Evaluasi Keandalan *Monocular Depth Estimation* Berbasis *Deep Learning* terhadap *Adversarial Patch*

Laporan Tugas Akhir

Disusun sebagai syarat kelulusan tingkat sarjana

Oleh

ARDHANA PUTRA WIBOWO

NIM : 18219009

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Januari 2026

Evaluasi Keandalan *Monocular Depth Estimation* Berbasis *Deep Learning* terhadap *Adversarial Patch*

Laporan Tugas Akhir

Oleh

ARDHANA PUTRA WIBOWO

NIM : 18219009

Program Studi Sistem dan Teknologi Informasi

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai Laporan Tugas Akhir

di Bandung, pada tanggal 11 Januari 2026

Pembimbing,

Signed by BUDI RAHARDJO (HA5937)
Signed on Jan 11, 2026 08:42:04

Ir. Budi Rahardjo, M.Sc., Ph.D.

NIP 109110001

LEMBAR PERNYATAAN

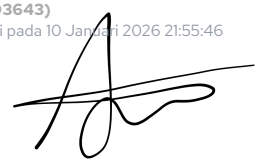
Dengan ini saya menyatakan bahwa:

1. Pengerjaan dan penulisan Laporan Tugas Akhir ini dilakukan tanpa menggunakan bantuan yang tidak dibenarkan.
2. Segala bentuk kutipan dan acuan terhadap tulisan orang lain yang digunakan di dalam penyusunan laporan tugas akhir ini telah dituliskan dengan baik dan benar.
3. Laporan Tugas Akhir ini belum pernah diajukan pada program pendidikan di perguruan tinggi mana pun.

Jika terbukti melanggar hal-hal di atas, saya bersedia dikenakan sanksi sesuai dengan Peraturan Rektor ITB No. 257 tahun 2019 tentang Penegakan Norma Akademik dan Kemahasiswaan Institut Teknologi Bandung.

Bandung, 10 Januari 2026

Ditandatangani oleh ARDHANA PUTRA WIBOWO (BO3643)
Ditandatangani pada 10 Januari 2026 21:55:46



Ardhana Putra Wibowo

NIM 18219009

ABSTRAK

EVALUASI KEANDALAN *MONOCULAR DEPTH ESTIMATION* BERBASIS *DEEP LEARNING* TERHADAP *ADVERSARIAL PATCH*

Oleh

ARDHANA PUTRA WIBOWO

NIM : 18219009

Monocular depth estimation (MDE) merupakan komponen penting dalam sistem persepsi kendaraan otonom yang menyediakan informasi jarak untuk fungsi lanjutan seperti rekonstruksi lingkungan dan deteksi rintangan. Walaupun berbagai model MDE berbasis *deep learning* telah mencapai kinerja *state-of-the-art*, diketahui bahwa model berbasis *deep learning* rentan terhadap serangan adversarial, termasuk *adversarial patch*. Tugas akhir ini bertujuan untuk mengetahui ketahanan model MDE terhadap serangan *adversarial patch* serta mengidentifikasi karakteristik *adversarial patch* yang memengaruhi efektivitas serangan. Tugas akhir ini menggunakan metodologi CRISP-DM. Proses pembangkitan dan evaluasi *patch* memanfaatkan dataset kendaraan otonom KITTI yang telah dimodifikasi. Model-model yang dibandingkan yaitu model berbasis CNN (Monodepth2 dan Manydepth) serta *transformer* (Depth Anything). Metrik-metrik yang digunakan selama evaluasi yaitu *absolute relative error*, *threshold accuracy*, dan keluaran *disparity* dari model secara langsung. Hasil evaluasi menujukkan bahwa *adversarial patch* yang dibangkitkan mampu menimbulkan kesalahan prediksi kedalaman yang signifikan, dengan peningkatan nilai *absolute relative error* menjadi 66% pada skenario *white-box* dan 26% pada skenario *black-box*. Penemuan dalam tugas akhir ini diharapkan dapat menjadi landasan pengembangan strategi *adversarial defense* untuk meningkatkan keandalan model MDE terhadap serangan adversarial.

Kata kunci: pembelajaran mendalam, *monocular depth estimation*, *adversarial patch*, *adversarial robustness*

KATA PENGANTAR

Puji dan syukur kepada Tuhan Yang Maha Esa, Allah SWT, atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Evaluasi Keandalan *Monocular Depth Estimation* berbasis *Deep Learning* terhadap *Adversarial Patch*" dengan baik. Tugas akhir ini disusun sebagai syarat kelulusan tingkat sarjana dalam Program Studi Sistem dan Teknologi Informasi Institut Teknologi Bandung. Selama penyusunan tugas akhir ini, penulis memperoleh banyak dukungan dan bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

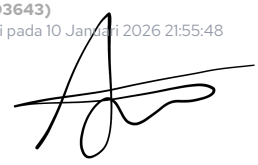
1. Bapak Ir. Budi Rahardjo, M.Sc., Ph.D. selaku dosen pembimbing, yang telah memberikan bimbingan, arahan, umpan balik, serta motivasi yang sangat membantu penulis dalam menyelesaikan tugas akhir ini.
2. Bapak I Gusti Bagus Baskara Nugraha, S.T., M.T., Ph.D selaku Ketua Program Studi Sistem dan Teknologi Informasi, penguji seminar proposal, dan dosen koordinator mata kuliah II4092 yang telah memberikan dukungan, arahan, dan sarana selama penulis menempuh pendidikan di program studi ini.
3. Ibu Dr. Fetty Fitriyanti Lubis, S.T., M.T., sebagai dosen koordinator mata kuliah II4092 yang telah membantu kelancaran proses akademik mata kuliah ini, serta telah memberikan dukungan, arahan, dan sarana selama penyelesaian tugas akhir ini.
4. Ibu Dr. Lenny Putri Yulianti, S.T., M.T. selaku dosen wali, yang telah memberikan bimbingan dan motivasi selama masa perkuliahan.
5. Kedua orang tua penulis, kedua adik penulis, serta keluarga besar penulis yang senantiasa memberikan dukungan dan doa sepanjang penyusunan tugas akhir ini.
6. Seluruh tenaga pendidik yang telah memberikan ilmu yang bermanfaat selama masa studi penulis di ITB.

7. Seluruh jajaran staf tata usaha Sekolah Teknik Elektro dan Informatika (STEI) ITB yang memfasilitasi urusan administrasi selama masa studi penulis.
8. Teman-teman penulis di grup “L.O.V.E.M.U.F.F.I.N.”, dan teman SMA penulis yang membantu mengurangi kejenuhan penulis selama penyusunan tugas akhir.
9. Teman-teman ASYNC, teman-teman magang penulis, dan teman seperjuangan bimbingan yang telah memberikan dukungan dan menemani penulis selama penyusunan tugas akhir.
10. Kanal YouTube ‘GothamChess’ yang menemani penulis selama penyusunan tugas akhir ini.
11. Seluruh pihak lain yang tidak bisa disebutkan satu per satu yang telah membantu penulis dalam proses pengerjaan tugas akhir.

Penulis berharap tugas akhir ini dapat memberikan kontribusi positif dalam pengembangan ilmu pengetahuan di bidang teknologi informasi, khususnya dalam bidang *adversarial robustness* dan *computer vision*.

Bandung, 10 Januari 2026

Ditandatangani oleh ARDHANA PUTRA WIBOWO (BO3643)
Ditandatangani pada 10 Januari 2026 21:55:48



Ardhana Putra Wibowo
NIM 18219009

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

I.1 Latar Belakang

Monocular depth estimation (MDE), atau perkiraan kedalaman dari kamera tunggal, adalah salah satu teknik dalam bidang *computer vision* yang membantu memperkirakan jarak setiap piksel terhadap kamera. Teknik ini telah menjadi komponen penting dalam berbagai aplikasi dunia nyata, khususnya pada sistem otonom seperti robotika dan kendaraan tanpa pengemudi (Dong dkk. 2022). Kendaraan otonom harus mampu memahami kondisi lingkungan (*scene understanding*) di sekitarnya untuk melakukan pengambilan keputusan selama proses kemudi. *Scene understanding* tidak hanya penting bagi kendaraan itu sendiri, tetapi juga bagi para pengguna kendaraan tersebut. Pengguna akan lebih nyaman dan lebih mempercayai suatu kendaraan otonom apabila kendaraan otonom tersebut mampu memahami dan memvisualisasikan kondisi lalu lintas di sekitarnya (Dikmen & Burns 2017).

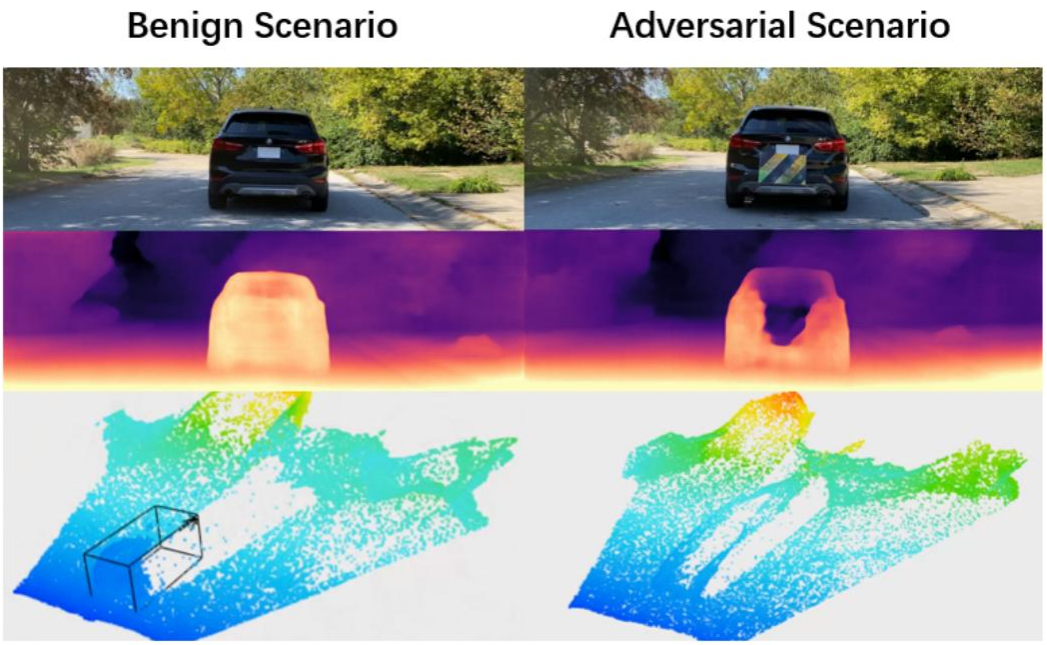
Deep neural network (DNN) yang merupakan bagian dari *deep learning* (pembelajaran mendalam), merupakan salah satu model komputasi yang banyak digunakan karena performanya yang tinggi pada proses inferensi terhadap pemrosesan data multidimensi, seperti gambar atau suara (LeCun 2015). Saat ini, telah banyak penelitian yang memanfaatkan DNN untuk meningkatkan performa dari MDE. Terdapat model MDE berbasis *convolutional neural network* seperti Monodepth2 (Godard dkk. 2019) dengan galat relatif mutlak sebesar 11,5%, serta Manydepth (Watson dkk. 2021) dengan galat relatif mutlak sebesar 9,3%. Terdapat juga MDE berbasis *transformer* seperti Depth Anything (Yang dkk. 2024) dengan galat relatif mutlak sebesar 8%.

Dalam industri kendaraan otonom, sejumlah perusahaan otomotif besar telah mengintegrasikan MDE sebagai bagian dari sistem persepsi kendaraan otonom. Perusahaan Tesla mengimplementasikan MDE pada sistem *Autopilot* sebagai alternatif yang dianggap lebih ekonomis dibandingkan penggunaan sensor jarak seperti radar dan LiDAR, karena teknologi berbasis kamera dinilai lebih ekonomis, memiliki biaya perangkat keras yang lebih rendah, serta lebih mudah diproduksi secara massal (Crowe 2019). Selain Tesla, perusahaan otomotif lain seperti Toyota juga menerapkan pendekatan serupa dalam pengembangan teknologi kendaraan otonom dengan mengeksplorasi penggunaan kamera untuk estimasi kedalaman dan pemahaman lingkungan (Certad dkk. 2022).

Akan tetapi, terdapat risiko keamanan dalam penggunaan MDE berbasis DNN tersebut. Penelitian yang dilakukan oleh Szegedy dkk. (2014) menunjukkan bahwa DNN dapat dikelabui oleh *adversarial example*, yaitu perubahan pada masukan citra yang tak kasat mata saat diobservasi oleh manusia namun dapat membuat model menghasilkan prediksi yang keliru. Model MDE berbasis DNN juga tidak luput dari serangan adversarial yang dapat menyebabkan estimasi kedalaman menyimpang jauh dari kondisi sebenarnya.

Salah satu *adversarial example* yang dapat direalisasikan dalam dunia nyata adalah *adversarial patch* (Brown dkk. 2018). Serangan ini terdiri dari sebuah *patch* (potongan gambar) yang bersifat tidak terikat pada citra, sehingga satu *patch* dapat digunakan pada berbagai kondisi dan adegan. Dalam lingkungan kendaraan otonom, *adversarial patch* dapat ditempelkan pada bagian belakang kendaraan lain yang tertangkap oleh kamera yang dipasang pada *ego vehicle* (kendaraan acuan). Gambar I.1 mengilustrasikan serangan tersebut.

Gambar I.1 memperlihatkan perbandingan antara skenario *benign* (tanpa serangan) dan skenario adversarial ketika *adversarial patch* ditempelkan pada bagian belakang kendaraan. *Patch* tersebut memicu model untuk mengeluarkan prediksi kedalaman yang keliru serta mengakibatkan rekonstruksi lingkungan pada kendaraan otonom menjadi terganggu.



Gambar I.1 Visualisasi serangan *adversarial patch* pada model *monocular depth estimation* (Cheng dkk. 2022)

Penelitian yang dilakukan oleh Cheng dkk. (2022) menunjukkan bahwa peletakan *adversarial patch* pada bagian belakang kendaraan, dengan ukuran hanya sekitar 10% dari luas permukaan belakang kendaraan, dapat mengelabui model MDE berbasis DNN seperti Monodepth2. Saat pengujian di dunia nyata, *patch* tersebut menyebabkan kesalahan estimasi jarak hingga 7 meter terhadap kendaraan yang diserang, meningkat drastis dibandingkan pada kondisi awal yang hanya memiliki kesalahan estimasi sekitar 0,5 meter. Guesmi dkk. (2023) melakukan pengujian serupa terhadap Monodepth2 dan menunjukkan bahwa serangan *adversarial patch* dapat meningkatkan galat relatif mutlak dari model tersebut menjadi 23%.

Kesalahan estimasi kedalaman berpotensi mengacaukan rekonstruksi lingkungan sekitar kendaraan otonom yang menjadi landasan bagi sejumlah fungsi penting dalam sistem kendaraan otonom, seperti deteksi rintangan (*obstacle detection*) dan perencanaan lintasan (*trajectory planning*) (Reda dkk. 2024). Gangguan pada fungsi deteksi rintangan dapat menyebabkan kendaraan otonom gagal mengenali objek pada jalur mengemudi, sehingga meningkatkan risiko tidak adanya respons

menghindar ataupun terjadinya pengereman yang terlambat. Sementara itu, gangguan pada fungsi perencanaan lintasan dapat menyebabkan kendaraan memilih jalur yang tidak aman ataupun menghasilkan keputusan manuver yang tidak sesuai dengan kondisi lingkungan.

Kegagalan pada fungsi-fungsi tersebut tidak hanya meningkatkan kemungkinan terjadinya kecelakaan, namun juga dapat menurunkan tingkat kepercayaan publik terhadap kendaraan otonom. Kepercayaan publik merupakan faktor utama dalam adopsi kendaraan otonom, karena teknologi ini hanya dapat diterapkan secara luas apabila pengguna meyakini bahwa sistem dalam kendaraan tersebut mampu mengambil keputusan secara aman (Nordhoff 2024). Penurunan kepercayaan publik akibat kegagalan sistem persepsi kendaraan otonom telah terlihat di Amerika Serikat pada kasus taksi otonom Cruise milik General Motors.

Pada bulan Oktober 2023, salah satu taksi otonom perusahaan tersebut menyeret seorang pejalan kaki hingga 6 meter, sehingga layanan operasional seluruh armada dihentikan dan menyebabkan perusahaan tersebut mengalami kerugian finansial hingga 2,7 miliar USD pada tahun yang sama (Hall & Shepardson 2024). Dampak kecelakaan ini tidak hanya dirasakan oleh Cruise, namun juga merambat ke ekosistem kendaraan otonom lainnya. Beberapa bulan kemudian, pada bulan Februari 2024, penurunan kepercayaan publik kembali terlihat dari kejadian pembakaran terhadap salah satu kendaraan otonom milik Waymo (Jankowicz 2024).

Selain penurunan kepercayaan publik, terdapat pula potensi celah *fraud* asuransi pada kendaraan otonom yang dapat merugikan produsen kendaraan otonom serta perusahaan asuransi (Luciano dkk. 2023). Risiko ini diperparah oleh regulasi serta tanggung jawab hukum yang belum jelas dalam kasus kecelakaan yang melibatkan kendaraan otonom (Sever & Contissa 2024). Seorang penyerang dapat merekayasa kecelakaan dengan memanipulasi lingkungan sekitar kendaraan melalui serangan adversarial, sehingga kecelakaan yang sebenarnya dipicu secara eksternal terlihat sebagai kegagalan internal sistem kendaraan otonom.

Oleh karena itu, dalam tugas akhir ini akan dilakukan evaluasi ketahanan model MDE terhadap serangan adversarial, khususnya *adversarial patch*, untuk mengukur kemampuan model dalam menghasilkan estimasi kedalaman selama serangan. Evaluasi ini penting dilakukan untuk memahami sejauh mana serangan tersebut dapat memengaruhi hasil estimasi kedalaman, mengidentifikasi kelemahan model dalam skenario serangan tertentu, serta memberikan informasi yang diperlukan untuk meminimalisir gangguan pada sistem persepsi kendaraan otonom.

I.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan pada subbab sebelumnya, rumusan masalah yang diperoleh adalah sebagai berikut:

1. Bagaimana tingkat ketahanan model *monocular depth estimation* berbasis *deep learning* terhadap *adversarial patch*?
2. Bagaimana karakteristik *adversarial patch* memengaruhi efektivitas serangan terhadap model *monocular depth estimation*?

I.3 Tujuan

Berdasarkan rumusan masalah yang telah dijabarkan pada subbab sebelumnya, terdapat tiga tujuan yang akan dicapai dalam tugas akhir ini. Tujuan-tujuan tersebut dijabarkan sebagai berikut:

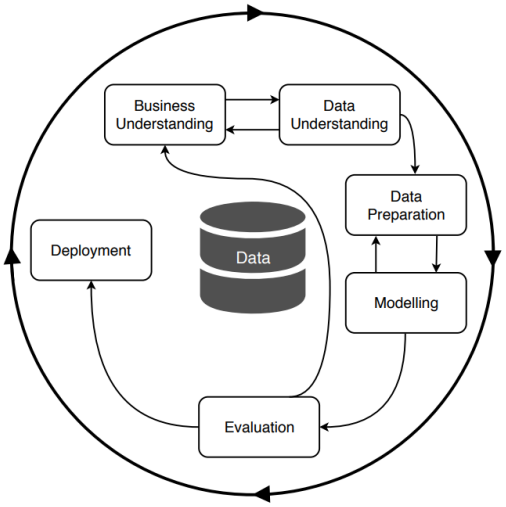
1. membangkitkan *adversarial patch* yang mampu mengganggu hasil estimasi kedalaman yang dihasilkan oleh model *monocular depth estimation*;
2. membandingkan tingkat ketahanan berbagai model *monocular depth estimation* terhadap serangan *adversarial patch* menggunakan metrik evaluasi yang relevan;
3. mengamati pengaruh karakteristik *adversarial patch* terhadap efektivitas serangan pada model *monocular depth estimation*.

I.4 Batasan Masalah

Berikut adalah batasan-batasan masalah yang digunakan selama pelaksanaan tugas akhir:

1. Pembangkitan *adversarial patch* ditujukan untuk dua model pralatih berbasis CNN (Monodepth2 dan Manydepth) dan satu model pralatih berbasis *transformer* (Depth Anything)
2. Serangan adversarial patch yang diujikan adalah serangan *adversarial patch* yang dibangkitkan melalui proses *backpropagation*.
3. *Adversarial patch* diujikan dalam lingkungan kendaraan otonom
4. *Adversarial patch* hanya dibangkitkan dan diujikan pada satu kategori kendaraan, yaitu mobil
5. Data latih dan data uji yang digunakan dalam pembangkitan dan evaluasi *patch* diambil dari dataset kendaraan otonom KITTI

I.5 Metodologi



Gambar I.2 Ilustrasi metodologi CRISP-DM (Martinez-Plumed dkk. 2021)

Metodologi yang digunakan dalam pelaksanaan tugas akhir ini adalah Cross-Industry Standard Process for Data Mining (CRISP-DM), sebuah metodologi yang banyak digunakan untuk melakukan proses dalam bidang *data mining*, *data*

science, dan pembelajaran mesin secara sistematis (Chapman dkk. 2000). Ilustrasi dari metodologi CRISP-DM dapat dilihat pada gambar I.2. CRISP-DM terdiri dari 6 fase utama yaitu *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment*. Akan tetapi, dalam tugas akhir ini, metodologi ini dimodifikasi untuk kebutuhan pembangkitan *adversarial patch* sehingga fase *deployment* tidak digunakan. Penjabaran lima fase CRISP-DM yang akan digunakan dalam tugas akhir ini adalah sebagai berikut:

1. *Business Understanding*

   Dalam tahap ini, dilakukan pemahaman masalah terkait kerentanan model *monocular depth estimation* (MDE) terhadap serangan adversarial, khususnya serangan berbasis *adversarial patch*. Tahap ini mencakup penelusuran terhadap model-model MDE *state of the art* yang umum digunakan dalam konteks persepsi kendaraan otonom, diikuti dengan analisis mengenai karakteristik dan dampak potensial serangan patch terhadap kualitas prediksi kedalaman. Penentuan tujuan penelitian berfokus pada pembangkitan *adversarial patch* dan perancangan metode evaluasi yang mampu mengukur tingkat ketahanan model-model MDE terhadap serangan tersebut, dengan memperhatikan kesenjangan antara penelitian terdahulu terhadap tugas akhir, yang diidentifikasi melalui studi literatur tentang *adversarial attack* pada MDE dan metode serangan adversarial berbasis *patch*. Studi literatur dilakukan dengan menelusuri *database* akademik seperti IEEE Xplore, ScienceDirect, arXiv, dan prosiding CVPR, dengan fokus pada kata kunci '*monocular depth estimation*', '*adversarial attack*', '*adversarial patch*', '*depth estimation evaluation metrics*', '*autonomous driving dataset*', dan penelitian-penelitian terkait.
2. *Data Understanding*

   Pada tahap ini, dilakukan proses eksplorasi dataset untuk memahami karakteristik data yang akan digunakan dalam tugas akhir. Eksplorasi ini mencakup pemeriksaan kelengkapan pasangan citra RGB dan *point cloud* LiDAR, peninjauan kualitas hasil proyeksi *point cloud* terhadap citra,

identifikasi anomali pada citra ataupun *point cloud*, serta peninjauan distribusi nilai kedalaman.

3. *Data Preparation*

   Pada tahap ini, dilakukan pengolahan pada data berdasarkan keterkaitannya dengan proyek, sebelum menjadi dataset akhir yang akan digunakan untuk tahap modeling dan evaluasi. Pengolahan tersebut mencakup pemilihan pasangan *ground truth* dari citra dan *point cloud* LiDAR yang valid, pembersihan data (jika diperlukan), penyelarasan citra dengan *point cloud*, normalisasi citra, penyesuaian ukuran citra, *formatting* pada anotasi, serta augmentasi pada citra.

4. *Modeling*

   Pada tahap ini, dilakukan pemilihan model MDE yang digunakan sebagai sasaran pembangkitan *adversarial patch*. Pemilihan model mempertimbangkan karakteristik arsitektur serta keterkaitannya dengan tujuan yang telah dirumuskan pada tahap *business understanding*. Tahap ini juga mencakup penetapan konfigurasi yang diperlukan dalam proses *modeling*, seperti pengaturan *hyperparameter*, pemilihan fungsi *loss*, dan pemantauan (*monitoring*) selama proses pembangkitan *patch*.

5. *Evaluation*

   Pada tahap ini, dilakukan evaluasi performa terhadap setiap model MDE ketika diuji pada kondisi *benign* (tanpa serangan) dan kondisi adversarial. Evaluasi ini bertujuan untuk menilai sejauh mana kualitas prediksi kedalaman mengalami penurunan setelah model melakukan inferensi terhadap citra yang telah disisipkan *adversarial patch* yang dibangkitkan pada tahap *modeling*. Hasil evaluasi diukur menggunakan metrik-metrik yang relevan untuk mengamati perubahan tertentu kualitas prediksi, dan digunakan untuk mengidentifikasi model yang menunjukkan ketahanan yang relatif lebih baik maupun yang lebih rentan terhadap serangan *adversarial patch*.

BAB II

STUDI LITERATUR

Bab II membahas tentang studi literatur yang berkaitan dengan permasalahan yang diangkat dalam tugas akhir ini. Pembahasan literatur mencakup konsep dasar *deep learning*, penggunaan *deep learning* dalam *monocular depth estimation*, metrik pengukuran terhadap prediksi kedalaman, serangan adversarial terhadap *deep neural network*, dan penelitian-penelitian terdahulu yang mendukung tugas akhir ini.

II.1 *Monocular Depth Estimation*

Monocular depth estimation (MDE), atau perkiraan kedalaman berbasis kamera tunggal, merupakan salah satu komponen penting dalam sistem persepsi kendaraan otonom. Metode ini memanfaatkan aliran citra dari kamera tunggal untuk memulihkan informasi kedalaman atau jarak setiap piksel pada citra terhadap kamera secara real-time. Walaupun sensor seperti radar dan LIDAR banyak digunakan dalam kendaraan otonom, perusahaan kendaraan otonom seperti Tesla sepenuhnya mengandalkan kamera untuk keperluan *scene understanding* pada fitur *full self driving* dalam kendaraan yang diproduksi. Pendekatan tersebut dilakukan karena biaya kamera yang lebih rendah dibandingkan perangkat keras lainnya, lebih banyak informasi semantik yang dapat diperoleh, serta membantu pemahaman adegan hanya dari satu kamera (Burns 2019).

Sebelum berkembangnya metode MDE berbasis *deep learning*, informasi mengenai jarak atau kedalaman diperoleh menggunakan kamera stereo. Kamera stereo adalah sebuah jenis kamera yang menggunakan dua lensa atau lebih untuk menangkap gambar dari sudut pandang yang berbeda. Pengambilan informasi jarak pada kamera stereo dapat diperoleh dengan memanfaatkan perbedaan (*disparity*) perspektif antara hasil tangkapan gambar pada setiap lensa kamera (Kok & Rajendran 2020).

Beberapa penelitian menunjukkan bahwa pengambilan informasi jarak dengan kamera stereo dapat menghasilkan prediksi dengan akurasi tinggi. Ginting, Patmasari, dan Aulia (2019) mengintegrasikan metode *computer vision* tradisional dengan kamera stereo pada sistem *embedded*, dan berhasil melakukan perkiraan pengukuran objek secara *real-time* dengan tingkat akurasi hingga 95%. Hamad, Khan, dan Mohamed (2024) juga melakukan hal serupa dan berhasil melakukan estimasi jarak objek secara *real-time* dengan hasil *mean absolute error* (MAE) yang rendah sebesar 2,6 cm terhadap jarak sebenarnya.

Akan tetapi, kamera stereo memiliki beberapa keterbatasan jika dibandingkan dengan *monocular depth estimation*, yaitu perlunya kalibrasi manual secara presisi pada kamera untuk setiap kondisi pengambilan gambar yang berbeda, perlunya *rig* khusus untuk mempertahankan kalibrasi pada kamera, serta kemungkinan terjadinya *blind spot*. Sebaliknya, MDE berbasis *deep learning* hanya memerlukan kamera tunggal tanpa bergantung pada rig khusus, serta mampu mengatasi masalah *blind spot* dengan memanfaatkan kemampuan *deep neural network* untuk mengisi area yang tidak memiliki informasi visual secara eksplisit (Tosi dkk. 2019).

Monocular Depth Estimation (MDE) memiliki peran penting dalam mendukung kebutuhan lebih lanjut pada kendaraan otonom (Wang dkk. 2020). MDE memungkinkan kendaraan otonom untuk melakukan deteksi objek sekeliling secara 3D (3D *lead object detection*), yang memungkinkan sistem mengenali dan memperkirakan posisi relatif kendaraan lain, pejalan kaki, serta rintangan (*obstacle*) di lingkungan sekelilingnya dalam ruang tiga dimensi. Informasi tersebut dapat direkonstruksi dan divisualisasikan kepada pengguna kendaraan otonom untuk meningkatkan kepercayaan pengguna (Dikmen & Burns 2017). Selain itu, informasi kedalaman yang dihasilkan juga dapat digunakan dalam perencanaan lintasan (*trajectory planning*), yaitu proses perancangan jalur optimal berdasarkan posisi objek dan rintangan yang terdeteksi. MDE juga berperan dalam penghindaran kecelakaan (*collision avoidance system*), dengan memungkinkan sistem untuk mengambil keputusan cepat seperti pengereman mendadak atau melakukan penghindaran terhadap *obstacle*.

Terdapat beberapa model DNN yang dikembangkan untuk kebutuhan pada MDE, baik dengan arsitektur *convolutional neural network* ataupun *transformer*. Model berbasis *convolutional neural network* (CNN) seperti Monodepthv2 yang dikembangkan oleh Godard dkk. (2019) mencapai *absolute relative error* (AbsRel) sebesar 0,115 pada dataset kendaraan otonom KITTI serta memiliki akurasi sebesar 87,7% pada *margin of error* 25% ($\delta < 1{,}25$). Model Manydepth yang dikembangkan oleh Watson dkk. (2021) merupakan pengembangan dari Monodepth2 dengan memanfaatkan informasi *multi-frame*, dan terbukti meningkatkan kualitas prediksi kedalaman terutama pada rangkaian video, dengan AbsRel 0,098 dan akurasi sebesar 90% pada *margin of error* 25%

Terdapat juga model berbasis *transformer* yang dikembangkan untuk kebutuhan MDE Depth Anything, yang dikembangkan oleh Yang dkk. (2024). Terdapat dua versi Depth Anything yang tersedia, yaitu Depth Anything Small (DA-Small) dengan basis ViT-Small dan Depth Anything Base (DA-Base) dengan basis ViT-Base. Saat diujikan pada dataset kendaraan otonom KITTI, Depth Anything memiliki performa yang lebih tinggi dibandingkan model berbasis CNN, dengan AbsRel 0,08 dan $\delta < 1{,}25$ sebesar 93,6% pada DA-Small, serta AbsRel serupa pada DA-Base dengan $\delta < 1{,}25$ sebesar 93,9%. Pembahasan lebih lanjut mengenai arsitektur yang membangun model tersebut akan dibahas pada subbab selanjutnya.

II.2 *Deep Learning*

Deep learning atau pembelajaran mendalam merupakan cabang dari pembelajaran mesin (*machine learning)* yang memanfaatkan *deep neural network* (jaringan saraf tiruan mendalam). Subbab ini diawali dengan pembahasan konsep dasar *machine learning* dan peran *neural network* sebagai landasan deep learning. Kemudian, akan dijelaskan cara kerja *deep neural network*, baik dengan arsitektur *convolutional neural network* maupun *transformer* dalam melakukan perkiraan kedalaman dari citra tunggal (*monocular depth estimation*).

II.2.1 *Machine Learning*

Pembelajaran mesin (*machine learning*) adalah salah satu ranah ilmu komputer yang membahas bagaimana komputer dapat melakukan suatu tugas yang tidak diprogram secara eksplisit. Berbeda dengan definisi algoritma secara umum memproses suatu masukan menjadi keluaran dengan aturan yang telah ditetapkan oleh pengguna komputer, *machine learning* mempelajari aturan tersebut yang menyebabkan suatu masukan menjadi keluaran (Heaton 2018). Terinspirasi dari sistem jaringan saraf pada makhluk hidup, *artificial neural network* (ANN) adalah salah satu metode pembelajaran mesin yang memodelkan sistem tersebut secara matematis (Heaton 2018). Seperti pada sistem jaringan saraf biologis, ANN terdiri atas sejumlah *neuron* yang memproses masukan dan mengirimkan hasil pemrosesan tersebut menuju *neuron* lainnya. Pemodelan matematika dari proses tersebut ditunjukkan pada persamaan II.1.

$$y = f(wx + b) \tag{II. 1}$$

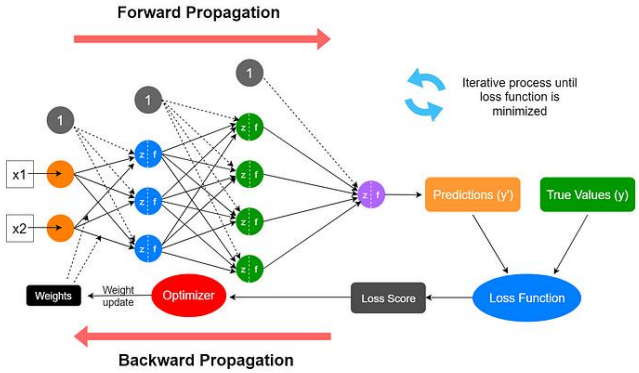
Pada persamaan II.1, variabel x merepresentasikan masukan , w merepresentasikan bobot (*weight*) yang mengatur seberapa besar pengaruh setiap masukan, dan b adalah bias yang memungkinkan fungsi aktivasi untuk dapat bergeser sehingga dapat mengenali pola-pola berbeda pada masukan. Variabel x, w, dan y kemudian diproses oleh fungsi aktivasi f, dan menghasilkan nilai keluaran y.

