BAB I

PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi di era *digital* telah menjadikan *computer vision* sebagai bidang yang sangat maju dan sangat bermanfaat di berbagai sektor, dari produksi industri hingga sistem keamanan. Teknologi ini mampu "melihat" dan memahami data *visual* berupa gambar dan video, sehingga dapat diterapkan dalam berbagai fungsi seperti pengenalan wajah, pemantauan lalu lintas, hingga identifikasi objek secara *otomatis*. Seiring dengan peningkatan kapasitas algoritma dan data yang tersedia, kemampuan *computer vision* dalam menginterpretasi lingkungan dan mendukung pengambilan keputusan menjadi semakin andal. Teknologi ini telah memudahkan *otomatisasi* dan memungkinkan keputusan yang lebih cepat dan akurat tanpa banyak *intervensi* manusia.

Diantara algoritma deteksi objek, YOLO *(You Only Look Once)* dikenal sebagai metode *deep learning* yang unggul dalam hal efisiensi dan kecepatan. YOLOv5, versi terbarunya, telah diadopsi dalam berbagai proyek untuk meningkatkan performa deteksi objek, terutama yang membutuhkan akurasi dan kecepatan dalam waktu nyata. Penelitian sebelumnya menunjukkan bahwa YOLOv5 menawarkan keseimbangan optimal antara akurasi dan kecepatan, menjadikannya *ideal* untuk aplikasi seperti pemantauan lalu lintas atau manajemen keramaian. Sebagai contoh, studi (Wibowo et al, 2023) mencatat bahwa YOLOv5

berhasil mencapai akurasi hingga 90% dalam mengidentifikasi kondisi buah segar dan busuk. Karena menggunakan arsitektur yang ringan dan efisien, YOLOv5 dapat diterapkan pada perangkat dengan kemampuan *komputasi* terbatas, sehingga menjadi pilihan utama bagi *praktisi* dan peneliti yang membutuhkan sistem deteksi objek yang berkinerja tinggi.

Penghitungan jumlah orang yang keluar masuk di area tertentu memiliki nilai penting dalam beragam *konteks*, seperti manajemen kapasitas ruang, keamanan, hingga perencanaan *operasional*. Tingkat akurasi dalam perhitungan ini berpengaruh langsung pada *efisiensi operasional* serta *optimalisasi* kapasitas manajemen. Selain itu, data ini menjadi dasar untuk kebijakan *strategis* yang lebih tepat, baik dalam pengaturan jadwal, penerapan protokol keamanan, atau *optimalisasi* fasilitas yang ada. Dengan teknologi yang mendukung perhitungan *otomatis*, pemantauan jumlah orang dapat dilakukan secara lebih cepat dan akurat, yang pada akhirnya membantu dalam pengambilan keputusan di tingkat *operasional* maupun *manajerial*.

Berdasarkan bahasan diatas, penulis berinisiatif mengajukan judul penelitian "IMPLEMENTASI ALGORITMA YOLOv5 PADA PERHITUNGAN OBJEK (MANUSIA)" diharapkan dapat diketahui seberapa akurat dan stabil sistem ini dalam kondisi *operasional* nyata. Pengujian dilakukan untuk *mengevaluasi* ketepatan sistem dalam mendeteksi dan menghitung manusia, serta untuk memastikan bahwa algoritma YOLOv5 dapat diandalkan dalam lingkungan yang memiliki lalu lintas yang padat. Dengan demikian, penelitian ini tidak hanya

berfokus pada penerapan teknologi, tetapi juga pada pengujian performa yang mencakup berbagai skenario dan kondisi yang relevan.

B. Rumusan Masalah

Berdasarkan latar belakang di atas diperoleh rumusan masalah sebagai berikut:

1. Bagaimana merancang sistem yang mampu mendeteksi dan menghitung jumlah orang yang keluar dan masuk secara *real-time* dengan menggunakan algoritma YOLOv5?
2. Bagaimana cara mengintegrasikan YOLOv5 dengan sistem *database* MySQL sebagai penyimpanan data?

C. Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Untuk mengetahui cara merancang sistem deteksi dan perhitungan jumlah orang yang keluar dan masuk secara *real-time* dengan menggunakan algoritma YOLOv5.
2. Untuk mengetahui cara mengintegrasikan algoritma YOLOv5 dengan sistem *database* MySQL sebagai penyimpanan data.

D. Batasan Masalah

Berdasarkan masalah yang diteliti, penulis membatasi masalah agar pembahasan-pembahasan dalam penelitian ini tidak meluas. Berikut merupakan Batasan masalah pada penelitian ini:

1. Penelitian ini hanya mencakup implementasi algoritma YOLOv5 untuk deteksi dan perhitungan orang yang keluar dan masuk, tanpa membahas deteksi objek lainnya seperti kendaraan atau barang.
2. Batasan objek yang dihitung hanya meliputi pergerakan orang yang melewati garis yang telah ditentukan (garis masuk dan keluar) pada area yang terdeteksi oleh kamera.

E. Manfaat Penelitian

Penelitian yang bijak adalah penelitian yang bermanfaat bagi diri sendiri, dan bermanfaat untuk semua orang baik dari akademik dan sosial serta dapat dijadikan bahan perbandingan untuk penelitian selanjutnya. Manfaat dari penelitian ini terbagi menjadi 3 (tiga) bagian yaitu:

1. Manfaat bagi Penulis

   Penulis dapat mengetahui cara mengimplementasikan algoritma YOLOv5 pada perhitungan keluar masuk manusia, berdasarkan pengetahuan yang diperoleh selama menjalani pendidikan di Universitas Muhammadiyah Parepare.

2. Manfaat bagi Akademik

   Penelitian ini dapat dijadikan referensi untuk penelitian berikutnya yang berkaitan dengan penerapan algoritma YOLOv5, terutama dalam aspek pemantauan.

3. Manfaat bagi Masyarakat

   Penelitian ini dapat menjadi alternatif solusi yang efektif bagi pengelola dalam memantau arus, meningkatkan efisiensi operasional, dan mendukung pengambilan keputusan secara lebih cepat dan akurat.

BAB II

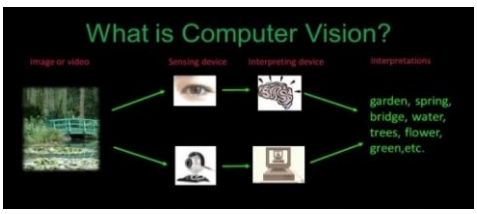
TINJAUAN PUSTAKA

A. Kajian Teori

1. Computer Vision

Computer Vision adalah salah satu cabang ilmu yang bertujuan untuk membantu komputer dalam mengambil keputusan berdasarkan analisis gambar atau citra, sehingga dapat mengenali objek fisik dan situasi nyata (Shapiro & Stockman, 2001). Teknologi ini memungkinkan komputer untuk "berfungsi seperti penglihatan manusia," mendekati kemampuan manusia dalam memahami informasi visual. Beberapa kemampuan utama dalam *Computer Vision* meliputi:

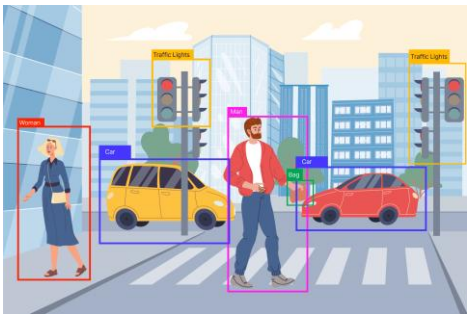
a. *Object Detection*: Mengidentifikasi keberadaan objek dalam suatu adegan dan menentukan batas-batasnya.
b. *Recognition:* Memberikan label atau identitas pada objek tertentu.
c. *Description:* Mengaitkan properti atau atribut tertentu pada objek.
d. 3D *Inference:* Menginterpretasikan adegan tiga dimensi (3D) dari data dua dimensi (2D).
e. *Interpreting Motion:* Menganalisis dan memahami gerakan dalam suatu adegan.



Gambar 2. 1 *Computer Vision*
(Sumber: *Google.com*)

2. Object Detection

Object Detection bertujuan untuk mengidentifikasi keberadaan suatu objek, menentukan cakupannya, serta menentukan lokasinya dalam sebuah gambar. Proses ini dapat dianggap sebagai bentuk lanjutan dari pengenalan objek, di mana satu kategori mewakili objek yang terdeteksi, sedangkan kategori lainnya mewakili bagian yang bukan objek. Deteksi objek dapat dikelompokkan menjadi dua jenis, yaitu *soft detection* dan *hard detection*. *Soft detection* hanya berfokus pada pendeteksian keberadaan objek, sedangkan *hard detection* mencakup pendeteksian keberadaan sekaligus lokasi objek (Jalled, 2016).



Gambar 2. 2 Contoh *Detection Object*
(Sumber: *Google.com*)

3. Deep Learning

Deep learning merupakan cabang dari *machine learning* yang menggunakan banyak lapisan pemrosesan *nonlinier* untuk melakukan *ekstraksi* fitur, mengenali pola, dan melakukan klasifikasi (Deng & Yu, 2014). (Goodfellow et al. 2016) menjelaskan bahwa *deep learning* adalah pendekatan untuk menyelesaikan masalah dalam sistem pembelajaran komputer dengan menerapkan konsep hierarki. Hierarki ini memungkinkan komputer untuk mempelajari konsep yang kompleks melalui

penggabungan konsep-konsep yang lebih sederhana. Jika diilustrasikan dalam bentuk grafik, konsep-konsep tersebut dibangun secara berlapis-lapis, dengan banyak tingkatan atau *layer*, sehingga pendekatan ini dikenal sebagai *"deep learning"* atau pembelajaran mendalam.

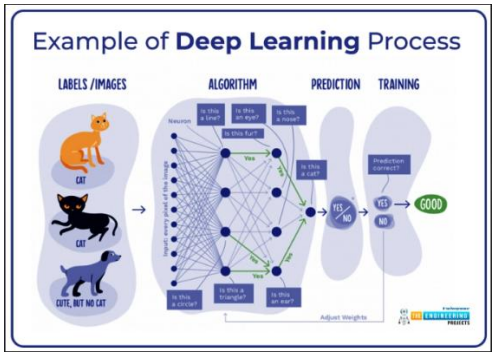
Deep learning bekerja dengan memanfaatkan arsitektur jaringan saraf tiruan yang dalam *(deep neural networks).* Proses kerjanya mirip dengan cara kerja jaringan saraf manusia (Astakona, 2024). Berikut adalah langkah-langkah cara kerja *deep learning* secara umum:

a. Pemberian Data: Proses dimulai dengan memberikan data kepada jaringan saraf tiruan.
b. *Propagasi* Maju (*Forward Propagation*): Data tersebut kemudian mengalir melalui berbagai lapisan jaringan, di mana setiap lapisan akan melakukan transformasi terhadap data tersebut.
c. Perhitungan *Output*: Selama *propagasi* maju, setiap lapisan akan melakukan perhitungan berdasarkan parameter-parameter yang dimilikinya untuk menghasilkan *output* yang diinginkan.
d. Perhitungan *Galat* (*Error Calculation*): *Output* yang dihasilkan akan dibandingkan dengan target yang sebenarnya. *Galat* antara *output* yang dihasilkan dengan target menjadi dasar untuk melakukan penyesuaian parameter jaringan.
e. *Propagasi* Mundur (*Backward Propagation*): *Galat* tersebut kemudian kembali ke jaringan dengan menggunakan algoritma pembelajaran yang

sesuai, yang kemudian disebarkan mundur melalui jaringan untuk menyesuaikan parameter-parameter *internal* jaringan.

f. *Optimisasi* Parameter: Proses *propagasi* mundur ini berulang kali dilakukan dengan menggunakan teknik *optimisasi* seperti *gradien* turun (*gradient descent*) untuk mengoptimalkan parameter-parameter jaringan.

g. *Iterasi*: Seluruh proses ini dilakukan secara berulang hingga jaringan mampu menghasilkan *output* yang sesuai dengan target dengan tingkat akurasi yang diinginkan.

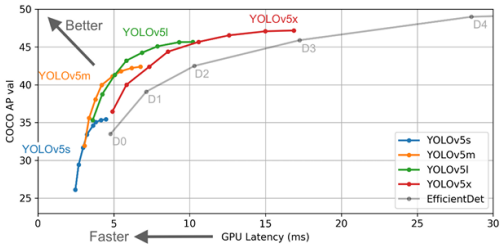


Gambar 2. 3 *Proses Deep Learning*
(Sumber: *Google.com*)

4. YOLOv5

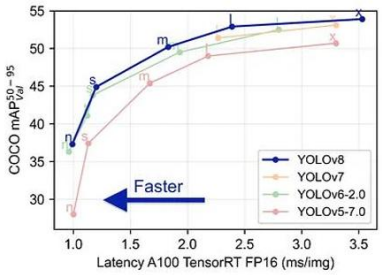
YOLOv5 adalah algoritma yang dirancang untuk mendeteksi banyak objek *(multiple object detection)* dan dikembangkan oleh *Ultralytics*. Arsitektur YOLOv5 secara umum terdiri dari tiga komponen utama, yaitu *backbone, neck,* dan *head*. Algoritma ini memanfaatkan berbagai teknik *augmentasi* data untuk meningkatkan kemampuan *generalisasi* model dan meminimalkan risiko *overfitting*. Teknik *augmentasi* tersebut meliputi *mosaic augmentation, copy-paste augmentation, random affine transformations, mix-up augmentation, albumentations,* HVS *augmentation,* serta random *horizontal flip*. Selain itu, strategi pelatihan tambahan

juga digunakan untuk mengoptimalkan performa model deteksi, seperti *multiscale training*, *autoanchor, warmup* dengan *cosine* LR *scheduler, exponential moving average, mixed precision training*, dan *hyperparameter evolution*. Pada training menggunakan YOLOv5, nilai *loss* atau kerugian dikomputasikan menggunakan tiga kombinasi komponen yang terdiri dari *classes loss, objectness loss,* dan *location loss* (Cahyani, 2023).



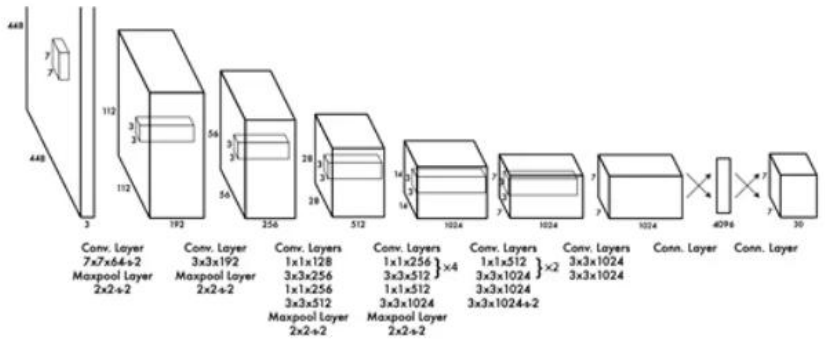
Gambar 2. 4 Perbandingan performa kerja pada tiap model YOLOv5
(Sumber: *Google.com*)

Dengan menggabungkan berbagai fitur baru, peningkatan, dan strategi pelatihan, YOLOv5 melampaui versi sebelumnya dari keluarga YOLO dalam hal performa dan *efisiensi*. Berdasarkan Gambar 2. 3, waktu *inferensi* yang dibutuhkan oleh YOLOv5 dalam melakukan deteksi berkisar pada rentang waktu 5 ms hingga 20 ms.



Gambar 2. 5 Performa YOLOv5 dengan YOLO versi terbaru
(Sumber: *Google.com*)

Gambar 2. 4 menunjukkan grafik perbandingan waktu *inferensi* yang dibutuhkan dan akurasi mAP_0.5:95 dalam melakukan deteksi menggunakan algoritma YOLOv5, YOLOv6, YOLOv7, dan YOLOv8. Berdasarkan grafik, YOLOv8 unggul dalam nilai akurasi dibandingkan YOLO versi lainnya. Akan tetapi, YOLOv5 lebih cepat dalam kecepatan deteksi, karena membutuhkan waktu *inferensi* yang lebih rendah dibandingkan YOLO versi lainnya dengan rentang waktu 1 ms hingga 3.25 ms. Adapun cara kerja YOLO dapat dilihat dari gambar berikut.



Gambar 2. 6 Cara kerja algoritma YOLO
(Sumber: *Google.com*)

YOLOv5 memiliki fungsi dan cara kerja sebagai berikut:

Fungsi YOLOv5:

a. Deteksi Objek *Real-Time*: YOLOv5 dirancang untuk mendeteksi objek dalam gambar secara *real-time*, dengan kecepatan tinggi.
b. Akurasi Tinggi: Meskipun cepat, YOLOv5 juga menawarkan tingkat akurasi yang tinggi dalam mendeteksi objek dalam berbagai konteks dan kondisi pencahayaan.

c. *Skalabilitas*: YOLOv5 dapat disesuaikan dengan berbagai skala gambar tanpa mengorbankan kinerja atau akurasi.
d. *Fleksibilitas*: Model ini dapat dilatih untuk mendeteksi berbagai kelas objek, mulai dari manusia, mobil, hingga hewan.

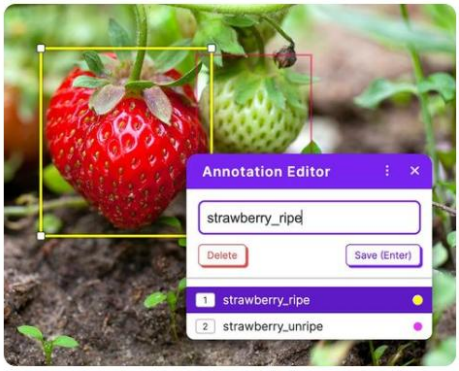
Cara Kerja YOLOv5:

a. *Preprocessing*: Gambar dimasukkan ke dalam model dan diproses untuk kecocokan format *input* model.
b. *Feature Extraction*: YOLOv5 menggunakan arsitektur *Convolutional Neural Network* (CNN) untuk mengekstraksi fitur dari gambar.
c. Deteksi Objek: Setelah fitur *diekstraksi*, YOLOv5 melakukan deteksi objek dengan membagi gambar menjadi *grid* dan memprediksi kotak pembatas (*bounding box*) dan kelas objek untuk setiap grid.
d. *Non-max Suppression:* Langkah ini menghilangkan kotak pembatas yang tumpang tindih dan menetapkan skor kepercayaan untuk setiap deteksi objek.
e. *Output: Output* dari YOLOv5 adalah kumpulan kotak pembatas yang mengandung objek, beserta label kelas dan skor kepercayaan untuk setiap deteksi.

