

**RANCANG BANGUN SISTEM MONITORING AKTIVITAS
KEDATANGAN DAN KEPERGIAN PADA RUANGAN PUSAT
DATA DENGAN MENGGUNAKAN ALGORITMA YOLO V5 DAN
DEEPSORT BERBASIS *INTERNET OF THINGS* (IoT)**

SKRIPSI

BAGUS BIMA ADINATA

201401135



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**RANCANG BANGUN SISTEM MONITORING AKTIVITAS
KEDATANGAN DAN KEPERGIAN PADA RUANGAN PUSAT
DATA DENGAN MENGGUNAKAN ALGORITMA YOLO V5 DAN
DEEPSORT BERBASIS *INTERNET OF THINGS* (IoT)**

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh
ijazah Sarjana Ilmu Komputer**

BAGUS BIMA ADINATA

201401135



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA**

MEDAN

2024

PERSETUJUAN

| | | |
|-----------------------|---|--|
| Judul | : | RANCANG BANGUN SISTEM MONITORING AKTIVITAS KEDATANGAN DAN KEPERGIAN PADA RUANGAN PUSAT DATA MENGGUNAKAN ALGORITMA YOLO V5 DAN DEEPSORT BERBASIS INTERNET OF THINGS (IoT) |
| Kategori | : | SKRIPSI |
| Nama | : | BAGUS BIMA ADINATA |
| Nomor Induk Mahasiswa | : | 201401135 |
| Program Studi | : | SARJANA (S-1) ILMU KOMPUTER |
| Fakultas | : | ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVRSITAS SUMATERA UTARA |

Telah diuji dan dinyatakan lulus di Medan, 15 Oktober 2024

Dosen Pembimbing I

Prof. Dr. Syahril Efendi S.Si., M.I.T.

NIP. 196711101996021001

Dosen Pembimbing II

Fauzan Nurahmadi S.Kom., M.Cs.

NIP. 198512292018051001

Diketahui/Disetujui Oleh
Ketua Program Studi S-1 Ilmu Komputer

Dr. Amalia, S.T., M.T

NIP. 19781221 201404 2 001

UNIVERSITAS SUMATERA UTARA

PERNYATAAN**RANCANG BANGUN SISTEM MONITORING AKTIVITAS
KEDATANGAN DAN KEPERGIAN PADA RUANGAN PUSAT
DATADENGAN MENGGUNAKAN ALGORITMA YOLO V5 DAN
DEEPSORT BERBASIS *INTERNET OF THINGS* (IoT)****SKRIPSI**

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

Medan, 1 Oktober 2024



Bagus Bima Adinata

20140135

PENGHARGAAN

Bismillahirrahmanirrahim, puji syukur senantiasa dipanjatkan kepada Allah SWT. yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi ini. Shalawat beriring salam tidak lupa dihadiahkan ke baginda Rasulullah *Shalallaahu 'Alayhi Wasallam* yang telah membawa manusia keluar ke zaman yang terang benderang.

Penulis sadar dalam menyelesaikan penyusunan skripsi ini ada banyaknya dukungan, bimbingan, nasehat dan motivasi dari beberapa pihak terkait. Maka dari itu, penulis akan mengucapkan ucapan terima kasih terhadap:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fasilkom-TI Universitas Sumatera Utara.
3. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fasilkom-TI Universitas Sumatera Utara.
4. Bapak Prof. Dr. Syahril Efendi S.Si., M.I.T., selaku Dosen Pembimbing I yang selalu memberikan bimbingan, dukungan serta meluangkan waktu untuk penulis selama penyusunan skripsi ini.
5. Bapak Fauzan Nurahmadi S.kom., M.Cs., selaku Dosen Pembimbing II yang telah memberikan banyak masukan, serta arahan kepada penulis selama penyusunan skripsi ini.
6. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku dosen pembimbing akademik yang telah memberikan dukungan penuh, semangat, bimbingan dan bantuan kepada penulis selama masa perkuliahan.
7. Dosen Penguji I
8. Dosen Penguji II
9. Seluruh Bapak dan Ibu dosen program studi S-1 Ilmu Komputer Universitas Sumatera Utara yang telah memberikan banyak pembelajaran berharga bagi penulis dibangku perkuliahan serta seluruh staff pegawai yang memberikan bantuan selama masa perkuliahan.
10. Bapak Bambang Priyatno S.Sos. dan Ibu Rini Pujiastuti selaku orang tua yang tidak pernah hentinya memberikan motivasi, dukungan, doa, saran, kasih sayang penuh

kepada penulis dalam menuntaskan Pendidikan dan selalu sabar membimbing penulis hingga saat ini.

11. Abang Bagus Andhika serta Adik Rayi Safia Amanditha yang selalu memberikan dukungan, doa serta keceriaan dalam menjalankan aktivitas kuliah dan sehari-hari hingga akhirnya menyelesaikan tugas akhir.
12. Special thanks kepada ECM yang selalu memberi semangat, mendoakan, bantuan dan menemani penulis dalam banyak kegiatan hingga dapat menyelesaikan Tugas Akhir ini.
13. Kepada party an Klop Nothing To Lose yang selalu memberikan keceriaan, pembelajaran, adrenalin serta selalu memberi tanpa pamrih kepada penulis.
14. Kepada kaum MUHAJIRIN yang selalu menemani dan membantu penulis untuk melewati masa – masa suram selama perkuliahan.
15. Seluruh teman – teman kuliah stambuk 20 yang banyak berperan dalam setiap proses berkembangnya penulis di masa perkuliahan.

Serta semua pihak yang telah memberikan bantuan kepada penulis, meskipun tidak dapat disebutkan satu per satu, semoga diberkahi oleh Allah SWT. atas segala bantuan yang diberikan.

Medan, 1 Oktober 2024

Penulis,



Bagus Bima Adinata

**“RANCANG BANGUN SISTEM MONITORING AKTIVITAS
KEDATANGAN DAN KEPERGIAN PADA RUANGAN PUSAT
DATA DENGAN MENGGUNAKAN ALGORITMA YOLO V5 DAN
DEEPSORT BERBASIS *INTERNET OF THINGS* (IoT)”**

ABSTRAK

Dalam era digital saat ini, ruang data center memiliki peran yang sangat penting dan harus dilindungi secara maksimal, baik secara fisik maupun non-fisik. Hal ini disebabkan karena data center berfungsi sebagai tempat penyimpanan perangkat teknologi informasi (TI), mulai dari server hingga pusat komunikasi data perusahaan. Oleh karena itu, pengamanan yang ketat sangat diperlukan untuk menjaga keamanan aset penting perusahaan. Penelitian ini bertujuan untuk merancang sistem pemantauan aktivitas keluar-masuk di ruang data center dengan menggunakan teknologi *Internet of Things* (IoT) dan *Artificial Intelligence* (AI). Algoritma YOLO V5 digunakan untuk mendeteksi objek, sementara algoritma *DeepSORT* berfungsi untuk melacak pergerakan objek yang terdeteksi. Sistem ini diusulkan untuk menyelesaikan masalah keamanan dan meningkatkan efisiensi operasional di ruang data center, yang sering kali masih menggunakan metode manual atau sistem monitoring yang kurang efisien. Dengan menggunakan perangkat ESP32-CAM sebagai alat IoT untuk menangkap video secara *real-time*, YOLO V5 akan mendeteksi objek yang masuk dan keluar dari ruang data center, dan *DeepSORT* akan melacak pergerakannya. Sistem ini diharapkan mampu memberikan solusi yang lebih efektif dan efisien untuk melakukan pemantauan. Melalui pemanfaatan IoT dan AI. Data hasil pemantauan akan disimpan secara otomatis dalam database, video akan ditampilkan secara *streaming* melalui *web* antarmuka berbasis *Django* serta ditampilkan pada halaman utama dan hasil pemantauan juga akan ditampilkan pada halaman utama.

Kata Kunci: *YOLOv5, DeepSORT, Internet of Things, monitoring, data centre, Django, Kecerdasan Buatan, ESP32-CAM.*

**“DESIGN AND DEVELOPMENT OF A MONITORING SYSTEM FOR
ENTERING AND LEAVING ACTIVITIES IN DATA CENTER ROOMS
USING THE YOLOV5 AND DEEPSORT ALGORITHMS BASED ON THE
INTERNET OF THINGS (IoT)”**

ABSTRACT

In today's digital era, the data center room has a very important role and must be protected to the maximum, both physically and non-physically. This is because the data center serves as a storage place for information technology (IT) devices, ranging from servers to corporate data communication centers. Therefore, strict security is needed to maintain the safety of the company's important assets. This research aims to design a monitoring system for entry and exit activities in the data center room using Internet of Things (IoT) technology and Artificial Intelligence (AI). The YOLO V5 algorithm is used to detect objects, while the DeepSORT algorithm serves to track the movement of detected objects. This system is proposed to solve security problems and improve operational efficiency in data center rooms, which often still use manual methods or less efficient monitoring systems. By using the ESP32-CAM device as an IoT tool to capture real-time video, YOLO V5 will detect objects entering and leaving the data center room, and DeepSORT will track their movements. This system is expected to provide a more effective and efficient solution for monitoring. Through the utilization of IoT and AI. The monitoring data will be saved automatically into the database page, the video will be streamed through the Django-based web interface and displayed on the main page and the monitoring results will also be displayed on the main page.

Keywords:. *YOLOv5, DeepSORT, Internet of Things, monitoring, data centre, Django, Artificial Intelligence, ESP32-CAM.*

DAFTAR ISI

| | |
|--|-------------|
| PERSETUJUAN..... | ii |
| PERNYATAAN..... | iii |
| PENGHARGAAN..... | iv |
| ABSTRAK | vi |
| ABSTRACT | vii |
| DAFTAR ISI..... | viii |
| DAFTAR TABEL..... | x |
| DAFTAR GAMBAR..... | xi |
| BAB I PENDAHULUAN | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Rumusan Masalah..... | 3 |
| 1.3 Batasan Masalah | 4 |
| 1.4 Tujuan Penelitian | 4 |
| 1.5 Manfaat Penelitian | 4 |
| 1.6 Metodologi Penelitian..... | 5 |
| 1.7 Penelitian Relevan | 6 |
| 1.8 Sistematika Penulisan | 7 |
| BAB II LANDASAN TEORI..... | 9 |
| 2.1 Internet of Things (IoT) | 9 |
| 2.2 Sistem Pemantauan | 10 |
| 2.3 Pusat Data | 10 |
| 2.4 ESP32-CAM..... | 10 |
| 2.5 Deep Learning | 11 |
| 2.6 You Only Look Once (YOLOv5)..... | 12 |
| 2.7 DeepSORT (Deep Simple Online Realtime Tracking) | 14 |
| BAB III ANALISIS DAN PERANCANGAN..... | 16 |
| 3.1 Analisis | 16 |
| 3.1.1 Analisis masalah..... | 16 |
| 3.1.2 Analisis kebutuhan | 16 |
| 3.1.3 Analisis Data | 18 |
| 3.2 Perancangan Sistem | 18 |
| 3.2.1 Arsitektur Umum Penelitian | 19 |
| 3.2.2 Arsitektur Perangkat IoT | 20 |

| | | |
|---------------|--|----|
| 3.2.3 | <i>Perancangan alat</i> | 21 |
| 3.2.4 | <i>Rancang Bangun Sistem</i> | 22 |
| 3.3 | Pemodelan Sistem | 24 |
| 3.3.1 | <i>Flowchart</i> (diagram alir) | 24 |
| 3.3.2 | <i>Use Case Diagram</i> | 26 |
| 3.3.3 | <i>Activity Diagram</i> | 27 |
| 3.3.4 | <i>Sequence Diagram</i> | 27 |
| 3.4 | Perancangan Interface | 28 |
| 3.4.1 | Halaman Login | 29 |
| 3.4.2 | Halaman Utama | 29 |
| 3.4.2 | Halaman <i>Database</i> | 30 |
| BAB IV | IMPLEMENTASI DAN PENGUJIAN | 31 |
| 4.1 | Implementasi | 31 |
| 4.1.1 | <i>Setup Perangkat IoT (ESP32-CAM)</i> | 31 |
| 4.1.2 | <i>Training Dataset</i> Menggunakan Algoritma <i>You Only Look Once</i> (YOLO) | 33 |
| 4.1.3 | Inisiasi Algoritma <i>DeepSORT</i> | 39 |
| 4.1.4 | Halaman Login | 43 |
| 4.1.5 | Halaman Utama | 44 |
| 4.1.6 | Halaman <i>Database</i> | 44 |
| 4.2 | Pengujian Fungsional | 45 |
| 4.2.1 | Pengujian Aliran Video | 45 |
| 4.2.2 | Pengujian Deteksi dan Pelacakan | 45 |
| BAB V | PENUTUP | 51 |
| 5.1 | Kesimpulan | 51 |
| 5.2 | Saran | 51 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 3. 1 Perancangan Alat..... | 21 |
| Tabel 4. 1 Hasil Pengujian..... | 48 |

DAFTAR GAMBAR

| | |
|--|----|
| Gambar 2. 1 Ilustrasi IoT | 9 |
| Gambar 2. 2 Perangkat ESP32-CAM | 11 |
| Gambar 2. 3 Arsitektur Deep Learning | 12 |
| Gambar 2. 4 YOLO Object Detection | 13 |
| Gambar 2. 5 Arsitektur Algoritma DeepSORT | 15 |
| Gambar 3. 1 Arsitektur Umum | 19 |
| Gambar 3. 2 Arsitektur Perangkat IoT..... | 20 |
| Gambar 3. 3 Perancangan ESP32-CAM..... | 22 |
| Gambar 3. 4 Sisi Depan | 23 |
| Gambar 3. 5 Sisi Atas dan Bawah | 23 |
| Gambar 3. 6 Sisi Kiri dan Kanan..... | 24 |
| Gambar 3. 7 Flowchart sistem..... | 25 |
| Gambar 3. 8 Use Case Diagram | 26 |
| Gambar 3. 9 Activity Diagram | 27 |
| Gambar 3. 10 Sequencen Diagram | 28 |
| Gambar 3. 11 Interface Halaman Login | 29 |
| Gambar 3. 12 Interface Halaman Utama | 29 |
| Gambar 3. 13 Halaman Database | 30 |
| Gambar 4. 1 Memprogram ESP32-CAM | 31 |
| Gambar 4. 2 Menu untuk memilih Board | 32 |
| Gambar 4. 3 Menu untuk memilih Serial ports | 32 |
| Gambar 4. 4 Kode setting WiFi..... | 32 |
| Gambar 4. 5 Output pada serial monitor..... | 33 |
| Gambar 4. 6 Inisisasi Kamera..... | 33 |
| Gambar 4. 7 Augmentasi dan Labeling di Roboflow | 34 |
| Gambar 4. 8 Clone Library YOLO V5..... | 34 |
| Gambar 4. 9 Install Requirements | 35 |
| Gambar 4. 10 Library Roboflow dan Dataset..... | 35 |
| Gambar 4. 11 Konfigurasi file YAML | 35 |
| Gambar 4. 12 Custom File YAML | 36 |
| Gambar 4. 13 Training Model | 37 |
| Gambar 4. 14 Hasil Pelatihan | 37 |

| | |
|---|----|
| Gambar 4. 15 Hasil Deteksi..... | 38 |
| Gambar 4. 16 Grafik pelatihan Model..... | 39 |
| Gambar 4. 17 Library DeepSORT..... | 39 |
| Gambar 4. 18 Konfigurasi data YAML..... | 40 |
| Gambar 4. 19 Inisiasi DeepSORT | 40 |
| Gambar 4. 20 Deklarasi Fungsi Pelacakan | 42 |
| Gambar 4. 21 Deklarasi Entering dan Leaving..... | 43 |
| Gambar 4. 22 Halaman Login | 43 |
| Gambar 4. 23 Autentikasi akun | 44 |
| Gambar 4. 24 Halaman Utama | 44 |
| Gambar 4. 25 Halaman Database | 45 |
| Gambar 4. 26 Streaming Aktif..... | 45 |
| Gambar 4. 27 Hasil Deteksi Bagus..... | 46 |
| Gambar 4. 28 Hasil Deteksi Erlin..... | 46 |
| Gambar 4. 29 Hasil Deteksi 2 orang..... | 47 |
| Gambar 4. 30 Hasil Penyimpanan Data..... | 47 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pusat data berfungsi sebagai fasilitas penyimpanan sentral untuk beragam peralatan teknologi informasi, mencakup server berkapasitas besar hingga infrastruktur komunikasi data korporat (Setyawan et al., 2021). Tujuan utama dari pusat data adalah mengkonsolidasikan seluruh aset teknologi informasi, bertindak sebagai pusat kendali jaringan, menopang aktivitas bisnis digital, serta menyediakan layanan yang andal dan berkelanjutan untuk operasi pengolahan data yang vital (Nirwana et al., 2018). Pada era digital saat ini, ruangan pusat data adalah tempat yang krusial dan harus dilindungi secara optimal dengan cara pemantauan fisik dan non fisik karena menjadi tempat yang penting untuk berbagai aktivitas komputasi dan penyimpanan data bagi organisasi dan bisnis di era saat ini. Namun, pengelolaan dan pemantauan aktivitas kedatangan dan kepergian perangkat atau personel pada ruangan pusat data masih sering dilakukan secara manual atau menggunakan sistem yang kurang efisien. Kondisi ini menimbulkan beberapa tantangan dalam menjamin keamanan dan efisiensi operasional.

