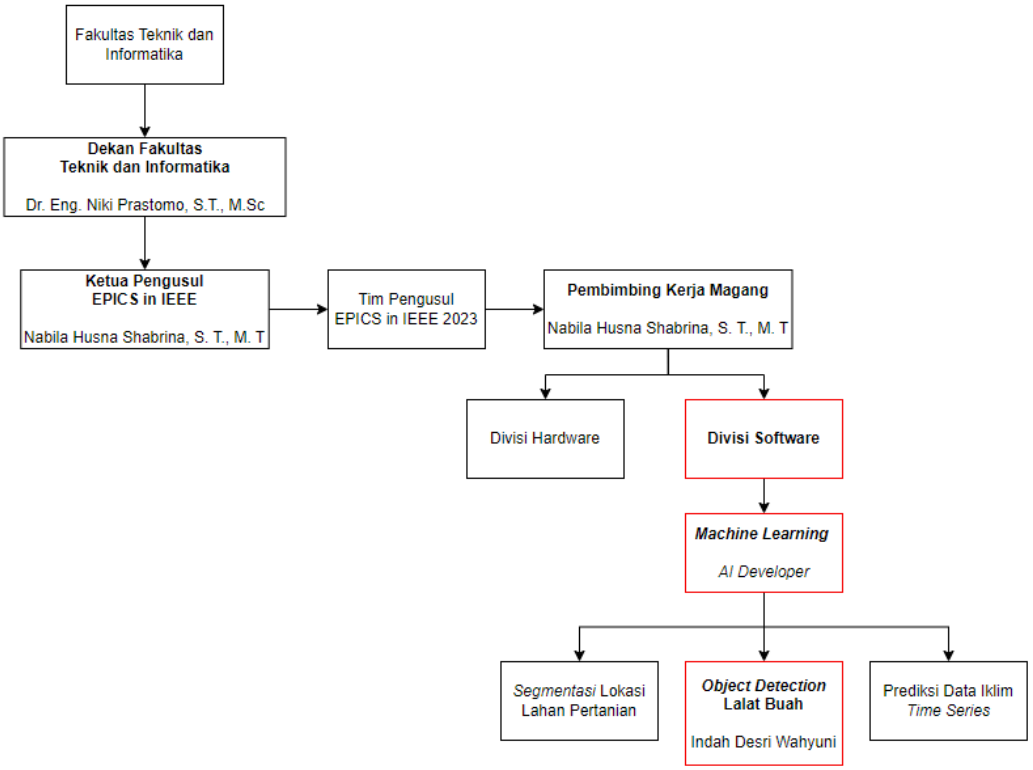
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada pelaksanaan kerja magang ini penulis tergabung dalam *project EPICS in IEEE* yaitu Sistem Pertanian Pintar dengan Integrasi *Artificial Intelligence* dan *Internet of Things* untuk Peramalan dan Pengelolaan Hama. Pada proyek ini Universitas Multimedia Nusantara (UMN) berkolaborasi dengan Universitas Gadjah Mada (UGM) mendapatkan dana hibah EPICS in IEEE mengenai Sistem Pertanian Pintar dengan Integrasi *Artificial Intelligence* dan *Internet of Things* untuk Peramalan dan Pengelolaan Hama. Pada project ini terdapat 2 divisi yaitu *Hardware* dan *Software*. Penulis tergabung dalam divisi *Software* bagian *Machine Learning* yang berperan sebagai mengembangkan sistem *Artificial Intelligence* (AI). Dalam divisi ini terdapat 3 bagian peran kerja pengembangan *Artificial Intelligence* (AI) untuk deteksi dan *counting* object lalat buah, segmentasi lokasi lahan pertanian, dan peramalan data sensor. Penulis bekerja sebagai AI *developer* bagian deteksi objek lalat buah di bawah bimbingan Ibu Nabila Husna Shabrina. Dalam pelaksanaan *project* ini divisi *Software* juga berkolaborasi dengan divisi *hardware* dalam pengambilan data primer pada sensor untuk peramalan secara *real-time*. Secara spesifik, alur kerja tanggung jawab, dan bagan kedudukan penulis sebagai *Machine Learning Developer* ditunjukkan pada Gambar 3.1.



Gambar 3.1 Struktur Spesifik divisi *Machine Learning Developer*

3.2 Tugas dan Uraian Kerja Magang

3.2.1 Tugas yang Dilakukan

Pada pelaksanaan kerja magang pada project EPICS in IEEE Sistem Pertanian Pintar dengan Integrasi *Artificial Intelligence* dan *Internet of Things* untuk Peramalan dan Pengelolaan Hama, penulis ditugaskan untuk dapat merancang sistem *Machine Learning* dengan memanfaatkan *artificial intelligence* (AI) untuk membuat deteksi otomatis pada *object* lalat yang masuk dalam perangkap sehingga dapat membantu petani dalam mengelola lahan pertanian agar memberikan tindakan yang tepat untuk menanggulangi hama.

Dalam melakukan perancangan dan pengembangan sistem AI untuk *object detection* ini, pada minggu pertama dilakukan studi literatur mengenai deteksi objek, model yang sering digunakan, parameter, metric

evaluasi, dan mengumpulkan dataset sekunder yang akan digunakan. Minggu kedua, penulis mencoba melakukan labelling dataset sekunder yang telah terkumpul dan melakukan perancangan model yang sudah dipilih untuk deteksi objek yaitu Yolo V3, Yolo V5, Yolo V7, Yolo V8, dan Fast RCNN dengan menggunakan flowchart untuk memudahkan dalam dokumentasi koding.

Minggu ketiga sampai ketujuh, penulis melakukan pelatihan model Yolo V3, Yolo V5, dan Yolo V7 dengan menggunakan google collab. Pelatihan dilakukan dengan mengganti parameter seperti img size, batch, dan epoch. Minggu kedelapan, penulis melakukan kunjungan lapangan ke Sleman, Yogyakarta untuk kontrak kerjasama dengan pihak Universitas Gadjah Mada (UGM), melakukan diskusi dengan CV. Mitra Turindo dan pengambilan dataset lahan pertanian. Dari minggu kesembilan hingga akhir masa magang penulis melakukan training model Yolo V8 dan Fast RCNN, dan menentukan model yang optimal.

3.2.2 Uraian Kerja Magang

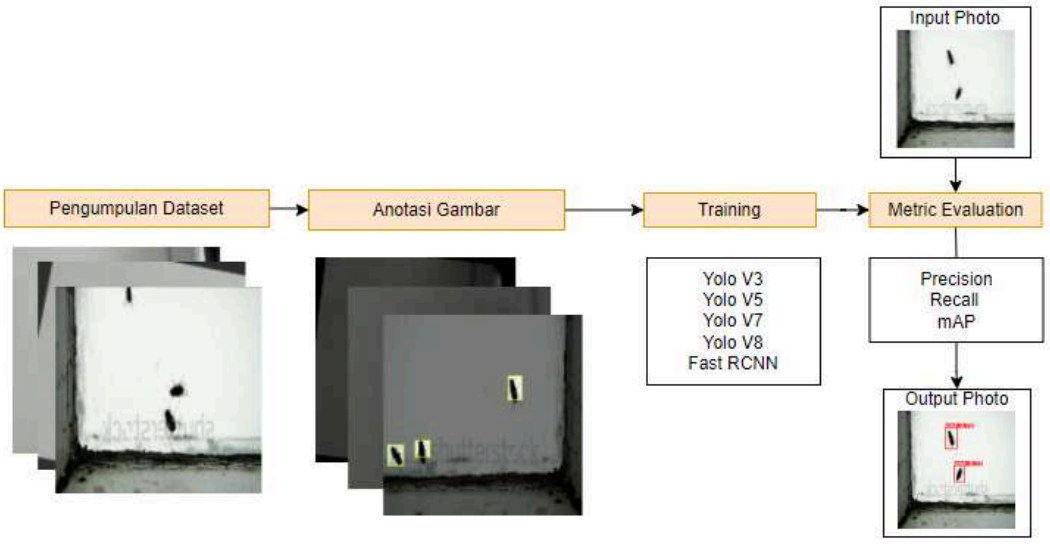
Tabel 3.1 Linimasa Kerja Magang

Minggu	Rincian Tugas yang Dilakukan
1	1. Briefing bersama pembimbing lapangan terkait project EPICS in IEEE Sistem Pertanian Pintar dengan Integrasi *Artificial Intelligence* dan *Internet of Things* untuk Peramalan dan Pengelolaan Hama dan pembagian tugas antar divisi. 2. Studi literatur mengenai perancangan AI untuk deteksi object berupa konsep, preprocessing, training, deteksi, matrik evaluasi.
2	1. Menentukan model yang akan dipakai dalam deteksi object yaitu Yolo V3, Yolo V5, Yolo V7, Yolo V8,

	dan Fast RCNN. 2. Pembuatan Flowchart dari model Yolo V3, Yolo V5, Yolo V7, Yolo V8, dan Fast RCNN secara detail untuk memudahkan dokumentasi algoritma.
3	Mengumpulkan dan melakukan labelling pada dataset yang terkumpul sebanyak 2789 foto. Hal ini dilakukan di framework roboflow.
4 - 7	Melakukan *training* dari model Yolo V3, Yolo V5, dan Yolo V7. *Training* dilakukan berulang kali dengan mengubah parameter model untuk mendapatkan hasil yang optimal.
8	Melakukan kunjungan lapangan dan pengumpulan dataset di Sleman, Yogyakarta. Kunjungan lapangan dilakukan untuk briefing terkait project yang akan dilaksanakan. Dataset yang dikumpulkan berupa foto lalat buah.
9 - 12	Melakukan *training* dari model Yolo V8 dan Fast RCNN. *Training* dilakukan berulang kali dengan mengubah parameter model untuk mendapatkan hasil yang optimal.
13 - 14	1. Melakukan perbandingan dari hasil *training* dan deteksi dari model Yolo V3, Yolo V5, Yolo V7, Yolo V8, dan Fast RCNN dengan membandingkan hasil nilai *precision*, *recall*, dan mAP. 2. Melakukan analisis dari setiap hasil *training* yang ditampilkan pada setiap model Yolo V3, Yolo V5, Yolo V7, Yolo V8, dan Fast RCNN untuk mengetahui kinerja dari masing-masing model
15 - 16	1. Menentukan model yang terbaik yaitu Yolo V5 untuk deteksi objek

