Analisis Algoritma *Object Detection* dan *Multiple-Object Tracking* Pada Sistem Pelacak Orang Untuk Menghitung Jumlah Pengunjung

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T.)

Disusun oleh:
GABRIEL FERNANDO
3332210047

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SULTAN AGENG TIRTAYASA

2025

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis Skripsi:

Judul	:	Analisis Algoritma *Object Detection* dan *Multiple-Object Tracking* Pada Sistem Pelacak Orang Untuk Menghitung Jumlah Pengunjung
Nama	:	Gabriel Fernando
NIM	:	3332210047
Fakultas	:	Teknik
Jurusan	:	Teknik Elektro

Menyatakan dengan sesungguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggungjawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini.

Cilegon, 5 Juni 2025

GABRIEL FERNANDO

3332210047

LEMBAR PENGESAHAN

Dengan ini ditetapkan bahwa Skripsi berikut:

Judul	:	Analisis Algoritma *Object Detection* dan *Multiple-Object Tracking* Pada Sistem Pelacak Orang Untuk Menghitung Jumlah Pengunjung
Nama	:	Gabriel Fernando
NIM	:	3332210047
Fakultas	:	Teknik
Jurusan	:	Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 4 Juni 2025 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa dan dinyatakan LULUS.

Dewan Penguji		Tanda Tangan
Pembimbing I	: Fadil Muhammad, S.T., M.T.	
Pembimbing II	: Masjudin, S.T., M.Eng.	
Penguji I	: Dr. Irma Saraswati, S.Si., M.T.	
Penguji II	: H. Alief Maulana, S.T., M.T.	

Mengetahui,
Ketua Jurusan

Dr. Eng. Rocky Alfanz, S.T., M.Sc.
NIP. 198103282010121001

PRAKATA

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Sultan Ageng Tirtayasa. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Dr. Eng. Rocky Alfanz, S.T., M.Sc., selaku Ketua Jurusan Teknik Elektro Universitas Sultan Ageng Tirtayasa.
2. Fadil Muhammad, S.T., M.T.. selaku dosen pembimbing pertama saya yang selalu bersedia menyediakan pendapat dan waktu untuk mengajari dan membantu proses penyusunan skripsi ini.
3. Masjudin, S.T., M.Eng., selaku dosen pembimbing kedua saya yang juga selalu bersedia meluangkan waktu dan tenaga selama proses penyusunan skripsi ini, serta memberikan banyak sekali arahan yang dapat saya kembangkan untuk dilakukan pada penelitian skripsi saya ini.
4. Orang tua saya yang terkasih karena telah memberikan bantuan dukungan, baik material atau pun moral, kepada saya untuk menyelesaikan skripsi ini.
5. Sahabat-sahabat saya yang setia kepada saya dan selalu menemani saya selama proses penyusunan skripsi ini.

Adapun penulis menyadari bahwa Penulis adalah manusia biasa yang jauh dari kata sempurna, sehingga terdapat kesalahan atau pun kekeliruan pada penelitian ini. Oleh karena keterbatasan ilmu dan adanya kesalahan pada penulisan skripsi ini, Penulis menyampaikan permohonan maaf yang sebesar-besarnya dan sangat berkenan untuk menerima perbaikan dan kritikan yang dapat membangun untuk mengembangkan penelitian skripsi ini.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Cilegon, 23 Juni 2025

Gabriel Fernando

3332210047

ABSTRAK

Gabriel Fernando
Teknik Elektro
Analisis Algoritma *Object Detection* dan *Multiple-Object Tracking* Pada Sistem Pelacak Orang Untuk Menghitung Jumlah Pengunjung

Seiring berkembangnya teknologi, seperti *Artificial Intelligence* dan *Machine Learning*, banyak permasalahan yang dapat diselesaikan secara praktis. *Computer Vision* yang merupakan cabang dari AI berguna pada kasus dengan data visi seperti gambar dan video. Model YOLO (*You Only Looks Once*) adalah model deteksi objek pada CV, dan dapat digabungkan dengan algoritma *Multiple-Object Tracking* untuk dapat melacak objek yang terdeteksi. Pada penelitian ini, dibentuk suatu sistem pelacak dan penghitung pengunjung dengan menggunakan YOLO11n dan BoT-SORT *Re-ID*. Kombinasi keduanya diuji pada 5 video rekaman dan mendapatkan rata-rata *error ID Switch*, *Precision*, *Recall*, dan Skor F1 secara berturut-turut sebesar 21.80%, 100%, 67.34%, dan 76.71%.

Kata Kunci:
Computer Vision, YOLO, Deteksi Objek, BoT-SORT, Tracker.

ABSTRACT

Gabriel Fernando
Electrical Engineering
Analyze of Object Detection and Multiple-Object Tracking Algorithms of People Tracker System to Counting People

Due to the development of technology, especially in Artificial Intelligence and Machine Learning, many problems can be solved easily. Computer Vision as a branch of AI is useful for cases that use data from images and videos. The YOLO (You Only Looks Once) is an object detection model CV and can be combined with Multiple-Object Tracking algorithms for tracking. In this research, a tracking and counting system was built by combining YOLO11n and BoT-SORT Re-ID. This combination was tested on 5 footages and achieved the mean of error ID Switch, Precision, Recall, and F1-Score respectively 21.80%, 100%, 67.34%, and 76.71%.

Keywords:
Artificial Intelligence, Machine Learning, Computer Vision, YOLO, Object Detection, BoT-SORT, MOT, Tracker

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring berkembangnya teknologi, semakin banyak permasalahan yang dapat dengan mudah diselesaikan dan menghasilkan hasil yang lebih baik jika dibandingkan dilakukan oleh manusia. Salah satu bidang yang terbantu karena teknologi adalah sistem deteksi kerumunan (*Crowd Detection System*), yang dapat membantu menghitung jumlah orang dalam suatu *frame* dengan bantuan teknologi, terutama *Artificial Intelligence* (AI) dan *Computer Vision* (CV). *Computer Vision* merupakan suatu sistem yang memungkinkan komputer untuk melihat, mempelajari, serta menghasilkan suatu keputusan berdasarkan algoritma yang dipasang [1], [2], [3], [4]. Algoritma YOLO (*You Only Look Once*) merupakan suatu algoritma yang dibentuk secara khusus untuk penugasan bidang *Computer Vision*, khususnya tugas deteksi objek [5]. Model YOLO juga dapat diterapkan pada sistem pemantauan dengan kamera sehingga dapat mendeteksi objek yang dikehendaki [6]. *Deep* SORT merupakan suatu algoritma pelacakan objek (*object tracking*), *Deep* SORT merupakan pengembangan lebih lanjut dari metode pendahulunya, SORT, yang mengatasi beberapa kelemahan metode sebelumnya dengan cara menempatkan metrik asosiasi dengan metrik yang lebih terinformasi yang menggabungkan informasi pergerakan dan penampilan objek yang terdeteksi [7]. Selain DeepSORT, terdapat juga sistem *Multiple-Object Tracker* lainnya, seperti BoT-SORT dan ByteTrack. Baik BoT-SORT atau pun ByteTrack memakai algoritma *Kalman Filtering* untuk melacak posisi dari objek yang terdeteksi pada *frame* saat tertentu, yang dimana objek terdeteksi didapatkan dari model YOLO akan menjadi *input* bagi MoT untuk data asosiasi [8].

Beberapa tempat, khususnya tempat makan seperti *cafe*, restoran, tempat wisata, masih menghitung jumlah pengunjung secara manual atau bahkan tidak menghitungnya. Menghitung jumlah pengunjung secara manual sangat tidak efektif, karena berbagai permasalahan yang dapat ditimbulkan, seperti kesalahan manusia saat menghitung dan tidak mendapatkan hasil yang sesuai [9]. Dengan perkembangan *Computer Vision*, pemantauan dan pelacakan semakin mudah untuk

diterapkan pada banyak hal, khususnya penghitung jumlah pengunjung suatu tempat apa pun itu. *Computer Vision* dapat digunakan untuk mendeteksi berbagai objek (*Multiple Object Tracking*) untuk mengidentifikasikan setiap objek pada setiap *frame* di sepanjang waktu [10], [11], [12].

Telah dilakukan beberapa penelitian yang bertujuan untuk membuat sistem terbaik untuk mendeteksi dalam melacak manusia berbasis kamera dan *Computer Vision*. Pada referensi [13], digunakan algoritma YOLOv3 dan *Simple Online and Real-time Tracking* (SORT) untuk mendeteksi dan melacak manusia secara real-time. Metode YOLOv3 yang digunakan berhasil mendapatkan nilai mAP sebesar 89.69% dan FPS sebesar 58.31. Pada referensi [14], digunakan metode Kalman Filter sebagai pelacak dan LOI (*Line of Interest*) sebagai penghitung pengunjung toko retail kecil. Metode ini berhasil mendapatkan *error* sebesar 45 yang jauh lebih kecil daripada metode lainnya yang dirujuk pada referensi ini.

Pada referensi [15], digunakan jaringan *Convolutional Neural Network* (CNN) dan *Max-average pooling* pada model untuk membuat sistem penghitung jumlah orang. Pada dataset UCF_CC_50, metode ini mendapatkan MAE sebesar 306.7 dan MSE sebesar 396.3. Pada referensi [16], dilakukan pengujian sistem pelacakan manusia berdasarkan sistem *multi* kamera. Penelitian ini juga mengemukakan metode baru, yakni menggabungkan kedua metode sebelumnya, yakni sistem hibrid antara *centralized* dan *distributed*, yang mendapatkan keunggulan yang lebih jikalau dibandingkan dengan kedua metode masing-masing.

Pada referensi [17], dilakukan penelitian untuk membentuk sistem pengidentifikasian dan pelacakan kelompok, pasangan, ataupun individu-individu pada domain restoran. Penelitian ini menggunakan algoritma YOLOv3 dan DeepSORT untuk membuat sistem pengidentifikasian kelompok ataupun pasangan pada kamera. Hasil pengujian pada penggabungan kedua dataset tersebut meraih nilai *precision*, *recall*, mAP@0.5, dan F1-score secara berturut-turut adalah sebesar 0.739, 0.809, 0.795, dan 0.773. Pada referensi [18], dilakukan sistem pelacakan untuk ruang terbuka, dengan menggunakan algoritma YOLOv7 sebagai *object detector*, BoT-SORT sebagai *object tracking*, dan Person *Re-ID* sebagai *re-identification*. Dengan menggabungkan keseluruhannya, metode ini meraih kecepatan di antara 3.8 FPS hingga 7 FPS.

Pada referensi [19], dilakukan pembuatan sistem MCPT (*Multi Camera People Tracking*) dengan 4 metode, yakni proses deteksi objek, *re-identification*, *spatio-temporal filtering*, dan pembentukan kelompok. Digunakan algoritma YOLOX sebagai *object detector*, Bytetrack sebagai pelacak objek, dan MGN sistem *re-identification*. Secara metrik IDF1, IDP, IDR, sistem yang dibentuk pada metode ini mendapatkan skor secara berturut-turut sebesar 78.91, 78.79, dan 79.03. Pada referensi [20], dibentuk sistem yang dapat menghitung dan mengestimasi kepadatan kerumunan dengan menggunakan metode *Extended* CAN, *floor-field*, dan *penalty-term*. Metode ini mendapatkan nilai RMSE pada dataset CrowdFlow sebesar 158.3, pada dataset FDST sebesar 7.29, pada dataset Venice sebesar 18, pada dataset CityStreet sebesar 23.05, dan pada dataset UCSD sebesar 1.02. Dengan menggabungkan keunggulan dari algoritma deteksi objek (*Object Detection*) seperti model YOLO dan pelacak (*Multiple-Object Tracker*) seperti BoT-SORT ataupun ByteTrack, akan mempermudah perhitungan jumlah pengunjung di suatu tempat keramaian secara otomatis, baik di luar ruangan (*outdoor*) ataupun di dalam ruangan (*indoor*). Hal ini akan mengurangi masalah dan kekeliruan yang terjadi saat menghitung pengunjung secara manual.

1.2. Rumusan Masalah

Berikut adalah beberapa rumusan masalah yang diharapkan dapat diselesaikan pada penelitian ini.

1. Bagaimana membentuk sistem deteksi kerumunan untuk menghitung jumlah pengunjung dengan menggunakan algoritma *Object Detection* dan *Multiple-Object Tracker*?
2. Bagaimana kinerja dan akurasi beberapa seri model YOLO sebagai *detector* objek manusia dalam sistem deteksi dan pelacak pengunjung?
3. Bagaimana kinerja beberapa algoritma *Mulitple Object Tracker* sebagai *tracker* dalam sistem deteksi dan pelacak pengunjung?
4. Bagaimana kinerja dari kombinasi algoritma YOLO dan *Multiple Object Tracker* untuk mendeteksi, melacak, dan menghitung jumlah pengunjung.

1.3. Tujuan Penelitian

Berikut ini adalah beberapa tujuan penelitian pada penelitian ini.

1. Membentuk sistem deteksi kerumunan untuk menghitung jumlah pengunjung dengan menggunakan algoritma *Object Detection* dan *Multiple-Object Tracker*.
2. Membandingkan hasil kinerja dan akurasi beberapa seri model YOLO sebagai *detector* dalam sistem deteksi dan pelacak pengunjung.
3. Membandingkan hasil kinerja dari beberapa algoritma *Multiple Object Tracker* sebagai *tracker* dalam sistem deteksi dan pelacak pengunjung.
4. Menganalisis hasil kinerja dari kombinasi algoritma YOLO dan *Multiple Object Tracker* yang ditentukan berdasarkan akurasi terbaik dari tugas masing-masing algoritma.

1.4. Manfaat Penelitian

Berikut ini adalah beberapa manfaat yang didapatkan dari penelitian ini.

1. Dapat membuat sistem pemantauan otonom yang mampu mendeteksi keberadaan manusia dan menghitung jumlah pengunjung pada suatu pusat perbelanjaan atau tempat lainnya.
2. Memberikan kemudahan dan akurasi yang lebih baik dalam pemantauan pengunjung.
3. Membantu dalam analisis pengunjung secara lebih luas namun lebih mudah karena menggunakan bantuan *Artificial Intelligence* dan *Computer Vision*.
4. Penelitian ini memiliki manfaat dalam pengembangan sistem *Computer Vision*, terutama dalam konteks sistem pemantauan.

1.5. Batasan Masalah

Berikut ini adalah beberapa batasan masalah dari penelitian yang dilakukan.

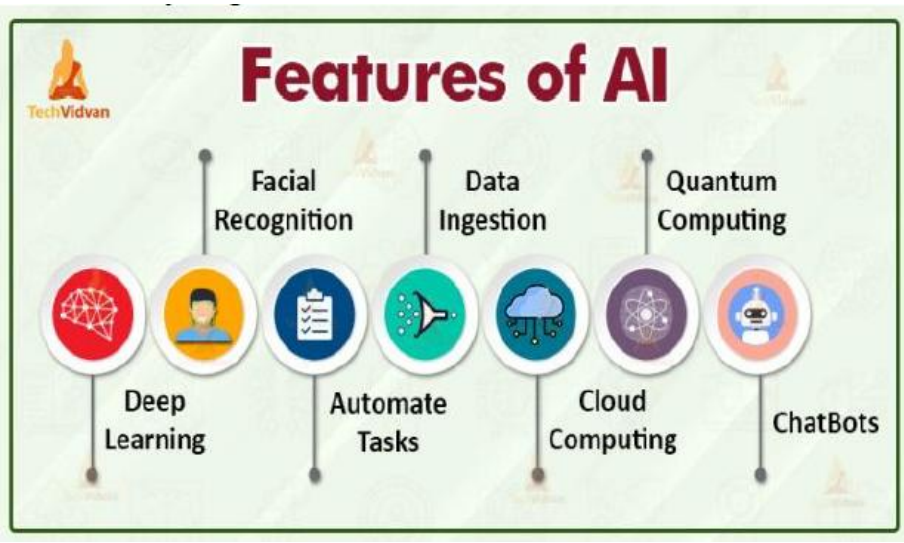
1. Model YOLO yang diuji pada penelitian berikut ini hanya dari versi 8 hingga 11.
2. Algoritma *Multiple Object Tracker* yang dianalisis hanya BoT-SORT dan ByteTrack.

3. Simulasi dilaksanakan menggunakan laptop dari hasil rekaman video menggunakan kamera *smartphone*.
4. Program tidak dijalankan secara real-time dengan integrasi *Mini Computer* dan *Cloud Service*.

BAB II
TINJAUAN PUSTAKA

2.1. *Artificial Intelligence*

Perkembangan teknologi yang semakin pesat di dunia ini menyadarkan banyak orang bahwa saat ini dunia sedang berada pada posisi keempat pada revolusi industri yang di mana teknologi mengaburkan batasan antara bidang fisik, digital, dan biologis [21]. Salah satu teknologi yang sangat berkembang dan berpengaruh pada era saat ini adalah kecerdasan buatan atau *Artificial Intelligent* (AI) [22]. Ai menjadi penggerak revolusi industri 4.0 karena menyediakan banyak kemudahan baik kepada pemerintah maupun industri [23]. Teknologi kecerdasan buatan ini telah menjalar ke berbagai bidang, salah satunya adalah Sistem Informasi [22], [23], [24]. Pemaparan dari fitur-fitur kecerdasan buatan dapat dilihat pada Gambar 2.1.

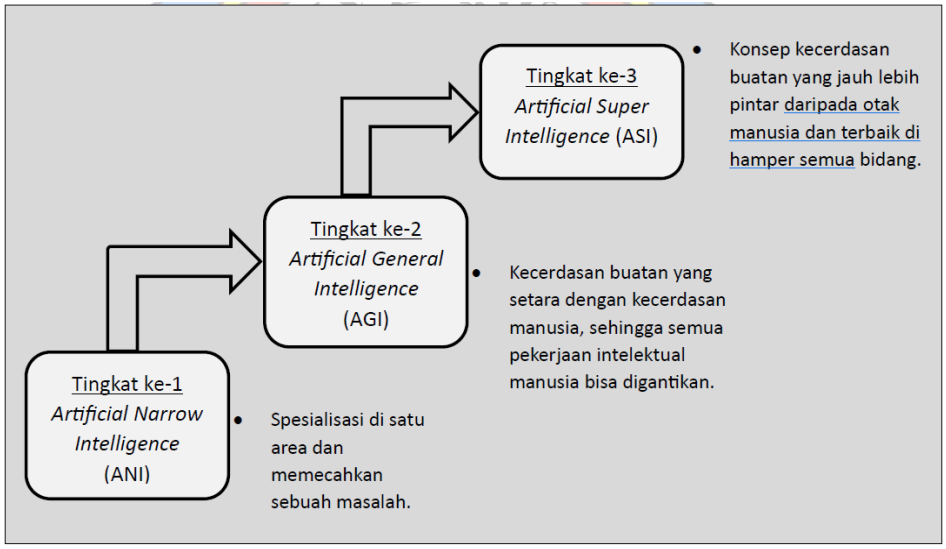


Gambar 2. 1 Tugas-tugas Kecerdasan Buatan

Berdasarkan Gambar 2.1, diketahui bahwa *Deep Learning*, *Facial Recognition*, otomatisasi, penyerapan data, *Cloud Computing*, *Quantum Computing*, dan *Chatbot* adalah contoh-contoh dari fitur yang diberikan oleh kecerdasan buatan. Kecerdasan buatan suatu sistem pada komputer yang dapat menyelesaikan tugas-tugas yang biasanya dibutuhkan kemampuan kecerdasan manusia. Proses yang terdapat pada AI mencakup *learning*, *reasoning*, dan *self-correction*, yang di mana hal tersebut mirip seperti manusia berpikir sebelum mengambil keputusan [21], [25].

