BAB I
PENDAHULUAN

1.1 Latar Belakang

Infrastruktur jalan memiliki peran krusial dalam mendukung sistem transportasi, mendorong pertumbuhan wilayah, serta mewujudkan pemerataan hasil pembangunan. Namun, kerusakan jalan dapat mengganggu fungsi utama infrastruktur ini dan menimbulkan kerugian dari berbagai aspek. Jalan berlubang merupakan salah satu permasalahan utama yang masih sering ditemukan pada jalan di Indonesia. Lubang jalan dapat disebabkan oleh berbagai faktor seperti faktor cuaca, beban kendaraan yang berlebihan, dan kurangnya pemeliharaan yang memadai (Jakubec et al., 2023). Lubang dapat menyebabkan dampak negatif, seperti menyebabkan kendaraan kehilangan kontrol dan mengakibatkan kecelakaan lalu lintas, terutama jika pengendara tidak dapat menghindari lubang tersebut. Kerusakan pada kendaraan juga dapat terjadi apabila kendaraan melintasi lubang dengan kecepatan tinggi, sehingga hal ini dapat menyebabkan menurunnya kenyamanan dan efisiensi berkendara.

Teknologi visi komputer telah mengalami perkembangan yang pesat dan telah dimanfaatkan secara luas pada berbagai bidang, salah satunya dalam pemantauan infrastruktur jalan (Indrabayu et al., 2023). Sistem ini memanfaatkan teknik pemrosesan citra digital untuk menganalisis data visual yang diperoleh melalui kamera. Dengan kemampuan mengenali pola dan fitur dari gambar atau video, visi komputer dapat digunakan untuk mendeteksi lubang jalan secara otomatis (Egaji et al., 2021). Deteksi lubang jalan berbasis visi komputer sangat penting untuk memastikan keselamatan lalu lintas. Selain itu, sistem ini juga dapat mendukung upaya pemeliharaan infrastruktur yang lebih efektif dan efisien, membantu mencegah kerusakan kendaraan, serta berpotensi untuk mengurangi biaya pemeliharaan jalan melalui deteksi dini dan penanganan yang lebih terencana (Ling et al., 2024).

Salah satu faktor yang dapat berpengaruh terhadap performa sistem dalam mendeteksi lubang yaitu kondisi pencahayaan lingkungan di sekitarnya. Deteksi lubang jalan pada siang hari cenderung lebih mudah karena pencahayaan yang cukup sehingga lubng pada permukaan jalan dapat terekam dengan jelas dan mudah untuk dideteksi. Namun, pada malam hari deteksi lubang jalan menghadapi berbagai tantangan yang kompleks yang dapat menyebabkan lubang jalan menjadi lebih sulit untuk terdeteksi. Salah satu tantangannya yaitu pencahayaan yang rendah (*low light*) ataupun pencahayaan yang tidak merata di malam hari (Ling et al., 2024). *Low light* dapat disebabkan oleh beberapa faktor seperti lampu jalan yang redup atau bahkan area jalan yang tidak memiliki lampu penerangan. Selain itu, lubang jalan yang tertutup oleh bayangan dari pohon atau bangunan di sekitar jalan yang menghalangi cahaya dari lampu penerangan sehingga menyebabkan area lubang yang tertutup bayangan tersebut menjadi lebih gelap. Kondisi pencahayaan yang rendah ini dapat menurunkan kualitas gambar yang diambil oleh kamera, sehingga berpotensi menyebabkan hilangnya detail pada citra dan berpotensi menurunkan kinerja sistem deteksi berbasis visi komputer (Rasheed et al., 2023).

Deteksi objek berbasis *deep learning* telah banyak digunakan dalam berbagai penelitian untuk melakukan deteksi lubang jalan yang akurat dan efisien. Salah satunya yaitu, algoritma *You Only Look Once* (YOLO) yang dikenal mampu melakukan proses deteksi secara cepat dengan akurasi yang tinggi. Sejumlah penelitian menunjukkan bahwa algoritma YOLO efektif dalam mendeteksi lubang jalan pada berbagai kondisi. Penelitian yang dilakukan oleh (Jakubec et al., 2023) menemukan bahwa YOLO menunjukkan respons deteksi yang lebih cepat dan akurasi yang lebih tinggi dalam mendeteksi lubang jalan pada kondisi cuaca cerah, hujan, dan malam hari. Selain itu, penelitian yang dilakukan (Asad et al., 2022) memperoleh hasil bahwa algoritma YOLO memiliki kinerja yang lebih unggul dalam mendeteksi lubang dibandingkan dengan algoritma *SSD-Mobilenet*.

Seiring dengan pembaruan pada arsitektur YOLO, algoritma ini terus mengalami perkembangan hingga saat ini seri YOLO telah mencapai versi ke-11. Studi yang dilakukan oleh (Zanevych et al., 2024) mengungkapkan bahwa YOLOv11 mencapai kinerja *mAP* yang lebih unggul dalam mendeteksi lubang jalan dibandingkan dengan model deteksi lain yang diuji. Pada penelitian yang dilakukan oleh (Ji & Jiang, 2025) menyatakan bahwa peningkatan performa YOLOv11 didukung oleh pengembangan arsitektur jaringan yang lebih efisien dalam mengekstraksi fitur multi-skala, sehingga model ini mampu mengenali objek dengan variasi ukuran dan kondisi visual yang kompleks. Keunggulan ini menunjukkan bahwa YOLOv11 berpotensi sebagai algoritma yang dapat diterapkan pada sistem deteksi lubang jalan berbasis visi komputer.

Untuk mengatasi tantangan pencahayaan rendah (*low light*), penelitian ini mengusulkan pendekatan *Adaptive Gamma Correction* (AGC) dan *Contrast Limited Adaptive Histogram Equalization* (CLAHE) sebagai tahapan awal sebelum proses deteksi lubang jalan dilakukan. AGC diterapkan untuk menyesuaikan tingkat kecerahan citra berdasarkan kondisi pencahayaan pada citra secara adaptif dengan mempertimbangkan karakteristik pencahayaan pada setiap citra masukan. Teknik ini menyeimbangkan distribusi piksel secara menyeluruh sehingga peningkatan kecerahan tetap stabil tanpa menimbulkan efek *overexposure* yang berpotensi menghilangkan informasi penting pada permukaan jalan (Rahman et al., 2016). Pendekatan tambahan CLAHE diterapkan untuk meningkatkan kontras citra sehingga dapat mempertajam detail citra secara lebih lokal. Metode ini membatasi peningkatan kontras secara berlebihan sehingga detail citra yang ingin dipertahankan tetap tampak jelas dan meminimalkan munculnya noise berlebih yang sering terjadi pada citra dengan pencahayaan rendah (Chen et al., 2023).

Kombinasi antara AGC dan CLAHE diharapkan dapat menghasilkan citra dengan kualitas visual yang lebih optimal, baik dari segi kecerahan maupun kontras, sehingga karakteristik visual lubang jalan dapat ditampilkan secara lebih jelas dan informatif. Peningkatan kualitas citra ini berperan penting dalam mendukung proses ekstraksi fitur oleh model deteksi berbasis *deep learning*, khususnya pada kondisi malam hari yang memiliki keterbatasan pencahayaan. Dengan citra masukan yang lebih representatif, sistem deteksi diharapkan mampu mengenali pola dan ciri visual lubang jalan secara lebih akurat dan konsisten, sehingga meningkatkan kinerja keseluruhan sistem deteksi lubang jalan.

1.2 Landasan Teori

1.2.1 Lubang Jalan (*Pothole*)

Lubang *(pothole)* merupakan salah satu kerusakan jalan berupa cekungan di permukaan jalan. Kerusakan ini umumnya terjadi akibat beberapa faktor seperti, tekanan beban kendaraan yang melewati jalan, kondisi cuaca yang ekstrem, serta kualitas konstruksi dan material jalan yang kurang memadai (Yik et al., 2021). Beban kendaraan yang terus-menerus dapat menyebabkan kelelahan material perkerasan, sehingga struktur jalan menjadi rentan terhadap retak dan deformasi. Retakan tersebut kemudian memungkinkan air masuk ke lapisan bawah perkerasan, yang selanjutnya mempercepat proses degradasi struktur jalan.

Pengaruh lingkungan juga memiliki peran signifikan dalam pembentukan lubang jalan. Air hujan yang menggenang pada permukaan jalan dapat meresap ke dalam retakan kecil dan melemahkan daya dukung lapisan perkerasan. Pada kondisi tertentu, perubahan suhu yang ekstrem dapat menyebabkan proses penyusutan material jalan, sehingga memperbesar retakan yang telah ada dan mempercepat terbentuknya lubang (Parvin et al., 2025). Kombinasi antara air, beban kendaraan, dan kelemahan material ini menjadikan lubang jalan sebagai bentuk kerusakan yang sulit dihindari, terutama pada jalan dengan tingkat lalu lintas yang tinggi.

Selain itu, jika pemeliharaan atau perbaikan jalan tidak dilakukan secara rutin, hal tersebut dapat memperparah kondisi kerusakan kecil pada jalan. Kerusakan kecil yang tidak segera ditangani dapat berkembang menjadi lubang yang lebih besar. Hal tersebut dapat menjadi masalah serius bagi pengendara dan meningkatkan risiko kecelakaan lalu lintas.

Gambar 1. Jalan Berlubang pada Malam Hari

1.2.2 Visi Komputer

Visi komputer (*Computer Vision*) merupakan salah satu bidang pada *Artifcial Intelligence* (AI) yang memproses gambar atau video. Visi komputer berfokus pada kemampuan komputer untuk memahami data visual dengan pemahaman tingkat tinggi. Hal ini memungkinkan komputer meniru kemampuan sistem visual manusia dalam menangkap informasi dari lingkungan sekitar, kemudian menginterpretasikannya menjadi pengetahuan yang bermakna. Selain itu, visi komputer dapat mengenali, menganalisis,

dan mengambil keputusan berdasarkan informasi visual yang diberikan (Szeliski, 2022). Proses ini mencakup berbagai tahapan, mulai dari akuisisi data visual, pemrosesan citra awal, hingga pemahaman visual tingkat lanjut yang melibatkan analisis konteks dan hubungan antar objek. Dengan kemampuan tersebut, visi komputer memungkinkan sistem untuk melakukan pengenalan pola, identifikasi objek, serta analisis kondisi visual secara sistematis. Selain itu, visi komputer juga berperan dalam mendukung proses pengambilan keputusan berbasis data visual, sehingga sistem dapat merespons kondisi lingkungan secara lebih akurat dan objektif. Oleh karena itu, visi komputer menjadi komponen penting dalam pengembangan transportasi cerdas yang membutuhkan persepsi visual sebagai dasar pemahaman dan interaksi dengan dunia nyata.

Seiring perkembangan teknologi pembelajaran berbasis data, kemampuan visi komputer terus mengalami peningkatan yang siginifikan. Penerapan berbagai pendekatan *deep learning* telah mendorong visi komputer tidak hanya terbatas pada ekstraksi fitur visual sederhana, tetapi juga mampu melakukan tugas yang lebih kompleks. Tugas-tugas tersebut seperti deteksi objek, pelacakan objek, pengenalan wajah, dan pengambilan keputusan secara otomatis. Kemampuan tersebut memungkinkan sistem untuk memahami informasi visual secara lebih menyeluruh, adaptif, dan akurat dalam menghadapi variasi kondisi lingkungan yang dinamis (Safyari et al., 2024). Dan hal ini juga menjadikan visi komputer banyak diterapkan pada berbagai bidang, termasuk pada sistem pengawasan kondisi jalan. Dalam hal ini visi komputer digunakan untuk memperoleh informasi kondisi permukaan jalan, seperti lubang jalan yang berpotensi membahayakan pengguna jalan. Pemanfaatan visi komputer dalam pengawasan kondisi jalan dapat membantu pihak terkait dalam proses evaluasi, perencanaan pemeliharaan, serta pengambilan keputusan yang lebih objektif dan berbasis data. Dengan demikian, penerapan visi komputer pada sistem pengawasan kondisi jalan berkontribusi dalam meningkatkan efektivitas pemantauan infrastruktur serta mendukung upaya peningkatan keselamatan dan kualitas layanan transportasi.

