BAB I

PENDAHULUAN

1.1 Latar Belakang

Pertanian modern semakin mengintegrasikan teknologi ke dalam praktiknya untuk meningkatkan produktivitas dan efisiensi. Salah satu aspek penting dalam pertanian adalah pemantauan kesehatan tanaman dan deteksi dini penyakit. Pada masa lalu, identifikasi penyakit tanaman melibatkan proses manual dan keterlibatan sistem pakar, tetapi tingkat efisiensi dan akurasi dalam diagnosis sangat terbatas, menghambat kemampuan pemantauan tanaman secara real-time. Deteksi dini penyakit tanaman pada tahap awal infeksi memiliki peran krusial dalam mewujudkan pencegahan yang efektif. Namun, sering kali sulit untuk mengidentifikasi penyakit dengan cepat dan tepat pada tanaman. Strawberry (Fragaria sp.) menjadi salah satu buah yang memiliki signifikansi penting di tingkat global, terutama di negara-negara yang memiliki iklim subtropis. Dengan kemajuan dalam bidang ilmu dan teknologi pertanian, minat terhadap pengembangan tanaman strawberry di wilayah iklim tropis semakin berkembang (Hassan dkk, 2023). Untuk mencapai budidaya Strawberry yang sukses, pengidentifikasian penyakit secara akurat dan respons cepat menjadi hal yang sangat penting. Strawberry bukan hanya memiliki nilai gizi yang tinggi tetapi juga memberikan nilai ekonomi yang signifikan. Dengan meningkatnya permintaan pasar terhadap Strawberry, luas budidaya Strawberry pun semakin meluas. Namun, tantangan utama yang dihadapi dalam budidaya Strawberry adalah penyakit-penyakit yang dapat menyebabkan kerugian besar. Bakteri, jamur, dan patogen lainnya menjadi penyebab utama penyakit Strawberry, dan mereka biasanya menyerang tanaman melalui daun, akar, dan batang.

Badan Pusat Statistik Nasional melaporkan bahwa antara tahun 2012 hingga 2017 produksi Strawberry nasional mengalami penurunan signifikan sekitar 75%. Pada tahun 2012 total produksi Strawberry Indonesia mencapai 169.796 ton tetapi, dalam beberapa tahun berikutnya produksinya terus mengalami penurunan hingga mencapai sekitar 12.225 ton pada tahun 2017. Penurunan produksi yang mencapai angka tersebut dipicu oleh serangan hama dan penyakit pada tahun 2014 yang menyebabkan produksi Strawberry turun drastis menjadi hanya sekitar 58.882 ton (Alif Violeta Efrilla dkk, 2020). Penyakit yang disebabkan oleh jamur, virus maupun bakteri dapat meningkatkan kehilangan hasil (yield loss) pada tanaman yang terinfeksi hingga 95% pada strawberry (I Gusti et al. 2018). Secara umum, pemahaman mengenai penyakit tanaman Strawberry dan pengelompokkannya umumnya bergantung pada observasi manual. Kelemahan ini dapat mengakibatkan hasil identifikasi yang tidak konsisten karena perbedaan persepsi visual manusia. Oleh karena itu, diperlukan teknik identifikasi yang melibatkan sensor kamera dan komputer untuk memastikan hasil yang lebih akurat dan konsisten. Identifikasi penyakit pada tanaman Strawberry menjadi sangat penting untuk mendeteksi dini adanya penyakit, sehingga tindakan pencegahan dapat diambil sebelum penyakit menyebar dan menyebabkan kerugian, seperti yang terjadi pada tahun 2014. Salah

satu metode utama yang digunakan adalah melalui pengamatan tanpa bantuan alat. Oleh karena itu, terkadang gejala penyakit, seperti bintik-bintik dan karat pada daun, dapat meluas sebelum tindakan penanganan dapat dilakukan (Sonata, 2020).

Penelitian yang dilakukan oleh Anwar pada tahun 2023 dengan judul "Klasifikasi Penyakit Tanaman Cabai Rawit Dilengkapi Dengan Segmentasi Citra Daun Dan Buah Menggunakan YOLOV7" menggunakan algoritma YOLO untuk membangun model deteksi penyakit tanaman mendapatkan mAP sebesar 97% (Anwar dkk, 2023). Penelitian lain yang juga menggunakan algoritma YOLO untuk membangun model deteksi penyakit tanaman juga dilakukan oleh Ardiansyah pada tahun 2023 dengan judul "Deteksi dan Klasifikasi Penyakit Pada Daun Kopi Menggunakan YOLOv7" menggunakan algoritma YOLO untuk membangun model deteksi penyakit pada tanaman mendapatkan mAP 95% (Ardiansyah dkk, 2023). Penelitian lain juga menggunakan algoritma YOLO untuk membangun model deteksi penyakit tanaman juga dilakukan oleh Laurenzia Setiana Riva dkk (Laurenzia Setiana Riva dkk, 2023) pada tahun 2023 dengan Judul "Deteksi Penyakit Tanaman Cabai Menggunakan Algoritma YOLOv5 Dengan Variasi Pembagian Data" menggunakan algoritma YOLO untuk membangun model deteksi penyakit tanaman mendapatkan mAP sebesar 94%. Penelitian yang dilakukan oleh Shunlong Chen , Yinghua Liao , Feng Lin, dan Bo Huang pada tahun 2023 dengan judul "An Improved Lightweight YOLOv5 Algorithm For Detecting Strawberry Diseases" menggunakan metode YOLOV5 dan didapatkan hasil Hasil eksperimen pada kumpulan data penyakit Strawberry sumber terbuka menunjukkan bahwa model mencapai presisi rata-rata rata-rata mAP0,5 sebesar 94,7%.

Maka dari itu, penelitian penelitian ini mengembangkan penelitian yang telah dilakukan sebelumnya dengan menggunakan algoritma YOLO (You Only Look Once) untuk mendapatkan akurasi tinggi. Hasil dari pemanfaatan dari YOLO ini dapat menjadi solusi dalam mendeteksi penyakit pada daun strawberry.

1.2 Teori

1.2.1 *Strawberry*

Strawberry (*Fragaria* × *ananassa*) adalah buah berwarna merah dari keluarga *Rosaceae* yang dikenal dengan rasa manisnya dan aroma harum yang khas. Buah *Strawberry* berasal dari Chili, Amerika. *Strawberry* adalah tanaman buah subtropis yang dapat tumbuh di berbagai wilayah geografis, mulai dari daerah tropis hingga daerah Arktik. Meskipun Indonesia merupakan negara tropis, daerah dataran tinggi di Indonesia mampu menghasilkan *Strawberry* yang bernilai ekonomis tinggi. Tanaman *Strawberry* tumbuh optimal di dataran tinggi dengan udara sejuk dan lembab. Secara fisiologis, *Strawberry* membutuhkan suhu antara 17-20°C, kelembapan 80-90%, penyinaran matahari 8-10 jam per hari, dan curah hujan antara 600-700 mm per tahun (Rahadatul et al., 2024). Tampilan buah *strawberry* ditunjukkan pada Gambar 1.

Gambar 1. Buah *Strawberry*

Strawberry (*Fragaria sp*.) memiliki berbagai manfaat kesehatan yang signifikan. Buah ini kaya akan vitamin C, kalium, serat, dan folat, serta rendah kalori. Selain itu, *Strawberry* juga mengandung asam ellagic yang berfungsi sebagai antioksidan, membantu melindungi tubuh dari kerusakan sel dan mengurangi risiko penyakit kronis seperti kanker dan penyakit jantung (Pratiwi et al., 2024). Meskipun *Strawberry* memiliki banyak manfaat kesehatan seperti kandungan vitamin C, serat, dan antioksidan yang tinggi, tanaman ini juga rentan terhadap berbagai penyakit. Penyakit seperti bercak daun, *blossom blight*, *powdery mildew*, dan *gray mold* sering menyerang daun, bunga, dan buah *Strawberry*, mengakibatkan penurunan kualitas dan kuantitas panen. Kerentanan ini menimbulkan tantangan besar bagi petani, terutama ketika kurangnya pengetahuan dan sumber daya untuk identifikasi dan pengelolaan penyakit secara cepat dan tepat dapat menyebabkan gagal panen yang signifikan (Sarfina, 2024).

1.2.2 Penyakit Daun *Strawberry*

Meskipun *Strawberry* memiliki banyak manfaat, tanaman ini rentan terhadap berbagai penyakit yang menyerang daun dan buahnya, termasuk bercak daun, hawar daun, dan busuk buah.

1. Bercak Daun

   Bercak daun pada *Strawberry* adalah penyakit yang disebabkan oleh jamur Bercak daun pada *Strawberry* disebabkan oleh jamur *Neopestalotiopsis sp*. Penyakit ini ditandai dengan munculnya bintik-bintik bulat berwarna coklat pada daun yang dapat menyebar dan menyebabkan daun menguning serta gugur. Kondisi ini mengurangi kemampuan fotosintesis daun dan melemahkan tanaman, yang akhirnya dapat menurunkan hasil panen. Pengendalian bercak daun memerlukan tindakan seperti rotasi tanaman, penggunaan varietas tahan penyakit, dan aplikasi fungisida yang sesuai (Fitri, 2022). Tampilan contoh bercak daun pada *strawberry* ditunjukkan pada Gambar 2 berikut.

Gambar 2. Contoh Bercak Daun pada *Strawberry*

2. Hawar Daun

   Hawar daun pada *Strawberry*, juga dikenal sebagai *Angular Leaf Spot*, disebabkan oleh bakteri *Xanthomonas fragariae*. Penyakit ini biasanya dimulai dengan munculnya bintik-bintik kecil, berair, dan berwarna gelap pada daun. Seiring waktu, bintik-bintik ini dapat membesar dan bergabung, menyebabkan daun menjadi kering dan mati. Gejala khas lainnya termasuk adanya lesi berbentuk sudut yang dikelilingi oleh warna kuning atau coklat. Hawar daun mengganggu fotosintesis dan kesehatan tanaman secara keseluruhan, yang dapat berdampak signifikan pada hasil panen (EFRILLA, 2019). Tampilan penyakit bercak yang menjangkit daun Strawberry ditunjukkan pada Gambar 3.

Gambar 3. Contoh Hawar Daun pada *Strawberry*

3. Busuk Buah

   Busuk buah pada *Strawberry* sering disebabkan oleh jamur *Botrytis Cinerea*, yang dikenal sebagai *gray mold*. Penyakit ini berkembang dalam kondisi lembap dan basah, dimulai dengan munculnya bintik-bintik kecil berwarna coklat pada buah. Seiring waktu, bintik-bintik ini membesar dan membentuk lapisan abu-abu berbulu yang menutupi permukaan buah sehingga membuatnya lunak, berair, dan tidak layak konsumsi. Gambar 4 menunjukkan penyakit busuk buah yang menjangkit buah *strawberry*.

Gambar 4. Contoh Busuk Buah pada *Strawberry*

1.2.3 *Artificial Intelligence*

Artificial Intelligence (AI) adalah cabang ilmu komputer yang berfokus pada pengembangan sistem yang dapat melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia. Ini mencakup kemampuan untuk belajar dari pengalaman, mengenali pola, membuat keputusan, dan memahami bahasa alami. AI dapat dibagi menjadi beberapa subbidang utama, termasuk pembelajaran mesin (*machine learning*), pemrosesan bahasa alami (natural language processing), visi komputer (*computer vision*), dan sistem berbasis pengetahuan (Sheikh et al., 2023). Konsep kecerdasan buatan telah ada sejak pertengahan abad ke-20, pada tahun 1950 Alan Turing mengajukan pertanyaan "Bisakah mesin berpikir?" dan memperkenalkan tes turing sebagai cara untuk mengukur kemampuan mesin dalam meniru kecerdasan manusia. Pada tahun 1956, istilah "Artificial Intelligence" pertama kali digunakan oleh John McCarthy dalam konferensi Dartmouth, yang dianggap sebagai kelahiran resmi AI sebagai bidang penelitian (Adami, 2021).

