungkinkan pengaturan tingkat heterogenitas data melalui parameter konsentrasi (α), di mana nilai α yang lebih kecil merepresentasikan kondisi Non-IID yang lebih ekstrem (heterogenitas tinggi), sedangkan nilai α yang lebih besar mendekati distribusi IID (heterogenitas rendah) (Zhou dkk., 2025). Dalam praktik literatur *Federated Learning* terutama pada dataset benchmark seperti CIFAR-10 atau MNIST nilai α sangat kecil (0.01–0.1) umumnya dianggap sebagai kondisi Non-IID sangat ekstrem (beberapa klien dominan hanya 1–2 kelas), α 0.1–0.5 sebagai ekstrem hingga sangat heterogen (sering untuk worst-case testing), α 0.5–1.0 (kadang hingga 2) sebagai heterogenitas sedang hingga moderat (nyata tapi wajar, dengan semua kelas masih ada di hampir semua klien), α 1–5 sebagai transisi sedang ke ringan (distribusi mulai mirip antar klien), α 5–10 sebagai ringan atau mendekati IID (perbedaan distribusi minim), serta $\alpha \geq 50$–100 sebagai hampir IID atau baseline seragam. Interpretasi ini tetap bergantung pada faktor seperti jumlah kelas dataset dan jumlah klien, tanpa adanya kesepakatan mutlak di literatur (Zhu dkk., 2021). Pendekatan berbasis Dirichlet banyak digunakan karena fleksibilitasnya dalam merepresentasikan berbagai tingkat ketidakseragaman data tanpa mengubah struktur fitur dasar dataset. Dengan demikian, penggunaan distribusi Dirichlet menjadi landasan teoretis yang relevan untuk mengevaluasi ketahanan dan stabilitas algoritma *Federated Learning* pada skenario distribusi data yang menyerupai kondisi nyata smart city.

2.5.4 Algoritma Agregasi

Algoritma agregasi berperan krusial dalam menggabungkan pembaruan model dari berbagai *client* untuk membentuk satu model global yang representatif. Penelitian ini membandingkan dua algoritma utama yang menjadi standar dalam literatur *Federated Learning*.

Federated Averaging

Federated Averaging (FedAvg) adalah algoritma agregasi dasar yang bekerja dengan prinsip rata-rata tertimbang (*weighted average*). Dalam algoritma ini, server pusat melakukan inisialisasi model global, kemudian mengirimkannya ke sejumlah *client* untuk dilatih secara lokal menggunakan data masing-masing selama beberapa *epoch*. Parameter model global

diperbarui berdasarkan rata-rata parameter model lokal yang dibobotkan dengan jumlah sampel data pada setiap *client* (McMahan dkk., 2017).

Formula agregasi FedAvg dinyatakan pada Persamaan (2.1).

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k} \quad (2.1)$$

Di mana w_{t+1} adalah parameter model global pada ronde $t+1$, n_k adalah jumlah sampel pada *client* k, n adalah total sampel dari seluruh *client* yang berpartisipasi, dan w_{t+1}^{k} adalah parameter hasil pelatihan lokal pada *client* k.

Federated Proximal

FedProx merupakan pengembangan dari FedAvg yang dirancang khusus untuk mengatasi tantangan heterogenitas data (Non-IID). Algoritma ini memperkenalkan *proximal term* pada fungsi tujuan (*loss function*) lokal untuk membatasi agar pembaruan model lokal tidak menyimpang terlalu jauh dari model global. Penambahan suku penalti ini secara teoretis mampu menekan fenomena *client drift* yang merupakan fenomena di mana model lokal pada setiap perangkat "melenceng" dari tujuan global karena data yang tidak seragam (non-IID), sehingga memperlambat konvergensi dan menurunkan akurasi model utama dan sering terjadi pada dataset heterogen seperti ToN-IoT (Li dkk., 2020).

Formula *loss function* pada sisi *client* dalam FedProx dinyatakan pada Persamaan (2.2).

$$\min_{w} h_{k(w;\, w^t)} = F_{k(w)} + \left(\frac{\mu}{2}\right) \|w - w^t\|^2 \quad (2.2)$$

Di mana $F_{k(w)}$ adalah fungsi *loss* tradisional, sedangkan $\left(\frac{\mu}{2}\right) \|w - w^t\|^2$ adalah *proximal term* dengan *hypermarameter* μ yang mengatur kekuatan penalti terhadap penyimpangan model lokal w dari model global w^t.

Tabel 2.9 Perbedaan FedAvg dengan FedProx

Aspek	FedAvg	FedProx

Fungsi Loss	Standar (misal: Cross-Entropy)	Standar dengan tambahan *Proximal Term*
Mekanisme Agregasi	*Weighted Average*	*Weighted Average*
Stabilitas Non-IID	Rentan terhadap *client drift*	Lebih stabil dan konsisten
Hyperparameter	Learning Rate, Epoch, Batch Size	Tambahan Koefisien μ
Konvergensi	Cepat pada data IID	Lebih lambat tapi stabil pada Non-IID

Perbedaan yang disajikan pada Tabel 2.9 menunjukkan bahwa perbedaan fundamental antara FedAvg dan FedProx terletak pada modifikasi fungsi tujuan di sisi *client*. Sementara FedAvg mengasumsikan bahwa setiap model lokal dapat berkontribusi secara langsung melalui rata-rata tertimbang, FedProx mengakui adanya risiko ketidakstabilan pada data yang heterogen (Non-IID).

Penggunaan *proximal term* (parameter μ) pada FedProx menjadi fitur kunci yang memberikan fleksibilitas bagi model untuk beradaptasi dengan variasi data lokal tanpa mengorbankan stabilitas model global. Meskipun FedAvg menawarkan kecepatan konvergensi yang lebih tinggi pada kondisi data yang seragam (IID), FedProx secara teoretis lebih stabil dalam memitigasi dampak *client drift* yang sering terjadi pada jaringan *smart city* yang kompleks.

2.6 Deep Learning

Deep Learning merupakan subbidang dari *Machine Learning* yang didasarkan pada konsep jaringan saraf tiruan (*Artificial Neural Networks*). Perbedaan utama *Deep Learning* terletak pada penggunaan struktur lapisan *(layer)* yang dalam dan hierarkis untuk mengekstraksi representasi data secara otomatis. Tidak seperti algoritma tradisional yang memerlukan rekayasa fitur manual secara intensif, *Deep Learning* mampu mempelajari pola-pola kompleks langsung dari data mentah melalui proses komputasi yang berlapis-lapis (LeCun dkk., 2015).

Dalam konteks keamanan jaringan, *Deep Learning* menawarkan keunggulan dalam menangani volume data trafik yang sangat besar dan mendeteksi anomali yang halus yang mungkin terlewatkan oleh metode berbasis statistik konvensional. Kemampuannya untuk

melakukan generalisasi terhadap pola serangan baru menjadikan teknologi ini sangat relevan untuk diimplementasikan dalam sistem deteksi intrusi yang dinamis (Ahmad dkk., 2021).

Penelitian ini secara spesifik mengkaji dua arsitektur Deep Learning sebagai model lokal dalam skema *Federated Learning* yaitu *Multi-Layer Perceptron* (MLP) dan *Convolutional Neural Network 1 Dimensional (CNN-1D).*

2.6.1 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) adalah varian jaringan saraf *feedforward* yang terdiri dari lapisan input, satu atau lebih lapisan tersembunyi (*hidden layers*), dan lapisan output yang terhubung secara penuh (*fully-connected*). Prinsip kerja MLP didasarkan pada kemampuan unit perseptron untuk mempelajari pemetaan non-linear antara input dan output melalui proses *backpropagation* (Ali dkk., 2024)

Secara matematis, operasi pada setiap lapisan MLP melibatkan perkalian matriks bobot (W), penambahan bias (b), dan penerapan fungsi aktivasi non-linear (f) untuk menghasilkan representasi fitur yang lebih abstrak dinyatakan pada Persamaan (2.3).

$$h^{(l)} = f\left(W^{(l)} \cdot h^{(l-1)} + b^{(l)}\right) \tag{2.3}$$

Dalam konteks NIDS, MLP efektif untuk memproses data tabular seperti statistik paket karena kemampuannya dalam menangkap hubungan kompleks antar fitur yang telah diekstraksi. Penggunaan fungsi aktivasi seperti *Rectified Linear Unit* (ReLU) membantu mempercepat konvergensi, sementara *Sigmoid* atau *Softmax* digunakan pada lapisan akhir untuk menghasilkan probabilitas prediksi serangan.

2.6.2 Convolutional Neural Network 1 Dimensional (CNN-1D)

Convolutional Neural Network 1 Dimensional (CNN-1D) merupakan adaptasi dari arsitektur CNN yang umumnya digunakan pada data citra yang dioptimalkan untuk data sekuensial atau satu dimensi. Dalam deteksi intrusi jaringan, CNN 1D bekerja dengan menggeser filter (kernel) sepanjang urutan fitur untuk mengekstraksi pola lokal pada urutan atau hubungan antar fitur yang berdekatan (Ahmad dkk., 2021).

Secara matematis, operasi pada lapisan konvolusi melibatkan pergeseran filter (kernel) di atas data input, di mana setiap elemen filter dikalikan dengan nilai input yang sesuai dan

dijumlahkan untuk menghasilkan *feature map*. Operasi konvolusi 1D didefinisikan sebagai pada Persamaan (2.4).

$$(x \cdot k)[i] = \Sigma_{m=0}^{M-1} x[i+m] \cdot k[m] \quad (2.4)$$

Dalam formula tersebut, x merepresentasikan sinyal input (fitur jaringan), k adalah filter atau kernel dengan panjang M, dan i adalah indeks posisi pada *feature map* yang dihasilkan.

2.7 *Hyperparameter*: *Grid Search*

Optimasi *hyperparameter* merupakan tahapan penting dalam pengembangan model pembelajaran mesin dan *deep learning*, yang bertujuan untuk menentukan konfigurasi parameter eksternal model yang tidak dipelajari secara otomatis selama proses pelatihan. Pemilihan *hyperparameter* yang tepat dapat memengaruhi kecepatan konvergensi serta performa akhir model dalam mendeteksi ancaman jaringan (Kunang dkk., 2021a).

2.7.1 Mekanisme *Grid Search*

Grid Search merupakan salah satu metode optimasi *hyperparameter* yang paling umum digunakan, dengan mekanisme pencarian sistematis melalui seluruh kombinasi nilai kandidat yang telah ditentukan dalam ruang pencarian (search space). Pendekatan ini bersifat *exhaustive*, sehingga setiap konfigurasi *hyperparameter* diuji secara menyeluruh (Kunang dkk., 2021a).

2.7.2 Parameter Kunci dalam Pembelajaran Terdistribusi

Dalam lingkungan *Federated Learning*, optimasi *hyperparameter* tidak hanya berfokus pada peningkatan akurasi model, tetapi juga pada keseimbangan antara beban komputasi lokal dan performa global model (McMahan dkk., 2017). Beberapa *hyperparameter* yang umum dioptimalkan dalam studi *Federated Learning* meliputi:

a. *Learning Rate*, merupakan kontrol besaran pembaruan bobot model dan berpengaruh langsung terhadap stabilitas konvergensi.
b. *Batch Size*, merupakan pentukan jumlah sampel yang diproses sebelum pembaruan gradien dan memengaruhi efisiensi pelatihan di sisi *client*.
c. *Epoch* Lokal, merupakan pengatur jumlah iterasi pelatihan lokal sebelum proses agregasi model global, serta memengaruhi tingkat kontribusi pembelajaran lokal (Li dkk., 2020).

2.7.3 Strategi Pemilihan *Hyperparameter* Berbasis Multikriteria

Beberapa penelitian mengusulkan pemilihan konfigurasi *hyperparameter* berdasarkan pendekatan multikriteria, dengan mempertimbangkan lebih dari satu metrik evaluasi, seperti Akurasi, *F1-Score*, dan *AUC-ROC*. Pendekatan ini bertujuan untuk memperoleh konfigurasi yang tidak hanya unggul pada satu metrik tertentu, tetapi juga stabil secara keseluruhan dan efisien dari sisi komputasi.

Dalam literatur, pendekatan ini sering dikaitkan dengan konsep *trade-off* performa dan efisiensi, serta dapat dianalisis menggunakan prinsip Pareto Optimal untuk menghindari konfigurasi yang dominan pada satu aspek namun lemah pada aspek lainnya (Deb dkk., 2002).

2.8 Teknik Evaluasi

Evaluasi kinerja sistem deteksi intrusi berbasis *Federated Learning* umumnya dilakukan melalui dua aspek utama, yaitu kualitas prediksi model dan efisiensi operasional pada lingkungan terdistribusi. Pada konteks keamanan jaringan *smart city*, kedua aspek ini sama-sama penting karena sistem tidak hanya dituntut akurat, tetapi juga stabil dan efisien ketika diimplementasikan pada banyak *gateway* dengan karakteristik data yang heterogen.

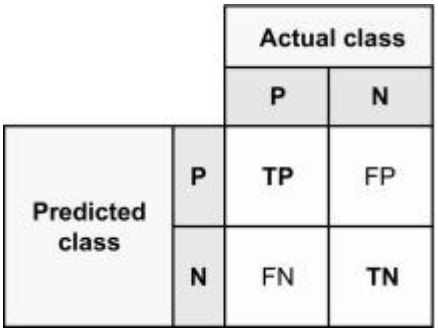
2.8.1 Confusion Matrix

Confusion Matrix merupakan alat evaluasi fundamental yang digunakan untuk merepresentasikan performa algoritma klasifikasi dengan membandingkan label aktual dan hasil prediksi model. Matriks ini memberikan informasi rinci mengenai kemampuan model dalam mengidentifikasi kelas positif (serangan) dan kelas negatif (lalu lintas normal), serta jenis kesalahan yang dihasilkan selama proses prediksi (Tiwari, 2022).

Struktur Confusion Matrix pada klasifikasi biner terdiri dari empat komponen utama:

a. *True Positive* (TP) : Serangan yang diprediksi benar sebagai serangan
b. *True Negative* (TN) : Aktivitas normal yang diprediksi benar sebagai normal
c. *False Positive* (FP) : Aktivitas normal yang salah diprediksi sebagai serangan (*false alarm*)
d. *False Negative* (FN): Serangan yang salah diprediksi sebagai aktivitas normal, yang berpotensi menyebabkan kebocoran keamanan

Pada skenario klasifikasi *multiclass*, Confusion Matrix diperluas menjadi matriks berdimensi $C \times C$, dengan C merepresentasikan jumlah kelas serangan, sehingga distribusi kesalahan prediksi antarkelas dapat dianalisis secara lebih spesifik.



Gambar 2.2 Ilustrasi *Confusion Matrix*

Sumber: (Tiwari, 2022)

Pada Gambar 2.2 dapat didefinisi sebagai berikut:

a. *True Positive* (TP) merupakan kondisi ketika model memprediksi kelas target secara benar.
b. *True Negative* (TN) menunjukkan kondisi ketika model secara tepat mengklasifikasikan data yang bukan termasuk kelas target.
c. *False Positive* (FP) terjadi ketika data yang bukan kelas target diprediksi sebagai kelas target yang dalam konteks keamanan jaringan sering disebut sebagai *false alarm*.
d. *False Negative* (FN) merepresentasikan kondisi ketika model gagal mendeteksi data yang sebenarnya termasuk ke dalam kelas target.

Pada permasalahan klasifikasi *multiclass*, *confusion matrix* dapat diperluas menjadi matriks berdimensi $C \times C$. Setiap baris merepresentasikan kelas aktual, sedangkan setiap kolom menunjukkan kelas hasil prediksi. Elemen pada diagonal utama menggambarkan prediksi yang benar, sementara elemen di luar diagonal menunjukkan pola misklasifikasi antarkelas yang dapat digunakan untuk analisis kesalahan model secara lebih mendalam.

2.8.2 Metrik Evaluasi Kinerja Model Klasifikasi

Berdasarkan nilai-nilai pada Confusion Matrix, performa model klasifikasi umumnya diukur menggunakan beberapa metrik evaluasi kinerja model klasifikasi sebagai berikut:

a. Akurasi (*Accuracy*)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Pada persamaan (2.5) mengukur proporsi prediksi yang benar terhadap seluruh sampel data (Tiwari, 2022).

b. Presisi (*Precision*)

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

Pada persamaan (2.6) menunjukkan tingkat keandalan model ketika memprediksi suatu aktivitas sebagai serangan (Tiwari, 2022).

c. Recall

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

Pada persamaan (2.7) mengukur kemampuan model dalam mendeteksi seluruh serangan yang terjadi. Dalam konteks keamanan jaringan, metrik ini sangat krusial karena berkaitan langsung dengan minimisasi *false negative (Tiwari, 2022)*.

d. F1-Score

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

Pada persamaan (2.8) merupakan rata-rata harmonik antara Presisi dan Recall, yang memberikan evaluasi performa lebih seimbang, terutama pada dataset dengan ketimpangan kelas (Tiwari, 2022).

e. *ROC-AUC* (Receiver Operating Characteristic – Area Under Curve)

Digunakan untuk mengukur kemampuan model dalam membedakan antar kelas pada berbagai ambang batas keputusan. Nilai *AUC* yang mendekati 1 menunjukkan kemampuan separabilitas yang sangat baik.

2.8.3 Evaluasi Kinerja Model *Federated Learning*

Evaluasi kinerja model *Federated Learning* tidak hanya berfokus pada kualitas prediksi, tetapi juga mencakup aspek efisiensi dan stabilitas pelatihan dalam lingkungan terdistribusi. Beberapa indikator yang umum digunakan dalam evaluasi ini meliputi.

a. Waktu Pelatihan (*Training Latency*)

Menggambarkan durasi yang dibutuhkan untuk menyelesaikan satu ronde pelatihan *federated*, yang mencakup pelatihan lokal di sisi *client* serta proses agregasi model di server pusat (McMahan dkk., 2017).

b. Dampak Heterogenitas Data (Parameter *Dirichlet* α)

Distribusi data Non-IID antar *client* sering dimodelkan menggunakan distribusi Dirichlet dengan parameter konsentrasi α. Nilai α yang kecil merepresentasikan heterogenitas data yang tinggi, sedangkan nilai yang lebih besar mendekati kondisi IID. Evaluasi terhadap parameter ini penting untuk mengukur ketangguhan algoritma *Federated Learning* terhadap ketidakseimbangan distribusi data (Zhou dkk., 2025).

c. Stabilitas Parameter Proksimal (μ)

Pada algoritma FedProx, parameter proksimal μ berperan dalam mengendalikan deviasi pembaruan model lokal terhadap model global. Evaluasi terhadap parameter ini digunakan untuk memahami keseimbangan antara akurasi model dan stabilitas konvergensi pada kondisi data heterogeny (Li dkk., 2020).

2.9 Penelitian Terkait

Federated Learning (FL) telah banyak diteliti sebagai pendekatan pembelajaran terdistribusi yang berfokus pada perlindungan privasi data, khususnya pada lingkungan dengan keterbatasan regulasi dan sensitivitas data tinggi. Dalam konteks keamanan jaringan dan sistem terdistribusi, FL menjadi relevan karena memungkinkan proses pelatihan model tanpa memusatkan data mentah dari berbagai node. Namun, tantangan utama dalam penerapan FL adalah heterogenitas data antar *client* (Non-IID), keterbatasan komunikasi, serta variasi kemampuan komputasi perangkat. Oleh karena itu, berbagai penelitian terdahulu berupaya mengembangkan dan mengevaluasi algoritma FL yang lebih stabil dan *robust* terhadap kondisi tersebut yand disajikan pada Tabel 2.10.

Tabel 2.10 Penelitian Terkait

Penulis	Pengaplikasian dalam Bidang Keamanan	Metode FL & Model ML	Dataset	Temuan
(McMahan dkk., 2017)	Fokus pada privasi & keamanan data pengguna: *federated learning* mengurangi risiko kebocoran karena data tetap di perangkat, hanya *update* model yang dikirim.	*Federated Learning* dengan algoritma Federated Averaging (FedAvg); model ML: MLP (2NN), CNN, LSTM (bahasa), word-level LSTM.	MNIST (digit recognition), CIFAR-10 (image classification), Shakespeare (language modeling), dataset besar dari 10 juta posting media sosial.	FedAvg robust terhadap data non-IID dan unbalanced. Komunikasi berkurang hingga 10–100× dibandingkan FedSGD. Model tetap mencapai akurasi tinggi meski data tersebar.
(Li dkk., 2020)	Tidak secara langsung membahas keamanan, namun relevan karena *federated learning* menjaga privasi data (data tetap di perangkat, tidak dikirim ke server).	*Federated Learning* dengan algoritma FedProx (generalization dari FedAvg). Model ML: Logistic Regression, LSTM, dan model klasifikasi lainnya.	Synthetic datasets (dengan variasi heterogenitas α, β), MNIST, FEMNIST, Sentiment140 (tweet sentiment), Shakespeare (next-character prediction).	FedProx lebih stabil dibanding FedAvg pada kondisi heterogen. Memberikan konvergensi lebih robust dan akurasi lebih tinggi.

(Bodagala & Priyanka, 2022)	Deteksi ancaman siber di lingkungan IoT (spoofing, breach, DoS).	Metode: *Federated Learning* (FL). Model: DNN, CNN, RNN.	Bot-IoT (utama), TON_IoT, MQTTset.	Algoritma FL > ML terpusat untuk keamanan & kerahasiaan IoT.
(Hijazi dkk., 2024)		Metode: FL + Fully Homomorphic Encryption (FHE); 4 pendekatan berbasis klaster (OUCM, OECM, MUCM, MECM). Model: CNN.	N-BaIoT (utama), WUSTL-EHMS-2020 (studi kasus).	Kombinasi FL + FHE (khususnya berbasis klaster) efektif mengurangi *overhead* (hingga 89.98%) & latensi (hingga 70.38%) sambil menjaga akurasi.
(Ahmadi & Javidan, 2023)	Deteksi serangan DDoS pada lingkungan IoT perkotaan nyata.	Metode: Federated Deep Learning (antara fog & cloud). Model: LSTM (RNN).	Dataset aktivitas IoT perkotaan nyata (4060 node) + simulasi DDoS.	Model FL mencapai performa sebanding dengan model terpusat, sambil menjaga privasi data & mendistribusikan beban komputasi.
(Thantharate & T, 2023)	Deteksi ancaman siber kolaboratif (Cybria) yang menjaga privasi.	Metode: Horizontal *Federated Learning* (TensorFlow Federated, Flower). Model: Deep Neural Network (DNN).	Bot-IoT.	Kerangka kerja FL (Cybria) mencapai akurasi lebih tinggi (~10%) & konvergensi lebih cepat dibandingkan model DNN terpusat.

(Olanrewaju-George & Pranggono, 2025)	IDS untuk IoT (keamanan & privasi data).	Metode: *Federated Learning* (FedAvgM). Model: Unsupervised AutoEncoder (AE) & Supervised Deep Neural Network (DNN).	N-BaIoT (9 perangkat IoT, botnet Mirai & BASHLITE).	Model unsupervised AE yang dilatih via FL adalah yang terbaik secara keseluruhan, terutama unggul dalam metrik False Positive Rate (FPR) / mengurangi alarm palsu.
(Thakur dkk., 2024)	Deteksi serangan jaringan (IDS) di lingkungan IoT.	Metode: *Federated Learning* (FedAvg). Model: CNN (ekstraksi fitur), KNN (deteksi anomali).	Tidak disebutkan (Implementasi umum pada data IoT).	Pendekatan FL secara signifikan mengungguli model non-FL (terpusat) dalam akurasi & data loss.
(Abdel-Basset dkk., 2024)	Perlindungan privasi data terhadap serangan GAN pada data Non-IID di IoT.	Metode: PP-FDL (FL + arsitektur GAN). Menggunakan 'private identifier' unik per kelas. Agregasi via FedProx. Model: CNN.	MNIST, CIFAR-10.	PP-FDL efektif melindungi data dari serangan GAN pada data Non-IID tanpa penurunan kinerja signifikan (seperti DP) atau *overhead* tinggi (seperti enkripsi).
Penelitian ini	ToN-IoT	HFL+FedAVG HFL+FedProx	Basic MLP Basic CNN	Temuan dibahas di Bab IV

Berdasarkan pemaparan penelitian terkait pada Tabel 2.10, terdapat beberapa celah penelitian (*research gap*) yang menjadi posisi utama dari penelitian ini. Sebagian besar penelitian terdahulu berfokus pada deteksi ancaman IoT secara umum atau menggunakan arsitektur pembelajaran mesin tunggal tanpa melakukan perbandingan mendalam antara efektivitas algoritma agregasi standar dengan algoritma yang dioptimalkan untuk kondisi data yang sangat tidak seragam.

Penelitian ini memposisikan diri dengan melakukan evaluasi komparatif antara algoritma FedAvg sebagai baseline standar dengan FedProx yang dirancang khusus untuk mengatasi tantangan Non-IIDs pada trafik jaringan *smart city*. Berbeda dengan penelitian "Federated Learning-Based Intrusion Detection System for the Internet of Things Using Unsupervised and Supervised Deep Learning Models" yang berfokus pada model *unsupervised* atau "CYBRIA - Pioneering *Federated Learning* for Privacy-Aware Cybersecurity with Brilliance" yang menggunakan model tunggal, penelitian ini mengkaji performa dua arsitektur *deep learning* yang berbeda, yaitu MLP dan CNN-1D (Olanrewaju-George & Pranggono, 2025; Thantharate & T, 2023). Hal ini dilakukan untuk menganalisis *trade-off* antara efisiensi komputasi pada *gateway* dengan akurasi deteksi pola serangan yang kompleks.

