Metode *Deep Learning* Untuk *Lane Detection* Pada Kendaraan Otomatis di Berbagai Skenario Menggunakan CARLA Simulator

Thomi Aditya Alhakiim[1], Yuni Yamasari[2]

[1,2] S1 Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya

[1]thomi.19057@mhs.unesa.ac.id
[2]yuniyamasari@unesa.ac.id

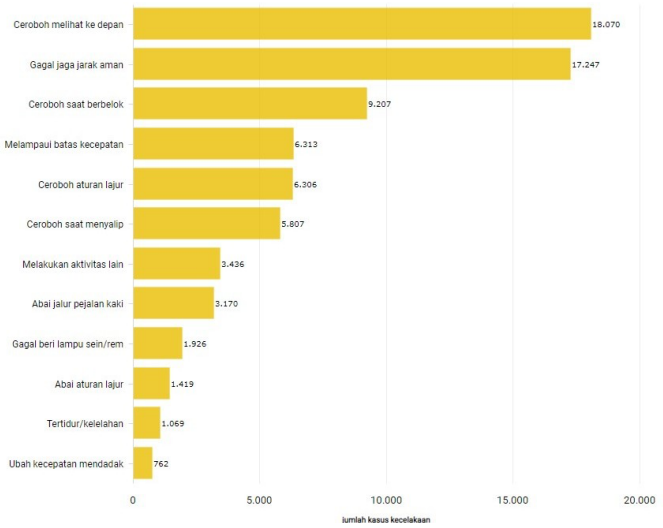
***Abstrak*— Dalam upaya meningkatkan keselamatan dan efisiensi kendaraan otomatis, akurasi dan kecepatan sistem *lane detection* di dalam kendaraan memainkan peranan penting. Namun, penelitian dalam domain itu tidak terlalu banyak. Oleh karena itu, penelitian ini memfokuskan pada analisis komparatif mendalam dari dua model yang digunakan: YOLOPv2 dan CLRNet. Kemudian, kedua model yang terbangun dilakukan simulasi dalam berbagai skenario cuaca dan kondisi lalu lintas. Untuk metrik evaluasi, penelitian ini menggunakan parameter *online* (FPS) dan *offline*. Hasil penelitian ini dapat digunakan untuk memandu pemilihan model *lane detection* untuk kendaraan otomatis berdasarkan aplikasi yang diinginkan dan kondisi cuaca yang berlaku. Hal ini diindikasikan dengan akurasi pada kedua model menunjukkan nilai yang sangat bagus di angka 96% untuk YOLOPv2 dan 98% untuk CLRNet. Kemudian, metrik lain menunjukkan keunggulan YOLOPv2 dalam hal kecepatan dan akurasi di hampir semua skenario yang diuji dengan IoU rata-rata 27% dan F1 *Score* 41%, dibandingkan dengan IoU rata-rata CLRNet sebesar 15% dan F1 *Score* 24%. YOLOPv2 juga diamati memiliki *False Positive* Rate yang lebih tinggi dibandingkan dengan CLRNet, menunjukkan potensi deteksi berlebih dalam beberapa skenario. Di sisi lain, CLRNet, meskipun lebih lambat dan kurang akurat, menunjukkan *False Positive* Rate yang sangat rendah, menjadikannya pilihan yang konservatif namun andal untuk situasi di mana kesalahan deteksi dapat memiliki konsekuensi yang parah.**

***Kata Kunci*— Kendaraan Otomatis, Deteksi Objek, YOLOPv2, CLRNet, Simulator CARLA, Kontrol Prediktif Model**

I. PENDAHULUAN

National *Highway Traffic Safety Administration* (NHTSA) telah menyajikan analisis keselamatan lalu lintas setiap tahunnya melalui publikasi terbaru berjudul "2020 *Traffic Safety Facts Annual Report*" [1] yang mencakup data statistik keselamatan lalu lintas dari tahun 2010 hingga 2020. Meskipun terjadi penurunan jumlah kecelakaan sekitar 22% pada tahun 2020 dibandingkan dengan tahun 2019, dan penurunan jumlah orang yang terluka sekitar 17%, angka kecelakaan fatal meningkat sekitar 6.8%. Kecelakaan fatal ini sebagian besar disebabkan oleh human *error*, seperti ketidakberkonsentrasian saat mengemudi, pengaruh alkohol, atau ketidakpatuhan terhadap penggunaan sabuk pengaman [1]. Di Indonesia, laporan dari Pusat Informasi Kriminal Nasional Polri menunjukkan bahwa faktor manusia juga mendominasi kecelakaan lalu lintas, dengan perilaku pengemudi menjadi penyebab utama [2].

Dalam konteks global, penelitian mengenai kendaraan otomatis atau *autonomous driving* menjadi sorotan, dengan



Gbr. 1 Data faktor penyebab terjadinya kecelakaan pada semester I 2022 [12].

harapan dapat mengurangi human *error* sebagai penyebab kecelakaan. National *Highway Traffic Safety Administration* (NHTSA) mencatat beberapa keuntungan kendaraan otomatis, seperti mengurangi jumlah kecelakaan, meningkatkan mobilitas, memberikan keuntungan ekonomi dan sosial, ramah lingkungan, dan meningkatkan efisiensi. Meskipun demikian, pengembangan kendaraan otomatis dihadapi dengan berbagai tantangan, termasuk kebutuhan akan sistem yang andal dan kepastian keselamatan penumpang [3].