Meskipun AI menawarkan banyak manfaat, ada juga tantangan yang perlu diatasi, seperti etika AI, bias dalam data dan algoritma, serta dampak sosial dan ekonomi dari otomatisasi. Penelitian dan pengembangan terus berlanjut untuk membuat AI lebih transparan, adil, dan dapat dipercaya. Masa depan AI terlihat menjanjikan dengan potensi untuk mengubah berbagai aspek kehidupan manusia, membuatnya lebih efisien dan cerdas.

1.2.4 Visi Komputer

Visi komputer adalah teknologi dalam kecerdasan buatan yang memungkinkan komputer untuk memahami dan menafsirkan gambar serta video. Dalam sektor pertanian, visi komputer digunakan untuk berbagai aplikasi seperti identifikasi penyakit tanaman, pemantauan pertumbuhan tanaman, dan penilaian kualitas buah. Teknologi ini membantu petani dalam membuat keputusan yang lebih cepat dan tepat, mengurangi ketergantungan pada ahli, serta meningkatkan efisiensi dan hasil panen (H. Tian et al., 2019).

Teknologi visi komputer menawarkan solusi inovatif untuk mengatasi berbagai tantangan dalam pertanian modern. Dengan menggunakan algoritma pembelajaran mesin dan jaringan saraf tiruan, sistem visi komputer dapat menganalisis citra

tanaman secara real-time untuk mendeteksi gejala awal penyakit, memantau kondisi tanaman, dan mengoptimalkan penggunaan sumber daya seperti air dan pupuk (Md Akbar et al., 2024). Penggunaan drone dan sensor kamera memungkinkan pemantauan area pertanian yang luas dengan cepat dan akurat. Implementasi teknologi ini tidak hanya meningkatkan produktivitas tetapi juga berkontribusi pada keberlanjutan pertanian dengan mengurangi penggunaan bahan kimia dan meminimalkan kerugian akibat penyakit tanaman (Ghadge et al., 2021).

1. Tantangan dan Solusi Visi Komputer dalam Pertanian
   Meskipun teknologi visi komputer menawarkan berbagai manfaat, penerapannya dalam pertanian tidak lepas dari tantangan. Salah satu tantangan utama adalah variabilitas kondisi lapangan, seperti perubahan cuaca, variasi pencahayaan, dan keragaman jenis tanaman. Faktor-faktor ini dapat mempengaruhi akurasi dan konsistensi sistem visi komputer dalam mendeteksi dan mengklasifikasikan kondisi tanaman. Selain itu, kebutuhan akan data pelatihan yang besar dan berkualitas tinggi menjadi kendala tersendiri. Pengumpulan dan anotasi data yang representatif dari berbagai kondisi lapangan memerlukan waktu dan sumber daya yang signifikan.
   Untuk mengatasi tantangan ini, penelitian dan pengembangan terus dilakukan untuk meningkatkan robustitas dan akurasi algoritma visi komputer. Salah satu pendekatan yang menjanjikan adalah penggunaan data augmentasi dan transfer learning yang memungkinkan model belajar dari dataset yang lebih kecil namun tetap efektif. Data augmentasi membantu memperluas variasi dataset dengan menghasilkan variasi gambar dari data yang ada, sementara transfer learning memungkinkan penggunaan model yang telah dilatih pada tugas serupa untuk mempercepat pelatihan model baru. Selain itu, integrasi teknologi sensor dan drone yang semakin canggih juga membantu mengurangi variabilitas kondisi lapangan dengan menyediakan data yang lebih konsisten dan berkualitas tinggi.
2. Implementasi dan Dampak Jangka Panjang
   Implementasi visi komputer dalam pertanian memiliki potensi untuk membawa perubahan signifikan dalam praktik pertanian. Dengan kemampuan untuk memantau tanaman secara *real-time*, petani dapat mengambil tindakan preventif lebih awal, mengurangi kerugian akibat penyakit dan hama. Teknologi ini juga memungkinkan penggunaan sumber daya yang lebih efisien, seperti air dan pupuk, dengan memantau kebutuhan spesifik setiap tanaman. Hal ini tidak hanya meningkatkan hasil panen tetapi juga mendukung praktik pertanian yang lebih berkelanjutan dengan mengurangi dampak lingkungan.

1.2.5 Deteksi Objek

Pendeteksian objek bertujuan untuk mengidentifikasi keberadaan serta lokasi suatu objek dalam gambar, beserta cakupan area yang ditempatinya. Dengan menggunakan metode pengenalan objek, pengguna dapat mengklasifikasikan objek ke dalam berbagai kategori berdasarkan data latihan (*training*). Proses deteksi objek umumnya dimulai dengan langkah pengenalan awal terhadap objek. Prinsip ini bisa

dianggap sebagai pengenalan objek dalam dua kelas, di mana satu kelas merepresentasikan objek yang dicari dan kelas lainnya mewakili benda bukan objek yang dicari. Langkah berikutnya dalam deteksi objek melibatkan pencarian setiap bagian dari gambar yang cocok dengan objek target, berdasarkan pengetahuan yang diambil dari proses training (Redmon et al., 2016).

1. Contoh Aplikasi *Deep Learning*
   Berikut adalah contoh-contoh aplikasi deep learning yang telah diterapkan dalam berbagai bidang:
   a. Pengenalan Gambar:
      Convolutional Neural Networks (CNNs): CNN digunakan secara luas dalam tugas-tugas pengenalan gambar. Model seperti AlexNet, VGGNet, dan ResNet telah menunjukkan performa tinggi dalam klasifikasi gambar dan deteksi objek. CNN bekerja dengan memproses gambar melalui beberapa lapisan konvolusi dan pooling untuk mengekstrak fitur-fitur penting.
   b. Pemrosesan Bahasa Alami (NLP):
      Recurrent Neural Networks (RNNs) dan *Transformer*: Dalam NLP, model seperti RNN dan LSTM (*Long Short-Term Memory*) digunakan untuk tugas-tugas seperti analisis sentimen, penerjemahan bahasa, dan pengenalan ucapan. Baru-baru ini, model transformer seperti BERT dan GPT telah menjadi standar dalam pemrosesan bahasa alami, menawarkan peningkatan signifikan dalam akurasi dan efisiensi.
   c. Deteksi dan Segmentasi Objek:
      YOLO (*You Only Look Once*) dan Mask R-CNN: Model-model ini digunakan untuk mendeteksi dan mengsegmentasi objek dalam gambar dan video secara real-time. *YOLO* terkenal karena kecepatan dan efisiensinya, sedangkan Mask R-CNN menawarkan kemampuan segmentasi objek yang lebih presisi.
   d. Kendaraan Otonom:
      Jaringan Saraf dalam Sistem ADAS: *Deep learning* digunakan dalam kendaraan otonom untuk tugas-tugas seperti deteksi jalur, pengenalan rambu lalu lintas, dan deteksi pejalan kaki. Kombinasi berbagai model deep learning membantu kendaraan untuk memahami dan menavigasi lingkungan gambar-gambar dengan aman.
2. Teknik dan Algoritma Deteksi Objek
   Deteksi objek melibatkan berbagai teknik dan algoritma yang memungkinkan sistem untuk mengenali dan melokalisasi objek dalam gambar atau video. Beberapa teknik dan algoritma populer yang digunakan dalam deteksi objek antara lain:
   a. *Convolutional Neural Networks* (CNNs):
      CNN adalah jaringan saraf tiruan yang dirancang khusus untuk analisis gambar. CNN bekerja dengan mengekstraksi fitur-fitur dari gambar melalui beberapa lapisan konvolusi dan pooling. Beberapa arsitektur

CNN populer untuk deteksi objek termasuk AlexNet, VGGNet, dan ResNet.

b. *YOLO* (*You Only Look Once*):
YOLO adalah algoritma deteksi objek real-time yang memproses seluruh gambar sekaligus untuk mendeteksi dan mengklasifikasikan objek. Keunggulan *YOLO* adalah kecepatannya, memungkinkan deteksi objek dalam waktu nyata.

1.2.6 Python

Python adalah bahasa pemrograman tingkat tinggi yang dikenal karena kesederhanaan dan fleksibilitasnya. Diciptakan oleh Guido van Rossum, Python dirancang untuk mudah dipahami dan digunakan, menjadikannya pilihan yang sangat baik bagi pemula maupun pengembang berpengalaman. Python mendukung berbagai paradigma pemrograman termasuk pemrograman berorientasi objek, pemrograman fungsional, dan pemrograman prosedural yang memungkinkan pengembangan aplikasi yang beragam (Srinath, 2017).

Python sangat populer dalam berbagai bidang aplikasi, seperti pengembangan web, analisis data, kecerdasan buatan, dan otomatisasi. Pustaka-pustaka seperti Django dan Flask digunakan untuk pengembangan web, sementara Pandas dan NumPy sangat berguna dalam analisis data. Di bidang kecerdasan buatan, pustaka seperti TensorFlow dan PyTorch menyediakan alat yang kuat untuk pengembangan dan implementasi model pembelajaran mesin. Python juga dikenal karena kemampuannya untuk berintegrasi dengan bahasa pemrograman lain dan menggunakan modul-modul yang dirancang dalam bahasa lain, sehingga meningkatkan fleksibilitas dan daya gunanya. Keberhasilan Python juga didukung oleh komunitasnya yang besar dan aktif, yang terus berkontribusi dengan pustaka dan alat baru, menjadikan Python salah satu bahasa pemrograman paling dinamis dan berkembang pesat di dunia teknologi. Python tidak hanya memfasilitasi pembelajaran pemrograman tetapi juga memungkinkan pengembangan solusi inovatif yang dapat mengatasi berbagai tantangan teknologi modern (Sundnes, 2020).

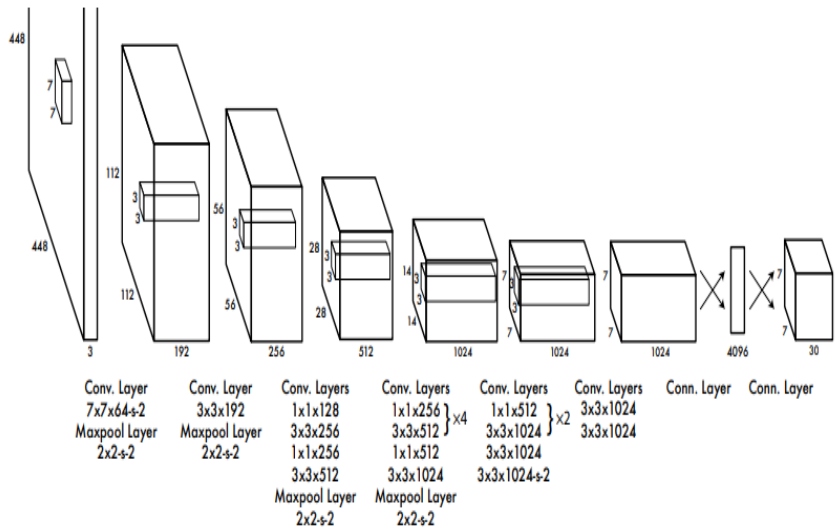
Selain itu, Python juga digunakan dalam pengembangan perangkat lunak ilmiah dan numerik. Pustaka seperti SciPy dan Matplotlib sangat berguna dalam analisis ilmiah dan visualisasi data. SciPy menyediakan berbagai fungsi matematika, ilmiah, dan teknik, sementara Matplotlib digunakan untuk membuat visualisasi yang kompleks dan interaktif. Di bidang pengembangan game, pustaka seperti Pygame memungkinkan pengembang untuk membuat game sederhana hingga kompleks dengan mudah. Python juga digunakan dalam pengembangan aplikasi desktop dengan pustaka seperti Tkinter, yang menyediakan antarmuka pengguna grafis. Kombinasi dari kemudahan penggunaan, pustaka yang luas, dan komunitas yang aktif menjadikan Python sebagai salah satu bahasa pemrograman paling serbaguna dan berpengaruh di dunia teknologi saat ini.

