TESIS

OPTIMASI PERFORMA DETEKSI KENDARAAN DENGAN MENGGABUNGKAN ALGORITMA HAAR CASCADE CLASSIFIER DAN CONVOLUTIONAL NEURAL NETWORK (CNN)

Disusun oleh:

Nama : **Indra Irawanto**
NIM : **21.55.2160**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

TESIS

OPTIMASI PERFORMA DETEKSI KENDARAAN DENGAN MENGGABUNGKAN ALGORITMA HAAR CASCADE CLASSIFIER DAN CONVOLUTIONAL NEURAL NETWORK (CNN)

OPTIMIZING VEHICLE DETECTION PERFORMANCE BY INTEGRATING HAAR CASCADE CLASSIFIER ALGORITHM AND CONVOLUTIONAL NEURAL NETWORK (CNN)

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama	**: Indra Irawanto**
NIM	**: 21.55.2160**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2024

HALAMAN PENGESAHAN

OPTIMASI PERFORMA DETEKSI KENDARAAN DENGAN MENGGABUNGKAN ALGORITMA HAAR CASCADE CLASSIFIER DAN CONVOLUTIONAL NEURAL NETWORK (CNN)

OPTIMIZING VEHICLE DETECTION PERFORMANCE BY INTEGRATING HAAR CASCADE CLASSIFIER ALGORITHM AND CONVOLUTIONAL NEURAL NETWORK (CNN)

Dipersiapkan dan Disusun oleh

Indra Irawanto

21.55.2160

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 08 Juli 2024

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, Senin, 08 Juli 2024
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

OPTIMASI PERFORMA DETEKSI KENDARAAN DENGAN MENGGABUNGKAN ALGORITMA HAAR CASCADE CLASSIFIER DAN CONVOLUTIONAL NEURAL NETWORK (CNN)

OPTIMIZING VEHICLE DETECTION PERFORMANCE BY INTEGRATING HAAR CASCADE CLASSIFIER ALGORITHM AND CONVOLUTIONAL NEURAL NETWORK (CNN)

Dipersiapkan dan Disusun oleh

Indra Irawanto

21.55.2160

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 08 Juli 2024

Pembimbing Utama

Dr. Andi Sunyoto, M. Kom.
NIK. 190302052

Anggota Tim Penguji

Dhani A., S.Kom., M.Kom., Ph.D.
NIK. 190302197

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

Pembimbing Pendamping

Kusnawi, S.Kom., M.Eng.
NIK. 190302112

Dr. Andi Sunyoto, M. Kom.
NIK. 190302052

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, Senin 08 Juli 2024
Direktur Program Pascasarjana

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Indra Irawanto
NIM : 21.55.2160
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Optimasi Performa Deteksi Kendaraan Dengan Menggabungkan Algoritma Haar Cascade Classifier Dan Convolutional Neural Network (CNN)
Dosen Pembimbing Utama : Dr. Andi Sunyoto, M.Kom.
Dosen Pembimbing Pendamping : Kusnawi, S.Kom., M.Eng.

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, Senin, 08 Juli 2024
Yang Menyatakan,

Indra Irawanto

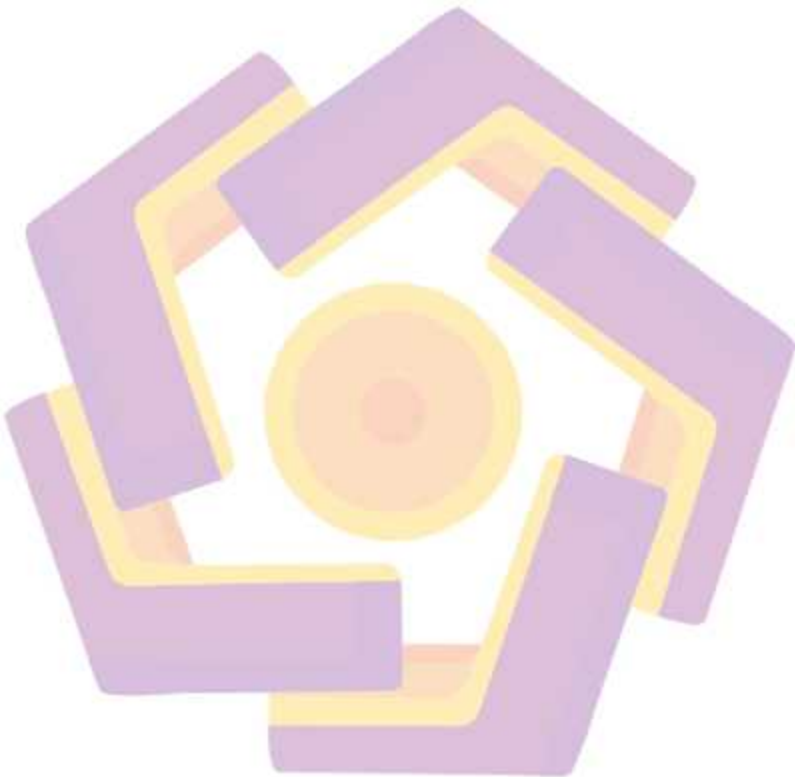
HALAMAN PERSEMBAHAN

Segala puji dan syukur kehadirat Allah SWT atas berkah, rahmat dan hidayah-Nya yang senantiasa dilimpahakan kepada kami, sehingga dapat merampungkan Tesis yang berjudul "**Optimasi Performa Deteksi Kendaraan Dengan Menggabungkan Algoritma Haar Cascade Classifier Dan Convolutional Neural Network (CNN)**". Semoga dapat diterima sebagai salah satu amal kebaikan.

Selaras dengan harapan kami. Penelitian ini kami persembahkan kepada kedua orang tua terkasih kami. Dengan segala bentuk dukungan lahir dan batin, kami bisa menyelesaikan studi Program Magister (S2) pada Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI) Universitas Amikom Yogyakarta.

HALAMAN MOTTO

"Setiap langkah dalam proses membawa kita bukan pada kesalahan, melainkan pada kesempatan untuk belajar dan bertumbuh."



KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT atas berkah, rahmat, dan hidayah-Nya yang senantiasa dilimpahkan kepada penulis, sehingga penulis dapat merampungkan Tesis yang berjudul "**Optimasi Performa Deteksi Kendaraan Dengan Menggabungkan Algoritma Haar Cascade Classifier Dan Convolutional Neural Network (CNN)**" ini dengan baik, sebagai salah satu syarat untuk menyelesaikan Program Magister (S2) pada Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI) Universitas Amikom Yogyakarta.

Dalam penyusunan Tesis ini, banyak hambatan serta rintangan yang penulis hadapi. Namun, pada akhirnya dapat dilalui berkat adanya dukungan dan bantuan dari berbagai pihak, baik secara moral maupun spiritual. Oleh karena itu, pada kesempatan ini, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Bapak Prof. Dr. M. Suyanto, M.M., selaku Rektor Universitas Amikom Yogyakarta.
2. Ibu Prof. Dr. Kusrini, M.Kom., selaku Direktur Program Pasca Sarjana Universitas Amikom Yogyakarta yang telah memberikan kesempatan dan izin untuk menempuh studi lanjut di Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI).
3. Bapak Dr. Andi Sunyoto, M.Kom., selaku pembimbing utama, yang bersedia menyempatkan waktu di sela-sela kesibukan beliau untuk memberikan dukungan, membimbing, mengoreksi, dan mengarahkan penulis demi kesempurnaan penulisan penelitian ini.

4. Bapak Kusnawi, S.Kom., M.Eng., selaku pembimbing pendamping yang di sela-sela kesibukannya dapat memberikan arahan, pendapat, dan semangat agar penulis dapat menyelesaikan penelitian.
5. Tim Penguji dari SPT, SHPT, hingga UT yang telah memberikan arahan dan wawasan lebih dalam proses penyempurnaan penulisan.
6. Seluruh Dosen Pengajar di Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI) Universitas Amikom Yogyakarta dari semester pertama hingga terakhir yang memberikan arahan, dukungan, semangat, dan berbagi pengetahuan sehingga penulis mendapatkan wawasan baru yang lebih luas dalam menyelesaikan tugas di setiap studi.
7. Segenap Civitas Akademika (Pengelola dan Admisi) Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI) Universitas Amikom Yogyakarta yang telah memberikan pelayanan dan bantuan sangat baik dalam kebutuhan studi.
8. Ibu dan Bapak terkasih, terima kasih atas segala doa dan dukungan. Penulis percaya bahwa setiap kemudahan dan kesuksesan yang penulis capai bukanlah semata-mata hasil jerih payah penulis, melainkan berkat secercah doa kedua orang tua yang menembus langit.
9. Teman-teman terdekat, rekan-rekan, dan guru-guru penulis, utamanya yang berada di bawah naungan Pondok Pesantren Nurul Jadid, terkhusus Fakultas Teknik Universitas Nurul Jadid. Terima kasih atas kesediaannya berbagi keluh kesah, sedih senang, tawa duka. Penulis tidak akan sampai pada titik

ini tanpa dukungan dan pendampingan kalian selama ini. Salam hormat penulis.

10. Teman-teman Mahasiswa Program Studi Pendidikan Jarak Jauh Magister Teknik Informatika (PJJ-MTI) Universitas Amikom Yogyakarta Angkatan 2021 Genap yang telah memberikan pengalaman, suasana, dan keluarga baru.
11. Istri tercinta yang selalu memberikan cinta, dukungan, dan kesabaran yang tak ternilai selama penulis menjalani studi ini. Tanpa kehadiran dan semangat darinya, penulis mungkin tidak akan mampu menyelesaikan penelitian ini.
12. Kedua mertua yang dengan penuh kasih sayang dan doa telah memberikan dukungan kepada penulis. Kehadiran mereka telah memberikan kekuatan tambahan bagi penulis dalam menghadapi setiap tantangan selama proses studi ini.

Penulis dengan senang hati menerima kritik dan saran yang membangun dari pembaca. Penulis menyadari sepenuhnya bahwa dalam penyusunan penelitian ini masih terdapat banyak kekurangan. Akhir kata, semoga penelitian ini dapat memberikan manfaat bagi pembacanya.

Yogyakarta, 08 Juli 2024

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Kemacetan lalu lintas menjadi masalah yang semakin meningkat seiring dengan meningkatnya jumlah pengguna kendaraan pribadi dibandingkan dengan transportasi umum untuk mobilitas sehari-hari. Berbagai solusi telah diusulkan untuk mengatasi masalah ini, salah satunya adalah implementasi Lampu Lalu Lintas Pintar untuk mengoptimalkan pengaturan lalu lintas. Beberapa metode dan algoritma deteksi objek telah diuji untuk menemukan pendekatan yang lebih efektif dalam pengenalan kendaraan di lalu lintas. Penelitian ini bertujuan untuk meningkatkan performa deteksi dan klasifikasi kendaraan dengan menggabungkan kelebihan dari algoritma *Haar Cascade Classifier* dan *Convolutional Neural Network* (CNN). Fokus penelitian ini adalah pada integrasi *Haar Cascade Classifier* dan CNN untuk pengenalan objek kendaraan, dengan penekanan pada evaluasi kinerja arsitektur CNN, khususnya *Xception* dan *VGG16*. Dataset yang digunakan adalah *BIT-Vehicle*, dan penelitian ini melibatkan *preprocessing* data, pelatihan model, serta pengujian. Pendekatan yang diusulkan terbukti efektif dalam mengoptimalkan performa deteksi dan klasifikasi kendaraan, dibuktikan dengan rata-rata akurasi sebesar 92.6% yang dicapai dengan model Xception. Selain itu, dengan mengimplementasikan *Bayesian Optimization* dalam proses pelatihan, akurasi model *Xception* meningkat lebih lanjut menjadi 95.3%. Peningkatan signifikan ini menunjukkan dampak *Bayesian Optimization* terhadap keberhasilan kombinasi *Haar Cascade Classifier* dan CNN dalam meningkatkan performa deteksi dan klasifikasi kendaraan.

Kata kunci: *Kemacetan Lalu Lintas, Lampu Lalu Lintas Pintar, Haar Cascade Classifier, Convolutional Neural Network, Xception, VGG16, Optimisasi Bayesian*

ABSTRACT

Traffic congestion has become an increasingly significant issue as the number of private vehicle users rises compared to public transportation for daily mobility. Various solutions have been proposed to address this problem, one of which is the implementation of Smart Traffic Lights to optimize traffic management. Several methods and object detection algorithms have been tested to find more effective approaches for vehicle recognition in traffic. This study aims to improve vehicle detection and classification performance by combining the strengths of the Haar Cascade Classifier and Convolutional Neural Network (CNN) algorithms. The focus of this research is on the integration of Haar Cascade Classifier and CNN for vehicle object recognition, with an emphasis on evaluating the performance of CNN architectures, specifically Xception and VGG16. The dataset used is BIT-Vehicle, and the study involves data preprocessing, model training, and testing. The proposed approach has proven effective in optimizing vehicle detection and classification performance, as evidenced by an average accuracy of 92.6% achieved with the Xception model. Additionally, by implementing Bayesian optimization in the training process, the accuracy of the Xception model was further improved to 95.3%. This significant improvement demonstrates the impact of Bayesian optimization on the successful combination of Haar Cascade Classifier and CNN in enhancing vehicle detection and classification performance.

Keywords: *Traffic Congestion, Smart Traffic Lights, Haar Cascade Classifier, Convolutional Neural Network, Xception, VGG16, Bayesian Optimization.*

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Bersamaan dengan kemajuan teknologi, sistem transportasi di Indonesia memainkan peran penting dalam mendukung efisiensi dan efektivitas sarana transportasi (Hafram & Asrib, 2022). Namun, semakin tingginya permintaan akan transportasi akibat kebutuhan sehari-hari masyarakat telah menyebabkan peningkatan penggunaan kendaraan pribadi daripada angkutan umum. Dampak dari pertumbuhan tersebut adalah kemacetan lalu lintas yang meluas karena jumlah kendaraan terus meningkat. Selain itu, masalah kemacetan juga disebabkan oleh keterbatasan sistem konvensional pada lampu lalu lintas (Hasanah et al., 2021).

Permasalahan kemacetan dapat diatasi dengan cara melakukan pendekatan melalui implementasi sistem Lampu Lalu Lintas Pintar (Smart Traffic Light) (Hasanah et al., 2021). Sistem ini menggunakan teknologi pendeteksian objek kendaraan terkini untuk mengatur durasi lampu lalu lintas secara dinamis berdasarkan kondisi aktual lalu lintas di setiap persimpangan jalan. Pada kasus ini, penting untuk membangun sistem deteksi dan klasifikasi kendaraan berdasarkan tingkat kepadatan lalu lintas sebagai langkah awal dalam mengoptimalkan pengaturan lalu lintas.

Berbagai penelitian telah dilakukan untuk menguji beberapa metode deteksi objek guna menemukan sebuah metode yang lebih baik untuk diterapkan pada pengenalan objek kendaraan di lalu lintas. (Butt et al., 2021a) yang fokus pada klasifikasi kendaraan dalam kondisi pencahayaan yang buruk, memberikan

kontribusi penting untuk pengembangan sistem transportasi cerdas. Penelitian tersebut menggunakan metode *Convolutional Neural Network* (CNN) dengan lima arsitektur yang berbeda seperti *AlexNet*, *VGG*, *GoogleNet*, *Inception-v3*, dan *ResNet*. Hasil penelitian menunjukkan bahwa arsitektur *ResNet* lebih baik dibandingkan dengan arsitektur lainnya. Penggunaan metode *CNN* juga dilakukan oleh Alghamdi et al., (2023) untuk menentukan model klasifikasi kendaraan yang paling optimal. Penelitian ini menggunakan arsitektur *VGG16* untuk mengekstraksi fitur dari gambar kendaraan. Fitur-fitur tersebut kemudian dioptimasi menggunakan *Genetic Algorithms* (GA) sebelum akhirnya diklasifikasikan melalui metode *Support Vector Machine* (SVM). Hasil eksperimen menunjukkan bahwa model yang diusulkan mencapai akurasi 99,78% dan mengungguli model sebelumnya dalam hal akurasi dan waktu pelatihan. Kemudian dari Khalifa et al., (2022) yang melakukan eksperimen dengan menganalisis berbagai metode deteksi kendaraan serta menguji penggunaan CNN berarsitektur YOLOv5s. Dalam pengujian ini, kombinasi tersebut juga melakukan optimalisasi kotak anchor melalui algoritma *K-Means*, yang diuji menggunakan dataset dalam dua kondisi: siang dan malam hari. Hasil eksperimen menunjukkan bahwa model yang diusulkan mencapai *mean average precision* (mAP) sebesar 97,8 pada kondisi siang dan 95,1 pada kondisi malam. (Avianto et al., 2022) juga melakukan eksperimen *CNN* dengan pendekatan *multi-task learning* untuk menghadapi tantangan dalam mengklasifikasikan merek dan model kendaraan yang sangat serupa. Hasil eksperimen menunjukkan bahwa metode yang diusulkan mencapai akurasi 98,73% untuk merek kendaraan dan 97,69% untuk model kendaraan. Hasil pengujian dari eksperimen yang dilakukan

oleh Chauhan et al., (2019) menunjukkan bahwa model CNN yang dilatih mencapai akurasi hingga 75% dalam mengklasifikasikan kendaraan. Penelitian terakhir yang dilakukan oleh Jahan et al., (2020) yang membahas penerapan *CNN* dalam pengklasifikasian kendaraan secara real-time. Tujuannya adalah mengurangi kecelakaan lalu lintas akibat pelanggaran aturan dengan mengidentifikasi jenis kendaraan yang terlibat. Mereka mengumpulkan dataset gambar dari empat jenis kendaraan umum: mobil, CNG, becak, dan sepeda. Menggunakan metode CNN, mereka berhasil melatih model dengan 2240 gambar latihan dan 560 gambar pengujian setelah mengatasi tantangan pengumpulan dan pemrosesan dataset. Hasil eksperimen menunjukkan model CNN mencapai akurasi sekitar 97% dalam mengklasifikasikan jenis kendaraan. Penelitian terbaru oleh Maiga et al. (2023) dalam jurnalnya mengkaji kelebihan dari arsitektur *Xception* dalam klasifikasi kendaraan. *Xception* memanfaatkan *depthwise separable convolutions* yang mengurangi jumlah parameter dan komputasi yang diperlukan, memungkinkan efisiensi komputasi yang lebih tinggi dan performa yang lebih baik dibandingkan *convolutions* konvensional. Hal ini menjadikan *Xception* sangat cocok untuk sistem dengan sumber daya terbatas. Dengan kemampuan untuk menangkap fitur-fitur yang lebih kompleks dan abstrak, arsitektur ini berkontribusi pada peningkatan akurasi dalam tugas klasifikasi dan deteksi kendaraan (Maiga et al., 2023).

Penggunaan teknik seperti *Haar Cascade Classifier* dan *CNN* memungkinkan deteksi kendaraan yang akurat dalam gambar atau video lalu lintas Singh Bhatia et al., (2020). Informasi yang diperoleh tentang jumlah kendaraan yang terdeteksi digunakan untuk mengatur alokasi waktu lampu lalu lintas di setiap

persimpangan jalan. Selain itu, klasifikasi kendaraan berdasarkan tingkat kepadatan lalu lintas menggunakan algoritma *Machine Learning* atau *Deep Learning* memungkinkan pengelompokkan kendaraan ke dalam kategori yang sesuai Singh Bhatia et al., (2020). Informasi klasifikasi ini digunakan untuk mengoptimalkan pengaturan waktu lampu lalu lintas dengan tujuan mengurangi kemacetan dan meningkatkan efisiensi lalu lintas secara keseluruhan.

Beijing Institute of Technology (BIT)-Vehicle dataset (Dong et al., 2015a) yang digunakan pada penelitian ini berupa *secondary dataset* yang didapatkan dari penelitian-penelitian sebelumnya. Dataset yang didapatkan kemudian akan diolah dan dibagi menjadi data *training*, data *validation*, dan data *testing*. Kami melakukan pra-pemrosesan pada dataset tersebut untuk meningkatkan kualitasnya dan mengurangi kebisingan yang terdapat dalam gambar-gambar tersebut. Setelah pra-pemrosesan, kami melatih dua model, yaitu model *Haar Cascade Classifer* dan model *Convolutional Neural Network* (CNN), menggunakan dataset yang sama untuk membandingkan performa keduanya.

Seiring dengan perkembangan teknologi dan penerapan berbagai model untuk deteksi dan klasifikasi kendaraan, muncul kebutuhan untuk mengoptimalkan performa model yang digunakan. Dalam pengembangan model deteksi dan klasifikasi kendaraan, optimasi hiperparameter memegang peran yang sangat penting. Proses optimasi hiperparameter adalah pencarian kombinasi parameter terbaik untuk meningkatkan performa model (Elshewey et al., 2023). Hiperparameter yang umum dioptimasi meliputi learning rate, dropout rate, dan jumlah filters pada lapisan konvolusi. Optimasi yang tepat dapat meningkatkan

akurasi model, mengurangi kesalahan klasifikasi, dan mempercepat waktu pelatihan (Elshewey et al., 2023).

Berbagai penelitian telah berhasil menghadirkan solusi-solusi cerdas yang dapat membantu mengatasi masalah kemacetan dan meningkatkan efisiensi lalu lintas. Penggunaan CNN dalam mengklasifikasikan jenis kendaraan telah terbukti mampu mencapai tingkat akurasi yang tinggi, seperti pada penelitian Khalifa et al., (2022) dan Alghamdi et al., (2023) yang menggunakan arsitektur *VGG16* dan teknik optimasi untuk meningkatkan performa klasifikasi. Selain itu, *Haar Cascade Classifier* juga memberikan kontribusi penting dalam mendeteksi objek kendaraan secara akurat dan real-time, seperti yang ditemukan dalam penelitian Singh Bhatia et al., (2020). Penggunaan algoritma ini dapat memberikan informasi tentang jumlah kendaraan yang terdeteksi, yang nantinya digunakan untuk mengatur alokasi waktu lampu lalu lintas dan mengoptimalkan efisiensi lalu lintas secara keseluruhan. Namun, pada titik ini, masih terdapat ruang untuk pengembangan lebih lanjut. Penelitian (Avianto et al., 2022) menunjukkan bahwa penggabungan antara CNN dan pendekatan *multi-task learning* mampu mencapai akurasi yang signifikan dalam mengklasifikasikan merek dan model kendaraan yang sangat serupa. Hasil eksperimen dari penelitian ini memberikan landasan kuat untuk usulan penelitian selanjutnya, yang bertujuan untuk mengoptimalkan performa deteksi dan klasifikasi kendaraan dengan menggabungkan algoritma *Haar Cascade Classifier* dan CNN dengan menggunakan arsitektur *Xception* yang kemudian akan dibandingkan dengan arsitektur VGG16. Pemilihan arsitektur *Xception* dan *VGG16* didasarkan pada penelitian-penelitian sebelumnya seperti penelitian yang dilakukan

oleh Maiga et al., (2023) yang menunjukkan keunggulan dari arsitektur *Xception* dalam tugas klasifikasi gambar. Dengan memadukan potensi kedua teknik tersebut, penelitian yang diusulkan dapat memperoleh keunggulan dalam mendeteksi dan mengklasifikasikan kendaraan secara lebih akurat dan efisien, serta memberikan solusi yang dapat membantu mengurangi kemacetan dan meningkatkan keamanan lalu lintas.

