ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – SM234801

# PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO

**ADITYA DWI GUMARA**
NRP 5002201070

Dosen Pembimbing
**Dr. Dwi Ratna Sulistyaningrum, S. Si, MT**
NIP 19690405 199403 2 003

**Program Studi Sarjana**
Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2026

**TUGAS AKHIR – SM234801**

# PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO

**ADITYA DWI GUMARA**
NRP 5002201070

Dosen Pembimbing
**Dr. Dwi Ratna Sulistyaningrum, S. Si, MT**
NIP 19690405 199403 2 003

**Program Studi Sarjana**
Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2026

**FINAL PROJECT – SM234801**

# COMPARISON OF FASTER R-CNN AND YOLOv8 FOR FIRE AND SMOKE DETECTION ON VIDEO DATA

**ADITYA DWI GUMARA**
NRP 5002201070

Supervisor
**Dr. Dwi Ratna Sulistyaningrum, S. Si, MT**
NIP 19690405 199403 2 003

**Bachelor Program**
Department of Mathematics
Faculty of Science and Data Analytics
Sepuluh Nopember Institute of Technology
Surabaya
2026

# LEMBAR PENGESAHAN

# LEMBAR PENGESAHAN

## PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO

**TUGAS AKHIR**

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Matematika pada
Program Studi S-1 Matematika
Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh: **Aditya Dwi Gumara**
NRP 5002201070

Surabaya, Agustus 2026

Disetujui oleh:

**Pembimbing:**

1. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT (.....)
   NIP 19690405 199403 2 003

**Penguji:**

2. Dr. Drs. Bandung Arry Sanjoyo, M.I.Komp (.....)
   NIP 19630605 198903 1 003
3. Alvida Mustika Rukmi, S.Si., M.Si (.....)
   NIP 19720715 199802 2 001
4. Dr. Budi Setiyono, S.Si., M.T (.....)
   NIP 19720207 199702 1 001

Mengetahui
**Kepala Departemen Matematika**
**Fakultas Sains dan Analitika Data**

Dr. Didik Khusnul Arif, S.Si., M.Si.
NIP 19730930 199702 1 001

# APPROVAL SHEET

# APPROVAL SHEET

## COMPARISON OF FASTER R-CNN AND YOLOv8 FOR FIRE AND SMOKE DETECTION ON VIDEO DATA

**FINAL PROJECT**

Submitted to fulfill one of the requirements
For obtaining a degree Bachelor of Mathematics at
Bachelor Program of Mathematics
Department of Mathematics
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember

By: **Aditya Dwi Gumara**
NRP 5002201070

Surabaya, Agustus 2026

Approved By:

**Advisor:**

1. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT (.....)
   NIP 19690405 199403 2 003

**Examiners:**

2. Dr. Drs. Bandung Arry Sanjoyo, M.I.Komp (.....)
   NIP 19630605 198903 1 003

3. Alvida Mustika Rukmi, S.Si., M.Si (.....)
   NIP 19720715 199802 2 001

4. Dr. Budi Setiyono, S.Si., M.T (.....)
   NIP 19720207 199702 1 001

Acknowledged by
**Head of Mathematics Department**
**Faculty of Science and Data Analytics**

Dr. Didik Khusnul Arif, S.Si., M.Si.
NIP 19730930 199702 1 001

# PERNYATAAN ORISINALITAS

# PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa : Aditya Dwi Gumara / 5002201070
Departemen : Matematika
Dosen Pembimbing : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT / 19690405 199403 2 003

Dengan ini menyatakan bahwa Tugas Akhir dengan judul **" PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO"** adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 3 Agustus 2026

Mengetahui,
Dosen Pembimbing

(Dr. Dwi Ratna Sulistyaningrum, S.Si, MT)
NIP

Mahasiswa

(Aditya Dwi Gumara)
NRP 5002201070

# STATEMENT OF ORIGINALITY

# STATEMENT OF ORIGINALITY

The undersigned below:

Student's Name/NRP : Aditya Dwi Gumara / 5002201070
Departement : Mathematics
Advisor : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT / 19690405 199403 2 003

Hereby declare that the Final Project with the title of **"COMPARISON OF FASTER R-CNN AND YOLOv8 FOR FIRE AND SMOKE DETECTION ON VIDEO DATA"** is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 3 Agustus 2026

Acknowledged,
Advisor

(Dr. Dwi Ratna Sulistyaningrum, S.Si, MT)
NIP 19690405 199403 2 003

Student

(Aditya Dwi Gumara)
NRP 5002201070

# ABSTRAK

# ABSTRAK

## PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO

Nama Mahasiswa / NRP : Aditya Dwi Gumara / 5002201070
Departemen : Matematika FSAD - ITS
Dosen Pembimbing : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

**Abstrak**

Kebakaran merupakan salah satu bencana yang dapat menimbulkan kerugian besar dari segi ekonomi, lingkungan, maupun keselamatan manusia. Beberapa peristiwa kebakaran besar, seperti kebakaran Gedung Kejaksaan Agung pada tahun 2020 dan kebakaran kilang minyak Pertamina Cilacap pada tahun 2021, menunjukkan pentingnya sistem deteksi dini yang mampu mendeteksi kebakaran secara cepat dan akurat. Sistem deteksi kebakaran konvensional yang memanfaatkan sensor api, gas, atau suhu masih memiliki keterbatasan, seperti cakupan area yang sempit dan keterlambatan dalam mendeteksi sumber kebakaran. Oleh karena itu, penelitian ini mengusulkan pendekatan berbasis computer vision menggunakan metode deep learning untuk mendeteksi objek api (*fire*) dan asap (*smoke*) pada data video. Model yang digunakan adalah Faster R-CNN ResNet-50 FPN dan YOLOv8m. Tahapan penelitian meliputi pengumpulan dan anotasi data video, pelatihan model melalui proses *hyperparameter tuning*, serta pengujian model pada berbagai skenario kondisi pencahayaan. Hasil *hyperparameter tuning* menunjukkan bahwa konfigurasi terbaik Faster R-CNN ResNet-50 FPN diperoleh menggunakan optimizer SGD dengan learning rate 0,1, menghasilkan mAP50 sebesar 75,57% dan mAP50-95 sebesar 43,10%. Sementara itu, konfigurasi terbaik YOLOv8m diperoleh menggunakan optimizer AdamW dengan learning rate 0,001, menghasilkan mAP50 sebesar 77,95% dan mAP50-95 sebesar 44,94%. Pada tahap pengujian, YOLOv8m secara umum menghasilkan nilai Precision, Recall, dan F1-Score yang lebih tinggi dibandingkan Faster R-CNN ResNet-50 FPN serta menunjukkan performa yang lebih stabil pada berbagai kondisi pencahayaan. Selain itu, YOLOv8m memiliki kecepatan inferensi sekitar 41–43 FPS, sedangkan Faster R-CNN ResNet-50 FPN hanya mencapai sekitar 10–11 FPS. Berdasarkan hasil tersebut, dapat disimpulkan bahwa YOLOv8m merupakan model yang lebih sesuai untuk diterapkan pada sistem deteksi dini kebakaran berbasis video secara real-time karena memberikan keseimbangan yang lebih baik antara akurasi deteksi dan kecepatan inferensi.

**Kata Kunci**: kebakaran, deteksi api dan asap, deep learning, Faster R-CNN, YOLOv8, video.

# ABSTARCT

# ABSTRACT

***COMPARISON OF FASTER R-CNN AND YOLOv8 FOR FIRE AND SMOKE DETECTION ON VIDEO DATA***


| | |
|---|---|
| *Student Name* / NRP | : Aditya Dwi Gumara / 5002201070 |
| *Department* | : Matematika FSAD - ITS |
| *Supervisor* | : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT |



***Abstract***

*Fires are one of the disasters that can cause significant losses in terms of the economy, the environment, and human safety. Several major fire incidents, such as the fire at the Attorney General's Office building in 2020 and the fire at the Pertamina Cilacap oil refinery in 2021, highlight the importance of an early-warning system capable of detecting fires quickly and accurately. Conventional fire detection systems that rely on flame, gas, or temperature sensors still have limitations, such as limited coverage and delays in detecting the source of a fire. Therefore, this study proposes a computer vision-based approach using deep learning methods to detect fire and smoke objects in video data. The models used are Faster R-CNN ResNet-50 FPN and YOLOv8m. The research stages include video data collection and annotation, model training through hyperparameter tuning, and model testing under various lighting conditions. The results of the hyperparameter tuning show that the best configuration for Faster R-CNN ResNet-50 FPN was obtained using the SGD optimizer with a learning rate of 0.1, yielding an mAP50 of 75.57% and an mAP50-95 of 43.10%. Meanwhile, the best configuration for YOLOv8m was obtained using the AdamW optimizer with a learning rate of 0.001, yielding an mAP50 of 77.95% and an mAP50-95 of 44.94%. Meanwhile, the best configuration for YOLOv8m was obtained using the AdamW optimizer with a learning rate of 0.001, yielding an mAP50 of 77.95% and an mAP50-95 of 44.94%. During the testing phase, YOLOv8m generally produced higher Precision, Recall, and F1-Score values compared to Faster R-CNN ResNet-50 FPN and demonstrated more stable performance under various lighting conditions. Furthermore, YOLOv8m has an inference speed of approximately 41–43 FPS, whereas Faster R-CNN ResNet-50 FPN only reaches about 10–11 FPS. Based on these results, it can be concluded that YOLOv8m is a more suitable model for implementation in real-time, video-based early fire detection systems because it provides a better balance between detection accuracy and inference speed.*



***Keywords***: *fire, fire and smoke detection, deep learning, Faster R-CNN, YOLOv8, video*.

# KATA PENGANTAR

# KATA PENGANTAR

Puji syukur kehadirat Allah SWT Tuhan Semesta Alam karena atas berkah, rahmat dan Ridho-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul :

”PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO”

sebagai salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember Surabaya. Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis ingin mengucapkan terima kasih dan penghargaan kepada :

1. Keluarga yang telah memberikan support, nasehat, dan motivasi sehingga penulis dapat menyelesaikan Tugas Akhir ini.

2. Bapak Dr. Didik Khusnul Arif, S.Si., M.Si selaku Kepala Departemen Matematika Institut Teknologi Sepuluh Nopember Surabaya.

3. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku dosen pembimbing yang telah memberikan arahan dengan penuh kesabaran kepada penulis.

4. Bapak Dr. Drs. Bandung Arry Sanjoyo, M.I.Komp, Ibu Alvida Mustika Rukmi, S.Si., M.Si, dan Bapak Dr. Budi Setiyono, S.Si, MT selaku dosen penguji yang telah memberikan arahan, kritik, saran, dan masukan yang membangun dalam pengerjaan tugas akhir ini.

5. Bapak Mohammad Iqbal, S.Si, M.Si, Ph.D selaku dosen wali yang telah memberikan arahan dan bantuan dengan penuh kesabaran kepada penulis selama masa perkuliahan.

6. Bapak/Ibu dosen pengajar yang tidak bisa penulis sebutkan satu per satu, yang telah memberikan ilmu dan pengalaman yang bermanfaat kepada penulis, serta segenap Karyawan, Tendik dan keluarga besar Departemen Matematika Institut Teknologi Sepuluh Nopember atas dukungan dan bantuannya.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, 3 Agustus 2026

Aditya Dwi Gumara

# DAFTAR ISI

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Kebakaran merupakan salah satu peristiwa bencana yang dapat menimbulkan kerugian besar, baik dari segi ekonomi, ekologi, maupun nyawa manusia. Peristiwa ini dapat terjadi kapan saja dan di berbagai lokasi, mulai dari hutan, pemukiman, hingga kawasan industri. Di Indonesia sendiri, kebakaran masih menjadi permasalahan serius, contohnya kebakaran Gedung Kejaksaan Agung di Jakarta tahun 2020 (Kompas, 2020) dan kebakaran kilang minyak Pertamina di Cilacap tahun 2021 (Tempo, 2021) yang mengakibatkan kerugian material signifikan dan ancaman terhadap keselamatan masyarakat. Faktor penyebab kebakaran pun beragam, seperti konsleting listrik, kelalaian manusia, hingga faktor alam seperti musim kemarau panjang yang meningkatkan risiko kebakaran hutan dan lahan (Dogan dkk., 2022). Kondisi ini menunjukkan urgensi adanya sistem deteksi dini yang mampu memberikan peringatan cepat sebelum kebakaran menyebar luas.

Salah satu pendekatan yang banyak digunakan untuk deteksi kebakaran adalah sistem sensor skalar, seperti *sensor* api, gas, dan suhu. Metode ini memang lebih murah dan sederhana, tetapi memiliki keterbatasan, antara lain area jangkauan yang sempit dan hanya efektif di dalam ruangan. Selain itu, *sensor skalar* membutuhkan waktu karena api atau asap harus terlebih dahulu mencapai langit-langit tempat *sensor* terpasang (Yar dkk., 2023). Untuk mengatasi kelemahan tersebut, sistem deteksi berbasis *vision* dengan memanfaatkan kamera pengawas menjadi alternatif yang lebih efektif. Sistem *vision* mampu memantau area yang lebih luas dengan respon cepat serta dapat diintegrasikan dengan metode kecerdasan buatan (AI) untuk mendeteksi kebakaran sejak dini.

Berbagai penelitian sebelumnya telah membuktikan bahwa metode berbasis *deep learning* memiliki kemampuan yang baik dalam mendeteksi api dan asap. Penelitian oleh (Li dkk., 2024) mengembangkan sistem deteksi api dan asap secara *real-time* menggunakan YOLOv8 yang mampu menghasilkan akurasi tinggi dengan kecepatan inferensi yang sesuai untuk aplikasi pemantauan kebakaran. (Fajri dkk., 2024) mengeksplorasi penggunaan Mask R-CNN untuk mendeteksi objek api dan asap, dengan memperoleh nilai rata-rata *precision* sebesar 0,38 dan *recall* sebesar 0,29. Selanjutnya, (Pincott dkk., 2024) membandingkan Faster R-CNN Inception V2 dan SSD MobileNet V2 untuk deteksi kebakaran dalam ruangan, di mana Faster R-CNN menunjukkan performa deteksi yang lebih baik meskipun membutuhkan waktu inferensi yang lebih lama. Selain itu, (Alharthi dkk., 2025) melakukan perbandingan antara YOLOv8 dan Faster R-CNN pada video pengawasan dan menunjukkan bahwa YOLOv8 unggul dalam kecepatan deteksi untuk aplikasi *real-time*, sedangkan Faster R-CNN memberikan akurasi yang lebih baik pada beberapa kondisi, terutama dalam mendeteksi objek api berukuran kecil. Hasil-hasil penelitian tersebut menunjukkan bahwa YOLOv8 dan Faster R-CNN merupakan dua metode yang memiliki keunggulan masing-masing sehingga layak untuk dibandingkan lebih lanjut dalam deteksi api dan asap pada data video.

Berdasarkan latar belakang tersebut, penelitian ini berfokus pada penerapan Faster R-CNN dan YOLOv8 untuk deteksi api dan asap pada data video. Kedua metode dipilih karena mewakili dua pendekatan utama dalam deteksi objek, yaitu two-stage detector dan one-stage detector, sehingga dapat memberikan gambaran yang komprehensif mengenai perbandingan antara akurasi deteksi dan kecepatan inferensi. Faster R-CNN dipilih sebagai representasi metode *two-stage* karena dikenal memiliki akurasi deteksi yang tinggi, terutama dalam mendeteksi objek berukuran kecil atau memiliki bentuk yang kompleks melalui mekanisme Region Proposal Network (RPN). Sementara itu, YOLOv8 dipilih sebagai representasi metode

*one-stage* karena mampu melakukan deteksi objek secara langsung dalam satu tahap sehingga memiliki kecepatan inferensi yang tinggi dan sesuai untuk aplikasi real-time. Dibandingkan metode lain seperti SSD, RetinaNet, atau Mask R-CNN, Faster R-CNN dan YOLOv8 merupakan dua algoritma yang paling banyak digunakan sebagai acuan (*benchmark*) dalam penelitian deteksi objek karena mampu memberikan keseimbangan antara akurasi dan efisiensi komputasi.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang diangkat dalam penelitian ini sebagai berikut:

1. Bagaimana implementasi model *Faster* R-CNN untuk deteksi api dan asap pada data video?
2. Bagaimana implementasi model *YOLOv8* untuk deteksi api dan asap pada data video?
3. Bagaimana perbandingan kinerja *Faster* R-CNN dan *YOLOv8* untuk deteksi api dan asap pada data video?

## 1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian yang dilakukan oleh peneliti sebagai berikut:

1. Dataset diperoleh dari 2 cara yaitu, data primer dan data sekunder. Data primer yang digunakan adalah kumpulan video dari kamera CCTV. Lalu data sekunder yang digunakan berasal dari *https://universe.roboflow.com/nghia-tdxoy/fire-smoke-wucvr* digunakan oleh (Kim, B. et al., 2019) (https://doi.org/10.3390/app9142862) tentang *FDS (Fire and Smoke detection)* merupakan *dataset* publik yang terdiri dari video *fire* dan *smoke* dalam kondisi berbeda.
2. Jenis model yang akan digunakan adalah *pre-trained* model seperti *Faster* R-CNN dengan *backbone ResNet-50 FPN* dari *Detectron2* dan *YOLOv8m* dari *Ultralytics.*

## 1.4 Tujuan

Tujuan dalam penelitian tugas akhir ini adalah sebagai berikut:

1. Mengimplementasikan model *Faster* R-CNN untuk deteksi api dan asap pada data video.
2. Mengimplementasikan model *YOLOv8* untuk deteksi api dan asap pada data video.
3. Membandingkan kinerja *Faster* R-CNN dan *YOLOv8* untuk deteksi api dan asap pada data video.

## 1.5 Manfaat

Manfaat dari penelitian ini diharapkan sebagai berikut:

1. Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan ilmu pengetahuan, khususnya di bidang *computer vision* dan *deep learning* untuk sistem deteksi dini kebakaran. Implementasi *Faster R-CNN* dan *YOLOv8* pada data video dapat menjadi rujukan bagi penelitian selanjutnya dalam meningkatkan efektivitas model deteksi objek, khususnya pada kasus kebakaran.
2. Hasil penelitian ini diharapkan dapat menjadi bagian dari sistem deteksi dini kebakaran yang lebih cepat, akurat, dan efisien.

# BAB II
# TINJAUAN PUSTAKA

# BAB II
# TINJAUAN PUSTAKA

## 2.1 Penelitian Terdahulu

Li et al. melakukan penelitian yang berjudul *"Real-Time Fire and Smoke Detection Based on YOLOv8 Deep Learning Model"*. Penelitian ini mengembangkan sistem deteksi api dan asap secara real-time menggunakan YOLOv8 pada citra dan video. Hasil pengujian menunjukkan bahwa model YOLOv8 mampu mendeteksi objek api dan asap dengan tingkat akurasi yang tinggi serta kecepatan pemrosesan yang memenuhi kebutuhan sistem pemantauan kebakaran secara langsung. (Li et al., 2024).

Fajri et al. melakukan penelitian yang berjudul *"Fire and Smoke Object Detection Using Mask R-CNN"*. Dalam penelitian ini, penulis mengeksplorasi potensi teknologi visi komputer, khususnya Mask R-CNN, sebagai solusi yang lebih efisien dan akurat untuk deteksi kebakaran dan asap dibandingkan metode tradisional. Hasil evaluasi menunjukkan bahwa model memperoleh nilai precision rata-rata sebesar 0,38 dan recall rata-rata sebesar 0,29. (Fajri et al., 2024).

Pincott et al. melakukan penelitian yang berjudul *"Indoor Fire Detection Utilizing Computer Vision-Based Strategies"*. Penelitian ini memanfaatkan model Faster R-CNN Inception V2 dan SSD MobileNet V2 untuk mendeteksi api dan asap pada lingkungan dalam ruangan. Model dilatih menggunakan kumpulan data citra kebakaran dengan variasi kepadatan asap, kemudian dievaluasi menggunakan beberapa video uji. Hasil penelitian menunjukkan bahwa Faster R-CNN memiliki kemampuan deteksi yang lebih baik dibandingkan SSD MobileNet V2, meskipun membutuhkan waktu inferensi yang lebih lama. (Pincott et al., 2024).

Alharthi et al. melakukan penelitian yang berjudul *"Comparative Analysis of YOLOv8 and Faster R-CNN for Fire Detection in Surveillance Videos"*. Penelitian ini membandingkan kinerja YOLOv8 dan Faster R-CNN dalam mendeteksi api pada video pengawasan. Hasil penelitian menunjukkan bahwa YOLOv8 memiliki kecepatan inferensi yang lebih tinggi sehingga lebih sesuai untuk aplikasi real-time, sedangkan Faster R-CNN menghasilkan akurasi deteksi yang lebih baik pada beberapa skenario dengan objek api berukuran kecil. (Alharthi et al., 2025).

Pada Tugas Akhir ini, telah dilakukan penelitian yang mengimplementasikan sistem deteksi dini kebakaran berbasis video menggunakan *Faster R-CNN* dan *YOLOv8*. Hasil akhir penelitian adalah sistem yang dapat mendeteksi api (*fire*) dan asap (*smoke*) pada data video.

## 2.2 Kebakaran

Kebakaran merupakan peristiwa bencana yang ditandai dengan munculnya api yang tidak terkendali dan dapat menimbulkan kerusakan lingkungan, kerugian ekonomi, serta mengancam keselamatan manusia (Dogan et al., 2022). Kebakaran dapat terjadi di berbagai lokasi, baik di lingkungan pemukiman, gedung perkantoran, maupun di area hutan dan lahan. Faktor penyebab kebakaran umumnya berasal dari konsleting listrik, kelalaian manusia, aktivitas industri, hingga faktor alam seperti musim kemarau panjang (Yar et al., 2023).

Kebakaran seringkali menimbulkan dampak lanjutan berupa munculnya asap tebal. Asap kebakaran mengandung partikel berbahaya seperti karbon monoksida, karbon dioksida, dan partikulat halus yang dapat mengganggu kesehatan pernapasan serta menurunkan kualitas udara secara signifikan (Biswas et al., 2023). Asap juga berperan sebagai indikator dini bahwa sedang terjadi proses pembakaran, sehingga deteksi asap sama pentingnya dengan deteksi api dalam sistem peringatan dini kebakaran.


Gambar 2.1 Contoh citra asap (*smoke*) dan api (*fire*)

Penelitian ini menggunakan dua kelas objek sebagai dasar dalam proses deteksi berbasis *computer vision,* yaitu kelas api (*fire*) dan asap (*smoke*). Pada Gambar 2.2 ditampilkan citra yang menunjukkan kumpulan dataset *fire* berupa api yang muncul pada area kebakaran. Gambar 2.3 menampilkan kumpulan dataset *smoke* berupa asap kebakaran yang menyebar di udara. Dengan adanya pembagian kelas ini sistem deteksi dapat dilatih supaya belajar untuk bisa membedakan api dan asap, baik dalam kasus kebakaran ataupun bukan kebakaran.

Gambar 2.2 Kumpulan *dataset fire*

Gambar 2.3 Kumpulan *dataset smoke*

## 2.3 Citra Digital

Citra ialah gambaran, kesamaan, atau tiruan dari suatu objek. Citra yang dihasilkan oleh sistem pengumpulan data dapat berupa gambar visual seperti foto, sinyal analog yang muncul dalam bentuk video seperti tampilan di layar televisi, atau format digital yang bisa langsung disimpan di sebuah perangkat penyimpanan. Citra terbagi menjadi dua kategori, yaitu citra tetap (*still image*) dan citra dinamis (*moving image*). Citra tetap adalah gambar tunggal yang tidak bergerak, sementara citra dinamis terdiri dari serangkaian citra tetap yang ditampilkan berturut-turut sehingga menciptakan ilusi gerakan.

Pada awalnya, citra yang dikenal oleh manusia memiliki bentuk citra kontinu. Ini merupakan gambaran objek yang dibuat oleh sistem optik yang menangkap sinyal analog dan disajikan dalam format dua dimensi. Rentang nilai cahaya yang terkandung dalam citra kontinu tidak terbatas. Contoh dari citra kontinu adalah penglihatan manusia dan kamera analog. Citra kontinu tidak bisa langsung direpresentasikan oleh komputer. Oleh karena itu, diperlukan proses untuk mengubah nilai-nilai dalam citra kontinu agar komputer mampu memahami dan menerjemahkan informasi yang ada di dalamnya. Hasil dari proses ini disebut sebagai citra digital.

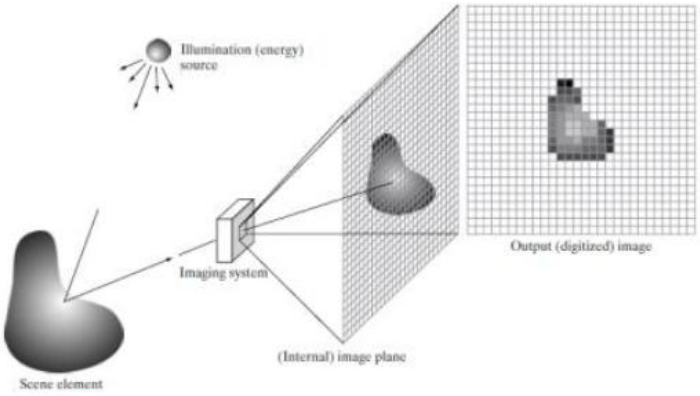
Dalam buku *Digital Image Processing* yang ditulis oleh Rafael C. Gonzalez dan Richard E. Woods, dijelaskan bahwa citra digital adalah fungsi dua dimensi yang bisa dinyatakan dengan fungsi f(x,y), di mana x dan y merupakan koordinat spasial. Nilai dari fungsi f pada koordinat (x,y) tertentu menunjukkan intensitas cahaya, yang merepresentasikan warna cahaya yang ada pada citra kontinu. Citra digital merupakan citra yang memiliki (x,y) dan nilai intensitas f yang terbatas (discrete quantities), yang telah melalui proses digitalisasi spasial serta kuantitas.

### 2.3.1 Digitalisasi Spasial (Sampling)

*Sampling* adalah aktivitas pengumpulan data dari citra kontinu yang memiliki dimensi tertentu untuk dipecah menjadi sejumlah blok kecil. Blok-blok ini disebut piksel. Dengan demikian, citra digital yang biasanya ditampilkan dalam format matriks memiliki ukuran M x N, di mana M mewakili jumlah baris dan N mewakili kolom. Ini juga dapat ditafsirkan sebagai citra digital yang terdiri dari M x N piksel. Notasi matriks untuk citra digital dapat dinyatakan sebagai berikut:

$$F = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & \cdots & \cdots & f(1,M-1) \\ \cdots & \cdots & \cdots & \cdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

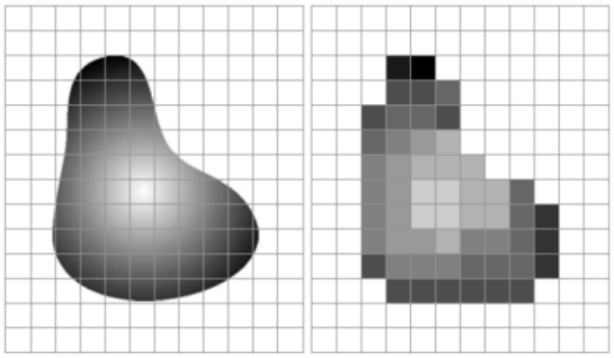
Proses *sampling* dari citra kontinu ke citra digital dapat dilihat pada Gambar 2. 4 di bawah ini:




Gambar 2.4 Proses *sampling* dari citra kontinu menuju citra digital

### 2.3.2 Digitalisasi Intensitas (Kuantisasi)

Kuantisasi adalah tahap memberikan nilai derajat keabuan pada setiap titik piksel yang menunjukkan warna asli dari citra kontinu. Rentang nilai keabuan berkisar antara 0 hingga 255.