Selain itu, penelitian ini mensimulasikan kondisi heterogenitas data yang ekstrem menggunakan distribusi Dirichlet $\alpha = 0.3$, guna merepresentasikan variasi trafik pada node jaringan *smart city* yang nyata secara lebih akurat dibandingkan pembagian data acak sederhana. Melalui penerapan skenario klasifikasi biner dan multi-kelas pada dataset ToN-IoT, penelitian ini bertujuan memberikan rekomendasi praktis mengenai pemilihan algoritma dan arsitektur model bagi implementasi NIDS terdistribusi yang menjaga privasi di lingkungan perkotaan yang komplek

BAB III
METODOLOGI PENELITIAN

Penelitian ini mengadopsi kerangka *CRISP-DM* (Cross-Industry Standard Process for Data Mining) sebagai fondasi metodologi, dengan penyesuaian pada konteks *Federated Learning* untuk *Network Intrusion Detection System* (NIDS) di lingkungan *smart city*. Pemilihan *CRISP-DM* didasarkan pada kemampuannya dalam menyediakan alur kerja yang sistematis dan iteratif, sekaligus fleksibel untuk diterapkan pada skenario pembelajaran mesin terdistribusi.

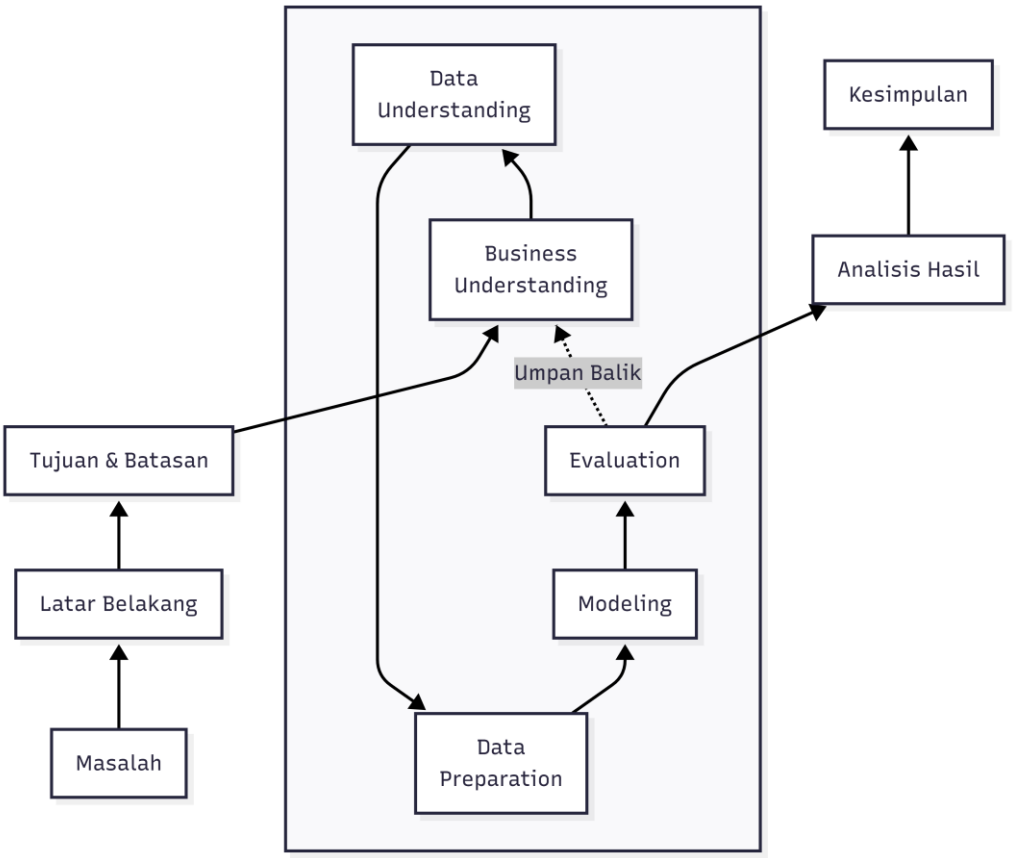
3.1 Alur Pengerjaan

Berbeda dengan pendekatan *data mining* terpusat, penelitian ini dirancang untuk mempertahankan desentralisasi data guna menjamin privasi lalu lintas jaringan pada level *gateway*. Oleh karena itu, setiap tahapan *CRISP-DM* diadaptasi agar selaras dengan skema *Horizontal Federated Learning*, di mana data tetap berada pada *client* lokal dan hanya parameter model yang dipertukarkan dengan server pusat.

Secara umum, alur penelitian diawali dengan pemetaan kebutuhan keamanan dan privasi *smart city* (*Business Understanding*), dilanjutkan dengan verifikasi karakteristik statistik dan kualitas dataset ToN-IoT (*Data Understanding*). Data kemudian ditransformasikan agar sesuai dengan kebutuhan pelatihan model terdistribusi (*Data Preparation*), sebelum masuk ke tahap perancangan arsitektur model dan mekanisme agregasi *Federated Learning* (*Modeling*). Tahap akhir mencakup evaluasi performa model global pada berbagai skenario distribusi data, baik IID maupun Non-IID (*Evaluation*).

Untuk memastikan reprodusibilitas dan konsistensi hasil eksperimen, seluruh proses penelitian diimplementasikan dalam lingkungan komputasi yang terkontrol, mencakup spesifikasi perangkat keras, perangkat lunak, serta pustaka pendukung yang digunakan selama simulasi *Federated Learning*.

Alur metodologi penelitian secara keseluruhan dirangkum pada Gambar 3.1 dalam bentuk diagram alur kerja.



Gambar 3.1 Alur kerja penelitian

Gambar 3.1 menunjukkan alur kerja penelitian berbasis CRISP-DM yang diadaptasi untuk Horizontal *Federated Learning*, dimulai dari perumusan masalah dan tujuan penelitian, dilanjutkan dengan tahapan pemahaman bisnis, pemahaman data, persiapan data, pemodelan, dan evaluasi. Tahap evaluasi memungkinkan terjadinya iterasi kembali ke pemahaman bisnis apabila hasil yang diperoleh belum memenuhi tujuan penelitian, sebelum diakhiri dengan analisis hasil dan kesimpulan.

3.2 Lingkungan Penelitian

Lingkungan penelitian pada studi ini dirancang untuk mendukung seluruh tahapan alur kerja yang telah dijelaskan pada Subbab 3.1 dengan fokus pada simulasi *Horizontal Federated Learning* serta pelatihan model *machine learning*. Seluruh eksperimen dilaksanakan pada satu mesin komputasi lokal dengan konfigurasi perangkat keras dan perangkat lunak yang didefinisikan secara eksplisit untuk menjamin konsistensi pelaksanaan eksperimen dan kemudahan replikasi hasil penelitian. Pengelolaan dependensi dilakukan menggunakan *Python virtual environment* tanpa memanfaatkan Conda, sementara akselerasi komputasi model *deep*

learning didukung oleh GPU melalui CUDA pada TensorFlow. Rincian lingkungan perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini masing-masing disajikan pada Tabel 3.1 *Local Host Hardware Enviroment*Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 *Local Host Hardware Enviroment*

Komponen	**Spesifikasi**
Motherboard	Gigabyte B450 Aorus M (6C/6T)
Processor	AMD Ryzen 5 3500 (*up to* 4.1 GHz) 6 core 6 threads
Memory	2×8 GB Single Rank + 2×16 GB *Dual Rank* DDR4 3200 MHz CL18 (48 GB total)
Storage	Adata SU650 240 GB SATA III
GPU	Zotac GTX 1650 Super
Operating System	Arch Linux (*rolling release*)
Kernel	Linux 6.17.9-zen (*freezed*)
Driver NVIDIA	580.105.08
CUDA Version	13.0

Tabel 3.2 Perangkat Lunak dan Pustaka Pendukung

Kategori	**Library**	**Versi**	**Fungsi**
Data Processing	pandas	2.3.3	Manipulasi dan analisis data
	numpy	2.3.5	Operasi numerik
	scipy	1.16.3	Komputasi ilmiah
	scikit-learn	1.7.2	Pra-pemrosesan dan evaluasi model
Deep learning	tensorflow	2.20.0	Framework deep learning berbasis CUDA
	keras	3.12.0	Abstraksi arsitektur model
Federated Learning	flwr	1.24.0	Orkestrasi dan agregasi FL
	grpcio	1.76.0	Komunikasi *client* ke *server*
	grpcio-health-checking	1.76.0	Monitoring koneksi FL
Imbalance Handler	imbalanced-learn	0.14.0	Teknik penanganan ketidakseimbangan kelas

GPU Accelerated for Pre-processing (opsional)	cupy	13.6.0	Operasi numerik berbasis CUDA
Visualization and EDA	matplotlib	3.10.7	Visualisasi data
	seaborn	0.13.2	Visualisasi statistik
UI/UX *Enviroment*	jupyter	1.0.0	*Notebook interface*
	jupyterlab	4.5.0	Pengembangan interaktif
	notebook	7.5.0	Eksekusi notebook
	ipykernel	7.1.0	Kernel Python
	ipython	9.7.0	*Interactive shell*
	ipywidgets	8.1.8	Komponen interaktif
Utils	pickle	built-in	Serialisasi objek
	pathlib	built-in	Operasi berkas
	rich	13.9.4	*Logging* dan *output* terminal
	tabulate	0.9.0	Penyajian tabel
Distributed computing	ray	2.51.1	Eksekusi terdistribusi

3.3 *Businesses Understanding*

Dalam konteks penelitian ini, lingkungan *smart city* dipandang sebagai sistem jaringan terdistribusi yang terdiri dari berbagai *gateway* IoT dengan karakteristik lalu lintas yang heterogen. Masing-masing *gateway* tersebut berperan sebagai titik akses utama bagi layanan publik penting, seperti transportasi cerdas, manajemen energi, dan sistem kesehatan. Gangguan pada lapisan jaringan, khususnya akibat serangan siber, berpotensi berdampak langsung terhadap kontinuitas layanan dan keandalan operasional kota.

Permasalahan keamanan jaringan pada *smart city* tidak hanya berkaitan dengan kemampuan mendeteksi serangan, tetapi juga dengan cara pengelolaan data lalu lintas jaringan. Pendekatan keamanan konvensional yang mengandalkan pemusatan data pada server terpusat menghadapi kendala serius terkait risiko kebocoran data dan kepatuhan terhadap regulasi privasi. Lalu lintas jaringan yang dianalisis sering kali memuat informasi sensitif yang merepresentasikan aktivitas pengguna dan infrastruktur kota.

Kondisi tersebut mendorong kebutuhan akan mekanisme deteksi intrusi yang mampu beroperasi secara terdistribusi pada tingkat *gateway*, tanpa memindahkan data mentah ke pusat dan tetap mempertahankan kemampuan deteksi yang andal pada skala kota. Kerangka inilah yang menjadi dasar dalam perumusan tujuan bisnis dan kriteria keberhasilan sistem pada tahap *business understanding*.

3.3.1 Tujuan Bisnis (*Business Objectives*)

Berdasarkan konteks permasalahan tersebut, tujuan bisnis dari penelitian ini adalah merancang mekanisme pertahanan siber yang mampu menyeimbangkan kebutuhan keamanan operasional dan perlindungan privasi data dalam lingkungan *smart city*. Sistem yang diusulkan diharapkan dapat memenuhi tujuan-tujuan berikut:

a. Menjamin Kontinuitas Layanan Publik

   Sistem harus mampu mendeteksi serangan jaringan, seperti Distributed Denial of Service, Injection, dan sebagainya secara dini pada tingkat *gateway* untuk mencegah gangguan terhadap layanan publik yang bersifat kritis.

b. Menjaga Privasi Data Lalu Lintas Jaringan

   Sistem harus memastikan bahwa data mentah lalu lintas jaringan tetap berada di lingkungan lokal masing-masing *gateway*, sehingga risiko kebocoran data dan pelanggaran privasi dapat diminimalkan.

c. Mendukung Operasi Terdistribusi Skala Kota

   Mekanisme yang dirancang harus memungkinkan pembaruan dan peningkatan kemampuan deteksi dilakukan secara terkoordinasi tanpa memerlukan sentralisasi data.

3.3.2 Kriteria Keberhasilan (*Success Criteria*)

Keberhasilan sistem yang diusulkan dievaluasi berdasarkan indikator teknis yang memiliki implikasi langsung terhadap operasional *smart city*, sebagai berikut:

a. Efektivitas Deteksi pada Data Heterogen (Non-IID)

   Model deteksi intrusi harus mampu mempertahankan performa yang konsisten meskipun setiap *gateway* memiliki distribusi lalu lintas jaringan yang berbeda. Ketidakmampuan menangani heterogenitas data berpotensi meningkatkan tingkat *false alarm* dan menurunkan keandalan system (Zhou dkk., 2025).

b. Stabilitas Pelatihan dalam Lingkungan Terdistribusi

Proses pelatihan menggunakan *Federated Learning* harus menghasilkan konvergensi model global yang stabil meskipun pelatihan dilakukan secara terfragmentasi pada banyak *client* (Li dkk., 2020).

c. Ketepatan Waktu Proses Pelatihan Federated

Ketepatan waktu pelatihan dievaluasi melalui pengukuran waktu pelatihan (training latency), yang merepresentasikan durasi yang dibutuhkan untuk menyelesaikan satu ronde pelatihan federated. Pengukuran ini mencakup waktu pelatihan lokal di sisi *client* serta proses agregasi model di server pusat. Indikator ini digunakan untuk menilai kelayakan penerapan sistem *Federated Learning* pada lingkungan smart city yang menuntut respons sistem dalam rentang waktu tertentu (McMahan dkk., 2017).

Indikator-indikator tersebut digunakan sebagai dasar dalam perancangan eksperimen dan evaluasi performa sistem pada tahap selanjutnya.

3.4 Data Understanding

Tahap *data understanding* dalam kerangka *CRISP-DM* bertujuan untuk memastikan bahwa dataset yang digunakan memiliki kualitas, karakteristik statistik, dan struktur distribusi yang sesuai dengan tujuan penelitian sebelum dilakukan pemrosesan lebih lanjut. Dataset ToN-IoT yang digunakan dalam penelitian ini telah dijelaskan secara komprehensif pada Bab II Sub Bab 2.3, mencakup arsitektur *testbed*, jenis serangan, serta deskripsi fitur. Oleh karena itu, tahap ini tidak mengulang pemaparan struktur data, melainkan berfokus pada evaluasi karakteristik data dari perspektif eksperimental.

Secara khusus, tahap *data understanding* diarahkan untuk memverifikasi kesesuaian dataset ToN-IoT dalam mensimulasikan skenario *Federated Learning* (FL) pada lingkungan *smart city*, di mana data bersifat terdistribusi, heterogen, dan tidak seimbang. Analisis eksploratori dilakukan untuk mengidentifikasi potensi permasalahan data yang dapat memengaruhi stabilitas pelatihan model dan validitas hasil eksperimen. Investigasi data difokuskan pada tiga aspek utama berikut.

3.4.1 Analisis Distribusi Kelas

Analisis distribusi kelas dilakukan untuk memetakan proporsi masing-masing kategori serangan dan lalu lintas normal dalam dataset. Dataset ToN-IoT secara inheren memiliki karakteristik *class imbalance*, di mana beberapa jenis serangan muncul dalam jumlah yang sangat terbatas dibandingkan kelas lainnya. Kondisi ini merepresentasikan situasi nyata pada

jaringan *smart city*, di mana sebagian besar waktu jaringan berada dalam kondisi normal dan hanya sebagian kecil serangan tertentu yang terjadi secara sporadis.

Pemahaman terhadap distribusi kelas ini menjadi faktor kunci dalam perancangan eksperimen *Federated Learning*. Informasi mengenai ketimpangan kelas digunakan sebagai dasar untuk merancang skema pembagian data ke setiap *client* agar mencerminkan kondisi Non-IID, di mana setiap *gateway* memiliki pola lalu lintas dan profil ancaman yang berbeda. Tanpa analisis ini, simulasi *Federated Learning* berisiko menghasilkan model yang bias terhadap kelas mayoritas atau tidak stabil pada kelas minoritas.

3.4.2 Analisis Integritas dan Kualitas Data

Analisis integritas data dilakukan untuk memastikan bahwa dataset bebas dari permasalahan mendasar yang dapat menurunkan kualitas proses pelatihan model. Pemeriksaan difokuskan pada dua aspek utama, yaitu keberadaan *missing values* dan duplikasi data.

Pemeriksaan *missing values* dilakukan untuk mengidentifikasi atribut yang tidak memiliki nilai, khususnya pada fitur-fitur yang bergantung pada protokol tertentu sebagaimana dijelaskan pada Subbab 2.3. Ketidakhadiran nilai pada fitur-fitur tersebut dianalisis secara kontekstual untuk membedakan antara kondisi yang muncul secara alami akibat ketiadaan protokol pada suatu koneksi dan indikasi kesalahan pencatatan data. Pendekatan ini penting agar penanganan data hilang tidak menghilangkan informasi yang secara semantik masih valid.

Selain itu, analisis duplikasi dilakukan untuk mendeteksi adanya pengulangan baris data yang identik. Keberadaan data duplikat berpotensi menyebabkan bias pada model karena pola yang sama dipelajari secara berulang. Oleh karena itu, identifikasi redundansi data menjadi langkah awal untuk menjaga objektivitas dan generalisasi model yang dilatih.

3.4.3 Analisis Korelasi Fitur

Analisis korelasi fitur dilakukan untuk mengevaluasi hubungan antar atribut numerik dalam dataset. Matriks korelasi membantu untuk melihat apakah ada data yang kembar atau saling tumpang tindih. Keberadaan fitur yang sangat berkorelasi dapat meningkatkan kompleksitas model tanpa memberikan informasi tambahan yang signifikan.

Hasil analisis korelasi ini digunakan sebagai landasan dalam tahap seleksi dan transformasi fitur pada fase Data Preparation. Dengan mengurangi redundansi fitur, model *Federated Learning* yang dibangun diharapkan lebih efisien dari sisi komputasi serta mengurangi beban komunikasi antar *client* dan server agregasi.

Melalui tahap *data understanding* ini, dataset ToN-IoT divalidasi kelayakannya sebagai representasi lalu lintas jaringan *smart city* yang kompleks dan terdistribusi. Analisis distribusi kelas, integritas data, dan korelasi fitur memberikan dasar empiris bagi perancangan strategi pemrosesan data dan pembagian data antar *client* pada tahap *data preparation*, sehingga eksperimen *Federated Learning* yang dilakukan memiliki fondasi data yang kuat dan realistis.

3.5 Data Preparation

Tahap *data preparation* bertujuan untuk mentransformasikan data mentah menjadi representasi numerik yang konsisten, efisien, dan sesuai untuk pelatihan model *Federated Learning* berbasis jaringan saraf tiruan. Mengingat model *deep learning* sangat sensitif terhadap skala, format, dan distribusi input, proses persiapan data dirancang secara sistematis dengan mempertimbangkan karakteristik spesifik dataset ToN-IoT serta keterbatasan lingkungan terdistribusi pada skenario *smart city*.

Berbeda dengan pendekatan generik, tahapan persiapan data pada penelitian ini disusun untuk menjaga integritas semantik lalu lintas jaringan, mempertahankan karakteristik Non-IID, serta meminimalkan risiko bias akibat kebocoran data. Proses ini terdiri atas enam tahapan utama sebagai berikut.

3.5.1 Pembersihan Data

Pembersihan data dilakukan dengan pendekatan *protocol-aware* yang kontekstual. Missing values pada fitur spesifik protokol *DNS*, *HTTP*, *SSL* diisi berdasarkan keberadaan trafik protokol terkait, misalnya dengan mode, 0, atau *none*. Kolom numerik lainnya diisi dengan median, dan kolom kategorikal dengan mode atau *unknown*. Pendekatan ini tetap mempertahankan makna semantik sebanyak mungkin sambil memastikan tidak ada nilai NaN yang mengganggu proses pelatihan model (Alsaedi dkk., 2020).

Ketiadaan nilai pada fitur-fitur tersebut tidak merepresentasikan kesalahan pencatatan data, melainkan menunjukkan bahwa suatu koneksi tidak menggunakan protokol terkait. Oleh karena itu, nilai kosong dipertahankan dan diperlakukan sebagai informasi implisit mengenai absennya protokol tertentu. Pendekatan ini memungkinkan model mempelajari perbedaan karakteristik lalu lintas berdasarkan jenis protokol yang aktif tanpa kehilangan makna semantik data (Alsaedi dkk., 2020).

3.5.2 Transformasi Variabel Kategorikal (*Encoding*)

Untuk mendukung komputasi berbasis matriks pada model *deep learning*, seluruh atribut non-numerik ditransformasikan ke dalam representasi numerik yang sesuai (Alsaedi dkk., 2020).

Variabel target untuk klasifikasi *multiclass*, yang mencakup sepuluh kategori lalu lintas (sembilan jenis serangan dan satu kelas normal), dikonversi menggunakan *label encoding*. Pendekatan ini dipilih karena label hanya berfungsi sebagai penanda kelas dan tidak mengandung hubungan ordinal (Alsaedi dkk., 2020).

Sementara itu, fitur kategorikal nominal seperti jenis layanan, status koneksi, dan atribut protokol ditransformasikan menggunakan teknik *one-hot encoding*. Metode ini digunakan untuk mencegah model mengasumsikan adanya hubungan hierarkis yang tidak relevan antar kategori, sehingga pola lalu lintas jaringan dapat dipelajari secara lebih representatif (Alsaedi dkk., 2020).

3.5.3 Seleksi Fitur Berbasis Variansi

Untuk meningkatkan efisiensi komputasi dan mengurangi kompleksitas model, dilakukan seleksi fitur menggunakan metode *variance thresholding*. Fitur-fitur dengan variansi sangat rendah, yaitu di bawah ambang batas 0.01, dieliminasi dari dataset (Alsaedi dkk., 2020).

Fitur dengan variansi mendekati nol cenderung memiliki nilai yang hampir konstan pada seluruh sampel, sehingga tidak memberikan kontribusi signifikan terhadap kemampuan model dalam membedakan antara lalu lintas normal dan aktivitas serangan. Penghapusan fitur-fitur ini membantu mengurangi risiko *overfitting* serta menurunkan beban komputasi dan komunikasi dalam skema *Federated Learning (Li dkk., 2020)* .

3.5.4 Standardisasi Fitur (Feature Scaling)

Dataset ToN-IoT memiliki fitur numerik dengan rentang nilai yang sangat bervariasi, seperti durasi koneksi dalam satuan detik dan volume lalu lintas dalam satuan *byte* yang dapat mencapai jutaan. Untuk memastikan setiap fitur memberikan kontribusi yang seimbang selama proses pelatihan, dilakukan standardisasi menggunakan teknik *standard scaling (Alsaedi dkk., 2020)*.

Teknik ini mentransformasi fitur numerik sehingga memiliki nilai rata-rata nol dan deviasi standar satu. Proses fitting scaler dilakukan secara eksklusif pada data latih dan kemudian diterapkan pada data uji. Pemisahan ini dilakukan secara ketat untuk mencegah terjadinya

kebocoran data (*data leakage*) yang dapat menyebabkan hasil evaluasi menjadi bias dan terlalu optimis (Alsaedi dkk., 2020).

3.5.5 Pembagian Data Terstratifikasi

Data yang telah diproses dibagi menjadi himpunan pelatihan dan pengujian dengan rasio 80% untuk data latih dan 20% untuk data uji. Pembagian dilakukan menggunakan metode stratified splitting untuk memastikan proporsi setiap kelas serangan pada kedua himpunan tetap konsisten dengan distribusi data asli (Alsaedi dkk., 2020).

Pendekatan ini sangat penting mengingat dataset ToN-IoT memiliki tingkat ketimpangan kelas yang tinggi. Dengan menjaga konsistensi distribusi kelas, evaluasi performa model terhadap kelas minoritas, seperti serangan *Man-in-the-Middle* (MITM), dapat dilakukan secara lebih adil dan representative (Alsaedi dkk., 2020).

3.5.6 Pembersihan Data

Penelitian ini secara sengaja tidak menerapkan teknik oversampling sintetis seperti *SMOTE* (Synthetic Minority Over-sampling Technique) yang umum digunakan pada sistem deteksi intrusi konvensional. Keputusan metodologis ini didasarkan pada pertimbangan validitas eksperimen *Federated Learning*.

Pertama, penerapan *SMOTE* akan mengubah distribusi alami data, sehingga mengaburkan karakteristik Non-IID yang menjadi fokus utama penelitian ini. Kedua, pada lingkungan terdistribusi, pembuatan data sintetis secara independen di setiap *client* berpotensi menghasilkan pola pelatihan yang tidak konsisten dan menghambat proses konvergensi model global. Ketiga, penanganan ketimpangan kelas diserahkan pada mekanisme algoritmik *Federated Learning*, khususnya melalui penggunaan algoritma FedProx yang memiliki *proximal regularization* term untuk menjaga stabilitas pelatihan meskipun terdapat dominasi kelas tertentu pada masing-masing *client* (Li dkk., 2020).

3.6 Modeling

Tahap Modeling merupakan inti dari penelitian ini, di mana arsitektur jaringan saraf tiruan dan algoritma agregasi *Federated Learning* dikonfigurasi untuk membangun sistem deteksi intrusi pada lingkungan *smart city* yang terdistribusi. Pada tahap ini, fokus diarahkan pada pemilihan arsitektur model, skema pelatihan federated, strategi penanganan heterogenitas data, serta konfigurasi *hyperparameter* yang sesuai dengan keterbatasan sumber daya *gateway* IoT.