Proses ini dilakukan dalam waktu singkat, memungkinkan YOLOv5 untuk bekerja secara *real-time* di berbagai *platform*. YOLOv5 juga dapat disesuaikan dan diperluas melalui proses pelatihan dengan dataset yang sesuai untuk kasus penggunaan tertentu.

5. Image Labeling

Image labeling adalah suatu teknik untuk melakukan *anotasi* suatu objek tertentu atau fitur dalam sebuah citra. Dengan melakukan *image labeling*, suatu model *computer vision* dapat mengidentifikasikan sutu objek dalam citra. Sebagai contoh, dalam suatu citra, dilakukan *anotasi* semua objek sebagai mobil, maka model dapat mengerti objek mana saja yang merupakan mobil (Cahyani, 2023). Berikut contoh *image labeling* pada strawberry:



Gambar 2. 7 Contoh *Image Labeling* pada strawberry
(Sumber: *Google.com*)

6. Roboflow

Roboflow adalah platform berbasis web yang menyediakan berbagai fungsi terkait pengelolaan dataset. Platform ini memungkinkan pengguna untuk berbagi serta memproses dataset. Dalam penelitian ini, beberapa fitur yang dimanfaatkan antara lain anotasi objek menggunakan bounding box untuk mendeteksi objek, serta kemampuan melakukan pre-processing dataset, seperti konversi ke grayscale. Roboflow juga menyediakan fitur augmentasi data dengan berbagai metode konvensional, seperti flip, rotasi, pengaturan kecerahan (brightness), eksposur,

shear, dan lainnya (Natalia, 2022). Berikut adalah fungsi dan mekanisme kerja Roboflow:

Fungsi *Roboflow*:

a. Pemrosesan Dataset: *Roboflow* memungkinkan anda mengimpor dataset gambar dari berbagai sumber, seperti lokal, *cloud storage*, dan *platform* lainnya.
b. Pembuatan dan Pelabelan Dataset: Anda dapat menggunakan antarmuka *Roboflow* untuk melakukan pelabelan manual pada gambar, atau menggunakan algoritma deteksi objek otomatis untuk mempercepat proses pelabelan.
c. *Augmentasi* Data: *Roboflow* menyediakan berbagai alat *augmentasi* data untuk meningkatkan keragaman dataset anda, seperti rotasi, pergeseran, dan peningkatan kontras.
d. *Ekspor* Dataset: Setelah dataset anda dipersiapkan, anda dapat mengekspornya dalam format yang sesuai untuk digunakan dalam pelatihan model AI, termasuk format yang didukung oleh berbagai kerangka kerja *machine learning*.

Cara Kerja *Roboflow*:

a. *Impor* Dataset: Anda dapat mengimpor dataset gambar yang ingin Anda gunakan untuk melatih model AI ke dalam platform *Roboflow*. Dataset dapat berasal dari berbagai sumber seperti *Dropbox, Google Drive*, atau penyimpanan lokal anda.

b. Pelabelan dan *Augmentasi*: Setelah dataset di *impor*, anda dapat memulai proses pelabelan objek pada gambar. Anda juga dapat menerapkan *augmentasi* data untuk meningkatkan keragaman dataset anda.
c. Pemrosesan: Setelah dataset dipersiapkan dan diperiksa, anda dapat memprosesnya untuk menghasilkan dataset yang siap digunakan untuk pelatihan model AI.
d. *Ekspor* Dataset: Setelah pemrosesan selesai, Anda dapat *mengekspor* dataset dalam format yang sesuai untuk digunakan dalam pelatihan model AI, seperti format YOLO, *TensorFlow, PyTorch,* dan lainnya.

Dengan menggunakan *Roboflow,* dapat mempercepat proses persiapan dataset dan melatih model AI Anda dengan lebih efisien, tanpa harus menghabiskan waktu untuk tugas-tugas yang *repetitif* dan memakan waktu seperti pelabelan manual dan augmentasi data.

Berikut adalah langkah-langkah umum untuk menggunakan *Roboflow*:

a. Daftar atau Masuk ke Akun *Roboflow*:

   Kunjungi situs *web Roboflow* dan daftar atau masuk ke akun anda.

b. *Impor* Dataset:

   Setelah masuk, anda akan melihat opsi untuk mengimpor dataset. Pilih sumber dataset anda, seperti *Google Drive, Dropbox*, atau unggah dari penyimpanan lokal anda.

c. Pelabelan Data *(Opsional)*:

Jika dataset anda belum dilabeli, anda dapat memilih untuk melabeli data anda secara manual atau menggunakan algoritma deteksi objek otomatis yang disediakan oleh *Roboflow.*

d. *Augmentasi* Data *(Opsional)*:

Roboflow menyediakan berbagai alat *augmentasi* data untuk meningkatkan keragaman dataset anda. Anda dapat menerapkan *augmentasi* seperti rotasi, pergeseran, atau perubahan kontras sesuai kebutuhan anda.

e. Pemrosesan Data:

Setelah dataset anda dilabeli atau di *augmentasi*, anda dapat memprosesnya dengan mengklik tombol "*Process*" atau "*Generate*". Ini akan menghasilkan dataset yang siap digunakan untuk pelatihan model AI.

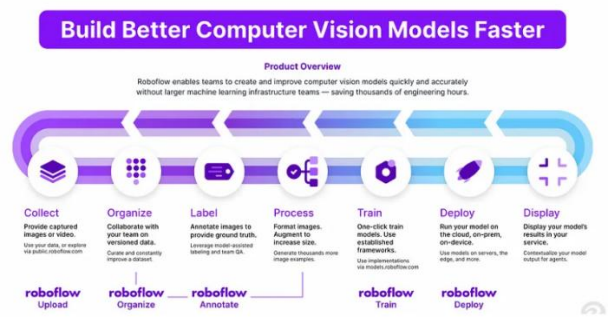
f. *Ekspor* Dataset:

Setelah pemrosesan selesai, anda dapat mengekspor dataset dalam format yang sesuai untuk digunakan dalam pelatihan model AI. *Roboflow* mendukung berbagai format seperti YOLO, *TensorFlow, PyTorch*, dan lainnya.

g. Gunakan Dataset untuk Pelatihan Model AI:

Sekarang dataset anda siap digunakan, anda dapat mengimpornya ke dalam kerangka kerja *machine learning* pilihan Anda (seperti *PyTorch* atau *TensorFlow*) untuk melatih model AI Anda.

Langkah-langkah di atas dapat sedikit bervariasi tergantung pada kebutuhan *spesifik* dan dataset yang di gunakan, tetapi ini memberikan gambaran umum tentang cara menggunakan *Roboflow* untuk mempersiapkan dataset untuk pelatihan model AI. Berikut adalah item yang ada pada *roboflow.*



Gambar 2. 8 Item yang ada pada *roboflow*
(Sumber: *Google.com*)

7. Google Colab

Colab, singkatan dari Colaboratory, adalah sebuah platform berbasis cloud dari Google yang menyediakan lingkungan pengembangan interaktif untuk bahasa pemrograman Python. Colab memanfaatkan teknologi notebook Jupyter dan menawarkan akses gratis ke berbagai sumber daya komputasi, termasuk GPU, yang sangat berguna untuk tugas-tugas komputasi intensif seperti machine learning dan deep learning. Meskipun gratis, penggunaan sumber daya Colab memiliki batasan tertentu dan ketersediaannya dapat bervariasi. Bagi pengguna yang membutuhkan kinerja yang lebih konsisten dan sumber daya yang lebih besar, Colab Pro merupakan opsi yang lebih baik, (Gabriel, 2022).

Berikut adalah beberapa fungsi dan cara kerja *Google Colaboratory*:

a. *Python Environment*: *Colab* menyediakan lingkungan *Python* yang dapat digunakan langsung dari *browser*. Pengguna dapat menulis dan mengeksekusi kode *Python*, serta menggunakan banyak pustaka populer seperti *NumPy*, *Pandas*, *TensorFlow,* dan lainnya.

b. *Interaktif*: *Colab* memungkinkan pengguna untuk menulis dan menjalankan kode *Python* secara *interaktif* dalam sel kode. Hal ini memungkinkan pengguna untuk menguji ide, eksplorasi data, dan membangun model *machine learning* secara langsung.

c. *Kolaborasi*: Seperti namanya, *Colab* memungkinkan pengguna untuk *berkolaborasi* dalam proyek dengan mudah. Pengguna dapat mengundang kolega atau teman untuk melihat atau bahkan mengedit *notebook Colab* secara bersamaan.

d. *Jupyter Notebooks*: *Colab* menggunakan format *notebook Jupyte*r, yang memungkinkan pengguna untuk menulis teks, kode, dan *visualisasi* dalam satu dokumen yang dapat dibagikan dan direproduksi.

e. *Integrasi Google Drive*: *Colab* terintegrasi dengan *Google Drive*, yang memungkinkan pengguna menyimpan *notebook* mereka secara *otomatis*, berbagi dengan orang lain, dan mengaksesnya dari perangkat mana pun yang terhubung ke internet.

f. Akses GPU dan TPU: *Colab* menyediakan akses gratis ke sumber daya GPU dan TPU, yang memungkinkan pelatihan model *machine learning* dengan cepat dan *efisien*.

g. Riwayat Eksekusi: *Colab* menyimpan riwayat eksekusi kode, memungkinkan pengguna untuk melihat dan mengulangi eksekusi kode sebelumnya.

Cara kerja *Google Colaboratory* relatif sederhana:

a. Akses: Pengguna dapat mengakses *Colab* melalui *browser web* dengan masuk ke akun *Google* mereka.

b. Buat atau Impor *Notebook*: Setelah masuk, pengguna dapat membuat *notebook* baru atau mengimpor *notebook* yang sudah ada dari *Google Drive* atau *GitHub*.

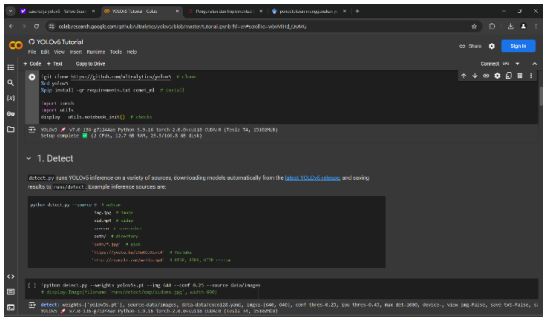
c. Menulis dan Mengeksekusi Kode: Dalam *notebook*, pengguna dapat menulis kode *Python* dalam sel kode dan mengeksekusinya dengan menekan tombol *play* atau menggunakan kombinasi tombol tertentu.

d. Menyimpan dan Berbagi: *Notebook* secara otomatis disimpan di *Google Drive* pengguna. Pengguna dapat membagikan *notebook* dengan mengundang orang lain untuk melihat atau mengeditnya.

e. Akses Sumber Daya: Pengguna dapat menggunakan sumber daya komputasi seperti GPU atau TPU jika diperlukan untuk tugas tertentu. *Google* menyediakan akses gratis untuk sumber daya ini.

Dengan fitur-fitur tersebut, *Colab* telah menjadi alat yang populer di kalangan pengembang, peneliti, dan praktisi *machine learning* karena kemudahan penggunaannya, integrasinya dengan infrastruktur *Google*, dan akses gratis ke

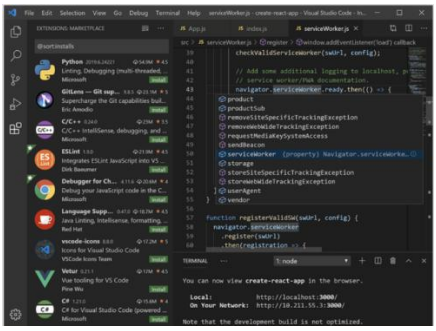
sumber daya komputasi berkinerja tinggi. Berikut merupakan tampilan awal *google colab*.



Gambar 2. 9 Tampilan awal *Google Colab*
(Sumber: *Google.com*)

8. Visual Studio Code

Visual Studio Code atau biasa disebut VS *Code*, adalah sebuah editor kode gratis, *open-source*, dan *cross-platform* yang dikembangkan oleh *Microsoft*. Berdasarkan survei *Stack Overflow Developer*, VS *Code* merupakan *environment* pengembang paling populer di tahun 2019, serta memiliki banyak fitur yang dapat dikostumisasi dan tidak hanya untuk melakukan pengeditan *source code* saja, tapi juga memiliki dukungan *built-in* untuk kolaborasi dan *environments cloud-hosted*. Dukungan *built-in* ini hanya untuk *Javascript*, *Typescript*, HTML, dan CSS. Meskipun begitu, VS *Code* juga mendukung banyak bahasa tambahan, seperti *Python* melalui *extensions* (Cahyani, 2023).



Gambar 2. 10 *Ilustrasi Visual Studio Code*
(Sumber: *Google.com*)

9. Bahasa Pemrograman Python

Python merupakan bahasa pemrograman yang sering diaplikasikan pada pengembangan situs *web*, perangkat lunak, ilmu data, dan *machine learning*. Dengan keunggulan *Python* yang efisien dan mudah dipelajari serta dapat dijalankan di berbagai *platform*, menjadikan *Python* sebagai salah satu bahasa pemrograman popular di kalangan pengembang. Dalam penggunaannya, *python* dibantu dengan berbagai *library*. *Library* adalah sekumpulan kode yang digunakan oleh pengembang dengan tujuan untuk mempercepat dalam penulisan kode dari awal (Cahyani, 2023). Pada *project* YOLOv5 dan *Deep* SORT digunakan beberapa *library* pendukung sebagai berikut:

a. *OpenCv-Python*

   OpenCV atau *Open Source Computer Vision* merupakan *library Python* yang digunakan pengembang dalam pembuatan aplikasi *computer vision*. Penggunaan *library* ini dapat dilakukan pada berbagai bahasa pemrograman yaitu C, C++, *Java*, dan *Python*. Pada *library opencv* memuat ratusan *algoritma computer vision*, serta memiliki struktur modular.

b. *Pillow*

Pillow merupakan *fork* dari PIL yang dibuat oleh Alex Clark dan para kontribusi. *Fork* yaitu repositori baru yang membagikan kode dan penyetelan *visibilitas* dengan *repositori* asli. PIL atau *Python Imaging Library* ciptaan Fredrik Lundh dan para *contributor*, adalah sebuah *library* yang digunakan untuk menambah kapabilitas dalam pemrosesan citra pada *interpreter Python*.

c. PyTorch

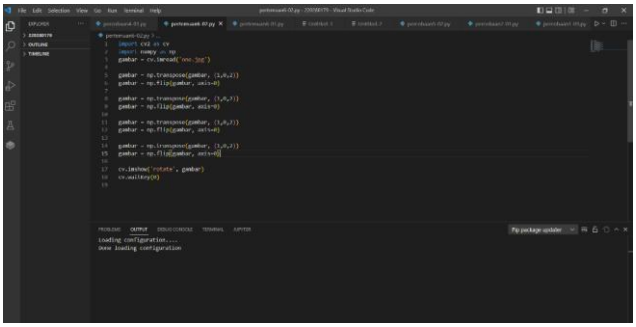
PyTorch adalah *library tensor* yang telah teroptimasi untuk keperluan *deep learning* menggunakan GPU dan CPU yang dikembangkan oleh *Facebook*. Pada *library*, 20 terdapat *package torch* yang berisi struktur data untuk tensor multi-dimensi dan pendefinisian operasi matematika pada tensor.

d. Tb-nightly

Tb-nightly adalah *package Python* sebagai *TensorBoard* yang bertujuan untuk memeriksa dan memahami proses dan grafik dari *tensorflow* pengembang.

e. Imageio

Imageio adalah *library Python* yang menyediakan antarmuka yang dengan mudah membaca dan menulis data citra secara luas, termasuk citra animasi, data *volumetric*, dan format ilmiah.



Gambar 2. 11 *Ilustration* pemprogramanan *Python*
(Sumber: *Google.com*)

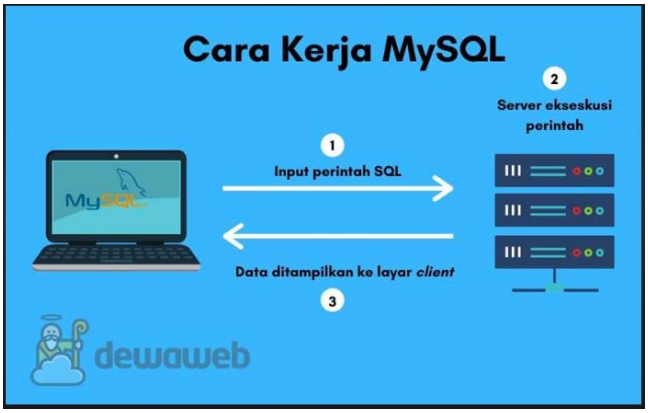
10. MySQL

MySQL adalah sistem basis data yang memiliki kemampuan untuk mengirim dan menerima data dengan cepat serta mendukung banyak pengguna secara bersamaan. MySQL menyediakan dua jenis lisensi, yaitu perangkat lunak bebas (free software) dan perangkat lunak berbagi (shareware). Dalam pembahasan ini, digunakan MySQL versi perangkat lunak bebas karena memungkinkan pengguna untuk mengelola basis data secara gratis, baik untuk keperluan pribadi maupun bisnis, tanpa perlu membeli atau membayar lisensi. MySQL yang berada di bawah lisensi GNU/GPL (General Public License) dapat diunduh melalui situs resminya di http://www.mysql.com (Wahana Komputer, 2010).

MySQL memiliki sejumlah keunggulan dibandingkan dengan sistem basis data lainnya, antara lain:

a. MySQL dikenal oleh banyak ahli sebagai server yang sangat cepat.
b. Sebagai sistem manajemen basis data open source, MySQL menyediakan kode sumber yang terbuka dan bebas digunakan oleh individu maupun organisasi tanpa biaya.

c. Performa MySQL tergolong tinggi namun tetap sederhana dalam penggunaannya.
d. MySQL mendukung bahasa SQL (Structured Query Language).
e. MySQL dapat diakses melalui protokol ODBC (Open Database Connectivity) yang dikembangkan oleh Microsoft, sehingga kompatibel dengan banyak perangkat lunak.
f. Beberapa klien dapat mengakses server secara bersamaan tanpa hambatan.
g. MySQL memungkinkan akses jarak jauh melalui internet dengan pengaturan hak akses tertentu.
h. MySQL mampu menangani data dalam jumlah besar hingga mencapai ukuran gigabyte.
i. MySQL dapat berjalan di berbagai sistem operasi seperti Linux, Windows, Solaris, dan lainnya.



