BAB III

PELAKSANAAN PROYEK

3.1 Kedudukan dan Koordinasi

Sebagai bagian dari tim proyek independen MBKM ini, penulis secara spesifik bertanggung jawab pada divisi pemprograman tingkat lanjut (*High-level Programming*). Peran ini bertanggung jawab dalam menentukan kecerdasan dan kemampuan otonom robot di lapangan. Tugas utama penulis meliputi aspek pemilihan dan menganalisis algoritma deteksi objek secara *real-time*.

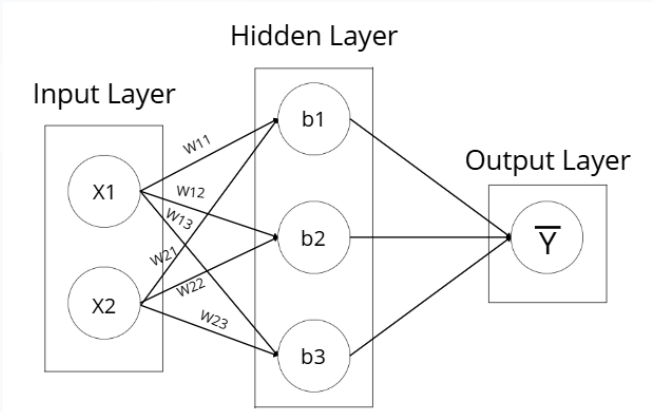
Deteksi objek atau *object detection* merupakan fitur fundamental bagi robot sepakbola beroda untuk mengidentifikasi objek-objek penting yang diperlukan seperti bola dan gawang. Untuk mencapai akurasi dan ketepatan yang dibutuhkan dalam kompetisi yang bergerak cepat, proyek ini mengimplementasikan algoritma *computer vision* yang tepat. *Computer vision* memiliki banyak algoritma dengan berbagai fungsi berdasarkan kemampuan presisi, kegunaan, kecepatan, ringan, maupun fitur-fitur lainnya. Oleh karena itu memilih algoritma yang tepat menjadi salah satu bagian yang krusial dalam mencapai deteksi objek yang akurat dan presisi.

3.2 Pendahuluan atau Dasar

Image processing merupakan salah satu fitur penting yang digunakan oleh robot sepak bola beroda, dengan fungsi untuk mendeteksi bola dan gawang. Dalam beberapa tahun terakhir *image processing* menggunakan berbagai dasar dari *machine learning* atau *deep learning* yaitu *Convolutional Neural Networks* (CNN), *Transformer*, *Histogram of Oriented Gradient* (HOG), dan masih banyak lagi yang berbasis *Neural Networks* (NN). Banyak algoritma *image processing* menggunakan dasar dari *neural network* untuk mengoptimalkan akurasi dan kecepatan proses *object detection*.

Secara teori, *Neural Networks* (NN) merupakan model komputasi yang terinspirasi dari konsep saraf manusia, terdiri dari banyak node yang mempresentasikan sebagai neuron dan saling terhubung [5]. Node-node atau

neuron ini bekerja sebagai memori dari data yang diproses melalui jaringan yang berada pada *neural network*. Struktur yang ada pada *neural network* adalah *input layer* merupakan *layer* yang berisi data input atau fitur yang ingin diproses, *hidden layer* merupakan *layer* yang biasanya terdiri dari satu atau lebih *layer* dengan fungsi untuk proses atau *training data*, dan *output layer* merupakan *layer* yang berisi hasil dari proses jaringan pada *neural network* [6].

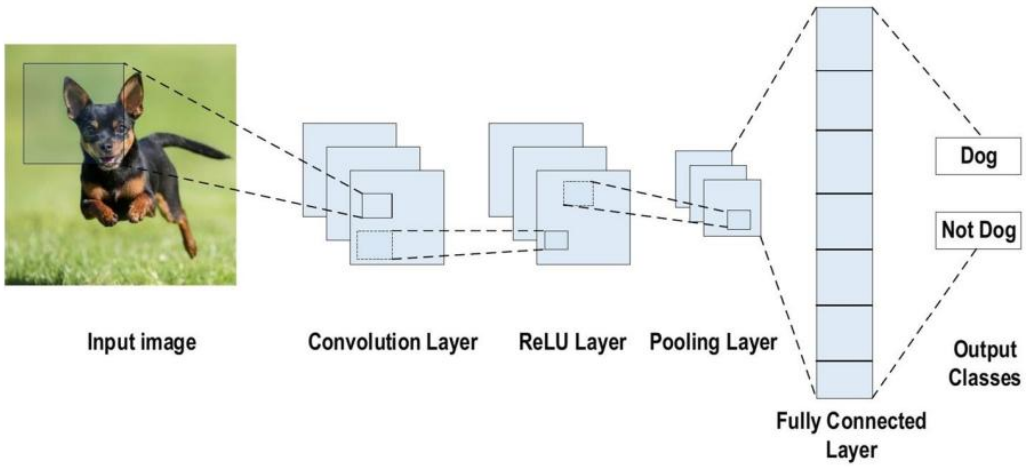


Gambar 3.1. *Neural Network*

Diatas merupakan gambaran proses *neural network* bekerja, proses latihan untuk *neural network* sendiri memiliki dua jenis yaitu *forward propagation* dan *backward propagation*. *Forward propagation* merupakan proses dari *input layer* ke *hidden layer* lalu akan mengeluarkan hasil di *output layer*. *Forward propagation* memiliki rumus $\bar{Y} = \sigma(\sum_{i=1}^{n} x_i w_i + b_i)$, Dimana *x* merupakan nilai dari input, *w* merupakan *weight* dari data untuk mementukan nilai dari hubungan antara dua node, *b* merupakan bias untuk menjadi *threshold* dan menggeser *activation function*, dan $\bar{Y}$ merupakan *output* sebagai hasil *training data*. *Backward propagation* merupakan kebalikannya dari *forward propagation* yaitu proses dari *output* ke *hidden layer* lalu ke *input layer*. Cara kerja dari *backward propagation* adalah ketika *forward propagation* dijalankan akan adanya perbedaan hasil *output* sebenarnya dengan *output* perhitungan sehingga mendapatkan *error* atau *loss function*, *error* atau *lost function* dari *output* ini akan diturunkan melalui *backward propagation* dengan menambahkan *bias* dan *weight*. Dari proses *neural network*

tersebut, banyak orang yang mengembangkan *neural network*, contohnya bisa dikembangkan menjadi *Convolutional Neural Network* (CNN) dan *Transformer*.