	2. Melakukan *training* ulang pada parameter yang terbaik 3. Melakukan *testing* berulang kali dengan dataset yang beragam untuk menentukan rata-rata kecepatan deteksi yaitu sebesar 13 ms 4. Pembuatan laporan magang

3.2.3 Gambaran Besar Sistem

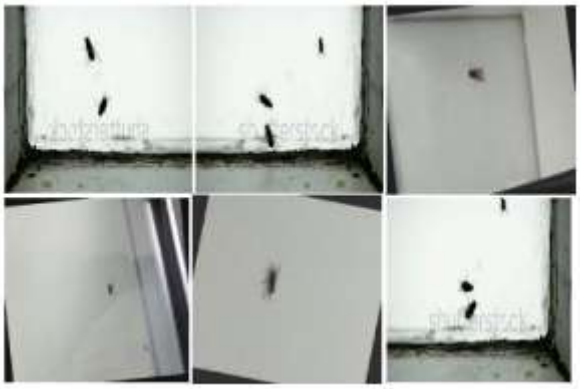


Gambar 3.2 Diagram Blok Sistem

Pengembangan metode objek deteksi dimulai dengan mengakuisisi foto. Proses foto diambil dari kamera, namun penulis melakukan pencarian foto tersebut melalui riset. Foto yang tersimpan dengan format JPG. Selanjutnya melakukan labelling dan disimpan dalam dataset. Model akan dilatih dengan menggunakan dataset yang telah dibuat agar dapat mengidentifikasi objek atau kelas dalam foto. Prediksi model digunakan untuk memprediksi objek atau kelas dalam foto kemudian ditampilkan sebagai output.

3.2.4 Pengumpulan Dataset Sekunder

Dataset yang digunakan dalam pengerjaan magang berasal dari *framework* roboflow yang berjumlah 2789 foto [2]. Pada Gambar 3.3 dapat dilihat contoh dataset yang dikumpulkan.

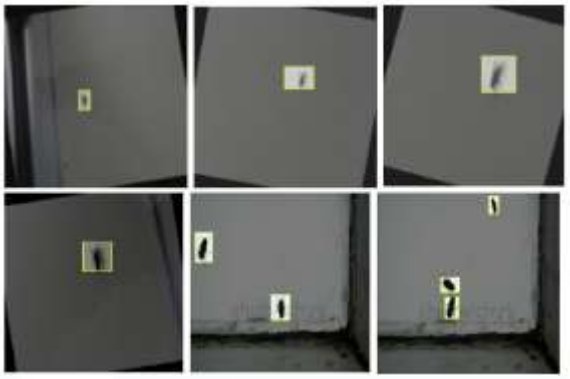


Gambar 3.3 Contoh Gambar Dataset

Foto tersebut dipilih menjadi dataset sekunder sebab merepresentasikan dataset primer yang mana objek kecil, objek lebih dari 1, dan latar belakang *clear*. Sehingga, harapannya model yang sudah dilatih dengan menggunakan dataset sekunder ini dapat bekerja secara optimal jika diimplementasikan pada dataset primer.

3.2.5 Anotasi Gambar

Anotasi gambar atau *image labelling* merupakan proses mengidentifikasi atau mengenai unit yang berbeda dalam suatu gambar. Jenis anotasi pelabelan data yang sering digunakan dalam *computer vision* adalah *bounding box* seperti yang ditunjukkan dalam Gambar 3.4. *Bounding box* berupa kotak persegi panjang yang digunakan untuk menentukan lokasi objek target. *Bounding box* biasanya direpresentasikan oleh dua koordinat (x1, y1) dan (x2, y2) atau dengan satu koordinat (x1, y1) dan lebar (w) dan tinggi (h) dari kotak pembatas [3].



Gambar 3.4 Anotasi Gambar

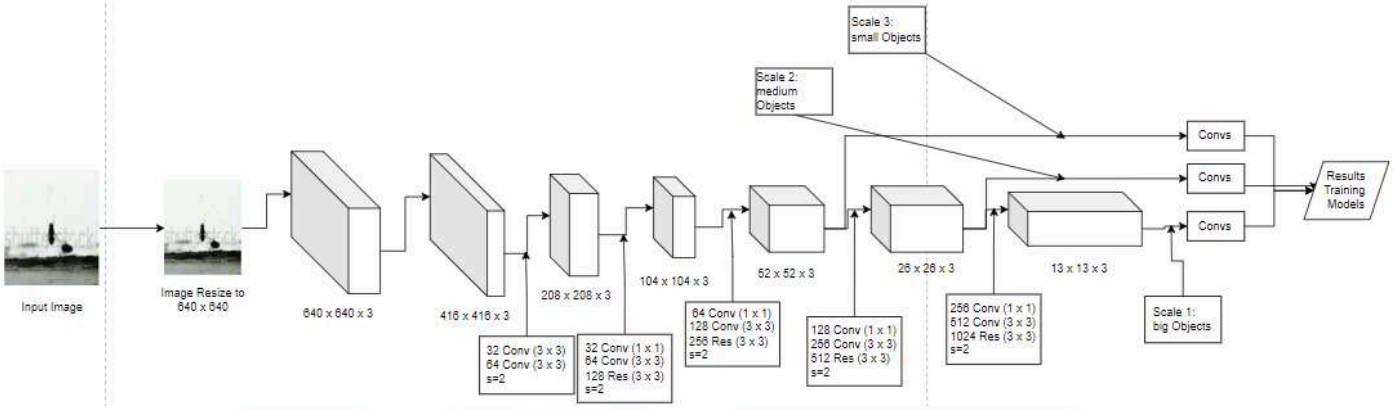
Sebelum melakukan training atau pelatihan pada model, maka dilakukan *preprocessing* setelah *labelling* pada dataset. Dataset dibagi menjadi data training, validation, dan testing dengan persentase 80:10:10. Selanjutnya, dilakukan scaling gambar sehingga ukuran gambar menjadi 640x640. Seluruh proses dari pembagian dataset dan scaling dilakukan menggunakan *framework* Roboflow.

3.2.6 Architecture Models

Pada pengerjaan magang ini akan diimplementasikan metode Yolo V3, Yolo V5, Yolo V7, Yolo V8, dan Fast RCNN dan dibandingkan performanya untuk mendapatkan model terbaik.

- Yolo V3

  Yolo V3 menggunakan arsitektur dari Darknet 53 yang artinya memiliki convolutional layers sebanyak 53. Arsitektur Yolo V3 yang terdiri dari struktur konvolusional dan residual [3]. Dapat dilihat Gambar 3.5 Architecture Yolo V3.