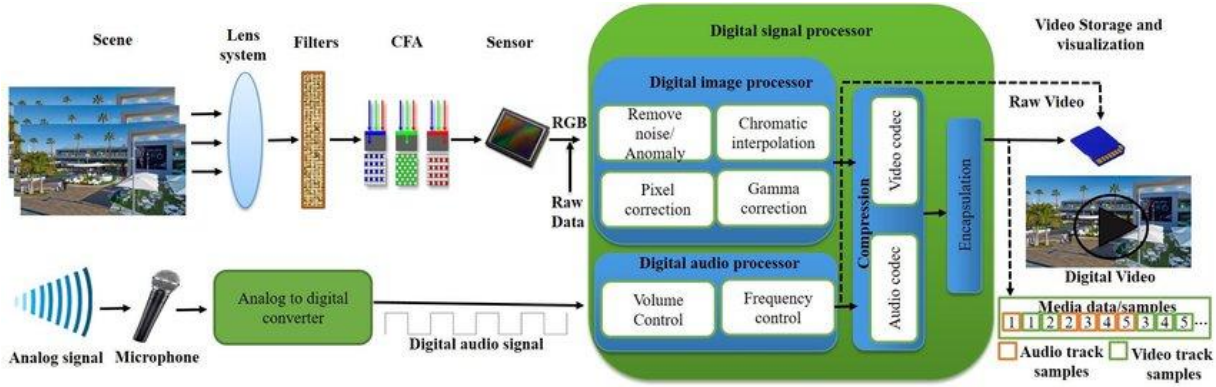


Gambar 2.5 Menunjukkan proses sampling dan kuantisasi. (kiri) Citra kontinu yang diletakkan pada *sensor array*. (kanan) Citra setelah dilakukan proses *sampling* dan kuantisasi, di mana setiap piksel pada citra tersebut memiliki derajat keabuan yang berbeda-beda.

## 2.4 Video Digital

Video adalah sebuah teknologi untuk merekam, mengambil, memproses, menyimpan, dan menghidupkan kembali serangkaian citra. Dalam bukunya yang berjudul *Handbook of Image and Video Processing*, Alan C. Bovik menjelaskan bahwa video digital adalah hasil dari pengambilan sampel dan kuantisasi video analog. Pada dasarnya, tidak ada perbedaan dalam proses pengambilan sampel dan kuantisasi antara citra digital dan video digital. Namun, video analog yang kita saksikan sehari-hari, seperti yang terlihat di TV analog, sebenarnya tidak sepenuhnya kontinu, melainkan terdiri dari beberapa frame yang ditampilkan pada kecepatan tertentu. Setiap frame berisi citra kontinu, dan kecepatan tampilan citra-citra tersebut dikenal sebagai *frame rate*, diukur dalam fps (*frame per second*). Jika *frame rate* cukup tinggi, maka video akan terlihat lebih halus.

Video analog dapat direpresentasikan dengan fungsi I(x,y,t), di mana (x,y) adalah nilai kontinu dari fungsi I dan t melambangkan waktu. Sebenarnya, tampilan video analog di televisi atau monitor adalah representasi dari fungsi sinyal elektrik satu dimensi V(t). Sinyal elektrik satu dimensi tersebut terdiri dari beberapa citra kontinu I(x,y,t) dengan jumlah citra (x,y) dan waktu (t) yang tertentu. Pada Gambar 2.6 ditunjukkan contoh proses pembuatan video digital.

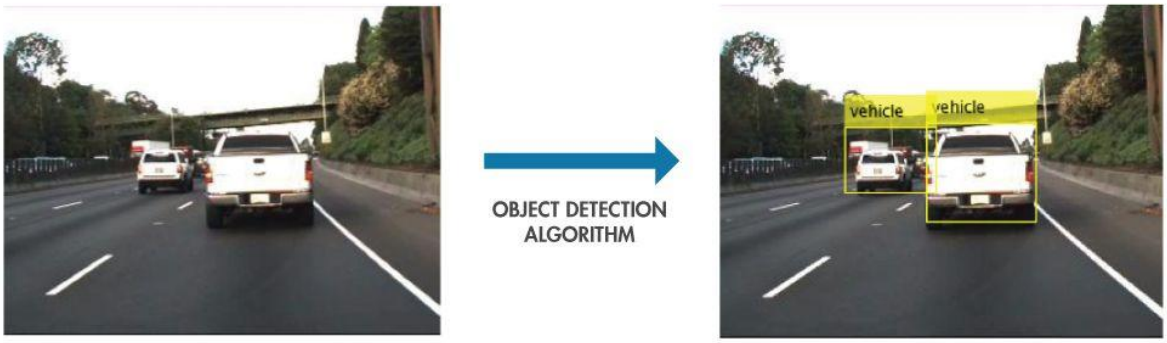



Gambar 2.6 Proses pembuatan video digital

## 2.5 Deteksi Objek

Deteksi objek merupakan salah satu bagian dari *computer vision* yang digunakan untuk mendeteksi suatu objek dari kelas tertentu pada suatu citra visual baik berupa gambar, video, ataupun secara *real-time* melalui kamera (Diwan et al., 2022). Sistem deteksi objek bekerja melalui tahapan *pre-processing*, ekstraksi fitur, dan klasifikasi. *Preprocessing* adalah pengolahan data asli untuk meningkatkan kualitas dari data sebelum diolah ke tahapan berikutnya. Ekstraksi dan klasifikasi adalah suatu proses untuk mendapatkan fitur dari data yang digunakan agar objek dapat dikategorikan jenisnya. Metode umum yang digunakan dalam deteksi objek antara lain adalah *Region-based Convolutional Neural Network* (RCNN), *Faster RCNN*, *Single Shot Multibox Detector* (SSD), dan *You Only Look Once* (YOLO).

Cara kerja deteksi objek adalah mendeteksi lokasi objek dalam gambar dan menggambar *bounding box* di sekitar objek tersebut. Setelah itu, objek dalam *bounding box* diklasifikasikan jenisnya. Secara umum deteksi objek adalah mengidentifikasi lokasi objek dalam gambar sedangkan klasifikasi adalah mengklasifikasikan gambar kedalam kategori tertentu. Contoh deteksi objek ditunjukkan pada Gambar 2.7.




Gambar 2.7 Contoh deteksi objek

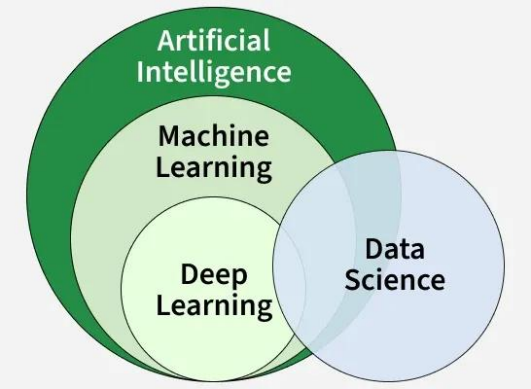
Penelitian terkait deteksi objek telah banyak dilakukan, salah satu contohnya adalah dengan penerapan *deep learning*.

## 2.6 *Deep Learning*

*Deep learning* adalah salah satu cabang dari pembelajaran mesin yang saat ini berkembang dengan sangat pesat. Meskipun pembelajaran mendalam mulai dikembangkan pada tahun 1940-an, saat itu belum begitu populer. Antara tahun 1940 dan 1960-an, pembelajaran mendalam dikenal dengan sebutan sibernetika, seiring dengan kemajuan teori-teori terkait biologi, dan

diterapkan dalam model awal, yakni *perceptron* yang memungkinkan pelatihan *neuron*. Kemudian, dari tahun 1980 hingga 1995, sibernetika berevolusi menjadi konektionisme atau yang lebih dikenal sebagai jaringan syaraf, yang dikembangkan bersamaan dengan teknik *back-propagation* untuk melatih jaringan syaraf dengan satu atau dua lapisan tersembunyi. Selanjutnya, sejak tahun 2006 hingga sekarang, konektionisme telah bertransformasi menjadi *Deep learning* (Goodfellow et al., 2016).

*Deep learning* merupakan algoritma yang terinspirasi oleh sistem syaraf alami (Goodfellow et al., 2016). *Deep learning* menggabungkan beberapa lapisan pemrosesan yang tidak linear, dengan elemen-elemen sederhana yang dioperasikan secara bersamaan. *Deep learning* terdiri dari lapisan *input*, beberapa lapisan tersembunyi, dan lapisan *output*. Lapisan-lapisan ini saling terhubung melalui *node* atau *neuron*, di mana setiap lapisan tersembunyi memanfaatkan *output* dari lapisan sebelumnya sebagai input untuk lapisan berikutnya (The Mathworks., 2018). Hubungan antara *Machine Learning* dan *Deep learning* ditunjukkan pada Gambar 2.8.




Gambar 2.8 Hubungan antara *Machine Learning* dan *Deep learning*

Pada tabel 2.1 diperlihatkan perbedaan antara *Machine Learning* dan *Deep learning*.

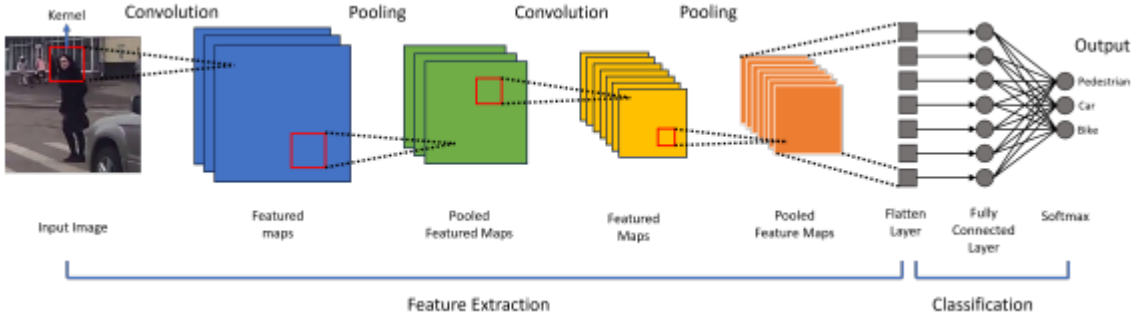
Tabel 2.1 Perbedaan *Machine Learning* dan *Deep learning*

| *Machine Learning* | *Deep learning* |
|---|---|
| Membutuhkan sedikit *dataset* | Membutuhkan banyak *dataset* |
| *Training* membutuhkan waktu relatif singkat | *Training* membutuhkan waktu relatif lebih lama |
| Ekstraksi fitur dan klasifikasi dilakukan secara manual | Ekstraksi fitur dan klasifikasi dilakukan secara otomatis |
| Membutuhkan daya komputasi yang lebih kecil | Membutuhkan daya komputasi yang lebih besar |

## 2.7 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network* (CNN) adalah salah satu algoritma *deep learning* yang sangat populer dan sering digunakan. Keunggulan utama CNN adalah kemampuannya untuk mempelajari fitur-fitur yang relevan pada sebuah data secara otomatis (O'Shea & Nash, 2015).

Berbagai macam aplikasi CNN telah dikembangkan hingga saat ini, seperti pengenalan suara, *computer vision*, pengenalan wajah manusia, dan deteksi objek. Jenis CNN yang umum digunakan biasanya terdiri dari beberapa *convolution layer*, *pooling layer*, dan diakhir susunan terdapat *fully connected layer*. Berdasarkan arsitekturnya, CNN termasuk salah satu *feed-forward artificial neural networks*, yaitu *neural network* paling sederhana (O'Shea & Nash, 2015). Arsitektur CNN pada umumnya diilustrasikan pada Gambar 2.9. Berikut adalah penjelasan dari beberapa komponen dalam CNN.




Gambar 2.9 Arsitektur Umum *Convolutional Neural Networks*

### 2.7.1 Convolutional Layers

*Layer* ini digunakan untuk mengekstrak fitur dari data yang diinputkan dengan menggunakan filter yang digeser secara parsial pada data (O'Shea & Nash, 2015). Setiap kali filter digeser, filter melakukan operasi matematika pada area data yang dicakup dan menghasilkan nilai baru yang disimpan dalam sebuah matriks yang disebut *feature map*. Filter yang digunakan dalam *convolutional layer* dapat memiliki ukuran yang berbeda-beda. Filter ini melakukan operasi matematika dari setiap data yang diinputkan pada setiap *channel*. Operasi matematika yang digunakan adalah konvolusi yang biasanya disimbolkan sebagai $*$. Operasi konvolusi merupakan operasi matematis yang terdiri atas operasi perkalian, pergeseran, dan penjumlahan. Pada suatu citra, operasi konvolusi menggeser *kernel* piksel per piksel dan hasilnya disimpan dalam matriks baru. Sebagai contoh, jika $I$ adalah citra berbentuk 2D dan $K$ adalah *kernel* berbentuk 2D maka persamaan konvolusi menjadi seperti berikut.

$$Y(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, j+n) K(m,n) \tag{2.2}$$

dengan:

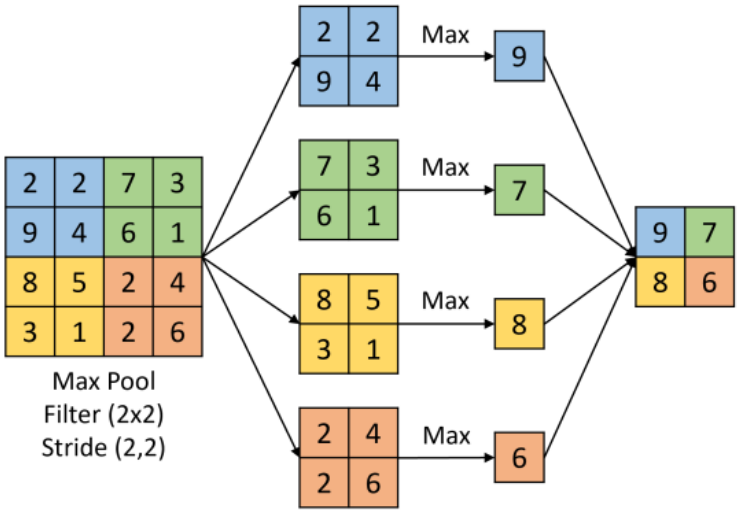
$X(i,j)$ : Piksel gambar *input* di (i, j)
$K(m,n)$ : Nilai *kernel* pada posisi (m, n)
$Y(i,j)$ : Hasil (*feature map*) pada posisi *output* (i, j)

### 2.7.2 Pooling Layers

*Pooling layer* adalah *layer* dalam arsitektur *neural network* yang digunakan untuk mengurangi dimensi data (O'Shea & Nash, 2015). Hal ini dilakukan dengan melakukan operasi pengurangan ukuran pada bagian-bagian tertentu dari data seperti pengambilan rata-rata atau maksimum. Hal ini bertujuan untuk mengurangi jumlah parameter yang harus diterjemahkan dan mengurangi *overfitting*. *Pooling* juga meningkatkan invariansi terhadap transformasi pada data seperti rotasi atau pergeseran. Berikut adalah beberapa jenis *pooling layer*:

1. *Max Pooling*

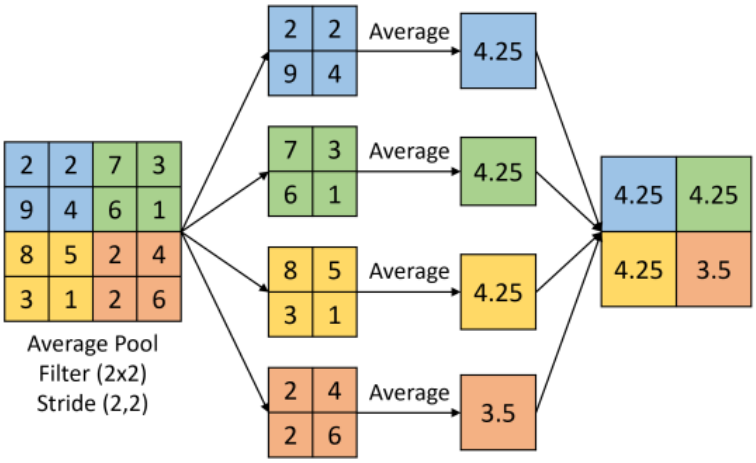
*Max pooling* adalah operasi *pooling* yang memilih elemen maksimum dari setiap submatriks dalam peta fitur yang diberikan. *Output* setelah layer *max pooling* adalah peta fitur yang berisi fitur yang paling menonjol dari peta fitur sebelumnya.




Gambar 2.10 Ilustrasi *Max Pooling*

2. *Average Pooling*

*Average pooling* menghitung rata-rata dari elemen yang ada dalam sub-matriks pada peta fitur yang diberikan. *Average pooling* memberikan peta fitur berisi rata-rata dari submatriks pada peta fitur sebelumnya.




Gambar 2.11 Ilustrasi *Average Pooling*

### 2.7.3 Batch Normalization Layers

Normalisasi pada set data mentransformasi data tersebut agar berada pada skala yang serupa. Salah satu cara umum untuk melakukan normalisasi adalah dengan menghitung rata-rata dan simpangan baku dari set data, lalu mentransformasi setiap sampel dengan mengurangkan rata-rata dan membaginya dengan simpangan baku. Normalisasi dapat membantu pelatihan jaringan saraf kita karena fiturfitur yang berbeda berada pada skala yang serupa, yang membantu menstabilkan langkah penurunan gradien, memungkinkan penggunaan *learning rate* yang lebih besar, atau membantu model konvergen lebih cepat untuk *learning rate* tertentu (Ioffe & Szegedy, 2015). Salah satu bentuk normalisasi yang dilakukan pada CNN adalah *batch normalization* (BN). BN melakukan standarisasi *input* dari sebuah lapisan pada

satu *batch* data. BN berfungsi untuk menstandarisasi *input* ke sebuah lapisan dengan tujuan untuk mengurangi masalah *internal covariate shift*, yaitu saat distribusi *input* setiap lapisan berubah selama pelatihan, seiring dengan perubahan parameter lapisan sebelumnya.

### 2.7.4 Fungsi Aktivasi

Fungsi aktivasi dalam *neural network* digunakan untuk menentukan *output* dari setiap *neuron* (O'Shea & Nash, 2015). Fungsi ini mengubah *input* yang diterima oleh *neuron* menjadi *output* yang digunakan oleh *neuron* berikutnya dalam jaringan. Berikut adalah beberapa fungsi aktivasi:

1. *Rectified Linear Units (*ReLU*)*

$$ReLU(x) = \max(0, x) \tag{2.3}$$

2. Softmax

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{2.4}$$

dengan:

$z_i$ : Nilai input mentah (logit) untuk kelas ke i
$K$ : Total jumlah kelas yang diprediksi
$e$ : Bilangan Euler (≈ 2.718)

3. *Sigmoid Linear Units* (*SiLU*)

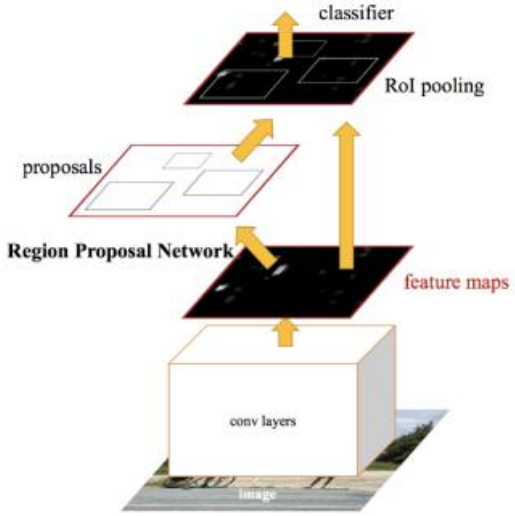
$$SiLU(x) = \frac{x}{1 + e^{-x}} \tag{2.5}$$

## 2.8 *Faster Regional – Convolutional Neural Network (Faster R – CNN)*

*Faster Regional – Convolutional Neural Network* (*Faster* R – CNN) adalah sebuah algoritma jaringan saraf dalam yang diperkenalkan oleh Ren dan rekan-rekannya (Nair et al. , 2010). *Faster* R – CNN merupakan pengembangan dari algoritma *Fast Regional Neural Network* (*Fast* R – CNN). Algoritma *Fast* R – CNN menggunakan metode pencarian selektif untuk mengekstraksi fitur dan menentukan lapisan proposal, namun penggunaan metode pencarian selektif membuat waktu pemrosesan *Fast* R – CNN menjadi lambat, karena pencarian selektif adalah algoritma terpisah yang hanya bisa dijalankan di CPU. Oleh karena itu, *Faster* R – CNN mengadopsi metode *Region Proposal Network* (RPN) guna menggantikan metode pencarian selektif, sehingga waktu pemrosesan pada *Faster* R – CNN dapat ditingkatkan. RPN menerima gambar (dalam berbagai ukuran) sebagai *input* dan memproduksi lapisan proposal. Untuk menghasilkan lapisan proposal, perlu dilakukan penggeseran sebuah jaringan kecil melalui *output feature maps* konvolusional dengan lapisan konvolusi terakhir yang dibagikan. Jaringan kecil tersebut menerima *input* berupa jendela spasial $n \times n$ dari *feature maps* konvolusional. (Zhang et al. , 2017)

Secara umum, arsitektur *Faster* R – CNN terdiri dari tiga elemen, yaitu CNN, RPN, dan *Fast* R – CNN. CNN bertugas menghasilkan *feature maps* yang digunakan dalam metode RPN dan *Fast* R – CNN. *Feature maps* ini dipakai oleh metode RPN untuk menciptakan *region proposals*. *Region proposal* menghasilkan *output* berupa lapisan proposal. Dalam *Fast* R – CNN, digunakan teknik pooling Region of Interest (RoI) pada *feature maps* dan lapisan

proposal untuk mendapatkan ukuran proposal yang seragam. Struktur dasar *Faster* R – CNN dapat dilihat pada Gambar 2.12.




Gambar 2.12 Arsitektur umum dari *Faster* R – CNN

Berdasarkan Gambar 2.12, langkah-langkah dalam algoritma *Faster* R – CNN dapat dijelaskan sebagai berikut:

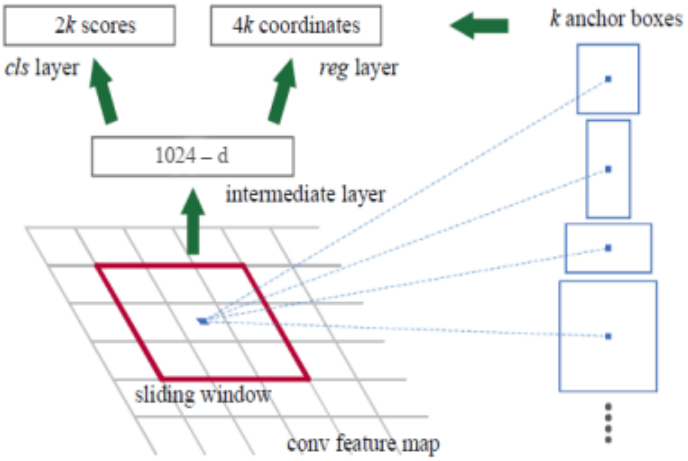
1. Citra dimasukkan untuk diproses oleh CNN (ConvNet).
2. Di dalam CNN, fitur diekstrak sehingga dihasilkan *feature maps* dari citra yang dimasukkan.
3. Selanjutnya, *feature maps* diproses melalui RPN untuk menghasilkan *proposal layer*.
4. Terapkan RoI pooling pada *proposal layer* dan *feature maps* agar semua proposal kembali ke ukuran yang seragam.
5. Kemudian, semua proposal diproses ke *fully – connected layer* yang mencakup *layer softmax* dan *layer regresi linear* untuk melakukan klasifikasi dan menentukan *predicted box* objek.

### 2.8.1 RPN

RPN merupakan sebuah algoritma yang digunakan dalam *Faster* R-CNN untuk menghasilkan *proposal layer* (Nair et al. , 2010). Dalam *Faster* R-CNN, fungsi pencarian selektif digantikan oleh RPN yang diintegrasikan ke dalam CNN untuk mempercepat proses klasifikasi. Langkah-langkah di dalam algoritma RPN dapat dirinci sebagai berikut:

1. Masukkan *input* yang berupa peta fitur ke dalam lapisan RPN.
2. Selanjutnya, dilakukan pergerakan filter (*sliding windows*) berukuran 3 x 3, dengan stride = 1 dan padding = 1 di seluruh peta fitur.
3. Di setiap lokasi *sliding windows*, beberapa proposal wilayah diprediksi secara simultan. Maksimal jumlah proposal yang dapat dihasilkan adalah sebanyak k.

k proposal adalah parameter yang berhubungan dengan k kotak acuan, yang dikenal sebagai *anchor*. *Anchor* ini terpusat pada jendela geser dan berhubungan dengan skala dan rasio aspek. Umumnya, terdapat 3 skala dan 3 rasio aspek yang digunakan, sehingga jumlah k = 9 (k = 3 x 3) anchor di setiap jendela geser. Skala yang digunakan adalah 642, 1282, dan 2562 dengan rasio aspek 1:1, 1:2, dan 2:1. *Anchor* diterapkan pada ukuran asli citra input. *Output* dari proses RPN dapat dilihat pada Gambar 2.13.



Gambar 2.13 *Output* RPN

Berdasarkan pada Gambar 2.13, dapat dijelaskan bahwa:

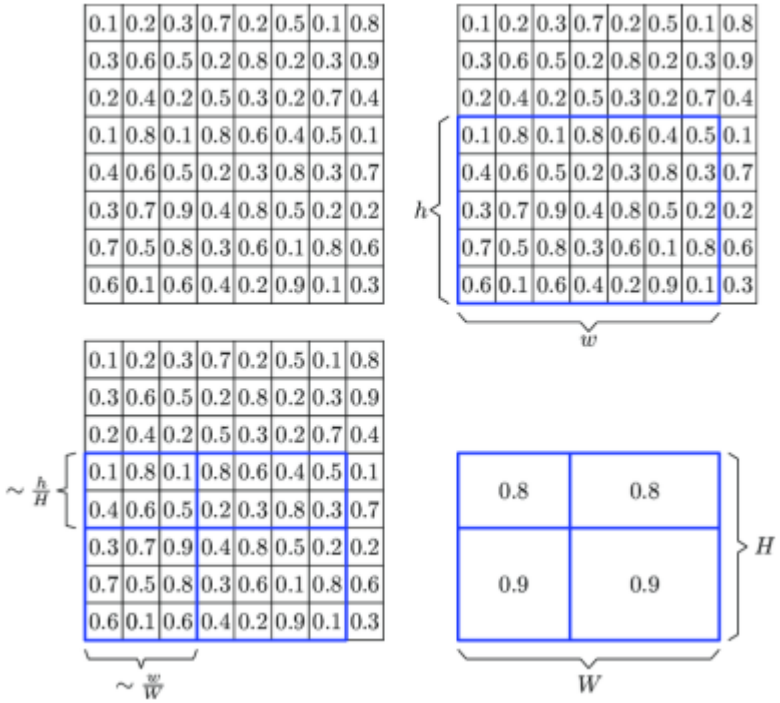
1. *Output* dari proses RPN memiliki dimensi 1024 $x$ 1 $x$ 1 untuk *proposal layer*, yang juga dikenal sebagai 1024 - d *feature vector*.
2. Lapisan cls (*classifier*) mengeluarkan skor sebanyak $2k$ untuk memprediksi apakah sebuah objek atau non-objek terdapat pada $k$ boxes.
3. Lapisan reg (*regressor*) menghasilkan skor sejumlah $4k$ untuk menentukan koordinat (posisi pusat box, lebar, dan tinggi) pada $k$ boxes.
4. Dengan dimensi $W$ $x$ $H$ *feature maps*, diperoleh total $WHk$ anchor.

### 2.8.2 Region of Interest Pooling

*Region of Interest Pooling* (*RoI Pooling*) merupakan lapisan dalam jaringan saraf yang bertujuan untuk mempercepat proses baik dalam fase pelatihan maupun saat klasifikasi. *RoI Pooling* pertama kali digunakan dalam *Fast* R – CNN yang diusulkan oleh Ross Girshick (He et al. , 2016). Dalam *RoI pooling*, ukuran peta fitur dan lapisan proposal dari tahap sebelumnya disesuaikan ukurannya. Proses dalam algoritma *RoI pooling* dapat dijelaskan sebagai berikut:

1. Masukkan *input* dalam bentuk peta fitur dan lapisan proposal dari RPN
2. Selanjutnya, bagi proposal menjadi beberapa bagian yang memiliki ukuran seragam (jumlah yang sesuai dengan dimensi output)
3. Kemudian, cari nilai maksimum di setiap bagian yang telah dibuat
4. Setelah itu, salin nilai maksimum yang ditemukan ke dalam *buffer output*.

Contoh langkah-langkah *RoI Pooling* dapat dilihat pada Gambar 2.14.




