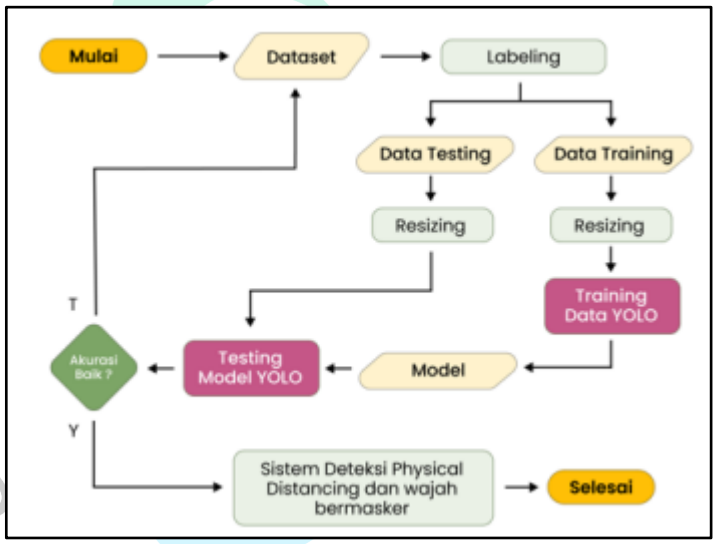
BAB IV
PERANCANGAN

Bab ini akan memaparkan hasil penelitian yang sudah dilakukan oleh peneliti sebelumnya melalui dua sub bab, yaitu hasil dan pembahasan. Penjelasan detail mengenai bab ini akan dijabarkan sebagai berikut.

4.1 Analisis Terdahulu

Analisis sistem terdahulu yang dijadikan referensi oleh peneliti dalam membuat dan merancang sistem deteksi wajah berfugsi untuk menentukan apakah kebaruan sistem yang dikembangkan oleh peniliti dapat menyempurnakan kekurangan sistem sebelumnya serta menjadi acuan dalam penelitian ini.



Gambar 4.1 Analisis Terdahulu

Gambar 4.1 merupakan langkah pembuatan model sistem deteksi wajah yang dirancang oleh peneliti sebelumnya, yaitu M. Dwisnanto Putro, Teguh Bharata Adji dan Bondan Winduratna. Pembuatan model sistem deteksi wajah tersebut menggunakan metode *Viola-Jones* dengan total 22 citra manusia dan 8 citra hewan. Akan tetapi pengujian pada sistem deteksi tersebut memiliki kekurangan, yaitu tidak dapat mendeteksi wajah pada kondisi tidak tegak lurus. Hal tersebut terjadi karena dataset yang digunakan dalam pembuatan model tersebut tidak memiliki banyak variasi.

4.2 Spesifikasi Kebutuhan Sistem Baru

Spesifikasi kebutuhan sistem baru dalam membuat dan merancang sistem deteksi wajah dibagi menjadi dua bagian, yaitu spesifikasi perangkat lunak (*software*) dan perangkat keras (*hardware*). Adapun penjelasan detail spesifikasi tersebut, sebagai berikut.

4.2.1 Spesifikasi Perangkat Lunak (*Software*)

Spesifikasi perangkat lunak dalam penelitian ini terdiri dari platform editor atau yang biasa dikenal dengan *development environment*, sistem operasi dan paltform desain digital. Perangkat lunak tersebut digunakan dalam proses perancangan desain sistem sampai dengan penulisan kode program. Adapun detail spesifikasi perangkat lunak tersebut, sebagai berikut.

Tabel 4.1 Spesifikasi Kebutuhan Perangkat Lunak

No.	Perangkat	Keterangan
1.	*Windows* 10 (32 bit)	Sistem operasi minimal yang dapat digunakan oleh *user*
2.	*Visual Studio Code*	*Code editor* yang digunakan untuk menulis kode program
3.	*Google Colaboratory*	Platform berbasis *cloud* yang digunakan untuk membuat dan melatih model
4.	*Roboflow*	Platform yang digunakan untuk mengelola *dataset*

Spesifikasi pada Tabel 4.1 merupakan daftar perangkat lunak yang digunakan oleh peneliti dalam merancang dan membuat sistem deteksi wajah. Adapun detail penjelasan kegunaan perangkat lunak tersebut dijelaskan secara rinci sebagai berikut.

a) Sistem Operasi (*Windows* 10)

Sistem operasi *Windows* 10 digunakan sebagai platform dasar untuk menjalankan semua perangkat lunak yang terlibat dalam proyek. Ini

menyediakan lingkungan yang stabil dan kompatibel untuk menjalankan *Google Colaboratory, Visual Studio Code*, dan alat-alat lain yang digunakan dalam pengembangan dan pelatihan model deteksi wajah.

b) *Google Colaboratory*

Google Colaboratory adalah platform *cloud* untuk menulis dan menjalankan kode Python, khususnya untuk pembuatan model *Machine Learning* sistem deteksi wajah yang sedang dikembangkan oleh peneliti.

c) *Code Editor (Visual Studio Code)*

Visual Studio Code (*VSCode*) adalah *editor* kode sumber yang ringan dan fleksibel. Peneliti menggunakan perangkat lunak ini untuk menjalankan dan mengintegrasikan model yang telah dibuat sebelumnya dan menerapkan logika program untuk mendeteksi indikasi pelanggaran peserta asesmen. Tidak hanya itu, *code editor* ini juga dapat menjalankan program dengan integrasi pada berbagai *library* yang dibutuhkan dalam membangun sistem deteksi wajah yang baik.

d) *Roboflow*

Roboflow adalah platform yang dirancang khusus dan digunakan peneliti untuk mengelola *dataset* dan mempersiapkannya untuk pelatihan model. Pengelolaan *dataset* pada platform ini mempermudah beberapa proses seperti, anotasi gambar, pra-pemrosesan dan augmentasi gambar.

4.2.2 Spesifikasi Perangkat Keras (*Hardware*)

Terdapat beberapa perangkat keras yang berperan penting dalam penelitian ini, yaitu prosesor, penyimpanan, dan *memory.* Adapun spesifikasi detail kebutuhan perangkat keras yang digunakan oleh peneliti dan spesifikasi minimumnya tercakup pada tabel di bawah ini

Tabel 4.2 Spesifikasi Perangkat Lebih Rendah

No.	Perangkat Keras	Keterangan
1.	Prosesor	1.6 Ghz

2.	Penyimpanan	256 GB Hardisk/SSD
3.	*Memory*	4 GB RAM

Adapun detail spesifikasi perangkat keras yang digunakan peneliti dalam membangun sistem deteksi wajah tercakup pada Tabel 4.3 di bawah ini.