1.2 Rumusan Masalah

Berdasarkan latar belakang terpapar dibagian 1, maka rumusan masalah pada penelitian ini adalah sebagai berikut:

a. Apakah penggunaan arsitektur *Xception* dapat meningkatkan performa deteksi dan klasifikasi kendaraan?
b. Apakah performa klasifikasi kendaraan dapat dimaksimalkan dengan mengoptimalkan penggunaan metode *Haar Cascade Classifier* dan CNN?

1.3 Batasan Masalah

Berikut batasan-batasan masalah pada penelitian ini:

a. Fokus dari penelitian ini adalah pada penggabungan metode *Haar Cascade Classifier* dan CNN dengan membandingkan arsitektur *Xception* dan *VGG16* dalam rangka meningkatkan performa deteksi dan klasifikasi kendaraan.
b. Penelitian ini difokuskan pada evaluasi kinerja metode CNN dengan membandingkan arsitektur *VGG16* dan *Xception* dalam pengenalan objek kendaraan, mengacu pada arsitektur terbaik yang telah teruji dalam penelitian-penelitian sebelumnya.

c. *Dataset* diperoleh dari Dong et al., (2015) yang merupakan sekumpulan gambar kendaraan sebagai sumber data utama untuk pelatihan model.
d. Penelitian ini tidak akan mencakup pengujian pada situasi lalu lintas yang ekstrem atau diluar cakupan dataset yang digunakan.
e. Penelitian ini akan mengklasifikasikan tiga jenis kendaraan, yaitu bus, mobil, dan truk, berdasarkan temuan-temuan penelitian sebelumnya.

1.4 Tujuan Penelitian

Bagian ini memuat penjelasan secara spesifik:

a. Menganalisis pengaruh arsitektur *VGG16* dan *Xception* pada metode CNN terhadap performa dalam pengenalan objek kendaraan.
b. Merancang solusi optimal untuk deteksi dan klasifikasi kendaraan dengan kualitas performa yang tinggi dalam kondisi lalu lintas yang kompleks melalui pendekatan gabungan *Haar Cascade Classifier* dan CNN.
c. Mengevaluasi performa deteksi dan klasifikasi kendaraan menggunakan kombinasi *Haar Cascade Classifier* dan CNN.
d. Membandingkan dan menganalisis hasil deteksi dan klasifikasi kendaraan yang dicapai melalui pendekatan kombinasi *Haar Cascade Classifier* dan CNN.

1.5 Manfaat Penelitian

Manfaat yang dapat diambil pada penelitian ini adalah sebagai berikut:

a. Hasil dari penelitian ini dapat menjadi dasar atau landasan untuk penelitian lebih lanjut yang mungkin ingin mengembangkan atau memperbaiki metode yang diusulkan.

b. Penelitian ini dapat menjadi rujukan bagi penelitian selanjutnya yang ingin membandingkan metode deteksi kendaraan atau menguji metode pada situasi yang lebih kompleks.

c. Dapat menjadi titik awal untuk penelitian lebih mendalam dalam mengoptimalkan parameter, arsitektur, atau teknik lainnya yang digunakan dalam deteksi kendaraan.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian ini merujuk pada sejumlah penelitian sebelumnya sebagai landasan dalam tinjauan pustaka untuk memberikan dukungan dan sebagai upaya penyempurnaan dari penelitian-penelitian sebelumnya. Berikut ini beberapa penelitian yang dijadikan referensi dalam penelitian ini.

Butt et al., (2021) mengusulkan sistem klasifikasi kendaraan berbasis CNN untuk meningkatkan efektivitas Sistem Transportasi Cerdas. Penelitian ini mengumpulkan dataset baru yang terdiri dari 10.000 gambar kendaraan dengan enam kelas yang mencakup kendaraan jalan umum di negara-negara Asia. Arsitektur CNN yang ada seperti AlexNet, VGG, GoogleNet, Inception-v3, dan ResNet disesuaikan dengan dataset yang dikumpulkan untuk memperoleh arsitektur akhir yang terbaik. Arsitektur tersebut kemudian disesuaikan lebih lanjut dengan dataset *VeRi* publik untuk meningkatkan kinerja dalam sistem transportasi cerdas yang berbeda. Hasil eksperimen menunjukkan bahwa sistem klasifikasi yang diusulkan mencapai akurasi yang lebih tinggi dibandingkan dengan sistem klasifikasi kendaraan yang ada. Saran dari penelitian ini ialah mempertimbangkan penggunaan teknik augmentasi data untuk meningkatkan keberagaman dataset.

Selanjutnya, penelitian yang dilakukan oleh Alghamdi et al., (2023) yang membahas mengenai klasifikasi kendaraan dengan penerapan *deep learning* dan *Genetic Algorithms* (GA). Fokus utama dari penelitian ini adalah mengembangkan model klasifikasi kendaraan yang menggabungkan fitur-fitur yang telah diambil

dari *CNN* dengan menggunakan arsitektur VGG16. Fitur-fitur ini dioptimasi menggunakan *Genetic Algorithms* (GA) untuk mencapai tingkat akurasi yang tinggi dalam mengklasifikasikan kendaraan. Metode penelitian ini melibatkan ekstraksi fitur menggunakan VGG16, optimisasi fitur dengan memanfaatkan algoritma genetika, dan klasifikasi menggunakan SVM. Hasil eksperimen menunjukkan bahwa model yang diajukan berhasil mencapai tingkat akurasi sebesar 99,78%, yang mengalami peningkatan yang signifikan dibandingkan dengan penelitian sebelumnya.

Khalifa et al., (2022) mengfokuskan pada deteksi kendaraan dalam Sistem Transportasi Cerdas (ITS) menggunakan Convolutional Neural Network (CNN). Penelitian ini bertujuan untuk mengoptimasi metode deteksi dengan memadukan CNN dengan arsitektur YOLOv5s dan algoritma k-means untuk memperbaiki performa deteksi kendaraan pada kondisi siang dan malam hari. Hasil penelitian menunjukkan model yang diusulkan mampu mencapai mean average precision (mAP) sebesar 97,8% pada siang hari dan 95,1% pada malam hari. Meski demikian, terdapat kekurangan pada waktu pelatihan model yang cukup lama. Rekomendasi yang dapat diambil dari penelitian ini adalah mengeksplorasi metode yang dapat mengurangi waktu pelatihan tanpa mengorbankan akurasi deteksi.

Penelitian berikutnya dilakukan oleh Fajri et al., (2020) yang membahas pengembangan program deteksi dan pengklasifikasi jenis kendaraan dengan CNN dan algoritma YOLO dalam konteks Deep Learning. Penelitian ini menggunakan citra kendaraan dengan empat kelas utama: bis, mobil, sepeda motor, dan truk. Metode CNN digunakan untuk ekstraksi fitur citra kendaraan, sementara algoritma

YOLO digunakan untuk deteksi real-time. Hasil pengujian menunjukkan akurasi sekitar 91,4% dalam mengklasifikasikan kendaraan dan akurasi berkisar antara 70% hingga 88,4% dalam uji coba faktor lingkungan.

Chauhan et al., (2019) mengaplikasikan *CNN* dalam mengklasifikasikan dan menghitung kendaraan dalam lalu lintas *non-laned* yang kompleks. Tujuannya adalah untuk mengotomatisasi proses klasifikasi dan penghitungan kendaraan dalam lingkungan lalu lintas yang tidak teratur. Penelitian ini menggunakan model CNN yang dilatih dengan data dari jalan-jalan di Delhi-NCR. Hasil pengujian mengindikasikan bahwa model CNN yang dihasilkan mencapai akurasi 75% dalam mengklasifikasikan kendaraan. Namun, peneliti mengakui kelemahan dalam menggunakan dataset anotasi dari negara maju, sehingga mereka membuat dataset sendiri dari video yang dikumpulkan di Delhi-NCR.

Penelitian selanjutnya dilakukan oleh Avianto et al., (2022) yang bertujuan untuk mengatasi tantangan dalam mengklasifikasikan merek dan model kendaraan yang sangat mirip. Metode yang diusulkan menggunakan arsitektur *CNN* dengan pendekatan *multi-task learning*. Pendekatan *multi-task learning* memungkinkan jaringan untuk mempelajari informasi merek dan model kendaraan secara bersamaan, dengan menggabungkan informasi produsen kendaraan ke dalam proses pembelajaran. Metode ini dievaluasi menggunakan *dataset InaV-Dash*, yang terdiri dari gambar kendaraan di Indonesia dengan penampilan yang sangat mirip. Hasil eksperimen menunjukkan bahwa metode yang diusulkan mencapai akurasi 98,73% untuk merek kendaraan dan 97,69% untuk model kendaraan. Hal ini menunjukkan bahwa metode tersebut mampu mengklasifikasikan kendaraan dengan tingkat

akurasi yang tinggi. Selain itu, penelitian ini juga menunjukkan bahwa metode yang diusulkan dapat meningkatkan kinerja metode dasar pada masalah klasifikasi kendaraan yang sangat mirip.

Penelitian yang dilakukan oleh Jahan et al., (2020) mengulas penerapan *CNN* dalam klasifikasi kendaraan secara real-time. Fokus utama penelitian ini adalah mengurangi insiden kecelakaan lalu lintas yang diakibatkan oleh pelanggaran aturan. Peneliti menyampaikan bahwa peningkatan signifikan jumlah kecelakaan lalu lintas selama beberapa tahun terakhir menjadi dorongan untuk mengembangkan sistem yang mampu mengidentifikasi kendaraan yang terlibat dalam kecelakaan tersebut. Guna mencapai tujuan yang ditetapkan, peneliti mengumpulkan dataset gambar kendaraan yang melibatkan empat jenis umum di negara tersebut: mobil, CNG, becak, dan sepeda. Ekstraksi fitur dan klasifikasi dijalankan menggunakan metode CNN. Model CNN, yang terkenal dalam pembelajaran mendalam, telah diterapkan luas dalam pengenalan objek serta klasifikasi gambar. Tantangan-tantangan beragam dihadapi dalam perjalanan penelitian ini, termasuk dalam proses pengumpulan dan pemrosesan dataset yang memakan waktu dan tenaga yang besar. Namun demikian, setelah melalui tahap pra-pemrosesan, model CNN dilatih dengan 2240 gambar untuk tahap pelatihan dan 560 gambar untuk tahap pengujian. Hasil eksperimen menunjukkan model CNN yang diusulkan berhasil mencapai tingkat akurasi sekitar 97% dalam tugas mengklasifikasikan jenis kendaraan.

Dong et al., (2015b) dalam artikelnya membahas tentang metode klasifikasi tipe kendaraan menggunakan *Convolutional Neural Network Semi-Supervised.*

Tujuannya adalah untuk dapat mengklasifikasikan tipe kendaraan dari gambar tampak depan kendaraan dengan akurat. Metode yang diusulkan adalah menggunakan *Convolutional Neural Network* yang terdiri dari dua tahap. Pada tahap pertama, *filter bank* pada lapisan konvolusi dipelajari secara *unsupervised* menggunakan *sparse Laplacian filter learning* (SLFL) dengan memanfaatkan data berlabel sedikit. Pada tahap kedua, *classifier softmax* pada lapisan *output* dilatih secara *supervised* menggunakan *multi-task learning* dengan data berlabel yang terbatas. Penulis membangun dataset *BIT-Vehicle* yang terdiri dari 9850 gambar tampak depan kendaraan beresolusi tinggi untuk mengevaluasi metode yang diusulkan. Selain itu, juga digunakan dataset publik lainnya. Hasil eksperimen pada dataset *BIT-Vehicle* dan dataset publik menunjukkan efektivitas dari metode yang diusulkan, dimana fitur yang dipelajari secara otomatis oleh *convolutional neural network* memperoleh akurasi sebesar 88.11%.

Penelitian yang terakhir dilakukan oleh (Irawanto et al., 2023) yang berjudul "*Deep Learning Based Car Detection System Using Convolutional Neural Network and Haar Cascade Classifier*" berfokus pada pemanfaatan teknologi deteksi objek kendaraan terkini untuk mengatur durasi lampu lalu lintas secara dinamis berdasarkan jumlah kendaraan yang terdeteksi pada setiap lintasan jalan. Peneliti menyatakan pertumbuhan jumlah kendaraan pribadi yang pesat telah menyebabkan kemacetan lalu lintas yang signifikan di berbagai wilayah. Untuk mengatasi tantangan ini, diperlukan solusi inovatif dalam mengelola lalu lintas, dan salah satu solusi yang diusulkan adalah penggunaan Sistem Lampu Lalu Lintas Pintar. Peneliti menyajikan pendekatan yang menggunakan dua metode deteksi kendaraan, yaitu

CNN dan *Haar Cascade Classifier* dengan harapan mendapatkan nilai akurasi yang cukup baik. Nilai akurasi rata-rata yang dihasilkan oleh penelitian ini sebesar 86%, namun metode ini menunjukkan tingkat presisi yang sangat tinggi, mencapai 100%, yang berarti hampir tidak ada kesalahan dalam mendeteksi kendaraan. *Recall* metode ini juga mencapai 86%, menunjukkan kemampuan untuk mendeteksi sebagian besar kendaraan yang melewati lintasan lalu lintas. Peneliti juga menyampaikan beberapa kelemahan dari penelitian ini, diantaranya sistem tidak dapat mengenali objek kendaraan yang berdekatan dan berada jauh dari pandangan kamera. Kemudian penelitian ini hanya fokus pada objek mobil pribadi saja, sehingga untuk penelitian selanjutnya diharapkan bisa memperluas jangkauannya dengan mencakup berbagai jenis kendaraan seperti truk, bus, dan mobil. Selain itu, penelitian berikutnya bisa difokuskan tidak hanya pada deteksi kendaraan, tetapi juga pada klasifikasi berbagai jenis kendaraan untuk memberikan informasi yang lebih komprehensif dan akurat dalam pengelolaan lalu lintas.

Dari tinjauan pustaka, dapat diidentifikasi beberapa kekurangan signifikan dalam penelitian sebelumnya. Meskipun teknik augmentasi data telah digunakan untuk meningkatkan keberagaman dataset (Butt et al., 2021b) , penerapannya masih dapat dieksplorasi lebih lanjut. Penelitian sebelumnya seringkali terbatas pada arsitektur CNN tertentu, padahal arsitektur lain seperti *Xception* dapat menawarkan performa yang lebih baik (Alghamdi et al., 2023). Selain itu, terdapat kebutuhan untuk pengembangan dataset yang relevan dengan kondisi lokal dan peningkatan efisiensi model (Chauhan et al., 2019). Irawanto et al. (2023) menggunakan kombinasi CNN dan Haar Cascade Classifier untuk mengatur durasi lampu lalu

lintas secara dinamis, namun sistem ini menunjukkan kelemahan dalam mendeteksi kendaraan yang berdekatan dan fokus hanya pada mobil pribadi.

Penelitian ini bertujuan untuk mengatasi kekurangan tersebut dengan menggabungkan *Haar Cascade Classifier* dalam mendeteksi fitur sederhana dan CNN, khususnya menggunakan arsitektur *Xception*, dalam mengenali pola yang lebih kompleks. Dengan memanfaatkan teknik augmentasi data dan dataset yang beragam, penelitian ini akan mengeksplorasi optimisasi model untuk meningkatkan akurasi dan efisiensi deteksi kendaraan, serta mengatasi tantangan dalam mendeteksi berbagai jenis kendaraan yang berdekatan. Hal ini diharapkan dapat menghasilkan sistem deteksi kendaraan yang lebih efisien dan akurat untuk mendukung Sistem Transportasi Cerdas.

2.2 Keaslian Penelitian

Table 2.1. Matriks literatur review dan posisi penelitian
Optimasi Performa Deteksi Kendaraan Dengan Menggabungkan Algoritma *Haar Cascade Classifier* Dan *Convolutional Neural Network* (CNN)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems	(Butt et al., 2021a) Complexity	mengusulkan sistem klasifikasi kendaraan berbasis CNN guna meningkatkan efektivitas Sistem Transportasi Cerdas	ResNet dengan 152 lapisan menunjukkan akurasi yang lebih baik dibandingkan dengan arsitektur lainnya, seperti VGG, GoogLeNet, AlexNet, dan Inception-v3.	Mempertimbangkan penggunaan teknik augmentasi data untuk meningkatkan variasi dataset.	Penelitian ini fokus pada peningkatan efektivitas Sistem Transportasi Cerdas melalui klasifikasi kendaraan menggunakan CNN. Sementara penelitian yang diajukan berkaitan dengan penggunaan metode Convolutional Neural Network (CNN) dan Haar Cascade Classifier dalam deteksi kendaraan dalam kondisi lalu lintas yang kompleks.
2	Vehicle Classification Using Deep Feature Fusion and Genetic Algorithms	(Gowri et al., 2022) Electronics (Switzerland)	Penelitian ini bertujuan untuk mencapai akurasi yang tinggi dalam mengklasifikasikan kendaraan berdasarkan fitur-fitur seperti	Model yang diusulkan, yang menggabungkan fitur dari CNN pre-trained VGG16 dan algoritma genetika, mampu mencapai akurasi yang sangat tinggi dalam klasifikasi	Eksperimen dapat dilakukan dengan menggunakan arsitektur CNN lainnya atau menggabungkan beberapa arsitektur untuk memperoleh hasil yang lebih baik.	penelitian yang diajukan berfokus pada deteksi kendaraan menggunakan kombinasi Convolutional Neural Network (CNN) dan Haar Cascade Classifier, sementara penelitian ini menggabungkan deep

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			warna, model, dan jenis kendaraan.	kendaraan. Model ini berhasil mencapai akurasi sebesar 99,78%, mengungguli penelitian sebelumnya.		learning dan algoritma genetika untuk klasifikasi kendaraan. Meskipun fokusnya berbeda, kedua penelitian menunjukkan bahwa kombinasi metode canggih dapat meningkatkan performa pengenalan kendaraan.
3	Vehicle Detection for Vision-Based Intelligent Transportation Systems Using Convolutional Neural Network Algorithm	(Khalifa et al., 2022) Journal of Advanced Transportation	Mengembangkan sistem klasifikasi kendaraan yang otomatis dan efektif menggunakan deep learning dan algoritma genetika. Sistem ini diharapkan dapat mengklasifikasikan kendaraan dengan akurasi tinggi dan waktu pelatihan yang lebih efisien.	Penggunaan deep feature fusion dari pre-trained CNN VGG16 dan algoritma genetika membantu meningkatkan performa model dalam hal akurasi.	Penelitian ini dapat dieksplorasi lebih lanjut dengan menggunakan teknik transfer learning untuk memanfaatkan model yang telah dilatih pada dataset yang lebih besar.	Fokus utama penelitian ini adalah pada penerapan arsitektur VGG16. Sementara itu, penelitian yang diajukan akan menitikberatkan pada pencarian arsitektur yang optimal.

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
4	CNN-Based Classification for Highly Similar Vehicle Model Using Multi-Task Learning	(Avianto et al., 2022) Journal of Imaging	Merancang program pendeteksi dan pengklasifikasi jenis kendaraan menggunakan metode CNN Deep Learning dengan algoritma YOLO.	program pendeteksi dan pengklasifikasi jenis kendaraan menggunakan metode CNN Deep Learning dengan algoritma YOLO memiliki tingkat akurasi yang baik. Pada pengujian berdasarkan	melakukan penambahan jumlah dan variasi data latih untuk meningkatkan akurasi program dalam mendeteksi kendaraan.	Memiliki tujuan yang sama dalam hal deteksi kendaraan dan penghitungan jumlah kendaraan secara real-time. namun memiliki perbedaan di bagian metode.
5	Embedded CNN based vehicle classification and counting in non-laned road traffic	(Chauhan et al., 2019) ACM International Conference Proceeding Series	Untuk mengembangkan metode pengklasifikasian dan penghitungan kendaraan dalam lalu lintas jalan non-laned menggunakan CNN berbasis objek deteksi.	model CNN yang dilatih dengan menggunakan dataset dari jalan-jalan di Delhi-NCR dapat mencapai akurasi hingga 75% dalam mengklasifikasikan kendaraan.	memperbaiki akurasi model CNN dengan memperhatikan perbedaan dalam kendaraan dan lalu lintas di wilayah berkembang seperti Delhi-NCR	Berfokus pada pengembangan metode deteksi dan klasifikasi kendaraan menggunakan Convolutional Neural Network (CNN). Namun, penelitian ini difokuskan pada deteksi kendaraan dalam lalu lintas jalan non-laned, sementara penelitian yang diajukan berkaitan dengan optimasi performa deteksi kendaraan dengan menggabungkan algoritma CNN dan Haar Cascade Classifier.

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
6	Real-Time Vehicle Classification Using CNN	(Jahan et al., 2020) 2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020	Tujuan dari penelitian ini adalah untuk mengembangkan sebuah sistem klasifikasi kendaraan menggunakan Convolutional Neural Network (CNN)	Model CNN yang diusulkan berhasil mencapai akurasi sekitar 97% dalam klasifikasi kendaraan. Hal ini menunjukkan bahwa CNN adalah pendekatan yang efektif dalam mengklasifikasikan jenis kendaraan	Meskipun penelitian ini menggunakan 2800 gambar, jumlah ini masih terbatas dalam konteks pembelajaran mesin. Dataset yang lebih besar dan lebih beragam dapat membantu meningkatkan performa dan generalisasi model.	Berfokus pada pengembangan metode klasifikasi kendaraan menggunakan Convolutional Neural Network (CNN). Namun, dalam penelitian ini mengkombinasikan CNN dengan Haar Cascade Classifier yang diharapkan dapat mengoptimasikan performa deteksi dan klasifikasi kendaraan
7	Vehicle Type Classification Using a Semisupervised Convolutional Neural Network	(Dong et al., 2015b) Ieee Transactions On Intelligent Transportation Systems	Mengusulkan metode klasifikasi tipe kendaraan menggunakan *Convolutional Neural Network* Semisupervised learning dari gambar tampak depan kendaraan.	Hasil eksperimen menunjukkan metode yang diusulkan mencapai akurasi yang efektif pada dataset BIT-Vehicle dengan nilai sebesar 88.11%	Penelitian selanjutnya dapat mengeksplorasi pendekatan yang mampu memanfaatkan berbagai sudut pandang untuk klasifikasi yang lebih akurat.	Fokus pada pengembangan metode klasifikasi kendaraan menggunakan Convolutional Neural Network (CNN). Namun, dalam penelitian ini mengkombinasikan CNN dengan Haar Cascade Classifier yang diharapkan dapat mengoptimasikan performa deteksi dan klasifikasi kendaraan

Tabel 2.1. (Lanjutan)

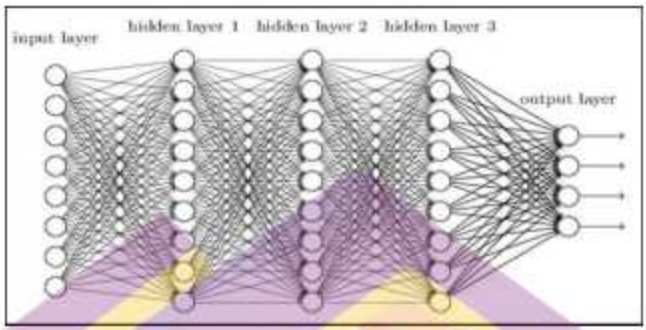
No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
8	Deep Learning Based Car Detection System Using Convolutional Neural Network and Haar Cascade Classifier	(Irawanto et al., 2023) 2023 6th International Conference on Information and Communications Technology (ICOIACT)	Penelitian ini bertujuan untuk menghasilkan pemahaman tentang efektivitas metode klasifikasi *Haar Cascade* dan CNN dalam mendeteksi kendaraan.	Dari hasil eksperimen menunjukan bahwa rata rata nilai akurasi yang diperoleh sebesar 86%.	Saran untuk penelitian selanjutnya adalah dengan mengubah objek dari deteksi mobil menjadi deteksi kendaraan dengan menambahkan proses klasifikasi. Kemudian menambahkan proses *transfer learning* agar dapat menjadi fokus penelitian untuk meningkatkan akurasi.	Penelitian ini dan penelitian yang akan diajukan memiliki kesamaan dalam hal algoritma yang digunakan namun memiliki perbedaan pada fokus penelitiannya. Penelitian ini fokus pada deteksi mobil sedangkan penelitian yang diajukan fokus pada deteksi dan klasifikasi kendaraan dengan menerapkan metode *transfer learning*.