Gambar 2.14 Contoh Perhitungan Tahap *RoI Pooling*

### 2.8.3 Loss Function

*Loss Function* adalah suatu rumus yang digunakan untuk mengukur perbedaan antara hasil yang diprediksi dan data nyata selama proses pelatihan. Hasil perhitungan ini dikenal sebagai nilai kesalahan. *Loss Function* yang diterapkan dalam metode *Faster* R – CNN dikenal sebagai *Loss Function* multiclass yang dapat dituliskan seperti berikut: (Nair et al. , 2010)

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \qquad (2.6)$$

dengan:

$i$ : *Index anchor* pada *mini – batch*
$p_i$ : Kemungkinan yang diprediksi bahwa anchor $i$ merupakan objek
$p_i^*$ : Label ground truth box, dimana { $p_i^*$ = 1, $untuk\ positif\ anchor$ $p_i^*$ = 0, $untuk\ negatif\ anchor$
$N_{cls}$ : Jumlah *anchor* pada *mini – batch*
$L_{cls}$ : *Log loss* antara objek dan non objek
$t_i$ : Vektor yang menyatakan parameterisasi dari 4 koordinat pada box yang diprediksi
$t_i^*$ : *Ground truth box* yang memiliki nilai positif
$\lambda$ : Nilai konstan
$N_{reg}$ : Jumlah total *anchor*
$L_{reg}$ : *Smooth* $L_1(t_i - t_i^*)$, Dimana *Smooth* $L_1(x) = \{0{,}5x^2;\ jika\ |x| < 1\ |x| - 0{,}5;\ |x|$ lainnya

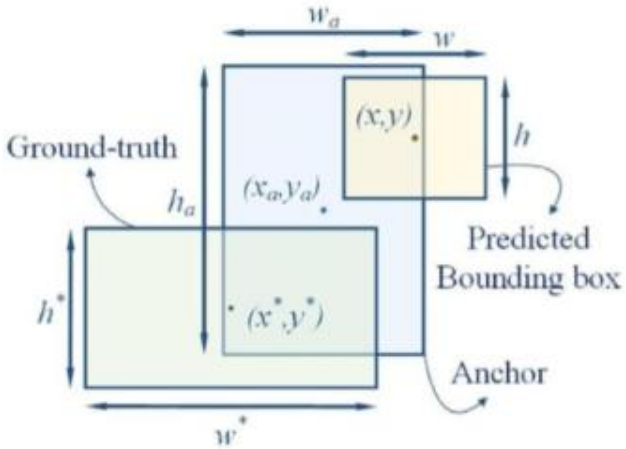
Untuk regresi *bounding box*, parameter diambil dari empat titik koordinat, yaitu dengan cara berikut:

$$
\begin{aligned}
t_x &= \frac{x - x_a}{w_a}, \qquad t_y = \frac{y - y_a}{h_a} \\
t_w &= \log\left(\frac{w}{w_a}\right), \quad t_h = \log\left(\frac{h}{h_a}\right) \\
t_x^* &= \frac{x^* - x_a}{w_a}, \qquad t_y^* = \frac{y^* - y_a}{h_a} \\
t_w^* &= \log\left(\frac{w^*}{w_a}\right), \quad t_h = \log\left(\frac{h^*}{h_a}\right)
\end{aligned} \tag{2.7}
$$

dengan:

| | |
|---|---|
| $x, y$ | : koordinat pusat *predicted box* |
| $x_a, y_a$ | : koordinat pusat *anchor box* |
| $x^*, y^*$ | : koordinat pusat *ground truth box* |
| $w$ | : lebar *predicted box* |
| $w_a$ | : lebar *anchor box* |
| $w^*$ | : lebar *ground truth box* |
| $h$ | : tinggi *predicted box* |
| $h_a$ | : tinggi *anchor box* |
| $h^*$ | : tinggi *ground truth box* |

Hal Ini bisa dikategorikan sebagai regresi *bounding box* dari sebuah *anchor box* menuju *box ground truth* terdekat (Nair et al., 2010). Sebuah contoh dari *anchor*, *predicted box*, dan *ground truth box* dapat dilihat pada Gambar 2.15.




Gambar 2.15 Contoh *Anchor*, *Box* Prediksi, dan *Ground Truth Box*

### 2.8.4 Intersection over Union (IoU)

Nilai *Intersection over Union* (IoU) menghitung luas area yang saling tumpang tindih dan membaginya dengan luas area total dari dua bounding box. IoU berguna untuk menilai seberapa baik *bounding box* yang diprediksi mencerminkan ukuran objek yang sebenarnya dalam citra (*ground truth*). IoU dapat dihitung menggunakan rumus sebagai berikut:

$$
IoU = \frac{\text{Area Irisan}}{\text{Area Gabungan}} \tag{2.8}
$$

## 2.9 *ResNet*-50

*ResNet*, atau Jaringan Residual, merupakan salah satu jenis model CNN yang dapat menangani lebih dari ratusan lapisan konvolusi. Model ini pertama kali diperkenalkan oleh Kaiming He dan rekan-rekannya (Ren et al. , 2017). Konsep utama yang diusulkan dalam *ResNet* adalah penggunaan pintasan untuk melewati blok lapisan konvolusi, yang membentuk apa yang disebut sebagai *residual block*. *Residual block* dapat dijelaskan sebagai berikut:

$$y_l(x, w) = f\big(h(x) + \mathcal{F}(x, w)\big) \tag{2.9}$$

dengan:

$x$ : *Input*
$h$ : Fungsi pintasan
$\mathcal{F}$ : Pemetaan yang dilakukan pada *convolutional layer* yang berurutan oleh satu blok atau lebih (dilewati)
$w$ : Bobot dari *convolutional layer*
$f$ : Fungsi aktivasi (*ReLU*)
$y_l$ : Pemetaan yang menggambarkan *residual block* $l$ sebagai fungsi

Fungsi pintasan umumnya berupa fungsi identitas atau lapisan konvolusi yang sangat sederhana ketika diperlukan kesesuaian dimensi. Fungsi pintasan bertujuan untuk mempercepat proses belajar pada pemetaan $\mathcal{F}$ sebagai gangguan dari input $x$, dimulai dari titik terdekat fungsi identitas. Jumlah lapisan dalam *ResNet* – 50 dapat dilihat pada Tabel 2. 2.

Tabel 2.2 *Layer ResNet – 50*

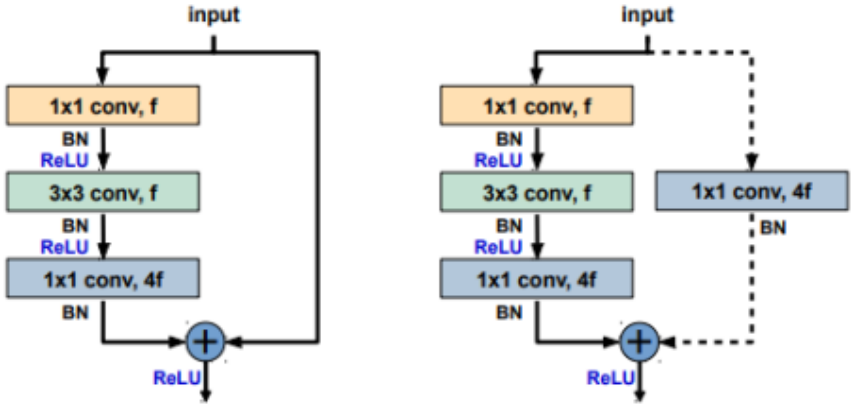
| Nama Layer | Ukuran Keluaran | Layer ResNet-50 |
|---|---|---|
| Conv1_x | $112 \times 112$ | $7 \times 7, 64, stride\ 2$ |
| Conv2_x | $56 \times 56$ | $3 \times 3\ max\ pool, stride\ 2$ |
| | | $\begin{bmatrix} 1 \times 1,64 \\ 3 \times 3,64 \\ 1 \times 1,256 \end{bmatrix} \times 3$ |
| Conv3_x | $28 \times 28$ | $\begin{bmatrix} 1 \times 1,128 \\ 3 \times 3,128 \\ 1 \times 1,512 \end{bmatrix} \times 4$ |
| Conv4_x | $14 \times 14$ | $\begin{bmatrix} 1 \times 1,256 \\ 3 \times 3,256 \\ 1 \times 1,1024 \end{bmatrix} \times 6$ |
| Conv5_x | $7 \times 7$ | $\begin{bmatrix} 1 \times 1,512 \\ 3 \times 3,512 \\ 1 \times 1,2048 \end{bmatrix} \times 3$ |
| - | $1 \times 1$ | $Average\ pool, 1000 - d\ fc, softmax$ |
| FLOPs | - | $3.8 \times 10\text{^}9$ |

Pada *ResNet* – 50, *residual block* yang paling dasar dikenal sebagai *bottleneck block*, terdiri dari tiga lapisan konvolusi yang saling berurutan dengan ukuran filter 1 $x$ 1, diikuti oleh 3 $x$ 3, dan diakhiri dengan lagi 1 $x$ 1.

Pintasan identitas dapat diterapkan secara langsung ketika dimensi pada *input* dan *output feature maps* sama. Namun, jika ada perubahan dimensi, dua opsi bisa dipertimbangkan, yakni:

1. Pintasan tetap melakukan pemetaan identitas sambil menambahkan *zero padding* untuk menambah dimensi filter (*depth*). Opsi ini tidak memerlukan parameter tambahan.
2. Menggunakan lapisan konvolusi 1 $x$ 1 untuk menyesuaikan dimensi. Opsi ini dikenal sebagai pintasan proyeksi.

Contoh penerapan kedua opsi ini ditampilkan pada Gambar 2.16.




Gambar 2.16 *Bottleneck Block* untuk *ResNet* – 50 (Kiri: Pintasan Identitas, Kanan: Pintasan Proyeksi)

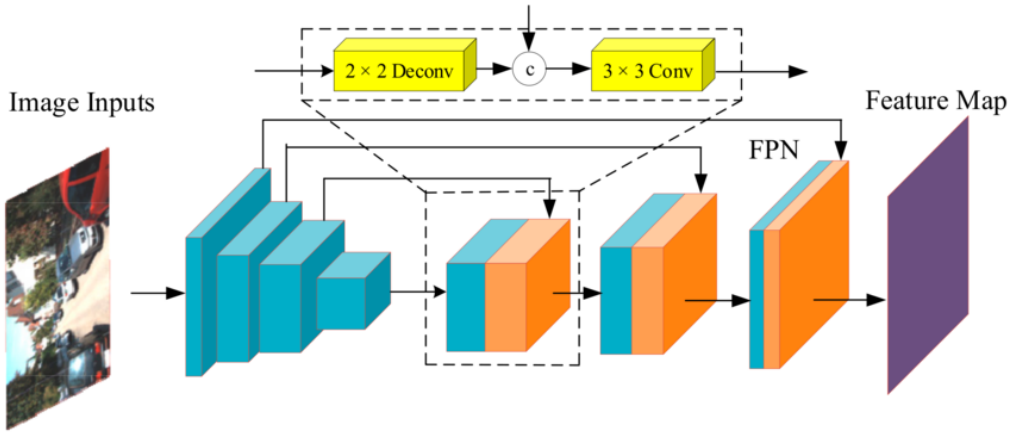
Arsitektur *ResNet* – 50 diakhiri dengan *average pooling layer* umum dan sebuah *fully connected layer* dengan 1000 dimensi menggunakan fungsi aktivasi *softmax*.

## 2.10 *Feature Pyramid Network* (*FPN*)

*Feature Pyramid Network* (FPN) merupakan arsitektur deep learning yang dirancang untuk meningkatkan kemampuan deteksi objek pada berbagai skala. Masalah utama dalam deteksi objek adalah variasi ukuran objek dalam sebuah citra. Objek yang berukuran kecil sering kali sulit dikenali karena informasi fitur yang terbatas. FPN hadir sebagai solusi dengan membangun piramida fitur yang bersifat multi-scale melalui kombinasi *bottom-up pathway* dan *top-down pathway* (Lin et al., 2017).

Pada *bottom-up pathway*, jaringan ekstraksi fitur seperti *ResNet* menghasilkan representasi hierarkis dari citra. Lapisan awal menyimpan detail spasial yang tinggi, sementara lapisan lebih dalam mengandung informasi semantik yang lebih kaya namun dengan resolusi rendah. FPN kemudian menambahkan *top-down pathway* yang melakukan *upsampling* pada fitur dengan resolusi rendah untuk digabungkan dengan fitur resolusi tinggi dari jalur bawah. Proses ini memungkinkan terbentuknya fitur yang konsisten di berbagai tingkat resolusi.

FPN terbukti meningkatkan performa deteksi objek pada berbagai ukuran. Pada *Faster* R-CNN, integrasi FPN memungkinkan *Region Proposal Network* (RPN) untuk menghasilkan *region proposal* yang lebih akurat terhadap objek kecil maupun besar. Hal ini menjadikan FPN sangat efektif dalam domain visi komputer seperti deteksi kebakaran, di mana objek seperti api dan asap sering kali memiliki ukuran dan bentuk yang bervariasi (Lin et al., 2017; Zhao et al., 2020). Arsitektur jaringan *Feature Pyramid Network* (FPN) ditunjukkan pada Gambar 2.17.

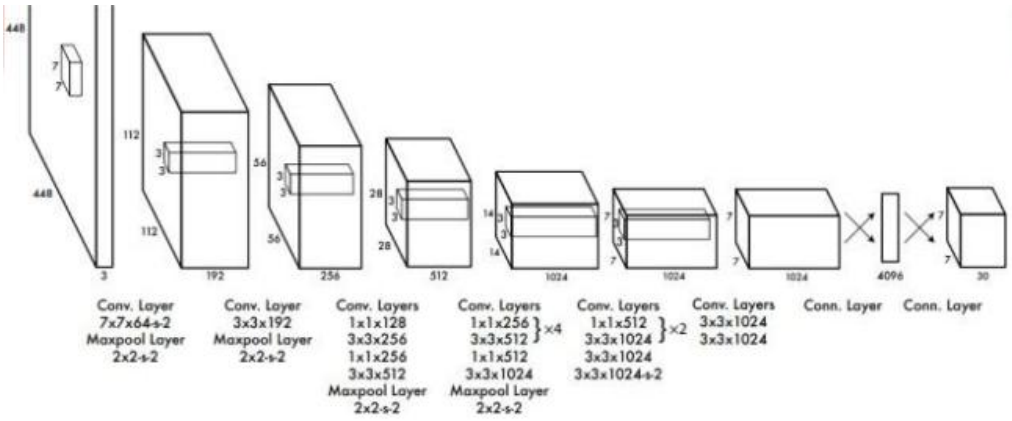



Gambar 2.17 Arsitektur *Feature Pyramid Network (FPN)*

## 2.11 *You Only Look Once (YOLO)*

YOLO merupakan *Real-time Object Detection* yang didasarkan pada *Convolutional Neural Network* dan menunjukkan hasil yang bagus pada pendeteksian objek dalam tingkat respons *real-time*. Dengan YOLO, deteksi objek dilakukan dengan melihat permasalahan tersebut sebagai masalah regresi untuk memisahkan secara spasial *bounding box* dan kelas probabilitas yang terkait terhadap *bounding box* tersebut (Diwan et al., 2022). Sebuah *neural network* memprediksi *bounding box* dan kelas prediksi langsung dari keseluruhan citra dari satu evaluasi seperti pada Gambar 2.18. Terdapat beberapa keunggulan dari penggunaan YOLO yakni, yang pertama proses yang terhitung cepat. Kedua, YOLO meminimalisir kesalahan dalam pendeteksian latar belakang dan juga kelas karena melihat gambar secara keseluruhan. Ketiga, YOLO mempelajari representasi objek yang dapat digeneralisir.

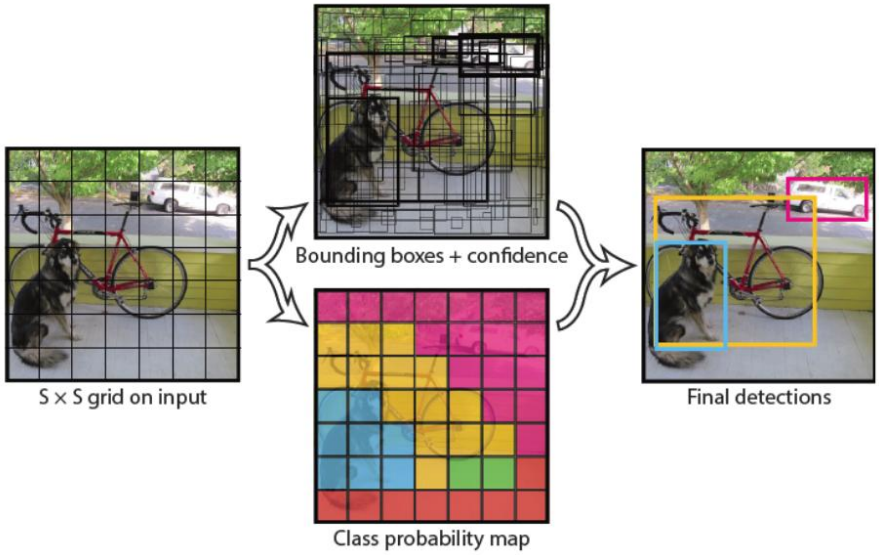
Jaringan deteksi YOLO memiliki 24 lapisan konvolusi (convolutional layer) yang diikuti oleh 2 lapisan yang terhubung penuh (fully connected layer). Beberapa lapisan konvolusi menggunakan lapisan reduksi 1x1 sebagai alternative dalam mengurangi kedalaman feature maps yang diikuti oleh 3x3 lapisan konvolusional (convolutional layer) seperti pada Gambar 2.18.




Gambar 2.18 Arsitektur YOLO

YOLO bekerja dengan menyatukan komponen yang terpisah dari objek yang terdeteksi menjadi kesatuan *neural network* dengan memanfaatkan fitur dari gambar secara keseluruhan untuk memprediksi setiap *bounding box* dalam semua kelas objek dari sebuah gambar secara bersamaan (Diwan et al., 2022). YOLO memungkinkan pelatihan secara keseluruhan dalam kondisi *real-time* dengan tetap mempertahankan tingkat prediksi. Ketika YOLO menerima

input gambar, YOLO membagi gambar tersebut menjadi S × S petak (grid). Setiap sel petak yang terdapat pusat objek didalamnya bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel petak juga bertugas memprediksi *bounding box* dan nilai keyakinan untuk kotak-kotak tersebut. Nilai keyakinan ini mencerminkan seberapa yakin kotak itu berisi objek beserta keakuratannya. Cara kerja YOLO ditunjukkan pada Gambar 2.19.

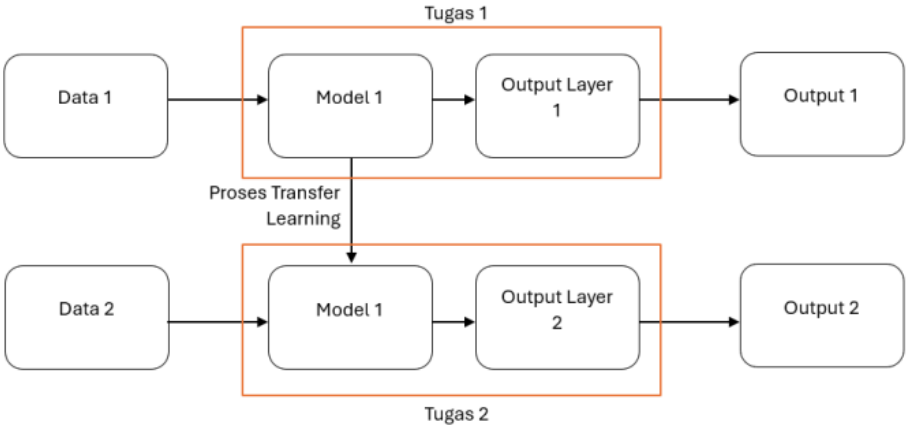



Gambar 2.19 Cara kerja YOLO

## 2.12 *Transfer Learning*

Suatu arsitektur CNN yang ada dan sudah dilatih dapat digunakan pada data yang berbeda. Maka daripada membuat arsitektur CNN baru dan melatihnya dari awal, model CNN yang sudah dilatih tadi diadaptasi untuk data yang baru dengan teknik yang disebut *transfer learning*. *Transfer learning* adalah proses menyalin pengetahuan dari jaringan yang sudah ada ke jaringan yang baru untuk menyelesaikan masalah serupa (Sewak, Karim, & Pujari, 2018). Teknik ini diimplementasikan untuk mempercepat proses *training* dan meningkatkan performa model (Patterson & Gibson, 2017).

Suatu model yang telah dilatih disebut *pre-trained* model. *Pre-trained* model dibentuk oleh seseorang/organisasi yang memiliki akses terhadap dataset skala besar dan sumber daya yang mumpuni dalam mengolah model. Contoh *pre-trained* model adalah Darknet, Xception, VGGNet, ResNet, Inception, MobileNet, DenseNet, NASNet dan GoogleNet. *Pre-trained* model tersebut dilatih pada dataset skala besar seperti *ImageNet*, yaitu dataset gambar yang disusun sesuai hierarki *WordNet*, mempunyai gambar sebanyak 14.197.122 buah dan 1,2 juta gambar yang terlabeli (ImageNet, 2016). Dua penggunaan *pre-trained* model yaitu, *Feature extractor* dan *Fine-tuning existing model. Feature extractor* yaitu penghapusan *Fully-connected layer* terakhir dan menggunakan sisanya sebagai *fixed feature extractor* untuk dataset yang lebih kecil, sedangkan *Fine-tuning existing model* yaitu melatih ulang *pre-trained* model dengan data baru dan melakukan *tuning* parameter (*weights*) dengan *backpropagation*. Diagram *transfer learning* ditunjukkan pada Gambar 2.20.




Gambar 2.20 Diagram *Transfer Learning*

## 2.13 SGD

*Optimizer Stochastic Gradient Descent* (SGD) adalah salah satu metode optimisasi dasar yang paling banyak digunakan dalam machine learning, khususnya untuk pelatihan model-model deep learning (Bottou et al., 2020). SGD adalah versi stochastic dari gradient descent yang melakukan pembaruan parameter menggunakan subset data acak. Rumus dasar SGD adalah sebagai berikut:

$$\theta_{t+1} = \theta_t - \frac{\alpha\, \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \alpha\, \omega d\, \theta_t \tag{2.10}$$

Di mana $\theta_t$ adalah parameter yang akan dioptimalkan, $\eta$ adalah learning rate dan $g_t$ adalah gradien dari fungsi kerugian yang diestimasi pada subset data.

## 2.14 AdamW

Optimizer AdamW adalah sebuah variasi dari optimizer Adam yang memperkenalkan teknik Weight Decay secara lebih eksplisit dalam proses optimisasi (Loshchilov et al., 2019). AdamW memodifikasi Adam dengan memisahkan weight decay dari proses update parameter. Ini mengatasi beberapa kekurangan Adam dalam mengelola regularisasi weight decay. Rumus AdamW serupa dengan Adam, tetapi weight decay diterapkan langsung ke parameter sebelum pembaruan berikutnya:

$$\theta_{t+1} = \theta_t - \eta g_t \tag{2.10}$$

Dimana $\omega d$ adalah weight decay yang diterapkan pada parameter.

## 2.15 Evaluasi Performa

Nilai evaluasi performa yang biasanya digunakan dalam penelitian adalah *Confusion Matrix*. *Confusion Matrix* berguna untuk menilai seberapa baik klasifikasi dalam model. Kualitas klasifikasi dapat diukur dengan menghitung nilai *precision* dan *recall* untuk setiap kelas pada setiap model. Dalam konteks klasifikasi, hasil dapat dikelompokkan menjadi empat kategori berdasarkan kombinasi kelas kebenaran dasar dan kelas yang diprediksi, yaitu: *true positive* (TP), *false positive* (FP), *true negative* (TN), dan *false negative* (FN). Penjelasan mengenai empat kategori dalam matriks kebingungan untuk multi-kelas ditampilkan pada Tabel 2.3.

Tabel 2.3 Confusion Matrix

| | | Nilai Sebenarnya | |
|---|---|---|---|
| | | TRUE | FALSE |
| Nilai Prediksi | TRUE | $TP$ (True Positive) Terdapat objek terdeteksi benar | $FP$ (False Positive) Terdapat objek terdeteksi salah |
| | FALSE | $FN$ (False Negative) Tidak terdapat objek terdeteksi salah | $TN$ (True Negative) Tidak terdapat objek terdeteksi benar |

Berdasarkan Tabel 2.3, *True Positive* (TP) mengacu pada jumlah prediksi di mana pengklasifikasi kelas positif pada *ground truth* sebagai prediksi positif dengan benar. *False Positive* (FP) mengacu pada jumlah prediksi di mana klasifikasi salah memprediksi kelas negatif pada *ground truth* sebagai prediksi positif, *True Negative* (TN) mengacu pada jumlah prediksi di mana pengklasifikasi kelas negatif pada *ground truth* dengan benar sebagai prediksi negatif, dan *False Negatif* (FN) mengacu pada jumlah prediksi di mana pengklasifikasi salah memprediksi kelas positif pada *ground truth* sebagai prediksi negatif. Dari hasil *confusion matrix* dapat dihitung *precision* dan *recall* dengan persamaan sebagai berikut :

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

$$F1score = \frac{(2 \times precision \times recall)}{(precision + recall)} \tag{2.15}$$

*Precision* menunjukkan seberapa baik model dapat mengenali objek yang relevan sebagai persentase dari prediksi yang benar. *Recall* adalah kemampuan model dalam mendeteksi semua objek yang relevan, dihitung sebagai persentase dari positif benar yang dapat dikenali dari semua kebenaran dasar. Setelah mendapatkan nilai *precision* dan *recall*, sebuah kurva yang disebut kurva P-R dapat disusun, dengan *recall* pada sumbu horizontal dan *precision* pada sumbu vertikal.

Nilai evaluasi kedua yang digunakan adalah mAP. Nilai mAP merupakan salah satu indikator evaluasi yang paling umum digunakan dalam tugas deteksi objek untuk membandingkan kinerja suatu model dengan model lainnya. *Mean Average Precision* (mAP) adalah rata-rata dari nilai *Average Precision*. *Average precision* diperoleh dari setiap nilai presisi terhadap item relevan yang dihasilkan, sementara item relevan yang tidak dihasilkan

oleh sistem diberi nilai 0. Perhitungan nilai presisi untuk *average precision* mempertimbangkan urutan item yang disajikan oleh sistem, sehingga presisi diberikan untuk setiap item yang dihasilkan. Persamaan di bawah ini adalah rumus untuk menghitung nilai mean average precision (Manning, Raghavan, dan Schutze, 2009).

$$mAP = \frac{1}{c}\sum_{i=1}^{c}\frac{1}{m_i}\sum_{k=1}^{m_i} p(R_{ik}) \tag{2.16}$$

Cara lain untuk mendapatkan nilai mAP menggunakan *Precision Recall Curve*, yaitu sebagai berikut:

$$mAP = \frac{1}{c}\sum_{i=1}^{c} AP_i \tag{2.17}$$

*Mean average precision* (mAP) adalah rata-rata nilai AP di semua kelas. Perhitungan nilai *average precision* (AP) dapat dilakukan dengan 2 metode interpolasi yaitu 11-*point interpolation* dan *allpoint interpolation*. Berikut merupakan persamaan dari metode 11- *point interpolation*:

$$AP = \frac{1}{11}\sum_{r\in\{0;0,1;\ldots,1\}} p_{\text{interp}}(r) \tag{2.18}$$

$$p_{\text{interp}} = \max_{\tilde{r}:\tilde{r}\geq r_n} p(\tilde{r}) \tag{2.19}$$

Interpolasi 11 titik meringkas bentuk *precision recall curve* dengan rata-rata presisi pada sebelas tingkat *recall* yang berjarak sama yaitu 0; 0,1; 0,2, ..., 1. Nilai presisi interpolasi didapatkan dengan mengambil presisi maksimum yang nilai *recall*nya lebih besar dari nilai *recall* saat ini. Untuk metode yang kedua yaitu *all-point interpolation* disajikan pada persamaan berikut:

$$AP = \sum_{n=0} (r_{n+1} - r_n) p_{\text{interp}}(r_{n+1}) \tag{2.20}$$

$$p_{\text{interp}}(r_{n+1}) = \max_{\tilde{r}:\tilde{r}\geq r_n+1} p(\tilde{r}) \tag{2.21}$$

Pada metode ini, AP diperoleh dengan menginterpolasi presisi di setiap titik, mengambil presisi maksimum yang nilai *recall*nya lebih besar atau sama dengan $r_{n+1}$.

