

**IMPLEMENTASI YOLO V8 UNTUK DETEKSI KANTUK PADA
PENGEMUDI KENDARAAN RODA EMPAT
SECARA *REAL-TIME***

SKRIPSI

FILDZAH ZATA AMANI NASUTION

201402030



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**IMPLEMENTASI YOLO V8 UNTUK DETEKSI KANTUK PADA
PENGEMUDI KENDARAAN RODA EMPAT
SECARA *REAL-TIME***

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Teknologi Informasi

FILDZAH ZATA AMANI NASUTION

201402030



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PERSETUJUAN

Judul : IMPLEMENTASI YOLO V8 UNTUK DETEKSI
KANTUK PADA PENGEMUDI KENDARAAN
RODA EMPAT SECARA *REAL-TIME*

Kategori : SKRIPSI

Nama : FILDZAH ZATA AMANI NASUTION

Nomor Induk Mahasiswa : 201402030

Program Studi : SARJANA (S1) TEKNOLOGI INFORMASI

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Medan, 16 Oktober 2024

Komisi Pembimbing :

Pembimbing 2



Dedy Arisandi S.T., M.Kom.

NIP. 197908312009121002

Pembimbing 1



Ivan Jaya S.Si., M.Kom.

NIP. 198407072015041001

Diketahui/disetujui oleh

Program Studi S1 Teknologi Informasi

Ketua,



Dedy Arisandi S.T., M.Kom.

NIP. 197908312009121002

PERNYATAAN

IMPLEMENTASI YOLO V8 UNTUK DETEKSI KANTUK PADA PENGEMUDI KENDARAAN RODA EMPAT SECARA *REAL-TIME*

SKRIPSI

Saya mengakui bahwa skripsi ini merupakan hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 16 Oktober 2024



Fildzah Zata Amani Nasution

201402030

UCAPAN TERIMA KASIH

Puji syukur kita panjatkan kehadirat Allah SWT, yang telah memberikan rahmat, hidayah, serta karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan judul “Implementasi YOLO V8 untuk Deteksi Kantuk pada Pengemudi Kendaraan Roda Empat Secara *Real-Time*”. Shalawat dan salam semoga tercurahkan kepada Nabi Muhammad SAW, keluarga, sahabat, dan pengikutnya yang setia.

Skripsi ini adalah salah satu syarat untuk mendapatkan sebuah gelar Sarjana Komputer pada Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Meskipun mengalami beberapa kendala selama proses penulisan, penulis berhasil mengatasi tantangan tersebut dengan sukses berkat dukungan dan panduan dari berbagai pihak.

Untuk berbagai pihak yang terlibat dalam penyelesaian skripsi ini, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Kepada kedua orang tua penulis, Abah Sjachrizal Indra Mulia Nst dan Ummi Ria Santika yang sudah membesarkan dan mendidik penulis serta tanpa henti memberikan doa, dukungan, dan motivasi.
2. Bapak Dr. Muryanto Amin, S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
3. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi.
4. Bapak Dedy Arisandi S.T., M.Kom. selaku Ketua Program Studi Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing II saya yang sudah meluangkan waktu, pemikiran, motivasi, kritik dan memberikan saran kepada penulis dalam menyelesaikan skripsi ini.
5. Bapak Ivan Jaya S.Si., M.Kom. selaku Dosen Pembimbing I saya yang telah memberikan banyak waktu, pemikiran, motivasi, kritik, dan saran untuk membantu penulis dalam menyelesaikan skripsi ini.
6. Bapak dan Ibu Dosen Fasilkom-TI USU yang telah memberikan ilmu baik di kelas perkuliahan maupun kegiatan akademik lainnya.
7. Seluruh staff Fasilkom-TI USU yang telah membantu dalam segala urusan administrasi selama masa perkuliahan.

8. Sahabat anti gedor, Faradila Hafiza, Amalia Rizkinta, Salsabila, Syifa Annisa, Lailan Sabila, dan Hanifah Luthfi yang sudah menjadi pendengar yang baik di saat susah maupun senang dan tidak henti memberikan semangat dan motivasi kepada penulis sejak masih di sekolah menengah atas hingga sekarang.
9. Teman-teman grup harem, Arbani, Irwansyah, Teruna, Retno, Syavira, Ali, dan Geby yang selalu memberikan hiburan selama masa perkuliahan. Kepada Dara Fadilah yang sudah menemani penulis dari masa perkuliahan, awal penyusunan tugas akhir, bimbingan dengan dosen pembimbing, hingga sekarang.
10. Teman-teman penulis, Iqbal Manalu, Vanissya Arbashika, Ivan Tandella, Tsabitah Muflihza, Nayla Rahmi, Ullayya Zhafirah yang selalu menemani penulis dari saat masih menjadi mahasiswa baru sampai masa penulisan dan penyusunan tugas akhir setiap harinya. Kepada Monica, Pretty, Jane, Arief, Zhafran, Yeftha, Ridha, Afifan, Zidan, Kak Andini, Wahyu, Putri Nahampun, Stephani, dan Manda, dan Yericho yang turut memberikan semangat dan motivasi kepada penulis dalam masa perkuliahan dan dalam menyelesaikan tugas akhir.
11. Teman-teman seperjuangan Teknologi Informasi Angkatan 2020 terutama Kom C, abang kakak Angkatan 2018 dan 2019, serta adik-adik 2021 dan 2022 yang sudah banyak membantu selama masa perkuliahan yang tidak dapat disebutkan satu persatu.
12. Keluarga penulis, Bunda Dini, Ayah Tamsil, Bang Mehta, Atok Fuad, Pak Windy, Bu Difa, dan yang lainnya yang tidak dapat penulis sebutkan satu persatu.
13. Dan terakhir kepada diri penulis, Fildzah Zata Amani Nst. Terima kasih sudah selalu berusaha untuk memberikan yang terbaik dan bertahan sejauh ini. Terima kasih untuk tidak pernah menyerah dalam keadaan apapun selama masa penyusunan tugas akhir hingga dapat menyelesaikannya dengan baik.

Semoga Allah SWT memberikan balasan yang berlipat ganda kepada setiap orang yang telah membantu dalam penulisan dan penyelesaian tugas akhir ini.

Medan, 16 Oktober 2024



Fildzah Zata Amani Nasution

ABSTRAK

Mengantuk saat berkendara (*sleepy driving*) termasuk dalam penyebab utama terjadinya kecelakaan lalu lintas. Diperkirakan dalam beberapa tahun terakhir, 20-30% kecelakaan di jalan raya disebabkan oleh mengantuk saat berkendara. Mengemudi dalam kondisi kurang tidur (< 6 jam) dikaitkan dengan peningkatan resiko kecelakaan hingga 4-5 kali lipat. Dari Badan Pusat Statistik (BPS), kecelakaan lalu lintas di Indonesia meningkat dari 100.028 kejadian di 2020 menjadi 103.645 di 2021. Oleh karenanya, sistem deteksi kantuk pada pengemudi kendaraan roda empat dibangun menggunakan algoritma You Only Look Once (YOLO) versi 8 dengan mempertimbangkan aktivitas pergerakan kepala, mata, dan mulut. Data pelatihan pada model penelitian menggunakan data citra pengemudi kendaraan roda empat yang diambil dari *roboflow* dan *kaggle* sedangkan data uji dikumpulkan mandiri secara *real-time*. Model terbaik didapatkan dengan menggunakan teknik *k-cross validation* dengan $k=5$ dan kombinasi *hyperparameter fold 3 epoch 250* dan *batch size 16*. Hasil pelatihan model diimplementasikan ke dalam aplikasi mobile berbasis android dengan tingkat akurasi mencapai 95% menggunakan metode perhitungan *confusion matrix*. Sistem aplikasi dapat memunculkan *bounding box* dengan waktu tunda sebesar 226 ms sampai 366 ms atau 0,2 hingga 0,3 detik pada perangkat dengan spesifikasi kamera depan 12 MP. Pengujian menghasilkan hasil yang maksimal dengan menggunakan perangkat dengan spesifikasi kamera depan 12MP ke atas dibandingkan dengan perangkat dengan spesifikasi kamera depan hanya 5MP.

Kata kunci : Deteksi Kantuk, *Microsleeping*, *Sleepy Driving*, YOLOv8

IMPLEMENTATION OF YOLO V8 FOR DROWSINESS DETECTION IN FOUR-WHEELED VEHICLE DRIVERS IN REAL-TIME

ABSTRACT

Sleepy driving is one of the main causes of traffic accidents. It is estimated that in recent years, 20-30% of road accidents are caused by sleepy driving. Sleep-deprived driving (<6 hours) is associated with a 4-5 times increased risk of accidents. According to the Central Bureau of Statistics (BPS), traffic accidents in Indonesia increased from 100,028 in 2020 to 103,645 in 2021. Therefore, a drowsiness detection system for four-wheeled vehicle drivers was built using the You Only Look Once (YOLO) algorithm version 8 by considering head, eye, and mouth movement activities. The training data in the research model uses four-wheeled vehicle driver image data taken from roboflow and kaggle while the test data is collected independently in real-time. The best model was obtained using the k-cross validation technique with $k=5$ and a combination of hyperparameter fold 3 epoch 250 and batch size 16. The model training results were implemented into an android-based mobile application with an accuracy rate of 95% using the confusion matrix calculation method. The application system can display the bounding box with a delay time of 226 ms to 366 ms or 0.2 to 0.3 seconds on devices with 12 MP front camera specifications. Testing produces maximum results using devices with front camera specifications of 12MP and above compared to devices with front camera specifications of only 5 MP.

Keywords: *Drowsiness Detection, Microsleeping, Sleepy Driving, YOLOv8*

DAFTAR ISI

| | |
|--|-------------|
| PERSETUJUAN | iii |
| PERNYATAAN | iv |
| UCAPAN TERIMA KASIH | v |
| ABSTRAK | vii |
| <i>ABSTRACT</i> | viii |
| DAFTAR ISI | ix |
| DAFTAR TABEL | xii |
| DAFTAR GAMBAR | xiii |
| BAB 1 PENDAHULUAN | 1 |
| 1.1. Latar Belakang | 1 |
| 1.2. Rumusan Masalah | 3 |
| 1.3. Tujuan Penelitian | 3 |
| 1.4. Batasan Masalah | 4 |
| 1.5. Manfaat Penelitian | 4 |
| 1.6. Metodologi Penelitian | 4 |
| 1.7. Sistematika Penulisan | 5 |
| BAB 2 LANDASAN TEORI | 7 |
| 2.1. <i>Microsleeping</i> | 7 |
| 2.2. <i>Real-Time Object Detection</i> | 7 |
| 2.3. <i>Citra Grayscale</i> | 7 |
| 2.4. <i>Convolutional Neural Network (CNN)</i> | 8 |
| 2.5. <i>You Only Look Once (YOLO)</i> | 11 |
| 2.6. <i>YOLOv8</i> | 12 |
| 2.7. <i>Confusion Matrix</i> | 14 |
| 2.7.1. <i>Accuracy</i> | 14 |
| 2.7.2. <i>Precision</i> | 15 |
| 2.7.3. <i>Recall</i> | 15 |
| 2.7.4. <i>F1-Score</i> | 15 |

| | |
|---|-----------|
| 2.8. Penelitian Terdahulu | 16 |
| 2.9. Perbedaan Penelitian | 19 |
| BAB 3 ANALISIS DAN PERANCANGAN SISTEM | 21 |
| 3.1. Dataset | 21 |
| 3.2. Analisis Sistem | 21 |
| 3.3. <i>Data Acquisition</i> | 22 |
| 3.4. <i>Preprocessing Data</i> | 22 |
| 3.4.1. <i>Image Anotationing</i> | 23 |
| 3.4.2. <i>Cropping</i> | 24 |
| 3.4.3. <i>Resizing</i> | 25 |
| 3.4.4. <i>Grayscaleing</i> | 26 |
| 3.5. <i>Processing Model</i> | 27 |
| 3.5.1. <i>Package Installation</i> | 27 |
| 3.5.2. <i>Data Configuration</i> | 27 |
| 3.5.3. <i>Pelatihan Model</i> | 28 |
| 3.6. <i>Learned Model</i> | 29 |
| 3.7. TF Lite Model | 30 |
| 3.8. Deployment | 30 |
| 3.8.1. <i>Model Integration to Mobile App</i> | 30 |
| 3.8.2. <i>Detection</i> | 30 |
| 3.9. <i>Output</i> | 30 |
| 3.10. <i>System Interface</i> | 31 |
| 3.10.1. <i>Activity Diagram</i> | 31 |
| 3.10.2. <i>Rancangan Halaman Splash Screen</i> | 31 |
| 3.10.3. <i>Rancangan Halaman Scan</i> | 32 |
| 3.10.4. <i>Rancangan Halaman Pop Up Pemberitahuan</i> | 32 |
| BAB 4 IMPLEMENTASI DAN PENGUJIAN | 34 |
| 4.1. Implementasi Sistem | 34 |
| 4.2. Implementasi Data | 35 |
| 4.3. Implementasi Model | 35 |

| | |
|--|-----------|
| 4.4. Implementasi Antarmuka Aplikasi | 40 |
| 4.4.1. Antarmuka Halaman <i>Splash Screen</i> | 40 |
| 4.4.2. Antarmuka Halaman <i>Scan</i> | 40 |
| 4.4.3. Antarmuka Halaman <i>Pop Up</i> Pemberitahuan | 41 |
| 4.5. Pengujian Sistem | 42 |
| BAB 5 KESIMPULAN DAN SARAN | 56 |
| 5.1. Kesimpulan | 56 |
| 5.2. Saran | 56 |
| DAFTAR PUSTAKA | 58 |

DAFTAR TABEL

| | |
|--|----|
| Tabel 2.1 Confusion Matrix | 14 |
| Tabel 2.2 Penelitian Terdahulu | 17 |
| Tabel 4.1 Hasil Percobaan 100 <i>epoch</i> | 36 |
| Tabel 4.2 Nilai <i>average</i> hasil percobaan dengan 3 <i>batch size</i> berbeda | 37 |
| Tabel 4.3 Hasil Percobaan 100 <i>epoch</i> K=5 | 37 |
| Tabel 4.4 Hasil percobaan 250 <i>epoch</i> K=5 | 38 |
| Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi | 42 |
| Tabel 4.6 Pengujian <i>Bounding Box</i> berdasarkan tingkat pencahayaan | 52 |
| Tabel 4.7 Hasil pengujian deteksi kantuk pada pengemudi | 53 |
| Tabel 4.8 Hasil perhitungan confusion matrix pada pengujian | 53 |
| Tabel 4.9 Nilai Precision, Recall, dan F1-Score | 54 |

DAFTAR GAMBAR

| | |
|--|----|
| Gambar 2.1 Citra <i>Grayscale</i> | 8 |
| Gambar 2.2 Arsitektur Convolutional Neural Network (CNN) | 9 |
| Gambar 2.3 Proses Konvolusi dengan Dua Filter | 9 |
| Gambar 2.4 Metode Max Pooling | 10 |
| Gambar 2.5 Sistem Deteksi YOLO | 11 |
| Gambar 2.6 Arsitektur YOLOv8 | 13 |
| Gambar 3.1 Visualisasi Citra Pengemudi | 21 |
| Gambar 3.2 Arsitektur Umum | 22 |
| Gambar 3.3 Membuat <i>Bounding Box</i> | 23 |
| Gambar 3.4 Memilih <i>Class</i> | 23 |
| Gambar 3.5 Isi File.txt | 24 |
| Gambar 3.6 Proses <i>Cropping</i> | 25 |
| Gambar 3.7 Proses <i>Resizing</i> | 25 |
| Gambar 3.8 Proses <i>Grayscale</i> | 27 |
| Gambar 3.9 File data.yaml | 28 |
| Gambar 3.10 Activity Diagram Sistem Aplikasi | 31 |
| Gambar 3.11 Rancangan Halaman <i>Splash Screen</i> | 32 |
| Gambar 3.12 Rancangan Halaman <i>Scan</i> | 32 |
| Gambar 3.13 Rancangan Halaman <i>Pop Up</i> Pemberitahuan | 33 |
| Gambar 4.1 Contoh Data Citra Pengendara Motor | 35 |
| Gambar 4.2 Grafik <i>fold 2 100 epoch K=5</i> | 38 |
| Gambar 4.3 Grafik <i>fold 3 250 epoch batch size 16</i> | 39 |
| Gambar 4.4 Grafik <i>train dan val fold 3 250 epoch batch size 16</i> | 39 |
| Gambar 4.5 Antarmuka Halaman <i>Splash Screen</i> | 40 |
| Gambar 4.6 Antarmuka Halaman <i>Scan</i> | 41 |
| Gambar 4.7 Antarmuka Halaman <i>Pop Up</i> Pemberitahuan | 41 |

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Mengantuk saat berkendara (*sleepy driving*) merupakan salah satu penyebab utama terjadinya kecelakaan lalu lintas. Berdasarkan analisis kecelakaan lalu lintas beberapa tahun terakhir, diperkirakan sekitar 20-30% kecelakaan di jalan raya disebabkan oleh pengemudi yang mengantuk di balik kemudi (Sinha et al., 2021).