ANN terdiri atas *input layer* (lapisan masukan), satu atau lebih *hidden layer* (lapisan tersembunyi), dan *output layer* (lapisan keluaran). Berbeda dengan *input layer* dan *output layer*, *hidden layer* disebut demikian karena proses abstraksi yang terjadi pada *neuron* di dalam lapisan tersebut tidak dapat diamati secara langsung. Penggunaan ANN pada pembelajaran mendalam dibahas lebih lanjut pada subbab selanjutnya.

II.2.2 *Deep Neural Network*

Deep learning atau pembelajaran mendalam adalah salah satu ranah *machine learning* yang melibatkan pembangunan model komputasi yang tersusun oleh

banyak lapisan pemrosesan untuk dapat mempelajari representasi dari suatu data multidimensi, misalnya gambar, dengan tingkat abstraksi beragam (LeCun 2015). Berbeda dengan metode *machine learning* lainnya yang memerlukan hasil representasi data yang telah diolah terlebih dahulu, *deep learning* dapat menerima data mentah secara langsung sekaligus melakukan ekstraksi fitur. Penerapan *deep learning* dilakukan menggunakan *deep neural network* (DNN), yaitu *artificial neural network* (ANN) yang memiliki jumlah *hidden layer* lebih banyak daripada ANN pada umumnya, seperti yang terlihat pada gambar II.1.



Gambar II.1 Visualisasi *backpropagation* dan *forward propagation* pada DNN (Pramoditha 2022)

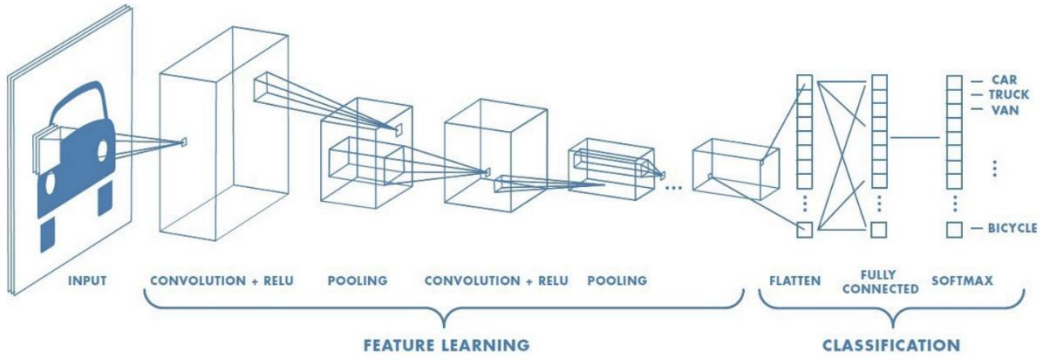
Seperti yang telah disebutkan pada persamaan II.1, suatu data masukan pada ANN akan diubah oleh neuron-neuron dalam ANN berdasarkan *weight* (bobot) dan bias yang terinisialisasi dalam setiap neuron. Proses *deep learning* dapat didefinisikan sebagai pembelajaran oleh DNN dalam meminimalkan *loss* dengan memperbaiki bobot dan bias pada neuron dengan proses *forward propagation* dan *backpropagation*. *Forward propagation* dapat didefinisikan sebagai proses saat data masukan melewati lapisan-lapisan dalam *deep neural network* untuk mengeluarkan sebuah prediksi.

Hasil keluaran tersebut lalu dibandingkan dengan *ground truth* untuk melihat penyimpangan antara hasil prediksi terhadap *ground truth* dari masukan tersebut. Penyimpangan ini direpresentasikan oleh nilai *loss* yang dihasilkan oleh *loss*

function. Nilai *loss* tersebut lalu diteruskan ke sebuah *optimizer* yang akan mengubah bobot dan bias dari setiap neuron yang ada dalam DNN tersebut. Proses ini disebut sebagai *backpropagation*.

II.2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis DNN yang dikhususkan untuk pemrosesan data berbentuk *grid*, misalnya citra, yang mana lokasi setiap pixel dalam citra tersebut dapat divisualisasikan dalam bentuk *grid* dua dimensi (Heaton 2018). CNN disebut demikian karena pada DNN, dilakukan operasi matematika berupa konvolusi yang dilakukan pada satu atau lebih lapisan pada DNN tersebut.



Gambar II.2 Arsitektur *convolutional neural network* (Saha 2018)

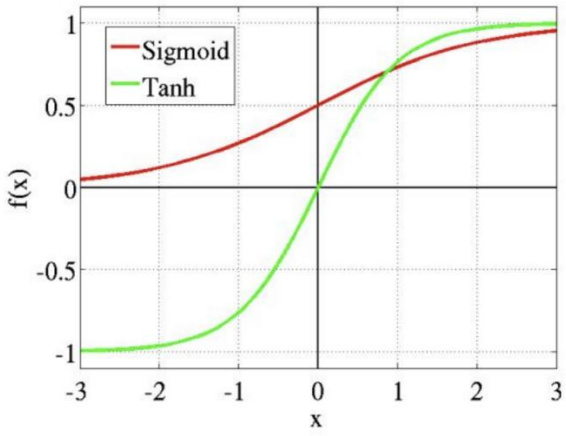
Convolutional Neural Network (CNN) terdiri dari beberapa jenis lapisan utama, yaitu lapisan konvolusi, lapisan *pooling*, dan lapisan *fully connected*. CNN secara umum banyak digunakan dalam tugas klasifikasi dan pendeteksian objek. Namun pada kebutuhan *monocular depth estimation* (MDE), arsitektur CNN yang digunakan adalah *fully convolutional network* (FCN), yang tidak memiliki lapisan *fully connected* pada lapisan akhir arsitektur.

Pada CNN, lapisan konvolusi bertujuan untuk mengekstraksi fitur pada gambar seperti tepi, garis, warna, dan ciri-ciri visual lainnya. Proses ekstraksi fitur ini dilakukan oleh filter dengan melakukan pemindaian terhadap gambar dengan penggeseran filter setiap langkah (*stride*) tertentu. Filter dapat diisi oleh beberapa

kernel dengan nilai berbeda. Pada setiap *stride*, dilakukan proses konvolusi antara filter dan setiap bagian dari *grid* gambar yang ditempati oleh filter. Proses ini akan menghasilkan *feature map* atau *activation map* yang menunjukkan keaktifan neuron terhadap ciri visual dari gambar. Berbeda dari ANN pada umumnya, *neuron* pada CNN bersifat tiga dimensi, yang terdiri dari lebar lapisan, tinggi lapisan, dan kedalaman dari filter pada lapisan yang akan mengekstraksi fitur dari gambar (O'Shea dan Nash 2015).

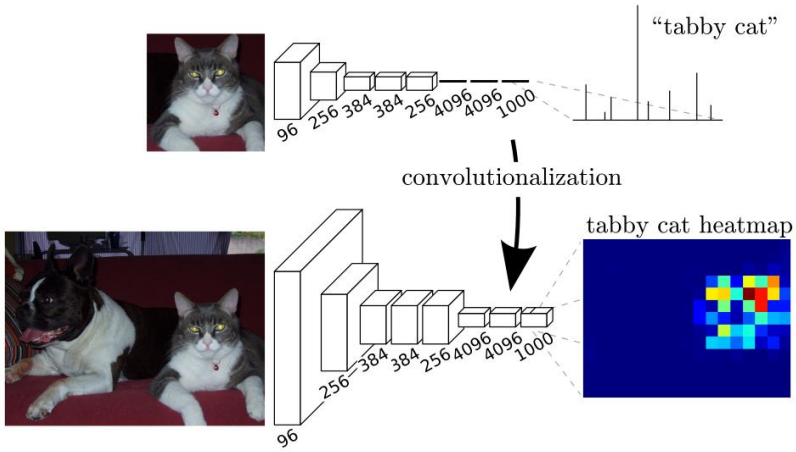
Kemudian, lapisan *pooling* akan mengurangi ukuran spasial dari *activation map* tersebut untuk mengurangi jumlah parameter dari model dan mempercepat proses komputasi (O'Shea dan Nash 2015). Operasi *pooling* dapat berupa *max pooling* atau *average pooling*. *Max pooling* mengembalikan nilai tertinggi dari bagian *activation map* yang ditempati oleh *kernel*, sedangkan *average pooling* mengembalikan rata-rata nilai dari bagian *activation map* yang ditempati oleh kernel.

Karena gambar merupakan data yang bersifat non-linier, diperlukan fungsi aktivasi secara non-linier pada CNN agar CNN dapat mempelajari pola pada gambar dengan lebih baik. Tanpa fungsi aktivasi non-linier, jaringan hanya akan melakukan transformasi linear antar lapisan, yang akan membuat CNN tersebut setara dengan satu lapisan konvolusi (Heaton 2018). Terdapat beberapa fungsi aktivasi non-linier yang dapat digunakan, yaitu *rectified linear unit* (ReLU), *softmax*, *sigmoid*, dan *tanh*. Seperti yang terlihat pada gambar II.2, digunakan beberapa fungsi aktivasi seperti ReLU dan *softmax*. Apabila masukan dimodelkan sebagai x, fungsi ReLU akan mengeluarkan x jika nilai x lebih dari nol, dan mengembalikan nol jika x kurang dari nol.



Gambar II.3 Perbandingan fungsi *sigmoid* dan tanh (Szandala 2021)

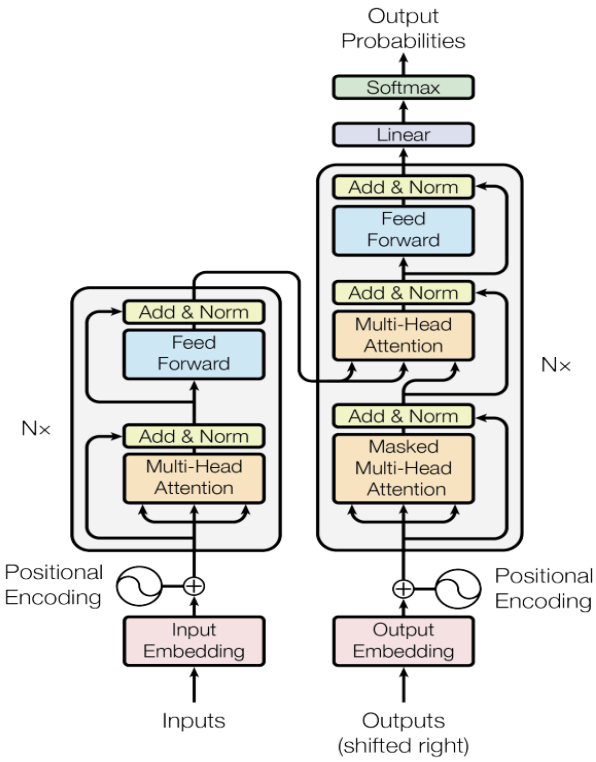
Terdapat fungsi *sigmoid* yang mentransformasi masukan x menjadi keluaran dengan nilai antara 0 hingga 1. Namun fungsi *sigmoid* memiliki masalah *vanishing gradient* (gradien semakin kecil) saat nilai x sangat kecil atau sangat besar yang menyebabkan terganggunya proses pelatihan CNN, khususnya pada *backpropagation* (Szandala 2021). Fungsi *tanh* (*hyperbolic tangent*) memiliki bentuk yang mirip dengan *sigmoid*, namun memiliki gradien yang lebih besar dibandingkan *sigmoid*, dan menghasilkan keluaran antara -1 hingga 1, seperti yang terlihat pada gambar II.3. Akan tetapi, ReLU tetap lebih banyak digunakan karena kecepatan komputasi yang lebih tinggi dibandingkan *tanh*.



Gambar II.4 Ilustrasi pengubahan lapisan *fully connected* dengan lapisan konvolusi tambahan (Shelhamer dkk. 2017)

Berbeda dengan CNN untuk klasifikasi yang diakhiri dengan lapisan *fully connected* dan fungsi *softmax* untuk menghasilkan keluaran, CNN untuk MDE menggunakan *fully convolutional network* (FCN). Pada FCN, seluruh jaringan hanya terdiri dari lapisan konvolusi dan lapisan *upsampling* tanpa lapisan *fully connected*, sehingga mampu menghasilkan keluaran dalam bentuk peta kedalaman (*depth map*) seperti yang terlihat pada gambar II.4. Beberapa model MDE yang tersedia secara publik seperti Monodepth2 dan Manydepth menggunakan arsitektur tersebut dalam melakukan estimasi kedalaman. Kedua model tersebut dilatih secara *self-supervised* dengan memanfaatkan rangkaian video dan dibantu oleh *pose network* berbasis ResNet18 untuk memprediksi pergerakan (*pose*) relatif kamera antar *frame* video. Hasil prediksi kedalaman kemudian digabungkan dengan prediksi *pose* untuk melakukan proyeksi ulang citra antar *frame* video yang berdekatan. Perbedaan antara citra asli dan hasil proyeksi citra tersebut didefinisikan sebagai *depth reprojection loss*, yang digunakan selama proses pelatihan kedua model tersebut (Godard dkk. 2019, Watson dkk. 2021).

II.2.4 *Transformer*



Gambar II.5 Arsitekur *transformer* (Vaswani dkk. 2023)

Transformer adalah salah satu arsitektur DNN yang memanfaatkan mekanisme *attention* untuk memperoleh keterkaitan antar elemen masukan secara paralel, tanpa perlu melakukan pemrosesan berulang seperti pada *recurrent neural network* (Vaswani dkk. 2023). Sifat ini membuat *transformer* sering digunakan dalam pemrosesan bahasa alami karena keunggulannya dalam mengenali hubungan antar *token* baik secara global (misalnya mempelajari aspek gramatikal dan menangani kata yang sama dalam konteks yang berbeda), ataupun dalam satu rangkaian (misalnya urutan kata dalam sebuah kalimat).

Transformer terdiri atas dua bagian utama, yaitu *encoder* dan *decoder*. Pada gambar II.5, *encoder* ditunjukkan pada bagian kiri arsitektur, sedangkan *decoder* ditunjukkan pada bagian kanan. Pada bagian *encoder*, akan dilakukan proses *input embedding* pada *token* masukan untuk mengubah token masukan menjadi sebuah vektor *embedding* dengan ukuran tetap. Kemudian, dilakukan proses *positional*

encoding dengan melakukan penomoran pada setiap *token* dalam vektor *embedding*.

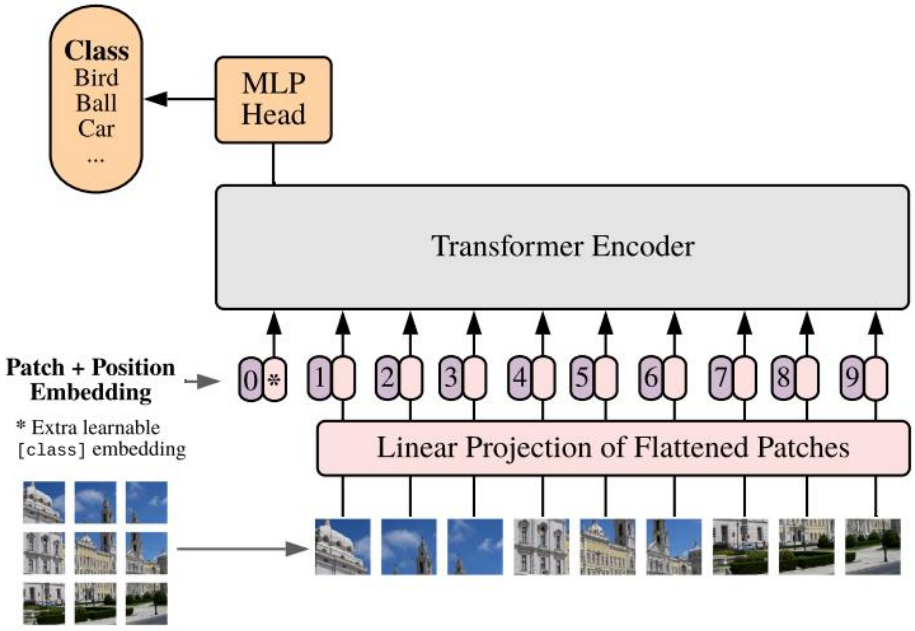
Setelah diberikan *positional encoding*, vektor tersebut dimasukkan ke dalam modul *multi-head attention* (MHA). MHA memproyeksikan vektor ke dalam beberapa *head* sehingga memungkinkan mekanisme *attention* untuk dilakukan secara paralel untuk memungkinkan model mempelajari berbagai jenis relasi antar token secara bersamaan.

Mekanisme *attention* dapat dideskripsikan sebagai proses perbandingan setiap token dengan token lainnya untuk menentukan besar pengaruh antar token. Setiap token pada vektor *embedding* diubah menjadi sebuah *query* yang dicocokkan dengan himpunan pasangan *key*-*value* pada token lainnya untuk menghasilkan sebuah keluaran. *Self attention* melakukan hal serupa dengan membandingkan sebuah token terhadap token itu sendiri (Vaswani dkk. 2023). Proses *attention* dapat dijelaskan seperti pada persamaan II.2, dengan Q merepresentasikan *query*, K merepresentasikan *key*, V merepresentasikan *value*, dan d_k merepresentasikan dimensi dari *key* pada vektor.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad \text{(II. 2)}$$

Arsitektur transformer yang awalnya dikembangkan untuk pemrosesan bahasa alami kemudian diadaptasi ke domain *computer vision* oleh Dosovitskiy dkk. (2021) dengan pengembangan *Vision Transformer* (ViT). Pada ViT, sebuah citra dua dimensi dibagi menjadi beberapa potongan kecil citra (*image patch*) berukuran tetap (misalnya 16×16 piksel). Setiap potongan kecil tersebut kemudian di-*flatten* dan diproyeksikan melalui lapisan linier (setara dengan lapisan *fully connected* pada CNN) sehingga menghasilkan vektor *embedding*. Informasi posisi spasial antar *patch* juga terjaga dengan dilakukannya penambahan *positional encoding* pada setiap potongan citra. Proses ini memungkinkan model *transformer* mengenali keterurutan spasial antar potongan citra, yang berperan penting bagi model untuk memahami struktur citra secara menyeluruh.

Pada tahap *positional* encoding, juga dapat ditambahkan *class embedding* (atau *readout token*) yang dapat dipelajari oleh model apabila ViT akan digunakan untuk kebutuhan klasifikasi. Representasi dari *class embedding* tersebut akan digunakan oleh bagian *head* dari *multi-layer perceptron* (MLP) untuk menghasilkan prediksi kelas. Seluruh rangkaian potongan citra yang telah ditambahkan dengan *positional encoding* kemudian dimasukkan ke dalam *encoder* pada arsitektur *transformer* untuk diproses lebih lanjut melalui mekanisme *multi-head self-attention*. Hasil keluaran dari *encoder* kemudian diubah menjadi prediksi kelas melalui bagian *head* dari MLP. Ilustrasi dari arsitektur ViT dapat dilihat pada gambar II.6.



Gambar II.6 Arsitektur *vision transformer* (Dosovitsky dkk. 2021)

Model MDE Depth Anything memanfaatkan *backbone* DINOv2 berbasis ViT sebagai *encoder* untuk mengekstraksi fitur dari citra masukan (Yang dkk. 2024). Model ini tidak meneruskan hasil keluaran dari *encoder* ke bagian *head* dari MLP, namun meneruskannya ke *dense prediction transformer* (DPT). DPT menyusun kembali *vektor embedding* hasil keluaran dari *encoder* sesuai dengan *positional encoding* sehingga membentuk sebuah representasi spasial (data dengan resolusi seperti gambar). Representasi tersebut kemudian diubah menjadi beberapa *feature*

map dengan resolusi beragam. Seluruh *feature map* tersebut kemudian dipadukan dengan mekanisme *upsampling* untuk menghasilkan prediksi akhir peta kedalaman (Ranftl dkk. 2021).

II.3 Metrik Evaluasi Estimasi Kedalaman

Untuk mengevaluasi kualitas dari *depth map* (peta kedalaman) yang dihasilkan oleh model MDE berbasis *deep learning*, digunakan beberapa metrik yang telah banyak digunakan pada penelitian MDE sebelumnya, seperti *absolute relative error* (AbsRel) atau galat relatif mutlak, *root mean square error* (RMSE), dan *threshold accuracy* (Eigen dkk. 2014). AbsRel mengukur rata-rata kesalahan relatif antara prediksi dan nilai sebenarnya dengan membandingkan proporsi antara selisih hasil prediksi dan nilai *ground truth* terhadap hasil *ground truth* dari kedalaman. Representasi matematis dari AbsRel ditunjukkan pada persamaan II.3, dengan N menyatakan total piksel pada *depth map*, i menyatakan prediksi kedalaman pada piksel ke-i, d_i menyatakan kedalaman sebenarnya, dan d'_i menyatakan kedalaman yang diprediksi oleh model MDE.

$$AbsRel = \frac{1}{N}\sum_{i=1}^{N}\frac{|d_i - d'_i|}{d_i} \qquad \text{(II. 3)}$$

RMSE menunjukkan rata-rata besar penyimpangan antara nilai prediksi dengan nilai kedalaman sebenarnya. Metrik ini sangat sensitif terhadap kesalahan besar (*outlier*). Semakin kecil nilai RMSE, semakin kecil penyimpangan hasil prediksi terhadap nilai kedalaman sebenarnya. Nilai RMSE yang tinggi mengindikasikan bahwa terdapat beberapa prediksi dengan dengan tingkat penyimpangan yang besar. Persamaan dari RMSE ditunjukkan pada persamaan II.4, dengan N menyatakan total piksel pada *depth map*, i menyatakan prediksi kedalaman pada piksel ke-i, d_i menyatakan kedalaman sebenarnya, dan d'_i menyatakan kedalaman yang diprediksi oleh model MDE.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - d'_i)^2} \quad \text{(II. 4)}$$

Threshold accuracy menunjukkan persentase piksel yang memiliki rasio prediksi kedalaman yang cukup dekat terhadap nilai sebenarnya dalam ambang batas tertentu. Persamaan dari *threshold accuracy* direpresentasikan pada persamaan II.5. Pada persamaan tersebut, δ (delta) merepresentasikan *threshold* accuracy, N menyatakan total piksel pada *depth map*, i menyatakan prediksi kedalaman pada piksel ke-i, d_i menyatakan kedalaman sebenarnya, d'_i menyatakan kedalaman yang diprediksi oleh model MDE, dan τ (tau) menunjukkan *threshold* atau batas kedekatan antara hasil prediksi kedalaman dengan nilai kedalaman sebenarnya. Terdapat tiga batas kedekatan yang biasanya digunakan dalam evaluasi model MDE, yaitu 1,25, $1{,}25^2$, dan $1{,}25^3$.

$$\delta = \frac{1}{N}\sum_{i=1}^{N} \max\left(\frac{d'_i}{d_i}, \frac{d_i}{d'_i}\right) < \tau \quad \text{(II. 5)}$$

II.4 Serangan Adversarial

Meskipun model *deep neural network* (DNN) telah banyak diaplikasikan dalam berbagai skenario di kehidupan nyata, Szegedy dkk. (2013) menemukan bahwa DNN rentan terhadap *adversarial examples*. *Adversarial examples* adalah sebuah sampel citra yang telah diberi gangguan (perturbasi) kecil yang tidak terlihat oleh manusia, namun dapat membuat model DNN salah dalam melakukan inferensi pada citra tersebut. *Adversarial example* disisipkan pada model oleh *adversary* (penyerang) saat model telah dipublikasikan. Kerentanan DNN terhadap serangan ini disebabkan oleh sifat dari DNN yang linier walaupun memiliki kompleksitas tinggi, yang dicontohkan dalam penggunaan fungsi aktivasi linier seperti ReLU (Goodfellow dkk. 2014). Secara matematis, persamaan dari *adversarial examples* direpresentasikan pada persamaan II.6, yang menggambarkan himpunan

adversarial example ($\Delta(x)$) dengan batas jarak *euclidean* sebesar ε antara data asal (x) dengan setiap *adversarial example* (x').

$$\Delta(x) = \{x' : \|x' - x\| \leq \varepsilon\} \quad \text{(II. 6)}$$

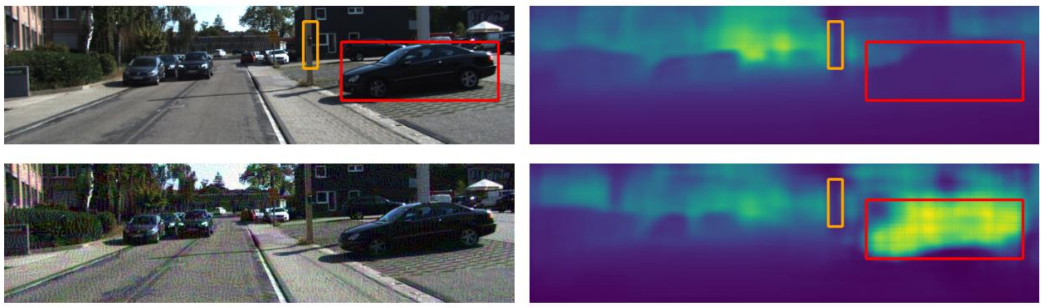
Berdasarkan tingkat pengetahuan penyerang terhadap model, serangan adversarial dapat dikategorikan menjadi dua jenis, yaitu serangan *white-box* dan serangan *black-box* (Papernot dkk. 2015). Pada serangan *white-box*, penyerang mengetahui dan memiliki akses secara penuh terhadap model yang diserang, termasuk data latih, arsitektur model yang digunakan, jumlah lapisan, fungsi aktivasi, *hyperparameter,* dan weights yang digunakan model.

Sebaliknya pada serangan *black-box*, penyerang hanya dapat mengamati keluaran dari model dan tidak mengetahui ataupun tidak memiliki akses langsung terhadap model DNN. Terdapat dua metode yang dapat dilakukan oleh penyerang untuk melakukan serangan secara *black*-box, yaitu dengan membangun model tiruan dan memanfaatkan *transferability* dari *adversarial sample*. Pada metode pertama, penyerang dalam serangan ini dapat melakukan penyerangan dengan teknik *model inversion* atau membangun model tiruan (*surrogate model*) dengan cara melakukan *query* terhadap API dari model tujuan dan melakukan perkiraan terhadap model tersebut. Sedangkan pada metode kedua, penyerang melakukan serangan secara *white-box* terhadap model subtitusi untuk menghasilkan *adversarial sample*, dan menggunakan sampel tersebut untuk menyerang model tujuan.

Berdasarkan hasil keluaran, terdapat empat tujuan serangan adversarial yang dapat digunakan, yaitu *confidence reduction, misclassifcation, targeted misclassification*, dan *source/target misclassification* (Papernot dkk. 2015). Pada *confidence reduction*, serangan yang dilakukan menurunkan tingkat *confidence* dari terhadap keluaran hasil prediksi. Serangan dengan tujuan *misclassification* mengubah keluaran kelas model menjadi kelas lainnya tanpa menentukan kelas tertentu yang menjadi target. Serangan dengan tujuan *targeted misclassification* mengubah keluaran kelas model menjadi kelas lainnya yang telah ditetapkan sebagai target. Serangan dengan tujuan *source/target misclassification* mengubah suatu kelas asal

menjadi kelas lain yang telah ditetapkan sebagai target. Akan tetapi, keluaran yang dihasilkan oleh prediksi model MDE bukan kelas, melainkan peta kedalaman. Konsep setara yang digunakan dalam melakukan serangan adversarial terhadap model MDE adalah *targeted misprediction*, yang memaksa model mengeluarkan prediksi jarak yang mendekati target nilai tertentu (Yamanaka dkk. 2020).

Berdasarkan domain serangan, serangan adversarial dapat dibedakan menjadi dua jenis, yaitu serangan secara digital dan serangan secara fisik. Pada serangan dalam domain digital, gangguan (*perturbation*) disisipkan langsung pada citra digital yang menjadi masukan model, seperti yang terlihat pada gambar II.7. Pada gambar tersebut, serangan FGSM diterapkan pada citra untuk menyesatkan prediksi kedalaman pada objek tertentu, seperti mobil hitam yang diberi kotak merah. Kolom kiri menampilkan citra sebenarnya, sedangkan kolom kanan menampilkan hasil prediksi kedalaman. Baris pertama menunjukkan citra asli, sementara baris kedua menunjukkan citra *adversarial*. Pada hasil prediksi kedalaman, warna kuning terang merepresentasikan nilai kedalaman yang besar (objek lebih jauh), sedangkan warna biru gelap menunjukkan nilai kedalaman yang kecil (objek lebih dekat).



Gambar II.7 Serangan FGSM pada model *monocular depth estimation* (Zhang dkk. 2020)

Beberapa metode umum yang dapat digunakan untuk menghasilkan serangan pada domain digital meliputi *fast gradient sign method* (FGSM) dan *projected gradient descent* (PGD). Pada metode FGSM, serangan adversarial dilakukan dengan menambahkan gangguan kecil pada masukan yang dihitung dari gradien fungsi *loss* terhadap masukan (Goodfellow dkk. 2014). Persamaan FGSM dapat dilihat pada

persamaan II.7, dengan x adalah masukan, x' adalah *adversarial example*, y adalah *ground truth*, ε menunjukkan besar gangguan, J menunjukkan fungsi *loss*, dan θ merepresentasikan parameter dari model.

$$x' = x + \varepsilon \cdot sign(\nabla_x J(\theta, x, y)) \quad \text{(II. 7)}$$

Pada PGD, serangan FGSM diperluas dengan melakukan serangan tersebut secara iteratif sekaligus menerapkan pembatasan jarak *euclidean* pada setiap iterasi. Pembatasan jarak *euclidean* dilakukan antara hasil gangguan oleh PGD terhadap himpunan *adversarial example* yang telah dituliskan pada persamaan II.6 untuk memastikan bahwa gangguan yang diberikan tidak melampaui ε (Madry dkk. 2017). Jika tidak dilakukan pembatasan, hasil serangan PGD berpotensi menyimpang terlalu jauh dari masukan sebenarnya, sehingga tidak lagi memenuhi definisi dari *adversarial example*. Persamaan PGD dapat dilihat pada persamaan II.8, dengan x adalah masukan, x_t merupakan *adversarial example* pada iterasi ke-t, y adalah *ground truth*, α menunjukkan besar gangguan, J menunjukkan fungsi *loss*, dan θ merepresentasikan parameter dari model.

$$x_{t+1} = \arg\min_{c \in \Delta x} \left\| c - (x_t + \alpha \cdot sign(\nabla_x J(\theta, x, y))) \right\| \quad \text{(II. 8)}$$

Berbeda dengan serangan pada domain digital, serangan pada domain fisik dilakukan dengan mencetak *adversarial example* tersebut dan meletakkannya pada objek di dunia nyata. Sebagai contoh, serangan di domain digital terjadi saat sebuah gambar diberikan *adversarial example*. Apabila *adversarial example* tersebut dicetak dan diletakkan pada suatu objek fisik di dunia nyata, serangan tersebut akan dikelompokkan dalam kategori serangan adversarial pada domain fisik.

Gambar II.8 Serangan *adversarial patch* terhadap *monocular depth estimation* (Guesmi dkk. 2023)

Salah satu pendekatan untuk menghasilkan *adversarial example* pada domain fisik adalah *adversarial patch* (Brown dkk. 2018). Serangan dengan *adversarial patch* dilakukan dengan menghasilkan potongan gambar yang dilakukan *perturbation* untuk ditempelkan pada objek fisik di dunia nyata dengan tujuan mengelabui model DNN. Model MDE tidak luput dari serangan *adversarial patch*, seperti yang terlihat pada gambar II.6.

Jika dibandingkan dengan serangan adversarial pada domain digital seperti PGD dan FGSM, perubahan citra akibat *adversarial patch* lebih mudah dikenali oleh manusia. Meskipun demikian, efektivitas serangan oleh *patch* lebih unggul dibandingkan serangan PGD maupun FGSM, karena serangan ini dapat diterapkan pada berbagai citra dan skenario tanpa perlu menghasilkan gangguan yang spesifik untuk setiap masukan citra, sebagaimana yang diperlukan oleh PGD dan FGSM. Selain itu, *adversarial patch* juga memiliki aspek *transferability*, yaitu kemampuan

patch untuk mengelabui beberapa model berbeda walaupun pembangkitan *patch* tersebut hanya dilakukan terhadap satu model tertentu (Brown dkk. 2018).

Kerentanan DNN akibat *adversarial patch* berpotensi membahayakan sistem berbasis DNN yang membutuhkan tingkat keamanan dan keandalan tinggi, seperti kendaraan otonom. Lu dkk. (2017) berpendapat bahwa serangan adversarial secara fisik terhadap kendaraan otonom bersifat tidak realistis karena efektivitas gangguan sangat bergantung pada jarak dan sudut pandang kamera yang berubah-ubah pada setiap pergerakan kendaraan. Akan tetapi, penelitian lebih lanjut oleh Athalye dkk. (2018) menunjukkan bahwa sebuah *adversarial example* fisik yang diberikan augmentasi seperti penambahan *noise*, kecerahan, distorsi, rotasi, translasi, dan perubahan ukuran, dapat menghasilkan *adversarial example* yang tahan terhadap perubahan kondisi dalam dunia nyata dan tetap dapat mengelabui model DNN. Rangkuman mengenai subbab ini dapat dilihat pada tabel II.1.