1.2.7 *Deep Learning*

Deep learning adalah subbidang dari pembelajaran mesin (machine learning) yang menggunakan jaringan saraf tiruan (artificial neural networks) dengan banyak lapisan (deep neural networks) untuk mempelajari representasi data yang kompleks. Jaringan saraf tiruan diilhami oleh cara kerja otak manusia, di mana neuron-neuron berinteraksi satu sama lain untuk memproses informasi. Konsep jaringan saraf tiruan pertama kali diperkenalkan pada tahun 1940-an oleh Warren McCulloch dan Walter Pitts, yang mengembangkan model matematis dari neuron buatan. Namun, kemajuan signifikan dalam deep learning baru terjadi beberapa dekade kemudian. Pada tahun 1980-an, Geoffrey Hinton dan rekannya memperkenalkan backpropagation, sebuah algoritma yang memungkinkan pelatihan jaringan saraf berlapis-lapis secara lebih efektif. Kemajuan dalam komputasi dan ketersediaan data dalam jumlah besar pada awal 2000-an membuka jalan bagi deep learning untuk mencapai hasil yang mengesankan dalam berbagai tugas, seperti pengenalan gambar dan pemrosesan bahasa alami (Talaei Khoei et al., 2023). Tonggak penting adalah kemenangan AlexNet pada kompetisi ImageNet 2012, yang menunjukkan keunggulan deep learning dalam pengenalan gambar (Alom et al., 2018). Aplikasi deep learning meliputi pengenalan gambar dengan Convolutional Neural Networks (CNNs), pemrosesan bahasa alami dengan Recurrent Neural Networks (RNNs) dan Transformers, serta deteksi objek dengan YOLO dan Mask R-CNN.

1.2.8 *YOLO* (*You Only Look Once*)

YOLO (*You Only Look Once*) adalah algoritma deteksi objek yang dikembangkan oleh Joseph Redmon dan timnya. *YOLO* pertama kali diperkenalkan pada tahun 2015 melalui makalah berjudul "*You Only Look Once*: *Unified, Real-Time Object Detection*". Ide dasar di balik *YOLO* adalah mengubah masalah deteksi objek menjadi masalah regresi langsung, yang memungkinkan sistem untuk memprediksi *bounding boxes* dan probabilitas kelas secara simultan dari gambar penuh. Pendekatan ini berbeda dari metode deteksi objek sebelumnya yang menggunakan proses berbasis *class*ifier atau region proposal (Redmon et al., 2016). Arsitektur YOLO ditunjukkan pada Gambar 5.



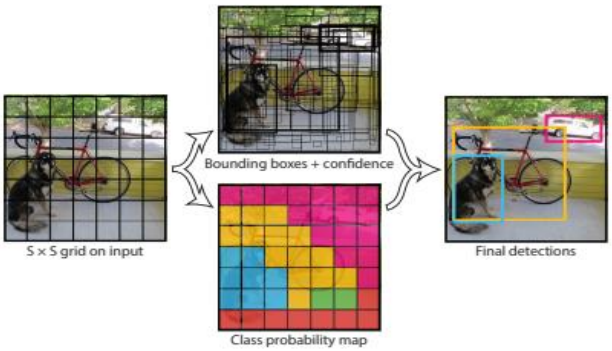
Gambar 5. Arsitektur YOLO

Sumber:(Joseph Redmon dkk, 2015)

YOLO (*You Only Look Once*) menyatukan komponen-komponen terpisah dari deteksi objek menjadi satu jaringan saraf tunggal. Jaringan ini menggunakan fitur dari seluruh gambar untuk memprediksi setiap kotak pembatas. Selain itu, jaringan ini memprediksi semua kotak pembatas di seluruh kelas untuk gambar secara bersamaan. Ini berarti jaringan *YOLO* mempertimbangkan keseluruhan gambar dan semua objek dalam gambar secara global saat membuat prediksi. Desain *YOLO* memungkinkan pelatihan *end-to-end* dan kecepatan real-time sambil mempertahankan presisi rata-rata yang tinggi. Sistem *YOLO* membagi gambar input menjadi grid dengan ukuran S × S. Jika pusat objek jatuh ke dalam sel grid, maka sel grid tersebut bertanggung jawab untuk mendeteksi objek itu (Redmon et al., 2016). Setiap sel grid memprediksi B kotak pembatas dan skor kepercayaan untuk kotak-kotak tersebut. Skor kepercayaan ini mencerminkan seberapa yakin model bahwa kotak tersebut berisi objek dan juga seberapa akurat model memprediksi kotak tersebut. Secara formal, kepercayaan didefinisikan sebagai Pr(Object)
∗ IOUtruth/pred.

Jika tidak ada objek dalam sel tersebut, skor kepercayaan harus nol. Jika ada objek, skor kepercayaan seharusnya sama dengan *intersection over union* (IOU) antara kotak yang diprediksi dan ground truth. Setiap kotak pembatas terdiri dari 5 prediksi: x, y, w, h, dan *confidence*. Koordinat (x, y) merepresentasikan pusat kotak relatif terhadap batas-batas sel *grid*. Lebar dan tinggi diprediksi relatif terhadap keseluruhan gambar. Prediksi kepercayaan merepresentasikan *IOU* antara kotak yang diprediksi dan kotak ground truth. Selain itu, setiap sel grid juga memprediksi probabilitas kelas kondisional, Pr(*Classi*|*Object*). Probabilitas ini dikondisikan pada

sel grid yang mengandung objek. Pada saat pengujian, kita mengalikan probabilitas kelas kondisional dan prediksi kepercayaan kotak individu Pr(*Classi* | *Object*) × Pr(*Object*) × *IOU(truth/pred)* = Pr(*Classi*) × *IOU(truth/pred)* yang memberikan kita skor kepercayaan spesifik kelas untuk setiap kotak. Skor ini mencakup baik probabilitas bahwa kelas tersebut muncul dalam kotak dan seberapa baik kotak yang diprediksi sesuai dengan objek. Cara kerja sederhana dari algoritma *YOLO* ditunjukkan pada Gambar 6.



Gambar 6. Cara Kerja Sederhana algoritma *YOLO*

Sumber:(Joseph Redmon dkk, 2015)

Gambar 6 menjelaskan model YOLO (You Only Look Once) dan bagaimana cara kerjanya dalam mendeteksi objek dalam sebuah gambar. Sistem ini memodelkan deteksi sebagai masalah regresi. YOLO membagi gambar input menjadi grid berukuran S × S. Setiap sel grid bertanggung jawab untuk mendeteksi objek yang pusatnya berada di dalam sel tersebut. Setiap sel grid memprediksi B bounding boxes, confidence untuk kotak-kotak tersebut, serta probabilitas kelas C. Prediksi ini dikodekan sebagai tensor S × S × (B × 5 + C). Dalam evaluasi YOLO pada dataset PASCAL VOC, digunakan S = 7, B = 2, dan PASCAL VOC memiliki 20 kelas berlabel sehingga C = 20. Prediksi akhir dinyatakan sebagai tensor berukuran 7 × 7 × 30. Setiap sel grid menghasilkan 2 bounding boxes, masing-masing dengan 5 parameter (x, y, w, h, dan confidence), ditambah dengan 20 probabilitas kelas, menghasilkan total 30 nilai per sel grid. YOLO menggabungkan prediksi ini untuk menghasilkan deteksi objek akhir yang akurat dan cepat dalam gambar atau video real-time (Redmon et al., 2016). Gambar 7 menampilkan timeline perjalanan versi YOLO.

Gambar 7. Sejarah Perkembangan Versi *YOLO*

Sumber: (Juan Terven dkk, 2023)

Gambar 7 menampilkan timeline perjalanan versi *YOLO*. Sejarah perkembangan *YOLO (You Only Look Once)* dimulai pada tahun 2015 ketika *YOLOv1* diperkenalkan oleh Joseph Redmon. *YOLO* merupakan terobosan dalam deteksi objek secara real-time karena pendekatan satu tahapannya yang mendeteksi dan mengklasifikasikan objek dalam satu pass dari jaringan neural. Ini berbeda dengan pendekatan dua tahap seperti *R-CNN* yang lebih lambat. Setelah itu, *YOLOv2* (dikenal juga sebagai *YOLO*9000) memperkenalkan konsep anchor box dan backbone yang lebih efisien, yaitu Darknet-19, yang meningkatkan akurasi deteksi. *YOLOv3*, yang diperkenalkan pada 2018, membawa peningkatan signifikan dengan penggunaan backbone Darknet-53 dan berbagai teknik augmentasi gambar. *YOLOv4* dan *YOLOv5* menambahkan berbagai peningkatan pada akurasi dan efisiensi, seperti penambahan fitur backbone CSP dan *PA-NET* di *YOLOv5*, yang membuatnya lebih mudah digunakan untuk transfer learning. Versi-versi terbaru, seperti *YOLOv8* yang dirilis pada tahun 2023, terus meningkatkan performa dalam hal akurasi dan kecepatan, menjadikannya salah satu framework deteksi objek paling populer saat ini (Terven & Cordova-Esparza, 2023).

1. Model Evaluasi *YOLO*

Berikut adalah beberapa model evaluasi yang digunakan oleh Algoritma *YOLO*:

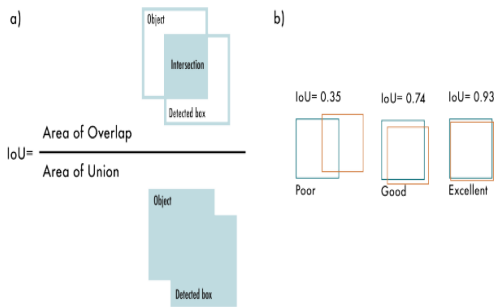
a. *Average Precision* (AP): *Average Precision*, sering kali disebut sebagai Mean *Average Precision* (mAP), adalah metrik yang umum digunakan untuk mengevaluasi kinerja model deteksi objek. AP mengukur presisi rata-rata di semua kategori, memberikan satu nilai untuk membandingkan berbagai model. Dataset COCO tidak membedakan antara AP dan mAP; dalam makalah ini, metrik ini akan disebut sebagai AP. Pada *YOLO*v1 dan *YOLO*v2, dataset yang digunakan untuk pelatihan dan *benchmarking* adalah PASCAL VOC 2007 dan VOC 2012. Namun, dari *YOLO*v3 dan seterusnya, dataset yang digunakan adalah Microsoft COCO (*Common Objects in Context*). AP dihitung dengan cara

yang berbeda untuk kedua dataset ini (Terven & Cordova-Esparza, 2023).

b. *Precision* dan *Recall*: *Precision* mengukur akurasi prediksi positif dari model, sedangkan recall mengukur proporsi kasus positif sebenarnya yang berhasil diidentifikasi oleh model. Ada *trade-off* antara *precision* dan *recall*; misalnya, meningkatkan jumlah objek yang terdeteksi (*recall* lebih tinggi) dapat menghasilkan lebih banyak *false positives* (*precision* lebih rendah). Untuk mengatasi *trade-off* ini, metrik AP menggabungkan *precision-recall curve* yang menggambarkan *precision* terhadap *recall* untuk berbagai ambang kepercayaan. Metrik ini memberikan penilaian seimbang antara *precision* dan *recall* dengan mempertimbangkan area di bawah *precision-recall curve* ini (Terven & Cordova-Esparza, 2023).

c. *Handling Multiple Object Categories*: Model deteksi objek harus mengidentifikasi dan menentukan lokasi beberapa kategori objek dalam gambar. Metrik AP menangani hal ini dengan menghitung *Average Precision* (AP) untuk setiap kategori secara terpisah dan kemudian mengambil rata-rata dari AP tersebut di semua kategori. Pendekatan ini memastikan bahwa kinerja model dievaluasi untuk setiap kategori secara individu, memberikan penilaian yang lebih komprehensif terhadap kinerja keseluruhan mode ini (Terven & Cordova-Esparza, 2023).

d. *Intersection over Union* (IoU): Deteksi objek bertujuan untuk secara akurat menentukan lokasi objek dalam gambar dengan memprediksi *bounding boxes. Metrik* AP menggabungkan ukuran *Intersection over Union* (IoU) untuk menilai kualitas *bounding boxes* yang diprediksi. IoU adalah rasio antara area perpotongan dan area gabungan dari *bounding box* yang diprediksi dan *bounding box ground truth* (Imran et al., 2024). Ini mengukur tumpang tindih antara *bounding box ground truth* dan yang diprediksi. *Benchmark* COCO mempertimbangkan beberapa ambang IoU untuk mengevaluasi kinerja model pada berbagai tingkat akurasi Lokasi ini (Terven & Cordova-Esparza, 2023).