Mengemudi dalam kondisi mengantuk dapat menurunkan kewaspadaan, memperlambat reaksi, dan mengaburkan penglihatan pengemudi sehingga risiko fatal akibat *human error* pun meningkat drastis (Schutte & Maldonado, 2003). Bahkan mengemudi dalam kondisi kurang tidur (<6 jam) dikaitkan dengan peningkatan risiko kecelakaan hingga 4-5 kali lipat (Sprajcer et al., 2023).

Berdasarkan data dari Badan Pusat Statistik (BPS), kecelakaan lalu lintas di Indonesia pada tahun 2020 mencapai 100.028 kejadian dan meningkat menjadi 103.645 kejadian pada tahun 2021 (Santika, 2023). Sebagian besar kecelakaan lalu lintas tersebut disebabkan oleh faktor kelalaian pengemudi, di mana mengantuk menduduki peringkat tertinggi kedua penyebab kecelakaan akibat kelalaian pengemudi setelah mabuk.

Meskipun angka kecelakaan lalu lintas yang disebabkan oleh kelalaian pengemudi dengan mengantuk masih terhitung tinggi setiap tahunnya, sampai saat ini belum banyak teknologi alternatif yang dapat secara akurat mendeteksi tanda-tanda mengantuk pada pengemudi mobil secara dini. Beberapa sensor dan kamera pada kendaraan hanya mendeteksi penyimpangan perilaku mengemudi akibat mengantuk, yang terkadang sudah terlambat. Oleh karenanya pengembangan teknologi untuk mencegah dan mendeteksi *sleepy driving* masih sangat dibutuhkan guna mengurangi insiden kecelakaan di jalan raya yang berakibat fatal.

Terdapat penelitian terkait yang dilakukan oleh Uma Maheswari et al., (2022) yang berjudul *Driver Drowsiness Prediction Based on Multiple Aspects Using Image Processing Techniques*. Peneliti menggunakan algoritma *Convolutional Neural Network* (CNN) dengan mempertimbangkan EAR dan MAR *calculations* dan *datasets* NTHU-DD, YawDD, dan EMOCDS (*Eye and Mouth Open Close Data Set*). Penelitian berhasil mencapai kesimpulan bahwa metode yang diajukan berhasil mengungguli akurasi

metode lain seperti KNN dan FaceNeT + SVM dengan mencapai tingkat akurasi sebesar 94.78% untuk penggunaan dataset YawDD, 79,98% untuk dataset NTHU-DD dan 95.67% untuk penggunaan dataset EMOCDS.

Penelitian lainnya dilakukan oleh Amzar et al., (2022) dengan judul *Driver Drowsiness Detection and Monitoring System* (DDDMS). Penelitian dilakukan untuk mengembangkan sistem monitoring pada pengemudi dengan memanfaatkan aktifitas kedipan mata, analisis, frekuensi menguap, estimasi pergerakan daerah kepala, dan pelacakan pada mata. Penelitian ini menggunakan metode SVM, HMM dan CNN dan berhasil mencapai tingkat akurasi 80.9% untuk deteksi kedipan mata, 96.3% untuk deteksi mata tertutup, dan 69.6% untuk deteksi menguap.

Penelitian lainnya dilakukan oleh N. Anjeana & Anusudha (2023) dengan judul *real time face recognition system based on YOLO and InsightFace*. Penelitian ini menggunakan custom dataset yang berisikan 12 kelas dengan setiap kelasnya terdapat 300 gambar wajah dengan berbagai pola. Metode yang digunakan adalah YOLO V7 dan InsightFace Model. Hasil yang dicapai pada penelitian ini membuktikan bahwa model algoritma yang diajukan dapat mengungguli kecepatan dan akurasi metode algoritma *face recognition* yang sudah ada. Metode ini juga dapat mengenali wajah pada berbagai kondisi efek cahaya dan wajah pada saat terhalangi suatu objek.

Metode YOLO (*You Only Look Once*) telah menjadi salah satu model deteksi objek yang paling populer dalam beberapa tahun terakhir. Model ini telah berkembang hingga versi terbarunya yaitu YOLOv8 yang dirilis pada Desember 2022. Seri YOLO menunjukkan potensi besar dalam pengenalan fitur wajah dibandingkan dengan metode multi-tahap seperti RCNN dan Fast-RCNN karena seri YOLO menawarkan kecepatan pendeteksian yang lebih cepat. YOLOv8, sebagai model terbaru dalam seri YOLO, mengatasi beberapa keterbatasan seperti kecepatan pendeteksian dan kinerja *real-time*, serta menyediakan fitur multi-skala dan model yang lebih sederhana dibanding versi YOLO sebelumnya.

Salah satu keunggulan utama YOLOv8 adalah pada *task* deteksi wajah. Berdasarkan pengujian menggunakan dataset *Wider Face*, akurasi YOLOv8 untuk mendeteksi wajah mencapai 91%. Angka ini jauh lebih tinggi dibandingkan dengan akurasi YOLOv7 yang hanya 84% pada dataset yang sama. Selain itu, kecepatan inferensi YOLOv8 juga terbilang sangat cepat, yaitu 83 FPS pada perangkat jetson sedangkan YOLOv7 hanya 72 FPS. Performa tinggi ini sangat cocok untuk aplikasi real-time. Dengan dua keunggulan tersebut, YOLOv8 berpotensi menjadi model deteksi wajah yang lebih baik daripada YOLOv7 (Hidayatulloh, 2021).

Berdasarkan penelitian terdahulu yang telah dibahas di atas, maka penulis melakukan penelitian dengan judul **“Implementasi YOLO V8 Untuk Deteksi Kantuk Pada Pengemudi Kendaraan Roda Empat Secara *Real-Time*”**. Parameter yang digunakan penulis dalam mendeteksi kantuk pada pengemudi kendaraan roda empat adalah aktivitas pergerakan mata, mulut, dan kepala. Penelitian ini akan menganalisis performa YOLO versi 8 dalam mendeteksi kantuk pada pengemudi.

1.2. Rumusan Masalah

Mengantuk saat mengemudi masih menjadi salah satu penyebab utama tingginya angka kecelakaan lalu lintas, namun teknologi yang ada saat ini masih belum mampu mendeteksi tanda-tanda mengantuk pada pengemudi secara akurat dan responsif. Sensor serta kamera pada kendaraan baru bisa memantau adanya penyimpangan pola mengemudi akibat kantuk yang terkadang sudah terlambat. Oleh karena itu, diperlukan pengembangan sistem yang lebih canggih dengan kemampuan pemantauan dan deteksi dini kondisi mengantuk pengemudi secara *real-time*, sehingga dapat memberikan peringatan lebih awal untuk mencegah terjadinya kecelakaan lalu lintas yang fatal.

1.3. Tujuan Penelitian

Tujuan dilakukannya penelitian ini adalah mengimplementasikan metode YOLO versi 8 dan melakukan analisis performanya untuk sistem deteksi kantuk pada pengemudi kendaraan roda empat secara *real-time*.

1.4. Batasan Masalah

Agar penelitian ini memiliki fokus yang lebih jelas dan tidak menyimpang dari tujuan yang telah ditetapkan, diperlukan adanya batasan-batasan dalam penelitian ini, yaitu sebagai berikut:

- 1) Data yang digunakan untuk *training* berupa citra dengan ekstensi file JPG.
- 2) Dataset yang digunakan untuk melatih sistem diambil dari *roboflow* dan *kaggle*.
- 3) Proses deteksi menggunakan data berupa citra pengemudi kendaraan roda empat yang diambil secara *real-time* dan tidak sedang menggunakan kaca mata hitam.
- 4) Parameter yang digunakan untuk melakukan deteksi kantuk diantaranya adalah aktivitas pergerakan mata, mulut, dan kepala.
- 5) Output terbagi menjadi 2 kategori, yaitu mengantuk (*sleepy*) dan terjaga (*awake*).

1.5. Manfaat Penelitian

Adapun manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut:

- 1) Mengetahui performa YOLO versi 8 dalam melakukan deteksi kantuk menggunakan parameter aktivitas gerakan mata, mulut, dan kepala.
- 2) Menjadi sumber rujukan untuk melakukan pengembangan terkait sistem deteksi kantuk maupun kelelahan pada pengemudi kendaraan bermotor dan penggunaan YOLO versi 8.
- 3) Hasil penelitian diharapkan dapat membantu dalam menurunkan tingkat kecelakaan bermotor akibat kelalaian manusia karena mengantuk saat mengemudi, khususnya pada pengemudi kendaraan roda empat.

1.6. Metodologi Penelitian

Adapun tahapan-tahapan penelitian yang akan dilakukan, yaitu sebagai berikut.

- 1) Studi Literatur

Pada tahap ini, penulis melakukan pencarian dan pengumpulan informasi dari pakar maupun publikasi penelitian terdahulu mengenai tanda-tanda awal gejala kantuk, sistem deteksi secara *real-time*, dan penggunaan YOLOv8.

- 2) Analisis Permasalahan

Setelah memperoleh informasi yang telah dikumpulkan sebelumnya, penulis melakukan analisis yang bertujuan untuk memahami implementasi YOLOv8 dalam sistem deteksi kantuk pada pengemudi kendaraan roda empat secara *real-time*.

3) Perancangan Sistem

Pada tahap selanjutnya, dilakukan perancangan sistem mulai dari perancangan arsitektur umum, pengumpulan data, pelatihan model, hingga implementasi ke aplikasi mobile.

4) Implementasi Sistem

Perancangan sistem yang telah dibuat sebelumnya akan diimplementasikan agar menciptakan sebuah sistem yang selaras dengan tujuan penelitian.

5) Pengujian Sistem

Tahap ini akan melakukan pengujian terhadap sistem yang telah diimplementasikan sebelumnya untuk memastikan bahwa sistem yang dirancang dapat digunakan dengan baik untuk memprediksi jenis kepribadian.

6) Penyusunan Laporan

Pada tahap ini peneliti akan melakukan penyusunan laporan dan dokumentasi proses penelitian sebagai bentuk representasi dari hasil penelitian yang telah dilakukan.

1.7. Sistematika Penulisan

Sistem penulisan skripsi ini terdiri dari beberapa bagian yaitu:

BAB 1: PENDAHULUAN

Pada bab ini berisikan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, serta metodologi dan sistematika penulisan.

BAB 2: LANDASAN TEORI

Pada bab ini berisikan teori yang berhubungan dengan helm pengendara, plat nomor polisi, pelanggaran lalu lintas, *computer vision*, *object detection*, Convolutional Neural Network, You Only Look Once sebagai metode yang diimplementasikan dalam penelitian.

BAB 3: ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini berisikan penjelasan mengenai arsitektur umum yang digunakan pada penelitian ini dimulai dari tahap *data acquisition*, *pre-processing*, *training*, *testing*, *output*, sampai perancangan antarmuka sistem.

BAB 4: IMPLEMENTASI DAN PENGUJIAN

Pada bab ini berisikan proses implementasi dan pengujian dari hasil rancangan sistem yang telah dijelaskan pada bab sebelumnya agar dapat memperlihatkan hasil penelitian ini dari pengujian apakah sudah mencapai tujuan.

BAB 5: KESIMPULAN DAN SARAN

Pada bab ini berisikan kesimpulan dari hasil penelitian yang telah dilakukan beserta saran yang dapat diterapkan pada penelitian kedepannya.

BAB 2

LANDASAN TEORI

2.1. *Microsleeping*

Microsleeping merupakan sebuah kondisi seseorang tertidur selama periode singkat, biasanya berlangsung hanya sepersekian detik hingga sampai 10 detik lamanya. Kondisi ini dapat terjadi secara tiba-tiba dan tanpa disadari, terutama ketika seseorang mencoba untuk tetap terjaga (Maudisha, 2023). Selama *microsleep*, otak dengan cepat bergantian antara tidur dan waspada. Fenomena ini dapat terjadi secara berulang, terutama saat melakukan aktivitas monoton atau saat mengalami kekurangan tidur (Summer & Rehman, 2023). *Microsleep* dapat menyebabkan risiko kecelakaan serius yang tinggi, terutama saat melakukan aktivitas yang membutuhkan perhatian, seperti mengemudi atau mengoperasikan alat berat.

2.2. *Real-Time Object Detection*

Deteksi objek adalah salah satu aspek dari visi komputer, yang bertujuan untuk mengidentifikasi dan mendeteksi objek dalam sebuah gambar berdasarkan warna, bentuk, dan data yang dikumpulkan (Lin et al., 2020).

Dalam pendeteksian objek secara *real-time*, kecepatan merupakan faktor penting yang harus diperhatikan. Mendeteksi objek pada gambar berbeda dengan mendeteksi objek pada video karena video dapat memproses lebih dari 24 frame per detik (FPS), dan jika proses pendeteksian objek terlalu lama, maka kualitas video yang dihasilkan akan buruk dan akan terjadi penundaan pada setiap frame, sehingga membuat video tampak patah-patah (Ding et al., 2019).

2.3. *Citra Grayscale*

Gambar skala abu-abu atau *grayscale* adalah gambar yang tidak memiliki warna, hanya skala abu-abu. Jenis gambar ini memiliki nilai bergradasi untuk tiap piksel dan mudah untuk diproses.



Gambar 2.1 Citra *Grayscale*

Pada Gambar 2.3 ditampilkan citra dataset yang sudah mengalami tahap *grayscale* dimana tingkatan dalam citra grayscale dapat diwakili oleh 256 tingkat keabuan yang disimpan dalam format bilangan bulat 8-bit. Tingkat keabuan ini berkisar dari 0 (hitam) hingga 255 (putih), dengan nilai 128 mewakili abu-abu netral.

