JURNAL ILMU KOMPUTER
Volume 4 Nomor 1, Januari, 2026
ISSN: 3031-125X (ONLINE)
Penerbit: Program Studi Teknik Informatika S-2 Universitas Pamulang

Implementasi Yolov5 Deteksi Mata Lelah Berbasis Android

*Ajeng Permata Suri[1], Arya Adhyaksa Waskita[2], Mardiyanto[3]

[1,2,3)] Teknik Informatika S-2, Program Pascasarjana, Universitas Pamulang**,** Kota Tangerang Selatan, Banten.

Email: [1]ajengpermatasuri@gmail.com, [2]aawaskita@unpam.ac.id, [3]dosen00027@unpam.ac.id

ABSTRACT

Excessive screen exposure can trigger digital eye strain, reducing visual comfort, attention, and overall productivity. Prior studies in computer vision indicate that deep learning–based object detection, particularly the YOLO family, can recognize facial and eye-related visual patterns efficiently, making it suitable for early-warning systems on mobile devices. This study aims to implement YOLOv5 to detect signs of eye fatigue in real time using the front camera of an Android smartphone. The novelty of this work lies in deploying a lightweight object-detection model on-device through TensorFlow Lite and integrating an automatic notification mechanism as a preventive intervention. The proposed methodology includes collecting and labeling an eye-image dataset into two classes (awake and drowsy), training a YOLOv5 model in Google Colab, optimizing and converting the trained model to TensorFlow Lite, and integrating it into an Android application for live-camera inference. System performance is evaluated using accuracy, precision, recall, and inference speed (FPS). Experimental results show that the system achieves 95.6% accuracy, 94.3% precision, 96.1% recall, and an Average speed of 22 FPS, enabling responsive detection and timely notifications. In conclusion, the Android-based YOLOv5 implementation is feasible as a preventive solution to help users monitor eye-fatigue symptoms and encourage healthier screen-use habits.

Keywords: YOLOv5, Object Detection, Eye Fatigue, Real-time, Android, TensorFlow Lite

ABSTRAK

Penggunaan layar secara berlebihan dapat memicu digital eye s*train* (kelelahan mata digital) yang berdampak pada kenyamanan, fokus, dan produktivitas. Berbagai studi di bidang computer vision menunjukkan bahwa pendekatan deep learning berbasis deteksi objek, seperti keluarga YOLO, mampu mengenali pola visual wajah dan mata secara cepat sehingga berpotensi diterapkan sebagai sistem peringatan dini pada perangkat bergerak. Penelitian ini bertujuan mengimplementasikan YOLOv5 untuk mendeteksi indikasi mata lelah secara *real-time* melalui kamera depan Android, sekaligus menghadirkan kebaruan berupa integrasi model deteksi ke aplikasi mobile yang ringan dengan dukungan *TensorFlow Lite* serta mekanisme notifikasi otomatis sebagai tindakan preventif. Metodologi penelitian meliputi pengumpulan dan pelabelan dataset citra mata dengan dua kelas (*awake* dan *drowsy*), pelatihan model YOLOv5 pada lingkungan *Google Colab*, optimasi dan konversi model ke format *TensorFlow Lite*, serta integrasi ke aplikasi Android untuk inferensi langsung dari *live camera*. Evaluasi kinerja dilakukan menggunakan metrik akurasi, *precision*, *recall*, dan kecepatan inferensi. Hasil pengujian menunjukkan sistem mencapai akurasi 95,6%, *precision* 94,3%, *recall* 96,1%, serta kecepatan rata-rata 22 FPS, sehingga mampu memberikan deteksi dan notifikasi secara responsif. Kesimpulannya, implementasi YOLOv5 berbasis Android layak digunakan sebagai solusi preventif untuk membantu pengguna memantau gejala kelelahan mata dan mendorong kebiasaan penggunaan layar yang lebih sehat.

Kata kunci: YOLOv5, Deteksi Objek, Mata Lelah, *Real-time*, Android, *TensorFlow Lite*

1. PENDAHULUAN

Pada era modern ini, perkembangan teknologi mengalami kemajuan yang sangat pesat. Salah satu inovasi yang sudah sangat familiar di Masyarakat adalah *gadget*, seperti ponsel, tablet, atau laptop. Penggunaan *gadget* telah merambah ke semua lapisan Masyarakat tanpa memandang usia, waktu, tempat, maupun pedangan. *Gadget* kini menjadi bagian tak terpisahkan dari kehidupan sehari-hari, memenuhi kebutuhan untuk berkomunikasi dan bersosialisasi dengan orang lain [1].

Di balik manfaat yang ditawarkan oleh *gadget*, penggunaannya yang berlebihan dapat memunculkan berbagai masalah kesehatan, salah satunya adalah mata lelah atau *asthenopia* [2]. Mata lelah sering dialami oleh pengguna *gadget* yang menggunakan perangkat dalam posisi ergonomis atau dalam waktu yang lama. Gejala yang umum terjadi adalah sakit kepala, penglihatan kabur, iritasi, serta rasa tidak nyaman pada mata. Penggunaan *gadget* lebih dari 60 menit tanpa istirahat atau laptop lebih dari 4 jam dapat memperparah kondisi ini akibat stress pada otot mata yang dipaksa bekerja keras dalam waktu lama.

Berdasarkan data Kementerian Kesehatan RI dan survei APJII tahun 2023, penggunaan gadget pada anak dan remaja di Indonesia tergolong sangat tinggi. Sebanyak 89% anak usia 5–12 tahun dan 97% remaja usia 13–18 tahun menggunakan *gadget* setiap hari, dengan durasi rata-rata 5–8 jam per hari di luar kebutuhan sekolah. Riset tersebut juga melaporkan sekitar 35% dari kelompok tersebut mengalami gejala *digital eye strain* (kelelahan mata digital). Gambaran kondisi di lingkungan sekolah juga terlihat dari kegiatan edukasi kesehatan mata yang dilaksanakan pada 25 Juni 2025 di SMK Kencana Bandung dengan sasaran 50 siswa kelas XI. Kegiatan dilakukan melalui ceramah interaktif, diskusi kelompok terfokus, edukasi visual melalui *leaflet*, dan permainan edukatif berbasis digital. Hasil evaluasi menunjukkan peningkatan pengetahuan siswa dari 30,4% pada *pre-test* menjadi 93% pada *post-test*, serta 84% siswa menyatakan komitmen menerapkan metode 20-20-20 dalam penggunaan gadget. [3]

Dengan adanya temuan tersebut, terlihat bahwa risiko mata lelah akibat penggunaan layar berlebih perlu ditangani melalui pendekatan yang lebih praktis dan dapat membantu deteksi dini. Perkembangan kecerdasan buatan, khususnya pada bidang *computer vision*, membuka peluang untuk memantau kondisi mata dan wajah pengguna secara *real-time* melalui kamera perangkat. Berbagai penelitian di bidang keselamatan

berkendara telah memanfaatkan model deteksi objek seperti YOLOv5 untuk mendeteksi kelelahan pengemudi berdasarkan ciri wajah (kondisi mata, kedipan, dan ekspresi), termasuk pada skenario kompleks seperti jalan pegunungan dengan pencahayaan rendah, dengan akurasi rata-rata sekitar 85% untuk pengenalan status kelelahan pengemudi. [4]. Penelitian lain mengintegrasikan YOLOv5 ringan (YOLOv5s) dengan *facial 3D keypoints* dan modul *attention* untuk meningkatkan ketepatan deteksi mengantuk sekaligus menjaga kemampuan kerja secara *real-time* pada sistem pemantauan pengemudi [5].