Gambar 8. Konsep *Intersection over Union* (IoU) (a) cara menghitung IoU dengan rumus IoU = *Area of Overlap* / *Area of Union*. (b) memberikan contoh visual dari tiga tingkat kualitas *bounding boxes* berdasarkan nilai IoU

Sumber: (Amna Imran, 2024)

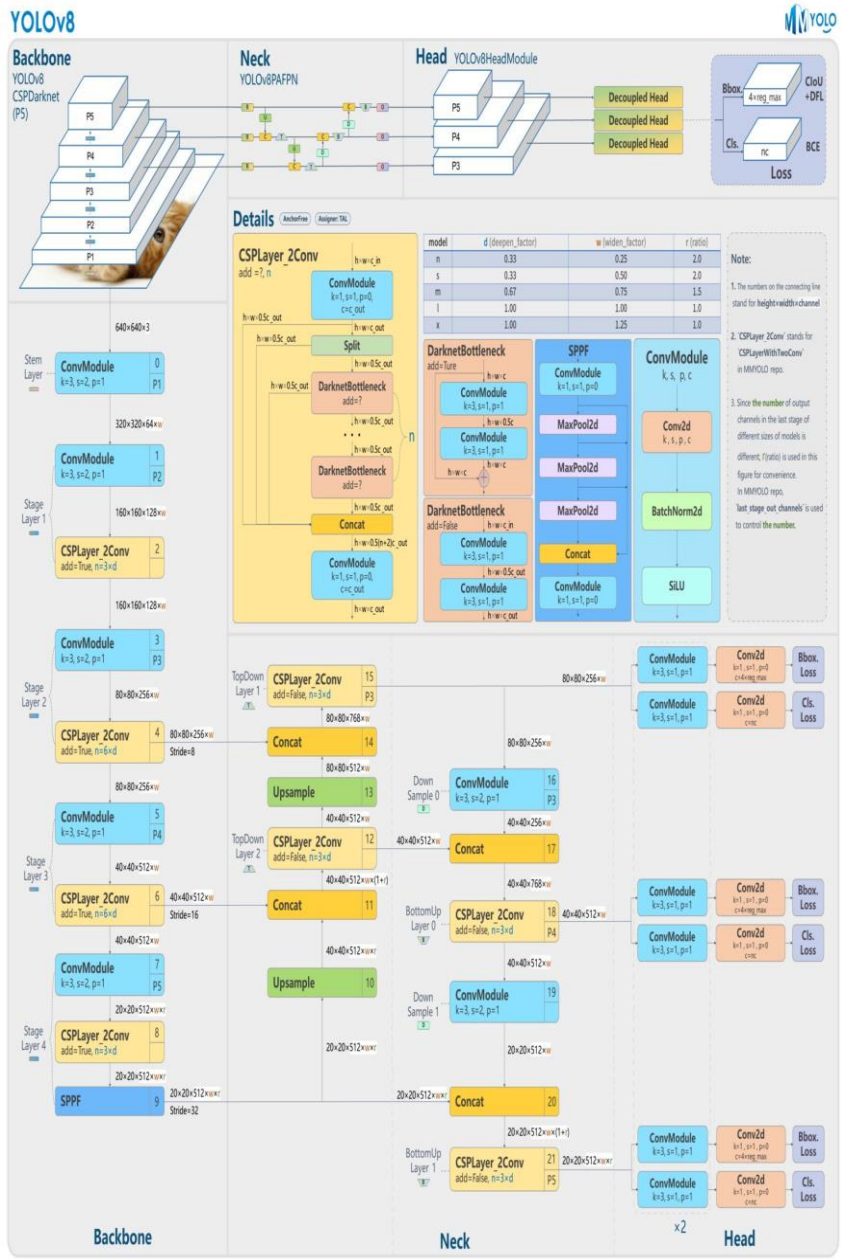
Gambar 8 menunjukkan konsep dari IoU. Intersection over Union (IoU) adalah metrik yang digunakan untuk mengevaluasi akurasi deteksi objek dalam tugas visi komputer. Pada bagian (a), IoU dihitung dengan membandingkan area tumpang tindih (intersection) antara kotak deteksi yang diprediksi dan kotak ground truth (representasi objek sebenarnya), dengan area gabungan dari kedua kotak tersebut (*union*). Semakin besar tumpang tindih antara kotak prediksi dan objek asli, semakin tinggi nilai IoU-nya, yang menunjukkan akurasi yang lebih baik (Imran et al., 2024). Bagian (b) memberikan contoh nilai IoU pada berbagai tingkat akurasi:

- IoU = 0.35 (*Poor*): Menunjukkan bahwa kotak prediksi dan objek sebenarnya hanya sedikit tumpang tindih, yang mengindikasikan prediksi yang buruk.
- IoU = 0.74 (*Good*): Ada tumpang tindih yang lebih signifikan antara kotak prediksi dan objek sebenarnya, menunjukkan prediksi yang cukup baik.
- IoU = 0.93 (*Excellent*): Kotak prediksi hampir sempurna mencocokkan posisi objek sebenarnya, yang menunjukkan akurasi deteksi yang sangat baik.

  Secara keseluruhan, IoU adalah alat penting untuk menilai kualitas prediksi bounding box dalam model deteksi objek, dengan nilai IoU yang lebih tinggi berarti hasil yang lebih akurat (Imran et al., 2024).

2. *YOLOv8*

   *YOLO*v8, atau *You Only Look Once* versi 8, adalah model deteksi objek mutakhir yang dibangun di atas kesuksesan versi sebelumnya. Dikenal karena efisiensi dan akurasinya dalam tugas deteksi objek *real-time*, *YOLO*v8 mengatasi keterbatasan versi sebelumnya sambil mempertahankan keseimbangan antara kecepatan dan presisi.



Gambar 9. Arsitektur Model *YOLO*v8

Sumber: (Gaudenz Boesch, 2023)

Gambar 9 menampilkan arsitektur model *YOLO*v8. Arsitektur *YOLO*v8 terdiri dari tiga komponen utama: *backbone*, *Neck*, dan *head*. *Backbone* bertanggung jawab untuk ekstraksi fitur dari gambar input, dengan menggunakan CSPDarknet53 atau EfficientDet. *Neck* menghubungkan *backbone* dengan *head*, yang penting untuk fusi fitur. *Head* memprediksi

bounding boxes, kelas objek, dan skor kepercayaan. *YOLO*v8 menawarkan varian model yang berbeda, seperti *YOLO*v8-tiny dan *YOLO*v8x, yang bervariasi dalam ukuran dan kompleksitas komputasi, memungkinkan pengguna memilih model yang sesuai dengan kebutuhan gambar-gambar. *YOLO*v8 juga mengintegrasikan strategi pelatihan canggih seperti optimasi *Rectified* Adam (RAdam) dan deteksi objek berbasis anchor atau *anchor-free*, yang berkontribusi pada konvergensi yang lebih cepat selama pelatihan dan peningkatan kinerja. Selain itu, sistem konfigurasi fleksibel memungkinkan penyesuaian parameter seperti ukuran input dan anchor boxes, membuat *YOLO*v8 dapat beradaptasi dengan berbagai dataset dan skenario aplikasi

a. *Backbone Network:*
   Backbone adalah komponen utama *YOLO*v8 yang bertanggung jawab untuk ekstraksi fitur dari gambar input. *YOLO*v8 menggunakan CSPDarknet53 sebagai *backbone*-nya, yang merupakan versi modifikasi dari arsitektur Darknet. CSPDarknet53 mengintegrasikan *Cross Stage Partial (CSP) networks*, yang meningkatkan kapasitas pembelajaran dan efisiensi jaringan dengan mengoptimalkan aliran informasi antara berbagai tahap jaringan. *Backbone YOLO*v8 menggunakan CSPDarknet53 untuk ekstraksi fitur dari gambar input. Terdiri dari beberapa lapisan konvolusi (Conv) dan lapisan C2f (Cross Stage Partial Networks). *Backbone* terdiri dari lima tahapan (P1 hingga P5) yang bertanggung jawab untuk menangkap fitur dari berbagai resolusi:
   - P1-P5: Lapisan konvolusi dengan berbagai ukuran kernel dan stride untuk mengecilkan dimensi gambar dan mengekstraksi fitur hierarkis.
   - SPPF: *Spatial Pyramid Pooling Fast*, digunakan untuk agregasi fitur dengan *pooling spasial*.

b. *Neck Architecture:*
   Neck adalah struktur yang menghubungkan *backbone* dengan *head*. *YOLO*v8 menggunakan *Path Aggregation Network* (PANet) sebagai *Neck*-nya, yang memfasilitasi aliran informasi di berbagai resolusi spasial. PANet memungkinkan model menangkap fitur multi-skala secara efektif, meningkatkan kemampuan model untuk mendeteksi objek dengan berbagai ukuran dalam gambar. *Neck YOLO*v8 menggunakan *Path Aggregation Network* (PANet) untuk fusi fitur dari berbagai skala:
   - *Conv, C2f, Concat, Upsample*: Berfungsi untuk menggabungkan dan meningkatkan resolusi fitur yang diekstraksi dari *backbone*.

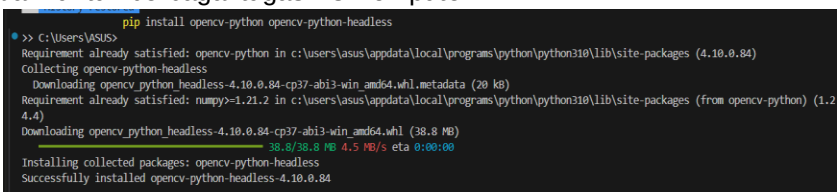
c. YOLO Head:
   YOLO head adalah komponen yang menghasilkan prediksi berdasarkan fitur yang diekstraksi oleh *backbone* dan *Neck*. *YOLO head* memprediksi koordinat *bounding box*, skor objektivitas, dan

probabilitas kelas untuk setiap *anchor box* yang terkait dengan sel *grid*. Penggunaan *anchor boxes* memungkinkan model memprediksi objek dengan berbagai bentuk dan ukuran secara efisien.

- Detection *Head*s: Menghasilkan prediksi berdasarkan fitur yang diproses oleh *backbone* dan *Neck*. Terdiri dari beberapa lapisan konvolusi yang mengekstraksi fitur lebih lanjut dan membuat prediksi deteksi pada berbagai skala (P3, P4, P5).
- *Loss Function*: Digunakan untuk menghitung kesalahan prediksi yang melibatkan *bounding box regression (Bbox), class prediction* (Cls), dan *objectness score* (Obj).

