# **SKRIPSI**

**PERBANDINGAN METODE DETEKSI OBJEK DALAM CITRA PENCAHAYAAN RENDAH MENGGUNAKAN METODE CNN BASED DAN TRANSFORMER BASED**

****

**RIDHO PANDHU AFRIANTO**

**NIM. 162112133062**

**PROGRAM SARJANA**

**TEKNOLOGI SAINS DATA**

**DEPARTEMEN TEKNIK**

**FAKULTAS TEKNOLOGI MAJU DAN MULTIDISIPLIN**

**UNIVERSITAS AIRLANGGA**

**2025**

# **LEMBAR PENGESAHAN**

**PERBANDINGAN METODE DETEKSI OBJEK DALAM CITRA PENCAHAYAAN RENDAH MENGGUNAKAN CNN BASED DAN TRANSFORMER BASED**

|  |  |
| --- | --- |
| Nama | : Ridho Pandhu Afrianto |
| NIM | : 162112133062 |
| Tanggal Seminar Proposal | : Kamis, 3 Oktober 2024 |

Surabaya, 24 Oktober 2024

|  |  |
| --- | --- |
| Pembimbing I  **Dr. Aziz Fajar, S.Kom., M.Kom.** | Pembimbing II  **Ratih Ardiati Ningrum, S.Si., M.Stat.** |
| **NIP.** 199410052024023101 | **NIP.** 199501262020013201 |

Mengetahui,

Koordinator Program Studi

S1 Teknologi Sains Data

**Dr. Dwi Rantini, S.Si.**

**NIP.** 199406152022033201

# **KATA PENGANTAR**

Puji syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga saya dapat menyelesaikan proposal skripsi ini yang berjudul "Perbandingan Metode Deteksi Objek dalam Citra Pencahayaan Rendah Menggunakan CNN-*Based* dan Transformer-*Based*". Proposal ini disusun sebagai salah satu syarat untuk menempuh seminar proposal skripsi di Program Studi Teknologi Sains Data, Fakultas Sains dan Teknologi, Universitas Airlangga.

Saya menyadari bahwa dalam penyusunan proposal ini, banyak pihak yang telah memberikan bantuan dan dukungan, baik berupa bimbingan, saran, maupun dorongan moral. Oleh karena itu, saya mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Aziz Fajar dan Ibu Ratih Ardiati Ningrum, selaku dosen pembimbing, atas arahan, bimbingan, serta kesabarannya dalam membimbing saya selama penyusunan proposal ini.
2. Keluarga tercinta, yang selalu memberikan doa dan dukungan tanpa henti.
3. Rekan-rekan mahasiswa dan semua pihak yang tidak dapat saya sebutkan satu per satu, atas dukungan dan bantuan yang telah diberikan.

Saya berharap proposal ini dapat memberikan kontribusi positif, khususnya dalam pengembangan ilmu pengetahuan di bidang teknologi sains data, serta bermanfaat bagi para pembaca. Akhir kata, saya menyadari bahwa proposal ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat saya harapkan demi perbaikan dan penyempurnaan di masa mendatang.

Surabaya, 25 September 2024

Ridho Pandhu Afrianto

# **DAFTAR ISI**

[**HALAMAN JUDUL** i](#_Toc181134033)

[**LEMBAR PENGESAHAN** ii](#_Toc181134034)

[**KATA PENGANTAR** iii](#_Toc181134035)

[**DAFTAR ISI** iv](#_Toc181134036)

[**DAFTAR TABEL** v](#_Toc181134037)

[**DAFTAR GAMBAR** vi](#_Toc181134038)

[**BAB 1 PENDAHULUAN** 1](#_Toc181134039)

[1.1 Latar Belakang 1](#_Toc181134040)

[1.2 Rumusan Masalah 5](#_Toc181134041)

[1.3 Hipotesis 5](#_Toc181134042)

[1.4 Tujuan Penelitian 6](#_Toc181134043)

[1.5 Manfaat Penelitian 6](#_Toc181134044)

[1.6 Batasan Masalah 6](#_Toc181134045)

[**BAB 2 TINJAUAN PUSTAKA** 7](#_Toc181134046)

[2.1 *Computer Vision* 7](#_Toc181134047)

[2.2 Analisis *Object Detection* 8](#_Toc181134048)

[2.3 Data *Preprocessing* 8](#_Toc181134049)

[2.3.1 *Contrast Enhancement* 9](#_Toc181134050)

[2.3.2 Normalisasi Data 10](#_Toc181134051)

[2.3.3 Augmentasi Data 12](#_Toc181134052)

[2.4 *Convolutional Neural Network* 13](#_Toc181134053)

[2.5 *You Only Look Once* 15](#_Toc181134054)

[*2.*6 Transformer 17](#_Toc181134055)

[2.7 CNN-Transformer (YOLOv10) 20](#_Toc181134056)

[2.8 *Hyperparameter Tuning* 21](#_Toc181134057)

[2.9 *Loss Function* 23](#_Toc181134058)

[2.10 Metrik Evaluasi Model 26](#_Toc181134059)

[2.11 Pareto Frontier 28](#_Toc181134060)

[**BAB 3 METODE PENELITIAN** 28](#_Toc181134061)

[3.1 Lokasi dan Waktu 28](#_Toc181134062)

[3.2 Bahan dan Alat 28](#_Toc181134063)

[3.3 Cara Kerja 29](#_Toc181134064)

[3.3.1 *Data Preprocessing* 31](#_Toc181134065)

[3.3.2 Implementasi Model 34](#_Toc181134066)

[3.3.3 Metode Perbandingan Model 41](#_Toc181134067)

[**DAFTAR PUSTAKA** 47](#_Toc181134068)

# **DAFTAR TABEL**

[Tabel 3.1 Deskripsi Parameter Augmentasi 34](#_Toc180616121)

[Tabel 3.2 Tipe Model YOLOv9 36](#_Toc180616122)

[Tabel 3.3 Tipe Model RT-DETR 38](#_Toc180616123)

[Tabel 3.4 Tipe Model YOLOv10 40](#_Toc180616124)

[Tabel 3.5 Deskripsi *Hyperparameter* yang Diuji 40](#_Toc180616125)

# **DAFTAR GAMBAR**

Gambar 1. 1 1

(a) Citra berpencahayaan rendah 1

(b) Contoh hasil dari model deteksi objek 1

[Gambar *2.* 1 *Object detection* dengan satu atau lebih objek 8](#_Toc180613991)

[Gambar 2. 2 Proses Konvolusi 2D 14](#_Toc180613992)

[Gambar 2. 3 Sistem deteksi objek dengan YOLO 15](#_Toc180613993)

[Gambar 2. 4 Arsitektur YOLO 15](#_Toc180613994)

[Gambar 2. 5 Arsitektur YOLOv9 16](#_Toc180613995)

[Gambar 2. 6 Arsitektur Transformer 18](#_Toc180613996)

[Gambar 2. 7 Arsitektur RT-DETR 19](#_Toc180613997)

[Gambar 2. 8 Arsitektur YOLOv10 21](#_Toc180613998)

[Gambar 2. 9 Perbedaan *ℓ2 loss dan IoU loss* 24](#_Toc180613999)

[Gambar 3.1 Sampel *Exclusively Dark* dataset.......................................................29](#_Toc180614704)

[Gambar 3.2 *Flowchart* Keseluruhan Proses 31](#_Toc180614705)

[Gambar 3. 3 Proses Inferensi Model berbasis CNN 35](#_Toc180614706)

[Gambar 3. 4 Proses Inferensi Model Berbasis Transformer (RE-DETR) 37](#_Toc180614707)

[Gambar 3. 5 Proses Inferensi Model berbasis CNN-Transformer (YOLOv10) 39](#_Toc180614708)

# **BAB I PENDAHULUAN**

## 1.1 Latar Belakang

 Dalam dunia keamanan modern, penggunaan CCTV telah menjadi pilihan utama dalam aktivitas di berbagai ruang publik dan privat, mulai dari pemantauan lalu lintas hingga perlindungan objek vital seperti bandara dan gedung pemerintahan. Namun, meskipun teknologi CCTV telah berkembang, masalah mendasar terkait deteksi objek dalam kondisi pencahayaan rendah tetap menjadi kendala signifikan. Pada malam hari atau di area yang minim penerangan, CCTV konvensional sering kali gagal menangkap gerakan mencurigakan dengan jelas, sehingga memberikan celah bagi tindakan kejahatan, termasuk terorisme. Kasus-kasus seperti pemboman dan serangan di ruang publik yang terekam CCTV menunjukkan bahwa saat ini identifikasi objek yang ada di dalamnya masih mengalami situasi yang sulit dan lambat​ (Putri, 2021). Keterbatasan ini menyoroti perlunya pengembangan lebih lanjut dalam teknologi pengawasan yang mampu beradaptasi dengan berbagai kondisi cahaya untuk memastikan keamanan maksimal.

(b)

(a)

(b)



(a)

(a)

(a)

Gambar 1.1 (a) Citra berpencahayaan rendah, (b) Contoh hasil dari model deteksi objek

Sumber: Loh & Chan (2018)

Salah satu permasalahan dari perangkat yang memerlukan waktu kerja 24 jam non-stop seperti CCTV, adalah kemampuannya untuk mendeteksi sebuah objek jika objek tersebut berada di pencahayaan yang rendah atau malam hari. Berdasarkan Gambar 1.1 (a) sudah dapat disimpulkan bahwa belum tentu manusia dapat mengetahui objek apa saja yang berada di gambar tersebut apalagi mengawasinya selama 24 jam. Oleh karena itu, perlu bantuan model deteksi objek seperi yang dicontohkan dalam Gambar 1.1 (b) untuk melakukan tugas demikian. Menurut Chen & Shah (2021), faktor-faktor seperti *noise, glare, lousy illumination, low contrast, shadows,* dan *reflectance* membuat deteksi objek dalam kondisi tersebut menjadi sulit. Berbagai penelitian dengan pendekatan yang beragam telah dilakukan untuk mengatasi masalah tersebut. Sebagian besar penelitian dalam peningkatan citra berfokus pada mempercantik subjek dan meningkatkan kualitas estetika gambar, dibandingkan dengan pemulihan fitur yang diperlukan untuk analisis lebih lanjut. Beberapa penelitian juga mencoba mengatasi masalah tersebut dengan menggunakan *hardware* yang lebih canggih, seperti kamera yang dilengkapi dengan sensor inframerah dan gambar termal. Namun, metode ini memiliki biaya yang tinggi dan sering kali menghasilkan citra yang kurang realistis (Chen & Shah, 2021). Salah satu pendekatan untuk meningkatkan kecerahan dalam kondisi pencahayaan rendah adalah dengan menaikkan ISO atau memperpanjang *exposure time*. Namun, strategi ini masing-masing cenderung meningkatkan *noise* dan menyebabkan *motion blur* (Chen *et al.,* 2018). Pendekatan lain yang dapat diterapkan adalah memanfaatkan *software* seperti Photoshop atau Lightroom untuk melakukan penyesuaian pencahayaan secara digital. Meskipun demikian, *software* ini memerlukan keterampilan artistik dan tidak efisien untuk dataset skala besar dengan kondisi pencahayaan yang beragam (Zheng & Gupta, 2022). Selebihnya, pendekatan-pendekatan tersebut menjadi kurang layak untuk aplikasi seperti CCTV yang memerlukan operasi non-stop selama 24 jam dan menuntut kecepatan maupun ketepatan deteksi.

Menyikapi masalah tersebut, Loh & Chan (2018) membuat sebuah dataset bernama ExDark atau *Exclusively Dark* untuk memfasilitasi para peneliti dalam memahami fenomena citra di pencahayaan rendah. ExDark memiliki 7363 citra yang terdiri dari citra berpencahayaan rendah hingga citra di berpencahayaan fajar dengan 12 kelas objek di dalamnya sehingga cocok untuk riset berbasis aplikasi seperti *object detection* (Loh & Chan, 2018). Dataset ExDark ini kemudian diadopsi oleh model PE-YOLO atau *Pyramid Enhancement - You Only Look Once* yang secara khusus dirancang untuk mendeteksi objek dalam pencahayaan rendah dengan menggunakan pendekatan *Convolutional Neural Networks* (CNN) (Yin *et al.,* 2023). Kelemahan model ini adalah pada model *backbone* yang terbilang lama (YOLOv3), serta mekanisme di dalamnya yang masih memiliki keterbatasan dalam penanganan *noise* terutama pada frekuensi tinggi.  Pada penelitian lain, Cui dkk. (2022) mengembangkan model *Multitask Auto-Encoding Transformation* (MAET).  MAET menawarkan pendekatan baru dalam mendeteksi objek di kondisi pencahayaan rendah dengan memanfaatkan regularisasi ortogonal untuk memisahkan fitur degradasi pencahayaan dan deteksi objek. Model ini menggunakan pipeline *image signal processor* (ISP) yang lebih realistis untuk mensintesis gambar berpencahayaan rendah, yang memungkinkan model untuk mengoptimalkan deteksi objek di lingkungan gelap tanpa menghasilkan artefak yang sering terjadi pada metode pemulihan pencahayaan tradisional. Disamping skor mAP yang bernilai 0,74 pada dataset ExDark, terdapat beberapa kelemahan yang membatasi model ini  (Cui *et al.,* 2022). Penggunaan regularisasi ortogonal menambah kompleksitas komputasi yang dapat menghambat implementasi model ini pada perangkat dengan sumber daya terbatas. Selain itu, MAET lagi-lagi juga masih mengandalkan YOLOv3 sebagai *backbone* yang merupakan arsitektur lama. Menggunakan pendekatan yang berbeda,  Cui dkk. (2022) melakukan penelitian lain dengan mekanisme Transformer untuk mengembangkan *Illumination Adaptive Transformer* (IAT) untuk memulihkan gambar sRGB dengan pencahayaan normal dari kondisi *low-light* atau *under/overexposure*. IAT mampu mencapai skor mAP sebesar 77,2 pada dataset ExDark dengan *time inference* 0,04 detik berkat komponen *simplified image signal processor* (ISP) yang dimilikinya (Cui *et al.,* 2022). Komponen *Simplified* ISP inilah yang membantu IAT mampu merealisasikan inferensi yang sangat cepat dengan parameter yang relatif kecil yaitu sebesar 90.000 parameter. Namun, justru dari *simplified* ISP inilah IAT mengalami keterbatasan generalisasi ketika diterapkan pada dataset dengan kondisi yang beragam.

Kedua pendekatan yang populer pada *object detection*, yaitu CNN dan Transformer telah terbukti bahwa selalu ada *trade-off* masing-masing. Oleh karena itu, akan sangat menarik untuk mengeksplorasi apakah penggunaan kedua pendekatan yang berbeda pada model-model *state of the art* terbaru berpotensi menghasilkan peningkatan performa yang signifikan dan *trade-off*  yang lebih optimal dibandingkan dengan penelitian-penelitian terdahulu. Definisi optimal dalam konteks ini adalah  model yang mampu memberikan skor *mean Average Precision* (mAP) yang tinggi sembari meminimalkan waktu inferensi. Model-model *state-of-the-art* saat ini yang menggunakan pendekatan CNN, Transformer, dan Hybrid (CNN-Transformer) adalah YOLOv9 yang dirancang oleh Wang dkk. (2024) , RT-DETR  oleh Zhao dkk. (2023), dan YOLOv10 oleh peneliti lain Wang dkk. (2024). Pemilihan model-model ini didasarkan tidak hanya pada statusnya sebagai model *state-of-the-art* saat ini, tetapi juga karena kemampuan mereka dalam melakukan deteksi multi-objek secara *real-time* dengan kecepatan yang tinggi, menjadikannya ideal untuk aplikasi yang membutuhkan deteksi cepat dan akurat.

Dalam membandingkan dua jenis metrik yang berbeda, pendekatan dengan perbandingan secara langsung sangat tidak memungkinkan untuk dilakukan karena mAP diukur dalam skala skor sementara waktu inferensi diukur dalam satuan waktu (*ms*). Oleh karena itu, pendekatan yang tepat untuk mengevaluasi kedua metrik ini adalah dengan menggunakan *Pareto Frontier*. *Pareto Frontier* memungkinkan evaluasi *trade-off* antara dua atau lebih metrik, di mana sebuah solusi dianggap optimal jika tidak ada solusi lain yang lebih baik dalam kedua metrik secara bersamaan. Perbandingan antara mAP dan waktu inferensi menjadi sangat penting, karena perangkat seperti CCTV memerlukan model yang tidak hanya cepat tetapi juga akurat.

Penelitian ini ditujukan untuk memberikan kontribusi yang signifikan dalam pengembangan metode deteksi objek pada citra berpencahayaan rendah dengan mengeksplorasi dan membandingkan model berbasis CNN dan pendekatan Transformer. Dengan mengatasi tantangan yang dihadapi oleh model deteksi objek dalam kondisi pencahayaan rendah, penelitian ini dapat membantu mengarahkan untuk meningkatkan akurasi serta kecepatan deteksi dalam berbagai aplikasi dunia nyata, seperti pengawasan CCTV dan industri lainnya yang bergantung pada deteksi objek secara *real-time*. Tujuan dari penelitian ini adalah dapat membantu mengidentifikasi solusi yang lebih efisien dan efektif baik secara biaya maupun sumber daya, terutama dalam kondisi pencahayaan yang beragam. Selebihnya, optimalisasi pada penelitian ini juga diharapkan dapat menjadi solusi alternatif pengganti *thermal-cam* dan *infrared* untuk CCTV konvensional yang ada di Indonesia.

## 1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana performa metode YOLOv9 terhadap objek dengan pencahayaan rendah dan apa faktor yang menyebabkannya?
2. Bagaimana performa metode RT-DETR terhadap objek dengan pencahayaan rendah dan apa faktor yang menyebabkannya?
3. Bagaimana performa metode YOLOv10 terhadap objek dengan pencahayaan rendah dan apa faktor yang menyebabkannya?
4. Apa metode yang terbaik dari segi *mean Average Precision* (mAP) dan inferensi waktuuntuk mendeteksi objek berpencahayaan rendah?

## 1.3 Hipotesis

Metode penggabungan CNN & Transformer akan menjadi yang paling unggul dibandingkan dengan metode CNN dan Transformer secara terpisah. Hal ini didasarkan pada kemampuan CNN-Transformer untuk mengintegrasikan keunggulan dari kedua metode, yaitu kemampuan CNN dalam mendeteksi objek dengan cepat dan kemampuan Transformer dalam mengenali fitur-fitur secara semantik.

## 1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Untuk mengetahui performa metode YOLOv9 terhadap objek dengan pencahayaan rendah dan mencari faktor yang menyebabkannya.
2. Untuk mengetahui performa metode RT- DETR terhadap objek dengan pencahayaan rendah dan mencari faktor yang menyebabkannya.
3. Untuk mengetahui performa metode YOLOv10 terhadap objek dengan pencahayaan rendah dan mencari faktor yang menyebabkannya.
4. Untuk mencari metode yang terbaik dalam *mean Average Precision* (mAP) dan inferensi waktudalam mendeteksi objek dengan pencahayaan rendah.

## 1.5 Manfaat Penelitian

Manfaat yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Menjadi referensi bagi penelitian selanjutnya dalam pengembangan model berbasis CNN ataupun Transformer untuk mendeteksi objek.
2. Model yang lebih optimal dalam mendeteksi objek dalam pencahayaan rendah akan bermanfaat untuk meningkatkan kapabilitas *hardware* dengan sumber daya terbatas dalam mendeteksi objek.

## 1.6 Batasan Masalah

Batasan masalah pada penelitian ini jumlah kelas yang dapat dideteksi hanya 12 kelas karena sesuai dengan data yang didapat dari ExDark.

# **BAB II TINJAUAN PUSTAKA**

## 2.1 *Computer Vision*

*Computer vision* adalah kemampuan komputer untuk memahami dan menganalisis data visual, terutama dari gambar atau video, dengan tujuan mengidentifikasi objek, pola, dan fitur lainnya. Ini adalah cabang ilmu komputer yang berkaitan dengan pengembangan sistem yang memungkinkan komputer untuk “melihat” dan memahami dunia sekitarnya melalui data visual yang diberikan kepadanya (Wibowo, 2016).

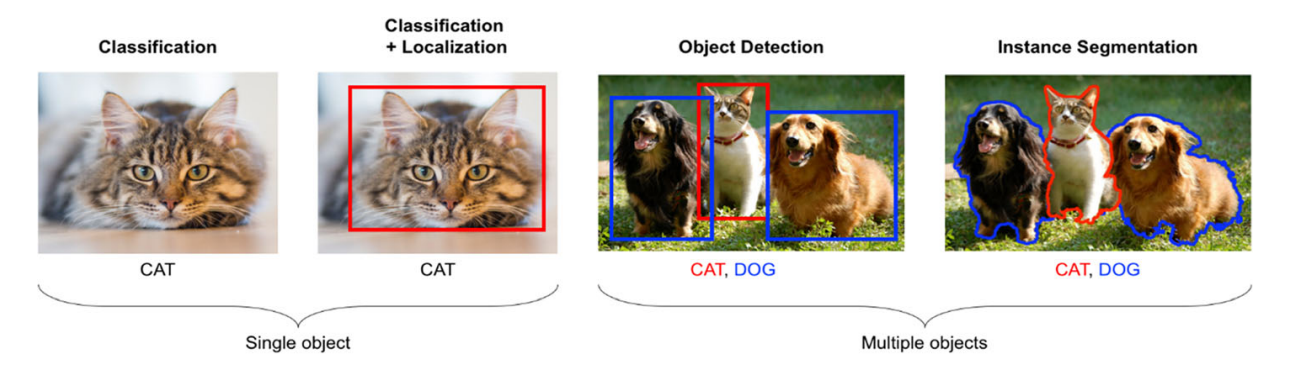
*Computer Vision* telah menjadi topik yang luas, meliputi mulai dari merekam data mentah hingga ekstraksi pola gambar dan interpretasi informasi. Ini menggabungkan konsep, teknik, dan ide dari pemrosesan gambar digital, pengenalan pola, kecerdasan buatan, dan grafika komputer. Sebagian besar tugas dalam *computer vision* berkaitan dengan proses mendapatkan informasi tentang peristiwa atau deskripsi dari gambar digital dan ekstraksi fitur. Tujuan utama *computer vision* adalah menciptakan model dan mengekstrak data serta informasi dari gambar, sementara pemrosesan gambar berkaitan dengan menerapkan transformasi komputasi pada gambar*. Computer vision* juga memiliki keterkaitan dengan *Human-Computer Interaction* (HCI), yang fokus pada desain, *interface*, dan interaksi antara manusia dan komputer (Wiley & Lucas, 2018).

Meskipun fungsinya serupa, sistem *computer vision* memiliki keterbatasan dibandingkan dengan mata manusia, terutama dalam hal sensitivitas parameter, kekuatan algoritma, dan akurasi hasil. Beberapa tantangan signifikan dalam teknik mereka termasuk pengukuran kinerja dan evaluasi algoritma untuk mencapai akurasi, kekuatan, atau skalabilitas. Ada upaya yang komprehensif untuk mengembangkan dan mengategorikan *computer vision* ke dalam berbagai aplikasi spesifik, seperti otomatisasi garis perakitan, sensor jarak jauh, robotika, komunikasi komputer dan manusia, serta alat bagi kaum difabel visual (Wiley & Lucas, 2018).

## 2.2 Analisis *Object Detection*

Deteksi objek merupakan proses penting dalam *computer vision* yang bertujuan untuk mengidentifikasi instansi dari objek-objek semantik tertentu dalam gambar digital dan video. Salah satu pendekatan umum dalam deteksi objek adalah dengan menciptakan sejumlah besar *candidate windows* yang kemudian dianalisis menggunakan fitur-fitur *Convolutional Neural Networks* (Voulodimos, 2018).

Pendeteksian objek digunakan untuk mengklasifikasikan dan menemukan lokasi setiap objek dalam kotak pembatas atau *bounding box*. Penelitian oleh Lee (2017) menghasilkan sebuah metode bernama *sliding window* yang mampu mengurangi kompleksitas waktu yang sudah menjadi teknik dasar untuk mendeteksi objek. Dalam metode ini, sebuah *window* atau jendela dengan ukuran misalnya M x N, dipilih untuk mencari di atas gambar objektif. Pada awalnya, sebuah pengklasifikasi disiapkan pada serangkaian gambar *train*, menyebar di atas objek yang dituju untuk dideteksi sebagai satu kelas dan objek tidak teratur sebagai kelas yang berbeda. Tes yang termasuk dalam objek yang dituju untuk dideteksi disebut gambar positif, sementara sampel tidak teratur disebut gambar negatif (Lee *et al.*, 2017).



Gambar 2. 1 Object detection dengan satu atau lebih objek

Sumber: Diwan, *et al.* (2022)

## 2.3 Data *Preprocessing*

Pada domain *computer vision, data preprocessing* adalah langkah penting yang bertujuan untuk meningkatkan akurasi dan efisiensi model. Dalam banyak studi terbaru, seperti yang dijelaskan oleh Rouhbakhshmeghrazi & Alizadeh (2023), teknik ini mencakup serangkaian metode seperti augmentasi gambar, peningkatan kontras, dan pengurangan *noise*, yang diterapkan untuk mempersiapkan gambar sebelum proses pelatihan model. Sesuai dasar tersebut, penelitian ini akan melakukan *data preprocessing* berupa normalisasi data dan augmentasi data.

### 2.3.1 Contrast Enhancement

*Contrast Enhancement* merupakan teknik dalam pemrosesan citra yang bertujuan untuk meningkatkan kualitas visual dengan memperbesar perbedaan intensitas antara objek dan latar belakangnya. Salah satu metode yang paling sering digunakan untuk meningkatkan kontras suatu citra adalah CLAHE.

*Contrast Limited Adaptive Histogram Equalization* (CLAHE) adalah teknik peningkatan gambar yang bertujuan untuk memperbaiki kontras pada area gambar dengan menggunakan histogram *equalization,* tetapi dengan batasan untuk menghindari amplifikasi *noise* secara berlebihan. Berbeda dengan *histogram* *equalization* biasa, CLAHE membagi gambar menjadi blok-blok kecil dan melakukan proses *equalization* secara lokal, yang membantu dalam mengatasi masalah *over-amplification* pada area terang atau gelap. Seperti yang dijelaskan oleh Yusuf dkk. (2024), CLAHE telah digunakan dalam berbagai aplikasi untuk meningkatkan visibilitas objek pada gambar yang memiliki variasi pencahayaan, seperti pada deteksi target menggunakan citra UAV. Teknik ini membantu meningkatkan detail gambar sambil mengurangi *noise* yang tidak diinginkan.

Selain itu, Manongga dkk. (2024) juga menunjukkan bahwa CLAHE digunakan dalam deteksi tanda jalan di kondisi pencahayaan rendah, dimana kombinasi CLAHE dan YOLOv7 membantu meningkatkan deteksi secara signifikan. CLAHE memastikan bahwa objek pada gambar yang memiliki kontras rendah tetap terdeteksi dengan jelas tanpa menghasilkan distorsi berlebih. Pendekatan ini sangat penting dalam aplikasi *real-time* yang memerlukan pengolahan gambar yang cepat dan akurat.

### 2.3.2 Normalisasi Data

Normalisasi data adalah langkah dalam *pre-processing* data yang digunakan untuk meningkatkan kinerja model *machine learning*. Proses ini melibatkan transformasi data sehingga setiap fitur memiliki skala yang seragam, yang membantu model untuk berfungsi dengan lebih baik dan menghasilkan hasil yang lebih akurat. Tanpa normalisasi, fitur-fitur dengan skala besar dapat mendominasi algoritma dan mengakibatkan model tidak mampu mempelajari data dengan efektif. Teknik normalisasi yang umum digunakan meliputi *min-max scaler*, *z-score normalization*, dan *normalization* berdasarkan distribusi. *Min-max scaler* mengubah data ke dalam rentang [0, 1] atau [-1, 1], sementara *z-score normalization* mengubah data sehingga memiliki *mean* 0 dan standar deviasi 1. Dalam penelitian ini, teknik normalisasi data yang digunakan adalah *min-max scaler* dengan persamaan yang dijelaskan sebagai berikut.

*=*  (2.1)

Penelitian terbaru menunjukkan bahwa normalisasi data sangat penting dalam berbagai aplikasi *machine learning*. Misalnya, dalam studi yang dilakukan oleh Peng dkk. (2023), normalisasi data digunakan untuk meningkatkan prediksi keletihan material dalam lingkungan ekstrem, menunjukkan bahwa data yang dinormalisasi mempermudah model untuk memprediksi dengan lebih akurat. Demikian juga, penelitian oleh Cao dkk. (2023) mengimplementasikan normalisasi data dalam pemodelan kognitif *fuzzy*, yang menunjukkan peningkatan interpretabilitas dan akurasi model. Studi ini menyoroti bahwa normalisasi tidak hanya meningkatkan kinerja model tetapi juga membantu dalam menangani data heterogen yang mungkin memiliki distribusi yang berbeda.

### 2.3.2 Downsampling

Dalam *preprocessing image data*, *downsampling* atau yang juga dikenal sebagai desimasi, merupakan proses fundamental dalam pengolahan citra digital yang bertujuan untuk mengurangi resolusi spasial suatu citra dengan mengurangi jumlah pikselnya (...).Proses ini menghasilkan representasi citra yang lebih kecil dan membutuhkan lebih sedikit sumber daya komputasi untuk diproses dan disimpan. Metode *downsampling* yang umum meliputi teknik seperti *nearest neighbor, bilinear,* dan *bicubic interpolation*. Masing-masing metode memiliki keunggulan dan keterbatasannya sendiri dalam hal kecepatan, kompleksitas, dan kualitas gambar yang dihasilkan.