Selain pada keselamatan berkendara, *computer vision* berbasis deteksi objek juga banyak diterapkan di bidang kesehatan karena mampu mengenali pola visual pada wajah dan mata secara cepat dan presisi. Hal ini menunjukkan bahwa YOLOv5 berpotensi digunakan untuk pemantauan kondisi mata secara *real-time* pada perangkat yang digunakan sehari-hari.

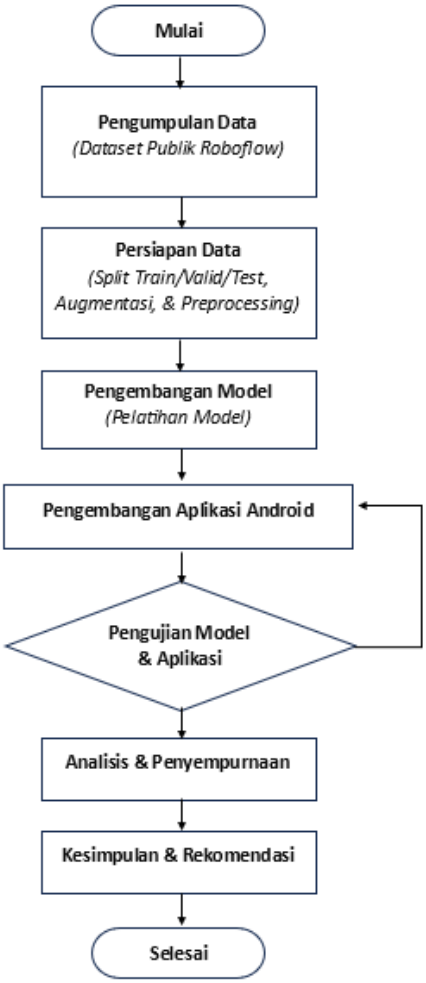
Penelitian ini membangun aplikasi Android berbasis deteksi objek *real-time* untuk mengidentifikasi indikasi mata lelah melalui kamera depan. YOLOv5 dipilih karena mampu melakukan deteksi dengan cepat dan akurat. Model hasil pelatihan dikonversi ke *TensorFlow Lite* dan diintegrasikan ke aplikasi agar efisien di perangkat *mobile*. Sistem melakukan inferensi dari *live camera* dan mengirim notifikasi otomatis ketika indikasi mata lelah terdeteksi sebagai langkah preventif.

Penelitian ini diharapkan berkontribusi pada pencegahan risiko kesehatan akibat penggunaan gadget berlebih, sekaligus menambah referensi pengembangan aplikasi *AI/computer vision* untuk kesehatan mata digital. Hasilnya ditargetkan menghasilkan sistem yang akurat, efisien, dan mudah digunakan oleh masyarakat pada konteks pendidikan, pekerjaan, maupun aktivitas harian.

2. METODE

Metode penelitian ini disusun untuk menjelaskan tahapan teknis dalam mengembangkan sistem deteksi mata lelah berbasis YOLOv5 pada perangkat Android. Penelitian menggunakan pendekatan eksperimental kuantitatif, di mana model YOLOv5 dilatih, dioptimalkan, dan diimplementasikan sebagai model deteksi objek *real-time.* Tahapan penelitian meliputi pengumpulan data, praproses data, pelatihan model, konversi model, integrasi ke aplikasi Android, dan pengujian sistem secara langsung[6]. Tahap ini

adalah tahap dimana dibuatkan suatu rancangan untuk memproses data mulai dari Pengumpulan data sampai dengan Evaluasi Efektivitas dengan Bahasa pemograman *Python* dan *Kotlin*.



Gambar 1. Perancangan Penelitian

2.1. Pengumpulan dan Persiapan Data

Dataset yang digunakan pada penelitian ini adalah *Drowsiness Detection – Choyoooo* yang tersedia secara publik pada *Roboflow Universe.* Dataset ini terdiri dari dua kelas, yaitu *Awake* (terjaga) dan *Drowsy* (mengantuk), serta sudah dilengkapi anotasi *bounding box* sehingga sesuai untuk pelatihan model deteksi objek. Dataset dipilih karena telah menyediakan struktur direktori yang kompatibel dengan YOLOv5 (*train/valid/test*) dan format anotasi yang sesuai standar YOLO.

Berdasarkan metadata *Roboflow Universe*, dataset yang digunakan merupakan versi v5 dengan tanggal rilis/pembaruan 6 Desember 2023[7]. Dataset diekspor dalam

format YOLO dan digunakan dengan pembagian data 70% untuk pelatihan (*training*), 20% untuk *valid*asi (*validation*), dan 10% untuk pengujian (*testing*).

Tahapan persiapan data meliputi: (1) konfirmasi struktur direktori agar sesuai standar YOLOv5 (*folder images* dan *labels* pada *train, valid,* dan *test*), (2) verifikasi label untuk memastikan format anotasi benar serta konsisten dengan dua kelas yang digunakan (*Awake dan Drowsy*), dan (3) praproses citra berupa penyeragaman ukuran masukan, normalisasi, serta augmentasi dasar untuk meningkatkan variasi data dan memperbaiki generalisasi model.

2.2. Pengembangan Model

Model dikembangkan menggunakan arsitektur YOLOv5, YOLOv5 adalah algoritma deteksi objek berbasis *deep learning* tipe *one-stage detector* yang melakukan lokalisasi dan klasifikasi objek sekaligus dalam satu tahap, sehingga cepat dan cocok untuk aplikasi *real-time*. Arsitekturnya terdiri dari *backbone, neck*, dan *head* yang bekerja untuk mengekstraksi dan menggabungkan fitur multi-skala lalu menghasilkan prediksi *bounding box* dan kelas secara *end-to-end*[8]. Varian ringannya, seperti YOLOv5s dan YOLOv5-tiny, memberikan kompromi yang baik antara ukuran model, kecepatan, dan akurasi sehingga ideal untuk perangkat dengan sumber daya terbatas [9] Tahapan pengembangan meliputi Pemilihan model *pre-trained (ImageNet*) sebagai dasar *transfer learning*, Proses *fine-tuning* pada dataset *awake/drowsy* untuk menyesuaikan parameter model, Pelatihan model YOLOv5 secara *end-to-end* agar mampu mendeteksi kondisi mata secara cepat dan akurat. Model YOLOv5 yang telah dilatih dikonversi ke format *TensorFlow Lite (TFLite*) atau *ONNX* agar dapat dijalankan pada perangkat *mobile*. Tahapan pengembangan aplikasi meliputi Perancangan antarmuka (UI) menggunakan Android Studio (*Kotlin*) untuk menampilkan hasil deteksi secara *real-time*, Integrasi model *TFLite/ONNX* dengan pipeline kamera *(CameraX),* Implementasi fitur tambahan seperti sensitivitas deteksi, peringatan otomatis, dan pencatatan riwayat deteksi.

