

**DETEKSI PENYAKIT TANAMAN JERUK BERBASIS CITRA DAUN
DENGAN METODE YOU ONLY LOOK ONCE VERSI 8 (YOLOV8)**

SKRIPSI

SAMUEL PARLINDUNGAN MALAU

191402092



**PROGRAM STUDI S-1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**DETEKSI PENYAKIT TANAMAN JERUK BERBASIS CITRA DAUN
DENGAN METODE YOU ONLY LOOK ONCE VERSI 8 (YOLOV8)**

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Teknologi Informasi**

SAMUEL PARLINDUNGAN MALAU

191402092



**PROGRAM STUDI S-1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PERSETUJUAN

Judul : Deteksi Penyakit Tanaman Jeruk Berbasis Citra Daun dengan Metode You Only Look Once Versi 8 (YOLOV8)

Kategori : Skripsi

Nama Mahasiswa : Samuel Parlindungan Malau

Nomor Induk Mahasiswa : 191402092

Program Studi : Sarjana (S-1) Teknologi Informasi

Fakultas : Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara

Medan, 11 Juli 2024

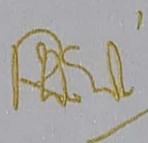
Komisi Pembimbing:

Pembimbing 2,



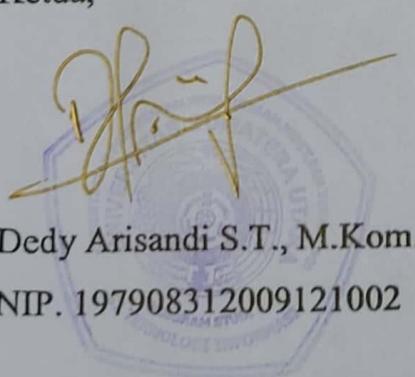
Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

Pembimbing 1,



Dr. Erna Budhiarti Nababan M.I.T.
NIP. 196210262017042001

Diketahui/disetujui oleh
Program Studi S-1 Teknologi Informasi
Ketua,



Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

PERNYATAAN

DETEKSI PENYAKIT TANAMAN JERUK BERBASIS CITRA DAUN DENGAN
METODE YOU ONLY LOOK ONCE VERSI 8 (YOLOV8)

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, ... Juni 2024

Samuel Parlindungan Malau

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Kuasa atas rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi sebagai salah satu syarat kelulusan program Sarjana dan memperoleh gelar Sarjana Komputer pada program studi S1 Teknologi Informasi pada Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara Penulis skripsi juga mengucapkan terima kasih kepada semua pihak atas dukungan, bantuan, serta doa yang diberikan kepada penulis dalam menyelesaikan skripsi ini. Sehubungan dengan itu, penulis ingin mengucapkan terima kasih kepada :

1. Tuhan Yesus Kristus yang selalu memberikan rahmat, kekuatan, penguatan, serta berkat yang melimpah sehingga penulis dapat menyelesaikan skripsi ini dengan baik
2. Keluarga penulis, Bapak Lorentius Malau dan Ibu Ratna Meilyna Sitanggang, yang selalu mendukung serta mendoakan dan memberikan nasehat, dorongan, dan semangat dalam proses perkuliahan hingga penyelesaian skripsi, begitu juga dengan saudara-saudari penulis yang terkasih dan tersayang, Imelda Loisa Yosephine Malau, Novalina Romaitona Malau, Michael Surung Faberius Malau, Helena Elma Rasita Malau, dan juga abang ipar saya, Julius Duha, yang senantiasa memberikan doa dan dukungan.
3. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
4. Ibu Dr. Maya Silvi Lydia, M.Sc selaku Dekan Fasilkom-TI Universitas Sumatera Utara.
5. Bapak Dedy Arisandi, S.Kom., M.Kom. selaku Ketua program studi Teknologi Informasi Universitas Sumatera Utara
6. Ibu Dr. Erna Budhiarti Nababan, M.IT selaku Dosen Pembimbing I dan Bapak Dedy Arisandi S.T., M.Kom. selaku Dosen Pembimbing II yang telah bersedia membimbing, memberikan saran dan kritik yang membangun.
7. Seluruh Dosen, Staff dan Pegawai Program Studi S1 Teknologi Informasi yang telah memberikan manfaat bagi penulis serta membantu proses perkuliahan penulis
8. Teman-teman penulis, Yesaya Silalahi, Jhuan Sitorus, Christoper Manurung, dan Monang Limbong, yang sering membantu dan mendorong penulis dalam penggerjaan penelitian ini.
9. Teman-teman kuliah penulis, Tomy Tambunan, Brian Tarihoran, Vincent Sirait, Josua Pandiangan, Sebastian Sitorus, Timothy Ginting, Gemilang Sibarani, Anri Marpaung, Mahdan, Zikri Ihsan, Anggi Pardede, dan Gideon Ginting, yang selalu menemani penulis selama perkuliahan dan menikmati hari-hari bersama dalam menjalani hidup perkuliahan di Medan terutama di Warung Kopi Andimed, tempat menciptakan memori yang indah ketika berkumpul dan bersenang-senang bersama.

10. Teman-teman SMA penulis, Efrin Pinem, Muhammad Fathurahman, Ginomgom Pane, Abdul Capah, Marco Sembiring, Rizky Tumangger, Nicolas Sinaga, dan Ade Sihombing, yang menemani penulis dalam menyegarkan pikiran ketika sedang banyak pikiran saat berada di kampung halaman penulis.
11. Teman-teman Angkatan 2019 serta senior dan junior yang juga ikut masuk ke dalam kehidupan penulis dalam menyelesaikan perkuliahan.
12. Kepada seluruh penyanyi yang ada di dalam *playlist* penulis yang membuat penulis selalu bersemangat mengerjakan segala tugas perkuliahan hingga menyelesaikan tugas akhir ini.
13. Dan kepada Kakak-Abang dan Adik-Adik lainnya yang tidak dapat saya sebutkan satu persatu yang telah memberikan dukungan serta saran dalam masa perkuliahan serta proses penyelesaian tugas akhir ini.

Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan, oleh karena itu penulis mengharapkan kritik dan saran yang membangun untuk penyempurnaan skripsi ini.

ABSTRAK

Tanaman jeruk adalah jenis tanaman berbuah yang banyak ditemui di dunia dan tersebar di seluruh Asia, Mediterania, Afrika, hingga Amerika Selatan dan Utara. Buah jeruk memiliki khasiat dan nilai gizi yang tinggi, sehingga sangat baik dan populer sebagai bahan makanan maupun minuman. Sebagai buah yang memiliki banyak khasiat, jeruk menjadi buah yang memiliki nilai ekonomi yang tinggi. Di satu sisi, tanaman jeruk juga rentan terhadap serangan penyakit dan hama yang membuat nilainya berkurang baik terutama dari segi harga. Semakin banyak peminat buah jeruk dapat membuat peningkatan akan jumlah petani jeruk yang tertarik dalam memulai perkebunan jeruk. Di antara banyaknya petani jeruk tersebut, tentu tidak sedikit petani yang kurang jeli dalam mendeteksi gejala penyakit jeruk melalui pengamatan secara langsung. Oleh karena itu diperlukan sebuah sistem yang dapat mendeteksi penyakit jeruk secara langsung sehingga petani jeruk dapat mengidentifikasi penyakit tersebut di tempat. Terdapat tiga kategori penyakit yang diklasifikasikan menggunakan citra daun pada penelitian ini, yaitu penyakit *canker* daun, penyakit lumut daun, dan penyakit karat daun. Penelitian ini menggunakan metode *You Only Look Once* versi 8 (YOLOv8) yang diintegrasikan pada perangkat android sehingga dapat melakukan pendekripsi menggunakan kamera secara *real time*. Jumlah data yang digunakan pada penelitian ini sebanyak 900 data citra yang digunakan untuk melatih model, memvalidasi model yang dihasilkan, dan menguji sistem. Berdasarkan pengujian yang telah dilakukan, sistem yang telah dibuat dapat mendeteksi penyakit daun jeruk secara *real time*. Penelitian yang menggunakan YOLOv8 ini mendapatkan akurasi yang jauh dari kata sempurna, yaitu 96,67%.

Kata Kunci : Penyakit Daun Jeruk, Deteksi *Real Time*, YOLOv8, Citra Digital.

**CITRUS PLANT DISEASES DETECTION BASED ON LEAF IMAGES USING
YOU ONLY LOOK ONCE VERSION 8 (YOLOV8) METHOD**

ABSTRACT

Citrus plants are a type plant that is often found around the world and is spread throughout Asia, the Mediterranean, Africa, to South and North America. Citrus fruit has high nutritional value which makes it very good and popular as a food and drink . As a fruit that has many merits, citrus become a kind of fruit that has high economic value. On the one hand, citrus plants are also susceptible to disease and pest attacks which reduce their value, especially in terms of market value. Because of the popularity of the citrus fruit, a plenty of people become interested in started owning an orange orchards. Among that many citrus farmers, of course there are quite a few farmers who are less observant in detecting symptoms of citrus disease through direct observation. Therefore, this needs a system that can detect citrus diseases directly so that orange farmers can identify the disease on the spot. There are three categories of disease that were classified using leaf images in this study, namely canker disease, citrus greasy spot disease, and citrus moss disease. This research uses the You Only Look Once version 8 (YOLOv8) method which is integrated on an Android device so that it can detect using the camera in real time. The amount of data used in this research was 900 image data which was used to train the model, validate the resulting model, and test the system. Based on the tests that have been carried out, the system that has been created can detect citrus leaf diseases in real time. Using YOLOv8 this research obtained an accuracy of 96.67% which was far from perfect.

Keyword : Citrus Disease, Real Time Object Detection, YOLOv8, Digital Image Processing.

DAFTAR ISI

PERNYATAAN	iv
UCAPAN TERIMA KASIH	v
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	ix
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Tujuan Penelitian	3
1.4. Batasan Masalah	3
1.5. Manfaat Penelitian	4
1.6. Metodologi Penelitian	4
1.7 Sistematika Penelitian	5
BAB 2 LANDASAN TEORI	6
2.1. Tanaman Jeruk	6
2.2. Penyakit Tanaman Jeruk	6
2.2.1 Citrus Moss (lumut daun jeruk)	7
2.2.2 Citrus Canker (kanker jeruk)	8
2.2.3 Citrus Greasy Spot (karat daun)	9
2.3. Citra Digital	9
2.4. <i>Computer Vision</i>	10
2.5. <i>Object Detection</i>	10
2.6. <i>Convolutional Neural Network (CNN)</i>	10
2.7. <i>You Only Look Once (YOLO)</i>	12
2.7.1 YOLOv1	12
2.7.2 YOLOv2	12
2.7.3 YOLOv3	14
2.7.4 Backbone, Neck, dan Head	14
2.7.5 YOLOv4	15
2.7.6 YOLOv5	16

2.7.7 YOLOv6	17
2.7.8 YOLOv7	17
2.7.9 YOLOv8	18
2.8 <i>Confusion Matrix</i>	25
2.8.1 Precision	26
2.8.2 Recall	26
2.8.3 Mean Average Precision (mAP):	26
2.8.4 F1-score	27
2.8.5 Accuracy	27
2.9 Roboflow	27
2.10 Google Colab	28
2.11 Ultralytics Hub	29
2.12 Tensorflow Lite	29
2.13 Android Studio	30
2.14. Penelitian Terdahulu	31
2.15 Perbedaan Penelitian Terdahulu	37
BAB 3 ANALISIS DAN PERANCANGAN SISTEM	39
3.1 Data yang digunakan	39
3.2 Analisis Sistem	39
3.2.1 Image Acquisition	41
3.2.2 Pre-processing	41
3.2.3 Image Classification	48
3.2.4 Proses Training	55
3.2.5 Learned Model dan Proses Validasi	56
3.2.6 Tensorflow Lite Model	57
3.2.7 Proses Pengujian	57
3.3 Perancangan Antarmuka Sistem	57
3.3.1 Rancangan tampilan beranda aplikasi	58
3.3.2 Rancangan tampilan menu detect	59
3.3.3 Rancangan tampilan menu about	60
BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM	61
4.1 Implementasi sistem	61
4.1.1 Perangkat keras dan perangkat lunak	61

4.1.2 Implementasi data	62
4.1.3 Implementasi pre-processing data	63
4.1.4 Implementasi desain interface	64
4.2 Prosedur Operasional	67
4.3 Pelatihan Sistem	67
4.4 Pengujian Sistem	73
BAB 5 KESIMPULAN DAN SARAN	85
5.1 Kesimpulan	85
5.2 Saran	85
DAFTAR PUSTAKA	86

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu	34
Tabel 3.1 Jenis-jenis model yang ada pada YOLOv8	54
Tabel 4.1 Hasil training	71
Tabel 4.2 Hasil pelatihan model	72
Tabel 4.3 Contoh hasil pengujian aplikasi	74
Tabel 4.4 <i>Confusion matrix</i>	80
Tabel 4.5 Hasil nilai <i>True Positive</i> dan <i>False Negative</i>	80
Tabel 4.6 Hasil nilai <i>precision</i> , <i>recall</i> , dan <i>F1-score</i>	81

DAFTAR GAMBAR

Gambar 2.1 Tanaman jeruk yang rusak akibat terkena penyakit cancer	7
Gambar 2.2 Daun jeruk sehat	7
Gambar 2.3 Daun jeruk yang terkena penyakit lumut daun	8
Gambar 2.4 Daun jeruk yang terkena penyakit cancer	8
Gambar 2.5 Daun jeruk yang terkena penyakit karat daun	9
Gambar 2.6 Contoh penerapan object detection	10
Gambar 2.7 Arsitektur umum Convolutional Neural Networks	11
Gambar 2.8 Arsitektur umum pendekripsi objek	15
Gambar 2.9 Arsitektur YOLOv4 untuk Object Detection	16
Gambar 2.10 Arsitektur YOLOv8	19
Gambar 2.11 Confusion Matrix untuk prediksi dua kelas	25
Gambar 2.12 Fitur augmentasi dan preprocessing Roboflow	28
Gambar 2.13 Arsitektur TensorFlow Lite	30
Gambar 3.1 Contoh citra penyakit jeruk cancer, karat daun, dan lumut daun	39
Gambar 3.2 Arsitektur Umum	40
Gambar 3.3 Contoh pelabelan data pada citra karat daun menggunakan LabelImg	42
Gambar 3.4 File predefined_classes.txt	42
Gambar 3.5 Isi file karat94.xml sebagai hasil dari pelabelan	43
Gambar 3.6 Contoh nilai-nilai yang digunakan pada pelabelan citra karat94	43
Gambar 3.7 File citra dan file label hasil pelabelan menggunakan LabelImg	44
Gambar 3.8 Hasil impor data pada platform Roboflow	44
Gambar 3.9 Gambaran proses resize fit (black edges) in dan fit (white edges) in	46
Gambar 3.10 Resize citra pada platform Roboflow	46
Gambar 3.11 Contoh proses augmentasi rotate dan flip pada salah satu citra	47
Gambar 3.12 Augmentasi flip dan rotation yang dilakukan pada platform Roboflow	47
Gambar 3.13 Pseudocode python untuk augmentasi data	48
Gambar 3.14 Contoh nilai RGB pada gambar	49
Gambar 3.15 Perhitungan feature extraction pada layar red pada gambar	49
Gambar 3.16 Ekstraksi fitur setiap dimensi warna pada <i>convolution layer</i>	51
Gambar 3.17 Proses pada blok konvolusi	53
Gambar 3.18 Piramida fitur yang dihasilkan SPPF	54

Gambar 3.19 Proses YOLOv8 hingga pemberian bounding box dan class	55
Gambar 3.20 Pseudocode proses training	56
Gambar 3.21 Diagram Aktivitas	58
Gambar 3.22 Rancangan tampilan beranda aplikasi	59
Gambar 3.23 Rancangan tampilan menu detect	59
Gambar 3.24 Rancangan tampilan menu about	60
Gambar 4.1 Beberapa contoh data citra penyakit cancer jeruk	62
Gambar 4.2 Beberapa contoh data citra penyakit karat daun jeruk	62
Gambar 4.3 Beberapa contoh data citra penyakit lumut daun jeruk	63
Gambar 4.4 Penggunaan auto-orient dan resize terhadap data	63
Gambar 4.5 Hasil augmentasi data	64
Gambar 4.6 Tampilan halaman beranda	65
Gambar 4.7 Tampilan halaman detect	66
Gambar 4.8 Tampilan halaman about	66
Gambar 4.9 Pemilihan data untuk dilatih pada platform Ultralytics Hub	68
Gambar 4.10 Konfigurasi model yang digunakan	69
Gambar 4.11 Ekspor ke google colab menggunakan API key colab	70
Gambar 4.12 Tampilan instalasi setup dan koneksi menggunakan API key	70
Gambar 4.13 Hasil validasi model	71
Gambar 4.14 Konversi file pytorch menjadi tflite	73
Gambar 4.15 Pengujian tflite pada data uji	73
Gambar 4.16 Diagram hasil penilaian aplikasi	84

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Tanaman Jeruk atau *Citrus* merupakan jenis tanaman buah-buahan yang dikenal sebagai tumbuhan asli Asia Tenggara dan banyak ditanam di daerah tropis dan subtropis di seluruh dunia, salah satunya adalah Indonesia. Jeruk merupakan sumber vitamin C yang sangat baik dan populer baik sebagai bahan makanan maupun minuman. Sebagai buah yang memiliki banyak khasiat dan menjadikannya buah yang banyak dikonsumsi oleh masyarakat, jeruk menjadi buah yang memiliki nilai ekonomi yang tinggi. Di satu sisi, tanaman jeruk juga rentan terhadap serangan penyakit dan hama yang membuat nilainya berkurang terutama dari segi harga.

Pendeteksian objek (*object detection*) dalam *machine learning* adalah teknologi komputer yang berhubungan dengan *computer vision* dan *image processing* yang digunakan untuk mendeteksi suatu objek citra digital yang dapat berupa struktur, bentuk, ukuran, maupun warna. Dalam hal ini pendeteksian objek sudah banyak dimanfaatkan di berbagai sektor, salah satunya adalah sektor pertanian. Pemanfaatan tersebut dapat digunakan dalam hal pemetaan lahan, pengklasifikasian kesehatan tanaman ataupun mutu buah, dan juga dalam pengidentifikasi atau pendeteksian penyakit tanaman.

Pada masa ini, meningkatnya jumlah perkebunan jeruk berbanding lurus dengan jumlah petani jeruk. Di antara banyaknya petani jeruk tersebut, tentu tidak sedikit petani yang kurang jeli dalam mendeteksi gejala penyakit jeruk dengan pengamatan secara langsung. Maka dari itu perlunya dibangun sistem dikarenakan diagnosis penyakit berdasarkan pengamatan optik pada daun tanaman jeruk yang dilakukan biasanya mengalami kekeliruan atau kurang tepat (Lestari, et al., 2019) .

Sehingga dengan adanya sistem dan dengan penanganan yang tepat pada penyakit yang menyerang tanaman jeruk, dapat mengatasi berkurangnya hasil produksi.

Penelitian sebelumnya terkait pengidentifikasi penyakit tanaman jeruk berdasarkan citra daun adalah penelitian yang dilakukan (Ariesdianto , et al., 2021) dalam mengidentifikasi penyakit Canker dan Ulat Peliang menggunakan *K-Nearest Neighbor*. Penelitian ini berhasil mengidentifikasi penyakit dengan akurasi 70% dengan variasi nilai $K = 21$. Berikutnya adalah penelitian yang dilakukan oleh (Lestari, et al., 2019) yang bertujuan untuk mengidentifikasi penyakit kanker daun, CVPD (*Citrus Vein Phloem Degeneration*) daun, dan ulat peliang daun menggunakan metode *Multicategory Support Vector Machines* (M-SVM). Pengujian dilakukan sebanyak 5 kali dengan kombinasi data latih dan data uji yang berbeda-beda. Hasil pengujian dari sistem didapatkan akurasi tertinggi sebesar 86,67%. Lalu penelitian berikutnya adalah penelitian yang dilakukan oleh (Febrinanto, et al., 2019), yaitu penelitian yang bertujuan untuk mengidentifikasi penyakit CVPD, *Downy Mildew*, and Cendawan Jelaga dengan menggunakan implementasi algoritma *K-Means*. Persentasi akurasi yang didapatkan dari studi ini adalah 90.83% .

Penelitian berikutnya adalah penelitian yang menggunakan metode *You Only Look Once* (YOLO) pada tanaman jeruk yang dilakukan oleh (Qiu, et al., 2022). Penelitian tersebut mempelajari tentang pengidentifikasi gejala penyakit *Huanglongbing* (HLB) terkhusus pada daun jeruk. Mereka menggunakan YOLOv5l dan membuat tiga jenis model berbeda, yaitu Yolov5l-HLB1, Yolov5l-HLB2, and Yolov5l-HLB3. Dari ketiga model tersebut, Yolov5l-HLB2 mendapat akurasi micro F1-score tertinggi yaitu 85,19% yang dimana dapat mengenali lima gejala penyakit HLB. Adapun penelitian yang dilakukan oleh (Gallo, et al., 2023) dalam mendekripsi rumput liar pada area penanaman tanaman gula. Penggunaan YOLOv7 pada dataset berjumlah 3373 gambar Hasil YOLOv7 yang diperoleh pada kumpulan data pertama mengungguli varian YOLO lain yang sudah dites dalam hal produksi metrik nilai masing-masing sebesar 56,6%, 62,1%, dan 61,3% untuk mAP@0.5 scores, recall, dan precision. Selain itu, model YOLOv7 yang diterapkan pada kumpulan data kedua melampaui hasil yang telah didapat sebelumnya dengan menggunakan YOLOv5 dimana akurasi nilai mAP@0.5 meningkat dari 51% menjadi 61% untuk mAP total, 67.5% menjadi 74.1% untuk mAP rumput liar, dan 34.6% untuk mAP tanaman gula.

Berdasarkan latar belakang yang telah dipaparkan, penulis bermaksud untuk melakukan pendekatan menggunakan metode *You Only Look Once* (YOLO) dengan harapan mendapatkan hasil yang lebih memuaskan dibandingkan penelitian terdahulu. Oleh karena itu, penulis mengajukan penelitian tugas akhir dengan judul “Deteksi Penyakit Tanaman Jeruk Berbasis Citra Daun dengan Metode You Only Look Once Versi 8 (YOLOv8)”

1.2. Rumusan Masalah

Diagnosis penyakit daun tanaman jeruk hanya berdasarkan pengamatan manual dapat mengalami kekeliruan atau kurang akurat, maka dari itu diperlukan satu pendekatan yang memadai dalam pendekatan penyakit sehingga dapat dilakukan penanganan yang tepat agar dapat mengatasi berkurangnya hasil produksi buah jeruk akibat serangan penyakit.

1.3. Tujuan Penelitian

Penelitian ini bertujuan untuk mendeteksi penyakit tanaman jeruk melalui citra daun secara *Real Time* dengan mengimplementasikan metode *You Only Look Once* (YOLO) pada aplikasi yang dibuat.

1.4. Batasan Masalah

Sesuai rumusan masalah yang telah dibuat, agar penelitian lebih terarah, maka dapat diberikan batasan masalah berupa:

1. Penyakit yang diidentifikasi hanya dikategorikan menjadi tiga jenis, yaitu Lumut Daun, Karat Daun dan *Canker*
2. Data yang diperoleh berasal dari perkebunan jeruk yang berada di daerah Sidikalang dan daerah Sumbul, Kabupaten Dairi, Sumatera Utara
3. Ekstensi data citra yang digunakan untuk kepentingan pelatihan sistem adalah .jpg
4. Data citra daun yang digunakan untuk pelatihan diperoleh dengan menggunakan kamera *smartphone* dan diambil langsung di kebun jeruk yang didampingi oleh ahli untuk mengenali penyakit yang dimaksud
5. Citra daun yang digunakan untuk pelatihan didapatkan dengan jarak pengambilan gambar 10-30 cm dari objek

6. Gambar daun jeruk yang digunakan dibatasi hanya menggunakan gambar daun jeruk manis, bukan jeruk purut maupun jeruk nipis.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat bagi petani jeruk untuk mendiagnosa ataupun mengenali penyakit jeruk sehingga dapat melakukan penanganan yang sesuai. Dengan dilakukannya penanganan tersebut, diharapkan dapat mencegah penurunan angka produksi jeruk.

1.6. Metodologi Penelitian

1) Studi Literatur

Hal yang pertama dilakukan dalam penelitian ini adalah mengumpulkan berbagai referensi yang berhubungan dengan penelitian ini dari berbagai sumber. Referensi tersebut dapat berupa buku, laporan penelitian lain, maupun forum penelitian. Dalam hal ini, penulis memilih referensi mengenai YOLO, tanaman jeruk, pendektsian objek, dan aplikasi android *real-time*.

2) Analisis Permasalahan

Tahap ini dilakukan agar penulis dapat memahami tentang penyakit tanaman jeruk yang diteliti, memahami cara menerapkan YOLO pada pendektsian penyakit tanaman, dan pengimplementasiannya pada perangkat *mobile*.

3) Perancangan

Sistem Setelah masalah dianalisis, maka dapat dilakukan perancangan sistem seperti pembuatan arsitektur umum model, pengumpulan data, dan perencanaan desain antar muka.

4) Implementasi

Tahap berikutnya adalah mengimplementasi rancangan yang telah dibuat pada tahap sebelumnya. Dimulai dari pelatihan model hingga pembuatan aplikasi *mobile*.

5) Pengujian

Aplikasi dengan model yang telah dibuat dapat diuji coba pada objek yang dikehendaki. Dalam hal ini, pengujian dilakukan pada daun tanaman jeruk.

6) Dokumentasi dan Penyusunan Laporan

Tahap ini merupakan tahap terakhir dimana dilakukan pengumpulan, penggabungan informasi, dan penulisan laporan yang relevan dengan penelitian yang telah dilakukan.

1.7 Sistematika Penelitian

Laporan penelitian ini dibagi menjadi lima bab yang secara singkat dapat dijelaskan sebagai berikut :

BAB 1 PENDAHULUAN

Bab pertama penelitian ini berisi latar belakang, rumusan masalah, tujuan dan manfaat penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab kedua penelitian ini berisi teori-teori yang berhubungan dengan penelitian ini seperti tanaman jeruk dan penyakit yang diteliti serta mengenai metode *You Only Look Once* (YOLO) dan alat-alat bantu yang digunakan untuk menyelesaikan penelitian ini.

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ketiga penelitian ini berisi penjelasan data yang digunakan, analisis sistem yang akan dibangun, serta perancangan antarmuka aplikasi yang akan dibuat.

BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab keempat penelitian ini berisi penjelasan implementasi sistem berdasarkan rancangan yang dibuat, prosedur pengoperasian aplikasi, rincian pelatihan sistem, dan hasil pengujian sistem.

BAB 5 KESIMPULAN DAN SARAN

Bab kelima penelitian ini berisi kesimpulan dari hasil penelitian yang dilakukan dan saran terhadap penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1. Tanaman Jeruk

Jeruk adalah buah yang berasal dari keluarga *Rutaceae* dan memiliki kulit cukup tebal dan memiliki daging beserta biji. Buah ini memiliki rasa yang asam dan manis, tergantung pada jenisnya. Jeruk umumnya berwarna jingga, tetapi terdapat juga buah yang berwarna kuning ataupun hijau tergantung jenis buahnya. Jeruk biasanya dikonsumsi sebagai buah segar, atau diproses menjadi minuman seperti jus atau sirup. Jeruk dikenal kaya akan vitamin C dan nutrisi lainnya, serta memiliki manfaat kesehatan yang baik bagi tubuh manusia (Anggreani, 2020).

2.2. Penyakit Tanaman Jeruk

Penyakit tanaman jeruk adalah salah satu faktor paling memengaruhi penurunan produksi buah jeruk. Tanaman jeruk dapat dianggap sakit jika kondisi fisiknya menyimpang dari keadaan yang normal, contohnya daun terlihat layu, tidak munculnya bunga, buah menjadi busuk, dan lain-lain. Penyakit pada tanaman jeruk bermacam-macam, diantaranya karena terkena serangan bakteri, virus, ataupun kekurangan gizi/unsur hara. Gambar 2.1 merupakan contoh tanaman jeruk yang terkena penyakit *canker* yang menyebabkan batang tanaman mengalami kerusakan. Sementara untuk Gambar 2.2 menunjukkan contoh daun yang sehat tanpa terkena penyakit.