Berdasarkan Luger dan Wiliam, kecerdasan buatan adalah suatu cabang dari ilmu komputer yang berhubungan pada sistem otomatisasi perilaku yang cerdas. Sedangkan, menurut Haag dan Peter, kecerdasan buatan adalah suatu bidang studi yang berkaitan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia ke dalam suatu sistem teknologi informasi sehingga sistem tersebut dapat mengambil keputusan-keputusan selayaknya manusia [26]. Dengan kecerdasan buatan, komputer dapat belajar dan beradaptasi dari data yang telah diberikan yang berguna untuk mengambil keputusan secara sendirinya tanpa adanya bantuan atau intervensi dari manusia [22].

Kecerdasan buatan memiliki 3 tingkatan yang berbeda, ketiga tingkatan tersebut dapat dilihat pada Gambar 2.2.



Gambar 2. 2 Tingkatan Kecerdasan Buatan

Berdasarkan Gambar 2.2 diketahui bahwa 3 tingkatan pada kecerdasan buatan, yakni ANI, AGI dan ASI. Berikut ini adalah penjelasan dari masing-masing tingkatan kecerdasan buatan.

a) Tingkatan pertama, *Artificial Narrow Intelligence* (ANI), pada tingkatan ini AI hanya mampu memecahkan suatu permasalahan saja, sehingga sering disebut kecerdasan buatan yang sempit (*Narrow AI*).
b) Tingkatan kedua, *Artificial General Intellegence* (AGI), merupakan kecerdasan buatan ini setara dengan tingkat kecerdasan manusia dan mampu memecahkan berbagai permasalahan yang sebelumnya harus manusia lakukan bisa

digantikan dengan sistem atau robot. Oleh sebab itu, kecerdasan buatan ini sering juga disebut sebagai Human Level *Aetificial Intelligence* (HLAI).

c) Tingkatan ketiga adalah *Artificial Super Intelligence* (ASI), merupakan kecerdasan buatan yang jauh lebih pintar daripada otak manusia. yang membedakan antara kecerdasan buatan ASI dengan kecerdasan manusia terdapat pada jumlah neuron, jika pada manusia terbatas pada beberapa miliar neuron, pada ASI jumlah neuron tidak terbatas [27].

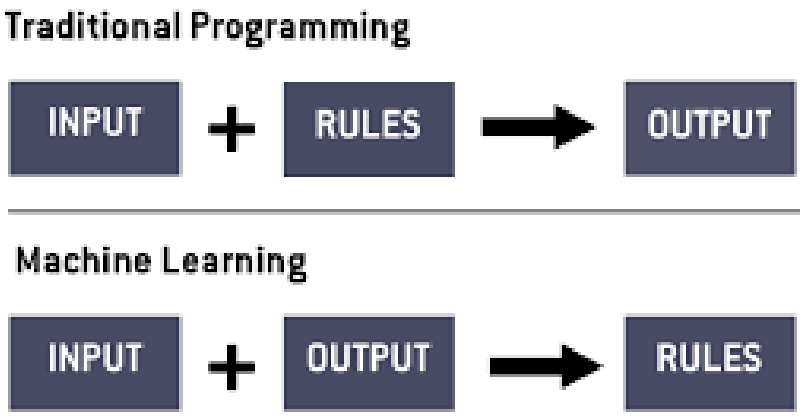
Kecerdasan Buatan telah memasuki berbagai bidang kehidupan, salah satunya adalah pendidikan dan sains [28][29]. Pemanfaatan aplikasi berbasis kecerdasan buatan, seperti ChatGPT, yang merupakan suatu jenis *chatbot* berbasis kecerdasan buatan yang dapat mengetahui *input* dan menghasilkan *output* berupa bahasa naturalnya manusia [30]. Selain keunggulan yang dapat diambil dari ChatGPT dalam proses pembelajaran, terdapat juga dampak negatif yang dihasilkannya, sehingga diperlukan tanggung jawab dan bijaksana dalam memakai aplikasi ini.

2.2. *Machine Learning*

Pembelajaran Mesin atau *Machine Learning* (ML) merupakan suatu metode dalam ilmu komputer yang memungkinkan komputer dapat mempelajari sesuatu berdasarkan data yang diberikan untuk menemukan pola dan menghasilkan keputusan yang diharapkan [31]. *Machine Learning* memiliki kemampuan yang meniru manusia dalam proses pembelajaran manusia dan mampu mengetahui pemahaman yang kompleks sehingga meningkatkan efisiensi komputer [32]. *Machine Learning* dapat melakukan tugas-tugas tertentu berdasarkan data yang diberikan kepada sistem dengan mempelajari algoritma dan model statistik yang ada [33], [34]. *Machine Learning* menjadi suatu fondasi dari banyaknya aplikasi yang penting, seperti *web search*, pengenalan suara, rekomendasi produk, email anti-spam, dan lain sebagainya [35].

Kemampuan *Machine Learning* ini menjadikannya lebih unggul jika dibandingkan dengan pemrograman tradisional. Jika pemrograman tradisional membuat aplikasi berdasarkan data *input* dan serangkaian program, berbeda dengan *Machine Learning* yang membuat suatu program berdasarkan data *input* dan *output*-nya untuk dipelajari oleh mesin. Data yang diberikan kepada mesin memiliki

fitur dan label, setiap fitur-fitur akan dipelajari untuk menghasilkan labelnya untuk setiap fitur. Oleh sebab itu, penggunaan *Machine Learnin*g sangat unggul untuk dipakai jikalau permasalahannya terlalu kompleks bila dilakukan dengan cara pemrograman tradisional. Perbedaan antara pemrograman tradisional dengan *Machine Learning* dapat diilustrasikan pada Gambar 2.3.



Gambar 2. 3 Diagram Perbandingan Pemrograman Tradisional vs *Machine Learning*

Berdasarkan Gambar 2.3, diketahui bahwa pemrograman tradisional membutuhkan peraturan-peraturan, yakni berupa program, agar program tersebut dapat menghasilkan keluaran yang diharapkan berdasarkan *input* yang diberikan. Sedangkan pada konsep *machine learning*, diketahui bahwa program tidak membutuhkan peraturan-peraturan secara spesifik oleh *programmer*, melainkan menaruh *input* dan *output* yang diharapkan pada suatu program, kemudian program tersebut dengan kemampuan pembelajarannya akan membentuk *rules* sendiri yang sungguh kompleks. Sehingga, pada konsep *machine learning*, program akan dapat secara otomatis menghasilkan jawaban berdasarkan seluruh *input* yang sudah dipelajari oleh mesin sebelumnya. Dengan mengambil manfaat dari AI dan *Machine Learning*, banyak aktivitas yang sukar untuk diselesaikan hanya dari sisi *engineering* menjadi lebih mudah untuk diselesaikan [23], [36]. Bidang *Machine Learning* terpopuler untuk saat ini adalah *Predictive Modelling* yang mampu memprediksi hasil berdasarkan tugas yang diberikan, sistem ini sudah dipakai banyak perusahaan besar, seperti Amazon, Alibaba, dan perusahaan raksasa lainnya [36]. Selain itu, pemanfaatan model *Machine Learning* juga dapat diterapkan dalam industri kesehatan seperti pendeteksi dini penyakit kardiovaskuler [37]. Juga dalam

industri pasar model, model *machine learning* dapat dipakai untuk mendeteksi harga saham pada waktu yang akan mendatang [38].

Meskipun banyak kecanggihan yang ditawarkan oleh *Machine Learning*, keamanan dan perlindungan dari sistem juga harus diperhatikan. Hal ini karena adanya serangan dari pihak ketiga yang dapat melawan model ML dan menjadikan model salah dalam mengklasifikasikan *input*, mengurangi akurasi prediksi, dan mengekstraksi data *training* model, bahkan mencuri datanya [31], [39]. Sehingga dibutuhkan suatu studi untuk mencegah ini, yakni *Adversarial Machine Learning* (AML) yang berfokus pada eksploitasi kerentanan ML dan mengembangkan mekanisme pertahanan untuk mitigasi eksploitasi kerentanan ML [31], [40]. Oleh sebab itu, dibutuhkan kebijaksanaan dan tanggung jawab dalam menggunakan ataupun membuat sistem *Machine Learning* agar data privasi yang bersifat rahasia tidak terekspos dan dijadikan hal negatif yang berdampak merugikan.

2.2.1. Deep Learning

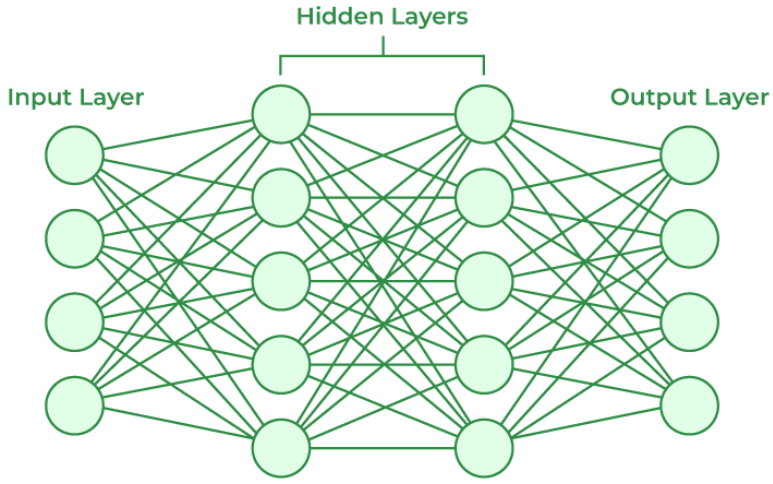
Deep Learning adalah salah satu cabang *Machine Learning* yang menjadi suatu solusi dalam pembelajaran hierarki (*Hierarchical Learning*) yang dapat membuat model mengabstraksikan data masukan secara bertingkat [41]. Model yang menggunakan konsep *deep learning* mampu secara otomatis mempelajari data dan meningkatkan fungsinya dengan menganalisis algoritma. *Deep Learning* sangat efektif karena mampu mempelajari data tak terstruktur (*unstructured data*), seperti gambar, video, audio, bahasa natural manusia, bahkan dataset medis seperti CT-Scan dan MRI [42]. Beberapa contoh dari lapisan *deep learning* adalah *dense layer* untuk menemukan pola data, *convolutional layer* untuk mengekstrak fitur-fitur pada gambar, *recurrent layer* untuk pemrosesan bahasa natural ataupun data kontinu, dan *pooling layer* untuk mereduksi parameter model.

Model *deep learning* menghasilkan akurasi yang lebih besar daripada pembuatan model dengan mengekstrak fitur secara manual, serta dengan menggabungkan beberapa model *deep learning* dengan *dense layer* akan menghasilkan akurasi klasifikasi yang tinggi [43]. Penelitian terkini, seperti dibentuknya dataset besar dan GPU, menjadikan *deep learning* sebagai solusi yang terjangkau untuk kendala *hardware* [41]. Karena *deep learning* saat ini dapat

dengan mudah dibentuk melalui berbagai *framework* seperti TensorFlow dan PyTorch. Selain itu *deep learning* juga dapat dibentuk dengan menggunakan layanan *notebook cloud*, seperti Jupyter *Notebook*, Google Colaboratory, dan Kaggle *Notebook*

2.2.2. Artificial Neural Network

Artificial Neural Network merupakan suatu model *Machine Learning* yang terinspirasi dari jaringan saraf biologis pada makhluk hidup [44]. ANN merupakan inti dari *Deep Learning*, karena kemampuannya yang mampu menyelesaikan tugas kompleks ML, seperti klasifikasi miliaran gambar, penguatan layanan pengenalan suara, dan rekomendasi video. Tugas umum yang dilakukan dengan menggunakan ANN adalah regresi dan klasifikasi [44]. Model pertama sekali dari ANN ditemukan oleh ahli saraf Warren McCulloch dan matematikawan Walter Pitts pada tahun 1943, yang dalam penelitiannya mereka menyimplifikasikan model komputasional dari cara kerja jaringan saraf pada otak binatang untuk melakukan komputasi yang kompleks menggunakan *propositional logic* [44]. Ilustrasi dari model jaringan saraf buatan atau *Artificial Neural Network* (ANN) dapat dilihat seperti pada Gambar 2.4.



Gambar 2. 4 Diagram *Artificial Neural Network* (ANN)

Berdasarkan Gambar 2.4, didapati bahwa terdapat 3 lapisan, yakni *input layer*, *output layer*, dan *hidden layer*. *Input layer* merupakan layer masukkan, sedangkan *output layer* merupakan layer keluaran. Sedangkan, *hidden layer*

merupakan sejumlah lapisan dengan algoritma-algoritma tertentu yang direkayasa berdasarkan kebutuhan dari program. Kemudian model ANN semakin berkembang dan ditemukannya *Perceptron*, *Perceptron* merupakan model paling sederhana dari ANN yang ditemukan oleh Frank Rosenblatt pada tahun 1957.

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{2.1}$$

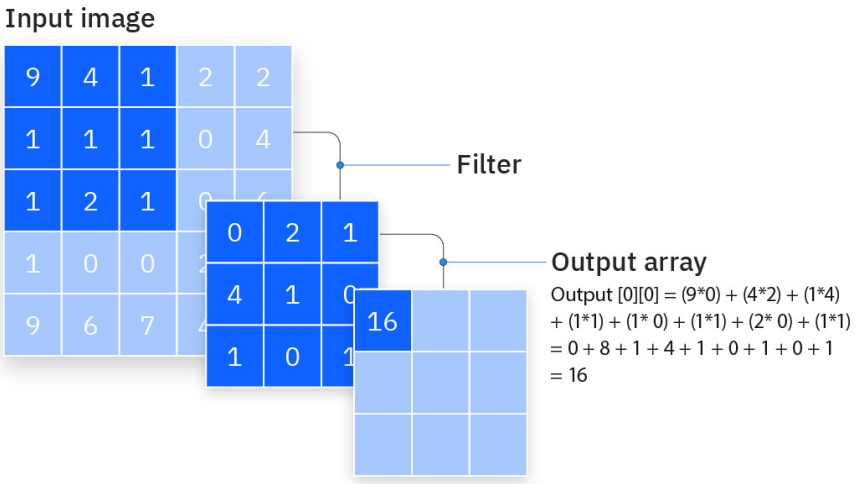
$$h_w(x) = step(z) \tag{2.2}$$

$$Heaviside(z) \begin{cases} 0\ jika\ z < 0 \\ 1\ jika\ z \geq 0 \end{cases} \qquad sgn(z) \begin{cases} -1\ jika\ z < 0 \\ 0\ jika\ z = 0 \\ 1\ jika\ z > 0 \end{cases} \tag{2.3}$$

Semua masukan dan keluaran dari *perceptron* adalah angka, dan setiap koneksi masukan terasosiasi dengan suatu beban (*weight*), kemudian akan dijumlahkan seluruh *input* seperti pada persamaan 2.1. Lalu dilakukan step *function* untuk hasil penjumlahan tersebut dan hasil keluarannya hasil fungsi step dari z seperti pada persamaan 2.2 [44]. Umumnya, fungsi step yang ada pada *perceptron* adalah *Heaviside step function* dan *Sign function* yang dapat dilihat pada persamaan 2.3.

2.2.3. Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan bagian dari *deep learning*, terdiri dari beberapa lapisan simpul. CNN memiliki suatu *input* dan *output*, dan diantaranya terdapat satu atau beberapa lapisan, yang disebut sebagai *hidden layer*. Lapisan konvolusional terdiri dari beberapa komponen atau *hyperparameter*, seperti *filter size*, *stride*, dan *padding*. Filter pada CNN berguna sebagai *feature detector* yang mendeteksi setiap *pixel* pada gambar untuk mendapatkan fitur-fitur pada gambar, proses tersebut dinamakan proses konvolusi. Selama filter mendeteksi *pixel*-per-*pixel* pada gambar, secara bersamaan dilakukan *dot product* sebagai hasil keluarannya. Ilustrasi dari algoritma *Convolutional Neural Network* (CNN) dapat dilihat pada Gambar 2.5.



Gambar 2. 5 Diagram *Convolutional Neural Network* (CNN)

Berdasarkan Gambar 2.5, didapati suatu contoh *input* gambar dengan nilai matriks 5x5 dengan angka bulat, Lalu terdapat matriks kedua yang berukuran 3x3 yang merupakan *filter* untuk *input* gambar. Hasil keluaran gambar merupakan hasil *Dot Product* antara matriks pada *input* gambar dengan matriks filter. Ukuran filter umumnya adalah matriks 3 x 3 dengan nilai-nilai khusus setiap elemen pada matriksnya. Filter akan bergerak pada suatu *stride*, dan mengulanginya sampai kepada titik akhir pada gambar dan menghasilkan *output* berupa *feature map*, *activation map*, atau *convolved feature*. Metode CNN secara meluas dimanfaatkan untuk klasifikasi gambar karena mampu menghasilkan performa yang baik pada prediksi, *staging*, dan prognosis dari CRC [42], [45], [46]. Sebagai *classifier*, biasanya model terbentuk dengan lapisan-lapisan konvolusi (*convolutional layers*) dan diakhiri dengan lapisan linier (linier layer). Lapisan konvolusi diawali ini digunakan untuk mengekstraksi fitur-fitur data yang kompleks, kemudian diakhiri dengan reduksi dimensi dan klasifikasi menggunakan lapisan linier [47].

2.3. *Computer Vision*

Visi Komputer (*Computer Vision*) merupakan cabang dari *Machine Learning* yang memungkinkan mesin untuk dapat memahami dan menginterpretasi informasi visual, dan berdasarkan informasi tersebut mesin dapat mengidentifikasi, melacak, dan mengklasifikasi. Misalnya dapat melacak dan mengklasifikasi kecacatan produk dalam industri manufaktur dengan menganalisis gambar atau video [48].

Computer Vision dipakai pada sistem klasifikasi kanker kulit menggunakan model *deep learning*, seperti CNN dan model *pre-trained deep learning* [49]. *Computer Vision* pernah dipakai untuk melawan dan mengontrol pandemik COVID-19, seperti penggunaan *Computed Thermography* (CT), *X-ray Imagery*, *Ultrasound Imaging*, dan pencegahan serta pengontrolan [50]. *Computer vision* dipakai untuk meningkatkan mekanisme *quality control* menggunakan 3 jenis *Machine Learning*, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning* [51]. Beberapa contoh tugas dalam *Computer Vision* adalah *Image Classification* dan *Object Detection* seperti yang dijelaskan di bawah berikut ini

2.3.1. Image Classification

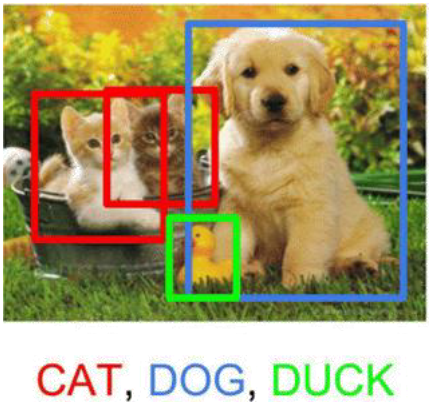
Image Classification merupakan tugas dalam *Computer Vision* yang membuat model dapat mengklasifikasikan kelas pada gambar yang diberikan. Misalnya, model yang diberikan gambar kucing akan memberikan hasil keluaran kucing, seperti yang dapat dilihat pada Gambar 2.6.

Gambar 2. 6 Contoh Klasifikasi Gambar

Berdasarkan Gambar 2.6, suatu model dapat mengklasifikasikan bahwa gambar yang diberikan adalah gambar seekor kucing berdasarkan sistem *Machine Learning* yang dilakukan *training* dengan menggunakan gambar-gambar kucing. Model yang umum dalam *Computer Vision* untuk tugas klasifikasi gambar adalah ResNet dan VGGNet. ResNet adalah suatu model yang memanfaatkan arsitektur CNN dan *dense layer* untuk menemukan pola kompleks pada informasi visual. Sedangkan VGGNet adalah model yang berbasis arsitektur CNN dengan memakai filter dengan matriks ukuran 3 x 3.