1.2.3 Pengolahan Citra

Citra digital merupakan gambar yang ditangkap melalui perangkat digital, seperti kamera digital. Citra tersebut tersusun oleh elemen-elemen piksel yang masing-masing menyimpan informasi intensitas cahaya atau warna, sehingga mampu merepresentasikan kondisi nyata dalam bentuk data numerik (Yudi, 2025). Supaya informasi visual tersebut dapat dimanfaatkan secara optimal, citra digital tersebut akan diproses menggunakan serangkaian algoritma komputer atau disebut dengan pengolahan citra. Pengolahan citra berfokus untuk mengekstrak informasi atau meningkatkan kualitas citra sehingga mudah dipahami oleh sistem. Secara umum, tahapan pada pengolahan citra meliputi, peningkatan kualitas citra (*image enhancement*), reduksi kebisingan *(noise reduction*), segmentasi citra *(image segmentation)*, penajaman citra (*image sharpening*), menghilangkan blur pada citra (*deblurring*), serta berbagai proses transformasi lainnya yang disesuaikan dengan karakteristik data dan tujuan pemrosesan yang dilakukan (Marques, 2011). Proses-proses tersebut dirancang untuk meminimalkan gangguan visual sekaligus memperjelas struktur atau objek penting yang terdapat di dalam citra.

Pengolahan citra merupakan tahapan awal atau pra-pemrosesan dalam visi komputer sebelum analisis lebih lanjut seperti deteksi objek dilakukan. Tahap pra-pemrosesan ini memiliki pengaruh yang signifikan terhadap kinerja sistem secara keseluruhan, karena kualitas citra hasil pengolahan akan menentukan seberapa baik algoritma lanjutan dapat bekerja (Yan Mengying, Qin Danyang, Zhang Gengxin, Tang Huapeng, 2023). Citra dengan kualitas yang baik akan membantu algoritma dalam mengenali pola, membedakan objek dari latar belakang serta mengurangi kemungkinan terjadinya kesalahan deteksi. Hal ini menjadi semakin krusial pada kondisi lingkungan yang menantang, seperti pencahayaan rendah atau kontras yang tidak merata. Pada kondisi tersebut, citra mentah cenderung sulit merepresentasikan objek secara optimal, sehingga diperlukan teknik pengolahan citra yang mampu menyesuaikan karakteristik visual tanpa menghilangkan informasi penting Oleh karena itu, pemilihan metode pengolahan citra yang tepat dan sesuai dengan permasalahan yang dihadapi menjadi salah satu faktor penting untuk meningkatkan kinerja suatu sistem.

1.2.4 Deteksi Objek

Deteksi objek merupakan salah satu pendekatan penting dalam bidang visi komputer untuk menemukan dan mengidentifikasi keberadaan suatu objek dalam sebuah gambar. Deteksi objek digunakan untuk menentukan posisi suatu objek dengan menggunakan kotak pembatas (*bounding box*) yang mengelilingi objek tersebut (Guan et al., 2025). Tujuan dari deteksi objek untuk mendeteksi semua kemunculan objek dari satu atau beberapa kelas yang telah ditentukan sebelumnya, seperti lubang pada permukaan jalan, kendaraan, orang, dan objek lainnya. Kemampuan untuk mendeteksi banyak objek secara simultan menjadikan deteksi objek sangat krusial dalam aplikasi dunia nyata, khususnya pada sistem transportasi cerdas, pengawasan visual, serta kendaraan otonom, di mana informasi mengenai jenis dan posisi objek digunakan sebagai dasar dalam pengambilan keputusan secara otomatis.

Dalam implementasinya, deteksi objek bekerja dengan menganalisis fitur visual pada citra seperti warna, tekstur, bentuk, dan pola intensitas piksel, untuk membedakan objek target dengan latar belakang (Tian et al., 2026). Fitur-fitur visual tersebut menjadi dasar dalam proses identifikasi objek, dalam hal ini sistem harus mampu mengenali karakteristik khas suatu objek meskipun berada pada lingkungan visual yang kompleks seperti pencahayaan rendah. Model deteksi modern umumnya memanfaatkan pendekatan *deep learning,* khususnya *Convolutional Neural Netwok* (CNN), yang mampu mengekstraksi fitur secara otomatis dan hierarkis dari data citra. Pendekatan ini memungkinkan sistem untuk mendeteksi objek dalam berbagai kondisi lingkungan, termasuk perubahan pencahayaan, sudut pandang, dan skala objek, sehingga menghasilkan kinerja deteksi yang lebih andal pada kondisi dunia nyata.

1.2.5 *Adaptive Gamma Correction* (AGC)

Adaptive Gamma Correction merupakan salah satu metode untuk *image enhancement* yang bertujuan untuk memperbaiki kualitas visual citra menyesuaikan nilai parameter gamma secara dinamis berdasarkan karakteristik statistik gambar dari citra input. Hal ini

dilakukan dengan meningkatkan kecerahan tanpa menimbulkan artefak visual yang dapat merusak kualitas visual citra (Rahman et al., 2016). Metode ini dapat digunakan untuk mengatasi pencahayaan rendah (*low light*) terutama untuk meningkatkan visibilitas objek dan deteksi objek pada malam hari.

Dalam pengaplikasiannya, AGC diterapkan pada *channel Value* (V) dalam ruang warna *Hue, Saturation,* Value. *Channel* V merepresentasikan tingkat kecerahan atau intensitas citra (Bataineh, 2023). Dalam hal ini, perubahan pada *channel V* tidak mempengaruhi informasi warna *(Hue)* dan kejenuhan warna *(Saturation*) secara langsung. Proses AGC yang diawali dengan analisis histogram atau distribusi intensitas pada *channel* V untuk menentukan karakteristik kontras citra. Pendekatan ini lebih stabil dibandingkan penerapan koreksi gamma langsung pada ruang warna RGB, yang berpotensi mengubah karakteristik warna citra.

Setelah pengkategorian tersebut, akan dilakukan penerapan fungsi transformasi gamma yang berbeda untuk setiap kategori citra. Untuk citra yang memiliki kontras rendah, gamma akan diatur dengan menggunakan fungsi logaritmik. Fungsi logaritmik mampu memperluas nilai intensitas pada area gelap sehingga detail permukaan jalan dan struktur lubang menjadi lebih terlihat. Sedangkan, untuk citra yang memiliki kontras tinggi diatur dengan menggunakan fungsi eksponensial. Fungsi transformasi berbasis eksponensial yang memungkinkan penyesuaian kecerahan secara lebih terkendali. Pendekatan ini membantu mempertahankan distribusi intensitas asli citra sekaligus memastikan bahwa detail penting, seperti kontur dan kedalaman lubang, tetap terjaga.

Dengan menggunakan fungsi transformasi yang berbeda sesuai dengan kategori kontras citra, AGC mampu meningkatkan visibilitas lubang jalan secara adaptif pada berbagai kondisi pencahayaan malam hari. Pendekatan ini tidak hanya meningkatkan kualitas visual citra, tetapi juga memperkuat representasi fitur penting yang dibutuhkan oleh model deteksi objek. Selain itu, pendekatan yang berbeda ini mampu meningkatkan kualitas citra tanpa kehilangan informasi penting dalam gambar.

1.2.6 *Contrast Limited Adaptive Histogram Equalization* (CLAHE)

Contrast Limited Adaptive Histogram Equalization merupakan metode pengembangan dari metode Adaptive Histogram Equalization (AHE). Metode ini digunakan untuk peningkatan kontras sebuah citra. Berbeda dengan *histogram equalization* konvensional yang memproses citra secara global, CLAHE dirancang untuk meningkatkan detail visual pada area-area tertentu tanpa menyebabkan peningkatan kontras yang berlebihan. Pengembangan ini dilakukan untuk mengatasi kelemahan AHE yang cenderung memperkuat noise (Chen et al., 2023), terutama pada citra dengan kondisi pencahayaan rendah.

Secara umum, CLAHE bekerja secara lokal dengan membagi citra menjadi beberapa daerah kecil yang disebut *tiles* dengan ukuran tertentu misalnya 8 kali 8. Pemilihan ukuran tile memengaruhi tingkat adaptivitas peningkatan kontras, di mana ukuran tile yang lebih kecil akan meningkatkan kontras lokal secara lebih agresif, sedangkan tile yang lebih besar menghasilkan peningkatan kontras yang lebih halus.

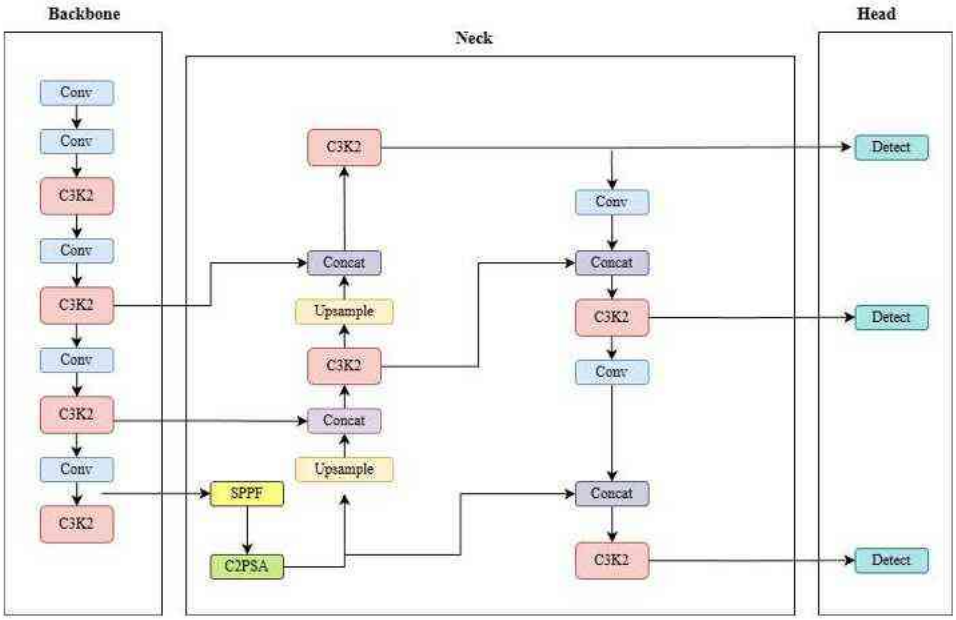
Tahap selanjutnya adalah pembentukan histogram intensitas untuk setiap tile. Metode ini memberikan nilai batas pada histogram. Nilai batas ini disebut dengan *clip*

limit yang menyatakan batas maksimum tinggi suatu histogram. Dalam hal ini *clip* limit berfungsi untuk membatasi jumlah piksel pada setiap level intensitas sehingga mencegah dominasi intensitas tertentu yang dapat menyebabkan *over* enhancement. Piksel yang melebihi batas *clip limit* dianggap sebagai *excess pixels* dan kemudian didistribusikan kembali secara merata ke seluruh level histogram pada tile tersebut sehingga distribusi intensitas menjadi lebih seimbang

Setelah histogram tiap tile dinormalisasi, nilai intensitas piksel dipetakan ulang menggunakan fungsi distribusi kumulatif (*Cumulative Distribution Function*) dari histogram yang sudah dibatasi. Pemetaan ini meningkatkan kontras lokal pada setiap tile. Karena setiap tile diproses terpisah, perbedaan kontras antar-tile bisa menimbulkan batas yang terlihat. Untuk mengatasinya, CLAHE menggunakan interpolasi bilinear antar-tile sehingga transisi intensitas menjadi halus dan citra akhir terlihat lebih natural. Mekanisme ini sangat penting pada citra jalan berlubang pada malam hari. Dalam hal ini, area gelap mendominasi citra dan rentan terhadap peningkatan noise apabila kontras ditingkatkan secara agresif.

1.2.7 *You Only Look Once Version* 11 (YOLOv11)

YOLOv11 merupakan seri terbaru pada model *You Only Look Once* (YOLO) yang dikembangkan oleh Ultralytics pada september 2024. Dilansir pada website Ultralytics bahwa model ini dirancang untuk deteksi objek secara real-time dengan akurasi, kecepatan, dan efisensi. Peningkatan utama dalam YOLOv11 meliputi arsitektur *backbone* dan *neck* untuk meningkatkan kemampuan ekstraksi fitur dalam deteksi objek yang lebih akurat dan kinerja yang lebih optimal dalam tugas yang kompleks (Zhu et al., 2025).