Gambar 2. 12 Cara kerja MySQL
(Sumber: *Google.com*)

11. Flowchart

Flowchart, atau sering dikenal sebagai diagram alir, adalah representasi visual dari algoritma atau serangkaian langkah instruksi yang tersusun secara berurutan dalam

suatu sistem. Diagram ini sering digunakan oleh seorang analis sistem sebagai alat dokumentasi untuk memberikan gambaran logis mengenai sistem yang dirancang, sehingga memudahkan komunikasi dengan programmer dalam proses pengembangannya (Rizqi Rosaly, Andy Prasetyo, 2019).

Tabel 2. 1 Simbol *Flowchart*

SIMBOL	NAMA	FUNGSI
	TERMINATOR	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	PREPARATION	Proses inisialisasi/ pemberian harga awal
	PROSES	Proses perhitungan/ proses pengolahan data
	INPUT/OUTPUT DATA	Proses input/output data, parameter, informasi
	PREDEFINED PROCESS (SUB PROGRAM)	Permulaan sub program/ proses menjalankan sub program
	DECISION	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	ON PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	OFF PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

12. *Blackbox Testing*

Blackbox testing bertujuan untuk mengevaluasi semua fungsi dalam suatu aplikasi. Pengujian ini berfokus pada memverifikasi apakah program bekerja sesuai dengan fungsi yang diinginkan tanpa melibatkan pemahaman terhadap kode sumber yang digunakan (Efendi, 2021).

Berikut adalah beberapa teknik dalam Blackbox Testing:

a. Equivalence Partitioning

Teknik ini membagi input data menjadi beberapa kelompok atau partisi untuk mengurangi jumlah kasus uji yang diperlukan.

b. Boundary Value Analysis

Fokus dari teknik ini adalah pada batas nilai input, baik nilai minimum, maksimum, maupun nilai yang berada di sekitar batas tersebut, untuk mendeteksi adanya error.

c. Fuzzing

Fuzzing adalah teknik untuk menemukan bug atau gangguan dengan memberikan input yang cacat atau tidak biasa, baik secara otomatis maupun semi-otomatis.

d. Cause-Effect Graph

Metode ini menggunakan grafik untuk menggambarkan hubungan antara penyebab dan efek dari suatu error, sehingga memudahkan identifikasi masalah.

e. Orthogonal Array Testing

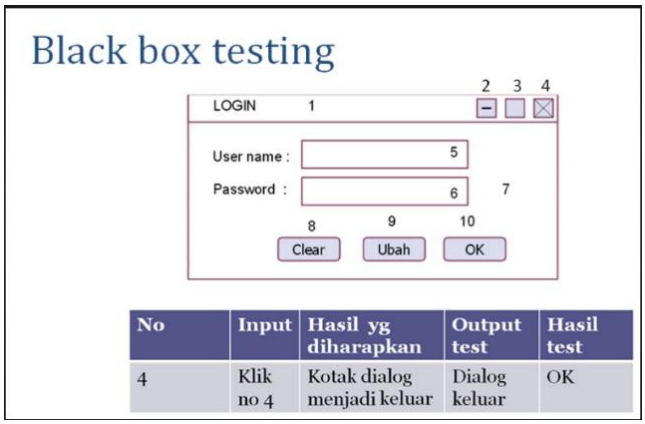
Digunakan ketika domain input relatif kecil namun sulit diterapkan pada skala yang lebih besar, dengan tujuan mengoptimalkan pengujian.

f. All Pair Testing

Teknik ini merancang pasangan-pasangan test case sehingga mencakup semua kombinasi input yang mungkin, untuk memastikan bahwa semua pasangan diuji.

g. State Transition

Pengujian ini digunakan untuk memverifikasi kondisi mesin atau status dan navigasi antarmuka pengguna (UI) dalam bentuk grafis



Gambar 2. 13 *Ilustrasi Blackbox testing*
(Sumber: *Google.com*)

13. *Whitebox Testing*

White Box Testing adalah metode pengujian aplikasi atau perangkat lunak yang berfokus pada pemrograman internal dengan menganalisis dan memeriksa kode sumber untuk memastikan tidak ada kesalahan. Jika modul yang diuji menghasilkan output yang tidak sesuai dengan spesifikasi, kode tersebut akan dikompilasi kembali dan diuji ulang hingga hasil yang diinginkan tercapai. Secara singkat, White Box Testing mengevaluasi aplikasi atau perangkat lunak dari sisi kode sumbernya, tanpa memperhatikan antarmuka pengguna atau tampilan (Irma, 2020).

Beberapa teknik dalam White Box Testing adalah sebagai berikut:

a. Basis Path Testing

Metode ini memungkinkan perancang test case untuk mengukur kompleksitas logis dari prosedur desain dan menggunakan hasil pengukuran

ini untuk menentukan jalur dasar yang perlu diuji. Test case yang dibuat akan menjamin bahwa setiap pernyataan dalam program akan dieksekusi minimal sekali selama pengujian.

b. Flow Graph

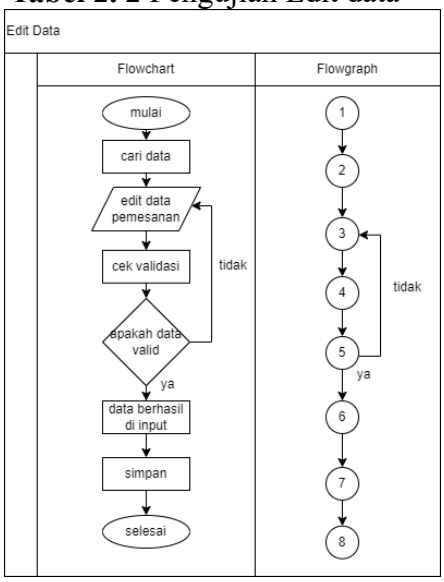
Flow graph adalah representasi grafis sederhana yang digunakan untuk menggambarkan alur kontrol dalam program.

c. Cyclomatic Complexity

Cyclomatic complexity digunakan untuk mengukur jumlah jalur yang harus diuji dalam program. Ini adalah metrik perangkat lunak yang memberikan ukuran kuantitatif dari kompleksitas logis program. Nilai cyclomatic complexity menentukan jumlah jalur independen dalam jalur dasar program dan memberikan jumlah tes minimal yang diperlukan untuk memastikan setiap pernyataan dalam program diuji setidaknya sekali.

Berikut adalah salah satu contoh cara kerja pengujian *Whitebox testing* yang di lakukan oleh Mohamd Arifin, (2020).

Tabel 2. 2 Pengujian Edit data



Berdasarkan gambar flowgraph Edit Data di atas, proses perhitungan dapat dilakukan sebagai berikut:

1. Menghitung Cyclomatic Complexity V(G) dari Edge dan Node:

   Menggunakan rumus: $V(G) = E - N + 2$

   Di mana:

   E (edge) = 8

   N (node) = 8

   Predikat Node (P) = 1

   Penyelesaian:

   $V(G) = E - N + 2$

   $= 8 - 8 + 2 = 2$

   Predikat (P) = P + 1

   $= 1 + 1 = 2$

2. Berdasarkan perhitungan Cyclomatic Complexity, flowgraph di atas memiliki Region = 2
3. Independent Path pada flowgraph di atas adalah:

   - Path 1 = 1 – 2 – 3 – 4 – 5 – 3

   - Path 2 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8

4. Grafik *Matriks*

Tabel 2. 3 Grafik *Matriks* pada *Flowgraph* Edit Data

	1	2	3	4	5	6	7	8	E – 1 = 0
1		1							1 – 1 = 0
2			1						1 – 1 = 0
3				1					1 – 1 = 0
4					1				1 – 1 = 0
5			1			1			2 – 1 = 1
6							1		1 – 1 = 0
7								1	1 – 1 = 0
8									0
Sum (E) + 1									1 + 1 =2

14. WebCam

WebCam adalah jenis kamera kecil yang dapat dipasang pada komputer atau laptop untuk menyiarkan video secara langsung. Seperti kamera digital pada umumnya, webcam berfungsi dengan menangkap cahaya melalui lensa kecil di bagian depan, menggunakan detektor cahaya mikroskopis yang terdapat pada

microchip penerima gambar, biasanya dengan teknologi Charge-Couple Device (CCD) atau CMOS image sensor. Gambar yang ditangkap kemudian diproses secara digital dan dapat disebarkan melalui internet. Berbeda dengan kamera digital, webcam tidak dilengkapi dengan media penyimpanan seperti memori atau kartu flash, karena tujuan utamanya adalah merekam dan mengirimkan gambar secara langsung tanpa perlu menyimpannya. Beberapa jenis webcam menggunakan kabel USB untuk menghubungkan perangkat (Sinta, 2020).

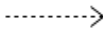
Gambar 2. 14 *Webcam*
(Sumber: *Google.com*)

15. *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah standar bahasa yang umum digunakan dalam industri untuk mendefinisikan kebutuhan, melakukan analisis dan perancangan, serta menggambarkan arsitektur dalam pemrograman berbasis objek (Razak, 2022).

Unified Modelling Language (UML) sudah banyak digunakan untuk membuat desain dari suatu sistem, adapun beberapa jenis UML yang sering digunakan seperti *Use Case Diagram, Sequence Diagram, Activity Diagram* dan *Class Diagram.* Adapun jenis simbol dari UML sebagai berikut:

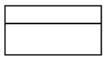
Tabel 2. 4 *Symbol Use Case Diagram*

No	Gambar	Nama	Keterangan
1		*Actor*	Orang, organisasi, atau sistem yang berinteraksi dengan sistem yang sedang dimodelkan. Ini direpresentasikan oleh gambar manusia.
2		*Dependency*	Penggunaan *Dependency* pada diagram *use case* membantu menyederhanakan dan mengorganisir kompleksitas hubungan antar elemen dalam sistem, yang pada gilirannya mendukung pemahaman yang lebih baik terhadap kebutuhan dan *fungsionalitas* sistem.
3		*Generalization*	*Generalization* membantu dalam memahami *hierarki use case*, di mana *use case* yang lebih umum dapat memiliki satu atau lebih *use case* anak yang lebih spesifik.
4		*Include*	*Include* digunakan untuk menggambarkan bagaimana suatu *use case* dapat memasukkan (menggunakan) *fungsionalitas* dari *use case* lain untuk melengkapi tugas atau skenario tertentu.
5		*Extend*	menggambarkan situasi di mana sebuah *use case* dapat menambah atau memperluas *fungsionalitas* dari *use case* lain dalam suatu sistem
6		*Association*	Hal yang menghubungkan satu objek dengan objek lainnya..
7		*System*	Menyebutkan paket yang menggambarkan sistem dalam ruang lingkup yang terbatas.
8		*Use Case*	Penjabaran urutan tindakan yang dilakukan oleh sistem untuk menghasilkan hasil yang dapat diukur bagi seorang aktor.

Lanjutan Tabel 2. 4

No.	Gambar	Nama	Keterangan
9		*Collaboration*	Interaksi antara aturan dan elemen-elemen lain yang bekerja bersama untuk menghasilkan perilaku yang lebih besar daripada sekadar jumlah elemen-elemen tersebut (sinergi).
10		*Note*	Elemen fisik yang ada saat aplikasi beroperasi dan menggambarkan sumber daya komputasi yang digunakan.

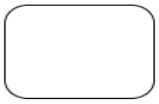
Tabel 2. 5 *Symbol Class Diagram*

No.	Gambar	Nama	Keterangan
1		*Generalization*	Hubungan di mana objek turunan berbagi perilaku dan struktur data dengan objek yang berada di atasnya, yaitu objek induk.
2		*Nary Association*	Upaya untuk mencegah keterkaitan dengan lebih dari dua objek.
3		*Class*	Sekelompok objek yang memiliki atribut dan operasi yang serupa.
4		*Collaboration*	Penjelasan mengenai rangkaian tindakan yang dilakukan oleh sistem, yang menghasilkan hasil yang dapat diukur bagi seorang aktor.
5		*Realization*	Tindakan yang sebenarnya dijalankan oleh suatu objek.
6		*Association*	Elemen yang menghubungkan satu objek dengan objek lainnya.

Tabel 2. 6 *Symbol Sequence Diagram*

No.	Gambar	Nama	Keterangan
1		*LifeLine*	Entitas objek, yang berfungsi sebagai antarmuka yang saling berinteraksi.
2		*Message*	Rincian mengenai interaksi antar objek yang mencakup informasi tentang aktivitas yang berlangsung.
3		*Message*	Deskripsi tentang interaksi antar objek yang mencakup informasi terkait aktivitas yang berlangsung.

Tabel 2. 7 *Symbol Activity Diagram*

No.	Gambar	Nama	Keterangan
1		*Actifity*	Menunjukkan bagaimana setiap kelas antarmuka berinteraksi satu sama lain.
2		*Action*	Keadaan sistem yang menggambarkan pelaksanaan suatu tindakan.
3		*Initial Node*	Proses pembentukan atau inisialisasi suatu objek.
4		*Actifity Final Node*	Proses pembentukan dan penghapusan objek.
5		*Fork Node*	Sebuah aliran yang pada titik tertentu bercabang menjadi beberapa aliran.

B. Kajian Penelitian Terdahulu

Sebuah penlitian perlu adanya bahan pertimbangan atau bahan acuan terhadap penelitian yang telah dilakukan sebelumnya, melalui berbagai sumber seperti skripsi dan jurnal yang relevan dengan penelitian terkait, agar memudahkan peneliti dalam menentukan langkah awal sebagai konsep untuk meneliti.

1. (Fu'adi dkk., 2024)Akademi Komunitas Negeri Pacitan. "Pengembangan Sistem Monitoring Kehadiran Mahasiswa Menggunakan YOLO Pendeteksi Obyek dan Pengenal Wajah Opencv". Tujuan dari penelitian ini untuk mengatasi permasalahan kehadiran mahasiswa yang seringkali memakan waktu dan rentan terhadap kesalahan manusiawi.
2. (Christopher, Silvia, 2022). AMIK JTC Semarang. "Deteksi Kematangan Buah Pepaya Menggunakan Metode Algoritma YOLO Berbasis Android". Tujuan dari penelitian adalah untuk dapat mengenali kematangan buah pepaya dengan bantuan komputer untuk mengklasifikasikan kematangan

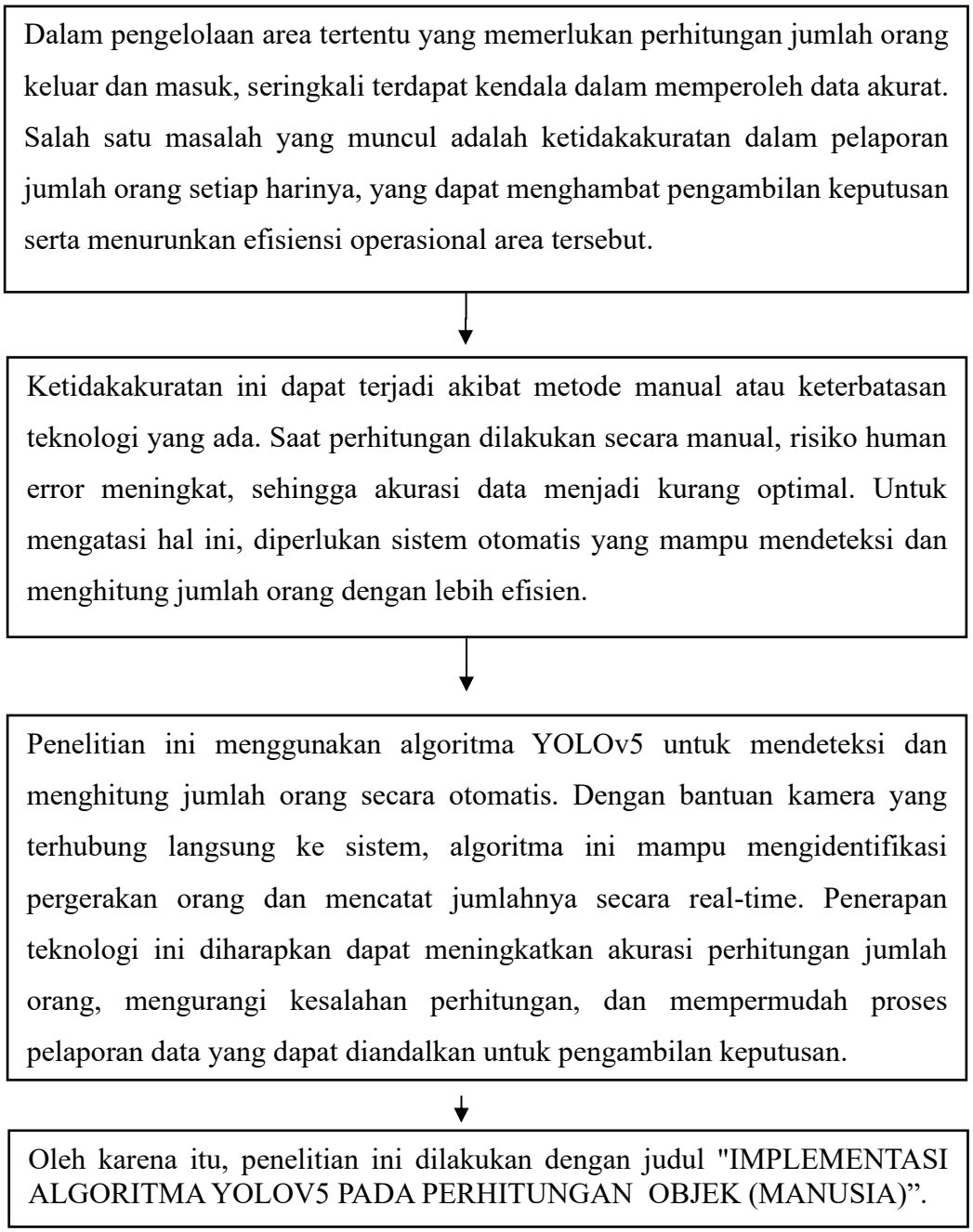
buah pepaya dengan benar yaitu dengan menggunakankemampuan teknologi pendeteksi objek.