Salah satu metode *downsampling* yang banyak digunakan saat ini adalah *Lanczos Resampling*. Metode ini merupakan salah satu metode interpolasi yang telah banyak diadopsi dalam proses downsampling karena kemampuannya menghasilkan citra dengan kualitas tinggi. Metode ini menggunakan fungsi sinc yang dilipat dengan sebuah *window* Lanczos, sehingga diperoleh *kernel support* terbatas yang mampu mengaproksimasi fungsi sinc (...). Secara matematis, kernel Lanczos didefinisikan sebagai:

(2.1)

Dimana *x* adalah jarak antara titik interpolasi dan sampel terdekat, sedangkan *a* merupakan parameter yang menentukan lebar *support kernel*. Fungi ini hanya bernilai ketika dan nol di luar interval tersebut, sehingga memiliki sifat lokal. Penggunaan fungsi sinc yang dilipat atau *windowed sinc* memungkinkan kernel Lanczos mendekati karakteristik ideal dari interpolasi sinyal secara teoritis, namun tetap efisien secara komputasi.

Lanczos resampling telah digunakan dalam berbagai aplikasi pengolahan citra, termasuk penajaman citra, analisis citra medis, dan penginderaan jauh. Dalam konteks penelitian terkini, Mottola dkk. (2021) menemukan bahwa interpolasi Lanczos paling efektif dalam mempertahankan informasi asli selama *resampling* citra CT untuk analisis radiomik pada pasien kanker ginjal. Selain itu, Li *et al.* (2023) dalam studi berjudul *Learning Steerable Function for Efficient Image Resampling* mengusulkan modifikasi kernel Lanczos dengan menambahkan adaptabilitas orientasi, sehingga mampu mengoptimalkan proses *resampling* pada citra medis dan citra penginderaan jauh dengan hasil yang lebih tajam dan detail. Sementara itu, penelitian oleh Sheibanifard dan Yu (2023) yang mengembangkan representasi neural implisit untuk data volumetrik juga mengintegrasikan modul *downsampling* berbasis Lanczos, yang secara signifikan mengurangi beban pelatihan dan penggunaan memori GPU tanpa mengorbankan kualitas visual. Hasil-hasil penelitian tersebut menegaskan metode Lanczos menjadi pilihan utama dalam *downsampling* karena kemampuannya dalam mempertahankan informasi frekuensi tinggi dan menghasilkan citra berkualitas tinggi.

### 2.3.3 Augmentasi Data

Augmentasi data adalah teknik yang digunakan dalam bidang *computer vision* untuk meningkatkan variasi data latih tanpa harus mengumpulkan data baru. Teknik ini menjadi sangat penting dalam pelatihan model *machine learning*, khususnya *deep learning*, yang membutuhkan jumlah data yang besar untuk mencapai kinerja optimal. Augmentasi data membantu mengatasi masalah *overfitting* dengan menyediakan variasi yang lebih besar dalam data latih, sehingga model dapat lebih general dan mampu menangani data baru dengan lebih baik.

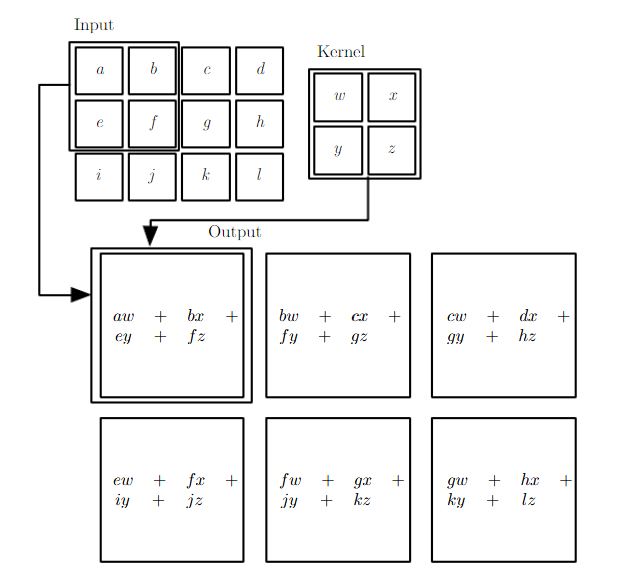
Beberapa teknik umum yang digunakan dalam augmentasi data meliputi rotasi, skala, translasi, *flipping*, pemotongan, perubahan kecerahan dan kontras, serta penambahan *noise*. Teknik ini memungkinkan model untuk belajar dari berbagai variasi gambar yang mungkin ditemui di dunia nyata. Studi terbaru oleh Shorten dan Khoshgoftaar (2019) menunjukkan bahwa augmentasi data dapat secara signifikan meningkatkan kinerja model *deep learning* dalam berbagai tugas *computer vision*. Selain itu, penelitian ini mengungkapkan bahwa kombinasi berbagai teknik augmentasi seringkali memberikan hasil terbaik.

Selain itu, terdapat jenis lain dari augmentasi data, yaitu *mosaic data augmentation*. Teknik ini merupakan teknik augmentasi yang menggabungkan potongan-potngna dari beberapa citra asli menjadi satu citra komposit. Teknik ini dirancang untuk menambah keragaman data *train* tanpa harus meningkatkan jumlah dataset secara signifikan. Sebagai contoh, dalam studi YOLOv4 yang dilakukan oleh Sandy dkk. (2024) , *mosaic augmentation* diterapkan dengan car amengintegrasikan empat citra secara simultan, sehingga menghasilkan variasi skala, latar belakang, dan konteks yang lebih kaya. Hasil penelitian tersebut menunjukkan bahwa penggunakan *mosaic data augmentation* mampu meningkatkan performa deteksi terutama dalam hal akurasi pendeteksian objek kecil dan membuat model lebih tahan terhadap variasi dalam *train set*.

## 2.4 *Convolutional Neural Network*

Jaringan saraf konvolusional atau *Convolutional Neural Network (CNN)* merupakan sebuah konsep yang diperkenalkan oleh LeCun dkk. pada tahun 1989. Penelitian tersebut mengacu pada penggunaan metode *backpropagation* yang diaplikasikan pada *Handwritten Zip Code* untuk *U.S. Postal Service*. Data yang digunakan pada penelitian tersebut berjumlah 9298 gambar tulisan tangan untuk sebuah kode pos yang ditulis oleh banyak orang yang berbeda, menggunakan berbagai macam ukuran, gaya penulisan, dan instrumen, dengan tingkat penulisan yang bervariasi. Pada penelitian tersebut, LeCun berhasil membuat sebuah jaringan tunggal yang dapat mempelajari seluruh operasi pengenalan, mulai dari gambar karakter yang dinormalisasi hingga klasifikasi akhir dan diberi nama *convolution networks* atau *convolutional neural network*.

Sesuai namanya, jaringan saraf konvolusional menunjukkan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Konvolusi adalah jenis operasi linier khusus. Jaringan *convolutional* hanyalah jaringan saraf yang menggunakan konvolusi sebagai pengganti perkalian matriks umum di setidaknya satu lapisannya (Goodfellow, 2016).



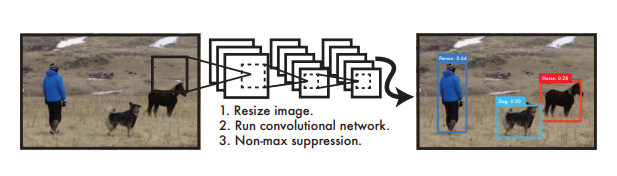
Gambar 2. 2 Proses Konvolusi 2D

Sumber: Goodfellow (2016)

Saat ini, CNN telah menjadi konsep yang sangat penting dalam bidang *computer vision* terutama dalam analisis citra. Arsitektur CNN meniru cara manusia memproses visual dengan mengadopsi konsep seperti *filter receptive field* dan penggabungan fitur hierarkis. Struktur CNN terdiri dari beberapa lapisan yang bekerja secara bersama-sama untuk mempelajari representasi fitur dari gambar input. Contohnya adalah lapisan konvolusi yang menggunakan *filter* atau *kernel* untuk memindai gambar dan mengekstrak fitur-fitur penting seperti tepi, tekstur, atau pola yang lebih kompleks. Setelah itu, lapisan aktivasi diterapkan untuk memperkenalkan non-linearitas ke dalam jaringan, yang membantu dalam pembelajaran representasi fitur yang lebih kompleks (Li *et al.*, 2021).

## 2.5 *You Only Look Once*

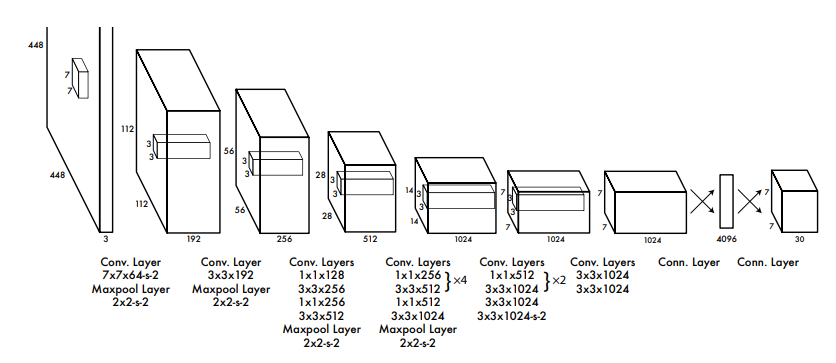
*You Only Look Once* (YOLO) adalah sebuah model deteksi objek revolusioner yang pertama kali diperkenalkan oleh Joseph Redmon dan Santosh Divvala pada tahun 2015. YOLO merombak pendekatan tradisional dalam deteksi objek dengan mengusulkan pendekatan yang lebih efisien dan cepat.



Gambar 2.3 Sistem deteksi objek dengan YOLO

Sumber: Redmon, *et al.* (2015)

Arsitektur YOLO terdiri dari beberapa lapisan konvolusi yang menghasilkan *grid* dari prediksi *bounding box* dan probabilitas kelas. Lapisan-lapisan konvolusi ini digabungkan dengan lapisan-lapisan pooling untuk menghasilkan representasi fitur dari gambar input. Kemudian, menggunakan *fully connected layer*, prediksi untuk *bounding box* dan probabilitas kelas dilakukan secara bersamaan (Redmon *et al.,* 2019).



Gambar 2.4 Arsitektur YOLO

Sumber: Redmon, *et al.* (2015)

Salah satu keunggulan utama dari YOLO adalah kemampuannya untuk mendeteksi objek dengan cepat dalam waktu *realtime*, karena memproses seluruh gambar dalam satu langkah. Hal ini membuatnya sangat cocok untuk aplikasi seperti deteksi objek dalam video atau pada perangkat dengan komputasi yang terbatas (Wang &Liao, 2024).



Gambar 2.5 Arsitektur YOLOv9

Sumber: Li, *et al.* (2024)

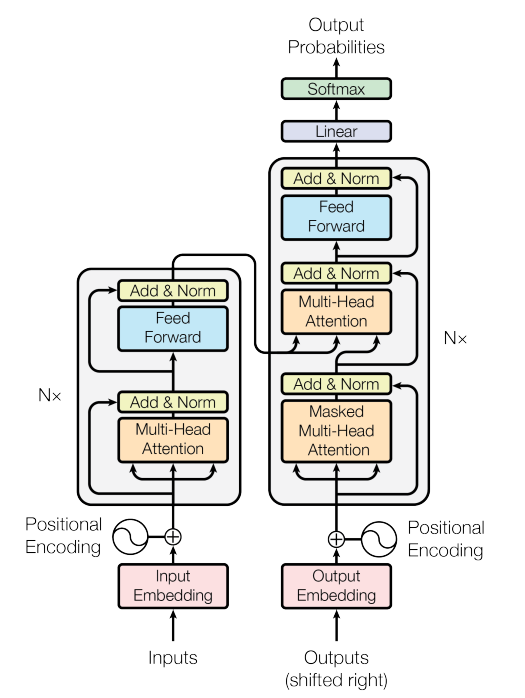
Seiring berkembangnya ilmu pengetahuan dan teknologi, muncul berbagai varian dari YOLO mulai dari YOLO9000 (YOLOv2), IA-YOLO, dan juga YOLOv9. YOLOv9 merupakan pengembangan dari sistem deteksi objek berbasis *Convolutional Neural Network* (CNN) yang mengombinasikan dua komponen utama, yaitu *Programmable Gradient Information* (PGI) dan *Generalized Efficient Layer Aggregation Network* (GELAN). Arsitektur YOLOv9 dimulai dengan pemrosesan input melalui serangkaian lapisan konvolusi pada GELAN, yang dirancang untuk memaksimalkan efisiensi jalur gradien dan parameter jaringan. PGI berperan dalam mengatasi kehilangan informasi yang sering terjadi selama proses *feedforward* dalam jaringan dalam (*information bottleneck*) dengan memperkenalkan mekanisme *reversible branch*. Mekanisme ini memastikan bahwa gradien yang diperoleh untuk memperbarui bobot jaringan optimal, sehingga jaringan dapat mempertahankan informasi penting dari fitur yang lebih dalam. Dalam YOLOv9, arsitektur GELAN menggunakan blok konvolusi yang dirancang untuk mencapai keseimbangan antara akurasi, *computational complexity*, dan *inference time*. Di tahap akhir, prediksi dilakukan menggunakan *head* deteksi yang telah dioptimalkan untuk efisiensi deteksi objek *multiscale*. Kombinasi PGI dan GELAN memungkinkan YOLOv9 untuk mencapai kinerja deteksi objek yang unggul dibandingkan dengan model sebelumnya, dengan parameter yang lebih sedikit namun akurasi yang lebih tinggi, khususnya pada dataset MS COCO (Wang *et al.,* 2024) .

Walau demikian, varian terbaru YOLO saat ini adalah YOLOv10 yang sudah menggunakan mekanisme Transformer di dalamnya. Setiap adanya kemunculan baru dari varian YOLO ini membuat YOLO menjadi lebih optimal yang secara spesifik dibutuhkan oleh pengembangnya. Jadi, bukan berarti karena YOLOv10 adalah versi terbaru, varian tersebut akan secara pasti mengungguli varian-varian lainnya dalam seluruh aspek maupun tugas yang dikerjakan, melainkan varian-varian dari YOLO ini akan menyesuaikan antara *trade-off* yang dibutuhkan. (Wang & Liao, 2024).

## *2.*6Transformer

Transformer merupakan sebuah arsitektur *neural network* yang menjadi *State of The Art* pada bidang *natural language processing* (NLP) sejak diperkenalkan oleh Vaswani dkk. (2017). Arsitektur Transformer menyajikan pendekatan yang berbeda dengan model *recurrent* atau CNN tradisional yang sebelumnya dominan dalam NLP.

Pusat dari Transformer adalah mekanisme *attention*, yang memungkinkan model untuk memberikan perhatian yang berbeda pada setiap bagian dari input *sequence*, terlepas dari jaraknya. Ini memungkinkan Transformer untuk memahami konteks semantik secara global dari sebuah *sequence*, yang merupakan keuntungan besar dalam NLP.

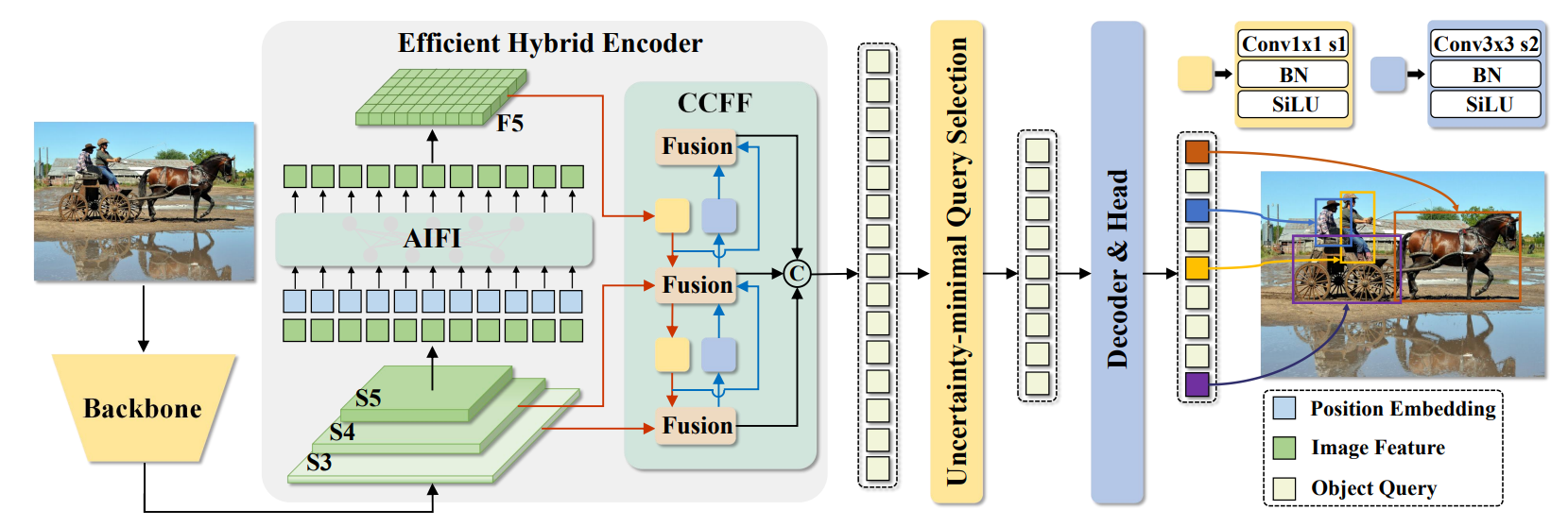


Gambar 2.6 Arsitektur Transformer

Sumber: Vaswani, *et al.* (2017)

Transformer terdiri dari dua komponen utama yaitu *encoder* dan *decoder*. *Encoder* bertugas untuk mengubah *input* *sequence* menjadi representasi fitur yang semantik, sedangkan *decoder* bertanggung jawab untuk menghasilkan *output* *sequence* dari representasi yang diberikan oleh *encoder*. Masing-masing *encoder* dan *decoder* terdiri dari beberapa *layer* yang terdiri dari *sub-layer*. *Sub-layer* dalam *encoder* terdiri dari *multi-head self-attention layer*, diikuti oleh *feed-forward neural network layer*. Di sisi lain, *sub-layer* dalam *decoder* memiliki *multi-head self-attention layer*, diikuti oleh *masked multi-head self-attention layer* dan *feed-forward neural network layer*. *Sub-layer* ini bekerja bersama-sama untuk memperoleh representasi fitur yang semantik dan menghasilkan *output* yang akurat (Vaswani *et al. ,* 2017).

Selain diterapkan dalam NLP, Transformer juga telah digunakan secara luas dalam bidang *computer vision*, khususnya dalam tugas-tugas seperti segmentasi gambar, pemrosesan gambar, dan pengenalan objek. Penggunaan Transformer dalam domain non-NLP menunjukkan fleksibilitas dan kemampuan adaptasi yang luar biasa dari arsitektur tersebut (Zhang *et al.*, 2021). Perluasan Transformer ke dalam domain *computer vision* ini salah satunya dipelopori oleh Carion dkk. di tahun 2020. Carion menciptakan suatu rancangan transformer yang dapat memahami konteks secara spasial dan diberi nama *DEtection TRansformer* atau DETR. DETR ini dapat menyaingi model *Faster* R-CNN secara signifikan dalam objek-objek besar berkat mekanisme *self-attention*-nya (Carion *et al.,* 2020). DETR ini kemudian dikembangkan oleh Zhao di tahun 2024 yang memungkinkan penggunaannya menjadi *realtime* dengan beberapa modifikasi seperti penghapusan mekanisme NMS, penerapan metode *Efficient Hybrid Encoder*, *Uncertainty-Minimal Query Selection*, dan menciptakan *Flexible Speed Tuning* yang dapat mengatur jumlah *decoder layers* sesuai kebutuhan tanpa perlu *retraining*. Varian *realtime* dari DETR yang dikembangkan oleh Zhao ini diberi nama *RealTime*-DETR atau RT-DETR (Zhao *et al.,* 2024)



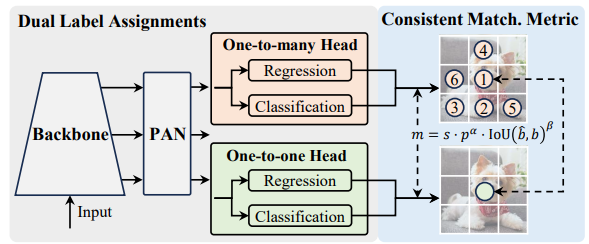
Gambar 2.7 Arsitektur RT-DETR

Sumber: Zhao, *et al.* (2024)

RT-DETR merupakan salah satu pengembangan dari arsitektur Transformer untuk deteksi objek secara *realtime*, dirancang untuk mengatasi keterbatasan model YOLO yang memerlukan *Non-Maximum Suppression* (NMS) dalam tahap *post-processing*. Arsitektur RT-DETR terdiri dari tiga komponen utama, yaitu *backbone*, *efficient hybrid encoder*, dan *decoder Transformer*. Fitur *multiscale* dari *backbone* diproses oleh *hybrid encoder* yang terdiri dari dua modul, yaitu *Attention-based Intra-scale Feature Interaction* (AIFI) untuk interaksi fitur *intrascale*, dan *CNN-based Cross-scale Feature Fusion* (CCFF) untuk *cross-scale fusion*. RT-DETR menggunakan mekanisme *uncertainty-minimal query selection* untuk memilih fitur *encoder* dengan ketepatan yang tinggi sebagai *query* awal untuk *decoder*, yang kemudian secara iteratif mengoptimalkan prediksi objek. Keunggulan utama dari RT-DETR adalah kemampuannya untuk menyelaraskan kecepatan dan akurasi, menghilangkan kebutuhan akan NMS, serta mendukung *tuning* kecepatan dengan menyesuaikan jumlah lapisan *decoder* tanpa perlu pelatihan ulang, menjadikannya solusi yang fleksibel untuk berbagai skenario deteksi *realtime*​ (Zhao *et al.,* 2024).

## 2.7 CNN-Transformer (YOLOv10)

Sesuai namanya, metode berbasis CNN-Transformer menggabungkan *Convolutional Neural Network* (CNN) dengan mekanisme Transformer. CNN dikenal unggul dalam mengekstraksi fitur spasial dari citra melalui jaringan konvolusi bertumpuk, sementara Transformer, yang awalnya digunakan dalam bidang *Natural Language Processing* (NLP), mampu menangkap hubungan jangka panjang dengan memanfaatkan mekanisme *self-attention*. Kombinasi ini bertujuan untuk mengatasi keterbatasan masing-masing metode, yaitu CNN yang terbatas dalam menangkap dependensi jarak jauh dan Transformer yang kurang optimal untuk pengolahan citra tanpa bantuan lapisan konvolusi. Metode *hybrid* ini memungkinkan sistem untuk melakukan deteksi objek secara lebih akurat, terutama dalam lingkungan yang kompleks seperti pencahayaan rendah atau perspektif yang variatif.



Gambar 2.8 Arsitektur YOLOv10

Sumber: Wang, *et al.* (2024)

YOLOv10 memperkenalkan serangkaian peningkatan signifikan dalam arsitektur deteksi objek berbasis CNN yang bertujuan untuk meningkatkan efisiensi dan akurasi. Arsitektur YOLOv10 mengimplementasikan strategi *dual label assignment*s untuk pelatihan tanpa *Non-Maximum Suppression* (NMS), yang menghilangkan kebutuhan *post-processing* dalam inferensi. Ini memungkinkan YOLOv10 mencapai deteksi *real-time* yang lebih efisien dengan menggabungkan kekuatan dari penugasan *one-to-many* dan *one-to-one*. Selain itu, desain arsitektur yang mengedepankan efisiensi-akurasi mencakup penggunaan *lightweight classification head*, *downsampling* yang terpisah antara *spatial* dan *channel*, serta *rank-guided block* untuk mengurangi redundansi komputasi tanpa mengorbankan kinerja. YOLOv10 juga memanfaatkan large-kernel convolution dan modul *partial self-attention* untuk memperluas *receptive field* dan meningkatkan kemampuan representasi global. Hasilnya, YOLOv10 berhasil melampaui model YOLO sebelumnya dan model *state-of-the-art* lainnya dalam hal latensi dan jumlah parameter, dengan tetap mempertahankan atau meningkatkan akurasi pada dataset MS COCO​ (Wang *et al.,* 2024).

## 2.8 *Hyperparameter Tuning*

*Hyperparameter* adalah parameter yang digunakan untuk mengontrol proses pelatihan model dan tidak dipelajari langsung dari data (Goodfellow *et al*., 2016). Berbeda dengan parameter model yang di-*update* selama pelatihan, seperti bobot dalam jaringan saraf, *hyperparameter* ditentukan sebelum proses pelatihan dimulai dan berperan penting dalam menentukan performa model. Beberapa contoh *hyperparameter* yang umum meliputi *Learning Rate*, *Batch Size*, dan *Epochs*.

*Hyperparameter* *Tuning* adalah proses mengoptimalkan nilai *hyperparameter* agar model dapat mencapai performa terbaik pada data tertentu (Bergstra & Bengio, 2012). Proses *tuning* ini biasanya melibatkan eksperimen dengan berbagai kombinasi nilai *hyperparameter*, menggunakan teknik seperti *grid search* atau *random search*, untuk menemukan konfigurasi yang paling sesuai dengan data dan tugas yang dihadapi. *Hyperparameter tuning* sangat penting karena *hyperparameter* yang tidak tepat dapat menyebabkan model *underfitting* ataupun *overfitting*, dimana model tidak mampu menangkap pola yang kompleks dari data atau justru terlalu menyesuaikan diri dengan data latih sehingga kinerjanya buruk pada data uji. Salah satu cara untuk mengoptimalkan performa model dengan *hyperparameter tuning* adalah mengkonfigurasi parameter *learning rate.*

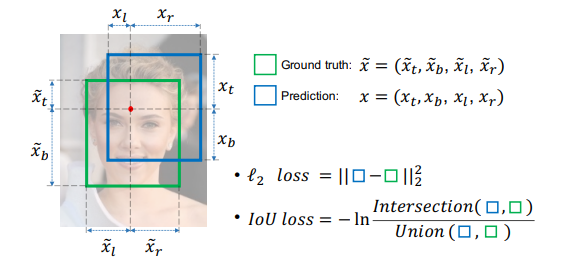
*Learning Rate* menentukan seberapa besar perubahan bobot dalam jaringan saraf terjadi setiap kali ada *update* berdasarkan *error* yang dihitung (Smith, 2017). *Learning Rate* yang terlalu tinggi dapat menyebabkan model menjadi tidak stabil karena langkah-langkah perbaikan yang terlalu besar sehingga tidak pernah mencapai titik minimum *error*. Sebaliknya, *Learning Rate* yang terlalu rendah dapat menyebabkan pelatihan model menjadi sangat lambat, memerlukan lebih banyak waktu dan sumber daya untuk mencapai konvergensi, dan bahkan bisa terjebak pada *local minimum* (Goodfellow *et al*., 2016).

*Batch Size* adalah *hyperparameter* lain yang memiliki pengaruh signifikan terhadap proses pelatihan model. *Batch Size* menentukan jumlah contoh data yang diproses sebelum bobot model diperbarui (Masters & Luschi, 2018). Pada umumnya, *Batch Size* yang lebih besar memungkinkan pelatihan yang lebih cepat dan stabil karena bisa lebih efisien dalam pemrosesan komputasi, terutama saat menggunakan GPU. Namun, *Batch Size* yang besar juga bisa mengurangi kemampuan model untuk keluar dari *local minimum*, yang dapat menyebabkan kinerja yang buruk pada generalisasi (Keskar *et al.,* 2017). Sebaliknya, *Batch Size* yang kecil seringkali menghasilkan variabilitas yang lebih tinggi dalam pembaruan bobot, yang bisa membantu model menghindari *local minimum*, meskipun dengan risiko pelatihan yang lebih lama.

## 2.9 *Loss Function*

*Loss function* merupakan komponen penting dalam pelatihan model terutama untuk *deep learning* dalam mendeteksi objek. *Loss function* digunakan untuk mengukur seberapa baik model memprediksi hasil yang diinginkan selama proses pelatihan. Secara umum, *object detection* memerlukan dua *loss function*, yaitu *loss function* untuk klasifikasi objek itu sendiri, dan yang *loss function* untuk *bounding box regression*. Salah satu *loss function* yang paling umum digunakan untuk *bounding box regression* adalah *Intersection over Union* (IoU), yang mengukur tumpang tindih antara prediksi *box* dengan *ground truth*. IoU loss ini pertama kali diperkenalkan oleh Yu (2016) untuk mengatasi permasalahan yang ada pada *loss function* sebelumnya yaitu ℓ2 loss (Yu *et al.,* 2016).

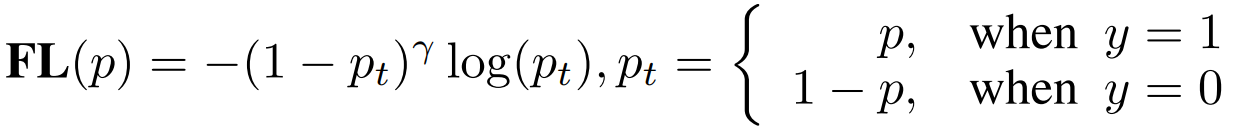
Kelemahan utama dari ℓ2 *loss* terletak pada koordinat sebuah *bounding box* yang dioptimalkan sebagai empat variabel yang independen. Asumsi ini mengabaikan fakta bahwa batas-batas suatu objek sangat berkorelasi. Hal ini dapat mengakibatkan beberapa kasus kegagalan di mana satu atau dua batas dari *bounding box* yang diprediksi sangat dekat dengan *ground truth*, tetapi keseluruhan *bounding box* tidak dapat diterima. IoU menyelesaikan masalah ini dengan menganggap satu *bounding box* adalah sebagai satu unit. Dengan begitu IoU *loss* dapat memberikan prediksi yang lebih akurat daripada ℓ2 *loss* (Yu *et al.* 2016). Secara umum, perbedaan ℓ2 *loss* dan IoU *loss* dapat dilihat pada Gambar 2.9.