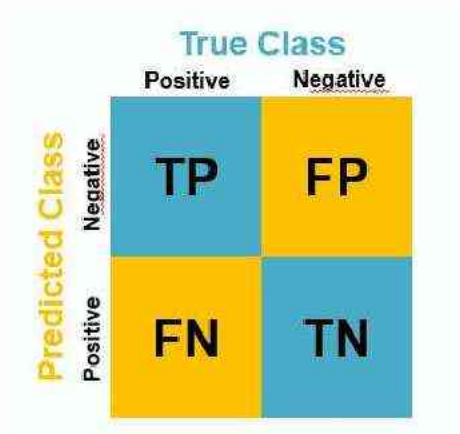
Gambar 2. Arsitektur YOLOv11

YOLOv11 memperluas dan melakukan pengembangan dari YOLOv8 dengan memperkenalkan inovasi arsitektur dan optimasi parameter untuk meningkatkan kinerja deteksi secara keseluruhan (Khanam & Hussain, 2024). Peningkatan modul arsitektur utama dalam YOLOv11 meliput: SPPF (*Spatial Pyramid Pooling Fast*), C2PSA (*Cross-Stage Partial Self-Attention*), dan C3k2 (*Convolutional Cross-Scale Kernel 2*). Blok C3k2 merupakan implementasi *Cross Stage Partial* (CSP) *Bottleneck* yang menggunakan dua konvolusi kecil yang dapat mengurangi beban secara komputasi dibandingkan satu konvolusi besar sehingga ekstraksi fitur menjadi lebih cepat. Selain itu, kemampuan blok C3k2 ini untuk mencapai kinerja tinggi dengan jumlah parameter yang lebih sedikit. SPFF tetap dipertahankan pada YOLOv11, tetapi memperkenalkan blok baru C2PSA (*Cross Stage Partial Self-Attention*) setelah SPFF. Blok C2PSA merupakan tambahan penting yang meningkatkan mekanisme *spatial attention* pada peta fitur (*feature map*). Mekanisme ini memungkinkan model untuk lebih fokus pada area yang lebih penting dalam gambar. Dengan melakukan *pooling* fitur secara spasial, blok C2PSA membantu YOLOv11 untuk berkonsentrasi pada area tertentu yang lebih relevan sehingga dapat meningkatkan akurasi deteksi terhadap objek dengan berbagai ukuran dan posisi.

1.2.8 *Confusion Matrix*

Confusion matrix digunakan untuk menyajikan informasi terkait performa model dengan memberikan perbandingan antara hasil prediksi model dan *ground truth*. (data yang sebenarnya). Terdapat 4 komponen utama dalam *confusion matrix*.

1. *True Positive* (TP)
   Data yang sebenarnya positif dan diprediksi positif oleh model.
2. *False Positive* (FP)
   Data yang sebenarnya negatif dan diprediksi positif oleh model.
3. *True Negative* (TN)
   Data yang sebenarnya negatif dan diprediksi negatif oleh model.
4. *False Negative* (FN)
   Data yang sebenarnya positif dan diprediksi negatif oleh model



Gambar 3. Confusion Matrix

Metrik evaluasi yang digunakan sebagai berikut.

1. *Precision* merupakan perbandingan antara *true positive* dengan banyaknya data yang diprediksi positif. *Precision* digunakan untuk menggambarkan tingkat ketepatan model dalam menghasilkan prediksi positif. Dalam hal ini sejauh mana objek yang terdeteksi oleh model benar-benar merupakan objek yang sesuai dengan kelas target. Dalam konteks deteksi lubang jalan, *precision* yang tinggi mengindikasikan bahwa lubang yang terdeteksi oleh sistem benar-benar merupakan lubang jalan, sehingga dapat mengurangi kesalahan deteksi terhadap objek lain yang memiliki karakteristik visual serupa.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

2. *Recall* merupakan perbandingan antara *true positive* dengan banyaknya data yang sebenarnya positif. *Recall* digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh objek yang seharusnya terdeteksi. Nilai *recall* yang tinggi sangat penting untuk memastikan bahwa sebanyak mungkin lubang jalan dapat teridentifikasi, sehingga risiko lubang yang tidak terdeteksi dapat diminimalkan.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

3. mAP (*mean average precision*) merupakan metrik evaluasi yang menggambarkan seberapa baik model bekerja secara keseluruhan di semua kelas. Metrik ini diperoleh dengan menghitung rata – rata dari *average precision* pada setiap kelas.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{3}$$

Penjelasan:

N : Jumlah total kelas

AP_i : Nilai *average precision* untuk kelas ke-*i*

1.3. Rumusan Masalah

Berdasarkan latar belakang masalah diatas, maka rumusan masalah pada penelitian ini yaitu sebagai berikut.

1. Bagaimana mendeteksi lubang pada malam hari dalam kondisi pencahayaan rendah (*low light*) dengan algoritma YOLOv11 dan menggunakan *preprocessing Adaptive Gamma Correction* dan *Contrast Limited Adaptive Histogram Equalization*?
2. Bagaimana performa algoritma YOLOv11 dalam mendeteksi lubang pada malam hari dalam kondisi pencahayaan rendah (*low light*) setelah penggunaan *preprocessing Adaptive Gamma Correction* dan *Contrast Limited Adaptive Histogram Equalization*?

1.4. Tujuan dan Manfaat

1.4.1 Tujuan

Berdasarkan rumusan masalah, tujuan penelitian ini sebagai berikut.

1. Membuat sistem yang dapat mendeteksi lubang pada di malam hari dalam kondisi pencahayaan rendah (*low light*) dengan algoritma YOLOv11 dan menggunakan teknik preprocessing *Adaptive Gamma Correction* dan *Contrast Limited Adaptive Histogram Equalization*.
2. Mengevaluasi performa YOLOv11 dalam mendeteksi lubang pada malam hari dalam kondisi pencahayaan rendah (*low light*) setelah penggunaan *preprocessing Adaptive Gamma Correction* dan *Contrast Limited Adaptive Histogram Equalization*.

1.4.2 Manfaat

Adapun manfaat penelitian ini sebagai berikut.

1. Menghasilkan sistem yang dapat mendeteksi lubang pada pada malam hari terutama dalam kondisi pencahayaan rendah (*low light*).
2. Menjadi referensi bagi penelitian selanjutnya yang berfokus pada sistem deteksi jalan berlubang dalam berbagai kondisi pencahayaan di malam hari.

1.5 Batasan Masalah

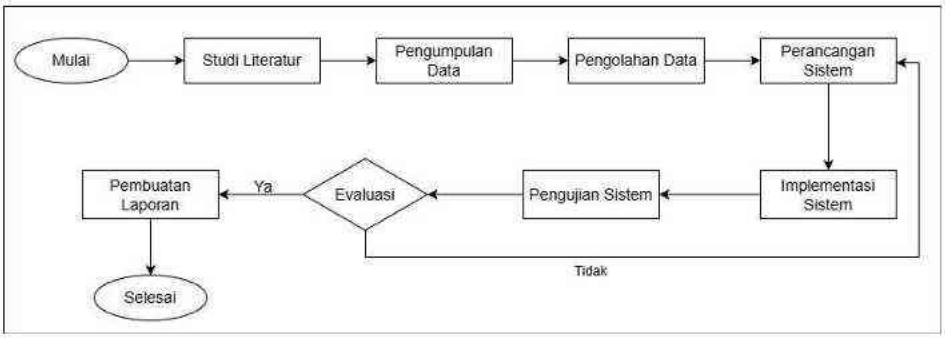
Batasan masalah pada penelitian ini yaitu.

1. Penelitian ini berfokus pada deteksi lubang pada malam hari dalam kondisi tidak hujan.
2. Pengambilan data dilakukan pada kecepatan 30km/jam, 40km/jam, 50km/jam, dan 60km/jam.

BAB II
METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini merupakan penelitian yang bersifat eksperimental dan analisis sehingga dapat dilakukan dengan metode studi pustaka, melalui beberapa literatur seperti buku, jurnal, prosiding atau artikel ilmiah lainnya, metode pengumpulan data lapangan (*field research*), perancangan sistem, dan analisis menggunakan metode pengolahan citra *(image processing*). Berikut disajikan tahapan penelitian seperti pada gambar 4.



Gambar 4. Tahapan Penelitian

2.1.1 Studi Literatur

Tahap studi literatur merupakan tahapan awal dalam penelitian ini. Pada tahap ini dilakukan pencarian referensi teori yang relevan terkait topik penelitian yatu deteksi lubang di jalan pada malam hari. Selain itu, studi literatur ini juga bertujuan untuk memahami perkembangan metode yang telah ada serta menentukan celah penelitian yang menjadi dasar dalam perancangan pemilihan metode yang digunakan dalam penelitian ini.

2.1.2 Pengumpulan Data

Tahap ini penulis melakukan pengumpulan dataset yang akan digunakan dalam penelitian. Dataset yang digunakan berupa foto jalan berlubang yang diambil pada malam hari dalam kondisi tidak hujan. Dataset yang dikumpulkan digunakan untuk melatih model yang akan dibangun.

2.1.3 Pengolahan Data

Tahap pengolahan data merupakan tahapan lanjutan setelah proses pengumpulan dataset. Tahapan ini mencakup beberapa proses seperti, anotasi data, *resize*, pembagian dataset, dan memperbaiki kualitas gambar yang memiliki pencahayaan rendah. Pengolahan data bertujuan untuk meningkakan kualitas dataset yang akan digunakan pada proses selanjutnya.

2.1.4 Perancangan Sistem

Pada tahapan ini dilakukan perancangan terhadap sistem yang nantinya akan dikembangkan untuk dapat menyelesaikan masalah dalam mendeteksi lubang pada malam hari dalam kondisi pencahayaan rendah. Tahapan ini, sistem dirancang secara rinci dan mudah untuk dipahami agar mempermudah dalam proses implementasi ke dalam program.

2.1.5 Implementasi Sistem

Implementasi sistem dilakukan setelah model YOLOv11 dilatih dengan menggunakan dataset yang telah disiapkan untuk mendeteksi lubang pada permukaan jalan di malam hari. Model hasil pelatihan digunakan untuk melakukan proses inferensi pada data uji. Dalam hal ini, citra atau video dimasukkan ke dalam sistem untuk diproses secara otomatis. Pada tahap ini, sistem secara otomatis mengekstraksi fitur visual dari permukaan jalan dan menghasilkan prediksi berupa lokasi lubang jalan dalam bentuk bounding box, kelas objek, serta nilai kepercayaan untuk setiap lubang yang terdeteksi.

2.1.6. Pengujian Sistem

Pada tahap pengujian sistem, pengujian dilakukan dengan menggunakan dataset uji yang terpisah dari dataset pelatihan. Setiap citra pada dataset uji terlebih dahulu diproses melalui tahap peningkatan kualitas citra untuk mengatasi kondisi pencahayaan rendah pada malam hari, sebelum dianalisis oleh sistem deteksi. Selanjutnya, sistem melakukan pendeteksian lubang pada permukaan jalan sesuai dengan skenario pengujian yang telah dirancang.

2.1.7 Evaluasi

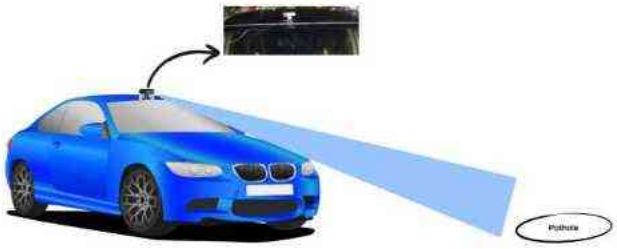
Evaluasi sistem merupakan tahapan untuk mengukur kinerja model deteksi lubang jalan yang telah dikembangkan secara menyeluruh. Tujuan evaluasi ini untuk menilai sejauh mana model berhasil mendeteksi keberadaan lubang pada permukaan jalan. Evaluasi dilakukan dengan membandingkan hasil deteksi sistem terhadap anotasi *ground truth* pada dataset uji, sehingga dapat diketahui tingkat ketepatan dan keandalan model dalam mengenali lubang jalan, khususnya pada kondisi malam hari dengan pencahayaan rendah.

2.1.8 Pembuatan Laporan

Pada bagian ini akan membahas hasil eksperimen dan analisis yang telah dilakukan. Pembahasan mencakup interpretasi terhadap temuan yang diperoleh serta perbandingan hasil penelitian ini dengan hasil penelitian terdahulu. Analisis difokuskan pada kemampuan sistem dalam mendeteksi lubang pada permukaan jalan, khususnya pada kondisi malam hari dengan pencahayaan rendah, sehingga dapat diketahui keunggulan dan keterbatasan metode yang diusulkan.

