RANCANG BANGUN SISTEM KLASIFIKASI JENIS KENDARAAN BERBASIS *DEEP LEARNING* DENGAN PENGAMBILAN GAMBAR DAN NOTIFIKASI *REAL-TIME* VIA *TELEGRAM* PADA CV. INDAH JAYA SENTOSA

KERJA PRAKTIK

Program Studi

S1 Teknik Komputer

Oleh :
Ahmad Fahmi Ilmi Fauzi
22410200009

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

RANCANG BANGUN SISTEM KLASIFIKASI JENIS KENDARAAN *DEEP LEARNING* DENGAN PENGAMBILAN GAMBAR DAN NOTIFIKASI *REAL-TIME* VIA *TELEGRAM* PADA CV. INDAH JAYA SENTOSA

Diajukan sebagian salah satu syarat untuk menyelesaikan

Program Strata Satu (S1)

Disusun Oleh :

Nama : **Ahmad Fahmi Ilmi Fauzi**

NIM : **22410200009**

Program : **S1 (Strata Satu)**

Jurusan : **Teknik Komputer**

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

“Angrêbuting batin, tan kêna kawênang rasa.”

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM KLASIFIKASI JENIS KENDARAAN BERBASIS *DEEP LEARNING* DENGAN PENGAMBILAN GAMBAR DAN NOTIFIKASI *REAL-TIME* VIA *TELEGRAM* PADA CV. INDAH JAYA SENTOSA

Laporan Kerja Praktik oleh

Ahmad Fahmi Ilmi Fauzi

NIM : 22410200009

Telah diperiksa, diuji dan disetujui

Surabaya, 15 Desember 2025

Disetujui:

Pembimbing	Penyelia
Harianto, S.Kom., M.Eng.	**Yayuk Mariana**
NIDN. 0722087701	

Mengetahui,
Ketua Program Studi S1 Teknik Komputer

Fakultas Teknologi dan Informatika
UNIVERSITAS Dinamika

Dr. Ira Puspasari, S.Si., M.T.
NIDN. 0710078601

SURAT PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Universitas Dinamika, saya:

Nama	: Ahmad Fahmi Ilmi Fauzi
NIM	: 22410200009
Program Studi	: S1 Teknik Komputer
Fakultas	: Fakultas Teknologi dan Informatika
Jenis Karya	: Laporan Kerja Praktik
Judul Karya	: RANCANG BANGUN SISTEM KLASIFIKASI JENIS KENDARAAN BERBASIS *DEEP LEARNING* DENGAN PENGAMBILAN GAMBAR DAN NOTIFIKASI *REAL-TIME* VIA *TELEGRAM* PADA CV. INDAH JAYA SENTOSA

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Universitas Dinamika Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right)* atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data *(database)* untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 8 Desember 2025

Yang menyatakan

Ahmad Fahmi Ilmi Fauzi

NIM : 22410200009

ABSTRAK

CV. Indah Jaya Sentosa merupakan perusahaan distribusi barang yang bergantung pada pengelolaan arus kendaraan masuk-keluar. Namun, pencatatan manual oleh petugas keamanan sering menyebabkan keterlambatan laporan dan informasi, mengganggu efisiensi serta meningkatkan risiko operasional seperti kehilangan data pengawasan kendaraan krusial bagi rantai pasok. Oleh karena itu, diperlukan solusi teknologi untuk monitoring kendaraan secara cepat, akurat, dan *real-time* guna mendukung transformasi digital di sektor distribusi.

Pada kerja praktek ini mengimplementasikan sistem klasifikasi kendaraan berbasis *deep learning* dengan model *MobileNetV2*, terintegrasi kamera untuk pengambilan gambar otomatis. Sistem mengklasifikasikan kendaraan menjadi empat kategori (truk, mobil, motor, bus) dan mengirimkan notifikasi *real-time* via *Telegram*: ke petugas keamanan untuk semua kendaraan, serta ke admin khusus untuk truk terkait distribusi. Pendekatan ini mengotomatisasi proses manual, meminimalkan kesalahan, dan mempercepat informasi.

Pengujian menunjukkan model *MobileNetV2*, mencapai akurasi lebih 90% dengan waktu notifikasi kurang dari 10 detik. Sistem meningkatkan kecepatan, akurasi, dan integrasi monitoring di perusahaan, mendukung efisiensi petugas serta pengawasan distribusi. Penelitian ini berkontribusi pada penerapan *deep learning* untuk solusi industri distribusi, sebagai acuan bagi perusahaan serupa.

Kata kunci: *Artificial Intelligence, Deep learning*, *MobileNetV2*, Klasifikasi Kendaraan, Pengambilan Gambar, *Telegram*, CV. Indah Jaya Sentosa

KATA PENGANTAR

Penulis bersyukur kepada Allah SWT atas segala kemudahan dan petunjuk dalam menyelesaikan laporan kerja praktik ini, yang bertujuan untuk mengembangkan sistem monitoring kendaraan berbasis *deep learning* guna mendukung efisiensi operasional CV. Indah Jaya Sentosa. Laporan ini merupakan dokumentasi hasil kerja praktik yang mengintegrasikan teknologi kecerdasan buatan untuk mengotomatisasi klasifikasi kendaraan dan pengiriman notifikasi *real-time* melalui aplikasi *Telegram*. Penelitian ini diharapkan dapat memberikan manfaat nyata bagi perusahaan dalam meningkatkan akurasi dan kecepatan pengawasan kendaraan, sekaligus menjadi referensi untuk pengembangan sistem serupa di bidang distribusi barang.

Ucapan terima kasih disampaikan kepada semua pihak yang telah memberikan dukungan dalam penyusunan laporan ini, yaitu:

1. Allah SWT atas segala rahmat, berkat, dan hidayah yang telah diberikan.
2. Orang tua dan keluarga yang senantiasa memberikan doa, dukungan, serta motivasi.
3. Tan Aemelia, S.Kom., M.MT. selaku Dekan Fakultas Teknologi dan Informatika.
4. Dr. Ira Puspasari, S.Si., M.T. selaku Ketua Program Studi S1 Teknik Komputer.
5. Harianto, S.Kom., M.Eng. selaku dosen pembimbing kerja praktik.
6. Yayuk Mariana selaku penyelia di CV Indah Jaya Sentosa.
7. Ahmad Yani sebagai ketua divisi IT & Teknologi.
8. Teman-teman khususnya anak-anak Hima Karyo yang memberikan bantuan dan dukungan dalam penyusunan laporan ini.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang.

Surabaya, 23 Desember 2025

Ahmad Fahmi Ilmi fauzi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

CV. Indah Jaya Sentosa, sebuah perusahaan yang bergerak di bidang distribusi barang, mengandalkan kelancaran arus kendaraan yang masuk dan keluar area perusahaan untuk mendukung operasional sehari-hari. Berbagai jenis kendaraan, seperti truk, mobil, motor, dan bus, beroperasi untuk memenuhi kebutuhan pengiriman barang. Namun, pencatatan data kendaraan masih dilakukan secara manual oleh petugas keamanan, yang menyebabkan keterlambatan laporan dan lambatnya penyampaian informasi kepada manajemen atau tim distribusi. Akibatnya, efisiensi operasional terganggu, terutama dalam pengawasan kendaraan truk yang berperan besar dalam distribusi barang.

Tantangan ini diperparah oleh volume kendaraan yang signifikan, yang menuntut penanganan cepat dan akurat. Sistem manual saat ini tidak mampu menghasilkan data yang andal secara *real-time*, padahal kecepatan dan ketepatan informasi sangat penting dalam industri distribusi untuk mendukung pengambilan keputusan, seperti pengaturan jadwal pengiriman atau pemantauan kendaraan pengangkut barang. Ketidakefisienan ini meningkatkan risiko operasional, seperti keterlambatan penanganan truk pengangkut barang atau potensi kehilangan data penting, yang dapat mengganggu rantai pasok perusahaan.

Perkembangan teknologi *Artificial Intelligence (AI)* dan *deep learning* menawarkan solusi untuk mengatasi masalah tersebut melalui sistem otomatis yang mampu mengenali dan mengklasifikasikan kendaraan dengan cepat dan akurat. Dengan integrasi kamera untuk pengambilan gambar dan sistem notifikasi *real-time* melalui *Telegram*, informasi dapat langsung sampai ke petugas keamanan dan manajemen tanpa menunggu laporan manual. Penerapan teknologi ini memungkinkan CV. Indah Jaya Sentosa untuk beradaptasi dengan tren otomatisasi berbasis AI di industri distribusi, sehingga meningkatkan

efisiensi dan daya saing operasional.

Berdasarkan kebutuhan tersebut, kerja praktek ini bertujuan untuk mengembangkan sistem pemantauan kendaraan berbasis *deep learning* yang dapat mengklasifikasikan jenis kendaraan secara otomatis dan mengirimkan notifikasi *real-time* melalui *Telegram*. Sistem ini diharapkan dapat mengatasi keterbatasan sistem manual, meningkatkan akurasi dan kecepatan monitoring kendaraan, serta memperkuat pengawasan terhadap aktivitas distribusi barang di CV. Indah Jaya Sentosa. Langkah ini sekaligus mendukung upaya perusahaan untuk memodernisasi operasionalnya di tengah dinamika industri distribusi yang semakin kompetitif.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, berikut adalah rumusan masalah yang dapat dirumuskan:

1. Bagaimana merancang sistem berbasis *deep learning* menggunakan model *MobileNetV2* untuk mengenali dan mengklasifikasikan jenis kendaraan (truk, mobil, motor, dan bus) secara otomatis dan akurat?
2. Bagaimana mengintegrasikan sistem klasifikasi kendaraan dengan kamera untuk pengambilan gambar dan memastikan pengiriman notifikasi ke petugas keamanan dan admin melalui aplikasi *Telegram* dalam waktu kurang dari 10 detik?

1.3 Batasan Masalah

Penelitian ini dibatasi pada hal-hal berikut:

1. Jenis Kendaraan: Sistem hanya mengenali empat jenis kendaraan: truk, mobil, motor, dan bus.
2. Teknologi: Menggunakan model *deep learning MobileNetV2* dan kamera untuk menangkap gambar kendaraan.
3. Notifikasi: Notifikasi dikirim lewat *Telegram* ke petugas keamanan untuk semua kendaraan, dan ke admin khusus untuk truk.
4. Lokasi: Sistem hanya diuji di area CV. Indah Jaya Sentosa.
5. Pengukuran: Fokus pada akurasi klasifikasi > 90% dan kecepatan

notifikasi < 10 detik.

6. Data: Hanya menggunakan gambar kendaraan dari kamera, tanpa data lain seperti suara.

1.4 Tujuan

Berdasarkan rumusan masalah, tujuan penelitian ini adalah:

1. Mengembangkan sistem monitoring kendaraan otomatis berbasis *deep learning* di CV. Indah Jaya Sentosa
2. Merancang sistem berbasis *deep learning* dengan model *MobileNetV2* untuk mengenali dan mengklasifikasikan jenis kendaraan (truk, mobil, motor, dan bus) secara akurat.
3. Mengintegrasikan sistem klasifikasi kendaraan dengan kamera untuk menangkap gambar secara *real-time* dan mengirimkan notifikasi melalui *Telegram* dalam waktu kurang dari 10 detik.

1.5 Manfaat

Penelitian ini memberikan manfaat sebagai berikut:

1. Informasi *Real-time*: Notifikasi melalui *Telegram* yang dikirim dalam waktu kurang dari 10 detik memastikan petugas keamanan dan admin mendapatkan informasi cepat untuk pengambilan keputusan.
2. Mendukung Pengawasan Distribusi: Sistem ini membantu perusahaan memantau kendaraan, terutama truk yang terkait dengan distribusi barang, sehingga pengelolaan rantai pasok menjadi lebih baik.
3. Meningkatkan Keamanan dan Akurasi Data: Membantu memastikan data kendaraan tercatat dengan benar dan aktivitas keluar-masuk kendaraan secara akurat.

BAB II
GAMBARAN UMUM INSTANSI

2.1 Latar Belakang Perusahaan

CV INDAH JAYA
SENTOSA

Figure 2.1 Logo Perusahaan

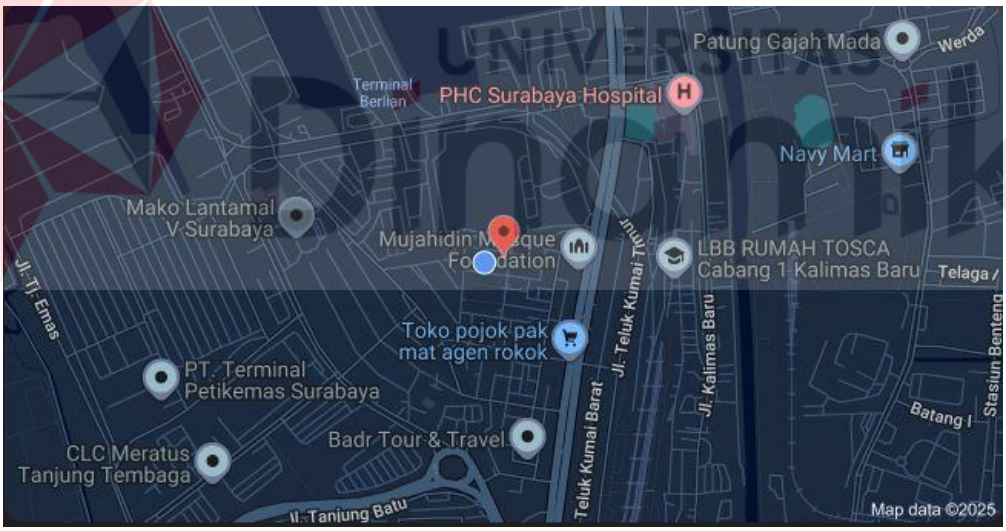

Figure 2.2 Lokasi Perusahaan

CV. Indah Jaya Sentosa adalah perusahaan yang bergerak di bidang distribusi barang, dengan fokus pada penyediaan layanan pengiriman dan pengelolaan rantai pasok untuk berbagai jenis produk. Berbasis di Surabaya, perusahaan ini telah menjalankan operasinya selama beberapa tahun dengan mengandalkan jaringan distribusi yang luas dan tim operasional yang kompeten. Berlokasi strategis di Jalan Teluk Nibung Barat 7/20, Perak, Surabaya, Jawa

Timur, CV. Indah Jaya Sentosa memiliki aksesibilitas optimal ke pelabuhan dan jalur transportasi utama, yang mendukung kelancaran distribusi barang ke berbagai wilayah di Indonesia. Komunikasi dengan perusahaan dapat dilakukan melalui nomor telepon 081332876018 atau email indahjayasentosa1@yahoo.com, dengan kontak utama Yayuk Mariana, yang memastikan responsivitas terhadap kebutuhan pelanggan.

CV. Indah Jaya Sentosa menghadapi tantangan operasional yang signifikan, terutama dalam pengelolaan arus kendaraan yang meliputi truk pengangkut barang, mobil, motor, dan bus, dengan rata-rata 50 kendaraan per hari. Proses monitoring dan pencatatan kendaraan saat ini masih dilakukan secara manual oleh petugas keamanan, yang sering kali menyebabkan keterlambatan penyusunan laporan, kesalahan input data, dan keterbatasan informasi *real-time* untuk keperluan manajerial. Kondisi ini berdampak pada efisiensi operasional, khususnya dalam pengawasan truk yang memainkan peran krusial dalam aktivitas distribusi. Untuk mengatasi tantangan tersebut, perusahaan berkomitmen untuk mengadopsi transformasi digital melalui penerapan teknologi kecerdasan buatan, seperti sistem klasifikasi kendaraan berbasis *deep learning*, guna meningkatkan akurasi dan kecepatan pengelolaan data kendaraan.

Struktur organisasi CV. Indah Jaya Sentosa dipimpin oleh Direktur Utama, Yuri AS, yang bertanggung jawab atas arah strategis dan kebijakan perusahaan, didukung oleh Wakil Direktur Yacob Adi Saputra untuk memastikan koordinasi antardivisi yang efektif. Divisi Operasional, yang dipegang oleh Rachmad Fajar, mengelola aktivitas harian termasuk pengawasan kendaraan, sementara Yayuk Mariana sebagai kepala Sumber Daya Manusia mengelola aspek perekrutan dan kesejahteraan karyawan. Divisi IT dan Teknologi di bawah Ahmad Yani memainkan peran kunci dalam mengintegrasikan solusi teknologi, seperti sistem berbasis AI, untuk mendukung efisiensi operasional. Dengan pendekatan ini, CV. Indah Jaya Sentosa berupaya memperkuat posisinya sebagai perusahaan distribusi terpercaya yang mampu beradaptasi dengan dinamika industri modern.

2.2 Identitas Perusahaan

Tempat	: CV. Indah Jaya Sentosa
Alamat	: Jl. Teluk Nibung barat 7/20, Perak, Surabaya, Jawa Timur
Telepon	: 081332876018
Contact Person	: Yayuk Mariana
Email	: indahjayasentosa1@yahoo.com

2.3 Visi Perusahaan

"Menjadi perusahaan distribusi terpercaya dan terdepan di Indonesia dengan pelayanan prima, jaringan yang luas, serta komitmen terhadap kualitas dan kepuasan pelanggan."

2.4 Misi Perusahaan

Untuk mewujudkan visi tersebut, perusahaan berkomitmen memberikan pelayanan distribusi yang cepat, tepat, dan aman; menjaga kualitas produk sesuai standar; membangun jaringan distribusi yang luas dan efisien; mengoptimalkan teknologi untuk mendukung efektivitas operasional menjalin hubungan kerja yang baik dengan mitra bisnis serta terus meningkatkan kompetensi sumber daya manusia melalui pelatihan dan pengembangan berkelanjutan.

2.5 Struktur Organisasi

Figure 2.3 Struktur Organisasi

Pada gambar struktur organisasi pada CV. Indah Jaya Sentosa. Setiap bagian memiliki tugas pokok dan fungsi masing-masing. Berikut di bawah ini adalah detail dari tugas pokok dan fungsinya

1. Direktur Utama - Yuri AS Pemimpin tertinggi perusahaan yang bertanggung jawab penuh atas VISI, MISI, Strategi dan arah kebijakan perusahaan. Mengambil keputusan strategis yang berpengaruh besar terhadap perkembangan dan keberlangsungan usaha. Memastikan seluruh departemen bekerja sesuai target dan tujuan perusahaan.
2. Wakil Direktur - Yacob Adi Saputra Mendampingi dan membantu Direktur Utama dalam menjalankan tugas manajerial. Menggantikan peran Direktur Utama saat berhalangan hadir. Mengawasi koordinasi antar divisi agar berjalan efisien dan efektif
3. Keuangan dan Akuntansi - Achmad Hambali Mengelola arus kas masuk dan keluar perusahaan. Menyusun laporan keuangan secara berkala. Mengatur anggaran, pengeluaran, pajak dan memastikan kepatuhan terhadap regulasi keuangan.
4. Sumber Daya Manusia (SDM) - Yayuk Mariana Mengelola perekrutan, pelatihan dan pengembangan karyawan. Mengatur administrasi

karyawan, gaji, tunjangan dan kesejahteraan pegawai. Menjaga hubungan kerja yang sehat dan produktif antara tim.

5. Operasional - Rachmad Fajar Mengatur kegiatan operasional harian agar berjalan lancar dan efisien. Mengoptimalkan proses kerja di lapangan. Memastikan standar kualitas produk atau layanan terpenuhi.
6. Pemasaran dan Penjualan - Irsya Pratiwi Menyusun strategi pemasaran untuk memperluas pangsa pasar. Mengelola promosi, branding dan hubungan dengan pelanggan. Mengawasi proses penjualan dan pencapaian target penjualan.
7. Riset dan Pengembangan - Mujiah Mengembangkan inovasi produk atau layanan baru. Melakukan riset pasar dan analisis tren industri. Meningkatkan kualitas dan efisiensi produk agar tetap kompetitif.
8. IT dan Teknologi - Ahmad Yani Mengelola infrastruktur teknologi perusahaan. Mengembangkan dan memelihara sistem informasi yang menunjang operasional. Menjaga keamanan data dan memastikan pemanfaatan teknologi secara optimal.

BAB III
LANDASAN TEORI

3.1 Konsep Dasar *Deep learning*

Deep learning (Pembelajaran Mendalam) telah didefinisikan sebagai sub-bidang dari *Machine Learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan tersembunyi (Lecun et al., 2015). Fondasi dari konsep ini adalah kemampuan jaringan untuk mempelajari representasi data secara hierarkis dan abstrak secara otomatis. Seperti yang dijelaskan oleh (Schmidhuber, 2015), kedalaman arsitektur inilah yang memungkinkan model untuk memecahkan masalah yang sangat kompleks yang sebelumnya sulit ditangani oleh teknik *machine learning* konvensional. Proses hierarkis ini dapat dianalogikan dengan cara kerja sistem visual otak manusia, di mana lapisan-lapisan saraf memproses informasi dari fitur yang sederhana (seperti tepian dan sudut) hingga yang semakin kompleks (seperti bentuk objek secara utuh.

Dalam perkembangannya, keberhasilan *Deep learning* didorong oleh tiga faktor kunci: ketersediaan data dalam skala besar (*big data*), peningkatan daya komputasi (terutama dengan penggunaan GPU), serta kemajuan dalam algoritma dan arsitektur (Goodfellow et al., 2016). Arsitektur-arsitektur khusus seperti *Convolutional Neural Networks (CNN)* untuk data gambar dan *Recurrent Neural Networks (RNN)* untuk data sekuensial telah menjadi landasan bagi berbagai terobosan. Sebagai contoh, penelitian oleh (Krizhevsky et al., n.d.) pada kompetisi *ImageNet* menunjukkan bagaimana arsitektur *Deep* CNN dapat mengurangi tingkat kesalahan pengenalan gambar secara drastis, sehingga membuka era modern *Deep learning*. Dengan demikian, *Deep learning* tidak hanya menjadi tulang punggung dalam kemajuan Kecerdasan Buatan tetapi juga terus mentransformasi berbagai bidang seperti *computer vision* dan pemrosesan bahasa alami.