2.3.2. Object Detection

Berbeda dengan klasifikasi gambar, tugas deteksi objek pada visi komputer adalah dengan melacak posisi keberadaan suatu objek dengan kelas tertentu. Tugas deteksi objek memungkinkan model untuk mengklasifikasikan lebih dari satu kelas dan memberikan *bounding-box* pada kelas-kelas yang terdeteksi dalam satu gambar. Misalnya, terdapat suatu gambar yang diberikan kepada model, di mana gambar tersebut terdapat 3 kelas objek, yakni anjing, kucing, dan angsa. Model tersebut akan dapat mengklasifikasikan seluruhnya serta memberi *bounding-box* pada lokasi setiap objek yang terdeteksi, seperti yang dapat dilihat pada Gambar 2.7.



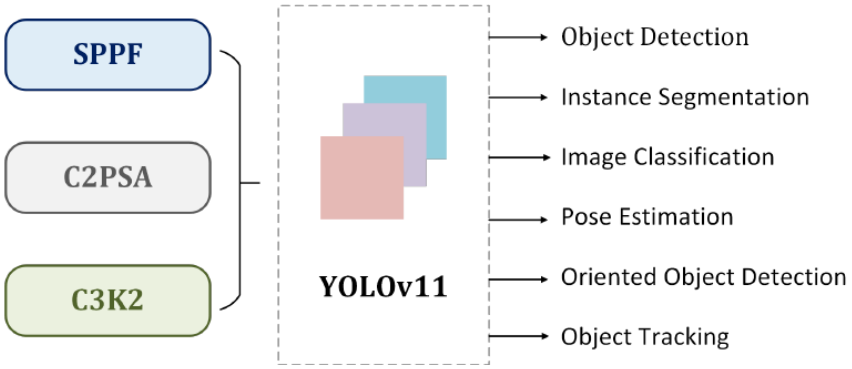
Gambar 2. 7 Contoh Deteksi Objek

Berdasarkan Gambar 2.7, model dapat mendeteksi objek dengan cara mengklasifikasikan objek dan memberikan hasil keluaran berupa *bounding box* pada setiap objek yang berhasil diklasifikasi. Penugasan Deteksi objek berbeda dengan klasifikasi, seperti yang dapat dilihat pada Gambar 2.6, keluaran model tidak memprediksi *bounding box* pada objek yang terdeteksi. Terdapat beberapa model yang diciptakan untuk penugasan untuk deteksi objek, yang berfungsi untuk melakukan klasifikasi dan regresi *bounding box*. Model-model yang umum digunakan pada tugas deteksi objek adalah *Faster* R-CNN, YOLO, dan SSD.

2.4. Algoritma *You Only Look Once*

YOLO merupakan singkatan dari *You Only Look Once* yang adalah suatu model yang menggunakan *convolutional neural network* untuk mendeteksi kelas objek dan *bounding-box* secara bersamaan dari suatu gambar. Model YOLO mengalami perkembangan dari seiring perkembangan teknologi, untuk saat ini terdapat 11 versi model YOLO, dari YOLOv1 hingga YOLOv11. YOLO menjadi suatu model pendeteksi yang bersifat objek *one-stage*, diikuti dengan SSD dan RetinaNet. Hampir dari seluruh seri dari model YOLO dilatih dengan menggunakan dataset Ms COCO [52]. Dataset Ms COCO atau COCO adalah suatu dataset dari Microsoft yang merupakan sekumpulan gambar dalam jumlah yang sangat besar yang digunakan secara umum untuk *object detection*, *captioning*, *segmentation*, dan lain sebagainya.

Seri terbaru dari algoritma YOLO saat ini adalah YOLOv11. YOLO versi ke-11 ini dibentuk dengan pendalaman yang signifikan sebagai teknologi real-time *object detection* dari seri-seri sebelumnya [53]. Gambar ringkasan arsitektur dari YOLOv11 dengan penugasan-penugasan yang dapat dilakukannya dapat dilihat seperti yang tertera pada Gambar 2.8.



Gambar 2. 8 Arsitektur Algoritma YOLOv11

Berdasarkan Gambar 2.8, arsitektur YOLOv11 terdiri dari 3 komponen yakni *Backbone*, *Neck*, dan *Head* [54]. Komponen *Backbone* menjadi komponen utama ekstraksi fitur dan memanfaatkan CNN untuk mengubah data gambar mentah menjadi pemetaan fitur multi-skala (*multi-scale feature maps).* Komponen berikutnya, yakni *Neck*, bekerja sebagai perantara pada tahap pemrosesan, serta menggunakan lapisan-lapisan terspesialisasi (*specialized layers*) untuk melakukan

agregasi dan meningkatkan representasi fitur-fitur pada berbagai skala yang berbeda. Dan terakhir adalah komponen *Head*, bertugas dengan memiliki fungsi sebagai mekanisme prediksi dan melahirkan hasil keluaran final (*final output*) serta klasifikasi pemetaan fitur yang diperhalus (*refined feature maps*). Lapisan *backbone* pada YOLOv11 terdiri dari lapisan konvolusi, SPPF dan C2PSA. Kemudian, pada komponen *neck* terdiri atas blok C3K2 dan *Attention Mechanism*. Lalu pada komponen akhir, *head*, terdiri atas blok C3K2, blok CBS, lapisan-lapisan konvolusi final, dan lapisan deteksi.

2.5. Kajian Pustaka

Pada referensi [13], metode yang digunakan adalah algoritma YOLOv3 sebagai deteksi orang dan algoritma *Simple Online and Real-Time Tracking* (SORT) sebagai pelacak yang membandingkan objek yang terdeteksi pada *frame* sebelumnya, untuk memastikan bahwa perubahan *frame* tidak mengubah *id* dari objek yang terdeteksi di *frame* sebelumnya. Membuat sistem pendeteksi dan pelacak orang secara *real-time* dengan algoritma YOLO. Dari hasil eksperimen yang membandingkan model YOLOv3 dengan Faster R-CNN, YOLO, dan SSD512, model YOLOv3 mendapatkan performa yang jauh lebih unggul dari metode lainnya, yakni mendapatkan nilai mAP sebesar 89.69% dan FPS sebesar 58.31.

Pada referensi [14], metode yang digunakan pada penelitian ini adalah Kalman filter sebagai pelacak dan LOI (*Line of Interest*) sebagai penghitung jumlah pengunjungnya. Metode yang digunakan diterapkan pada *microcomputer* Raspberry Pi yang secara meluas telah digunakan pada sistem-sistem berbasis *real-time image processing*. Membuat sistem yang murah dan mudah digunakan untuk mendapatkan pola pengunjung pada toko retail kecil yang dapat bekerja pada transmisi nirkabel untuk membawakan informasi statistik yang akurat mengenai jumlah orang yang ada pada toko. Penelitian ini membandingkan yang dibentuk dengan metode *Flow Mosaicking* (FM), *Directional People Counter* (DPC), *Counting People Crossing a line* (CPC) yang diuji pada 4 dataset, yakni SU1, SU2, DU1, dan DU2. Secara keseluruhan, metode yang digunakan menghasilkan *error* sebesar 45, jauh lebih kecil dari pada CPC dengan *error* sebesar 83, DPC dengan *error* sebesar 325, dan FM dengan *error* sebesar 154. Dengan berdasarkan performa

rata-rata *Recall* sebesar 0.98, rata-rata *Precision* sebesar 0.96, rata-rata F1 sebesar 0.97, dan rata-rata akurasi sebesar 0.94. Berdasarkan perbandingan yang dilakukan, metode yang digunakan memiliki performa yang lebih unggul daripada metode FM, DPC, dan CPC.

Pada referensi [15], metode yang digunakan adalah fitur *global-density* yang dipasang pada jaringan CNN (*Convolutional Neural Network*). *Max pooling* pada model dasar diubah dengan *max-average pooling* untuk menjaga ketelitian fitur dan lapisan dekonvolusi ditambahkan untuk mengembalikan detail-detail yang hilang akibat proses *down-sampling* sebelumnya. Eksperimen ini dilakukan dengan 2 dataset, yakni UCF_CC_50 dan ShanghaiTech, dengan menggunakan *framework* Pytorch pada konfigurasi *hardware* GPU GeForce GTX 1080Ti dengan VRAM sebesar 11 GB, CPU ES-2630, RAM 32 GB pada sistem operasi Ubuntu 16.04.3. Digunakan fitur *global-density* yang ditangkap dari sub-tugas klasifikasi kepadatan dan *max-average pooling* untuk menjaga fitur-fitur lebih detail. Pada dataset UCF_CC_50, metode yang digunakan pada penelitian ini menghasilkan MAE sebesar 306.7 dan MSE sebesar 396.3, yang dimana jauh lebih unggul dari metode-metode lainnya yang dirujuk. Sedangkan pada dataset Shanghaitech *Part* A, didapatkan MAE dan MSE berturut-turut sebesar 86.6 dan 129.7. Pada dataset Shanghaitech *Part* B mendapatkan MAE dan MSE sebesar 19.3 dan 35.3.

Pada referensi [16], dilakukan analisis sistem pelacak manusia berdasarkan sistem *multi-camera* Dilakukan analisis 2 metode, yakni Tersentralisasi (*Centralized*) dan Terdistribusi (*Distributed*). Pendekatan Tersentralisasi adalah pendekatan yang mendeteksi dan melacak objek, yang dimana hasil dari deteksi setiap kamera difusi untuk melacak objek (*early fusion*). Sedangkan pendekatan terdistribusi adalah pendekatan yang di mana pendeteksian dan pelacakan objek dilakukan pada masing-masing kamera, kemudian hasilnya difusi untuk membentuk lintasan orang-orang (*late fusion*). Referensi ini membuat sistem hibrid dari yang mengelompokkan kamera menjadi kluster-kluster dengan fusi lokal pusat (*local fusion centers*). Metode ini melacak dengan kamera yang sudah diklasterisasi, pelacakan dilakukan dengan menggunakan pendekatan tersentralisasi (*centralized*) dan menggunakan algoritma *re-identification* untuk membentuk trayek dari orang yang terlacak. Selain sebagai pelacak orang, metode

yang dirujuk ini juga dapat digunakan sebagai sistem pengawasan berbasis AI, kontrol kemacetan, penghitung pedestrian, analisis olahraga.

Pada referensi [17], metode yang digunakan pada penelitian ini adalah YOLOv3 yang dilatih ulang agar hanya mendeteksi objek orang saja, bukan objek lain, sebagai pendeteksi objek, DeepSORT sebagai pelacak objek, dan algoritma pengelompokan untuk mengidentifikasi apakah orang-orang yang berdekatan dalam *frame* termasuk dalam kelompok atau tidak. Algoritma pengelompokan menggunakan *bounding-box* dan *id* yang telah didapatkan dari 2 algoritma sebelumnya, yakni YOLOv3 dan DeepSORT. Penelitian ini memberikan suatu solusi berbasis *Computer Vision* yang mengidentifikasikan jikalau suatu jarak antar orang kurang dari atau sama dengan dari batas yang telah ditentukan, maka orang-orang tersebut dinyatakan sebagai suatu kelompok atau pasangan. Namun, apabila orang-orang tersebut melebihi batas jarak yang ditentukan maka tidak dianggap sebagai suatu kelompok, yakni individu saja. *Training* model *retrained* YOLOv3 menggunakan dataset COCO dan juga dataset *custom* pribadi, hasil *training* pada kedua dataset yang digabungkan menghasilkan nilai *precision*, *recall*, mAP@0.5, dan F1 *score* secara berturut-turut adalah 0.739, 0.809, 0.795, 0.773. Sedangkan hasil uji (testing) mendapatkan nilai precision, recall, mAP@0.5, dan F1-*score* secara berturut-turut adalah 0.746, 0.818, 0.801, dan 0.780. Lalu dilakukan evaluasi dari sistem pengidentifikasian suatu kelompok dengan mengambil video dari CCTV dan menghasilkan nilai *precision* sebesar 0.7083, *recall* sebesar 0.7906, dan F1-*score* sebesar 0.7471.

Pada referensi [18], metode yang digunakan pada penelitian ini adalah YOLOv7 sebagai deteksi objek, BoT-SORT sebagai pelacak objek, dan *Person Re-ID* sebagai *re-identification*. Pembuatan sistem *smart campus* dengan 4 lapisan, yakni lapisan fisik, lapisan jaringan, lapisan integrasi, dan lapisan aplikasi. Pada lapisan fisik, diletakkan kamera IP untuk menangkap video dan mengirimkannya ke server untuk pemrosesan. Kemudian lapisan jaringan yang memungkinkan komunikasi nirkabel antara sub sistem sensor dan *middleware*. Lapisan integrasi yang menyediakan *database* temporal, dan lapisan aplikasi yang mengoleksi data untuk analisis dan visualisasi. Penelitian ini membandingkan performa objek deteksi beberapa seri YOLO pada dataset COCO, yakni YOLOv5n, YOLOR-CSP,

YOLOR-CSP-X, YOLOv7 tiny, YOLOv7, dan YOLOv7X. Berdasarkan perbandingan yang dilakukan, dipilih model YOLOv7 yang memiliki mAp sebesar 51.4%, GPU APT selama 6.2 ms, dan CPU APT selama 463 ms. Kemudian dilakukan perbandingan beberapa algoritma MOTO seperti SORT, DeepSORT, StrongSORT++, BoT SORT, BoT-SORT-ReID dengan pengujian pada MOT17 dan MOT20. Dipilih algoritma BoT-SORT sebagai pelacak yang memiliki nilai MOTA sebesar 80.6 pada MOT17 dan MOTA sebesar 77.7 pada MOT20. Setelah sistem dibentuk, dilakukan testing secara *real-time* dan mendapatkan kecepatan rata-rata pendeteksian dan pelacakan orang di antara 142.8 ms (7 FPS) dan 263 ms (3.8 FPS).

Pada referensi [19], metode yang digunakan pada penelitian ini ada sistem MCPT yang berbasis *spatio-temporal filtering* dan *group-aware matching* dari fitur-fitur penampilan seseorang. Penelitian ini menggunakan dataset Kumanoto Castle. Sistem yang diciptakan pada penelitian ini terdiri dari 4 proses, yakni proses deteksi objek, *re-identification*, *spatio-temporal filtering*, pembentukan grup. Eksperimen dilakukan dengan menggunakan *framework* Pytorch1.11 yang dijalankan pada GV100 dengan RAM 32 GB. Model deteksi objek yang digunakan pada penelitian ini adalah YOLOX, model MOT yang digunakan pada penelitian adalah ByteTrack, dan model *Re-identification* pada penelitian ini adalah MGN. Penelitian ini meningkatkan akurasi dari antara kamera-kamera menggunakan data yang didapatkan dari spot-spot turis. Dengan menggunakan *spatio-temporal*, sistem dapat berjalan secara efektif pada *scene* yang kompleks. Dan menggunakan penyesuaian *group-aware* (*group-aware matching*) dapat membuat sistem lebih kokoh dalam melacak kerumunan. Hasil dari sistem yang dibentuk diuji dengan metrik IDF1, IDP, dan IDR yang secara berturut-turut sebesar 78.91, 78.79, dan 79.03, yang di mana ketiga metrik pada metode yang digunakan lebih unggul daripada metode-metode lainnya yang dirujuk pada penelitian ini, yakni HCMIU-CVIP (Deep SORT) dan HCMIU-CVIP (ByteTrack).

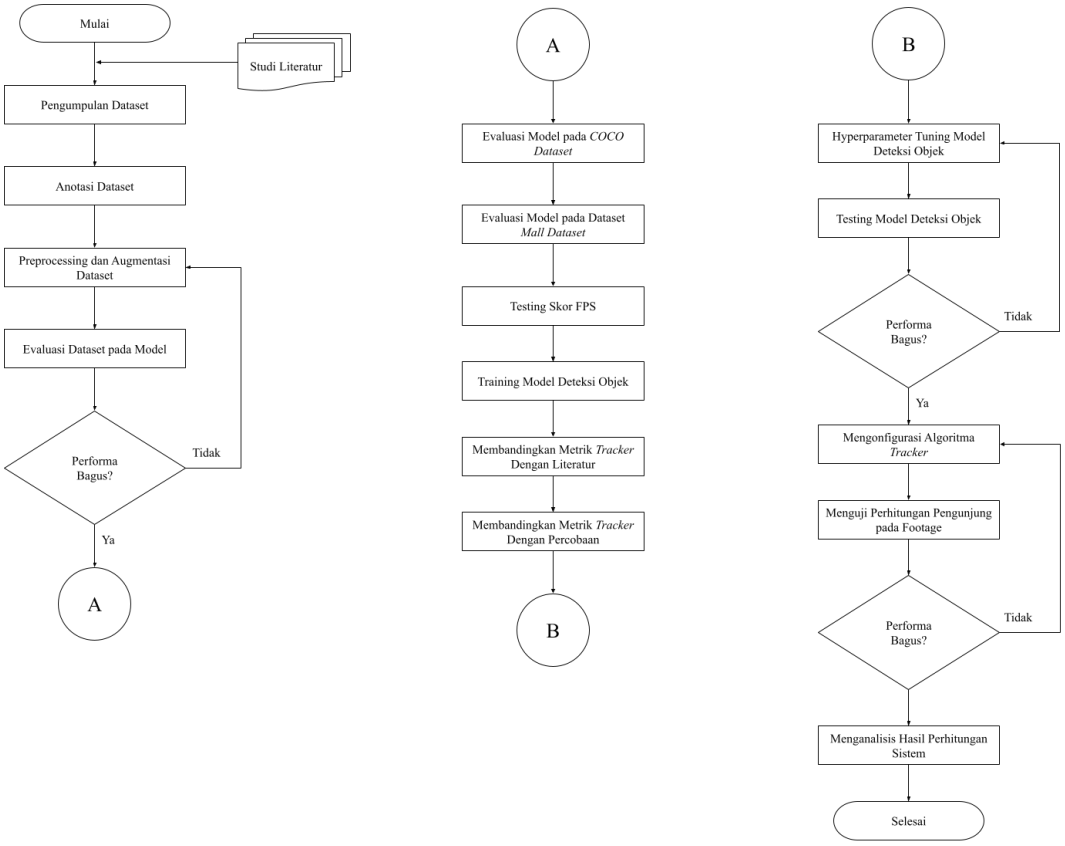
Pada referensi [20], metode yang digunakan adalah dengan menggabungkan dinamika kerumunan dengan *Extended* CAN (*Context Aware Network*). Metode ini menggunakan model *floor-field* dinamis dan statis serta penambahan suatu ketentuan hukuman (*penalty term*). *Floor-field* statis digunakan sebagai representasi jarak antara hambatan dan tujuan, sedangkan *floor-field* dinamis

digunakan sebagai pengarah arah pergerakan berdasarkan kepadatan manusia di setiap sel yang berdekatan. Dataset yang digunakan pada metode ini adalah CrowdFlow, FDST, Venice, CityStreet, dan UCSD. Seluruh dataset itu mengandung sejumlah gambar orang dan koordinat dari setiap individu sebagai label-label yang yang sudah dianotasi. Penggunaan *Extended* CAN yang ditambah dengan *floor-field* model dan *penalty term* dapat memiliki akurasi yang lebih baik dalam menghitung banyaknya orang pada kerumunan jikalau dibandingkan dengan model dasarnya, *Extended* CAN konvensional. Pengujian RMSE (*Root Mean Square Error*) *penalty term* pada penelitian ini untuk dataset CrowdFlow, FDST, Venice, CityStreet, dan UCSD berturut-turut adalah 158.3, 7.29, 18, 23.05, 1.02. Metrik yang dihasilkan selalu lebih unggul daripada metode lainnya, seperti MCNN, CAN, ECAN, *Extended* CAN konvensional, dan lain sebagainya.