Beberapa persamaan untuk mengonversi gambar berwarna dengan nilai matriks untuk masing-masing komponen R, G, dan B ke dalam gambar dengan nilai Y tunggal (saluran pencahayaan). Persamaan pertama adalah menemukan nilai rata-rata masing-masing komponen warna R, G, dan B.

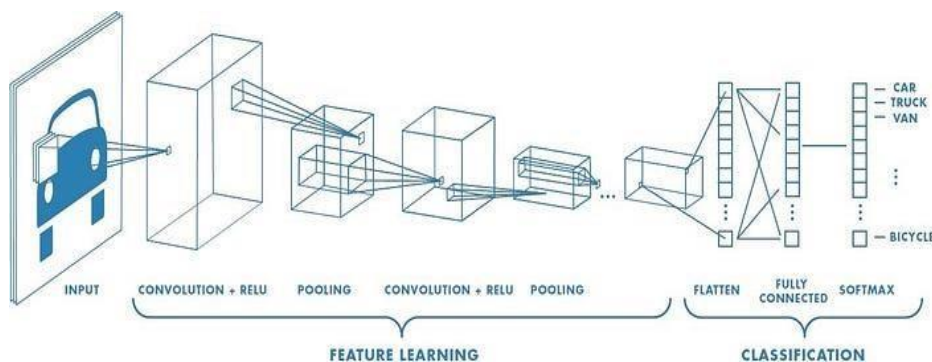
$$Y = \frac{(R + G + B)}{3} \dots\dots\dots (2.1)$$

Persamaan lainnya ialah dengan menggunakan nilai skala tertentu dari masing-masing komponen warna.

$$Y = 0.299R + 0.587G + 0.144B \dots\dots\dots (2.2)$$

2.4. *Convolutional Neural Network* (CNN)

CNN atau *Convolutional Neural Network* merupakan salah satu kelas *neural network* pada *deep learning* dan merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang secara khusus dirancang untuk pengolahan citra dengan berspesialisasi dalam memproses data yang memiliki topologi seperti *grid*, oleh karenanya CNN mampu melakukan pengenalan dan klasifikasi pada citra gambar.



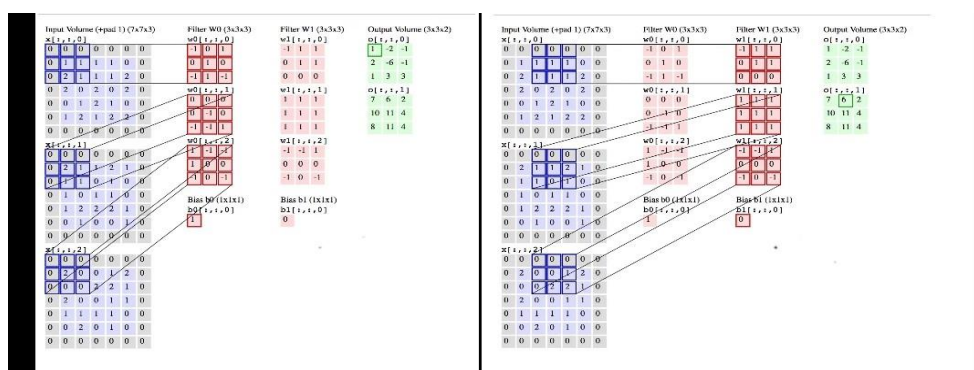
Gambar 2.2 Arsitektur Convolutional Neural Network (CNN)

(Sumber: towardsdatascience.com)

Seperti yang ditunjukkan pada Gambar 2.2, arsitektur CNN terdiri dari dua bagian utama, yaitu lapisan ekstraksi fitur dan lapisan yang terhubung sepenuhnya. Pada bagian pertama, yaitu lapisan ekstraksi fitur, gambar diubah menjadi peta fitur yang terdiri dari beberapa angka yang mewakili fitur-fitur gambar. Bagian ini terdiri dari *convolution layer* dan *pooling layer*. Pada bagian *Fully Connected Layer* diproses hasil pemetaan objek dari *feature extraction layer* untuk dapat diklasifikasikan.

1) Convolution Layer

Merupakan bagian utama serta lapisan utama pada metode CNN yang didalamnya ciri pada data citra akan diekstrak. Konvolusi memeriksa karakteristik gambar dengan menggunakan operasi matematika antara matriks gambar dan filter atau inti untuk mempertahankan hubungan antara piksel. Kernel adalah sebuah matriks, biasanya 3*3 atau 5*5, dengan nilai antara -1 dan 1. *Gambar 2.3* menunjukkan contoh proses konvolusi dengan dua filter W0 dan W1.



Gambar 2.3 Proses Konvolusi dengan Dua Filter

(Sumber: medium.com)

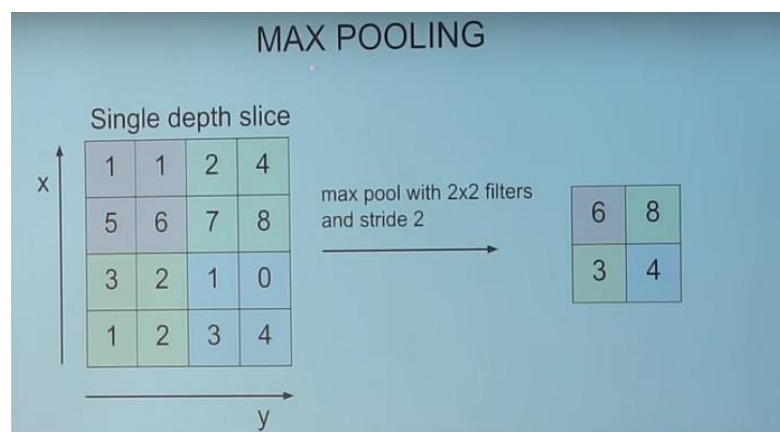
Pada Gambar 2.3 , setiap posisi dalam gambar menghasilkan angka yang merupakan hasil perkalian titik-titik antara bagian gambar dan filter yang digunakan. Memindahkan (mengecilkan) filter pada tiap posisi filter yang memungkinkan dalam gambar, akan menghasilkan gambar permulaan.

Pada gambar di atas, ukuran spasial hasil pemrosesan dapat dihitung dengan menggunakan rumus $(N - F + 2P)/S + 1$, di mana N adalah ukuran spasial gambar input (tinggi $H1$ = lebar $W1$), F adalah ukuran spasial filter, P adalah berapa kali gambar diubah ukurannya (biasanya 0), dan S adalah jumlah pergerakan filter untuk setiap perhitungan.

Contohnya, seperti ditunjukkan di atas, jika gambar input adalah $7 \times 7 \times 3$, filter ($W0$) adalah $3 \times 3 \times 3$, $S = 2$, dan $P = 1$, maka hasil akhir (gambar permulaan) adalah $(7 - 3 + 2) / 2 + 1 = 2$, dan proses tersebut diulangi dengan filter yang berbeda hingga diperoleh “gambar” baru, misalnya, sekumpulan gambar permulaan.

2) Pooling Layer

Fungsi zona *pooling* ini adalah untuk mengurangi jumlah input atau parameter dalam ruang dengan membagi operasi. Biasanya metode *pooling* maksimum atau nilai maksimum dalam segmen digunakan, tetapi ada metode *pooling* lain seperti *pooling* rata-rata atau *pooling* norma L2.



Gambar 2.4 Metode Max Pooling

(Sumber: medium.com)

Seperti halnya dengan *convolutional layer*, jika data input dari Merge Layer memiliki dimensi $W1 \times H1 \times D1$, maka data output dari *pooling layer* tersebut adalah:

$$W2 = \frac{(W1 - F)}{S+1} = H2 \dots\dots\dots (2.3)$$

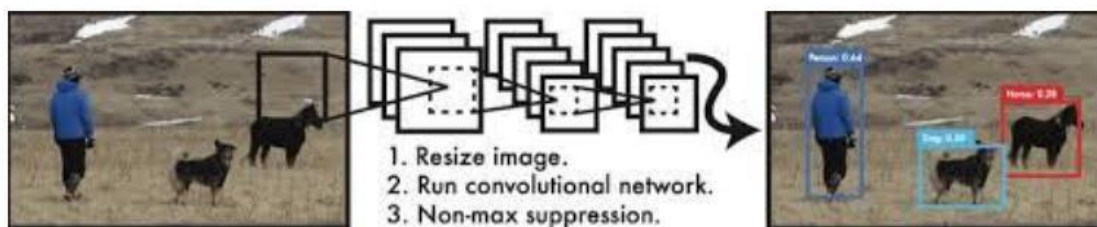
Seperti ditunjukkan pada Gambar 6, filter dalam *max pooling* hanya memilih nilai piksel tertinggi dalam bidang terima. Contohnya, pada gambar, dari empat piksel dengan nilai 1, 1, 5, dan 6 dalam bidang Receive, filter ini memilih piksel dengan nilai tertinggi, yaitu 6.

3) *Fully Connected Layer*

Berdasarkan susunan variasi multidimensi dari peta fitur yang diperoleh dari lapisan akuisisi fitur, bagian-bagian vektor dihasilkan yang dapat diatur ke dalam lapisan yang terhubung sepenuhnya. Lapisan yang terhubung sepenuhnya ini mirip dengan jaringan saraf. Dalam CNN, lapisan yang terhubung penuh adalah tahap di mana setiap neuron di lapisan sebelumnya terhubung ke setiap neuron di lapisan berikutnya. Setiap neuron di lapisan ini memiliki bobot dan biasnya sendiri. Kemudian, semua fitur yang dibentuk oleh jaringan saraf digabungkan untuk membentuk sebuah model.

2.5. You Only Look Once (YOLO)

You Only Look Once (YOLO) adalah algoritma yang dirancang untuk deteksi objek secara real-time. YOLO sangat cepat dan dapat memproses gambar dengan kecepatan 45 frame per detik. YOLO mensegmentasi gambar input ke dalam kisi-kisi $S \times S$, di mana setiap sel kisi-kisi memprediksi batas dan menghasilkan nilai untuk setiap kategori. Setiap wilayah terdiri dari lima nilai yang diprediksi yaitu: x-pusat, y-pusat, lebar sel (w), tinggi sel (h), dan keyakinan (Redmon, 2016).



Gambar 2.5 Sistem Deteksi YOLO

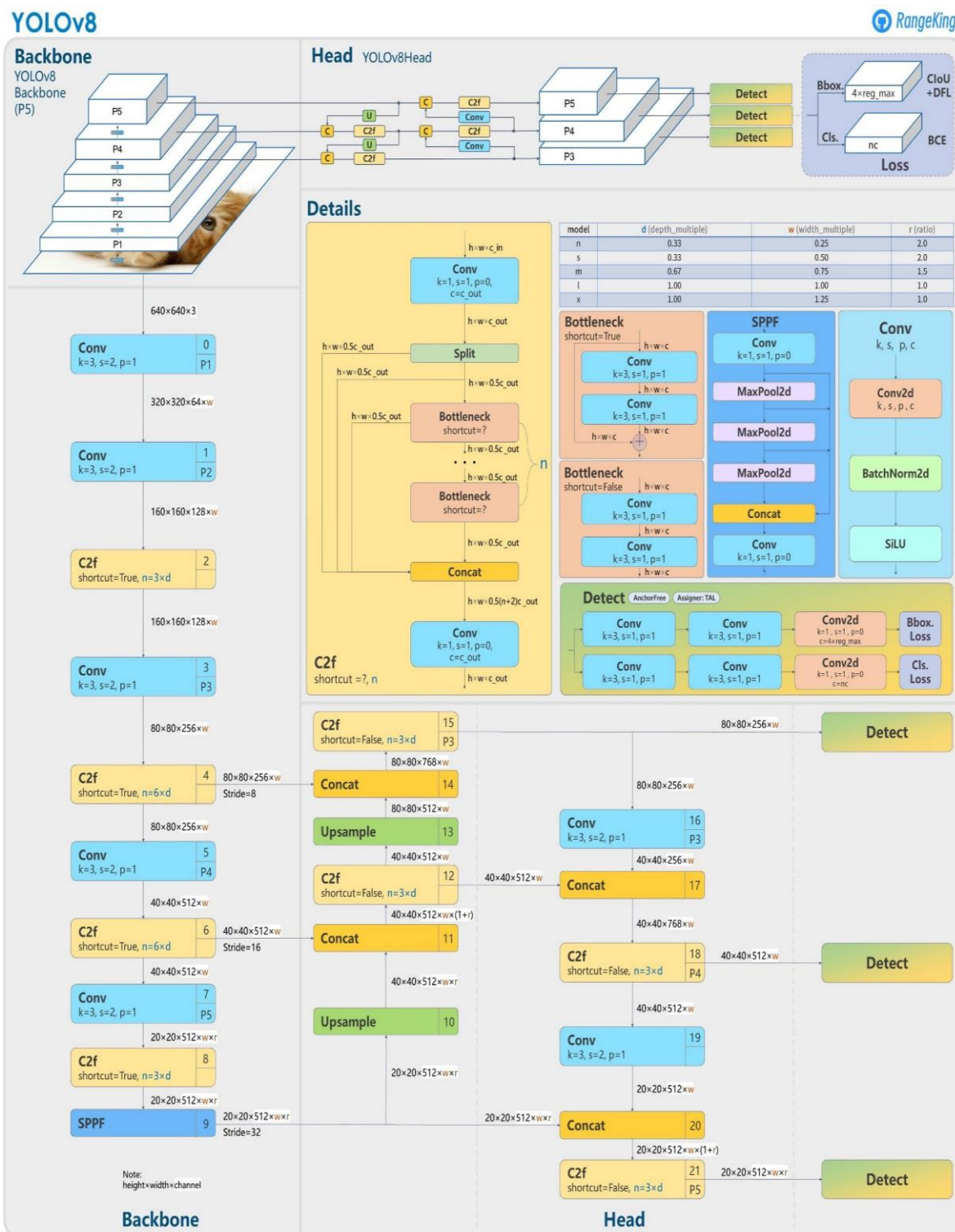
(Sumber: Fajri et al., 2020)

Pada Gambar 2.5, terdapat beberapa tahap dalam sistem deteksi YOLO. Hal pertama yang dilakukan oleh sistem adalah tahap *Resize* pada gambar yang tujuannya untuk mengubah ukuran gambar menjadi ukuran yang lebih mudah untuk diolah oleh sistem. Selanjutnya tahap *Run Convolutional Network*, proses ini digunakan untuk *task* pemrosesan gambar dimana pada gambar dilakukan ekstraksi fitur pada gambar dan pembelajaran model. Lalu tahap selanjutnya adalah *Non-max Supression* dimana biasanya teknik ini digunakan pada algoritma deteksi objek untuk mengurangi *bounding box* yang berlebihan dan memilih yang paling baik untuk objek yang sedang dideteksi pada gambar.

2.6. YOLOv8

YOLOv8 adalah merupakan algoritma yang dikembangkan berdasarkan arsitektur yang ada pada YOLOv5. YOLOv8 masih menggunakan arsitektur dasar yang sama dengan YOLOv5, yakni *CSPDarknet53*. *CSPDarknet53* merupakan varian dari model *Darknet53* yang dimodifikasi untuk meningkatkan akurasi dan efisiensi. Modifikasi ini dilakukan dengan menerapkan metode CSPNet (*Cross Stage Partial Network*).

Adapun arsitektur dari YOLOv8 dapat dilihat pada Gambar 2.6.



