BAB I

PENDAHULUAN

1.1 Latar Belakang

Kecelakaan lalu lintas merupakan permasalahan serius yang mengancam keselamatan publik di Indonesia. Sejumlah besar kecelakaan terjadi karena faktor kelalaian pengguna jalan itu sendiri. Data statistik menunjukkan bahwa 61% penyebab kecelakaan terbesar disebabkan oleh kemampuan dan karakter pengemudi, 9% disebabkan oleh faktor kendaraan terkait dengan pemenuhan persyaratan teknik laik jalan, dan 30% disebabkan oleh faktor prasarana dan lingkungan (Marolli, 2017). Berdasarkan data dari Kepolisian Republik Indonesia, rata-rata jumlah kecelakaan lalu lintas di Indonesia sejak tahun 2019 hingga 2022 melebihi angka 100.000 setiap tahunnya. Pada tahun 2023, angka kecelakaan lalu lintas di Indonesia mencapai 148.575 kasus, mengalami kenaikan 6,6% dari tahun sebelumnya yang mencatat 139.364 kasus. Tren peningkatan ini mengindikasikan perlunya intervensi teknologi yang efektif untuk meningkatkan keselamatan berkendara, khususnya dalam situasi dengan visibilitas rendah.

Kemajuan pesat dalam kecerdasan buatan dan teknologi otomasi telah membuka peluang baru dalam pengembangan sistem transportasi yang lebih aman dan efisien. Teknologi pendeteksi lajur telah menjadi landasan dalam bidang kendaraan *autonomous*, berfungsi sebagai aspek fundamental dari sistem navigasi dan kontrol otomatis. Deteksi lajur memainkan peran kritis dalam menggambarkan batas-batas jalan, merencanakan rute navigasi, dan mengeluarkan peringatan keluar lajur (*lane departure warning*). Teknologi ini merupakan bagian integral dari sistem kendaraan *autonomous*, yang membantu dalam pengambilan keputusan penting dalam mengemudi dan meningkatkan keselamatan kendaraan dalam *Advanced Driver Assistance Systems* (ADAS) (Zhang et al., 2024).

Peningkatan arus lalu lintas, keselamatan berkendara, kenyamanan pengemudi, efisiensi transportasi, dan pengurangan kecelakaan di jalan raya merupakan target utama dari kota cerdas dan sistem transportasi cerdas di masa depan. Kendaraan *autonomous* dan tanpa pengemudi yang melakukan perencanaan jalur, pengambilan keputusan, persepsi, dan kontrol secara cerdas dan andal sangat penting untuk mencapai tujuan-tujuan ini. Dalam beberapa tahun terakhir, pengembangan metode untuk perencanaan gerak, persepsi, penginderaan, dan algoritma pengambilan keputusan sistem otonom telah mendapatkan banyak perhatian karena kemajuan yang luar biasa dalam kecerdasan buatan serta teknologi perangkat lunak dan perangkat keras.

Meskipun teknologi deteksi lajur telah berkembang pesat, masih terdapat tantangan signifikan dalam operasi pada kondisi cahaya rendah, khususnya di malam hari. Karakteristik citra lajur jalan pada malam hari menunjukkan kompleksitas teknis yang unik dan menantang. Garis lajur putih dalam kondisi cahaya rendah

mengalami penurunan kontras hingga 60-70% dibandingkan kondisi siang hari, dengan nilai piksel rata-rata turun dari 220-255 menjadi 150-180 pada skala grayscale (He et al., 2019). Fenomena ini diperparah oleh *noise gaussian* yang meningkat 3-5 kali lipat akibat *gain* tinggi pada sensor CMOS, yang secara signifikan mengurangi rasio *signal-to-noise* (SNR) dan mengakibatkan degradasi kualitas citra yang serius (Özcan et al., 2023).

Penelitian terbaru mengungkapkan bahwa model segmentasi konvensional mengalami penurunan performa yang dramatis pada kondisi malam hari. Ma et al. (2024) melaporkan bahwa akurasi segmentasi melesat turun sebesar 32.7% ketika model diuji pada dataset kondisi malam hari dibandingkan siang hari. Sistem kamera ADAS generasi terkini menunjukkan keterbatasan kritis dalam operasi malam hari, di mana intensitas cahaya di bawah 1 lux (kondisi jalan perkotaan malam hari) menyebabkan penurunan akurasi deteksi lajur hingga 48.2% (He et al., 2019). Lebih mengkhawatirkan lagi, sistem pengereman darurat otomatis (*Automatic Emergency Braking*/AEB) menunjukkan efektivitas hanya 19.3% pada kondisi malam hari berdasarkan uji coba di jalan tol (Ma et al., 2024). Temuan ini konsisten dengan laporan NHTSA (2023) yang menyatakan bahwa 78% kecelakaan fatal malam hari melibatkan kegagalan sistem ADAS. Keterbatasan ini mengidentifikasi kebutuhan mendesak untuk mengembangkan algoritma deteksi lajur yang *robust* dan adaptif terhadap variasi kondisi pencahayaan.

Untuk mengatasi tantangan deteksi lajur pada kondisi malam hari, penelitian terkini mengeksplorasi pendekatan berbasis *deep learning* yang menggabungkan arsitektur segmentasi semantik canggih dengan modul peningkatan kualitas citra *low-light*. Pengembangan terbaru arsitektur *DeepLabv3+* dengan modul *Multi-scale Feature Extraction Enhancement* menunjukkan hasil yang menjanjikan, dengan peningkatan akurasi segmentasi lajur malam hari hingga 88.22% mIoU (*mean Intersection over Union*) pada dataset kompleks (Ma et al., 2024). Implementasi *Convolutional Block Weighted Attention Module* (CBWAM) berhasil mengurangi *false positive* pada kondisi cahaya rendah sebesar 41.7% melalui mekanisme *attention spatial-channel* terpadu yang mengoptimalkan fokus model pada fitur-fitur relevan.

Namun demikian, studi menunjukkan bahwa masih terdapat batasan dalam generalisasi model pada variasi intensitas cahaya ekstrem. Penelitian Zhang et al. (2024) mengungkapkan bahwa model ini mengalami fluktuasi akurasi mencapai ±15.3% ketika dihadapkan pada variasi intensitas cahaya yang ekstrem, mengindikasikan perlunya peningkatan lebih lanjut dalam *robustness* model. Pengamatan ini menunjukkan bahwa meskipun arsitektur *DeepLabv3+* telah memberikan kontribusi signifikan, masih diperlukan penelitian tambahan untuk meningkatkan stabilitas dan konsistensi performa pada berbagai kondisi pencahayaan yang ekstrem.

Untuk mengatasi keterbatasan tersebut, pendekatan *Fused Low-Light Enhancement Network* (FLLENet) menawarkan solusi inovatif yang menggabungkan peningkatan kualitas citra dengan segmentasi semantik secara terintegrasi. FLLENet berhasil mencapai efisiensi komputasi yang luar biasa dengan *parameter* hanya 25K, jauh lebih ringan dibandingkan metode *enhancement* konvensional yang mencapai

500K+ *parameter* (Özcan et al., 2023). Arsitektur ini menggabungkan *pixel-wise scene adaptation* dan *denoising* berbasis *channel attention*, menghasilkan peningkatan PSNR (*Peak Signal-to-Noise Ratio*) sebesar 8.6 dB pada dataset gelap ekstrem. Dibandingkan dengan metode *style-transfer* GANs yang memerlukan 10.2M *parameter* namun hanya memberikan peningkatan akurasi 5.8% pada kondisi serupa, FLLENet menunjukkan keseimbangan superior antara efisiensi komputasi dan kualitas *enhancement* (T. Liu et al., 2020).

Keunggulan FLLENet dalam hal *lightweight architecture* menjadikannya sangat cocok untuk implementasi pada *embedded systems* dan perangkat ADAS dengan keterbatasan komputasi. Pendekatan ini menunjukkan bahwa dengan desain arsitektur yang cerdas dan pemanfaatan *attention mechanisms* yang tepat, adalah mungkin untuk mencapai peningkatan performa signifikan tanpa menambah beban komputasi yang berlebihan.

Penelitian ini diberi judul "Sistem Deteksi Lajur Jalan Pada Malam Hari Menggunakan *DeepLabv3+* Pada Sistem *Autonomous Car*" untuk memperluas penelitian terdahulu yang dilakukan oleh A. Ichsan Mudatsir. Penelitian sebelumnya mengidentifikasi bahwa sistem masih memperlihatkan kesalahan segmentasi yang signifikan yang diakibatkan oleh lajur jalan yang tidak terlihat jelas oleh kamera karena kurangnya cahaya pada kondisi malam hari. Kondisi ini menyebabkan sistem kesulitan dalam mengidentifikasi batas lajur dengan tepat, dengan akurasi yang masih jauh dari standar yang diperlukan untuk aplikasi ADAS yang aman (Mudatsir, 2024).

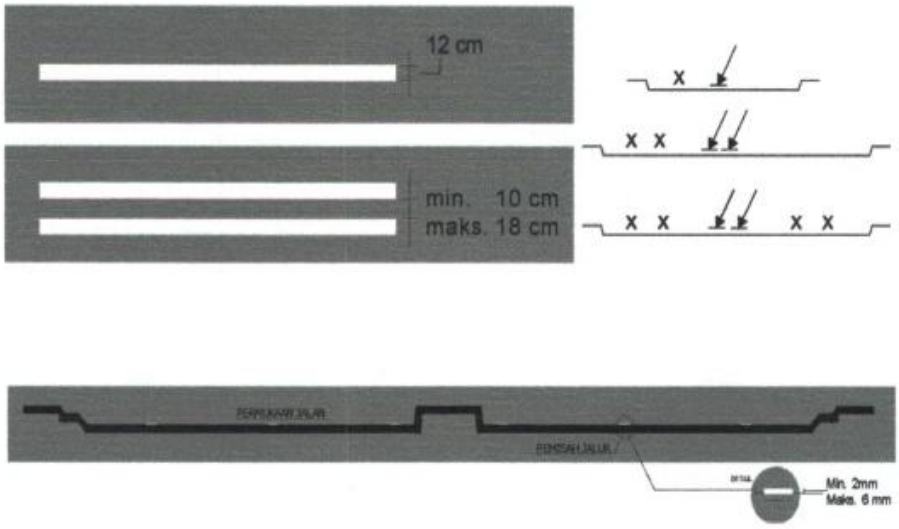
Kontribusi penelitian ini terletak pada pengintegrasian dua elemen kunci untuk meningkatkan *robustness* deteksi lajur pada malam hari: (1) penggunaan arsitektur *DeepLabv3+* yang telah terbukti efektif dalam segmentasi semantik, dan (2) integrasi FLLENet sebagai modul peningkatan kualitas citra *low-light* yang efisien.

1.2 Landasan Teori

1.2.1 Marka jalan

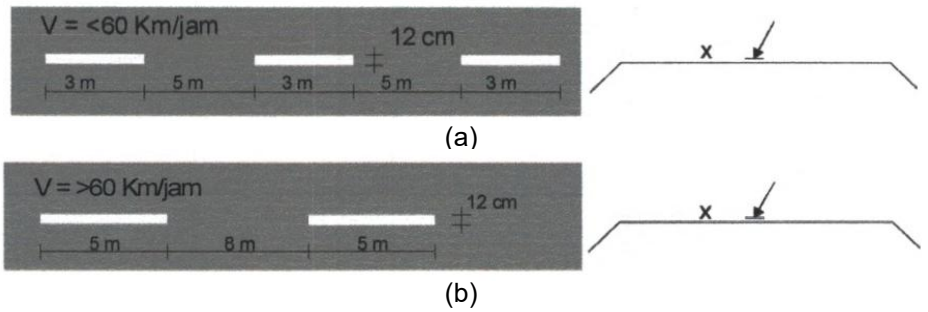
Marka jalan merupakan suatu tanda yang terdapat pada permukaan jalan atau di atas permukaan jalan yang terdiri dari peralatan atau tanda yang membentuk garis membujur, garis melintang, atau garis serong serta lambang yang dimaksudkan untuk mengarahkan arus lalu lintas dan membatasi area yang penting bagi lalu lintas (Kementerian Perhubungan Republik Indonesia, 2014). Berdasarkan Peraturan Menteri Perhubungan Republik Indonesia Nomor 34 Tahun 2014 yang telah diperbarui menjadi Peraturan Menteri Perhubungan Nomor 67 Tahun 2018, marka jalan diklasifikasikan menjadi beberapa jenis yaitu marka membujur, marka melintang, marka serong, marka lambang, dan marka lainnya.

Marka membujur adalah marka jalan yang sejajar dengan sumbu jalan, terdiri atas garis utuh, garis putus-putus, garis ganda yang terdiri dari garis utuh dan garis putus-putus, serta garis ganda yang terdiri dari dua garis utuh. Marka membujur berwarna putih dan kuning, dengan ketebalan paling rendah 2 (dua) milimeter dan paling tinggi 30 (tiga puluh) milimeter di atas permukaan jalan. Marka melintang berupa garis utuh yang digunakan sebagai batas berhenti pada persimpangan yang dilengkapi dengan rambu dan/atau alat pemberi isyarat lalu lintas, sedangkan marka melintang berupa garis putus-putus berfungsi untuk menyatakan batas yang tidak dapat dilampaui kendaraan sewaktu memberi kesempatan kepada kendaraan yang mendapat hak utama pada persimpangan (Kementerian Perhubungan Republik Indonesia, 2018).



Gambar 1. Garis membujur utuh
Sumber: (Kementerian Perhubungan Republik Indonesia, 2018)

Garis membujur utuh. Garis membujur utuh adalah marka garis yang bersifat terus menerus tanpa putus, berfungsi sebagai garis larangan yang mengindikasikan pengemudi tidak diizinkan untuk menyilang atau pindah jalur. Garis utuh digunakan pada tempat-tempat khusus atau area tikungan dengan jarak pandang yang kurang memadai untuk memberikan peringatan kepada pengemudi tentang bahaya menyiap (Kementerian Perhubungan Republik Indonesia, 2018) Warna garis utuh bervariasi tergantung fungsi dan lokasi: pada jalan nasional, garis utuh kuning berfungsi sebagai peringatan tanda tepi jalur sisi kanan, sedangkan garis utuh putih berfungsi sebagai peringatan tanda tepi jalur sisi kiri. Spesifikasi teknis garis utuh meliputi lebar 12 cm dengan panjang segmen minimal 2 mm dan maksimal 6 mm untuk konsistensi visual.



Gambar 2. Garis membujur putus: (a) Jalan 2 jalur; (b) Jalan lebih 2 jalur
Sumber: (Kementerian Perhubungan Republik Indonesia, 2018)

Garis membujur putus. Garis putus-putus adalah marka garis yang terdiri dari segmen-segmen garis yang terputus secara berkala, berfungsi sebagai garis pemisah atau pembagi lajur yang memungkinkan pengemudi untuk mengubah jalur atau lajur dengan aman. Menurut standar yang berlaku, garis putus-putus memiliki pola dengan panjang garis dan jarak putus yang ditentukan berdasarkan kecepatan lalu lintas dan fungsi garis (Kementerian Perhubungan Republik Indonesia, 2018). Pada jalan nasional dengan kecepatan 60 km/jam, pola garis putus-putus untuk pembagi lajur adalah 3 meter garis dan 5 meter putus atau 5 meter garis dan 8 meter putus untuk jalan dengan lebih dari dua jalur. Warna garis putus-putus kuning pada jalan nasional berfungsi sebagai pembatas dan pembagi jalur, sedangkan warna putih berfungsi sebagai pembagi lajur (Kementerian Perhubungan Republik Indonesia, 2018).

Garis kuning. Garis kuning adalah marka garis berwarna kuning yang khusus digunakan pada jalan nasional untuk memberikan identifikasi khusus dan meningkatkan visibilitas marking dalam berbagai kondisi pencahayaan. Menurut Peraturan Menteri Perhubungan Nomor 67 Tahun 2018, garis kuning dapat berupa garis utuh atau garis putus-putus, dengan fungsi spesifik: garis utuh kuning berfungsi sebagai peringatan tanda tepi jalur di sisi kanan jalan nasional (Kementerian Perhubungan Republik Indonesia, 2018). Garis putus-putus kuning berfungsi sebagai pembatas dan pembagi jalur untuk mengarahkan lalu lintas. Penggunaan garis kuning secara konsisten pada jalan nasional membantu pengemudi mengidentifikasi dengan cepat bahwa mereka berada di jalur nasional utama, meningkatkan kesadaran dan keselamatan berkendara. Warna kuning dipilih karena kontras visualnya yang tinggi, terutama pada kondisi malam hari dan cuaca buruk.

Garis putih. Garis putih adalah marka garis standar yang digunakan pada semua jenis jalan, baik jalan nasional maupun jalan selain nasional. Pada jalan nasional, garis putih berfungsi sebagai pembagi lajur (garis putus-putus) dan peringatan tanda tepi jalur sisi kiri jalan (garis utuh) (Kementerian Perhubungan

Republik Indonesia, 2018). Pada jalan selain nasional, garis putih merupakan satu-satunya warna marka membujur yang digunakan untuk semua fungsi penandaan, termasuk pembagi lajur, tepi perkerasan, dan garis pengarah pada persilangan. Lebar garis putih standar adalah 12 cm, dengan spesifikasi teknis yang sama dengan garis kuning dalam hal ketebalan dan durabilitas. Material marka putih dipilih untuk reflektivitas yang baik dan visibilitas tinggi dalam berbagai kondisi pencahayaan.