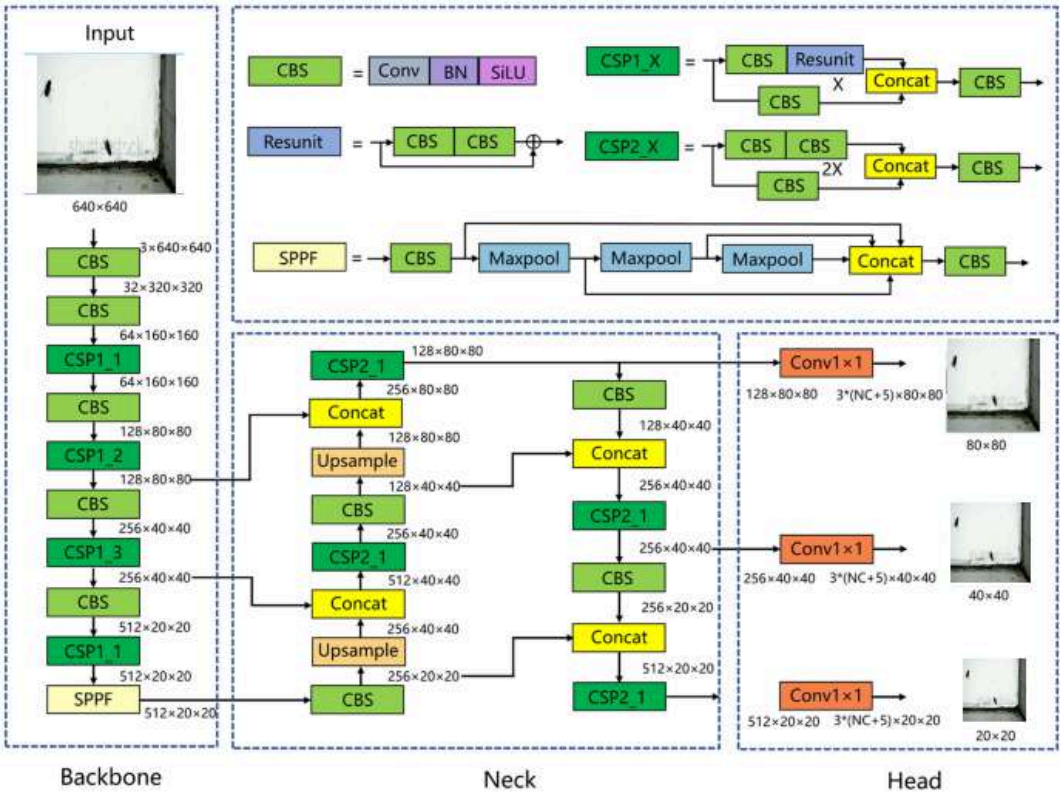
Gambar 3.5 *Architecture* Yolo V3

Yolo V3 tidak hanya mengekstrak proposal wilayah, tetapi hanya memproses gambar masukan lengkap dengan menggunakan *Fully Convolutional Neural Network* yang memprediksikan kotak pembatas dan probabilitas kelasnya yang sesuai. Di dalam Yolo V3, terdapat tiga peta fitur dengan skala berbeda digunakan untuk mendeteksi objek. Seperti jumlah dan skalanya peta fitur *bounding box*, ukuran batas sebelumnya kotak juga perlu disesuaikan. Yolo V3 menggunakan *k-means clustering* untuk mendapatkan ukuran kotak pembatas sebelumnya dengan mengatur tiga kotak pembatas sebelumnya untuk setiap *downsampling* skala [4].

- Yolo V5

  Algoritma Yolo V5 terdiri dari tiga modul yaitu CSP-DarkNet *backbone*, FPN+PAN *neck*, dan *prediction head*. Seperti yang ditunjukan pada Gambar 3.6, gambar dengan ukuran 640 x 640 x 3 dimasukkan ke dalam jaringan. Pada bagian *backbone*, lapisan CBS digunakan untuk *downsampling*, dan modul CSP digunakan untuk ekstraksi fitur. Setelah 5 kali *downsampling*, ukuran peta fitur menjadi 512×20×20. Terakhir, modul SPPF dihubungkan untuk mewujudkan penggabungan peta fitur dari bidang reseptif

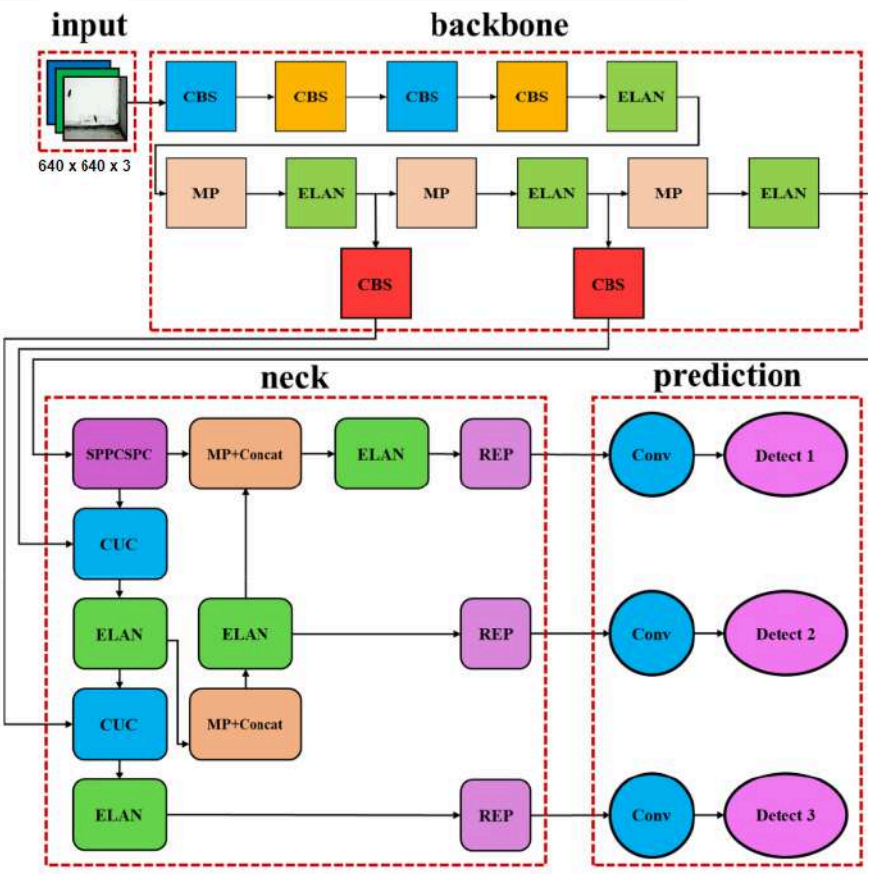
yang berbeda. Pada bagian jaringan *neck*, peta fitur pertama-tama melewati jalur pengurangan dimensi, dan kemudian melalui jalur peningkatan dimensi. Peta fitur dengan ukuran 512 × 20 × 20, 256 × 40 × 40, dan 128 × 80 × 80 digabungkan sepenuhnya melalui dua jalur. Pada bagian jaringan *head*, peta fitur dengan tiga ukuran memasuki kepala pendeteksian dan kemudian melewati lapisan konvolusi 1 × 1. Ukurannya tetap tidak berubah, dan jumlah saluran menjadi 3 × (NC + 5), di mana 3 mewakili tiga jenis kotak jangkar dengan rasio aspek yang berbeda, NC mewakili jumlah kategori, dan 5 mewakili 4 parameter yang digunakan untuk mengindikasikan posisi bingkai jangkar ditambah 1 probabilitas latar depan bingkai jangkar [5].



Gambar 3.6 Arsitektur Jaringan Yolo V5 [5]

- Yolo V7

  Model Yolo V7 mengoptimalkan arsitektur dalam proses pelatihan atau training. Yolo V7 memberikan jaringan regresi lapisan yang diperluas dan efisien serta keterampilan penskalaan model dalam arsitektur model. Dalam proses pelatihan, Yolo V7 mengadopsi model keterampilan yang di parameterisasi ulang untuk menggantikan keterampilan tersebut modul asli dan menggunakan strategi penetapan label dinamis untuk menetapkan label berbeda lapisan keluaran [6]. Gambar 3.6 menunjukkan arsitektur model Yolo V7 yang terdiri dari *input*, *backbone*, *neck*, dan *prediction*.



Gambar 3.6 *Architecture* Model Yolo V7 [7]

Di *backbone*, lapisan CBS adalah unit konvolusional dasar, termasuk konvolusi, normalisasi batch, dan fungsi aktivasi SiLU. Lapisan ELAN terdiri dari beberapa. Struktur CBS, dan keluaran

peta fiturnya terdiri dari tiga bagian. Di sini, fiturnya peta dibagi rata menjadi dua kelompok berdasarkan saluran. Kemudian kelompok pertama melaksanakan lima operasi konvolusi untuk mendapatkan bagian pertama, kelompok kedua mengimplementasikan satu operasi konvolusi untuk mendapatkan bagian kedua, dan bagian ketiga terdiri dari hasil konvolusi pertama dan konvolusi ketiga kelompok pertama. Lapisan MP membagi peta fitur menjadi dua kelompok. Grup pertama melakukan penggabungan maksimum ke mengekstrak informasi yang lebih penting, dan kelompok kedua melakukan konvolusi untuk mengekstraksi informasi fitur. Akhirnya, menggabungkan dua kelompok memperoleh hasil.