Gambar 2.1 Tanaman jeruk yang rusak akibat terkena penyakit canker

(Referensi : (Alberto , et al., 2020))



Gambar 2.2 Daun jeruk sehat

(Sumber : <https://www.klikdokter.com/gaya-hidup/diet-nutrisi/manfaat-daun-jeruk-untuk-kesehatan>)

2.2.1 *Citrus Moss* (*lumut daun jeruk*)

Citrus Moss adalah penyakit pada tanaman jeruk yang disebabkan oleh berbagai jenis lumut, seperti *Heterochlorella*, *Sporidiobolus*, dan *Cladosporium* (Lei, et al., 2020). Penyakit ini dapat dikenali dari gejalanya, seperti pertumbuhan tebal pada struktur daun yang dilapisi oleh lapisan hijau berupa lumut. Hal ini mengakibatkan gangguan pada penerimaan cahaya matahari oleh daun jeruk. Penyakit ini umumnya disebabkan oleh kondisi lingkungan yang memiliki kelembaban tinggi. Pengendalian penyakit ini dapat dilakukan dengan dua cara, yakni melakukan pengikisan lumut secara manual pada daun dan menggunakan penyemprotan fungisida pada daun yang terinfeksi penyakit ini. Daun yang terkena penyakit lumut daun dapat dilihat pada Gambar 2.3 berikut.



Gambar 2.3 Daun jeruk yang terkena penyakit lumut daun
(Referensi : (Lei, et al., 2020))

2.2.2 *Citrus Canker* (*kanker jeruk*)

Kanker jeruk adalah penyakit tanaman jeruk yang disebabkan oleh bakteri *X. axonopodis* pv. *citri*. atau disebut juga *Citrus Bacterial Canker* (CBC). Penyakit ini menyerang daun, ranting, dan buah pohon sitrus, menyebabkan lesi, keropeng, dan bercak pada kulit buah seperti yang terlihat pada Gambar 2.4. Gejala-gejala ini dapat menyebabkan buah kehilangan nilai komersialnya, serta dapat menyebabkan penurunan produksi dan kualitas buah. Penyakit ini dapat dikendalikan dengan penahanan angin, penggunaan bakterisida berbahan dasar tembaga, dan pengendalian hama penggerek daun jeruk (Alberto , et al., 2020). Penyakit ini sangat menular dan dapat menyebar melalui air hujan, angin, dan aktivitas manusia seperti pemangkasan dan peralatan pertanian yang tidak dibersihkan dengan baik.



Gambar 2.4 Daun jeruk yang terkena penyakit *canker*
(Sumber : <https://content.govdelivery.com/accounts/USDAAPHIS/bulletins/30df76f>)

2.2.3 *Citrus Greasy Spot* (karat daun)

Citrus Greasy Spot pada jeruk atau yang dikenal juga dengan nama penyakit Karat daun adalah penyakit yang disebabkan oleh jamur *Mycosphaerella citri* atau sebelumnya disebut *Zasmidium citri-griseum* (Aguilera-Cogley, et al., 2023). Ciri-ciri penyakit karat daun ini melibatkan gejala-gejala berupa bercak-bercak minyak berwarna gelap yang muncul pada permukaan atas daun. Bercak-bercak ini dapat berukuran kecil hingga besar. Sama halnya dengan lumut daun, daun-daun yang terinfeksi karat daun cenderung mengalami penurunan daya fotosintesis karena lapisan lilin yang berlebihan menghambat penyerapan cahaya matahari. Gambar 2.5 merupakan contoh daun yang terkena penyakit karat daun. Terlihat bercak hitam di satu bagian daun yang merupakan gejala daun yang terserang penyakit ini. Perawatan untuk penyakit ini cukup mudah. Perawatan terbaik adalah dengan menggunakan salah satu fungisida tembaga yang tersedia dan menyemprot pohon dengan fungisida tersebut. Gunakan fungisida tembaga sesuai petunjuk untuk membunuh jamur pohon jeruk.



Gambar 2.5 Daun jeruk yang terkena penyakit karat daun

(Sumber : <http://www.spchcmc.vn/EN/Plant-doctor-Detail/GREASY-SPOTS-DISEASE-ON-CITRUS-5-9510.html>)

2.3. Citra Digital

Citra digital adalah representasi dari fungsi intensitas cahaya dalam bentuk diskrit pada bidang dua dimensi (Ratna, 2020). Secara umum, citra digital adalah cara untuk menunjukkan seberapa terang atau gelapnya sesuatu di dalam gambar, tapi disajikan

dalam bentuk kotak-kotak di dalam dua dimensi. Citra digital menjadi representasi digital dari sebuah gambar atau foto yang dihasilkan melalui proses pemrosesan digital. Proses ini melibatkan konversi sinyal analog menjadi sinyal digital menggunakan perangkat keras seperti kamera digital atau *scanner*.

2.4. Computer Vision

Computer vision adalah bidang ilmu pengetahuan di bagian teknologi komputer yang bertujuan untuk mengembangkan pengidentifikasi objek pada komputer menggunakan input dari sensor, *artificial intelligence*, *machine learning*, dan *deep learning* sehingga diharapkan dapat mereplikasi cara kerja sistem penglihatan manusia. *Computer vision* sudah diterapkan secara luas di berbagai kepentingan, seperti *medical imaging*, kendaraan tanpa pengemudi, pembuatan model 3D, *motion capture*, pengenalan sidik jari, kamera pengawas (*security camera*), dan *retail* (Szeliski, 2022).

2.5. Object Detection

Object Detection (pendeteksian objek) adalah teknik dalam bidang *computer vision* yang bertujuan untuk mengenali objek-objek seperti buah, mobil, manusia, dan benda lainnya, terutama melalui gambar atau rekaman video. Secara khusus, deteksi objek menarik kotak pembatas di sekitar objek yang terdeteksi ini, yang memungkinkan kita menemukan lokasi objek tersebut (Saputra & Imran, 2023).



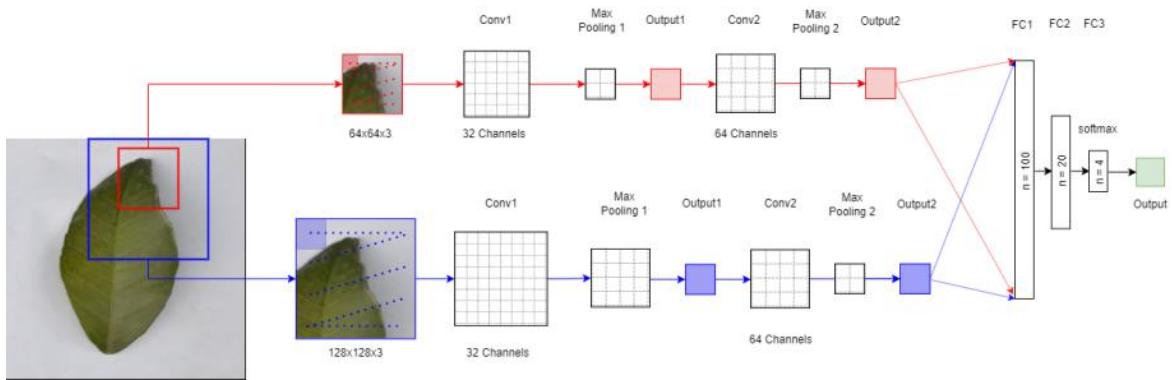
Gambar 2.6 Contoh penerapan object detection

(Sumber : docs.ultralytics.com/)

2.6. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN), yang diperkenalkan oleh LeCun et al., merupakan kelas jaringan saraf tiruan/*Artificial Neural Network* (ANN) yang

terinspirasi secara biologis pada bagaimana otak manusia bekerja. CNN diperkenalkan oleh LeCun dimana dapat menyelesaikan persamaan dengan melewatan X melalui serangkaian filter konvolusional dan non-linearitas sederhana. Pelatihan CNN dilakukan dengan menggunakan data latih yang telah dilabeli (*supervised learning*).



Gambar 2.7 Arsitektur umum *Convolutional Neural Networks*

(Referensi : (Schlegi, et al., 2015))

CNN termasuk ke dalam jenis *Multi-Layer Perceptrons* (MLP), yaitu jenis arsitektur *neural network* yang terdiri dari satu atau lebih lapisan di antara lapisan *input* dan *output*. Arsitektur CNN terdiri atas lapisan-lapisan konvolusi dan *pooling*. Lapisan konvolusi memiliki tugas untuk ekstraksi fitur-fitur pada citra melalui operasi konvolusi dengan filter atau kernel konvolusi yang terdiri dari sekumpulan *weights* (bobot) dan bias. Sementara itu, lapisan *pooling* memiliki tugas mengurangi dimensi citra, yaitu dengan cara mengabaikan informasi yang tidak dibutuhkan dan mempertahankan fitur-fitur penting. Setelah melewati lapisan konvolusi dan *pooling*, informasi fitur tersebut yang didapat akan diteruskan ke lapisan *fully connected* yang memiliki tugas dalam menghasilkan keluaran akhir. Arsitektur CNN dirancang untuk memroses data citra dan menghasilkan *output* yang sesuai dengan informasi visual pada citra tersebut. Teknik *backpropagation* dilakukan untuk mengoptimalkan *weight* dan bias pada setiap lapisan.

Backpropagation adalah teknik penting dalam pelatihan *Artificial Neural Network* (ANN) karena memungkinkan model untuk memperbarui parameter-parameternya berdasarkan gradien yang dihitung, sehingga model dapat memperbaiki prediksi dan membuat hasil lebih akurat. Dalam *backpropagation*, kesalahan yang dihasilkan oleh model dikirim kembali dari lapisan *output* ke lapisan-lapisan

sebelumnya dengan memanfaatkan aturan rantai (*chain rule*) dari turunan untuk menghitung gradien fungsi kerugian terhadap setiap parameter dalam jaringan.

2.7. You Only Look Once (YOLO)

YOLO pertama kali diperkenalkan oleh Joseph Redmon et al. yang dipublikasikan pada tahun 2016. Sesuai namanya, YOLO atau *You Only Look Once* merujuk pada cara kerja metode ini yang merupakan suatu metode detektor yang memiliki pendekatan terpadu (*unified*), di mana satu jaringan saraf tunggal mampu langsung memprediksi kotak pembatas (*Bounding Box*) dan probabilitas kelas pada seluruh gambar dalam satu pemindaian. Model YOLO mampu mengolah gambar input dengan kecepatan hingga 45 FPS (*frame per second*), dan varian jaringan yang lebih ringan, seperti *Fast YOLO*, bahkan dapat mencapai 155 FPS. Kelebihan tersebut membuat YOLO menjadi salah satu algoritma deteksi tercepat dibandingkan dengan metode real-time lainnya, seperti Fast R-CNN dan Faster R-CNN yang hanya mampu mencapai sekitar 0.5 FPS dan 7 FPS. Kecepatan ini dicapai melalui penerapan teknik perhitungan *single shot detection*, dimana *Convolutional Neural Network* (CNN) hanya dieksekusi sekali dalam proses deteksi objek (Maleh, et al., 2023).

2.7.1 YOLOv1

YOLOv1 merupakan versi pertama dari metode YOLO. YOLOv1 memiliki 24 lapisan konvolusi dan 2 lapisan *fully-connected* yang terinspirasi dari arsitektur model *GoogleNet*. YOLO menggunakan lapisan konvolusi berukuran 1x1 untuk mengurangi jumlah *feature maps* dan menjaga parameter agar tidak melebihi jumlah yang seharusnya. YOLOv1 membagi gambar input menjadi kotak berukuran S x S dan jika suatu objek terdapat di dalam kotak prediksi, maka kotak prediksi tersebut yang akan bertanggung jawab untuk mendeteksi objek tersebut (Terven, et al., 2023). YOLOv1 menggunakan kerangka kerja Darknet yang dilatih pada ImageNet-1000, namun memiliki kesulitan dalam mendeteksi objek kecil dan generalisasi objek jika gambar dari dimensi berbeda

2.7.2 YOLOv2

Pada tahun 2017, Joseph Redmon dan Ali Farhadi menerbitkan versi baru YOLO yang lebih baik, yaitu YOLOv2. YOLO versi ini menggunakan arsitektur yang

dinamakan *Darknet-19*, yaitu arsitektur yang menggunakan 19 lapisan konvolusi dan 5 layer *max-pooling*. Beberapa peningkatan YOLO2 dibandingkan versi sebelumnya yaitu (Terven, et al., 2023) :

1. *Batch Normalization* pada semua lapisan konvolusional meningkatkan konvergensi dan bertindak sebagai pengatur untuk mengurangi *overfitting*.
2. Peningkatan resolusi untuk input yang awalnya ditargetkan dapat melatih data berukuran 224x224 meningkat menjadi 448x448 yang dilatih pada model ImageNet dengan sepuluh epoch.
3. YOLOv2 tidak lagi menggunakan *Dense Layer*
4. Penggunaan *Anchor Box*, yaitu sekumpulan kotak yang memiliki bentuk yang telah ditentukan. Beberapa *anchor box* ditetapkan untuk setiap *sel grid* lalu sistem memprediksi koordinat dan kelas dari setiap *anchor box*. Hasil keluaran jaringan sebanding dengan jumlah *anchor box* per *sel grid*.
5. Pengelompokan atau penggolongan kotak pembatas (bounding boxes) berdasarkan karakteristik atau dimensi tertentu. Mereka memilih kotak-kotak pembatas yang baik dari hasil klastering dengan menggunakan *K-Means* yang dimana menghasil lima kotak yang baik. Lima kotak pembatas tersebut memberikan keseimbangan yang baik antara kemampuan untuk mendekripsi objek (*recall*) dan kompleksitas model yang digunakan untuk melakukan prediksi.
6. YOLOv2 mempunyai cara berbeda dalam memprediksi di mana objek berada. Beda dengan metode lain yang memperkirakan pergeseran, YOLOv2 mengikuti pendekatan yang lebih sederhana dengan menebak koordinat lokasi terkait sel grid. Jadi, jaringan ini membuat prediksi untuk lima kotak pembatas di setiap sel grid. Masing-masing kotak memiliki lima nilai, yaitu tx, ty, tw, th, dan to.
7. Dibandingkan dengan YOLOv1, YOLOv2 menghilangkan satu lapisan *pooling* untuk mendapatkan *feature maps* atau sel *grid* berukuran 13×13 untuk gambar berukuran 416×416 . YOLOv2 menggunakan lapisan *passthrough* yang mengambil peta fitur berukuran $26 \times 26 \times 512$ dan mengatur ulangnya dengan menggabungkan fitur-fitur yang berdekatan ke dalam saluran yang berbeda. Dengan begitu menghasilkan peta fitur berukuran $13 \times 13 \times 2048$ yang digabungkan ke dalam dimensi saluran dengan peta beresolusi lebih

rendah berukuran $13 \times 13 \times 1024$ untuk mendapatkan peta fitur berukuran $13 \times 13 \times 3072$.

8. Pada YOLOv2 juga sudah diterapkan penggunaan arsitektur *multi-scale*. Para pengembang versi ini sudah mencoba menggunakan ukuran *input* yang berbeda-beda, yaitu dari 320×320 hingga 608×608 dan melatihnya dengan jumlah epoch sepuluh.

2.7.3 YOLOv3

YOLOv3 masih dikembangkan dan dipublikasikan oleh Joseph Redmon dan Ali Farhadi. YOLOv3 menggunakan pendekatan klasifikasi multi-label, di mana setiap label menggunakan *binary cross-entropy loss*, alih-alih menggunakan *mean square error* dalam menghitung *loss* klasifikasi. DarkNet-53 digunakan sebagai *backbone* menggantikan DarkNet-19, dimana setiap blok DarkNet-53 memiliki struktur *bottleneck* 1×1 , yang diikuti oleh filter 3×3 . DarkNet-53 memiliki beberapa lapisan konvolusi dengan *stride* 2, yang digunakan untuk mengurangi dimensi. YOLOv3 menunjukkan peningkatan yang signifikan dalam pendekripsi objek kecil dan berjalan dengan baik dalam kaitannya dengan kecepatan (Lawal, 2021).

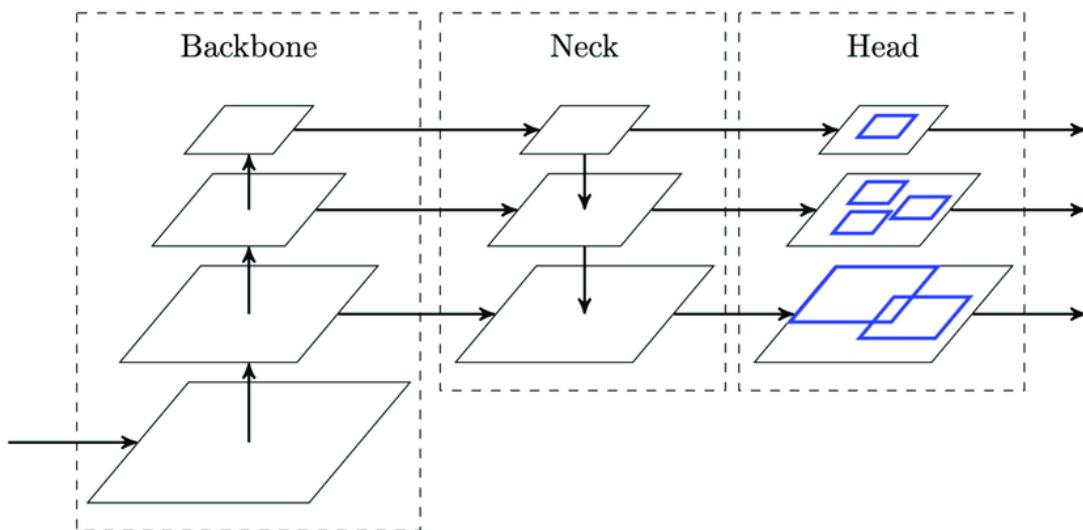
2.7.4 Backbone, Neck, dan Head

Saat itu, arsitektur pendekripsi objek mulai dipisahkan dalam tiga bagian, yaitu *backbone*, *neck*, dan *head*. Gambar menunjukkan gambaran ketiga bagian dalam arsitektur *object detection* modern.

- a) *Backbone* bertanggung jawab untuk mengekstraksi fitur dari gambar *input*. Biasanya merupakan kumpulan jaringan saraf konvolusional (CNN) yang dilatih pada tugas klasifikasi gambar skala besar, seperti *ImageNet*. *Backbone* mengambil fitur dengan hierarki skala yang berbeda. Fitur tingkat rendah (misalnya tepi dan tekstur) diekstraksi di lapisan awal dan fitur tingkat tinggi (misalnya bagian objek dan informasi semantik) diekstraksi di lapisan yang lebih dalam.
- b) *Neck* merupakan komponen perantara yang menghubungkan *backbone* dan *head*. Fungsi *neck* adalah mengumpulkan dan menyempurnakan fitur yang diekstraksi oleh *backbone*, sering juga berfokus pada peningkatan informasi

spasial dan semantik di berbagai skala. Bagian *neck* dapat mencakup lapisan konvolusional tambahan, jaringan piramida fitur (FPN), atau mekanisme lain untuk meningkatkan representasi fitur.

- c) *Head* adalah bagian terakhir dari detektor objek modern. Bagian ini bertanggung jawab dalam membuat prediksi berdasarkan fitur yang sudah diberikan oleh *backbone* dan *neck*. Bagian *head* biasanya terdiri dari satu atau lebih subjaringan yang masing-masing memiliki tugas tertentu seperti melakukan klasifikasi, lokalisasi, dan, yang lebih baru, segmentasi *instance* dan estimasi pose. *Head* memproses fitur yang disediakan *neck*, menghasilkan prediksi untuk setiap kandidat objek. Pada akhirnya, langkah pasca-pemrosesan, seperti penekanan non-maksimum (NMS), menyaring prediksi yang tumpang tindih dan hanya mempertahankan deteksi yang paling meyakinkan.



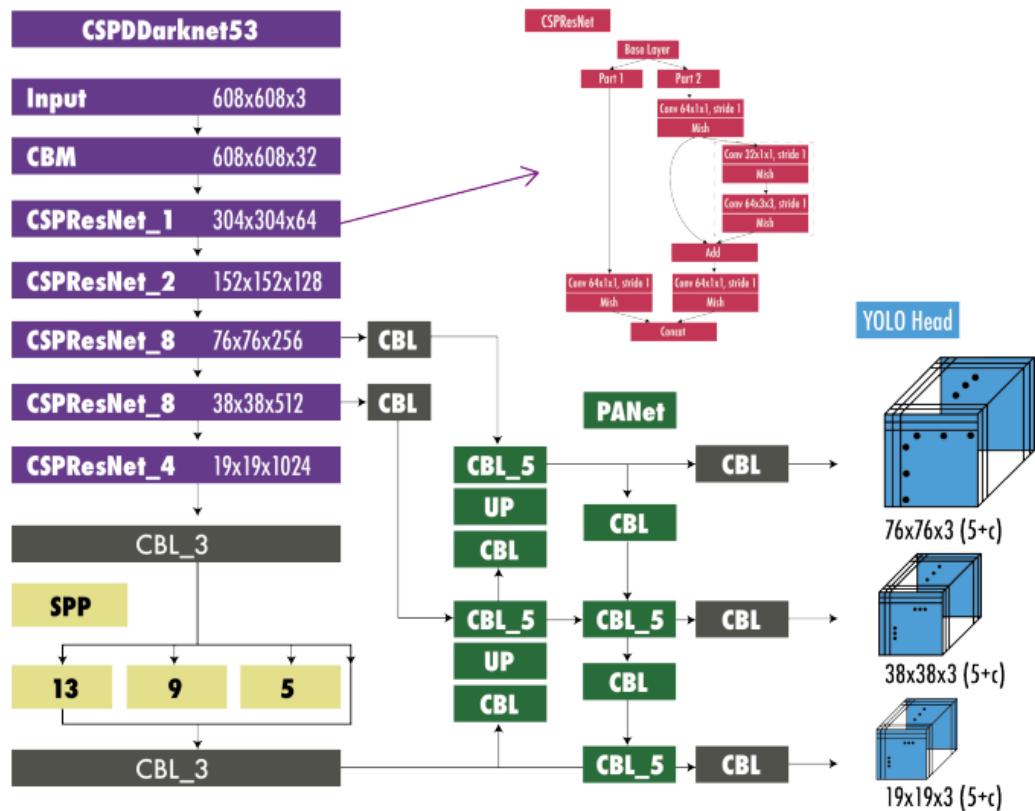
Gambar 2.8 Arsitektur umum pendeksi objek

(Sumber : <https://medium.com/>)

2.7.5 YOLOv4

YOLOv4 adalah versi YOLO yang dikembangkan oleh Alexey Bochkovsky, Chien-Yao Wang, and Hong-Yuan Mark Liao (Bochkovskiy, et al., 2020). YOLOv4 menggunakan teknik pengembangan baru yang dinamakan *Bag-of-Freebies* (BoF) dan *Bag-of-Tricks* (BoT). *Bag-of-Freebies* (BoF) dan *Bag-of-Tricks* (BoT) adalah teknik pengembangan baru yang digunakan dalam YOLOv4. BoF adalah teknik yang

digunakan untuk meningkatkan performa model dengan menambahkan beberapa fitur tambahan, seperti augmentasi data, regularisasi, dan optimasi. Sedangkan BoT adalah teknik yang digunakan untuk meningkatkan akurasi model dengan menambahkan beberapa fitur tambahan, seperti penggunaan *learning rate schedule*, penggunaan *focal loss*, dan penggunaan *self-adversarial training*. Kedua teknik ini digunakan untuk meningkatkan performa dan akurasi YOLOv4 dalam pendekalian objek. Pengembang mencoba menggunakan beberapa model arsitektur sebagai backbone dari versi ini, seperti ResNeXt50, EfficientNet-B3, dan Darknet-53. Arsitektur dengan kinerja terbaik adalah Darknet-53 yang dimodifikasi dengan CSPNet, dan *Mish activation function*.



Gambar 2.9 Arsitektur YOLOv4 untuk Object Detection (Terven, et al., 2023)

2.7.6 YOLOv5

Di tahun yang sama dengan rilisnya YOLOv4 (pada bulan April tahun 2020), tepatnya dua bulan setelah dipublikasikannya YOLOv4, Glenn Jocher bersama timnya merilis YOLOv5 di bulan Juni tahun 2020. YOLOv5 menggunakan pengembangan di Pytorch, sebagai alternatif ke Darknet yang digunakan dalam YOLOv4. YOLOv5

menggunakan algoritma AutoAnchor, yang adalah pre-training tool yang dikembangkan oleh Ultralytics. AutoAnchor memeriksa dan mengatur anchor boxes jika mereka tidak sesuai dengan dataset dan setting pengujian, seperti ukuran gambar. AutoAnchor menggunakan fungsi k-means untuk membuat kondisi awal untuk algoritma evolusi genetik (GE). GE algoritma mengembangkan anchor boxes selama 1000 generasi default, menggunakan *loss* CIoU dan *fitness function Best Possible Recall* (Zheng, et al., 2020). YOLOv5 menunjukkan peningkatan yang signifikan dalam performa dan akurasi pendekripsi objek

2.7.7 YOLOv6

YOLOv6 dipublikasikan pada September 2022 oleh Meituan Vision AI Department melalui ArXiv (Li, et al., 2022). YOLOv6 adalah model deteksi objek satu tahap yang mengungguli model-model mutakhir sebelumnya seperti YOLOv5, YOLOX, dan PP-YOLOE dalam hal akurasi dan kecepatan. Model ini terdiri atas backbone dengan blok *RepVGG* atau *CSPStackRep*, *PAN (Path Aggregation Network) topology neck*, dan *head* yang menggunakan *hybrid-channel strategy*. Model ini mencapai kinerja yang lebih kuat daripada YOLOv5 ketika diuji dengan dataset MS COCO. YOLOv6 menyediakan berbagai model yang telah dilatih sebelumnya dengan skala yang berbeda. Peningkatan arsitektur dalam YOLOv6 mencakup pengenalan modul BiC (*Bidirectional Concatenation*) di bagian *neck detector* dan strategi *Anchor-Aided Training* (AAT). Model ini juga menggabungkan teknik kuantisasi yang disempurnakan dengan menggunakan kuantisasi pasca-pelatihan dan penyulingan berdasarkan saluran, sehingga menghasilkan detektor yang lebih cepat dan lebih akurat.

2.7.8 YOLOv7

YOLOv7 adalah sebuah model deteksi objek tahap tunggal yang diterbitkan di ArXiv pada bulan Juli 2022 oleh para penulis yang sama dengan YOLOv4 dan YOLOR (Wang, et al., 2023). Pada saat itu, YOLOv7 melampaui semua detektor objek yang diketahui dalam hal kecepatan dan akurasi dalam kisaran 5 FPS hingga 160 FPS. Seperti YOLOv4, YOLOv7 dilatih hanya menggunakan kumpulan data MS COCO tanpa backbone yang sudah melakukan pre-train. YOLOv7 memberikan beberapa

perubahan arsitektur dan serangkaian metode *bag-of-freebies*, yang meningkatkan akurasi tanpa memengaruhi kecepatan inferensi, hanya memengaruhi waktu pelatihan.

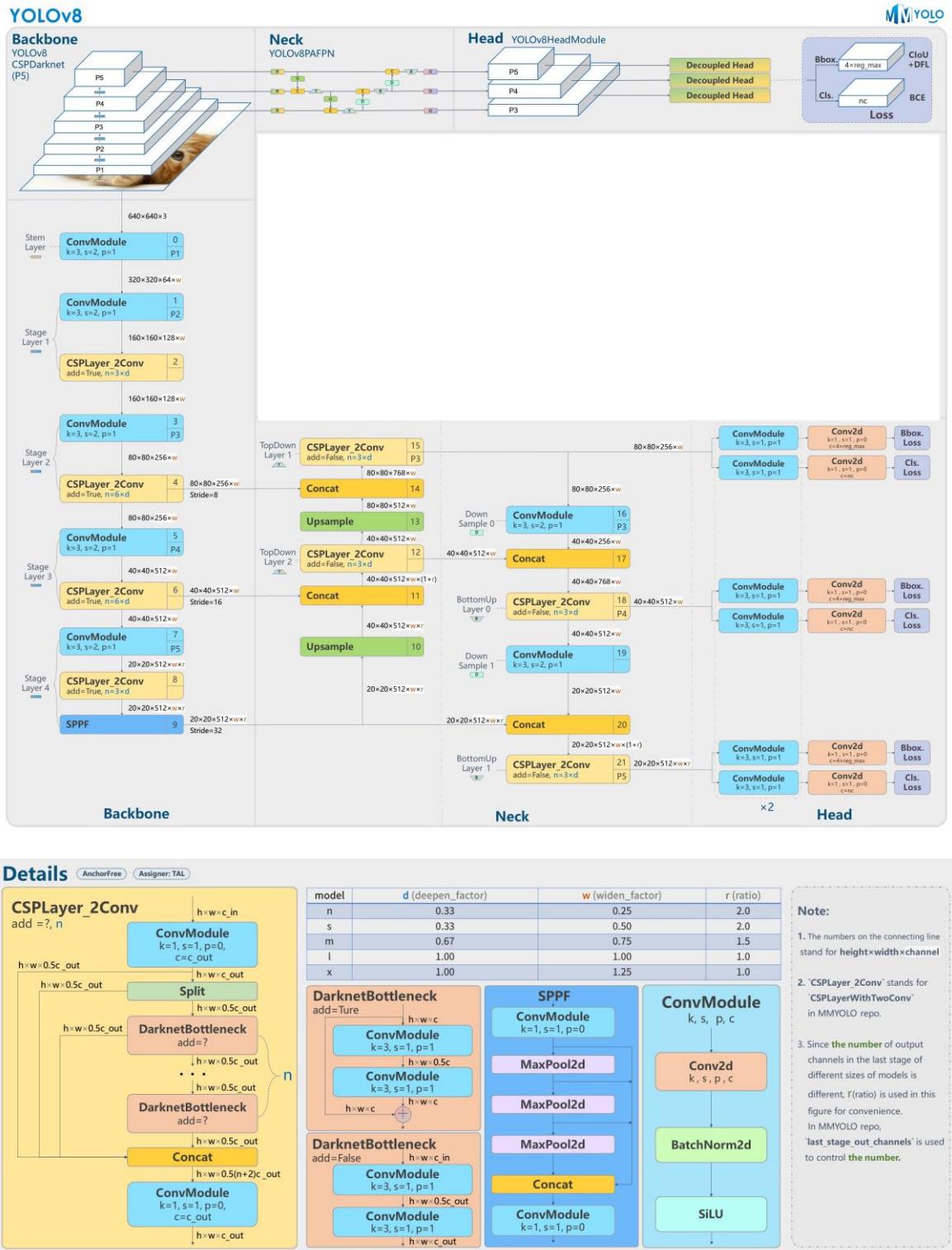
Dalam YOLOv7, terdapat beberapa perubahan arsitektur utama yang meningkatkan kecepatan dan akurasi pendekripsi objek. Model ini memiliki beberapa varian, seperti YOLOv7, YOLOv7-tiny, dan YOLOv7-W6, yang dioptimalkan untuk komputasi GPU dan penggunaan parameter yang lebih efisien. YOLOv7 juga mengurangi jumlah parameter dan persyaratan komputasi, sementara meningkatkan kecepatan inferensi. Dibandingkan dengan YOLOv6, YOLOv7 memiliki arsitektur yang lebih efisien, membutuhkan lebih sedikit parameter, dan memberikan kecepatan inferensi yang lebih tinggi. Dengan menggabungkan kecepatan dan akurasi yang lebih tinggi daripada model sebelumnya dan menawarkan berbagai varian yang dioptimalkan untuk berbagai kebutuhan komputasi dan akurasi, YOLOv7 menjadi terobosan dalam pendekripsi objek.