1.2.9 OpenCV

OpenCV (*OpenSource Computer Vision Library*) adalah pustaka *open-source* yang digunakan untuk pemrosesan gambar dan visi komputer. Pustaka ini menyediakan berbagai algoritma yang dapat digunakan untuk tugas-tugas seperti pengenalan wajah, deteksi objek, klasifikasi objek, dan pelacakan Gerakan (Fu'adi, 2024). Kegunaan utama OpenCV mencakup pemrosesan gambar dan video, pengenalan wajah, deteksi objek, dan pelacakan gerakan. Dalam pemrosesan gambar, OpenCV menyediakan berbagai teknik untuk filtrasi, transformasi, dan deteksi tepi, memungkinkan pengguna meningkatkan kualitas gambar sebelum analisis lebih lanjut. Untuk pengenalan wajah, OpenCV menggunakan algoritma seperti *Haar cascades* dan jaringan saraf tiruan untuk mendeteksi dan mengenali wajah dalam berbagai kondisi pencahayaan dan pose. Selain itu, OpenCV memungkinkan deteksi objek secara *real-time* menggunakan algoritma seperti *YOLO* (*You Only Look Once*) dan SSD *(Single Shot MultiBox Detector*), yang banyak digunakan dalam aplikasi keamanan, otomotif, dan robotika. Fungsi pelacakan objek dalam OpenCV, seperti *Optical Flow* dan MeanShift, digunakan dalam pengawasan dan analisis video. OpenCV juga mendukung integrasi dengan pustaka pembelajaran mesin seperti TensorFlow dan PyTorch, memungkinkan penggunaan model pembelajaran mendalam untuk berbagai tugas visi komputer.



Gambar 10. Contoh Kode Untuk Instal OpenCV

Gambar 10 menampilkan cara atau syntax untuk melakukan install open cv pada laptop.

1.2.10 *Google Colaboratory*

Google Colaboratory, sering disebut sebagai Google Colab, adalah lingkungan pengembangan terintegrasi (IDE) berbasis cloud yang memungkinkan pengguna

menulis dan menjalankan kode Python melalui browser. Google Colab menyediakan akses gratis ke sumber daya komputasi yang kuat, termasuk GPU dan TPU, yang sangat bermanfaat untuk tugas-tugas pemrosesan data besar dan pembelajaran mesin. Dengan Google Colab, pengguna dapat membuat, menyimpan, dan berbagi *notebook* berbasis Jupyter, menjadikannya alat yang ideal untuk kolaborasi dalam penelitian dan pengembangan aplikasi berbasis data. Keunggulan utama dari Google Colab adalah kemampuannya untuk menjalankan eksperimen pembelajaran mesin secara efisien tanpa perlu konfigurasi perangkat keras lokal, serta mendukung berbagai pustaka Python yang populer seperti TensorFlow, PyTorch, dan Keras.

1.2.11 Roboflow

RoboFlow adalah platform yang dirancang untuk mempermudah pengembangan model visi komputer, khususnya dalam deteksi objek. Platform ini menyediakan alat untuk mengumpulkan, mengelola, dan meningkatkan dataset citra, serta melatih dan menerapkan model deteksi objek dengan mudah.

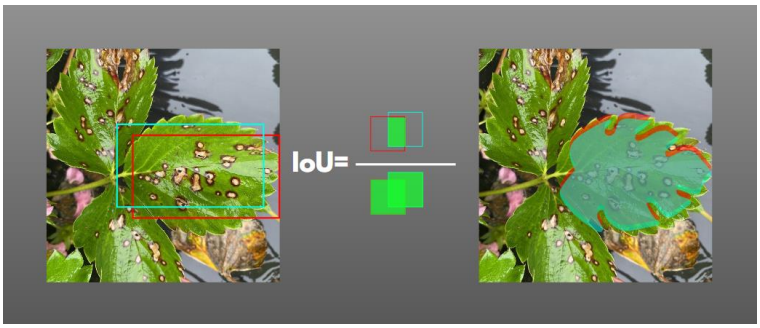
a. Fitur Utama RoboFlow
   - Pengelolaan Dataset: RoboFlow memungkinkan pengguna untuk mengunggah dan mengatur dataset citra gambar-gambar. Platform ini mendukung berbagai format anotasi dan menyediakan alat untuk mengkonversi antara format yang berbeda. Selain itu, RoboFlow menawarkan fitur augmentasi data untuk meningkatkan jumlah dan variasi data pelatihan.
   - Anotasi Data: RoboFlow menyediakan antarmuka anotasi yang intuitif yang memudahkan pengguna dalam memberi label pada gambar. Pengguna dapat menambahkan kotak pembatas (*bounding boxes*) untuk objek yang ingin dideteksi.
   - Augmentasi Data: Dengan RoboFlow, pengguna dapat melakukan augmentasi data seperti rotasi, pemotongan, perubahan ukuran, dan penyesuaian warna untuk memperkaya dataset dan meningkatkan kinerja model.
   - Pelatihan Model: RoboFlow memungkinkan pengguna untuk melatih model deteksi objek menggunakan berbagai algoritma pembelajaran mendalam seperti *YOLO*, SSD, dan Faster R-CNN. Platform ini juga terintegrasi dengan framework pembelajaran mesin populer seperti TensorFlow dan PyTorch.
   - Penerapan Model: Setelah model dilatih, RoboFlow menyediakan alat untuk menerapkan model di lingkungan produksi. Pengguna dapat mengekspor model dalam berbagai format dan mengintegrasikannya ke dalam aplikasi gambar-gambar.

1.2.12 *Mean Average Precision* (mAP)

Mean Average Precision (mAP) adalah metrik evaluasi yang digunakan untuk mengukur kinerja model deteksi objek dan segmentasi. Metrik ini menggabungkan *precision* dan *recall* untuk memberikan gambaran menyeluruh tentang seberapa baik model dalam mendeteksi objek dalam gambar.

1. *Intersection Over Union* (IoU)

   Intersection over Union (IoU) adalah metrik yang digunakan untuk mengukur tingkat tumpang tindih antara dua wilayah, biasanya antara kotak prediksi (*predicted bounding box*) dan kotak kebenaran dasar (*ground truth bounding box*) dalam konteks deteksi objek dan segmentasi.



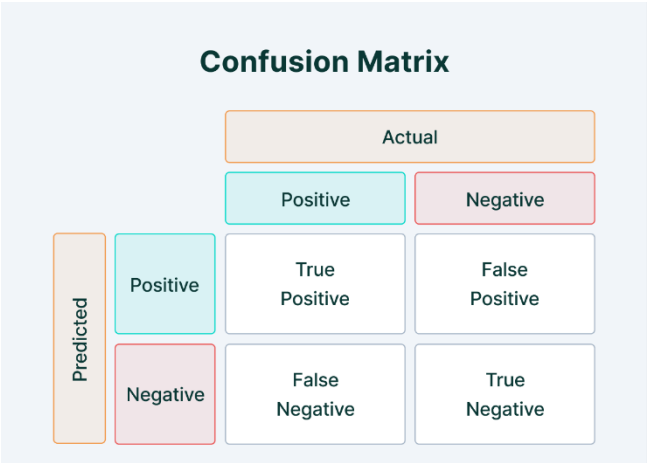
Gambar 11. Contoh Perhitungan IoU

Sumber: (Kukil, 2022)

Gambar 11 menunjukkan contoh *bounding box* IoU. Nilai IoU berkisar dari 0 hingga 1. Dengan menggunakan nilai ambang (*threshold*) IoU, kita bisa menentukan apakah sebuah prediksi adalah *True Positive* (TP), *False Positive* (FP), atau *False Negative* (FN).

2. Confusion Matrix

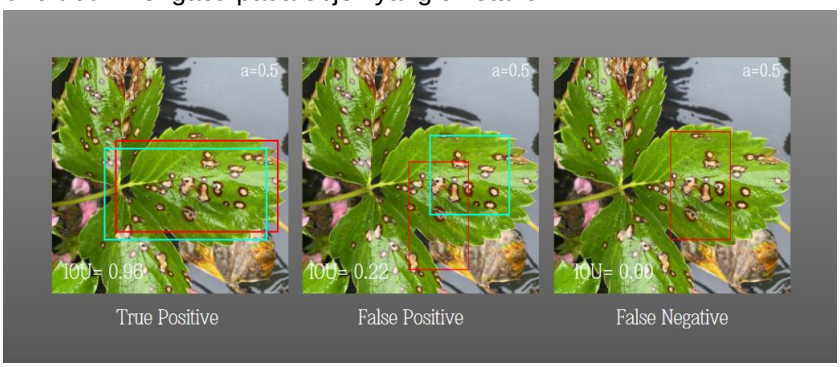
Confusion Matrix adalah matriks yang digunakan untuk menilai kinerja model klasifikasi dalam pembelajaran mesin. Matriks ini memberikan gambaran rinci tentang prediksi yang benar dan salah yang dibuat oleh model, dan dibagi menjadi beberapa kategori: *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN). Matriks ini membantu dalam menghitung berbagai metrik evaluasi seperti accuracy, precision, recall, dan F1 score.



Gambar 12. Confusion Matrix

Sumber: (Deval Shah, 2022)

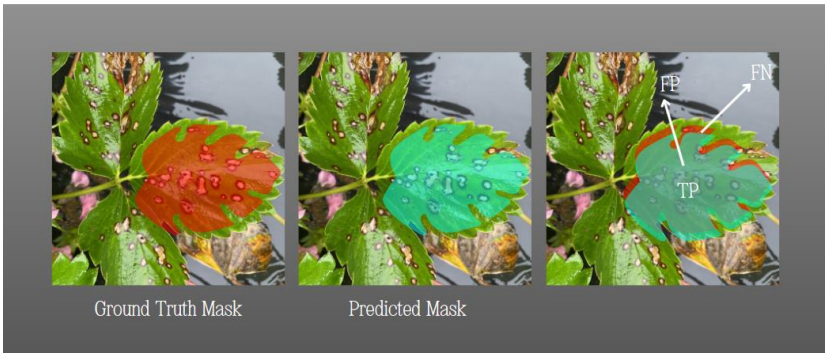
Gambar 12 menampilkan confussion matrix. Dalam deteksi objek, kebenaran prediksi (TP, FP, atau FN) ditentukan dengan menggunakan ambang batas IoU (*Intersection over Union*). Sedangkan dalam segmentasi gambar, kebenaran prediksi ditentukan dengan merujuk pada piksel *Ground Truth. Ground truth* mengacu pada objek yang diketahui.



Gambar 13. *Object detection predictions based on IoU*

Sumber: (Kukil, 2022)

Gambar 13 menunjukkan contoh dari *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN) dalam deteksi objek menggunakan metrik *Intersection over Union* (IoU). *True Positive* (TP) terjadi ketika prediksi memiliki IoU tinggi dengan *ground truth* (0.96), *False Positive* (FP) terjadi ketika prediksi memiliki IoU rendah (0.22), dan *False Negative* (FN) terjadi ketika tidak ada prediksi yang tumpang tindih dengan *ground truth.*



Gambar 14. *Image segmentation predictions based on Ground Truth Mask*

Sumber: (Kukil, 2022)

Gambar 14 menunjukkan contoh dari segmentasi gambar dengan menggunakan masker *ground truth* dan masker prediksi. Di gambar pertama, masker *ground truth* menunjukkan area sebenarnya dari objek (Daun) dengan warna merah. Di gambar kedua, masker prediksi menunjukkan area yang diprediksi oleh model dengan warna biru. Gambar ketiga menggabungkan kedua masker tersebut untuk menunjukkan perbedaan antara prediksi dan *ground truth*. Area hijau menunjukkan *True Positive* (TP) di mana prediksi dan ground truth tumpang tindih, area merah menunjukkan *False Negative* (FN) di mana ground truth tidak terdeteksi oleh prediksi, dan area biru menunjukkan *False Positive* (FP) di mana model memprediksi objek yang sebenarnya tidak ada.