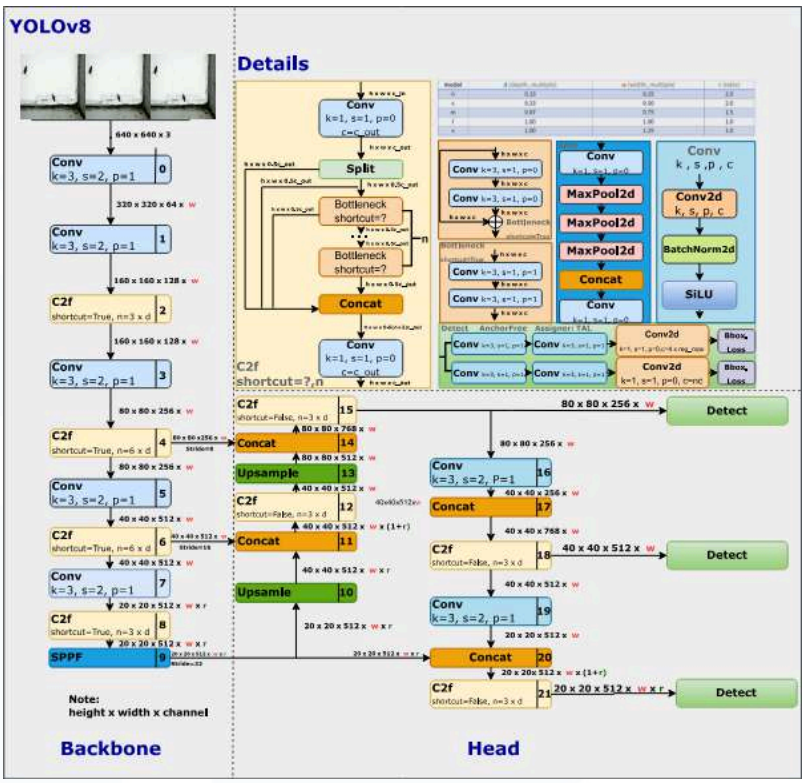
Pada bagian *neck*, keluaran peta fitur lapisan SPPCSPC terdiri dari dua bagian. Di Sini, peta fitur dibagi menjadi dua kelompok berdasarkan saluran. Kelompok pertama melaksanakan tiga operasi konvolusi dan kemudian tiga penggabungan maksimum berturut-turut untuk memisahkan tiga kumpulan peta fitur dengan ukuran berbeda untuk memperoleh banyak fitur. Setelah menggabungkan beberapa fiturnya, ia mengimplementasikan operasi konvolusi kedua untuk mendapatkan bagian pertama. Kedua grup mengimplementasikan operasi konvolusi untuk mendapatkan bagian kedua. Lapisan CUC adalah unit dasar kombinasi peta fitur, termasuk konvolusi, *up-sampling*, dan menggabungkan peta fitur. Lapisan REP adalah konsep baru yang menggunakan keterampilan struktural reparameterisasi untuk menyesuaikan struktur dalam inferensi untuk meningkatkan kinerja model. Dengan tiga bagian, lapisan REP dapat memperoleh keluaran peta fitur selama proses pelatihan. Bagian pertama dan kedua menerapkan operasi konvolusi dan normalisasi *batch*. Bagian ketiga hanya mengimplementasikan normalisasi *batch*. Kesimpulan dari REP hanya mempertahankan bagian kedua dari

struktur dengan menggunakan reparameterisasi struktural, mengurangi sumber daya komputasi, dan meningkatkan kinerja model [7].

Dalam prediksi, lapisan prediksi Yolo V7 mengeluarkan peta fitur yang sama tiga ukuran. Peta fitur berukuran kecil cocok untuk mendeteksi objek berukuran besar, dan sebaliknya, peta fitur berukuran besar cocok untuk mendeteksi objek berukuran besar. Cocok untuk mendeteksi benda kecil.

- Yolo V8

  Arsitektur Yolo V8 terdiri dari jaringan tulang punggung (*backbone network*), leher (*neck*), dan kepala (*head*) ditunjukkan pada Gambar 3.7. Jaringan tulang punggung menggunakan *Feature Pyramid Network* (FPN) untuk mengekstraksi fitur dari gambar input, sedangkan leher menggunakan serangkaian Cross-LayerConnection (CLC) untuk menyempurnakan fitur ini. Kepala mengambil fitur yang disempurnakan dan memprediksi kotak pembatas, skor kelas objek, dan akurasi untuk setiap objek dalam gambar. Memiliki 105 layer [8].



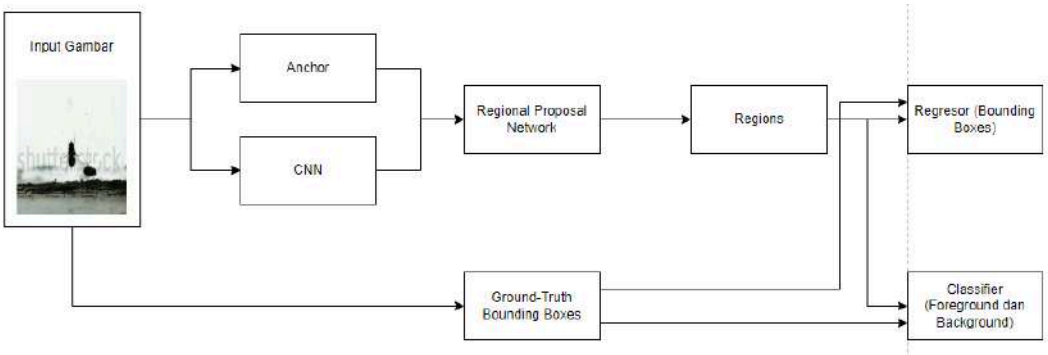
Gambar 3.7 *Architecture* Yolo V8 [9]

Proses pelatihan Yolo V8 diharapkan menjadi lebih cepat dibandingkan dengan model deteksi objek dua tahap. Pada sistem *backbone* mengalami modifikasi mengganti C3 dengan C2f dan mengintegrasikan konsep ELAN dari Yolo V7. Secara khusus, konvolusi 6x6 pertama pada stem diganti dengan konvolusi 3x3. Modul C3 adalah terdiri dari tiga ConvModules dan n DarknetBottleNecks, sedangkan modul C2f menggabungkan dua ConvModules dan n DarknetBottleNecks yang terhubung melalui *Split* dan *Concat*. ConvModule disusun dengan Conv-BN-SiLU, dan 'n' menunjukkan kuantitas *bottlenecks*. Selain itu, di C2f, keluaran dari *Bottleneck*, yang terdiri dari dua konvolusi 3x3 dengan sambungan sisa, digabungkan, sementara masuk C3, hanya keluaran dari *Bottleneck* terakhir yang digunakan. *Bottleneck* di Yolo V8 tetap sama seperti di Yolo V5, kecuali perubahan ukuran kernel konvolusi pertama dari 1x1 menjadi 3x3.

Pada sistem *head* menggunakan pendekatan yang menggabungkan *head* yang dipisahkan. *Head* yang dipisahkan yaitu *head* klasifikasi dan deteksi. Model tersebut menghilangkan cabang objektivitas, hanya mempertahankan klasifikasi dan cabang regresi. *Anchor-Base* menggunakan banyak jangkar pada gambar untuk memastikan empat *offset* regresi objek dari jangkar, menyempurnakan lokasi tepat objek dengan bantuan jangkar dan offset yang sesuai [9].

- Fast RCNN

  Fast RCNN adalah model deteksi target CNN berdasarkan usulan wilayah yang diajukan oleh Hu pada tahun 2015. Struktur dasar Fast RCNN masih berupa CNN, namun model ini mengabaikan pencarian selektif tradisional algoritma dalam proses mengekstraksi kemungkinan calon daerah, yaitu rekomendasi daerah sasaran pada gambar, lalu menambahkan jaringan konvolusi penuh (RPN) setelah peta fitur konvolusi lapisan terakhir jaringan saraf konvolusi[10].

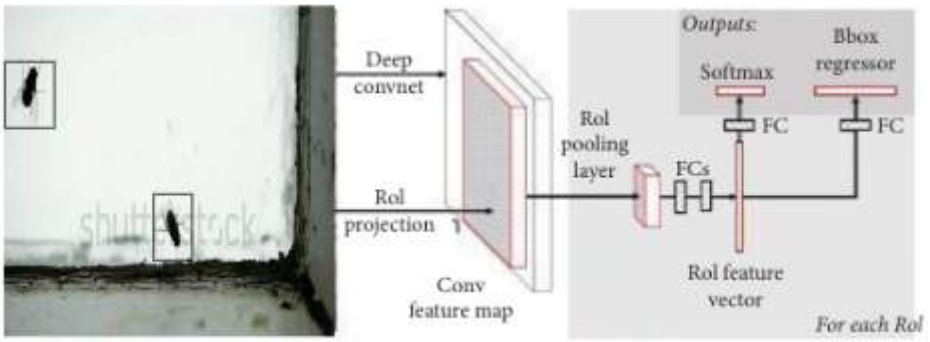


Gambar 3.8 RPN Fast RCNN

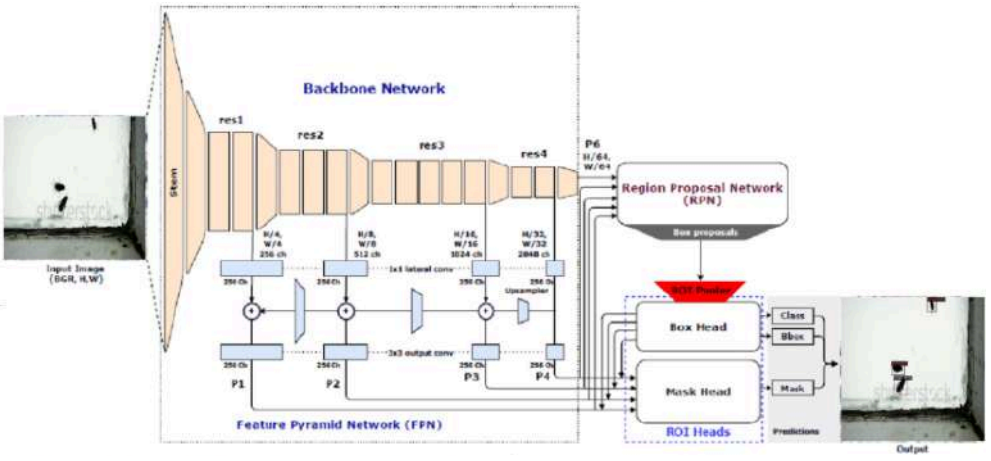
RPN atau *Region Proposal Network* ditunjukkan pada Gambar 3.8 digunakan untuk menghasilkan kotak pembatas yang berisi objek. RPN menggunakan anchor boxes sebagai titik awal menghasilkan proposal. RPN menggunakan lapisan konvolusi, lapisan regresi,

dan lapisan klasifikasi. Setiap lapisan menghasilkan koefisien untuk mengubah ukuran, posisi, dan proporsi dari kotak pembatas.