1.2.2 *Autonomous car*

Autonomous car atau kendaraan otonom adalah kendaraan yang dapat melakukan sebagian hingga semua tugas yang dilakukan oleh pengemudi manusia secara mandiri dan dapat bergerak ke mana pun kendaraan konvensional dapat pergi tanpa memerlukan intervensi manusia (Wibowo et al., 2023). Teknologi kendaraan otonom berkembang pesat dan menjadi salah satu kemajuan paling revolusioner dalam bidang kecerdasan buatan, yang terus mendorong batas-batas budaya dan menciptakan kemungkinan baru dalam bidang transportasi. Menurut Karangwa et al. (2023), *autonomous vehicle* mengandalkan deteksi dan pengenalan objek yang akurat di lingkungan sekitarnya, yang merupakan persyaratan penting untuk operasi yang aman, terutama dalam lalu lintas padat dengan berbagai jenis kendaraan.

Pada tahun 2014, badan standarisasi otomotif, SAE *International* (*Society of Automotive Engineers*) merilis sistem klasifikasi berdasarkan enam tingkat, mulai dari tanpa otomatisasi hingga sistem yang sepenuhnya otomatis. Sistem ini dijelaskan dalam standar J3016, berjudul "*Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*". Meskipun keduanya berkaitan erat, klasifikasi ini didasarkan pada tingkat intervensi dan kewaspadaan pengemudi yang diperlukan, bukan pada kemampuan kendaraan. Pada 30 September 2016, SAE memperbarui standarnya untuk klasifikasi tingkat otomatisasi berkendara. Laporan teknis ini tidak hanya menetapkan tingkat otomatisasi berkendara yang saling eksklusif, tetapi juga menetapkan terminologi standar untuk ide-ide yang terkait dengan kendaraan yang mengandung sistem otomatis. Standar SAE J3016 ini sejajar dengan tingkatan yang ditentukan oleh Germany Federal Highway Research Institute (BASt) dan sejauh ini sesuai dengan National Highway Traffic Safety Administration (NHTSA) (Ondruš et al., 2020). Enam tingkatan tersebut meliputi:

Level 0 : Tanpa automasi, di mana pada tingkat ini sebagian besar mobil berada di tahap ini. pengemudi mengendalikan kemudi, gas dan rem, memantau keadaan sekitar serta menavigasi dan menentukan kapan harus menggunakan lampu sein, pindah jalur dan belok. Meskipun ada beberapa sistem peringatan (peringatan titik buta dan tabrakan).

Level 1 : Asisten pengemudi, di mana kendaraan pada tingkat ini dapat menangani kemudi atau gas dan rem, tetapi tidak dalam semua situasi, dan pengemudi harus siap untuk mengambil alih fungsi tersebut jika diminta oleh

kendaraan. Artinya, pengemudi harus tetap menyadari apa yang dilakukan mobil dan siap untuk ikut campur jika diperlukan.

Level 2 : Automasi parsial, di mana pada tahap ini mobil menangani kemudi, gas, dan rem, tetapi segera membiarkan pengemudi mengambil alih jika mendeteksi objek atau peristiwa yang tidak direspon oleh mobil. Pada tiga tingkat pertama ini, pengemudi bertanggung jawab untuk memantau sekitar, lalu lintas, cuaca dan kondisi jalan.

Level 3 : Automasi bersyarat, di mana mobil memantau sekitar dan mengurus semua kemudi, gas, dan rem dalam lingkungan tertentu, seperti jalan bebas hambatan. Namun, pengemudi harus siap untuk campur tangan jika mobil memintanya.

Level 4 : Automasi tinggi, di mana mobil menangani kemudi, gas, dan rem, serta memantau sekitar dalam berbagai lingkungan, meskipun tidak semua, seperti cuaca ekstrem. Pengemudi hanya mengaktifkan sistem pengemudi otomatis saat kondisi aman untuk melakukannya. Setelah itu, pengemudi tidak diharuskan melakukan tindakan lebih lanjut.

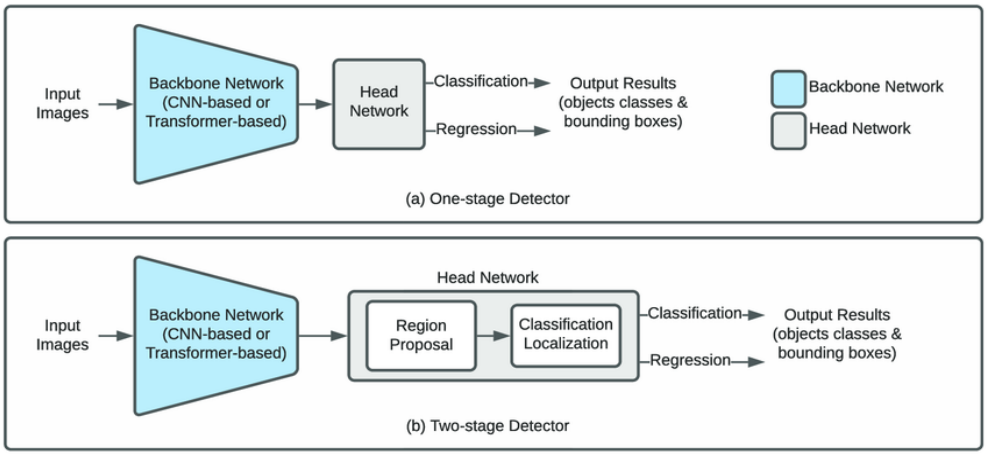
Level 5 : Automasi sepenuhnya, di mana pengemudi hanya perlu menetapkan tujuan dan memulai mobil, selanjutnya mobil menangani semua tugas lainnya. Mobil dapat menuju ke tujuan apa pun dan membuat keputusan sendiri selama perjalanan.engemudi harus tetap menyadari apa yang dilakukan mobil dan siap untuk ikut campur jika diperlukan.

Gambar 3. SAE's *levels of driving automation*

Sumber: (Standard SAE J3016, 2021)

1.2.3 *Computer vision*

Computer vision adalah cabang penelitian yang substansial dan krusial dalam domain kecerdasan buatan, mencakup spektrum aplikasi yang luas. Bidang ini bertujuan agar komputer dan mesin dapat mereplikasi sistem visual manusia untuk memperoleh, mengekstrak, memahami, dan menganalisis informasi yang terkandung dalam gambar maupun video (Hanggoro, 2025). Seiring dengan kemajuan teknologi, *computer vision* telah melalui transformasi besar berkat kemunculan arsitektur *deep learning* yang semakin kompleks. Menurut Tsirtsakis et al. (2025) *vision* memiliki banyak aplikasi mendasar yang merentang di berbagai sektor, seperti keamanan, penginderaan jarak jauh, rekayasa, biologi, kedokteran, transportasi otonom, dan analisis medis.



Gambar 4. Arsitektur *One-stage Detector* dan *Two-stage Detector*

Dalam domain *computer vision*, *deep learning* telah menjadi teknologi transformatif yang mendorong kemajuan signifikan pada berbagai tugas, seperti klasifikasi citra, deteksi objek, segmentasi, dan deteksi anomali di berbagai bidang (Noor & Ige, 2025). Secara spesifik pada tugas pengenalan objek (*object recognition*), Tsirtsakis et al. (2025) menyoroti perkembangan pesat algoritma *deep learning*. Perkembangan ini bergerak dari metode dua tahap (*two-stage*) seperti R-CNN dan *Faster* R-CNN, menuju metode satu tahap (*one-stage*) seperti YOLO dan SSD. Dalam perkembangan ini, pertimbangan utama dalam memilih arsitektur, terutama untuk aplikasi *real-time* seperti kendaraan otonom dan sistem pengawasan, adalah kompromi (*trade-off*) antara akurasi dan kecepatan inferensi.

Computer vision memainkan peran penting dalam pengembangan kendaraan otonom (*autonomous car*) dengan meningkatkan efisiensi operasional dan keselamatan. Teknologi ini memungkinkan sistem untuk melakukan rekonstruksi lingkungan dalam bentuk peta tiga dimensi (3D), serta menjalankan tugas-tugas persepsi krusial seperti deteksi dan klasifikasi objek. Melalui pemrosesan data visual, kendaraan otonom mampu mencapai tingkat kesadaran situasional (*situational awareness*) yang tinggi, memungkinkannya untuk mengambil keputusan berdasarkan kondisi lingkungan yang dinamis. Sebagai komponen fundamental,

computer vision berfungsi untuk menerjemahkan data visual mentah menjadi informasi yang dapat dipahami oleh mesin, sebuah proses yang esensial untuk navigasi otonom. Dalam konteks ini, computer vision bertanggung jawab untuk menganalisis data gambar pada level piksel dan mengekstraksi fitur-fitur relevan. Dengan demikian, teknologi ini menjadi pilar utama dalam mengotomatisasi proses pengambilan keputusan dan memungkinkan fungsionalitas otonom pada kendaraan modern (Gajjar & Sanyal, 2023).

1.2.4 Sistem deteksi *lajur* jalan

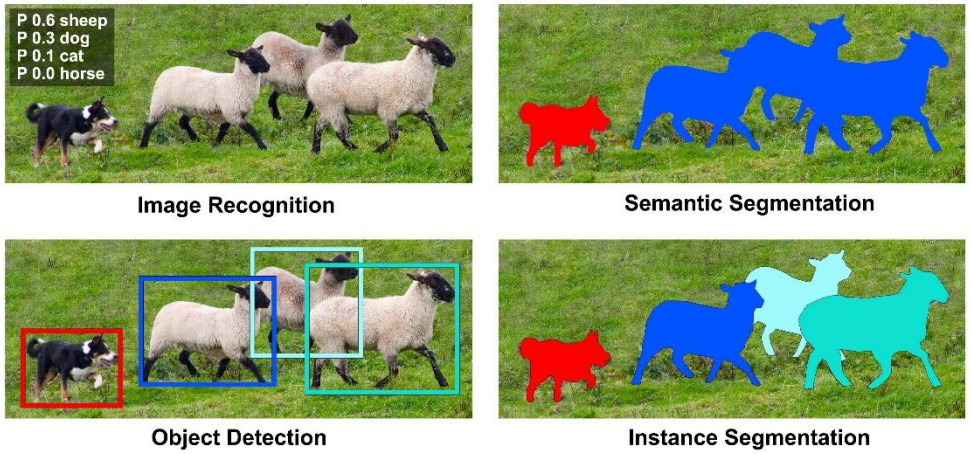
Sistem deteksi lajur jalan merupakan sebuah teknologi fundamental dalam ranah *computer vision* dan kendaraan otonom, yang berfungsi esensial untuk meningkatkan keselamatan di jalan raya serta sistem navigasi kendaraan. Seiring dengan meningkatnya otomatisasi kendaraan dan adopsi A*dvanced Driver Assistance Systems* (ADAS), kemampuan untuk mendeteksi dan melacak lajur jalan secara akurat telah menjadi krusial (Zakaria et al., 2023). Teknologi ini bekerja dengan cara mendeteksi batas-batas lajur dan mengirimkan umpan balik *real-time* ke sistem kontrol kendaraan, yang dicapai melalui kombinasi teknik pemrosesan citra yang canggih, algoritma *machine learning*, dan data sensor. Menurut Aron & Florin (2024), penelitian lebih lanjut di bidang ini masih sangat diperlukan untuk meningkatkan performa sistem deteksi lajur. Fokus penelitian tersebut adalah pada pengembangan algoritma yang lebih canggih namun tetap mudah diimplementasikan dan tidak menuntut sumber daya perangkat keras yang besar, terutama agar sistem dapat beroperasi secara andal dalam kondisi lalu lintas yang kompleks dan kondisi cuaca yang buruk.

Sistem deteksi lajur berbasis visi (*vision-based lane detection*) memanfaatkan kamera yang umumnya diposisikan di belakang kaca depan kendaraan untuk menangkap citra jalan. Melalui interpretasi marka jalan, seperti garis putih, sistem ini mengidentifikasi jalur berkendara dan akan memberikan peringatan kepada pengemudi jika kendaraan terdeteksi keluar dari lajurnya (Zakaria et al., 2023). Dalam konteks implementasi pada *Advanced Driver Assistance Systems* (ADAS) di kondisi minim cahaya, arsitektur sistem ini secara umum dapat dikategorikan menjadi tiga segmen: deteksi lajur (*lane detection*), komputasi tepi (*edge computing*), dan pengambilan keputusan (*decision making*). Modul *lane detection* bertanggung jawab untuk mengidentifikasi posisi dan geometri lajur saat ini melalui analisis *feed* video dari kamera. Selanjutnya, *edge computing* menggunakan sumber daya komputasi onboard serta teknologi *cloud computing* untuk menganalisis hasil deteksi tersebut, yang kemudian diintegrasikan dengan kondisi jalan *real-time* dan data status kendaraan guna memfasilitasi penilaian situasional yang komprehensif (Zhang et al., 2024).

1.2.5 *Semantic segmentation*

Semantic segmentation adalah algoritma *deep learning* yang melakukan klasifikasi pada level piksel gambar, di mana setiap piksel diberi label kategori tertentu sehingga memungkinkan identifikasi area atau objek yang berbeda dalam citra (Lv et al., 2023). Teknik ini membagi sebuah gambar menjadi beberapa segmen berdasarkan kesamaan sifat visual dan *semantics* sehingga dapat memberikan informasi detail dan kaya tentang bagian-bagian gambar. Berbeda dengan deteksi objek yang menggunakan *bounding box*, *semantic segmentation* memberikan hasil yang lebih halus dan komprehensif pada tingkat piksel. Berbeda secara fundamental dari klasifikasi citra yang memberikan label tunggal untuk keseluruhan gambar atau deteksi objek yang melokalisasi objek menggunakan kotak pembatas (*bounding box*), segmentasi semantik menawarkan analisis yang lebih mendalam di tingkat piksel, menjadikannya krusial untuk aplikasi yang menuntut presisi dan detail tinggi. Menurut Ulku & Akagündüz (2022), bidang segmentasi semantik berbasis *deep learning* telah menjadi area riset yang sangat dinamis, ditandai oleh kemunculan berbagai arsitektur inovatif seperti *Fully Convolutional Networks* (FCN), U-Net, SegNet, PSPNet, dan seri DeepLab, yang secara kolektif telah mendorong batasan performa (state-of-the-art) pada berbagai dataset benchmark standar industri seperti PASCAL VOC, *Cityscapes*, dan ADE20K.

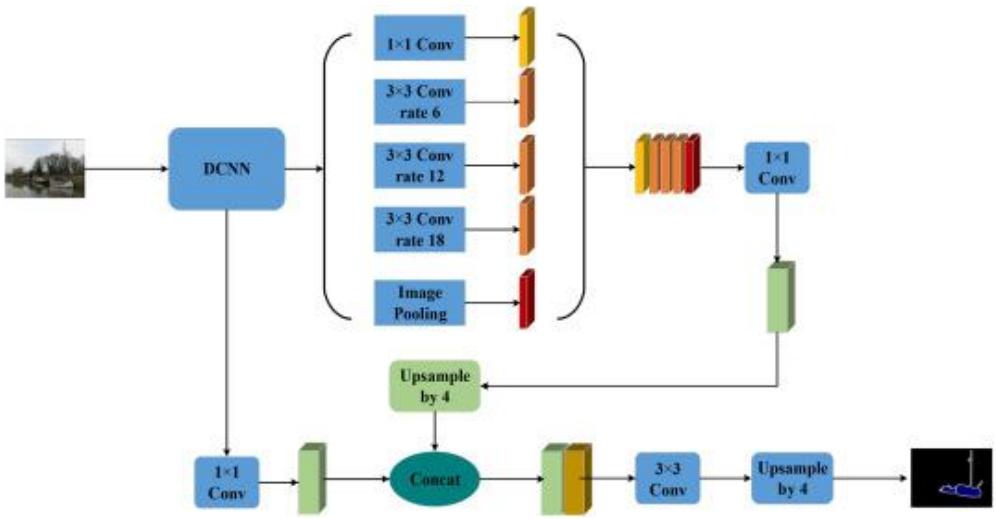
Lv et al. (2023) menjelaskan bahwa semantic segmentation memainkan peran kunci dalam aplikasi penginderaan jarak jauh dan *autonomous driving*, yang memungkinkan pemahaman mendalam terhadap kondisi lingkungan visual dengan mengidentifikasi jalan, kendaraan, pejalan kaki dan objek lain secara tepat. Seiring perkembangan pemrosesan citra digital dan *deep learning*, metode berbasis *Convolutional Neural Networks* (CNN) dan *attention* telah menjadi arsitektur dominan dalam performa segmentasi, mengatasi keterbatasan metode tradisional seperti *thresholding* dan *clustering*.



Gambar 5. Perbedaan tugas visi komputer
Sumber: (MIT, 2025)

Terdapat perbedaan fundamental antara *semantic segmentation*, *object detection*, dan *instance segmentation* dalam cara mereka menganalisis dan mengklasifikasikan objek dalam sebuah citra. *Semantic segmentation* bekerja dengan menetapkan label kategori pada setiap piksel, mengelompokkan semua piksel yang memiliki kesamaan (misalnya, semua mobil) ke dalam satu kelas yang sama tanpa membedakan antar individu objek (Lv et al., 2023). Sebaliknya, object detection berfokus pada identifikasi dan pelokalan objek menggunakan kotak pembatas (*bounding box*), yang hanya menunjukkan lokasi dan kategori objek tanpa memberikan deliniasi piksel yang presisi (Zakaria et al., 2023). Di sisi lain, *instance segemntation* adalah gabungan dari kedua pendekatan tersebut; ia tidak hanya mengklasifikasikan setiap piksel ke dalam kelasnya, tetapi juga secara unik membedakan setiap *instance* atau objek individual dalam kelas yang sama (Lv et al., 2023; Ulku & Akagündüz, 2022).