Tabel 4.3 Spesifikasi Perangkat Keras Peneliti

No.	**Perangkat Keras**	**Keterangan**
1.	Prosesor	4 Ghz
2.	Penyimpanan	512 GB SSD
3.	*Memory*	16 GB RAM

a) Prosesor

Prosesor merupakan otak dari sistem komputer dan memiliki peran penting dalam melakukan berbagai operasi perhitungan serta pengolahan data. Pada penelitian ini, prosesor berfungsi untuk memproses *input* video atau gambar secara *real-time* dan mengeksekusi algoritma YOLOv8 yang sudah diterapkan pada model sistem deteksi wajah yang telah dibuat menggunakan bantuan prosesor *google colaboratory.*

b) Penyimpanan

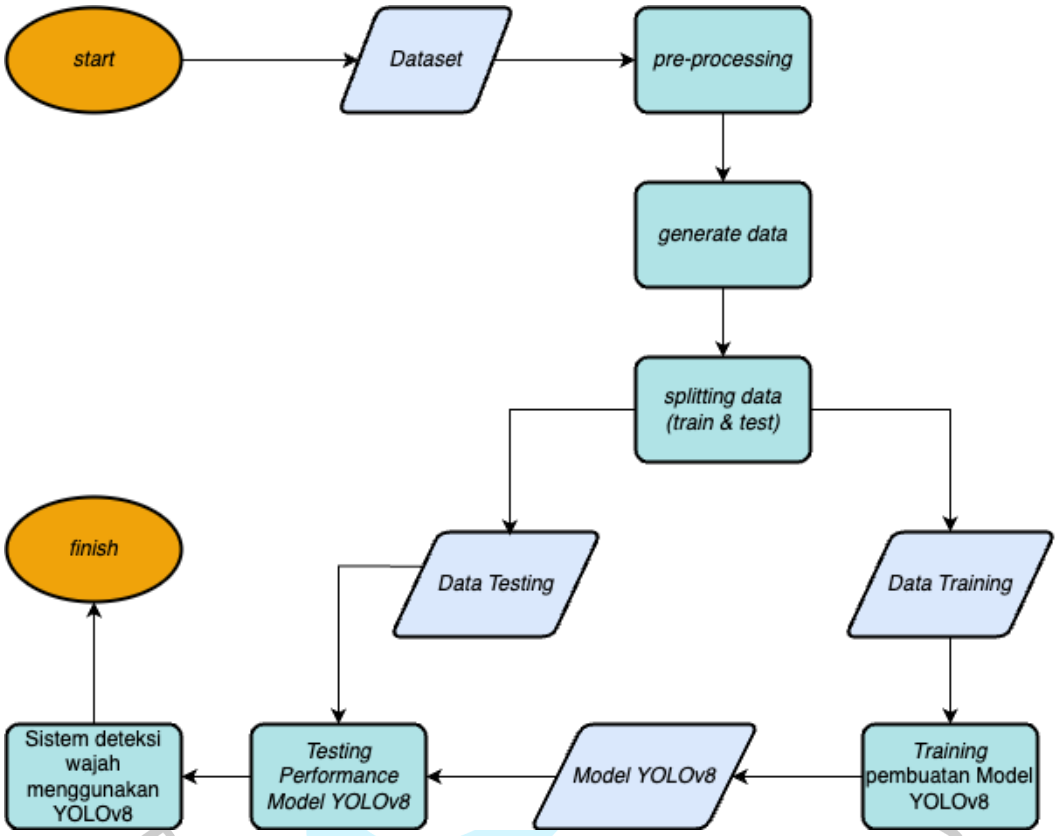
Penyimpanan berfungsi untuk menyimpan data yang diperlukan oleh sistem deteksi wajah, seperti model YOLOv8, dataset pelatihan, dan implementasi kode program pada model yang sudah dibuat sebelumnya.

c) *Memory*

Memory berfungsi sebagai penyimpanan sementara yang digunakan oleh sistem komputer untuk menjalankan program dan menyimpan data selama proses berlangsung secara *real-time.* Selain itu, perangkat keras ini digunakan untuk menyimpan model YOLOv8 yang di-*load* ke dalam RAM selama *runtime* sedang berlangsung.

4.3 Perancangan dan Pembuatan Sistem

Tahap perancangan sistem deteksi wajah manusia yang berfungsi untuk mengetahui indikasi pelanggaran pada saat pelaksanaan *online assessment* terdapat beberapa tahap, seperti yang tercakup pada gambar di bawah ini.



Gambar 4.2 Tahap Pembuatan Model

Tahapan pembuatan model pada Gambar 4.2 merupakan rangkaian proses yang dilakukan oleh peneliti dalam membuat sistem deteksi wajah menggunakan Algoritma YOLOv8, mulai dari persiapan *dataset* sampai dengan implementasi kode program. Adapun penjelasan masing-masing proses pada gambar di atas, dijabarkan sebagai berikut:

4.3.1 Pengumpulan Dataset

Pengumpulan *Dataset* merupakan tahapan utama yang dilakukan oleh peneliti dalam proses pembuatan sistem deteksi wajah. Adapun *dataset* yang dikumpulkan dan dipersiapkan oleh peneliti dalam penelitian ini, yaitu berupa

gambar/citra yang berasal dari beberapa wajah partisipan dan diklasifikasikan dalam dua kondisi, yaitu tegak lurus dan menghadap samping (kanan dan kiri). Seluruh gambar/citra tersebut diambil menggunakan *smartphone* pribadi miliki peneliti dengan total gambar sebanyak 262 buah.

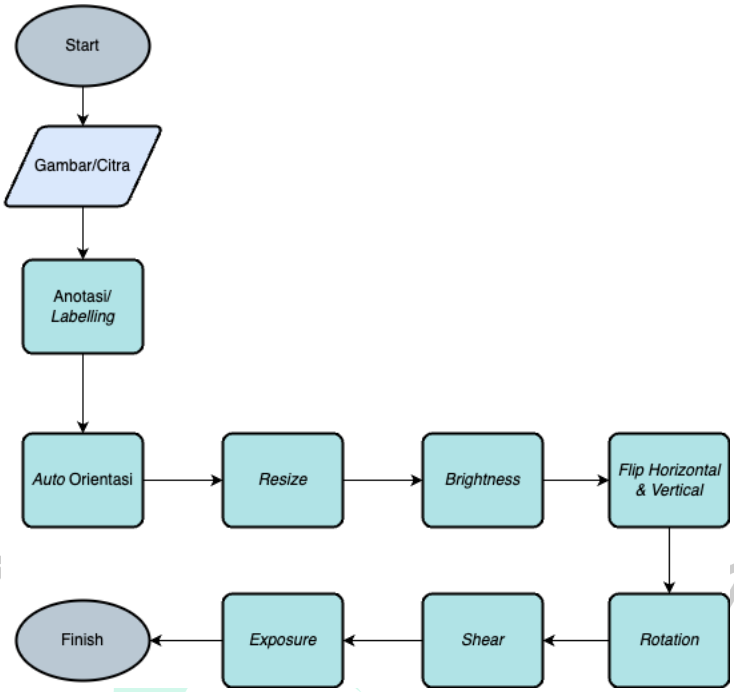
Tabel 4.4 Contoh Gambar Pada Dataset Sistem Deteksi Wajah

No.	Gambar/Citra	Keterangan
1.		Contoh gambar/citra menghadap samping yang akan di anotasi dengan label (*face_side)*
2.		Contoh gambar/citra tegak lurus yang akan di anotasi dengan label (*face_front)*

Tabel 4.4 mencakup contoh citra yang diambil menggunakan perangkat *smartphone* pribadi milik peneliti dengan dimensi 3472×4624 piksel. Pada baris pertama merupakan contoh pengumpulan gambar wajah menghadap samping yang akan dilakukan anotasi dengan label (*face_side)*, sedangkan pada baris kedua merupakan contoh gambar wajah tegak lurus yang akan di anotasi dengan label (*face_front)* pada tahapan *pre-processing* dan augmentasi. Seluruh citra yang sudah dikumpulkan oleh peneliti akan dikumpulkan menjadi *dataset* yang akan diproses pada tahapan berikutnya, yaitu pra-pemrosesan dan augmentasi gambar menggunakan platform Roboflow.