Proses pelatihan model Fast RCNN menggunakan model arsitektur Detectron2. Detectron2 digunakan untuk mempercepat siklus pelatihan. Detectron2 adalah perangkat lunak yang dikembangkan oleh *Facebook AI Research* (FAIR) yang mengimplementasikan objek canggih algoritma deteksi. Banyak model dasar terlatih tersedia di Detectron2 [11].



Gambar 3.9 *Architecture* Fast RCNN [11]



Gambar 3.10 *Architecture* Fast RCNN + Detectron 2 [12]

Dua model tulang punggung dasar yang biasa digunakan untuk R-CNN Lebih Cepat adalah R101-FPN dan X101-FPN. Kotak

R-CNN yang lebih cepat, *Average Precision* (AP) dibandingkan bidang lainnya. Struktur jaringan R-CNN Lebih Cepat terdiri dari tiga komponen utama dan ditunjukkan pada Gambar 3.9. Tiga komponen utama tersebut adalah jaringan *backbone*, yaitu jaringan usulan wilayah, dan *box heads*.

Berdasarkan Gambar 3.10 pengoperasian arsitektur jaringan ini adalah sebagai berikut:

1. Seluruh gambar input dipastikan sebagai masukan ke lapisan konvolusional Faster R-CNN untuk menghasilkan peta fitur.
2. Kemudian, untuk mengidentifikasi usulan wilayah pada peta fitur, jaringan usulan wilayah digunakan untuk memprediksi usulan wilayah.
3. Kotak jangkar yang dipilih dan peta fitur dihitung dengan model CNN awal bersama-sama diumpankan ke lapisan pengumpulan RoI untuk dibentuk kembali, dan keluaran dari RoI lapisan penyatuan dimasukkan ke dalam lapisan FC untuk klasifikasi akhir dan kotak pembatas regresi.

Bagian yang mengekstrak fitur dari masukan yang diberikan ke arsitektur jaringan adalah jaringan tulang punggung. Penelitian ini menggunakan Bidang *Feature Pyramid Network* (FPN) sebagai jaringan tulang punggung. Jaringan proposal wilayah mendeteksi kandidat kotak berdasarkan skor objektivitas (yang mewakili kemungkinan ditemukannya suatu objek di suatu wilayah) dan kotak delta (yang menyarankan pusat berdasarkan ukuran gambar masukan). Selanjutnya, RoI *heads* melakukan proses klasifikasi dan prediksi. Sebelum tahap ini, kumpulan kotak yang disarankan dilewatkan melalui lapisan pengumpulan RoI untuk menstandarkannya [12].

3.2.7 Pengaturan Hyperparameter

Training dilakukan dengan mengganti *hyperparameter* yang digunakan dapat membantu model untuk berlatih terhadap dataset yang dimiliki. Besarnya *batch size* mempengaruhi komputasi pelatihan model, namun perlu disesuaikan untuk mengurangi resiko *overfitting*. Jumlah *epochs* yang cukup penting untuk membuat model untuk belajar pola-pola dalam data secara menyeluruh. Namun, jika terlalu banyak *epoch training* yang dilakukan maka membuat model menjadi *overfitting*. Maka, *Hyperparameter* perlu diatur sedemikian rupa dan tepat untuk menghasilkan model dengan performa terbaik.

Tabel 3.2 *Hyperparameter Training* Model Yolo

Training ke-	*Hyperparameter*	
	Batch Size	*Epoch*
1	16	3
2	64	3
3	16	10
4	16	20
5	16	25
6	16	35

Pada Tabel 3.2 merupakan *hyperparameter* yang digunakan untuk *training* pada model Yolo V3, Yolo V5, Yolo V7, dan Yolo V8. *Training* pada model dilakukan sebanyak 6 kali untuk menghasilkan matrik evaluasi yang berbeda agar menunjukkan nilai yang paling optimal.

3.2.8 *Metric Evaluation*

Evaluasi metrik perlu dilakukan untuk menemukan kombinasi model yang terbaik. Evaluasi metrik dengan membandingkan indikator nilai

yang dihasilkan seperti presisi, *recall*, dan mAP. Presisi adalah perbandingan nilai prediksi benar positif dibandingkan dengan total hasil dengan prediksi positif. *Recall* adalah perbandingan nilai prediksi benar positif dengan seluruh data yang benar positif. *Mean Average Precision* (mAP) yang merupakan alat ukur yang sering digunakan dalam mengukur akurasi dari deteksi objek. Kurva dibentuk dari keterkaitan setiap deteksi *instance* dengan *ground truth* yang *overlapping*. Daerah yang masuk ke dalam IoU dengan *ground* yang memiliki nilai di atas ambang batas bisa dikategorikan sebagai *True Positive*, untuk yang lain yang tidak masuk bisa dikategorikan *False Positive*. Nilai *Precision*, *Recall*, dan mAP 0.5 dapat dicari menggunakan persamaan (1), (2), dan (3) [13].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$\text{mAP } 0.5 = \frac{1}{N}\sum_{i=1}^{N} APi \quad (3)$$

3.2.9 Hasil dan Analisis

Setelah dilakukan *training* pada setiap model, maka didapatkan hasil nilai matrik yang berbeda-beda.

- Yolo V3

  Tabel 3.3 menunjukkan hasil eksekusi *training* Yolo V3 yang berisi mengenai *precision*, *recall*, mAP, dan waktu yang dibutuhkan untuk eksekusi.

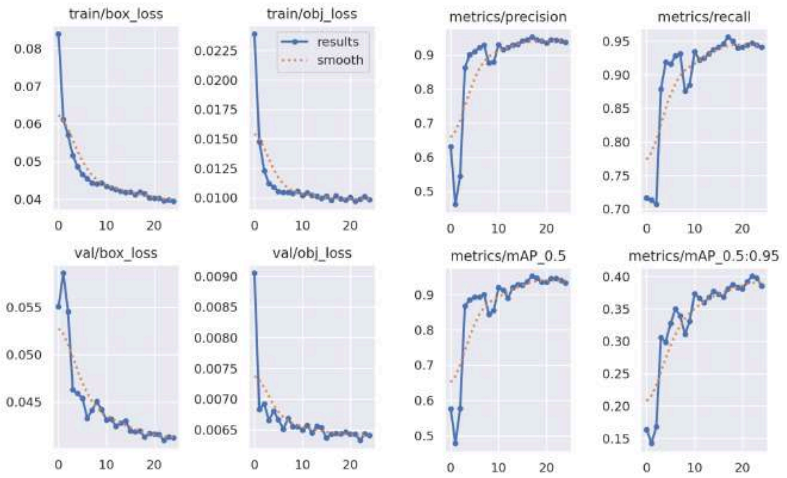
Tabel 3.3 Hasil Eksekusi *Training* Yolo V3

Training ke-	Hasil			Waktu
	P	R	mAP	
1*	0.7888	0.801	0.789	6 jam

1	0.7888	0.801	0.789	4 menit
2	0.576	0.757	0.57	3 menit
3	0.920	0.935	0.913	8 menit
4	0.932	0.944	0.942	16 menit
5	0.943	0.947	0.946	20 menit
6	0.943	0.947	0.946	30 menit

Keterangan 1* : *Training* dengan menggunakan *Hardware CPU Google Collab*

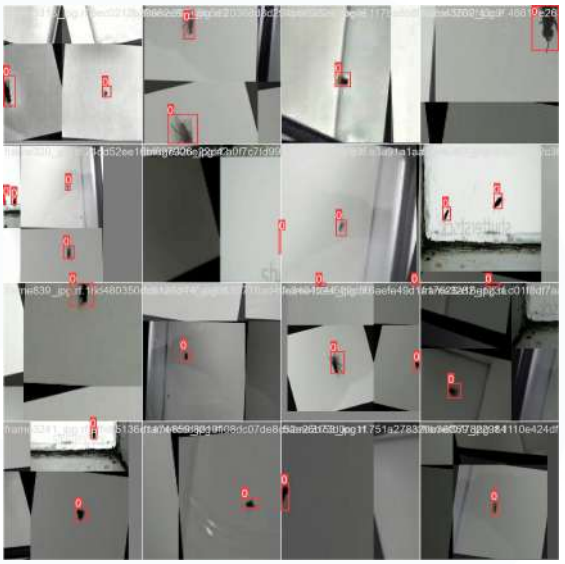
Berdasarkan Tabel 3.3 dapat dilihat bahwa *Training* ke-5 dengan *hyperparameter batch size* 16 dan *epoch* sebanyak 20 menghasilkan nilai precision 0.943, recall 0,947, dan mAP 0,946 dengan eksekusi memakan waktu 16 menit. Nilai tersebut termasuk tinggi sehingga menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi objek dan meminimalkan kesalahan. Hal ini dibuktikan dari visualisasi grafik hasil *training* pada 3.16 dibawah ini.