BAB III
METODOLOGI PENELITIAN

3.1. Alur Penelitian

Diagram alir (*flowchart*) dari penelitian ini terdiri dari beberapa tahapan proses seperti yang dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Diagram Alur Penelitian

Berdasarkan Gambar 3.1, diagram alir dari penelitian ini dapat dipaparkan lebih lengkap dengan penjelasan sebagai berikut.

1. Pengumpulan Dataset

   Pengumpulan gambar diambil dengan menggunakan dataset bernama “Mall Dataset”. Dataset ini merupakan suatu dataset yang berupa kumpulan *frame* sebanyak 2000 gambar. Gambar yang terdapat pada dataset ini diambil dari

suatu web cam publik pada suatu mall yang secara umumnya digunakan dengan tujuan untuk menghitung kerumunan (*crowd counting*).

2. Anotasi Dataset

   Adapun anotasi dataset adalah suatu proses penentuan titik-titik lokasi terdapat objek pada suatu gambar dalam *pixel*, anotasi bertujuan untuk menentukan *bounding-box* pada objek yang kemudian akan dipakai untuk *training* dan evaluasi model.

3. *Preprocessing* Dataset

   Pada tahap ini, dataset diolah agar dapat digunakan untuk pengujian model, hal ini dikenal sebagai tahap *preprocessing* dataset. *Preprocessing* yang dilakukan pada penelitian ini adalah *resizing* dan *grayscaling*. Beberapa proses *preprocessing* yang diterapkan pada dataset adalah proses *Auto-Orient*, *Resizing* menjadi 640x640 yang merupakan ukuran umumnya pada model YOLO, penerapan *Grayscale* untuk meningkatkan performa model dalam pelatihan, dan penerapan *Auto-Adjust Contrast* dengan metode *Contrast Stretching*.

4. Augmentasi Dataset

   Pada metode ini dilakukan augmentasi dataset dengan mengubah beberapa aturan pada gambar, seperti *vertical flip*, *horizontal flip*, 90° *rotation*, *saturation*, *brightness*. Hal ini dilakukan agar dataset semakin bervariasi yang dapat membuat model dapat mempelajari suatu gambar yang sama namun dengan beragam modifikasi, sehingga membuat model semakin akurat dalam mendeteksi objek sekalipun mengalami modifikasi.

5. Seleksi Model Detektor

   Pada tahap ini, seleksi terhadap seri model YOLO akan dipakai. Model YOLO yang diuji terdiri dari YOLO seri ke-8 hingga ke-11 yang terkini. Untuk mempercepat proses komputasi, maka digunakan varian terendah dari seluruh seri YOLO 8 hingga 11. Yakni YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n. Seleksi model dilakukan dengan mengevaluasi setiap seri YOLO dengan varian terendah menggunakan dataset COCO. Seri YOLO yang akan dipakai sebagai detektor pada sistem penghitung pengunjung adalah sero YOLO yang menghasilkan keluaran terbaik di antara lainnya.

6. *Training* dan *Evaluation* Model

   Pada tahap kali ini, model yang sudah dibuat akan diuji dengan menggunakan dataset yang siap dipakai setelah dilakukannya *preprocessing* pada dataset. Kemudian evaluasi model, hal ini bertujuan untuk mengetahui keakurasian model jika mendeteksi pada kasus nyata tanpa menggunakan dataset yang sudah dipakai untuk dilatih.

7. *Benchmarking* Algoritma MOT

   Pada tahap ini dilakukan *benchmarking* algoritma *tracker* (pelacak) bertujuan untuk mendapatkan hasil keluaran terbaik dari algoritma *tracker*. Hal ini dilakukan dengan membandingkan nilai beberapa metrik untuk kemudian ditentukan yang terbaik untuk dipakai pada sistem penghitung pengunjung. Algoritma MOT yang akan di-*benchmark* adalah BoT-SORT dan ByteTrack.

8. *Hyperparameter Tuning*

   Proses *Hyperparameter Tuning* merupakan proses pengubahan beberapa parameter model YOLO. Proses ini dilakukan secara berulang-ulang hingga didapati hasil performa yang semaksimal mungkin atau dianggap sudah mencukupi.

9. Konfigurasi Algoritma *Tracker*

   Pada proses ini, dilakukan konfigurasi algoritma *tracker* yang dipilih. Proses ini dijalankan secara berulang-ulang hingga didapati hasil yang semaksimal mungkin.

10. Pembuatan Sistem Penghitung Pengunjung

    Pada tahap ini, dilakukan pembuatan sistem penghitung pengunjung dengan mengombinasikan algoritma *detector* dan *tracker*. *Detector* yang digunakan adalah seluruh varian algoritma YOLO dengan seri yang terbaik yang didapatkan pada tahap seleksi model detektor. Sedangkan, *tracker* yang digunakan adalah algoritma *tracker* dengan metrik terbaik yang didapatkan dari hasil *benchmarking*.

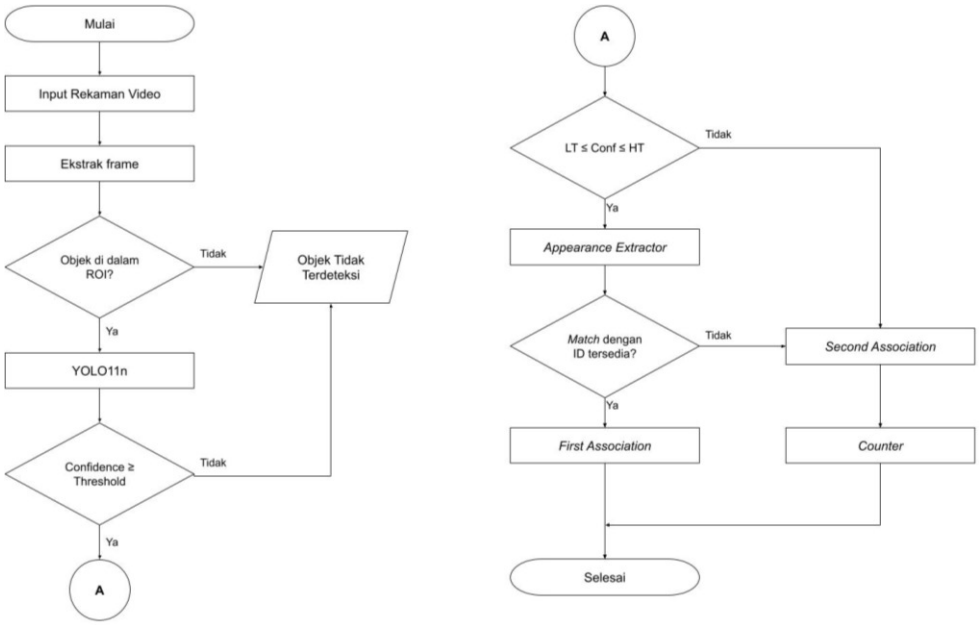
11. Menganalisis Hasil Keluaran Sistem Pelacak

    Pada tahap ini, sistem yang sudah terbentuk dari gabungan model deteksi YOLO terpilih dengan algoritma *tracker* yang terpilih akan diuji coba dengan menggunakan rekaman video, baik *indoor* atau pun *outdoor*. Hasil yang

diambil pada penelitian ini berupa persentase *error ID Switch* dan *Confusion Matrix*. Serta metrik yang dihasilkan untuk penelitian ini adalah *Precision*, *Recall*, dan skor F1

12. Diagram Alir Sistem

Sistem yang akan dibentuk pada penelitian kali ini akan menggunakan algoritma YOLO dan MOT. Sehingga, gambar diagram mekanisme dari sistem untuk menghitung pengunjung secara otomatis dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Diagram Alir Sistem Deteksi dan Penghitung Jumlah Pengunjung

Berdasarkan Gambar 3.2, sistem diberikan suatu masukan berupa rekaman video, setelah itu video tersebut akan diekstrak menjadi beberapa frame dari keseluruhan video. Lalu, sistem akan memotong daerah berdasarkan ROI agar hanya gambar di dalam area ROI saja yang dideteksi dan dilacak. Seluruh objek yang berada dalam ROI akan dilanjutkan kepada model YOLO11n untuk diberikan bounding box dengan skor confidence tertentu masing-masingnya. Kemudian, dilakukan pengujian apakah skor confidence dari deteksi diantara batas bawah (LT) dan batas atas (HT), jikalau benar maka akan dilanjutkan ke dalam sistem Appereance Extractor, namun jikalau tidak akan diberikan ID baru pada proses Second Association. Untuk objek yang

dilanjutkan kepada Appereance Extractor, maka akan dilakukan penyesuaian berdasarkan penampilan pada ID yang sudah terlacak sebelumnya, akan ada percabangan pada situasi ini, jikalau sesuai maka akan diberikan ID yang sudah ada (First Association), namun jikalau tidak sesuai maka akan di berikan ID baru (Second Association). Setiap ID baru akan dihitung dan kemudian akan ditotalkan pada sistem pelacak dan penghitung pengunjung secara otomatis.

3.2. Komponen Penelitian

Berikut ini adalah beberapa komponen penelitian yang digunakan pada penelitian kali ini.

1. PyCharm

   PyCharm merupakan suatu platform dari JetBrains yang digunakan untuk pembuatan suatu program dengan menggunakan bahasa pemrograman Python. PyCharm akan digunakan untuk melakukan simulasi hasil sistem deteksi objek dan penghitung jumlah kendaraan.

2. Google Colaboratory

   Google Colaboratory merupakan platform yang biasa digunakan untuk kasus-kasus *Machine Learning*, *Data Scientist*, dan AI berbasis *cloud*. Google Colaboratory akan digunakan pada penelitian ini untuk membuat model, melatih model, dan mengevaluasi model. Hal ini dikarenakan spesifikasi yang ditawarkan Google Colaboratory lebih unggul dari pada spesifikasi PC/Laptop.

3. Roboflow

   Roboflow merupakan platform yang menyediakan perlatan yang dibutuhkan oleh developer *Computer Vision*. Pada penelitian ini, Roboflow digunakan untuk pengumpulan dataset, anotasi dataset, *preprocessing* dataset, augmentasi dataset, dan pengaturan dataset lainnya.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1. Pengumpulan Dataset

Dalam penelitian kali ini, digunakan beberapa dataset yang digunakan dengan tujuan yang berbeda-beda. Dataset yang digunakan antara lain adalah dataset COCO, Mall Dataset, MOT-20, dan dataset pribadi. Ada pun masing-masing dataset digunakan dengan tujuan masing-masing seperti berikut.

1. COCO (*Common Objects in Context*) Dataset, merupakan dataset yang menampung banyak gambar objek dan secara khusus digunakan untuk model-model objek deteksi. Dataset COCO ini juga biasa digunakan sebagai data untuk membandingkan performa antar model-model yang berkenaan dengan deteksi objek. Dalam penelitian kali ini, digunakan versi ringan dari dataset COCO, yakni COCO128. COCO128 merupakan dataset yang terdiri dari 128 gambar pertama pada *train-set* dataset COCO 2017.
2. *Mall* Dataset, merupakan suatu set data yang dikoleksi dari *webcam* publik dan umumnya digunakan untuk menghitung jumlah kerumunan (*crowd counting*). Dataset ini terdiri dari 2000 gambar yang diambil dari 2000 *frame* video.
3. MOT20 Dataset, merupakan suatu dataset yang bertujuan untuk pengujian ataupun testing algoritma pelacak (*tracker*). *Multiple-object Tracking* 20 (MOT20) dataset terdiri dari 8 video yang terbagi menjadi 2 set bagian, 4 *train-set* dan 4 *test-set*. Anotasi dataset ini sudah memiliki tingkat akurasi yang sangat tinggi dan dengan protokol yang sangat ketat. Pada penelitian kali ini, digunakan *train-set* sebagai dataset untuk *benchmarking* algoritma pelacak, yakni ByteTrack dan BoT-SORT.
4. Dataset pribadi, merupakan dataset yang diambil dengan cara direkam sebagai pengujian performa sistem secara kasus nyata (*real case*).

4.2. Pemrosesan Dataset

Hal berikutnya setelah tahap mengumpulkan dataset adalah memproses dataset. Pemrosesan terdiri dari proses *preprocessing* dan augmentasi. Berikut ini

adalah penjelasan dari masing-masing proses *preprocessing* dan augmentasi dataset.

4.2.1. *Preprocessing* Dataset

Proses *preprocessing* adalah suatu proses pengolahan data yang dilakukan untuk mempersiapkan dataset agar sesuai dan juga layak untuk dipakai untuk proses training model. Proses *preprocessing* pada penelitian kali ini meliputi *resizing*, *grayscaling*, dan *contrast stretching*. Berikut adalah beberapa proses *preprocessing* yang dilakukan pada penelitian kali ini.

1. *Resizing*, merupakan proses pengubahan ukuran panjang atau lebar gambar. Ukuran umumnya dalam *resizing* untuk model YOLO adalah 640 x 640 px. Oleh sebab itu, pada penelitian kali ini, dilakukan perubahan ukuran (*resize*) gambar-gambar pada dataset menjadi 640 x 640 px.
2. *Contrast Stretching*, merupakan proses peningkatan kontras pada gambar. Proses ini dilakukan untuk memperjelas bagian-bagian antar pixel, dengan cara meningkatkan perbedaan intensitas nilai antar pixel, sehingga ketajaman objek semakin terlihat jelas dan mudah diproses oleh model. Pada penelitian ini, dilakukan proses penajaman kontras dengan metode *Contrast Stretching*, yang meningkatkan kontras setiap pixel dengan meningkatkan rentang nilai intensitasnya.

4.2.2. Augmentasi Dataset

Proses augmentasi merupakan proses mendiversifikasi gambar-gambar dalam dataset, sehingga jumlah gambar dalam dataset bertambah dengan modifikasi berbagai macam yang digunakan untuk membuat model agar mampu menggeneralisir gambar dan meningkatkan performa model pada gambar yang tidak ada di dataset sebelumnya, proses augmentasi dalam penelitian kali ini meliputi *flipping*, 90° *rotation*, *rotation*, *shear*, *saturation*, dan *brightness*.

1. *Horizontal flip* dan *vertical flip*, merupakan proses membalikkan gambar baik secara horizontal atau pun vertikal. Proses ini dilakukan secara otomatis dengan kemungkinan membalikkan gambar secara horizontal dan

vertikal sebesar 50%. Berikut ini adalah hasil dari augmentasi *flip* seperti yang ditampilkan pada Gambar 4.1.

Gambar 4. 1 Gambar Hasil Augmentasi *Flip*

Berdasarkan Gambar 4.1, didapati bahwa *input* gambar dilakukan proses *flip* baik secara horizontal dan vertikal yang dengan probabilitas masing-masing adalah 50%.

2. 90° *Rotation*, merupakan proses rotasi sebesar 90 derajat (90°). Proses ini dilakukan secara otomatis untuk 3 kemungkinan, yakni tidak dirotasi, rotasi searah jarum jam (*clockwise*), dan rotasi berlawanan arah jarum jam (*anti-clockwise*). Masing-masing kemungkinan tersebut memiliki kemungkinan yang sama besar, sehingga setiap kemungkinan memiliki probabilitas terjadi sebesar 33,33%. Berikut ini adalah hasil dari augmentasi rotasi 90° seperti yang ditampilkan pada Gambar 4.2.

Gambar 4. 2 Gambar Hasil Augmentasi Rotasi 90 Derajat

Berdasarkan Gambar 4.2, didapati bahwa dataset akan secara otomatis memperkaya gambar-gambar dengan augmentasi 90°, masing-masing memiliki probabilitas 33.33%.

3. *Saturation*, proses augmentasi pada saturasi merupakan proses augmentasi yang memperkaya dataset dengan mengubah saturasi warna pada gambar-gambar. Saturasi merupakan tingkat kejernihan warna suatu gambar, sehingga model akan diberi dengan beberapa *input* gambar yang memiliki tingkat kejernihan warna yang berbeda-beda, semakin tinggi nilai saturasi pada gambar dataset, maka akan semakin jernih gambar pada dataset. Maka dari itu, pada proses augmentasi ini dilakukan augmentasi saturasi dari rentang -15% dan 15%, nilai augmentasi saturasi ini tidak terlalu rendah atau tinggi, sehingga tidak membuat model kesulitan mendeteksi pada saat proses *training* berlangsung. Berikut ini adalah hasil dari augmentasi saturasi seperti yang tertera pada Gambar 4.3.

Gambar 4. 3 Gambar Hasil Augmentasi Saturasi

Berdasarkan Gambar 4.3, dataset diperkaya dengan augmentasi saturasi, yakni tingkat kejernihan warna dari gambar. Hasil augmentasi saturasi ini terdistribusi secara merata dari rentang -15% hingga 15%.

4. *Brightness*, merupakan proses meningkatkan atau menurunkan level kecerahan pada gambar. Proses ini dilakukan agar model dapat meningkatkan performanya pada saat mendeteksi objek dengan kecerahan yang berbeda dengan gambar asli dalam dataset, sehingga model dapat mendeteksi pada pencahayaan yang lebih gelap maupun lebih terang dari tingkat kecerahan aslinya. Pada penelitian kali ini, augmentasi pada *brightness*, dilakukan secara acak dengan pencahayaan antara -15% dan +15%. Berikut ini adalah hasil dari augmentasi dari *brightness* seperti yang tertera pada Gambar 4.4.

Gambar 4. 4 Gambar Hasil Augmentasi *Brightness*

Pada Gambar 4.4, diketahui bahwa nilai *brightness* tinggi membuat gambar lebih cerah dari gambar aslinya (gambar kiri), sedangkan nilai *brightness* rendah akan membuat gambar lebih gelap dari gambar aslinya (gambar kanan).

4.3. Seleksi Model *Object Detection* YOLO

Tahap selanjutnya setelah pemrosesan dataset adalah melakukan seleksi model deteksi objek. Seleksi model deteksi objek dilakukan untuk menentukan versi terbaik dari beberapa versi model YOLO terbaru, yakni dari YOLO versi 8, YOLO versi 9, YOLO versi 10, dan YOLO versi 11. Masing-masing versi dari model YOLO memiliki 5 varian berbeda sesuai dengan jumlah parameter modelnya. Oleh sebab itu, pada penelitian ini dilakukan seleksi model dengan mengambil masing-masing varian terendah pada setiap versi model YOLO. Hal ini dilakukan dengan tujuan mengurangi waktu komputasi saat evaluasi model karena mengurangi tingkat kompleksitas saat evaluasi model. Sehingga, dipilih varian YOLOv8n, YOLOv9t, YOLOv10n, dan YOLO11n.

Proses seleksi model dilakukan dengan melakukan evaluasi model pada dataset "COCO" dan "*Mall Dataset*". Dataset COCO dilakukan sebagai tahap awal seleksi, karena dataset ini merupakan dataset yang umum digunakan untuk melakukan *benchmarking* model-model deteksi objek. Hal ini dilakukan untuk mengetahui seberapa baik performa model dalam mendeteksi objek secara umum. Sedangkan, tahap seleksi berikutnya dengan melakukan evaluasi pada dataset "*Mall Dataset*", hal ini dikarenakan dataset tersebut mengandung *frame-frame* yang menunjukkan objek khusus, yakni orang. Karena pada penelitian kali ini bertujuan untuk

mendeteksi pada rekaman pengunjung, maka dataset ini sangat sesuai dengan kebutuhan dari tujuan penelitian kali ini. Proses seleksi model kali ini dilakukan dengan menggunakan Google Colaboratory dengan menggunakan tipe *runtime* GPU T4. Berikut ini adalah penjelasan lebih lengkap mengenai tahap masing-masing evaluasi model.

4.3.1. Evaluasi pada *COCO Dataset*

Tahap pertama pada seleksi model adalah dengan melakukan evaluasi model dengan menggunakan dataset bernama COCO (*Common Objects in Context*). Setelah dilakukan evaluasi terhadap model-model, maka berikut ini adalah hasil-hasil beberapa metrik seperti yang terdapat pada Tabel 4.1.