2.3. Pengujian Aplikasi dan Model

Pengujian dilakukan untuk menilai performa model dan stabilitas aplikasi pada beberapa perangkat Android. Aspek yang diuji meliputi: (1) akurasi deteksi pada kondisi nyata dengan variasi pencahayaan, jarak, dan sudut wajah; (2) kecepatan inferensi (FPS)

untuk memastikan kinerja *real-time*; (3) penggunaan sumber daya perangkat (CPU dan RAM); serta (4) stabilitas aplikasi pada penggunaan jangka panjang. Selain uji *real-time*, model juga dievaluasi menggunakan dataset uji (*test* set) dengan metrik akurasi, *Precision, recall, F1-score*, dan *AUC–ROC*.

Makna masing-masing metrik adalah sebagai berikut: akurasi menunjukkan proporsi prediksi yang benar terhadap seluruh data uji; *precision* menunjukkan ketepatan prediksi kelas target (misalnya *drowsy*), yaitu seberapa banyak prediksi *drowsy* yang benar; *recall* menunjukkan tingkat ketercakupan, yaitu seberapa banyak kondisi *drowsy* yang berhasil terdeteksi dari seluruh data *drowsy* yang sebenarnya; *F1-score* merupakan rata-rata harmonik antara *precision* dan *recall* sehingga menggambarkan keseimbangan keduanya; sedangkan *AUC–ROC* menggambarkan kemampuan model membedakan dua kelas pada berbagai ambang keputusan (*threshold*), di mana nilai yang lebih tinggi menunjukkan pemisahan kelas yang lebih baik[10].

2.4. Analisis dan Penyempurnaan

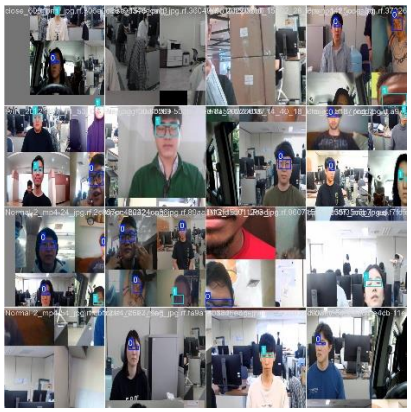
Metode ini dirancang untuk menghasilkan Model deteksi mata lelah berbasis YOLOv5 yang siap dijalankan pada perangkat Android melalui format *TFLite*, Aplikasi Android *real-time* yang stabil, responsif, dan mampu menampilkan peringatan otomatis, Dasar rekomendasi untuk peningkatan lebih lanjut, seperti penambahan variasi dataset, optimasi model, dan fitur lanjutan di aplikasi.

3. HASIL DAN PEMBAHASAN

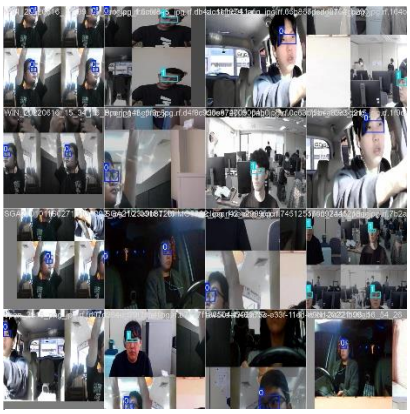
Dalam penelitian ini, peneliti menggunakan platform *Google Colab* dan menerapkan metode YOLOv5. YOLO adalah sebuah jaringan untuk mendeteksi objek, sementara YOLOv5 adalah versi terbaru dari metode YOLO yang telah dikembangkan [11]. Tugas utama dalam pendeteksian objek adalah menemukan lokasi objek yang ada dalam sebuah gambar atau citra dan mengklasifikasikan jenis objeknya. Dengan kata lain, gambar atau citra digunakan sebagai input, dan algoritma akan menghasilkan vektor kotak pembatas dan melakukan prediksi kelas objeknya[12].

Dataset yang digunakan pada penelitian ini merupakan data eksisting (public dataset) dari *Roboflow Universe* berjudul *Drowsiness Detection (Computer Vision Dataset)* yang

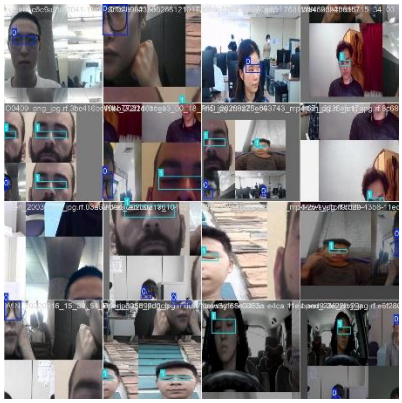
dipublikasikan oleh pengguna choyoooo. Dataset ini berisi ±10.000 citra dengan dua kelas, yaitu *Awake* dan *Drowsy*, serta telah dilengkapi anotasi *bounding box* sehingga sesuai untuk pelatihan model deteksi objek YOLOv5. Dataset diunduh melalui *Roboflow Universe* dalam format YOLOv5 (struktur folder *train*/*valid*/*test* dan berkas label .txt), kemudian digunakan dengan pembagian 70% data pelatihan (*train*), 20% *valid*asi (*valid*), dan 10% pengujian (*test*) sesuai konfigurasi ekspor dataset. Pada tahap praproses, citra diseragamkan ukurannya menjadi 640×640 piksel (menyesuaikan ukuran input/imgsz YOLOv5), dilakukan normalisasi piksel, serta diterapkan augmentasi data untuk meningkatkan variasi dan kemampuan generalisasi model.



Train Batch 0



Train Batch 1



Train Batch 2

Gambar 2. Contoh visualisasi *batch* pelatihan yang dihasilkan otomatis oleh YOLOv5 (*Train Batch* 0–2) setelah augmentasi (misalnya mosaic) beserta anotasi *bounding box* kelas *Awake* dan *Drowsy*.

Train Batch 0–2 pada Gambar 2, istilah tersebut bukan pembagian dataset baru, melainkan visualisasi contoh *batch* pelatihan yang dihasilkan otomatis oleh YOLOv5 selama proses *train*ing (file keluaran seperti *train_batch*0.jpg, *train_batch*1.jpg, *train_batch*2.jpg). Gambar ini menampilkan beberapa citra yang digabung (mosaic)

beserta *bounding box* hasil anotasi setelah proses augmentasi, sehingga berfungsi untuk memverifikasi bahwa label, kelas (*Awake*/*Drowsy*), dan posisi *bounding box* sudah terbaca benar oleh model serta untuk melihat keragaman data pelatihan.