Gambar 2.6 Arsitektur YOLOv8

(Sumber: github.com)

Pada Gambar 2.6, arsitektur YOLOv8 terdiri dari jaringan tulang punggung (*backbone network*) dan kepala (*head*). Jaringan tulang punggung menggunakan Feature Pyramid Network (FPN) untuk melakukan ekstraksi fitur dari gambar input dan kepala

mengambil fitur yang disempurnakan dan memprediksi kotak pembatas, skor kelas objek, dan akurasi untuk setiap objek dalam citra.

2.7. Confusion Matrix

Confusion matrix merupakan sebuah metode yang digunakan untuk mengukur *performance* dalam permasalahan terkait klasifikasi yang menghasilkan *output* berupa dua kelas ataupun lebih. Perbedaan dari nilai prediksi dan nilai aktual pada *Confusion matrix* menghasilkan empat kombinasi. Adapun tabel dari *confusion matrix* terdapat pada Tabel 2.1.

Tabel 2.1 Confusion Matrix

| | | Prediksi | |
|--------|-----------------|-----------------|-----------------|
| | | <i>Positive</i> | <i>Negative</i> |
| Aktual | <i>Positive</i> | TP | FN |
| | <i>Negative</i> | FP | TN |

Berdasarkan tabel *confusion matrix* tersebut, hasil dari proses klasifikasi digambarkan menjadi empat istilah, yaitu:

- TP (*True Positive*) adalah data yang bernilai positif dan menghasilkan prediksi benar.
- FN (*False Negative*) adalah data yang bernilai positif, namun menghasilkan prediksi salah.
- FP (*False Positive*) adalah data yang bernilai negatif, namun menghasilkan prediksi benar.
- TN (*True Negative*) adalah data yang bernilai negatif dan menghasilkan prediksi salah.

Confusion matrix dapat digunakan untuk menghitung berbagai *performance matrix* seperti berikut ini:

2.7.1. Accuracy

Accuracy merupakan nilai yang didapat dari jumlah data yang bernilai positif dengan prediksi yang bernilai positif juga dan data yang bernilai negatif dengan prediksi yang bernilai negatif juga. Setelah itu, nilai tersebut dibagi dengan jumlah dari seluruh data

pada penelitian. Semakin banyak nilai data dengan diprediksi benar maka semakin tinggi nilai akurasi dari suatu penelitian.

$$Accuracy = \frac{TP + TN}{Total\ Data} \dots\dots\dots (2.4)$$

2.7.2. Precision

Precision merupakan rasio antara jumlah prediksi positif yang benar dengan total prediksi positif yang dilakukan. Dalam kata lain, *precision* mengukur sejauh mana hasil prediksi positif yang diperoleh konsisten dan akurat.

$$Precision = \frac{TP}{TP + FP} \dots\dots\dots (2.5)$$

2.7.3. Recall

Recall dikenal sebagai sensitivitas atau tingkat positif benar, mengukur seberapa baik model dapat mengidentifikasi dan mendapatkan informasi yang benar dari seluruh data yang sebenarnya bernilai positif. Ini merupakan persentase keberhasilan model dalam mendeteksi dan mengambil data positif yang relevan. *Recall* adalah ukuran kemampuan model dalam menghindari pengabaian terhadap kasus positif dan memperoleh pemahaman yang komprehensif terhadap kelas positif.

$$Recall = \frac{TP}{TP + FN} \dots\dots\dots (2.6)$$

2.7.4. F1-Score

F1-score merupakan metrik yang menggabungkan nilai *recall* dan *precision* untuk memberikan perbandingan atau rata-rata. *F1-score* memberikan informasi tentang sejauh mana model mampu mencapai keseimbangan antara mengenali data positif dengan benar (*recall*) dan memberikan prediksi yang benar positif (*precision*). Dengan demikian, *F1-score* memberikan gambaran tentang kemampuan model dalam mencapai akurasi yang seimbang antara menemukan data positif yang relevan dan memberikan prediksi yang tepat.

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \dots\dots\dots (2.7)$$

2.8. Penelitian Terdahulu

Beberapa penelitian mengenai deteksi kantuk pada pengemudi sudah pernah dilakukan, di antaranya adalah penelitian oleh (Uma Maheswari et al., 2022) yang melakukan prediksi kelelahan pada pengemudi kendaraan berdasarkan beberapa aspek menggunakan teknik *image processing* dengan mempertimbangkan kalkulasi EAR (*Eye Aspect Ratio*) dan MAR (*Mouth Aspect Ratio*) dengan menggunakan dataset NTHUDD, YawDD, dan EMOCDS (*Eye and Mouth Open Close Data Set*). Penelitian Mempertimbangkan EAR dan MAR calculations dan datasets NTHU-DD, YawDD, dan EMOCDS (*Eye and Mouth Open Close Data Set*). Penelitian berhasil mencapai kesimpulan bahwa metode yang diajukan berhasil mengungguli akurasi metode lain seperti KNN dan FaceNeT + SVM dengan mencapai tingkat akurasi sebesar 94.78% untuk penggunaan dataset YawDD, 79,98% untuk dataset NTHU-DD dan 95.67% untuk penggunaan dataset EMOCDS.

Penelitian lain dilakukan oleh (Amzar et al., 2022) yang menunjukkan bahwa suatu sistem *monitoring* pada pengemudi dapat dikembangkan dengan menggabungkan metode Support Vector Machine (SVM), Hidden Markov Model (HMM) dan CNN. Dengan memanfaatkan aktifitas kedipan mata, analisis, frekuensi menguap, estimasi pergerakan daerah kepala, dan pelacakan pada mata pada dataset custom dengan menggunakan perangkat Raspberry Pi, penelitian ini berhasil mencapai tingkat akurasi 80.9% untuk deteksi kedipan mata, 96.3% untuk deteksi mata tertutup, dan 69.6% untuk deteksi menguap.

Penelitian lainnya dalam deteksi kelelahan pada pengemudi kendaraan bermotor dilakukan oleh (Mu et al., 2022) terhadap 14.051 citra gambar wajah dengan multipostur dan citra gambar wajah berskala abu-abu dalam database FERET *face*. Menggunakan metode *Hough Transform*, *Adaboost*, dan PERCLOS dalam pengolahannya penelitian ini menghasilkan tingkat akurasi rata-rata sebesar 85,19% dalam mendeteksi potensi kelelahan pada pengemudi.

Penelitian selanjutnya adalah penelitian yang dilakukan oleh (Indra et al., 2023) yang menunjukkan bahwa metode YOLO (*You Only Look Once*) dapat digunakan dalam mengembangkan sistem deteksi kantuk pada pengemudi secara *real-time*. Penelitian ini menunjukkan bahwa YOLO versi 5 mampu menghasilkan tingkat akurasi yang tinggi dalam deteksi kantuk pada pengemudi. Dengan menggunakan 247 sampel data,

diantaranya data testing yang diambil dengan kamera *webcam* 1080p, penelitian ini berhasil mencapai akurasi rata-rata sebesar 84%.

Berikut adalah data penelitian terdahulu yang serupa:

Tabel 2.2 Penelitian Terdahulu

| No. | Penulis | Judul | Tahun | Keterangan |
|-----|---------------------------|---|-------|---|
| 1. | V Uma Maheswari et al. | <i>Driver Drowsiness Prediction Based on Multiple Aspects Using Image Processing Techniques</i> | 2022 | Mempertimbangkan EAR dan MAR <i>calculations</i> dan <i>datasets</i> NTHUDD, YawDD, dan EMOCDS (Eye and Mouth Open Close Data Set). Penelitian berhasil mencapai kesimpulan bahwa metode yang diajukan berhasil mengungguli akurasi metode lain seperti KNN dan FaceNeT + SVM dengan mencapai tingkat akurasi sebesar 94.78% untuk penggunaan <i>dataset</i> YawDD, 79,98% untuk <i>dataset</i> NTHUDD dan 95.67% untuk penggunaan <i>dataset</i> EMOCDS. |
| 2. | Raz Amzar et al. | <i>Driver Drowsiness Detection and Monitoring System (DDDMS)</i> | 2022 | Penelitian dilakukan untuk mengembangkan sistem <i>monitoring</i> pada pengemudi dengan memanfaatkan aktifitas kedipan mata, analisis, frekuensi menguap, estimasi pergerakan daerah wajah, dan pelacakan pada mata. Penelitian ini menggunakan metode SVM, HMM dan CNN dan berhasil mencapai tingkat akurasi 80.9% untuk deteksi kedipan mata, 96.3% untuk deteksi mata tertutup, dan 69.6% untuk deteksi menguap. |

Tabel 2.2 Penelitian Terdahulu

| | | | | |
|----|---------------------------|---|------|--|
| 3. | Zhendong Mu et al. | <i>Research on a Driver Fatigue Detection Model Based on Image Processing</i> | 2022 | <p>Penelitian ini menggunakan metode <i>Hough Transform</i>, <i>Adaboost</i>, dan PERCLOS dalam mengolah FERET <i>face database</i> yang berisikan 14.051 gambar wajah dengan multi-postur dan gambar wajah berskala abu-abu dan mencapai tingkat akurasi rata-rata sebesar 85,19% dalam mendeteksi potensi kelelahan pada pengemudi.</p> |
| 4. | Dw Ayu Agung Indra et al. | Implementasi Algoritma YoloV5 untuk Kantuk Detektor pada Pengemudi Kendaraan Bermotor secara Realtime | 2023 | <p>Penelitian ini menggunakan metode YOLO versi 5 pada 247 sampel data. Dataset dibagi menjadi data training yang diambil dari <i>roboflow</i> dan data testing yang diambil menggunakan kamera <i>webcam</i> 1080p. hasil penelitian mencapai akurasi rata-rata sebesar 84% dengan nilai precision 85% untuk kelas positif dan 84% untuk kelas negatif, serta nilai recall sebesar 84% untuk kelas positif dan 86% untuk kelas negatif.</p> |
| 5. | Siti Maslikah et al. | Sistem Deteksi Kantuk Pada Pengendara Roda Empat Menggunakan <i>Eye Blink Detection</i> | 2020 | <p>Peneliti menggunakan metode <i>Haar Cascade Classifier</i> dan menggunakan Pi Camera yang tersambung dengan Raspberry Pi dan mampu mencapai tingkat akurasi sebesar 90% untuk kondisi di siang hari dan 85% di malam hari dengan jarak optimal antara kamera dan pengendara sebesar 30-50 cm dan tingkat kemiringan 0-45 derajat untuk deteksi yang akurat.</p> |

Tabel 2.2 Penelitian Terdahulu

| | | | | |
|----|-------------------------|---|------|---|
| 6. | Furkat Safarov et al. | <i>Real-Time Deep Learning-Based Drowsiness Detection : Leveraging Computer-Vision and Eye-Blink Analyses for Enhanced Road Safty</i> | 2023 | Peneliti menggunakan <i>framework MediaPipe</i> untuk mengekstraksi landmark pada bagian wajah dan tatapan mata menggunakan persamaan EAR dan mampu mencapai tingkat akurasi sebesar 96% untuk kasus dimana pengemudi menyangarkan kepala, dan 85% untuk kelas menguap dimana mata pengemudi tertutup dan mulut terbuka. |
| 7. | Anjeana N & K. Anusudha | <i>Real time face recognition system based on YOLO and InsightFace</i> | 2023 | <p>Penelitian ini menggunakan <i>custom</i> dataset yang berisikan 12 kelas dengan setiap kelasnya terdapat 300 gambar wajah dengan berbagai pola.</p> <p>Metode yang digunakan adalah YOLO V7 dan InsightFace Model. Hasil yang dicapai pada penelitian ini membuktikan bahwa model algoritma yang diajukan dapat mengungguli kecepatan dan akurasi metode algoritma <i>face recognition</i> yang sudah ada. Metode ini juga dapat mengenali wajah pada berbagai kondisi efek cahaya dan wajah pada saat terhalangi suatu objek.</p> |

2.9. Perbedaan Penelitian

Penelitian ini memiliki perbedaan dengan penelitian yang telah dilakukan sebelumnya, dimana pada penelitian (Uma Maheswari et al., 2022) dan (Safarov et al., 2023) metode yang diajukan adalah metode yang mempertimbangkan kalkulasi EAR (*Eye Aspect Ratio*) dan MAR (*Mouth Aspect Ratio*) pada citra wajah pengemudi, dan pada penelitian (Mu et al., 2022) menggunakan metode *Hough Transform*, *Adaboost*, dan *PERCLOS*

untuk melakukan deteksi. Sedangkan pada penelitian ini saya menggunakan YOLO versi 8 untuk melakukan deteksi secara *real-time*.

Terdapat juga perbedaan pada perangkat implementasi sistem dan pengambilan citra data testing pada penelitian. Penelitian yang dilakukan oleh (Amzar et al., 2022) dan (Maslikah et al., 2020) menggunakan perangkat *Raspberry Pi* yang tersambung dengan *Pi Camera* dan penelitian (Indra et al., 2023) yang menggunakan 1080p *webcam camera*. Sedangkan pada penelitian ini saya menggunakan perangkat *smartphone* dengan pengambilan citra data testing secara *real-time* pada interval waktu 0,5 detik.

Perbedaan selanjutnya adalah pada parameter yang penulis gunakan pada penelitian ini meliputi aktivitas gerakan pada mata, mulut, dan kepala pengemudi. Berbeda dengan penelitian sebelumnya yang dilakukan oleh (Uma Maheswari et al., 2022) dan (Amzar et al., 2022) yang hanya menggunakan parameter aktivitas gerakan pada area mata dan mulut saja.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Pada bab 3 ini menjelaskan tentang analisis dan perancangan sistem untuk melakukan deteksi kantuk pada pengemudi kendaraan roda empat. Bab ini akan membahas mengenai sumber data yang digunakan, proses input, pra-pemrosesan data, pelatihan data, dan pengujian, serta hasil yang didapatkan.