Implementasi pemodelan dilakukan menggunakan kerangka kerja simulasi *Federated Learning* berbasis Python dengan Flower framework, yang banyak digunakan dalam penelitian akademik untuk mengevaluasi algoritma FL pada skenario terkontrol.

3.6.1 Strategi *Horizontal Federated Learning*

Penelitian ini menerapkan skema *Horizontal Federated Learning* (HFL), di mana setiap *client* memiliki ruang fitur yang identik namun distribusi sampel yang berbeda. Skema ini dipilih karena secara realistis merepresentasikan kondisi *smart city*, di mana setiap *gateway* jaringan mengamati jenis protokol yang sama tetapi dengan pola lalu lintas lokal yang berbeda.

Dalam HFL, pelatihan model dilakukan secara lokal pada masing-masing *client*, sementara hanya parameter model yang dikirimkan ke server pusat untuk proses agregasi. Pendekatan ini terbukti efektif dalam menjaga privasi data mentah sekaligus memungkinkan pembelajaran kolaboratif lintas node terdistribusi, sebagaimana ditunjukkan dalam berbagai studi Federated Learning pada sistem IoT dan keamanan jaringan (McMahan dkk., 2017; Li dkk., 2020).

3.6.2 Arsitektur *Deep Learning*

Berdasarkan pertimbangan keseimbangan antara kompleksitas komputasi dan kemampuan generalisasi pada lingkungan *gateway* IoT yang memiliki keterbatasan sumber daya, penelitian ini mengevaluasi dua arsitektur utama, yaitu *multi-layer perceptron* (MLP) sebagai model *baseline* yang ringan dan *convolutional neural network 1-dimension* (CNN-1D) yang memiliki keunggulan dalam ekstraksi fitur spasial. Kedua arsitektur tersebut diadaptasi ke dalam dua skenario klasifikasi dengan tujuan operasional yang berbeda, di mana klasifikasi biner diterapkan untuk memenuhi kebutuhan deteksi intrusi tahap awal yang cepat dalam membedakan trafik normal dan serangan, sedangkan klasifikasi *multiclass* digunakan untuk keperluan analisis forensik dan mitigasi spesifik melalui identifikasi jenis serangan secara mendetail (Devi dkk., 2025).

Multi-Layer Perceptron (MLP)

MLP digunakan sebagai model dasar (*baseline*) untuk mempelajari hubungan non-linear antar fitur statistik jaringan. Arsitektur ini banyak digunakan dalam penelitian NIDS karena kesederhanaannya serta efisiensi komputasi pada lingkungan dengan keterbatasan sumber daya (Ferrag dkk., 2020; Kunang dkk., 2021a; Qazi dkk., 2022a). Untuk strategi sebagai berikut:

a. Konfigurasi *Binary*:

Arsitektur terdiri dari lima lapisan tersembunyi (Input → 256 → 128 → 64 → 32 → Output). Lapisan keluaran menggunakan fungsi aktivasi sigmoid untuk memodelkan probabilitas serangan (Kunang dkk., 2021b; Qazi dkk., 2022b).

b. Konfigurasi *Multiclass*:

Untuk menangkap kompleksitas pola serangan yang lebih beragam, digunakan arsitektur yang lebih dalam (Input → 512 → 256 → 128 → 64 → Output), dengan fungsi aktivasi *softmax* pada lapisan keluaran untuk memprediksi sepuluh kelas lalu lintas (Kunang dkk., 2021b).

c. Regularisasi:

Untuk mengurangi risiko *overfitting*, diterapkan dropout dengan rasio 0.2–0.4 serta regularisasi L2 dengan koefisien $\lambda = 0.001$. Kombinasi ini terbukti efektif dalam meningkatkan generalisasi model pada data jaringan yang kompleks dan tidak seimbang (Ferrag dkk., 2020; Kunang dkk., 2021b).

***Convolutional Neural Network 1D* (CNN-1D)**

CNN-1D digunakan untuk mengekstraksi pola lokal dari vektor fitur lalu lintas jaringan. Pendekatan ini telah banyak digunakan dalam literatur NIDS karena kemampuannya menangkap dependensi lokal antar fitur tanpa memerlukan rekonstruksi data menjadi format citra salah satunya dari (Ferrag dkk., 2020). Untuk stateginya sebagai berikut:

a. Struktur Ekstraksi Fitur:

Model terdiri dari tiga lapisan konvolusi bertingkat. Untuk klasifikasi *binary* digunakan konfigurasi filter (64 → 32 → 16), sedangkan untuk klasifikasi *multiclass* digunakan konfigurasi yang lebih lebar (128 → 64 → 32). Setiap lapisan konvolusi diikuti oleh *batch normalization* untuk mempercepat konvergensi dan meningkatkan stabilitas pelatihan (Ferrag dkk., 2020).

b. Klasifikasi:

Fitur hasil konvolusi diratakan (*flatten*) dan diproses melalui dua lapisan *fully connected* (128 → 64) sebelum masuk ke lapisan keluaran (Ferrag dkk., 2020).

Arsitektur *CNN-1D* dipilih karena telah terbukti memberikan performa yang kompetitif dalam deteksi intrusi berbasis trafik jaringan, khususnya pada dataset dengan dimensi fitur tinggi seperti ToN-IoT (Moustafa, 2021).

3.6.3 Strategi Distribusi Data Antar *Client* (Non-IID Simulation)

Dalam *Federated Learning*, performa model global sangat dipengaruhi oleh karakteristik distribusi data pada masing-masing *client* (Li dkk., 2020). Untuk menyimulasikan kondisi heterogenitas data yang merepresentasikan lingkungan *smart city*, penelitian ini menerapkan Distribusi *dirichlet* sebagai mekanisme pembagian data antar *client* (Zhu dkk., 2021).

Distribusi *dirichlet* digunakan untuk mengatur proporsi kelas serangan yang dimiliki oleh setiap *client* tanpa mengubah struktur fitur dataset. Pendekatan ini memungkinkan simulasi berbagai tingkat ketidakseragaman data (Non-IID) secara terkontrol melalui parameter konsentrasi (α). Nilai α yang kecil menghasilkan distribusi data yang sangat timpang, di mana *client* cenderung didominasi oleh kelas tertentu, sedangkan nilai α yang lebih besar menghasilkan distribusi yang lebih mendekati kondisi IID (Zhu dkk., 2021).

Dalam konteks penelitian ini, penggunaan distribusi *dirichlet* dipilih karena dua alasan utama. Pertama, pendekatan ini banyak digunakan dalam penelitian *Federated Learning* untuk mengevaluasi ketahanan algoritma agregasi terhadap heterogenitas data (Li dkk., 2020). Kedua, Dirichlet mampu merepresentasikan kondisi realistis *smart city*, di mana setiap *gateway* mengamati pola lalu lintas jaringan yang berbeda sesuai dengan karakteristik wilayah operasionalnya (Al-Huthaifi dkk., 2023).

Simulasi lingkungan *smart city* dilakukan dengan membagi dataset ke dalam lima *client* menggunakan distribusi Dirichlet. Pemilihan jumlah lima *client* ini didasarkan pada representasi sektoral fungsional dalam ekosistem *smart city* yang mencakup area residensial, transportasi publik, fasilitas kesehatan, area industri, dan layanan pemerintahan. Namun demikian, karakteristik *client* 1 hingga 5 dalam simulasi ini masih bersifat global dan tergeneralisasi (*generalized*). Artinya, pembagian data dilakukan secara statistik berdasarkan distribusi Dirichlet tanpa pelabelan spesifik terhadap sektor tertentu; sebagai contoh, *client* 1 tidak secara eksplisit ditetapkan sebagai fasilitas kesehatan maupun *client* 2 sebagai area residensial.

Pendekatan ini bertujuan untuk menjaga kesederhanaan simulasi serta memfokuskan analisis pada efek heterogenitas data secara umum, sedangkan representasi sektoral hanya digunakan sebagai landasan konseptual dalam penentuan jumlah *client*. Meskipun pelabelan spesifik dapat ditambahkan pada eksperimen lanjutan untuk meningkatkan realisme, namun pada tahap ini penelitian tetap berfokus pada distribusi data tanpa diferensiasi tematik. Di sisi lain, jumlah ini juga disesuaikan dengan batasan sumber daya perangkat keras (*computing hardware resources*) pada lingkungan simulasi guna memastikan efisiensi eksekusi tanpa

mengorbankan validitas pengujian terhadap variasi pola trafik. Selanjutnya, parameter konsentrasi α digunakan sebagai pengendali tingkat heterogenitas data antar-*client* dengan konfigurasi sebagai berikut:

a. Non-IID Ekstrem ($\alpha = 0.3$)

Nilai α yang kecil (0.3) dipilih untuk mensimulasikan kondisi Non-IID atau heterogenitas ekstrem. Pemilihan ini didasarkan pada karakteristik perangkat IoT di dunia nyata yang secara alami memiliki profil risiko berbeda misalnya, trafik CCTV didominasi video dan rentan DDoS, sedangkan smart meter didominasi data teks. Meskipun dalam simulasi ini client tidak dilabeli secara spesifik sebagai perangkat tertentu, penggunaan $\alpha = 0.3$ secara statistik memaksa distribusi data menjadi sangat timpang guna menguji ketangguhan model global dalam belajar dari client yang memiliki bias kelas ekstrem akibat spesialisasi fungsi perangkat di lapangan.

b. IID-like ($\alpha = 5.0$)

Nilai α yang lebih besar (5.0) dipilih sebagai baseline komparatif. Pada kondisi ini, distribusi data mendekati seragam (IID), di mana setiap client diasumsikan memiliki sampel yang mewakili seluruh populasi data global secara proporsional. Skenario ini merepresentasikan kondisi ideal teoretis atau situasi di mana gateway melakukan load balancing sempurna, sehingga variasi serangan tersebar merata untuk mengukur seberapa jauh degradasi performa terjadi saat dihadapkan pada realitas statistik data yang terfragmentasi.

Sebagaimana dijelaskan pada anak subbab 2.5.3 distribusi Dirichlet banyak digunakan dalam literatur *Federated Learning* untuk memodelkan dan menganalisis kondisi *Non-IID* secara sistematis. Pendekatan ini memungkinkan pengaturan tingkat heterogenitas data melalui parameter konsentrasi α, di mana nilai α yang lebih kecil merepresentasikan heterogenitas data yang lebih tinggi, sedangkan nilai yang lebih besar mendekati kondisi IID, dengan interpretasi yang bergantung pada jumlah *client* dan kelas dataset (Cha & Chang, 2025; Zhou dkk., 2025; Zhu dkk., 2021). Fleksibilitas tersebut menjadikan distribusi Dirichlet sebagai landasan teoretis yang relevan dalam mengevaluasi ketahanan dan stabilitas algoritma *Federated Learning* pada skenario distribusi data yang menyerupai kondisi nyata *smart city*.

Simulasi *Non-IID* yang dihasilkan melalui distribusi Dirichlet ini digunakan sebagai dasar untuk mengevaluasi perbedaan performa antara algoritma *Federated Averaging* (FedAvg) dan *Federated Proximal* (FedProx), khususnya dalam hal stabilitas konvergensi dan kemampuan generalisasi model global pada tingkat heterogenitas data yang berbeda.

3.6.4 Optimasi *Hyperparameter* dan Reprodusibilitas

Optimasi *hyperparameter* pada penelitian ini dilakukan menggunakan pendekatan *limited Grid Search* untuk menyeimbangkan antara performa model dan efisiensi komputasi. Strategi ini dipilih karena pelatihan *Federated Learning* melibatkan biaya komunikasi dan komputasi yang lebih tinggi dibandingkan pelatihan terpusat. Oleh karena itu, ruang pencarian *hyperparameter* dibatasi berdasarkan praktik umum dan rekomendasi pada literatur *Federated Learnin*g dan *Network Intrusion Detection System* (NIDS).

Kombinasi *hyperparameter* yang dievaluasi meliputi:

a. Local Epochs: 1 dan 2

   Jumlah *epoch* pelatihan lokal dibatasi pada nilai kecil untuk mengurangi risiko *client drift*, khususnya pada skenario distribusi data Non-IID. Pelatihan lokal yang terlalu lama dapat menyebabkan pembaruan model *client* menyimpang signifikan dari model global, sehingga menghambat konvergensi *federated* (McMahan dkk., 2017; Li dkk., 2020).

b. Batch Size: 256, 512, dan 1024

   Variasi ukuran *batch* digunakan untuk mengevaluasi *trade-off* antara stabilitas gradien dan efisiensi komputasi pada dataset lalu lintas jaringan berskala besar. Ukuran *batch* yang lebih besar cenderung menghasilkan gradien yang lebih stabil, namun meningkatkan kebutuhan memori dan waktu komputasi pada perangkat *gateway* (Tian dkk., 2021).

c. Learning Rate: 0.001 dan 0.0005

   Rentang learning rate dipilih berdasarkan praktik umum pada pelatihan model deep learning menggunakan *Adam optimizer*. Nilai ini bertujuan menjaga proses konvergensi yang stabil serta menghindari osilasi atau divergensi selama pelatihan terdistribusi (Ferrag dkk., 2020).

d. Nilai (μ) : 0.01 dan 0.001 pada FedProx

   Evaluasi parameter proksimal (μ) dilakukan pada tahap lanjutan setelah konfigurasi terbaik FedAvg diperoleh melalui *Grid Search*. Nilai μ yang lebih kecil memberikan fleksibilitas pada pembaruan model lokal, sedangkan nilai yang lebih besar memperkuat regularisasi untuk menekan dampak heterogenitas data antar *client*. Pendekatan ini memungkinkan analisis terfokus terhadap pengaruh mekanisme FedProx tanpa memperluas ruang pencarian *hyperparameter* secara berlebihan (Li dkk., 2020).

e. Partisipasi *Client* (*fraction_fit*): 1.0

   Parameter *fraction_fit* ditetapkan sebesar 1.0, yang mengindikasikan bahwa seluruh client (100%) berpartisipasi aktif dalam setiap ronde pelatihan. Secara ilmiah, keputusan

ini diambil untuk menjaga stabilitas model global dalam menghadapi kondisi Non-IID yang ekstrem pada simulasi dengan jumlah *node* terbatas. Partisipasi penuh memastikan bahwa agregasi model menangkap seluruh spektrum variasi distribusi data dari setiap sektor (*client*) di setiap iterasi, sehingga meminimalkan dampak *client drift* yang dapat terjadi jika hanya sebagian kecil *client* yang dipilih secara acak. Meskipun pada implementasi dunia nyata dengan ribuan perangkat nilai 1.0 mungkin tidak realistis karena kendala *bandwidth*, dalam konteks simulasi ini, nilai tersebut krusial untuk memvalidasi efektivitas algoritma pada data heterogen (Devi dkk., 2025; Farooq dkk., 2025).

Pemilihan konfigurasi hyperparameter optimal tidak didasarkan pada satu metrik tunggal, melainkan menggunakan pendekatan multi-kriteria, sebagaimana dijelaskan pada landasan teori Subbab 2.7.3. Setiap konfigurasi dievaluasi berdasarkan lima metrik utama, yaitu *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan *AUC-ROC*, dengan bobot yang sama sebesar 20% untuk masing-masing metrik. Pendekatan ini bertujuan memperoleh konfigurasi hyperparameter yang tidak hanya unggul pada satu metrik tertentu, tetapi juga stabil secara keseluruhan pada berbagai aspek kinerja klasifikasi.

Untuk memastikan keseimbangan performa antara tugas klasifikasi biner dan multi-kelas, digunakan *balance score* yang didasarkan pada nilai *weight score* WS. Skor terbobot ini dihitung dengan menjumlahkan kontribusi dari lima metrik evaluasi utama, yaitu *accuracy*, *precision*, *recall*, *f1-score*, dan *auc-roc*. Secara matematis, penentuan WS didefinisikan dalam Persamaan (3.1).

$$WS = \sum_{m \in M} (V_m \cdot W_m) \tag{3.1}$$

Di mana V_m mewakili nilai (*value*) dari metrik m, dan W_m adalah bobot yang diberikan untuk masing-masing metrik. Dalam penelitian ini dilakukan normalisasi bobot di mana total seluruh bobot harus memenuhi syarat $\sum W_m = 1$. Dengan menetapkan bobot yang setara sebesar 0.20 untuk setiap kriteria, formula tersebut dapat dijabarkan pada Persamaan (3.2).

$$WS = (Acc \cdot 0.2) + (Prec \cdot 0.2) + (Rec \cdot 0.2) + (F1 \cdot 0.2) + (AUC \cdot 0.2) \tag{3.2}$$

Pendekatan ini digunakan untuk menghindari pemilihan konfigurasi yang menunjukkan performa tinggi pada satu jenis tugas saja, tetapi mengalami degradasi signifikan pada tugas lainnya. Penentuan Konfigurasi Terbaik $C_{\{best\}}$ dilakukan dengan mencari nilai argumen maksimum dari skor terbobot pada ruang pencarian konfigurasi C pada Persamaan (3.3)

$$C_{best} = \arg\max_{c \in C}(WS_c) \tag{3.3}$$

Pendekatan ini digunakan untuk menghindari pemilihan konfigurasi yang menunjukkan performa tinggi pada satu jenis tugas, tetapi mengalami degradasi signifikan pada tugas lainnya. Dengan demikian, konfigurasi hyperparameter yang dipilih diharapkan mampu memberikan performa yang konsisten dan seimbang pada seluruh skenario evaluasi (Di Mauro dkk., 2021).

Evaluasi parameter proksimal μ pada algoritma FedProx dilakukan pada tahap lanjutan setelah konfigurasi terbaik FedAvg diperoleh melalui pendekatan multi-kriteria tersebut. Nilai μsebesar 0.01 dan 0.001 digunakan untuk menganalisis pengaruh regularisasi proksimal terhadap stabilitas konvergensi pada kondisi data heterogen, tanpa memperluas ruang pencarian hyperparameter secara berlebihan (Li dkk., 2020).

Untuk menjamin reprodusibilitas eksperimen, seluruh proses pelatihan dikendalikan dengan penetapan *random seed* 42 dan *Python hash seed* 42 yang konsisten pada seluruh pustaka utama yang digunakan, termasuk NumPy, TensorFlow, dan lingkungan sistem. Pendekatan ini memastikan bahwa variasi hasil eksperimen semata-mata disebabkan oleh perbedaan konfigurasi hyperparameter dan algoritma federated, bukan oleh faktor acak yang tidak terkontrol (Ferrag dkk., 2020).

3.7 *Evaluation*

Tahap evaluasi dalam penelitian ini bertujuan untuk menetapkan kerangka penilaian kinerja sistem deteksi intrusi berbasis *Federated Learning* yang diusulkan. Evaluasi dirancang untuk menilai dua aspek utama, yaitu kualitas deteksi serangan dan karakteristik operasional pelatihan terdistribusi, dengan mengacu pada landasan teori evaluasi yang telah dijabarkan pada Bab II Bagian 2.8. Kerangka evaluasi ini digunakan sebagai acuan dalam pelaporan dan analisis hasil eksperimen pada Bab IV.

3.7.1 Evaluasi Kinerja Klasifikasi

Evaluasi performa klasifikasi dilakukan menggunakan Confusion Matrix sebagai dasar pengukuran, sebagaimana dijelaskan pada Subbab 2.8.1. Untuk skenario klasifikasi *binary*, confusion matrix digunakan untuk merepresentasikan kemampuan model dalam membedakan antara lalu lintas normal dan lalu lintas serangan. Pada skenario klasifikasi *multiclass*, confusion matrix diperluas menjadi matriks berdimensi $C \times C$ untuk memetakan distribusi prediksi pada setiap kategori serangan. Berdasarkan confusion matrix pada kinerja model dievaluasi menggunakan metrik-metrik berikut:

a. *Accuracy*, untuk mengukur tingkat ketepatan prediksi secara keseluruhan.
b. *Precision*, untuk mengevaluasi keandalan model dalam menghasilkan peringatan serangan.
c. *Recall*, untuk mengukur kemampuan model dalam mendeteksi seluruh serangan yang terjadi.
d. *F1-Score*, sebagai ukuran keseimbangan antara *precision* dan *recall* pada kondisi ketimpangan kelas.
e. *AUC-ROC*, untuk menilai kemampuan model dalam membedakan kelas berdasarkan variasi nilai ambang keputusan (*decision threshold*) pada keluaran probabilistik model.

Seluruh metrik dihitung pada data uji yang terpisah dari data pelatihan guna menjaga objektivitas evaluasi.

3.7.2 Evaluasi pada Lingkungan Federated Learning

Selain performa klasifikasi, evaluasi juga difokuskan pada karakteristik pelatihan dalam lingkungan Federated Learning yang bersifat terdistribusi.

Evaluasi Distribusi Data Antar *Client*

Evaluasi dilakukan pada skenario distribusi data IID dan Non-IID. Kondisi Non-IID disimulasikan menggunakan pendekatan partisi data berbasis Distribusi Dirichlet, di mana parameter konsentrasi α digunakan untuk mengontrol tingkat heterogenitas data antar *client*. Pendekatan ini memungkinkan pengujian sistem pada berbagai tingkat ketimpangan distribusi data yang merepresentasikan kondisi nyata lingkungan *smart city*.

Evaluasi Stabilitas Pelatihan Terdistribusi

Stabilitas pelatihan dievaluasi dengan mengamati konsistensi pembaruan model global pada setiap ronde komunikasi. Evaluasi ini dilakukan untuk menilai kemampuan algoritma

Federated Learning dalam mempertahankan proses pelatihan yang stabil pada kondisi distribusi data yang tidak seragam.

Evaluasi Efisiensi Operasional

Efisiensi sistem *Federated Learning* dievaluasi dari aspek operasional, meliputi:

a. Waktu pelatihan per ronde komunikasi, yang mencerminkan beban komputasi dan agregasi model.
b. Jumlah ronde komunikasi, sebagai indikator efisiensi proses pembelajaran terdistribusi.

Evaluasi efisiensi ini bertujuan untuk memastikan bahwa sistem yang diusulkan tetap sesuai dengan keterbatasan sumber daya pada perangkat *gateway* IoT.

3.7.3 Keterkaitan Evaluasi dengan Tujuan Penelitian

Kerangka evaluasi yang diterapkan dalam penelitian ini dirancang untuk secara langsung mendukung pencapaian tujuan penelitian, khususnya dalam:

a. Menilai efektivitas *Federated Learning* untuk deteksi dan klasifikasi serangan jaringan.
b. Menganalisis dampak heterogenitas data antar *client* terhadap performa sistem.
c. Mengevaluasi kesesuaian pendekatan yang diusulkan untuk diterapkan pada lingkungan *smart city* yang bersifat terdistribusi dan sensitif terhadap privasi data.

BAB IV
HASIL dan Pembahasan

Bab ini menyajikan hasil eksperimen dan analisis kinerja sistem deteksi intrusi berbasis *Federated Learning* yang telah dirancang pada bagian BAB III Metodologi. Evaluasi dilakukan untuk menilai kemampuan model dalam mendeteksi dan mengklasifikasikan serangan jaringan pada lingkungan *smart city* yang bersifat terdistribusi dan heterogen.

Seluruh eksperimen dilaksanakan berdasarkan skenario pemodelan, pembagian data, serta konfigurasi *hyperparameter* yang telah dijelaskan pada tahap pemodelan dan persiapan data. Evaluasi difokuskan pada dua aspek utama, yaitu kualitas prediksi model deteksi intrusi dan stabilitas pelatihan dalam skema *Federated Learning*.

Kinerja sistem dianalisis menggunakan metrik evaluasi klasifikasi yang telah dibahas pada Bab II, termasuk *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan *ROC-AUC*, serta metrik operasional *Federated Learning* seperti stabilitas konvergensi dan sensitivitas terhadap heterogenitas data. Hasil yang diperoleh kemudian dibahas secara komparatif untuk menjawab rumusan masalah penelitian.

4.1 Hasil Data Understanding

Tahap *data understanding* dalam kerangka *CRISP-DM* bertujuan untuk memverifikasi kesesuaian dataset terhadap skenario eksperimen yang dirancang. Dataset ToN-IoT Network yang digunakan dalam penelitian ini telah dijelaskan secara rinci pada Bab II, meliputi arsitektur *testbed*, jenis serangan, dan struktur fitur. Oleh karena itu, pada tahap ini analisis difokuskan pada karakteristik data yang relevan secara langsung terhadap implementasi *Federated Learning* (FL) dalam lingkungan *smart city*.

Secara khusus, analisis data diarahkan untuk mengevaluasi distribusi kelas, kualitas data, serta karakteristik statistik fitur guna memastikan bahwa dataset mampu merepresentasikan kondisi data *heterogeneity* dan *class imbalance* yang lazim terjadi pada sistem jaringan terdistribusi.