Gambar 2.9 Perbedaan ℓ2 loss dan IoU loss

Sumber: Yu, *et al.* (2016)

Namun, IoU *loss* ini hanya digunakan untuk memprediksi *bounding box* pada *ground truth* dan belum dapat mengklasifikasikan kelas objek. Agar dapat mengklasifikasikan kelas objek, terdapat sebuah *loss function* bernama *Focal Loss* (FL) yang menerapkan konsep modulasi pada *cross entropy loss* untuk memfokuskan pembelajaran pada contoh-contoh yang sulit dan mengurangi bobot dari *easy negatives* (Lin *et al.,* 2017). Persamaan *Focal Loss* ini dijelaskan sebagai berikut.



(2.2)

Dimana *y* ∈ {1, 0} menentukan kelas *ground-truth* dan *p* ∈ [1, 0] menunjukkan estimasi probabilitas untuk kelas dengan label *y = 1* , γ adalah parameter fokus yang dapat dilakukan *tuning*. Secara spesifik, FL terdiri dari bagian *cross entropy − log(pt)* dan bagian *dynamically scaling factor (1 − pt) γ* , dimana *scaling factor (1 − pt) γ* secara otomatis menurunkan bobot kontribusi dari *easy examples* selama pelatihan dan dengan cepat memfokuskan model pada *hard examples*.

*Focal Loss* ini kembali dikembangkan menjadi *Generalized Focal Loss* (GFL) yang menggeneralisasi *Focal Loss* dari diskrit {1,0} ke versi kontinu. GFL dapat dikhususkan menjadi *Quality Focal loss* (QFL) dan *Distribution Focal Loss* (DFL), di mana QFL mendorong untuk mempelajari representasi gabungan yang lebih baik dari kualitas klasifikasi dan lokalisasi, dan DFL memberikan estimasi *bounding box* yang lebih informatif dan tepat dengan memodelkan lokasinya sebagai *General Distributions* (Li *et al.,* 2020).

IoU juga dikembangkan menjadi *Generalized IoU* (GIoU) *loss*. GIoU adalah perbaikan dari IoU *loss* yang digunakan untuk menghitung seberapa baik *bounding box* prediksi sesuai dengan *ground truth*. IoU biasa hanya mengukur tumpang tindih antara dua kotak, GIoU memperbaiki kelemahan IoU ketika tidak ada tumpang tindih antara prediksi dan *ground truth*. GIoU memperkenalkan area pembungkus minimal yang mencakup kedua kotak tersebut dan menghitung rasio antara area pembungkus dan selisih area antara kotak prediksi dengan *ground truth*. Dengan pendekatan ini, GIoU memberikan nilai yang lebih representatif bahkan ketika *bounding box* prediksi dan *ground truth* tidak bertumpang tindih, yang sering terjadi pada awal pelatihan model deteksi objek (Rezatofighi *et al*., 2019). Persamaan GIoU ini dijelaskan sebagai berikut.

(2.3)

Dimana *A* dan *B* merupakan dua bentuk *convex* sembarang dan *C* adalah bentuk *convex* terkecil yang dapat menyelimuti kedua bentuk A dan B secara keseluruhan.

Seiring dengan perkembangan *loss function* untuk prediksi *bounding box*, terdapat pula kemajuan dalam *loss function* untuk *object classification* yang dikenal sebagai *Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss)*. *BCEWithLogitsLoss* menggabungkan dua operasi penting yaitu *sigmoid activation* dan *binary cross-entropy loss* dalam sat langkah yang lebih efisien. Pada dasarnya, *loss* ini digunakan untuk mengukur kesalahan antara prediksi model (*logits*) dan target yang diinginkan (label biner). Pada fungsi BCE biasa, tahap yang dilakukan pertama kali adalah aktivasi sigmoid pada *output* model untuk mengubah nilai *logit* menjadi probabilitas antara 0 dan 1. Kemudian, probabilitas ini dibandingkan dengan target menggunakan binary *cross-entropy loss*, yang menghitung seberapa jauh perbedaan antara distribusi prediksi dan target. Namun, dalam *BCEWithLogitsLoss*, proses aktivasi sigmoid dilakukan secara internal, sehingga mengurangi risiko masalah numerik seperti *gradient vanishing* yang sering menggunakan sigmoid secara terpisah (Ruby & Yendapalli, 2020). Persamaan *BCEWithLogitsLoss* ini dijelaskan sebagai berikut.

(4)

Dimana *y* adalah label sebenarnya (0 atau 1) , *x* adalah *logit* atau *output* dari model, dan adalah fungsi sigmoid dari *logit* tersebut.

## 2.10 Metrik Evaluasi Model

Metrik evaluasi model digunakan untuk mengukur kinerja model. Dalam penelitian yang berkaitan dengan deteksi objek pada kondisi pencahayaan rendah, pemilihan metrik evaluasi model menjadi elemen krusial dalam menentukan performa model secara akurat. *F1-Score*, *Precision*, *Recall*, *Accuracy*, dan *Mean Average Precision* (mAP) adalah beberapa metrik yang sering digunakan dalam berbagai penelitian untuk menilai efektivitas model deteksi objek. *F1-Score* menggabungkan *Precision* dan *Recall* dalam satu ukuran, memberikan bobot yang seimbang antara *false positives* dan *false negatives*, yang sangat penting dalam kasus data yang tidak seimbang (Chicco & Jurman, 2020). *Precision* menggambarkan proporsi dari prediksi positif yang benar-benar akurat, sementara *Recall* mengukur sejauh mana model dapat mendeteksi semua contoh positif yang ada, keduanya adalah metrik yang esensial dalam konteks pengenalan objek yang sering kali mengalami masalah *misalignment* atau *noise* pada citra (Powers, 2020). Persamaan untuk metrik-metrik tersebut dijelaskan sebagai berikut,

, (2.5)

, (2.6)

, (2.7)

*accuracy =*  (2.8)

Dimana *TP* adalah *True Positive*, yaitu jumlah data di kelas positif yang berhasil diprediksi dengan benar sebagai positif, dan *TN* adalah *True Negative*, yaitu jumlah data di kelas negatif yang juga diprediksi dengan benar sebagai negatif. Sebaliknya, *FN* (*False Negative*) mengacu pada kasus di mana data yang seharusnya berada di kelas positif justru diprediksi sebagai negatif, dan *FP* (*False* *Positive*) terjadi ketika data dari kelas negatif salah diklasifikasikan sebagai positif.

Metrik-metrik dasar ini sangat penting dalam mengevaluasi performa model karena memberikan informasi rinci tentang jenis kesalahan yang dibuat oleh model. *Precision*, misalnya, sangat sensitif terhadap *False* *Positives* dan berguna dalam skenario di mana kesalahan deteksi positif (misalnya, mendeteksi objek yang sebenarnya tidak ada) harus diminimalkan. *Recall*, di sisi lain, lebih relevan dalam situasi di mana tidak terlewatkan satu pun deteksi benar sangat penting, seperti pada sistem keamanan atau aplikasi medis, di mana *False* *Negatives* (objek yang ada tetapi tidak terdeteksi) lebih berbahaya.

*Accuracy* sering kali digunakan sebagai metrik dasar dalam evaluasi model, tetapi metrik ini bisa menyesatkan terutama pada dataset yang tidak seimbang, di mana jumlah contoh dari satu kelas jauh lebih besar dibandingkan kelas lainnya (Saito *et al*., 2015). Oleh karena itu, untuk deteksi objek dalam citra, *mean Average Precision* (mAP) dianggap sebagai metrik yang lebih komprehensif karena memperhitungkan akurasi deteksi untuk setiap kelas objek, dan memberikan gambaran keseluruhan performa model dalam mendeteksi berbagai jenis objek (Zhao *et al*., 2019). Penggunaan mAP dalam tugas deteksi objek telah menjadi standar industri, terutama dalam kompetisi seperti COCO dan PASCAL VOC, yang mengharuskan model tidak hanya mendeteksi objek dengan benar, tetapi juga menentukan lokasi dengan presisi tinggi (Everingham *et al*., 2015). Persamaan mAP ini dijelaskan sebagai berikut,

*mAP=*  (2.9)

Dimana *n* adalah jumlah kelas yang ada dan adalah rerata nilai *precision* dari kelas *k*.

## 2.11 Pareto Frontier

Pareto Frontier, atau juga dikenal dengan Pareto Optimum, merupakan suatu *framework* yang digunakan dalam *multi-objective optimization* dan berasal dari teori ekonomi. Dalam domain *machine learning*, Pareto Frontier juga digunakan untuk mengidentifikasi solusi optimal ketika terdapat lebih dari satu tujuan yang saling bertentangan, seperti akurasi model dan waktu inferensi.

Solusi pada Pareto Frontier adalah solusi yang dinamakan Pareto Optimal, yang berarti tidak ada solusi lain yang dapat meningkatkan satu tujuan tanpa merugikan tujuan lainnya. Dengan demikian, Pareto Frontier memungkinkan identifikasi model yang mencapai keseimbangan optimal antara berbagai metrik (Benmeziane *et al*., 2022).

Dalam *multi-objective optimization*, tujuan yang sering dicapai adalah untuk meminimalkan atau memaksimalkan beberapa fungsi objektif secara bersamaan. Dalam optimasi dua *objective*, dan , solusi dianggap Pareto Optimal jika tidak ada solusi yang memenuhi:

*dan*  *dengan*  *atau*  (2.10)

Dengan kata lain, solusi adalah optimal jika setiap perbaikan pada salah satu tujuan akan menyebabkan kerugian pada tujuan lainnya. Sekumpulan solusi Pareto optimal ini membentuk Pareto Frontier, yang menggambarkan *trade-off* antara tujuan-tujuan yang saling bertentangan tersebut (Deb, 2001). Konsep Pareto Frontier telah terbukti sangat berguna dalam berbagai aplikasi termasuk *machine learning*. Salah satu contoh yang menonjol adalah dalam *Neural Architecture Search* (NAS), dimana Pareto Frontier digunakan untuk mengidentifikasi arsitektur yang optimal dalam hal latensi dan akurasi inferensi (Ye *et al.*, 2024). Selain itu, penelitian juga menunjukkan potensi Pareto Frontier dalam mengoptimalkan beberapa tujuan secara simultan, seperti *fairness*, *privacy*, dan *utility* dalam *modeling* (Yaghini et al., 2023).

## 2.12 *Optimizer*

*Optimizer* adalah sebuah konsep dalam *deeplearning*  yang berfungsi untuk meminimalkan *loss function* dengan cara memperbarui bobot atau *weights* jaringan secara iteratif. Menurut Cacciola dkk. (2023) proses pembaruan parameter melalui *optimizer* sangat menentukan laju konvergensi dan kestabilan pelatihan, terutama ketika menangani dataset berukuran besar. Li dan Liang (2018) menunjukkan bahwa *optimizer*, terutama metode *Stochastic Gradient Descent* (SGD), memiliki efek regularisasi implisit yang mendorong model menuju solusi yang lebih general dan mencegah *overfitting*. Wang dan Wu (2024) menjelaskan *Optimizer* SGD ini dengan persamaan berikut:

(2.11)

Persamaan SGD di atas memperlihatkan mekanisme dasar dari optimasi model melalui pembaruan parameter secara stokastik. Variabel merepresentasikan sekumpulan bobot atau parameter jaringan pada iterasi ke-*t*, yang bertugas mengatur perilaku model. *Learning rate* η menentukan seberapa besar perubahan atau *step* yang diambil dalam setiap iterasi, nilai yang terlalu besar dapat menyebabkan konvergensi tidak stabil, sedangkan nilai yang terlalu kecil dapat membuat proses pelatihan berjalan terlalu lambat. Gradien adalah vektor gradien yang menggambarkan arah peningkatan tercepat dari *loss function*  ketika parameter berada pada kondisi . Oleh karena itu, dengan mengurangkan gradien ini, SGD secara bertahap menyesuaikan parameter agar fungsi kerugian berkurang. Perhitungan gradien dilakukan berdasarkan satu atau sekumpulan kecil data acak (*minibatch*), sehingga metode ini efisien dalam menghadapi dataset berukuran besar dan memiliki efek regularisasi implisit.

# 

# **BAB III METODE PENELITIAN**

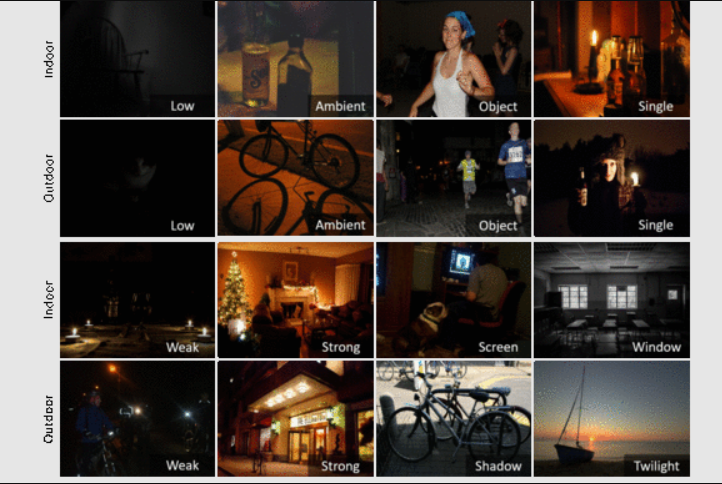
## 3.1 Lokasi dan Waktu

Penelitian dilakukan di Laboratorium MPL, Gedung Kuliah Bersama, Kampus C Universitas Airlangga yang beralamat di Jalan Dr. Ir. H. Soekarno, Mulyorejo, Surabaya, Jawa Timur. Penelitian ini berlangsung dari bulan Juli 2024 hingga bulan Desember 2024, sesuai dengan jadwal yang telah ditentukan. Dalam periode tersebut, peneliti melakukan perencanaan, pengumpulan data, dan penyusunan skripsi.

Penelitian ini dilaksanakan dengan menggunakan *hardware* yang terdiri dari 12.7 GB RAM, GPU NVIDIA Tesla T4, dan CPU Intel(R) Xeon(R). Penggunaan platform perangkat lunak melibatkan torch-2.1.0+cu121 dan Python-3.10.12.

## 3.2 Bahan dan Alat

Data yang digunakan pada penelitian ini merupakan data sekunder *The Exclusively Dark* atau *ExDark* yang dibuat oleh Loh, Y. P., & Chan, C. S. (2019). Data ini merupakan kumpulan dari 7363 citra dengan pencahayaan sangat rendah hingga pencahayaan fajar dengan 12 kelas objek. Objek-objek tersebut adalah *table, people, motorbike, dog, cup, chair, bicycle, boat, bottle, bus, car,* dan *cat.* Sebagian besar data citra ini diperoleh dari *website* dan *search engine* seperti *Flickr.com, Photo-bucket.com, Imgur.com, Deviantart.com, Gettyimages.com* dan *Google Search* dengan kata kunci *dark, low-light, nighttime,* dan lain-lain. ExDark juga mengambil sampel dari dataset-dataset besar seperti PASCAL VOC, ImageNet, dan Microsoft COCO serta menambah variasi data dengan mengekstraksi setiap *frame* berpencahayaan rendah dalam film-film dan mengambil gambar secara manual menggunakan beragam model *smartphone* dan kamera.



Gambar 3.1 Sampel Exclusively Dark dataset

Sumber: Loh, Y. P., & Chan, C. S. (2019)

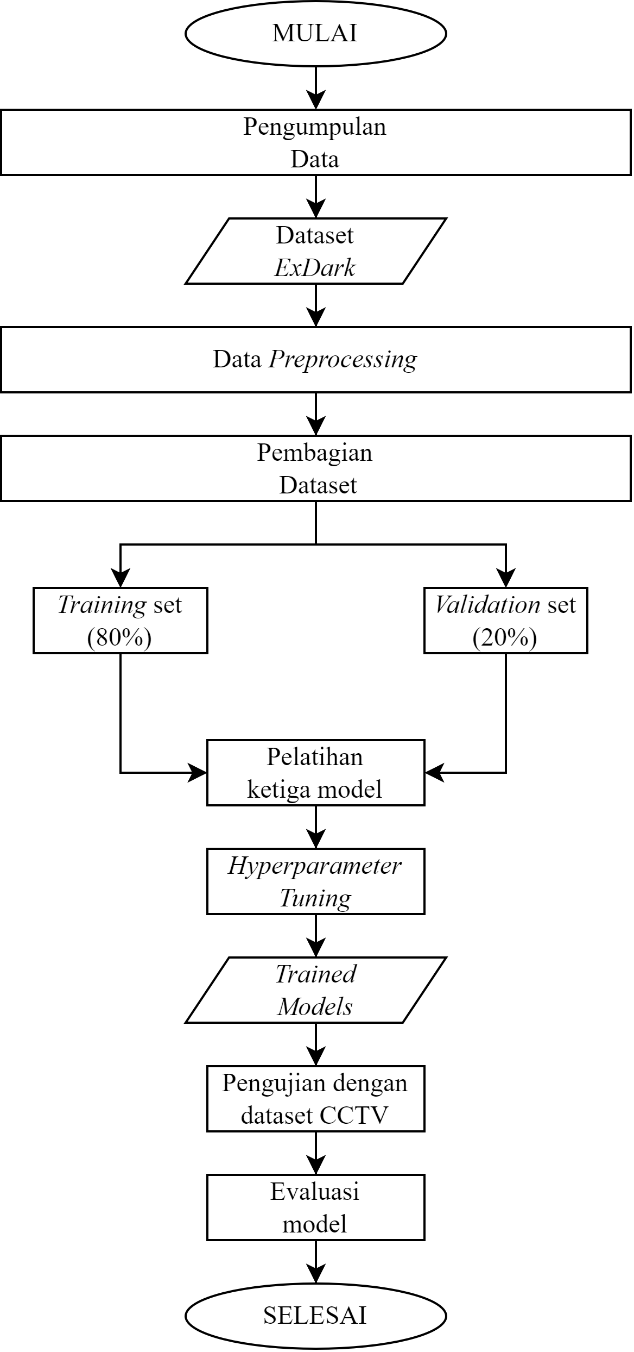
## 3.3 Cara Kerja

Penelitian ini tidak secara langsung menguji model pada perangkat CCTV atau sistem dengan sumber daya terbatas. Namun, salah satu kekuatan utama dari model deteksi objek modern adalah kemampuannya untuk melakukan generalisasi dari berbagai jenis dataset, termasuk dari dataset ExDark yang mengandung gambar dalam kondisi pencahayaan rendah yang sangat mirip dengan gambar yang dihasilkan oleh kamera CCTV di malam hari atau area yang gelap dengan berbagai macam sudut kamera.

Metode penelitian ini terdiri dari beberapa tahapan. Tahapan pertama dalam penelitian ini adalah dengan membagi dataset ke dalam dua set, yaitu *train* (80%), *validation* (20%). Pada tahap ini, pemilihan *train* set dan *validation* set akan menggunakan teknik *Stratified Random Sampling*. Teknik ini dipilih karena setiap gambar dalam dataset terbagi ke dalam 12 kelas yang berbeda, dan tujuan dari stratifikasi adalah memastikan bahwa distribusi kelas dalam set *validation* tetap seimbang, sehingga mewakili proporsi yang sama dengan dataset asli. Teknik ini penting untuk menghindari bias dalam proses validasi model dan memastikan bahwa semua kelas terwakili secara proporsional selama evaluasi.

Setelah pembagian data, dilakukan beberapa tahapan *preprocessing* seperti pengurangan resolusi menggunakan *lanczos resampling* lalu pengubahan format label *bounding box* yang disesuaikan untuk kebutuhan model YOLO dan RT-DETR. Data kemudian ditingkatkan kualitasnya dengan teknik *contrast enhancement* menggunakan CLAHE. Tahap terakhir dalam *preprocessing* ini adalah dengan augmentasi data menggunakan parameter-parameter yang ditentukan. Tahap augmentasi data ini juga mencakup algoritma *mosaic data*.

Setelah tahap *preprocessing* dan pembagian data, *train set* dan *validation set* digunakan untuk melatih tiga model (YOLOv9, RT-DETR, dan YOLOv10) dengan proses *hyperparameter tuning*. Setelah pelatihan selesai, model terlatih (*trained* *models*) akan dievaluasi berdasarkan performanya, termasuk *mean Average Precision* (mAP) dan waktu inferensi, kemudian dibandingkan menggunakan analisis *Pareto Frontier*. Model terlatih ini kemudian diuji menggunakan *test set* yang berupa dataset rekaman CCTV asli. Rekaman CCTV diperoleh dari sumber daring dengan kata kunci "*CCTV footage of a [subject*]", dimana *subject* mengacu pada jenis objek yang terdapat dalam dataset ExDark. Durasi rekaman bervariasi antara 5 hingga 10 detik, dengan 1 *frame* per detik.



Gambar 3.2 Flowchart Keseluruhan Proses

### 3.3.1 Data Preprocessing

Pada tahap ini dilakukan *data preprocessing* pada *train set* yang meliputi:

3.3.1.1 *Downsampling*

Tahap pertama dalam proses preprocessing adalah penerapan downsampling menggunakan algoritma *Lanczos Resampling*. Proses ini dilakukan di awal untuk mengurangi kompleksitas komputasi pada tahapan *preprocessing* berikutnya. Penting untuk dicatat bahwa model-model yang akan dilatih tidak menggunakan citra dengan resolusi hasil *downsampling*, karena masing-masing model telah mengkonfigurasi parameter input secara *default* sebesar 640×640 piksel. Dengan demikian, berapapun resolusi citra awal, semua akan diseragamkan menjadi 640×640 piksel. Langkah *downsampling* ini berfungsi untuk mengurangi detail-detail yang dianggap kurang signifikan, sehingga memungkinkan model mencapai generalisasi yang lebih baik ketika dilakukan resampling atau interpolasi otomatis melalui arsitektur model yang telah disesuaikan.

3.3.1.2 Pengubahan format label *bounding box*

Setiap citra yang sudah melalu tahap downsampling akan membutuhkan anotasi *bounding box* baru.Setiap anotasi di dataset ExDark mencakup beberapa informasi, antara lain 16 karakter pertama yang berisi data dari alat anotasi, kolom pertama yang berisi nama kelas objek, serta kolom kedua hingga kelima yang menyatakan koordinat *bounding box* dalam bentuk [l t w h], di mana l adalah jarak dari tepi kiri gambar, t adalah jarak dari tepi atas gambar, w adalah lebar *bounding box*, dan h adalah tinggi *bounding box*. Selain itu, anotasi ini juga memiliki informasi tambahan mengenai oklusi dan orientasi yang tidak akan digunakan.

Untuk mengubah format tersebut menjadi format yang dapat diterima oleh seluruh model, dilakukan proses konversi yang melibatkan beberapa langkah penting. Pertama, nama kelas objek dikonversi menjadi ID kelas yang sesuai. Selanjutnya, koordinat *bounding box* yang awalnya dinyatakan dalam bentuk [l t w h] dikonversi menjadi [center\_x, center\_y, width, height] di mana center\_x dan center\_y adalah koordinat pusat *bounding box*, sedangkan *width* dan *height* tetap merupakan dimensi dari *bounding box*. Semua nilai ini kemudian dinormalisasi dengan membagi nilaix*,* y*,* width*,* dan heightdengan lebar dan tinggi gambar. Hasil akhir dari proses ini dituliskan ke dalam *file* dengan format .txt yang sesuai untuk digunakan dalam pelatihan model.

3.3.1.3 *Contrast Enhancement*

Setelah citra pada *train set* dilakukan *downsampling*, tahap selanjutnya dalam metodologi ini adalah penerapan teknik normalisasi data menggunakan *Contrast Limited Adaptive Histogram Equalization* (CLAHE). Teknik ini dipilih karena kemampuannya dalam meningkatkan kontras gambar, khususnya pada kondisi pencahayaan rendah, seperti yang terdapat pada dataset ExDark. CLAHE bekerja dengan membagi gambar menjadi blok-blok kecil, kemudian melakukan *equalization* pada histogram setiap blok secara adaptif. Langkah ini memastikan bahwa kontras lokal pada setiap bagian gambar dapat ditingkatkan secara proporsional, tanpa menyebabkan amplifikasi *noise* yang berlebihan.

Dalam penelitian ini, CLAHE diterapkan untuk meningkatkan visibilitas objek pada gambar, dengan tujuan memaksimalkan akurasi deteksi objek pada kondisi pencahayaan yang beragam. Proses ini penting karena peningkatan kontras yang dihasilkan dari CLAHE membantu model deteksi objek dalam mengidentifikasi objek secara lebih efektif, terutama dalam skenario dengan pencahayaan yang tidak merata.

3.3.1.3 Normalisasi Data dengan *min-max scaler*

Pada tahap ini, seluruh data gambar dalam dataset dinormalisasi menggunakan teknik *min-max scaler*. Proses ini dilakukan dengan mengubah nilai piksel gambar ke dalam rentang 0 hingga 1, sehingga seluruh data memiliki skala yang seragam. Teknik *min-max scaler* dipilih karena sesuai dengan karakteristik dataset citra yang memiliki nilai piksel dalam rentang yang terbatas, dan teknik ini menjaga distribusi relatif antar nilai piksel. Hal ini penting untuk memastikan model deteksi objek dapat memproses gambar secara optimal, tanpa distorsi skala yang berlebihan. *min-max scaler* cocok untuk digunakan pada model seperti CNN dan Transformer yang sensitif terhadap perbedaan skala antar *input*, terutama pada tugas deteksi objek dalam citra pencahayaan rendah, di mana perbedaan kontras dapat mempengaruhi performa model.

3.3.1.4 Pembagian data

Tahap selanjutnya dalam *preprocessing* ini adalah membagi dataset ke dalam dua set, yaitu *train* (80%), *validation* (20%), dan akan diuji pada *test* set berupa rekaman asli CCTV dari dunia nyata. Pada tahap ini, pemilihan *train* set dan *validation* set akan menggunakan teknik *Stratified Random Sampling*. Teknik ini dipilih karena setiap gambar dalam dataset terbagi ke dalam 12 kelas yang berbeda, dan tujuan dari stratifikasi adalah memastikan bahwa distribusi kelas dalam set *validation* tetap seimbang, sehingga mewakili proporsi yang sama dengan dataset asli. Teknik ini penting untuk menghindari bias dalam proses validasi model dan memastikan bahwa semua kelas terwakili secara proporsional selama evaluasi.

3.3.1.5 *Data* *Augmentation*

Setelah membagi dataset ke dalam *train set* dan juga *validation set*, tahap terakhir adalah melakukan augmentasi data pada *train set* yang sudah dibuat. Proses augmentasi data ini dilakukan dengan menggunakan bantuan dari modul *albumentations*. Parameter-parameter augmentasi untuk digunakan dalam modul *albumentations* dijelaskan dalam Tabel 3.1.

Tabel 3.1 Deskripsi Parameter Augmentasi

| ***Parameter*** | ***Value Range*** |
| --- | --- |
| *Rotation augmentation range* | 0,0 ; 45,0 |
| *Translation augmentation range* | 0,0 ; 0,9 |
| *Scaling augmentation range* | 0,0 ; 0,9 |
| *Shear augmentation range* | 0,0 ; 10,0 |
| *Perspective augmentation range* | 0,0 ; 0,001 |
| *Horizontal flip augmentation probability* | 0,0 ;1,0 |

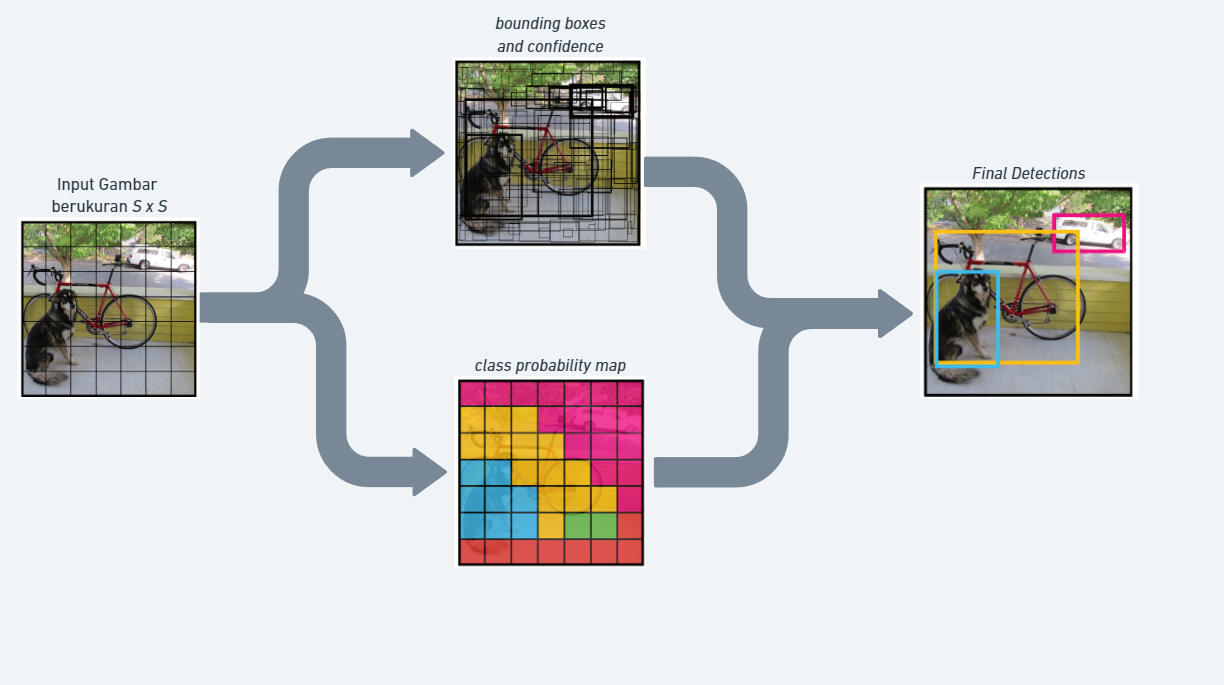
Setelah seluruh citra mengalami proses augmentasi sesuai dengan parameter yang telah ditetapkan, langkah berikutnya adalah penerapan konsep mosaic data untuk meningkatkan keberagaman data yang dapat dipelajari model. Proses mosaic ini dilakukan pada citra asli sebelum augmentasi dan mengelompokkan citra ke dalam satu set yang terdiri dari enam citra, tanpa dilakukan pengambilan data secara berulang.