Tabel II.1 Rangkuman serangan adversarial

Aspek	**Kategori**	**Penjelasan**
Jenis Serangan	*White-box*	Penyerang memiliki akses penuh terhadap seluruh detail model, termasuk arsitektur, jumlah lapisan, fungsi aktivasi, hyperparameter, weights, dan data latih.
	Black-box	Penyerang tidak memiliki akses internal model dan hanya dapat mengamati keluaran model. Penyerang tidak mengetahui arsitektur, bobot, atau data latih dari model. Terdapat dua metode serangan dalam skenario ini, yaitu: 1. *Surrogate model/model inversion*, yaitu membangun model tiruan melalui *querying* ke API model yang ingin diserang. 2. *Transferability*, yaitu membangkitkan *adversarial*

Aspek	Kategori	Penjelasan
		sample secara *white-box* dari model substitusi dan menggunakannya untuk menyerang model tujuan.
Tujuan Serangan	*Confidence reduction*	Menurunkan tingkat *confidence* dari suatu prediksi
	Misclassification	Mengubah keluaran prediksi kelas suatu model menjadi kelas lain secara acak (tidak ditargetkan)
	Targeted misclassification	Mengubah keluaran prediksi kelas suatu model menjadi kelas lain yang telah ditentukan
	Source/target misclassification	Mengubah keluaran prediksi dengan kelas asal tertentu menjadi kelas tujuan yang telah ditentukan
	Targeted misprediction	Pada model *monocular depth estimation*, keluaran *depth map* dari model diarahkan menjadi nilai kedalaman yang telah ditentukan
Domain Serangan	Digital	Serangan disisipkan langsung pada citra digital. Contoh serangan pada domain digital yaitu PGD dan FGSM. Pada domain ini, serangan biasanya tidak terlihat oleh manusia.
	Fisik	Serangan dicetak dan ditempelkan pada objek di dunia nyata. Contoh serangan pada domain fisik yaitu *adversarial patch*. *Adversarial patch* lebih terlihat oleh manusia, namun sangat efektif dan memiliki *transferability* tinggi pada beragam kondisi citra.

II.5 Optimasi *Adversarial Patch* pada Domain Fisik

Dalam menerapkan serangan adversarial dalam sistem pada dunia nyata seperti kendaraan otonom, ancaman yang realistis tidak lagi mengasumsikan bahwa penyerang memiliki akses secara *white-box*. Sebaliknya, model ancaman (*threat model*) yang lebih kuat dan sesuai dengan kondisi nyata berasumsi bahwa penyerang hanya dapat memodifikasi elemen pada lingkungan fisik, misalnya dengan menempelkan *patch* pada objek yang terlihat oleh kamera.

Threat model ini pertama kali dieksplorasi oleh Eykholt dkk. (2018), yang menunjukkan bahwa *adversarial example* dalam bentuk fisik tetap dapat menyesatkan model *traffic sign recognition* pada kendaraan otonom dengan tingkat *transferability* tinggi, bahkan setelah *patch* tersebut dicetak dan dipotret ulang dalam berbagai kondisi pencahayaan dan sudut pengambilan gambar.

Untuk menghasilkan *adversarial patch* dengan serangan yang efektif terhadap variasi kondisi dalam dunia nyata, berbagai teknik optimasi telah dikembangkan. Salah satu pendekatan untuk meningkatkan keandalan serangan *patch* adalah *Expectation over Transformation* (EoT), yang melibatkan augmentasi *patch* terhadap berbagai perubahan kondisi fisik yang mungkin terjadi di dunia nyata, seperti rotasi, perubahan skala, pencahayaan, hingga *noise* pada kamera selama proses pelatihan *patch* (Athalye dkk. 2018).

Rentang augmentasi tersebut ditetapkan secara manual sebelum pembangkitan *patch* dan tidak menjadi variabel yang dioptimalkan oleh *optimizer*. Secara matematis, EoT dapat direpresentasikan pada persamaan II.9, dengan p adalah *patch*, x adalah citra masukan, X adalah data latih yang digunakan untuk pengambilan sampel citra, t adalah augmentasi, T adalah himpunan augmentasi yang dapat dipilih, θ merepresentasikan parameter dari model, fungsi A merepresentasikan proses peletakan *patch* p pada citra x setelah proses augmentasi t, dan y adalah *ground truth*.

$$\arg\max_{p} \mathrm{E}_{x\sim X, t\sim T}\, J(\theta, A(x, p, t), y) \qquad \text{(II. 9)}$$

Gambar II.9 Perbedaan persebaran warna antara citra digital dengan hasil cetak (Eykholt dkk. 2018)

Selain augmentasi pada *patch*, keandalan *adversarial patch* pada domain fisik juga bergantung pada kemampuan *patch* tersebut untuk dapat dicetak secara akurat oleh *printer*. Ukuran tersebut dapat diukur dengan *non-printability score* (NPS), yang menghitung seberapa besar perbedaan sebaran warna antara *patch* pada domain *digital*, dengan himpunan warna yang dapat dicetak oleh *printer* (Sharif dkk. 2016). Warna-warna yang sulit direproduksi oleh *printer* dapat menyebabkan perbedaan hasil cetak, sehingga menurunkan efektivitas serangan *patch*.

Persamaan NPS direpresentasikan oleh persamaan II.10, dengan P merupakan himpunan piksel dalam *patch*, dan C merupakan himpunan warna yang dapat dicetak oleh *printer*. Semakin rendah nilai NPS, warna *patch* yang dihasilkan pada domain *digital* semakin mendekati warna-warna yang dapat dicetak oleh *printer*, sehingga mengurangi risiko menurunnya tingkat keberhasilan serangan dari *patch*. Berbeda dengan EoT, nilai NPS dari *patch* merupakan parameter dicari yang dioptimalkan langsung oleh *optimizer*. Nilai tersebut diperoleh dengan mengikutsertakan NPS pada *loss function* selama proses pembangkitan *patch*.

$$NPS(P) = \sum_{p \in P} \min_{c \in C} |p - c| \qquad \text{(II. 10)}$$

Eykholt dkk. (2018) menunjukkan bahwa *patch* dengan piksel yang terlalu tajam cenderung lebih mudah dikenali oleh manusia dan lebih sensitif terhadap perubahan jarak dan sudut pandang kamera, sehingga dapat menurunkan *attack success rate* serangan dalam skenario penyerangan kendaraan otonom. Untuk meningkatkan kealamian dan mengurangi ketajaman dari *patch,* digunakan teknik *total variation* selama proses optimasi *patch* untuk memastikan *smoothness* dari *patch* yang dihasilkan. Teknik ini mendorong keselarasan antar piksel dalam *patch*, sehingga menghasilkan tampilan visual yang tidak mencolok dan menyerupai pola alami pada objek fisik.

Total variation ditunjukkan pada persamaan II.11, dengan P merepresentasikan himpunan piksel pada *patch*, dan $p_{i,\,j}$ menunjukkan nilai piksel dari *patch* pada koordinat (i, j). Nilai *total variation* akan bernilai rendah apabila piksel-piksel yang berdekatan memiliki nilai yang serupa, dan akan bernilai tinggi jika terdapat perbedaan tajam antar piksel yang berdekatan. Serupa dengan NPS, tingkat keselarasan antar piksel diminimalkan langsung oleh *optimizer* dengan mengikutsertakan persamaan *total variation* selama proses pembangkitan *patch*. Rangkuman dari subbab ini dapat dilihat pada tabel II.2.

$$TV(P) = \sum_{i,j} \sqrt{\left(\left(p_{i,j} - p_{i+1,j}\right)^2 + \left(p_{i,j} - p_{i,j+1}\right)^2\right)} \qquad \text{(II. 11)}$$

Selain keselarasan antar piksel, luminansi juga berperan penting dalam menentukan tingkat keterdeteksian *patch*. Dalam persepsi visual manusia, luminansi menjadi patokan (*cue*) utama dalam proses pendeteksian objek (White dkk. 2017). Objek yang memiliki luminansi yang lebih tinggi cenderung lebih mudah terdeteksi oleh manusia, begitu pula sebaliknya, terlepas dari kondisi lingkungan di sekitarnya. Intensitas warna (*saturation*) berperan sebagai patokan sekunder dalam menentukan keterdeteksian suatu objek, sedangkan warna objek (*hue*) tidak menunjukkan pengaruh apapun terhadap keterdeteksian objek.

Tabel II.2 Rangkuman optimasi *adversarial patch* pada domain fisik

No	Istilah	Keterangan
1	*Expectation over Transformation* (EoT)	Metode optimasi *patch* dengan melakukan augmentasi fisik (rotasi, skala, translasi, pencahayaan, *noise* kamera) pada *patch*. Optimasi pada *patch* dilakukan dengan mendefinisikan rentang augmentasi (misalnya -10% hingga +10%) sebelum dilakukan proses pembangkitan *patch*.
2	*Non-Printability Score* (NPS)	Metrik yang mengukur perbedaan sebaran warna antara *patch* pada domain digital dengan himpunan warna yang dapat dicetak oleh *printer*. Optimasi pada *patch* dilakukan dengan menyertakan metrik ini pada *loss function* selama proses pembangkitan *patch*.
3	*Total Variation*	Metrik yang mengukur keselarasan warna antar piksel pada *patch*. Optimasi pada *patch* dilakukan dengan menyertakan metrik ini pada *loss function* selama proses pembangkitan *patch*.

II.6 Penelitian Terkait

Dalam kondisi normal, model MDE mampu menghasilkan prediksi kedalaman yang cukup akurat untuk kebutuhan persepsi pada kendaraan otonom. Evaluasi pada skenario dunia nyata menunjukkan bahwa kesalahan estimasi kedalaman pada kondisi *benign* berada pada kisaran 0,52-0,77 meter, dengan akurasi pendeteksian objek tiga dimensi mencapai 90,7% (Cheng dkk. 2022).

Tetapi, ketika model MDE mengalami serangan adversarial, terjadi peningkatan kesalahan estimasi kedalaman secara drastis. Cheng dkk. (2022) menunjukkan bahwa *patch* fisik yang ditempelkan pada kendaraan dapat menghasilkan rata-rata kesalahan estimasi hingga 6 meter, serta menurunkan akurasi deteksi objek tiga dimensi dari 90,7% menjadi 5,16%. Kondisi ini berpotensi menyebabkan

terganggunya sistem persepsi pada kendaraan otonom, seperti pendeteksian objek, *collision avoidance*, dan *lane assist*, yang sangat bergantung pada akurasi infromasi kedalaman.

Terdapat beberapa penelitian yang telah mengkaji serangan adversarial terhadap model MDE dalam konteks kendaran otonom. Metode-metode yang dilakukan dalam penelitian tersebut mencakup penempatan objek *benign* namun dapat mengganggu proses estimasi jarak, pembangkitan serangan digital dengan PGD dan FGSM, serta pembangkitan serangan fisik dengan *adversarial patch*. Penelitian-penelitian terkait yang membahas metode penyerangan tersebut dapat dilihat pada tabel II.3. Pada tabel tersebut, dijabarkan judul penelitian, metode, serta hasil kesalahan yang ditimbulkan oleh model MDE.

Dalam tugas akhir ini, akan ada beberapa peningkatan kontribusi dibandingkan penelitian-penelitian terdahulu. Pertama, sebagian besar penelitian terkait hanya mengevaluasi model berbasis CNN (Zheng dkk. 2024, Zhang dkk. 2020), sementara penelitian yang melibatkan model berbasis *transformer* (Guesmi dkk. 2024) tidak menyerang model *state of the art* serta hanya mengevaluasi *patch* berdasarkan ukurannya. Oleh karena itu, tugas akhir ini akan mencakup evaluasi terhadap model berbasis CNN maupun model *state of the art* berbasis *transformer* seperti Depth Anything.

Kedua, tugas akhir ini akan mengevaluasi serangan *adversarial patch* tidak hanya berdasarkan ukuran, tetapi juga berdasarkan tingkat kecerahan *patch*. Aspek kecerahan dari *patch* tidak dibahas dalam penelitian-penelitian sebelumnya. Seperti yang telah dibahas pada subbab II.5, perbedaan tingkat luminansi suatu objek akan memengaruhi kemampuan deteksi objek tersebut oleh manusia. Oleh karena itu, perlu dianalisis apakah *patch* yang kecerahannya disesuaikan untuk lingkungan tertentu dapat memengaruhi efektivitas serangan.

Ketiga, tugas akhir ini akan mengevaluasi serangan *adversarial patch* dengan memisahkan performa prediksi kedalaman ke dalam rentang jarak tertentu. Hal ini mengatasi keterbatasan *interpretability* dari metrik yang lazim digunakan seperti

AbsRel dan RMSE, yang hanya menunjukkan kesalahan prediksi kedalaman secara keseluruhan tanpa menjelaskan variasi kesalahan pada jarak-jarak tertentu.

Tabel II.3 Penelitian terkait dalam serangan adversarial terhadap *monocular depth estimation*

Judul Penelitian	Metode	Hasil
Physical-World Adversarial Attack on Monocular Depth Estimation with Perspective Hijacking (Zheng dkk. 2024)	Melakukan serangan secara *black-box* terhadap model MDE dengan menempatkan objek fisik biasa yang dioptimasi menggunakan algoritma *particle swarm optimization* (PSO).	Kesalahan estimasi kedalaman meningkat bernilai 14 meter pada pengujian fisik terhadap model CNN pralatih. Serangan tetap efektif pada model yang telah melalui *adversarial training*, dengan kesalahan estimasi kedalaman bernilai 12 meter. Pada kondisi *benign*, kesalahan estimasi hanya sekitar 2-5 meter.
Adversarial Attacks on Monocular Depth Estimation (Zhang dkk. 2020)	Melakukan serangan *white-box* dan *black-box* dalam domain digital terhadap model MDE berbasis CNN menggunakan FGSM, dengan nilai epsilon 16/255.	*Root mean square error* (RMSE) model berbasis ResNet-50 meningkat 10 kali lipat (dari 4,17 meter menjadi 42,27 meter) pada pengujian secara *white-box* dengan dataset KITTI *adversarial*. Pada pengujian *black-box* dengan serangan *transfer* dari ResNet-50 ke ResNet-18, RMSE dari ResNet-18 meningkat 4 kali lipat (dari 4,22 meter menjadi 16,7 meter).
SSAP: A Shape-Sensitive Adversarial Patch for Comprehensive Disruption of Monocular Depth Estimation in Autonomous Navigation Applications (Guesmi dkk. 2024)	Melakukan serangan *white-box* terhadap model MDE berbasis CNN dan *transformer* dengan menempatkan *adversarial patch* secara fisik pada kendaraan. Dilakukan dalam 500 *epoch* dengan ukuran *batch* 8.	*Mean depth error* pada model CNN mencapai 50%, sedangkan *mean depth error* pada model *transformer* mencapai 59% saat diujikan pada dataset kendaraan otonom KITTI yang telah disisipkan *adversarial patch*. Pada kondisi *benign*, *mean depth error* bernilai 0%.

BAB III

ANALISIS MASALAH

Pada Bab III, akan dibahas mengenai analisis permasalahan yang diangkat pada tugas akhir ini, dan analisis kebutuhan berdasarkan masalah yang diangkat. Analisis permasalahan dilakukan melalui tahapan *business understanding*, sedangkan analisis kebutuhan dilakukan melalui *data understanding* pada metodologi CRISP-DM. Tahap *business understanding* bertujuan untuk memahami kebutuhan proyek dari segi bisnis, dan mengkonversi permasalahan tersebut menjadi permasalahan *data mining*. Sementara itu, tahap *data understanding* berfokus terhadap pengumpulan dan pendalaman data yang akan digunakan, dengan menyesuaikan pada kebutuhan bisnis yang telah diidentifikasi pada tahap *business understanding*. Kedua tahapan tersebut akan dibahas secara berturut-turut pada subbab berikut.

III.1 Analisis Kondisi Saat Ini

Bab ini menjelaskan proses dalam menganalisis masalah untuk mengidentifikasi kebutuhan yang diperlukan pada tugas akhir ini berdasarkan tahapan pertama dalam metodologi CRISP-DM, yaitu *business understanding*. Mengacu pada Shalev-Shwartz dkk. (2019), kendaraan otonom merupakan salah satu sistem robotik yang mengikuti arsitektur *sense-plan-act*, yaitu pembagian fungsi sistem robotik berdasarkan tiga tahap utama: persepsi (*sense*), pemrosesan dan perencanaan (*plan*), serta eksekusi tindakan (*act*). Arsitektur ini banyak digunakan dalam sistem robotika dan menjadi kerangka kerja utama dalam teknologi kendaraan otonom. Tahapan-tahapan dalam kendaraan otonom menurut arsitektur tersebut dapat dijabarkan sebagai berikut:

1. *Sense*

   Pada tahap *sense*, kendaraan otonom mengumpulkan data dari lingkungan menggunakan berbagai sensor, seperti kamera, radar, dan LiDAR. Sensor-sensor ini menangkap informasi spasial dan semantik seperti jarak ke objek,

jenis objek, dan kondisi jalan. Data yang terkumpul menjadi landasan untuk *scene understanding* dan pengambilan keputusan oleh kendaraan otonom.

2. *Plan*

   Informasi yang telah dikumpulkan oleh sensor kemudian diproses melalui *sensor fusion* dan algoritma *scene understanding* untuk merancang strategi navigasi, perencanaan lintasan, dan memprediksi interaksi dengan partisipan lalu lintas sekitar kendaraan.

3. *Act*

   Setelah perencanaan aksi telah dilakukan, kendaraan otonom akan menerjemahkan perencanaan tersebut menjadi sebuah aksi pergerakan kendaraan sebenarnya, dengan menggunakan sistem aktuator pada kendaraan, seperti sistem menyetir, pengereman, dan akselerasi.

Estimasi jarak merupakan salah satu komponen penting dalam tahap *sensing*. Informasi jarak yang akurat memungkinkan kendaraan otonom mengambil keputusan navigasi dan pergerakan secara tepat. Seiring meningkatnya kompetisi industri otomotif dalam menghadirkan kendaraan otonom, kebutuhan akan estimasi jarak yang andal juga semakin mendesak. Saat ini, sejumlah produsen otomotif terkemuka memanfaatkan metode *computer vision* untuk mengestimasi jarak berdasarkan citra kamera. Salah satu pendekatan yang banyak digunakan adalah *monocular depth estimation* (MDE) atau prediksi kedalaman dari kamera tunggal, yang telah diimplementasikan oleh perusahaan seperti Tesla dan BYD sebagai bagian dari *advanced driving assistance system* (ADAS). Tesla bahkan sepenuhnya mengandalkan kamera untuk proses *sensing* dan rekonstruksi lingkungan sekeliling kendaraan.

Walaupun tidak terdapat informasi secara publik mengenai model pembelajaran mendalam spesifik yang digunakan oleh produsen otomotif tersebut, studi literatur yang telah dilakukan menunjukkan beberapa model MDE *state-of-the-art* (SOTA) yang tersedia secara publik dengan kinerja tinggi saat pengujian pada lingkungan kendaraan otonom, seperti Monodepth2, Manydepth, dan Depth Anything.

Monodepth2 dan ManyDepth dikembangkan dengan arsitektur *encoder–decoder* berbasis jaringan saraf konvolusional, yang menggunakan ResNet-18 sebagai *backbone*. Kedua model tersebut dilatih secara *self-supervised* dengan kumpulan data (*dataset*) KITTI. Monodepth2 memiliki kinerja estimasi jarak dengan hasil AbsRel mencapai 0,106 dengan $\delta < 1{,}25$ sebesar 87,4%. ManyDepth merupakan pengembangan dari Monodepth2 dengan menambahkan pemanfaatan informasi *multi-frame* (temporal), sehingga lebih tangguh terhadap pergerakan objek. Model ManyDepth mencapai hasil AbsRel 0,098 dengan $\delta < 1{,}25$ sebesar 90%.

Selain model berbasis CNN, terdapat juga model berbasis *transformer* seperti Depth Anything yang memanfaatkan mekanisme *self-attention* untuk memperoleh konteks spasial secara global dari citra. Model ini dilatih dengan proses distilasi oleh DINOv2 (sebagai *encoder* berbasis *vision transformer*) dan *dense prediction transformer* sebagai *decoder*. Depth Anything menunjukkan performa yang lebih tinggi dibanding model berbasis CNN, dengan AbsRel 0,08 dan $\delta < 1{,}25$ sebesar 93,6% pada ViT-Small, dan AbsRel serupa pada ViT-Base dengan $\delta < 1{,}25$ sebesar 93,9%.

Sebuah model MDE dapat dianggap andal untuk pengaplikasian dalam lingkungan kendaraan otonom jika model tersebut memiliki AbsRel kurang dari 0,15 dan nilai $\delta < 1{,}25$ sekurang-kurangnya 80% (Huang dkk. 2025). Berdasarkan kriteria tersebut, seluruh model yang telah dibahas memiliki performa yang andal untuk digunakan dalam sistem persepsi kendaraan otonom.

Akan tetapi, seperti yang telah dibahas dalam studi literatur pada Bab II, ditemukan bahwa kendaraan otonom memiliki titik lemah pada tahap *sensing*, khususnya prediksi kedalaman menggunakan kamera tunggal/*monocular depth estimation* (MDE) berbasis DNN yang rentan terhadap serangan adversarial. Beberapa penelitian telah melakukan evaluasi keandalan model MDE terhadap serangan adversarial, akan tetapi umumnya terbatas pada satu aspek saja. Contohnya, pada serangan berbasis gradien (PGD, FGSM), serangan hanya dievaluasi pada variasi nilai epsilon (tingkat gangguan), dan serangan pada citra yang dimanipulasi dengan

adversarial patch hanya dievaluasi berdasarkan ukuran. Selain itu, belum ada penelitian mengenai keandalan terhadap serangan adversarial yang secara khusus menguji model MDE berbasis *transformer* pada kendaraan otonom.

Kesalahan prediksi kedalaman berdampak langsung pada sistem kendali kendaraan otonom. Prediksi kedalaman yang terlalu dekat terhadap kedalaman sebenarnya memicu pengereman lebih awal dari yang semestinya, yang dapat meningkatkan risiko tabrakan dari belakang ketika kendaraan berhenti mendadak. Sebaliknya, prediksi kedalaman yang terlalu jauh terhadap kedalaman sebenarnya akan menyebabkan kendaraan otonom mengalami keterlambatan pengereman sehingga meningkatkan risiko tabrakan bagian depan.

Oleh karena itu, diperlukan analisis perbandingan model MDE untuk membuktikan model yang memiliki kinerja terbaik saat mengalami serangan adversarial, sehingga dapat diimplementasikan dengan aman dalam sistem *sensing* pada kendaraan otonom. Analisis ini dilakukan untuk menemukan titik lemah dari setiap model terhadap berbagai jenis serangan, sekaligus mengidentifikasi model yang paling andal untuk dijadikan *baseline*. Selain untuk mengidentifikasi pilihan model pralatih terbaik, analisis ini juga bermanfaat bagi peneliti dan praktisi industri dalam menentukan model estimasi jarak yang paling sesuai dan aman digunakan pada sistem robotik, termasuk kendaraan otonom. Perkiraan jarak yang akurat dapat meningkatkan keselamatan lalu lintas, mendukung perencanaan navigasi kendaraan otonom, serta meningkatkan kepercayaan publik terhadap kendaraan otonom sehingga dapat mempercepat proses adopsinya.

III.2 Analisis Kebutuhan

Analisis kebutuhan pada tugas akhir ini dilakukan dengan tahapan kedua dari CRISP-DM, yaitu *data understanding*, yang meliputi pemahaman dan eksplorasi karakteristik data yang sesuai untuk digunakan dalam mencapai tujuan bisnis. Dalam melakukan pembangkitan *adversarial patch* untuk menyerang model MDE, akan digunakan dataset publik yang umum digunakan dalam pelatihan MDE pada lingkungan kendaraan otonom, yaitu dataset KITTI. Sebelum melakukan tahap *data*

preparation, karakteristik dari dataset tersebut akan ditelusuri terlebih dahulu untuk memastikan kesesuaian dataset dalam kebutuhan pembangkitan *adversarial patch* secara realistis. Rincian karakteristik dari dataset KITTI dapat dilihat pada tabel III.1.

Tabel III.1 Rincian karakteristik dari *dataset* KITTI

Karakteristik	**Rincian**
Resolusi Citra	1240x376 *pixel*
Jumlah Gambar	42382
Format Gambar	.png
Resolusi *Depth Map*	1240x376 *pixel*
Sensor *Depth*	LiDAR dan kamera stereo
Format *Depth*	*Depth map* dalam format PNG 16-bit, *point cloud* dalam format .bin
Rentang *Depth*	80 meter

Untuk memenuhi kebutuhan kriteria pada *business understanding* dan memenuhi batasan pada tugas akhir, akan dilakukan penapisan pada dataset dengan hanya mengikutsertakan gambar yang memiliki objek mobil. Dikarenakan dataset ini tidak memiliki data anotasi mengenai objek yang ada pada gambar, anotasi akan dilakukan dengan menggunakan model pendeteksi objek YOLOv8. Anotasi yang dilakukan adalah anotasi label dan *bounding box* objek yang telah dinormalisasi. Pendeteksian dibatasi dengan mengikutsertakan kandidat mobil yang memiliki hasil *confidence* di atas 0,85. Setiap gambar dibatasi untuk memuat satu objek mobil saja. Setelah dilakukan penapisan terhadap dataset dengan memperhatikan kriteria-kriteria tersebut, diperoleh total gambar berjumlah 8.167 yang memenuhi syarat untuk digunakan dalam pembangkitan *adversarial patch*.

Walaupun terdapat data *depth map* (peta kedalaman) dari kamera stereo, data peta kedalaman yang digunakan dalam proses pembangkitan *patch* hanya berasal dari

sensor LiDAR, karena akurasi pengukuran kedalaman yang dihasilkan sensor tersebut lebih tinggi dibandingkan dengan kamera stereo. Pemilihan ini dilakukan untuk memastikan bahwa evaluasi terhadap hasil pembangkitan *adversarial patch* dapat bersifat secara lebih realistis dan representatif terhadap kondisi nyata.

Peta kedalaman yang berasal dari *point cloud* hasil pemindaian sensor LiDAR pada dataset KITTI disimpan dalam berkas dengan format *.bin*. Setiap berkas tersebut berisi kumpulan nilai bertipe *float* dengan struktur data berukuran (N, 4), dengan N adalah jumlah titik hasil pemindaian pada satu *frame*. Setiap baris data berisi koordinat tiga dimensi (x, y, z) dalam sistem koordinat LiDAR serta nilai reflektansi dari setiap titik hasil pengukuran.

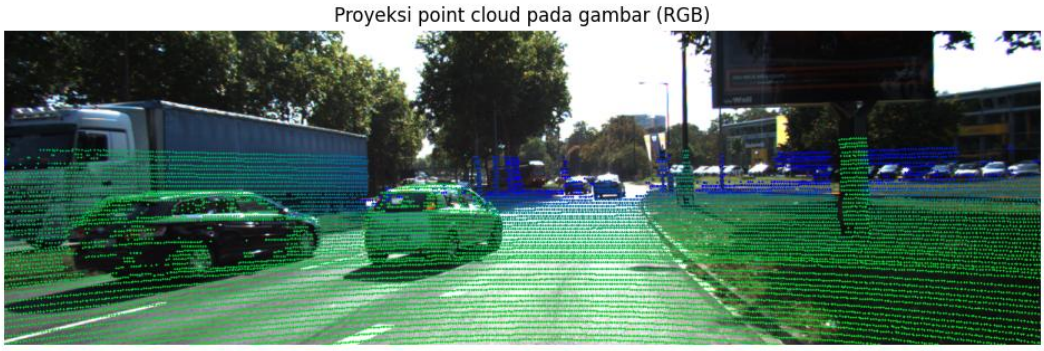
Gambar pada dataset KITTI diperoleh dengan empat kamera yang terpasang pada kendaraan otonom yang masing-masing diberi nomor berturut-turut: kamera 0 untuk citra *grayscale* sisi kiri, kamera 1 untuk citra *grayscale* sisi kanan, kamera 2 untuk citra berwarna sisi kiri, dan kamera 3 untuk citra berwarna sisi kanan. Pada tugas akhir ini, hanya digunakan citra yang diperoleh dari kamera 2 (citra berwarna dari sisi kiri kendaraan). Akan tetapi, parameter kalibrasi yang digunakan tetap mengacu pada kamera 0, karena matriks rektifikasi (R_{rect}) yang tersedia pada dataset KITTI diperoleh berdasarkan kalibrasi pada kamera 0 sebagai kamera acuan.

Data hasil pemindaian LiDAR kemudian dikonversi menjadi peta kedalaman dua dimensi melalui proses proyeksi ke bidang kamera. Konversi ini dilakukan untuk mengubah data kedalaman tiga dimensi hasil pemindaian LiDAR menjadi peta kedalaman dua dimensi yang sejajar dengan citra hasil tangkapan kamera, sehingga setiap titik LiDAR dapat dipetakan ke piksel yang bersesuaian pada citra. Proyeksi tersebut dilakukan menggunakan parameter kalibrasi yang disediakan oleh dataset KITTI, dengan persamaan III.1.

$$Y = P_{rect} R_{rect} T_{velo}^{cam} X \qquad \text{(III. 1)}$$

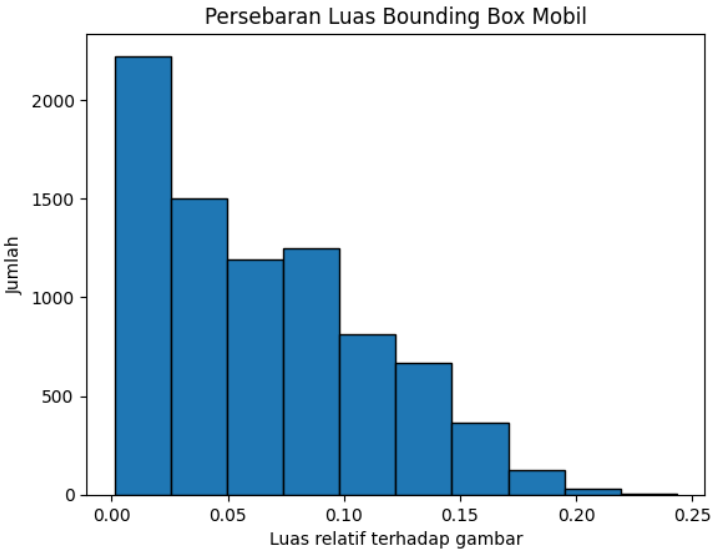
Pada persamaan III.1, X adalah vektor yang merepresentasikan data hasil pemindaian LiDAR, Y adalah peta kedalaman (hasil akhir proyeksi titik pindai LiDAR pada bidang kamera), matriks T_{velo}^{cam} merupakan matriks transformasi kaku

(tidak mengubah ukuran dan bentuk) yang mengubah sistem koordinat LiDAR ke sistem koordinat kamera, R_{rect} merupakan matriks rektifikasi yang berfungsi menghilangkan distorsi dan menyamakan bidang pandang antar kamera berbeda, dan P_{rect} merupakan matriks proyeksi yang mengubah koordinat tiga dimensi hasil rektifikasi menjadi koordinat dua dimensi pada citra sehingga setiap titik LiDAR dapat dipetakan dengan setiap piksel pada citra. Contoh *depth map* yang dihasilkan dari proyeksi menggunakan persamaan tersebut dapat dilihat pada gambar III.1.



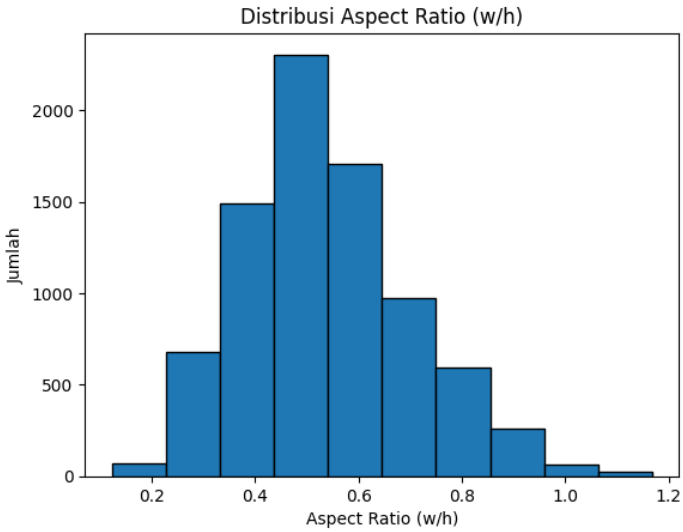
Gambar III.1 Contoh peta kedalaman hasil proyeksi *point cloud*

Untuk memastikan dataset memiliki distribusi data yang seimbang, dilakukan analisis terhadap luas setiap objek mobil dibandingkan dengan keseluruhan citra, serta rasio panjang terhadap lebar dari masing-masing objek. Visualisasi persebaran kedua aspek tersebut berguna untuk mengidentifikasi variasi ukuran dan bentuk objek pada dataset.



Gambar III.2 Persebaran luas objek pada *dataset*

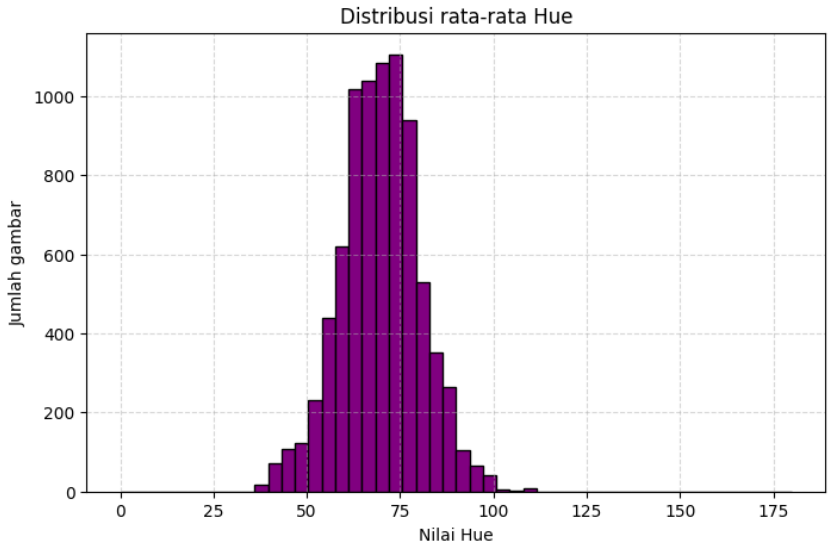
Informasi mengenai luas dan rasio diperoleh dengan memanfaatkan koordinat *bounding box* yang telah dinormalisasi. Persebaran luas dari dataset dapat dilihat pada gambar III.2, dan persebaran dari rasio objek mobil pada dataset dapat dilihat pada gambar III.3.



