IMPLEMENTASI ALGORITMA *DETECTION TRANSFORMER* (DETR) DALAM MENDETEKSI PARKIR KENDARAAN TIDAK TERTIB DI AREA KAMPUS

LAPORAN TUGAS AKHIR

Proposal ini disusun untuk Memenuhi Salah Satu Syarat Memperoleh Gelar
Sarjana Strata 1 (S1) pada Program StudiTeknik Informatika
Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang

DISUSUN OLEH:

ADELLA MAULANA ANNUR RAMADHAN
32602100009

FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM SULTAN AGUNG
SEMARANG
2025

FINAL PROJECT

IMPLEMENTATION OF THE DETECTION TRANSFORMER (DETR) ALGORITHM FOR DETECTING IMPROPER VEHICLE PARKING IN CAMPUS AREAS

Proposed to complete the requirement to obtain a bachelor's degree (S1) at
Informatics Engineering Departement of Industrial Technology Faculty
Sultan Agung Islamic University

Arranged By:

ADELLA MAULANA ANNUR RAMADHAN
32602100009

MAJORING OF INFORMATICS ENGINEERING
INDUSTRIAL TECHNOLOGY FACULTY
SULTAN AGUNG ISLAMIC UNIVERSITY
SEMARANG
2025

LEMBAR PENGESAHAN
TUGAS AKHIR

Implementasi Algoritma *Detection Transformer* (DETR) Dalam Mendeteksi Parkir Kendaraan Tidak Tertib di Area Kampus

ADELLA MAULANA ANNUR RAMADHAN
32602100009

Telah dipertahankan di depan tim penguji ujian sarjana tugas akhir
Program Studi Teknik Informatika
Universitas Islam Sultan Agung
Pada tanggal:

TIM PENGUJI UJIAN SARJANA :

Bagus Satrio Waluyo Poetro, S.Kom., M.Cs
NIK. 210616051
(Ketua Penguji)
.... 24/11/2025

Sam Farisa Chaerul Haviana, ST., M.Kom
NIK. 210615046
(Anggota Penguji)
.... 28/11/2025

Imam Much Ibnu Subroto, St., MSc., Ph.D
NIK. 210600017
(Pembimbing)
.... 01/12-2025

Semarang,
Mengetahui,
Kaprodi Teknik Informatika
Universitas Islam Sultan Agung

Moch Taufik, S.T., M.IT
NIK. 210604034

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini:

Nama : Adella Maulana Annur Ramadhan

NIM : 32602100009

Judul Tugas Akhir : Implementasi Algoritma *Detection Transformer* (DETR) Dalam Mendeteksi Parkir Kendaraan Tidak Tertib di Area Kampus

Bahwa dengan ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apabila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 13 November 2025

Yang menyatakan,

10000
SEPULUH RIBU RUPIAH
METERAI TEMPEL
CEFC9ANX167455804

Adella Maulana Annur Ramadhan

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan di bawah ini:

Nama : Adella Maulana Annur Ramadhan
NIM : 32602100009
Program Studi : Teknik Informatika
Fakultas : Teknologi Industri
Alamat Asal : Semarang

Dengan ini menyatakan karya ilmiah berupa tugas akhir dengan judul : Implementasi Algoritma *Detection Transformer* (DETR) Dalam Mendeteksi Parkir Kendaraan Tidak Tertib di Area Kampus.

Menyetujui menjadi hak milik Universitas Islam Sultan Agung serta memberikan hak bebas royalti non-Eksklusif untuk disimpan, dialihmediakan, dikelola dan pangkalan data dan dipublikasikan di internet dan media lain untuk kepentingan akademis selama tetap menyantumkan nama penulis sebagai pemilik hak cipta. Pernyataan ini saya buat dengan sungguh-sungguh. Apabila di kemudian hari terbukti ada pelanggaran Hak Cipta/Plagiarisme dalam karya ilmiah ini, maka segala bentuk tuntutan hukum yang timbul akan saya tanggung secara pribadi tanpa melibatkan Universitas Islam Sultan Agung.

Semarang, 13 November2025

Yang menyatakan,

METERAI TEMPEL
4C6BAANX167455809

Adella Maulana Annur Ramadhan

KATA PENGANTAR

Dengan mengucap syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul “Implementasi Algoritma *Detection Transformer* (DETR) Dalam Mendeteksi Parkir Kendaraan Tidak Tertib di Area Kampus” dengan baik.

Tugas akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, berupa materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada:

1. Rektor Universitas Islam Sultan Agung Bapak Prof. Dr. H. Gunarto, S.H., M.H yang telah mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Ir. Hj. Novi Marlyana, S.T., M.T., IPU., ASEAN Eng.
3. Kaprodi Teknik Informatika Bapak Moch.Taufik, S.T., M.IT.
4. Dosen pembimbing penulis Bapak Imam Much Ibnu Subroto, St., MSc., Ph.D yang telah meluangkan waktu dan memberi ilmu dalam penyusunan tugas akhir.
5. Orang tua penulis tercinta Bapak Nuryadi dan Ibu Anik Harwiyanti yang telah memberikan doa serta dorongan baik moral maupun material.
6. Adik Penulis Saudari Sekarinakit Saiskapitu dan Saudara Kanugrahan Nurani Sakti yang menjadi alasan kuat penulis selama masa studi.
7. Saudari Amanda Ardita Amelia yang telah memberikan semangat, dukungan, dan motivasi kepada penulis dalam proses menyelesaikan tuga akhir ini
8. Teman seperjuangan yang telah menjadikan perjalanan perkuliahan ini menjadi hal yang tak terlupakan dalam kehidupan penulis.
9. Dan kepada semua pihak yang tidak dapat disebutkan satu persatu.

Semarang, 13 November 2025

Adella Maulana Annur Ramadhan

DAFTAR ISI

LAPORAN TUGAS AKHIR i

LEMBAR PENGESAHAN TUGAS AKHIR iii

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR iv

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH v

KATA PENGANTAR vi

DAFTAR ISI vii

DAFTAR GAMBAR x

DAFTAR TABEL xi

ABSTRAK xii

BAB I PENDAHULUAN 1

- 1.1. Latar Belakang 1
- 1.2. Perumusan Masalah 2
- 1.3. Pembatasan Masalah 3
- 1.4. Tujuan 3
- 1.5. Manfaat 3
- 1.6. Sistematika Penulisan 4

BAB II TINJAUAN PUSTAKA DAN DASAR TEORI 6

- 2.1. Tinjauaan Pustaka 6
- 2.2. Dasar Teori 9
  - 2.2.1. Tempat Parkir 9
  - 2.2.2. *Computer Vision* 10
  - 2.2.3. *Machine Learning* 11
  - 2.2.4. *Deep Learning* 11
  - 2.2.5. Deteksi Objek 12
  - 2.2.6. *Transformer* 13
  - 2.2.7. *Detection Transformer* (DETR) 15

BAB III METODOLOGI PENELITIAN 20

- 3.1. Pengumpulan Data 20
- 3.2. *Pre-Processing* 21
  - 3.2.1. Anotasi Objek 21

- 3.2.2. Augmentasi 22
- 3.2.3. Split *Dataset*......22
- 3.2.4. *Set-Up Environment* 22
- 3.2.5. Membuat *Dataloader*......23
- 3.2.6. Visualisasi Data......23
- 3.3. *Training Model*......23
  - 3.3.1. Arsitektur *Model* 23
  - 3.3.2. Inisialisasi dan *Hyperparameter* 24
  - 3.3.3. Strategi *Training* 25
  - 3.3.4. Infrasruktur Komputasi......26
  - 3.3.5. *Output Training*......26
- 3.4. Pengujian......26
- 3.5. Evaluasi......27

BAB IV HASIL DAN ANALISIS PENELITIAN......30

- 4.1. Hasil Pengunpulan Data......30
- 4.2. Hasil *Pre-Processing*......31
  - 4.2.1. Hasil Anotasi Objek 31
  - 4.2.2. Hasil Augmentasi 32
  - 4.2.3. Hasil Split *Dataset*......36
  - 4.2.4. Hasil *Set-Up Environmnet*......36
  - 4.2.5. Hasil *Dataloader*......38
  - 4.2.6. Hasil Visualisasi......38
- 4.3. Hasil *Training* Model 40
  - 4.3.1. Arsitektur Model......40
  - 4.3.2. Inisialisasi dan *Hyperparameter* 40
  - 4.3.3. Strategi *Training* 40
  - 4.3.4. Komputasi......41
  - 4.3.5. Output Training......41
- 4.4. Hasil Pengujian 49
- 4.5. Hasil Evaluasi 60

BAB V KESIMPULAN DAN SARAN 64

- 5.1 Kesimpulan 64

5.2 Saran 64

DAFTAR PUSTAKA

DAFTAR GAMBAR

Gambar 2. 1 Tempat Parkir 10
Gambar 2. 2 Kendaraan yang parkir tidak tertib 10
Gambar 2. 3 Sejarah Perkembangan Deteksi Objek(Arkin dkk., 2023) 13
Gambar 2. 4 Arsitektur pada *Transformer* 14
Gambar 2. 5 Prinsip Kerja Algotitma DETR 15
Gambar 2. 6 Alur dalam model DETR 16
Gambar 3. 1 Alur Penelitian 20
Gambar 3. 2 Tampilan Roboflow 21
Gambar 3. 3 Alur Pengujian 27
Gambar 4. 1 Data yang Terkumpul 30
Gambar 4. 2 Anotasi dengan Class Proper 31
Gambar 4. 3 Anotasi dengan Class Improper 32
Gambar 4. 4 *Horizontal Flip* 32
Gambar 4. 5 90° *Rotate* 33
Gambar 4. 6 *Rotation* +15° 34
Gambar 4. 7 *Rotation* -15° 34
Gambar 4. 8 *Saturation* 35
Gambar 4. 9 *Brightness* 35
Gambar 4. 10 Struktur *Folder Dataset* 38
Gambar 4. 11 Visusalisasi Data *Class Proper* 39
Gambar 4. 12 Visualisasi Data *Class Improper* 39
Gambar 4. 13 *Loss Curve Epoch* 20 43
Gambar 4. 14 Hasil *Confusion Matrix Epoch* 20 43
Gambar 4. 15 *Loss Curve Epoch* 30 44
Gambar 4. 16 Hasil *Confusion Matrix Epoch* 30 45
Gambar 4. 17 *Loss Curve Epoch* 40 46
Gambar 4. 18 Hasil *Confusion Matrix Epoch* 40 46
Gambar 4. 19 *Loss Curve Epoch* 40 47
Gambar 4. 20 Hasil *Confusion Matrix Epoch* 70 48
Gambar 4. 21 Tampilan implementasi pada streamlit 49
Gambar 4. 22 Fitur pada implementasi model 49
Gambar 4. 23 Fitur *treshold* 49
Gambar 4. 24 Fitur *upload file* pada *Inference* gambar dan video 50
Gambar 4. 25 Tampilan hasil pada fitur gambar 50
Gambar 4. 26 Tampilan hasil pada fitur video 50

DAFTAR TABEL

Tabel 1. 1 Sistematika Penulisan....4

Tabel 4. 1 *Split Data* 36
Tabel 4. 2 Penggunaan *Library* 37
Tabel 4. 3 Hasil Distribusi Dataloader 38
Tabel 4. 4 *Hyperparameter epoch* 20....41
Tabel 4. 5 *Hyperparameter epoch* 30....44
Tabel 4. 6 *Hyperparameter epoch* 40....45
Tabel 4. 7 *Hyperparameter epoch* 70....47
Tabel 4. 8 Komposisi *Hyperparameter* untuk Pengujian....48
Tabel 4. 9 Tabel Hasil Pengujian....51
Tabel 4. 10 Hasil Evaluasi....62

ABSTRAK

Permasalahan parkir kendaraan tidak tertib di lingkungan kampus menjadi isu yang berdampak pada kenyamanan, aksesibilitas, serta keselamatan pengguna jalan. Untuk mengatasi hal tersebut, penelitian ini mengimplementasikan algoritma *Detection Transformer* (DETR) dengan *backbone* ResNet-50 sebagai model deteksi objek untuk mengidentifikasi kendaraan yang parkir tidak sesuai aturan. *Dataset* yang digunakan berjumlah 583 citra dari Roboflow, yang kemudian melalui tahap *pre-processing* dan augmentasi sebelum dilatih dengan konfigurasi *hyperparameter* tertentu. Pada evaluasi model dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, dan *f1-score* pada berbagai nilai *threshold*. Hasil penelitian menunjukkan bahwa pada *threshold* 0,7 model mencapai performa terbaik dengan *accuracy* sebesar 92%, *precision* 96,7%, *recall* 90,9%, dan *f1-score* 93,7%. Nilai *recall* dan *f1-score* menurun secara signifikan ketika *threshold* dinaikkan, meskipun precision meningkat. Hal ini mengindikasikan bahwa model menjadi terlalu selektif sehingga banyak deteksi positif yang hilang. Dengan demikian, *threshold* 0,7 direkomendasikan sebagai nilai optimal untuk implementasi model ini karena memberikan keseimbangan terbaik antara sensitivitas dan ketepatan.

Kata kunci: Detection Transformer, ResNet-50, Deteksi Kendaraan, Parkir Tidak Tertib, *Computer Vision*.

ABSTRACT

The problem of improper vehicle parking on campus significantly affects comfort, accessibility, and road user safety. To address this issue, this study implements the Detection Transformer (DETR) algorithm with a ResNet-50 backbone as an object detection model to identify vehicles parked against regulations. The dataset consists of 583 images collected from Roboflow, which were then subjected to preprocessing and augmentation before being trained with specific setting hyperparameter configurations. Model evaluation was conducted using accuracy, precision, recall, and f1-score metrics across various threshold values. The results show that at a 0.7 threshold, the model achieved its best performance with 92% accuracy, 96.7% precision, 90,9% recall, and a 93.7% f1-score. Recall and f1-score decreased significantly when the threshold increased, although precision improved. This indicates the model became overly selective, resulting in missed positive detections. Therefore, a 0.7 threshold is recommended as the optimal value, providing the best balance between sensitivity and precision.

Keywords: Detection Transformer, ResNet-50, Vehicle Detection, Improper Parking, Computer Vision.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi mengalami kemajuan yang sangat pesat dan bertransformasi menjadi salah satu pendorong utama bagi perubahan sosial, ekonomi, dan industri di seluruh dunia. Salah satu satu contoh perkembangan teknologi tersebut adalah kecerdasan buatan atau *Artificial Intelligence* (AI). Salah satu cabang utama dari AI yang mengalami perkembangan pesat adalah *Computer Vision*, yang memungkinkan komputer untuk "melihat" dan memahami dunia disekitarnya, mirip dengan kemampuan penglihatan manusia. *Computer Vision* telah mengalami revolusi besar dalam cara sistem komputer dapat menganalisis, mengenali, dan menginterpretasi gambar dan video. *Computer Vision* memiliki banyak bidang, salah satunya adalah deteksi objek. Ini bekerja dengan mengenali objek yang ada pada gambar dan letaknya. Deteksi objek dapat digunakan untuk berbagai tujuan, seperti untuk keamanan, informasi, pengawasan, tingkat produktivitas, dan sebagainya.(Dompeipen dkk., 2021)

Bidang dari *Computer Vision* yaitu *Object Detection* dapat digunakan juga untuk pendeteksian objek berupa kendaraan. Kendaraan sudah menjadi kebutuhan pokok setiap orang untuk menjalani kehidupan sehari-hari. Banyaknya pemilik kendaraan tentunya membutuhkan lahan parkir yang banyak juga. Dengan adanya lahan parkir yang terbatas mengakibatkan pemilik kendaraan yang memarkirkan kendaraaan tidak sesuai dengan tempatnya terutama di area kampus. Permasalahan parkir kendaraan sembarangan di area kampus menjadi isu yang signifikan. Banyaknya kendaraan yang memasuki area kampus, baik dari mahasiswa, dosen, maupun pengunjung sering kali melebihi kapasitas area parkir yang tersedia. Akibatnya, banyak kendaraan yang parkir di tempat tidak semestinya, seperti di trotoar, jalan kampus, atau area yang dilarang untuk parkir. Serta adapun kendaraan yang parkir tidak pada tempatnya atau dengan posisi salah, sehingga memakan banyak tempat pada area parkir. Hal ini tidak hanya mengganggu estetika dan kenyamanan lingkungan, tetapi juga dapat menyebabkan masalah keselamatan, aksesbilitas, dan kerusakan infrastruktur(Dompeipen dkk., 2021).

Pada umunya deteksi kendaraan yang parkir tidak teritb mengandalkan patrol manual atau sistem pengawasan yang kurang efesien. Dengan perkembangan teknologi khusunya dalam bidang *Artificial Inteligence* dan *Computer Vision*, muncul peluang untuk menerapkan metode yang lebih canggih. Algoritma *Detection Transformer* (DETR) merupakan salah satu inovasi terkini yang menunjukkan kemampuan tinggi dalam mendeteksi objek dalam gambar dengan akurasi yang baik. Implementasi algoritma DETR untuk mendeteksi parkir sembarangan di area kampus tidak hanya dapat meningkatkan efektivitas, tetapi juga memberikan data analitis yang berguna untuk perencanaan dan pengelolaan ruang parkir yang lebih baik (Rosnita dkk., 2024).

Oleh karena itu, penelitian ini ingin mengeksplorasi potensi penggunaan algoritma DETR dalam mendeteksi kendaraan yang diparkir sembarangan, serta dampak positifnya terhadap manajemen parkir di lingkungan kampus. Dimana peneliti mengangkat judul "Implementasi Algoritma *Detection Transformer* (DETR) dalam Mendeteksi Parkir Kendaraan Tidak Tertib di Area Kampus". Adapun tujuan dari penelitian ini adalah dapat memaksimalkan penggunaan algoritma *Detection Transformer* untuk mendeteksi dan klasifikasi kendaraan yang diparkir sembarangan di area kampus.