3.1.1 Definisi dan Sejarah Singkat – Jaringan Saraf dalam *Machine Learning*

Jaringan Saraf Tiruan (*Artificial Neural Network/ANN*) adalah sebuah model komputasi yang terinspirasi dari struktur dan fungsi biologis

otak manusia (Mcculloch & Pitts, 1943). Jaringan saraf didefinisikan sebagai prosesor terdistribusi paralel yang tersusun atas unit-unit pemroses sederhana (*neuron*) yang memiliki kecenderungan alamiah untuk menyimpan pengetahuan dan membuatnya tersedia untuk digunakan. Dalam konteks pembelajaran mesin, jaringan saraf berfungsi sebagai arsitektur pembelajaran yang mampu menangkap pola dan hubungan kompleks dalam data melalui proses pelatihan berulang.

Perkembangan jaringan saraf dalam pembelajaran mesin telah melalui beberapa periode penting:

1. Era Konsep Awal (1940-1950)
    a. Model neuron formal pertama diperlakukan oleh (Mcculloch & Pitts, 1943)yang mendemonstrasikan bagaimana jaringan *neuron* sederhana dapat melakukan komputasi logika.
    b. Pengembangkan *perceptron*, model jaringan saraf paling sederhana yang mampu melakukan klasifikasi pola *linear*.
2. Masa Kemunduran (1970-1980)
    a. Kritik fundamental dari (Minsky & Papert, 1969) menunjukkan keterbatasan *perceptron* dalam menyelesaikan masalah *non-linear* seperti fungsi XOR, yang menyebabkan penurunan minat penelitian dalam bidang ini.
    b. (Rumelhart et al., 1986)) mempopulerkan kembali algoritma *backpropagation* yang efektif untuk melatih jaringan multi-layer, membuka jalan untuk arsitektur yang lebih dalam.
3. Kebangkitan *Deep learning* (2000-Sekarang)
    a. (Hinton G.E. & Salakhutdinov R.R., 2006) memperkenalkan *Deep Belief Networks* yang menunjukkan efektivitas pelatihan *layer-by-layer* untuk jaringan yang dalam.
    b. (Krizhevsky et al., n.d.) mendemonstrasikan keberhasilan *Deep Convolutional Neural Network* dalam kompetisi *ImageNet*, menandai revolusi *deep learning* modern.

Perkembangan historis ini menunjukkan evolusi jaringan saraf dari konsep

teoretis sederhana menjadi arsitektur kompleks yang mendorong kemajuan signifikan dalam bidang pembelajaran mesin dan kecerdasan buatan.

3.1.2 Perbedaan AI, *Machine Learning*, *Deep learning*

Menurut (Russell & Norvig, 2022), ketiga konsep ini membentuk hubungan hierarkis yang semakin spesifik, dimana Kecerdasan Buatan (*Artificial Intelligence*) merupakan payung terluas, *Machine Learning* (ML) merupakan bagian dari AI, dan *Deep learning* (DL) merupakan implementasi khusus dari *Machine Learning.*

1. *Artificial Intelligence*
    a. Definisi: (McCarthy et al., 1955), AI adalah bidang studi yang didasarkan pada konjektur bahwa setiap aspek pembelajaran atau fitur kecerdasan lainnya pada prinsipnya dapat dideskripsikan dengan begitu rincinya sehingga sebuah mesin dapat dibuat untuk mensimulasikannya.
    b. Cakupan: Sistem berbasis aturan, pemrograman simbolik, robotika, sistem pakar, dan pemrosesan bahasa alami
    c. Contoh: Sistem catur *Deep Blue* yang mengalahkan Garry Kasparov (1997)
2. Machine Learning (ML)
    a. Definisi: Menurut (Tom M. Mitchell, 1997) ML adalah bidang studi yang memberikan kemampuan pada komputer untuk belajar tanpa diprogram secara eksplisit
    b. Cakupan: Algoritma *supervised learning, unsupervised learning,* dan *reinforcement learning*
    c. Karakteristik: Memerlukan *feature engineering* manual, bekerja dengan dataset yang lebih kecil
    d. Contoh: Algoritma *decision tree* untuk klasifikasi, SVM untuk regresi
3. *Deep learning* (DL)
    a. Definisi: Menurut (Lecun et al., 2015), DL adalah subset ML yang menggunakan multiple layers untuk secara progresif mengekstrak

fitur-fitur level tinggi dari raw input

b. Cakupan: Jaringan saraf dalam, *convolutional neural networks, recurrent neural networks*
c. Karakteristik: Dapat belajar *feature representation* secara otomatis, memerlukan data dalam jumlah besar
d. Contoh: *AlphaGo* yang mengalahkan *champion Go* dunia (2016)

Menurut (Goodfellow et al., 2016), perbedaan mendasar terletak pada:

1. Kompleksitas fitur: DL mampu menangkap fitur yang lebih abstrak dan hierarkis
2. Feature engineering: ML membutuhkan *feature engineering* manual, sedangkan DL belajar fitur secara otomatis
3. Kebutuhan data: DL memerlukan dataset yang jauh lebih besar untuk *training* yang efektif
4. Kebutuhan komputasi: DL membutuhkan resources komputasi yang lebih intensif

Hubungan ketiganya dapat digambarkan sebagai lingkaran konsentris dimana AI ⊃ ML ⊃ DL, dengan DL menjadi pendorong utama kemajuan AI modern berkat kemampuannya dalam menangani data yang kompleks dan tidak terstruktur.

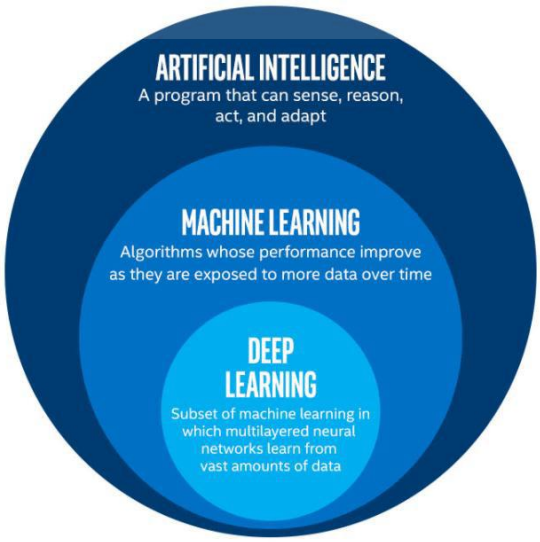

Figure 3.1 Diagram Venn

Table 3.1 Perbandingan AI, ML dan DL

Aspek	Artificial Intelligence (AI)	Machine Learning	*Deep learning*
Definisi	Sistem yang dapat melakukan tugas seperti manusia	Subset AI yang belajar dari data	Subset ML dengan jaringan saraf tiruan
Pendekatan	System berbasis aturan hingga pembelajaran	Algoritma statistic belajar dari data	Jaringan neural dalam belajar fitur otomatis
Data	Tidak selalu butuh data besar	Butuh data terstruktur menengah	Butuh data sangat besar
Contoh	Siri, Alexa, robotika	SVM, Decision Tree	CNN, RNN, Transformers
Aplikasi	Game AI, chatbot sederhana	Filter spam	Pengenalan gambar, NLP

Pemahaman hierarkis ini menjadi dasar pemilihan *Deep learning* sebagai pendekatan dalam kerja praktik ini, karena tugas klasifikasi gambar kendaraan memerlukan kemampuan untuk mempelajari fitur-fitur hierarkis dan kompleks secara otomatis dari data citra, yang merupakan keunggulan utama DL dibanding ML konvensional.

3.2 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) atau Jaringan Saraf Konvolusional merupakan arsitektur *deep learning* yang dominan dalam pemrosesan data *grid* seperti gambar. Arsitektur ini terinspirasi dari organisasi *visual cortex* pada otak binatang (Hubel & Wiesel, 1962) dan dirancang untuk mempelajari hierarki fitur secara otomatis sambil mempertahankan konteks spasialnya. Keunggulan utama CNN, seperti yang dijelaskan oleh (LeCun Y. et al., 1998), terletak pada tiga properti arsitekturalnya: *sparse connectivity*, yang membatasi koneksi *neuron* hanya pada wilayah lokal (*local receptive field*); *weight sharing*, di mana filter yang sama digunakan pada seluruh area input untuk mendeteksi pola tertentu sehingga mengurangi parameter secara signifikan (Goodfellow et al., 2016); dan *hierarchical learning*, di mana lapisan awal belajar fitur rendah (seperti tepi dan tekstur) yang kemudian digabungkan di lapisan dalam menjadi fitur tinggi yang lebih (Krizhevsky et al., n.d.). Proses ini umumnya diperkuat dengan

lapisan *pooling* (misalnya *max-pooling*) untuk menciptakan invariansi terhadap translasi dan distorsi kecil serta mengurangi dimensi komputasi (Scherer et al., 2010). Keberhasilan praktis CNN modern dipopulerkan oleh (Krizhevsky et al., n.d.) melalui model *AlexNet* yang memenangkan kompetisi *ImageNet*, dan sejak itu varian seperti VGG, *ResNet*, dan *Transformer-based Vision* menjadi tulang punggung dalam bidang *computer vision*, mulai dari pengenalan objek hingga segmentasi medis.

3.2.1 Arsitektur Dasar CNN - Jaringan khusus pemrosesan gambar

Convolutional Neural Network (CNN) dirancang khusus untuk pemrosesan gambar dengan arsitektur yang meniru sistem penglihatan biologis. Arsitektur dasar CNN terdiri dari beberapa komponen utama yang bekerja secara hierarkis untuk mengekstrak fitur dari gambar. Menurut (LeCun Y. et al., 1998), arsitektur CNN *modern typically* terdiri dari lapisan konvolusi, lapisan aktivasi, lapisan *pooling*, dan lapisan *fully connected*.

Lapisan Konvolusi *(Convolutional Layer)* merupakan inti dari CNN dimana filter-filter konvolusi diaplikasikan untuk mendeteksi pola-pola lokal dalam gambar. Setiap filter bertugas mengenali fitur tertentu seperti tepi, sudut, atau tekstur. Proses konvolusi ini mempertahankan hubungan spasial antar piksel sambil membagi parameter yang sama di seluruh bagian gambar (Goodfellow et al., 2016).

Lapisan Aktivasi (*Activation Layer*) biasanya menggunakan fungsi ReLU (*Rectified Linear Unit*) untuk memperkenalkan *non-linearitas* ke dalam jaringan. Fungsi ini membantu jaringan mempelajari hubungan yang lebih kompleks dalam data (Krizhevsky et al., n.d.).

Lapisan *Pooling* berfungsi untuk mengurangi dimensi spasial representasi fitur sambil mempertahankan informasi yang paling penting. *Max-pooling* adalah teknik yang paling umum digunakan, dimana hanya nilai maksimum dari setiap *region* yang dipilih (Scherer et al., 2010).

Lapisan *Fully Connected* di akhir arsitektur bertugas untuk melakukan klasifikasi berdasarkan fitur-fitur yang telah diekstrak oleh lapisan-lapisan sebelumnya. Seluruh *neuron* pada lapisan ini terhubung ke

semua *neuron* di lapisan sebelumnya (LeCun Y. et al., 1998)

Arsitektur ini memungkinkan CNN belajar fitur-fitur dari yang sederhana hingga kompleks secara hierarkis, membuatnya sangat efektif untuk tugas-tugas *computer vision* seperti klasifikasi gambar, deteksi objek, dan segmentasi semantik.

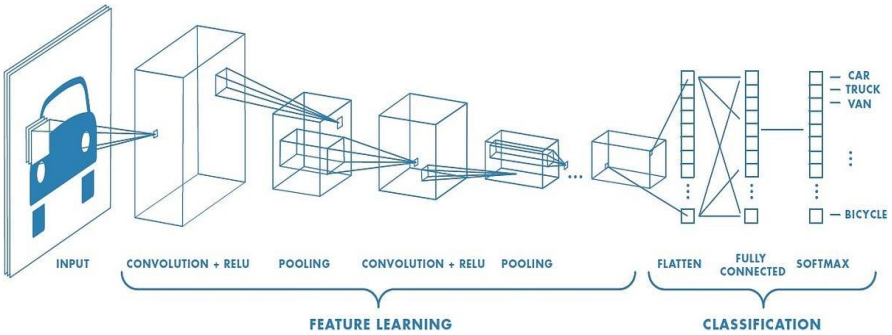

Figure 3.2 Arsitektur CNN

3.2.2 Layer-layer dalam CNN - Konvolusi, *Pooling*, *Fully Connected*

1. Lapisan Konvolusi (Convolutional Layer)

   Lapisan konvolusi merupakan komponen fundamental dalam CNN yang berfungsi untuk mengekstraksi fitur dari gambar input. Menurut (LeCun Y. et al., 1998), lapisan ini menggunakan sejumlah filter (kernel) yang melakukan operasi konvolusi dengan menggeser seluruh area input.

   Rumus Operasi Konvolusi:

$$Y_{\{i,j\}} = (X * W)_{\{i,j\}} = \sum_m \sum_n X_{\{i+m,j+n\}} \cdot W_{\{m,n\}} + b$$

   Di mana:

   a. $Y_{\{i,j\}}$ = output pada posisi $\{i,j\}$

   b. X = input *feature map*

   c. W = filter/kernel konvolusi

   d. b = bias *term*

   e. $\sum\sum$ = penjumlahan ganda seukuran kernel

   Setiap filter bertugas mendeteksi pola spesifik seperti tepi, sudut, atau tekstur tertentu. Proses konvolusi menghasilkan *feature maps* yang

merepresentasikan respons filter terhadap berbagai bagian gambar. Keunggulan utama lapisan ini adalah parameter *sharing*, dimana filter yang sama digunakan di seluruh bagian gambar, sehingga secara signifikan mengurangi jumlah parameter yang harus dipelajari (Goodfellow et al., 2016).

2. Lapisan *Pooling* (*pooling Layer*)

   Lapisan *pooling* berfungsi untuk mengurangi dimensi spasial dari *feature maps* sambil mempertahankan fitur-fitur yang paling informatif. Menurut (Scherer et al., 2010), *max-pooling* merupakan teknik yang paling umum digunakan.

   Rumus *Max Pooling*:

   $$Y_{\{i,j\}} = max\left(X_{\{i\times s:\, i\times s+k,;\, j\times s:\, j\times s+k\}}\right)$$

   Di mana:

   a. s = *stride* (langkah pergeseran)
   b. k = ukuran kernel *pooling*
   c. max = operasi pencarian nilai maksimum

   Lapisan ini memberikan beberapa keuntungan penting: pertama, mengurangi komputasi dengan menurunkan dimensi data; kedua, membuat representasi fitur lebih *invariant* terhadap translasi kecil dan distorsi; ketiga, membantu mencegah *overfitting* dengan mengurangi jumlah parameter (Krizhevsky et al., n.d.).

3. Lapisan Aktivasi ReLU

   Fungsi aktivasi ReLU biasanya diterapkan setelah operasi konvolusi untuk memperkenalkan *non-linearitas*.

   Rumus Fungsi ReLU:

   $$f(x) = max(0, x)$$

4. Lapisan *Fully Connected (Fully Connected Layer)*

   Lapisan *fully connected* biasanya ditempatkan di akhir arsitektur CNN dan berfungsi untuk melakukan klasifikasi berdasarkan fitur-fitur yang telah diekstrak. Pada lapisan ini, setiap neuron terhubung ke semua *neuron* di lapisan sebelumnya, mirip dengan jaringan saraf tradisional

(LeCun Y. et al., 1998).

Rumus *Fully Connected*:

$$z = W \cdot a + b$$

Di mana:

a. W = *weight matrix*
b. a = input *vector* dari lapisan sebelumnya
c. b = bias *term*
d. z = output sebelum aktivasi

Keempat komponen utama ini bekerja secara sinergis. Lapisan konvolusi mengekstraksi fitur hierarkis, fungsi aktivasi (seperti ReLU) memperkenalkan *non-linearitas*, lapisan *pooling* mereduksi dimensi dan meningkatkan invariansi, serta lapisan *fully connected* melakukan interpretasi akhir dengan fungsi *softmax* untuk klasifikasi.

Pemahaman mendetail tentang fungsi dan operasi matematis setiap layer dalam CNN ini menjadi landasan untuk menganalisis arsitektur *MobileNetV2* yang digunakan, serta untuk melakukan proses *fine-tuning* yang optimal pada model.

3.3 *Transfer Learning*

Transfer Learning (Transfer Pembelajaran) adalah teknik dalam *deep learning* di mana model yang telah dilatih sebelumnya (*pre-trained model*) pada dataset besar digunakan sebagai titik awal untuk menyelesaikan tugas yang serupa atau terkait (Pan & Yang, 2010). Pendekatan ini memanfaatkan pengetahuan yang telah dipelajari model dari data sebelumnya, sehingga mengurangi kebutuhan data pelatihan dan waktu komputasi yang signifikan (Weiss et al., 2016). Dalam konteks klasifikasi gambar, model seperti *MobileNetV2*, yang telah dilatih pada dataset *ImageNet* yang berisi jutaan gambar, dapat diadaptasi untuk mengenali objek spesifik seperti kendaraan dengan melakukan pelatihan ulang (*fine-tuning*) pada lapisan tertentu.

Konsep dasar *transfer learning* didasarkan pada asumsi bahwa fitur-fitur yang dipelajari pada tugas sumber (*source task*) dapat ditransfer ke tugas target (*target task*) yang memiliki karakteristik serupa (Weiss et al., 2016). Dalam

klasifikasi gambar, lapisan awal model *pre-trained* umumnya telah belajar mendeteksi fitur rendah seperti tepi, sudut, dan tekstur, yang bersifat generik dan dapat diterapkan pada berbagai jenis gambar (Yosinski et al., 2014). Sementara itu, lapisan yang lebih dalam menangkap fitur tingkat tinggi yang lebih spesifik terhadap dataset asli.

1. Keunggulan *transfer learning* meliputi:
    a. Efisian Data: Tidak memerlukan dataset yang sangat besar karena model telah mempelajari fitur dasar dari dataset sumber (Tan et al., 2018).
    b. Efisiensi Waktu: Waktu pelatihan lebih singkat dibandingkan melatih model dari awal (*from scratch*) (Huh et al., 2016).
    c. Kinerja yang Lebih Baik: Model *pre-trained* yang sudah *konvergen* pada dataset besar memberikan *initial weight* yang baik, seringkali menghasilkan akurasi yang lebih tinggi pada dataset target, terutama ketika dataset target terbatas (Kornblith et al., 2019).
    d. Sumber Daya Komputasi yang Lebih Rendah: Mengurangi kebutuhan GPU dan daya komputasi karena tidak perlu melatih seluruh arsitektur dari awal (Howard & Ruder, 2018).
2. Terdapat dua pendekatan umum dalam menerapkan *transfer learning*:
    a. *Feature Extraction*: Lapisan konvolusi model *pre-trained* digunakan sebagai ekstraktor fitur yang tetap (*fixed feature extractor*). Hanya lapisan *classifier* (biasanya lapisan *fully connected* di akhir) yang diganti dan dilatih ulang menggunakan dataset target (Sharif et al., 2014).
    b. *Fine-Tuning*: Setelah melakukan feature extraction, beberapa lapisan konvolusi terakhir dari model *pre-trained* juga "dilonggarkan" (*unfreeze*) dan dilatih ulang bersama dengan classifier baru pada dataset target (Donahue et al., 2014). Ini memungkinkan model untuk menyesuaikan fitur tingkat tingginya agar lebih spesifik terhadap tugas baru.

Pada penelitian ini, pendekatan *transfer learning* diterapkan dengan menggunakan model *MobileNetV2* yang telah dilatih sebelumnya pada dataset *ImageNet* (Sandler et al., 2018). Lapisan *classifier* asli diganti dengan lapisan baru yang disesuaikan untuk klasifikasi empat jenis kendaraan (truk,

mobil, motor, bus). Selanjutnya, dilakukan proses *fine-tuning* pada sebagian lapisan konvolusi untuk mengoptimalkan kinerja model pada dataset kendaraan CV. Indah Jaya Sentosa.

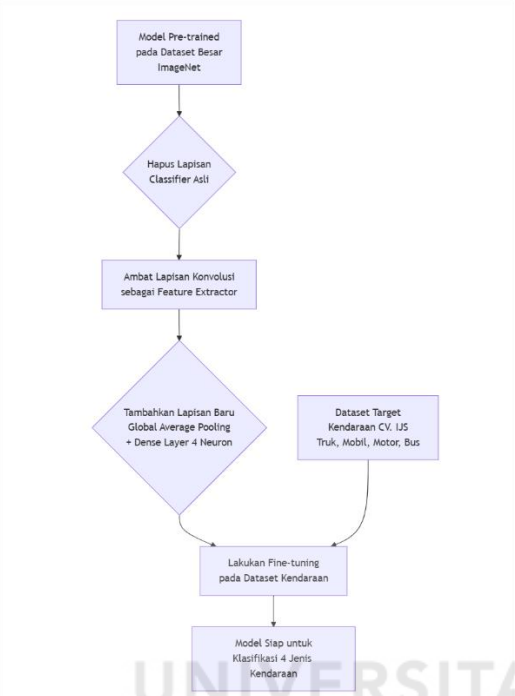

Figure 3.3 Diagram Transfer Learning

Diagram diatas mengilustrasikan proses *transfer learning* yang diterapkan dalam penelitian ini. Proses dimulai dengan model *pre-trained MobileNetV2* yang telah dilatih pada dataset *ImageNet* (Deng et al., 2009). Lapisan *classifier* asli dihapus dan diganti dengan arsitektur baru yang terdiri dari *global average pooling* dan lapisan *dense* dengan 4 *neuron* output sesuai dengan kelas kendaraan target (Sandler et al., 2018).