1.2.6 *DeepLabv3+*



Gambar 6. Arsitektur *DeepLabv3+*
Sumber: (Y. Liu et al., 2024)

DeepLabv3+ adalah sebuah arsitektur segmentasi semantik yang merupakan pengembangan dari *DeepLabv3*, yang secara spesifik menambahkan modul *decoder* sederhana namun efektif untuk menyempurnakan hasil segmentasi, terutama pada bagian batas-batas objek. Arsitektur ini secara efektif mengatasi permasalahan yang disebabkan oleh *downsampling* berulang pada CNN, yang sering kali mengurangi resolusi *feature map* sehingga mengakibatkan penurunan akurasi prediksi dan hilangnya detail batas dalam segmentasi (Y. Liu et al., 2024). *DeepLabv3+* menerapkan struktur *encoder-decoder* di mana *encoder* bertugas menangkap informasi kontekstual jarak jauh (*long-range*) dengan mengurangi dimensi spasial secara bertahap, sementara decoder berfungsi untuk memulihkan informasi spasial dan menghasilkan deliniasi objek yang lebih tajam. Menurut Y. Liu et al. (2024), optimalisasi lebih lanjut pada *DeepLabv3+* melalui integrasi mekanisme atensi, seperti *Polarized Self-Attention* (PSA) setelah modul ASPP dan *Channel Attention Mechanism* (ECA-Net) setelah fitur low-level, dimana dapat meningkatkan kemampuan ekstraksi informasi detail dari *feature map* dan menghasilkan pemulihan batas segmentasi yang lebih presisi, yang dibuktikan dengan peningkatan mean IoU sebesar 2,8% pada dataset *Cityscapes*.

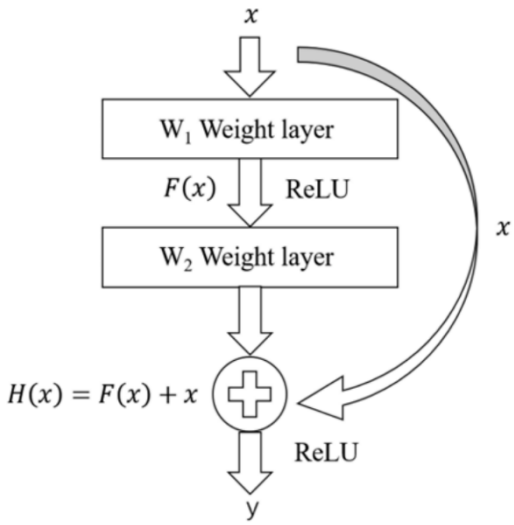
Komponen *encoder* pada *DeepLabv3+* memanfaatkan arsitektur *DeepLabv3* yang telah dimodifikasi, dengan mengintegrasikan modul *Atrous Spatial Pyramid Pooling* (ASPP) untuk menangkap informasi kontekstual multi-skala melalui proses pooling fitur pada beberapa resolusi secara paralel. ASPP sendiri menggunakan *atrous convolution* (juga dikenal sebagai *dilated convolution*), yang merupakan sebuah teknik untuk menyempurnakan *effective field of view* dari konvolusi dengan memodifikasi *field of view* menggunakan parameter yang disebut *atrous rate* (Yunidar

et al., 2024). Pendekatan *atrous convolution* ini adalah metode yang sederhana namun kuat untuk memperluas *field of view* sebuah filter tanpa menambah beban komputasi ataupun jumlah parameter. Dalam implementasinya, Yunidar et al. (2024) menunjukkan bahwa penggunaan *DeepLabv3+* dengan *backbone* ResNet-50 pada tugas segmentasi wajah stunting mampu mencapai akurasi sempurna 100% dan mengurangi *loss* hingga 0,45%. Performa ini dicapai setelah melakukan preprocessing data menggunakan *DeepLabv3+* itu sendiri untuk menghilangkan fitur-fitur yang tidak relevan dan background, yang sekaligus menggarisbawahi pentingnya *preprocessing* data berkualitas tinggi untuk meningkatkan presisi dan reliabilitas model.

Modul *decoder* pada *DeepLabv3+* memiliki peran krusial dalam menyempurnakan hasil segmentasi, khususnya dengan memulihkan detail objek yang mungkin hilang. Proses ini dimulai ketika *feature map* dari *encoder* yang umumnya memiliki *output stride* 16 di-*upsample* secara bilinear dengan faktor 4. Hasil *upsampling* ini kemudian digabungkan (dikonkatenasi) dengan fitur *low-level* yang relevan dari *backbone network*, yang memiliki resolusi spasial yang sama (Chai et al., 2025). Untuk menjaga efisiensi komputasi, sebuah konvolusi 1x1 diterapkan terlebih dahulu pada fitur *low-level* tersebut guna mereduksi jumlah *channel* sebelum proses penggabungan. Setelah digabungkan, beberapa lapisan konvolusi 3x3 diaplikasikan untuk menyempurnakan fitur gabungan, yang kemudian diikuti oleh tahap *upsampling bilinear* sederhana terakhir (faktor 4) untuk menghasilkan prediksi segmentasi akhir dengan resolusi yang sama persis dengan citra masukan (Wang et al., 2024). Dalam penelitian terkait, Chai et al. (2025) mengusulkan peningkatan arsitektur dengan mengintegrasikan mekanisme *Transformer* dan *Channel Attention* (CA), serta menggunakan dua *backbone* untuk ekstraksi fitur. Pendekatan ini terbukti secara signifikan meningkatkan pengumpulan konteks global dan lokal, yang pada akhirnya menghasilkan perbaikan performa segmentasi yang substansial.

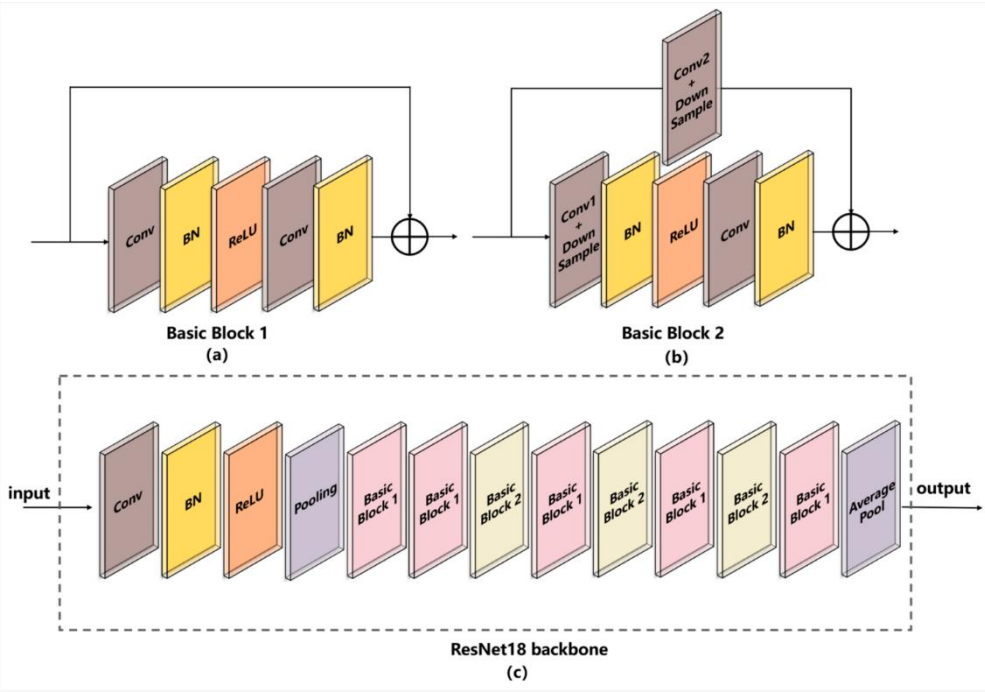
1.2.7 *ResNet*-18 *backbone*

ResNet (*Residual Network*) adalah sebuah arsitektur jaringan saraf konvolusional (convolutional neural network) yang banyak digunakan, di mana strukturnya dibangun dari rangkaian blok-blok residual. Sebagaimana diilustrasikan pada Gambar 19, sebuah blok residual memiliki masukan x dan keluaran y. Di dalam blok tersebut, W_1 dan W_2 melambangkan bobot (*weights*) dari lapisan pertama dan kedua. Ciri khasnya adalah adanya koneksi *shortcut* (ditunjukkan oleh panah melengkung) yang menghubungkan masukan awal langsung ke keluaran. Secara fungsional, masukan x diproses melalui lapisan pertama (bobot W_1) dan fungsi aktivasi untuk menghasilkan $F(x)$. Setelah $F(x)$ melalui transformasi linier oleh lapisan kedua (bobot W_2), hasilnya dijumlahkan secara aditif dengan masukan asli x untuk mendapatkan $H(x)$. Nilai $H(x)$ ini kemudian dilewatkan ke fungsi aktivasi ReLU untuk menghasilkan keluaran akhir y. Berkat pemanfaatan blok-blok residual ini, ResNet mampu mengoptimalkan lapisan-lapisan jaringan yang sangat dalam secara efektif sekaligus mengurangi redundansi (B. Li & Gu, 2023).



Gambar 7. *Residual Block*
Sumber: (B. Li & Gu, 2023)

Mengacu pada Gambar 7, arsitektur *backbone* ResNet18 diketahui terdiri dari enam tahapan utama, yang secara berurutan meliputi: *Conv* 1, *Layer* 1, *Layer* 2, *Layer* 3, *Layer* 4, dan sebuah lapisan *Pool*. Tahap inisial, *Conv* 1, tersusun atas kombinasi lapisan konvolusional, lapisan *batch normalization*, fungsi aktivasi ReLU, dan sebuah lapisan *pooling*. Parameter spesifik untuk lapisan konvolusional pada tahap ini adalah ukuran *kernel* 7x7, *stride* 2, dan *padding* 3, sedangkan lapisan pooling yang mengikutinya menggunakan *kernel* 3x3, *stride* 2, dan *padding* 1. Selanjutnya, *Layer* 1 dibentuk oleh dua unit *Basic Block* 1. Sementara itu, *Layer* 2, *Layer* 3, dan *Layer* 4 masing-masing terdiri dari satu unit *Basic Block* 1 dan satu unit *Basic Block* 2. Detail arsitektur untuk kedua tipe *Basic Block* tersebut (Gambar 8a dan 8b) dirinci lebih lanjut oleh B. Li & Gu (2023).



Gambar 8. Skema diagram ResNet-18 backbone:
(a) *Basic Block* 1; (b) *Basic Block* 2; (c) ResNet-18 *Backbone*
Sumber: (B. Li & Gu, 2023)

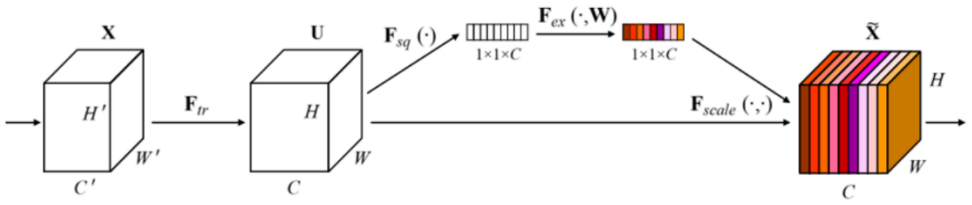
Struktur *Basic Block* 1 terdiri dari dua lapisan konvolusi yang berurutan, di mana kedua lapisan tersebut memiliki konfigurasi parameter yang identik, yakni ukuran *kernel* 3x3, *stride* 1, dan *padding* 1. Berbeda halnya dengan *Basic Block* 2, yang tersusun atas tiga lapisan konvolusi. Pada blok ini, dua lapisan konvolusi pertama secara spesifik berfungsi untuk melakukan *down-sampling*. Lapisan *down-sampling* pertama menggunakan *kernel* 3x3, *stride* 2, dan *padding* 1. Lapisan *down-sampling* kedua menggunakan *kernel* 1x1, *stride* 2, dan *padding* 1. Sementara itu, lapisan konvolusi ketiga pada *Basic Block* 2 dirancang dengan *kernel* 3x3, *stride* 1, dan *padding* 1. Rincian lengkap mengenai arsitektur *backbone* ResNet-18 ini didokumentasikan lebih lanjut pada Tabel 1 (B. Li & Gu, 2023).

Tabel 1. Detail ResNet-18 *Backbone*

Stage	*Output*	*Structure Details*
Conv1	112 x 112 x 64	$7\ x\ 7$, 64, $s = 2$, *p = 3*
Layer1	56 x 56 x 64	$3\ x\ 3$, *max-pooling*, $s\ =\ 2, p\ =\ 1$
Layer2	28 x 28 x 128	$\begin{bmatrix} 3\ x\ 3.64 \\ 3\ x\ 3.64 \end{bmatrix} x\ 2$
Layer3	14 x 14 x 256	$\begin{bmatrix} 3\ x\ 3.128 \\ 3\ x\ 3.128 \end{bmatrix} x\ 2$
Layer4	7 x 7 x 512	$\begin{bmatrix} 3\ x\ 3.512 \\ 3\ x\ 3.512 \end{bmatrix} x\ 2$
Pool	1 x 1 x 512	*Global average pooling*

Sumber: (B. Li & Gu, 2023)

1.2.8 *Squeeze and Excitation* (SE)



Gambar 9. *Squeeze-and-Excitation block*

Sumber: (Hu et al., 2018)

Struktur fundamental dari blok *Squeeze-and-Excitation* (SE) diilustrasikan secara skematis pada Gambar 9. Untuk transformasi F_{tr} (misalnya, sebuah operasi konvolusi) yang memetakan masukan X ke peta fitur U (di mana $U \in R^{H \times W \times C}$), dapat dikonstruksikan sebuah blok SE yang berkorespondensi untuk melakukan rekalibrasi fitur adaptif. Operasi pertama dalam blok SE adalah *squeeze*, yang mengagregasi informasi spasial global ($H \times W$) dari peta fitur U untuk menghasilkan sebuah deskriptor saluran (*channel descriptor*). Deskriptor ini berfungsi sebagai *embedding* yang merangkum distribusi respons global untuk setiap *channel* fitur, memungkinkan pemanfaatan informasi dari *receptive field* global jaringan oleh lapisan-lapisan berikutnya. Operasi *squeeze* ini diikuti oleh operasi *excitation*, yang merupakan mekanisme *self-gating* sederhana. Mekanisme ini menggunakan *embedding* hasil *squeeze* sebagai masukan untuk menghasilkan sekumpulan bobot modulasi spesifik untuk setiap *channel*. Bobot-bobot modulasi ini kemudian diterapkan (dikalikan secara *channel-wise*) pada peta fitur U untuk menghasilkan keluaran akhir dari blok SE, yang selanjutnya dapat diumpankan sebagai masukan ke lapisan jaringan berikutnya (Hu et al., 2018).

Sebuah jaringan SE (SENet) dapat dikonstruksi dengan menumpuk sejumlah blok SE. Alternatifnya, blok SE ini juga dapat diintegrasikan sebagai substitusi langsung pada blok arsitektur asli pada berbagai kedalaman jaringan. Meskipun templat blok SE ini bersifat generik, peran fungsionalnya terdiferensiasi berdasarkan kedalamannya di dalam jaringan. Pada lapisan-lapisan awal, blok SE beroperasi secara *class-independent* (tidak bergantung pada kelas). Fungsinya adalah untuk mengeksitasi (memperkuat) fitur-fitur informatif, sehingga memperkaya representasi *low-level* yang dibagikan. Sebaliknya, pada lapisan yang lebih dalam (akhir), blok SE menjadi semakin *class-specific* (spesifik terhadap kelas), di mana ia merespons masukan dengan cara yang sangat spesifik tergantung pada kelas objek yang dideteksi. Akibatnya, manfaat yang dihasilkan dari proses rekalibrasi fitur yang dilakukan oleh blok SE ini dapat terakumulasi di seluruh *channel* seiring bertambahnya kedalaman jaringan (Hu et al., 2018).

Squeeze (global information embedding). Guna menangani masalah eksploitasi dependensi *channel* (*channel dependency*), langkah pertama adalah mengevaluasi sinyal keluaran pada setiap *channel* fitur. Dalam operasi konvolusi standar, setiap filter yang dipelajari beroperasi dalam bidang reseptif yang terbatas secara lokal. Akibatnya, setiap unit individual pada peta fitur keluaran U tidak dapat memanfaatkan informasi kontekstual yang berada di luar wilayah lokal tersebut. Untuk mengatasi limitasi ini, diusulkan sebuah operasi untuk memadatkan (*squeeze*) informasi spasial global ke dalam satu set deskriptor *channel*. Operasi ini diimplementasikan secara efisien dengan menggunakan *global average pooling* untuk menghasilkan statistik pada level *channel* (Hu et al., 2018). Secara formal, statistik z (deskriptor *channel*) dihasilkan dengan "mengecilkan" peta fitur U di sepanjang dimensi spasialnya ($H \times W$). Elemen ke-c dari z (yaitu z_c) dihitung sebagai berikut:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \quad (1)$$

Keluaran z_c *dari* transformasi U ini dapat diinterpretasikan sebagai statistik yang merepresentasikan informasi agregat dari keseluruhan gambar untuk *channel* ke-c. Pemanfaatan informasi global semacam ini merupakan praktik yang umum ditemukan dalam berbagai pendekatan *feature engineering* sebelumnya (Hu et al., 2018).