1.2. Perumusan Masalah

Berdasarkan latar belakang permasalahan, maka dapat diidentifikasikan permasalahan yang akan dibahas sebagai berikut:

1. Bagaimana sistem dapat mengola *dataset* kendaraan yang terparkir tidak tertib dengan menggunakan algoritma *Detection Transformer* (DETR)?

2. Bagaimana efektivitas algoritma *Detection Transformer* (DETR) dalam mendeteksi kendaraan yang diparkir tidak tertib?

1.3. Pembatasan Masalah

Adapun pembatasan masalah dalam penulisan proposal ini digunakan untuk mencegah perluasan atau penyimpangan darin inti permasalahan, sehingga dapat lebih berfokus. Oleh karena itu, berikut adalah penjelasan mengenai ruang lingkup dari pembatasan masalah yang diberikan:

1. Algoritma yang digunakan untuk deteksi kendaraan yang diparkir tidak tertib adalah Algoritma *Detection Transformer* (DETR).
2. Sistem hanya mendeteksi kendaraan yang diparkir tidak tertib di area parkir yang memiliki garis parkir.
3. *Dataset* yang digunakan untuk melatih model diambil dari sumber terbuka seperti Roboflow atau Kaggle yang memiliki anotasi dan kualitas gambar sesuai dengan kebutuhan deteksi pelanggaran parkir.
4. Objek yang dideteksi hanya 1 jenis kendaraan bermotor yaitu mobil roda empat.

1.4. Tujuan

Adapun tujuan tugas akhir dari penulisan proposal ini yaitu sebagai berikut:

1. Mengimplementasikan algoritma *Detection Transformer* (DETR) untuk mendeteksi kendaraan yang parkir tidak tertib di area kampus.
2. Menganalisis model dalam mendeteksi kendaraan untuk memastikan bahwa sistem yang dikembangkan dapat beroperasi di lingkungan kampus.

1.5. Manfaat

Manfaat yang diharapkan dari sistem ini adalah:

1. Penelitian ini diharapkan dapat menjadi referensi dalam literatur tentang deteksi objek, khusunya dengan algoritma *Detection Transformer* (DETR).
2. Meningkatkan efesiensi penertiban parkir kendaraan tidak tertib sehingga terlihat lebih rapi dan estetik.

1.6. Sistematika Penulisan

Sistematika penulisan yang akan digunakan oleh penulis dalam pembuatan laporan tugas akhir seperti pada table 1.1:

Tabel 1. 1 Sistematika Penulisan

BAB 1	:	PENDAHULUAN
		Pada bab ini penulis mengutarakan latar belakang pemilihan judul, rumusan masalah, Batasan masalah, tujuan penelitian, metodologi penelitian, dan sistematika penulisan
BAB 2	:	TINJAUAN PUSTAKA DAN DASAR TEORI
		Bab ini memuat penelitian-penelitian sebelumnya dan dasar teori yang berguna untuk membantu penulis untuk memahami bagaimana konsep kerja dari sistem deteksi parkir tidak tertib dari sebuah kendaraan menggunakan *Detection Transformer* (DETR).
BAB 3	:	METODE PENELITIAN
		Bab ini menguraikan tentang proses tahapan-tahapan penelitian dimulai dari perancangan sistem, alur kerja sistem, dan pengujian sistem.
BAB 4	:	HASIL PENELITIAN
		Pada bab ini penulis mengungkapkan hasil penilitian yakni pendeteksian parkir tidak tertib kendaraan menggunakan *Detection Transformer* (DETR).
BAB 5	:	KESIMPULAN DAN SARAN

		Bab ini penulis memaparkan Kesimpulan pada proses penelitian dari awal proses penelitian hingga akhir serta saran yang akan dikembangkan.

BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauaan Pustaka

Penelitian sebelumnya menerapkan algoritma DETR untuk mendeteksi kendaraan yang ada pada jalan raya. Tujuan utama pelatihan adalah untuk menunjukkan kinerja algoritma DETR dalam mencapai hasil terdepan dalam tugas deteksi objek berdasarkan pengukuran COCO. Pelatihan ini melibatkan penggunaan ResNet-50 dan ResNet-101 sebagai backbone, dan metode DETR dibandingkan dengan DETR-DC5. Secara keseluruhan, DETR-DC5 meningkatkan deteksi kendaraan. Secara keseluruhan, AP50-95 konsisten antara DETR-DC5 dengan ResNet-50 dan DETR-DC5 dengan ResNet-101, masing-masing dengan nilai 0,665 masing-masing. DETR-DC5 dengan ResNet-101 mencapai nilai tertinggi 0,957 pada AP50 dan menunjukkan lebih banyak kemampuan untuk mendeteksi objek kecil dengan nilai APS 0,494. Namun, perlu diingat bahwa DETR-DC5 dengan Backbone ResNet-50 dapat dianggap sebagai pilihan yang efektif karena memiliki kombinasi kinerja yang baik dengan jumlah parameter yang relatif rendah, yaitu 35.5(Khoiriyah & Hendrawan, 2024).

Penelitian lain menggunakan algoritma *Detection Transformer* (DETR) dengan tujuan mendeteksi buah jeruk dalam gambar. Penelitian ini bertujuan untuk mengatasi tantangan seperti pencahayaan yang tidak merata, tumpang tindih antar objek, dan keberadaan *noise* visual yang umum dijumpai dalam lingkungan pertanian. DETR dinilai unggul karena arsitektur transformer-nya yang mampu menangkap hubungan spasial secara global antara piksel dalam gambar, berbeda dengan metode konvensional yang mengandalkan region proposal atau anchor-based detection. Hasil eksperimen menunjukkan bahwa DETR berhasil mendeteksi buah jeruk dengan akurasi mAP (*mean Average Precision*) sebesar 91,24% dan IoU (*Intersection over Union*) mencapai 84,6%, yang lebih tinggi dibandingkan dengan YOLOv8 dan *Faster* R-CNN dalam skenario yang sama.Penelitian ini membuktikan bahwa DETR sangat cocok digunakan dalam deteksi objek di lingkungan tidak terstruktur dan memiliki potensi besar untuk diadopsi pada studi kasus lain seperti deteksi kendaraan atau objek lain di area terbuka. Kemampuannya dalam mengurangi kebutuhan terhadap proses post-processing seperti *Non-Maximum Suppression* (NMS) juga menjadi nilai tambah dibandingkan metode deteksi tradisional(Jrondi dkk., 2024).

Penelitian lainnya menggunakan HHS-RT-DETR untuk mendeteksi penyakit *citrus* telah meningkatkan performa deteksi penyakit *citrus greening* atau *huanglongbing* pada tanaman jeruk melalui pengembangan algoritma RT-DETR. Dalam penelitian ini, penulis mengusulkan model baru bernama HHS-RT-DETR yang mengintegrasikan *Hierarchical Hybrid Selective Feature Pyramid Network* (HS-FPN) untuk meningkatkan kualitas fitur spasial dan semantik pada citra daun jeruk yang diperoleh dari lingkungan nyata. Selain itu, model ini juga menggunakan fungsi kerugian *Shape* IoU *Loss* yang dirancang untuk memperbaiki akurasi prediksi bentuk dan posisi objek yang terdeteksi. Melalui serangkaian eksperimen, HHS-RT-DETR menunjukkan kinerja yang lebih unggul dibandingkan model *baseline* seperti RT-DETR-r18. Model ini berhasil mencapai akurasi tinggi dengan nilai presisi sebesar 90,5%, *recall* sebesar 83,7%, *f1-score* sebesar 83,0%, serta *mean Average Precision* (mAP) sebesar 78,5%. Hasil tersebut menunjukkan bahwa HHS-RT-DETR mampu secara efisien mendeteksi daun jeruk yang terinfeksi dengan akurasi tinggi dan kompleksitas model yang rendah, menjadikannya cocok untuk diterapkan di lingkungan pertanian nyata yang membutuhkan deteksi cepat dan akurat. Penelitian ini membuktikan bahwa pengembangan arsitektur *backbone* dan pemilihan fungsi kerugian yang tepat dapat memberikan dampak signifikan terhadap performa model deteksi objek berbasis DETR(Huangfu dkk., 2024).

Selanjutnya adalah penelitian menggunakan *Mask* R-CNN untuk mendeteksi ketersediaan tempat parkir di area Fakultas Teknologi Industri

Unissula. Berdasarkan penelitian tersebut *Mask* R-CNN mampu mendeteksi ketersediaan tempat parkir di Fakultas Teknologi Industri Unissula dengan hasil yang cukup baik dengan model yang telah dilatih terdiri dari 877 gambar dan saat pengujian mendapat nilai akurasi sebesar 94,3%, presisi 100%, *recall* 93%, dan *f1-score* 96%(Ayudhawara dkk., 2023).

Penelitian berikutnya bertujuan untuk menggunakan algoritma *Intelligent Transportation System* (YOLO) untuk mendeteksi pelanggaran parkir di bahu jalan tol. Penelitian ini menemukan hasil pembelajaran terbaik pada 0,06 dengan batch maksimum 4000, nilai mAP 97,96%, dan akurasi yang dihasilkan 80%. Dengan menggunakan *bot* telegram yang mengirimkan pesan, sistem dapat mendeteksi dan memberikan pemberitahuan kepada petugas jalan tol. “Terdeteksi Kendaraan Melanggar”. Menggunakan Jupyter *Notebook* pada saat *training*(Yusfian dkk., 2022).

Kemudian terdapat penilitan untuk mendeteksi pelanggar rambu larangan parkir dan berhenti di Kawasan Telkom *Univesity*. Terdapat tiga video yang diuji dengan objek yang berbeda, yaitu mobil, motor, dan manusia. Oleh karena itu, setiap video diuji dengan objek yang berbeda dan menghasilkan tiga parameter: luas kontur, tinggi bidang batas, dan lebar bidang batas. Selain itu, pengujian ini menggunakan nilai terendah dari kategori objek sebelumnya: luas kontur lebih dari 94177, tinggi lebih dari 299, dan lebar lebih dari 480. Hasil dari penelitian ini menunujukkan bahwa objek mobil memiliki memiliki luas kontur 134571, tinggi 301, dan lebar 573. Adapun untuk objek berupa motor memiliki luas kontur 58493, tinggi 280, dan lebar 348. Sedangkan objek manusia memiliki luas kontur 11356, tinggi 218, dan lebar 114(Nurza dkk., 2020).

Penelitian ini akan mengembangakan beberapa penelitian diatas sebagai acuan serta dapat menciptakan sistem yang lebih efektif dalam menangani kasus parkir sembarangan yang ada di area kampus.

2.2. Dasar Teori

2.2.1. Tempat Parkir

Tempat parkir merupakan salah satu fasilitas penunjang yang keberadaannya sangat penting dan banyak dijumpai pada berbagai lokasi, seperti pusat perbelanjaan, kawasan bisnis, perkantoran, hingga instansi pemerintah. Di Indonesia sendiri, sistem parkir yang digunakan pada umumnya dapat dibedakan menjadi dua jenis, yaitu sistem parkir tradisional yang masih mengandalkan tenaga manusia, serta sistem parkir modern yang sudah memanfaatkan teknologi untuk pengelolaannya. Seiring dengan meningkatnya jumlah penduduk, pertumbuhan jumlah kendaraan bermotor juga mengalami peningkatan yang cukup signifikan dari tahun ke tahun. Kondisi ini secara langsung berdampak pada semakin terbatasnya ketersediaan lahan parkir, khususnya di kawasan perkotaan yang padat aktivitas. Akibat keterbatasan lahan tersebut, tidak jarang pengendara terpaksa memarkirkan kendaraannya secara sembarangan atau bahkan di area yang sebenarnya tidak diperuntukkan sebagai tempat parkir, sehingga menimbulkan permasalahan baru seperti kemacetan, gangguan ketertiban, serta penurunan kenyamanan bagi pengguna jalan lainnya.(Ayudhawara dkk., 2023).

Minimnya lahan parkir atau tempat parkir dapat menyebabkan kesulitannya para pengendara dalam memarkirkan kendaraan mereka. Sehingga para pengendara kendaraan cenderung memarkirkan kendaraan mereka asal-asalan atau tidak tertib. Parkir tidak tertib atau letak kendaraan yang tidak sesuai dengan area parkir juga sering dijumpai pada area kampus. Dosen, Mahasiswa, serta pengunjung terkadang memarkirkan kendaraan nya tidak sesuai dengan tempat parkir yang tersedia.

Gambar 2. 1 Kendaraan yang parkir tertib

Pada gambar 2.1 menunjukkan kondisi parkir mobil yang tertib atau sesuai dengan garis parkir yang tersedia.

Gambar 2. 2 Kendaraan yang parkir tidak tertib

Pada gambar 2.2 menunjukkan kondisi parkir kendaraan yang tidak tertib dengan melanggar garis parkir yang tersedia.

2.2.2. *Computer Vision*

Computer Vision adalah ilmu komputer yang bekerja dengan cara meniru kemampuan visual manusia. Pembelajaran menganalisis gambar dan video untuk memperoleh hasil serupa dengan apa yang bisa dilakukan manusia adalah inti dari komputer visi. Pada hakikatnya, komputer visi mencoba meniru cara kerja sistem visual manusia (Human Vision). Orang menggunakan indra penglihatan mereka (mata) untuk melihat sesuatu, kemudian gambar tersebut dikirim ke otak untuk diinterpretasikan, sehingga orang dapat memahami apa yang mereka lihat dengan mata mereka(Dompeipen dkk., 2021).

Perkembangan di bidang ini diikuti oleh kemampuan manusia untuk mengumpulkan informasi melalui indra penglihatan. Berbeda dengan disiplin grafik komputer, computer vision adalah bidang ilmu yang mempelajari bagaimana mengambil informasi secara menyeluruh dari gambar. Kemajuan dalam bidang ini sangat bergantung pada kemajuan teknologi komputer, baik dalam meningkatkan kualitas gambar maupun dalam mengenali objek dalam gambar. Dalam hal teknik yang digunakan, ada dasar yang mirip dengan pemrosesan gambar, dan beberapa penulis bahkan menggunakan istilah komputer visi dan pemrosesan gambar secara bergantian(Babatunde dkk., 2015).

2.2.3. *Machine Learning*

Machine Learning adalah salah satu bidang ilmu komputer yang dikenal sebagai pembelajaran mesin adalah bidang yang memungkinkan komputer untuk mengetahui sesuatu tanpa pemrograman yang jelas. Makin belajar memiliki dua fungsi utama: klasifikasi dan prediksi(Dompeipen dkk., 2021).

Pada dasarnya, machine *learning* adalah proses komputer untuk belajar dari data; tanpa data, komputer tidak akan dapat belajar apa-apa. Meskipun data tidak berbeda, ada algoritma dan teknik yang berbeda untuk mendapatkan hasil terbaik(Dompeipen dkk., 2021).

2.2.4. *Deep Learning*

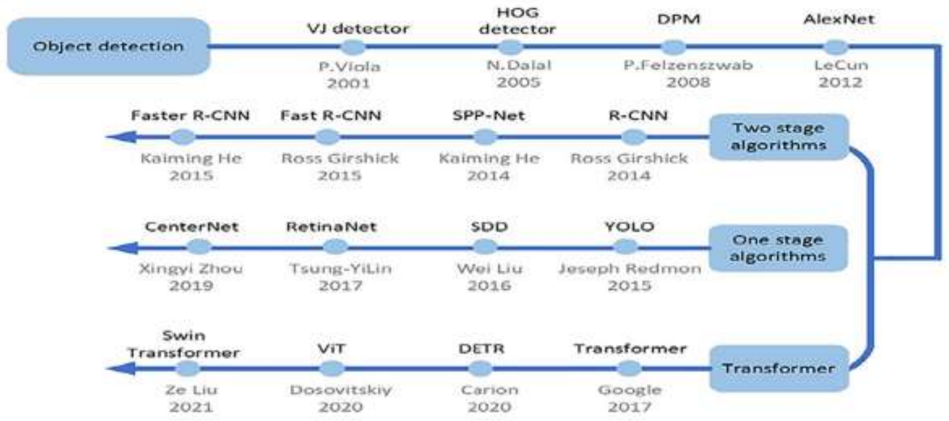
Teknik pembelajaran mendalam memberikan arsitektur yang sangat baik untuk pembelajaran supervisi, dan merupakan salah satu bidang pembelajaran mesin yang menggunakan jaringan syaraf tiruan untuk menyelesaikan masalah dengan *dataset* yang sangat besar. Dengan menambahkan lebih banyak lapisan, model pembelajaran tersebut dapat mewakili data gambar berlabel dengan lebih baik. Teknik dalam pembelajaran mesin melibatkan penggunaan fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi gambar dan mengenali suara. Namun demikian, metode ini masih memiliki beberapa kekurangan dalam hal akurasi dan kecepatan(Dompeipen dkk., 2021).

Model *Deep Learning* menggunakan berbagai lapisan. Dalam konteks jaringan saraf tiruan, *Multi-Layer Perceptron* (MLP) dengan lebih dari dua lapisan tersembunyi disebut *Deep* Model. *Convolution*, *dropout*, *fully connected*, *pooling*, dan *Relu layer* adalah beberapa lapisan yang sering digunakan. Setelah beberapa *layer* diterapkan, lebih banyak data akan dikumpulkan, dan untuk menghindari *overfitting*, beberapa teknik, yang paling umum adalah regularisasi dan augmentasi data(Ponti dkk., 2017).

2.2.5. Deteksi Objek

Metode untuk mengidentifikasi keberadaan objek tertentu dalam gambar digital dikenal sebagai deteksi objek.(Rizkatama dkk., 2021). Orang dapat langsung mengenali objek, letak, dan interaksi saat melihat gambar. Manusia dapat melakukan tugas-tugas kompleks dengan sistem visual manusia yang cepat dan akurat.(Mustafa dkk., 2024).