### 3.3.2 Implementasi Model

Pada tahap ini model berbasis CNN, Transformer, dan CNN-Transformer akan dilatih menggunakan *training set* serta akan divalidasi dengan *validation set*. Setelah itu akan dilakukan proses *hyperparameter tuning* untuk mendapatkan kombinasi parameter optimal.

3.3.2.1 Model berbasis CNN

Proses inferensi model berbasis CNN yaitu YOLOv9 yang digunakan pada penelitian ini ditunjukkan pada Gambar 3.3. Proses dimulai dengan sebuah gambar *input* berukuran S×S. dimana setiap sel grid memiliki tugas untuk mendeteksi objek yang pusatnya jatuh dalam sel tersebut.



Gambar 3.3 Proses Inferensi Model berbasis CNN

YOLO bekerja dengan dua cabang utama. Cabang pertama bertugas memprediksi *bounding boxes* dan *confidence scores*. Setiap sel pada grid memprediksi beberapa *bounding boxes*, yang masing-masing berisi informasi tentang koordinat (x, y), lebar (*width*), tinggi (*height*), dan nilai kepercayaan yang mencerminkan seberapa yakin model bahwa kotak tersebut mengandung objek dan seberapa tepat prediksi tersebut dibandingkan dengan objek sebenarnya.

Cabang kedua dari arsitektur ini menghasilkan *class probability map*. Setiap sel *grid* juga memprediksi probabilitas kelas untuk berbagai jenis objek yang mungkin berada dalam sel tersebut. Probabilitas ini menunjukkan seberapa besar kemungkinan objek dalam sel tersebut termasuk dalam setiap kelas yang ada.

Hasil dari kedua cabang ini digabungkan untuk menghasilkan deteksi *final*. *Bounding boxes* yang memiliki nilai kepercayaan tinggi dan probabilitas kelas yang signifikan akan dipilih sebagai hasil deteksi akhir, ditampilkan sebagai kotak yang mengelilingi objek yang terdeteksi dalam gambar.

YOLOv9 memiliki lima tipe model yang dibedakan berdasarkan jumlah parameternya. Namun, karena keterbatasan sumber daya, penelitian ini hanya akan menggunakan tiga tipe model yang berada di tengah-tengah spektrum jumlah parameter. Pemilihan ketiga tipe ini dilakukan dengan tujuan untuk memprediksi tren kinerja, yang diharapkan dapat mewakili performa dua tipe model lainnya yang tidak digunakan dalam penelitian ini. Tipe-tipe model YOLOv9 yang akan diuji pada penelitian ini dijelaskan dalam Tabel 3.2.

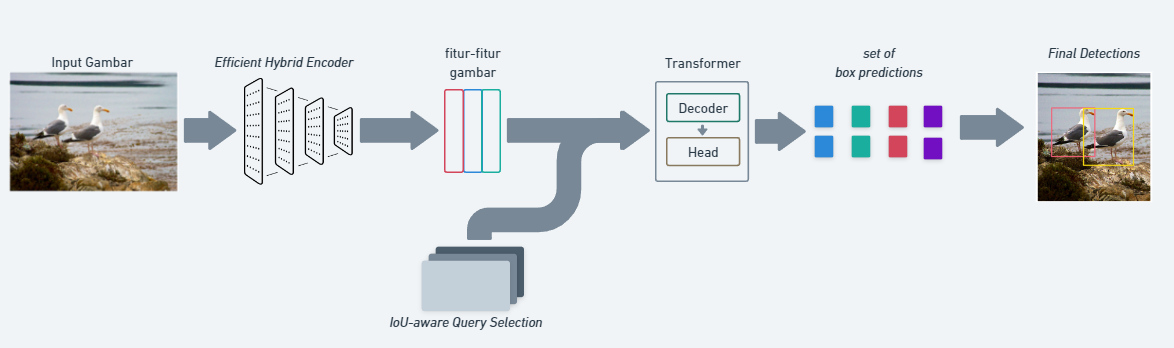
Tabel 3.2 Tipe Model YOLOv9

|  |  |
| --- | --- |
| **Tipe** | **Jumlah Parameter**  **(dalam juta)** |
| *YOLOv9s* | 7,2 |
| *YOLOv9m* | 20,1 |
| *YOLOv9c* | 25,5 |

Proses inferensi model berbasis CNN, yaitu YOLOv9, yang digunakan dalam penelitian ini akan bersumber dari library *Ultralytics*. Library ini menyediakan implementasi YOLO yang telah dioptimalkan untuk berbagai aplikasi deteksi objek secara *real-time*. Dalam implementasinya, *Ultralytics* menggunakan beberapa *loss function* yang penting untuk meningkatkan kinerja deteksi objek pada YOLOv9, yaitu *box-loss, cls-loss,* dan *dfl-loss*. Berdasarkan dokumentasi *Ultralytics*,algoritma yang digunakan oleh *box-loss* adalah *Intersection over Union* *loss*, untuk *cls-loss* adalah *BCEWithLogitsLoss*, dan *dfl-loss* adalah *Distribution Focal Loss*. *Optimizer* SGD akan dipilih sebagai *optimizer* pada ketiga tipe model YOLOv9.

3.3.2.2 Model berbasis Transformer

Proses inferensi model berbasis Transformer yang menggunakan model *Real-Time DEtection TRansformer* (RT-DETR) yang digunakan pada penelitian ini ditunjukkan pada Gambar 3.4. Proses dimulai dengan memasukkan gambar *input* ke dalam *convolutional neural network* (CNN). CNN berfungsi untuk mengekstraksi fitur-fitur penting dari gambar, menghasilkan representasi fitur yang lebih terperinci dan kompak.



Gambar 3.4 Proses Inferensi Model Berbasis Transformer (RE-DETR)

Fitur-fitur gambar yang dihasilkan dari CNN kemudian diberi pengkodean posisi (*positional encoding*) untuk mempertahankan informasi spasial dari gambar asli. *Positional encoding* ini penting karena Transformer memerlukan informasi posisi untuk memproses urutan data secara efektif.

Selanjutnya, fitur-fitur gambar yang telah diberi pengkodean posisi ini dimasukkan ke dalam Transformer. Transformer terdiri dari dua komponen utama, *Encoder* dan *Decoder*. *Encoder* bertugas untuk memproses dan memahami representasi fitur gambar secara menyeluruh, sementara *Decoder* menghasilkan prediksi *bounding boxes* dan kelas objek dari fitur-fitur yang telah diproses.

Set *box prediction* yang dihasilkan oleh *Decoder Transformer* mencakup informasi koordinat dan kategori objek yang terdeteksi. Pada tahap akhir, prediksi-prediksi ini digabungkan untuk menghasilkan deteksi akhir, di mana objek-objek yang terdeteksi ditandai dengan *bounding boxes*  pada gambar asli.

Tipe-tipe model RT-DETR yang akan diuji pada penelitian ini dijelaskan dalam Tabel 3.3.

Tabel 3.3 Tipe Model RT-DETR

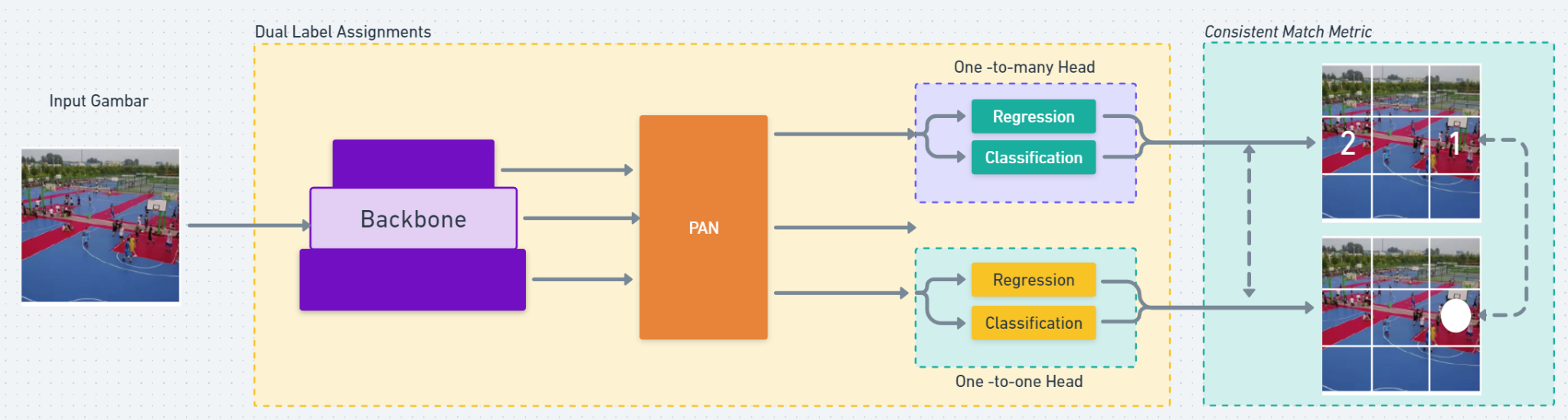
|  |  |
| --- | --- |
| **Tipe** | **Jumlah Parameter**  **(dalam juta)** |
| *RT-DETR-L* | 32,9 |
| *RT-DETR-X* | 67,4 |

Proses inferensi model berbasis Transformer, yaitu RT-DETR, yang digunakan dalam penelitian ini akan bersumber dari library *Ultralytics*. Dalam implementasinya, *Ultralytics* menggunakan beberapa *loss function* yang penting untuk meningkatkan kinerja deteksi objek pada RT-DETR, yaitu *GIoU-loss* dan *cls-loss*. Berdasarkan dokumentasi *Ultralytics*,algoritma *loss function* yang digunakan oleh adalah *Generalized Intersection over Union* *loss* dan untuk *cls-loss* adalah *BCEWithLogitsLoss*. *Optimizer* SGD akan dipilih sebagai *optimizer* pada kedua tipe model RT-DETR.

3.3.2.3 Model berbasis CNN-Transformer

Proses inferensi model berbasis CNN-Transformer yaitu model YOLOv10 yang digunakan pada penelitian ini ditunjukkan pada Gambar 3.5. YOLOv10 mengimplementasikan *Dual Label Assignments* untuk meningkatkan kinerja deteksi objek, yaitu melalui pendekatan *one-to-one* dan *one-to-many*. Pada pendekatan *one-to-one*, setiap prediksi dicocokkan dengan satu *ground truth*. Strategi ini menghindari penggunaan proses *post-processing* seperti *Non-Maximum Suppression* (NMS) yang biasa digunakan untuk menangani prediksi *overlapping predictions*. Namun, kelemahan dari *one-to-one* adalah kurangnya pengawasan yang optimal, yang dapat mempengaruhi akurasi dan kecepatan konvergensi selama pelatihan.

Sebaliknya, pendekatan *one-to-many* memungkinkan satu prediksi dicocokkan dengan beberapa *ground truth*. Meskipun pendekatan ini memberikan *supervised signal* yang lebih baik, metode ini memerlukan NMS untuk mengurangi *overlapping predictions*. YOLOv10 menggabungkan kedua metode ini melalui *dual label assignments*. Dalam tahap pelatihan, dua "*head*" model yang mewakili kedua pendekatan ini dioptimalkan secara bersamaan. Namun, pada tahap inferensi, hanya *one-to-one* *head* yang digunakan untuk membuat prediksi, sehingga memungkinkan penerapan YOLOv10 secara *end-to-end* tanpa *cost* tambahan pada waktu inferensi.



Gambar 3.5 Proses Inferensi Model berbasis CNN-Transformer (YOLOv10)

Selama proses *label assignment*, baik pendekatan *one-to-one* maupun *one-to-many* menggunakan *Consistent Matching Metri*c untuk mengevaluasi kesesuaian antara prediksi dengan *instance* objek. Metrik yang digunakan oleh YOLOv10 adalah fungsi yang mempertimbangkan skor klasifikasi , IoU, dan prior spasial, yaitu apakah titik prediksi berada di dalam *instance* objek.

YOLOv10 memiliki enam tipe model yang dibedakan berdasarkan jumlah parameternya. Namun, karena keterbatasan sumber daya, penelitian ini hanya akan menggunakan tiga tipe model yang berada di tengah-tengah spektrum jumlah parameter. Pemilihan ketiga tipe ini dilakukan dengan tujuan untuk memprediksi tren kinerja, yang diharapkan dapat mewakili performa tiga tipe model lainnya yang tidak digunakan dalam penelitian ini. Tipe-tipe model YOLOv10 yang akan diuji pada penelitian ini dijelaskan dalam Tabel 3.4.

Tabel 3.4 Tipe Model YOLOv10

|  |  |
| --- | --- |
| **Tipe** | **Jumlah Parameter**  **(dalam juta)** |
| *YOLOv10-S* | 7,2 |
| *YOLOv10-M* | 15,4 |
| *YOLOv10-B* | 20,5 |

Proses inferensi model berbasis CNN-Transformer, yaitu YOLOv10, yang digunakan dalam penelitian ini akan bersumber dari library *Ultralytics*. Sama halnya seperti YOLOv9, *Ultralytics* menggunakan beberapa *loss function* pada YOLOv10, yaitu *box-loss, cls-loss,* dan *dfl-loss*. Algoritma yang digunakan oleh *box-loss* adalah *Intersection over Union* *loss*, untuk *cls-loss* adalah *BCEWithLogitsLoss*, dan *dfl-loss* adalah *Distribution Focal Loss*. *Optimizer* SGD akan dipilih sebagai *optimizer* pada ketiga tipe model YOLOv10.

3.3.2.4 *Hyperparameter Tuning*

Pada ketiga model dilakukan *hyperparameter tuning* dengan parameter yang ditunjukkan pada Tabel 3.5.

Tabel 3.5 Deskripsi Hyperparameter yang Diuji

|  |  |
| --- | --- |
| **Parameter** | ***Value*** |
| *Learning Rate* | 0,001 ; 0,01 |
| *Batch Size* | 8 ; 16 |

Dalam penelitian ini,konfigurasi *Learning rate & Batch Size* menjadi esensial karena model deteksi objek harus dapat menangani kompleksitas citra dengan pencahayaan rendah. Sementara itu, Epochs tidak dijadikan sebagai parameter yang di-*tuning* karena penggunaan *callbacks* seperti *early stopping* dan *patience* sudah cukup efektif dalam menghindari *overfitting*, dengan cara menghentikan pelatihan ketika tidak ada perbaikan pada performa model dalam beberapa iterasi (Prechelt, 1998). Epochs yang akan digunakan adalah 1000 dengan *patience 30.* Pendekatan ini memastikan bahwa model tidak dilatih terlalu lama sehingga tetap efisien dan optimal dalam performa.

### 3.3.3 Metode Perbandingan Model

Dalam penelitian ini, perbandingan kinerja antara ketiga model, yaitu YOLOv9, RT-DETR, dan YOLOv10, akan dilakukan dengan mempertimbangkan akurasi dan *time complexity* menggunakan Pareto Frontier. Prosedur perbandingan model ini adalah sebagai berikut:

1. Pengumpulan Data Waktu Inferensi dan Nilai *mean Average Precision* (mAP)

Setiap model akan diuji untuk mengumpulkan data mAP dan waktu inferensi pada dataset yang sama. Selain itu, *cls\_loss* (*BCEWithLogitsLoss)* akan digunakan sebagai alat ukur *(loss function)* untuk ketiga model dikarenakan *loss function* ini dapat digunakan untuk ketiga model yang bersangkutan, yakni YOLOv9, RT-DETR, dan YOLOv10.

1. Konstruksi *Pareto Frontier*

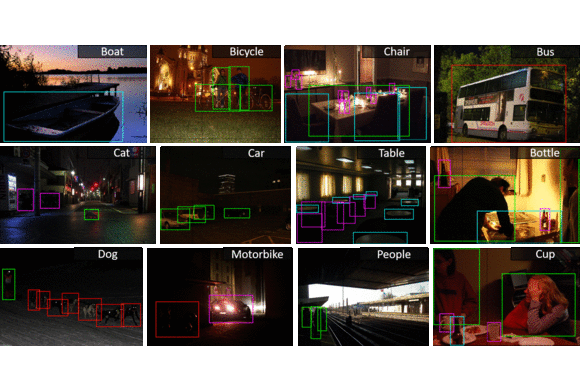
Berdasarkan hasil pengujian, nilai-nilai akurasi dan waktu inferensi dari tiap model akan dibentuk ke dalam plot untuk menghasilkan *Pareto Frontier*. *Pareto Frontier* mengidentifikasi model yang optimal secara relatif, yaitu model yang menawarkan kombinasi terbaik dari mAP tinggi dan waktu inferensi rendah.

1. Analisis Perbandingan

Model-model yang terletak pada *Pareto Frontier* dikategorikan sebagai solusi Pareto optimal. Jika dua model memiliki tingkat mAP yang setara, maka model dengan waktu inferensi yang lebih singkat akan lebih diprioritaskan. Sebaliknya, jika waktu inferensi kedua model serupa, model dengan akurasi yang lebih tinggi akan dianggap lebih unggul. Model-model yang berada di luar Pareto Frontier dinilai kurang efisien karena terdapat model lain yang dapat memberikan kinerja yang lebih bagus, baik dalam mAP maupun waktu inferensi.

# **BAB IV HASIL DAN PEMBAHASAN**

**4.1 Pengambilan data**

Dataset yang digunakan dalam penelitian ini diperoleh dari sebuah publikasi ilmiah yang menyediakan tautan menuju repositori GitHub. Repositori tersebut telah berisi seluruh citra yang dibutuhkan, yang telah dikelompokkan berdasarkan kelas masing-masing, serta dilengkapi dengan anotasi bounding box yang sesuai. Gambar di bawah ini menunjukkan contoh data yang digunakan dalam penelitian ini:

Gambar 4.1 Sampel Exclusively Dark Dataset

Setelah seluruh dataset beserta label anotasinya berhasil dikumpulkan, langkah selanjutnya adalah membagi dataset tersebut menjadi dua bagian, yaitu training set sebesar 80% dan validation set sebesar 20%. Tujuan dari pemisahan ini adalah untuk menjaga kemurnian validation set dari proses preprocessing, sehingga data tersebut dapat merepresentasikan kondisi nyata di dunia sebenarnya. Proses pembagian dilakukan menggunakan teknik stratified random sampling guna memastikan distribusi kelas objek tetap seimbang pada masing-masing subset.

**4.2 *Data Preprocessing***

**4.2.1 *Downsampling***

Tahap awal dalam proses data preprocessing adalah penyesuaian resolusi citra melalui metode downsampling menggunakan algoritma *Lanczos resampling*. Proses ini diterapkan pada training set dengan tujuan untuk mengurangi kompleksitas data sekaligus mempercepat proses pelatihan model.

Langkah pertama dalam penyesuaian resolusi ini dilakukan dengan menguji sejauh mana pengurangan resolusi sebesar 20% dari ukuran asli tetap menghasilkan citra yang dapat digunakan secara efektif. Gambar berikut menunjukkan contoh hasil downsampling sebesar 20% dari resolusi awal:



Gambar 4.2 Downsampling sebanyak 20%

Berdasarkan penilaian subjektif dari peneliti, kualitas visual citra hasil *downsampling* sebesar 20% masih dianggap memadai untuk digunakan dalam pelatihan model. Oleh karena itu, dilakukan pengujian lanjutan dengan mengurangi resolusi lebih lanjut, yakni sebesar 40% dari resolusi asli. Contoh hasil *downsampling* sebesar 40% ditunjukkan pada gambar berikut:



Gambar 4.3 Downsampling sebanyak 40%

Hasil visual yang ditunjukkan di atas menunjukkan bahwa *downsampling* hingga 40% tetap menghasilkan citra yang layak untuk digunakan. Oleh karena itu, pengurangan resolusi sebesar ini dianggap sesuai untuk digunakan dalam tahap *preprocessing*, karena selain mempercepat proses pemrosesan data, juga dapat membantu mengurangi *irrelevant noise* dan memangkas *unnecessary details* sehingga model dapat mencapai generalisasi yang lebih optimal.

**4.2.2 Pengubahan format label *bounding box***

Setelah memperoleh citra beserta anotasi *bounding box*-nya, langkah selanjutnya adalah melakukan konversi format *bounding box* ke dalam bentuk yang sesuai dengan standar yang dapat diterima oleh model-model yang digunakan dalam penelitian ini, yaitu YOLOv9, RT-DETR, dan YOLOv10.

Format awal anotasi *bounding box* yang diperoleh adalah sebagai berikut:

% bbGt version=3

Bicycle 204 28 271 193 0 0 0 0 0 0 0

Format tersebut kemudian dikonversi ke dalam format baru yang telah disesuaikan dengan kebutuhan masing-masing arsitektur model. Berikut adalah representasi hasil konversi *bounding box* ke dalam format akhir yang digunakan dalam proses pelatihan model:

[ PLACEHOLDER ]

**4.2.2 *Contrast Enhancement***

Teknik *contrast enhancement* yang diterapkan dalam penelitian ini adalah *Contrast Limited Adaptive Histogram Equalization* (CLAHE). Metode CLAHE bekerja dengan menerapkan peningkatan kontras secara lokal pada gambar, sehingga menghasilkan citra dalam *grayscale* dengan tingkat kecerahan yang disesuaikan secara adaptif, terutama pada area-area yang sebelumnya kurang pencahayaan. Contoh hasil pemrosesan citra menggunakan metode CLAHE ditunjukkan pada gambar berikut:

[ PLACEHOLDER ]

**4.2.2 *Data Augmentation***

Tahap terakhir dalam proses *preprocessing* adalah *data augmentation*, yang dilakukan menggunakan modul *Albumentations*. Augmentasi ini diterapkan pada data *training* sebanyak {x} citra, sehingga jumlah total citra setelah augmentasi menjadi dua kali lipat, yaitu sebanyak {x+x}. Dengan menggunakan parameter-parameter augmentasi yang telah ditentukan, contoh hasil augmentasi dapat dilihat pada gambar berikut:

[ PLACEHOLDER ]

Langkah lanjutan dalam proses augmentasi adalah penerapan algoritma *mosaic*, yang menggabungkan enam gambar berbeda dalam satu citra gabungan. Dengan konfigurasi seperti ini, jumlah total citra pada *training set* bertambah menjadi {m}. Contoh hasil dari teknik *mosaic augmentation* ditunjukkan pada gambar berikut:

[ PLACEHOLDER ]

**4.3 Implementasi YOLOv9**

**4.3.1 Pelatihan YOLOv9**

Model YOLOv9 yang digunakan dalam penelitian ini terdiri dari tiga varian, yaitu YOLOv9s, YOLOv9m, dan YOLOv9c. Setiap varian model dilatih menggunakan konfigurasi *hyperparameter* yang telah ditentukan sebelumnya melalui pendekatan *grid search*, yang menghasilkan empat kombinasi parameter berbeda. Dengan demikian, total model yang dihasilkan dari proses pelatihan YOLOv9 adalah sebanyak 12 model (3 varian × 4 kombinasi). Seluruh hasil pelatihan dari model-model tersebut dirangkum dalam Tabel {y}.

| BATCH 8 | Lr 0.01 | | | Lr 0.001 | | |
| --- | --- | --- | --- | --- | --- | --- |
| s | m | c | s | m | c |
| total\_epoch | 153 | 123 | 182 | 50 | 36 | 38 |
| total\_time | 24528.6 | 17141.7 | 27158.1 | 7983.14 | 4994.8 | 5658.07 |
| train/box\_loss | 116.087 | 100.368 | 0.85547 | 116.476 | 104.929 | 0.97495 |
| train/cls\_loss | 0.93253 | 0.74765 | 0.61091 | 0.91759 | 0.75938 | 0.66978 |
| train/dfl\_loss | 13.208 | 119.547 | 109.861 | 131.245 | 121.221 | 115.929 |
| metrics/precision(B) | 0.79881 | 0.80415 | 0.78834 | 0.77952 | 0.81313 | 0.82603 |
| metrics/recall(B) | 0.70113 | 0.70362 | 0.71021 | 0.69448 | 0.72082 | 0.72515 |
| metrics/mAP50(B) | 0.7792 | 0.77866 | 0.77686 | 0.76621 | 0.80037 | 0.80051 |
| metrics/mAP50-95(B) | 0.50553 | 0.5069 | 0.50874 | 0.4986 | 0.52844 | 0.53259 |
| val/box\_loss | 133.479 | 135.412 | 137.849 | 131.088 | 129.887 | 129.319 |
| val/cls\_loss | 0.90956 | 0.92488 | 0.95876 | 0.93619 | 0.88004 | 0.87685 |
| val/dfl\_loss | 154.166 | 155.132 | 160.583 | 14.645 | 144.284 | 144.121 |

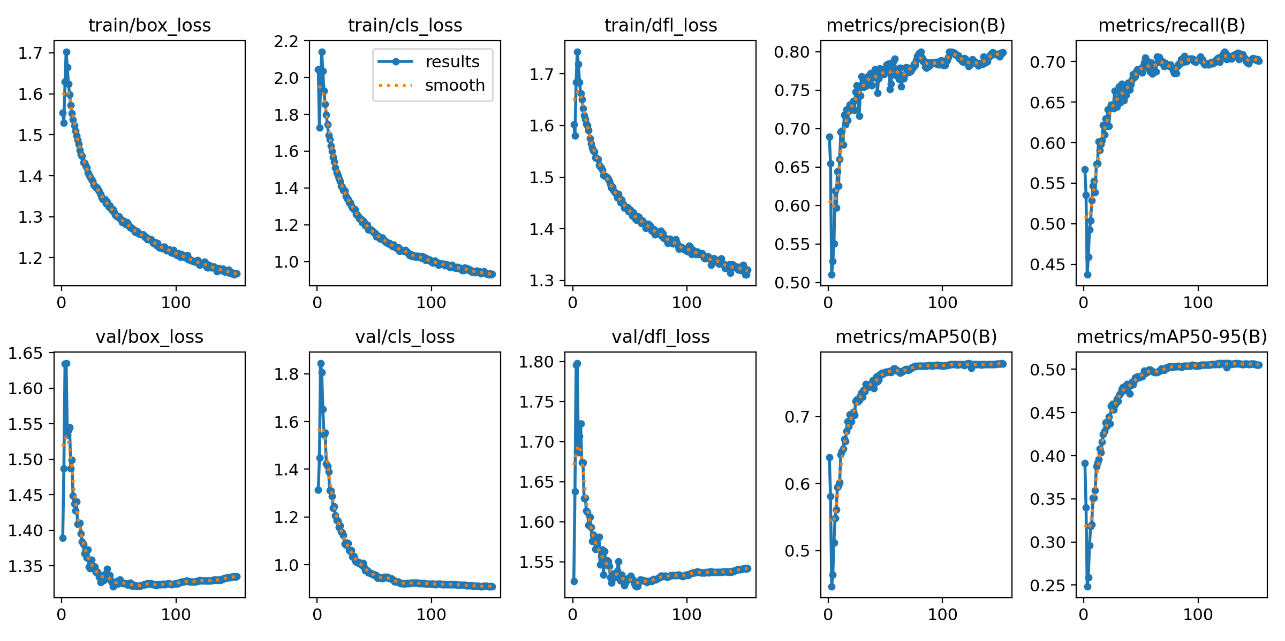
| BATCH 16 | Lr 0.01 | | | Lr 0.001 | | |
| --- | --- | --- | --- | --- | --- | --- |
| s | m | c | s | m | c |
| total\_epoch | 139 | 97 | 31 | 46 | 36 | 43 |
| total\_time | 13430.4 | 11619.9 | 4144.79 | 4433.32 | 4323.72 | 5735.19 |
| train/box\_loss | 113.872 | 100.394 | 119.269 | 114.262 | 101.155 | 0.91356 |
| train/cls\_loss | 0.88159 | 0.73445 | 0.95252 | 0.8772 | 0.69955 | 0.60739 |
| train/dfl\_loss | 129.174 | 118.485 | 132.722 | 127.248 | 117.283 | 110.651 |
| metrics/precision(B) | 0.77846 | 0.78923 | 0.77959 | 0.78191 | 0.81178 | 0.82808 |
| metrics/recall(B) | 0.70355 | 0.71027 | 0.66315 | 0.68447 | 0.70649 | 0.71274 |
| metrics/mAP50(B) | 0.77258 | 0.7757 | 0.75295 | 0.7596 | 0.789 | 0.79429 |
| metrics/mAP50-95(B) | 0.50055 | 0.50545 | 0.47686 | 0.49252 | 0.51436 | 0.52796 |
| val/box\_loss | 134.283 | 134.797 | 136.475 | 132.618 | 133.798 | 131.601 |
| val/cls\_loss | 0.931 | 0.93049 | 100.895 | 0.96442 | 0.90462 | 0.90149 |
| val/dfl\_loss | 154.146 | 154.237 | 152.299 | 14.601 | 146.205 | 14.593 |

**4.3.2 Kombinasi YOLOv9 terbaik**

Berdasarkan evaluasi menggunakan metrik yang tercantum pada Tabel {y}, diperoleh kombinasi *hyperparameter* terbaik untuk masing-masing varian YOLOv9 sebagaimana dijelaskan pada bagian berikut.