2.2 Sumber Data

Data yang digunakan dalam penelitian ini diambil pada sejumlah ruas jalan di Kota Makassar, Kabupaten Gowa, dan Kabupaten Pangkep Provinsi Sulawesi Selatan. Pengambilan data menggunakan kamera *dashcam* yang dipasang pada bagian luar mobil, tepatnya di posisi atas mobil dengan arah menghadap ke depan. Penempatan *dashcam* pada posisi ini untuk mendapatkan sudut pandang yang lebih luas tanpa terhalang oleh kaca mobil.



Gambar 5. Ilustrasi Pengambilan Data

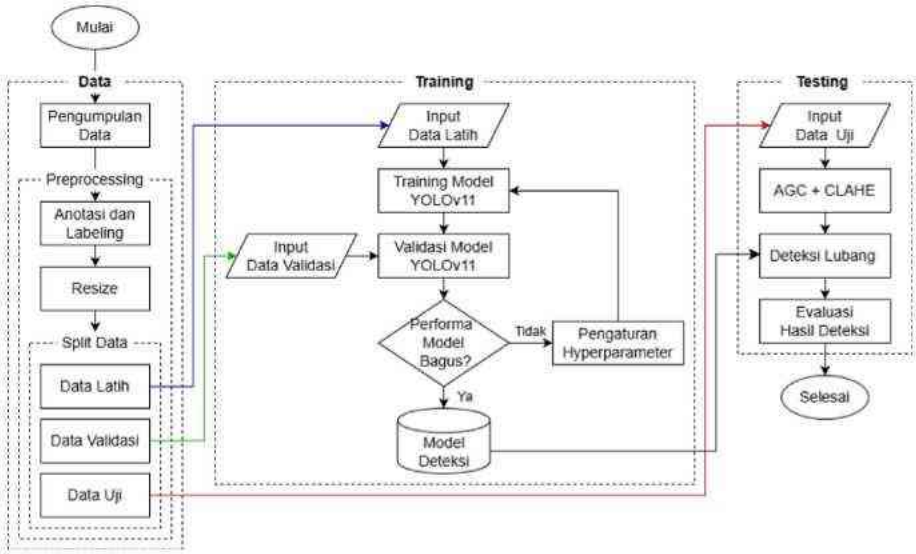
2.3. Perangkat Penelitian

Adapun perangkat-perangkat yang digunakan pada penelitian ini adalah sebagai berikut:

1. Perangkat Keras (*Hardware*):
a. Laptop
b. *Dashcam* DDPAI mola N3 dengan maksimal resolusi 1600p
2. Perangkat Lunak (*Software*):
a. Sistem Operasi Windows 11-64 bit
b. Bahasa Pemrograman *Python*
c. Jupyter Notebook
d. Google Colaboratory
e. Roboflow
f. Kaggle
g. Microsoft Office

2.4. Rancangan Sistem

Pada tahapan ini menjelaskan rancangan sistem yang digunakan dalam penelitian, mulai dari alur kerja sistem hingga komponen utama yang terlibat dalam setiap tahapan proses.



Gambar 6. Rancangan Sistem

2.4.1 Pengumpulan Data

Pengumpulan data pada penelitian ini dilakukan dengan cara merekam kondisi permukaan jalan berlubang secara langsung dengan menggunakan kamera *dashcam* dengan resolusi 1440P dan kecepatan 30 *frames per second* (fps). Proses perekaman dilakukan pada malam hari untuk merepresentasikan kondisi pencahayaan rendah (*low light*), sesuai dengan fokus penelitian. Perekaman dilakukan pada beberapa ruas jalan yang memiliki lubang dengan variasi kondisi lingkungan, seperti kondisi jalan yang memiliki lampu penerangan atapun jalan tanpa lampu penerangan. Variasi kondisi ini bertujuan untuk menghasilkan data yang merepresentasikan kondisi nyata serta keberagaman dataset yang digunakan dalam penelitian. Setelah proses perekaman selesai, video yang dihasilkan diekstrak menjadi gambar (*frame)*. Proses ekstraksi dilakukan dengan 5 *frame* untuk setiap detik.

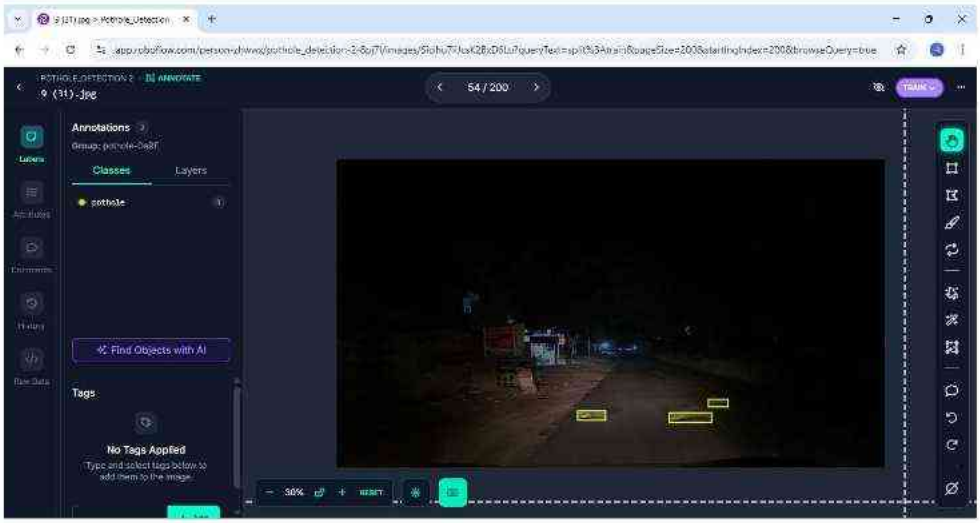
Gambar 7. *Dashcam* DDPAI Mola N3

2.4.2 Preprocessing

Setelah tahap pengumpulan data dilakukan, penelitian dilanjutkan ke tahap pra-pemrosesan yang bertujuan untuk mempersiapkan gambar sebelum pengolahan lebih lanjut.

a. Anotasi dan Labeling

Pada tahap ini dilakukan proses anotasi gambar dengan memberikan *bounding box* pada setiap objek jalan berlubang yang diperoleh pada kondisi malam hari. Selanjutnya, proses *labeling* dilakukan dengan menetapkan label atau kelas pada setiap objek yang telah dianotasi. Tahapan ini menggunakan aplikasi *website* Roboflow. Setelah seluruh proses anotasi selesai, data hasil anotasi disimpan dan dikonversi ke dalam format YOLO.



Gambar 8. Proses Anotasi dan Labeling pada Roboflow

b. *Resize*

Proses perubahan ukuran gambar (*resize)* dilakukan dengan mengubah gambar asli beresolusi 2560 x 1440 piksel menjadi 960 x 960 piksel. Tahapan ini bertujuan untuk menyesuaikan ukuran gambar dengan kebutuhan serta spesifikasi input model deteksi objek yang digunakan. Selain itu, tahapan *resize* berperan dalam mengurangi beban pemrosesan. Dan hal ini dapat membantu proses pelatihan dan pengujian model berjalan lebih efisien tanpa menghilangkan informasi visual penting yang merepresentasikan keberadaan lubang pada jalan.

c. Split Data

Tahapan selanjutnya yaitu split data dengan membagi dataset ke dalam tiga bagian dengan pembagian 70% data training, 20% data validasi, dan 10% data uji. Adapun jumlah dataset yang digunakan pada penelitian ini yaitu 766 data.

1) Data Training

Data training digunakan untuk melatih model agar mampu mengenali serta mempelajari karakteristik objek lubang pada permukaan jalan. Pada penelitian ini berfokus pada satu kelas, yaitu lubang. Jumlah data training yang digunakan sebanyak 512 citra jalan berlubang yang diambil pada malam hari.

2) Data Validation

Data validation digunakan untuk mengevaluasi performa model selama proses pelatihan berlangsung. Jumlah data validasi yang digunakan yaitu 174 citra.

3) Data Testing

Data testing digunakan untuk menilai kinerja akhir model setelah proses pelatihan dan validasi selesai dilakukan. Data testing yang digunakan merupakan data yang belum pernah dilihat oleh model sebelumnya. Pada penelitian ini, data testing diambil pada berbagai variasi kecepatan kendaraan, yaitu 30 km/jam, 40 km/jam, 50 km/jam, dan 60 km/jam. Menurut Kementerian Perhubungan (Kemenhub) mengeluarkan Peraturan Menteri Perhubungan Nomor 111 tahun 2015 mengenai tata cara penetapan batas kecepatan kendaraan bahwa kecepatan paling tinggi untuk Kawasan perkotaan 50 km/jam dan kecepatan paling rendah dalam kondisi arus bebas 60 km/jam (Permenhub, 2015). Adapun data testing untuk setiap kecepatan masing-masing terdiri dari 80 citra.

Gambar 9. Contoh Dataset

2.4.3 Training

a. Input Data Train

Tahapan pertama yaitu menginput data train yang berjumlah 512 citra jalan berlubang yang telah melalui *preprocessing*. Data train ini digunakan untuk melatih model deteksi dalam mengenali fitur-fitur atau karakteristik lubang pada permukaan jalan.

b. Training Model YOLOv11

Proses pelatihan model dalam penelitian ini menggunakan arsitektur YOLOv11. Secara umum arsitektur YOLO terdiri dari tiga komponen utama. Pertama, bagian *backbone* berfungsi untuk mengekstraksi fitur atau ciri -ciri penting dari citra inputan. *Neck* berfungsi untuk menggabungkan informasi fitur-fitur yang telah diekstrak dari *backbone*. Terakhir, *head* berfungsi melakukan prediksi untuk lokalisasi dan klasifikasi objek. Penjelasan lebih lanjut terkait fungsi masing-masing modul diuraikan pada penjelasan dibawah ini.

1) Modul *Convolution* (Conv)

Pada YOLOv11, modul *convolution* disusun secara *sequential* yang mencakup tiga tahapan utama seperti pada gambar 10.

Gambar 10. Alur *Convolution*

Tiga tahapan utama dalam *convolution*, yaitu:

2D *Convolution* (Conv2d). Tahapan Conv2d berfungsi untuk mengekstraksi fitur spasial citra inputan dengan mengalikan kernel pada citra inputan dan menghitung hasil perkaliannya. Secara matematis, operasi Conv2d dirumuskan pada persamaan 4.

$$x_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} I_{(i+m),(j+n)} \cdot W_{m,n} \tag{4}$$

Penjelasan:

$I_{(i+m),(j+n)}$: nilai piksel pada posisi $(i+m, j+n)$ dari citra masukan

$w_{m,n}$: bobot kernel pada posisi (m, n)

$x_{i,j}$: hasil konvolusi pada posisi (i, j)

***Batch Normalization* (BN).** Tahapan ini untuk menstabilkan nilai keluaran dari tahapan Conv2d. *Batch Normalization* membuat nilai keluaran menjadi lebih terkontrol dan tidak terlalu besar atau terlalu kecil, sehingga proses pembelajaran menjadi lebih stabil. Perhitungan pada BN dilakukan dengan cara menormalkan nilai keluaran berdasarkan rata-rata dan variansi. Berikut persamaan BN.

$$\widehat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (5)$$

$$y_i = \gamma \hat{x}_i + \beta \quad (6)$$

Penjelasan:

x_i : nilai keluaran hasil konvolusi

μ_B : nilai rata-rata mini-batch

σ_B^2 : variansi mini-batch

ϵ : konstanta kecil untuk mencegah pembagian nol

γ, β : parameter skala dan pergeseran yang dapat dilatih

***Sigmoid Linear Unit* (SiLU).** Fungsi ini bekerja dengan mengombinasikan linearitas dan non-linearitas melalui perkalian antara input dan fungsi sigmoid-nya. Persamaan SiLU seperti persamaan 7 di bawah ini.

$$\text{SiLU}(x) = x \cdot \sigma(x) \; dengan \; \sigma(x) = \frac{x}{1 + e^{-x}} \quad (7)$$

Penjelasan:

x : Nilai input hasil keluaran dari proses BN

$\sigma(x)$: Fungsi sigmoid yang memberikan pembobot non-linear terhadap nilai x

Pada modul *Convolution*, setiap proses *convolution* tidak hanya menghasilkan fitur spasial, tetapi juga mengubah ukuran atau memodifikasi dimensi data, seperti ukuran *height*, *width,* serta *depth* (jumlah *channel*). Secara matematis, ukuran keluaran dari proses konvolusi dua dimensi dapat dirumuskan pada persamaan 8.

$$H_{out} = \lfloor \frac{H_{in} - K + 2P}{S} \rfloor + 1, \quad W_{out} = \lfloor \frac{W_{in} - K + 2P}{S} \rfloor + 1 \quad (8)$$

Penjelasan:

H_{in}, W_{in} : tinggi dan lebar citra masukan

H_{out}, W_{out} : tinggi dan lebar keluaran

K : ukuran kernel (misalnya 3)

P : jumlah *padding*

S : ukuran *stride*

Selain mempengaruhi dimensi spasial, *Convolution* juga berpengaruh pada jumlah *channel* keluaran (C_{out}). Pada arsitektur YOLOv11, jumlah *channel* keluaran dari setiap lapisan telah ditentukan secara eksplisit dalam konfigurasi model. Jumlah kernel yang digunakan dalam proses *Convolution* akan menyesuaikan dengan jumlah *channel* keluaran tersebut. Hal ini dikarenakan setiap kernel menghasilkan satu *feature map*. Berdasarkan penjelasan tersebut, jumlah kernel dan jumlah *channel* keluaran dapat dirumuskan seperti persamaan 9.