Keamanan pusat data harus dibangun seketat mungkin tujuannya agar aset perusahaan yang berharga dapat tersimpan dengan aman, penjagaan keamanan pusat data dilakukan baik secara fisik maupun non-fisik (Matin et al., 2018). Kebutuhan sistem pemantauan kedatangan dan kepergian untuk sebuah perusahaan besar yang memiliki ruangan pusat data merupakan sebuah kebutuhan mendasar dalam setiap ruangan pusat data tentu saja harus dilengkapi oleh sebuah sistem pemantauan yang canggih dan akurat demi menjaga kesterilan atau keamanan pada ruangan pusat data tersebut. Sistem ini diharapkan mampu meningkatkan efisiensi operasional dan keamanan ruangan pusat data.

Dalam rangka untuk memecahkan permasalahan tersebut dibutuhkan sebuah Solusi yang tepat agar sistem dapat bekerja sebaik mungkin, yaitu memanfaatkan *Internet of Thing (IOT)* dan *Artificial Intelligence (AI)* Perancangan berfokus pada pembuatan sistem monitoring aktivitas kedatangan dan kepergian

pada ruangan pusat data secara *real-time* menggunakan teknologi *Internet Of Thing* (IoT), algoritma YOLOV5 dan Deepsort.

YOLO termasuk dalam jajaran metode deteksi objek yang paling efisien, menunjukkan performa *real-time* yang unggul serta tingkat ketepatan yang tinggi dan telah ditingkatkan sejak diusulkan (Fang et al., 2020). YOLO dikatakan cepat karena YOLO hanya perlu memasukkan gambar ke dalam jaringan untuk mendapatkan hasil deteksi akhir, sehingga yolo juga dapat merealisasikan deteksi waktu video, YOLO secara langsung menggunakan global gambar untuk deteksi, yang dapat menyandikan informasi global dan mengurangi kesalahan dalam mendeteksi latar belakang sebagai objek (Jiang et al., 2021). YOLO menerapkan prinsip kerja dengan membagi citra input menjadi beberapa sel *grid*, lalu memprediksi *bounding box* dan probabilitas untuk setiap sel tersebut. Metode ini menugaskan sel yang mengandung pusat suatu objek untuk bertanggung jawab dalam mendeteksi objek tersebut. Selanjutnya, tiap sel melakukan prediksi *bounding box* serta menghitung nilai keyakinan (*confidence score*) untuk setiap *bounding box* yang terdeteksi (Sugandi & Hartono, 2022).

Metode selanjutnya ialah memasukan algoritma *tracking* yaitu *DeepSort*. *DeepSORT* (*Deep Simple Online Realtime Tracking*) adalah algoritme pelacakan objek mutakhir yang menggabungkan detektor objek berbasis pembelajaran mendalam dengan algoritme pelacakan untuk mencapai akurasi dan ketahanan yang tinggi di lingkungan yang ramai dan kompleks (Ahmad, n.d.). *DeepSort* menggunakan konsep ReID (*Re-identification*), ReID adalah teknik untuk mengenali objek yang sama tetapi dari *bounding box* yang berbeda. Dalam *DeepSORT* diperlukan pengaturan konfigurasi seperti *max_dist*, *min_confidence*, *max_iou_distance Indeks IoU max_age*. Keluaran dari tahap ini adalah sistem mampu mendeteksi dan menandai setiap kendaraan yang terdeteksi dengan *ID* tracking (Setya Budi et al., 2024).

Setelah tahapan pada prosesi sudah disiapkan selanjutnya ialah mempersiapkan alat atau kamera yang akan digunakan untuk memantau aktivitas dengan memanfaatkan *Internet of Thing* (IoT). IoT merupakan paradigma yang memungkinkan berbagai benda yang terhubung ke jaringan untuk saling bertukar data secara mandiri. Konsep ini meniadakan kebutuhan akan interaksi langsung

antar manusia atau antara manusia dengan perangkat komputer.. (Wibisono Darmawan et al., n.d.). IoT berdampak pada sebagian besar aplikasi seperti perawatan kesehatan, rumah pintar, pertanian pintar, pabrik otomatisasi dan industri 4.0, sistem transportasi cerdas, kota pintar, pemantauan infrastruktur, industri ritel, pemantauan lingkungan, air pintar, dan jaringan listrik, dll (Swamy & Kota, 2020). Inovasi seperti IoT dapat memberikan peluang untuk memudahkan pekerjaan setiap manusia.

Alat – alat yang dipersiapkan untuk mendukung penelitian ini antara lain seperti, ESP32-CAM, FTDI, kabel jumper, dan kabel mini usb. Dengan alat tersebut memungkinkan untuk memprogram perangkat agar saling terhubung dengan sistem yang akan dirancang. Alur sistem yang akan dirancang adalah, ESP32-CAM sebagai perangkat IoT untuk menangkap dan mengirimkan video *streaming*, YOLOV5 sebagai algoritma pendeteksi objek dalam video *streaming*, *DeepSort* sebagai algoritma untuk melacak pergerakan individu yang terdeteksi, Django sebagai platform backend untuk mengelola video, Database untuk menyimpan data terkait aktivitas dan log, dan mendesain web antarmuka untuk menampilkan hasil monitoring berupa video *live streaming* dan data *historis*. Pemanfaatan antara infrastruktur IoT dengan algoritma YOLOV5 sebagai deteksi dan algoritma *DeepSort* untuk pelacakan objek diharapkan memberikan solusi yang tepat untuk kebutuhan sistem yang akurat dan efisien.

1.2 Rumusan Masalah

Pada era digital saat ini, ruangan pusat data adalah tempat krusial yang harus dilindungi secara optimal. Hal ini dikarenakan pusat data menjadi tempat yang penting untuk berbagai aktivitas komputasi dan penyimpanan data bagi organisasi dan bisnis di era saat ini. Pemantauan pusat data dilakukan secara fisik dan non-fisik secara ketat agar berbagai aset perusahaan yang berharga dapat tersimpan dengan aman. Salah satu upaya pemantauan pusat data yang dapat dilakukan yaitu dengan memantau aktivitas kedatangan dan kepergian perangkat atau personel pada ruangan pusat data menggunakan sistem pemantauan yang mampu mendeteksi aktivitas tersebut demi menjaga kesterilan atau keamanan pada ruangan pusat data. Sistem ini diharapkan mampu meningkatkan efisiensi operasional dan keamanan ruangan pusat data.

1.3 Batasan Masalah

Penelitian ini memiliki batasan masalah dalam beberapa hal seperti:

1. Ruang lingkup penelitian ini hanya pada ruangan pusat data sebagai area pemantauan, dengan fokus pemantauan aktivitas kedatangan dan kepergian individu pada area tersebut.
2. Perangkat IoT yang digunakan pada penelitian ini adalah ESP32-CAM sebagai kamera untuk menangkap video *live streaming*.
3. Algoritma yang digunakan hanyalah YoloV5 untuk mendeteksi objek dan algoritma DeepSort untuk pelacakan objek.
4. Sistem dirancang hanya untuk bekerja dalam kondisi normal dengan pencahayaan yang cukup.
5. Hasil data monitoring hanya dibatasi pada penggunaan *framework* Django, fokus pada penyimpanan di server lokal.
6. Evaluasi kinerja sistem monitoring akan dilakukan berdasarkan metrik tertentu seperti akurasi deteksi, kecepatan respons dan simulasi keamanan.
7. Sistem tidak membahas tentang aspek keamanan data hasil monitoring.

1.4 Tujuan Penelitian

Tujuan dari penelitian yaitu untuk mengembangkan sebuah sistem monitoring dengan pemanfaatan *Internet of Thing* (IOT) dan *Artificial Intelligence* (AI) yang efektif, hemat biaya dan energi untuk monitoring aktivitas kedatangan dan kepergian staff atau personel pada ruangan data centre secara *real-time*.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sistem dapat memberikan kenyamanan, kemudahan, penghematan biaya dan efisien bagi staff atau personel dalam memonitoring aktivitas kedatangan dan kepergian tanpa perlu melakukan pencatatan manual serta diharapkan penelitian ini memberikan kontribusi pada pengembangan teknologi berbasis IoT dan AI dalam bidang keamanan fisik.

1.6 Metodologi Penelitian

1. Studi Pustaka

Pada tahapan pertama penelitian akan dilakukan pengumpulan berbagai sumber bacaan atau referensi yang ada pada jurnal, artikel ilmiah ataupun internet yang bertujuan untuk memahami konsep *Internet of things* dengan pemanfaatan fitur *Artificial Intelligence* dan memahami konsep untuk rancang bangun sistem monitoring.

2. Analisis dan Perancangan Sistem

Pada tahap ini akan melakukan analisis kebutuhan dengan mengidentifikasi informasi yang ingin dipantau seperti aktivitas kedatangan dan kepergian, dan melakukan perancangan arsitektur sistem dengan memilih platform IoT yang sesuai dan mempertimbangkan infrastruktur perangkat keras dan perangkat lunak yang diperlukan. Perancangan arsitektur akan dibangun sesuai dengan permasalahan dan perancangan sistem, perancangan sistem ini akan dirancang dalam bentuk diagram alir untuk menggambarkan kinerja penelitian.

3. Pengumpulan Data

Pada penelitian ini menggunakan *dataset* primer, dataset primer adalah data yang didapatkan atau diperoleh dengan cara observasi atau terjun langsung kelapangan. *Dataset* yang digunakan adalah gambar wajah yang dikumpulkan sendiri oleh peneliti.

4. Implementasi Sistem

Tahap ini adalah proses perancangan dan pembuatan sistem berdasarkan diagram alir yang dirancang serta menghubungkan sistem dengan perangkat IoT yang telah dibangun, integrasi dengan server Django untuk pengelolaan video *streaming* dan menyimpan data hasil deteksi dan pelacakan ke dalam *database* lokal.

5. Pengujian Sistem

Melakukan pengujian akurasi deteksi dan pelacakan, selanjutnya menguji kecepatan pemrosesan video *streaming* dari ESP32-CAM hingga hasil deteksi dan pelacakan untuk ditampilkan ke web antarmuka pengguna.

6. Dokumentasi Sistem

Pada tahapan ini akan dilakukan penyusunan laporan dalam bentuk skripsi dimulai dari tahapan analisis hingga pengujian sistem.

1.7 Penelitian Relevan

Penelitian ini dilakukan dengan melihat beberapa penelitian yang relevan terdahulu, antara lain:

1. Penelitian yang dilakukan oleh (Razzok et al., 2023) yang berjudul “Sistem Deteksi Laju dan Plat Nomor Kendaraan Berbasis Video Rekaman Menggunakan YOLOv5-DeepSORT dan HyperLPR” menjelaskan bahwa Aplikasi penghitung kecepatan mobil berbasis YOLO berhasil diterapkan Berdasarkan hasil tersebut sistem dapat mendeteksi objek kendaraan dengan akurat, dengan memiliki nilai Precision sebesar 100%. Hasil tersebut didapatkan karena masukan yang digunakan pada pengujian berupa video yang memiliki kualitas yang baik serta fps yang memadai, lalu dalam mengukur keakuratan deteksi laju kendaraan dilakukan dengan menggunakan Mean Absolute Percentage Error (MAPE) Nilai MAPE pada sistem didapatkan sebesar 7,2%. Semakin kecil persentase MAPE, maka performa dari sistem tersebut akan semakin baik, sedangkan pengujian deteksi plat nomor kendaraan dilakukan dengan menguji 10 kendaraan yang terdeteksi pada beberapa masukan video. Hasil pengujian tersebut didapatkan hasil akurasi karakter secara keseluruhan sebesar 94.82%. Berdasarkan hasil pendeteksian karakter menggunakan algoritma HyperLPR dihasilkan nilai yang lebih baik dari CNN, Optical Character Recognition (OCR) dan OCR.
2. Penelitian yang dilakukan oleh Razzok et al., (2023) yang berjudul “Pedestrian Detection and Tracking System Based on Deep-SORT, YOLOv5, and New Data Association Metrics” menjelaskan bahwa penggunaan kombinasi algoritma Deep-SORT dan YOLOv5 dapat meningkatkan akurasi pelacakan pejalan kaki dengan lebih baik. Penelitian ini berhasil mengatasi tantangan occlusion (tumpang tindih antar pejalan kaki) dan meningkatkan performa pelacakan objek ganda (MOT) dengan mengembangkan matriks biaya baru untuk asosiasi data. Evaluasi terhadap dataset MOT17 menunjukkan peningkatan kinerja dibandingkan dengan matriks default berbasis intersection over union (IoU), Namun masih ada beberapa keterbatasan pada penelitian ini

yang ingin di atasi dalam penelitian di masa depan. Salah satu keterbatasan ini adalah kemampuan untuk melacak pejalan kaki dalam jangka waktu yang lebih lama, yang membutuhkan identifikasi ulang individu yang sama dari waktu ke waktu.

1.8 Sistematika Penulisan

Penelitian ini menggunakan metode penulisan sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, penelitian relevan, dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Bab ini akan menjelaskan beberapa konsep yang memiliki hubungan dengan penelitian, seperti *Internet of Things (IoT)*, *You Only Look Once*, *Microcontroller ESP32*, dan *DeepSort*

BAB 3 ANALISIS DAN PERANCANGAN

Pada bab ini akan menjelaskan hal-hal yang dianalisis serta membuat perancangan dari rancangan sistem yang menggunakan *ESP32-CAM*, algoritma *YOLO*, *DeepSort* serta Django sebagai platform *backend* untuk pemrosesan video dan mengelola aliran data. Selanjutnya membuat rancangan diagram alir terkait sistem tersebut.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan menjelaskan tahap penerapan yang didapat dari sistem yang sudah dibangun serta dilakukan uji pada sistem untuk mendapatkan hasil dari analisis kinerja sistem tersebut.

BAB 5 KESIMPULAN DAN SARAN

Pada bab ini akan menjelaskan beberapa simpulan berupa rangkuman berdasarkan pemaparan beberapa bab sebelumnya dan juga berupa saran

yang dapat peneliti gunakan sebagai masukan untuk penelitian selanjutnya.

BAB II LANDASAN TEORI

2.1 *Internet of Things (IoT)*

Internet of Things merupakan inovasi teknologi yang memfasilitasi interkoneksi antar benda di lingkungan kita melalui infrastruktur internet. Konsep ini menggambarkan sebuah ekosistem di mana berbagai objek dibekali kemampuan untuk mengirim dan menerima data melalui jaringan yang terhubung (Widodo et al., 2020).

Internet of Things (IoT) bertujuan untuk menghubungkan dunia fisik dan virtual dengan berinteraksi dan bertukar data melalui internet (Rehman et al., 2022). *Smart cities, smart homes, connected vehicles, linked industries, smart energy, smart agriculture, health care, connected buildings and campuses*, ini adalah contoh dari pengaplikasian IoT.



Gambar 2. 1 Contoh penerapan IoT
(Sumber : (itbox.id))

Untuk dapat terkoneksi dengan internet, setiap perangkat perlu dilengkapi dengan alamat Internet Protocol (IP) yang unik. Alamat IP ini berfungsi sebagai pengenalan dalam jaringan, memungkinkan perangkat tersebut menerima instruksi dari perangkat lain yang berada dalam jaringan yang sama. Selanjutnya, alamat-alamat IP dari berbagai perangkat ini akan diintegrasikan ke dalam infrastruktur internet yang lebih luas.