Gambar III.3 Persebaran *aspect ratio* objek pada dataset

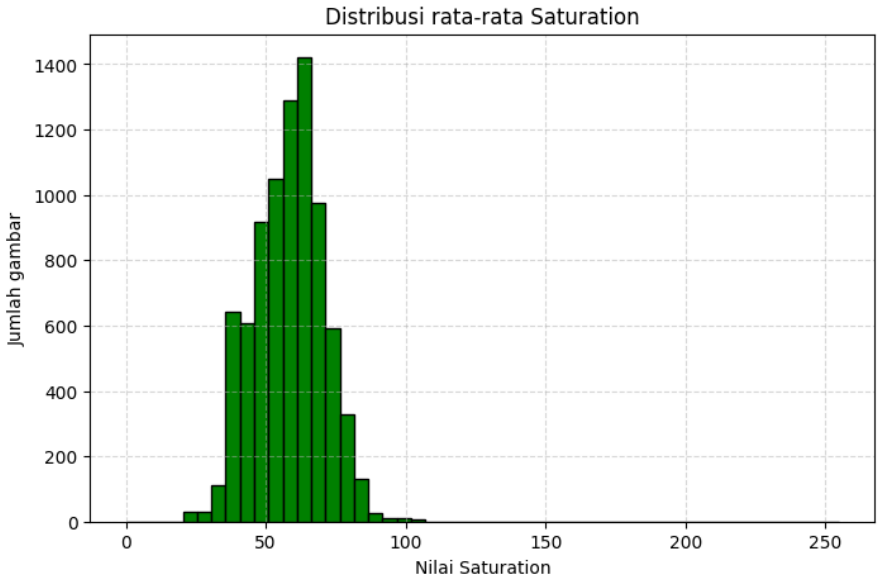
Berdasarkan gambar III.2 dan III.3, diperoleh bahwa dataset memiliki keragaman luas dan rasio, sehingga cukup merepresentasikan berbagai kondisi yang akan

memastikan pembangkitan *adversarial patch* dapat dilakukan secara *robust*. Selain analisis terhadap luas dan rasio objek, juga dilakukan analisis terhadap persebaran warna setiap gambar dalam dataset. Informasi ini diperoleh dengan mengonversi citra dari ruang warna RGB ke ruang warna HSV.



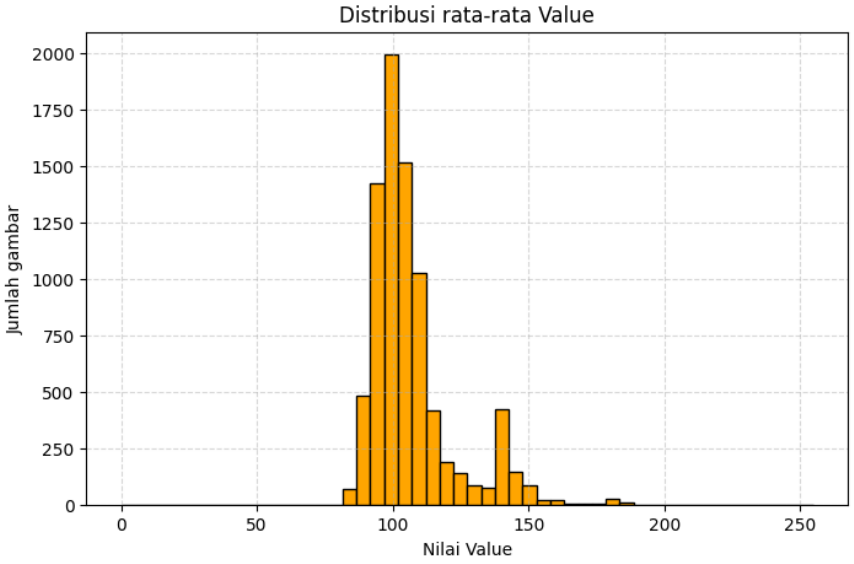
Gambar III.4 Persebaran *hue* pada *dataset*

Penggunaan ruang warna HSV dipilih karena pemisahan antara informasi warna (*hue*), tingkat kejenuhan warna (*saturation*), dan intensitas pencahayaan (*value*) lebih sesuai dalam merepresentasikan distribusi karakteristik citra dibandingkan dengan langsung menggunakan ruang RGB. Pada ruang warna HSV, kanal H (*hue*) merepresentasikan jenis warna dominan dalam citra. Persebaran *hue* pada dataset dapat dilihat pada gambar III.4.



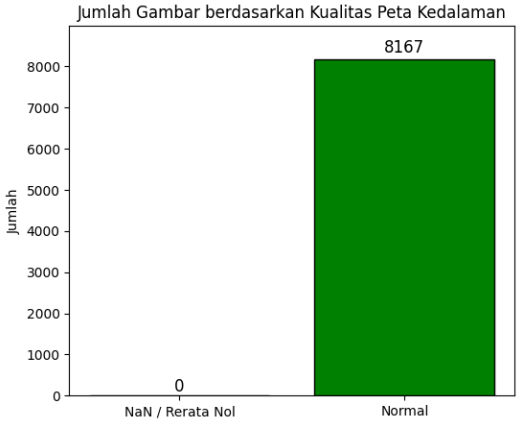
Gambar III.5 Persebaran *saturation* pada *dataset*

Kanal S (*saturation*) menunjukkan tingkat kejenuhan atau intensitas warna, sedangkan kanal V (*value*) menunjukkan tingkat kecerahan atau intensitas cahaya relatif pada citra. Persebaran informasi *saturation* dan *value* pada dataset dapat dilihat secara berturut-turut pada gambar III.5 dan III.6.



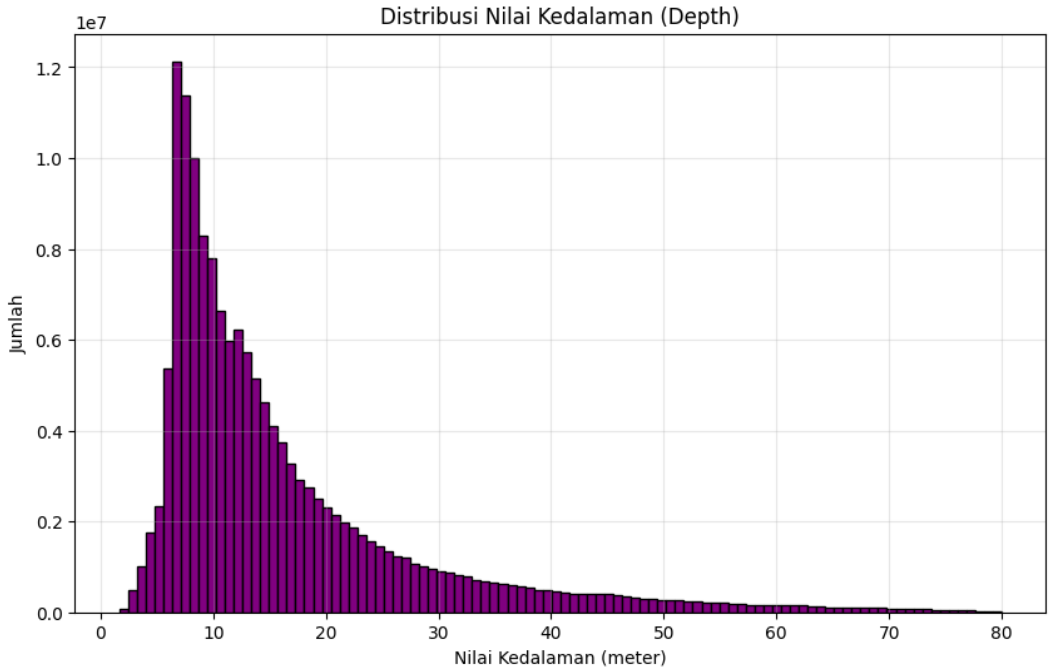
Gambar III.6 Persebaran *value* pada *dataset*

Setelah dilakukan analisis terhadap data citra untuk mengeksplorasi karakteristik dataset dari segi visual, akan dilakukan eksplorasi karakteristik dataset dari segi kedalaman dengan menganalisis *depth map* hasil proyeksi point cloud. Setiap gambar pada dataset KITTI memiliki pasangan *depth map*-nya masing-masing. Analisis dari segi kedalaman mencakup pemeriksaan kualitas peta kedalaman dengan mengidentifikasi adanya nilai nol atau *NaN (not a number)* yang akan mengganggu proses evaluasi *attack success rate* dari *adversarial patch*. Persebaran gambar berdasarkan kualitas peta kedalaman dapat dilihat pada gambar III.7. Berdasarkan gambar tersebut, *depth map* pada keseluruhan citra menunjukkan kualitas yang baik tanpa adanya nilai kedalaman yang akan mengganggu proses evaluasi serangan *adversarial patch*.



Gambar III.7 Persebaran gambar berdasarkan kualitas peta kedalaman

Selain itu, dilakukan juga analisis distribusi titik pada *point cloud* dari dataset berdasarkan jarak kedalaman untuk menilai apakah dataset bersifat representatif terhadap kondisi nyata. Visualisasi distribusi nilai kedalaman dapat dilihat pada gambar III.8.



Gambar III.8 Persebaran jumlah nilai kedalaman dari *point cloud* pada *dataset*

Berdasarkan hasil tersebut, peta kedalaman menunjukkan distribusi yang representatif, dengan mayoritas titik terkonsentrasi pada jarak dekat hingga menengah (0–20 m). Hal ini wajar karena sebagian besar objek yang relevan dengan persepsi kendaraan otonom, seperti kendaraan lain, pejalan kaki, dan infrastruktur jalan, umumnya berada pada jarak tersebut. Sementara itu, titik dengan jarak yang lebih jauh (>20 m hingga 80 m) tetap terdistribusi, meskipun jumlahnya relatif lebih sedikit.

III.3 Analisis Pemilihan Solusi

Terdapat beberapa alternatif solusi berdasarkan penelitian terkait yang dapat dilakukan untuk melakukan evaluasi model MDE berbasis DNN terhadap serangan adversarial dengan penjabaran sebagai berikut:

1. Penempatan Objek Fisik

   Pada solusi ini, penyerang melakukan rekonstruksi perkiraan citra kendaraan otonom dan mengelabui model MDE menggunakan objek fisik sehari-hari seperti tiang, rumput, atau *traffic cone*. Objek fisik tersebut

dimodelkan dalam bentuk 3D, dan ditempatkan pada kandidat lokasi yang diprediksi dapat memberikan tingkat kesalahan maksimal terhadap estimasi kedalaman, dengan optimasi oleh algoritma *particle swarm optimization* (PSO). Kelebihan dari metode ini adalah tingkat *stealthiness* yang tinggi, karena objek yang ditempatkan bersifat *benign*, serta dapat dilakukan secara *black-box*. Akan tetapi, metode ini membutuhkan rekonstruksi 3D secara akurat dari lingkungan kendaraan otonom yang akan diserang, serta dibutuhkan parameter kamera yang presisi.

2. Gangguan Piksel secara Digital (*Digital Perturbation*)

   Pada solusi ini, penyerang melakukan serangan terhadap citra masukan MDE dengan modifikasi piksel pada domain digital menggunakan metode *projected gradient descent* (PGD) atau *fast gradient sign method* (FGSM). Gangguan ini tidak kasat mata bagi manusia namun dapat menyebabkan kesalahan besar pada hasil estimasi kedalaman, sehingga metode ini memiliki tingkat *stealthiness* yang tinggi. Namun, metode ini tidak mempertimbangkan aspek realisme, hanya dapat dilakukan secara *white-box*, serta memiliki *transferability* rendah karena setiap serangan memiliki arah gradien yang berbeda-beda.

3. *Adversarial Patch*

   Pada solusi ini, penyerang menghasilkan sebuah *patch* (potongan kecil) gambar yang dicetak dan ditempelkan pada objek nyata di lingkungan kendaraan otonom. Patch ini dioptimalkan menggunakan proses augmentasi yang mensimulasikan berbagai kondisi dunia nyata, termasuk pencahayaan, *noise*, dan transformasi *affine*. Fungsi *loss* pada pelatihan *patch* ini menggabungkan komponen *non-printability* dan *total variation*. Komponen *non-printability* memaksa rentang warna *patch* pada domain digital agar tidak jauh berbeda dengan rentang warna *patch* saat dicetak. Komponen *total variation* memastikan warna *patch* tidak memiliki banyak *noise*, serta memiliki rentang warna yang relatif selaras. *Patch* dilatih menggunakan model MDE yang menjadi target, dengan melatih *patch* untuk menyebabkan

model mengeluarkan nilai kedalaman tertentu pada area *patch.* Pelatihan dilakukan dengan memperhatikan batasan warna antara citra digital dengan citra fisik (*non-printability*) dan keselarasan warna (*total variation*).

Dalam melakukan analisis pemilihan solusi, akan dilakukan evaluasi terhadap beberapa alternatif solusi yang telah disebutkan berdasarkan empat kriteria, yaitu *replicability*, *reliability*, *transferability*, dan *stealthiness*. *Replicability* mengacu pada seberapa besar kemungkinan suatu serangan adversarial dapat direplikasi oleh pihak lain. *Reliability* didefinisikan sebagai tingkat keberhasilan serangan ketika diujikan dalam berbagai kondisi pada domain fisik. *Transferability* menggambarkan sejauh mana serangan yang efektif pada suatu model tertentu dapat tetap berhasil ketika diujikan pada model lain. *Stealthiness* menunjukkan tingkat kealamian atau tidak mencoloknya suatu serangan, sehingga sulit terdeteksi oleh manusia maupun oleh *adversarial defense* yang tersedia pada DNN. Antara ketiga alternatif solusi yang telah diusulkan sebelumnya, akan dipilih satu solusi terbaik berdasarkan analisis *weighted sum* dengan mempertimbangkan keempat kriteria tersebut. Analisis *weighted sum* dilakukan dengan memberikan bobot pada masing-masing kriteria sesuai urgensinya, lalu menjumlahkan nilai dari tiap solusi berdasarkan nilai yang diberikan. Penilaian dilakukan dengan menggunakan skala nilai 1 hingga 5, dengan nilai 1 menunjukkan performa yang sangat buruk terhadap suatu kriteria, dan nilai 5 menunjukkan performa yang sangat baik terhadap kriteria tersebut. Analisis *weighted sum* ditunjukkan pada tabel III.2.

Berdasarkan hasil analisis pemilihan solusi menggunakan metode *weighted sum*, serangan *adversarial patch* dipilih untuk diujikan karena memperoleh nilai tertinggi pada sebagian besar kriteria evaluasi. Metode ini memiliki tingkat *replicability* yang tinggi, karena *patch* dapat dihasilkan menggunakan model pralatih yang tersedia secara publik. Dari segi *reliability*, *adversarial patch* menunjukkan tingkat keberhasilan serangan yang konsisten dalam berbagai kondisi lingkungan di domain fisik. Selain itu, serangan berbasis *patch* juga memiliki tingkat *transferability* yang tinggi, yang mana telah dibuktikan oleh Brown dkk. (2018).

Tabel III.2 Analisis pemilihan solusi dengan *weighted sum analysis*

Kriteria	**Bobot**	**Penempatan Objek Fisik**	***Digital Perturbation***	***Adversarial Patch***
Replicability	0,3	1	2	5
Reliability	0,3	5	1	4
Transferability	0,2	3	2	4
Stealthiness	0,2	5	4	3
Total		3,4	2,1	**4,1**

BAB IV

DESAIN SOLUSI

Bab ini berisi penjelasan detail mengenai desain sistem yang diusulkan, yaitu pembangkitan serangan adversarial terhadap *monocular depth estimation* (MDE). Berdasarkan analisis masalah yang telah dilakukan pada bagian III.3, diperlukan evaluasi keandalan kendaraan otonom pada tahap *sensing*, khususnya dengan model MDE, terhadap *adversarial example*. Untuk melakukan evaluasi tersebut, akan dibangkitkan *adversarial patch* yang ditempelkan pada objek yang menjadi target dalam citra masukan. Penjabaran mengenai pembangkitan *adversarial patch* tersebut akan dijabarkan lebih lanjut dalam bab ini.

IV.1 Tahapan Desain

Dalam tugas akhir ini, proses evaluasi *adversarial robustness* dari sebuah model *deep learning* mengadopsi beberapa tahapan evaluasi *adversarial robustness* yang diusulkan oleh Papernot dkk. (2015). Proses tersebut terdiri dari beberapa tahap yang akan dibahas lebih lanjut pada subbab-subbab berikut.

IV.1.1 Perumusan *Threat Model*

Tahap pertama dalam mengevaluasi *adversarial robustness* adalah merumuskan *threat model*. Pada pembuatan *threat model*, akan ditetapkan kondisi dan asumsi yang akan menjadi tolok ukur ketahanan model. Papernot dkk. (2015) mengusulkan pembuatan *threat model* menjadi tiga elemen pokok, yaitu tujuan dari *adversary,* ukuran kemampuan *adversary* dalam melakukan serangan, dan pengetahuan *adversary* terhadap model yang akan diserang. Perincian dari ketiga aspek tersebut akan membuat evaluasi *adversarial robustness* menjadi lebih terukur.

IV.1.2 Penentuan Batas Kemampuan *Adversary*

Setelah perumusan *threat model*, langkah selanjutnya adalah menentukan tingkat keandalan model terhadap serangan yang dilakukan oleh *adversary*. Serangan yang

akan dievaluasi perlu didefinisikan sedemikian rupa agar *adversary* melakukan serangan *adversarial patch* tersebut secara realistis sesuai dengan kondisi nyata, sehingga pengujian *patch* dapat dilakukan secara sistematis dan merepresentasikan berbagai kondisi dalam lingkungan kendaraan otonom.

IV.1.3 Evaluasi Model terhadap Serangan Adversarial

Dalam mengevaluasi ketahanan model MDE terhadap serangan adversarial, serangan yang diterapkan harus bersifat adaptif (strategi serangan disesuaikan dengan karakteristik model dan mekanisme pertahanan yang diuji). Salah satu pendekatan dalam melakukan penyerangan tersebut yaitu dengan membangkitkan *adversarial patch* yang efektif pada domain fisik. Pembangkitan *adversarial patch* tersebut dapat dilakukan melalui proses optimasi berbasis *backpropagation*.

IV.2 Hasil Desain

Subbab ini akan membahas hasil dari proses desain dan evaluasi *adversarial robustness* model yang telah dijelaskan pada subbab sebelumnya. Hasil disusun berdasarkan skenario pengujian yang dirancang sesuai dengan *threat model* dan metodologi evaluasi yang telah ditetapkan.

IV.2.1 Desain *Threat Model*

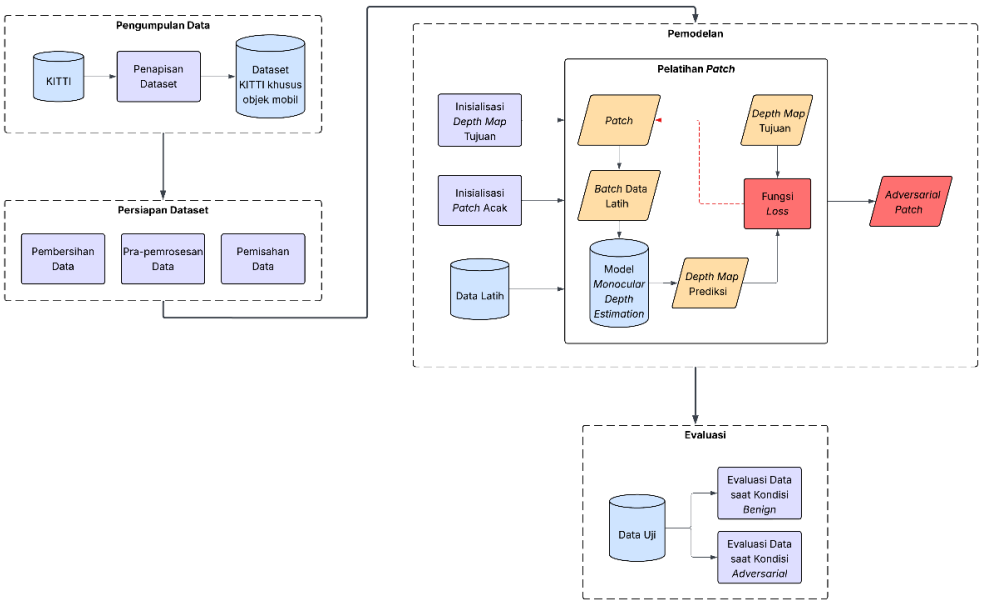
Threat model menggambarkan sebuah kondisi saat seorang *adversary* akan melakukan serangan terhadap model target tertentu. Seperti yang telah dijelaskan pada subbab IV.1, sebuah *threat model* dapat dibuat dengan memperhatikan tujuan *adversary*, kemampuan *adversary*, dan pengetahuan *adversary* terhadap model yang akan diserang. Terdapat dua *threat model* yang didefinisikan pada tugas akhir ini, seperti yang terlihat pada tabel IV.1.

Tabel IV.1 Penjabaran *threat model*

Threat Model	**Tujuan *Adversary***	**Kemampuan *Adversary***	**Pengetahuan *Adversary***
TM-01	*Targeted Misprediction*	*Adversary* dapat meletakkan *patch* secara realistis pada objek yang akan diserang, untuk mengelabui model agar menghasilkan prediksi kedalaman sesuai yang diinginkan *adversary*	*White-box* (*adversary* mengetahui arsitektur dari model, *hyperparameter* model, dan memiliki akses ke data latih model)
TM-02	*Targeted Misprediction*	*Adversary* dapat meletakkan *patch* secara realistis pada objek yang akan diserang, untuk mengelabui model agar menghasilkan prediksi kedalaman sesuai yang diinginkan *adversary*	*Black-box (adversary* tidak mengetahui arsitektur dari model, tidak mengetahui *hyperparameter* dari model, namun memiliki akses ke data latih model)

IV.2.2 Desain Pembangkitan *Adversarial Patch*

Gambar IV.1 mengilustrasikan proses pembangkitan *adversarial patch*. Pembangkitan *adversarial patch* dimulai dari tahap pengumpulan data menggunakan dataset KITTI yang telah ditapis sehingga hanya berfokus pada objek mobil. Dataset hasil penapisan selanjutnya melalui tahap persiapan dataset, yang meliputi pembersihan data, pra-pemrosesan data, serta pemisahan dataset menjadi kelompok data latih dan data uji. Pada tahap pemodelan, *adversarial patch* dibangkitkan dengan menginisialisasi *patch* acak serta peta kedalaman tujuan dalam mengelabui DNN.



Gambar IV.1 Desain pembangkitan *adversarial patch*

Patch kemudian diaplikasikan pada *batch* data latih dan diinferensikan pada model MDE untuk menghasilkan prediksi peta kedalaman. Prediksi tersebut dibandingkan dengan peta kedalaman tujuan melalui fungsi *loss*, yang digunakan untuk memperbarui *patch* secara iteratif hingga terbentuk *adversarial patch* yang optimal. Terakhir, tahap evaluasi dilakukan dengan menguji performa model pada kondisi *benign* (normal) dan adversarial (dengan penyisipan *adversarial patch*) untuk menilai sejauh mana *patch* memengaruhi ketahanan model dalam estimasi kedalaman.

IV.3 Implementasi Pembangkitan *Adversarial Patch*

Implementasi pembangkitan *adversarial patch* mengacu pada metodologi pengembangan yang digunakan pada tugas akhir ini, yaitu CRISP-DM, khususnya pada tahap *data preparation*, *modeling*, dan *evaluation*. Hal ini dilakukan untuk memastikan bahwa proses pembangkitan *patch* mengikuti alur yang sistematis, sehingga hasil yang diperoleh dapat dipertanggungjawabkan secara ilmiah dan sesuai dengan tujuan penelitian.

IV.3.1 *Data Preparation*

Setelah menentukan dataset yang digunakan pada tahap *data understanding*, proses selanjutnya adalah *data preparation*. Pada tahap ini, akan dilakukan persiapan *dataset* untuk membentuk *dataset* akhir yang akan digunakan untuk tahap *modeling*. Tahap ini terdiri atas pembersihan data (*data cleaning*) dan pra-pemrosesan data (*preprocessing*) berdasarkan hasil eksplorasi sebelumnya, serta persiapan data sebelum digunakan pada tahap *modeling*.

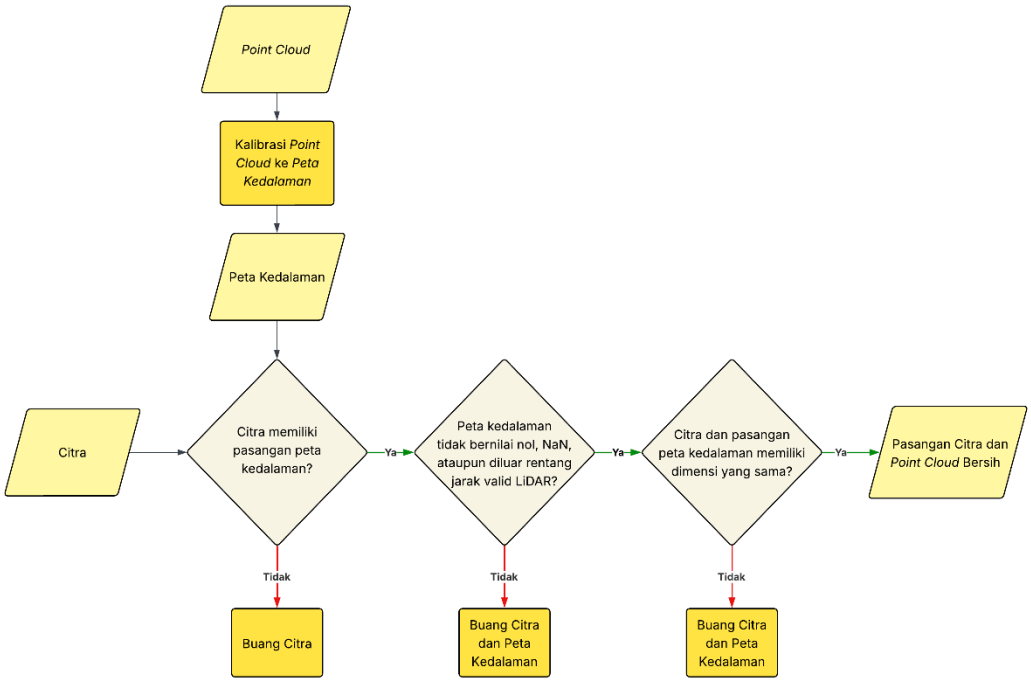
IV.3.1.1 Pembersihan Dataset

Pembersihan data berfokus pada penghapusan *outlier* atau data yang tidak valid, misalnya citra yang rusak atau hilang, peta kedalaman yang mengandung nilai *not a number* (NaN) atau nol secara keseluruhan, serta sampel data yang tidak memiliki pasangan *image–depth* yang sesuai. Pembersihan ini penting untuk menjaga keabsahan hasil perkiraan kedalaman yang dihasilkan oleh model sebagai tolok ukur pembangkitan *adversarial patch*. Selain itu, dilakukan pemeriksaan agar setiap sampel memenuhi kriteria berikut:

1. setiap citra memiliki pasangan *ground truth depth map* yang benar;
2. peta kedalaman tidak mengandung nilai anomali seperti nilai nol, NaN, atau nilai kedalaman di luar rentang yang didukung oleh sensor kedalaman;
3. dimensi citra dan *depth map* konsisten agar dapat diproses oleh *dataloader* pada saat proses pembangkitan *patch*.

Seperti yang telah dibahas pada subbab III.2, data yang dikumpulkan untuk pembentukan *dataset* adalah gambar adegan (*scene*) kendaraan otonom beserta kedalaman sebenarnya dari setiap objek yang terdapat dalam gambar tersebut. Dataset ini dikumpulkan dari dataset KITTI (Geiger, Lenz, dan Urtasun 2012) dan dimodifikasi untuk menyesuaikan kebutuhan pembangkitan *adversarial patch*. Data *point cloud* dari LiDAR diproyeksikan ke koordinat piksel citra menggunakan matriks kalibrasi KITTI untuk menghasilkan peta kedalaman dari citra tersebut.

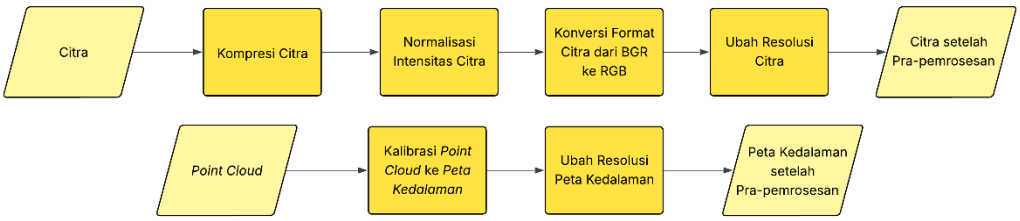
Dilakukan pemeriksaan pada *dataset* untuk memastikan bahwa *dataset* yang digunakan memiliki kualitas gambar dan anotasi yang mumpuni. Pemeriksaaan tersebut dilakukan menggunakan pustaka bawaan Python, yaitu NumPy, untuk memastikan bahwa ketiga kriteria yang telah disebutkan pada paragraf sebelumnya terpenuhi. Pada subbab tersebut, telah dipastikan bahwa seluruh data bersih dari nilai anomali dan dapat digunakan untuk pelatihan, sehingga tidak diperlukan kakas tambahan untuk melakukan pembersihan. Ilustrasi pembersihan dataset dapat dilihat pada gambar IV.2.



Gambar IV.2 Ilustrasi proses pembersihan *dataset*

IV.3.1.2 Pra-pemrosesan Dataset

Setelah dilakukan pembersihan, tahap selanjutnya adalah pra-pemrosesan data. Proses ini mencakup kompresi citra, proyeksi *point cloud* ke peta kedalaman, normalisasi intensitas citra, konversi format warna ke RGB, serta penyesuaian resolusi citra dan peta kedalaman agar sesuai dengan kebutuhan model pralatih. Ilustrasi dari proses yang dilakukan selama pra-pemrosesan dataset dapat dilihat pada gambar IV.3.



Gambar IV.3 Ilustrasi pra-pemrosesan *dataset*

Setiap gambar pada dataset dikonversi ke format .jpg dengan kualitas 92% menggunakan kakas ImageMagick. Kompresi ini bertujuan untuk mengurangi ukuran gambar sekaligus mempercepat proses pemuatan data saat pembangkitan *patch*, tanpa menurunkan kualitas citra secara signifikan. Ukuran dataset yang semula mencapai 22 GB berhasil dikurangi menjadi 9,61 GB.

Data *point cloud* dari LiDAR kemudian diproyeksikan ke koordinat piksel untuk menghasilkan peta kedalaman dua dimensi seperti yang telah dibahas pada bagian *data understanding*. Proyeksi ini diperlukan karena kamera dan sensor LiDAR memiliki perbedaan *field of view* serta posisi fisik pada kendaraan, sehingga keduanya harus dipetakan melalui matriks kalibrasi KITTI agar peta kedalaman sejajar dengan piksel citra. Proses proyeksi dilakukan menggunakan *script* baku yang telah tersedia pada dataset KITTI agar hasil proyeksi dari data LiDAR dapat dipetakan dengan tepat pada citra.

Kemudian, dilakukan normalisasi intensitas citra menggunakan pustaka OpenCV. Setiap nilai piksel pada citra dibagi dengan 255 sehingga rentang intensitas berada pada interval 0 hingga 1. Normalisasi ini berguna untuk meningkatkan kualitas pembangkitan *patch* dengan mempercepat konvergensi selama proses pembangkitan, mencegah masalah *exploding gradient*, serta menyesuaikan tipe data dengan fungsi aktivasi pada DNN.

Konversi format warna ke RGB dilakukan karena citra yang dibaca melalui OpenCV secara *default* berada dalam format BGR, sedangkan model pralatih yang digunakan dilatih menggunakan citra berformat RGB. Oleh karena itu, konversi diperlukan agar citra dapat dimuat pada *framework* pelatihan PyTorch.

Perubahan ukuran resolusi citra dan peta kedalaman dilakukan agar sesuai dengan resolusi masukan dari model pralatih yang akan ditargetkan, karena setiap model pralatih memiliki resolusi inferensi yang berbeda-beda (640×192 pada Monodepth2 dan Manydepth, 518×518 pada Depth Anything). Proses perubahan ukuran kedua data tersebut dilakukan menggunakan pustaka PyTorch.