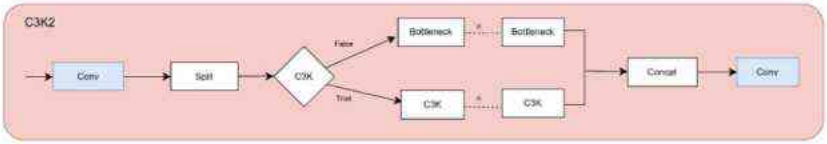
$$C_{out} = N_{kernel} \quad (9)$$

Sebagai ilustrasi, pada lapisan awal YOLOv11 dengan inputan berukuran 640 x 640 x 3, kernel berukuran 3 x 3, *stride* 2, dan jumlah kernel sebanyak 64 sehingga menghasilkan keluaran berukuran 320 x 320 x 64. Penurunan ukuran spasial terjadi akibat penerapan *stride* dua, sementara peningkatan jumlah *channel* disebabkan oleh jumlah kernel yang digunakan. Semakin banyak kernel yang diterapkan, semakin

banyak pula representasi fitur yang dapat dipelajari oleh model. Hal ini memungkinkan model untuk menangkap pola visual yang lebih kompleks pada tahapan berikutnya.

2) Modul C3K2

Pada modul C3K2 terdiri dari lima tahapan utama, yaitu *Convolution*, *Split*, *Bottleneck* (jika C3K = *False*), C3K (jika C3K=*True*), dan *Concat*. Tahapan pada modul C3K2 dapat dilihat pada gambar 11.



Gambar 11. Alur C3K2

Secara umum, proses pada modul ini diawali dengan *Convolution* awal untuk menyesuaikan jumlah kanal masukan. Selanjutnya membagi fitur melalui tahap *split*. Hasil split terbagi menjadi dua jalur yang akan diproses pada *Bottlencek* atau C3K. Hasil yang di dapatkan akan digabung pada proses *concat*. Penjelasan lebih rinci akan diuraikan di bawah ini.

Pada bagian ini, hanya akan berfokus pada pembahasan 4 tahapan utama lainnya selain *Convolution*. Hal ini dikarenakan, bagian *Convolution* telah dipaparkan sebelumnya.

Split. Tahapan ini untuk berfungsi untuk membagi fitur hasil keluaran dari *Convolution* menjadi dua bagian dengan menggunakan operasi *chunk* pada dimensi kanal. Operasi ini menghasilkan dua jalur, yaitu jalur utama dan jalur *shortcut*. Jalur utama akan diproses lebih lanjut melalui beberapa lapisan *Convolution*, sedangkan jalur *shortcut* akan dilewatkan langsung tanpa banyak transformasi. Proses pembagian fitur dapat dituliskan seperti persamaan 10.

$$[A, B] = \text{split}(C) \tag{10}$$

Penjelasan:

C : Jumlah kanal masukan
A, B : Bagian kanal untuk proses selanjutnya
Split(·) : Fungsi pemisahan kanal

Bottleneck Jalur utama diproses menggunakan blok *Bottleneck* standar jika kondisi C3K = False. Blok ini terdiri dari dua *Convolution*, yaitu *Convolution* 1 x 1 dan 3 x 3 dan proses *residual connection* yang berfungsi untuk menjumlahkan fitur awal dan hasil *Convolution*. Berikut persamaan *Bottleneck*.

$$Y = X + f_{3\times3}(f_{1\times1}(X)) \tag{10}$$

Penjelasan:

X : fitur masukan dari jalur transformasi
$f_{1\times1}$: konvolusi 1×1

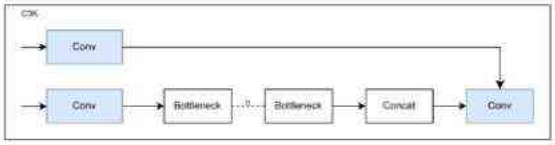
$f_{3\times3}$: konvolusi 3×3
Y : hasil keluaran dari blok Bottleneck
$+$: menunjukkan koneksi residu (*Residual connection*)

C3K. Jalur transformasi akan menggunakan blok C3K ketika *argument* C3K = *True*. Blok ini memanfaatkan 2 *Convolution* 1 x 1 untuk memisahkan dan menyeimbangkan fitur, serta blok *Bottleneck* di dalamnya. Persamaan pada blok C3K dapat dirumuskan pada persamaan 11.

$$Y = f_{1\times1}\left(\left[f_{\text{Bottleneck}}\left(f_{1\times1}(X)\right), f_{1\times1}(X)\right]\right) \tag{11}$$

Penjelasan:
X : fitur masukan
$f_{1\times1}$: konvolusi 1×1
$f_{\text{Bottleneck}}$: blok Bottleneck
$[,]$: operasi *concatenation* antar fitur
Y : hasil keluaran setelah penggabungan dua jalur fitur

Berikut merupakan alur C3K sebagaimana yang terlihat pada gambar 12.



Gambar 12. Alur C3K

***Concatenation* (Concat).** Pada tahap ini dilakukan penggabungan dua fitur berdasarkan dimensi *channel*. Salah satu fitur diperoleh dari hasil pemrosesan melalui blok C3K atau *Bottleneck*, sedangkan fitur lainnya berasal dari jalur *Convolution* yang dilewatkan secara langsung tanpa pemrosesan lanjutan. Hasil dari kedua fitur tersebut digabungkan secara berdampingan sehingga jumlah *channel* meningkat dua kali lipat. Secara matematis, proses tersebut dapat dinyatakan sebagai berikut.

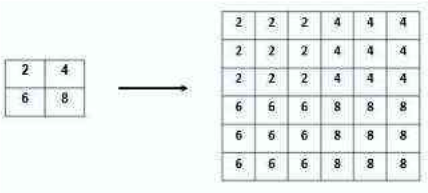
$$C_{out} = [C_{in\,1}, C_{in\,2}] \tag{12}$$

Sebagai contoh, apabila fitur C_{in1} dan C_{in2} masing-masing memiliki 32 *channel*, maka hasil penggabungan akan menghasilkan fitur keluaran berjumlah 64 *channel*.

3) Modul *Upsample*

Pada arsitektur YOLOv11, tahapan *upsample* berfungsi untuk mengubah ukuran spasial (*height* dan *width*) dari *feature map* menjadi lebih besar. Hal ini dikarenakan *feature map* yang dihasilkan oleh proses *convolution* berukuran kecil. Selain itu, tahapan ini bertujuan untuk menggabungkan fitur-fitur yang berasal dari berbagai tingkat resolusi dapat dimanfaatkan secara optimal dalam proses deteksi. Tahapan ini dilakukan dengan menggunakan fungsi *torch.nn.Upsample* dari pustaka *PyTorch* dengan parameter mode sama dengan '*nearest*'. Parameter ini menerapkan metode *nearest-neighbor interpolation* yaitu memperbesar ukuran fitur baru dengan cara mereplikasi nilau piksel terdekat.

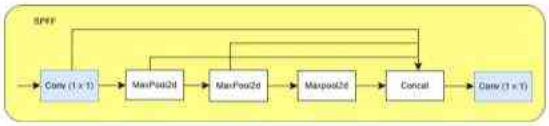
Pada tahapan ini, setiap nilai piksel pada *feature map* awal akan disalin secara horizontal dan vertikal berdasarkan faktor skala yang ditentukan. Sebagai ilustrasi, apabila *feature map* awal berukuran 2 x 2, kemudian faktor skala sama dengan 3, maka ukuran *feature map* yang baru akan menjadi 6 x 6 setelah dilakukan *upsample*. Nilai piksel pada *feature map* awal akan diperluas menjadi blok dengan ukuran 3 x 3 dengan nilai yang sama. Gambar 13 merupakan ilustrasi pada tahapan *upsample.*



Gambar 13. Ilustrasi Tahapan *Upsample*

4) Modul *Spatial Pyramid Pooling-Fast* (SPPF)

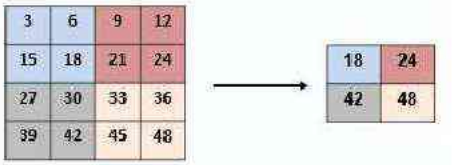
Tahapan SPFF berperan untuk memperluas cakupan konteks spasial dari fitur yang dihasilkan oleh lapisan sebelumnya tanpa mengubah resolusi spasilanya. Tahapan ini diawali dengan keluaran dari lapisan *convolution*, kemudian dilanjutkan dengan tiga proses *max pooling* yang dilakukan secara berurutan. Setiap tahap *pooling* ini akan mempertahankan ukuran spasial yang sama, namun tetap memperluas cakupan informasi yang diekstraksi. Setelah tiga tahap *maxpooling* dilakukan, semua keluaran yang diperoleh akan digabungkan pada dimensi kanal, termasuk fitur awal sebelum *max pooling* juga akan digabungkan. Dan hasil penggabungan akan dilanjutkan dengan proses terakhir pada SPFF ini yaitu *convolution* untuk menyesuaikan jumlah *channel* keluaran. Alur SPFF dapa terlihat pada gambar 14.



Gambar 14. Alur SPPF

Max Pooling. *Max pooling* dilakukan dengan cara membagi *feature map* inputan ke dalam beberapa bagian kecil atau disebut dengan kernel. Setelah itu, nilai maksimum dari setiap bagian tersebut akan digunakan sebagai nilai representasi pada fitur baru. Gambar 15 memberikan ilustrasi proses *max pooling.*

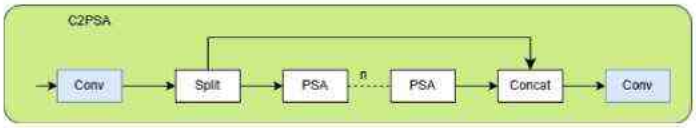
Pada gambar 15 terdapat fitur masukan dengan ukuran 4 x 4 yang diproses memggunakan kernel 2 x 2 dan *stride* 2. Setiap bagian kecil 2 x 2 pada fitur inputan akan menghasilkan satu nilai sebagai keluaran fitur baru yang merupakan nilai maksimum dari empat nilai lainnya yang berada pada bagian tersebut.



Gambar 15. Proses *Max Pooling*

5) Modul *Cross-Stage Partial Spatial Attention* (C2PSA)

Modul C2PSA merupakan kombinasi antara mekanisme *Cross-Stage Partial* (CSP) dengan *Position-Sensitive Spatial Attention* (PSA). C2PSA ini dirancang untuk memperkaya informasi spasial serta memperkuat keterkaitan antar-piksel, sehingga jaringan mampu menekankan area-area penting pada fitur secara adaptif. Pada tahap awal, fitur inputan terbagi menjadi dua jalur, yaitu jalur yang berfungsi sebagai *shortcut* untuk menjaga stabilitas aliran gradien, sementara jalur lainnya diproses melalui mekanisme PSA yang mencakup komponen *Attention* dan *Feed Forward Network* (FFN). Tampilan alur C2PSA dapat dilihat pada gambar 16.