Gambar 3.11 Hasil *Training* Yolo V3

Berdasarkan Gambar 3.11 dapat dilihat bahwa grafik *training* dan *validation box loss* menunjukkan penurunan ketika pelatihan model seiring bertambahnya *epoch* karena model sedang menyesuaikan dataset. Begitu dengan grafik *training* dan *validation object loss* mengalami penurunan. Sedangkan pada grafik matrik nilai *precision*, *recall*, mAP, dan mAP IoU 0,5:0,95

mengalami peningkatan dan menunjukkan nilai yang stabil. Hal ini dibuktikan pada Gambar 3.12 sebagai hasil training Yolo V3.



Gambar 3.12 Hasil *Training* Yolo V3

- Yolo V5

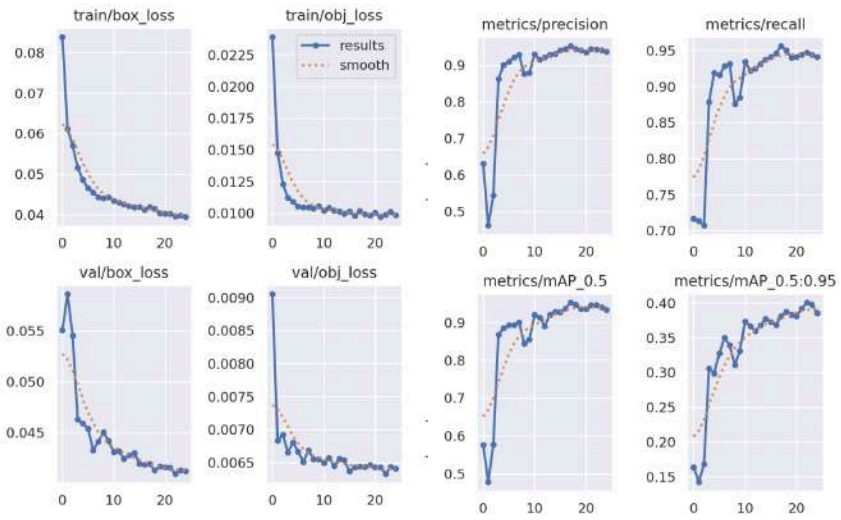
Tabel 3.4 menunjukkan hasil eksekusi *training* Yolo V5 yang berisi mengenai *precision*, *recall*, mAP, dan waktu yang dibutuhkan untuk eksekusi.

Tabel 3.4 Hasil Eksekusi *Training* Yolo V5

Training ke-	Hasil			Waktu
	P	R	mAP	
1	0.788	0.801	0.789	3 menit
2	0.576	0.757	0.57	3 menit
3	0.920	0.937	0.906	7 menit
4	0.943	0.947	0.946	16 menit
5	0.944	0.947	0.946	17 menit
6	0.940	0.948	0.944	29 menit

Berdasarkan Tabel 3.4 dapat dilihat bahwa *Training* ke-5 dengan *hyperparameter batch size* 16 dan *epoch* sebanyak 25 menghasilkan nilai *precision* 0.944, *recall* 0,947, dan mAP 0,946

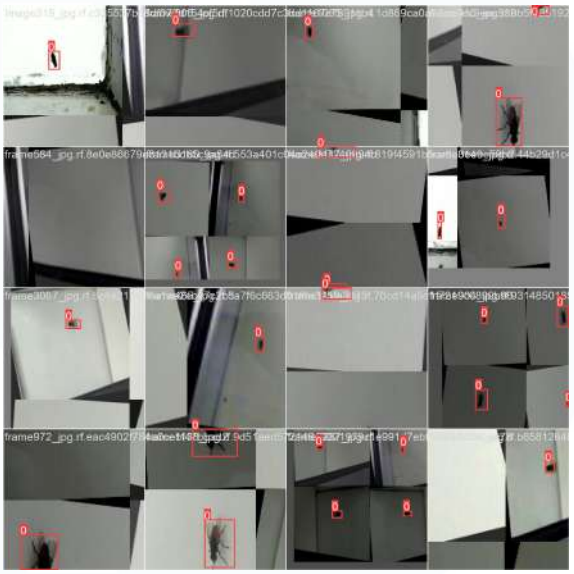
dengan eksekusi memakan waktu 17 menit. Nilai tersebut termasuk tinggi sehingga menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi objek dan meminimalkan kesalahan. Hal ini dibuktikan dari visualisasi grafik hasil training pada Gambar 3.13 dibawah ini.



Gambar 3.13 Grafik *Training* Yolo V5

Berdasarkan Gambar 3.13 dapat dilihat bahwa pada grafik perubahan nilai *training loss* selama pelatihan model terlihat menurun secara bertahap seiring bertambahnya *epoch* pelatihan. Sementara, *validation loss* selama proses pelatihan model awalnya menurun tetapi kemudian cenderung mendatar pada *epoch* selanjutnya. Grafik *metric precision* terlihat bahwa nilai *precision* meningkat selama beberapa *epoch* awal, kemudian stabil pada nilai sekitar 0,9 untuk epoch-epoch selanjutnya. Grafik *metric recall* selama pelatihan terlihat meningkat cukup signifikan pada epoch-epoch awal, kemudian stabil pada nilai 0.7 setelah melewati sekitar 10 epoch pelatihan. Grafik nilai metric mAP 0.5 selama training terlihat meningkat pada awal pelatihan, kemudian menjadi relatif stabil pada nilai sekitar 0,8 setelah beberapa epoch. Grafik metric mAP 0.5:0.95 yang lebih rinci terlihat meningkat di awal

pelatihan, lalu menjadi stabil pada kisaran 0,9 hingga 0,95 setelah beberapa epoch pelatihan. Dapat dilihat pada Gambar 3.8 menunjukkan bahwa keberhasilkan model dalam training sehingga mampu mendeteksi *object* dengan tepat.



Gambar 3.19 Hasil *Training* Yolo V5

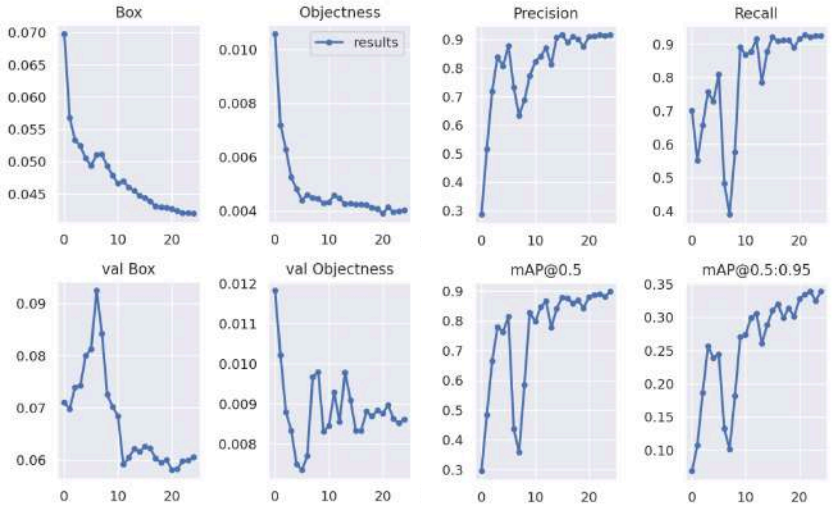
- Yolo V7

Tabel 3.5 menunjukkan hasil eksekusi *training* Yolo V7 yang berisi mengenai *precision, recall*, mAP, dan waktu yang dibutuhkan untuk eksekusi.