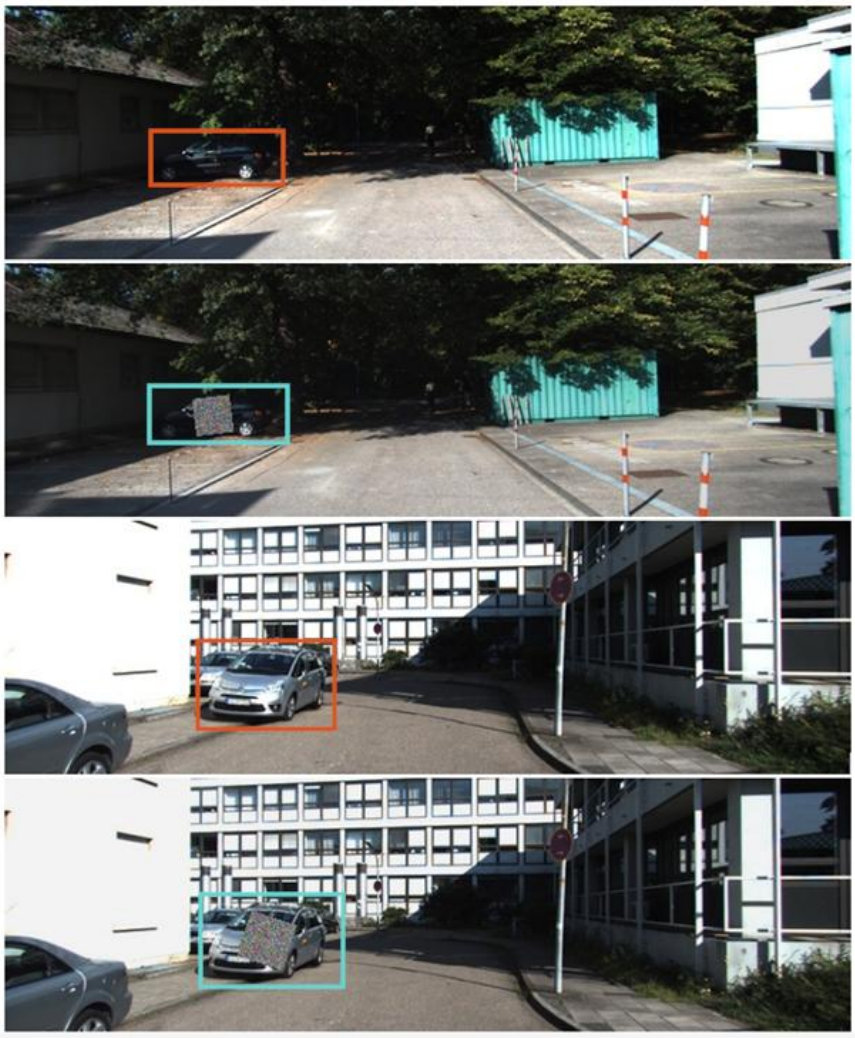
IV.3.1.3 Persiapan Dataset

Setelah dilakukan pembersihan dan pra-pemrosesan dataset, tahap berikutnya adalah augmentasi pada dataset. Augmentasi dilakukan menggunakan salah satu pustaka Python yang lumrah digunakan untuk kebutuhan *computer vision*, yaitu Torchvision. Augmentasi pada citra dilakukan dengan memberikan variasi kontras dengan rentang -10% hingga +10%, serta variasi kecerahan dengan rentang -10% hingga +10%. Tidak dilakukan translasi pada citra, dengan pertimbangan bahwa translasi akan merusak kesesuaian antara citra dan pasangan peta kedalaman (*depth map*) yang digunakan sebagai *ground truth*.

Selain augmentasi pada citra, *patch* juga diaugmentasi dengan memberikan variasi kontras dengan rentang -10% hingga +10%, variasi kecerahan dengan rentang -10% hingga +10%, penambahan *noise* hingga intensitas maksimum 10%, serta rotasi terhadap *patch* dengan rentang -20° hingga +20°. Setelah seluruh proses augmentasi selesai, diperoleh dataset akhir dengan jumlah 8.167 citra. Dataset tersebut kemudian dibagi menjadi dua kelompok, yakni 90% untuk *training* (sebanyak 7349 citra) dan 10% untuk *testing* (818 citra). Pemilihan rasio ini digunakan karena keberhasilan penerapan rasio serupa dalam penelitian sebelumnya yang melakukan pembangkitan *patch* terhadap model pendeteksi objek (Zolfi dkk. 2021).

Contoh hasil augmentasi ditunjukkan pada gambar IV.4, dengan baris ganjil menunjukkan citra asli, dan baris genap menunjukkan hasil augmentasi dari citra beserta *patch* yang telah ditempelkan. Area citra yang diberi kotak merah menampilkan objek asli sebelum augmentasi pada citra dan *patch*, sedangkan area

citra yang diberi kotak biru menampilkan objek setelah augmentasi pada citra dan *patch* diterapkan.



Gambar IV.4 Contoh hasil augmentasi

IV.3.1.4 Data Formatting

Formatting pada dataset dilakukan untuk menyesuaikan struktur data dengan format yang didukung oleh model-model MDE seperti Monodepth2, Manydepth, dan Depth Anything. Ketiga model tersebut mengikuti format dataset KITTI, yang memisahkan citra, data *point cloud*, dan berkas kalibrasi ke dalam folder terstruktur. Setiap folder urutan rekaman memuat citra berwarna dari kamera kiri yang tersimpan dalam subfolder ‘image_02’, serta data *point cloud* LiDAR yang

tersimpan dalam subfolder ‘velodyne_points’. Peta kedalaman tidak disediakan secara langsung, namun dapat dihitung dari *point cloud* mengggunakan parameter kalibrasi yang tersedia. Parameter tersebut tersimpan dalam berkas kalibrasi ‘calib_velo_to_cam.txt’ dan ‘calib_cam_to_cam.txt’.

Berkas kalibrasi ‘calib_velo_to_cam.txt’ berisi matriks kalibrasi T_{velo}^{cam} yang mengubah sistem koordinat LiDAR ke sistem koordinat kamera, sedangkan berkas kalibrasi ‘calib_cam_to_cam.txt’ berisi R_{rect} (matriks rektifikasi) dan P_{rect} (matriks yang memproyeksikan titik tiga dimensi ke bidang dua dimensi). Data-data tersebut tersimpan dalam folder yang terpisah berdasarkan pengelompokannya pada *batch train* dan *test*. Ilustrasi dari struktur folder tersebut dapat dilihat pada gambar IV.5.

```
.
├── lists/
│   ├── car_dataset_train.txt
│   └── car_dataset_test.txt
└── datasets/
    └── content/
        ├── KITTI_TA_car_train_split/
        │   └── 2011_09_26/
        │       ├── calib_cam_to_cam.txt
        │       ├── calib_velo_to_cam.txt
        │       ├── 2011_09_26_drive_0001_sync/
        │       │   ├── image_02
        │       │   └── velodyne_points
        │       ├── 2011_09_26_drive_0002_sync/
        │       │   ├── image_02
        │       │   └── velodyne_points
        │       ├── 2011_09_26_drive_0005_sync/
        │       │   ├── image_02
        │       │   └── velodyne_points
        │       └── ...
        └── KITTI_TA_car_test_split/
            └── 2011_09_26/
                ├── calib_cam_to_cam.txt
                ├── calib_velo_to_cam.txt
                ├── 2011_09_26_drive_0104_sync/
                │   ├── image_02
                │   └── velodyne_points
                ├── 2011_09_26_drive_0106_sync/
                │   ├── image_02
                │   └── velodyne_points
                ├── 2011_09_26_drive_0113_sync/
                │   ├── image_02
                │   └── velodyne_points
                └── 2011_09_26_drive_0117_sync/
                    ├── image_02
                    └── velodyne_points
```

Gambar IV.5 Struktur folder dari *dataset*

Setiap berkas teks yang tersimpan dalam folder ‘lists’ berisi daftar pasangan citra dan *point cloud* yang digunakan pada tahap pembangkitan dan pengujian *patch*. Berkas teks tersebut memiliki format ‘folder_name’, ‘file_name’, dan ‘side’, dengan setiap baris merepresentasikan satu pasangan gambar dan *point cloud*. Nilai *side* menunjukkan posisi kamera pada pengambilan citra dalam dataset KITTI, yaitu ‘l’ untuk kamera kiri dan ‘r’ untuk kamera kanan. Akan tetapi, pada tugas akhir ini hanya digunakan citra dari kamera kiri. Ilustrasi dari format berkas teks tersebut dapat dilihat pada gambar IV.6.

```
2011_09_26/2011_09_26_drive_0106_sync 0000000020 l
2011_09_26/2011_09_26_drive_0106_sync 0000000021 l
2011_09_26/2011_09_26_drive_0106_sync 0000000022 l
2011_09_26/2011_09_26_drive_0106_sync 0000000023 l
2011_09_26/2011_09_26_drive_0106_sync 0000000024 l
2011_09_26/2011_09_26_drive_0106_sync 0000000025 l
2011_09_26/2011_09_26_drive_0106_sync 0000000026 l
2011_09_26/2011_09_26_drive_0106_sync 0000000027 l
2011_09_26/2011_09_26_drive_0106_sync 0000000028 l
2011_09_26/2011_09_26_drive_0106_sync 0000000029 l
```

Gambar IV.6 Format berkas teks untuk pembangkitan dan pengujian *patch*

Untuk memfasilitasi peletakan *adversarial patch* pada objek tujuan, berkas teks pada folder ‘lists’ dimodifikasi dengan menambahkan empat nilai koordinat *bounding box* objek, yaitu ‘x1’, ‘y1’, ‘x2’, ‘y2’. Keempat nilai tersebut merupakan koordinat yang telah dinormalisasi berdasarkan lebar dan tinggi citra, sehingga berada dalam rentang 0 hingga 1.

Setelah penambahan tersebut, format akhir berkas berubah menjadi ‘folder_name’, ‘file_name’, ‘side’, ‘x1’, ‘y1’, ‘x2’, ‘y2’. Format ini digunakan untuk menentukan lokasi objek tujuan pada citra, sehingga *patch* dapat ditempatkan tepat di dalam area *bounding box* tersebut. Ilustrasi dari format berkas setelah penambahan koordinat *bounding box* dapat dilihat pada Gambar IV.7.

```
2011_09_26/2011_09_26_drive_0106_sync 0000000020 l 0.0345 0.5577 0.2402 0.8935
2011_09_26/2011_09_26_drive_0106_sync 0000000021 l 0.0337 0.5582 0.2394 0.8932
2011_09_26/2011_09_26_drive_0106_sync 0000000022 l 0.0332 0.5578 0.2393 0.8957
2011_09_26/2011_09_26_drive_0106_sync 0000000023 l 0.0327 0.5582 0.2391 0.8951
2011_09_26/2011_09_26_drive_0106_sync 0000000024 l 0.0329 0.5572 0.2393 0.8943
2011_09_26/2011_09_26_drive_0106_sync 0000000025 l 0.0329 0.5573 0.2388 0.8950
2011_09_26/2011_09_26_drive_0106_sync 0000000026 l 0.0328 0.5578 0.2390 0.8949
2011_09_26/2011_09_26_drive_0106_sync 0000000027 l 0.0330 0.5578 0.2392 0.8948
2011_09_26/2011_09_26_drive_0106_sync 0000000028 l 0.0329 0.5582 0.2393 0.8949
2011_09_26/2011_09_26_drive_0106_sync 0000000029 l 0.0326 0.5594 0.2392 0.8968
```

Gambar IV.7 Format berkas teks setelah penambahan koordinat *bounding box*

IV.3.2 *Modeling*

Data yang telah dipersiapkan pada tahap *data preparation* kemudian digunakan pada tahap *modeling* untuk melakukan pembangkitan *adversarial patch*. Pembangkitan dilakukan melalui optimasi berbasis gradien (*backpropagation*) yang secara iteratif memperbarui nilai piksel dari *patch* untuk memaksimalkan efektivitas serangan terhadap model target.

Patch yang digunakan sebagai titik awal pembangkitan *adversarial patch* adalah citra dalam ruang warna RGB dengan nilai piksel yang diinisialisasi secara acak dengan ukuran resolusi yang telah ditentukan. *Patch* tersebut kemudian ditempelkan pada objek dengan menyesuaikan ukuran objek tersebut.

Untuk meningkatkan kemampuan serangan dalam berbagai kondisi lingkungan kendaraan otonom, dilakukan augmentasi dengan metode *expectation over transformation* (Athalye dkk. 2017) terhadap *patch* dengan memanfaatkan pustaka Torchvision. Augmentasi ini meliputi translasi, perubahan kontras, perubahan kecerahan, penambahan *noise*, serta perubahan ukuran secara acak (Athalye dkk., 2017). Rincian augmentasi pada *patch* telah dibahas pada subbab IV.3.1.3.

Tabel IV.2 menunjukkan model beserta *hyperparameter* lainnya yang digunakan selama proses pembangkitan *adversarial patch*. *Hyperparameter* yang digunakan dibagi menjadi dua kategori, yaitu *hyperparameter* tetap dan *hyperparameter* dinamis. *Hyperparameter* tetap bernilai konstan untuk seluruh model selama proses pembangkitan *patch*, sedangkan *hyperparameter* dinamis adalah *hyperparameter* yang nilainya disesuaikan melalui beberapa percobaan.

Tabel IV.2 Model dan *hyperparameter* terkait untuk pembangkitan *adversarial patch*

Hyperparameter	**Jenis *Hyperparameter***	**Nilai**
Model	Tetap	Monodepth2, Manydepth, Depth Anything
Backbone	Tetap	1. Monodepth2: ResNet-18 2. Manydepth: ResNet-18 3. Depth Anything: ViT-Small dan ViT-Base
Input Size	Tetap	1. Monodepth2: 640x192 2. Manydepth: 640x192 3. Depth Anything: 518x518
Epoch	Dinamis	10-20
Batch Size	Dinamis	16-32
Optimizer	Tetap	Adam
Learning Rate	Dinamis	0,003-0,01
Loss Function	Tetap	1. Prediksi kedalaman: *Binary Cross-Entropy* (BCE) *Loss* 2. Pembatasan warna: *Non-Printability Score* (NPS) *Loss* 3. Keselarasan warna: *Total Variation* (TV) *Loss*
Loss Factor (BCE)	Tetap	1
Loss Factor (NPS)	Dinamis	0,01-0,5
Loss Factor (TV)	Tetap	2
Target Disparity	Tetap	0,00001

Backbone merepresentasikan komponen pada *deep neural network* (DNN) yang melakukan ekstraksi fitur dari citra masukan. *Input size* menunjukkan ukuran resolusi citra yang dapat diterima oleh model. *Epoch* adalah satu siklus penuh saat proses pelatihan telah melewati seluruh dataset sebanyak satu kali, sedangkan *batch size* merepresentasikan jumlah sampel yang diproses secara bersamaan dalam satu iterasi pelatihan.

Optimizer berfungsi memperbarui bobot (*weight*) dari model berdasarkan hasil perhitungan *loss function* agar proses pembelajaran dapat berjalan dengan lebih cepat dan stabil. *Learning rate* menentukan besar langkah pembaruan bobot dari model pada setiap iterasi *batch*. *Loss function* merepresentasikan fungsi yang digunakan untuk mengukur selisih antara keluaran prediksi model dengan nilai sebenarnya.

Setiap komponen *loss function* tersebut diberikan *loss factor*, yaitu faktor pengali dari setiap *loss* sehingga sehingga hasil akhir pembangkitan *patch* lebih dapat dikendalikan. *Target disparity* menjadi nilai kedalaman acuan yang akan dicapai oleh *patch.*

Seperti yang telah dibahas pada studi literatur (Subbab II.5), proses pembangkitan patch juga harus memperhatikan keselarasan warna dan tingkat *printability* dari patch. Hal ini disebabkan karena warna yang dapat dicetak oleh *printer* hanya sebagian dari ruang warna RGB. Tanpa pembatasan warna pada *patch*, *patch* berpotensi kehilangan efektivitas serangannya ketika diwujudkan pada domain fisik.

Untuk mengatasi hal tersebut, digunakan fungsi *loss* gabungan yang tidak hanya mendorong *patch* untuk menghasilkan gangguan prediksi kedalaman, tetapi juga menjaga agar *patch* tetap berada dalam ruang warna yang dapat dicetak serta memiliki warna yang selaras. Dalam tugas akhir ini, acuan nilai warna yang dapat dicetak oleh *printer* diperoleh dari Sharif dkk. (2016), yang tersedia pada *repository* GitHub (tautan disertakan dalam lampiran) dalam direktori '/lists/ printable30values.txt'.

Bentuk akhir dari *loss function* selama pembangkitan *patch* dapat direpresentasikan sebagai L dalam persamaan IV.1, dengan L_d merupakan *disparity loss* yang mendorong *patch* menghasilkan kesalahan kedalaman pada model tujuan, L_{NPS} merupakan *non-printability score loss* yang menjaga warna *patch* agar berada dalam ruang warna yang dapat dicetak, dan L_{TV} merupakan *total variation loss* yang menjaga keselarasan *warna* dari *patch*. α, β, γ berturut-turut merupakan faktor pengali L_d, L_{NPS}, dan L_{TV}.

$$L = \alpha . L_d + \beta . L_{NPS} + \gamma . L_{TV} \quad \text{(IV. 1)}$$

Pembangkitan *adversarial patch* akan dilakukan pada dua kategori, yaitu kategori tanpa pembatasan kecerahan dan dengan pembatasan kecerahan. Pembatasan kecerahan relatif diterapkan dengan tujuan untuk mengurangi tingkat keterlihatan (*saliency*) dari *patch*. Pembangkitan *patch* dengan pembatasan kecerahan relatif dibagi menjadi tiga kategori berdasarkan komponen nilai V pada ruang warna HSV, yaitu *patch* dengan V di bawah 0,4, *patch* dengan V mulai dari 0,4 hingga 0,7, dan *patch* dengan V di atas 0,7.

Seluruh proses pembangkitan *patch* dilakukan menggunakan *framework* PyTorch, dan dipantau (*monitoring*) dengan kakas CometML. Seluruh proses pembangkitan *patch* dilakukan pada *platform* Google Colab, dengan spesifikasi yang tersedia sebagai berikut:

- dua (2) vCPU;
- RAM 12 GB;
- GPU NVIDIA Tesla T4 dengan VRAM 16 GB.

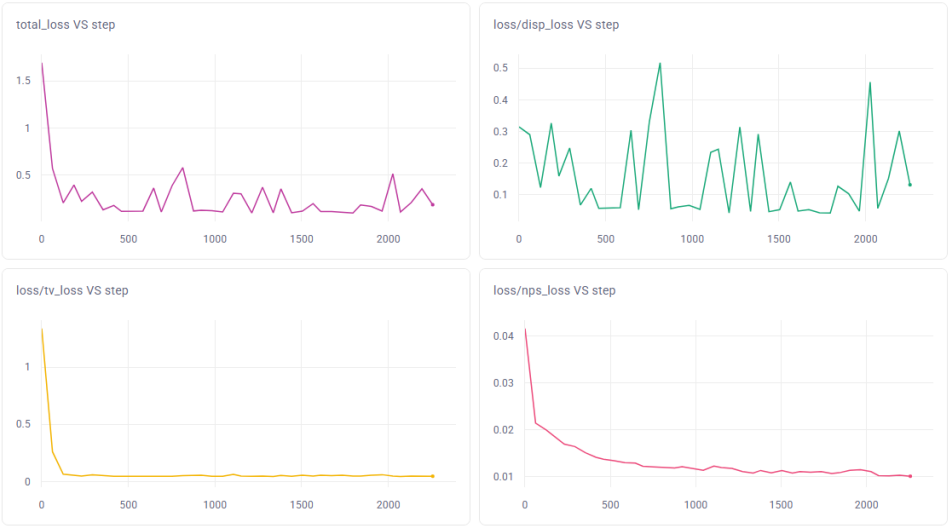
Rangkuman proses pembangkitan *patch* untuk setiap model ditampilkan pada Tabel IV.3. Pembahasan lebih rinci mengenai proses pembangkitan *patch* pada masing-masing model dijabarkan pada subbab-subbab berikutnya.

Tabel IV.3 Rangkuman proses pembangkitan *adversarial patch*

Model	Jenis Patch	Kesimpulan
Monodepth2	Tanpa pembatasan kecerahan	*Disparity loss* fluktuatif, dan hanya mendekati nol pada sebagian langkah pelatihan. *TV loss* dan *NPS loss* konvergen.
	Dengan pembatasan kecerahan	*Disparity loss* fluktuatif, dan hanya mendekati nol pada sebagian langkah pelatihan. *TV loss* dan *NPS loss* konvergen. *NPS loss* pada *patch* dengan V < 0,4 bernilai tinggi pada awal langkah pelatihan.
Manydepth	Tanpa pembatasan kecerahan	*Disparity loss* fluktuatif, dan hanya mendekati nol pada sebagian langkah pelatihan. *TV loss* dan *NPS loss* konvergen.
	Dengan pembatasan kecerahan	*Disparity loss* fluktuatif, dan hanya mendekati nol pada sebagian langkah pelatihan. *TV loss* dan *NPS loss* konvergen. *NPS loss* pada *patch* dengan V < 0,4 bernilai tinggi pada awal langkah pelatihan.
Depth Anything – ViT Small	Tanpa pembatasan kecerahan	*Disparity loss* turun lebih stabil dibandingkan Monodepth2 dan Manydepth, namun tetap hanya mendekati nol pada sebagian langkah pelatihan. *TV loss* dan *NPS loss* konvergen.
	Dengan pembatasan kecerahan	*Disparity loss* lebih fluktuatif dan sulit turun dibandingkan saat proses pembangkitan tanpa pembatasan kecerahan. *TV loss* dan *NPS loss* menurun signifikan pada langkah awal pelatihan, lalu naik dan mengalami konvergensi.
	Tanpa pembatasan kecerahan	*Disparity loss* menurun stabil, dengan kecenderungan menuju nol pada

Model	Jenis Patch	Kesimpulan
Depth Anything – ViT Base		sebagian besar langkah dalam pelatihan. *TV loss* dan *NPS loss* konvergen.
	Dengan pembatasan kecerahan	*Disparity loss* dari *patch* dengan V < 0,4 terendah, kemudian diikuti dengan *patch* V > 0,7, dan terakhir V >= 0,4 dan V <= 0,7. *TV loss* dan *NPS loss* menurun signifikan pada langkah awal pelatihan, lalu naik dan mengalami konvergensi.

IV.3.2.1 Monodepth2

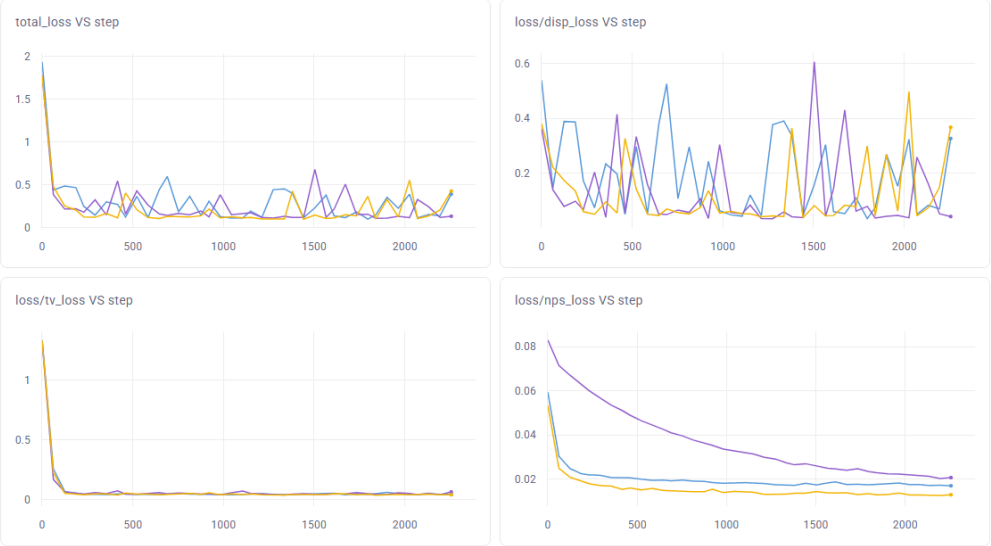


Gambar IV.8 Kurva pembangkitan *adversarial patch* terhadap Monodepth2

Pembangkitan *adversarial patch* pada model MDE Monodepth2 dilakukan secara *white-box*, dengan mengubah resolusi citra masukan dari dataset KITTI (1242×375 piksel) menjadi 640x192 agar sesuai dengan arsitektur model. Proses pembangkitan dilakukan selama 10 *epoch* dengan *batch size* sebesar 32, menggunakan *optimizer* Adam dan *learning rate* 0,01. *Loss factor* pada fungsi *loss* yang digunakan berturut-turut adalah 1 untuk BCE *loss*, 0,01 untuk *NPS* loss, dan 2 untuk *TV* loss.

Perkembangan nilai *loss* selama proses pembangkitan *patch* untuk menyerang model Monodepth2 secara *white-box* dapat dilihat pada Gambar IV.8.

Sedangkan pada pembangkitan patch dengan pembatasan kecerahan relatif, dilakukan penyesuaian agar *optimizer* dapat mengarahkan patch ke dalam rentang warna tertentu. Untuk itu, *loss factor* pada *NPS loss* ditingkatkan menjadi 0,5. Selain itu, *array* berisi kumpulan nilai RGB yang digunakan oleh *NPS loss* sebagai referensi warna yang dapat dicetak juga dimodifikasi agar sesuai dengan kategori kecerahan. Modifikasi dilakukan dengan hanya menyertakan nilai RGB yang memiliki komponen *value (V)* mendekati rentang kecerahan masing-masing kategori setelah dikonversi ke ruang warna HSV. Perkembangan nilai *loss* selama proses pembangkitan *patch* dengan pembatasan kecerahan untuk menyerang model Monodepth2 secara *white-box* dapat dilihat pada Gambar IV.9.

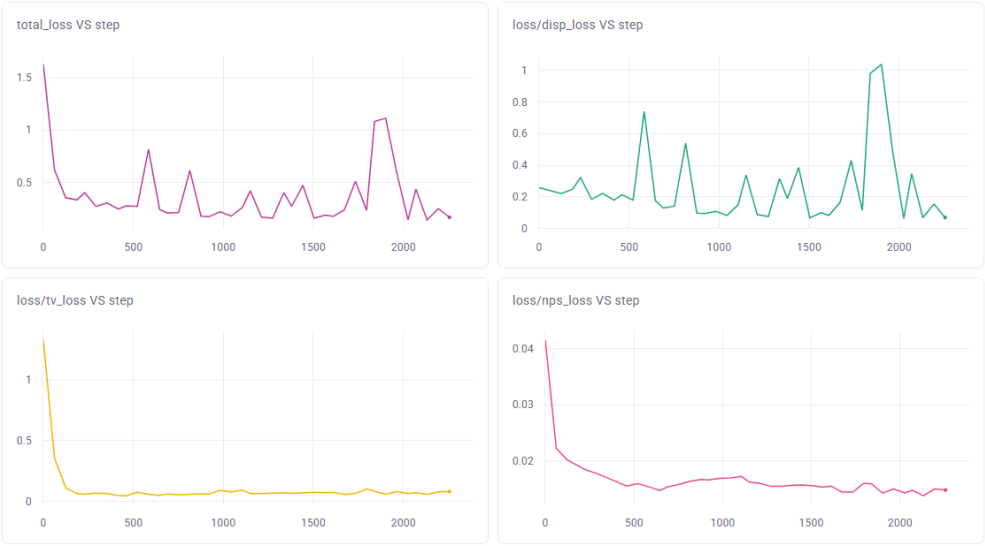


Gambar IV.9 Kurva pembangkitan *adversarial patch* terhadap Monodepth2 dengan pembatasan kecerahan

IV.3.2.2 Manydepth

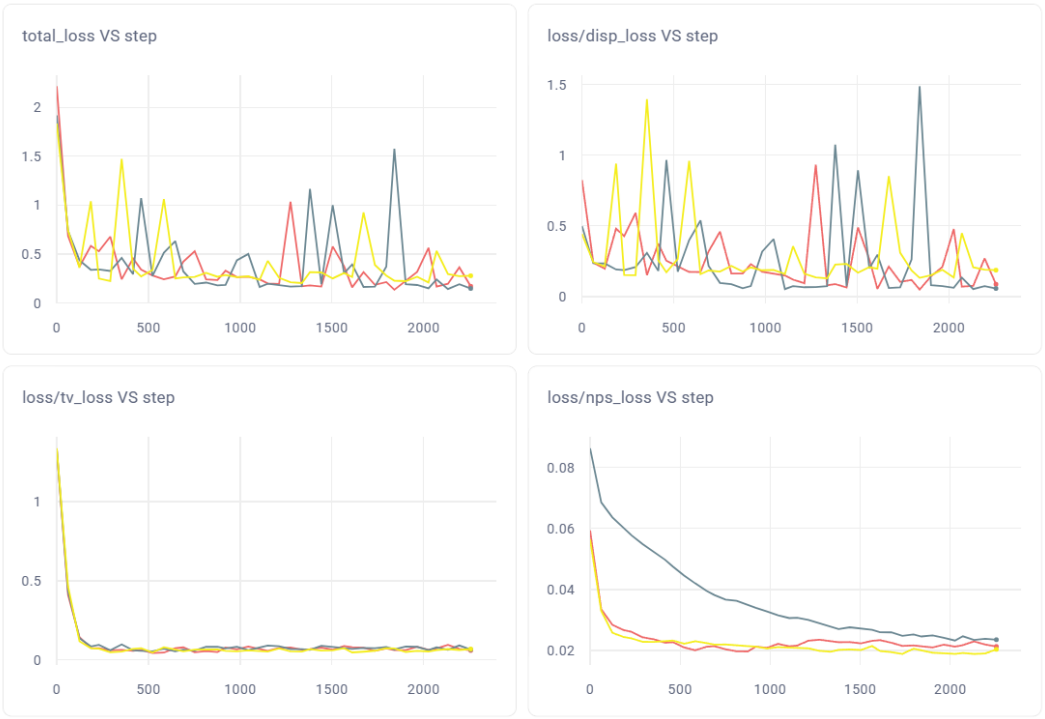
Pembangkitan *adversarial patch* pada model MDE Manydepth dilakukan secara *white-box*, dengan mengubah resolusi citra masukan dari dataset KITTI (1242×375 piksel) menjadi 640x192 agar sesuai dengan arsitektur model. Proses pembangkitan

dilakukan selama 10 *epoch* dengan *batch size* sebesar 32, menggunakan *optimizer* Adam dan *learning rate* 0,01. *Loss factor* pada fungsi *loss* yang digunakan berturut-turut adalah 1 untuk BCE *loss*, 0,01 untuk NPS loss, dan 2 untuk TV loss. Perkembangan nilai *loss* selama proses pembangkitan *patch* untuk menyerang model Manydepth secara *white-box* dapat dilihat pada gambar IV.10.



Gambar IV.10 Kurva pembangkitan *adversarial patch* terhadap Manydepth

Untuk skenario pembangkitan *patch* dengan pembatasan kecerahan, nilai faktor pengali yang digunakan pada NPS *loss* serupa dengan yang digunakan pada model Monodepth2, yaitu 0,5. Modifikasi *array* yang digunakan oleh NPS *loss* juga dilakukan dengan prosedur yang sama seperti pada Monodepth2. Perkembangan nilai *loss* selama proses pembangkitan patch dengan pembatasan kecerahan untuk menyerang model Manydepth secara *white-box* dapat dilihat pada gambar IV.11.

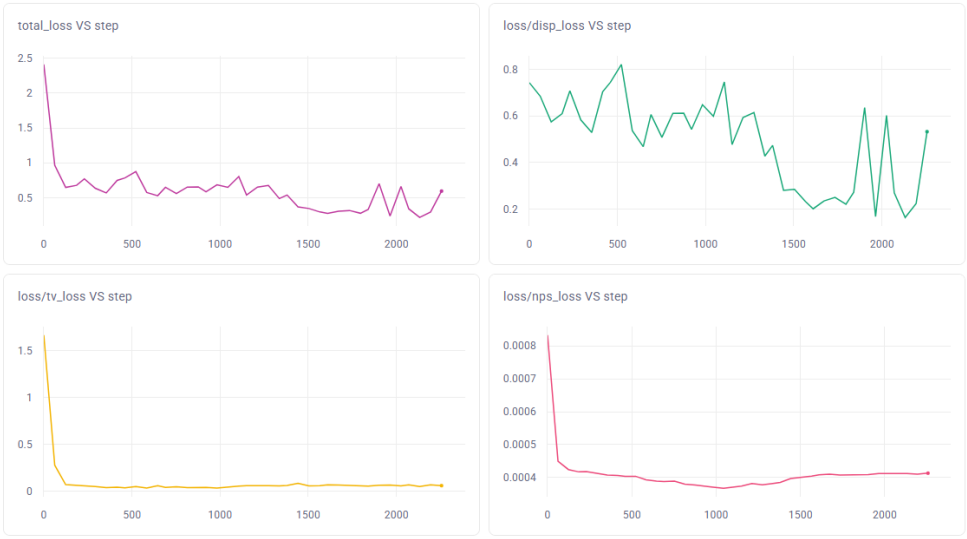


Gambar IV.11 Kurva pembangkitan *adversarial patch* terhadap Manydepth dengan pembatasan kecerahan

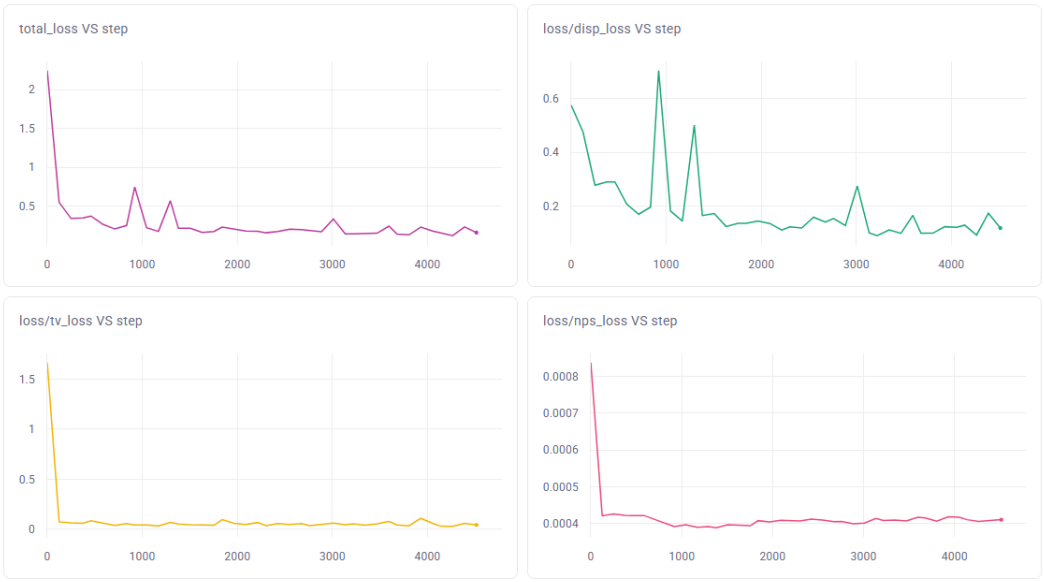
IV.3.2.3 Depth Anything

Pembangkitan adversarial patch pada model MDE Depth Anything dilakukan secara *white-box* dengan menyesuaikan resolusi citra masukan dari dataset KITTI (1242×375 piksel) menjadi 514×514 agar sesuai dengan arsitektur model. Proses pembangkitan diterapkan pada dua varian Depth Anything, yaitu model dengan *backbone* berbasis *vision transformer* jenis ViT-Small dan ViT-Base.