Dalam upaya mengatasi tantangan tersebut, penelitian mengenai kendaraan otomatis memanfaatkan simulasi sebagai alat untuk pengujian dan evaluasi [4][5]. Platform simulasi seperti CARLA[6] menjadi populer dalam penelitian kendaraan otomatis karena menyediakan lingkungan yang terkontrol dan aman. Salah satu aspek kritis yang dihadapi kendaraan otomatis adalah *Lane Detection*, yang merupakan sub-problem dari *perception* problem [7]. *Lane Detection* bertujuan untuk mendeteksi garis jalan, penting untuk membantu kendaraan otomatis menghindari tabrakan. Oleh karena itu, penelitian ini bertujuan untuk menganalisis dan membandingkan teknik *deep learning* yang digunakan untuk *Lane Detection* dalam kendaraan otomatis, terutama dalam skenario simulasi CARLA.

Melalui pemahaman latar belakang tersebut, penelitian ini ditujukan untuk menganalisis dan membandingkan berbagai teknik *deep learning* untuk *Lane Detection* dalam kendaraan

otomatis, terutama dalam skenario simulasi CARLA. Diharapkan hasil penelitian memberikan wawasan yang berguna bagi pengembangan sistem *Lane Detection* yang lebih baik dan *robust* untuk kendaraan otomatis, serta memberikan evaluasi komprehensif pada berbagai teknik *Lane Detection* khususnya pada lingkungan CARLA simulator.

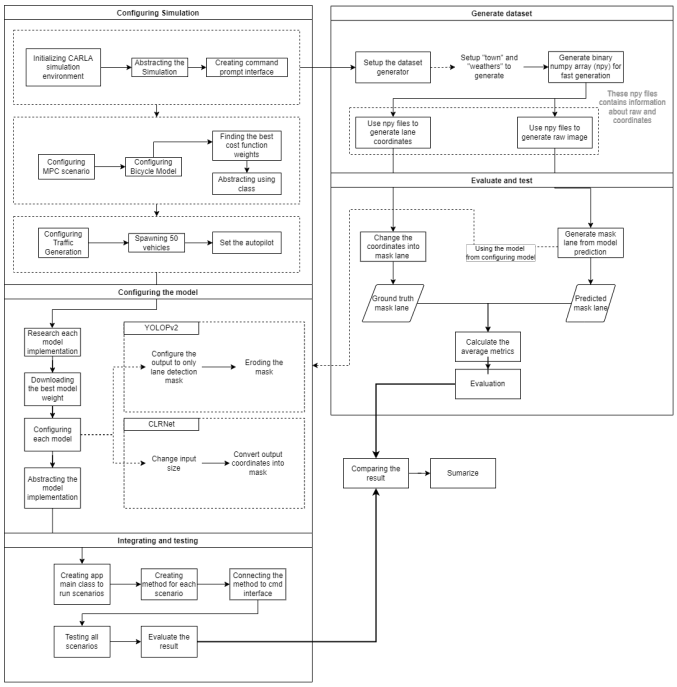
II. METODOLOGI PENELITIAN

A. *Metode Penelitian*

Penelitian ini akan mengadopsi metode penelitian kuantitatif dengan alasan bahwa data numerik akan digunakan untuk merumuskan kesimpulan. Pendekatan kuantitatif dipilih karena memungkinkan penggunaan data yang dapat diukur secara objektif. Metode ini akan diterapkan pada tahap pengumpulan data, analisis data, dan penarikan kesimpulan.

Dalam konteks desain penelitian, penelitian ini akan menggunakan pendekatan komparatif dan evaluatif. Penelitian komparatif bertujuan untuk membandingkan dua atau lebih variabel atau faktor. Dalam konteks ini, penelitian akan membandingkan model *deep learning* untuk *lane detection* guna mengidentifikasi model yang paling efektif dan efisien untuk digunakan dalam kendaraan otomatis. Variabel tidak akan dimanipulasi, melainkan dibandingkan untuk menemukan model yang paling sesuai untuk kendaraan otomatis.

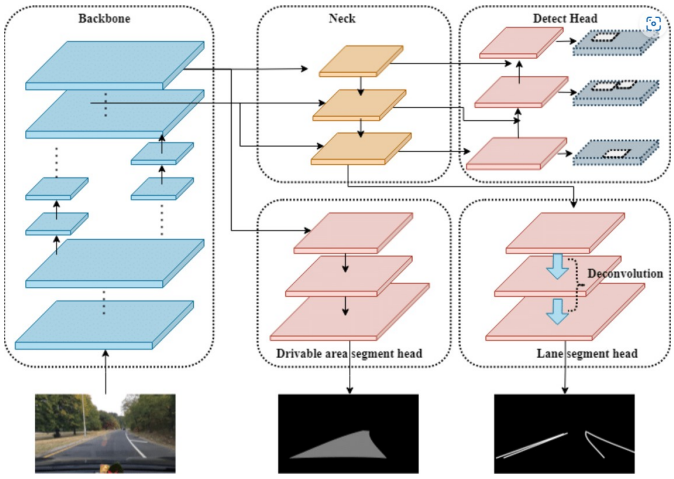
Di sisi lain, penelitian evaluatif bertujuan untuk mengevaluasi kinerja suatu sistem atau model. Fokus penelitian ini adalah menilai kinerja model *lane detector* yang diimplementasikan dalam kendaraan otomatis. Tujuan evaluasi adalah untuk menentukan teknik mana yang memberikan kinerja terbaik dalam berbagai skenario. Dengan kombinasi kedua desain tersebut, diharapkan penelitian ini dapat memberikan kontribusi yang signifikan dalam pengembangan sistem *lane detection* yang efektif dan dapat diandalkan untuk kendaraan otomatis.