2.3 Landasan Teori

2.3.1 Deep Learning

Deep Learning adalah cabang dari machine learning yang menggunakan jaringan syaraf tiruan untuk menangani dataset besar. Dalam hal ini, Multi Layer Perceptron (MLP) atau jaringan dengan banyak lapisan digunakan untuk memproses informasi non-linier guna ekstraksi fitur, pengenalan pola, dan klasifikasi. Meskipun metode *machine learning* memiliki kelemahan dalam hal kecepatan dan akurasi, pendekatan deep learning mengatasi masalah ini dengan memanfaatkan konsep hierarkis (Nurhikmat, 2018). Konsep ini memungkinkan komputer memahami konsep kompleks melalui gabungan konsep yang lebih sederhana. Arsitektur *deep learning* terdiri dari komponen-komponen seperti:

a. **Penyimpanan Data**: Tempat di mana seluruh citra digital pelatihan disimpan untuk diolah.
b. **Layer**: Komponen yang bertugas memproses citra digital yang tersimpan dalam penyimpanan data. Hasil proses ini digunakan untuk membandingkan dan akhirnya memberikan kesimpulan mengenai isi citra digital tersebut.



Gambar 2.1. Arsitektur *Deep Learning*

Sederhananya, *deep learning* merupakan teknik di dalam machine learning yang menggunakan jaringan syaraf tiruan untuk memproses data kompleks melalui serangkaian lapisan, memungkinkan pemahaman yang lebih baik tentang informasi yang terkandung dalam data tersebut.

2.3.2 Haar Cascade Classifiers

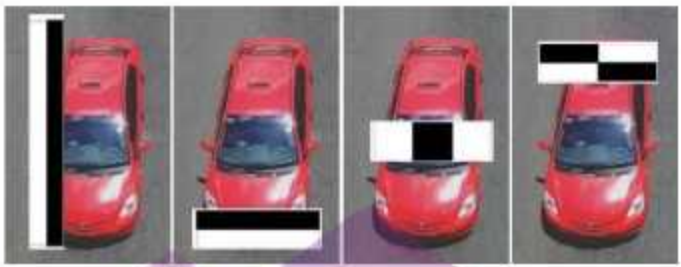
Haar Cascade Classifier merupakan fungsi persegi (rectangular features) dalam citra atau gambar yang memberikan petunjuk khusus. Fitur Haar-like bekerja dengan prinsip mengenali objek berdasarkan nilai fitur di dalam persegi, bukan nilai piksel keseluruhan dari gambar objek (Viola & Jones, 2001). Keuntungan utama dari metode ini adalah kecepatan komputasi yang sangat tinggi, karena hanya mengambil jumlah piksel dalam persegi, tidak perlu mempertimbangkan setiap nilai piksel di seluruh citra (Syarif, 2015). Pendekatan Viola-Jones, atau dikenal sebagai metode Haar Cascade Classifiers, memiliki akurasi yang tinggi, mencapai sekitar 93,7%, dengan kecepatan 15 kali lebih cepat dibandingkan dengan detektor

Rowley Baluja-Kanade, dan sekitar 600 kali lebih cepat dibandingkan dengan detektor Schneiderman-Kanade (Triatmoko et al., 2014). Metode ini dikembangkan oleh Paul Viola dan Michael Jones pada tahun 2001 dengan menggabungkan empat komponen utama: *Haar Like Feature*, *Integral Image*, *Adaboost learning*, dan *Cascade classifier*.

a. *Haar like Features*

Haar Feature didasarkan pada konsep Wavelet Haar, yang diperkenalkan oleh (Viola & Jones, 2001). *Wavelet Haar* adalah gelombang bujur sangkar tunggal dengan satu interval tinggi dan satu interval rendah. Dalam dua dimensi, terdapat satu bagian terang dan satu bagian gelap, yang selanjutnya dikombinasikan dalam berbagai bentuk kotak untuk meningkatkan pendeteksian objek visual. Setiap *Haar-like feature* terdiri dari gabungan kotak-kotak hitam dan putih (Mahmudi & Rusda, 2014).

Fitur Haar diidentifikasi dengan mengurangi rata-rata piksel di area gelap dari rata-rata piksel di area terang. Jika perbedaan nilai ini melebihi ambang batas tertentu, fitur tersebut dianggap ada. Nilai dari *Haar-like feature* dihitung sebagai selisih antara jumlah piksel *gray* level dalam kotak hitam dan kotak putih. Proses ini dapat dilakukan dengan cepat menggunakan "*integral image*" (Syarif, 2015). Gambar 2.4 menunjukkan proses pendeteksian mobil menggunakan Haar Like Features.



Gambar 2.2. Proses Deteksi Mobil Dengan *Haar-Like Features*

b. Integral Image

Integral Image adalah sebuah teknik yang memungkinkan perhitungan nilai fitur dengan cepat melalui transformasi nilai setiap piksel menjadi representasi gambar baru (Triatmoko et al., 2014). Citra integral pada koordinat x,y dapat dihitung menggunakan persamaan 2.1 (Viola & Jones, 2001).

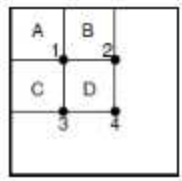
$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y') \quad \text{---------- (2.1)}$$

Keterangan:

$ii(x, y)$ = Citra integral pada lokasi x,y

$i(x', y')$ = Nilai piksel pada citra asli

Gambar 2.3 menunjukkan contoh perhitungan nilai fitur. Jika nilai integral image titik 1 adalah A, titik 2 adalah A+B, titik 3 adalah A+C, dan di titik 4 adalah A+B+C+D, maka jumlah piksel di daerah D dapat diketahui dengan cara 4 + 1-(2+ 3) (Triatmoko, Pramono, & Dachlan, 2014).



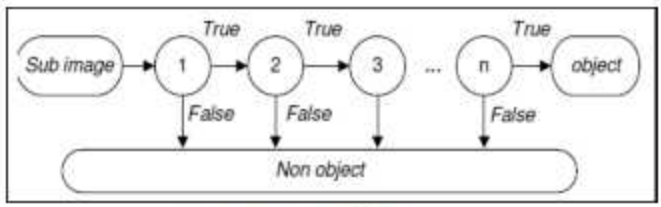
Gambar 2.3. Perhitungan Nilai Fitur

c. *Algoritma Adaboost learning*

Algoritma Adaboost learning, digunakan untuk meningkatkan kinerja klasifikasi dengan pembelajaran sederhana untuk menggabungkan banyak classifier lemah menjadi satu classifier kuat (Hadriansa & Kristian, 2015). *Classifier* lemah adalah suatu jawaban benar dengan tingkat kebenaran yang kurang akurat.

d. *Cascade classifier*

Cascade classifier adalah sebuah rantai *stage classifier*, dimana setiap *stage classifier* digunakan untuk mendeteksi apakah di dalam image sub window terdapat objek yang ingin dideteksi *(object of interest)* (Mahmudi & Rusda, 2014). Sedangkan menurut (Hadriansa & Kristian, 2015), *Cascade classifier* adalah sebuah metode untuk mengkombinasikan *classifier* yang kompleks dalam sebuah struktur bertingkat yang dapat meningkatkan kecepatan pendeteksian objek dengan memfokuskan pada daerah citra yang berpeluang saja. Struktur cascade classifier disajikan pada Gambar 2.4.



Gambar 2.4. *Cascade classifier*

Gambar 2.4 menjelaskan proses penyeleksian keberadaan objek. Di asumsikan suatu *sub image* di evaluasi oleh *classifier* pertama dan berhasil melewati *classifier* tersebut, hal ini mengindikasikan *sub image* berpotensi terkandung objek dan dilanjutkan pada *classifier* ke dua sampai dengan ke-n, jika berhasil melewati keseluruhan *classifier*, maka disimpulkan terdapat objek yang dideteksi. Jika tidak, proses evaluasi tidak dilanjutkan ke *classifier* berikutnya dan disimpulkan tidak terdapat objek (Hadriansa & Kristian, 2015).

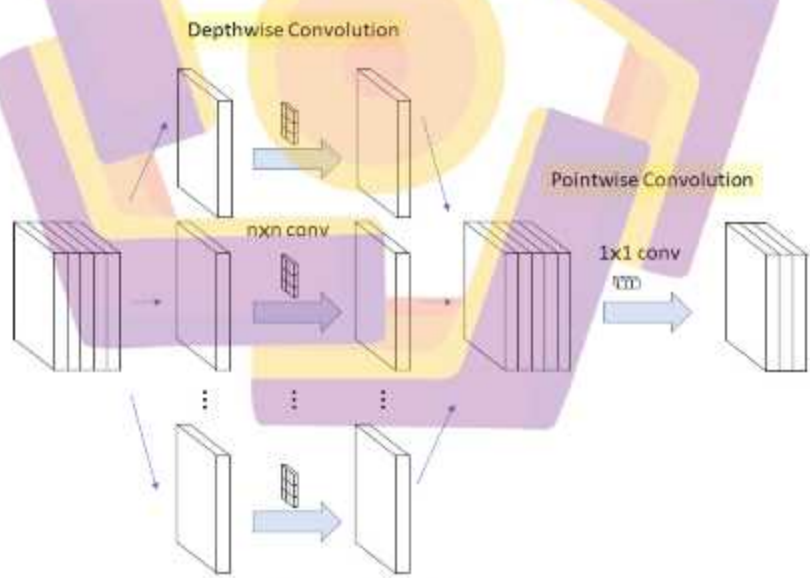
2.3.3 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan *supervised feed-forward* networks (jaringan umpan maju) yang membuktikan kinerja yang sangat signifikan pada aplikasi objek bersekala besar. Struktur dasar dari CNN terinspirasi oleh korteks visual utama dalam otak manusia, yang bertanggung jawab atas pemrosesan informasi visual (Butt et al., 2021a). CNN merupakan salah satu model algoritma dalam ranah *deep learning* yang mereplikasi prinsip kerja otak manusia. Arsitektur CNN meliputi lapisan input, beberapa lapisan konvolusi, lapisan padat yang sering disebut sebagai fully-connected layers (lapisan yang sepenuhnya terhubung), dan lapisan output. CNN berperan sebagai algoritma deep learning yang menerima

gambar sebagai input, menetapkan bobot dan bias pada objek khusus dalam gambar, dan memiliki kemampuan untuk membedakan serta mengklasifikasikan berbagai gambar satu sama lain (Vyshnavi et al., 2022) .

2.3.4 Xception

Arsitektur *Xception* diperkenalkan oleh Chollet, (2017) dalam makalahnya "*Xception: Deep Learning with Depthwise Separable Convolutions*" pada tahun 2017. *Xception*, yang merupakan singkatan dari "*Extreme Inception*," adalah pengembangan lebih lanjut dari arsitektur *Inception*. Model ini didasarkan pada ide bahwa kedalaman jaringan neural bisa ditingkatkan tanpa menambah jumlah parameter yang besar, dengan menggunakan operasi convolusi yang lebih efisien.



Gambar 2.5. Arsitektur Xception

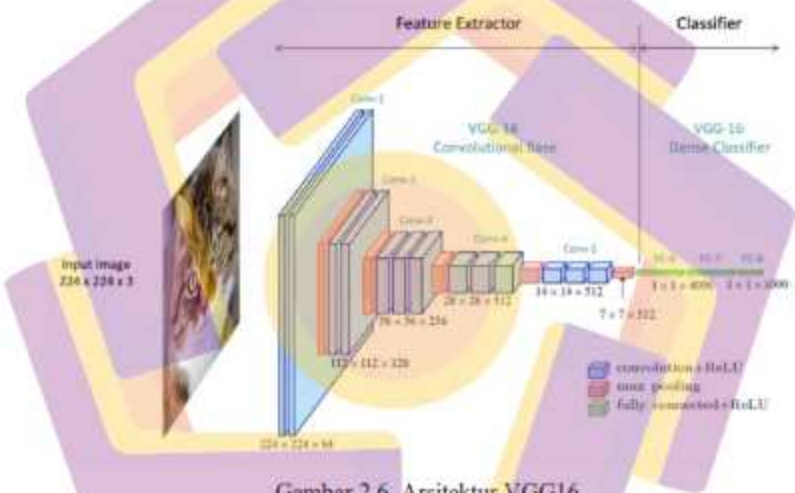
Xception menggunakan *depthwise separable convolutions*, yang memisahkan proses convolusi menjadi dua langkah: *depthwise convolution* dan *pointwise convolution*. Langkah pertama, *depthwise convolution*, melakukan convolusi pada setiap channel input secara independen. Langkah kedua, *pointwise convolution*, menggabungkan hasil dari *depthwise convolution* dengan menggunakan kernel 1x1 untuk menghasilkan *feature map* yang baru. Dengan metode ini, *Xception* berhasil mengurangi jumlah parameter dan meningkatkan efisiensi komputasi tanpa mengorbankan performa model (A. Khan et al., 2020).

2.3.5 VGG16

Arsitektur VGG16 diperkenalkan oleh Simonyan & Zisserman, (2014) dalam makalah mereka "*Very Deep Convolutional Network for Large Scale Image Recognition*". VGG16 adalah salah satu arsitektur CNN yang terkenal karena kesederhanaannya dan penggunaan *kernel* berukuran kecil. Model ini menggunakan *kernel* 3x3 dengan *Max Pooling* serta *stride* bernilai 1 untuk ekstraksi fitur. Pada bagian akhir arsitekturnya, terdapat tiga lapisan *fully connected*. Dengan ukuran kernel tersebut, kedalaman *neural network* dapat ditingkatkan sehingga menghasilkan akurasi yang lebih baik dibandingkan dengan arsitektur lainnya. Setiap lapisan konvolusi diikuti oleh fungsi aktivasi ReLU untuk mengurangi risiko *overfitting* (S. Khan et al., 2018).

Arsitektur *VGG16* menerima input berupa citra dengan dimensi 224x224 dan 3 channel (R, G, dan B). Citra ini akan dimasukkan ke dua *convolutional layer* dengan kernel 3x3, yang kemudian diikuti oleh fungsi aktivasi *ReLU*. Proses

konvolusi ini menggunakan *stride* bernilai 1 dan padding bernilai 1, sehingga ukuran *feature map* yang dihasilkan sama dengan dimensi citra input. *Feature map* ini kemudian dimasukkan ke *Max Pooling layer* dengan ukuran 2x2 dan *stride* bernilai 2, sehingga dimensi *feature map* berkurang menjadi 112x112x64. Proses ini diulangi beberapa kali, menghasilkan *feature map* akhir dengan dimensi 7x7x512 seperti pada gambar 2. 6 (Manju D et al., 2021).



Gambar 2.6. Arsitektur VGG16

Arsitektur VGG16 sangat baik digunakan untuk masalah seperti pengenalan dan klasifikasi citra, serta untuk deteksi dan lokalisasi objek dalam citra. Keunggulan VGG16 terletak pada kesederhanaan arsitekturnya dan akurasinya yang tinggi.

2.3.6 Convolutional Layer

Convolutional Layer merupakan lapisan terpenting dalam CNN yang digunakan untuk mendeteksi fitur-fitur seperti tepi, garis, warna, dan elemen visual lainnya. Semakin banyak *kernel* yang digunakan pada CNN, semakin banyak fitur yang dapat dideteksi (Yamashita et al., 2018). *Kernel* sendiri adalah objek berbentuk persegi yang digunakan untuk memindai sebuah citra. Pada *convolutional layer*, terdapat beberapa parameter yang menentukan keluaran dari lapisan ini, di antaranya:

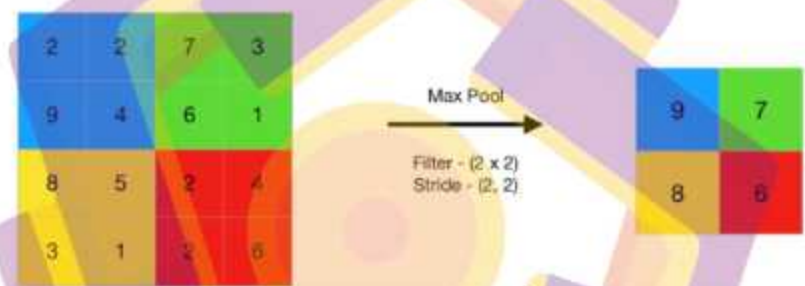
- Jumlah kernel akan memengaruhi banyaknya *feature map* yang dihasilkan.
- Ukuran kernel akan memengaruhi dimensi keluaran yang dihasilkan.
- *Stride* adalah langkah pergeseran kernel secara *horizontal* pada proses konvolusi. Semakin kecil nilai *stride*, semakin banyak informasi yang didapatkan.
- *Padding* adalah piksel bernilai nol yang ditambahkan pada sisi-sisi piksel masukan untuk memanipulasi dimensi keluaran.
- Fungsi aktivasi digunakan untuk menghasilkan keluaran yang nonlinier.

2.3.7 Pooling Layer

Sama seperti *Convolutional Layer*, *pooling layer* adalah lapisan yang berfungsi untuk mengurangi dimensi data, sehingga mengurangi daya komputasi yang diperlukan dalam pemrosesan data. Selain itu, *pooling layer* juga membantu dalam mengekstrak fitur-fitur dominan agar proses pelatihan menjadi lebih efektif.

Parameter utama pada *pooling layer* mencakup luas area yang diproses (spatial extent) dan langkah perpindahan area pemampatan (stride) (Yamashita et al., 2018).

Operasi *Pooling* yang paling umum digunakan adalah *Max Pooling*. *Max Pooling* mengekstrak potongan-potongan *feature map input* dan menghasilkan nilai maksimum dari setiap potongan, sekaligus membuang semua nilai lainnya. Biasanya, *Max Pooling* menggunakan filter berukuran 2 × 2 dengan langkah perpindahan (*stride*) sebesar 2 (S. Khan et al., 2018).



Gambar 2.7. Proses *Max Pooling* Pada *Feature map* 4 x 4

Gambar 2. 7 menunjukkan proses *Max Pooling* pada sebuah *feature map* dengan ukuran 4 × 4 menggunakan *filter* berukuran 2 × 2 dan *stride* sebesar 2. Pada gambar sebelah kiri, terdapat sebuah *feature map* dengan nilai-nilai piksel yang berbeda. Filter 2 × 2 diterapkan pada *feature map* ini, dan langkah perpindahan sebesar 2 berarti filter tersebut melompat dua langkah baik secara *horizontal* maupun *vertikal* untuk mengekstrak nilai maksimum dari setiap potongan 2 × 2. Prosesnya adalah sebagai berikut:

1. Filter 2 × 2 pertama kali diterapkan pada bagian kiri atas dari *feature map* yang mencakup nilai-nilai (2, 2, 9, 4). Nilai maksimum dari potongan ini

adalah 9, sehingga 9 ditetapkan pada posisi kiri atas dari *feature map* yang baru (gambar kanan).

2. Filter kemudian berpindah dua langkah ke kanan dan mencakup nilai-nilai (7, 3, 6, 1). Nilai maksimum dari potongan ini adalah 7, sehingga 7 ditetapkan pada posisi kanan atas dari *feature map* yang baru.
3. Filter kemudian kembali ke posisi kiri bawah dari *feature map* yang mencakup nilai-nilai (8, 5, 3, 1). Nilai maksimum dari potongan ini adalah 8, sehingga 8 ditetapkan pada posisi kiri bawah dari *feature map* yang baru.
4. Terakhir, filter berpindah dua langkah ke kanan dari posisi sebelumnya dan mencakup nilai-nilai (2, 4, 2, 6). Nilai maksimum dari potongan ini adalah 6, sehingga 6 ditetapkan pada posisi kanan bawah dari *feature map* yang baru.

2.3.8 Confusion Matrix

Confusion matrix merupakan sebuah alat untuk mengevaluasi kinerja algoritma machine learning yang berisi informasi tentang klasifikasi dan prediksi actual. Ada empat indikator yang diukur di dalamnya: *accuracy, precision, recall* dan *F1-Score* (Prastyo et al., 2020). Pada evaluasi klasifikasi ada 4 kemungkinan yang bisa terjadi dari hasil klasifikasi satu data. Bila data positif dan prediksi positif akan dihitung sebagai *true positive* dan jika data positif dan prediksi negatif maka akan dihitung sebagai *false negative*. Pada data negatif jika prediksi negatif makan

akan dihitung sebagai *true negative* sedangkan jika prediksi positif maka akan dihitung sebagai *false positive* (Suryati et al., 2023).

Table 2.2. *Confusion Matrix*

Actual	Prediction	
	Positif	Negatif
Positif	True Positive (TP)	True Negative (TN)
Negatif	False Positive (FP)	False Negative (FN)

a) *Precision*

Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban *precision*. Dihitung dengan rumus berikut

$$Precission = \frac{TP}{(TP + FP)}$$

b) *Recall*

Recall merupakan perhitungan keakuratan prediksi yang digunakan sebagai ukuran tingkat keberhasilan sistem dalam menemukan sebuah informasi. Dapat dihitung melalui rumus berikut

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

c) *Accuracy*

Accuracy ialah tingkat kedekatan antara nilai prediksi dengan nilai asli atau *actual*. Dihitung dalam rumus berikut

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

d) F1-*score*

F1-*score* memperhitungkan kedua *precision* dan *recall*, sehingga lebih sensitive terhadap kesalahan dalam memprediksi kelas minoritas. Dihitung dalam rumus dibawah

$$F1 - Measure = 2 \times \frac{precission \times recall}{precission + recall}$$

BAB III

METODE PENELITIAN

3.1 Jenis, Sifat dan Pendekatan Penelitian

Penelitian ini merupakan penelitian eksperimental yang bertujuan untuk menguji pengaruh dari penggunaan arsitektur *Xception* dan VGG16 dalam CNN serta kombinasi antara *Haar Cascade Classifier* dan CNN terhadap akurasi dan performa deteksi dan klasifikasi kendaraan dalam berbagai kondisi lalu lintas. Dalam penelitian ini, kami menggunakan dataset *BIT-Vehicle* yang telah diolah menjadi data training dan data testing. Metode eksperimental ini memungkinkan kami untuk mengontrol variabel-variabel yang berpengaruh pada hasil dan menganalisis dampak dari penggunaan berbagai teknik dalam pengenalan dan deteksi kendaraan.