Pada dasarnya, IoT beroperasi dengan menggunakan serangkaian instruksi atau perintah pemrograman. Setiap perintah ini menghasilkan bahasa yang dapat

diinterpretasikan oleh perangkat-perangkat yang saling terhubung. Proses komunikasi ini berlangsung secara otomatis tanpa memerlukan intervensi atau keterlibatan langsung dari pengguna bahkan dari jarak yang sangat jauh (Selay et al., 2022).

2.2 Sistem Pemantauan

Pemantauan adalah proses pengamatan yang mencakup kesadaran terhadap informasi yang ingin diketahui. Pemantauan yang lebih mendalam dilakukan untuk mengukur perubahan dari waktu ke waktu, yang menunjukkan kemajuan menuju tujuan atau sebaliknya. (Widiastuti & Susanto, n.d.) Tujuan utama sistem pemantauan adalah untuk memberikan informasi yang akurat dan *real-time* tentang kondisi atau aktivitas yang dipantau, sehingga pengguna sistem dapat memantau, mengawasi, atau membuat keputusan yang tepat berdasarkan informasi yang mereka terima.

2.3 Pusat Data

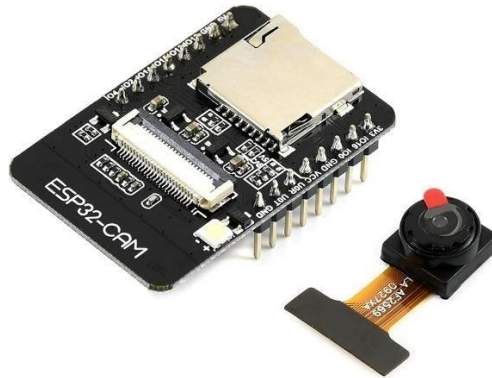
Pusat Data merupakan fasilitas yang dirancang untuk menyimpan, mengelola dan memproses data secara efisien. Pusat Data berupa kumpulan server, perangkat penyimpanan dan infrastruktur yang saling terhubung. Pusat data berfungsi untuk mendukung penyimpanan data besar dalam menjalankan sebuah bisnis. Keamanan dan keandalan data center adalah kunci, mengingat data yang disimpan di dalamnya sering kali bersifat sensitif dan krusial bagi organisasi.

Telecommunication Industry Association (TIA-942) mendefinisikan Pusat data sebagai sebuah struktur atau bagian dari struktur yang berfungsi utama sebagai ruang komputer beserta area pendukungnya Tujuan pokok data center meliputi mengkonsolidasikan seluruh sumber daya teknologi informasi, menjadi pusat operasional jaringan, mendukung aktivitas bisnis digital, dan menyediakan layanan yang berkelanjutan tanpa interupsi untuk proses pengolahan data yang vital (Nirwana et al., 2018)

2.4 ESP32-CAM

ESP32-CAM merupakan perangkat yang dibekali dengan kamera berkualitas tinggi, termasuk model OV2640. Perangkat ini dilengkapi kemampuan terhubung melalui jaringan nirkabel dan bluetooth dengan konsumsi daya rendah. Selain itu,

modul ini menyediakan slot MicroSD untuk penyimpanan tambahan. ESP32-CAM memiliki kemampuan untuk mentransmisikan video secara real-time dan melakukan pengenalan wajah. Dengan menggunakan ESP32-CAM, pengguna dapat mengembangkan sistem berbasis Internet of Things, misalnya CCTV yang terhubung ke internet dan dapat diprogram menggunakan platform Arduino IDE (Ahmad Rio et al., 2022).



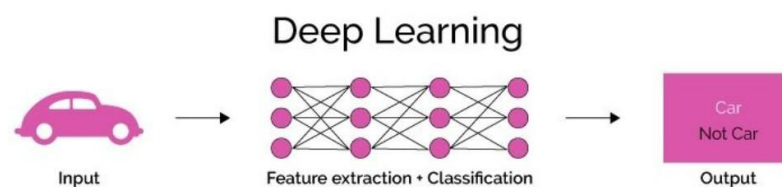
Gambar 2. 2 Perangkat ESP32-CAM
(Sumber : (amazon.ca))

ESP32 dirilis oleh Espressif Systems pada tahun 2016. Mikrokontroler yang kuat dan murah yang dirilis dalam bentuk chip, yang dibangun berdasarkan teknologi TSMC 40nm. ESP32 dilengkapi dengan prosesor dual-core 32- prosesor 32-bit, dengan clock 80, 160 atau 240 MHz. Mikrokontroler memiliki kabel khusus untuk menghubungkan kamera OV2640, slot untuk kartu memori microSD dan chip ESP32-S itu sendiri. Gambar bisa jadi disimpan pada kartu SD dalam format JPEG (Salikhov et al., 2021). ESP32-CAM diilustrasikan pada Gambar 2.2.

2.5 *Deep Learning*

Deep learning merupakan sebuah teknik dalam pembelajaran mesin yang dilatih agar memiliki kemampuan seperti manusia dalam mengidentifikasi pola abstrak seperti video, gambar dan audio dan membuat keputusan berdasarkan data yang rumit. Algoritma ini beroperasi dengan mereplikasi struktur dan fungsi otak, yang disebut dengan jaringan saraf tiruan (Rochmawati et al., n.d.). Jaringan saraf dalam deep learning terdiri dari beberapa lapisan yang saling terhubung, di mana data diproses secara bertahap melalui lapisan-lapisan tersebut untuk mengekstrak fitur

dan menghasilkan prediksi atau keputusan. Salah satu kelebihan deep learning adalah kemampuannya untuk secara otomatis mengekstrak fitur dari data mentah (seperti gambar, teks, atau suara) tanpa memerlukan pra-pemrosesan secara manual. Setiap lapisan pada jaringan saraf bertugas mengenali fitur yang lebih kompleks, mulai dari bentuk sederhana hingga objek yang lebih rumit. Hal ini sangat bermanfaat dalam berbagai aplikasi, seperti pengenalan wajah, dan suara.



Gambar 2.3 Arsitektur Deep Learning
(sumber : (viso.ai))

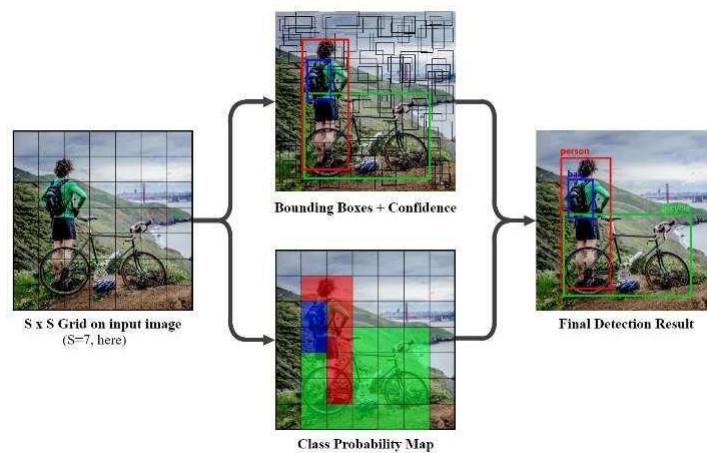
Berdasarkan gambar ilustrasi 2.3 diatas dapat dijabarkan, gambar kucing adalah inputan data mentah yang akan diproses oleh model *deep learning* tahapan selanjutnya ialah fitur ekstrasi dan klasifikasi, fitur tersebut termasuk kedalam komponen penting dalam *deep learning* yang terdiri dari lapisan – lapisan neuron yang saling berhubungan, dimana setiap lapisan memiliki tugas tertentu untuk mengidentifikasi gambar yang telah di input, sebagai contoh pada gambar meng input gambar kucing kemudian pada tahapan ekstraksi dan klasifikasi fitur akan memproses piksel gambar tersebut secara bertahap. Lapisan pertama mendeteksi garis tepi, lapisan kedua mengenali bentuk dan lapisan terakhir akan membuat sebuah Keputusan akhir. Tahapan selanjutnya setelah melewati tahapan ekstrasi dan klasifikasi, model akan menghasilkan output apakah gambar tersebut adalah kucing atau bukan kucing.

2.6 You Only Look Once (YOLOv5)

Metode You Only Look Once (YOLO) adalah metode yang efektif untuk mendeteksi objek secara cepat dan akurat. Metode ini bahkan dapat melebihi kemampuan algoritma lain hingga dua kali lipat (Rahma et al., 2021). Kelebihan utama YOLO adalah efisiensinya, dimana YOLO mampu mendeteksi objek dengan kecepatan yang tinggi tanpa mengorbankan akurasi yang signifikan. Metode ini merupakan pendekatan mutakhir yang dapat mengolah citra secara

langsung dengan performa unggul, menjadikannya sesuai untuk beragam penerapan yang memerlukan tanggapan segera seperti kendaraan otonom, pemantauan keamanan, dan identifikasi wajah (Nafis Alfarizi et al., 2023). Pada 2015, tim peneliti yang dipimpin oleh Joseph Redmon memperkenalkan sistem pionir dalam deteksi objek yang mengintegrasikan seluruh langkah krusial untuk mengenali objek menggunakan satu jaringan saraf. Algoritma YOLO (You Only Look Once) (Do Thuan, n.d.). YOLO memiliki banyak versi yang sering digunakan mulai dari YOLOv1, YOLOv2 hingga versi terbaru adalah YOLOv9.

Sejalan dengan meningkatnya kepopuleran, YOLO telah berkembang menjadi salah satu teknik pengenalan objek yang paling luas penggunaannya dalam ranah pengolahan citra digital. Beragam studi dan inisiatif telah memanfaatkan YOLO untuk berbagai penerapan, yang memacu kemajuan dan inovasi dalam teknologi identifikasi objek. Pada penelitian kali ini menggunakan YOLOv5, satu bulan setelah YOLOv4 dirilis peneliti Glenn Joacher beserta tim nya menerbitkan versi terbaru dari keluarga YOLO yaitu YOLOv5. YOLOv5 memiliki struktur yang hampir sama dengan YOLOv4 tetapi versi 5 lebih terukur, dapat di generalisasi dan lebih fleksibel.



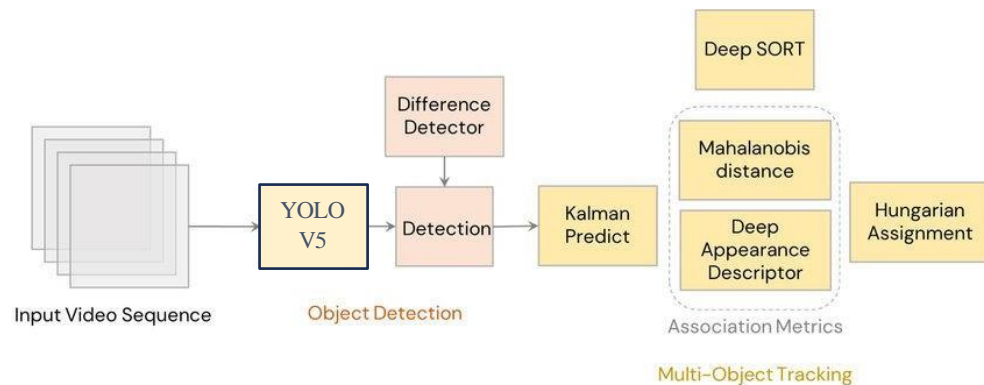
Gambar 2. 4 YOLO Object Detection
(sumber : (researchgate.net))

Berdasarkan ilustrasi dari gambar 2.4 dapat dijelaskan bagaimana cara kerja dari algoritma YOLO, proses dimulai dari sebuah citra asli yang ingin di deteksi seperti gambar diatas terdapat mobil yang akan di deteksi oleh algoritma ini, kemudian citra dibagi menjadi *grid* (jaring-jaring) dengan ukuran $S \times S$ pembagian

ini berfungsi untuk memudahkan deteksi dari sebagian citra yang berbeda-beda, selanjutnya tahapan generalisasi kelas peta probabilitas, pada tahap ini setiap sel *grid* model akan menghasilkan kelas peta probabilitas peta ini menunjukkan kemungkinan adanya objek didalam nya, probabilitas memiliki nilai 0 – 1. Angka 0 menunjukkan bahwa tidak ada objek didalam sel tersebut dan angka 1 menunjukkan kemungkinan lebih tinggi bahwa ada objek didalam sel tersebut. Selanjutnya tahapan generalisasi *bounding box* dan *confidence score*, setelah mendapatkan kelas peta probabilitas model akan menghasilkan *bounding box* (kotak pembatas) di sekitaran objek yang terdeteksi kemudian model juga menghasilkan *confidence score* (nilai kepercayaan) untuk setiap *bounding box* yang mengartikan seberapa benar bahwa objek yang dideteksi adalah benar, berdasarkan *bounding box* dan *confidence score* yang dihasilkan, model akan dapat menentukan kelas dan Lokasi objek yang ada pada citra. Dalam gambar diatas dapat dilihat model berhasil mendeteksi objek dengan menampilkan *bounding box* dan *confidence score* yang sesuai.

2.7 DeepSORT (Deep Simple Online Realtime Tracking)

DeepSORT adalah algoritma pelacakan objek canggih yang menggunakan *deep learning* untuk meningkatkan akurasi dan ketangguhan pelacakan objek dalam aliran video *real-time*. Algoritma ini merupakan perluasan dari algoritma *SORT*, yang merupakan algoritma pelacakan online yang sederhana dan efisien (Razzok et al., 2023). *SORT* memiliki kelemahan dalam melacak beberapa objek yang berdekatan atau terhalang, *DeepSORT* mampu mengatasi masalah ini dengan memanfaatkan *deep learning* untuk menghubungkan deteksi objek yang sama di berbagai frame. *DeepSORT* menggunakan konsep ReID (*Re-identification*) untuk mengidentifikasi dan melacak objek dari satu bingkai ke bingkai berikutnya, *ReID* adalah teknik yang membandingkan fitur visual dari objek untuk mengenali objek yang sama di berbagai bingkai (Setya Budi et al., 2024). Dengan menggunakan *DeepSORT*, sistem ini dapat melacak individu secara konsisten di berbagai frame video, bahkan jika individu tersebut sempat menghilang dari pandangan dan kemudian muncul kembali.



Gambar 2. 5 Arsitektur Algoritma DeepSORT
(sumber : (medium.com))

Proses pelacakan dimulai dengan menerima inputan frame video yang berisi informasi visual dari setiap adegan yang direkam. Model YOLOv5 digunakan untuk mendeteksi objek-objek pada setiap frame, seperti kendaraan atau manusia, dan hasilnya berupa kotak pembatas (bounding box) serta label objek yang terdeteksi, seperti penjelasan yang sebelumnya. Informasi dari hasil deteksi ini kemudian digunakan dalam langkah berikutnya. Dengan menggunakan kalman filter, sistem memprediksi posisi objek pada frame berikutnya berdasarkan lokasi objek pada frame sebelumnya, sambil memperhitungkan ketidakpastian pengukuran. Hasil prediksi ini dibandingkan dengan hasil deteksi terkini melalui tahap Difference Detector untuk menilai keakuratan deteksi. Fitur-fitur visual yang unik diekstraksi dari setiap objek oleh model Deep Appearance Descriptor, yang membantu membedakan objek yang berbeda. Kemudian, Mahalanobis Distance digunakan untuk mengukur tingkat kemiripan antara fitur visual dari dua objek yang berbeda, di mana jarak yang lebih kecil menunjukkan kemiripan yang lebih tinggi. Algoritma Hungarian Assignment dipakai untuk mencocokkan hasil deteksi terbaru dengan data pelacakan dari frame sebelumnya, dengan fokus meminimalkan jarak Mahalanobis. Semua tahapan ini digabungkan oleh algoritma *DeepSORT* untuk memberikan pelacakan objek multi-frame yang lebih akurat.

BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis

Analisis adalah suatu proses yang dilakukan untuk memecah situasi atau masalah yang akan diselesaikan menjadi bagian yang lebih kecil sehingga dapat diatasi oleh sistem atau program dengan lebih mudah. Tujuan analisis adalah untuk menemukan dan memahami secara mendalam tentang suatu masalah atau situasi, dengan cara mengidentifikasi, mengevaluasi dan memahami informasi yang di dapat guna mengambil keputusan yang tepat.