Keterangan:

| | |
|---|---|
| $c$ | : Jumlah kelas |
| $R$ | : item relevan yang dihasilkan oleh system |
| $m$ | : jumlah item relevan yang dihasilkan pada suatu kelas |
| $p$ | : presisi |
| $p_{\text{interp}}$ | : interpolasi presisi |
| $r$ | : recall |

$p(\tilde{r})$ : perhitungan presisi pada recall r

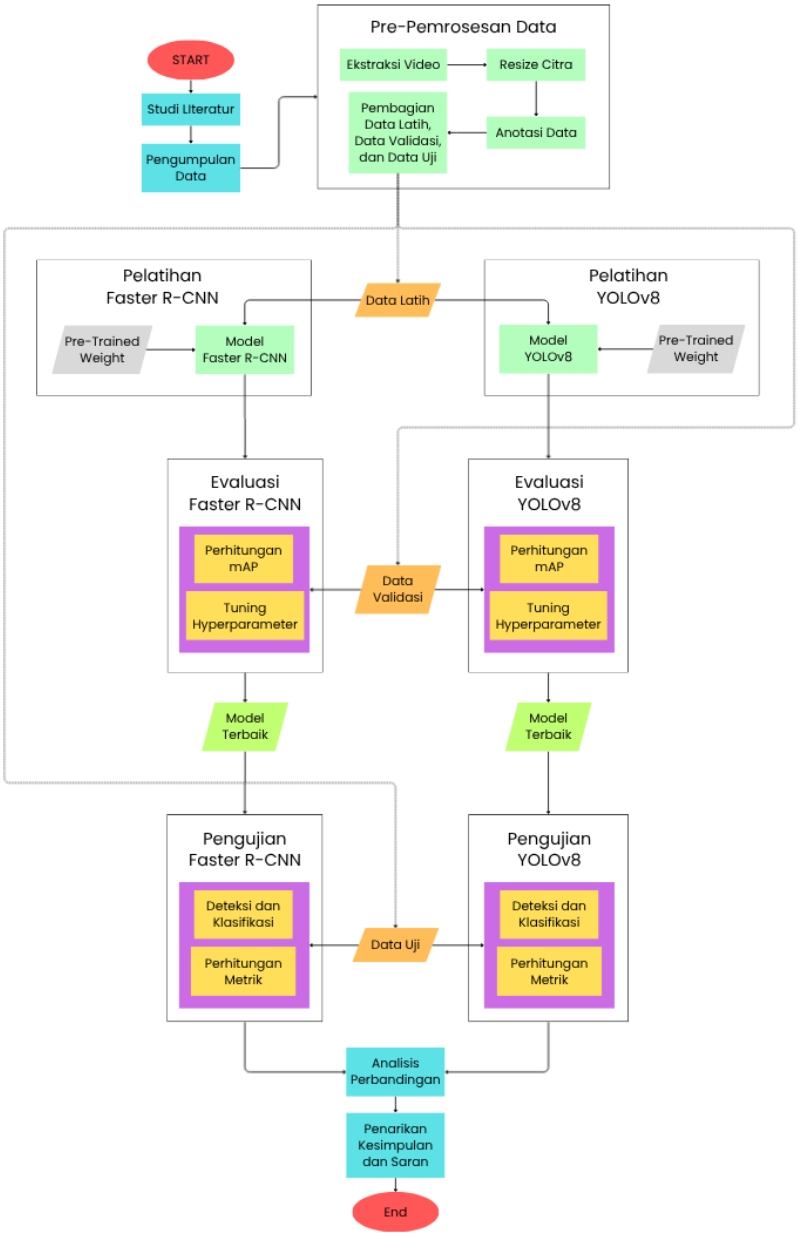
Perhitungan *average precision* (AP) dalam deteksi objek dapat diperoleh dengan mendapatkan nilai IoU terlebih dahulu. IoU akan digunakan untuk menentukan apakah kotak pembatas yang diprediksi adalah *True Prositive* (TP), *False Positive* (FP) atau *False Negative* (FN). *True Negative* (TN) tidak dievaluasi karena setiap gambar diasumsikan memiliki objek di dalamnya. Prediksi didefinisikan sebagai TP jika IoU ≥ 0,5. Ada 2 skenario dalam mendefinisikan FP, pertama apabila klasifikasi kelas benar namun IoU < 0,5 dan kedua apabila terdapat duplikat *bounding box* prediksi. Selanjutnya ada 2 skenario dalam mendefinisikan FN, yaitu, pertama apabila IoU ≥ 0,5 namun salah klasifikasi dan kedua apabila tidak dideteksi objek yang seharusnya terdapat *ground truth* objek. Hasil deteksi diurutkan sesuai *confidence* yang tertinggi pada perhitungan *Avearage Precision* (AP).

# BAB III
# METODOLOGI PENELITIAN

# BAB III
# METODOLOGI PENELITIAN

## 3.1 Diagram Alir

Pada subbab ini dijelaskan langkah-langkah dalam melakukan proses penelitian dan penerapan model *Faster R-CNN* dan *YOLOv8* untuk deteksi api dan asap pada data video. Secara singkat, langkah-langkah tahapan penelitian ditunjukkan pada Gambar 3.1.




Gambar 3.1 Diagram Alir Penelitian

### 3.1.1 Studi Literatur

Literatur yang dibaca terkait dengan penelitian model *Pre-trained Faster* R-CNN dan *YOLOv8* yang diterapkan untuk menyelesaikan Tugas Akhir yaitu," PERBANDINGAN FASTER R-CNN DAN YOLOv8 UNTUK DETEKSI API DAN ASAP PADA DATA VIDEO". Literatur yang digunakan adalah penelitian terdahulu, jurnal ilmiah, buku, serta

artikel. Sehingga didapatkan informasi sebagai dasar pertimbangan pembahasan dalam penelitian ini.

### 3.1.2 Pengumpulan Data

Data yang digunakan pada penelitian ini terdiri dari data primer dan data sekunder. Data primer yang digunakan adalah kumpulan video dari kamera CCTV di sekitar rumah penulis. Data sekunder yang digunakan berasal dari *https://universe.roboflow.com/nghia-tdxoy/fire-smoke-wucvr* digunakan oleh (Kim, B. et al., 2019) (https://doi.org/10.3390/app9142862) tentang *FDS (Fire and Smoke detection)* merupakan dataset publik yang terdiri dari 31 klip video *fire* dan *smoke* dalam kondisi berbeda. Dataset ini juga memiliki resolusi, durasi waktu, dan *frame rate* (fps) yang berbeda pada setiap video. Citra diberikan anotasi *fire dan smoke*, kemudian semua anotasi disimpan dalam format COCO (.json) dan YOLO (.txt).

### 3.1.3 Pra-Pemrosesan Data

Tahapan-tahapan yang akan dilakukan pada pra-pemrosesan data yaitu sebagai berikut:

1. Ekstrasi Video
   Proses ini adalah mengubah data video menjadi kumpulan citra. Hasil citra tersebut langsung dikelompokkan berdasarkan kegunaannya yaitu untuk melatih dan menguji model.

2. *Resize* Citra
   Proses ini adalah memperkecil ukuran masing-masing citra. Tujuannya untuk efisiensi dan mempercepat proses pelatihan dan pengujian model.

3. Anotasi Data
   Pada proses ini dibuat anotasi *bounding box* untuk objek *fire* dan *smoke* pada data primer dan digunakan anotasi yang telah disediakan pada data sekunder, kemudian diubah formatnya sedemikian hingga dapat digunakan untuk melatih model *Faster* R-CNN dan *YOLOv8*. Format anotasi yang digunakan pada model *Faster* R-CNN berupa COCO (.json) dan *YOLOv8* berupa YOLO (.txt).

4. Pembagian Data Latih, Data Validasi, dan Data Uji
   Data video yang telah siap, selanjutnya dibagi menjadi data latih 70%, data validasi 20%, dan data uji 10%. Data latih digunakan dalam pelatihan model dan data validasi digunakan untuk evaluasi kinerja model berdasarkan metrik yang digunakan. Data uji digunakan untuk menguji model terbaik hasil dari pelatihan dan evaluasi lalu diukur berdasarkan keakuratannya mendeteksi api dan asap.

### 3.1.4 Pelatihan Faster R-CNN

Pada tahap ini dilakukan proses pelatihan model Faster R-CNN ResNet-50 Feature Pyramid Network (FPN) menggunakan framework Detectron2. Pelatihan menerapkan metode transfer learning dengan memanfaatkan pre-trained weights dari dataset COCO sehingga model telah memiliki kemampuan dasar dalam mengenali berbagai karakteristik objek. Penggunaan bobot pralatih bertujuan mempercepat proses konvergensi, meningkatkan efisiensi komputasi, serta mengurangi kebutuhan data pelatihan dibandingkan dengan pelatihan dari awal (*training from scratch*).

Proses pelatihan diawali dengan memasukkan data latih ke dalam model. Citra masukan terlebih dahulu diproses oleh ResNet-50 sebagai *backbone* untuk mengekstraksi fitur visual. Selanjutnya, Feature Pyramid Network (FPN) membangun representasi fitur pada berbagai skala sehingga model mampu mendeteksi objek api dan asap dengan ukuran yang beragam. Feature map yang dihasilkan kemudian diproses oleh Region Proposal Network (RPN) untuk menghasilkan kandidat wilayah objek (*region proposals*). Setiap kandidat wilayah selanjutnya diproses menggunakan RoI Pooling, kemudian diteruskan ke RoI Head untuk melakukan klasifikasi objek (*fire*, *smoke*, atau *background*) sekaligus memperbaiki koordinat *bounding box* melalui proses regresi.

Selama pelatihan, bobot model diperbarui secara iteratif menggunakan algoritma optimasi hingga diperoleh model yang mampu mengenali pola api dan asap pada data latih. Hasil dari tahap ini berupa model Faster R-CNN yang selanjutnya akan dievaluasi pada data validasi.

### 3.1.5 Pelatihan YOLOv8

Tahap pelatihan model pembanding dilakukan menggunakan YOLOv8m melalui framework Ultralytics dengan menerapkan pendekatan transfer learning menggunakan pre-trained weights dari dataset COCO. Pendekatan ini memungkinkan model memanfaatkan pengetahuan yang telah dipelajari sebelumnya sehingga proses pelatihan menjadi lebih cepat dan efisien.

YOLOv8 merupakan metode deteksi objek berbasis one-stage detector, di mana proses ekstraksi fitur, klasifikasi objek, dan prediksi *bounding box* dilakukan secara bersamaan dalam satu jaringan. Pada bagian backbone, model menggunakan Modified CSPDarknet dengan blok C2f untuk menghasilkan representasi fitur yang lebih kaya. Selanjutnya, fitur diproses pada bagian neck menggunakan Path Aggregation Network (PANet) untuk menggabungkan informasi dari berbagai tingkat resolusi sehingga objek dengan ukuran yang berbeda tetap dapat dideteksi. Pada bagian head, YOLOv8 menerapkan pendekatan anchor-free dan decoupled head sehingga proses klasifikasi objek dan regresi *bounding box* dilakukan secara terpisah.

Selama pelatihan, bobot jaringan diperbarui secara bertahap menggunakan algoritma optimasi hingga diperoleh model yang mampu mendeteksi objek api dan asap pada data latih. Model hasil pelatihan kemudian digunakan pada tahap evaluasi.

### 3.1.6 Evaluasi Faster R-CNN

Tahap evaluasi dilakukan menggunakan data validasi yang tidak digunakan selama proses pelatihan. Tujuan evaluasi adalah untuk mengukur kemampuan model dalam melakukan deteksi objek pada data yang belum pernah dipelajari sekaligus memantau kemampuan generalisasi model.

Evaluasi dilakukan dengan menghitung nilai mean Average Precision (mAP) yang terdiri atas mAP@0.5 dan mAP@0.5:0.95. Nilai mAP digunakan sebagai indikator utama dalam menilai kualitas deteksi objek. Apabila hasil evaluasi belum mencapai performa yang diharapkan, dilakukan hyperparameter tuning dengan menyesuaikan parameter seperti *learning rate*, *batch size*, *optimizer*, maupun jumlah *epoch*. Proses pelatihan dan evaluasi dilakukan secara berulang hingga diperoleh konfigurasi parameter yang memberikan performa terbaik. Model dengan nilai mAP tertinggi kemudian disimpan sebagai model terbaik (best model) untuk digunakan pada tahap pengujian.

### 3.1.7 Evaluasi YOLOv8

Evaluasi YOLOv8 dilakukan menggunakan data validasi untuk mengukur kemampuan model dalam mendeteksi objek api dan asap pada data yang tidak digunakan selama pelatihan.

Performa model dievaluasi menggunakan metrik mAP@0.5 dan mAP@0.5:0.95. Berdasarkan hasil evaluasi tersebut dilakukan hyperparameter tuning apabila performa model belum optimal. Penyesuaian dilakukan terhadap beberapa parameter pelatihan, seperti *learning rate*, *batch size*, *optimizer*, dan jumlah *epoch*. Tahapan pelatihan dan evaluasi dilakukan secara berulang hingga diperoleh model dengan performa terbaik pada data validasi. Model tersebut kemudian disimpan sebagai best model untuk digunakan pada tahap pengujian.

### 3.1.8 Pengujian Faster R-CNN

Tahap pengujian dilakukan menggunakan best model Faster R-CNN terhadap data video uji yang belum pernah digunakan pada proses pelatihan maupun evaluasi. Video diproses secara frame by frame menggunakan framework Detectron2.

Pada setiap frame, model melakukan deteksi dan klasifikasi objek api maupun asap. Hasil deteksi ditampilkan dalam bentuk bounding box, label kelas, dan confidence score. Selanjutnya dilakukan perhitungan metrik evaluasi berdasarkan hasil deteksi, meliputi precision, recall, F1-score, dan Frames Per Second (FPS) untuk mengukur tingkat akurasi serta kecepatan inferensi model.

### 3.1.9 Pengujian YOLOv8

Tahap pengujian YOLOv8 dilakukan menggunakan best model terhadap data video uji yang belum pernah digunakan sebelumnya. Video diproses secara frame by frame menggunakan framework Ultralytics.

Pada setiap frame, model melakukan deteksi dan klasifikasi objek api maupun asap, kemudian menampilkan hasil berupa bounding box, label kelas, dan confidence score. Selanjutnya dilakukan perhitungan metrik evaluasi, yaitu precision, recall, F1-score, dan Frames Per Second (FPS). Nilai-nilai tersebut digunakan sebagai dasar untuk membandingkan performa YOLOv8 dengan Faster R-CNN.

### 3.1.10 Analisis Perbandingan

Tahap analisis perbandingan dilakukan dengan membandingkan performa Faster R-CNN dan YOLOv8 berdasarkan hasil pengujian pada data video. Perbandingan dilakukan menggunakan metrik precision, recall, F1-score, dan Frames Per Second (FPS). Selain membandingkan nilai metrik kuantitatif, analisis juga dilakukan terhadap kualitas visual hasil deteksi, seperti ketepatan klasifikasi objek, akurasi penempatan *bounding box*, serta kemampuan kedua model dalam mendeteksi objek api dan asap pada berbagai kondisi video. Hasil analisis ini digunakan untuk menentukan model yang memiliki performa paling baik dan paling sesuai untuk diterapkan pada sistem deteksi dini kebakaran berbasis video.

### 3.1.11 Penarikan Kesimpulan dan Saran

Penarikan kesimpulan merupakan hasil akhir yang didapatkan berdasarkan hasil evaluasi serta analisis yang telah dilakukan. Selain itu, akan disampaikan saran untuk penelitian selanjutnya.

# BAB IV
# PERANCANGAN DAN IMPLEMENTASI

# BAB IV
# PERANCANGAN DAN IMPLEMENTASI

## 4.1 Perancangan Data

Perancangan data adalah tahap untuk mempersiapkan *dataset* yang digunakan dalam proses pelatihan hingga pengujian model agar dapat ditarik kesimpulan terhadap penelitian ini. Tahap persiapan data dimulai dari pengambilan data dan pra-pemrosesan data.

### 4.1.1 Pengambilan Data

Pada penelitian ini digunakan dua jenis data, diantaranya yaitu data primer dan data sekunder. Data primer yang digunakan adalah kumpulan video dari kamera CCTV di sekitar rumah penulis. Data sekunder yang digunakan berasal dari *https://universe.roboflow.com/nghia-tdxoy/fire-smoke-wucvr* digunakan oleh (Kim, B. et al., 2019) (https://doi.org/10.3390/app9142862) tentang *FDS (Fire and Smoke detection)* merupakan dataset publik yang terdiri dari video *fire* dan *smoke* dalam kondisi berbeda. Dataset ini juga memiliki resolusi, durasi waktu, dan *frame rate* (fps) yang berbeda pada setiap video. Citra diberikan anotasi *fire dan smoke*, kemudian semua anotasi disimpan dalam format COCO (.json) dan YOLO (.txt).

Total data video yang berhasil diambil adalah 6 video yang disajikan dalam Tabel 4.1.

Tabel 4.1 Jumlah Dataset yang Digunakan

| No | Nama Video | Durasi | Pembagian Data | Jumlah Frame |
|---|---|---|---|---|
| 1 | yt-Ew0cd2TRlgo | 3 menit 9 detik | Latih | 396 |
| | | | Validasi | 113 |
| | | | Uji | 58 |
| 2 | RobotCam-Day-1_mp4 | 1 menit 27 detik | Latih | 183 |
| | | | Validasi | 52 |
| | | | Uji | 27 |
| 3 | RobotStore-Day-1_mp4 | 1 menit 21 detik | Latih | 170 |
| | | | Validasi | 48 |
| | | | Uji | 26 |
| 4 | yt-Le6KNI9YsH0 | 1 menit 30 detik | Latih | 189 |
| | | | Validasi | 54 |
| | | | Uji | 27 |
| 5 | RobotCam-Static-1-Day-1_mp4 | 49 detik | Latih | 104 |
| | | | Validasi | 29 |
| | | | Uji | 16 |
| 6 | RobotStore-Part-2-Day-2_mp4 | 1 menit 2 detik | Latih | 130 |
| | | | Validasi | 37 |
| | | | Uji | 20 |
| **Total Data Latih** | | | | **1172** |
| **Total Data Valid** | | | | **333** |
| **Total Data Uji** | | | | **174** |

### 4.1.2 Pra-Pemrosesan Data

Pada tahap ini dilakukan pengolahan data dengan tujuan mempersiapkan data untuk melatih model *Faster* R-CNN dan *YOLOv8*. Proses pengolahan data memiliki beberapa tahap sebagai berikut:

1. Ekstraksi video

   Pada tahap ini, video dibaca dengan bantuan *library Python OpenCV*. Proses membaca video dengan *OpenCV* yaitu video menjadi sebuah iterator sehingga dapat diiterasi setiap frame dalam video tersebut. Selanjutnya, setiap frame tersebut disimpan dalam format JPG untuk masuk tahap selanjutnya. Citra tersebut kemudian dikelompokkan menjadi data latih dan data uji untuk model *Faster* R-CNN dan *YOLOv8*.

2. *Resize* citra

   Pada tahap ini, citra yang diperoleh dari tahap sebelumnya dikecilkan ukurannya menjadi resolusi 640x640. Hal ini dilakukan untuk menyesuaikan data latih dengan ukuran *input* 640x640 agar proses pelatihan dapat berlangsung lebih cepat. Metode *resize* citra yang digunakan pada penelitian ini adalah *interpolasi bilinear*. Metode ini melakukan *resize* dengan cara mencari nilai piksel baru di antara piksel-piksel yang ada dalam citra asli dengan menghitung bobot berdasarkan jarak relatif dari lokasi piksel baru terhadap piksel-piksel yang ada. Langkah melakukan *interpolasi bilinear* pada suatu titik $P(y, x)$ yang terletak didalam bidang yang dibatasi oleh 4 titik yaitu $Q_{11}(y_1, x_1)$, $Q_{12}(y_1, x_2)$, $Q_{21}(y_2, x_1)$, dan $Q_{22}(y_2, x_2)$ seperti pada Gambar 4.1 adalah

| | $x_1$ | $x$ | $x_2$ |
|---|---|---|---|
| $y_1$ | $Q_{11}$ | $R_1$ | $Q_{12}$ |
| $y$ | | $P$ | |
| $y_2$ | $Q_{21}$ | $R_2$ | $Q_{22}$ |

Gambar 4.1 *Interpolasi Bilinear* Secara Umum.

(a) Cari *interpolar linear* pada $R_1(y_1, x)$ dengan menggunakan titik $Q_{11}(y_1, x_1)$ dan $Q_{12}(y_1, x_2)$.

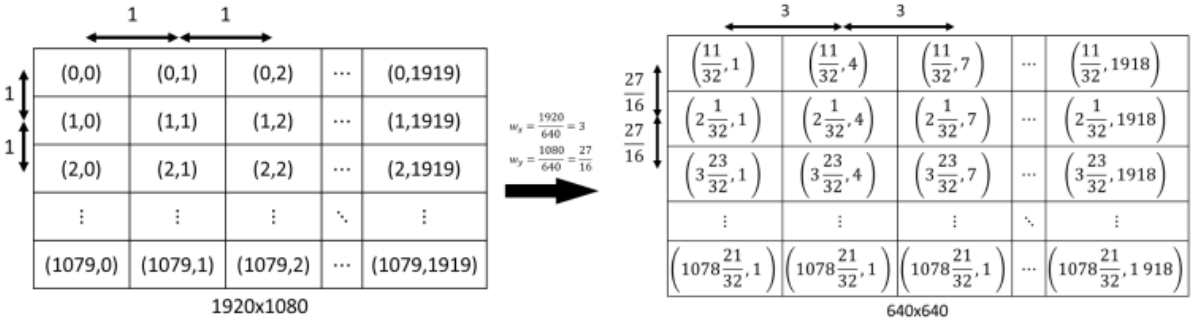
$$R_1 = \frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{12} \tag{4.1}$$

(b) Cari *interpolasi linear* pada $R_2(y_2, x)$ dengan menggunakan titik $Q_{21}(y_2, x_1)$ dan $Q_{22}(y_2, x_2)$.

$$R_2 = \frac{x_2 - x}{x_2 - x_1} Q_{21} + \frac{x - x_1}{x_2 - x_1} Q_{22} \tag{4.2}$$

(c) Cari *interpolasi linear* pada $P(y,x)$ dengan menggunakan titik $R_1(y_1,x)$ dan $R_2(y_2,x)$.

$$P = \frac{y_2 - y}{y_2 - y_1} R_1 + \frac{y - y_1}{y_2 - y_1} R_2 \tag{4.3}$$




Gambar 4.2 Proses *Resize* dengan *Interpolasi Bilinear*

Selanjutnya diberikan contoh proses melakukan *resize* dengan *interpolasi bilinear* pada citra ukuran 1920x1080 menjadi 640x640 yang diilustrasikan pada Gambar 4.2. Cara mendapatkan titik-titik baru setelah *resize*, dapat dilakukan dengan menggunakan barisan aritmatika dimana untuk baris pada citra memiliki elemen pertama $a_{baris} = \frac{11}{32}$ dan beda $b_{baris} = \frac{27}{16}$ sedangkan untuk kolom pada citra memiliki elemen pertama $a_{kolom} = 1$ dengan beda $b_{kolom} = 3$. Jadi untuk piksel ke (640,640) yang baru, diperoleh seperti berikut:

Barisan Aritmatika: $U_n = a + b \times (n - 1)$

Baris 640: $U_{640} = a_{baris} + b_{baris} \times (640 - 1) = \frac{11}{32} + \frac{27}{16} \times 639 = \frac{34517}{32} = 1078\frac{21}{32}$

Kolom 640: $U_{640} = a_{kolom} + b_{kolom} \times (640 - 1) = 1 + 3 \times 639 = 1918$

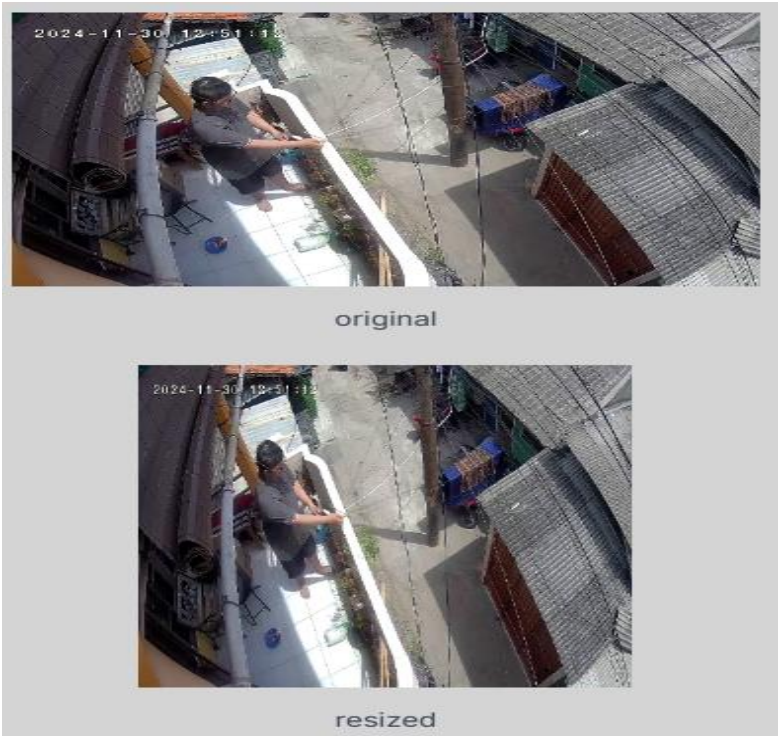
Sebagai contoh dihitung hasil piksel *interpolasi* di titik $\left(1078\frac{21}{32},1\right)$ dengan citra yang digunakan memiliki nilai piksel seperti berikut:

$$\begin{matrix} & 0 & 1 \\ 1078 & 74 & 74 \\ 1079 & 76 & 76 \end{matrix}$$

Dalam menghitung titik tersebut hanya diperlukan *interpolasi* antara dua baris yaitu antara baris 1078 dan baris 1079 dikarenakan kolom tepat pada 1. *Interpolasi linear* dari titik (1078,1) dan titik (1079,1) untuk titik $\left(1078\frac{21}{32},1\right)$ dihitung dengan rumus (4.3) seperti di bawah ini:

$$P\left(1078\frac{21}{32},1\right) = \frac{1079 - 1078\frac{21}{32}}{1079 - 1078} R_1(1078,1) + \frac{1078\frac{21}{32} - 1028}{1079 - 1078} R_2(1079,1)$$
$$= \frac{11}{32} \times 74 + \frac{21}{32} \times 76 = \frac{1205}{16} = 75.3125 \approx 75$$

Jadi nilai piksel baru setelah *resize* pada titik $\left(1078\frac{21}{32}, 1\right)$ adalah 75. Contoh visualisasi hasil *resize* citra ditunjukkan pada Gambar 4.3.