Convolutional Neural Network (CNN) menggunakan konsep proses neuron-neuron yang saling terhubung seperti saraf manusia dan juga mirip dengan konsep *Neural Network* (NN). Yang berbeda dari CNN adalah bisa secara otomatis mengidentifikasi fitur-fitur pada input yang relevan tanpa pengawasan manusia [7]. Struktur proses yang dimiliki CNN adalah *input layer*, *convolution layer*, *activation function layer*, *pooling layer*, *fully connected layer*, dan *output*. *Input layer* berisi *input data*, *convolution layer* berisi *kernel* yang berfungsi untuk mengekstrak fitur penting dari gambar, *activation function layer* berisi tentang model matematika non-linear sehingga jaringan bisa memperlajari hubungan kompleks, *pooling layer* berfungsi untuk mengecilkan ukuran *feature map* sambil mempertahankan informasi penting, *fully connected layer* merupakan sistem *neural network* untuk klasifikasi, dan terakhir *output layer* untuk menghasilkan *output* dari proses CNN.

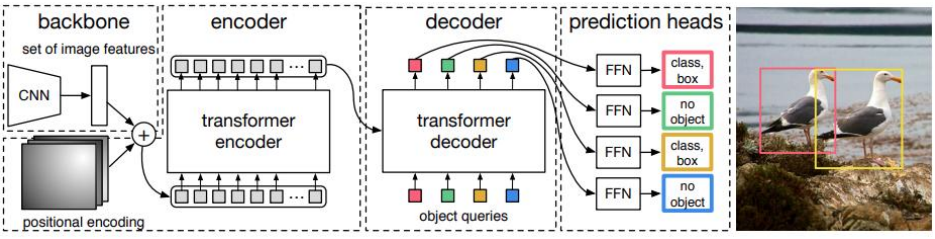


Gambar 3.2. *Convolutional Neural Networks*

Dapat dilihat dari gambar diatas merupakan gambaran besar dari proses CNN dan juga proses untuk *training* sama seperti NN karena memiliki *forward propagation* dan *backward propagation*. Proses CNN sama dengan *Neural Networks* yang membedakannya adalah CNN bisa membedakan fitur-fitur input yang berbeda-beda dalam prosesnya. *Forward propagation*-nya berawal dari menginput data lalu masuk ke *convolution layer*, dimana *matrix* dari gambar akan

difilter dengan *kernel filter*, setelah itu ada *activation function layer* dimana model matematika digunakan untuk *threshold* data setelah *convolution* [8]. Setelah ada *activation*, akan ada *pooling layer* dimana ada dua jenis *max* dan *min pooling* berguna untuk mengurangi jumlah neuron, lanjut ke *fully connected layer* berguna untuk menambahkan *layer* untuk klasifikasi gambar, dan terakhir menjadi *output* dengan hasil klasfikasi tersebut. Untuk *backward propagation* juga sama dengan *Neural Network*, setelah *forward propagation* akan adanya *loss function* lalu akan diproses mundur.

Transformer dalam *object detection* merupakan pendekatan modern yang mengadaptasi arsitektur transformer dari bidang *natural language processing* (NLP) ke dalam visi komputer. Pada awalnya, *transformer* dirancang untuk memahami hubungan antar kata dalam kalimat melalui mekanisme *self-attention*, namun kini teknologi ini berhasil diterapkan untuk memahami konteks spasial dalam gambar. *Transformer* bekerja secara *end-to-end* tanpa memerlukan komponen seperti *region proposal* atau *anchor box* seperti pada metode konvensional (misalnya *Faster* R-CNN atau YOLO). Arsitekturnya terdiri dari *backbone* (biasanya CNN seperti ResNet) untuk ekstraksi fitur, kemudian hasilnya masuk ke dalam *encoder* dan *decoder transformer*. *Encoder* bertugas memahami hubungan antar fitur dari seluruh gambar, sedangkan *decoder* menghasilkan prediksi objek menggunakan *object queries*, yang merupakan vektor-vektor pembelajaran yang mewakili kemungkinan keberadaan objek. Terakhir adalah head dimana hasil dari prediksi atau *output*.



Gambar 3.2. *Transformer*

Salah satu keunikan *Transformer* adalah proses pencocokan *output* dengan *ground truth* menggunakan *Hungarian Matching*, yang memastikan setiap prediksi

diasosiasikan secara optimal dengan target tanpa tumpang tindih (tanpa *Non-Maximum Suppression*). Untuk menghasilkan prediksi akhir, model memberikan informasi *bounding box* (koordinat posisi objek) dan label kelas objek. Walaupun hasil deteksinya sangat akurat dalam skenario dengan objek besar dan kompleks, model awal DETR memiliki kelemahan seperti waktu pelatihan yang lama dan kesulitan dalam mendeteksi objek kecil, sehingga dikembangkanlah varian seperti Deformable DETR, DINO, dan RT-DETR untuk meningkatkan efisiensi dan akurasi.

3.3 Pengembangan Algoritma *Object Detection*

Object detection merupakan salah satu fitur penting dengan basis *image processing*, pemilihan algoritma untuk mencapai akurasi dan kecepatan yang presisi menjadi hal yang krusial. Ada berbagai macam algoritma untuk *object detection* yaitu YOLO, R-CNN, *Fast* R-CNN, *Faster* R-CNN, RT-DETR, dan masih banyak lagi. Kali ini akan membahas algoritma YOLO dan RT-DETR, dimana algoritma ini cocok untuk robot sepak bola beroda karena bisa mendeteksi objek secara *real-time*.

