

**DETEKSI KECURANGAN PADA CBT ONLINE  
MENGUNAKAN YOLOV8**

**SKRIPSI**

YERICHO NATANAEL  
201402092



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

**DETEKSI KECURANGAN PADA CBT ONLINE  
MENGUNAKAN YOLOV8**

**SKRIPSI**

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

YERICHO NATANAEL  
201402092



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

## PERSETUJUAN

Judul : Deteksi Kecurangan Pada CBT Online Menggunakan YOLOv8

Kategori : Skripsi

Nama Mahasiswa : Yericho Natanael

Nomor Induk Mahasiswa : 201402092

Program Studi : Sarjana (S-1) Teknologi Informasi

Fakultas : Ilmu Komputer dan Teknologi Informasi  
Universitas Sumatera Utara

Medan, 16 Oktober 2024

Komisi Pembimbing:

Pembimbing 2,



Umayya Ramadhani Putri Nasution S.TI., M.Kom.  
NIP. 199104112021022001

Pembimbing 1,

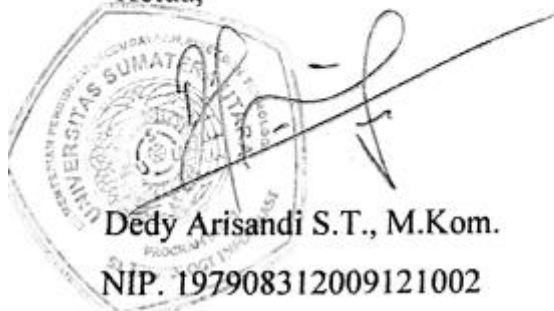


Dedy Arisandi, S.Kom, M.Kom  
NIP. 197908312009121002

Diketahui/disetujui oleh

Program Studi S-1 Teknologi Informasi

Ketua,



Dedy Arisandi S.T., M.Kom.  
NIP. 197908312009121002

**PERNYATAAN**

**DETEKSI KECURANGAN PADA CBT ONLINE  
MENGUNAKAN  
YOLOv8**

**SKRIPSI**

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

Medan, 16 Oktober 2024

A handwritten signature in black ink, appearing to read 'Yericho Natanael', with a horizontal line underneath.

Yericho Natanael

NIM. 201402092

## DETEKSI KECURANGAN PADA CBT ONLINE MENGGUNAKAN YOLOV8

### ABSTRAK

Kemajuan teknologi saat ini mengubah metode ujian tulis saat ini. Ujian tulis saat ini dilakukan dengan menggunakan komputer atau biasanya dikenal dengan istilah *Computer Based Test* (CBT). Ujian CBT ini biasanya dilakukan dengan menggunakan *E-Learning* dan harus terhubung dengan jaringan internet. CBT *online* memberikan kebebasan para peserta mengerjakan ujian di mana pun dengan waktu yang telah ditentukan. CBT *online* saat ini tidak memiliki pengawasan yang baik untuk setiap peserta. Oleh karena itu, diperlukan sistem yang dapat mendeteksi kecurangan para peserta. Penelitian ini mengimplementasikan algoritma *You Only Look Once* (YOLO) versi 8 untuk mendeteksi tiga jenis objek kecurangan pada ujian *online*: *phone*, OrangLain dan *book* dengan jumlah data yang digunakan sebanyak 4119 data gambar dan 4815 anotasi yang dilakukan. Anotasi terdiri dari 1882 data sebagai data *phone*, 1563 data OrangLain dan 1370 data *book*. Hasil menunjukkan bahwa algoritma YOLOv8 mampu mendeteksi objek kecurangan secara *real-time* dengan nilai *accuracy* 97%, *precision* 97.5%, *recall* 98.7%, dan F1-Score 98.1%. Model ini juga mampu memprediksi lebih dari satu objek kecurangan dalam satu frame dengan catatan objek tersebut berada dalam jarak optimal di depan kamera. Dari hasil tersebut menunjukkan bahwa sistem yang dibuat menggunakan algoritma YOLO V-8 telah berhasil dengan baik dalam mendeteksi objek kecurangan pada CBT *online*.

**Kata Kunci:** Kecurangan, CBT *Online*, YOLOv8, Deteksi Objek, *Real-time*, Citra Digital

## **FRAUD DETECTION IN ONLINE CBT USING YOLOV8**

### **ABSTRACT**

*The current written exam is being administered differently due to advancements in technology. Computers are being used to administer written exams, also referred to as computer-based tests (CBT). E-learning is typically used for CBT exams, and an internet connection is required. Participants in online CBT have the flexibility to take the test at any time and from any location. Currently, there is inadequate supervision for each participant in online CBT. As a result, a system that can identify participant dishonesty is required. This research implemented the You Only Look Once (YOLO) version 8 algorithm to detect three types of fraudulent objects in online tests: phone, Other people and book with the amount of data used as much as 4119 image data and 4815 annotations performed. The annotation consists of 1882 data as data phone, 1563 data from other people and 1370 data books. According to the findings, the YOLOv8 algorithm has a 97% accuracy rate, 97.5% precision rate, 98.7% recall rate, and 98.1% F1-Score when it comes to real-time cheating object detection. As long as the object is in front of the camera at the ideal distance, this model may also identify many cheating objects in a single frame. These findings demonstrate how effectively the YOLO V-8 algorithm-based system has performed in identifying deceptive objects during online cognitive behavioral therapy.*

*Keywords: Cheating, CBT Online, YOLOv8, Object Detection, Real-time, Digital Imagery*

## UCAPAN TERIMA KASIH

Puji dan Syukur kepada Tuhan Yang Maha Esa, yang senantiasa memberikan berkat, rahmat, dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Dalam proses menyelesaikan penulisan tugas akhir ini, penulis telah menerima banyak bimbingan, dukungan, bantuan, serta doa dari berbagai pihak. Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada:

1. Keluarga penulis, Papa F Sitanggang dan Mama R A P selaku orang tua penulis yang telah mendidik dan membesarkan penulis, selalu memberikan semangat dan doa terbaik untuk penulis, begitu juga dengan adik penulis Tadeus Vito, Matthew Hazel dan Darrel Ignazio yang senantiasa memberikan doa dan dukungan dalam setiap kondisi.
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 Universitas Sumatera Utara.
4. Bapak Dedy Arisandi ST., M.Kom. selaku Dosen Pembimbing 1 dan Ketua Program Studi S1 Teknologi Informasi yang telah banyak membimbing dan mengarahkan, serta memberikan saran pada penulis dalam proses pengerjaan skripsi ini.
5. Ibu Umayra Ramadhani Putri Nasution S.TI., M.Kom. selaku Dosen Pembimbing 2 yang juga sangat banyak membantu penulis dalam membimbing serta memberi saran dan arahan dalam proses pengerjaan skripsi.
6. Seluruh Dosen Program Studi S1 Teknologi Informasi Sumatera Utara yang telah mengajar dan memberikan ilmu yang berguna selama masa perkuliahan penulis.
7. Seluruh *Staff* dan Pegawai Fasilkom-TI Universitas Sumatera Utara yang telah membantu segala urusan administrasi selama masa perkuliahan dan menyelesaikan skripsi penulis.

8. Putri Patricia M yang telah membantu dan menemani penulis dari awal perkuliahan hingga akhir pengerjaan tugas akhir.
9. Teman – teman seperjuangan angkatan 20, Jethro Sihotang, Felix Rumahorbo, M. Zidan Lubis, Azriel Simanjuntak, Albert Pangaribuan, Wahyu Marpaung, Yefta Pardede, Pretty, Ullaya, Grace, Irwansyah, Fildzah, Syavira, Rio Siregar dan masih banyak lagi yang tidak bisa penulis sebutkan satu persatu.
10. Bang Daniel Situmeang, Bang Christoper, Bang Geylfedra, dan Par-Andimed yang tidak bisa penulis sebutkan satu persatu.
11. Kepada Senior, Junior, Kerabat, teman-teman dan pihak lainnya yang juga berperan dalam memberikan semangat dan dukungan serta doa selama masa perkuliahan dan dalam menyelesaikan penulisan skripsi.

Medan, 16 Oktober 2024

Penulis



Yericho Natanael  
201402092



## DAFTAR ISI

<b>PERSETUJUAN</b>	iii
<b>PERNYATAAN</b>	iv
<b>ABSTRAK</b>	v
<b><i>ABSTRACT</i></b>	vi
<b>UCAPAN TERIMA KASIH</b>	vii
<b>DAFTAR ISI</b>	ix
<b>DAFTAR TABEL</b>	xi
<b>DAFTAR GAMBAR</b>	xii
<b>BAB 1 PENDAHULUAN</b>	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	5
<b>BAB 2 LANDASAN TEORI</b>	6
2.1 Kecurangan Akademik	6
2.2 Citra <i>Grayscale</i>	7
2.3 <i>Convolutional Neural Network</i>	7
2.4 Algoritma <i>You Only Look Once</i> (YOLO)	8
2.5 YOLOv8	9
2.6 <i>Mediapipe</i>	11
2.7 <i>OpenCV</i>	11
2.8 <i>Confusion Matrix</i>	12
2.9 Penelitian Terdahulu	13
<b>BAB 3 ANALISIS DAN PERANCANGAN SISTEM</b>	18
3.1 Dataset	18
3.2 Analisis Sistem	19
3.3 Perancangan Antarmuka Sistem	29
3.3.1 Tampilan Awal	29

3.3.2 Tampilan Beranda	30
3.3.3 Tampilan <i>Register Face</i>	30
3.3.4 Tampilan <i>Face Recognition</i>	31
3.3.5 Tampilan Halaman Ujian	32
3.3.6 Tampilan <i>Profile</i>	32
3.3.7 Tampilan <i>Live Proctor</i>	33
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM</b>	34
4.1 Implementasi Sistem	34
4.2 Implementasi Data	35
4.2.1 Data Objek Kecurangan	35
4.2.2 Data Wajah	35
4.3 Implementasi Rancangan Antarmuka	36
4.3.1 Tampilan Awal	36
4.3.2 Tampilan Beranda	37
4.3.3 Tampilan <i>Register Face</i>	37
4.3.4 Tampilan <i>Face Recognition</i>	38
4.3.5 Tampilan Halaman Ujian	39
4.3.6 Tampilan Profil	39
4.3.7 Tampilan <i>Live Proctor</i>	39
4.3 Implementasi API	40
4.4 Pelatihan Sistem	41
4.5 Pengujian Sistem	47
4.5.1 Pengujian Sistem <i>Face Recognition</i>	47
4.5.2 Pengujian Sistem <i>Object-Detection</i>	49
4.5.3 Pengujian Sistem <i>Face Landmarks</i>	55
<b>BAB 5 KESIMPULAN</b>	57
5.1 Kesimpulan	57
5.2 Saran	57
<b>DAFTAR PUSTAKA</b>	58

**DAFTAR TABEL**

Tabel 2.1 Penelitian Terdahulu	16
Tabel 4.1 Hasil Pelatihan Sistem	41
Tabel 4.2 Pengujian Pencahayaan dan Jarak Face Recognition	49
Tabel 4.3 Pengujian Sistem YOLOv8	49
Tabel 4.4 Confusion Matrix	53
Tabel 4.5 Nilai Precision, Recall dan F1-Score	54
Tabel 4.6 Penggunaan CPU & GPU	56

## DAFTAR GAMBAR

Gambar 2.1 Arsitektur Umum <i>Convolutional Neural Network</i>	8
Gambar 2.2 Arsitektur YOLO	9
Gambar 2.3 Arsitektur YOLOv8	10
Gambar 2.4 Face Landmark Mediapie	11
Gambar 2.5 Confusion Matrix	12
Gambar 3.1 Image Datasets	18
Gambar 3.2 Alur Kerja Sistem	19
Gambar 3.3 Arsitektur Umum	20
Gambar 3.4 Arsitektur Proses Face Recognition	21
Gambar 3.5 Pseudocode resize	22
Gambar 3.6 Pseudocode grayscalling	22
Gambar 3.7 Pseudocode histogram eqlization	23
Gambar 3.8 Pseudocode menghitung LBP	23
Gambar 3.9 Pseudocode membagi LBP menjadi beberapa grid	24
Gambar 3.10 Arsitektur Pemrosesan Deteksi Objek YOLO	24
Gambar 3.11 Data yang di-labelling	25
Gambar 3.12 Penerapan <i>Auto-orient</i>	26
Gambar 3.13 Arsitektur Proses Face Landmark	28
Gambar 3.14 Pseudocode MediaPipe	29
Gambar 3.15 Tampilan Login	29
Gambar 3.16 Tampilan Beranda	30
Gambar 3.17 Tampilan <i>Register</i> Wajah	31
Gambar 3.18 Tampilan Verifikasi Wajah	31
Gambar 3.19 Tampilan Halaman Ujian	32
Gambar 3.20 Tampilan <i>Profile</i>	32
Gambar 3.21 Tampilan Live Proctor	33
Gambar 4.1 Data Objek Kecurangan	35
Gambar 4.2 Data Wajah	36
Gambar 4.3 Tampilan Awal	37
Gambar 4.4 Tampilan Beranda	37
Gambar 4.5 Tampilan Register Face	38
Gambar 4.6 Tampilan face recognition	38
Gambar 4.7 Tampilan Halaman Ujian	39
Gambar 4.8 Tampilan Profile	39
Gambar 4.9 Tampilan Live Proctor	40
Gambar 4.10 API	40
Gambar 4.11 Kaggle Notebook	41
Gambar 4.12 Grafik Box Loss	44
Gambar 4.13 Grafik Cls Loss	45
Gambar 4.14 Grafik DFL	45
Gambar 4.15 Grafik Precision	46
Gambar 4.16 Grafik Recall	46

Gambar 4.17 Grafik mAP50	47
Gambar 4.18 Grafik mAP50-95	47
Gambar 4.19 Hasil Face Recognition	48
Gambar 4.20 Dataset Wajah	48
Gambar 4.21 Face Landmark Mediapipe	55

# **BAB 1**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Menurut KBBI ujian adalah percobaan untuk mengetahui mutu sesuatu (ketulenan, kecakapan, ketahanan, dan sebagainya). Ujian biasanya dilaksanakan ketika sudah menyelesaikan sebuah pembelajaran ataupun untuk masuk ke jenjang berikutnya yang lebih tinggi dari sebelumnya. Hasil dari ujian nanti yang menjadi sebuah tolak ukur seseorang untuk lolos atau tidak ke jenjang berikutnya. Ujian memiliki berbagai jenis, salah satu ujian yang paling banyak digunakan adalah ujian tulis. Metode yang dilakukan ujian tulis biasanya memerlukan kertas dan alat tulis untuk mengerjakannya.

Kemajuan teknologi saat ini mengubah metode saat ujian tulis saat ini. Ujian tulis saat ini dilakukan dengan menggunakan komputer atau biasanya dikenal dengan istilah *Computer Based Test* (CBT). Ujian CBT ini biasanya dilakukan dengan menggunakan *E-Learning* dan harus terhubung dengan jaringan internet. CBT *online* memberikan kebebasan para peserta mengerjakan ujian di mana pun dengan waktu yang telah ditentukan. CBT *online* saat ini tidak memiliki pengawasan yang baik untuk setiap peserta. Pengawasan yang dilakukan hanya merekam setiap aktivitas dan akan melakukan *random checking* serta deteksi *screenshot*.

Semenjak pandemik COVID-19 banyak aktivitas dalam berbagai bidang mengalami perubahan yang sangat signifikan. Perubahan ini diakibatkan oleh pembatasan sosial berskala besar di seluruh dunia, termasuk Indonesia. Salah satu bidang yang terkena dampak yaitu bidang pendidikan. Dalam bidang pendidikan yang sebelumnya menggunakan metode pengajaran secara tatap muka telah beralih metode

*online* begitu juga dengan penilaian telah beralih dari tatap muka menjadi *online*. Hal ini membuat tindakan curang dalam pelaksanaan ujian semakin memburuk.

Pengawasan yang dilakukan kurang efektif dan menimbulkan banyak pertanyaan bagi peserta yang mengikuti ujian. Selain itu faktor keamanan dari peserta juga kurang kuat karena hanya memasukkan *username* dan *password*. Hal ini yang menyebabkan banyaknya kecurigaan dalam melaksanakan ujian CBT *online* dan sulit menentukan sebuah hasil yang baik untuk peserta dan panitia pelaksanaan ujian.

Untuk itu diperlukan sebuah sistem yang mampu memberikan keamanan dan pengawasan yang berbasis AI. Keamanan biometrik diperlukan agar terhindar dari joki yang biasanya dibayar untuk mengerjakan sebuah CBT *online* peserta. Selain itu, pengawasan berbasis AI yang mempermudah kerja panitia dalam menemukan kecurangan yang dilakukan oleh peserta ujian. Kecurangan yang biasanya dilakukan dalam bentuk melihat buku, *handphone* dan meminta bantuan orang lain.(Noorbebhahani *et al.*, 2022) Sistem ini dapat memperkecil kemungkinan peserta dalam melakukan kecurangan sekalipun ujian dilakukan secara *online*.

Pada beberapa penelitian terdahulu masih banyak ditemukan penggunaan dari algoritma YOLOv3 sebagai pendeteksi objek dan juga Dlib sebagai *face landmark* untuk menentukan *head, eye and mouth* pose dari para peserta CBT *online*. Sebagai solusi untuk menyelesaikan permasalahan pengawasan pada saat ujian *online* dengan metode yang terbaru dan dilakukan secara *real time*. Penelitian ini juga akan menggunakan Mediapipe sebagai pengganti Dlib dalam mendeteksi pergerakan kepala, mata dan mulut. Mediapipe memiliki kemampuan lebih baik dengan menggunakan 468 *key points* untuk mendeteksi bagian wajah sedangkan Dlib hanya menggunakan 68 *key points*.

Penelitian ini menggunakan YOLOv8 yang telah mengalami peningkatan dalam *backbone* dan peningkatan performa sebesar 33.21% dari YOLOv5 untuk YOLOv8n dalam akurasi deteksi objek. Hal ini menjadi alasan penulis dalam memilih metode YOLOv8 dalam mendeteksi objek kecurangan.

## 1.2 Rumusan Masalah

Permasalahan saat ini yang sering dihadapi pada saat melakukan ujian CBT *online* adalah kesulitan dalam mengontrol pergerakan yang mencurigakan dari setiap peserta

ujian. Hal ini disebabkan karena masih menggunakan media *meeting online* seperti *zoom* untuk mengawasi peserta dan menggunakan rekaman video peserta sebagai bahan analisis untuk menemukan ada atau tidaknya kecurangan yang dilakukan oleh peserta. Selain itu masalah yang terjadi adalah waktu menentukan adanya kecurangan yang sangat lama untuk diproses. Dibutuhkannya pemrosesan dari deteksi kecurangan yang berlangsung secara *real-time* dengan akurasi yang baik. Hal ini dapat membantu pengawas untuk mengawasi setiap peserta yang ingin melakukan kecurangan dengan cepat dan tepat.