3.1.1 Analisis masalah

Analisis masalah adalah proses pemecahan suatu masalah dengan mengamati sebab akibat sehingga nantinya akan didapatkan Solusi yang efektif. Pada penelitian kali ini permasalahan yang dijumpai adalah, Pusat data beroperasi selama 24/7 tentu saja pasti memerlukan pengawasan yang efektif dan hemat biaya untuk memastikan keamanan ruangan. Namun, pengelolaan dan pemantauan aktivitas kedatangan dan kepergian pada ruangan pusat data tidak jarang masih dapat dijumpain dilakukan secara manual, dalam setiap ruangan pusat data tentu saja harus dilengkapi oleh sebuah sistem pemantauan yang canggih demi menjaga kesterilan atau keamanan pada ruangan pusat data tersebut, untuk mendapatkan sebuah sistem canggih tersebut tentu saja memiliki harga yang relatif mahal, sehingga nantinya akan menimbulkan pembengkakan pada biaya operasional. Oleh karena itu diperlukannya sebuah sistem monitoring yang hemat biaya dan energi, sistem ini diharapkan mampu meningkatkan efisiensi operasional sehingga akan terjadi penghematan biaya dalam jangka panjang.

3.1.2 Analisis kebutuhan

Analisis kebutuhan berfungsi untuk menjelaskan tahapan dalam menentukan dan memahami kebutuhan dari sistem yang akan dirancang atau kondisi yang harus dipenuhi sesuai kebutuhan sistem untuk mencapai tujuannya. Dalam penelitian

ini analisis kebutuhan terbagi menjadi dua bagian utama, yaitu kebutuhan fungsional dan non-fungsional.

1. Kebutuhan fungsional

Kebutuhan fungsional adalah kebutuhan yang berisi rincian atau layanan yang nantinya harus disediakan oleh sistem. Adapun beberapa kebutuhan fungsional untuk penelitian ini yaitu:

- Sistem dapat mendeteksi objek yang masuk dan keluar ruangan data center dengan menggunakan algoritma YOLOv5.
- Sistem dapat melacak pergerakan objek yang terdeteksi dengan menggunakan algoritma *DeepSORT*.
- ESP32-CAM harus mampu menampilkan video secara *realtime* ke server untuk diproses oleh sistem.
- Sistem harus dapat menyimpan log aktivitas kedatangan dan kepergian ke dalam database lokal, termasuk waktu dan identitas objek.
- Sistem dapat menyediakan antarmuka web berbasis Django untuk menampilkan video *live streaming* serta data historis aktivitas.
- Sistem memiliki kemampuan untuk menyimpan data aktivitas.
- Sistem menyediakan fitur autentikasi pengguna untuk memastikan bahwa tidak semua bisa mengakses server.

2. Kebutuhan non-fungsional

Kebutuhan atau karakteristik sistem yang tidak terkait langsung dengan fungsi atau fitur utamanya disebut sebagai kebutuhan non-fungsional. Kebutuhan ini lebih berkaitan dengan cara sistem beroperasi secara keseluruhan dan mencakup elemen kualitas yang mendukung kinerja. Adapun beberapa kebutuhan non-fungsional pada penelitian ini yaitu:

- Performa

Mencakup kebutuhan untuk sistem beroperasi dengan respons waktu yang cepat dan konsisten dan memproses data seperti yang diinginkan.

- Ketersediaan

Mencakup kebutuhan sistem untuk selalu tersedia dan dapat diakses, tanpa adanya gangguan.

- User experience

Mencakup kebutuhan untuk menyediakan antarmuka yang responsive dan gampang digunakan.

- Hemat Biaya

Sistem dirancang sedemikian rupa untuk mengurangi biaya implementasi dan operasional, baik dari segi perangkat keras (IoT) maupun perangkat lunak yang diterapkan.

3.1.3 Analisis Data

Penelitian menggunakan *dataset* primer yang artinya adalah data di dapatkan oleh penulis sendiri, *dataset* berasal dari foto yang diambil langsung menggunakan *handphone* untuk melatih model YOLO.

Dataset yang digunakan berupa wajah dari beberapa orang termasuk penulis. Berisikan 9 kelas yaitu Bagus , Erlin, Said, Rizki, Nabil, Dustin, Dito, Fatih dan Orang-mencurigakan. Dengan pembagian Bagus sebanyak 224 gambar, Erlin sebanyak 207 gambar, Said sebanyak 155 gambar, Rizki sebanyak 152 gambar, Nabil sebanyak 157 gambar, Dustin sebanyak 150 gambar, Dito sebanyak 169 gambar, Fatih sebanyak 175 dan Orang-mencurigakan sebanyak 116 gambar.

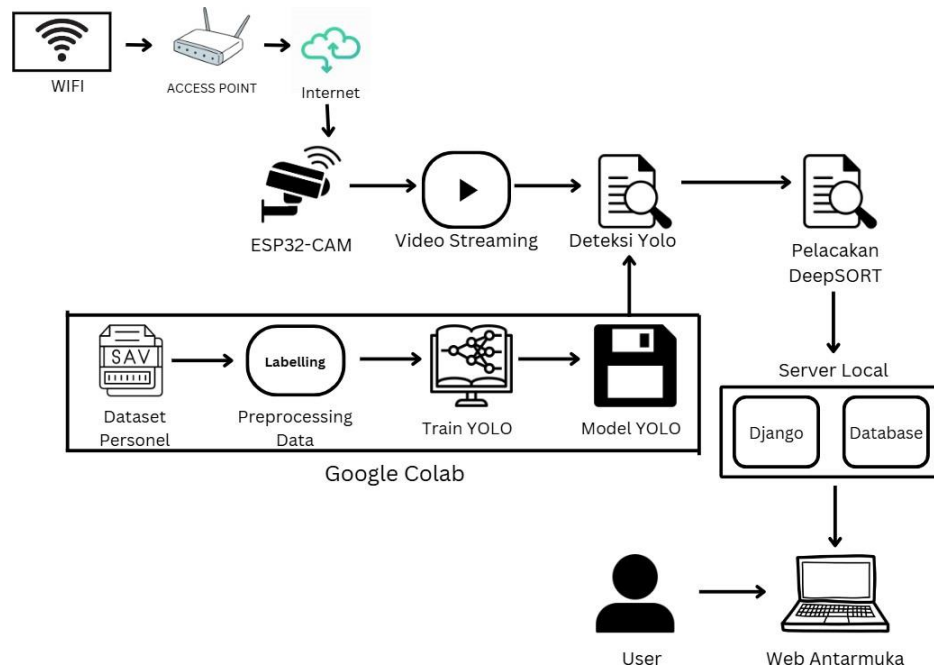
Total *dataset* yang didapat berjumlah 1.326 gambar, *dataset* tersebut terbagi menjadi 958 gambar sebagai data pelatihan, 275 gambar data validasi dan 135 gambar sebagai data tes.

3.2 Perancangan Sistem

Perancangan sistem ialah berupa tahapan yang menjelaskan hasil dari analisis yang sudah dilakukan untuk penelitian ini. Dalam perancangan sistem untuk penelitian ini dibutuhkan bentuk perancangan seperti diagram umum, arsitektur sistem, perancangan alat, rancang bangun sistem dan pemodelan sistem.

3.2.1 Arsitektur Umum Penelitian

Pada penelitian ini, arsitektur umum digunakan sebagai acuan untuk membangun sistem. Berikut ini akan ditampilkan arsitektur umum sistem pada Gambar 3.1 :



Gambar 3. 1 Arsitektur Umum

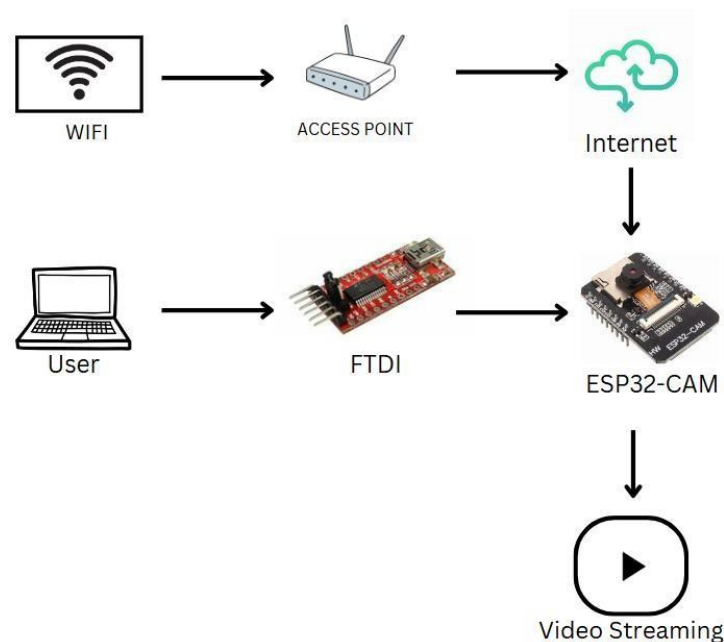
Berdasarkan diagram umum penelitian pada gambar 3.1, di mana penelitian ini dibangun dengan rancangan arsitektur sebagai berikut :

1. Tahapan pertama sambungkan wifi dengan ESP32-CAM.
2. Tahapan pertama, ESP32-CAM berfungsi sebagai kamera yang dihubungkan dengan internet.
3. Selanjutnya kamera akan secara terus menerus merekam video dan menangkap video, streaming video yang telah ditangkap akan diproses oleh YOLO.
4. Tahapan ketiga pelatihan model, Model YOLO memerlukan dataset yang berisi gambar-gambar dengan label tertentu untuk dilatih.
5. Data gambar yang sudah dikumpulkan akan melalui tahap preprocessing agar siap digunakan dalam pelatihan model.
6. Model YOLO kemudian dilatih dengan dataset yang telah diproses untuk mengajarkannya mengenali objek-objek tertentu, lalu model disimpan dan siap untuk dipakai untuk melakukan deteksi pada objek.

7. Video yang diterima kemudian diolah menggunakan algoritma deteksi objek YOLO, yang bertugas mendeteksi objek-objek sesuai model yang dibangun.
8. Setelah objek terdeteksi, algoritma pelacakan *DeepSORT* digunakan untuk melacak pergerakan objek-objek tersebut dari satu frame ke frame berikutnya.
9. Selanjutnya server berfungsi sebagai pusat kendali sistem. Server menjalankan algoritma deteksi dan pelacakan serta menyimpan data yang telah diproses.
10. Kemudian hasil deteksi dan pelacakan akan tampil pada web antarmuka, dan hanya user yang telah terkonfigurasi yang dapat mengakses web antarmuka.

3.2.2 Arsitektur Perangkat IoT

Pada penelitian ini, arsitektur perangkat IoT dirancang untuk menciptakan sebuah sistem yang efisien dalam melakukan tugasnya yaitu memantau, mengolah dan mengirimkan data melalui jaringan internet. Selain itu, arsitektur ini bisa menjadi pedoman untuk memastikan perangkat dapat bekerja secara optimal dalam lingkungan IoT. Berikut ini adalah rancangan arsitektur perangkat IoT yang akan dibangun yang ditampilkan pada Gambar 3.2.



Gambar 3. 2 Arsitektur Perangkat IoT

3.2.3 Perancangan alat

Dalam penelitian ini, dibutuhkan beberapa alat untuk mengimplementasikan sistem IoT. Pada Tabel 3.1 dibawah ini terdapat daftar alat yang digunakan sistem beserta deskripsi fungsinya masing-masing. Adapun rincian dari alat-alat tersebut adalah sebagai berikut :

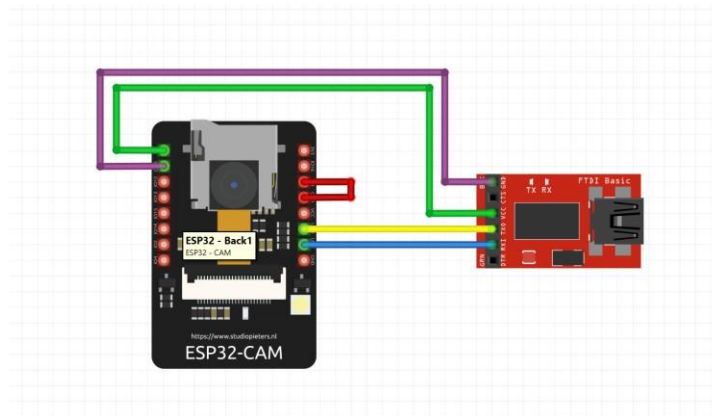
Tabel 3. 1 Perancangan Alat

| No | Alat yang Digunakan | Fungsi |
|----|---------------------|---|
| 1. | ESP32-CAM | Mengambil video |
| 2. | Kabel Jumper | Menjadi penghubung komponen dengan pin pada mikrokontroler |
| 3. | FTDI | FTDI berfungsi untuk mengonversi komunikasi serial TTL (<i>Transistor-Transistor Logic</i>), yang umumnya digunakan oleh mikrokontroler ESP32-CAM, menjadi sinyal USB sehingga dapat terhubung ke komputer. |
| 4. | Kabel Mini USB | Menjadi penghubung antara FTDI ke komputer agar ESP32-CAM dapat di program. |

Prototype sistem yang dirancang dalam penelitian ini dibuat menggunakan perangkat lunak Fritzing. Fritzing memfasilitasi perancangan skematik rangkaian elektronik yang digunakan pada sistem, mulai dari komponen mikrokontroler, sensor, hingga modul komunikasi.

Visualisasi dari hasil perancangan prototipe sistem tersebut disajikan dalam Gambar 3.3. Gambar ini menampilkan tata letak serta koneksi dari setiap komponen yang digunakan dalam implementasi sistem, sehingga memudahkan dalam memahami konfigurasi perangkat keras yang diperlukan untuk pengujian sistem secara fisik. Adanya visualisasi ini juga membantu dalam proses

pembuatan perangkat secara nyata, karena memberikan panduan yang jelas mengenai penyusunan dan pengkabelan komponen secara akurat.



Gambar 3. 3 *Prototype ESP32-CAM*

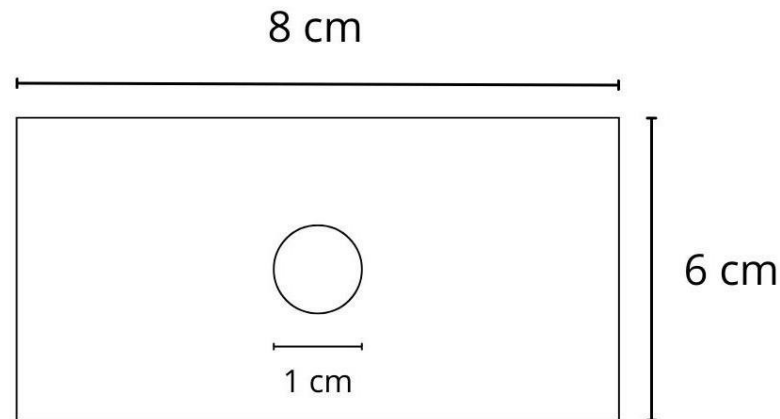
Berikut ini akan menjelaskan masing - masing komponen yang terhubung untuk memprogram ESP32-CAM sebagai berikut :

- Kabel ungu dihubungkan dengan pin *ground* dari FTDI ke ESP32-CAM.
- Kabel hijau pada FTDI dihubungkan dengan pin 5v yang ada pada ESP32-CAM sebagai tegangan positif.
- Kabel hijau pada ESP32-CAM 1 dihubungkan ke pin 5V sebagai aliran positif dan kabel hijau dihubungkan dengan pin Ground sebagai aliran negative dan begitu juga pada ESP32-CAM 2.
- Kabel yang terhubung dengan pin TX pada FTDI dihubungkan dengan pin RX yang ada pada ESP32-CAM.
- Sebaliknya kabel yang tergabung dengan pin RX pada FTDI dihubungkan dengan pin TX yang ada pada ESP32-CAM,
- Kabel merah yang dihubungkan dengan pin Io dan *ground* pada ESP32-CAM berfungsi sebagai kabel jumper, yang nantinya akan di lepas pasang saat pemrograman.

3.2.4 Rancang Bangun Sistem

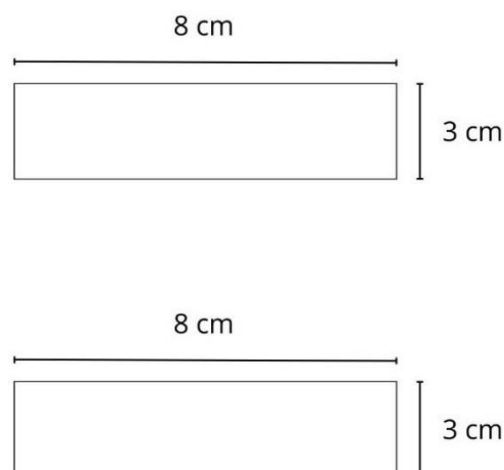
Sistem ini akan ditempatkan di dalam sebuah penutup (*cover*) berbentuk balok yang dirancang untuk melindungi dan menjaga komponen perangkat keras agar tetap aman serta tertata dengan rapi. Cover ini terdiri dari beberapa bagian yang

dirancang sesuai dengan ukuran dan kebutuhan sistem. Adapun bagian dari cover tersebut yaitu sisi cover bagian depan seperti pada Gambar 3.4, sisi cover bagian atas dan bawah seperti pada Gambar 3.5 dan sisi cover bagian kiri dan kanan seperti Gambar 3.6.