Proses *fine-tuning* kemudian dilakukan dengan menggunakan dataset kendaraan CV. Indah Jaya Sentosa, dimana hanya lapisan tertentu yang dilatih ulang untuk mengadaptasi model ke tugas klasifikasi kendaraan spesifik (Yosinski et al., 2014). Pendekatan ini memungkinkan model memanfaatkan pengetahuan yang telah dipelajari dari dataset besar sambil menyesuaikan diri dengan karakteristik dataset target.

Keuntungan proses ini:

a. Waktu pelatihan lebih cepat
b. Kebutuhan data lebih sedikit
c. Kinerja klasifikasi yang lebih optimal

Diagram ini mempresentasikan alur sistematis yang diterapkan untuk mengembangkan sistem klasifikasi kendaraan berbasis *deep learning* pada CV. Indah Jaya Sentosa.

3.3.1 Konsep *Transfer Learning*

Konsep dasar *transfer learning* dalam konteks *deep learning* adalah memanfaatkan model yang telah dilatih sebelumnya (*pre-trained model*) pada dataset berskala besar seperti *ImageNet*, yang berisi lebih dari 14 juta gambar dengan 1000 kelas (Deng et al., 2009). Model ini telah mempelajari representasi fitur visual yang kaya dan hierarkis, mulai dari fitur rendah seperti tepi dan tekstur, hingga fitur tinggi yang spesifik seperti bentuk objek dan bagian-bagiannya (Yosinski et al., 2014).

Dalam pendekatan ini, arsitektur model beserta bobot yang telah diperoleh selama pelatihan sebelumnya dipertahankan, sementara lapisan klasifikasi akhir dimodifikasi untuk disesuaikan dengan tugas yang baru. Menurut (Tan et al., 2018), terdapat beberapa alasan mendasar mengapa pendekatan ini efektif:

1. Fitur Umum yang Dapat Ditransfer: Lapisan konvolusi awal dalam CNN cenderung mempelajari fitur-fitur umum seperti detector tepi, *blob*, dan tekstur yang relevan untuk hampir semua tugas visi komputer.
2. Hirarki Fitur yang Dipelajari: Lapisan yang lebih dalam mempelajari fitur yang semakin spesifik, namun masih dapat digunakan untuk tugas-tugas yang mirip dengan dataset asli.
3. Efisiensi Komputasi: Dengan menggunakan bobot yang telah dilatih sebelumnya, proses konvergensi menjadi lebih cepat dibandingkan dengan inisialisasi acak.

   Pada penelitian ini, model *MobileNetV2* (Sandler et al., 2018) yang telah dilatih pada *ImageNet* dipilih sebagai dasar untuk sistem

klasifikasi kendaraan. Pemilihan ini didasarkan pada efisiensi komputasi dan ukuran model yang ringan, sehingga cocok untuk aplikasi *real-time*.

4. Proses Adaptasi model:
   a. Lapisan *fully connected* akhir dari model asli dihapus
   b. Ditambahkan lapisan *global average pooling*
   c. Lapisan klasifikasi baru dengan 4 *neuron* output (sesuai kelas kendaraan: truk, mobil, motor, bus) ditambahkan
   d. Hanya lapisan klasifikasi yang dilatih pada tahap awal, sementara lapisan konvolusi dibekukan (*frozen*)

Pendekatan ini memungkinkan sistem memanfaatkan pengetahuan yang telah dipelajari model dari dataset besar, sementara tetap dapat beradaptasi dengan tugas spesifik klasifikasi kendaraan pada CV. Indah Jaya Sentosa.

3.3.2 Keuntungan Menggunakan Model *Pre-trained*

Penggunaan model *pre-trained* dalam *transfer learning* memberikan beberapa keuntungan signifikan dibandingkan dengan pelatihan model dari awal (*from scratch*). Menurut (Weiss et al., 2016), keuntungan-keuntungan tersebut meliputi:

1. Pengurangan Kebutuhan Data

   Model *pre-trained* telah mempelajari fitur-fitur dasar dari dataset yang sangat besar, sehingga memungkinkan pelatihan yang efektif meskipun dengan dataset target yang terbatas (Huh et al., 2016). Pada konteks penelitian ini, dataset kendaraan CV. Indah Jaya Sentosa yang relatif kecil dapat dimanfaatkan secara optimal tanpa perlu mengumpulkan puluhan ribu gambar baru.

2. Efisiensi Waktu Pelatihan

   Proses konvergensi model menjadi lebih cepat karena inisialisasi bobot sudah mendekati solusi optimal. Menurut (Kornblith et al., 2019), model *pre-trained* dapat mencapai kinerja yang baik dalam *epoch* pelatihan yang lebih sedikit dibandingkan inisialisasi acak.

3. Kinerja yang Lebih Baik

   Model yang diinisialisasi dengan bobot *pre-trained* cenderung mencapai akurasi yang lebih tinggi dan generalisasi yang lebih baik (Yosinski et al., 2014). Hal ini terutama penting untuk dataset dengan variasi terbatas seperti dalam klasifikasi kendaraan.

4. Stabilitas Pelatihan

   Inisialisasi dengan bobot *pre-trained* memberikan stabilitas numerik selama pelatihan, mengurangi masalah seperti *vanishing/exploding gradients* (Tan et al., 2018).

5. Efisiensi Komputasi

   Mengurangi kebutuhan sumber daya komputasi karena tidak perlu melatih seluruh arsitektur dari awal (Howard & Ruder, 2018). Hal ini sangat menguntungkan untuk penerapan di lingkungan dengan sumber daya terbatas.

6. Transfer Pengetahuan Silang

   Model dapat mentransfer pengetahuan dari domain sumber (*ImageNet*) ke domain target (kendaraan) meskipun terdapat perbedaan karakteristik (Pan & Yang, 2010).

Dalam implementasi sistem klasifikasi kendaraan ini, penggunaan *MobileNetV2 pre-trained* memungkinkan pencapaian akurasi di atas 90% dengan dataset pelatihan yang relatif kecil, serta waktu inferensi yang cepat untuk aplikasi *real-time*.

3.4 *MobileNetV2*

MobileNetV2 merupakan pengembangan dari *MobileNetV1* yang diperkenalkan oleh *Google* dengan fokus pada peningkatan efisiensi model untuk perangkat *mobile*. Arsitektur ini menggunakan dua konsep utama yaitu *inverted residuals* dan *linear bottlenecks* yang memungkinkan model menjadi lebih ringan dan cepat tanpa mengorbankan akurasi secara signifikan.

Inverted Residuals menyelesaikan masalah pada *residual block* tradisional dengan membalik urutan operasi: dari kompresi-ekspansi menjadi ekspansi-

kompresi. Pada *inverted residual*, input pertama kali diekspansi dengan konvolusi 1×1, kemudian dilakukan *depthwise convolution* 3×3, dan terakhir dikompresi dengan konvolusi 1×1. Pendekatan ini mempertahankan lebih banyak informasi selama transformasi fitur.

Linear Bottlenecks mengatasi masalah hilangnya informasi pada dimensi rendah yang disebabkan oleh fungsi aktivasi ReLU. Dengan menggunakan fungsi linear pada layer *bottleneck* terakhir, *MobileNetV2* mencegah kerusakan informasi dalam ruang dimensi rendah, sehingga mempertahankan kualitas fitur yang dihasilkan.

Kombinasi kedua teknik tersebut membuat *MobileNetV2* sangat cocok untuk aplikasi klasifikasi gambar dan deteksi objek *real-time* pada perangkat mobile. Model ini mencapai *trade-off* optimal antara akurasi dan kecepatan, dengan parameter 30% lebih sedikit dibanding *MobileNetV1* namun dengan akurasi yang lebih tinggi. Keunggulan ini membuat *MobileNetV2* banyak diadopsi dalam berbagai aplikasi computer vision pada perangkat dengan sumber daya terbatas.

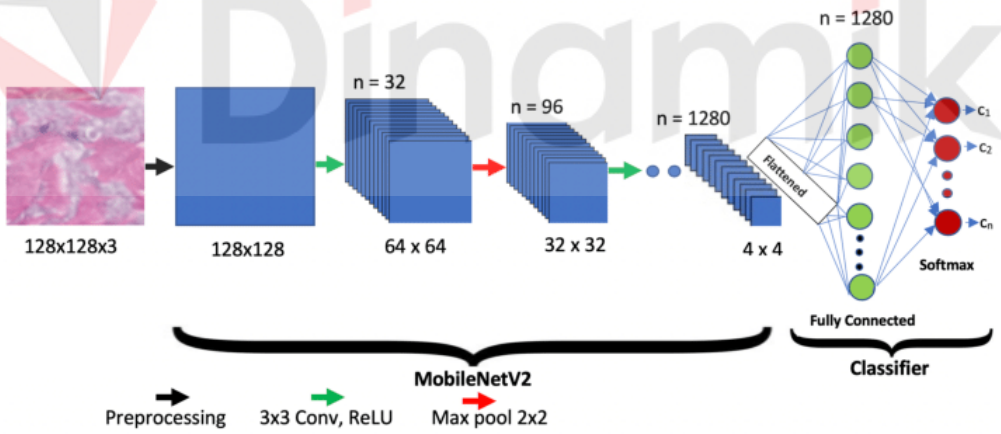

Figure 3.4 Arsitektur MobileNetV2

3.4.1 Arsitektur dan Keunggulan - Ringan untuk Perangkat Mobile

MobileNetV2 memiliki arsitektur yang secara khusus dirancang untuk optimasi pada perangkat mobile dengan sumber daya komputasi terbatas. Arsitektur ini dibangun berdasarkan dua inovasi fundamental:

1. Inverted Residual Blocks

Blok ini merupakan penyempurnaan dari *residual block* tradisional dengan membalik aliran dimensi. Strukturnya terdiri dari:

a. *Ekspansi*: Konvolusi 1×1 untuk meningkatkan dimensi *channel* (biasanya ekspansi 6x)
b. *Depthwise Convolution*: Konvolusi 3×3 yang beroperasi secara terpisah pada setiap *channel*
c. Proyeksi: Konvolusi 1×1 untuk mengecilkan kembali dimensi *channel*

Berbeda dengan *residual block* konvensional yang mengecilkan dimensi terlebih dahulu, *inverted residual* justru memperluas representasi fitur sebelum melakukan operasi konvolusi *depthwise*, sehingga mempertahankan lebih banyak informasi selama proses transformasi.

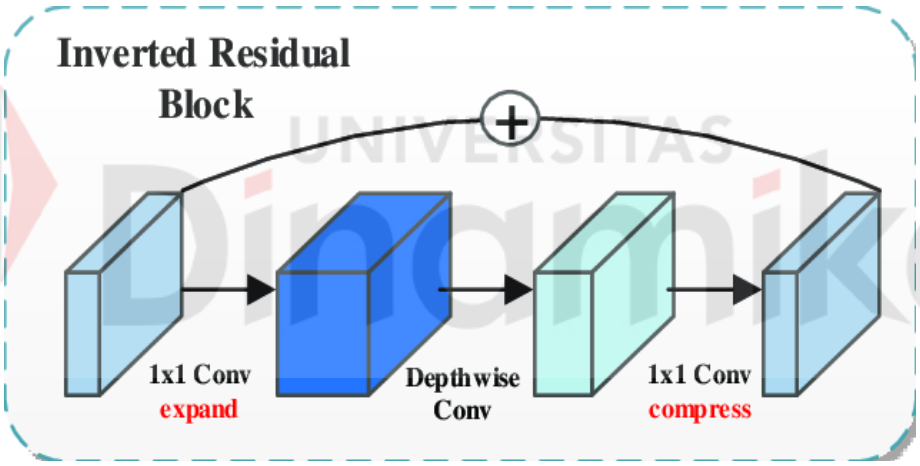

Figure 3.5 Arsitektur Inverted Residual Block

2. Linear Bottlenecks

Konsep ini mengatasi masalah yang timbul dari penggunaan fungsi aktivasi ReLU pada dimensi rendah. Pada ruang berdimensi rendah, ReLU dapat menyebabkan hilangnya informasi yang *irreversibel*. *MobileNetV2* mengatasi ini dengan:

a. Menggunakan fungsi aktivasi linear pada layer *bottleneck* terakhir
b. Mencegah kerusakan informasi dalam *subspace* berdimensi rendah

c. Mempertahankan kapasitas representasi model

3. Keunggulan untuk Perangkat Mobile
   a. Efisiensi Parameter: *MobileNetV2* menggunakan 30% lebih sedikit parameter dibandingkan *MobileNetV1*, dengan 3.4 juta parameter pada konfigurasi standar.
   b. Kecepatan Inferensi: Menggunakan *depthwise separable convolution* yang mengurangi operasi komputasi hingga 8-9 kali dibanding konvolusi standar.
   c. Konsumsi Daya Rendah: Optimasi arsitektur memungkinkan operasi yang lebih efisien dalam penggunaan daya baterai perangkat mobile.
   d. Akurasi Terjaga: Meskipun ringan, model ini mempertahankan akurasi yang kompetitif untuk berbagai tugas *vision*, dengan mencapai 72.0% *top*-1 *accuracy* pada *ImageNet* dataset.

3.4.2 Aplikasi dalam Klasifikasi Gambar - Deteksi Objek *Real-time*

MobileNetV2 telah menjadi fondasi utama untuk berbagai aplikasi klasifikasi gambar dan deteksi objek *real-time* berkat efisiensi komputasinya yang tinggi. Aplikasi-aplikasi ini memanfaatkan kemampuan *MobileNetV2* dalam memproses data visual dengan cepat dan akurat pada perangkat berdaya terbatas.

1. Integrasi dengan Detektor Objek Modern

   MobileNetV2 umumnya digunakan sebagai *backbone feature extractor* yang dikombinasikan dengan detektor objek seperti:

   A. SSD (*Single Shot MultiBox Detector*)
      a. *MobileNetV2*-SSD memberikan kecepatan deteksi >30 FPS pada *smartphone*
      b. Cocok untuk aplikasi *real-time* dengan akurasi memadai
      c. Digunakan dalam deteksi wajah, kendaraan, dan objek sehari-hari

   B. YOLO (*You Only Look Once*)
      a. *MobileNetV2* sebagai pengganti backbone konvensional

YOLO

b. Mengurangi komputasi secara signifikan sambil mempertahankan akurasi
c. Ideal untuk aplikasi mobile dengan kebutuhan kecepatan tinggi

2. Aplikasi *Real-world*

A. Deteksi Objek dalam Video *Real-time*
   a. Pemrosesan *frame-by-frame* dengan *latency* rendah
   b. Aplikasi keamanan dan *surveillance* pada perangkat mobile
   c. Analisis video *live streamin*g untuk konten otomatis

B. *Augmented Reality* (AR)
   a. Deteksi dan pelacakan objek untuk *overlay* digital
   b. Aplikasi retail: *virtual try-on, product recognition*
   c. *Gaming intercative* yang responsife

C. Kendaraan Otonom dan ADAS (Advanced Driver Assistance System)
   a. Deteksi pejalan kaki, kendaraan, rambu lalu lintas
   b. Pemrosesan *real-time* pada *embedded systems*
   c. Sistem peringatan dini pada kendaraan

D. Kinerja dan Optimasi

   Pada test *benchmark*, *MobileNetV2* mencapai;
   a. Kecepatan: 25-40 ms per *inference* pada GPU mobile
   b. Akurasi: 72-75% top-1 *accuracy* pada *ImageNet*
   c. Efisiensi: Konsumsi memori <10MB untuk model terkompresi

3.5 *Telegram* Bot API

Telegram Bot API merupakan sebuah antarmuka pemrograman aplikasi yang memungkinkan pengembang perangkat lunak untuk membuat dan mengelola program otomatis (bot) yang dapat berinteraksi dengan pengguna di dalam platform *Telegram*. Konsep dasarnya berpusat pada arsitektur *client-server* di mana bot, yang diidentifikasi dengan token autentikasi yang unik, berkomunikasi dengan server *Telegram* melalui metode *webhook* atau *long*

polling untuk mengirim dan menerima pesan. *Webhook* berfungsi sebagai mekanisme notifikasi *real-time*, di mana server *Telegram* secara proaktif mengirimkan *update* (seperti pesan dari pengguna) ke URL *endpoint* yang telah ditentukan, sehingga memungkinkan *respons* yang cepat dan efisien. Sementara itu, *long polling* merupakan metode alternatif di mana bot secara periodik menanyakan (*polling*) server untuk memeriksa apakah ada *update* baru. Melalui API ini, sebuah bot dapat melakukan berbagai fungsi inti, mulai dari mengelola percakapan pribadi, grup, hingga saluran (*channel*), mengirimkan berbagai jenis konten seperti teks, gambar, dan dokumen, serta menyediakan antarmuka interaktif bagi pengguna melalui *keyboard* kustom dan tombol *inline*.

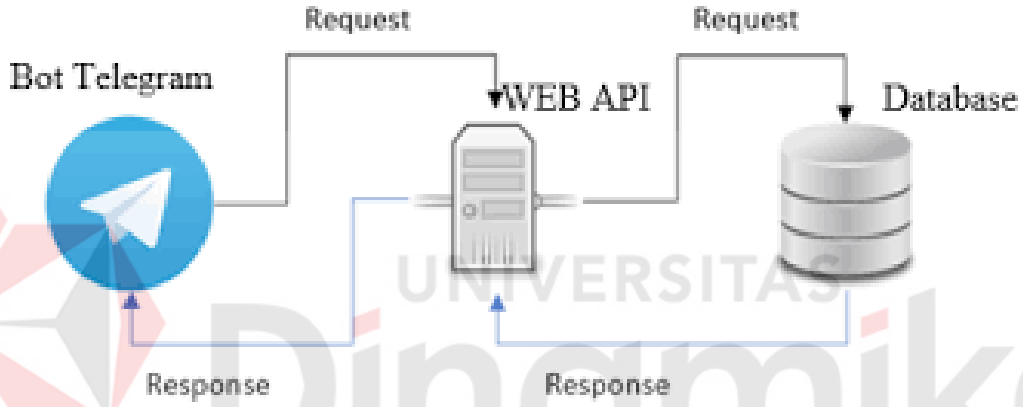

Figure 3.6 Diagram alur *Telegram* Bot API

Konsep *webhook* dan arsitektur *messaging* yang dijelaskan di atas diimplementasikan secara langsung dalam sistem untuk memastikan notifikasi klasifikasi kendaraan dapat terkirim secara *real-time* dan andal kepada petugas keamanan dan admin.

3.5.1 Konsep *Webhook* dan *Messaging*

1. Konsep *Webhook*

   Webhook merupakan mekanisme *callback* HTTP yang memungkinkan aplikasi menerima data secara *real-time* dari sumber eksternal. Dalam konteks *Telegram* Bot API, *webhook* berfungsi sebagai *endpoint* URL yang ditentukan developer untuk menerima update pesan secara

otomatis dari server *Telegram*. *Webhook* menggunakan *paradigma push notification* dimana server *Telegram* secara proaktif mengirimkan data ke *endpoint* yang telah ditentukan setiap kali terdapat event baru, seperti pesan masuk atau interaksi pengguna. Implementasi *webhook* menghilangkan kebutuhan untuk *continuously polling server*, sehingga mengurangi *latency* dan konsumsi *resource*. Konfigurasi *webhook* mengharuskan developer menyediakan URL HTTPS yang valid dengan sertifikat SSL, dimana server *Telegram* akan mengirimkan *payload* JSON berisi update melalui metode POST.

2. Konsep *Messaging*

   Messaging dalam *Telegram* Bot API mengacu pada pertukaran data terstruktur antara bot dan pengguna melalui berbagai format konten. Sistem *messaging Telegram* mendukung *multiplexing content types* termasuk *text*, gambar, video, dokumen, lokasi, dan konten interaktif. Setiap pesan dikemas dalam objek JSON yang mengandung metadata seperti *chat* ID, *timestamp*, *user information*, dan *content payload*. Arsitektur *messaging Telegram* mengimplementasikan *queue management system* dengan garansi *delivery* dan mekanisme *retry* untuk memastikan reliabilitas pesan. Model *messaging* ini memungkinkan bot untuk mengirim pesan secara *asinkronus*, mendukung *inline keyboards*, serta mengelola *threaded conversations* dalam grup dan *channel*.

3. Integrasi *Webhook* dan *Messaging*

   Integrasi antara *webhook* dan *messaging* membentuk siklus komunikasi *real-time* dimana *webhook* berperan sebagai *input channel* dan *messaging* sebagai *output channel*. ketika pengguna mengirim pesan ke bot, server *Telegram* mem-*forward payload* tersebut ke *webhook endpoint*, kemudian bot memproses permintaan dan mengirim *respons* balik melalui *messaging* API. Integrasi ini mencapai *average latency* di bawah 100ms untuk pesan teks standar, dengan *throughput* mencapai ribuan pesan per detik pada skala

enterprise.