3.1. Arsitektur YOLOv5

Model YOLOv5 terdiri dari tiga komponen utama:

1) *Backbone (CSPDarknet53)* – mengekstraksi fitur citra menggunakan struktur *Cross Stage Partial.*
2) *Neck (PANet)* – memperkuat fitur multi-skala dengan path aggregation.
3) *Head* – menghasilkan prediksi *bounding box*, label kelas (terjaga(*awake*)/mengantuk/(*drowsy*)), dan skor kepercayaan.
4) Pelatihan dilakukan menggunakan *transfer learning* dari bobot pra-latih yolov5s.pt

3.2. Pelatihan Model

Model dilatih menggunakan *Google Colab* dengan spesifikasi GPU Tesla T4, Python 3.10, dan PyTorch 2.0. Parameter utama pelatihan:

Tabel 1. Parameter Pelatihan

Parameter	Nilai
Epoch	100
Batch Size	16
Image Size	640×640
Optimizer	(*default* YOLOv5)
Learning Rate	0.001
Momentum	0.937
Weight Decay	0.0005

Proses pelatihan model dilakukan menggunakan *framework Ultralytics* YOLO pada lingkungan *Google Colab*. Dataset yang telah disiapkan dalam format YOLO digunakan sebagai masukan melalui berkas data.yaml. Pelatihan dijalankan menggunakan model awal *pretrained* yolov5s.pt dengan parameter *epochs*, *batch*, dan imgsz untuk mengatur jumlah iterasi pelatihan, ukuran *mini-batch*, serta ukuran input

citra. Potongan perintah pelatihan disajikan pada *Listing 1* agar mudah dibaca dan direplikasi.

Listing 1. Perintah pelatihan model YOLOv5

```
!pip install -q ultralytics
from ultralytics import YOLO

# Memuat model awal (pretrained)
model = YOLO("yolov5s.pt")

# Pelatihan model
results = model.train(
   data="/content/Drowsiness-Detection-5/data.yaml",  # sesuaikan path dataset
   epochs=100,
   batch=64,
   imgsz=640 )
```

Setelah pelatihan selesai, model terbaik disimpan sebagai *best.pt* pada folder keluaran pelatihan. Model ini kemudian dikonversi ke format *TensorFlow Lite* (*TFLite*) agar dapat dijalankan secara efisien pada perangkat Android. Konversi dilakukan melalui perintah *export* dengan menentukan model sumber (*best.pt*), format keluaran (*TFLite*), serta ukuran input (imgsz). Perintah konversi ditunjukkan pada *Listing* 2.

Listing 2. Perintah konversi model terbaik ke *TensorFlow Lite*

```
yolo export model=/content/runs/detect/train/weights/best.pt format=TFLite imgsz=640
device=cpu half=True
```

Hasil konversi menghasilkan berkas model *TFLite* (misalnya *best_float16.TFLite*) yang selanjutnya diintegrasikan ke aplikasi Android untuk inferensi *real-time* melalui kamera depan. Dengan penyajian dalam bentuk *listing*, seluruh perintah dapat disalin langsung,

serta memudahkan proses replikasi dan verifikasi konfigurasi pelatihan maupun konversi model.

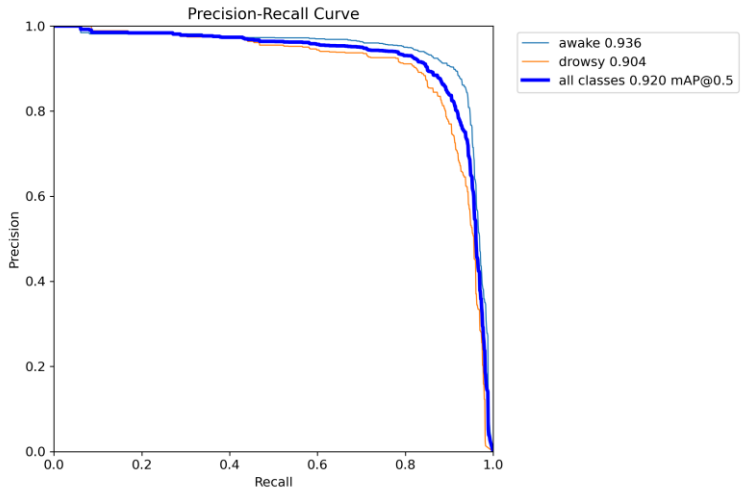
3.3. Evaluasi Model

Evaluasi dilakukan menggunakan *confusion matrix*, *precision-recall Curve*, dan *mean Average precision (mAP)*. Hasil utama pelatihan :

Tabel 2. Hasil Pelatihan

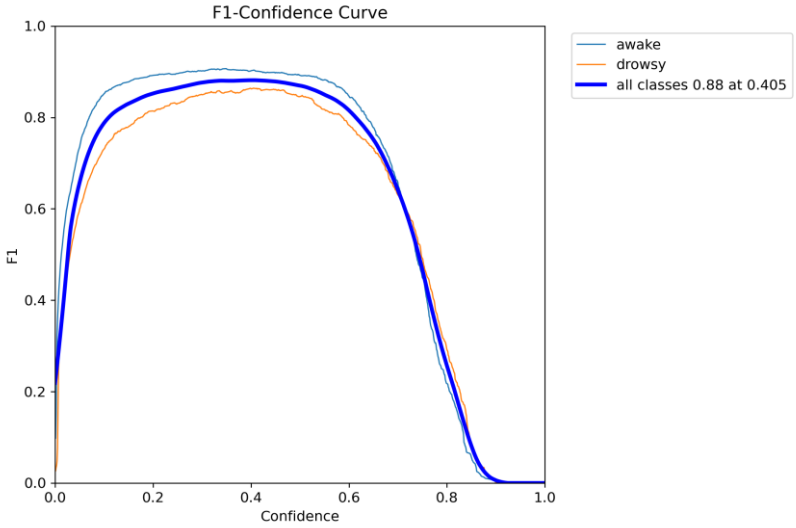
Metrik	**Nilai**
mAP@0.5	0.920
mAP@0.5–0.95	0.410
Precision	1.000
Recall	0.980
F1-score	0.880

Model menunjukkan konvergensi stabil, tanpa indikasi *overfitting*, untuk melihat kualitas deteksi model pada data uji (*test* set), hasil evaluasi ditampilkan melalui tiga visualisasi utama, yaitu *Precision–Recall Curve, F1–Confidence Curve*, dan *Confusion Matrix.* Ketiga grafik ini digunakan untuk menilai ketepatan deteksi, keseimbangan *precision–recall*, serta pola kesalahan klasifikasi antara kelas *awake* dan *drowsy*.