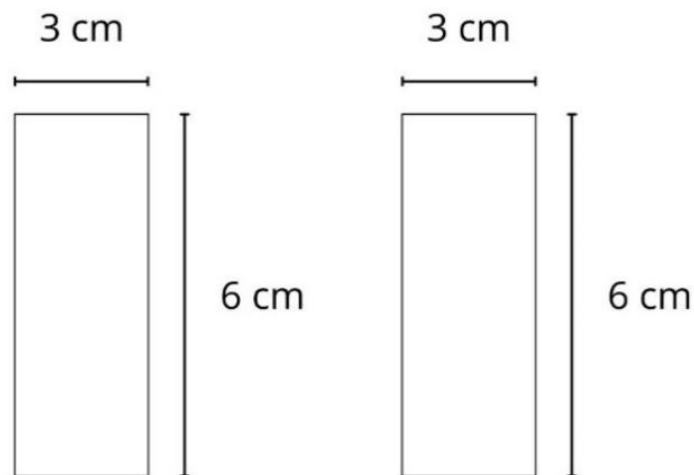
Gambar 3. 4 Sisi Depan

Pada Gambar 3.4 ditampilkan sisi cover bagian depan dengan ukuran 8 cm x 6 cm. Bagian ini dirancang sedemikian rupa untuk memuat tampilan atau indikator yang diperlukan. Pada bagian tengah terdapat lubang dengan diameter 1 cm yang fungsinya untuk meletakkan lensa dari kamera modul ESP32-CAM.



Gambar 3. 5 Sisi Atas dan Bawah

Pada Gambar 3.5 ditampilkan sisi cover bagian atas dan bawah dengan ukuran keduanya yang sama yaitu 8 cm x 3 cm. Sisi ini didesain untuk memberikan struktur yang kokoh dan stabil bagi sistem. Sisi atas berfungsi sebagai pelindung dan penutup kamera, sedangkan sisi bawah berfungsi sebagai alas atau tempat pemasangan sistem di permukaan yang datar.



Gambar 3. 6 Sisi Kiri dan Kanan

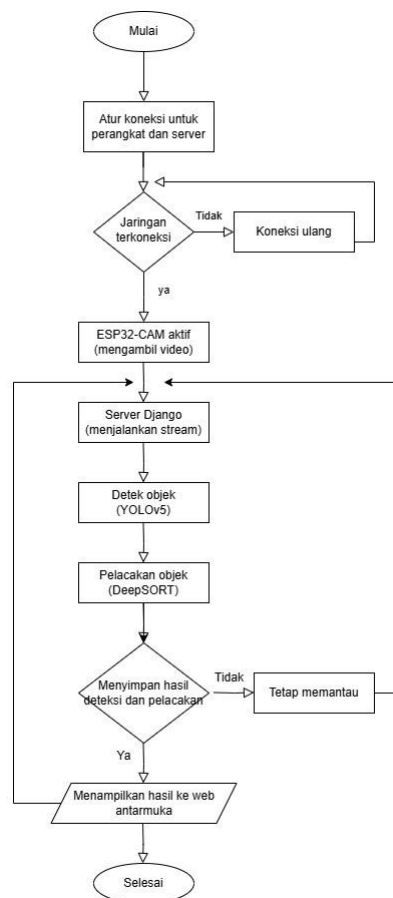
Pada Gambar 3.6 ditampilkan sisi cover bagian kiri dan kanan dengan ukuran keduanya yang sama yaitu 3 cm x 6 cm. Sisi ini menjadi pelengkap struktur penutup kamera modul.

3.3 Pemodelan Sistem

Pemodelan sistem merupakan representasi dari sistem yang akan dikembangkan. Pada penelitian ini, pemodelan sistem akan ditampilkan dan dijelaskan melalui beberapa diagram, yaitu flowchart sistem, diagram use case, diagram aktivitas, dan diagram sequence.

3.3.1 Flowchart (diagram alir)

Flowchart adalah representasi visual dari serangkaian proses atau langkah yang tersusun secara berurutan dalam sebuah sistem. Flowchart digambarkan menggunakan berbagai simbol standar yang menjelaskan alur logika dalam sistem. Pada Gambar 3.7 berikut ini akan ditampilkan flowchart yang digunakan dalam penelitian ini:



Gambar 3. 7 Flowchart sistem

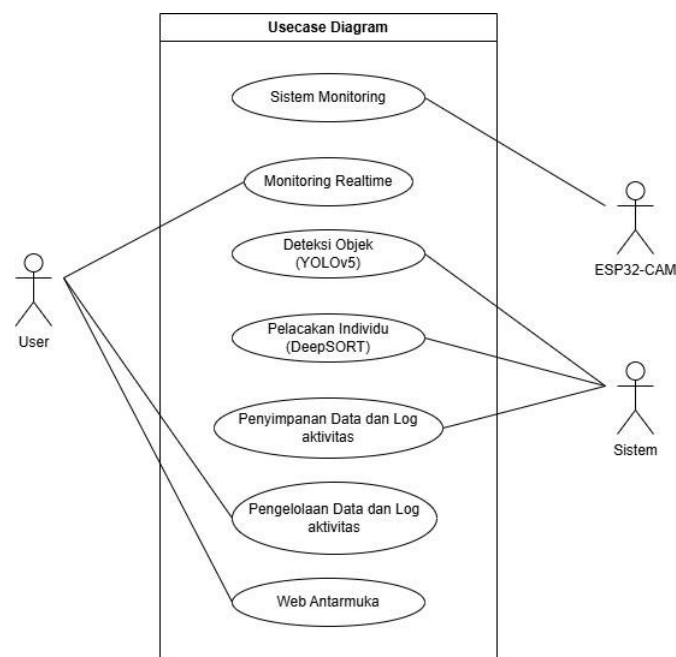
Berdasarkan ilustrasi diatas dapat dijelaskan sebagai berikut :

- Mulai: Proses dimulai dengan menginisialisasi sistem.
- Koneksi: Sistem akan mencoba menghubungkan perangkat (ESP32-CAM) dengan server, Jika koneksi berhasil, proses dilanjutkan ke langkah berikutnya. Jika gagal, sistem akan mencoba menghubungkan ulang.
- ESP32-CAM: Setelah koneksi berhasil, kamera ESP32-CAM diaktifkan untuk mulai merekam video.
- Server Django Memproses Streaming: Video yang direkam oleh ESP32-CAM dikirim ke server dan diproses menggunakan framework Django.
- Deteksi Objek (YOLOv5): Video yang diterima oleh server diproses menggunakan algoritma YOLOv5 untuk mendeteksi objek-objek dalam video.
- Pelacakan Objek (DeepSORT): Setelah objek terdeteksi, algoritma DeepSORT digunakan untuk melacak pergerakan objek dari satu frame ke frame berikutnya.

- Penyimpanan Hasil: Hasil deteksi dan pelacakan disimpan, baik dalam bentuk database maupun file log.
- Tampilan Hasil: Hasil yang disimpan kemudian ditampilkan pada antarmuka web yang dibangun dengan Django. Pengguna dapat mengakses hasil deteksi dan pelacakan melalui antarmuka ini.
- Selesai: Setelah hasil ditampilkan, sistem kembali ke awal untuk terus memantau dan memproses video baru.

3.3.2 Use Case Diagram

Dalam use case diagram ini, akan digambarkan bagaimana interaksi antara pengguna dan sistem yang dirancang. Diagram ini memperlihatkan fungsionalitas sistem dalam menjalankan suatu tindakan serta bagaimana pengguna berinteraksi dengan sistem tersebut.



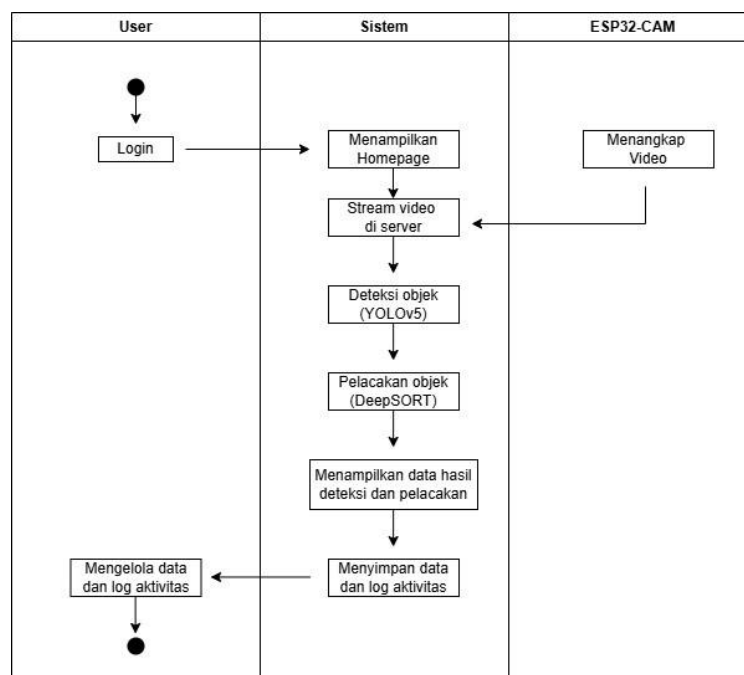
Gambar 3. 8 Use Case Diagram

Pada Gambar 3.8 diatas dapat dijelaskan, alur yang pertama adalah ESP32-CAM Sebagai perangkat keras yang berfungsi menangkap video dari lingkungan sekitar, selanjutnya User dapat memantau secara langsung aktivitas yang sedang berlangsung melalui sistem ini. Kemudian Sistem mampu mendeteksi objek-objek tertentu dalam video yang diambil. Algoritma YOLOv5 digunakan untuk fungsi ini, tidak hanya mendeteksi, tetapi juga melacak pergerakan individu dari

waktu ke waktu. Algoritma DeepSORT digunakan untuk fungsi pelacakan ini. Dan juga Sistem menyimpan semua data yang dihasilkan dari proses deteksi dan pelacakan, termasuk log aktivitas. Kemudian User dapat mengelola data dan log aktivitas yang telah tersimpan, misalnya untuk keperluan analisis lebih lanjut dan User berinteraksi dengan sistem melalui antarmuka web yang disediakan.

3.3.3 Activity Diagram

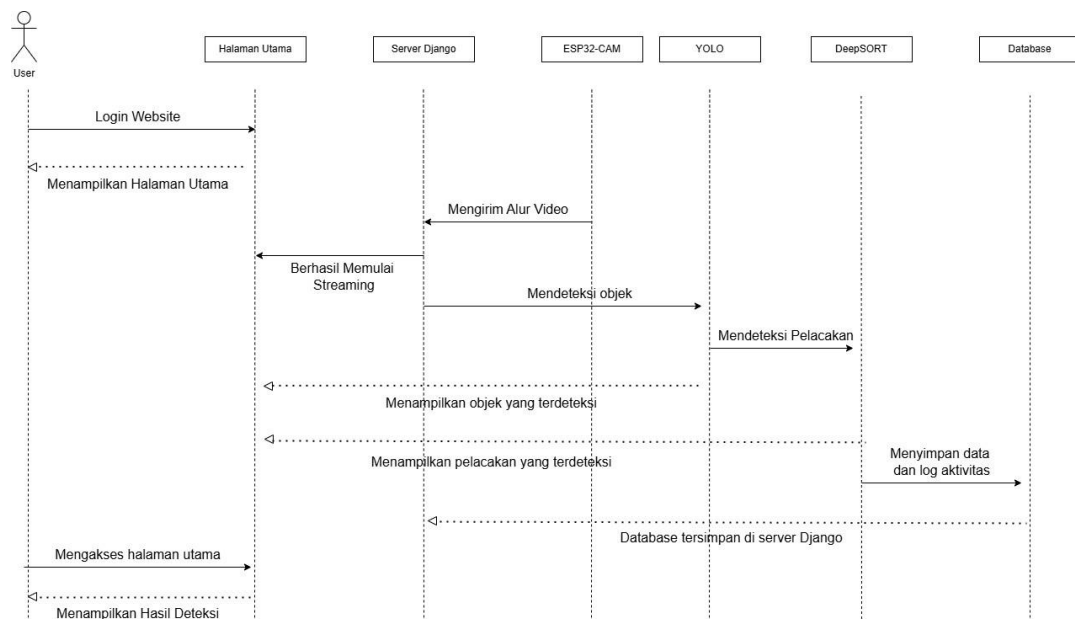
Activity diagram adalah visualisasi dari alur proses yang menggambarkan rangkaian aktivitas dalam suatu sistem, mulai dari awal hingga selesai. Diagram ini menampilkan urutan langkah-langkah, termasuk percabangan, keputusan, dan interaksi antara aktivitas, sehingga memudahkan dalam memahami cara kerja sebuah proses dalam sistem, seperti yang ada pada Gambar 3.9.



Gambar 3. 9 Activity Diagram

3.3.4 Sequence Diagram

Sequence diagram adalah jenis diagram yang menggambarkan urutan interaksi antara berbagai elemen atau aktor dalam sistem dari perspektif waktu. Diagram ini menunjukkan bagaimana pesan atau perintah dikirim antara objek atau entitas dalam sistem sepanjang waktu, memberikan gambaran tentang bagaimana berbagai komponen berkoordinasi untuk mencapai hasil tertentu.



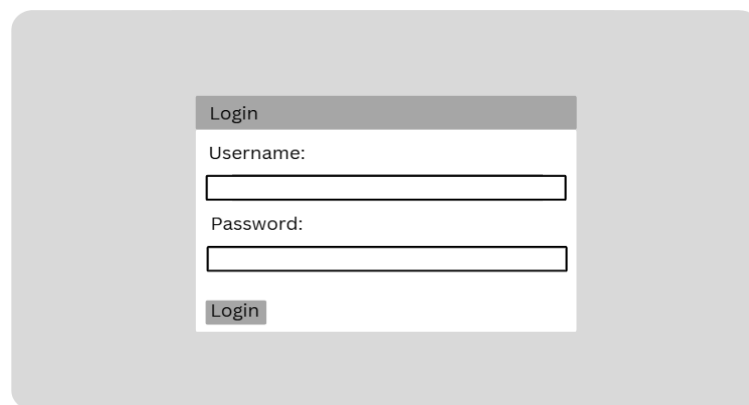
Gambar 3. 10 *Sequence Diagram*

Berdasarkan gambar 3.10 diatas dapat dijelaskan bahwa Pengguna mengakses sistem melalui web, kamera ESP32-CAM secara terus-menerus mengirimkan video ke server, server memproses video menggunakan algoritma YOLOv5 dan *DeepSORT* untuk mendeteksi dan melacak objek, hasil deteksi dan pelacakan ditampilkan kepada pengguna dan disimpan dalam database. Diagram aktivitas ini memberikan gambaran yang jelas tentang bagaimana sistem bekerja dan interaksi antara komponen-komponennya.

3.4 Perancangan *Interface*

Perancangan *interface* melibatkan proses mendesain struktur sistem yang akan dikembangkan. Interface dalam sistem berfungsi sebagai elemen penting yang memungkinkan interaksi antara pengguna dan sistem. Memahami struktur sistem secara mendalam sebelum merancang antarmuka sangat penting untuk memastikan implementasi yang efektif. Desain interface perlu mempertimbangkan kebutuhan pengguna dengan menyediakan tampilan yang intuitif dan mudah digunakan untuk memfasilitasi interaksi yang lancar.