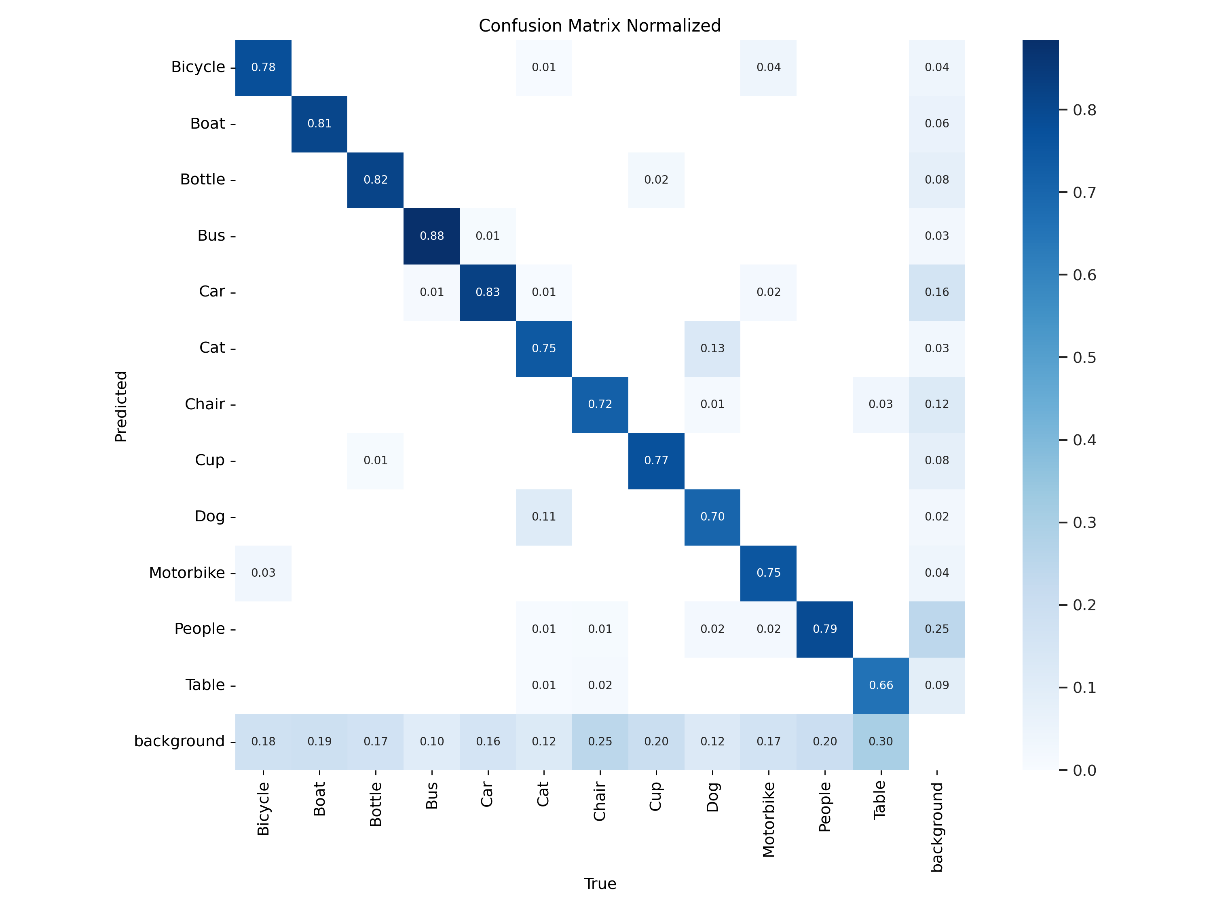
| Tipe Model | Batch Size | Learning Rate | mAP50 | map50-95 | val/cls\_loss |
| --- | --- | --- | --- | --- | --- |
| YOLOv9s | 8 | 0.01 | 0.7792 | 0.50553 | 0.90956 |
| YOLOv9m | 8 | 0.001 | 0.80037 | 0.52844 | 0.88004 |
| YOLOv9c | 8 | 0.001 | 0.80051 | 0.53259 | 0.87685 |

**4.3.2.1 YOLOv9s**

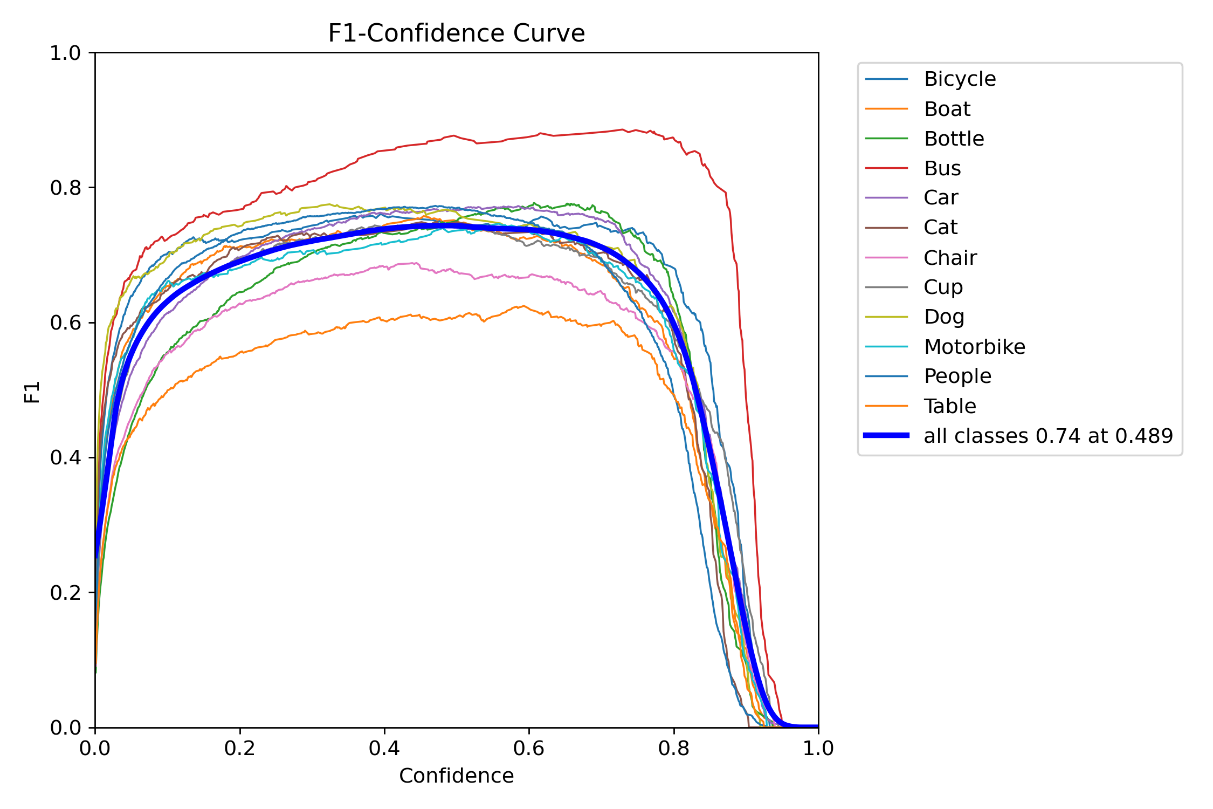


Gambar {x} di atas menunjukkan grafik hasil pelatihan model YOLOv9s dengan konfigurasi *epoch 1000*, *optimizer SGD*, dan *patience 30*. Grafik *train/box\_loss, train/cls\_loss*, dan *train/dfl\_loss* secara konsisten menurun dari nilai yang relatif tinggi di awal menuu nilai yang lebih rnedah dan stabil di akhir pelatihan. Pada grafik *metrics/precision(B)* dan *metrics/recall(B)*, nilai kedua metrik cenderung meningkat secara bertahap, menandakan model semakin mampu mengurangi *false positives* dan *false negatives* seiring bertambahnya epoch*.*

Selanjutnya, tren *loss* pada *validation set*, yaitu *val/box\_loss, val/cls\_loss*, dan *val/dfl\_loss*, menunjukkan pola fluktuatif cukup luas yang mungkin saja terjadi karena kapasitas modelnya yang kecil sehingga lebih sensitif terhadap perubahan kondisi data dan *hyperparameter*. *Overfitting* juga dapat terjadi jika model terlalu terpaku pada *train set*, tetapi penerapan mekanisme *early stopping* berupa *patience 30* dapat menahan masalah ini agar tidak berlanjut secara ekstrim. Peningkatan metrik *mAP50(B)* dan *map50-95(B)* di *validation set* memperlihatkan bahwa secara umum performa deteksi YOLOv9s kian membaik setelah beberapa epoch tertentu, menandakan kemampuan model untuk secara konsisten menempatkan *bounding box* dengan tepat pada berbagai rentang IoU. Meski demikian, karena YOLOv9s memiliki arsitektur yang sangat ringan, lanju peningkatan mAP mungkin sedikit lambat untuk mencapai nilai puncak yang setara dengan model yang secara parameter lebih besar dan memiliki jaringan yang lebih dalam.



Gambar {x} merupakan gambar *confusion matrix normalized* untuk YOLOv9s yang memperlihatkan hasil cukup akurat, dapat telrihat dari nilai diagonal yang relatif tinggi. Namun, beberapa kelas dengan kemiripan visual seperti *Bicycle* dan *Motorbike* atau *Chair* dan *Table*, masih sering tertukar, terutama karena kesulitan mengekstrasi fitur halus terlebih lagi pada kondisi pencahayaan rendah. Selain itu, beberapa objek yang cenderung kecil atau kurang kontras seperti *Bottle* terkadan diklasifikasikan sebagai latar belakang.



Gambar {x} di atas adalah kurva *F1-Confidence* untuk YOLOv9s yang menunjukkan bagaimana *precision* dan *recall* saling berinteraksi pada variasi nilai *threshold* *confidence* pada model. Beberapa kelas yang memiliki kontur visual lebih jelas, seperti *Bus* dan *Car*, terlihat mencapat puncak F1 pada level *confidence* menengah hingga tinggi, menandakan bahwa model memiliki keyakinan yang cukup kuat untuk mendeteksi objek tesebut tanpa banyak mengorbankan *recall*. Namun, kurva untuk kelas-kelas yang cenderung mirip atau sering tumpang tindih seperti *Chair* dan *Table*, relatif lebih rentan terpengaruh oleh perubahan *threshold confidence*.

|  |  |  |
| --- | --- | --- |
| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| **1.** |  |  |
| **2.** |  |  |
| **3.** |  |  |

Tabel {x} di atas menunjukkan sampel dari hasil prediksi YOLOv9s yang umumnya sudah mendekati *ground truth*, dimana beberapa objek berhasil terdeteksi dengan *confidence* yang relatif baik meski berada pada kondisi cahaya minim. YOLOv9s ini juga mampu mendeteksi objek yang tidak diberi label dalam *ground* *truth*, contohnya seperti objek *Table* pada gambar 1b. Namun, tampak pula adanya perbedaan letak dan skala *bounding box* pada beberapa objek seperti, khususnya jika objek tersebut tumpang tindih atau berukuran kecil seperti pada gambar 2a dan 2b.

**4.3.2.2 YOLOv9m**

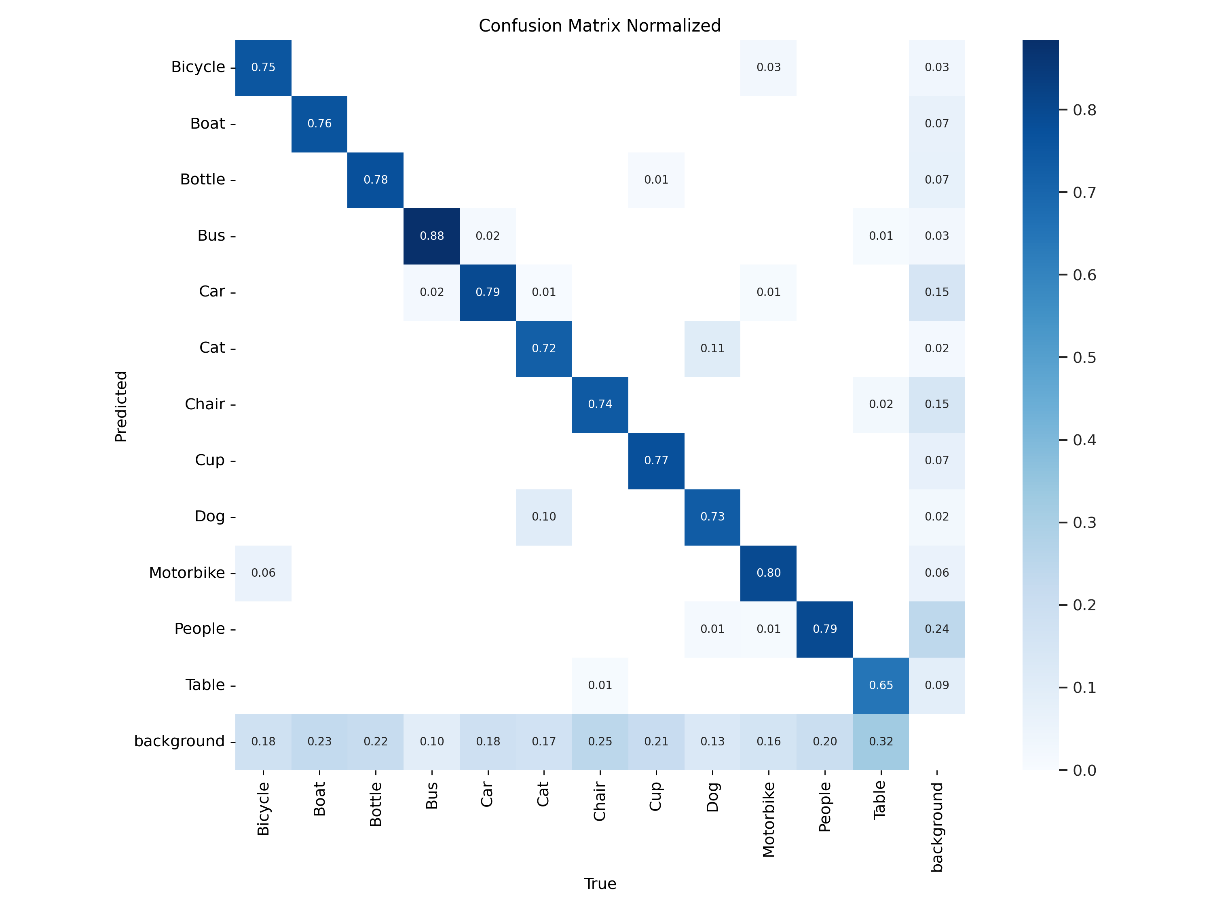


Gambar {x} menunjukkan beberapa metirk utama yang digunakan untuk mengevaluasi kinerja kombinasi terbaik model YOLOv9m. Pada *train/box\_loss , train/cls\_loss,* dan *train/dfl\_loss*, secara umum terlihat tren menurun yang menandakan bahwa proses pelatihan berjalan cukup baik. Pada awal pelatihan, nilai ketiga *loss* tersebut cukup tinggi, kemudian secara konsisten menurun dan semakin stablik mendekati titik konvergen di akhir pelatihan. Data ini menunjukkan bahwa model semakin terlatih dalam mengekstraksi fitur dan menyesuaikan *weights* sesuai target.

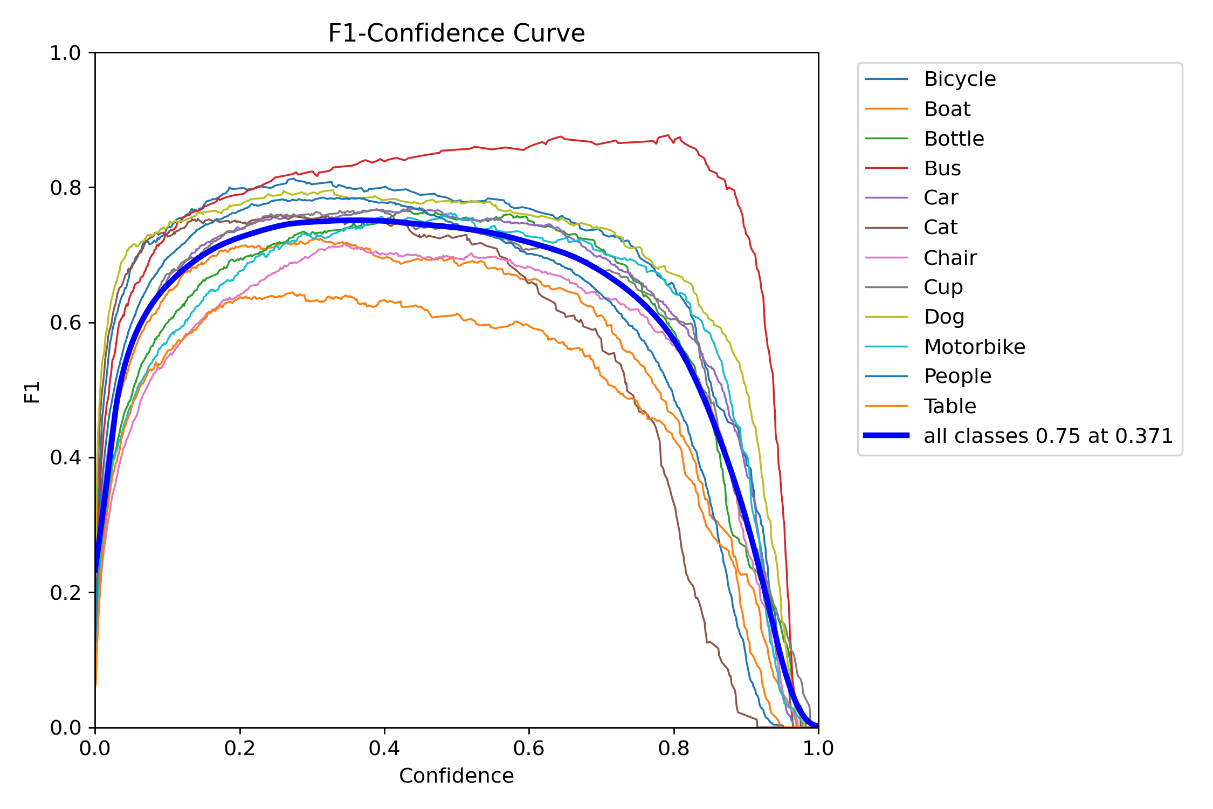
Selanjutnya pada kurva metrik evaluasi di bagian training untuk *precision dan recall*, tampak terjaid peningkatan bertahap sepanjang epoch. Peningkatan *precision* mengindikasikan bahwa model semakin jarang memberikan *false positives*. Sementara itu peningkatan *recall* menandakan model semakin jarang melakukan *false negatives*.

Jika dibandinkan dengan grafik pada *validation set, val/box\_loss, val/cls\_loss,* dan *val/dfl)loss*, terlihat bahwa ketiga loss di set validasi cenderung fluktuatif. Pada beberapa titik bahkan mengalami kenaikan, yang dapat menandakan potensi *overfitting*. Meski demikian, adanya mekanisme *early stopping* dengan *patience = 30* dapat membantu menghentikan pelatihan ketika performa validasi tidak lagi meningkat dalam 30 epoch, sehingga diharapkan dapat mengurangi efek *overfitting* ini.

Di sisi lain, *map50* dan *map50-95* pada set validasi menunjukkan peningkatan secara umu, meskipun terlihat sedikit ketidakstabilan di beberapa bagian. Data ini menunjukkan bahwa model semakin mampu melakukan deteksi dengan akurasi lebih baik pada berbagai nilai IoU.



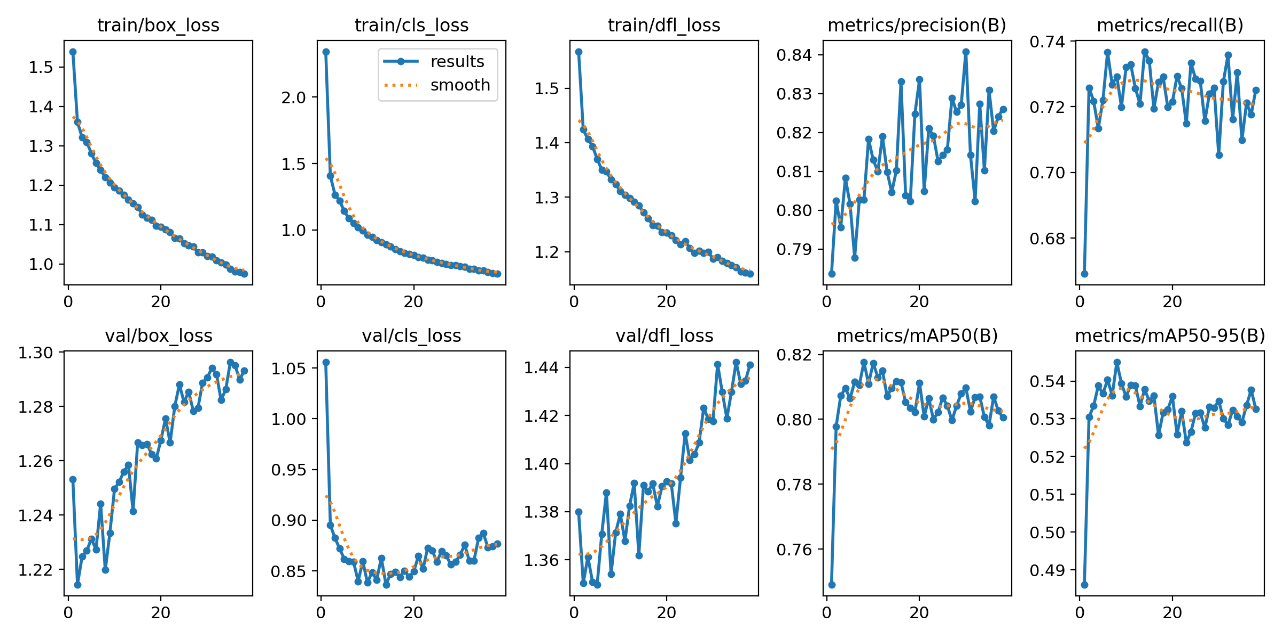
Nilai diagonal yang tinggi pada Gambar {x} di atas menunjukkan bahwa meskipun model secara umum mampu melakukan deteksi dengan benar, terdapat kesalahan klasifikasi antar kelas dengan kemiripan fitur, seperti *Bicycle* dengan *Motorbike* dan *Chair* dengan *Table*, yang mengindikasikan ambiguitas dalam ekstraksi fitur halus pada kondisi pencahayaan rendah.



Gambar {x} di atas menunjukkan sebagian besar kelas mencapai puncah F1 pada kisaran *confidence* yang relatif mirip, mencerminkan bahwa terdapat *sweet spot* dimana model cukup yakin saat mendeteksi objek tanpa mengorbankan terlalu banyak *recall*. Dalam kondisi pencahayaan rendah, fluktuasi bentuk kurva di area puncak mengindikasikan bahwa untuk beberapa kelas, perubahan sedikit saja pada *threshold* dapat memengaruhi *precision* dan *recall* secara signifikan.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** |  |  |
| **2.** |  |  |
| **3.** |  |  |

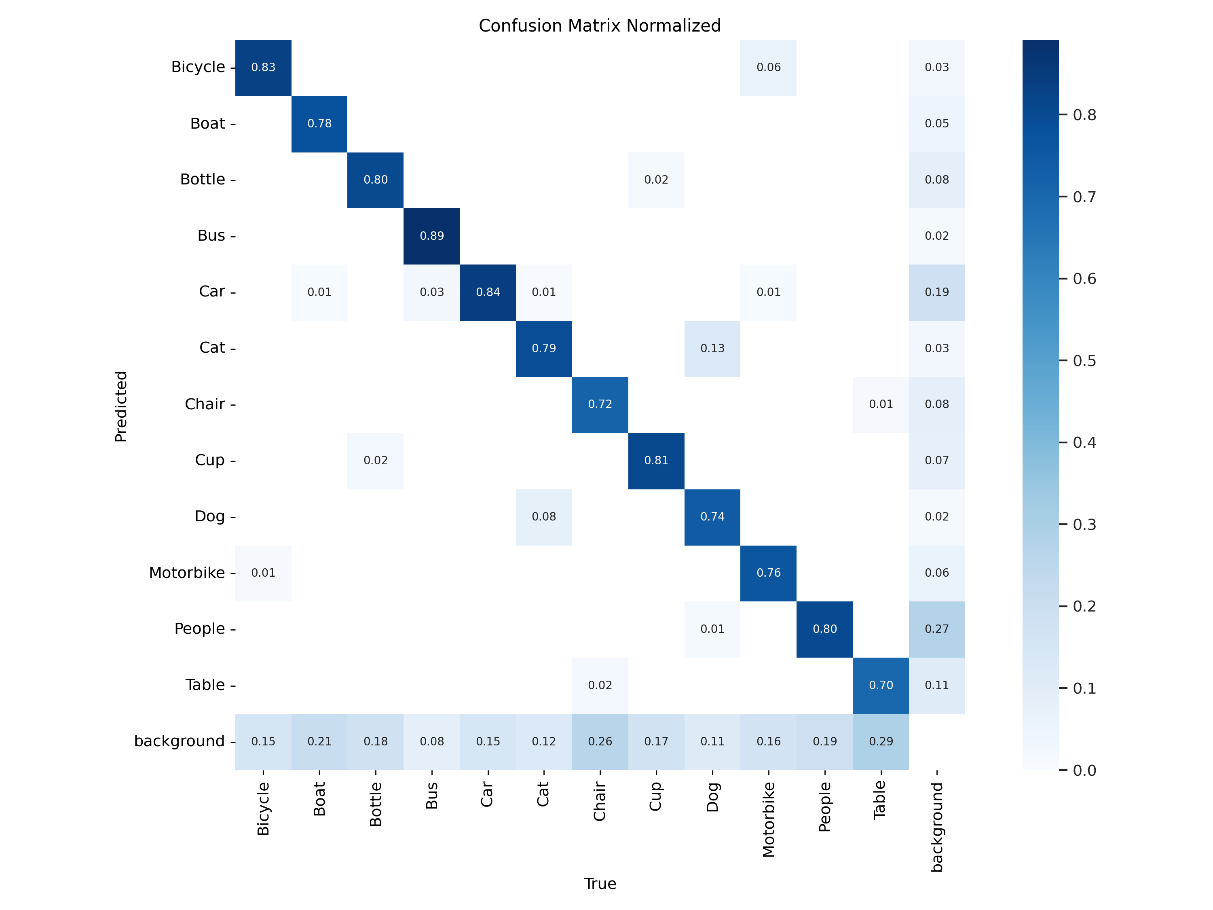
Tabel {x} di atas adalah sampel hasil dari pendeteksian objek oleh model YOLOv9m. YOLOv9m yang memiliki jumlah parameter lebih besar daripada YOLOv9s tampak lebih mampu mendeteksi objek dengan lebih akurat dan stabil. Dari tabel terlihat bahwa *bounding box* yang dihasilkan lebih mendekati *ground truth,* khususnya pada objek yang berukuran kecil atau sedikit tumpang tindih seperti pada gambar 2b dan 3b. Walau demikian, masih terdapat pula beberapa prediksi yang belum sempurna seperti terdeteksinya dua *People* dalam gambar 1b. Secara keseluruhan, peningkatan jumlah parameter pada YOLOv9m membantu model mengenali objek dengan *confidence* yang lebih besar dibandingkan YOLOv9s.

**4.3.2.3 YOLOv9c**

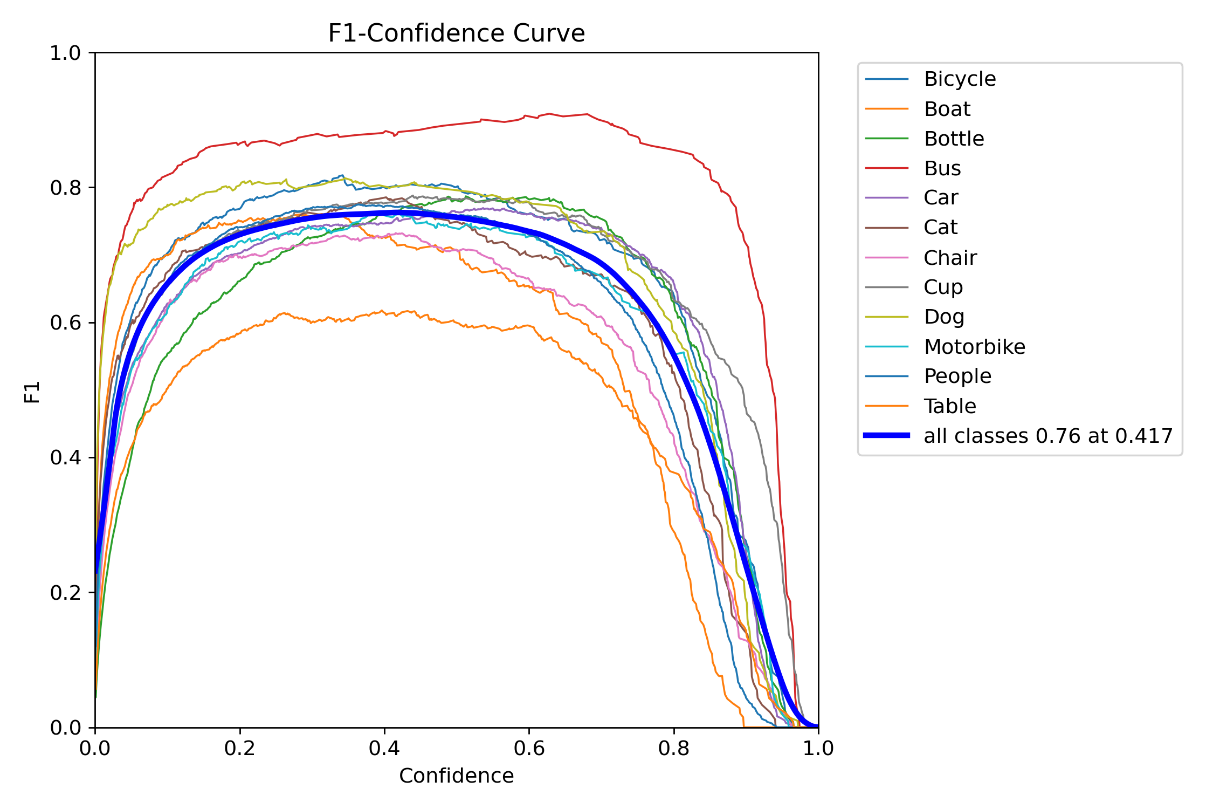
Gambar {x} di atas menunjukkan grafik dari hasil pelatihan YOLOv9c dengan *optimizer SGD*, selama 1000 epoch. Terlihat bahwa *train loss* secara konsisten menurun dari nilai awal yang tinggi menuju nilai yang lebih stabil di akhir proses. Hal ini menunjukkan bahwa model mampu mempelajari fitur-fitur penting dalam dataset. Selain itu, metrik *precision* dan *recall* di sisi *training* mengalami peningkatan seiring waktu, menandakan bahwa model semakin tepat dalam melakukan prediksi *bounding box* serta semakin jarang melewatkan objek.

