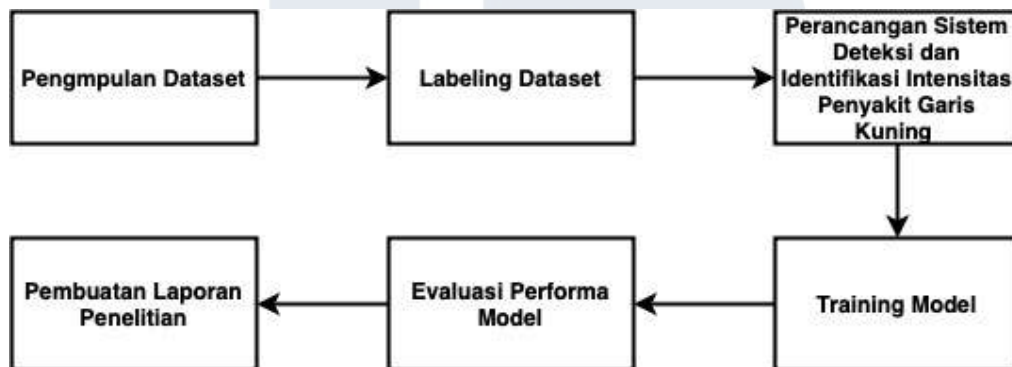


## BAB III

### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1 Metode Penelitian

Pada penelitian ini, terdapat beberapa tahap perancangan sistem yang dilakukan oleh penulis untuk mendapatkan hasil penelitian yang baik yaitu dengan studi literatur, pengambilan dataset, *labeling dataset*, perancangan sistem deteksi dan identifikasi intensitas penyakit garis kuning, kemudian melakukan evaluasi performa model, dan pembuatan laporan penelitian. Alur dari perancangan sistem yang dibuat dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Penelitian

#### 3.2 Perancangan Modul

##### 3.2.1 Perancangan Sistem Deteksi dan Identifikasi Intensitas Penyakit Garis Kuning

Dalam penelitian ini, digunakan model *Faster R-CNN* sebagai metode utama untuk mendeteksi dan mengidentifikasi area yang terdampak penyakit. *Faster R-CNN* dipilih karena kemampuannya dalam menghasilkan deteksi yang akurat dengan kecepatan tinggi dibandingkan model deteksi objek konvensional. Model ini akan menerima *input* berupa citra dari *drone* yang telah diberi label sebelumnya, kemudian melalui proses ekstraksi fitur dan klasifikasi, sistem akan mengidentifikasi serta menentukan tingkat keparahan penyakit berdasarkan empat kelas yang telah ditentukan. Tahap ini juga

mencakup perancangan arsitektur sistem, *Preprocessing data*, pelatihan model, serta evaluasi performa untuk memastikan sistem bekerja secara optimal dalam mengidentifikasi intensitas penyakit garis kuning pada kelapa sawit.

### 3.2.2 Pengumpulan Dataset

Pengumpulan data dilakukan dengan menggunakan *drone*, yaitu DJI Air 2S yang memiliki resolusi 5472 x 3078. *Drone* tersebut dikendalikan menggunakan aplikasi *DroneLink*, pada aplikasi tersebut penulis memanfaatkan fitur *mapping* untuk menjalankan *drone* secara otomatis sesuai dengan jalur yang dibuat. Semua data yang dikumpul adalah *aerial image* dari beberapa kebun yang merupakan bagian dari Koperasi Unit Desa (KUD) Kembang Sari.