Gambar 4.3 *Resize* Citra

3. Anotasi bounding box

Pada tahap ini, citra yang telah diubah ukurannya diberikan anotasi bounding box pada setiap objek *fire* dan *smoke* yang terdapat pada masing-masing citra. Anotasi ini digunakan sebagai label pelatihan model *Faster* R-CNN dan *YOLOv8* agar dapat mendeteksi letak objek api dan asap pada suatu citra. Anotasi *bounding box* dataset FDS sudah tersedia, namun format yang digunakan berbeda dengan format anotasi yang digunakan untuk melatih *Faster* R-CNN dan *YOLOv8*. Pada dataset FDS, format anotasi *bounding box* YOLOv8 $[class, x_{center}, y_{center}, w, h]$ sedangkan Faster R-CNN Resnet-50 FPN lebih cocok menggunakan format COCO json $[x_{min}, y_{min}, width, height]$. Format anotasi tersebut dapat dirubah dengan menggunakan perhitungan berikut:

Misalkan ukuran gambar: W = lebar citra dan H = tinggi citra

YOLO (*normalized*): $x = x_{center},\ y = y_{center},\ w = width,\ h = height$

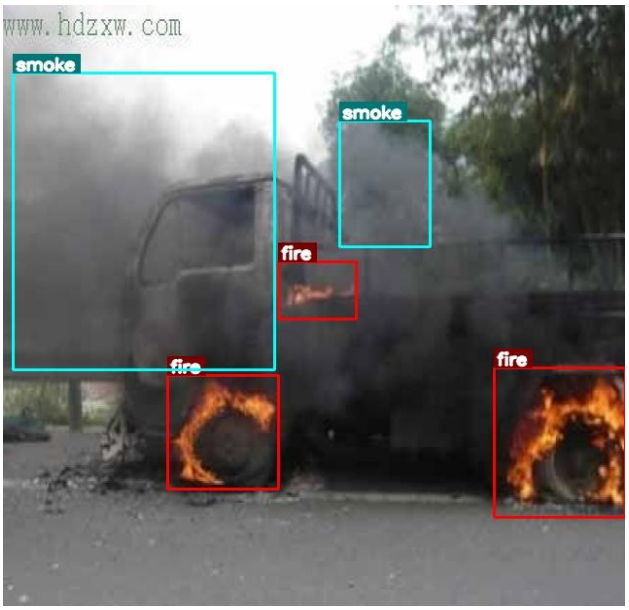
Denormalisasi:

$$x_{center} = x \times W$$
$$y_{center} = y \times H$$
$$w_{box} = w \times W$$
$$h_{box} = h \times H$$

Konversi ke COCO (*pixel-based*):

$$x_{min} = x_{center} - \frac{w_{box}}{2}$$

$$y_{min} = y_{center} - \frac{h_{box}}{2}$$
$$width = w_{box}$$
$$height = h_{box}$$




Gambar 4.4 Ilustrasi *Bounding Box* pada objek *fire* dan *smoke*

## 4.2 Perancangan Sistem

Pada subbab ini dibahas mengenai ruang lingkup perangkat lunak yang digunakan, perancangan proses pelatihan dan perancangan pengujian meliputi uji coba skenario.

### 4.2.1 Ruang Lingkup Perangkat Lunak

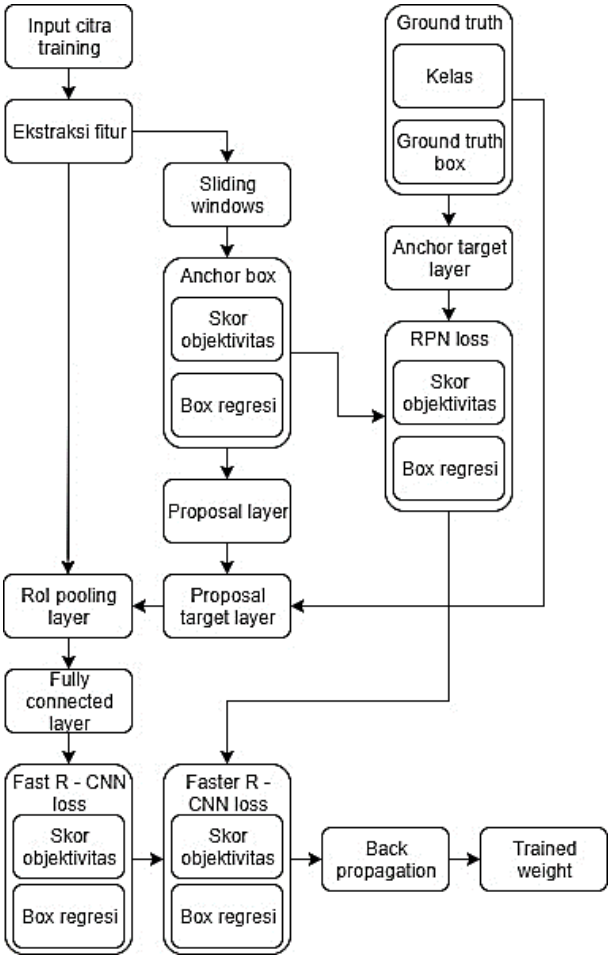
Ruang lingkup sistem merupakan perangkat-perangkat yang digunakan selama penelitian ini berlangsung. Perangkat tersebut digunakan sesuai dengan tujuan penggunaannya.

Tabel 4.2 Ruang Lingkup Sistem

| Nama Perangkat | Spesifikasi | Media yang Digunakan | Tujuan |
|---|---|---|---|
| ASUS TUF Gaming A15 FA506II_FX506II | OS : Windows 11 Home Single Language 64-bit (10.0, Build 26200)<br>Processor : AMD Ryzen 5 4600H with Radeon Graphics (3.00 GHz)<br>RAM : 8 GB<br>GPU : NVIDIA GeForce GTX 1650 Ti | Visual Studio Code dan Kaggle Notebook (GPU T4 x2) | Pengolahan Data, Proses Pelatihan dan Pengujian |

### 4.2.2 Perancangan Pelatihan Faster R-CNN

Pada tahap ini dilakukan perancangan pelatihan untuk melatih model *Faster* R-CNN. Perancangan proses pelatihan *Faster* R-CNN dapat di lihat pada Gambar 4.5.




Gambar 4.5 Blok Diagram Pelatihan *Faster* R-CNN

1. Input Data Training

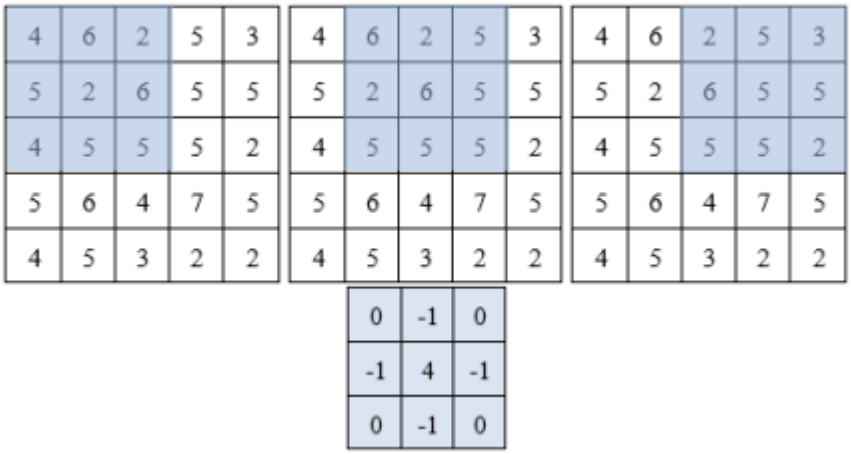
   Pada tahap pelatihan, sistem menerima input berupa data citra hasil ekstraksi frame dari video kebakaran. Setiap video diekstraksi menjadi beberapa frame citra statis, kemudian dilakukan proses anotasi *bounding box* untuk objek api (*fire*) dan asap (*smoke*). Data citra training telah melalui tahap *preprocessing* yang meliputi penyesuaian ukuran citra, normalisasi nilai piksel, dan augmentasi ringan untuk meningkatkan kemampuan generalisasi model.

2. *Ground Truth*

   *Ground truth* merupakan informasi anotasi yang berisi label kelas objek (*fire* dan *smoke*) serta koordinat *bounding box* yang mengelilingi objek kebakaran pada setiap citra training. *Ground truth* disimpan dalam format COCO (.json) dan berfungsi sebagai acuan utama dalam proses pelatihan untuk menghitung selisih antara hasil prediksi model dengan kondisi sebenarnya. Pada arsitektur *Faster* R-CNN, *ground truth* digunakan untuk membentuk *anchor target* pada *Region Proposal Network* (RPN) serta proposal target pada tahap *Fast* R-CNN.

3. *Ekstraksi Fitur*

Citra *input* kemudian diproses pada tahap *ekstraksi fitur* menggunakan backbone *ResNet*-50 yang telah dilengkapi dengan *Feature Pyramid Network* (FPN). Lapisan konvolusi pada *ResNet*-50 melakukan operasi konvolusi berlapis untuk menghasilkan *feature maps* multiskala, sehingga objek api dan asap dengan ukuran kecil maupun besar dapat direpresentasikan dengan baik. Pada penelitian ini digunakan pendekatan transfer learning, yaitu bobot awal *ResNet*-50 diinisialisasi dari model yang telah dilatih pada dataset COCO. Ilustrasi operasi konvolusi ditunjukkan pada Gambar 4.6.




Gambar 4.6 Ilustrasi Operasi Konvolusi

Pada Gambar 4.6. diketahui bahwa citra $f(x, y)$ berukuran $5 * 5$, kernel $g(x, y)$ berukuran $3 * 3$, dan stride 1. Perhitungan berdasarkan ilustrasi tersebut dituliskan sebagai berikut.

1) $h(1,1)$
$(4 * 0) + \big(6 * (-1)\big) + (2 * 0) + \big(5 * (-1)\big) + (2 * 4) + (6 * (-1)) + (4 * 0) + (5 * (-1)) + (5 * 0) = -14$
2) $h(1,2)$
$(6 * 0) + \big(2 * (-1)\big) + (5 * 0) + \big(2 * (-1)\big) + (6 * 4) + (5 * (-1)) + (5 * 0) + (5 * (-1)) + (5 * 0) = 10$
3) $h(1,3)$
$(2 * 0) + \big(5 * (-1)\big) + (3 * 0) + \big(6 * (-1)\big) + (5 * 4) + (5 * (-1)) + (5 * 0) + (5 * (-1)) + (2 * 0) = -1$
4) $h(2,1)$
$(5 * 0) + \big(2 * (-1)\big) + (6 * 0) + \big(4 * (-1)\big) + (5 * 4) + (5 * (-1)) + (5 * 0) + (6 * (-1)) + (4 * 0) = 3$
5) $h(2,2)$
$(2 * 0) + \big(6 * (-1)\big) + (5 * 0) + \big(5 * (-1)\big) + (5 * 4) + (5 * (-1)) + (6 * 0) + (4 * (-1)) + (7 * 0) = 0$
6) $h(2,3)$
$(6 * 0) + \big(5 * (-1)\big) + (5 * 0) + \big(5 * (-1)\big) + (5 * 4) +$

$$(2 * (-1)) + (4 * 0) + (7 * (-1)) + (5 * 0) = 1$$

7) $h(3,1)$

$$(4 * 0) + \big(5 * (-1)\big) + (5 * 0) + \big(5 * (-1)\big) + (6 * 4) + (4 * (-1)) + (4 * 0) + (5 * (-1)) + (3 * 0) = 5$$

8) $h(3,2)$

$$(5 * 0) + \big(5 * (-1)\big) + (5 * 0) + \big(6 * (-1)\big) + (4 * 4) + (7 * (-1)) + (5 * 0) + (3 * (-1)) + (2 * 0) = 0$$

9) $h(3,3)$

$$(5 * 0) + \big(5 * (-1)\big) + (2 * 0) + \big(4 * (-1)\big) + (7 * 4) + (5 * (-1)) + (3 * 0) + (2 * (-1)) + (2 * 0) = 0$$

Hasil operasi konvolusi tersebut ditunjukkan pada Gambar 4.7

| | | |
|---|---|---|
| -14 | 10 | -1 |
| 3 | 0 | 1 |
| 5 | -5 | 12 |

Gambar 4.7 Hasil Ilustrasi Operasi Konvolusi

4. *Sliding Windows*

   *Feature maps* hasil ekstraksi fitur kemudian diproses oleh *Region Proposal Network* (RPN). RPN menerapkan mekanisme *sliding window* pada setiap lokasi *feature maps* untuk menghasilkan sejumlah *anchor box* dengan berbagai skala dan rasio aspek. Pada setiap posisi *sliding window*, RPN memprediksi skor objektivitas (*objectness score*) dan regresi *bounding box* untuk menentukan apakah suatu region berpotensi mengandung objek kebakaran.

5. *Anchor Box*

   Hasil dari proses *sliding window* berupa sekumpulan *anchor box* dengan skor objektivitas dan nilai regresi *bounding box*. *Anchor box* yang memiliki skor objektivitas tinggi kemudian diseleksi dan disimpan sebagai *proposal layer* (*region proposals*). *Proposal layer* ini merepresentasikan kandidat lokasi objek api dan asap yang akan diproses lebih lanjut pada tahap klasifikasi.

6. RPN *Loss*

   Setelah *proposal layer* diperoleh, dilakukan perhitungan RPN *loss* dengan membandingkan hasil prediksi *anchor box* terhadap *anchor target* yang berasal dari *ground truth*. RPN *loss* terdiri atas dua komponen, yaitu: *loss* klasifikasi, untuk menentukan apakah *anchor* termasuk objek atau latar belakang dan *loss* regresi *bounding box*, untuk mengukur ketepatan posisi *anchor* terhadap *ground truth*.

7. *RoI Pooling Layer*
   *Feature maps* hasil ekstraksi fitur dan *proposal layer* hasil RPN kemudian menjadi *input* pada *RoI Pooling layer*. *RoI Pooling* berfungsi untuk mengekstraksi fitur dari setiap proposal dengan membagi *input* menjadi bagian dengan ukuran yang sama. Hasil dari *RoI Pooling* berupa fitur dengan dimensi tetap yang siap diproses pada tahap klasifikasi.

8. *Fully Connected Layer*
   *Output* dari *RoI Pooling* kemudian diproses oleh *fully connected layer*. Pada tahap ini dilakukan klasifikasi objek ke dalam kelas *fire*, *smoke*, atau background, serta regresi *bounding box* untuk menyempurnakan posisi kotak deteksi.

9. *Fast* R-CNN *Loss*
   Setelah melewati *fully connected layer*, dihitung *Fast* R-CNN *loss* dengan membandingkan hasil prediksi kelas dan *bounding box* terhadap proposal target dari *ground truth*. *Fast* R-CNN *loss* terdiri atas: loss klasifikasi, untuk menentukan kelas objek dan *loss* regresi *bounding box*, untuk memperbaiki koordinat *bounding box* hasil prediksi.

10. *Faster R-CNN Loss*
    Nilai *Faster* R-CNN *loss* diperoleh dari penjumlahan antara RPN *loss* dan *Fast* R-CNN *loss*. *Loss* ini digunakan sebagai fungsi objektif utama untuk memperbarui bobot model selama proses pelatihan.

11. *Backpropagation*
    Setelah nilai *Faster* R-CNN *loss* diperoleh, dilakukan proses *backpropagation* untuk memperbarui bobot model. Proses ini dilakukan dengan menghitung turunan *loss* terhadap bobot jaringan menggunakan algoritma optimasi (misalnya *Stochastic Gradient Descent* atau *Adam*). Secara matematis, proses pembaruan bobot dapat dituliskan sebagai:

$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\partial E}{\partial w}$$

dengan:

$w_{\text{new}}$ : bobot ($w$) baru
$w_{\text{old}}$ : bobot ($w$) lama
$\alpha$ : nilai *learning rate*
$\frac{\partial E}{\partial w}$ : turunan *Faster* R – CNN *loss* terhadap bobot ($w$)

12. *Trained Weight*

Bobot hasil pelatihan kemudian disimpan sebagai *trained weight*. *Trained weight* ini digunakan pada tahap deteksi untuk melakukan identifikasi objek api dan asap pada data video uji.

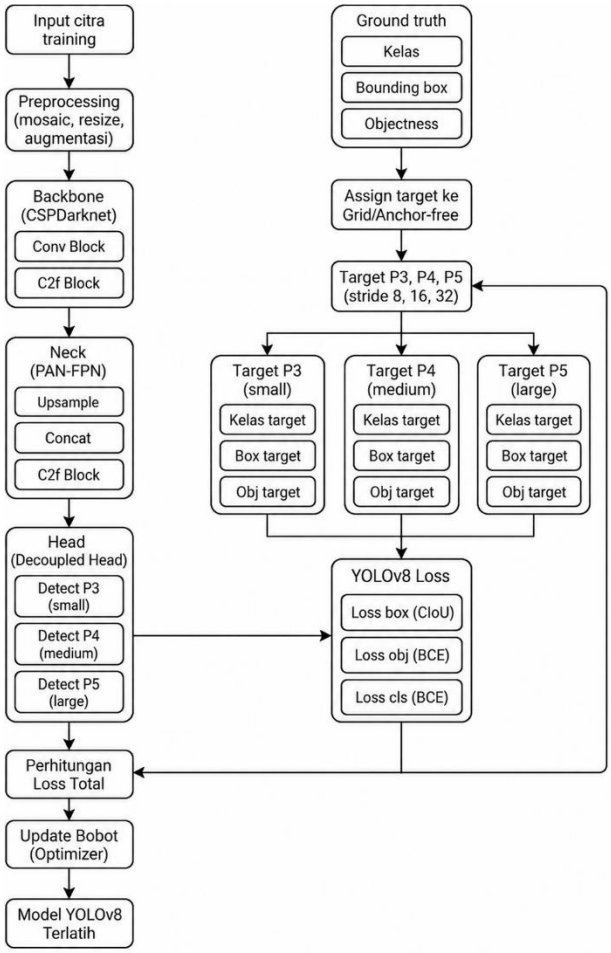
- *Tuning Hyperparameter* Pelatihan *Faster* R-CNN
  Pada pelatihan model *Faster* R-CNN digunakan tuning hyperparameter yang cocok dengan GPU T4 x2 pada Kaggle Notebook seperti pada tabel 4.3 berikut:

Tabel 4.3 Tuning Hyperparameter Pelatihan Faster R-CNN

| Hyperparameter | Deskripsi | Nilai |
|---|---|---|
| Epochs | Frekuensi dilatihnya model terhadap suatu dataset | 25 |
| Batch Size | Jumlah data yang diambil setiap melatih model | 32 |
| Pretrained | Konfigurasi untuk melatih dari model pretrained | True |
| AMP | Latih dengan Automatic Mixed Precision (dapat mempercepat training dengan GPU) | True |
| Lr0 | Tingkat pembelajaran awal model | 0.1, 0.01, 0.001 dan 0.0001 |
| Optimizer | Algoritma optimasi yang digunakan model | SGD dan AdamW |

### 4.2.3 Perancangan Pelatihan YOLOv8

Pada tahap ini dilakukan perancangan pelatihan untuk melatih model *YOLOv8*. Perancangan proses pelatihan *YOLOv8* dapat di lihat pada Gambar 4.8.




Gambar 4.8 Blok Diagram Pelatihan *YOLOv8*

1. Input Data Training

   Pada tahap pelatihan, sistem menerima input berupa data citra hasil ekstraksi frame dari video kebakaran. Setiap video diekstraksi menjadi sejumlah frame citra statis, kemudian dilakukan proses anotasi objek api (*fire*) dan asap (*smoke*) menggunakan *bounding box*. Sebelum digunakan untuk pelatihan, citra melalui tahap *preprocessing* yang meliputi penyesuaian ukuran citra (*resize*) sesuai ukuran input model, normalisasi nilai piksel, serta augmentasi data seperti *mosaic augmentation*, *horizontal flip*, rotasi, perubahan kecerahan (*brightness adjustment*), dan transformasi geometris lainnya. Tahap ini bertujuan untuk meningkatkan keragaman data dan kemampuan generalisasi model terhadap berbagai kondisi lingkungan.

2. *Ground Truth*

   *Ground truth* merupakan data anotasi yang berisi informasi kelas objek dan koordinat *bounding box* pada setiap citra pelatihan. Pada penelitian ini, kelas objek terdiri atas *fire* dan *smoke*. Format anotasi yang digunakan pada YOLOv8 adalah format YOLO (*.txt*) yang menyimpan informasi kelas objek beserta koordinat titik pusat (*x_center*, *y_center*) dan ukuran *bounding box* (*width*, *height*) yang telah dinormalisasi terhadap ukuran citra. Selain informasi kelas

dan *bounding box*, YOLOv8 juga membentuk target *objectness* untuk menentukan keberadaan objek pada suatu lokasi. Data *ground truth* digunakan sebagai acuan dalam proses pelatihan untuk menghitung kesalahan prediksi model.

3. *Backbone*
   Citra hasil *preprocessing* kemudian diproses pada bagian *backbone* untuk melakukan ekstraksi fitur. YOLOv8m menggunakan arsitektur *backbone* berbasis CSPDarknet yang terdiri atas beberapa lapisan konvolusi (*Convolution Block*) dan modul C2f (*Cross Stage Partial with Two Convolutions Faster*). Struktur ini dirancang untuk mengekstraksi fitur visual secara efisien dengan kompleksitas komputasi yang lebih rendah. Pada tahap ini, model mempelajari berbagai karakteristik visual objek kebakaran, seperti warna, tekstur, bentuk, dan pola penyebaran api maupun asap pada berbagai kondisi pencahayaan dan ukuran objek. Dalam penelitian ini, proses pelatihan menggunakan pendekatan *transfer learning*, yaitu dengan menginisialisasi bobot awal model menggunakan bobot pralatih (*pre-trained weights*) dari dataset COCO.

4. *Neck*
   *Feature map* yang dihasilkan oleh *backbone* selanjutnya diproses pada bagian *neck*. YOLOv8 menggunakan kombinasi struktur PAN-FPN (*Path Aggregation Network–Feature Pyramid Network*) yang terdiri atas proses *upsampling*, *concatenation*, dan modul C2f untuk menggabungkan informasi fitur dari berbagai skala. Tahap ini bertujuan untuk meningkatkan kemampuan model dalam mendeteksi objek dengan ukuran yang bervariasi, baik objek kecil maupun besar. Penggabungan fitur multiskala memungkinkan model mengenali karakteristik api dan asap secara lebih akurat pada berbagai kondisi.

5. *Head Detection*
   Hasil penggabungan fitur pada bagian *neck* kemudian diteruskan ke *detection head*. YOLOv8 menggunakan *decoupled head*, yaitu struktur yang memisahkan proses prediksi klasifikasi dan regresi *bounding box* sehingga proses pembelajaran menjadi lebih optimal. Pada tahap ini, model menghasilkan prediksi pada tiga tingkat skala fitur yaitu, Detect P3 untuk objek berukuran kecil (*small object*), Detect P4 untuk objek berukuran sedang (*medium object*), Detect P5 untuk objek berukuran besar (*large object*). Setiap skala menghasilkan prediksi berupa koordinat *bounding box*, skor *objectness*, dan probabilitas kelas objek.

6. *Assign Target ke Grid Anchor-Free*
   Berbeda dengan versi YOLO sebelumnya yang menggunakan *anchor box*, YOLOv8 menerapkan pendekatan *anchor-free*. Pada tahap ini, informasi *ground truth* dipetakan secara langsung ke grid pada *feature map* tanpa

menggunakan *anchor box*. Proses *assign target* menentukan lokasi prediksi positif pada setiap level fitur, yaitu P3, P4, dan P5, berdasarkan ukuran dan posisi objek. Selanjutnya, dibentuk target untuk setiap level deteksi yang terdiri atas:

- *Class target*, yaitu label kelas objek.
- *Box target*, yaitu koordinat *bounding box.*
- *Objectness target*, yaitu indikator keberadaan objek.

Target-target tersebut digunakan sebagai acuan dalam perhitungan fungsi *loss*.

7. *YOLOv8 Loss*

   Pada tahap pelatihan, hasil prediksi model dibandingkan dengan *ground truth* untuk menghitung nilai kesalahan (*loss*). Fungsi *loss* pada YOLOv8 terdiri atas tiga komponen utama, yaitu:

   - *Bounding Box Loss (CIoU Loss), digunakan untuk mengukur tingkat kesesuaian antara bounding box prediksi dan ground truth.*
   - *Objectness Loss (Binary Cross-Entropy/BCE), digunakan untuk mengukur kemampuan model dalam menentukan keberadaan objek pada suatu lokasi.*
   - *Classification Loss (Binary Cross-Entropy/BCE), digunakan untuk mengukur kesalahan prediksi kelas objek.*

   Ketiga komponen tersebut dijumlahkan untuk memperoleh nilai *total loss* yang digunakan sebagai fungsi objektif selama proses pelatihan.

8. Perhitungan *Loss* Total

   Nilai *bounding box loss*, *objectness loss*, dan *classification loss* dari seluruh level deteksi (P3, P4, dan P5) digabungkan untuk menghasilkan *total loss*. Semakin kecil nilai *total loss*, semakin baik kemampuan model dalam memprediksi lokasi dan kelas objek sesuai dengan *ground truth*. Nilai *total loss* digunakan sebagai dasar untuk memperbarui parameter model pada proses optimasi.

9. Update Bobot (*Optimizer*)

   Setelah nilai *total loss* diperoleh, dilakukan proses pembaruan bobot model menggunakan algoritma optimasi seperti *Stochastic Gradient Descent* (SGD) atau AdamW. Proses optimasi dilakukan melalui mekanisme *backpropagation*, yaitu menghitung gradien dari fungsi *loss* terhadap setiap parameter jaringan, kemudian memperbarui bobot model untuk meminimalkan nilai *loss*. Secara matematis, pembaruan bobot dapat dinyatakan sebagai berikut:

$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\partial E}{\partial w}$$

   dengan:

   $w_{\text{new}}$ : bobot ($w$) baru

   $w_{\text{old}}$ : bobot ($w$) lama

$\alpha$ : nilai *learning rate*

$\frac{\partial E}{\partial w}$ : turunan *loss* terhadap bobot ($w$)

Proses ini dilakukan secara iteratif pada setiap *batch* hingga seluruh *epoch* pelatihan selesai.

*10. Trained Weight*

Setelah seluruh proses pelatihan selesai, bobot model hasil pembelajaran disimpan sebagai model YOLOv8 terlatih (*trained weights*). Model terlatih ini selanjutnya digunakan pada tahap pengujian untuk mendeteksi objek api dan asap pada data video uji.

- *Tuning Hyperparameter* Pelatihan *YOLOv8*

  Pada pelatihan model *YOLOv8* digunakan tuning hyperparameter yang cocok dengan GPU T4 x2 pada Kaggle Notebook seperti pada tabel 4.4 berikut:

Tabel 4.4 Tuning Hyperparameter Pelatihan *YOLOv8*

| Hyperparameter | Deskripsi | Nilai |
|---|---|---|
| Epochs | Frekuensi dilatihnya model terhadap suatu dataset | 100 |
| Batch Size | Jumlah data yang diambil setiap melatih model | 32 |
| Pretrained | Konfigurasi untuk melatih dari model pretrained | True |
| AMP | Latih dengan Automatic Mixed Precision (dapat mempercepat training dengan GPU) | True |
| Lr0 | Tingkat pembelajaran awal model | 0.1, 0.01, 0.001 dan 0.0001 |
| Optimizer | Algoritma optimasi yang digunakan model | SGD dan AdamW |

### 4.2.4 Perancangan Pengujian

Pada tahap ini, model terbaik diuji performansinya terhadap data uji sebanyak 6 video. Proses pengujian dilakukan untuk memberikan hasil visual terhadap pendeteksian api dan asap dengan data video. Model diuji menggunakan data primer dan data sekunder. Tujuan dilakukan pengujian dengan kedua data ini disebabkan perbedaan lingkungan pembuatan masing-masing *dataset*. Pengujian dilakukan dengan cara memberikan *input* berupa video data primer dan sekunder ke dalam sistem deteksi api dan asap yang sudah di latih sebelumnya. Keluaran dari model berupa video yang terdapat *bounding box* dengan *label* dan *confidence score*. Pengujian ini meliputi pengukuran presisi, recall, dan F1 score untuk setiap video. Berikut adalah penjelasan dari pengukuran yang digunakan:

- *Precision* adalah proporsi prediksi positif yang benar terhadap semua prediksi positif (lihat persamaan 2.12).
- *Recall* adalah proporsi kasus positif yang terprediksi dengan benar terhadap semua kasus positif (lihat persamaan 2.13).
- *F1-Score* dihitung sebagai rata-rata terbobot *precision* dan *recall* antara 0 dan 1, dimana nilai skor F1 yang ideal adalah 1 (lihat persamaan 2.15).

Pendefinisian *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (FN) dijelaskan sebagai berikut:

TP : Jumlah objek *ground truth* yang terdeteksi dengan benar oleh model, yaitu memiliki kelas yang sesuai dan nilai IoU ≥ *threshold*.

FP : Jumlah objek hasil prediksi yang tidak sesuai dengan *ground truth*, baik karena tidak memiliki pasangan *ground truth*, kelas prediksi salah, atau nilai IoU < *threshold*.

FN : Jumlah objek *ground truth* yang gagal dideteksi oleh model, baik karena tidak terdeteksi, kelas prediksi salah, atau nilai IoU < *threshold*.