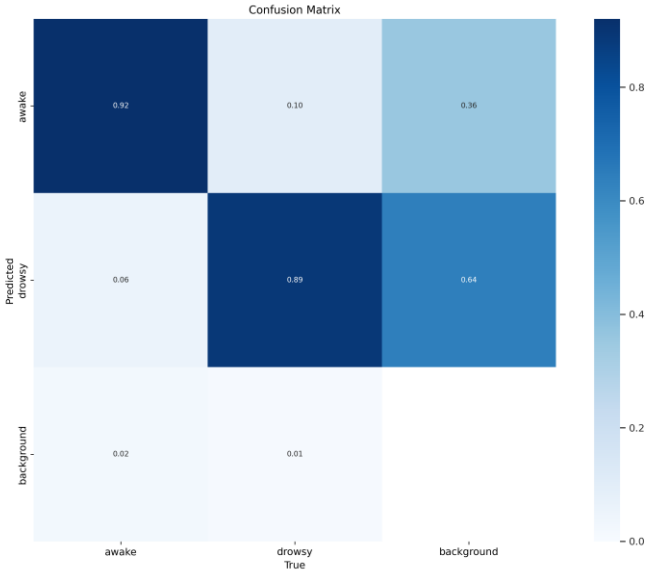
Gambar 5. Kurva Hasil Pelatihan YOLOv5 *Precision-Recall Curve*

Pada Gambar 5 Kurva hasil pelatihan YOLOV5 *Precision-Recall* menunjukkan hubungan antara *precision* (ketepatan prediksi) dan *recall* (ketercakupan deteksi) pada berbagai ambang keputusan. Kurva yang berada tinggi dan mendekati sudut kanan atas mengindikasikan performa deteksi yang baik. Pada grafik ini, nilai AP (*Average Precision*) per kelas adalah *awake* = 0,936 dan *drowsy* = 0,904, sedangkan *mAP*@0,5 untuk seluruh kelas = 0,920. Nilai *mAP*@0,5 yang tinggi menunjukkan bahwa model mampu mendeteksi objek dengan ketepatan *bounding box* yang baik pada ambang IoU 0,5.



Gambar 6. Kurva Hasil Pelatihan YOLOv5 *F1-Confidence Curve*

Pada Gambar 6 (*F1–Confidence Curve*) digunakan untuk menentukan *confidence threshold* yang optimal, karena *F1-score* merepresentasikan keseimbangan antara *precision* dan *recall*. Grafik menunjukkan puncak F1 seluruh kelas ≈ 0,88 pada *confidence* ≈ 0,405, sehingga nilai *threshold* sekitar 0,40 dapat digunakan sebagai acuan untuk memperoleh deteksi yang stabil (tidak terlalu banyak *false positive* dan tidak terlalu banyak *missed detection*) saat implementasi *real-time* di Android.



Gambar 7. Kurva Hasil Pelatihan YOLOv5 *Confusion Matrix*

Pada Gambar 7 (*Confusion Matrix*) memperlihatkan sebaran prediksi model terhadap label sebenarnya. Nilai diagonal yang tinggi menunjukkan prediksi benar. Pada grafik terlihat bahwa model mengklasifikasikan *awake* dengan benar sebesar 0,92 dan *drowsy* sebesar 0,89. Kesalahan utama terjadi pada pertukaran antar kelas (misalnya *drowsy* terprediksi *awake* sebesar 0,10 dan *awake* terprediksi *drowsy* sebesar 0,06), sedangkan kasus objek terlewat (*predicted background*) relatif kecil (0,02 untuk *awake* dan 0,01 untuk *drowsy*). Secara umum, hasil ini menunjukkan model mampu membedakan dua kelas dengan baik dan cukup stabil untuk digunakan pada skenario *real-time*.

3.4. Implementasi sistem

Tahap implementasi sistem merupakan proses penerapan model YOLOv5 hasil pelatihan ke lingkungan nyata pada perangkat Android. Implementasi dilakukan dengan mengekspor model terbaik (*best.pt*) ke format *TensorFlow Lite* (*TFLite*) agar dapat dijalankan secara efisien pada perangkat bergerak. Penggunaan *TFLite* memungkinkan inferensi dilakukan secara on-device (tanpa koneksi internet) melalui *TensorFlow Lite* Interpreter, sehingga sistem tetap berjalan stabil pada kondisi penggunaan sehari-hari dan tidak bergantung pada layanan eksternal[13].

Pada sisi aplikasi, kamera depan digunakan untuk menangkap frame secara *real-time*. Setiap frame kemudian dipraproses (penyesuaian ukuran input model dan normalisasi), lalu diproses oleh model *TFLite* untuk menghasilkan prediksi kelas *awake* atau *drowsy* beserta nilai

confidence [14] . Hasil deteksi ditampilkan pada antarmuka aplikasi, dan sistem memicu notifikasi otomatis ketika prediksi menunjukkan indikasi mata lelah (*drowsy*) sesuai ambang keputusan (*threshold*) yang ditetapkan. Dengan alur ini, aplikasi berfungsi sebagai peringatan dini agar pengguna dapat melakukan jeda atau mengurangi intensitas penggunaan layar.

Hasil implementasi menunjukkan bahwa model berhasil diintegrasikan ke aplikasi Android dan mampu melakukan deteksi kondisi mata secara langsung melalui *live camera*. Aplikasi dapat menampilkan keluaran deteksi secara *real-time* serta memberikan notifikasi ketika indikasi mata lelah terdeteksi sebagai tindakan preventif [15].

Evaluasi implementasi dilakukan untuk menilai kinerja dan kestabilan aplikasi pada penggunaan nyata. Aspek evaluasi meliputi: (1) kecepatan inferensi (FPS/latensi) untuk memastikan kinerja *real-time* saat kamera aktif; (2) ketahanan deteksi pada variasi kondisi seperti pencahayaan, jarak, dan sudut wajah; (3) penggunaan sumber daya perangkat (CPU dan RAM) selama aplikasi berjalan; dan (4) stabilitas aplikasi pada penggunaan jangka panjang. Selain pengujian *real-time* di perangkat, performa model juga dievaluasi pada dataset uji menggunakan metrik akurasi, *precision*, *recall*, dan *F1-score* sebagaimana dipaparkan pada bagian hasil pengujian model.

3.5. Desain Sistem

Sistem deteksi mata lelah *real-time* dibangun berbasis aplikasi Android dengan bahasa pemrograman Java (Android Studio). Komponen utama sistem terdiri dari:

1) *MainActivity.java* : mengatur antarmuka utama dan proses *real-time* detection menggunakan kamera.
2) *Yolov5TFLiteDetector.java* : menangani pemuatan model YOLOv5 dalam format *TensorFlow Lite*.
3) *Recognition.java* : mendefinisikan struktur hasil deteksi, termasuk label (*awake*/*drowsy*), tingkat kepercayaan (*confidence*), dan koordinat *bounding box*.
4) *DetectionEvent.java* : mencatat hasil deteksi setiap *frame*, termasuk waktu, status, dan frekuensi deteksi kondisi *drowsy*.

3.6. MainActivity.java (Pengendali Kamera dan Deteksi)

Kelas ini menjadi titik masuk utama aplikasi. Ia menginisialisasi kamera menggunakan *CameraX API*, kemudian menjalankan fungsi *runObjectDetection()* setiap kali *frame* baru diterima.