Seiring dengan perkembangan teknologi, muncul pertanyaan mengenai bagaimana jika sebuah komputer mampu melaksanakan tugas-tugas yang kompleks. Untuk mewujudkan hal tersebut, dibutuhkan suatu algoritma yang mampu diakses secara cepat dan akurat sehingga komputer dapat melakukan proses serupa dan bahkan berpotensi menyelesaikan tugas bersifat umum. Dalam ranah *computer vision*, salah satu fungsi utamanya adalah mengenali objek. Proses untuk menentukan keberadaan suatu objek tertentu dalam citra digital dikenal dengan istilah *object detection* atau deteksi objek.(Dompeipen dkk., 2021).

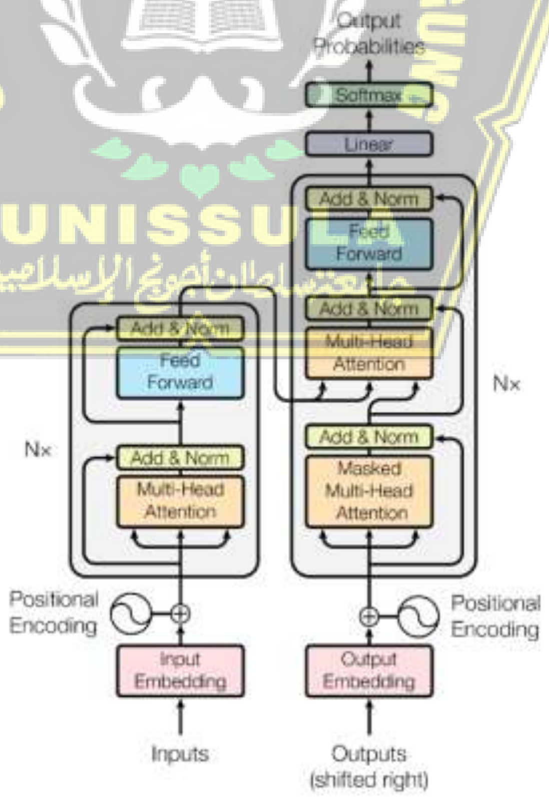

Object detection
VJ detector
P.Viola
2001
HOG detector
N.Dalal
2005
DPM
P.Felzenszwab
2008
AlexNet
LeCun
2012
Two stage algorithms
R-CNN
Ross Girshick
2014
SPP-Net
Kaiming He
2014
Fast R-CNN
Ross Girshick
2015
Faster R-CNN
Kaiming He
2015
One stage algorithms
YOLO
Jeseph Redmon
2015
SDD
Wei Liu
2016
RetinaNet
Tsung-YiLin
2017
CenterNet
Xingyi Zhou
2019
Transformer
Transformer
Google
2017
DETR
Carion
2020
ViT
Dosovitskiy
2020
Swin Transformer
Ze Liu
2021

Gambar 2. 3 Sejarah Perkembangan Deteksi Objek(Arkin dkk., 2023)

Pada gambar 2.3 menunjukkan perkembangan metode *object detection* dari masa ke masa. Awalnya, deteksi objek dilakukan menggunakan pendekatan tradisional seperti VJ detector (2001), HOG detector (2005), dan DPM (2008), hingga kemudian muncul era *deep learning* dengan AlexNet (2012) yang menjadi tonggak penting dalam visi komputer. Selanjutnya, algoritma berkembang menjadi dua kategori utama, yaitu *two-stage algorithms* seperti R-CNN, *Fast* R-CNN, dan *Faster* R-CNN yang fokus pada tahap proposal objek sebelum klasifikasi, serta *one-stage algorithms* seperti YOLO, SSD, dan RetinaNet yang lebih cepat karena langsung melakukan deteksi. Perkembangan terbaru menghadirkan arsitektur berbasis Transformer, seperti DETR, ViT, dan Swin Transformer, yang memanfaatkan mekanisme *self-attention* untuk menangkap hubungan global dalam citra, sehingga menghasilkan performa deteksi yang lebih akurat dan efisien.

2.2.6. *Transformer*

Perkembangaan teknologi saat ini tentang pengklasifikasian objek benda sudah sangat banyak. Pengembangan arsitektur baru sudah mulai bermunculan dengan berbagai kelebihan dan kekurangan. Pengklasifikasian objek mulai merambah ke arsitektur *transfomer.* Banyak penelitian terbaru menunjukkan bahwa arsitektur Convolutional Neural Network (CNN) mungkin tidak lagi menjadi pendekatan yang paling optimal untuk menyelesaikan berbagai permasalahan dalam bidang visi komputer. Hal ini disebabkan oleh keterbatasan CNN dalam menangkap ketergantungan global antar piksel pada citra yang kompleks. Sebagai alternatif, pendekatan berbasis *self-attention* yang diimplementasikan dalam arsitektur Transformer memberikan kemampuan bagi model untuk menganalisis interaksi global antara piksel-piksel dalam sebuah gambar. Mekanisme tersebut memungkinkan model tidak hanya mengenali detail lokal, tetapi juga memahami konteks spasial yang lebih luas sehingga citra dapat dipahami secara menyeluruh. Dengan demikian, model mampu menangkap fitur-fitur yang lebih relevan, menghasilkan representasi yang lebih bermakna, serta meningkatkan kualitas klasifikasi. Selain itu, penggunaan Transformer juga membuka peluang pengembangan metode yang lebih fleksibel dan dapat diadaptasi untuk berbagai jenis tugas pada domain visi komputer, sehingga performa sistem secara keseluruhan menjadi lebih unggul dibandingkan pendekatan konvensional berbasis CNN(Arkin dkk., 2023).

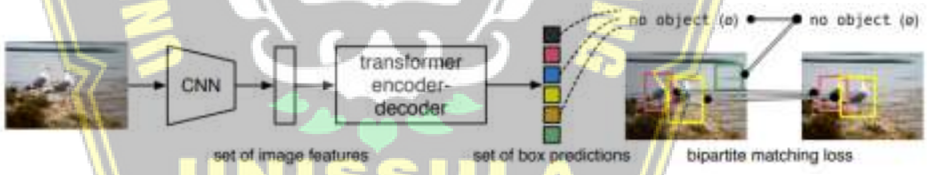

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Gambar 2. 4 Arsitektur pada *Transformer*

Gambar 2.4 menunjukkan alur pemrosesan data dari *embedding input* hingga prediksi *output* melalui *encoder* dan *decoder*. *Encoder* memproses *input* dengan *self-attention* dan *feed-forward network*, sedangkan *decoder* menambahkan *cross-attention* untuk menghasilkan *output* akhir. *Positional encoding* digunakan untuk mempertahankan urutan data, dan hasil akhir diprediksi melalui *layer linear* dan *softmax.*

2.2.7. *Detection Transformer* (DETR)

DETR berbeda dari model deteksi objek konvensional karena dapat memprediksi kelas objek dan kotak pembatas secara langsung. Arsitektur transformator encoder-decoder dan pencocokan bipartit memungkinkan kemampuan ini. Dengan menghilangkan komponen yang dirancang secara manual, DETR mempersederhanakan pipa deteksi. Metode ini berbeda dari banyak metode deteksi lainnya yang memerlukan lapisan yang dimodifikasi. DETR dapat direplikasi dengan mudah dalam kerangka apa pun yang memiliki kelas CNN dan transformator konvensional. Pada gambar 2.5 menampilkan prinsip kerja dari DETR.



Gambar 2. 5 Prinsip Kerja Algotitma DETR(Carion dkk., 2020)

Memprediksi kerugian bermanfaat untuk model pelatihan yang memprediksi serangkaian keluaran untuk deteksi objek (kelas, posisi, dan ukuran). Kerugian regresi kotak pembatas dan kerugian klasifikasi adalah dua komponen umum dari kerugian yang ditentukan. *Detection Transformer* (DETR) merupakan arsitektur deteksi objek yang terdiri dari tiga komponen utama, yaitu CNN *Backbone*, *Transformer Encoder-Decoder*, *Prediction Feed-Forward Network* (FFN).

1. CNN *Backbone*

    CNN *Backbone* adalah jaringan konvolusional yang berfungsi sebagai ekstraktor fitur pada gambar input. Ia mengubah gambar menjadi representasi fitur 2D beresolusi lebih rendah yang memuat informasi penting seperti bentuk, tekstur, dan pola visual. CNN backbone biasanya berupa jaringan dalam seperti ResNet yang telah terbukti efektif dalam ekstraksi fitur untuk pengolahan citra.

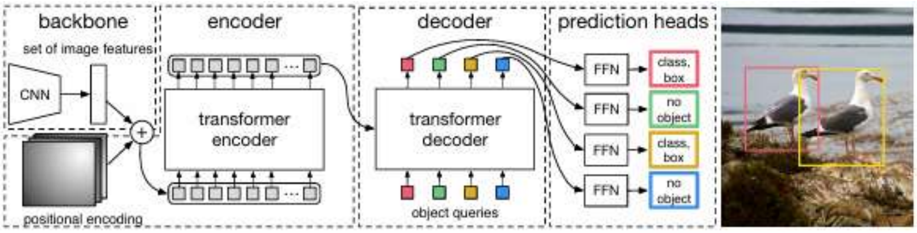
2. *Transformer Encoder-Decoder*

   Komponen ini menerapkan arsitektur *transformer* untuk memahami konteks *global* dan hubungan antar objek dalam gambar. *Encoder* menerima *output* fitur dari CNN *backbone*, meratakannya dan menambahkan positional encoding agar informasi spasial tetap terjaga. *Decoder* menggunakan sejumlah *embedding* yang disebut *object queries*, yang secara paralel memperhatikan hasil *encoder* untuk memprediksi objek di gambar. Mekanisme *attention multi-head* dalam *encoder-decoder* memungkinkan model menangkap relasi rumit antar bagian gambar dan objek.

3. *Prediction Feed-Forward Network* (FFN)

   FFN adalah *multilayer perceptron* tiga lapis yang menerima embedding dari decoder dan mengubahnya menjadi prediksi akhir. FFN memproses setiap output embedding secara independen menggunakan fungsi aktivasi *non-linear* seperti ReLU. Hasilnya berupa koordinat *bounding box* ter-normalisasi (koordinat tengah, tinggi, lebar) dan prediksi kelas objek menggunakan *softmax*. FFN bertugas menerjemahkan representasi fitur abstrak menjadi hasil deteksi konkret.

Pada gambar 2.6 adalah diagram yang memperlihatkan alur data dari gambar input hingga prediksi objek akhir dalam model DETR. Secara umum, gambar ini menampilkan beberapa komponen utama dan prosesnya.


backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
FFN
FFN
FFN
class, box
no object
class, box
no object

Gambar 2. 6 Alur dalam model DETR(Carion dkk., 2020)

DETR menggunakan CNN konvensional sebagai *backbone* untuk mempelajari representasi dua dimensi dari gambar *input*. Representasi ini kemudian diproses menggunakan mekanisme *attention* dan dilengkapi dengan *positional encoding* sebelum dimasukkan ke dalam *transformer encoder*. Selanjutnya, *transformer decoder* menerima sejumlah kecil *positional embedding* yang sudah dipelajari sebelumnya, yang disebut *object queries*. *Decoder* ini juga melakukan *attention* terhadap *output* dari *encoder*. Setiap output dari *decoder* kemudian diteruskan ke sebuah jaringan *feed-forward* (FFN) bersama yang bertugas untuk memprediksi apakah *output* tersebut merupakan deteksi objek, yaitu kelas objek dan posisi *bounding box* atau kelas “tidak ada objek”(Carion dkk., 2020).

DETR terdiri dari tiga komponen utama, yaitu CNN *Backbone*, *Transformer*, dan *Feed-Forward Network* (FFN) untuk klasifikasi dan prediksi *bounding box*. Bagian pertama dari arsitektur DETR adalah *backbone convolutional neural network* (misalnya ResNet-50), yang bertugas mengekstrak fitur visual dari gambar. Gambar berukuran $H \times W$ 3 akan dikonversi menjadi *feature map* berdimensi $C \times H' \times W'$, dimana C adalah jumlah saluran (channels), dan $H' \times W'$ adalah dimensi spasial hasil *downsampling*(Carion dkk., 2020).

Karena *Transformer* tidak memiliki pemahaman posisi spasial seperti CNN, DETR menambahkan *positional encoding* berbasis sinusoida pada setiap posisi spasial. Adapun rumus *positional encoding* seperti pada rumus nomor 1:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d}}) \tag{1}$$

Keterangan:

- pos : posisi dalam urutan
- i : indeks dimensi
- d : dimensi *embedding*

Feature map + positional encoding dikirim ke *Transformer Encoder*. *Encoder* terdiri dari beberapa *layer* yang masing-masing mengandung:

- *Multi-head Self-Attention* (MSA)
- *Feed-Forward Neural Network* (FFN)
- *Residual Connection* dan *Layer Normalization*

Adapun rumus *self-attention* seperti pada rumus nomor 2:

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

Keterangan:

- Q, K, V : *Query*, *Key*, dan *Value* matriks
- d_k : dimensi dari *key*

Selanjutnya *decoder* menerima *object queries*, yaitu vektor *embedding* yang belajar mewakili kemungkinan keberadaan objek. Setiap object query mengalami *self-attention* diantara *object queries*, *Cross-attention* terhadap *output encoder*, dan FFN. *Output decoder* akan berupa sekumpulan representasi prediktif yang siap untuk diklasifikasikan dan diubah menjadi *bounding box*. Setiap hasil dari decoder akan diteruskan ke dua cabang FFN, yaitu klasifikasi objek (dengan softmasx) dan prediksi bounding box dengan format $[cx, cy, w, h]$ dalam skala relatif [0,1]. DETR menggunakan *Hungarian Matching* untuk memetakan prediksi dan *ground truth* satu per satu secara optimal, lalu dihitung *loss*-nya. Rumus total *loss* seperti pada rumus nomor 3:

$$\mathcal{L}_{total} = \sum_{i=1}^{N}\left[\mathcal{L}_{cls}\left(ci, \hat{c}_{\sigma(i)}\right) + \lambda_{box}\cdot \mathcal{L}_{box}\left(b_i, \hat{b}_\sigma(i)\right)\right] \qquad (3)$$

Keterangan:

- c_i : kelas *ground-truth*
- $\hat{c}_{\sigma(i)}$: kelas prediksi yang dicocokkan
- b_i : *bounding box ground-truth*
- $\hat{b}_\sigma(i)$: *bounding box* prediksi
- λ_{box} : bobot untuk *box loss*

Adapun untuk rumus *bounding box loss* seperti pada rumus nomor 4:

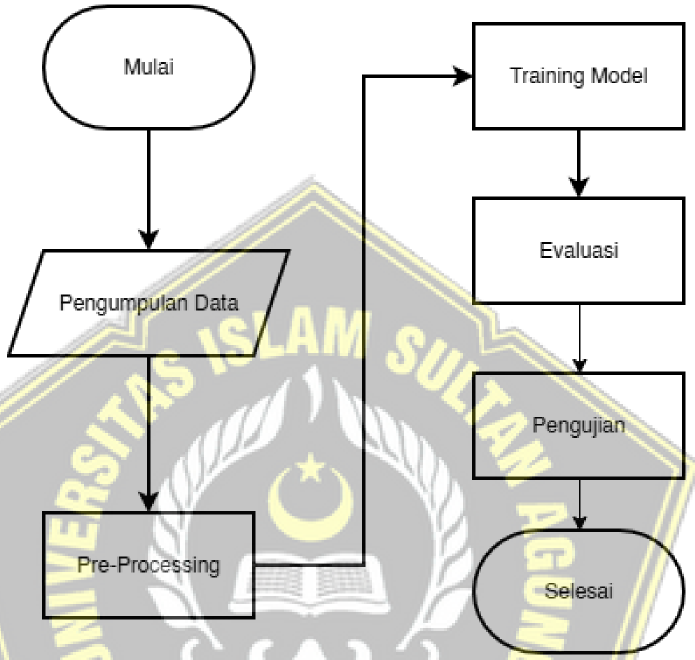
$$\mathcal{L}_{box} = \lambda_{L1} \cdot \left\| b - \hat{b} \right\|_1 + \lambda_{\text{GIoU}} \cdot \mathcal{L}_{\text{GIoU}}\left(b, \hat{b}\right) \tag{4}$$

Keterangan:

- $L1$ loss menghitung jarak absolut
- GIoU (*Generalized* IoU) mempertimbangkan *overlap* dan area minimum *bounding box*

BAB III
METODOLOGI PENELITIAN

Dalam penelitian ini, sistem deteksi parkir kendaraan tidak tertib akan menggunakan metode *Detection Transformer* dalam pembuatan modelnya.



Gambar 3. 1 Alur Penelitian

Pada gambar 3.1 menjelaskan tahapan alur dari penelitian yang dilakukan oleh peneliti. Proses dimulai dari tahap pengumpulan data yang diambil dari sumber *dataset* publik seperti Roboflow. Selanjutnya, dilakukan penentuan objek yang akan dideteksi, yaitu kendaraan yaitu mobil roda 4 yang diberi label "tertib" dan "tidak tertib" sesuai posisi parkirnya. Setelah objek ditentukan, data gambar yang tersedia akan diproses melalui tahapan *pre-processing* seperti *resize*, normalisasi, dan augmentasi untuk menyiapkan data dalam format yang sesuai dengan kebutuhan model.

3.1. Pengumpulan Data

Pada tahap ini, peneliti merencanakan pengumpulan data berupa citra kendaraan roda empat yang digunakan sebagai data latih dan data uji. Sumber data diperoleh dari *dataset* publik yang tersedia pada *platform Roboflow*. Pemilihan *dataset* dari *Roboflow* dipertimbangkan karena ketersediaannya yang sesuai dengan konteks penelitian, serta kemudahan dalam proses pengelolaan dan integrasi dengan tahap pelabelan maupun pelatihan model.