2.7.9 YOLOv8

Pada bulan Januari 2023, Ultralytics merilis versi terbaru dari YOLO, yaitu YOLOv8. Ultralytics adalah perusahaan yang berlokasi di Los Angeles, Amerika Serikat, yang sebelumnya sudah merilis YOLOv5 pada tahun 2020. Sebagai model terbaru dari YOLO, YOLOv8 memiliki banyak kesuksesan dibandingkan dengan versi sebelumnya. Versi ini memperkenalkan fitur-fitur baru dan peningkatan kinerja, fleksibilitas, dan efisiensi. YOLOv8 dapat melakukan berbagai tugas *artificial intelligence*, termasuk deteksi objek, segmentasi, *pose estimation*, pelacakan, dan klasifikasi (Ultralytics, 2022). Arsitektur YOLOv8 dapat dilihat pada Gambar 2.9.

YOLOv8 menggunakan *backbone* yang mirip dengan YOLOv5 namun melakukan beberapa perubahan pada *CSPLayer*, yang sekarang disebut sebagai modul C2f. Modul C2f menggabungkan fitur tingkat tinggi dengan informasi kontekstual untuk meningkatkan akurasi deteksi. YOLOv8 menggunakan fungsi loss CIoU (*Complete Intersection over Union*) dan DFL (*Distribution Focal Loss*) untuk *bounding-box loss* dan entropi silang biner dalam pendekripsi kerugian klasifikasi. Penghitungan kerugian ini telah meningkatkan kinerja pendekripsi objek, terutama

ketika berhadapan dengan objek dengan skalar yang lebih kecil (Terven, et al., 2023).



Gambar 2.10 Arsitekur YOLOv8

(Sumber: <https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov8>)

YOLOv8 memiliki struktur model yang terdiri atas 3 bagian, yaitu *backbone*, *neck*, dan *head*. Pemberian nama atau angka urutan dilakukan dari blok konvolusi pertama dengan angka 0 hingga blok C2f terakhir dengan angka 21.

a) *Backbone*

Backbone adalah bagian penting pada YOLO secara umum yang disebut sebagai *feature extractor*. Bagian *backbone* adalah serangkaian lapisan konvolusional yang bertugas sebagai pengekstrak fitur yang relevan dari gambar masukan. Model yang digunakan oleh YOLOv8 pada bagian *backbone* adalah model CSPDarknet53. Bagian ini terdiri atas sejumlah blok *convolution* yang bertugas untuk mengekstrak fitur dari berbagai resolusi masukan. Proses pengekstrakan fitur pada layer konvolusi dilakukan dengan menggunakan *kernel*. *Kernel* akan bergerak ke seluruh bagian gambar dan melakukan proses pada gambar sehingga menghasilkan *feature map*.

Proses pada *backbone* ini dimulai dengan dua blok konvolusi (*ConvModule-0* dan *ConvModule-1*) dengan ukuran *kernel* tiga, *stride* dua, dan *padding* satu. Berikutnya adalah blok C2f-2 yang memiliki dua parameter, yaitu *shortcut* dan ‘n’. Parameter *shortcut* pada blok ini adalah *true*, yang berarti koneksi *shortcut* akan digunakan pada blok *bottleneck*. Sementara parameter n yang digunakan bernilai 3xd. Jika model YOLO yang digunakan adalah YOLOv8n maka nilai d sama dengan 0,33. Sehingga nilai parameter n akan menjadi satu, yang berarti jumlah blok *bottleneck* yang digunakan pada blok C2f-2 ini adalah satu. Blok yang berikutnya adalah blok *ConvModule-3* yang sama dengan dua blok pertama, lalu diikuti dengan blok C2f-4 dengan parameter *shortcut* true dan parameter ‘n’ sama dengan 6xd. Keluaran dari blok ini, selain terhubung ke blok konvolusi berikutnya, juga terhubung ke bagian *neck*. Proses tersebut berulang hingga berujung ke blok SPPF-9. SPPF digunakan setelah blok konvolusi terakhir pada bagian *backbone*. SPPF akan menghasilkan *feature map* dari berbagai variasi ukuran yang telah didapat dari blok-blok sebelumnya. Selanjutnya blok ini terhubung pada blok *upsample-10* dan *concat-11* yang ada pada *neck*.

b) *Neck*

Bagian *neck* merupakan komponen perantara yang menghubungkan *backbone* dengan *head*. Bagian ini mengumpulkan dan menyempurnakan fitur yang diekstraksi oleh *backbone*, juga meningkatkan informasi spasial dan semantik di berbagai skala.

(Jocher, 2023). Pada bagian ini, fitur yang didapat dari blok SPPF-9 pada *backbone* akan diproses pada *upsample*-10. Layer ini bertugas untuk meningkatkan resolusi *feature map* yang didapat dari SPPF-9 agar sesuai dengan *feature map* yang didapat dari blok C2f-6. Hasil dari peningkatan tersebut akan digabungkan dengan *feature map* yang didapat dari blok C2f-6 menggunakan *concat*. Berikutnya hasil penggabungan tersebut akan diberikan pada blok C2f-12 berikutnya yang memiliki nilai parameter ‘n’ sama dengan 3xd namun tidak mengaktifasi koneksi *shortcut*. Setelah itu hasil dari blok C2f-12 akan melakukan penggabungan lagi dengan *feature map* yang didapatkan dari blok C2f-4. Hasil penggabungan diberikan lagi pada blok C2f-15 dengan parameter yang sama dengan blok sebelumnya.

Keluaran yang didapat pada blok inilah akan dijadikan masukan kepada blok *Detect1* yang ada pada bagian *head*. Selain menjadi masukan untuk *head*, hasil yang didapat dari C2f-15 akan diberikan kepada blok *ConvModule*-16 dengan ukuran *kernel* tiga, *stride* dua, dan *padding* satu agar dilakukan kembali ekstraksi fitur. Dengan begitu resolusi *feature map* yang didapat akan berkurang sehingga dapat dilakukan penggabungan dengan keluaran dari blok C2f-12. Berikutnya dihasilkan lagi *feature map* sehingga dapat menjadi masukan untuk blok C2f-18. *Feature map* yang didapat dari C2f-18 akan menjadi masukan kepada blok *Detect2* pada *head*. Sama seperti sebelumnya, hasil dari C2f-18 akan diproses kembali pada blok konvolusi berikutnya dan dilakukan penggabungan pada *feature map* yang didapat dari SPPF-9 lalu menjadi masukan pada C2f-21. Blok ini menjadi blok terakhir yang ada pada *neck* dan *feature map* yang didapat akan diberikan pada blok *Detect3*.

c) *Head*

Head adalah komponen terakhir dari struktur YOLO. *Head* bertanggung jawab untuk membuat prediksi berdasarkan fitur yang disediakan oleh *backbone* dan *neck*. Bagian *head* biasanya terdiri dari satu atau lebih subjaringan tugas tertentu yang melakukan klasifikasi dan lokalisasi. *Head* memproses fitur yang disediakan *neck*, menghasilkan prediksi untuk setiap objek. Bagian head bertanggungjawab dalam mengolah peta fitur yang telah diekstrak dari *backbone* serta mengolahnya lebih lanjut hingga mendapatkan keluaran akhir berupa *bounding box* dan kelas objek. Pada YOLOv8, bagian *head* dirancang untuk bekerja secara terpisah, dengan kata lain tugas objektivitas, klasifikasi, dan regresi dilakukan secara independen. Rancangan tersebut

memungkinkan setiap cabang untuk fokus pada tugasnya masing-masing sehingga dapat meningkatkan akurasi model secara keseluruhan. Untuk memroses peta fitur, *head* menggunakan serangkaian lapisan konvolusional diikuti oleh lapisan linier untuk memprediksi *bounding box* dan probabilitas kelas. Desain *head* dioptimalkan untuk kecepatan dan akurasi, dengan perhatian khusus diberikan pada banyaknya saluran dan ukuran kernel setiap lapisan untuk memaksimalkan kinerja. (Jocher, 2023).

Blok *detect* terdiri atas dua bagian, yaitu bagian yang menghasilkan *bounding box* dan bagian yang memberikan prediksi kelas. Kedua bagian memiliki jumlah blok konvolusi yang sama, yaitu dua buah blok konvolusi dengan ukuran *kernel* tiga, *stride* satu, dan *padding* satu. Blok konvolusi pertama dihubungkan dengan layar conv2d sehingga dapat diproses dan menghasilkan *bounding box*. Blok konvolusi kedua dihubungkan juga dengan layar conv2d sehingga dapat diproses dan menghasilkan prediksi kelas. Komponen penyusun blok deteksi dapat dilihat pada gambar 3.16. *Feature map* yang didapat dari *neck* masing-masing diproses pada tiap blok pendekripsi. Hasil yang didapat dari blok C2f-15 akan menjadi masukan pada blok *Detect1* yang bertugas mendekripsi objek berukuran kecil. Lalu hasil yang didapat dari blok C2f-18 akan menjadi masukan pada blok *Detect2* yang bertugas mendekripsi objek berukuran sedang. Dan hasil yang didapat dari blok C2f-21 akan menjadi masukan pada blok *Detect3* yang bertugas mendekripsi objek berukuran lebar.

Ketiga bagian dari struktur YOLO tersebut juga terdiri atas blok-blok yang saling terhubung seperti *Convolution Module*, *C2f block*, *Bottleneck*, dan SPPF.

a) *Convolution Module*

Blok yang paling umum digunakan adalah blok *convolution*. Blok tersebut terdiri atas 2D *convolution layer*, 2D *batch normalization*, dan SILU. (Ultralytics, 2023)

- Lapisan konvolusional 2D adalah bagian yang bertugas untuk mengekstrak fitur dari gambar masukan. Dengan menerapkan filter konvolusional (kernel) di seluruh gambar, lapisan tersebut menangkap berbagai aspek informasi visual, seperti tepi, tekstur, dan pola. Fitur yang diekstraksi ini penting untuk mengidentifikasi objek dalam gambar.
- Lapisan 2D *batch normalization* digunakan dalam blok modul konvolusi untuk meningkatkan kinerja dan stabilitas model selama pelatihan. Untuk

menghitung proses normalisasi menggunakan 2d *batch normalization* dapat melakukan tahap-tahap dan rumus berikut :

- Hitung nilai *mean* (μ) dengan menggunakan rumus 2.1 berikut :

$$\mu = \frac{1}{9} \sum_{i=1}^3 \sum_{j=1}^3 x_{ij}$$

(2.1)

- Hitung nilai *Variance* (σ^2) dengan menggunakan rumus 2.2 berikut :

$$\sigma^2 = \frac{1}{9} \sum_{i=1}^3 \sum_{j=1}^3 (x_{ij} - \mu)^2$$

(2.2)

- Normalisasikan nilai dengan menggunakan rumus 2.3 berikut :

$$x_{ij} = \frac{x_{ij} - \mu}{\sigma + \epsilon}$$

(2.3)

Dengan nilai $\epsilon = 0,001$, dimana digunakan untuk mencegah pembagian dengan nol selama langkah normalisasi

- *Scale(γ) and shift(β)*, adalah proses yang memungkinkan jaringan mempelajari distribusi optimal untuk nilai yang dinormalisasi, secara efektif mengadaptasi distribusi data agar lebih sesuai dengan persyaratan lapisan berikutnya dalam jaringan. Proses ini dihitung dengan menggunakan rumus :

$$y_{ij} = \gamma * x_{ij} + \beta = x_{ij}$$

(2.4)

- SiLU atau fungsi aktivasi *Sigmoid Linear Unit* memiliki fungsi meningkatkan kemampuan model untuk mempelajari representasi kompleks dan meningkatkan efisiensi pelatihan. Sama halnya dengan *batch normalization* untuk menghitung proses normalisasi menggunakan SiLU dapat melakukan tahap-tahap dan rumus berikut :
 - Hitung *Sigmoid function* untuk tiap nilai pada *feature map* dengan menggunakan rumus berikut :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

(2.5)

- Hitung SiLU Activation untuk tiap nilai pada *feature map* dengan menggunakan :

$$SiLU(x) = x * \sigma(x) \quad (2.6)$$

b) *C2f block*

C2f block atau dapat disebut juga sebagai *CSP Bottleneck with 2 Convolutions block* adalah blok yang berfungsi untuk mengambil *feature map* yang dihasilkan oleh lapisan konvolusional dan mengubahnya menjadi format yang sesuai untuk klasifikasi dan regresi kotak pembatas. Blok C2f memuat sebuah blok konvolusi yang dimana hasil dari *feature map* yang didapat akan melakukan *splitting*. Hasil *splitting* terbagi menjadi dua, yang pertama mengarah ke blok *bottleneck* dan yang satunya mengarah langsung ke blok *concat*. Blok C2f dapat memiliki banyak blok *bottleneck* yang direpresentasikan sebagai ‘n’. Di ujung blok terdapat satu blok konvolusi akhir sebagai pember hasil untuk diberikan pada blok berikutnya.

c) *Bottleneck*

Bottleneck block adalah blok yang terdiri dari dua lapisan konvolusional 3x3 dengan koneksi *shortcut*. Penggunaan dua lapisan konvolusional 3x3 memungkinkan ekstraksi dan pemrosesan fitur yang lebih kompleks, sementara koneksi *shortcut* membantu menjaga informasi spasial dan mengurangi jumlah parameter. Koneksi *shortcut* dapat dimiliki dan dapat juga tidak dimiliki oleh sebuah blok *bottleneck* tergantung penggunaanya.

d) *SPPF*

Lapisan SPPF adalah lapisan yang awalnya bernama lapisan SPP (*Spatial Pyramid Pooling*) yang digunakan pada YOLOv3 lalu dikembangkan menjadi SPPF pada YOLOv5 yang dimana adalah versi yang lebih optimal. Tujuan dari digunakannya SPPF adalah untuk membuat peta fitur yang lebih bervariasi dengan tersedianya representasi dari skala-skala yang berbeda. Dengan menggabungkan skala yang berbeda, SPPF memungkinkan model menangkap fitur di berbagai tingkat abstraksi. Hal ini khususnya berguna dalam pendekripsi objek, di mana objek dengan ukuran berbeda mungkin perlu didekripsi (Jocher, 2023). Di dalam blok SPPF terdapat blok konvolusi di awal, diikuti dengan tiga

2D *maxpooling layer*. Setiap hasil yang diberikan oleh tiap *maxpooling layer* akan digabungkan sebelum diberikan pada blok konvolusi yang ada di akhir.

2.8 Confusion Matrix

Confusion matrix adalah media yang digunakan dalam pengukuran kinerja yang digunakan dalam masalah klasifikasi pembelajaran mesin. Confusion matrix terdiri atas empat jenis nilai yang dihasilkan dengan mengevaluasi efektivitas model klasifikasi dengan membandingkan kelas yang diprediksi dengan kelas yang diharapkan. Keempat nilai tersebut adalah *True Positives (TP)*, *False Positives (FP)*, *True Negatives (TN)*, dan *False Negatives (FN)*.

- True Positive (TP)*: didapat ketika model memprediksi kelas yang benar dan data yang diberikan juga berada dalam kelas yang benar.
- False Positive (FP)*: didapat ketika model memprediksi kelas yang benar tetapi data yang diberikan berada dalam kelas yang tidak benar.
- True Negative (TN)*: didapat ketika model memprediksi kelas yang tidak benar dan data yang diberikan juga berada dalam kelas yang tidak benar.
- False Negative (FN)*: didapat ketika model memprediksi kelas yang tidak benar tetapi data yang diberikan berada dalam kelas yang benar.

Untuk penerapan *confusion matrix* dua kelas dapat dilihat pada gambar 2.11 :

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Gambar 2.11 *Confusion Matrix* untuk prediksi dua kelas

(Sumber: <https://towardsdatascience.com/>)

Dari hasil confusion matrix tersebut, maka nilai *precision*, *recall*, mAP, *F1-score*, dan *accuracy* dapat didapat dengan cara sebagai berikut:

2.8.1 *Precision*

Precision merupakan persentase dari semua prediksi yang benar. *Precision* dapat diterapkan untuk mengukur persentase dari semua prediksi yang benar dari semua prediksi yang dilakukan oleh model. Misalnya, jika model memprediksi 1000 kasus dan hanya 800 kasus benar, maka precision model adalah 80%. *Precision* ini dapat dihitung dengan rumus 2.7 berikut :

$$\text{Precision} = (TP / (TP + FP)) \quad (2.7)$$

2.8.2 *Recall*

Recall merupakan persentase dari semua kasus yang benar yang ditemukan oleh model. Dalam konteks confusion matrix, recall dapat diterapkan untuk mengukur persentase dari semua kasus yang benar yang ditemukan oleh model dari semua kasus yang ada. Misalnya, jika model memprediksi 1000 kasus dan hanya menemukan 800 kasus yang benar, maka recall model adalah 80%. *Recall* ini dapat dihitung dengan rumus 2.8 berikut:

$$\text{recall} = (TP / (TP + FN)). \quad (2.8)$$

2.8.3 *Mean Average Precision (mAP)*:

Mean Average Precision (mAP) merupakan rata-rata dari *precision* dan *recall*. Dalam konteks *object detection*, mAP digunakan untuk mengukur kinerja model dalam mendeteksi objek. Nilai ini dapat dihitung dengan rumus 2.9 berikut:

$$\text{mAP} = (1 / n) * \Sigma (\text{precision} * \text{recall}) \quad (2.9)$$

2.8.4 *F1-score*

F1-score merupakan nilai harmonic mean dari precision dan recall. *F1-score* digunakan untuk mengukur kinerja model dalam mengklasifikasikan data. *F1-score* ini dapat dihitung dengan rumus:

$$F1\ Score = 2 * (precision * recall) / (precision + recall) \quad (2.10)$$

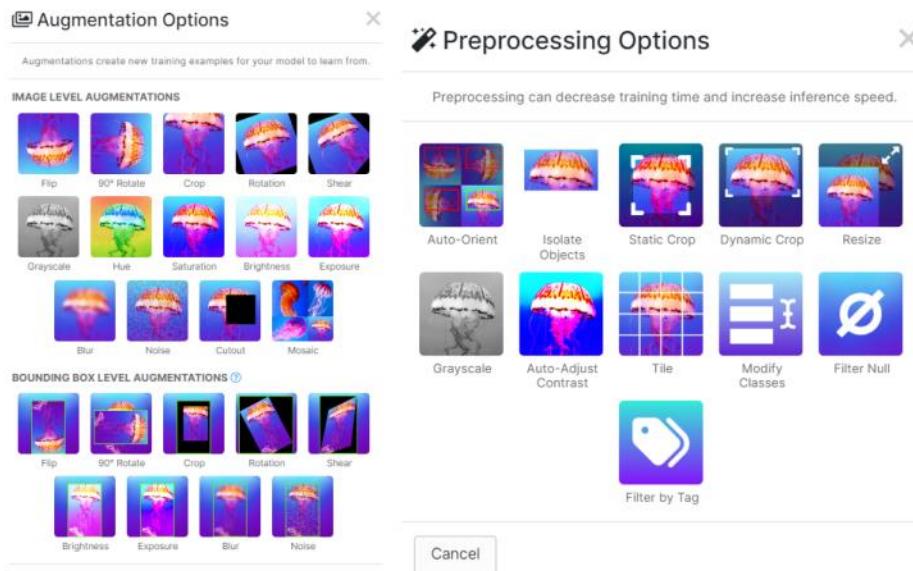
2.8.5 *Accuracy*

Accuracy merupakan persentase dari semua prediksi yang benar dari semua prediksi yang dilakukan oleh model. Dalam konteks confusion matrix, accuracy dapat diterapkan untuk mengukur persentase dari semua prediksi yang benar dari semua prediksi yang dilakukan oleh model. Misalnya, jika model memprediksi 1000 kasus dan hanya 900 kasus benar, maka accuracy model adalah 90%. Accuracy ini dapat dihitung dengan rumus:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (2.11)$$

2.9 Roboflow

Roboflow adalah sebuah platform yang dikembangkan oleh perusahaan Roboflow yang memiliki banyak fungsi dalam mempermudah pekerjaan yang berhubungan dengan pengembangan pembelajaran mesin. Roboflow menyediakan tiga kategori yang dapat dipilih pengguna, yaitu *object detection*, *classification*, dan *instance segmentation*. Roboflow memberikan pilihan pada pengguna juga dalam melakukan pembagian data, yaitu *data training*, *validation*, dan *testing*. Rasio pembagian data dapat disesuaikan dengan keinginan pengguna Roboflow, misalnya seperti rasio pembagian pada umumnya yaitu 7:2:1. Roboflow menawarkan pengguna fitur yang dapat memproses dataset tersebut serta menggunakan alat untuk anotasi objek menggunakan *bounding box*, *pre-processing* seperti *grayscale* dan augmentasi. Fitur *preprocessing* dan augmentasi yang disediakan oleh Roboflow juga beragam. Gambar 2.12 menunjukkan fitur augmentasi dan *preprocessing* yang disediakan oleh Roboflow.



Gambar 2.12 Fitur augmentasi dan *preprocessing* Roboflow

(Sumber: <https://app.roboflow.com/>)

Penggunaan fitur augmentasi gambar pada Roboflow memiliki batas pada pengguna yang tidak menggunakan akun level premium. Roboflow hanya memberikan batas maksimal augmentasi tiga kali lipat dari data asli pengguna. Misalkan pengguna awalnya memiliki tiga ratus gambar yang sudah dikelola dalam Roboflow, pada kasus penulis sudah dilakukan pelabelan, maka Roboflow akan memberikan augmentasi tiga kali lipat pada data training saja. Sementara untuk pengguna *premium*, augmentasi data dapat dilakukan hingga lima puluh kali jumlah data awal.

Roboflow juga menyediakan alat untuk mengorganisir visual data menggunakan CLIP *vector embeddings*, *automatik labeling* dengan *zero-shot generalization* menggunakan Segment Anything (SAM), dan *deploy model foundation* seperti SAM dan CLIP melalui hosted API atau di edge. Dalam projek *object detection*, pengguna dapat mengekspor dataset yang telah diproses dalam berbagai format, seperti JSON, XML, TXT, dan CSV. Hasil dari ekspor tersebut nantinya akan digunakan dalam pelatihan model yang akan dibuat.

2.10 Google Colab

Google Colaboratory, atau disingkat dengan “Colab”, adalah produk dari Google Research. Platform ini memungkinkan pengguna untuk menulis dan mengeksekusi kode python arbitrer melalui browser, dan sangat cocok untuk pembelajaran mesin, analisis data, dan penelitian. Secara lebih teknis, Colab adalah layanan notebook

Jupyter yang dihosting yang tidak memerlukan penyiapan untuk digunakan, sekaligus memberikan akses gratis ke sumber daya komputasi termasuk GPU.

Dengan Colab, pengguna dapat mengimpor kumpulan data gambar, melatih pengklasifikasi gambar di dalamnya, dan mengevaluasi model selama masih menggunakan bahasa pemrograman yang diterima oleh Colab. Notebook Colab mengeksekusi kode di server cloud Google, artinya pengguna dapat memanfaatkan kelebihan perangkat keras Google, termasuk GPU dan TPU, terlepas dari spesifikasi perangkat yang digunakan pengguna.

2.11 Ultralytics Hub

Ultralytics HUB adalah platform yang dikembangkan oleh perusahaan Ultralytics yang digunakan untuk pengembangan, pelatihan, dan penerapan *model vision AI*. Platform ini dirancang untuk menyederhanakan proses pembuatan model *computer vision*, sehingga dapat diakses oleh pelajar, peneliti, pengembang, dan juga untuk bisnis. Pengguna dapat dengan mudah mengelola, melatih, dan menerapkan model *computer vision*. Ultralytics mendukung berbagai tugas *computer vision*, seperti deteksi objek, segmentasi gambar, dan klasifikasi.

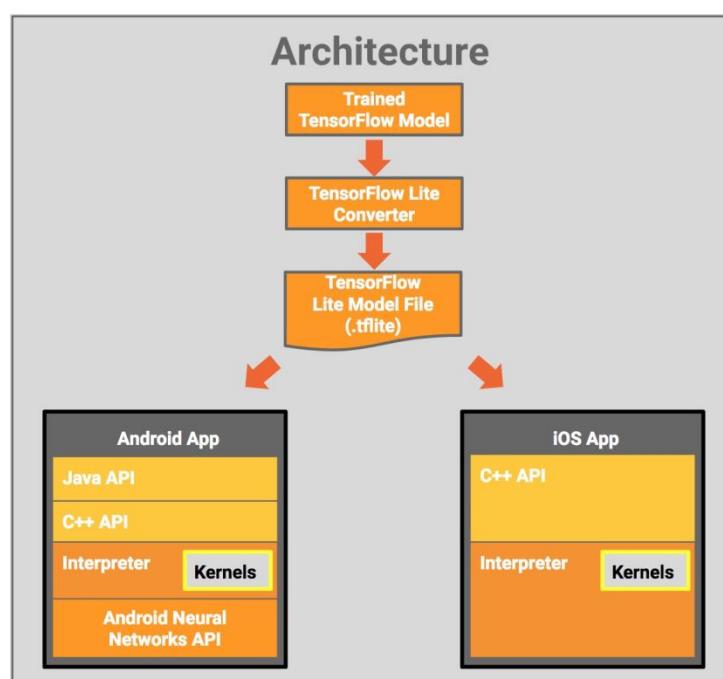
Ultralytics HUB menawarkan beberapa dataset untuk pengguna yang menggunakan platform ini untuk sekedar mencoba pembuatan model. Pengguna juga dapat mengunggah dataset yang dimiliki sendiri dengan format standar YOLO. Dataset yang telah diunggah tersebut dapat langsung dilatih dengan model yang telah disediakan, yaitu YOLOv8, YOLOv5u, dan YOLOv5u6. Untuk pengaturan pelatihan model seperti epoch, batch, dan lainnya, pengguna dapat menentukan sendiri meskipun Ultralytics sudah memiliki pengaturan bawaan. Setelah model sudah diatur, pengguna akan ditawarkan tiga pilihan untuk platform pelatihan model, yaitu Ultralytics Cloud, Google Colab, dan lokal.

2.12 Tensorflow Lite

Tensorflow lite adalah sebuah library yang diperkenalkan oleh Tensorflow untuk melakukan inferensi jaringan pada perangkat seluler, mikrokontroler (MCU), dan perangkat edge lainnya. TFLite menggunakan skema kuantisasi untuk mengurangi konsumsi daya dan memungkinkan penggunaan model machine learning di perangkat dengan komputasi terbatas (Parra, et al., 2023). Tensorflow lite mendukung berbagai

format model, seperti TensorFlow, Caffe, ONNX, dan lainnya. Dengan didukungnya model-model tersebut, pengguna dapat menggunakan model seperti tensorflow lalu mengoptimalkannya untuk penggunaan pada perangkat mobile.