Pengujian dilakukan berdasarkan skenario yang dibuat untuk mengetahui kemampuan sistem deteksi dengan kondisi tertentu. Skenario yang akan digunakan yaitu:

1. Skenario Pengaruh Tingkat Pencahayaan
   Skenario ini bertujuan untuk mengukur keandalan dan stabilitas model terhadap perubahan intensitas cahaya di sekitar lokasi. Pengujian dilakukan menggunakan tiga tingkat pencahayaan yang berbeda yaitu, normal, cerah, dan redup.

## 4.3 Implementasi

Pada subbab ini dilakukan implementasi proses pelatihan dan pengujian model menggunakan program dengan bahasa Python. Untuk mempermudah pemahaman, implementasi program yang dijelaskan adalah rangkuman langkah utama dari seluruh sub-program yang digunakan. Implementasi sistem seluruhnya menggunakan *Kaggle Notebook*. Tahap implementasi program menjelaskan mulai dari tahap implementasi *Faster* R-CNN hingga implementasi *YOLOv8*.

### 4.3.1 Implementasi *Faster* R-CNN

Pada tahap ini diimplentasikan program untuk proses pelatihan dan pengujian model *Faster* R-CNN. Tahap-tahap tersebut diantaranya adalah proses install *Detectron2*, proses cek class dan direktori dataset, proses pelatihan *Faster* R-CNN, dan proses pengujian *Faster* R-CNN.

1. Proses install *Detectron2*
   Proses instalasi *Detectron2* pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.1.

Kode Program 4.1 install *Detectron2*

```
!pip install -q torch torchvision

!python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'

import torch
import detectron2
```

```
print("Torch :", torch.__version__)
print("Detectron2 :", detectron2.__version__)
```

Kode tersebut digunakan untuk *download* dan *install* seluruh *requirements* dari *framework Detectron2* melalui link resminya.

2. Proses cek class dan direktori dataset

   Proses pengecekan class dan direktori dataset pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.2.

Kode Program 4.2 cek class dan direktori dataset

```
%%writefile data_configs/dataset.yaml

train_images: "/kaggle/working/dataset_fixed/train/images"
train_json: "/kaggle/working/dataset_fixed/train/labels/_annotations.coco_train.json
"

val_images: "/kaggle/working/dataset_fixed/valid/images"
val_json: "/kaggle/working/dataset_fixed/valid/labels/_annotations.coco_valid.json"

test_images: "/kaggle/working/dataset_fixed/test/images"
test_json: "/kaggle/working/dataset_fixed/test/labels/_annotations.coco_test.json"

nc: 2

names:
 - fire
 - smoke
```

Kode tersebut digunakan untuk menentukan *path* atau direktori dan banyak kelas yang dipakai pada *dataset*. Kelas yang di pakai ada 2 yaitu *fire* dan *smoke*.

3. Proses pelatihan *Faster* R-CNN

   Proses pelatihan *Faster* R-CNN pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.3.

Kode Program 4.3 pelatihan *Faster* R-CNN

```
!python train.py \
--model=faster_rcnn_R_50_FPN_3x \
--data=data_configs/dataset.yaml \
--epochs 100 \
--imgsz 640 \
--batch 16 \
--optimizer SGD \
--lr0 0.01 \
--momentum 0.937 \
--patience 10
```

Kode tersebut digunakan untuk melatih *pre-trained* model *Faster R-CNN Resnet-50 FPN* dari *Detectron2* pada folder *train* dan mengevaluasi model pada folder *valid*

dengan *file images* berukuran 640 x 640 , jumlah *epoch* = 100, *batch size* = 16, optimizer SGD, *learning rate* awal = 0.001, *momentum* = 0.937, dan *early stopping* saat *patience* mencapai 10.

4. Proses pengujian *Faster* R-CNN
   Proses pengujian *Faster* R-CNN pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.4.

Kode Program 4.4 pengujian *Faster* R-CNN

```
!python test.py \
--conf=0.25 \
--iou=0.5 \
--brightness=0 \
--contrast=1.0
```

Kode tersebut digunakan untuk menguji bobot model hasil *training* pada data video di folder *test* dengan *confidence threshold* = 0.25 dan *intersection over union threshold* = 0.5.

### 4.3.2 Implementasi *YOLOv8*

Pada tahap ini diimplentasikan program untuk proses pelatihan dan pengujian model *YOLOv8*. Tahap-tahap tersebut diantaranya adalah proses install *YOLOv8*, proses cek class dan direktori dataset, proses pelatihan *YOLOv8*, dan proses pengujian *YOLOv8.*

1. Proses install *YOLOv8*
   Proses instalasi *YOLOv8* pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.5.

Kode Program 4.5 install *YOLOv8*

```
!pip install -q ultralytics

from ultralytics import YOLO
import ultralytics

ultralytics.checks()
```

Kode tersebut digunakan untuk *download* dan *install* seluruh *requirements* dari *Ultralytics* melalui link resminya.

2. Proses cek class dan direktori dataset
   Proses pengecekan class dan direktori dataset pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.6.

Kode Program 4.6 cek class dan direktori dataset

```
%%writefile data_configs/data.yaml

train: "/kaggle/working/dataset_fixed/train/images"
val: "/kaggle/working/dataset_fixed/valid/images"
```

```
test: "/kaggle/working/dataset_fixed/test/images"

nc: 2

names:
  0: fire
  1: smoke
```

Kode tersebut digunakan untuk menentukan *path* atau direktori dan banyak kelas yang dipakai pada *dataset*. Kelas yang di pakai ada 2 yaitu *fire* dan *smoke.*

3. Proses pelatihan *YOLOv8*

   Proses pelatihan *YOLOv8* pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.7.

Kode Program 4.7 pelatihan *YOLOv8*

```
!yolo task=detect \
mode=train \
model=yolov8m.pt \
data=data_configs/data.yaml \
epochs=100 \
imgsz=640 \
batch=16 \
optimizer=SGD \
lr0=0.01 \
momentum=0.937 \
patience=10 \
plots=True
```

Kode tersebut digunakan untuk melatih *pre-trained* model *YOLOv8m* dari *Ultralytics* pada folder *train* dan mengevaluasi model pada folder *valid* dengan *file images* berukuran 640 x 640 , jumlah *epoch* = 100, *batch size* = 16, optimizer SGD, *learning rate* awal = 0.001, *momentum* = 0.937, dan *early stopping* saat *patience* mencapai 10.

4. Proses pengujian *YOLOv8*

   Proses pengujian *YOLOv8* pada *kaggle notebook* dengan GPU T4 dapat dilakukan melalui Kode Program 4.8.

Kode Program 4.8 pengujian *YOLOv8*

```
!python test.py \
--conf=0.25 \
--iou=0.5 \
--brightness=0 \
--contrast=1.0
```

Kode tersebut digunakan untuk menguji bobot model hasil *training* pada data video di folder *test* dengan *confidence threshold* = 0.25 dan *intersection over union threshold* = 0.5.

# BAB V
# HASIL DAN PEMBAHASAN

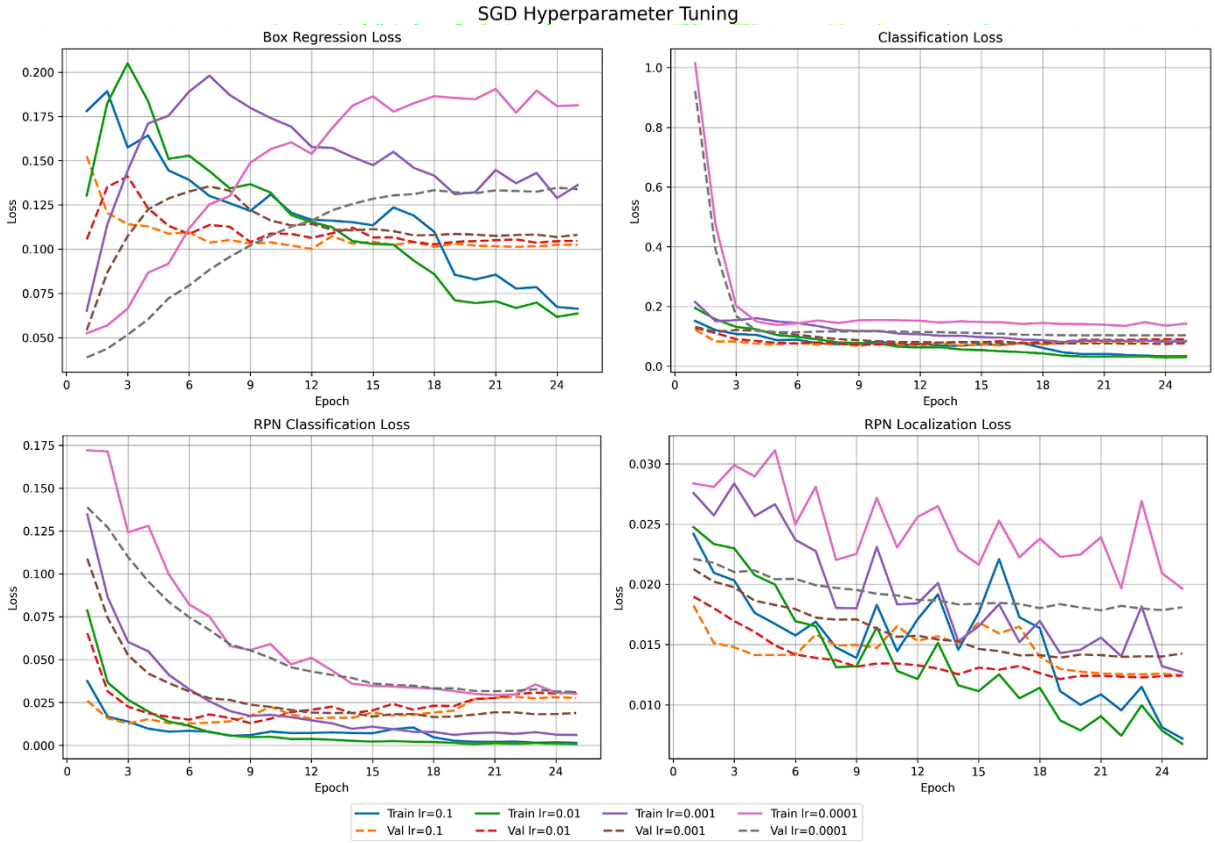
# BAB V
# HASIL DAN PEMBAHASAN

## 5.1 Hasil Pelatihan dan Evaluasi Model

Pada subbab ini membahas hasil dari pelatihan dan evaluasi model menggunakan data latih dan data validasi yang telah ditentukan. Analisa hasil pelatihan dan evaluasi model dilihat berdasarkan grafik train loss dan grafik valid loss serta grafik performa model *Faster R-CNN* dan *YOLOv8*.

### 5.1.1 Hasil Pelatihan *Faster R-CNN*

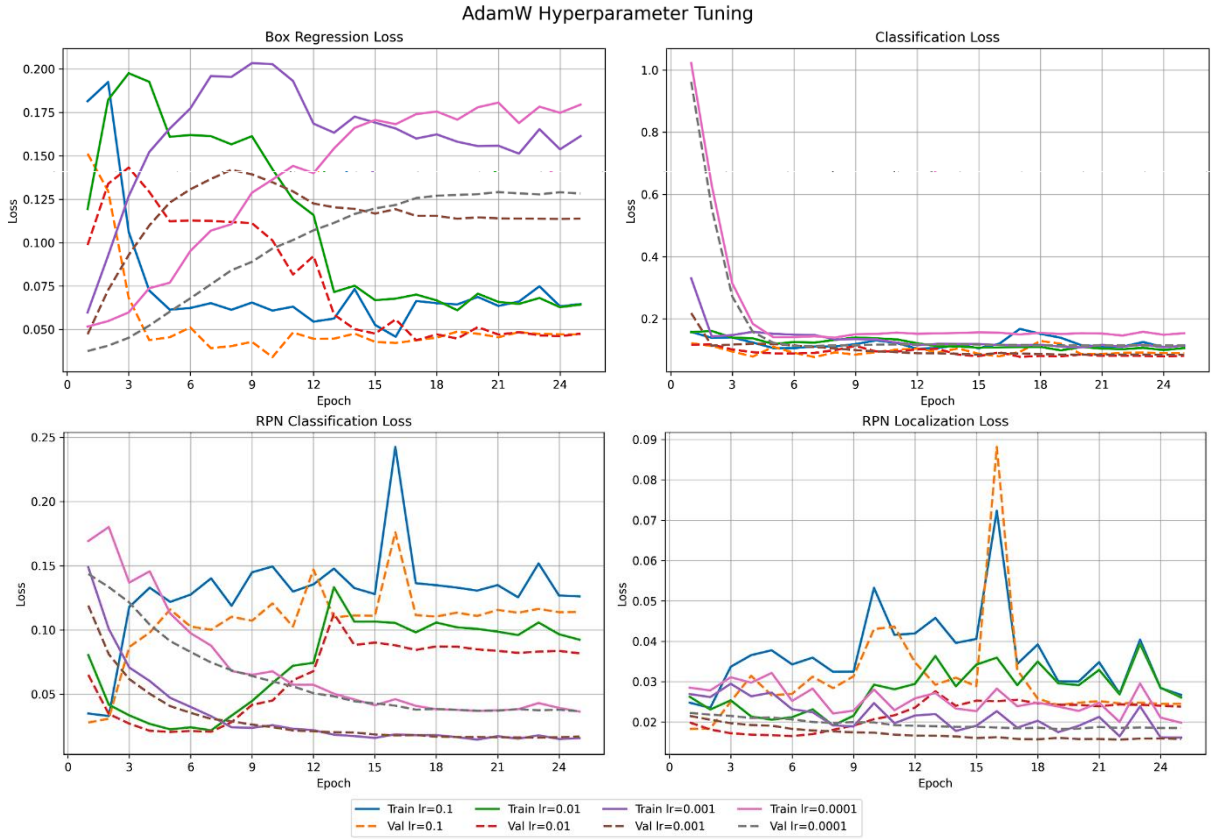
Pada tahap ini pelatihan dan evaluasi *Faster* R-CNN untuk mendeteksi api dan asap menggunakan jenis model *Faster R-CNN Resnet-50 FPN*. Rangkuman grafik nilai loss yang dihasilkan model dapat di lihat pada Gambar 5.1. dan Gambar 5.2.




Gambar 5.1 Perbandingan Loss ROI Head dan Loss RPN dengan SGD untuk *Faster* R-CNN

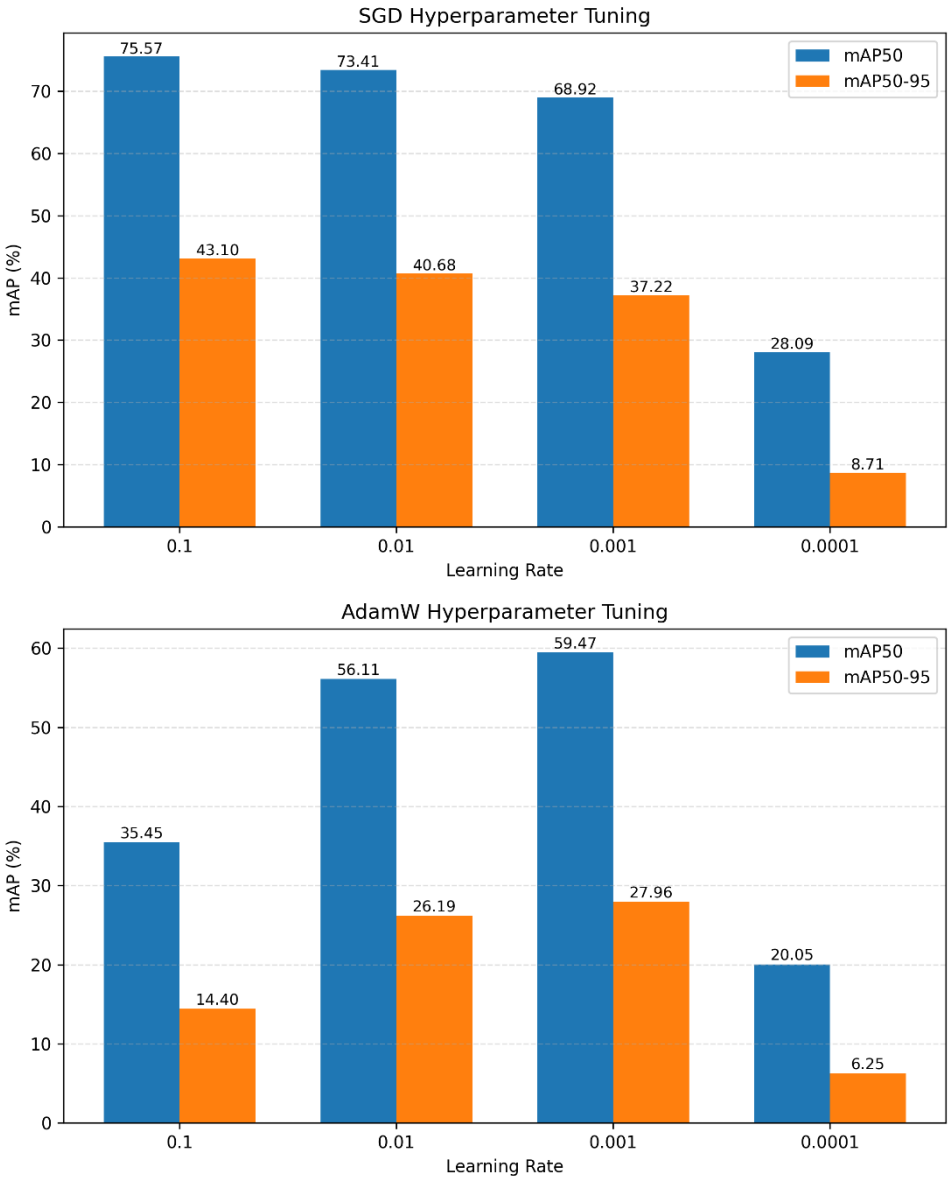
Berdasarkan Gambar 5.1, terlihat bahwa seluruh konfigurasi learning rate pada optimizer SGD menunjukkan penurunan *training loss* dan *validation loss* pada komponen ROI Head (Box Regression Loss dan Classification Loss) maupun RPN (RPN Classification Loss dan RPN Localization Loss) pada beberapa epoch awal, kemudian cenderung stabil hingga akhir pelatihan. Konfigurasi learning rate $10^{-2}$ (garis hijau) menghasilkan *training loss* yang paling rendah dan memiliki selisih yang relatif kecil terhadap *validation loss* pada sebagian besar komponen loss, sehingga menunjukkan proses pembelajaran yang lebih stabil dan kemampuan generalisasi yang lebih baik. Sebaliknya, learning rate $10^{-4}$ (garis merah muda) menghasilkan *training loss* yang relatif lebih tinggi dan fluktuatif, sementara *validation loss* cenderung tidak

mengalami penurunan yang signifikan. Adapun learning rate $10^{-1}$ dan $10^{-3}$ mampu menurunkan nilai loss, namun hasilnya masih kurang konsisten dibandingkan learning rate $10^{-2}$.




Gambar 5.2 Perbandingan Loss ROI Head dan Loss RPN dengan AdamW untuk *Faster* R-CNN

Berdasarkan Gambar 5.2, terlihat bahwa seluruh konfigurasi *learning rate* pada optimizer AdamW mampu menurunkan *training loss* dan *validation loss* pada komponen ROI Head (Box Regression Loss dan Classification Loss) maupun RPN (RPN Classification Loss dan RPN Localization Loss) selama proses pelatihan. Konfigurasi learning rate $10^{-2}$ (garis hijau) menunjukkan penurunan *training loss* yang relatif konsisten dengan selisih yang tidak terlalu besar terhadap *validation loss* pada sebagian besar komponen loss, sehingga mengindikasikan proses pembelajaran yang lebih stabil. Sebaliknya, learning rate $10^{-4}$ (garis merah muda) menghasilkan *training loss* yang lebih tinggi dan cenderung meningkat pada Box Regression Loss, sedangkan *validation loss* tidak mengalami penurunan yang signifikan. Learning rate $10^{-1}$ (garis biru) menunjukkan fluktuasi yang cukup besar, terutama pada RPN Classification Loss dan RPN Localization Loss, sementara learning rate $10^{-3}$ (garis ungu) menghasilkan penurunan loss yang lebih baik dibandingkan $10^{-4}$, namun masih kurang konsisten dibandingkan $10^{-2}$.




Gambar 5.3 Perbandingan mAP50 dan mAP50-95 antara SGD dan AdamW untuk *Faster* R-CNN

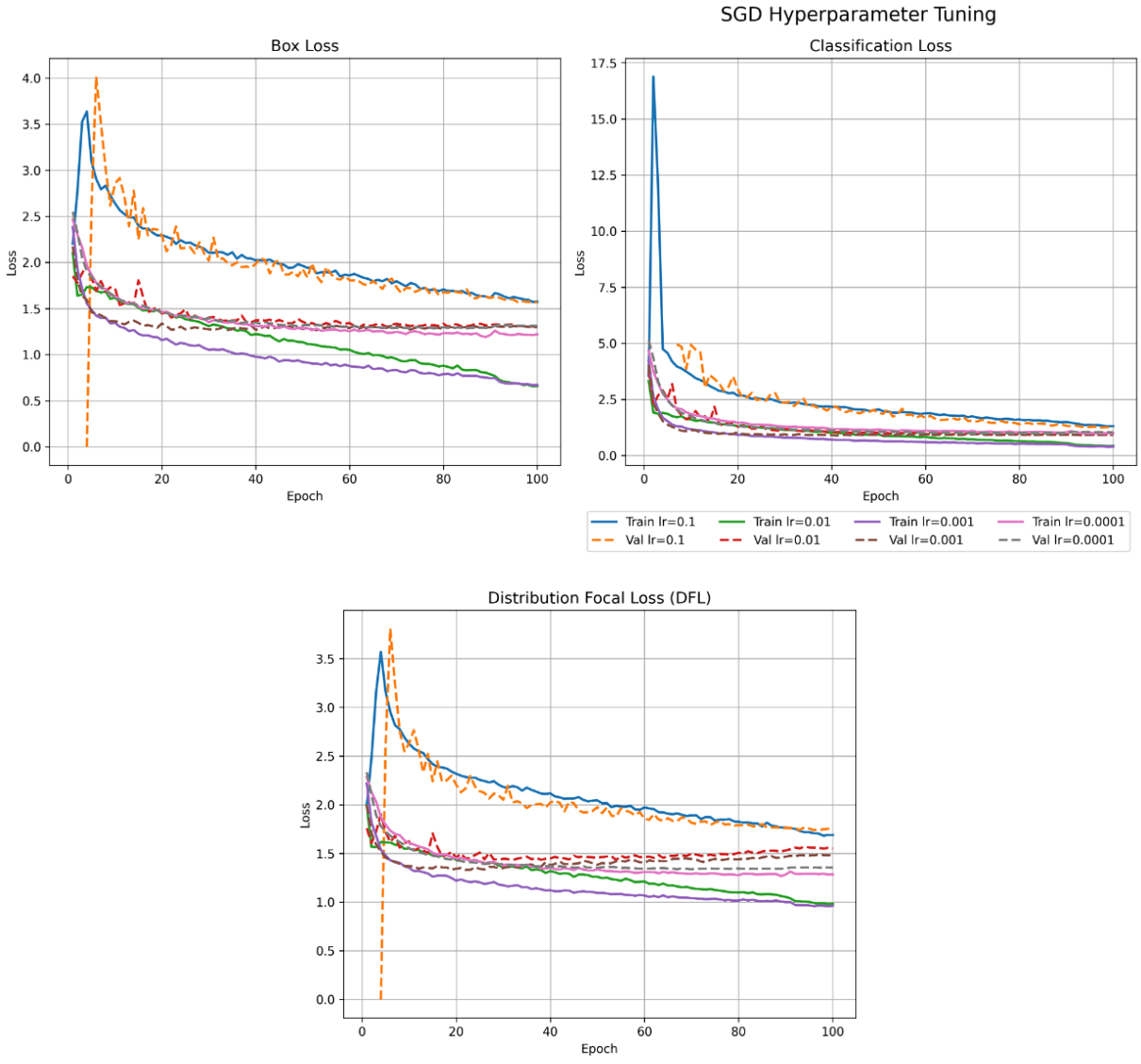
Berdasarkan hasil pelatihan menggunakan delapan kombinasi hyperparameter, terlihat bahwa nilai *loss* yang rendah tidak selalu menghasilkan nilai mAP yang tinggi. Hal ini menunjukkan bahwa nilai *loss* dan mAP tidak memiliki hubungan yang bersifat linier karena masing-masing komponen *loss* pada Faster R-CNN, yaitu ROI Head dan RPN, memberikan kontribusi yang berbeda terhadap kemampuan model dalam melakukan deteksi objek. Oleh karena itu, pemilihan model terbaik dilakukan berdasarkan hasil evaluasi menggunakan metrik mAP50 dan mAP50-95 pada data validasi, karena kedua metrik tersebut lebih merepresentasikan kemampuan deteksi model dibandingkan hanya melihat nilai *loss* selama proses pelatihan.

Berdasarkan Gambar 5.3, optimizer SGD memberikan performa yang lebih baik dibandingkan AdamW pada sebagian besar konfigurasi *learning rate*. Pada optimizer SGD, konfigurasi learning rate 0,1 menghasilkan performa terbaik dengan mAP50 sebesar 75,57% dan mAP50-95 sebesar 43,10%, diikuti oleh *learning rate* 0,01 dengan mAP50 sebesar 73,41% dan mAP50-95 sebesar 40,68%. Sementara itu, pada optimizer AdamW, performa terbaik

diperoleh pada learning rate 0,001 dengan mAP50 sebesar 59,47% dan mAP50-95 sebesar 27,96%. Berdasarkan hasil tersebut, konfigurasi terbaik yang dipilih sebagai best model adalah optimizer SGD dengan learning rate 0,1, karena menghasilkan nilai mAP50 dan mAP50-95 tertinggi di antara seluruh kombinasi hyperparameter yang diuji.

### 5.1.2 Hasil Pelatihan *YOLOv8*

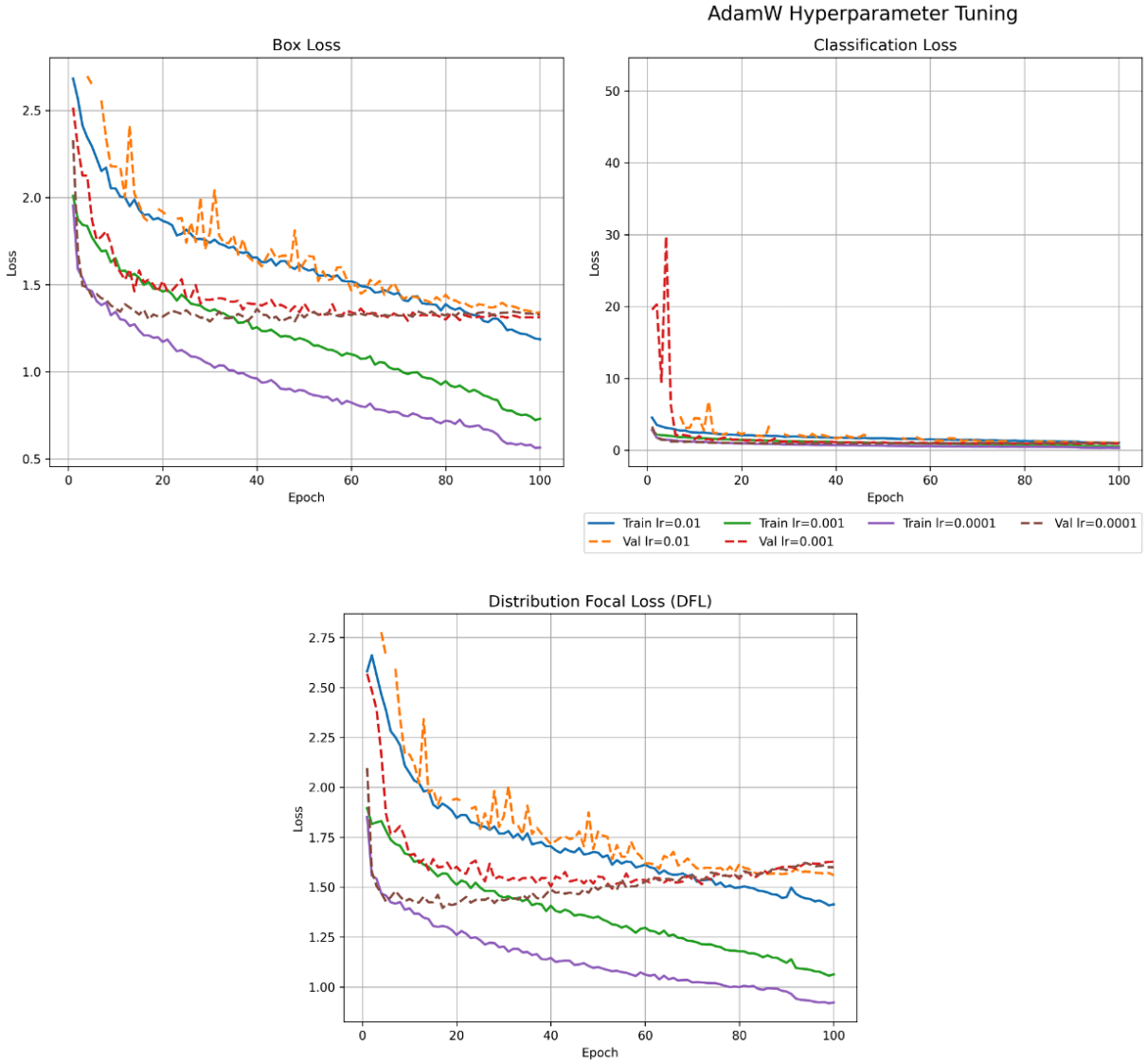
Pada tahap ini pelatihan dan evaluasi model *YOLOv8* untuk mendeteksi api dan asap menggunakan jenis model *YOLOv8m*. Rangkuman grafik nilai loss yang dihasilkan model dapat di lihat pada Gambar 5.4. dan Gambar 5.5.