3.3.1 Pengambilan *Dataset*

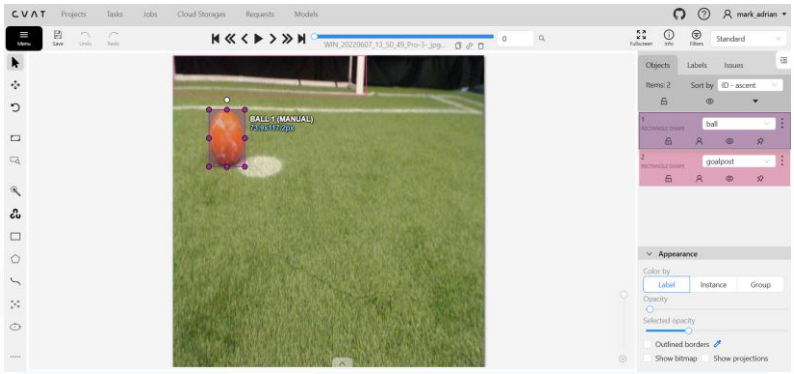
Dalam proses pengembangan algoritma *object detection*, menyediakan *dataset* yang representatif dan berkualitas merupakan proses yang penting. Pada penelitian ini, *dataset* yang dikumpulkan merupakan gambar bola dan gawang menggunakan dua sumber yang berbeda, yaitu data yang dikumpulkan secara manual dan *dataset* yang dapat diakses publik dari Roboflow. *Dataset* yang digunakan sebanyak 7094 gambar, sebanyak 4094 gambar dikumpulkan secara manual menggunakan kamera, yang mengambil gambar dengan berbagai macam kondisi pencahyaan, sudut pandang, dan posisi objek untuk meningkatkan variasi data. Sisa jumlah *dataset*nya, yaitu 3000 gambar diperoleh dari platform Roboflow, yang menyediakan *dataset* yang bisa akses publik dan *dataset* dari Roboflow dipilih berdasarkan kesesuaian objek, sekanrio pengambilan gambar, dan keberagaman *class* yang ingin dilatih.

Gambar 3.4. *Dataset* yang dikumpulkan

3.3.2 Persiapan dan Proses *Training*

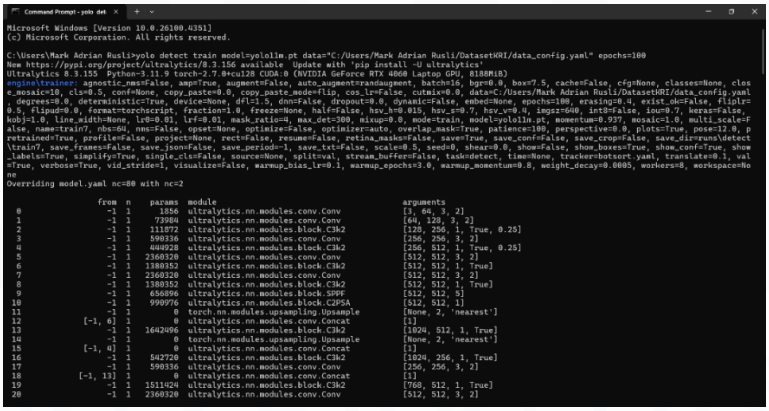
Proses *training* merupakan bagian penting dari pengembangan model *object detection*, di mana model dilatih untuk mengenali dan mengklasifikasikan objek berdasarkan data gambar yang sudah di anotasi. Pada penelitian ini, sebanyak 7094 gambar yang tekah dikumpulkan akan dibagi dua bagian. Sebanyak 4981 gambar akan digunakan sebagai data training, sedangkan 2113 gambar sisanya digunakan sebagai data validation. Pembagian ini mempertimbangkan proposi kelas agar tetap seimbang dikedua bagian subsset, diaman *training* sebesar 70.2% sedangkan *validation* 29,8%. Setelah dibagi menjadi dua bagian subset, semua gambar akan diatur ukuran nya menjadi 640 x 640 *pixel* untuk mempermudah *training model*.

Setelah di atur ukuran gambar menjadi sama, dilakukan proses anotasi menggunakan CVAT (*Computer Vision Annotation Tool*). CVAT merupakan platform berbasis web yang mendukung anotasi manual untuk berbagai format standar seperti YOLO dan COCO. Pada proses ini, setiap gambar akan diberi penandaan objek melalui *bounding box* serta label kelas sesuai dengan yang diinginkan. Anotasi berguna untuk mengetahui letak *pixel* objek yang diberi *bounding box*. Setelah itu, data akan diekspor sesuai dengan format yang diinginkan dimana penulis mengekspor dalam format *.txt.*



Gambar 3.5. *Annotation* Gambar Menggunakan CVAT

Training dilakukan menggunakan GPU CUDA, CUDA merupakan driver dari NVIDIA yang khusus untuk *deep learning* dan *machine learning* yang berintegrasi dengan Pytorch, dengan menggunakan CUDA akan menghasilkan waktu training yang lebih cepat dibandingkan menggunakan CPU. Spesifikasi GPU dan CPU PC yang digunakan adalah GPU NVIDIA GeForce RTX 4050 dan CPU AMD Ryzen 5 7535HS. Proses *training* dilakukan melalui *Command Line Interface* (CLI) yang disediakan oleh *framework* Ultralytics. Sebelum pelatihan dimulai, *file* konfigurasi disiapkan, yang berisi path direktori *dataset* dan nama-nama kelas yang digunakan. CLI untuk training berisi model algoritma, data konfigurasi, dan *hyperparameter* yang akan digunakan.