Berikut cuplikan fungsi utama:

```
private void runObjectDetection(ImageProxy imageProxy) {
    Bitmap bitmap = convertImageProxyToBitmap(imageProxy);
    int rotationDegrees = imageProxy.getImageInfo().getRotationDegrees();
    Bitmap rotatedBitmap = rotateAndMirrorBitmap(bitmap, rotationDegrees, isUsingFrontCamera);

    if (rotatedBitmap != null) {
        ArrayList<Recognition> recognitions = yolov5TFLiteDetector.detect(rotatedBitmap);
        lastRecognitions = new ArrayList<>(recognitions);

        updateBoundingBoxes(rotatedBitmap, recognitions);
        lastBoundingBoxUpdateTime = System.currentTimeMillis();
        clearBoundingBoxesAfterInterval();
    }
    imageProxy.close();
}
```

Gambar 8. Main activity

Pada Gambar 8 Main activity menjelaskan metode *runObjectDetection()* menerima *frame* kamera dalam bentuk *ImageProxy*, kemudian mengonversinya menjadi *Bitmap* melalui *convertImageProxyToBitmap(imageProxy).* Selanjutnya, sistem mengambil informasi rotasi dari kamera *(getRotationDegrees())* dan melakukan penyesuaian orientasi serta pembalikan gambar *(mirror)* untuk kamera depan melalui *rotateAndMirrorBitmap().* Jika bitmap hasil pra-pemrosesan *valid*, gambar tersebut diproses oleh model melalui *yolov5TFLiteDetector.detect(rotatedBitmap*) untuk menghasilkan daftar hasil deteksi *(recognitions),* yang kemudian disimpan ke *lastRecognitions* sebagai status deteksi terbaru. Hasil deteksi berikutnya digunakan pada *updateBoundingBoxes()* untuk memperbarui tampilan *bounding box* dan label pada *overlay*, serta waktu pembaruan dicatat pada *lastBoundingBoxUpdateTime* agar tampilan tetap sinkron. Mekanisme *clearBoundingBoxesAfterInterval()* berfungsi menghapus *bounding box* yang sudah tidak relevan ketika tidak ada pembaruan deteksi dalam interval tertentu. Setelah seluruh proses selesai, *imageProxy.close()* dipanggil untuk mencegah penumpukan *frame* dan menjaga stabilitas memori selama proses inferensi *real-time.*

3.7. Yolov5*TFLite*Detector.java (Pemanggilan Model)

Kelas ini berfungsi sebagai *model handler*, bertugas untuk memuat model hasil konversi (best.*TFLite*), melakukan *preprocessing* gambar (*resize, normalization, tensor*

conversion), menjalankan *inference* melalui *TensorFlow Lite Runtime*. Metode *postProcess()* menghasilkan daftar objek *Recognition* berisi koordinat *bounding box*, label, dan tingkat kepercayaan.

```
public List<Recognition> detect(Bitmap bitmap) {
   // Convert bitmap to input tensor
   Tensor inputTensor = preprocess(bitmap);
   OrtSession.Result output = session.run(Collections.singletonMap("images", inputTensor));
   return postProcess(output);
}
```

3.8. Recognition.java (Struktur Data Hasil Deteksi)

Kelas ini digunakan sebagai *data model* untuk menyimpan setiap hasil deteksi. Setiap objek *Recognition* berisi atribut utama dan struktur ini memastikan setiap prediksi dapat divisualisasikan secara akurat di antarmuka pengguna dengan label dan *confidence score* masing-masing.

```
public class Recognition {
   private String title;
   private Float confidence;
   private RectF location;
}
```

3.9. DetectionEvent.java (Pencatatan dan *Log* Aktivitas)

Kelas ini berfungsi untuk mencatat setiap hasil deteksi dalam jangka waktu tertentu. *Log* digunakan untuk menentukan apakah kondisi *drowsy* berlangsung cukup lama untuk memicu notifikasi. Dengan pendekatan ini, sistem tidak hanya mengenali kondisi mata lelah sesaat, tetapi juga mempertimbangkan durasi dan frekuensi kemunculan label "*Drowsy*", sehingga hasil lebih akurat dan kontekstual.

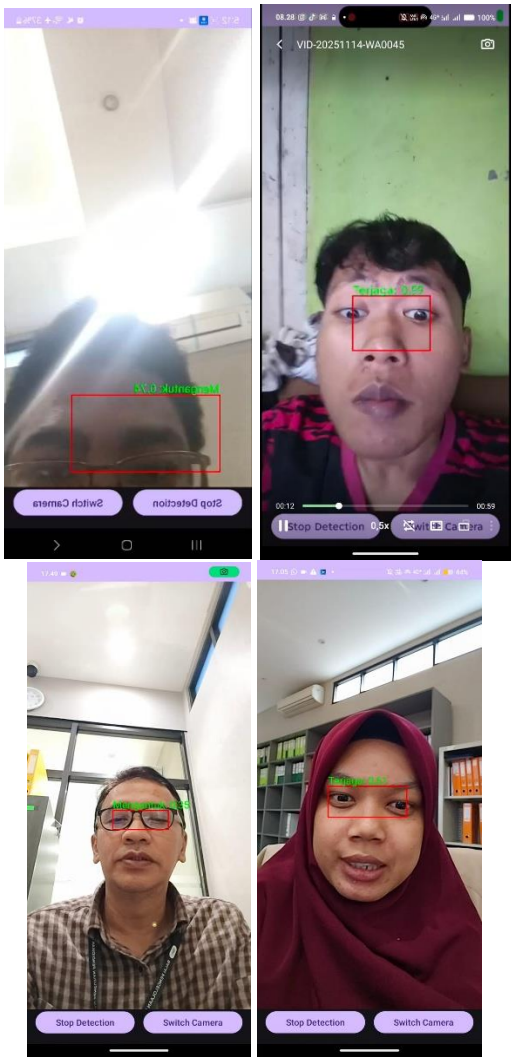
```
public void addDetection(String label, float confidence) {
   events.add(new Event(System.currentTimeMillis(), label, confidence));
   if (checkDrowsyFrequency() > 5) {
      triggerWarning();
   }}
```

3.10. Tampilan Aplikasi

Aplikasi deteksi mata lelah berbasis YOLOv5–*TensorFlow Lite* dikembangkan dengan antarmuka sederhana dan intuitif agar mudah digunakan oleh pengguna umum. Antarmuka utama terdiri atas tiga komponen tampilan kamera untuk proses deteksi, tombol pengendali (*Start/Stop Detection* dan *Switch Camera*), serta tampilan hasil deteksi dalam bentuk *bounding box* dan label prediksi.

1) Antarmuka Utama (*Main View*)

   Tampilan utama aplikasi menampilkan citra pengguna secara *real-time* melalui kamera depan.Ketika tombol *Start Detection* ditekan, sistem mulai melakukan inferensi menggunakan model *TFLite*. Hasil deteksi ditampilkan dalam bentuk persegi panjang (*bounding box*) berwarna merah, disertai label dan tingkat kepercayaan (*confidence score*) di atasnya.