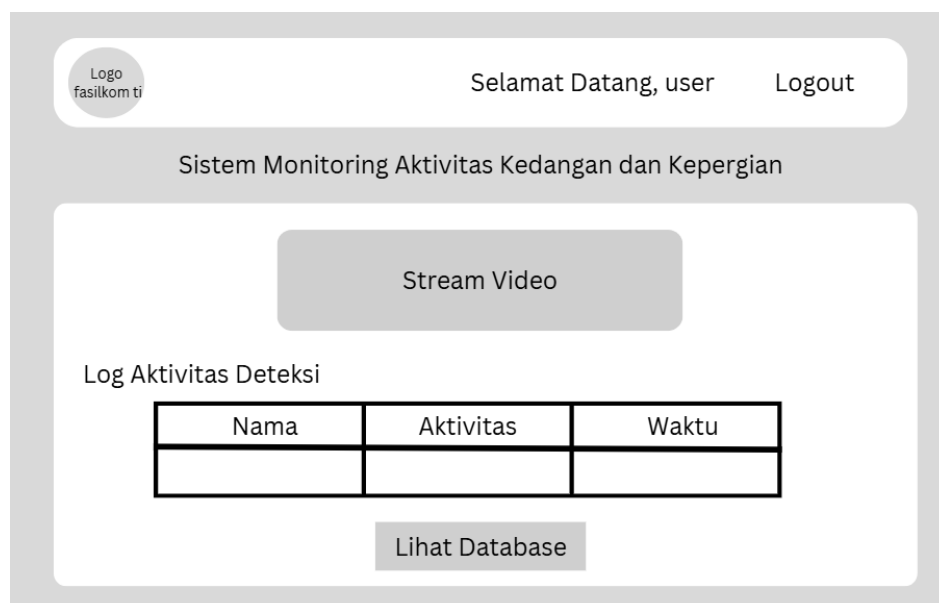
3.4.1 Halaman Login



Gambar 3. 11 *Interface Halaman Login*

Gambar 3.11 diatas menunjukan secara sederhana untuk *interface login*, tampilan ini adalah tampilan pertama saat pengguna menggunakan *website*. Sebelum memasuki halaman utama diharuskan untuk autentikasi user dahulu sebelum nantinya dapat mengakses halaman utama.

3.4.2 Halaman Utama

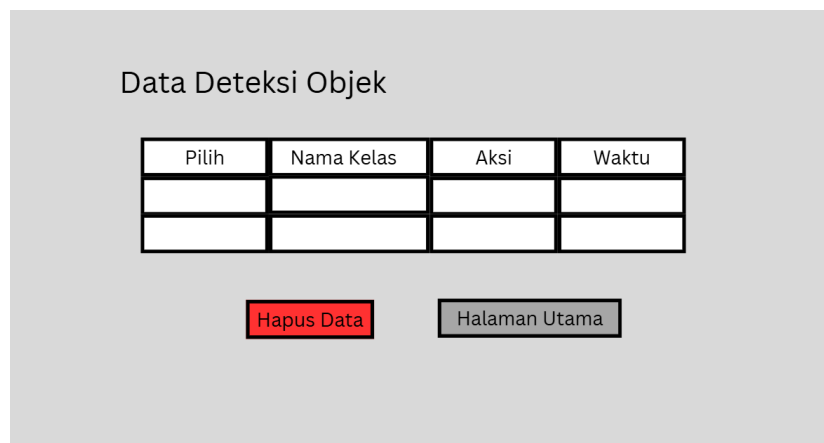


| Nama | Aktivitas | Waktu |
|------|-----------|-------|
| | | |

Gambar 3. 12 *Interface Halaman Utama*

Pada Gambar 3.12 ditampilkan halaman utama, halaman ini hanya bisa diakses oleh admin atau user yang sudah terdaftar dan sudah melakukan login sebelumnya. Gambar menunjukkan bagaimana tampilan halaman utama. Terdapat beberapa fitur *stream* video, dan tabel yang nantinya akan menampilkan data objek yang telah dideteksi yang selalu update setiap 5 detik sekali jika ada objek yang terdeteksi, kemudian terdapat navbar didalamnya terdapat logo, ucapan selamat datang kepada username yang telah terdaftar dan juga ada fungsi *logout* yang apabila di tekan akan otomatis kembali ke halaman *login*.

3.4.2 Halaman *Database*



Gambar 3. 13 Halaman *Database*

Pada Gambar 3.13 adalah *interface* pada halaman *database*, Secara keseluruhan, halaman ini berfungsi sebagai antarmuka yang sederhana dan mudah digunakan oleh pengguna untuk mengelola data hasil deteksi objek. Dapat dilihat terdapat table yang berfungsi untuk menampung data hasil deteksi dan pelacakan di halaman utama.

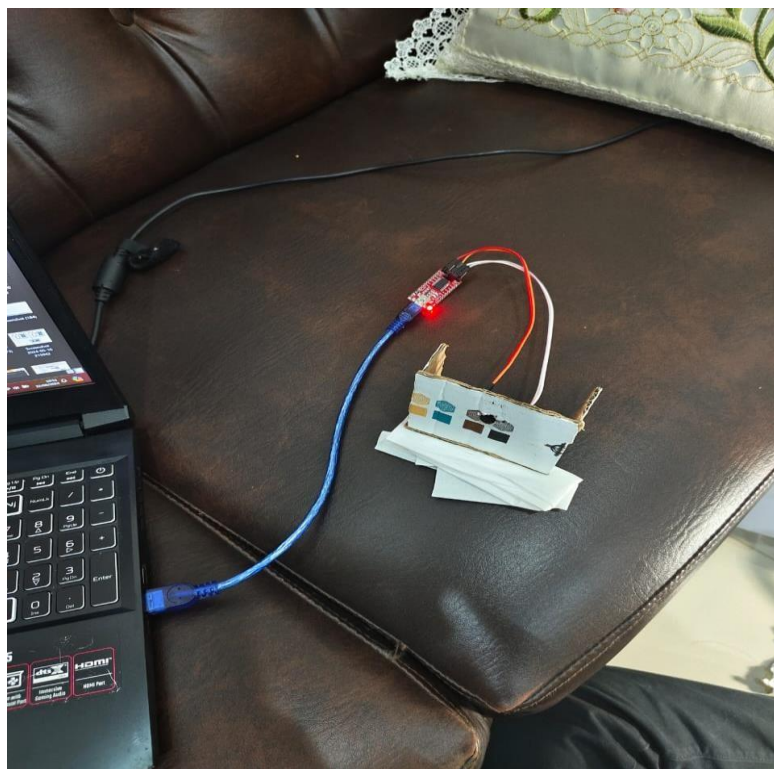
BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada tahap ini, akan dilakukan pengimplementasian sistem setelah di analisis dan dirancang. Sistem akan diimplementasikan menggunakan Visual Studio Code, Arduino IDE, Framework Django, Google Colaboratory, Roboflow, CSS, HTML, Javascript dan *protocol* yang digunakan ialah HTTP (Hypertext Transfer Protocol), HTTP adalah protokol yang digunakan untuk mengirim data antara web browser dan server.

4.1.1 Setup Perangkat IoT (ESP32-CAM)

Pada Gambar 4.1 Dibawah adalah rangkaian untuk memprogram atau memasukan kode kedalam ESP32-CAM, *module* kamera di program menggunakan *software* Arduino IDE dengan menggunakan bahasa pemrograman C.



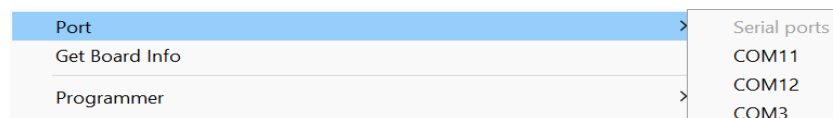
Gambar 4. 1 *Setup* Perangkat ESP32-CAM

Setelah mempersiapkan kamera *module* maka selanjutnya akan menggunakan Arduino IDE dengan menggunakan Bahasa C untuk memasukkan kode kedalam kamera. Arduino IDE telah menyediakan *template* kode untuk memprogram kamera ESP32-CAM, untuk itu ada beberapa hal yang harus disiapkan atau diperhatikan demi keberhasilan untuk memprogram kamera *module*.



Gambar 4. 2 Menu untuk memilih *Board*

Setelah membuka *software* Arduino dan telah memilih *template* untuk kamera *module*, hal yang harus diperhatikan adalah untuk *setting board* pada menu *tools* di Arduino seperti pada Gambar 4.2 *board* sudah diganti menggunakan "AI Thinker ESP32-CAM".



Gambar 4. 3 Menu untuk memilih Serial ports

Kemudian memilih serial ports sesuai dengan yang sudah ter *install* pada komputer seperti pada Gambar 4.3.

```
// =====
// Enter your WiFi credentials
// =====
const char *ssid = "INTERGRITAS 1";
const char *password = "Ramadhan2020april";
```

Gambar 4. 4 Kode *setting WiFi*

Selanjutnya memasukkan *ssid* dan *password WiFi* pribadi sesuai dengan yang terpasang pada komputer seperti pada Gambar 4.4., lalu setelah persyaratan dilaksanakan kode siap di run dan jika berhasil maka output IP kamera akan tertulis di serial monitor dan siap untuk digunakan seperti pada Gambar 4.5.

```

.....
WiFi connected
Camera Ready! Use 'http://192.168.100.134' to connect

```

Gambar 4. 5 Output pada serial monitor

Setelah IP kamera telah didapatkan, selanjutnya adalah integrasi kamera IoT dengan server Django melalui visual studio sehingga kamera bisa mengirimkan video streaming kepada server untuk ditampilkan di web antarmuka, dengan cara *inputkan* IP kamera pada fungsi VideoCapture ('http://alamat_ip_esp32cam:port/stream'), 81 adalah port default yang digunakan oleh ESP32-CAM dan stream adalah fungsi untuk melakukan *streaming* video, ditunjukkan pada Gambar 4.6.

```

# Stream function for video capture and processing
def stream():
    cap = cv2.VideoCapture("http://192.168.100.134:81/stream")

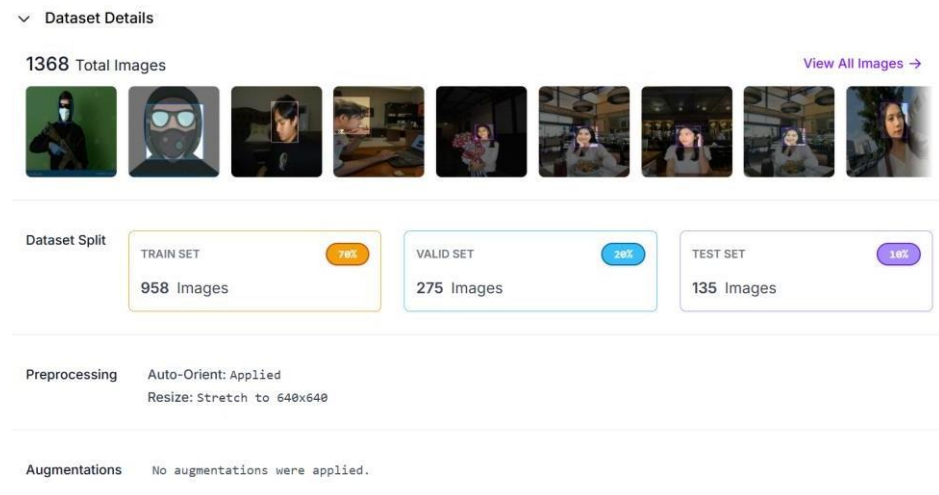
```

Gambar 4. 6 Inisiasi Kamera

4.1.2 Training Dataset Menggunakan Algoritma *You Only Look Once* (YOLO)

Berikut ini adalah tahapan untuk mendapatkan model YOLO, Pembangunan sistem pertama kali adalah melakukan *custom dataset*, *custom dataset* dibuat menggunakan bantuan *Roboflow* untuk melakukan proses *labeling*, didapatkan *dataset* sebanyak 1.368 gambar dengan pembagian 958 gambar sebagai data pelatihan, 275 gambar data validasi dan 135 gambar sebagai data tes.

ditunjukkan pada Gambar 4.7.



Gambar 4. 7 *Custom Dataset*

Setelah *dataset* selesai di *custom* selanjutnya adalah *Training dataset* menggunakan *google colab*, seperti pada Gambar 4.8 lakukan *clone* reposit untuk mendapatkan library YOLO V5 dan diletakan di folder *yolov5*

```
!git clone https://github.com/ultralytics/yolov5 # clone repo
%cd yolov5
```

Gambar 4. 8 *Clone Library YOLO V5*

Pada Gambar 4.9 Kode digunakan untuk mempersiapkan lingkungan deep learning dengan PyTorch dan memeriksa apakah GPU tersedia untuk mempercepat komputasi. Pertama, dependensi yang diperlukan diinstal dari file *requirements.txt* menggunakan perintah *pip*, kemudian beberapa pustaka diimpor, seperti PyTorch untuk keperluan deep learning, serta fungsi dari IPython untuk menampilkan gambar dan membersihkan output pada notebook. Setelah itu, kode memeriksa ketersediaan GPU (CUDA) untuk mempercepat proses pelatihan, dan menampilkan informasi tentang GPU jika tersedia. Jika tidak, CPU akan digunakan,

```
!pip install -qr requirements.txt # install dependencies (ignore errors)
import torch

from IPython.display import Image, clear_output # to display images
from utils.downloads import attempt_download # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))
```

Gambar 4. 9 *Install Requirements Yolo*

Selanjutnya install library Roboflow. Lalu impor kelas roboflow yang didapat sebelumnya untuk mengakses kunci API yang diberikan untuk mengambil dan mengelola proyek dataset yang telah di *custom* untuk di proses di Google Colab, ditunjukkan oleh Gambar 4.10.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="JBkvuCM4LzDrNhSH8AYP")
project = rf.workspace("datasetskripsi-wetq5").project("skripsidataset-hbdbbr")
version = project.version(5)
dataset = version.download("yolov5")
```

Gambar 4. 10 *Import Roboflow dan Dataset*

Kode berfungsi untuk membaca file konfigurasi YAML yang berisi informasi tentang dataset, pustaka yaml diimpor untuk memarsing file YAML, ditunjukkan pada Gambar 4.11.

```
import yaml
with open(dataset.location + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

Gambar 4. 11 Konfigurasi file YAML

Kode ini adalah template untuk file konfigurasi model YOLOv5 yang disimpan dalam file *custom_yolov5s.yaml* untuk mendefinisikan arsitektur model yang akan digunakan untuk mendeteksi objek dengan YOLOv5 pada dataset kustom, ditunjukkan pada Gambar 4.12.

```

%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes} # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]], # 0-P1/2
  [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
  [-1, 3, BottleneckCSP, [128]],
  [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
  [-1, 9, BottleneckCSP, [256]],
  [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
  [-1, 9, BottleneckCSP, [512]],
  [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
  [-1, 1, SPP, [1024, [5, 9, 13]]],
  [-1, 3, BottleneckCSP, [1024, False]], # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 6], 1, Concat, [1]], # cat backbone P4
  [-1, 3, BottleneckCSP, [512, False]], # 13

  [-1, 1, Conv, [256, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 4], 1, Concat, [1]], # cat backbone P3
  [-1, 3, BottleneckCSP, [256, False]], # 17 (P3/8-small)

  [-1, 1, Conv, [256, 3, 2]],
  [[-1, 14], 1, Concat, [1]], # cat head P4
  [-1, 3, BottleneckCSP, [512, False]], # 20 (P4/16-medium)

  [-1, 1, Conv, [512, 3, 2]],
  [[-1, 10], 1, Concat, [1]], # cat head P5
  [-1, 3, BottleneckCSP, [1024, False]], # 23 (P5/32-large)

  [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]

```

Gambar 4. 12 Custom File YAML

Lanjut ke proses *training* model YOLO dengan mengakses dataset dengan ukuran batch (jumlah gambar yang diproses secara bersamaan dalam satu iterasi) sebanyak 16, epoch sebanyak 100 kali, img 416 (Menetapkan ukuran gambar input sebesar 416x416 piksel untuk pelatihan), dan menggunakan bobot awal dari model yolov5s.pt (untuk membantu pelatihan model), ditunjukkan pada Gambar 4.13.

```
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml
--weights 'yolov5s.pt' --name yolov5s_results --cache
```

Gambar 4. 13 Pelatihan Model

Epoch ke 100 didapatkan model dengan nilai mAP50 sebesar 0.947 serta model summary yang terlihat pada setiap kelasnya. Selama epoch didapatkan bahwa loss menurun seiring berjalannya epoch, dan metrik evaluasi seperti precision, recall, dan mAP meningkat, menunjukkan bahwa model semakin baik dalam mendeteksi objek pada dataset yang diuji, seperti ditunjukkan pada Gambar 4.14.

```
100 epochs completed in 0.280 hours.
Optimizer stripped from runs/train/yolov5s_results2/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/yolov5s_results2/weights/best.pt, 14.8MB

Validating runs/train/yolov5s_results2/weights/best.pt...
Fusing layers...
custom_YOLOv5s summary: 182 layers, 7268094 parameters, 0 gradients

```

| Class | Images | Instances | P | R | mAP50 | mAP50-95 |
|--------------------|--------|-----------|-------|-------|-------|----------|
| all | 275 | 289 | 0.944 | 0.89 | 0.947 | 0.689 |
| Bagus | 275 | 42 | 0.79 | 0.905 | 0.918 | 0.662 |
| Dito | 275 | 26 | 1 | 0.805 | 0.987 | 0.781 |
| Dustin | 275 | 28 | 0.971 | 0.857 | 0.883 | 0.673 |
| Erlin | 275 | 45 | 0.962 | 0.867 | 0.986 | 0.629 |
| Fatih | 275 | 36 | 0.985 | 0.861 | 0.925 | 0.672 |
| Nabil | 275 | 27 | 0.985 | 0.863 | 0.967 | 0.773 |
| Orang-mencurigakan | 275 | 23 | 1 | 0.924 | 0.99 | 0.655 |
| Rizki | 275 | 34 | 0.987 | 0.971 | 0.975 | 0.803 |
| Said | 275 | 28 | 0.82 | 0.857 | 0.891 | 0.551 |

```
Results saved to runs/train/yolov5s_results2
CPU times: user 8.92 s, sys: 1.26 s, total: 10.2 s
Wall time: 12min 28s
```

Gambar 4. 14 Hasil Pelatihan model

Didapatkan uji hasil Pelatihan model dengan menggunakan algoritma YOLO dapat mendeteksi dengan benar ditunjukkan oleh Gambar 4.15.