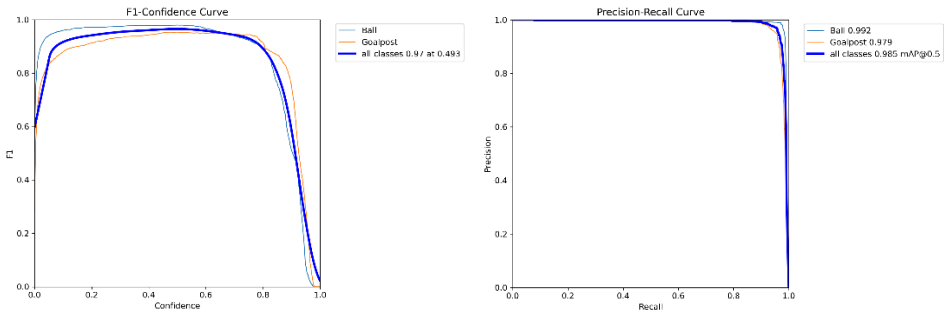


Gambar 3.6. Training Menggunakan *Command Line Interface*

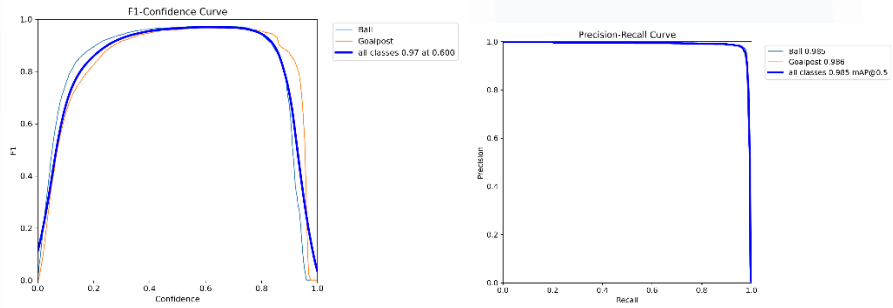
3.3.3 Perbandingan YOLO dengan RT-DETR

You Look Only Once (YOLO), merupakan arsitektur jaringan konvolusional tunggal yang secara bersamaan memprediksi banyak bounding box dan probabilitas kelas dengan training seluruh gambar dan secara langsung mengoptimalkan kinerja deteksi [9]. Sedangkan RT-DETR merupakan arsitektur yang menggunakan dasar dari arsitektur DETR yang menggunakan konsep end-to-end object detector, dimana menghilangkan komponen Non-Maximum Supression (NMS) yang bisa menyebabkan memperlambat inferensi dan mengatasi masalah DETR sebelumnya yang mempunyai computation cost yang tinggi [10]. Yang menjadi pembeda untuk YOLO dan RT-DETR adalah YOLO berdasarkan one stage detector, artinya region proposal dan object detection stage digabungkan dalam satu proses [11], [12]. Sedangkan RT-DETR berdasarkan end-to-end detector, artinya berbasis transformer tanpa proses region proposal atau pasca-pemprosesan NMS [10].

Untuk membandingkan algoritma yang dipilih sesuai dengan kebutuhan robot sepakbola beroda, akan dilakukan percobaan antara algortima YOLO dan RT-DETR dengan mengumpulkan 30 data confidence level dan *inference time* dari menguji bola dan gawang menggunakan *webcam* yang dipakai robot dan menganalisa hasil F1 *score* dan PR (*Precission-Recall*) *curve*. Webcam yang dipakai merupakan Logitech C920, *dataset* yang ditraining menggunakan *hyperparameters* 5 epochs dan batch size 8 dengan keadaan pengambilam data mengambil 1 data setiap 3 detik. Pengujian bola akan ada tiga variasi yaitu pada bola ditengah dan di kanan kiri dengan sudut 30 derajat sejauh 3 meter, sedangkan gawang akan ada dua variasi yaitu ditengah kamera dengan jarak berbeda 3 meter dan 6 meter.

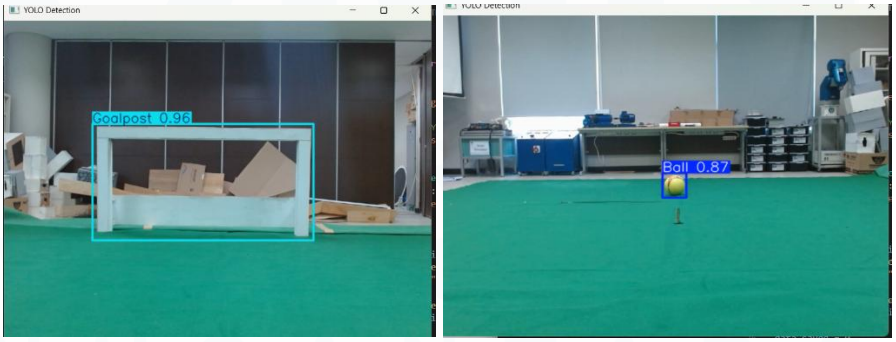


Gambar 3.7. Kurva *F1-score* dan PR YOLOv8l



Gambar 3.8. Kurva *F1-score* dan PR RT-DETR-L

Dari hasil kurva *F1-score* YOLO mendapatkan nilai *F1-score* maksimum sebesar 0.98 pada *confidence* 0.443, sedangkan RT-DETR mendapatkan nilai *F1-score* maksimum 0.97 pada *confidence* 0.6. Menunjukkan bahwa YOLO dapat performa optimal pada *threshold* yang lebih rendah, sedangkan RT-DETR sebaliknya optimal saat *threshold* lebih tinggi. Pada grafik PR, YOLO mendapatkan nilai *mean Average Precision* (mAP) sebesar 0.99, sedangkan RT-DETR mendapatkan mAP 0.985. Menunjukkan bahwa YOLO lebih unggul dan kurva YOLO menunjukkan lebih stabil dari RT-DETR.



Gambar 3.9. Dokumentasi Pengujian

Tabel 3.1. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Tengah

Data Percobaan ke-	YOLO		RT-DETR	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.842	0.0396	0.899	0.0514
2	0.842	0.0329	0.898	0.0516
3	0.842	0.0306	0.897	0.0466
4	0.842	0.0309	0.896	0.0475
5	0.843	0.0302	0.902	0.0526
6	0.842	0.0323	0.900	0.0501
7	0.842	0.0305	0.899	0.0513
8	0.842	0.0303	0.898	0.0502
9	0.842	0.0298	0.899	0.049
10	0.842	0.0301	0.898	0.0575
11	0.842	0.0298	0.898	0.0491
12	0.842	0.0309	0.900	0.0498
13	0.843	0.0329	0.899	0.0505
14	0.842	0.0306	0.898	0.0509
15	0.843	0.0313	0.901	0.0494
16	0.842	0.0308	0.899	0.0514
17	0.843	0.0299	0.900	0.0485
18	0.842	0.0319	0.900	0.0488
19	0.842	0.0301	0.898	0.052
20	0.843	0.0299	0.899	0.0506
21	0.842	0.0308	0.899	0.0524
22	0.843	0.0303	0.898	0.0494
23	0.842	0.0298	0.900	0.0527
24	0.842	0.0300	0.899	0.0554
25	0.842	0.0355	0.898	0.0517
26	0.842	0.0315	0.899	0.0528
27	0.843	0.0313	0.898	0.0502
28	0.842	0.0318	0.898	0.0521
29	0.842	0.0310	0.898	0.0513
30	0.842	0.0306	0.898	0.0534