Excitation (adaptive recalibration). Untuk memanfaatkan informasi yang telah diagregasi oleh operasi *squeeze*, langkah selanjutnya adalah operasi *excitation* (eksitasi). Operasi ini bertujuan untuk menangkap dependensi antar-*channel* secara penuh. Fungsi ini dirancang agar memiliki fleksibilitas untuk mempelajari interaksi non-linear antar *channel* dan memungkinkan penekanan pada beberapa *channel* secara simultan. Hal ini dicapai dengan menggunakan

mekanisme *gating* sederhana yang memanfaatkan aktivasi *sigmoid* (σ). Mekanisme ini, yang direpresentasikan oleh rumus:

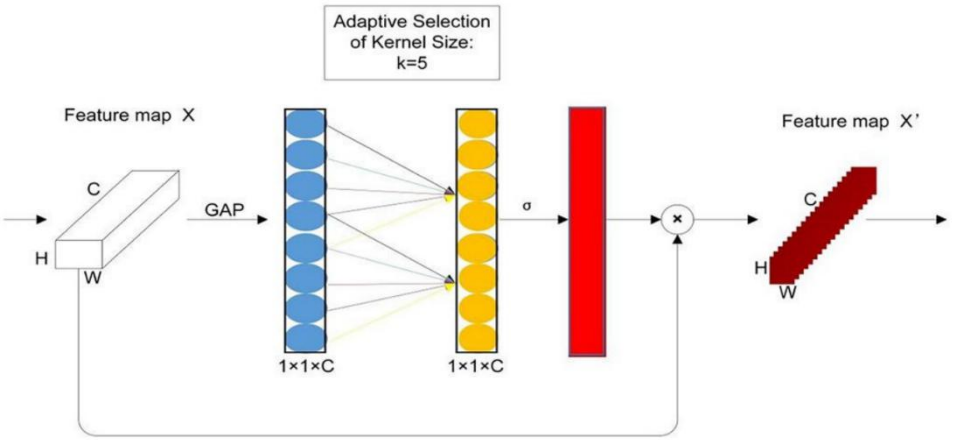
$$s = F_{ex}(z, W) = \sigma\big(g(z, W)\big) = \sigma\big(W_2 \delta(W_1 z)\big) \quad (2)$$

Secara efektif mengimplementasikan dua lapisan *fully-connected* (FC) yang membentuk *bottleneck*. Dalam rumus tersebut, δ adalah fungsi aktivasi ReLU, $W_1 \in R^{\frac{C}{r} \times C}$, dan $W_2 \in R^{C \times \frac{C}{r}}$. Lapisan FC pertama ($W_1$) mereduksi dimensi *channel* dengan rasio reduksi r (diikuti oleh aktivasi ReLU), dan lapisan FC kedua (W_2) mengembalikan dimensi ke C (Hu et al., 2018).

Operator *excitation* ini berfungsi untuk memetakan deskriptor masukan z (yang berisi informasi global) ke sekumpulan bobot *channel* (s). Proses ini *memperkenalkan* dinamika yang dikondisikan oleh masukan (*input-conditioned dynamics*) dan berperan sebagai mekanisme *self-attention* pada *channel*, yang operasinya tidak dibatasi oleh *receptive field* (bidang reseptif) lokal. Strategi ini pada akhirnya meningkatkan sensitivitas jaringan terhadap fitur-fitur yang informatif, yang kemudian dapat dieksploitasi oleh transformasi-transformasi berikutnya, sehingga mengoptimalkan kinerja model secara keseluruhan (Hu et al., 2018).

1.2.9 *Efficient Channel Attention* (ECA)

Modul *Efficient Channel Attention* (ECA) (Wang et al., 2020) adalah sebuah mekanisme atensi *channel* yang dirancang sebagai penyempurnaan dari blok *Squeeze-and-Excitation* (SE) (Hu et al., 2018). ECA memperkenalkan strategi interaksi *cross-channel* lokal yang secara signifikan menghindari reduksi dimensi, dan dapat diimplementasikan secara efisien melalui konvolusi satu dimensi (1D). Berbeda dengan SE, yang menggunakan *global average pooling* diikuti oleh dua lapisan *fully-connected* (FC) untuk menangkap interaksi *cross-channel* non-linear sebelum menghasilkan bobot atensi melalui fungsi *sigmoid*, ECA mengambil pendekatan yang berbeda. Sementara SE berfokus pada korelasi antar *channel* secara global dan menggunakannya untuk memperkuat fitur penting serta menekan fitur yang kurang relevan, ECA mempertahankan dimensi *channel* setelah operasi *global average pooling*. Alih-alih menggunakan FC *layer*, ECA secara langsung menangkap informasi interaksi *cross-channel* dengan mempertimbangkan interaksi antara setiap *channel* dan K tetangga terdekatnya. Keunggulan utama ECA terletak pada efisiensinya, dimana ia dapat diintegrasikan sebagai modul *plug-and-play* yang sangat ringan ke dalam berbagai arsitektur CNN untuk meningkatkan kinerja representasi fitur (Gao et al., 2020). Proses implementasi modul ECA diilustrasikan pada Gambar 10.



Gambar 10. *Efficient Channel Attention Module* (ECA)
Sumber: (Yuan et al., 2022)

Interaksi lintas-saluran (*cross-channel interaction*) merupakan sebuah metode kombinasi fitur inovatif yang berfungsi untuk memperkaya ekspresi fitur dari semantik tertentu. Setelah menerapkan *global average pooling* pada peta fitur, modul ECA secara spesifik menangkap interaksi *cross-channel* lokal dengan menganalisis hubungan antara setiap *channel* fitur dan K channel tetangganya. Keunggulan utama dari pendekatan interaksi *cross-channel* ini adalah kemampuannya untuk menghindari reduksi dimensi (yang biasa terjadi pada modul atensi lain) dan secara efektif mewujudkan interaksi *multi-channel* melalui penerapan konvolusi satu dimensi (Gao et al., 2020). Berdasarkan mekanisme ini, bobot dari fitur y dapat dihitung sebagai berikut:

$$w_i = \sigma\left(\sum_{j=1}^{k} \alpha^j y_i^j\right), y_i^j \in \Omega_i^k \tag{3}$$

Pada rumus tersebut, σ melambangkan fungsi aktivasi *sigmoid*, sementara $\boldsymbol{\Omega_i^k}$ merepresentasikan himpunan dari k *channel* tetangga (*neighbor channels*) dari y_i. Implikasi penting dari Persamaan (3) adalah bahwa seluruh *channel* berbagi parameter (bobot) yang sama. Mekanisme berbagi parameter ini secara signifikan berkontribusi pada efisiensi model yang lebih tinggi (Gao et al., 2020).

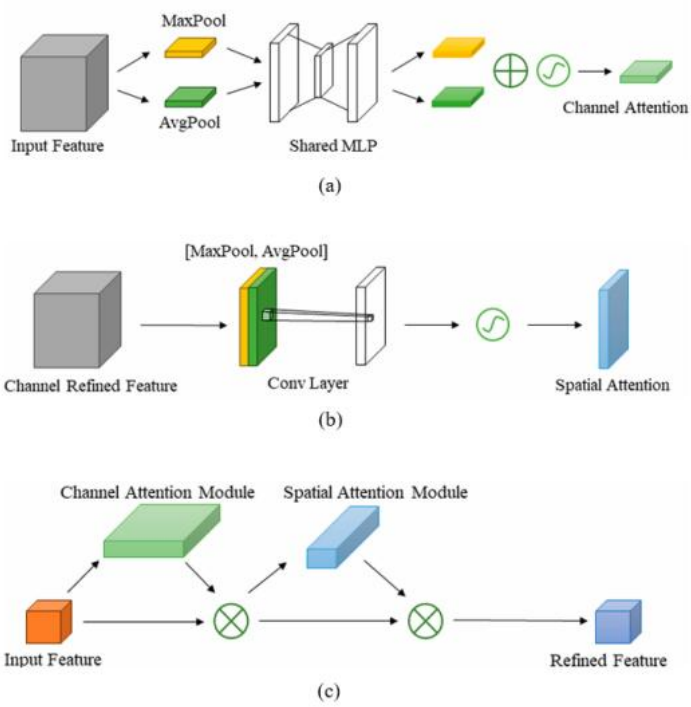
$$w = \sigma(C1D_k(y)) \tag{4}$$

Persamaan (4) merepresentasikan realisasi fitur *channel* yang dicapai melalui konvolusi satu dimensi lintas-*channel* (disingkat C1D). Dalam formulasi ini, C1D adalah notasi untuk operasi konvolusi satu dimensi itu sendiri, sementara parameter k melambangkan cakupan dari interaksi *cross-channel* lokal. Secara lebih spesifik,

nilai k menentukan jumlah *channel* tetangga yang dilibatkan dalam kalkulasi (prediksi) atensi untuk *channel* yang sedang dievaluasi (Gao et al., 2020).

1.2.10 *Convolutional Block Attention Module* (CBAM)

Convolutional Block Attention Module (CBAM) memiliki tujuan fundamental untuk mengekstrak informasi kontekstual yang terkandung dalam sebuah citra. Ekstraksi ini berfungsi untuk menangkap relevansi serta membantu model agar dapat memprioritaskan area-area penting (informatif) sambil secara simultan mengabaikan informasi yang tidak relevan. Mekanisme CBAM ini terdiri dari dua komponen utama, yaitu *channel attention module* (CA) dan *spatial attention module* (SA), yang masing-masing memberikan penekanan pada signifikansi saluran fitur (*feature channels*) dan area spasial dalam citra tersebut. Secara spesifik, *channel attention module* (CA) berkontribusi dalam mengidentifikasi saluran fitur yang krusial untuk tugas tertentu. Hal ini dicapai dengan menganalisis interdependensi antar saluran yang berbeda dan mengoptimalkan alokasi peta fitur, yang pada akhirnya bertujuan untuk meningkatkan kinerja model secara keseluruhan (H. Li et al., 2023).



Gambar 11. Struktur CBAM: (a) *Channel Attention Module*; (b) *Spatial Attention Module*; (c) CBAM

Sumber: (H. Li et al., 2023)

Spatial attention module atau SA, di sisi lain, berfokus pada penentuan signifikansi area piksel individual dalam sebuah citra. Implementasi modul ini

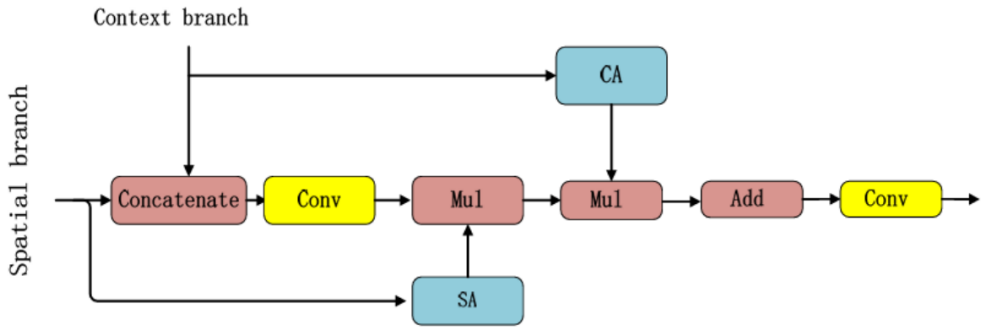
memfasilitasi pemahaman spasial yang lebih baik, sebuah kapabilitas yang sangat krusial untuk mengekstraksi fitur-fitur tepi secara akurat. Arsitektur CBAM secara inovatif mengintegrasikan kedua mekanisme atensi ini (*channel attention* dan *spatial attention*) dengan menerapkannya dalam struktur yang berurutan, sehingga mampu melakukan analisis secara komprehensif pada dimensi channel sekaligus dimensi spasial. Hal ini memungkinkan jaringan saraf untuk memproses fitur citra secara lebih cermat, sambil tetap memperhatikan informasi pada berbagai skala (*multi-scale*). Keunggulan lain dari CBAM adalah arsitekturnya yang ringan dan kemampuannya untuk diintegrasikan secara mulus ke dalam berbagai arsitektur jaringan saraf, sehingga berkontribusi pada peningkatan fleksibilitas dan kinerja model. Kesimpulannya, penggunaan mekanisme atensi terpadu ini memungkinkan model untuk secara adaptif fokus pada informasi yang paling penting, yang mengarah pada ekstraksi fitur tepi yang lebih presisi dan efisien, sehingga pada akhirnya meningkatkan kinerja model dan pemahamannya terhadap konten visual (H. Li et al., 2023).

Arsitektur *Convolutional Block Attention Module* (CBAM), sebagaimana diilustrasikan pada Gambar 11, tersusun atas dua modul atensi yang diaplikasikan secara sekuensial: *Channel Attention Module* (CA) dan *Spatial Attention Module* (SA). Pada tahap pertama (CA), operasi *pooling* diterapkan pada peta fitur masukan (*input feature map*) untuk mengekstrak dan memperoleh informasi bobot atensi untuk setiap *channel*. Peta fitur yang telah disempurnakan oleh atensi *channel* ini kemudian diteruskan ke tahap kedua, yaitu SA. Modul SA ini selanjutnya beroperasi dengan menerapkan *maximum pooling* dan *average pooling* secara efektif di sepanjang dimensi *channel* untuk setiap titik spasial pada peta fitur. Operasi agregasi lintas-*channel* ini memungkinkan modul untuk menangkap fitur spasial pada berbagai skala (*multi-scale*). Bobot atensi spasial untuk setiap titik fitur kemudian dihitung menggunakan prosedur yang serupa dengan yang digunakan dalam CA. Akhirnya, bobot spasial yang diperoleh ini digunakan untuk melakukan modulasi (perkalian berbobot) pada peta fitur masukan (yang telah disempurnakan oleh CA), sehingga menghasilkan fitur keluaran akhir (*refined feature*) yang telah mengintegrasikan informasi kontekstual multi-skala. Singkatnya, struktur hierarkis CBAM, melalui penerapan atensi *channel* dan spasial secara berurutan, memungkinkan ekstraksi fitur multi-skala yang lebih efektif, yang pada gilirannya menghasilkan pemahaman konten citra yang lebih baik serta peningkatan kapabilitas analisis dan pengenalan pada model (H. Li et al., 2023).

1.2.11 *Feature Cross Attention* (FCA)

Modul *Feature Cross-Attention* (FCA), yang arsitekturnya diilustrasikan pada Gambar 12, merupakan mekanisme yang dirancang untuk mengintegrasikan informasi dari level fitur yang berbeda secara sinergis. Modul ini secara strategis memanfaatkan *spatial attention module* untuk mengekstraksi dan menyempurnakan informasi spasial yang detail, yang umumnya terkandung dalam *low-level feature*. Secara bersamaan, *channel attention mechanism* digunakan untuk secara efektif

menangkap informasi kontekstual yang dominan pada *high-level feature*. Dengan demikian, FCA memadukan kekuatan kedua jenis atensi untuk menghasilkan representasi fitur yang lebih komprehensif, di mana detail spasial dari fitur tingkat rendah diperkaya dengan konteks semantik dari fitur tingkat tinggi (Zeng et al., 2020).

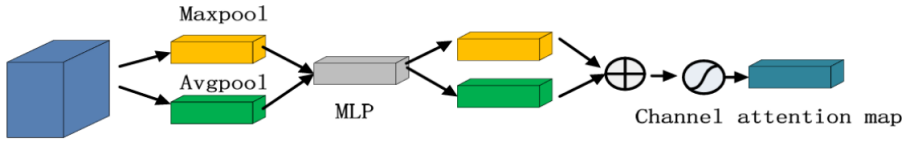


Gambar 12. *Feature Cross Attention (FCA)*
Sumber: (Zeng et al., 2020)

Operasionalisasi modul *Feature Cross Attention* (FCA) secara sinergis memanfaatkan keluaran dari mekanisme atensi yang berbeda untuk menghasilkan representasi fitur yang kaya. *High-level features*, yang dominan mengandung informasi kontekstual dan diekstraksi melalui *channel attention module*, digabungkan dengan *low-level features* yang kaya akan detail spasial dan disempurnakan oleh *spatial attention module* (SA). Proses integrasi ini diawali dengan penggabungan (*concatenation*) keluaran fitur dari kedua cabang (*spatial branch* dan *context branch*), yang kemudian diikuti oleh serangkaian operasi standar, yaitu konvolusi, *batch normalization*, dan aktivasi ReLU. Untuk lebih menyempurnakan pemetaan piksel, fitur yang telah digabungkan tersebut kemudian dimodulasi oleh keluaran dari SA *module* (yang telah melalui normalisasi dan konvolusi non-linier berbentuk S) melalui perkalian *element-wise*. Secara paralel, pada cabang konteks (*context branch*), fitur kontekstual dikompresi secara spasial menggunakan *global pooling* dan *maximum pooling* untuk menghasilkan dua vektor representasi. Vektor-vektor ini selanjutnya diproses melalui lapisan *fully connected* dan operator Sigmoid untuk menghasilkan peta atensi (*attention graph*). Tahap akhir dari modul FCA melibatkan fusi akhir melalui operasi konvolusi, *batch normalization*, dan ReLU (Zeng et al., 2020).

***Channel attention module* (CA).** Mekanisme atensi (*attention mechanism*) telah menjadi paradigma yang diadopsi secara luas dalam berbagai sub-bidang *deep learning* dalam beberapa tahun terakhir, dengan demonstrasi peningkatan performa yang signifikan pada tugas-tugas seperti pemrosesan citra (*image processing*), pengenalan ucapan (*speech recognition*), dan pemrosesan bahasa alami (*natural language processing*). Salah satu wujud implementasi atensi yang berpengaruh adalah SENet (*Squeeze-and-Excitation Network*), yang

bekerja dengan mempelajari bobot relevansi fitur secara adaptif berdasarkan fungsi *loss*. Melalui mekanisme ini, SENet mampu memberikan bobot yang lebih besar pada peta fitur yang informatif dan menekan peta fitur yang kurang signifikan, sehingga mengarahkan model untuk mencapai performa yang lebih baik. Keunggulan utama SENet adalah kemampuannya untuk diintegrasikan ke dalam arsitektur jaringan klasifikasi yang sudah ada (*existing networks*) dengan penambahan parameter dan kompleksitas komputasi yang relatif minimal. Modul atensi *channel* yang digunakan dalam penelitian ini pada dasarnya mengadopsi prinsip yang serupa dengan SENet, namun dengan modifikasi krusial. Modul ini memanfaatkan kedua strategi agregasi spasial, yaitu *max pooling* dan *average pooling* (sementara SENet standar hanya mengandalkan *average pooling*). Keluaran akhir dari modul atensi channel ini diperoleh melalui penjumlahan dari hasil yang diproses melalui kedua jalur *pooling* tersebut. Guna menjaga efisiensi parameter, kedua jalur *pooling* berbagi *Multilayer Perceptron* (MLP) yang identik, yang memastikan bahwa representasi fitur *channel* yang diagregasi berada dalam ruang *embedding* semantik yang koheren. Representasi skematis dari modul atensi *channel* ini disajikan pada Gambar 8 (Zeng et al., 2020).