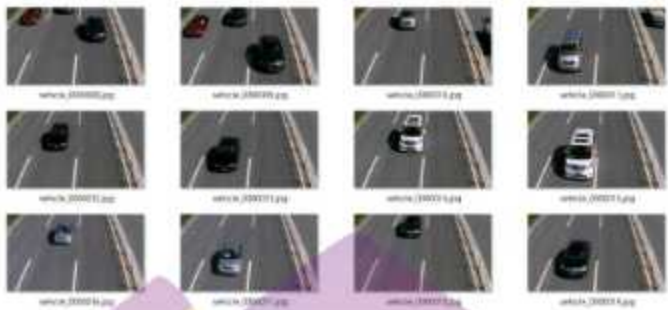
Penelitian ini memiliki sifat kuantitatif, dimana data yang diperoleh berupa angka-angka yang dapat diukur dan dianalisis secara statistik. Dalam penelitian ini, kami akan mengukur performa deteksi kendaraan dalam hal akurasi, presisi, confusion matrix, dan recall. Data-data ini akan diolah dan dianalisis menggunakan metode statistik untuk mendapatkan informasi yang dapat mendukung atau menolak hipotesis yang telah diajukan.

Penelitian ini akan menggunakan pendekatan kombinasi antara pendekatan eksperimental dan analisis data. Pendekatan eksperimental digunakan dalam menguji pengaruh dari arsitektur *Xception* dan kombinasi dengan *Haar Cascade Classifier* terhadap akurasi deteksi kendaraan. Kami akan mengukur performa

deteksi pada kondisi lalu lintas yang berbeda dan membandingkannya dengan hasil pengenalan menggunakan metode individu. Pendekatan analisis data digunakan untuk memproses data yang diperoleh dari eksperimen, termasuk menghitung akurasi, presisi, dan recall serta menganalisis perbedaan performa antara metode yang berbeda. Dalam pendekatan ini, data empiris akan dianalisis untuk mendukung temuan dan kesimpulan yang dihasilkan dari penelitian ini.

3.2 Metode Pengumpulan Data

Dataset yang digunakan dalam penelitian ini ialah *BIT-Vehicle* yang terdiri dari sekumpulan gambar kendaraan yang diambil menggunakan dua kamera pada berbagai waktu dan lokasi. Ukuran gambar bervariasi antara 1600x1200 dan 1920x1080 piksel. Dataset ini mencakup perubahan kondisi pencahayaan, skala, warna permukaan kendaraan, serta sudut pandang yang berbeda. Beberapa gambar mungkin tidak menampilkan bagian atas atau bawah kendaraan karena keterlambatan dalam pengambilan gambar dan ukuran kendaraan. Setiap gambar bisa mengandung satu atau dua kendaraan, dengan lokasi masing-masing kendaraan telah diannotasi sebelumnya. Kendaraan-kendaraan dalam dataset ini terbagi ke dalam enam kategori: Bus, Mikrobus, Minivan, Sedan, SUV, dan Truk, dengan jumlah kendaraan per kategori secara berurutan adalah 558, 883, 476, 5.922, 1.392, dan 822. Ini akan menjadi sumber berharga dalam pengembangan dan penelitian terkait pengenalan serta deteksi kendaraan dalam berbagai kondisi.



Gambar 3.1. Dataset Kendaraan

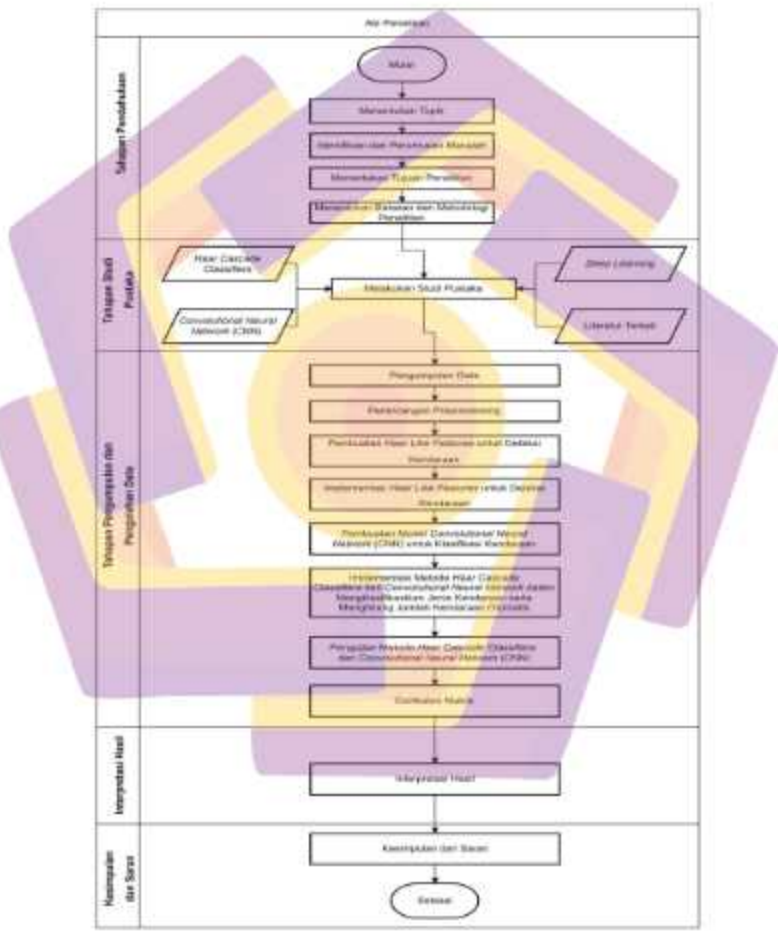
3.3 Metode Analisis Data

Metode analisis data dalam penelitian ini mencakup beberapa langkah kunci. Pertama, data dari hasil deteksi kendaraan menggunakan *Haar Cascade Classifiers* dan CNN akan diolah dan divisualisasikan untuk memahami hasil deteksi dan distribusi jenis kendaraan seperti bus, mobil, dan truk. Analisis statistik deskriptif akan digunakan untuk merumuskan informasi dasar seperti jumlah total kendaraan yang terdeteksi berdasarkan jenisnya. Selanjutnya, analisis perbandingan hasil deteksi dari kedua metode tersebut akan dilakukan untuk mengevaluasi keakuratan dan performa sistem, serta kemampuan sistem dalam mengklasifikasikan jenis kendaraan secara akurat. Pendekatan ini akan membantu memahami efisiensi dan keandalan sistem deteksi kendaraan yang diusulkan dalam penelitian ini.

3.4 Alur Penelitian

Dalam menjalankan penelitian ini, prosesnya dapat dibagi menjadi beberapa tahapan yang saling berkaitan dan berkesinambungan. Alur penelitian ini dirancang

untuk memastikan bahwa setiap langkah memiliki tujuan dan kontribusi yang jelas terhadap keseluruhan penelitian. Tahapan-tahapan tersebut meliputi tahapan pendahuluan, tahapan studi pustaka, tahapan pengumpulan dan pengelolahan data, interpretasi hasil, serta kesimpulan dan saran.



Gambar 3.2. Alur Penelitian

3.4.1 Tahapan Pendahuluan

Pada tahap ini, penelitian dimulai dengan menentukan topik penelitian, merumuskan permasalahan, tujuan penelitian, dan menentukan Batasan dan metodologi penelitian. Penelitian juga akan mempersiapkan kerangka kerja dan metode penelitian yang akan digunakan. Bagian ini bertujuan untuk memberikan gambaran umum tentang tujuan dan arah penelitian

3.4.2 Tahapan Studi Pustaka

Pada tahap ini, penelitian akan melakukan tinjauan pustaka untuk mengumpulkan informasi terkait dengan topik penelitian. Ini mencakup kajian literatur yang mencakup teori-teori yang relevan, temuan penelitian terdahulu, dan konsep-konsep yang berkaitan. Bagian ini digunakan sebagai dasar untuk memahami konteks penelitian dan merumuskan kerangka teoritis.

3.4.3 Tahapan Pengumpulan Data dan Pengelolahan Data

Tahap ini melibatkan pengumpulan data yang diperlukan untuk penelitian. Data dapat berupa gambar, video, atau informasi lain yang relevan dengan penelitian yang dilakukan. Seperti yang telah dipaparkan pada bagian "Metode Pengumpulan Data", penelitian ini menggunakan data publik yang sudah digunakan pada penelitian-penelitian sebelumnya. Setelah data terkumpul, dilakukan proses pengolahan data yang melibatkan algoritma, rute, pemodelan-pemodelan, desain, yang terkait dengan aspek perancangan sistem. Berikut penjelasan dari tahapan – tahapan yang dilakukan:

- **Pembuatan *Haar Like Features* untuk Deteksi Kendaraan**

  Pada tahapan pengumpulan dan pengelolahan data dalam penelitian ini dimulai dengan langkah pembuatan *Haar-like features* untuk deteksi kendaraan. *Haar-like features* adalah pola piksel yang menjadi dasar dalam metode *Haar Cascade Classifiers*. Tahap awal melibatkan proses annotasi untuk mengonversi data, termasuk koordinat bounding box (BBox) dan label kelas kendaraan, yang awalnya tersimpan dalam format *.mat, ke dalam format *.txt. Hasil dari annotasi tersebut digunakan untuk ekstraksi *Haar-like features* dari citra kendaraan. Proses ekstraksi ini melibatkan pembentukan pola-pola piksel yang memungkinkan deteksi objek tertentu pada citra. Setelah proses ekstraksi selesai, langkah berikutnya adalah melatih data ekstraksi tersebut. Tujuannya adalah untuk mempersiapkan data agar dapat digunakan dalam implementasi filter *Haar Cascade Classifier*. Proses pelatihan ini melibatkan pembelajaran dari contoh-contoh positif (gambar kendaraan) dan contoh negatif (gambar bukan kendaraan).

- **Implementasi *Haar Like Features* untuk Deteksi Kendaraan**

  Setelah mendapatkan *Haar-like features*, langkah selanjutnya adalah implementasi filter *Haar Cascade Classifier* untuk deteksi kendaraan pada video CCTV yang didapatkan dari laman *Youtube*. Proses ini bertujuan untuk mengukur performa dari

filter *Haar Cascade Classifier* dalam mendeteksi kendaraan pada video tersebut.

- **Pembuatan Model *Convolutional Neural Network* (CNN) Untuk Klasifikasi Kendaraan**

  Tahap selanjutnya ialah membuat model *CNN* yang bertujuan untuk mengklasifikasikan gambar dan memprediksi apakah gambar tersebut menggambarkan kendaraan atau bukan. Pada arsitektur CNN, gambar masukan akan dianalisis atribut atau fiturnya menggunakan konvolusi 3 filter, diikuti oleh proses *max pooling* untuk menghasilkan gambar dengan resolusi lebih rendah. Kemudian, hasil *max pooling* dimasukkan ke dalam *hidden layer Multi Layer Perceptron* (MLP). Model ini menggunakan beberapa lapisan konvolusi dan *max pooling* sebelum hasilnya dimasukkan ke dalam *hidden layer* MLP. Proses ini dapat diulangi beberapa kali untuk meningkatkan performa model.

- **Implementasi Metode *Haar Cascade Classifiers* dan *Convolutional Neural Network* dalam Mengklasifikasikan serta Menghitung Jumlah kendaraan Otomatis**

  Langkah selanjutnya melibatkan implementasi metode *Haar Cascade Classifiers* dan *CNN* untuk melakukan proses klasifikasi terhadap jenis kendaraan yang terdeteksi secara otomatis. *Metode Haar Cascade Classifiers* digunakan untuk mendeteksi objek

kendaraan pada video CCTV secara real-time. Setelah proses deteksi dilakukan, langkah berikutnya melibatkan *CNN* yang bertugas untuk mengklasifikasikan jenis kendaraan yang telah berhasil dideteksi sebelumnya.

- **Uji Coba**

  Tahapan uji coba dilakukan untuk mengevaluasi keberhasilan pengembangan metode ini. Proses pengujian dilakukan menggunakan data uji yang mencakup berbagai jenis kendaraan yang terekam dalam kondisi lalu lintas yang berbeda. Data uji yang digunakan berasal dari gambar-gambar yang diambil dari video CCTV lalu lintas jalan raya, yang ditemukan melalui pencarian berbagai kata kunci seperti "*traffic CCTV*," "*roads*," "*cars*," dan "*highway*" di platform *YouTube*. Dalam proses uji coba, tingkat keberhasilan metode dievaluasi berdasarkan sejumlah parameter, termasuk tingkat akurasi, presisi, dan sensitivitas. Setiap parameter ini dihitung berdasarkan performa metode pada setiap data uji yang digunakan. Selain itu, pengujian dilakukan dengan menggunakan berbagai arsitektur CNN yang berbeda untuk menentukan arsitektur yang paling optimal dalam mendeteksi dan mengklasifikasikan jenis kendaraan. Hal ini bertujuan untuk mengidentifikasi metode yang paling efektif dalam mengatasi permasalahan deteksi dan klasifikasi kendaraan dalam berbagai kondisi lalu lintas.

Untuk menghitung tingkat akurasi (*accuracy*), tingkat presisi (*precision*), dan tingkat sensitivitas (*recall*) suatu metode, pemahaman mengenai variabel yang akan dievaluasi sangat penting. Terdapat empat kemungkinan variabel hasil deteksi yang harus diperhatikan, yaitu nilai *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Penjelasan terperinci mengenai variabel-variabel ini akan diuraikan pada Tabel 3.1.

Table 3.1. Parameter Uji Coba

Actual Class	*Predicted Class*		
		Class Yes	*Class No*
	Class Yes	*True Positive* (TP)	*False Negative* (FN)
	Class No	*False Positive* (FP)	*True Negative* (TN)

a. *True Positive* (TP) atau Benar Positif adalah kasus dimana objek diprediksi positif oleh sistem dan prediksi sistem memang benar adanya.

b. *True Negative* (TN) atau Benar Negatif adalah kasus dimana objek diprediksi negatif oleh sistem dan prediksi sistem memang benar adanya.

c. *False Positive* (FP) atau Salah Positif adalah kasus dimana objek diprediksi positif oleh sistem tetapi kenyataannya adalah objek bernilai negatif.

d. *False Negative* (FN) atau Salah Negatif adalah kasus dimana objek diprediksi negatif oleh sistem tetapi kenyataannya adalah objek bernilai positif.

Pengertian dan persamaan masing-masing parameter uji coba seperti accuracy, precision, dan recall akan dijelaskan berdasarkan variabel pada Tabel 3.1.

a. Accuracy

Accuracy atau akurasi merupakan rasio prediksi benar positif dan benar negatif (TP + TN) dibandingkan dengan keseluruhan data (TP + FP + TN + FN). Dapat dihitung melalui rumus berikut.

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

b. Precision

Precision atau presisi merupakan rasio prediksi benar positif (TP) dibandingkan dengan keseluruhan hasil yang diprediksi positif (TP + FP). Dapat dihitung melalui rumus berikut.

$$Precission = \frac{TP}{(TP + FP)}$$

c. Recall

Recall atau sensitivitas merupakan rasio prediksi benar positif (TP) dibandingkan dengan keseluruhan data yang bernilai positif pada data sebenarnya atau yang berada pada actual class-yes

(TP + FN) seperti pada Tabel 3.1. Dapat dihitung melalui rumus berikut.

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

3.4.4 Tahapan Interpretasi Hasil

Setelah data diproses, langkah selanjutnya adalah melakukan analisis hasil. Ini mencakup penggunaan metode analisis yang sesuai untuk menjawab pertanyaan penelitian dan menginterpretasikan hasilnya. Bagian ini akan menjelaskan temuan-temuan yang didapatkan dari data dan bagaimana temuan tersebut berkaitan dengan tujuan penelitian.

3.4.5 Tahapan Kesimpulan dan Saran

Pada tahap terakhir, saatnya melakukan analisis hasil dari uji coba metode yang telah diterapkan, termasuk perbandingan antara berbagai arsitektur yang digunakan. Evaluasi ini akan memberikan wawasan tentang keunggulan dan kelemahan dari masing-masing arsitektur dalam mendeteksi dan mengklasifikasikan kendaraan secara real-time. Hasil dari analisis ini akan menjadi landasan untuk menentukan langkah-langkah selanjutnya dalam pengembangan penelitian ini.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Dalam penelitian ini menggunakan dua jenis dataset yaitu :

1. Dataset Kendaraan
2. Dataset Non-Kendaraan (Pemandangan)

4.1.1 Dataset Kendaraan

Dataset yang digunakan dalam penelitian ini adalah *BIT-Vehicle* (Dong et al., 2015a) yang terdiri dari sekumpulan gambar kendaraan yang diambil menggunakan dua kamera pada berbagai waktu dan lokasi. Ukuran gambar bervariasi antara 1600x1200 dan 1920x1080 piksel. Dataset ini mencakup variasi kondisi pencahayaan, skala, warna permukaan kendaraan, serta sudut pandang yang berbeda. Beberapa gambar mungkin tidak menampilkan bagian atas atau bawah kendaraan karena keterlambatan dalam pengambilan gambar dan variasi ukuran kendaraan. Setiap gambar dapat mengandung satu atau dua kendaraan, dengan lokasi masing-masing kendaraan telah diannotasi sebelumnya. Kendaraan-kendaraan dalam dataset ini awalnya terbagi ke dalam enam kategori dengan jumlah yang bervariasi seperti yang telah disajikan dalam Tabel 4.1.

Table 4.1. Kategori Kendaraan

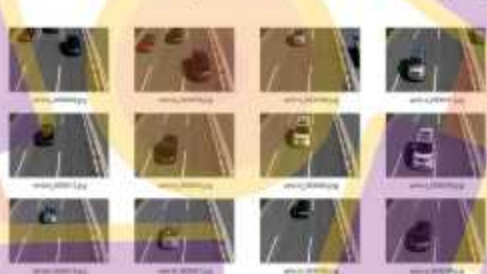
No	Kategori	Jumlah
1	Bus	555
2	Mikrobus	883
3	Minivan	476
4	Sedan	5.922
5	SUV	1.392
6	Truk	822

Namun, untuk memudahkan analisis, keenam kategori tersebut kemudian disederhanakan menjadi tiga kategori, yaitu Bus, Mobil, dan Truk. Asumsinya adalah sedan, SUV, minivan, dan mikrobus termasuk dalam jenis mobil. Berikut adalah rincian jumlah kendaraan dalam setiap kategori setelah disederhanakan.

Table 4.2. Kategori Kendaraan Setelah Disederhanakan

No	Kategori	Jumlah
1	Bus	555
2	Car	8529
3	Truck	822

Dataset ini akan menjadi sumber informasi yang berharga dalam pengembangan dan penelitian terkait deteksi serta identifikasi kendaraan dalam berbagai kondisi. Gambar 4.1 merupakan contoh dari dataset yang kami gunakan.



Gambar 4.1. Dataset Non-Kendaraan

4.1.2 Dataset NonKendaraan (Pemandangan)

Untuk melengkapi dataset kendaraan, kami menyertakan citra non-kendaraan dari dataset pemandangan yang dikumpulkan oleh (Oliva & Torralba, 2001). Dataset ini terdiri dari sekitar 26.000 gambar pemandangan dengan resolusi mulai dari 1683×1080 hingga 1080×1620 piksel, dengan ukuran file

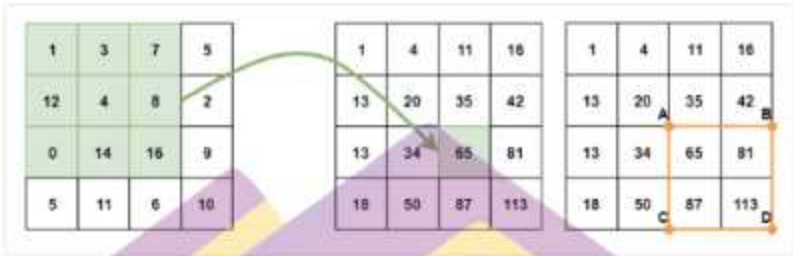
yang kurang dari 1MB dan disimpan dalam format warna RGB. Dalam rangka penelitian ini, kami secara selektif memilih 10.000 citra pemandangan dari dataset tersebut. Proses selektif yang kami lakukan bertujuan untuk menghindari citra pemandangan yang mengandung objek kendaraan. Gambar 4.2 menampilkan contoh citra pemandangan yang telah kami seleksi untuk digunakan dalam penelitian kami.

Gambar 4.2. Dataset Non-Kendaraan

4.2 Pembuatan Haar Like Features

Haar Like Features adalah fitur yang digunakan dalam deteksi objek yang diimplementasikan dalam algoritma *Haar Cascade Classifier*. Fitur ini bekerja dengan mendeteksi perubahan intensitas pada gambar, seperti tepi, garis, dan sudut, yang merupakan karakteristik penting dalam mengenali bentuk dan kontur objek. Manfaat utama dari *Haar Like Features* adalah kemampuannya untuk mendeteksi pola spesifik dengan efisien dan cepat. Karena sifatnya yang sederhana dan cepat dihitung, *Haar Like Features* sangat efektif dalam pemrosesan gambar waktu nyata,

yang sangat penting untuk aplikasi seperti sistem transportasi cerdas dan pengawasan lalu lintas.



Gambar 4.3. Gambar 5x5 Piksel dari Sebuah Citra

Pada proses pengolahan citra menggunakan *Haar Cascade Classifier*, salah satu langkah awal adalah mengubah gambar asli menjadi gambar integral. Contoh sederhana dari proses ini dapat dijelaskan melalui gambar berikut.

Gambar 4.3 menunjukkan gambar asli berukuran 5x5 piksel (A) dan hasil transformasinya menjadi gambar integral (B). Gambar integral ini menghitung jumlah kumulatif dari piksel di atas dan di kiri setiap piksel, termasuk piksel itu sendiri. Misalnya, untuk piksel pada koordinat (1,1) di gambar asli, nilai integralnya adalah 1, yaitu nilai piksel itu sendiri. Sedangkan untuk piksel pada koordinat (1,2), nilai integralnya adalah 4, yang diperoleh dari penjumlahan nilai piksel pada (1,1) dan (1,2), yaitu 1 + 3.