Tabel 4. 1 Tabel Hasil Evaluasi Model pada COCO Dataset

Model	*Precision*	*Recall*	mAP50	mAP50-95	*Fitness*
YOLOv8n	0. 639	0. 536	0. 607	0. 448	0. 464
YOLOv9t	0. 679	0. 548	0. 606	0. 453	0. 468
YOLOv10n	0. 645	0. 558	0. 625	0. 462	0. 478
YOLOv11n	0. 663	0. 589	0. 671	0. 505	0. 522

Berdasarkan Tabel 4.1, didapati bahwa, nilai *precision* pada YOLOv9t mengungguli versi lainnya dengan nilai 0.679. Sedangkan, pada metrik *recall*, model YOLOv11n mengungguli versi lainnya dengan nilai 0.589. Lalu, pada metrik mAP50 dan mAP50-95, model YOLOv11n mengungguli versi lainnya dengan nilai secara berurutan sebesar 0.671 dan 0.505. Dan terakhir pada metrik *fitness*, nilai terbesar masih dimiliki oleh YOLOv11n dengan nilai 0.522. Oleh sebab itu, untuk tahap awal seleksi model, maka dipilih model YOLOv11n sebagai model deteksi objek, akan tetapi akan dilakukan tahap kedua seleksi model dengan *task* yang sesuai dengan tujuan penelitian kali ini, yakni dengan menggunakan Mall Dataset.

4.3.2. Evaluasi pada *Mall Dataset*

Setelah selesai dilakukan evaluasi tahap pertama, yakni evaluasi model-model menggunakan dataset COCO, tahap berikutnya adalah dengan mengevaluasi model-model dengan menggunakan dataset bernama "*Mall Dataset*". Hal ini dikarenakan dataset ini sesuai dengan tujuan penelitian, yakni mendeteksi dan

menghitung jumlah pengunjung. Tahap evaluasi kedua ini dilakukan dengan menggunakan Google Colaboratory pada tipe *runtime* GPU T4. Dataset ini sudah melewati tahap *preprocessing* dan augmentasi, sehingga dataset ini lebih kompleks daripada dataset orisinalnya.

Jika proses evaluasi untuk semua model sudah selesai, maka hasil metrik-metrik setiap model secara otomatis akan tersimpan dan ditampilkan. Hasil dari evaluasi model-model pada *Mall Dataset* dapat dilihat pada Tabel 4.2.

Tabel 4. 2 Tabel Hasil Evaluasi Model pada Mall Dataset

Model	*Precision*	*Recall*	mAP50	mAP50-95	*Fitness*
YOLOv8n	0.853	0.745	0.854	0.631	0.653
YOLOv9t	0.856	0.760	0.869	0.654	0.675
YOLOv10n	0.834	0.730	0.843	0.634	0.655
YOLOv11n	0.867	0.771	0.881	0.653	0.676

Berdasarkan nilai pada Tabel 4.2, didapati bahwa secara keseluruhan nilai tertinggi setiap metrik dimiliki oleh YOLOv11n. Model YOLOv11n mengungguli nilai-nilai metrik *recall*, mAP50, mAP50-95, dan *fitness* dari pada model-model lainnya, akan tetapi nilai metrik mAP50-95 masih diungguli oleh model YOLOv9t. Model YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n secara berurutan pada metrik *precision* adalah 0.853, 0.856, 0.834, dan 0.867. Sedangkan, model YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n secara berurutan pada metrik *recall* adalah 0.745, 0.760, 0.730, dan 0.771. Kemudian, model YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n secara berurutan pada metrik mAP50 adalah 0.854, 0.869, 0.843, dan 0.881. Kemudian, model YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n secara berurutan pada metrik mAP50-95 adalah 0.631, 0.654, 0.634, dan 0.653. Dan metrik terakhir, yakni *fitness*, model YOLOv8n, YOLOv9t, YOLOv10n, dan YOLOv11n secara berurutan adalah 0.653, 0.675, 0.655, dan 0. 676.

Berdasarkan hasil *benchmarking* kepada 2 dataset, yakni COCO dan Mall Dataset, didapati bahwa model YOLOv11n memiliki keunggulan jikalau dibandingkan dengan model YOLO versi 8, 9, dan 10. *Benchmarking* menggunakan varian terendah pada setiap versi model YOLO versi 8, 9, 10, dan 11, sehingga versi yang unggul akan digunakan pada penelitian kali ini. Oleh karena model

YOLOv11n mengungguli versi lainnya, maka akan dipilih seluruh varian dari model YOLO versi 11, yakni YOLOv11n, YOLOv11s, YOLOv11m, YOLOv11l, dan YOLOv11x. Hal berikutnya adalah dengan melakukan *benchmarking* untuk keseluruhan varian dari model YOLO versi 11 dengan melakukan *training*.

4.3.3. *Training* Model

Tahap berikutnya setelah mendapatkan versi terbaik dari model YOLO adalah dilakukannya proses *training* model pada seluruh varian model YOLO versi 11, yakni YOLOv11n, YOLOv11s, YOLOv11m, YOLOv11l, dan YOLOv11x. Proses *training* kali ini digunakan dengan menggunakan dataset yang sudah dipersiapkan sebelumnya pada platform Roboflow, yakni dataset yang bernama "Mall Dataset", hal ini dikarenakan "Mall Dataset" memiliki kasus yang serupa dengan penugasan pada penelitian saat ini. Proses *training* model dilakukan dengan menggunakan platform Google Colaboratory dengan spesifikasi tipe *runtime* GPU A100 dan RAM sebesar 80 GB dan VRAM sebesar 40GB. Terdapat beberapa tahapan dalam *training* model-model YOLO versi 11 ini, dimulai dari dengan menginstall beberapa *library* yang dibutuhkan, mengimpor beberapa *library*, mengambil dataset dengan API "Roboflow", memanggil model-model dari "Ultralytics", melakukan proses *training*, hingga menyimpan dan mengunduh hasil dari *training* model-model.

Setiap model dilakukan *training* dengan *epoch* sebanyak 100 kali dan *optimizer* diatur secara otomatis dengan pengaturan *default* dari *library* Ultralytics. Proses training ini dilakukan dengan tujuan untuk melakukan *update* bias dan *weigth* hingga menemukan akurasi yang meningkat dari setiap *epoch*. Setiap *epoch* dari proses *training* dilakukan dengan cara mempelajari keseluruhan dataset yang diberikan, kemudian model akan melakukan *update* parameter, seperti *weight* dan bias, dengan menggunakan *optimizer* dari pembelajaran pada *epoch* tersebut. Setelah proses *training* selesai dilakukan, maka akan terdapat nilai dari beberapa metrik deteksi objek, yakni *precision*, *recall*, mAP50, mAP50-95, lamanya waktu *training* yang dibutuhkan. Hasil *training* dari seluruh varian model YOLO versi 11 tertera pada Tabel 4.3.

Tabel 4. 3 Tabel Hasil *Training* YOLOv11

Model	*Precision*	*Recall*	mAP50	mAP50-95	Waktu *Training* (menit)
YOLO11n	0.952	0.947	0.986	0.905	39.78
YOLO11s	0.957	0.949	0.988	0.924	42.48
YOLO11m	0.959	0.946	0.987	0.928	54.48
YOLO11l	0.958	0.948	0.987	0.927	73.08
YOLO11x	0.956	0.953	0.987	0.927	90.84

Berdasarkan data pada Tabel 4.3, didapatkan bahwa metrik *precision* dari varian *nano* (n), *small* (s), *medium* (m), *large* (l), dan *extra large* (x) secara berturut-turut adalah 0.952, 0.957, 0.959, 0.958, dan 0.956 yang di mana YOLO11m (*medium*) memiliki nilai terbesar pada metrik ini. Berikutnya didapatkan bahwa metrik *recall* dari varian *nano* (n), *small* (s), *medium* (m), *large* (l), dan *extra large* (x) secara berturut-turut adalah 0.947, 0.949, 0.946, 0.948, dan 0.953, sehingga YOLO11x (*extra large*) memiliki nilai terbesar pada metrik ini. Selanjutnya didapati bahwa metrik mAP50 dari varian *nano* (n), *small* (s), *medium* (m), *large* (l), dan *extra large* (x) secara berturut-turut adalah 0.986, 0.988, 0.987, 0.987, dan 0.987, di mana YOLO11s (*small*) memiliki nilai terbesar pada metrik ini. Dan yang terakhir didapati bahwa metrik mAP50 dari varian *nano* (n), *small* (s), *medium* (m), *large* (l), dan *extra large* (x) secara berturut-turut adalah 0.905, 924, 0.928, 0.927 dan 0.927, di mana YOLO11m (*medium*) memiliki nilai terbesar pada metrik ini. Oleh sebab itu, didapati bahwa YOLO11m (*medium*) mengungguli mayoritas metrik hasil evaluasi di antara varian model lainnya, yakni unggul pada metrik *precision* dan mAP50-95. Sehingga, pada tahap seleksi model ini, untuk sementara digunakan model YOLO11m (*medium*), kemudian dilanjutkan pada tahap hasil uji skor FPS pada kasus nyata untuk menemukan model dengan performa yang baik namun memiliki kecepatan yang memungkinkan untuk digunakan.

4.3.4. Test FPS Score

Setelah diketahui bahwa model YOLO11m menjadi yang unggul karena memiliki performa terbaik secara keseluruhan di antara varian lainnya, maka hal berikutnya yang harus diperhatikan adalah tingkat kekompleksan dari model untuk

dijalankan secara *real-time* pada suatu PC. Semakin kompleks parameter pada suatu model, maka akan semakin lama suatu model akan melakukan pemrosesan gambar, baik pada saat *training*, evaluasi, ataupun deteksi secara *real-time*. Oleh sebab itu, pada penelitian kali ini dilakukan pengujian untuk mengetahui tingkat kekompleksan model pada perangkat dengan mengukur rata-rata skor FPS (*Frame per Second*) yang dihasilkan dari setiap model pada percobaan dengan menggunakan video *footage* yang sama untuk keseluruhan varian model YOLO versi 11. Pengujian skor *Frame Per Second* (FPS) berikut ini dilakukan dengan menggunakan laptop dengan spesifikasi seperti yang tertera pada Tabel 4.4.

Tabel 4. 4 Tabel Spesifikasi PC

Komponen	Spesifikasi
OS	Windows 11 64 bit
CPU	AMD Ryzen 3 3250U 2.60 GHz
GPU	AMD Radeon Graphics 2GB VRAM
RAM	12 GB
Storage	Intel SSDPEKNU512GZ

Berdasarkan Tabel 4.4 didapati bahwa spesifikasi PC yang digunakan untuk melakukan sistem pelacakan tidak menggunakan bantuan API CUDA (*Compute Unified Device Architecture*) yang memungkinkan performa *image processing* menjadi lebih cepat karena gambar dikerjakan secara paralel dengan GPU. Sehingga, karena spesifikasi PC hanya menggunakan bantuan komputasi CPU, maka akan dilakukan pengujian skor FPS pada masing-masing varian agar mendapatkan model dengan performa dan kecepatan yang seimbang untuk diaplikasikan. Hasil dari nilai rata-rata FPS pada masing-masing varian model YOLO11 dapat dilihat pada Tabel 4.5.

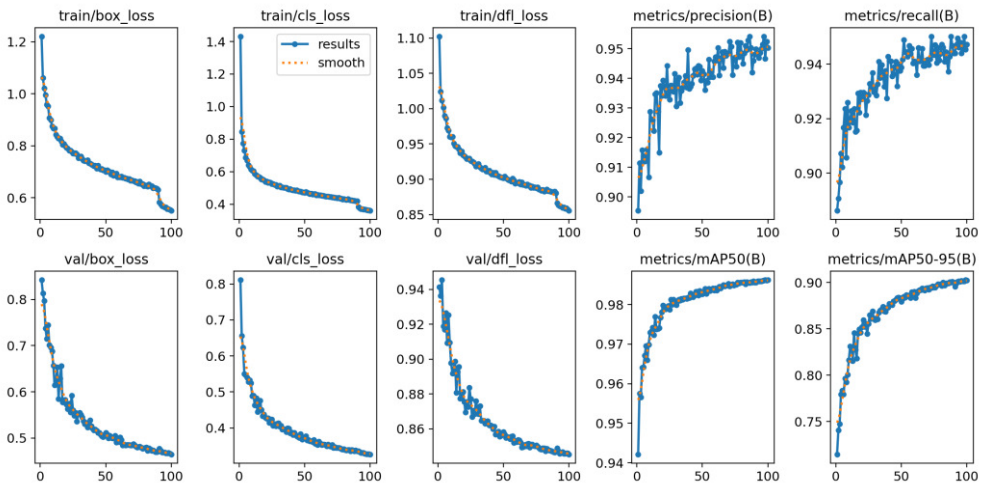
Tabel 4. 5 Tabel Hasil Skor Rata-rata FPS

Model	Rata-rata Skor FPS
YOLO11n	3.22
YOLO11s	1.53
YOLO11m	0.53

YOLO11l	0.42
YOLO11x	0.19

Berdasarkan Tabel 4.5, didapatkan bahwa model YOLO11n memiliki skor rata-rata FPS tertinggi dari varian lainnya, hal ini karena varian *nano* merupakan varian terendah dari varian lainnya pada model YOLO versi 11. Jikalau suatu video memiliki *frame* sebesar 30 FPS, berarti setiap detik pada video tersebut terdapat 30 *frame*. Maka, jikalau menggunakan YOLO11n, dibutuhkan sekitar 9.31 detik untuk menyelesaikan 1 detik video orisinalnya. Sedangkan dibutuhkan 19.60 detik untuk YOLO11s, 56 detik untuk YOLO11m, dibutuhkan 71.42 detik untuk model YOLO11l, dan dibutuhkan 157.89 detik untuk model YOLO11x.

Oleh karena kepentingan komputasi, maka digunakan versi terendah, yakni model YOLO11n. Walaupun model YOLO11n merupakan versi terendah dan paling ringan sehingga mampu memproses video dengan cepat dibandingkan dengan varian lainnya, tetapi hal ini tidak mengorbankan performa yang besar seperti hasil beberapa metrik dari proses *training* pada Tabel 4.3. Di mana metrik *precision* antara versi *nano* dan *medium* hanya berbeda 0.0007, kemudian pada metrik *recall* perbedaannya hanya 0.001, lalu pada metrik mAP50 perbedaannya hanya 0.001, dan pada metrik mAP50-95 perbedaannya hanya 0.023. Dengan perbedaan performa yang sangat kecil tersebut, model YOLO11n dapat menjalankan sistem 6 kali lebih cepat dibandingkan dengan model YOLO11m. Grafik hasil *training* dari model YOLO11n dapat dilihat pada Gambar 4.5



Gambar 4. 5 Gambar Hasil *Training* YOLO11n

Berdasarkan Gambar 4.5, terdapat 6 grafik *loss*, grafik-grafik tersebut menjelaskan seberapa besar penurunan *loss* (galat) pada model selama proses *training*, yakni 100 *epoch*. Pada grafik *box loss*, didapati bahwa terdapat penurunan eksponensial baik pada *train-set* maupun *val-set*, hal ini menandakan bahwa terjadi performa optimal pada model sehingga tidak ada *overfitting* maupun *underfitting*. Demikian juga untuk *cls_loss* dan *dfl_loss*, keduanya memiliki bentuk grafik yang sama baik pada *train-set* dan *val-set* yang terus menurun secara eksponensial. Sedangkan, 4 grafik lainnya (*precision*, *recall*, mAP50, dan mAP50-95) mengalami peningkatan secara eksponensial di setiap *epoch*-nya.

4.4. Seleksi Algoritma MoT (*Multiple-Object Tracking*)

Hal berikutnya yang dilakukan setelah selesai dengan segala urusan pada model objek deteksi adalah membandingkan algoritma pelacak (*tracker*) untuk melacak objek yang sudah terdeteksi. *Tracker* pada penelitian ini digunakan untuk melacak pergerakan objek yang sudah terdeteksi, sehingga jikalau terdapat objek dengan id yang sama pada rentang *frame* tertentu, *id* objek tersebut tidak akan terdeteksi sebagai objek yang berbeda pada *frame* berikutnya, sehingga akan tetap terhitung sebagai 1 *id* objek saja. Pada penelitian kali ini, akan dibandingkan 2 algoritma *tracker* populer, yakni BoT-SORT dan ByteTrack. Tahap pemilihan *tracker* yang terbaik dilakukan dengan *benchmarking* kedua *tracker* tersebut seperti yang akan dijelaskan di bawah berikut ini.

Oleh sebab itu, pada penelitian ini, dibandingkan kedua hasil evaluasi pada *test-set* dari hasil resmi masing-masing developer *tracker*. Pada algoritma BoT-SORT, dapat dilihat pada *link* GitHub berikut: "https://github.com/NirAharon/BoT-SORT". Sedangkan untuk algoritma ByteTrack dapat dilihat pada *link* GitHub berikut: "https://github.com/ifzhang/ByteTrack". Keduanya dievaluasi dengan menggunakan model yang sama, yakni model YOLOX. Kedua *tracker* ini juga diuji dengan menggunakan dataset yang bernama MOT20 (*Multiple-Object Tracking* 20) dan keduanya juga menggunakan *sequence* pada *test-set*. Hasil dari nilai-nilai beberapa metrik utama yang didapatkan dari kedua algoritma *tracker* pada MOT20 (*test-set*) dapat dilihat pada Tabel 4.6.

Tabel 4. 6 Tabel Hasil Evaluasi Resmi dari *Tracker*

Tracker	MOTA	HOTA	IDF1
ByteTrack	77.8	61.3	75.2
BoT-SORT	77.7	62.6	76.3

Berdasarkan Tabel 4.6, didapati bahwa kedua *tracker* memiliki performa yang hampir sama satu sama lainnya, akan tetapi BoT-SORT memiliki nilai metrik yang sedikit lebih baik dari metrik IDF1 dan HOTA. Metrik IDF1 pada BoT-SORT memiliki nilai 1.1 lebih tinggi, sedangkan pada metrik HOTA BoT-SORT memiliki nilai metrik 1.3 lebih tinggi.

Hal berikutnya setelah mengetahui nilai metrik-metrik pada studi literatur resminya, hal berikutnya yang dilakukan adalah dengan membandingkan kedua algoritma *tracker*, yakni BoT-SORT dan ByteTrack, dengan menggunakan TrackEval. TrackEval merupakan suatu program yang digunakan untuk mengetahui metrik evaluasi dari suatu *tracker*. TrackEval membutuhkan minimal 2 data untuk dibandingkan, yakni data *ground-truth* dan data hasil *tracking*. Data *ground-truth* merupakan data asli dari anotasi manual, data tersebut digunakan sebagai pembanding dengan data hasil pelacakan pada *tracker*. Pada penelitian kali ini, digunakan model YOLO11n untuk membandingkan kedua *tracker* saja, tidak untuk mendapatkan nilai metrik yang baik.

Dataset yang akan digunakan untuk mengevaluasi kedua *tracker* adalah dataset MOT-20 (*Multiple Object Tracking* 20) yang berisikan banyak *frame* dengan jumlah orang yang sangat besar. Pada proses *benchmarking tracker*, digunakan *train-set* dari dataset MOT-20 dan hanya mengambil sekuen pertama saja, yakni MOT20-01. Oleh karena dataset ini memiliki frame-frame yang sangat kompleks, maka akan sangat mungkin bahwa hasil metrik akan buruk, akan tetapi pada tahap kali ini hanya akan melihat perbandingan kedua performa dari *tracker*. Hasil dari evaluasi kedua *tracker* dengan menggunakan TrackEval dapat dilihat pada Tabel 4.7.