TensorFlow Lite mendukung berbagai tipe data, seperti integer, float, dan boolean. TensorFlow Lite juga mendukung berbagai algoritma machine learning, seperti klasifikasi, regresi, dan deteksi objek. Untuk menggunakan TensorFlow Lite, diperlukan beberapa alat yang disediakan oleh Google, seperti TensorFlow Lite Model Maker, yang memungkinkan pengguna untuk membuat model baru atau mengoptimalkan model yang telah ada sehingga kemudian dapat diimplementasikan pada perangkat mobile. Tensorflow juga menyediakan TensorFlow Lite Interpreter yang memungkinkan pengguna untuk menggunakan model tensorflow lite dengan kode C++ atau Java.



Gambar 2.13 Arsitektur TensorFlow Lite

(Sumber : <https://blog.tensorflow.org/2018/03/using-tensorflow-lite-on-android.html>)

2.13 Android Studio

Android Studio adalah IDE (*Integrated Development Environment*) untuk *Google Android Development* yang diluncurkan pada 16 Mei 2013. Android studio berisi semua alat android yang digunakan untuk merancang, menguji, men-debug, dan membuat profil aplikasi android. Android studio menggunakan bahasa pemrograman

java dan kotlin untuk mengembangkan aplikasi. Android studio juga memberikan fitur *gradle build system*, yaitu sebuah sistem pembangunan otomatis yang digunakan untuk membantu pengelolaan proyek dalam android studio.

2.14. Penelitian Terdahulu

Hingga saat ini, penelitian terkait penyakit tanaman jeruk sudah cukup banyak dilakukan. Salah satunya adalah penelitian yang dilakukan oleh (Luo, et al., 2023). Penelitian ini bertujuan untuk meningkatkan akurasi dalam pengidentifikasi penyakit dan hama yang menyerang tanaman jeruk menggunakan *Lightweight Self-Attention YOLOv8* (Light-SA YOLOV8) secara *real time*. Ada lima jenis penyakit yang menjadi objek penelitian ini, yaitu : *anthracnose*, *citrus canker*, melanosis, scab, dan *bacterial brown spot-along* dan satu jenis serangga hama, yaitu *citrus shallow leaf moth*. Data diambil dalam waktu tiga hingga enam bulan pada tahun 2023 dari sebuah perkebunan di Qingyuan City, China. Dataset yang digunakan terdiri atas 3202 gambar yang terbagi menjadi enam kategori dan dibagi menjadi data *train*, *validation*, dan *test* dengan rasio 8:1:1. Labeling dilakukan dengan menyorot dan melaukan pengkotakan bagian daun yang memiliki gejala gangguan. Platform percobaan yang digunakan dalam penelitian ini adalah sistem operasi Ubuntu 18.04, model CPU Intel Core i7-10700, dengan kapasitas memori 32GB. model GPU kartu grafisnya adalah Nvidia GeForce RTX3090, dan kapasitas memori video 24GB. Penelitian ini diuji coba dengan menggunakan lima jenis modal dan berhasil menghasilkan akurasi presisi 92,6%, *recall* 89,4%, *F1-score* 91%, dan mAP 0.5 92,5% menggunakan model Light-SA YOLOv8.

Penelitian berikutnya adalah yang sebuah penelitian yang menggunakan metode YOLO, lebih spesifiknya menggunakan metode YOLOv5 dalam mengidentifikasi penyakit tanaman jeruk yaitu Huanglongbing (Qiu, et al., 2022). Mereka menggunakan YOLOv5l dan membuat tiga jenis model berbeda, yaitu Yolov5l-HLB1, Yolov5l-HLB2, and Yolov5l-HLB3. Ketiga model tersebut dilatih menggunakan data gambar daun jeruk sehat dan daun jeruk yang bergejala dengan total jumlah 7516 gambar tanpa augmentasi. Dari jumlah tersebut, 1413 gambar adalah daun jeruk yang sehat, 3017 gambar adalah daun jeruk yang memiliki gejala HLB (seperti bintik-bintik bercak, buah “*red-nose*”, kekurangan *zinc*, urat menguning, dan menguning seragam), dan sisanya adalah gambar daun jeruk yang memiliki gejala

penyakit lain seperti cancer, lumut daun, karat daun, dan beberapa penyakit lainnya. Model yang telah dibuat lalu dihubungkan dengan aplikasi *android* yang dapat langsung mendeteksi infeksi HLB pohon jeruk secara *real-time*. Dari ketiga model tersebut, Yolov5l-HLB2 mendapat akurasi micro F1-score tertinggi yaitu 85,19% yang dimana dapat mengenali lima gejala penyakit HLB.

Kemudian terdapat penilitian yang menggunakan YOLOv8 oleh (Talaat & ZainEldin, 2023). Penelitian ini berfokus untuk meningkatkan pendekatan daerah kebakaran dengan mengajukan sebuah sistem bernama *smart fire detection system* (SFDS). Dataset yang digunakan adalah dataset gambar situasi kebakaran, yaitu gambar-gambar yang mengandung api, baik api kecil maupun api besar, serta bertempat di dalam dan di luar ruangan. Dataset terdiri atas 26520 gambar terdiri atas gambar yang mengandung api, asap, keduanya, dan gambar situasi normal tanpa ada keduanya. Dataset dibagi menjadi 21216 untuk data latih dan 5304 untuk data uji. SFDS tersebut telah diuji dan mendapatkan hasil presisi dan recall yang tinggi dengan nilai 97.1% untuk semua kelas. Pendekatan yang diusulkan memiliki potensi untuk dapat diterapkan penggunaanya, termasuk pada manajemen keselamatan kebakaran di area publik, pemantauan kebakaran hutan, dan sistem keamanan intelijen.

Berikutnya adalah penelitian yang dilakukan oleh (Ariesdianto , et al., 2021) dalam mengidentifikasi penyakit Canker dan Ulat Peliang menggunakan *K-Nearest Neighbor*. Pada penelitian ini, data yang digunakan sebanyak 180 citra jeruk yang dibagi lagi menjadi data uji dan data latih. Dari 180 data citra tersebut dilakukan pembagian menjadi 3 kelas yaitu, 50 data citra daun jeruk sehat, 50 data citra daun jeruk yang terkena penyakit kanker dan 50 data citra daun jeruk yang terkena penyakit ulat peliang untuk data latih, sedangkan untuk data uji jumlah datanya terdiri dari 10 data. Tahap yang dilakukan berikutnya setelah pembagian data adalah melakukan *cropping* citra menjadi 500 x 500 pixel untuk mengurangi beban komputasi. Citra yang telah melakukan proses pemotongan selanjutnya akan menerima modifikasi pada ruang warna RGB. Setelah proses pemecahan komponen warna, dilakukan proses pengurangan pada masing-masing komponen warna, untuk membentuk ruang warna baru. Hal tersebut juga berguna untuk mempermudah proses selanjutnya yaitu ekstraksi fitur. Didapatkan dari pengujian setiap ruang warna RG, RB, GR, GB, BR dan BG, ruang warna GB merepresentasikan tekstur paling baik

sehingga dipilih untuk proses ekstraksi fitur tekstur GLCM (*Gray Level Co-Occurrence Matrix*). Hasil dari ekstraksi fitur digunakan untuk pengklasifikasian menggunakan K-Nearest Neighbor. Penelitian ini berhasil mengidentifikasi penyakit dengan akurasi 70% dengan variasi nilai K = 21.

Berikutnya penelitian yang dilakukan oleh (Lestari, et al., 2019) yang bertujuan untuk mengidentifikasi penyakit kanker daun, CVPD (*Citrus Vein Phloem Degeneration*) daun, dan ulat peliang daun menggunakan metode *Multicategory Support Vector Machines* (M-SVM). Penelitian ini menggunakan 150 citra daun jeruk siam berpenyakit yang terdiri atas 50 citra penyakit kanker daun, 50 citra penyakit CVPD pada daun, dan 50 citra penyakit ulat peliang daun. Data latih dan uji mengalami proses yang sama yaitu, *preprocessing*, ekstraksi fitur, dan identifikasi menggunakan M-SVM. Pengujian dilakukan sebanyak 5 kali dengan kombinasi data latih dan data uji yang berbeda-beda. Pengujian pertama menggunakan citra latih sebanyak 10 citra pada setiap jenis penyakit dan menghasilkan akurasi sebesar 54,17%, pengujian kedua menggunakan citra latih sebanyak 20 citra pada setiap jenis penyakit menghasilkan akurasi sebesar 62,22%, pengujian ketiga menggunakan citra latih sebanyak 30 citra pada setiap jenis penyakit menghasilkan akurasi sebesar 76,67%, pengujian keempat menggunakan citra latih sebanyak 40 citra pada setiap jenis penyakit menghasilkan akurasi sebesar 86,67% dan pengujian V menggunakan citra latih sebanyak 12 citra pada setiap jenis penyakit menghasilkan akurasi sebesar 53,125%. Hasil tersebut menunjukkan bahwa jumlah citra latih mempengaruhi tingkat keberhasilan identifikasi daun jeruk siam berpenyakit. Berdasarkan hasil analisa peneliti, penerapan metode M-SVM untuk mengidentifikasi penyakit tanaman jeruk berdasarkan citra daun memberikan hasil yang cukup baik.

Penelitian selanjutnya adalah penelitian yang dilakukan oleh (Yang, et al., 2023). Mereka meneliti tentang penerapan YOLOv8m pada segmentasi gambar yang tepat dari masing-masing daun yang menjadi objek penelitian. Model YOLOv8 tersebut dipadukan dengan arsitektur DeepLabv3+, yaitu sebuah arsitektur segmentasi semantik. YOLOv8m digunakan untuk mengurangi gangguan latar belakang pada tugas segmentasi daun semenetara DeepLabv3+ diusulkan agar lebih efisien dalam menangkap citra daun pada batang dan tangkai daun yang ramping. Data yang digunakan adalah gambar yang didapat dari kamera ponsel, iPads, dan kamera digital.

Penangkapan gambar dilakukan di daerah perkotaan Nanjing, Changzhou, Linyi, dan Hangzhou pada Oktober 2022. Terdapat 9763 total gambar daun yang didapat setelah dilakukan augmentasi. Hasil penelitian menggunakan penggabungan dua metode tersebut mencapai nilai rata-rata mIoU (*mean intersection over the union*) sebesar 90,8% untuk segmentasi daun pada dataset yang ada.

Penelitian yang terakhir adalah sebuah penelitian penuakit tanaman jeruk yang dilakukan oleh (Dang-Ngoc, et al., 2021). Penelitian ini bertujuan untuk meningkatkan akurasi dalam pengklasifikasian empat jenis penyakit jeruk sehingga dapat dibedakan dengan daun yang sehat melalui pemeriksaan ciri-ciri daun jeruk tersebut. Empat penyakit jeruk yang menjadi kategori penelitian ini yaitu *canker*, *sooty mold*, *greening*, dan *leaf-miner*. Tahap penelitian yang dilakukan dimulai dengan pre-processing, ekstraksi fitur, dan klasifikasi. Metode yang digunakan pada penelitian ini adalah Hierarchical Support Vector Machine (SVM). Model SVM ini memberikan tingkat deteksi daun yang terinfeksi sebesar 92,5%, dan tingkat akurasi yang tinggi sebesar 91,76%.

Tabel 2.1 berikut adalah rangkuman penelitian terdahulu yang menjadi referensi penulis dalam penggerjaan penelitian ini :

Tabel 2.1 Penelitian terdahulu

No	Peneliti	Judul Penelitian	Tahun	Keterangan
1	Luo, Dehuan, Yueju Xue, Xinru Deng, Bin Yang, Haifei Chen, and Zhujiang Mo	Citrus Diseases and Pests Detection Model Based on Self-Attention YOLOV8.	2023	Pendeteksian lima jenis penyakit jeruk dan satu jenis gangguan hama dengan menggunakan YOLOv8 yang dimodifikasi. Digunakan 3202 gambar untuk pelatihan, validasi, dan pengujian model. Hasil akhir untuk akurasi mAP 0.5 adalah 92,5%. Pendekstnsian dilakukan secara <i>real-time</i> dengan

Lanjutan **Tabel 2.1** Penelitian terdahulu

No	Peneliti	Judul Penelitian	Tahun	Keterangan
2	Qiu, R.Z., Chen,S.P., M.X., Wang, R.B., Huang, T., G.C, Zhao, J., Weng, Q.Y.iu, R..	An automatic identification system for citrus greening disease (Huanglongbing) using a YOLO convolutional neural network	2022	Deteksi penyakit Huanglongbing pada jeruk menggunakan metode YOLOv5. Dilakukan pengembangan dengan menggunakan 3 jenis model YOLOv5 yang berbeda. Model kedua, yaitu Yolov5l-HLB2 mendapat akurasi micro F1-score tertinggi yaitu 85,19%.
3	Fatma M Talaat & Hanaa ZainEldin	An improved fire detection approach based on YOLO-v8 for smart cities	2023	Peningkatan pendekslan daerah kebakaran dengan menggunakan metode YOLOv8. Dataset dibagi menjadi 21216 untuk data latih dan 5304 untuk data uji dan menghasilkan nilai akurasi 97.1% untuk semua kelas.
4	Rifqi Hakim Ariesdianto, Zilvanhisna Emka Fitri, Abdul Madjid, Arizal Mujibtamala Nanda Imron	Identifikasi Penyakit Daun Jeruk Siam Menggunakan K-Nearest Neighbor	2021	Identifikasi penyakit cancer dan ulat peliang pada jeruk menggunakan KNN. Dari 180 data, dilakukan modifikasi pada ruang warna RGB dan pengujian pada setiap ruang warna. Ruang warna GB (Green Blue) didapati sebagai

Lanjutan **Tabel 2.1** Penelitian terdahulu

No	Peneliti	Judul Penelitian	Tahun	Keterangan
5	Friska Rahayu Lestari, Ika Purwanti Ningrum Purnama, Adha Mashur Sajiah, LM.Bahtiar Aksara	Identifikasi Penyakit Tanaman Jeruk Siam Menggunakan Metode M-SVM	2019	ruang warna terbaik lalu dilakukan ekstraksi fitur tekstur GLCM menghasilkan akurasi identifikasi sebesar 70%.
6	Yang, Tingting, Suyin Zhou, Aijun Xu, Junhua Ye, dan Jianxin Yin	An approach for plant leaf image segmentation based on YOLOv8 and the improved DEEPLABV3+	2023	Identifikasi penyakit kanker daun jeruk, CVPD, dan ulat peliang menggunakan metode M-SVM. Pengujian dilakukan sebanyak 5 kali dengan kombinasi data latih dan data uji yang berbeda-beda. Pengujian keempat menjadi pengujian dengan hasil paling baik menggunakan citra latih sebanyak 40 citra pada setiap jenis penyakit dan menghasilkan akurasi sebesar 86,67%. Penelitian yang bertujuan untuk melakukan segmentasi daun menggunakan penggabungan YOLOv8 dan DEEPLABV3+ tersebut mencapai nilai rata-rata mIoU sebesar 90,8% untuk segmentasi daun pada dataset yang ada.

Lanjutan **Tabel 2.1** Penelitian terdahulu

No	Peneliti	Judul Penelitian	Tahun	Keterangan
7	Hanh Dang-Ngoc, Trang NM Cao, dan Chau Dang-Nguyen	Citrus leaf disease detection and classification using hierarchical support vector machine	2021	Pendeteksian penyakit jeruk melalui citra daun menggunakan Hierarchical Support Vector Machine. Penyakit yang dideteksi adalah empat jenis penyakit tanaman jeruk. Menghasilkan tingkat deteksi daun yang terinfeksi sebesar 92,5%, dan tingkat akurasi yang tinggi sebesar 91,76%.

2.15 Perbedaan Penelitian Terdahulu

Penelitian yang dilakukan penulis memiliki beberapa perbedaan dengan penelitian-penelitian sebelumnya. Penelitian yang dilakukan oleh (Luo, et al., 2023) adalah sebuah penelitian yang melakukan pendekstian gejala penyakit pada daun jeruk menggunakan YOLOv8. Perbedaannya dengan penelitian yang dilakukan oleh penulis adalah dari segi jenis penyakit dan implementasi. Penulis meneliti tiga kategori penyakit yaitu cancer, lumut, dan karat daun serta penulis mengimplementasikan model YOLOv8 pada sistem *mobile android*. Penulis juga mendekripsi objek daun secara keseluruhan dan mengidentifikasi penyakitnya, sementara penelitian yang dilakukan oleh (Luo, et al., 2023) mendekripsi gejala penyakit daun dimana sistem yang dibuat akan mengotakai gejala penyakit bukan daun secara keseluruhan.

Untuk penelitian yang dilakukan oleh (Talaat & ZainEldin, 2023) dan (Yang, et al., 2023) adalah penelitian-penelitian yang sama-sama menggunakan metode YOLOv8 tetapi diterapkan pada bidang dan objek yang berbeda dengan penulis. (Talaat & ZainEldin, 2023) meneliti tentang implementasi YOLOv8 dalam pendekstian daerah kebakaran. Lalu (Yang, et al., 2023) mengimplementasikan YOLOv8 dalam segmentasi jenis-jenis daun tanaman yang ada di daerah tertentu.

Berbeda dengan kedua penelitian tersebut, penulis meneliti mengenai pendektsian penyakit jeruk secara *real-time* melalui citra daun.

Sementara untuk penelitian lainnya memiliki perbedaan dalam hal metode yang digunakan dengan penulis namun memiliki persamaan dalam hal objek penelitian yaitu penyakit tanaman jeruk. Penulis menggunakan YOLOv8 dalam pendektsian penyakit cancer, lumut, dan karat daun jeruk. Untuk (Qiu, et al., 2022), mereka menggunakan YOLOv5 dalam pendektsian satu penyakit tanaman jeruk, yaitu Huanglongbing. (Ariesdianto , et al., 2021) meneliti tentang identifikasi penyakit cancer dan ulat peliang pada daun jeruk menggunakan algoritma *K-Nearest Neighbors*. Lalu (Lestari, et al., 2019) menggunakan metode M-SVM dalam mengidentifikasi penyakit cancer, CVPD, dan ulat peliang. Dan terakhir, (Dang-Ngoc, et al., 2021) menggunakan Hierarchical Support Vector Machine untuk mendekksi penyakit jeruk pada empat jenis penyakit jeruk, yaitu cancer, *sooty mold*, *greening*, dan *leaf-miner*.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Data yang digunakan

Data yang digunakan pada penelitian ini adalah citra yang didapat dengan cara pengambilan gambar secara langsung dari kebun jeruk di daerah Kabupaten Dairi, Sumatera Utara menggunakan kamera *smartphone Realme C21Y*. Dari pengambilan gambar tersebut, didapat total 450 gambar yang dibagi menjadi 150 gambar tiap kategori penyakit. Gambar 3.1 menunjukkan contoh data yang digunakan untuk melatih model yang akan dibuat :



Gambar 3.1 Contoh citra penyakit jeruk canker, karat daun, dan lumut daun

Untuk meningkatkan variasi data yang tersedia untuk pelatihan model dan membantu mencegah *overfitting* serta meningkatkan kemampuan model untuk menggeneralisasi dengan baik pada data baru, maka dilakukan augmentasi pada data tersebut. Augmentasi dilakukan dengan menggunakan platform Roboflow. Jenis augmentasi yang dilakukan kali ini adalah *flip vertical* dan *rotation -45 and 45*.

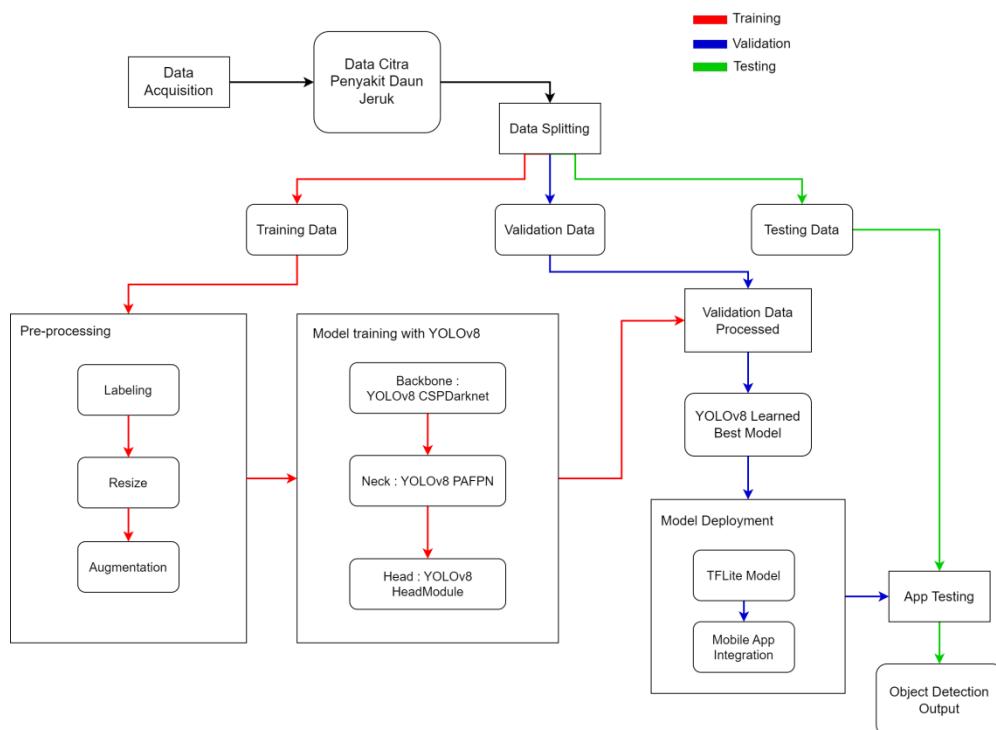
3.2 Analisis Sistem

Untuk memungkinkan perancangan sistem yang efektif dan efisien, maka perlu dilakukan analisis sistem. Analisis sistem ini mencakup penjelasan tahap-tahap perancangan sistem dari awal hingga sistem berhasil dibuat.

Langkah pertama yang dilakukan adalah pengumpulan data citra penyakit daun jeruk yang terbagi atas tiga kategori penyakit, yaitu *canker*, karat daun, dan lumut daun. Data didapatkan dengan menggunakan kamera *smartphone*. Setelah data

didapatkan, maka tahap selanjutnya adalah *pre-processing* data. Dimulai dengan proses *labeling*, pada setiap data dilakukan anotasi dengan cara memberikan *bounding box* pada bagian gambar yang terkena penyakit. Tahap selanjutnya adalah *resizing*, yaitu mengubah ukuran gambar agar sesuai dengan format yang diharapkan model sehingga proses pelatihan data dapat dilakukan dengan lebih lancar. Untuk penelitian ini, ukuran gambar diubah menjadi 640x640. Lalu data yang sudah di-*resize* akan di-augmentasi.

Augmentasi yang diterapkan adalah augmentasi jenis *flip* dan *rotate*. Hasil augmentasi data tersebut lalu digunakan untuk melatih model YOLOv8 pada tahap terakhir. Tahap terakhir adalah pendekslsian objek menggunakan model yang sudah terlatih dan sudah diimplementasikan pada perangkat android. Hasil yang diharapkan pada penelitian ini adalah sebuah aplikasi yang dapat mendekksi penyakit yang dialami pada tanaman jeruk secara *real time*. Arsitektur umum untuk keseluruhan tahapan penelitian ini dapat dilihat pada gambar 3.2 berikut :



Gambar 3.2 Arsitektur Umum

Berikut adalah penjelasan atas arsitektur umum pada gambar 3.2 :

3.2.1 *Image Acquisition*

Secara umum, tahap *image acquisition* adalah proses pengambilan atau perekaman citra dari berbagai sumber, seperti kamera digital, scanner, atau perangkat lainnya. Pada penelitian ini, *image acquisition* dilakukan dengan menggunakan kamera *smartphone* 3.8 MP dengan hasil gambar berekstensi ‘.jpg’. Citra yang dipotret adalah citra daun-daun jeruk yang terkena penyakit *canker*, karat daun, dan lumut daun dengan latar belakang perkebunan jeruk di daerah Kabupaten Dairi.

3.2.2 *Pre-processing*

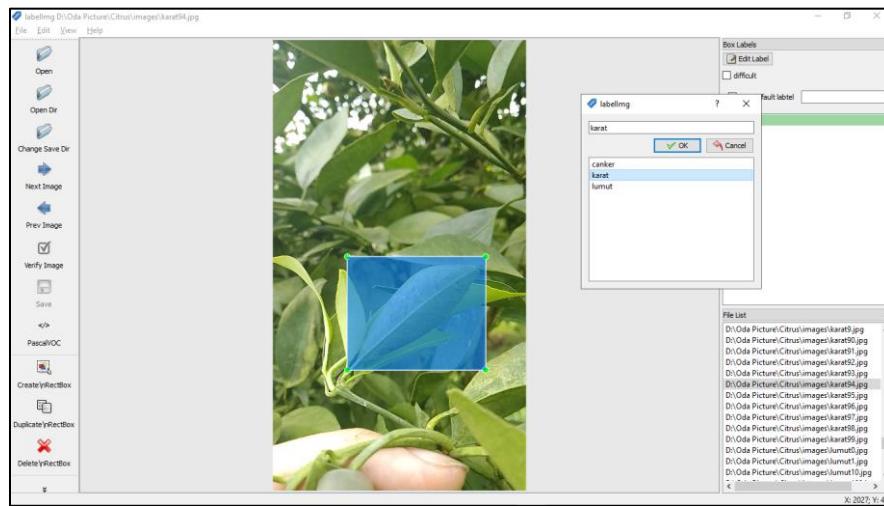
Tahap *pre-processing* adalah proses persiapan dan penyempurnaan data sebelum data tersebut diproses lebih lanjut. Tujuan utama dari *pre-processing* adalah untuk mengubah data mentah menjadi bentuk yang lebih mudah diolah dan lebih cocok untuk analisis atau penggunaan tertentu. *Pre-processing* dapat melibatkan berbagai teknik tergantung pada jenis data dan tujuan pengolahan. Teknik yang digunakan pada penelitian ini adalah sebagai berikut :

a) *Labeling*

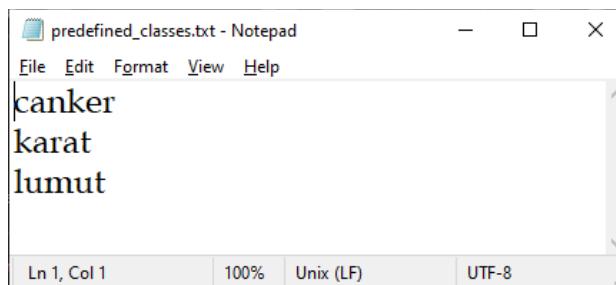
Labeling data adalah proses memberikan label atau kategori kepada data. Pada penelitian ini, pelabelan data dilakukan menggunakan aplikasi *open source* yaitu ‘LabelImg’. LabelImg adalah alat anotasi gambar grafis. Anotasi umumnya disimpan sebagai file XML dalam format PASCAL VOC, format yang digunakan oleh *ImageNet*. Selain itu juga mendukung format YOLO dan CreateML. Pelabelan dilakukan dengan cara mengotaki (*bounding box*) citra daun pada setiap data dengan label penyakit yang ada. Pemberian bounding box dilakukan pada area yang memiliki informasi penting yang dapat melabeli objek tersebut sebagai salah satu kelas atau kategori penyakit daun.