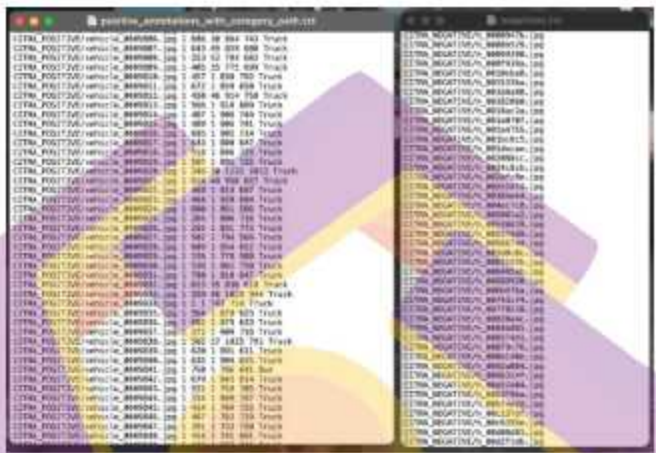
Dalam gambar integral (B), nilai pada setiap piksel merupakan jumlah dari semua nilai piksel yang berada di atas dan di kiri piksel tersebut dalam gambar asli. Contoh lainnya adalah piksel pada koordinat (2,2) di gambar asli dengan nilai 4.

Nilai integralnya di gambar (B) adalah 20, yang diperoleh dari penjumlahan nilai pada piksel (1,1), (1,2), (2,1), dan (2,2) di gambar asli, yaitu 1 + 3 + 12 + 4.

Proses ini memudahkan perhitungan *Haar Like Features* karena integral dari area mana pun dalam gambar asli dapat dihitung dengan cepat menggunakan beberapa operasi sederhana pada gambar *integral*. Dengan begitu, deteksi tepi, garis, dan elemen visual lainnya dapat dilakukan lebih efisien dan cepat, memungkinkan implementasi real-time dalam berbagai aplikasi seperti sistem transportasi cerdas dan pengawasan lalu lintas.

Dalam implementasi praktis, khususnya untuk deteksi mobil menggunakan *OpenCV*, proses pembuatan *Haar Like Features* dimulai dengan pembuatan file anotasi yang berisi informasi lokasi dan ukuran objek kendaraan pada gambar. File anotasi ini sangat penting karena memberikan panduan bagi algoritma untuk mengenali dan mengekstraksi *fitur-fitur Haar* yang relevan dari gambar. Dengan informasi lokasi dan ukuran objek yang tepat, algoritma dapat belajar dengan lebih baik dan meningkatkan akurasi deteksi objek kendaraan dalam gambar. Proses ini mencakup beberapa langkah, termasuk pembacaan data anotasi, pemrosesan gambar, dan pelatihan model menggunakan fitur Haar untuk mencapai deteksi yang optimal. Selanjutnya, dilakukan proses pembuatan detektor negatif untuk mengumpulkan gambar-gambar non-kendaraan(pemandangan). Detektor negative dalam proses ini bertujuan untuk memberikan kontras antara gambar-gambar kendaraan (sampel positif) dengan gambar-gambar non-kendaraan (sampel negatif) serta membantu dalam meningkatkan ketepatan deteksi dengan memberikan

referensi objek yang bukan kendaraan, sehingga algoritma dapat membedakan antara objek kendaraan dan objek lainnya dengan lebih baik. Gambar 4.4 merupakan hasil dari file annotasi dan detektor negatif.



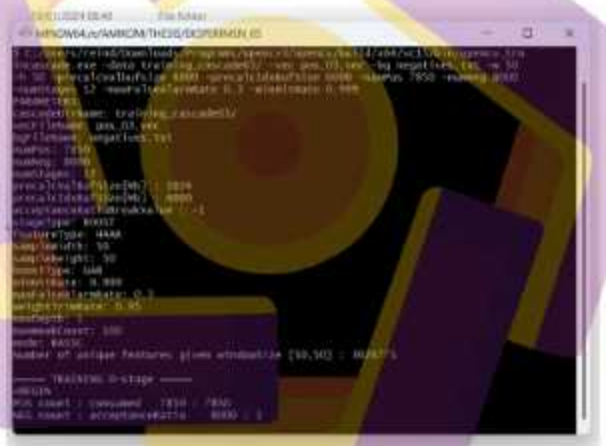
Gambar 4.4. Hasil File Annotasi dan Deteksi Negatif

Selanjutnya adalah proses pembuatan file vektor dengan menggunakan alat bawaan OpenCV. Langkah-langkah dalam pembuatan file vektor meliputi konversi gambar-gambar pelatihan ke dalam format *grayscale*, ekstraksi fitur-fitur Haar dari setiap gambar pelatihan, penyusunan fitur-fitur *Haar* ke dalam vektor yang sesuai dengan format yang diterima oleh algoritma pelatihan *Cascade Classifier*, dan pemrosesan vektor untuk menghilangkan redundansi serta meningkatkan efisiensi penggunaan memori. Tujuan dari pembuatan file vektor ini adalah untuk menyederhanakan proses pelatihan dan meningkatkan efisiensi algoritma dengan menyimpan fitur-fitur yang relevan dari gambar-gambar pelatihan dalam format yang lebih terstruktur dan terorganisir.

Gambar 4.5. Pembuatan File Vektor

Dari gambar 4.5, *Createsamples.exe* merupakan tool bawaan dari OpenCV yang digunakan untuk menghasilkan sampel positif dari citra objek yang diinginkan dan mengkonversinya menjadi file vektor yang nantinya akan digunakan dalam pelatihan *Cascade Classifier*. "-info *positive_annotations_with_category_path.txt*" merujuk pada lokasi file annotasi yang sebelumnya telah dibuat. Sedangkan "-vec *positive_samples.vec*" menunjukkan lokasi tempat file vektor akan disimpan setelah menjalankan perintah ini. "-num 9000" menentukan jumlah citra positif yang akan digunakan dalam proses pembuatan vektor. Sedangkan parameter "-w 50 -h 50" menandakan lebar dan tinggi dari objek dalam citra. Setelah perintah tersebut dijalankan, *Createsamples.exe* akan memuat dokumen *positive_annotations_with_category_path.txt* untuk menyimpan semua citra objek positif ke dalam sebuah file vektor yang bernama *positive_samples.vec*. Hasil vektor tersebut akan tersimpan di dalam folder "Thesis" setelah proses berhasil dieksekusi.

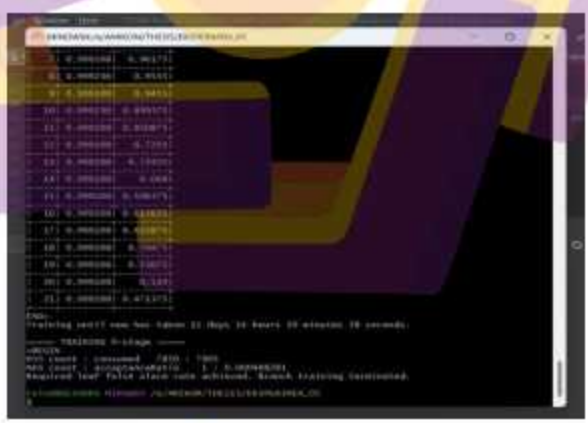
Langkah yang terakhir adalah melatih *Haar Like Features* dengan menggunakan alat pelatihan *Cascade Classifier* yang disediakan oleh OpenCV (Kenda & Witanti, 2021). Proses pelatihan ini mencakup penentuan sampel positif dan negatif, dimana sampel positif mencakup gambar-gambar kendaraan yang telah dianotasi, sedangkan sampel negatif mencakup gambar-gambar non-kendaraan yang telah didefinisikan. Setelah menentukan sampel, model dilatih dengan menggunakan algoritma pembelajaran mesin untuk mengidentifikasi fitur-fitur *Haar* (Darmawan, 2022) yang mewakili objek kendaraan. Untuk proses pelatihannya dapat dilihat pada gambar 4.6.



Gambar 4.6. Proses Pelatihan *Features Haar Like*

OpenCV_traincascade.exe adalah alat bawaan dari OpenCV yang digunakan untuk melatih *Features Haar Like* dengan menggunakan sampel positif dan negatif. Proses pelatihan dilakukan dengan mengatur berbagai parameter, seperti *-data training_cascade03* untuk lokasi penyimpanan hasil training, *-vec*

pos_03.vec untuk lokasi file vektor hasil ekstraksi fitur Haar sebelumnya, dan *-bg negatives.txt* untuk lokasi daftar citra negatif. Jumlah citra positif dan negatif ditentukan oleh *-npos* dan *-nneg*, dengan syarat jumlah citra negatif harus lebih besar atau sama dengan jumlah citra positif. Tahapan training ditentukan oleh *-nstages*, sementara *-mem* digunakan untuk mengatur total memori yang akan digunakan dalam Mb. Ukuran lebar dan tinggi objek ditentukan oleh *-w* dan *-h*, yang harus sesuai dengan ukuran yang telah ditentukan sebelumnya. Proses training bisa berhenti sebelum mencapai jumlah tahapan yang ditentukan jika kriteria yang dibutuhkan telah terpenuhi. Dengan menggunakan semua parameter tersebut, proses *training* memakan waktu sekitar 11 hari 14 Jam ketika dijalankan diperangkat Thinkpad X240 dengan spesifikasi RAM 8GB, Intel® Prosesor Core i7-4600U (3.30GHz). Gambar 4.7 merupakan gambaran dari proses pembuatan *Features Haar Like*.



Gambar 4.7. Hasil Proses *Training Features Haar Like*

Setelah proses pelatihan selesai, akan dihasilkan file *cascade* dengan format .XML. File ini berisi informasi tentang model *Cascade Classifier* yang telah dilatih, termasuk kriteria pemilihan fitur, bobot dari setiap fitur, dan struktur classifier yang telah disesuaikan selama proses *training*. File XML ini dapat digunakan untuk diterapkan pada proses implementasi *filter haar cascade classifier* untuk Deteksi kendaraan Pada video *CCTV*. Gambar 4.8 merupakan hasil dari proses pembuatan *Haar Like Features*.

Gambar 4.8. Hasil Model dari proses pembuatan *Haar Like Features*

4.3 Implementasi *Haar Like Features* untuk Deteksi Kendaraan

Setelah berhasil membuat file *cascade.xml*, langkah selanjutnya adalah melakukan implementasi model tersebut. Proses ini bertujuan untuk mengevaluasi performa model yang telah dibuat dalam mendeteksi kendaraan dalam berbagai kondisi lingkungan dan situasi. Implementasi model akan dilakukan dengan menggunakan dataset uji yang terdiri dari satu video dari *CCTV* dan gambar ruas lalu lintas. Video tersebut akan digunakan untuk menguji kemampuan model dalam mendeteksi kendaraan secara real-time, sedangkan gambar ruas lalu lintas akan

digunakan sebagai data uji statis. Selama proses implementasi, akan dievaluasi seberapa baik model mampu mengidentifikasi kendaraan dalam video, termasuk deteksi kendaraan dalam berbagai skenario lalu lintas dan kondisi pencahayaan yang berbeda. Selain itu, gambar ruas lalu lintas akan digunakan untuk menguji kemampuan model dalam mendeteksi kendaraan dalam citra statis.

Dari hasil uji tersebut, diketahui bahwa model belum sepenuhnya dapat mendeteksi objek kendaraan secara optimal, terutama pada objek yang berdempetan dan terlalu jauh dari kamera. Hal ini menunjukkan adanya beberapa kendala dalam performa deteksi, seperti kehilangan detail pada objek yang berdekatan atau terlalu kecil, serta kesulitan dalam mengidentifikasi objek yang terlalu jauh dari bidang pandang kamera. Gambar dibawah merupakan hasil dari implementasi model *cascade.xml* pada video *CCTV*

Gambar 4.9. Hasil Proses Implementasi Model *cascade.xml*

Penyebab dari permasalahan tersebut dapat berasal dari beberapa faktor yang memengaruhi performa deteksi model. Salah satu penyebabnya adalah pengaturan parameter pelatihan seperti jumlah tahapan (*nstages*), ukuran jendela deteksi (w dan h), atau tingkat kecermatan (*minhitrate* dan *maxfalsealarm*) juga dapat mempengaruhi performa model. Penyesuaian yang kurang tepat pada parameter-parameter ini dapat mengakibatkan model menjadi kurang sensitif terhadap objek yang berdekatan atau terlalu jauh dari kamera, serta meningkatkan risiko terjadinya kesalahan deteksi.

4.4 Pembuatan Model *Convolutional Neural Network* (CNN) Untuk Klasifikasi Kendaraan

Tahap selanjutnya akan difokuskan pada pembuatan Model CNN untuk klasifikasi kendaraan. Penyusunan model CNN ini dilakukan menggunakan bahasa pemrograman *Python* dengan bantuan *library Tensorflow* dan *Keras*. Proses pelatihan model dilakukan pada *platform Jupyter Notebook* yang dijalankan di *Cursor AI*. Selain itu, pelatihan dilakukan menggunakan perangkat MacBook Pro 2017 dengan spesifikasi yang mencakup prosesor Intel Core i7, RAM 16GB, dan GPU Intel Iris Plus Graphics 640.

4.4.1 Splitting Data

Kemudian tahap selanjutnya dataset *BIT-Vehicle* dibagi menjadi tiga bagian yang terpisah: *training*, *validation*, dan *testing*. Pembagian ini sangat penting dalam proses pengembangan model, karena setiap bagian memiliki peranannya masing-masing. Bagian *training* digunakan untuk melatih model, sementara bagian

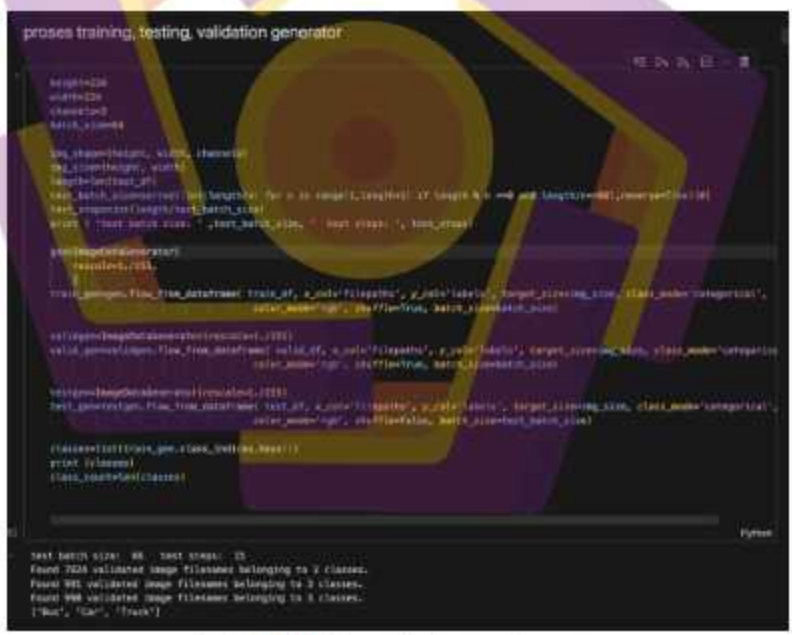
validation digunakan untuk mengevaluasi performa model secara berkala selama proses pelatihan, dan bagian *testing* digunakan untuk menguji performa akhir model setelah pelatihan selesai. Dengan adanya pembagian ini, kami dapat memastikan bahwa model yang dikembangkan mampu menggeneralisasi data dengan baik dan memiliki kinerja yang stabil saat diterapkan pada data baru.

Gambar 4.10. Proses Pembagian Dataset

Gambar 4.10 menunjukkan bagaimana dataset dibagi menjadi tiga bagian: training, testing, dan validation. Variabel 'train_split' dan 'test_split' menunjukkan seberapa besar bagian dataset yang akan digunakan untuk masing-masing bagian. Dataset dibagi menjadi dua bagian menggunakan fungsi 'train_test_split' dari library scikit-learn. Variabel 'train_df', digunakan untuk melatih model, sementara variabel 'dummy_df', dibagi lagi untuk testing dan validation yang dibagi menggunakan fungsi 'train_test_split' sebagai pemisah data testing dan validation. Proporsi data testing ditentukan oleh variabel 'test_split', sedangkan data validation akan memperoleh sisa dari 'dummy_df'. Dengan cara ini, kami memastikan model dilatih dengan data yang cukup, diuji dengan data yang belum pernah dilihat sebelumnya, dan dievaluasi secara berkala selama proses pelatihan untuk memastikan kinerjanya yang optimal.

4.4.2 Generator (Augmentasi Data)

Tahap selanjutnya adalah pembuatan generator menggunakan *ImageDataGenerator* dari *library Keras*. *Generator* ini secara otomatis menangani augmentasi gambar dan *preprocessing* seperti *rescaling*. Penggunaan generator ini memudahkan pengelolaan memori dan meningkatkan efisiensi pelatihan dengan mengelola batch gambar secara otomatis. Dengan menggunakan generator, kami dapat dengan mudah mengatur dan mengelola aliran data gambar yang akan digunakan dalam proses pelatihan dan pengujian model. Untuk proses pembuatan generatornya seperti pada gambar 4.11.



Gambar 4.11. Proses Pembuatan Generator

Pertama, kami menetapkan dimensi gambar (*height*, *width*, *channels*) dan ukuran gambar (*img_size*) sesuai dengan nilai yang telah ditentukan sebelumnya. Kemudian, kami menghitung *batch size* untuk data pengujian (*test_batch_size*) berdasarkan panjang data uji untuk memastikan ukuran batch yang optimal untuk efisiensi pengujian, dan menentukan jumlah langkah (*test_steps*) yang akan dilakukan selama proses pengujian. Selanjutnya, kami menggunakan *ImageDataGenerator* dari *library Keras* untuk membuat generator untuk data pelatihan, validasi, dan pengujian. Generator ini melakukan *rescaling* pada nilai piksel gambar agar berada dalam rentang 0 hingga 1. Untuk data pelatihan (*train_gen*) dan validasi (*valid_gen*), kami menggunakan *flow_from_dataframe* untuk mengonversi *dataframe* ke dalam generator yang dapat digunakan langsung dalam proses pelatihan dan validasi model. Sedangkan untuk data pengujian (*test_gen*), kami menggunakan *flow_from_dataframe* dengan menonaktifkan pengacakan (*shuffle=False*) karena pengujian memerlukan urutan data yang tetap. Terakhir, kami menyimpan daftar kelas (*classes*) berdasarkan indeks yang diberikan oleh generator pelatihan, serta menghitung jumlah kelas (*class_count*) yang akan digunakan dalam proses klasifikasi. Dengan langkah-langkah ini, kami siap untuk melanjutkan proses pembuatan dan pelatihan model CNN untuk klasifikasi kendaraan. Pada gambar 4.12 merupakan hasil dari proses *ImageGenerator*

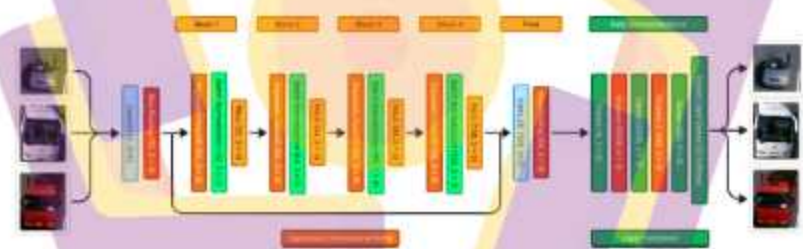
Gambar 4.12. Hasil Dari *ImageGenerator*

4.4.3 Pembuatan dan Pelatihan Model

Pada tahap pembuatan atau pelatihan model, kami menggunakan arsitektur *Xception* dan *VGG16* sebagai dasar model dengan menambahkan beberapa layer tambahan untuk klasifikasi. Untuk memahami perbedaan performa kedua model, penting untuk mengetahui cara kerja *VGG16* dan *Xception*.

a. Xception

Xception adalah model CNN yang telah diakui keunggulannya dalam berbagai tugas klasifikasi citra. Keunggulan utama dari *Xception* adalah kemampuannya untuk memecah proses *convolution* menjadi dua tahap, yakni *depthwise convolution* dan *pointwise convolution*. Ini memungkinkan model untuk menangkap variasi yang lebih detail dalam data gambar dengan komputasi yang lebih efisien. Dalam penelitian ini, model *Xception* dimodifikasi dan diterapkan dengan memanfaatkan pendekatan *transfer learning*, yang berarti model ini diinisialisasi dengan bobot dari pelatihan sebelumnya pada *dataset* besar seperti *ImageNet*, dan kemudian disesuaikan dengan dataset kendaraan spesifik.



Gambar 4.13. Struktur Arsitektur *Xception*

Gambar 4.13 menunjukkan struktur modifikasi model *Xception* yang digunakan dalam penelitian ini. Modifikasi ini mencakup penambahan lapisan-lapisan baru yang disesuaikan dengan tugas spesifik, yaitu klasifikasi kendaraan menjadi tiga kelas: mobil, truk, dan bus. Dalam pendekatan transfer learning ini, bagian awal model, yang bertugas sebagai fitur extractor, tetap mempertahankan bobot dari pelatihan awal (*pre-trained*), sementara bagian

akhir model diubah untuk menyesuaikan dengan klasifikasi yang diinginkan. Penjelasan berikut akan menguraikan secara detail cara kerja model berdasarkan struktur yang ditunjukkan pada Gambar 4.14, termasuk bagaimana model mengolah gambar input melalui lapisan-lapisan *convolution*, *batch normalization*, *activations*, hingga akhirnya menghasilkan prediksi pada lapisan *fully connected*.

1. Proses Input dan Blok Konvolusi Awal

Proses klasifikasi kendaraan dimulai dengan menerima gambar input yang dapat berupa salah satu dari tiga kelas: mobil, truk, atau bus. Gambar ini pertama kali diproses oleh lapisan Conv2D dengan 32 filter ukuran 3x3. Fungsi utama dari lapisan ini adalah untuk mengekstraksi fitur dasar dari gambar, seperti tepi, sudut, dan pola tekstur. Hasil dari proses ini adalah representasi fitur awal dari gambar yang akan digunakan dalam tahap-tahap selanjutnya. Selain itu, operasi max pooling dengan ukuran 2x2 dan stride 2x2 diterapkan untuk mengurangi dimensi fitur, yang membantu mengurangi beban komputasi dan memfokuskan pada fitur yang paling penting.

2. Blok Konvolusi Terpisah (Separable Convolutional Blocks)

Setelah melalui blok konvolusi awal, data fitur kemudian diproses melalui serangkaian blok konvolusi terpisah. Setiap blok terdiri dari lapisan *Separable Conv2D*, *BatchNormalization*, dan *ReLU*. *Separable Conv2D* adalah teknik yang memisahkan konvolusi standar menjadi dua tahap:

depthwise convolution, yang memproses setiap channel fitur secara terpisah, dan *pointwise convolution*, yang menggabungkan *output* dari *depthwise convolution*. Pendekatan ini lebih efisien dan mampu menangkap fitur yang lebih kompleks. *Batch normalization* diterapkan untuk menstabilkan distribusi data dan mempercepat proses pelatihan, sementara aktivasi ReLU menambahkan non-linearitas untuk membantu jaringan dalam menangani variasi yang lebih kompleks dalam data. Di setiap tahap blok, jumlah filter meningkat (32, 64, 96, 128) untuk memungkinkan ekstraksi fitur yang lebih dalam dan kompleks.