Tabel 4. 7 Tabel Hasil Evaluasi Dengan TrackEval

Tracker	MOTA	HOTA	IDF1
ByteTrack	17.8	32.7	36.5
BoT-SORT	17.8	32.7	36.5

Berdasarkan nilai pada Tabel 4.7, didapatkan hasil yang sangat buruk bagi keduanya, akan tetapi hal ini sangat wajar oleh karena *tracking* dijalankan hanya dengan menggunakan model seadanya yang tidak disiapkan secara khusus untuk mendeteksi kerumunan masa secara ekstrem seperti yang ada pada gambar-gambar di dataset MOT20. Akan tetapi, tujuannya adalah hanya untuk membandingkan performa pelacakan antara BoT-SORT dan ByteTrack. Maka didapatkan hasil yang sama dan tidak ada perbedaan secara signifikan dari keduanya pada dataset MOT20 *sequence* pertama (MOT20-01). Setelah membandingkannya dengan *train-set* pada MOT20, maka berikutnya adalah dengan membandingkan keduanya pada *test-set* MOT20. *Test-set* tidak dapat dijalankan secara publik, hanyalah lembaga atau organisasi tertentu saja yang terdaftar pada "motchallenge.net" saja yang dapat melakukan pengujian pada *test-set* MOT20. Berdasarkan Tabel 7 dan Tabel 8, kedua *tracker* memiliki performa yang hampir sama satu sama lainnya, akan tetapi BoT-SORT memiliki keunggulan sedikit lebih baik dari metrik IDF1 dan HOTA. Selain itu, fitur BoT-SORT juga dilengkapi dengan *Global Motion Compensation* (gmc) dan *re-ID*, sehingga membantu mengatasi pergeseran atau getaran pada video karena rekaman diambil dengan dipegang menggunakan tangan, selain itu juga membantu mengurangi galat *ID Switch* dengan bantuan *re-ID*.

4.5. *Hyperparameter* Tuning

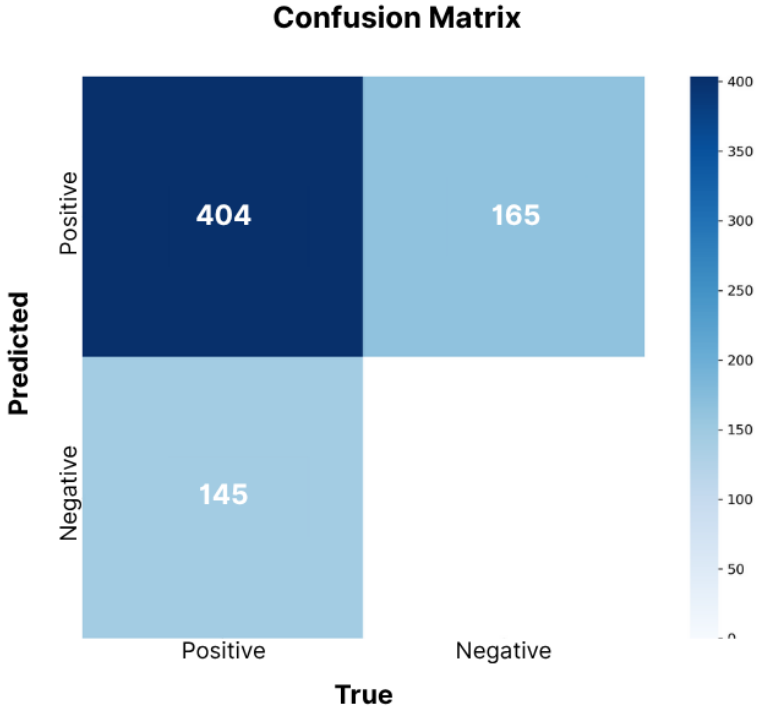
Hal berikutnya untuk memperkuat performa model deteksi adalah dengan melakukan *hyperparameter tuning*, *hyperparameter tuning* merupakan proses pengubahan beberapa konfigurasi pada model yang dapat memengaruhi kinerja model. Terdapat banyak parameter-parameter yang ada pada suatu model YOLO, pada penelitian kali ini *hyperaprameter tuning* dilakukan secara otomatis dengan menggunakan *library* Ultralytics. Dilakukan perubahan *optimizer*, yang semula adalah menggunakan optimizer dari AdamW, digantikan dengan menggunakan *optimizer* dari SGD (*Stochastic Gradient Descent*).*Hyperparamter*. Selain itu, dilakukan juga perubahan nilai learning rate, dari 0.002 menjadi 0.01, hal ini bertujuan untuk membuat proses *tuning* menjadi lebih cepat. *Tuning* kali ini dilakukan dengan menggunakan dataset gabungan dari seluruh *footage* rekaman yang diambil dengan menggunakan kamera *smartphone*. Hasil dari perbandingan

nilai metrik model sebelum dan setelah dilakukan *hyperparameter tuning* dapat dilihat Tabel 4.8.

Tabel 4. 8 Tabel Hasil Evaluasi Dengan TrackEval

Model	*Precision*	*Recall*	mAP50	mAP50-95	*Fitness*
Sebelum *Hyperparameter Tuning*	0.250	0.027	0.029	0.017	0.019
Setelah *Hyperparameter Tuning*	0.687	0.705	0.734	0.340	0.380

Berdasarkan Tabel 4.8, didapati bahwa terdapat perbedaan yang sangat signifikan dari nilai metrik-metrik seperti *Precision*, *Recall*, mAp50, mAp50-95, dan *Fitness*. Hal ini dikarenakan proses *hyperparameter tuning* ini merupakan iterasi dari *training-evaluation-logging* secara berkala hingga akhirnya menemukan parameter terbaik secara otomatis dari model untuk dataset yang spesifik. Untuk mendapatkan kesimpulan yang lebih baik, dilakukan visualisasi dengan menggunakan *Confusion Matrix*. Hasil dari *Confusion Matrix* untuk hasil evaluasi model setelah dilakukan *Hyperparameter Tuning* dapat dilihat pada Gambar 4.6.



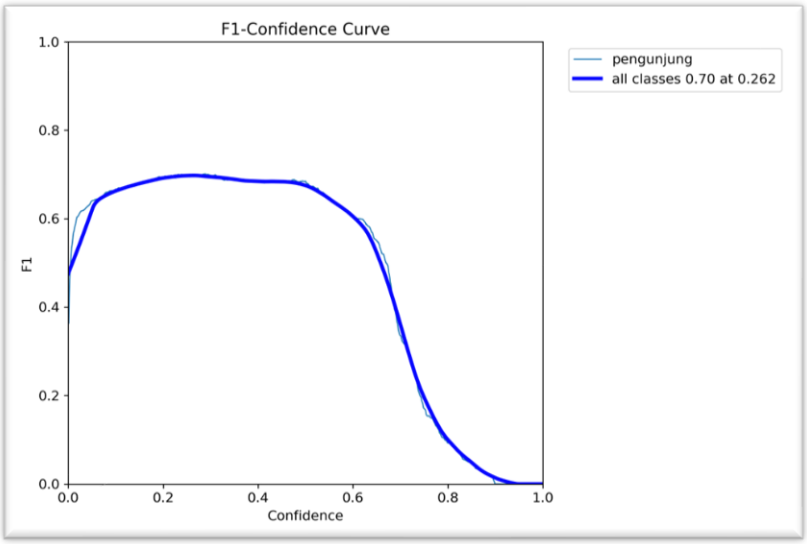
Gambar 4. 6 Gambar *Confusion Matrix* Model YOLO

Berdasarkan Gambar 4.6, terdapat suatu gambar matriks yang disebut sebagai *Confusion Matrix*. Pada *Confusion Matrix* tersebut terdapat 4 sel, yang terdiri dari *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN). Sel pertama pada matriks merupakan nilai dari *True Posituve* (TP), lalu sel kedua pada matriks merupakan nilai dari *False Positive* (FP), kemudian sel ketiga pada matriks merupakan nilai dari *False Negative* (FN), dan sel keempat adalah nilai dari *True Negative* (TN). Kondisi dinyatakan sebagai *True Positive* ketika model berhasil mendeteksi objek dan klasifikasi objek dengan kelas yang tepat, sedangkan kondisi *False Postive* terjadi ketika model mendeteksi namun salah dalam mengklasifikasikan kelas objek, lalu kondisi *False Negative* terjadi ketika model gagal mendeteksi objek yang seharusnya terdeteksi pada dataset *Ground Truth*, dan yang terakhir adalah kondisi *True Negative* merupakan kondisi ketika model berhasil untuk tidak mendeteksi objek-objek yang memang seharusnya tidak terdeteksi.

Hasil *Confusion Matriks* pada Gambar 4.6 mendapatkan nilai *True Positive* (TP) sebesar 404 pada sel kiri atas, yakni ketika model berhasil mendeteksi "pengunjung" ketika nilai aslinya juga "pengunjung". Setelah itu, terdapat nilai *False Positive* (FP) pada sel kanan atas sebesar 165, yakni ketika model mendeteksi objek sebagai "pengunjung" ketika seharusnya tidak ada (*background*). Kemudian, terdapat nilai *False Negative* (FN) pada sel kiri bawah, yakni ketika model tidak mendeteksi objek sebagai "pengunjung" yang seharusnya terdeteksi sebagai "pengunjung". Dan terakhir terdapat nilai *True Negative* (TN) pada sel kanan bawah, yang bernilai 0, nilai pada sel ini bertujuan untuk mengetahui seberapa banyak model tidak mendeteksi objek (*background*) yang seharusnya memang tidak terdeteksi (*background*).

Selain hasil *Confusion Matrix*, didapatkan juga hasil keharmonisan antara metrik *precision* dan *recall*, yang dinamakan sebagai metrik F1-Score. Hal ini dikarenakan metrik *precision* berbanding lurus dengan F1-score, akan tetapi metrik *recall* berbanding terbalik dengan metrik F1-Score. Hal ini menyebabkan ketika *Confidence* ditingkatkan, maka *precision* akan meningkat akan tetapi *recall* menurun, dan begitu pula sebaliknya. Maka diperlukan keharmonisan antara

precision dan *recall* yang direpresentasikan oleh nilai dari metrik F1-Score. Grafik dari F1-Score dapat dilihat pada Gambar 4.7.



Gambar 4. 7 Kurva F1-Confidence

Berdasarkan Gambar 4.7 terdapat suatu kurva yang merupakan kurva relasi antara skor *Confidence* dengan nilai F1. Skor *Confidence* merupakan skor kepercayaan model untuk mendeteksi suatu objek dengan kelas yang benar, sedangkan skor F1 merupakan skor harmoni antara *Precision* dan *Recall*. Nilai *Precision* merupakan nilai seberapa tepat model memprediksi seluruh objek yang terdeteksi dengan benar, sedangkan *Recall* merupakan nilai seberapa sering model memprediksi objek. Jikalau skor *Confidence* terlalu tinggi maka nilai *Precision* akan tinggi pula, akan tetapi nilai *Recall* akan rendah. Hal ini karena skor *Confidence* berbanding lurus dengan *Precision*, namun berbanding terbalik dengan *Recall*. Sehingga, nilai F1 pada Gambar 13 merepresentasikan nilai harmoni antara *Precision* dan *Recall* untuk menghasilkan performa deteksi yang baik. Pada hasil evaluasi, didapatkan bahwa skor *Confidence* 0.262 menghasilkan skor F1 sebesar 0.70 atau 70%.

4.6. Pengujian Kasus Nyata

Setelah sudah selesai melakukan *hyperparameter tuning* model deteksi objek, maka selanjutnya adalah mengombinasikannya dengan *tracker* yang bernama

"BoT-SORT". Sama halnya seperti model YOLO, algoritma BoT-SORT juga memiliki beberapa konfigurasi yang dapat diatur sedemikian rupa, sehingga memiliki keluasan dalam mengatur performa pelacakan berdasarkan kesulitan masing-masing kasus yang disebabkan oleh beberapa faktor, misalnya pencahayaan tempat, oklusi, ukuran objek-objek pada *frame*, dan lain-lainnya. Pada tahap ini, dilakukan konfigurasi terhadap *tracker* untuk menyesuaikan dengan model dan hasil rekaman agar dapat menghasilkan performa yang maksimal. Setiap rekaman diberikan *Region Of Interest* (ROI) untuk membatasi daerah, agar daerah tertentu saja yang dilakukan perhitungan jumlah pengunjung, tidak terhadap keseluruhan *frame* rekaman. Berikut ini adalah penjelasan lebih lengkap untuk pengujian kasus nyata.

4.6.1. Melakukan Konfigurasi *Tracker*

Tahap selanjutnya sebelum melakukan percobaan perhitungan jumlah orang adalah dengan melakukan konfigurasi pada algoritma *tracker*. Beberapa konfigurasi pada BoT-SORT dapat diubah,, seperti *track_high_thresh, track_low_thresh, new_track_thresh, track_buffer, match_thresh, proximity_thresh,* dan *appearance_thresh*. Nilai-nilai tersebut akan memengaruhi performa dari proses pelacakan, hal ini dapat diatur sedemikian rupa berdasarkan kasus-kasus tertentu, karena setiap kasus memiliki beberapa faktor yang berbeda, sehingga diberikan keluasan untuk mengatur nilai konfigurasi agar lebih maksimal dan dapat digunakan untuk kasus yang bervariasi. Nilai-nilai tersebut ditentukan secara manual pada *footage* untuk performanya pada percobaan kasus nyata. Pada penelitian kali ini dilakukan percobaan pengujian masing-masing konfigurasi secara manual pada *footage* hingga mendapatkan hasil yang terbaik berdasarkan kombinasi keseluruhan konfigurasi. Konfigurasi-konfigurasi yang digunakan pada algoritma *tracker* kali ini dapat dilihat seperti yang tertera pada Tabel 4.9.

Tabel 4. 9 Tabel Konfigurasi Algoritma *Tracker*

Konfigurasi	Nilai
Tipe *Tracker*	BoT-SORT
Track High Threshold	0.50
Track Low Threshold	0.39

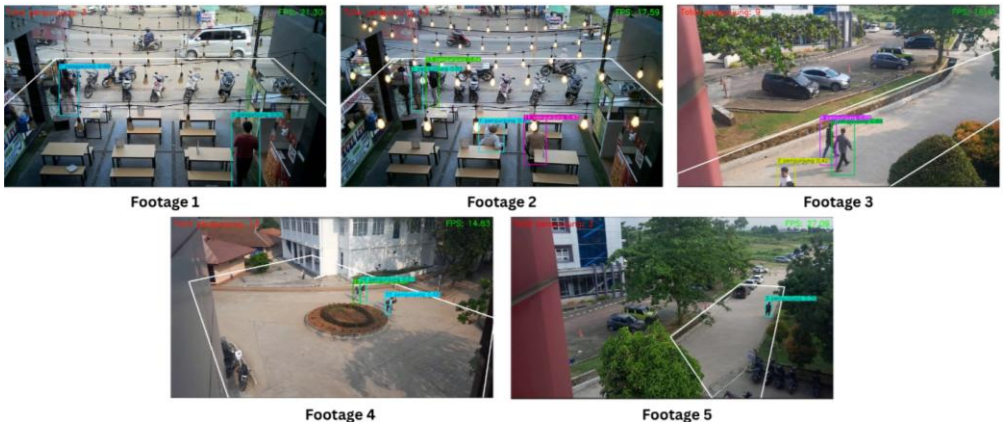
New Track Threshold	0.85
Track Buffer	900
Match Thresh	0.80
Fuse Score	True
GMC Method	sparseOptFlow
Proximity Threshold	0.60
Appearance Threshold	0.30
Re-ID	True
Model Re-ID	YOLO11n-cls

Berdasarkan Tabel 4.9, didapati bahwa nilai *Track High Threshold, Track Low Threshold, dan New Track Threshold* secara berturut-turut adalah 0.50, 0.39, dan 0.85. Ketiga nilai ini merupakan batas-batas yang dijelaskan pada Gambar 3.2, batas atas (*Track High Threshold*) merupakan nilai batas tracker untuk memberikan second association kepada identitas yang terlacak. Sedangkan rentang antara batas atas (*Track High Threshold*) dan batas bawah (*Track Low Threshold*) merupakan kondisi tracker ketika objek tersebut gagal diberikan *First Association*, sehingga *tracker* mengurangi batas agar semakin mempermudah pemberian ID pada objek tersebut. Jikalau kedua metode tersebut gagal dilakukan oleh *tracker*, maka *tracker* akan mengaktifkan pemberian ID baru yang disebut sebagai *Second Association* dengan batas *New Track Threshold*. Batas atas diatur dengan nilai 0.50 sedangkan batas bawah diberikan nilai 0.39, hal ini bertujuan agar *tracker* tidak terlalu *overfitting* ketika melacak, karena jikalau batas atas diatur terlalu tinggi, maka menyebabkan *tracker* memberikan ID yang baru secara terus-menerus kepada ID yang seharusnya tetap. Sedangkan batas bawah diberikan nilai 0.39, karena model YOLOv11 diatur dengan *Confidence Score* sebesar 0.262 seperti yang terdapat pada Gambar 4.7, akan tetapi ditingkatkan sedikit karena berdasarkan percobaan nilai terlalu rendah mengakibatkan permasalahan ID *switch*, yakni *tracker* memberikan ID yang baru secara berlebihan kepada ID yang seharusnya tetap.

4.6.2. Hasil Perhitungan Sistem

Langkah terakhir yang dilakukan pada penelitian ini adalah dengan menerapkan sistem langsung pada beberapa rekaman *footage* kasus nyata, tidak lagi

dengan menggunakan dataset seperti yang sebelumnya. Terdapat 5 rekaman yang diambil pada penelitian kali ini, kelimanya sudah merupakan campuran dari berbagai situasi dan kondisi yang berbeda. Rekaman-rekaman tersebut sudah termasuk kondisi *indoor* dan *outdoor*, serta sudah termasuk yang minim *occlusion* hingga banyak *occlusion*. Sebelum dilakukan *tracking*, kelima *footage* diberi daerah penanda untuk daerah khusus dilakukannya *tracking*, daerah tersebut dinamakan *region points* atau disebut juga sebagai *Region of Interest* (ROI). Hal ini bertujuan agar hanya daerah yang diberi ROI saja yang menghitung pengunjung dan mengabaikan objek yang berada di luar ROI. Contoh hasil penerapan RoI dan sistem penghitung pada video rekaman atau pun *real-time* yang digunakan pada penelitian kali ini dapat dilihat seperti pada Gambar 4.8.



Gambar 4. 8 Contoh Hasil Sistem Penghitung Jumlah Pengunjung

Berdasarkan Gambar 4.8, diberikan masing-masing *Region Of Interest* (ROI) secara poligon yang berbeda-beda pada masing-masing rekaman video. Hal ini dikarenakan lokasi yang berbeda, sudut yang berbeda, dan oklusi yang berbeda pada masing-masing rekaman video. Setelah dilakukan pengamatan dengan menggunakan model yang sudah dipilih, yakni YOLO11n, dan algoritma *tracker* yang sudah dikonfigurasi, maka akan dihasilkan nilai-nilai seperti yang tertera pada Tabel 4.10.