3. Presisi dan Recall
    1. *Presisi dan recall* adalah dua metrik evaluasi yang penting dalam konteks deteksi objek dan segmentasi gambar. *Presisi* mengukur proporsi prediksi positif yang benar-benar positif, yaitu seberapa banyak dari semua prediksi positif yang benar-benar tepat. Dengan kata lain, *presisi* memberikan gambaran tentang seberapa reliabel model dalam membuat prediksi positif. Rumus untuk *presisi* adalah:

        $$P = \frac{TP}{TP+FP} \quad (1)$$

    2. Sementara itu, *recall* mengukur proporsi kasus positif yang benar-benar terdeteksi oleh model, yaitu seberapa banyak dari semua kasus positif yang berhasil dikenali oleh model. *Recall* memberikan gambaran tentang seberapa baik model dalam menemukan semua kasus positif. Rumus untuk *recall* adalah:

        $$R = \frac{TP}{TP+FN} \quad (2)$$

4. *Mean Average Precision* (mAP)

   Mean Average Precision (mAP) dihitung dengan cara menemukan *Average Precision* (AP) untuk setiap kelas objek, kemudian merata-ratakan nilai AP tersebut dari semua kelas yang ada.

   $AP = \frac{1}{11} \sum(precision\ yang\ diinterpolasi\ pada\ 11\ titik)$ (3)

   Artinya, mAP adalah rata-rata dari nilai-nilai AP yang dihitung untuk setiap kelas individu, memberikan ukuran keseluruhan tentang kinerja model deteksi *object* di seluruh kelas yang diuji (J. Tian et al., 2024).

   $mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$ (4)

1.3 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka rumusan masalah yang timbul adalah berikut:

1. Bagaimana membangun model deteksi penyakit pada daun *strawberry* secara realtime menggunakan algoritma *YOLO*?
2. Bagaimana kinerja algoritma *YOLO* pada deteksi penyakit pada daun *strawberry*?

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Membangun sistem untuk deteksi penyakit pada daun *strawberry* secara Realtime menggunakan Algoritma *YOLO*.
2. Menganalisis kinerja Algoritma *YOLO* pada deteksi penyakit pada daun *strawberry*.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Bagi petani, sistem yang dibangun diharapkan dapat digunakan oleh petani untuk mendeteksi penyakit pada daun *strawberry*.
2. Bagi peneliti, penelitian ini diharapkan dapat menjadi referensi bagi peneliti selanjutnya yang ingin melakukan penelitian serupa.

1.6 Ruang Lingkup

Ruang lingkup penelitian ini adalah

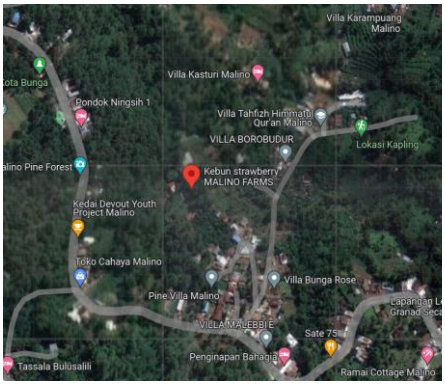
1. Jumlah *class* pada objek penyakit pada daun *strawberry* ada 3 *class* yaitu: hawar daun, bercak daun dan karat daun.
2. Menggunakan algoritma *YOLOv8*.

BAB II

METODE PENELITIAN

2.1 Lokasi Penelitian

Penelitian ini dilakukan dari bulan September 2023 hingga Juli 2024. Pengumpulan data dilakukan di Kebun *Strawberry* di Malino, Kabupaten Gowa, dan di Laboratorium CC Teknik Informatika, Universitas Hasanuddin, Makassar. Tampilan lokasi penelitian ditunjukkan pada Gambar 16.



(a) Kebun *Strawberry* Malino

(b) Departemen Teknik Informatika Universitas Hasanuddin

Gambar 15. Lokasi Penelitian

2.2 Instrumen Penelitian

Penelitian ini menggunakan beberapa perangkat, yaitu:

2.2.1 Perangkat Keras

Ada beberapa perangkat keras yang digunakan pada penelitian ini, di antaranya:

a. Laptop ASUS ROG STRIX G513IH dengan spesifikasi sebagai berikut:
   1) AMD Ryzen 7 4900HS @3.00GHz
   2) RAM 16 GB
   3) NVIDIA GeForce GTX 1660 Ti
   4) SSD NVMe 1 TB
   5) Windows 11 Pro
b. Ponsel iPhone XS untuk pengambilan gambar di lapangan.

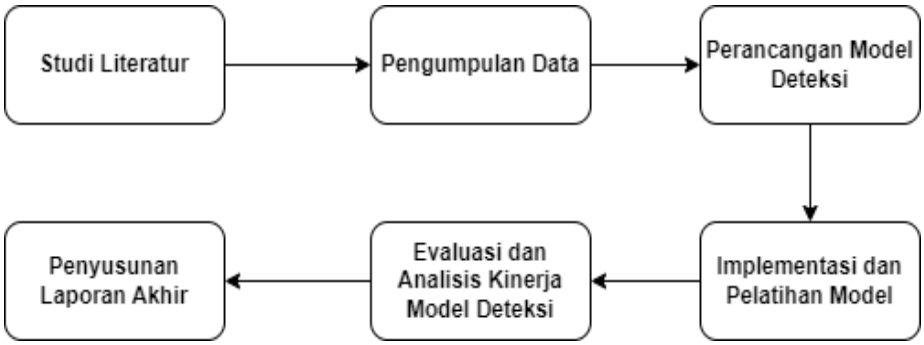
2.2.2 Perangkat Lunak

Ada beberapa perangkat lunak yang digunakan pada penelitian ini, di antaranya:

a. Google Colaboratory
b. Roboflow
c. Python 3.9
d. PyCharm Community Edition 2023.1
e. Microsoft Office 365 (Word, Excel, PowerPoint)
f. NVIDIA CUDA Toolkit v11.3
g. OpenCV 4.5.3
h. *YOLO*v8
i. Scikit-Learn 1.0.2

2.3 Tahapan Penelitian

Gambar 16 menampilkan tahapan penelitian yang diterapkan pada penelitian deteksi penyakit Strawberry menggunakan *YOLOv8*.



Gambar 16. Tahapan Penelitian

Tahapan-tahapan pada Gambar 16 diuraikan sebagai berikut:

1. Studi Literatur

   Tahapan ini bertujuan untuk memahami konteks penelitian dengan menelusuri dan menganalisis literatur yang relevan. Kajian literatur difokuskan pada dua bidang utama: deteksi penyakit tanaman, khususnya pada *strawberry*, dan teknologi *YOLO*v8 yang digunakan untuk deteksi objek. Studi literatur ini dimulai dengan penelusuran artikel ilmiah, buku, dan laporan penelitian yang membahas metode-metode deteksi penyakit tanaman menggunakan teknologi komputer vision. Sumber-sumber dari database akademis seperti Google Scholar, IEEE Xplore, dan PubMed digunakan untuk mengumpulkan literatur yang relevan. Selanjutnya, literatur yang terkumpul dianalisis untuk mengidentifikasi celah penelitian yang ada, yaitu area-area yang belum banyak diteliti atau memerlukan pengembangan

lebih lanjut dalam deteksi penyakit *strawberry*. Peneliti juga mempelajari penerapan *YOLO*v8 dalam berbagai bidang, termasuk pertanian, untuk memahami kelebihan dan keterbatasannya dalam konteks deteksi penyakit tanaman.

2. Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan dengan mengumpulkan gambar daun dan buah *strawberry* dari kebun *strawberry* di Malino menggunakan kamera ponsel iPhone untuk memastikan kualitas gambar yang tinggi dan variasi kondisi pencahayaan serta sudut pengambilan. Selain itu, data sekunder juga diambil dari Kaggle, sebuah platform yang menyediakan berbagai dataset publik. Kombinasi dari data primer yang diambil langsung dari lapangan dan data sekunder dari Kaggle memberikan variasi yang cukup dalam dataset, sehingga dapat meningkatkan robustnes model *YOLO*v8 dalam mendeteksi penyakit pada daun dan buah *strawberry*.

3. Peracanganan Model Deteksi

Perancangan model deteksi penyakit pada daun dan buah *strawberry* menggunakan *YOLO*v8 melibatkan pemilihan arsitektur *YOLO*v8 yang sesuai, penentuan parameter pelatihan seperti *learning rate, batch size*, dan jumlah *epoch*, serta penerapan metode *augmentasi* data seperti rotasi, *flipping*, dan penyesuaian kecerahan gambar. Data kemudian dibagi menjadi data latih, validasi, dan uji, dan model dilatih menggunakan dataset tersebut. Proses pelatihan diikuti oleh evaluasi kinerja model dengan data validasi untuk mengukur akurasi dan mengidentifikasi area yang memerlukan perbaikan. Setelah model dilatih, evaluasi akhir dilakukan menggunakan data uji untuk mengukur akurasi, *presisi, recall*, dan F1-*score*, memastikan model mampu mendeteksi penyakit pada daun dan buah *strawberry* dengan akurat dan efisien.

4. Implementasi dan Pelatihan Model

Setelah model dilatih, tahap implementasi dilakukan dengan menguji model secara langsung di kebun *strawberry* di Malino. Uji coba ini melibatkan penggunaan model yang telah dilatih untuk mendeteksi penyakit pada daun dan buah *strawberry* secara real-time di lapangan, menggunakan kamera ponsel iPhone untuk mengambil gambar dan mendeteksi penyakit langsung di kebun. Hasil deteksi dibandingkan dengan kondisi nyata untuk mengevaluasi akurasi dan efektivitas model dalam kondisi lapangan sebenarnya. Implementasi ini bertujuan untuk memastikan bahwa model *YOLO*v8 tidak hanya efektif di lingkungan pengujian tetapi juga dalam aplikasi praktis di lapangan, membantu petani mengidentifikasi dan mengatasi penyakit dengan cepat dan akurat.

5. Evaluasi dan Analisis Kinerja Model Deteksi

Evaluasi dan analisis kinerja model deteksi *YOLO*v8 dilakukan untuk menilai efektivitas model dalam mendeteksi penyakit pada daun dan buah *strawberry*. Proses evaluasi menggunakan data uji yang telah dipersiapkan, yang terdiri dari gambar-gambar yang tidak pernah dilihat oleh model selama pelatihan, untuk menguji kemampuan generalisasi model. Metrik utama yang digunakan dalam evaluasi meliputi *presisi, recall*, dan *mean Average Precision* (mAP). Presisi mengukur proporsi prediksi positif yang benar-benar positif, menunjukkan ketepatan model dalam mendeteksi penyakit. *Recall* mengukur kemampuan model untuk mendeteksi semua kasus positif, menilai sensitivitas model dalam mendeteksi penyakit. *Mean Average Precision* (mAP) adalah metrik yang menggabungkan presisi dan *recall* pada berbagai tingkat ambang batas, memberikan gambaran keseluruhan tentang kinerja model dalam mendeteksi penyakit.

6. Penyusunan Laporan Akhir

Penyusunan laporan akhir adalah tahap akhir dalam penelitian deteksi penyakit pada daun dan buah *strawberry* menggunakan *YOLO*v8. Laporan dimulai dengan pendahuluan yang mencakup latar belakang, tujuan, dan rumusan masalah, memberikan konteks penting untuk penelitian. Bagian kajian literatur mengulas penelitian sebelumnya dan teknologi yang digunakan, seperti *YOLO*v8. Metodologi penelitian dijelaskan secara rinci, mencakup pengumpulan data, persiapan data, perancangan model, implementasi, dan evaluasi. Hasil dan pembahasan mempresentasikan kinerja model berdasarkan metrik *presisi, recall*, dan *mean Average Precision* (mAP), serta analisis kekuatan dan kelemahan model. Kesimpulan merangkum temuan utama, menjawab tujuan penelitian, dan memberikan rekomendasi untuk penelitian masa depan. Laporan juga mencantumkan daftar pustaka dan lampiran relevan, memastikan transparansi dan kelengkapan informasi.