### 1.3 Batasan Masalah

Agar penelitian ini tetap di arah yang benar, maka diperlukan batasan masalah. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Objek kecurangan berupa buku, *handphone* dan *person*.
2. Aplikasi mendeteksi kecurangan dengan *webcam* laptop.
3. Aplikasi mendeteksi pergerakan kepala hanya depan, kanan, kiri, atas dan bawah.
4. Aplikasi akan mengirimkan gambar peserta dari *website* ujian ke server setiap 3s.
5. Deteksi pergerakan kepala peserta dilakukan secara *real time* pada *live proctor*.

### 1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk mengimplementasikan metode *You Only Look Once* (YOLO), *LBPH* dan *Mediapipe* dalam pembuatan mesin pendeteksi tindakan kecurangan pada saat *Computer Based Test* (CBT) berbasis *online*.

### 1.5 Manfaat Penelitian

Manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut :

1. Dengan penelitian ini, dapat membuktikan bahwa komputer dapat membantu para pengawas ujian *online* dalam menentukan kecurangan
2. Dapat membuat laporan log dari setiap aktivitas kecurangan tiap peserta dan dapat melihat secara langsung kondisi peserta



## 1.6 Metodologi Penelitian

Tahapan- tahapan yang akan dilakukan pada penelitian ini adalah sebagai berikut :

1. Studi Literatur

Pada tahap ini dilakukan studi literatur dengan mengumpulkan bahan referensi dari jurnal, artikel, *website*, dan sumber lainnya mengenai tindakan curang, *face recognition*, *image processing*, *face landmark*, *You Only Look Once*, *LBPH* dan *Mediapipe*.

2. Analisis Permasalahan

Pada tahapan ini penulis melakukan analisis terhadap tahapan yang telah dilakukan sebelumnya yaitu studi literatur untuk mendapatkan pemahaman mengenai *metode face recognition*, *face landmark* dan *image processing* yang akan diimplementasikan dalam penelitian untuk membangun mesin deteksi objek kecurangan.

3. Perancangan Sistem

Pada tahap ini dilakukan proses perancangan sistem berdasarkan analisis permasalahan yang telah dilakukan sebelumnya. Perancangan sistem meliputi perancangan arsitektur, pengumpulan data, dan pembagian data *training* dan data *testing*.

4. Implementasi

Pada tahap ini dilakukan implementasi ke dalam bentuk *source code* berdasarkan analisis dan rancangan sistem untuk menghasilkan sistem mesin pendeteksi kecurangan.

5. Pengujian Sistem

Pada tahap ini dilakukan pengujian terhadap sistem yang telah dibangun untuk mengetahui tingkat akurasi yang didapatkan dari implementasi deteksi kecurangan menggunakan YOLO, *face recognition* menggunakan *LBPH* dan *face landmark* menggunakan *Mediapipe* dalam pengembangan mesin pendeteksi kecurangan.

6. Dokumentasi dan Penyusunan Laporan

Pada tahap ini dilakukan dokumentasi dan penyusunan laporan yang menjabarkan hasil dari penelitian yang telah dilakukan.

## **1.7 Sistematika Penulisan**

Sistem penulisan skripsi ini terdiri dari lima bagian, yaitu :

### **BAB 1 : Pendahuluan**

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

### **BAB 2 : Landasan Teori**

Bab ini berisi tentang penjelasan dari teori-teori yang relevan dengan sistem dan permasalahan yang diangkat dalam penelitian ini.

### **BAB 3 : Analisis dan Perancangan Sistem**

Bab ini berisi tentang arsitektur umum dan metode yang diterapkan dalam pembuatan sistem.

### **BAB 4 : Implementasi dan Pengujian**

Bab ini berisi tentang implementasi atau penerapan sistem berdasarkan analisis dan rancangan yang telah dibuat sebelumnya. Bab ini juga berisi tentang tampilan sistem yang telah dibangun dan pengujian untuk mengetahui akurasi sistem dalam mendeteksi kecurangan, *face recognition* dan pergerakan kepala, mata dan mulut.

### **BAB 5 : Kesimpulan dan Saran**

Bab ini berisi tentang kesimpulan dari hasil penelitian dan saran untuk penelitian-penelitian selanjutnya.

## BAB 2

### LANDASAN TEORI

#### 2.1 Kecurangan Akademik

Kecurangan akademik adalah hal yang sangat tidak dibenarkan untuk mendapatkan hasil prestasi yang lebih baik. Kecurangan akademik merupakan fenomena pendidikan secara global yang terjadi hampir di setiap instansi pendidikan di dunia. Kecurangan akademik sering terjadi dalam proses pembelajaran dan proses penilaian bahkan sampai pada proses penyusunan tugas akhir (Nursalam, 2016). Salah satu bentuk kecurangan akademik adalah menyontek. Kegiatan menyontek menjadi hal yang sering dilakukan pada saat proses penilaian. Faktor penyebab dari kegiatan menyontek adalah adanya tekanan untuk mendapatkan nilai yang bagus dan memperoleh peringkat yang baik.

Kegiatan menyontek pada saat ujian *online* sering terjadi karena kurangnya efektivitas pengawasan ujian *online* yang hanya dilakukan oleh satu orang dalam *platform meeting conference*. Pengawas biasanya mengawasi kurang lebih 10 – 20 orang dalam satu *platform meeting conference*. Pengawas membutuhkan waktu yang cukup lama untuk melihat setiap aktivitas peserta karena harus mengganti halaman *platform meeting*. Kemudian koneksi internet peserta juga menjadi sebuah permasalahan yang cukup serius karena dengan kurang baiknya koneksi internet mengakibatkan pergerakan video dari peserta menjadi terputus – putus.

Kecurangan pada saat ujian sering sekali dilakukan karena ujian merupakan evaluasi dari hasil pembelajaran. Hasil dari evaluasi ini merupakan sebuah tolak ukur keberhasilan dari sebuah pembelajaran. Tolak ukur ini membuat seseorang ingin mendapatkan nilai yang tinggi dan menjadi kebanggaan orang tua serta merasa pintar

di lingkungan sekitarnya. Pandangan seperti ini yang membuat seseorang melakukan berbagai upaya dalam meraih nilai yang tinggi dengan cara melakukan kecurangan akademik.

Hasil penelitian dari (Indriani, 2019) menyatakan bahwa 55% dari keseluruhan responden melakukan tindakan kecurangan yaitu menyontek pada saat ujian. Hal ini sudah menjadi kebiasaan yang buruk dalam meraih nilai yang tinggi. Untuk kegiatan menyontek tidak melihat kalangan usia mulai dari yang muda hingga yang tua. Hal ini disebabkan oleh keinginan meraih nilai yang baik secara instan tanpa melalui proses pembelajaran yang baik.

## **2.2 Citra *Grayscale***

Citra *grayscale* adalah citra yang direpresentasikan dalam skala keabuan, di mana setiap piksel hanya memiliki satu nilai kecerahan tunggal yang mencerminkan tingkat keabuan atau intensitas cahaya pada posisi piksel tersebut. Citra *grayscale* menggambarkan kecerahan pada setiap titik tanpa memperhatikan warna. Citra *grayscale* juga merupakan perhitungan dari intensitas cahaya pada setiap piksel pada spektrum elektromagnetik *single band*. Citra *grayscale* disimpan pada format 8 bit untuk setiap sampel piksel yang memungkinkan sebanyak 256 intensitas.

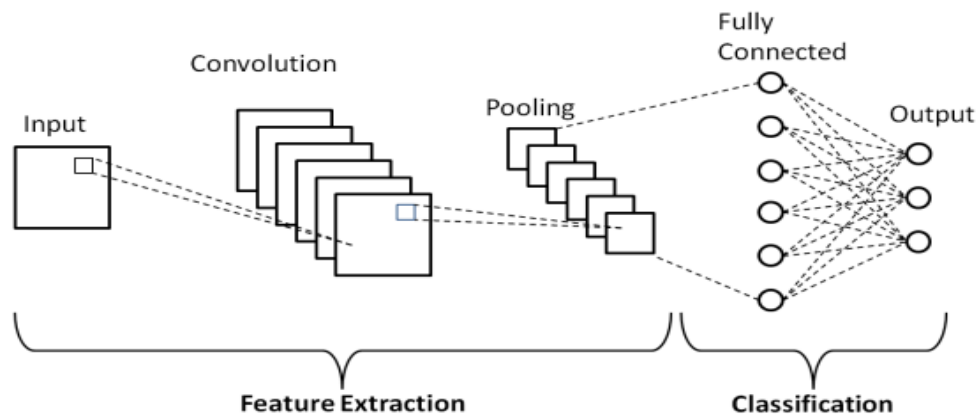
## **2.3 *Convolutional Neural Network***

*Convolutional Neural Networks* atau yang biasa dikenal CNN merupakan salah satu *deep learning* model yang sangat baik dalam memproses dan menganalisis data visual yang berbentuk gambar dan video. CNN sudah merevolusi bidang *Computer Vision* dalam pengenalan gambar, deteksi objek dan segmentasi gambar.

CNN didesain untuk memproses *input* data dengan *grid-like topology*, seperti citra, dengan cara mengaplikasikan sebuah sistem filter yang juga dikenal sebagai kernel atau *weight* untuk *input* yang diterima. Setiap kernel menekankan fitur – fitur tertentu dalam citra. Hasil konvolusi menghasilkan peta fitur yang mewakili fitur yang diambil. Fungsi aktivasi ReLU (*Rectified Linear Unit*) diterapkan pada hasil konvolusi untuk memperkenalkan non-linearitas agar membantu jaringan lebih mudah mempelajari pola yang kompleks dan abstrak. Lapisan *pooling* melakukan *subsampling* untuk mengurangi dimensi pada *feature map* dan mengurangi jumlah parameter.

Lapisan *output* memberikan hasil klasifikasi atau regresi berdasarkan pola yang telah dipelajari oleh jaringan.

Proses ini akan berulang melewati beberapa siklus pelatihan sehingga jaringan mendapatkan prediksi yang akurat pada data. Bisa dilihat pada gambar 2.1 untuk arsitektur umum CNN.



**Gambar 2.1 Arsitektur Umum *Convolutional Neural Network***

(Sumber : <https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697>)

Pada persamaan 2.1 merupakan persamaan untuk menghitung dimensi dari *feature map*.

$$Output = \frac{W - N + 2P}{S} + 1 \dots\dots\dots (2.1)$$

W = Panjang / Tinggi *Input*

N = Panjang / Tinggi Filter

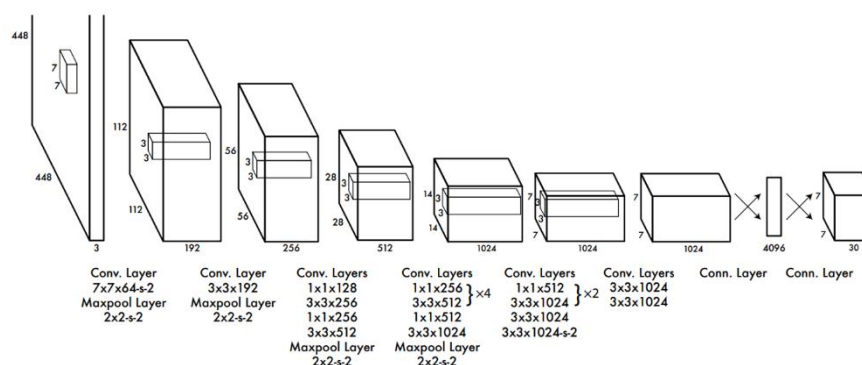
P = *Zero Padding*

S = Stride

## 2.4 Algoritma *You Only Look Once* (YOLO)

*You Only Look Once* (YOLO) merupakan algoritma yang menerapkan konsep *neural network* untuk mempelajari pola sehingga dapat mendeteksi objek dalam waktu *real-time*. YOLO memungkinkan untuk mendeteksi objek dalam satu kali pengamatan sekaligus, tanpa harus memerlukan proses komputasi yang berulang – ulang. YOLO memahami tampilan citra sebagai masalah regresi dan menerapkan *deep learning* untuk menghasilkan *bounding box*. YOLO membagi citra menjadi beberapa *cell* yang di mana

setiap *cell* memiliki fungsi untuk memprediksi jumlah dari *bounding box*, *confidence level* dari setiap *cell* dan probabilitas setiap *cell*. Pada gambar 2.2 dapat dilihat untuk arsitektur YOLO.



**Gambar 2.2 Arsitektur YOLO**

(Sumber : <https://pacmann.io/blog/cara-kerja-object-detection-dengan-yolo>)

Arsitektur umum dari YOLO menggunakan CNN yang terdiri dari 24 *convolutional layers* dan diikuti dengan 2 *connected layers*. YOLO menggunakan *Initial Convolutional Layers* sebagai pengekstrak *feature* dari gambar dan *connected layers* memprediksi probabilitas dan koordinat untuk objek yang mungkin muncul dari gambar (Redmon et al., 2015a). Koordinat memuat informasi tentang letak (*bounding box*) dari objek yang terdeteksi pada gambar.

## 2.5 YOLOv8

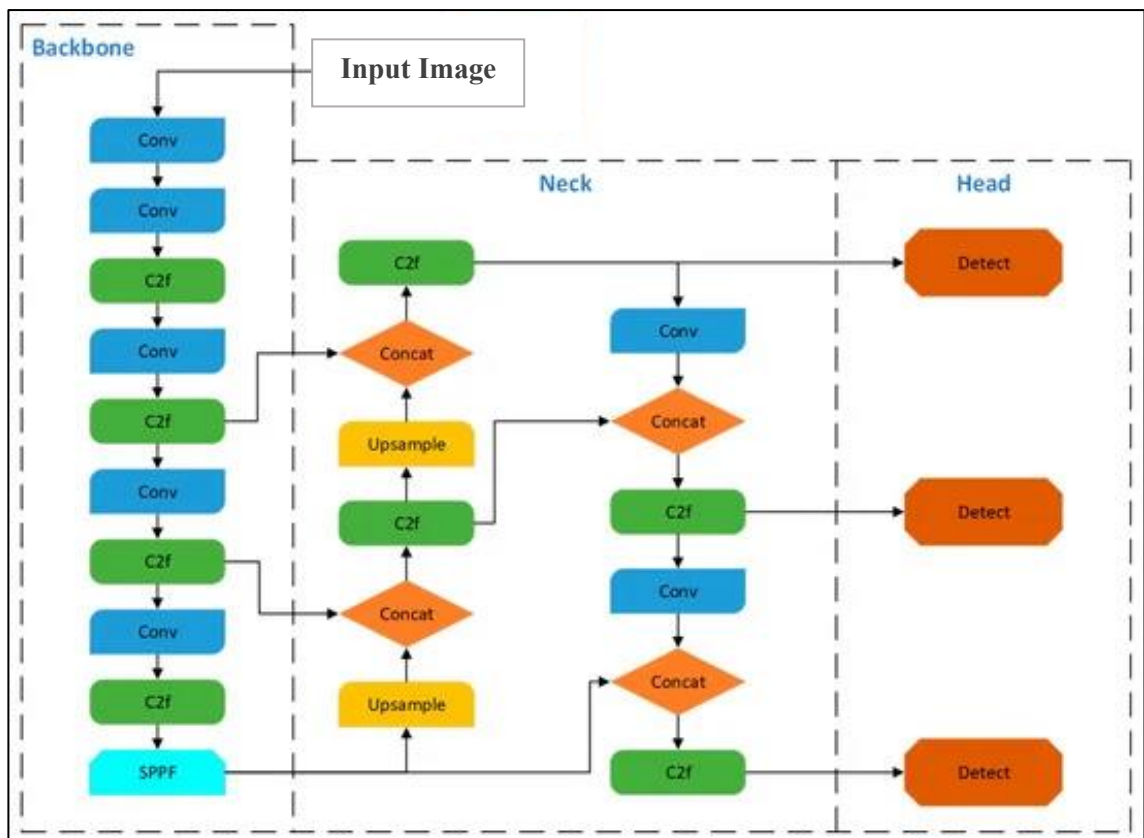
YOLOv8 merupakan algoritma hasil pengembangan arsitektur yang ada pada YOLOv5. YOLOv8 memperkenalkan berbagai modifikasi seperti *spatial attention*, *feature fusion* dan *context aggregation modules*. Hasil dari modifikasi ini meningkatkan waktu pendeteksian objek lebih cepat dan juga akurat. YOLOv8 menggunakan CNN dengan membaginya menjadi 2 bagian yaitu *backbone* dan *head*. Pada bagian *backbone* terdapat arsitektur CSPDarknet53 yang digunakan pada YOLOv5. Arsitektur CSPDarknet53 telah dimodifikasi sehingga terdapat 53 *convolutional layers* dan menggunakan *cross-stage partial* untuk meningkatkan peredaran informasi antar *layer* yang berbeda. Pada bagian *head* beberapa *convolutional layers* yang diikuti oleh *fully connected layers*. Layers ini bertanggung jawab untuk memprediksi *bounding box*,

*object scores* dan probabilitas dari objek yang terdeteksi. Pada bagian *head* YOLOv8 menggunakan *self-attention mechanism*.

Arsitektur yang ada pada YOLOv8 telah mengalami beberapa pembaharuan dari arsitektur YOLOv5. Beberapa pembaharuan itu adalah sebagai berikut :

1. *Backbone* pada YOLOv8 mengalami perubahan dengan mengganti C3 dengan C2f dan melakukan pergantian 6x6 *convolution* dengan 3x3 *convolution*. Pada C2f menggabungkan dua dari 3x3 *convolution* dengan koneksi sisa untuk output dari *bottleneck*.
2. Dua dari *convolutions* yang ada pada arsitektur YOLOv5 telah dihapus (#10 dan #14)
3. *Convolution* pertama pada ukuran kernel telah berubah dari 1x1 menjadi 3x3. Perubahan ini menunjukkan pergeseran menuju blok ResNet.

Adapun arsitektur YOLOv8 dapat dilihat pada gambar 2.3.