Pada bagian validasi, *val loss* cenderung kebih fluktuatif meski terdapat kecenderungan penurunan secara umum. Fluktuasi ini bisa saja menandakan bahwa model mengalami kesulitan dalam menyelaraskan pembelajaran pada data yang belum pernah dilihat atau adanya kecenderungan *overfitting,* khususnya saat jumlah epoch terus bertambah. Namun, mekanisme *early stopping* dengan *patience 30* dapat membantu mengontrol hal tersebut agar pelatihan berhenti ketika peningkatan kinerja tidak lagi signifikan.

Dari segi metrik mAP, baik mAP50(B) maupun mAP50-95(B) mengalami tren kenaikan yang menggambarkan kemampuan model untuk mendeteksi objek dengan berbagai tingkat IoU. Peningkatan ini mengindikasikan bahwa YOLOv9c berhasil menyerap pola-pola penting pada kondisi pencahayaan rendah dan mempertahankan kinerja generalisasi yang relatif baik.



Gambar {x} di atas merupakan *confusion matrix normalized* untuk YOLOv9c yang menunjukkan peningkatan akurasi cukup signifikan dengan nilai diagonal rata-rata lebih tinggi pada kelas *Bus* atau *Bicycle* yang mencapai lebih dari 0,8. Namun terlihat pula beberapa *misclassifications* yang masih terjadi, seperti *People* yang kadan terklasifikasi menjadi *background* atau *Motorbike*, membuktikan bahwa objek dengan kontur yang tidak terlalu tajam atau posisi yang tumpang tindih masih menjadi hal yang cukup sulit bahkan bagi model terbesar untuk YOLOv9 pada penelitian ini.

****

Gambar {x} di atas merupakan kurva *F1-Confidence* untuk YOLOv9c. Kurva masing-masing kelas cenderung naik hingga mencapai puncak F1 di kisaran *confidence* tertentu, kemudian turun drastis ketika *threshold* semakin ketat. Hal ini mengindikasikan bahwa model YOLOv9c, dengan kapasitas jaringan yang lebih besar, mampu menyeimbangkan *precision* dan *recall* yang lebih optimal.

Dari sisi implementasi, penentuan *threshold* yang tepat menjadi kunci untuk mendapatkan hasil deteksi terbaik. Pada titik F1 tertinggi (misalnya pada *confidence* 0,4-0,5), model dapat menghasilkan prediksi yang relatif baik dengan minim *false positives* maupun *false negatives*. Namun di kelas tertentu yang bentuknya lebih samar atau sering tumpang tindih, kurvanya dapat berbeda sedikit lebih rendah dan rentan berubah ketika *threshold* dinaikkan atau diturunkan.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** |  |  |
| **2.** | **A screenshot of a computer  AI-generated content may be incorrect.** |  |
| **3.** | **A car parked on the street  AI-generated content may be incorrect.** |  |

Pada tabel {x}, terlihat sampel hasil prediksi YOLOv9c yang menunjukkan deteksi objek yang lebih mendekati *ground* truth dari segi *bounding box* dengan *confidence* yang lebih stabil. Meski masih terdapat sedikit kekeliruan pada beberapa objek yang penampakannya serupa atau berdekatan seperti pada gambar 1a, YOLOv9c dengan parameter terbesar di keluarga YOLOv9 membantu mengenali ciri-ciri visual lebih baik, sehingga secara umum hasil deteksi berada lebih dekat dengan *ground truth* dibanding model-model sebelumnya.

**4.4 Implementasi RT-DETR**

**4.4.1 Pelatihan RT-DETR**

Model RT-DETR yang digunakan terdiri dari dua varian, yaitu RTDETR-L dan RTDETR-X. Sama seperti pada YOLOv9, masing-masing varian dilatih menggunakan empat kombinasi *hyperparameter* hasil dari teknik *grid search*, sehingga menghasilkan total 8 model (2 varian × 4 kombinasi). Hasil dari seluruh pelatihan model RT-DETR ini dirangkum dalam Tabel {y2}.

| BATCH 8 | Lr 0.01 | | Lr 0.001 | |
| --- | --- | --- | --- | --- |
| L | X | L | X |
| total\_epoch | 113 | 138 | 54 | 59 |
| total\_time | 33011.3 | 50155.7 | 15668.5 | 21347.9 |
| train/giou\_loss | 0.44008 | 0.50669 | 0.41779 | 0.38656 |
| train/cls\_loss | 0.6008 | 0.58061 | 0.59373 | 0.56301 |
| train/l1\_loss | 0.19798 | 0.20241 | 0.20204 | 0.18394 |
| metrics/precision(B) | 0.80956 | 0.81877 | 0.83277 | 0.84243 |
| metrics/recall(B) | 0.71213 | 0.69621 | 0.68728 | 0.72232 |
| metrics/mAP50(B) | 0.76971 | 0.76535 | 0.75949 | 0.79138 |
| metrics/mAP50-95(B) | 0.4838 | 0.47706 | 0.48304 | 0.50915 |
| val/giou\_loss | 0.46754 | 0.47455 | 0.46317 | 0.45104 |
| val/cls\_loss | 0.67114 | 0.66686 | 0.68041 | 0.64739 |
| val/l1\_loss | 0.38581 | 0.38788 | 0.37794 | 0.36973 |

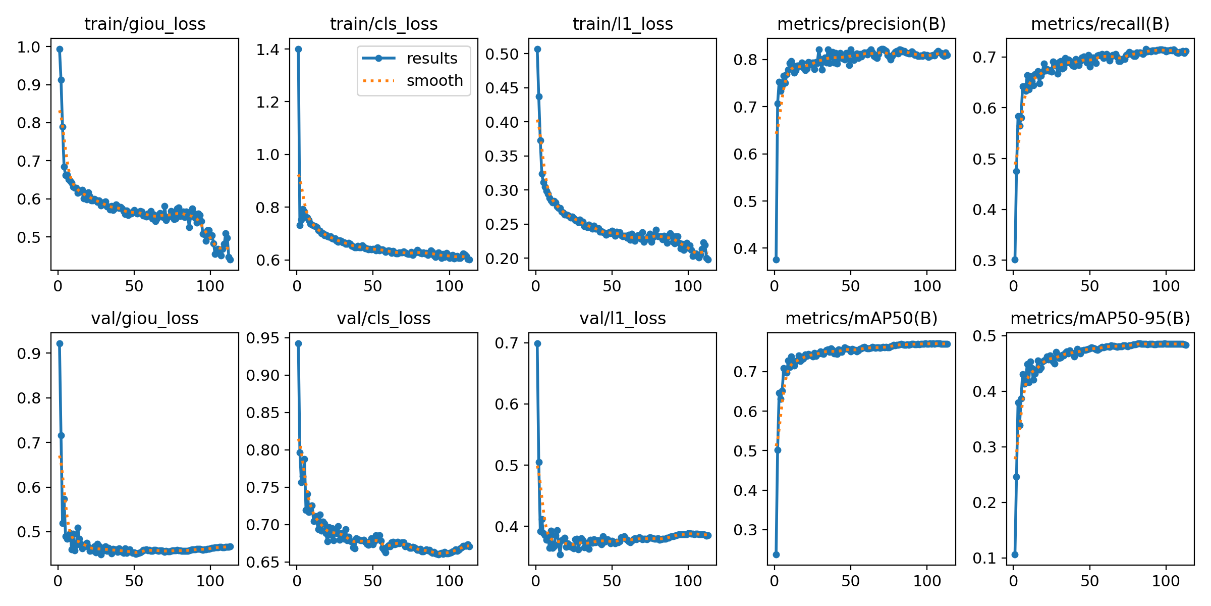
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BATCH 16 | Lr 0.01 | | Lr 0.001 | |
| L | X | L | X |
| total\_epoch | 104 | 61 | 48 | 48 |
| total\_time | 23291 | 17971.5 | 10759.5 | 14122.8 |
| train/giou\_loss | 0.47121 | 0.45605 | 0.41929 | 0.39596 |
| train/cls\_loss | 0.5811 | 0.58457 | 0.58013 | 0.55788 |
| train/l1\_loss | 0.19763 | 0.19389 | 0.19391 | 0.18118 |
| metrics/precision(B) | 0.80266 | 0.81163 | 0.81956 | 0.85686 |
| metrics/recall(B) | 0.68876 | 0.70686 | 0.68933 | 0.73583 |
| metrics/mAP50(B) | 0.74552 | 0.76933 | 0.75277 | 0.80794 |
| metrics/mAP50-95(B) | 0.47541 | 0.48726 | 0.47302 | 0.51841 |
| val/giou\_loss | 0.46774 | 0.47212 | 0.48153 | 0.44628 |
| val/cls\_loss | 0.69491 | 0.67639 | 0.69523 | 0.64611 |
| val/l1\_loss | 0.37382 | 0.38233 | 0.38661 | 0.35431 |

**4.4.2 Kombinasi RTDETR terbaik**

Berdasarkan metrik yang ditampilkan pada Tabel {y2}, diperoleh konfigurasi *hyperparameter* terbaik untuk masing-masing varian RT-DETR, yang dijelaskan lebih lanjut pada bagian berikut.

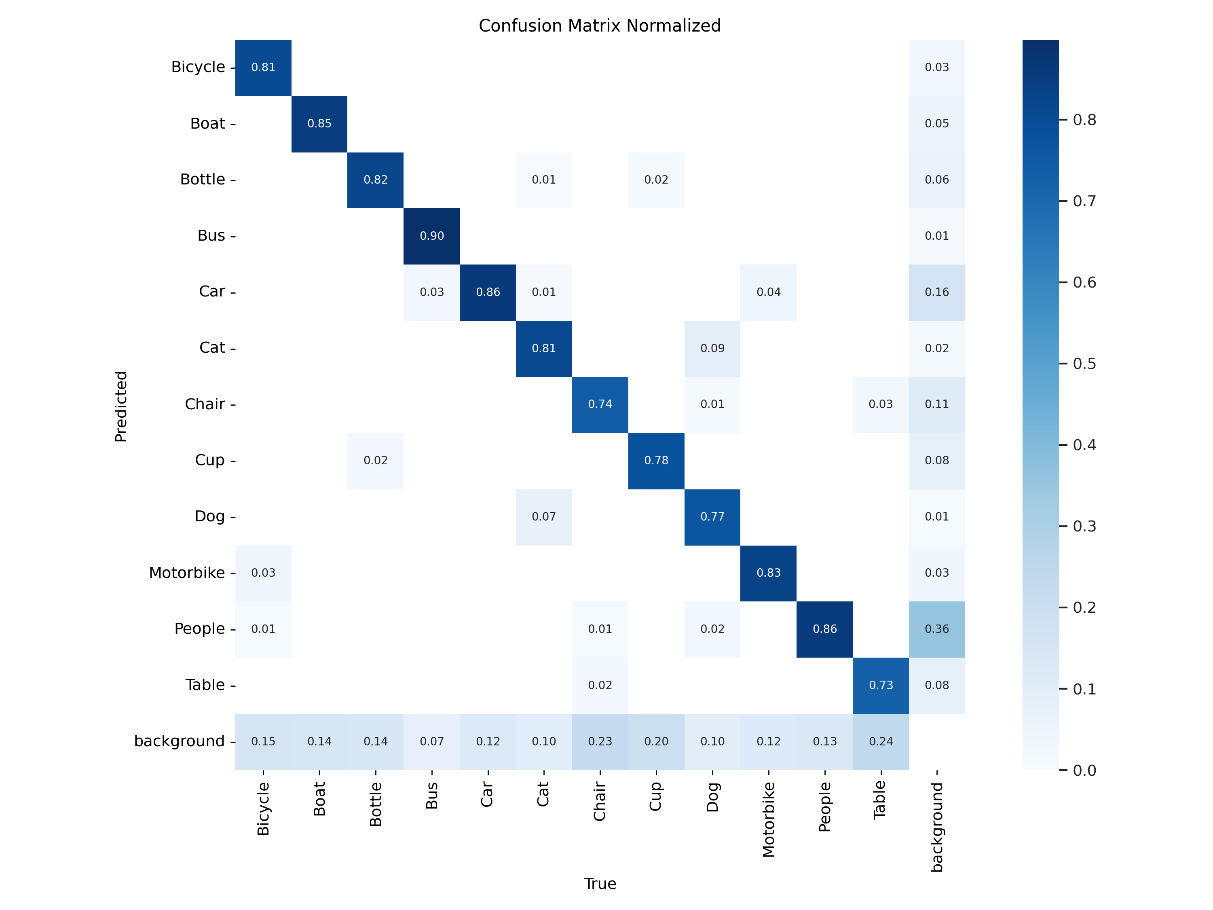
| Tipe Model | Batch Size | Learning Rate | mAP50 | map50-95 | val/cls\_loss |
| --- | --- | --- | --- | --- | --- |
| RTDETR-L | 8 | 0.01 | 0.76971 | 0.4838 | 0.67114 |
| RTDETR-X | 16 | 0.001 | 0.80794 | 0.51841 | 0.64611 |

**4.4.2.1 RTDETR-L**

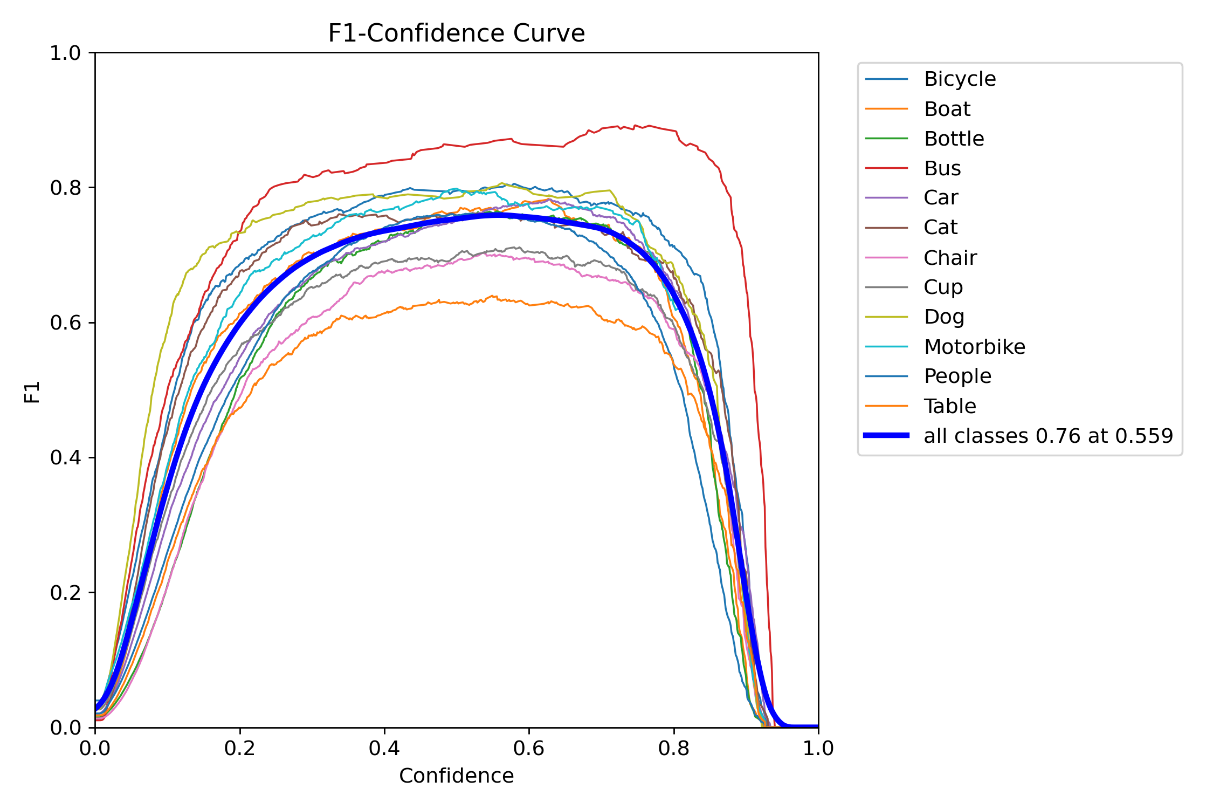


Gambar {x} merupakan kurva hasil pelatihan RTDETR-L. Terlihat bahwa performa model meningkat secara signifikan seiring bertambahnya epoch. Hal ini ditunjukkan dari tren penurunan konsisten pada metrik *train loss* utama yaitu *train/giou\_loss, train/cls\_loss,* dan train/*l1\_loss,* yang mengindikasikan bahwa model berhasil mempelajari representasi fitur spasial, kelas objek, dan posisi dengan baik. Kinerja ini juga tercermin pada metrik evaluasi seperti *precision* dan *recall* di data pelatihan yang mengalami peningkatan cukup tajam di awal dan kemudian stabil pada nilai tinggi.

Evaluasi pada data validasi pun memberikan hasil yang sejalan. Grafik *val/giou\_loss*, *val\_cls\_loss*, dan *val/l1\_loss* menunjukkan pola penurunan yang stabil hingga mencapai titik konvergen, tanpa lonjakan signifikan yang menandakan *overfitting*. Hal ini memperkuat bukti bahwa model mampu melakukan generalisasi terhadap data baru yang tidak dilihat selama pelatihan. Selain itu, metrik mAP50(B) dan mAP50-95(B) yang terus meningkat dan stabil di akhir pelatihan menjadi indikator utama bahwa RTDETR-L menghasilkan prediksi *bounding box* yang presisi pada berbagai tingkatan *threshold* IoU.

****

Gambar {x} di atas adalah gambar *confusion matrix normalized* untuk RTDETR-L yang memperlihatkan bahwa model dengan parameter besar ini mampu membedakan mayoritas kelas dengan akurasi tinggi, dapat dilihat dari nilai diagonal yang dominan di atas 0,8 untuk beberapa kelas seperti *Bus, Car*, dan *Dog*. Hal ini menunjukkan kapasitas model yang lebih besar membantu mengenali detail visual walaupun pencahayaan minim. Meski demikian, terlihat pula kekeliruan pada beberapa kelas yang memiliki kesamaan bentuk, misalnya *Motorbike* yang terkadang terklasifikasi sebagai *Bicycle*, menandakan bahwa fitur dan kontur halus lagi-lagi masih sulit untuk dibedakan sepenuhnya oleh model.

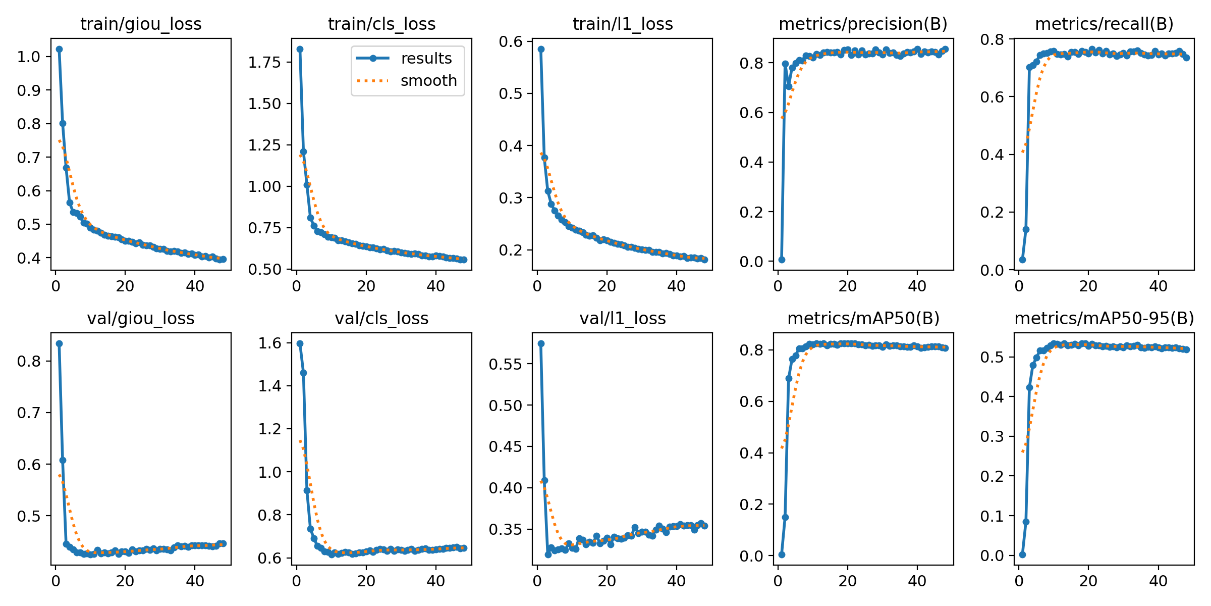
****

Gambar {x} di atas merupakan kurva *F1-Confidence* untuk RTDETR-L yang memperlihatkan bahwa model ini mampu mencapai keseimbangan *precision* dan *recall* di rentang *confidence* tertentu yang relatif lebih luas dibandingkan model YOLOv9 secara keseluruhan. Secara umum, mayoritas kelas menunjukkan kenaikan F1 yang cukup tajam di awal saat *confidence threshold* masih rendah, kemudian mencapai titik puncak dan mulai menurun ketika *threshold* semakin tinggi. Meskipun demikian, terdapat pula kelas tertentu yang kurvanya relatif lebih cepat turun saat *confidence* *threshold* meningkat, mengindikasikan bahwa perbedaan visual objek pada cahaya rendah masih saja menimbulkan tantangan pada pemisahan fitur yang halus.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** |  |  |
| **2.** |  |  |
| **3.** |  |  |

Tabel {x} di atas merupakan sampel hasil dari model RTDETR-L. Dengan parameter yang lebih besar dibandingkan dengan YOLOv9, RTDETR-L tampak menandingi bahkan melebihi akurasi deteksi, terutama pada objek yang lebih kecil atau saling berdekatan seperti pada gambar 3b. Model ini secara keseluruhan memiliki prediksi *bounding box* yang lebih tepat dan *confidence score* yang relatif stabil.

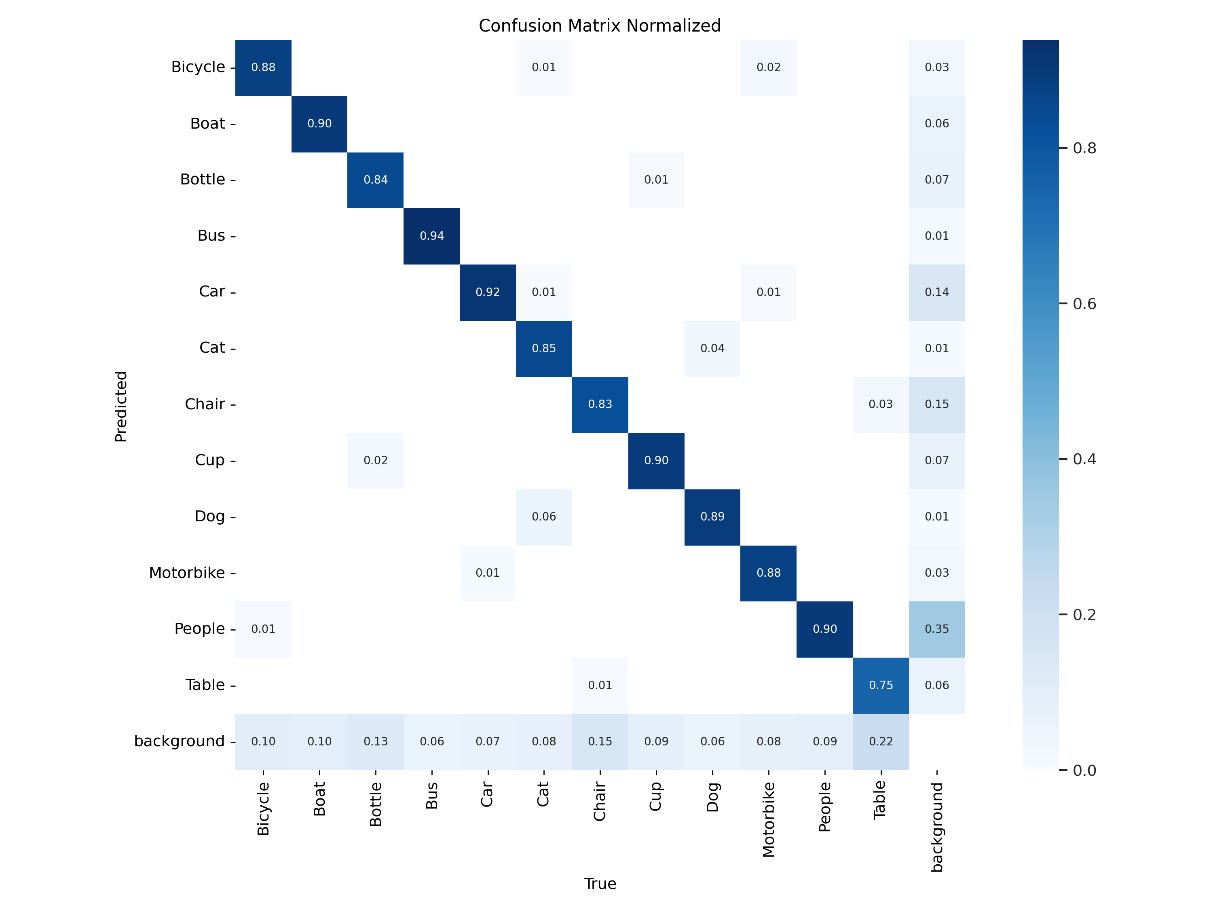
**4.4.2.2 RTDETR-X**



Gambar {x} di atas adalah gambar performa model RTDETR-X selama pelatihan. Tampak bahwa *train loss* turun secara konsisten, menandakan bahwa model dengan arsitektur serta parameter yang paling besar ini mampu mempelajari fitur-fitur penting pada citra dengan efektif. Tren kurva *precision* dan *recall* pada *train set* juga menunjukkan peningkatan yang stabil hingga mencapai nilai mendekati konvergen di akhir pelatihan, menandakan model semakin baik dalam mendeteksi objek dengan akurasi tinggi dan minim kesalahan deteksi.

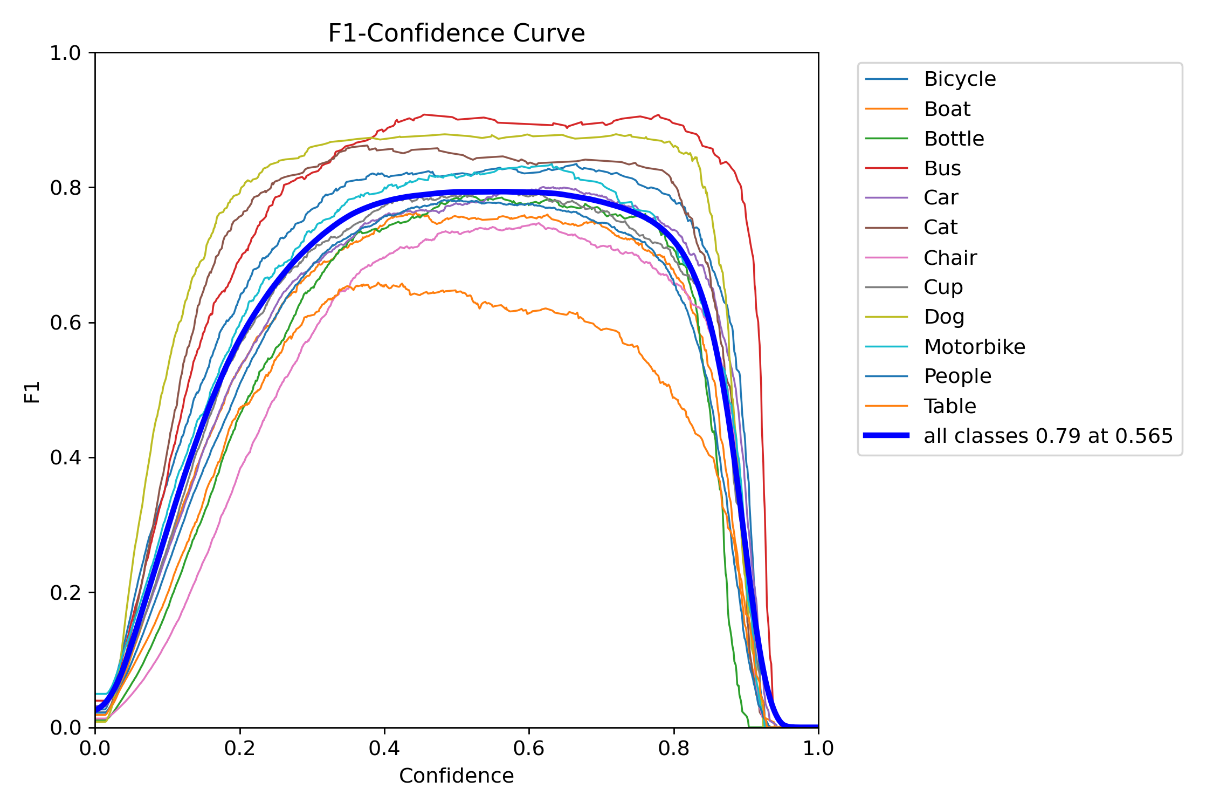
Dari sisi validasi, kurva-kurvanya cenderung menurun secara umum meski sempat mengalami fluktuasi pada beberpaa titik. Namun, mekanisme *early stopping* dengan *patience* 30 membantu menjaga model agar tidak *overfitting*. Peningkatan pada metrik mAP50(B) dan mAP50-95(B) di *validation set* juga mendukung pemahaman bahwa RTDETR-X dapat menangkap berbagai variasi objek dalam kondisi cahaya minim dengan rentang IoU yang luas.