B. *Pemilihan* Lane Detection *Model*

Pemilihan model *lane detection* yang tepat sangat penting untuk memastikan bahwa kendaraan otomatis dapat beroperasi dengan aman dan efisien. Model yang dipilih akan didasarkan dengan popularitas, ketersediaan, dan kinerja model tersebut. Berikut adalah beberapa model yang akan dipertimbangkan untuk digunakan dalam penelitian ini.

1) ***YOLOPv2:*** yang dijelaskan dalam *paper* "YOLOPv2: *Better*, *Faster*, *Stronger for Panoptic Driving Perception*" [8]merupakan pengembangan model YOLO khusus untuk persepsi mengemudi panoptik. Arsitektur ini dirancang untuk tugas deteksi objek lalu lintas, segmentasi area yang dapat dilalui, dan *lane detection*. YOLO sendiri adalah model efisien untuk *object detection* real-*time* yang menggunakan CNN untuk memprediksi *bounding box* dan probabilitas kelas langsung dari gambar secara keseluruhan. YOLOPv2 mempertahankan konsep desain YOLO tetapi dengan menggunakan *backbone* yang lebih kuat untuk ekstraksi fitur. Penggunaan skema deteksi *multi-scale* berbasis *anchor* dan penggunaan *Path Aggregation* Network (PAN) membuatnya mencapai kinerja terbaik dengan MAP 0.83 untuk deteksi objek,

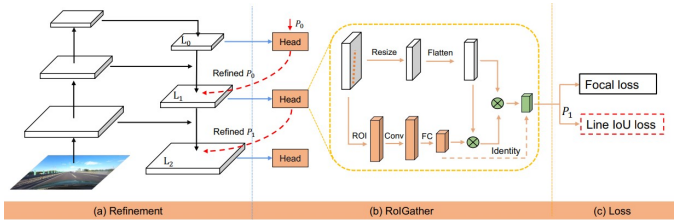


Gbr. 2 Arsitektur YOLOPv2 pada paper "YOLOPv2: Better, Faster, Stronger for Panoptic Driving Perception"[8].

MIOU 0.93 untuk segmentasi area yang dapat dilalui, dan akurasi 87.3 untuk *lane detection*.

2) ***CLRNet:*** "*Cross* Layer *Refinement* Network *for Lane Detection*"[9] memperkenalkan CLRNet sebagai kerangka kerja baru untuk deteksi jalur dalam kendaraan otomatis. CLRNet menggunakan fitur tingkat rendah dan tinggi untuk deteksi jalur, memanfaatkan konsep "*Lane Prior*" yang mencakup bentuk jalur sebelumnya. Model ini memprediksi *prior* jalur yang terdiri dari probabilitas latar depan dan belakang, panjang, titik awal, sudut, dan *offset*. Dengan progresi lintas lapisan, CLRNet melakukan penyempurnaan dari lapisan tinggi hingga rendah. Modul ROI-*Gather*

diperkenalkan untuk mengumpulkan informasi kontekstual yang berguna untuk belajar fitur jalur dengan lebih baik. Arsitektur ini berhasil mencapai hasil terbaik dengan memanfaatkan fitur lintas lapisan dan memperkenalkan modul ROIGather untuk mendukung deteksi jalur yang lebih baik.



Gbr. 4 Arsitektur CLRNet pada paper "Cross Layer Refinement Network for Lane Detection" [9].

C. *Skenario Realtime (Online)*

Dalam penelitian otomasi kendaraan, validasi model di lingkungan simulasi realistis sangat penting sebelum implementasi di dunia nyata. Simulasi *online* (realtime) dipilih sebagai metode utama validasi dengan alasan utama, termasuk pengujian model dalam skenario sulit atau berbahaya, seperti variasi kondisi cuaca dan kepadatan lalu lintas. Simulasi *online* memungkinkan pengukuran kinerja model, seperti *Frames Per Second* (FPS), dalam lingkungan dinamis, memberikan wawasan tentang respons model terhadap stimulus dalam waktu nyata.

Rumus untuk menghitung FPS adalah:

$$\text{FPS} = \frac{1}{\text{Average Time per Frame}} \quad (1)$$

Dalam penelitian ini, simulasi menggunakan CARLA versi 0.9.12, platform simulasi open-*source* terkemuka untuk penelitian kendaraan otomatis. Berbagai kondisi cuaca diuji, termasuk "*Clear Sunset*," "*Clear Night*," "*Hard Rain Sunset*," "*Hard Rain Night*," "*Cloudy Sunset*," "*Cloudy* Night," "*Wet Sunset*," dan "*Wet Night*." Setiap kondisi cuaca diuji dalam dua skenario: satu dengan lalu lintas yang dihasilkan oleh *traffic* generator dan satu lagi dengan kontrol menggunakan Model *Predictive Control* (MPC). Konfigurasi skenario ini memberikan pemahaman komprehensif tentang performa model dalam kondisi yang berbeda.

D. *Skenario Offline*

Setelah simulasi dan pengumpulan data selesai, langkah berikutnya adalah melakukan evaluasi *offline* terhadap model yang telah dikembangkan. Evaluasi *offline* merupakan tahap krusial dalam penelitian ini karena memungkinkan untuk menilai kinerja model dalam lingkungan yang terkontrol dan konsisten, tanpa variabilitas yang mungkin terjadi dalam simulasi real-*time*. *Dataset* yang telah dihasilkan dari simulator CARLA melalui *repository* "Carla-*Lane-Detection-Dataset-Generation*"[10] digunakan sebagai dasar untuk evaluasi ini. *Repository* ini menyajikan kustomisasi simulator berbagai kondisi jalan, cuaca, dan situasi lalu lintas yang memungkinkan evaluasi komprehensif terhadap model.