4.1.1 Distribusi Kelas dan Implikasi Non-IID

Analisis distribusi kelas dilakukan untuk mengidentifikasi proporsi masing-masing kategori serangan dan lalu lintas normal dalam dataset. Hasil analisis menunjukkan bahwa dataset ToN-IoT memiliki karakteristik class imbalance yang signifikan, di mana kelas lalu

lintas normal mendominasi keseluruhan data, sementara beberapa kategori serangan memiliki jumlah sampel yang jauh lebih terbatas.

```
def analyze_target_distribution(self):
    """
    Analisis distribusi label dan type (target variables)
    Hanya menampilkan Plot 3 (Horizontal Bar)
    """
    display(HTML("<h2>2. ANALISIS TARGET VARIABLES (Label & Type)</h2>"))

    # Label Distribution
    if 'label' in self.df_raw.columns:
        display(HTML("<h3>Distribusi LABEL (Binary Classification)</h3>"))
        label_counts = self.df_raw['label'].value_counts()
        label_pct = self.df_raw['label'].value_counts(normalize=True) * 100

        label_df = pd.DataFrame({
            'Label': ['Normal' if x == 0 else 'Attack' for x in label_counts.index],
            'Count': label_counts.values,
            'Percentage': [f"{x:.2f}%" for x in label_pct.values]
        })
        display(label_df)

        # Imbalance ratio
        if len(label_counts) == 2:
            ratio = max(label_counts) / min(label_counts)
            display(HTML(f"<p><b>Imbalance Ratio:</b> {ratio:.2f}:1</p>"))
            if ratio > 10:
                display(HTML("<p   style='color:orange;'><b>WARNING:</b>   Highly
Imbalanced!</p>"))

    # Type Distribution
    if 'type' in self.df_raw.columns:
        display(HTML("<h3>Distribusi TYPE (Multiclass Classification)</h3>"))
        type_counts = self.df_raw['type'].value_counts()
        type_pct = self.df_raw['type'].value_counts(normalize=True) * 100

        type_df = pd.DataFrame({
            'Attack Type': type_counts.index,
            'Count': type_counts.values,
            'Percentage': [f"{x:.2f}%" for x in type_pct.values]
        })
        display(type_df)

        # --- Visualisasi: Plot 3 (Horizontal Bar) ---
```

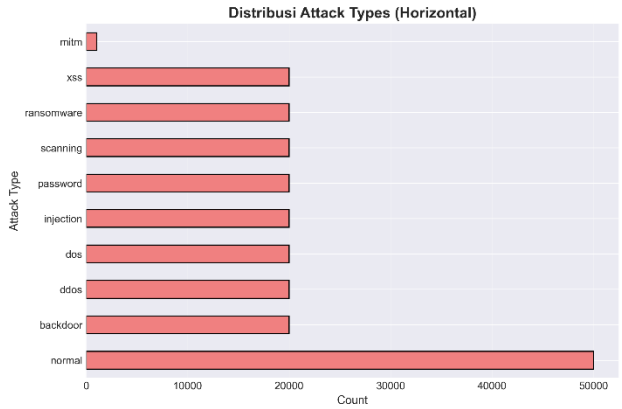
```
        plt.figure(figsize=(10, 6)) # Membuat canvas tunggal

        # Mengurutkan agar yang terbesar ada di atas (opsional tapi disarankan
untuk horizontal bar)
        type_counts.sort_values(ascending=True).plot(
            kind='barh',
            color='lightcoral',
            edgecolor='black'
        )

        plt.title('Distribusi    Attack    Types    (Horizontal)',    fontsize=14,
fontweight='bold')
        plt.xlabel('Count', fontsize=11)
        plt.ylabel('Attack Type', fontsize=11)
        plt.grid(axis='x', alpha=0.3)

        plt.tight_layout()
        plt.savefig('toniot_target_distribution.png',                    dpi=300,
bbox_inches='tight')
        display(HTML("<p><b>Visualisasi                                 disimpan:</b>
toniot_target_distribution.png</p>"))
        plt.show()

    # Markdown report
    self.markdown_report.append("## 2. Target Distribution\n\n")
    self.markdown_report.append("### Attack Types\n")
    if 'type' in self.df_raw.columns:
        self.markdown_report.append(type_df.to_markdown(index=False))
    self.markdown_report.append("\n\n")
```

Gambar 4.1 *Plotting* Distribusi kategori serangan pada dataset ToN-IoT Network

Distribusi hasil kode dari Gambar 4.1 ini ditunjukkan pada Tabel 2.8 Distribusi Data dan Karakteristik Kelas dan Gambar 4.2, di mana sebagian besar kategori serangan memiliki proporsi relatif seimbang. Sesuai dengan Tabel 2.8 Distribusi Data dan Karakteristik Kelas pada Gambar 4.2 terdapat ketimpangan ekstrem pada kelas *Man-in-the-Middle* (MITM) yang hanya mencakup sebagian kecil dari total sampel. Kondisi ini mencerminkan skenario realistis pada lingkungan *smart city*, di mana kemunculan serangan tertentu bersifat jarang dengan dampak yang tinggi.

Ketimpangan distribusi kelas ini menjadi dasar dalam perancangan skema pembagian data antar *client* menggunakan distribusi *dirichlet*. Pendekatan ini digunakan untuk

mensimulasikan kondisi Non-IID pada *Federated Learning*, sehingga setiap *client* memiliki karakteristik data lokal yang berbeda, baik dari sisi proporsi kelas maupun jumlah sampel.



Gambar 4.2 Distribusi kategori serangan pada dataset ToN-IoT Network

4.1.2 Kualitas dan Integritas Data

Tahap awal evaluasi kualitas data dilakukan dengan menganalisis keberadaan nilai yang hilang (*missing values*) serta penggunaan *placeholder* (seperti karakter '-') yang sering ditemukan dalam dataset network traffic. Kode yang digunakan untuk melakukan analisis ini disajikan pada Gambar 4.3.

```
def analyze_missing_values(self):
        """
        Analisis komprehensif missing values dan placeholder '-'
        """
        display(HTML("<h2>3. ANALISIS MISSING VALUES & PLACEHOLDER</h2>"))

        # Count actual NaN
        null_counts = self.df_raw.isnull().sum()
        null_pct = (null_counts / len(self.df_raw)) * 100

        # Count placeholder '-'
        dash_counts = (self.df_raw == '-').sum()
        dash_pct = (dash_counts / len(self.df_raw)) * 100

        # Total missing (NaN + '-')
        total_missing = null_counts + dash_counts
        total_missing_pct = (total_missing / len(self.df_raw)) * 100

        # Filter kolom dengan missing values
        missing_df = pd.DataFrame({
            'Column': self.df_raw.columns,
```

```
            'NaN_Count': null_counts.values,
            'NaN_Pct': null_pct.values,
            'Dash_Count': dash_counts.values,
            'Dash_Pct': dash_pct.values,
            'Total_Missing': total_missing.values,
            'Total_Missing_Pct': total_missing_pct.values
        })

        missing_df = missing_df[missing_df['Total_Missing'] > 0].sort_values(
            'Total_Missing_Pct', ascending=False
        )

        if len(missing_df) > 0:
            display(HTML("<h3>Kolom dengan Missing Values (Top 20)</h3>"))

            display_df = missing_df.head(20).copy()
            display_df['NaN_Pct']    =    display_df['NaN_Pct'].apply(lambda    x:
f"{x:.2f}%")
            display_df['Dash_Pct']   =   display_df['Dash_Pct'].apply(lambda   x:
f"{x:.2f}%")
            display_df['Total_Missing_Pct']                                      =
display_df['Total_Missing_Pct'].apply(lambda x: f"{x:.2f}%")

            display(display_df)

            # Summary statistics
            summary_df = pd.DataFrame({
                'Metric': [
                    'Kolom dengan missing',
                    'Kolom dengan >50% missing',
                    'Kolom dengan >90% missing'
                ],
                'Value': [
                    f"{len(missing_df)} dari {len(self.df_raw.columns)}",
                    str((missing_df['Total_Missing_Pct'] > 50).sum()),
                    str((missing_df['Total_Missing_Pct'] > 90).sum())
                ]
            })

            display(HTML("<h3>Ringkasan Missing Values</h3>"))
            display(summary_df)

            # Visualisasi
            top_missing = missing_df.head(15)
```

```
            plt.figure(figsize=(14, 8))
            x = np.arange(len(top_missing))
            width = 0.35

            plt.bar(x - width/2, top_missing['NaN_Pct'],
                   width, label='NaN', color='#ff6b6b', edgecolor='black')
            plt.bar(x + width/2, top_missing['Dash_Pct'],
                   width,      label="Placeholder      '-'",      color='#4ecdc4',
edgecolor='black')

            plt.xlabel('Kolom', fontsize=12, fontweight='bold')
            plt.ylabel('Persentase Missing (%)', fontsize=12, fontweight='bold')
            plt.title('Top   15   Kolom   dengan   Missing   Values',   fontsize=14,
fontweight='bold')
            plt.xticks(x, top_missing['Column'], rotation=45, ha='right')
            plt.legend(fontsize=11)
            plt.grid(axis='y', alpha=0.3)
            plt.tight_layout()
            plt.savefig('toniot_missing_values.png',                      dpi=300,
bbox_inches='tight')
            display(HTML("<p><b>Visualisasi                              disimpan:</b>
toniot_missing_values.png</p>"))
            plt.show()

            # Markdown report
            self.markdown_report.append("## 3. Missing Values Analysis\n\n")
            self.markdown_report.append(summary_df.to_markdown(index=False))
            self.markdown_report.append("\n\n")
        else:
            display(HTML("<p  style='color:green;'><b>Tidak  ada  missing  values
dalam dataset!</b></p>"))
```

Gambar 4.3 Kode Alalisis *Missing Values* pada Fitur Spesifik Protokol

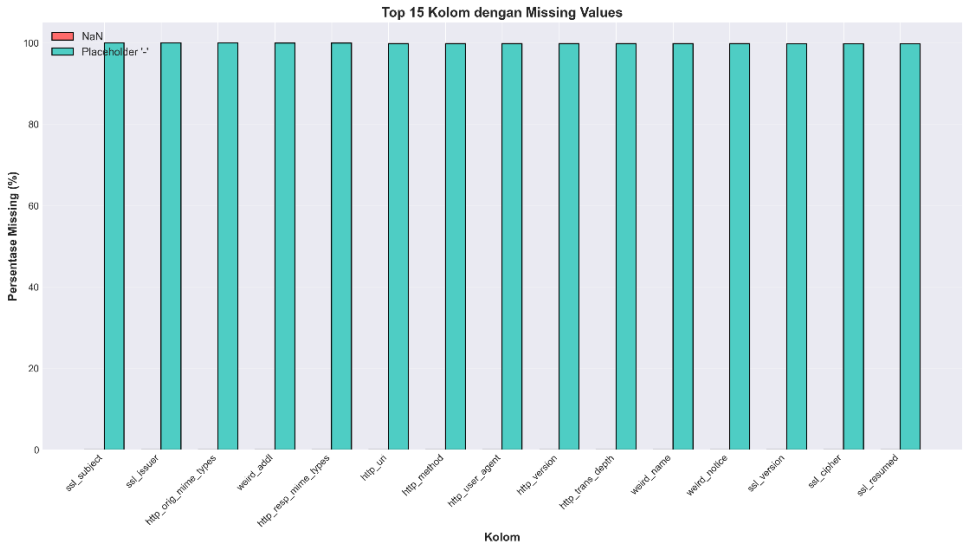
Hasil evaluasi dari blok kode pada Gambar 4.3 dilakukan untuk memastikan bahwa dataset bebas dari permasalahan mendasar yang dapat memengaruhi proses pelatihan model. Pemeriksaan *missing values* menunjukkan bahwa sejumlah fitur memiliki nilai kosong dalam proporsi tinggi, sebagaimana terlihat pada Gambar 4.4 dan Tabel 4.1 Kondisi ini tidak disebabkan oleh kesalahan pencatatan data, melainkan merupakan konsekuensi alami dari karakteristik lalu lintas jaringan. Fitur-fitur tersebut hanya relevan pada protokol tertentu

seperti HTTP, DNS, atau SSL, sehingga nilai kosong mengindikasikan bahwa koneksi jaringan tidak menggunakan protokol yang bersangkutan.

Tabel 4.1 Kolom dengan *Missing Values* (Top 20)

Number Label	Column	NaN_Count	NaN_Pct	Dash_Count	Dash_Pct	Total_Missing	Total_Missing_Pct
27	ssl_subject	0	0.00%	211032	99.99%	211032	99.99%
28	ssl_issuer	0	0.00%	211032	99.99%	211032	99.99%
37	http_orig_mime_types	0	0.00%	211027	99.99%	211027	99.99%
40	weird_addl	0	0.00%	210886	99.93%	210886	99.93%
38	http_resp_mime_types	0	0.00%	210839	99.90%	210839	99.90%
31	http_uri	0	0.00%	210756	99.86%	210756	99.86%
30	http_method	0	0.00%	210756	99.86%	210756	99.86%
36	http_user_agent	0	0.00%	210756	99.86%	210756	99.86%
32	http_version	0	0.00%	210745	99.86%	210745	99.86%
29	http_trans_depth	0	0.00%	210740	99.86%	210740	99.86%
39	weird_name	0	0.00%	210687	99.83%	210687	99.83%
41	weird_notice	0	0.00%	210687	99.83%	210687	99.83%
23	ssl_version	0	0.00%	210642	99.81%	210642	99.81%
24	ssl_cipher	0	0.00%	210642	99.81%	210642	99.81%

Number Label	Column	NaN_Count	NaN_Pct	Dash_Count	Dash_Pct	Total_Missing	Total_Missing_Pct
25	ssl_resumed	0	0.00%	210642	99.81%	210642	99.81%
26	ssl_established	0	0.00%	210642	99.81%	210642	99.81%
15	dns_query	0	0.00%	176198	83.49%	176198	83.49%
19	dns_AA	0	0.00%	176030	83.41%	176030	83.41%
20	dns_RD	0	0.00%	176030	83.41%	176030	83.41%
22	dns_rejected	0	0.00%	176030	83.41%	176030	83.41%



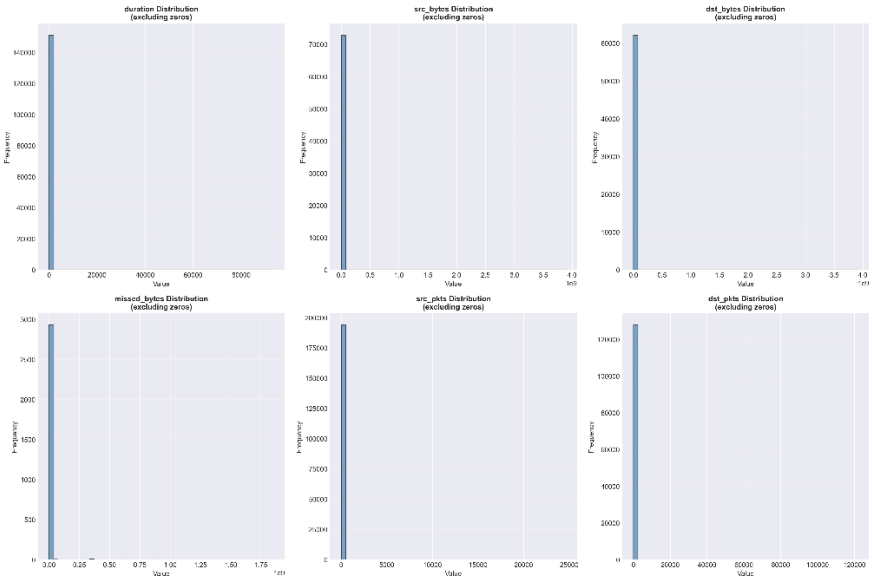
Gambar 4.4 Persentase *Missing Values* pada Fitur Spesifik Protokol

Oleh karena itu, nilai hilang ini dipertahankan sebagai informasi semantik dan ditangani secara *protocol-aware* pada tahap *data preparation*.

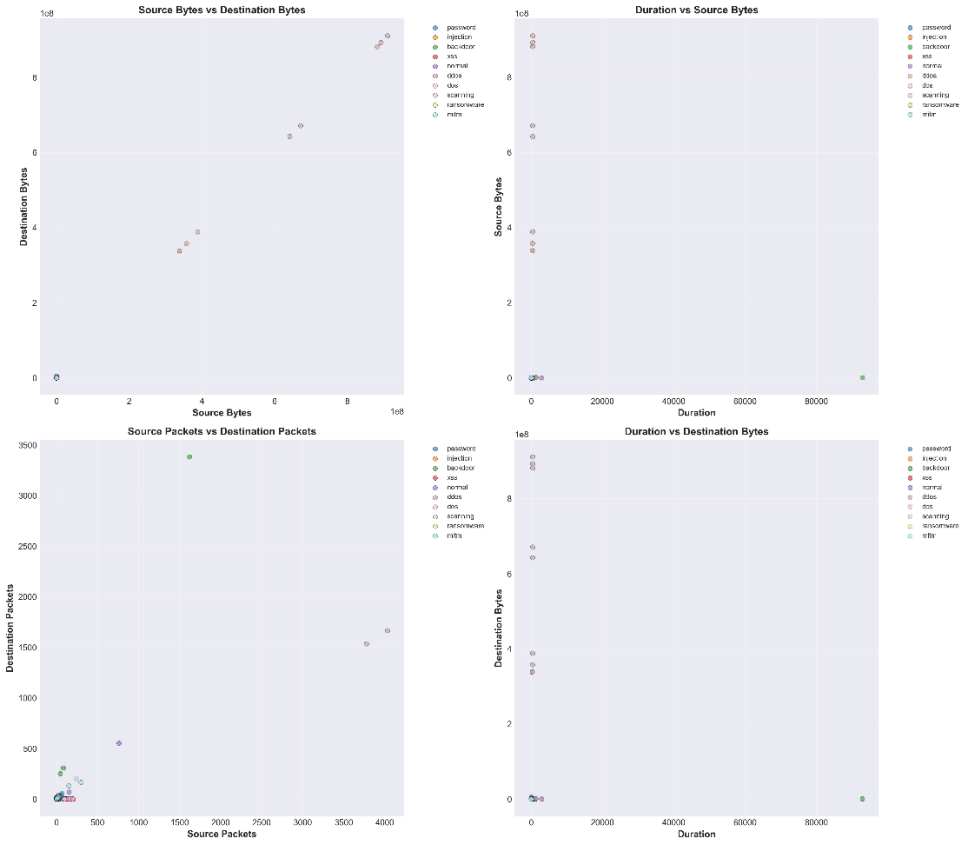
4.1.3 Karakteristik Statistik dan Relevansi Fitur

Analisis statistik dilakukan terhadap fitur numerik utama untuk memahami pola lalu lintas jaringan yang membedakan antara aktivitas normal dan berbagai jenis serangan. Hasil analisis pada Gambar 4.3 menunjukkan bahwa sebagian besar fitur numerik memiliki distribusi yang

skewed, dengan konsentrasi nilai pada rentang rendah dan keberadaan outlier ekstrem pada koneksi tertentu.



Gambar 4.5 Distribusi Fitur Numerik Utama



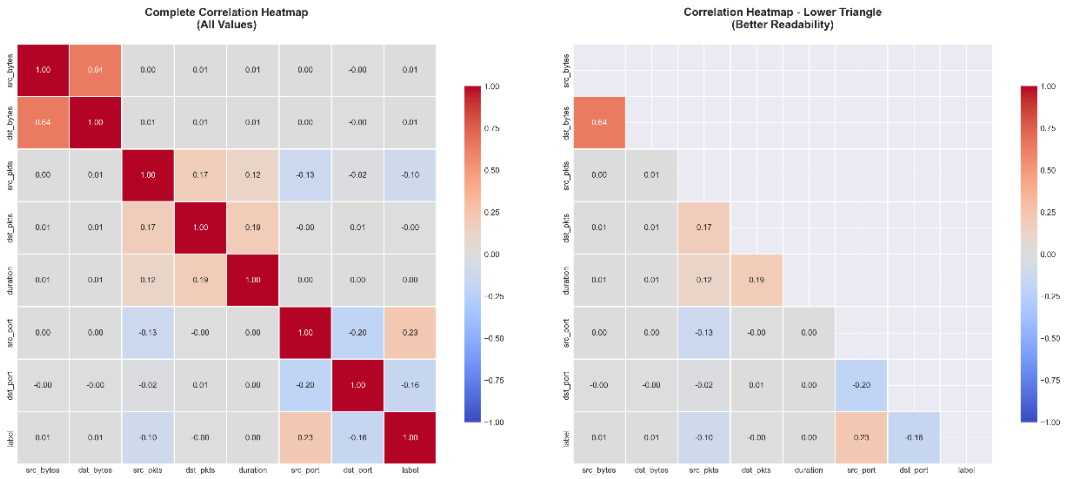
Gambar 4.6 *Scatter Plots* Persebaran Data per *Attack Type*

Pola ini mengindikasikan bahwa sebagian besar lalu lintas jaringan bersifat ringan dan berdurasi singkat. Namun, seperti terlihat pada scatter plot di Gambar 4.4, serangan tertentu seperti *Denial of Service* (DoS) dan *Distributed Denial of Service* (DDoS) menghasilkan volume data yang jauh lebih besar Gambar 4.4 kanan atas. Sebaliknya, serangan *Man-in-the-Middle* (MITM) cenderung memiliki durasi koneksi yang lebih panjang dibandingkan kategori lainnya. Hal ini dikonfirmasi oleh statistik pada Tabel 4.2.

Tabel 4.2 Statistik Deskriptif Fitur Utama per Tipe Serangan

Attack Type	Sample Size	src_bytes_mean	dst_bytes_mean	duration_mean
backdoor	20000	173.17	794.41	37.02
ddos	20000	2.38395e+06	2.49887e+06	17.08
dos	20000	21.19	4613.17	0.49
injection	20000	194818	200242	1.45
mitm	1043	14528.9	4657.85	28.05
normal	50000	2127.62	9767.99	3.99
password	20000	127.49	1222.27	0.71
ransomware	20000	0	0	0
scanning	20000	4.05	27.53	0.31
xss	20000	138478	510.22	12.76

Terakhir, analisis korelasi antar fitur numerik dilakukan untuk mendeteksi redundansi. Berdasarkan heatmap pada Gambar 4.7, terlihat bahwa sebagian besar pasangan fitur memiliki tingkat korelasi yang rendah mendekati nol, warna abu-abu.



Gambar 4.7 Heatmap Korelasi Pearson Antar Fitur Numerik

Tidak ditemukan korelasi tinggi yang mengindikasikan redundansi fitur secara signifikan. Temuan ini menunjukkan bahwa setiap fitur numerik membawa informasi yang relatif independen, sehingga proses seleksi fitur berbasis korelasi tidak diperlukan. Oleh karena itu, strategi seleksi fitur difokuskan pada metode berbasis variansi (Variance Threshold) untuk menghilangkan fitur dengan kontribusi informasi yang rendah.

4.2 Hasil Data Preparation

Tahap persiapan data bertujuan untuk mentransformasikan data mentah dataset ToN-IoT menjadi format numerik yang konsisten dan siap digunakan dalam pelatihan model *Federated Learning*. Implementasi tahap ini dilakukan melalui serangkaian proses sistematis yang mencakup pembersihan data berbasis protokol, transformasi fitur, seleksi fitur, dan standardisasi. Seluruh proses dirancang untuk menjaga integritas semantik data dan mencegah kebocoran informasi (*data leakage*) antara data latih dan data uji.

4.2.1 Implementasi Pembersihan Data

Berdasarkan strategi yang dirancang pada Bab 3, proses pembersihan data menangani *missing values* secara kontekstual. Nilai yang hilang pada kolom spesifik protokol tidak dihapus atau diimputasi dengan rata-rata global, melainkan diisi dengan nilai yang merepresentasikan "ketiadaan protokol" atau modus dari protokol tersebut jika aktif.

Realisasi dari strategi penanganan tersebut secara rinci dipaparkan pada Tabel 4.3 menjelaskan pembagian kategori fitur beserta alasan semantik di balik pemilihan nilai imputasi yang digunakan.

Tabel 4.3 Strategi Implementasi Penanganan *Missing Values*

Kategori Fitur	**Kolom Terkait**	**Nilai Imputasi**	**Alasan Semantik**
DNS	dns_query, dns_rcode, dll.	Mode (jika trafik DNS) atau -1	Membedakan koneksi DNS valid dan non-DNS.
HTTP	http_method, http_uri, dll.	Mode (jika trafik HTTP) atau 0	Menandai koneksi yang bukan aktivitas web.
SSL	ssl_cipher, ssl_subject, dll.	'none'	Menandai koneksi yang tidak terenkripsi.
Numerik Lain	Durasi, bytes, pkts	Median	Mengatasi *skewness* distribusi data.
Kategorikal Lain	Service, proto, state	Mode atau 'unknown'	Menangani kategori yang hilang.

Setelah proses imputasi yang tertera pada Tabel 4.3 selesai dilakukan, tahap selanjutnya adalah reduksi fitur. Kolom-kolom yang tidak relevan untuk deteksi intrusi atau yang bersifat unik untuk setiap koneksi, seperti alamat IP (*source/destination IP*) dan nomor *port*, dihapus dari dataset. Langkah ini bertujuan untuk mengurangi *noise* dan mencegah model mengalami *overfitting* atau menghafal alamat spesifik yang dapat menurunkan kemampuan generalisasi model pada trafik baru.

4.2.2 Transformasi dan Seleksi Fitur

Transformasi variabel kategorikal dilakukan untuk mengonversi data non-numerik menjadi format matriks yang dapat diproses oleh *deep learning*.

a. Label Encoding: Diterapkan pada variabel target *type* untuk klasifikasi *multiclass*, mengubah 10 jenis serangan menjadi integer 0-9 pada Tabel 4.4.

Tabel 4.4 Pemetaan Label *Encoding* (*Multiclass*)

Encoded Label (Int)	Kategori Serangan (Asli)	Deskripsi Singkat
0	backdoor	Serangan akses belakang
1	ddos	Distributed Denial of Service
2	dos	Denial of Service
3	injection	Serangan injeksi kode/SQL
4	mitm	Man-in-the-Middle (Kelas Minoritas)
5	normal	Lalu Lintas Normal
6	password	Serangan brute-force password
7	ransomware	Serangan enkripsi data jahat
8	scanning	Pemindaian port/jaringan
9	xss	Cross-Site Scripting

b. One-Hot Encoding: Diterapkan pada fitur kategorikal nominal seperti *service*, *conn_state* untuk menghilangkan asumsi urutan ordinal.