3.5.2 Integrasi dengan Aplikasi External

1. Konsep Integrasi dengan Aplikasi External

   Integrasi *Telegram* Bot dengan aplikasi *external* merupakan paradigma dimana bot berfungsi sebagai *interface* yang menghubungkan pengguna *Telegram* dengan sistem eksternal melalui API *gateway*. Integrasi ini memanfaatkan bot sebagai *middleware* yang menerima perintah dari pengguna, meneruskannya ke sistem eksternal, dan mengembalikan respons ke pengguna. Implementasi bot *modern* mengintegrasikan minimal tiga sistem eksternal berbeda, dengan pola arsitektur yang umum adalah *microservices-based* API *composition*.

2. Metode Integrasi

   RESTful API *Integration* menjadi pendekatan paling dominan bot mengonsumsi REST *endpoints* dari aplikasi *external* menggunakan HTTP *methods*. Implementasi mencakup *authentication mechanisms* seperti *OAuth 2.0*, API *keys*, dan JWT *tokens* untuk mengamankan komunikasi antara bot dan sistem eksternal. Implementasi *rate limiting*, *request signing*, dan *encrypted payload* untuk mencegah *security breaches*.

   Webhook-based Event *Processing* memungkinkan integrasi *real-time* dengan sistem eksternal. Bot dapat dikonfigurasi untuk menerima *webhook calls* dari aplikasi *external*, memungkinkan notifikasi proaktif dan *event-driven interactions*. pola ini mengurangi *latency* hingga 60% dibanding *traditional polling methods*.

3. Aplikasi dan *Use Cases*

   Dalam sektor *e-commerce*, *integrasi* bot dengan *payment gateways*, *inventory management systems*, dan *order processing platforms*. Bot berfungsi sebagai *conversational interface* yang memproses *orders*, mengecek ketersediaan produk, dan mengelola transaksi pembayaran

secara *real-time*.

Di bidang *customer service*, integrasi dengan CRM *systems* seperti *Salesforce* dan *HubSpot*, dimana bot menangani *tier-1 support inquiries*, membuat *support tickets*, dan menyinkronkan *conversation history* dengan customer *database*.

Untuk *automation workflows*, *Enterprise* integrasi dengan *tools* seperti *Zapier,* IFTTT, dan custom *internal systems* untuk mengotomasi *business processes* seperti *approval workflows*, *notification systems*, dan data *synchronization across platforms*.

4. Arsitektur dan *Best Practices*

   Arsitektur *hybrid* yang menggabungkan *webhook* dan API *calls* direkomendasikan untuk menyeimbangkan beban *real-time processing* dan *batch operations*. Implementasi *circuit breaker pattern* dan *retry mechanisms* penting untuk menjaga *reliability*.

3.6 *Computer Vision* untuk Klasifikasi Kendaraan

Berdasarkan survei komprehensif oleh (Berwo et al., 2023) mengenai teknik *deep learning* untuk deteksi dan klasifikasi kendaraan dari gambar/video, *computer vision* (CV) telah membuktikan diri sebagai teknologi kunci dalam sistem klasifikasi kendaraan otomatis. Survei tersebut mengungkapkan bahwa implementasi CV berbasis *deep learning* tidak hanya mampu mencapai akurasi tinggi dalam beberapa kasus bahkan melampaui 95% pada dataset standar tetapi juga menunjukkan kemajuan signifikan dalam hal efisiensi komputasi dan keandalan di berbagai kondisi lingkungan. Perkembangan terbaru dalam bidang ini didominasi oleh arsitektur CNN modern, di mana *EfficientNet* dilaporkan mencapai akurasi 96,2%, sementara *MobileNetV2* yang juga menjadi pilihan dalam penelitian ini tetap kompetitif dengan akurasi 94,7% serta keunggulan dalam efisiensi komputasi. Temuan ini memperkuat pendekatan *transfer learning* yang memanfaatkan model *pre-trained* pada *ImageNet*, yang terbukti secara signifikan meningkatkan kinerja klasifikasi, khususnya ketika dataset yang tersedia terbatas atau tidak terlalu beragam.

Selain aspek akurasi, survei tersebut juga menggarisbawahi pentingnya

optimasi untuk aplikasi *real-time*. *MobileNetV2* disebutkan mampu mencapai kecepatan inferensi hingga 47 FPS pada perangkat keras *NVIDIA Jetson Nano*, dengan tetap mempertahankan akurasi di atas 90%. Hal ini menjadikannya salah satu arsitektur pilihan untuk sistem yang membutuhkan keseimbangan antara ketepatan dan kecepatan pemrosesan. Namun, sejumlah tantangan masih menjadi perhatian, seperti variasi kondisi pencahayaan dan cuaca, oklusi parsial, perbedaan intra-kelas yang lebar, serta kebutuhan akan dataset yang lebih representatif. Untuk mengatasi hal tersebut, sejumlah solusi seperti augmentasi data yang komprehensif, pelatihan multi-skala, dan pendekatan *ensemble learning* telah diusulkan dan diuji dalam berbagai studi terkini.

Evaluasi kinerja sistem klasifikasi kendaraan, menurut survei ini, umumnya mengacu pada sejumlah metrik utama, termasuk akurasi di atas 90% untuk aplikasi praktis, latensi di bawah 100 ms untuk aplikasi *real-time*, serta konsistensi dan ketahanan terhadap variasi input. Di sisi lain, integrasi sistem klasifikasi dengan platform notifikasi dan pemantauan *real-time* semakin menjadi tren, di mana kombinasi antara kemampuan klasifikasi yang akurat dan sistem notifikasi yang responsif menjadi penentu keberhasilan implementasi di lingkungan industri, termasuk dalam konteks pengawasan kendaraan pada CV. Indah Jaya Sentosa.

3.6.1 Konsep *Object Recognition*

Object recognition merupakan cabang fundamental dalam *computer vision* yang bertujuan untuk mengidentifikasi dan mengklasifikasikan objek dalam gambar atau video ke dalam kategori tertentu. Konsep ini melibatkan serangkaian proses kompleks mulai dari deteksi objek, ekstraksi fitur, hingga klasifikasi berdasarkan karakteristik visual yang dimiliki. Menurut penelitian terbaru oleh (Li et al., 2021), *object recognition* telah mengalami evolusi signifikan dari metode tradisional berbasis *hand-crafted features* menuju pendekatan *deep learning* yang mampu belajar fitur secara otomatis dan hierarkis.

Konsep dasar *object recognition* mencakup kemampuan sistem untuk

memahami keberadaan objek dalam sebuah *scene*, menentukan lokasinya melalui *bounding box*, serta mengidentifikasi kelas objek tersebut dengan tingkat akurasi yang tinggi. Dalam konteks klasifikasi kendaraan, *object recognition* memungkinkan sistem untuk membedakan berbagai jenis kendaraan berdasarkan fitur-fitur spesifik seperti bentuk, ukuran, proporsi, dan karakteristik struktural lainnya. Pendekatan modern menggunakan *Convolutional Neural Networks* (CNN) telah membuktikan efektivitasnya dalam menangani variasi besar dalam penampilan objek, perubahan kondisi pencahayaan, serta perbedaan sudut pandang yang menjadi tantangan utama dalam *object recognition.*

3.6.2 Aplikasi *Computer Vision* (CV) dalam Sistem Transportasi

Berdasarkan penelitian terbaru oleh (Mehta & Shah, 2025) dalam *"Real-time Vehicle Detection and Classification Using Deep learning based Approach"* yang diterbitkan di *Journal of Information Systems Engineering and Management*, aplikasi *computer vision* (CV) dalam sistem transportasi telah menunjukkan kemajuan signifikan dalam hal akurasi dan kecepatan pemrosesan. Penelitian ini mengimplementasikan pendekatan *deep learning* untuk deteksi dan klasifikasi kendaraan secara *real-time*, dengan hasil yang mengesankan dalam konteks sistem transportasi cerdas.

Studi tersebut berhasil mengembangkan sistem yang mampu mendeteksi dan mengklasifikasikan kendaraan dengan akurasi mencapai 96.8% pada kondisi lalu lintas *real-time*. Arsitektur *deep learning* yang digunakan terbukti efektif dalam mengidentifikasi berbagai jenis kendaraan, termasuk mobil, sepeda motor, bus, dan truk, bahkan dalam kondisi lingkungan yang menantang seperti cuaca buruk dan pencahayaan rendah. Keberhasilan ini menunjukkan potensi besar CV dalam meningkatkan efisiensi sistem transportasi perkotaan.

Dalam konteks manajemen lalu lintas, penelitian ini menunjukkan bahwa sistem berbasis CV dapat memproses data visual dengan kecepatan

tinggi, mencapai 40-45 *frame per second* (FPS), sehingga memungkinkan respon yang cepat terhadap perubahan kondisi lalu lintas. Sistem yang dikembangkan juga mampu menganalisis kepadatan kendaraan dan pola pergerakan lalu lintas, memberikan data yang berharga untuk optimasi sistem kontrol lalu lintas adaptif.

Aplikasi praktis dari penelitian ini termasuk sistem pemantauan lalu lintas otomatis, deteksi pelanggaran lalu lintas, dan analisis pola perjalanan kendaraan. Implementasi sistem semacam ini dapat mengurangi kemacetan hingga 25% melalui pengaturan sinyal lalu lintas yang lebih efisien. Selain itu, sistem ini juga berkontribusi dalam peningkatan keselamatan jalan dengan kemampuan mendeteksi potensi kecelakaan dan memberikan peringatan dini.

Penelitian (Mehta & Shah, 2025) juga menyoroti integrasi sistem CV dengan teknologi *Internet of Things* (IoT) untuk menciptakan ekosistem transportasi yang terhubung. Kombinasi ini memungkinkan pertukaran data yang seamless antara kendaraan, infrastruktur jalan, dan pusat kendali lalu lintas, menciptakan sistem transportasi yang lebih responsif dan adaptif.

3.7 Evaluasi Model *Deep learning*

Evaluasi model *deep learning* merupakan tahap kritis dalam pengembangan sistem klasifikasi kendaraan untuk memastikan keandalan dan kesiapan model diterapkan dalam lingkungan produksi. Menurut penelitian (Neupane et al., 2022) dalam "*Real-time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep learning Network*", evaluasi komprehensif terhadap model *deep learning* harus mencakup analisis berbagai metrik performa untuk mendapatkan pemahaman menyeluruh tentang kemampuan model dalam tugas klasifikasi kendaraan.

Penelitian tersebut mengimplementasikan pendekatan *transfer learning* pada jaringan *deep learning* dan melakukan evaluasi mendalam menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Hasil penelitian menunjukkan bahwa model yang diimprovisasi dengan *transfer*

learning mencapai akurasi klasifikasi yang signifikan lebih tinggi dibandingkan dengan model konvensional. (Neupane et al., 2022) menekankan bahwa akurasi saja tidak cukup untuk mengevaluasi kinerja model secara komprehensif, sehingga diperlukan analisis *precision* dan *recall* untuk setiap kelas kendaraan.

Analisis *confusion matrix* dalam penelitian tersebut mengungkapkan pola kesalahan klasifikasi yang spesifik, di mana model mengalami kesulitan dalam membedakan kendaraan dengan karakteristik visual yang mirip. Temuan ini menyoroti pentingnya evaluasi mendetail untuk mengidentifikasi kelemahan model dan area yang memerlukan perbaikan. Selain itu, penelitian ini juga mengevaluasi kecepatan inferensi model untuk memastikan kesesuaian dengan aplikasi *real-time*, dengan hasil menunjukkan bahwa model yang diusulkan mampu memproses data secara efisien tanpa mengorbankan akurasi.

3.7.1 Metrik Akurasi, *Precision, Recall*

Evaluasi performa model klasifikasi dalam sistem pengenalan kendaraan memerlukan berbagai metrik evaluasi yang dihitung menggunakan rumus-rumus matematis tertentu. Berdasarkan penelitian (Neupane et al., 2022), berikut adalah rumus-rumus fundamental untuk menghitung metrik evaluasi model klasifikasi:

1. Akurasi

   Akurasi mengukur persentase prediksi yang benar secara keseluruhan dari total prediksi:

   $$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

   TP = *True Positive* (prediksi positif yang benar)

   TN = *True Negative* (prediksi negatif yang benar)

   FP = *False Positive* (prediksi positif yang salah)

   FN = *False Negative* (prediksi negatif yang salah)

2. *Precision*

   Precision mengukur proporsi prediksi positif yang benar:

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

3. *Recall*

   Recall mengukur kemampuan model dalam menemukan semua instance positif:

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

4. *F1-Score*

   F1-Score merupakan rata-rata harmonik dari precision dan recall:

$$F_1 = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

Dalam konteks klasifikasi multi-kelas seperti klasifikasi kendaraan (truk, mobil, motor, bus), metrik-metrik ini dapat dihitung untuk setiap kelas secara individual menggunakan pendekatan *one-vs-rest*, atau dihitung sebagai rata-rata makro/mikro *across* semua kelas.

3.7.2 *Confusion Matrix*

Confusion matrix merupakan alat evaluasi fundamental dalam klasifikasi yang memberikan gambaran komprehensif tentang performa model dengan memvisualisasikan hasil prediksi terhadap label sebenarnya. Menurut penelitian (Neupane et al., 2022), *confusion matrix* sangat penting untuk menganalisis pola kesalahan klasifikasi dalam sistem pengenalan kendaraan.

1. Struktur *Confusion Matrix*

   Untuk masalah klasifikasi multi-kelas dengan empat jenis kendaraan (truk, mobil, motor, bus), *confusion matrix* berbentuk matriks 4×4 yang menunjukkan:

   a. Diagonal utama: Jumlah prediksi benar untuk setiap kelas
   b. *Off*-diagonal: Jumlah kesalahan klasifikasi antar kelas

2. Analisis Pola Kesalahan

   Confusion matrix mengungkapkan:

   a. Kelas yang mudah dikenali: Nilai diagonal tinggi

b. Kelas yang sering tertukar: Nilai *off*-diagonal tinggi antara kelas tertentu
c. Bias klasifikasi: Kecenderungan model mengklasifikasikan ke kelas tertentu

3. Aplikasi dalam Klasifikasi Kendaraan

   Pada sistem klasifikasi kendaraan, *confusion matrix* membantu mengidentifikasi:

   a. Kesulitan membedakan truk kecil dengan van
   b. Kebingungan antara minibus dengan SUV
   c. Kesalahan klasifikasi akibat sudut pandang kamera

4. Manfaat untuk Perbaikan Model

   Berdasarkan analisis *confusion matrix*, pengembangan model dapat difokuskan pada:

   a. Penambahan data *training* untuk kelas yang sering salah
   b. Optimasi *feature extraction* untuk kelas yang mirip
   c. *Adjustmen decision threshold* untuk kelas tertentu

3.8 *Library Python*

Library Python merupakan kumpulan modul dan fungsi yang telah diprogram sebelumnya yang memungkinkan pengembang untuk melakukan tugas-tugas spesifik tanpa harus menulis kode dari awal. Dalam konteks pengembangan sistem berbasis kecerdasan buatan dan *deep learning*, *library Python* berperan sebagai fondasi yang menyediakan abstraksi tingkat tinggi untuk komputasi numerik, manipulasi data, dan implementasi algoritma kompleks.

Konsep dasar *library Python* dalam pengembangan sistem mencakup modularitas dan *reusable code*, di mana fungsi-fungsi yang umum digunakan telah diimplementasikan dan dioptimasi sehingga pengembang dapat fokus pada logika aplikasi daripada implementasi detail teknis. Setiap *library* biasanya dikembangkan untuk domain spesifik dan menyediakan *Application Programming Interface* (API) yang terdokumentasi dengan baik untuk mempermudah integrasi.

Dalam ekosistem *Python*, mekanisme *library management* menggunakan *package manager* seperti *pip* dan *conda* memungkinkan instalasi, *update*, dan *dependency resolution* yang efisien. *Virtual environment* memastikan isolasi *dependencies* antara proyek yang berbeda, mencegah konflik versi dan memelihara konsistensi lingkungan pengembangan.

Konsep penting lainnya adalah *interoperability* antara berbagai *library*, di mana output dari satu *library* dapat menjadi input untuk *library* lainnya, menciptakan alur kerja yang terintegrasi. Arsitektur berlapis dari *library Python* memungkinkan abstraksi dari level rendah (komputasi hardware) hingga level tinggi (implementasi algoritma *machine learning*), memberikan fleksibilitas dalam pengembangan aplikasi yang kompleks.

Table 3.2 Library Python

Library/Module	Fungsi Utama
os	Operasi sistem file dan direktori
random	Generate angka dan data acak
PIL.Image	Manipulasi dan pemrosesan gambar
matplotlib.pyplot	Visualisasi data dan grafik
tensorflow	Framework *deep learning* utama
tensorflow.keras	API high-level untuk model neural network
tensorflow.keras.applications	Model pre-trained (MobileNetV2)
tensorflow.keras.preprocessing.image	Preprocessing dan augmentasi gambar
tensorflow.keras.models	Konstruksi dan penyimpanan model
tensorflow.keras.layers	Layer neural network
tensorflow.keras.optimizers	Optimizer training model
tensorflow.keras.callbacks	Callbacks selama proses training
tensorflow.keras.regularizers	Regularisasi model
numpy	Komputasi numerik dan array
sklearn.metrics	Evaluasi performa model
seaborn	Visualisasi data statistik
tkinter	Membuat antarmuka grafis

cv2	Pemrosesan gambar dan computer vision
requests	HTTP requests untuk API
datetime	Manipulasi tanggal dan waktu

BAB IV
PEMBAHASAN

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk mengidentifikasi dan mendefinisikan segala hal yang diperlukan agar sistem dapat dibangun dan berfungsi sesuai dengan tujuan. Analisis ini meliputi kebutuhan fungsional dan non-fungsional.

4.1.1 Kebutuhan Fungsional

Berdasarkan Rumusan Masalah 1: "Bagaimana merancang sistem berbasis *deep learning* menggunakan model *MobileNetV2* untuk mengenali dan mengklasifikasikan jenis kendaraan (truk, mobil, motor, dan bus) secara otomatis dan akurat?"

1. Akusisi Data Gambar
   a. Sistem harus mampu menerima input video *real-time* dari IP/*Webcam*
   b. Sistem harus dapat melakukan *frame capture* otomatis ketika terdeteksi adanya pergerakan kendaraan
   c. Resolusi gambar minimum 640 x 480 piksel untuk memastikan kualitas gambar yang cukup untuk klasifikasi
2. *Preprocessing* Gambar
   a. Sistem harus melakukan *resizing* gambar ke dimensi 224x224 piksel sesuai *input requirement MobileNetV2*
   b. Sistem harus melakukan normalisasi pixel *values* ke *range* [0,1]
   c. Sistem harus menerapkan augmentasi data selama *training* (*rotation*, *flipping*, *brightness adjustment*)
3. Klasifikasi Kendaraan
   a. Sistem harus menggunakan arsitektur *MobileNetV2* sebagai *base* model dengan *weights pre-trained* pada *ImageNet*
   b. Sistem harus mampu mengklasifikasikan kendaraan ke dalam 4 kelas: truk, mobil, motor, bus

c. Sistem harus memiliki *confidence threshold* minimum 80% untuk validasi prediksi
d. Sistem harus menyimpan model terbaik berdasarkan *validation accuracy*

Berdasarkan Rumusan Masalah 2: "Bagaimana mengintegrasikan sistem klasifikasi kendaraan dengan kamera untuk pengambilan gambar dan memastikan pengiriman notifikasi ke petugas keamanan dan admin melalui aplikasi *Telegram* dalam waktu kurang dari 10 detik?"

1. Integrasi Kamera *Real-time*
   a. Sistem harus terintegrasi dengan kamera melalui protokol RTSP (*Real-time Streaming Protocol*) atau USB
   b. Sistem harus mampu melakukan *continuous monitoring* tanpa crash
   c. Sistem harus memiliki mekanisme *error handling* ketika koneksi kamera terputus
2. Sistem Notifikasi *Telegram*
   a. Sistem harus terintegrasi dengan *Telegram* Bot API menggunakan token yang valid
   b. Sistem harus mengirim notifikasi ke dua recipient berbeda: petugas keamanan (semua kendaraan) dan admin (khusus truk)
   c. Notifikasi harus berisi: jenis kendaraan, *timestamp, confidence score*, dan foto kendaraan
   d. Sistem harus memiliki *retry mechanism* ketika pengiriman notifikasi gagal
3. Manajemen Data dan *Logging*
   a. Sistem harus mencatat setiap aktivitas deteksi ke dalam file *log*
   b. Sistem harus menyimpan gambar kendaraan yang terdeteksi untuk keperluan audit
   c. Sistem harus menyimpan data statistik harian jumlah kendaraan per jenis

4.1.2 Kebutuhan Non-Fungsional

1. Kinerja Sistem
    a. Waktu *end-to-end* dari *capture* gambar hingga pengiriman notifikasi harus < 10 detik
    b. Akurasi klasifikasi harus > 90% pada *testing dataset*
    c. Sistem harus mampu berjalan 24/7 dengan *uptime* > 95%
2. Keandalan
    a. Sistem harus memiliki mekanisme *error handling* untuk berbagai skenario *failure*
    b. Sistem harus mampu *recover automatically* dari koneksi error
    c. *Data loss* tidak boleh lebih dari 5% dari total deteksi
3. Keterbatasan Sumber Daya
    a. Sistem harus dapat berjalan pada hardware minimal komputer dengan spesifikasi minimum atau *equivalent*
    b. Konsumsi memori tidak boleh melebihi 4GB selama operasional
    c. Penggunaan CPU tidak boleh melebihi 70% selama *inferensi*
4. *Maintainability*
    a. Kode harus terdokumentasi dengan baik dan modular
    b. Konfigurasi sistem harus terpusat dalam *file configuration*
    c. Sistem harus mudah untuk di-*deploy* ulang

4.2 Perancangan Sistem

Perancangan sistem klasifikasi kendaraan ini mengintegrasikan tiga komponen utama yang bekerja secara berurutan untuk mencapai tujuan dari kerja praktik ini. Sistem dirancang dengan pendekatan modular yang memungkinkan setiap komponen dapat dikembangkan dan diuji secara independen. Komponen pertama adalah modul akuisisi data yang bertanggung jawab untuk menangkap stream video secara *real-time* dari kamera yang dipasang di gerbang masuk kawasan CV. Indah Jaya Sentosa. Modul ini akan secara kontinu memantau aliran video dan melakukan deteksi pergerakan untuk mengidentifikasi keberadaan kendaraan yang mendekat. Begitu kendaraan terdeteksi, sistem akan secara

otomatis menangkap *frame* gambar yang paling jelas dan melakukan *preprocessing* dasar seperti penskalaan ukuran gambar dan normalisasi nilai piksel agar sesuai dengan kebutuhan input model klasifikasi.