Rata-Rata	0.842	0.0313	0.899	0.0510

Tabel 3.2. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Kanan

Data Percobaan ke-	YOLO		RT-DETR	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.859	0.0188	0.898	0.0325
2	0.858	0.0188	0.898	0.0324
3	0.859	0.018	0.898	0.0326
4	0.859	0.018	0.898	0.0336
5	0.859	0.0176	0.897	0.033
6	0.859	0.0176	0.898	0.0322
7	0.859	0.0181	0.898	0.0336
8	0.858	0.0181	0.897	0.0335
9	0.859	0.0174	0.898	0.0327
10	0.859	0.0174	0.898	0.0319
11	0.859	0.0177	0.897	0.033
12	0.860	0.0177	0.898	0.033
13	0.875	0.0258	0.898	0.0345
14	0.857	0.0175	0.898	0.0328
15	0.873	0.0178	0.898	0.0324
16	0.866	0.0178	0.898	0.0554
17	0.876	0.0177	0.898	0.0336
18	0.874	0.0174	0.898	0.0401
19	0.872	0.0176	0.899	0.0328
20	0.871	0.0174	0.899	0.0339
21	0.871	0.0181	0.898	0.0347
22	0.870	0.0178	0.897	0.0327
23	0.869	0.0178	0.898	0.0334
24	0.870	0.0176	0.895	0.0334
25	0.870	0.0176	0.890	0.0335
26	0.869	0.0176	0.902	0.0334

27	0.869	0.018	0.902	0.0328
28	0.868	0.018	0.904	0.0326
29	0.867	0.0177	0.904	0.0557
30	0.867	0.0177	0.906	0.0326
Rata-Rata	0.865	0.01807	0.898	0.0348

Tabel 3.3. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Kiri

Data Percobaan ke-	YOLO		RT-DETR	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.890	0.0174	0.903	0.0538
2	0.896	0.0174	0.903	0.0493
3	0.890	0.0180	0.902	0.0335
4	0.892	0.0180	0.903	0.0366
5	0.890	0.0205	0.902	0.0397
6	0.890	0.0205	0.904	0.0382
7	0.890	0.0177	0.903	0.0329
8	0.890	0.0177	0.902	0.0335
9	0.890	0.0175	0.903	0.0329
10	0.889	0.0175	0.902	0.0335
11	0.889	0.0191	0.902	0.0332
12	0.889	0.0191	0.904	0.0334
13	0.890	0.0175	0.902	0.0325
14	0.890	0.0175	0.903	0.0330
15	0.889	0.0177	0.903	0.0365
16	0.889	0.0177	0.902	0.0328
17	0.890	0.0176	0.904	0.0331
18	0.890	0.0176	0.903	0.0333
19	0.890	0.0176	0.903	0.0422
20	0.890	0.0176	0.903	0.0328
21	0.890	0.0269	0.903	0.0328
22	0.890	0.0269	0.902	0.0395
23	0.890	0.0176	0.903	0.0337

24	0.890	0.0176	0.903	0.0328
25	0.890	0.0178	0.902	0.0336
26	0.890	0.0178	0.902	0.0425
27	0.890	0.0175	0.902	0.0335
28	0.890	0.0175	0.903	0.0337
29	0.889	0.0176	0.903	0.0328
30	0.889	0.0176	0.902	0.0328
Rata-Rata	0.890	0.0185	0.903	0.0358

Tabel 3.4. Data *Confidence Level* dan *Inference Time* untuk Gawang

Data Percobaan ke-	YOLO				RT-DETR			
	3 meter		6 meter		3 meter		6 meter	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.887	0.0176	0.930	0.0175	0.973	0.0326	0.958	0.0338
2	0.887	0.0175	0.928	0.0182	0.974	0.0387	0.957	0.0331
3	0.883	0.0201	0.929	0.0177	0.973	0.0325	0.958	0.0329
4	0.885	0.0184	0.929	0.0175	0.973	0.0326	0.957	0.0372
5	0.885	0.0179	0.930	0.0258	0.974	0.0324	0.958	0.0494
6	0.883	0.0176	0.930	0.0176	0.973	0.0443	0.957	0.0330
7	0.885	0.0176	0.929	0.0177	0.974	0.0325	0.957	0.0343
8	0.884	0.0175	0.929	0.0176	0.974	0.0324	0.957	0.0334
9	0.883	0.0176	0.928	0.0175	0.973	0.0325	0.958	0.0330
10	0.883	0.0176	0.929	0.0176	0.973	0.0328	0.957	0.0337
11	0.886	0.0176	0.929	0.0177	0.973	0.0325	0.957	0.0393
12	0.885	0.0176	0.928	0.0177	0.973	0.0324	0.958	0.0351
13	0.883	0.0184	0.928	0.0177	0.974	0.0328	0.958	0.0417
14	0.881	0.0176	0.929	0.0182	0.974	0.0342	0.958	0.0329
15	0.883	0.0177	0.930	0.0177	0.974	0.0366	0.958	0.0332
16	0.885	0.0176	0.929	0.0180	0.973	0.0322	0.958	0.0329
17	0.886	0.0199	0.929	0.0176	0.973	0.0330	0.958	0.0328
18	0.884	0.0177	0.929	0.0178	0.974	0.0323	0.958	0.0322
19	0.884	0.0175	0.928	0.0179	0.973	0.0335	0.957	0.0330
20	0.884	0.0175	0.929	0.0176	0.973	0.0340	0.957	0.0354
21	0.884	0.0177	0.928	0.0177	0.974	0.0329	0.958	0.0326
22	0.882	0.0176	0.929	0.0176	0.973	0.0331	0.958	0.0328
23	0.884	0.0175	0.928	0.0178	0.973	0.0324	0.958	0.0328