Setelah *encoding*, fitur-fitur yang memiliki variansi sangat rendah (nilai konstan atau hampir konstan) dieliminasi menggunakan metode *variance threshold* dengan ambang batas 0.01. Hal ini menghasilkan pengurangan dimensi fitur yang signifikan tanpa mengorbankan informasi diskriminatif.

Selanjutnya, fitur numerik distandardisasi menggunakan *StandardScaler* untuk mencapai distribusi dengan rata-rata 0 dan deviasi standar 1. Proses *fitting scaler* dilakukan secara eksklusif pada data latih (*training set*) untuk mencegah kebocoran statistik ke data uji.

4.2.3 Hasil Pembagian Data (Data Splitting)

Data dibagi menggunakan metode stratified splitting dengan rasio 80:20. Metode ini memastikan bahwa proporsi kelas minoritas, khususnya serangan MITM (0.49%), tetap terjaga baik di data latih maupun data uji distribusinya pada Tabel 4.5 berikut.

Tabel 4.5 Dimensi Data Hasil *Preprocessing*

Himpunan Data	Jumlah Sampel	Persentase	Keterangan

Training Set	168,834	80%	Digunakan untuk pelatihan lokal di *client* FL
Test Set	42,209	20%	Digunakan untuk evaluasi global di server
Total	211,043	100%	Total records setelah cleaning

4.2.4 Implementasi *Pipeline* (*Core Code*)

Proses *preprocessing* diimplementasikan dalam dua *pipeline* terpisah untuk skenario klasifikasi *binary* dan *multiclass*. Kode berikut menunjukkan implementasi inti yang menggabungkan seluruh tahapan di atas dengan *core code* pada Gambar 4.8.

```
def clean_data (self, df):
        """
        Protocol-aware preprocessing for TON_IoT dataset.
        Handles missing values based on protocol context.
        """
        print("\n" + "="*70)
        print("SMART PREPROCESSING (Protocol-Aware)")
        print("="*70)

        # DNS columns
        dns_cols = ['dns_query', 'dns_qclass', 'dns_qtype', 'dns_rcode',
                    'dns_AA', 'dns_RD', 'dns_RA', 'dns_rejected']

        # HTTP columns
        http_cols = ['http_trans_depth', 'http_method', 'http_uri',
                     'http_version', 'http_request_body_len',
                     'http_response_body_len', 'http_status_code',
                     'http_user_agent', 'http_orig_mime_types',
                     'http_resp_mime_types']

        # SSL columns
        ssl_cols = ['ssl_version', 'ssl_cipher', 'ssl_resumed',
                    'ssl_established', 'ssl_subject', 'ssl_issuer']

        print("Processing DNS columns...")
        for col in dns_cols:
            if col in df.columns:
                # If DNS traffic (has dns_query), fill with -1; else with 0
                mask_dns = df['dns_query'].notna() if 'dns_query' in df.columns
else pd.Series([False]*len(df))
                df[col]     =     df[col].fillna(df[col].mode()[0]     if     not
df[col].mode().empty else -1)

        print("Processing HTTP columns...")
```

```
        for col in http_cols:
            if col in df.columns:
                # If HTTP traffic (has http_method), fill with mode; else with 0
                mask_http = df['http_method'].notna() if 'http_method' in
df.columns else pd.Series([False]*len(df))
                df[col] = df[col].fillna(df[col].mode()[0] if not
df[col].mode().empty else 0)

        print("Processing SSL columns...")
        for col in ssl_cols:
            if col in df.columns:
                # If SSL traffic, fill with mode; else with 'none'
                df[col] = df[col].fillna('none')

        # Fill remaining numeric columns with median
        print("Processing remaining numeric columns...")
        numeric_cols = df.select_dtypes(include=[np.number]).columns
        for col in numeric_cols:
            if df[col].isna().sum() > 0:
                df[col] = df[col].fillna(df[col].median())

        # Fill remaining categorical columns with mode
        print("Processing remaining categorical columns...")
        cat_cols = df.select_dtypes(include=['object']).columns
        for col in cat_cols:
            if df[col].isna().sum() > 0:
                mode_val = df[col].mode()[0] if not df[col].mode().empty else
'unknown'
                df[col] = df[col].fillna(mode_val)

        remaining_nan = df.isna().sum().sum()
        print(f"✓ Smart preprocessing complete. Remaining NaN: {remaining_nan}")

        return df
```

Gambar 4.8 *Pipeline Core Code Cleaning Data*

Pipeline **Klasifikasi** ***Binary***

Pada Gambar 4.9 seluruh kategori serangan disatukan menjadi label 1 (Attack) dan lalu lintas normal menjadi 0 (Normal).

```
def prepare_binary_classification(self, df):
    # 1. Protocol-Aware Cleaning
    df = self.clean_data(df)
```

```
    # 2. Define Target (0: Normal, 1: Attack)
    # Drop detailed attack 'type', keep binary 'label'
    df_binary = df.drop(columns=['type'] + self.DROP_COLS, errors='ignore')
    X = df_binary.drop('label', axis=1)
    y = df_binary['label']

    # 3. Stratified Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    # 4. Encoding & Selection Pipeline
    # OneHotEncoder -> VarianceThreshold -> StandardScaler
    # Note: Fit only on X_train to prevent leakage
    X_train_processed = self.pipeline.fit_transform(X_train)
    X_test_processed = self.pipeline.transform(X_test)

    return {
        'X_train': X_train_processed, 'X_test': X_test_processed,
        'y_train': y_train.values,    'y_test': y_test.values
    }
```

Gambar 4.9 *Pipeline* Klasifikasi *Binary Class*

Pipeline Klasifikasi *Multiclass*

Pada Gambar 4.10 target variabel adalah 10 kelas spesifik serangan. Implementasi mencakup Label Encoding untuk target variabel.

```
def prepare_multiclass_classification(self, df):
    # 1. Protocol-Aware Cleaning
    df = self.clean_data(df)

    # 2. Define Target (10 Classes)
    # Drop binary 'label', keep detailed 'type'
    df_multi = df.drop(columns=['label'] + self.DROP_COLS, errors='ignore')
    X = df_multi.drop('type', axis=1)
    y = df_multi['type']

    # 3. Label Encoding for Target
    le = LabelEncoder()
    y_encoded = le.fit_transform(y)
```

```
    # 4. Stratified Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(
        X, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
    )

    # 5. Encoding & Selection Pipeline
    # OneHotEncoder -> VarianceThreshold -> StandardScaler
    X_train_processed = self.pipeline.fit_transform(X_train)
    X_test_processed = self.pipeline.transform(X_test)

    return {
        'X_train': X_train_processed, 'X_test': X_test_processed,
        'y_train': y_train,           'y_test': y_test,
        'num_classes': len(le.classes_), 'classes': le.classes_
    }
```

Gambar 4.10 *Pipeline* Klasifikasi *Multiclass*

4.2.5 Output Artifacts

```
def save_artifacts(self, artifacts, filename):
        filepath = self.config.OUTPUT_DIR / filename
        with open(filepath, 'wb') as f:
            pickle.dump(artifacts, f, protocol=pickle.HIGHEST_PROTOCOL)

        size_mb = filepath.stat().st_size / (1024 * 1024)
        print(f"\nSaved: {filepath}")
        print(f"   Size: {size_mb:.2f} MB")

        return filepath

    def run(self, mode='full'):
        print("\n" + "="*70)
        print("PREPROCESSING PIPELINE - NEURAL NETWORK OPTIMIZED")
        print("="*70)
        print(f"Mode: {mode}")
        print(f"Output: {self.config.OUTPUT_DIR}")

        total_start = time.time()

        df = self.load_data()

        results = {}
```

```
        if mode in ['full', 'binary']:
            print("\n" + "PROCESSING BINARY CLASSIFICATION...")
            binary_artifacts = self.prepare_binary_classification(df)
            binary_path          =          self.save_artifacts(binary_artifacts,
'binary_preprocessed.pkl')
            results['binary'] = binary_path

        if mode in ['full', 'multi']:
            print("\n" + "PROCESSING MULTICLASS CLASSIFICATION...")
            multi_artifacts = self.prepare_multiclass_classification(df)
            multi_path           =           self.save_artifacts(multi_artifacts,
'multiclass_preprocessed.pkl')
            results['multi'] = multi_path

        total_time = time.time() - total_start

        print("\n" + "="*70)
        print("PREPROCESSING COMPLETED")
        print("="*70)
        print(f"Total time: {total_time:.2f}s ({total_time/60:.2f} minutes)")
        print(f"\nSaved artifacts:")
        for key, path in results.items():
            print(f"  - {key}: {path}")

        print("\nUsage in notebooks:")
        print("  >>> import pickle")
        print("    >>>  with  open('processed_artifacts/binary_preprocessed.pkl',
'rb') as f:")
        print("  >>>     artifacts = pickle.load(f)")
        print("  >>> X_train = artifacts['X_train']")
        print("  >>> X_test = artifacts['X_test']")
        print("  >>> y_train = artifacts['y_train']")
        print("  >>> y_test = artifacts['y_test']")

        return results
```

Gambar 4.11 *Pipeline Eksekusi dan Serialisasi Artefak Preprocessing*

Hasil akhir dari proses Gambar 4.11 disimpan dalam format serialisasi *.pkl* yang memuat data latih dan uji yang telah diskalakan, label yang telah dienkripsi, serta metadata *pipeline scaler* dan *encoder* yang telah di-*fitting*. Artefak ini (binary_preprocessed.pkl dan multiclass_preprocessed.pkl) menjadi input langsung bagi algoritma *Federated Learning*

untuk mendistribusikan data ke *client* tanpa perlu melakukan pemrosesan ulang, menjamin konsistensi fitur di seluruh eksperimen.

4.3 Modeling dan Implementasi Sistem

Tahap Modeling merupakan inti dari implementasi penelitian ini, di mana arsitektur jaringan saraf tiruan dan algoritma agregasi *Federated Learning* dikonfigurasi untuk membangun sistem deteksi intrusi yang terdistribusi. Fokus utama pada tahap ini adalah penerjemahan desain arsitektur model *MLP* dan *CNN-1D* ke dalam kode yang dapat dieksekusi, serta konfigurasi strategi pelatihan *federated* untuk menangani heterogenitas data pada lingkungan *smart city*.

Implementasi dilakukan menggunakan kerangka kerja simulasi Flower (flwr) berbasis Python, yang memungkinkan orkestrasi pelatihan antara server pusat dan *client* (*gateway*) secara efisien.

4.3.1 Konfigurasi Lingkungan Eksperimen

Untuk menjamin validitas dan reprodusibilitas hasil eksperimen, lingkungan simulasi diatur dengan parameter tetap yang konsisten di seluruh skenario pengujian. Tabel 4.6 Konfigurasi Lingkungan Tetap merinci spesifikasi teknis lingkungan eksperimen yang digunakan.

Tabel 4.6 Konfigurasi Lingkungan Tetap

Parameter	**Nilai**
Framework	Flower (flwr) 1.24.0
Backend	TensorFlow 2.20.0
Jumlah *Client*	5
Jumlah Rounds	20
Distribusi Data	Non-IID (Dirichlet)
Random Seed dan Python Hash Seed	42

Selain parameter di atas, kontrol variabilitas acak diterapkan secara ketat pada level kode untuk memastikan setiap eksekusi menghasilkan keluaran yang deterministik. Implementasi *seeding* pada Gambar 4.12 .

```
import numpy as np
import tensorflow as tf
import os

# Set random seeds untuk reprodusibilitas
np.random.seed(42)
```

```
tf.random.set_seed(42)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
os.environ['PYTHONHASHSEED'] = '42'
```

Gambar 4.12 *Seed Value*

Konfigurasi ini memastikan bahwa perbedaan performa yang diamati semata-mata disebabkan oleh variasi algoritma atau *hyperparameter* yang diuji, bukan oleh inisialisasi bobot yang acak.

4.3.2 Strategi Horizontal *Federated Learning*

Sesuai dengan desain penelitian, sistem ini menerapkan skema *Horizontal Federated Learning* (HFL). Dalam skema ini, setiap *client* (merepresentasikan *gateway* IoT) memiliki struktur fitur yang identik (35 fitur hasil preprocessing), namun memiliki sampel data yang berbeda berdasarkan lokasi dan pola trafik lokalnya .

Dalam implementasi ini, server bertugas menginisialisasi model global dan mendistribusikannya ke 5 *client*. Setiap *client* melakukan pelatihan lokal (*local training*) menggunakan data privat mereka dan hanya mengirimkan pembaruan parameter model (*weights*) kembali ke server untuk agregasi. Data mentah lalu lintas jaringan tetap berada di sisi *client*, sehingga privasi data terjaga secara inheren.

4.3.3 Implementasi Arsitektur Deep Learning

Dua arsitektur model, yaitu Multi-Layer Perceptron (MLP) dan Convolutional Neural Network 1D (CNN-1D), diimplementasikan menggunakan TensorFlow/Keras. Kedua model dirancang fleksibel untuk menangani tugas klasifikasi *binary output* 1 *neuron* dengan *sigmoid* maupun *multiclass* output 10 neuron dengan *softmax*.

Multi-Layer Perceptron (MLP)

Model MLP dirancang sebagai *baseline* yang efisien untuk mempelajari hubungan non-linear antar fitur statistik. Arsitektur ini menggunakan lapisan Dense bertingkat dengan regularisasi Dropout dan L2 untuk mencegah *overfitting*.

```
    def create_mlp_model(input_shape, num_classes=1, learning_rate=0.001):
        model = Sequential()

        # Layer 1: Input & Hidden
        model.add(Dense(256 if num_classes == 1 else 512, activation='relu',
                        input_shape=(input_shape,), kernel_regularizer=l2(0.001)))
```

```
        model.add(BatchNormalization())
        model.add(Dropout(0.3 if num_classes == 1 else 0.4))

        # Layer 2
        model.add(Dense(128 if num_classes == 1 else 256, activation='relu',
                        kernel_regularizer=l2(0.001)))
        model.add(Dropout(0.3 if num_classes == 1 else 0.4))

        # Layer 3 & 4 (Simplified)
        model.add(Dense(64 if num_classes == 1 else 128, activation='relu'))
        model.add(Dropout(0.2))

        # Output Layer
        if num_classes == 1:
            model.add(Dense(1, activation='sigmoid'))
            loss = 'binary_crossentropy'
        else:
            model.add(Dense(num_classes, activation='softmax'))
            loss = 'sparse_categorical_crossentropy'

        model.compile(optimizer=Adam(learning_rate=learning_rate),
                      loss=loss, metrics=['accuracy'])
        return model
```

Gambar 4.13 *Pipeline* MLP

Convolutional Neural Network 1D (CNN-1D)

Model *CNN-1D* diimplementasikan untuk menangkap pola lokal spasial dari fitur jaringan. Input data di-*reshape* menjadi format 3D *samples, Features, 1* agar kompatibel dengan lapisan konvolusi 1D.

```
    def create_cnn_model(input_shape, num_classes=1, learning_rate=0.001):
        model = Sequential()

        # Reshape input for CNN (batch, steps, channels)
        model.add(Input(shape=(input_shape, 1)))

        # Conv Block 1
         model.add(Conv1D(64 if num_classes == 1 else 128, 3, activation='relu',
padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling1D(2))
```

```
        # Conv Block 2
         model.add(Conv1D(32 if num_classes == 1 else 64, 3, activation='relu',
padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling1D(2))

        # Flatten & Dense
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.4))

        # Output Layer
        if num_classes == 1:
            model.add(Dense(1, activation='sigmoid'))
            loss = 'binary_crossentropy'
        else:
            model.add(Dense(num_classes, activation='softmax'))
            loss = 'sparse_categorical_crossentropy'

        model.compile(optimizer=Adam(learning_rate=learning_rate),
                      loss=loss, metrics=['accuracy'])
        return model
```

Gambar 4.14 *Pipeline* CCN-1D

4.3.4 Simulasi Distribusi Data Non-IID

Untuk mensimulasikan kondisi heterogenitas data *smart city*, implementasi menggunakan Distribusi Dirichlet. Parameter konsentrasi α digunakan untuk mengontrol tingkat ketimpangan data antar *client*. Untuk menyimulasikan kondisi Non-IID, nilai dari α adalah 0.3, sedangkan untuk mensimulasikan kondisi IID, nilai dari α adalah 5 untuk mendekati distribusi seimbang antara *client*.

```
    def split_data_non_iid(X, y, n_clients=5, alpha=0.3):
        # Dapatkan indeks untuk setiap kelas
        n_classes = len(np.unique(y))
        client_indices = [[] for _ in range(n_clients)]

        for k in range(n_classes):
            idx_k = np.where(y == k)[0]
            np.random.shuffle(idx_k)

            # Generate proporsi menggunakan Dirichlet
```

```
            proportions = np.random.dirichlet(np.repeat(alpha, n_clients))

            # Split indeks kelas k ke setiap client berdasarkan proporsi
                    splits = np.split(idx_k, (np.cumsum(proportions)[:-1] *
len(idx_k)).astype(int))

            for i in range(n_clients):
                client_indices[i].extend(splits[i])

        return client_indices
```

Gambar 4.15 *Pipeline Dirichlet*

Distribusi Data per *Client* pada *Binary classification*

Tabel 4.7 Distribusi Data *Client Binary Classification* (α=0.3 - Non-IID)

Client	Total Samples	Normal	Normal %	Attack	Attack %
Client 1	56,382	34,004	60.30%	22,378	39.70%
Client 2	102,976	5,332	5.20%	97,644	94.80%
Client 3	1,419	252	17.80%	1,167	82.20%
Client 4	**5,411**	**6**	**0.10%**	**5,405**	**99.90%**
Client 5	2,646	406	15.30%	2,240	84.70%
TOTAL	168,834	40,000	-	128,834	-

Tabel 4.8 Distribusi Data *Client Binary Classification* (α=5 - IID)

Client	Total Samples	Normal	Normal %	Attack	Attack %
Client 1	18,363	9,011	49.10%	9,352	50.90%
Client 2	52,347	12,919	24.70%	39,428	75.30%
Client 3	**42,125**	**6,591**	**15.60%**	**35,534**	**84.40%**
Client 4	34,395	6,444	18.70%	27,951	81.30%
Client 5	21,604	5,035	23.30%	16,569	76.70%
TOTAL	168,834	40,000	-	128,834	-

Tabel 4.7 dan Tabel 4.8 menyajikan distribusi data *binary classification* (normal vs *attack*) pada masing-masing *client* untuk dua konfigurasi tingkat heterogenitas data, yaitu $\alpha = 0.3$ (Non-IID) dan $\alpha = 5.0$ (IID-like).

Pada skenario Non-IID ($\alpha = 0.3$), terlihat bahwa distribusi kelas antar *client* sangat tidak seimbang. Beberapa *client* didominasi hampir sepenuhnya oleh data serangan, sementara *client* lain memiliki proporsi lalu lintas normal yang relatif lebih tinggi. Sebagai contoh, Client 4 memiliki proporsi data serangan sebesar 99.9%, sedangkan Client 1 masih memiliki distribusi

yang relatif lebih seimbang dengan 39.7% data serangan. Rentang rasio serangan yang sangat lebar, yaitu 0.397 hingga 0.999, menunjukkan tingkat heterogenitas data yang ekstrem antar *client*.

Tingginya variasi distribusi ini tercermin dari nilai standar deviasi rasio serangan sebesar 0.213, yang mengindikasikan kondisi Non-IID yang kuat. Kondisi ini merepresentasikan skenario realistis pada lingkungan IoT dan *smart city*, di mana setiap *gateway* atau node dapat mengamati pola lalu lintas jaringan yang sangat berbeda bergantung pada fungsi dan lokasi perangkat.

Sebaliknya, pada konfigurasi IID-like ($\alpha = 5.0$) yang ditunjukkan pada Tabel 4.8, distribusi kelas antar *client* menjadi lebih merata. Proporsi data serangan pada setiap *client* berada dalam rentang yang lebih sempit, yaitu 0.509 hingga 0.844, dengan standar deviasi yang menurun menjadi 0.118. Penurunan variasi ini menunjukkan bahwa data antar *client* semakin mendekati asumsi IID, meskipun distribusi kelas belum sepenuhnya identik.

Distribusi Data per *Client* pada multiclass classification

Tabel 4.9 Distribusi Data *Client Multiclass Classification* (α=0.3 - Non-IID)

Client	Total	0	1	2	3	4	5	6	7	8	9
Client 1	44,505	1.59 %	24.98 %	25.27 %	0.84 %	0.01%	5.10%	13.21%	22.61%	4.12%	2.27%
Client 2	61,553	23.78 %	4.42 %	3.89%	0.11 %	0.99%	58.47 %	0.25%	4.30%	2.80%	0.99%
Client 3	26,032	0.00 %	0.00 %	4.24%	48.20 %	0.01%	2.87%	0.00%	2.95%	28.31 %	13.41 %
Client 4	12,697	0.06 %	14.43 %	1.24%	1.30 %	1.64%	0.00%	2.21%	0.19%	6.79%	72.14 %
Client 5	24,047	2.70 %	1.37 %	4.54%	11.84 %	0.05%	4.13%	40.28%	10.39%	17.51 %	7.20%
TOTAL	168,834	9.48 %	9.48 %	9.48%	9.48 %	0.49%	23.69 %	9.48%	9.48%	9.48%	9.48%

Tabel 4.10 Distribusi Data *Client Multiclass Classification* (α=5 - IID)

Client	Total	0	1	2	3	4	5	6	7	8	9
Client 1	41.509	6.36%	12.29%	7.64 %	9.09 %	0.36 %	27.74%	6.99%	10.29 %	13.85%	5.39 %
Client 2	31.124	6.11%	12.70%	8.65 %	4.77 %	0.50 %	31.13%	15.74 %	5.17%	7.87 %	7.36 %
Client 3	27.492	10.19 %	7.30 %	10.00%	11.40%	0.60 %	32.19%	5.85%	7.78%	5.22 %	9.47 %
Client 4	33.395	14.51 %	10.42%	13.55%	12.57%	0.24 %	10.37%	5.14%	11.15 %	14.56%	7.49 %
Client 5	35.314	10.79 %	4.13 %	8.10 %	9.66 %	0.81 %	18.36%	13.80 %	12.06 %	4.26 %	18.03%

TOTAL	168,834	9.48%	9.48%	9.48%	9.48%	0.49%	23.69%	9.48%	9.48%	9.48%	9.48%

Distribusi data untuk skenario *multiclass* classification ditampilkan pada Tabel 4.9 (α = 0.3) dan Tabel 4.10 (α = 5.0). Dataset terdiri atas sepuluh kelas, termasuk sembilan jenis serangan dan satu kelas normal.

Pada skenario Tabel 4.9 Non-IID (α = 0.3), ketimpangan distribusi antar *client* terlihat jauh lebih kompleks dibandingkan kasus binary. Beberapa *client* hanya memiliki subset kelas tertentu, sementara kelas lain hampir tidak muncul sama sekali. Sebagai contoh, Client 3 didominasi oleh kelas *scanning* dan *xss* dengan hampir tidak adanya kelas *backdoor* dan *injection*. Sementara itu, Client 2 memiliki dominasi kuat pada kelas *normal* dan *backdoor*, tetapi sangat minim pada kelas *password* dan *injection*.

Pola ini menunjukkan bahwa pada kondisi Non-IID, setiap *client* tidak hanya memiliki proporsi kelas yang berbeda, tetapi juga keragaman jenis serangan yang sangat terbatas. Kondisi tersebut mencerminkan tantangan utama *Federated Learning*, di mana model lokal dilatih pada distribusi data yang sangat berbeda satu sama lain.

Pada konfigurasi IID-like (α = 5.0), distribusi kelas *multiclass* antar *client* menjadi jauh lebih seimbang. Setiap *client* memiliki seluruh kelas dengan proporsi yang relatif seragam, meskipun variasi kecil masih dapat diamati. Tidak ada lagi *client* yang hanya didominasi oleh satu atau dua kelas tertentu, sehingga setiap model lokal memiliki cakupan pengetahuan yang lebih luas terhadap berbagai jenis serangan.

Perbandingan Tingkat Heterogenitas

Tabel 4.7, Tabel 4.8, Tabel 4.9, dan Tabel 4.10 menyajikan ringkasan perbandingan tingkat heterogenitas antara skenario Non-IID (α = 0.3) dan IID-like (α = 5.0). Pada kasus binary, standar deviasi rasio serangan pada α = 0.3 tercatat 1.8 kali lebih tinggi dibandingkan α = 5.0. Perbedaan ini menjadi lebih signifikan pada skenario *multiclass*, di mana rata-rata standar deviasi distribusi kelas pada α = 0.3 mencapai sekitar 3.6 kali lebih besar dibandingkan skenario IID-like.

Hasil ini mengonfirmasi bahwa heterogenitas data pada klasifikasi *multiclass* bersifat lebih kompleks dan lebih menantang dibandingkan *binary classification*. Dengan demikian, evaluasi performa model *Federated Learning* pada kedua konfigurasi ini menjadi penting untuk menilai kemampuan algoritma dalam menangani variasi distribusi data yang ekstrem maupun mendekati IID.

4.3.5 Implementasi *Federated Learning* dengan Flower

Implementasi sistem FL menggunakan kelas NumPyClient dari Flower. *Client* menangani pelatihan lokal, sementara strategi server mengatur algoritma agregasi.