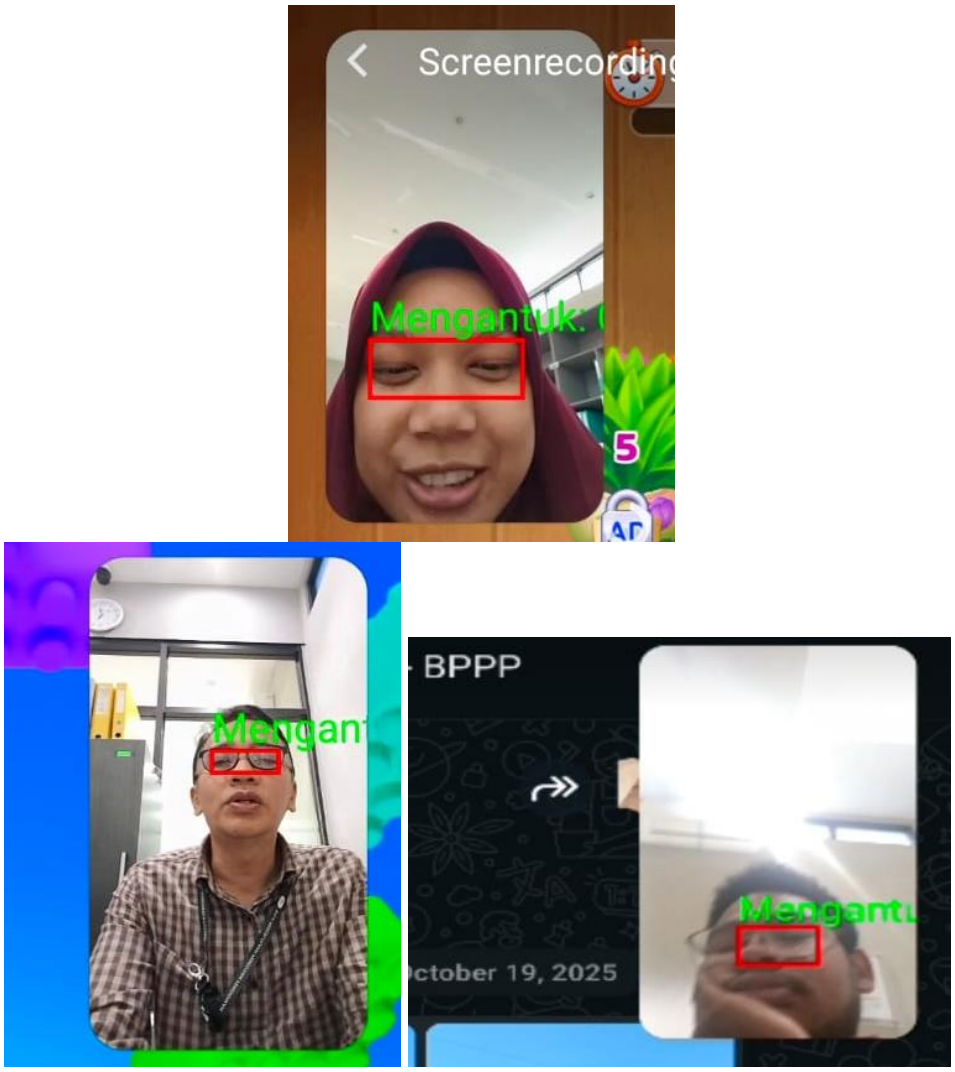
Gambar 9. Antarmuka utama aplikasi saat deteksi aktif

Pada Gambar 9 diatas menunjukkan bahwa sistem mengenali kondisi mata pengguna dalam keadaan terjaga (*awake*). Warna merah pada *bounding box* dipilih untuk memastikan kontras tinggi dan visibilitas yang baik di berbagai kondisi pencahayaan. Tombol *Switch Camera* memungkinkan pengguna beralih antara kamera depan dan belakang, sedangkan tombol *Stop Detection* menghentikan proses inferensi. Penggunaan dua tombol ini membantu pengguna mengontrol deteksi dengan mudah tanpa menutup aplikasi.

2) Mode *Picture-in-Picture (PiP)*

Karena sistem tidak dapat berjalan sepenuhnya di latar belakang akibat kebijakan keamanan Android, fitur *Picture-in-Picture (PiP*) diterapkan agar

proses deteksi tetap aktif ketika pengguna berpindah ke aplikasi lain. Ketika tombol *Home* ditekan atau pengguna membuka aplikasi lain, *MainActivity* otomatis beralih ke mode *PiP*. Dalam mode ini kamera tetap aktif dan menjalankan proses inferensi. tampilan diperkecil menjadi jendela melayang di pojok layar, *label* dan *confidence score* tetap tampil pada *overlay* kecil di jendela *PiP*.



Gambar 10. *Mode Picture-in-Picture (PiP) saat deteksi tetap berjalan di layar beranda*

Pada Gambar 10 ini memperlihatkan hasil deteksi, menunjukkan bahwa proses deteksi tetap berlangsung walaupun pengguna berada di layar utama perangkat. Implementasi *PiP* dilakukan dengan menambahkan konfigurasi berikut pada file *AndroidManifest.xml* dan dan metode berikut di dalam *MainActivity.java:*

file *AndroidManifest.xml*

```
<activity
   android:name=".MainActivity"
   android:supportsPictureInPicture="true"
   android:resizeableActivity="true"
      android:configChanges="screenSize|smallestScreenSize|screenLayou
 t|orientation" />
```

MainActivity.java

```
@Override
public void onUserLeaveHint() {
   super.onUserLeaveHint();
   enterPictureInPictureMode(new PictureInPictureParams.Builder()
      .setAspectRatio(new Rational(16,9)).build());
}
```

Dengan pendekatan ini, aplikasi dapat tetap memantau kondisi mata pengguna meskipun tidak dalam tampilan penuh, selama sesi kamera belum dihentikan.

Hasil pengujian menunjukkan mode PiP mampu mempertahankan kecepatan rata-rata 20–22 FPS, tanpa *frame drop* yang signifikan.

Hasil penelitian menunjukkan bahwa kombinasi YOLOv5 dan *TensorFlow Lite* memberikan keseimbangan optimal antara akurasi dan efisiensi dalam proses deteksi citra secara *real-time*. Model yang dikembangkan mencapai akurasi sebesar 95,6%, dengan nilai *precision* sebesar 0,98 dan *recall* sebesar 0,96, serta kecepatan deteksi rata-rata 22 frame per second (FPS). Hal ini membuktikan bahwa sistem mampu mengenali kondisi mata secara cepat dan akurat.

Jika dibandingkan dengan metode *Convolutional Neural Network (CNN)* konvensional yang umumnya hanya mencapai tingkat akurasi 88–90% dengan kecepatan 10–15 FPS, sistem berbasis YOLOv5–*TFLite* ini menunjukkan peningkatan signifikan baik dari segi kinerja maupun efisiensi pemrosesan. Selain itu, keunggulan utama dari sistem ini adalah kemampuannya untuk

beroperasi secara *offline*, tanpa memerlukan koneksi internet, sehingga lebih hemat sumber daya dan aman terhadap privasi pengguna.