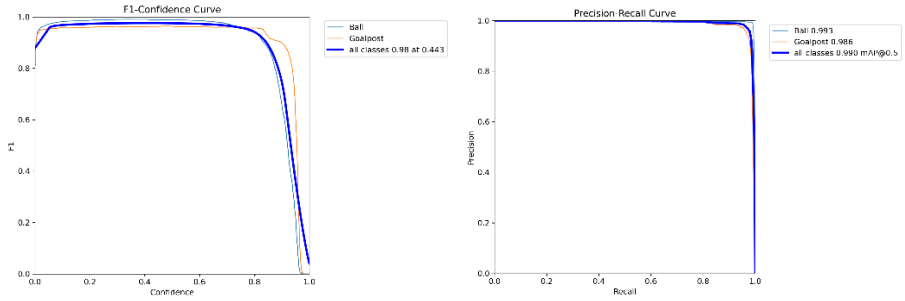
24	0.885	0.0180	0.928	0.0180	0.974	0.0327	0.957	0.0438
25	0.884	0.0189	0.929	0.0177	0.974	0.0325	0.957	0.0328
26	0.885	0.0177	0.928	0.0178	0.974	0.0338	0.957	0.0339
27	0.887	0.0227	0.928	0.0177	0.974	0.0326	0.957	0.0333
28	0.886	0.0203	0.928	0.0178	0.974	0.0321	0.957	0.0324
29	0.884	0.0184	0.928	0.0227	0.974	0.0327	0.958	0.0322
30	0.883	0.0178	0.928	0.0176	0.973	0.0329	0.957	0.0328
Rata-rata	0.884	0.0182	0.929	0.0182	0.974	0.0335	0.958	0.0347

Pada Berdasarkan hasil pengujian terhadap algoritma YOLO dan RT-DETR dalam mendeteksi objek bola dan gawang pada berbagai kondisi posisi dan jarak, dapat disimpulkan bahwa RT-DETR memiliki tingkat kepercayaan (*confidence level*) yang lebih tinggi dan stabil dibandingkan YOLO di seluruh skenario. Rata-rata *confidence level* RT-DETR untuk deteksi bola berada pada posisi tengah, kanan, dan kiri masing-masing sebesar 0.899, 0.898, dan 0.903, sedangkan YOLO menghasilkan nilai rata-rata 0.842, 0.865, dan 0.890 untuk posisi yang sama. Untuk deteksi gawang pada jarak 3 meter dan 6 meter, RT-DETR mencatat *confidence level* rata-rata sebesar 0.974 dan 0.958, lebih tinggi dibandingkan YOLO yang hanya mencapai 0.884 dan 0.929. Meskipun demikian, dari segi kecepatan proses deteksi (*inference time*), YOLO secara konsisten lebih cepat, dengan waktu rata-rata untuk deteksi bola di posisi tengah, kanan, dan kiri masing-masing sebesar 0.0313 detik, 0.0181 detik, dan 0.0185 detik. Sebaliknya, RT-DETR mencatat waktu lebih lama yaitu 0.0510 detik, 0.0348 detik, dan 0.0358 detik untuk kondisi yang sama. Pada deteksi gawang, *inference time* YOLO rata-rata adalah 0.0182 detik untuk kedua jarak, sedangkan RT-DETR membutuhkan waktu rata-rata 0.0335 detik (3 meter) dan 0.0347 detik (6 meter). Dengan demikian, RT-DETR unggul dalam akurasi deteksi, namun YOLO lebih unggul dalam kecepatan. Sehingga YOLO sesuai dengan kebutuhan robot sepak bola beroda karena *inference time* dan hasil metriks lebih unggul untuk diaplikasikan secara real-time meskipun *confidence level* dibawah RT-DETR.

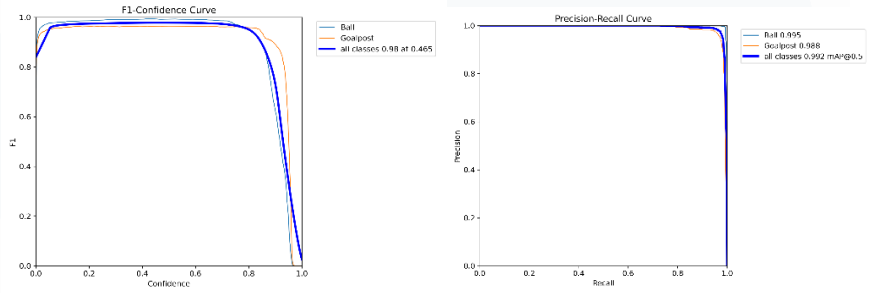
3.3.4 Perbandingan YOLOv8n dengan YOLO11n

You Look Only Once (YOLO) menggunakan konsep proses neuron-neuron yang saling terhubung seperti saraf manusia dan juga mirip dengan konsep *Neural Networks* (NNs). Dengan mengambil dasar CNN, YOLO juga bisa mengidentifikasi fitur-fitur yang ada pada gambar. YOLO mempunyai proses dengan 3 struktur yaitu *backbone*, *neck*, *head* dari konsep *Feature Pyramid Network* (FPN) [13]. Bagian *backbone* mempunyai fungsi untuk mengekstrak *data input* menggunakan CNN dengan mempelajari fitur-fitur pada gambar. Bagian neck mempunyai fungsi untuk mengumpulkan data fitur menggunakan *Path Aggretation Network* (PAN). Bagian *head* mempunyai fungsi untuk menghasilkan hasil atau *output*. YOLOv8 dan YOLO11 memakai dasar prinsip tersebut tetapi mempunyai pembeda, YOLOv8 mempunyai pendekatan *anchor-free detection* pada bagian *head* untuk menyederhankan model dan meningkatkan performa terutama pada objek kecil [14]. Sedangkan YOLO11 memiliki komponen C3k2 block, *Spatial Pyramid Pooling -Fast* (SPPF), dan C2PSA (*Convolution Block with Parallel Spatial Attention*) yang mendukung untuk *feature extraction*, *pose estimation*, *tracking*, dan *classification* meningkat [15].

Untuk membandingkan algoritma yang dipilih sesuai dengan kebutuhan robot sepakbola beroda, akan dilakukan percobaan antara algortima YOLOv8n dan YOLO11n dengan mengumpulkan 30 data confidence level dan *inference time* dari menguji bola dan gawang menggunakan *webcam* yang dipakai robot dan menganalisa hasil F1 *score* dan PR (*Precission-Recall*) *curve*. Webcam yang dipakai merupakan Logitech C920, *dataset* yang ditraining menggunakan *hyperparameters* 100 epochs dan batch size 16 dengan keadaan pengambilam data dengan mengambil 1 data setiap 3 detik. Pengujian bola akan ada tiga variasi yaitu pada bola ditengah dan di kanan kiri dengan sudut 30 derajat sejauh 3 meter, sedangkan gawang akan ada dua variasi yaitu ditengah kamera dengan jarak berbeda 3 meter dan 6 meter.