Berdasarkan batasan masalah yang sudah ditentukan pada bab pertama, data latih yang sudah didapat akan dilabeli sesuai dengan kategori penyakit pada penelitian ini yaitu *canker*, karat daun, dan lumut daun. Gambar 3.4 menunjukkan bahwa ketiga kategori tersebut disimpan pada *file predefined_classes.txt* yang dibuat agar aplikasi LabelImg mengetahui kelas apa saja yang dapat dipilih pada proses pelabelan. Pada penelitian ini terdapat 450 gambar yang harus dilabeli agar data uji dapat digunakan

pada proses selanjutnya. Pelabelan menggunakan aplikasi LabelImg dapat dilihat pada gambar 3.3 berikut :



Gambar 3.3 Contoh pelabelan data pada citra karat daun menggunakan LabelImg



Gambar 3.4 File predefined_classes.txt

Setelah *labeling* data selesai dilakukan, maka file ‘.xml’ hasil pelabelan data tersimpan pada direktori yang sama dengan data citra yang dilabeli. File ‘.xml’ tersebut berisi informasi mengenai pelabelan yang telah dilakukan pada objek, yaitu nama kelas dan nilai koordinat $xmin$, $ymin$, $xmax$, dan $ymax$ yang dapat dilihat pada gambar 3.5. Dimana nilai $xmin$ dan $xmax$ adalah nilai koordinat *bounding box* secara horizontal sementara nilai $ymin$ dan $ymax$ adalah nilai koordinat secara vertikal. Contoh nilai-nilai tersebut dapat dilihat pada gambar 3.6 dengan menggunakan karat94 sebagai percontohan.

```

<annotation>
    <folder>images</folder>
    <filename>karat94.jpg</filename>
    <path>D:\Oda Picture\Citrus\images\karat94.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1152</width>
        <height>2048</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>karat</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>339</xmin>
            <ymin>985</ymin>
            <xmax>968</xmax>
            <ymax>1500</ymax>
        </bndbox>
    </object>
</annotation>

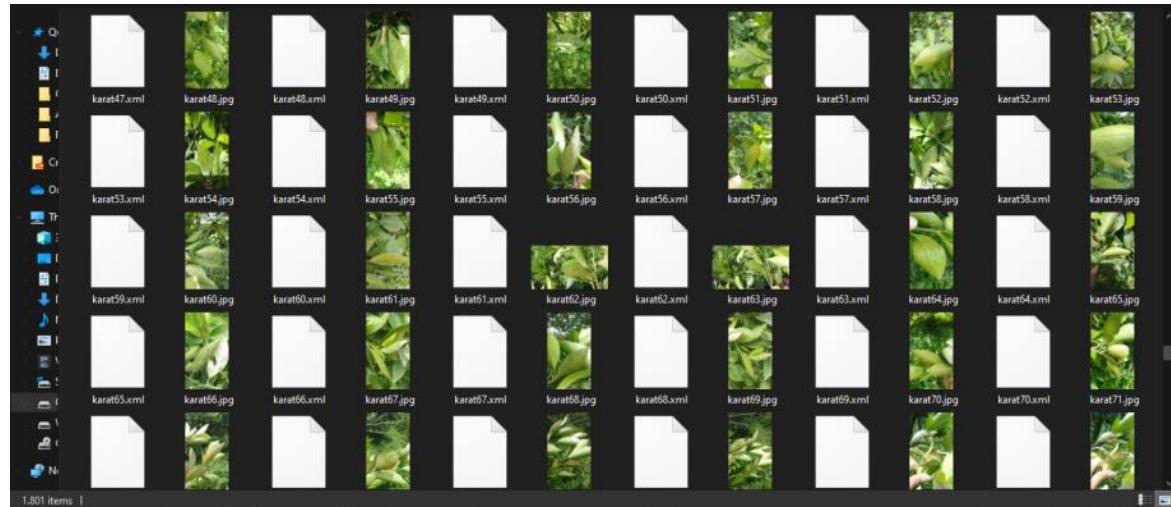
```

Gambar 3.5 Isi file karat94.xml sebagai hasil dari pelabelan



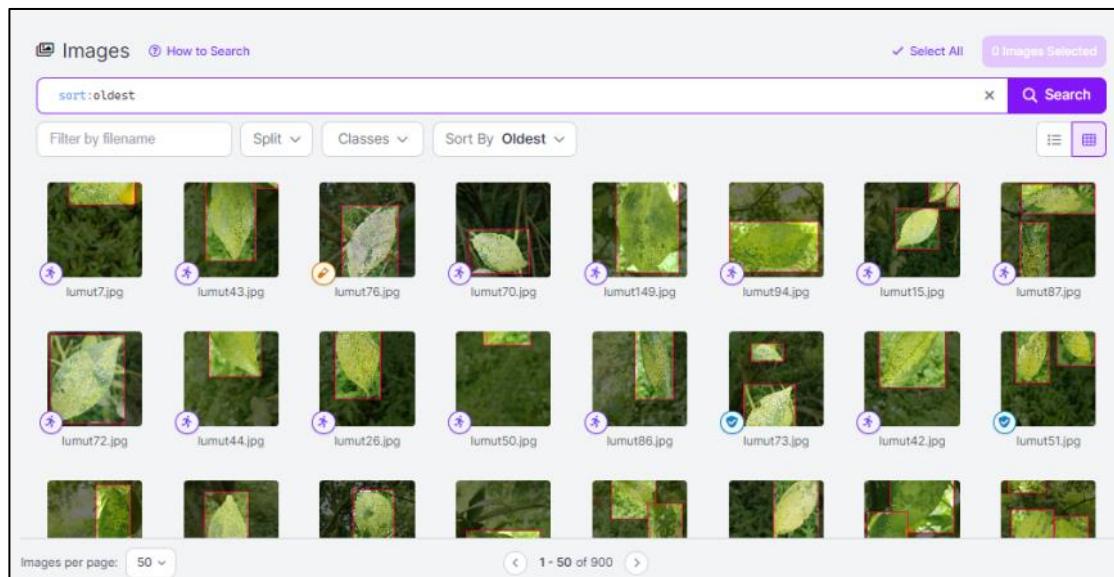
Gambar 3.6 Contoh nilai-nilai yang digunakan pada pelabelan citra karat94

Untuk dapat dilanjutkan ke tahap *pre-processing* selanjutnya, yang dimana dilakukan pada platform Roboflow, data perlu di-*import* ke dalam platform. Data yang di-*import* ke dalam platform terdiri atas *file* citra beserta *file labeling* yang berekstensi ‘.xml’ tersebut. Hasil pelabelan data tersimpan pada direktori yang sama dengan data citra yang dilabeli dapat dilihat pada gambar 3.7.



Gambar 3.7 File citra dan file label hasil pelabelan menggunakan LabelImg

Untuk hasil impor data dapat dilihat pada gambar 3.8.



Gambar 3.8 Hasil impor data pada platform *Roboflow*

b) Resize

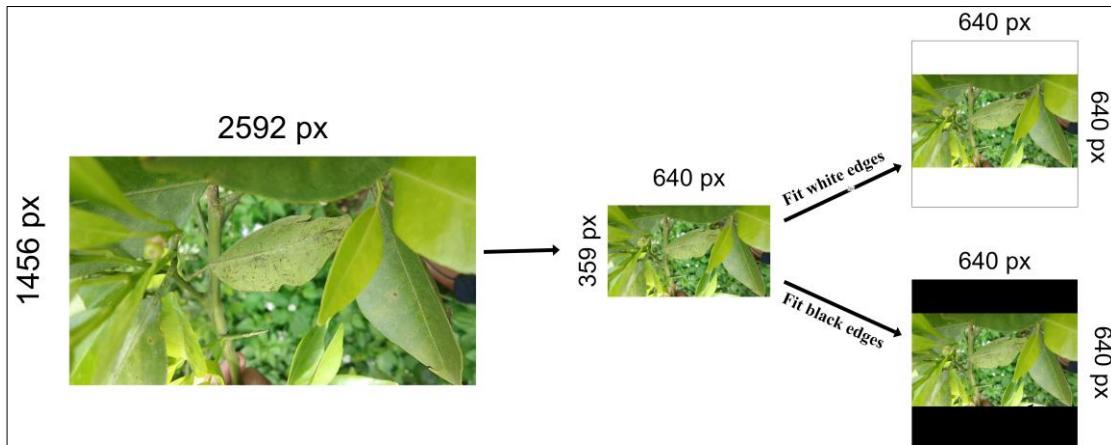
Resize pada *pre-processing* adalah proses mengubah ukuran dimensi gambar menjadi ukuran yang diinginkan sebelum data tersebut dimasukkan ke dalam model atau algoritma pengenalan objek. Pelaksanaan *resize* dilakukan agar sistem nantinya tidak perlu lagi melakukan *resize* mandiri yang dapat memperlambat proses *training*. Pada penelitian ini, data citra memiliki dua variasi ukuran dimensi, yaitu 1456x2592 untuk *portrait* dan 2592x1456 untuk *landscape*. Oleh karena itu, untuk mengurangi jumlah piksel yang perlu diproses oleh model, mengurangi beban komputasi, dan

mempercepat waktu inferensi, maka perlu dilakukan proses *resize* gambar ke ukuran dimensi yang lebih kecil.

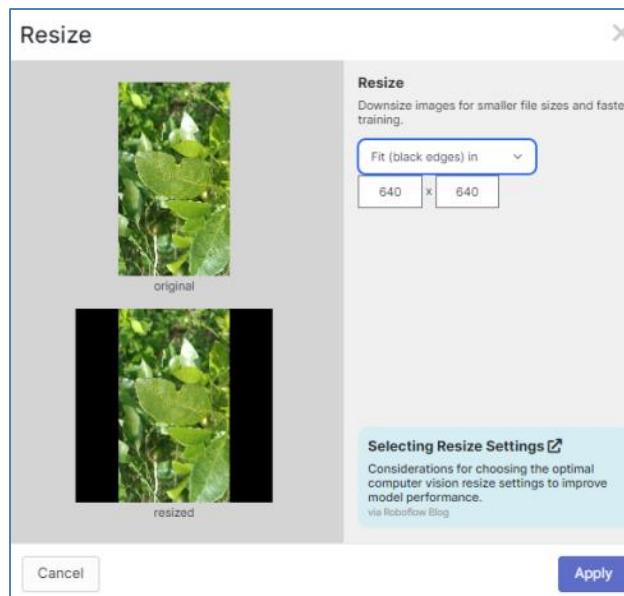
Proses *resize* gambar juga dilakukan pada platform Roboflow. Terdapat enam jenis proses resize yang dapat dilakukan pada platform tersebut, yaitu *stretch to*, *fill (with center crop) in*, *fit within*, *fit (reflect edges) in*, *fit (black edges) in*, dan *fit (white edges) in*. Secara singkat pilihan *stretch to*, *fill (with center crop) in*, *fit within*, dan *fit (reflect edges) in* akan memotong latar belakang citra sehingga dapat mengurangi informasi yang akan didapat oleh model nantinya sehingga penulis tidak memilih jenis proses *resize* tersebut. Pilihan yang tersisa adalah *fit (black edges) in* dan *fit (white edges) in* yang dimana keduanya adalah pilihan yang tidak terlalu berbeda.

Kedua jenis proses ini memiliki cara kerja yang hampir sama. Diambil contoh dimensi citra yang digunakan pada penelitian ini, yaitu dimensi 1456x2592 sebagai percontoh. Pada kasus tersebut, jika opsi *resize* yang digunakan adalah 640x640, maka dimensi yang memiliki ukuran dimensi lebih panjang, yaitu 2592 piksel, akan diperkecil menjadi 640 piksel. Lalu untuk ukuran yang lebih pendek, yaitu 1456 piksel, juga tetap akan diperkecil menjadi ukuran yang sebanding dengan 2592 piksel, yaitu sekitar 359 piksel. Agar dimensi citra menjadi 640x640, maka citra yang tadinya 359x640 akan disisipi area kosong berwarna hitam ataupun putih. Proses *resize* ini dapat dilihat pada gambar 3.9.

Proses *resize* yang digunakan pada penelitian ini adalah jenis *resize* dengan menambahkan area hitam (*Fit black edges*). Ukuran input gambar model default yang digunakan untuk melatih model YOLOv8 adalah 640. Dikarenakan metode yang digunakan untuk penelitian ini adalah YOLOv8, maka ukuran *resize* yang ditetapkan untuk penelitian ini adalah 640x640. Proses *resize* pada platform *Roboflow* dapat dilihat pada gambar 3.10.



Gambar 3.9 Gambaran proses *resize fit (black edges) in* dan *fit (white edges) in*



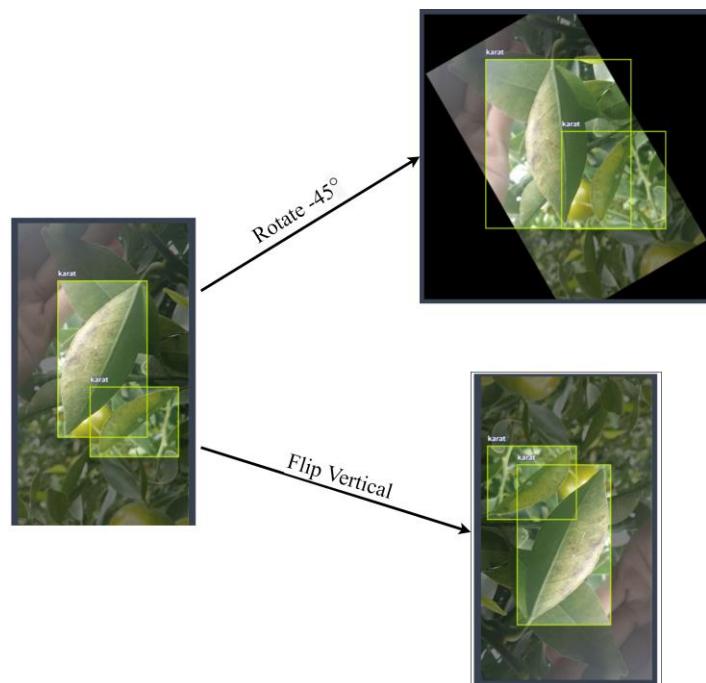
Gambar 3.10 *Resize* citra pada platform *Roboflow*

c) Augmentation

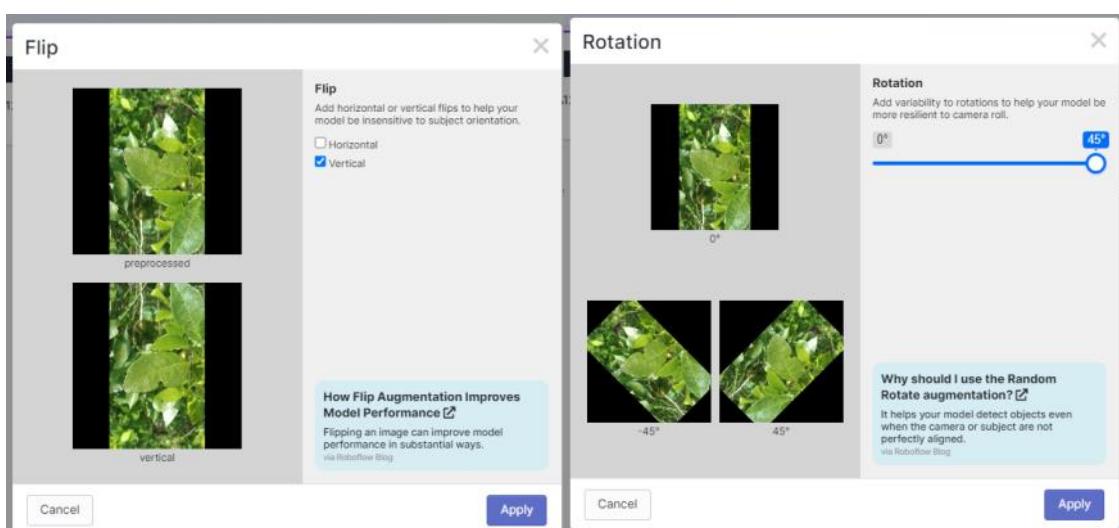
Augmentation atau proses augmentasi data adalah teknik yang digunakan untuk meningkatkan jumlah dan variasi data latih dengan membuat modifikasi terkontrol pada data yang ada. Tujuan dari augmentasi data adalah untuk memperluas data latih dengan cara yang realistik sehingga model yang dilatih menjadi lebih umum, toleran terhadap variasi, dan lebih baik dalam menggeneralisasi pada data baru yang belum pernah dilihat sebelumnya.

Terdapat banyak teknik augmentasi data yang dapat digunakan. Untuk penelitian ini, teknik augmentasi yang dilakukan adalah *flip* dan *rotate*. Augmentasi *flip* adalah teknik yang dilakukan dengan cara membalik atau memutar gambar secara

horizontal atau vertikal. Sementara itu, augmentasi *rotate* adalah teknik augmentasi data yang melibatkan memutar gambar dalam berbagai sudut. Sudut augmentasi *rotate* yang digunakan pada penelitian ini adalah 45° dan -45° . Dapat dilihat pada gambar 3.11 contoh pelaksanaan proses augmentasi *rotate* dan *flip*. *Bounding box* pada kedua proses hasil augmentasi tersebut tetap disesuaikan dengan objek pada citra. Sehingga meskipun dilakukan augmentasi, pelabelan citra juga tidak terganggu. Untuk pengaplikasian augmentasi pada *dataset* dapat dilihat pada gambar 3.12.



Gambar 3.11 Contoh proses augmentasi *rotate* dan *flip* pada salah satu citra



Gambar 3.12 Augmentasi *flip* dan *rotation* yang dilakukan pada platform *Roboflow*

Proses augmentasi dapat dilihat pada gambar 3.13 di bawah ini dalam bentuk *pseudocode*.

```

BEGIN
IMPORT necessary libraries:
    os for directory operations
    imageio.v2 as imageio for image reading and writing
    imgaug.augmenters as iaa for image augmentation
    numpy as np for numerical operations
    Image from PIL for image processing

SET input_folder to "Image_Folder"
SET output_folder to "Augmented_Folder"

IF output_folder does not exist THEN
    CREATE output_folder

DEFINE augmentation_pipeline AS iaa.Sequential([
    iaa.Fliplr(0.5),
    iaa.Affine(rotate=(180))
]

END

```

Gambar 3.13 *Pseudocode python* untuk augmentasi data

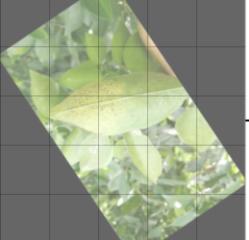
3.2.3 Image Classification

Pada penelitian ini, algoritma pendekripsi objek yang diterapkan adalah YOLOv8. YOLOv8 merupakan penyempurnaan dari YOLO versi sebelumnya, terutama YOLOv5. YOLO bekerja dengan melakukan deteksi objek secara '*single forward pass*', yang artinya proses pengolahan gambar dilakukan hanya sekali tanpa perlu iterasi atau pengulangan. Cara pengerjaan tersebutlah yang membuat YOLO bekerja secara cepat, mengurangi biaya komputasi, dan sesuai dengan pengaplikasian *real-time*. Seperti yang terlihat pada gambar 2.10, arsitektur YOLO terbagi menjadi tiga bagian besar dan memiliki peran masing-masing. Tiga bagian utama tersebut adalah *backbone*, *neck*, dan *head*.

a) *Backbone*

Backbone adalah bagian yang bertugas sebagai *feature extractor* pada YOLO. Bagian *backbone* terdiri atas serangkaian lapisan konvolusional yang bertugas sebagai pengekstrak fitur yang relevan dari gambar masukan. Model yang digunakan oleh YOLOv8 pada bagian *backbone* adalah model CSPDarknet53. Bagian ini terdiri atas sejumlah blok *convolution* yang bertugas untuk mengekstrak fitur dari berbagai resolusi masukan. Proses pengekstrakan fitur pada layer konvolusi yang ada pada blok

convolution module dilakukan dengan menggunakan *kernel*. *Kernel* akan bergerak ke seluruh bagian gambar dan melakukan proses pada gambar sehingga menghasilkan *feature map*. Gambar yang digunakan pada penelitian ini adalah gambar dengan dimensi 640x640x3, dengan kata lain masing-masing gambar memiliki nilai *red*, *green*, dan *blue*. Contoh nilai warna pada gambar yang akan digunakan dapat dilihat pada gambar 3.14.



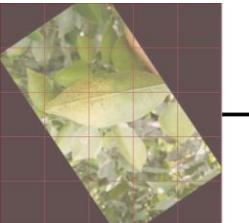
640x640x3 Image					Red	Green					Blue				
25	74	54	7	0	25	87	75	13	0	24	80	63	11	0	
69	131	91	37	0	69	163	133	65	0	71	155	115	52	0	
16	159	88	64	6	17	184	138	120	17	18	187	121	98	13	
0	74	81	80	39	0	102	148	140	72	0	93	133	118	54	
0	5	51	49	15	0	7	83	72	29	0	6	73	63	27	

Gambar 3.14 Contoh nilai RGB pada gambar

Dapat dilihat pada gambar 3.14 tersebut, sebagai contoh, sebuah gambar dibagi menjadi 5x5 grid piksel. Sebuah gambar berwarna pastinya memiliki tiga dimensi warna sehingga sebuah gambar memiliki nilai tiga nilai warna yang berbeda. Nilai yang didapat tersebut dapat diubah ke dalam matriks sehingga nantinya dapat dilakukan *feature extraction* menggunakan layar konvolusi yang ada pada YOLO.

- *Feature Extraction*

Untuk dapat mendapatkan *feature map* yang sesuai, maka *kernel* yang digunakan tentunya perlu disesuaikan. Dikarenakan gambar berwarna yang ingin diproses memiliki dimensi 640x640x3, maka *kernel* yang digunakan adalah *kernel* berukuran 3x3x3, yaitu *kernel* 3x3 yang memroses masing-masing *channel* RGB pada gambar. Gambar 3.15 merupakan contoh perhitungan *feature extraction* yang dilakukan pada nilai *red* pada RGB:



$$\begin{bmatrix} 25 & 74 & 54 & 7 & 0 \\ 69 & 131 & 91 & 37 & 0 \\ 16 & 159 & 88 & 64 & 6 \\ 0 & 74 & 81 & 80 & 39 \\ 0 & 5 & 51 & 49 & 15 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} A1/262 & A2/145 & A3/23 \\ B1/486 & B2/45 & B3/109 \\ C1/125 & C2/112 & C3/167 \end{bmatrix}$$

Gambar 3.15 Perhitungan *feature extraction* pada layar *red* pada gambar

Berikut rincian perhitungan yang dilakukan pada ekstraksi fitur tersebut :

$$\begin{aligned} A1 &= (0x25)+(-1x74)+(0x53)+(-1x69x)+(5x131)+(-1x91)+(0x16)+(-1x159)+(0x88) \\ &= 262 \end{aligned}$$

$$\begin{aligned} A2 &= (0x74)+(-1x54)+(0x7)+(-1x131)+(5x91)+(-1x37)+(0x159)+(-1x88)+(0x64) \\ &= 145 \end{aligned}$$

$$\begin{aligned} A3 &= (0x54)+(-1x7)+(0x0)+(-1x91)+(5x91)+(-1x37)+(0x159)+(-1x88)+(0x64) \\ &= 23 \end{aligned}$$

$$\begin{aligned} B1 &= (0x69)+(-1x131)+(0x91)+(-1x16)+(5x159)+(-1x88)+(0x0)+(-1x74)+(0x81) \\ &= 486 \end{aligned}$$

$$\begin{aligned} B2 &= (0x131)+(-1x91)+(0x37)+(-1x159)+(5x88)+(-1x64)+(0x74)+(-1x81)+(0x80) \\ &= 45 \end{aligned}$$

$$\begin{aligned} B3 &= (0x91)+(-1x37)+(0x0)+(-1x88)+(5x64)+(-1x6)+(0x81)+(-1x80)+(0x39) \\ &= 109 \end{aligned}$$

$$\begin{aligned} C1 &= (0x16)+(-1x159)+(0x88)+(-1x0)+(5x74)+(-1x81)+(0x0)+(-1x5)+(0x51) \\ &= 125 \end{aligned}$$

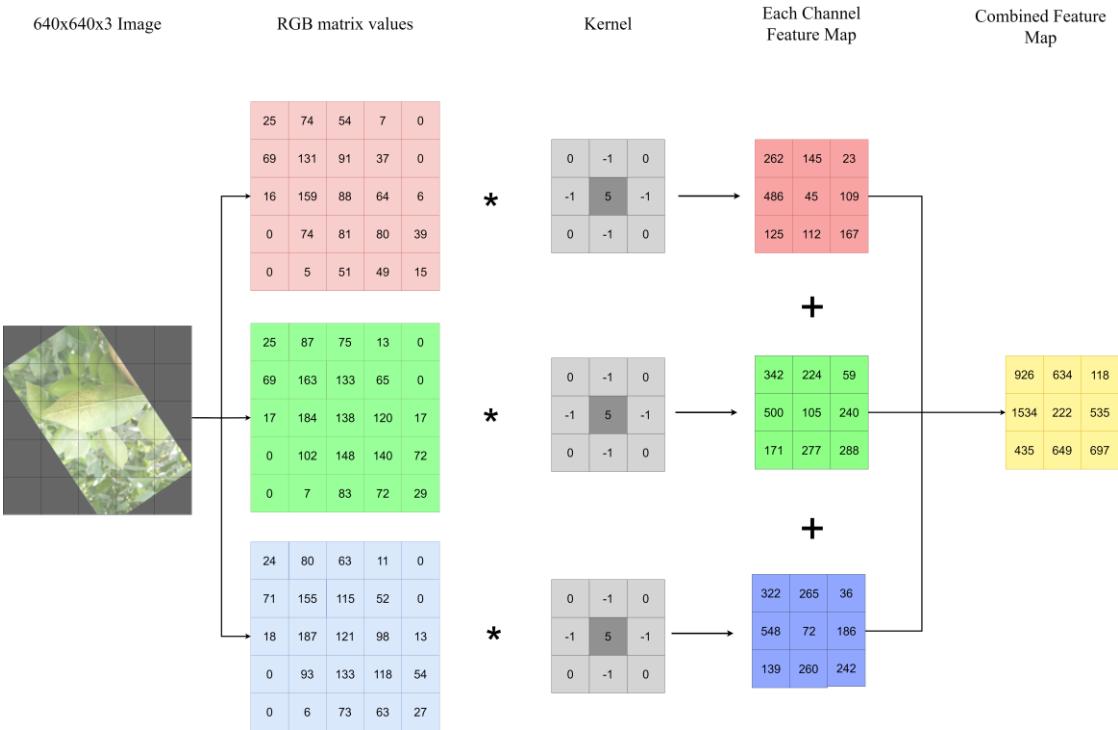
$$\begin{aligned} C2 &= (0x159)+(-1x88)+(0x64)+(-1x74)+(5x81)+(-1x80)+(0x5)+(-1x51)+(0x49) \\ &= 112 \end{aligned}$$

$$\begin{aligned} C3 &= (0x88)+(-1x64)+(0x6)+(-1x81)+(5x80)+(-1x39)+(0x51)+(-1x49)+(0x15) \\ &= 167 \end{aligned}$$

Setelah ketiga dimensi warna diproses dengan kernel pada lapisan konvolusi, maka akan dihasilkan *feature map* untuk lapisan merah, hijau, dan biru. Untuk mendapatkan nilai matriks akhir, maka dilakukan penjumlahan matriks dari setiap matriks yang didapat tiap lapisan warna seperti berikut :

$$\begin{aligned} &\begin{bmatrix} 262 & 145 & 23 \\ 486 & 45 & 109 \\ 167 & 112 & 167 \end{bmatrix} + \begin{bmatrix} 342 & 224 & 59 \\ 500 & 105 & 240 \\ 171 & 277 & 288 \end{bmatrix} + \begin{bmatrix} 322 & 265 & 36 \\ 548 & 72 & 186 \\ 139 & 260 & 242 \end{bmatrix} \\ &= \begin{bmatrix} 926 & 634 & 118 \\ 1534 & 222 & 535 \\ 455 & 649 & 697 \end{bmatrix} \end{aligned}$$

Perhitungan tersebut juga dapat ditunjukkan pada gambar 3.16.



Gambar 3.16 Ekstraksi fitur setiap dimensi warna pada *convolution layer*

Setelah *feature map* didapatkan, maka proses di dalam blok konvolusi berlanjut ke lapisan *batch normalization*.

