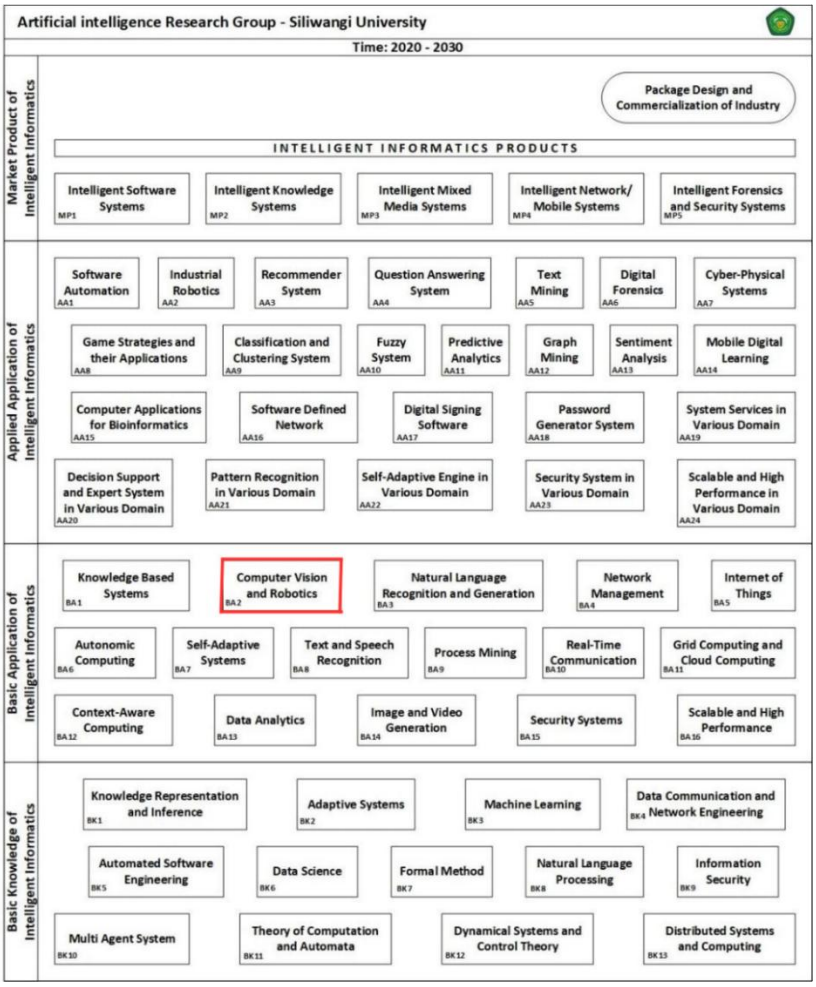
BAB III

METODOLOGI

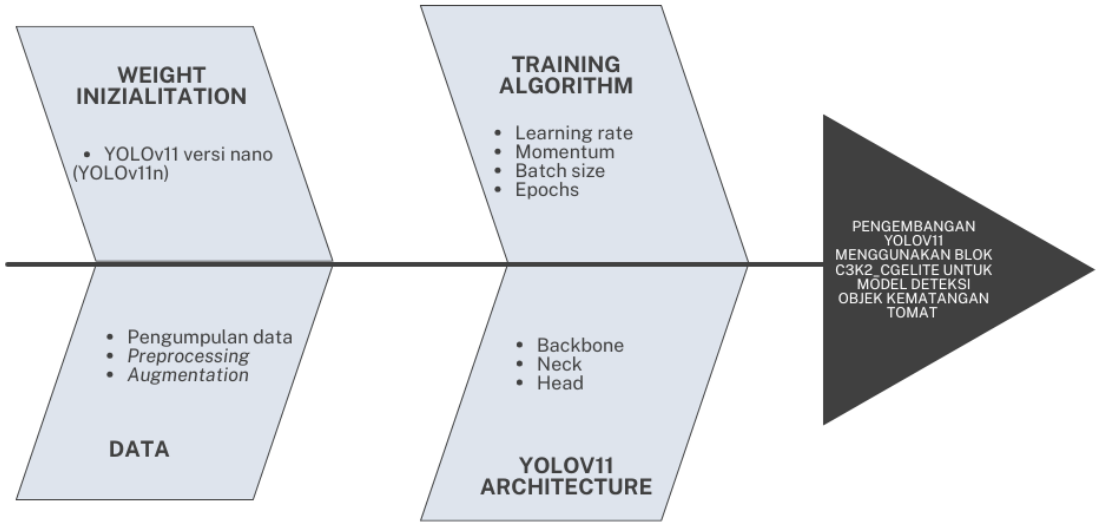
3.1 Peta Jalan (*Roadmap*) Penelitian

Penelitian ini memiliki *roadmap* yang selaras dengan peta jalan penelitian Universitas Siliwangi di bidang *artificial intelligence*. Fokus penelitian ini adalah pada *computer vision and robotics*, yang termasuk dalam ranah *basic application of intelligent informatics*. Peta jalan tersebut disajikan dalam Gambar 3.1.



Gambar 3. 1 Peta jalan penelitian (AIS Universitas Siliwangi, 2024)

Fokus penelitian terletak pada pengembangan model deteksi objek berbasis *deep learning* yang efisien dan mempertahankan akurasi melalui modifikasi model *pretained* YOLOv11 menggunakan *block* baru yaitu C3k2_Lite. Secara keilmuan, penelitian ini berkontribusi pada penguatan fondasi penerapan *intelligent visual system* melalui pengembangan algoritma pada *block* baru C3k2_Lite. Hasil yang diperoleh dapat menjadi dasar bagi pengembangan lebih lanjut pada sistem *computer vision and robotics,* serta mendukung