Gambar 16. Alur C2PSA

PSA *Block*. PSABlock memproses *feature map* inputan dengan menggunakan dua tahapan utama, yaitu melalui mekanisme *Attention* dan *Feed Forward Network* (FFN). Kedua mekanisme ini masing – masing dihubungkan dengan menggunakan *residual connection*. Pada tahap awal, fitur inputan *X* diproses oleh modul *Attention* untuk menghasilkan transformasi fitur dan hasil keluaran digabungkan kembali dengan fitur awal melalui *residual connection* seperti persamaan 13.

$$F_1 = X + \text{Attention}(X) \tag{13}$$

Penjelasan:

X : Fitur masukan

F_1 : Fitur hasil integrasi antara masukan awal dan *output attention*

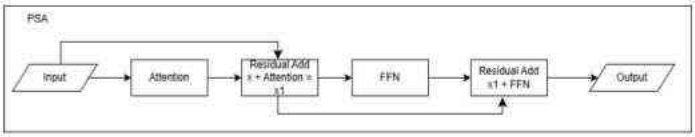
Tahap berikutnya memproses F_1 melalui FFN, kemudian hasilnya kembali dijumlahkan dengan fitur sebelumnya melalui *residual connection* kedua dapat dituliskan seperti persamaan 14 dibawah ini.

$$F_{out} = F_1 + \text{FFN}(F_1) \tag{14}$$

Penjelasan:

F_{out} : keluaran akhir blok PSA

Gambar 17 merupakan alur dari pemrosesan pada PSA *Block*.



Gambar 17. Alur PSA *Block*

Attention. Proses *Attention* diawali dengan menerapkan lapisan *convolution* 1 x 1 pada fitur inputan untuk menghasilkan representasi awal yang digunakan dalam pembentukan *Query, Key,* dan *Value*. Hasil dari operasi *convolution* kemudian diubah bentuknya (*reshape*) agar sesuai dengan dimensi tensor yang dibutuhkan, yang sebelumnya dibagi menjadi 3 bagian yaitu Q, K, dan V, sesuai dengan proporsi kanal yang telah ditetapkan. Setelah Q dan K diperoleh, kedua komponen tersebut diproses melalui operasi perkalian antara Q dan *transpose* dari K yang akan menghasilkan skor keterkaitan antar elemen fitur. Perhitungan skor *Attention* dapat dirumuskan seperti persamaan 15.

$$S = (QK^T) \cdot \text{scale} \tag{15}$$

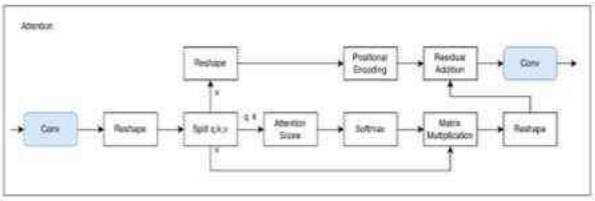
Nilai *S* kemudian dinormalisasi menggunakan fungsi *softmax* untuk menghasilkan matriks bobot *Attention A*, seperti pada persamaan 16.

$$A = \text{Softmax}(S) \tag{16}$$

Matriks bobot A selanjutnya digunakan untuk mengalikan komponen *value* V dengan operasi perkalian matriks. Hasil dari perhitungan ini akan menghasilkan keluaran fitur berbobot F_{att}, sebagaimana dinyatakan pada persamaan 17.

$$F_{att} = AV \tag{17}$$

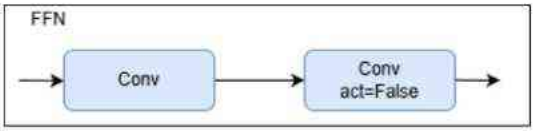
Kemudian keluaran F_{att} ukurannya diubah kembali (*reshape*) agar sesuai dengan dimensi pada fitur dua dimensi. Secara paralel, komponen V juga mengalami proses *reshape* dan akan dilanjutkan dengan proses *convolution depthwise* berukuran 3 x 3 untuk menghasilkan *positional encoding*. Hasil dari *positional encoding* tersebut ditambahkan kembali ke fitur hasil *Attention* melalui *residual addition*. Pada tahap akhir, diterapkan *convolution* 1 x 1 yang berperan sebagai proyeksi linear untuk menghasilkan keluaran akhir dari blok *Attention*.



Gambar 18. Alur Attention

FFN (Feed Forward Network). Hasil dari keluaran blok *Attention* akan diproses pada FNN dengan 2 tahapan *convolution* secara berurutan. Proses diawali dengan menerapkan *convolution* 1 x 1 pertama yang berfungsi untuk mengubah dimensi

channel fitur sehingga menghasilkan reprsentasi antara sebelum memasuki tahap selanjutnya. Keluaran dari tahapan ini diteruskan ke lapisan *convolution* 1 x 1 kedua yang berfungsi untuk mengembalikan jumlah *channel* ke dimensi awal. Pada *convolution* kedua ini fungsi aktivasi tidak diterapkan (*act* sama dengan *False*), sehingga keluaran FFN berupa transformasi linear murni. Alur FFN dapat dilihat pada gambar 19.



Gambar 19. Alur FFN

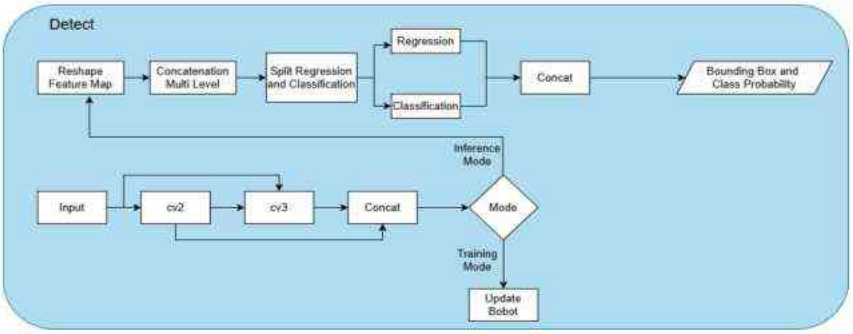
6) Modul *Detect*

Modul *detect* pada bagian *head* YOLOv11 berperan sebagai komponen akhir yang menghasilkan prediksi akhir dari model. Modul ini menerima *feature map* hasil pemrosesan dari tahap *backbone* dan *neck*, yang selanjutnya memproses setiap level *feature map*. melalui dua jalur utama, yaitu *bounding box regression head* dan *classification head*. Kedua jalur tersebut menghasilkan prediksi *bounding box*, probabilitas kelas objek, dan nilai *confidence score*. Modul *detect* memanfaatkan tiga level *feature map*, P3 digunakan untuk mendeteksi objek kecil, P4 untuk objek berukuran sedang, dan P5 untuk objek berukuran besar. Masing-masing dirancang menangani objek dengan skala yang berbeda.

Secara umum *detect* memiliki dua alur kerja yaitu *training mode* dan *inference mode*. Kedua mode selalu diawali oleh proses yang sama untuk menghasilkan prediksi mentah tersebut. Pada *training mode*, keluaran mentah ini langsung diteruskan ke modul *loss* untuk menghitung nilai kerugian (*loss function*), yang selanjutnya digunakan dalam proses *backpropagation* untuk memperbarui bobot model pada setiap iterasi pelatihan. Sebaliknya, pada *inference mode*, modul *detect* melakukan proses tambahan untuk mengubah prediksi mentah menjadi koordinat *bounding box* akhir serta skor probabilitas kelas.

Pada tahap awal, modul *detect* menerima inputan yang berasal dari tiga *feature map* dari *neck*. Kemudian, setiap *feature map* diproses secara terpisah dengan dua jalur. Jalur pertama, cv2 terdiri dari dua lapisan *convolution* dengan fungsi normalisasi dan fungsi aktivasi yang diikuti *convolution* 1 x 1. Proses ini menghasilkan keluaran regresi *bounding box* untuk *feature map* tersebut. Sedangkan, jalur kedua, cv3 memanfaatkan dua lapisan *depthwise convolution* dengan fungsi normalisasi dan fungsi aktivasi serta *convolution* 1 x 1. Jalur ini berfungsi untuk menghasilkan prediksi klasifikasi. Kedua jalur ini menerima input yang sama, akan tetapi menjalankan pemrosesan yang berbeda sesuai fungsinya.

Hasil dari cv2 dan cv3 akan digabungkan pada dimensi kanal. Hasil penggabungan ini menghasilkan satu keluaran yang berisi informasi awal tentang posisi objek dan kelas objek yang terdeteksi. Hasil keluarannya masih berupa prediksi mentah, namun menjadi dasar untuk tahapan selanjutnya. Berikut alur *detect* dapat dilihat pada gambar 20.



Gambar 20. Alur Modul *Detect*

Training Mode. Pada tahap training, setial level *feature map* diproses secara terpisah oleh jalur cv2 dan cv3, kemudian hasil dari kedua jalur tersebut digabungkan tanpa mengubah struktur atau menggabungkan antarlevel *feature map*. Keluaran hasil penggabungan ini berupa prediksi pada masing-masing level yang selanjutnya diteruskan ke modul loss. *Loss* bertugas membandingkan hasil prediksi dengan data *ground truth*. Serta menghitung nilai *loss* berdasarkan komponen regresi dari cv2 dan komponen klasifikasi dari cv3. Nilai *loss* yang diperoleh kemudian digunakan dalam proses *backpropagation* untuk memperbarui bobot jaringan pada setiap iterasi pelatihan.

Inference Mode. Pada inference mode, hasil keluaran dari proses *concatenation* pada setiap *level feature map* diproses lebih lanjut untuk menghasilkan prediksi objek akhir. Tahapan ini dimulai dengan melakukan *reshape* pada *feature map* sehingga setiap posisi spasial dianggap sebagai satu kandidat deteksi. Seluruh kandidat deteksi dari berbagai skala kemudian digabungkan melalui proses *concatenation* lintas level untuk membentuk representasi prediksi yang terintegrasi. Representasi ini selanjutnya dipisahkan ke dalam dua jalur, yaitu jalur regresi dan jalur klasifikasi. Pada jalur regresi, dilakukan proses *decoding* untuk memperoleh koordinat *bounding box* yang merepresentasikan posisi objek pada citra masukan. Sementara itu, jalur klasifikasi memproses fitur yang sama untuk menghasilkan probabilitas kelas objek. Keluaran dari kedua jalur tersebut kemudian digabungkan kembali melalui operasi *concatenation*. Dengan demikian, keluaran akhir modul *detect* pada mode *inference* berupa koordinat *bounding box* beserta probabilitas kelas yang merepresentasikan hasil prediksi objek.

c. Validasi Model

Validasi model digunakan untuk mengevaluasi performa algoritma YOLOv11 terhadap dataset yang tidak digunakan selama proses pelatihan. Pada tahapan ini, model akan menghasilkan prediksi berupa *bounding box,* kelas objek, dan nilai *confidence* dari setiap objek yang terdeteksi. Hasil prediksi tersebut dibandingkan dengan *ground truth* dan dilakukan perhitungan metrik evaluasi seperti *precision*, *recall*, mAP50, dan mAP50-95. Nilai tersebut berfungsi untuk menilai kestabilan model selama proses pelatihan serta menganalisis kemungkinan terjadinya *overfitting* ataupun *underfitting*.

d. Pengaturan Hyperparameter

Proses pengaturan ini dilakukan dengan menyesuaikan nilai-nilai *hyperparameter* yang dapat mempengaruhi proses pelatihan seperti, jumlah *epoch*, *batch size*, *image size*, serta *hyperparameter* lainnya. Setelah penentuan nilai *hyperparameter* dilakukan, model akan dilatih kembali dan dievaluasi menggunakan data validasi untuk menilai performa model. Proses ini akan dilakukan secara berulang hingga diperoleh model yang memiliki performa yang stabil. Tabel 1 berikut menyajikan beberapa *hyperparameter* beserta nilainya yang digunakan selama proses pelatihan YOLOv11.

Tabel 1. Hyperparameter yang Digunakan dalam Training YOLOv11

Hyperparameter	Value
Image Size	960 x 960
Epochs	170
Batch Size	32
Optimizer	AdamW
Confidence Threshold	0.25
IoU Threshold	0.45

e. Model Deteksi

Setelah tahapan pelatihan model selesai dan diperoleh performa model yang cukup bagus, maka model YOLOv11 disimpan sebagai hasil akhir pelatihan. Model ini menyimpan bobot dan parameter yang diperoleh selama proses pembelajaran dan akan digunakan untuk proses deteksi objek pada tahap pengujian.