Tabel 4. 10 Tabel Hasil *Tracking* Pada Rekaman

Footage	**Total ID *Real***	**Total ID Terdeteksi**	***ID Switch***	**Total Objek Terdeteksi**	**% *Error* IDS**
Footage-1	6	11	5	6	45.45%

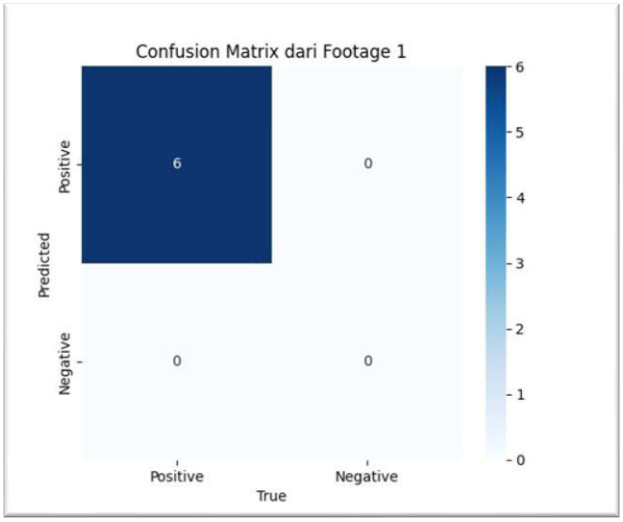
Footage-2	13	19	6	13	31.57%
Footage-3	15	11	1	10	9.09%
Footage-4	39	16	1	15	6.25%
Footage-5	19	6	1	6	16.67%

Berdasarkan Tabel 4.10, didapatkan bahwa terdapat nilai persentase *error* IDS untuk *footage* pertama, kedua, ketiga, keempat, dan kelima secara berturut-turut adalah 45.45%, 31.57%, 9.09%, 6.25%, dan 16.67%%. Nilai dari persentase *error* IDS menunjukkan seberapa besar kegagalan *tracker* untuk tetap mempertahankan ID yang tetap pada objek yang sama. Pada hasil Tabel 4.10 didapati bahwa pada *footage* pertama dan kedua terdapat persentase *error* IDS yang cukup besar yakni 45.45% pada *footage* pertama dan 31.57% pada *footage* kedua. Hal ini terjadi karena pada kedua *footage* tersebut berada pada tempat yang sama dan hanya waktu serta pencahayaan saja yang berbeda.

Lokasi pada *footage* pertama dan kedua adalah di kafe, pada rekaman tersebut terdapat banyak oklusi, yakni lampu bohlam yang digantung, sehingga sering terjadi objek pengunjung terhalang oleh lampu tersebut. Oklusi ini menyebabkan model kesulitan untuk mendeteksi pengunjung, sehingga karena model kesulitan saat mendeteksi secara konsisten pada objek, maka *tracker* juga memiliki performa yang buruk, karena setiap deteksi baru pada model akan mengakibatkan *tracker* melakukan *second association* atau pemberian ID baru meskipun terhadap orang yang sama.

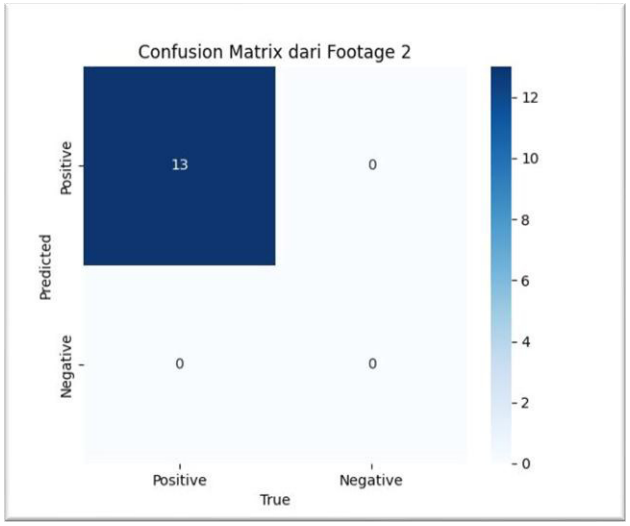
Selain mendapatkan nilai-nilai seperti yang tertera pada Tabel 4.10, untuk dapat menganalisis sistem secara lebih mendalam dapat menggunakan *Confusion Matrix*. *Confusion* Matrix berguna untuk memetakan nilai-nilai *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN) dalam bentuk matriks 2x2. Nilai-nilai dari TP, FP, FN, dan FN dihitung secara manual pada masing-masing rekaman video. Nilai TP diambil ketika model berhasil mendeteksi dengan benar, kemudian nilai FP didapatkan ketika model mendeteksi namun salah dalam mengklasifikasikan kelas objek. Lalu, nilai FN didapatkan ketika model gagal mendeteksi objek yang seharusnya dideteksi, dan terakhir adalah nilai TN yakni ketika model tidak mendeteksi yang seharusnya memang tidak terdeteksi.

Kemudian nilai-nilai tersebut digunakan untuk mencari metrik seperti *Precision*, *Recall*, dan skor F1. Dan berikut ini adalah *Confusion Matrix* dari masing-masing *footage* yang didapati secara perhitungan manual pada pemrosesan video secara *real-time*. Hasil dari *Confusion Matrix* untuk *footage* pertama dapat dilihat seperti pada Gambar 4.9.



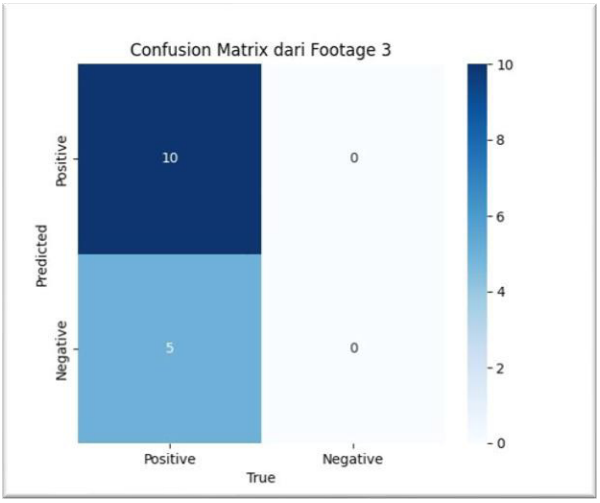
Gambar 4. 9 *Confusion Matrix* pada *Footage* 1

Berdasarkan Gambar 4.9, didapati bahwa nilai TP sebanyak 6, FP sebanyak 0, FN sebanyak 0, dan TN sebanyak 0. Maka dari itu, didapati nilai *Precision* sebesar 1 atau 100%, nilai *Recall* didapatkan sebesar 1 atau 100%. Dan skor F1 dari *footage* pertama ini adalah 1 atau 100% juga. Hasil dari *Confusion Matrix* untuk *footage* kedua dapat dilihat seperti pada Gambar 4.10.



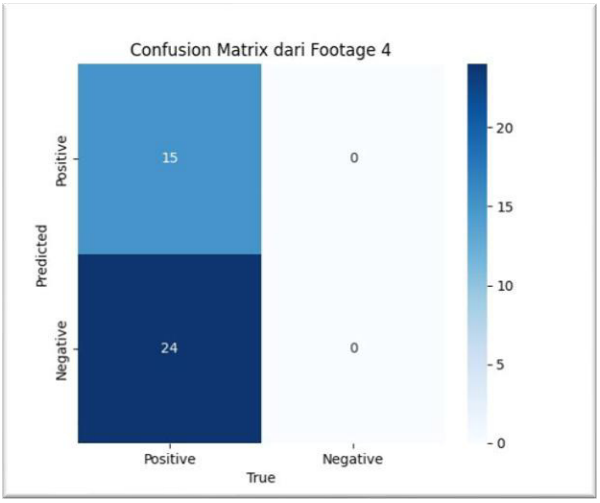
Gambar 4. 10 *Confusion Matrix* pada *Footage* 2

Berdasarkan Gambar 4.10, didapati bahwa nilai TP sebanyak 13, FP sebanyak 0, FN sebanyak 0, dan TN sebanyak 0. Maka dari itu, didapati nilai *Precision* sebesar 1 atau 100%, nilai *Recall* didapatkan sebesar 1 atau 100%. Dan skor F1 dari *footage* pertama ini adalah 1 atau 100% juga. Hasil dari *Confusion Matrix* untuk *footage* ketiga dapat dilihat seperti pada Gambar 4.11.



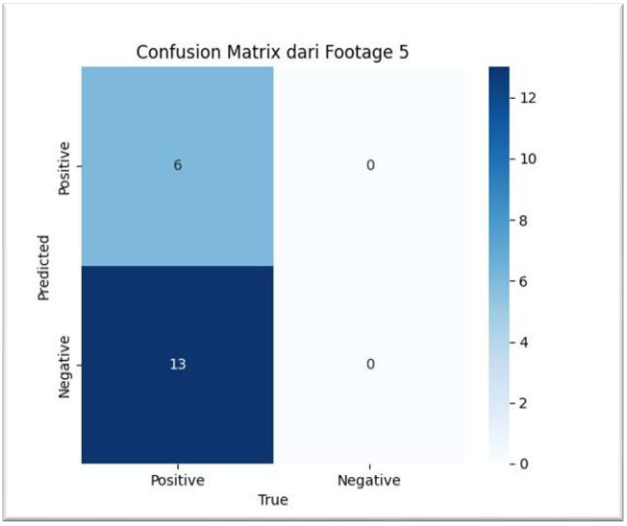
Gambar 4. 11 *Confusion Matrix* pada *Footage* 3

Berdasarkan Gambar 4.11, didapati bahwa nilai TP sebanyak 10, FP sebanyak 0, FN sebanyak 5, dan TN sebanyak 0. Maka dari itu, didapati nilai *Precision* sebesar 1 atau 100%, nilai *Recall* didapatkan sebesar 0.6667 atau 66.67%. Dan skor F1 dari *footage* pertama ini adalah 0.80 atau 80% juga. Hasil dari *Confusion Matrix* untuk *footage* keempat dapat dilihat seperti pada Gambar 4.12.



Gambar 4. 12 *Confusion Matrix* pada *Footage* 4

Berdasarkan Gambar 4.12, didapati bahwa nilai TP sebanyak 15, FP sebanyak 0, FN sebanyak 24, dan TN sebanyak 0. Maka dari itu, didapati nilai *Precision* sebesar 1 atau 100%, nilai *Recall* didapatkan sebesar 0.3846 atau 38.46%. Dan skor F1 dari *footage* pertama ini adalah 0.5556 atau 55.56% juga. Hasil dari *Confusion Matrix* untuk *footage* kelima dapat dilihat seperti pada Gambar 4.13.



Gambar 4. 13 *Confusion Matrix* pada *Footage* 5

Berdasarkan Gambar 4.13, didapati bahwa nilai TP sebanyak 6, FP sebanyak 0, FN sebanyak 13, dan TN sebanyak 0. Maka dari itu, didapati nilai *Precision* sebesar 1 atau 100%, nilai *Recall* didapatkan sebesar 0.3158 atau 31.58%. Dan skor F1 dari *footage* pertama ini adalah 0.48 atau 48% juga. Setelah dilakukan pengamatan terhadap kelima *footage*, maka didapatkan ringkasan hasil seperti yang terdapat pada Tabel 4.11.

Tabel 4. 11 Tabel Ringkasan Hasil Pengamatan Pada Rekaman

	***Footage* 1**	***Footage* 2**	***Footage* 3**	***Footage* 4**	***Footage* 5**	**Rata-rata**
% IDS	45.45%	31.57%	9.09%	6.25%	16.67%	21.81%
Precision	100%	100%	100%	100%	100%	100.00%
Recall	100%	100%	66.67%	38.64%	31.58%	67.38%
F1 Score	100%	100%	80%	55.56%	48%	76.71%

Berdasarkan Tabel 4.11, didapatkan bahwa nilai rata-rata dari seluruh *footage* pada metrik % *Error ID Switch*, *Precision*, *Recall*, dan Skor F1 secara berturut-turut adalah 21.81%, 100%, 67.38%, dan 76.71%. Nilai persentase *error ID Switch* sebesar 21.81% menandakan sistem yang dibentuk tidak memiliki banyak kesalahan pada proses pemberian ID unik kepada semua objek yang terdeteksi. Kemudian didapatkan nilai *precision* yang sangat maksimal, yakni 100%, hal ini dikarenakan model hanya diatur hanya mendeteksi satu kelas saja, yakni orang, sehingga tidak ada lagi kemungkinan model salah mendeteksi pengunjung dengan kelas lain. Sedangkan nilai metrik *recall* berhasil mencapai 67.38%, hal ini menandakan bahwa sistem berhasil membentuk *bounding box* pada orang dengan keakurasian 67.38%. Kemudian, didapatkan skor F1 sebesar 76.71% yang menandakan sistem memiliki keharmonisan antara *precision* dan *recall* sebesar 76.71%. Dengan nilai rata-rata dari seluruh metrik pada Tabel 4.11, maka didapati bahwa sistem sudah dapat dijalankan secara optimal dan berhasil mendapatkan tingkat akurasi yang sudah mencapai hasil ekspektasi.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka didapati beberapa kesimpulan seperti berikut ini.

1. Telah berhasil dibentuknya suatu sistem pelacak dan penghitung jumlah pengunjung secara otomatis dengan menggunakan kombinasi 2 algoritma, YOLO11n sebagai detektor dan BoT-SORT sebagai *tracker*.
2. Model YOLOv11 merupakan seri terkini yang mempunyai performa mengungguli seri-seri sebelumnya yang diuji pada penelitian ini. Versi terendah pada YOLOv11 mampu menghasilkan performa deteksi yang diharapkan.
3. Berdasarkan hasil penelitian, algoritma MOT dari BoT-SORT mengungguli ByteTrack dari segi metrik HOTA dan IDF1. Selain itu, karena BoT-SORT memiliki 2 fitur tambahan yang tidak dimiliki oleh *tracker* ByteTrack, yakni *Global Motion Compensation* dan *Re-Identification*.
4. Selama pengujian dengan menggunakan 5 *footage*, kombinasi dari YOLO11n dan BoT-SORT menghasilkan nilai rata-rata persentase IDS, *Precision*, *Recall*, *F1 Score* secara berturut-turut adalah 21.81%, 100%, 67.38, dan 76.71%.

5.2. Saran

Adapun terdapat beberapa saran yang dapat diberikan untuk dapat mengembangkan penelitian seperti berikut ini.

1. Untuk mendapatkan performa yang lebih baik, dapat menggunakan varian YOLO11 yang lebih kompleks, seperti YOLO11s, YOLO11m, YOLO11l, dan YOLO11x.
2. Menggunakan *hardware* GPU yang *support* terhadap CUDA (seperti GPU NVIDIA) untuk mempercepat proses *training*, evaluasi, hingga pemrosesan video *real-time*.
3. Menggunakan *Re-ID* model yang lebih kompleks dan melakukan *training* pada model *Re-ID* untuk memaksimalkan performa *tracking* secara *real-time*.

4. Pengintegrasian menggunakan *Mini Computer* atau *Cloud Computing* dengan kamera yang terpasang pada suatu tempat agar dapat digunakan untuk menghitung jumlah pengunjung secara otomatis sebagai bahan penelitian selanjutnya.

DAFTAR PUSTAKA

[1] D. Indra, H. Herman, dan F. S. Budi, "Implementasi Sistem Penghitung Kendaraan Otomatis Berbasis Computer Vision," *Komputika : Jurnal Sistem Komputer*, vol. 12, no. 1, hlm. 53–62, Mei 2023, doi: 10.34010/komputika.v12i1.9082.

[2] Z. Munawar *dkk.*, *VISI KOMPUTER: Konsep, Metode, dan Aplikasi*. Bandung: Kaizen Media Publishing, 2023.

[3] S. Xu, J. Wang, W. Shou, T. Ngo, A.-M. Sadick, dan X. Wang, "Computer Vision Techniques in Construction: A Critical Review," *Archives of Computational Methods in Engineering*, vol. 28, no. 5, hlm. 3383–3397, 2021, doi: 10.1007/s11831-020-09504-3.

[4] J. Shanahan, *Introduction to Computer Vision and Realtime Deep Learning-based Object Detection*. 2020. doi: 10.1145/3340531.3412177.

[5] P. Jiang, D. Ergu, F. Liu, Y. Cai, dan B. Ma, "A Review of Yolo Algorithm Developments," dalam *Procedia Computer Science*, Elsevier B.V., 2021, hlm. 1066–1073. doi: 10.1016/j.procs.2022.01.135.

[6] W. Wu dan J. Lai, "Multi Camera Localization Handover Based on YOLO Object Detection Algorithm in Complex Environments," *IEEE Access*, vol. 12, hlm. 15236–15250, 2024, doi: 10.1109/ACCESS.2024.3357519.

[7] N. Wojke, A. Bewley, dan D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," Mar 2017, [Daring]. Tersedia pada: http://arxiv.org/abs/1703.07402

[8] R. Pereira, G. Carvalho, L. Garrote, dan U. J. Nunes, "Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics," *Applied Sciences (Switzerland)*, vol. 12, no. 3, Feb 2022, doi: 10.3390/app12031319.

[9] K. Sinha, R. Rashmi, dan S. Anand, "International Journal of Allied Practice, Research and Review VISITOR COUNTER," *UGC JOURNAL NUMBER 44557 IJAPRR International Peer Reviewed Refereed Journal*, hlm. 35–40, 2018, [Daring]. Tersedia pada: www.ijaprr.com

[10] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, dan K. Schindler, “MOT16: A Benchmark for Multi-Object Tracking,” Mar 2016, [Daring]. Tersedia pada: http://arxiv.org/abs/1603.00831

[11] Y. Yuan dan Y. Liu, “BEVEFNet: A Multiple Object Tracking Model Based on LiDAR-Camera Fusion,” dalam *Procedia Computer Science*, Elsevier B.V., 2024, hlm. 560–567. doi: 10.1016/j.procs.2024.08.106.

[12] S. P. K. Reddy, J. Harikiran, dan B. S. Chandana, “Deep CNN Based Multi Object Detection and Tracking in Video Frames with Mean Distributed Feature Set,” dalam *Procedia Computer Science*, Elsevier B.V., 2024, hlm. 723–734. doi: 10.1016/j.procs.2024.04.069.

[13] A. Srazhdinova, A. ’ Ahmetova, dan S. Anvarov, “Detection and tracking people in real-time with YOLO object detector,” doi: 10.31643/2020.010.

[14] S. I. Cho dan S. J. Kang, “Real-Time People Counting System for Customer Movement Analysis,” *IEEE Access*, vol. 6, hlm. 55264–55272, 2018, doi: 10.1109/ACCESS.2018.2872684.

[15] Z. Liu, Y. Chen, B. Chen, L. Zhu, D. Wu, dan G. Shen, “Crowd Counting Method Based on Convolutional Neural Network with Global Density Feature,” *IEEE Access*, vol. 7, hlm. 88789–88798, 2019, doi: 10.1109/ACCESS.2019.2926881.

[16] R. Iguernaissi, D. Merad, K. Aziz, dan P. Drap, “People tracking in multi-camera systems: a review,” *Multimed Tools Appl*, vol. 78, no. 8, hlm. 10773–10793, Apr 2019, doi: 10.1007/s11042-018-6638-5.

[17] T. Sandeepanie dan S. Fernando, “Identification and Tracking of Groups of People using Object Detection and Object Tracking,” *International Journal on Advances in ICT for Emerging Regions (ICTer)*, vol. 16, no. 1, hlm. 12–21, Jun 2023, doi: 10.4038/icter.v16i1.7259.

[18] H. Matos dan H. Santos, “People Tracking in a Smart Campus context using Multiple Cameras,” 2023. [Daring]. Tersedia pada: https://www.linkedin.com/in/henriquematos99/

[19] S. Sakaguchi, M. Amagasaki, M. Kiyama, dan T. Okamoto, “Multi-Camera People Tracking with Spatio-Temporal and Group Considerations,” *IEEE*

Access, vol. 12, hlm. 36066–36073, 2024, doi: 10.1109/ACCESS.2024.3371860.

[20] T. Habara dan R. Kojima, "Floor-Field-Guided Neural Model for Crowd Counting," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3483252.

[21] A. Zein, "Kecerdasan Buatan Dalam Hal Otomatisasi Layanan," *Jurnal Ilmu Komputer*, vol. 4, no. 2, hlm. 16–25, Des 2021.

[22] U. Muzakir, Baharuddin, A. Manuhutu, dan H. Widoyo, "PENERAPAN KECERDASAN BUATAN DALAM SISTEM INFORMASI TINJAUAN LITERATUR TENTANG APLIKASI, ETIKA, DAN DAMPAK SOSIAL," *Jurnal Review Pendidikan dan Pengajaran*, vol. 6, no. 4, hlm. 1163–1169, Okt 2023.