Dibandingkan dengan model-model sebelumnya, RTDETR-X memiliki parameter dan kedalaman aristektur yang jauh lebih besar, sehingga mampu menyerap lebih banyak informasi visual. Meskipun memerlukan sumber daya komputasi yang lebih besar, hasil pelatihan yang konsisten dan peningkatan mAP mencerminkan daya generalisasi dan ketepatan deteksi yang lebih tinggi.

****

*Confusion Matrix Normalized* pada gambar {x} menunjukkan bahwa model dengan parameter jauh lebih besar mampu mengklasifikasikan mayoritas kelas dengan akurasi tinggi, terlihat dari nilai diagonal yang dominan di atas 0,8. Pada beberapa kelas seperti *Bus* dan *Car*, kemampuan deteksi di kondisi pencahyaan rendah tampak semakin baik. Meski demikian, lagi-lagi masih terdapat kebingungan kecil di antara kelas yang mirip, seperti *Bicycle* dan *Motorbike*, menandakan bahwa bentuk, sudut, atau kontur yang serupa dapat menimbulkan *misclassification.*

Dari sisi performa secara keseluruhan, RTDETR-X menunjukkan peningkatan yang substansial dibandingkan model-model sebelumnya, terutama dalam mengenali objek berukuran kecil relatif kecil atau objek yang cenderung tersamar di *background*.

****

Kurva *F1-Confidence* pada Gambar {x} memperlihatkan bahwa model RTDETR-X mampu mencapai keseimbangan *precision* dan *recall* yang cukup baik di rentang *confidence* menengah, terlihat kenaikan F1 yang stabil sebelum akhirnya menurun saat ambang *confidence* makin tinggi. Pada beberapa kelas, seperti *Car* dan *Bus*, kurva cenderung lebih tinggi dan lebih stabil dibanding kelas yang memiliki kemiripan ciri fisik seperti *Bicycle* dan *Motorbike* sehingga rentan terjadi kesalahan klasifikasi. Hal ini mengindikasikan bahwa kapasitas model yang lebih besar membantu pengenalan ciri objek yang jelas, tetapi tetap memiliki tantangan yang sama seperti model-model sebelumnya saat fitur objek berbeda tipis dalam pencahayaan gelap.

Dibandingkan dengan model berparameter lebih kecil, RTDETR-X menampilkan puncak F1 yang lebih tinggi dan area puncak yang cenderung lebih lebar, menegaskan kemampuan model untuk mepertahankan keseimbangan antara menghindari *false positives* dan tidak melewatkan objek. Kondisi *low-light* pada ExDark masih menjadi kendala di kelas-kelas tertentu, namun peningkatan F1 di berbagai rentang *confidence* menekankan bahwa RTDETR-X lebih tangguh dalam mengenali perbedaan tekstur dan kontras yang minimal.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** |  |  |
| **2.** |  |  |
| **3.** | **A screenshot of a computer  AI-generated content may be incorrect.** |  |

Tabel {x} di atas merupakan sampel hasil dari deteksi objek menggunakan model RTDETR-X. Terlihat bahwa model yang memiliki parameter terbesar ini memiliki *bounding box* yang lebih mendekati *ground truth*, terutama untuk objek yang kecil atau tumpang tindih dengan *confidence* yang lebih tinggi seperti pada gambar 3b. Tetapi, justru menciptakan kebingungan baru seperti pada gambar 2b yang cukup jauh berbeda dengan gambar 2a. Selebihnya, model ini juga dapat memilah atau mengambil suatu objek secara lebih detail seperti ditunjukkan pada objek-objek *People* di gambar 1b yang terlihat sangat terpilah dengan rapi.

**4.5 Implementasi YOLOv10**

**4.5.1 Pelatihan YOLOv10**

Model YOLOv10 yang digunakan terdiri dari tiga varian, yaitu YOLOv10s, YOLOv10m, dan YOLOv10b. Seperti halnya dua model sebelumnya, masing-masing varian YOLOv10 diuji dengan empat kombinasi *hyperparameter* melalui teknik *grid search*, menghasilkan total 12 model (3 varian × 4 kombinasi). Seluruh hasil pelatihan dari model YOLOv10 ini disajikan dalam Tabel {y3}.

| BATCH 8 | Lr 0.01 | | | Lr 0.001 | | |
| --- | --- | --- | --- | --- | --- | --- |
| s | m | b | s | m | b |
| total\_epoch | 109 | 174 | 150 | 117 | 188 | 54 |
| total\_time | 12322 | 23914.4 | 23594.6 | 13003.2 | 16472.6 | 8470.51 |
| train/box\_loss | 243.436 | 201.078 | 192.587 | 199.731 | 198.211 | 190.896 |
| train/cls\_loss | 198.149 | 148.982 | 141.913 | 144.895 | 147.546 | 134.995 |
| train/dfl\_loss | 240.514 | 231.333 | 228.432 | 21.366 | 229.442 | 223.828 |
| metrics/precision(B) | 0.79927 | 0.78046 | 0.82146 | 0.80476 | 0.80271 | 0.82051 |
| metrics/recall(B) | 0.67451 | 0.71035 | 0.67349 | 0.68341 | 0.69555 | 0.70915 |
| metrics/mAP50(B) | 0.75557 | 0.77337 | 0.77444 | 0.75887 | 0.77108 | 0.79314 |
| metrics/mAP50-95(B) | 0.48525 | 0.50709 | 0.51212 | 0.49627 | 0.50846 | 0.52548 |
| val/box\_loss | 280.277 | 278.851 | 279.928 | 280.489 | 278.668 | 270.202 |
| val/cls\_loss | 209.757 | 209.591 | 206.692 | 223.116 | 208.539 | 203.039 |
| val/dfl\_loss | 282.986 | 317.003 | 323.184 | 279.423 | 316.092 | 301.239 |

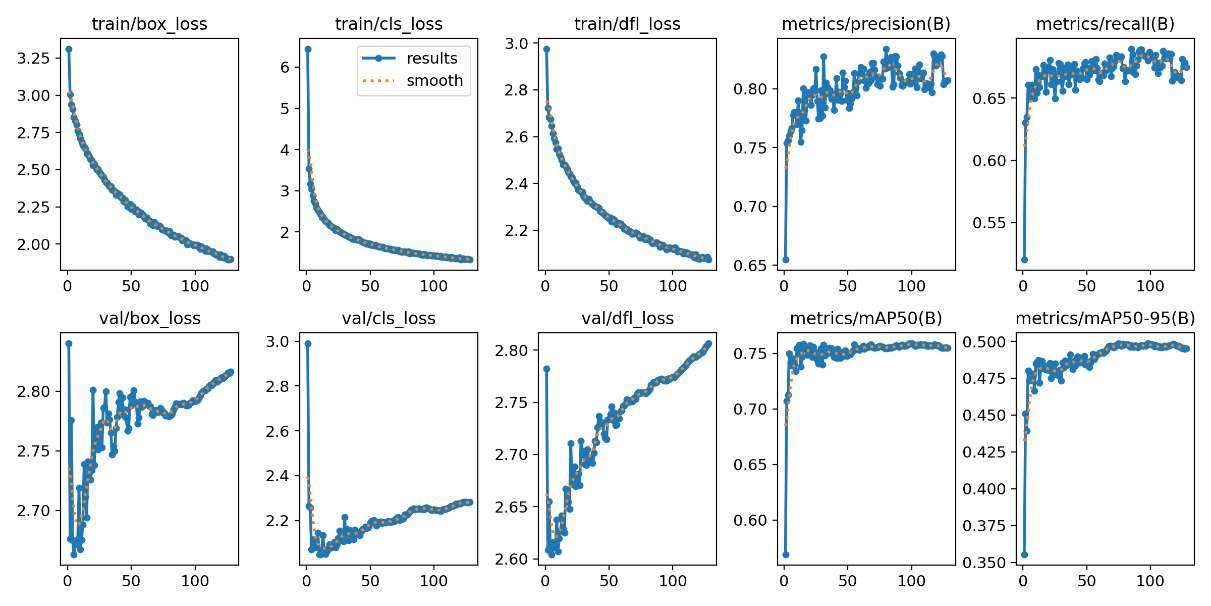
| BATCH 16 | Lr 0.01 | | | Lr 0.001 | | |
| --- | --- | --- | --- | --- | --- | --- |
| s | m | b | s | m | b |
| total\_epoch | 145 | 139 | 107 | 128 | 54 | 38 |
| total\_time | 10807.6 | 15082.1 | 14816.3 | 9511.46 | 5874.28 | 5244.12 |
| train/box\_loss | 226.507 | 200.573 | 198.476 | 189.915 | 196.896 | 199.209 |
| train/cls\_loss | 172.874 | 146.065 | 142.712 | 132.568 | 138.849 | 141.819 |
| train/dfl\_loss | 227.508 | 229.004 | 229.662 | 207.403 | 224.978 | 227.501 |
| metrics/precision(B) | 0.77576 | 0.81938 | 0.81252 | 0.80703 | 0.80623 | 0.80625 |
| metrics/recall(B) | 0.67603 | 0.67797 | 0.6814 | 0.67497 | 0.70078 | 0.69801 |
| metrics/mAP50(B) | 0.74654 | 0.7749 | 0.76618 | 0.75515 | 0.77668 | 0.78232 |
| metrics/mAP50-95(B) | 0.48057 | 0.50907 | 0.49944 | 0.49512 | 0.51339 | 0.52283 |
| val/box\_loss | 286.094 | 278.814 | 279.254 | 281.667 | 272.104 | 268.403 |
| val/cls\_loss | 218.469 | 209.035 | 213.922 | 228.229 | 21.165 | 202.321 |
| val/dfl\_loss | 285.825 | 313.941 | 31.927 | 280.646 | 300.242 | 295.285 |

**4.3.2 Kombinasi YOLOv10 terbaik**

Berdasarkan metrik yang tercantum pada Tabel {y3}, diperoleh kombinasi *hyperparameter* terbaik untuk masing-masing varian YOLOv10 yang akan diuraikan pada bagian berikut.

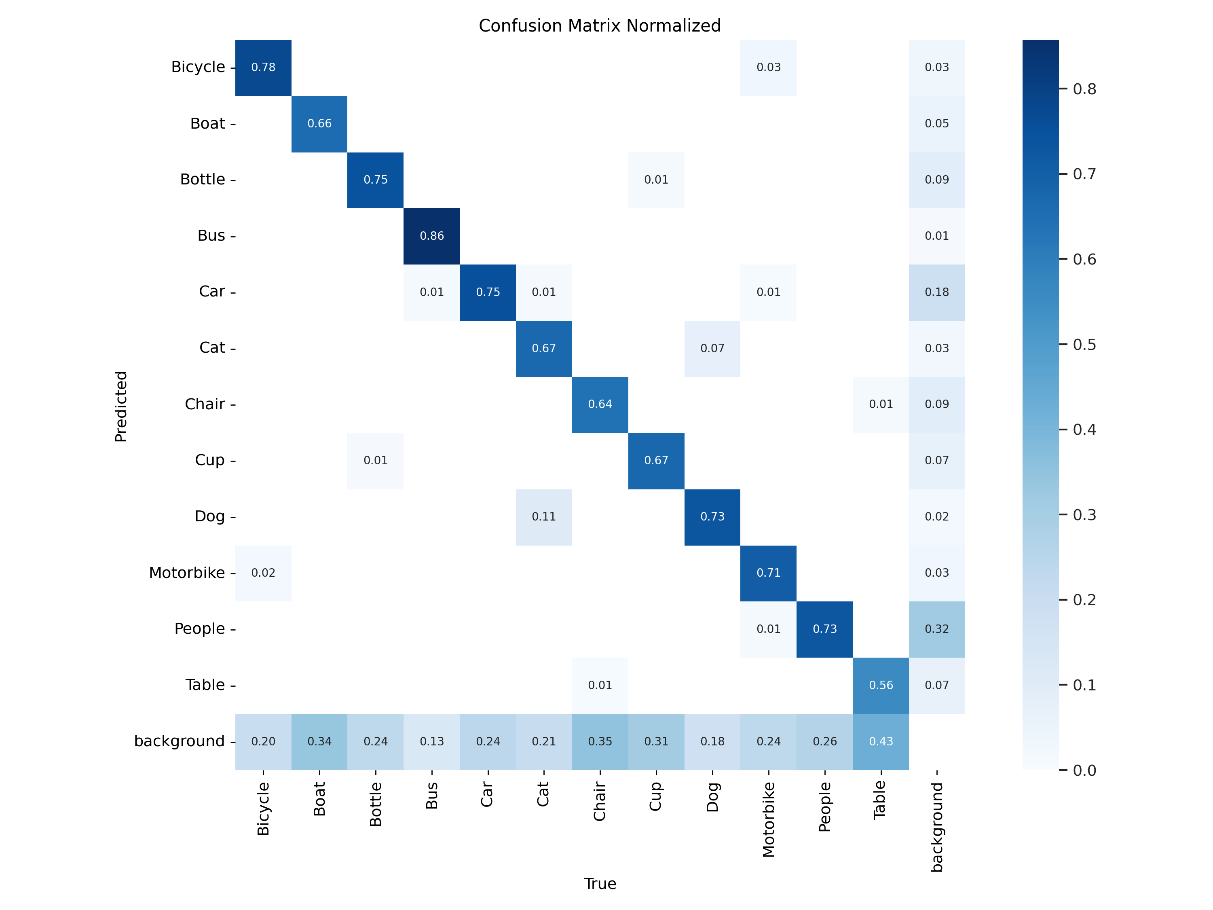
| Tipe Model | Batch Size | Learning Rate | mAP50 | map50-95 | val/cls\_loss |
| --- | --- | --- | --- | --- | --- |
| YOLOv10s | 16 | 0.001 | 0. 75515 | 0. 49512 | 228.229 |
| YOLOv10m | 16 | 0.001 | 0. 77668 | 0. 51339 | 0. 21.165 |
| YOLOv10b | 8 | 0.01 | 0. 77444 | 0. 51212 | 206.692 |

**4.3.2.1 YOLOv10s**



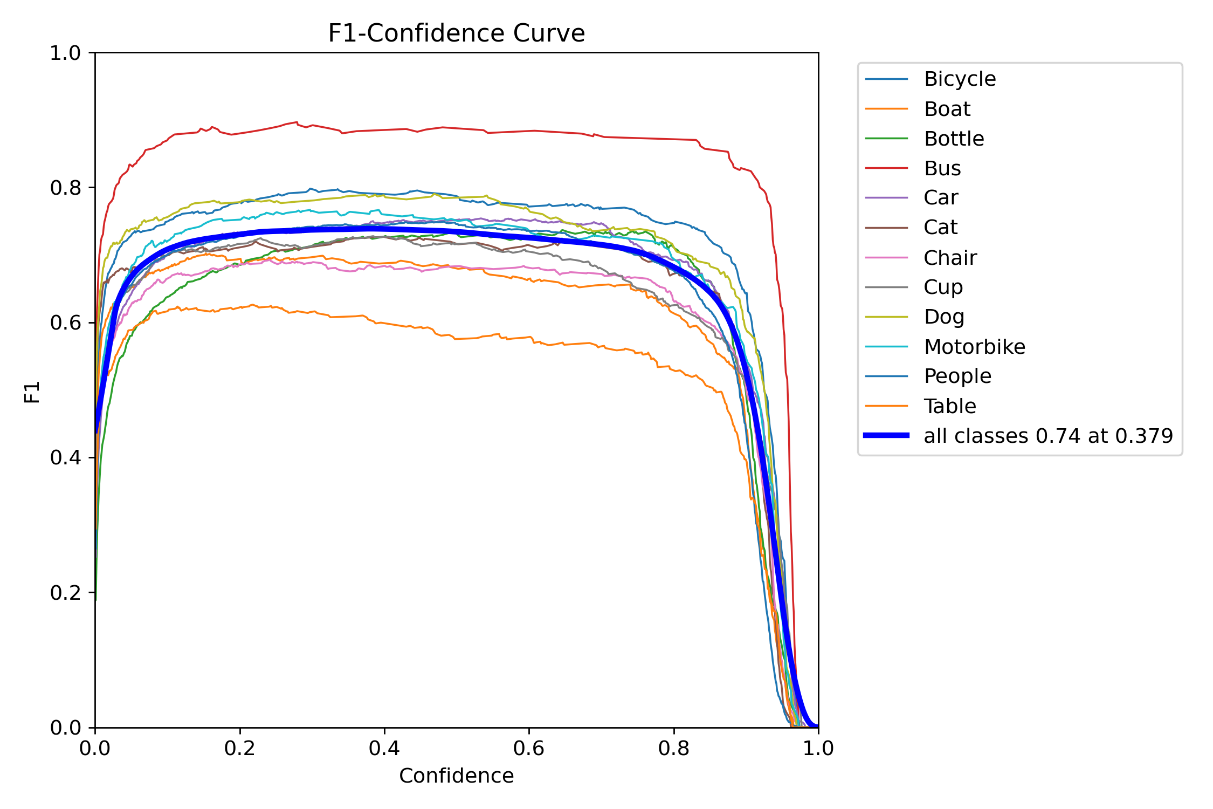
Pada gambar {x}, grafik hasil pelatihan YOLOv10s, ketiga loss utama di bagian *training* tampak menrun secara konsisten dari nilai tinggi di awal menuju tingkat yang semakin stabil. Peningkatan metrik *precision(B)* dan *recall(B)* mengindikasikan bahwa model semakin baik dalam mengurangi *false positives* serta jarang melewatkan objek yang seharusnya terdeteksi.

Pada *validation set* cenderung mengalami fluktuasi, namun masih membentuk tren penurunan hingga titik tertentu. Meskipun sempat naik di beberapa titik, mekanisme *early stopping* dengan *patience 30* membantu menjaga model dari *overfitting.* Grafik mAP50(B) dan mAP50-95(B) memperlihatkan kenaikan kinerja yang signifikan di fase awal, kemudian tetap stabil dan seidkit meningkat di akhir pelatihan.



*Confusion Matrix Normalized* untuk YOLOv10s pada Gambar {x} menunjukkan bahwa model ini secara umum berhasil mengklasifikasikan objek dengan akurasi yang cukup baik, terbukti dari nilai diagonal yang cukup tinggi. Meski demikian, terlihat pula adanya kekeliruan pada kelas tertentu yang berukuran kecil seperti *Cup* yang kadang terdeteksi sebagai *Bottle*, dan lagi-lagi *Bicycle* dan *Motorbike* yang sesekali tertukar.

Secara keseluruhan, meskipun beberapa objek masih menimbulkan kebingungan dengan *backgorund*, hasil ini menandakan bahwa tiper terkecil dari YOLOv10 yaitu YOLOv10s sudah cukup baik untuk mendeteksi objek untuk citra berpencahayaan rendah.



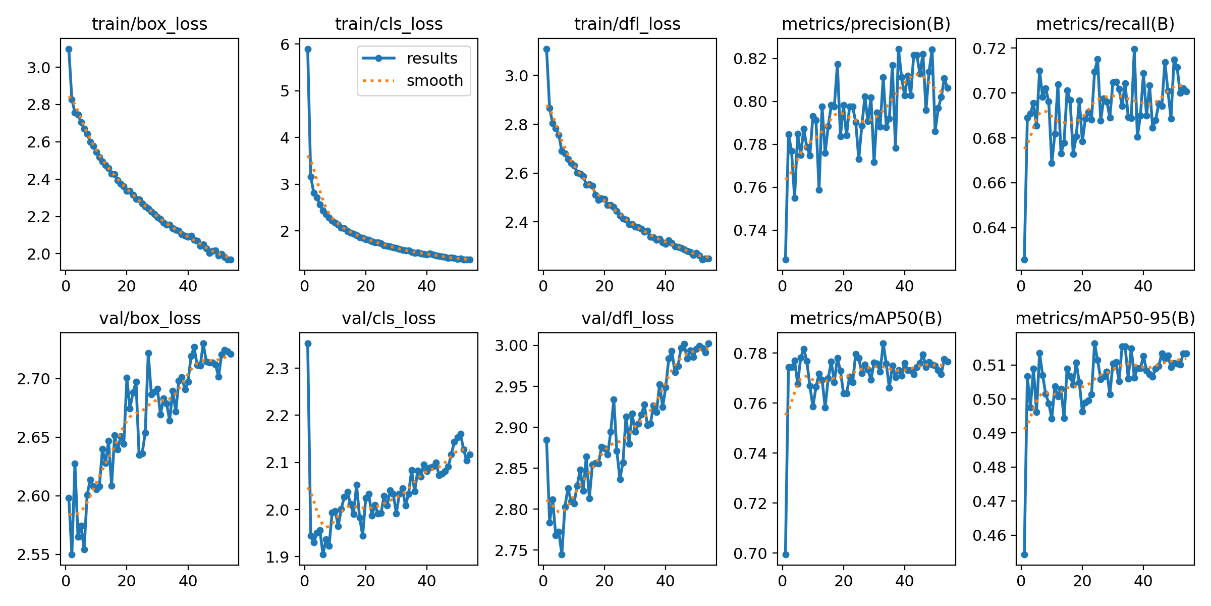
Gambar {x} menunjukkan *F1-Confidence Curve* untuk YOLOv10s yang membuktikan bahwa mayoritas kelas memiliki puncak F1 yang cukup tinggi di rentang *threshold* menengah, kemudian menurun drastis ketika *threshold* diperketat. Hal ini menandakan bahwa meski model semakin yakin terhadap prediksinya, YOLOv10s juga lebih rentan kehilangan beberapa deteksi. Pada beberapa kelas dengan ciri fisik yang jelas seperti *Bus* atau *Car*, puncak F1 terlihat lebih tinggi dan stabil dibanding kelas yang memiliki kemiripan ciri seperti *Bicycle* dan *Motorbike*.

Secara keseluruhan, capaian F1 di angka 0,84 (pada *confidence 0,379*), memberikan indikasi bahwa model telah menemukan *sweet spot* untuk menyeimbangkan *false positives* dan *false negatives*. Namun, adanya variasi kurva antarkelas menunjukkan bahwa pencahayaan minim dan fitur objek yang serupa masih dapat menimbulkan kesalahan klasifikasi. Meskipun demikian, bentuk kurva yang relatif mulus dan puncak yang cukup stabil pada beberapa kelas menegaskan kemampuan YOLOv10s sebagai parameter terkecil dari YOLOv10 lainnya, mampu untuk mempertahankan kinerja yang baik.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** |  |  |
| **2.** |  |  |
| **3.** |  |  |

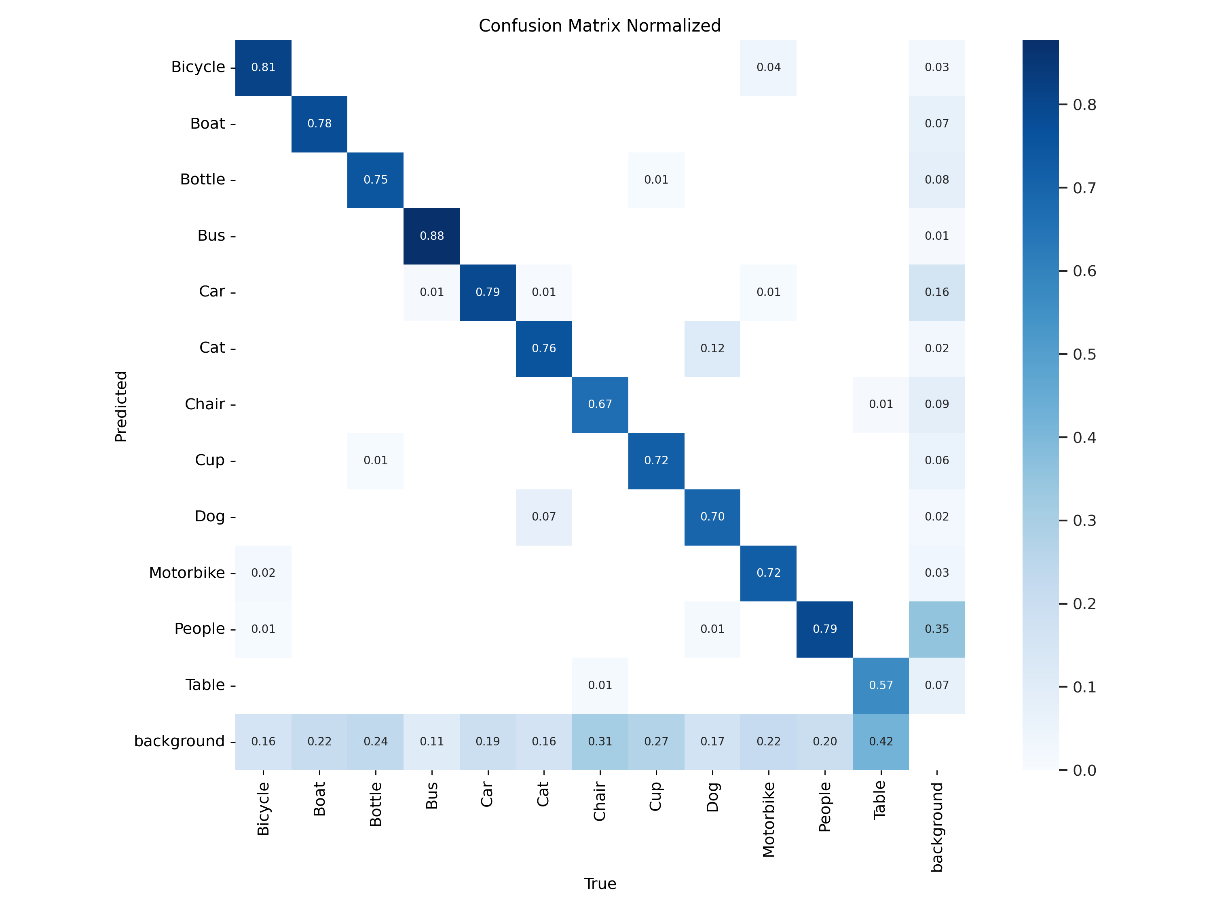
Tabel {x} di atas adalah sampel hasil dari deteksi oleh model YOLOv10s yang memiliki parameter terkecil diantara YOLOv10 lainnya. Tabel di atas menunjukkan kemampuan deteksi yang cukup baik secara *bounding box* walau masih tedapat sedikit ketidakaturan pada kelas-kelas yang saling tumpang tindih atau objek kecil. Namun uniknya, model ini mampu mendeteksi objek yang tidak terdaftar dalam *ground truth* tetapi terlihat secara kasat mata secara cukup meyakinkan seperti pada objek *Car* di gambar 2b dan *Table* di gambar 3b.

**4.3.2.2 YOLOv10m**



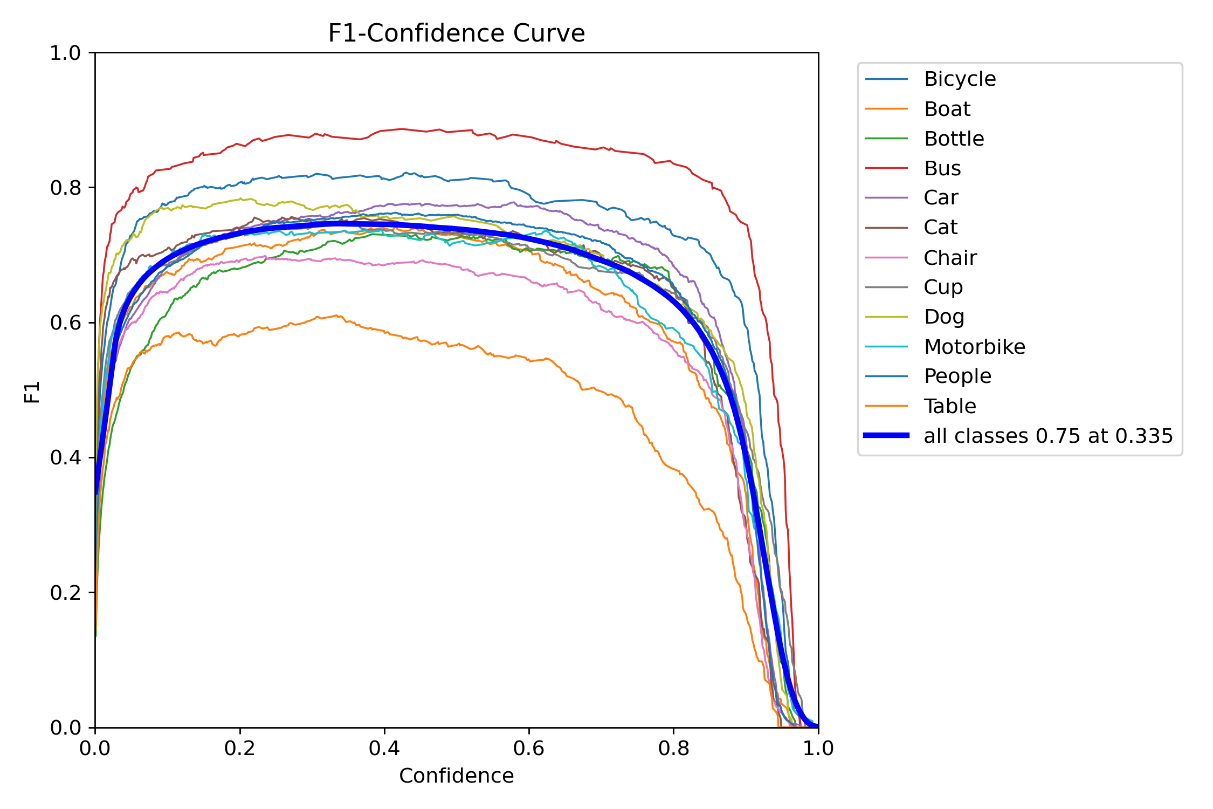
Kurva-kurva untuk model YOLOv10m pada gambar {x} di atas menunjukkan bahwa selama pelatihan, ketiga *loss* utama di fase *training* menurun secara konsisten hingga mencapai titik yang makin stabil di akhir. Di saat yang sama, metrik *precision(B)* dan *recall(B)* menunjukkan tren peningkatan, menandakan kemampuan model yang kian baik dalam menempatkan *bounding box* dengan akurasi tinggi dan semakin jarang melewatkan objek.