Dalam mengevaluasi kinerja model untuk segmentasi jalur, beberapa metrik digunakan dengan rumus sebagai berikut:

1) **Intersection *over Union (IoU):***

$$IoU = \frac{\text{Area of Overlap (antara prediksi dan ground truth)}}{\text{Area } of \text{ Union (antara prediksi dan } ground\ truth)} \quad (2)$$

2) **Dice Coefficient *(DSC):***

$$DSC = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (3)$$

3) **Precision, Recall, *dan F1* Score*:***

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

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

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{Precision + \text{Recall}} \quad (4)$$

4) ***False* Positive *Rate dan* False Negative *Rate:***

$$False\ Positive\ \text{Rate} = \frac{FP}{FP + TN}$$

$$False\ Negative\ \text{Rate} = \frac{FN}{FN + TP} \quad (5)$$

Dengan menggunakan metrik-metrik ini, evaluasi *offline* dapat memberikan gambaran yang holistik tentang kinerja model dalam deteksi jalur pada lingkungan simulasi yang telah dikustomisasi.

E. *Konfigurasi Model* Predictive Control *(MPC) pada Simulasi CARLA*

1) ***Model Kendaraan:***

Model kinematik kendaraan yang digunakan dalam penelitian ini adalah model kinematik *bicycle* model. Meskipun sederhana, model ini memberikan representasi yang cukup baik untuk keperluan kontrol dalam lingkungan simulasi. Persamaan kinematik model ini mencakup posisi, orientasi, kecepatan, kesalahan lintasan, dan kesalahan orientasi kendaraan. Dengan asumsi dua roda terpisah yang terhubung oleh batang, model ini memberikan perkiraan yang memadai tanpa mempertimbangkan gaya kompleks seperti gaya ban atau aerodinamika.

Persamaan Kinematik [11]**:**

$$x[t] = x[t-1] + v[t-1] \times \cos(\psi[t-1]) \times dt$$

$$y[t] = y[t-1] + v[t-1] \times \sin(\psi[t-1]) \times dt$$

$$\psi[t] = \psi[t-1] + v[t-1] \times \delta[t-1] \times dt \quad (6)$$

$$v[t] = v[t-1] + a[t-1] \times dt$$

$$cte[t] = f(x[t-1]) - y[t-1] + v[t-1] \times \sin(\epsilon[t-1]) \times dt$$

$$\epsilon[t] = \psi[t] - \psi_{\text{des}}[t-1] + v[t-1] \times \delta[t-1] \times dt$$

2) Fungsi Biaya (Cost Function):

Fungsi biaya dalam MPC didefinisikan sebagai kombinasi dari kesalahan keadaan dan *input* kontrol, dengan bobot untuk masing-masing komponen. Berikut adalah komponen fungsi biaya yang melibatkan kesalahan lintasan, kesalahan orientasi, kecepatan, percepatan, sudut kemudi, perubahan percepatan, dan perubahan sudut kemudi.

$$J = CTEW \times CTE^2 + EPSIW \times \epsilon^2 + VW \times (v - v_{\text{desired}}) + AW \times a^2 + DELTAW \times \delta^2 + ArateW \times \left(\frac{\Delta t}{\Delta \delta}\right)^2 \quad (7)$$

3) Latensi:

Latensi dalam konteks MPC untuk mengemudi adalah penundaan antara saat perintah dikeluarkan dan kendaraan benar-benar mulai berbelok. Dengan mempertimbangkan faktor-faktor seperti keadaan saat ini dan koefisien polinomial, latensi diatur ke 100 milidetik, memberikan nilai realistis untuk kendaraan.

4) Referensi Trajektori:

Trajektori referensi direpresentasikan sebagai polinomial yang diperbarui berdasarkan keadaan kendaraan dan jalur yang diinginkan. Kesalahan lintasan dan orientasi dihitung, memungkinkan MPC menghasilkan *input* kontrol optimal untuk meminimalkan kesalahan tersebut. Trajektori referensi dihasilkan dan diperbarui melalui CARLA simulator, diperbarui setiap kendaraan melewati *waypoint* yang terdekat.

Melalui pemahaman dan integrasi elemen-elemen ini, MPC pada simulasi kendaraan otomatis diharapkan dapat memberikan kontrol yang efisien dan aman.

Sebagai catatan, rumus dan konsep dalam rangkuman ini disesuaikan berdasarkan informasi yang disediakan sebelumnya dan pengaturan spesifik dalam penelitian tersebut. Jika ada informasi tambahan atau detail yang diperlukan, disarankan untuk merujuk langsung pada sumber asli jurnal penelitian.

F. Konfigurasi Skenario Trafflc Generation

Dalam evaluasi *online* sistem, konfigurasi lalu lintas memegang peranan krusial untuk mendapatkan gambaran realistis dalam kondisi dunia nyata. Dalam simulasi tersebut, kendaraan utama yang digunakan adalah tipe *'model3'*, dilengkapi dengan sensor kamera pada posisi tertentu, dan dioperasikan dalam mode *autopilot*. *Autopilot* CARLA menggunakan pengontrol PID (*Proportional*, *Integral*, *Derivative*) untuk mengendalikan gerakan kendaraan. Meskipun PID telah digunakan secara luas, terdapat keterbatasan, terutama dalam menghadapi perubahan lingkungan yang tidak dapat diprediksi.