3. Blok Lapisan Fully Connected

Setelah melewati blok konvolusi terpisah, fitur-fitur yang telah diproses diratakan menggunakan lapisan *Flatten* menjadi vektor 1D. Vektor ini kemudian melewati beberapa lapisan *dense*, yang terdiri dari sejumlah neuron yang terhubung penuh dengan neuron di lapisan sebelumnya. Setiap lapisan *dense* belajar untuk menggabungkan fitur-fitur yang dihasilkan sebelumnya dan menginterpretasikannya dalam konteks klasifikasi. *Dropout* diterapkan setelah beberapa lapisan *dense* untuk mencegah *overfitting* dengan cara secara acak mengabaikan sejumlah neuron selama pelatihan, yang membantu model untuk menjadi lebih general. Lapisan *dense* terakhir menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas untuk setiap kelas kendaraan, yaitu mobil, truk, atau bus.

4. Output dan Klasifikasi Akhir

Pada tahap akhir, model menghasilkan output berupa probabilitas untuk masing-masing kelas kendaraan. Nilai probabilitas ini menunjukkan tingkat keyakinan model terhadap input gambar yang termasuk dalam setiap kelas. Klasifikasi akhir ditentukan dengan memilih kelas dengan probabilitas tertinggi. Misalnya, jika model memprediksi probabilitas tertinggi untuk kelas 'mobil', maka gambar input akan diklasifikasikan sebagai mobil. Hasil akhir ini dapat digunakan untuk berbagai aplikasi, seperti pemantauan lalu lintas, sistem parkir otomatis, atau analisis data kendaraan.

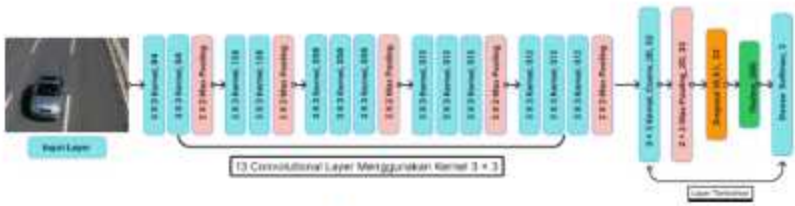
Kesimpulannya, modifikasi model *Xception* dengan *transfer learning* memberikan efisiensi dan efektivitas tinggi dalam klasifikasi kendaraan. Dengan memanfaatkan bobot yang telah dilatih sebelumnya, proses pelatihan menjadi lebih cepat dan memerlukan lebih sedikit data. Struktur model, yang melibatkan lapisan convolutional dan *separable convolutional*, secara efektif menangkap fitur kompleks dengan bantuan *batch normalization* dan fungsi aktivasi *ReLU*. Lapisan *fully connected* yang telah dimodifikasi untuk klasifikasi kendaraan memungkinkan model untuk menghasilkan prediksi akhir yang akurat untuk kelas mobil, truk, dan bus. Penerapan *dropout* juga berperan penting dalam mencegah *overfitting*, sehingga meningkatkan kinerja model pada data baru. Secara keseluruhan, modifikasi ini menunjukkan bahwa *Xception* adalah model yang kuat dan efisien untuk tugas klasifikasi objek dalam domain kendaraan.

b. VGG-16

VGG16 merupakan salah satu arsitektur CNN yang sangat terkenal dan digunakan secara luas dalam bidang pengenalan gambar. Arsitektur ini dikenal karena kesederhanaan strukturnya yang hanya menggunakan lapisan konvolusional berukuran 3×3 yang ditumpuk satu sama lain untuk membangun jaringan yang dalam. *VGG16* terdiri dari 16 lapisan yang mencakup 13 lapisan konvolusional dan 3 lapisan *fully connected*. Gambar 4.14 menunjukkan struktur arsitektur *VGG16* secara rinci.

Gambar 4.14. Struktur Arsitektur *VGG16*

Pada implementasi kami, struktur arsitektur *VGG16* telah dimodifikasi dengan menambahkan beberapa lapisan tambahan setelah lapisan ke-13 untuk meningkatkan kemampuan klasifikasi model. Modifikasi ini mencakup penghapusan *fully connected* dan *softmax* terakhir dan menambahkan lapisan konvolusional, *max pooling*, *dropout*, *flatten*, dan *dense* yang dirancang untuk lebih mengakomodasi kebutuhan klasifikasi spesifik kami. Gambar 4.15 menunjukkan struktur arsitektur VGG16 yang telah kami dimodifikasi.



Gambar 4.15. Struktur Arsitektur *VGG16*

Berikut adalah penjelasan detail tentang penggunaan tiap bagian arsitektur yang diterapkan pada penelitian kami:

1. Lapisan Input menerima gambar berukuran 224x224 piksel dengan 3 saluran warna (RGB) yang kemudian diteruskan ke lapisan konvolusi.
2. Lapisan konvolusi pertama dan kedua terdiri dari 64 filter fitur dengan ukuran filter 3×3. Ketika gambar input (gambar kendaraan RGB) melewati lapisan konvolusi pertama dan kedua, dimensi gambar berubah menjadi 224x224x64. Output yang dihasilkan kemudian diteruskan ke lapisan *max pooling* dengan *stride* 2, yang mengurangi dimensi menjadi 112x112x64.
3. Lapisan konvolusi ketiga dan keempat terdiri dari 128 filter fitur dengan ukuran filter 3×3. Kedua lapisan ini diikuti oleh lapisan *max pooling* dengan *stride* 2, dan *output* yang dihasilkan berkurang menjadi 56x56x128.
4. Lapisan konvolusi kelima, keenam, dan ketujuh menggunakan filter 3×3 dengan 256 *feature map* . Ketiga lapisan ini diikuti oleh lapisan *max pooling* dengan *stride* 2, yang mengurangi dimensi menjadi 28x28x256.
5. Lapisan kedelapan hingga ketiga belas adalah dua set lapisan konvolusional dengan ukuran kernel 3×3, masing-masing terdiri dari 512 filter kernel.

Lapisan-lapisan ini diikuti oleh lapisan *max pooling* dengan *stride* 2, yang mengurangi dimensi menjadi 14x14x512 dan kemudian menjadi 7x7x512.

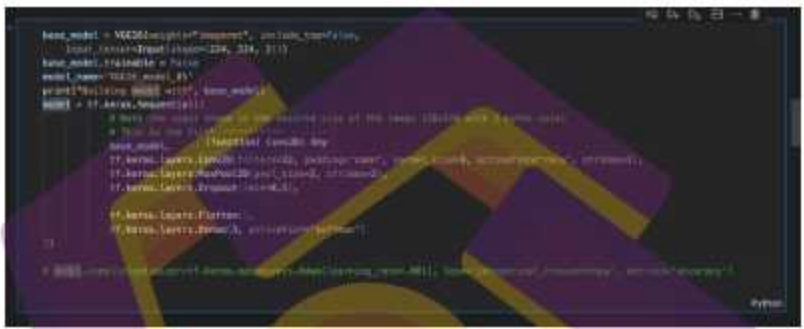
6. Lapisan keempat belas atau (*Convolutional 2D layer*) adalah lapisan konvolusional tambahan dengan 32 filter dengan ukuran kernel 3x3 dan aktivasi ReLU. Lapisan ini mengambil output dari lapisan ketiga belas dan menghasilkan 32 *feature map* dengan dimensi yang sama (7x7). Lapisan ini bertujuan untuk mengekstraksi lebih banyak fitur spesifik dari gambar input.
7. Lapisan kelima belas (*MaxPooling2D layer*) mengurangi dimensi *feature map* menjadi 3x3 sambil mempertahankan jumlah filter (32). Lapisan *max pooling* ini bertugas untuk mengurangi resolusi spasial dari fitur yang dipetakan, sehingga mengurangi jumlah parameter dan komputasi dalam jaringan.
8. Lapisan keenam belas (*dropout layer*) dengan *rate* 0.5. Lapisan ini secara acak menonaktifkan 50% *neuron* selama pelatihan untuk mengurangi *overfitting*. *Dropout layer* membantu meningkatkan generalisasi model dengan mencegah model dari terlalu cocok dengan data pelatihan.
9. Lapisan ketujuh belas (*Flatten layer*) meratakan *output* dari lapisan konvolusional dan *pooling* menjadi vektor 1D dengan panjang 288 (3 * 3 * 32). Lapisan ini mengubah data multi-dimensi menjadi satu dimensi untuk dapat diteruskan ke lapisan *fully connected*.

10. Lapisan kedelapan belas (*Dense layer*) merupakan lapisan *fully connected* dengan 3 unit dan aktivasi *softmax*. Lapisan ini menghasilkan probabilitas untuk setiap kelas dari 3 kelas yang ada. Lapisan ini bertugas untuk melakukan klasifikasi akhir berdasarkan fitur-fitur yang telah diekstraksi oleh lapisan-lapisan sebelumnya.

Kesimpulannya dari model yang dimodifikasi ini, dimulai dengan lapisan input yang menerima gambar RGB berukuran 224x224. Kemudian, model *VGG16* digunakan sebagai *feature extractor*, yang menghasilkan *feature map* dengan dimensi 7x7x512. *Feature map* ini kemudian diproses lebih lanjut oleh lapisan konvolusional tambahan dengan 32 filter, diikuti oleh lapisan max pooling yang mengurangi dimensi *feature map* menjadi 3x3. Lapisan dropout diterapkan untuk mengurangi *overfitting*, dan hasilnya diratakan oleh lapisan flatten menjadi vektor 1D. Akhirnya, vektor ini diteruskan ke lapisan *fully connected* dengan 3 unit dan aktivasi *sofimax* untuk klasifikasi ke dalam 3 kelas.

Dengan menggunakan *Xception* dan *VGG16* sebagai dasar model, kami dapat memanfaatkan pengetahuan yang telah dipelajari oleh model tersebut dan fokus pada pelatihan layer-layer tambahan yang lebih spesifik untuk tugas klasifikasi kendaraan. Hal ini membantu mempercepat proses pelatihan dan meningkatkan performa model secara keseluruhan. Gambar 4.16 dan 4.17 merupakan proses pembuatan kedua model tersebut.

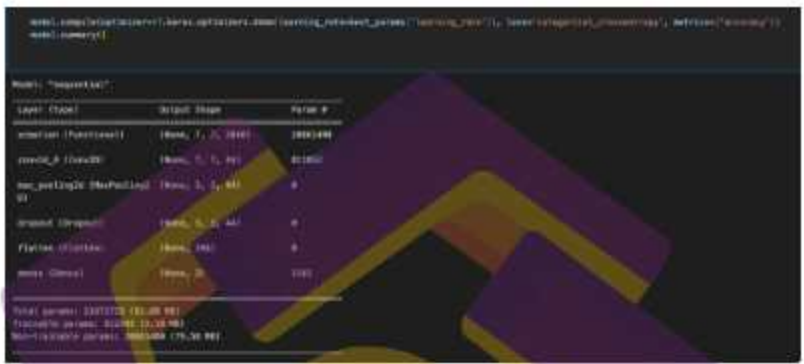
Gambar 4.16. Proses Pembuatan Model Xception



Gambar 4.17. Proses Pembuatan Model VGG16

Kedua model yang telah dilatih sebelumnya dengan dataset *ImageNet* digunakan sebagai dasar model. Kami mengatur parameter (*include_top*) menjadi *False* untuk menghilangkan layer *fully connected* di bagian atas model, sehingga kami dapat menambahkan layer klasifikasi sendiri sesuai kebutuhan. Selanjutnya, kami menetapkan layer-layer model yang akan dilatih ulang. Dalam hal ini, semua layer diatur sebagai tidak dapat dilatih, kecuali layer-layer yang kami tambahkan setelahnya. Model dibangun secara berurutan menggunakan pendekatan *sequential*, dimulai dengan penambahan layer konvolusi untuk mengekstraksi fitur dari gambar, diikuti oleh layer *max pooling* untuk mengurangi dimensi fitur. Untuk mencegah *overfitting*, kami juga menambahkan layer *dropout*. Setelah itu, fitur diubah menjadi vektor satu dimensi menggunakan layer *flatten*, sebelum

dilanjutkan dengan penambahan layer *fully connected* untuk klasifikasi. Layer ini menggunakan fungsi aktivasi *softmax* untuk *output* kelas. Selanjutnya merupakan bagian untuk melakukan kompilasi model seperti pada gambar 4.18 dan 4.19.

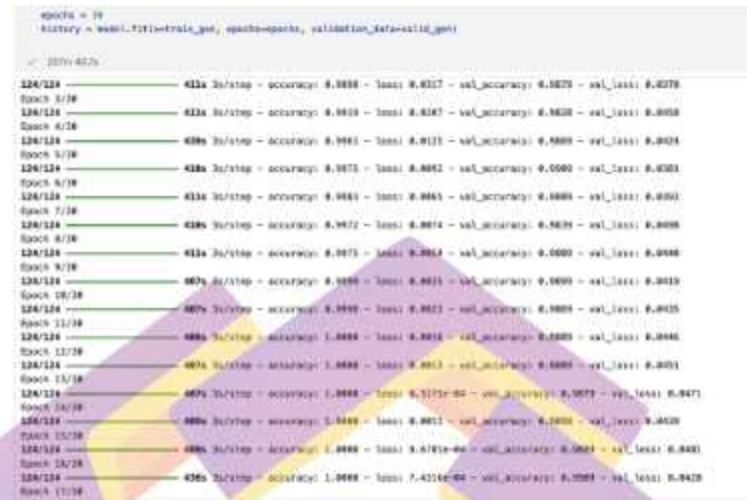


Gambar 4.18. Tahapan Kompilasi Model *Xception*

Gambar 4.19. Tahapan Kompilasi Model *VGG16*

Kami menggunakan *Adam optimizer* untuk menyesuaikan bobot model selama pelatihan, karena *optimizer* ini efektif dalam menangani masalah pelatihan. Sebagai fungsi *loss*, kami memilih *categorical crossentropy* karena sesuai untuk masalah klasifikasi multi-kelas. Untuk mengukur performa model selama pelatihan, kami menggunakan akurasi sebagai metrik evaluasi utama. Setelah pelatihan, kami melakukan visualisasi model yang mencakup ringkasan model, termasuk layer-layer yang ditambahkan, jumlah parameter, dan *output shape* dari setiap layer. Ini membantu kami memahami struktur model secara keseluruhan dan mengidentifikasi potensi perubahan yang diperlukan. Dengan langkah-langkah ini, kami siap untuk melanjutkan proses pelatihan model dan mengevaluasi hasilnya.

Selanjutnya adalah bagian pelatihan model, yang mana merupakan langkah kunci dalam pembuatan model CNN. Proses pelatihan model dilakukan menggunakan data dari generator *training* dan *validation*. Jumlah *epochs* (iterasi melalui seluruh dataset) yang digunakan untuk pelatihan adalah 30 epoch untuk model *Xception* dan 100 *epoch* untuk VGG16. Metode pelatihan menggunakan metode *fit* dengan memakan waktu pelatihan sekitar 3 Jam 45menit untuk model *Xception* dan 8 Hari 2 Jam untuk model *VGG16*. Gambar 4.20 dan 4.21 merupakan contoh hasil dari proses training dari kedua model tersebut.



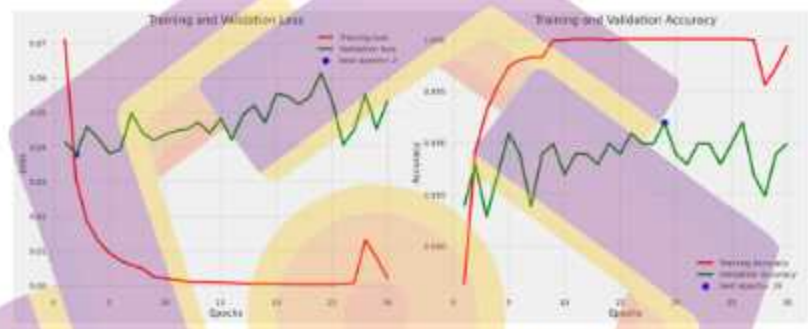
Gambar 4.20. Proses *Training* Model *Xception*

Gambar 4.21. Proses *Training* Model *VGG16*

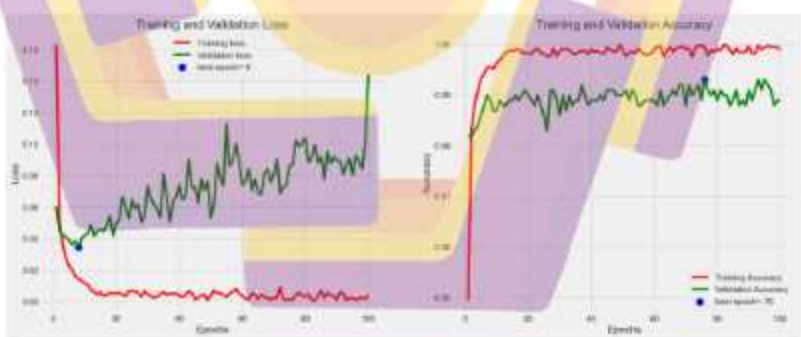
4.4.4 Evaluasi Model

Setelah proses pelatihan selesai, tahap selanjutnya adalah evaluasi model. Evaluasi dilakukan dengan menggunakan data testing untuk mengetahui akurasi model dalam mengklasifikasikan data baru yang belum pernah dilihat sebelumnya.

Proses evaluasi ini memungkinkan kita untuk mengukur kinerja model dan memastikan bahwa model yang telah dilatih mampu menggeneralisasi dengan baik pada data yang baru. Dengan memahami akurasi model, kita dapat mengetahui seberapa baik model dapat memprediksi kelas dari sampel-sampel yang belum pernah dilihat sebelumnya, sehingga dapat digunakan secara efektif dalam aplikasi praktis.



Gambar 4.22. Grafik Dari Evaluasi Model *Xception*



Gambar 4.23. Grafik Dari Evaluasi Model *VGG16*

Grafik yang ditampilkan menggambarkan dinamika *loss* dan akurasi selama proses pelatihan dan validasi model *xception*. Dari grafik "*Training and Validation Loss*", kita dapat melihat bahwa *loss* pelatihan (ditunjukkan dengan garis merah)

menurun secara konsisten seiring dengan bertambahnya epoch, mencapai nilai yang sangat rendah di akhir pelatihan. Ini menunjukkan bahwa model semakin baik dalam memprediksi label yang benar dari data pelatihan. Sementara itu, *loss* validasi (ditunjukkan dengan garis hijau) pada epoch ke-2 untuk model *Xception* dan epoch ke-8 untuk model *VGG16*, terdapat penanda biru yang menunjukkan itu sebagai "*best epoch*", di mana *loss* validasi mencapai nilai terendahnya. Ini berarti model memiliki performa terbaiknya pada epoch tersebut dalam hal mengurangi kesalahan prediksi.

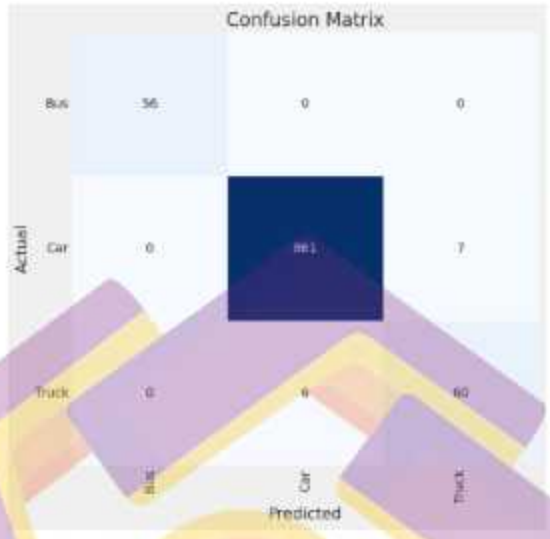
Pada grafik "*Training and Validation Accuracy*", akurasi pelatihan (garis merah) meningkat secara signifikan pada awal pelatihan dan tetap stabil di atas 98% setelah itu, menunjukkan bahwa model sangat akurat dalam mengklasifikasikan data pelatihan. Akurasi validasi (garis hijau), meskipun umumnya lebih rendah dibandingkan akurasi pelatihan, tetap berada di kisaran yang tinggi, mencapai puncaknya pada *epoch* ke-19 untuk model *Xception* dan *epoch* ke-76 untuk model *VGG16*, yang ditandai sebagai "best epoch" dengan akurasi validasi sekitar 98.99% untuk *Xception* dan 98.69% untuk *VGG16*. Fluktuasi dalam akurasi validasi menunjukkan beberapa tingkat *overfitting*, di mana model mungkin terlalu menyesuaikan diri dengan data pelatihan dan kurang generalisasi pada data yang tidak terlihat.

Kesimpulannya, grafik ini menunjukkan bahwa model memiliki performa yang sangat baik pada data pelatihan dan validasi, dengan beberapa tanda *overfitting* yang perlu diwaspadai. *Epoch* ke-19 (*Xception*) dan epoch ke-76

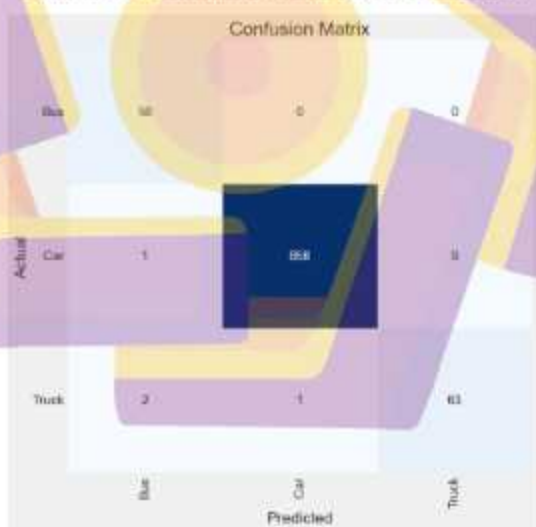
(*VGG16*) diidentifikasi sebagai titik di mana model mencapai keseimbangan terbaik antara loss dan akurasi pada data validasi, menjadikannya sebagai kandidat terbaik untuk digunakan dalam aplikasi praktis.

4.4.5 Confusion Matrix

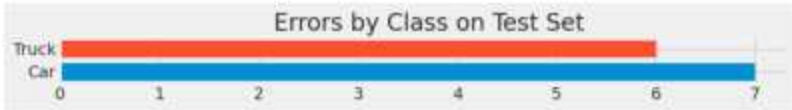
Pada tahap ini, model yang telah dilatih akan digunakan untuk membuat prediksi pada dataset uji (test set), yang terpisah dari data yang digunakan selama pelatihan dan validasi. Dengan menggunakan prediksi yang dihasilkan, kita dapat membuat *confusion matrix* yang menunjukkan seberapa baik model dapat mengklasifikasikan setiap kelas. *Confusion matrix* memberikan gambaran yang jelas tentang performa model dalam mengenali kelas-kelas yang berbeda, termasuk seberapa sering mungkin terjadi kesalahan antara kelas yang salah diklasifikasikan. Selain itu, laporan klasifikasi (*classification report*) juga disusun untuk memberikan ringkasan statistik yang lebih terperinci, termasuk *presisi*, *recall*, dan *F1-score* untuk setiap kelas. Melalui tahap ini, kita dapat mengevaluasi secara komprehensif kinerja model dan memperoleh wawasan yang berharga tentang kekuatan dan kelemahan model yang telah dikembangkan.