Gambar 3.2 Alur Pengumpulan Dataset

Berdasarkan Gambar 3.2, langkah pertama yang dilakukan adalah membuat *mapping plan* atau jalur terbang *drone* pada aplikasi *DroneLink* berdasarkan peta kebun KUD Kembang Sari. Setelah itu *mapping plan* di verifikasi oleh pihak KUD Kembang Sari untuk memastikan *mapping plan* yang dibuat sudah sesuai dengan kebun milik KUD Kembang Sari. Selanjutnya adalah menghubungkan *drone* dengan aplikasi *DroneLink* yang sudah di *install* pada *smartphone* kemudian *controller* dari *drone* dihubungkan dengan *smartphone*. Setelah *drone* tersambung dengan aplikasi, maka *mapping mission* akan dijalankan, tugas dari user yang mengendalikan *drone* adalah hanya menagatasi *error* apabila terdapat *error*. Setelah misi selesai, data yang sudah didapatkan akan diverifikasi.

### 3.2.3 Labeling Data

Dalam penelitian ini, proses pelabelan data dilakukan menggunakan web aplikasi Roboflow untuk mengidentifikasi dan mengategorikan tingkat penyakit kuning pada kelapa sawit. Data yang diperoleh dari citra *drone* diberi label berdasarkan empat kelas, yaitu: tingkat Sehat, tingkat Ringan, tingkat Sedang, dan tingkat Berat. Pelabelan ini bertujuan untuk mempermudah proses pelatihan model dalam mengenali pola penyakit serta meningkatkan akurasi klasifikasi pada tahap analisis selanjutnya. Proses pelabelan data ini dibantu oleh ekspertis yang ahli dalam bidang penyakit pada kelapa sawit yaitu Dr. Supriyanto, SP, M.Sc., untuk memvalidasi label yang diberikan pada data.

Pelabelan data dilakukan dengan memberikan *bounding box* pada objek yaitu tanaman kelapa sawit, label yang diberikan disesuaikan dengan tingkatan intensitas sebaran penyakit garis kuning pada kelapa sawit. Pada proses anotasi, terdapat tiga *tool* yang digunakan yaitu:

- *Bounding Box Tool*, yaitu sebuah *tool* yang digunakan untuk memberikan *bounding box* pada sebuah objek setelah *bounding box* dibuat, selanjutnya memberikan kelas pada objek sesuai dengan tingkatan intensitas sebaran penyakit.
- *Drag Tool*, digunakan untuk menggeser gambar apabila gambar di *zoom in*, sehingga bisa memberikan label pada semua objek yang terdapat pada suatu gambar.
- *Delete Tool*, untuk menghapus label pada suatu objek, biasanya digunakan apabila salah memberikan label pada objek.

Contoh gambar dari data asli dan gambar yang sudah di anotasi dapat dilihat pada gambar 3.3 dan gambar 3.4



Gambar 3.3 Gambar sebelum dianotasi



Gambar 3.4 Gambar setelah dianotasi

Berdasarkan gambar 3.4. objek dengan *bounding box* berwarna hijau adalah objek pohon yang sehat, *bounding box* berwarna ungu adalah objek pohon dengan penyakit garis kuning tingkat intensitas ringan, *bounding box* berwarna merah adalah objek pohon dengan tingkat intensitas berat, dan *bounding box* berwarna orange adalah objek pohon dengan penyakit garis kuning tingkat intensitas berat.



Gambar 3.5 Tanaman kelapa sawit “Sehat”



Gambar 3.6 *Patch yellow* “Ringan”



Gambar 3.7 *Patch yellow* “Sedang”



Gambar 3.8 *Patch yellow* “Berat”

Berdasarkan gambar-gambar di atas, gambar 3.5 adalah contoh gambar dari tanaman kelapa sawit yang sehat, gambar 3.6 adalah gambar tanaman kelapa sawit yang mengalami intensitas serangan penyakit *patch yellow* ringan, gambar 3.7 adalah tanaman kelapa sawit yang mengalami intensitas serangan penyakit *patch yellow* sedang, dan gambar 3.8 adalah gambar tanaman kelapa sawit yang mengalami intensitas serangan penyakit *patch yellow* berat.