Berbeda dengan metode MPC, yang memungkinkan prediksi respons masa depan berdasarkan model matematika sistem dan mengoptimalkan tindakan kontrol. Dalam konteks mengemudi, MPC dapat merencanakan lintasan dengan mempertimbangkan rintangan atau kondisi lalu lintas yang dikenal di depan, menunjukkan keunggulan dalam keadaan kompleks dan dinamis. Meskipun kontrol PID mungkin memadai untuk beberapa skenario simulasi, keberadaan pendekatan MPC akan memberikan performa yang lebih baik dalam kondisi lalu lintas kota nyata.

Selain itu, untuk menciptakan situasi lalu lintas yang lebih realistis, ditambahkan sejumlah kendaraan tambahan ke dalam simulasi menggunakan *Traffic* Generator. Kendaraan-kendaraan ini ditempatkan secara acak di seluruh peta dan diatur dalam mode *autopilot*, menciptakan lalu lintas yang dinamis dan interaksi dengan kendaraan utama. Pendekatan ini bertujuan untuk meningkatkan realisme dan kompleksitas lingkungan simulasi.

III. HASIL DAN PEMBAHASAN

Pengujian dilakukan menggunakan GPU NVIDIA *GeForce* GTX 750Ti, yang mungkin tidak sebanding dengan GPU kelas atas yang umumnya digunakan dalam penelitian penglihatan komputer. Meskipun GTX 750Ti terlihat kurang canggih dibandingkan dengan varian GPU terbaru, penelitian ini memberikan signifikansi pada penggunaan GPU ini dalam konteks eksperimen.

Dalam simulator CARLA, metrik kinerja utama adalah *Frames* Per *Second* (FPS), menjadi penting karena CARLA memerlukan komputasi intensif untuk simulasi lingkungan bergerak secara realistis. Menggabungkannya dengan model *lane detection* real-*time* menempatkan tekanan tambahan pada GPU. Oleh karena itu, penggunaan GTX 750Ti bertujuan untuk memberikan metrik kinerja dasar, menunjukkan bagaimana model-model ini berkinerja dalam lingkungan dengan sumber daya yang terbatas.

A. Komparasi Karakteristik Model YOLOPv2 dan CLRNet

Dilakukan perbandingan antara dua metode *deep learning*, YOLOPv2 dan CLRNet, yang telah dipilih untuk deteksi jalur dalam kendaraan otomatis. Keduanya bertujuan mengatasi tantangan *lane detection* yang melibatkan berbagai kondisi lingkungan. CLRNet menggunakan fitur tingkat tinggi dan tinggi dengan ukuran *input* 640x360, menghasilkan koordinat garis lajur dengan 23,3 juta parameter, dan mengoptimalkan tujuannya dengan Line IoU *loss* dan *Cross entropy loss*, mencapai F1-score 71,8% pada *dataset* CULane. Di sisi lain, YOLOPv2, dengan ukuran *input* 1280x720, melakukan *multi-task learning* untuk deteksi objek, segmentasi area dapat dilalui, dan *lane detection*. Dengan 61,9 juta parameter, mengoptimalkan tujuannya dengan *Focal loss* dan *Dice loss*, dan mencapai F1-score 69,5% pada *dataset* CULane. Keduanya menggunakan *multi-scale feature fusion* dan *post-*

processing, tetapi berbeda dalam cara *encode* dan *decode* informasi lajur, kompleksitas komputasi, dan kinerja pada *dataset benchmark*. Perbandingan karakteristik utama diuraikan dalam Tabel I.

Tabel I.
Perbandingan karakteristik CLRNet dan YOLOPv2

Karakteristik	**CLRNet**	**YOLOPv2**
Ukuran *input*	640x360	1280x720
Format *input*	Koordinat garis lajur	*Mask* biner garis lajur
Jumlah parameter	23,3 juta	61,9 juta
Fungsi *loss*	Line IoU *loss* + *Cross entropy loss*	*Focal loss* + *Dice loss*
Akurasi pada salah satu *dataset* (CU- *Lane*)	F1-score: 71,8%	F1-score: 69,5%

B. Presentasi Hasil Eksperimen Online

Pada bagian ini, akan dibedah dengan mendalam hasil eksperimen dari dua model *lane detection* yang sedang dalam sorotan, yakni YOLOPv2 dan CLRNet. Sebuah eksperimen yang memadai akan mempertimbangkan berbagai skenario operasi, terutama dalam konteks sistem kendaraan otomatis di mana kondisi cuaca dan lingkungan memainkan peran kritis.

Untuk memperjelas perbandingan, akan dibandingkan juga dengan hasil dari metode *OpenCV* Tradisional (tanpa menggunakan *deep learning*). Berikut adalah tabel-tabel hasil.