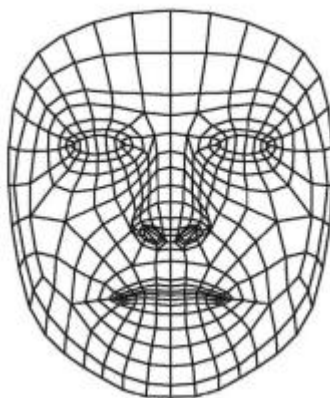


**Gambar 2.3 Arsitektur YOLOv8**

(Sumber : <https://www.mdpi.com/2079-9292/12/17/3664>)

## 2.6 Mediapipe

*Mediapipe* merupakan sebuah *framework* untuk membangun *pipeline* untuk melakukan pada data sensorik seperti gambar dan video (Lugaresi et al., 2019). *Mediapipe* biasa digunakan untuk pembuatan prototipe cepat saluran persepsi dengan inferensi model dan komponen lain yang dapat digunakan kembali. Dengan menggunakan *Mediapipe*, tujuan dari *pipeline* dapat dituangkan dalam bentuk grafik komponen modular termasuk inferensi model, algoritma pemrosesan media dan transformasi data, dan sebagainya. Data-data seperti audio dan video masuk ke grafik kemudian diterjemahkan dalam aliran lokalisasi objek dan *landmark* keluar dari grafik. Penggunaan *Mediapipe* untuk *face landmark* untuk melihat pergerakan dari kepala, mata dan mulut. *Mediapipe* menggunakan 468 titik yang akan disematkan di wajah untuk mendeteksi wajah. Hal ini lebih baik dari Dlib yang hanya menggunakan 68 titik. Adapun gambar dari *face landmark Mediapipe* dapat dilihat pada gambar 2.4.



**Gambar 2.4 Face Landmark Mediapipe**

(Sumber : [https://dblib.rsreu.ru/data/publications/8358\\_text.pdf](https://dblib.rsreu.ru/data/publications/8358_text.pdf))

## 2.7 OpenCV

*OpenCV* merupakan *library open-source* dalam *computer vision*. *OpenCV* memiliki kemampuan yang lebih efisien dalam *face recognition* daripada Dlib. *OpenCV* dapat digunakan oleh banyak bahasa pemrograman, seperti *python*, C++ dan *Java*. *OpenCV* digunakan untuk menganalisis wajah, artefak dan tulisan tangan manusia. Ketika dipasangkan dengan banyak *library* dapat meningkatkan kinerja lebih baik. *OpenCV* pertama kali diluncurkan secara resmi pada tahun 1999.



## 2.8 Confusion Matrix

*Confusion matrix* adalah suatu metode pengukuran performa untuk mengevaluasi masalah klasifikasi *machine learning* yang memberikan hasil berupa dua kelas atau lebih. *Confusion matrix* biasa disebut juga *error matrix*. *Confusion matrix* memberikan informasi mengenai perbandingan antara model dengan hasil klasifikasi sebenarnya. *Confusion matrix* berbentuk sebuah tabel matriks yang menggambarkan kinerja dari model klasifikasi pada sekumpulan data uji yang nilai sebenarnya sudah diketahui. Adapun gambar yang menjelaskan mengenai 4 kombinasi nilai prediksi dan nilai aktual yang dapat dilihat pada gambar 2.5.

		<b>Actual Values</b>	
		<b>1 (Positive)</b>	<b>0 (Negative)</b>
<b>Predicted Values</b>	<b>1 (Positive)</b>	<b>TP</b> (True Positive)	<b>FP</b> (False Positive) <i>Type I Error</i>
	<b>0 (Negative)</b>	<b>FN</b> (False Negative) <i>Type II Error</i>	<b>TN</b> (True Negative)

**Gambar 2.5 Confusion Matrix**

(Sumber : <https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f>)

Berdasarkan gambar 2.5 terdapat 4 istilah dalam hasil proses klasifikasi pada *confusion matrix*, yaitu :

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

Dari keempat istilah tersebut, *confusion matrix* digunakan untuk menghitung berbagai *performance matrix* yang digunakan untuk mengukur kinerja model yang telah dibuat. Penghitungan *performance matrix* sebagai berikut :

- *Accuracy*

*Accuracy* menggambarkan keakuratan model untuk dapat mengklasifikasikan dengan benar. *Accuracy* juga merupakan kedekatan antara nilai prediksi dengan nilai aktual.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \dots\dots\dots (2.2)$$

- *Precision*

*Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model.

$$Precision = \frac{TP}{TP + FP} \dots\dots\dots (2.3)$$

- *Recall*

*Recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi.

$$Recall = \frac{TP}{TP + FN} \dots\dots\dots (2.4)$$

- *F1-Score*

*F1-Score* merupakan metrik hasil evaluasi *machine learning* yang mengukur akurasi model.

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \dots\dots\dots (2.5)$$

## 2.9 Penelitian Terdahulu

Terdapat beberapa penelitian mengenai deteksi kecurangan yang sebelumnya sudah pernah dilakukan, di antaranya adalah penelitian oleh (Ahmad et al., 2021) yang berjudul “*A Novel Deep Learning-based Online Proctoring System using Face Recognition, Eye Blinking, and Object Detection Techniques*”. Penelitian ini berfokus pada *face recognition* dengan metode *Histograms of Oriented Gradients* (HOG) dan juga mendeteksi objek kecurangan seperti buku, ipad, *handphone* dan laptop. Data yang digunakan untuk melatih *face recognition* diambil dari *FDDB datasets* dengan mengubah ukuran piksel menjadi 86x86. Kemudian data tersebut dievaluasi

menggunakan LFW *datasets*. Penelitian ini membuat parameter apabila tidak ada mata berkedip selama 30 detik itu menandakan gambar wajah palsu. Penelitian ini menggunakan YOLOv3 untuk mendeteksi objek kecurangan.

Penelitian selanjutnya tentang deteksi kecurangan adalah penelitian yang dilakukan oleh (Abdul Elah Alkhalisy & Hameed Abid, 2023) dengan judul “*The Detection of Students' Abnormal Behavior in Online Exams Using Facial Landmarks in Conjunction with the YOLOv5 Models*”. Penelitian ini menggunakan *dataset* yang dikumpulkan dengan merekam perilaku peserta menggunakan aplikasi yang telah dimodifikasi untuk simulasi ujian. *Dataset* ini diambil dari 6 orang. Hasil dari rekaman tersebut diubah menjadi kumpulan *frame*. Kemudian *frame* dari *dataset* yang berukuran 1280x720 piksel di *resize* menjadi 640x640 sebelum digunakan pada saat proses *training*. Penelitian ini menggunakan YOLOv5 untuk deteksi perilaku peserta pada saat ujian, seperti menggunakan *handphone*, pergerakan tangan, pergerakan mata, dan pergerakan kepala. Kemudian penelitian ini menggunakan YOLOv3 untuk mendeteksi objek dan *multi person*. Selanjutnya untuk *facial landmark* menggunakan Dlib.

Penelitian lainnya mengenai deteksi kecurangan adalah penelitian yang dilakukan oleh (Yulita et al., 2023) dengan judul “*Educational Innovation Faced with COVID-19: Deep Learning for Online Exam Cheating Detection*”. Penelitian ini menggunakan *dataset* dari Michigan State University. *Dataset* ini kemudian masuk ke tahap *preprocessing* yang dimana akan dilakukan augmentasi data dan *resize* data. Setelah itu, *dataset* dibagi menjadi data *training* dan data validasi. Kemudian data diproses dengan model CNN dengan arsitektur MobileNetV2. Penelitian ini menghasilkan nilai F1-score sebesar 84.52% dengan *learning rate* 0.0001 dan ukuran *batch* sebesar 16. Penelitian ini menggunakan *dataset* dengan 4 kelas yaitu *no cheat*, *read text*, *ask friend*, dan *call friend*.

Selanjutnya penelitian mengenai deteksi kecurangan yang dilakukan oleh (Gopalakrishnan et al., 2022) dengan judul “*Online proctoring system using image processing and machine learning*”. Penelitian ini berfokus pada improvisasi *gaze tracking* dan pergerakan mulut terbuka atau tertutup. Penelitian ini menggunakan 20 data video berdurasi 1 menit dengan pembagian 10 data video dengan kecurangan dan 10 data video tidak mengandung kecurangan. *Face recognition* menggunakan metode *Local Binary Patterns Histograms* (LBPH) dengan *library OpenCV2*. Untuk

pergerakan mulut dan *gaze tracking*, penelitian ini menggunakan *Dlib library*. Penelitian ini mendapatkan akurasi sebesar 90%.

Penelitian lainnya tentang deteksi kecurangan pernah dilakukan oleh (Malhotra et al., 2022) dengan judul “*Smart Artificial Intelligence Based Online Proctoring System*”. Penelitian ini menggunakan *COCO dataset* untuk mendeteksi orang dan *handphone* dengan algoritma YOLOv3. Kemudian untuk mendeteksi wajah, penelitian ini menggunakan modul OpenCV’s DNN untuk melakukan *preprocessing dataset*. Dalam proses *preprocessing* dilakukan *reflip* dan *resize dataset*. Setelah itu dilakukan konversi *grayscale* menggunakan OpenCV. Kemudian untuk pendeteksi wajah menggunakan algoritma *Dlib’s Histogram of Oriented Gradients*. Selanjutnya penelitian ini menggunakan algoritma *Dlib library’s 68 point facial landmark* untuk mendeteksi pergerakan kepala, mata, dan mulut.

Penelitian lainnya mengenai deteksi kecurangan pernah dilakukan oleh (Motwani et al., 2021) dengan judul “*AI-Based Proctoring System for Online Tests*”. Penelitian ini mengumpulkan data wajah peserta pada saat registrasi. Kemudian menggunakan *Local Binary Pattern Histogram (LBPH)* untuk algoritma *face recognition*. Penelitian ini juga menggunakan YOLOv3 untuk mendeteksi orang dan *handphone* dengan nilai *top-5 accuracy* sebesar 93.3%. Penelitian ini menganalisis hasil dari deteksi yang dikumpulkan dalam bentuk *log* yang akan dipertimbangkan berdasarkan nilai yang didapatkan pada saat ujian *online*.

Penelitian selanjutnya mengenai deteksi kecurangan pernah dilakukan oleh (Anagha, 2023) dengan judul “*Automated Online Remote Proctoring System*”. Penelitian ini menggunakan YOLOv3 untuk menghitung jumlah orang dan mendeteksi *handphone*. Kemudian penelitian ini menggunakan *Dlib face landmark* untuk mendeteksi pergerakan kepala dan *gaze tracking*. Penelitian ini mendapatkan hasil 6.76 *frame per second* dengan menggunakan *processor i5*. Lalu penelitian ini juga menerapkan API Google’s Speech untuk merekam audio kemudian mengubahnya menjadi teks. Lalu teks disimpan dalam file .txt yang akan dikirimkan pada pengawas. Penelitian ini juga merekam jendela aktif setiap 5 detik dan mematikan fungsi *cut, paste, copy*, dan mengambil *screenshoot*.

Ringkasan dari penelitian – penelitian terdahulu data dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis	Tahun	Metode	Keterangan
1.	Ahmad et al.	2021	YOLOv3 dan Dlib	Menggunakan YoloV3 untuk mendeteksi orang dan handphone dengan nilai mAP 57.9% dalam waktu 51 milisekon. Mendapatkan akurasi 97.21% dalam <i>face recognition</i> dengan metode HOG.
2.	Abdul Elah Alkhalisy & Hameed Abid	2023	YOLOv3, YOLOv5, dan Dlib	Penelitian ini membahas tentang deteksi kecurangan dan penelitian ini mendapatkan <i>over all accuracy</i> 95%.
3.	Yulita et al.	2023	MobileNetV2	MobileNetV2 yang digunakan untuk mendeteksi peserta apabila melakukan kecurangan berupa membaca buku, bertanya dengan teman, dan memanggil teman dengan rata-rata F1-Score sebesar 84.52%
4.	Gopalakrishnan et al.	2022	YOLO dan Dlib	Penelitian ini menggunakan MPGazell untuk <i>gaze tracking</i> dan VGG16 untuk <i>face recognition</i> . Penelitian ini mendapatkan akurasi sebesar 90% dari 20 <i>datasets</i> video berdurasi 1 menit.

Tabel 2.1 Penelitian Terdahulu ( Lanjutan )

No.	Penulis	Tahun	Metode	Keterangan
5.	Malhotra et al.	2022	YOLOv3 dan Dlib	Penelitian ini menggunakan COCO dataset untuk deteksi objek kecurangan dan menggunakan Dlib untuk tracking kepala, mulut dan mata. Pengenalan wajah mendapatkan hasil <i>19.5 frame per second</i> .
6.	Motwani et al.	2021	YOLOv3	Penelitian ini membahas tentang deteksi kecurangan. Penelitian ini menggunakan LBPH untuk <i>face recognition</i> . Penelitian ini mendapatkan top-5 <i>accuracy</i> 93.3%.
7.	Anagha et al.	2023	YOLOv3 dan Dlib	Penelitian ini menggunakan YOLOv3 untuk mendeteksi objek kecurangan dan Dlib untuk pergerakan mata dan kepala. Pada penelitian ini menggunakan Google's Speech API untuk mengkonversi suara menjadi bentuk teks.

Perbedaan yang terdapat dalam penelitian ini adalah pada bagian penggunaan metode YOLOv8 yang digunakan sebagai metode untuk mendeteksi objek kecurangan. Selain itu, penelitian ini juga menggunakan Mediapipe sebagai *face landmark* untuk menentukan arah pergerakan kepala, mulut dan mata dari para peserta. Penelitian ini juga menerapkan *face recognition* dengan metode LBPH yang akan digunakan sebelum memasuki sesi ujian.

## BAB 3

### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1 Dataset

Dataset yang digunakan pada penelitian ini berupa *image*. Data *image* dibagi menjadi dua bagian, seperti gambar wajah dan gambar objek kecurangan. Data gambar wajah didapatkan dari proses registrasi sebelum memulai ujian. Data gambar objek kecurangan merupakan gambar buku, *phone* dan orang lain. Aplikasi akan dilatih untuk dapat mengerjakan pengenalan wajah peserta ujian, mendeteksi pergerakan kepala, mata dan mulut peserta ujian dan juga dapat mendeteksi objek kecurangan. Aplikasi diharapkan dapat memahami dan memberikan hasil deteksi yang kemudian menjadi laporan kepada pengawas. Contoh *image dataset* dapat dilihat pada gambar 3.1.

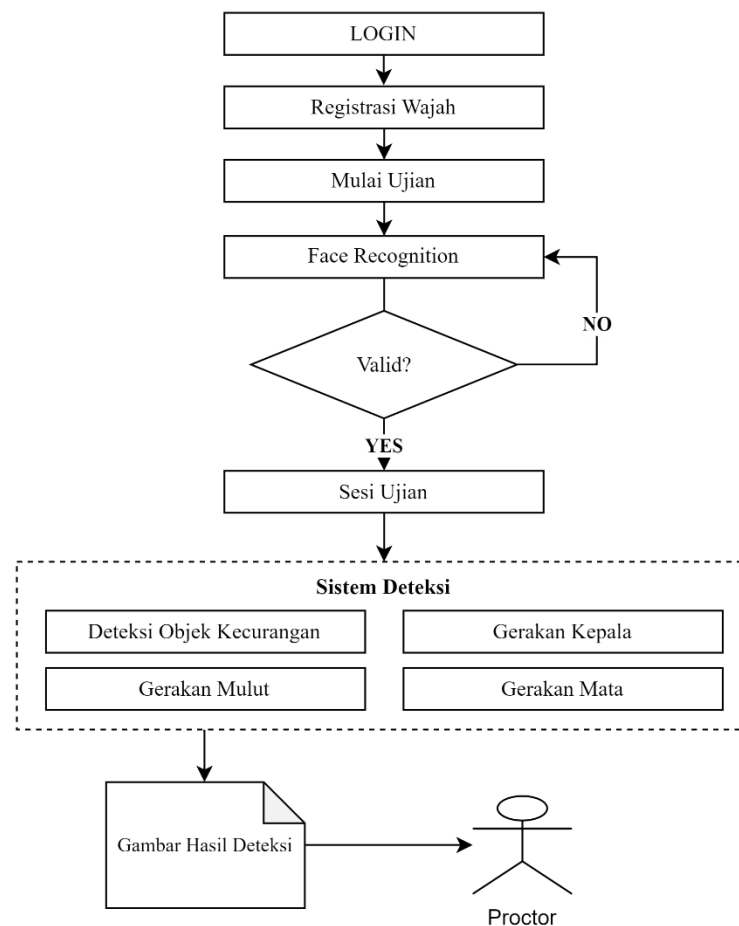


**Gambar 3.1 *Image Datasets***

Pada gambar 3.1 merupakan contoh dari masing – masing dataset yang telah terkumpul. Sesuai dengan yang digunakan pada penelitian ini yang terdiri dari buku, *phone* dan orang lain. Dataset ini akan dilatih menggunakan YOLO versi 8 untuk mendeteksi objek kecurangan, selain itu wajah peserta akan dilatih untuk proses *face recogniton* dengan LBPH. Kemudian pergerakan wajah tiap peserta akan dideteksi dengan menggunakan *face landmark* Mediapipe.

### 3.2 Analisis Sistem

Penelitian ini akan menggunakan beberapa metode dalam mendeteksi kecurangan. Sesuai dengan gambar 3.2 yang menjelaskan alur kerja sistem dari penelitian ini. Pada gambar 3.3 akan menjelaskan mengenai arsitektur umum yang digunakan dalam penelitian ini. Penelitian ini akan menjelaskan arsitektur umum *face recognition* pada gambar 3.4. Selanjutnya untuk arsitektur umum deteksi objek kecurangan akan dijelaskan pada gambar 3.10. Selanjutnya untuk arsitektur dari *face landmark Mediapipe* akan dijelaskan pada gambar 3.15.