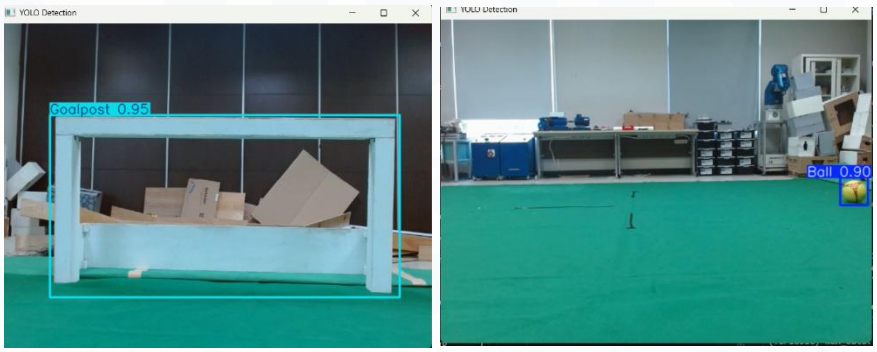


Gambar 3.10. Kurva *F1-score* dan PR YOLOv8n



Gambar 3.11. Kurva *F1-score* dan PR YOLOv11n

Dari hasil kurva *F1-score* YOLOv8n mendapatkan nilai *F1-score* maksimum sebesar 0.98 pada *confidence* 0.443, sedangkan YOLO11n mendapatkan nilai *F1-score* maksimum 0.98 pada *confidence* 0.465. Menunjukkan bahwa YOLOv8n dapat performa optimal pada *threshold* yang lebih rendah tetapi tidak signifikan perbedaannya, sedangkan YOLO11n sebaliknya optimal saat *threshold* lebih tinggi. Pada grafik PR, YOLOv8n mendapatkan nilai *mean Average Precision* (mAP) sebesar 0.99, sedangkan YOLO11n mendapatkan mAP 0.992.



Gambar 3.12. Dokumentasi Pengujian

Tabel 3.5. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Tengah

Data Percobaan ke-	YOLOv8n		YOLO11n	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.879	0.0131	0.871	0.0145
2	0.878	0.0120	0.871	0.0145
3	0.878	0.0129	0.868	0.0172
4	0.880	0.0129	0.872	0.0162
5	0.877	0.0162	0.870	0.0146
6	0.879	0.0118	0.870	0.0148
7	0.879	0.0125	0.871	0.0167
8	0.879	0.0128	0.870	0.0176
9	0.877	0.0125	0.870	0.0144
10	0.877	0.0123	0.870	0.0142
11	0.879	0.0124	0.869	0.0168
12	0.878	0.0175	0.870	0.0140
13	0.878	0.0123	0.868	0.0189
14	0.879	0.0135	0.870	0.0145
15	0.877	0.0128	0.872	0.0145
16	0.880	0.0132	0.868	0.0145
17	0.880	0.0206	0.871	0.0166
18	0.877	0.0128	0.868	0.0142
19	0.878	0.0133	0.869	0.0149
20	0.878	0.0147	0.872	0.0166
21	0.878	0.0127	0.870	0.0147
22	0.876	0.0124	0.869	0.0173
23	0.880	0.0135	0.871	0.0146
24	0.877	0.0134	0.870	0.0159
25	0.882	0.0144	0.883	0.0141
26	0.879	0.0125	0.883	0.0169
27	0.878	0.0134	0.881	0.0148
28	0.876	0.0130	0.885	0.0147
29	0.878	0.0201	0.879	0.0141

30	0.879	0.0150	0.885	0.0146
Rata-Rata	0.878	0.0138	0.873	0.0154

Tabel 3.6. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Kanan

Data Percobaan ke-	YOLOv8n		YOLO11n	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.879	0.0230	0.869	0.0128
2	0.886	0.0112	0.865	0.0135
3	0.885	0.0118	0.869	0.0133
4	0.887	0.0209	0.868	0.0134
5	0.888	0.0121	0.867	0.0134
6	0.887	0.0120	0.870	0.0133
7	0.887	0.0129	0.867	0.0151
8	0.882	0.0122	0.867	0.0134
9	0.884	0.0120	0.870	0.0144
10	0.883	0.0130	0.869	0.0139
11	0.880	0.0131	0.867	0.0129
12	0.885	0.0127	0.867	0.0135
13	0.889	0.0151	0.868	0.0130
14	0.887	0.0130	0.867	0.0130
15	0.892	0.0125	0.865	0.0138
16	0.886	0.0124	0.872	0.0131
17	0.889	0.0126	0.864	0.0260
18	0.888	0.0208	0.870	0.0154
19	0.888	0.0148	0.867	0.0178
20	0.886	0.0126	0.866	0.0134
21	0.886	0.0122	0.868	0.0134
22	0.883	0.0140	0.864	0.0134
23	0.883	0.0116	0.866	0.0129
24	0.888	0.0189	0.866	0.0164
25	0.889	0.0126	0.869	0.0136

26	0.884	0.0122	0.869	0.0166
27	0.885	0.0209	0.863	0.0134
28	0.889	0.0124	0.867	0.0135
29	0.889	0.0127	0.868	0.0132
30	0.892	0.0151	0.865	0.0131
Rata-Rata	0.886	0.0141	0.867	0.0143

Tabel 3.7. Data *Confidence Level* dan *Inference Time* saat Kondisi Bola di Kiri

Data Percobaan ke-	YOLOv8n		YOLO11n	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.891	0.0105	0.905	0.0135
2	0.892	0.0111	0.905	0.0141
3	0.887	0.0114	0.905	0.0135
4	0.889	0.0125	0.904	0.0133
5	0.890	0.0149	0.907	0.0128
6	0.890	0.0127	0.905	0.0141
7	0.891	0.0122	0.904	0.0128
8	0.889	0.0133	0.904	0.0132
9	0.889	0.0205	0.903	0.0130
10	0.887	0.0123	0.906	0.0131
11	0.892	0.0129	0.904	0.0135
12	0.888	0.0149	0.906	0.0172
13	0.893	0.0128	0.904	0.0130
14	0.892	0.0128	0.904	0.0248
15	0.891	0.0120	0.903	0.0137
16	0.888	0.0123	0.904	0.0128
17	0.887	0.0144	0.905	0.0134
18	0.890	0.0143	0.904	0.0130
19	0.889	0.0219	0.902	0.0221
20	0.891	0.0212	0.905	0.0142
21	0.889	0.0224	0.906	0.0160
22	0.887	0.0121	0.906	0.0132