4.3.2 *Pre-processing* dan Augmentasi

Tahapan ini memiliki beberapa proses yang dilakukan untuk memperluas variasi dataset menggunakan proses augmentasi gambar. Augmentasi merupakan sebuah proses/teknik yang digunakan untuk mengubah dan memodifikasi dataset yang sudah dikumpulkan oleh peneliti agar menjadi lebih beragam. Proses tersebut dilakukan dalam beberapa metode untuk mempersiapkan citra sebelum dilakukan proses berikutnya. Tahap ini berperan penting dalam memastikan keakuratan dan kualitas hasil penelitian. Adapun detail tahapan ini tercakup pada gambar di bawah ini:



Gambar 4.3 *Pre-processing and Augmentation*

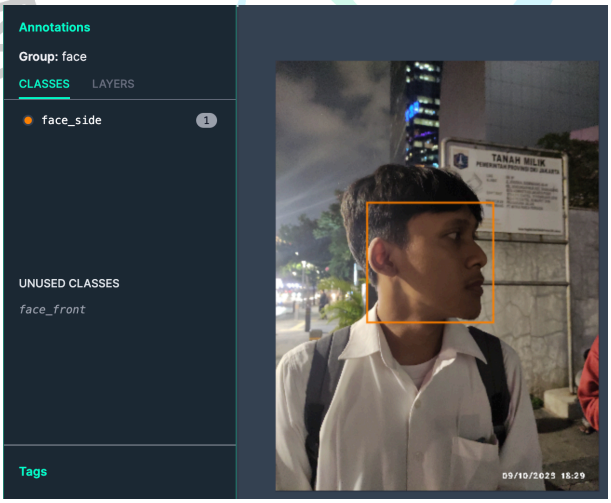
Gambar 4.3 merupakan alur *pre-processing* dan augmentasi yang dilakukan oleh peneliti untuk menambah variasi *dataset* sebelum dibagi menjadi data pelatihan dan pengujian. Terdapat beberapa proses pada gambar di atas, mulai dari proses anotasi, kemudian dilanjutkan dengan proses *auto* orientasi, *resize, brightness, flip horizontal* dan *vertical, rotation, shear*, dan diakhiri dengan proses *exposure.* Adapun penjelasan secara rinci setiap proses yang tercakup pada gambar di atas, dijabarkan sebagai berikut:

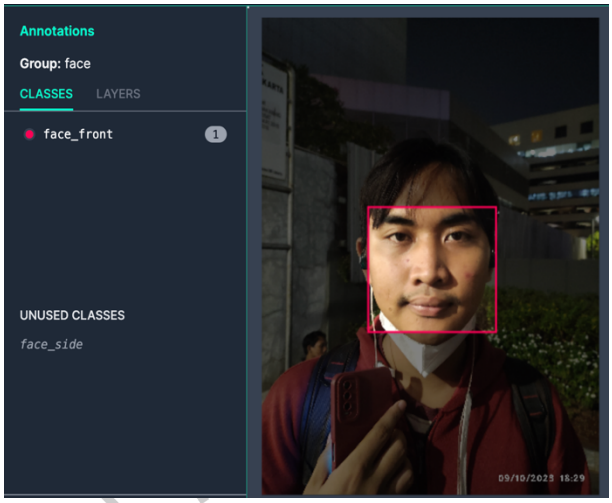
a) Tahap Anotasi/*Labelling*

Proses ini merupakan salah satu tahap terpenting yang dilakukan peneliti untuk mempersiapkan dataset sebelum pembuatan model sistem deteksi wajah dilakukan. Proses anotasi/*labelling* dilakukan secara manual melalui platform roboflow sebanyak jumlah gambar/citra yang sudah dikumpulkan sebelumnya, yaitu 262 buah. Langkah pemberian label pada gambar, informasi tambahan ditambahkan ke setiap citra data yang akan digunakan dalam pelatihan dan pengujian algoritma YOLO. Proses ini melibatkan penggunaan alat yang dikenal sebagai labeling, yang dirancang untuk melakukan anotasi gambar dengan cara yang cepat dan efisien.

Labelling merupakan salah satu alat yang umum dan langkah utama yang digunakan dalam proses penandaan gambar dalam mengelola *dataset*. Melalui proses ini, pengguna dapat menandai dan mengidentifikasi lokasi objek dalam setiap gambar dengan menentukan kotak pembatas. Selain itu, pengguna juga dapat menetapkan label atau kategori yang sesuai untuk objek yang diberi anotasi. Proses ini membantu menciptakan dataset yang dianotasi dengan baik, yang selanjutnya dapat digunakan untuk melatih model pembelajaran mesin dalam tugas-tugas seperti klasifikasi gambar, dan deteksi objek Roboflow menyediakan antarmuka yang memudahkan peneliti untuk melakukan anotasi dan manajemen dataset secara efisien.

Tabel 4.5 Tahap Anotasi Gambar pada Dataset Sistem Deteksi Wajah

No.	Gambar/Citra	Keterangan
1.		Contoh anotasi pada gambar yang sudah di anotasi dengan label "*face_side*"

2.		Contoh anotasi pada gambar yang sudah di anotasi dengan label "*face_front*"

Tabel 4.5 mencakup contoh gambar saat proses anotasi dilakukan. Pada baris pertama merupakan contoh gambar wajah menghadap samping yang sudah dilakukan anotasi dengan label "*face_side*" dan pada baris kedua merupakan contoh gambar wajah tegak lurus dengan anotasi label "*face_front*". Setelah proses ini selesai dilakukan, maka seluruh citra tersebut akan dibagi menjadi data pelatihan (test) dan pengujian (test). Selain membagi gambar pada dua jenis data tersebut, peneliti juga melakukan set data valid pada beberapa gambar.

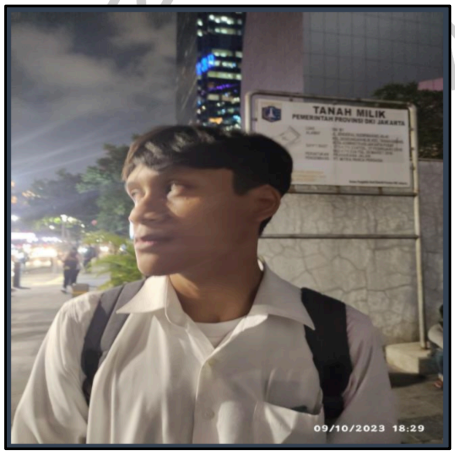
Setelah proses penandaan selesai, hasil anotasi gambar disimpan dalam format yang sesuai dengan format anotasi gambar YOLO. Setiap gambar akan memiliki *file* .txt dengan nama yang sama, yang berisi informasi tentang kelas objek, koordinat objek, tinggi, dan lebar objek yang terdapat dalam gambar tersebut. Format anotasi gambar YOLO mengikuti struktur yang telah ditetapkan, di mana setiap baris dalam file .txt mencerminkan sebuah objek dalam gambar.

b) Tahap *Auto* Orientasi

Fungsi "*Auto Orient*" pada pra-pemrosesan *dataset Roboflow* adalah fitur yang memungkinkan otomatisasi penyesuaian orientasi citra. Citra-citra yang dimasukkan ke dalam dataset pelatihan bisa memiliki orientasi yang berbeda, seperti citra yang diambil dalam orientasi potret (vertikal) atau lanskap

(horizontal), atau bahkan citra yang miring atau terbalik. Tentunya proses ini akan sangat membantu proses pengelolaan seluru citra menjadi lebih selaras.