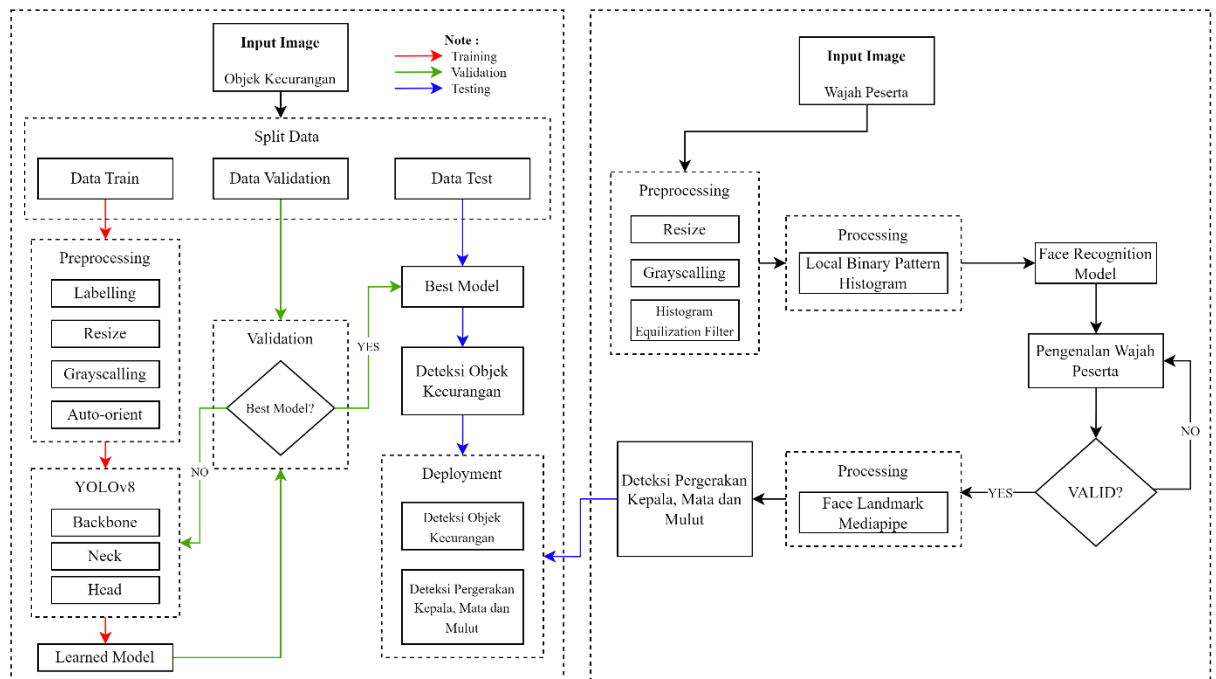


**Gambar 3.2 Alur Kerja Sistem**

Pada gambar 3.2 merupakan alur sistem dari penelitian ini. Alur ini dimulai dari proses *login* oleh peserta ujian. Kemudian akan dilakukan registrasi wajah peserta pada menu *profile*. Setelah itu pada saat peserta memulai ujian, peserta wajib melakukan verifikasi

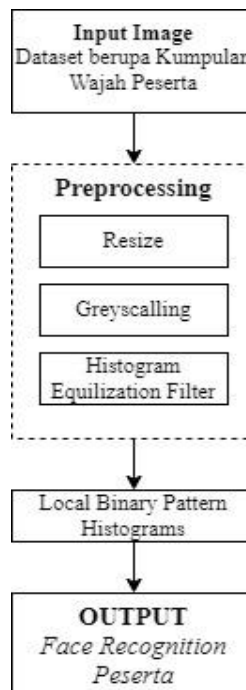


biometrik menggunakan *face recognition*. Pada saat verifikasi biometrik akan dilakukan proses validasi, apabila wajah tidak cocok akan dilakukan verifikasi ulang dan apabila berhasil akan masuk ke dalam sesi ujian dengan *live webcam*. Pada saat sesi ujian ini akan ada beberapa metode yang akan digunakan untuk mendeteksi kecurangan seperti deteksi objek kecurangan dan deteksi pergerakan kepala, mulut dan mata. *Log* dari setiap aktivitas yang mencurigakan akan direkam dan diberikan kepada pengawas.



**Gambar 3.3 Arsitektur Umum**

Pada gambar 3.3 menjelaskan arsitektur umum yang akan digunakan dalam penelitian ini. Penelitian ini mengusulkan suatu metode yang digunakan untuk mendeteksi suatu kecurangan dalam CBT *online*. Berdasarkan arsitektur umum pada gambar 3.3, penelitian ini dibagi menjadi tiga tahapan yakni *face recognition* dalam pengolahan gambar wajah peserta menggunakan metode LBPH, *object detection* dalam pengolahan gambar objek kecurangan dan *face landmark* dalam pengolahan gambar wajah peserta menggunakan Mediapipe. Metodologi sebagaimana yang ditunjukkan pada gambar 3.4, gambar 3.10 dan gambar 3.13 diharapkan dapat menerima *input* secara *real time* dari perangkat *webcam* pengguna. Hasil dari proses yang ada di dalam metodologi ini berupa hasil deteksi yang diperoleh pada saat peserta melaksanakan ujian.



**Gambar 3.4** *Arsitektur Proses Face Recognition*

Pada gambar 3.4 menjelaskan arsitektur umum dari *face recognition* dalam penelitian ini. Berikut penjelasan rinci mengenai arsitektur umum *face recognition* :

1. *Input*

Tahapan yang pertama kali akan dilakukan adalah meng-*input* data ke sistem berupa data gambar wajah dari peserta. Data gambar wajah peserta ini dikumpulkan pada saat registrasi.

2. *Preprocessing*

Tahapan kedua adalah *preprocessing* pada data wajah yang sudah ada di dalam sistem dengan tujuan mengolah data wajah agar mendapatkan hasil yang lebih baik untuk masuk ke dalam tahapan berikutnya. Tahapan yang dilakukan pada *preprocessing* adalah *Resize*, *Gray Scalling*, dan *Histogram Equalization Filter*.

a. *Resize*

Pada tahap ini dilakukan perubahan dimensi dari data wajah peserta. Tahap ini dilakukan agar data gambar dari setiap peserta memiliki kesamaan ukuran menjadi 100x100 piksel.

```
original_width <-- getWidth(captured_image)
original_height <-- getHeight(captured_image)
target_width <-- 100
target_height <-- 100
resized_image <-- createEmptyImage(target_width, target_height)
scale_x <-- original_width / target_width
scale_y <-- original_height / target_height
for y from 0 to target_height - 1 do
  for x from 0 to target_width - 1 do
    source_x <-- x * scale_x
    source_y <-- y * scale_y
    source_pixel <-- captured_image[round(source_x), round(source_y)]
    resized_image[x, y] <-- source_pixel
  end for
end for
```

**Gambar 3.5 Pseudocode resize**

b. *Gray Scalling*

Setelah tahap *resize* akan dilakukan perubahan warna citra wajah peserta dari RGB menjadi abu. Tujuan perubahan warna ini agar pada saat proses ekstraksi fitur wajah lebih mudah tanpa harus menangani data warna yang lebih kompleks.

```
grayscale_image <-- createEmptyImage(target_width, target_height)
for y from 0 to target_height - 1 do
  for x from 0 to target_width - 1 do
    pixel <-- resized_image[x, y]
    red <-- getRed(pixel)
    green <-- getGreen(pixel)
    blue <-- getBlue(pixel)
    grayscale_value <-- 0.299 * red + 0.587 * green + 0.114 * blue
    grayscale_image[x, y] <-- grayscale_value
  end for
end for
```

**Gambar 3.6 Pseudocode grayscaling**

c. *Histogram Equalization Filter*

Selanjutnya akan ada tahapan untuk meningkatkan kontras dari citra yang telah melalui tahap *Gray Scalling*. Tujuan dilakukan peningkatan kontras ini agar memperjelas fitur wajah dan juga menghilangkan ketidakseimbangan cahaya.

```

histogram <-- calculateHistogram( grayscale_image )
cdf <-- calculateCumulativeDistributionFunction( histogram )
equalized_image <-- createEmptyImage( target_width, target_height )
total_pixels <-- target_width * target_height
min_cdf <-- findMinNonZeroValue( cdf )
for y from 0 to target_height - 1 do
  for x from 0 to target_width - 1 do
    original_pixel_value <-- grayscale_image[ x, y ]
    equalized_pixel_value <-- round( (cdf[original_pixel_value] - min_cdf) /
                                     (total_pixels - min_cdf) * 255 )
    equalized_image[ x, y ] <-- equalized_pixel_value
  end for
end for

```

**Gambar 3.7 Pseudocode histogram equalization**

### 3. Local Binary Pattern Histograms

Selanjutnya citra yang telah melewati *preprocessing* akan masuk tahapan ekstraksi fitur menggunakan metode *Local Binary Pattern Histograms* (LBPH). Pada tahapan ini LBPH akan mengubah setiap piksel menjadi nilai biner dengan membandingkan nilai piksel dengan nilai piksel di sekitarnya. Setelah menjadi *LBP image*, histogram dari *LBP image* digunakan menjadi fitur wajah. Setelah itu memisahkan hasil LBP menjadi bagian kecil yang disebut *grid* dan kemudian digabungkan menjadi satu histogram baru. Histogram baru ini yang akan mengenali wajah.

```

Parameters: radius (default <-- 1), neighbors (default <-- 8),
            grid_x (default <-- 8), grid_y (default <-- 8)

lbp_image <-- createEmptyImage( target_width, target_height )
for y from radius to target_height - radius - 1 do
  for x from radius to target_width - radius - 1 do
    center_pixel <-- equalized_image[ x, y ]
    binary_pattern <-- ""
    for neighbor in 0 to neighbors - 1 do
      angle <-- 2 * PI * neighbor / neighbors
      dx <-- round( radius * cos( angle ) )
      dy <-- round( radius * sin( angle ) )
      neighbor_pixel <-- equalized_image[ x + dx, y + dy ]
      if neighbor_pixel > center_pixel then
        binary_pattern.append(1)
      else
        binary_pattern.append(0)
      end for
    lbp_value <-- binaryToDecimal( binary_pattern )
    lbp_image[ x, y ] <-- lbp_value
  end for
end for

```

**Gambar 3.8 Pseudocode menghitung LBP**

```

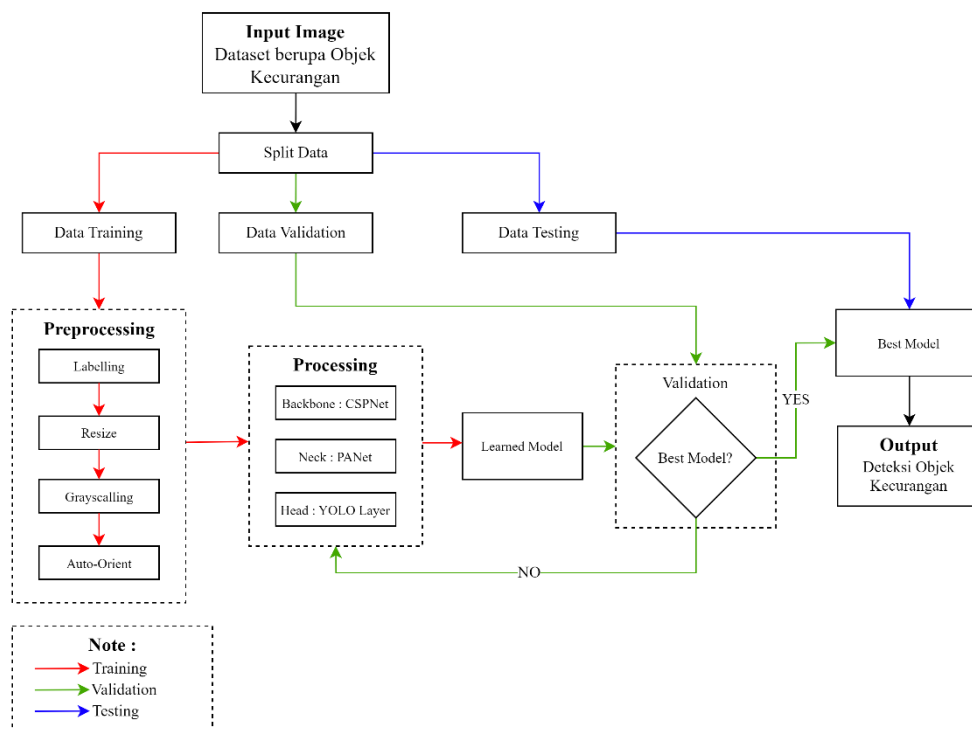
grid_width <-- target_width / grid_x
grid_height <-- target_height / grid_y
histograms <-- []
for gy from 0 to grid_y - 1 do
  for gx from 0 to grid_x - 1 do
    grid_histogram <-- createEmptyHistogram()
    for y from gy * grid_height to (gy + 1) * grid_height - 1 do
      for x from gx * grid_width to (gx + 1) * grid_width - 1 do
        lbp_value <-- lbp_image[x, y]
        grid_histogram[lbp_value] +<-- 1
      end for
    end for
    histograms.append(grid_histogram)
  end for
end for
feature_vector <-- concatenateHistograms(histograms)

```

**Gambar 3.9 Pseudocode membagi LBP menjadi beberapa grid**

#### 4. Output

Setelah melewati proses pengenalan wajah, *output* dari *face recognition* ini akan digunakan dalam proses verifikasi biometrik dan juga pada saat sesi ujian berlangsung untuk mengenali wajah peserta ujian.



**Gambar 3.10 Arsitektur Pemrosesan Deteksi Objek YOLO**

Berikut penjelasan mengenai proses pada gambar 3.10 arsitektur umum deteksi objek kecurangan menggunakan YOLOv8 :

### 1. *Input*

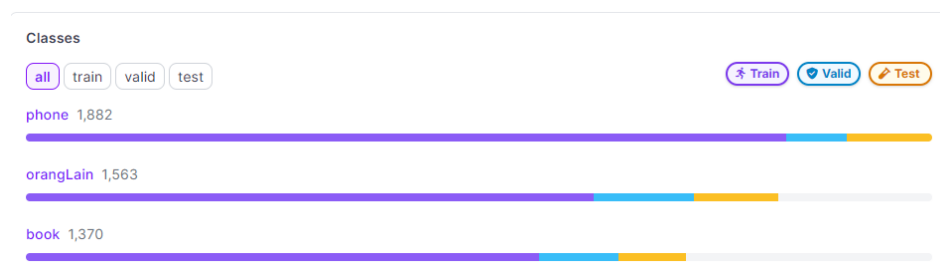
Tahapan awal ini menginput data citra objek kecurangan ke dalam sistem. Adapun objek yang digunakan pada penelitian ini adalah *handphone*, *buku*, dan *person*. Data citra objek kecurangan ini data yang diambil dengan kamera *webcam*. Selanjutnya data citra dibagi ke dalam tiga bagian, yaitu : data *training*, data *validation*, dan data *testing*.

### 2. *Preprocessing*

Tahapan kedua adalah preprocessing data citra objek kecurangan yang sudah ada di dalam sistem. Tujuan pada *preprocessing* ini adaah agar mendapatkan hasil maksimal pada saat melanjutkan ke tahapan berikutnya. Tahapan yang dilakukan pada *preprocessing* adalah *resize*, *labelling*, dan *grayscale*.

#### a. *Labelling*

*Labelling* merupakan tahapan untuk memberikan label atau penamaan pada data citra yang akan digunakan pada pengujian sistem. *Labelling* dilakukan untuk memberikan tanda kelas pada citra objek kecurangan yang akan digunakan.



**Gambar 3.11 Data yang di-labelling**

Data citra akan diberi label dengan menganotasi pada setiap citra, terdiri dari empat kelas yang sudah ditetapkan, yaitu *phone*, *person* dan *book*. Setelah proses pelabelan, hasilnya akan disimpan dalam bentuk folder yang nantinya digunakan dalam tahap pelatihan data.

#### b. *Resize*

Pada tahap ini dilakukan pengubahan dimensi dari data citra objek kecurangan agar semua data memiliki ukuran piksel yang sama. Ukuran

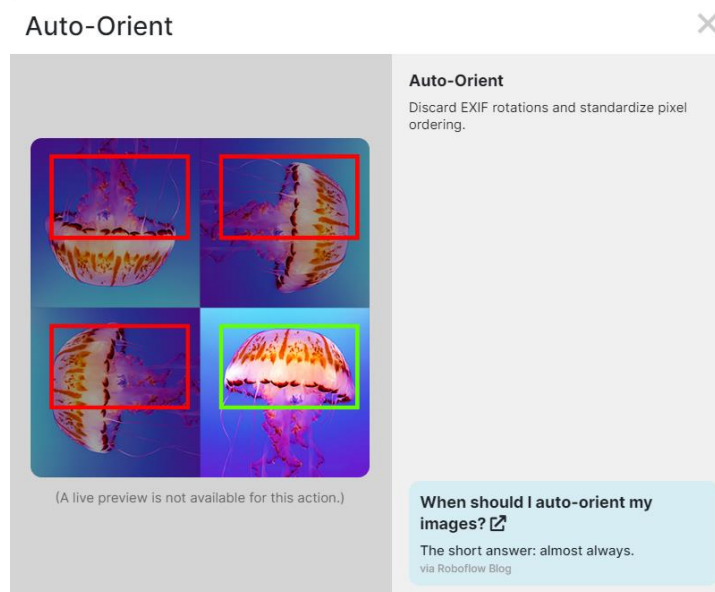
piksel mempengaruhi pemrosesan sistem, apabila ukuran piksel besar akan memakan waktu yang lama.

c. *Gray Scalling*

Setelah tahap *labelling* akan dilakukan perubahan warna citra objek kecurangan dari RGB menjadi abu. Tujuan perubahan warna ini agar pada saat proses ekstraksi citra objek lebih mudah tanpa harus menangani data warna yang lebih kompleks.

d. *Auto - Orient*

*Auto - orient* pada gambar atau foto biasanya merujuk pada penyesuaian otomatis orientasi gambar berdasarkan metadata EXIF (*Exchangeable Image File Format*) yang tersimpan di dalam *file* gambar. Kamera digital dan smartphone menyimpan informasi orientasi pada metadata EXIF, yang mencakup apakah gambar diambil dalam orientasi potrait atau landscape.



**Gambar 3.12 Penerapan *Auto-orient***

*Auto - orient* memiliki kegunaan untuk memastikan bahwa data atau gambar berada dalam orientasi yang benar dan konsisten sebelum melanjutkan ke tahap analisis atau pemrosesan lebih lanjut, sehingga mengurangi kesalahan dan meningkatkan akurasi hasil model yang dihasilkan.

### 3. *Processing*

Pada tahap ini citra akan diproses menggunakan algoritma YOLO versi 8. YOLOv8 merupakan salah satu algoritma yang dapat mendeteksi objek sesuai dengan kelas yang telah ditetapkan sebelumnya. Adapun proses yang ada pada tahapan processing ini adalah :

#### a. *Backbone : CSPDarknet53*

*Backbone network* merupakan tahapan yang mengacu pada ekstraksi fitur untuk mendeteksi objek. Pada tahapan ini menggunakan CSPDarknet53 yang terdiri dari dua gabungan yaitu CSP (*Cross Spatial Partial*) dan lapisan dasar *convolutional*. Berdasarkan perbandingan dengan CSPResNet50 dan EfficientNet-B3, CSPDarknet53 menghasilkan akurasi yang lebih baik pada pendeteksian objek.

#### b. *Neck : PANet*

Pada tahap ini ada penambahan layer dan nantinya akan dibentuk *pyramid feature* yang berguna untuk menghasilkan deteksi pada piksel yang lebih kecil sehingga meningkatkan proses segmentasi.

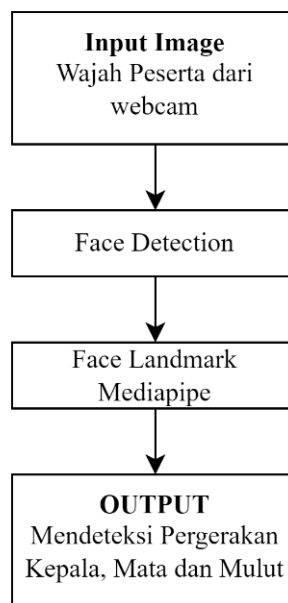
#### c. *Head*

Tahapan ini menghasilkan kotak deteksi yang digunakan untuk mendapatkan koordinat atau bagian dari *bounding box* objek dan menentukan *confidence score* pada suatu kelas. Pada tahap ini akan diterapkan fungsi IOU (*Intersection Over Union*) dan NMS (*Non Max Suppression*) untuk mencegah hasil kotak yang tumpang tindih.