Gambar 4.24. Confusion Matrix Dari Model *Xception*



Gambar 4.25. Confusion Matrix Dari Model *VGG16*



Gambar 4.26. Jumlah Kesalahan pada Dataset Uji Model *Xception*

Gambar 4.27. Jumlah Kesalahan pada Dataset Uji Model *VGG16*

Gambar 4.24 menggambarkan *confusion matrix* yang mencerminkan kinerja model klasifikasi pada suatu set data. Matriks ini membandingkan prediksi yang dihasilkan oleh model dengan kenyataan sebenarnya untuk tiga kelas: Bus, Car, dan Truck. Dari matriks tersebut, dapat dilihat bahwa model telah menunjukkan tingkat akurasi yang tinggi dalam mengklasifikasikan mobil (*car*), dengan 861 prediksi yang benar dan hanya 7 kesalahan. Untuk kelas Bus, terdapat 56 prediksi benar (*true positives*) dan tidak ada kesalahan (false positives). Sementara untuk kelas Truck, terdapat 60 prediksi benar (*true positives*), namun terdapat 7 kesalahan di mana truk salah diklasifikasikan sebagai mobil. Untuk lebih jelasnya dapat dilihat pada contoh gambar 4.246

(a)

(b)

Gambar 4.28. Contoh Hasil Prediksi

(a) Prediksi Benar (b) Prediksi Salah

Analisis tersebut memberikan gambaran yang jelas tentang kinerja model dalam mengklasifikasikan setiap kelas, serta kesalahan yang terjadi dalam prosesnya. Dengan pemahaman ini, kita dapat mengidentifikasi area di mana model perlu ditingkatkan dan mengoptimalkan performanya untuk penggunaan yang lebih baik di masa depan.

4.5 Implementasi Metode Haar Cascade dan CNN

Langkah pertama dalam proses klasifikasi dan penghitungan kendaraan secara otomatis melibatkan implementasi dua metode utama: Haar Cascade dan Convolutional Neural Network (CNN).

Pertama-tama, gambar uji diambil dan dikonversi menjadi skala abu-abu untuk memfasilitasi deteksi kendaraan menggunakan metode Haar Cascade. Setelah deteksi, setiap kendaraan dipotong dari gambar asal dan diubah ukurannya menjadi 224x224 piksel. Kemudian, gambar yang telah diubah ukuran dinormalisasi dan diperiksa untuk memastikan bahwa teksturnya memenuhi kriteria yang telah ditetapkan untuk analisis lebih lanjut. Gambar yang memenuhi kriteria tersebut kemudian diproses melalui model CNN yang telah dilatih sebelumnya. Model ini memberikan prediksi dengan tingkat kepercayaan tertentu. Prediksi yang memiliki tingkat kepercayaan di atas ambang batas yang telah ditetapkan akan dipertimbangkan untuk langkah selanjutnya. Setelah prediksi diperoleh, dilakukan teknik *Non Maximum Suppression* (NMS) untuk menghilangkan kotak pembatas yang berlebihan, memastikan bahwa setiap kendaraan hanya diwakili oleh satu

kotak pembatas. Kotak pembatas yang tersisa kemudian ditampilkan pada gambar asli dengan warna yang berbeda, menunjukkan jenis kendaraan yang terdeteksi. Terakhir, sistem menghitung jumlah kendaraan dari setiap jenis yang terdeteksi dan menampilkan informasi ini pada gambar. Dengan demikian, proses ini menghasilkan identifikasi dan penghitungan kendaraan secara otomatis dari gambar yang diberikan. Gambar 4.29 merupakan hasil dari tahapan Implementasi Metode Haar Cascade dan (CNN)

Gambar 4.29. Hasil Implementasi Metode Haar Cascade dan (CNN)

4.6 Pengujian Metode *Haar Cascade* dan CNN

Setelah berhasil mengembangkan integrasi metode *Haar Cascade Classifiers* dan *Convolutional Neural Network* (CNN), pada tahapan ini metode tersebut akan diuji coba menggunakan data uji sebanyak 15 gambar yang telah disiapkan. Data uji merupakan kumpulan gambar yang diekstrak dari rekaman video *CCTV* dari laman *YouTube*. Gambar 4.30 merupakan contoh data uji yang digunakan pada tahapan ini.

Gambar 4.30. Data Pengujian

Setelah data uji disiapkan, langkah selanjutnya adalah menjalankan program yang telah dibuat. Pada proses uji coba, tingkat keberhasilan metode diukur berdasarkan tingkat akurasi, tingkat presisi, dan tingkat sensitivitas. Tingkat akurasi diukur dengan membandingkan jumlah kendaraan yang terdeteksi dan diklasifikasikan dengan benar terhadap total kendaraan yang sebenarnya ada dalam data pengujian. Presisi, di sisi lain, mengacu pada proporsi prediksi yang benar

terhadap jumlah total prediksi yang dibuat oleh sistem. Ini penting untuk memastikan bahwa sistem minim menghasilkan hasil positif palsu, yang dapat mengganggu efektivitas aplikasi nyata. Masing-masing parameter akan dihitung pada setiap data uji yang digunakan.

4.6.1 Uji Coba Metode Haar Cascade Classifiers dan CNN dengan Model Xception

Output dari metode ini berupa gambar yang telah ditandai dengan objek yang terdeteksi, dilengkapi dengan labelnya serta informasi mengenai tingkat akurasi dan estimasi jumlah kendaraannya. Sebagai contoh, Gambar 4.27 menunjukkan hasil dari penerapan metode Haar Cascade Classifiers dan Convolutional Neural Network menggunakan model Xception. Ini adalah contoh konkret bagaimana sistem ini mengintegrasikan kedua pendekatan untuk menghasilkan hasil yang akurat dan informatif dalam analisis kendaraan secara otomatis.

(a)

(b)

Gambar 4.31. Hasil Pengujian Haar Cascade dan Xception

Pada Gambar 4.31 (a) dan 4.31 (b) sistem berhasil mendeteksi dan mengklasifikasikan berbagai jenis kendaraan, termasuk mobil, truk, dan bus, dengan tingkat akurasi yang memuaskan. Kotak pembatas yang digunakan untuk menandai kendaraan terlihat jelas dan tepat mengelilingi setiap objek, menunjukkan

efektivitas algoritma *Haar Cascade* dalam deteksi objek. Namun, terdapat kelemahan dalam mendeteksi objek yang tumpang tindih, yang dapat mengurangi akurasi klasifikasi. Selain itu, terdapat kesalahan dalam mengklasifikasikan kendaraan yang berada jauh dari pandangan kamera, seperti contohnya pada gambar 4.31 (b), dimana sebuah mobil yang salah diberi label sebagai truk. Hal ini menandakan perlunya peningkatan sensitivitas dan akurasi sistem, terutama dalam situasi di mana objek tidak terlihat jelas atau saat objek saling bertumpang tindih. Pada Tabel 4.3 merupakan hasil pengujian untuk melihat tingkat keberhasilan metode.

Table 4.3. Hasil Pengujian Menggunakan Model Xception

Nama File	Total Objek	Nilai Prediksi			Aktual			Prediksi			Akurasi	Presisi	*Recall*
		TP	FP	FN	C	T	B	C	T	B			
Testing1.jpg	12	10	0	2	12	0	0	10	0	0	83.3%	100%	83.3%
Testing2.jpg	18	13	1	4	16	1	1	11	2	1	72.2%	92.9%	76.5%
Testing3.jpg	6	6	0	0	4	2	0	4	2	0	100%	100%	100%
Testing4.jpg	9	9	0	0	8	0	1	8	0	1	100%	100%	100%
Testing5.jpg	9	9	0	0	9	0	0	9	0	0	100%	100%	100%
Testing6.jpg	10	10	0	0	10	0	0	10	0	0	100%	100%	100%
Testing7.jpg	9	9	0	0	8	1	0	8	1	0	100%	100%	100%
Testing8.jpg	12	11	0	1	12	0	0	11	0	0	91.7%	100%	91.7%
Testing9.jpg	18	15	0	3	18	0	0	15	0	0	83.3%	100%	83.3%
Testing10.jpg	5	5	0	0	4	1	0	4	1	0	100%	100%	100%
Testing11.jpg	6	6	0	0	6	0	0	6	0	0	100%	100%	100%
Testing12.jpg	10	10	0	0	10	0	0	10	0	0	100%	100%	100%
Testing13.jpg	10	8	0	2	10	0	0	8	0	0	80%	100%	80%
Testing14.jpg	9	8	0	1	8	1	0	7	1	0	88.9%	100%	88.9%
Testing15.jpg	18	16	0	2	18	0	0	16	0	0	89.9%	100%	88.9%
Rata-rata											**92.6%**	**99.5%**	**92.3%**

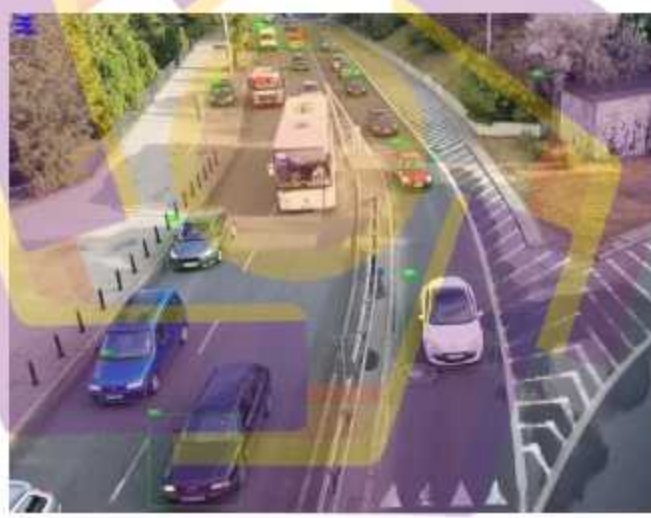
Tabel 4.3 menyajikan hasil evaluasi klasifikasi dan penghitungan kendaraan secara otomatis menggunakan metode *Haar Cascade* dan *Convolutional Neural Network* (CNN). Setiap baris mewakili sebuah gambar uji (Testing1.jpg hingga Testing15.jpg) dengan informasi tentang jumlah total objek kendaraan dalam gambar, nilai prediksi yang diberikan oleh sistem, jumlah kendaraan actual dan prediksi, serta nilai akurasi, presisi, dan recall untuk setiap gambar. Akurasi merupakan persentase dari prediksi yang benar dibandingkan dengan total objek, sementara presisi mengukur keberhasilan sistem dalam mengklasifikasikan objek yang diprediksi positif, dan *recall* mengukur seberapa baik sistem dalam menemukan semua instance objek yang sebenarnya positif.

Dari tabel tersebut, terlihat bahwa sistem memiliki tingkat akurasi yang tinggi secara keseluruhan, dengan nilai rata-rata akurasi mencapai 92.6%. Hal ini menunjukkan bahwa sebagian besar prediksi sistem sesuai dengan objek aktual yang terdapat dalam gambar. Namun, terdapat beberapa kekurangan yang perlu diperhatikan. Salah satunya adalah terjadinya kesalahan dalam prediksi beberapa jenis kendaraan, seperti mobil yang diprediksi sebagai truk dan objek yang gagal terdeteksi. Hal ini dapat disebabkan oleh beberapa faktor yang diantaranya adalah variasi dalam tampilan kendaraan, jarak objek kendaraan dari kamera, serta adanya objek yang bertumpang tindih. Meskipun demikian, nilai akurasi yang tinggi menunjukkan bahwa sistem memiliki kemampuan yang baik dalam mengklasifikasikan kendaraan secara keseluruhan, meskipun masih ada ruang

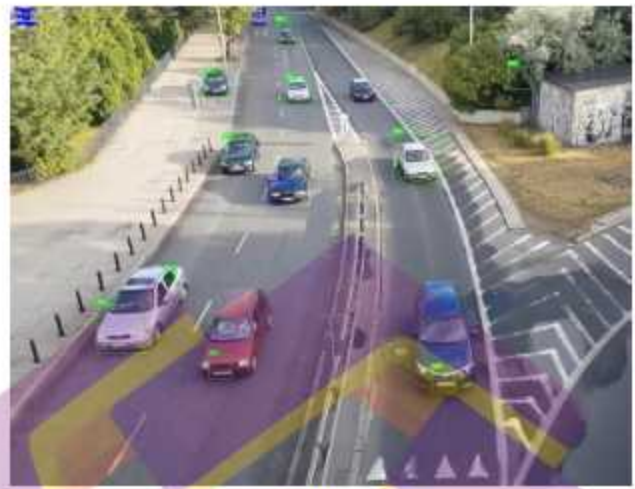
untuk perbaikan dalam mengenali dan mengklasifikasikan jenis kendaraan tertentu dengan lebih baik.

4.6.2 Uji Coba Metode Haar Cascade Classifiers dan CNN dengan Model VGG16

Output dari pengujian metode *Haar Cascade Classifiers* dan CNN dengan model VGG16 menghasilkan gambar yang menandai objek yang terdeteksi, dengan label dan informasi akurasi serta estimasi jumlah kendaraan, sebagaimana halnya dengan output dari pengujian menggunakan model Xception sebelumnya. Sebagai contoh, seperti yang terlihat pada Gambar 4.28.



(a)



(b)

Gambar 4.32. Hasil Pengujian *Haar Cascade* dan *VGG16*

Pada Gambar 4.32 (a) dan 4.32 (b), sistem berhasil mendeteksi dan mengklasifikasikan berbagai jenis kendaraan, termasuk mobil, truk, dan bus, dengan hasil yang cukup mirip dengan model sebelumnya, yaitu Xception. Namun, perlu diperhatikan bahwa meskipun CNN dengan model VGG16 memberikan kemampuan analisis yang lebih mendalam, terdapat beberapa kekurangan yang perlu dicatat. Salah satunya adalah terlihat pada Gambar 4.32 (b), dimana rumput terdeteksi sebagai mobil, dan terkadang objek mobil terdeteksi sebagai bus. Hal ini menunjukkan bahwa model *VGG16* mengalami kesulitan dalam mengenali objek dengan presisi yang tinggi, terutama saat objek memiliki kemiripan visual atau ketika terjadi kebingungan antara kelas objek yang berbeda. Meskipun demikian, penggunaan gabungan kedua metode ini masih memberikan kontribusi yang signifikan dalam meningkatkan kemampuan sistem dalam menganalisis kendaraan

secara otomatis. Pada Tabel 4.4 merupakan hasil pengujian untuk melihat tingkat keberhasilan metode.

Table 4.4. Hasil Pengujian Menggunakan Model VGG16

Nama File	Total Objek	Nilai Prediksi			Aktual			Prediksi			Akurasi	Presisi	*Recall*
		TP	FP	FN	C	T	B	C	T	B			
Testing1.jpg	12	9	0	3	12	0	0	9	0	0	75%	100%	75%
Testing2.jpg	18	11	2	6	16	1	1	10	2	1	61.1%	84.6%	64.7%
Testing3.jpg	6	6	1	0	4	2	0	5	2	0	100%	85.7%	100%
Testing4.jpg	9	9	0	0	8	0	1	8	0	1	100%	100%	100%
Testing5.jpg	9	8	1	1	9	0	0	9	0	0	88.9%	88.9%	88.9%
Testing6.jpg	10	8	3	0	10	0	0	8	2	1	80%	72.7%	100%
Testing7.jpg	9	7	2	0	8	1	0	8	1	0	77.8%	77.8%	100%
Testing8.jpg	12	10	1	1	12	0	0	10	0	1	83.3%	90.9%	90.9%
Testing9.jpg	18	12	2	4	18	0	0	12	1	1	66.7%	85.7%	75%
Testing10.jpg	5	5	0	0	4	1	0	4	1	0	100%	100%	100%
Testing11.jpg	6	6	0	0	6	0	0	6	0	0	100%	100%	100%
Testing12.jpg	10	10	0	0	10	0	0	10	0	0	100%	100%	100%
Testing13.jpg	10	8	0	2	10	0	0	8	0	0	80%	100%	80%
Testing14.jpg	9	8	0	1	8	1	0	7	1	0	88.9%	100%	88.9%
Testing15.jpg	18	16	0	2	18	0	0	16	0	0	89.9%	100%	88.9%
Rata-rata											**86%**	**92.4%**	**90%**

Berdasarkan hasil pengujian yang disajikan dalam Table 4.4, kombinasi model *Haar Cascade* dengan CNN menggunakan model *VGG16* menunjukkan performa yang baik. Rata-rata akurasi yang dicapai adalah 86%, yang menunjukkan bahwa model secara keseluruhan mampu mengklasifikasikan objek dengan benar sebanyak 86% dari total objek yang diuji. Akurasi tertinggi sebesar 100% dicapai pada beberapa pengujian (Testing3.jpg, Testing4.jpg, Testing10.jpg, Testing11.jpg dan Testing12.jpg), menunjukkan bahwa dalam beberapa kasus, model ini sangat

efektif. Selain itu, rata-rata presisi yang diperoleh adalah 92.4%, yang berarti sebagian besar prediksi positif yang diberikan oleh model adalah benar. Hal ini terlihat dari beberapa hasil pengujian yang menunjukkan presisi 100%, mengindikasikan bahwa semua prediksi positif pada gambar-gambar tersebut benar-benar merupakan objek yang sesuai. Rata-rata *recall* yang dicapai adalah 90.2%, yang menunjukkan bahwa model mampu menemukan sebagian besar objek yang sebenarnya ada di gambar. *Recall* yang tinggi sangat penting dalam aplikasi di mana menemukan semua objek yang relevan lebih kritis daripada memiliki sedikit prediksi yang salah.

Namun, terdapat beberapa kelemahan yang mungkin menyebabkan akurasi tidak mencapai nilai maksimum. Salah satu kelemahan adalah adanya nilai *False Negatives* (FN) yang cukup tinggi pada beberapa gambar, seperti Testing2.jpg dengan FN=6 dan Testing9.jpg dengan FN=4. Hal ini menunjukkan bahwa model sering kali gagal mendeteksi beberapa objek yang ada di gambar. Selain itu, nilai *False Positives* (FP) juga hadir pada beberapa gambar, seperti Testing2.jpg dengan FP=2 dan Testing6.jpg dengan FP=3, yang menunjukkan bahwa model kadang-kadang memberikan prediksi positif yang salah, sehingga menurunkan presisi model. Performa model juga sangat bervariasi tergantung pada gambar yang diuji, seperti yang terlihat pada Testing2.jpg yang memiliki akurasi dan *recall* lebih rendah dibandingkan gambar lainnya. Variasi ini bisa disebabkan oleh perbedaan dalam data uji yang mungkin tidak terwakili dengan baik dalam data pelatihan.

4.7 Analisis Hasil Optimasi

Dalam penelitian ini, kami menerapkan beberapa proses optimasi untuk meningkatkan kinerja model deteksi kendaraan. Kami menggunakan *Bayesian Optimization* untuk menemukan kombinasi hiperparameter yang optimal. *Bayesian Optimization* adalah metode optimasi berbasis probabilistik yang bertujuan untuk menemukan nilai optimal dari fungsi yang mahal untuk dievaluasi. Alasan kami menggunakan *Bayesian Optimization* adalah karena metode ini dapat secara efisien mengeksplorasi ruang parameter dan meminimalkan jumlah evaluasi yang diperlukan untuk menemukan nilai optimal. Metode ini membangun model probabilistik dari fungsi objektif dan menggunakan model tersebut untuk memilih parameter berikutnya yang akan dievaluasi. Keunggulan utama dari *Bayesian Optimization* adalah kemampuannya untuk secara efektif mengeksplorasi ruang parameter yang besar dan kompleks dengan jumlah evaluasi yang relatif sedikit, dibandingkan dengan metode optimasi lain seperti *Grid Search* atau *Random Search*. Dengan demikian, *Bayesian Optimization* mampu mengurangi jumlah eksperimen yang diperlukan untuk menemukan parameter optimal, sehingga menghemat waktu dan sumber daya komputasi. Selain itu, metode ini juga membantu mengurangi risiko *overfitting* dengan melakukan evaluasi yang lebih efisien dan menyeluruh, sehingga ruang parameter dapat dieksplorasi dengan baik. Gambar 4.29 menggambarkan proses optimasi menggunakan *Bayesian Optimization* dengan model *Xception*.

Gambar 4.33. Proses Optimasi Untuk Meningkatkan Kinerja Model

Pada gambar 4.33, kami mendefinisikan model dengan menggunakan *learning rate*, *dropout rate*, dan jumlah *filters* sebagai hiperparameter yang akan dioptimasi. Model ini dibangun menggunakan arsitektur *Xception* sebagai model dasar. Setiap iterasi dalam *Bayesian Optimization* melibatkan pelatihan model dengan kombinasi hiperparameter yang berbeda. Hasil akurasi validasi pada akhir setiap iterasi digunakan sebagai metrik untuk menilai kinerja model. Ruang parameter yang dioptimasi mencakup *learning rate* dengan rentang antara 1e-4 hingga 1e-2 (0.0001 hingga 0.01), *dropout rate* dengan rentang antara 0.1 hingga 0.5 (10% hingga 50%), dan jumlah filters (rentang antara 16 hingga 64). Proses *Bayesian Optimization* dilakukan melalui beberapa iterasi untuk menemukan kombinasi parameter yang menghasilkan kinerja terbaik. Pada setiap iterasi, model

dilatih dan divalidasi, dan hasil akurasi validasi digunakan untuk memperbarui model probabilistik dalam *Bayesian Optimization*. Table 4.5 merupakan hasil dari setiap iterasi yang dilakukan.

Table 4.5. Tabel Hasil Bayesian Optimization

Iterasi	Target	Dropout	Filters	Learning
1	0.9788	0.3786	29.73	0.002346
2	0.9798	0.3205	50.53	0.004289
3	0.9869	0.4923	48.87	0.004861
4	0.9859	0.278	45.07	0.0008031
5	0.9859	0.3297	45.13	0.01
6	0.9929	0.1897	44.99	0.0001
7	0.9707	0.4088	23.15	0.00997
8	0.9879	0.1963	44.58	0.0001
9	0.9839	0.3115	48.22	0.004782
10	0.9869	0.1	49.33	0.00305
11	0.9788	0.1018	43.6	0.008378
12	0.9788	0.5	49.58	0.01
13	0.9899	0.1248	32.03	0.004006

Setelah proses optimalisasi selesai, parameter terbaik yang ditemukan meliputi *learning rate*, *dropout rate*, dan jumlah *filters*. Berdasarkan hasil dari tabel *Bayesian Optimization*, parameter terbaik terdapat pada iterasi ke-6 dengan nilai *target* 0.9929, *dropout rate* sebesar 0.1897, jumlah *filters* sebanyak 44.99, dan *learning rate* sebesar 0.0001. Parameter ini kemudian digunakan untuk membangun dan melatih model akhir.

Model akhir dibangun menggunakan parameter optimal yang ditemukan dari proses *Bayesian Optimization* dan dilatih menggunakan data pelatihan serta divalidasi dengan data validasi selama 30 epochs. Dengan menggunakan *Bayesian*

Optimization untuk menemukan parameter yang optimal, model kami dapat mencapai kinerja yang lebih baik dalam tugas deteksi kendaraan.