Gambar 13. *Channel Attention Module* (CA)
Sumber: (Zeng et al., 2020)

Setiap *channel* dalam *channel attention module* dapat dianalogikan sebagai detektor fitur spesifik, sehingga memungkinkan modul untuk secara selektif memfokuskan perhatian pada jenis-jenis fitur tertentu yang relevan. Guna melakukan agregasi informasi spasial secara efektif, modul ini mengimplementasikan dua strategi pooling secara paralel, yaitu *average pooling* dan *maximum pooling*. Penggunaan kedua metode ini bertujuan untuk menangkap aspek informasi kontekstual yang komplementer dari *feature map*. Proses operasional modul atensi *channel* ini secara matematis dirumuskan dalam persamaan (5) dan (6) (Zeng et al., 2020).

$$M_c = \sigma\left(MLP\left(AvgPool\left(F\right)\right) + MLP\left(MaxPool\left(F\right)\right)\right) \quad (5)$$

$$M_c = \sigma\left(W_1(W_2(F^c_{avg})) + W_1(W_0(F^c_{max}))\right) \quad (6)$$

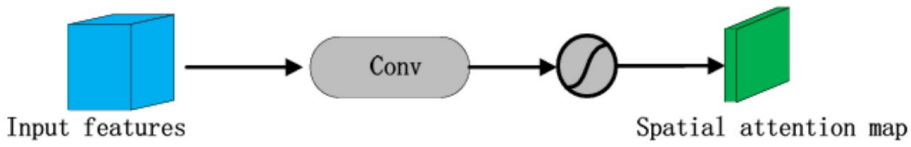
Dalam konteks ini, MLP merepresentasikan *Multilayer Perceptron*, sementara σ adalah fungsi aktivasi sigmoid. Modul ini menerima masukan berupa peta fitur F dengan dimensi $H \times W \times C$ ($F \in \mathbb{R}^{H \times W \times C}$). Langkah pertama

melibatkan agregasi spasial melalui *average pooling* dan *maximum pooling* secara paralel, yang masing-masing menghasilkan deskriptor saluran (*channel descriptor*) berdimensi $1 \times 1 \times C$. Kedua deskriptor ini kemudian diproses secara independen namun menggunakan arsitektur MLP bersama (*shared* MLP) yang terdiri dari dua lapisan, yaitu lapisan tersembunyi pertama dengan C/r neuron dan fungsi aktivasi ReLU, diikuti oleh lapisan keluaran dengan C neuron. Keluaran yang dihasilkan dari pemrosesan kedua deskriptor melalui MLP bersama ini kemudian dijumlahkan secara *element-wise*. Hasil penjumlahan tersebut selanjutnya dilewatkan ke fungsi aktivasi *sigmoid* (σ) untuk menghasilkan koefisien bobot final M_C. Tahap akhir melibatkan perkalian *element-wise* antara koefisien bobot M_C ini dengan peta fitur masukan asli F, menghasilkan peta fitur baru yang telah diskalakan dan teratensi secara channel (Zeng et al., 2020).

***Spatial attention module* (SA).** Modul atensi spasial (*spatial attention module*) dirancang untuk secara adaptif memfokuskan pemrosesan pada lokasi-lokasi spasial yang paling informatif atau bermakna (*meaningful features*) dalam sebuah peta fitur. Ilustrasi skematis dari arsitektur modul ini disajikan secara mendetail pada Gambar 9. Serupa dengan mekanisme atensi *channel*, *spatial attention module* beroperasi pada peta fitur masukan F dengan dimensi $H \times W \times C$ ($F \in \mathbb{R}^{\mathrm{H} \times \mathrm{W} \times \mathrm{C}}$). Langkah pertama melibatkan penerapan operasi *average pooling* dan *maximum pooling* secara paralel di sepanjang dimensi *channel*. Hal ini menghasilkan dua peta fitur (*feature maps*) 2D ($H \times W \times 1$) yang masing-masing merangkum informasi rata-rata dan maksimum di seluruh *channel* untuk setiap lokasi spasial. Kedua peta fitur deskriptif ini kemudian digabungkan (*concatenated*) berdasarkan dimensi *channel* untuk membentuk peta fitur $H \times W \times 2$. Peta fitur gabungan ini selanjutnya dilewatkan melalui sebuah lapisan konvolusi dengan ukuran *kernel* 7x7, diikuti oleh fungsi aktivasi *sigmoid* (σ). Proses ini menghasilkan peta atensi spasial (*spatial attention map*) M_s ($H \times W \times 1$). Akhirnya, peta atensi M_s ini dikalikan secara *element-wise* dengan peta fitur masukan awal F' (yang bisa jadi merupakan F itu sendiri atau hasil dari modul sebelumnya) untuk menghasilkan peta fitur baru yang telah disempurnakan secara spasial, di mana area-area penting secara spasial diberi bobot lebih tinggi (Zeng et al., 2020). Proses operasional ini dirumuskan dalam Persamaan (7) dan (8).

$$M_s(F) = \sigma\left(f^{7x7}([AvgPool(F);\ MaxPool(F)])\right) \quad (7)$$

$$M_s(F) = \sigma\left(f^{7x7}(F^s_{avg};\ F^s_{max})\right) \quad (8)$$

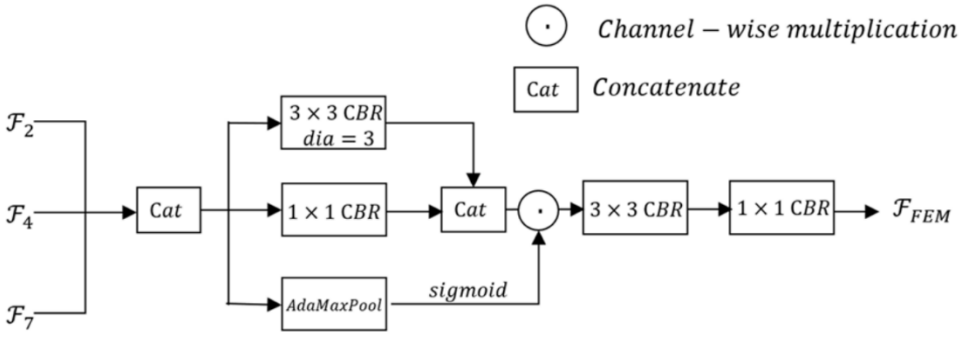


Gambar 14. *Spatial Attention Module (SA)*
Sumber: (Zeng et al., 2020)

Dimana $\boldsymbol{\sigma}$ merepresentasikan fungsi aktivasi *sigmoid*, f melambangkan lapisan konvolusi, dan operator $[;]$ menunjukkan operasi penggabungan (*concatenation*) peta fitur di sepanjang dimensi *channel* (Zeng et al., 2020).

1.2.12 *Feature Enhancement Module (FEM)*

Feature Enhancement Module (FEM) diperkenalkan sebagai sebuah mekanisme yang dirancang untuk mengintegrasikan atau memfusikan (menggabungkan) fitur dari berbagai tingkatan (*multi-level features*). Tujuan dari fusi ini adalah untuk merekonstruksi sebuah peta fitur keluaran yang kaya akan detail (informasi spasial) sekaligus memiliki informasi semantik yang memadai (Yao et al., 2022). Struktur arsitektural dari FEM yang digunakan dalam proses ini diilustrasikan pada Gambar 15.



Gambar 15. Arsitektur *Feature Enhancement Module* (FEM)
Sumber: (Yao et al., 2022)

Secara spesifik, modul FEM menerima masukan dari tiga tingkatan fitur yang berbeda, yaitu *high-level feature* ($f_7 \in R^{C_7 \times H_7 \times W_7}$), *middle-level feature* ($f_4 \in R^{C_4 \times H_4 \times W_4}$), dan *bottom-level feature* ($f_2 \in R^{C_2 \times H_2 \times W_2}$). Karena ketiga peta fitur ini memiliki resolusi spasial ($H \times W$) yang tidak seragam, f_4 dan f_7 (yang memiliki resolusi lebih rendah) harus terlebih dahulu diskalakan ke atas (*upscaled*) agar sesuai dengan dimensi spasial f_2 (yaitu, $H_2 \times W_2$). Proses *upsampling* ini dicapai dengan menggunakan interpolasi bilinear. Setelah ketiga peta fitur tersebut memiliki resolusi yang seragam, ketiganya digabungkan (*concatenated*) di sepanjang dimensi *channel*. Operasi ini menghasilkan satu peta fitur fusi, f_{concat}, dengan dimensi akhir

$R^{(C_2+C_4+C_7)\times H_2\times W_2}$ (Yao et al., 2022). Proses matematis untuk operasi penggabungan ini direpresentasikan sebagai berikut:

$$f_{concat} = cat(f_2, f_4, f_7) \quad (9)$$

Dimana operator '*cat*' merepresentasikan operasi konkatenasi (*concatenation*) yang dieksekusi di sepanjang dimensi *channel*. Operasi konkatenasi ini bersifat melestarikan informasi (*information-preserving*), artinya jumlah informasi yang terkandung dalam setiap lapisan fitur masukan tidak terpengaruh atau terdegradasi. Oleh karena itu, peta fitur yang dihasilkan, f_{concat}, secara efektif mengagregasi representasi fitur yang heterogen, yang secara simultan mencakup informasi detail (spasial) dari lapisan dangkal dan informasi semantik (kontekstual) dari lapisan dalam. Selanjutnya, studi ini menerapkan dua operasi pemrosesan berikut pada peta fitur gabungan f_{concat} (Yao et al., 2022):

$$f_{branch} = cat(w_1 * f_{concat}, w_2 * f_{concat}) \quad (10)$$

$$f_{FEM} = sigmoid(AdaMaxPool(f_{concat}))f_{branch} \quad (11)$$

Dimana simbol '*****' merepresentasikan operasi konvolusi. w_1 adalah lapisan CBR (*Convolution*, *Batch Normalization*, ReLU) dengan *kernel* 1x1, sementara w_2 adalah lapisan CBR dengan *kernel* 3x3 yang diaplikasikan dengan dilasi (tingkat dilasi 3). *AdaMaxPool* merujuk pada operasi *adaptive maximum pooling*. Secara fungsional, cabang w_1 dirancang untuk memperkuat konektivitas antar *channel*, memfasilitasi aliran informasi lintas *channel*. Sebaliknya, cabang w_2, dengan *kernel* dilasi, bertujuan untuk memperluas *receptive field* (bidang reseptif) dari f_{concat}, sehingga memungkinkan penangkapan informasi kontekstual yang lebih global. Fitur-fitur yang dihasilkan dari kedua cabang pemrosesan ini ($w_1 * f_{concat}$ dan $w_2 * f_{concat}$) kemudian digabungkan melalui operasi konkatenasi (*'cat'*) untuk membentuk peta fitur f_{branch}, yang mengintegrasikan informasi dari kedua jalur tersebut.

Untuk merekalibrasi f_{branch} agar lebih fokus pada *channel-channel* yang informatif, mekanisme atensi sederhana diterapkan. Operasi *AdaMaxPool* diaplikasikan pada f_{concat} untuk menghasilkan vektor $f_{AdaMaxPool}$ berdimensi $R^{(C_2+C_4+C_7)\times 1\times 1}$. Vektor ini kemudian dilewatkan melalui fungsi aktivasi *sigmoid* untuk menskalakan nilainya ke rentang [0, 1], menghasilkan bobot atensi. Selanjutnya, bobot atensi $f_{AdaMaxPool}$ (setelah *sigmoid*) dikalikan secara *channel-wise* dengan f_{branch}. Peta fitur hasil perkalian inilah yang didefinisikan sebagai keluaran akhir modul, f_{FEM}, dengan dimensi $R^{(C_2+C_4+C_7)\times H_2\times W_2}$ (Yao et al., 2022).

Tahap pemrosesan akhir dalam *Feature Enhancement Module* (FEM) melibatkan aplikasi sekuensial dari dua lapisan CBR (Convolution, Batch Normalization, ReLU) pada peta fitur f_{FEM}. Lapisan CBR pertama, dengan *kernel* 3x3, memiliki dua tujuan utama, mereduksi dimensi *channel* dari $(C_2 + C_4 + C_7)$ menjadi C (di mana $C = 256$ dalam implementasi ini) dan memperluas *receptive field*

(bidang reseptif) untuk menangkap konteks spasial yang lebih luas. Lapisan CBR kedua, dengan *kernel* 1x1, bertujuan untuk meningkatkan konektivitas antar *channel*, yang secara lebih lanjut memfasilitasi aliran informasi lintas *channel*. Perlu dicatat bahwa penambahan modul FEM hanya memperkenalkan sejumlah kecil parameter baru ke dalam *backbone*. Secara total $5(C_2 + C_4 + C_7)^2 + 9C(C_2 + C_4 + C_7) + C^2$ parameter, yang setara dengan sekitar 0,76 juta. Meskipun demikian, FEM dapat meningkatkan beban komputasi (*computational overhead*) pada *backbone* hingga batas tertentu, terutama karena sebagian besar operasinya dilakukan pada peta fitur beresolusi tinggi. Hal ini berpotensi menyebabkan peningkatan latensi inferensi (*inference speed*) (Yao et al., 2022).

1.2.13 *Zero-Reference Deep Curve Estimation* (Zero-DCE)

Zero-Reference Deep Curve Estimation (Zero-DCE) adalah metode untuk *low-light image enhancement* yang memformulasikan peningkatan cahaya sebagai tugas estimasi kurva *image-specific* menggunakan *deep neural network*, tanpa memerlukan *paired* atau *unpaired training data* (Guo et al., 2020). Pendekatan ini berbeda secara fundamental dari metode berbasis CNN dan GAN yang ada, karena tidak membutuhkan data referensi untuk training. Metode ini melatih *lightweight deep network* yang disebut DCE-Net untuk memperkirakan *pixel-wise* dan *high-order curves* untuk *dynamic range adjustment* dari gambar input. Zero-DCE menarik perhatian karena kesederhanaan konsepnya yang efektif dalam meningkatkan kualitas citra dalam kondisi cahaya rendah, sambil tetap mempertahankan efisiensi komputasi yang tinggi (Guo et al., 2020).

Zero-DCE terdiri dari tiga komponen utama: (1) *Light-Enhancement Curve* (LE-*Curve*) yang berfungsi untuk memetakan nilai piksel input ke output, (2) *Deep Curve Estimation Network* (DCE-Net) yang merupakan *neural network lightweight* untuk memprediksi parameter kurva, dan (3) *Non-Reference Loss Functions* yang didesain khusus untuk mengevaluasi kualitas enhancement tanpa memerlukan reference images (Guo et al., 2020). Arsitektur ini dirancang dengan tiga prinsip desain utama: pertama, setiap nilai piksel dari enhanced image harus berada dalam normalized range untuk menghindari *information loss* dari *overflow truncation*; kedua, kurva harus *monotonic* untuk mempertahankan *contrast* dari *neighboring pixels*; dan ketiga, bentuk kurva harus sederhana dan *differentiable* untuk memungkinkan *gradient backpropagation* selama *training* (Guo et al., 2020).

Zero-DCE menunjukkan beberapa keunggulan signifikan dibandingkan metode enhancement existing. Pertama, pendekatan zero-reference mengeliminasi kebutuhan untuk paired atau unpaired training data. Kedua, simple dan differentiable curve mapping memungkinkan efficient inference dan training dengan computational overhead yang minimal. Ketiga, carefully formulated non-reference losses memungkinkan implicit evaluation dari enhancement quality tanpa memerlukan ground-truth enhanced images (Guo et al., 2020).

LE-*Curve* adalah persamaan matematika yang dirancang untuk melakukan *pixel-wise adjustment* pada *dynamic range* dari citra. Kurva ini didefinisikan sebagai:

$$LE(I(x);\alpha) = I(x) + \alpha I(x)\big(1 - I(x)\big) \quad (12)$$

di mana x adalah koordinat piksel, $I(x)$ adalah nilai piksel input yang dinormalisasi ke range, $LE(I(x);\alpha)$ adalah *enhanced version* dari input, dan $\alpha \in [-1,1]$ adalah *trainable curve parameter* yang mengkontrol *exposure level* (Guo et al., 2020). Desain persamaan quadratic ini memastikan bahwa: (1) output selalu berada dalam *valid range* karena struktur matematisnya yang *carefully designed*, (2) kurva bersifat *monotonic* karena turunan pertama selalu *non-negative*, dan (3) kurva *differentiable* untuk *gradient-based optimization*. LE-*Curve* diaplikasikan secara terpisah pada ketiga RGB channel daripada hanya pada *illumination channel* saja, karena *three-channel adjustment* dapat lebih baik mempertahankan *inherent color* dan mengurangi risiko *oversaturation* (Guo et al., 2020).