implementasi teknologi *smart agriculture* di masa mendatang. Gambar 3.2 menyajikan diagram *fishbone* penelitian sebagai turunan dari bidang yang dipilih dalam peta jalan penelitian yang ditampilkan pada Gambar 3.1 yang terinspirasi dari (Kumah dkk., 2024).



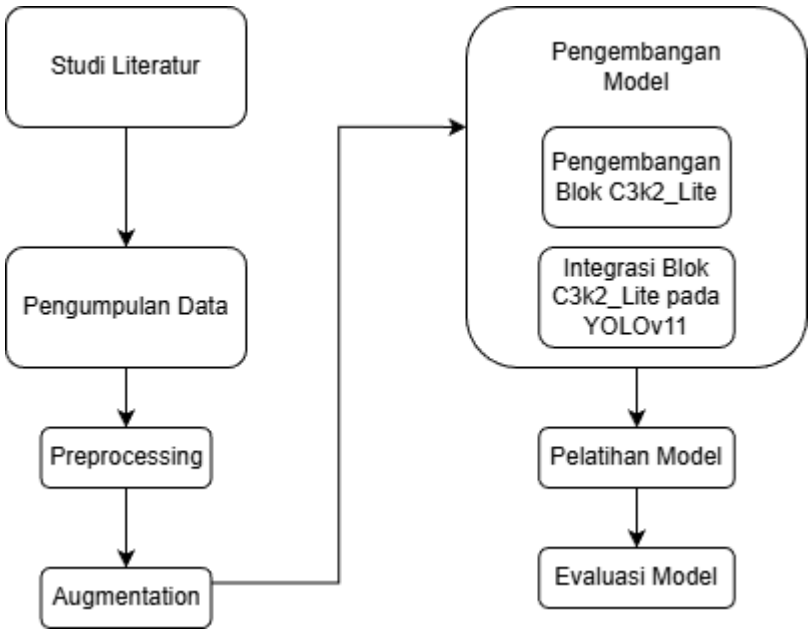
Gambar 3. 2 Diagram *fishbone* untuk menunjukkan proses pembuatan sistem

Langkah penelitian dimulai dari menentukan kebutuhan yang perlu di siapkan sebagai langkah awal setiap proses yang akan dilakukan. Berikutnya, adalah proses pengolahan data dilakukan untuk mengumpulkan data dengan objek yang sudah ditentukan di tahapan sebelumnya, lalu dilakukan proses *augmentation* pada data

yang sudah terkumpul. Selanjutnya, adalah dilakukan pengembangan model dengan memodifikasi arsitektur YOLOv11 dengan *block* baru yaitu C3k2_Lite. Setelah dilakukan modifikasi pada model maka tahap berikutnya adalah melakukan evaluasi kinerja model yang sudah dikembangkan, menggunakan evaluasi matriks mAP50, *recall, precision, inference time,* dan beban memori.