3. (Fauzi Bagaskara, (2017).Universitas Islam Indonesia. “Evaluasi Sistem Antrian pada Usaha Pencucian Mobil di Prima Gold Carwash Yogyakarta”. Penelitian ini memfokuskan pada analisis sistem antrian car wash di Prima Gold Carwash Yogyakarta untuk mengoptimalkan sistem antrian pada proses pencucian mobil. Sehubungan dengan tingginya minat pelanggan untuk mendapatkan pelayanan pencucian mobil dengan kualitas yang baik dan waktu pelayanan yang cepat, dengan sistem antrian yang baik dan optimal dapat meningkatkan kualitas pelayanan.
4. (Sugandi & Hartono, 2022). Politeknik Negeri Bandung. “Implementasi Pengolahan Citra pada Quadcopter untuk Deteksi Manusia Menggunakan Algoritma YOLO”. Penelitian ini bertujuan untuk pemetaan lahan, pemantauan udara, bahkan evakuasi korban bencana.

Berdasarkan beberapa penelitian di atas menggunakan algoritma YOLO dalam pendeteksian objek dan faktor pembeda dari penelitian terletak pada objek sasaran deteksi dan pengimplementasiannya. Dengan mempertimbangkan kelebihan dan kekurangan pada penelitian terkait, pada penelitian ini, maka dilakukan perancangan sistem perhitungan keluar masuk manusia menggunakan algoritma YOLOv5. sistem ini menggunakan algoritma YOLOv5 karena memiliki performa paling stabil untuk sistem *realtime.*

C. Kerangka Pikir

Untuk memahami alur dari penelitian maka dibuatlah kerangka berpikir yang dibuat dalam bentuk diagram:



BAB III

METODE PENELITIAN

A. Jenis Penelitian

Penelitian ini menggunakan metode pengembangan sistem *(System Development Methodology)* yang digunakan oleh peneliti untuk merancang, mengembangkan, dan menguji sistem berbasis algoritma YOLOv5 dalam memproses, menganalisis, dan memahami gambar serta video. Metode ini melibatkan langkah-langkah seperti Perancangan sistem, Detail Sistem, serta pengujian dan evaluasi kinerja sistem untuk memastikan keandalan dan efektivitasnya.

B. Waktu dan Tempat Penelitian

Lokasi penelitian di laksanakan di Parepare selama ± 3 bulan.

Tabel 3. 1 Jadwal Pelaksanaan Penelitian

No	Uraian Kegiatan	Tahun 2024		
		Agustus	September	Oktober
1	Observasi			
2	Studi Literatur			
3	Perancangan Sistem			
4	Pembuatan Sistem			
5	Pengujian			

C. Alat dan Bahan

Untuk melakukan proses penelitian dalam pembuatan sistem, maka diperlukan alat dan bahan guna mendukung kegiatan penelitian tersebut. Berikut alat dan bahan yang dibutuhkan dalam penyelesaian penelitian ini sebagai berikut:

1. Alat

Alat yang digunakan untuk membuat sistem perhitungan objek (manusia) dapat dilihat pada tabel berikut:

Tabel 3. 2 Spesifikasi Alat

Perangkat	**Spesifikasi**
Laptop	*Asus Vivobook* AMD *Ryzen* 5 5800H, RAM 24 GB, SSD 512
Webcam	Tipe USB *webcam, resolusi* 1920 x 1080 px

2. Bahan

Bahan yang digunakan untuk membuat sistem perhitungan objek (manusia) dapat dilihat pada tabel berikut:

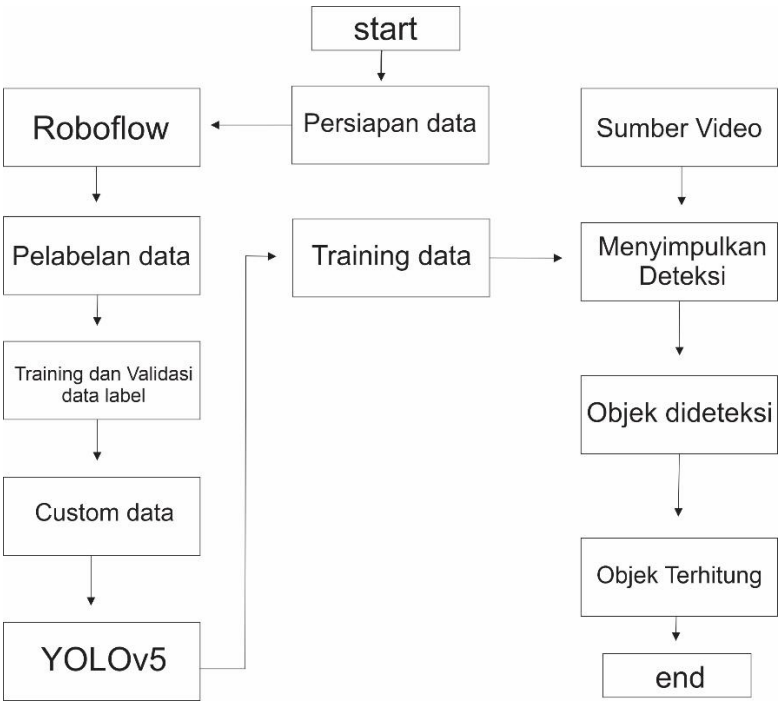
Tabel 3. 3 Spesifikasi bahan

Perangkat	**Spesifikasi**
Python	3.12.1
YOLO	Model YOLOv5
Google Colaboratory	-
Roboflow	-
Visual Studio Code	Versi 1.70.2
Dataset (Manusia)	File png. dan jpg.

D. Rancangan Sistem

Perancangan sistem implementasi algoritma YOLOv5 pada perhitungan objek (manusia) ini meliputi perancangan perangkat keras dan perancangan perangkat lunak. Berikut blok diagram dan sistem yang akan di buat:

1. *Block Diagram Sistem*

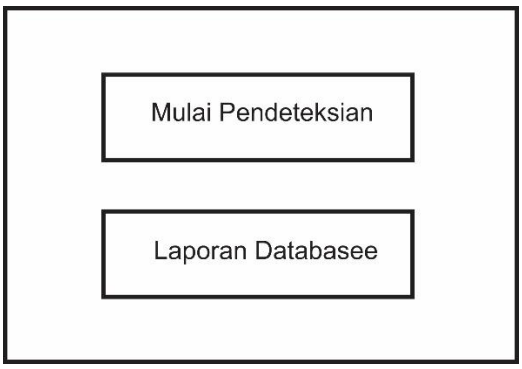


Gambar 3. 1 *Block Diagram Sistem*

a. Dalam perancangan sistem yang pertama di siapkan adalah datanya.
b. Selanjutnya menggunakan *Roboflow* Dengan menggunakan *Roboflow* ini dapat membagikan dataset sekaligus memproses dataset tersebut contohnya untuk pelabelan data dan training data.
c. Setelah melakukan pelabelan dan training data kemudian melakukan *Custom* data dan memasukkan Algoritma YOLOv5.
d. Setelah melakukan pelabelan, training data dan *custom* data, maka perlu untung di training data lagi secara keseluruhan.

e. Ketika sumber Video telah ada maka akan di Proses, apabila objek yang telah melewati pelabelan dan validasi masuk dalam video, maka akan terdeteksi dan akan terhitung ketika telah menyentu garis yang telah di tetapkan pada algoritma YOLOv5.

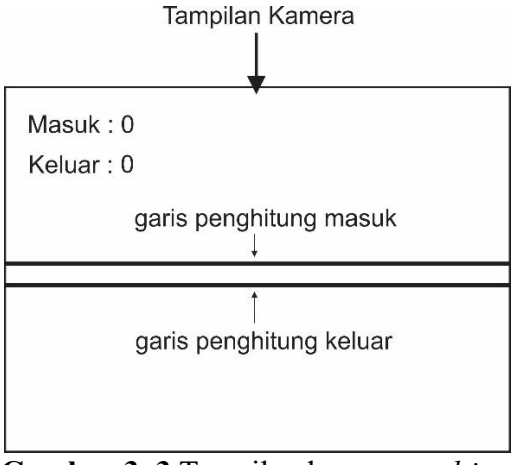
2. Desain *Interface*



Gambar 3. 2 Tampilan Awal

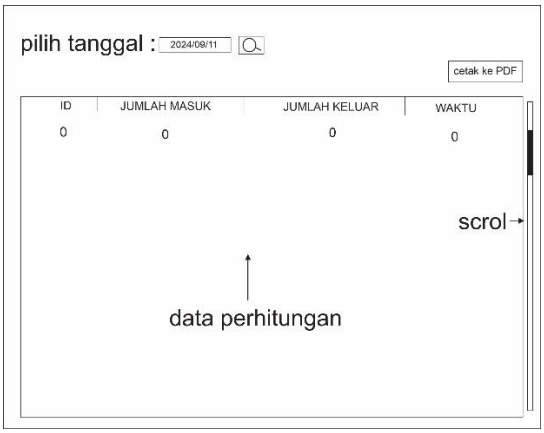
a. Berdasarkan Gambar 3. 2 di atas, ada 2 (Dua) pilihan menu yang pertama "Mulai Pendeteksian" untuk membuka tampilan *realtime.*

b. Menu yang kedua "Laporan Database" untuk membuka data yang telah tersimpan.



Gambar 3. 3 Tampilan kamera *realtime*

a. Gambar 3. 3 merupakan tampilan kamera *realtime* lengkap dengan 2 (dua) perhitungan keluar dan masuk, serta memiliki garis yang berfungsi untuk menghitung objek (manusia).



Gambar 3. 4 Tampilan data perhitungan

a. Pada gambar 3. 4 di atas merupakan tampilan Data perhitungan objek (manusia) serta menyediakan pencarian sesuai tanggal dan menampilkan data yang telah di hitung mulai dari data harian, bulanan dan tahunan. Tampilan di atas juga menyediakan menu untuk di cetak ke pdf.

E. Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan sebagai berikut:

1. Observasi

Metode pengumpulan data melalui pengamatan langsung. atau peninjauan secara cermat pada setiap tempat lalulintas keluar masuk.

2. Studi Literatur

Pencarian informasi dalam bentuk membaca atau mengambil informasi dari berbagai literatur yang berkaitan dengan judul penelitian contohnya jurnal, artikel,

laporan penelitian. setelah data penelitian terkumpul, kemudian dikelolah sehingga dapat diperoleh kesimpulan yang objektif dari peneliti.

F. Tahapan Penelitian

Tahap penelitian ini dilakukan dengan beberapa tahap yaitu:

1. Persiapan Penelitian

Pada tahap ini, peneliti mempersiapkan berbagai referensi seperti buku dan artikel yang relevan dengan topik yang diteliti, serta perangkat keras dan perangkat lunak yang diperlukan dalam penelitian.

2. Pengumpulan Data

Peneliti mengumpulkan informasi yang mendalam mengenai topik penelitian melalui kajian pustaka.

3. Perancangan Sistem

Tahap ini melibatkan perancangan aplikasi yang disesuaikan dengan solusi yang diperlukan untuk mengatasi masalah yang telah diidentifikasi.

4. Pembuatan Sistem

Sistem dibangun dengan mengikuti algoritma yang telah ditentukan sebelumnya.

5. Pengujian

Setelah sistem dirancang, peneliti melakukan pengujian untuk memastikan sistem bekerja sesuai harapan. Jika ditemukan kekurangan atau kerusakan pada hasil rancangan, maka proses akan kembali ke tahap analisis untuk diperbaiki.

G. Metode Pengujian

Setelah aplikasi telah dibuat maka dilakukan pula pengujian pada aplikasi tersebut. Dalam penelitian ini digunakan 2 (Dua) metode untuk melakukan pengujian pada sistem aplikasi yaitu *blackbox testing* dan *whiitebox testing* sebagai berikut:

1. Blackbox Testing

Pada pengujian *Blackbox testing* ini akan menguji beberapa tampilan atau fungsi yang ada pada sistem, apakah berjalan sesuai yang diinginkan atau tidak sesuai.

2. Whitebox Testing

Jika pengujian *Blackbox testing* akan menguji beberapa tampilan atau fungsi yang ada pada sistem, maka *Whitebox Testing* akan memeriksa dan menganalisis kode program ada yang salah atau tidak.

BAB IV

HASIL DAN PEMBAHASAN

A. Perancangan Sistem

Aplikasi yang dikembangkan dalam tugas akhir ini adalah implementasi Algoritma YOLOv5 pada perhitungan objek (manusia). Sistem ini mencakup 2 (dua) komponen utama, yaitu perangkat keras dan perangkat lunak, sehingga rancangan sistemnya terbagi menjadi rancangan perangkat keras dan rancangan perangkat lunak.

1. Rancangan Perangkat Keras

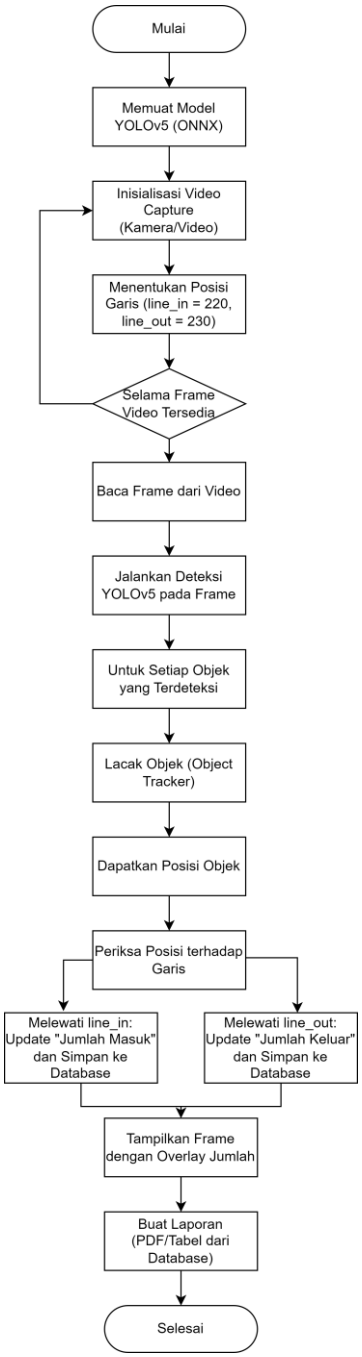
Adapun perangkat keras yang digunakan untuk mengimplementasikan rancangan pada penelitian ini adalah berupa *Laptop* ASUS Vivobook dengan spesifikasi AMD Ryzen 5 (lima) 5800H, RAM 24 GB, SSD 512 dan menggunakan kamera *Webcam* dengan spesifikasi Tipe USB *Webcam*, resolusi 1080 px.

2. Rancangan Perangkat Lunak

Perangkat lunak yang digunakan dalam implementasi rancangan penelitian ini mencakup beberapa komponen untuk pengembangan perangkat lunak, yaitu sistem operasi *Windows* 11 *enterprise* 64-bit, bahasa pemrograman *Python*, serta berbagai *tools* seperti *Visual Studio Code*, XAMPP, *Google Colab*, dan *Roboflow*. Jenis file yang relevan juga digunakan selama penelitian ini. Selanjutnya, perancangan sistem dilakukan untuk memberikan gambaran yang jelas dan

komprehensif mengenai desain dan implementasi sistem yang dikembangkan sebagai berikut:

a. Diagram alir (*Flowchart*)



Gambar 4. 1 *Flowchart*

Berikut adalah deskripsi lengkap dari diagram alur (*flowchart*) sistem:

1) Mulai

Proses dimulai dengan menjalankan program untuk melakukan perhitungan jumlah objek (manusia) yang masuk dan keluar.

2) Memuat Model YOLOv5 (ONNX)

Sistem memuat model deteksi YOLOv5 yang disimpan dalam format ONNX. Model ini akan digunakan untuk mendeteksi objek (manusia) di dalam video atau kamera.

3) Inisialisasi Video *Capture* (Kamera/Video)

Sistem menginisialisasi video *capture* dari kamera atau video yang digunakan sebagai input untuk melakukan deteksi.

4) Menentukan Posisi Garis (*line_in* = 220, *line_out* = 230)

Sistem menentukan posisi garis masuk *(line_in)* dan garis keluar *(line_out)*. Garis ini akan digunakan sebagai acuan untuk menentukan apakah objek (manusia) masuk atau keluar berdasarkan pergerakan mereka melewati garis tersebut.

5) Selama *Frame* Video Tersedia

Sistem akan terus melakukan deteksi selama masih ada *frame* video yang tersedia atau selama video masih berlangsung

6) Baca *Frame* dari Video

Sistem membaca *frame* dari video yang sedang berlangsung. *Frame* ini kemudian akan diproses untuk melakukan deteksi objek.

7) Jalankan Deteksi YOLOv5 pada *Frame*

Pada setiap *frame* yang dibaca, sistem menjalankan model YOLOv5 untuk mendeteksi objek (penumpang) di dalam *frame* tersebut.

8) Untuk Setiap Objek yang Terdeteksi

Sistem akan melakukan tindakan berikutnya untuk setiap objek (manusia) yang berhasil terdeteksi oleh YOLOv5 di dalam frame.

9) Lacak Objek *(Object Tracker)*

Sistem melacak pergerakan setiap objek yang terdeteksi menggunakan algoritma pelacak objek. Pelacak ini memungkinkan sistem untuk mengetahui pergerakan objek dari satu *frame* ke frame berikutnya.

10) Dapatkan Posisi Objek

Sistem mendapatkan posisi terkini dari setiap objek yang terdeteksi dan dilacak.

11) Periksa Posisi terhadap Garis

Sistem memeriksa posisi setiap objek terhadap garis *line_in* dan *line_out*. Langkah ini menentukan apakah objek tersebut melewati garis masuk atau keluar.

12) Melewati *line_in*: Update "Jumlah Masuk" dan Simpan ke Database

Jika objek melewati garis masuk *(line_in)*, sistem akan meningkatkan nilai jumlah masuk dan menyimpan data tersebut ke dalam database.

13) Melewati *line_out*: Update "Jumlah Keluar" dan Simpan ke Database

Jika objek melewati garis keluar *(line_out)*, sistem akan meningkatkan nilai jumlah keluar dan menyimpan data tersebut ke dalam database.

14) Tampilkan *Frame* dengan *Overlay* Jumlah

Sistem menampilkan *frame* video yang sedang diproses dengan *overlay* yang menunjukkan jumlah penumpang yang masuk dan keluar hingga saat ini.