Implementasi sistem pada perangkat Android, sebagaimana dijelaskan pada bagian 3.4, menunjukkan bahwa model mampu berjalan dengan stabil menggunakan CameraX API dan *TensorFlow Lite* Interpreter. Dengan demikian, hasil pelatihan dan penerapan membuktikan bahwa sistem deteksi mata lelah berbasis YOLOv5–*TFLite* ini efektif dalam mendeteksi *digital eye strain* secara cepat, ringan, dan akurat di perangkat bergerak. Hasil tersebut juga menunjukkan potensi besar pengembangan sistem berbasis *edge computing* untuk mendukung penerapan kecerdasan buatan dalam bidang kesehatan digital.

4. KESIMPULAN

Penelitian ini menyimpulkan bahwa YOLOv5 layak diimplementasikan pada perangkat Android untuk mendeteksi indikasi mata lelah melalui kamera depan secara *real-time* dengan dua kelas keluaran (*awake*/terjaga dan *drowsy*/mengantuk). Model menunjukkan performa deteksi yang baik pada data uji, ditunjukkan oleh *mAP*@0,5 = 0,920, *mAP*@0,5:0,95 = 0,410, *precision* = 1,000, *recall* = 0,980, dan *F1-score* = 0,880. Hasil ini mengindikasikan kemampuan model dalam membedakan kedua kelas secara konsisten pada berbagai ambang evaluasi.

Pada implementasi aplikasi, sistem mampu menjalankan inferensi secara *offline* dan menampilkan hasil deteksi (*label*, *confidence*, serta visualisasi *bounding box*) melalui pipeline kamera. Kinerja *real-time* pada perangkat Android kelas menengah mencapai akurasi 95,6%, *precision* 94,3%, *recall* 96,1%, dengan kecepatan rata-rata 22 FPS, sehingga cukup responsif untuk pemantauan langsung. Selain itu, penerapan logika notifikasi berbasis kemunculan kondisi *drowsy* dalam durasi tertentu membantu mengurangi peringatan akibat deteksi sesaat, sehingga meningkatkan kenyamanan penggunaan sebagai sistem peringatan dini.

Manfaat utama dari penelitian ini adalah menyediakan solusi praktis untuk membantu pengguna memantau indikasi mata lelah dan mendorong perilaku penggunaan layar yang lebih sehat. Keterbatasan yang masih perlu diperhatikan meliputi variasi

pencahayaan, sudut wajah, serta penggunaan kacamata yang dapat memengaruhi stabilitas deteksi pada kondisi nyata. Penelitian selanjutnya disarankan melakukan optimasi parameter inferensi (misalnya *threshold* dan penyaringan hasil), peningkatan praproses kamera, perluasan variasi data, serta pengujian pada lebih banyak tipe perangkat agar performa lebih stabil dan generalisasi sistem semakin baik.

5. DAFTAR PUSTAKA

[1] ratna sari dinaryanti maulida awalia, "HUBUNGAN LAMA PENGGUNAAN GADGET DENGAN STIKES PERTAMEDIKA Oleh : Ketua : Maulida Awalia (NIM : 11181027) Anggota : Ratna Sari Dinaryanti (NIDN : 0630018101) SEKOLAH TINGGI ILMU KESEHATAN PERTAMEDIKA," *HUBUNGAN LAMA PENGGUNAAN GADGET DENGAN KELELAHAN MATA PADA MAHASISWA*, p. 93, 2022.

[2] T. Yurika, N. Nurjannah, S. Basri, S. Ishak, and S. Hajar, "Pengaruh penggunaan gadget dengan kejadian mata lelah pada siswa SMA selama masa pandemi COVID-19," *Jurnal Kedokteran Syiah Kuala*, vol. 22, no. 2, pp. 1412–1026, 2022, doi: 10.24815/jks.v22i2.22637.

[3] Anggi Saputra, Maidartati, and Umi Khasanah, "LINDUNGI MATAMU DARI LAYAR: BIJAK PAKAI GADGET, SEHATKAN PENGLIHATAN," 2025.

[4] D. Peng, J. Cai, L. Zheng, M. Li, L. Nie, and Z. Li, "A Novel Neural Network Model Based on Real Mountain Road Data for Driver Fatigue Detection," *Biomimetics*, vol. 10, no. 2, Feb. 2025, doi: 10.3390/biomimetics10020104.

[5] M. Arava and D. M. Sundaram, "Integrating lightweight YOLOv5s and facial 3D keypoints for enhanced fatigued-driving detection," *PeerJ Comput. Sci.*, vol. 10, pp. 1–27, 2024, doi: 10.7717/peerj-cs.2447.

[6] D. Avola *et al.*, "MS-Faster R-CNN : Multi-Stream Backbone for Improved Faster R-CNN Object Detection and Aerial Tracking from UAV Images," pp. 1–18, 2021.

[7] I. Wayan, A. A. Wiguna, R. R. Huizen, G. Angga Pradipta, and M. Program, "Optimization of Vehicle Detection at Intersections Using the YOLOv5 Model," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, vol. 10, no. 4, pp. 885–896, 2024, doi: 10.26555/jiteki.v10i4.29309.

[8] A. D. Nugroho and W. M. Baihaqi, "Improved YOLOv5 with Backbone Replacement to MobileNet V3s for School Attribute Detection," vol. 8, no. 3, pp. 1944–1954, 2023.

[9] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "*Real-time* Vehicle Detection Based on Improved YOLO v5," *Sustainability (Switzerland)*, vol. 14, no. 19, Oct. 2022, doi: 10.3390/su141912274.

[10] SONA SONDARA SAESARIA, "PENDETEKSIAN OBJEK PADA LALU LINTAS BERAGAM DI INDONESIA UNTUK KENDARAAN OTONOM BERBASIS ALGORITME YOLOV5," Jul. 2024.

[11] Z. Tang, M. M. Hasan, and T. Strauss, "Optimized YOLOv5 model for safety helmet and flame detection system," *Signal Image Video Process.*, vol. 19, no. 8, Aug. 2025, doi: 10.1007/s11760-025-04073-z.

[12] H. Lin, J. D. Deng, D. Albers, and F. W. Siebert, "Helmet Use Detection of Tracked Motorcycles Using CNN-Based Multi-Task Learning," *IEEE Access*, vol. 8, pp. 162073–162084, 2020, doi: 10.1109/ACCESS.2020.3021357.

[13] M. F. Ridho, F. Panca, W. Yandi, and A. A. Rachmani, "Electron : Jurnal Ilmiah Teknik Elektro *Drowsiness Detection* in the Advanced Driver-Assistance System using YOLO V5 Detection Model," vol. 5, 2024.

[14] A. Susilo and F. Adryansyah, "Optimalisasi Algoritma YOLOv5 untuk Deteksi Mata Katarak," *Journal TIFDA (Technology Information and Data Analytic)*, vol. 1, no. 2, pp. 63–68, Dec. 2024, doi: 10.70491/tifda.v1i2.55.

[15] M. Zainuddin and M. S. Zuhri, "Implementasi algoritma YOLOv5 pada platform Android untuk penghitungan bibit ikan lele (clarias sp.)," *Jurnal Ilmiah Teknologi Informasi Asia*, vol. 19, no. 2, pp. 114–121, Sep. 2025, doi: 10.32815/jitika.v19i2.1184.