Tabel 3.5 Hasil Eksekusi *Training* Yolo V7

Training ke-	Hasil			Waktu
	P	R	mAP	
1	0.859	0.779	0.771	7 menit
2	0.670	0.556	0.577	8 menit
3	0.902	0.912	0.884	22 menit
4	0.920	0.919	0.860	41 menit
5	0.923	0.927	0.894	54 menit
6	0.923	0.927	0.894	63 menit

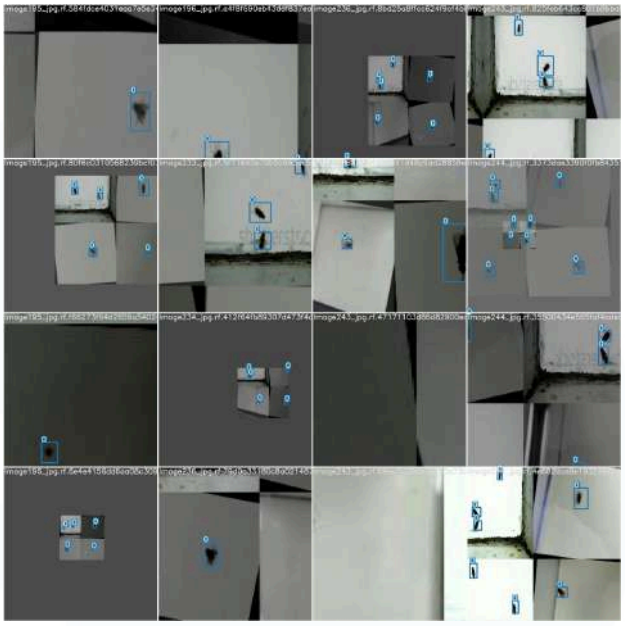
Berdasarkan Tabel 3.5 dapat dilihat bahwa *Training* ke-5 dengan *hyperparameter batch size* 16 dan *epoch* sebanyak 25 menghasilkan nilai *precision* 0.923, *recall* 0,927, dan mAP 0,894 dengan eksekusi memakan waktu 54 menit. Nilai tersebut termasuk tinggi sehingga menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi objek dan meminimalkan kesalahan. Kemudian, dapat dilihat pada Gambar 3.20 merupakan visualisasi grafik hasil *training*.



Gambar 3.20 Hasil *Training* Yolo V7

Pada Gambar 3.20 menunjukkan grafik hasil pelatihan model Yolo V7 yang memiliki performa terbaik. Pada *box training loss* menunjukkan grafik menurun secara bertahap seiring meningkatnya *epoch* karena pada awalnya model belum mempelajari pola data dengan tepat, seiring berjalannya epoch model belajar dan menyesuaikan parameter agar lebih baik dalam memprediksi kotak pembatas objek sehingga membantu model mencapai nilai *loss* yang paling rendah seiring berjalan waktu. Grafik *box validation loss* mengalami peningkatan kemudian penurunan seiring dengan peningkatan jumlah epoch karena terjadi *overfitting* awal sehingga model sedang menyesuaikan diri dengan

regularisasi yang diterapkan. Grafik *objectness training* mengalami penurunan saat peningkatan jumlah epoch karena model menjadi lebih baik dalam mempelajari representasi objek dalam gambar mencakup fitur-fitur yang digunakan. Grafik *objectness validation loss* mengalami peningkatan kemudian penurunan karena adanya *overfitting* dalam data pelatihan diawal yang tidak general, namun model mulai mempelajari representasi yang lebih umum dari data. Grafik tersebut mulai stabil dinilai 0.08. Grafik *metric precision* terlihat bahwa nilai *precision* meningkat selama beberapa epoch awal, kemudian stabil pada nilai sekitar 0,9. Grafik metric *recall* mengalami peningkatan dan penurunan karena model awalnya mempelajari fitur-fitur yang spesifik pada data pelatihan yang tidak umum atau tidak relevan dengan data validasi. Grafik *recall* stabil pada nilai sekitar 0,9 seiring bertambahnya epoch. Grafik nilai metric mAP pada ambang batas 0,5 selama *training* terlihat meningkat dan menurun pada awal pelatihan, kemudian menjadi relatif stabil pada nilai sekitar 0,8 setelah beberapa epoch. Grafik nilai metric mAP pada ambang batas IoU dari 0,5 hingga 0,95 meningkat dan menurun di awal pelatihan, lalu menjadi stabil pada kisaran 0,9 setelah beberapa epoch pelatihan. Dapat dilihat pada Gambar 3.21 hasil training Yolo V7.



Gambar 3.21 Hasil *Training* Yolo V7

- Yolo V8

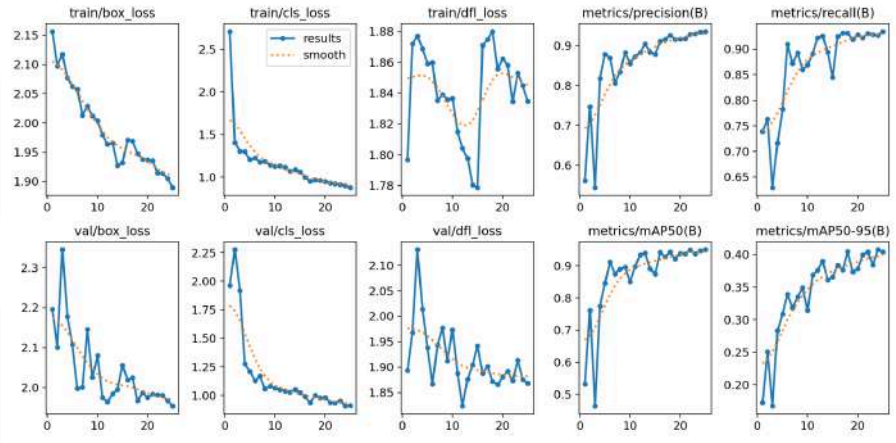
Tabel 3.6 menunjukkan hasil eksekusi *training* Yolo V8 yang berisi mengenai *precision, recall*, mAP, dan waktu yang dibutuhkan untuk eksekusi.

Tabel 3.6 Hasil Eksekusi *Training* Yolo V8

Training ke-	Hasil			Waktu
	P	R	mAP	
1	0.853	0.919	0.901	3 menit
2	0.901	0.925	0.925	12 menit
3	0.919	0.925	0.930	30 menit
4	0.934	0.927	0.947	32 menit
5	0.934	0.927	0.948	41 menit

Berdasarkan Tabel 3.6 dapat dilihat bahwa Training ke-5 dengan *hyperparameter batch size* 16 dan *epoch* sebanyak 25 menghasilkan nilai *precision* 0.934, *recall* 0,927, dan mAP 0,947 dengan eksekusi memakan waktu 32 menit. Nilai tersebut termasuk tinggi sehingga menunjukkan bahwa model memiliki kemampuan

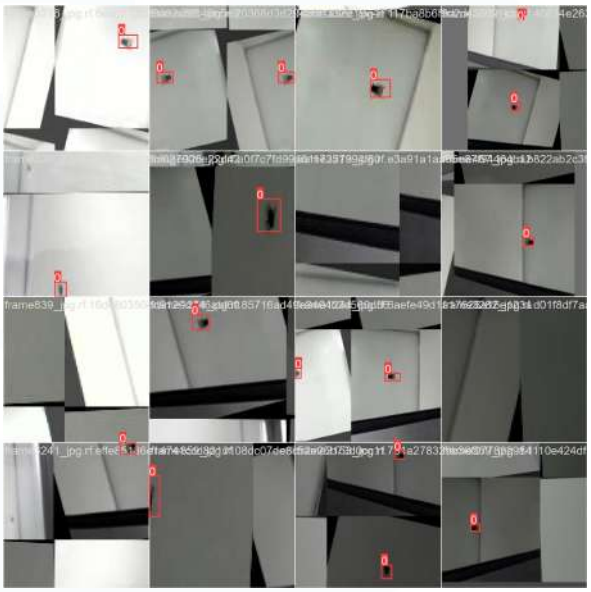
yang baik dalam mendeteksi objek dan meminimalkan kesalahan. Hal ini dibuktikan dari visualisasi grafik hasil *training* pada Gambar 3.22 dibawah ini.



Gambar 3.22 Grafik Training Yolo V8

Gambar 3.22 menunjukkan grafik selama pelatihan pada model Yolo V8 dalam performa yang paling optimal. Grafik training *box loss* selama pelatihan terlihat menurun secara bertahap karena model sedang melakukan penyesuaian terhadap dataset. Grafik *validation box loss* selama proses pelatihan mengalami naik turun namun mulai stabil pada epoch selanjutnya yang nilainya kurang dari 2. Grafik *training* dan *validation class loss* menunjukkan nilai penurunan yang signifikan hal ini membuktikan bahwa model mampu memahami kondisi class yang digunakan dalam proses pelatihan. Grafik *training deep learning framework* (DFL) mengalami naik turun saat pelatihan ini menunjukkan bahwa model awalnya *overfitting* menghafal data kemudian, seiring bertambahnya epoch model mulai menyesuaikan dengan mempelajari pola dalam data. Grafik validation dfl loss menunjukkan penurunan meskipun sempat meningkat, namun mulai turun stabil dinilai 1.90. Grafik *precision* dan *recall* menunjukkan peningkatan seiring bertambahnya epoch, dan stabil pada nilai sekitar 0.9. Grafik nilai metric mAP pada ambang batas

0,5, dan nilai metric mAP pada ambang batas IoU dari 0,5 hingga 0,95 menunjukkan peningkatan seiring bertambahnya epoch, dan stabil pada nilai sekitar 0.9. Dapat dilihat pada Gambar 3.23 hasil training Yolo V8.