15) Buat Laporan (PDF/Tabel dari Database)

Sistem menghasilkan laporan dari data yang tersimpan di database. Laporan ini dapat berupa tabel atau file PDF yang berisi informasi jumlah objek (manusia) yang masuk dan keluar.

16) Selesai

Proses berakhir ketika video selesai atau sistem dihentikan.

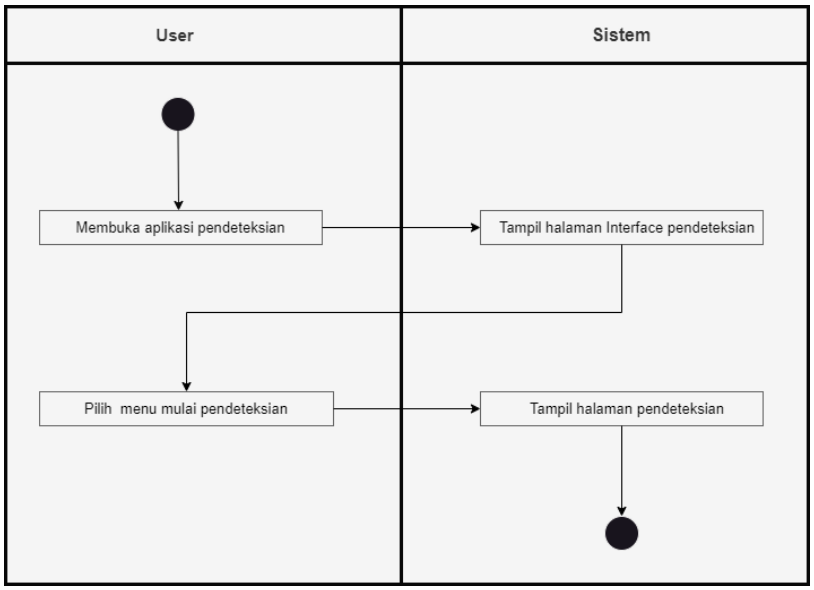
b. Perancangan model UML

Perancangan sistem dilakukan dengan memanfaatkan diagram UML, yaitu diagram aktivitas *(Activity Diagram)* dan diagram urutan *(Sequence Diagram)*.

1) *Activity* diagram

Activity diagram memodelkan alur kerja *(workflow)* dari suatu proses serta menunjukkan urutan langkah-langkah dalam proses tersebut. Pada sistem yang dirancang oleh penulis, *activity* diagram yang disusun di antaranya adalah.

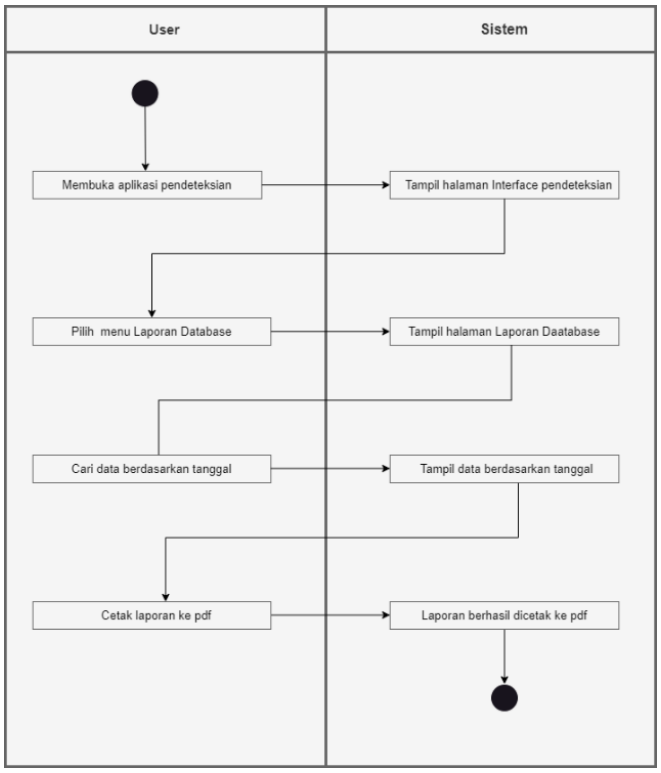
a) *Activity* Diagram Pendeteksian



Gambar 4. 2 *Activity Diagram Pendeteksian*

Gambar 4. 2 diatas merupakan *activity* diagram pendeteksian, dimana jika *user* akan membuka aplikasi pendeteksian maka akan ditampilkan halaman *Interface* Pendeteksian, untuk masuk pada pendeteksian pilih menu mulai pendeteksian maka sistem akan menampilkan halaman pendeteksian.

b) *Activity* Diagram Laporan Database



Gambar 4. 3 *Activity Diagram* Laporan Database

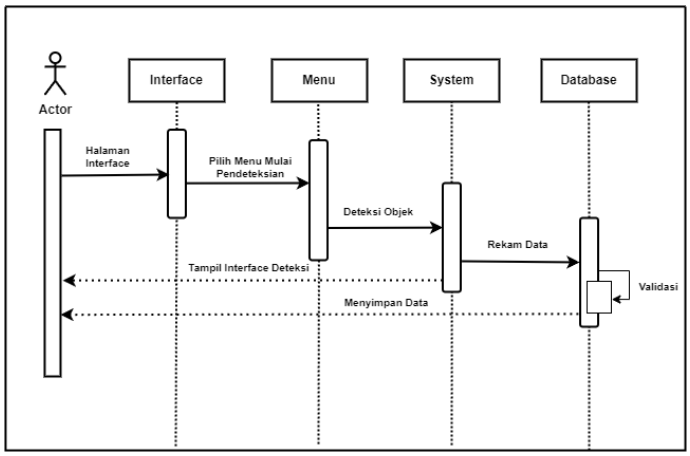
Gambar 4. 3 diatas merupakan *activity* diagram laporan database, dimana setelah *user* masuk pada sistem dan memilih menu laporan database, maka sistem akan menampilkan halaman laporan database yang berisi semua data perhitungan objek (manusia), jika *user* mencari data berdasarkan tanggal maka sistem akan menampilkan data berdasarkan tanggal yang telah di pilih, dan jika *user* akan mencetak laporan ke file pdf maka sistem akan mencetak data yang telah dipilih ke file pdf.

2) *Sequence* Diagram

Diagram *sequence* adalah salah satu jenis diagram interaksi yang menggambarkan alur proses yang terjadi dalam sebuah sistem. Diagram ini

menunjukkan urutan pesan yang dikirimkan antar objek atau komponen dalam sistem, serta waktu kapan pesan tersebut dikirim. Diagram *sequence* digunakan untuk memahami bagaimana berbagai elemen dalam sistem berinteraksi secara berurutan untuk menyelesaikan sebuah proses atau mencapai tujuan tertentu.

a) *Sequence* Diagram Pendeteksian



Gambar 4. 4 *Sequence Diagram* Pendeteksian

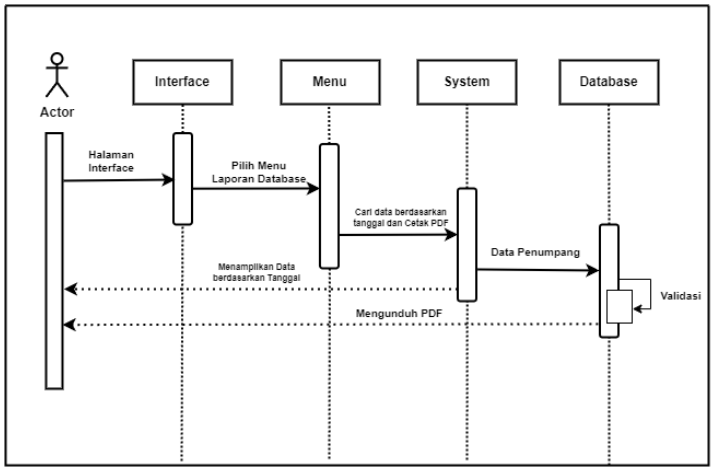
Gambar 4. 4 diatas adalah diagram *sequence* yang menunjukkan alur proses pendeteksian objek dalam sistem yang digunakan oleh pengguna. Berikut adalah deskripsi dari setiap langkah dalam diagram:

1. *Actor* (pengguna) memulai dengan membuka Halaman *Interface* dari sistem.
2. Pada *Interface*, pengguna memilih Menu Mulai Pendeteksian untuk memulai proses pendeteksian objek.
3. Menu kemudian mengirimkan perintah ke *System* untuk memulai proses Deteksi Objek.

4. *System* melakukan deteksi objek dan merekam data hasil pendeteksian. Data ini dikirimkan ke Database.
5. Database melakukan *Validasi* data yang diterima dari *System* untuk memastikan data tersebut benar sebelum disimpan.
6. Setelah data tervalidasi dan tersimpan di *Database*, Menu akan menampilkan *Interface* Deteksi kepada pengguna.
7. *Interface* memberikan konfirmasi kepada pengguna bahwa data telah berhasil disimpan.

Diagram ini menggambarkan aliran proses pendeteksian dari saat pengguna memulai hingga data hasil pendeteksian tersimpan di *database* dengan *validasi* yang tepat.

b) *Sequence* Diagram Laporan Database



Gambar 4. 5 *Sequence Diagram* Laporan Database

Gambar 4. 5 diatas adalah diagram *sequence diagram* Laporan *Database*, yang menggambarkan alur interaksi antar komponen dalam sistem laporan

pendeteksian objek (manusia). Berikut penjelasan dari setiap langkah dalam diagram:

1. *Actor* (pengguna) memulai dengan membuka Halaman *Interface* dari sistem.
2. Pada *Interface*, pengguna memilih Menu Laporan *Database* untuk melihat data laporan pendeteksian objek (manusia).
3. Setelah itu, Menu akan mengirimkan permintaan ke *System* untuk mencari data berdasarkan tanggal tertentu dan mencetaknya dalam format PDF.
4. *System* kemudian berinteraksi dengan *Database* untuk mengambil Data objek (manusia) yang sesuai dengan permintaan, dan *Database* melakukan proses *Validasi* terhadap data yang diminta.
5. Setelah data divalidasi dan dikirimkan kembali ke *System*, hasil data yang ditemukan akan dikembalikan ke Menu.
6. Menu kemudian mengirimkan *respons* ke *Interface* untuk menampilkan data berdasarkan tanggal yang diminta.
7. Pengguna juga memiliki opsi untuk Mengunduh PDF dari data yang telah ditampilkan melalui *Interface*.

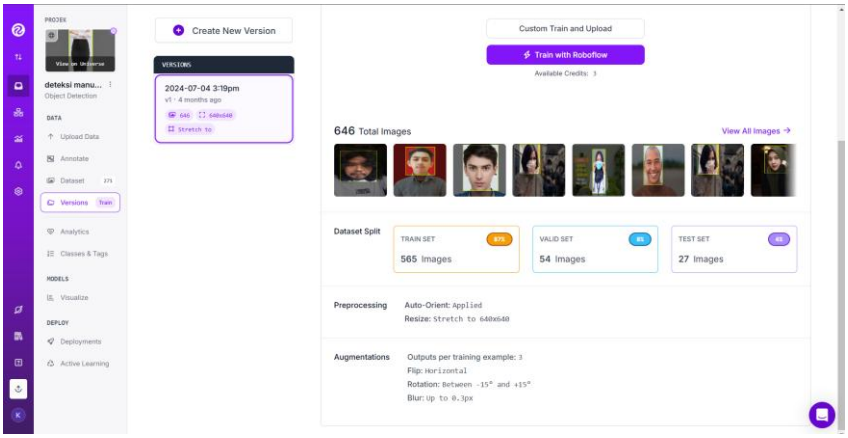
Diagram ini menunjukkan aliran proses dari interaksi pengguna hingga pengambilan dan pengunduhan data laporan dari sistem.

B. Detail Aplikasi

Berikut adalah rincian aplikasi yang menunjukkan langkah awal pembuatan sistem dan berbagai fitur yang tersedia. Adapun detail pembuatan sistem mencakup elemen-elemen sebagai berikut:

1. Langkah awal pembuatan sistem

a. Pembuatan dataset menggunakan *roboflow*



Gambar 4. 6 Pembuatan Dataset menggunakan *roboflow*

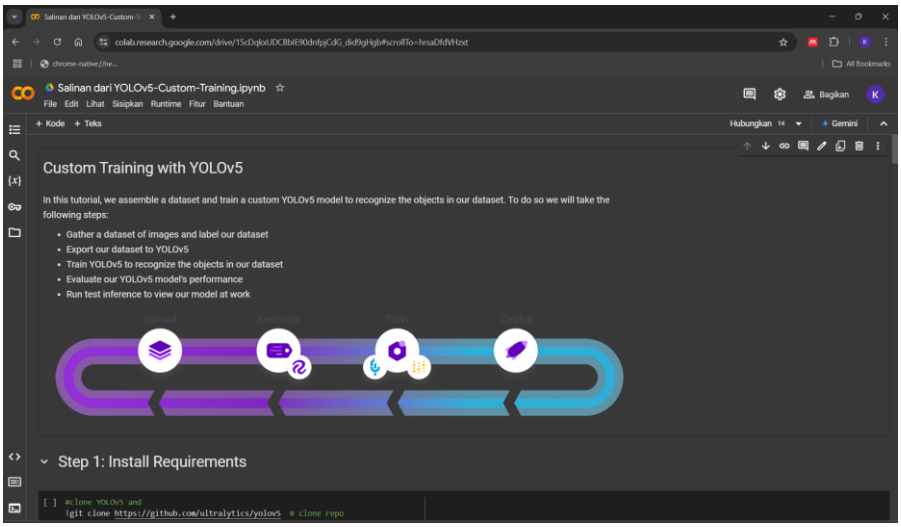
Gambar 4. 6 tersebut menunjukkan antarmuka pengguna dari *platform* pelatihan model pembelajaran mesin, yaitu objek deteksi. Berikut uraikan langkah demi langkah:

1) Proyek dan Data
   a) Proyek, *Platform* ini mengelola proyek-proyek pembelajaran mesin, dalam hal ini "deteksi manusia".
   b) Data, Terdapat data yang diunggah untuk proyek ini, berupa 271 gambar. Gambar-gambar ini telah dianotasi (diberi label) untuk objek yang ingin dideteksi.

2) Anotasi dan Total Gambar
    a) Anotasi, *Platform* ini memungkinkan untuk menanotasi data, yaitu memberi label pada objek-objek yang ada dalam gambar.
    b) Total Gambar, Terdapat total 646 gambar dalam dataset. Gambar-gambar ini dibagi menjadi tiga set yaitu *train*, validasi, dan tes.
3) Pemisahan Dataset (Dataset Split)
    a) Train Set, 87% dari data digunakan untuk melatih model. Total 565 gambar digunakan untuk pelatihan.
    b) Validasi Set, 8% dari data digunakan untuk memvalidasi model selama pelatihan. Total 54 gambar digunakan untuk validasi.
    c) Tes Set, 4% dari data digunakan untuk menguji model setelah pelatihan. Total 27 gambar digunakan untuk tes.
4) Pra-Pemrosesan (*Preprocessing*)
    a) *Auto-Orient*, *Platform* secara otomatis mengoreksi orientasi gambar sebelum pemrosesan.
    b) *Resize*, Gambar diubah ukurannya menjadi 640x640 piksel untuk konsistensi dan efisiensi pelatihan.
5) *Augmentasi* Data
    a) *Flip*, Gambar dibalik secara *horizontal* untuk meningkatkan variasi data dan mencegah *overfitting*.
    b) *Rotation*, Gambar diputar secara acak antara -15° dan +15° untuk meningkatkan ketahanan model terhadap rotasi.

c) *Blur*, Gambar diberi efek blur secara acak untuk meningkatkan ketahanan model terhadap *noise* dan kualitas gambar yang rendah.

b. Pelatihan model YOLOv5 pada *Google Colab*



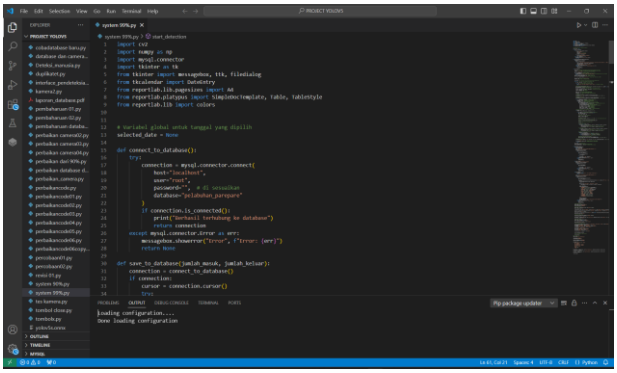
Gambar 4. 7 Pelatihan model YOLOv5 pada *Google Colab*

Gambar 4. 7 diatas menunjukkan tutorial pelatihan model YOLOv5 pada *Google Colab*. Tutorial ini terdiri dari 6 (enam) langkah:

1) Mengumpulkan data gambar dan memberi label pada data gambar tersebut.
2) Meng-*ekspor* data gambar yang telah diberi label ke dalam YOLOv5.
3) Melatih model YOLOv5 untuk mengenali objek-objek di dalam data gambar.
4) Mengevaluasi performa model YOLOv5.
5) Melakukan inferensi dengan data pengujian untuk melihat hasil model YOLOv5.
6) Mengkloning repo YOLOv5 untuk mengunduh dataset.

Gambar 4. 7 tersebut menunjukkan contoh cara mengkloning *repo* YOLOv5 dari *github*, dan menggambarkan proses pelatihan YOLOv5 dalam diagram berbentuk lingkaran, dengan setiap langkah pelatihan direpresentasikan dengan ikon yang berbeda dan deskripsi teks. Gambar ini dapat membantu pengguna memahami langkah-langkah pelatihan model YOLOv5 secara visual, dan memberikan panduan bagi mereka yang ingin membangun dan melatih model YOLOv5 sendiri.

c. Deteksi dan perhitungan



Gambar 4. 8 Deteksi dan perhitungan

Gambar 4. 8 ini menunjukkan kode *Python* yang digunakan untuk membangun model YOLOv5 dan mengakses *database* MySQL. Berikut adalah rinciannya:

1) YOLOv5:

Model YOLOv5 adalah model deteksi objek yang populer dan efisien. Kode *Python* ini mungkin berisi fungsi untuk melatih model YOLOv5, memuat model yang sudah dilatih, atau memproses data gambar untuk deteksi objek.