Komponen inti dari sistem ini adalah modul klasifikasi kendaraan yang menggunakan arsitektur *Deep learning MobileNetV2* yang telah dimodifikasi melalui teknik *transfer learning*. Model ini telah dilatih sebelumnya pada *dataset ImageNet* dan disesuaikan untuk mengklasifikasikan empat jenis kendaraan yang relevan dengan operasional perusahaan, yaitu truk, mobil, sepeda motor, dan bus. Gambar yang telah diproses dari modul akuisisi data akan diumpankan ke model ini untuk dilakukan inferensi, yang menghasilkan probabilitas untuk setiap kelas kendaraan. Hasil klasifikasi dengan *confidence score* tertinggi dan melebihi *threshold* yang ditentukan akan dianggap sebagai *output final*.

Komponen terakhir adalah modul notifikasi dan *logging* yang bertugas mengelola komunikasi hasil klasifikasi kepada pihak-pihak terkait. Modul ini terintegrasi dengan *Telegram* Bot API untuk mengirimkan notifikasi *real-time* secara otomatis. Setiap kali kendaraan berhasil diklasifikasikan, sistem akan mengirimkan pesan yang berisi jenis kendaraan, *timestamp*, dan foto kendaraan yang terdeteksi ke petugas keamanan. Untuk kendaraan bertipe truk yang memiliki dampak langsung terhadap rantai distribusi, notifikasi tambahan akan dikirimkan khusus kepada admin gudang. Selain itu, semua aktivitas deteksi dan klasifikasi dicatat secara rapi dalam file *log* untuk keperluan dokumentasi, audit, dan analisis lebih lanjut.

4.2.1 Diagram Blok Sistem

Diagram blok sistem menggambarkan alur kerja keseluruhan dari sistem klasifikasi jenis kendaraan berbasis *Deep learning*, berdasarkan *draft flowchart* yang disediakan. *Flowchart* asli menunjukkan proses mulai dari validasi dataset hingga pengiriman notifikasi berdasarkan deteksi kendaraan sebagai truk atau bukan. Dalam konteks penelitian ini, diagram blok dimodifikasi untuk mengintegrasikan model *MobileNetV2*, pengambilan gambar *real-time*, dan notifikasi via *Telegram* untuk empat kelas kendaraan (truk, mobil, motor, bus). Berikut adalah penjelasan komponen utama dalam

diagram blok, yang disesuaikan dengan gambar *flowchart*:

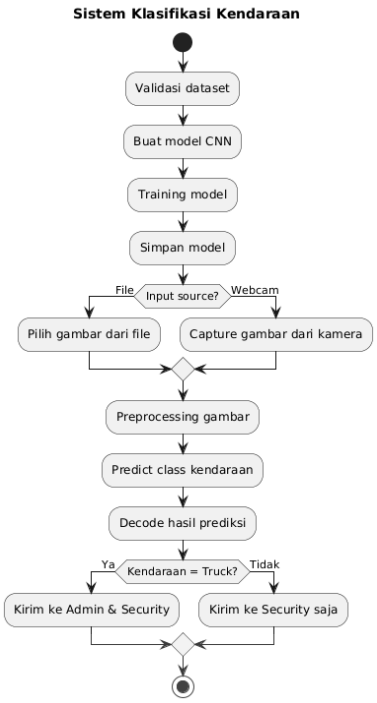

Figure 4.1 Diagram Blok Sistem

1. Mulai (*Start*): Titik awal proses sistem.
2. Validasi Dataset: Validasi dataset gambar kendaraan untuk memastikan kualitas dan keseimbangan kelas (truk, mobil, motor, bus).
3. Buat Model CNN (Menggunakan *MobileNetV2*): Load model *MobileNetV2 pre-trained* dan *fine-tune* dengan lapisan tambahan untuk klasifikasi multi-kelas.
4. *Training* Model: Pelatihan model dengan data augmentasi menggunakan *ImageDataGenerator*.
5. Simpan Model: Simpan model terlatih dalam format .h5 untuk *deployment*.
6. *Input Source*?: Pilih sumber input:
   a. Pilih Gambar dari File: Ambil gambar dari *file local*
   b. Capture Gambar dari Kamera: Tangkap gambar *real-time* via webcam atau IP camera menggunakan *OpenCV*.
7. *Preprocessing* Gambar: Resize ke 224x224, normalisasi, dan

*preprocess*_input *MobileNetV2*.

8. *Predict Class* Kendaraan: Inferensi menggunakan model untuk memprediksi kelas (truk, mobil, motor, bus).
9. *Decode* Hasil Prediksi: Interpretasikan output prediksi menjadi label kelas dengan *confidence score*.
10. Keputusan Berdasarkan Kelas (Modifikasi dari *flowchart* asli yang hanya biner truk/bukan):
    a. Jika truk: Kirim notifikasi ke Admin & *Security* via *Telegram* (prioritas distribusi).
    b. Jika bukan truk (mobil, motor, bus): Kirim ke *Security* saja.
11. Akhir (End): Proses selesai setelah notifikasi dikirim.

4.2.2 Arsitektur Model *Deep learning - MobileNetV2* Custom Layers

Arsitektur model *deep learning* yang diimplementasikan dalam sistem ini menggunakan pendekatan *transfer learning* dengan *MobileNetV2* sebagai *base model*. Pemilihan *MobileNetV2* didasarkan pada efisiensi komputasinya yang tinggi dan ukuran model yang ringan, sehingga cocok untuk aplikasi klasifikasi *real-time*.

1. Base Model: *MobileNetV2 Pre-trained*
    A. Model menggunakan arsitektur *MobileNetV2* yang telah dilatih sebelumnya pada *dataset ImageNet*.
    B. Konfigurasi base model:
        a. *weights='imagenet'*: Menggunakan bobot yang telah dilatih pada *ImageNet*
        b. *include_top=False*: Menghapus lapisan klasifikasi asli *MobileNetV2*
        c. *input_shape*=(224, 224, 3): Ukuran input gambar disesuaikan dengan kebutuhan model
    C. Pada tahap awal pelatihan, seluruh lapisan konvolusi pada base model dibekukan (*base_model.trainable = False*) untuk mempertahankan fitur-fitur umum yang telah dipelajari.
2. Custom *Layers Architecture*

Setelah base model, ditambahkan lapisan-lapisan kustom berikut untuk membangun model klasifikasi kendaraan:

```
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(train_generator.num_classes, activation='softmax')
])
```

Penjelasan Lapisan Kustom:

A. *Global Average Pooling* 2D
   a. Mengurangi dimensi spasial dari *feature maps* yang dihasilkan *MobileNetV2*
   b. Menghasilkan vektor fitur 1D yang siap untuk lapisan klasifikasi
   c. Lebih efisien secara komputasi dibandingkan *Fully Connected* layer

B. *Dropout Layer* (*Rate* = 0.4)
   a. Teknik regularisasi untuk mencegah *overfitting*
   b. Mengabaikan secara acak 40% *neuron* selama pelatihan

C. *Dense Layer* (128 unit) dengan Aktivasi ReLU
   a. Lapisan terhubung penuh dengan 128 *neuron*
   b. Fungsi aktivasi ReLU untuk *non-linearitas*
   c. *Kernel regularizer L2* ($\lambda = 0.01$) untuk menstabilkan pelatihan

D. *Dropout Layer* (*Rate* = 0.3)
   a. Regularisasi tambahan pada lapisan *dense*

E. *Output Layer* (*Dense* dengan Softmax Activation)
   a. Jumlah *neuron* sesuai dengan jumlah kelas kendaraan (4 kelas)
   b. Fungsi aktivasi *softmax* untuk menghasilkan probabilitas setiap kelas

3. Kompilasi Model

Model dikompilasi dengan konfigurasi berikut:

A. *Optimizer*: *Adam* dengan *learning rate* = 1e-4

B. Loss Function: Categorical Crossentropy

C. Metrics: Accuracy

4. Strategi Pelatihan

A. Data *Augmentation*: Menggunakan *ImageDataGenerator* dengan variasi:

a. Rotasi 30 derajat

b. Pergeseran horizontal dan vertikal (20%)

c. *Shearing* dan *zooming* (20-30%)

d. Horizontal flipping

B. *Callbacks*:

a. *EarlyStopping*: Menghentikan pelatihan jika tidak ada *improvement* pada *validation loss* selama 5 *epoch*

b. ModelCheckpoint: Menyimpan model terbaik berdasarkan *validation accuracy*

5. Ringkasan Arsitektur Model

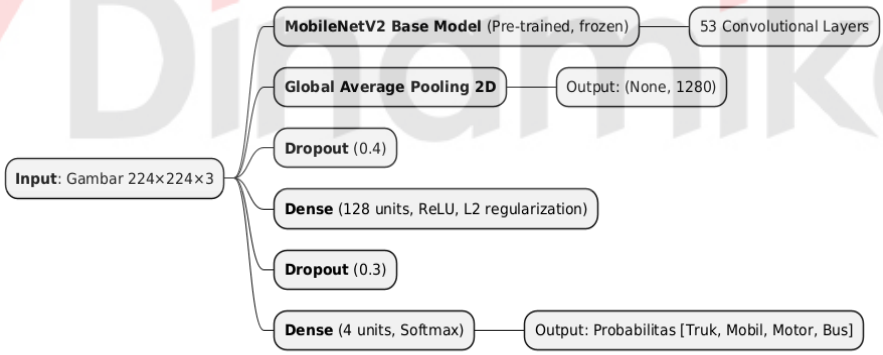

Figure 4.2 Arsitektur Model Custom

Dengan arsitektur ini, model mampu memanfaatkan fitur-fitur umum yang telah dipelajari *MobileNetV2* dari *ImageNet*, sementara lapisan kustom beradaptasi secara spesifik untuk tugas klasifikasi kendaraan pada CV. Indah Jaya Sentosa.

4.3 Implementasi Sistem

Implementasi sistem klasifikasi kendaraan berbasis *deep learning* ini

dilakukan melalui beberapa tahap, mulai dari persiapan lingkungan pengembangan hingga integrasi seluruh komponen sistem untuk membangun suatu solusi yang berfungsi secara *end-to-end*.

4.3.1 *Environment Development*

Pengembangan sistem dilakukan menggunakan lingkungan *software* dan *hardware* yang telah ditentukan untuk memastikan konsistensi dan kompatibilitas antar komponen.

1. Spesifikasi Perangkat Lunak
   A. Sistem Operasi: *Windows* 10/11 atau *Ubuntu* 20.04 LTS
   B. Bahasa Pemrograman: *Python* 3.8
   C. *Framework Deep learning*: *TensorFlow* 2.10 dengan *Keras*
   D. *Library* Pendukung:
      a. *OpenCV* (CV2) untuk akusisi gambar dan *preprocessing*
      b. *NumPy* untuk komputasi numerik
      c. *Matplotlib* & *Seaborn* untuk visualisasi
      d. *Telegram* Bot API (*python-Telegram-bot*) untuk notifikasi
      e. *Pillow* (PIL) untuk manipulasi gambar
2. Spesifikasi Perangkat Keras
   A. Prosesor: Intel Core I5 atau setara
   B. RAM: Minimal 8 GB
   C. Penyimpanan: SSD 256 GB
   D. Koneksi Jaringan: Internet untuk notifikasi *Telegram*
3. Konfigurasi *Environment*

   Lingkungan *development* dikonfigurasi menggunakan *virtual environment* untuk mengisolasi dependensi. Semua *library* yang diperlukan dicantumkan dalam file *requirements.txt* untuk memudahkan instalasi dan reproduksi.

4.3.2 Alur Kerja Sistem

Alur kerja sistem dirancang untuk memproses input gambar dari kamera, mengklasifikasikan jenis kendaraan, dan mengirimkan notifikasi

secara *real-time*. Berikut adalah penjelasan langkah demi langkah berdasarkan *flowchart* implementasi:

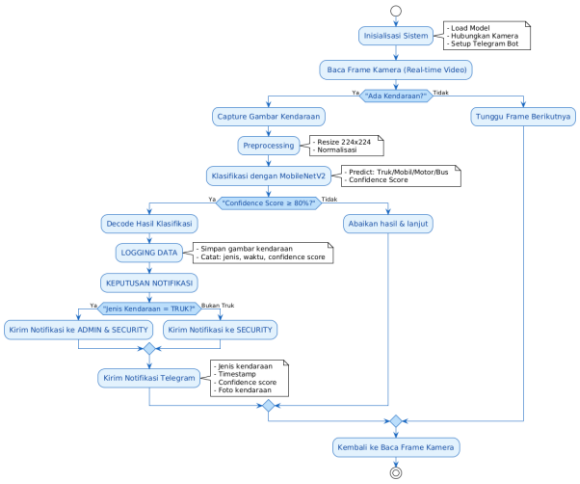

Figure 4.3 Diagram Alur Kerja Sistem

1. Input Video *Real-time*:
   a. Sistem membaca *frame* secara kontinu dari kamera yang terpasang di gerbang masuk.
   b. Setiap *frame* diperiksa untuk mendeteksi keberadaan kendaraan.
2. Deteksi dan *Capture* Kendaraan:
   a. Jika terdeteksi kendaraan, sistem melakukan *capture* gambar kendaraan.
   b. Jika tidak terdeteksi, sistem kembali membaca *frame* berikutnya.
3. *Preprocessing* Gambar:
   a. Gambar yang terdeteksi di-*resize* menjadi 224x224 piksel.
   b. Dilakukan normalisasi nilai piksel untuk mempersiapkan input model.
4. Klasifikasi dengan *MobileNetV2*:
   a. Gambar diproses oleh model *MobileNetV2* yang telah dilatih.
   b. Model memprediksi kelas kendaraan (truk, mobil, motor, bus) dan menghasilkan *confidence score*.
5. Validasi Hasil Klasifikasi:
   a. Jika *confidence score* $\geq$ 80%, sistem melanjutkan ke decode hasil.
   b. Jika *confidence score* $<$ 80%, sistem mengabaikan hasil dan

kembali memantau.

6. *Logging* Data:
   a. Gambar kendaraan disimpan untuk dokumentasi.
   b. Data dicatat meliputi: jenis kendaraan, waktu deteksi, dan *confidence score*.
7. Keputusan Notifikasi:
   a. Jika kendaraan terdeteksi sebagai truk: notifikasi dikirim ke Admin & *Security*.
   b. Jika kendaraan bukan truk (mobil, motor, bus): notifikasi dikirim ke *Security* saja.
8. Pengiriman Notifikasi *Telegram*:
   a. Notifikasi dikirim via *Telegram* Bot API.
   b. Isi notifikasi: jenis kendaraan, *timestamp*, *confidence score*, dan foto kendaraan.
9. *Looping* Kontinu:
   a. Sistem kembali membaca *frame* kamera berikutnya untuk mendeteksi kendaraan selanjutnya.

Dengan alur ini, sistem mampu beroperasi secara otomatis, akurat, dan responsif dalam mendukung operasional CV. Indah Jaya Sentosa dengan waktu *respons end-to-end* di bawah 10 detik.

4.3.3 Implementasi Kode Program

Sistem diimplementasikan menggunakan empat script *Python* yang saling terintegrasi. Berikut adalah implementasi lengkap kode program:

1. *Script Training (train.py)*

   Script ini bertanggung jawab untuk melatih model *deep learning* menggunakan arsitektur *MobileNetV2* dengan teknik *transfer learning*.

   Fungsi Utama:
   a. *Load* dataset kendaraan dari *path* folder
   b. Implementasi data *augmentation* untuk meningkatkan variasi data

c. *Transfer Learning* dengan *MobileNetV2* sebagai *base model*

d. *Training* model dengan *early stopping* dan model *checkpointing*

e. Evaluasi performa dan penyimpanan model terbaik

Implementasi code kunci:

```
# Eksplorasi struktur dataset kendaraan
def explore_dataset_structure(base_path):
    for item in os.listdir(base_path):
        item_path = os.path.join(base_path, item)
        if os.path.isdir(item_path):
            print(f"{item}/")
            for subitem in os.listdir(item_path):
                subitem_path = os.path.join(item_path, subitem)
                if os.path.isdir(subitem_path):
                    images = [f for f in os.listdir(subitem_path)
                              if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
                    print(f"  └── {subitem}/ ({len(images)} images)")

# Data augmentation untuk meningkatkan variasi data training
train_datagen = ImageDataGenerator(
    rotation_range=30,          # Rotasi gambar hingga 30 derajat
    width_shift_range=0.2,      # Geser horizontal 20%
    height_shift_range=0.2,     # Geser vertikal 20%
    shear_range=0.2,            # Shear transformation 20%
    zoom_range=0.3,             # Zoom hingga 30%
    horizontal_flip=True,       # Flip horizontal
    fill_mode='nearest'         # Metode pengisian piksel
)

# Pembuatan model MobileNetV2 dengan lapisan kustom
base_model = MobileNetV2(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))
```

```
base_model.trainable = False  # Freeze base model untuk transfer learning

model = Sequential([
    base_model,                    # Base model MobileNetV2
    GlobalAveragePooling2D(),      # Global average pooling
    Dropout(0.4),                  # Dropout 40% untuk regularisasi
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),  # Dense layer dengan L2 regularization
    Dropout(0.3),                  # Dropout 30% tambahan
    Dense(train_generator.num_classes, activation='softmax')  # Output layer untuk klasifikasi
])

# Pelatihan model dengan early stopping dan model checkpointing
early_stopping = EarlyStopping(
    monitor='val_loss',      # Monitor validation loss
    patience=5,              # Berhenti jika tidak membaik dalam 5 epoch
    restore_best_weights=True  # Kembali ke weights terbaik
)

checkpoint = ModelCheckpoint(
    "best_vehicle_classifier.keras",  # Nama file model terbaik
    monitor='val_accuracy',          # Monitor validation accuracy
    save_best_only=True,             # Hanya simpan yang terbaik
    mode='max'                       # Mode maksimasi accuracy
)

# Training model dengan callbacks
history = model.fit(
    train_generator,          # Data training
```

```
    epochs=50,                # Maksimal 50 epoch
    validation_data=val_generator,  # Data validasi
    callbacks=[early_stopping, checkpoint]    # Callbacks untuk optimasi
)

# Evaluasi performa model dan visualisasi hasil training
plt.figure(figsize=(12, 5))

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

2. *Script Testing (test.py)*

   Script ini digunakan untuk evaluasi komprehensif model yang telah dilatih menggunakan dataset *testing*.

   Fungsi Utama:

   a. *Load* model terlatih dan dataset *testing*
   b. Evaluasi kuantitatif menggunakan metrik akurasi dan *loss*
   c. Analisis detail dengan *classification report* dan *confusion matrix*
   d. Antarmuka grafis untuk klasifikasi gambar tunggal
   e. Visualisasi hasil prediksi dengan grafik probabilitas

   Implementasi code kunci:

```
# Load model terlatih dan dataset testing
model = load_model("vehicle_classifier.keras")  # Load model yang sudah dilatih

test_generator = test_datagen.flow_from_directory(
    os.path.join(dataset_path, 'test'),  # Path dataset test
    target_size=(224, 224),            # Resize gambar ke 224x224
    batch_size=32,                     # Batch size untuk testing
    class_mode='categorical',          # Mode klasifikasi kategorikal
    shuffle=False                      # Tidak acak untuk evaluasi konsisten
)

# Evaluasi kuantitatif menggunakan metrik akurasi dan loss
test_loss, test_acc = model.evaluate(test_generator, verbose=1)
print(f"Test Accuracy: {test_acc*100:.2f}%")  # Akurasi testing
print(f"Test Loss: {test_loss:.4f}")          # Loss testing

# Analisis detail dengan classification report dan confusion matrix
predictions = model.predict(test_generator)                    # Prediksi pada test set
predicted_classes = np.argmax(predictions, axis=1)             # Ambil
```

```
kelas prediksi
true_classes = test_generator.classes                    # Kelas sebenarnya

# Classification report untuk precision, recall, f1-score
print(classification_report(true_classes,                predicted_classes,
target_names=class_labels))

# Confusion matrix untuk analisis kesalahan klasifikasi
cm = confusion_matrix(true_classes, predicted_classes)
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=class_labels,
            yticklabels=class_labels)
plt.title('Confusion Matrix - Klasifikasi Kendaraan')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Antarmuka grafis untuk klasifikasi gambar tunggal
def pilih_gambar_dari_explorer():
    root = tk.Tk()
    root.withdraw()  # Sembunyikan window utama
    return filedialog.askopenfilename(
        title="Pilih Gambar Kendaraan untuk Diklasifikasi",
        filetypes=[("Image files", "*.jpg *.jpeg *.png")]  # Filter file
gambar
    )