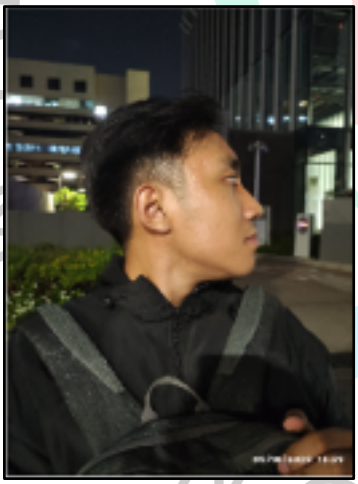
Tabel 4.6 Penerapan Proses *Auto* Orientasi pada Seluruh Gambar

No.	Gambar/Citra	Keterangan
1.		Penerapan proses *Auto* Orientasi pada wajah tegak lurus yang sudah di anotasi dengan label (*face_front)*
2.		Penerapan proses *Auto* Orientasi pada wajah tegak lurus yang sudah di anotasi dengan label (*face_side)*
3.		Penerapan proses *Auto* Orientasi pada wajah tegak lurus yang sudah di anotasi dengan label (*face_side)*

Tabel 4.6 mencakup penerapan proses *auto* orientasi pada seluruh gambar. Pada baris pertama merupakan contoh gambar yang sudah dilakukan proses *auto* orientasi pada wajah tegak lurus, sedangkan pada baris kedua dan ketiga merupakan contoh gambar/citra yang sudah dilakukan proses *auto* orientasi pada wajah menghadap samping. Tentunya seluruh gambar yang terletak pada tabel di atas sudah melalui proses pemberian label/anotasi. Proses ini dilakukan sebelum memasuki proses augmentasi agar seluruh citra memiliki tata letak yang sama.

c) Tahap *Resize*

Setelah proses data *auto* orientasi selesai dilakukan, terdapat metode augmentasi yang digunakan peneliti dalam memperluas *dataset*, yaitu dengan melakukan *resize* citra. Proses ini dilakukan dengan mengubah ukuran citra setiap gambar sesuai dengan ukuran yang diharapkan algoritma YOLOv8 yang akan digunakan dalam proses pembuatan model deteksi wajah.



Gambar 4.4 Sebelum Proses *Resize*

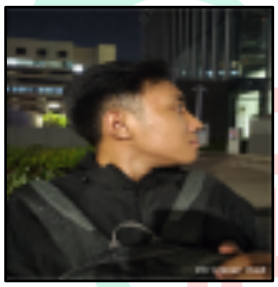
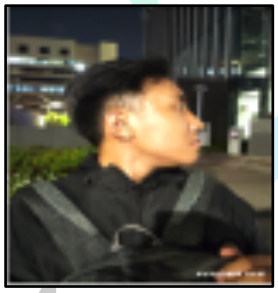
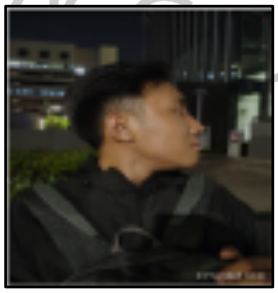
Gambar 4.5 Sesudah Proses *Resize*

Ukuran citra dalam dataset diubah menjadi 640x640 piksel untuk menjaga konsistensi dan kesesuaian dengan arsitektur serta parameter yang telah ditetapkan untuk model YOLOv8, seperti yang tercakup pada Gambar 4.5 Tujuan dari penyesuaian ini adalah agar model YOLOv8 dapat menerima citra input dengan ukuran yang tetap, yang merupakan persyaratan penting untuk memastikan pengolahan yang efektif oleh algoritma deteksi objek. Penyesuaian ukuran ini dilakukan sebelum memasuki proses augmentasi berikutnya.

d) Tahap *Brightness*

Proses *brightness* dilakukan pada seluruh citra yang terdapat di dalam dataset dengan melakukan variasi tingkat kecerahan gambar yang berfungsi untuk membantu model terbiasa dengan perubahan pencahayaan dan pengaturan kamera. Terdapat dua jenis pencahayaan pada proses ini, yaitu dengan menambah dan memeperkecil tingkat pencahayaan. Melalui proses ini, sistem deteksi wajah diharapkan dapat memberikan tingkat keakuratan dan tingkat *confidence* yang tinggi.

Tabel 4.7 Proses *Brightness* Gambar pada *Dataset*

No.	Gambar/Citra	Keterangan
1.		Sebelum gambar diterapkan proses *brightness*
2.		Penerapan *brightness* pada gambar dengan presentase 25%
3.		Penerapan *brightness* pada gambar dengan presentase -25%

Pada Tabel 4.7 dapat dilihat, bahwa gambar pada baris kedua merupakan proses peningkatan pencahayaan sebesar 25%, gambar pada baris ketiga

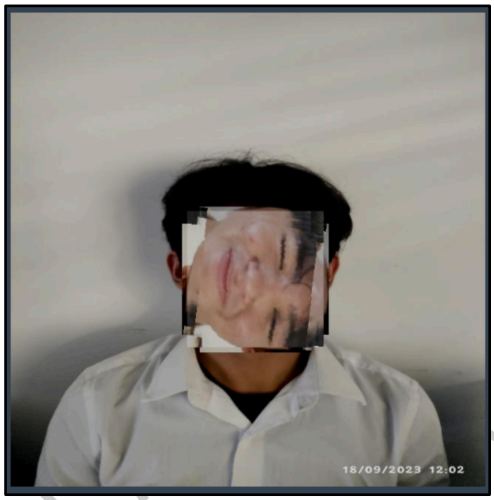
merupakan proses penurunan pencahayaan sebesar -25%. Setelah proses ini dilakukan, maka variasi baru pada *dataset* sistem deteksi wajah yang sedang dirancang akan disimpan untuk tahap proses augmentasi berikutnya.

e) Tahap *Flip Horizontal* dan *Vertical*

Proses *Flip* Horizontal adalah salah satu teknik peningkatan data yang diterapkan dalam penelitian ini. Proses ini melibatkan pembalikan citra secara horizontal, mengubah orientasi dari kanan ke kiri atau sebaliknya. Tujuan dari teknik peningkatan ini adalah untuk menghasilkan variasi citra dengan sudut pandang yang berbeda dalam dataset sistem deteksi wajah menggunakan algoritma YOLOv8. Proses *flip* horizontal dilakukan dengan memanipulasi citra asli, menciptakan citra yang merupakan cerminan horizontal dari citra aslinya. Dengan kata lain, citra yang sebelumnya menghadap ke kanan akan menjadi citra yang menghadap ke kiri, dan sebaliknya. Proses ini diterapkan pada semua citra awal dalam *dataset*.