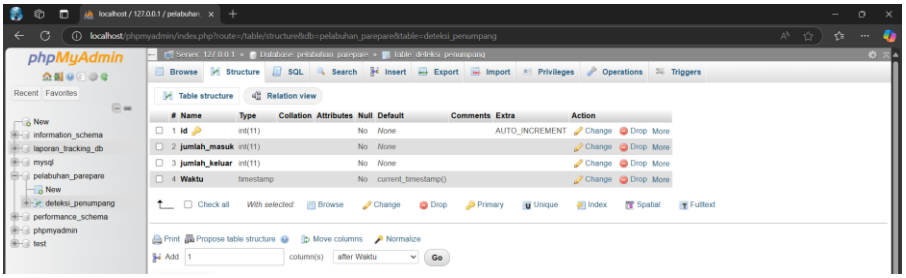
2) MySQL:

Database MySQL digunakan untuk menyimpan dan mengambil data terkait model YOLOv5. Kode ini mungkin berisi fungsi untuk menghubungkan ke *database*, menyimpan data pelatihan, atau mengambil hasil deteksi objek dari *database*.

3) Kode *Python*:

Kode *Python* ini ditulis dalam bahasa pemrograman *Python*, yang merupakan bahasa yang populer untuk pengembangan *deep learning* dan pemrosesan data.

d. Integrasi dengan database



Gambar 4. 9 Integrasi dengan *database*

Gambar 4. 9 diatas merupakan *database* yang memiliki struktur tabel untuk melacak data orang keluar dan masuk. Berikut penjelasan Kolom *Database* di atas:

1) id

id Ini adalah kolom kunci utama *(primary key)* dan merupakan angka unik yang mengidentifikasi setiap *entri* data penumpang. Dengan *AUTO_INCREMENT*, setiap entri baru akan secara otomatis mendapatkan *id* yang unik.

3) jumlah_masuk

Kolom ini mencatat jumlah orang yang masuk.

4) jumlah_keluar

Kolom ini mencatat jumlah orang yang keluar.

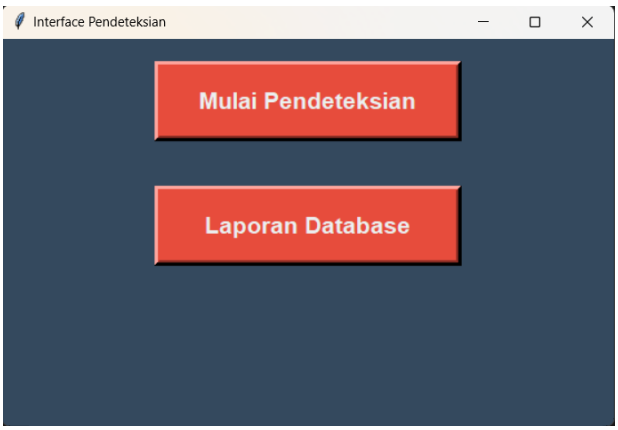
5) Waktu

Kolom ini mencatat waktu *(timestamp)* ketika data dicatat, dengan *default* menggunakan waktu saat ini *(current_timestamp())*.

2. Tampilan fitur pada sistem

a. Tampilan halaman *Interface* Pendeteksian

Pada tampilan ini merupakan tampilan awal aplikasi pada saat kita membuka aplikasi yang memiliki 2 (dua) pilihan menu yaitu menu mulai pendeteksian dan laporan *database* yang masing-masing memiliki fungsi tersendiri.



Gambar 4. 10 Halaman *Interface* Pendeteksian

Scrip Ini adalah bagian dari kode yang berfungsi untuk menampilkan antarmuka pengguna (GUI) menggunakan *Tkinter*, yang terdiri dari dua tombol utama: "Mulai Pendeteksian" dan "Laporan *Database*".

```
import tkinter as tk
from tkinter import ttk
from tkcalendar import DateEntry

# Buat jendela utama
root = tk.Tk()
root.title("Interface Pendeteksian")

root.geometry("500x300")
root.configure(bg="#34495e")

frame = tk.Frame(root, bg="#34495e")
frame.pack(expand=True, fill='both')

button_style = {
    "font": ("Helvetica", 16, "bold"),
    "fg": "#ecf0f1",
    "bg": "#e74c3c",
    "activebackground": "#c0392b",
    "activeforeground": "#ecf0f1",
    "bd": 5,
    "relief": "raised",
    "width": 20,
    "height": 2
}

btn_detection = tk.Button(frame, text="Mulai Pendeteksian",
command=start_detection, **button_style)
btn_detection.pack(pady=20)

btn_report = tk.Button(frame, text="Laporan Database",
command=view_database_report, **button_style)
btn_report.pack(pady=20)

root.mainloop()
```

Kode *Python* diatas berfungsi untuk membuat antarmuka pengguna sederhana menggunakan *Tkinter* untuk aplikasi yang kemungkinan terkait dengan pendeteksian dan pelaporan data. Penjelasan kode diatas dari langkah ke langkah sebagai berikut:

2) *Import Modul*

   a) *import tkinter as tk,* Mengimpor modul *Tkinter* untuk membuat antarmuka grafis.

   b) *from tkinter import* ttk, Mengimpor modul ttk dari *Tkinter* untuk menggunakan *widget* ttk yang lebih modern.

   c) from tkcalendar *import DateEntry*, Mengimpor modul *DateEntry* dari *tkcalendar* untuk membuat *widget entri* tanggal.

3) Buat Jendela Utama

   a) *root* = tk.Tk(), Membuat objek Tk yang merupakan jendela utama aplikasi.

   b) *root.title*("*Interface* Pendeteksian"), Mengatur judul jendela menjadi "*Interface* Pendeteksian".

   c) *root.geometry*("500x300"), Mengatur ukuran jendela menjadi 500 piksel lebar dan 300 piksel tinggi.

   d) *root.configure*(bg="#34495e"), Mengatur warna latar belakang jendela menjadi "#34495e" (warna abu-abu gelap).

4) Buat *Frame*

   a) *frame* = tk.*Frame*(root, bg="#34495e"), Membuat objek *Frame* sebagai kontainer untuk *widget* lainnya.

   b) *frame.pack*(expand=True, fill='both'), Menambahkan *frame* ke jendela dan membuatnya mengisi seluruh area jendela.

6) Definisikan Gaya Tombol

a) *button_style* = {...}, Membuat *dictionary button_style* untuk menyimpan pengaturan gaya untuk tombol.

b) *font*: Menetapkan *font* tombol menjadi "Helvetica", ukuran 16, dan tebal.

c) Fg, Menetapkan warna teks menjadi putih ("#ecf0f1").

d) Bg, Menetapkan warna latar belakang tombol menjadi merah ("#e74c3c").

e) *Activebackground*, Menetapkan warna latar belakang saat tombol ditekan ("#c0392b").

f) *Activeforeground*, Menetapkan warna teks saat tombol ditekan ("#ecf0f1").

g) Bd, Menetapkan lebar batas tombol menjadi 5.

h) *Relief*, Menetapkan *relief* tombol menjadi "*raised*" (terangkat).

b. Tampilan halaman mulai pendeteksian

Pada halaman ini merupakan halaman pendeteksian dan perhitungan manusia secara *realtime* yang menampilkan 2 (dua) buah garis *in* atau masuk yang berwarna hijau dan garis *out* atau keluar yang berwarna merah. Disudut kiri juga menampilkan jumlah perhitungan berapa yang masuk dan keluar.

Gambar 4. 11 halaman mulai pendeteksian

Bagian ini mengatur proses pendeteksian objek menggunakan YOLOv5, menghitung objek yang melewati garis masuk dan keluar, serta menyimpannya ke dalam *database*, berikut adalah scripnya:

```
import cv2
import numpy as np
from collections import deque
def start_detection():
    count_in = 0  # Variabel untuk menghitung jumlah objek yang melewati garis masuk
    count_out = 0  # Variabel untuk menghitung jumlah objek yang melewati garis keluar
    classes = ["manusia-lpAE"]  # Daftar kelas objek yang terdeteksi
    cap = cv2.VideoCapture(0)  # Mengakses kamera
    if not cap.isOpened():  # Memeriksa apakah kamera berhasil diakses
        messagebox.showerror("Error", "Kamera tidak dapat diakses.")  # Menampilkan pesan error jika tidak
        return
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)  # Mengatur lebar frame
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)  # Mengatur tinggi frame
    net = cv2.dnn.readNetFromONNX("yolov5s.onnx")  # Memuat model YOLOv5 dari file ONNX
    line_in_y = 190  # Posisi Y untuk garis masuk
    line_out_y = 320  # Posisi Y untuk garis keluar
    line_crossed = {}  # Dictionary untuk melacak objek yang telah melewati garis
```

```
    while True:  # Loop utama untuk menangkap dan memproses
frame
        ret, img = cap.read()  # Mengambil frame dari kamera
        if not ret:  # Memeriksa apakah frame berhasil diambil
            print("Warning: Tidak dapat mengambil frame dari
video.")  # Peringatan jika gagal
            break
        # Menyiapkan blob untuk input ke model
        blob = cv2.dnn.blobFromImage(img, scalefactor=1/255,
size=(640, 640), mean=[0, 0, 0], swapRB=True, crop=False)
        net.setInput(blob)  # Mengatur input untuk model
        detections = net.forward()[0]  # Melakukan inferensi dan
mendapatkan deteksi
        # Proses deteksi
        for detection in detections:
            confidence = detection[4]
            if confidence > 0.5:  # Ambang batas kepercayaan
                class_id = np.argmax(detection[5:])  # ID kelas
objek
                if class_id == 0:  # Jika objek terdeteksi
adalah manusia
                    x_center = int(detection[0] * img.shape[1])
                    y_center = int(detection[1] * img.shape[0])
                    width = int(detection[2] * img.shape[1])
                    height = int(detection[3] * img.shape[0])
                    # Mendefinisikan bounding box
                    x1 = int(x_center - width / 2)
                    y1 = int(y_center - height / 2)
                    x2 = int(x_center + width / 2)
                    y2 = int(y_center + height / 2)
                    # Menambahkan ID unik untuk setiap objek
                    obj_id = str(detection[0]) +
str(detection[1])
                    if obj_id not in line_crossed:
                        line_crossed[obj_id] = {"in": False,
"out": False}  # Menandai objek ini belum melewati garis
                    # Deteksi objek yang melewati garis masuk
                    if y2 > line_in_y and not
line_crossed[obj_id]["in"]:
                        count_in += 1
                        line_crossed[obj_id]["in"] = True  #
Menandai objek sudah melewati garis masuk
                    # Deteksi objek yang melewati garis keluar
                    if y1 < line_out_y and not
line_crossed[obj_id]["out"]:
                        count_out += 1
                        line_crossed[obj_id]["out"] = True  #
Menandai objek sudah melewati garis keluar
                    # Gambar bounding box pada objek yang
terdeteksi
                    cv2.rectangle(img, (x1, y1), (x2, y2), (0,
255, 0), 2)
```

```
        # Menggambar garis masuk dan keluar
        cv2.line(img, (0, line_in_y), (img.shape[1], line_in_y),
(0, 0, 255), 2)  # Garis masuk (red)
        cv2.line(img, (0, line_out_y), (img.shape[1],
line_out_y), (0, 255, 255), 2)  # Garis keluar (yellow)
        # Menampilkan jumlah objek yang melewati garis
cv2.putText(img, f'Jumlah Masuk: {count_in}', (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
cv2.putText(img, f'Jumlah Keluar: {count_out}', (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
        # Menampilkan frame hasil deteksi
        cv2.imshow("Pendeteksian Penumpang", img)
        # Menunggu input dari keyboard untuk keluar
        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):  # Jika tekan 'q', keluar dari loop
            break
    cap.release()  # Melepaskan kamera
    cv2.destroyAllWindows()  # Menutup semua jendela OpenCV
    save_to_database(count_in, count_out)  # Menyimpan data ke
database
```

Kode *Python* ini adalah tentang bagaimana alur pendeteksian dan perhitungan. Berikut penjelasannya:

1) Pengaturan dan Pemuatan Model YOLOv5

Menggunakan cv2.dnn.readNetFromONNX untuk memuat model YOLOv5 yang telah disimpan dalam format ONNX.

2) Deteksi dan Perhitungan Objek

Pada setiap *frame*, objek manusia yang terdeteksi akan dihitung apakah melewati garis masuk *(line_in_y)* atau keluar *(line_out_y)*.

Setiap objek yang melewati garis masuk atau keluar akan dihitung dan statusnya disimpan dalam *dictionary line_crossed*.

4) Menampilkan Hasil Deteksi

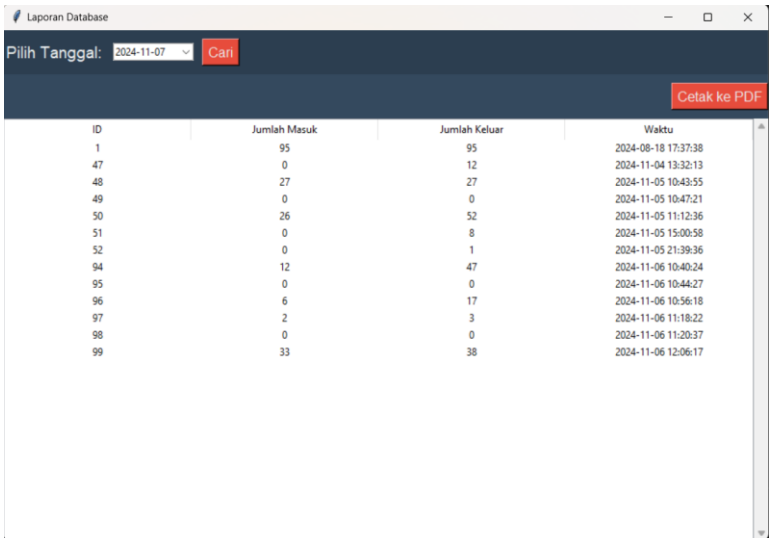
Bounding box objek akan digambar di atas gambar, dan jumlah objek yang masuk serta keluar akan ditampilkan.

5) Penyimpanan Data

Setelah selesai, hasil perhitungan akan disimpan dalam *database* dengan memanggil *save_to_database(count_in, count_out)*.

c. Tampilan halaman Laporan Database

Pada halaman ini merupakan tampilan halaman hasil perhitungan penumpang kapal, tampilan dibawa juga menampilkan menu untuk mencari data sesuai tanggal yang ingin di cari dan menyediakan menu untuk mencetak data ke format pdf.



Gambar 4. 12 halaman laporan *Database*

Kode berikut ini menghubungkan aplikasi dengan MySQL dan menyimpan hasil deteksi objek ke dalam tabel deteksi_penumpang:

```
import mysql.connector
def connect_to_database():
    try:
        connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",  # di sesuaikan
            database="pelabuhan_parepare"
        )
        if connection.is_connected():
            print("Berhasil terhubung ke database")
            return connection
    except mysql.connector.Error as err:
        messagebox.showerror("Error", f"Error: {err}")
        return None
def save_to_database(jumlah_masuk, jumlah_keluar):
    connection = connect_to_database()
    if connection:
        cursor = connection.cursor()
        try:
            query = "INSERT INTO deteksi_manusia (jumlah_masuk,
jumlah_keluar, Waktu) VALUES (%s, %s, NOW())"
            cursor.execute(query, (jumlah_masuk, jumlah_keluar))
            connection.commit()
            print("Data berhasil disimpan ke database")
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Gagal menyimpan
data: {err}")
        finally:
            cursor.close()
            connection.close()
```

Kode *Python* ini adalah contoh sederhana tentang bagaimana cara menyimpan data ke *database* MySQL. Berikut penjelasannya:

1) *Import* Modul, Kode ini pertama-tama mengimpor modul mysql.connector yang digunakan untuk berinteraksi dengan *database* MySQL.

2) Fungsi *connect_to_database*

a) Fungsi ini mencoba membuat koneksi ke *database* MySQL dengan informasi *host (localhost)*, *username (root)*, *password* (diganti dengan *password* Anda), dan nama *database* (pelabuhan_parepare).

b) Jika koneksi berhasil, fungsi ini mencetak pesan keberhasilan dan mengembalikan objek koneksi.

c) Jika koneksi gagal, fungsi ini menampilkan pesan *error* menggunakan *messagebox.showerror.*

3) Fungsi *save_to_database*

a) Fungsi ini menerima dua argumen: jumlah_masuk dan jumlah_keluar.

b) Fungsi ini memanggil fungsi *connect_to_database* untuk mendapatkan koneksi ke *database.*

c) Akhirnya, fungsi ini menutup kursor dan koneksi ke *database*.

Secara keseluruhan, kode ini berfungsi sebagai contoh untuk menunjukkan cara terhubung ke *database* MySQL dengan *Python*, Menunjukkan cara menyimpan data baru ke tabel *database*, Menangani *error* yang mungkin terjadi selama proses koneksi dan penyimpanan data.

d. Tampilan hasil pencarian berdasarkan tanggal

Halaman ini merupakan tampilan ketika *user* ingin mencari data sesuai tanggal yang dibutuhkan, maka sistem akan menampilkan tampilan seperti pada gambar di bawa yang hanya menampilkan data yang sesuai dengan tanggal yang di cari.