23	0.890	0.0126	0.904	0.0131
24	0.892	0.0203	0.906	0.0135
25	0.893	0.0135	0.902	0.0165
26	0.891	0.0124	0.904	0.0139
27	0.890	0.0122	0.904	0.0134
28	0.887	0.0136	0.906	0.0137
29	0.890	0.0130	0.905	0.0258
30	0.891	0.0127	0.905	0.0138
Rata-Rata	0.890	0.0142	0.905	0.0148

Tabel 3.8. Data *Confidence Level* dan *Inference Time* untuk Gawang

Data Percobaan ke-	YOLOv8n				YOLO11n			
	3 meter		6 meter		3 meter		6 meter	
	Confidence Level	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)	*Confidence Level*	*Inference Time* (detik)
1	0.970	0.0103	0.956	0.0114	0.939	0.0144	0.935	0.0165
2	0.967	0.0106	0.959	0.0108	0.939	0.0150	0.930	0.0133
3	0.966	0.0115	0.956	0.0120	0.940	0.0158	0.944	0.0133
4	0.970	0.0127	0.959	0.0135	0.940	0.0144	0.923	0.0129
5	0.969	0.0119	0.957	0.0127	0.940	0.0163	0.937	0.0129
6	0.972	0.0146	0.956	0.0150	0.942	0.0168	0.945	0.0132
7	0.970	0.0122	0.956	0.0132	0.945	0.0146	0.936	0.0130
8	0.972	0.0134	0.958	0.0152	0.946	0.0144	0.938	0.0129
9	0.970	0.0140	0.958	0.0120	0.946	0.0152	0.941	0.0136
10	0.971	0.0121	0.958	0.0125	0.945	0.0136	0.944	0.0127
11	0.971	0.0140	0.957	0.0218	0.946	0.0162	0.935	0.0250
12	0.971	0.0123	0.958	0.0211	0.943	0.0149	0.936	0.0126
13	0.968	0.0129	0.957	0.0148	0.944	0.0140	0.943	0.0127
14	0.968	0.0121	0.958	0.0219	0.942	0.0144	0.946	0.0143
15	0.968	0.0131	0.958	0.0131	0.943	0.0145	0.926	0.0242
16	0.971	0.0123	0.958	0.0125	0.944	0.0144	0.933	0.0129
17	0.973	0.0122	0.957	0.0126	0.944	0.0160	0.944	0.0185
18	0.970	0.0206	0.957	0.0158	0.941	0.0147	0.945	0.0127
19	0.970	0.0129	0.958	0.0128	0.945	0.0143	0.941	0.0131
20	0.969	0.0122	0.959	0.0128	0.939	0.0137	0.946	0.0170
21	0.971	0.0120	0.958	0.0128	0.945	0.0405	0.942	0.0131
22	0.972	0.0124	0.957	0.0143	0.946	0.0136	0.940	0.0145

23	0.972	0.0138	0.959	0.0207	0.942	0.0144	0.931	0.0126
24	0.971	0.0130	0.958	0.0160	0.944	0.0137	0.943	0.0132
25	0.970	0.0128	0.959	0.0127	0.943	0.0134	0.945	0.0131
26	0.967	0.0120	0.957	0.0127	0.946	0.0138	0.938	0.0128
27	0.969	0.0189	0.957	0.0127	0.945	0.0134	0.944	0.0158
28	0.966	0.0131	0.957	0.0136	0.942	0.0140	0.947	0.0130
29	0.971	0.0129	0.956	0.0129	0.945	0.0135	0.942	0.0129
30	0.971	0.0289	0.959	0.0116	0.941	0.0134	0.945	0.0131
Rata-rata	0.970	0.0136	0.958	0.0143	0.943	0.0154	0.939	0.0144

Berdasarkan hasil pengujian dan analisis terhadap performa algoritma YOLOv8n dan YOLO11n, dapat disimpulkan bahwa YOLO11n unggul dalam hal tingkat kepercayaan (*confidence level*) pada sebagian besar kondisi, sedangkan YOLOv8n lebih unggul dalam kecepatan waktu inferensi (*inference time*). Pada pengujian deteksi bola, YOLO11n mencatat rata-rata *confidence level* tertinggi pada kondisi bola di kiri sebesar 0.905, dibandingkan YOLOv8n yang mencatat 0.890. Hal serupa juga terjadi pada kondisi bola di tengah (YOLO11n: 0.873, YOLOv8n: 0.878) dan di kanan (YOLO11n: 0.867, YOLOv8n: 0.886), meskipun pada kondisi bola di tengah dan kanan, YOLOv8n sedikit lebih unggul dalam *confidence level*. Namun demikian, dari sisi kecepatan, YOLOv8n mencatat waktu inferensi rata-rata lebih rendah di hampir seluruh skenario: 0.0138 detik (tengah), 0.0141 detik (kanan), dan 0.0142 detik (kiri), dibandingkan dengan YOLO11n yang memerlukan 0.0154 detik, 0.0143 detik, dan 0.0148 detik masing-masing. Pada pengujian deteksi gawang dengan jarak 3 meter dan 6 meter, YOLO11n tetap menunjukkan *confidence level* yang lebih tinggi secara konsisten: 0.943 (3 m) dan 0.939 (6 m), dibandingkan YOLOv8n yang mencatat 0.970 (3 m) dan 0.958 (6 m). Namun, waktu inferensi YOLOv8n tetap lebih cepat dengan rata-rata 0.0136 detik (3 m) dan 0.0143 detik (6 m), sementara YOLO11n mencatat 0.0154 detik (3 m) dan 0.0144 detik (6 m). Sehingga dalam pengujian YOLO11n seddikit lebih unggul dalam *confidence level* dan sedikit lebih rendah dalam *inference time* dibandingkan YOLOv8n. Sehingga dipilih YOLOv11n mempertimbangkan hasil *metrics*, *confidence level*, *inference time* yang tidak jauh

berbeda, dan fitur-fitur seperti *tracking*, *segmentation*, *classification* yang lebih baik dari versi sebelumnya.