Explore the Roboflow Universe

Gambar 3. 2 Tampilan *Roboflow*

Pada gambar 3.2 merupakan tampilan pada *platform Roboflow* yang akan dijadikan peneliti untuk pengumpulan data. *Dataset* yang digunakan direncanakan memiliki jumlah yang memadai dengan variasi posisi kendaraan, sudut pandang kamera, serta kondisi pencahayaan yang berbeda. Variasi tersebut diharapkan dapat membantu sistem dalam mempelajari karakteristik objek secara lebih baik, sehingga model mampu mendeteksi kendaraan roda empat dengan tingkat akurasi yang optimal.

3.2. *Pre-Processing*

Sebelum dilakukan pelatihan model, *dataset* yang telah dikumpulkan akan melalui tahap *preprocessing*. Tahapan ini bertujuan untuk menyiapkan data agar sesuai dengan kebutuhan model serta meningkatkan kualitas data yang digunakan. Adapun langkah-langkah *preprocessing* yang direncanakan adalah sebagai berikut:

3.2.1. Anotasi Objek

Anotasi dilakukan dengan memberikan *bounding box* pada objek kendaraan roda empat. Proses ini penting karena algoritma deteksi objek, termasuk DETR, membutuhkan informasi posisi serta ukuran objek dalam citra untuk dapat belajar mengenali pola. Pada penelitian ini, anotasi dilakukan menggunakan platform Roboflow yang menyediakan antarmuka interaktif, sehingga proses pelabelan dapat dilakukan dengan lebih mudah, konsisten, dan terstandarisasi.

Dataset yang diunduh dari Roboflow telah dilengkapi dengan *file* anotasi dalam format COCO. Format ini menyimpan informasi *bounding box* dalam bentuk koordinat $[\mathcal{X}_{center}, \mathcal{Y}_{center}, width, height\,]$, sehingga dapat langsung dimanfaatkan pada tahap selanjutnya tanpa perlu dilakukan konversi manual. Dengan anotasi yang akurat dan format data yang sesuai, diharapkan model mampu membedakan antara area objek (mobil) dan latar belakang (background) secara efektif.

3.2.2. Augmentasi

Agar *dataset* lebih bervariasi, dilakukan proses augmentasi seperti *rotation*, *flip horizontal*, perubahan pencahayaan, dan skala. Augmentasi berfungsi untuk mensimulasikan berbagai kondisi nyata, misalnya kendaraan yang dipotret dari sudut berbeda atau pencahayaan yang berubah-ubah. Dengan cara ini, model diharapkan lebih robust dan mampu beradaptasi terhadap kondisi lingkungan yang berbeda.

3.2.3. Split *Dataset*

Dataset yang terkumpul dan telah dianotasi dibagi ke dalam tiga *subset*, yaitu:

1. *Training set* digunakan untuk melatih model agar mengenali pola pada data.
2. *Validation set* digunakan untuk memantau kinerja model selama proses pelatihan dan menghindari *overfitting*.
3. *Test set* digunakan untuk mengevaluasi performa akhir model pada data yang belum pernah dilihat sebelumnya.

Pembagian ini dilakukan otomatis oleh Roboflow untuk memastikan distribusi data merata, sehingga model tidak bias terhadap kondisi tertentu.

3.2.4. *Set-Up Environment*

Sebelum *dataset* digunakan dalam pelatihan, dilakukan penyiapan lingkungan pemrograman. Tahap ini mencakup instalasi pustaka yang diperlukan, seperti *PyTorch* untuk pemodelan *deep learning* dan *Transformers* untuk arsitektur DETR.

3.2.5. Membuat *Dataloader*

Dataset yang diekspor dari Roboflow dalam format COCO kemudian dimuat menggunakan *dataloader*. Tahap ini bertujuan agar data dapat diakses model dalam bentuk batch, sehingga proses pelatihan menjadi lebih mudah. Format COCO dipilih karena kompatibel dengan banyak algoritma deteksi objek, termasuk DETR.

3.2.6. Visualisasi Data

Sebelum digunakan dalam pelatihan, sebagian *dataset* divisualisasikan untuk memastikan bahwa anotasi dan *bounding box* telah sesuai. Visualisasi ini juga berfungsi sebagai tahap verifikasi kualitas data, karena kualitas anotasi akan sangat memengaruhi hasil akhir model.

3.3. *Training Model*

Proses pelatihan (*training*) model merupakan tahap penting dalam penelitian ini. Setelah data terkumpul dan melalui tahap praproses, langkah selanjutnya adalah melatih model agar dapat mengenali pola dari citra kendaraan yang telah dianotasi. Pelatihan model bertujuan agar sistem mampu menggeneralisasi data yang diberikan sehingga dapat mendeteksi objek baru dengan tingkat akurasi yang baik. Pada penelitian ini digunakan model *Detection Transformer* (DETR) yang memanfaatkan kombinasi arsitektur *Convolutional Neural Network (CNN)* sebagai ekstraktor fitur dan *Transformer* sebagai mekanisme perhatian (*attention mechanism*) untuk mendeteksi objek dalam gambar.

3.3.1. Arsitektur *Model*

Arsitektur yang digunakan dalam penelitian ini adalah *Detection Transformer* (DETR) dengan *backbone* ResNet-50. DETR dipilih karena mampu melakukan deteksi objek tanpa tahap *post-processing* tambahan seperti *Non-Maximum Suppression (*NMS*)*, sehingga proses deteksi lebih sederhana. Backbone ResNet-50 berfungsi mengekstraksi fitur dari citra, kemudian diproses oleh mekanisme *transformer* untuk mengenali objek kendaraan. Penyesuaian dilakukan pada lapisan keluaran agar sesuai dengan jumlah kelas dalam *dataset*, yaitu kendaraan parkir sesuai aturan (*proper*) dan tidak sesuai aturan (*improper*). Dengan dukungan mekanisme *attention*, model diharapkan dapat mendeteksi kendaraan dengan variasi posisi, sudut pandang, maupun pencahayaan.

3.3.2. Inisialisasi dan *Hyperparameter*

Proses pelatihan jaringan memerlukan sejumlah parameter yang harus ditentukan sebelum training. Parameter-parameter ini disebut *hyperparameter* dan sangat memengaruhi performa model.

1. *Learning Rate* (LR)

   Nilai *learning rate* digunakan mengatur besarnya langkah pembaruan bobot setiap kali model melakukan proses *backpropagation*. Nilai yang terlalu besar dapat membuat pelatihan tidak stabil, sedangkan nilai yang terlalu kecil menyebabkan proses konvergensi berjalan lambat.

2. *Weight Decay*

   Merupakan teknik regularisasi yang ditambahkan pada proses optimisasi untuk mencegah model mengalami *overfitting*. Dengan menurunkan bobot yang terlalu besar, *weight decay* membantu model agar lebih general dalam menghadapi data baru.

3. *Optimizer*

   Berfungsi untuk memperbarui bobot jaringan berdasarkan hasil perhitungan gradien. Pemilihan *optimizer* berpengaruh pada kecepatan dan kestabilan pelatihan, serta kemampuan model untuk mencapai solusi optimal.

4. *Scheduler*

   Digunakan untuk menyesuaikan nilai *learning rate* selama proses pelatihan. Dengan penurunan yang teratur, *scheduler* membantu model agar tetap stabil dan menghindari jebakan pada titik minimum lokal.

5. *Epoch*

   Menentukan berapa kali keseluruhan *dataset* dilalui oleh model selama pelatihan. Semakin banyak *epoch*, semakin besar peluang model untuk belajar, namun terlalu banyak *epoch* juga berisiko menimbulkan *overfitting*.

3.3.3. Strategi *Training*

Dalam proses pelatihan, tidak cukup hanya dengan menentukan arsitektur dan hyperparameter, melainkan juga diperlukan strategi tambahan agar pembelajaran berlangsung lebih efektif dan stabil. Strategi ini bertujuan untuk menghindari masalah umum seperti *overfitting*, lambatnya konvergensi, atau hilangnya gradien selama proses optimisasi. Dengan menerapkan strategi pelatihan yang tepat, model diharapkan mampu memaksimalkan kemampuan generalisasi serta memperoleh hasil yang lebih konsisten pada data validasi.

1. *Early Stopping*

   Digunakan untuk menghentikan pelatihan lebih awal jika performa model pada data validasi tidak menunjukkan peningkatan setelah sejumlah *epoch*. Hal ini mencegah model belajar terlalu lama hingga terjadi *overfitting*.

2. *Gradient Clipping*

   Membatasi nilai gradien maksimum pada saat *backpropagation* agar tidak terjadi *exploding gradient*. Teknik ini membantu menjaga stabilitas proses pelatihan, terutama pada model yang kompleks.

3. *Accumulate Gradient*

   Strategi ini memungkinkan akumulasi gradien dari beberapa batch kecil sebelum dilakukan pembaruan bobot. Cara ini efektif digunakan ketika keterbatasan GPU tidak memungkinkan penggunaan *batch size* besar.

4. Monitoring *Learning* Rate

Dengan memantau perubahan nilai *learning rate* selama pelatihan, peneliti dapat memastikan bahwa penyesuaian yang dilakukan oleh *scheduler* berjalan sesuai harapan.

3.3.4. Infrasruktur Komputasi

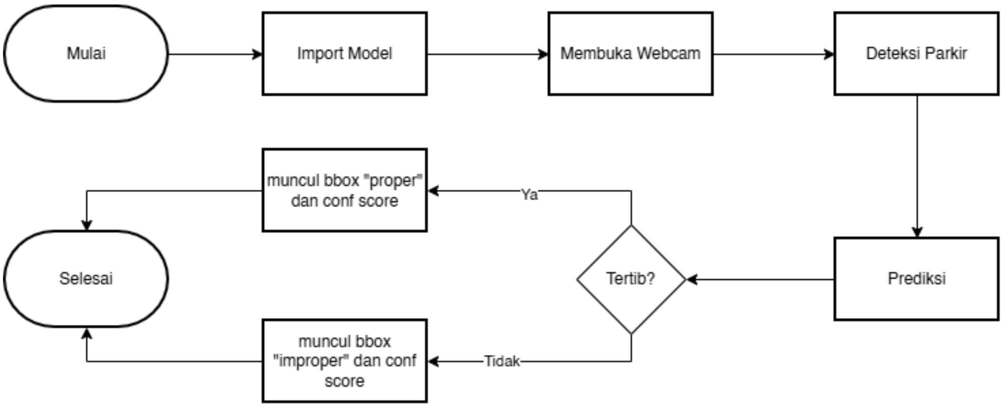
Pelatihan model *deep learning* membutuhkan sumber daya komputasi yang cukup besar, terutama karena proses optimisasi dilakukan dengan data dalam jumlah banyak serta parameter yang kompleks. Oleh karena itu, penelitian ini dijalankan menggunakan perangkat keras yang mendukung percepatan komputasi, seperti GPU, yang mampu mempercepat proses pelatihan dibandingkan hanya menggunakan CPU. Selain itu, perangkat lunak dan pustaka yang digunakan juga berperan penting dalam memastikan kompatibilitas dan efisiensi selama pelatihan berlangsung.

3.3.5. *Output Training*

Setiap proses pelatihan menghasilkan keluaran berupa rekaman metrik dan visualisasi yang menggambarkan perkembangan model dari waktu ke waktu. Informasi ini biasanya mencakup nilai *loss* pada data pelatihan dan validasi, akurasi, serta indikator lain yang dapat digunakan untuk mengevaluasi performa model. Hasil tersebut kemudian menjadi dasar untuk menilai sejauh mana model mampu belajar dengan baik dan apakah strategi pelatihan yang diterapkan sudah optimal.

3.4. Pengujian

Pada tahap pengujain ini dilakukan setelah tahap pelatian atau *Training* Model. Model diuji dengan data yang belum pernah dipelajari oleh algoritma. Tujuannya adalah untuk mengevaluasi kemampuan model dalam mendeteksi kendaraan yang terparkir di area kampus dengan kondisi “tertib” dan “tidak tertib”. Saat pengujian akan meng-*import* model yang telah disimpan pada *device*. Pengujian dilakukan secara *real-time* menggunakan *webcam* yang telah terhubung ke sistem serta melakukan deteksi terhadap objek. Proses dari pengujian ini dapat dilihat pada gambar 3.3 sebagai berikut.



Gambar 3. 3 Alur Pengujian

Pada gambar 3.3 terlihat alur dalam pengujian yang akan dilakukan penulis setelah mendapat hasil pelatihan yang baik. Pengujain juga dilakukan dengan aplikasi streamlit dari model yang sudah didapat dari pelatihan. Tujuan dari pengaplikasian dengan streamlit yaitu untuk memudahkan implementasi dan demo saat menjalankan model.

3.5. Evaluasi

Tahap evaluasi bertujuan untuk mengukur performa model dalam mendeteksi objek kendaraan serta mengklasifikasikan ketertiban parkir berdasarkan prediksi *bounding box* dan label yang dihasilkan oleh model DETR. Evaluasi dilakukan dengan membandingkan hasil prediksi terhadap anotasi ground truth menggunakan beberapa metrik evaluasi standar dalam tugas *object detection*. Metrik-metrik ini memberikan gambaran menyeluruh tentang ketepatan prediksi bounding box, kemampuan klasifikasi model, serta efektivitas sistem secara keseluruhan. Berikut ini adalah penjelasan metrik serta rumusnya:

1. *Accuracy*

Akurasi mengukur seberapa sering model menghasilkan prediksi yang benar dibandingkan dengan semua prediksi yang dilakukan. Namun, dalam konteks *object detection* seperti pada DETR, metrik *accuracy* tidak terlalu dominan karena lebih relevan untuk klasifikasi murni (tanpa mempertimbangkan lokasi objek). Meskipun demikian, akurasi tetap bisa digunakan untuk mengukur apakah sistem berhasil memberikan label yang tepat terhadap objek yang berhasil terdeteksi.

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

(5)

Keterangan:

- TP (*True Positive*): Mobil memiliki kondisi parkir tertib dan sistem berhasil mendeteksi parkir tertib.
- TN (*True Negative*): Mobil memiliki kondisi parkir tidak tertib dan sistem berhasil mendeteksi parkir tidak tertib.
- FP (*False Positive*): Mobil memiliki kondisi parkir tertib dan sistem tidak berhasil mendeteksi parkir tertib, sehingga terdeteksi tidak tertib.
- FN (*False Negative*): Mobil memiliki kondisi parkir tidak tertib dan sistem tidak berhasil mendeteksi parkir tidak tertib, sehingga terdeteksi tertib.

2. *Precision*

Presisi mengukur seberapa akurat model dalam mendeteksi objek yang diklaim sebagai "tidak tertib". Artinya, dari seluruh prediksi objek yang dianggap tidak tertib oleh model, berapa banyak yang benar-benar tidak tertib.

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

Presisi tinggi menunjukkan bahwa model jarang salah memberi label "tidak tertib" pada kendaraan yang seharusnya tertib, yang penting dalam konteks sistem monitoring agar tidak banyak *false alarm*.

3. *Recall*

Recall mengukur seberapa banyak dari seluruh kendaraan tidak tertib yang berhasil ditemukan dan diklasifikasikan dengan benar oleh model.

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

Nilai *recall* yang tinggi menandakan bahwa model memiliki kemampuan mendeteksi hampir semua pelanggaran parkir yang ada, sehingga sangat penting untuk sistem yang mengutamakan keseluruhan deteksi.

4. *F1-Score*

F1-Score adalah rata-rata harmonik antara *precision* dan *recall*. Metrik ini berguna ketika diperlukan keseimbangan antara menghindari *false positive* dan mengurangi *false negative*.

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

(8)

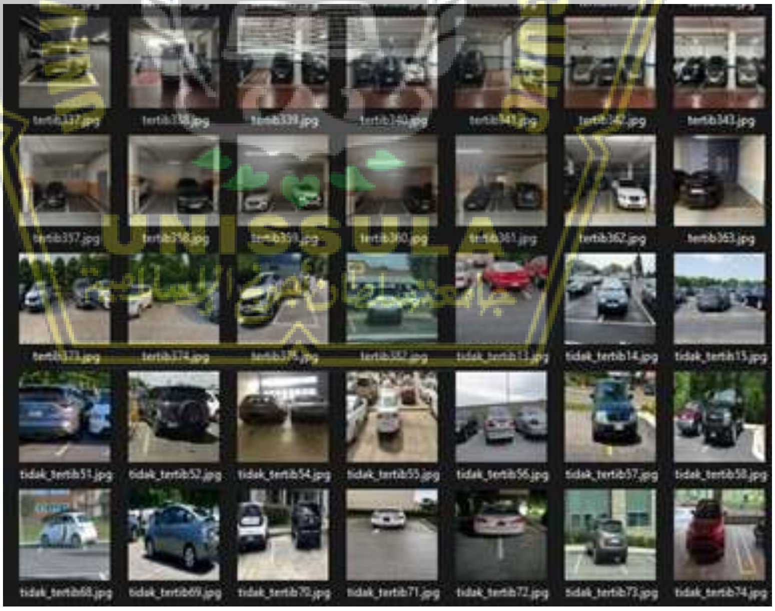
F1-score sangat direkomendasikan dalam evaluasi DETR karena mempertimbangkan aspek klasifikasi dan deteksi sekaligus.