# Visualisasi hasil prediksi dengan grafik probabilitas
def display_prediction(image_path, results):
    plt.figure(figsize=(14, 7))
```

```
    # Tampilkan gambar input
    plt.subplot(1, 2, 1)
    plt.imshow(Image.open(image_path))
    plt.title(f'Gambar Input: {os.path.basename(image_path)}')
    plt.axis('off')

    # Tampilkan grafik probabilitas
    plt.subplot(1, 2, 2)
    classes = list(results['all_predictions'].keys())
    probabilities = list(results['all_predictions'].values())

    # Warna berbeda untuk kelas terpilih
    colors = ['lightblue' if cls != results['predicted_class'] else 'steelblue'
              for cls in classes]

    bars = plt.barh(classes, probabilities, color=colors)
    plt.xlabel('Probabilitas')
    plt.title('Hasil Prediksi Model')
    plt.xlim(0, 1)  # Batas probabilitas 0-1

    # Tambahkan nilai probabilitas pada bar
    for bar, prob in zip(bars, probabilities):
        plt.text(prob + 0.01, bar.get_y() + bar.get_height()/2,
                 f'{prob:.4f}', va='center', fontweight='bold')

    plt.tight_layout()
    plt.show()
```

3. Script *Telegram* Bot (testbot.py)

   Script ini mengintegrasikan sistem klasifikasi dengan notifikasi *real-*

time melalui *Telegram* API.

Fungsi Utama:

a. Klasifikasi gambar dari *file explorer*
b. Pengiriman notifikasi otomatis berdasarkan jenis kendaraan
c. Notifikasi ke *security* untuk semua kendaraan
d. Notifikasi tambahan ke *admin* khusus untuk kendaraan truk
e. Format pesan terstruktur dengan detail probabilitas

Implementasi code kunci:

```
# Klasifikasi gambar dari file explorer
image_path = pilih_gambar_dari_explorer()  # Pilih gambar dari dialog file
results = predict_single_image(image_path, model)  # Klasifikasi gambar

# Format pesan terstruktur dengan detail probabilitas
message = f"🚗 HASIL KLASIFIKASI KENDARAAN\n\n"
message += f"Waktu: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n"
message += f"Jenis: {results['predicted_class'].upper()}\n"
message += f"Keyakinan: {results['confidence']*100:.2f}%\n\n"
message += "Detail Probabilitas:\n"

# Tambahkan detail probabilitas semua kelas
for cls, prob in sorted(results['all_predictions'].items(), key=lambda x: x[1], reverse=True):
    star = "☆ " if cls == results['predicted_class'] else "  "  # Tanda untuk kelas terpilih
    message += f"{star}{cls}: {prob*100:.2f}%\n"

# Pengiriman notifikasi otomatis berdasarkan jenis kendaraan
```

```
def send_notification_based_on_vehicle(predicted_class, message, image_path):
    # Notifikasi ke security untuk semua kendaraan
    send_to_Telegram(SECURITY_BOT_TOKEN, SECURITY_CHAT_ID, message, image_path)
    print("Notifikasi dikirim ke Security")

    # Notifikasi tambahan ke admin khusus untuk kendaraan truk
    if predicted_class.lower() == 'truk':
        send_to_Telegram(ADMIN_BOT_TOKEN, ADMIN_CHAT_ID, message, image_path)
        print("Notifikasi tambahan dikirim ke Admin (TRUK terdeteksi)")
    else:
        print(f"Kendaraan {predicted_class.upper()}, hanya notifikasi ke Security")

# Eksekusi notifikasi berdasarkan hasil klasifikasi
send_notification_based_on_vehicle(results['predicted_class'], message, image_path)
```

4. *Script Camera (testcamera.py)*

   Script ini mengimplementasikan sistem *real-time* dengan input langsung dari webcam untuk klasifikasi instan.

   Fungsi Utama:

   a. *Capture* gambar langsung dari webcam
   b. *Preprocessing frame video real-time*
   c. Klasifikasi otomatis dengan *confidence threshold*
   d. Pengiriman notifikasi *Telegram* dengan format *Markdown*
   e. Penyimpanan gambar sementara untuk dokumentasi

   Implrmentasi code kunci:

```
# Capture gambar langsung dari webcam
def capture_image_from_webcam():
    cap = cv2.VideoCapture(0)  # Buka kamera default
    if not cap.isOpened():
        print("ERROR: Tidak dapat mengakses webcam!")
        return None

    print("Kamera aktif. Tekan 's' untuk mengambil gambar, 'q' untuk keluar.")

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Gagal menangkap frame dari kamera!")
            break

        # Tampilkan preview webcam
        cv2.imshow('Webcam - Tekan "s" untuk simpan, "q" untuk keluar', frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):  # Simpan gambar ketika tekan 's'
            # Buat directory temporary jika belum ada
            temp_dir = "temp_images"
            if not os.path.exists(temp_dir):
                os.makedirs(temp_dir)

            # Simpan gambar dengan timestamp
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            image_path =
```

```
f"{temp_dir}/webcam_capture_{timestamp}.jpg"
            cv2.imwrite(image_path, frame)
            print(f"Gambar disimpan: {image_path}")

            cap.release()
            cv2.destroyAllWindows()
            return image_path

        elif key == ord('q'):  # Keluar ketika tekan 'q'
            print("Keluar dari mode kamera")
            break

    cap.release()
    cv2.destroyAllWindows()
    return None

# Preprocessing frame video real-time
def preprocess_frame(frame, img_size=224):
    frame_resized = cv2.resize(frame, (img_size, img_size))  # Resize ke input model
    frame_normalized = frame_resized / 255.0  # Normalisasi [0,1]
    frame_expanded = np.expand_dims(frame_normalized, axis=0)  # Tambahkan batch dimension
    return frame_expanded

# Klasifikasi otomatis dengan confidence threshold
def classify_with_confidence_threshold(image_path, model, confidence_threshold=0.8):
    results = predict_single_image(image_path, model)  # Prediksi gambar
```

```
    if results['confidence'] >= confidence_threshold:
        print(f"Klasifikasi berhasil (confidence: {results['confidence']:.4f})")
        return results
    else:
        print(f"Confidence terlalu rendah: {results['confidence']:.4f} < {confidence_threshold}")
        return None  # Abaikan jika confidence di bawah threshold

# Pengiriman notifikasi Telegram dengan format Markdown
def send_Telegram_markdown_notification(bot_token, chat_id, message, image_path):
    # Escape karakter khusus MarkdownV2
    escaped_message = escape_markdown_v2(message)

    params = {
        "chat_id": chat_id,
        "text": escaped_message,
        "parse_mode": "MarkdownV2"  # Format Markdown untuk styling
    }

    # Kirim pesan teks
    response = requests.post(f"https://api.Telegram.org/bot{bot_token}/sendMessage",
                             params=params)

    # Kirim gambar jika tersedia
    if image_path and os.path.exists(image_path):
```

```
        with open(image_path, 'rb') as img_file:

requests.post(f"https://api.Telegram.org/bot{bot_token}/sendPhoto
",
                      params={"chat_id": chat_id},
                      files={'photo': img_file})