3.1. Dataset

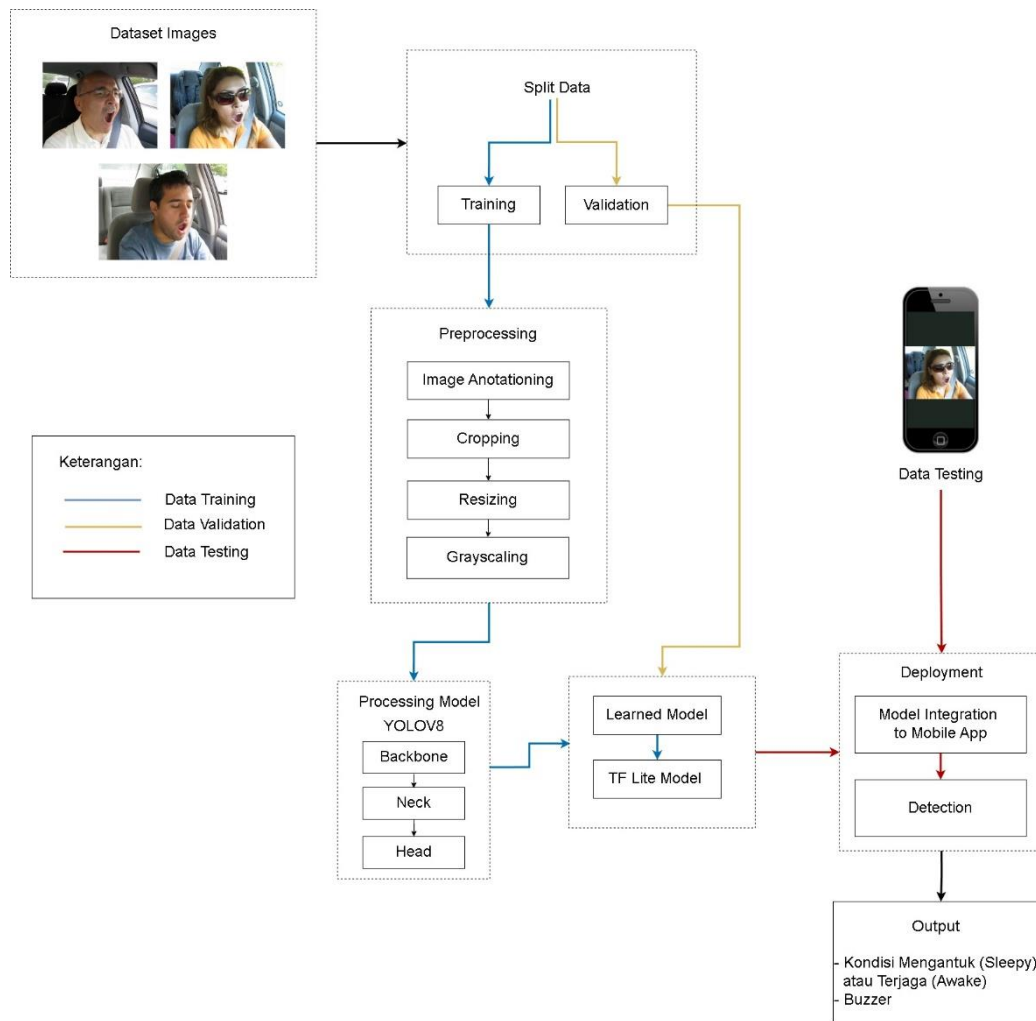
Data yang digunakan pada penelitian ini diambil melalui website *roboflow* dan *kaggle* dengan jumlah 2124 data. Pada Gambar 3.1, pada gambar sebelah kiri terdapat pengemudi yang terindikasi mengantuk atau tertidur dengan tampilan aktivitas pergerakan kepala yang miring ke belakang, mata tertutup dan mulut tertutup. Sedangkan pada gambar sebelah kanan merupakan citra pengemudi yang terjaga atau sadar dengan tampilan aktivitas pergerakan kepala yang tegak, mata terbuka, dan mulut tertutup.



Gambar 3.1 Visualisasi Citra Pengemudi

3.2. Analisis Sistem

Pada penelitian ini dilakukan beberapa tahapan, Gambar 3.2 menunjukkan arsitektur umum dari penelitian ini. Data pengemudi akan dikumpulkan dalam bentuk gambar dengan format JPG yang kemudian akan dibagi menjadi data *train* dan data *validation*.



Gambar 3.2 Arsitektur Umum

Penjelasan terkait arsitektur umum pada Gambar 3.2 adalah sebagai berikut:

3.3. Data Acquisition

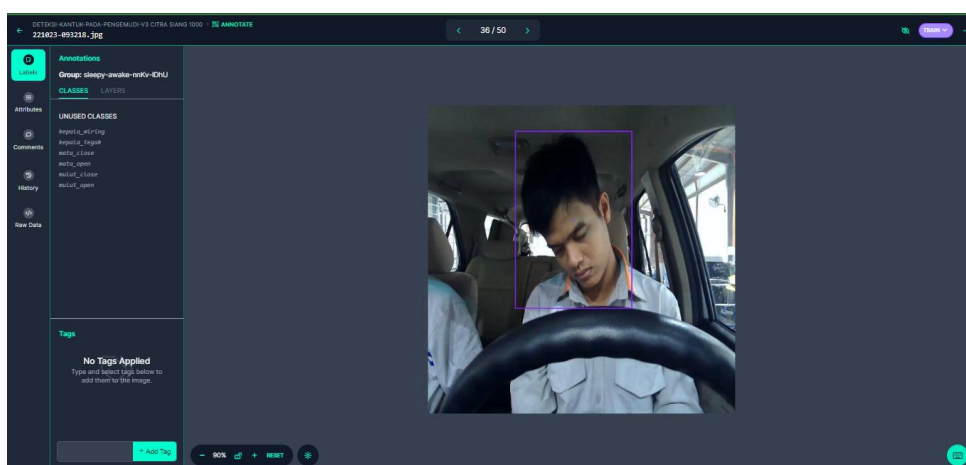
Tahap awal yang dilakukan adalah mengumpulkan data input yang akan digunakan sesuai dengan kebutuhan peneliti. Penelitian ini menggunakan dataset yang diambil dari *roboflow* dan *kaggle*. Kemudian data yang telah dikumpulkan dibagi menjadi dua bagian yaitu data *training* dan data *validation*.

3.4. Preprocessing Data

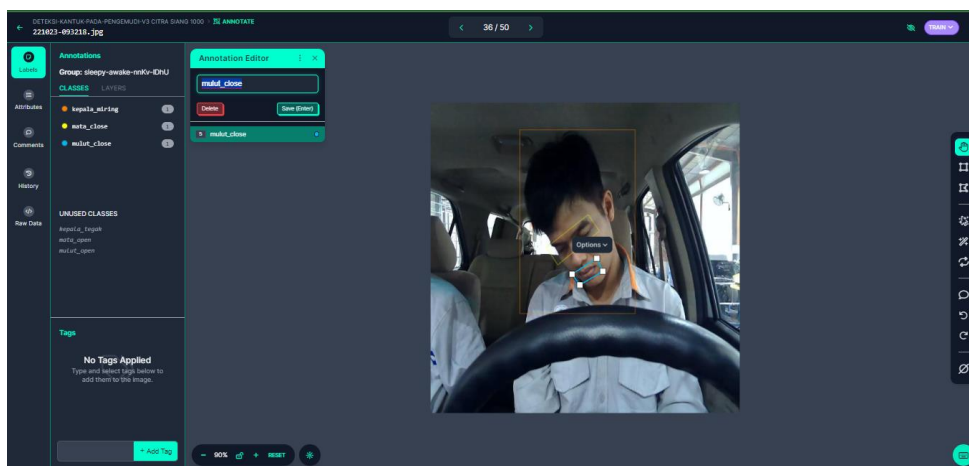
Pada tahap ini, data citra akan melalui beberapa proses terlebih dahulu agar dapat memperoleh hasil yang lebih baik pada tahap selanjutnya. Tahapan *preprocessing* yang dilakukan, yaitu *resizing*, *cropping*, *image anotating*, dan *grayscale*.

3.4.1. Image Anotationing

Image Anotationing merupakan proses menambahkan informasi tambahan pada citra data berupa teks, lingkaran, kotak, tanda panah, ataupun bentuk lainnya untuk mengidentifikasi fitur tertentu pada sebuah gambar. Pada penelitian ini, *image anotationing* digunakan untuk menargetkan lokasi aktivitas pergerakan mata, mulut, dan kepala pada data citra. Proses pembuatan *bounding box* pada roboflow dapat dilihat pada Gambar 3.3. Setelahnya penulis masuk ke tahap pemilihan class agar nantinya model akan mempelajari dan dapat membedakan setiap citra pengemudi berdasarkan indikasi aktivitas pergerakan kepala, mata, dan mulut seperti yang terdapat dalam Gambar 3.4.



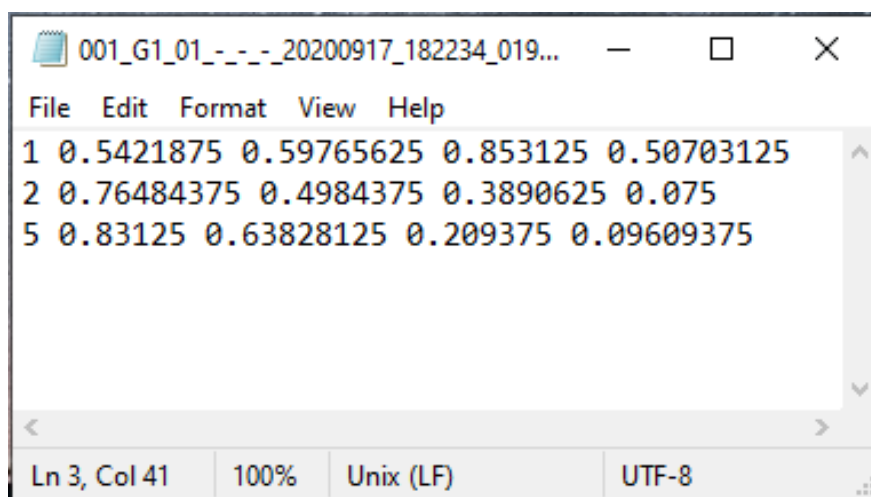
Gambar 3.3 Membuat *Bounding Box*



Gambar 3.4 Memilih *Class*

Hasil dari tahap *image anotationing* adalah label pada citra yang disimpan dalam bentuk file.txt. Pada Gambar 3.5 merupakan file.txt *labeling* citra yang berisikan format anotasi. Format anotasi data dimana nilai pertama (1, 2, dan 5) mewakili label kelas, dan nilai-nilai berikutnya merupakan fitur atau atribut dari data tersebut.

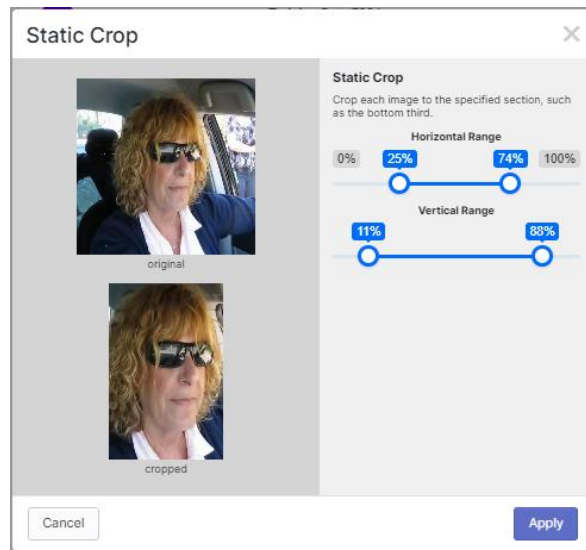
```
<class_id> <x_center> <y_center> <width> <height>
```



Gambar 3.5 Isi File.txt

3.4.2. Cropping

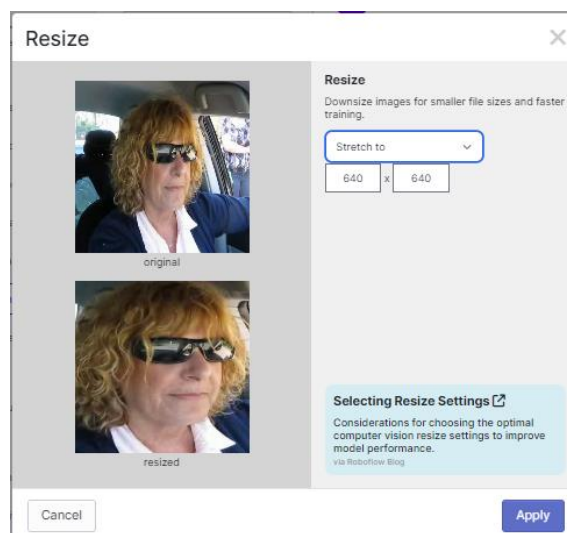
Cropping merupakan proses memotong bagian tertentu pada suatu citra. Pada penelitian ini *cropping* dilakukan untuk memfokuskan citra dataset hanya pada area wajah dan kepala pengemudi seperti yang terdapat pada Gambar 3.6. Pada Gambar 3.6 *cropping* dilakukan dengan *range* 25% - 74% secara horizontal dan *range* 11% - 88% secara vertikal. Adapun beberapa tujuan *cropping* dalam tahap *preprocessing* citra dataset selain memfokuskan citra pada area yang penting, *cropping* juga dapat meningkatkan kualitas citra data dengan menghilangkan *noise* yang berada di luar area objek yang diteliti dan membuat citra pada dataset menjadi ukuran yang seragam.



Gambar 3.6 Proses *Cropping*

3.4.3. Resizing

Resizing merupakan proses mengubah ukuran atau skala citra menjadi dimensi yang lebih kecil. Tahap ini bertujuan untuk meningkatkan efisiensi pada proses ekstraksi fitur pada citra dan proses pembelajaran pada model. Seperti pada Gambar 3.7, penelitian ini citra dataset akan di *resize* menjadi ukuran 640x640 piksel dengan beberapa alasan diantaranya keseimbangan antara akurasi dan kecepatan, kompatibilitas dengan model pra-latih serta kemampuan untuk memproses gambar besar.



Gambar 3.7 Proses *Resizing*

3.4.4. Grayscale

Proses selanjutnya adalah merubah citra hasil anotasi gambar dari dimensi warna lain menjadi nuansa warna monokromatik. Proses ini akan menghasilkan citra dengan warna diantara gradasi hitam dan putih seperti pada Gambar 3.8. Tujuan dilakukan *grayscale* adalah untuk menyetarakan warna pada citra pengemudi sehingga fitur dan *background* dapat dibedakan dengan mudah dan proses pendeteksian objek dapat dilakukan lebih cepat.

Perubahan citra berwarna menjadi citra *grayscale* diawali dengan mengekstraksi nilai matrik pada masing-masing RGB pada citra menggunakan OpenCV, setelahnya mengubah nilai matrik masing-masing komponen RGB menjadi citra *grayscale* dengan satu nilai kanal. Perhitungan dapat menggunakan persamaan 2.1 seperti berikut:

$$\begin{bmatrix} [43 & 42 & 38] & [15 & 6 & 2] & [50 & 40 & 33] \\ [62 & 64 & 58] & [53 & 72 & 107] & [91 & 76 & 58] \\ [51 & 46 & 45] & [82 & 105 & 137] & [48 & 32 & 25] \end{bmatrix}$$

$$P1 = \frac{(R + G + B)}{3} = \frac{(43 + 42 + 38)}{3} = 41$$

$$P2 = \frac{(R + G + B)}{3} = \frac{(15 + 6 + 2)}{3} = 7$$

$$P3 = \frac{(R + G + B)}{3} = \frac{(50 + 40 + 33)}{3} = 41$$

$$P4 = \frac{(R + G + B)}{3} = \frac{(62 + 64 + 58)}{3} = 61$$

$$P5 = \frac{(R + G + B)}{3} = \frac{(53 + 72 + 107)}{3} = 77$$

$$P6 = \frac{(R + G + B)}{3} = \frac{(91 + 76 + 58)}{3} = 75$$

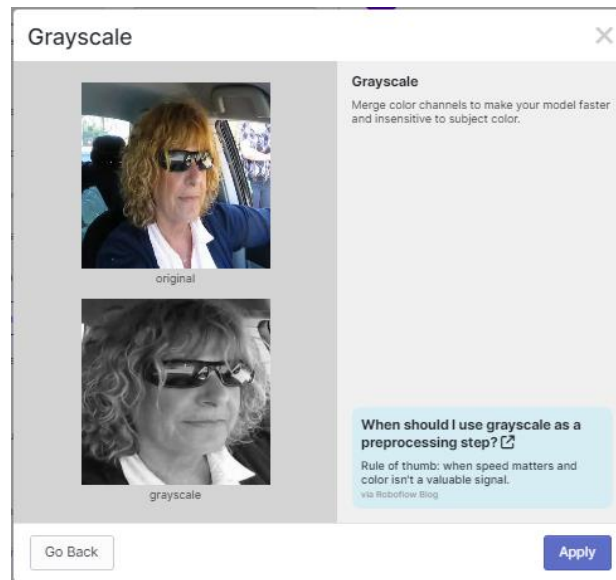
$$P7 = \frac{(R + G + B)}{3} = \frac{(51 + 46 + 45)}{3} = 47$$

$$P8 = \frac{(R + G + B)}{3} = \frac{(82 + 105 + 137)}{3} = 108$$

$$P6 = \frac{(R + G + B)}{3} = \frac{(48 + 32 + 25)}{3} = 35$$

Berdasarkan perhitungan tersebut, didapatkan nilai metrik baru citra *grayscale* dengan satu nilai kanal sebagai berikut:

$$\begin{bmatrix} 41 & 7 & 41 \\ 61 & 77 & 75 \\ 47 & 108 & 35 \end{bmatrix}$$



Gambar 3.8 Proses *Grayscale*

3.5. *Processing Model*

Tahap ini merupakan tahap pembelajaran model pada data *training* dataset citra yang sudah melalui tahap *Preprocessing* agar dapat mengenali tanda-tanda mengantuk pada seseorang berdasarkan aktivitas gerakan mata, mulut, dan kepala menggunakan algoritma YOLO versi 8.