Client Flower (*Local Training*)

Client menerima bobot global, melatih model pada data lokalnya, dan mengembalikan bobot baru serta metrik evaluasi ke server.

```
    class FlowerClient(fl.client.NumPyClient):
        def __init__(self, model, X_train, y_train, X_val, y_val, batch_size,
epochs):
            self.model = model
            self.X_train, self.y_train = X_train, y_train
            self.X_val, self.y_val = X_val, y_val
            self.batch_size = batch_size
            self.epochs = epochs

        def fit(self, parameters, config):
            self.model.set_weights(parameters)
            history = self.model.fit(
                self.X_train, self.y_train,
                epochs=self.epochs,
                batch_size=self.batch_size,
                verbose=0
            )
                return self.model.get_weights(), len(self.X_train), {'loss':
history.history['loss'][-1]}

        def evaluate(self, parameters, config):
            self.model.set_weights(parameters)
            loss, accuracy = self.model.evaluate(self.X_val, self.y_val, verbose=0)
            return loss, len(self.X_val), {"accuracy": accuracy}
```

Gambar 4.16 *Pipeline client*t

Algoritma Agregasi (Server)

Penelitian ini mengimplementasikan dua strategi agregasi utama, yaitu *Federated Averaging* (FedAvg) yang menggunakan rata-rata tertimbang berdasarkan jumlah sampel data *client*, serta *Federated Proximal* (FedProx) yang menambahkan *proximal term* (μ) pada fungsi

loss *client* untuk membatasi deviasi model lokal dari model global, sehingga menjadi krusial dalam menangani data Non-IID.

```
# Contoh inisialisasi strategi FedAvg
strategy = fl.server.strategy.FedAvg(
    fraction_fit=1.0,  # Semua client berpartisipasi setiap round
    min_fit_clients=5,
    min_available_clients=5,
    eval_fn=get_server_eval_fn() # Evaluasi global pada server
)

# Simulasi dijalankan dengan konfigurasi server
fl.simulation.start_simulation(
    client_fn=client_fn,
    num_clients=5,
    config=fl.server.ServerConfig(num_rounds=20),
    strategy=strategy
)
```

Gambar 4.17 Contoh inisiasi server *Federated Learning*

4.3.6 Simulasi Distribusi Data Non-IID

Untuk memastikan performa optimal dalam batasan sumber daya yang ada, dilakukan *Grid Search* terbatas pada parameter kunci. Pencarian ini bertujuan menemukan keseimbangan antara stabilitas konvergensi dan efisiensi komputasi.

Ruang Pencarian *Hyperparameter*:

a. Batch Size: [*256, 512, 1024*]
b. Local Epochs: [*1, 2*]
c. Learning Rate: [*0.001, 0.0005*]
d. FedProx μ: [*0.01, 0.001*] (khusus skenario FedProx)

Setiap kombinasi *hyperparameter* diuji pada skenario Non-IID ($\alpha = 0.3$) dan IID-like ($\alpha = 5$) untuk mengevaluasi ketahanan model terhadap heterogenitas data.

Hasil *Grid Search* pada Kondisi Non-IID (α = 0.3)

Tabel 4.11 Hasil *Grid Search* (α=0.3 – Non-IID)

Batch Size	E	LR	Model	Binary Acc	Binary F1	Binary AUC	Multi Acc	Multi F1	Multi AUC	Train Time (s)
256	1	0.001	MLP	95.52%	97.14%	99.28%	71.11%	68.31 %	96.43%	162.2

256	1	0.001	CNN	95.73%	97.27%	99.18%	43.19%	32.99%	89.43%	436.9
256	1	0.0005	MLP	95.00%	96.82%	99.33%	67.16%	63.49%	96.48%	162.2
256	1	0.0005	CNN	95.72%	97.27%	99.67%	72.23%	69.73%	96.90%	426.4
256	2	0.001	MLP	95.46%	97.10%	99.23%	69.36%	66.26%	97.84%	229.8
256	2	0.001	CNN	88.91%	92.22%	99.09%	52.29%	45.81%	85.73%	660.3
256	2	0.0005	MLP	95.61%	97.20%	99.35%	71.02%	68.25%	96.39%	228.1
256	2	0.0005	CNN	95.74%	97.28%	99.75%	70.98%	66.26%	97.47%	639
512	1	0.001	MLP	95.62%	97.21%	99.31%	67.23%	63.57%	96.20%	140
512	1	0.001	CNN	95.60%	97.19%	99.57%	45.22%	36.25%	93.36%	445.7
512	**1**	**0.0005**	**MLP**	**95.62%**	**97.20%**	**99.33%**	**66.77%**	**63.12%**	**95.84%**	**141.5**
512	**1**	**0.0005**	**CNN**	**95.71%**	**97.26%**	**99.59%**	**73.27%**	**70.99%**	**97.64%**	**447.4**
512	2	0.001	MLP	95.68%	97.24%	99.37%	71.18%	68.14%	97.56%	186.4
512	2	0.001	CNN	86.44%	91.80%	92.39%	75.55%	73.47%	96.99%	683
512	2	0.0005	MLP	95.62%	97.20%	99.43%	70.47%	66.73%	96.68%	190.2
512	2	0.0005	CNN	90.48%	94.13%	96.42%	71.46%	66.68%	98.40%	691.7
1024	1	0.001	MLP	95.55%	97.16%	99.26%	66.35%	62.06%	95.88%	129.3
1024	1	0.001	CNN	95.53%	97.14%	99.45%	65.47%	59.41%	95.37%	454.2
1024	1	0.0005	MLP	95.39%	97.06%	99.27%	68.34%	65.36%	95.37%	130.7
1024	1	0.0005	CNN	95.73%	97.27%	99.60%	71.04%	68.14%	97.04%	458.9
1024	2	0.001	MLP	95.53%	97.15%	99.35%	70.42%	67.20%	96.39%	172.8
1024	2	0.001	CNN	95.78%	97.30%	99.71%	75.11%	72.96%	98.37%	704.8
1024	2	0.0005	MLP	95.56%	97.17%	99.32%	67.62%	63.84%	96.32%	175.5
1024	2	0.0005	CNN	95.80%	97.31%	99.74%	74.75%	73.17%	97.86%	711.2

Hasil lengkap *Grid Search* pada kondisi Non-IID ekstrem ($\alpha = 0.3$) ditunjukkan pada Tabel 4.11. Secara umum, hasil eksperimen menunjukkan bahwa *binary classification* relatif stabil pada hampir seluruh konfigurasi, dengan nilai *accuracy* dan *AUC-ROC* yang konsisten tinggi (di atas 95% dan mendekati 99%).

Sebaliknya, performa *multiclass* classification menunjukkan variasi yang lebih signifikan antar konfigurasi. Hal ini mengindikasikan bahwa klasifikasi *multiclass* pada kondisi Non-IID jauh lebih sensitif terhadap pemilihan *hyperparameter*, terutama pada kombinasi *Batch Size*, *Local Epochs*, dan arsitektur model.

Dari sisi arsitektur, model CNN cenderung menghasilkan performa *multiclass* yang lebih tinggi dibandingkan MLP, namun dengan konsekuensi waktu pelatihan yang lebih lama. Beberapa konfigurasi CNN dengan Local Epochs = 2 menunjukkan peningkatan akurasi *multiclass*, tetapi diiringi dengan peningkatan waktu training yang signifikan, sehingga kurang efisien untuk skenario terdistribusi.

Hasil *Grid Search* pada Kondisi IID (α = 5)

Tabel 4.12 Hasil *Grid Search* (α=5 – IID)

Batch Size	E	LR	Model	Binary Acc	Binary F1	Binary AUC	Multi Acc	Multi F1	Multi AUC	Train Time (s)
256	1	0.001	MLP	95.63%	97.20%	99.36%	76.21%	75.08%	97.81%	160.6
256	1	0.001	CNN	96.76%	97.85%	99.70%	60.57%	53.89%	91.04%	445.8
256	1	0.0005	MLP	95.68%	97.24%	99.35%	73.45%	71.81%	97.63%	164
256	1	0.0005	CNN	95.93%	97.39%	96.45%	78.07%	76.37%	98.04%	440.7
256	2	0.001	MLP	96.62%	97.76%	99.54%	77.09%	76.37%	97.88%	209.7
256	2	0.001	CNN	96.88%	97.92%	99.71%	61.90%	58.13%	92.31%	623.3
256	2	0.0005	MLP	96.58%	97.74%	99.53%	77.95%	77.42%	98.08%	212.4
256	2	0.0005	CNN	95.58%	97.17%	92.35%	57.47%	48.64%	89.82%	624
512	1	0.001	MLP	95.67%	97.24%	99.37%	75.24%	74.28%	97.73%	150.2
512	1	0.001	CNN	96.57%	97.72%	99.65%	59.67%	55.26%	94.23%	463
512	**1**	**0.0005**	**MLP**	**95.68%**	**97.24%**	**99.35%**	**73.30%**	**72.28%**	**96.45%**	**142.8**
512	**1**	**0.0005**	**CNN**	**96.67%**	**97.79%**	**99.62%**	**75.88%**	**74.61%**	**97.79%**	**457.3**
512	2	0.001	MLP	96.61%	97.76%	99.59%	79.05%	78.46%	98.06%	181.5
512	2	0.001	CNN	94.21%	96.08%	99.53%	40.25%	30.58%	71.32%	675
512	2	0.0005	MLP	95.70%	97.25%	99.61%	78.27%	77.70%	98.00%	181.2
512	2	0.0005	CNN	96.60%	97.74%	99.64%	77.13%	75.02%	97.69%	682.2
1024	1	0.001	MLP	95.69%	97.25%	99.36%	71.19%	67.82%	95.99%	126.8
1024	1	0.001	CNN	94.92%	96.58%	99.41%	56.81%	45.38%	93.36%	479.5
1024	1	0.0005	MLP	95.69%	97.25%	99.29%	69.26%	64.73%	95.24%	126.9
1024	1	0.0005	CNN	96.32%	97.55%	99.46%	61.15%	57.17%	93.94%	470.8
1024	2	0.001	MLP	95.69%	97.24%	99.48%	77.79%	77.16%	98.01%	168.7
1024	2	0.001	CNN	95.78%	97.30%	99.71%	75.11%	72.96%	98.37%	704.8
1024	2	0.0005	MLP	95.56%	97.17%	99.32%	67.62%	63.84%	96.32%	175.5

1024	2	0.0005	CNN	95.80%	97.31%	99.74%	74.75%	73.17%	97.86%	711.2

Tabel 4.12 menyajikan hasil *Grid Search* pada kondisi IID-like (α = 5.0). Dibandingkan dengan skenario Non-IID, performa *multiclass* classification pada kondisi ini meningkat secara konsisten di hampir seluruh konfigurasi. Hal ini menunjukkan bahwa distribusi data yang lebih homogen antar *client* mempermudah proses konvergensi model global.

Model CNN kembali menunjukkan keunggulan pada *task multiclass*, dengan beberapa konfigurasi mencapai akurasi di atas 80%. Namun demikian, pola trade-off antara performa dan biaya komputasi tetap terlihat, di mana peningkatan performa sering kali diiringi dengan waktu pelatihan yang lebih panjang.

Seleksi *Hyperparameter* Optimal

Berdasarkan pendekatan seleksi multi-kriteria yang telah dijelaskan pada anak subbab 3.6.4, setiap kombinasi hyperparameter hasil *limited Grid Search* dievaluasi menggunakan lima metrik utama, yaitu *Accuracy, Precision, Recall, F1-Score,* dan *AUC-ROC*, baik pada skenario klasifikasi biner maupun multi-kelas. Seluruh metrik tersebut diberikan bobot yang sama untuk menghindari dominasi satu metrik tertentu dalam proses seleksi.

Untuk setiap konfigurasi hyperparameter, nilai *Weighted Score* dihitung sebagai agregasi tertimbang dari kelima metrik evaluasi. Selanjutnya, evaluasi dilakukan secara terpisah pada *task* klasifikasi biner dan multi-kelas pada skenario Non-IID dengan parameter α = 0.3. Konfigurasi terbaik pada masing-masing *task* ditentukan berdasarkan nilai *Weighted Score* tertinggi.

Hasil perhitungan tersebut kemudian dikombinasikan untuk memperoleh *Balance Score*, yang digunakan sebagai indikator keseimbangan performa antara klasifikasi biner dan multi-kelas. *Balance Score* mempertimbangkan rata-rata performa kedua *task* serta penalti terhadap selisih performa antar *task*, sehingga konfigurasi yang unggul pada satu *task* namun lemah pada *task* lainnya tidak secara otomatis terpilih sebagai konfigurasi optimal. Ilustrasi proses seleksi hyperparameter berbasis multi-kriteria serta perbandingan nilai performa antar konfigurasi ditunjukkan pada Gambar 4.18.

```
# Metric weights
METRIC_WEIGHTS = {
    'accuracy': 0.20,
    'precision': 0.20,
```

```
    'recall': 0.20,
    'f1_score': 0.20,
    'auc_roc': 0.20
}

# Filter Non-IID alpha = 0.3
df_alpha_03 = df[df['alpha'] == 0.3].copy()

# Compute weighted score
df_alpha_03['weighted_score'] = (
    df_alpha_03['accuracy']  * METRIC_WEIGHTS['accuracy'] +
    df_alpha_03['precision'] * METRIC_WEIGHTS['precision'] +
    df_alpha_03['recall']    * METRIC_WEIGHTS['recall'] +
    df_alpha_03['f1_score']  * METRIC_WEIGHTS['f1_score'] +
    df_alpha_03['auc_roc']   * METRIC_WEIGHTS['auc_roc']
)

# Best configuration per task
best_binary = (
    df_alpha_03[df_alpha_03['classification'] == 'Binary']
    .nlargest(1, 'weighted_score')
)

best_multiclass = (
    df_alpha_03[df_alpha_03['classification'] == 'Multiclass']
    .nlargest(1, 'weighted_score')
)

# Top-k configurations (optional, for analysis)
top10_binary = (
    df_alpha_03[df_alpha_03['classification'] == 'Binary']
    .nlargest(10, 'weighted_score')
)

top10_multiclass = (
    df_alpha_03[df_alpha_03['classification'] == 'Multiclass']
    .nlargest(10, 'weighted_score')
)
```

Gambar 4.18 Pemilihan *Hyperparameter* Berdasarkan Balance Score

Konfigurasi dengan nilai *Balance Score* tertinggi dipilih sebagai konfigurasi optimal karena menunjukkan performa yang relatif seimbang antara klasifikasi biner dan multi-kelas, serta tidak mengalami degradasi signifikan pada salah satu *task*. Konfigurasi terpilih ini

selanjutnya digunakan sebagai dasar evaluasi lanjutan pada algoritma Federated Averaging (FedAvg) dan Federated Proximal (FedProx).

Hasil ini menunjukkan bahwa pendekatan seleksi berbasis multi-kriteria lebih representatif dibandingkan pemilihan hyperparameter berdasarkan satu metrik tunggal, khususnya pada skenario *Federated Learning* dengan tingkat heterogenitas data yang tinggi, di mana performa model dapat berbeda secara signifikan antar *task* klasifikasi.

Konfigurasi Terpilih

Berdasarkan analisis Cost-Benefit dan prinsip Pareto Optimal, konfigurasi terbaik pada kondisi Non-IID ($\alpha = 0.3$) dipilih sebagai berikut:

a. Batch Size: [*512*]
b. Local Epochs: [*1*]
c. Learning Rate: [*0.0005*]
d. FedProx μ: [*0.01, 0.001*] (khusus skenario FedProx)

Konfigurasi ini memberikan Balance Score tertinggi dengan waktu pelatihan yang relatif efisien, yaitu sekitar 447,4 detik (~7,5 menit) untuk CNN dan 141,5 detik (~2,4 menit) untuk *MLP*.

Pada konfigurasi terpilih, performa *binary classification* mencapai Accuracy 95.71%, F1-Score 97.26%, dan AUC-ROC 99.59%, menunjukkan kemampuan deteksi serangan yang sangat baik. Sementara itu, pada *multiclass* classification, model mencapai Accuracy 73.27%, F1-Score 70.99%, dan AUC-ROC 97.64%, meskipun dihadapkan pada ketimpangan kelas yang signifikan, khususnya pada kelas MITM yang hanya memiliki 834 sampel.

4.4 Analisis *Federated Learning* dari Eksperimen

Konfigurasi terpilih di atas menjadi landasan (*baseline*) yang digunakan untuk seluruh skenario pengujian yang dibahas pada bagian selanjutnya. Meskipun hasil awal menunjukkan performa yang menjanjikan, evaluasi lebih lanjut sangat krusial untuk mengidentifikasi cara model merespons kompleksitas data jaringan yang bersifat heterogen. Subbab berikut akan menguraikan analisis komparatif antara arsitektur MLP dan CNN, serta menguji batas kemampuan algoritma FedAvg dan FedProx dalam menjaga stabilitas model global. Fokus utama analisis ditekankan pada kemampuan sistem mempertahankan deteksi serangan minoritas di tengah kondisi data *imbalance* yang ekstrem. Pembahasan akan dimulai dari

gambaran performa secara umum menggunakan metrik global sebelum masuk ke analisis mendalam pada level kelas serangan.

4.4.1 Gambaran Umum Hasil Eksperimen (*Global Metrics*)

Subbab ini menyajikan ringkasan performa global dari seluruh konfigurasi eksperimen *Federated Learning* (FL) yang diuji, mencakup variasi arsitektur model (MLP dan CNN), variasi distribusi data (IID dan Non-IID), serta variasi algoritma agregasi (FedAvg dan FedProx) dengan variasi parameter proximal (μ). Evaluasi global menggunakan metrik Accuracy, Precision, Recall, F1-Score, dan AUC-ROC untuk menggambarkan kualitas model secara agregat.

Secara umum, pada skenario IID performa klasifikasi biner cenderung mencapai kondisi jenuh (ceiling effect) sehingga perbedaan antar algoritma tidak terlalu menonjol. Sebaliknya, pada skenario *multiclass*, perbedaan arsitektur dan algoritma lebih terlihat karena model perlu membedakan pola serangan yang lebih beragam dan memiliki kemiripan fitur antar kelas.

Ringkasan metrik global untuk skenario IID disajikan pada Tabel 4.13 menunjukkan bahwa konfigurasi CNN *Multiclass* dengan FedProx (μ=0.001) memberikan performa terbaik pada IID (Accuracy 0.7861 dan F1-Score 0.7802), sedangkan tugas biner relatif setara pada seluruh algoritma.

Tabel 4.13 Perbandingan Performa Algoritma (Global Metrics) pada Skenario IID

Model	Algoritma	Accuracy	Precision	Recall	F1 Score	AUC-ROC
MLP Binary	FedAvg	0.9569	0.9487	0.9974	0.9725	0.9929
MLP Binary	FedProx (μ=0.01)	0.9569	0.9487	0.9974	0.9725	0.9933
MLP Binary	FedProx (μ=0.001)	0.9569	0.9487	0.9974	0.9725	0.9931
CNN Binary	FedAvg	0.9698	0.9916	0.9686	0.98	0.9968
CNN Binary	FedProx (μ=0.01)	0.9687	0.9928	0.966	0.9792	0.9969
CNN Binary	FedProx (μ=0.001)	0.966	0.9918	0.9634	0.9774	0.9962
MLP Multi	FedAvg	0.7111	0.7462	0.7111	0.6757	0.9582
MLP Multi	FedProx (μ=0.01)	0.7027	0.7292	0.7027	0.6816	0.9647
MLP Multi	FedProx (μ=0.001)	0.7036	0.7389	0.7036	0.6815	0.9745
CNN Multi	FedAvg	0.7778	0.8018	0.7778	0.7664	0.985
CNN Multi	FedProx (μ=0.01)	0.7718	0.8083	0.7718	0.7649	0.9815
CNN Multi	FedProx (μ=0.001)	0.7861	0.8192	0.7861	0.7802	0.9817

Analisis per kelas pada skenario IID untuk konfigurasi CNN *Multiclass* dirangkum pada Tabel 4.14menunjukkan bahwa peningkatan terbesar pada kelas DoS terjadi saat menggunakan FedProx (μ=0.01), sementara performa Injection justru terbaik pada FedProx (μ=0.001). Hal ini menegaskan bahwa dampak regularisasi dapat berbeda antar kelas.

Tabel 4.14 Detail Per Kelas (F1-Score) untuk CNN *Multiclass* pada Skenario IID

Kelas Serangan	**FedAvg (F1)**	**FedProx (μ=0.01) (F1)**	**FedProx (μ=0.001) (F1)**	**Tren Terbaik**
Backdoor	0.63	0.65	0.63	FedProx μ=0.01
DDoS	0.94	0.94	0.94	Stabil
DoS	0.25	0.37	0.34	FedProx (naik signifikan)
Injection	0.77	0.64	0.79	FedProx μ=0.001
MITM	0.26	0.19	0.1	FedAvg (sulit dideteksi)
Normal	0.92	0.93	0.93	FedProx
Password	0.73	0.66	0.72	FedAvg
Ransomware	0.76	0.79	0.79	FedProx
Scanning	0.8	0.81	0.8	FedProx μ=0.01
XSS	0.88	0.88	0.88	Stabil

Dari sisi efisiensi, Tabel 4.15menyajikan perbandingan waktu pelatihan pada skenario IID.

Tabel 4.15 memperlihatkan bahwa FedProx menambah *overhead* waktu dibanding FedAvg karena komputasi proximal term. Pada IID, *overhead* ini tidak selalu sebanding untuk tugas biner (karena performa sudah jenuh), tetapi dapat dipertimbangkan pada CNN *Multiclass* ketika terjadi peningkatan kinerja sebagaimana pada Tabel 4.13.

Tabel 4.15 Lama Waktu Pelatihan (Training Cost) pada Skenario IID

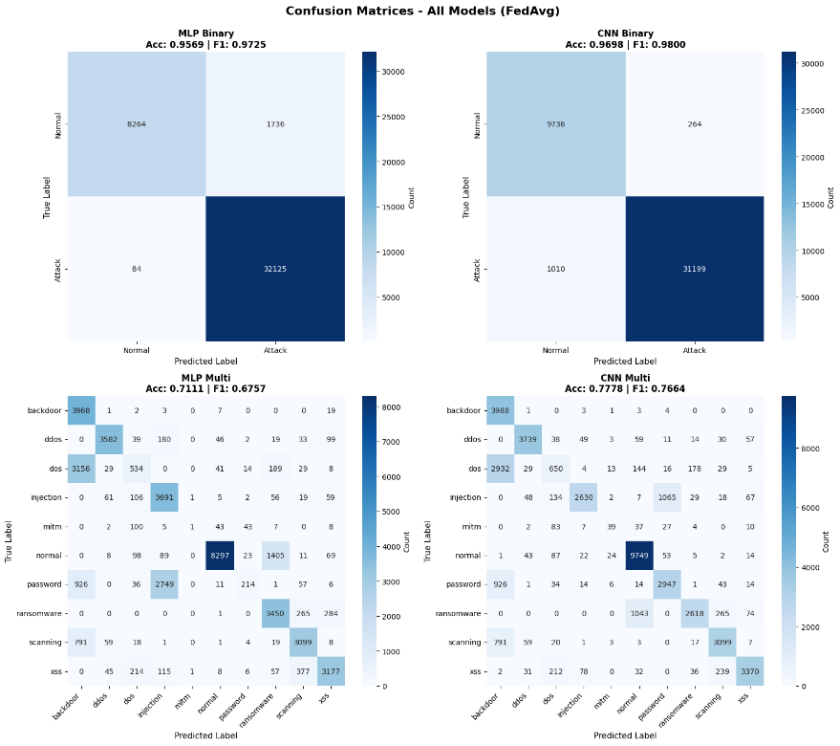
Algoritma	**Model**	**Waktu Total (detik)**	**Rata-rata per Round (detik)**	**Keterangan**
FedAvg	MLP Binary	62.93	3.11	Tercepat
FedAvg	CNN Binary	160.82	7.97	-
FedAvg	MLP Multi	88.37	4.37	-
FedAvg	CNN Multi	305.22	15.1	-
FedProx (μ=0.01)	MLP Binary	87.38	4.3	Lebih lambat ~38% vs FedAvg
FedProx (μ=0.01)	CNN Binary	185.51	9.17	Lebih lambat ~15% vs FedAvg
FedProx (μ=0.01)	MLP Multi	115.19	5.67	-
FedProx (μ=0.01)	CNN Multi	327.97	16.2	-
FedProx (μ=0.001)	MLP Binary	86.39	4.25	Mirip μ=0.01

FedProx (μ=0.001)	CNN Binary	185.18	9.14	-
FedProx (μ=0.001)	MLP Multi	118.68	5.85	-
FedProx (μ=0.001)	CNN Multi	330.25	16.31	-

4.4.2 Analisis Performa Arsitektur Model: MLP vs CNN Skenario Standar

Skenario *baseline* pada pembahasan ini didefinisikan sebagai FedAvg dengan distribusi data IID, karena kondisi tersebut merepresentasikan situasi ideal ketika data antar *client* relatif seragam dan algoritma agregasi standar umumnya sudah memadai. Analisis *baseline* berfokus pada pengaruh arsitektur model terhadap kemampuan deteksi intrusi.

Berdasarkan Tabel 4.13 baris FedAvg), CNN menunjukkan keunggulan konsisten dibanding MLP, terutama pada tugas *multiclass classification*. CNN *Multiclass* mencapai Accuracy 0.7778 dan F1-Score 0.7664, lebih tinggi daripada MLP *Multiclass* (Accuracy 0.7111 dan F1-Score 0.6757). Temuan ini mengindikasikan bahwa CNN lebih efektif mengekstraksi fitur diskriminatif untuk membedakan berbagai kelas serangan.