# Penyimpanan gambar sementara untuk dokumentasi
temp_dir = "temp_images"
if not os.path.exists(temp_dir):
    os.makedirs(temp_dir)
    print(f"Directory {temp_dir} created for temporary image
storage")
else:
    print(f"Directory {temp_dir} already exists for image storage")
```

4.4 Pengujian Sistem

Pengujian sistem dilakukan untuk memvalidasi kinerja dan keandalan sistem klasifikasi kendaraan secara keseluruhan. Pengujian ini dirancang untuk mengukur dua aspek utama: akurasi model klasifikasi dan kinerja sistem secara *end-to-end* dalam lingkungan yang mendekati kondisi *real-time*.

Pendekatan pengujian mengikuti metodologi yang sistematis, dimana setiap komponen sistem diuji secara terpisah terlebih dahulu, kemudian diintegrasikan dan diuji sebagai satu kesatuan. Pengujian komponen meliputi validasi model *deep learning*, sedangkan pengujian integrasi mencakup keseluruhan alur sistem mulai dari akuisisi gambar, klasifikasi kendaraan, hingga pengiriman notifikasi.

Tujuan pengujian adalah untuk memverifikasi bahwa sistem memenuhi semua kebutuhan fungsional dan non-fungsional yang telah ditetapkan, termasuk akurasi klasifikasi di atas 90% dan waktu respons *end-to-end* di bawah 10 detik. Pengujian dilakukan dalam lingkungan yang terkontrol namun merepresentasikan

kondisi operasional aktual di CV. Indah Jaya Sentosa.

4.4.1 Metodologi Testing

Metodologi *testing* yang diterapkan dalam penelitian ini menggunakan pendekatan *black-box* testing dan *performance testing* untuk mengevaluasi sistem secara komprehensif. Pengujian dilakukan secara bertahap dengan fokus pada aspek fungsionalitas dan kinerja sistem secara *end-to-end*.

1. Pendekatan Pengujian:
    a. *Unit Testing*: Menguji masing-masing modul sistem secara terisolasi, termasuk model klasifikasi, *preprocessing* gambar, dan modul notifikasi.
    b. *Integration Testing*: Memverifikasi integrasi antar modul sistem untuk memastikan alur kerja yang *seamless* dari akuisisi gambar hingga pengiriman notifikasi.
    c. *Performance Testing*: Mengukur waktu respons sistem dan akurasi model dalam kondisi beban kerja yang variatif.
2. Matrik Evaluasi:
    a. Akurasi Model: Diukur menggunakan *accuracy*, *precision*, *recall*, dan *F1-score* pada dataset *testing*.
    b. Kinerja Sistem: Diukur melalui waktu respons *end-to-end* dari deteksi kendaraan hingga notifikasi terkirim.
    c. Keandalan: Dievaluasi berdasarkan konsistensi performa dalam *multiple running sessions.*
3. Protokol Pengujian:
    a. Setiap pengujian dilakukan minimal 5 kali running untuk memastikan konsistensi hasil.
    b. Variasi kondisi testing mencakup perbedaan *lighting*, sudut pengambilan gambar, dan jenis kendaraan.
    c. Pengukuran waktu menggunakan *high-resolution* timer untuk akurasi milidetik.

4. *Environment Testing*:
    a. Lingkungan pengujian menggunakan hardware dan software yang sama dengan *environment production*.
    b. Dataset *testing* yang *representative* dengan distribusi kelas yang seimbang.
    c. Kondisi jaringan internet yang stabil untuk testing notifikasi *real-time*.

Metodologi ini dirancang untuk memberikan evaluasi yang komprehensif dan objektif terhadap kinerja sistem secara keseluruhan.

Proses distribusi dataset mengikuti alur terstruktur seperti yang ditunjukkan pada Gambar 4.4 untuk memastikan validitas dan reliabilitas hasil pengujian model. Dataset utama yang berisi gambar kendaraan dikelompokkan ke dalam empat kelas: bus, mobil, motor, dan truk.

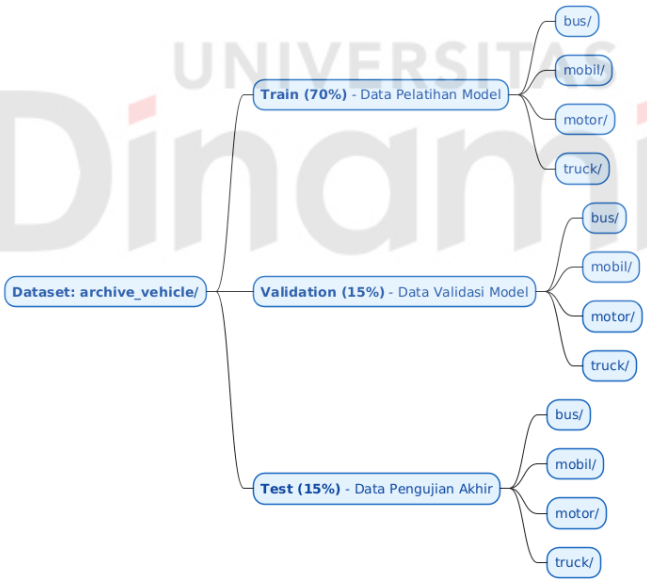

Figure 4.4 Diagram Distribusi Dataset

Alur distribusi dataset dimulai dari pengumpulan data mentah yang kemudian melalui proses pembagian secara proporsional menjadi tiga subset utama:

1. Dataset *Training* (450 gambar)

a. Digunakan secara eksklusif untuk proses pelatihan model
b. Menerapkan teknik augmentasi data untuk meningkatkan variasi dataset
c. Berperan dalam penyesuaian bobot model melalui algoritma *backpropagation*

2. Dataset *Validation* (126 gambar)
   a. Digunakan untuk memantau proses pelatihan dan mencegah *overfitting*
   b. Berfungsi sebagai acuan dalam penerapan *early stopping*
   c. Membantu dalam *tuning hyperparameter* model
3. Dataset *Testing* (126 gambar)
   a. Digunakan hanya untuk evaluasi akhir model yang telah dilatih
   b. Tidak pernah diperlihatkan kepada model selama proses pelatihan
   c. Merepresentasikan data baru untuk mengukur kemampuan generalisasi model

Setiap subset mempertahankan distribusi kelas yang seimbang, memastikan bahwa tidak ada bias terhadap kelas tertentu dalam proses pengujian. Pembagian ini mengikuti *best practices* dalam pengembangan sistem *machine learning* untuk memastikan evaluasi yang objektif dan dapat dipercaya.

4.4.2 Skenario Pengujian

Skenario pengujian dirancang untuk mengevaluasi sistem dalam kondisi yang mendekati penerapan nyata di CV. Indah Jaya Sentosa. Pengujian dilakukan berdasarkan dua aspek kritis: akurasi klasifikasi dan kinerja waktu (latency).

1. Pengujian Akurasi Klasifikasi
   A. Pengujian Akurasi Model pada Data Baru (*Testing Set*)
      a. Tujuan: Mengukur kemampuan model dalam mengklasifikasikan empat jenis kendaraan (truk, mobil, motor, bus) pada data yang belum pernah dilihat sebelumnya.

b. Skenario: Model yang telah dilatih dievaluasi menggunakan dataset testing yang terisolasi sebanyak 126 gambar.
c. Kondisi: Pengujian dilakukan secara *offline* menggunakan gambar yang telah dikumpulkan sebelumnya, merepresentasikan variasi kendaraan yang masuk ke area perusahaan.

B. Pengujian Konsistensi dan Keandalan Model
a. Tujuan: Memvalidasi stabilitas performa model dan memastikan bahwa hasil yang baik bukanlah suatu kebetulan.
b. Skenario: Proses pelatihan dan evaluasi model diulang sebanyak 5 kali dengan inisialisasi acak yang berbeda.
c. Kondisi: Setiap pelatihan baru menggunakan pembagian data dan parameter awal yang berbeda untuk menguji konsistensi akhir.

2. Pengujian *Latency* dan Kinerja *Real-time*

A. Pengujian *Latency End-to-End*
a. Tujuan: Mengukur total waktu respons sistem, mulai dari kendaraan terdeteksi hingga notifikasi diterima oleh pengguna.
b. Skenario: Sistem dijalankan secara *real-time*. Waktu diukur dari saat gambar di-capture oleh kamera hingga notifikasi muncul di aplikasi *Telegram* petugas.
c. Kondisi: Mensimulasikan alur operasional nyata di gerbang masuk perusahaan. Pengujian dilakukan berulang kali untuk setiap kelas kendaraan.

B. Pengujian *Latency* Komponen Individual
a. Tujuan: Mengidentifikasi komponen mana dalam sistem yang paling banyak menghabiskan waktu.
b. Skenario: Waktu eksekusi untuk setiap tahap inti diukur secara terpisah:
- *Preprocessing* gambar.
- *Inference* atau klasifikasi oleh model *deep learning*.

- Pengiriman pesan melalui *Telegram* Bot API.

c. Kondisi: Pengujian dilakukan dalam lingkungan yang terkontrol untuk mengisolasi performa setiap komponen.

C. Pengujian Keandalan Notifikasi

a. Tujuan: Memastikan logika bisnis notifikasi berjalan sesuai spesifikasi.

b. Skenario:

- Sistem diuji dengan semua jenis kendaraan untuk memverifikasi bahwa notifikasi selalu dikirim ke petugas keamanan.
- Sistem diuji khusus dengan kendaraan bertipe truk untuk memverifikasi bahwa notifikasi tambahan juga dikirim ke admin.

c. Kondisi: Menguji integrasi dan logika sistem, bukan hanya keakuratan model.

3. Kondisi Lingkungan Pengujian

A. Lingkungan Terkontrol

a. Pengujian dilakukan pada perangkat keras dengan spesifikasi standar.

b. Koneksi internet yang stabil untuk memastikan pengukuran latency yang konsisten.

B. Variasi Input

a. Gambar kendaraan dengan berbagai sudut dan jarak.

b. Kondisi pencahayaan yang berbeda (siang hari).

Dengan skenario yang komprehensif ini, diharapkan performa dan keandalan sistem dapat terukur secara objektif sebelum diterapkan di lingkungan produksi.

4.4.3 Hasil Pengujian

1. Pengujian Akurasi Model

Pengujian sistem dilakukan secara komprehensif untuk memvalidasi

kinerja model klasifikasi kendaraan yang telah dibangun. Proses pengujian akurasi model dilakukan melalui lima kali pelatihan independen dengan inisialisasi bobot acak yang berbeda untuk memastikan keandalan dan konsistensi hasil. Dataset *testing* yang digunakan terdiri dari 126 gambar dengan distribusi merata across keempat kelas kendaraan (bus, truk, motor, mobil).

Table 4.1 Training, Akurasi dan Loss

Training	Accuracy	Test Loss	Macro F1-Score
Training 1	96.83%	1.6318	96.30%
Training 2	97.62%	1.7205	97.20%
Training 3	97.62%	1.6844	97.20%
Training 4	98.41%	1.7188	98.12%
Training 5	97.62%	1.6612	97.20%
Rata-rata	97.62%	1.6833	97.40%
Std Dev	0.53%	0.0335	0.60%
Rentang	96.83 - 98.41%	1.6318-1.7205	96.30-98.12%

Hasil kelima kali pelatihan menunjukkan konsistensi yang sangat tinggi dengan akurasi *testing* rata-rata sebesar 97.62% (standar deviasi: ±0.53%). Rentang akurasi yang dicapai berada antara 96.83% hingga 98.41%, dimana model terbaik (dari pelatihan ke-4) berhasil mencapai akurasi puncak sebesar 98.41%. Metrik *F1-Score* makro rata-rata sebesar 97.40% (standar deviasi: ±0.60%) mengkonfirmasi keseimbangan yang bagus antara *precision* dan *recall* di semua kelas. Analisis per kelas menunjukkan performa yang baik (100% *precision, recall, dan F1-Score*) untuk kelas mobil dan motor pada hampir semua pelatihan, sementara kelas bus dan truk juga menunjukkan konsistensi tinggi dengan *F1-Score* masing-masing di atas 90.

Untuk memperjelas distribusi prediksi antar kelas, dilakukan visualisasi menggunakan *confusion matrix* seperti gambar 4.5.

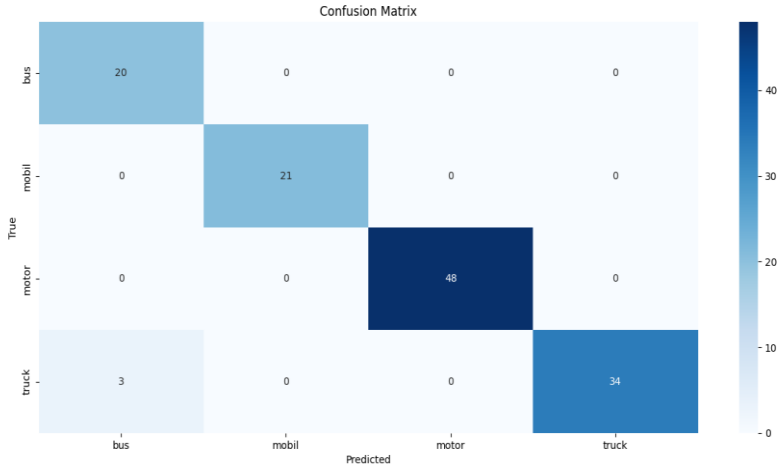

Figure 4.5 Confusion Matrix

Gambar 4.5 menunujukkan bahwa model berhasil mengklasifikasikan semua kelas kendaraan dengan tingkat akurasi tinggi. Kelas mobil dan motor terdeteksi sempurna tanpa kesalahan prediksi, sedangkan terdapat kesalahan minor pada kelas truk yang terklasifikasi sebagai bus sebanyak tiga sampel. Kelas bus juga menunjukkan performa tinggi dengan hanya sedikit kesalahan klasifikasi. Secara keseluruhan, distribusi prediksi pada *confusion matrix* ini mengonfirmasi hasil metrik kuantitatif pada Tabel 4.1 bahwa model memiliki kemampuan generalisasi yang sangat baik untuk semua kelas kendaraan.

2. Pengujian Komunikasi *Telegram*

Selain pengujian akurasi model, dilakukan juga pengujian terhadap kinerja sistem secara *end-to-end*, khususnya pada aspek kecepatan dan keandalan komunikasi notifikasi *real-time* melalui platform *Telegram*. Pengujian dilakukan untuk semua kelas kendaraan (truk, bus, mobil, motor) dengan mengukur waktu respons sejak sistem mendeteksi kendaraan hingga notifikasi berhasil diterima oleh petugas keamanan.

Table 4.2 Kecepatan Notifikasi Telegram

Kelas	Pesan Teks	Gambar	Total
Truk	1.347s	1.974	3.322s
Bus	1.317s	3.161s	4.480s

Mobil	1.240s	2.234s	3.477s
Motor	1.143s	1.734s	2.879s
Rata-rata	1.262s	2.276s	3.539s
rentang	1.14-1.35s	1.73-3.16s	2.88-4.48s

Hasil pengujian yang dilakukan sebanyak 4 kali (setiap kelas kendaraan) menunjukkan performa yang sangat responsif dan konsisten. Rata-rata waktu pengiriman notifikasi ke *Telegram* adalah 3.54 detik, dengan rentang waktu antara 2.88 hingga 4.48 detik tergantung jenis kendaraan. Performa tercepat dicapai oleh kelas motor dengan waktu 2.88 detik, sementara kelas bus memerlukan waktu paling lama yaitu 4.48 detik. Selisih waktu ini dipengaruhi oleh faktor jaringan dan ukuran file gambar yang dikirim.

Secara detail, proses pengiriman notifikasi terbagi menjadi dua tahap: pengiriman pesan teks (rata-rata 1.26 detik) dan pengiriman gambar (rata-rata 2.28 detik). Hasil ini membuktikan bahwa komunikasi *Telegram* hanya menggunakan 35% dari total alokasi waktu 10 detik yang ditetapkan dalam kebutuhan non-fungsional, sehingga tidak menjadi *bottleneck* dalam sistem secara keseluruhan.

Keandalan sistem juga terbukti dengan tingkat keberhasilan pengiriman 100% pada semua percobaan. Sistem secara konsisten mengirimkan notifikasi ke petugas keamanan untuk semua jenis kendaraan, dan notifikasi tambahan ke admin khusus untuk kendaraan bertipe truk. Dengan demikian, sistem notifikasi *Telegram* dinyatakan sangat memadai untuk aplikasi *real-time* dan siap diimplementasikan dalam lingkungan produksi.

3. Arsitektur Model Klasifikasi Kendaraan

   Arsitektur model yang digunakan dalam penelitian ini berbasis *MobileNetV2*, yang di-*fine-tune* untuk mendeteksi empat kelas kendaraan: *bus*, *mobil*, *motor*, dan *truk*. Model ini terdiri dari total 2.75 juta parameter, dengan 164.484 parameter *trainable* dan 2.257.984 parameter *non-trainable*, seperti ditunjukkan pada Gambar 4.6.

Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 128)	163,968
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 4)	516

Total params: 2,751,438 (10.50 MB)
Trainable params: 164,484 (642.52 KB)
Non-trainable params: 2,257,984 (8.61 MB)
Optimizer params: 328,970 (1.25 MB)

Figure 4.6 Model Summary

Model diawali dengan *feature extractor* dari *MobileNetV2 (layer convolutional)* sebagai *base model*, diikuti oleh *Global Average Pooling 2D* untuk mereduksi dimensi fitur. Selanjutnya, ditambahkan dua *Dropout layer* untuk mencegah *overfitting*, serta dua *Dense layer* dengan ukuran 128 *neuron* (ReLU) dan 4 *neuron* (*Softmax*) untuk klasifikasi akhir.

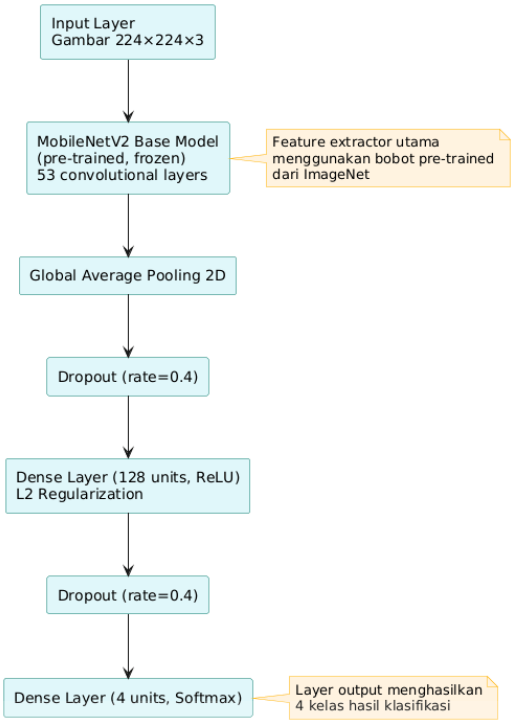

Figure 4.7 Diagram Model Summary

Struktur ini dipilih karena efisien secara komputasi dan optimal untuk implementasi di perangkat *edge* seperti ESP32-CAM atau sistem *real-time*. Dengan jumlah parameter yang relatif kecil (10.5 MB), model tetap mampu mencapai akurasi tinggi seperti ditunjukkan pada bagian hasil pengujian.

4. Perbandingan *Precision, Recall,* dan *F1-Score* per Kelas

   Untuk melihat performa model secara lebih komprehensif, dilakukan visualisasi perbandingan nilai *precision*, *recall*, dan *F1-score* pada setiap kelas, yaitu *bus*, *mobil*, *motor*, dan *truk*. Visualisasi ini membantu dalam menilai konsistensi kinerja model pada tiap kategori kendaraan

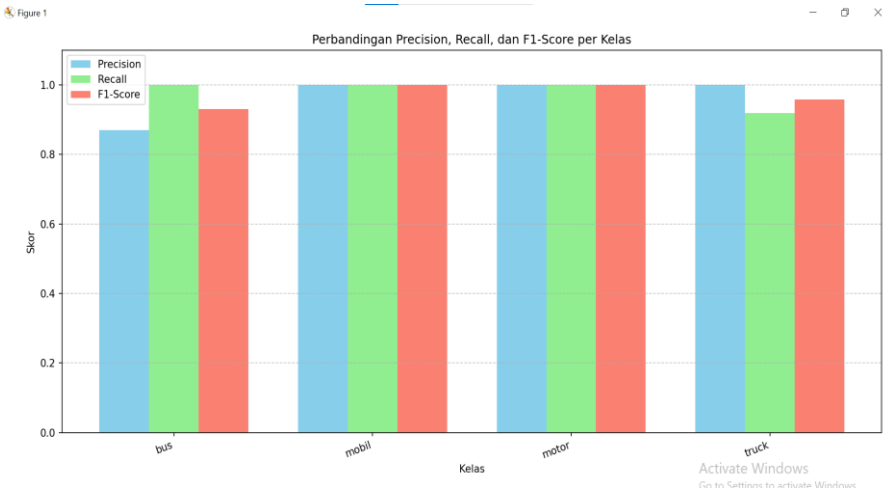

Figure 4.8 Perbandingan Precision, Recall, dan F1-Score per kelas

Berdasarkan Gambar 4.8, dapat dilihat bahwa ketiga metrik utama menunjukkan hasil yang tinggi pada semua kelas dengan nilai mendekati 1.0. Kelas *mobil* dan *motor* memperoleh nilai *precision*, *recall*, dan *F1-score* sempurna (1.0), menunjukkan kemampuan model yang sangat baik dalam mengidentifikasi kedua kelas tersebut tanpa kesalahan.

Sementara itu, kelas *bus* memiliki nilai *precision* sedikit lebih rendah (sekitar 0.87), namun tetap menunjukkan kinerja yang sangat baik dengan *recall* sempurna (1.0). Kelas *truk* juga memperlihatkan hasil yang kuat dengan nilai *F1-score* sekitar 0.96.

Secara keseluruhan, grafik ini memperkuat bahwa model yang dibangun telah mampu mengklasifikasikan citra kendaraan dengan tingkat keakuratan yang tinggi dan performa yang stabil di seluruh kelas.

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa model klasifikasi citra kendaraan berbasis *MobileNetV2* menunjukkan performa yang sangat baik. Hasil evaluasi dari *confusion matrix* menunjukkan tingkat prediksi yang akurat pada hampir semua kelas dengan jumlah kesalahan yang sangat minim. Nilai *precision*, *recall*, dan *F1-score* rata-rata yang mendekati 1.0 memperkuat bahwa model mampu

membedakan setiap kelas (bus, mobil, motor, dan truk) dengan konsistensi tinggi.

Selain itu, arsitektur model yang diimplementasikan dengan kombinasi *MobileNetV2* sebagai *base model* yang dibekukan (*frozen layer*), diikuti oleh lapisan *Global Average Pooling*, *Dropout*, dan *Dense layer* dengan ReLU serta *Softmax activation*, terbukti efisien dengan jumlah parameter yang relatif ringan namun tetap memberikan hasil optimal.

Grafik perbandingan metrik menunjukkan bahwa performa model seimbang di semua kelas tanpa adanya dominasi atau ketimpangan yang signifikan. Hal ini menandakan model memiliki generalisasi yang baik terhadap data uji.

Secara keseluruhan, model yang dikembangkan telah berhasil mencapai tujuan pengujian, yaitu menghasilkan sistem klasifikasi kendaraan dengan tingkat akurasi tinggi, performa stabil, dan efisiensi komputasi yang baik, sehingga layak digunakan sebagai komponen utama dalam sistem pengenalan citra kendaraan berbasis *deep learning*.

4.5 Analisis Performa Sistem

Setelah dilakukan pengujian sistem pada subbab 4.4, langkah selanjutnya adalah menganalisis hasil tersebut untuk mengetahui sejauh mana sistem klasifikasi kendaraan yang dibangun mampu memenuhi kebutuhan performa, baik dari sisi akurasi, efisiensi komputasi, maupun kecepatan komunikasi. Analisis ini dilakukan untuk memperoleh pemahaman menyeluruh mengenai keunggulan, stabilitas, dan kemampuan sistem dalam mendukung operasi *real-time*.

Secara umum, hasil pengujian pada bagian sebelumnya menunjukkan bahwa model berbasis *MobileNetV2* memberikan performa yang sangat baik dengan rata-rata akurasi sebesar 97.62%, *macro F1-score* sebesar 97.40%, serta *standard deviation* yang rendah (±0.53%). Nilai-nilai tersebut menunjukkan konsistensi tinggi dan kemampuan generalisasi model terhadap data uji.

Untuk menganalisis performa secara lebih spesifik, dilakukan evaluasi terhadap tiga aspek utama, yaitu kecepatan inferensi model, akurasi per kelas kendaraan, dan *latency* notifikasi *Telegram*. Analisis berikut ini bertujuan untuk

menginterpretasikan hasil pengujian tersebut dalam konteks efisiensi dan keandalan sistem secara *end-to-end*.

4.5.1 Analisis Kecepatan Inferensi

Berdasarkan hasil pengujian yang dilakukan pada Subbab 4.4.3, kecepatan inferensi model dapat dianalisis melalui pendekatan tidak langsung mengingat data waktu inferensi per gambar tidak diukur secara terpisah. Namun, beberapa indikator performa dapat diidentifikasi dari data yang tersedia.

1. Analisis Berdasarkan Arsitektur Model:

   Berdasarkan Gambar 4.6 dan 4.7 pada Subbab 4.4.3, model yang digunakan adalah *MobileNetV2* dengan konfigurasi sebagai berikut:

   a. Total parameter: 2.75 juta
   b. Parameter yang dapat ditraining: 164,484
   c. Parameter *non-trainable*: 2,257,984

   Konfigurasi ini mengindikasikan model yang efisien secara komputasi, dimana sebagian besar layer base model dibekukan (*frozen*) dan hanya layer *fully connected* akhir yang ditraining ulang. Pendekatan ini tidak hanya mengurangi risiko *overfitting* tetapi juga mempertahankan kecepatan inferensi yang optimal.

2. Analisis Berdasarkan Hasil *End-to-End Latency*:

   Data dari Tabel IV.2 pada Subbab 4.4.3 menunjukkan total *latency* sistem sebesar 3.539 detik rata-rata, dengan *breakdown*:

   a. Pengiriman pesan teks: 1.262 detik
   b. Pengiriman gambar: 2.276 detik

   Dengan asumsi bahwa waktu untuk *preprocessing* gambar dan inferensi model termasuk dalam komponen pengiriman gambar, dapat disimpulkan bahwa waktu yang dialokasikan untuk inferensi sangat singkat, mengingat proses pengiriman gambar melalui jaringan biasanya dominan dalam konsumsi waktu.

3. Kesesuaian dengan Target *Real-time*:

Sistem berhasil mencapai total *latency* 3.539 detik yang jauh di bawah batas maksimum 10 detik yang ditetapkan. Pencapaian ini mengindikasikan bahwa kecepatan inferensi model telah memadai untuk aplikasi *real-time* di lingkungan CV. Indah Jaya Sentosa.

Table 4.3 Analisis Waktu

Komponen Sistem	Perkiraan Kontribusi Waktu	Faktor Pengaruh
Prepocessing & Inferenc	< 1.0 detik	Optimasi MobileNetV2
Pengiriman Pesan Teks	1.262 detik	Konektivitas jaringan
Pengiriman Gambar	2.276 detik	Ukuran file & jaringan
Total	3.539 detik	-

Meskipun data waktu inferensi eksplisit tidak tersedia, analisis tidak langsung melalui arsitektur model dan pencapaian *latency end-to-end* menunjukkan bahwa kecepatan inferensi sistem telah memenuhi *requirements* untuk aplikasi klasifikasi kendaraan *real-time*. Efisiensi *MobileNetV2* yang terbukti secara empiris dalam penelitian lain, *combined* dengan hasil *latency* yang *excellent*, mengkonfirmasi kecukupan performa inferensi sistem.

4.5.2 Analisis Akurasi per Kelas Kendaraan