Gambar 4. 15 Hasil *Training* Deteksi YOLO

Gambar 4.16 dibawah menunjukkan grafik garis dari yang didapat dan diukur selama proses pelatihan model YOLOv5. Grafik dibagi menjadi dua bagian yaitu val (validasi) dan train (pelatihan) Bagian train menunjukkan performa model selama proses pelatihan. Metrik yang ditampilkan adalah :

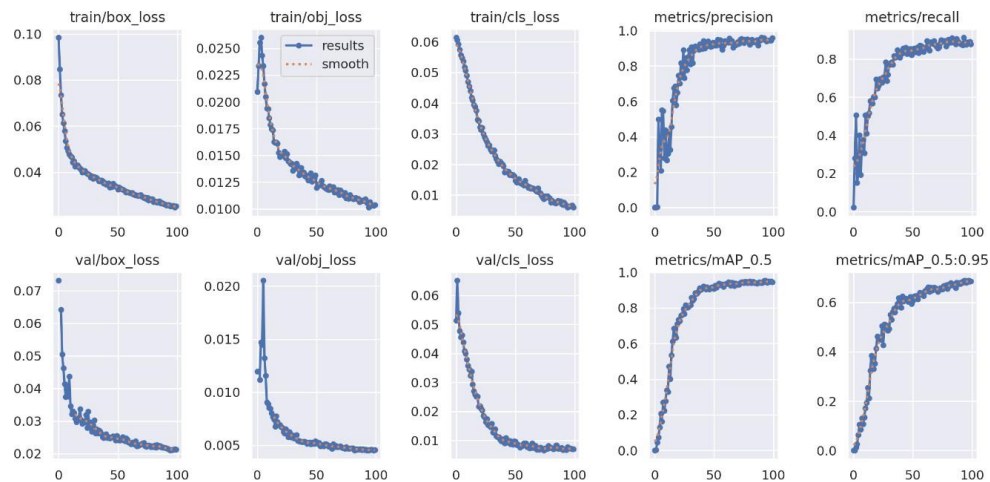
train/box_loss, *train/obj_loss*, *train/cls_loss* ini merupakan tiga jenis fungsi loss yang sering digunakan dalam model deteksi objek.

- *Box_loss*: Menilai seberapa tepat model dalam memprediksi *bounding box* (kotak pembatas) untuk objek.
- *Obj_loss*: Mengevaluasi seberapa baik model dalam mendeteksi keberadaan objek pada suatu gambar.
- *Cls_loss*: Mengukur ketepatan model dalam mengklasifikasikan objek.

metrics/precision, *metrics/recall* kedua metrik ini digunakan untuk melakukan evaluasi kinerja model.

- *Precision* : Mengukur persentase prediksi positif yang benar.
- *Recall* : Mengukur persentase contoh positif yang benar-benar terdeteksi sebagai positif.

Dan *val/box_loss*, *val/obj_loss*, *val/cls_loss*, *metrics/mAP0.5*, ini adalah metrik yang sama, tetapi diterapkan pada data validasi.



Gambar 4. 16 Grafik pelatihan Model

Dapat dilihat pada Gambar 4.16 terdapat sumbu X dan Y, sumbu X adalah jumlah epoch atau iterasi pelatihan epoch adalah satu kali melewati seluruh dataset pelatihan. Semakin besar nilai sumbu X, semakin lama model dilatih dan sumbu Y Menggambarkan nilai metrik tertentu. metrik adalah ukuran kinerja model.

Dilihat dari grafik diatas nilai loss menurun seiring dengan bertambahnya epoch, menunjukkan model semakin baik dalam mempelajari data pelatihan dan nilai metrik (precision, recall, mAP) meningkat seiring dengan bertambahnya epoch, menunjukkan kinerja model yang semakin baik.

4.1.3 Inisiasi Algoritma *DeepSORT*

Inisiasi algoritma *DeepSORT* dilakukan menggunakan aplikasi visual studio berikut Langkah – Langkah pengerjaanya :

```
from deep_sort.utils.parser import get_config
from deep_sort.deep_sort import DeepSort
```

Gambar 4. 17 Library *DeepSORT*

Langkah pertama dalam proses ini adalah mengimpor library algoritma *DeepSORT* yang sebelumnya telah diinstall, berdasarkan Gambar 4.18 itu adalah kode yang diperlukan yaitu 'get_config' dari 'deep_sort.utils.parser' dan kelas 'DeepSort' dari 'deep_sort.deep_sort'. Dengan mengimpor komponen-komponen tersebut berfungsi untuk mengakses konfigurasi yang relevan dan memanfaatkan fungsionalitas yang disediakan oleh kelas *DeepSORT* untuk pelacakan objek.


```
cfg = get_config()
cfg.merge_from_file("deep_sort/configs/deep_sort.yaml")
```

Gambar 4. 18 Konfigurasi data YAML

Sebelum menginisialisasi objek *DeepSORT* seperti yang ditunjukkan pada Gambar 4.19, langkah selanjutnya adalah membaca konfigurasi yang diperlukan dari file YAML. Proses ini dilakukan dengan memanggil fungsi `get_config()`, yang memungkinkan penggabungan pengaturan dari file konfigurasi `deep_sort.yaml`. Dengan cara ini, semua parameter yang diperlukan untuk pengaturan *DeepSORT* dapat diatur secara tepat sebelum proses pelacakan objek dimulai.

```
deepsort = DeepSort('osnet_x0_25',
                    max_dist=cfg.DEEPSORT.MAX_DIST,
                    max_iou_distance=cfg.DEEPSORT.MAX_IOU_DISTANCE,
                    max_age=cfg.DEEPSORT.MAX_AGE,
                    n_init=cfg.DEEPSORT.N_INIT,
                    nn_budget=cfg.DEEPSORT.NN_BUDGET, use_cuda=True)
```

Gambar 4. 19 Inisiasi *DeepSORT*

Setelah konfigurasi telah dibaca, maka *DeepSORT* dapat diinisiasi berdasarkan parameter yang ingin ditetapkan dalam konfigurasi. Beberapa parameter yang dibutuhkan adalah :

- `max_dist` : Menentukan batas jarak maksimum untuk mencocokkan deteksi yang ada. Jika jarak antara deteksi dan pelacakan melebihi nilai ini, objek tersebut tidak akan diakui sebagai yang sama.
- `max_iou_distance` : Menetapkan ambang batas maksimum untuk Intersection over Union (IoU), yang berfungsi untuk menilai kesamaan antara dua bounding box. Apabila nilai IoU melebihi batas ini, deteksi tersebut akan dianggap sebagai objek yang sama.
- `max_age` : Ini adalah jumlah maksimum frame yang dapat dilalui tanpa adanya pembaruan untuk objek yang sedang dilacak, sebelum objek tersebut dihapus dari sistem pelacakan.
- `n_init` : Parameter ini menunjukkan jumlah deteksi yang diperlukan untuk memulai pelacakan objek. Dengan kata lain, sebelum suatu objek

dapat dilacak, harus ada sejumlah deteksi berurutan yang menegaskan keberadaannya.

- `nn_budget` : Ini adalah batas maksimum anggaran yang dapat digunakan untuk menyimpan fitur objek di dalam memori. Parameter ini berfungsi untuk mengelola penggunaan memori dan meningkatkan efisiensi saat melacak banyak objek secara bersamaan.
- `use_cuda` : Parameter ini menunjukkan apakah algoritma akan menggunakan GPU (CUDA) untuk meningkatkan kecepatan proses. Jika diatur ke `True`, DeepSORT akan memanfaatkan CUDA untuk melakukan pemrosesan yang lebih efisien

Langkah selanjutnya seperti pada Gambar 4.20 dibawah mendeklarasikan fungsi `'count_obj'` bertugas melacak pergerakan objek dan menentukan apakah objek tersebut sedang "masuk" atau "keluar" berdasarkan posisinya terhadap garis vertikal yang ada di tengah tampilan video. Fungsi ini menerima informasi berupa titik koordinat objek (berada di bagian atas paling kiri) , ukuran frame (lebar dan tinggi), serta jenis objek yang sedang dipantau. Untuk memantau objek, fungsi ini menggunakan dua kamus: `'track_dict'` untuk mencatat posisi terakhir objek, dan `'direction_dict'` untuk menentukan arah pergerakannya (ke kanan atau ke kiri).

Kode dirancang untuk melacak pergerakan objek dalam sebuah frame atau video setelah proses deteksi objek selesai dilakukan. Kode menganalisis posisi objek dari frame ke frame untuk menentukan apakah objek tersebut melintasi garis vertikal tengah frame dan arah pergerakannya., informasi ini disimpan dalam log dengan waktu kejadian yang tercatat, dan juga ditampilkan di layar.

```

def count_obj(box, w, h, obj_class):
    global object_count, track_dict, direction_dict, logs
    centre_x = int(box[0])
    centre_y = int(box[1])
    centre_coordinates = (centre_x, centre_y)

    # Perhitungan posisi garis vertikal berada di tengah frame
    vertical_line_position = w // 2
    tolerance = 10 # Toleransi dalam piksel

    if obj_class in track_dict:
        prev_position = track_dict[obj_class]
        direction = direction_dict[obj_class]

        # Memeriksa apakah objek telah melewati garis vertikal dengan toleransi
        if prev_position[0] < vertical_line_position - tolerance and centre_x >= vertical_line_position + tolerance:
            entering(obj_class, centre_x) # Panggil fungsi untuk menangani entering
        elif prev_position[0] > vertical_line_position + tolerance and centre_x <= vertical_line_position - tolerance:
            leaving(obj_class, centre_x) # Panggil fungsi untuk menangani leaving

        # Menentukan arah geraknya
        if centre_x > prev_position[0]:
            direction_dict[obj_class] = "right"
        else:
            direction_dict[obj_class] = "left"
    else:
        direction_dict[obj_class] = "right" if centre_x >= vertical_line_position else "left"

    track_dict[obj_class] = centre_coordinates # Update posisi terakhir

```

Gambar 4. 20 Deklarasi Fungsi Pelacakan

Selanjutnya pada Gambar 4.21 adalah kode Kode Python yang diuraikan memiliki dua fungsi utama, yaitu *entering* dan *leaving*, yang berperan dalam mencatat pergerakan objek yang masuk atau keluar dari suatu area. Fungsi *entering* menghitung dan mencatat objek yang masuk, menyimpan data terkait objek dan waktu ke dalam database, serta menampilkan pesan konfirmasi di konsol. Sementara itu, fungsi *leaving* bekerja serupa, namun fokus pada pencatatan objek yang keluar. Kedua fungsi ini sangat penting dalam sistem pelacakan objek untuk mengumpulkan dan menyimpan data, serta membantu analisis lebih lanjut terkait aktivitas objek di area tertentu.

```
def entering(obj_class, centre_x):
    global object_count
    object_count["entering"] += 1
    waktu = datetime.now()

    # Simpan data ke database
    DeteksiObjek.objects.create(nama_kelas=obj_class, aksi="Entering", waktu=waktu)

    print(f"{obj_class} Entering at {waktu}")

def leaving(obj_class, centre_x):
    global object_count
    object_count["leaving"] += 1
    waktu = datetime.now()

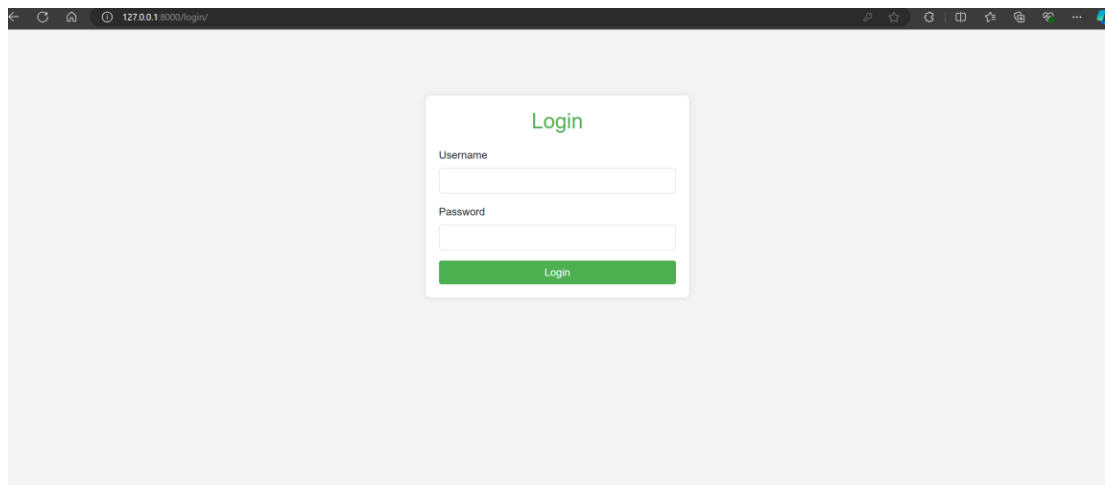
    # Simpan data ke database
    DeteksiObjek.objects.create(nama_kelas=obj_class, aksi="Leaving", waktu=waktu)

    print(f"{obj_class} Leaving at {waktu}")
```

Gambar 4. 21 Deklarasi *Entering* dan *Leaving*

4.1.4 Halaman Login

Setelah memiliki akun, pengguna dapat masuk ke sistem melalui halaman login yang sama untuk semua pengguna. Pada halaman tersebut, pengguna cukup memasukkan username dan password mereka. Jika username atau password yang dimasukkan tidak benar, pengguna tidak akan dapat melanjutkan ke halaman berikutnya dan akan diarahkan kembali ke halaman login, halaman login ditunjukkan pada Gambar 4.22.