UNIVERSITAS ISLAM SULTAN AGUNG UNISSULA جامعة سلطان أجونج الإسلامية

BAB IV
HASIL DAN ANALISIS PENELITIAN

4.1. Hasil Pengunpulan Data

Pada penelitian ini, *dataset* kendaraan roda empat diperoleh dari platform Roboflow dengan format COCO agar sesuai dengan kebutuhan arsitektur model DETR. Adapun dataset yang diambil manual oleh penulis di area kampus sesuai dengan studi kasus. *Dataset* ini terdiri dari 538 citra dengan 522 citra diambil dari *platform roboflow* dan hanya 16 citra diambil pada area kampus Universitas Islam Sultan Agung yang dibagi ke dalam *subset train*, *validation*, dan *test*. Pemisahan ini bertujuan agar model dapat dilatih sekaligus evaluasi secara objektif. Roboflow dipilih karena menyediakan *dataset* yang terorganisir dengan baik, lengkap dengan anotasi bounding box, sehingga dapat langsung digunakan pada tahap *preprocessing* tanpa perlu konversi manual.


tertib337.jpg
tertib338.jpg
tertib339.jpg
tertib340.jpg
tertib341.jpg
tertib342.jpg
tertib343.jpg
tertib357.jpg
tertib358.jpg
tertib359.jpg
tertib360.jpg
tertib361.jpg
tertib362.jpg
tertib363.jpg
tertib373.jpg
tertib374.jpg
tidak_tertib13.jpg
tidak_tertib14.jpg
tidak_tertib15.jpg
tidak_tertib51.jpg
tidak_tertib52.jpg
tidak_tertib54.jpg
tidak_tertib55.jpg
tidak_tertib56.jpg
tidak_tertib57.jpg
tidak_tertib58.jpg
tidak_tertib68.jpg
tidak_tertib69.jpg
tidak_tertib70.jpg
tidak_tertib71.jpg
tidak_tertib72.jpg
tidak_tertib73.jpg
tidak_tertib74.jpg

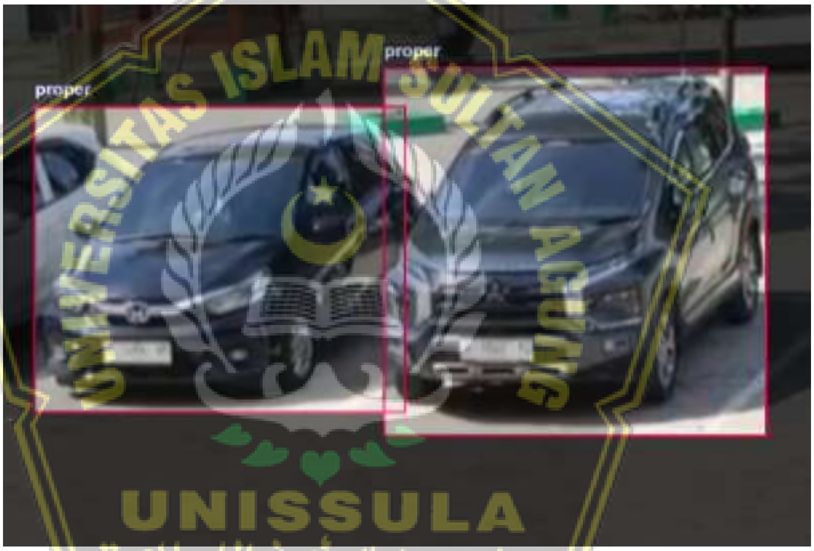
Gambar 4. 1 Data yang Terkumpul

Pada gambar 4.1 menunjukkan gambar pada *dataset* yang sudah terkumpul. Selain jumlah yang memadai, *dataset* ini memiliki keragaman dalam posisi kendaraan, sudut pandang kamera, serta kondisi pencahayaan. Variasi tersebut berperan penting untuk melatih model agar mampu mengenali objek mobil secara lebih general dan tidak hanya terbatas pada pola tertentu. Dengan adanya data yang beragam dan anotasi yang akurat, tahap pengumpulan data ini menjadi dasar penting untuk mendukung kinerja deteksi pada model.

4.2. Hasil *Pre-Processing*

4.2.1. Hasil Anotasi Objek

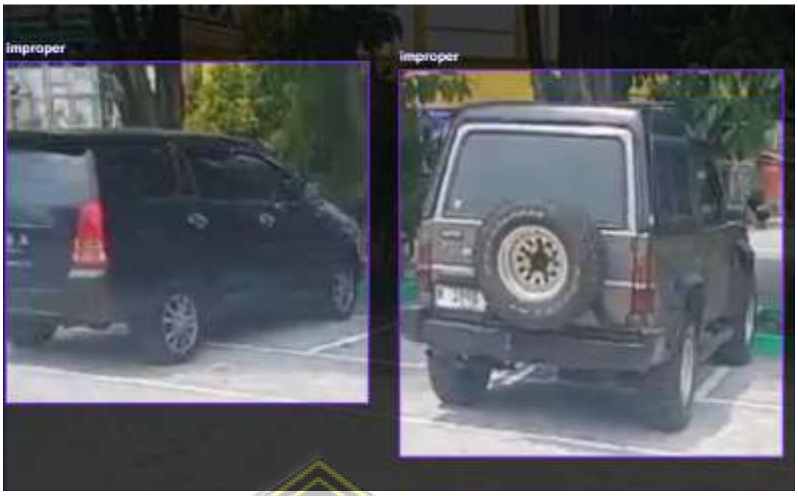
Anotasi yang dilakukan oleh peneliti dengan membuat 2 *class* utama yaitu "*proper*" dan "*improper*". Dengan tujuan untuk memudahkan sistem mengenali objek dan mendeteksinya.



Gambar 4. 2 Anotasi dengan *Class Proper*

Pada gambar 4.2 menunjukkan gambar yang sudah teranotasi dan dikelompokkan pada *class* yang sesuai dengan kondisi menyesuaikan dari mobil yang terparkir tertib atau tidak melanggar garis.

UNISSULA
جامعة سلطان أجوع الإسلامية
UNIVERSITAS ISLAM SULTAN AGUNG



Gambar 4. 3 Anotasi dengan *Class Improper*

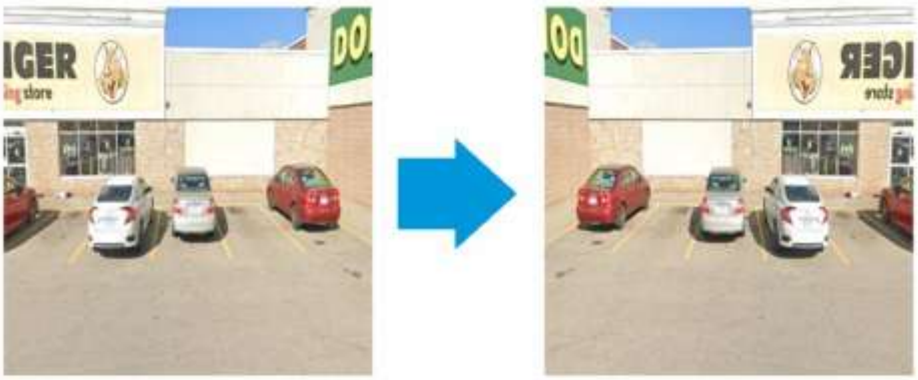
Pada gambar 4.3 menunjukkan 2 mobil dengan *Class improper* karena kondisi parkir yang tidak tertib atau melanggar garis.

4.2.2. Hasil Augmentasi

Setelah *resize* gambar maka dilakukan augmentasi data dilakukan untuk meningkatkan variasi citra sehingga model lebih adaptif terhadap kondisi nyata, Jenis augmentasi yang digunakan dalam penelitian ini meliputi:

1. *Flip Horizontal*

   Citra kendaraan dibalik secara *horizontal* sehingga arah kendaraan menjadi terbalik. Augmentasi ini membantu model untuk mengenali kendaraan dari sisi kiri maupun kanan, sehingga deteksi tidak hanya terbatas pada orientasi asli gambar.

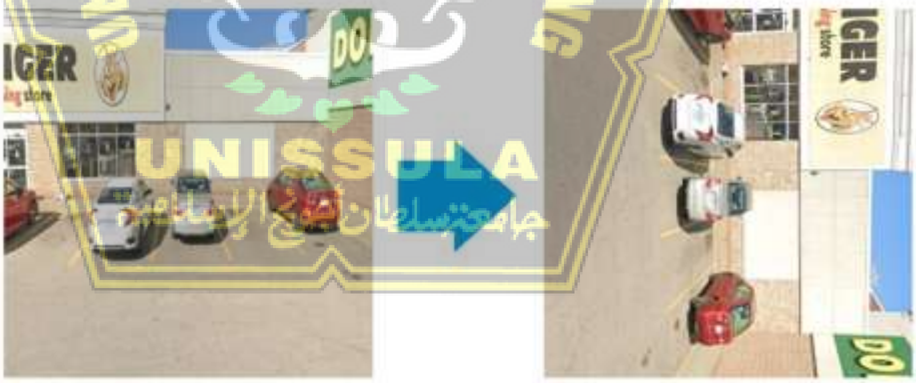


Gambar 4. 4 *Horizontal Flip*

Pada gambar 4.4 memperlihatkan perbandingan citra pada *dataset* sebelum dan sesudah dilakukan proses *Horizontal Flip Augmentation*. Teknik augmentasi ini bekerja dengan membalik gambar secara horizontal (dari kiri ke kanan atau sebaliknya), sehingga posisi objek pada citra menjadi terpantul seperti cermin. Penerapan metode ini bertujuan untuk meningkatkan variasi data latih tanpa perlu menambah jumlah data secara manual, sekaligus membantu model agar lebih robust dalam mengenali objek meskipun mengalami perbedaan orientasi. Dengan demikian, *Horizontal Flip Augmentation* dapat meningkatkan kemampuan generalisasi model terhadap data baru yang memiliki variasi posisi objek serupa.

2. *90° Rotate*

   Memutar citra sebesar 90 derajat, baik searah maupun berlawanan jarum jam. Tujuannya agar model mampu beradaptasi dengan kondisi ekstrem, misalnya ketika kamera dipasang dalam posisi tidak standar atau terjadi perubahan orientasi gambar.



Gambar 4. 5 90° Rotate

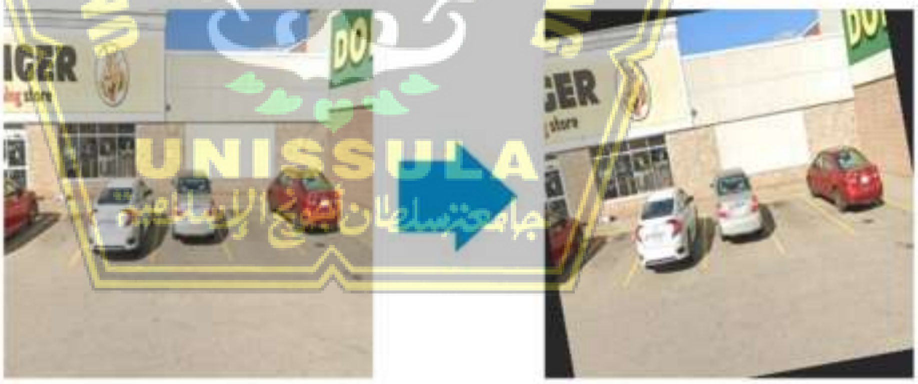
Pada gambar 4.5 menunjukkan citra pada *dataset* sebelum dan sesudah dilakukan *90° Rotate Augmentation*. Proses ini memutar gambar sebesar 90 derajat sehingga menghasilkan variasi orientasi objek. Tujuannya adalah menambah keragaman data latih agar model lebih adaptif dalam mengenali objek dengan posisi yang berbeda.

3. *Rotation*

Rotasi kecil pada rentang -15° hingga 15° meniru kondisi nyata saat pengambilan gambar, misalnya kamera miring atau kendaraan difoto dari sudut berbeda. Hal ini membuat model lebih fleksibel dalam mengenali objek dengan kemiringan tertentu.

Gambar 4. 6 *Rotation* +15°

Pada gambar 4.6 menunjukkan gambar pada *dataset* sebelum dan sesudah dilakukan *Rotation* +15° *Augmentation*.



Gambar 4. 7 *Rotation* -15°

Pada gambar 4.7 menunjukkan gambar pada *dataset* sebelum dan sesudah dilakukan *Rotation* -15° *Augmentation*.

4. *Saturation*

Penyesuaian saturasi bertujuan untuk menghadirkan variasi warna. Kendaraan yang difoto pada kondisi cahaya berbeda dapat tampak lebih pudar atau lebih mencolok, sehingga augmentasi ini membantu model menjadi lebih robust terhadap perbedaan intensitas warna.

Gambar 4. 8 *Saturation*

Pada gambar 4.8 menunjukkan gambar pada *dataset* sebelum dan sesudah dilakukan *Saturation Augmentation*. Teknik ini mengubah tingkat kejenuhan warna pada gambar sehingga menghasilkan variasi tampilan visual. Tujuannya adalah memperkaya data latih agar model lebih robust dalam mengenali objek pada kondisi pencahayaan dan warna yang berbeda.

5. *Brightness*

Mengatur kecerahan citra agar model dapat mengenali kendaraan baik pada kondisi terang (siang hari) maupun gelap (malam hari atau tempat minim cahaya). Augmentasi ini juga mencegah model terlalu bergantung pada pencahayaan tertentu.

Gambar 4. 9 *Brightness*

Pada gambar 4.9 menunjukkan gambar pada *dataset* sebelum dan sesudah dilakukan *Brightness Augmentation*.

4.2.3. Hasil Split *Dataset*

Setelah proses augmentasi selesai, *dataset* dibagi menjadi beberapa subset agar dapat digunakan secara optimal pada tahap pelatihan dan evaluasi model. Pembagian ini umumnya terdiri dari data latih (*training set*), data validasi (*validation set*), dan data uji (*test set*). Data latih digunakan sebagai dasar pembelajaran model, data validasi berfungsi untuk memantau kinerja model selama proses training serta mencegah overfitting, sedangkan data uji digunakan untuk menilai performa akhir model pada data yang belum pernah dilihat sebelumnya.

Tabel 4. 1 Split Data

No	***Subset***	**Jumlah Gambar**
1	*Train Set*	352
2	*Valid Set*	132
3	*Test Set*	54
	Total Gambar	538

Pada tabel 4.1 merupakan penentuan *split* data yang dibagi menjadi 3 *subset*. Proses split data pada penelitian ini dilakukan secara otomatis melalui platform Roboflow, sehingga pembagian data menjadi lebih terkontrol dan konsisten.

4.2.4. Hasil *Set-Up Environmnet*

Pada tahap ini dilakukan setup environment untuk memastikan seluruh kebutuhan perangkat lunak terpenuhi. Instalasi *library* menggunakan pip difokuskan pada paket yang mendukung pemanggilan model, pengolahan data, serta evaluasi, dengan tetap menyesuaikan kompatibilitas terhadap arsitektur DETR dan *framework* PyTorch.

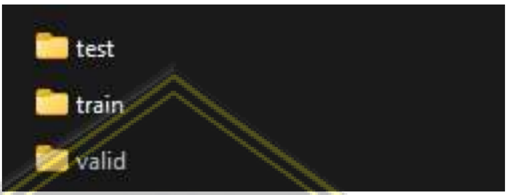
Tabel 4. 2 Penggunaan Library

No	*Library*	Versi	Fungsi
1	*supervision*	0.3.0	Manajemen paket Python untuk instalasi dan pembaruan *library*.
2	*transformers*	4.41.2	Digunakan untuk membantu proses anotasi, visualisasi, dan evaluasi deteksi objek.
3	*pytorch-lightning*	2.5.2	*Framework* untuk menyederhanakan training *loop* PyTorch agar lebih efisien.
4	timm	1.0.19	Koleksi pretrained model vision (CNN/*Transformer*) untuk computer vision.
5	cython	3.1.3	Optimasi kode Python agar lebih cepat melalui integrasi dengan C/C++.
6	pycocotools	2.0.10	Mendukung format anotasi COCO untuk evaluasi deteksi objek.
7	scipy	1.16.1	Digunakan untuk komputasi ilmiah, terutama operasi numerik dan optimasi.
8	roboflow	1.2.6	Memudahkan akses *dataset*, preprocessing, dan integrasi dengan pipeline training.

Pada table 4.2 menunjukkan library yang dipakai oleh peneliti dan fungsi dari library tersebut. Dengan konfigurasi tersebut, environment penelitian menjadi lebih terstandarisasi sehingga mengurangi potensi error pada saat implementasi. Selain itu, penggunaan library yang terintegrasi dengan baik juga mempermudah peneliti dalam melakukan eksperimen, mulai dari tahap persiapan data hingga evaluasi performa model.

4.2.5. Hasil *Dataloader*

Setelah melalui tahap *preprocessing*, *dataset* berhasil disusun dalam format COCO dan dibagi ke dalam tiga subset utama: *train*, *validation*, dan *test*. Tujuan dari pembagian ini adalah agar model tidak hanya dilatih, tetapi juga divalidasi secara bertahap serta diuji menggunakan data yang benar-benar belum pernah dilihat sebelumnya. Dengan demikian, performa model dapat dievaluasi secara lebih objektif dan terhindar dari *overfitting*.



Gambar 4. 10 Struktur Folder *Dataset*

Struktur folder *dataset* dapat dilihat pada Gambar 4.10 yang menunjukkan pembagian *dataset* menjadi 3 folder utama yang mewakili setiap *subset*, yaitu *train set*, *valid set*, *test set*.

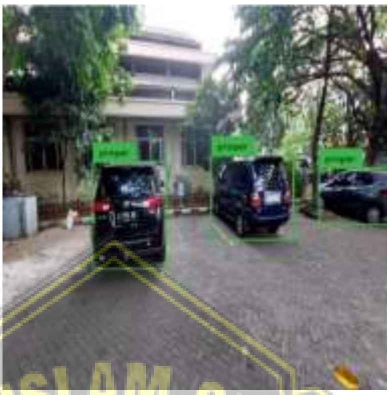
Tabel 4. 3 Hasil Distribusi Dataloader

No	***Subset***	**Jumlah Gambar**
1	*Train Set*	1056
2	*Valid* Set	132
3	*Test Set*	54
Total Gambar		1242

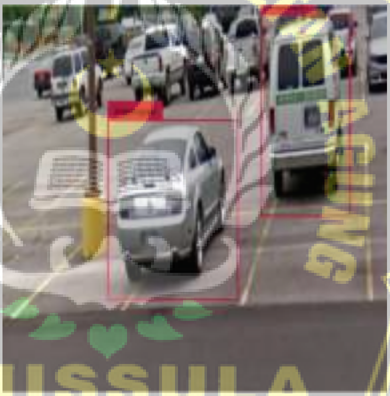
Distribusi jumlah data pada masing-masing subset ditunjukkan pada Tabel 4.3. Pembagian ini bertujuan agar setiap subset memiliki fungsi spesifik, di mana data latih digunakan untuk pembelajaran model, data validasi untuk memantau performa selama pelatihan, dan data uji untuk mengukur generalisasi model secara menyeluruh.