Gambar 5.4 Perbandingan Box Loss, Class Loss dan DFL Loss dengan SGD untuk *YOLOv8*

Berdasarkan Gambar 5.4, terlihat bahwa seluruh konfigurasi *learning rate* pada optimizer SGD menunjukkan tren penurunan *train loss* dan *validation loss* pada ketiga komponen loss, yaitu Box Loss, Classification Loss, dan Distribution Focal Loss (DFL Loss), seiring bertambahnya epoch. Penurunan loss yang terjadi pada beberapa epoch awal menunjukkan bahwa model mampu mempelajari karakteristik data dengan baik, kemudian kurva cenderung

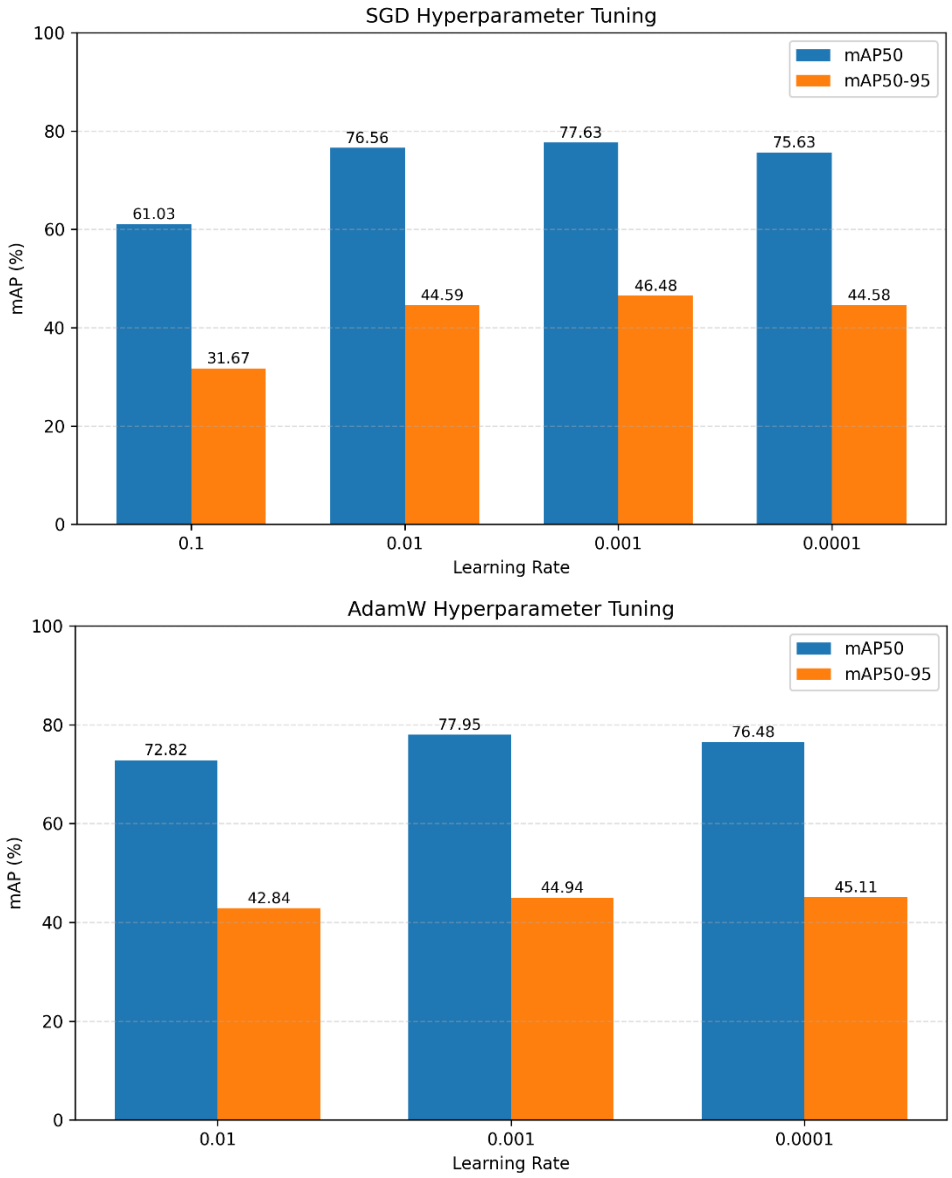
melandai hingga akhir proses pelatihan yang menandakan model mulai mencapai kondisi konvergen. Konfigurasi learning rate 0,001 (garis ungu) menghasilkan nilai *train loss* terendah dan menunjukkan penurunan yang paling konsisten pada ketiga komponen loss. Sementara itu, learning rate 0,1 (garis biru) dan 0,01 (garis hijau) juga mengalami penurunan loss yang stabil meskipun nilai akhirnya lebih tinggi dibandingkan *learning rate* 0,001. Sebaliknya, learning rate 0,0001 (garis merah muda) menunjukkan penurunan loss yang lebih lambat sehingga mengindikasikan proses pembelajaran yang kurang optimal.




Gambar 5.5 Perbandingan Box Loss, Class Loss dan DFL Loss dengan AdamW untuk *YOLOv8*

Berdasarkan Gambar 5.5, terlihat bahwa seluruh konfigurasi *learning rate* pada optimizer AdamW menunjukkan tren penurunan *train loss* pada ketiga komponen loss, yaitu Box Loss, Classification Loss, dan Distribution Focal Loss (DFL Loss), seiring bertambahnya epoch. Penurunan yang signifikan terjadi pada beberapa epoch awal, kemudian kurva cenderung melandai hingga akhir pelatihan yang menunjukkan bahwa model mulai mencapai kondisi konvergen. Konfigurasi learning rate 0,001 (garis ungu) menghasilkan *train loss* terendah pada

ketiga komponen loss dan menunjukkan proses pembelajaran yang paling konsisten. Sementara itu, learning rate 0,01 (garis hijau) juga memperlihatkan penurunan loss yang stabil dengan nilai akhir yang lebih rendah dibandingkan learning rate 0,1 (garis biru). Adapun learning rate 0,0001 (garis merah muda) menghasilkan *train loss* yang relatif lebih tinggi dan *validation loss* yang cenderung stagnan pada akhir pelatihan, sehingga mengindikasikan kemampuan pembelajaran yang kurang optimal.




Gambar 5.6 Perbandingan mAP50 dan mAP50-95 antara SGD dan AdamW untuk *YOLOv8*

Berdasarkan hasil pelatihan menggunakan delapan kombinasi hyperparameter, terlihat bahwa nilai *loss* yang rendah tidak selalu menghasilkan nilai mAP yang lebih tinggi. Hal ini menunjukkan bahwa hubungan antara *loss* dan mAP tidak bersifat linier karena setiap komponen *loss* pada YOLOv8, yaitu Box Loss, Classification Loss, dan Distribution Focal Loss (DFL Loss), memiliki kontribusi yang berbeda terhadap kemampuan model dalam mendeteksi objek. Oleh karena itu, pemilihan model terbaik dilakukan berdasarkan hasil evaluasi menggunakan metrik mAP50 dan mAP50-95, yang lebih mencerminkan performa deteksi model dibandingkan hanya melihat nilai *loss* selama proses pelatihan.

Berdasarkan Gambar 5.6, optimizer SGD dan AdamW menunjukkan performa yang kompetitif pada beberapa konfigurasi *learning rate*. Pada optimizer SGD, konfigurasi learning rate 0,001 menghasilkan mAP50 sebesar 77,63% dan mAP50-95 sebesar 46,48%, yang merupakan nilai mAP50-95 tertinggi di antara seluruh konfigurasi yang diuji. Sementara itu, pada optimizer AdamW, konfigurasi learning rate 0,001 menghasilkan mAP50 sebesar 77,95%, yang merupakan nilai mAP50 tertinggi, dengan mAP50-95 sebesar 44,94%. Berdasarkan hasil tersebut, konfigurasi AdamW dengan learning rate 0,001 dipilih sebagai best model karena menghasilkan nilai mAP50 tertinggi, meskipun nilai mAP50-95 tertinggi diperoleh oleh SGD dengan learning rate 0,001.

## 5.2 Hasil Pengujian Skenario : Pengaruh Tingkat Pencahayaan

Pada subbab ini membahas hasil dari pengujian model dengan skenario menggunakan data uji yang telah ditentukan. Hasil pengujian skenario dengan pengaruh tingkat pencahayaan disajikan dalam Tabel 5.3.

Tabel 5.3 Hasil Pengujian Skenario dengan Pengaruh Tingkat Pencahayaan

| Video ID | Jenis Model | Pencahayaan | TP | FP | FN | Precision (%) | Recall (%) | F1-Score (%) | FPS |
|---|---|---|---|---|---|---|---|---|---|
| yt-Ew0cd2TRlgo | *Faster R-CNN Resnet-50 FPN* | Normal | 1434 | 1077 | 157 | 57.11 | 90.13 | 69.92 | 10.34 |
| | | Cerah | 1146 | 971 | 445 | 54.13 | 72.03 | 61.81 | 10.45 |
| | | Redup | 1009 | 487 | 582 | 67.45 | 63.42 | 65.37 | 10.33 |
| | *YOLOv8m* | Normal | 1415 | 487 | 176 | 74.4 | 88.94 | 81.02 | 42.67 |
| | | Cerah | 1162 | 435 | 429 | 72.76 | 73.04 | 72.9 | 41.86 |
| | | Redup | 1236 | 459 | 355 | 72.92 | 77.69 | 75.23 | 42.02 |
| RobotCam-Day-1_mp4 | *Faster R-CNN Resnet-50 FPN* | Normal | 157 | 78 | 39 | 66.81 | 80.1 | 72.85 | 10.45 |
| | | Cerah | 164 | 84 | 32 | 66.13 | 83.67 | 73.87 | 10.48 |
| | | Redup | 158 | 35 | 38 | 81.87 | 80.61 | 81.23 | 10.41 |
| | *YOLOv8m* | Normal | 195 | 2 | 1 | 98.98 | 99.49 | 99.24 | 42.08 |
| | | Cerah | 193 | 2 | 3 | 98.97 | 98.47 | 98.72 | 42.02 |
| | | Redup | 156 | 1 | 40 | 99.36 | 79.59 | 88.39 | 41.9 |
| RobotStore-Day-1_mp4 | *Faster R-CNN Resnet-50 FPN* | Normal | 228 | 6 | 2 | 97.44 | 99.13 | 98.28 | 10.35 |
| | | Cerah | 6 | 17 | 224 | 26.09 | 2.61 | 4.74 | 10.48 |
| | | Redup | 14 | 2 | 216 | 87.5 | 6.09 | 11.38 | 10.52 |
| | *YOLOv8m* | Normal | 229 | 1 | 1 | 99.57 | 99.57 | 99.57 | 41.71 |
| | | Cerah | 226 | 1 | 4 | 99.56 | 98.26 | 98.91 | 41.99 |
| | | Redup | 227 | 19 | 3 | 92.28 | 98.7 | 95.38 | 41.69 |
| yt-Le6KNI9YsH0 | *Faster R-CNN Resnet-50 FPN* | Normal | 542 | 541 | 189 | 50.05 | 74.15 | 59.76 | 10.22 |
| | | Cerah | 512 | 697 | 219 | 42.35 | 70.04 | 52.78 | 10.27 |
| | | Redup | 309 | 356 | 422 | 46.47 | 42.27 | 44.27 | 10.29 |
| | *YOLOv8m* | Normal | 565 | 275 | 166 | 67.26 | 77.29 | 71.93 | 41.3 |
| | | Cerah | 532 | 347 | 199 | 60.52 | 72.78 | 66.09 | 42.18 |
| | | Redup | 476 | 256 | 255 | 65.03 | 65.12 | 65.07 | 41.76 |

| Video ID | Jenis Model | Pencahayaan | TP | FP | FN | Precision (%) | Recall (%) | F1-Score (%) | FPS |
|---|---|---|---|---|---|---|---|---|---|
| RobotCam-Static-1-Day-1_mp4 | *Faster R-CNN Resnet-50 FPN* | Normal | 83 | 2 | 39 | 97.65 | 68.03 | 80.19 | 10.41 |
| | | Cerah | 95 | 7 | 27 | 93.14 | 77.87 | 84.82 | 10.35 |
| | | Redup | 3 | 0 | 119 | 100 | 2.46 | 4.8 | 10.36 |
| | *YOLOv8m* | Normal | 122 | 3 | 0 | 97.6 | 100 | 98.79 | 41.15 |
| | | Cerah | 121 | 4 | 1 | 96.8 | 99.18 | 97.98 | 42.24 |
| | | Redup | 67 | 1 | 55 | 98.53 | 54.92 | 70.53 | 42.02 |
| RobotStore-Part-2-Day-2_mp4 | *Faster R-CNN Resnet-50 FPN* | Normal | 65 | 55 | 24 | 54.17 | 73.03 | 62.2 | 10.6 |
| | | Cerah | 48 | 69 | 41 | 41.03 | 53.93 | 46.6 | 10.54 |
| | | Redup | 7 | 1 | 82 | 87.5 | 7.87 | 14.43 | 10.56 |
| | *YOLOv8m* | Normal | 68 | 32 | 21 | 68 | 76.4 | 71.96 | 41.64 |
| | | Cerah | 56 | 17 | 33 | 76.71 | 62.92 | 69.14 | 42.55 |
| | | Redup | 29 | 7 | 60 | 80.56 | 32.58 | 46.4 | 42.31 |

Berdasarkan Tabel 5.3, dapat dilihat bahwa tingkat pencahayaan memengaruhi performa deteksi kedua model, meskipun besarnya pengaruh berbeda pada setiap video. Secara umum, perubahan dari kondisi pencahayaan normal menjadi cerah maupun redup menyebabkan penurunan nilai Recall dan F1-Score, terutama pada video dengan kondisi lingkungan yang lebih kompleks. Namun demikian, YOLOv8m mampu mempertahankan nilai Precision yang relatif tinggi pada sebagian besar skenario karena menghasilkan jumlah False Positive (FP) yang lebih sedikit dibandingkan Faster R-CNN ResNet-50 FPN. Sebagai contoh, pada video RobotCam-Day-1_mp4, YOLOv8m memperoleh Precision sebesar 98,98%, 98,97%, dan 99,36% pada kondisi normal, cerah, dan redup, sedangkan Faster R-CNN memperoleh 66,81%, 66,13%, dan 81,87% pada kondisi yang sama.

Pada beberapa video, seperti RobotCam-Day-1_mp4 dan RobotStore-Day-1_mp4, kedua model masih mampu memberikan hasil deteksi yang baik pada kondisi pencahayaan normal. Akan tetapi, ketika pencahayaan diubah menjadi lebih cerah atau lebih redup, performa Faster R-CNN mengalami penurunan yang cukup signifikan. Sebagai contoh, pada RobotStore-Day-1_mp4, nilai F1-Score Faster R-CNN turun drastis dari 98,28% pada kondisi normal menjadi 4,74% pada kondisi cerah dan 11,38% pada kondisi redup akibat meningkatnya jumlah False Negative (FN). Sebaliknya, YOLOv8m tetap mampu mempertahankan performa yang tinggi dengan F1-Score masing-masing sebesar 99,57%, 98,91%, dan 95,38% pada ketiga kondisi pencahayaan tersebut.

Pada video dengan karakteristik yang lebih kompleks, seperti yt-Ew0cd2TRlgo, yt-Le6KNI9YsH0, dan RobotStore-Part-2-Day-2_mp4, penurunan kualitas pencahayaan menyebabkan jumlah True Positive (TP) menurun dan False Negative (FN) meningkat pada kedua model sehingga nilai Recall dan F1-Score ikut mengalami penurunan. Sebagai contoh, pada video yt-Le6KNI9YsH0, F1-Score YOLOv8m menurun dari 71,93% pada kondisi normal menjadi 66,09% pada kondisi cerah dan 65,07% pada kondisi redup. Sementara itu, Faster R-CNN mengalami penurunan yang lebih besar, yaitu dari 59,76% pada kondisi normal menjadi 52,78% pada kondisi cerah dan 44,27% pada kondisi redup. Hasil tersebut menunjukkan bahwa perubahan pencahayaan memberikan dampak yang lebih besar terhadap performa Faster R-CNN dibandingkan YOLOv8m.

Secara keseluruhan, YOLOv8m menunjukkan performa yang lebih konsisten dibandingkan Faster R-CNN ResNet-50 FPN pada berbagai kondisi pencahayaan. Hal ini

ditunjukkan oleh nilai Precision, Recall, dan F1-Score yang umumnya lebih tinggi pada hampir seluruh video uji, serta jumlah False Positive yang lebih sedikit sehingga mampu mempertahankan tingkat presisi yang tinggi. Selain itu, dari sisi kecepatan inferensi, YOLOv8m menghasilkan FPS sekitar 41–43 FPS, sedangkan Faster R-CNN hanya mencapai sekitar 10–11 FPS. Dengan demikian, YOLOv8m lebih sesuai untuk diterapkan sebagai sistem deteksi kebakaran berbasis video secara real-time, terutama pada lingkungan dengan kondisi pencahayaan yang bervariasi.

### 5.3 Analisis Perbandingan

Berdasarkan seluruh hasil pengujian yang telah dilakukan, mulai dari proses *tuning* hyperparameter hingga pengujian pada berbagai skenario, dapat dilakukan analisis terhadap kinerja model Faster R-CNN ResNet-50 FPN dan YOLOv8m dalam mendeteksi objek api dan asap pada video. Tahap *tuning* hyperparameter menunjukkan bahwa kedua model memiliki konfigurasi terbaik yang berbeda. Pada model Faster R-CNN ResNet-50 FPN, konfigurasi terbaik diperoleh menggunakan optimizer SGD dengan learning rate 0,1, yang menghasilkan mAP50 sebesar 75,57% dan mAP50-95 sebesar 43,10%. Sementara itu, pada model YOLOv8m, konfigurasi terbaik diperoleh menggunakan optimizer AdamW dengan learning rate 0,001, yang menghasilkan mAP50 sebesar 77,95% dan mAP50-95 sebesar 44,94%. Hasil tersebut menunjukkan bahwa kedua model mampu menghasilkan performa deteksi yang baik, namun YOLOv8m memperoleh nilai mAP yang sedikit lebih tinggi sehingga memiliki kemampuan deteksi yang lebih baik pada tahap evaluasi.

Pada tahap pengujian menggunakan berbagai video dengan karakteristik objek dan lingkungan yang berbeda, kedua model menunjukkan karakteristik performa yang berbeda. Secara umum, YOLOv8m menghasilkan nilai Precision, Recall, dan F1-Score yang lebih tinggi pada sebagian besar video dibandingkan Faster R-CNN ResNet-50 FPN. Hal ini disebabkan oleh kemampuan YOLOv8m dalam menghasilkan jumlah False Positive (FP) dan False Negative (FN) yang lebih sedikit sehingga proses deteksi menjadi lebih akurat. Sebagai contoh, pada video RobotCam-Day-1_mp4, YOLOv8m memperoleh F1-Score sebesar 99,24%, sedangkan Faster R-CNN memperoleh 72,85%. Hasil serupa juga terlihat pada video RobotStore-Day-1_mp4, di mana YOLOv8m mencapai F1-Score sebesar 99,57%, sedangkan Faster R-CNN memperoleh 98,28% pada kondisi pencahayaan normal. Meskipun demikian, Faster R-CNN masih mampu memberikan hasil deteksi yang baik pada beberapa kondisi tertentu, terutama pada video dengan objek yang relatif jelas dan jumlah objek yang tidak terlalu banyak.

Pengujian pada berbagai kondisi pencahayaan menunjukkan bahwa perubahan intensitas cahaya memberikan pengaruh terhadap performa kedua model. Pada beberapa video, perubahan dari kondisi normal menjadi cerah maupun redup menyebabkan jumlah True Positive (TP) menurun dan False Negative (FN) meningkat sehingga nilai Recall dan F1-Score ikut mengalami penurunan. Akan tetapi, YOLOv8m mampu mempertahankan performa yang lebih stabil dibandingkan Faster R-CNN. Sebagai contoh, pada video RobotStore-Day-1_mp4, Faster R-CNN mengalami penurunan F1-Score yang sangat signifikan dari 98,28% pada kondisi normal menjadi 4,74% pada kondisi cerah dan 11,38% pada kondisi redup. Sebaliknya, YOLOv8m tetap mampu mempertahankan F1-Score di atas 95% pada ketiga kondisi pencahayaan. Pada video yang lebih kompleks, seperti yt-Ew0cd2TRlgo, yt-Le6KNI9YsH0, dan RobotStore-Part-2-Day-2_mp4, kedua model sama-sama mengalami penurunan performa, namun penurunan pada YOLOv8m relatif lebih kecil dibandingkan Faster R-CNN sehingga menunjukkan kemampuan generalisasi yang lebih baik terhadap perubahan pencahayaan.

Selain akurasi deteksi, kecepatan inferensi juga menjadi faktor penting dalam implementasi sistem deteksi dini kebakaran berbasis video. Berdasarkan hasil pengujian, YOLOv8m mampu

melakukan inferensi dengan kecepatan sekitar 41–43 FPS, sedangkan Faster R-CNN ResNet-50 FPN hanya mencapai sekitar 10–11 FPS. Dengan demikian, YOLOv8m memiliki kecepatan inferensi sekitar empat kali lebih tinggi dibandingkan Faster R-CNN. Kecepatan tersebut memungkinkan YOLOv8m memproses video secara lebih responsif sehingga lebih sesuai untuk aplikasi deteksi dini kebakaran yang memerlukan pemrosesan secara real-time.

Secara keseluruhan, hasil penelitian menunjukkan bahwa YOLOv8m memiliki performa yang lebih unggul dibandingkan Faster R-CNN ResNet-50 FPN pada sistem deteksi dini kebakaran berbasis video. Keunggulan tersebut ditunjukkan oleh nilai mAP50 dan mAP50-95 yang sedikit lebih tinggi pada tahap *tuning* hyperparameter, nilai Precision, Recall, dan F1-Score yang umumnya lebih baik pada berbagai skenario pengujian, kemampuan yang lebih stabil dalam menghadapi perubahan kondisi pencahayaan, serta kecepatan inferensi yang jauh lebih tinggi. Di sisi lain, Faster R-CNN ResNet-50 FPN masih mampu memberikan hasil deteksi yang baik pada beberapa kondisi tertentu, terutama pada video dengan karakteristik objek yang lebih sederhana. Namun, model ini memiliki waktu inferensi yang lebih lama dan performanya cenderung lebih sensitif terhadap perubahan kondisi pencahayaan. Oleh karena itu, berdasarkan seluruh hasil pengujian yang telah dilakukan, YOLOv8m lebih direkomendasikan sebagai model untuk implementasi sistem deteksi dini kebakaran berbasis video secara real-time, sedangkan Faster R-CNN ResNet-50 FPN lebih sesuai digunakan pada aplikasi yang mengutamakan akurasi deteksi dan tidak memiliki batasan kecepatan pemrosesan yang ketat.

# BAB VI
# PENUTUP

# BAB VI
# PENUTUP

## 6.1 Kesimpulan

Berdasarkan hasil pengujian dan pembahasan yang telah didapatkan, diperoleh kesimpulan dari Tugas Akhir ini sebagai berikut:

1. Penelitian ini berhasil mengimplementasikan model Faster R-CNN ResNet-50 FPN dan YOLOv8m untuk mendeteksi objek api (*fire*) dan asap (*smoke*) pada data video. Tahapan penelitian meliputi pengumpulan dan anotasi dataset, pelatihan model menggunakan konfigurasi *hyperparameter* terbaik, serta pengujian model pada skenario kondisi pencahayaan (normal, cerah, dan redup). Performa kedua model dievaluasi menggunakan metrik mAP50, mAP50-95, Precision, Recall, F1-Score, dan Frames Per Second (FPS).

2. Hasil *tuning hyperparameter* menunjukkan bahwa konfigurasi terbaik untuk Faster R-CNN ResNet-50 FPN diperoleh menggunakan optimizer SGD dengan learning rate 0,1, yang menghasilkan mAP50 sebesar 75,57% dan mAP50-95 sebesar 43,10%. Sementara itu, konfigurasi terbaik untuk YOLOv8m diperoleh menggunakan optimizer AdamW dengan learning rate 0,001, yang menghasilkan mAP50 sebesar 77,95% dan mAP50-95 sebesar 44,94%. Hasil tersebut menunjukkan bahwa YOLOv8m memiliki performa deteksi yang sedikit lebih baik dibandingkan Faster R-CNN ResNet-50 FPN pada tahap evaluasi model.

3. Berdasarkan hasil pengujian pada skenario, YOLOv8m secara umum menghasilkan nilai Precision, Recall, dan F1-Score yang lebih tinggi dibandingkan Faster R-CNN ResNet-50 FPN. Selain itu, YOLOv8m menghasilkan jumlah False Positive (FP) dan False Negative (FN) yang lebih sedikit sehingga performanya lebih konsisten pada berbagai kondisi pencahayaan. Meskipun kedua model mengalami penurunan performa pada kondisi pencahayaan cerah maupun redup, penurunan yang terjadi pada YOLOv8m relatif lebih kecil dibandingkan Faster R-CNN ResNet-50 FPN, sehingga menunjukkan kemampuan generalisasi yang lebih baik terhadap perubahan kondisi lingkungan.

4. Dari sisi kecepatan inferensi, YOLOv8m mampu memproses video dengan kecepatan sekitar 41–43 FPS, sedangkan Faster R-CNN ResNet-50 FPN hanya mencapai sekitar 10–11 FPS. Dengan performa deteksi yang lebih baik, kemampuan yang lebih stabil pada berbagai skenario pengujian, serta kecepatan inferensi yang sekitar empat kali lebih tinggi, YOLOv8m lebih direkomendasikan untuk implementasi sistem deteksi dini kebakaran berbasis video secara real-time. Sementara itu, Faster R-CNN ResNet-50 FPN masih dapat menjadi alternatif pada aplikasi yang lebih mengutamakan kualitas deteksi dan tidak memiliki kebutuhan pemrosesan secara real-time.

## 6.2 Saran

Adapun beberapa hal yang penulis sarankan untuk dikembangkan pada penelitian berikutnya, yaitu:

1. Jumlah dan variasi dataset video perlu diperbanyak agar model memperoleh karakteristik objek api dan asap yang lebih beragam. Penambahan data sebaiknya mencakup berbagai kondisi lingkungan, seperti lokasi indoor dan outdoor, ukuran api yang berbeda, tingkat kepadatan asap, serta objek yang memiliki kemiripan visual dengan api atau asap (misalnya pantulan cahaya, lampu, kabut, dan uap), sehingga dapat mengurangi kesalahan deteksi (*false positive*) maupun kegagalan deteksi (*false negative*).