Gambar 4.19 Confusion Matrix FedAvg pada Skenario IID (MLP Binary, CNN Binary, MLP *Multiclass*, CNN *Multiclass*)

Pada Gambar 4.19 klasifikasi biner memperlihatkan perbedaan karakter keputusan model: MLP cenderung lebih agresif mendeteksi serangan (False Negative kecil) namun menghasilkan False Positive lebih tinggi sehingga meningkatkan potensi false alarm. Sebaliknya, CNN biner lebih seimbang karena lebih baik mengidentifikasi trafik normal. Pada *multiclass*, CNN menunjukkan prediksi kelas Normal yang lebih bersih dan peningkatan akurasi pada kelas tertentu misalnya XSS dan Scanning. Meskipun masih terdapat kebingungan konsisten antara DoS dan Backdoor.

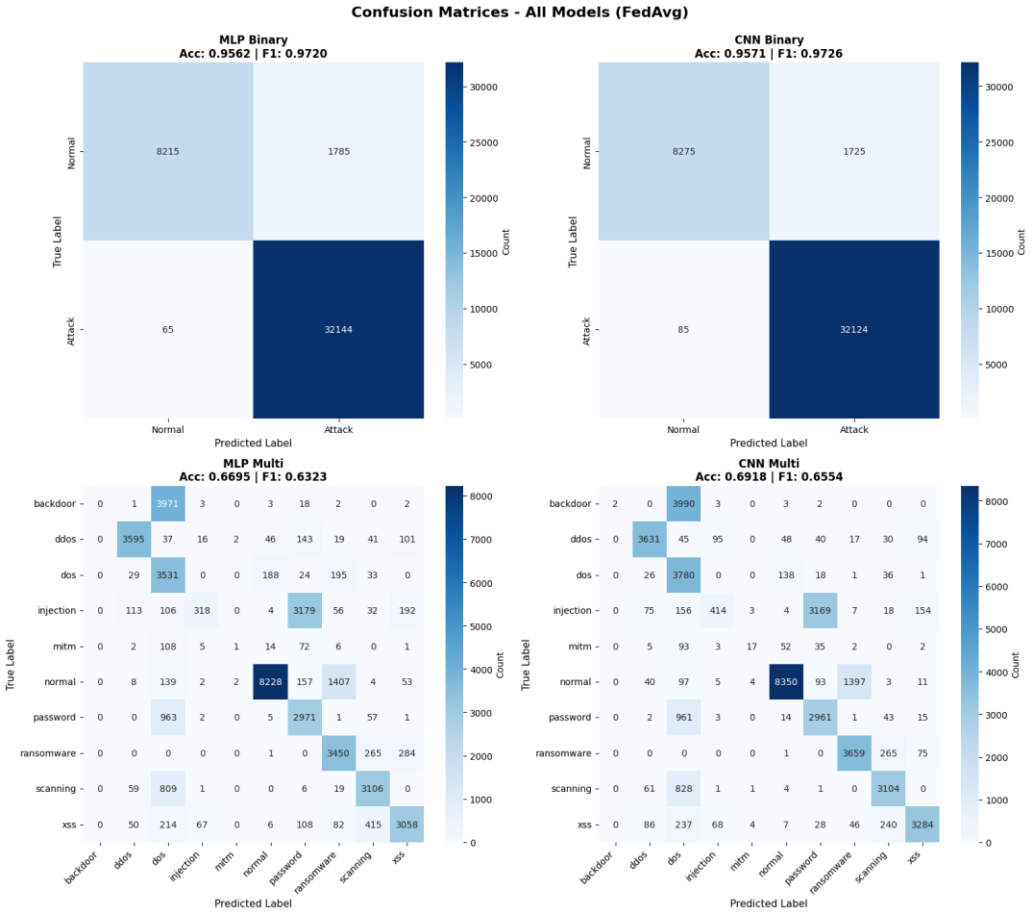
4.4.3 Dampak Heterogenitas Data: IID vs Non-IID

Heterogenitas data merupakan tantangan utama dalam FL karena distribusi kelas pada tiap *client* dapat berbeda. Pada skenario IID, *update* lokal cenderung memiliki arah yang selaras sehingga model global stabil. Pada skenario Non-IID (Dirichlet α=0.3), sebagian *client* dapat didominasi oleh kelas tertentu sehingga *update* lokal dapat saling bertabrakan dan model global berisiko bias terhadap kelas mayoritas.

Penurunan performa akibat heterogenitas terlihat jelas pada tugas *multiclass classification*. Perbandingan metrik global Non-IID ditunjukkan pada Tabel 4.15. Pada CNN *Multiclass*, FedAvg turun dari Accuracy 0.7778 (IID; Tabel 4.13) menjadi 0.6918 (Non-IID; Tabel 4.16), yang mengindikasikan degradasi kemampuan generalisasi ketika distribusi data antar *client* timpang.

Tabel 4.16 Perbandingan Performa Algoritma (Global Metrics) pada Skenario Non-IID (Dirichlet α=0.3)

Model	Algoritma	Accuracy	Precision	Recall	F1 Score	AUC-ROC
MLP Binary	FedAvg	0.9562	0.9474	0.998	0.972	0.9927
MLP Binary	FedProx (μ=0.01)	0.9561	0.9474	0.9978	0.972	0.9934
MLP Binary	FedProx (μ=0.001)	0.9562	0.9474	0.998	0.972	0.9932
CNN Binary	FedAvg	0.9571	0.949	0.9974	0.9726	0.9961
CNN Binary	FedProx (μ=0.01)	0.9574	0.9486	0.9982	0.9728	0.996
CNN Binary	FedProx (μ=0.001)	0.9578	0.9491	0.9983	0.9731	0.9959
MLP Multi	FedAvg	0.6695	0.683	0.6695	0.6323	0.9604
MLP Multi	FedProx (μ=0.01)	0.6976	0.7353	0.6976	0.671	0.9614
MLP Multi	FedProx (μ=0.001)	0.6686	0.6868	0.6686	0.6317	0.9578
CNN Multi	FedAvg	0.6918	0.7923	0.6918	0.6554	0.9742
CNN Multi	FedProx (μ=0.01)	0.7304	0.7724	0.7304	0.7143	0.9734
CNN Multi	FedProx (μ=0.001)	0.6937	0.7924	0.6937	0.6547	0.976



Gambar 4.20 Confusion Matrix FedAvg pada Skenario Non-IID (MLP Binary, CNN Binary, MLP *Multiclass*, CNN *Multiclass*)

Gambar 4.20 menunjukkan bahwa klasifikasi biner tetap relatif robust pada Non-IID karena hanya memisahkan dua kelas. Namun, pada *multiclass* terjadi kegagalan lebih serius: kelas Backdoor hampir tidak terdeteksi mendekati 0%, serta kebingungan kuat antara Injection dan Password. Fenomena ini konsisten dengan client *drift* dan *catastrophic forgetting*, yaitu pengetahuan kelas minoritas/langka tertimpa oleh *update* dari *client* yang tidak memiliki kelas tersebut.

4.4.4 Analisis Efektivitas Algoritma: FedAvg vs FedProx

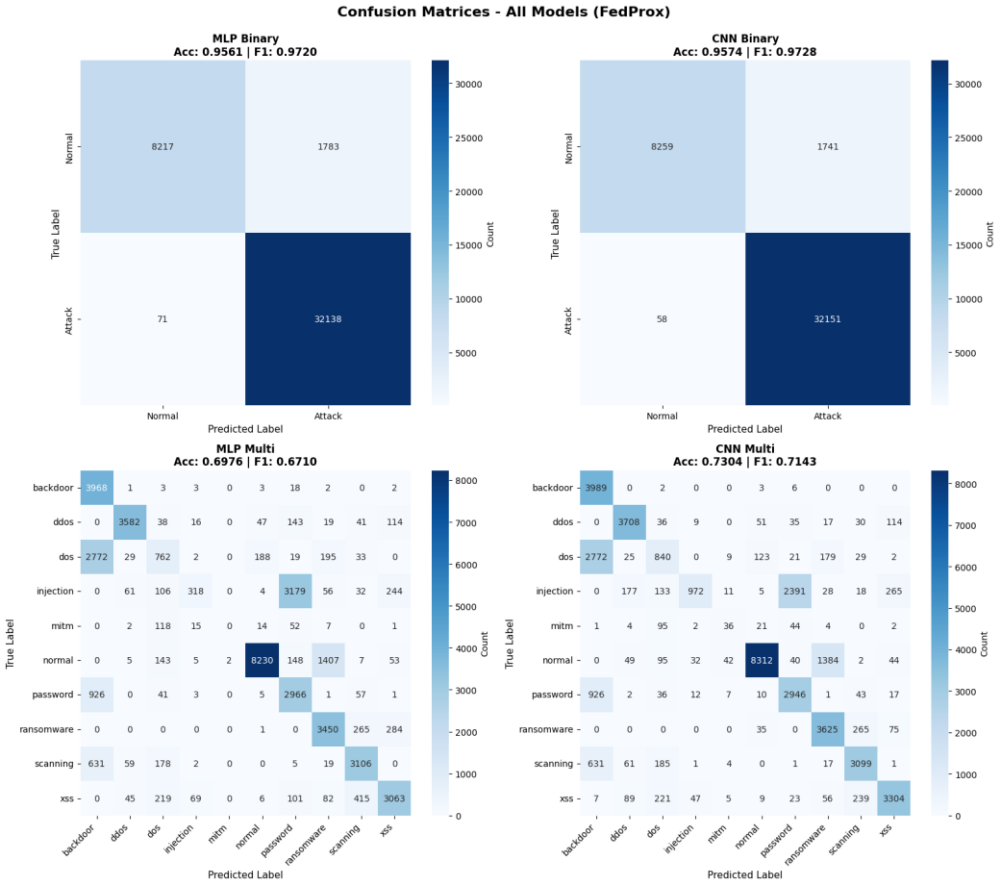
FedAvg merupakan metode agregasi standar yang menghitung rata-rata bobot dari *client*. Pada kondisi IID, FedAvg cenderung stabil karena *update client* relatif homogen. Akan tetapi, pada Non-IID, FedAvg rentan bias terhadap *client* mayoritas sehingga pengetahuan kelas minoritas dapat tenggelam. FedProx memperbaiki hal tersebut dengan menambahkan proximal

term (μ) yang membatasi penyimpangan model lokal terhadap model global, sehingga menekan client *drift*.

Efektivitas FedProx pada Non-IID tercermin pada Tabel 4.15, khususnya CNN *Multiclass*: FedProx (μ=0.01) meningkatkan Accuracy dari 0.6918 (FedAvg) menjadi 0.7304 dan F1-Score dari 0.6554 menjadi 0.7143. Bukti per kelas pada Tabel 4.16 menunjukkan bahwa peningkatan ini terutama berasal dari pemulihan kelas minoritas seperti Backdoor dan Injection.

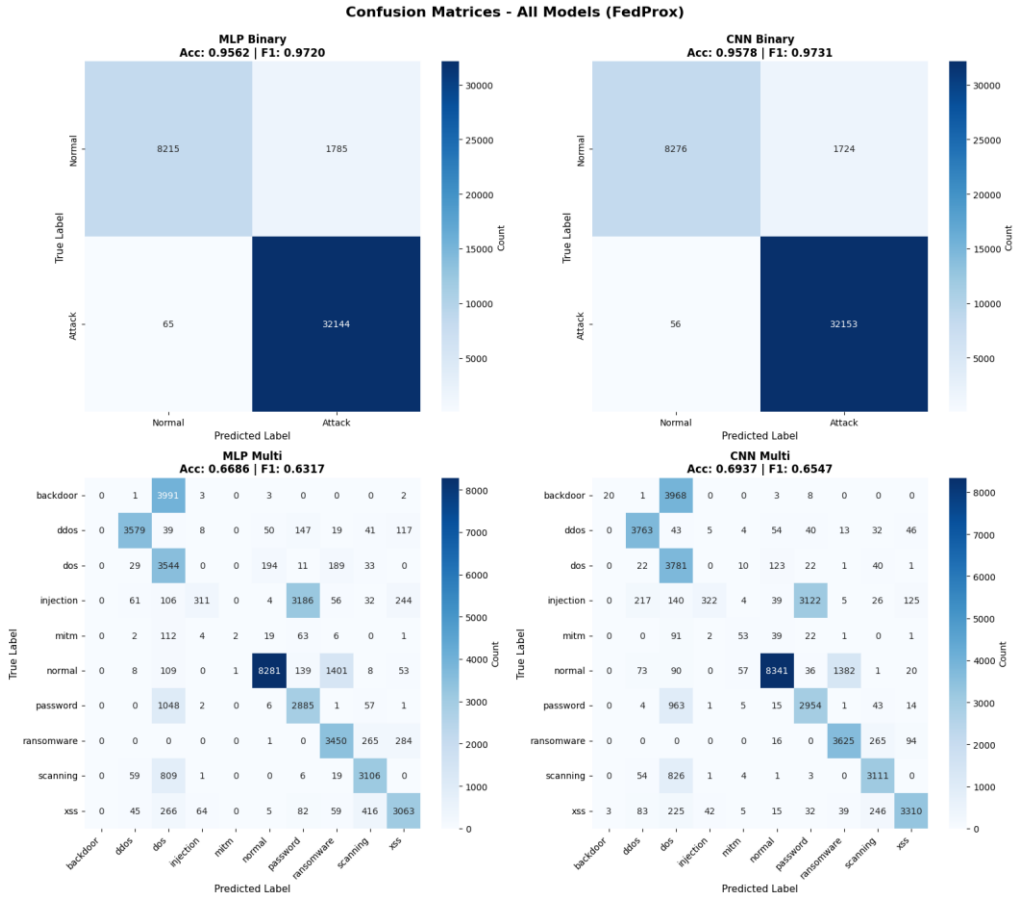
Tabel 4.17 Detail Per Kelas (F1-Score) untuk CNN *Multiclass* pada Skenario Non-IID (Dirichlet α=0.3)

Kelas Serangan	**FedAvg (F1)**	**FedProx (μ=0.01) (F1)**	**FedProx (μ=0.001) (F1)**	**Keterangan**
Backdoor	0.0	0.65	0.01	Peningkatan masif pada μ=0.01
DDoS	0.92	0.91	0.92	Stabil
DoS	0.53	0.3	0.54	FedAvg/μ=0.001 sedikit lebih baik
Injection	0.18	0.38	0.15	μ=0.01 membaik
MITM	0.14	0.22	0.3	Sulit dideteksi
Normal	0.9	0.9	0.89	Stabil
Password	0.57	0.62	0.58	μ=0.01 membaik
Ransomware	0.8	0.78	0.8	Stabil
Scanning	0.8	0.8	0.8	Stabil
XSS	0.86	0.84	0.87	Stabil



Gambar 4.21 *Confusion Matrix* FedProx (μ=0.01) pada Skenario Non-IID

Pada Gambar 4.21 *Confusion Matrix* FedProx (μ=0.01) pada Skenario Non-IID, FedProx (μ=0.01) memperlihatkan pemulihan deteksi Backdoor yang sebelumnya hampir hilang pada FedAvg Non-IID. Meski terdapat efek samping berupa kebingungan terbalik antara DoS dan Backdoor, konfigurasi ini lebih aman untuk IDS karena lebih baik salah mengklasifikasikan tipe serangan dibanding gagal mendeteksi serangan langka sama sekali (Bierbrauer dkk., 2025).



Gambar 4.22 *Confusion Matrix* FedProx (μ=0.001) pada Skenario Non-IID

Gambar 4.22 menunjukkan bahwa FedProx (μ=0.001) tidak cukup kuat pada Non-IID berat. Pola misklasifikasi kembali mirip FedAvg, khususnya kegagalan mendeteksi kelas minoritas seperti Backdoor dan rendahnya performa Injection. Hal ini memperkuat argumen bahwa pada distribusi data sangat heterogen, diperlukan μ lebih besar untuk menahan *drift* model lokal.

Dari sisi biaya komputasi pada Non-IID, Tabel 4.18 menunjukkan bahwa FedProx menambah *overhead* waktu dibanding FedAvg. Namun, pada konteks Non-IID *overhead* ini dapat dipertimbangkan sebagai investasi, karena peningkatan performa *multiclass* (khususnya pada CNN *Multiclass*) bersifat substantif sebagaimana terlihat pada Tabel 4.16 dan Tabel 4.17.

Tabel 4.18 Lama Waktu Pelatihan (*Training Cost*) pada Skenario Non-IID

Algoritma	**Model**	**Waktu Total (detik)**	**Rata-rata per Round (detik)**	***Overhead* vs FedAvg**
FedAvg	MLP Binary	63.17	3.12	-

FedAvg	CNN Binary	151.44	7.51	-
FedAvg	MLP Multi	87.43	4.32	-
FedAvg	CNN Multi	304.76	15.08	-
FedProx (μ=0.01)	MLP Binary	89.42	4.4	+41%
FedProx (μ=0.01)	CNN Binary	179.92	8.87	+18%
FedProx (μ=0.01)	MLP Multi	108.77	5.35	+24%
FedProx (μ=0.01)	CNN Multi	333.27	16.46	+9%
FedProx (μ=0.001)	MLP Binary	89.35	4.39	+41%
FedProx (μ=0.001)	CNN Binary	179.98	8.87	+18%
FedProx (μ=0.001)	MLP Multi	111.97	5.51	+28%
FedProx (μ=0.001)	CNN Multi	325.66	16.08	+6%

4.4.5 Analisis Sensitivitas Parameter Proximal μ

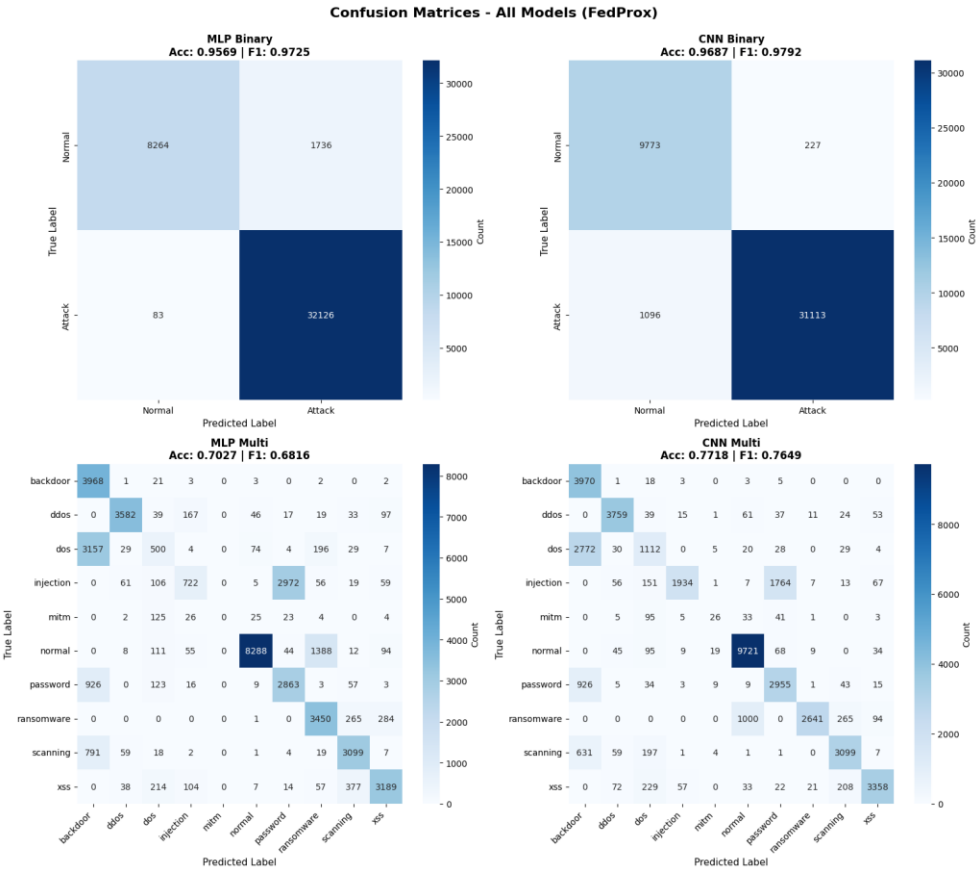
Analisis sensitivitas parameter proximal (μ) menjelaskan cara kekuatan regularisasi FedProx memengaruhi stabilitas dan fleksibilitas pembelajaran. Secara intuitif, μ berperan sebagai *redline* antara model lokal dan model global: μ kecil memberi fleksibilitas lebih besar pada pembelajaran lokal, sedangkan μ besar menahan *update* lokal agar tidak terlalu menyimpang.

Rangkuman perilaku μ pada dua kondisi distribusi data disajikan pada Tabel 4.19 menegaskan bahwa pemilihan μ bersifat kontekstual: μ kecil cenderung cocok untuk IID karena menghindari *over-regularization*, sementara μ lebih besar diperlukan pada Non-IID untuk menahan *client drift* dan memulihkan kelas minoritas.

Tabel 4.19 Perilaku FedProx terhadap Variasi Parameter Proximal (μ) pada Skenario IID dan Non-IID

Skenario Data	**Perilaku μ Kecil (0.001)**	**Perilaku μ Sedang/Besar (0.01)**

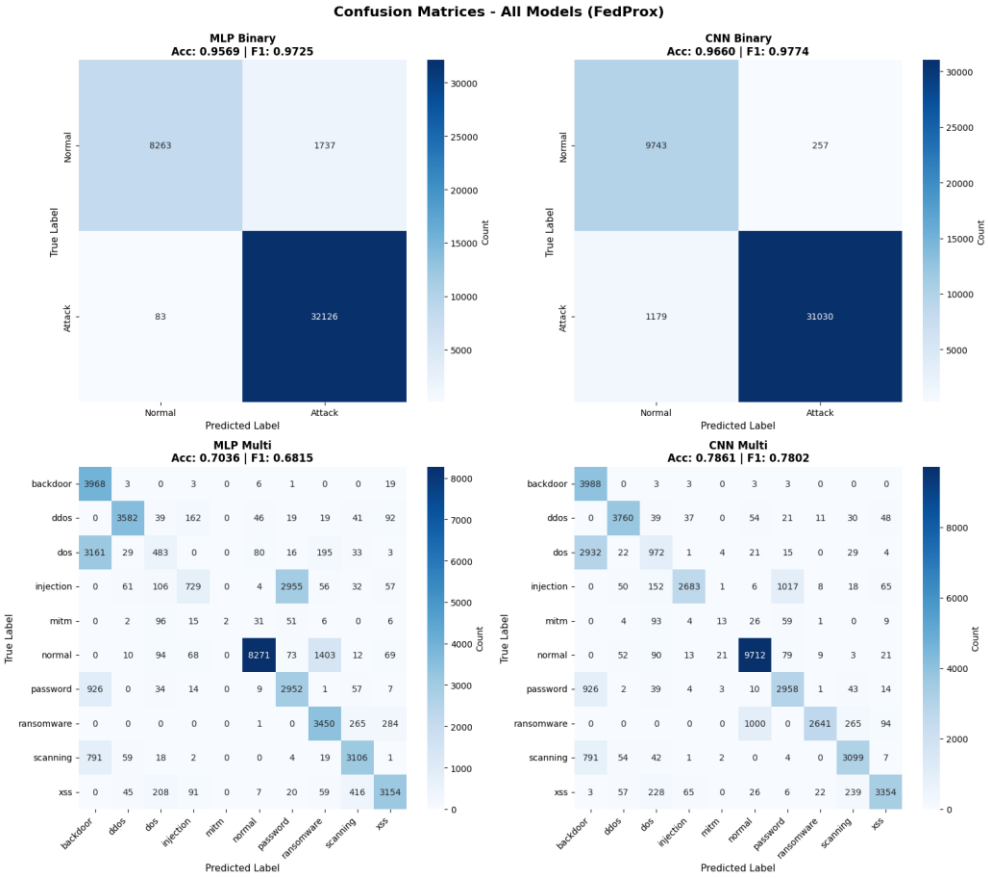
IID (Seragam)	Efektif sebagai regularisasi ringan; membantu generalisasi tanpa menghambat pembelajaran fitur lokal yang relevan. Hasil optimal terlihat pada CNN *Multiclass*.	Berpotensi terlalu kaku (over-regularization); pada beberapa kelas dapat menurunkan performa karena model lokal dipaksa terlalu mirip global, padahal data lokal sudah representatif.
Non-IID (Timpang)	Gagal/terlalu lemah (*too weak*); penalti kecil sehingga model lokal tetap divergen mengikuti bias data lokal, menghasilkan performa mirip FedAvg.	Sangat efektif; mampu menahan client *drift* serta memulihkan pembelajaran kelas minoritas/langka (misalnya Backdoor) yang hilang pada FedAvg.



Gambar 4.23 *Confusion Matrix* FedProx (μ=0.01) pada Skenario IID

Berdasarkan Gambar 4.23, pada data IID penggunaan μ=0.01 cenderung menghasilkan regularisasi yang terlalu kuat untuk sebagian kelas *multiclass*. Dampaknya, beberapa kelas

(misalnya DoS) dapat membaik, namun kelas Injection cenderung menurun karena model menjadi kurang fleksibel dalam mempelajari pemisahan fitur yang halus.



Gambar 4.24 *Confusion Matrix* FedProx (μ=0.001) pada Skenario IID

Gambar Gambar 4.24 menunjukkan bahwa μ=0.001 pada IID menjadi titik keseimbangan (sweet spot) karena regularisasi ringan mampu meningkatkan generalisasi tanpa menghambat pembelajaran. Hal ini konsisten dengan Tabel 4.13 (CNN *Multiclass* mencapai Accuracy 0.7861) dan Tabel 4.14 yang menunjukkan pemulihan performa Injection.