3.5.1. *Package Installation*

Pada tahap ini, peneliti akan melakukan instalasi seluruh *requirement* yang diperlukan pelatihan model oleh YOLOv8. Terdapat dua cara dalam melakukan instalasi YOLOv8 yaitu dengan menggunakan pip sesuai rekomendasi dari *developer* dan secara langsung melalui sumber repositori git ultralytics.

3.5.2. *Data Configuration*

Pada tahap ini, peneliti akan melakukan *import* dataset yang sudah melalui tahap *preprocessing* dengan menggunakan API Roboflow. Kemudian pada folder dataset yang

sudah di-*import* terdapat citra data untuk *training* dan data untuk *validation* beserta labelnya serta file `data.yaml` yang merupakan file konfigurasi dataset yang digunakan. Adapun konfigurasi data pada file `data.yaml` dapat dilihat pada Gambar 3.9. *Train*, *Val*, dan *Test* menunjukkan *path* dari data image, *nc* menunjukkan jumlah *class* yang ada pada dataset, dan *names* menunjukkan nama *class* yang digunakan.



Gambar 3.9 File `data.yaml`

3.5.3. Pelatihan Model

Tahap ini citra pengemudi akan dikategorikan pada kelasnya menggunakan algoritma You Only Look Once versi 8. YOLOv8 merupakan salah satu algoritma yang mampu mendeteksi objek berdasarkan kemungkinan kelas yang telah ditetapkan secara *real-time*.

3.5.3.1. Backbone

Jaringan *backbone* pada arsitektur YOLOv8 berfungsi untuk melakukan ekstraksi fitur kepala, mata, dan mulut dari gambar input yang sudah berupa citra *grayscale*.

3.5.3.2. Neck

Jaringan *Neck* memiliki struktur PAN-FPN (*Path Aggregation Network with Feature Pyramid Network*) yang terinspirasi oleh PANet. Pada jaringan ini terdapat penambahan *layer* yang nantinya akan membentuk *pyramid feature* yang digunakan untuk menghasilkan prediksi berupa deteksi pada pixel yang kecil untuk meningkatkan proses segmentasi.

3.5.3.3. Head

Jaringan *Head* pada arsitektur YOLOv8 berfungsi untuk melakukan prediksi objek berdasarkan fitur-fitur yang diekstraksi oleh jaringan “*backbone*” dan prediksi oleh jaringan “*neck*” dengan menghasilkan *bounding box* pada objek yang dideteksi beserta skor kelas objek, dan akurasi pada objek di dalam citra.

3.6. Learned Model

Pada tahap ini, sistem akan mempelajari *learned model* atau hasil dari proses *training model* dengan YOLOv8. *Learned model* nantinya digunakan untuk proses deteksi pada citra baru dan mengelompokkannya ke dalam kategori yang telah ditentukan. Citra Data yang kemudian akan dikelompokkan menjadi dua kelompok: *awake* (terjaga) dan *sleepy* (mengantuk) berdasarkan dengan kombinasi *class* yang terdeteksi pada citra.

Adapun kombinasi *class* pada citra untuk kelompok *awake* (terjaga) adalah:

- kepala_tegak, mata_open, mulut_close
- kepala_tegak, mata_open, mulut_open
- kepala_miring, mata_open, mulut_close
- kepala_miring, mata_open, mulut_open
- mata_open, mulut_close
- mata_open, mulut_open
- kepala_tegak, mata_open
- kepala_miring, mata_open

Adapun kombinasi *class* pada citra untuk kelompok *sleepy* (mengantuk) adalah:

- kepala_tegak, mata_close, mulut_close
- kepala_tegak, mata_close, mulut_open
- kepala_miring, mata_close, mulut_close

- kepala_miring, mata_close, mulut_open
- mata_close, mulut_close
- mata_close, mulut_open
- kepala_tegak, mata_close
- kepala_miring, mata_close

3.7. TF Lite Model

Pada tahap ini *learned model* akan diubah formatnya menjadi *TensorFlow Lite* agar dapat digunakan pada perangkat mobile. Proses ini dapat dilakukan dengan memanfaatkan *TensorFlow Lite Converter*.

3.8. Deployment

3.8.1. Model Integration to Mobile App

Pada tahap ini model berformat *TensorFlow Lite* diintegrasikan ke dalam aplikasi *mobile*. Tahap ini meliputi beberapa proses seperti menambahkan *file* model *TensorFlow Lite* ke proyek *Android Studio* dengan menggunakan bahasa Kotlin untuk menggunakan model.

3.8.2. Detection

Hasil model dari proses pelatihan kemudian dikonversi dengan *TensorFlow Lite* untuk menghasilkan file dengan ekstensi *tflite*. Model kemudian diintegrasikan ke dalam aplikasi *mobile* dan dimodifikasi untuk dapat mengakses kamera perangkat. Setelahnya aplikasi dapat diuji menggunakan data *testing* yang didapatkan secara *real-time* untuk memastikan bahwa model sudah berjalan dengan baik pada perangkat aplikasi.

3.9. Output

Tahap terakhir dari rangkaian proses sistem deteksi yang akan menghasilkan *output* terdeteksi atau tidaknya tanda-tanda kantuk pada pengemudi. Pada sistem ditetapkan bahwa *bounding box* akan muncul pada setiap 0,5detik yang kemudian sistem akan menyimpan informasi deteksi kelas hasil kombinasi dari setiap kelas pada *bounding box* yang ditampilkan. Apabila sistem mendeteksi tanda-tanda kantuk pada pengemudi dengan terdeteksinya kelas mengantuk (*sleepy*) dari kombinasi kelas *bounding box* yang

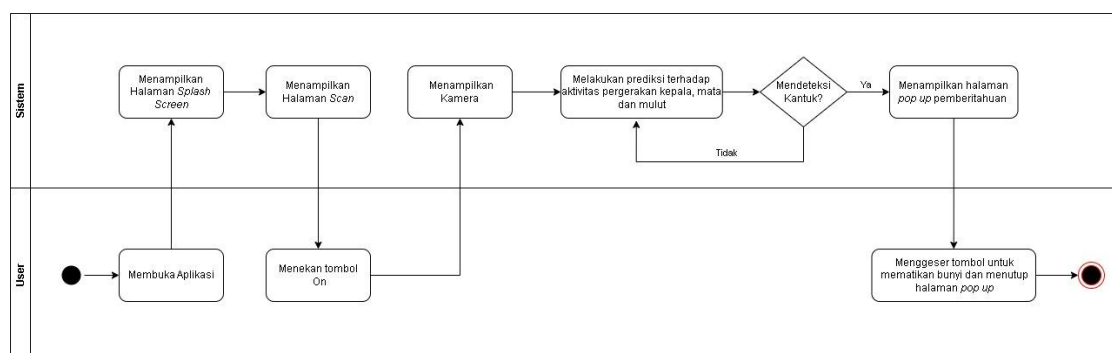
muncul dalam kurun waktu 2 detik atau 4 kali berturut-turut maka *buzzer* berupa bunyi serta tampilan pemberitahuan pada layar perangkat akan muncul sebagai peringatan awal dan upaya untuk pengemudi tetap terjaga.

3.10. System Interface

Tahap ini merupakan tahap perancangan tampilan antarmuka pengguna aplikasi. Adapun tujuan dari tahap ini adalah memperlihatkan dan memberikan penjelasan ilustrasi dari sistem aplikasi yang akan dibuat. Halaman yang dirancang yaitu halaman *splash screen*, halaman utama (deteksi), dan halaman *pop-up alert* pemberitahuan.

3.10.1. Activity Diagram

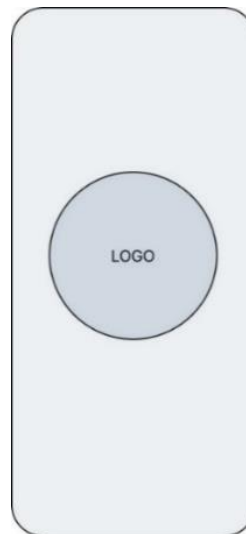
Diagram aktivitas digunakan untuk menunjukkan alur aktivitas atau tindakan yang terjadi dari aplikasi. Diagram ini menggambarkan alur kerja dari suatu kegiatan ke kegiatan yang lain melalui serangkaian langkah-langkah yang terjadi pada sistem aplikasi untuk memudahkan pemakaian oleh pengguna. Diagram aktivitas dapat dilihat pada Gambar 3.10.



Gambar 3.10 Activity Diagram Sistem Aplikasi

3.10.2. Rancangan Halaman *Splash Screen*

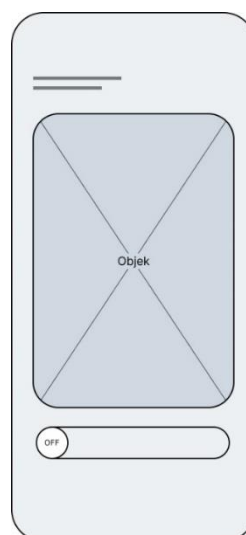
Halaman *splash screen* merupakan halaman yang akan tampil saat pengguna pertama kali membuka aplikasi. Pada halaman *splash screen* ini ditampilkan logo aplikasi. Perancangannya dapat dilihat pada Gambar 3.11.



Gambar 3.11 Rancangan Halaman *Splash Screen*

3.10.3. Rancangan Halaman *Scan*

Halaman *scan* atau halaman utama aplikasi dimana terdapat fitur deteksi yang berfungsi untuk mendeteksi tanda-tanda mengantuk pada pengemudi kendaraan roda empat. Untuk dapat menggunakan aplikasi, Pengguna harus menghidupkan sistem kamera pada aplikasi dengan menggeser tombol *off* pada halaman *scan* menjadi *on*. Kemudian aplikasi akan mengakses kamera pada ponsel dan mulai mendeteksi secara *real-time*. Rancangan halaman *scan* dapat dilihat pada Gambar 3.12.

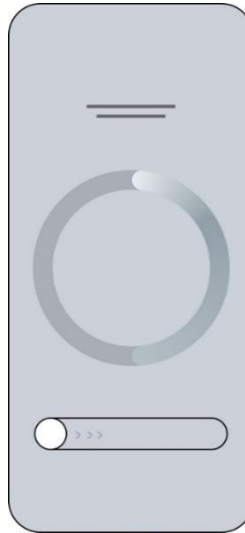


Gambar 3.12 Rancangan Halaman *Scan*

3.10.4. Rancangan Halaman *Pop Up* Pemberitahuan

Halaman pemberitahuan merupakan halaman yang muncul apabila sistem berhasil mendeteksi kantuk pada pengemudi. Aplikasi kemudian akan memunculkan bunyi

berupa *alarm* peringatan sebagai upaya untuk pengemudi tetap terjaga. Pada halaman ini terdapat tombol yang harus digeser oleh pengemudi untuk dapat mematikan *alarm* yang berbunyi, dengan ini pengemudi dapat kembali sadar dan terjaga untuk dapat mematikan *alarm*. Rancangan halaman pemberitahuan dapat dilihat pada Gambar 3.12.



Gambar 3.13 Rancangan Halaman *Pop Up* Pemberitahuan

BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini berisikan tentang penjelasan mengenai proses implementasi dan hasil uji coba algoritma YOLO (You Only Look Once) versi 8 dalam mendeteksi gejala kantuk pada pengemudi kendaraan roda empat sesuai dengan rancangan sistem pada bab sebelumnya.

4.1. Implementasi Sistem

Pada tahap ini akan dilakukan implementasi algoritma YOLO (You Only Look Once) versi 8 untuk deteksi kantuk pada pengemudi kendaraan roda empat.

4.1.1. Perangkat Keras dan Perangkat Lunak

Komponen perangkat keras yang digunakan penulis dalam merancang sistem tersebut adalah PC MSI MAG Infinite S3 13th dengan spesifikasi sebagai berikut:

- 1) CPU: Intel Core i7-13700F
- 2) RAM: 32GB
- 3) Storage: 1TB
- 4) GPU: Nvidia GeForce RTX 4070
- 5) OS: Windows 11 Home Single Language 64-bit

Komponen perangkat lunak dan *library* yang digunakan untuk membangun sistem yaitu sebagai berikut:

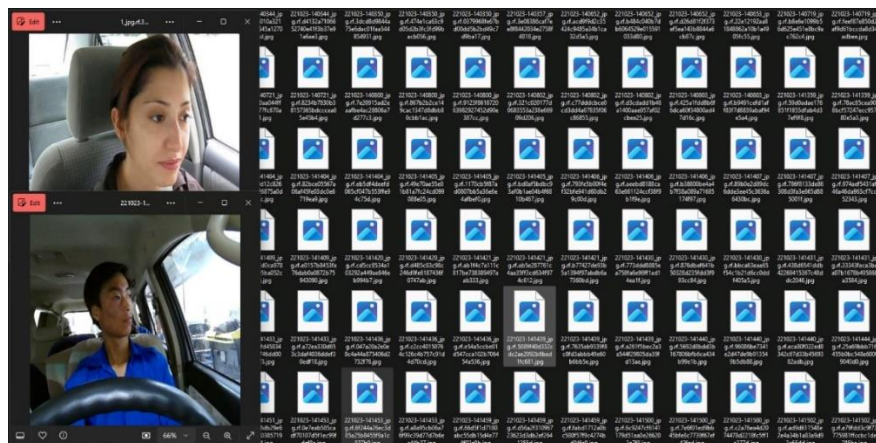
- 1) Visual Studio Code
- 2) Ultralytics 8.0.196
- 3) Bahasa Pemrograman Python 3.12.4
- 4) PyTorch 2.3.0
- 5) TensorFlow Lite
- 6) Android Studio Koala 2024.1.1

Spesifikasi perangkat *mobile* yang digunakan untuk menguji aplikasi secara *real-time* yaitu sebagai berikut:

- a. Perangkat *Mobile 1*
 - 1) OS Version: Android10 QP1A.190711.020
 - 2) CPU: Octa-core Max 2.30GHzStorage: 64GBRAM: 4 GB
 - 3) Kamera: 5 MP
- b. Perangkat *Mobile 2*
 - 1) OS Version: Android 13
 - 2) CPU: Snapdragon 8 Gen 2 3.36 GHz
 - 3) Storage: 256 GB
 - 4) RAM: 12 GB
 - 5) Kamera: 12 MP

4.2. Implementasi Data

Penelitian ini menggunakan data berupa citra pengemudi kendaraan roda empat yang menunjukkan bagian kepala hingga pundak pengemudi. Data yang digunakan dikumpulkan dari dataset yang diambil dari website *roboflow* dan *kaggle*. Beberapa contoh data pengemudi kendaraan roda empat yang telah dikumpulkan dapat dilihat pada Gambar 4.1.