3.2 Tahapan Penelitian

Tujuan penelitian ini adalah mengembangkan model dengan kemampuan mendeteksi tingkat kematangan buah tomat yang efisien dan akurat. Strategi yang digunakan adalah memodifikasi arsitektur model YOLOv11 menggunakan *block* baru yaitu C3k2_Lite. Maka dari itu terbentuklah tahapan penelitian yang akan digunakan pada penelitian ini sebagaimana pada Gambar 3.3.



Gambar 3. 3 Metodologi penelitian dalam pengembangan sistem deteksi kematangan pada buah tomat

3.2.1 Studi Literatur

Proses ini pertama kali dilakukan dengan membaca *survey* paper yang berkaitan dengan penelitian yang dilakukan. Selanjutnya diikuti dengan langkah mencari dan membaca *technical* paper yang relevan dan ter indeks *scopus* jika berasal dari jurnal internasional dan ter indeks SINTA jika bersumber dari jurnal nasional, sehingga hasil dari pengetahuan yang di ekstrak dapat menciptakan *state of the art* yang mendukung jalannya penelitian ini.

3.2.2 Pengumpulan Data

Data diambil dari *paper* yang berjudul "*A dataset of tomato fruits images for object detection in the complex lighting environment of plant factorie*s" milik (Wu dkk., 2023). Data diunduh melalui mendeley *dataset repository* pada tanggal 12 Maret 2025. Dataset ini memiliki gambar berjumlah 499 gambar dengan 2 kelas yaitu 0 (*green*/*unripe*) dan 1 (*red*/*ripe*). Data ini dikumpulkan di *Laboratorium Artificial Light Plant Factory di Henan Institute of Science and Technology (HIST*), yang berlokasi di *Xinxiang, Tiongkok.* Menggunakan pencahayaan buatan, tempat tomat dibudidayakan. Pengambilan gambar buah tomat dilakukan pada berbagai siklus, mulai dari fase pembungaan tomat pada Desember 2021 hingga saat terdapat banyak buah tomat yang matang pada Februari 2022. Tujuannya adalah agar data dapat beragam dan mencakup berbagai kondisi dari buah tomat.

Pembagian *dataset* pada penelitian ini memilih rasio 80:10:10, yaitu 80% sebagai data untuk proses pelatihan model, 10% untuk memvalidasi hasil latihan

model, 10% untuk data uji model. Rasio tersebut mengikuti set *default* yang digunakan pada YOLOv11.

3.2.3 *Preprocessing*

Tahapan ini memiliki tujuan utama untuk menstandarkan seluruh data masukan agar sesuai dengan kebutuhan arsitektur model dan mendukung proses pelatihan yang efisien serta stabil. YOLOv11 merupakan model deteksi objek berbasis *single-stage architecture* yang menggunakan *grid prediction system* oleh karena itu, ukuran *input* yang tidak seragam dapat menyebabkan dimensi pada *feature map* tidak konsisten, sehingga dapat memperlambat proses pelatihan. Penlitian ini melakukan metode *resized* 640x480 dikarenakan tidak terlalu tinggi dan juga tidak terlalu rendah untuk resolusinya. Namun, tetap stabil saat proses pelatihan model.

3.2.4 *Augmentation*

Augmentation gambar dilakukan guna meningkatkan keberagaman data dengan tidak membuat kelas yang baru. Tahapan ini melibatkan penggunaan rotasi gambar, penambahan eksposur pada gambar, *hue*, *noise*. Sehingga dapat berguna dalam meningkatkan pemahaman model serta akurasi.