Proses pembangkitan dilakukan selama 10 *epoch* dengan *batch size* sebesar 32, menggunakan *optimizer* Adam dan *learning rate* 0,01. *Loss factor* pada fungsi *loss* yang digunakan berturut-turut adalah 1 untuk BCE *loss*, 0,01 untuk *NPS* loss, dan 2 untuk *TV* loss. Perkembangan nilai *loss* selama proses pembangkitan *patch* untuk menyerang model Depth Anything dengan *backbone* ViT-Small secara *white-box* dapat dilihat pada gambar IV.12.



Gambar IV.12 Kurva pembangkitan *adversarial patch* terhadap Depth Anything – ViT Small

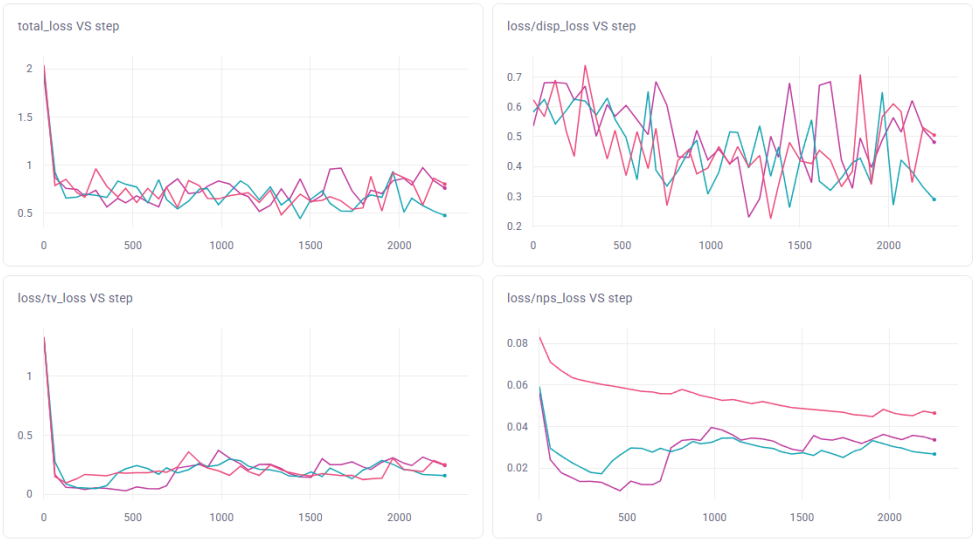


Gambar IV.13 Kurva pembangkitan *adversarial patch* terhadap Depth Anything – ViT Base

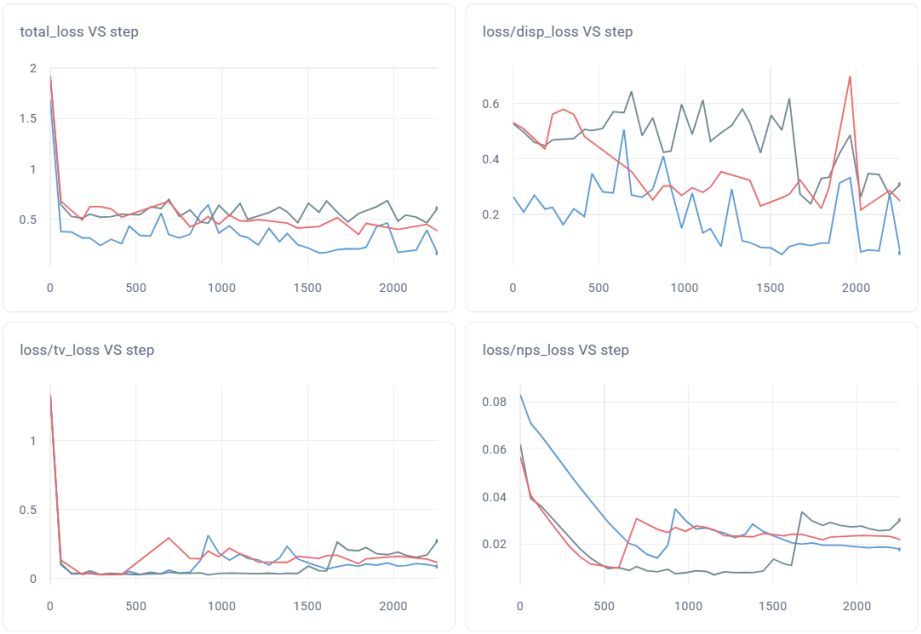
Proses pembangkitan serupa juga dilakukan pada model ViT-Base dengan *hyperparameter* serupa. Perkembangan nilai *loss* selama proses pembangkitan

patch untuk menyerang model Depth Anything dengan *backbone* ViT-Base secara *white-box* dapat dilihat pada gambar IV.13.

Pada model berbasis *transformer* seperti Depth Anything, faktor pengali pada NPS loss sedikit berbeda dibandingkan dengan model CNN. Untuk skenario pembangkitan *patch* dengan pembatasan kecerahan, nilai *loss factor* dari NPS *loss* diturunkan menjadi 0,25 dari yang awalnya 0,5 pada model CNN. Penyesuaian ini dilakukan karena pembangkitan *patch* dengan *loss factor* NPS bernilai 0,5 mengalami kegagalan, karena *patch* tidak mampu memicu kesalahan prediksi. Sementara itu, modifikasi *array* yang digunakan pada NPS loss dilakukan dengan prosedur yang sama seperti pada Monodepth2 dan Manydepth.



Gambar IV.14 Kurva pembangkitan *adversarial patch* terhadap Depth Anything – ViT Small dengan pembatasan kecerahan



Gambar IV.15 Kurva pembangkitan *adversarial patch* terhadap Depth Anything – ViT Base dengan pembatasan kecerahan

Perkembangan nilai loss selama proses pembangkitan patch dengan pembatasan kecerahan untuk menyerang model ViT-Small secara *white-box* dapat dilihat pada gambar IV.14, sedangkan perkembangan nilai *loss* pada *patch* serupa pada ViT-Base juga dapat dilihat pada gambar IV.15.

BAB V

EVALUASI

V.1 Metode Evaluasi

Seperti yang telah disebutkan pada subbab IV.1, proses evaluasi *adversarial robustness* dari DNN dilakukan berdasarkan proses sistematis yang telah diusulkan oleh Papernot dkk. (2015). Evaluasi dilakukan dalam dua tahap. Pertama, model diuji menggunakan data *benign* (tanpa gangguan adversarial) untuk memperoleh performa awal. Setelah itu, model dievaluasi menggunakan data uji yang telah disisipi *adversarial patch* untuk menilai penurunan performa yang disebabkan oleh serangan.

Evaluasi dilakukan menggunakan metrik yang umum digunakan dalam kebutuhan estimasi kedalaman, seperti *absolute relative error (*AbsRel*)* dan *threshold accuracy* ($\delta < 1{,}25$). Metrik AbsRel digunakan untuk mengukur rata-rata kesalahan relatif antara hasil prediksi kedalaman terhadap nilai kedalaman sebenarnya. Nilai ini diperoleh dengan menghitung selisih mutlak antara prediksi kedalaman dengan kedalaman sebenarnya, membagi selisih tersebut terhadap kedalaman sebenarnya, dan merata-ratakan hasil pembagian tersebut pada seluruh piksel dalam citra. Nilai AbsRel yang semakin kecil menandakan bahwa akurasi relatif dari model semakin baik.

Sementara itu, metrik *threshold accuracy* digunakan untuk mengukur jumlah proporsi piksel yang memiliki nilai prediksi kedalaman dalam ambang batas tertentu terhadap nilai *ground truth*, yang direpresentasikan oleh nilai ambang (δ). Prediksi kedalaman pada suatu piksel dianggap benar apabila proporsi nilai prediksi terhadap *ground truth* tidak melampaui batas δ. Semakin tinggi nilai *threshold accuracy*, semakin banyak proporsi piksel dengan prediksi kedalaman yang mendekati nilai sebenarnya.

Sebelum kedua metrik tersebut dihitung, prediksi keluaran model harus dikonversi terlebih dahulu ke dalam kedalaman (*depth*). Hal ini dilakukan karena keluaran dari model *monocular depth estimation* bukan merupakan peta kedalaman secara langsung, namun berupa *disparity*. *Depth* merepresentasikan jarak atau kedalaman suatu objek dari kamera, sedangkan *disparity* menyatakan pergeseran relatif posisi piksel antara dua titik pandang dan berbanding terbalik dengan *depth* (objek yang lebih jauh memiliki nilai *disparity* yang lebih kecil).

Konversi *disparity* ke kedalaman dilakukan menggunakan persamaan V.1 (Godard dkk. 2019), dengan D_{pred} menyatakan hasil konversi *disparity* ke kedalaman, d_{pred} menyatakan nilai *disparity* hasil prediksi dan s menyatakan faktor skala. Nilai s adalah rasio antara median kedalaman *ground truth* dan median kedalaman prediksi dalam satu citra yang sama. Setelah konversi dilakukan, nilai kedalaman kemudian dibatasi dalam rentang antara 0,001 hingga 80 meter (mengikuti batasan dataset KITTI).

$$D_{pred} = s \times \frac{1}{d_{pred}} \tag{V.1}$$

Pada bagian evaluasi yang menyinggung area penempatan patch, proses evaluasi kedalaman tidak dapat dilakukan secara langsung karena peta kedalaman ground truth bersifat *sparse*, hanya tersedia pada titik-titik tertentu berdasarkan hasil pemindaian LiDAR. Oleh karena itu, evaluasi pada bagian tersebut dilakukan menggunakan keluaran *disparity* secara langsung, karena nilai *disparity* tersedia pada seluruh piksel.

Dalam tugas akhir ini, kriteria keberhasilan pertama dinyatakan terpenuhi jika galat relatif yang dihasilkan oleh *adversarial patch* lebih besar dibandingkan dengan galat pada penelitian sebelumnya. Kriteria keberhasilan kedua dinyatakan terpenuhi jika proses pembangkitan *adversarial patch* dapat mencapai konvergensi dalam jumlah iterasi pelatihan yang lebih sedikit dibandingkan penelitian sebelumnya. Sebagai acuan, pembangkitan *adversarial patch* oleh Guesmi dkk. (2024)

membutuhkan sekitar 4000 iterasi, yang diperoleh dari 500 *epoch* dengan *batch size* sebesar 8, dan menimbulkan galat kedalaman hingga 55%.

V.1.1 Konfigurasi Sistem

Pengujian dilakukan oleh satu orang (penulis sendiri). Data yang digunakan dalam melakukan evaluasi adalah *batch* uji dari dataset kendaraan otonom pada subbab IV.3.1 yang telah disisipkan *adversarial patch*. Pembangkitan *adversarial patch* dilakukan menggunakan berbagai perangkat keras, perangkat lunak, dan pustaka sebagai berikut:

1. Perangkat Keras

   Terdapat dua kelompok perangkat keras yang digunakan pada tugas akhir ini, yaitu komputer milik pribadi, dan perangkat keras yang disediakan oleh Google Colab. Perangkat keras yang tersedia pada komputer milik penulis adalah sebagai berikut:

   - CPU Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz,
   - RAM 16 GB
   - GPU NVIDIA GeForce RTX 3070 dengan VRAM 8 GB

   Perangkat keras yang tersedia pada Google Colab adalah sebagai berikut:

   - Dua (2) vCPU
   - RAM 12 GB
   - GPU NVIDIA Tesla T4 dengan VRAM 16 GB

2. Perangkat Lunak

   Perangkat lunak yang akan digunakan adalah Visual Studio Code dan Google Colab sebagai *Integrated Development Environment* (IDE), NVIDIA CUDA sebagai kakas pemrosesan komputasi paralel, dan Github sebagai *version control system*.

3. Pustaka

   Evaluasi *adversarial patch* dilakukan pada *framework* PyTorch. Pustaka torch digunakan untuk mempersiapkan dataset, memfasilitasi inferensi pada model MDE, serta memfasilitasi proses optimisasi dalam pembangkitan

patch. Pustaka torchvision dimanfaatkan untuk melakukan berbagai transformasi citra yang diperlukan selama proses pelatihan *patch*, inferensi, dan evaluasi.

V.1.2 Skenario Evaluasi

Tugas akhir ini dirancang dengan tiga skenario pengujian yang bertujuan untuk mengevaluasi performa dan ketahanan model MDE terhadap serangan adversarial secara menyeluruh. Skenario pengujian pertama dilakukan dengan menguji model pada data uji *benign* untuk memperoleh performa *baseline* dari model. Hasil pengujian dari skenario ini menjadi tolok ukur dalam menilai tingkat penurunan performa model terhadap hasil pengujian dari skenario-skenario berikutnya. Selain itu, dilakukan juga pengujian tambahan dengan mengevaluasi hasil prediksi model terhadap citra yang disisipkan *random patch* (*patch* yang diinisialisasi secara acak). Tujuan dari pengujian tambahan ini adalah untuk memastikan bahwa penurunan performa disebabkan oleh sifat *adversarial* dari *patch* tersebut, bukan semata-mata akibat oklusi.

Skenario pengujian kedua dilakukan dengan menguji model terhadap serangan *white-box*, dengan asumsi bahwa penyerang memiliki akses penuh terhadap informasi internal dari model, seperti arsitektur model, *hyperparameter*, dan gradien dari fungsi *loss*. Dalam skenario ini, *adversarial patch* dibangkitkan secara langsung terhadap model tujuan, disisipkan pada data uji *benign*, dan diinferensikan pada model yang sama untuk mengevaluasi dampak *patch* terhadap performa prediksi kedalaman dari model. Skenario ini menggambarkan tingkat kerentanan model dalam kondisi terburuk, yaitu ketika seluruh informasi internal model diketahui oleh penyerang.

Skenario pengujian ketiga dilakukan dengan menguji model terhadap serangan *black-box*, yaitu kondisi ketika penyerang tidak memiliki akses terhadap arsitektur, *hyperparameter*, maupun gradien fungsi *loss* dari model tujuan. Tujuan dari skenario ini adalah untuk mengukur tingkat *transferability* dari *adversarial patch*, yaitu sejauh mana *patch* yang dihasilkan dari satu model tetap efektif dalam

menurunkan performa model lainnya dengan arsitektur yang berbeda. Skenario ini lebih mencerminkan kondisi dunia nyata, karena penyerang jarang memiliki akses penuh terhadap model yang diserang. Jika performa model tetap menurun signifikan, hal tersebut menunjukkan adanya celah keamanan yang dapat dieksploitasi, bahkan tanpa perlu mengetahui model yang akan diserang. Sebaliknya, jika penurunan performa tidak signifikan, hal ini menunjukkan bahwa efektivitas *patch* sangat bergantung terhadap arsitektur model awal yang menjadi basis pembangkitan *patch*, sekaligus menunjukkan bahwa model tujuan memiliki ketahanan yang lebih baik terhadap serangan antar model.

V.2 Hasil Evaluasi

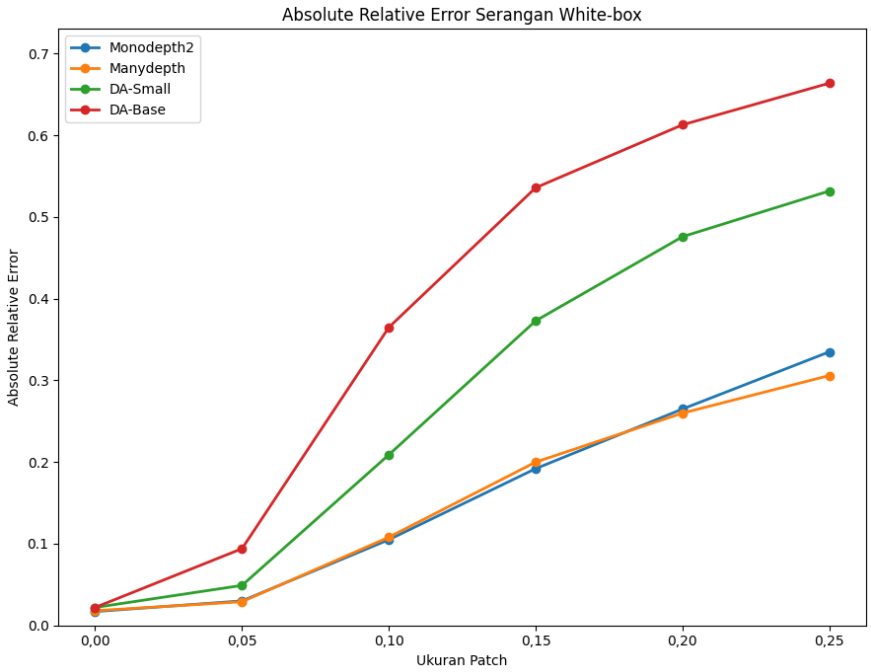
Subbab ini menyajikan hasil evaluasi serangan *adversarial patch* terhadap model MDE berdasarkan variasi ukuran *patch*, jarak objek terhadap kamera, letak *patch* pada citra, serta kecerahan *patch*. Evaluasi dilakukan dengan membandingkan performa model pada kondisi *benign* dan kondisi saat diserang, dengan menggunakan metrik yang telah disebutkan pada metode evaluasi. Untuk analisis yang berkaitan dengan letak patch, nilai *disparity* digunakan secara langsung karena keterbatasan dari *ground truth* kedalaman. Hasil evaluasi untuk setiap aspek dipaparkan pada subbab-subbab berikutnya.

V.2.1 Kinerja Model berdasarkan Ukuran *Patch*

Pemaparan kinerja dari model berdasarkan variasi ukuran patch akan dibagi menjadi dua subbab yang masing-masing membahas satu skenario serangan, yaitu serangan *white-box* dan *black-box*. Pada skenario *benign* (tanpa *patch*), kinerja model dievaluasi dengan menghitung galat relatif mutlak (AbsRel) serta *threshold accuracy* (δ) antara prediksi kedalaman seluruh piksel pada citra terhadap nilai *ground truth*-nya. Sedangkan pada skenario dengan serangan, baik *white-box* maupun *black-box*, evaluasi dilakukan dengan menghitung galat relatif mutlak dan *threshold accuracy* pada area yang ditempati *patch*. Ukuran *patch* pada subbab ini dinyatakan dalam bentuk rasio *patch* terhadap luas objek tempat patch ditempelkan.

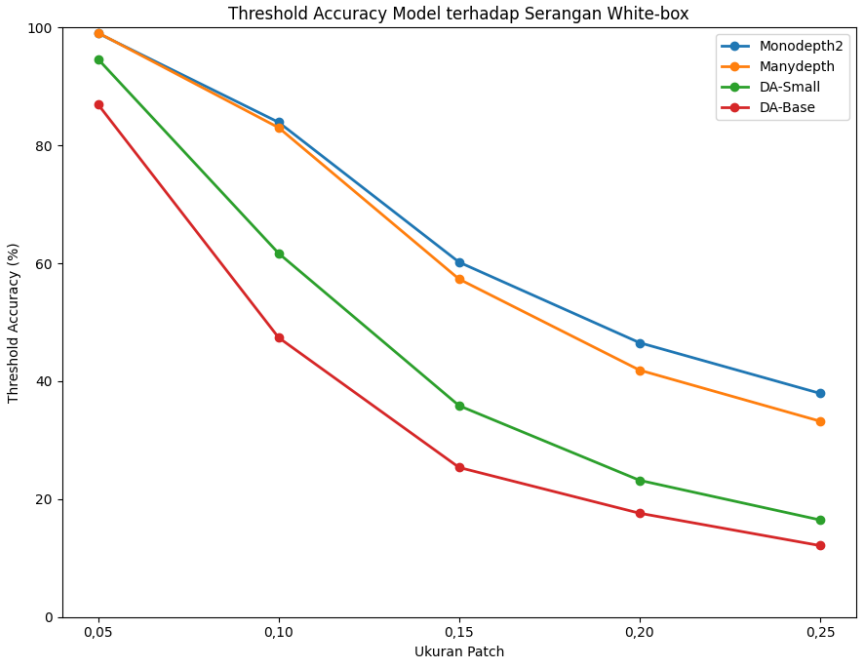
Nilai 0,05, 0,10, 0,15, 0,20, dan 0,25 masing-masing merepresentasikan *patch* dengan luas sebesar 5%, 10%, 15%, 20%, dan 25% dari luas *bounding box* objek.

V.2.1.1 Serangan White-Box



Gambar V.1 Ilustrasi penurunan performa model terhadap serangan *white-box* (dalam AbsRel)

Rincian performa model terhadap serangan adversarial secara *white-box* yang diukur menggunakan metrik AbsRel dapat dilihat pada tabel V.1, sedangkan hasil pengukuran menggunakan metrik δ terhadap serangan adversarial secara *white-box* dapat dilihat pada tabel V.2. Nilai yang dicetak tebal pada kedua tabel tersebut menunjukkan model dengan performa terbaik pada setiap serangan. Ilustrasi penurunan performa model dengan metrik AbsRel dan δ berturut-turut dapat dilihat pada gambar V.1 dan V.2.



Gambar V.2 Ilustrasi penurunan performa model terhadap serangan *white-box* (dalam *threshold accuracy*)

Berdasarkan tabel V.1 dan tabel V.2, model pralatih memiliki performa *baseline* yang sangat baik saat dievaluasi dengan data uji. Performa *baseline* terbaik berturut-turut dimiliki oleh Monodepth2 dengan AbsRel 0,017 dan $\delta < 1,25$ sebesar 99,3%, lalu Manydepth dengan AbsRel 0,018 dan $\delta < 1,25$ sebesar 99,2%, serta Depth Anything yang memiliki AbsRel 0,022 dan $\delta < 1,25$ sebesar 99,2% baik pada varian DA-Small dan DA-Base. Hal ini menunjukkan bahwa keseluruhan model tersebut memiliki kinerja yang sangat baik dalam melakukan estimasi kedalaman terhadap citra yang tidak diserang.

Saat model diserang dalam skenario *white-box*, terlihat bahwa nilai dari metrik *threshold accuracy* maupun AbsRel memburuk secara signifikan seiring dengan bertambahnya ukuran *patch* pada seluruh model yang diuji. Nilai AbsRel dari model berbasis CNN seperti Monodepth2 dan Manydepth menunjukkan kenaikan yang relatif serupa hingga ukuran *patch* 0,2. Pada ukuran patch 0,25, Monodepth2 sedikit melampaui ManyDepth dengan nilai AbsRel sebesar 0,335, sedangkan nilai AbsRel dari ManyDepth adalah 0,306.

Tabel V.1 Rincian performa model terhadap serangan adversarial secara *white-box* (dalam AbsRel)

Serangan	**Ukuran Patch**	**AbsRel**			
		Monodepth2	**Manydepth**	**DA-Small**	**DA-Base**
Tidak ada	-	**0,017**	0,018	0,022	0,022
Random	0,05	0,020 (+0,003)	**0,019 (+0,001)**	0,028 (+0,006)	0,025 (+0,003)
	0,10	0,034 (+0,017)	**0,030 (+0,012)**	0,055 (+0,033)	0,042 (+0,020)
	0,15	0,048 (+0,031)	**0,041 (+0,023)**	0,072 (+0,050)	0,051 (+0,029)
	0,20	0,058 (+0,041)	**0,051 (+0,033)**	0,089 (+0,067)	0,064 (+0,042)
	0,25	0,067 (+0,050)	**0,061 (+0,043)**	0,110 (+0,088)	0,079 (+0,057)
White-box	0,05	0,030 (+0,013)	**0,029 (+0,011)**	0,049 (+0,027)	0,094 (+0,072)
	0,10	**0,105 (+0,088)**	0,108 (+0,090)	0,209 (+0,187)	0,365 (+0,343)
	0,15	**0,192 (+0,175)**	0,200 (+0,182)	0,373 (+0,351)	0,536 (+0,514)
	0,20	0,265 (+0,248)	**0,260 (+0,242)**	0,476 (+0,454)	0,613 (+0,591)
	0,25	0,335 (+0,318)	**0,306 (+0,288)**	0,532 (+0,510)	0,664 (+0,642)

Tabel V.2 Rincian performa model terhadap serangan adversarial secara *white-box* (dalam *threshold accuracy*)

Serangan	**Ukuran Patch**	**δ < 1,25 (%)**			
		Monodepth2	**Manydepth**	**DA-Small**	**DA-Base**
Tidak ada	-	99,3	**99,2**	**99,2**	**99,2**
Random	0,05	99,89 (+0,59)	**99,85 (+0,65)**	99,65 (+0,45)	99,72 (+0,52)
	0,10	99,49 (+0,19)	**99,58 (+0,38)**	98,13 (−1,07)	99,06 (−0,14)
	0,15	98,69 (−0,61)	**99,16 (−0,04)**	96,52 (−2,68)	98,6 (−0,60)
	0,20	97,43 (−1,87)	**98,21 (−0,99)**	94,27 (−4,93)	97,4 (−1,80)
	0,25	96,51 (−2,79)	**96,84 (−2,36)**	91,07 (−8,13)	95,35 (−3,85)
White-box	0,05	99,03 (−0,27)	**99,11 (−0,09)**	94,63 (−4,57)	86,95 (−12,25)
	0,10	**83,93 (−15,37)**	83,02 (−16,18)	61,7 (−37,5)	47,4 (−51,8)
	0,15	**60,19 (−39,11)**	57,32 (−41,88)	35,85 (−63,35)	25,37 (−73,83)
	0,20	**46,53 (−52,77)**	41,88 (−57,32)	23,19 (−76,01)	17,61 (−81,59)
	0,25	**37,94 (−61,36)**	33,22 (−65,98)	16,47 (−82,73)	12,12 (−87,08)

Akan tetapi, berdasarkan metrik *threshold accuracy*, terlihat bahwa penurunan performa terjadi lebih cepat pada Manydepth dibandingkan Monodepth2 ketika ukuran *patch* melebihi 0,1. Pada ukuran *patch* 0,15, akurasi Manydepth turun menjadi 57,32%, sementara Monodepth2 masih lebih andal dengan nilai akurasi

60,19%. Pola ini berlanjut hingga ukuran *patch* terbesar, yaitu 0,25, dengan nilai akurasi 33,22% pada model Manydepth, lebih rendah dibandingkan Monodepth2 dengan nilai akurasi 37,94%.

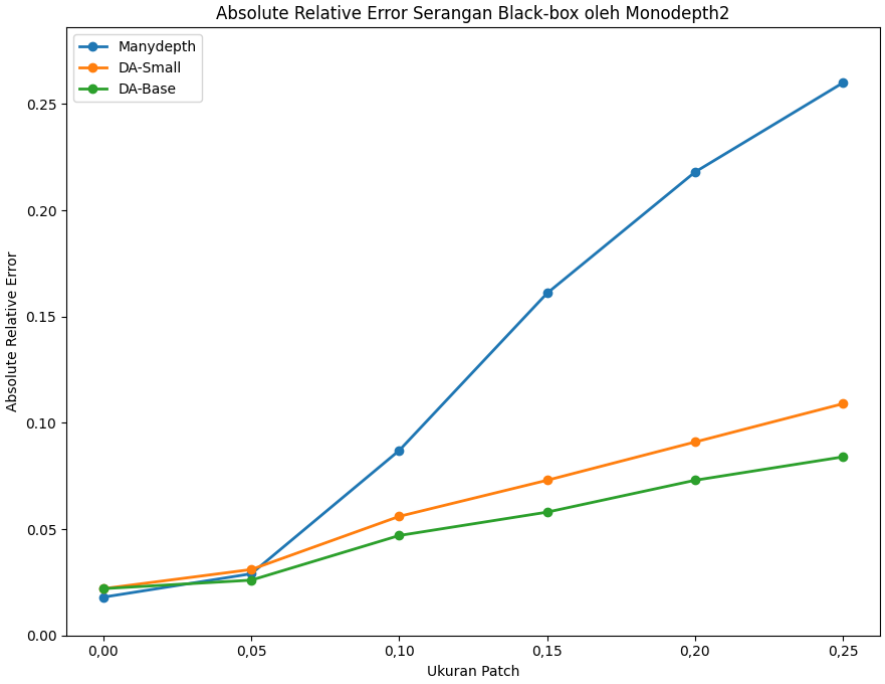
Sementara itu, model berbasis *transformer* seperti Depth Anything menunjukkan kerentanan yang lebih tinggi terhadap serangan *white-box* dibandingkan model berbasis CNN. Nilai AbsRel meningkat tajam seiring bertambahnya ukuran *patch*, khususnya pada varian Depth Anything Base, yang mengalami kenaikan hingga tujuh kali lipat, dari nilai awal 0,094 pada ukuran *patch* 0,05 menjadi 0,664 pada ukuran 0,25. Bahkan pada ukuran *patch* 0,1, Depth Anything Base telah mengalami peningkatan nilai AbsRel secara signifikan, dengan kenaikan dari 0,094 menjadi 0,365. Varian Depth Anything Small menunjukkan kenaikan AbsRel yang lebih bertahap dan sedikit lebih ringan dibandingkan Depth Anything Base, namun dengan nilai akhir yang tetap signifikan, yaitu 0,532 pada ukuran *patch* 0,25.

Penurunan performa juga sangat jelas terlihat pada metrik *threshold accuracy*. Depth Anything Base mengalami penurunan signifikan mulai dari ukuran *patch* 0,1, dengan akurasi 99,06% menjadi 47,4%, dan terus menurun hingga 12,12% pada ukuran 0,25. Depth Anything Small menunjukkan pola serupa dengan akurasi yang menurun dari 98,13% menjadi 61,7% pada ukuran 0,1, dan terus menurun hingga 16,47% pada ukuran 0,25.

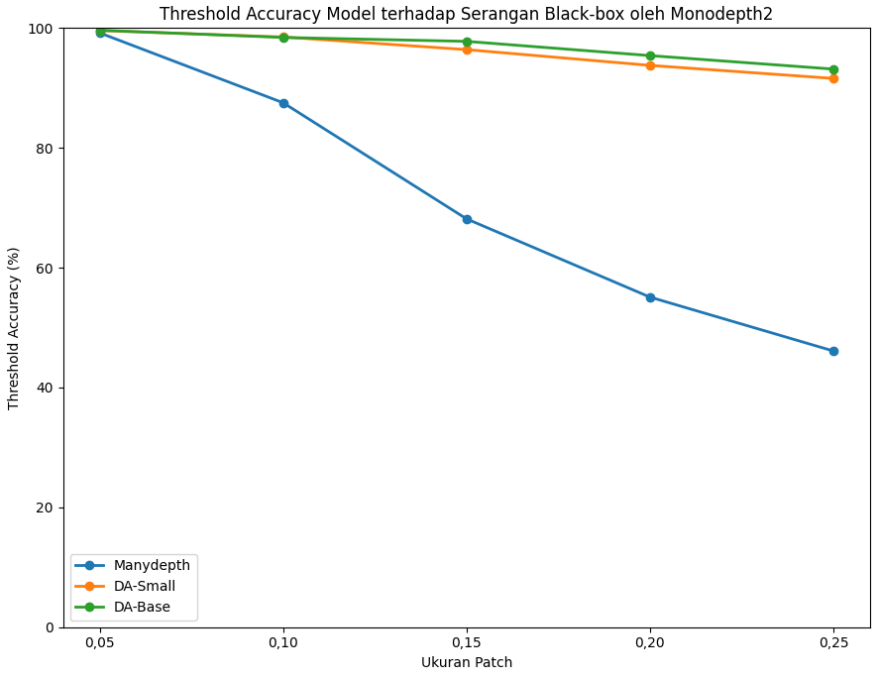
Sebagai pembanding, penyisipan *patch* acak hanya menyebabkan penurunan performa yang sangat kecil pada seluruh model. Nilai AbsRel dari seluruh model tidak meningkat signifikan, dengan nilai dibawah 0,1 bahkan pada ukuran *patch* terbesar. *Threshold accuracy* dari seluruh model juga tidak menurun signifikan, dengan nilai di atas 90% saat disisipkan *patch* acak pada ukuran serupa. Oleh karena itu, dapat dipastikan bahwa peningkatan keberhasilan serangan dan kesalahan estimasi kedalaman bukan semata-mata disebabkan oleh efek penutupan area objek, melainkan akibat optimasi serangan adversarial yang secara langsung mengeksploitasi kelemahan internal dari DNN.

V.2.1.2 Black-Box

Rincian performa model terhadap serangan adversarial secara *black-box* yang diukur menggunakan metrik AbsRel dapat dilihat pada tabel V.3, sedangkan hasil pengukuran menggunakan metrik δ terhadap serangan adversarial secara *black-box* dapat dilihat pada tabel V.4.