Gambar 4. 13 tampilan hasil berdasarkan tanggal

Berikut ini adalah *scrip* untuk mencari data dalam *database* berdasarkan tanggal yang dipilih dari *DateEntry*.

```
def fetch_data_by_date(date):
    global selected_date
    selected_date = date
    query = "SELECT * FROM deteksi_manusia WHERE DATE(Waktu) =
%s"
    connection = connect_to_database()
    if connection:
        cursor = connection.cursor()
        try:
            cursor.execute(query, (date,))
            rows = cursor.fetchall()
            for row in tree.get_children():
                tree.delete(row)
            for row in rows:
                tree.insert("", "end", values=row)
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Error: {err}")
        finally:
            cursor.close()
            connection.close()
```

Kode *Python* ini bertanggung jawab untuk mengambil data dari *database* MySQL berdasarkan tanggal tertentu dan menampilkannya di sebuah pohon *widget*. Berikut penjelasannya:

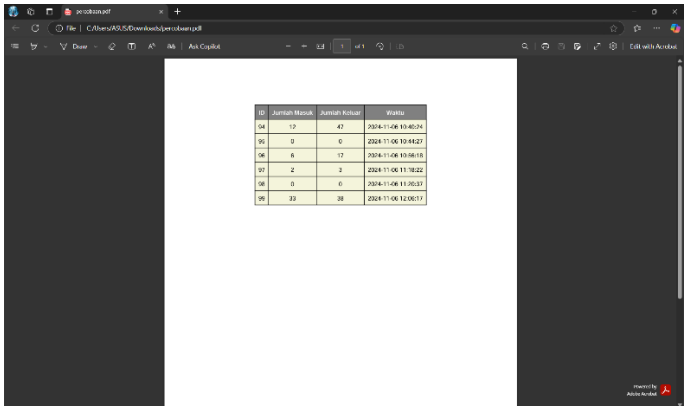
1) Fungsi *fetch_data_by_date(date)* menerima parameter *date* yang merupakan tanggal yang ingin diambil datanya.
2) *global selected_date* mengakses variabel global *selected_date* yang digunakan untuk menyimpan tanggal yang dipilih.
3) *query* = "SELECT * FROM deteksi_manusia *WHERE DATE*(Waktu) = %s" adalah query SQL yang digunakan untuk mengambil semua data dari tabel deteksi_manusia di mana tanggal waktu sama dengan *date*.
4) *connection* = *connect_to_database*() menghubungkan ke *database* MySQL.

5) *cursor = connection.cursor*() membuat *cursor* untuk menjalankan *query* SQL.
6) *cursor.execute(query, (date,))* menjalankan *query* SQL dengan tanggal yang diinputkan sebagai parameter.
7) *rows = cursor.fetchall()* mengambil semua data yang didapat dari *query* SQL.
8) *tree.get_children*() mengambil semua node anak pada pohon *widget*.
9) *tree.delete(row)* menghapus semua *node* anak yang ada di pohon *widget*.
10) *tree.insert*("", "*end*", *values=row*) menambahkan data yang didapat dari *database* ke dalam pohon *widget.*
11) Jika terjadi kesalahan *(error)*, *messagebox.showerror("Error", f"Error*: {err}") menampilkan pesan *error*.
12) Akhirnya, *cursor.close()* menutup cursor dan *connection.close()* menutup koneksi ke *database.*

Penjelasan di atas menjelaskan fungsi *fetch_data_by_date* mengambil data dari *database* MySQL berdasarkan tanggal yang diinputkan dan menampilkannya di pohon *widget*.

e. Tampilan hasil cetak ke pdf

Tampilan ini merupakan tampilan halaman hasil data laporan perhitungan manusia kapal yang telak di cetak ke format pdf, yang menampilkan ukuran kertas A4 berisi tabel perhitungan manusia.



Gambar 4. 14 tampilan hasil cetak ke pdf

Bagian ini memungkinkan pengguna untuk mencetak laporan yang berisi data berdasarkan tanggal yang dipilih, dalam format PDF menggunakan *ReportLab*, berikut adalah scripnya:

```
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Table,
TableStyle
from reportlab.lib import colors
def print_to_pdf():
    if selected_date is None:
        messagebox.showerror("Error", "Pilih tanggal terlebih
dahulu.")
        return
    query = "SELECT * FROM deteksi_manusia WHERE DATE(Waktu) =
%s"
    connection = connect_to_database()
    if connection:
        cursor = connection.cursor()
        try:
            cursor.execute(query, (selected_date,))
            rows = cursor.fetchall()
            file_path =
filedialog.asksaveasfilename(defaultextension=".pdf",
filetypes=[("PDF files", "*.pdf")])
            if file_path:
                doc = SimpleDocTemplate(file_path, pagesize=A4)
                elements = []
                data = [['ID', 'Jumlah Masuk', 'Jumlah Keluar',
'Waktu']]
                for row in rows:
                    data.append([str(row[0]), str(row[1]),
str(row[2]), str(row[3])])
                table = Table(data)
```

```
                table.setStyle(TableStyle([
                    ('BACKGROUND', (0, 0), (-1, 0),
colors.grey),
                    ('TEXTCOLOR', (0, 0), (-1, 0),
colors.whitesmoke),
                    ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
                    ('BACKGROUND', (0, 1), (-1, -1),
colors.beige),
                    ('GRID', (0,0), (-1,-1), 1, colors.black),
                    ('FONT', (0,0), (-1,0), 'Helvetica-Bold'),
                    ('FONT', (0,1), (-1,-1), 'Helvetica'),
                    ('FONTSIZE', (0,0), (-1,-1), 10),
                ]))
                elements.append(table)
                doc.build(elements)
                messagebox.showinfo("Sukses", "Laporan berhasil
dicetak sebagai PDF.")
            except mysql.connector.Error as err:
                messagebox.showerror("Error", f"Error: {err}")
            finally:
                cursor.close()
                connection.close()
```

Kode *Python* ini membuat laporan PDF dari data tabel deteksi_manusia di *database* MySQL.

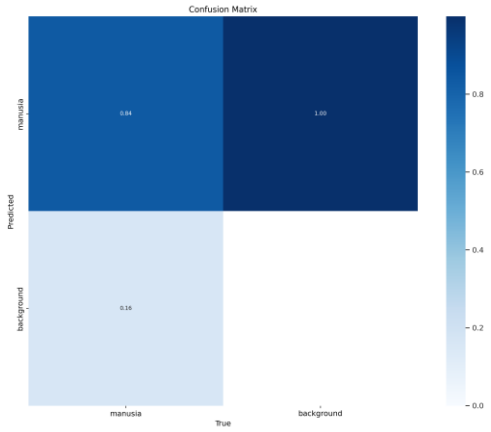
1) Memilih tanggal, *User* memilih tanggal laporan yang diinginkan.
2) Mengambil data, kode mengambil data dari tabel deteksi_manusia berdasarkan tanggal yang dipilih.
3) Membuat tabel PDF, Data diubah menjadi tabel PDF dengan kolom: ID, Jumlah Masuk, Jumlah Keluar, dan Waktu.
4) Menyimpan PDF, *User* memilih nama dan lokasi file PDF.
5) Menampilkan pesan, kode menampilkan pesan sukses atau *error* jika proses berhasil atau gagal.

C. Pengujian sistem

Pada tahap pengujian sistem, penulis melaksanakan proses pengujian secara langsung terhadap setiap komponen program yang telah dikembangkan. Selain dari *Training* model yang digunakan dan analisis performa, sistem pengujian ini juga menggunakan dua pendekatan utama, yaitu metode *blackbox* dan metode *whitebox*. Melalui metode *blackbox*, penulis menguji fungsi-fungsi aplikasi berdasarkan *input* dan *output* tanpa melihat kode *internal*, sehingga fokus pada bagaimana aplikasi berinteraksi dengan pengguna. Sedangkan, dengan metode *whitebox*, penulis memeriksa struktur *internal* dari kode program untuk memastikan setiap alur logika berjalan sesuai rencana. Pengujian ini bertujuan untuk mengevaluasi sejauh mana aplikasi mampu mengatasi kondisi-kondisi yang tidak diinginkan atau tidak terduga, sekaligus memastikan keandalan serta ketangguhan aplikasi saat digunakan dalam berbagai skenario.

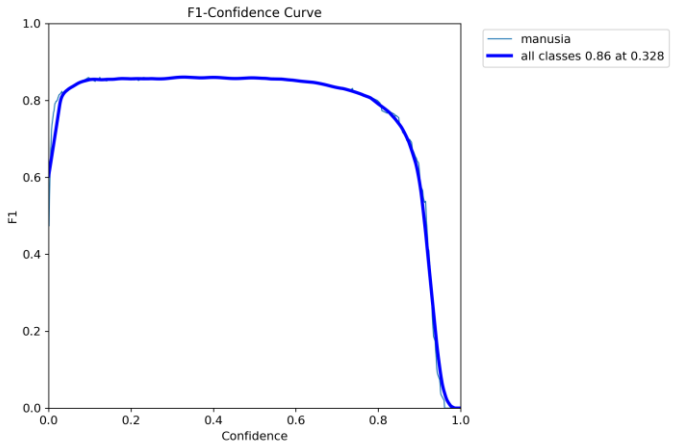
1. Analisis Model Training YOLOv5

Model YOLOv5 yang disesuaikan dikembangkan melalui proses *eksperimental* yang dilakukan secara berulang menggunakan *dataset* yang sudah diproses. Proses ini bertujuan untuk menemukan konfigurasi terbaik yang mampu memberikan kinerja optimal. Hasil dari pelatihan model pada data validasi mencakup *confusion matrix*, F1-*Confidence Curve*, serta *Precision-Recall Curve* sebagai berikut:



Gambar 4. 15 *Confusion matrix*

Gambar 4. 15 tersebut menampilkan *matriks* konfusi untuk model klasifikasi yang mengidentifikasi apakah sebuah gambar menampilkan manusia atau latar belakang. *Matriks* menunjukkan bahwa, Model mengklasifikasikan 84% gambar manusia dengan benar sebagai manusia. Model mengklasifikasikan 100% gambar latar belakang dengan benar sebagai latar belakang. 16% dari gambar manusia secara keliru diklasifikasikan sebagai latar belakang. 0% dari gambar latar belakang secara keliru diklasifikasikan sebagai manusia. Kesimpulannya, model ini sangat baik dalam mengklasifikasikan gambar latar belakang, tetapi masih ada ruang untuk peningkatan dalam mengidentifikasi gambar manusia secara akurat.

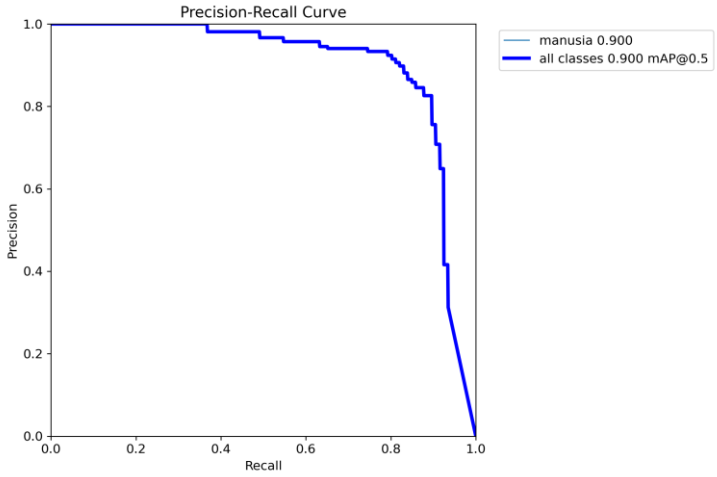


Gambar 4. 16 F1-*Confidence curve*

Gambar 4. 16 tersebut menunjukkan *kurva* kepercayaan F1 untuk model klasifikasi. *Kurva* kepercayaan F1 mengukur kinerja model klasifikasi pada berbagai ambang batas kepercayaan. Sumbu *horizontal* menunjukkan ambang batas kepercayaan, yang berkisar dari 0 hingga 1. Ambang batas kepercayaan adalah nilai yang digunakan untuk menentukan apakah suatu prediksi *positif* atau *negatif*. Misalnya, jika ambang batas kepercayaan adalah 0,5, maka model akan memprediksi kelas *positif* jika kepercayaan prediksinya lebih besar dari 0,5, dan akan memprediksi kelas *negatif* jika kepercayaan prediksinya kurang dari 0,5. Sumbu *vertikal* menunjukkan skor F1, yang merupakan ukuran kinerja yang menggabungkan presisi dan *re* panggilan. Skor F1 berkisar dari 0 hingga 1, dengan nilai 1 menunjukkan kinerja sempurna dan nilai 0 menunjukkan kinerja yang buruk.

Kurva biru menunjukkan skor F1 untuk kelas "manusia" pada berbagai ambang batas kepercayaan. *Kurva* biru tua menunjukkan skor F1 untuk semua kelas pada berbagai ambang batas kepercayaan. Skor F1 untuk semua kelas dihitung sebagai rata-rata tertimbang dari skor F1 untuk setiap kelas. Teks "*all classes* 0.86

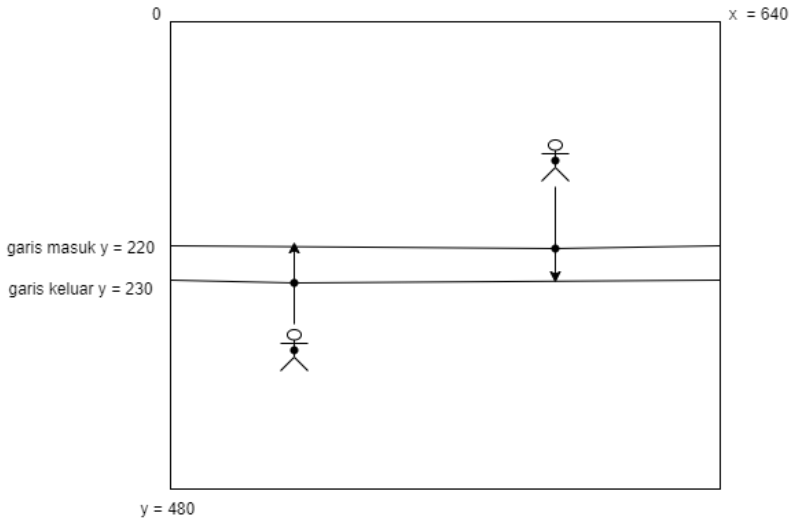
at 0.328" menunjukkan bahwa skor F1 *maksimum* untuk semua kelas adalah 0.86, dan nilai tersebut dicapai pada ambang batas kepercayaan 0.328. Secara keseluruhan, grafik menunjukkan bahwa model tersebut memiliki kinerja yang baik dalam mengklasifikasikan kelas "manusia", dan juga memiliki kinerja yang baik secara keseluruhan untuk semua kelas. Model mencapai kinerja terbaiknya pada ambang batas kepercayaan sekitar 0.328.



Gambar 4. 17 *Precision-recall curve*

Gambar 4. 17 ini menunjukkan *kurva* presisi-penarikan balik untuk model yang dilatih untuk mendeteksi objek "manusia". *Kurva* ini menunjukkan bahwa model memiliki performa yang sangat baik dalam mendeteksi manusia, dengan nilai mAP@0.5 sebesar 0.900. Artinya, model dapat mengidentifikasi manusia dengan benar 90% dari waktu, dengan tingkat presisi yang tinggi. *Kurva* presisi-penarikan balik adalah alat yang berguna untuk menilai performa model deteksi objek. *Kurva* ini menunjukkan hubungan antara presisi atau akurasi model dan penarikan balik kemampuan model untuk mengidentifikasi semua objek yang relevan.

2. Analisis Tentang Masuk dan Keluar menggunakan Garis



Gambar 4. 18 Deteksi orang

Berikut sedikit penjelasan mengenai Cara Kerja Pendeteksian dan Perhitungan objek (manusia):

Berikut penjelasan Letak dan Peran Koordinat:

a. Koordinat Bounding Box

1) Setiap manusia yang terdeteksi memiliki koordinat *bounding box* yang mencakup:

- Xmin, Ymin: Titik sudut kiri atas.

- Xmax, Ymax: Titik sudut kanan bawah.

2) Koordinat ini menunjukkan posisi objek dalam *frame* video yang memiliki *frame* 640 x 480.

b. Titik Tengah *Bounding Box*

1) Titik tengah *bounding box* dihitung berdasarkan rata-rata koordinat kiri atas dan kanan bawah, menghasilkan nilai Xmid dan Ymid.

2) Xmid menunjukkan posisi *horizontal* objek, sedangkan Ymid menunjukkan posisi vertikal yang digunakan untuk memeriksa apakah objek melewati garis penghitung.

c. Koordinat Garis Penghitung

1) Garis hijau memiliki nilai Y tetap, Y = 220.
2) Garis merah juga memiliki nilai Y tetap, Y = 230.
3) Karena garis ini bersifat *horizontal*, koordinat X tidak digunakan.

d. Interaksi Titik Tengah dengan Garis

1) Ketika Ymid, yaitu koordinat vertikal titik tengah objek, melintasi salah satu garis dan sama dengan garis yang satunya maka dikatakan terhitung.

3. Analisis Performa Sistem Deteksi dan perhitungan penumpang

Secara umum, performa sistem deteksi dan perhitungan objek manusia secara *realtime* sangat dipengaruhi oleh model yang digunakan serta beban komputasi sistem. Pada sistem ini telah dilakukan pengujian beberapa kali pada tanggal 05 September 2024 sampai tanggal 06 September 2024. Setelah melakukan pengujian sistem akan dilakukan perhitungan tingkat keberhasilan sistem, perhitungan tingkat keberhasilan sistem Berdasarkan hasil pengujian model di berbagai kondisi pencahayaan, hasil yang diperoleh menunjukkan hal-hal sebagai berikut:

Tabel 4. 1 Perhitungan tingkat keberhasilan

Tanggal/waktu	Kategori	Jumlah manual	Jumlah sistem	Tingkat keberhasilan (%)
05-11-2024 jam 10	Jumlah masuk	40	27	$\frac{27}{40}$ x 100% =67,5%
05-11-2024 jam 10	Jumlah keluar	35	27	$\frac{27}{35}$ x 100% = 77,14%
05-11-2024 jam 11	Jumlah masuk	29	26	$\frac{26}{29}$ x 100% = 89,66%
05-11-2024 jam 11	Jumlah keluar	61	52	$\frac{52}{61}$ x 100% = 82,25%
05-11-2024 jam 13	Jumlah masuk	2	0	$\frac{0}{2}$ x 100% = 0, 0%
05-11-2024 jam 13	Jumlah keluar	13	8	$\frac{8}{13}$ x 100% = 61,54%
06-11-2024 jam 11	Jumlah masuk	46	33	$\frac{33}{46}$ x 100% = 71,74%
06-11-2024 jam 11	Jumlah keluar	60	38	$\frac{38}{60}$ x 100% = 63,33%

$$\text{Tingkat keberhasilan} = \frac{Total\ tingkat\ keberhasilan\ semua\ percobaan}{jumlah\ percobaan}$$

Total tingkat keberhasilan semua percobaan:

67,5% + 77,14% + 89,66% + 82,25% + 0,0% + 61,54% + 71,74% + 63,33% = 516,16%

Jumlah percobaan:

Terdapat 8 (delapan) total percobaan.

$$\text{Tingkat keberhasilan} = \frac{516{,}16\%}{8} = \mathbf{64{,}52\%}$$

Pada pengujian sistem, tingkat keberhasilan sebesar 64,52% menunjukkan bahwa YOLOv5 belum sepenuhnya akurat dalam mendeteksi penumpang dengan tingkat keyakinan tinggi. Nilai ini berarti model cukup yakin terhadap deteksi manusia, tetapi masih rentan terhadap kesalahan seperti *false positives* (objek yang

bukan manusia terdeteksi sebagai manusia) atau *false negatives* (manusia tidak terdeteksi). Hal ini dapat terjadi akibat faktor lingkungan seperti pencahayaan yang tidak stabil atau sudut pandang kamera yang kurang ideal. Dengan meningkatkan ambang batas *confidence* menjadi sekitar 70-80% dan menyesuaikan lingkungan deteksi, akurasi dapat ditingkatkan sehingga sistem lebih andal dalam pendeteksian dan perhitungan. Berikut adalah kondisi *false negatives* pada saat pengujian sistem.