4.2.6. Hasil Visualisasi

Pada tahap ini dilakukan proses visualisasi data untuk memastikan bahwa anotasi yang telah dibuat dapat terbaca dengan baik oleh sistem. Visualisasi dilakukan dengan cara menampilkan citra dari *dataset* secara acak dan menambahkan bounding box beserta label kelas yang sesuai, yaitu *proper* dan *improper*. Proses ini memanfaatkan *library Supervision* yang berfungsi untuk menggambar *bounding box* sekaligus menampilkan label pada objek terdeteksi.



Gambar 4. 11 Visusalisasi Data *Class Proper*



Gambar 4. 12 Visualisasi Data *Class Improper*

Hasil visualisasi pada gambar 4.11 dan 4.12 menunjukkan bahwa setiap citra telah memiliki anotasi yang sesuai dengan posisi dan kategori objek di dalam gambar. Hal ini membuktikan bahwa proses anotasi, pembagian *dataset*, serta tahap preprocessing sebelumnya telah berjalan dengan baik. Dengan demikian, *dataset* yang digunakan siap untuk dilatih pada model DETR.

4.3. Hasil *Training* Model

4.3.1. Arsitektur Model

Arsitektur yang digunakan adalah *Detection Transformer* (DETR) dengan *backbone* ResNet-50. Hal ini konsisten dengan penjelasan pada Bab 3, dimana ResNet-50 berperan sebagai ekstraktor fitur, sedangkan *transformer* menangani mekanisme attention untuk mengenali objek. Implementasi dilakukan dengan memanfaatkan kelas DetrForObjectDetection dari pustaka HuggingFace Transformers, yang kemudian dibungkus ke dalam modul LightningModule. Penyesuaian dilakukan pada lapisan keluaran agar sesuai dengan dua kelas pada *dataset*, yaitu *proper* dan *improper*.

4.3.2. Inisialisasi dan *Hyperparameter*

Hyperparameter yang digunakan dalam pelatihan ditentukan saat inisialisasi kelas Detr. Beberapa pengaturan penting meliputi:

1. *Learning* Rate (LR): 1e-4 untuk parameter umum, sedangkan *backbone* diberikan nilai lebih kecil (1e-5) agar pembaruan bobot lebih stabil.
2. *Weight Decay*: 1e-3, digunakan untuk mengurangi risiko overfitting dengan menekan nilai bobot yang terlalu besar.
3. *Optimizer* & *Scheduler*: AdamW dipilih sebagai *optimizer*, dikombinasikan dengan *StepLR* untuk menurunkan nilai *learning* rate secara bertahap.
4. *Epoch*: Jumlah maksimum *epoch* ditetapkan sebanyak 30.

4.3.3. Strategi *Training*

Sejumlah strategi tambahan diterapkan untuk menjaga stabilitas pelatihan, antara lain:

1. Early Stopping disiapkan dengan parameter *patience=20*, meskipun pada implementasi akhir belum diaktifkan.
2. Gradient Clipping dengan nilai 0.1 untuk menghindari *exploding gradient*.
3. Accumulate Gradient dengan akumulasi 4 batch untuk mengatasi keterbatasan GPU.

4. Monitoring *Learning* Rate diaktifkan menggunakan callback *Learning*RateMonitor, walaupun tidak selalu ditampilkan pada log akhir.

4.3.4. Komputasi

Proses training dilakukan pada perangkat dengan dukungan GPU. Hal ini difasilitasi oleh *PyTorch Lightning Trainer* dengan pengaturan accelerator="gpu" dan devices=1. Penggunaan GPU secara signifikan mempercepat proses optimisasi dibandingkan CPU, terutama mengingat kompleksitas arsitektur DETR.

4.3.5. Output Training

Selama proses training, model menghasilkan metrik berupa loss training dan loss validasi, yang dicatat setiap *epoch* melalui mekanisme logging pada LightningModule. Selain itu, nilai rata-rata loss per *epoch* disimpan dalam variabel train_losses dan val_losses, sehingga memudahkan analisis perkembangan performa model dari waktu ke waktu. Hasil ini akan ditampilkan dalam bentuk grafik dan digunakan untuk mengevaluasi efektivitas strategi pelatihan yang diterapkan. *Training Model* dilakukan dengan beberapa percobaan *hyperparameter* terutama bagian *epoch*. Maka *output training* dikelompokkan berdasarkan *epoch* yang digunakan.

1. Percobaan *training model epoch* 20

   Pada model deteksi parkir tidak tertib ini dibuat dengan beberapa parameter yang sebelumnya sudah ditentukan nilainya, seperti pada tabel 4.4 sebagai berikut.

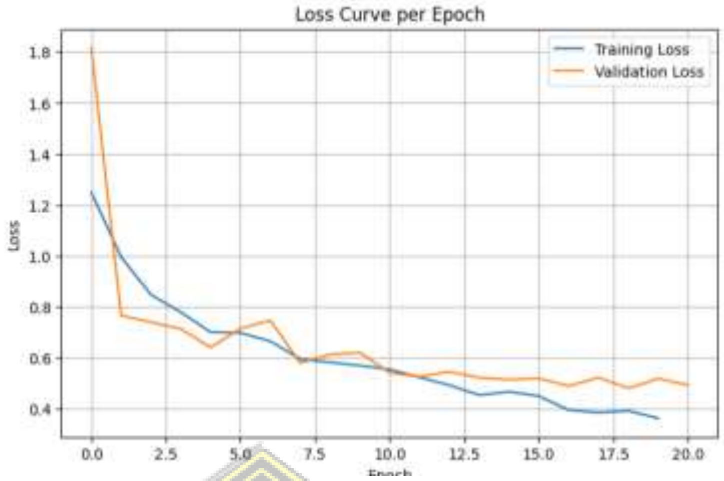
   Tabel 4. 4 *Hyperparameter epoch* 20

Parameter	Nilai
Batch Size	4
Epoch	20
Learning Rate	1e-4
Optimizer	AdamW

Tabel 4.4 menampilkan konfigurasi *hyperparameter* yang digunakan dalam proses pelatihan model deteksi aktivitas merokok. *Batch size* ditetapkan sebesar 4, yang berarti model memproses lima gambar sekaligus dalam satu iterasi selama pelatihan. Pemilihan *batch size* ini bertujuan untuk menjaga keseimbangan antara efisiensi komputasi dan stabilitas pembaruan bobot. Proses pelatihan dilakukan selama 20 *epoch*, di mana satu *epoch* merepresentasikan satu siklus penuh melalui seluruh *dataset* pelatihan, sehingga model memperoleh kesempatan yang lebih banyak untuk belajar dari data.

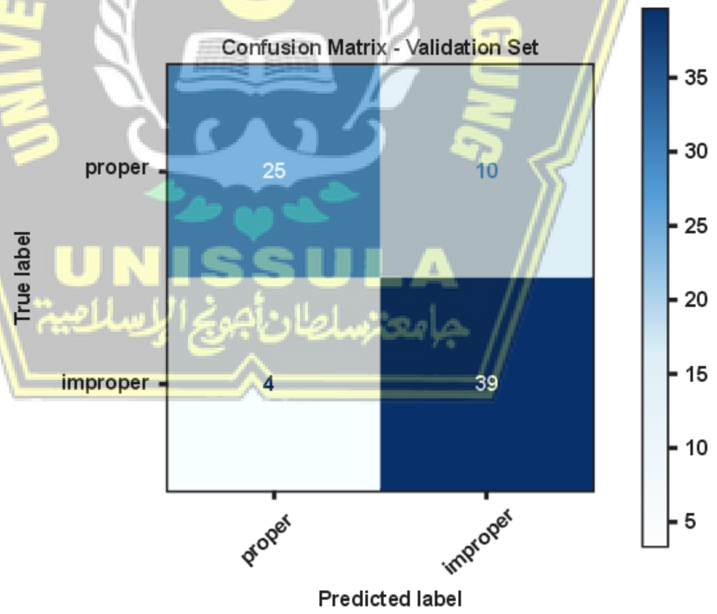
Nilai *learning rate* yang digunakan adalah sebesar 3×10^{-5} (0.00003). Nilai yang relatif kecil ini dipilih agar proses pembaruan bobot berlangsung secara bertahap dan stabil, sehingga dapat mengurangi risiko *overshooting* maupun kegagalan konvergensi. Adapun algoritma optimisasi yang digunakan adalah AdamW, yaitu varian dari Adam yang dilengkapi dengan mekanisme *weight decay* untuk mengatasi *overfitting*. Optimizer ini banyak digunakan dalam pelatihan model modern karena mampu memberikan konvergensi yang lebih baik dibandingkan optimisasi konvensional.

Setelah *Training Model* dilakukan, menunjukkan hasil dari *loss curve*. Grafik yang menunjukkan Hasil grafik memperlihatkan *training loss* yang turun secara signifikan sedangkan *validation loss* tidak seperti *training loss* seperti pada gambar 4.13 ebagai berikut.



Gambar 4. 13 *Loss Curve Epoch* 20

Evaluasi juga dilakukan pada *Training Model* menggunakan *confusion matrix* untuk melihat hasil dari *training* model. Evaluasi dengan *confusion matix* ini menggunakan data dari validasi yang berjumlah 72. Terlihat pada gambar 4.14 sebagai berikut.



Gambar 4. 14 Hasil *Confusion Matrix Epoch* 20

Gambar 4.14 menunjukkan hasil evaluasi dengan *confusion matrix* yang lumayan baik, dikarenakan nilai *false* pada *class proper* hanya 10 sedangkan pada *class improper* hanya 4.

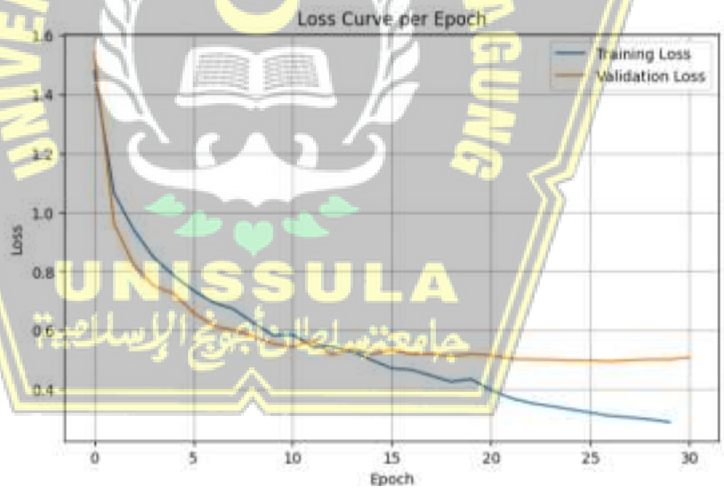
2. Percobaan *training model epoch* 30

Pada percobaan *training model* untuk *epoch* 30 menggunakan *hyperparameter* seperti pada tabel 4.5 sebagai berikut.

Tabel 4. 5 *Hyperparameter epoch* 30

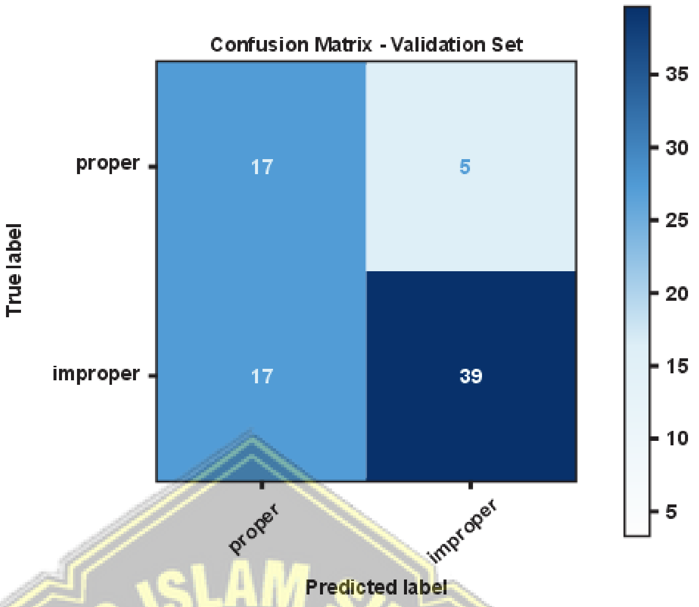
Parameter	**Nilai**
Batch Size	4
Epoch	30
Learning Rate	1e-4
Optimizer	AdamW

Pada percobaan ini, model dilatih selama 30 kali penuh melalui dataset, memungkinkan pembelajaran yang lebih mendalam dibandingkan epoch sebelumnya, dan diharapkan mampu meningkatkan akurasi serta kemampuan generalisasi model terhadap data baru.



Gambar 4. 15 *Loss Curve Epoch* 30

Pada gambar 4.15 terlihat pada grafik *loss curve* masih menunjukkan hasil *training model* yang *overfitting*. Terlihat pada *Validation Loss* 0.5 dan *Training Loss* 0.3 sehingga menimbulkan *overfitting*.



Gambar 4. 16 Hasil *Confusion Matrix Epoch* 30

Pada gambar 4.16 menunjukkan hasil evaluasi *confusion matrix* dengan hasil yang kurang memuaskan karena miss pada *true label improper* yang terdeteksi *proper.*

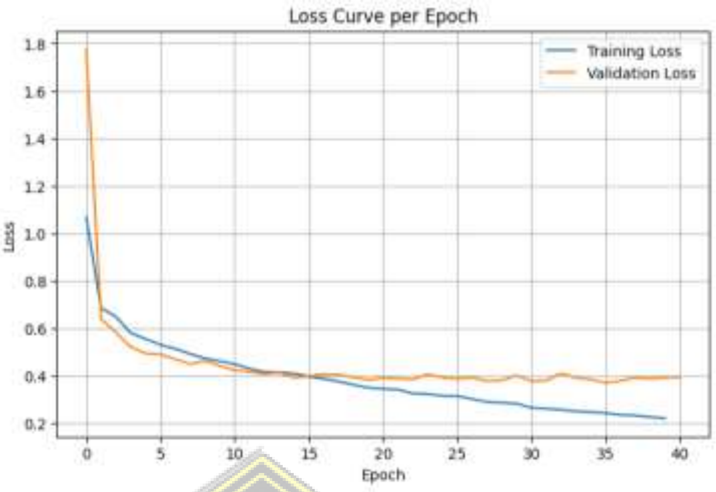
3. Percobaan *training model epoch* 40

Pada percobaan *training model* untuk *epoch* 40 menggunakan *hyperparameter* seperti pada tabel 4.5 sebagai berikut.

Tabel 4. 6 *Hyperparameter epoch* 40

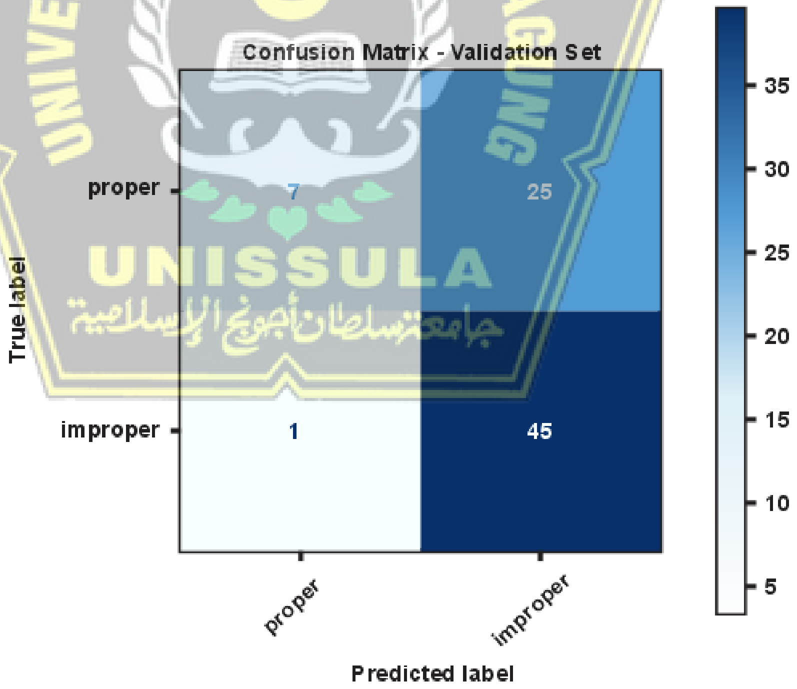
Parameter	**Nilai**
Batch Size	4
Epoch	40
Learning Rate	1e-4
Optimizer	AdamW

Pelatihan dilakukan selama 40 siklus penuh dataset, bertujuan memperkuat pemahaman model terhadap fitur penting dari data, serta meningkatkan stabilitas dan performa deteksi.



Gambar 4. 17 *Loss Curve Epoch* 40

Pada gambar 4.17 terlihat pada grafik *loss curve* masih menunjukkan hasil *training model* yang lumayan membaik dari *epoch* 30. Tetapi terlihat pada *Validation Loss* 0.4 dan *Training Loss* 0.2 masih terlihat *overfitting*.



Gambar 4. 18 Hasil *Confusion Matrix Epoch* 40

Pada gambar 4.18 menunjukkan hasil evaluasi *confusion matrix* dengan hasil yang tidak memuaskan karena data bias ke *class improper* sehingga hasil *true* pada *class proper* hanya 7.