Pada bagian validasi, kurva *val/box\_loss, val/cls\_loss,* dan *val/dfl\_loss* memperlihatkan pola yang cenderung naik-turun, namun secara umum masih membentuk tren penurunan di beberapa segmen, sebelum akhirnya stagnan atau fluktuatif. Meski menandakan kecenderungan *overfitting* di beberapa titik yang ditunjukkan oleh *val loss* yang kadang naik, penerapan mekanisme *early stopping* membantu mencegah peurunan yang lebih parah. Hal ini didukung oleh kenaikan metrik mAP50(B) dan mAP50-95(B) yang menggambarkan kemampuan deteksi model pada berbagai rentang IoU.

****

Gambar {x} di atas adalah *confusion matrix* yang sudah dinormalisasi untuk YOLOv10m, yang menunjukkan bahwa model secara umum memiliki tingkat akurasi klasifikasi yang cukup tinggi, terlihat dari nilai diagonal yang dominan pada beberapa kelas seperti *Bus* dan *Car*. Namun, lagi-lagi masih saja terdapat kekeliruan pada kelas dengan bentuk yang mirip seperti *Bicycle* dan *Motorbike*, serta objek-objek berukuran kecil yang cenderung sulit diidentifikasi pada kondisi cahaya rendah. Beberapa *misclassifications* juga muncul antara *People* dan *background*, menandakan bahwa model belum sepenuhnya mampu membedakan objek berkontras rendah dari latar belakang.

Secara keseluruhan, hasil ini menekankan bahwa meski YOLOv10m dapat mengenali mayoritas kelas dengan baik, kondisi pencahayaan minim masih menjadi tantangan yang memicu kebingungan di antara kelas tertentu.

****

Kurva *F1-Confidence* pada gambar {x} di atas memperlihatkan bahwa YOLOv10m mencapai puncak keseimbangan antara *precision* dan *recall* di rentang *confidence* menengah. Nilai F1 tertinggi di kisaran 0,75 pada *confidence* sekitar 0,335 menggambarkan bahwa di titik tersebut, model dapat meminimalkan kesalahan deteksi, baik *false positives* maupun *false negatives*, dengan mempertahankan tingkat keyakinan yang memadai terhadap prediksinya.

Di beberapa kelas, terutama yang cirinya lebih kontras seperti *Car* atau *Bus*, kurva terlihat lebih stabil dan mencapai puncak yang lebih tinggi, menunjukkan model lebih mudah mengenali objek yang jelas meski dalam kondisi cahaya rendah. Namun pada kelas dengan detail serupa atau berukuran kecil, seperti *Bottle* dan *Cup*, lagi-lagi kurva cenderung kurang stabil dan puncaknya lebih rendah, menandakan tantangan dalam membedakan ciri-ciri visual yang samar.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** | **A screenshot of a video  AI-generated content may be incorrect.** |  |
| **2.** | **A group of people under a red umbrella  AI-generated content may be incorrect.** |  |
| **3.** | **A group of bottles with shadows  AI-generated content may be incorrect.** |  |

Sampel perbandingan *ground truth* dan *predicted* pada tabel {x} di atas menunjukkan bahwa model kemampuan deteksi YOLOv10m semakin mendekati *ground truth*. Perbedaan paling terlihat adalah pada gambar 3b yang memiliki tingkat kemiripan dan *confidence* yang sangat tinggi. Terlebih lagi, objek *Table* yang sebelumnya muncul di gambar 3b, tidak lagi muncul pada YOLOv10m ini. Tapi tetap saja, pada gambar 2b masih terdeteksi *Car* yang sebenarnya tidak terdaftar pada *grund truth label*. Secara umum model dengan parameter yang lebih besar daripada YOLOv10s ini menunjukkan perkembangan yang lebih baik dari segi *bounding box* pada objek-objek yang saling tumpang tindih.

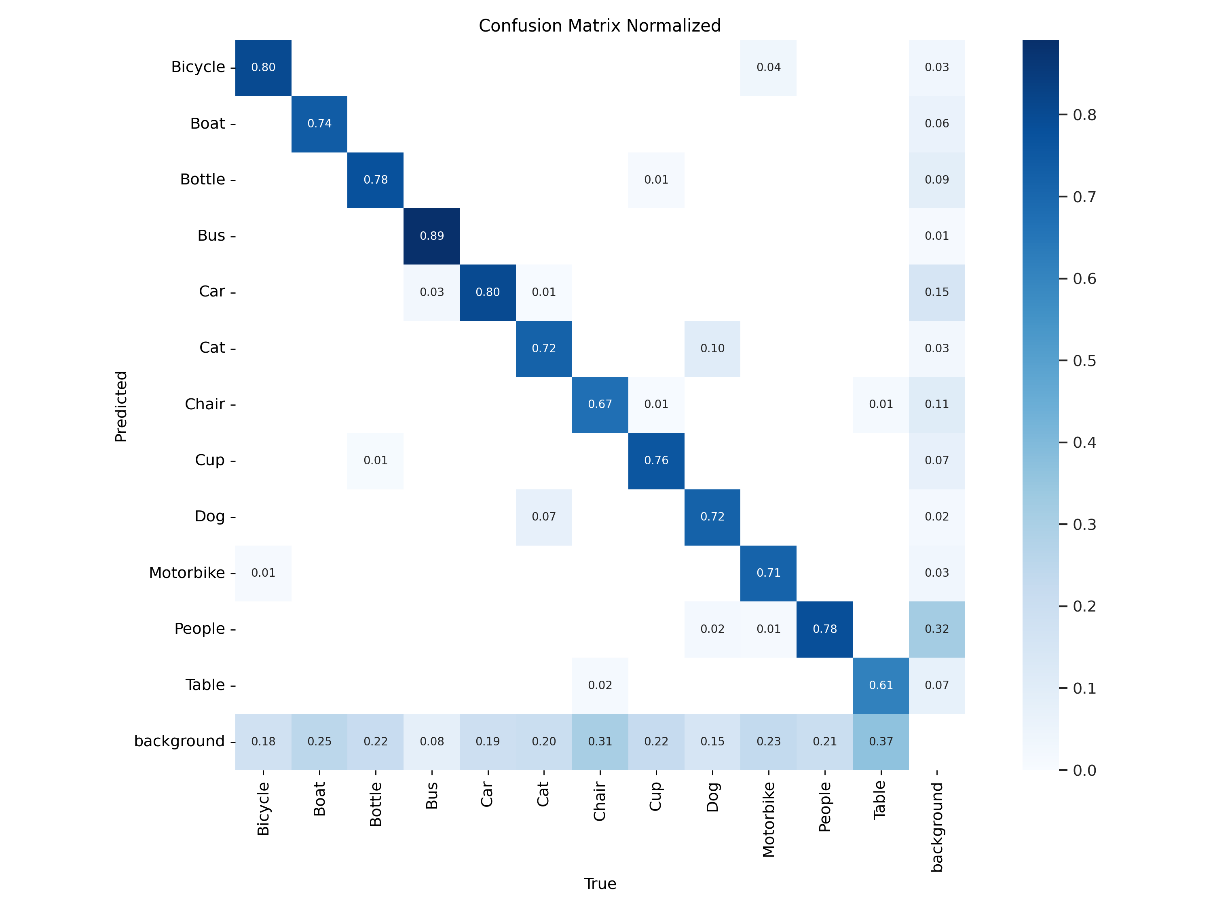
**4.3.2.3 YOLOv10b**



Kurva-kurva dari Gambar {x} di atas menunjukkan pola konvergensi yang baik. Pada grafik *loss training* (*train/box\_loss, train/cls\_loss, train/dfl\_loss*), terlihat penurunan nilai loss yang signifikan pada awal training dan kemudian stabil pada nilai yang lebih rendah, menandakan proses pembelajaran yang efektif. Perbandingan antara kurva hasil aktual dan kurva *smoothing* menunjukkan stabilitas proses training tanpa fluktuasi yang berlebihan.

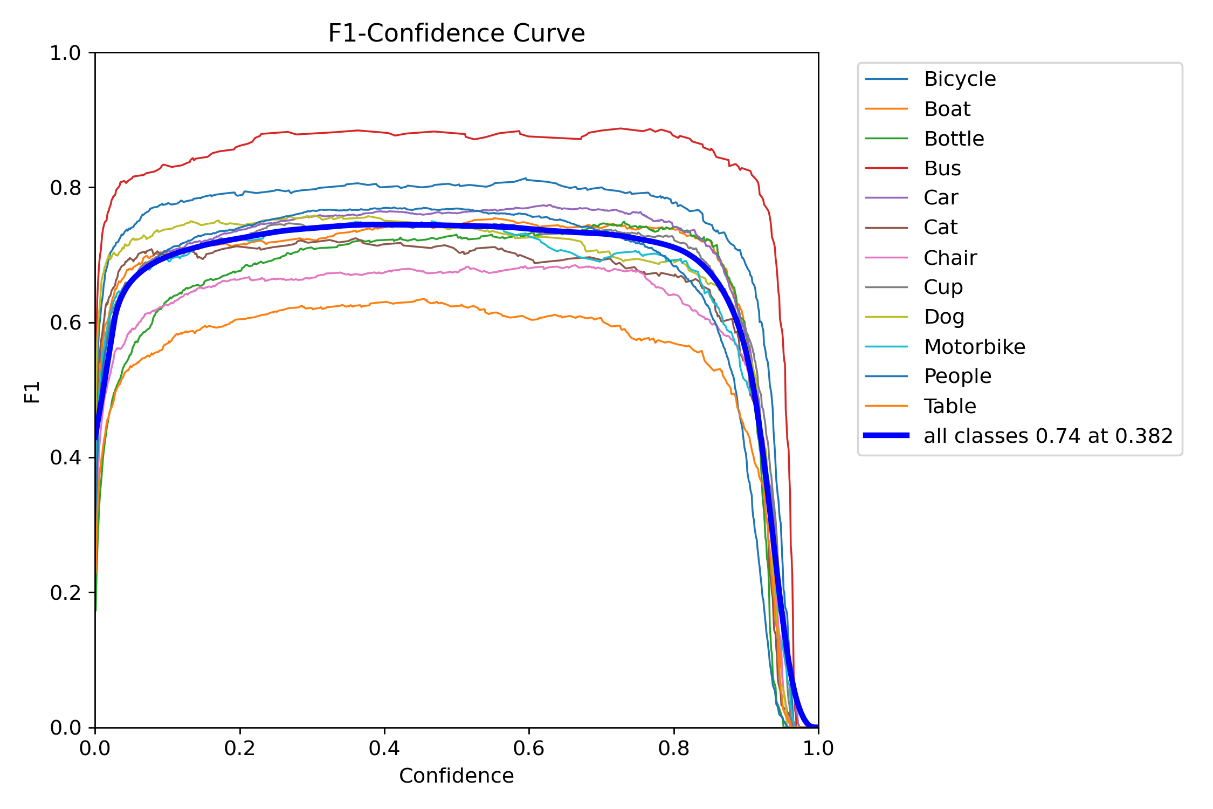
Pada metrik validasi, *val/box\_loss dan val/cls\_loss* menunjukkan pola penurunan yang serupa dengan metrik *training*, mengindikasikan bahwa model tidak mengalami *overfitting* yang signifikan. Namun, terdapat sedikit peningkatan pada *val/dfl\_loss* setelah epoch ke-100, yang menunjukkan potensi keterbatasan model dalam memprediksi distribusi lokasi objek pada dataset validasi. Metrik performa model menunjukkan hasil yang menjanjikan, dengan nilai *precision* mencapai sekitar 0,8 dan *recall* sekitar 0,7 yang mengindikasikan keseimbangan yang baik antara *true positive* dan *false positive* dalam deteksi objek pada kondisi pencahayaan rendah.

Metrik mAP50(B) yang mencapai nilai sekitar 0,75 menunjukkan performa deteksi yang baik pada threshold IoU 0,5 , sedangkan mAP50-95(B) yang mencapai sekitar 0,5 menggambarkan kemampuan model dalam mendeteksi objek dengan presisi lokasi yang bervariasi.

****

Gambar {x} menunjukkan *confusion matrix* yang sudah dinormalisasi untuk YOLOv10b, yang membuktikan dominasi nilai diagonal tingi pada hampir semua kelas, misalnya *Bus* dan *Car* yang berada di atas 0,8. Hal ini mencerminkan keunggulan arsitektur YOLOv10b yang memiliki parameter terbesar di kalangan YOLOv10, dalam mengekstraksi fitur lebih detail. Meski demikian, masih terlihat kekeliruan pada beberapa kelas yang bentuknya mirip, seperti *Bicycle* dan *Motorbike*, menandakan tantangannya masih saja sama seperti model-model sebelumnya.

Secara keseluruhan, hasil ini mengindikasikan bahwa YOLOv10b dapat memberikan performa deteksi yang baik meskipun berada di lingkungan pencahayaan rendah. Hal ini dapat dilihat dari kemampuan model dalam membedakan objek yang memiliki kontras lebih tegas serta menekan kesalahan prediksi antara objek dan latar belakang.

****

Gambar {x} di atas menunjukkan kurva *F1-Confidence* untuk YOLOv10b yang menampilkan peningkatan keseimbangan *precision* dan *recall* yang lebih konsisten di rentang *confidence* menengah dibandingkan model serupa dengan parameter lebih kecil. Mayoritas kurva kelas mencapai puncak F1 pada *confidence* sekitar 0,3-0,4 , mencerminkan bahwa model telah mempelajari ciri-ciri objek secara mendalam sehingga mampu mengurangi *false positives* sekaligus mempertahankan *recall*. Kelas-kelas dengan kontur jelas seperti *Car* atau *Bus* cenderung memperlihatkan puncak F1 yang lebih stabil dan lebih tinggi dibanding kelas yang cirinya saling menyerupai seperti *Bicycle* *vs Motorbike*.

Meskipun demikian, beberapa kelas menurun cukup drastis ketika *confidence* lebih tinggi dari 0,7 , menandakan bahwa model memang semakin selektif dalam menyatakan objek tetapi dengan risiko melewatkan target yang lebih samar (menurunkan *recall*). Secara keseluruhan, hasil tersebut menggarisbawahi bahwa kapasitas besar YOLOv10b mampu mempelajari beragam pola visual pada kondisi pencahayaan rendah, namun pemilihan *confidence* *threshold* tetap menjadi faktor penting guna memperoleh kinerja deteksi terbaik untuk tiap kelas.

| **No.** | ***Ground Truth* (a)** | ***Predicted* (b)** |
| --- | --- | --- |
| **1.** | **A screenshot of a video  AI-generated content may be incorrect.** |  |
| **2.** | **A group of people under a red umbrella  AI-generated content may be incorrect.** |  |
| **3.** | **A group of bottles with shadows  AI-generated content may be incorrect.** |  |

Tabel {x} di atas menunjukkan perbandingan dari sampel *ground truth* dan *predicted* untuk model YOLOv10b yang tampak tidak terlalu berubah secara signifikan dibandingkan model YOLOv10m sebelumnya. Dari sampel di atas, terlihat bahwa yang dapat langsung dikenali adalah nilai *confidence* yang lebih tinggi dan skala serta posisi *bounding box* yang semakin akurat. Walau demikian model masih saja kesulitan untuk mengenali objek tertentu yang terlihat mirip ataupun dekat seperti objek *Chair* pada gambar 2b.

**4.6 *Testing Model***

Dataset yang digunakan untuk *testing* adalah data berupa video yang sudah diambil satu *frame* untuk setiap detiknya serta sudah memiliki ke-12 objek yang dapat diterima oleh model. Namun, sebelum memulai tahap terakhir ini, pemberian *night filter* sangat diperlukan untuk mengukur sejauh mana kinerja model karena data *test* ini kebanyakan diambil pada siang hari. Hasil dari pengaplikasian *night filter* ini adalah sebagai gambar {x} di bawah.



1. (b)

Gambar {x}(a) adalah *sample frame* dari video yang ada pada *test set*. Dari citra yang nampak terang tersebut, diberlakukan *night filter* dengan bantuan modul *opencv-python*. Langkah pertama adalah mengurangi kecerahan gambar secara keseluruhan sebesar 70%, kemudian dilanjut dengan langkah kedua yaitu menambahkan *blue tint* dengan cara menggabungkan citra gelap dengan modifikasi di *channel* biru ditambah nilai sebesar 50. Hasil penggabungan ini menggunakan bobot 0,8 untuk gambar asli yang sudah digelapkan dan 0,2 untuk gambar yang ditambahkan *blue tint*, menciptakan efek malam yang cukup realistis seperti pada gambar {x}(b).

**4.6.1 YOLOv9s**

**4.6.2 YOLOv9m**

**4.6.3 YOLOv9c**

**4.6.4 RTDETR-L**

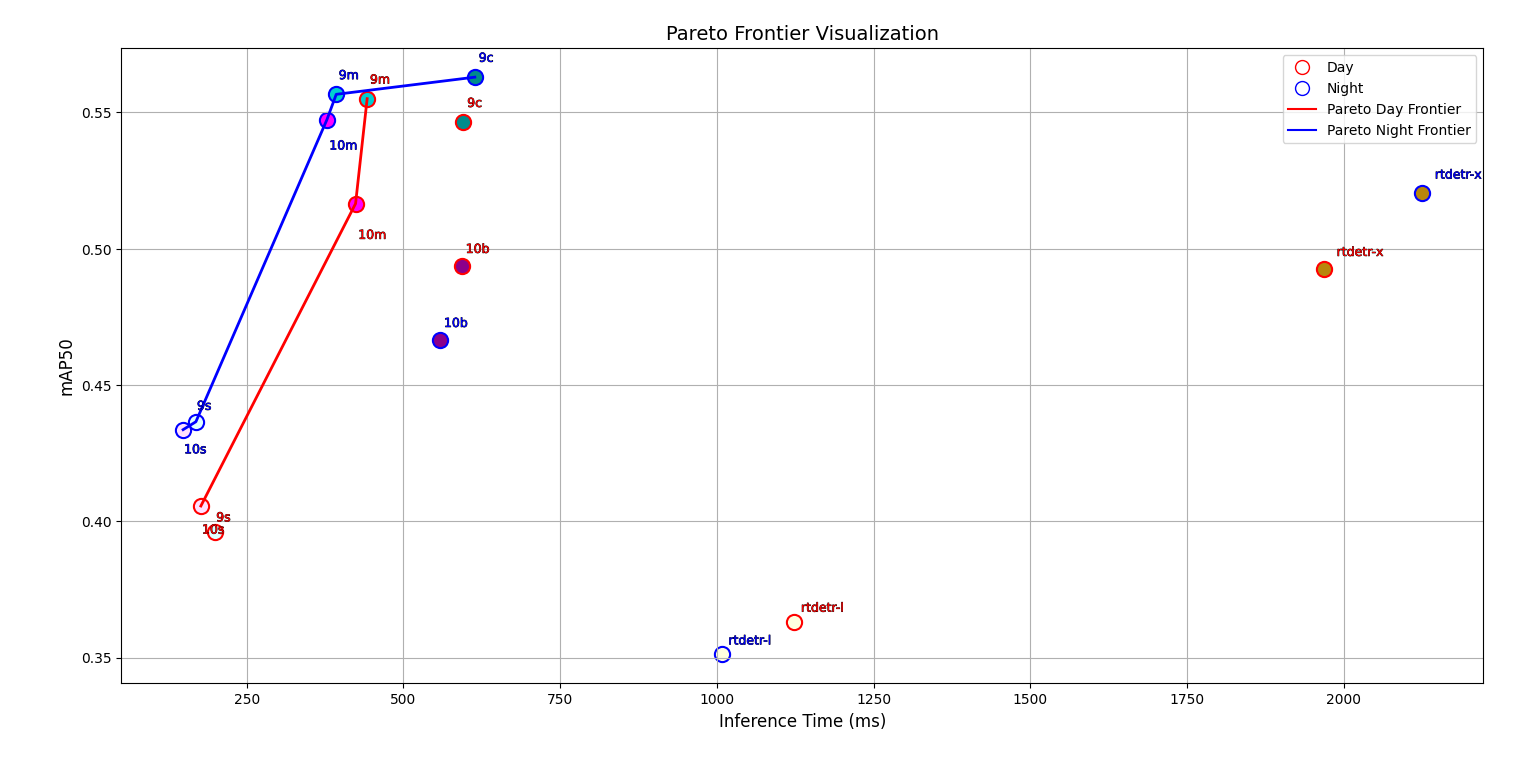
**4.6.5 RTDETR-X**

**4.6.6 YOLOv10s**

**4.6.7 YOLOv10m**

**4.6.8 YOLOv10b**

**4.7 *Pareto Frontier Analysis***

 Setelah seluruh model selesai melalui tahap pelatihan dan evaluasi, metrik utama yang digunakan untuk menilai performa model adalah *mean Average Precision* (mAP) dan *inference time*. Visualisasi hasil evaluasi terhadap kedua metrik ini ditunjukkan dalam bentuk kurva *Pareto Frontier* berikut:

**4.8 Diskusi**

# **BAB V KESIMPULAN DAN SARAN**

**5.1 Kesimpulan**

**5.2 Saran**

# **DAFTAR PUSTAKA**

Alif, M. A. R., & Hussain, M. (2024). YOLOv1 to YOLOv10: *A comprehensive review of YOLO variants and their application in the agricultural domain*. *arXiv preprint arXiv:2406.10139*.

Benmeziane, H., Niar, S., Ouarnoughi, H., & El Maghraoui, K. (2022, May). *Pareto* *rank surrogate model for hardware-aware neural architecture search.* In 2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 267-276). IEEE.

Bergstra, J., & Bengio, Y. (2012). *Random search for hyper-parameter optimization.* Journal of machine learning research, 13(2).

Cao, X., Zhang, L., & Liu, Y. (2023). *Information Flow-Based Fuzzy Cognitive Maps with Enhanced Interpretability*. Springer.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, August). *End-to-end object detection with transformers. In European conference on computer vision* (pp. 213-229). Cham: Springer International Publishing.

Chen, C., Chen, Q., Xu, J., & Koltun, V. (2018). *Learning to see in the dark*. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3291-3300).

Chen, W., & Shah, T. (2021). *Exploring low-light object detection techniques*. *arXiv preprint arXiv:2107.14382*.

Chicco, D., & Jurman, G. (2020). *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC genomics, 21, 1-13.

Cui, Z., Li, K., Gu, L., Su, S., Gao, P., Jiang, Z., ... & Harada, T. (2022*). You only need 90k parameters to adapt light: a light weight transformer for image enhancement and exposure correction*. *arXiv preprint arXiv:2205.14871*.

Cui, Z., Qi, G. J., Gu, L., You, S., Zhang, Z., & Harada, T. (2021*). Multitask aet with orthogonal tangent regularity for dark object detection*. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2553-2562).

Deb, K. (2001). *Multiobjective Optimization Using Evolutionary Algorithms*. Wiley, New York.

Dhillon, A., & Verma, G. K. (2019). *Convolutional neural network: a review of models, methodologies and applications to object detection*. Progress in Artificial Intelligence, *9*(2), 85–112. https://doi.org/10.1007/s13748-019-00203-0.

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022). *Object detection using YOLO: challenges, architectural successors, datasets and applications*. Multimedia Tools and Applications, *82*(6), 9243–9275. <https://doi.org/10.1007/s11042-022-13644-y>.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). *The pascal visual object classes (voc) challenge*. International journal of computer vision, 8*8*, 303-338.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. In *MIT Press eBooks*. https://dl.acm.org/citation.cfm?id=3086952.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). *On large-batch training for deep learning: Generalization gap and sharp minima*. arXiv preprint arXiv:1609.04836.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). *Backpropagation applied to handwritten zip code recognition*. Neural computation, *1*(4), 541-551.

Lee, J., Bang, J., & Yang, S. (2017). *Object detection with sliding window in images including multiple similar object*s. 2017 International Conference on Information and Communication Technology Convergence *(*ICTC), Pp. 803–806. <https://doi.org/10.1109/ictc.2017.8190786>.

Li, J., Feng, Y., Shao, Y., & Liu, F. (2024). IDP-YOLOV9: *Improvement of Object Detection Model in Severe Weather Scenarios from Drone Perspective*. Applied Sciences, *14*(12), 5277.

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., & Yang, J. (2020). *Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection*. Advances in Neural Information Processing Systems, *33*, 21002-21012.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). *A survey of convolutional neural networks: analysis, applications, and prospects*. IEEE transactions on neural networks and learning systems, 33(12), 6999-7019.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal loss for dense object detection.* In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

Loh, Y. P., & Chan, C. S. (2019*). Getting to know low-light images with the exclusively dark dataset*. Computer Vision and Image Understanding, *178*, 30-42.

Manongga, W. E., Chen, R. C., & Jiang, X. *Enhancing road marking sign detection in low-light conditions with YOLOv7 and contrast enhancement techniques*.

Masters, D., & Luschi, C. (2018). *Revisiting small batch training for deep neural networks*. arXiv preprint arXiv:1804.07612.

Peng, X., Li, J., & Wang, Y. (2023). *Data-driven machine learning approach for predicting dwell fatigue.* Scripta Materialia.

Powers, D. M. (2020). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. arXiv preprint arXiv:2010.16061.

Putri, G. R. C. K. B. (2021). *Peran Kamera Pengawas Closed-Circuit Television (CCTV) dalam Kontra Terorisme*. *Jurnal Lemhannas RI*, *9*(4), 100–116. https://doi.org/10.55960/jlri.v9i4.418

Prechelt, L. (2002). *Early stopping-but when?. In Neural Networks: Tricks of the trade* (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg.

Redmon, J., & Farhadi, A. (2018). YOLOV3*: an incremental improvement*. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1804.02767.

Redmon, J., Santosh, D. H. H., Ross, G., & Farhadi, A. (2015). *You only look once: Unified, Real-Time Object Detection.* arXiv (Cornell University*)*. <https://doi.org/10.48550/arxiv.1506.02640>.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). *Generalized intersection over union: A metric and a loss for bounding box regression*. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 658-666).

Rouhbakhshmeghrazi, A., & Alizadeh, G. *Instance Segmentation of Messier Objects: YOLO vs. Mask R-CNN*.

Ruby, U., & Yendapalli, V. (2020). *Binary cross entropy with deep learning technique for image classification*. *Int. J. Adv. Trends Comput. Sci. Eng*, *9*(10).

Saito, T., & Rehmsmeier, M. (2015). *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets.* PloS one, *10*(3), e0118432.

Shorten, C., & Khoshgoftaar, T. M. (2019). *A survey on Image Data Augmentation for Deep Learning*. Journal of Big Data*, 6*(1), 60.

Smith, L. N. (2017, March). *Cyclical learning rates for training neural networks.* In 2017 IEEE winter conference on applications of computer vision (WACV) (pp. 464-472). IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is All you Need*. arXiv (Cornell University), 30, 5998–6008. https://arxiv.org/pdf/1706.03762v5.

Voulodimos, A., Doulamis, A., & Protopapadakis, E. (2018). *Deep Learning for Computer Vision: A Brief review.* Computational Intelligence and Neuroscience, 2018, 1–13. <https://doi.org/10.1155/2018/7068349>.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). Yolov10: *Real-time end-to-end object detection*. arXiv preprint arXiv:2405.14458..

Wang, C. Y., & Liao, H. Y. M. (2024). YOLOv1 to YOLOv10: *The fastest and most accurate real-time object detection systems.* arXiv preprint arXiv:2408.09332.

Wang, C. Y., Yeh, I. H., & Liao, H. Y. M. (2024). *Yolov9: Learning what you want to learn using programmable gradient information*. arXiv preprint arXiv:2402.13616.

Wiley, V., & Lucas, T. W. (2018). *Computer Vision and Image Processing: A paper review*. International Journal of Artificial Intelligence Research, 2(1), 22. <https://doi.org/10.29099/ijair.v2i1.42>.

Yu, J., Jiang, Y., Wang, Z., Cao, Z., & Huang, T. (2016, October). *Unitbox: An advanced object detection network.* In Proceedings of the 24th ACM international conference on Multimedia (pp. 516-520).

Yaghini, M., Liu, P., Boenisch, F., & Papernot, N. (2023). *Learning with impartiality to walk on the pareto frontier of fairness, privacy, and utility*. arXiv preprint arXiv:2302.09183.

Ye, R., Chen, L., Liao, W., Zhang, J., & Ishibuchi, H. (2024). *Data-Driven Preference Sampling for Pareto Front Learning.* arXiv.

Yin, X., Yu, Z., Fei, Z., Lv, W., & Gao, X. (2023, September). *Pe-yolo: Pyramid enhancement network for dark object detection*. In *International Conference on Artificial Neural Networks* (pp. 163-174). Cham: Springer Nature Switzerland.

Yusuf, M. O., Hanzla, M., Al Mudawi, N., Sadiq, T., Alabdullah, B., Rahman, H., & Algarni, A. (2024). *Target Detection and Classification via EfficientDet and CNN over Unmanned Aerial Vehicles*. Frontiers in Neurorobotics, 18, 1448538.

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2024). *Detrs beat yolos on real-time object detection.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 16965-16974).

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). *Object detection with deep learning: A review*. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.

Zheng, S., & Gupta, G. (2022). *Semantic-guided zero-shot learning for low-light image/video enhancement*. In *Proceedings of the IEEE/CVF Winter conference on applications of computer vision* (pp. 581-590).