Tabel 4.8 Penerapan Proses *Flip Vertical* dan *Horizontal*

No.	Gambar/Citra	Keterangan
1.		Penerapan *Flip Vertical* pada gambar yang sudah di anotasi

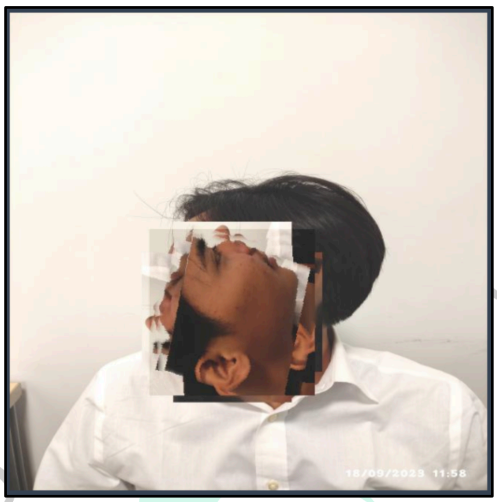
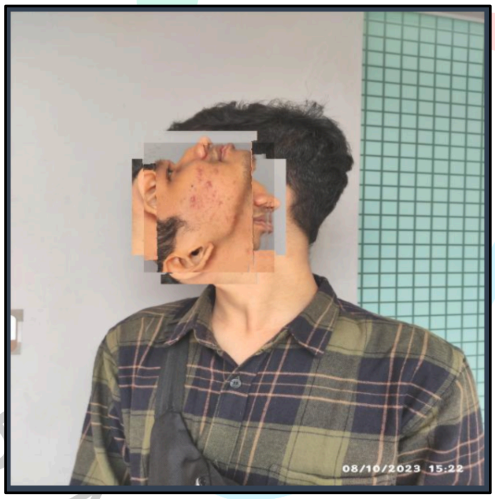
2.		Penerapan *Flip Horizontal* pada gambar yang sudah di anotasi

Penerapan proses augmentasi ini, dilakukan dengan melakukan perputaran gambar secara vertikal dan horizontal. Hal ini bermanfaat dalam meningkatkan kemampuan model atau algoritma dalam mengenali dan memahami bahasa isyarat yang terwakili dalam citra-citra tersebut. Variasi sudut pandang yang lebih beragam membuat model dapat menjadi lebih adaptif dan akurat dalam mengidentifikasi bahasa isyarat dalam berbagai posisi dan orientasi

f) Tahap *Rotation*

Tahap augmentasi rotasi dilakukan dengan cara memutar citra pada sudut tertentu. Dalam penelitian ini, dilakukan pergeseran gambar secara acak dalam kisaran -20 hingga 20 piksel pada sumbu x dan y. Pergeseran ini bersifat acak, yang berarti gambar dapat bergeser ke kiri, kanan, atas, atau bawah dengan jarak antara -20 hingga 20 piksel. Dalam konteks ini, sumbu x mengacu pada arah horizontal (kiri-kanan) dan sumbu y mengacu pada arah vertikal (atas-bawah). Oleh karena itu, pergeseran gambar secara acak dapat terjadi ke kiri atau kanan pada sumbu x, serta ke atas atau bawah pada sumbu y.

Tabel 4.9 Proses Rotasi pada Dataset

No.	Gambar/Citra	Keterangan
1.		Penerapan proses rotasi pada gambar yang sudah di anotasi
2.		Penerapan proses rotasi pada gambar yang sudah di anotasi

Pergeseran gambar secara acak yang dilakukan pada tahap ini bertujuan menciptakan variasi posisi yang lebih beragam pada citra-citra dalam *dataset* sistem deteksi wajag. Hal ini membantu memperkaya *dataset* dengan berbagai variasi posisi wajah dalam bahasa isyarat, sehingga meningkatkan kemampuan model dalam mengenali dan memahami bahasa isyarat yang melibatkan perubahan posisi wajah.

g) Tahap *Shear*

Tahap ini dilakukan untuk meningkatkan ketahanan model terhadap perubahan perspektif. Hal ini dikarenakan perspektif dapat dipengaruhi oleh

jarak kamera dari objek yang dideteksi. Jika perspektif dalam gambar tidak sesuai dengan perspektif yang diharapkan, maka algoritma YOLOv8 akan kesulitan untuk mendeteksi wajah.

Tabel 4.10 Proses *Shear* pada Gambar *Dataset*

No.	Gambar/Citra	Keterangan
1.		Penerapan proses *shear* pada gambar yang sudah di anotasi
2.		Penerapan proses *shear* pada gambar yang sudah di anotasi

h) Tahap *Exposure*

Tahap penyesuaian kecerahan (*brightness adjustment*) adalah salah satu aspek dari penyesuaian *exposure* dalam konteks pengolahan citra. Namun, ada perbedaan penting antara kedua konsep ini:

1. Penyesuaian Kecerahan (*Brightness Adjustment*)

a. Penyesuaian kecerahan adalah tindakan mengubah nilai piksel dalam citra untuk membuat seluruh gambar menjadi lebih terang atau lebih gelap.

b. Proses ini melibatkan penambahan atau pengurangan nilai kecerahan dari setiap piksel dalam citra.

c. Hasilnya adalah perubahan keseluruhan dalam tingkat kecerahan citra tanpa mempengaruhi kontras antara objek dan latar belakang.

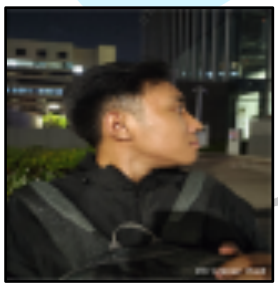
2. Penyesuaian Exposure

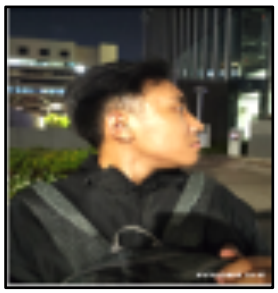
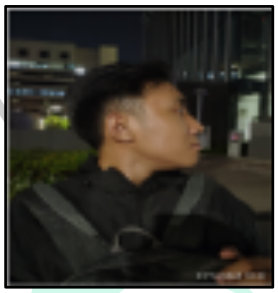
a. Penyesuaian exposure lebih kompleks dan melibatkan beberapa parameter fotografi, termasuk waktu bukaan (*shutter speed*), bukan lensa (*aperture*), dan sensitivitas ISO.

b. Penyesuaian exposure dapat mempengaruhi sejauh mana cahaya mencapai sensor kamera saat mengambil gambar.

c. Ini bisa mencakup penyesuaian kecerahan, kontras, dan sejumlah faktor lainnya yang mempengaruhi cara citra direkam.

Tabel 4.11 Proses *Exposure* Gambar pada *Dataset*

No.	Gambar/Citra	Keterangan
1.		Sebelum gambar diterapkan proses *brightness*

2.		Penerapan *exposure* pada gambar dengan presentase 25%
3		Penerapan *exposure* pada gambar dengan presentase -25%

Sama halnya dengan proses sebelumnya, bahwa proses *exposure* berfungsi untuk menambah variasi pada dataset sistem deteksi wajah yang dimiliki oleh peneliti. Peningkatan *exposure* yang diatur dalam setiap citra, yaitu sebesar 25% dan juga penurunan *exposure* sebesar -25%.

4.3.3 Generate Data

Tahapan ini akan membagi *dataset* yang sudah dilakukan pengolahan pada proses sebelumnya, yaitu *pre-processing* dan augmentasi. Jumlah gambar/citra setelah dilakukan variasi dengan berbagai metode pada proses augmentasi, terkumpul sebanyak 628 data. Kemudian akan di bagi menjadi ke dalam dua jenis data, yaitu data pelatihan (*training)* dan data pengujian *(test)*. Pembagian data tersebut dijelaskan secara rinci sebagai berikut:

a. Data *Training*

Presentase data yang disiapkan peneliti untuk data pelatihan, yaitu sebesar 80% dengan jumlah 503 gambar/citra. Data tersebut nantinya akan dilatih untuk menghasilkan model sistem deteksi wajah.

b. Data *Testing*

Data ini mencakup dua jenis data, yaitu valid dan *test*. Presentase data yang disiapkan peneliti untuk data pengujian, yaitu sebesar 20% dengan jumlah 125

gambar/citra. Data tersebut nantinya akan dijadikan acuan dan validasi saat proses pembuatan model sistem deteksi wajah dilakukan.