Tabel II.
Keseluruhan hasil FPS di berbagai kondisi cuaca. (FPS: Frame*/s)*

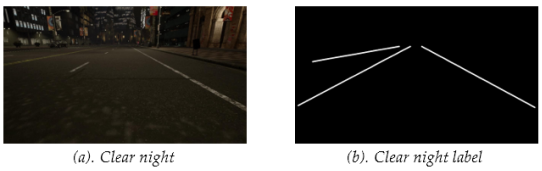
Skenario Online-MPC			
Kondisi cuaca	**YOLOPv2**	**CLRNet**	***Baseline***
ClearSunset	2.4165	1.4472	2.6866
ClearNight	2.4189	1.6055	2.7625
HardRainSunset	2.6002	1.4988	2.9161
HardRainNight	2.4619	1.3402	2.7875
CloudySunset	2.5097	1.6216	3.1112
CloudyNight	2.4941	1.5708	2.4039
WetSunset	2.4134	1.7088	2.9866
WetNight	2.4089	1.6721	2.6757
***Skenario Online*-Traffic Generation**			
Kondisi cuaca	**YOLOPv2**	**CLRNet**	***Baseline***
ClearSunset	0.7134	0.1139	10.235
ClearNight	0.5772	0.1155	5.7883
HardRainSunset	0.6544	0.11	6.1778
HardRainNight	0.5641	0.1001	6.4111
CloudySunset	0.7977	0.1212	7.1958
CloudyNight	0.6465	0.1173	5.7673
WetSunset	0.7331	0.1277	7.3933
WetNight	0.679	0.1249	6.6391

*1) **Skenario Online-MPC:*** YOLOPv2, dievaluasi dalam skenario Online-MPC, menunjukkan performa FPS yang stabil pada berbagai kondisi cuaca, seperti ClearSunset dan ClearNight dengan FPS sekitar 2.4165 dan 2.4189. Bahkan pada kondisi hujan lebat dan matahari terbenam (HardRainNSunset), model mencapai FPS tertingginya, yaitu 2.6002, menunjukkan adaptabilitasnya terhadap kondisi cuaca yang kurang ideal. Sebaliknya, CLRNet dalam skenario yang sama dengan penggunaan MPC menunjukkan FPS lebih rendah dibandingkan YOLOPv2. Performanya konsisten dengan FPS tertinggi pada WetSunset (1.7088 FPS) dan terendah pada HardRainNight (1.3402 FPS). CLRNet menunjukkan stabilitas performa namun mungkin memerlukan lebih banyak sumber daya atau waktu untuk proses deteksi, terutama dalam kondisi visual kompleks. Meskipun kedua model ini masih tergolong lambat jika dibandingkan dengan *Baseline* tanpa *deep learning*, YOLOPv2 tetap menjadi pilihan lebih baik dalam mempertahankan kecepatan. Namun, evaluasi lebih lanjut diperlukan untuk aspek lain dari performa, seperti akurasi, reliabilitas, dan kemampuan adaptasi.

2) ***Skenario Online*-Traffic Generation*:*** Dalam skenario Online-*Traffic Generation*, YOLOPv2 mengalami penurunan signifikan dalam FPS pada kondisi ClearSunset, ClearNight, HardRainSunset, dan HardRainNight, mencapai 0.7134, 0.5772, 0.6544, dan 0.5641 masing-masing. Kompleksitas tambahan dari objek bergerak dan kondisi cuaca ekstrem mempengaruhi performa model. Sementara itu, CLRNet menunjukkan FPS yang lebih rendah daripada YOLOPv2 pada kondisi ClearSunset dan ClearNight (0.1146 dan 0.1155), serta terus menurun pada HardRainSunset (0.1100) di bawah kondisi hujan lebat. YOLOPv2 tetap lebih cepat meskipun mengalami penurunan, menunjukkan potensi adaptasi yang lebih baik terhadap lingkungan dengan gangguan dan perubahan. CLRNet, tampaknya rentan terhadap kompleksitas lalu lintas bergerak, mungkin lebih sesuai untuk situasi dengan latar belakang statis atau kurang kompleks. Model *Baseline*, dengan kontrol PID yang membutuhkan sedikit sumber daya, berada jauh di bawah kedua model *deep learning*, menunjukkan efek dari proses deteksi yang kompleks pada performa model tersebut.

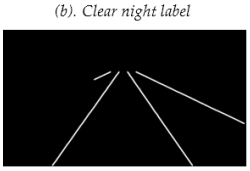
C. Dataset Generation

Dalam proses evaluasi *offline*, ada beberapa aspek penting yang harus dipertimbangkan ketika membandingkan kinerja dari dua model deteksi objek yang berbeda, dalam hal ini YOLOPv2 dan CLRNet. Evaluasi ini bertujuan untuk memberikan gambaran komprehensif tentang bagaimana setiap


(a). Clear night

(b). Clear night label


(c). Clear sunset


(d). Clear sunset label

Tabel III.
Metrik evaluasi pada setiap model yang diuji. Metrik berupa average (rata-rata).

Model	IoU (%)	F1 (%)	Prec (%)	Rec(%)	DC(%)	FPR(%)	FNR(%)	Acc (%)
YOLOPv2	**0.2668**	**0.4095**	**0.3143**	**0.7055**	**0.4095**	**0.0283**	**0.2945**	**0.9675**
CLRNet	**0.0840**	**0.1422**	**0.3255**	**0.0947**	**0.1422**	**0.0021**	**0.9053**	**0.9835**
Baseline	**0.0840**	**0.1422**	**0.3255**	**0.0947**	**0.1422**	**0.0021**	**0.9053**	**0.9835**

model berperilaku dalam skenario yang berbeda dan berdasarkan berbagai metrik.

Untuk melakukan evaluasi metrik keakuratan deteksi model, maka harus dilakukan *generation* untuk mengambil *sample ground truth*, lalu membandingkannya dengan hasil dari model-model yang diteliti. Contoh hasil *dataset generation* ada pada Gbr. 6.