Gambar 4.1 Contoh Data Citra Pengendara Motor

4.3. Implementasi Model

Implementasi model menggunakan *software* Visual Studio Code (VS Code). VS code merupakan *Integrated Development Environment* (IDE) yang banyak digunakan dan mudah dioperasikan. VS Code sama halnya dengan Google Collab, namun VS Code

menggunakan konfigurasi lokal dalam menjalankan file notebook. Pelatihan model memanfaatkan *library* dari Ultralytics yang menyediakan YOLOv8.

Pelatihan sistem dilakukan dengan menggunakan VS Code sebagai *tools* dan model *pre-trained* yolov8n.pt, dimana yolov8n merupakan model paling kecil dari algoritma yolov8. Model pertama kali dilatih menggunakan 2000 data citra pengemudi dengan pembagian 70% data *training* dan 30 % data *validation* dengan perulangan 100 *epoch* dengan tiga *batch size* yang berbeda yaitu, *batch size* 8, *batch size* 16, dan *batch size* 32. Adapun *batch size* berfungsi untuk menentukan jumlah sampel data yang akan diproses pada satu waktu, semakin besar nilai *batch size* semakin cepat pula waktu pelatihannya.

Hasil pelatihan dengan 100 *epoch* dan berbagai nilai *batch size* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Percobaan 100 *epoch*

| Batch size | Precision | Recall | mAP50(B) |
|------------|-----------|--------|----------|
| 8 | 0.83276 | 0.79 | 0.821 |
| 16 | 0.82641 | 0.78 | 0.814 |
| 32 | 0.8311 | 0.81 | 0.821 |

Nilai *Precision* memiliki fungsi sebagai pengukur proporsi prediksi positif yang benar dari semua prediksi positif yang dibuat oleh model. *Recall* berfungsi sebagai pengukur proporsi prediksi positif yang benar dari semua sampel yang benar-benar positif. Nilai mAP50 atau *mean Average Precision* (mAP) pada IoU 0.5 adalah metrik yang sering digunakan dalam tugas pendeteksian objek, yang berfungsi sebagai pengukur rata-rata *precision* di berbagai tingkat *recall* untuk *intersection over union* (IoU) dengan threshold 0.5. Sedangkan *Accuracy* memiliki fungsi sebagai rasio prediksi benar positif dan negatif dengan keseluruhan data.

Setelah melakukan seluruh proses pelatihan, penulis menemukan nilai *precision* dan *accuracy* masih terlalu kecil untuk ketiga *batch size* dan terdapat kemungkinan untuk nilai tersebut dapat menjadi lebih tinggi dari yang sudah ada. Kemudian, penulis melakukan teknik *K-Cross Validation* dengan K=5 dan 100 *epoch* pada *batch size* 8,

batch size 16, dan *batch size* 32 dengan *average* pada nilai *precision*, *recall*, mAP50(B) pada setiap *batch size* dapat dilihat pada tabel 4.2.

Tabel 4.2 Nilai *average* hasil percobaan dengan 3 *batch size* berbeda

| Batch size | Precision | Recall | mAP50(B) |
|------------|-----------|--------|----------|
| 8 | 0.9127 | 0.88 | 0.902 |
| 16 | 0,93391 | 0,89 | 0,919 |
| 32 | 0.9121 | 0.89 | 0.902 |

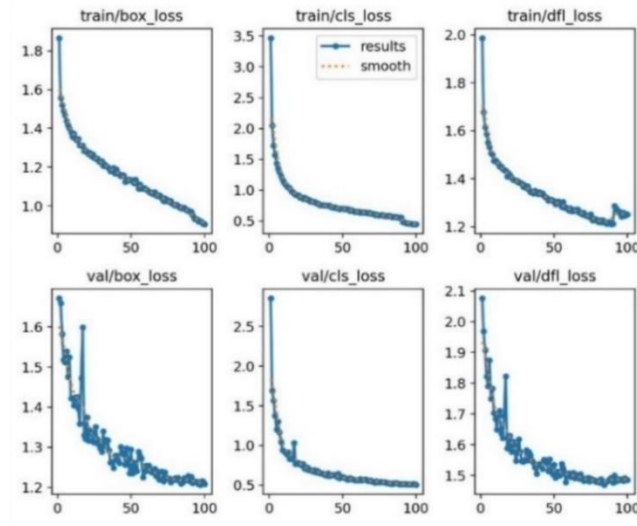
Berdasarkan nilai *average* dari setiap *batch size* pada Tabel 4.2, *batch size* 16 memiliki nilai *average* yang lebih baik dibandingkan dengan dua *batch size* lainnya. Oleh karenanya untuk penelitian selanjutnya penulis melakukan pengujian model dengan menggunakan *batch size* 16 dan 100 data uji untuk mendapatkan nilai *accuracy* pada setiap *fold*nya. Hasil *fold* pelatihan dengan K=5, 100 *epoch*, dan batch 16 dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Percobaan 100 *epoch* K=5

| Fold | Precision | Recall | mAP50(B) | Accuracy |
|----------------|-----------|--------|----------|----------|
| 0 | 0.92088 | 0.90 | 0.922 | 91% |
| 1 | 0.92537 | 0.89 | 0.907 | 90% |
| 2 | 0.95861 | 0.92 | 0.952 | 93% |
| 3 | 0.9388 | 0.88 | 0.907 | 90% |
| 4 | 0.9259 | 0.89 | 0.908 | 91% |
| Average | 0,933912 | 0,89 | 0,919 | 91% |

Meskipun pada pelatihan setiap *fold* 100 *epoch* ini nilai *precision* sudah naik dari percobaan pelatihan sebelumnya, berdasarkan grafik hasil percobaan *fold* 2 yang ditampilkan pada Gambar 4.2, masih terdapat fluktuasi berlebihan terutama pada grafik validasi dimana masih terdapat ketidakstabilan dalam pelatihan data. Hasil percobaan

juga dinilai *overfitting* akibat kerugian (*loss*) pada data *train* menurun dengan baik tetapi kerugian (*loss*) pada validasi tidak turut menurun atau bahkan mengalami peningkatan yang memungkinkan model tidak belajar dengan baik.



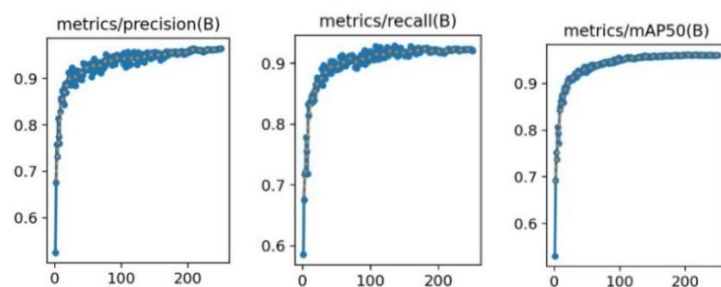
Gambar 4.2 Grafik *fold 2 100 epoch K=5*

Untuk mengatasi permasalahan tersebut penulis melakukan perubahan pada data yang digunakan dalam percobaan penelitian dengan menambahkan tahapan *preprocessing* augmentasi pada beberapa data yang jumlah dataset dalam satu kelasnya masih dinilai sedikit dibandingkan dengan jumlah dataset pada kelas lainnya, yaitu dataset pada kelas kepala_miring, kelas mata_close, dan kelas mulut_open. Selanjutnya penulis kembali melakukan percobaan penelitian menggunakan teknik *K-Cross Validation* dengan $K=5$, 250 *epoch*, dan *batch size* 16. Hasil dari setiap *fold* dapat dilihat pada Tabel 4.4.

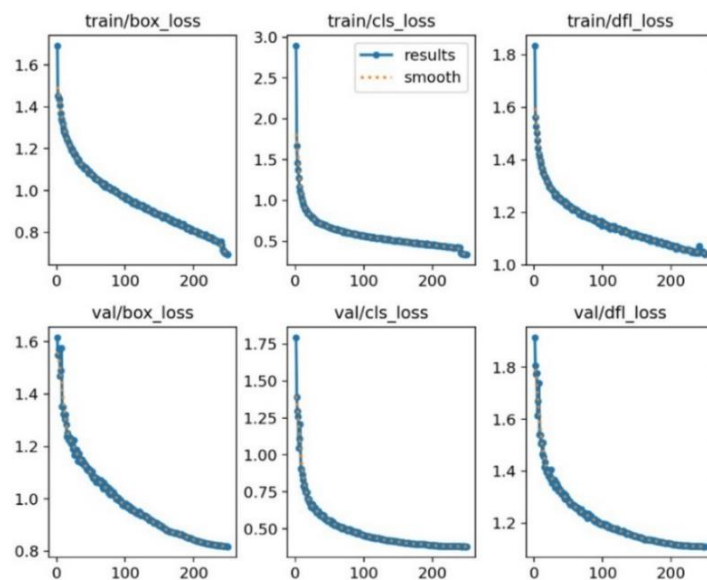
Tabel 4.4 Hasil percobaan 250 *epoch K=5*

| Fold | Precision | Recall | mAP50(B) | Accuracy |
|----------------|-----------|--------|----------|----------|
| 0 | 0.91606 | 0.90 | 0.939 | 93% |
| 1 | 0.94485 | 0.92 | 0.960 | 94% |
| 2 | 0.95529 | 0.93 | 0.964 | 93% |
| 3 | 0.96511 | 0.92 | 0.961 | 95% |
| 4 | 0.93444 | 0.92 | 0.960 | 93% |
| Average | 0,94315 | 0,91 | 0,956 | 93% |

Berdasarkan hasil percobaan pada Tabel 4.3, hasil terbaik diperoleh pada *fold 3* dengan nilai *precision* 0.96511, *recall* 0.92, dan mAP 0.961. Pada Gambar 4.3 grafik *precision* mengalami peningkatan dari mendekati 0.60 ke sekitar 0.90 seiring dengan bertambahnya *epoch*. Hal ini menunjukkan bahwa seiring dengan pelatihan, model menjadi lebih akurat dalam memprediksi kelas positif. Kemudian pada grafik *recall* juga mengalami peningkatan dari mendekati 0,60 ke sekitar 0.90. Hal ini menunjukkan bahwa model semakin baik dalam menemukan semua sampel positif dalam dataset. Selanjutnya pada grafik mAP50 juga terjadi peningkatan kinerja deteksi objek secara keseluruhan dari model.



Gambar 4.3 Grafik *fold 3* 250 *epoch* batch size 16



Gambar 4.4 Grafik *train* dan *val* *fold 3* 250 *epoch* batch size 16

Berdasarkan Gambar 4.4, pada grafik validasi sudah tidak terdapat fluktuasi berlebihan yang menunjukkan bahwa pelatihan pada data sudah cukup stabil. Kerugian

(*loss*) pada data validasi juga ikut menurun seiring dengan turunnya kerugian (*loss*) pada data *train* menunjukkan bahwa model sudah belajar dengan baik.

4.4. Implementasi Antarmuka Aplikasi

4.4.1. Antarmuka Halaman *Splash Screen*

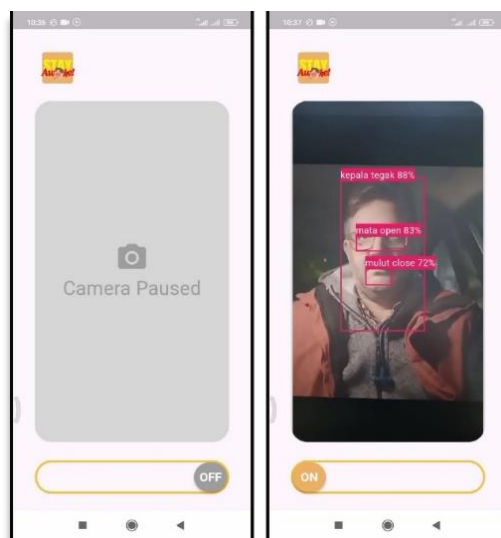
Halaman *splash screen* merupakan halaman yang akan tampil saat pengguna pertama kali membuka aplikasi. Pada halaman *splash screen* ini ditampilkan logo aplikasi. Tampilan antarmuka halaman *splash screen* dapat dilihat pada Gambar 4.5.



Gambar 4.5 Antarmuka Halaman *Splash Screen*

4.4.2. Antarmuka Halaman *Scan*

Halaman *scan* atau halaman utama aplikasi dimana terdapat fitur deteksi yang berfungsi untuk mendeteksi tanda-tanda mengantuk pada pengemudi kendaraan roda empat. Untuk dapat menggunakan aplikasi, Pengguna harus menghidupkan sistem kamera pada aplikasi dengan menggeser tombol *off* pada halaman *scan* menjadi *on*. Kemudian aplikasi akan mengakses kamera pada ponsel dan mulai mendeteksi secara *real-time*. Tampilan antarmuka halaman *scan* dapat dilihat pada Gambar 4.6.



Gambar 4.6 Antarmuka Halaman *Scan*

4.4.3. Antarmuka Halaman *Pop Up* Pemberitahuan

Halaman pemberitahuan merupakan halaman yang muncul apabila sistem berhasil mendeteksi kantuk pada pengemudi. Aplikasi kemudian akan memunculkan bunyi berupa *alarm* peringatan sebagai upaya untuk pengemudi tetap terjaga. Pada halaman ini terdapat tombol yang harus digeser oleh pengemudi untuk dapat mematikan *alarm* yang berbunyi, dengan ini pengemudi dapat kembali sadar dan terjaga untuk dapat mematikan *alarm*. Tampilan antarmuka halaman pemberitahuan dapat dilihat pada Gambar 4.7.



Gambar 4.7 Antarmuka Halaman *Pop Up* Pemberitahuan





4.5. Pengujian Sistem

Pada tahap ini, uji coba dilakukan pada model pelatihan (*trained model*) hasil pelatihan dari proses sebelumnya untuk mengetahui kemampuan model dalam mengenali pengemudi kendaraan roda empat yang sedang dalam kondisi mengantuk (*sleepy*) atau terjaga (*awake*). Hasil pengujian dapat dilihat pada Tabel 4.5.