Presentase pembagian data yang ditetapkan oleh peneliti akan tetap sama walaupun variasi dataset dilakukan untuk memperbanyak citra dan meningkatkan kemampuan model dalam melakukan deteksi. Pada setiap data uji dan data pelatihan terdapat dua berkas/folder yang mencakup "*labels"* dan "*images"*. Pada masing-masing folder tersebut terdapat terdapat dua sub folder yang mewakili dua kondisi wajah manusia sebagai indikasi pelanggaran pada saat proses pelaksanaan asesmen, yaitu wajah tegak lurs dan juga wajah mengahadap samping (kanan dan kiri). Citra-citra tersebut disimpan dalam folder *"images"* dan juga terdapat file dengan format *.txt* untuk menyimpan hasil *labeling* pada setiap citra-citra gambar tersebut.

4.3.4 Pembuatan Model

Pada langkah ini, dilakukan pengembangan dan pembuatan model sistem deteksi wajah menggunakan algoritma YOLOv8 melalui dataset yang sudah dirancang sebelumnya menggunakan platform Roboflow. Pembuatan model ini melalui platform *Google Colaboratory* karena memiliki prosesor yang sangat baik, yaitu "T4" karena untuk pengembangan *machine learning* dan juga penyimpanan dan memori yang sangat besar, sehingga pembuatan model akan memakan waktu yang singkat serta efisien. Berikut merupakan implementasi kode program pada pembuatan model sistem deteksi wajah:

```
!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)

!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
from ultralytics import YOLO

from IPython.display import display, Image
```

Gambar 4. 6 Potongan Kode Program 1

Potongan kode pada Gambar 4.6 befungsi untuk menampilkan informasi tentang GPU yang sedang digunakan. Perintah ini memberikan informasi seperti model GPU, penggunaan memori, dan informasi lainnya terkait GPU. Kode tersebut juga befungsi untuk melihat informasi tentang GPU yang dialokasikan untuk sesi Colab Anda, termasuk penggunaan GPU saat ini dan sisa kapasitas memori yang tersedia. Kemudian kode tersebut juga berfungsi untuk mengimpor modul sistem operasi di Python serta mendapatkan direktori kerja saat ini (*current working directory*) menggunakan fungsi getcwd() dari modul os. Hasilnya kemudian disimpan dalam variabel "*HOME*".

Selain itu, kode tersebut juga berfungsi untuk menginstal versi tertentu dari pustaka *Ultralytic* dan menghapus output sel sebelumnya di *notebook*. Fungsi clear_output() dari modul display digunakan untuk membersihkan *output* di sel saat ini. Selanjutnya, kode tersebut juga akan mengimpor pustaka Ultralytics yang telah diinstal dan memanggil fungsi checks() dari pustaka *Ultralytics*. Fungsi ini mungkin dirancang untuk melakukan pemeriksaan atau menampilkan informasi tertentu terkait instalasi atau konfigurasi *Ultralytics*. Fungsi terakhir kode program pada Gambar 4.6 berfungsi untuk mengimpor kelas YOLO dari pustaka *Ultralytics* dan mengimpor fungsi *display* dan *Image* dari modul IPython.display. Fungsi ini digunakan untuk menampilkan gambar di dalam lingkungan *notebook*.

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

!pip install roboflow --quiet

from roboflow import Roboflow
rf = Roboflow(api_key="VRvB9RZBTRMtscIM7L8Y")
project = rf.workspace("yolooooooo").project("face-detection-hmah3")
dataset = project.version(5).download("yolov8")
```

Gambar 4.7 Potongan Kode Program 2

Potongan kode program pada Gambar 4.7 berfungsi untuk membuat direktori bernama "*datasets*" di direktori kerja saat in, kemudian mengganti direktori kerja saat ini ke direktori yang baru dibuat ("*datasets*"). "%cd" adalah *IPython Magic Command* yang digunakan untuk mengubah direktori kerja. Kode tersebut juga berfungsi untuk menginstal pustaka *Roboflow*, mengimpor kelas Roboflow dan membuat objek rf dari kelas *Roboflow* dengan menyediakan kunci API. Setelah itu, kode tersebut akan mengakses *workspace dengan nama "yolooooooo"* dengan *project "face-detection-hmah3"* yang dimilki peneliti dalam menyiapakan *dataset.*

```
%cd {HOME}
print(dataset.location)

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=100 imgsz=800 plots=True
```

Gambar 4.8 Potongan Kode Program 3

Potongan kode program pada Gambar 4.8 berfungsi untuk mencetak lokasi atau *path dataset* ke layar. Variabel *dataset* telah diinisialisasi sebelumnya saat *dataset* diunduh dari Roboflow. Kemudian kode tersebut berfungsi sebagai perintah *shell* yang menggunakan utilitas YOLO untuk melatih model deteksi objek. Adapun beberapa argumen yang disertakan, yaitu:

a. "task=detect": Menunjukkan bahwa program akan melatih model untuk tugas deteksi objek.
b. "mode=train": Menunjukkan bahwa program sedang dalam mode pelatihan.
c. "model=yolov8s.p"t: Menunjukkan model YOLO yang akan digunakan dalam pelatihan.

d. “data={dataset.location}/data.yaml”: Menunjukkan lokasi konfigurasi data dalam format YAML. Ini adalah file yang berisi konfigurasi *dataset* untuk pelatihan.
e. “epochs=150”: Menunjukkan jumlah *epoch* pelatihan yang akan dilakukan (jumlah kali model melihat seluruh *dataset* pelatihan).
f. “imgsz=800”: Menunjukkan ukuran gambar input yang akan digunakan selama pelatihan (dalam piksel).
g. “plots=True”: Menunjukkan apakah ingin menampilkan plot hasil pelatihan. Jika diatur “*True*”, akan menampilkan grafik evaluasi selama pelatihan.

```
from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/MyDrive/

import shutil

source_file = '/content/runs/detect/train/weights/best.pt'
destination_folder = '/content/drive/My Drive/'

shutil.copy(source_file, destination_folder)
```

Gambar 4.9 Potongan Kode Program 4

Potongan kode program pada Gambar 4.9 berfungsi untuk mengimpor modul *drive* dari *library* google.colab, yang menyediakan fungsi untuk melakukan interaksi dengan *Google Drive* dalam lingkungan *Google Colab* dan mengaitkan Google Drive dengan sesi Google Colab dan me-mount (menghubungkan) drive ke direktori “/content/drive”, kemudian mengubah direktori kerja saat ini ke direktori yang disebut "MyDrive" dalam *Google Drive*. %cd adalah *IPython Magic Command* yang digunakan untuk mengubah direktori kerja. Selain itu, kode program diatas berfungsi sebagai operasi file dan direktori, termasuk menyalin berkas. Kemudian, menetapkan *path* berkas yang akan disalin ke variabel “source_file”. Selanjutnya, menetapkan direktori tujuan untuk penyalinan ke variabel “destination_folder” dan Menggunakan fungsi *copy* dari modul *shutil* untuk menyalin berkas dari “source_file” ke “destination_folder”. Berkas “best.pt” akan disalin dari direktori “/content/runs/detect/train/weights/” ke direktori “My Drive” di *Google Drive*.

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/confusion_matrix.png', width=600)
```

Gambar 4.10 Potongan Kode Program 5

Gambar 4.10 berfungsi untuk mengubah direktori kerja saat ini ke direktori yang telah ditentukan sebelumnya dan disimpan dalam variabel "HOME". %cd digunakan untuk mengubah direktori kerja, kemudian, menggunakan modul *Image* dari "IPython.display" untuk menampilkan gambar dengan nama file "confusion_matrix.png" yang berada di dalam direktori "{HOME}/runs/detect/train/". Lebar gambar diatur menjadi 600 piksel.