### 4. *Output*

Setelah melewati *processing* akan ada pengecekan hasil dengan data validasi. Apabila hasilnya bagus akan menghasilkan *best model* yang akan diuji dengan data *testing*. Setelah selesai melakukan *testing* sistem akan menghasilkan *output* berupa deteksi objek kecurangan yang akan digunakan pada sesi ujian.





**Gambar 3.13** Arsitektur Proses *Face Landmark*

Berikut penjelasan mengenai arsitektur *face landmark Mediapipe* yang berada pada gambar 3.13 :

1. *Input*

Tahapan pertama ini memiliki proses untuk menginput wajah peserta dari *webcam* yang akan disimpan ke dalam sistem.

2. *Face Detection*

Setelah melewati proses *input*, selanjutnya akan dilakukan pendeteksian terhadap wajah. Proses ini sangat penting dalam *face landmark* karena akan berpengaruh untuk penempatan titik *landmark*.

3. *Face Landmark Mediapipe*

Setelah wajah terdeteksi, *Mediapipe* akan memproses wajah dan menemukan lokasi untuk penempatan titik *landmark*. Pada penempatan titik terdapat daerah yang merupakan titik kunci pada proses ini yaitu seperti mata, hidung, dan mulut. Penggunaan *face landmark* pada penelitian ini akan mendeteksi pergerakan mata, mulut, dan kepala. Gerakan yang dihasilkan oleh mata, mulut dan kepala diperoleh dari perubahan posisi pada setiap *frame*.

4. *Output*

Setelah melewati proses sebelumnya, *output* yang dihasilkan adalah untuk mendeteksi pergerakan mata, mulut, dan kepala. Hasil dari analisis pergerakan

mata, mulut, dan kepala akan dicatat menjadi *log*. Apabila mulut terbuka akan merekam *video* yang kemudian akan disimpan ke dalam *database*.

```
Function checkHeadMovement(face_landmarks):
    head_position <-- calculateHeadPosition(face_landmarks)
    return head_position
End

Function checkEyeMovement(face_landmarks):
    left_eye_position <-- calculateLeftEyePosition(face_landmarks)
    right_eye_position <-- calculateRightEyePosition(face_landmarks)
    return (left_eye_position, right_eye_position)
End

Function checkMouthMovement(face_landmarks):
    mouth_position <-- calculateMouthPosition(face_landmarks)
    return mouth_position
End

Function displayMovements(image, head_movement, eye_movement, mouth_movement):
    drawHeadMovement(image, head_movement)
    drawEyeMovement(image, eye_movement)
    drawMouthMovement(image, mouth_movement)
End
```

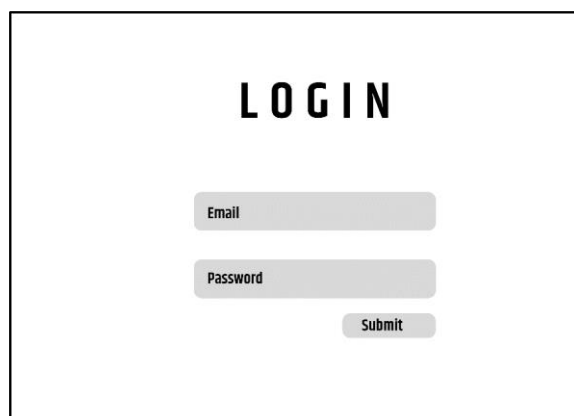
**Gambar 3.14 Pseudocode MediaPipe**

### 3.3 Perancangan Antarmuka Sistem

Pada tahapan ini, dilakukan perancangan tampilan antarmuka dari aplikasi pengawasan CBT Online yang diberi nama EduProctor. Sistem akan dibuat dalam bentuk aplikasi berbasis *website*. Berikut adalah rancangan tampilan – tampilan yang ada pada aplikasi EduProctor.

#### 3.3.1 Tampilan Awal

Halaman merupakan tampilan halaman pertama pada saat mengakses aplikasi EduProctor. Laman ini berisikan kolom untuk para peserta melakukan *login* untuk memasuki ke dalam aplikasi EduProctor dan berisikan informasi mengenai aplikasi EduProctor secara singkat.

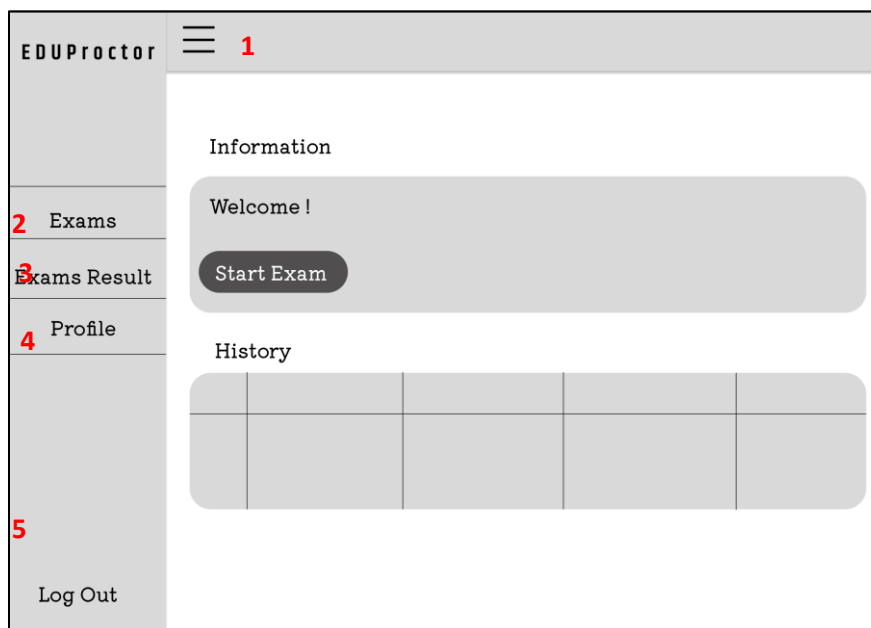


The image shows a login interface within a rectangular frame. At the top center, the word "LOGIN" is displayed in a large, bold, black, sans-serif font. Below the title, there are two input fields: the first is labeled "Email" and the second is labeled "Password". Both labels are in a small, black, sans-serif font and are positioned to the left of their respective input boxes. The input boxes are light gray with rounded corners. Below the "Password" input box, there is a "Submit" button, also with a light gray background and rounded corners, containing the word "Submit" in a small, black, sans-serif font.

**Gambar 3.15 Tampilan Login**

### 3.3.2 Tampilan Beranda

Tampilan ini adalah halaman beranda yang mana akan menjadi halaman pembuka pada saat berhasil melakukan *login*. Halaman ini berisikan informasi mengenai ujian para peserta. Halaman ini akan memunculkan daftar ujian dan juga informasi mengenai hasil ujian para peserta.



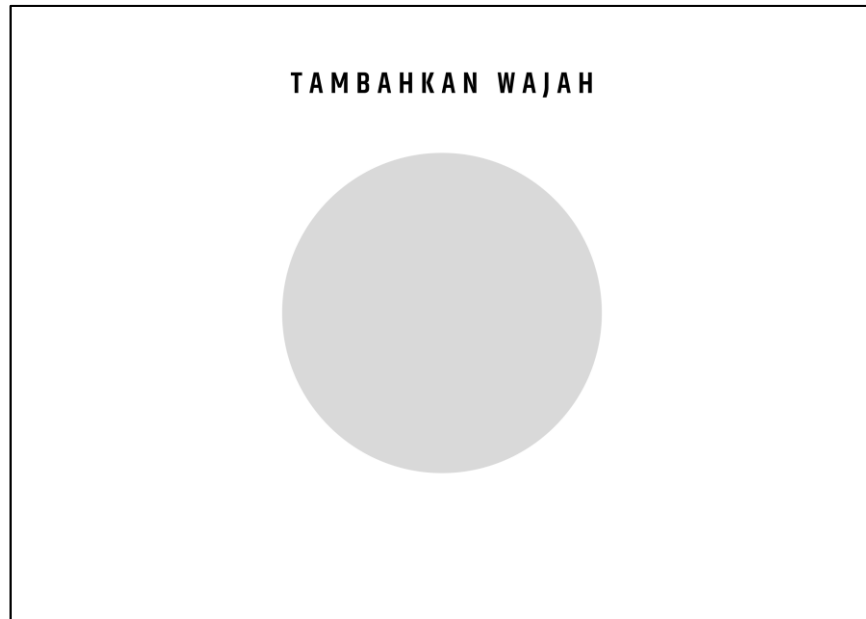
**Gambar 3.16 Tampilan Beranda**

Keterangan :

1. Label 1 adalah *button* untuk membuka dan menutup *sidebar*.
2. Label 2 adalah menu yang akan menampilkan halaman daftar ujian yang dimiliki oleh peserta.
3. Label 3 adalah menu yang akan menampilkan halaman hasil ujian para peserta.
4. Label 4 adalah menu yang akan menampilkan halaman *profile* para peserta.
5. Label 5 adalah menu untuk keluar dari *website* EduProctor.

### 3.3.3 Tampilan *Register Face*

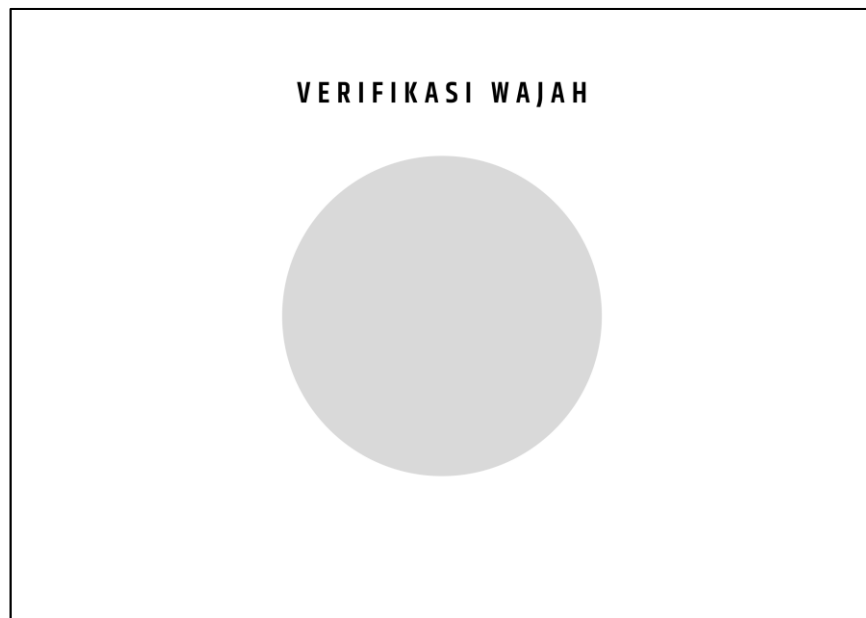
Tampilan ini adalah halaman untuk para peserta mendaftarkan wajah sebelum melakukan ujian. Halaman ini akan meminta akses kamera dari laptop peserta untuk mengambil *video* wajah peserta. Halaman ini akan muncul apabila peserta sudah menambahkan wajah pada halaman *profile*.



**Gambar 3.17 Tampilan *Register* Wajah**

#### **3.3.4 Tampilan *Face Recognition***

Tampilan ini adalah halaman untuk para peserta melakukan verifikasi wajah sebelum masuk ke halaman ujian. Pada halaman ini, wajah peserta akan diverifikasi berdasarkan data yang didapatkan pada saat *register face*.



**Gambar 3.18 Tampilan Verifikasi Wajah**

### 3.3.5 Tampilan Halaman Ujian

Tampilan ini adalah halaman yang akan diakses oleh para peserta ujian untuk melaksanakan ujian *online*. Halaman ini akan berisikan soal – soal ujian. Pada halaman ini akan dilakukan deteksi objek dan pergerakan kepala peserta secara *real time*.

EDUProctor

Time xx:xx

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

VIDEO WEBCAM

☐ Lorem ipsum dolor

☐ Lorem ipsum dolor

☐ Lorem ipsum dolor

☐ Lorem ipsum dolor

1 2 3

4

Submit

**Gambar 3.19 Tampilan Halaman Ujian**

### 3.3.6 Tampilan *Profile*

Tampilan ini adalah halaman untuk menampilkan mengenai data diri peserta. Pada halaman ini juga terdapat status wajah peserta dalam pendaftaran wajah untuk melakukan verifikasi pada saat ingin melakukan ujian.

EDUProctor

Exams

Exams Result

Profile

Log Out

PROFILE

User Data

NIM : XXXXXXXX

Nama : XXXXXXXX

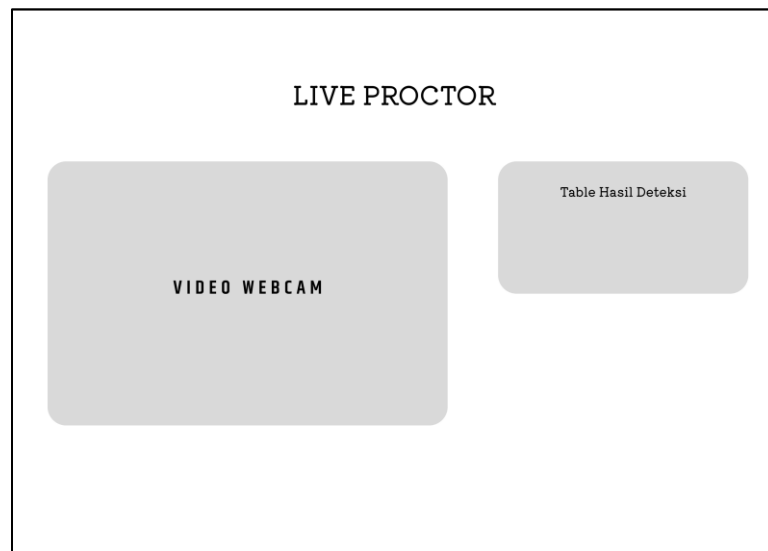
Face Status : NOT REGISTERED

ADD FACE

**Gambar 3.20 Tampilan *Profile***

### 3.3.7 Tampilan *Live Proctor*

Tampilan ini adalah halaman untuk *proctor* dapat melakukan pengawasan secara *real time*. Tampilan ini memunculkan *video webcam* dari para peserta ujian. Terdapat juga tabel untuk mencatat hasil dari deteksi *video webcam* peserta yang berfungsi untuk membantu *proctor* dalam melihat aktivitas mencurigakan yang dilakukan oleh peserta.



**Gambar 3.21 Tampilan *Live Proctor***

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1 Implementasi Sistem

Pada proses pembuatan aplikasi deteksi kecurangan pada CBT *online* menggunakan YOLOv8 dibutuhkan perangkat keras, perangkat lunak dan *library* pendukung agar dapat berjalan dengan baik. Adapun perangkat dan data yang digunakan dalam pembuatan aplikasi. Perangkat keras yang digunakan penulis dalam merancang aplikasi deteksi kecurangan pada CBT *online* adalah Laptop Acer Aspire E5-475G dengan spesifikasi sebagai berikut :

- 1) Processor : Intel(R) i5-7200U CPU @ 2.50GHz (4 CPUs), ~2.7GHz
- 2) RAM : 12288MB
- 3) Storage : 128GB SSD + 1TB HDD
- 4) GPU : NVIDIA GeForce 940MX
- 5) OS : Windows 10 Pro 64-bit

Perangkat lunak dan *library* yang digunakan penulis dalam merancang sistem deteksi kecurangan pada CBT *online* sebagai berikut :

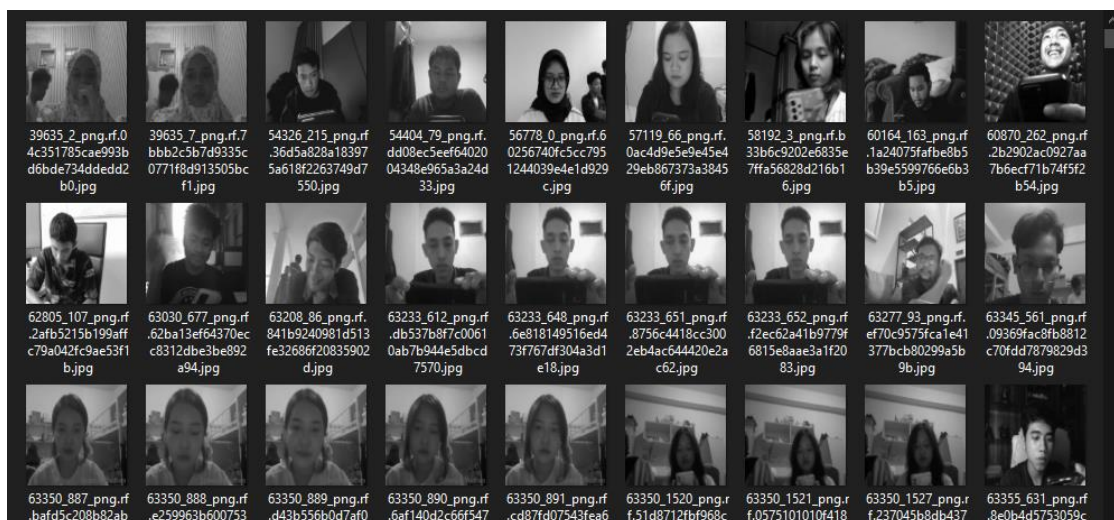
- 1) Kaggle
- 2) Visual Studio Code
- 3) Python 3.12.1
- 4) NextJs 14
- 5) FastAPI
- 6) MySQL
- 7) Roboflow
- 8) Ultralytics

## 4.2 Implementasi Data

Pada penelitian ini terdapat dua jenis data yang digunakan yaitu data objek kecurangan dan juga data wajah. Masing – masing data akan diproses menggunakan metode yang berbeda. Untuk data objek kecurangan akan diproses menggunakan YOLOv8 dan untuk data wajah peserta akan diproses menggunakan LBPH. Adapun data – data yang digunakan pada penelitian ini sebagai berikut :

### 4.2.1 Data Objek Kecurangan

Pada penelitian ini terdapat sebanyak 4119 data objek kecurangan yang telah dibagi menjadi 3 label, yakni *phone*, *book* dan OrangLain. Data ini diambil dengan menggunakan *webcam* dengan format .mp4 dan kemudian data dibagi menjadi *frame*. Setelah menjadi *frame* data akan diuji ke model YOLOv8n.pt, berikut adalah contoh beberapa kumpulan data objek kecurangan dapat dilihat pada gambar 4.1.