3.2.5 Pengembangan Model Usulan

Pada tahap ini dilakukan proses pengembangan model deteksi objek berbasis YOLOv11 yang dimodifikasi untuk mendeteksi kematangan buah tomat. Proses pengembangan model meliputi tahapan pemilihan arsitektur dasar, integrasi

blok baru, serta penyesuaian konfigurasi agar sesuai dengan karakteristik dataset yang digunakan. Model YOLOv11 dipilih karena memiliki keseimbangan optimal antara akurasi deteksi dan efisiensi komputasi, serta telah terbukti unggul dalam tugas deteksi objek pada lingkungan kompleks.

Modifikasi dilakukan untuk meningkatkan kemampuan model dalam mengekstraksi fitur penting dari objek target (buah tomat) pada lingkungan dunia nyata. Untuk mencapai tujuan tersebut, dilakukan penggantian blok C3k2 yang semula digunakan pada *backbone* dan *neck* arsitektur YOLOv11 dengan blok baru yang dikembangkan, yaitu C3k2_Lite. Pengembangan model ini melibatkan dua aspek utama, yaitu:

1. Desain Blok Baru (C3k2_Lite): perancangan struktur blok yang lebih ringan dengan memanfaatkan mekanisme *cross-gating enhancement*.
2. Integrasi ke Arsitektur YOLOv11: modifikasi struktur *backbone* dan *neck* untuk menggantikan blok lama dengan blok baru tanpa mengubah fungsionalitas utama arsitektur.

3.2.5.1 Pengembangan Blok C3k2_Lite

Blok C3k2_Lite dikembangkan sebagai varian ringan (*lightweight variant*) dari blok C3k2 pada arsitektur YOLOv11. Tujuan utama pengembangannya adalah untuk meningkatkan efisiensi komputasi dan memperkuat kemampuan ekstraksi fitur kontekstual, khususnya pada lingkungan dengan kondisi latar belakang yang kompleks dan pencahayaan bervariasi, tanpa mengorbankan akurasi deteksi.

Secara struktural, blok C3k2_Lite memadukan tiga komponen utama, yaitu *FastContextUnit* (FCU), *Cross-Gating Enhancement* (CGE), dan *Shortcut Connection*. Kombinasi ketiganya membentuk alur pemrosesan fitur yang adaptif, efisien, dan stabil selama proses pelatihan.

Fast Context Unit (FCU) berperan sebagai unit ekstraksi fitur berkecepatan tinggi dan berbiaya rendah. Unit ini memanfaatkan *depthwise convolution* dengan *kernel* 3×3 untuk menangkap fitur lokal, serta *depthwise convolution* ber-*kernel* lebih besar atau *dilated convolution* untuk memperluas jangkauan reseptif dan memperoleh konteks spasial yang lebih luas. Hasil dari kedua konvolusi tersebut kemudian digabungkan (*concatenate*) dan diproyeksikan kembali menggunakan *pointwise convolution* (1×1) agar dimensi fitur tetap konsisten. Pendekatan ini memungkinkan FCU untuk menangkap informasi *multiscale* secara efisien, menekan jumlah parameter dan operasi (*FLOPs*), serta meningkatkan sensitivitas model terhadap variasi warna objek.

Setelah fitur diproses oleh FCU, mekanisme *Cross-Gating Enhancement* (CGE) diterapkan untuk memperkuat fitur yang relevan (selektivitas fitur) dengan objek target dan menekan gangguan dari latar belakang gambar. Mekanisme ini bekerja dengan cara menghasilkan sinyal *gating* adaptif dari kombinasi dua aliran fitur berbeda. Sinyal *gate* dibentuk melalui konvolusi 1×1 dan proses *average pooling multiscale* (skala 1, 2, dan 4), kemudian di-*upsample* dan dijumlahkan untuk menghasilkan representasi *attention* global. Selanjutnya, hasil tersebut diproyeksikan kembali dan dilewatkan ke fungsi *aktivasi sigmoid* untuk menghasilkan peta *gating* yang bernilai antara 0 dan 1. Peta *gate* ini digunakan

untuk mengalikan masing-masing fitur *input* secara elemen per elemen (*element-wise multiplication*), sehingga hanya fitur yang memiliki kontribusi tinggi terhadap proses deteksi yang diperkuat. Sementara itu, sebagian kanal yang tidak mengalami *gating* akan dilewatkan secara identitas (*partial gating*), guna menjaga kelengkapan informasi awal.