Gambar 4. 22 Halaman *Login*

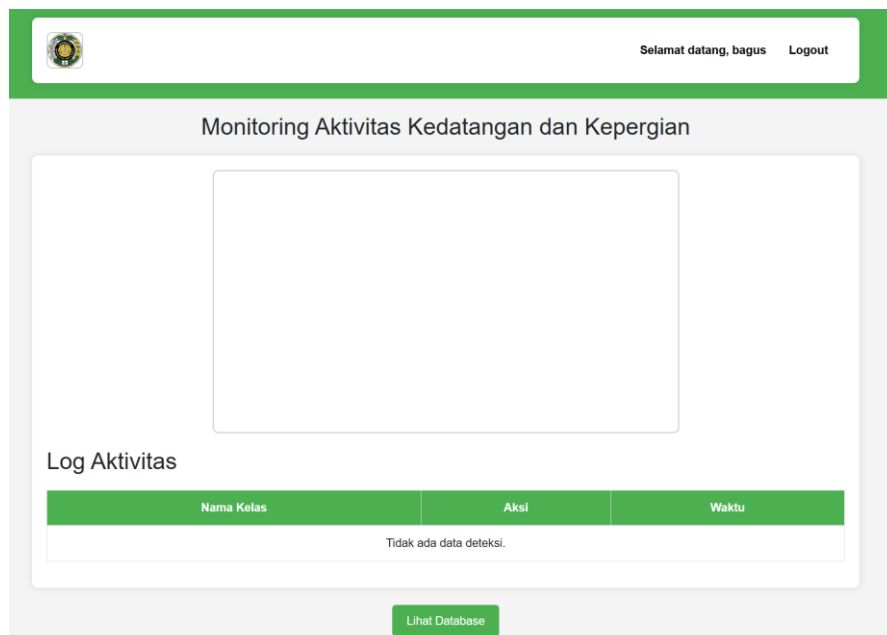
Untuk memiliki akun, pembuatan akun dilakukan langsung di dalam server Django. Seperti ditunjukkan pada Gambar 4.23.

```
Successfully loaded imagenet pretrained weights from "C:\Users\bagus/.cache\torch\checkpoints\osnet_x0_25_imagenet.pth"
Username: uratmangun
Email address: uratmangun@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Gambar 4. 23 Autentikasi akun

4.1.5 Halaman Utama

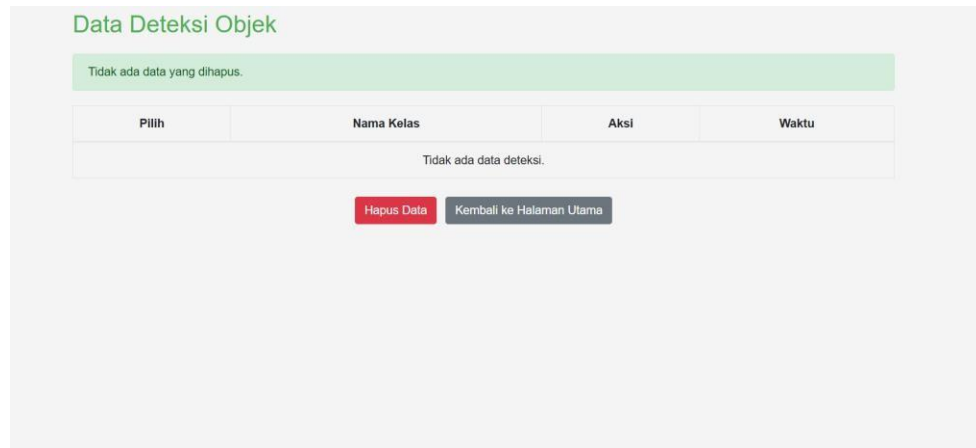
Setelah berhasil autentikasi pengguna dan berhasil login maka user dapat mengakses halaman utama yang seperti pada Gambar 4.24 terlihat kalau kamera belum terkonfigurasi dengan server Django.



Gambar 4. 24 Halaman Utama

4.1.6 Halaman Database

Pada halaman *database* ditunjukkan pada Gambar 4.25 digunakan untuk menampilkan data yang disimpan dalam database. Ini bisa berupa daftar entri, seperti objek yang terdeteksi dalam aplikasi deteksi objek. Pengguna dapat melihat informasi yang relevan, seperti nama kelas, aksi, dan waktu deteksi.



Gambar 4. 25 Halaman *Database*

4.2 Pengujian Fungsional

4.2.1 Pengujian Aliran Video

Dapat dilihat pada Gambar 4.26 server berhasil menerima aliran video dari ESP32-CAM dan menampilkannya pada web antarmuka.

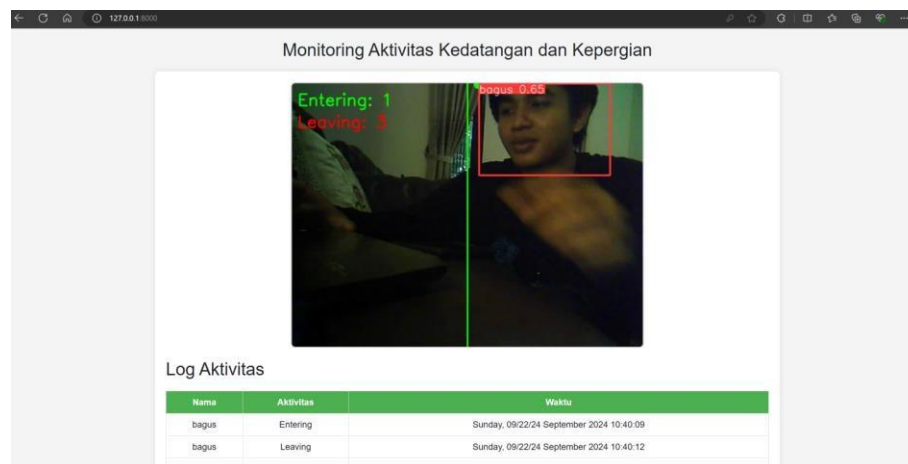


Gambar 4. 26 *Streaming* Aktif

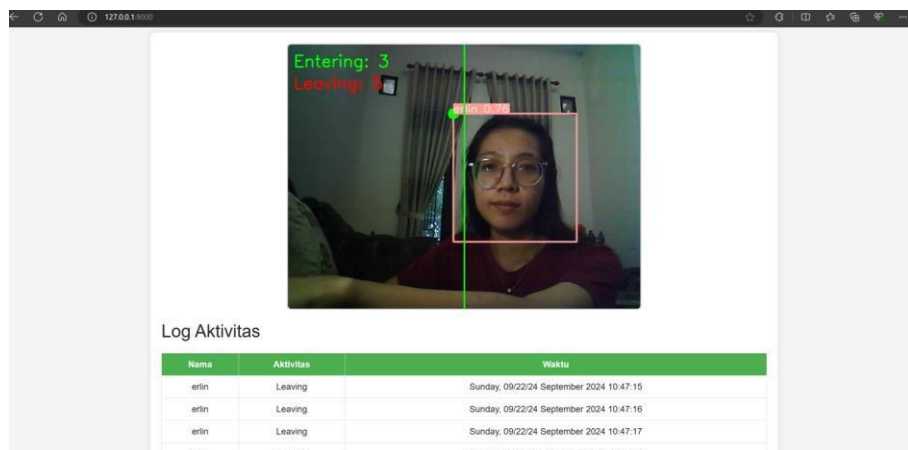
4.2.2 Pengujian Deteksi dan Pelacakan

Pengujian ini untuk melihat apakah objek dapat terdeteksi secara *realtime* pada video *streaming*, dapat dilihat pada Gambar 4.24 dan Gambar 4.25 sistem berhasil mendeteksi objek dengan benar, terdeteksi dalam video yang telah di *capture* sebelumnya kalau objek tersebut adalah "Bagus" dan objek kedua adalah "Erlin" sesuai dengan data yang sudah dilatih sebelumnya.

Dan juga sistem berhasil melacak pergerakan objek, dapat dilihat pada *bounding box* pada objek ada titik disebelah kiri nama objek, seperti yang dijelaskan sebelumnya bahwa titik itu adalah titik kordinat yang fungsi nya sebagai pemicu jika objek bergerak dari kiri ke kanan lalu melewati garis maka objek terindikasi melakukan aktivitas kedatangan dan begitupun sebaliknya jika objek bergerak dari arah kanan ke kiri dan jalan terus melewati garis *vertical* maka objek terindikasi melakukan aktivitas kepergian.



Gambar 4. 27 Hasil Deteksi Bagus



Gambar 4. 28 Hasil Deteksi Erlin

Sistem juga berhasil mendeteksi 2 objek berbeda dalam 1 frame video *streaming* seperti yang ditunjukkan oleh Gambar 4.26.



Gambar 4. 29 Hasil Deteksi 2 orang.

Sistem juga berhasil menyimpan data yang telah ditangkap di halaman utama dan dipindahkan ke halaman *database*, seperti pada Gambar 4.30 dibawah ini.






Data Deteksi Objek

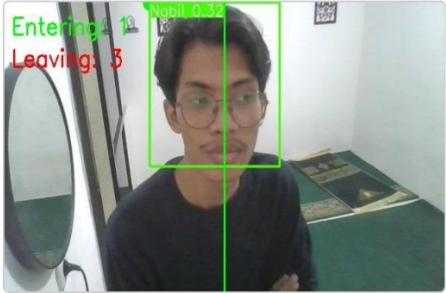

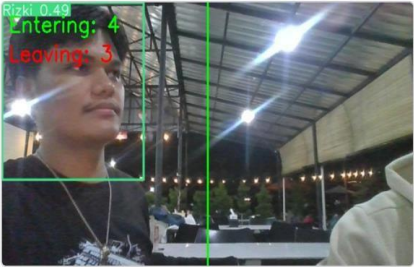

| Pilih | Nama Kelas | Aksi | Waktu |
|--------------------------|------------|----------|---------------------|
| <input type="checkbox"/> | bagus | Entering | 2024-09-23 20:18:47 |
| <input type="checkbox"/> | erlin | Leaving | 2024-09-23 20:18:51 |
| <input type="checkbox"/> | erlin | Entering | 2024-09-23 20:18:55 |
| <input type="checkbox"/> | bagus | Entering | 2024-09-23 20:28:13 |
| <input type="checkbox"/> | bagus | Leaving | 2024-09-23 20:28:20 |
| <input type="checkbox"/> | bagus | Leaving | 2024-09-23 20:28:32 |
| <input type="checkbox"/> | bagus | Entering | 2024-09-23 20:28:39 |
| <input type="checkbox"/> | erlin | Leaving | 2024-09-23 20:28:45 |


Hapus Data Terpilih Kembali ke Halaman Utama

Gambar 4. 30 Hasil Penyimpanan Data

Tabel 4. 1 Hasil Pengujian

| No. | Gambar Wajah | Hasil Deteksi | Keterangan |
|-----|---|--|---|
| 1 |  | <div>Bagus</div> <div>Entering</div> | Sistem berhasil mendeteksi individu sebagai “Bagus” dan sesuai arah gerak nya individu terdeteksi “Entering”. |
| 2 |  | <div>Erlin</div> <div>Leaving</div> | Sistem berhasil mendeteksi individu sebagai “Erlin” dan sesuai arah gerak nya individu terdeteksi “Leaving”. |
| 3 |  |  | Sistem berhasil mendeteksi individu sebagai “Intruder” dan berhasil memberikan peringatan. |
| 4 |  | <div>Fatih</div> <div>Leaving</div> | Sistem berhasil mendeteksi individu sebagai “Fatih” dan sesuai arah gerak nya individu terdeteksi “Leaving”. |

| | | | |
|---|---|------------------------------|---|
| 5 |  | <p>Nabil</p> <p>Entering</p> | Sistem berhasil mendeteksi individu sebagai “Nabil” dan sesuai arah gerak nya individu terdeteksi “Entering”. |
| 6 |  | <p>Bagus</p> <p>Entering</p> | Sistem gagal mendeteksi individu “Dustin” dan terdeteksi sebagai “Bagus”. |
| 7 |  | <p>Rizki</p> <p>Entering</p> | Sistem berhasil mendeteksi individu sebagai “Rizki” dan sesuai arah gerak nya individu terdeteksi “Entering”. |
| 8 |  | <p>Dito</p> <p>Leaving</p> | Sistem berhasil mendeteksi individu sebagai “Dito” dan sesuai arah gerak nya individu terdeteksi “Leaving”. |

| | | | |
|---|---|--|--|
| 9 |  | <div data-bbox="863 264 1141 365">Said</div> <div data-bbox="874 398 1129 510">Leaving</div> | <p>Sistem berhasil mendeteksi individu sebagai “Said” dan sesuai arah gerak nya individu terdeteksi “Leaving”.</p> |
|---|---|--|--|

Hasil pengujian dari 9 kelas individu terdapat 8 hasil yang benar dan 1 salah. Dikarenakan kurangnya variasi gambar pada dataset sehingga hasil deteksi tidak akurat. Sehingga didapat hasil akurasi jumlah benar/jumlah seluruhnya = $8/9 = 88,8\%$.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan tahap implementasi yang dilakukan pada penelitian Rancang Bangun Sistem Monitoring Aktivitas Kedatangan dan Kepergian Menggunakan Algoritma YOLO V5 dan DeepSORT Berbasis Internet of Things (IoT), disimpulkan bahwa :

1. Penelitian ini berhasil merancang dan mengimplementasikan sistem monitoring aktivitas kedatangan dan kepergian pada ruang data center menggunakan teknologi Internet of Things (IoT) dan kecerdasan buatan (AI).
2. ESP32-CAM dapat digunakan sebagai perangkat IoT untuk menangkap video streaming, yang kemudian diproses oleh model AI untuk mendeteksi dan melacak pergerakan objek.
3. Algoritma YOLOv5 dan *DeepSORT* dapat digunakan untuk mendeteksi dan melacak objek dalam video streaming secara real-time.
4. Hasil akhir pengujian sistem dalam mendeteksi individu mendapatkan akurasi 88,8%.

5.2 Saran

Setelah melakukan penelitian ini, ada beberapa saran yang bisa dijadikan pertimbangan untuk mengembangkan penelitian ini, yaitu :

1. Disarankan menggunakan algoritma YOLO paling terbaru.
2. Disarankan menggunakan perangkat kamera yang lebih bagus.
3. Sebaiknya dilakukan pengujian pada lebih banyak skenario nyata, seperti ruang data center dengan kondisi pencahayaan dan objek yang berbeda-beda, untuk memastikan keandalan sistem dalam berbagai situasi.
4. Untuk meningkatkan keamanan di ruang data center, sistem monitoring yang menggunakan YOLOv5 dan DeepSORT dapat dihubungkan dengan sistem keamanan lainnya, seperti kontrol akses fisik (sistem kunci pintu) atau alarm keamanan.

5. Pemanfaatan teknologi cloud untuk penyimpanan data aktivitas dan pelaporan, hal ini akan memberikan skalabilitas dan keamanan lebih baik.

- Rochmawati, N., Hidayati, H. B., Yamasari, Y., Peni, H., Tjahyaningtijas, A., Yustanti, W., & Prihanto, A. (n.d.). *Analisa Learning rate dan Batch size Pada Klasifikasi Covid Menggunakan Deep learning dengan Optimizer Adam*.
- Salikhov, R. B., Abdrakhmanov, V. K., & Safargalin, I. N. (2021). Internet of things (IoT) security alarms on ESP32-CAM. *Journal of Physics: Conference Series*, 2096(1). <https://doi.org/10.1088/1742-6596/2096/1/012109>
- Selay, A., Andgha, G. D., Alfarizi, M. A., Izdhihar, M., Wahyudi, B., Falah, M. N., Khaira, M., & Encep, M. (2022). Internet Of Things. In *Karimah Tauhid* (Vol. 1).
- Setya Budi, A. H., Baiquni, M. A., Mulyanti, B., & Nasution, M. F. (2024a). Sistem Deteksi Laju dan Plat Nomor Kendaraan Berbasis Video Rekaman Menggunakan YOLOv5-DeepSORT dan HyperLPR. *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali Dan Elektronika Terapan*, 11(2), 140–149. <https://doi.org/10.34010/telekontran.v11i2.10900>
- Setyawan, R. A., Muttaqin, A., Khulud, H., & Elektro, J. T. (2021). Aplikasi NODEMCU ESP8266 sebagai Pemantau Suhu dan Kelembaban Ruang Data Center. In *Jurnal EECCIS* (Vol. 15, Issue 1). <https://jurnaleeccis.ub.ac.id/>
- Sugandi, A. N., & Hartono, B. (2022). *Prosiding The 13th Industrial Research Workshop and National Seminar Bandung*.
- Swamy, S. N., & Kota, S. R. (2020). An empirical study on system level aspects of Internet of Things (IoT). *IEEE Access*, 8, 188082–188134. <https://doi.org/10.1109/ACCESS.2020.3029847>
- Wibisono Darmawan, C., U A Sompie, S. R., & Kambey, F. D. (n.d.). Mei-Agustus 2020, hal. *Jurnal Teknik Elektro Dan Komputer*, 9(2), 91–100.
- Widiastuti, N. I., & Susanto, R. (n.d.). Kajian Sistem Monitoring Dokumen Akreditasi Teknik Informatika Unikom. In *Majalah Ilmiah UNIKOM* (Vol. 12, Issue 2).
- Widodo, Y. B., Ichsan, A. M., & Sutabri, T. (2020). Perancangan Sistem Smart Home Dengan Konsep Internet Of Things Hybrid Berbasis Protokol Message Queuing Telemetry Transport. *Jurnal Teknologi Informatika Dan Komputer*, 6(2), 123–136. <https://doi.org/10.37012/jtik.v6i2.302>

LAMPIRAN