Gambar 3.23 Hasil *training* Yolo V8

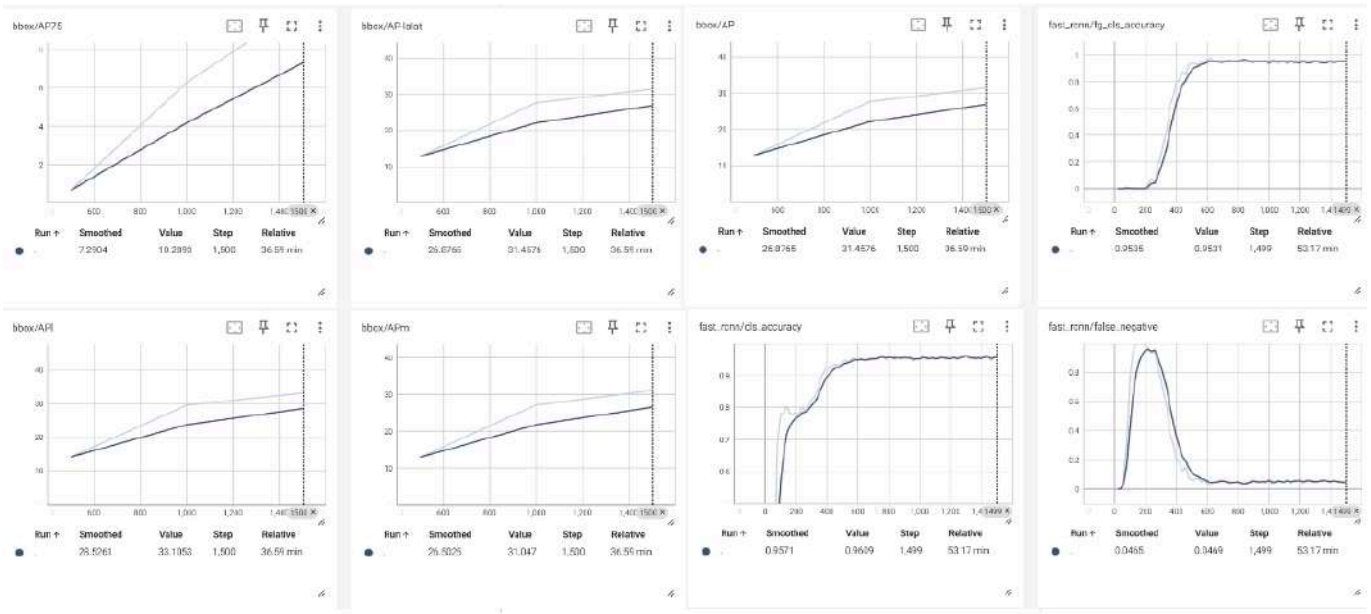
- Fast RCNN

Tabel 3.7 Hasil *Training* Fast RCNN

Iteration 1,000			
P	IoU 0.50	IoU 0.75	IoU
	49.415%	0.661%	12.866%
R	IoU (Det.1)	IoU (Det.10)	IoU (Det.100)
	20.3%	38.8%	41.1%
mAP	12.866%		
Iteration 2,000			
P	IoU 0.50	IoU 0.75	IoU

	84.729%	6.270%	27.648%
R	IoU (Det.1)	IoU (Det.10)	IoU (Det.100)
	32.3%	42.6%	43.4%
mAP	27.648%		
Iteration 3,000			
P	IoU 0.50	IoU 0.75	IoU
	87.2%	10.3%	31.5%
R	IoU (Det.1)	IoU (Det.10)	IoU (Det.100)
	34.5%	46.6%	46.6%
mAP	31.5%		

Berdasarkan Tabel 3.7 selama proses pelatihan nilai dari matrik evaluasi tersebut mengalami peningkatan. Hal ini juga dibuktikan dari grafik pada Gambar 3.24.



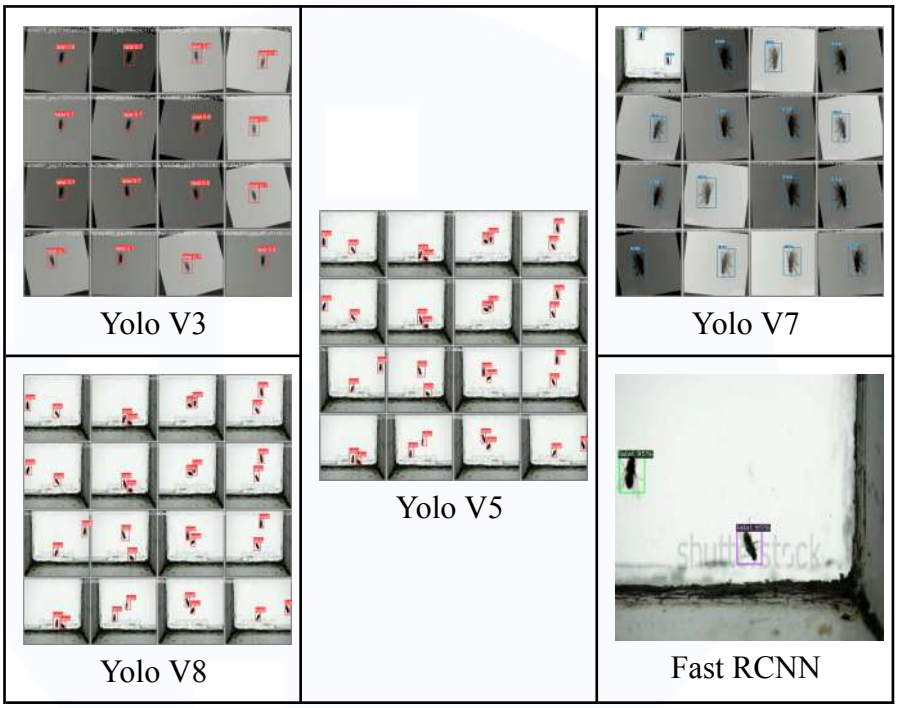
Gambar 3.24 Grafik Training Fast RCNN

Berdasarkan gambar 3.14 dapat dilihat bahwa nilai *average precision* mengalami peningkatan seiring bertambahnya pelatihan. Begitu juga dengan nilai akurasi mengalami peningkatan yang sangat signifikan. Grafik yang mengalami penurunan tersebut adalah grafik yang menunjukkan kemampuan model mendeteksi kesalahan dalam objek atau *false negative*.

3.2.10 Perbandingan Model

Perbandingan ini menggunakan model yang memiliki hasil yang terbaik. Dari hasil *training* dan analisis matrik evaluasi yang diberikan maka selanjutnya dilakukan perbandingan hasil deteksi dari setiap model Maka diperoleh hasil deteksi yang ditunjukkan pada tabel 3.2 dibawah ini.

Gambar 3.25 Hasil Deteksi Model



Berdasarkan Gambar 3.25 dapat dilihat bahwa masing-masing model mampu mendeteksi objek dengan tepat. Namun, untuk menentukan model yang tepat guna maka dilakukan perbandingan dengan nilai matrik evaluasi dari performa terbaik masing-masing model. Hasil perbandingan model ditunjukkan pada Tabel 3.12.

Tabel 3.8 Perbandingan Model

Model	Matrik Evaluasi			Waktu Deteksi
	P	R	mAP	
Yolo Y3	0.943	0.947	0.946	48.9
Yolo V5	0.943	0.947	0.946	12 ms
Yolo V7	0.923	0.927	0.894	0.8 s
Yolo V8	0.934	0.927	0.947	24 ms
Fast RCNN	0.872	0.315	0.345	15 ms

Pada Tabel 3.8 dapat dilihat bahwa mode yang memiliki performa paling optimal yaitu Yolo V5. Model tersebut menunjukkan memiliki nilai P, R, dan mAP yang tinggi dengan waktu deteksi yang relatif singkat. Hal ini juga dapat dilihat bahwa komputasi training pada model tersebut juga tergolong ringan jika dibandingkan dengan model yang lain.

3.3 Kendala yang Ditemukan

Kendala yang dihadapi selama melakukan rancang bangin pengembangan smart farming ini adalah:

1. Pada awalnya solusi proposal EPICS in IEEE yang ditawarkan berupa identifikasi dan klasifikasi untuk jenis hama yang menyerang di pertanian salak. Namun, kenyataannya ketika setelah melakukan kunjungan ke lokasi Sleman, Yogyakarta ternyata dataset hama susah untuk ditemukan. Terutama untuk jenis tikus dan tupai.
2. Ukuran model dan dataset yang besar sehingga menyebabkan proses pelatihan tidak dapat dilakukan menggunakan laptop pribadi maupun Google Collab free version.

3.4 Solusi atas Kendala yang Ditemukan

Adapun solusi untuk menghadapi kendala yang ditemukan antara lain:

1. Mengubah sistem yang awalnya untuk menjadi deteksi otomatis pada lalat yang ditangkap oleh petani pada perangkap pertanian yang telah dibuat.
2. Menggunakan akun *Pro Google Collab* milik Supervisi selama proses pengerjaan pelatihan agar memudahkan dalam pengerjaan dan tidak memakan banyak waktu komputasi saat pelatihan model.