Untuk mengatasi *challenging low-light conditions* dengan *dynamic range* yang sangat lebar, Zero-DCE memperkenalkan *Higher-Order Curve* yang diperoleh dengan *iterative application* dari LE-*Curve*:

$$LE_n(x) = LE_{n-1}(x) + \alpha_n LE_{n-1}(x)\big(1 - LE_{n-1}(x)\big) \quad (13)$$

di mana n adalah jumlah iterasi yang mengkontrol *curvature* (Guo et al., 2020). Dengan *iterative application* ini, *higher-order curves* dapat mencapai *powerful adjustment capability* dengan *curvature* yang lebih besar dibandingkan *single-iteration curves*, memungkinkan *handling* dari *extreme low-light conditions* yang lebih efektif. Untuk mengatasi *global enhancement* yang menyebabkan *over* ataupun *under enhancement* di *local regions*, Zero-DCE merumuskan α sebagai *pixel-wise parameter*, sehingga setiap piksel memiliki *corresponding curve* dengan *best-fitting* α. Ini menghasilkan perhitungan:

$$LE_n(x) = LE_{n-1}(x) + A_n(x) LE_{n-1}(x)\big(1 - LE_{n-1}(x)\big) \quad (14)$$

di mana A adalah parameter map dengan ukuran yang sama dengan *input image*, dan *neighboring pixels* diasumsikan memiliki *similar intensity* dan *adjustment curves* (Guo et al., 2020).

Keunikan Zero-DCE terletak pada penggunaan *set* dari *differentiable non-reference losses* yang memungkinkan *zero-reference learning* tanpa *paired* atau *unpaired training data*. Empat tipe *loss function* digunakan dalam training DCE-Net (Guo et al., 2020):

***Spatial Consistency Loss* (L_{spa}).** *Loss* ini mendorong *spatial coherence* dari *enhanced image* dengan mempertahankan *difference* dari *neighboring regions* antara input dan *enhanced image*:

$$L_{spa} = \frac{1}{K}\sum_{i=1}^{K}\sum_{j\in\Omega(i)}\left(\left|Y_i - Y_j\right| - \left|I_i - I_j\right|\right)^2 \quad (15)$$

di mana K adalah jumlah *local regions*, $\Omega(i)$ adalah empat *neighboring regions* (*top, down, left, right*) yang berpusat pada region i, Y dan I adalah *average intensity value* dari *local region* dalam *enhanced* dan *input image* (Guo et al., 2020).

Exposure Control Loss **(L_{exp})**. *Loss* ini dirancang untuk *merestrain under-exposed* atau *over-exposed regions* dengan mengukur jarak antara *average intensity value* dari *local region* ke *well-exposedness* level E:

$$L_{\text{exp}} = \frac{1}{M}\sum_{k=1}^{M}\left|Y_k - E\right| \quad (16)$$

di mana M merepresentasikan jumlah *non-overlapping local regions* ukuran 16×16, dan E ditetapkan ke 0.6 berdasarkan *gray level* dalam RGB *color space* (Guo et al., 2020).

Color Constancy Loss **(L_{col}).** *Loss* ini mengikuti *Gray-World color constancy hypothesis* yang menyatakan bahwa warna dalam setiap sensor *channel* rata-rata ke gray di seluruh *image*. *Loss* ini didefinisikan sebagai:

$$L_{\text{col}} = \sum_{\forall(p,q)\in\varepsilon}\left(J_p - J_q\right)^2, \varepsilon = \{(R,G),(R,B),(G,B)\} \quad (17)$$

di mana J_p adalah *average intensity value* dari *channel* p dalam *enhanced image*, dan (p, q) merepresentasikan *pair* dari *channels* (Guo et al., 2020).

Illumination Smoothness Loss **(L_{tvA}).** *Loss* ini mempertahankan *monotonicity relations* antara *neighboring pixels* dengan menerapkan *total variation regularization* pada setiap *curve parameter map*:

$$L_{tvA} = \frac{1}{N}\sum_{n=1}^{N}\sum_{c\in\xi}\left(\left|\nabla_x A_n^c\right| + \left|\nabla_y A_n^c\right|\right)^2, \quad \xi = \{R,G,B\} \quad (18)$$

di mana N adalah jumlah iterasi, ∇_x dan ∇_y merepresentasikan horizontal dan vertical *gradient operations* (Guo et al., 2020).

Total Loss Function. Empat *loss function* tersebut kemudian dikombinasikan menjadi *total loss*:

$$L_{\text{total}} = L_{\text{spa}} + L_{\text{exp}} + W_{\text{col}}L_{\text{col}} + W_{\text{tvA}}L_{\text{tvA}} \quad (19)$$

di mana W_{col} dan W_{tvA} adalah *weights* dari *losses* untuk *balancing scale* (Guo et al., 2020).

1.2.14 Evaluasi model

***Intersection over Union* (IoU).** Metrik evaluasi ini juga dikenal sebagai *Jaccard Index*, merupakan metrik evaluasi fundamental untuk segmentasi semantik yang mengukur tingkat tumpang tindih antara *mask* hasil prediksi (*predicted mask*) dan *mask* kebenaran (*ground-truth mask*). Metrik ini didefinisikan sebagai:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (20)$$

di mana A merepresentasikan area *ground truth* dan B merepresentasikan area hasil prediksi. $A \cap B$ adalah area irisan di mana prediksi model sesuai dengan *ground truth*, sedangkan $A \cup B$ adalah area gabungan dari kedua *mask* tersebut.

$$mIoU = \frac{1}{C}\sum_{c=1}^{C} IoU_c \quad (21)$$

di mana C adalah jumlah kelas (Rainio et al., 2024). mIoU merupakan metrik standar baku untuk evaluasi segmentasi semantik karena *robust* terhadap ketidakseimbangan kelas, memperlakukan setiap kelas secara setara dalam proses perhitungan rata-rata.

***Dice Coefficient*.** Metrik evaluasi ini juga dikenal sebagai skor F1 dalam konteks segmentasi, adalah metrik alternatif yang mengukur kesamaan tumpang tindih antara *mask* hasil prediksi dan *mask* kebenaran:

$$Dice = \frac{2 \times |A \cap B|}{|A| + |B|} \quad (22)$$

Nilai *Dice Score* juga berkisar dari 0 hingga 1. Hubungan matematis antara *Dice* dan IoU adalah:

$$Dice = \frac{2 \times IoU}{1 + IoU} \quad (23)$$

Ini menunjukkan bahwa *Dice Score* selalu lebih besar atau sama dengan IoU untuk nilai tumpang tindih yang sama (Hirling et al., 2023). Perbedaan ini signifikan ketika area segmentasi sangat kecil: *Dice* memberikan bobot lebih besar kepada

irisan, sehingga lebih toleran terhadap kesalahan batas. Untuk segmentasi multi-kelas, rata-rata *Dice Score* (mDice) dihitung dengan merata-ratakan nilai *Dice* untuk semua kelas.

IoU lebih ketat dalam memberikan penalti terhadap kesalahan kurang segmentasi dan lebih segmentasi karena dinormalisasi oleh gabungan dari kedua wilayah. Sebaliknya, *Dice Score* lebih toleran terhadap kesalahan batas karena memberikan bobot lebih besar kepada irisan (Hirling et al., 2023). Dalam praktik, kedua metrik biasanya dilaporkan bersama-sama untuk memberikan evaluasi yang komprehensif. IoU lebih cocok untuk aplikasi di mana presisi batas adalah krusial, sedangkan *Dice Score* lebih cocok untuk aplikasi di mana deteksi kehadiran wilayah lebih penting daripada presisi batas, terutama ketika menghadapi ketidakseimbangan ukuran kelas.

1.3 Rumusan Masalah

1. Bagaimana merancang dan mengimplementasikan arsitektur *end-to-end* yang mampu meningkatkan hasil segmentasi semantik lajur jalan pada kondisi minim cahaya di malam hari?
2. Bagaimana kinerja arsitektur *end-to-end* yang diusulkan dalam meningkatkan hasil segmentasi lajur jalan pada malam hari jika dibandingkan dengan model *baseline*, serta seberapa baik performa arsitektur tersebut divalidasi melalui analisis metrik evaluasi?

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan sebuah arsitektur *end-to-end* untuk meningkatkan hasil segmentasi semantik lajur jalan dalam kondisi minim cahaya pada malam hari.
2. Mengevaluasi dan menganalisis peningkatan kinerja arsitektur untuk memvalidasi performa arsitektur dalam meningkatkan hasil segmentasi lajur jalan pada malam hari.

1.4.2 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Membantu dan mendorong penelitian mengenai pengembangan autonomous car di Indonesia.
2. Mendukung terciptanya kendaraan otonom yang lebih aman, sehingga dapat mengurangi tingkat kecelakaan lalu lintas akibat kesalahan manusia.

1.5 Ruang Lingkup Penelitian

Adapun ruang lingkup yang menjadi batasan dalam penelitian ini meliputi:

1. Kamera yang digunakan berupa kamera dashcam yang dihadapkan ke arah depan.
2. Pengambilan data dilakukan pada malam hari dalam kondisi cuaca yang baik (tidak hujan).
3. Objek deteksi penelitian difokuskan pada jalan, marka garis sambung dan marka garis putus yang masih dapat diinferensi dengan jelas.
4. Wilayah pengambilan data terbatas pada area Kota Makassar.

BAB II

METODE PENELITIAN

2.1 Tempat dan Waktu Penelitian

Keseluruhan penelitian ini dilaksanakan dalam rentang waktu sepuluh bulan, dimulai pada Januari 2025 hingga Oktober 2025. Seluruh proses analisis, pengembangan model, dan pengujian teknis dipusatkan di laboratorium *Artificial Intelligence* (AI), Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin. Adapun untuk tahap pengumpulan data primer, pengambilan data dilakukan di beberapa ruas jalan utama di Kota Makassar, yang meliputi kawasan *Center Point of Indonesia* (CPI) pemerintah Makassar, Jl. Somba Opu, Jl. Penghibur, Jl. Botolempangan, Jl. Nusantara dan Jl. Ahmad Yani

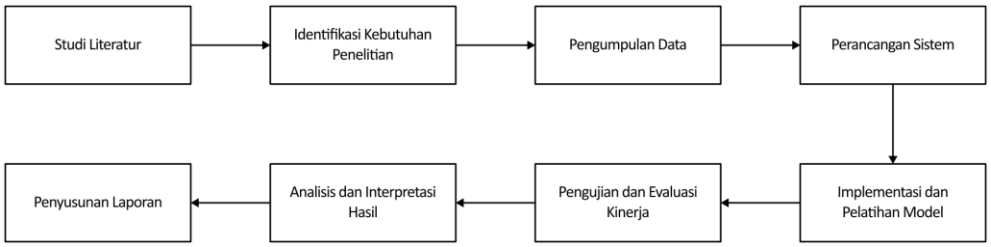
2.2 Instrumen Penelitian

Instrumen-instrumen penelitian yang digunakan dalam penelitian ini, meliputi:

1. Perangkat lunak:
   a. Roboflow
   b. Python
   c. Visual Studio Code
   d. Kaggle
   e. Microsoft Word
2. Perangkat Keras:
   a. Laptop HP Pavilion Gaming 15, RAM 16 GB, SSD 512 GB
   b. *Dashcam* DDPAI Mola N3
3. Algoritma:
   a. ZeroDCE++
   b. Deeplabv3+
4. *Library*
   a. Torch (PyTorch) v2.6.0+cu124, digunakan sebagai framework deep learning utama untuk komputasi tensor, serta untuk membangun arsitektur model segmentasi, mendefinisikan *loss function*, dan menjalankan proses pelatihan (*training loop*).
   b. Torchvision v0.21.0+cu124, digunakan untuk menyediakan fungsi transformasi gambar standar, khususnya *Normalize* untuk menormalisasi data citra masukan sesuai standar *pre-trained* model.
   c. Timm (PyTorch *Image Models*) v1.0.15, digunakan untuk memuat arsitektur model *computer vision* yang telah dilatih sebelumnya (pre-*trained*), yang kemungkinan berfungsi sebagai *encoder* atau *backbone* pada model segmentasi.

d. TorchMetrics v1.7.3, digunakan untuk mengevaluasi performa model segmentasi secara efisien, khususnya untuk menghitung metrik *DiceScore* (*Dice Score*) dan *JaccardIndex* (*Intersection over Union*/IoU).
e. Numpy v1.26.4, digunakan untuk operasi numerik fundamental, terutama dalam memanipuasi array multidimensi yang merepresentasikan citra dan *mask* segmentasi.
f. Pandas v2.2.3, digunakan untuk manajemen data dan tabulasi, seperti membaca file CSV yang berisi metadata dataset (misalnya, *path* gambar dan *mask*) dan mengorganisasi hasil eksperimen.
g. OpenCV (cv2) v4.11.0.86, digunakan untuk tugas pemrosesan citra, seperti membaca file gambar dari disk dan melakukan konversi ruang warna atau operasi visual dasar.
h. Pillow (PIL) v11.2.1, digunakan untuk operasi dasar I/O (Input/Output) citra, seperti membuka dan memanipulasi file gambar dari dataset.
i. Albumentations v2.0.8, digunakan sebagai *library* utama untuk melakukan augmentasi citra (*image augmentation*) yang kompleks pada dataset pelatihan, serta untuk mengonversi citra dan mask ke format Torch Tensor.
j. Scikit-learn (sklearn) v1.2.2, digunakan untuk keperluan strategi validasi model, khususnya untuk mengimplementasikan pembagian dataset menggunakan StratifiedKFold (*K-Fold Cross-Validation*).
k. Matplotlib v3.7.2, digunakan untuk visualisasi data, seperti menampilkan contoh gambar dan *mask* hasil prediksi model, serta memplot grafik kurva pelatihan (misalnya, *loss* dan skor metrik).
l. TQDM v4.67.1, digunakan untuk menampilkan progress bar (bilah kemajuan) yang interaktif selama proses iterasi dataset (pelatihan dan evaluasi) untuk memonitor kemajuan.
m. OS, *Glob*, *Random*, GC, *Collections*, *library* standar Python yang digunakan untuk utilitas sistem; os dan *glob* untuk interaksi file dan direktori (misalnya, mencari *path* data), *random* untuk operasi acak, gc untuk manajemen memori (*garbage collection*), dan *collections* untuk struktur data lanjutan.

2.3 Tahapan Penelitian



Gambar 16. Tahapan penelitian

Tahapan penelitian yang digunakan, sebagaimana diilustrasikan pada Gambar 16, dapat diuraikan sebagai berikut:

1. Studi literatur

   Tahap awal penelitian adalah melakukan studi literatur secara komprehensif. Kegiatan ini berfokus pada pengumpulan dan peninjauan literatur yang relevan dengan topik segmentasi lajur jalan, khususnya pada tantangan yang muncul dalam kondisi minim cahaya (*low-light*). Kajian ini mencakup pemahaman mendalam terhadap metode-metode *state-of-the-art* dalam segmentasi semantik, teknik *low-light image enhancement* (seperti Zero-DCE), serta arsitektur deep learning yang ada.

2. Identifikasi kebutuhan penelitian

   Berdasarkan studi literatur, dilakukan identifikasi kesenjangan penelitian (*research gap*) dan perumusan masalah. Tahap ini mencakup analisis kebutuhan fungsional sistem, seperti kemampuan untuk mendeteksi lajur secara akurat dalam berbagai skenario pencahayaan buruk. Selain itu, dilakukan persiapan teknis untuk pengumpulan data, termasuk penentuan spesifikasi perangkat keras (kamera *dashcam*) dan skenario pengambilan data (misalnya, jalan perkotaan dalam keadaan sepi maupun ramai saat malam hari).

3. Pengumpulan data

   Tahap ini berfokus pada pengumpulan dataset yang diperlukan untuk melatih dan menguji model. Proses ini melibatkan perekaman video lajur jalan secara spesifik pada kondisi malam hari atau minim cahaya. Data video mentah tersebut kemudian diproses untuk mengekstrak *frame-frame* individual. *Frame* yang representatif kemudian melalui proses anotasi manual untuk membuat *ground truth* (label segmentasi) yang presisi serta esensial untuk pelatihan model secara *supervised*.

4. Perancangan sistem

   Pada tahap ini, dilakukan perancangan arsitektur model dan *pipeline* pemrosesan data secara mendetail. Ini mencakup desain arsitektur model yang akan digunakan, misalnya dengan mengadaptasi model segmentasi semantik yang dari penelitian terkait (*DeepLabv3+*) dan mengintegrasikannya dengan modul *low-light enhancement*. Selain itu,

dirancang pula strategi *preprocessing* data, teknik augmentasi data yang relevan, dan penentuan *hyperparameter* awal untuk pelatihan.

5. Implementasi dan pelatihan model

   Rancangan arsitektur sistem kemudian diimplementasikan menggunakan *framework* PyTorch. Proses ini dilanjutkan dengan tahap pelatihan model (*training*) menggunakan dataset yang telah disiapkan dan dianotasi. Selama pelatihan, *parameter* model dioptimalkan untuk meminimalkan *loss function* yang telah ditentukan (*Dice Loss, Focal Loss* dan *Cross-Entropy Loss*), sehingga model dapat mempelajari pola dan fitur lajur jalan dari citra masukan.

6. Pengujian dan evaluasi sistem

   Model yang telah dilatih kemudian diuji kinerjanya menggunakan data uji (*test set*) yang terpisah dan belum pernah dilihat oleh model sebelumnya. Evaluasi dilakukan untuk mengukur keakuratan dan robustisitas model. Metrik evaluasi standar untuk segmentasi semantik, seperti *Intersection over Union* (IoU) dan *Dice Score*, dihitung untuk menilai seberapa baik model mampu menghasilkan prediksi segmentasi yang sesuai dengan *ground truth*.