**Gambar 4.1 Data Objek Kecurangan**

### 4.2.2 Data Wajah

Selain data objek kecurangan, penelitian ini juga menggunakan data wajah yang akan dilatih untuk *face recognition* menggunakan LBPH. Data ini diambil menggunakan *webcam* dan kemudian akan langsung diproses menggunakan LBPH. Berikut adalah beberapa kumpulan data wajah dapat dilihat pada gambar 4.2.





**Gambar 4.2 Data Wajah**

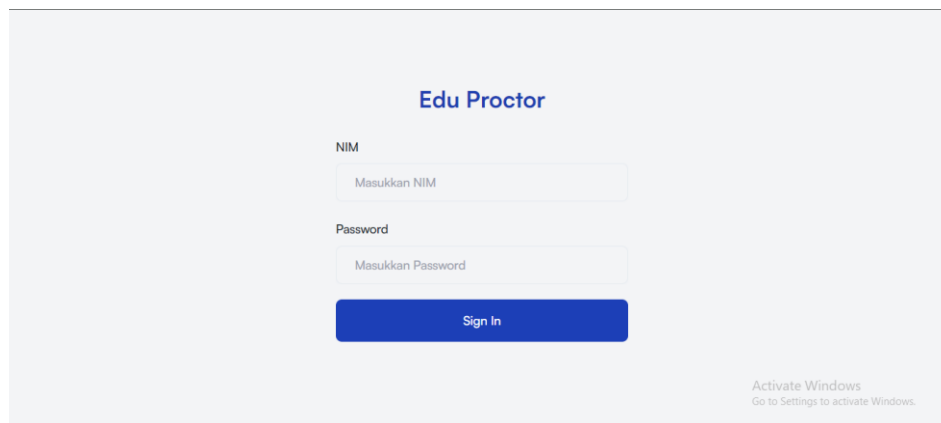
Gambar 4.2 merupakan contoh data yang telah melalui *preprocessing*. *Preprocessing* yang dilakukan adalah *grayscale*, *resize* dan *histogram equalization*. Setelah melewati *preprocessing* data akan diproses menggunakan LBPH dan *output* yang dihasilkan untuk mengenali wajah.

### 4.3 Implementasi Rancangan Antarmuka

Bagian ini akan menjelaskan mengenai rancangan antarmuka yang telah diimplementasi pada sisi *frontend* dari aplikasi deteksi kecurangan pada CBT *online* yang diberi nama EduProctor. EduProctor dibangun dengan menggunakan NextJs 14. Untuk bagian *Live Proctor* dibangun menggunakan streamlit. Berikut merupakan penjelasan mengenai setiap bagian dari implementasi rancangan antarmuka sistem.

#### 4.3.1 Tampilan Awal

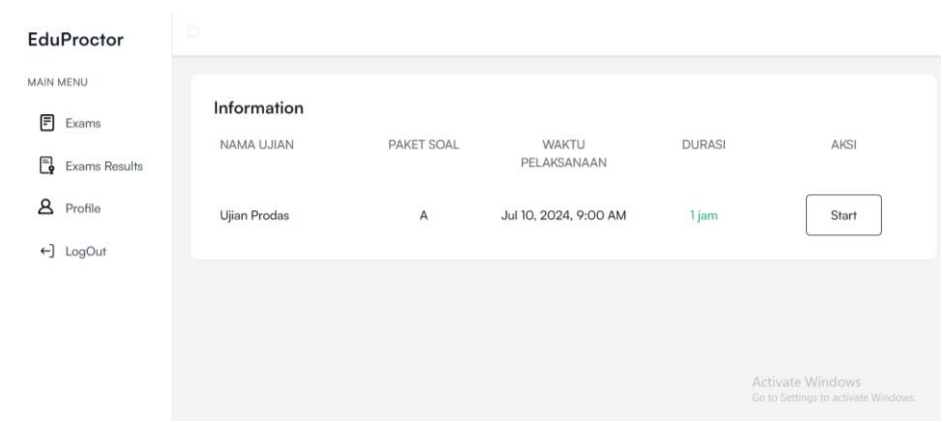
Halaman ini adalah tampilan awal pada saat mengakses EduProctor. Halaman ini berisi *form* untuk para peserta melakukan *login*. Halaman ini juga akan menampilkan nama dari aplikasi. Tampilan halaman awal dapat dilihat pada gambar 4.3.



**Gambar 4.3 Tampilan Awal**

#### 4.3.2 Tampilan Beranda

Halaman beranda adalah halaman utama yang menjadi halaman yang pertama kali dikunjungi oleh peserta. Pada halaman ini terdapat *main menu* pada *sidebar* yang berisi menu untuk peserta mengakses halaman ujian, halaman hasil ujian dan halaman *profile*. Selain itu terdapat satu tombol untuk peserta melakukan *logout* dari aplikasi EduProctor.



**Gambar 4.4 Tampilan Beranda**

#### 4.3.3 Tampilan *Register Face*

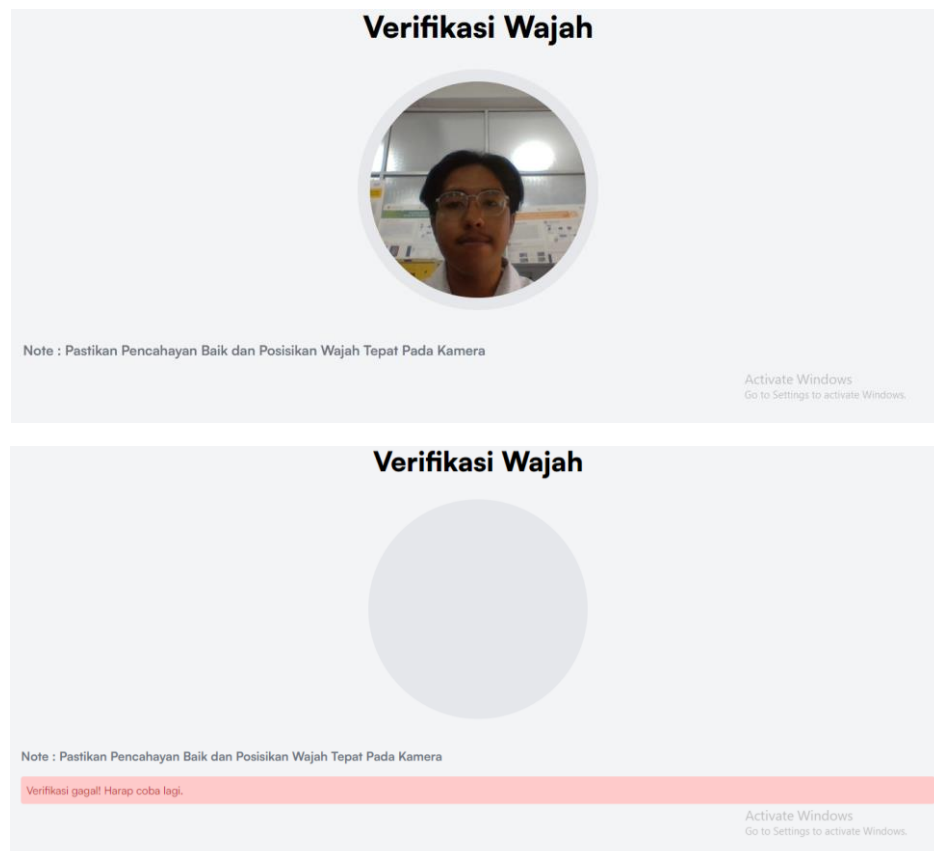
Tampilan ini adalah halaman untuk para peserta mendaftarkan wajah sebelum melakukan ujian. Halaman ini akan meminta akses kamera dari laptop peserta untuk mengambil *video* wajah peserta. Halaman ini akan muncul apabila peserta sudah menambahkan wajah pada halaman *profile*. Pada halaman ini peserta diwajibkan mengarahkan wajah tepat pada lingkaran yang telah disediakan dengan pencahayaan yang baik. Hal ini bertujuan agar data wajah lebih jelas pada saat *capture*.



**Gambar 4.5 Tampilan *Register Face***

#### **4.3.4 Tampilan *Face Recognition***

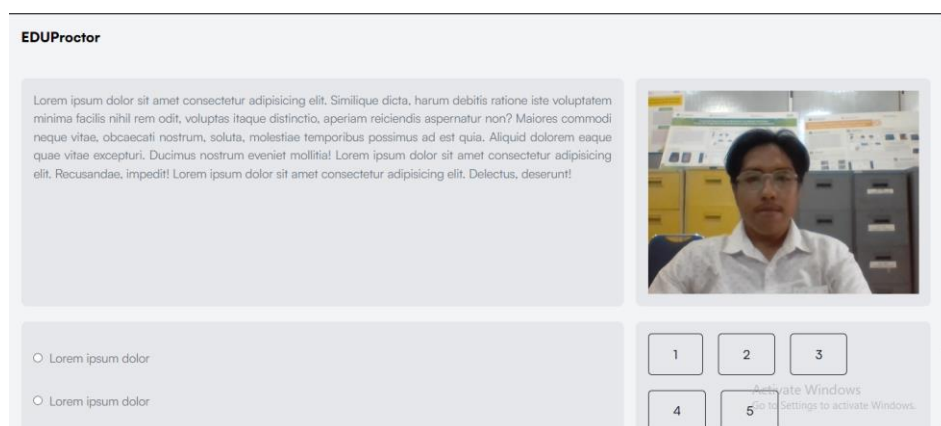
Tampilan ini adalah halaman untuk para peserta melakukan verifikasi wajah sebelum masuk ke halaman ujian. Pada halaman ini, wajah peserta akan diverifikasi berdasarkan data yang didapatkan pada saat *register face*. Ada beberapa hal yang harus diperhatikan pada saat verifikasi wajah seperti pencahayaan dan juga posisi wajah yang tepat pada kamera. Aplikasi akan mengeluarkan *alert* apabila wajah tidak dikenali oleh sistem.



**Gambar 4.6 Tampilan *Face Recognition***

#### 4.3.5 Tampilan Halaman Ujian

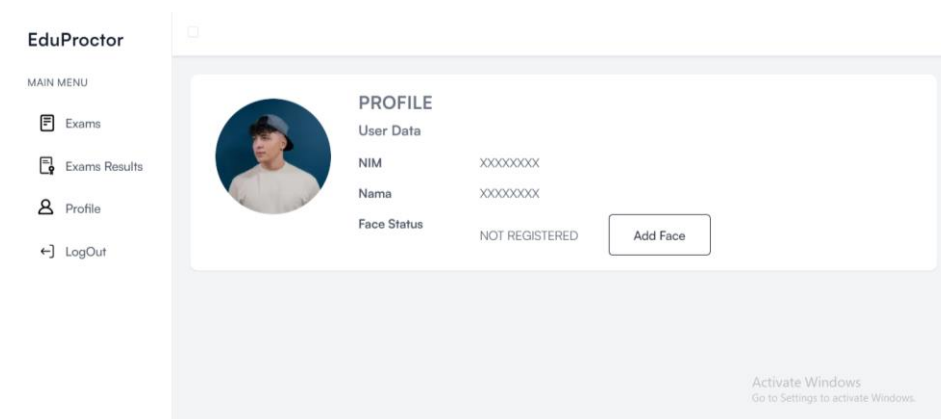
Tampilan ini adalah halaman yang akan diakses oleh para peserta ujian untuk melaksanakan ujian *online*. Halaman ini akan berisikan soal – soal ujian. Pada halaman ini akan dilakukan deteksi objek dan pergerakan kepala peserta secara *real time*. Terdapat *box* untuk menampilkan *webcam* para peserta yang sedang melakukan ujian.



**Gambar 4.7 Tampilan Halaman Ujian**

#### 4.3.6 Tampilan Profil

Tampilan ini adalah halaman untuk menampilkan mengenai data diri peserta. Pada halaman ini juga terdapat status wajah peserta dalam pendaftaran wajah untuk melakukan verifikasi pada saat ingin melakukan ujian.

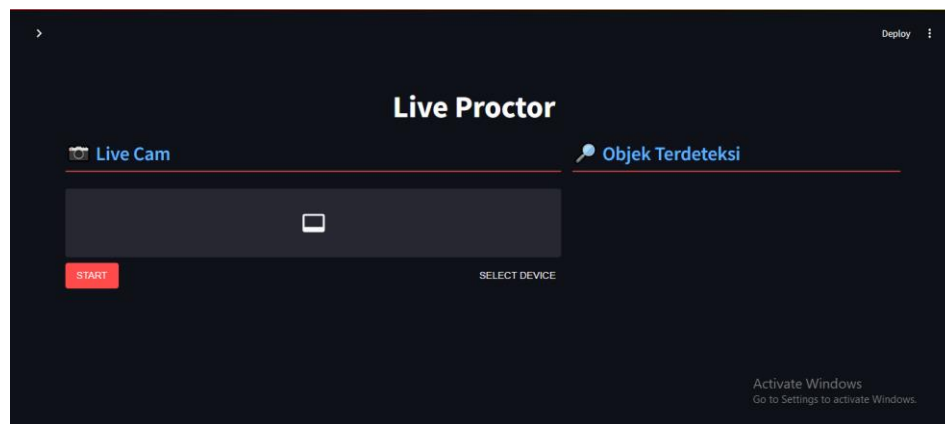


**Gambar 4.8 Tampilan Profile**

#### 4.3.7 Tampilan Live Proctor

Tampilan ini adalah halaman untuk *proctor* dapat melakukan pengawasan secara *real time*. Tampilan ini memunculkan *video webcam* dari para peserta ujian. Terdapat juga

tabel untuk mencatat hasil dari deteksi *video webcam* peserta yang berfungsi untuk membantu *proctor* dalam melihat aktivitas mencurigakan yang dilakukan oleh peserta.

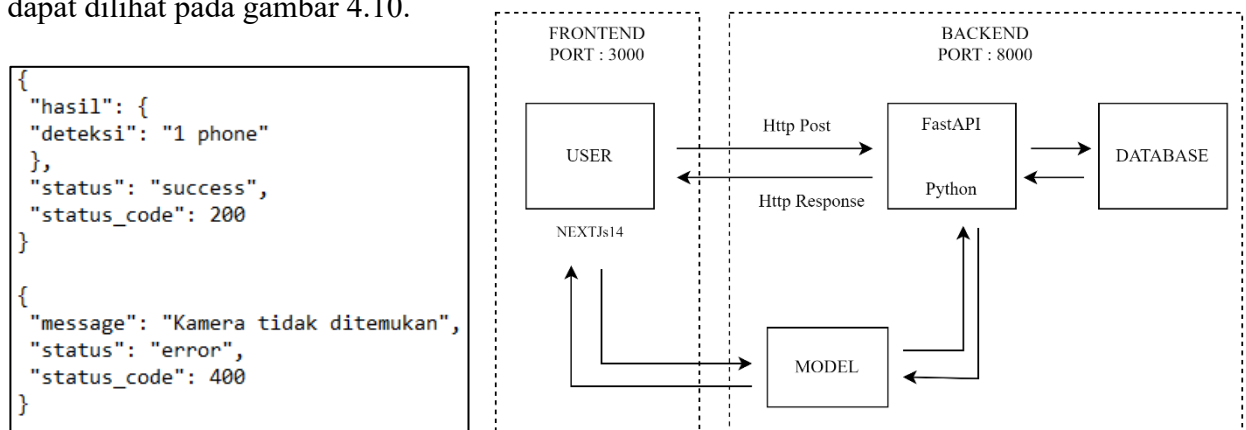


Gambar 4.9 Tampilan *Live Proctor*

### 4.3 Implementasi API

API Deteksi Objek Kecurangan hanya mengimplementasikan satu metode request, yaitu *post*. URL yang digunakan sebagai jalur di sisi *frontend* menuju API adalah `http://localhost/8000`. API yang digunakan dihasilkan dari penelitian ini dijalankan pada *local server*. *Request* ini akan dibatasi 3 detik untuk tiap pengiriman gambar ke *server* dari *input* yang diambil melalui *webcam*.

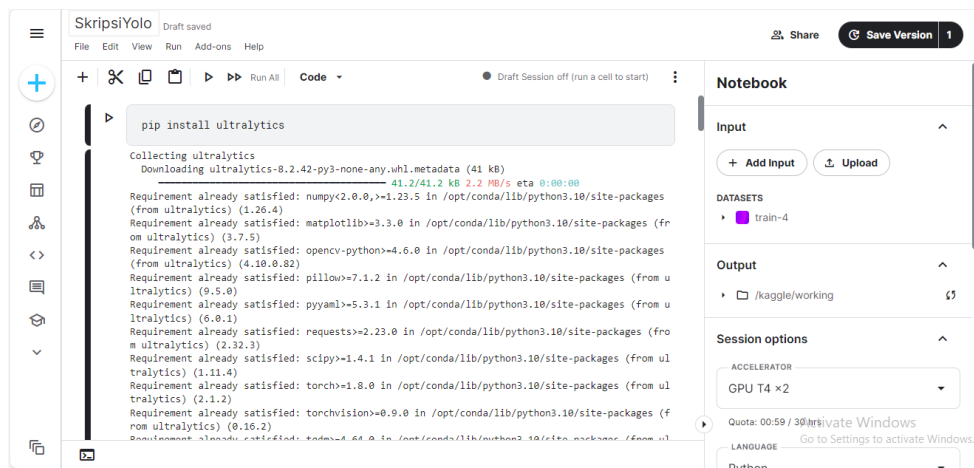
Setiap *request* yang diteruskan akan diproses oleh model yang telah dibuat dan akan disimpan ke dalam database. *Response* yang diberikan oleh API secara garis besar terbagi menjadi dua bagian, yaitu *response* untuk mengirim hasil deteksi dan *response* untuk mengirimkan pesan *error*. Adapun *response* berhasil dan *response error* yang dapat dilihat pada gambar 4.10.



Gambar 4.10 API

#### 4.4 Pelatihan Sistem

Pelatihan sistem deteksi objek kecurangan dilakukan dengan menggunakan algoritma YOLOv8. Pelatihan ini memiliki tujuan agar model yang dilatih memiliki kemampuan yang baik dalam mengenali objek. Proses pelatihan ini dilakukan menggunakan Kaggle Notebook yang dapat dilihat pada gambar 4.11.



**Gambar 4.11 Kaggle Notebook**

Pada pelatihan deteksi objek kecurangan menggunakan data sejumlah 4215 data yang terbagi menjadi tiga *class*, yaitu ; *phone*, OrangLain dan *book*. Sebelum masuk ke proses pelatihan data telah terlebih dulu melewati *preprocessing* yang dikerjakan pada roboflow. Pada pelatihan ini menggunakan jumlah *batch* 16 dan jumlah *epoch* 100. Hasil yang diperoleh dari nilai *batch* dan *epoch* yang sudah ditentukan mencapai mAP50(B) tertinggi pada *epoch* ke-81 dengan nilai *metrics* 0.98799. Hasil pelatihan dapat dilihat pada tabel 4.1.