[23] R. D. Yogaswara, "ARTIFICIAL INTELLIGENCE SEBAGAI PENGGERAK INDUSTRI 4.0 DAN TANTANGANNYA BAGI SEKTOR PEMERINTAH DAN SWASTA. ARTIFICIAL INTELLIGENCE AS AN ACTIVATOR FOR INDUSTRY 4.0 AND ITS CHALLENGES FOR GOVERNMENT AND PRIVATE SECTORS," *Jurnal Masyarakat Telematika dan Informasi*, vol. 10, no. 1, hlm. 67–72, Agu 2019, [Daring]. Tersedia pada: http://www.mplsvpn.info/2017/11/what-is-neuron-and-

[24] N. Putu Wulantari *dkk.*, "The Role Of Gamification In English Language Teaching: A Literature Review," *Journal on Education*, vol. 6, no. 1, hlm. 2847–2856, 2023.

[25] F. Hasyim, K. Malik, F. Rizal, U. Nurul Jadid, R. Artikel, dan K. Kunci Batik, "Implementasi Algoritma Convolutional Neural Networks (CNN) Untuk Klasifikasi Batik," *COREAI: Jurnal Kecerdasan Buatan, Komputasi dan Teknologi Informasi*, hlm. 40–47, Nov 2021, [Daring]. Tersedia pada: https://ejournal.unuja.ac.id/index.php/core

[26] A. Octaviani dan P. Dewi, "Kecerdasan Buatan sebagai Konsep Baru pada Perpustakaan," *ANUVA*, vol. 4, no. 4, hlm. 453–460, 2020.

[27] S. Maihani *dkk.*, "PERAN KECERDASAN BUATAN ARTIFICIAL INTELLIGENCE (AI) DALAM INOVASI PEMASARAN," *Jurnal Warta Dharmawangsa*, vol. 17, no. 4, hlm. 1651–1661, Okt 2023.

[28] E. Susanto, “Analisis Implementasi Kecerdasan Buatan Dalam Pembelajaran,” *Sindoro Cendikia Pendidikan*, vol. 1, no. 8, hlm. 101–112, 2023.

[29] D. Manongga, U. Rahardja, I. Sembiring, N. Lutfiani, dan A. B. Yadila, “Dampak Kecerdasan Buatan Bagi Pendidikan,” *ADI Bisnis Digital Interdisiplin Jurnal*, vol. 3, no. 2, hlm. 110–124, Nov 2022, doi: 10.34306/abdi.v3i2.792.

[30] T. Merentek, E. Usoh, dan J. Lengkong, “Implementasi Kecerdasan Buatan ChatGPT dalam Pembelajaran,” *Jurnal Pendidikan Tambusai*, vol. 7, no. 3, hlm. 26862–26869, 2023.

[31] F. V. Jedrzejewski, L. Thode, J. Fischbach, T. Gorschek, D. Mendez, dan N. Lavesson, “Adversarial Machine Learning in Industry: A Systematic Literature Review,” *Comput Secur*, vol. 145, Okt 2024, doi: 10.1016/j.cose.2024.103988.

[32] Q. Wang, Z. Chen, Y. Zhou, Z. Liu, dan Z. Peng, “Real-time monitoring and optimization of machine learning intelligent control system in power data modeling technology,” *Machine Learning with Applications*, vol. 18, hlm. 100584, Des 2024, doi: 10.1016/j.mlwa.2024.100584.

[33] E. Erwin *dkk.*, *TRANSFORMASI DIGITAL*. PT. Sonpedia Publishing Indonesia, 2023. [Daring]. Tersedia pada: https://www.researchgate.net/publication/379374858

[34] H. S. Le *dkk.*, “Predictive model for customer satisfaction analytics in E-commerce sector using machine learning and deep learning,” *International Journal of Information Management Data Insights*, vol. 4, no. 2, Nov 2024, doi: 10.1016/j.jjimei.2024.100295.

[35] Andrew NG, *Machine Learning Yearning*. DeepLearning.AI, 2018.

[36] M. S. Lubis, “IMPLEMENTASI ARTIFICIAL INTELLIGENCE PADA SYSTEM MANUFAKTUR TERPADU,” *Prosiding Seminar Nasional Teknik UISU (SEMNASTEK)*, hlm. 1, 2021.

[37] T. M. N. U. Akhund dan W. M. Al-Nuwaiser, “Improving Prediction Efficiency of Machine Learning Models for Cardiovascular Disease in IoST-Based Systems through Hyperparameter Optimization,” *Computers,*

Materials and Continua, vol. 80, no. 3, hlm. 3485–3506, 2024, doi: 10.32604/cmc.2024.054222.

[38] S. Abolmakarem, F. Abdi, K. Khalili-Damghani, dan H. Didehkhani, "A multi-stage machine learning approach for stock price prediction: Engineered and derivative indices," *Intelligent Systems with Applications*, vol. 24, hlm. 200449, Des 2024, doi: 10.1016/j.iswa.2024.200449.

[39] K. Sadeghi, A. Banerjee, dan S. K. S. Gupta, "A System-Driven Taxonomy of Attacks and Defenses in Adversarial Machine Learning," *IEEE Trans Emerg Top Comput Intell*, vol. 4, no. 4, hlm. 450–467, 2020, doi: 10.1109/TETCI.2020.2968933.

[40] A. E. Cinà *dkk.*, "Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning," *ACM Comput Surv*, vol. 55, no. 13s, hlm. 1–39, Jul 2023, doi: 10.1145/3585385.

[41] S. Jawed, I. Faye, dan A. S. Malik, "Deep Learning-Based Assessment Model for Real-Time Identification of Visual Learners Using Raw EEG," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 32, hlm. 378–390, 2024, doi: 10.1109/TNSRE.2024.3351694.

[42] J. Guo, W. Cao, B. Nie, dan Q. Qin, "Unsupervised Learning Composite Network to Reduce Training Cost of Deep Learning Model for Colorectal Cancer Diagnosis," *IEEE J Transl Eng Health Med*, vol. 11, hlm. 54–59, 2023, doi: 10.1109/JTEHM.2022.3224021.

[43] A. Craik, Y. He, dan J. L. Contreras-Vidal, "Deep learning for electroencephalogram (EEG) classification tasks: A review," 2019, *Institute of Physics Publishing*. doi: 10.1088/1741-2552/ab0ab5.

[44] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[45] K. Bera, K. A. Schalper, D. L. Rimm, V. Velcheti, dan A. Madabhushi, "Artificial intelligence in digital pathology — new tools for diagnosis and precision oncology," *Nat Rev Clin Oncol*, vol. 16, no. 11, hlm. 703–715, 2019, doi: 10.1038/s41571-019-0252-y.

[46] S. Kuntz *dkk.*, "Gastrointestinal cancer classification and prognostication from histology using deep learning: Systematic review," *Eur J Cancer*, vol. 155, hlm. 200–215, Sep 2021, doi: 10.1016/j.ejca.2021.07.012.

[47] Q. Zhu dan X. Zu, "Fully Convolutional Neural Network Structure and Its Loss Function for Image Classification," *IEEE Access*, vol. 10, hlm. 35541–35549, 2022, doi: 10.1109/ACCESS.2022.3163849.

[48] R. Szeliski, "Computer Vision: Algorithms and Applications," 2010. [Daring]. Tersedia pada: http://szeliski.org/Book/.

[49] A. Magdy, H. Hussein, R. F. Abdel-Kader, dan K. A. El Salam, "Performance Enhancement of Skin Cancer Classification Using Computer Vision," *IEEE Access*, vol. 11, hlm. 72120–72133, 2023, doi: 10.1109/ACCESS.2023.3294974.

[50] A. Ulhaq, J. Born, A. Khan, D. P. S. Gomes, S. Chakraborty, dan M. Paul, "COVID-19 control by computer vision approaches: A survey," 2020, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2020.3027685.

[51] M. Raisul Islam *dkk.*, "Deep Learning and Computer Vision Techniques for Enhanced Quality Control in Manufacturing Processes," *IEEE Access*, vol. 12, hlm. 121449–121479, 2024, doi: 10.1109/ACCESS.2024.3453664.

[52] D. Reis, J. Kupec, J. Hong, dan A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," Mei 2023, [Daring]. Tersedia pada: http://arxiv.org/abs/2305.09972

[53] A. Vina, "Ultralytics' 2024 highlights: Driving innovation in Vision AI."

[54] R. Khanam dan M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," Okt 2024, [Daring]. Tersedia pada: http://arxiv.org/abs/2410.17725

LAMPIRAN

```
!pip install ultralytics

from ultralytics import YOLO

models = [YOLO("yolov8n.pt"),
          YOLO("yolov9t.pt"),
          YOLO("yolov10n.pt"),
          YOLO("yolo11n.pt")]

results = dict()

for model in models:
  metrics = model.val(data='coco128.yaml', imgsz=640,
epochs=100, save=True, plots=True, verbose=False)

  model_name = model.model_name.split('.')[0]
  result     = metrics.results_dict
  precision, recall, map50, map5095, fitness = result.values()

  results[model_name] = dict()
  results[model_name]["precision"]  = float(round(precision, 3))
  results[model_name]["recall"]     = float(round(recall, 3))
  results[model_name]["map50"]      = float(round(map50, 3))
  results[model_name]["map5095"]    = float(round(map5095, 3))
  results[model_name]["fitness"]    = float(round(fitness, 3))

for model in results:
  print(model)

  for metric in results[model]:
    print(f"{metric}: {results[model][metric]}")
  print("=" * 20)
```

```
!pip install ultralytics
!pip install roboflow

from ultralytics import YOLO
```

```
from roboflow import Roboflow

models = [YOLO("yolov8n.pt"),
          YOLO("yolov9t.pt"),
          YOLO("yolov10n.pt"),
          YOLO("yolo11n.pt")]

rf = Roboflow(api_key="GSwlztLrfT4RvpTyPmjx")
project = rf.workspace("myproject-cc5hp").project("people-
tracking-and-counting")
version = project.version(3)
dataset = version.download("yolov11")

dataset_path = 'People Tracking and Counting'

results = dict()

for model in models:
  metrics = model.val(data='coco128.yaml', imgsz=640,
epochs=100, save=True, plots=True, verbose=False)

  model_name = model.model_name.split('.')[0]
  result     = metrics.results_dict
  precision, recall, map50, map5095, fitness = result.values()

  results[model_name] = dict()
  results[model_name]["precision"]  = float(round(precision, 3))
  results[model_name]["recall"]     = float(round(recall, 3))
  results[model_name]["map50"]      = float(round(map50, 3))
  results[model_name]["map5095"]    = float(round(map5095, 3))
  results[model_name]["fitness"]    = float(round(fitness, 3))

for model in results:
  print(model)

  for metric in results[model]:
    print(f"{metric}: {results[model][metric]}")

  print("=" * 20)
```

```
!pip install ultralytics
!pip install roboflow

import pandas as pd
import numpy as np
import os
import ultralytics as ut

from roboflow import Roboflow
from ultralytics import YOLO

rf = Roboflow(api_key="GSwlztLrfT4RvpTyPmjx")
project = rf.workspace("myproject-cc5hp").project("people-
tracking-and-counting")
version = project.version(3)
dataset = version.download("yolov11")

rf_dataset_yaml = "/content/People-Tracking-and-Counting-
3/data.yaml"

models = [YOLO("yolo11n.pt"),
          YOLO("yolo11s.pt"),
          YOLO("yolo11m.pt"),
          YOLO("yolo11l.pt"),
          YOLO("yolo11x.pt")]

output_dir = "/content/trained_results"

os.makedirs(output_dir, exist_ok=True)

for model in models:
  model.train(data=rf_dataset_yaml, epochs=100, imgsz=640,
batch=16, device='cuda', workers=12, project=output_dir,
name=model.model_name, plots=True)
```

```
import pandas as pd
import numpy as np
import os
```

```
import time

from ultralytics import YOLO

DATASET_DIR       = "/content/drive/MyDrive/Colab
Notebooks/[Skripsi] People Tracking with YOLO and Deep
SORT/dataset"
MOT_VER           = "MOT20"
MOT_DIR           = os.path.join(DATASET_DIR, MOT_VER)
TRAIN_DIR         = os.path.join(MOT_DIR, 'train')
TRAIN_SEQUENCES   = sorted(os.listdir(TRAIN_DIR))
TRAIN_SEQ_DIRS    = [os.path.join(TRAIN_DIR, seq) for seq in
TRAIN_SEQUENCES]
TRAIN_IMG_DIRS    = [os.path.join(train_seq_dir, "img1") for
train_seq_dir in TRAIN_SEQ_DIRS]
IMG_EACH_SEQ      = [os.listdir(train_img_dir) for train_img_dir
in TRAIN_IMG_DIRS]

IMGS_SEQUENCES   = dict()
for i in range(len(TRAIN_IMG_DIRS)):
  IMGS_SEQUENCES[TRAIN_SEQUENCES[i]] = list()
  for img in IMG_EACH_SEQ[i]:
    img_path = os.path.join(TRAIN_IMG_DIRS[i], img)
    IMGS_SEQUENCES[TRAIN_SEQUENCES[i]].append(img_path)

def extract_results(source, model, tracker, mot_ver, seq_names):
  results_dict = dict()
  time_dict = dict()

  tracker_name = tracker.split(".")[0]

  start = time.time()

  for i, seq_dir in enumerate(source):
      mot_results = []

      for j, img in enumerate(os.listdir(seq_dir)):
          frame = j + 1
```

```
          img_path = os.path.join(seq_dir, img)
          results = model.track(img_path, tracker=tracker,
conf=0.20, save=False, show_labels=False, show_conf=False,
show_boxes=False, verbose=False)

          id = results[0].boxes.id
          conf = results[0].boxes.conf

          xyxy = results[0].boxes.xyxy
          x1, y1 = xyxy[:, 0], xyxy[:, 1]

          xywh = results[0].boxes.xywh
          w, h = xywh[:, 2], xywh[:, 3]

          if id is not None:
            for k in range(len(id)):
              mot_results.append([frame, id[k], x1[k], y1[k],
w[k], h[k], conf[k], -1, -1, -1])

      results_dict[seq_names[i]] = mot_results

  end = time.time()

  processing_speed = end - start
  time_dict[tracker_name] = processing_speed

  print(f"Processing Time {mot_ver}-{tracker_name}:
{processing_speed} seconds")

  return results_dict

def convert_to_df(results):
  mots = dict()

  for i in results:
    df = pd.DataFrame(results[i]).astype(np.float64)
    mots[i] = df

  return mots
```

```
def save_results(mots, tracker, split, model, mot_ver):
  print(f"Saving Results")

  model_name = model.model_name.split(".")[0]
  tracker_name = tracker.split(".")[0]

  mot_dir = mot_ver
  model_directory = os.path.join(mot_dir, model_name)
  tracker_directory = os.path.join(model_directory,
tracker_name)
  output_directory = os.path.join("/content", tracker_directory,
split)

  os.makedirs(output_directory, exist_ok=True)

  for mot in mots:
    mots[mot].to_csv(f"{output_directory}/{mot}.csv",
index=False, header=False)
    print(f"Result saved to: {output_directory}/{mot}.csv")
    print(f"=" * 50)

def extract_convdf_save(source, model, tracker, seq_names,
mot_ver, split):
  results_dict = extract_results(source, model, tracker,
mot_ver, seq_names)
  mots = convert_to_df(results_dict)
  save_results(mots, tracker, split, model, mot_ver)

model_path  = "/content/drive/MyDrive/Colab Notebooks/[Skripsi]
People Tracking with YOLO and Deep
SORT/models/yolo11n/weights/best.pt"
model       = YOLO(model_path)
model.fuse()

trackers  = ["botsort.yaml", "bytetrack.yaml"]
sources   = TRAIN_IMG_DIRS
seq_names = TRAIN_SEQUENCES
```

```
for tracker in trackers:
  extract_convdf_save(sources, model, tracker, seq_names,
"MOT20", "train")
```

```
import ultralytics as ut
import math
import time
import cv2
import os

from ultralytics import solutions

footages_dir =
"path_menuju_direktori_penyimpanan_seluruh_footage"
footage_1 = dict(path = "path_menuju_footage_1", region = [(211,
208), (0, 398), (0, 720), (1280, 720), (1280, 398), (1069,
208)])
footage_2 = dict(path = "path_menuju_footage_2", region = [(211,
208), (0, 398), (0, 720), (1280, 720), (1280, 398), (1069,
208)])
footage_3 = dict(path = "path_menuju_footage_3", region = [(0,
664), (1280, 170), (1280, 442), (835, 720), (0, 720)])
footage_4 = dict(path = "path_menuju_footage_4", region = [(73,
256), (558, 152), (1280, 400), (1234, 720), (230, 718)])
footage_5 = dict(path = "path_menuju_footage_5", region = [(634,
473), (909, 265), (1085, 276), (971, 573), (991, 720), (802,
720)])

def run_tracking(model_path, tracker_path, footage, region,
conf_score, iou_score, saving=True):
  cap = cv2.VideoCapture(footage)
  assert cap.isOpened(), "Error reading video file"

  w, h, fps = (int(cap.get(x)) for x in
(cv2.CAP_PROP_FRAME_WIDTH, cv2.CAP_PROP_FRAME_HEIGHT,
cv2.CAP_PROP_FPS))

  # Membuat Directory Khusus Hasil Tracking
  output_dir = "/content/drive/MyDrive/Track Results"
```

```
  os.makedirs(output_dir, exist_ok=True)

  print(f"Now processing {footage.split('/')[-
1].split('.')[0]}")
  print(f"Tracker used is '{tracker_path.split('/')[-
1].split('.')[0]}'")

  output_filename = f"trackzone_{footage.split('/')[-
1].split('.')[0]}_{tracker_path.split('/')[-
1].split('.')[0]}.avi"
  output_path     = os.path.join(output_dir, output_filename)

  # Video writer
  video_writer  = cv2.VideoWriter(output_path,
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h))
  region_points = region  # For rectangle region tracking

  # Menginisiasi Sistem Pelacak Track Zone
  trackzone = solutions.TrackZone(region=region_points,
model=model_path, tracker=tracker_path, conf=conf_score,
iou=iou_score, device=0, show=True)

  # Memproses Video
  total_ids = set()
  while cap.isOpened():
      success, im0 = cap.read()
      if not success:
          print("Frame tidak terdeteksi atau sudah selesai
memproses video!")
          break

      start_time = time.time()
      results    = trackzone(im0) # Melakukan Pelacakan
      end_time   = time.time()

      # Menghitung FPS
      fps = 1 / (end_time - start_time)

      plot_im = results.plot_im
```

```
      for id in trackzone.track_ids:
        total_ids.add(id)

      # Mendrawing skor FPS dan Total Pengunjung
      fps_xpos, fps_ypos = math.ceil(w - (w * 15 / 100)),
math.ceil((h - (h * 95 / 100)))
      visitor_xpos, visitor_ypos = math.ceil(w - (w * 99 /
100)), math.ceil((h - (h * 95 / 100)))

      cv2.putText(plot_im, f"FPS: {fps:.2f}", (fps_xpos,
fps_ypos), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
      cv2.putText(plot_im, f"Total pengunjung:
{len(total_ids)}", (visitor_xpos, visitor_ypos),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

      # cv2_imshow(plot_im)
      video_writer.write(plot_im)   # membuat video dari frame

  cap.release()
  video_writer.release()
  cv2.destroyAllWindows()

model_path   = "/content/drive/MyDrive/Colab Notebooks/[Skripsi]
People Tracking with YOLO and Deep
SORT/models/yolo11n/weights/v5_no_gray_ht1.pt"
footages = [footage_1, footage_2, footage_3, footage_4,
footage_5]

for footage in footages:
  run_tracking(model_path=model_path,
                   tracker_path="path_menuju_file_tracker",
                   footage=footage['path'],
                   region=footage['region'],
                   conf_score=0.27,
                   iou_score=0.50,
                   saving=True)
```

Untuk kode-kode selengkapnya dapat dilihat pada *link* GitHub berikut

GitHub: *https://github.com/dthief55/skripsi*