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)
```

Gambar 4.11 Potongan Kode Program 6

Kode program pada Gambar 4.11 berfungsi untuk menampilkan gambar dengan nama *file* "results.png" yang berada di dalam direktori "{HOME}/runs/detect/train/". Lebar gambar diatur menjadi 600 piksel.

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg', width=600)
```

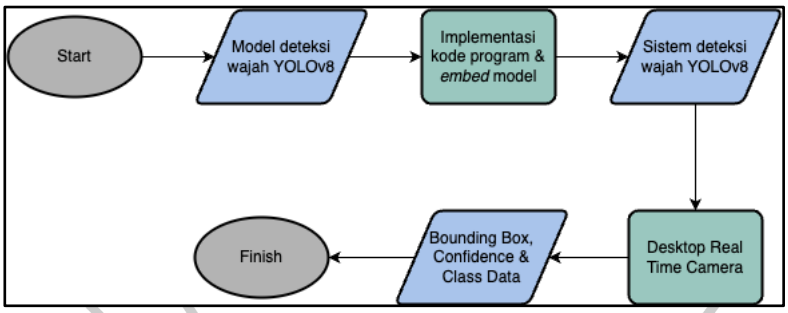
Gambar 4.12 Potongan Kode Program 7

Potongan kode program pada Gambar 4.12 berfungsi untuk engubah direktori kerja saat ini ke direktori yang telah ditentukan sebelumnya dan disimpan dalam variabel "HOME" dan menggunakan modul *Image* dari IPython.display untuk menampilkan gambar dengan nama *file*""val_batch0_pred.jpg" yang berada di dalam direktori "{HOME}/runs/detect/train/". Lebar gambar diatur menjadi 600 piksel.

4.4 Integrasi Model Deteksi Wajah

Tahapan ini berfungsi untuk melakukan integrasi model deteksi wajah yang sudah dirancang dan dibuat sebelumnya melalui berbagai proses. Integrasi yang dilakukan pada tahap ini menggunakan aplikasi *visual studio code* sebagai *code editor*. Bahasa pemrograman yang digunakan dalam tahap ini, yaitu menggunakan

python dan juga sebagai penerapan logika agar model dapat mendeteksi indikasi pelanggaran peserta asesmen berdasarkan kondisi yang sudah ditetapkan sebelumnya. Adapun alur integrasi model tersebut tercakup pada gambar di bawah ini.



Gambar 4. 13 Tahapan Integrasi Model

Proses implementasi kode program dan *embed* model deteksi wajah yang tertera pada Gambar 4.13 berfungsi agar sistem deteksi wajah dapat dijalankan secara *real-time webcam* melalui platform *desktop*. Penjelasan detail mengenai proses tersebut, dijabarkan sebagai berikut:

```
from ultralytics import YOLO
import cv2
import math
from tkinter import messagebox
from tkinter import *
import time
import matplotlib
matplotlib.use("TKAgg")
from matplotlib import pyplot as plt
```

Gambar 4. 14 Potongan Kode Program 8

Gambar 4.14 berfungsi untuk mengimpor beberapa modul yang dibutuhkan untuk menjalankan model sistem deteksi wajah. Penjelasan detail pada masing-masing baris program tersebut, dijabarkan sebagai berikut:

a. Baris pertama pada gambar di atas berfungsi untuk mengimpor kelas YOLO dari pustaka *Ultralytics*

b. Baris kedua pada gambar di atas berfungsi untuk melakukan impor pustaka OpenCV untuk pengolahan citra dan komputer visi.

c. Baris ketiga pada gambar di atas berfungsi untuk melakukan fungsi-fungsi matematika dasar.

d. Baris keempat pada gambar di atas berfungsi untuk

e. Baris kelima pada gambar di atas berfungsi untuk mengimpor modul *messagebox* dari pustaka Tkinter

f. Baris keenam pada gambar di atas berfungsi untuk melakukan impor semua fungsi dan kelas dari pustaka Tkinter.

g. Baris ketujuh pada gambar di atas berfungsi untuk mengimpor modul *time*, yang menyediakan fungsi-fungsi waktu.

h. Baris kedelapan pada gambar di atas berfungsi untuk melakukan impor pustaka matplotlib

i. Baris kesembilan berfungsi untuk mengatur *back-end* matplotlib menggunakan "TKAgg", hal tersebut berfungsi untuk rendering grafik pada antarmuka pengguna melalui TKinter.

j. Baris terakhir pada gambar di atas berfungsi untuk mengimpor modul pyplot dari matplotlib sebagai plt.

```
plt.plot()
# start webcam
cap = cv2.VideoCapture(0)
cap.set(3, 640) # Lebar gambar
cap.set(4, 480) # Tinggi gambar
# model
model = YOLO('models/best.pt')
# object classes
classNames = ["face_front",
              "face_side"
              ]

start_time_no_face = time.time()
start_time_face_side = time.time()
face_detected = False

def popup_message(msg):
    root = Tk()
    root.withdraw()
    messagebox.showinfo("Alert", msg)
    root.destroy()
```

Gambar 4. 15 Potongan Kode Program 9

Potongan kode program pada Gambar 4.15 berfungsi untuk memanggil fungsi plot() dari matplotlib, kemudian membuka koneksi webcam menggunakan OpenCV, mengatur lebar gambar webcam menjadi 640 piksel dan tinggi gambar webcam menjadi 480 piksel. Selain itu kode di atas juga berfungsi untuk membuat objek model YOLO dengan menggunakan berkas model sistem deteksi wajah yang sebelumnya sudah dibuat dan disimpan di direktori "models" dengan nama "best.pt", kemudian menetapkan dua nama *class*, yaitu "*face_front*" dan "*face_side*". Selain itu, kode di atas juga berfungsi untuk menginisialisasi "*variabel start_time_no_face*" dan "*start_time_face_side*" dengan waktu saat ini serta membuat variable baru bernama "*face_detected*" dan menginisialisasinya dengan nilai "*false*". Fungsi terakhir pada kode di atas, yaitu untuk menampilkan pesan pop-up dengan judul "Alert" dan isi pesan yang dapat ditentukan. Setelah pengguna menutup pesan pop-up, jendela Tkinter akan ditutup, nantinya fungsi ini akan di panggil pada kondisi tertentu sesuai dengan indikasi pelanggaran peserta asesmen.

```
while True:
    success, img = cap.read()
    results = model(img, stream=True)
    no_face_detected_duration = time.time() - start_time_no_face
    face_side_detected_duration = time.time() - start_time_face_side
    face_detected_current = False
    face_side_detected_current = False
    faces_count = 0
```