#### 3.2.4 *Preprocessing Data*

*Preprocessing* data yang dilakukan pada penelitian ini adalah melakukan *resize* gambar menjadi  $640 \times 640$  piksel. Hal tersebut dilakukan karena ukuran gambar asli pada dataset adalah  $5472 \times 3648$  piksel, yang akan membuat proses pelatihan menjadi sangat berat, memakan banyak *resource*, dan membutuhkan waktu yang lama. Proses *resize* ini dilakukan tanpa memotong gambar (*cropping*), melainkan dengan melakukan penskalaan ulang secara proporsional dan menambahkan *padding* agar gambar tetap dalam rasio yang sesuai dengan *input* model. Dengan cara ini, seluruh informasi dalam gambar tetap terjaga tanpa kehilangan bagian penting.

Selain itu, *Auto-Orient* juga digunakan pada tahap *preprocessing* untuk memastikan gambar diputar sesuai dengan orientasi yang benar. *Undersampling* terhadap kelas mayoritas juga dilakukan guna

mengurangi jumlah data pada kelas yang dominan agar seimbang dengan kelas lainnya. *Preprocessing* terakhir adalah augmentasi pada kelas minoritas, yang bertujuan untuk meningkatkan jumlah data pada kelas tersebut sehingga distribusi kelas menjadi lebih seimbang.

### 3.2.5 Training Model

Dalam penelitian ini, pelatihan model dilakukan menggunakan Google Colab Pro, yang menyediakan akses ke GPU dengan performa tinggi, memungkinkan proses *training* berjalan lebih cepat dan efisien. Dataset yang telah dilabeli sebelumnya dimasukkan ke dalam pipeline pelatihan, yang mencakup beberapa tahap utama, yaitu *Preprocessing* data, *undersampling*, augmentasi, *Split dataset* yang digunakan adalah 80% *training*, 10% *validation* dan 10% testing. Dataset yang sudah dibagi menjadi *train*, *val*, dan *test* kemudian dimaukan ke dalam *data loader* untuk digunakan dalam proses *training*, *validation*, dan testing nantinya.

Google Colab Pro dipilih karena keunggulannya dalam menyediakan sumber daya komputasi yang lebih besar dibandingkan versi gratisnya, sehingga memungkinkan pelatihan model berjalan lebih optimal. Proses *training* model dilakukan dengan PyTorch dan Torchvision, tepatnya *torch* dan *torch.models.detection*. Detail *hyperparameter* yang digunakan bisa dilihat pada tabel 3.1.

Tabel 3.1 *Hyperparameter training model*

| <i>Hyperparameter</i> | <i>Value</i> |
|-----------------------|--------------|
| <i>Optimizer</i>      | SGD          |
| <i>Learning Rate</i>  | 0.005        |
| <i>Batch Size</i>     | 16           |
| <i>Epoch</i>          | 20           |
| <i>Wheight</i>        | None         |
| <i>Treshhold</i>      | 0.2          |

Dalam proses pelatihan model *Faster R-CNN*, pemilihan *hyperparameter* dilakukan secara hati-hati untuk mencapai performa deteksi yang optimal. *Optimizer* yang digunakan adalah *Stochastic Gradient Descent* (SGD) karena terbukti efektif dan stabil dalam melatih model deteksi objek, khususnya pada dataset berskala kecil hingga menengah. *Learning rate* ditetapkan sebesar 0.005, yang merupakan nilai moderat agar pembelajaran model tidak terlalu lambat namun tetap menghindari divergensi. *Batch size* sebesar 16 dipilih sebagai kompromi antara efisiensi komputasi dan kestabilan gradien selama pelatihan. Jumlah *epoch* sebanyak 20 digunakan untuk memberikan waktu yang cukup bagi model dalam belajar dari data, namun juga mencegah terjadinya *overfitting*. Untuk parameter *weight*, digunakan None, yang artinya model dilatih dari awal tanpa bobot pralatih, yang memungkinkan model lebih bebas dalam menyesuaikan diri dengan dataset khusus yang digunakan. Terakhir, *threshold* deteksi ditetapkan pada 0.2 untuk memastikan bahwa model mampu menampilkan deteksi dengan sensitivitas tinggi, terutama dalam kasus penyakit ringan yang mungkin memiliki fitur visual yang tidak terlalu jelas. Fungsi callback *early\_stopping*, metrik *val\_loss* dijadikan acuan dalam *monitoring* dengan *patience* 5. Artinya, jika dalam 5 epoch terakhir nilai dari *val\_loss* tidak meningkat, maka proses *training* akan berhenti secara otomatis.