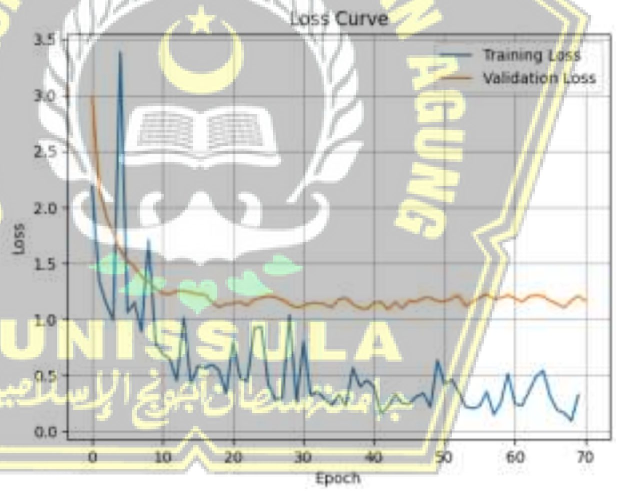
4. Percobaan *training model epoch* 70

Pada percobaan *training model* untuk *epoch* 70 menggunakan *hyperparameter* seperti pada tabel 4.5 sebagai berikut.

Tabel 4. 7 *Hyperparameter epoch* 70

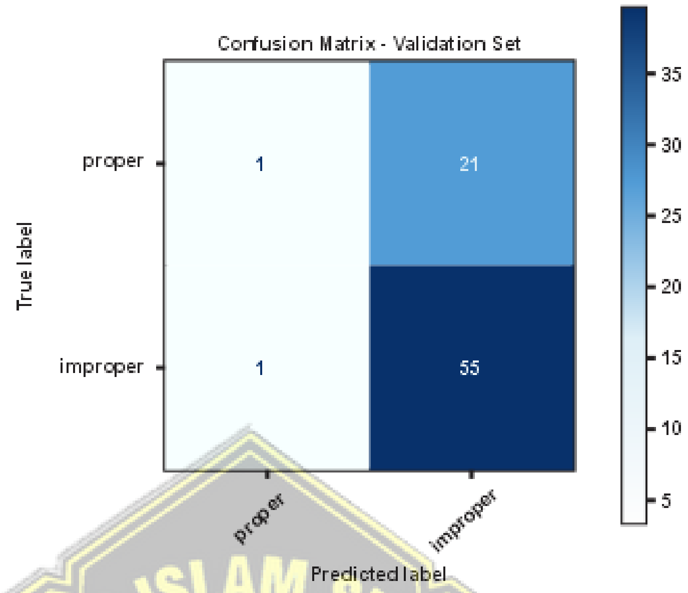
Parameter	**Nilai**
Batch Size	4
Epoch	70
Learning Rate	1e-4
Optimizer	AdamW

Model dilatih lebih lama, yaitu selama 70 epoch, untuk mencoba mengoptimalkan kemampuan belajar dan meminimalisasi bias serta overfitting, demi mendapatkan hasil yang lebih baik.



Gambar 4. 19 *Loss Curve Epoch* 40

Pada gambar 4.19 terlihat pada grafik *loss curve* menunjukkan hasil yang berantakan, terutama pada grafik *Training Loss* yang tidak stabil dan *overfitting*.



Gambar 4. 20 Hasil *Confusion Matrix Epoch* 70

Pada gambar 4.20 menunjukkan hasil evaluasi *confusion matrix* dengan hasil yang lebih parah daripada *epoch* 40 karena data bias ke *class improper* sehingga hasil *true* pada *class proper* hanya 1, hasil ini sangat tidak baik digunakan pada uji coba.

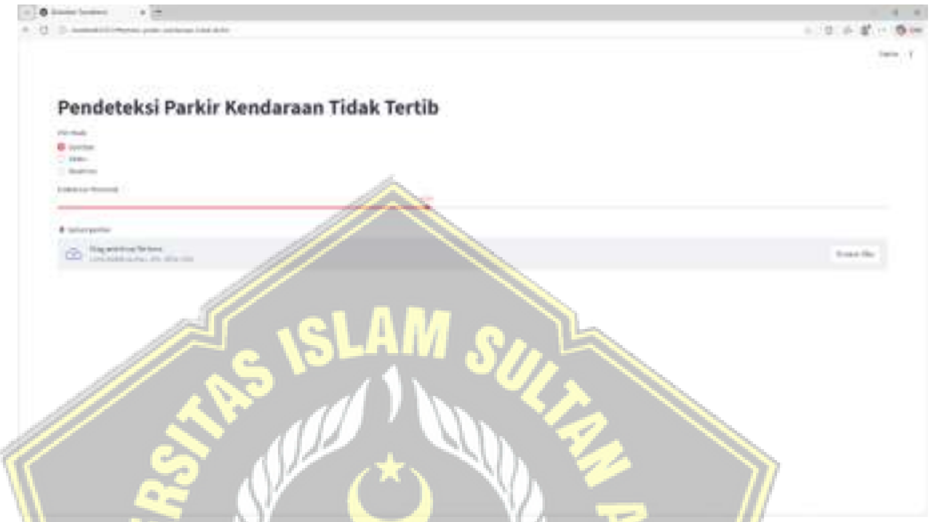
Setelah beberapa kali percobaan, penulis menyimpulkan bahwa pada percobaan *epoch* 20 yang memiliki hasil *balance*. Pada *loss curve* dan *confusion matrix* terlihat tidak buruk dan memberikan performa yang lumayan baik. Oleh karena itu, penulis akan menggunakan komposisi *hyperparameter* seperti pada tabel 4.8 sebagai berikut.

Tabel 4. 8 Komposisi *Hyperparameter* untuk Pengujian

Parameter	**Nilai**
Batch Size	4
Epoch	20
Learning Rate	1e-4
Optimizer	AdamW

4.4. Hasil Pengujian

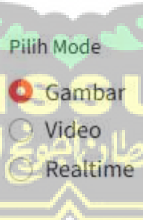
Model yang sudah dilatih dan didapatkan hasil pelatihan paling baik akan diimplementasikan pada *streamlit* dengan tampilan *website*. Penggunaan *streamlit* dilakukan untuk memudahkan pengimplementasian secara *real-time* deteksi parkir kendaraan,



Pendeteksi Parkir Kendaraan Tidak Tertib

Gambar 4. 21 Tampilan implementasi pada *streamlit*

Pada gambar 4.21 merupakan tampilan webdari pendeteksi parkir kendaraan tidak tertib.

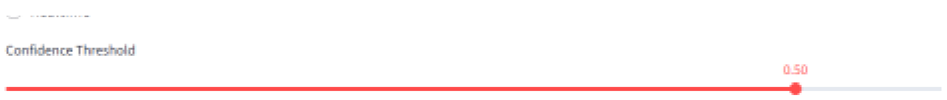


Pilih Mode

● Gambar
○ Video
○ Realtime

Gambar 4. 22 Fitur pada implementasi model

Pada gambar 4.22 merupakan tampilan implementasi yang memiliki 3 fitur, yaitu Gambar, Video, Realtime. Inference memilki 3 pilihan agar *website* fleksibel dalam melakukan uji coba model yang sudah dilatih.



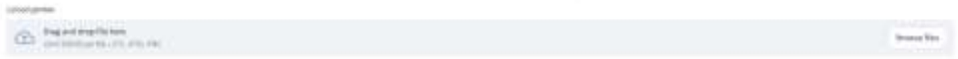
Confidence Threshold

0.50

Gambar 4. 23 Fitur *treshold*

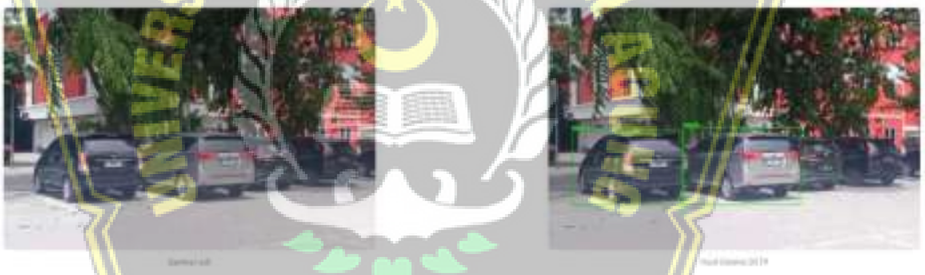
Gambar 4.23 merupakan tampilan fitur *threshold* yang disediakan untuk menyesuaikan dengan kebutuhan dan hasil. Semakin rendah threshold

yang diatur maka *bounding box* yang muncul juga semakin banyak dan jika semakin tinggi maka *bounding box* akan sedikit, karena tinggi rendah nya threshold juga mempengaruhi akurasi hasil.



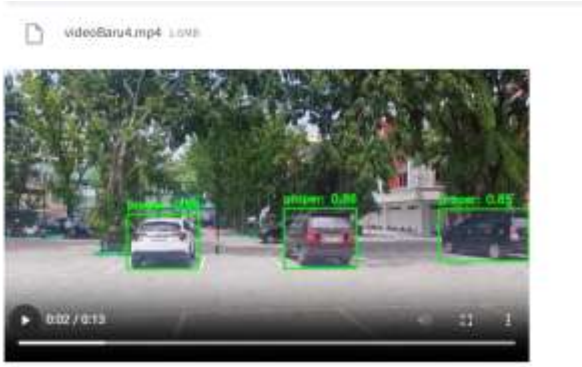
Gambar 4. 24 Fitur *upload file* pada *Inference* gambar dan video

Pada gambar 4.24 menunjukkan tombol untuk *upload file*, tombol ini ada pada fitur Gambar dan Video karena membutuhkan *file* pada *device* yang akan dideteksi, berbeda dengan *real-time* karena deteksi dilakukan secara langsung.



Gambar 4. 25 tampilan hasil pada fitur gambar

Gambar 4.25 menunjukkan hasil uji coba dengan fitur Gambar, terlihat ada 2 gambar yaitu gambar pertama merupakan gambar asli dan yang kedua merupakan gambar hasil setelah dideteksi.

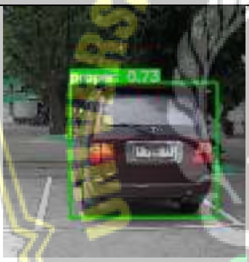
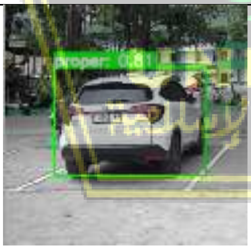
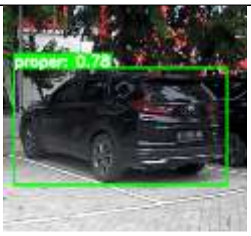


Gambar 4. 26 Tampilan hasil pada fitur video

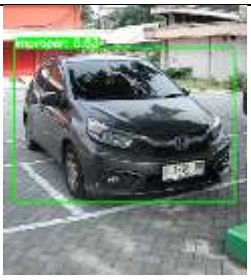
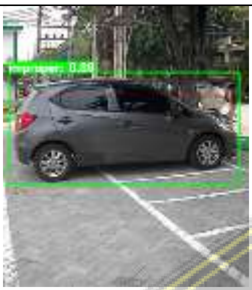
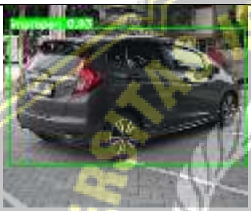
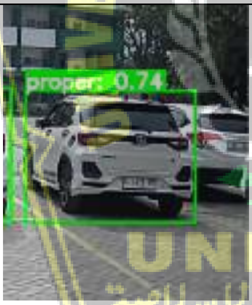
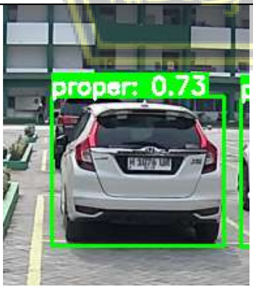
Pada gambar 4.26 merupakan tampilan hasil pada fitur Video, output yang dihasilkan berupa 1 file video yang bisa langsung diputar pada aplikasi dan dapat di-*download* jika diperlukan.

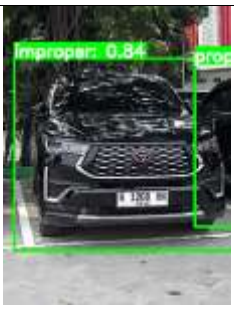
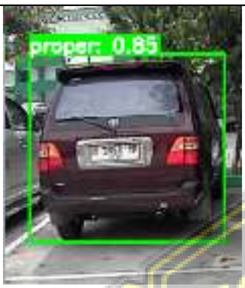
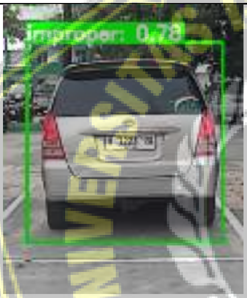
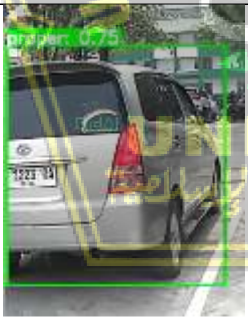
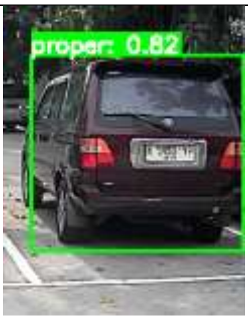
Fitur *Realtime* menjadi pilihan saat penulis melakukan uji coba model. Karena dengan adanya fitur *Realtime*, dapat diketahui dan diambil kesimpulan bahwa model dapat bekerja secara langsung. Pengujian secara *real-time* dilakukan pada area parkir kampus Unissula, data diambil dari parkiran Fakultas Teknologi Industri, Fakuktas Teknik, dan Kedai Khumaira. Jumlah data yang diambil berjumlah 50 data dan diperoleh hasil pengujian sebagaimana dalam tabel 4.9 sebagai berikut.

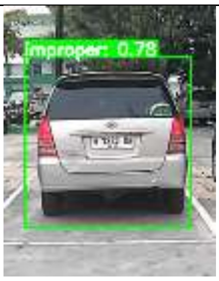
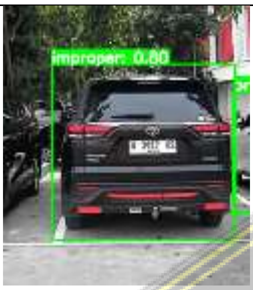
Tabel 4. 9 Tabel Hasil Pengujian

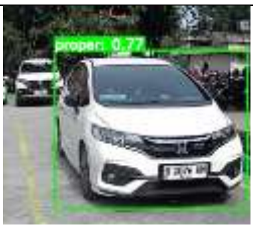
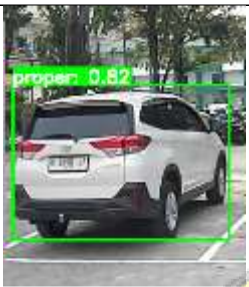
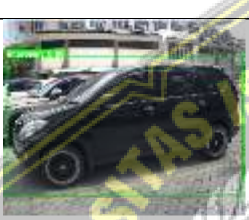
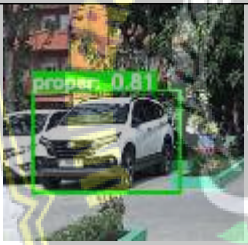
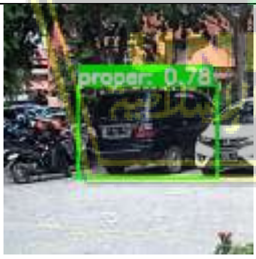
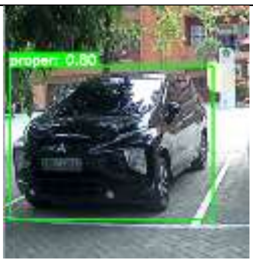
No	Hasil	*Pred*	*Act*	*Conf. Score*	*Threshold*				
					0.5	0.6	0.7	0.8	0.9
1		Tertib	Tertib	0.75	TP	TP	TP	FN	FN
2		Tertib	Tertib	0.81	TP	TP	TP	TP	FN
3		Tertib	Tertib	0.78	TP	TP	TP	FN	FN

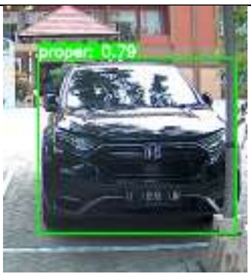
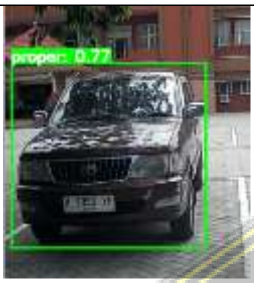
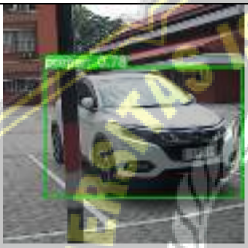
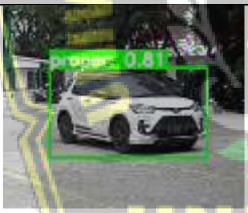
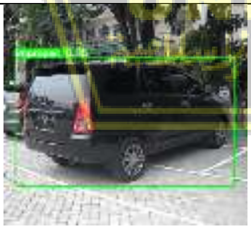
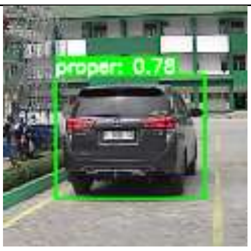
4		Tertib	Tidak Tertib	0.80	FP	FP	FP	FP	TN
5	improper: 0.72	Tidak Tertib	Tidak Tertib	0.72	TN	TN	TN	FP	FP
6		Tidak Tertib	Tidak Tertib	0.88	TN	TN	TN	TN	FP
7		Tidak Tertib	Tidak Tertib	0.80	TN	TN	TN	TN	FP
8		Tertib	Tertib	0.81	TP	TP	TP	TP	FN
9		Tertib	Tertib	0.81	TP	TP	TP	TP	FN