4.8 Interpretasi Hasil

Dalam analisis performa model deteksi dan klasifikasi kendaraan, dua pendekatan utama yang digunakan adalah kombinasi metode *Haar Cascade Classifiers* dengan CNN menggunakan model *VGG16* dan *Xception*. Selain itu, penelitian ini juga membandingkan dua varian dari model *Xception*: satu tanpa menggunakan *Bayesian optimization* dan satu lagi dengan menggunakan *Bayesian optimization*. Pendekatan *Bayesian optimization* diterapkan untuk mengoptimalkan *hyperparameter* model, dengan tujuan untuk meningkatkan akurasi dan efisiensi klasifikasi. Perbandingan performa dari kedua model *Xception* ini, baik yang dioptimalkan maupun yang tidak, disajikan secara rinci pada Tabel 4.6 dan Table 4.7

Table 4.6. Perbandingan Performa Model Xception dan VGG16

Nama File	Total Objek	Xception			VGG16		
		Akurasi	Presisi	Recall	Akurasi	Presisi	Recall
Testing1.jpg	12	83.3%	83.3%	83.3%	75%	100%	75%
Testing2.jpg	18	72.2%	81.2%	72.2%	61.1%	84.6%	64.71%
Testing3.jpg	6	100%	100%	100%	100%	85.7%	100%
Testing4.jpg	9	100%	100%	100%	100%	100%	100%
Testing5.jpg	9	100%	100%	100%	88.8%	88.8%	88.8%
Testing6.jpg	10	100%	100%	100%	80%	72.7%	100%
Testing7.jpg	9	100%	100%	100%	77.7%	77.7%	100%
Testing8.jpg	12	91.7%	90.9%	100%	83.3%	90.9%	90.9%
Testing9.jpg	18	83.3%	83.3%	100%	66.6%	85.7%	75%

Table 4.6. (Lanjutan)

Nama File	Total Objek	Xception			VGG16		
		Akurasi	Presisi	Recall	Akurasi	Presisi	Recall
Testing10.jpg	5	100%	100%	100%	100%	100%	100%
Testing11.jpg	6	100%	100%	100%	100%	100%	100%
Testing12.jpg	10	100%	100%	100%	100%	100%	100%
Testing13.jpg	10	80%	100%	80%	80%	100%	80%
Testing14.jpg	9	88.9%	100%	88.9%	88.9%	100%	88.9%
Testing15.jpg	18	89.9%	100%	88.9%	89.9%	100%	88.9%
Rata-rata		**92.6%**	**99.5%**	**92.3%**	**86%**	**92.4%**	**90.2%**

Hasil pengujian menunjukkan bahwa model VGG16 memiliki rata-rata akurasi sebesar 86%, presisi 92.4%, dan *recall* 90.2%. Meskipun performa keseluruhan baik, model ini memiliki kelemahan dalam mendeteksi beberapa objek (*False Negatives*) dan memberikan beberapa prediksi positif yang salah (*False Positives*). Variabilitas performa yang signifikan tergantung pada gambar yang diuji menunjukkan bahwa data pelatihan mungkin tidak cukup mewakili berbagai kondisi di data uji. Selain itu, kompleksitas model VGG16 meningkatkan potensi *overfitting* jika data pelatihan tidak cukup beragam.

Model *Xception* menunjukkan performa yang lebih tinggi dengan rata-rata akurasi sebesar 92.6%, presisi 99.5%, dan recall 92.3%. Tingginya nilai akurasi, presisi, dan recall pada model *Xception* menunjukkan kemampuan yang lebih baik dalam mendeteksi dan mengklasifikasikan kendaraan secara keseluruhan. Kesalahan prediksi dari model *Xception* lebih rendah dibandingkan dengan *VGG16*, menunjukkan bahwa *Xception* lebih efektif dalam mengurangi kesalahan klasifikasi

dan meningkatkan akurasi deteksi objek. Kesimpulannya, meskipun kedua model menunjukkan performa yang baik, model *Xception* lebih unggul dibandingkan *VGG16*. Untuk meningkatkan performa lebih lanjut, perlu dilakukan perluasan dan variasi *dataset* pelatihan serta optimasi model untuk mengurangi kesalahan klasifikasi dan meningkatkan generalisasi, terutama dalam situasi dengan objek yang bertumpang tindih atau jauh dari kamera.

Selain membandingkan performa antara model VGG16 dan Xception, penelitian ini juga mengevaluasi perbedaan performa antara model *Xception* tanpa optimasi *hyperparameter* dan *Xception* yang menggunakan *Bayesian optimization*. Hasil pengujian menunjukkan bahwa *Xception* yang dioptimalkan memiliki keunggulan performa dibandingkan dengan versi yang tidak dioptimalkan. Sebagai contoh, rata-rata akurasi untuk *Xception* tanpa optimasi adalah 92.6%, sementara *Xception* dengan *Bayesian optimization* mencapai rata-rata akurasi 95.3%. Begitu pula, presisi meningkat dari 99.5% menjadi 99.3%, dan recall dari 92.3% menjadi 95.3%. Data ini menunjukkan bahwa optimasi Bayesian berhasil meningkatkan kemampuan model dalam mendeteksi dan mengklasifikasikan objek kendaraan, mengurangi kesalahan prediksi, dan memperbaiki generalisasi. Perbaikan signifikan ini menggarisbawahi pentingnya optimasi hyperparameter dalam mencapai performa model yang optimal, terutama dalam aplikasi klasifikasi gambar kompleks seperti ini. Hasil perbandingan rinci dari kedua model Xception dapat dilihat pada Tabel 4. 8, yang menunjukkan perbedaan akurasi, presisi, dan recall untuk setiap gambar uji.

Table 4.7. Evaluasi Performa Xception Optiasi Hyperparameter

Nama File	Total Objek	Xception			Xception (*optimization*)		
		Akurasi	Presisi	Recall	Akurasi	Presisi	Recall
Testing1.jpg	12	83.3%	83.3%	83.3%	91.7%	100%	91.7%
Testing2.jpg	18	72.2%	81.2%	72.2%	77.8%	100%	77.8%
Testing3.jpg	6	100%	100%	100%	100%	100%	100%
Testing4.jpg	9	100%	100%	100%	100%	100%	100%
Testing5.jpg	9	100%	100%	100%	100%	100%	100%
Testing6.jpg	10	100%	100%	100%	100%	100%	100%
Testing7.jpg	9	100%	100%	100%	100%	100%	100%
Testing8.jpg	12	91.7%	90.9%	100%	91.7%	100%	91.7%
Testing9.jpg	18	83.3%	83.3%	100%	94.4%	94.4%	94.4%
Testing10.jpg	5	100%	100%	100%	100%	100%	100%
Testing11.jpg	6	100%	100%	100%	100%	100%	100%
Testing12.jpg	10	100%	100%	100%	100%	100%	100%
Testing13.jpg	10	80%	100%	80%	90.0%	100%	90.0%
Testing14.jpg	9	88.9%	100%	88.9%	88.9%	100%	88.9%
Testing15.jpg	18	88.9%	100%	88.9%	94.4%	94.4%	94.4%
Rata-rata		**92.6%**	**99.5%**	**92.3%**	**95.3%**	**99.3%**	**95.3%**

Peningkatan performa yang diamati pada model *Xception* setelah penerapan *Bayesian optimization* dapat dijelaskan oleh peningkatan dalam penyesuaian hyperparameter yang tepat, yang merupakan elemen kunci dalam memaksimalkan efektivitas model. *Bayesian optimization* memungkinkan pencarian *hyperparameter* terbaik dengan cara yang lebih sistematis dan efisien, dibandingkan dengan metode manual yang memakan waktu dan sumber daya lebih besar. *Hyperparameter* seperti tingkat pembelajaran (*learning rate*), ukuran batch (*Bath Size*), jumlah layer (*filters*), dan tingkat *dropout* sangat mempengaruhi

kemampuan model untuk belajar dari data pelatihan dan menggeneralisasi pada data yang tidak terlihat.

Dalam konteks model *Xception* yang digunakan untuk deteksi dan klasifikasi kendaraan, peningkatan ini berarti model menjadi lebih sensitif dan tepat dalam mengenali fitur-fitur penting yang membedakan kelas-kelas kendaraan, seperti mobil, truk, dan bus. Hasilnya adalah peningkatan akurasi, presisi, dan recall, dengan penurunan jumlah kesalahan klasifikasi, seperti *false positives* (misalnya, rumput yang salah terdeteksi sebagai kendaraan) dan *false negatives* (misalnya, kendaraan yang tidak terdeteksi). Dengan *hyperparameter* yang dioptimalkan, model *Xception* dapat lebih baik mengatasi variasi dalam data uji, termasuk variasi dalam pencahayaan, sudut pandang, dan kondisi lingkungan, yang semuanya merupakan tantangan umum dalam aplikasi dunia nyata.

Dengan kata lain, *Bayesian optimization* membantu model *Xception* untuk mempelajari representasi fitur yang lebih baik, meningkatkan kemampuannya untuk membuat prediksi yang lebih akurat dan handal. Ini menjelaskan mengapa terjadi peningkatan signifikan dalam kinerja model setelah penerapan teknik optimasi ini.

Pada tahap akhir, penelitian ini membandingkan hasil dari model *Xception* yang digunakan dalam deteksi dan klasifikasi kendaraan dengan hasil dari penelitian sebelumnya oleh (Irawanto et al., 2023). Pada Table 4.9 merupakan hasil dari perbandingannya.

Table 4.8. Hasil Perbandingan Penelitian Sebelumnya

Penelitian	Model	Akurasi	Presisi	Recall
(Irawanto et al., 2023)	*Haar cascade* & CNN	86%	100%	86%
Proposed Method	***Xception & optimization***	**95.3%**	**99.3%**	**95.3%**

Penelitian Irawanto et al., (2023) yang berfokus pada deteksi mobil pribadi menggunakan kombinasi *CNN* dan *Haar Cascade Classifier* menghasilkan rata-rata akurasi sebesar 86%, dengan presisi mencapai 100% dan *recall* sebesar 86%. Sementara itu, penelitian ini menunjukkan bahwa model *Xception* memiliki performa yang lebih unggul dengan rata-rata akurasi sebesar 95.3%, presisi 99.3%, dan *recall* 95.3%. Tabel perbandingan menunjukkan bahwa model *Xception* tidak hanya mampu mendeteksi kendaraan dengan akurasi yang lebih tinggi tetapi juga mampu mengklasifikasikan berbagai jenis kendaraan dengan lebih efektif dibandingkan metode sebelumnya yang hanya fokus pada mobil pribadi. Perbedaan ini mengindikasikan bahwa penggunaan model *Xception* memberikan peningkatan signifikan dalam hal akurasi dan generalisasi deteksi kendaraan, yang sangat penting untuk aplikasi pengelolaan lalu lintas yang lebih kompleks dan dinamis. Penelitian ini juga menegaskan perlunya pengembangan lebih lanjut untuk mengatasi tantangan dalam mendeteksi objek kendaraan yang berdekatan atau yang berada jauh dari pandangan kamera, serta pentingnya memperluas jangkauan deteksi untuk mencakup berbagai jenis kendaraan demi meningkatkan efektivitas sistem deteksi dan klasifikasi secara keseluruhan.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah melakukan eksperimen terkait Optimasi Performa Deteksi Kendaraan Dengan Menggabungkan Algoritma *Haar Cascade Classifier* dan *Convolutional Neural Network* (CNN). Berdasarkan hasil penelitian dan analisis data, kesimpulan yang diperoleh adalah sebagai berikut:

1. Penggunaan arsitektur CNN, khususnya model *Xception*, menghasilkan performa deteksi dan klasifikasi kendaraan yang tinggi. Hasil pengujian menunjukkan bahwa model *Xception* mencapai rata-rata akurasi sebesar 92.6%, presisi 99.5%, dan recall 92.3%.
2. Dibandingkan dengan model *VGG16*, yang menunjukkan rata-rata akurasi 86%, presisi 92.4%, dan recall 90.2%, *Xception* terbukti lebih efektif dalam mendeteksi dan mengklasifikasikan berbagai jenis kendaraan, termasuk mobil, truk, dan bus.
3. Kombinasi metode *Haar Cascade Classifier* dan CNN berhasil meningkatkan performa deteksi dan klasifikasi kendaraan. Metode ini menggabungkan kekuatan Haar Cascade dalam deteksi objek dengan kemampuan CNN dalam klasifikasi yang lebih mendalam dan akurat.
4. Model *Xception* yang dioptimalkan menggunakan *Bayesian optimization* menunjukkan performa yang lebih unggul, dengan peningkatan rata-rata akurasi menjadi 95.3%, presisi 99.3%, dan recall 95.3%. Optimasi ini membantu dalam pemilihan *hyperparameter* yang

lebih optimal, sehingga model menjadi lebih akurat dan efisien dalam mendeteksi dan mengklasifikasikan kendaraan.

5. Model *Xception* dengan *Bayesian optimization* juga menunjukkan peningkatan performa dibandingkan dengan penelitian sebelumnya yang menggunakan kombinasi CNN dan *Haar Cascade Classifier*, yang mencapai akurasi 86%, presisi 100%, dan recall 86% (Irawanto et al., 2023).

5.2 Saran

Berdasarkan kesimpulan yang dipaparkan, penelitian ini masih memiliki kekurangan dan kendala, maka akan lebih baik pada penelitian selanjutnya dapat memperbaiki kekurangan yang ada. Berikut beberapa saran untuk penelitian selanjutnya:

1. Disarankan untuk memperluas dataset yang digunakan dalam pelatihan model dengan mencakup lebih banyak variasi kondisi lalu lintas, waktu, dan cuaca. Ini akan membantu model untuk lebih robust dan akurat dalam berbagai situasi nyata.
2. Untuk penelitian selanjutnya, disarankan untuk memperluas jangkauan deteksi mencakup berbagai jenis kendaraan selain mobil, truk, dan bus, serta meningkatkan sensitivitas sistem dalam kondisi yang lebih kompleks, seperti deteksi objek yang tumpang tindih dan objek yang berada jauh dari pandangan kamera.

3. Penelitian masa depan juga dapat fokus pada pengujian dalam situasi lalu lintas yang lebih ekstrem untuk menguji batas kemampuan sistem yang telah dikembangkan.
4. Penelitian lebih lanjut bisa difokuskan pada pengembangan algoritma yang lebih efektif dalam menangani objek yang bertumpang tindih, seperti menggunakan metode *Region Proposal Networks* (RPN) atau mengintegrasikan teknik segmentasi semantik.

DAFTAR PUSTAKA

Alghamdi, A. S., Saeed, A., Kamran, M., Mursi, K. T., & Almukadi, W. S. (2023a). Vehicle Classification Using Deep Feature Fusion and Genetic Algorithms. *Electronics (Switzerland)*, *12*(2). https://doi.org/10.3390/electronics12020280

Alghamdi, A. S., Saeed, A., Kamran, M., Mursi, K. T., & Almukadi, W. S. (2023b). Vehicle Classification Using Deep Feature Fusion and Genetic Algorithms. *Electronics (Switzerland)*, *12*(2). https://doi.org/10.3390/electronics12020280

Avianto, D., Harjoko, A., & Afiahayati. (2022). CNN-Based Classification for Highly Similar Vehicle Model Using Multi-Task Learning. *Journal of Imaging*, *8*(11). https://doi.org/10.3390/jimaging8110293

Butt, M. A., Khattak, A. M., Shafique, S., Hayat, B., Abid, S., Kim, K. Il, Ayub, M. W., Sajid, A., & Adnan, A. (2021a). Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems. *Complexity*, *2021*. https://doi.org/10.1155/2021/6644861

Butt, M. A., Khattak, A. M., Shafique, S., Hayat, B., Abid, S., Kim, K. Il, Ayub, M. W., Sajid, A., & Adnan, A. (2021b). Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems. *Complexity*, *2021*. https://doi.org/10.1155/2021/6644861

Chauhan, M. S., Singh, A., Khemka, M., Prateek, A., & Sen, R. (2019, January 4). Embedded CNN based vehicle classification and counting in non-laned road

traffic. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3287098.3287118

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Darmawan, R. (2022). *Perancangan Sistem Absensi menggunakan Face Recognition dengan Haar Cascade Classifier* (Vol. 5, Issue 2).

Dong, Z., Wu, Y., Pei, M., & Jia, Y. (2015a). Vehicle Type Classification Using a Semisupervised Convolutional Neural Network. *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 2247–2256. https://doi.org/10.1109/TITS.2015.2402438

Dong, Z., Wu, Y., Pei, M., & Jia, Y. (2015b). Vehicle Type Classification Using a Semisupervised Convolutional Neural Network. *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 2247–2256. https://doi.org/10.1109/TITS.2015.2402438

Elshewey, A. M., Shams, M. Y., El-Rashidy, N., Elhady, A. M., Shohieb, S. M., & Tarek, Z. (2023). Bayesian Optimization with Support Vector Machine Model for Parkinson Disease Classification. *Sensors*, *23*(4). https://doi.org/10.3390/s23042085

Fajri, R. G., Santoso, I., Alvin, Y., & Soetrisno, A. (2020). PERANCANGAN PROGRAM PENDETEKSI DAN PENGKLASIFIKASI JENIS KENDARAAN DENGAN METODE CONVOLUTIONAL NEURAL

NETWORK (CNN) DEEP LEARNING. In *TRANSIENT* (Vol. 9, Issue 1). https://ejournal3.undip.ac.id/index.php/transient

Gowri, S., Surendran, R., Divya Bharathi, M., & Jabez, J. (2022). Improved Sentimental Analysis to the Movie Reviews using Naive Bayes Classifier. *Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022, May*, 1831–1836. https://doi.org/10.1109/ICEARS53579.2022.9752408

Hadriansa, & Kristian, Y. (2015). AUTOMATIC HEAD ROTATING SYSTEM PADA DIGITAL PET MEMANFAATKAN FACE DETECTION. *IDeaTech*.

Hafram, St. M., & Asrib, A. R. (2022). Traffic Conditions and Characteristics: Investigation of Road Segment Performance. *International Journal of Environment, Engineering and Education*, *4*(3), 108–114. https://doi.org/10.55151/ijeedu.v4i3.77

Hasanah, M., Qorik, G., Pratamasunu, O., Pawening, R. E., Jadid, U. N., Zaini Mun'im Karanganyar, J. K., & Probolinggo, P. (2021). Automatic Car Detection Using Haar Cascade Classifier and Convolutional Neural Network for Traffic Density Estimation. *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, *4*(1), 11–18. https://doi.org/10.24014/ijaidm.v4i1.10785

Irawanto, I., Sunyoto, A., & Setiaji, B. (2023). Deep Learning Based Car Detection System Using Convolutional Neural Network and Haar Cascade Classifier. *2023 6th International Conference on Information and Communications*

Technology, ICOIACT 2023, 389–394. https://doi.org/10.1109/ICOIACT59844.2023.10455934

Jahan, N., Islam, S., & Foysal, M. F. A. (2020a, July 1). Real-Time Vehicle Classification Using CNN. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. https://doi.org/10.1109/ICCCNT49239.2020.9225623

Jahan, N., Islam, S., & Foysal, M. F. A. (2020b, July 1). Real-Time Vehicle Classification Using CNN. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. https://doi.org/10.1109/ICCCNT49239.2020.9225623

Kenda, P., & Witanti, A. (2021). Sistem Presensi Berbasis Wajah Dengan Metode Haar Cascade. *Konvergensi Teknologi Dan Sistem Informasi*.

Khalifa, O. O., Wajdi, M. H., Saeed, R. A., Hashim, A. H. A., Ahmed, M. Z., & Ali, E. S. (2022). Vehicle Detection for Vision-Based Intelligent Transportation Systems Using Convolutional Neural Network Algorithm. In *Journal of Advanced Transportation* (Vol. 2022). Hindawi Limited. https://doi.org/10.1155/2022/9189600

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*(8), 5455–5516. https://doi.org/10.1007/s10462-020-09825-6

Khan, S., Rahmani, H., Afaq, S., Shah, A., & Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision. *Morgan & Claypool Publishers*.

Mahmudi, A., & Rusda, M. T. (2014). DETEKSI SENJATA TAJAM DENGAN METODE HAAR CASCADE CLASSIFIER MENGGUNAKAN TEKNOLOGI SMS GATEWAY. *MATICS*, 27–30.

Maiga, B., Dalveren, Y., Kara, A., & Derawi, M. (2023). Convolutional Neural Network-Based Vehicle Classification in Low-Quality Imaging Conditions for Internet of Things Devices. *Sustainability*, *15*(23), 16292. https://doi.org/10.3390/su152316292

Manju D, Seetha M, & Sammulal P. (2021). EARLY ACTION PREDICTION USING VGG16 MODEL AND BIDIRECTIONAL LSTM. *IT in Industry*, *9*(1), 2021.

Nurhikmat, T. (2018). *IMPLEMENTASI DEEP LEARNING UNTUK IMAGE CLASSIFICATION MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN) PADA CITRA WAYANG GOLEK*.

Oliva, A., & Torralba, A. (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope *. In *International Journal of Computer Vision* (Vol. 42, Issue 3).

Prastyo, P. H., Sumi, A. S., Dian, A. W., & Permanasari, A. E. (2020). Tweets Responding to the Indonesian Government's Handling of COVID-19: Sentiment Analysis Using SVM with Normalized Poly Kernel. *Journal of Information Systems Engineering and Business Intelligence*, *6*(2), 112. https://doi.org/10.20473/jisebi.6.2.112-122

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Published as a Conference Paper*. http://arxiv.org/abs/1409.1556

Singh Bhatia, M., Aggarwal, A., & Kumar, N. (2020). Smart Traffic Light System to Control Traffic Congestion PJAEE, 17 (9) (2020) Smart Traffic Light System to Control Traffic Congestion. *Palarch's Journal Of Archaeology Of Egypt/Egyptology*.

Suryati, E., Styawati, & Ari Aldino, A. (2023). Analisis Sentimen Transportasi Online Menggunakan Ekstraksi Fitur Model Word2vec Text Embedding Dan Algoritma Support Vector Machine (SVM). *Jurnal Teknologi Dan Sistem Informasi*, *4*(1), 96–106.

Syarif, M. (2015). *DETEKSI KEDIPAN MATA DENGAN HAAR CASCADE CLASSIFIER DAN CONTOUR UNTUK PASSWORD LOGIN SISTEM* (Vol. 14, Issue 4).

Triatmoko, H. A., Pramono, H. S., & Dachlan, S. H. (2014). Penggunaan Metode Viola-Jonesdan Algoritma Eigen Eyes dalam Sistem Kehadiran Pegawai. *Jurnal EECCIS* .

Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION* .

Vyshnavi, P., Suvarna, C. S., & Yadav, K. Y. (2022). Implementation of Attendance Marking System using Haar Cascade Classifier and Convolutional

Neural Network. In *International Journal of Creative Research Thoughts* (Vol. 10, Issue 6). www.ijcrt.org

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In *Insights into Imaging* (Vol. 9, Issue 4, pp. 611–629). Springer Verlag. https://doi.org/10.1007/s13244-018-0639-9

Yang, L., Luo, P., Loy, C. C., & Tang, X. (2015). A Large-Scale Car Dataset for Fine-Grained Categorization and Verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. http://arxiv.org/abs/1506.08959