Pada proses *training*, informasi *train loss*, *validation loss*, mAP (*Mean Average Precision*), *precision*, *Recall*, dan *F1 Score* ditampilkan pada akhir setelah proses *training* berhenti. Hasilnya kemudian disimpan dengan format .pth, dan semua informasi yang ditampilkan disimpan ke dalam sebuah variabel dan hasilnya akan divisualisasikan pada tahap evaluasi model.

### 3.2.6 Evaluasi Performa Model

Dalam proses evaluasi hasil *training* keempat model, digunakan beberapa pengujian yang meliputi:

#### 3.2.6.1 *Learning curve*

Grafik *Learning curve* yang diperoleh dari hasil *training* sebuah model menunjukkan *loss* dari setiap *epoch*. Sumbu x mengindikasikan *epoch*, sementara sumbu y menunjukkan nilai *loss* yang diperoleh. Hasil *training* dan hasil validasi divisualisasikan melalui grafik ini, dan dari hasil tersebut dapat digunakan untuk menentukan apakah model berstatus *overfitting*, yang terjadi apabila *train loss* nilainya lebih kecil, tetapi apabila nilai *val loss* lebih besar statusnya akan menjadi *underfitting* yaitu kebalikan dari *overfitting*, atau *goodfit* di mana *train loss* dan *val loss* nilainya kecil, yang artinya model yang di *training* cocok untuk dataset yang digunakan.

#### 3.2.6.2 *Evaluation Metrics*

Dalam evaluasi performa model terhadap *test dataset*, terdapat 5 jenis *evaluation metrics* yang digunakan yaitu:

- **F1 Score** menggunakan hasil *Precision* dan *Recall* serta memperhitungkan *harmonic mean* dari keduanya dengan nilai berkisar antara 0 sampai 1. Semakin mendekati nilai 1, maka rasio *Precision* dan *Recall* dari model tersebut semakin bagus, artinya model memiliki ketelitian yang tinggi dalam proses pelabelan, karena kemungkinan terjadinya *false positive* dan *false negative* sangat rendah. Dalam konteks deteksi tingkat intensitas penyakit garis kuning pada tanaman kelapa sawit, nilai F1 Score yang tinggi menunjukkan bahwa model mampu mengidentifikasi keberadaan serta tingkat keparahan penyakit dengan akurat dan konsisten. Hal ini sangat penting untuk mendukung pengambilan keputusan di lapangan, seperti tindakan perawatan atau pemupukan yang sesuai dengan kondisi nyata tanaman. Dengan demikian, F1 Score menjadi

indikator utama dalam mengevaluasi sejauh mana model dapat membedakan antara tanaman sehat dan yang terinfeksi pada berbagai tingkat intensitas penyakit secara seimbang, tanpa bias terhadap satu kategori tertentu.