**Tabel 4.1 Hasil Pelatihan Sistem**

epoch	box_loss	cls_loss	dfl_loss	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
1	1.0371	2.2035	1.2796	0.71835	0.48302	0.62443	0.44857
2	1.0403	1.5285	1.256	0.61335	0.55477	0.57141	0.35015
3	1.0349	1.3084	1.2551	0.68291	0.6688	0.71527	0.48026
4	0.99429	1.1594	1.2289	0.80958	0.71682	0.79615	0.56795
5	0.95077	1.0345	1.201	0.87376	0.87037	0.91713	0.68968
6	0.90397	0.91108	1.168	0.87943	0.86159	0.91067	0.70638

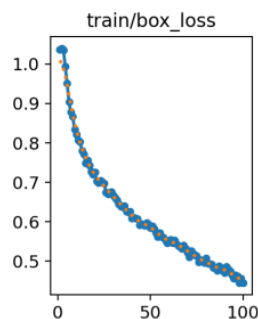
epoch	box_loss	cls_loss	dfl_loss	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
7	0.87811	0.86232	1.1524	0.91051	0.82571	0.9258	0.72799
8	0.86663	0.85698	1.1492	0.91171	0.86918	0.94448	0.76671
9	0.8349	0.80039	1.1254	0.92698	0.89169	0.94452	0.74648
10	0.82101	0.77188	1.1258	0.84352	0.84552	0.91337	0.75364
11	0.80704	0.74802	1.1108	0.90145	0.90811	0.9456	0.75482
12	0.80238	0.73806	1.1064	0.95516	0.89678	0.95934	0.76633
13	0.78037	0.69828	1.1016	0.94812	0.93016	0.96054	0.80116
14	0.77206	0.68177	1.1032	0.92189	0.91914	0.95171	0.78308
15	0.74853	0.6565	1.0819	0.92772	0.92548	0.96371	0.79957
16	0.75531	0.65166	1.0892	0.95102	0.9411	0.97031	0.81116
17	0.74308	0.64187	1.0767	0.94203	0.93931	0.9667	0.81503
18	0.72745	0.6238	1.0669	0.90525	0.88627	0.94724	0.75804
19	0.72	0.61749	1.0629	0.95439	0.93195	0.96536	0.81107
20	0.72538	0.60862	1.0667	0.9405	0.92092	0.9655	0.81217
21	0.70305	0.59293	1.0551	0.95188	0.94811	0.96563	0.80402
22	0.69817	0.58307	1.0528	0.94948	0.94819	0.97227	0.82852
23	0.70467	0.57436	1.0582	0.93068	0.94167	0.96506	0.81303
24	0.70049	0.57675	1.0551	0.92683	0.89763	0.95197	0.79368
25	0.69739	0.56814	1.0484	0.95569	0.95449	0.97591	0.82844
26	0.67518	0.56219	1.0412	0.94771	0.94941	0.97397	0.83538
27	0.6704	0.54216	1.0388	0.95506	0.95668	0.97492	0.83469
28	0.6738	0.54657	1.0472	0.94137	0.95199	0.97773	0.82485
29	0.67568	0.53322	1.0425	0.96144	0.94372	0.97319	0.82644
30	0.6653	0.5204	1.0355	0.97069	0.95329	0.97474	0.83211
31	0.66089	0.52128	1.0293	0.94214	0.94959	0.9655	0.81525
32	0.65416	0.51027	1.0229	0.95296	0.95805	0.97146	0.83573
33	0.6441	0.50403	1.0276	0.96489	0.94645	0.96885	0.83603
34	0.63951	0.50268	1.0196	0.95674	0.9491	0.97089	0.83328
35	0.63764	0.4943	1.0218	0.95826	0.95898	0.97818	0.8435
36	0.64065	0.49738	1.0266	0.95499	0.94847	0.97529	0.83992
37	0.62535	0.48812	1.0086	0.96767	0.94904	0.97292	0.84379
38	0.62447	0.48255	1.0197	0.96171	0.95668	0.97494	0.84317
39	0.62582	0.47526	1.0137	0.96719	0.95531	0.97921	0.85042
40	0.60896	0.47202	1.007	0.95535	0.94215	0.97679	0.84741
41	0.6102	0.46848	1.0072	0.96287	0.95978	0.97895	0.85313
42	0.60936	0.47047	1.0083	0.94974	0.95937	0.97677	0.84694
43	0.60706	0.45865	0.99962	0.9538	0.96547	0.97482	0.84698
44	0.59159	0.45956	1.0001	0.96419	0.96285	0.98095	0.85305

epoch	box_loss	cls_loss	dfl_loss	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
45	0.59474	0.45128	0.99936	0.95792	0.95131	0.97491	0.85092
46	0.59336	0.44704	0.99122	0.96513	0.94902	0.9778	0.85927
47	0.59063	0.43679	0.9939	0.97192	0.95971	0.97888	0.85979
48	0.59751	0.44149	0.99559	0.96752	0.96026	0.97705	0.86692
49	0.59264	0.4364	0.99677	0.96216	0.96402	0.97671	0.85683
50	0.58454	0.42968	0.99052	0.969	0.96	0.97711	0.8646
51	0.58879	0.4266	0.9926	0.9596	0.96297	0.97917	0.86973
52	0.58202	0.42571	0.98901	0.9701	0.96218	0.9784	0.86076
53	0.5718	0.4145	0.98796	0.96283	0.95816	0.98079	0.85685
54	0.5616	0.41351	0.9857	0.97168	0.96203	0.98069	0.86482
55	0.57159	0.42104	0.99348	0.96917	0.95861	0.98268	0.85675
56	0.56139	0.40513	0.98217	0.97067	0.94692	0.97834	0.86133
57	0.56036	0.39886	0.97848	0.97031	0.96259	0.9806	0.86674
58	0.55321	0.39833	0.97524	0.97553	0.96149	0.97985	0.86772
59	0.54588	0.40374	0.97685	0.97367	0.9538	0.98238	0.86986
60	0.55332	0.39834	0.97614	0.96354	0.96863	0.98546	0.87485
61	0.55485	0.39492	0.97715	0.97272	0.96432	0.97824	0.86393
62	0.54603	0.393	0.97069	0.97364	0.9657	0.98081	0.8712
63	0.55251	0.39686	0.97857	0.96456	0.96558	0.9797	0.86457
64	0.54477	0.39843	0.98245	0.96845	0.96369	0.97747	0.86937
65	0.53887	0.38329	0.96746	0.96667	0.96579	0.97998	0.8728
66	0.53593	0.38195	0.96219	0.96583	0.96088	0.97781	0.87514
67	0.53214	0.37856	0.96791	0.97751	0.9592	0.97957	0.87698
68	0.53969	0.37578	0.97052	0.9746	0.96291	0.98238	0.87386
69	0.53187	0.37634	0.97293	0.97179	0.96348	0.97864	0.86697
70	0.5219	0.37506	0.96702	0.96861	0.96759	0.9783	0.87426
71	0.51112	0.35907	0.95814	0.97239	0.95939	0.98339	0.87738
72	0.52661	0.36806	0.96874	0.9737	0.97065	0.98516	0.88007
73	0.51859	0.369	0.96455	0.96692	0.96401	0.98404	0.87926
74	0.51619	0.36766	0.96149	0.97456	0.97085	0.9834	0.87969
75	0.51383	0.35653	0.96214	0.97341	0.96959	0.98503	0.88272
76	0.49713	0.34886	0.95298	0.9785	0.96729	0.98247	0.87907
77	0.5034	0.33822	0.95138	0.97912	0.96022	0.98307	0.88629
78	0.49973	0.34612	0.95722	0.97469	0.96962	0.98228	0.88238
79	0.4965	0.34717	0.95483	0.97896	0.96561	0.98477	0.88537
80	0.50707	0.34228	0.96103	0.96992	0.96921	0.98584	0.88545
81	0.49503	0.34206	0.95715	0.97389	0.97343	0.98799	0.88594
82	0.49399	0.33961	0.95618	0.97286	0.96591	0.98615	0.88778



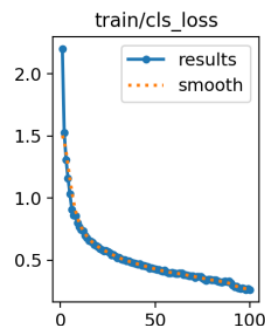
epoch	box_loss	cls_loss	dfl_loss	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
83	0.49174	0.33329	0.94921	0.97764	0.96693	0.98436	0.88363
84	0.48377	0.32928	0.95	0.98142	0.96975	0.9869	0.89252
85	0.48314	0.32313	0.94672	0.97156	0.9796	0.98475	0.89034
86	0.47686	0.32837	0.94828	0.97341	0.97551	0.98612	0.88753
87	0.48759	0.3347	0.95051	0.97142	0.97586	0.98556	0.88756
88	0.48338	0.32612	0.94488	0.96512	0.97623	0.98464	0.89083
89	0.47621	0.33097	0.94997	0.97747	0.97478	0.98572	0.89246
90	0.47012	0.3179	0.94886	0.97203	0.97776	0.98754	0.89131
91	0.48651	0.29775	0.96042	0.97257	0.97857	0.98498	0.88669
92	0.48101	0.28665	0.95702	0.98019	0.97003	0.98606	0.88845
93	0.4748	0.2894	0.94976	0.97254	0.97549	0.98173	0.88628
94	0.46533	0.27908	0.94345	0.97613	0.973	0.98486	0.89004
95	0.4564	0.27427	0.94539	0.97644	0.97174	0.98371	0.88956
96	0.46493	0.27808	0.95114	0.97506	0.97458	0.985	0.89306
97	0.46153	0.27429	0.94406	0.9739	0.97456	0.98382	0.89126
98	0.44513	0.26206	0.93551	0.97014	0.97403	0.98433	0.89433
99	0.45597	0.27208	0.94308	0.97221	0.97478	0.98371	0.89329
100	0.44455	0.26439	0.93356	0.97891	0.97263	0.98426	0.89484

Berdasarkan tabel 4.1, terjadi peningkatan kemampuan model untuk mendeteksi objek kecurangan. Hal ini dapat dilihat dari penurunan angka pada kolom *box\_loss*, kondisi ini berbanding terbalik dengan kemampuan model, semakin kecil *box\_loss* artinya kemampuan model semakin baik dalam memprediksi koordinat *bounding box* (kotak batas) dari objek yang terdeteksi. Grafik penurunan *box\_loss* dapat dilihat pada gambar 4.12.



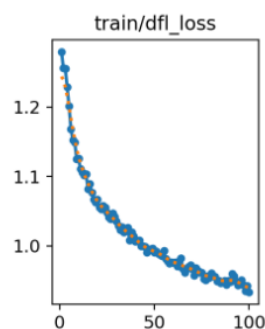
**Gambar 4.12 Grafik Box Loss**

Selain *box\_loss* adanya penurunan pada *cls\_loss* juga meningkatkan kemampuan model dalam mengklasifikasikan objek terhadap label hasil anotasi. Grafik penurunan *cls\_loss* dapat dilihat pada gambar 4.13.



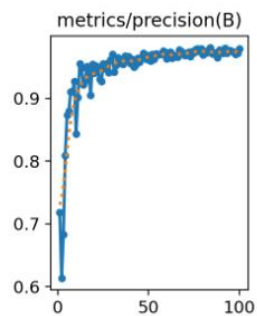
**Gambar 4.13 Grafik Cls Loss**

DFL (*Distribution Focal Loss*) adalah variasi dari *focal loss* yang digunakan untuk meningkatkan performa deteksi objek dalam kasus ketidakseimbangan kelas. DFL berfokus pada sampel data yang sulit atau sering mengalami kesalahan pada saat pengklasifikasian. Penurunan nilai DFL menandakan peningkatan kemampuan model untuk mendeteksi model yang sulit untuk diklasifikasikan. Grafik DFL dapat dilihat pada gambar 4.14.



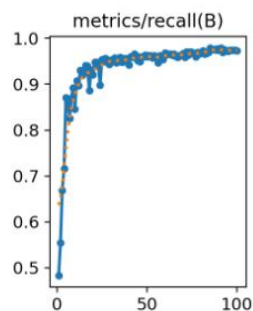
**Gambar 4.14 Grafik DFL**

Selain *box\_loss*, *cls\_loss* dan *dfi\_loss*, hal yang menjadi perhatian dalam peningkatan kemampuan model YOLOv8 adalah meningkatnya nilai *precision*. *Precision* memiliki fungsi untuk mengukur sejauh mana hasil deteksi dari model bernilai benar dalam deteksi positif. Grafik *precision* dapat dilihat pada gambar 4.15.



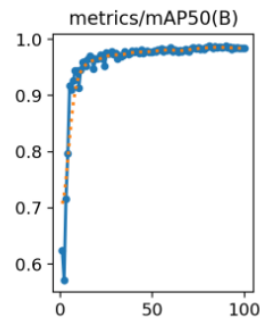
**Gambar 4.15 Grafik Precision**

Peningkatan dari *recall* juga perlu diperhatikan dalam menilai kemampuan model YOLOv8 yang dihasilkan dari pelatihan. *Recall* berfungsi untuk mengukur sejauh mana model mampu menemukan semua objek yang sebenarnya ada pada gambar. Peningkatan nilai *recall* menunjukkan bahwa *output* deteksi semakin membaik. Grafik *recall* dapat dilihat pada gambar 4.16.



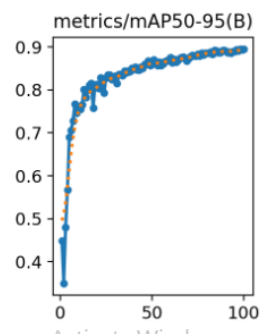
**Gambar 4.16 Grafik Recall**

*Mean Average Precision* at IoU 50% atau mAP50 adalah sebuah metrik yang menggabungkan *precision* dan *recall* dengan menggunakan IoU (*Intersection over Union*) sebesar 50% sebagai ambang batas. Nilai mAP50 diperoleh dengan cara menghitung rata-rata presisi pada berbagai ambang batas IoU yang bervariasi mulai dari 50% hingga 95% dengan peningkatan interval 5% yang tetap. Peningkatan nilai mAP50 menunjukkan gambaran peningkatan kemampuan model yang dihasilkan dalam menemukan objek dengan nilai *precision* yang tinggi pada berbagai aspek tumpang tindih. Grafik mAP50 dapat dilihat pada gambar 4.17.



**Gambar 4.17 Grafik mAP50**

Selain mAP50, ada juga mAP50-95 (*Mean Average Precision* dari IoU 50% hingga IoU 95%). Perbedaan antara mAP50 dan mAP50-95 adalah mAP50-95 mengukur rata-rata presisi dari rentang IoU 50% hingga 95% dengan interval 5%. Interval ini berarti bahwa mAP50-95 menghitung presisi secara bertahap, dimulai dari *true positive* ketika IoU melebihi 50%, kemudian *true positive* ketika IoU melebihi 55%, dan seterusnya dengan interval 5%. Berdasarkan Tabel 4.1, terdapat peningkatan nilai mAP50-95 dari epoch awal hingga epoch ke-100. Peningkatan mAP50-95 dapat dilihat pada Gambar 4.17, di mana sumbu x menunjukkan epoch dan sumbu y menunjukkan mAP50-95 (B).



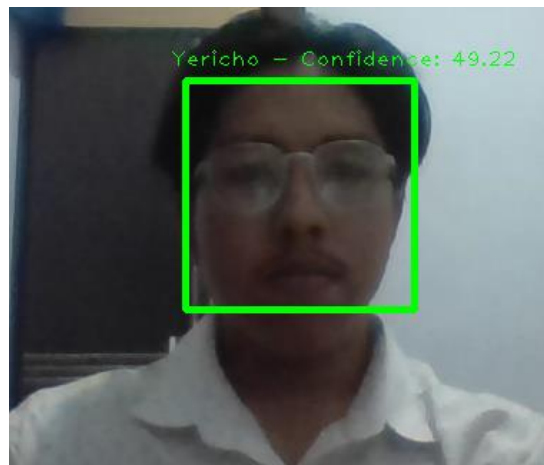
**Gambar 4.18 Grafik mAP50-95**

## 4.5 Pengujian Sistem

### 4.5.1 Pengujian Sistem *Face Recognition*

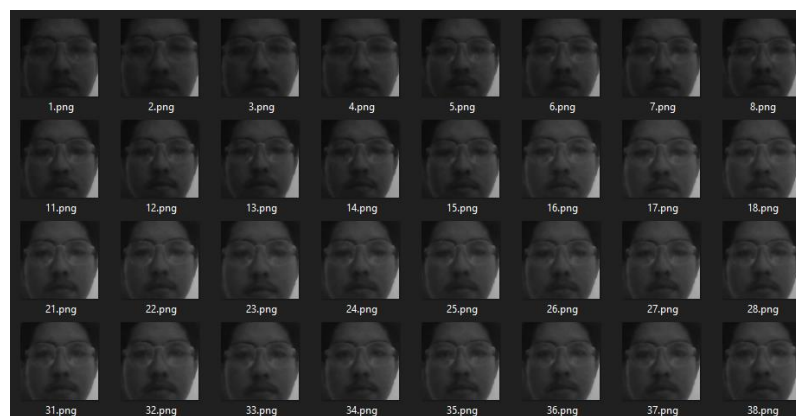
Pengenalan wajah (*face recognition*) menggunakan algoritma LBPH (Local Binary Patterns Histogram) telah berhasil diimplementasikan. Pengenalan wajah ini telah melalui proses *pre-processing* sebelum akhirnya diproses menggunakan LBPH. Pada gambar 4.19 terlihat bahwa sistem pengenalan wajah berhasil mendeteksi dan mengenali wajah dengan nama "Yericho" dengan tingkat kepercayaan (*confidence*)

sebesar 49.22%. Proses ini melibatkan pengambilan fitur wajah dari gambar, konversi ke histogram, dan pencocokan dengan data yang sudah ada.



**Gambar 4.19 Hasil *Face Recognition***


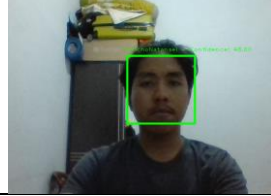
Adapun dataset yang telah dikumpulkan sebelum akhirnya sistem dapat mengenali wajah dapat dilihat pada gambar 4.20. Dataset ini akan disimpan di dalam sebuah folder. Folder ini akan berisi 200 gambar hasil tangkapan layar dari menu *register face*.