D. Skenario Offline *menggunakan* Dataset Generation

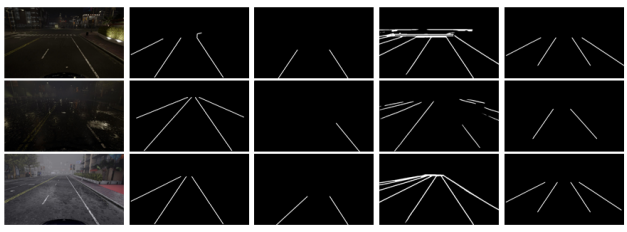
Model dievaluasi menggunakan *dataset* generasi yang mencakup gambar yang telah diolah dengan segmentasi *lane* untuk deteksi objek. Evaluasi dilakukan di Google Colab dengan fokus pada metrik utama seperti *Average* IoU, F1 *Score*, *Precision*, dan lainnya.

Hasil evaluasi menunjukkan perbedaan signifikan antara YOLOPv2 dan CLRNet. YOLOPv2 unggul dalam beberapa metrik utama, menunjukkan kemampuan identifikasi objek dengan tepat dan akurasi yang lebih tinggi, terutama dalam konteks deteksi objek di lingkungan lalu lintas. Namun, CLRNet memiliki *False Positive* Rate yang lebih rendah, menunjukkan kehati-hatian dalam menghasilkan deteksi dan mengurangi risiko mengidentifikasi objek palsu.

Tabel III memberikan rincian metrik evaluasi untuk model *baseline*, YOLOPv2, dan CLRNet. Kedua model menunjukkan peningkatan signifikan dibandingkan dengan model *baseline*, menunjukkan bahwa model yang dievaluasi lebih baik dalam mengidentifikasi objek dengan akurasi yang lebih tinggi. Namun, penilaian terhadap model sebaiknya tidak hanya berdasarkan angka, tetapi juga mempertimbangkan konteks aplikasi dan prioritas yang diberikan pada berbagai aspek kinerja.

Dalam Tabel III, metrik evaluasi untuk model *baseline* menunjukkan variasi performa antar kategori. Sebagai contoh, HRN memiliki nilai *Average* IoU sebesar 0.0840, F1 *Score* sebesar 0.1422, dan *Precision* sebesar 0.3255. Sebaliknya, YOLOPv2 menunjukkan peningkatan yang signifikan, dengan nilai *Average* IoU sebesar 0.2668, F1 *Score* sebesar 0.4095, dan *Precision* sebesar 0.3143. Begitu juga dengan CLRNet yang menunjukkan peningkatan kinerja, meskipun dengan karakteristik yang berbeda dengan nilai *Average* IoU sebesar 0.1489, F1 *Score* sebesar 0.2381, dan *Precision* sebesar 0.3317.

Perlu diingat bahwa nilai-nilai ini memberikan gambaran komprehensif tentang keakuratan dan performa kedua model, namun keputusan pemilihan model sebaiknya dipertimbangkan secara menyeluruh, termasuk konteks aplikasi dan kepentingan khusus dalam *lane detection*.



(a). Gambar (b). GT (c). Baseline (d). YOLO (e). CLRNet

Gbr. 7 Visualisasi hasil prediksi dari kedua model.

E. Perbandingan dan Analisis Model YOLOPv2 dan CLRNet

Melalui evaluasi pada berbagai skenario, YOLOPv2 menunjukkan keunggulan dalam hampir semua aspek dibandingkan dengan CLRNet, baik dalam metrik *online* maupun *offline*. Dari sisi FPS, YOLOPv2 menonjol dengan kemampuan deteksi real-*time* yang impresif, menegaskan respons cepatnya. Pada metrik *offline* seperti IoU dan F1 *Score*, YOLOPv2 kembali mendominasi, menunjukkan tingkat akurasi yang tinggi dalam mengidentifikasi objek secara tepat.

YOLOPv2, dengan arsitektur yang dioptimalkan, menggabungkan kecepatan dan akurasi, menjadikannya pilihan favorit untuk aplikasi real-*time* yang membutuhkan respons cepat. Namun, perlu dicatat bahwa model ini memiliki tingkat *False Positive* yang relatif tinggi, yang dapat menjadi tantangan dalam konteks *lane detection* karena cenderung mengidentifikasi objek yang bukan target deteksi. Ini dapat menjadi masalah serius dalam aplikasi keamanan atau navigasi kendaraan otomatis.

Di sisi lain, CLRNet, meskipun tidak secepat atau seakurat YOLOPv2, memiliki keunggulan khusus dalam mengurangi *False Positive*. Tingkat kesalahan deteksi yang rendah menunjukkan kehati-hatian dan selektivitas model ini dalam mengidentifikasi objek. Oleh karena itu, CLRNet mungkin menjadi pilihan lebih baik untuk aplikasi di mana kesalahan deteksi dapat memiliki konsekuensi fatal, seperti dalam bidang medis atau inspeksi industri. Pendekatan konservatif CLRNet menjadi prioritas untuk situasi di mana kesalahan kecil dapat berdampak serius.

IV. KESIMPULAN

YOLOPv2 lebih unggul dibanding CLRNet dengan rata-rata IoU 27% dan F1 *Score* 41% berbanding 15% dan 24%. Selain itu, YOLOPv2 mendeteksi objek di sekitar kendaraan lebih baik, meskipun kurang efektif mengenali objek jauh karena *backbone* yang lebih sederhana. Kecepatan dan akurasi tinggi

dari YOLOPv2 diimbangi dengan tingkat *False Positive* yang relatif tinggi. Untuk CLRNet, model ini lebih lambat dan kurang akurat, namun memiliki keunggulan dalam mengurangi tingkat *False Positive* sehingga model ini ebih cocok untuk aplikasi di mana kesalahan deteksi dapat berdampak serius, seperti kendaraan otomatis.