- *2D Batch Normalization*

Lapisan *batch normalization* adalah lapisan yang berfungsi untuk menormalkan *output* dari layer sebelumnya (atau *input* dari layer saat ini) sebelum memasukkannya ke layer berikutnya. Untuk memroses 3x3x1 *feature map* yang sudah didapat dari *convolution layer*, dapat dimulai dengan cara berikut:

- *Feature map* yang telah didapat :

$$\begin{bmatrix} 926 & 634 & 118 \\ 1534 & 222 & 535 \\ 455 & 649 & 697 \end{bmatrix}$$

- Hitung nilai *Mean* (μ) dengan menggunakan rumus (2.1)

$$\begin{aligned} \mu &= \frac{1}{9} (926 + 634 + 118 + 1534 + 222 + 535 + 455 + 649 + 697) \\ &= \frac{5770}{9} \approx 641 \end{aligned}$$

- Hitung *Variance* (σ^2) dengan menggunakan rumus (2.2):

$$\begin{aligned}\sigma^2 &= \frac{1}{9}((926 - 641)^2 + (634 - 641)^2 + (118 - 641)^2 + (1534 \\ &\quad - 641)^2 + (222 - 641)^2 + (535 - 641)^2 + (455 \\ &\quad - 641)^2 + (649 - 641)^2 + (697 - 641)^2) = \frac{1376845}{9} \\ &= 152983\end{aligned}$$

- Normalisasikan setiap nilai. Gunakan rumus (2.3) pada setiap nilai pada *feature map*. Ambil satu contoh nilai :

$$x_{11} = \frac{926 - 641}{\sqrt{152983 + 0,001}} = 0,72834$$

- Proses *Scale(γ) and shift(β)*. Untuk kasus yang sederhana, gunakan nilai $\gamma=1$ and $\beta=0$ lalu implementasikan pada rumus (2.4), sehingga nilai akhir tetap menjadi :

$$y_{ij} = 1 * x_{ij} + 0 = x_{ij}$$

Dari perhitungan tersebut, maka dapat diaplikasikan kepada semua nilai *feature map* sehingga menghasilkan nilai matrix berikut :

$$\begin{bmatrix} 0,728 & -0,017 & -1,336 \\ 2,282 & -1,071 & -0,271 \\ -0,475 & 0,02 & 0,143 \end{bmatrix}$$

- *Sigmoid Linear Unit (SiLU)*

Setelah melewati *batch normalization*, maka *feature map* akan lanjut diproses dengan menggunakan fungsi aktivasi SiLU yang ada pada blok konvolusi sesuai dengan arsitektur YOLOv8 pada gambar 2.10. Dari 3x3 *feature map* yang sudah dinormalisasi sebelumnya, maka langkah-langkah yang dilakukan pada fungsi aktivasi ini adalah :

- Hitung *Sigmoid function* untuk tiap nilai pada *feature map* menggunakan rumus (2.5). Contoh dengan menggunakan salah satu nilai pada matrix:

$$\sigma(0,728) = \frac{1}{1 + e^{-0,728}} = 0,325559204$$

Sehingga nilai *Sigmoid function* tiap *feature map* yang didapatkan menjadi :

$$\begin{bmatrix} 0,326 & 0,504 & 0,792 \\ 0,093 & 0,745 & 0,567 \\ 0,617 & 0,495 & 0,464 \end{bmatrix}$$

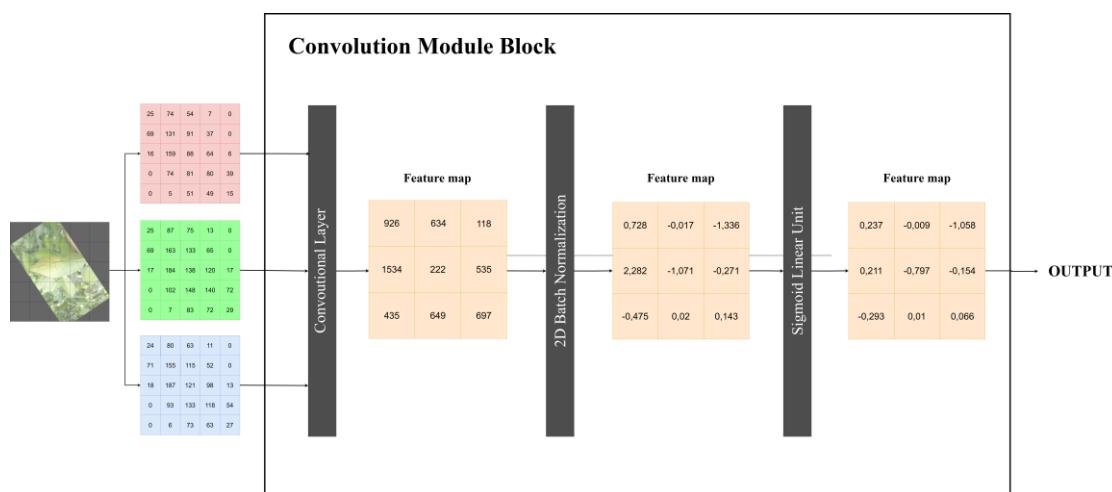
- Hitung *SiLU Activation* untuk tiap nilai pada *feature map* dengan rumus (2.6). Contoh dengan menggunakan salah satu nilai pada matrix:

$$SiLU(0,728) = 0,728 * \sigma(0,728)$$

Sehingga nilai *feature map* yang didapatkan menjadi :

$$\begin{bmatrix} 0,237 & -0,009 & -1,058 \\ 0,211 & -0,797 & -0,154 \\ -0,293 & 0,01 & 0,066 \end{bmatrix}$$

Dari kedua proses tersebut, *batch normalization* menstabilkan dan mempercepat pelatihan, sementara SiLU memberikan transformasi non-linier yang mulus untuk meningkatkan kekuatan representasi jaringan. Kedua teknik tersebut memainkan peran penting dalam pelatihan model, namun keduanya memiliki tujuan yang berbeda dan memengaruhi peta fitur dengan cara yang berbeda. Proses yang dilakukan pada blok konvolusi ini dapat digambarkan pada gambar 3.17 berikut :



Gambar 3.17 Proses pada blok konvolusi

Pada YOLOv8, setelah proses pada SiLU selesai, maka proses pada sebuah blok konvolusi juga selesai dan menghasilkan sebuah keluaran *feature map*. Keluaran tersebut akan diproses pada blok berikutnya seperti blok C2f dan blok konvolusi berikutnya sesuai dengan alur arsitektur YOLO. Blok C2f terdiri atas blok konvolusi dan juga *Darknet Bottleneck*, yang dimana *bottleneck* juga berisi blok konvolusi. Lalu di ujung *backbone* terdapat blok SPPF (*Spatial Pyramid Pooling-Fast*) yang dimana menghubungkan bagian *backbone* kepada blok-blok yang ada di bagian *neck*. Dalam arsitektur YOLO, modul SPPF biasanya dianggap sebagai bagian dari *neck* dan bukan sebagai *backbone*.

b) Neck

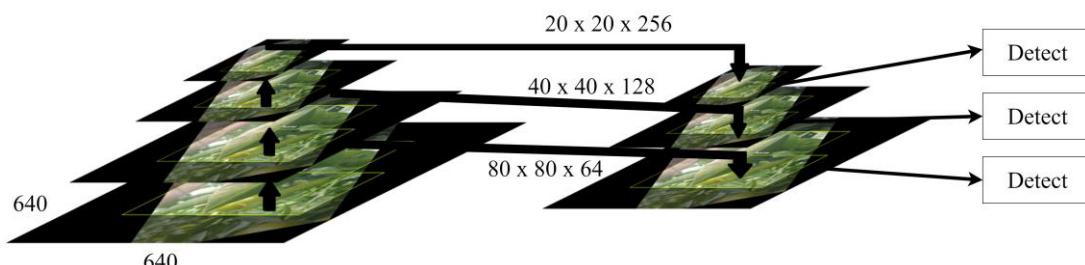
Bagian *neck* merupakan komponen perantara yang menghubungkan *backbone* dengan *head*. Di bagian ini terdapat modul SPPF, beberapa blok C2f, dan blok konvolusi yang digunakan untuk mengekstrak fitur yang didapat dari *backbone* lalu disampaikan kepada bagian *head* sehingga dapat dilakukan pendekripsi. Modul SPPF khususnya digunakan untuk meningkatkan kemampuan jaringan dalam menangani objek pada skala yang berbeda.

SPPF (*Spatial Pyramid Pooling-Fast*) mengumpulkan fitur-fitur pada berbagai skala spasial, memungkinkan jaringan menangkap informasi konteks pada resolusi berbeda. Dalam hal ini masukan dengan dimensi 640x640x3 akan diproses terlebih dahulu pada blok konvolusi dan C2f hingga dimensi *feature map* menjadi 80x80x256w. Nilai ‘w’ yang dimaksud pada dimensi tersebut merujuk kepada nilai ‘*widen_factor*’ yang berbeda-beda nilainya disesuaikan pada jenis model YOLOv8 yang digunakan. Pada penelitian ini, model YOLOv8 yang digunakan adalah YOLOv8n dengan ‘n’ adalah singkatan dari nano. Terdapat beberapa jenis model YOLO yang dikembangkan pada versi ini dan dapat dilihat pada tabel 3.1.

Tabel 3.1 Jenis-jenis model yang ada pada YOLOv8

model	d (deepen_factor)	w (widen_factor)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

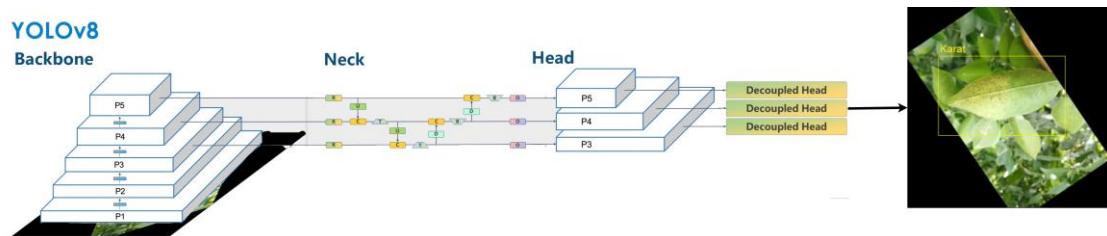
Model ini memiliki nilai ‘w = 0,25’ sehingga *feature map* yang didapat adalah 80x80x64. Selain skala tersebut, nantinya akan didapatkan skala lain, yaitu 40x40x128 dan 20x20x256. Sehingga dapat dihasilkan piramida fitur yang dapat dilihat pada gambar 3.18.



Gambar 3.18 Piramida fitur yang dihasilkan SPPF

c) Head

Head adalah bagian terakhir dari struktur YOLO dan bertanggung jawab untuk membuat prediksi berdasarkan fitur yang disediakan oleh *backbone* dan yang diberikan oleh *neck*. Di bagian ini juga terdapat blok konvolusi yang dihubungkan dengan layar konvolusi dan layar prediksi berupa hasil koordinat *bounding box* dan *class probabilities*. Untuk menghasilkan *output* yang diharapkan, proses pendekripsi pada bagian ini dimulai dengan mendapatkan *feature map* dari tiga skala berbeda, yaitu 80x80x64, 40x40x128, dan 20x20x256. Lalu hasil yang didapat dari tiap skala akan menjadi masukan pada tiap blok deteksi yang mendekripsi citra yang berskala kecil, sedang, dan besar. Dari fitur-fitur yang telah dipelajari tersebut, maka detektor akan memberikan kotak pembatas pada objek yang dianggap memenuhi kriteria sebagai objek yang diharapkan dan juga memberikan jenis kelas yang bersangkutan.



Gambar 3.19 Proses YOLOv8 hingga pemberian *bounding box* dan *class*

3.2.4 Proses Training

Proses pelatihan atau *training* adalah langkah dimana sebuah model pembelajaran mesin, dalam hal ini model YOLO, mempelajari pola dari data pelatihan untuk membuat prediksi atau keputusan yang akurat pada data baru. Proses pelatihan model yang dilakukan pada penelitian ini yaitu dengan dibantu oleh platform *ultralytics* dan *google colab*.

Pada tahap ini, data yang akan di-*input* adalah data *training*. Data *training/latih* adalah data yang digunakan untuk melatih model sehingga model dapat mempelajari berbagai fitur yang didapatkan dari setiap gambar, seperti warna dan pola gambar. Data latih yang digunakan pada penelitian ini sebanyak 80% dari total data citra, yaitu 720 citra. Data tersebut didapatkan dari augmentasi 450 data sehingga menghasilkan 900 data dan dilakukan *splitting* data. Data *training* terdiri atas berbagai jenis citra yang berbeda dan sudah dilabeli ke dalam tiga kategori penyakit yang berbeda.

Semakin beragam ciri citra yang dipelajari sistem maka akan semakin bagus performa model yang dihasilkan. Model yang didapatkan dari proses pelatihan ini nantinya akan digunakan untuk memvalidasi keberhasilan model sehingga akan mendapat model terbaik, Proses *training* dapat dilihat pada gambar 3.20 berikut dalam bentuk *pseudocode*.

```
# Pseudocode for Training YOLO Model using Google Colab, Roboflow, and Ultralytics

# Step 1: Mount Google Drive
mount_google_drive()

# Step 2: Install Required Packages
install_package("roboflow")
install_package("ultralytics")

# Step 3: Import Required Libraries
import_library("ultralytics")
import_library("roboflow")

# Step 4: Initialize Roboflow and Download Dataset
initialize_roboflow(api_key="your_api_key")
project = get_project("workspace_name", "project_name")
version = get_project_version(project, version_number=1)
dataset = download_dataset(version, format="yolov8")

# Step 5: Train YOLO Model
train_yolo_model(model="best_float16.tflite", data=dataset.location + "/data.yaml", epochs=10, close_mosaic=0,
batch_size=16)
```

Gambar 3.20 Pseudocode proses training

3.2.5 Learned Model dan Proses Validasi

Setelah proses pelatihan model dengan menggunakan dataset dilakukan, maka akan dihasilkan sebuah *learned model* yang berupa file ber-ekstensi ‘.pt’. Learned model adalah hasil yang didapatkan dari pelatihan data dimana nantinya model ini akan digunakan oleh data validasi untuk menguji keakuratan pengklasifikasian pada proses validasi. Proses validasi adalah langkah-langkah yang diambil untuk memastikan bahwa hasil dari algoritma atau model yang digunakan memenuhi kriteria yang diharapkan dan dapat diandalkan. Proses ini dilakukan setelah model selesai dilatih.

Pada tahap validasi, data yang digunakan adalah data *validation*. Data *validation* adalah data yang digunakan untuk memvalidasi atau mengevaluasi model setelah proses pelatihan dilakukan. Proses validasi ini berfungsi untuk memastikan bahwa hasil model yang sudah dilatih memenuhi kriteria yang diharapkan dan merupakan model yang terbaik. Data validasi yang digunakan pada penelitian ini sebanyak 10% dari total data citra, yaitu 90 citra.

3.2.6 Tensorflow Lite Model

Model yang sudah dilatih akan menghasilkan sebuah *learned model* dalam bentuk file ber-ekstensi ‘.pt’. File tersebut belum dapat digunakan untuk pengimplementasian pada aplikasi *mobile*, sehingga perlu dilakukan transformasi ke dalam bentuk yang dapat diterima oleh aplikasi *mobile*. Dalam hal ini, file yang dapat digunakan adalah *TFLite Model*. Dengan menggunakan perintah YOLO, file ‘.pt’ yang sudah didapatkan dapat diekspor menjadi *TFLite Model*.

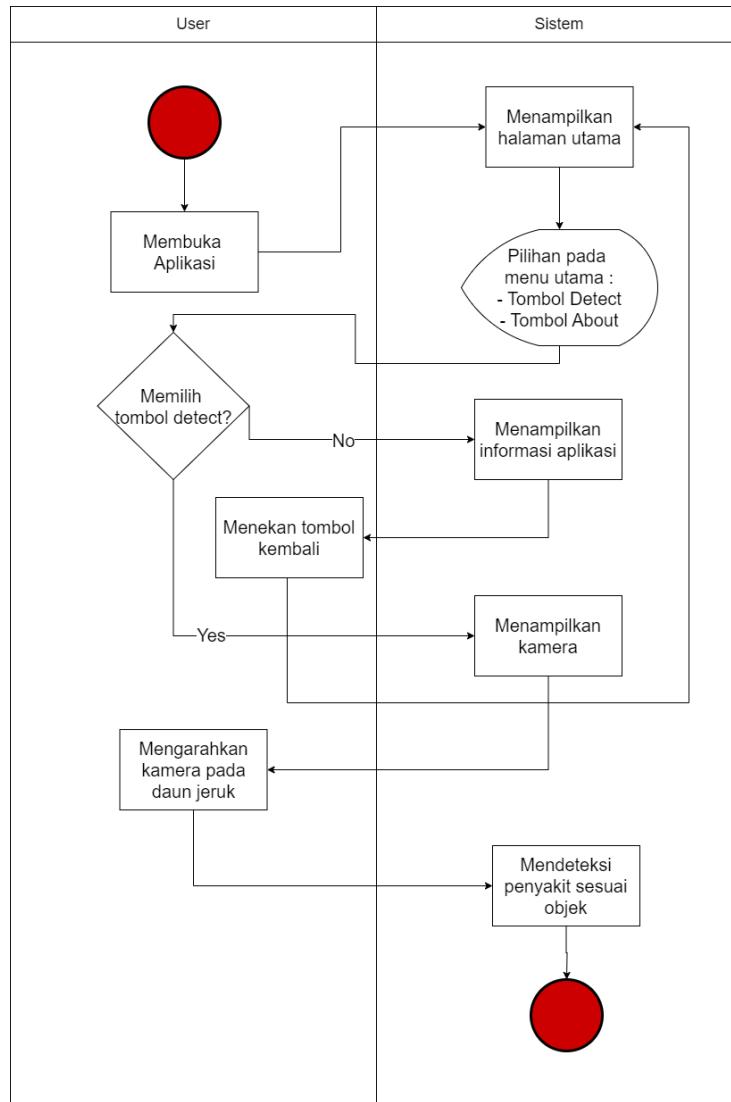
3.2.7 Proses Pengujian

Model TFLite yang telah dihasilkan akan diuji dalam proses pengujian (*testing*). Hal ini dilakukan untuk mengetahui kinerja model dalam mendeteksi objek dalam citra yang belum pernah dilihat. Pada tahap ini, data yang digunakan adalah data *testing*. Data *testing/uji* adalah data yang digunakan untuk pengujian model setelah model terbaik didapatkan dari proses validasi. Data ini digunakan dalam proses pengujian untuk mengukur seberapa baik model tersebut bekerja pada data yang belum pernah dilihat sebelumnya. Data uji yang digunakan pada penelitian ini sebanyak 10% dari total data citra, yaitu 90 citra. Proses pengujian ini juga dilakukan untuk mengetahui apakah model tersebut dapat memberikan *output* yang diharapkan, yaitu mendeteksi penyakit pada kategori yang benar seperti *canker*, karat daun, ataupun lumut daun.

3.3 Perancangan Antarmuka Sistem

Agar pengembangan aplikasi dapat dilaksanakan dengan lebih lancar dan sistematis, maka diperlukan tahap perancangan antarmuka sistem. Dikarenakan penelitian ini menggunakan perangkat *mobile android*, maka rancangan antarmuka yang akan dilakukan adalah rancangan aplikasi yang dikhususkan untuk *android*.

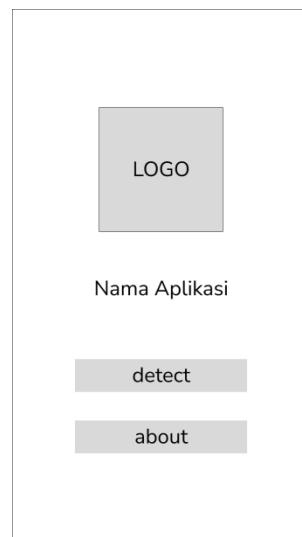
Untuk mengerti alur aktivitas aplikasi yang akan dibuat, perlunya dibuat diagram aktivitas yang dapat menggambarkan aktivitas pengguna. *Activity diagram* atau diagram aktivitas adalah diagram yang digunakan untuk menggambarkan interaksi antara sistem dan entitas luarannya, yang disebut aktor. Aktor dapat berupa individu, organisasi, atau sistem luar yang berinteraksi dengan sistem. Diagram aktivitas untuk sistem yang akan dibuat dapat dilihat pada gambar 3.21 berikut :



Gambar 3.21 Diagram Aktivitas

3.3.1 Rancangan tampilan beranda aplikasi

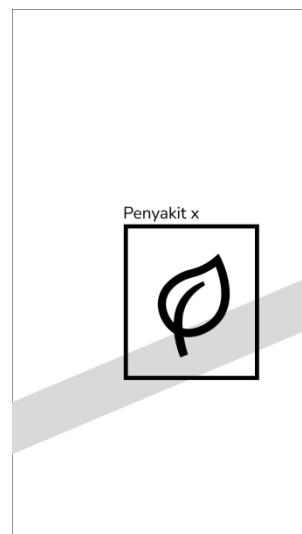
Beranda merupakan tampilan yang muncul pertama kali ketika aplikasi dimulai. Pada halaman ini, terdapat logo aplikasi, nama aplikasi, dan dua tombol yang digunakan untuk membawa pengguna ke halaman berikutnya sehingga alur aplikasi dapat berjalan. Desain halaman beranda dapat dilihat pada gambar 3.22 :



Gambar 3.22 Rancangan tampilan beranda aplikasi

3.3.2 Rancangan tampilan menu *detect*

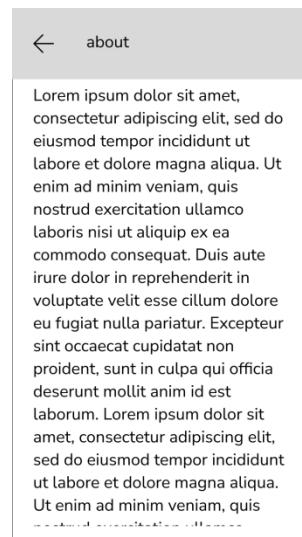
Ketika pengguna menekan *button detect* pada halaman beranda, maka aplikasi akan menampilkan halaman *detect* ini. Pada awalnya halaman ini akan meminta izin penggunaan kamera karena halaman ini membutuhkan akses kamera agar dapat berjalan. Setelah pengguna menyetujui izin akses kamera, maka tampilan kamera akan muncul dan pengguna dapat menyorot daun tanaman jeruk untuk mendeteksi penyakit yang dialami tanaman tersebut. Untuk rancangan halaman *detect* ini dapat dilihat pada gambar 3.23 berikut :



Gambar 3.23 Rancangan tampilan menu *detect*

3.3.3 Rancangan tampilan menu *about*

Halaman *about* adalah halaman yang akan berisi penjelasan aplikasi serta penyakit-penyakit yang dapat dideteksi dengan menggunakan aplikasi ini. Sama seperti halaman *detect*, halaman ini dapat diakses setelah menekan *button about* pada halaman beranda. Untuk rancangan halaman *about* ini dapat dilihat pada gambar 3.24 berikut :



Gambar 3.24 Rancangan tampilan menu *about*

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi sistem

Deteksi objek penyakit daun jeruk menggunakan metode YOLOv8 dikembangkan dan diimplementasikan ke dalam sistem dan memerlukan perangkat keras, perangkat lunak, serta pendukung lainnya, yaitu :

4.1.1 Perangkat keras dan perangkat lunak

Spesifikasi perangkat keras dan perangkat lunak yang digunakan untuk mengembangkan aplikasi ini adalah :

1. Laptop *Acer Aspire 5 A514-51G-52C3*
2. Processor *Intel Core i5 8265U @ 1.60GHz*
3. Storage *HDD 931GB TOSHIBA MQ04ABF100*
4. Random Access Memory (RAM) 8GB
5. Sistem Operasi *Windows 10 Pro 64-bit*
6. *Android Studio Hedgehog*
7. *Ultralytics Hub*
8. *Roboflow*
9. *Google Colab*
10. *Python 3.10*

Sementara spesifikasi perangkat berupa *smartphone* yang digunakan untuk memperoleh data latih dan melakukan testing aplikasi adalah:

1. *Smartphone Realme C21Y model RMX3261*
 - a. Random Access Memory (RAM) 4.0 GB
 - b. 64 GB internal storage
 - c. Android 11 OS
 - d. Octa-core (2x1.8 GHz Cortex-A75 & 6x1.8 GHz Cortex-A55)
 - e. Camera 13 MP/1080p
2. *Smartphone Samsung Galaxy A05*
 - a. Random Access Memory (RAM) 4.0 GB
 - b. 64 GB internal storage
 - c. Android 13 OS

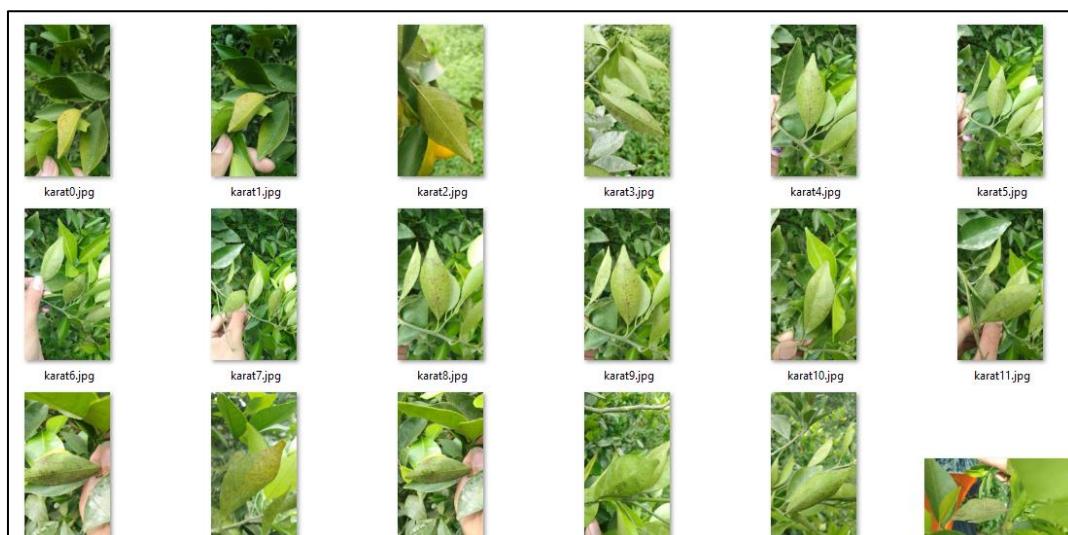
- d. Octa-core (2x2.0 GHz Cortex-A75 & 6x1.8 GHz Cortex-A55)
- e. Camera 50 MP/1080p

4.1.2 Implementasi data

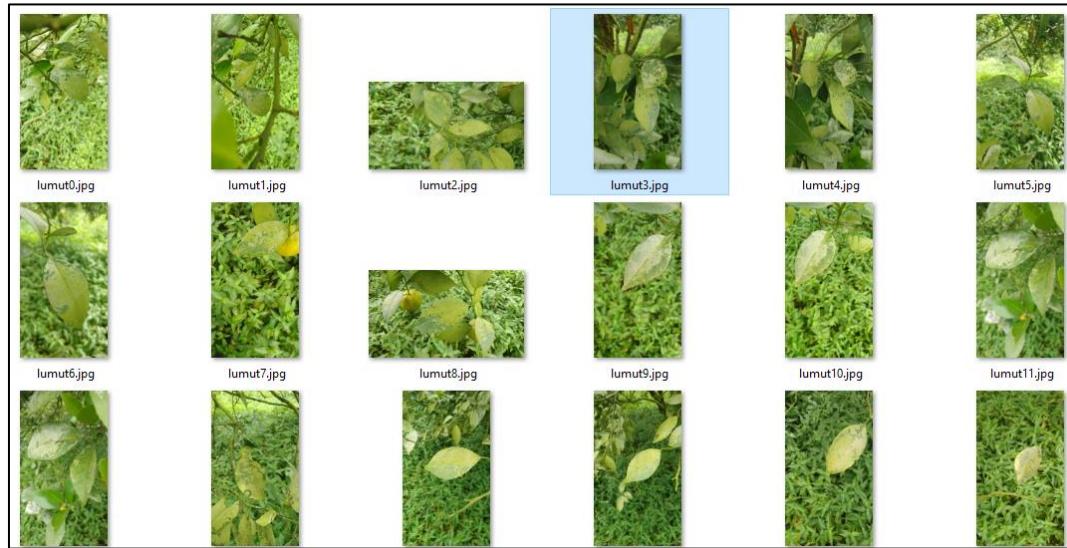
Penelitian ini menggunakan data yang diperoleh dari kebun jeruk di daerah Kabupaten Dairi, khususnya Daerah Sidikalang dan Daerah Sumbul. Pengambilan gambar dilakukan dengan menggunakan kamera resolusi 3.8 MP pada dua kali kunjungan yang berbeda saat mendatangi lokasi kebun jeruk. Jumlah data yang diambil sebanyak 450 citra yang dibagi menjadi 150 citra setiap kategori penyakit. Contoh data setiap data dapat dilihat pada gambar 4.1, 4.2, dan 4.3 berikut :



Gambar 4.1 Beberapa contoh data citra penyakit canker jeruk



Gambar 4.2 Beberapa contoh data citra penyakit karat daun jeruk



Gambar 4.3 Beberapa contoh data citra penyakit lumut daun jeruk

4.1.3 Implementasi pre-processing data

Tahap *pre-processing* adalah tahap yang dilakukan sebelum melakukan training data pada model yang ditentukan. Sesuai yang dipaparkan pada bab 3, proses yang dilakukan terdiri atas *labeling*, *resize*, dan *augmentation*. Untuk proses labeling sudah dilakukan dengan menggunakan aplikasi *labelImg*. Untuk proses *resize* dilakukan pada platform *Roboflow* seperti yang ditunjukkan pada gambar 4.4 berikut :

Preprocessing

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient	Edit	×
Resize Fit (black edges) in 640x640	Edit	×

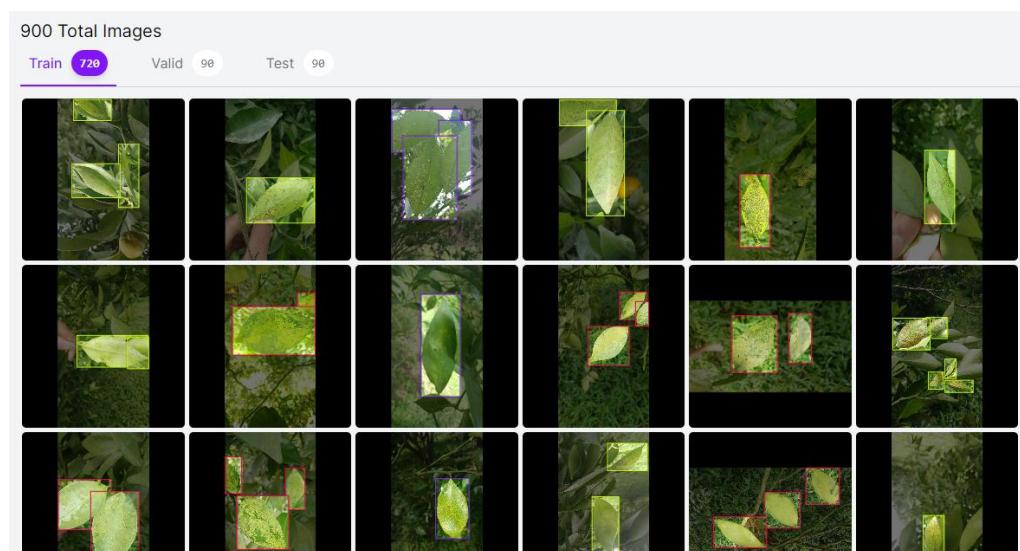
Gambar 4.4 Penggunaan *auto-orient* dan *resize* terhadap data

Auto-orient pada gambar 4.4 mengacu pada proses otomatisasi orientasi gambar. Proses ini biasanya dikonfigurasi secara otomatis pada platform *roboflow*. Langkah ini dilakukan untuk memastikan bahwa gambar-gambar yang masuk ke

dalam model *object detection* memiliki orientasi yang konsisten. Beberapa kamera atau perangkat yang menghasilkan gambar mungkin tidak secara otomatis menyesuaikan orientasi gambar tergantung pada cara gambar tersebut diambil. Sebagai contoh, gambar yang diambil dalam mode potret atau lanskap dapat memiliki orientasi yang berbeda. Hal tersebut dapat menyebabkan masalah dalam analisis gambar, termasuk dalam pengenalan objek.