$$F1\ Score = \frac{2 \times precision \times recall}{precision + recall}$$

- *Precision* merupakan rasio antara jumlah prediksi benar untuk kelas positif (*True Positive*) dibandingkan dengan seluruh prediksi positif yang dihasilkan. Nilai *Precision* yang tinggi menunjukkan bahwa model memiliki tingkat kesalahan prediksi positif (*False Positive*) yang rendah, atau dengan kata lain, model jarang menganggap piksel yang bukan bagian dari kelas tertentu—dalam hal ini tanaman kelapa sawit—sebagai termasuk dalam kelas tersebut.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- *Recall*, adalah perbandingan antara jumlah prediksi benar untuk kelas positif (*True Positive*) dengan total jumlah data yang memang termasuk dalam kelas positif tersebut. *Recall* yang tinggi menunjukkan bahwa model memiliki tingkat kesalahan *False Negative* yang rendah, artinya model jarang melewatkan atau salah mengabaikan piksel yang seharusnya termasuk dalam suatu kelas—dalam konteks ini, model berhasil mengenali hampir semua piksel yang benar-benar merupakan bagian dari kelas tertentu.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- IoU (*Intersection over Union*) atau yang juga dikenal sebagai *Jaccard Index*, adalah metrik yang digunakan untuk mengukur seberapa besar tumpang tindih antara area prediksi

model dan area *ground truth* pada sebuah gambar. Nilai IoU berkisar antara 0 hingga 1, di mana semakin mendekati angka 1, berarti hasil prediksi model semakin akurat dan mendekati posisi serta ukuran objek yang sebenarnya. Jika nilainya mencapai 1, maka prediksi model sepenuhnya sesuai dengan *ground truth*. Dengan kata lain, semakin tinggi nilai IoU, semakin dapat diandalkan hasil deteksi objeknya.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

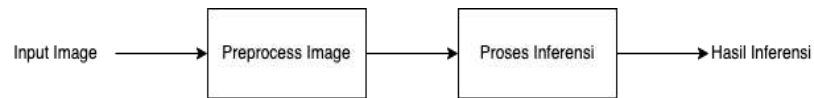
- mAP (*mean Average Precision*) adalah metrik evaluasi dalam deteksi objek yang menunjukkan seberapa baik model mengenali dan melokalisasi objek dari semua kelas. Nilai mAP merupakan rata-rata dari nilai *Average Precision* (AP) tiap kelas, yang menggabungkan aspek *Precision* dan *Recall* pada berbagai ambang batas. Semakin tinggi nilai mAP (mendekati 1), maka semakin akurat dan andal hasil deteksi objek yang dihasilkan oleh model.

$$\text{mAP} = \frac{1}{n} \sum_{i=1}^n \text{AP}_i$$

### 3.2.7 Pengujian Inferensi pada Sampel Gambar

Pada tahap ini, model dengan *validation* loss terbaik hasil *training* sudah tersimpan dan kembali di-*load* dan diubah ke mode *validation*. Kemudian akan dilakukan 2 jenis inferensi, yaitu inferensi terhadap gambar yang ada pada *test dataset*, dan pada gambar hasil tangkapan *drone*. Hasil inferensi terhadap *test dataset* akan ditampilkan secara acak sebanyak 5 gambar, dan dibandingkan dengan *ground truth* yang ada.

Sementara itu hasil inferensi terhadap foto tangkapan *drone* akan ditampilkan berupa gambar asli. Dalam melakukan inferensi terhadap gambar tangkapan *drone* yang beresolusi tinggi, dilakukan beberapa tahap seperti yang terlihat pada gambar 3.5



Gambar 3.5 Proses inferensi gambar asli

Gambar tersebut menunjukkan dua tahapan utama dalam proses deteksi objek. Tahap pertama adalah *Preprocess Image*, di mana citra awal dipersiapkan sebelum dimasukkan ke dalam model. Pada tahap ini, gambar diubah ukurannya menjadi 640 x 640 piksel agar sesuai dengan format *input* yang dibutuhkan oleh model deteksi objek. Proses ini penting untuk memastikan konsistensi ukuran *input* dan membantu model melakukan prediksi secara optimal. Setelah itu, gambar yang telah diproses akan masuk ke tahap kedua yaitu Proses Inferensi, di mana model melakukan analisis terhadap citra untuk mendeteksi dan mengklasifikasikan objek yang ada di dalamnya berdasarkan *bounding box* dan label yang dihasilkan.