Gambar 4. 16 Potongan Kode Program 10

Potongan program pada Gambar 4.16 berfungsi berfungsi sebagai langkah awal dalam proses deteksi objek menggunakan model YOLO pada citra yang diambil secara *real-time* dari *webcam*, kemudian disimpan pada variable "*results*" dan diolah menggunakan model dengan algoritma YOLOv8 yang sudah diimpor sebelumnya. Selain itu, beberapa variabel seperti *"no_face_detected_duration"* dan *"face_side_detected_duration"* berfungsi untuk menghitung serta melacak berapa lama sejak waktu terakhir tidak ada wajah atau wajah menghadap ke samping terdeteksi. Sedangkan variabel *"face_detected_current"* dan *"faces_count"* digunakan untuk melacak status deteksi wajah pada iterasi saat ini. Seluruh kode tersebut berfungsi untuk penerapan logika lebih lanjut jika nantinya suatu kondisi deteksi terpenuhi, seperti menampilkan notifikasi *pop-up*.

```
for r in results:
    boxes = r.boxes

    for box in boxes:
        # bounding box
        x1, y1, x2, y2 = box.xyxy[0]
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        face_detected_current = True

        if classNames[int(box.cls[0])] == "face_side":
            face_side_detected_current = True

        faces_count += 1

        # put box in cam
        cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)

        # confidence
        confidence = math.ceil((box.conf[0] * 100)) / 100
        print("Confidence --->", confidence)

        # class name
        cls = int(box.cls[0])
        print("Class name -->", classNames[cls])

        # object details
        org = [x1, y1]
        font = cv2.FONT_HERSHEY_SIMPLEX
        fontScale = 1
        color = (255, 0, 0)
        thickness = 2

        cv2.putText(img, classNames[cls], org, font, fontScale, color, thickness)
```

Gambar 4. 17 Potongan Kode Program 11

Potongan kode program pada Gambar 4.17 merupakan bagian utama dari proses deteksi wajah menggunakan model yang sudah diimpor sebelumnya. Program tersebut akan mengambil informasi melalui kotak pembatas pada wajah

yang terdeteksi dari *webcam* secara *real-time*. Setiap kotak pembatas memberikan koordinat (x, y) yang digunakan untuk menggambarkan area wajah pada citra *webcam* menggunakan bantuan *libary* OpenCV. Selain itu, program juga akan menghitung tingkat keyakinan (*confidence*) dari model deteksi wajah dan mencetaknya pada konsol *code editor*. Tidak hanya itu, program juga akan mencetak jenis *class* wajah yang terdeteksi dari model pada *log code editor*. Kode tersebut juga akan menyimpan data jumlah wajah terdeteksi pada variabel *"faces_count"* serta status deteksi wajah saat ini yang akan disimpan pada variable *"face_detected_current*" dan *"face_side_detected_current"*.

```
    cv2.imshow('Real Time Webcam', img)

    if face_detected_current:
        face_detected = True
    elif not face_detected_current and face_detected:
        start_time_no_face = time.time()
        face_detected = False

    if cv2.waitKey(1) == ord('q'):…

cap.release()
cv2.destroyAllWindows()
```

Gambar 4. 18 Potongan Kode Program 12

Potongan kode program pada Gambar 4.18 berfungsi untuk menampilkan citra secara *real-time* dari *webcam* pada jendela OpenCV dengan kotak deteksi wajah serta mengelola status deteksi dan menyimpannya pada variabel *"face_detected*" berdasarkan hasil deteksi pada setiap iterasi. Jika wajah terdeteksi, maka "*face_detected*" akan bernilai *True* dan jika tidak ada wajah yang terdeteksi setelah pendeteksian sebelumnya dilakukan, maka waktu mulai dan status "*face_detected*" akan diatur ulang. Program juga dapat dihentikan dengan menekan tombol "q" pada jendela OpenCV. Setelah program selesai dijalankan, video real-time akan dihentikan menggunakan fungsi "*cap.release*" dan semua jendela OpenCV akan ditutup menggunakan "*cv2.destroyAllWindows*".

Setelah proses implementasi kode program dan *embed* model deteksi wajah selesai dilakukan, maka sistem deteksi wajah menggunakan Algoritma YOLOv8 sudah dapat dijalankan secara *real-time* menggunakan webcam. Detail hasil deteksi sistem tersebut akan dijabarkan pada bab 5, yaitu pada sub bab hasil dan pembahasan.

4.5 Skenario Pengujian

Setelah model deteksi wajah berhasil dirancang dan dibuat serta diintegrasikan dengan kode program sehingga terbentuknya sistem deteksi wajah, terdapat beberapa skenario pengujian untuk mengevaluasi model dan logika kode program menggunakan beberapa metode yang sudah ditetapkan pada bab 3. Adapun detail skenario pengujian dijabarkan sebagai berikut.

4.5.1 Skenario Pengujian *Confusion Matrix*

Pengujian ini berfungsi untuk mengukur performa model dalam mendeteksi kondisi wajah manusia terutama pengukuran akurasi, karena nilai akurasi yang semakin mendekati nilai 1, maka semakin baik performa model sistem deteksi wajah tersebut.

Tabel 4.12 Skenario Pengujian *Confusion Matrix*

No.	Pengukuran *Performance*	Perhitungan	Hasil yang diharapkan
1.	Akurasi	$\frac{TP + TN}{TP + TN + FP + FN}$	> 0 - 1
2.	Presisi	$\frac{TP}{TP + FP}$	> 0 - 1
3.	Recall	$\frac{TP}{TP + FN}$	> 0 - 1

4.5.2 Skenario Pengujian *Black Box*

Pengujian ini berfungsi untuk mengevaluasi apakah model sistem deteksi wajah yang sudah dirancang dan dibuat menggunakan Algoritma YOLOv8 mampu mendeteksi wajah manusia dalam beberapa kondisi yang sudah dijelaskan pada bab 1, di dalam batasan masalah.

Tabel 4.13 Skenario Pengujian *Black Box*

No.	Skenario Pengujian	Hasil yang di harapkan
1.	Wajah tegak lurus	Berhasil mendeteksi menggunakan *bounding box* dan anotasinya.
2.	Wajah menghadap samping (kanan/kiri)	Berhasil mendeteksi menggunakan *bounding box* dan anotasinya.

3.	Lebih dari satu wajah	Berhasil mendeteksi menggunakan *bounding box* dan anotasinya.

4.5.3 Skenario Pengujian *White Box*

Pengujian ini berfungsi untuk menguji sistem berjalan sesuai dengan kondisi yang sudah ditetapkan. Detail skenario pengujian *white box* sebagai berikut.

Tabel 4. 14 Skenario Pengujian Metode *White Box*

No.	Kode Program	Hasil yang di harapkan

1.	`if not face_detected_current:`
	`  if no_face_detected_duration > 5:`
	`  popup_message("No person detected.")`
	`  start_time_no_face = time.time()`
	`else:`
	`  start_time_no_face = time.time()`
		Mengeluarkan *alert* dengan keterangan *"No person detected"*

2.	`if faces_count >= 2:` `  popup_message("More then 1 person detected.")`	Mengeluarkan *alert* dengan keterangan *"More then one person detected"*

3.	`if face_side_detected_current:`
	`  if face_side_detected_duration > 5:`
	`  popup_message("Face is not looking at the camera.")`
	`  start_time_face_side = time.time()`
	`else:`
	`  start_time_face_side = time.time()`
		Mengeluarkan *alert* dengan keterangan *"face not facing to the camera"*

Skenario pengujian *white box* pada Tabel 4.14 mencakup beberapa indikasi pelanggaran yang akan dideteksi ketika proses asesmen secara daring dilaksanakan. Kondisi indikasi pelanggaran tersebut, meliputi tidak ada objek di depan kamera dan lebih dari satu objek di depan kamera berdasarkan wajah manusia serta wajah yang menengok ke arah kanan maupun kiri selama lima detik. Apabila kondisi tersebut terdeteksi oleh kamera secara *real-time,* maka program akan secara otomatis menampilkan peringatan (*alert)* menggunakan bantuan *library* TKinter.