Dalam menjaga kestabilan pelatihan dan menghindari hilangnya informasi penting, blok C3k2_Lite dilengkapi dengan *shortcut connection* atau *residual path*. Mekanisme ini memungkinkan aliran informasi dari input awal untuk langsung diteruskan ke output akhir dengan cara melakukan penjumlahan (*element-wise addition*) antara *input* F_{in} dan hasil transformasi fitur ($F(F_{in}, W)$. Dengan demikian, gradien dapat mengalir lebih baik selama proses *backpropagation*, yang pada akhirnya mempercepat konvergensi pelatihan dan meningkatkan stabilitas model.

Secara keseluruhan, proses pada blok C3k2_Lite diawali dengan proyeksi awal melalui konvolusi 1×1 (*CBA layer*), diikuti oleh pengolahan konteks *multiscale* menggunakan [FCU]×n, kemudian diperkuat dengan mekanisme CGE, dan akhirnya diproyeksikan kembali melalui konvolusi 1×1 sebelum dihubungkan dengan jalur *shortcut*. Alur ini dapat dirumuskan sebagai berikut yang diturunkan dari persamaan matematis milik (J. Hu dkk, 2020) dan (He dkk., 2016).

$$F_{out} = F_{in} + \sigma\left(W_g \times F(F_{in})\right) \odot F(F_{in}) \quad (1)$$

Secara matematis, blok C3k2_Lite dirancang untuk memproses fitur masukan dengan cara yang efisien dan adaptif. Pada persamaan yang digunakan,

F_{in} merepresentasikan fitur *input* yang masuk ke dalam blok, sedangkan $F(F_{in})$ merupakan hasil transformasi fitur yang diperoleh melalui rangkaian *FastContextUnit (FCU)* dan konvolusi 1×1 yang berfungsi mengekstraksi serta memproyeksikan fitur. Parameter W_g menunjukkan bobot konvolusi yang digunakan dalam mekanisme *gating* untuk menghasilkan sinyal penguatan adaptif. Fungsi σ adalah fungsi *aktivasi sigmoid* yang memetakan nilai sinyal *gate* ke dalam rentang [0,1], sehingga memungkinkan pengendalian tingkat penguatan fitur secara halus. Operator ⊙ merepresentasikan perkalian elemen per elemen (*element-wise multiplication*), yang digunakan untuk mengalikan sinyal *gate* dengan fitur hasil transformasi. Sementara itu, F_{out} menunjukkan fitur keluaran akhir yang dihasilkan setelah proses integrasi antara mekanisme *gating* dan *shortcut connection*.

Rancangan ini memungkinkan blok C3k2_Lite untuk menghasilkan fitur yang lebih informatif, kontekstual, serta adaptif terhadap berbagai variasi lingkungan, sekaligus mempertahankan efisiensi memori dan waktu inferensi pada proses deteksi objek.

3.2.5.2 Integrasi Blok C3k2_Lite Pada Arsitektur YOLOV11

Proses modifikasi dilakukan dengan mengganti seluruh blok C3k2 yang terdapat pada *backbone* dan *neck* arsitektur YOLOv11 dengan blok C3k2_Lite. *Backbone* bertugas mengekstraksi fitur dari citra input, sedangkan neck berfungsi menggabungkan fitur dari berbagai skala untuk meningkatkan kemampuan deteksi terhadap objek dengan ukuran berbeda. Tahapan modifikasi meliputi:

1. Analisis Struktur YOLOv11: Mengidentifikasi posisi blok C3k2 pada konfigurasi *backbone* dan *neck*.
2. Implementasi Blok C3k2_Lite: Menggantikan setiap *instance* C3k2 dengan C3k2_Lite melalui pembaruan berkas konfigurasi yolov11.yaml, *block.py*, *tasks.py,* dan modul *__init__.py.*

Dengan demikian, modifikasi ini tidak hanya bertujuan untuk meningkatkan efisiensi dan mempertahankan akurasi, tetapi memastikan juga kompatibilitas dan stabilitas model selama proses pelatihan dan inferensi. Hasil modifikasi bertujuan untuk mempertahankan nilai akurasi dengan beban komputasi yang lebih ringan dibandingkan model *baseline*.