Gambar 4. 19 Kasus *False Negative*

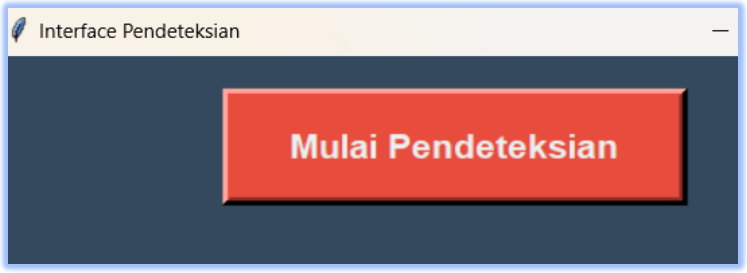
Gambar 4. 19 diatas menunjukkan kasus *false negative* di mana seorang penumpang yang melintasi garis deteksi tidak terdeteksi oleh sistem. Meskipun objek berada di antara garis *"In"* dan *"Out,"* sistem tidak mengenalinya sebagai penumpang. Hal ini kemungkinan disebabkan oleh pencahayaan yang rendah, jarak objek yang kurang optimal, atau ketidak jelasan objek dalam *frame,* sehingga model kesulitan melakukan deteksi.

4. Pengujian *Blackbox*

Pengujian Blackbox *(blackbox testing)* merupakan salah satu metode pengujian yang difokuskan pada aspek fungsionalitas sistem. Metode ini dilakukan dengan membandingkan hasil keluaran dari sistem dengan *output* yang diharapkan. Jika hasil keluaran sistem sesuai dengan *ekspektasi*, maka dapat disimpulkan bahwa aplikasi berfungsi sesuai dengan spesifikasi yang telah ditetapkan sebelumnya.

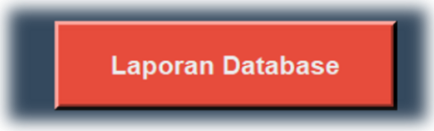
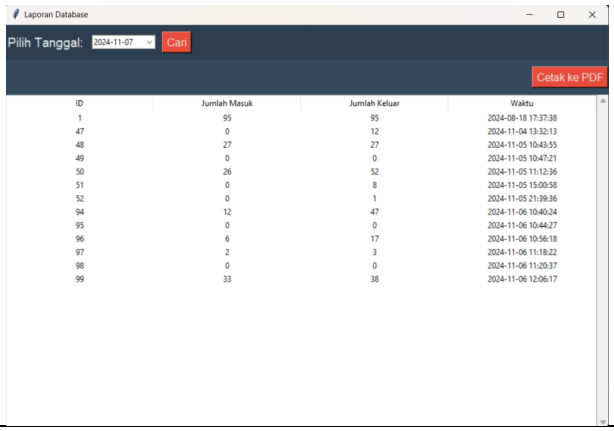
a. Blackbox Testing mulai pendeteksian

Tabel 4. 2 *Blackbox Testing* mulai pendeteksian

Tes Faktor	Hasil	Keterangan
Pilih menu Mulai Pendeteksian	✓	Berhasil, tampil halaman Pendeteksian
Screenshot		
 		

b. Blackbox Testing Laporan Database

Tabel 4. 3 *Blackbox Testing* Laporan Database

Tes Faktor	Hasil	Keterangan
Pilih menu Laporan *Database*	✓	Berhasil, tampil halaman Laporan *Database*
Screenshot		
 		

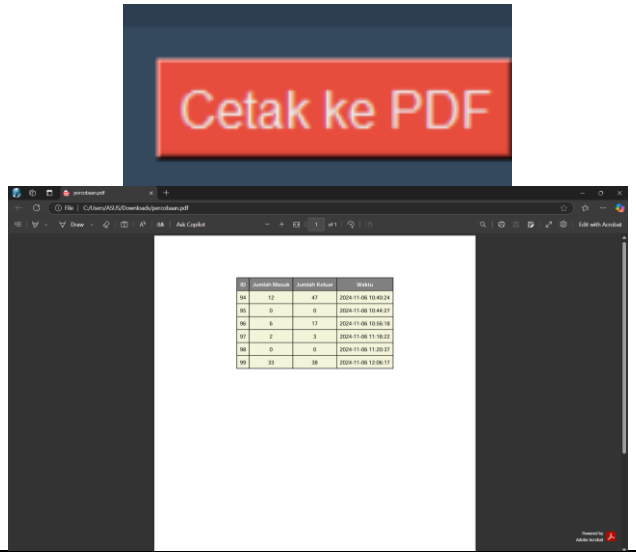
c. Blackbox Testing Pencarian tanggal

Tabel 4. 4 *Blackbox Testing* Pencarian Tanggal

Tes Faktor	Hasil	Keterangan
Pilih Tanggal dan cari	✓	Berhasil, tampil data berdasarkan tanggal yang di cari
Screenshot		
 		

d. *Blackbox Testing* Cetak ke pdf

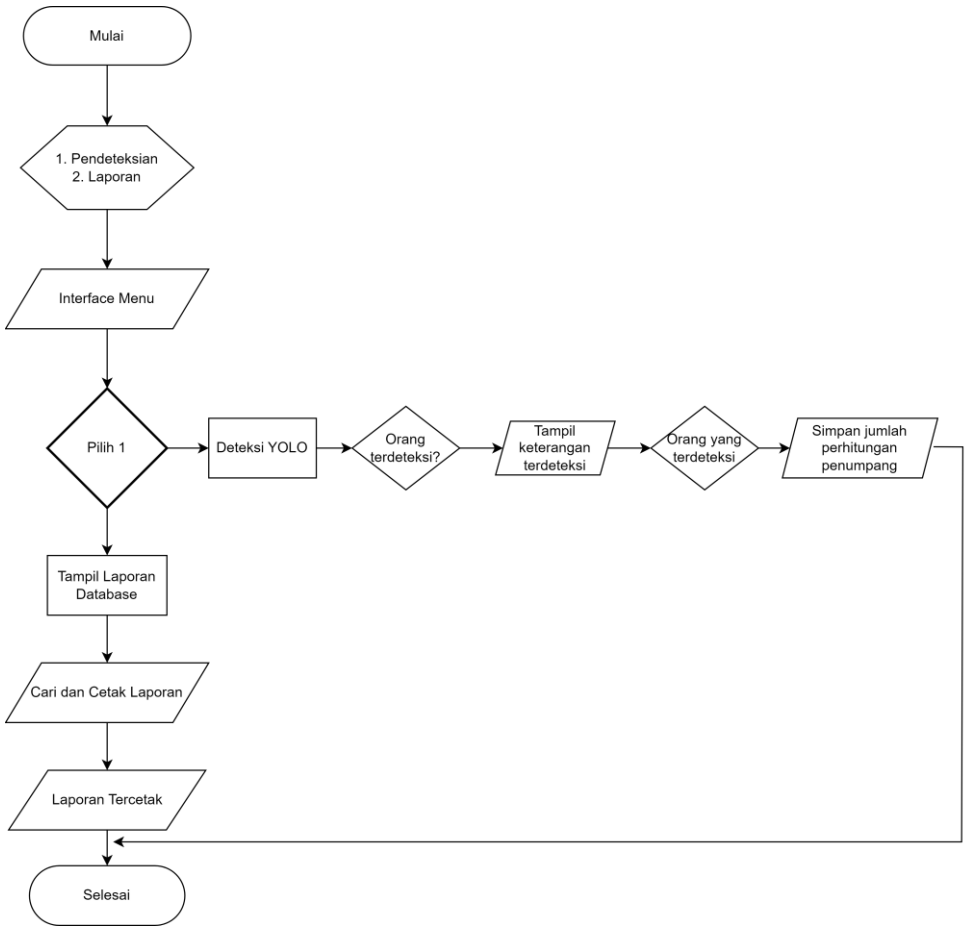
Tabel 4. 5 *Blackbox Testing* Cetak ke pdf

Tes Faktor	Hasil	Keterangan
Pilih menu cetak ke pdf	✓	Berhasil, tampil hasil data perhitungan dalam bentuk file pdf
Screenshot		
		

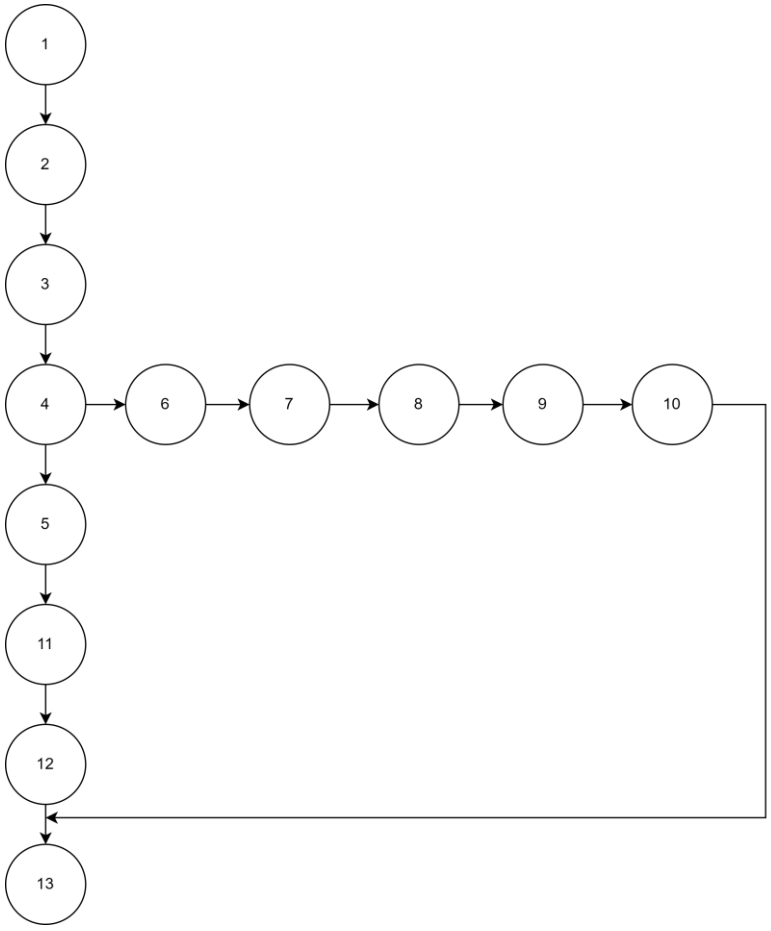
5. Pengujian *Whitebox*

Pengujian ini menampilkan *flowchart* dan *flowgraph* yang menggambarkan alur kerja sistem yang telah dirancang secara rinci. Pada bagian ini, disajikan hasil dari pengujian menggunakan metode *whitebox testing*, yang bertujuan untuk memeriksa logika *internal* serta alur program agar sesuai dengan spesifikasi yang diinginkan.

b. Pengujian *Flowchart* dan *Flowgraph*



Gambar 4. 20 *Flowchart* Pengujian *whitebox*



Gambar 4. 21 *Flowgraph* pengujian *Whitebox*

1) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node.*

Pada rumus V(G) = E – N + 2

E (*edge*) = 13

N (*node*) = 13

P (Predikat *node*) = 1

Penyelesaian :

Jadi : V(G) = E – N + 2

= 13 – 13 + 2

= 2

P (Predikat *node*) = 1 + 1

= 2

2) Dari perhitungan *cyclomatic complexity flowgraph* di atas memiliki region = 2.

3) *Independent path* pada *Flowgraph* tersebut yakni :

Path 1 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 13

Path 2 = 1 – 2 – 3 – 4 – 5 – 11 – 12 – 13

4) Grafik matriks

Tabel 4. 6 *Grafik matriks*

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**E - 1**
1		1												1 – 1 = 0
2			1											1 – 1 = 0
3				1										1 – 1 = 0
4					1	1								2 – 1 = 1
5											1			1 – 1 = 0
6							1							1 – 1 = 0
7								1						1 – 1 = 0
8									1					1 – 1 = 0
9										1				1 – 1 = 0
10													1	1 – 1 = 0
11												1		1 – 1 = 0
12													1	1 – 1 = 0
13														0
SUM (E + 1)														1 + 1 = 2

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil pengujian sistem yang memperoleh akurasi sebesar 64,52%, dapat disimpulkan bahwa sistem sudah menunjukkan performa yang cukup baik dalam mendeteksi dan menghitung orang yang keluar dan masuk, namun masih ada ruang untuk perbaikan. Pengujian melalui metode *blackbox testing* dan *whitebox testing* menunjukkan bahwa meskipun sistem dapat berfungsi sesuai dengan spesifikasi, terdapat kemungkinan adanya faktor *eksternal* atau *internal* yang mempengaruhi kinerjanya, seperti kualitas video input, kondisi pencahayaan, atau pengaturan parameter model. Hasil ini menandakan bahwa meskipun algoritma YOLOv5 sudah dapat melakukan deteksi objek, akurasi yang belum optimal menunjukkan bahwa perbaikan lebih lanjut diperlukan.

B. Saran

Untuk meningkatkan akurasi sistem, pelatihan ulang model YOLOv5 dengan dataset yang sesuai dengan kondisi lingkungan penerapan sangat disarankan. Selain itu, mempertimbangkan penggunaan model yang lebih canggih, seperti YOLOv8, dapat membantu meningkatkan hasil. Dataset juga sebaiknya mencakup lebih banyak variasi, seperti perbedaan pencahayaan, sudut kamera, dan jenis objek, sehingga sistem dapat berfungsi dengan baik di berbagai kondisi. Teknik tambahan seperti augmentasi gambar, pengurangan gangguan visual, dan

peningkatan kontras dapat digunakan untuk memperbaiki kualitas gambar yang akan diproses oleh model deteksi.

DAFTAR PUSTAKA

Astakona, (2024). Ai Machain Learning, Apa itu Deep Learning. https://astakona.id/id/blogz/ai-machine-learning/apa-itu-deep-learning/. Di

\akses pada 17 April.

Cahyaningtyas. (2023). Pengaruh Perkembangan Teknologi Pada Era Revolusi Industri 4.0 Terdapat Sumber Daya Manusia dab Ketenagakerjaan di Pasar Tenaga Kerja.

Cahyani, P. A. (2023). Sistem Perhitungan Kendaraan Menggunakan Algoritma YOLOv5 dan Deepsort. Laporan Proyek Akhir ,Universitas Lampung,Bandar Lampung.

Cristopher, Silvia, (2022). Deteksi Kematangan Buah Pepaya Menggunakan Metode Algoritma YOLO Berbasis Android. AMIK JTC Semarang.

Deng, L. & Yu, D., (2014). Deep Learning: Methods and Application, Foundations and Trends in Signal Processing.

Efendi, J. (2021). Black-Box Testing : Analisis Kualitas Aplikasi Source Code Bank Programming. Jurnal Teknologi Informasi dan Komunikasi*)*, *5*(1), 2021. https://doi.org/10.35870/jti

Fauzi Bagaskara, R. (t.t.). Evaluasi Sistem Antrian Pada Usaha Pencucian Mobil Di Prima Gold Carwash Yogyakarta.

Fu'adi, A., Prianggono, A., Juliartha, B., Putra, M., Hikmawan, B., Komunitas, A., Pacitan, N., Id, A. A., Id, A. A., Id, B. A., & Id, B. A. (2024). Pembangunan Sistem Monitoring Kehadiran Mahasiswa Menggunakan Yolo Pendeteksi Obyek dan Pengenal Wajah Opencv. Jurnal Ilmiah Teknologi Informasi Asia, 18(1).

Goodfellow, dkk. (2016). Deep Learning. MIT Press.

Gabriel Irfon Elrohi Soen, Marlina, Renni. 2022. Implementasi Cloud Computing dengan Google Colaboratory pada Aplikasi Pengolah Data Zoom Participants. Makassar: STMIK Kharisma Makassar.

Irma, (2020). Teknik Dalam Whitebox Testing dan Blackbox Testing.

Liunanda, C. N., Rostianingsih, S., & Purbowo, A. N. (t.t.). Implementasi Algoritma YOLO pada Aplikasi Pendeteksi Senjata Tajam di Android.

Mohamad Arifin, (2020). Aplikasi Informasi Monitoring Hasil Diluar Kawasan Hutan CDK Wilayah II di Kota Parepare Berbasis Web. Parepare: Universitas Muhammadiyah Parepare.

Natalia Angeline, (2022). Rancang Bangun Sistem Identifikasi Nematoda di Indonesia Menggunakan Faster R-CNN. Tangerang: Universitas Multimedia Nusantara Tangerang.

Rosaly Rizqi, & Prasetyo Andy, (2019). Pengertian Flowchart Beserta Fungsi dan Simbol-simbol Flowchart yang Paling Umum Digunakan.

Razak, M. F. (2022). Implementasi Metode Unified Modelling Language (UML) Pada Website Presensi Pegawai. Jurnal Informatika, 1(1), 39–45. https://doi.org/10.37150/jift.v1i1.2225

Sinta, (2020). Apa Itu Webcam, Fungsi, Jenis-Jenis Webcam.

Shapiro, L.G. dan Stockman, G.C., (2001). Computer Vision. 1st Edition, Prentice Hall.

Sugandi, A. N., & Hartono, B. (2022). *Prosiding The 13th Industrial Research Workshop and National* Seminar Bandung.

Wahana Komputer. (2010). *Panduan Belajar MySQL Database Server*. Jakarta: Media Kita.

Wibowo, et al., (2023). Implementasi Algoritma Deep Learning You Only Look Once (YOLOv5) Untuk Deteksi Buah Segar Dan Busuk. Universitras Subang.