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| No. | Citra | Class Aktual | Class Prediksi | Persentase Akurasi Class | Kombinasi Class Aktual | Kombinasi Class Prediksi | Keterangan |
|-----|---|--|--|--------------------------|------------------------|--------------------------|------------|
| 1. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 2. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 3. |  | kepala_tegak mata_open mulut_open | kepala_tegak mata_open mulut_open | 100% | <i>Awake</i> | <i>Awake</i> | Benar |





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|----|---|---|---|--------|--------|-------|-------|
| 4. |  | kepala_tegak mata_open mulut_open | kepala_tegak mata_open mulut_open | 100% | Awake | Awake | Benar |
| 5. |  | kepala_tegak mata_open mulut_open | kepala_tegak mata_open mulut_open | 100% | Awake | Awake | Benar |
| 6. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_open | 100% | Awake | Awake | Benar |
| 7. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_open mulut_open | 33,33% | Sleepy | Awake | Salah |

Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|--|---|--------|---------------|---------------|-------|
| 8. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close | 66,67% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 9. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 10. |  | kepala_miring mata_close mulut_close | kepala_tegak mata_close mulut_close | 66,67% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 11. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|---|---|--------|---------------|--------------------|-------|
| 12. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 13. |  | kepala_tegak mata_open mulut_close | kepala_tegak | 33,33% | <i>Awake</i> | <i>[undefined]</i> | Salah |
| 14. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_close | 100% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 15. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |

Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|---|---|--------|---------------|---------------|-------|
| 16. |  | kepala_tegak mata_open mulut_close | mata_open mulut_close | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 17. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_close | 100% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 18. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 19. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|---|---|--------|---------------|---------------|-------|
| 20. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_close | 100% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 21. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 22. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 23. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_close | 100% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|---|---|--------|---------------|---------------|-------|
| 24. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 25. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_close | 100% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 26. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_close mulut_close | 66,67% | <i>Awake</i> | <i>Sleepy</i> | Salah |
| 27. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |





Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|---|--|--------|---------------|---------------|-------|
| 28. |  | kepala_tegak mata_open mulut_close | mata_open mulut_close | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 29. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 30. |  | kepala_tegak mata_close mulut_close | kepala_tegak mata_close mulut_open | 66,67% | <i>Sleepy</i> | <i>Sleepy</i> | Benar |
| 31. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |


Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|--|--|--------|--------------|--------------|-------|
| 32. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 33. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open | 66,67% | <i>Awake</i> | <i>Awake</i> | Benar |
| 34. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 35. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |

Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|--|--|--------|--------------|--------------------|-------|
| 36. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 37. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 38. |  | kepala_tegak mata_open mulut_close | kepala_tegak mata_open mulut_close | 100% | <i>Awake</i> | <i>Awake</i> | Benar |
| 39. |  | kepala_tegak mata_open mulut_close | kepala_tegak | 33,33% | <i>Awake</i> | <i>[undefined]</i> | Benar |

Tabel 4.5 Hasil Pengujian Sistem Deteksi Kantuk Pada Pengemudi

| | | | | | | | |
|-----|---|--|---|--------|--------|--------|-------|
| 40. |  | kepala_miring mata_close mulut_close | kepala_tegak mata_close mulut_close | 66,67% | Sleepy | Sleepy | Benar |
|-----|---|--|---|--------|--------|--------|-------|

Penelitian ini menggunakan 2 perangkat *mobile* sistem aplikasi dengan versi sistem operasi android dan spesifikasi kamera depan yang berbeda. Didapatkan pengujian sistem aplikasi dengan perangkat *mobile* 2 yang memiliki spesifikasi kamera depan 12MP mampu memunculkan *bounding box* sesuai dengan hasil pelatihan model pada sistem dibandingkan dengan pengujian sistem aplikasi dengan perangkat *mobile* 1 yang spesifikasi kamera depannya hanya 5MP.

Berdasarkan hasil pengujian menggunakan 120 data uji, terdapat 5data uji yang tidak dapat teridentifikasi *classnya* atau terdapat data yang benar-benar ada namun tidak diberi label atau diprediksi oleh sistem dikarenakan faktor *delay* dalam waktu 226 ms sampai 366 ms pada sistem yang membuat lambatnya sistem aplikasi memunculkan *bounding box* pada data uji. Adapun skenario pengujian berdasarkan tingkat pencahayaan pada sistem aplikasi untuk dapat memunculkan *bounding box* pada data uji dapat dilihat pada Tabel 4.6.

Tabel 4.6 Pengujian *Bounding Box* berdasarkan tingkat pencahayaan

| No. | Nilai Lux (lx) | Hasil |
|-----|----------------|--------------|
| 1. | $735 \geq$ | Tidak Muncul |
| 2 | 26 - 735 | Muncul |
| 3. | ≤ 26 | Tidak Muncul |

Berdasarkan Tabel 4.6, apabila pengujian dilakukan pada tingkat pencahayaan dibawah 26 lx maka sistem aplikasi tidak dapat memunculkan *bounding box* akibat tingkat pencahayaan yang terlalu gelap sehingga sistem tidak dapat mendeteksi fitur pada citra pengemudi. Begitu pun pengujian dengan tingkat pencahayaan diatas 735 lx,

sistem aplikasi tidak dapat memunculkan *bounding box* akibat tingkat pencahayaan yang terlalu terang sehingga sistem aplikasi tidak dapat menangkap citra pengemudi secara jelas. Oleh karenanya, lingkungan dengan tingkat pencahayaan diantara 26 lx sampai 735 lx merupakan lingkungan yang ideal untuk sistem aplikasi dapat bekerja dengan baik.

Hasil perhitungan pada data yang dapat terdeteksi oleh sistem yang dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil pengujian deteksi kantuk pada pengemudi

| | | Aktual | | Total |
|----------|---------------|--------------|---------------|-------|
| | | <i>Awake</i> | <i>Sleepy</i> | |
| Prediksi | <i>Awake</i> | 88 | 3 | 91 |
| | <i>Sleepy</i> | 2 | 22 | 24 |
| Total | | 90 | 25 | 115 |

Berdasarkan pada Tabel 4.7, pada pengujian ini terdapat beberapa kesalahan dalam hasil deteksi. Sistem keliru dalam mendeteksi kelas *awake* sebanyak dua kali yang menghasilkan kelas *sleepy*. Pada kelas *sleepy* juga terdapat kesalahan deteksi sebanyak tiga kali dengan menghasilkan kelas *awake*. Kesalahan ini dapat terjadi dikarenakan waktu tunda (*delay*) pada sistem dalam memunculkan *bounding box*. Berdasarkan hasil pengujian tersebut, dapat diketahui nilai kombinasi pada *confusion matrix* yang dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil perhitungan confusion matrix pada pengujian

| <i>Class</i> | TP (<i>True Positive</i>) | FP (<i>False Positive</i>) | FN (<i>False Negative</i>) |
|---------------|--------------------------------|---------------------------------|---------------------------------|
| <i>Awake</i> | 88 | 2 | 3 |
| <i>Sleepy</i> | 22 | 3 | 2 |
| Total | 110 | 5 | 5 |

Berikut perhitungan nilai *precision*, *recall*, F1-Score, dan *accuracy*:

a. *Precision*

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

$$Precision\ Awake = \frac{88}{88 + 2} = 0.97$$

$$Precision\ Sleepy = \frac{22}{22 + 3} = 0.88$$

b. *Recall*

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

$$Recall\ Awake = \frac{88}{88 + 3} = 0.96$$

$$Recall\ Sleepy = \frac{22}{22 + 2} = 0.91$$

c. *F1-Score*

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

$$F1 - Score\ Awake = 2 \times \frac{0.96 \times 0.97}{0.96 + 0.97} = 0.96$$

$$F1 - Score\ Sleepy = 2 \times \frac{0.91 \times 0.88}{0.91 + 0.88} = 0.89$$

d. *Accuracy*

$$Accuracy = \frac{TP + TN}{Total\ Data} \times 100 \%$$

$$Accuracy = \frac{88 + 22}{115} \times 100\% = 95\%$$

Tabel 4.9 Nilai Precision, Recall, dan F1-Score

| <i>Class</i> | <i>Precision</i> | <i>Recall</i> | <i>F1-Score</i> |
|---------------|------------------|---------------|-----------------|
| <i>Awake</i> | 0.97 | 0.96 | 0.96 |
| <i>Sleepy</i> | 0.88 | 0.91 | 0.89 |

Pada Tabel 4.9 terdapat hasil perhitungan nilai metrik *precision*, *recall*, dan *F1-Score* pada masing-masing kelas. Nilai *Precision* pada kelas *Awake* sebesar 97% sedangkan

pada kelas *Sleepy* sebesar 88%, Nilai *Recall* pada kelas *Awake* sebesar 96% sedangkan pada kelas *Sleepy* sebesar 91%, dan Nilai *F1-Score* pada kelas *Awake* sebesar 96% dan pada kelas *Sleepy* sebesar 89%.

Deteksi kelas *Sleepy* pada pengemudi memiliki nilai *precision* yang lebih rendah dibandingkan dengan kelas *Awake* dikarenakan pada sistem aplikasi terdapat *delay* dalam munculnya *bounding box* dengan keadaan *real-time* pengemudi. Oleh karenanya, aktivitas perubahan pergerakan pengemudi yang dapat berubah dari sikap terjaga (*Awake*) ke munculnya tanda kelelahan atau mengantuk (*sleepy*) pada pengemudi lalu kembali ke sikap terjaga (*Awake*) dalam waktu kurang dari yang sudah diatur penangkapannya oleh kamera sistem aplikasi yaitu per 0,5 detik dapat mempengaruhi hasil *bounding box* yang muncul pada sistem aplikasi pada saat yang bersamaan dengan keadaan *real-time*.

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah melalui beberapa pengujian deteksi kantuk pada pengemudi kendaraan roda empat, dapat diambil kesimpulan sebagai berikut:

- 1) Penggunaan algoritma YOLO sebagai model deteksi kantuk pada pengemudi kendaraan roda empat bekerja dengan sangat baik.
- 2) Model algoritma You Only Look Once (YOLO) versi 8 memberikan akurasi sebesar 95%.
- 3) Model pada sistem dapat melakukan deteksi secara *real-time* menggunakan perangkat android.
- 4) Sistem aplikasi dapat memunculkan *bounding box* dengan baik pada perangkat dengan spesifikasi kamera depan 12 MP dengan waktu *delay* sebesar 226 ms sampai 366 ms atau 0,2 hingga 0,5 detik dalam menampilkan *bounding box*.
- 5) Penggunaan sistem kurang optimal apabila terjadi perubahan aktivitas pergerakan kepala, mata, dan mulut dari sikap terjaga (*awake*) ke tanda kelelahan atau mengantuk (*sleepy*) dan kembali lagi ke sikap terjaga (*awake*) saat kamera sistem belum menangkap perubahan dalam kurun waktu yang sudah ditentukan pada sistem.

5.2. Saran

Adapun beberapa saran untuk penelitian-penelitian berikutnya meliputi:

- 1) Penelitian berikutnya dapat menggunakan metode yang berbeda dan metode tambahan untuk meningkatkan akurasi seperti metode *Transfer Learning* agar dapat menjadi perbandingan.
- 2) Menggunakan data yang lebih variatif dengan kuantitas data yang proporsional antar kelas data agar tidak terdapat ketimpangan pada data dan dapat menjadi perbandingan.

- 3) Mengembangkan sistem untuk mendeteksi aktivitas lainnya pada pengemudi kendaraan roda empat untuk meningkatkan keselamatan dalam berkendara.

DAFTAR PUSTAKA

- Amzar, R., Rozali, F., Fadilah, S. I., Rahman, A., Shariff, M., Mohd Zaini, K., Karim, F., Helmy, M., Wahab, A., Thangaveloo, R., Samad, A., & Shibghatullah, B. (2022). Driver Drowsiness Detection and Monitoring System (DDDMS). In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 13, Issue 6). www.ijacsa.thesai.org
- Ding, C., Wang, S., Liu, N., Xu, K., Wang, Y., & Liang, Y. (2019). REQ-YOLO: A resource-aware, efficient quantization framework for object detection on FPGAS. *FPGA 2019 - Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 33–42. <https://doi.org/10.1145/3289602.3293904>
- Hidayatulloh, M. S. (2021). *Sistem Pengenalan Wajah Menggunakan Metode YOLO (YouOnlyLookOnce)*. <https://repository.dinamika.ac.id/id/eprint/5568/2/15410200028-2021-UniversitasDinamika.pdf>
- Indra, D. A. A., Yanda, N., Purnamasari, D., & Arrafi, M. F. (2023). Implementasi Algoritma YoloV5 untuk Kantuk Detektor pada Pengemudi Kendaraan Bermotor secara Realtime. *Jurnal Ilmiah Komputasi*, 22(4).
- Lin, H., Deng, J. D., Albers, D., & Siebert, F. W. (2020). Helmet Use Detection of Tracked Motorcycles Using CNN-Based Multi-Task Learning. *IEEE Access*, 8, 162073–162084. <https://doi.org/10.1109/ACCESS.2020.3021357>
- Maslikah, S., Alfita, R., & Ibadillah, A. F. (2020). *Sistem Deteksi Kantuk Pada Pengendara Roda Empat Menggunakan Eye Blink Detection*. <https://journal.fortei7.org/index.php/fortech/article/view/221>
- Maudisha. (2023, January). *Mengenal Microsleep dan Bahayanya*. <https://Www.Ui.Ac.Id/Mengenal-Microsleep-Dan-Bahayanya/>.
- Mu, Z., Jin, L., Yin, J., & Wang, Q. (2022). Research on a Driver Fatigue Detection Model Based on Image Processing. *Human-Centric Computing and Information Sciences*, 12. <https://doi.org/10.22967/HGIS.2022.12.017>
- N, A., & Anusudha, K. (2023). Real time face recognition system based on YOLO and InsightFace. *Multimedia Tools and Applications*. <https://doi.org/10.1007/s11042-023-16831-7>

- Redmon, J. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- Safarov, F., Akhmedov, F., Abdusalomov, A. B., Nasimov, R., & Cho, Y. I. (2023). Real-Time Deep Learning-Based Drowsiness Detection: Leveraging Computer-Vision and Eye-Blink Analyses for Enhanced Road Safety. *Sensors*, 23(14). <https://doi.org/10.3390/s23146459>
- Santika, E. F. (2023, May 24). *Ini Statistik Kejadian hingga Jumlah Korban Kecelakaan LaluLintasdiIndonesia*. <https://Databoks.Katadata.Co.Id/Datapublish/2023/05/24/Ini-Statistik-Kejadian-Hingga-Jumlah-Korban-Kecelakaan-Lalu-Lintas-Di-Indonesia>.
- Schutte, P., & Maldonado, C. (2003). *Factors affecting driver alertness during the operation of haul trucks in the South African mining industry*.
- Sinha, A., Aneesh, R. P., & Gopal, S. K. (2021, March 25). Drowsiness Detection System Using Deep Learning. *Proceedings of 2021 IEEE 7th International Conference on Bio Signals, Images and Instrumentation, ICBSII 2021*. <https://doi.org/10.1109/ICBSII51839.2021.9445132>
- Sprajcer, M., Dawson, D., Kosmadopoulos, A., Sach, E. J., Crowther, M. E., Sargent, C., & Roach, G. D. (2023). How Tired is Too Tired to Drive? A Systematic Review Assessing the Use of Prior Sleep Duration to Detect Driving Impairment. In *Nature and Science of Sleep* (Vol. 15, pp. 175–206). Dove Medical Press Ltd. <https://doi.org/10.2147/NSS.S392441>
- Summer, J., & Rehman, A. (2023, January 6). *Microsleep: What Is It, What Causes It, and Is It Safe?* Sleep Foundation. <https://www.sleepfoundation.org/how-sleep-works/microsleep>
- Uma Maheswari, V., Aluvalu, R., Prasad Kantipudi, M. V. V., Chennam, K. K., Kotecha, K., & Saini, J. R. (2022). Driver Drowsiness Prediction Based on Multiple Aspects Using Image Processing Techniques. *IEEE Access*, 10, 54980–54990. <https://doi.org/10.1109/ACCESS.2022.3176451>