7. Analisis dan interpretasi hasil

   Hasil dari evaluasi kinerja kemudian dianalisis secara mendalam. Tahap ini tidak hanya mencatat skor metrik kuantitatif, tetapi juga melibatkan interpretasi kualitatif terhadap hasil prediksi visual. Dilakukan analisis kegagalan (*failure case analysis*) untuk memahami skenario spesifik di mana model berkinerja kurang optimal, serta membandingkan performa sistem arsitektur *end-to-end* dengan metode *baseline* atau penelitian terkait.

8. Penyusunan laporan

   Pada tahap akhir, seluruh rangkaian proses penelitian dirangkum dan didokumentasikan. Seluruh temuan ini disusun secara sistematis dan terstruktur ke dalam sebuah dokumen skripsi.

2.4 Teknik Pengambilan Data

Data yang digunakan dalam penelitian ini seluruhnya merupakan data primer, yang diperoleh melalui proses pengambilan data langsung di lapangan. Pengumpulan data dilakukan menggunakan *dashcam* DDPAI Mola N3, yang memiliki resolusi tinggi 2560x1600 piksel dan sudut pandang (*view angle*) lebar 140°. Untuk mendapatkan perspektif yang optimal, kamera diposisikan di bagian dalam kaca depan kendaraan pribadi, tepat di atas dashboard dengan sudut tegak lurus dengan jalanan. Kendaraan yang digunakan juga dilengkapi dengan kaca film (*riben*) 60%, yang

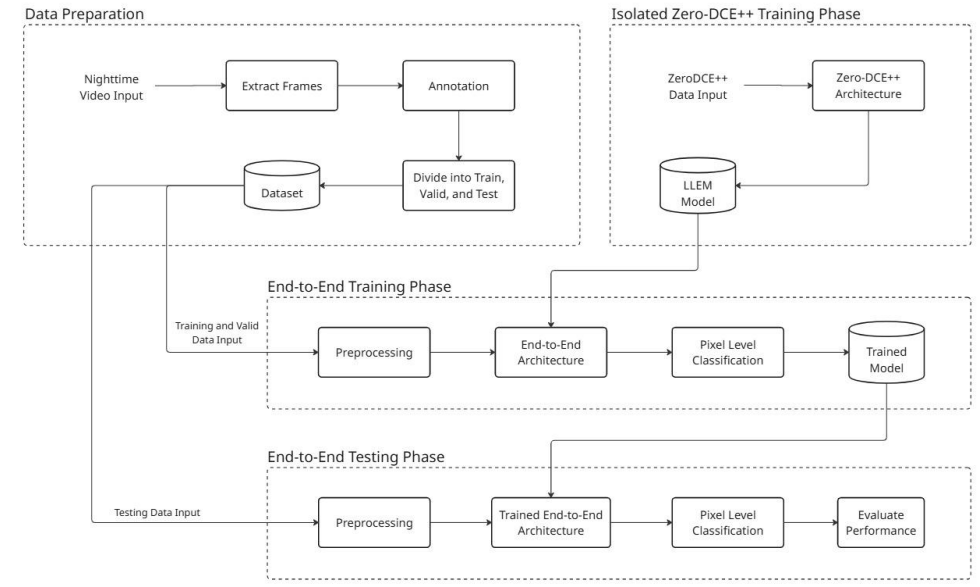
menjadi variabel kondisi nyata karena turut mereduksi intensitas cahaya yang diterima oleh sensor kamera.

Gambar 17. Ilustrasi pengambilan data

Proses perekaman difokuskan secara spesifik pada kondisi malam hari (minim cahaya) di beberapa ruas jalan utama Kota Makassar, yang meliputi *Center Point of Indonesia* (CPI) pemerintah, Jl. Somba Opu, Jl. Penghibur, dan Jl. Botolempangan. Selama proses akuisisi data, kecepatan kendaraan relatif pada rentang 30 km/jam hingga 50 km/jam untuk mensimulasikan kondisi berkendara yang realistis.

Data yang dikumpulkan dirancang untuk mencakup skenario yang beragam, meliputi berbagai tipe marka jalan seperti marka garis membujur penuh, marka garis membujur putus-putus, serta bagian jalan yang tidak memiliki salah satu dari kedua jenis marka garis. Selain itu, kondisi lalu lintas yang bervariasi juga direkam, mencakup jalan yang relatif ramai dengan keberadaan objek lain (seperti mobil atau sepeda motor) maupun kondisi jalan yang cenderung sepi dan area tikungan.

2.5 Perancangan dan Implementasi Sistem



Gambar 18. Diagram alur sistem

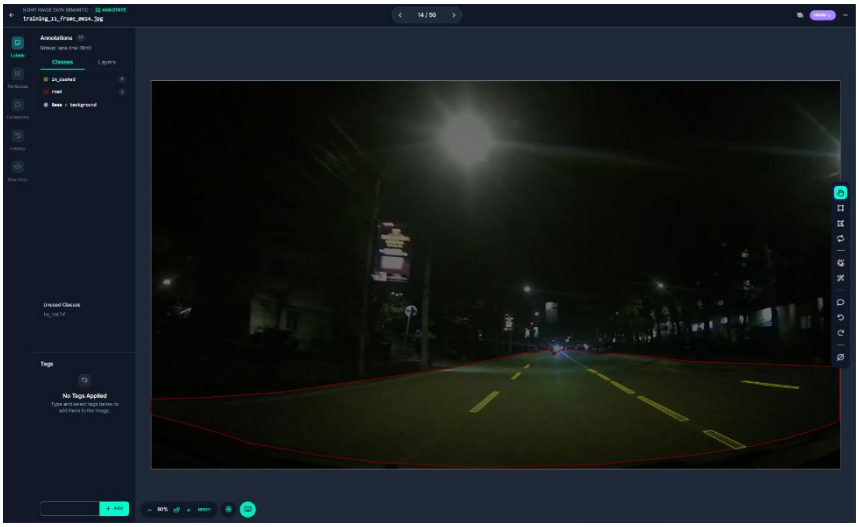
2.5.1 Data preparation

Proses preparasi data (*data preparation)*, sebagaimana diilustrasikan pada Gambar 18, merupakan tahapan fundamental dalam penelitian ini yang terdiri dari beberapa langkah sekuensial. Alur kerja preparasi data ini dapat dirinci sebagai berikut:

Nighttime Video input. Tahap ini diawali dengan penggunaan data masukan berupa rekaman video mentah. Video ini merupakan data primer yang diperoleh dari hasil akuisisi menggunakan dashcam selama berkendara pada kondisi malam hari di berbagai ruas jalan yang telah dijelaskan sebelumnya.

Ekstraksi *frame* (*extract frames*). Rekaman video tersebut kemudian diproses untuk mengekstrak *frame* citra individual. Diterapkan laju pengambilan sampel (*sampling rate*) sebesar dua *frame* per detik (2 FPS). Pemilihan laju ini bertujuan untuk menjaga variasi dan keunikan (*uniqueness*) data, sekaligus mengurangi redundansi antar *frame* yang berurutan.

Anotasi (*Annotation*). *Frame* citra yang telah diekstrak selanjutnya melalui proses anotasi (pelabelan) manual untuk membuat *ground truth*. Proses ini dilakukan menggunakan *platform* Roboflow dengan memanfaatkan fitur *instance segmentation*. Hasil anotasi kemudian diekspor dalam format PASCAL VOC, yang kemudian dikonversi menjadi mask segmentasi biner yang merepresentasikan area lajur jalan secara presisi.



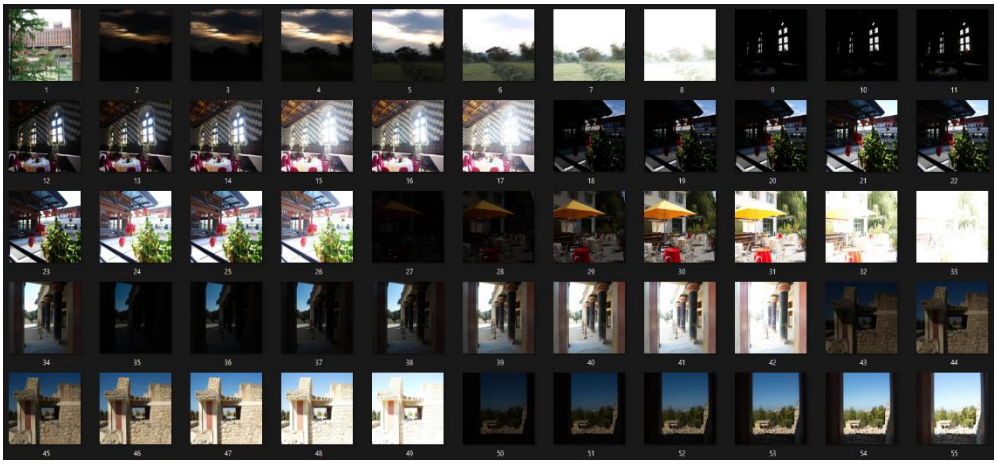
Gambar 19. Proses anotasi data menggunakan platform Roboflow

Pembagian dataset (*divide into train, valid, and test*). Dataset yang telah dianotasi kemudian dibagi menjadi tiga himpunan: pelatihan (*training*), validasi (*valid*), dan pengujian (*test*). Strategi pembagian data ini diatur secara spesifik berdasarkan lokasi (ruas jalan). Ruas jalan yang dialokasikan untuk himpunan pengujian (data *testing*) sepenuhnya berbeda dari ruas jalan yang digunakan untuk himpunan pelatihan (data *training*). Pendekatan ini memastikan bahwa model dievaluasi pada data yang benar-benar baru dan belum pernah ditemui selama fase pelatihan model, sehingga pengujian dapat menilai kemampuan generalisasi model secara lebih objektif.

Himpunan data ini setelah dilakukan proses-proses *data preparation* di atas menghasilkan jumlah data sebanyak: *train* berjumlah 206, *valid* berjumlah 68, dan *test* berjumlah 30 dengan kondisi jalan yang diterangi cahaya lampu jalan dan kondisi jalan yang tidak diterangi cahaya lampu jalan serta 3 video yang akan digunakan untuk melakukan uji perbandingan kemampuan model dalam melakukan inferensi segmentasi langsung pada video tersebut. Himpunan data ini (direpresentasikan sebagai dataset) kemudian disimpan dan siap digunakan untuk tahap pelatihan model, yang sekaligus mengakhiri tahap preparasi data.

2.5.2 Tahap *isolated Zero-DCE++ training*

Tahap ini merupakan proses pendahuluan yang fundamental, karena pelatihannya dilakukan secara terpisah dari arsitektur deteksi lajur utama. Proses ini diawali dengan ZeroDCE++ *data input*, yang memanfaatkan dataset sekunder berjumlah 2002 yang disediakan oleh pengembang asli algoritma Zero-DCE++.



Gambar 20. Sampel dataset sekunder Zero-DCE++

Data masukan tersebut kemudian diproses oleh arsitektur Zero-DCE++. Tujuan utama dari pelatihan ini adalah agar model dapat belajar secara *unsupervised* (tanpa pengawasan) untuk mengestimasi kurva peningkatan luminansi. Dengan kata lain, model ini dilatih untuk secara otomatis untuk meningkatkan visibilitas fitur dan kecerahan (luminansi) pada citra masukan, tanpa memerlukan data *ground truth* berupa pasangan gambar terang (Zhang et al., 2024).

Hasil dari proses pelatihan ini adalah sebuah model, yang berisi bobot (*weights*) yang telah terlatih. Bobot inilah yang kemudian dimuat sebagai bobot pra-latih (*pretrained weights*) untuk *Low-Light Enhancement Module* dalam arsitektur deteksi lajur utama.

2.5.3 Tahap *end-to-end training*

Tahapan *End-to-End Training* merupakan alur kerja inti dari penelitian ini. Pada tahap ini, arsitektur ini dilatih secara *end-to-end* dari masukan data hingga keluaran prediksi. Tahapan ini, sebagaimana ditunjukkan pada Gambar 18, terdiri dari beberapa langkah:

1. *Training and Valid Data Input*

   Proses pelatihan *end-to-end* diawali dengan *training and valid data input*. Data yang digunakan bersumber dari dataset (himpunan data latih dan data validasi) yang telah dihasilkan pada tahap *data preparation*. Setiap sampel data terdiri dari sepasang citra masukan (gambar minim cahaya) dan

mask anotasi *ground truth* yang merepresentasikan lajur jalan. Data ini dimuat ke dalam memori secara *batch* untuk memulai iterasi pelatihan.

2. *Preprocessing*

Setiap citra masukan dan *mask* anotasi yang dimuat akan melewati tahap *preprocessing*. Tahap ini sangat penting untuk variabilitas data dan kesiapan model. Proses ini mencakup serangkaian transformasi yang diterapkan secara *real-time*, seperti augmentasi data untuk meningkatkan robustisitas model terhadap variasi kondisi dan konversi citra/mask ke format *PyTorch Tensor* agar kompatibel dengan arsitektur jaringan.

Resize. Pada proses ini setiap citra masukan diubah ukurannya secara seragam ke dimensi 256 x 448 piksel. Langkah ini memiliki dua tujuan strategis yang krusial. Pertama, *resize* berfungsi sebagai standarisasi dimensi, yang merupakan prasyarat teknis untuk memastikan bahwa semua *tensor* masukan dalam satu *batch* memiliki ukuran yang konsisten (*uniform*) saat diproses oleh arsitektur *deep learning*. Kedua, yang tidak kalah penting, adalah untuk efisiensi komputasi. Dengan mengurangi resolusi citra mentah ke dimensi yang lebih kecil, jumlah total piksel yang harus diproses oleh setiap lapisan *convolutional* berkurang secara substansial. Pengurangan ini secara langsung berkontribusi pada percepatan waktu pelatihan per *epoch* dan penurunan konsumsi memori (VRAM) secara signifikan, sehingga memungkinkan proses eksperimen yang lebih efisien.

(a) (b)

Gambar 21. Contoh hasil *resize*: (a) sebelum *resize;* (b) setelah *resize*

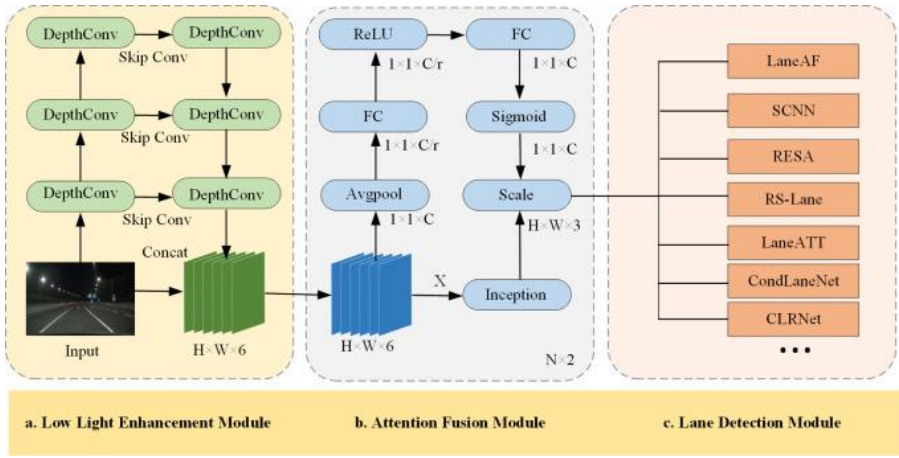
Horizontal flip. Proses ini mencerminkan gambar secara horizontal di sepanjang sumbu vertikalnya. Dalam konteks deteksi lajur jalan, augmentasi ini sangat efektif untuk menambahkan variasi data geometris. Sebagai contoh, data citra yang menampilkan tikungan ke kiri akan ditransformasi menjadi simulasi tikungan ke kanan. Dengan melatih model pada data asli sekaligus data cermin, model "dipaksa" untuk mempelajari representasi fitur lajur (seperti marka atau batas jalan) yang invarian (tidak terpengaruh) terhadap orientasi arah. Hal ini

secara signifikan meningkatkan kemampuan generalisasi model terhadap skenario jalan yang beragam, seperti tikungan ke arah yang berbeda.

(a) (b)

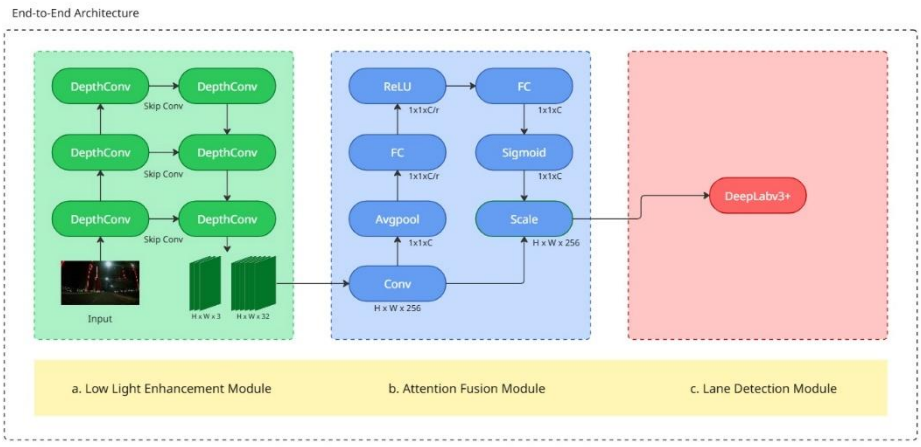
Gambar 22. Contoh hasil horizontal *flip*: (a) sebelum horizontal *flip*; (b) setelah horizontal *flip*

3. *End-to-End Architecture*



Gambar 23. Arsitektur FLLENet
Sumber: (Zhang et al., 2024)

Arsitektur *end-to-end* yang diusulkan, seperti ditunjukkan pada Gambar 24, dirancang sebagai kerangka kerja terintegrasi untuk mengatasi tantangan deteksi lajur jalan dalam skenario minim cahaya pada malam hari. Arsitektur ini terdiri dari tiga modul sekuensial: (a) *Low Light Enhancement Module* (LLEM) untuk prapemrosesan pencerahan gambar, (b) *Attention Fusion Module* (AFM) untuk mempertahankan fitur-fitur pada citra, dan (c) *Lane Detection Module*, yang dalam penelitian ini diimplementasikan menggunakan *Modified DeepLabv3+* untuk segmentasi semantik.