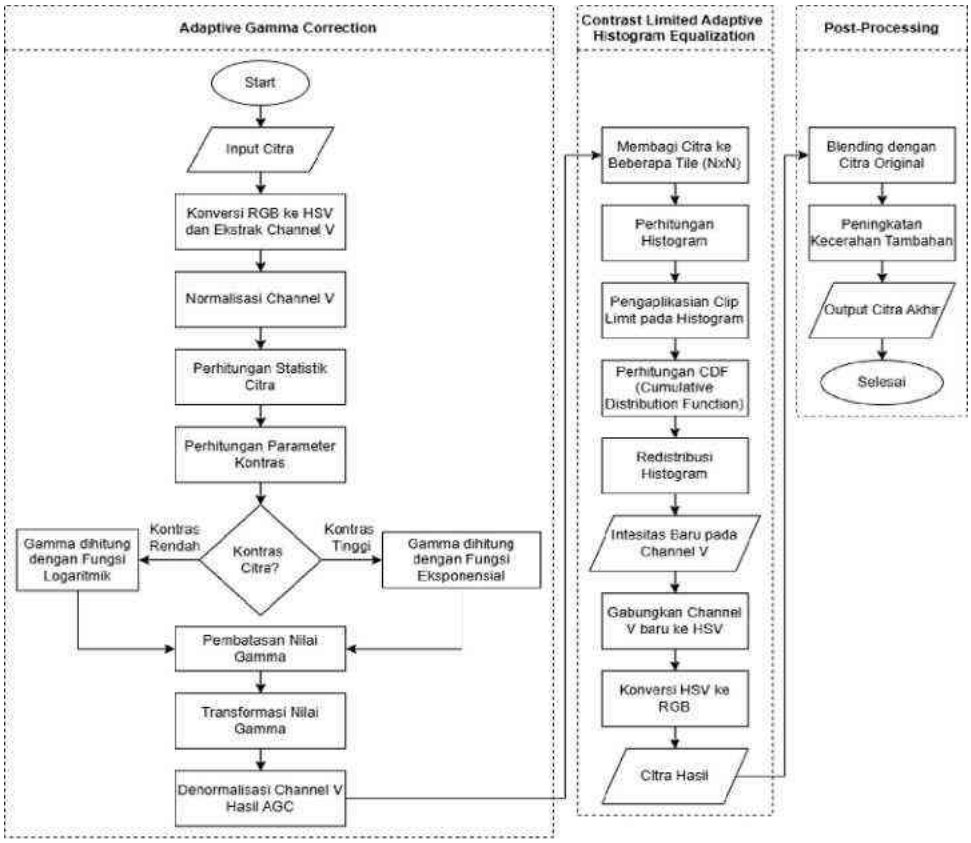
2.4.4 Testing

a. Input Data Testing

Data testing yang telah dipersiapkan sebelumnya dimasukkan ke dalam sistem untuk diproses oleh model YOLOv11. Data testing yang digunakan telah dibagi berdasarkan skenario kecepatan yang ditentukan.

b. Peningkatan Kualitas Citra dengan AGC dan CLAHE

Peningkatan kualitas citra dilakukan untuk mengatasi tantangan pencahayaan rendah pada citra jalan berlubang di malam hari yang menyebabkan lubang jalan sulit dikenali secara visual. Penerapan AGC dan CLAHE hanya dilakukan pada tahap testing. Alur kerja AGC CLAHE seperti pada gambar 21.



Gambar 21. Alur Kerja AGC dan CLAHE

Berikut penjelasan setiap tahapan pada metode AGC dan CLAHE.

1) Bagian AGC

Input Citra. Pada tahap ini, citra inputan yang digunakan merupakan citra berwarna dalam ruang warna *Red, Green, Blue* (RGB).

15	14	13	...	4	4	4
13	12	11	...	4	4	4
11	10	9	...	4	4	4
...	...	...	...	...	...	...
7	7	7	...	4	4	4
6	6	6	...	4	4	4
6	6	6	...	4	4	4

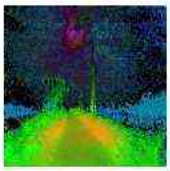
16	15	14	...	0	0	0
14	13	12	...	0	0	0
12	11	10	...	0	0	0
...	...	...	...	...	...	...
23	23	23	...	3	3	3
22	22	22	...	3	3	3
22	22	22	...	3	3	3

14	13	12	...	12	14	14
12	11	10	...	12	14	14
10	9	8	...	12	14	14
...	...	...	...	...	...	...
35	35	35	...	5	5	5
34	34	34	...	5	5	5
34	34	34	...	5	5	5

(a) **(b)** **(c)** **(d)**

Gambar 22. (a) Citra Input, (b) Nilai Piksel *Channel Red*, (c) Nilai Piksel *Channel Green*, (d) Nilai Piksel *Channel Blue*

Konversi RGB ke HSV. Tahapan selanjutnya dengan mengonversi citra RGB ke ruang warna *Hue, Saturation, Value* (HSV). Konversi ini dilakukan untuk memisahkan informasi warna *Hue* dan *Saturation* dari informasi kecerahan *Value.*



75	75	75	...	170	171	171
75	75	75	...	170	171	171
75	75	75	...	170	171	171
...	...	...	...	...	...	...
17	17	17	...	165	165	165
17	17	17	...	165	165	165
17	17	17	...	165	165	165

32	34	36	...	255	255	255
36	39	43	...	255	255	255
43	46	51	...	255	255	255
...	...	...	...	...	...	...
204	204	204	...	102	102	102
210	210	210	...	102	102	102
210	210	210	...	102	102	102

16	15	14	...	12	14	14
14	13	12	...	12	14	14
12	11	10	...	12	14	14
...	...	...	...	...	...	...
35	35	35	...	5	5	5
34	34	34	...	5	5	5
34	34	34	...	5	5	5

(a) (b) (c) (d)

Gambar 23. (a) Citra HSV, (b) Nilai Piksel *Channel Hue*, (c) Nilai Piksel *Channel Saturation*, (d) Nilai Piksel *Channel Value*

Ekstraksi *Channel Value* (V). Pada penelitian ini, hanya *channel* V yang diproses lebih lanjut, karena *channel* tersebut merepresentasikan tingkat kecerahan citra. Dengan memfokuskan proses peningkatan kecerahan hanya pada *channel* V tanpa mengubah warna asli citra.

16	15	14	...	12	14	14
14	13	12	...	12	14	14
12	11	10	...	12	14	14
...	...	...	...	...	...	...
35	35	35	...	5	5	5
34	34	34	...	5	5	5
34	34	34	...	5	5	5

(a) (b)

Gambar 24. (a) Citra *Channel Value*, (b) Nilai Piksel *Channel Value*

Normalisasi Intensitas Piksel *Channel* V. *Channel* V yang semula berada pada rentang 0-255 dinormalisasi menjadi 0-1 dengan membagi setiap piksel dengan 255. AGC menggunakan fungsi gamma dan fungsi ini bekerja paling stabil ketika nilai piksel yang diproses berada pada rentang 0-1. Berikut intensitas piksel *channel* V yang telah dinormalisasi dapat dilihat pada gambar 25.

0.06	0.05	0.05	...	0.04	0.05	0.05
0.05	0.05	0.04	...	0.04	0.05	0.05
0.04	0.04	0.03	...	0.04	0.05	0.05
...	...	...	...	...	...	...
0.13	0.13	0.13	...	0.01	0.01	0.01
0.13	0.13	0.13	...	0.01	0.01	0.01
0.13	0.13	0.13	...	0.01	0.01	0.01

Gambar 25. Piksel Channel V Hasil Normalisasi

Perhitungan Statistik Citra. *Channel* V yang telah dinormalisasi, akan dilanjutkan dengan menghitung mean (μ) dan standar deviasi (σ). Nilai mean merepresentasikan tingkat kecerahan rata-rata citra, sedangkan standar deviasi merepresentasikan

tingkat penyebaran intensitas piksel yang berkaitan langsung dengan kontras citra. Perhitungan mean seperti pada persamaan 18.

$$\mu = \frac{1}{N}\sum_{k=1}^{N} I_k \tag{18}$$

μ : mean (rata-rata) intensitas citra
I_k : nilai intensitas piksel ke-*k*
N : jumlah total piksel pada citra

Perhitungan mean dimulai dengan menjumlahkan nilai intesitas piksel. Kemudian menghitung jumlah piksel. Hasil penjumlahan dari nilai intensitas piksel dibagi dengan hasil penjumlahan piksel untuk mendapatkan nilai mean. Sehingga diperoleh, hasil mean intensitas, 0.1577. Adapun, perhitungan standar deviasi seperti pada persamaan 19.

$$\sigma = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(I_k - \mu)^2} \tag{19}$$

σ : standar deviasi intensitas citra
I_k : nilai intensitas piksel ke-k
N : jumlah total piksel

Perhitungan standar deviasi dilakukan dengan menghitung selisih setiap nilai piksel dengan nilai mean Selisih antara nilai piksel dan mean kemudian dikuadratkan. Seluruh nilai hasil pengkuadratan selish intensitas dijumlahkan untuk memperoleh total penyimpangan intensitas piksel terhadap mean dalam citra. Total nilai kuadrat selisih kemudian dibagi dengan jumlah piksel *N*. hasil pembagian diambil akar kuadratnya untuk memperoleh nilai standar deviasi (σ). Diperoleh hasil standar deviasi 0.1841.

Penentuan Parameter Kontras (D). Perhitungan parameter kontras ini digunakan untuk mengklasifikasikan citra menjadi dua kategori, yaitu citra kontras rendah dan citra kontras tinggi. Perhitungan tersebut dapat dilihat seperti persamaan 20.

$$\mathrm{D} = 4 \times \sigma \tag{20}$$

Jika nilai D lebih kecil atau sama dengan 0.3, citra dianggap memiliki kontras rendah. Jika nilai D lebih dari 0.3, citra dianggap memiliki kontras tinggi. Dari perhitungan diatas, diketahui nilai D sama dengan 0.7367.

Penentuan Nilai Gamma. Jika citra dikategorikan sebagai citra kontras rendah, nilai Gamma dihitung dengan fungsi logaritmik untuk meningkatkan perbedaan intensitas piksel pada citra gelap. Fungsi logaritmik bergantung pada nilai standar deviasi. Persamaan fungsi logaritmik,dapat dirumuskan pada persamaan 21 dibawah ini.

$$\gamma = -log2(\sigma + \varepsilon) \tag{21}$$

Penjelasan:
γ : nilai parameter gamma
$log2$: logaritma basis 2
σ : standar deviasi intensitas piksel
ε : konstanta kecil untuk menghindari nilai algoritma nol

Sebaliknya, jika citra dikategorikan sebagai kontras tinggi, maka nilai Gamma dihitung dengan fungsi eksponensial yang mempertimbangkan nilai mean dan standar deviasi, sebagaimana persamaan 22.

$$\gamma = exp\left(\frac{1-(\mu+\sigma)}{1.5}\right) \quad (22)$$

Penjelasan:

μ : nilai rata-rata (mean) intensitas piksel

σ : standar deviasi intensitas piksel

exp : fungsi eksponensial natural

Nilai D yang diperoleh dari perhitungan pada persamaan 20 sama dengan 0.7367, maka D lebih besar dari 0.3. Jadi citra dianggap kontras tinggi dan dihitung dengan fungsi eksponensial, sehingga diperoleh hasil berikut.

$$\gamma = exp\left(\frac{1-(0.1577+0.1841)}{1.5}\right) = 1.5506$$

Pembatasan Nilai Gamma (*Clipping*). Nilai gamma yang diperoleh kemudian dibatasi pada rentang 0.9 ≤ γ ≤1.3. Pembatasan ini bertujuan untuk mencegah terjadinya peningkatan kecerahan yang berlebihan (*over-enhancement*) atau penurunan kecerahan yang terlalu drastis,

Jika γ < 0.9, maka γ = 0.9

Jika γ > 1.3, maka γ = 1.3

Jika 0.9 ≤ γ ≤1.3, maka nilai gamma tidak berubah

Gamma = 1.5506, jadi Gamma setelah *clipping* = 1.3

Nilai pembatasan gamma 0.9-1.3 tidak diambil dari referensi tertentu, tetapi diperoleh melalui beberapa eksperimen yang dilakukan pada penelitian ini. Pada penelitian ini dilakukan beberapa eksperimen rentang gamma, mulai dari tanpa pembatasan, rentang lebar seperti 0.5-1.5 hingga rentang yang lebih sempit. Selain itu, berdasarkan hasil evaluasi, rentang 0.9-1.3 menunjukkan performa yang lebih unggul dibandingkan rentang lainnya.