3.2.6 Pelatihan model

Pelatihan model dilakukan untuk mengoptimalkan parameter model hasil modifikasi sehingga mampu mendeteksi kematangan tomat. Proses pelatihan dilaksanakan menggunakan *framework* PyTorch pada lingkungan Google Colaboratory dengan dukungan GPU T4. Hasil dari tahap pelatihan berupa berkas *weight model* (best.pt) yang berisi parameter terlatih. Berkas ini selanjutnya digunakan pada tahap evaluasi model untuk menilai performa berdasarkan metrik mAP50, *recall, precision, inference time*, dan beban memori.

3.2.7 Evaluasi Model

Model yang telah selesai melalui proses pelatihan akan menghasilkan sebuah berkas yang disebut *weight model*. Berkas ini memuat parameter-parameter terlatih yang merepresentasikan pengetahuan yang telah dipelajari oleh model selama

proses pelatihan berlangsung. *Weight model* inilah yang selanjutnya digunakan pada tahap evaluasi untuk mengukur kinerja model melalui berbagai *performance metrics* yaitu mAP, *recall, precision, inference time,* dan beban memori model (Padilla dkk., 2020).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$AP = \int_0^1 Precision\,(t)dt \quad (4)$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \quad (5)$$

$$Inference\ time\ (ms) = \frac{T_{total}}{N} \text{ x } 1000 \quad (6)$$

Persamaan (2) dan (3), True Positive (TP) merupakan kondisi ketika model berhasil mendeteksi dan mengklasifikasikan objek dengan benar. *False Positive* (FP) terjadi saat model mendeteksi objek yang sebenarnya tidak ada dalam citra, sedangkan *False Negative* (FN) menunjukkan kondisi ketika objek sebenarnya ada, tetapi tidak berhasil dideteksi oleh model (Karimi, n.d.). Selanjutnya, *Average Precision* (AP) pada persamaan (4) digunakan untuk mengukur tingkat akurasi dan kelengkapan deteksi terhadap satu kelas objek, dengan mempertimbangkan kombinasi antara nilai *precision* (ketepatan deteksi) dan *recall* (kelengkapan deteksi). Sedangkan *Mean Average Precision* (mAP) pada persamaan (5) menggambarkan rata-rata nilai AP dari seluruh kelas objek dan menjadi indikator utama dalam menilai kinerja keseluruhan model deteksi objek.

Adapun *Inference Time* (*ms*) pada persamaan (6) digunakan untuk mengukur rata-rata waktu yang dibutuhkan model dalam melakukan prediksi terhadap satu citra uji. Dimana Ttotal adalah total waktu *inference* untuk seluruh citra uji (dalam detik), N adalah jumlah citra yang diuji, lalu dikalikan 1000 untuk mengonversi satuan waktu dari detik ke mili detik. Nilai *Inference Time* yang lebih kecil menunjukkan model memiliki waktu prediksi yang lebih cepat, sehingga lebih efisien untuk implementasi *real-time object detection*. Metrik performansi pada beban memori digunakan untuk mengevaluasi efisiensi sumber daya komputasi yang dibutuhkan oleh model selama proses *training* maupun *inference*, khususnya pada sistem dengan keterbatasan memori seperti GPU atau perangkat *edge*. Metrik ini memberikan gambaran mengenai seberapa besar kapasitas memori yang digunakan model untuk menyimpan parameter, fitur antar-lapisan, serta hasil komputasi sementara selama proses deteksi objek berlangsung.