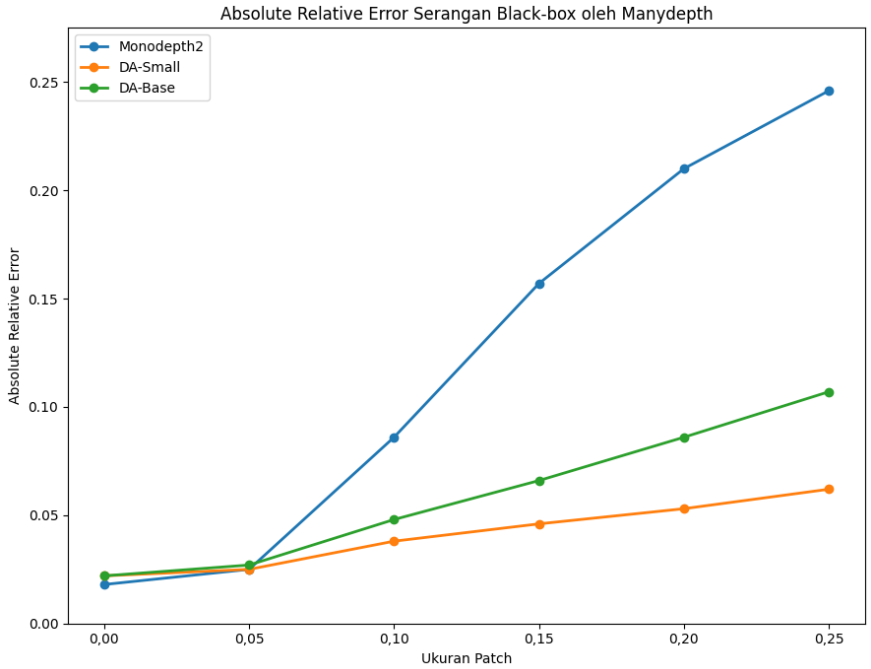


Gambar V.3 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Monodepth2 (dalam AbsRel)

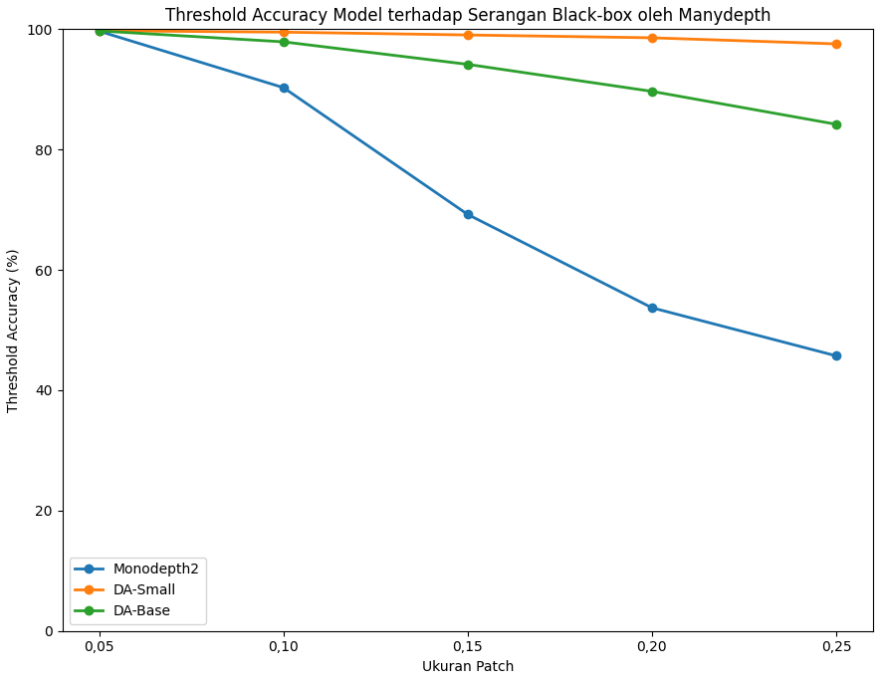


Gambar V.4 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Monodepth2 (dalam *threshold accuracy*)

Ilustrasi penurunan performa model akibat serangan *black-box* oleh *patch* dari Monodepth2 dapat dilihat pada gambar V.3 dan V.4. Serangan patch yang dibangkitkan dari Monodepth2 berhasil menurunkan performa Manydepth dengan pola penurunan yang mirip dengan serangan white-box, meskipun tingkat penurunannya tidak sebesar skenario white-box. Pada ukuran 0,05, nilai AbsRel dari ketiga model relatif mirip, namun pada ukuran di atas 0,05, Manydepth mengalami kenaikan nilai AbsRel yang lebih besar dibandingkan DA-Small dan DA-Base. Pada ukuran patch 0,25, nilai AbsRel Manydepth pada serangan white-box mencapai 0,306, sedangkan pada skenario black-box nilai AbsRel mencapai 0,260, sekitar 15% lebih rendah dibandingkan serangan white-box. Namun, serangan patch tersebut tidak memberikan dampak signifikan terhadap model berbasis transformer, baik pada DA-Small maupun DA-Base.

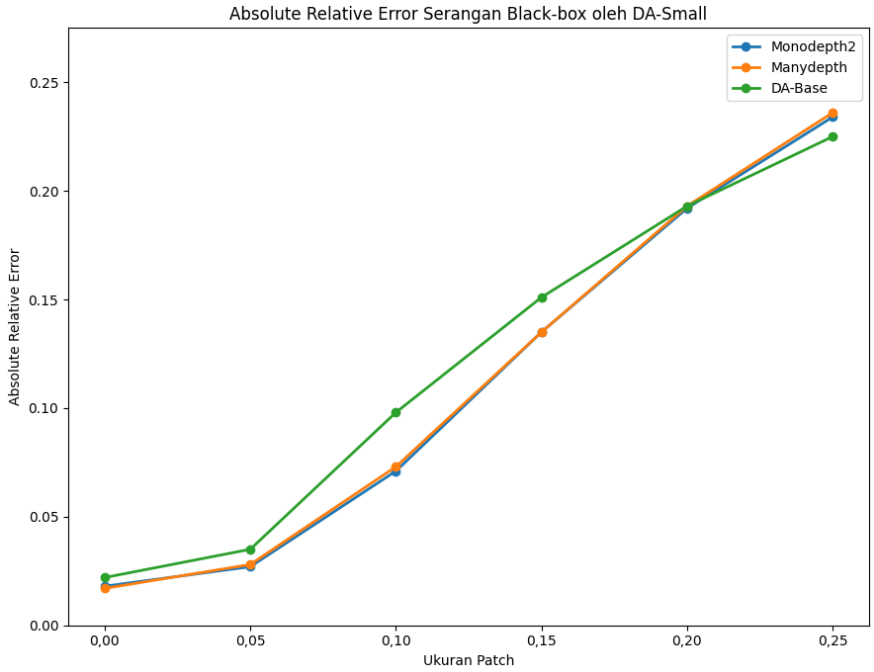


Gambar V.5 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Manydepth (dalam AbsRel)

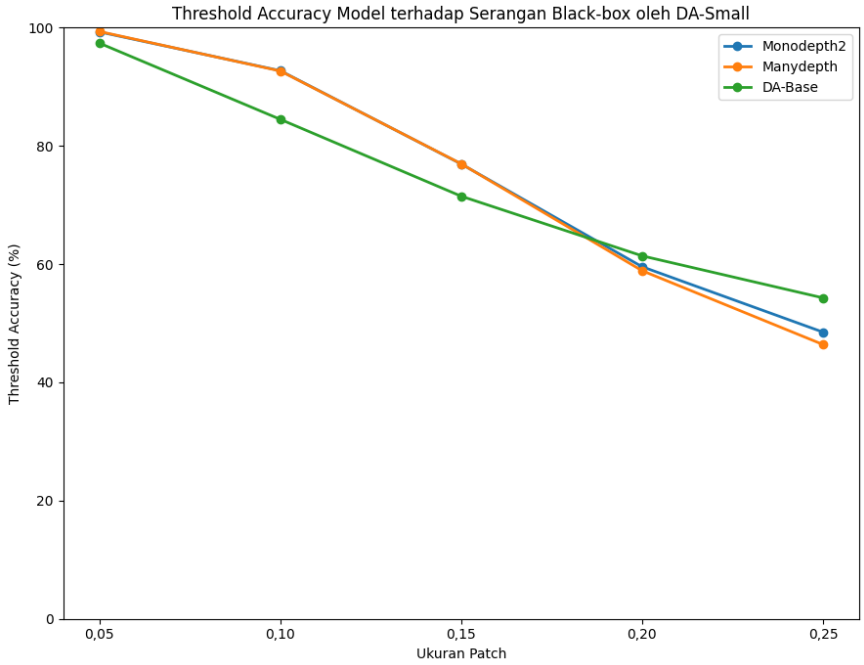


Gambar V.6 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Manydepth (dalam *threshold accuracy*)

Pola serupa juga terlihat ketika patch dibangkitkan dari ManyDepth. Patch tersebut mampu meningkatkan AbsRel pada Monodepth2 dari 0,025 menjadi 0,246 serta menurunkan threshold accuracy menjadi 45,72% pada ukuran patch 0,25. Patch dari ManyDepth juga lebih mampu menyerang DA-Base dibandingkan patch dari Monodepth2, meskipun dampak serangannya tetap lebih kecil dibandingkan dampak serangan terhadap model dengan arsitektur serupa seperti Monodepth2. Ilustrasi penurunan performa model akibat serangan *black-box* oleh *patch* dari Manydepth dapat dilihat pada gambar V.5 dan V.6.

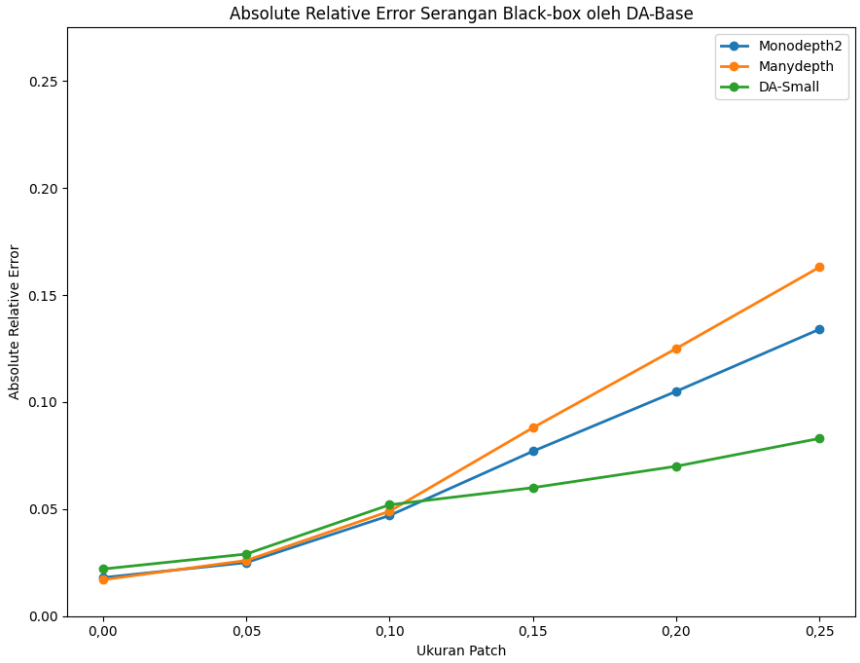


Gambar V.7 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Depth Anything – ViT Small (dalam AbsRel)

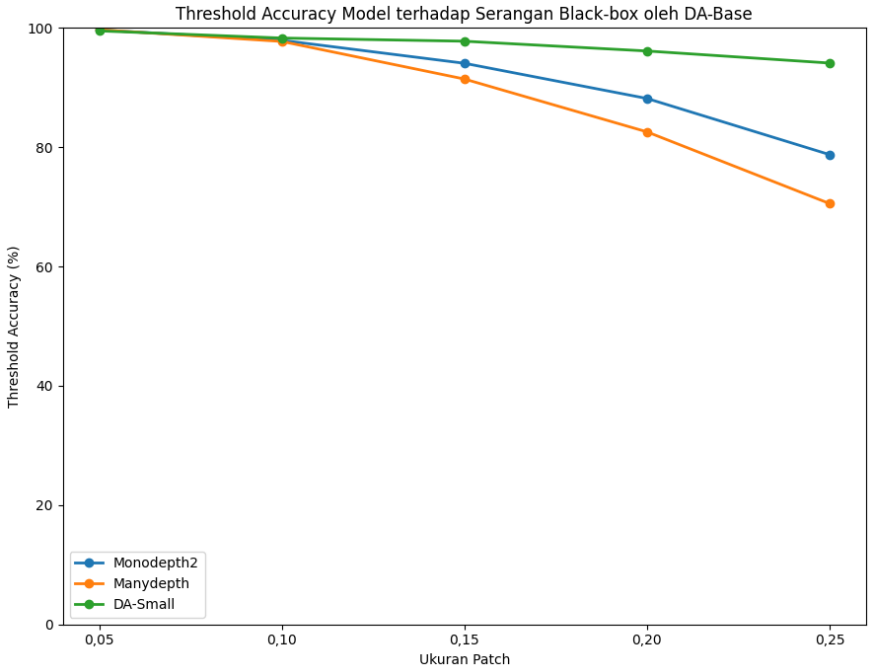


Gambar V.8 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Depth Anything – ViT Small (dalam *threshold accuracy*)

Ilustrasi penurunan performa model akibat serangan *black-box* oleh *patch* dari DA-Small dapat dilihat pada gambar V.7 dan V.8. Patch yang dibangkitkan dari DA-Small memiliki *transferability* serangan tertinggi terhadap seluruh model yang diuji. Hal ini terlihat dari penurunan performa yang signifikan pada ketiga model, baik pada model berbasis CNN maupun transformer. Pada ukuran *patch* 0,25, model DA-Base mengalami peningkatan AbsRel menjadi 0,225 dan penurunan threshold accuracy menjadi 54,29%. Pada ukuran patch 0,05 hingga 0,15, DA-Base mengalami degradasi yang lebih besar dibandingkan Monodepth2 dan ManyDepth. Namun, pada ukuran patch 0,20 ke atas, performa Monodepth2 dan ManyDepth terdegradasi lebih besar dibandingkan DA-Base. Pada ukuran patch 0,25, nilai AbsRel kedua model tersebut meningkat menjadi 0,234 pada Monodepth2 dan 0,236 pada ManyDepth, dengan penurunan *threshold accuracy* masing-masing model menjadi 48% dan 46%.



Gambar V.9 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Depth Anything – ViT Base (dalam AbsRel)



Gambar V.10 Ilustrasi penurunan performa model terhadap serangan *black-box* oleh Depth Anything – ViT Base (dalam *threshold accuracy*)

Sebaliknya, patch yang dibangkitkan dari DA-Base memiliki *transferability* serangan paling rendah. Kenaikan nilai AbsRel yang dihasilkan relatif kecil jika dibandingkan dengan serangan patch dari tiga model sebelumnya, yakni hanya bernilai 0,134 pada Monodepth2, 0,163 pada ManyDepth, dan 0,083 pada DA-Small pada ukuran patch 0,25. Penurunan *threshold accuracy* juga relatif ringan, dengan nilai masing-masing 78,76% pada Monodepth2, 70,55% pada ManyDepth, dan 94,11% pada DA-Small. Pada ukuran patch 0,10 ke bawah, patch DA-Base menunjukkan tingkat serangan yang sangat rendah terhadap seluruh model. Namun pada ukuran patch di atas 0,10, terlihat bahwa model berbasis CNN menjadi lebih rentan dibandingkan DA-Small, meskipun dampak serangannya tetap lemah. Ilustrasi penurunan performa model akibat serangan *black-box* oleh *patch* dari DA-Base dapat dilihat pada gambar V.9 dan V.10.

Tabel V.3 Rincian performa model terhadap serangan adversarial secara *black-box* (dalam AbsRel)

Serangan	**Ukuran Patch**	**AbsRel**			
		Monodepth2	**Manydepth**	**DA-Small**	**DA-Base**
Black-box (dibangkitkan dari Monodepth2)	0,05	–	0,029 (+0,011)	0,031 (+0,009)	0,026 (**+0,004**)
	0,10	–	0,087 (+0,069)	**0,056 (+0,034)**	0,047 (+0,025)
	0,15	–	0,161 (+0,143)	**0,073 (+0,051)**	0,058 (+0,036)
	0,20	–	0,218 (+0,200)	**0,091 (+0,069)**	0,073 (+0,051)
	0,25	–	0,260 (+0,242)	**0,109 (+0,087)**	0,084 (+0,062)
Black-box (dibangkitkan dari Manydepth)	0,05	0,025 (+0,008)	–	**0,025 (+0,003)**	0,027 (+0,005)
	0,10	0,086 (+0,069)	–	**0,038 (+0,016)**	0,048 (+0,026)

Serangan	Ukuran Patch	AbsRel			
		Monodepth2	**Manydepth**	**DA-Small**	**DA-Base**
	0,15	0,157 (+0,140)	–	**0,046 (+0,024)**	0,066 (+0,044)
	0,20	0,210 (+0,193)	–	**0,053 (+0,031)**	0,086 (+0,064)
	0,25	0,246 (+0,229)	–	**0,062 (+0,040)**	0,107 (+0,085)
Black-box (dibangkitkan dari DA-Small)	0,05	0,027 (+0,010)	0,028 (+0,010)	–	**0,035 (+0,013)**
	0,10	0,071 (+0,054)	0,073 (+0,055)	–	**0,098 (+0,076)**
	0,15	0,135 (+0,118)	0,135 (+0,117)	–	**0,151 (+0,129)**
	0,20	0,192 (+0,175)	0,193 (+0,175)	–	**0,193 (+0,171)**
	0,25	0,234 (+0,217)	0,236 (+0,218)	–	**0,225 (+0,203)**
Black-box (dibangkitkan dari DA-Base)	0,05	0,025 (+0,008)	0,026 (+0,008)	**0,029 (+0,007)**	–
	0,10	0,047 (+0,030)	0,049 (+0,031)	**0,052 (+0,030)**	–
	0,15	0,077 (+0,060)	0,088 (+0,070)	**0,060 (+0,038)**	–
	0,20	0,105 (+0,088)	0,125 (+0,107)	**0,070 (+0,048)**	–
	0,25	0,134 (+0,117)	0,163 (+0,145)	**0,083 (+0,061)**	–

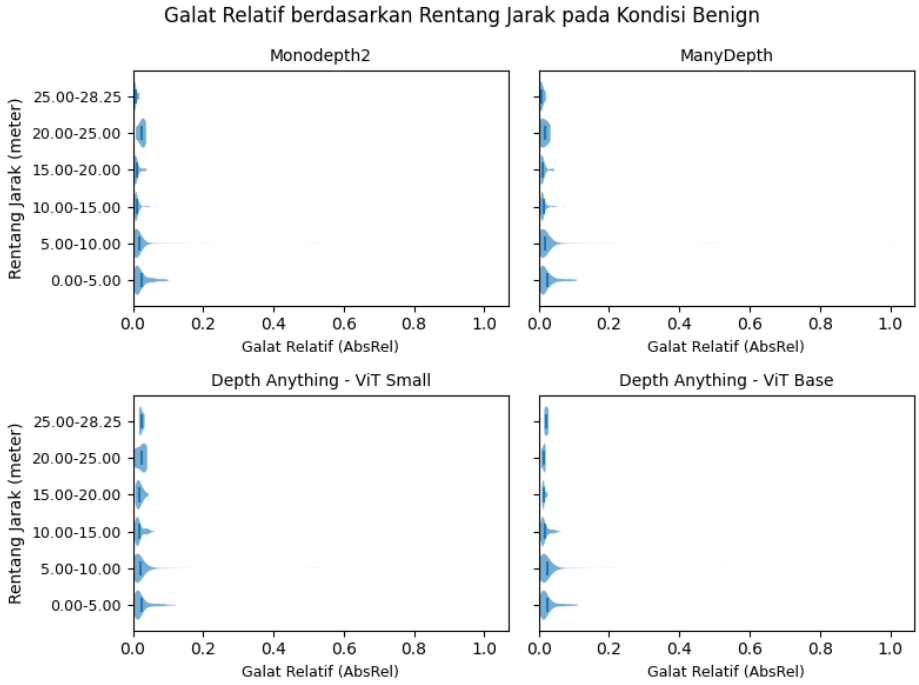
Tabel V.4 Rincian performa model terhadap serangan adversarial secara *black-box* (dalam *threshold accuracy*)

Serangan	**Ukuran Patch**	**δ < 1,25 (%)**			
		Monodepth2	**Manydepth**	**DA-Small**	**DA-Base**
Black-box (dibangkitkan dari Monodepth2)	0,05	–	99,19 (−0,01)	99,58 (+0,38)	**99,64 (+0,44)**
	0,1	–	87,53 (−11,67)	**98,51 (−0,69)**	98,43 (−0,77)
	0,15	–	68,15 (−31,05)	96,40 (−2,80)	**97,78 (−1,42)**
	0,2	–	55,09 (−44,11)	93,77 (−5,43)	**95,40 (−3,80)**
	0,25	–	46,10 (−53,10)	91,59 (−7,61)	**93,15 (−6,05)**
Black-box (dibangkitkan dari Manydepth)	0,05	99,67 (+0,37)	–	**99,75 (+0,55)**	99,72 (+0,52)
	0,1	90,29 (−9,01)	–	**99,51 (+0,31)**	97,90 (−1,30)
	0,15	69,22 (−30,08)	–	**99,04 (−0,16)**	94,16 (−5,04)
	0,2	53,71 (−45,59)	–	**98,57 (−0,63)**	89,67 (−9,53)
	0,25	45,72 (−53,58)	–	**97,56 (−1,64)**	84,20 (−15,00)
Black-box (dibangkitkan dari DA-Small)	0,05	99,28 (−0,02)	**99,35 (+0,15)**	–	97,38 (−1,82)
	0,1	**92,72 (−6,58)**	92,67 (−6,53)	–	84,49 (−14,71)
	0,15	76,94 (−22,36)	**76,97 (−22,23)**	–	71,49 (−27,71)

	0,2	59,56 (−39,74)	58,89 (−40,31)	–	**61,40 (−37,80)**
	0,25	48,50 (−50,80)	46,33 (−52,87)	–	**54,29 (−44,91)**
Black-box (dibangkitkan dari DA-Base)	0,05	99,67 (+0,37)	**99,70 (+0,50)**	99,50 (+0,30)	–
	0,1	97,92 (−1,38)	97,73 (−1,47)	**98,30 (+0,10)**	–
	0,15	94,07 (−5,23)	91,43 (−7,77)	**97,77 (−1,43)**	–
	0,2	88,17 (−11,13)	82,57 (−16,63)	**96,13 (−3,07)**	–
	0,25	78,76 (−20,54)	70,55 (−28,65)	**94,11 (−5,09)**	–

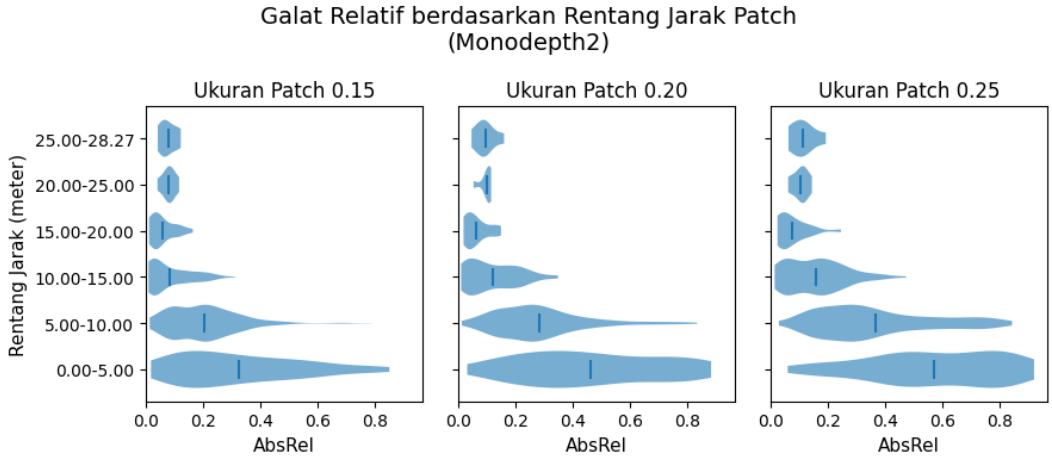
V.2.2 Kinerja Model berdasarkan Jarak *Patch*

Pada subbab ini, akan dijabarkan gambaran kinerja model menggunakan metrik AbsRel. Kinerja model saat diserang digambarkan dalam satu gambar yang terdiri atas tiga *grid*, dengan setiap *grid* mewakili ukuran *patch* yang berbeda. Setiap *grid* menampilkan *violin plot*, dengan sumbu x menunjukkan nilai penyimpangan dalam metrik AbsRel, dan sumbu y menunjukkan rentang jarak tempat *patch* ditempelkan. Garis biru tua pada setiap *violin plot* menunjukkan nilai rata-rata AbsRel pada rentang jarak tersebut, sedangkan *violin plot* berwarna biru muda menunjukkan distribusi nilai AbsRel pada rentang tersebut. Seluruh pengujian *patch* pada subbab ini dilakukan dalam skenario *white-box*.



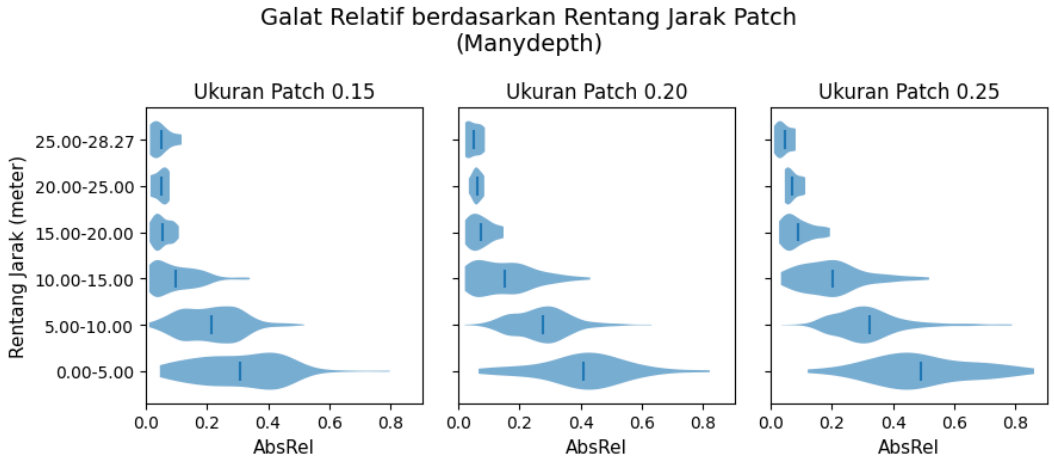
Gambar V.11 Kinerja model berdasarkan rentang jarak saat kondisi *benign*

Sebelum memvisualisasikan kinerja model saat diserang, akan ditunjukkan kinerja model pada kondisi *benign* seperti yang tertera pada gambar V.11. Pada kondisi *benign*, seluruh model memperlihatkan kinerja yang sangat baik, ditunjukkan dengan distribusi serta rata-rata nilai galat relatif yang mendekati nol pada semua rentang jarak.



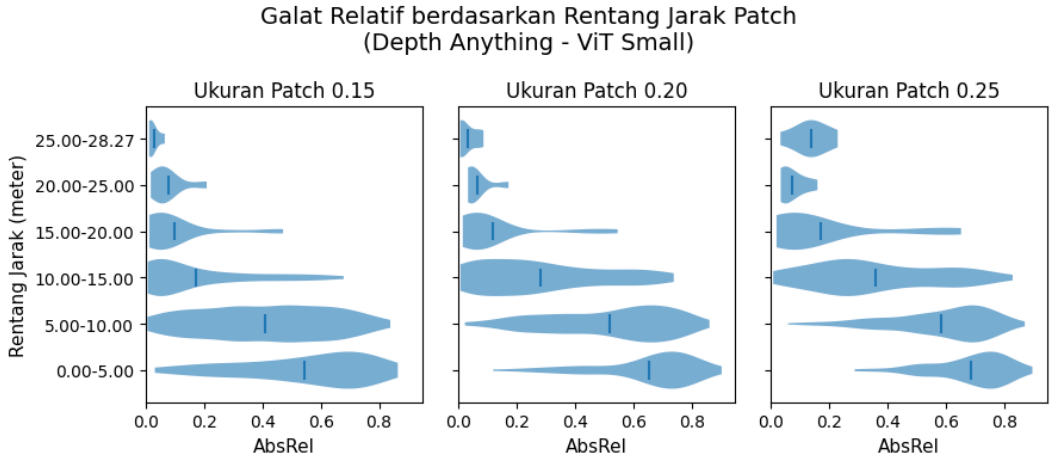
Gambar V.12 Galat relatif Monodepth2 berdasarkan rentang jarak *patch*

Gambar V.12 menunjukkan galat relatif berdasarkan rentang jarak *patch* pada model Monodepth2. Pada model ini, galat relatif tertinggi muncul pada rentang jarak 0-5 meter, kemudian menurun secara bertahap hingga mencapai nilai yang lebih kecil pada rentang 15-20 meter. Pada rentang di atas 15 meter, nilai galat relatif sedikit lebih tinggi dibandingkan kondisi *benign*, namun selisihnya kecil dan median galat tetap berada pada kisaran yang hampir sama dengan keadaan *benign*.



Gambar V.13 Galat relatif Manydepth berdasarkan rentang jarak *patch*

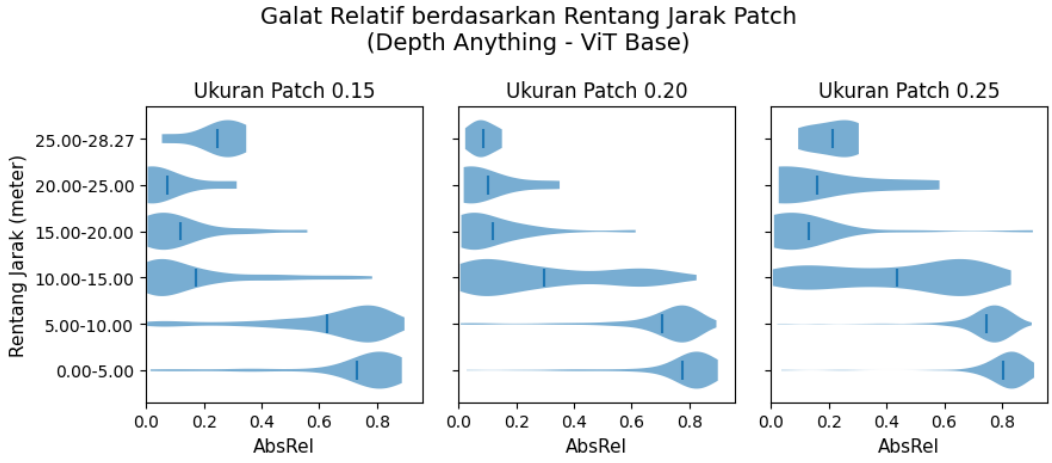
Gambar V.13 menunjukkan galat relatif berdasarkan rentang jarak patch pada model ManyDepth. Pola distribusi galat dari model ini serupa dengan Monodepth2, dengan galat terbesar pada rentang 0-5 meter dan nilai galat yang menurun secara bertahap hingga rentang jarak 15-20 meter. Pada jarak di atas 20 meter, ManyDepth terlihat sedikit lebih andal dibandingkan Monodepth2, ditunjukkan oleh median dan distribusi galat yang lebih rendah. Namun, pada rentang 10-20 meter, Monodepth2 menunjukkan kinerja yang lebih baik, dengan selisih kinerja terhadap Manydepth yang semakin besar seiring bertambahnya ukuran *patch*. Pada rentang 0-10 meter, Monodepth2 lebih andal pada ukuran patch 0,15 dan 0,20, namun ManyDepth justru lebih unggul pada ukuran patch 0,25.



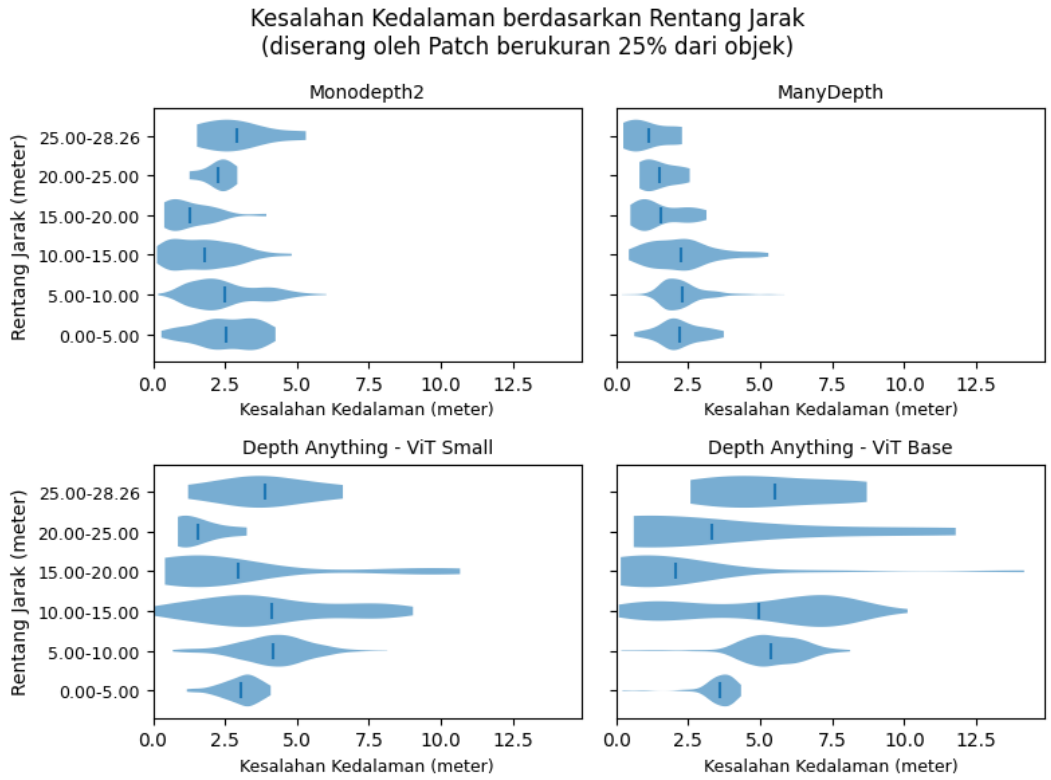
Gambar V.14 Galat relatif Depth Anything – ViT Small berdasarkan rentang jarak *patch*

Gambar V.14 menunjukkan galat relatif berdasarkan rentang jarak *patch* pada model Depth Anything – ViT Small. Dibandingkan dua model sebelumnya, Depth Anything – ViT Small memperlihatkan kerentanan yang lebih tinggi, ditunjukkan oleh sebaran galat yang bergeser ke nilai yang lebih tinggi pada seluruh rentang jarak yang ditampilkan. Galat terbesar muncul pada rentang jarak 0-10 meter, dengan pusat serta distribusi galat yang melampaui Monodepth2 maupun Manydepth. Penurunan galat mulai terlihat pada rentang 10-15 meter. Pada rentang di atas 15 meter, distribusi galat mulai menyempit dan rata-rata galat menurun, walaupun masih terdapat *outlier* pada rentang 15–20 meter.