V. SARAN

Saran untuk penelitian mendatang termasuk penyesuaian YOLOPv2 untuk mengurangi *False Positive*, melalui augmentasi data atau *fine-tuning* dengan *dataset* jalan raya yang lebih beragam. Pertimbangan untuk mengintegrasikan keunggulan CLRNet dengan kecepatan dan akurasi YOLOPv2 dapat dieksplorasi melalui *ensemble* atau integrasi fitur keduanya.

Uji model dalam kondisi berkendara yang lebih ekstrim dan integrasi sensor tambahan seperti LIDAR atau radar dapat meningkatkan deteksi objek. *Kerjasama* dengan industri otomotif dianjurkan untuk mengaitkan penelitian dengan kebutuhan kendaraan otonom di lapangan.

UCAPAN TERIMA KASIH

Dengan penuh syukur, saya ingin menyampaikan rasa terima kasih kepada Allah SWT atas berkah dan rahmat-Nya, yang membimbing saya hingga berhasil menyelesaikan penelitian dengan judul "Metode *Deep Learning* Untuk *Lane Detection* Pada Kendaraan Otomatis di Berbagai Skenario Menggunakan CARLA Simulator." Keberhasilan ini tidak lepas dari dukungan berbagai pihak, dan pada kesempatan ini, saya ingin mengucapkan terima kasih kepada:

1) Orang tua tercinta, yang senantiasa memberikan dukungan, motivasi, dan semangat, memungkinkan saya menyelesaikan penelitian ini.
2) Dr. Yuni Yamasari S.Kom., M.Kom., sebagai Dosen Pembimbing yang telah memberikan arahan dan bimbingan dengan penuh dedikasi.
3) Dr. Ricky Eka Putra, S.Kom., M.Kom., dosen penguji yang memberikan saran berharga dalam penyusunan penelitian ini.
4) Ronggo Alit, M.M., M.T., selaku dosen penguji yang memberikan masukan dan saran yang berkontribusi pada penelitian ini.
5) Seluruh Dosen Teknik Informatika yang telah berbagi ilmu dan pengalaman selama penulis menjalani studi di Program Studi S1 Teknik Informatika.

REFERENSI

[1] N. Media, "NHTSA releases 2020 traffic crash data,"
NHTSA. NHTSA, Maret 2022. [Daring]. Tersedia pada: https://www.nhtsa.gov/press-releases/2020-traffic-crash-data-fatalities

[2] P. B. Polri, "Jurnal Data Pusiknas Bareskrim Polri Semester I Tahun 2022," *Jurnal Data Pusiknas Bareskrim Polri*, 2022.

[3] N. H. T. S. A. (NHTSA), "*Automated vehicles for safety*," *NHTSA*. [Daring]. Tersedia pada: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[4] L. *Bruck*, B. *Haycock*, dan A. *Emadi*, "A *review of driving simulation technology and applications*," *IEEE Open* Journal of Vehicular *Technology*, vol. 2, hlm. 1–16, 2021, doi: 10.1109/ojvt.2020.3036582.

[5] G. Yang *dkk.*, "*Survey on Autonomous Vehicle Simulation Platforms*," *2021 8th International* Conference on Dependable *Systems* and Their Applications *(DSA)*, 2021, doi: 10.1109/dsa52907.2021.00100.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. *López*, dan V. *Koltun*, "CARLA: An Open Urban *Driving* Simulator," *CoRR*, vol. *abs*/1711.03938, 2017, [Daring]. Tersedia pada: http://arxiv.org/abs/1711.03938

[7] *Abdulhakam*. AM. Assidiq, O. O. Khalifa, Md. R. Islam, dan S. Khan, "Real *time lane detection for Autonomous Vehicles*," *2008 International* Conference on Computer and Communication *Engineering*, 2008, doi: 10.1109/iccce.2008.4580573.

[8] C. Han, Q. *Zhao*, S. *Zhang*, Y. Chen, Z. *Zhang*, dan J. Yuan, "YOLOPv2: *Better*, *Faster*, *Stronger for Panoptic Driving Perception*." 2022.

[9] T. *Zheng dkk.*, "CLRNet: *Cross* Layer *Refinement* Network *for Lane Detection*." 2022.

[10] M. G. *Heck*, "Glutamat42/Carla-*lane-detection-dataset-generation*: *Create* a *dataset to train* a *lane detection* neural *network with* Carla," GitHub. 2021. [Daring]. Tersedia pada: https://github.com/Glutamat42/Carla-Lane-Detection-Dataset-Generation

[11] A. Escarlate, "*Implementing* a model *predictive control for* a *self-driving car*," *Medium*. Medium, Januari 2018. [Daring]. Tersedia pada: https://cacheop.medium.com/implementing-a-model-predictive-control-for-a-self-driving-car-7ee6212a04a8

[12] "*Hati-Hati* di Jalan, ini 12 Faktor Penyebab Kecelakaan Lalu Lintas: *Databoks*," *Pusat Data Ekonomi dan Bisnis Indonesia*. Februari 2023. [Daring]. Tersedia pada: https://databoks.katadata.co.id/datapublish/2023/02/09/hati-hati-di-jalan-ini-12-faktor-penyebab-kecelakaan-lalu-lintas