**Gambar 4.20 Dataset Wajah**

Adapun dilakukannya pengujian pada sistem *face recognition* dalam keadaan pencahayaan yang terang begitu juga dengan keadaan jarak 40 cm dan 60 cm dari *webcam* laptop. Hasil pengujian ini bisa dilihat pada tabel 4.2.



**Tabel 4.2 Pengujian Pencahayaan dan Jarak *Face Recognition***



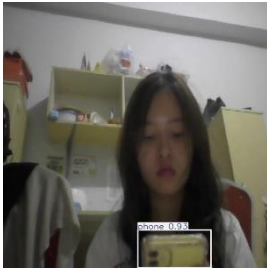


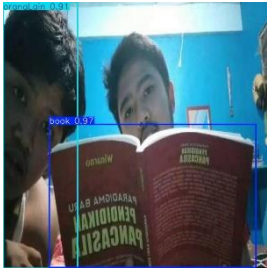
Gambar	Jarak	Hasil ( <i>confidence</i> )	Aktual	Waktu
	40 CM	YerichoNatanael (55.79%)	Benar	50ms
	60 CM	YerichoNatanael (48%)	Benar	50ms







#### 4.5.2 Pengujian Sistem *Object-Detection*

Tahapan pengujian sistem yang pertama dilakukan dengan menguji model YOLOv8 terhadap potongan-potongan gambar untuk mendeteksi objek kecurangan. Data gambar yang digunakan tanpa melewati tahapan *preprocessing*. Adapun proses pengujian model YOLOv8 dapat dilihat pada Tabel 4.3.





**Tabel 4.3 Pengujian Sistem YOLOv8**

Gambar	Deteksi	Aktual	Status	Waktu
	1 OrangLain	1 Orang Lain	Benar	10ms
	1 OrangLain	1 Orang Lain	Benar	10ms

Gambar	Deteksi	Aktual	Status	Waktu
	1 OrangLain	1 Orang Lain	Benar	10ms
	1 Phone	1 Phone	Benar	10ms
	1 Phone	1 Phone	Benar	10ms
	2 OrangLain	1 Orang Lain	Benar	10ms
	1 Phone	1 Phone	Benar	10ms
	1 Book 1 OrangLain	1 Book 1 Orang Lain	Benar	10ms

Gambar	Deteksi	Aktual	Status	Waktu
	1 Book	1 Book	Benar	10ms
	1 OrangLain 1 Phone	1 Orang Lain 1 Phone	Benar	10ms
	1 Book	1 Book	Benar	10ms
	1 Book	1 Book	Benar	10ms
	1 Phone 1 OrangLain	1 Phone 1 Orang Lain	Benar	10ms
	1 Phone 1 Book 1 OrangLain	1 Phone 1 Book 1 Orang Lain	Benar	10ms



Gambar	Deteksi	Aktual	Status	Waktu
	2 Phone 1 OrangLain	2 Phone 1 Orang Lain	Benar	10ms
	1 Phone 1 OrangLain	1 Phone 2 Orang Lain	Salah	10ms
	1 Book	1 Book	Benar	10ms
	1 OrangLain	1 Orang Lain	Benar	10ms
	1 Phone	1 Phone	Benar	10ms

Tabel 4.3 merupakan contoh hasil pengujian terhadap model YOLOv8 dalam mendeteksi objek kecurangan. Total data yang digunakan dalam pengujian model YOLOv8 adalah 411 *sample* data dengan rata – rata waktu deteksi 10ms. Berikut pada tabel 4.4 merupakan hasil perhitungan dari hitungan *confusion matrix*.

**Tabel 4.4 Confusion Matrix**

	TP	FP	FN
<i>Phone</i>	136	0	1
<i>OrangLain</i>	127	8	3
<i>Book</i>	136	2	1
<b>Total</b>	<b>399</b>	<b>10</b>	<b>5</b>

Berikut adalah perhitungan untuk mencari *precision*, *recall* dan *F1-score* :

1) *Precision*

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

$$Precision\ Phone = \frac{136}{136} \times 100\% = 100\%$$

$$Precision\ OrangLain = \frac{127}{135} \times 100\% = 94\%$$

$$Precision\ Book = \frac{136}{138} \times 100\% = 98.6\%$$

2) *Recall*

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

$$Recall\ Phone = \frac{136}{137} \times 100\% = 99.3\%$$

$$Recall\ OrangLain = \frac{127}{130} \times 100\% = 97.7\%$$

$$Recall\ Book = \frac{136}{137} \times 100\% = 99.3\%$$

3) *F1-score*

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

$$F1 - Score Phone = 2 \times \frac{(99.3\% \times 100\%)}{(99.3\% + 100\%)} = 99.7\%$$

$$F1 - Score OrangLain = 2 \times \frac{(97.7\% \times 94\%)}{(97.7\% + 94\%)} = 95.7\%$$

$$F1 - Score Book = 2 \times \frac{(99.3\% \times 98.6\%)}{(99.3\% + 98.6\%)} = 98.9\%$$

Dari seluruh penghitungan yang telah dilakukan, maka dapat dilihat hasil yang diperoleh pada tabel 4.5.

**Tabel 4.5 Nilai *Precision*, *Recall* dan *F1-Score***

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
<b><i>Phone</i></b>	100%	99.3%	99.7%
<b><i>OrangLain</i></b>	94%	97.7%	95.7%
<b><i>Book</i></b>	98.6%	99.3%	98.9%
<b><i>Average</i></b>	97.5%	98.7%	98.1%

Berdasarkan seluruh proses uji sistem yang telah dilakukan pada model aplikasi deteksi kecurangan menggunakan YOLOv8 maka didapatkan nilai akurasi sebagai berikut :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

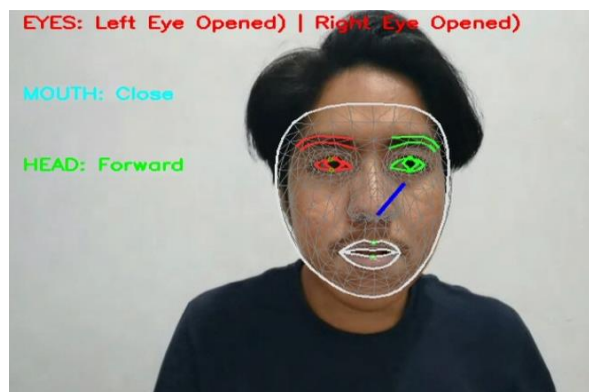
$$Accuracy = \frac{399}{411} \times 100\% = 97\%$$

Dengan nilai akurasi sebesar 97%, maka dapat dikatakan bahwa sistem dapat melakukan deteksi pada 3 jenis objek kecurangan secara *realtime* dengan baik di mana sistem dapat dijalankan pada *website*. Di sisi lain, sistem juga menghasilkan beberapa kesalahan dalam prediksi yang menunjukkan bahwa metode yang digunakan masih

belum sempurna dan memiliki kekurangan dalam mendeteksi objek kecurangan. Kesalahan tersebut disebabkan oleh kurangnya jumlah dan variasi data yang dilatih sehingga apabila adanya kemiripan objek, sistem akan mendeteksi sebagai objek kecurangan.

#### 4.5.3 Pengujian Sistem *Face Landmarks*

Pengujian *face landmark* menggunakan Mediapipe untuk melacak pergerakan wajah peserta dalam ujian. Pada gambar 4.21 dapat dilihat terdapat 468 titik yang akan dipetakan ke dalam wajah peserta untuk mencakup fitur wajah yang digunakan dalam melihat pergerakan seperti, mata, hidung, mulut dan bibir. Implementasi titik-titik ini dilakukan dengan menggunakan algoritma pembelajaran mesin yang dilatih untuk mendeteksi dan melacak fitur wajah secara *real-time*, bahkan dalam kondisi pencahayaan dan posisi wajah yang beragam.



**Gambar 4.21 *Face Landmark* Mediapipe**

Berikut penjelasan keterangan yang ada pada gambar 4.21 sebagai berikut :

*a. Eyes*

Pada bagian ini akan menampilkan hasil deteksi pergerakan mata yaitu terbuka atau tertutup. Bagian ini juga mendeteksi kedua bagian mata yaitu kanan dan kiri.

*b. Mouth*

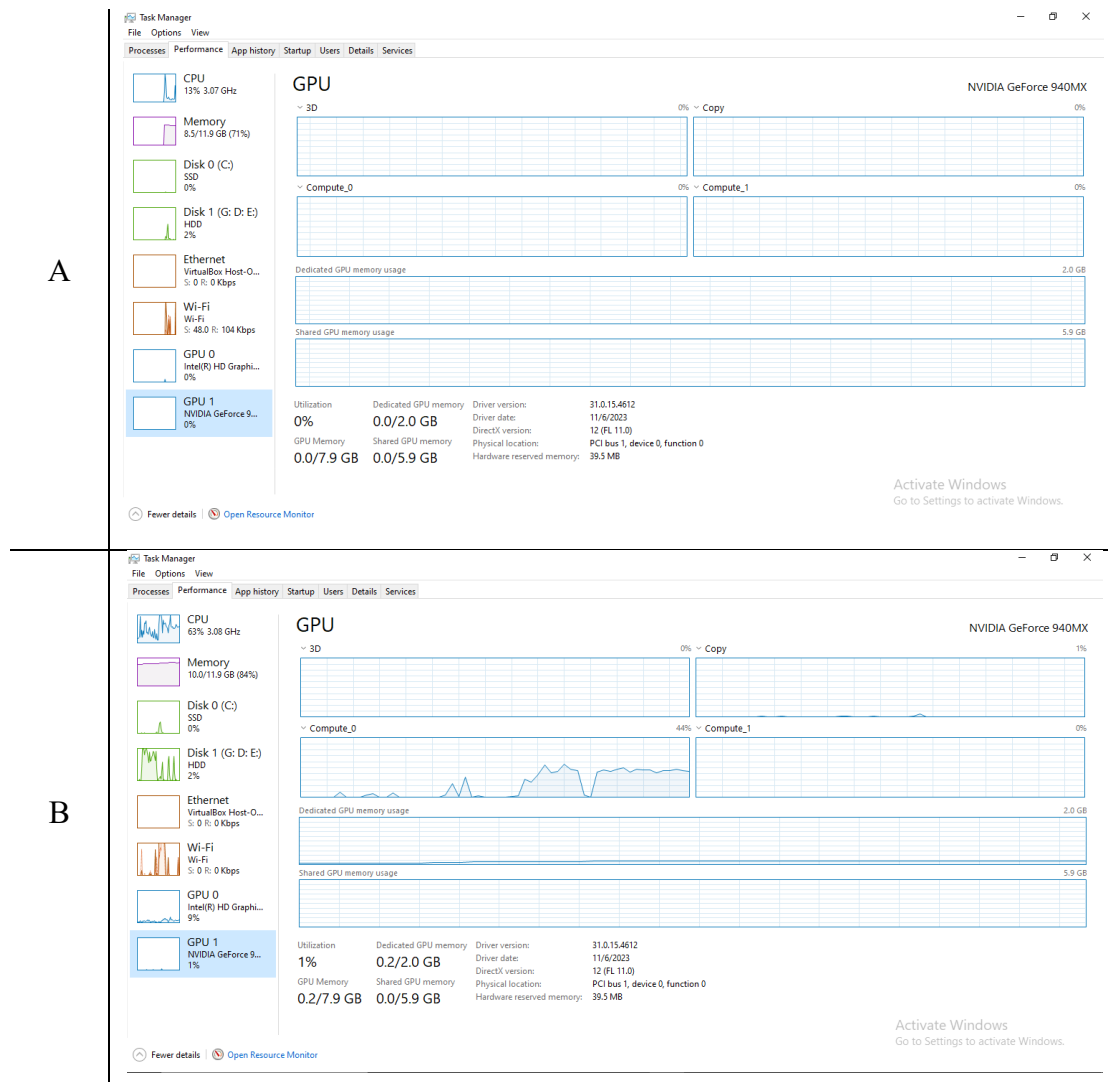
Pada bagian ini akan menampilkan hasil deteksi mulut apakah dalam keadaan terbuka atau tidak.

*c. Head*

Pada bagian ini akan menampilkan hasil deteksi berupa arah kepala yaitu atas, bawah, depan, kanan dan kiri.

Berikut ini merupakan perbandingan dari penggunaan CPU dan GPU pada saat aplikasi dijalankan. Bisa dilihat pada tabel 4.6.

**Tabel 4.6 Penggunaan CPU & GPU**



Pada tabel 4.6 bisa dilihat adanya peningkatan pada penggunaan CPU dan GPU, pada gambar A adalah kondisi sebelum aplikasi berjalan dan gambar B adalah kondisi saat aplikasi berjalan. Pada gambar A terlihat penggunaan CPU sebesar 13% dan penggunaan GPU 1 sebesar 0% dengan nilai *compute\_0* sebesar 0%. Pada gambar B penggunaan CPU sebesar 63% dan penggunaan GPU 1 sebesar 1% dengan nilai *compute\_0* sebesar 44%.

## BAB 5

### KESIMPULAN

#### 5.1 Kesimpulan

Terdapat beberapa kesimpulan yang diperoleh dari keseluruhan hasil pengujian pada penelitian untuk membuat aplikasi deteksi objek kecurangan pada CBT *online* menggunakan YOLOv8, *face recognition* menggunakan LBPH dan *face landmark* menggunakan Mediapipe, yaitu :

- 1) Penggunaan algoritma *You Only Look Once* (YOLO) versi 8 mampu mendeteksi 3 jenis objek kecurangan yang terdiri dari *phone*, *person* dan *book* secara *realtime*.
- 2) Pendeteksian menggunakan algoritma *You Only Look Once* (YOLO) versi 8 memperoleh hasil akurasi yang tergolong sangat baik sebesar 97% dengan nilai rata-rata *precision* sebesar 97.5%, *recall* sebesar 98.7%, dan *F1-Score* sebesar 98.1%.
- 3) Penggunaan LBPH dalam mengenali wajah peserta dengan baik di keadaan pencahayaan yang terang dengan rata – rata *confidence* 0.6 dengan 200 gambar wajah per *user* yang sudah melewati *preprocessing*.
- 4) Penggunaan Mediapipe dengan 468 titik yang disematkan ke setiap wajah *user* berjalan dengan baik dalam menentukan pergerakan postur wajah dan kepala. Pencahayaan sangat penting dalam membantu Mediapipe bekerja untuk menentukan posisi mata, kepala dan mulut.
- 5) Kegagalan YOLOv8 dalam mendeteksi objek kecurangan disebabkan oleh kurangnya variasi posisi objek kecurangan dan tingkat kemiripan data terlalu tinggi.

## 5.2 Saran

Adapun beberapa saran yang dapat digunakan untuk penelitian selanjutnya yang memiliki tema yang sama, yaitu :

- 1) Diharapkan pada penelitian selanjutnya menggunakan data yang lebih variatif dengan posisi objek kecurangan yang lebih banyak dan juga dengan tingkat kemiripan data yang lebih kecil.
- 2) Melakukan penelitian dengan penerapan metode algoritma lain dan terbaru untuk dapat dibandingkan hasilnya dengan hasil yang didapat menggunakan algoritma YOLOv8.
- 3) Penelitian selanjutnya diharapkan mencoba *batch size* yang lebih tinggi dan *epoch* yang lebih bervariasi apabila menggunakan spesifikasi *software* dan *hardware* yang lebih tinggi.
- 4) Penelitian selanjutnya dapat menjalankan pendeteksian pergerakan peserta secara *real time* dari sisi *website* peserta.

## DAFTAR PUSTAKA

- Abdul Elah Alkhalisy, M., & Hameed Abid, S. (2023). *The Detection of Students' Abnormal Behavior in Online Exams Using Facial Landmarks in Conjunction with the YOLOv5 Models*. Iraqi Journal for Computers and Informatics, 49(1).
- Ahmad, I., AlQurashi, F., Abozinadah, E., & Mehmood, R. (2021). *A Novel Deep Learning-based Online Proctoring System using Face Recognition, Eye Blinking, and Object Detection Techniques*. International Journal of Advanced Computer Science and Applications, 12(10).
- Anagha, U., H., C.S., S., & T., Dr. S. (2023). *Automated Online Remote Proctoring System*. © 2023 IJRTI |, 8(7).
- Fauzan, A., Novamizanti, L., Si, S., & Fuadah, Y. N. (2018). *Perancangan Sistem Deteksi Wajah Untuk Presensi Kehadiran Menggunakan Metode LBPH ( Local Binary Pattern Histogram) Berbasis Android*.
- Gopalakrishnan, K., Dhiyaneshwaran, N., & Yuges, P. (2022). *Online Proctoring System Using Image Processing and Machine Learning*. International Journal of Health Sciences.
- Hussain, M. (2023). *YOLO-v1 to YOLO-v8, The Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection*. In *Machines*. Multidisciplinary Digital Publishing Institute (MDPI) (Vol. 11, Issue 7).
- Indriani, D. T. (2019). *Bentuk Kecurangan Akademik di Kalangan Mahasiswa*. In Universitas Negeri Semarang.
- Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). *MediaPipe: A Framework for Building Perception Pipelines*.
- Malhotra, N., Suri, R., Verma, P., & Kumar, R. (2022). *Smart Artificial Intelligence Based Online Proctoring System*. 2022 IEEE Delhi Section Conference, DELCON 2022.



- Motwani, S., Nagpal, C., Motwani, M., Nagdev, N., & Yeole, A. (2021). *AI-Based Proctoring System for Online Tests*. SSRN Electronic Journal.
- Noorbehbahani, F., Mohammadi, A., & Aminazadeh, M. (2022). *A Systematic Review of Research on Cheating in Online Exams from 2010 to 2021*. *Education and Information Technologies*, 27(6).
- Nursalam, N., Suddin, B., & Munirah, M. (2016). *Bentuk Kecurangan Akademik (Academic Cheating) Mahasiswa*. *Jurnal Ilmu Tarbiyah Dan Keguruan*.
- Prasanna, M., & Reddy, G. (2017). *Development of Real Time Face Recognition System Using OpenCV*. *International Research Journal of Engineering and Technology*, 4(12).
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*.
- Santoso, B., & Kristianto, R. P. (2020). *Implementasi Penggunaan OpenCV Pada Face Recognition Untuk Sistem Presensi Perkuliahan Mahasiswa*.
- Shafiee, M. J., Chywl, B., Li, F., & Wong, A. (2017). *Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video*.
- TH. Hasan, R., & Bibo Sallow, A. (2021). *Face Detection and Recognition Using OpenCV*. *Journal of Soft Computing and Data Mining*, 2(2).
- Yulita, I. N., Hariz, F. A., Suryana, I., & Prabuwo, A. S. (2023). *Educational Innovation Faced with COVID-19: Deep Learning for Online Exam Cheating Detection*. *Education Sciences*, 13(2).