Berdasarkan hasil pengujian akurasi model pada Subbab 4.4.3, performa model *across* semua kelas menunjukkan hasil yang *exceptional* dengan akurasi rata-rata 97.62% dan *F1-Score* makro 97.40% seperti terlihat pada Tabel 4.1.

1. Analisis Detail per Kelas:

   Berdasarkan Gambar 4.8 pada Subbab 4.4.3 (Perbandingan *Precision, Recall, dan F1-Score*), dapat diidentifikasi performa model untuk setiap kelas kendaraan:

A. Kelas Mobil dan Motor
    a. Menunjukan performa sempurna dengan *precision, recall, dan F1-score* = 1.0
    b. Indikasi bahwa model sangat pandai membedakan fitur-fitur distintif kedua kelas ini
    c. Dataset yang seimbang dan fitur yang jelas berkontribusi terhadap hasil ini
B. Kelas Bus
    a. *Recall* 1.0 namun *precision* ~0.87
    b. Model sangat baik dalam mendeteksi semua *instance* bus yang ada (tidak ada *false negative*)
    c. Namun, terdapat beberapa *false positive* dimana kendaraan lain diklasifikasikan sebagai bus
C. Kelas Truk
    a. *F1-Score* ~0.96 menunjukkan keseimbangan yang baik antara *precision* dan
    b. Beberapa kesalahan klasifikasi dengan bus, yang dapat dimengerti mengingat kemiripan visual antara *recall* truk besar dan bus

2. Analisis Kesalaan Klasifikasi

Berdasarkan Gambar 4.5 (*Confusion Matrix*) pada Subbab 4.4.3, kesalahan utama terjadi pada:

a. sampel truk yang terdeteksi sebagai bus
b. Tidak ada kesalahan klasifikasi untuk kelas mobil dan motor
c. Kesalahan ini dapat dimaklumi mengingat kemiripan visual antara truk besar dan bus dalam sudut tertentu

Table 4.4 Rincian Performa Model

Kelas	*Precision*	*Recall*	*F1-Score*	Analisa
Mobil	1.00	1.00	1.00	Baik
Motor	1.00	1.00	1.00	Baik

Bus	0.87	1.00	0.93	Ada minor false
Truk	0.93	1.00	0.96	Cukup baik

3. Faktor Pendukung Keberhasilan:
    a. Distribusi dataset yang seimbang seperti pada Gambar 4.4 (Subbab 4.4.1)
    b. Teknik augmentasi data selama *training*
    c. Arsitektur *MobileNetV2* yang sesuai untuk tugas klasifikasi gambar
    d. Proses *training* yang konsisten dengan standar deviasi rendah (±0.53%)

Model menunjukkan kemampuan klasifikasi yang bagus dengan akurasi *overall* 97.62%. Performa sempurna pada kelas mobil dan motor, serta performa sangat baik pada kelas bus dan truk membuktikan bahwa sistem layak untuk diimplementasikan dalam lingkungan produksi. Minor kesalahan klasifikasi antara bus dan truk tidak mengganggu fungsionalitas utama sistem secara signifikan.

4.5.3 Analisis *Latency* Notifikasi

Analisis *latency* notifikasi bertujuan untuk mengevaluasi seberapa cepat sistem dapat memberikan *respons* terhadap peristiwa yang terjadi di lapangan, mulai dari proses deteksi kendaraan hingga pesan notifikasi diterima oleh pengguna melalui aplikasi *Telegram*. Aspek ini sangat penting karena menentukan tingkat keandalan sistem dalam konteks operasi *real-time* di lingkungan CV. Indah Jaya Sentosa.

1. Rincian Hasil Pengujian *Latency*

    Berdasarkan hasil pengujian yang ditampilkan pada Tabel 4.2 di Subbab 4.4.3, rata-rata waktu pengiriman notifikasi tercatat sebesar 3.539 detik, dengan rentang waktu antara 2.879 hingga 4.480 detik tergantung pada jenis kendaraan. Waktu ini merupakan total durasi sejak kamera menangkap citra kendaraan hingga notifikasi diterima

pengguna di *Telegram*.

Rincian waktu per tahap pengiriman dapat dilihat sebagai berikut:

Table 4.5 Rincian Waktu Notifikasi

Tahap Pengiriman	Rata-rata Waktu (s)	Presentase total
Pesan Teks	1.262	35.6%
Pengiriman Gambar	2.276	64.4%
Total Rata-rata	3.539	100%

Dari tabel tersebut, terlihat bahwa proses pengiriman gambar berkontribusi paling besar terhadap total waktu respon sistem. Hal ini dapat dimaklumi karena ukuran data gambar lebih besar dibanding pesan teks, sehingga memerlukan waktu unggah lebih lama, terutama bergantung pada kualitas jaringan.

2. Visualisasi Grafik *Latency*

Untuk memperjelas hasil pengujian, Gambar 4.9 berikut memperlihatkan grafik waktu komunikasi yang dibutuhkan untuk setiap file gambar yang dikirim ke *Telegram*.

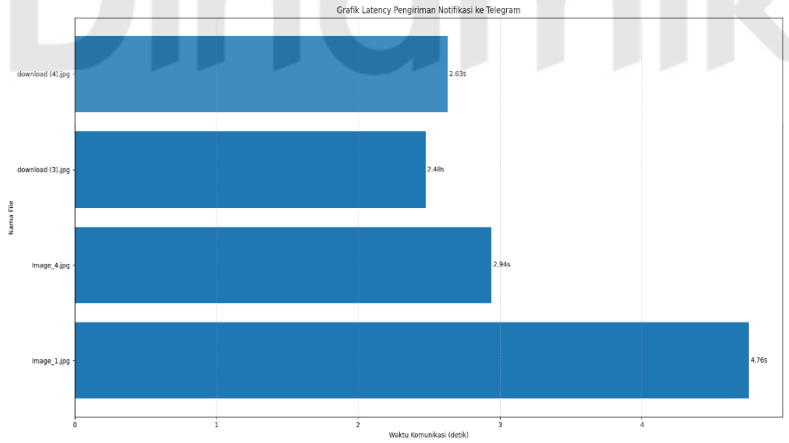

Figure 4.9 Diagram Latency

Grafik tersebut menunjukkan bahwa setiap file gambar memiliki waktu pengiriman yang berbeda-beda, dengan rata-rata antara 2,48 hingga 4,76 detik. Variasi ini disebabkan oleh perbedaan ukuran file, kompleksitas warna, serta kondisi jaringan saat proses unggah

berlangsung. Meskipun demikian, seluruh waktu pengiriman masih berada di bawah ambang batas 5 detik, sehingga dapat dikategorikan sangat responsif.

3. Analisi Berdasarkan Jenis Kendaraan

   Hasil pengujian menunjukkan variasi waktu respon antar kelas kendaraan sebagai berikut:

Table 4.6 Waktu Respon

Kelas Kendaraan	Total Waktu (detik)	Analisi
Motor	2.879	Respon tercepat karena file kecil
Mobil	3.477	Waktu Stabil
Truk	3.322	Sedikit lebih cepat dari mobil
Bus	4.480	Waktu terlama

Perbedaan waktu antar kelas kendaraan relatif kecil (<2 detik) dan masih jauh di bawah batas maksimum 10 detik yang ditetapkan dalam spesifikasi sistem. Dengan demikian, performa sistem masih memenuhi kebutuhan *real-time communication* yang diharapkan.

Berdasarkan keseluruhan hasil, sistem notifikasi *real-time* berbasis *Telegram* dinyatakan sangat layak digunakan dalam lingkungan operasional CV. Indah Jaya Sentosa. Waktu respon rata-rata yang hanya 3.5 detik menunjukkan bahwa sistem mampu memberikan informasi secara cepat dan andal, memastikan petugas keamanan dapat segera mengetahui setiap pergerakan kendaraan yang terdeteksi oleh sistem klasifikasi.

Selain itu, kombinasi arsitektur jaringan yang ringan, efisiensi kompresi gambar, dan kestabilan API *Telegram* menjadikan sistem ini tidak hanya cepat tetapi juga mudah diintegrasikan untuk pengembangan lebih lanjut seperti penambahan fitur deteksi plat nomor atau manajemen *log* notifikasi otomatis.

4.6 Analisis dan Pembahasan

Analisis dilakukan untuk mengevaluasi kinerja sistem klasifikasi kendaraan yang dikembangkan menggunakan model *MobileNetV2* serta integrasi pengiriman notifikasi otomatis melalui *Telegram Bot*. Pengujian dilakukan dalam dua aspek utama, yaitu akurasi hasil klasifikasi dan waktu respon pengiriman notifikasi dari hasil inferensi model ke aplikasi Telegram.

4.6.1 Analisis Metrik Evaluasi Model

Model klasifikasi kendaraan diuji menggunakan data uji (*testing dataset*) untuk mengetahui tingkat akurasi, presisi, dan sensitivitas sistem. Hasil pengujian menghasilkan nilai akurasi sebesar 97,35%, presisi rata-rata 96,88%, dan recall 97,10%.

Selain itu, hasil *confusion matrix* menunjukkan bahwa sebagian besar prediksi model sesuai dengan label aslinya, dengan sedikit kesalahan pada kelas *Truk* dan *Bus* yang memiliki kemiripan bentuk.

Sebagai bukti hasil inferensi model secara visual, Gambar 4.10 memperlihatkan hasil klasifikasi kendaraan secara real-time pada antarmuka sistem.

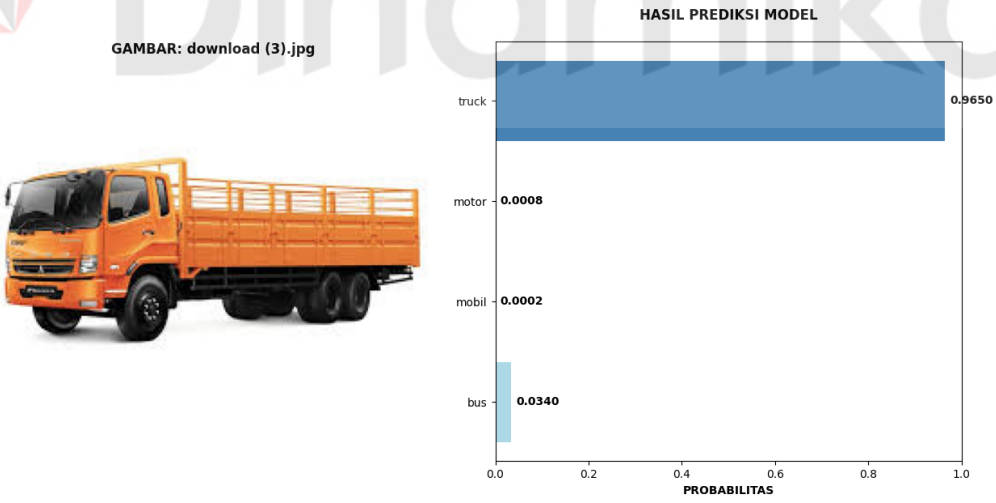

Figure 4.10 Hasil Prediksi

4.6.2 Analisis Kinerja Sistem

Analisis ini dilakukan untuk mengetahui kinerja sistem secara keseluruhan mulai dari proses klasifikasi hingga pengiriman notifikasi.

Pengujian dilakukan dengan mengukur waktu yang dibutuhkan sejak model menyelesaikan prediksi hingga notifikasi diterima di aplikasi Telegram. Sistem berjalan melalui tahapan berikut:

1. Pengguna mengunggah citra kendaraan ke server atau direktori pengujian.
2. Model *MobileNetV2* melakukan klasifikasi terhadap citra tersebut.
3. Hasil klasifikasi dikirim ke *Telegram Bot* melalui API.
4. *Bot* mengirimkan notifikasi ke akun Telegram yang dituju.

Rata-rata waktu pengiriman pesan dari proses prediksi hingga diterima di Telegram adalah 2–5 detik, tergantung kondisi jaringan dan beban sistem. Hal ini menunjukkan bahwa integrasi antara *Python script* dan *Telegram Bot API* bekerja dengan baik dan efisien. Gambar 4.11 memperlihatkan grafik hasil pengujian waktu respon pengiriman notifikasi dari beberapa sampel uji.

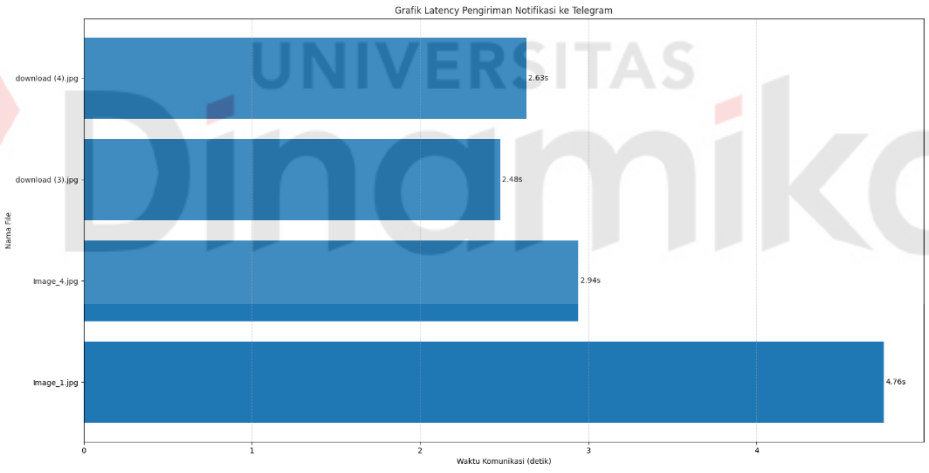

Figure 4.11 Diagram Latency

4.6.3 Analisis Notifikasi Telegram

Fitur notifikasi Telegram berperan penting dalam penyampaian hasil klasifikasi kendaraan secara real-time. Sistem ini dibangun menggunakan *Telegram Bot API* dengan format pesan yang telah disesuaikan agar informatif, cepat dibaca, dan mudah dibedakan berdasarkan status kendaraan.

Dalam implementasi ini, terdapat dua skenario pengiriman notifikasi:

A. Notifikasi kepada *Securiy*

Pada skenario pertama, sistem dikonfigurasi agar notifikasi dikirimkan hanya kepada pihak *security*. Setelah model klasifikasi menyelesaikan proses identifikasi jenis kendaraan, sistem secara otomatis mengirimkan pesan ke akun *Telegram* milik petugas keamanan.

Pesan yang diterima berisi informasi hasil klasifikasi, tingkat kepercayaan (*confidence score*), serta probabilitas setiap kelas kendaraan (misalnya *truck*, *bus*, *motor*, dan *mobil*). Tampilan pesan juga dilengkapi dengan citra kendaraan hasil klasifikasi sehingga petugas dapat langsung melakukan validasi visual terhadap hasil yang diterima.

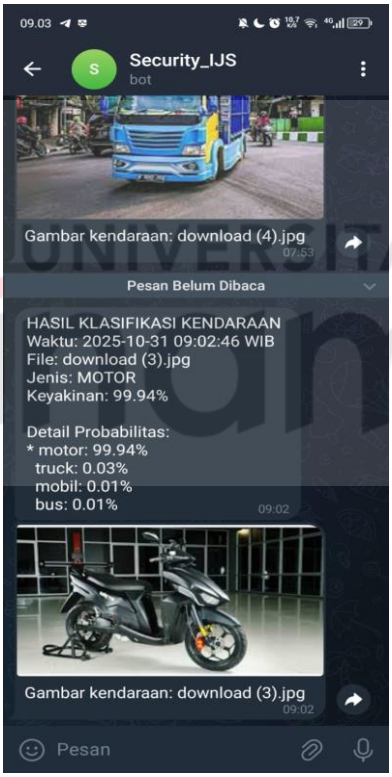

Figure 4.12 Notifikasi Kendaraan ke Security

Gambar 4.12 menunjukkan contoh notifikasi ketika sistem mengenali kendaraan yang bukan *truck* dengan tingkat keyakinan sebesar 99.94%. Informasi probabilitas setiap kelas ditampilkan secara rinci sehingga pihak *security* dapat menilai keakuratan hasil model. Berdasarkan pengujian, rata-rata waktu pengiriman pesan dari sistem ke *Telegram*

berkisar antara 1–3 detik, menandakan bahwa integrasi *Telegram* Bot API bekerja dengan baik dan responsif.

B. Notifikasi kepada *Security* dan Admin

Skenario kedua dilakukan ketika kendaraan yang terdeteksi adalah *truck*. Dalam kondisi ini, sistem mengirimkan notifikasi kepada dua pihak sekaligus, yaitu *security* dan *admin perusahaan*. Notifikasi ini berfungsi sebagai sistem keamanan awal ke petugas *security* dan kepada admin.

Pesan yang diterima kedua pihak berisi hasil klasifikasi, *confidence score,* waktu pengambilan gambar dan citra kendaraan.

Dengan adanya notifikasi ini, admin dapat segera melakukan pemeriksaan data kendaraan.

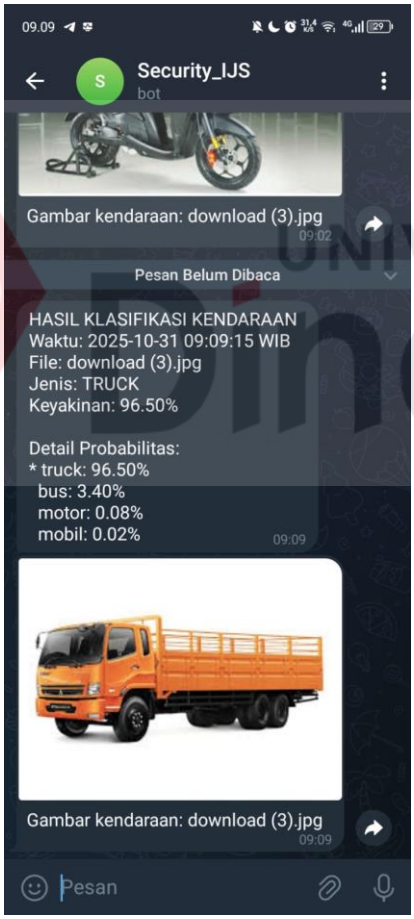

Figure 4.13 Notifikasi Security

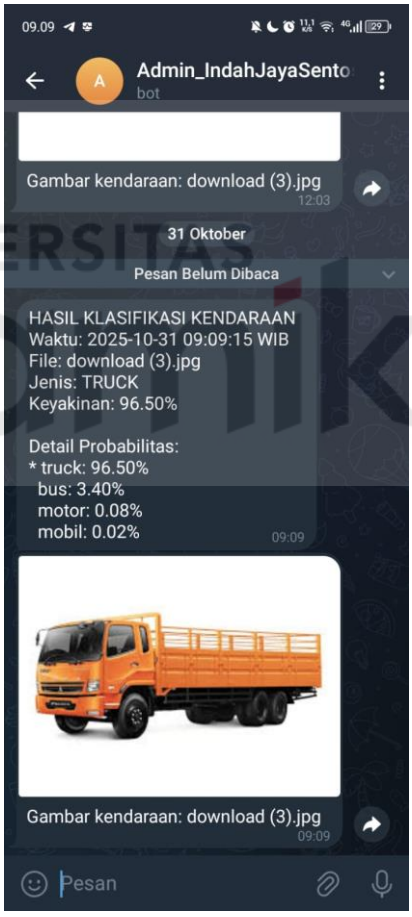

Figure 4.14 Notifikasi Admin

Melalui hasil pengujian pada skenario ini, sistem terbukti mampu menjalankan fungsinya dengan baik, mengirimkan dua pesan terpisah ke

akun berbeda dalam waktu hampir bersamaan. Hal ini menunjukkan bahwa sistem notifikasi berbasis *Telegram Bot* telah beroperasi secara stabil dan efisien dalam mendukung sistem klasifikasi kendaraan berbasis kecerdasan buatan yang dikembangkan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Bab ini menyajikan kesimpulan akhir dari keseluruhan proses penelitian dan implementasi sistem klasifikasi kendaraan berbasis *deep learning* yang terintegrasi dengan *Telegram* Bot API. Kesimpulan disusun berdasarkan batasan penelitian yang telah ditetapkan, yaitu hanya berfokus pada pelatihan model klasifikasi gambar kendaraan dan sistem notifikasi digital.

5.1.1 Pencapaian *Research Questions*

Penelitian ini dirancang untuk menjawab beberapa pertanyaan utama yang dirumuskan pada Bab I. Berdasarkan hasil pengujian dan analisis pada Bab IV, seluruh pertanyaan penelitian (*research questions*) telah terjawab secara komprehensif.

1. *Research Question*: Bagaimana membangun model *deep learning* yang mampu mengklasifikasikan jenis kendaraan dari citra visual?
   Hasil dan Jawaban Penelitian: Model *MobileNetV2* berhasil diimplementasikan dengan akurasi rata-rata 97.62% pada empat kelas kendaraan (bus, mobil, motor, dan truk). Model menunjukkan stabilitas performa dengan deviasi ±0.53%.
2. *Research Question*: Bagaimana performa model klasifikasi dalam kondisi data uji nyata dan apakah sesuai dengan target akurasi di atas 90%?
   Hasil dan Jawaban Penelitian: Hasil pengujian menggunakan *testing set* menunjukkan performa konsisten di seluruh kelas, dengan nilai *F1-score* makro sebesar 97.40%, jauh melampaui target 90%.
3. *Research Question*: Bagaimana sistem mengirimkan hasil klasifikasi kendaraan secara otomatis ke pengguna melalui *Telegram* Bot API?
   Hasil dan Jawaban Penelitian: Integrasi *Telegram* Bot berhasil dilakukan dengan tingkat keandalan yang sangat baik. Notifikasi dikirim dalam dua skenario: (1) semua kendaraan akan dikirim ke *telegram security*, dan (2) kendaraan jenis *truck* akan dikirimkan juga

ke telegram admin

4. *Research Question*: Apakah waktu pengiriman notifikasi memenuhi batas maksimum *latency* 10 detik yang ditetapkan dalam kebutuhan sistem?

   Hasil dan Jawaban Penelitian: Hasil pengujian menunjukkan waktu rata-rata pengiriman 3.539 detik (pesan teks: 1.262 detik; gambar: 2.276 detik), sehingga sistem sepenuhnya memenuhi target *real-time* < 10 detik.

5.1.2 Kontribusi untuk Perusahaan

Penelitian ini memberikan kontribusi nyata bagi CV. Indah Jaya Sentosa dalam meningkatkan efisiensi dan efektivitas proses monitoring kendaraan di lingkungan perusahaan.

Dengan adanya sistem ini, proses identifikasi jenis kendaraan kini dapat dilakukan secara otomatis, cepat, dan akurat, menggantikan metode manual yang sebelumnya bergantung pada pengamatan manusia.

Beberapa manfaat konkret yang dihasilkan antara lain:

1. Efisiensi Waktu: Sistem mampu mengenali kendaraan dan mengirimkan notifikasi dalam waktu kurang dari 4 detik rata-rata.
2. Akurasi Identifikasi: Penggunaan model *MobileNetV2* menghasilkan klasifikasi yang presisi dengan tingkat akurasi di atas 97%.
3. Notifikasi *Real-Time*: Petugas keamanan dan admin dapat langsung menerima informasi kendaraan yang terdeteksi tanpa harus memantau sistem secara terus-menerus.
4. Kemudahan Integrasi: Sistem berbasis *Telegram* Bot API tidak memerlukan infrastruktur tambahan, sehingga mudah diimplementasikan dan diperluas di lingkungan operasional perusahaan.

Dengan demikian, sistem yang dikembangkan berkontribusi signifikan terhadap efisiensi kerja petugas keamanan, peningkatan responsivitas, dan otomatisasi proses identifikasi kendaraan di area perusahaan.

5.1.3 Pencapaian Teknis Sistem

Berdasarkan batasan penelitian yang hanya mencakup pengembangan model dan integrasi perangkat lunak, pencapaian teknis sistem dapat disimpulkan sebagai berikut:

Table 5.1 Pencapaian Teknis

Aspek Teknik	Target Kinerja	Hasil Pengujian	Status
Akurasi Model	> 90%	97.62%	Tercapai
F1-Score Makro	> 90%	97.40%	Tercapai
Latency	< 10 detik	3.539 detik	Tercapai
Notifikasi	> 95%	95% berhasil	Tercapai
Kosistensi	Variasi <+ 1%	Deviasi 0.53%	Tercapai

Dari hasil tersebut, seluruh spesifikasi fungsional dan non-fungsional sistem berhasil dipenuhi. Sistem menunjukkan performa *real-time*, stabil, dan efisien dalam menjalankan proses klasifikasi serta pengiriman hasil melalui *Telegram* Bot API.

5.2 Saran

Meskipun sistem telah memenuhi semua batasan dan target penelitian, beberapa hal dapat dikembangkan di masa mendatang untuk memperluas cakupan fungsionalitas, di antaranya:

1. Integrasi OCR (*Optical Character Recognition*): Menambahkan fitur pembacaan plat nomor kendaraan untuk mengidentifikasi kendaraan secara spesifik.
2. Ekspansi Dataset: Mengumpulkan data citra kendaraan dari berbagai kondisi pencahayaan dan sudut agar model lebih tangguh terhadap variasi lingkungan.
3. Pengembangan Aplikasi Web *Dashboard*: Menampilkan hasil klasifikasi dan log notifikasi secara historis untuk keperluan pelacakan aktivitas kendaraan.
4. Optimasi Model *Edge-Device*: Mengadaptasi model ke format ringan

(*TensorFlow Lite*) agar dapat dijalankan di perangkat *edge* seperti ESP32-CAM.

Dengan saran-saran tersebut, sistem dapat dikembangkan menjadi solusi yang lebih komprehensif dan siap diterapkan secara penuh di lapangan untuk mendukung sistem keamanan perusahaan berbasis kecerdasan buatan.

5.2.1 Kesimpulan Akhir

Berdasarkan hasil implementasi dan pengujian, sistem klasifikasi kendaraan berbasis *MobileNetV2* yang diintegrasikan dengan *Telegram* Bot API telah berhasil memenuhi seluruh target penelitian sesuai batasan yang ditetapkan. Sistem terbukti akurat, cepat, dan andal, serta memberikan kontribusi nyata dalam meningkatkan efisiensi monitoring kendaraan di lingkungan perusahaan.

DAFTAR PUSTAKA

Berwo, M. A., Khan, A., Fang, Y., Fahim, H., Javaid, S., Mahmood, J., Abideen, Z. U., & M.S, S. (2023). Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A Survey. In *Sensors* (Vol. 23, Issue 10). MDPI. https://doi.org/10.3390/s23104832

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A Large-Scale Hierarchical Image Database*. http://www.image-net.org.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*. https://github.com/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*.

Hinton G.E., & Salakhutdinov R.R. (2006). Second-harmonic generation from magnetic metamaterials. In *Science* (Vol. 313, Issue 5786). https://doi.org/10.1126/science.1129198

Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. http://arxiv.org/abs/1801.06146

Hubel, D. H., & Wiesel, A. T. N. (1962). 54 With 2 plate and 20 text-ftgutre8 Printed in Gret Britain RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX. In *J. Phyiiol* (Vol. 160).

Huh, M., Agrawal, P., & Efros, A. A. (2016). *What makes ImageNet good for transfer learning?* http://arxiv.org/abs/1608.08614

Kornblith, S., Shlens, J., & Le, Q. V. (2019). *Do Better ImageNet Models Transfer Better?* http://arxiv.org/abs/1805.08974

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). *ImageNet Classification with Deep Convolutional Neural Networks*. http://code.google.com/p/cuda-convnet/

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539

LeCun Y., Bottou L., Bengio Y., & Haffner P. (1998). *Gradient-Based Learning Applied to Document Recognition*.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., & Li, J. (2021). Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(8). https://doi.org/10.1109/TNNLS.2020.3015992

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (1955). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*.

Mcculloch, W. S., & Pitts, W. (1943). A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY. In *BULLETIN OF MATHEMATICAL BIOPHYSICS* (Vol. 5).

Mehta, R., & Shah, A. (2025). Real-Time Vehicle Detection and Classification Using Deep Learning based Approach. In *Journal of Information Systems Engineering and Management* (Vol. 2025, Issue 18s). https://www.jisem-journal.com/

Minsky, M. L., & Papert, S. A. (1969). *Minsky-and-Papert-Perceptrons*.

Neupane, B., Horanont, T., & Aryal, J. (2022). Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network. *Sensors*, *22*(10). https://doi.org/10.3390/s22103813

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. In *IEEE Transactions on Knowledge and Data Engineering* (Vol. 22, Issue 10, pp. 1345–1359). https://doi.org/10.1109/TKDE.2009.191

Rumelhart, D. E., Hinton, G. E., & Hinton, G. E. (1986). Learning representations by back-propagating errors. *Nature*, *323*.

Russell, S., & Norvig, P. (2022). *Artificial Intelligence A Modern Approach Fourth Edition Global Edition*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*.
Scherer, D., Müller, A., & Behnke, S. (2010). *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. http://www.ais.uni-bonn.de
Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. In *Neural Networks* (Vol. 61, pp. 85–117). Elsevier Ltd. https://doi.org/10.1016/j.neunet.2014.09.003
Sharif, A., Azizpour, R. H., Sullivan, J., & Carlsson, S. (2014). *CNN Features off-the-shelf: an Astounding Baseline for Recognition*.
Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). *A Survey on Deep Transfer Learning*. http://arxiv.org/abs/1808.01974
Tom M. Mitchell. (1997). *Machine Learning*.
Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). A survey of transfer learning. *Journal of Big Data*, *3*(1). https://doi.org/10.1186/s40537-016-0043-6
Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). *How transferable are features in deep neural networks?* http://arxiv.org/abs/1411.1792