Gambar V.15 menunjukkan galat relatif berdasarkan rentang jarak *patch* pada model Depth Anything – ViT Base. Depth Anything – ViT Base menjadi model yang paling rentan dibandingkan model lainnya terhadap serangan *patch* pada seluruh rentang jarak, ditunjukkan dengan sebaran galat serta rata-rata galat ke nilai yang melampaui model lainnya.



Gambar V.15 Galat relatif Depth Anything – ViT Base berdasarkan rentang jarak *patch*



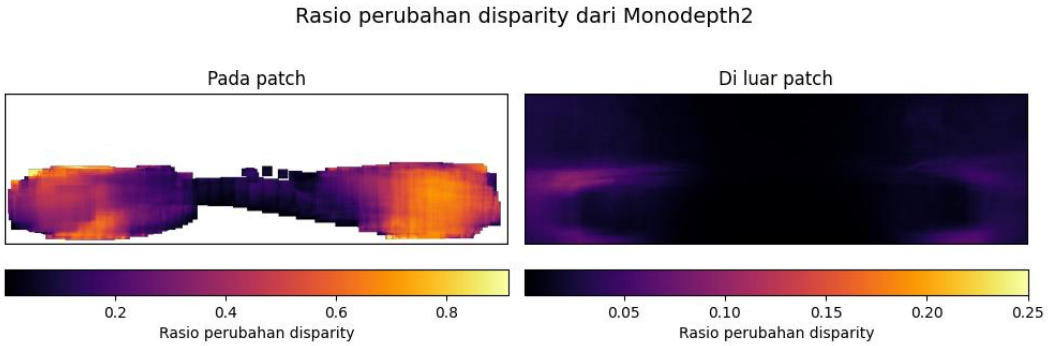
Gambar V.16 Distribusi kesalahan kedalaman sebenarnya berdasarkan rentang jarak

Untuk memberikan gambaran mengenai bagaimana kesalahan kedalaman terdistribusi terhadap kedalaman sebenarnya, gambar V.16 menampilkan sebaran galat kedalaman mutlak pada berbagai rentang jarak. Galat yang tertera pada gambar adalah galat saat model diserang oleh patch berukuran 25% dari ukuran objek. Pada Monodepth2, terdapat penyimpangan galat yang cukup besar pada rentang 0-15 meter, dengan pusat sebaran berada pada 2-2,5 meter serta galat maksimum yang mencapai 5 meter. Manydepth memperlihatkan pola serupa, namun pusat sebarannya cenderung lebih kecil serta sebarannya lebih menyempit seiring bertambahnya jarak.

Pada model DA-Small maupun DA-Base, tingkat kesalahan maupun variansinya lebih tinggi dibandingkan model berbasis CNN. Pada rentang 0-15 meter, pusat sebaran kedua model berada pada 3-5 meter, dengan variansi yang meningkat pada rentang 10-15 meter. Dalam rentang tersebut, DA-Small mencapai galat hingga 9 meter, sedangkan DA-Base mencapai 10 meter. Pada rentang 15-20 meter, besar galat kedua model menurun, namun tetap menghasilkan *outlier* hingga 11 meter pada DA-Small dan 14 meter pada DA-Base. Model DA-Small menunjukkan ketahanan terbaik pada rentang 20-25 meter, dengan pusat galat yang lebih rendah dibandingkan DA-Base maupun model CNN. Pada jarak lebih dari 25 meter, galat kedalaman pada kedua model Depth Anything kembali meningkat.

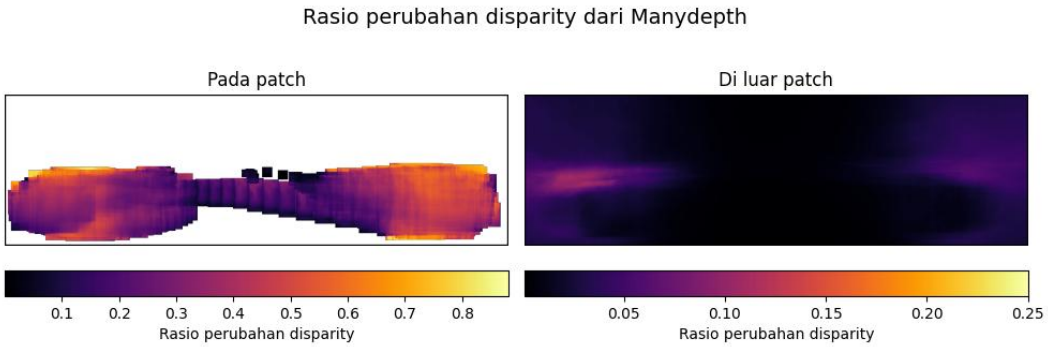
V.2.3 Kinerja Model berdasarkan Peletakan Patch

Kinerja model berdasarkan peletakan *patch* dijelaskan melalui tingkat perubahan *disparity* yang terjadi pada setiap model. Pembatasan rentang nilai pada *heatmap* rasio perubahan *disparity* di luar area *patch* dilakukan untuk mencegah *outlier* menutupi perubahan *disparity* lain yang lebih kecil, sehingga bagian citra dengan perubahan *disparity* yang lebih rendah tidak tampak gelap seluruhnya. Pada setiap visualisasi *heatmap* perubahan *disparity* di luar area *patch*, nilai perubahan *disparity* melebihi 25% ditampilkan sebagai warna kuning terang untuk menandai *outlier* tersebut. Seluruh pengujian *patch* pada subbab ini dilakukan dalam skenario *white-box*.



Gambar V.17 Rasio perubahan *disparity* dari Monodepth2

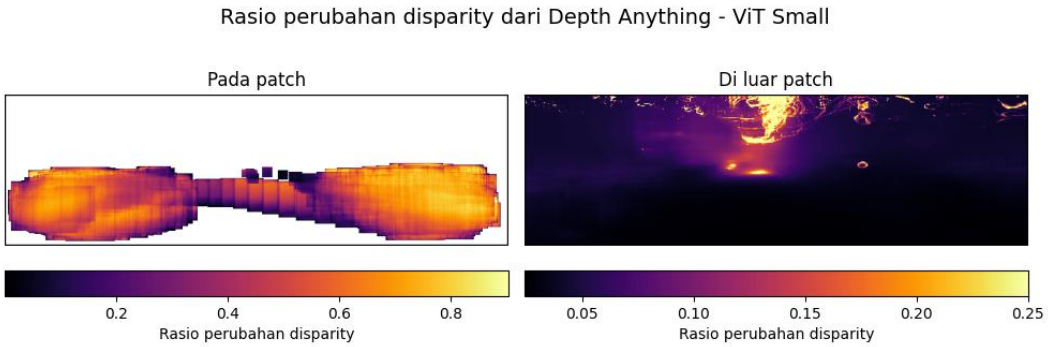
Visualisasi rasio perubahan *disparity* dari model Monodepth2 dapat dilihat pada gambar V.17. Pada Monodepth2, perubahan *disparity* pada bagian citra yang ditempelkan *patch* terlihat tidak merata. Bagian kanan citra mengalami perubahan *disparity* yang lebih tinggi dibandingkan bagian kiri, sedangkan bagian tengah citra relatif tidak terpengaruh. Pada bagian citra di luar *patch*, perubahan *disparity* cenderung rendah dan sebagian besar perubahan berada di bawah 10 persen. Walaupun demikian, bagian kiri pada area citra di luar *patch* menunjukkan perubahan yang sedikit lebih tinggi dibandingkan bagian kanan.



Gambar V.18 Rasio perubahan *disparity* dari Manydepth

Visualisasi rasio perubahan *disparity* dari model Manydepth dapat dilihat pada gambar V.18. Pada Manydepth, pola perubahan *disparity* pada area *patch* menunjukkan kemiripan dengan Monodepth2. Bagian kanan citra tetap lebih rentan terhadap gangguan dibandingkan bagian kiri, meskipun tingkat perubahannya sedikit lebih rendah dibandingkan Monodepth2. Bagian tengah citra pada model ini

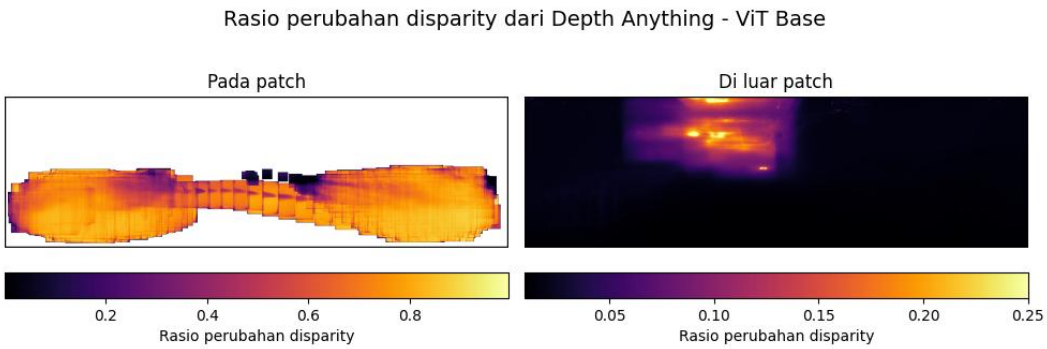
menunjukkan perubahan *disparity* yang sedikit lebih tinggi dibandingkan Monodepth2. Pola perubahan *disparity* pada area citra di luar *patch* juga serupa dengan Monodepth2.



Gambar V.19 Rasio perubahan *disparity* dari Depth Anything – ViT Small

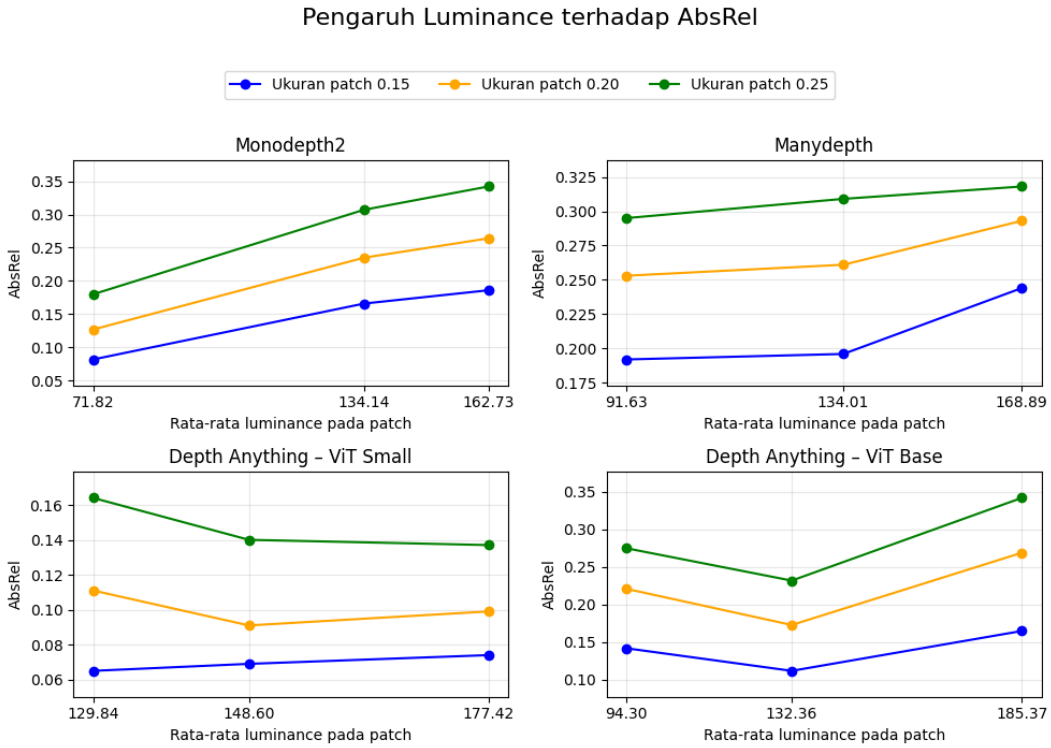
Visualisasi rasio perubahan *disparity* dari model Depth Anything – ViT Small dapat dilihat pada gambar V.19. Pada Depth Anything – ViT Small, perubahan *disparity* pada area *patch* terlihat lebih merata. Perubahan *disparity* yang signifikan terjadi pada sisi kanan, sisi kiri, serta sebagian area tengah bagian atas dari citra. Sementara itu, pengaruh serangan *patch* terhadap bagian bawah citra relatif lebih rendah. Serangan juga memberikan dampak yang cukup besar pada area di luar *patch*, terutama pada bagian tengah atas. Perubahan *disparity* di luar *patch* pada bagian bawah citra relatif lebih rendah dibandingkan Monodepth2 dan Manydepth. Namun, jika dirata-ratakan terhadap seluruh piksel pada area citra di luar *patch*, Depth Anything – ViT Small tetap lebih rentan dibandingkan kedua model yang telah disebutkan.

Visualisasi rasio perubahan *disparity* dari model Depth Anything – ViT Base dapat dilihat pada gambar V.20. Pada Depth Anything – ViT Base, perubahan *disparity* pada area citra yang ditempelkan *patch* terlihat lebih merata, dengan sedikit penurunan rasio pada bagian kiri tengah citra. Perubahan *disparity* pada area citra di luar *patch* terlihat lebih terlokalisasi dibandingkan model Depth Anything - ViT Small. Walaupun demikian, nilai perubahan tersebut tetap lebih besar dibandingkan model berbasis CNN.



Gambar V.20 Rasio perubahan *disparity* dari Depth Anything – ViT Base

V.2.4 Kinerja Model berdasarkan Kecerahan Patch



Gambar V.21 Kinerja model berdasarkan kecerahan *patch* (dalam AbsRel)

Hasil kinerja model berdasarkan kecerahan patch dapat dilihat pada gambar V.21. Seluruh pengujian *patch* pada subbab ini dilakukan dalam skenario *white-box*. Hasil pengujian menunjukkan bahwa serangan terhadap model berbasis CNN seperti Monodepth2 dan Manydepth semakin efektif seiring meningkatnya *luminance* dari *patch*. Pada Monodepth2, peningkatan galat pada semua ukuran *patch* selalu

berbanding lurus dengan peningkatan *luminance*. Manydepth menunjukkan pola peningkatan serupa, namun dengan tingkat kenaikan yang lebih landai. Efektivitas serangan pada Manydepth berada pada titik terendah ketika *luminance* dari *patch* berada pada nilai terendah, kemudian meningkat secara bertahap hingga *luminance* tertinggi.

Berbeda dengan model CNN, model berbasis *transformer* tidak memperlihatkan adanya korelasi yang jelas antara peningkatan *luminance* dengan meningkatnya efektivitas serangan. Depth Anything – ViT Small menjadi model yang paling tidak terpengaruh oleh perubahan *luminance*, dengan nilai AbsRel terendah sekitar 0,065 pada *luminance* terendah dengan ukuran *patch* 0,15. Pada ukuran *patch* 0,15, terdapat sedikit peningkatan efektivitas serangan seiring meningkatnya *luminance*. Namun, pada ukuran *patch* 0,2 hubungan antara *luminance* dan efektivitas serangan menjadi tidak linier. Sementara itu, pada ukuran *patch* 0,25, peningkatan *luminance* justru menurunkan efektivitas serangan.

Depth Anything – ViT Base juga tidak menunjukkan korelasi yang jelas antara *luminance* dengan efektivitas serangan. Pada semua ukuran patch, peningkatan *luminance* dari nilai rendah ke menengah menghasilkan penurunan nilai galat, tetapi peningkatan *luminance* dari nilai menengah ke tinggi kembali menaikkan galat tersebut.

V.3 Diskusi

Dalam subbab ini, akan dibahas beberapa temuan yang diamati berdasarkan hasil evaluasi. Pada skenario serangan secara *white-box*, model berbasis CNN cenderung menunjukkan ketahanan yang lebih baik dibandingkan model berbasis *transformer*. Perbedaan ini berkaitan dengan cara kedua arsitektur dalam memproses citra. Model berbasis *transformer* menggabungkan informasi secara global melalui mekanisme *self-attention*, sehingga setiap token dapat dipengaruhi oleh token lain pada seluruh area citra. Akibat hal tersebut, satu atau lebih token yang memuat *adversarial patch* berpotensi mengubah hasil prediksi pada banyak area sekaligus, termasuk area yang jauh dari posisi *patch* ditempatkan. Sebaliknya, model berbasis

CNN memiliki bidang reseptif dan kernel konvolusi yang hanya mengolah informasi pada bagian tertentu dari citra, sehingga pengaruh *adversarial patch* pada model berbasis CNN cenderung tetap terlokalisasi di sekitar tempat *patch* diletakkan.

Kondisi ini terlihat pada hasil evaluasi kinerja model berdasarkan variasi peletakan *patch*, pada pengamatan perubahan *disparity* di luar area *patch*. Pada model CNN, *patch* yang dibangkitkan dari model CNN umumnya menghasilkan perubahan rasio *disparity* di bawah 10% pada area di luar *patch*. Sebaliknya, *patch* yang dibangkitkan dari model berbasis *transformer* menimbulkan perubahan *disparity* yang menyebar ke berbagai bagian citra dan dapat mencapai rasio perubahan *disparity* hingga lebih dari 25%.

Pada skenario serangan secara *black-box*, efektivitas serangan antara Monodepth2 dan ManyDepth cenderung mirip karena keduanya menggunakan *backbone* yang serupa, yaitu ResNet-18. Patch yang dibangkitkan dari kedua model CNN tersebut menunjukkan keteralihan (*transferability*) serangan yang tinggi ketika menyerang sesama model CNN, dengan peningkatan nilai AbsRel hingga 0,24-0,26. Akan tetapi, ketika patch yang dibangkitkan dari model berbasis CNN digunakan untuk menyerang model berbasis transformer, peningkatan AbsRel yang dihasilkan jauh lebih kecil, yakni hanya berada pada kisaran 0,04–0,09.

Sementara itu, fenomena menarik terlihat pada Depth Anything karena varian ViT-S justru memiliki keteralihan serangan *patch* yang lebih tinggi serta keandalan yang lebih baik dibandingkan ViT-B. Patch yang dibangkitkan dari varian ViT-Small mampu meningkatkan nilai AbsRel pada model lain hingga sekitar 0,20–0,22, sedangkan patch dari ViT-Base menunjukkan keteralihan serangan terburuk dengan peningkatan AbsRel yang hanya berada pada kisaran 0,06–0,15.

Padahal dalam keadaan *benign*, ViT-B menunjukkan kinerja yang lebih baik dibandingkan ViT-S. Temuan ini bertentangan dengan beberapa penelitian sebelumnya, yang menunjukkan bahwa model ViT dengan jumlah parameter yang

lebih banyak cenderung memiliki *adversarial robustness* yang lebih tinggi (Mahmood dkk. 2021).

Pada beberapa lokasi peletakan *patch*, seperti pada area tengah citra, peningkatan nilai *disparity* terlihat sangat kecil. Kondisi ini bukan disebabkan oleh kegagalan serangan *adversarial patch*, melainkan karena nilai *disparity* pada area tersebut sejak awal (kondisi *benign*) memang mendekati nol. Bagian tengah citra umumnya ditempati objek yang berada pada jarak yang jauh, sehingga nilai *disparity* yang dihasilkan model cenderung kecil. Sebagai catatan, *adversarial patch* dalam tugas akhir ini memang ditujukan agar model mengeluarkan *disparity* mendekati nol.

Selain perbedaan cakupan lokasi yang dipengaruhi *patch*, terdapat juga perbedaan cara kedua arsitektur bereaksi terhadap komponen *luminance* dari *adversarial patch*. Model berbasis CNN memproses nilai intensitas piksel secara langsung melalui operasi konvolusi dan cenderung mengandalkan tekstur citra dalam proses inferensi (Geirhos dkk. 2018). Pada *patch* dengan *luminance* rendah, tekstur adversarial menjadi kurang menonjol sehingga gangguan yang masuk ke dalam *feature map* relatif melemah dan menyebabkan efektivitas serangan menurun.

Sebaliknya, semakin tinggi *luminance*, tekstur adversarial terlihat semakin jelas sehingga *feature map* lebih mudah terganggu dan efektivitas serangan meningkat. Di sisi lain, mekanisme *self-attention* tidak membuat *transformer* terpengaruh terhadap tekstur *patch*. *Transformer* memproses citra dalam bentuk token dan mengandalkan *self-attention* yang mencakup seluruh area citra, sehingga pengambilan keputusan model lebih dipengaruhi oleh hubungan antar-token secara menyeluruh.

BAB VI

KESIMPULAN DAN SARAN

VI.1 Kesimpulan

Tabel VI.1 Performa terburuk setiap model dalam setiap skenario

Model	**Skenario 1 (*Benign*)**	**Skenario 2 (*White-Box)***	**Skenario 3 (*Black-Box*)**
	AbsRel		
Monodepth2	0,067	0,335	0,246
Manydepth	0,061	0,306	0,26
Depth Anything – ViT Small	0,11	0,532	0,109
Depth Anything – ViT Base	0,079	0,664	0,225

Tabel VI.1 menunjukkan performa terburuk seluruh model MDE pada tiga skenario evaluasi, yang diukur menggunakan metrik *absolute relative error* (AbsRel). Setiap sel pada tabel menunjukkan galat tertinggi yang didapat oleh model tertentu ketika diuji pada skenario tertentu, berdasarkan hasil evaluasi pada subbab V.2.1. Secara umum, seluruh model menunjukkan peningkatan galat pada skenario *white-box* dan *black-box* dibandingkan dengan skenario *benign*, dengan tingkat peningkatan galat yang bervariasi antar model.

Rumusan masalah dari tugas akhir ini adalah menentukan tingkat ketahanan model *monocular depth estimation* (MDE) terhadap *adversarial patch*, serta mengidentifikasi bagaimana karakterisitik *adversarial patch* memengaruhi

efektivitas serangan terhadap model tersebut. Hasil evaluasi menunjukkan bahwa rumusan masalah dari tugas akhir ini telah terjawab, dengan beberapa kesimpulan sebagai berikut:

1. pemeriksaan *adversarial robustness* pada suatu model MDE dapat dilakukan dengan membandingkan performa model dengan menggunakan metrik AbsRel (galat relatif mutlak), *threshold accuracy*, ataupun dengan memeriksa langsung hasil keluaran *disparity* dari model;
2. pembangkitan *adversarial patch* yang dapat mengelabui model MDE berhasil dilakukan dan memenuhi kriteria keberhasilan pertama, dengan hasil galat kedalaman yang mencapai 66% pada skenario *white-box* dan mencapai 26% pada skenario *white-box*;
3. kriteria keberhasilan kedua terpenuhi karena *adversarial patch* berhasil mencapai konvergensi yang lebih cepat dibandingkan penelitian sebelumnya, dengan hanya membutuhkan 640 iterasi (20 epoch dengan *batch size* 32);
4. model MDE berbasis CNN menunjukkan ketahanan yang lebih tinggi terhadap serangan *adversarial patch* dalam skenario *white-box* dibandingkan model berbasis *transformer*, namun model MDE berbasis *transformer* memiliki *transferability* yang lebih tinggi pada skenario *black-box*;
5. karakteristik *adversarial patch* memiliki pengaruh yang berbeda terhadap model berbasis CNN dan *transformer*, baik akibat peletakan maupun luminansi.

VI.2 Saran

Terdapat beberapa saran yang dapat dipertimbangkan untuk melakukan penelitian selanjutnya, yaitu sebagai berikut:

1. mengeksplorasi *adversarial training* sebagai strategi pertahanan untuk meningkatkan ketahanan model *monocular depth estimation*, baik terhadap serangan *adversarial patch* maupun serangan adversarial lainnya;

2. menganalisis proses internal model terhadap serangan adversarial melalui visualisasi peta aktivasi maupun peta *attention*;
3. mengevaluasi dampak serangan adversarial pada model MDE pada tugas lanjutan dalam sistem persepsi kendaraan otonom, seperti deteksi objek tiga dimensi dan rekonstruksi lingkungan;
4. menggunakan metode lain dalam melakukan pembangkitan *adversarial patch* yang tampak alami, misalnya melalui pendekatan *style transfer* ataupun pemanfaatan model generatif lainnya.

DAFTAR PUSTAKA

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2018). Synthesizing Robust Adversarial Examples. *Proceedings of the 35th International Conference on Machine Learning*, 284–293. https://proceedings.mlr.press/v80/athalye18b.html

Brown, T. B., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (2018). Adversarial patch. *arXiv preprint arXiv:1712.09665*. https://doi.org/10.48550/arXiv.1712.09665

Burns, M. (2019, April 22). "Anyone Relying on Lidar is Doomed," Elon Musk says. TechCrunch. https://techcrunch.com/2019/04/22/anyone-relying-on-lidar-is-doomed-elon-musk-says/

Certad, N., Morales-Alvarez, W., Novotny, G., & Olaverri-Monreal, C. (2022). JKU-ITS Automobile for Research on Autonomous Vehicles. *18th International Conference on Computer Aided Systems Theory (EUROCAST 2022)*, 90–91. https://doi.org/10.1007/978-3-031-25312-6_38

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. *SPSS inc*, *9*(13), 1-73.

Cheng, Z., Liang, J., Choi, H., Tao, G., Cao, Z., Liu, D., & Zhang, X. (2022). Physical Attack on Monocular Depth Estimation with Optimal Adversarial Patches. *European Conference on Computer Vision*, 514–532. https://doi.org/10.1007/978-3-031-19839-7_30

Crowe, S. (2019, April 25). *Researchers back Tesla's non-LiDAR approach to self-driving cars.* The Robot Report. https://www.therobotreport.com/researchers-back-teslas-non-lidar-approach-to-self-driving-cars/

Dikmen, M., & Burns, C. (2017). Trust in autonomous vehicles: The case of Tesla Autopilot and Summon. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1093–1098. https://doi.org/10.1109/SMC.2017.8122757

Dong, X., Garratt, M. A., Anavatti, S. G., & Abbass, H. A. (2022). Towards Real-Time Monocular Depth Estimation for Robotics: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(10), 16940–16961. https://doi.org/10.1109/TITS.2022.3160741

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*. https://doi.org/10.48550/arXiv.2010.11929

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *arXiv preprint arXiv:1406.2283*. https://doi.org/10.48550/arXiv.1406.2283

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Tramer, F., Prakash, A., Kohno, T., & Song, D. (2018). Physical Adversarial Examples for Object Detectors. *arXiv preprint arXiv:1807.07769*. https://doi.org/10.48550/arXiv.1807.07769

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*. https://doi.org/10.48550/arXiv.1811.12231

Ginting, R., Patmasari, R., & Aulia, S. (2018). Sistem Orientasi Objek Dengan Metode Stereo Vision Berbasis Raspberry Pi. *Journal Research and Development (ITJRD)*, *3*(1). https://doi.org/10.25299/itjrd.2018.vol3(1).xxxx

Godard, C., Aodha, O. mac, Firman, M., & Brostow, G. (2019). Digging Into Self-Supervised Monocular Depth Estimation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3827–3837. https://doi.org/10.1109/ICCV.2019.00393

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*. https://doi.org/10.48550/arXiv.1412.6572

Guesmi, A., Hanif, M. A., Alouani, I., Ouni, B., & Shafique, M. (2024). SSAP: A Shape-Sensitive Adversarial Patch for Comprehensive Disruption of Monocular Depth Estimation in Autonomous Navigation Applications. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2786–2793. https://doi.org/10.1109/IROS58592.2024.10802252

Hall, K., & Shepardson, D. (2025, Februari 4). General Motors acquires full ownership of Cruise autonomous business. *Reuters*. https://www.reuters.com/business/autos-transportation/general-motors-acquires-full-ownership-cruise-autonomous-business-2025-02-04/

Hamad, L., Khan, M. A., & Mohamed, A. (2024). Object Depth and Size Estimation Using Stereo-Vision and Integration with SLAM. *IEEE Sensors Letters*, *8*(4), 1–4. https://doi.org/10.1109/LSENS.2024.3367956

Heaton, J. (2018). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines*, *19*(1–2), 305–307. https://doi.org/10.1007/s10710-017-9314-z

Huang, Z., Ali, M. A., Nukman, Y., Xu, H. L., Zhang, S., Chen, H., & Alkhedher, M. (2025). A Systematic Review of Monocular Depth Estimation for

Autonomous Driving: Methods and Dataset Benchmarking. *Results in Engineering*, 105359.

Jankowicz, M. (2024, Februari 12). A crowd set fire to a Waymo taxi in San Francisco, as tensions about driverless tech grow. Business Insider. https://www.businessinsider.com/san-francisco-waymo-driverless-car-torched-crowd-2024-2

Kok, K. Y., & Rajendran, P. (2020). A Review on Stereo Vision Algorithm: Challenges and Solutions. *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, *13*(2), 112–128. https://doi.org/10.37936/ecti-cit.2019132.194324

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lu, J., Sibai, H., Fabry, E., & Forsyth, D. (2017). NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. *arXiv preprint arXiv:1707.03501*. https://doi.org/10.48550/arXiv.1707.03501

Luciano, E., Cattaneo, M., & Kenett, R. (2023). Adversarial AI in Insurance: Pervasiveness and Resilience. *arXiv preprint arXiv:2301.07520*. https://doi.org/10.48550/arXiv.2301.07520

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*. https://doi.org/10.48550/arXiv.1706.06083

Mahmood, K., Mahmood, R., & van Dijk, M. (2021). On the Robustness of Vision Transformers to Adversarial Examples. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 7818–7827. https://doi.org/10.1109/ICCV48922.2021.00774

Martinez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernandez-Orallo, J., Kull, M., Lachiche, N., Ramirez-Quintana, M. J., & Flach, P. (2021). CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science

Trajectories. *IEEE Transactions on Knowledge and Data Engineering*, *33*(8), 3048–3061. https://doi.org/10.1109/TKDE.2019.2962680

Nordhoff, S. (2024). Resistance towards autonomous vehicles (AVs). *Transportation Research Interdisciplinary Perspectives*, *26*, 101117. https://doi.org/10.1016/j.trip.2024.101117

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*. https://doi.org/10.48550/arXiv.1511.08458

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The Limitations of Deep Learning in Adversarial Settings. *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 372–387. https://doi.org/10.1109/EuroSP.2016.36

Pramoditha, R. (2020, April 24). *Overview of a neural network's learning process.* Medium.https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa

Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 12159–12168. https://doi.org/10.1109/ICCV48922.2021.01196

Reda, M., Onsy, A., Haikal, A. Y., & Ghanbari, A. (2024). Path planning algorithms in the autonomous driving system: A comprehensive review. *Robotics and Autonomous Systems*, *174*, 104630. https://doi.org/10.1016/j.robot.2024.104630

Sever, T., & Contissa, G. (2024). Automated driving regulations – where are we now?. *Transportation Research Interdisciplinary Perspectives*, *24*, 101033. https://doi.org/10.1016/j.trip.2024.101033

Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2017). On a Formal Model of Safe and Scalable Self-driving Cars. *arXiv preprint arXiv:1708.06374*. https://doi.org/10.48550/arXiv.1708.06374

Sharif, M., Bhagavatula, S., Bauer, L., & Reiter, M. K. (2016). Accessorize to a Crime. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 1528–1540. https://doi.org/10.1145/2976749.2978392

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651. https://doi.org/10.1109/TPAMI.2016.2572683

Szandała, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. *Bio-inspired Neurocomputing*, 203-224. https://doi.org/10.1007/978-981-15-5495-7

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. https://doi.org/10.48550/arXiv.1312.6199

Tosi, F., Aleotti, F., Poggi, M., & Mattoccia, S. (2019). Learning Monocular Depth Estimation Infusing Traditional Stereo Knowledge. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9791–9801. https://doi.org/10.1109/CVPR.2019.01003

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2023). Attention Is All You Need. *arXiv preprint arXiv:1706.03762*. https://doi.org/10.48550/arXiv.1706.03762

Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., & Weinberger, K. Q. (2019). Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8437–8445. https://doi.org/10.1109/CVPR.2019.00864

Watson, J., mac Aodha, O., Prisacariu, V., Brostow, G., & Firman, M. (2021). The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth.

2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1164–1174. https://doi.org/10.1109/CVPR46437.2021.00122

White, T. E., Rojas, B., Mappes, J., Rautiala, P., & Kemp, D. J. (2017). Colour and luminance contrasts predict the human detection of natural stimuli in complex visual environments. *Biology Letters*, *13*(9), 20170375. https://doi.org/10.1098/rsbl.2017.0375

Yamanaka, K., Matsumoto, R., Takahashi, K., & Fujii, T. (2020). Adversarial Patch Attacks on Monocular Depth Estimation Networks. *IEEE Access*, 8, 179094-179104. https://doi.org/10.1109/ACCESS.2020.3027372

Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10371–10381. https://doi.org/10.1109/CVPR52733.2024.00987

Zhang, Z., Zhu, X., Li, Y., Chen, X., & Guo, Y. (2020). Adversarial attacks on monocular depth estimation. *arXiv preprint arXiv:2003.10315*. https://doi.org/10.48550/arXiv.2003.10315

Zheng, T., Hu, J., Tan, R., Zhang, Y., He, Y., & Luo, J. (2024). π-Jack:Physical-World Adversarial Attack on Monocular Depth Estimation with Perspective Hijacking. *33rd USENIX Security Symposium (USENIX Security 24)*, 7321-7338.

Zolfi, A., Kravchik, M., Elovici, Y., & Shabtai, A. (2021). The Translucent Patch: A Physical and Universal Attack on Object Detectors. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 15232-15241. https://doi.org/10.1109/CVPR46437.2021.01498

LAMPIRAN A
SOURCE CODE

A.1 Source Code

Source code yang digunakan pada tahap *data understanding* dapat diakses pada tautan https://colab.research.google.com/drive/1Owh0I4kIG3m1koaxaIOmwnNV-L4twooJ?usp=sharing. *Source code* yang digunakan pada tahap *data preparation*, *modeling*, dan *evaluation* dapat diakses pada *repository* GitHub melalui tautan https://github.com/ardhanapw/Adversarial-Patch-Monocular-Depth-Estimation/tree/main.