2. Proses pelatihan model dapat ditingkatkan dengan mengeksplorasi teknik augmentasi data yang lebih beragam, seperti pengaturan kecerahan, kontras, saturasi, rotasi, *motion blur*, maupun penambahan efek cuaca. Selain itu, optimasi hyperparameter yang lebih luas juga dapat dilakukan untuk memperoleh konfigurasi model yang memberikan performa deteksi lebih optimal pada berbagai kondisi pengujian.

3. Skenario pengujian dapat diperluas dengan mengevaluasi performa model pada kondisi yang lebih bervariasi, seperti perubahan cuaca, guncangan kamera, variasi resolusi video, perubahan jarak objek terhadap kamera, maupun penggunaan kamera dengan spesifikasi yang berbeda. Pengujian pada lingkungan nyata (*real-world environment*) juga perlu dilakukan untuk mengetahui tingkat keandalan model ketika dihadapkan pada kondisi operasional yang sesungguhnya.

4. Penelitian selanjutnya juga dapat membandingkan Faster R-CNN dan YOLOv8 dengan model deteksi objek lainnya, baik yang termasuk keluarga YOLO maupun arsitektur deteksi objek modern lainnya. Perbandingan tersebut diharapkan dapat memberikan gambaran yang lebih komprehensif mengenai model yang paling sesuai untuk sistem deteksi dini kebakaran berbasis video, baik dari aspek akurasi, kecepatan inferensi, maupun kebutuhan komputasi.

# DAFTAR PUSTAKA

# DAFTAR PUSTAKA


Alharthi, A., Alqahtani, M., & Alotaibi, S. (2025). *Comparative Analysis of YOLOv8 and Faster R-CNN for Fire Detection in Surveillance Videos*. IEEE Access.

BBC News Indonesia. (2019). *Kabut Asap: Titik Api di Kalimantan dan Sumatera Membara, Asap Tebal di Pekanbaru*. https://www.bbc.com/indonesia/indonesia-49700670. Diakses pada 8 Oktober 2024 pukul 03:15 PM.

Dogan, G., Sahin, H., & Aydin, M. (2022). Fire Detection Systems: A Review. Fire Technology, 58, 2047–2073. https://doi.org/10.1007/s10694-021-01230-3.

Fajri, M., Putra, A., & Nugroho, R. (2024). *Fire and Smoke Object Detection Using Mask R-CNN*. Journal of Robotics and Control.

Goodfellow, I., Y. Bengio dan A. Courville, Deep Learning, MIT Press, 2016.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

ImageNet. (2016). *ImageNet*. Retrieved from http://www.imagenet.org/. Diakses pada 8 Oktober 2024 pukul 05:00 PM.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, *35*(1), 221–231.

Kompas. (2020). *Kebakaran Gedung Kejaksaan Agung, Apa yang Terjadi?*. https://nasional.kompas.com/read/2020/08/23/06545851/kebakaran-gedung-kejaksaan-agung-apa-yang-terjadi. Diakses pada 8 Oktober 2024 pukul 01:00 PM.

Li, H., Chen, Y., & Zhao, W. (2024). *Real-Time Fire and Smoke Detection Based on YOLOv8 Deep Learning Model*. Applied Sciences.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2117–2125. https://doi.org/10.1109/CVPR.2017.106

Manning, C., Raghavan, P., & Schutze, H. (2009). *An Introduction to Information Retriveral.* Cambridge : Cambridge University Press.

Nair, V., dan G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair," Proceedings of ICML, vol. 27, pp. 807 - 814, 2010.

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

Patterson, J., & Gibson, A. (2017). *Deep Learning: A Practitioner's Approach* (1st ed.). United Syates of America: O'Reilly Media, Inc.

Pincott, T., Goonetilleke, A., & Sanderson, C. (2024). *Indoor Fire Detection Utilizing Computer Vision-Based Strategies*. Fire.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (NIPS), 28, 91-99.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 1137 - 1149.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2015). “Rethinking the Inception Architecture for Computer Vision”. arXiv:1512.00567

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818-2826.

Tempo. (2021). *Kilang Minyak Cilacap Terbakar, Pertamina Lakukan Evakuasi*. https://bisnis.tempo.co/read/1473696/kilang-minyak-cilacap-terbakar-pertamina-lakukan-evakuasi. Diakses pada 8 Oktober 2024 pukul 02:30 AM.

The MathWorks, Inc., Introducing Deep Learning With Matlab, The MathWorks, Inc., 2018.

Wahyuni, E. S., & Hendri, M. (2019). Smoke and Fire Detection Base on Convolutional Neural Network. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*.

Wu, S., Zhang, X., Liu, R., and Li, B.: A dataset for fire and smoke object detection, Multimed. Tools Appl., 82, 6707–6726, https://doi.org/10.1007/s11042-022-13580-x, 2023.

Yar, K., Alsaadi, F., & Ahmed, M. (2023). A Comparative Study of Sensor-based and Vision-based Fire Detection Systems. IEEE Access, 11, 14520–14535. https://doi.org/10.1109/ACCESS.2023.3241234.

Zhang, J. Li, D., J. Zhang, J. Zhang, T. Li, Y. Xia, Q. Yan dan L. Xun, “Facial Expression Recognition with Faster R-CNN,” Procedia Computer Science, vol. 107, pp. 135 - 140, 2017.

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2020). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

# DAFTAR LAMPIRAN

# DAFTAR LAMPIRAN

## Lampiran 1
## Tabel Hasil Pelatihan dan Evaluasi Model Terbaik Faster R-CNN

| epoch | Loss box reg | Loss cls | Loss rpn cls | Loss rpn loc | validation/ loss box reg | validation/ loss cls | validation/ loss rpn cls | validation/ loss rpn loc | validation/ AP50 | validation/ AP |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.178 | 0.1516 | 0.0375 | 0.0241 | 0.152 | 0.122 | 0.0259 | 0.0182 | 30.17 | 10.38 |
| 2 | 0.189 | 0.1207 | 0.0167 | 0.0209 | 0.120 | 0.082 | 0.0157 | 0.0151 | 63.36 | 28.04 |
| 3 | 0.157 | 0.1078 | 0.0138 | 0.0203 | 0.114 | 0.082 | 0.0127 | 0.0147 | 68.75 | 33.50 |
| 4 | 0.164 | 0.1044 | 0.0097 | 0.0176 | 0.112 | 0.075 | 0.0152 | 0.0141 | 72.74 | 35.79 |
| 5 | 0.144 | 0.0872 | 0.0079 | 0.0167 | 0.108 | 0.073 | 0.0129 | 0.0141 | 72.27 | 36.79 |
| 6 | 0.139 | 0.0886 | 0.0084 | 0.0157 | 0.109 | 0.078 | 0.0126 | 0.0141 | 69.67 | 36.78 |
| 7 | 0.130 | 0.0764 | 0.0079 | 0.0168 | 0.103 | 0.072 | 0.0132 | 0.0158 | 72.98 | 38.94 |
| 8 | 0.125 | 0.0738 | 0.0055 | 0.0147 | 0.105 | 0.076 | 0.0140 | 0.0149 | 73.79 | 38.25 |
| 9 | 0.121 | 0.0742 | 0.0059 | 0.0138 | 0.103 | 0.067 | 0.0171 | 0.0149 | 71.62 | 37.53 |
| 10 | 0.130 | 0.0835 | 0.0080 | 0.0182 | 0.103 | 0.077 | 0.0225 | 0.0147 | 70.65 | 36.81 |
| 11 | 0.120 | 0.0701 | 0.0071 | 0.0144 | 0.102 | 0.072 | 0.0179 | 0.0165 | 70.75 | 37.36 |
| 12 | 0.116 | 0.0739 | 0.0072 | 0.0170 | 0.100 | 0.067 | 0.0156 | 0.0153 | 70.91 | 36.88 |
| 13 | 0.116 | 0.0692 | 0.0075 | 0.0191 | 0.107 | 0.078 | 0.0162 | 0.0156 | 70.19 | 38.12 |
| 14 | 0.115 | 0.0691 | 0.0072 | 0.0145 | 0.103 | 0.068 | 0.0160 | 0.0150 | 71.73 | 37.39 |
| 15 | 0.113 | 0.0727 | 0.0070 | 0.0176 | 0.104 | 0.074 | 0.0191 | 0.0168 | 71.19 | 38.50 |
| 16 | 0.123 | 0.0714 | 0.0094 | 0.0220 | 0.102 | 0.068 | 0.0174 | 0.0159 | 73.92 | 38.67 |
| 17 | 0.118 | 0.0772 | 0.0104 | 0.0172 | 0.104 | 0.079 | 0.0178 | 0.0164 | 68.91 | 35.55 |
| 18 | 0.109 | 0.0613 | 0.0047 | 0.0163 | 0.101 | 0.073 | 0.0191 | 0.0139 | 73.76 | 40.98 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 19 | 0.085 | 0.0464 | 0.0026 | 0.0111 | 0.103 | 0.076 | 0.0202 | 0.0129 | 75.56 | 42.66 |
| 20 | 0.082 | 0.0401 | 0.0021 | 0.0099 | 0.102 | 0.082 | 0.0269 | 0.0127 | 74.39 | 41.93 |
| 21 | 0.085 | 0.0413 | 0.0020 | 0.0108 | 0.101 | 0.079 | 0.0275 | 0.0126 | 75.42 | 42.52 |
| 22 | 0.077 | 0.0380 | 0.0022 | 0.0095 | 0.101 | 0.081 | 0.0283 | 0.0125 | 73.99 | 43.03 |
| 23 | 0.078 | 0.0352 | 0.0015 | 0.0114 | 0.101 | 0.083 | 0.0274 | 0.0125 | 74.42 | 43.09 |
| 24 | 0.067 | 0.0329 | 0.0017 | 0.0081 | 0.102 | 0.086 | 0.0280 | 0.0125 | 74.31 | 42.73 |
| 25 | 0.066 | 0.0334 | 0.00149 | 0.00719 | 0.1024 | 0.0854 | 0.0276 | 0.0124 | 74.137 | 42.718 |

# Lampiran 2
# Tabel Hasil Pelatihan dan Evaluasi Model Terbaik YOLOv8

| epoch | train/ box loss | train/ cls loss | train/ dfl loss | metrics/ mAP50(B) | metrics/ mAP50-95(B) | val/ box loss | val/ cls loss | val/ dfl loss |
|---|---|---|---|---|---|---|---|---|
| 1 | 2.00927 | 2.81803 | 1.89498 | 0.06671 | 0.02011 | 2.5154 | 19.5598 | 2.568 |
| 2 | 1.87534 | 2.17528 | 1.81627 | 0.0598 | 0.02063 | 2.29096 | 20.2957 | 2.48417 |
| 3 | 1.84288 | 2.11496 | 1.82399 | 0.14956 | 0.08184 | 2.12609 | 9.52689 | 2.38998 |
| 4 | 1.83579 | 2.05525 | 1.831 | 0.10123 | 0.03968 | 2.12538 | 29.7359 | 2.16635 |
| 5 | 1.77106 | 1.98988 | 1.78276 | 0.32101 | 0.14825 | 1.8724 | 6.24498 | 1.86639 |
| 6 | 1.72941 | 1.90872 | 1.73705 | 0.42103 | 0.20777 | 1.75553 | 2.04821 | 1.76504 |
| 7 | 1.69182 | 1.81838 | 1.71575 | 0.45668 | 0.21923 | 1.77448 | 2.25082 | 1.78034 |
| 8 | 1.69469 | 1.84527 | 1.70654 | 0.40213 | 0.18936 | 1.80613 | 2.08725 | 1.80591 |
| 9 | 1.62773 | 1.75301 | 1.66814 | 0.38007 | 0.18781 | 1.73321 | 1.9709 | 1.74916 |
| 10 | 1.65183 | 1.71895 | 1.66117 | 0.58642 | 0.28742 | 1.61121 | 1.53338 | 1.66036 |
| 11 | 1.57752 | 1.64959 | 1.62621 | 0.4794 | 0.24827 | 1.57099 | 1.96585 | 1.66618 |
| 12 | 1.58051 | 1.66959 | 1.63296 | 0.53715 | 0.2928 | 1.527 | 1.51427 | 1.62204 |
| 13 | 1.54801 | 1.62304 | 1.6119 | 0.60716 | 0.31165 | 1.58821 | 1.52429 | 1.63789 |
| 14 | 1.56127 | 1.5692 | 1.60317 | 0.60534 | 0.32726 | 1.46131 | 1.40976 | 1.58589 |
| 15 | 1.53827 | 1.56475 | 1.57821 | 0.60435 | 0.32606 | 1.58152 | 1.49851 | 1.64076 |
| 16 | 1.49762 | 1.50659 | 1.55404 | 0.56485 | 0.29948 | 1.51282 | 1.72575 | 1.59975 |
| 17 | 1.51483 | 1.51622 | 1.56858 | 0.57576 | 0.30897 | 1.52737 | 1.78437 | 1.60181 |
| 18 | 1.47682 | 1.48401 | 1.56625 | 0.63484 | 0.34878 | 1.50065 | 1.47405 | 1.62219 |
| 19 | 1.47586 | 1.46517 | 1.53383 | 0.59813 | 0.32341 | 1.45875 | 1.52668 | 1.58483 |
| 20 | 1.45983 | 1.43447 | 1.51161 | 0.60331 | 0.31784 | 1.5213 | 1.46654 | 1.60172 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 21 | 1.46893 | 1.45567 | 1.53267 | 0.66639 | 0.3581 | 1.46274 | 1.32352 | 1.57568 |
| 22 | 1.44425 | 1.40804 | 1.52417 | 0.65549 | 0.35857 | 1.46711 | 1.45408 | 1.56689 |
| 23 | 1.40819 | 1.37758 | 1.49472 | 0.67794 | 0.36296 | 1.49638 | 1.33792 | 1.61871 |
| 24 | 1.44182 | 1.36888 | 1.52215 | 0.6249 | 0.33906 | 1.53081 | 1.43912 | 1.63239 |
| 25 | 1.41882 | 1.3642 | 1.49742 | 0.68786 | 0.3809 | 1.42284 | 1.22479 | 1.5675 |
| 26 | 1.39025 | 1.32659 | 1.48025 | 0.66601 | 0.37705 | 1.42926 | 1.29322 | 1.52898 |
| 27 | 1.38437 | 1.33833 | 1.48127 | 0.53244 | 0.28976 | 1.49688 | 1.62916 | 1.61656 |
| 28 | 1.37933 | 1.30224 | 1.48043 | 0.7018 | 0.40195 | 1.39175 | 1.21519 | 1.53998 |
| 29 | 1.35949 | 1.27104 | 1.45714 | 0.69047 | 0.38114 | 1.41322 | 1.21692 | 1.55269 |
| 30 | 1.34741 | 1.2605 | 1.44756 | 0.72293 | 0.41144 | 1.4128 | 1.13972 | 1.54528 |
| 31 | 1.35858 | 1.25604 | 1.45348 | 0.67617 | 0.37359 | 1.41562 | 1.23165 | 1.53417 |
| 32 | 1.33978 | 1.25131 | 1.44114 | 0.71463 | 0.40132 | 1.42214 | 1.11839 | 1.54549 |
| 33 | 1.33411 | 1.23735 | 1.44527 | 0.69767 | 0.38723 | 1.41875 | 1.21883 | 1.53122 |
| 34 | 1.31662 | 1.20201 | 1.4296 | 0.70582 | 0.40381 | 1.40207 | 1.11152 | 1.54423 |
| 35 | 1.31387 | 1.21448 | 1.44063 | 0.66424 | 0.37294 | 1.40083 | 1.22531 | 1.54583 |
| 36 | 1.28 | 1.18061 | 1.4095 | 0.70795 | 0.4008 | 1.40634 | 1.11807 | 1.54928 |
| 37 | 1.28151 | 1.17055 | 1.41778 | 0.70115 | 0.41272 | 1.35649 | 1.11671 | 1.51865 |
| 38 | 1.29578 | 1.19476 | 1.41351 | 0.7228 | 0.40393 | 1.39251 | 1.07997 | 1.54585 |
| 39 | 1.24675 | 1.15343 | 1.38013 | 0.70244 | 0.39944 | 1.38883 | 1.17012 | 1.54438 |
| 40 | 1.2553 | 1.12555 | 1.40603 | 0.69416 | 0.39185 | 1.38889 | 1.12336 | 1.50347 |
| 41 | 1.23352 | 1.11471 | 1.38115 | 0.6881 | 0.38781 | 1.37959 | 1.13414 | 1.55288 |
| 42 | 1.2329 | 1.12129 | 1.37423 | 0.75128 | 0.41419 | 1.39219 | 1.07052 | 1.54402 |
| 43 | 1.24095 | 1.10845 | 1.38718 | 0.72848 | 0.40829 | 1.41214 | 1.10873 | 1.57549 |
| 44 | 1.22176 | 1.09097 | 1.37784 | 0.73778 | 0.42749 | 1.3704 | 1.07063 | 1.52857 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 45 | 1.20153 | 1.05373 | 1.35696 | 0.72847 | 0.41249 | 1.3846 | 1.10497 | 1.53331 |
| 46 | 1.20471 | 1.08372 | 1.36219 | 0.73725 | 0.41841 | 1.37073 | 1.07014 | 1.54085 |
| 47 | 1.20073 | 1.05919 | 1.35918 | 0.73399 | 0.42675 | 1.34727 | 1.01672 | 1.51833 |
| 48 | 1.18271 | 1.03902 | 1.35185 | 0.73542 | 0.41174 | 1.37547 | 1.09751 | 1.54225 |
| 49 | 1.19235 | 1.05205 | 1.34599 | 0.7547 | 0.43645 | 1.34092 | 1.02145 | 1.5176 |
| 50 | 1.18444 | 1.05648 | 1.35253 | 0.70277 | 0.39296 | 1.3919 | 1.10608 | 1.5521 |
| 51 | 1.17165 | 1.01759 | 1.33721 | 0.74171 | 0.43029 | 1.34931 | 1.02924 | 1.55063 |
| 52 | 1.15009 | 0.98464 | 1.32789 | 0.73103 | 0.42997 | 1.34483 | 1.06676 | 1.51887 |
| 53 | 1.15029 | 0.98952 | 1.31308 | 0.75325 | 0.43236 | 1.32648 | 1.01505 | 1.49242 |
| 54 | 1.13861 | 0.96253 | 1.31012 | 0.72874 | 0.42052 | 1.33279 | 1.05512 | 1.51884 |
| 55 | 1.12648 | 0.98596 | 1.29714 | 0.75754 | 0.43775 | 1.32802 | 0.99771 | 1.5199 |
| 56 | 1.12382 | 0.97751 | 1.30509 | 0.69751 | 0.38643 | 1.38712 | 1.10886 | 1.56764 |
| 57 | 1.11222 | 0.95197 | 1.28938 | 0.72191 | 0.42011 | 1.32506 | 1.01631 | 1.52358 |
| 58 | 1.09198 | 0.93525 | 1.27114 | 0.73272 | 0.41687 | 1.32455 | 1.01203 | 1.53614 |
| 59 | 1.10597 | 0.92536 | 1.29039 | 0.71663 | 0.4032 | 1.35033 | 1.02664 | 1.53747 |
| 60 | 1.09951 | 0.96071 | 1.29648 | 0.74434 | 0.42657 | 1.34058 | 0.98539 | 1.53417 |
| 61 | 1.09192 | 0.93356 | 1.28115 | 0.73821 | 0.42467 | 1.32977 | 1.01136 | 1.52307 |
| 62 | 1.07359 | 0.89938 | 1.27806 | 0.73421 | 0.42073 | 1.36195 | 0.99085 | 1.54515 |
| 63 | 1.0755 | 0.88635 | 1.26523 | 0.74867 | 0.42673 | 1.33241 | 0.98917 | 1.54404 |
| 64 | 1.08792 | 0.91433 | 1.28171 | 0.76024 | 0.44225 | 1.33316 | 0.93511 | 1.5234 |
| 65 | 1.04051 | 0.89283 | 1.25556 | 0.72519 | 0.4267 | 1.33618 | 0.99321 | 1.53786 |
| 66 | 1.05496 | 0.87565 | 1.26236 | 0.74086 | 0.42406 | 1.32008 | 0.97106 | 1.5232 |
| 67 | 1.05186 | 0.85342 | 1.24601 | 0.74842 | 0.43626 | 1.32119 | 0.97062 | 1.52321 |
| 68 | 1.02547 | 0.86633 | 1.24399 | 0.75951 | 0.44406 | 1.30913 | 0.94505 | 1.52796 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 69 | 1.01357 | 0.82827 | 1.23118 | 0.75328 | 0.4373 | 1.33429 | 0.93989 | 1.55701 |
| 70 | 1.01425 | 0.84203 | 1.22966 | 0.76652 | 0.44731 | 1.3207 | 0.92331 | 1.54098 |
| 71 | 0.99554 | 0.80508 | 1.22326 | 0.75199 | 0.43839 | 1.32934 | 0.93373 | 1.53608 |
| 72 | 0.98671 | 0.8222 | 1.21387 | 0.75733 | 0.44719 | 1.29249 | 0.95443 | 1.51426 |
| 73 | 0.99593 | 0.82221 | 1.21253 | 0.7495 | 0.4375 | 1.33959 | 0.95965 | 1.55137 |
| 74 | 0.99672 | 0.80101 | 1.21282 | 0.77952 | 0.44942 | 1.31825 | 0.91304 | 1.53385 |
| 75 | 0.97089 | 0.76343 | 1.20399 | 0.74506 | 0.43139 | 1.32529 | 0.95165 | 1.56971 |
| 76 | 0.96322 | 0.76659 | 1.19962 | 0.75775 | 0.44433 | 1.32303 | 0.92981 | 1.56147 |
| 77 | 0.96019 | 0.76614 | 1.18623 | 0.76601 | 0.44285 | 1.32494 | 0.9225 | 1.55565 |
| 78 | 0.94429 | 0.76036 | 1.18245 | 0.76029 | 0.43521 | 1.32423 | 0.9421 | 1.55581 |
| 79 | 0.92621 | 0.73856 | 1.18153 | 0.75636 | 0.43872 | 1.31397 | 0.92074 | 1.56018 |
| 80 | 0.94554 | 0.74105 | 1.17821 | 0.76685 | 0.44516 | 1.29952 | 0.9248 | 1.5413 |
| 81 | 0.92017 | 0.73267 | 1.17778 | 0.75456 | 0.43011 | 1.32698 | 0.92355 | 1.57325 |
| 82 | 0.91188 | 0.72703 | 1.16848 | 0.76188 | 0.44352 | 1.3211 | 0.91119 | 1.57303 |
| 83 | 0.92035 | 0.72256 | 1.16857 | 0.75942 | 0.438 | 1.29521 | 0.91071 | 1.54528 |
| 84 | 0.90223 | 0.71446 | 1.16111 | 0.75881 | 0.43516 | 1.32476 | 0.91995 | 1.55824 |
| 85 | 0.88455 | 0.69331 | 1.14872 | 0.75766 | 0.43978 | 1.31552 | 0.91381 | 1.58567 |
| 86 | 0.89625 | 0.69892 | 1.15814 | 0.75179 | 0.4413 | 1.30939 | 0.91978 | 1.5707 |
| 87 | 0.8831 | 0.68188 | 1.14727 | 0.74975 | 0.43747 | 1.3149 | 0.93115 | 1.58459 |
| 88 | 0.8646 | 0.67696 | 1.1458 | 0.75572 | 0.43986 | 1.31358 | 0.91892 | 1.59293 |
| 89 | 0.85644 | 0.66616 | 1.13247 | 0.75767 | 0.43496 | 1.31602 | 0.92629 | 1.59985 |
| 90 | 0.84275 | 0.65173 | 1.12114 | 0.75413 | 0.43502 | 1.32822 | 0.90016 | 1.60218 |
| 91 | 0.8365 | 0.61724 | 1.13876 | 0.75078 | 0.43562 | 1.31363 | 0.90843 | 1.60102 |
| 92 | 0.78795 | 0.56328 | 1.09475 | 0.75407 | 0.4374 | 1.31605 | 0.89943 | 1.59251 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 93 | 0.77703 | 0.55633 | 1.09139 | 0.7555 | 0.4363 | 1.30965 | 0.91321 | 1.60531 |
| 94 | 0.77615 | 0.55144 | 1.08959 | 0.75329 | 0.43625 | 1.32291 | 0.89976 | 1.61967 |
| 95 | 0.75981 | 0.54501 | 1.08577 | 0.74932 | 0.43425 | 1.31828 | 0.92619 | 1.61967 |
| 96 | 0.75082 | 0.53484 | 1.07819 | 0.75457 | 0.44026 | 1.30817 | 0.9094 | 1.61736 |
| 97 | 0.75391 | 0.5351 | 1.07609 | 0.7657 | 0.44167 | 1.31174 | 0.90742 | 1.61705 |
| 98 | 0.74316 | 0.52928 | 1.06578 | 0.76444 | 0.44307 | 1.31252 | 0.90942 | 1.62378 |
| 99 | 0.72226 | 0.51117 | 1.05583 | 0.76123 | 0.44216 | 1.31113 | 0.90867 | 1.62556 |
| 100 | 0.72989 | 0.51232 | 1.06299 | 0.75816 | 0.44032 | 1.31364 | 0.91844 | 1.62755 |

# UCAPAN TERIMA KASIH

# UCAPAN TERIMA KASIH

Penyelesaian penulisan tugas akhir ini tidak lepas dari orang-orang terdekat penulis yang telah mendukung, mengkritisi, mendorong dan memotivasi penulis sehingga penyusunan Tugas Akhir ini dapat terselesaikan. Oleh sebab itu, penulis mengucapkan terima kasih kepada:

1. Bapak Taryana dan Ibu Evvy Andriana selaku Orang Tua penulis yang telah membesarkan penulis, selalu bersabar mendidik, dan selalu mendoakan penulis agar dapat menyelesaikan Tugas Akhir ini.

2. Ryan, Okky, dan Dika sebagai Saudara kandung penulis yang telah memberikan dukungan dan kepedulian, serta berperan penting dalam kehidupan penulis sehingga memberikan semangat hidup bagi penulis.

3. Teman-teman Lab yang telah memberikan banyak masukan, ide, serta partner berkonsultasi bagi penulis dalam proses pengerjaan Tugas Akhir ini.

4. Teman-teman angkatan Matematika 2020 ITS, yang memberikan pengalaman tak terlupakan bagi penulis dalam kehidupan perkuliahan.

Penulis juga mengharapkan kritik dan saran yang membangun dari berbagai pihak untuk penyempurnaan isi tugas akhir ini serta demi perkembangan ilmu pengetahuan. Akhir kata, semoga tugas akhir ini bermanfaat bagi semua pihak yang bersangkutan.

Surabaya, 3 Agustus 2026

Aditya Dwi Gumara

# BIODATA PENULIS

# BIODATA PENULIS

**Aditya Dwi Gumara** atau biasa dipanggil Adit lahir di Jakarta, 4 Maret 2002. Penulis menempuh pendidikan di SDN 15 Pagi Kemayoran (2008-2014), SMPN 10 Jakarta (2014-2017), dan SMAN 27 Jakarta (2017-2020). Saat ini penulis menempuh pendidikan S1 di Departemen Matematika ITS sejak Tahun 2020. Penulis aktif mengikuti kegiatan organisasi dan pelatihan. Diantaranya adalah sebagai Anggota UKM Voli ITS periode 2020/sekarang, lalu sebagai Anggota UKM Kendo ITS periode 2020/sekarang, Anggota OK2BK (Orientasi Keprofesian dan Kompetensi Berbasis Kurikulum) HIMATIKA ITS periode 2020/2021, Anggota Gerakan Integralistik (GERIGI) ITS periode 2020/2021, serta Anggota Scientific Class ASCI HIMATIKA ITS periode 2021/2022. Selama kuliah penulis mengambil bidang Pemrograman dan Komputasi Visual (PROVIKOM).

Jika pembaca ingin mendapat informasi lebih lanjut serta menyampaikan kritik dan saran terkait laporan ini, pembaca dapat menghubungi penulis melalui e-mail penulis di adityadwigumara80@gmail.com. Semoga laporan ini memberikan informasi baru dan bermanfaat bagi para pembaca.