Gambar 24. Arsitektur *end-to-end*

a. *Low light enhancement module* (LLEM)

Modul ini berfungsi sebagai komponen fundamental dalam arsitektur. Tujuan utamanya adalah untuk meningkatkan luminansi (kecerahan) dan kualitas visual dari citra masukan yang diambil dalam kondisi minim cahaya. Pada kondisi gelap, fitur-fitur penting seperti marka lajur menjadi sulit terlihat, sehingga LLEM dirancang untuk memperjelas fitur-fitur tersebut sebelum diproses lebih lanjut (Zhang et al., 2024).

Desain modul ini terinspirasi dari prinsip Zero-DCE (Guo et al., 2020, seperti dikutip dalam Zhang et al., 2024), yang bekerja dengan mempelajari sekumpulan "kurva peningkatan luminansi" (LE-*curve*) untuk menyesuaikan nilai piksel citra masukan secara presisi. Rumus dasar untuk kurva LE orde pertama dapat diekspresikan sebagai (Zhang et al., 2024, Eq. 1):

$$LE(I(x);\, \alpha) = I(x) + \alpha I(x)(1 - I(x)) \tag{24}$$

di mana $I(x)$ adalah nilai piksel pada citra masukan, dan α adalah parameter kurva yang dapat dilatih ($\alpha \in [-1, 1]$) yang mengontrol tingkat eksposur. Untuk memberikan penyesuaian yang lebih adaptif terhadap berbagai kondisi pencahayaan yang kompleks, pendekatan ini

dapat diterapkan secara rekursif (berulang). Arsitektur LLEM kemudian menyempurnakan ini dengan menjadikan parameter kurva bersifat *pixel-wise* (spesifik untuk tiap piksel), bukan global. Dengan demikian, penyesuaian n-iterasi dinyatakan sebagai (Zhang et al., 2024):

$$LE_n(x) = LE_{n-1}(x) + \mathcal{A}_n(x)LE_{n-1}(x)\big(1 - LE_{n-1}(x)\big) \tag{25}$$

di mana $\mathcal{A}_n$ adalah parameter map yang dipelajari oleh jaringan untuk iterasi ke-n. Ini memungkinkan setiap piksel memiliki kurva penyesuaiannya sendiri, sehingga menghindari peningkatan berlebih (*over-enhancement*) atau kekurangan (*under-enhancement*) pada area lokal. Sesuai diagram, modul LLEM ini diimplementasikan menggunakan enam lapisan *depth-separable convolution* (*DepthConv*) dengan *skip connections* (*Skip Conv*). Keluaran dari modul ini adalah *enhancement map* 3-*channel* ($H \times W \times 3$) yang akan digunakan untuk konfigurasi kecerahan dari citra dan *feature map* 32-*channel* ($H \times W \times 32$), yang merupakan fitur-fitur pada citra dari layer yang paling dangkal untuk mempertahankan kualitas fitur spasial yang akan diterima sebagai input dari *Attention Fusion Module* (AFM).

b. Attention Fusion Module (AFM)

Setelah LLEM menghasilkan *feature map 32-channel*, *Attention Fusion Module* (AFM) bertugas untuk menyeimbangkan dan memperkuat *feature map* tersebut. Yang dimana, marka lajur jalan memiliki karakteristik visual yang sempit dan memanjang, sehingga rentan hilang atau terdistorsi oleh *noise* dalam kondisi minim cahaya. Oleh karena itu, AFM diperkenalkan untuk mengekstrak fitur lajur secara efektif dengan menerapkan mekanisme atensi *channel* (*channel attention*), yang desainnya mengacu pada blok *Squeeze-and-Excitation* (SE). Proses di dalam AFM dapat diuraikan melalui tiga operasi matematis (Zhang et al., 2024, Eq. 4, 5, 6):

***Squeeze* (Avgpool).** Operasi *squeeze* mengagregasi informasi spasial global menjadi deskriptor *channel*. Ini dicapai dengan menggunakan *global average pooling* (Avgpool) pada *feature map* masukan u_c (di mana c adalah *channel*) (Zhang et al., 2024).

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W} u_c(i,j) \tag{26}$$

Operasi ini menghasilkan vektor z $(1 \times 1 \times C)$, di mana setiap nilai z_c merepresentasikan informasi global dari *channel* ke-c (Zhang et al., 2024).

***Excitation* (FC → ReLU → FC → Sigmoid).** Operasi *excitation* bertujuan untuk menangkap dependensi antar-*channel* dengan menggunakan dua lapisan *Fully Connected* (FC) dan fungsi aktivasi non-linear (Zhang et al., 2024).

$$s = F_{ex}(z, W) = \sigma\big(W_2 \delta(W_1 z)\big) \tag{27}$$

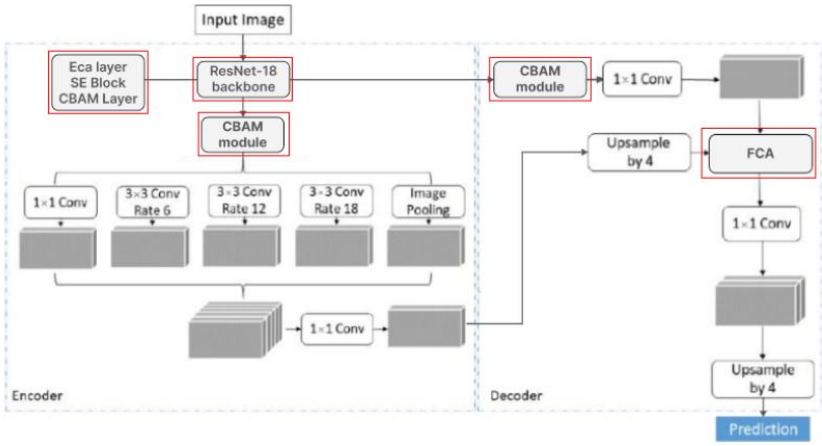
di mana δ adalah fungsi aktivasi ReLU, σ adalah fungsi aktivasi Sigmoid, W_1 dan W_2 adalah bobot dari lapisan FC. Vektor keluaran s berisi bobot (atensi) untuk setiap *channel*, yang bernilai antara 0 dan 1 (Zhang et al., 2024).

***Scale* (Penskalaan).** Langkah terakhir adalah merekalibrasi *feature map* masukan asli u_c dengan mengalikannya secara *element-wise* dengan bobot atensi s_c yang telah dipelajari.

$$\tilde{X}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c \tag{28}$$

Operasi ini (*scale*) memperkuat *channel* fitur yang informatif (penting untuk deteksi lajur) dan menekan *channel* yang kurang relevan (Zhang et al., 2024). Keluaran dari AFM adalah feature map $H \times W \times 256$ yang telah dilakukan penguatan melalui blok atensi dan siap untuk diproses oleh modul segmentasi untuk digabungkan dengan output dari *backbone* modul segmentasi.

c. Lane Detection Module



Gambar 25. Arsitektur *Modified DeepLabv3+*
Sumber: (Mudatsir, 2024)

Komponen terakhir dalam arsitektur *end-to-end* ini adalah *Lane Detection Module*. Berbeda dari implementasi standar, modul ini mengadopsi arsitektur *DeepLabv3+* yang telah dimodifikasi, hasil

pengembangan berdasarkan kajian literatur ekstensif dan temuan penelitian sebelumnya (Mudatsir, 2024). Tujuan utama dari modifikasi ini adalah untuk mengoptimalkan kapabilitas ekstraksi fitur dan meningkatkan presisi segmentasi lajur jalan. Peningkatan ini secara spesifik dicapai melalui integrasi berbagai mekanisme atensi, sebagaimana diilustrasikan pada Gambar 25 (Mudatsir, 2024, hlm. 58).

Kerangka kerja fundamental dari modul ini tetap mengadopsi arsitektur *encoder-decoder DeepLabv3+* (Mudatsir, 2024). Pada bagian *encoder*, digunakan *backbone ResNet-18* (B. Li & Gu, 2023, seperti dikutip dalam Mudatsir, 2024) yang berfungsi untuk mengekstraksi representasi fitur secara hierarkis dari citra masukan. Fitur-fitur yang dihasilkan ini selanjutnya diolah oleh modul *Atrous Spatial Pyramid Pooling* (ASPP), yang dirancang untuk menangkap informasi kontekstual pada berbagai skala (*multi-scale*). Hal ini dicapai melalui penggunaan *atrous convolution* dengan *rate* (tingkat dilasi) yang bervariasi (6, 12, 18), serta dilengkapi dengan *image pooling* untuk menangkap konteks global (Mudatsir, 2024). Selanjutnya, bagian decoder secara efektif menggabungkan fitur kaya konteks yang berasal dari ASPP dengan fitur *low-level* (yang mempertahankan detail spasial) dari *encoder*, sehingga memungkinkan rekonstruksi *mask* segmentasi yang presisi dan detail (Mudatsir, 2024).

Arsitektur *DeepLabv3+* standar dimodifikasi dengan penambahan beberapa modul atensi pada titik-titik strategis untuk meningkatkan representasi fitur (Mudatsir, 2024):

1) Integrasi atensi pada *backbone* (*ResNet*-18):

   Serangkaian modul atensi disisipkan setelah blok *backbone* ResNet-18 untuk menyempurnakan fitur-fitur awal yang diekstraksi. Modul-modul ini meliputi:

   ECA layer (Efficient Channel Attention). Lapisan ECA (Mudatsir, 2024) ditambahkan untuk menangkap interaksi *cross-channel* lokal secara efisien tanpa memerlukan reduksi dimensi, memungkinkan pembelajaran bobot atensi *channel* yang adaptif.

   SE block (Squeeze-and-Excitation). Blok SE (Hu et al., 2018, seperti dikutip dalam Mudatsir, 2024) diintegrasikan untuk secara eksplisit memodelkan interdependensi antar *channel* fitur. Blok ini melakukan agregasi informasi global (*squeeze*) dan kemudian merekalibrasi respons channel (*excitation*).

   CBAM layer (Convolutional Block Attention Module). Lapisan CBAM (H. Li et al., 2023, seperti dikutip dalam Mudatsir, 2024)

diterapkan untuk menghasilkan peta atensi secara sekuensial pada dimensi *channel* dan spasial, memungkinkan model untuk fokus pada fitur 'apa' dan 'di mana' yang paling informatif. Penambahan ketiga mekanisme atensi ini secara kolektif bertujuan untuk memperkaya dan menyaring fitur-fitur dasar sebelum diproses lebih lanjut oleh modul ASPP (Mudatsir, 2024).

2) Integrasi atensi pada *encoder* (sebelum ASPP):

CBAM layer (Convolutional Block Attention Module). Modul CBAM kedua ditempatkan setelah rangkaian atensi pada backbone dan sebelum fitur dimasukkan ke dalam ASPP. Tujuannya adalah untuk melakukan penyempurnaan fitur lebih lanjut, memastikan bahwa representasi fitur yang paling relevan (baik secara channel maupun spasial) yang akan diolah oleh ASPP (Mudatsir, 2024).

3) Integrasi atensi pada *decoder*:

CBAM layer (Convolutional Block Attention Module). Modul CBAM ketiga diterapkan pada fitur *low-level* yang diambil dari *encoder* sebelum proses fusi di *decoder*. Hal ini membantu *decoder* untuk secara selektif fokus pada detail spasial yang paling signifikan dari fitur resolusi tinggi (Mudatsir, 2024).

***Feature Cross Attention (FCA)*.** Setelah fitur *low-level* (yang telah melalui CBAM) dan fitur *high-level* (dari ASPP yang telah di-upsample) digabungkan, modul FCA diterapkan (Mudatsir, 2024). Modul FCA (Zeng et al., 2020, seperti dikutip dalam Mudatsir, 2024) dirancang untuk memfusikan informasi spasial dan kontekstual secara efektif dengan memanfaatkan mekanisme atensi silang (*cross-attention*) antara fitur spasial dan fitur *channel*, menghasilkan representasi fitur yang lebih kaya sebelum tahap prediksi akhir.

Setelah melalui serangkaian lapisan konvolusi dan operasi upsampling akhir di dalam *decoder* (termasuk modul FCA), *Lane Detection Module* ini menghasilkan peta segmentasi akhir (*prediction*). Peta keluaran ini mengklasifikasikan setiap piksel ke dalam salah satu dari empat kelas yang telah ditentukan sebelumnya (yaitu, *background*, jalan, marka garis membujur penuh, dan marka garis membujur putus-putus), sehingga menyediakan representasi visual yang terperinci mengenai lokasi lajur jalan pada citra masukan.

Dengan mengintegrasikan berbagai mekanisme atensi ini pada titik-titik strategis, arsitektur *DeepLabv3+* yang dimodifikasi diharapkan memiliki kemampuan yang lebih baik dalam menangani variabilitas fitur

lajur jalan dan mencapai akurasi segmentasi yang lebih tinggi dibandingkan dengan arsitektur *DeepLabv3+* standar.

4. *Pixel Level Classification*

   Arsitektur *end-to-end* menghasilkan keluaran yang kemudian diproses sebagai tugas klasifikasi tingkat piksel (*pixel level classification*). Pada dasarnya, ini adalah tahap di mana model menghasilkan *mask* prediksi (*prediction mask*) yang memiliki resolusi spasial yang sama dengan *mask ground truth*. *Mask* prediksi ini kemudian dibandingkan dengan *mask ground truth* (dari *Data Input*) untuk menghitung nilai *loss*. Fungsi *loss* yang digunakan mengkuantifikasi ketidaksesuaian antara prediksi dan label sebenarnya, yang menjadi sinyal utama untuk proses optimisasi bobot.

2.5.4 Tahap *end-to-end testing*

Tahapan *End-to-End Testing* merupakan alur kerja intiyang menjadi proses pengujian dari performa model yang telah di-*training* sebelumnya. Pada tahap ini, dilakukan evaluasi performa arsitektur yang telah dilatih sebelumnya secara *end-to-end*. Tahapan ini, sebagaimana ditunjukkan pada Gambar 18, terdiri dari beberapa langkah:

1. *Data Input*

   Proses pengujian *end-to-end* diawali dengan tahap pemasukan data (*data input*). Data yang digunakan bersumber dari himpunan data uji *(test set)*, yang telah disiapkan pada tahap preparasi data. Setiap sampel dalam himpunan ini terdiri dari sepasang citra masukan dan *mask* anotasi *ground truth* yang merepresentasikan lajur jalan. Himpunan data uji ini secara fundamental berbeda dari data yang telah digunakan selama fase pelatihan (*training*), dan secara sengaja mencakup skenario atau kondisi pencahayaan yang bervariasi. Penggunaan data yang "asing" (*unseen data*) ini sangat krusial untuk mengukur kemampuan generalisasi model secara objektif terhadap data baru.

2. *Preprocessing*

   Setiap citra masukan dan *mask* anotasi dari himpunan data uji (*test set*) juga akan melewati tahap *preprocessing* sebelum dievaluasi oleh model. Tahap ini sangat krusial untuk memastikan bahwa data masukan memiliki format yang konsisten dengan data yang digunakan selama pelatihan. Berbeda tahap pelatihan, pada tahap evaluasi ini tidak diterapkan augmentasi data (seperti *random crop* atau *horizontal flip*). Hal ini bertujuan agar model diuji pada data asli yang tidak dimodifikasi, sehingga memberikan evaluasi kinerja yang objektif. Satu-satunya transformasi spasial yang diterapkan adalah *resize*, di mana setiap citra uji diubah ukurannya secara seragam ke dimensi 448×256 piksel, agar sesuai secara presisi dengan dimensi masukan yang diterima model selama pelatihan.

3. *End-to-End Architecture*

   Setelah arsitektur *end-to-end* (LLEM + AFM + *DeepLabv3+* Modifikasi) selesai dilatih, model dievaluasi menggunakan alur kerja inferensi yang juga bersifat *end-to-end*. Pada tahap ini, citra masukan dari *dataset* uji (*test set*) dilewatkan melalui keseluruhan modul secara sekuensial dimulai dari *Low-Light Enhancement Module* (LLEM), dilanjutkan ke *Attention Fusion Module* (AFM), dan diakhiri pada *Lane Detection Module* (*DeepLabv3+* yang telah dimodifikasi untuk menghasilkan peta segmentasi prediktif akhir. Kinerja model kemudian diukur dengan membandingkan hasil prediksi ini terhadap *mask ground truth* menggunakan metrik *Intersection over Union* (IoU) dan *Dice Score*.

4. *Ablation Study*

   Selanjutnya, untuk menganalisis kontribusi spesifik dan ketahanan (*robustness*) dari arsitektur yang diusulkan, sebuah studi ablasi (*ablation study*) tambahan dilakukan. Studi ablasi ini difokuskan secara khusus untuk menguji pengaruh variasi resolusi citra masukan terhadap kinerja model. Tujuan utama dari pengujian ini adalah untuk mengevaluasi seberapa baik model yang diusulkan (dengan LLEM dan AFM) mampu mempertahankan akurasinya ketika dihadapkan pada pengecilan ukuran (degradasi resolusi) citra. Kinerja robustisitas model ini kemudian dibandingkan secara langsung (*head-to-head*) dengan model baseline dari penelitian sebelumnya (Mudatsir, 2024), yang hanya mengimplementasikan arsitektur *DeepLabv3+* standar (tanpa integrasi modul LLEM dan AFM).