Transformasi Gamma. Transformasi gamma merupakan teknik *non-linear* yang digunakan untuk menyesuaikan tingkat kecerahan citra dengan memodifikasi hubungan antara intensitas piksel input dan output. Teknik *non-linear* berarti piksel gelap lebih banyak diubah, sedangkan piksel terang lebih sedikit diubah.

$$I_{out}(x,y) = I_{in}(x,y)^{\gamma clipping} \quad (23)$$

Penjelasan:

$I_{out}(x, y)$: Intensitas piksel keluaran

$I_{in}(x, y)$: Intensitas piksel masukan

$\gamma clipping$: Nilai gamma setelah *clipping*

Sehingga diperoleh piksel hasil transformasi seperti pada gambar 25.

0.027	0.025	0.022	...	0.018	0.022	0.022
0.022	0.020	0.018	...	0.018	0.022	0.022
0.018	0.016	0.014	...	0.018	0.022	0.022
...	...	...	...	...	...	...
0.075	0.075	0.075	...	0.006	0.006	0.006
0.072	0.072	0.072	...	0.006	0.006	0.006
0.072	0.072	0.072	...	0.006	0.006	0.006

Gambar 26. Piksel Hasil Transformasi

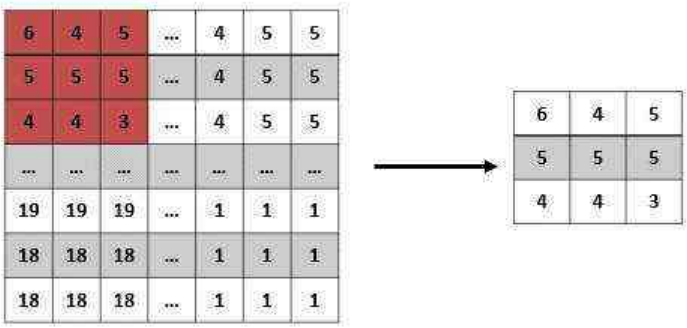
Konversi Kembali ke Rentang 0-255. Hasil transformasi *Channel* V pada langkah sebelumnya akan dikonversi ke rentang 0-255, sehingga diperoleh nilai piksel seperti gambar 26 dibawah ini.

6	4	5	...	4	5	5
5	5	5	...	4	5	5
4	4	3	...	4	5	5
...	...	...	...	...	...	...
19	19	19	...	1	1	1
18	18	18	...	1	1	1
18	18	18	...	1	1	1

Channel V yang telah dikonversi ke rentang 0-255 pada gambar 26 akan digunakan untuk proses selanjutnya pada CLAHE.

2) Proses CLAHE

Membagi Citra Menjadi Beberapa Tile. Proses CLAHE menerima masukan berupa *Channel Value* yang telah diproses pada AGC. *Channel Value* kemudian dibagi menjadi beberapa wilayah kecil (*tile*) Pembagian ini bertujuan untuk peningkatan kontras dilakukan secara lokal, sehingga setiap area citra dapat diproses sesuai karakteristik intensitas piksel di wilayah tersebut.



Gambar 28. Gambar Setelah Citra Dibagi Menjadi Beberapa *Tile*

Perhitungan Histogram untuk Intensitas Piksel. Perhitungan histogram merepresentasikan distribusi frekuensi intensitas piksel pada setiap *tile*. Histogram ini menggambarkan seberapa sering setiap nilai intensitas muncul di dalam *tile* dan menjadi dasar untuk proses peningkatan kontras lokal.

Intensitas	Frekuensi
3	1
4	3
5	4
6	1

Gambar 29. Gambar Perhitungan Histogram

Pengaplikasian *Clip Limit* Pada Histogram. Jika nilai frekuensi dari intensitas piksel lebih besar dari *clip limit* yang ditentukan, maka kelebihan nilainya di distribusikan ke nilai intensitas lainnya. Misalnya, *clip limit* sama dengan 3. Penerapan *clip limit* ini bertujuan untuk mencegah peningkatan kontras yang berlebihan serta mengurangi munculnya noise, terutama pada area citra yang homogen atau gelap.

Intensitas	Frekuensi
3	1
4	3
5	4
6	1

Intensitas	Frekuensi
3	1,5
4	3
5	3
6	1,5

(a) (b)

Gambar 30. (a) Nilai Frekuensi Sebelum Pengaplikasian *Clip Limit*, (b) Nilai Frekuensi Setelah Pengaplikasian *Clip Limit*

Menghitung *Cumulative Distribution Function* (CDF). CDF diperoleh dengan menjumlahkan frekuensi histogram secara kumulatif dan berfungsi sebagai dasar pemetaan intensitas. Adapun hasil CDF yang diperoleh seperti pada gambar 31.

Intensitas	Frekuensi	CDF
3	1,5	1,5
4	3	4,5
5	3	7,5
6	1,5	9

Gambar 31. Hasil CDF

Normalisasi *Cumulative Distribution Function* (CDF). Setelah nilai CDF diperoleh untuk setiap level intensitas, sistem melakukan normalisasi CDF untuk menyesuaikan hasil pemetaan intensitas dengan rentang nilai piksel 0-255. nilai CDF yang semula berupa frekuensi kumulatif diubah menjadi nilai intensitas baru yang merepresentasikan tingkat kecerahan piksel hasil peningkatan kontras. Normalisasi CDF dapat dirumuskan seperti persamaan 24.

$$\text{CDF\_normalized} = \left(\frac{\text{CDF} - \min(\text{CDF})}{\max(\text{CDF}) - \min(\text{CDF})}\right) \times 255 \qquad (24)$$

Intensitas	Frekuensi	CDF	Intesitas Baru (Setelah Normalisasi)
3	1,5	1,5	0
4	3	4,5	102
5	3	7,5	204
6	1,5	9	255

Gambar 32. Hasil Normalisasi CDF

Redistribusi Histogram. Nilai intensitas baru didistribusikan kembali secara merata ke seluruh bin histogram untuk menjaga keseimbangan distribusi frekuensi intensitas piksel. Melalui proses redistribusi histogram, CLAHE mampu meningkatkan kontras lokal secara efektif sekaligus menjaga kualitas visual citra agar tetap seimbang dan representatif terhadap kondisi asli citra.

Sebelum CLAHE

6	4	5
5	5	5
4	4	3

(a)

Setelah CLAHE

255	102	204
204	204	204
102	102	0

(b)

Gambar 33. (a) Nilai Intensitas Piksel Sebelum Penerapan CLAHE, (b) Nilai Intensitas Piksel Setelah Penerapan CLAHE

Langkah-langkah CLAHE diatas akan dilakukan secara berulang pada setiap tile hingga seluruh piksel pada citra di proses.

Menggabungkan Kembali Citra HSV. Setelah proses peningkatan kualitas citra pada *channel* V selesai dilakukan menggunakan AGC dan CLAHE, sistem menggabungkan kembali channel V hasil pemrosesan dengan *channel* H dan S yang tidak mengalami perubahan. Penggabungan ini dilakukan untuk mempertahankan informasi warna asli citra, karena *channel* H dan S merepresentasikan karakteristik warna yang tidak berkaitan langsung dengan tingkat kecerahan.

75	75	75	...	170	171	171
75	75	75	...	170	171	171
75	75	75	...	170	171	171
...	...	...	...	...	...	...
17	17	17	...	165	165	165
17	17	17	...	165	165	165
17	17	17	...	165	165	165

32	34	36	...	255	255	255
36	39	43	...	255	255	255
43	46	51	...	255	255	255
...	...	...	...	...	...	...
204	204	204	...	102	102	102
210	210	210	...	102	102	102
210	210	210	...	102	102	102

255	102	204	...	12	14	14
204	204	204	...	12	14	14
102	102	0	...	12	14	14
...	...	...	...	...	...	...
44	44	44	...	4	4	4
42	42	42	...	4	4	4
42	42	42	...	4	4	4

(a) (b) (c) (d)

Gambar 34. (a) Citra HSV, (b) Nilai Piksel Channel Hue, (c) Nilai Piksel Channel Saturation, (d) Nilai Piksel Channel Value

Konversi Kembali ke RGB. Proses konversi kembali ke ruang warna RGB bertujuan untuk mengembalikan representasi citra ke format standar yang umum digunakan dalam pengolahan citra digital.

14	14	12	..	3	4	4
12	12	10	..	3	4	4
10	10	7	..	3	4	4
..	..	..	..	..	..	..
8	8	8	..	3	3	3
7	7	7	..	3	3	3
7	7	7	..	3	3	3

16	16	13	..	0	0	0
13	13	11	..	0	0	0
11	11	8	..	0	0	0
..	..	..	..	..	..	..
28	28	28	..	2	2	2
27	27	27	..	2	2	2
27	27	27	..	2	2	2

13	13	11	..	12	14	14
11	11	9	..	12	14	14
9	9	6	..	12	14	14
..	..	..	..	..	..	..
44	44	44	..	4	4	4
42	42	42	..	4	4	4
42	42	42	..	4	4	4

(a) (b) (c) (d)

Gambar 35. (a) Citra RGB, (b) Nilai Piksel *Channel Red*, (c) Nilai Piksel *Channel Green*, (d) Nilai Piksel *Channel Blue*

3) *Post-Processing*

***Blending* Dengan Citra Original**. Citra hasil *preprocessing* AGC dan CLAHE dikombinasikan dengan citra asli untuk menjaga kestabilan hasil *enhancement*. Melalui proses *blending*, citra akhir tetap memiliki detail yang lebih jelas dibandingkan citra asli, namun tidak mengalami distorsi visual yang berlebihan. Meskipun AGC dan CLAHE telah dirancang agar menghasilkan peningkatan kontras yang natural, proses *blending* memastikan bahwa citra akhir tetap konsisten dengan karakter visual citra asli serta robust terhadap variasi kondisi pencahayaan.

Peningkatan Kecerahan Tambahan (*Brightnes Boost*). Tahap ini berfungsi untuk menambah visibilitas citra lubang pada malam hari tanpa merusak detail. Citra hasil *blending* dikonversi kembali ke HSV. Setelah itu, *channel* V dikalikan dengan faktor peningkatan kecerahan (1.1). Kemudian, nilai pada *channel* V akan dibatasi pada rentang (0, 255). Dan hasil akhirnya citra dikonversi kembali ke format RGB. Hasil akhir dari semua proses diatas dapat dilihat pada gambar 36.

14	14	12	...	3	4	4
12	12	10	...	3	4	4
10	10	7	...	3	4	4
...	...	...	...	...	...	...
8	8	8	...	2	2	2
7	7	7	...	2	2	2
7	7	7	...	2	2	2

17	17	14	...	0	0	0
14	14	12	...	0	0	0
12	12	9	...	0	0	0
...	...	...	...	...	...	...
28	28	28	...	1	1	1
27	27	27	...	1	1	1
27	27	27	...	1	1	1

13	13	11	...	13	15	15
11	11	9	...	13	15	15
9	9	6	...	13	15	15
...	...	...	...	...	...	...
44	44	44	...	4	4	4
42	42	42	...	4	4	4
42	42	42	...	4	4	4

(a) **(b)** **(c)** **(d)**

Gambar 36 (a) Citra Output Akhir, (b) Nilai Piksel *Channel Red*, (c) Nilai Piksel *Channel Green*, (d) Nilai Piksel *Channel Blue*

c. Deteksi Lubang

Proses deteksi dilakukan setelah citra lubang melalui tahap peningkatan kualitas citra dengan menggunakan metode AGC dan CLAHE. Citra yang telah diproses kemudian dimasukkan ke dalam model YOLOv11 sebagai input utama. Selanjutnya, model YOLOv11 akan menganalisis citra tersebut untuk mengenali pola visual yang merepresentasikan keberadaan lubang pada permukaan jalan.

d. Evaluasi Hasil Deteksi

Evaluasi hasil deteksi dilakukan untuk menilai performa model dalam mendeteksi lubang di jalan secara akurat. Hasil prediksi yang dihasilkan oleh model dibandingkan dengan data *ground truth* untuk mengetahui sejauh mana model mampu untuk mengenali lubang di jalan dengan benar. Metrik evaluasi yang digunakan yaitu *confusion matrix* yang terdiri dari empat variabel yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

2.5 Waktu dan Lokasi Penelitian

Penelitian ini dimulai pada bulan April 2025-Januari 2026. Penelitian dilaksanakan di Laboratorium Kecerdasan Buatan, Departemen Teknik Informatika, Universitas Hasanuddin, Jalan Poros Malino, Kelurahan Borongloe, Kecamatan Bontomarannu, Kabupaten Gowa, Provinsi Sulawesi Selatan