4.4.6 Analisis Pola Misklasifikasi dan Dampak Keamanan

Selain metrik performa agregat, analisis mendalam terhadap pola kesalahan klasifikasi (*misclassification patterns*) pada *confusion matrix* mengungkapkan fenomena "kebingungan" model yang konsisten terjadi, baik pada algoritma FedAvg maupun FedProx. Kebingungan ini ditandai dengan tingginya tingkat *false positive* atau *false negative* pada pasangan kelas serangan tertentu.

a. Fenomena Kebingungan Backdoor dan DoS

Sebagaimana terlihat pada Gambar 4.21, algoritma FedProx $\mu = 0.01$ berhasil memulihkan deteksi kelas *Backdoor* yang sebelumnya gagal dikenali oleh FedAvg. Namun, keberhasilan ini disertai efek samping berupa peningkatan misklasifikasi timbal balik (*reciprocal misclassification*) antara kelas *backdoor* dan *DoS*. Sebagian trafik *backdoor* diprediksi sebagai *DoS*, dan sebaliknya.

Fenomena ini disebabkan oleh kemiripan karakteristik fitur statistik tertentu pada level jaringan. Serangan *backdoor* modern sering kali melibatkan komunikasi *keep-alive* yang repetitif atau transfer data yang terselubung, yang secara pola trafik (durasi, interval paket) dapat menyerupai karakteristik serangan *DoS* tipe *slow-rate* atau *flooding* skala kecil (Jin dkk., 2025). Regularisasi *proximal term* pada FedProx memaksa model untuk menangkap fitur yang generalisasi dari kedua kelas tersebut, sehingga batas keputusan (*decision boundary*) menjadi kurang tajam dalam memisahkan keduanya secara spesifik.

b. Fenomena Kebingungan Injection dan Password

Pola kebingungan juga teramati secara konsisten antara kelas *Injection* dan *Password*. Model sering kali kesulitan membedakan kedua jenis serangan ini, terutama pada skenario *Non-IID*. Hal ini kemungkinan besar disebabkan oleh kesamaan vektor serangan yang beroperasi pada lapisan aplikasi (*Layer* 7) protokol HTTP (Quincozes dkk., 2021). Kedua serangan ini sering kali melibatkan manipulasi parameter pada *request* HTTP misalnya pada penyisipan kode SQL pada *injection* atau percobaan *brute-force* kredensial pada *password attack.*

c. Kegagalan Deteksi MITM

Berbeda dengan dua kasus di atas yang merupakan kebingungan antar-serangan (*Inter-Attack Confusion*), pola yang paling kritis adalah kegagalan deteksi pada serangan *Man-in-the-Middle* (MITM). Seperti ditunjukkan pada Tabel 4.17, kelas MITM memiliki F1-Score yang sangat rendah (0.14 - 0.30), yang berarti sebagian besar serangan MITM salah diklasifikasikan sebagai trafik *normal*.

Penyebab utamanya adalah karakteristik serangan MITM seperti ARP *spoofing* yang sering kali bekerja secara pasif atau menyisipkan diri ke dalam sesi komunikasi yang sah tanpa mengubah pola statistik trafik secara drastis seperti volume atau durasi (Rehman dkk., 2025). Akibatnya, fitur-fitur yang digunakan model gagal menangkap anomali halus tersebut.

4.4.7 Analisis Pola Misklasifikasi dan Dampak Keamanan

Meskipun penggunaan algoritma FedProx telah menunjukkan peningkatan stabilitas pada kondisi data yang tidak seragam, penelitian ini mengidentifikasi limitasi fundamental terkait penerapan satu model global (*global model*) dalam ekosistem *smart city* yang heterogen. Implementasi satu model tunggal untuk seluruh client memiliki keterbatasan dalam menangani spesifisitas serangan yang bersifat unik pada tiap kategori perangkat IoT.

Keterbatasan ini muncul karena model global merupakan hasil agregasi rata-rata atau konsensus dari berbagai pembaruan bobot lokal . Akibatnya, model tersebut cenderung menjadi terlalu general dan kurang sensitif terhadap variasi serangan spesifik yang hanya muncul pada jenis perangkat tertentu. Sebagai contoh, kerentanan *Cross-Site Scripting* (XSS) yang sangat spesifik dan kritikal pada perangkat *smart meter* berbasis antarmuka web tidak akan relevan bagi sensor lingkungan yang hanya menggunakan protokol komunikasi ringan berbasis MQTT. Dalam proses agregasi *Federated Learning*, pembaruan bobot dari *client* yang mendeteksi XSS akan terdelusi oleh pembaruan dari banyak *client* lain yang tidak memiliki karakteristik serangan tersebut.

Implikasi teknis dari limitasi ini terlihat jelas pada sulitnya model dalam mendeteksi kelas minoritas dalam skenario Non-IID ekstrem $\alpha = 0.3$. Sebagaimana ditunjukkan pada hasil eksperimen di subbab 4.4.6 sebelumnya Gambar 4.17, serangan seperti MITM dan *backdoor* sering kali tenggelam dalam konsensus global karena tidak semua *gateway* IoT merepresentasikan pola trafik serangan tersebut secara konsisten. Oleh karena itu, pendekatan satu model global untuk semua ragam IoT yang diklasifikasikan sama berisiko menurunkan akurasi deteksi pada tingkat individual perangkat, yang pada akhirnya dapat meninggalkan celah keamanan pada unit IoT tertentu dalam jaringan *smart city*.

4.5 Dokumentasi Hasil Penelitian

Seluruh kode sumber dan artefak penelitian ini didokumentasikan menggunakan sistem *version control* pada platform GitHub. Hal ini bertujuan untuk menjamin transparansi, memudahkan penelusuran riwayat pengembangan, serta mendukung reprodusibilitas hasil eksperimen. Dokumentasi teknis beserta panduan penggunaan lengkap dapat diakses melalui tautan berikut: https://github.com/elnoersan/NIDS-FL.

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil eksperimen yang telah dilakukan, dapat disimpulkan bahwa karakteristik lalu lintas jaringan pada lingkungan *smart city* yang bersifat heterogen menyebabkan distribusi data antar *gateway* cenderung Non-IID. Kondisi ini berdampak langsung pada performa sistem *Network Intrusion Detection System* (NIDS) *berbasis Federated Learning*, terutama pada skenario klasifikasi *multiclass* yang melibatkan kelas serangan langka. Temuan ini menunjukkan bahwa evaluasi performa tidak dapat hanya mengandalkan metrik agregat global, tetapi perlu mempertimbangkan perilaku model terhadap distribusi data yang tidak merata antar *client*.

Hasil perbandingan algoritma menunjukkan bahwa *Federated Averaging* (FedAvg) memiliki performa yang stabil dan efisien pada tugas deteksi biner, baik pada kondisi IID maupun Non-IID. Namun, pada skenario *multiclass* dengan heterogenitas data yang tinggi (α=0.3), FedAvg cenderung mengalami degradasi performa yang signifikan pada beberapa kelas minoritas, seperti *backdoor* dan MITM, yang pada beberapa konfigurasi tidak berhasil dipelajari sama sekali. Hal ini mengindikasikan bahwa pendekatan agregasi standar berpotensi mengabaikan informasi penting yang hanya muncul pada sebagian kecil *client*.

Penerapan *Federated Proximal* (FedProx) dengan parameter $\mu = 0.01$ menunjukkan kemampuan yang lebih baik dalam menjaga stabilitas pembelajaran pada kondisi Non-IID. Secara khusus, FedProx mampu memulihkan performa pada kelas serangan yang sebelumnya gagal dipelajari oleh FedAvg, meskipun dengan konsekuensi peningkatan waktu pelatihan. Temuan ini menegaskan bahwa mekanisme regularisasi proximal efektif dalam mengurangi *client drift*, terutama ketika distribusi data antar *client* sangat tidak seimbang.

Dari sisi arsitektur model, CNN 1D secara konsisten menunjukkan performa yang lebih baik dibandingkan MLP pada tugas klasifikasi *multiclass*, terutama dalam menangkap pola kompleks pada lalu lintas jaringan. Namun demikian, peningkatan performa tersebut disertai dengan beban komputasi yang lebih tinggi, sehingga pemilihan model perlu disesuaikan dengan keterbatasan sumber daya pada perangkat *edge*. Untuk kebutuhan deteksi biner yang lebih sederhana, MLP dengan FedAvg tetap menjadi alternatif yang rasional karena efisiensi dan kestabilannya.

Selanjutnya, dari analisis mendalam terhadap pola misklasifikasi mengungkapkan adanya fenomena kebingungan model yang menetap pada algoritma FedAvg maupun FedProx, di mana kemiripan karakteristik fitur statistik menyebabkan terjadinya misklasifikasi timbal balik antara serangan *backdoor* dan *DoS*, serta antara *injection* dan *password attack* karena kesamaan vektor serangan pada lapisan aplikasi protokol HTTP. Kegagalan paling kritis teridentifikasi pada deteksi serangan *Man-in-the-Middle* (MITM) yang secara konsisten diklasifikasikan sebagai trafik normal karena sifat serangannya yang pasif tidak mengubah pola statistik trafik secara drastis, sehingga fitur yang digunakan model gagal menangkap anomali halus tersebut. Fenomena ini mempertegas limitasi model global yang cenderung kurang sensitif terhadap spesifisitas serangan unik pada perangkat IoT tertentu, yang pada akhirnya mengakibatkan rendahnya kemampuan deteksi terhadap ancaman kelas minoritas dalam skenario *Non-IID* ekstrem.

Sejalan dengan temuan tersebut, penelitian ini mengidentifikasi limitasi bahwa penerapan satu model global (*global model*) memiliki keterbatasan dalam menangani spesifisitas serangan yang unik pada tiap perangkat. Karena model global merupakan hasil agregasi rata-rata (konsensus), model tersebut mungkin kurang sensitif terhadap serangan spesifik yang hanya muncul pada jenis perangkat IoT tertentu (misalnya kerentanan *xss* yang spesifik pada *smart meter* berbasis web, namun tidak relevan bagi sensor lingkungan berbasis MQTT). Hal ini terlihat dari kesulitan model dalam mendeteksi kelas minoritas pada skenario Non-IID ekstrem.

Saran untuk penelitian ini masih dilakukan dalam bentuk simulasi terkontrol dengan dataset statis dan lingkungan eksperimen terbatas. Oleh karena itu, pengembangan selanjutnya diperlukan untuk mendekatkan skenario eksperimen dengan kondisi operasional sistem nyata. Berdasarkan temuan dan limitasi yang ada, berikut adalah saran pengembangan lanjutan

a. Implementasi Lingkungan Terkontainerisasi

   Penelitian selanjutnya disarankan mengimplementasikan *Federated Learning* menggunakan framework seperti Flower pada lingkungan berbasis *container*, misalnya Docker, agar simulasi komunikasi *client-to-server* lebih mendekati kondisi nyata serta mudah direplikasi.

b. Pengujian Ketahanan Beban (*Stress Testing*)

   Evaluasi lanjutan dapat difokuskan pada ketahanan sistem melalui pengujian beban, seperti peningkatan trafik secara tiba-tiba, *traffic burst*, distribusi input yang sangat tidak seimbang antar *client*, serta kondisi keterbatasan sumber daya (CPU dan *memory*).

c. Penggunaan Data Dinamis

Penggunaan data sintetis dan alat tes penetrasi keamanan untuk menyimulasikan anomali atau serangan langka dapat dipertimbangkan guna menguji stabilitas model di luar skenario dataset yang bersifat statis.

d. Evaluasi Skalabilitas dengan Partial Client Participation

Mengingat penggunaan parameter $fraction_fit = 1.0$ pada penelitian ini ditujukan untuk stabilitas simulasi pada jumlah *node* kecil, penelitian selanjutnya disarankan untuk menguji ketahanan model pada skenario *partial participation* misalnya *fraction_fit* = 0.1 hingga 0.5 dengan jumlah *client* yang diperbesar (puluhan hingga ratusan *node*). Pengujian ini penting untuk mensimulasikan kondisi nyata jaringan IoT *smart city* di mana keterbatasan *bandwidth*, perangkat yang *offline* (*drop-out*), atau kendala baterai tidak memungkinkan seluruh perangkat berpartisipasi serentak. Penelitian lanjutan perlu menganalisis apakah algoritma FedProx tetap mampu mempertahankan akurasi global ketika hanya sebagian kecil representasi data lokal yang tersedia di setiap ronde agregasi.

e. Penerapan *Personalized Federated Learning* (PFL)

Berdasarkan keterbatasan model global dalam menangani karakteristik unik tiap *gateway* IoT, penelitian selanjutnya disarankan untuk mengadopsi pendekatan *Personalized Federated Learning* (PFL). Dalam skema saat ini, distribusi Dirichlet digunakan untuk membagi data, namun model yang dihasilkan tetap dipaksakan menjadi satu model global yang seragam. PFL memungkinkan setiap *client* (*gateway*) untuk memiliki model yang dipersonalisasi mempertahankan lapisan dasar (*base layers*) dari model global namun melakukan *fine-tuning* pada lapisan akhir sesuai dengan distribusi data lokalnya. Pendekatan ini diharapkan dapat meningkatkan akurasi pada *gateway* yang memiliki profil serangan spesifik (misalnya *gateway* yang rawan DDoS vs *gateway* yang rawan Injection) tanpa mengorbankan pengetahuan kolektif dari jaringan.

f. Peningkatan Granularitas Deteksi (Hierarchical Classification)

Berdasarkan temuan adanya kebingungan klasifikasi timbal balik (*reciprocal misclassification*) antara kategori serangan yang memiliki kemiripan statistik (seperti *Backdoor* vs *DoS* dan *Injection* vs *Password*), penelitian selanjutnya disarankan untuk menerapkan pendekatan *Hierarchical Classification*. Pada pendekatan ini, model pertama bertugas mendeteksi biner (Normal vs Serangan) untuk memastikan keamanan dasar, kemudian model sekunder yang lebih spesifik digunakan untuk membedakan jenis

serangan. Hal ini bertujuan untuk mempertahankan tingkat deteksi serangan (*True Positive*) yang tinggi sekaligus meningkatkan presisi dalam mengidentifikasi jenis ancaman.

g. Integrasi Fitur Berbasis Protokol (*Feature Engineering*)

Mengingat tingginya tingkat *False Negative* pada deteksi serangan *Man-in-the-Middle* (MITM) yang cenderung diklasifikasikan sebagai trafik normal, penelitian masa depan perlu memperkaya *feature engineering* dengan atribut yang lebih sensitif terhadap anomali protokol, bukan hanya statistik aliran (*flow statistics*). Penggunaan fitur berbasis *Deep Packet Inspection* (DPI) sederhana atau analisis *header* ARP dan DNS disarankan untuk menangkap karakteristik serangan pasif yang sering kali tidak mengubah volume atau durasi trafik secara signifikan.

h. Evaluasi Spesifik per Tipe Perangkat

Melakukan evaluasi performa tidak hanya secara agregat global, melainkan melakukan *breakdown* akurasi berdasarkan tipe profil *gateway* yang disimulasikan (misal: *Gateway* Profil Video, *Gateway* Profil Sensor) untuk melihat efektivitas model pada vektor serangan yang relevan saja.

i. Peningkatan Realisme Distribusi Data melalui Manual *Skewing*

Meskipun distribusi Dirichlet $\alpha = 0.3$ telah berhasil mensimulasikan ketimpangan data Non-IID secara statistik, distribusi tersebut masih bersifat probabilistik dan belum sepenuhnya merepresentasikan spesifisitas fungsional perangkat di dunia nyata misalnya, *gateway* CCTV seharusnya secara eksklusif didominasi trafik video dan rentan terhadap serangan pada protokol RTSP/HTTP, berbeda dengan *gateway* sensor lingkungan yang didominasi trafik MQTT. Penelitian selanjutnya disarankan untuk menerapkan teknik *manual skewing* atau modifikasi bobot *dirichlet* yang dipersonalisasi (*guided dirichlet distribution*). Pendekatan ini bertujuan untuk memetakan kategori serangan dan jenis trafik normal secara deterministik ke *client* tertentu sesuai dengan profil perangkat aslinya di lapangan, sehingga simulasi tidak hanya heterogen secara statistik tetapi juga valid secara kontekstual operasional *smart city*.

DAFTAR PUSTAKA

Abdel-Basset, M., Hawash, H., Moustafa, N., Razzak, I., & Abd Elfattah, M. (2024). Privacy-preserved learning from non-i.i.d data in fog-assisted IoT: A federated learning approach. *Digital Communications and Networks*, *10*(2), 404–415. https://doi.org/10.1016/j.dcan.2022.12.013

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1). https://doi.org/10.1002/ett.4150

Ahmadi, K., & Javidan, R. (2023). DDoS Attack Detection in a Real Urban IoT Environment Using Federated Deep Learning. *Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience, CSR 2023*, 117–122. https://doi.org/10.1109/CSR57506.2023.10224916

Al-Ghadi, T. Q., Manickam, S., Widia, I. D. M., Wulandari, E. R. N., & Karuppayah, S. (2025). Leveraging federated learning for DoS attack detection in IoT networks based on ensemble feature selection and deep learning models. *Cyber Security and Applications*, *3*, 100098. https://doi.org/10.1016/j.csa.2025.100098

Al-Huthaifi, R., Li, T., Huang, W., Gu, J., & Li, C. (2023). Federated learning in smart cities: Privacy and security survey. *Information Sciences*, *632*, 833–857. https://doi.org/10.1016/j.ins.2023.03.033

Ali, A., Assam, M., Khan, F. U., Ghadi, Y. Y., Nurdaulet, Z., Zhibek, A., Shah, S. Y., & Alahmadi, T. J. (2024). An optimized multilayer perceptron-based network intrusion detection using Gray Wolf Optimization. *Computers and Electrical Engineering*, *120*, 109838. https://doi.org/10.1016/j.compeleceng.2024.109838

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access*, *8*, 165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862

Barker, L., & Brooks, A. (2023). *IDC FutureScape IDC FutureScape: Worldwide Smart Cities and Communities 2024 Predictions*. https://www.idc.com/wp-content/uploads/2025/03/IDC_FutureScape_Worldwide_Smart_Cities_and_Communities_2024_Predictions_-_2023_Oct.pdf

Bierbrauer, D. A., Coffey, S. M., Willeke, M. R., Beggs, J. D., & Bastian, N. D. (2025). Data-efficient Federated Learning for Edge Network Intrusion Detection. *Engineering Applications of Artificial Intelligence*, *150*, 110685. https://doi.org/10.1016/j.engappai.2025.110685

Bodagala, H., & Priyanka, H. (2022). Security for IoT using Federated Learning. *Proceedings - 2022 International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems, ICMACC 2022*, 131–136. https://doi.org/10.1109/ICMACC54824.2022.10093557

Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise AgüeraAg, H., & Arcas, A. (2017). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. https://proceedings.mlr.press/v54/mcmahan17a.html.

Cha, N., & Chang, L. (2025). Addressing Non-IID with Data Quantity Skew in Federated Learning. *Information*, *16*(10), 861. https://doi.org/10.3390/info16100861

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Devi, M., Nandal, P., & Sehrawat, H. (2025). Federated learning-enabled lightweight intrusion detection system for wireless sensor networks: A cybersecurity approach against DDoS attacks in smart city environments. *Intelligent Systems with Applications*, *27*, 200553. https://doi.org/10.1016/j.iswa.2025.200553

Di Mauro, M., Galatro, G., Fortino, G., & Liotta, A. (2021). Supervised feature selection techniques in network intrusion detection: A critical review. *Engineering Applications of Artificial Intelligence*, *101*, 104216. https://doi.org/10.1016/j.engappai.2021.104216

Farooq, M. S., Saleem, M., Khan, M. A., Khan, M. F., Siddiqui, S. Y., Aslam, M. S., & Adnan, K. M. (2025). Interpretable Federated Learning Model for Cyber Intrusion Detection in Smart Cities with Privacy-Preserving Feature Selection. *Computers, Materials & Continua*, *85*(3), 5183–5206. https://doi.org/10.32604/cmc.2025.069641

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, 102419. https://doi.org/10.1016/j.jisa.2019.102419

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Hijazi, N. M., Aloqaily, M., Guizani, M., Ouni, B., & Karray, F. (2024). Secure Federated Learning with Fully Homomorphic Encryption for IoT Communications. *IEEE Internet of Things Journal*, *11*(3), 4289–4300. https://doi.org/10.1109/JIOT.2023.3302065

Javeed, D., Saeed, M. S., Adil, M., Kumar, P., & Jolfaei, A. (2024). A federated learning-based zero trust intrusion detection system for Internet of Things. *Ad Hoc Networks*, *162*, 103540. https://doi.org/10.1016/j.adhoc.2024.103540

Jin, L.-X., Jiang, W., Wen, X.-Y., Lin, M.-Y., Zhan, J.-Y., Zhou, X.-Z., Habtie, M. A., & Werghi, N. (2025). A survey of backdoor attacks and defences: From deep neural networks to large language models. *Journal of Electronic Science and Technology*, *23*(3), 100326. https://doi.org/10.1016/j.jnlest.2025.100326

Kunang, Y. N., Nurmaini, S., Stiawan, D., & Suprapto, B. Y. (2021a). Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications*, *58*, 102804. https://doi.org/10.1016/j.jisa.2021.102804

Kunang, Y. N., Nurmaini, S., Stiawan, D., & Suprapto, B. Y. (2021b). Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications*, *58*, 102804. https://doi.org/10.1016/j.jisa.2021.102804

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). *FEDERATED OPTIMIZATION IN HETEROGENEOUS NETWORKS*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2023). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. http://arxiv.org/abs/1602.05629

Morgan, S. (2025, Desember 11). *2025 Cybersecurity Almanac: 100 Facts, Figures, Predictions And Statistics*. Cybercrime Magazine.

Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, *72*, 102994. https://doi.org/10.1016/j.scs.2021.102994

Olanrewaju-George, B., & Pranggono, B. (2025). Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. *Cyber Security and Applications*, *3*. https://doi.org/10.1016/j.csa.2024.100068

Prazeres, N., Costa, R. L. de C., Santos, L., & Rabadão, C. (2023). Engineering the application of machine learning in an IDS based on IoT traffic flow. *Intelligent Systems with Applications*, *17*, 200189. https://doi.org/10.1016/j.iswa.2023.200189

Qazi, E.-H., Imran, M., Haider, N., Shoaib, M., & Razzak, I. (2022a). An intelligent and efficient network intrusion detection system using deep learning. *Computers and Electrical Engineering*, *99*, 107764. https://doi.org/10.1016/j.compeleceng.2022.107764

Qazi, E.-H., Imran, M., Haider, N., Shoaib, M., & Razzak, I. (2022b). An intelligent and efficient network intrusion detection system using deep learning. *Computers and Electrical Engineering*, *99*, 107764. https://doi.org/10.1016/j.compeleceng.2022.107764

Quincozes, S. E., Albuquerque, C., Passos, D., & Mossé, D. (2021). A survey on intrusion detection and prevention systems in digital substations. *Computer Networks*, *184*, 107679. https://doi.org/10.1016/j.comnet.2020.107679

Rehman, S. ur, Alhulayyil, H., Alzahrani, T., AlSagri, H., Khalid, M. U., & Gruhn, V. (2025). Intrusion detection system framework for cyber-physical systems. *Egyptian Informatics Journal*, *30*, 100600. https://doi.org/10.1016/j.eij.2024.100600

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, *181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199

Statista Research Department. (2025, Desember 1). *Number of internet and social media users worldwide as of October 2025*. Statista Research Department. https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=Number%20of%20internet%20and%20social%20media%20users%20worldwide%202025&text=As%20of%20October%202025%2C%206.04,population%2C%20were%20social%20media%20users.

Thakur, A., Tyagi, R., Tripathy, H. K., Yang, T., Rathore, R. S., Mo, D., & Wang, L. (2024). Detecting Network Attack using Federated Learning for IoT Devices. *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, 1–6. https://doi.org/10.1109/IACIS61494.2024.10721980

Thantharate, P., & T, A. (2023). CYBRIA - Pioneering Federated Learning for Privacy-Aware Cybersecurity with Brilliance. *2023 IEEE 20th International Conference on Smart*

Communities: Improving Quality of Life using AI, Robotics and IoT (HONET), 56–61. https://doi.org/10.1109/HONET59747.2023.10374608

Tian, P., Chen, Z., Yu, W., & Liao, W. (2021). Towards asynchronous federated learning based threat detection: A DC-Adam approach. *Computers & Security*, *108*, 102344. https://doi.org/10.1016/j.cose.2021.102344

Tiwari, A. (2022). Supervised learning: From theory to applications. Dalam *Artificial Intelligence and Machine Learning for EDGE Computing* (hlm. 23–32). Elsevier. https://doi.org/10.1016/B978-0-12-824054-0.00026-5

Upreti, D., Yang, E., Kim, H., & Seo, C. (2024). A Comprehensive Survey on Federated Learning in the Healthcare Area: Concept and Applications. *Computer Modeling in Engineering & Sciences*, *140*(3), 2239–2274. https://doi.org/10.32604/cmes.2024.048932

Zhou, Y., Wang, J., Qin, Y., Kong, X., Xie, X., Qi, H., & Zeng, D. (2025). Federated Learning with complete service commitment of data heterogeneity. *Knowledge-Based Systems*, *310*, 112937. https://doi.org/10.1016/j.knosys.2024.112937

Zhu, H., Xu, J., Liu, S., & Jin, Y. (2021). *Federated Learning on Non-IID Data: A Survey*. https://doi.org/doi.org/10.1016/j.neucom.2021.07.098

LAMPIRAN