Tahap *pre-processing* terakhir yang dilakukan adalah augmentasi. Untuk membuat data menjadi lebih banyak dan bervariasi maka diperlukan augmentasi. Dikarenakan fitur augmentasi pada Roboflow terbatas hingga tiga kali lipat, maka pada penelitian ini dilakukan augmentasi awal sehingga membuat data menjadi dua kali lipat lebih banyak. Augmentasi awal yang dilakukan adalah berupa *flip horizontal* dan *rotate 180*.

Dari hasil augmentasi tersebut, didapatkan jumlah data menjadi 900. Hasil augmentasi data yang didapat adalah 900 citra, yang dibagi menjadi 720 data latih, 90 data validasi, dan 90 data uji yang dilihat pada gambar 4.5. Dilakukan juga augmentasi hingga 2154 data yang nantinya digunakan sebagai pembanding hasil *training*.



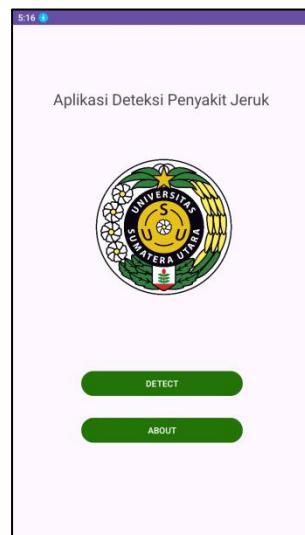
Gambar 4.5 Hasil augmentasi data

4.1.4 Implementasi desain interface

Sesuai dengan rancangan yang sudah dibuat pada bab sebelumnya, maka berikut adalah hasil implementasi desain aplikasi penelitian ini :

- Halaman beranda

Halaman beranda merupakan tampilan yang muncul pertama kali ketika aplikasi dimulai. Pada halaman ini, terdapat logo Universitas Sumatera Utara, nama aplikasi, yaitu ‘Aplikasi Deteksi Penyakit Jeruk’, dan dua tombol yang digunakan untuk mengantar pengguna ke halaman berikutnya sehingga alur aplikasi dapat berjalan. Adapun tampilan halaman beranda pada aplikasi dapat dilihat pada gambar 4.6:



Gambar 4.6 Tampilan halaman beranda

- Halaman *detect*

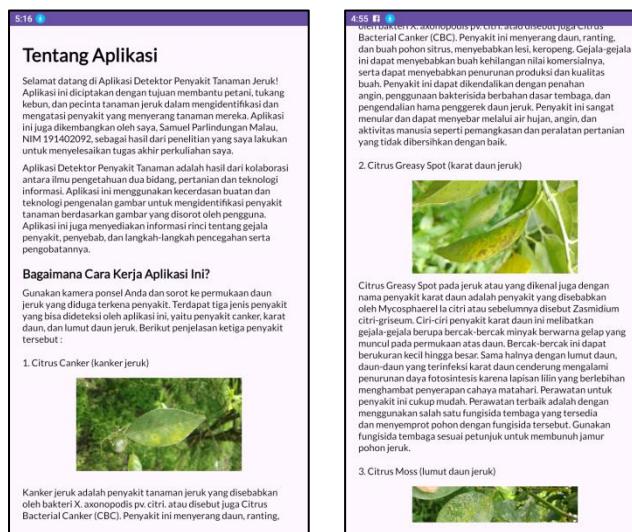
Ketika pengguna menekan tombol *detect* pada halaman beranda, maka aplikasi menampilkan halaman *detect* ini. Halaman ini meminta izin penggunaan kamera terlebih dahulu karena halaman ini membutuhkan akses kamera agar dapat berjalan. Setelah pengguna menyetujui izin akses kamera, maka tampilan kamera muncul dan pengguna dapat menyorot daun tanaman jeruk untuk mendeteksi penyakit yang dialami tanaman tersebut. Adapun tampilan halaman *detect* pada aplikasi dapat dilihat pada gambar 4.7:



Gambar 4.7 Tampilan halaman *detect*

- Halaman *about*

Halaman ini adalah halaman yang berisi penjelasan dan tujuan aplikasi serta penyakit-penyakit yang dapat dideteksi dengan menggunakan aplikasi ini. Sama seperti halaman *detect*, halaman ini dapat diakses setelah menekan *button about* pada halaman beranda. Adapun tampilan halaman *about* pada aplikasi dapat dilihat pada gambar 4.8:



Gambar 4.8 Tampilan halaman *about*

4.2 Prosedur Operasional

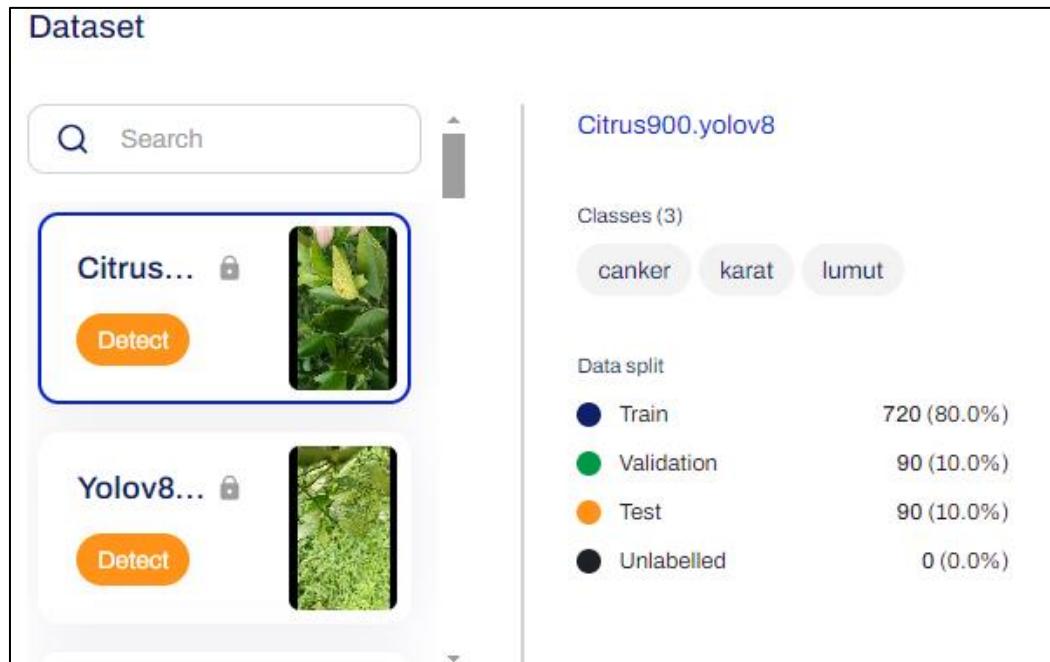
Sistem yang dikembangkan memiliki alur operasi yang dimulai dengan membuka aplikasi. Ketika pengguna membuka aplikasi, maka akan muncul halaman utama atau beranda. Halaman akan menampilkan nama aplikasi, logo, tombol *detect*, dan tombol *about*. Jika pengguna menekan tombol *about*, maka sistem akan membawa pengguna ke halaman *about*. Halaman ini berisi tentang cara penggunaan aplikasi dan penyakit-penyakit yang dapat dideteksi.

Jika pengguna menekan tombol *detect*, maka sistem akan membawa pengguna ke halaman *detect*. Pada halaman ini pengguna akan diminta untuk memberikan izin terhadap akses kamera karena tujuan utama dari halaman ini adalah mendeteksi penyakit tanaman jeruk dengan kamera. Setelah memberikan izin, maka pengguna akan mengarahkan kamera pada daun jeruk yang terkena penyakit. Terdapat tiga penyakit yang mampu dideteksi, yaitu *canker*, karat daun, dan lumut daun.

4.3 Pelatihan Sistem

Agar sistem dapat digunakan dan diimplementasikan pada perangkat *mobile*, maka sistem harus dilatih terlebih dahulu dengan data yang ada. Metode yang digunakan untuk melatih sistem adalah YOLOv8 dengan beberapa variasi jumlah epoch dengan tujuan untuk menemukan model yang paling baik. Dikarenakan *google colab* versi *non-premium* membatasi penggunaan unit komputasi, maka penelitian ini menggunakan bantuan platform *Ultralytics Hub*.

Seperti yang sudah dijelaskan pada bab 2, *Ultralytics Hub* memberikan kemudahan dalam pelatihan model terkhususnya YOLOv5 dan YOLOv8. Hal yang pertama dilakukan adalah mengunggah dataset yang diperoleh dari *Roboflow* ke dalam website *Ultralytics Hub*. Setelah data diunggah, maka dapat dilakukan pembuatan model dengan memilih data yang digunakan. Proses ini dapat dilihat pada gambar 4.9.



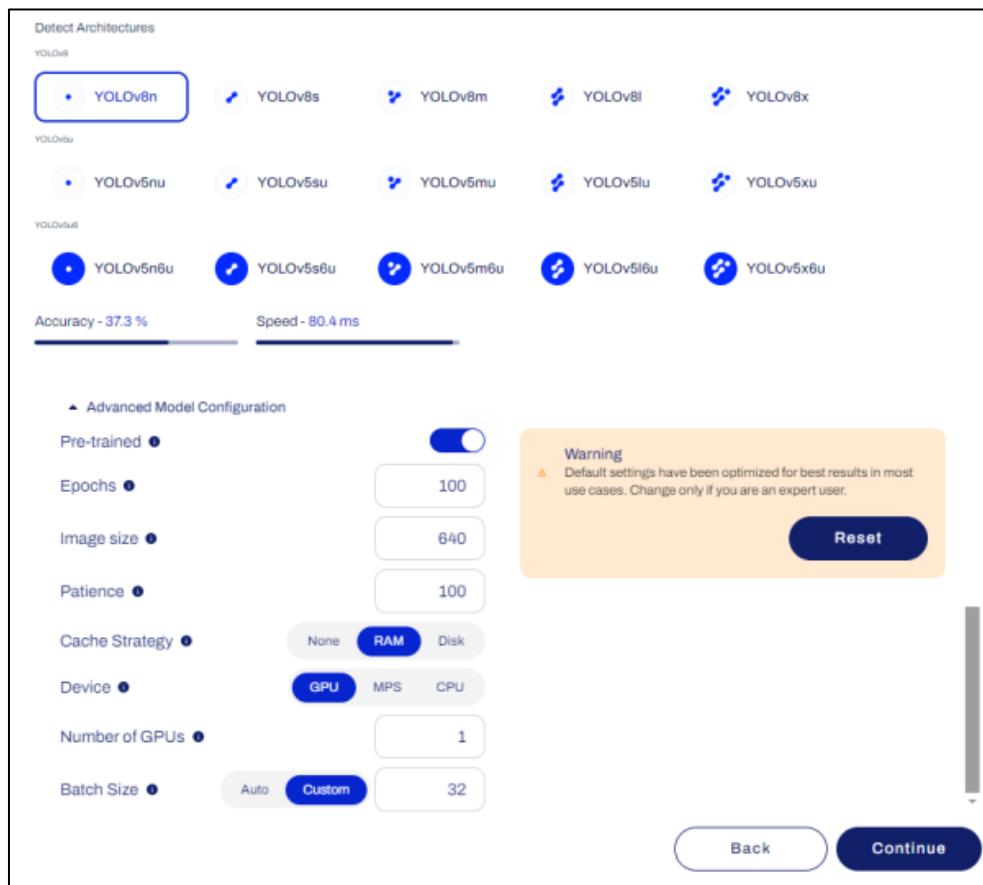
Gambar 4.9 Pemilihan data untuk dilatih pada platform *Ultralytics Hub*

Berikutnya setelah memilih dataset, dilanjut dengan memilih jenis arsitektur detektor yang diperlukan. Pada penelitian ini, detektor yang digunakan adalah YOLOv8n karena sesuai untuk pekerjaan yang ringan yaitu implementasi pada perangkat *mobile*. Setelah detektor dipilih, maka dilanjutkan dengan penentuan konfigurasi pelatihan model. *Ultralytics Hub* memiliki pengaturan bawaan, tetapi pengguna juga dapat mengubah konfigurasi agar dapat melatih model dengan pengaturan yang bervariasi untuk mencari hasil yang terbaik. Untuk tahap ini dapat dilihat pada gambar 4.10 Berikut penjelasan untuk setiap konfigurasi :

1. *Pre-trained* : Mengurangi waktu pelatihan dengan memulai proses *training* menggunakan model yang sudah dilatih
2. *Epochs* : Jumlah epoch (iterasi melalui keseluruhan dataset) untuk melatih model. Epoch yang lebih banyak biasanya menghasilkan waktu pelatihan yang lebih lama dan hasil akhir yang lebih baik
3. *Image size* : Ukuran gambar (dalam piksel) untuk melatih model. Semua gambar akan diubah ukurannya ke nilai ini pada sisi panjangnya selama pelatihan. Ukuran gambar yang lebih besar (e.g. 1280) meningkatkan akurasi sementara ukuran gambar yang lebih kecil (e.g. 320) berjalan lebih cepat
4. *Patience* : Seberapa banyak *epoch* yang diperlukan untuk tetap melakukan pelatihan setelah *Early Stopper* tidak menunjukkan peningkatan pada validasi.

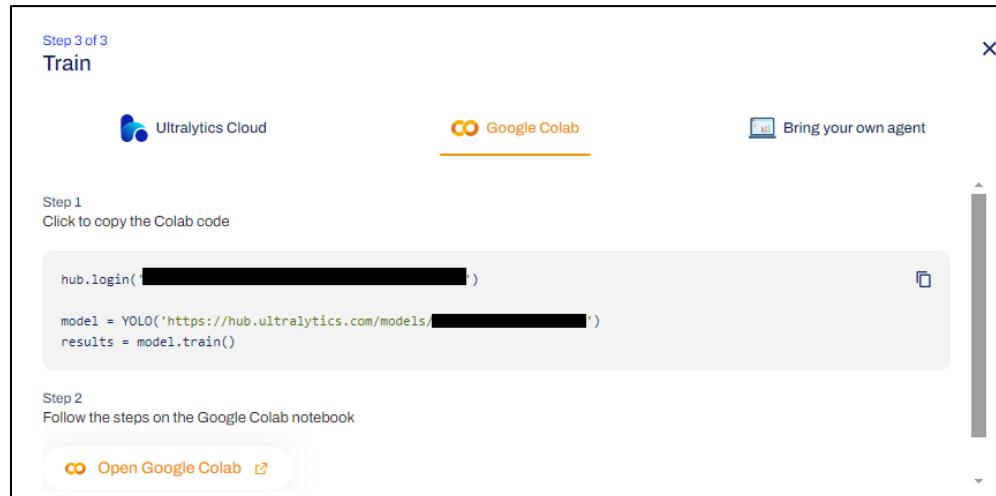
Kurangnya nilai *Patience* dapat mengurangi waktu pelatihan namun juga dapat menghentikan pelatihan sebelum waktunya.

5. *Cache Strategy* : *Caching* menghasilkan pelatihan yang lebih cepat dan direkomendasikan jika sumber daya memungkinkan.
6. *Device* : Perangkat CUDA yang digunakan. Secara default diatur pada penggunaan GPU
7. *Number of GPUs* : Jumlah GPU yang akan digunakan untuk melakukan pelatihan. Opsi ini digunakan untuk melakukan pelatihan dengan menggunakan beberapa GPU sekaligus. Jika hanya memiliki satu GPU, nilai sebaiknya tetap pada angka 1
8. *Batch size* : Berapa banyak gambar yang akan digunakan menjadi satu *batch* selama pelatihan. Nilai *batch* yang lebih besar menghasilkan pelatihan yang lebih cepat. AutoBatch diaktifkan secara default dan membantu memaksimalkan pemanfaatan CUDA untuk pelatihan tercepat dan hasil terbaik.

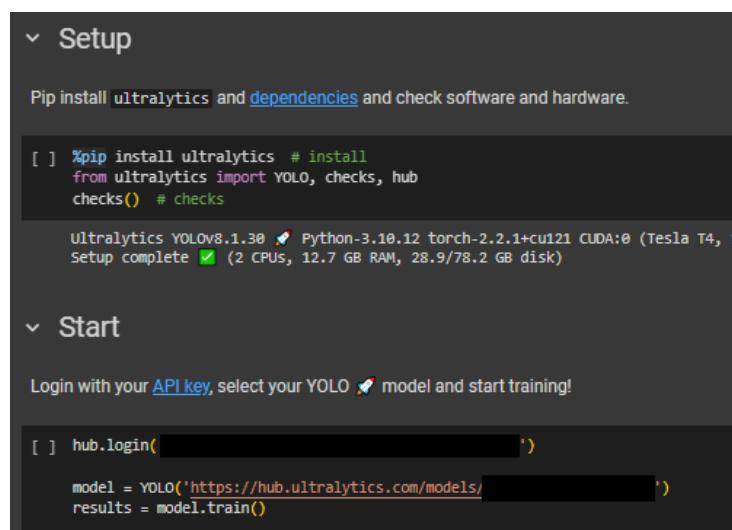


Gambar 4.10 Konfigurasi model yang digunakan

Tahap terakhir pada pelatihan pada platform *google colab*. *Ultralytics Hub* juga menyediakan pengaksesan *google colab* langsung dengan menggunakan model yang sudah dikonfigurasi tadi. Proses yang dilakukan dapat dilihat pada gambar 4.11 dan 4.12 :



Gambar 4.11 Ekspor ke *google colab* menggunakan API key *colab*



Gambar 4.12 Tampilan instalasi setup dan koneksi menggunakan API key

Pada penelitian ini pelatihan dilakukan beberapa kali dan dengan menggunakan dua jenis dataset serta beberapa konfigurasi yang berbeda. Untuk merangkum hasil pelatihan beberapa model pada penelitian dapat dilihat pada tabel 4.1 berikut :

Tabel 4.1 Hasil training

Model	epoch	precision	recall	mAP50	mAP50-95	train/box_loss	train/cls_loss
Yolov8m 100 epoch	100	0.92739,	0.90614,0.92913,0.82543,	0.35389,	0.22921,		
Yolov8n 100 epoch	100	0.88636,	0.92377,0.91922,0.80977,	0.34603,	0.23826,		
Yolov8n 150 epoch	150	0.91578,	0.94169,0.93429,0.81377,	0.29477,	0.20538,		
Yolov8n 200 epoch	200	0.89546,	0.90913,0.93456,0.81587,	0.26407,	0.18515,		
Yolov8n 50 epoch	50	0.9317,	0.81362,0.90771,0.7916,	0.43391,	0.31428,		
Yolov8n 40 epoch	40	0.91787,	0.84662,0.93448,0.77124,	0.62873,	0.38484,		
Yolov8n 50 epoch	50	0.91292,	0.89652,0.93358,0.77547,	0.58804,	0.35805,		
Yolov8n 100 epoch	100	0.88399,	0.88987,0.9236,0.76371,	0.48199,	0.29027,		
Yolov8n 150 epoch	150	0.89248,	0.90876,0.92902,0.77461,	0.42705,	0.26237,		
Yolov8n 200 epoch	200	0.91267,	0.85997,0.91665,0.77261,	0.39198,	0.23177,		
Yolov8s 50 epoch	50	0.89325,	0.90542,0.93017,0.76718,	0.53635,	0.32609,		
		900 data :					
		2154 data :					

Penamaan model diberikan menggunakan format *detector* yang digunakan dan jumlah epoch yang digunakan. Pada penelitian ini juga dilakukan pelatihan model dengan jumlah data 2154 yang didapat dengan cara melakukan augmentasi beberapa kali pada data awal. Namun hasil yang didapatkan dari pelatihan model tersebut kurang memuaskan dibandingkan dengan model yang menggunakan 900 data.

Dari rangkaian pelatihan yang dilakukan, maka didapatkan hasil yang paling baik dilakukan oleh model ‘Yolov8n 150 epoch’ dengan nilai *precision*, *recall*, *mAP50*, dan *map50-95* yang tinggi dibandingkan model lainnya. Setelah dilakukan training, maka model memerlukan proses validasi untuk mengevaluasi model. Hasil validasi model ‘Yolov8n 150 epoch’ dapat dilihat pada gambar 4.13 berikut :

```
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.1.22 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
    Class   Images   Instances   Box(P      R      mAP50   mAP50-95): 100% |██████████| 1/1
        all       33       64     0.906     0.926     0.942     0.822
        canker    33       28     0.826     0.964     0.95      0.837
        karat     33       18       1     0.878     0.956     0.846
        lumut     33       18     0.894     0.934     0.92      0.783
Speed: 0.2ms preprocess, 2.1ms inference, 0.0ms loss, 1.8ms postprocess per image
Results saved to runs/detect/train
```

Gambar 4.13 Hasil validasi model

Model terbaik yang didapat adalah model Yolov8n yang dibuat menggunakan konfigurasi jumlah *epoch* 150, *batch size* 16, *image size* 640, dan *patience* 100. Adapun hasil yang didapatkan dari pelatihan model Yolov8n dengan *batch size* 16 dan *epoch* 150 tersebut dapat dilihat pada tabel 4.2 berikut.

Tabel 4.2 Hasil pelatihan model

epoch	box loss	precision	recall	mAP50
10	0.96265	0.84143	0.24795	0.2997
15	0.92406	0.76871	0.76023	0.81781
20	0.87155	0.70042	0.77646	0.85083
25	0.87134	0.8602	0.7502	0.82205
30	0.78296	0.75788	0.8191	0.86504
35	0.78893	0.87852	0.82262	0.86598
40	0.75181	0.8075	0.91183	0.87415
45	0.73344	0.77829	0.88886	0.90311
50	0.72141	0.83241	0.8545	0.88885
55	0.69238	0.86366	0.89366	0.93915
60	0.69172	0.88619	0.92832	0.91632
65	0.66715	0.88679	0.89505	0.89054
70	0.65787	0.91017	0.83853	0.92132
75	0.65644	0.82737	0.89929	0.87734
80	0.62547	0.88745	0.86668	0.90645
85	0.6099	0.90655	0.85317	0.91202
90	0.61314	0.81669	0.87797	0.90272
95	0.57882	0.8803	0.87963	0.91127
100	0.55343	0.89399	0.84101	0.87466
105	0.54271	0.88197	0.90395	0.9116
110	0.55821	0.87417	0.91371	0.92246
115	0.50864	0.88538	0.92721	0.93242
120	0.51343	0.88339	0.89423	0.91843
125	0.48261	0.85067	0.93425	0.92754
130	0.47754	0.89914	0.93708	0.93102
135	0.46602	0.89606	0.93549	0.93437
140	0.46374	0.91624	0.91906	0.9439
145	0.304	0.90932	0.93904	0.9412
150	0.29477	0.91578	0.94169	0.93429

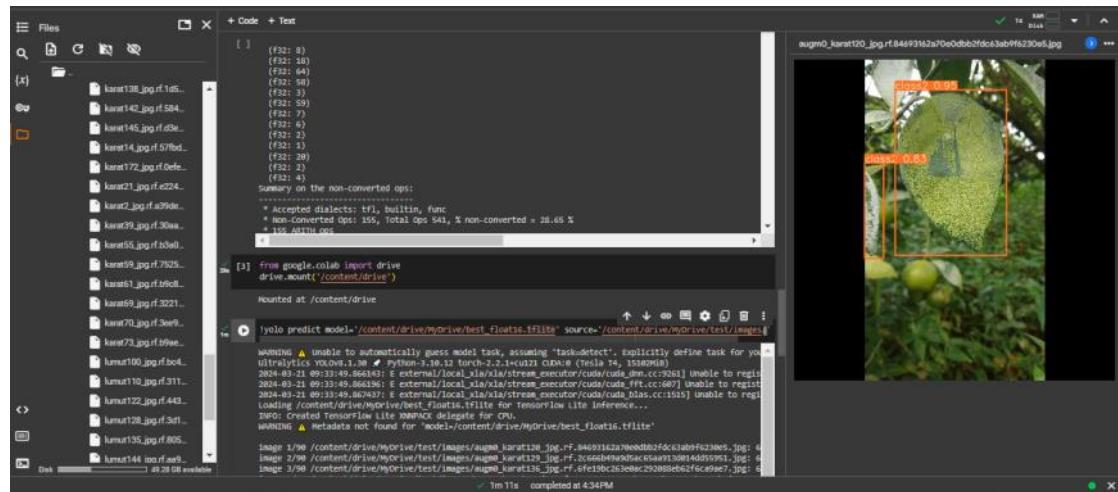
Setelah pelatihan dan validasi model dilakukan, maka dihasilkan file ‘.pt’ (*pytorch*) yang digunakan untuk konversi menjadi bentuk ‘.tflite’ (*tensorflow lite*) agar dapat diimplementasi pada aplikasi berbasis *android*. Model yang dihasilkan pada pelatihan YOLO ada dua, yaitu ‘last.pt’ dan ‘best.pt’. Model ‘last.pt’ merupakan model yang disimpan pada akhir setiap *epoch* selama proses pelatihan sementara model ‘best.pt’ adalah model yang disimpan ketika model mencapai kinerja terbaik

pada data validasi selama proses pelatihan. Konversi menjadi file ‘.tflite’ dilakukan menggunakan kode pada gambar 4.14 berikut :

```
!yolo export model=/content/runs/detect/train/weights/best.pt format=tflite
ultralytics YOLOv8.1.22 🚀 Python-3.10.12 torch-2.1.0+cu121 CPU (Intel Xeon 2.20GHz)
<class 'ultralytics.nn.tasks.DetectionModel'>
Model summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
```

Gambar 4.14 Konversi file pytorch menjadi tflite

Untuk membuktikan file tflite benar-benar dapat berfungsi dengan baik, maka dapat dilakukan pengujian terlebih dahulu pada *google colab* menggunakan data uji. Contoh hasil pengujian dapat dilihat pada gambar 4.15 berikut :

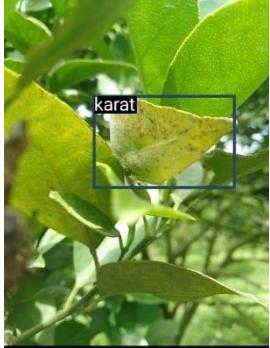
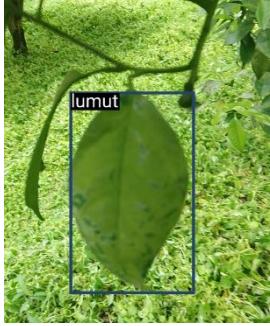


Gambar 4.15 Pengujian tflite pada data uji

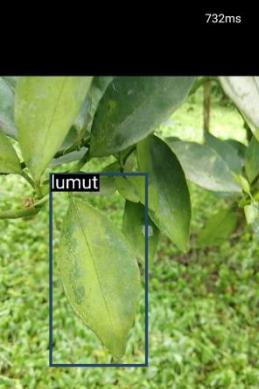
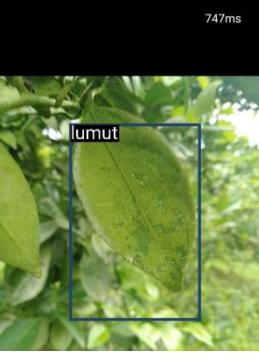
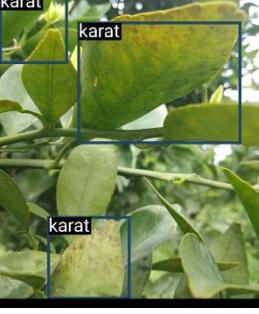
4.4 Pengujian Sistem

Pengujian sistem untuk penelitian ini dilakukan secara *real-time* dan dilakukan langsung pada tanaman jeruk yang terkena penyakit. Pendekripsi penyakit tanaman daun jeruk dibagi atas tiga kelas, yaitu *canker*, karat daun, dan lumut daun. Adapun contoh hasil pengujian pendekripsi penyakit menggunakan aplikasi yang sudah dikembangkan dapat dilihat pada tabel 4.3.