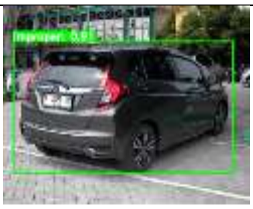
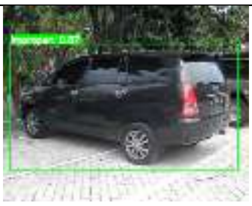
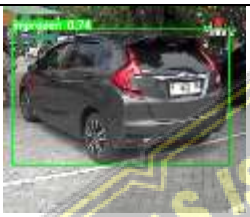
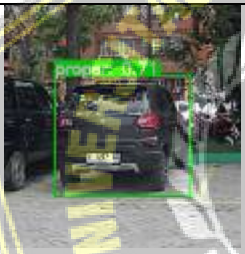
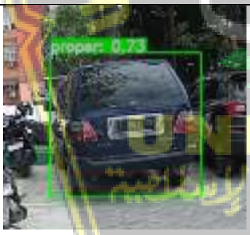
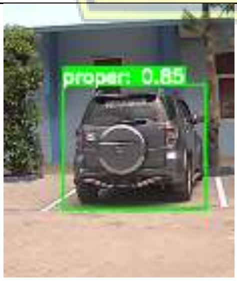
10		Tidak Tertib	Tidak Tertib	0.83	TN	TN	TN	TN	FP
11		Tidak Tertib	Tidak Tertib	0.89	TN	TN	TN	TN	FP
12		Tidak Tertib	Tidak Tertib	0.93	TN	TN	TN	TN	FP
13		Tertib	Tertib	0.74	TP	TP	TP	FN	FN
14		Tertib	Tertib	0.73	TP	TP	TP	FN	FN

15	 improper: 0.84 prop	Tidak Tertib	Tertib	0.84	FN	FN	FN	FN	TP
16		Tertib	Tertib	0.85	TP	TP	TP	TP	FN
17	 improper: 0.78	Tidak Tertib	Tertib	0.78	FN	FN	FN	TP	TP
18		Tertib	Tertib	0.75	TP	TP	TP	FN	FN
19		Tertib	Tertib	0.82	TP	TP	TP	TP	FN

20		Tidak Tertib	Tertib	0.78	TN	TN	TN	FP	FP
21		Tidak Tertib	Tertib	0.80	FN	FN	FN	FN	TP
22		Tidak Tertib	Tidak Tertib	0.88	TN	TN	TN	TN	FP
23		Tidak Tertib	Tidak Tertib	0.85	TN	TN	TN	TN	FP
24		Tidak Tertib	Tidak Tertib	0.91	TN	TN	TN	TN	TN
25		Tidak Tertib	Tidak Tertib	0.84	TN	TN	TN	TN	FP

26		Tertib	Tertib	0.77	TP	TP	TP	FN	FN
27		Tertib	Tertib	0.82	TP	TP	TP	TP	FN
28		Tidak Tertib	Tidak Tertib	0.86	TN	TN	TN	TN	FP
29		Tertib	Tertib	0.81	TP	TP	TP	TP	FN
30		Tertib	Tertib	0.78	TP	TP	TP	FN	FN
31		Tertib	Tertib	0.80	TP	TP	TP	TP	FN

32		Tertib	Tertib	0.79	TP	TP	TP	FN	FN
33		Tertib	Tertib	0.77	TP	TP	TP	FN	FN
34		Tertib	Tertib	0.78	TP	TP	TP	FN	FN
35		Tertib	Tertib	0.81	TP	TP	TP	TP	FN
36		Tidak Tertib	Tidak Tertib	0.85	TN	TN	TN	TN	FP
37		Tertib	Tertib	0.78	TP	TP	TP	FN	FN

38		Tidak Tertib	Tidak Tertib	0.91	TN	TN	TN	TN	TN
39		Tidak Tertib	Tidak Tertib	0.87	TN	TN	TN	TN	FP
40		Tidak Tertib	Tidak Tertib	0.78	TN	TN	TN	FP	FP
41		Tertib	Tertib	0.71	TP	TP	TP	FN	FN
42		Tertib	Tertib	0.73	TP	TP	TP	FN	FN
43		Tertib	Tertib	0.85	TP	TP	TP	TP	FN

44		Tertib	Tertib	0.88	TP	TP	TP	TP	FN
45		Tertib	Tertib	0.81	TP	TP	TP	TP	FN
46		Tertib	Tertib	0.91	TP	TP	TP	TP	TP
47		Tertib	Tertib	0.83	TP	TP	TP	TP	FN
48	.82 proper: 0.91	Tertib	Tertib	0.91	TP	TP	TP	TP	TP

49		Tertib	Tertib	0.89	TP	TP	TP	TP	FN
50		Tertib	Tertib	0.89	TP	TP	TP	TP	FN

4.5. Hasil Evaluasi

Evaluasi model ini dilakukan dengan menggunakan beberapa metrik penting, yaitu *accuracy*, *precision*, *recall*, dan *f1-score*. Metrik accuracy dihitung menggunakan rumus nomor 5 dan berfungsi untuk mengukur seberapa besar proporsi prediksi yang benar dibandingkan dengan seluruh data yang diuji. Semakin tinggi nilai *accuracy* yang diperoleh, maka semakin baik pula kinerja model secara keseluruhan karena menunjukkan kemampuan model dalam mengenali pola secara umum. *Precision* dihitung menggunakan rumus nomor 6 dan digunakan untuk melihat seberapa akurat model saat memprediksi kelas positif, yaitu berapa banyak prediksi positif yang benar dibandingkan seluruh prediksi positif. Semakin tinggi nilai *precision*, maka semakin sedikit prediksi positif yang salah dilakukan oleh model, sehingga tingkat kesalahan dalam mendeteksi kelas positif dapat diminimalkan. *Recall*, yang dihitung menggunakan rumus nomor 7, digunakan untuk mengukur seberapa banyak data positif yang berhasil terdeteksi oleh model dibandingkan dengan seluruh data positif yang ada. Nilai *recall* yang tinggi menunjukkan bahwa semakin sedikit data positif yang terlewatkan atau tidak terdeteksi oleh model.

Selain itu, *f1-score* yang dihitung dengan rumus nomor 8 merupakan metrik penting yang memberikan gambaran kinerja model secara lebih menyeluruh. *f1-score* menghitung rata-rata harmonis antara *precision* dan *recall* sehingga memberikan keseimbangan antara kedua metrik tersebut. Nilai *f1-score* yang tinggi menunjukkan bahwa model mampu menjaga keseimbangan antara kemampuan mendeteksi data positif (*recall*) dan ketepatan dalam melakukan prediksi positif (*precision*), sehingga kinerjanya dapat dinilai lebih stabil dan andal dalam menghadapi data yang tidak 39 seimbang atau imbalanced *dataset*. Berdasarkan hasil pengujian yang telah dilakukan, diperoleh nilai dari masing-masing metrik ini yang mencerminkan performa model dalam memproses data uji. Hasil evaluasi ini menjadi landasan penting untuk menilai apakah model sudah layak digunakan pada aplikasi nyata atau masih memerlukan perbaikan lebih lanjut, baik dari sisi arsitektur, data latih, maupun parameter training yang digunakan. Adapun dari hasil pengujian yang dilakukan diperoleh hasil sebagai berikut:

1. Pada nilai *threshold* 0.5 menghasilkan sebanyak 30 *true positive*, 1 *false positive*, 16 *true negative*, dan 3 *false negative*. Sehingga diperoleh nilai *accuracy* sebesar 92%, *precission* sebesar 96,7%, *recall* sebesar 90,9%, dan *f1-score* sebesar 93,7%.
2. Pada nilai threshold 0.6 menghasilkan sebanyak 30 *true positive*, 1 *false positive*, 16 *true negative*, dan 3 *false negative*. Sehingga diperoleh nilai *accuracy* sebesar 92%, *precission* sebesar. 96,7%, *recall* sebesar 90,9%, dan *f1-score* sebesar 93,7%.
3. Pada nilai threshold 0.7 menghasilkan sebanyak 30 *true positive*, 1 *false positive*, 16 *true negative*, dan 3 *false negative*. Sehingga diperoleh nilai *accuracy* sebesar 92%, *precission* sebesar 96,7%, *recall* sebesar 90,9%, dan *f1-score* sebesar 93,7%.
4. Pada nilai threshold 0.8 menghasilkan sebanyak 19 *true positive*, 3 *false positive*, 13 *true negative*, dan 15 *false negative*. Sehingga diperoleh nilai *accuracy* sebesar 64%, *precission* sebesar 86,3%, *recall* sebesar 80,7%, dan *f1-score* sebesar 73,4%.
5. Pada nilai threshold 0.9 menghasilkan sebanyak 5 *true positive*, 13 *false positive*, 3 *true negative*, dan 29 *false negative*. Sehingga diperoleh nilai

accuracy sebesar 16%, *precission* sebesar 27,8%, *recall* sebesar 14,7%, dan *f1-score* sebesar 19,2%.

Tabel 4. 10 Hasil Evaluasi

Nilai	Threshold				
	0.5	0.6	0.7	0.8	0.9
True Positive	30	30	30	19	5
False Positive	1	1	1	3	13
True Negative	16	16	16	13	3
False Negative	3	3	3	15	29
Accuracy	0.92	0.92	0.92	0.64	0.16
Precission	0.967	0.967	0.967	0.863	0.278
Recall	0.909	0.909	0.909	0.807	0,147
F1-Score	0.937	0.937	0.937	0.734	0.192

Tabel 4.6 merupakan hasil performa model dilakukan dengan mengamati pengaruh perubahan nilai threshold terhadap sejumlah metrik evaluasi, yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), *False Negative* (FN), *accuracy*, *precision*, *recall*, dan *f1-score*. Berdasarkan hasil yang diperoleh,terlihat bahwa nilai True Positive (TP) stabil di angka 30 pada rentang 0,5 hingga 0,7, kemudian turun drastis menjadi 5 pada rentang 0,9, menunjukkan bahwa model lebih konservatif dalam membuat prediksi positif. Tingkat False Positive (FP) berkisar antara 1 hingga ambang 0,7, kemudian turun menjadi 3 pada ambang 0,8 dan 13 pada ambang 0,9, menunjukkan bahwa risiko meningkat pada ambang tinggi. Pada tingkat tertinggi, nilai Negatif Palsu (TN) turun dari 16 menjadi 3. Penurunan TP dan kenaikan FN menyebabkan akurasi, presisi, recall, dan skor f1 menurun secara signifikan pada ambang batas. Karena memberikan keseimbangan metrik terbaik, ambang batas 0,5–0,7 direkomendasikan untuk model tersebut.

Model memiliki *accuracy* stabil pada *threshold* 0.5, 0.6, dan 0.7 yaitu 92% yang menunjukkan kinerja model yang baik dan konsisten dalam mendeteksi objek. Namun, pada *threshold* 0.8, *accuracy* turun pada menjadi 64% hingga turun drastis pada threshold 0.9 yaitu 16%, hal ini meunjukkan penurunan performa yang sangat signifikan pada *threshold* 0.9. Pada *precission* tetap tinggi untuk *threshold* 0.5, 0.6, dan 0.7 yaitu 96,7%. Untuk *threshold* 0.8 menurun pada 86,3% dan turun lagi sangat rendah pada *threshold* o.9 yaitu 27,8%. Model juga memiliki *recall* yang tinggi pada *threshold* 0.5, 0.6, dan 0.7 stabil pada 90,9% sebelum turun ke 80,7% pada *threshold* 0.8 dan akhirnya turun ke 14,7% pada *threshold* 0.9. Menunjukkan bahwa kemampuan model model mendeteksi telah menurun. Karena adanya penurunan dari *recall*, maka *f1-score* turun dari 93,7% pada *threshold* 0.5, 0.6, dan 0.7 kemudian menjadi 73,4% pada *threshold* 0.8 dan *threshold* 0.9 akhirnya turun ke 19,2%. Oleh karena itu, ambang 0,5–0,7 disarankan sebagai ambang keputusan terbaik karena memberikan keseimbangan terbaik antara ketepatan, recall, dan ketepatan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa model deteksi objek dengan menggunakan *Detection Transformer* (DETR) dan *backbone* ResNet-50 menunjukkan performa terbaik pada *threshold* 0,5 hingga 0,7. Pada rentang threshold ini, model mempertahankan nilai *accuracy* sekitar 92%, *precision* 96,7%, *recall* 90,9%, dan *f1-score* 93,7%, yang menunjukkan keseimbangan baik antara kemampuan mengenali objek dan meminimalkan kesalahan prediksi. Namun, peningkatan *threshold* ke 0,8 dan 0,9 menyebabkan penurunan signifikan pada *recall*, *accuracy*, dan *f1-score*, walaupun precision menurun secara drastis pada threshold tertinggi. Hal ini menunjukkan model menjadi terlalu selektif dan kehilangan banyak deteksi positif pada threshold tinggi. Oleh karena itu, *threshold* 0,5 sampai 0,7 direkomendasikan sebagai ambang keputusan optimal untuk implementasi model ini, guna menjaga keseimbangan sensitivitas dan ketepatan deteksi.

5.2 Saran

Berikut adalah saran yang dapat dipertimbangkan untuk pengembangan dan perbaikan model deteksi parkir tidak tertib di masa mendatang. Meskipun model DETR dengan backbone ResNet-50 telah menunjukkan performa yang baik pada *threshold* 0,5, masih terdapat beberapa aspek yang dapat ditingkatkan untuk mencapai akurasi dan generalisasi yang lebih baik pada berbagai kondisi nyata. Selain itu, pemilihan *threshold* yang optimal juga perlu disesuaikan dengan kebutuhan aplikasi, apakah lebih mementingkan sensitivitas atau ketepatan.

Beberapa saran pengembangan ke depan adalah sebagai berikut:

1. Eksplorasi backbone alternatif selain ResNet-50, misalnya ResNet-101, Swin Transformer, atau ConvNeXt, guna memperoleh representasi fitur yang lebih kaya dan akurat.
2. Penerapan teknik augmentasi data yang lebih variatif, sehingga model mampu beradaptasi dengan kondisi data nyata yang beragam dan meningkatkan kemampuan generalisasi.
3. Implementasi model dalam bentuk aplikasi berbasis web atau mobile, sehingga dapat dimanfaatkan secara praktis untuk sistem pemantauan maupun deteksi otomatis.

DAFTAR PUSTAKA

Arkin, E., Yadikar, N., Xu, X., Aysa, A., & Ubul, K. (2023). A survey: object detection methods from CNN to transformer. *Multimedia Tools and Applications*, *82*(14), 21353–21383. https://doi.org/10.1007/s11042-022-13801-3

Ayudhawara, A., Poetro, B. S. W., & Qomaruddin, M. (2023). Deteksi Ketersediaan Tempat Parkir menggunakan Mask R-CNN (Studi Kasus : Fakultas Teknologi Industri Universitas Islam Sultan Agung). Dalam *Seminar Riset Mahasiswa-Computer & Electrical (SERIMA-CE* (Vol. 1, Nomor 1).

Babatunde, O. H., Armstrong, L., Leng, J., & Diepeveen, D. (2015). A survey of computer-based vision systems for automatic identification of plant species. *Journal of Agricultural Informatics*, *6*(1). https://doi.org/10.17700/jai.2015.6.1.152

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). *End-to-End Object Detection with Transformers*. http://arxiv.org/abs/2005.12872

Dompeipen, T. A., Sompie, S. R. U. A., & Najoan, M. E. I. (2021). *Computer Vision Implementation for Detection and Counting the Number of Humans*.

Huangfu, Y., Huang, Z., Yang, X., Zhang, Y., Li, W., Shi, J., & Yang, L. (2024). HHS-RT-DETR: A Method for the Detection of Citrus Greening Disease. *Agronomy*, *14*(12). https://doi.org/10.3390/agronomy14122900

Jrondi, Z., Moussaid, A., & Hadi, M. Y. (2024). Exploring End-to-End object detection with transformers versus YOLOv8 for enhanced citrus fruit detection within trees. *Systems and Soft Computing*, *6*. https://doi.org/10.1016/j.sasc.2024.200103

Khoiriyah, R., & Hendrawan, A. (2024). Vehicle Detection on The Traffic Using Detection Transformer (DETR) Algorithm. *International Journal of Artificial Intelligence and Science*, *1*(1), 14–24. https://doi.org/10.63158/IJAIS.v1.i1.4

Mustafa, O., Ali, K., Bibi, A., Siddiqi, I., & Moetesum, M. (2024). *Object Detection for Vehicle Dashcams using Transformers*. http://arxiv.org/abs/2408.15809

Nurza, W. A., Hapsari, G. I., & Alfarisi, M. R. (2020). *Pendeteksi Pelanggar Rambu Larangan Parkir dan Berhenti di Kawasan Telkom University*.

Ponti, M. A., Ribeiro, L. S. F., Nazare, T. S., Bui, T., & Collomosse, J. (2017). *Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask*.

Rizkatama, G. N., Nugroho, A., & Suni, A. F. (2021). Edu Komputika Journal Sistem Cerdas Penghitung Jumlah Mobil untuk Mengetahui Ketersediaan Lahan Parkir berbasis Python dan YOLO v4. *Edu Komputika*, *8*(2). http://journal.unnes.ac.id/sju/index.php/edukom

Rosnita, L., Retno, S., & Hariono, S. N. (2024). *Penerapan Sistem Deteksi Pengisian Ruang Parkir Kendaraan Roda 4 Menggunakan Metode Computer Vision di Orbit Future Academy* (Vol. 15).

Yusfian, M. H., Setianingsih, C., & Astuti, R. (2022). *Deteksi Pelanggaran Parkir Pada Bahu Jalan Tol Dengan Intelligent Transportation System MenggunakanParking Violation Detection On The Roadside Of Toll Roads With Intelligent Transportation System Using Yolo Algorithm)*.