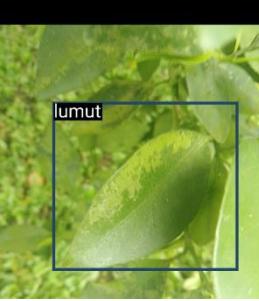
Tabel 4.3 Contoh hasil pengujian aplikasi

No.	Citra	Desired Output	Actual Output	Status
1	 735ms	Karat	Karat	Benar
2	 750ms	Lumut	Lumut	benar
3	 746ms	Canker	Canker	benar

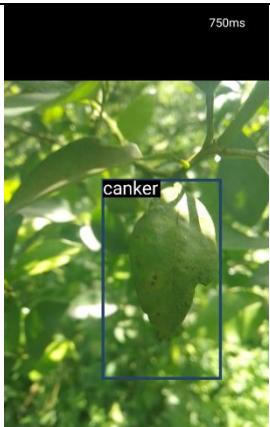
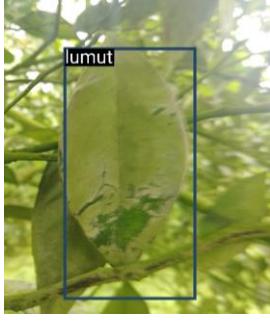
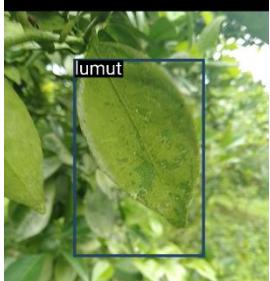
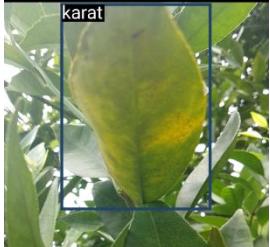
Lanjutan Tabel 4.3 Contoh hasil pengujian

No.	Citra	Desired Output	Actual Output	Status
4	 732ms	lumut	lumut	benar
5	 747ms	lumut	lumut	benar
6	 748ms	karat	karat	benar
7	 748ms	Karat, karat, karat	Karat, karat, karat	benar

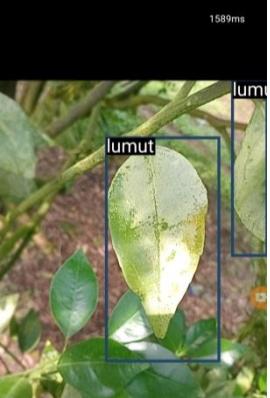
Lanjutan Tabel 4.3 Contoh hasil pengujian

No.	Citra	Desired Output	Actual Output	Status
8	 736ms	karat	karat	benar
9	 737ms	lumut	lumut	benar
10	 737ms	Lumut, lumut, lumut	Lumut, lumut, canker	salah
11	 735ms	Canker, canker, canker, canker	Canker, canker, canker, canker	benar

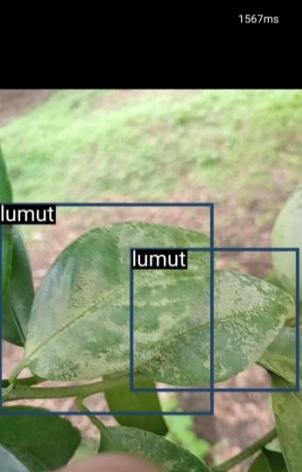
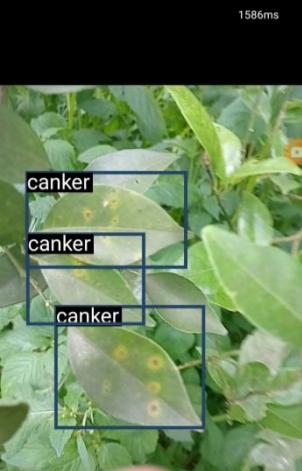
Lanjutan Tabel 4.3 Contoh hasil pengujian

No.	Citra	Desired Output	Actual Output	Status
12	 750ms	canker	Canker	benar
13	 735ms	lumut	lumut	benar
14	 747ms	lumut	Lumut	benar
15	 738ms	karat	karat	benar

Lanjutan **Tabel 4.3** Contoh hasil pengujian

No.	Citra	Desired Output	Actual Output	Status
16	 1589ms	Lumut, lumut	Lumut, lumut	benar
17	 1569ms	canker	canker	benar
18	 1586ms	lumut	karat	salah

Lanjutan **Tabel 4.3** Contoh hasil pengujian

No.	Citra	Desired Output	Actual Output	Status
19		Lumut, lumut	Lumut, lumut	benar
20		Canker, canker, canker	Canker, canker, canker	benar

Tabel 4.3 tersebut merupakan beberapa contoh hasil pengujian aplikasi yang telah dikembangkan dan dilakukan terhadap tanaman jeruk yang terkena penyakit. Sementara untuk laporan pengujian secara lengkap dapat dilihat pada lampiran penelitian.

Pengujian aplikasi dilakukan dengan mengujinya pada tanaman jeruk secara langsung dan mendapatkan 90 uji dalam bentuk tangkapan layar yang diperlukan untuk menentukan nilai *confusion matrix*. Jenis *confusion matrix* yang digunakan pada penelitian ini adalah *Image-level confusion matrix*, dimana setiap nilai dalam matriks sesuai dengan jumlah prediksi dalam satu gambar. Dalam hal ini ada tidaknya hasil suatu prediksi tiap objek pada gambar dianggap secara keseluruhan. Adapun hasil *confusion matrix* penelitian ini dapat dilihat pada tabel 4.4 berikut.

Tabel 4.4 Confusion matrix

		Aktual		Total
		Canker	Karat Daun	
Prediksi	Canker	30	0	2
	Karat Daun	0	30	1
	Lumut Daun	0	0	27
Total		30	30	30
				90

Tabel 4.4 menunjukkan terdapat beberapa kesalahan prediksi pada kelas lumut daun. Kesalahan prediksi terjadi karena kurang efektifnya variasi data dan pelabelan pada kategori lumut daun sehingga sistem kesulitan dalam membedakan kelas daun tersebut dalam beberapa kasus. Dari data tersebut, dapat disimpulkan hasil nilai *true positive* dan *false negative*. Untuk nilai *true negative* dan *false positive* tidak ditemukan pada 90 citra yang diuji coba. Hasil nilai tersebut dapat dilihat pada tabel 4.5 berikut.

Tabel 4.5 Hasil nilai *True Positive* dan *False Negative*

	True Positive	False Negative	Total
Canker	30	0	30
Karat Daun	30	0	30
Lumut Daun	27	3	30
Total	87	3	90

Berdasarkan nilai-nilai tersebut, dapat dihitung nilai *precision*, *recall*, dan *F1-score* untuk setiap kelas penyakit.

a. *Precision*

Nilai *precision* dapat didapat dengan menggunakan rumus 2.7.

$$\text{Precision cancer} : \frac{30}{(30 + 0)} \times 100\% = 100\%$$

$$\text{Precision karat daun} : \frac{30}{(30 + 0)} \times 100\% = 100\%$$

$$\text{Precision lumut daun} : \frac{27}{(27 + 0)} \times 100\% = 100\%$$

b. *Recall*

Nilai *recall* dapat didapat dengan menggunakan rumus 2.8.

$$\text{Recall cancer} : \frac{30}{(30 + 0)} \times 100\% = 100\%$$

$$\text{Recall karat daun : } \frac{30}{(30 + 0)} \times 100\% = 100\%$$

$$\text{Recall lumut daun: } \frac{27}{(27 + 3)} \times 100\% = 90\%$$

c. *F1-score*

Nilai *F1-score* dapat didapat dengan menggunakan rumus 2.9.

$$F1 - score \text{ canker} : 2x \frac{(100\% * 100\%)}{(100\% + 100\%)} = 100\%$$

$$F1 - score \text{ karat daun} : 2x \frac{(100\% * 100\%)}{(100\% + 100\%)} = 100\%$$

$$F1 - score \text{ lumut daun}: 2x \frac{(100\% * 90\%)}{(100\% + 90\%)} = 94,74\%$$

Dari perhitungan yang telah dilakukan, maka nilai-nilai tersebut dapat dirangkum pada tabel 4.6 berikut.

Tabel 4.6 Hasil nilai *precision*, *recall*, dan *F1-score*

	Precision	Recall	F1-score
Canker	100,00%	100,00%	100,00%
Karat Daun	100,00%	100,00%	100,00%
Lumut Daun	100,00%	90,00%	94,74%

Berdasarkan hasil pengujian yang telah dilakukan dengan menggunakan aplikasi yang dibuat menggunakan YOLOv8 tersebut, maka dihasilkan nilai akurasi sistem dengan menggunakan rumus 2.11.

$$\text{Accuracy} = \frac{(87 + 0)}{(87 + 0 + 0 + 3)} \times 100\% = 96,67\%$$

Sistem dapat melakukan deteksi 3 jenis penyakit untuk daun jeruk yang dimana dapat berjalan pada perangkat *mobile*. Pengujian sistem yang telah dihasilkan ini dilakukan pada dua waktu yang berbeda dan dengan dua perangkat yang berbeda. Pengujian pertama dilakukan menggunakan *Smartphone Realme C21Y model RMX3261* dengan tujuan untuk melengkapi laporan penelitian ini sehingga hasil yang didapat hanya sekitar 30 gambar pengujian. Pengujian menggunakan perangkat ini dapat mendeteksi penyakit jeruk dengan waktu inferensi sebesar 700 hingga 850 ms. Waktu inferensi dalam deteksi objek mengacu pada jumlah waktu yang diperlukan model terlatih

untuk memproses gambar masukan dan menghasilkan prediksi keluaran yang sesuai diukur dengan satuan *milisecond* (ms) atau *second* (s).

Sementara itu pengujian kedua dilakukan dengan menggunakan *Smartphone Samsung Galaxy A05*. Perangkat ini digunakan untuk pendokumentasi sebagian besar gambar pengujian pendekripsi penyakit daun jeruk serta untuk kepentingan perekaman video uji coba di kebun jeruk. Pengujian dengan perangkat ini dapat mendekripsi penyakit jeruk dengan waktu inferensi sebesar 1500 hingga 2000 ms. Kedua perangkat memiliki spesifikasi perangkat yang mirip namun memiliki *Operating System* (OS) yang berbeda. Ketidaksesuaian spesifikasi tersebut dengan aplikasi yang dibuat dapat memberikan hasil waktu inferensi yang berbeda. Masalah tersebut dapat diatasi dengan penyesuaian kembali ketika pembuatan ulang aplikasi dilakukan.

Adapun sistem juga menghasilkan beberapa kesalahan prediksi sebesar 3,33% yang menunjukkan bahwa metode yang digunakan belum sempurna dan memiliki kekurangan dalam mendekripsi objek daun yang terkena penyakit. Kesalahan tersebut disebabkan oleh kurangnya jumlah dan variasi data yang dilatih sehingga sistem menjadi tidak efektif dalam menentukan jenis penyakit pada beberapa daun yang memiliki ciri yang dianggap mirip oleh sistem melalui kamera perangkat. Pada penelitian ini juga tidak ada kondisi tambahan pengujian yang digunakan, seperti jarak pendekripsi dan pengaruh intensitas cahaya. Jika kedua kondisi tersebut digunakan pada pengujian ini, maka kemungkinan akan membuat nilai akurasi berkurang. Sebagai hasil akurasi penelitian yang didapatkan menggunakan rumus 2.11 pada penelitian deteksi penyakit tanaman jeruk berbasis citra daun dengan menggunakan metode YOLOv8 ini adalah 96.67% yang dianggap sebagai hasil yang cukup memuaskan meskipun tidak mencapai angka sempurna.

Adapun pada penelitian ini juga dilakukan survey atau pengambilan data kepuasan pengguna dengan cara penyebaran kuisioner. Hal ini dilakukan dengan tujuan untuk mengetahui apakah aplikasi ini sudah cukup layak dan baik untuk digunakan baik oleh petani jeruk maupun masyarakat umum. Kuesioner ini berisi pernyataan dan pertanyaan yang dibuat dengan *google form*. Penyebaran *link google form* dilakukan melalui media sosial berupa *Whatsapp* dan *Facebook*. Jawaban yang didapat dari responden akan digunakan dalam menyimpulkan hasil dari survey ini.

Dari hasil kuisioner yang disebar hingga 8 Juli 2024, didapatkan 15 responden yang dibagi menjadi dua jenis, yaitu pihak yang pernah memiliki atau mengelola kebun jeruk dan masyarakat umum yang tidak pernah. Penilaian dengan kuisioner ini dilakukan pada dua aspek yakni pemahaman (*understandability*) dan kegunaan (*functionality*). Para pengisi kuisioner diminta untuk mencoba aplikasi ini dan diminta untuk memberikan penilaian berdasarkan skala likert dimana skala likert melakukan penilaian dengan skala dari skala 1 (sangat negatif) sampai 5 (sangat positif). Berikut penjelasan kedua aspek tersebut :

- *Understandability*

Aspek ini menunjukkan seberapa paham pengguna terhadap aktivitas yang dilakukan pada aplikasi yang digunakan. Terdapat lima skala penilaian yang diterapkan pada penilaian aspek ini, yaitu :

- Skala 1 : menyatakan aplikasi sangat sulit dipahami.
- Skala 2 : menyatakan aplikasi sulit dipahami.
- Skala 3 : menyatakan aplikasi cukup mudah dipahami.
- Skala 4 : menyatakan aplikasi mudah dipahami.
- Skala 5 : menyatakan aplikasi sangat mudah dipahami.

Adapun pertanyaan yang menggunakan aspek *understandability* yang digunakan adalah :

1. Apakah instruksi penggunaan aplikasi tersebut mudah dimengerti?
2. Apakah instruksi atau tutorial penggunaan aplikasi tersebut dalam halaman "About" berguna?
3. Apakah pengguna merasa tertarik dengan tampilan dari aplikasi tersebut ?
4. Jika pengguna menggunakan aplikasi tersebut, apakah pengguna merasa kesulitan dalam penggunaannya ?

- *Functionality*

Aspek ini menunjukkan seberapa bermanfaat aplikasi ini pada pengguna. Terdapat lima skala penilaian yang diterapkan pada penilaian aspek ini, yaitu:

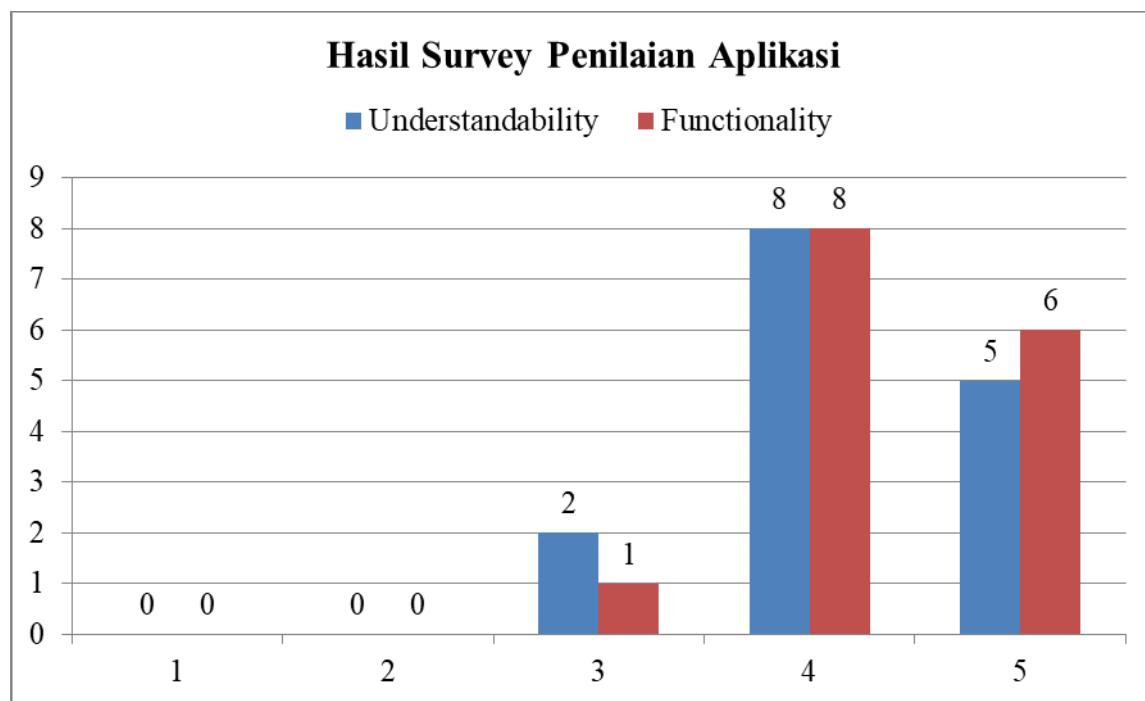
- Skala 1 : menyatakan aplikasi sangat tidak bermanfaat.
- Skala 2 : menyatakan aplikasi kurang bermanfaat.
- Skala 3 : menyatakan aplikasi cukup bermanfaat.

- Skala 4 : menyatakan aplikasi bermanfaat.
- Skala 5 : menyatakan aplikasi sangat bermanfaat.

Adapun pertanyaan yang menggunakan aspek *understandability* yang digunakan adalah :

1. Apakah informasi yang diberikan aplikasi tentang penyakit daun tanaman jeruk sudah informatif?
2. Apakah aplikasi tersebut sudah cukup membantu dalam mengklasifikasikan penyakit pada tanaman jeruk?
3. Apakah aplikasi tersebut dapat berguna saat digunakan untuk mengelola kebun jeruk?

Dari 15 responden yang sudah memberikan jawaban atas kuisisioner yang diberikan maka didapatkan hasil data survey yang dapat dilihat pada diagram pada gambar 4.16 berikut.



Gambar 4.16 Diagram hasil penilaian aplikasi

Berdasarkan hasil penilaian oleh 15 responden menggunakan skala likert dan didasarkan pada dua aspek yakni aspek *understandability* dan aspek *functionality* terhadap fitur pada aplikasi yang telah dibuat, Dapat disimpulkan bahwa aplikasi yang dihasilkan pada penelitian ini memiliki kemudahan pemahaman dan fungsionalitas yang baik.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang sudah dilakukan menggunakan metode *You Only Look Once* (YOLO) versi 8 mengenai pendektsian penyakit tanaman jeruk melalui citra daun secara *real time* diperoleh kesimpulan berupa :

1. Berhasil melakukan pendektsian penyakit tanaman jeruk melalui citra daun secara *Real Time* dengan mengimplementasikan metode *You Only Look Once* (YOLO) pada aplikasi yang dibuat.
2. Metode *You Only Look Once* (YOLO) versi 8 mampu mendekksi tiga jenis penyakit berbeda pada citra daun jeruk dengan akurasi 96,67% dari 90 data yang diuji.
3. Sistem dapat diimplementasikan pada perangkat *mobile*, yaitu *smartphone android*, dan dapat melakukan pendektsian penyakit menggunakan kamera perangkat secara *real time*.

5.2 Saran

Penelitian ini jauh dari kata sempurna sehingga dapat dikembangkan pada penelitian berikutnya dengan saran-saran berupa :

1. Kesalahan prediksi terjadi karena kurang efektifnya variasi data dan pelabelan pada kategori lumut daun sehingga dalam beberapa kasus, sistem kesulitan dalam membedakan kategori daun tersebut dengan kategori lain. Sehingga pada penelitian selanjutnya dapat dilakukan pelabelan data yang lebih baik.
2. Diharapkan penelitian berikutnya dapat menghasilkan akurasi yang lebih baik.
3. Penelitian berikutnya dapat memberikan tambahan variasi pengujian sistem seperti mempertimbangkan jarak pendektsian dari kamera dan perbedaan intensitas cahaya.
4. Penelitian berikutnya dapat menambah kategori penyakit dan juga jumlah data yang lebih banyak.
5. Menggunakan metode lain ataupun versi YOLO yang lebih terbaru seperti YOLOv9 dan YOLOv10.

DAFTAR PUSTAKA

- Aguilera-Cogley, V., Sedano, E. & Vincent, A., 2023. Inoculum and disease dynamics of citrus greasy spot caused by *Zasmidium citri-griseum* in sweet orange in Panama. *Plant pathology* 72.4 (2023), pp. 696-707.
- Alberto , G. M. et al., 2020. Panorama of citrus canker in the United States. *Tropical Plant Pathology* 45 (2020), pp. 192-199.
- Anggreani, N., 2020. Analisis Kadar Vitamin C pada Jeruk Lokal di Provinsi Bengkulu. *Jurnal Ilmiah Pharmacy Sekolah Tinggi Kesehatan Al-Fatah Bengkulu*, Volume 7, p. 2.
- Ariesdianto , R., Fitri, Z., Madjid, A. & Imron, A., 2021. Identifikasi Penyakit Daun Jeruk Siam Menggunakan K-Nearest Neighbor. *Jurnal Ilmu Komputer dan Informatik (JIKI)*, pp. 133-140.
- Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M., 2020. Yolov4: Optimal speed and accuracy of object detection.. *arXiv preprint arXiv:2004*, p. 10934.
- Dang-Ngoc, H., Cao, T. N. & Dang-Nguyen, C., 2021. Citrus leaf disease detection and classification using hierarchical support vector machine. *International Symposium on Electrical and Electronics Engineering (ISEE)*, pp. 69-74.
- Febrinanto, F., Dewi, C. & Triwiratno, A., 2019. The Implementation of K-Means Algorithm as Image Segmenting Method in Identifying the Citrus Leaves Disease. *IOP Conf. Series: Earth and Environmental Science*, p. 243 012024.
- Gallo, I. et al., 2023. Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images. *Remote Sensing*, 15(2), p. 539.
- Jocher, G., 2023. *Github*. [Online] Available at: <https://github.com/ultralytics/yolov5/issues/8785> [Diakses 27 May 2024].
- Jocher,G., 2023. *Github*. [Online] Available at: <https://github.com/ultralytics/ultralytics/issues/189#issuecomment-1520733133> [Diakses 28 May 2024].
- Jocher, G., 2023. *Github*. [Online] Available at: <https://github.com/ultralytics/ultralytics/issues/189#issuecomment-1489780341> [Diakses 28 May 2024].
- Lawal, M. O., 2021. Tomato detection based on modified YOLOv3 framework. *Scientific Reports*, 11(1), p. 1447.

- Lei, Y. et al., 2020. Characteristics of nutrients and leaf surface organisms under different citrus moss severities. *Acta Agriculturae Zhejiangensis*, 32(4), p. 632.
- Lestari, F., Purnama, I., Sajiah, A. & Aksara, L., 2019. IDENTIFIKASI PENYAKIT TANAMAN JERUK SIAM MENGGUNAKAN METODE M-SVM. *Seminar Nasional Aptikon (SEMNASTIK)*.
- Li, C. et al., 2022. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.
- Luo, D. et al., 2023. Citrus Diseases and Pests Detection Model Based on Self-Attention YOLOV8. *IEEE Access*.
- Maleh, I. M. D. et al., 2023. Implementasi Algoritma You Only Look Once (YOLO) Untuk Object Detection Sarang Orang Utan. *Jurnal Informatika*, 10(1).
- Naharsari, N. D., 2007. *Bercocok Tanam Jeruk*. Jawa Barat: Azka Press.
- Parra, D., Escobar Sanabria, D. & Camargo, C., 2023. A Methodology and Open-Source Tools to Implement Convolutional Neural Networks Quantized with TensorFlow Lite on FPGAs.. *Electronics*, 12(20), p. 4367.
- Qiu, R.-Z. et al., 2022. An automatic identification system for citrus greening disease (Huanglongbing) using a YOLO convolutional neural network. *Front Plant Sci*. 2022;13:5337..
- Ratna, S., 2020. Pengolahan Citra Digital Dan Histogram Dengan Phyton Dan Text Editor Phycharm. *Technologia: Jurnal Ilmiah*, 11(3), pp. 181-186.
- Saputra, D. H. & Imran, B., 2023. OBJECT DETECTION UNTUK MENDETEKSI CITRA BUAH-BUAHAN MENGGUNAKAN METODE YOLO. *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, 2(2), pp. 70-80.
- Schlegi, T. et al., 2015. Predicting semantic descriptions from medical images with convolutional neural networks. *International Conference on Information Processing in Medical Imaging*, pp. 437-448.
- Szeliski, R., 2022. *Computer vision: algorithms and applications*. s.l.:Springer Nature.
- Talaat, F. M. & ZainEldin, H., 2023. An improved fire detection approach based on YOLO-v8 for smart cities. *Neural Computing and Applications*, 35(28), pp. 20939-20954.

- Terven, J., Coroda-Esparza, D. M. & Romero-Gonzales, J. A., 2023. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4), pp. 1680-1716.
- Ultralytics, 2022. *Ultralytics YOLOv8 Docs.* [Online] Available at: <https://docs.ultralytics.com/> [Diakses 30 01 2024].
- Ultralytics, 2023. *Ultralytics.* [Online] Available at: <https://docs.ultralytics.com/> [Diakses 2 June 2024].
- Wang, C.-Y., Bochovskiy, A. & Liao, H.-Y. M., 2023. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2464-2475.
- Yang, T. et al., 2023. An approach for plant leaf image segmentation based on YOLOV8 and the improved DEEPLABV3+. *Plants* 12, Issue 19, p. 3438.
- Zheng, Z. et al., 2020. Distance-IoU loss: Faster and better learning for bounding box regression.. *Proceedings of the AAAI conference on artificial intelligence*, 34(07), pp. 12993-13000.



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI**
UNIVERSITAS SUMATERA UTARA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

Jalan Universitas No. 9A Gedung A, Kampus USU Medan 20155, Telepon: (061) 821007
Laman: <http://Fasilkomti.usu.ac.id>

**KEPUTUSAN
DEKAN FAKULTAS ILMU KOMPUTER
DAN TEKNOLOGI INFORMASI
NOMOR : 2725/UN5.2.14.D/SK/SPB/2024**

**DEKAN FAKULTAS ILMU KOMPUTER
DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA**

Membaca : Surat Permohonan Mahasiswa Fasilkom-TI USU tanggal 10 Juli 2024 perihal permohonan ujian skripsi:
Nama : SAMUEL PARLINDUNGAN MALAU
NIM : 191402092
Program Studi : Sarjana (S-1) Teknologi Informasi
Judul Skripsi : Deteksi Penyakit Tanaman Jeruk Berbasis Citra Daun Dengan Metode You Only Once Versi 8 (YOLOV8)

Memperhatikan : Bawa Mahasiswa tersebut telah memenuhi kewajiban untuk ikut dalam pelaksanaan Meja Hijau Skripsi Mahasiswa pada Program Studi Sarjana (S-1) Teknologi Informasi Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara TA 2023/2024.

Menimbang : Bawa permohonan tersebut diatas dapat disetujui dan perlu ditetapkan dengan surat keputusan
Mengingat : 1. Undang-undang Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional.
2. Peraturan Pemerintah Nomor 17 tahun 2010 tentang pengelolaan dan penyelenggara pendidikan.
3. Keputusan Rektor USU Nomor 03/UN5.1.R/SK/SPB/2021 tentang Peraturan Akademik Program Sarjana Universitas Sumatera Utara.
4. Surat Keputusan Rektor USU Nomor 1876/UN5.1.R/SK/SDM/2021 tentang pengangkatan Dekan Fasilkom-TI USU Periode 2021-2026

MEMUTUSKAN

Menetapkan :
Pertama : Membentuk dan mengangkat Tim Penguji Skripsi mahasiswa sebagai berikut:
Ketua : Sarah Purnamawati ST., MSc.
NIP: 198302262010122003
Sekretaris : Annisa Fadhillah Puhungan S.Kom, M.Kom
NIP: 199308092020012001
Anggota Penguji : Dr. Erna Budhiarti Nababan M.IT
NIP: 196210262017042001
Anggota Penguji : Dedy Arisandi ST., M.Kom.
NIP: 197908312009121002
Moderator : -
Panitera : -
Kedua : Segala biaya yang diperlukan untuk pelaksanaan kegiatan ini dibebankan pada Dana Penerimaan Bukan Pajak (PNPB) Fasilkom-TI USU Tahun 2024.
Ketiga : Keputusan ini berlaku sejak tanggal ditetapkan dengan ketentuan bahwa segala sesuatunya akan diperbaiki sebagaimana mestinya apabila dikemudian hari terdapat kekeliruan dalam surat keputusan ini.

Tembusan :
1. Ketua Program Studi Sarjana (S-1) Teknologi Informasi
2. Yang bersangkutan
3. Arsip

Medan
Ditandatangani secara elektronik oleh:
Dekan



Maya Silvi Lydia
NIP 197401272002122001