2.4 Teknik Pengumpulan Data

Penelitian ini menggunakan dua sumber data utama: data primer yang diperoleh daripengambilan langsung di lapangan dan data sekunder yang diunduh dari Kaggle

1. Pengambilan Data Langsung

Data primer dikumpulkan melalui pengambilan gambar langsung di Kebun *Strawberry* di Malino, Kabupaten Gowa, Sulawesi Selatan. Pengambilan gambar dilakukan menggunakan ponsel iPhone XS untuk memastikan kualitas gambar yang tinggi dan variasi kondisi pencahayaan serta sudut pengambilan. Gambar diambil dari daun dan buah *strawberry* yang sehat dan yang menunjukkan gejala penyakit. Proses ini melibatkan dokumentasi kondisi tanaman secara menyeluruh untuk mendapatkan dataset yang representatif.

Gambar 15. Contoh Skenario pengambilan Dataset

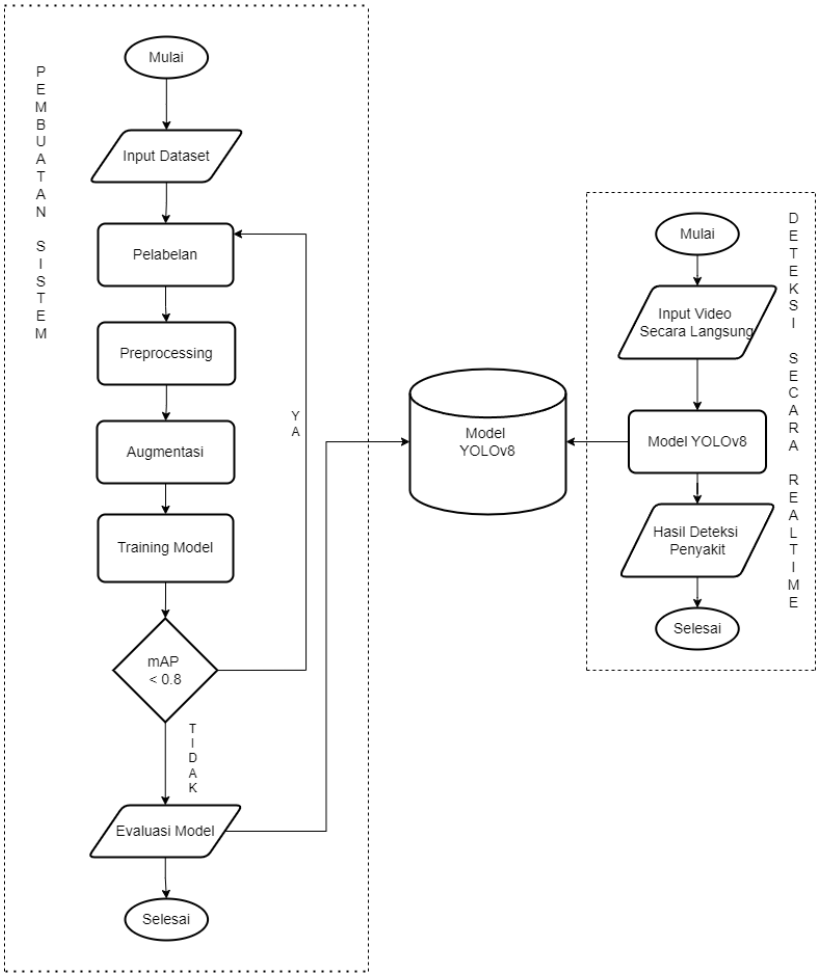
Gambar 17 menunjukkan skenario pengambilan dataset, di mana gambar diambil dari jarak 15 - 30 cm dari objek. Pengambilan gambar dari jarak ini dilakukan untuk memastikan detail dan fitur penting dari objek dapat ditangkap dengan jelas, sehingga dapat mendukung proses ekstraksi fitur dan klasifikasi lebih lanjut. Jarak yang konsisten juga membantu menjaga keseragaman dalam dataset dan meminimalisir variasi yang tidak diinginkan akibat perubahan jarak pengambilan gambar.

2. Data Sekunder dari *Kaggle*

Selain data primer, penelitian ini juga memanfaatkan data sekunder yang diunduh dari *Kaggle. Kaggle* menyediakan berbagai dataset publik yang dapat digunakan untuk melatih dan menguji model deteksi penyakit. Dataset dari *Kaggle* meliputi gambar-gambar daun dan buah *strawberry* dalam berbagai kondisi kesehatan, termasuk berbagai jenis penyakit yang umum ditemukan. Data ini digunakan untuk melengkapi dan memperkaya dataset yang dikumpulkan dari lapangan, memberikan variasi yang lebih luas dalam data pelatihan dan pengujian

2.5 Perancangan Sistem

Gambar 18 menampilkan racangan sistem penelitian ini:



Gambar 16. Flowchart Rancangan Sistem

Tahapan-tahapan pada Gambar 18 diuraikan sebagai berikut.

2.5.1 *Dataset*

Dataset yang berhasil dikumpulkan pada penelitian ini sebanyak 1078 data. Gambar menampilkan contoh data yang berhasil dikumpulkan. Gambar menampilkan contoh data yang berhasil di kumpulkan dari kebun *strawberry* Malino, dan gambar menampilkan contoh data yang berhasil dikumpulkan dari Kaggle.

Gambar 17. Dataset Primer

Gambar 18. Dataset Sekunder

2.5.2 Pelabelan

Pelabelan pada penelitian ini menggunakan Roboflow. Dataset terbagi menjadi 3 kelas yaitu Bercak Daun, Hawar Daun dan Busuk Buah. Berikut adalah beberapa contoh gambar dari masing-masing kelas yang telah dilabel.

Gambar 19. Pelabelan Kelas Busuk Buah

Gambar 20. Pelabelan Kelas Hawar Daun

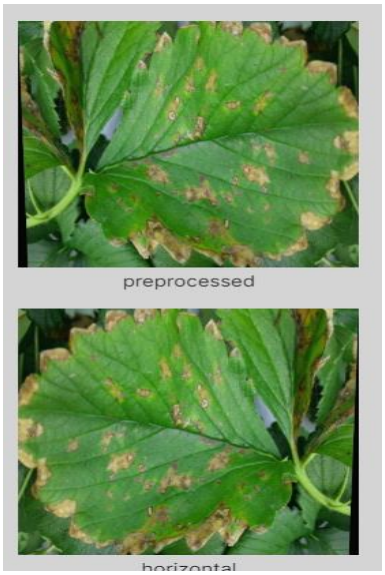
2.5.3 *Preprocessing* data

Preprocessing yang digunakan pada penelitian ini adalah *resize*, yaitu proses mengubah ukuran gambar ke dimensi tertentu untuk memastikan konsistensi ukuran, mengurangi beban komputasi, meningkatkan kinerja model, mengurangi noise, dan menstandarisasi proses pelatihan. *Resize* yang digunakan adalah 640 * 640.

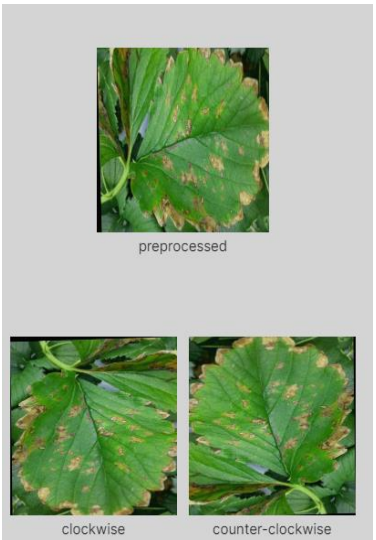
Gambar 24. Contoh Proses *Resize*

2.5.4 *Augmentasi*

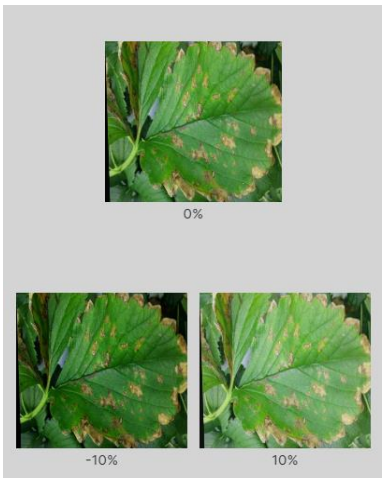
Penelitian ini menggunakan beberapa teknik *augmentasi* data untuk meningkatkan variasi dalam dataset, yaitu *flip horizontal*, rotasi 90°, penyesuaian eksposur, dan penambahan noise. *Flip horizontal* membalik gambar secara *horizontal*, sedangkan rotasi 90° memutar gambar searah atau berlawanan arah jarum jam. Penyesuaian eksposur mengubah kecerahan gambar antara -10% hingga +10%, dan penambahan *noise* memperkenalkan gangguan acak hingga 0.85% dari piksel.



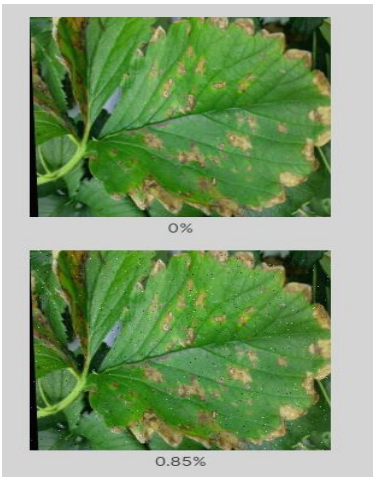
(a) *Flip*



(b) *Rotasi*



(c) *Exposure*

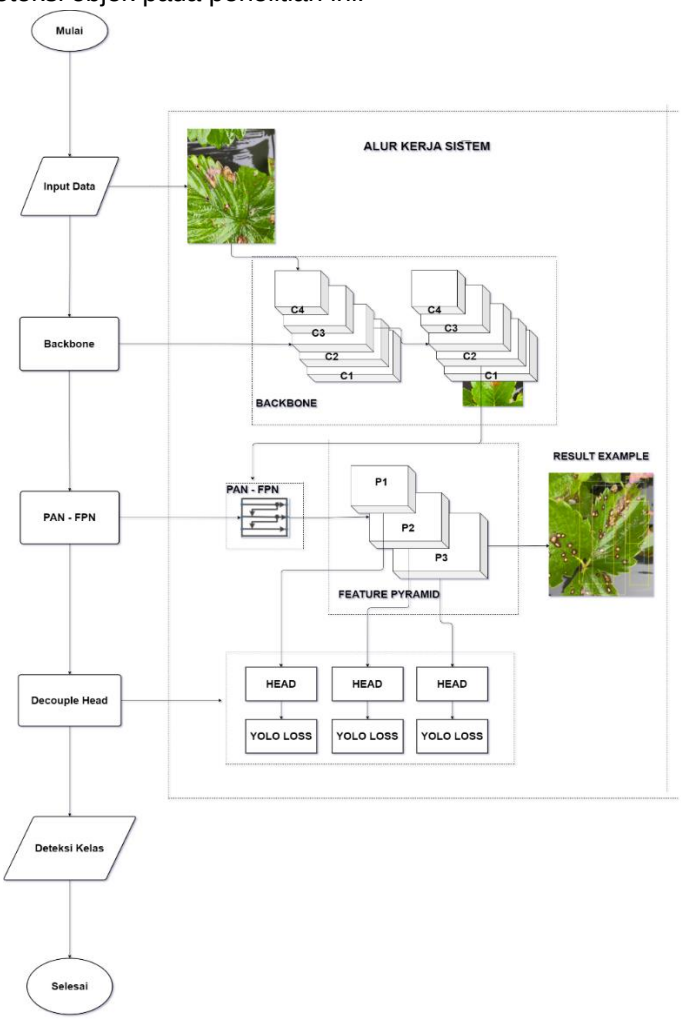


(d) *Noise*

Gambar 25 Teknik Augmentasi

2.5.5 Pembangunan Model *YOLO*v8

Setelah *dataset* melalui proses *preprocessing* dan *augmentasi, dataset* tersebut siap untuk diproses oleh model *YOLO*v8. Gambar menunjukkan *flowchart* dari *YOLO*v8 dalam mendeteksi objek pada penelitian ini:



Gambar 21. *Flowchart YOLOv8*

1. Input Data
   Proses dimulai dengan memasukkan gambar daun dan buah *strawberry* yang akan dideteksi ke dalam sistem. Gambar-gambar ini merupakan data input mentah yang akan diproses lebih lanjut oleh model *YOLO*v8.
2. *Backbone*
   Setelah gambar dimasukkan, gambar-gambar diproses oleh *backbone* dari model *YOLO*v8. *Backbone* ini terdiri dari beberapa lapisan konvolusi (C1, C2, C3, C4) yang bertujuan untuk mengekstraksi fitur-fitur penting dari gambar. Setiap lapisan konvolusi menangkap pola dan fitur pada berbagai tingkat abstraksi, memungkinkan model untuk memahami konten gambar secara

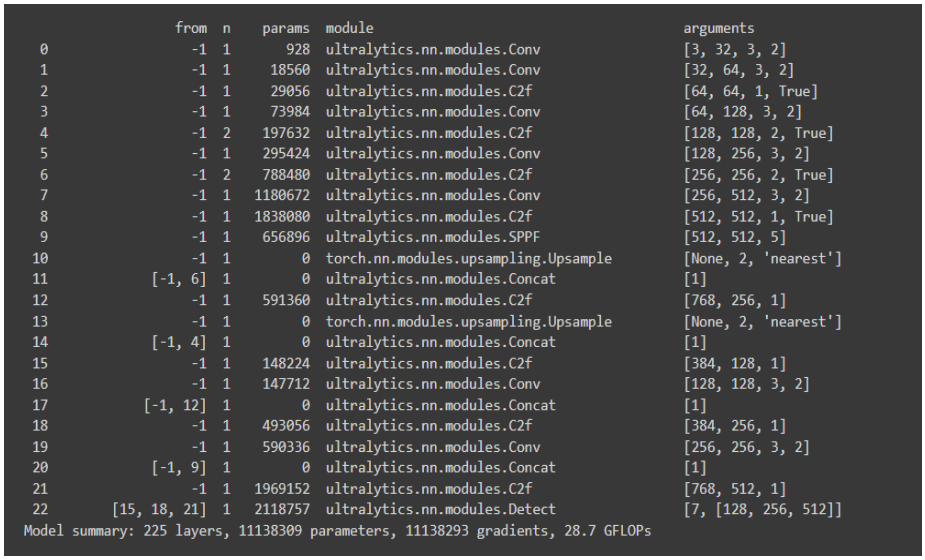
mendalam. Lapisan-lapisan ini bekerja secara berurutan, di mana setiap lapisan mengolah *output* dari lapisan sebelumnya untuk membentuk representasi fitur yang lebih kompleks.

3. PAN – FPN (*Path Aggregation Network – Feature Pyramid Network*)
   Setelah fitur-fitur dasar diekstraksi oleh *backbone*, gambar-gambar diteruskan ke jaringan PAN – FPN. Jaringan ini menggabungkan informasi dari berbagai skala fitur yang diekstraksi oleh *backbone*. FPN membangun piramida fitur yang memungkinkan deteksi objek pada berbagai skala, dari yang kecil hingga besar. PAN kemudian menggabungkan fitur-fitur ini untuk memperkaya informasi yang digunakan dalam deteksi objek, sehingga meningkatkan kemampuan model dalam mendeteksi objek dengan berbagai ukuran dalam gambar yang sama.
4. *Feature Pyramid*
   Fitur-fitur yang dihasilkan oleh PAN – FPN kemudian diintegrasikan dalam *Feature Pyramid* (P1, P2, P3). *Feature Pyramid* ini bertujuan untuk mempertahankan resolusi tinggi dari fitur-fitur yang diproses melalui jaringan. Dengan mempertahankan informasi resolusi tinggi, model dapat mendeteksi objek yang lebih detail dan kecil, yang sangat penting dalam deteksi penyakit pada daun dan buah *strawberry* yang mungkin menunjukkan gejala kecil atau halus.
5. *Decouple Head*
   Setelah fitur-fitur diolah melalui *Feature Pyramid*, gambar-gambar diteruskan ke bagian *head* dari model yang dipisahkan (*decouple head*). *Head* ini bertanggung jawab untuk menghasilkan prediksi akhir berupa kotak pembatas (*bounding boxes*) dan kelas objek untuk setiap kotak pembatas tersebut. *Decouple head* terdiri dari beberapa lapisan yang mengolah fitur-fitur yang telah digabungkan untuk menghasilkan deteksi yang akurat dan relevan. Lapisan-lapisan dalam *head* ini memastikan bahwa setiap objek dalam gambar diberi kotak pembatas dan label kelas yang tepat.
6. *YOLO Loss*
   Setiap prediksi yang dihasilkan oleh *head* dievaluasi menggunakan fungsi kerugian *YOLO* (*YOLO Loss*). *YOLO Loss* menghitung kesalahan antara prediksi model dan label sebenarnya (*ground truth*) dalam hal posisi kotak pembatas dan klasifikasi objek. Evaluasi ini penting untuk mengoptimalkan model selama pelatihan, memastikan bahwa model belajar dari kesalahan dan meningkatkan akurasi prediksinya. Fungsi kerugian ini mencakup perhitungan kesalahan dalam prediksi posisi kotak, ukuran kotak, dan kelas objek.
7. Deteksi Kelas
   Hasil akhir dari prediksi setelah melewati proses evaluasi dan penyesuaian adalah deteksi kelas objek. Pada tahap ini, sistem menentukan kelas dari objek yang terdeteksi berdasarkan fitur-fitur yang telah diekstraksi dan diproses melalui jaringan. Deteksi kelas ini menghasilkan label kelas untuk

setiap objek yang terdeteksi, seperti jenis penyakit yang ditemukan pada daun atau buah *strawberry*.

8. Selesai

   Proses berakhir dengan keluaran berupa kotak pembatas dan label kelas untuk setiap objek yang terdeteksi dalam gambar. Ini merupakan tahap akhir di mana hasil deteksi disajikan, memberikan informasi yang dapat digunakan untuk analisis lebih lanjut atau tindakan lebih lanjut dalam konteks aplikasi nyata, seperti pengelolaan penyakit tanaman. Gambar 27 menampilkan aristektur model *YOLOv8* yang digunakan pada penelitian ini.



	from	n	params	module	arguments
0	-1	1	928	ultralytics.nn.modules.Conv	[3, 32, 3, 2]
1	-1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
2	-1	1	29056	ultralytics.nn.modules.C2f	[64, 64, 1, True]
3	-1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
4	-1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
5	-1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
6	-1	2	788480	ultralytics.nn.modules.C2f	[256, 256, 2, True]
7	-1	1	1180672	ultralytics.nn.modules.Conv	[256, 512, 3, 2]
8	-1	1	1838080	ultralytics.nn.modules.C2f	[512, 512, 1, True]
9	-1	1	656896	ultralytics.nn.modules.SPPF	[512, 512, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	1	591360	ultralytics.nn.modules.C2f	[768, 256, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	-1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
16	-1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	-1	1	493056	ultralytics.nn.modules.C2f	[384, 256, 1]
19	-1	1	590336	ultralytics.nn.modules.Conv	[256, 256, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.Concat	[1]
21	-1	1	1969152	ultralytics.nn.modules.C2f	[768, 512, 1]
22	[15, 18, 21]	1	2118757	ultralytics.nn.modules.Detect	[7, [128, 256, 512]]

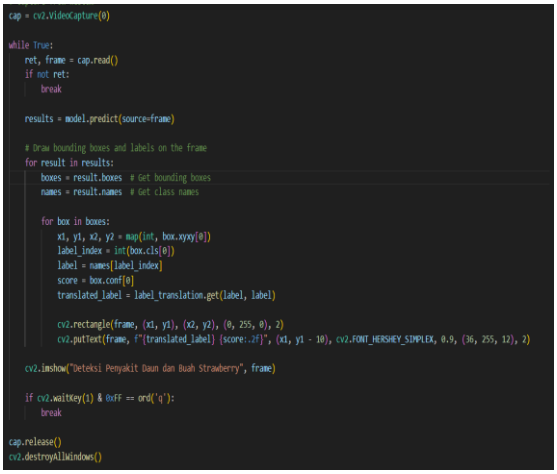
Model summary: 225 layers, 11138309 parameters, 11138293 gradients, 28.7 GFLOPs

Gambar 22. Arsitektur model *YOLO*v8 pada Penelitian

Proses dimulai dengan lapisan konvolusi awal yang mengekstraksi fitur dasar dari gambar input. Layer 0 dan 1 menggunakan konvolusi dengan kernel 3x3 dan stride 2 untuk menghasilkan filter yang lebih besar, masing-masing menghasilkan 32 dan 64 filter. Selanjutnya, layer 2 menggunakan modul C2f (*Cross Stage Partial with Convolution*) untuk memproses fitur lebih lanjut dan memungkinkan komunikasi antar-lapisan. Layer 3 dan 4 melanjutkan proses dengan konvolusi dan C2f untuk meningkatkan kompleksitas fitur yang diekstraksi. Proses ini diulang pada layer 5 hingga 8 dengan jumlah filter yang meningkat, hingga mencapai 512 filter pada layer 7. Layer 9 menggunakan modul SPPF (*Spatial Pyramid Pooling – Fast*) untuk menggabungkan informasi dari berbagai skala fitur. Setelah itu, fitur yang diekstraksi melalui beberapa tahap upsampling dan concatenation pada layer 10 hingga 13. Akhirnya, pada layer 22, modul Detect menghasilkan prediksi akhir berupa kotak pembatas dan label kelas untuk setiap objek yang terdeteksi. Keseluruhan arsitektur ini dioptimalkan dengan menggunakan fungsi kerugian *YOLO* (*YOLO* Loss) yang menghitung kesalahan antara prediksi dan label ground truth, memastikan deteksi yang akurat dan efisien.

2.5.6 Setup Model Untuk Deteksi Objek Secara *Real Time*

Untuk setup deteksi objek secara *real-time* dalam penelitian ini, digunakan kamera eksternal yang dihubungkan melalui kabel *USB*. Kamera *eksternal* ini termasuk kamera GoPro Hero 9 *Black* dan *iPhone* 14 *Pro Max*, yang terhubung langsung ke *MacBook Pro* menggunakan kabel USB. Proses pengaturan dimulai dengan menghubungkan kamera eksternal ke *MacBook Pro* melalui *port USB*, dan memastikan bahwa *driver* atau perangkat lunak yang diperlukan untuk mengakses kamera telah terinstal di sistem.



```
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = model.predict(source=frame)

    # Draw bounding boxes and labels on the frame
    for result in results:
        boxes = result.boxes  # Get bounding boxes
        names = result.names  # Get class names

        for box in boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            label_index = int(box.cls[0])
            label = names[label_index]
            score = box.conf[0]
            translated_label = label_translation.get(label, label)

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, f"{translated_label} {score:.2f}", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2)

    cv2.imshow("Deteksi Penyakit Daun dan Buah Strawberry", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Gambar 23. Source Code untuk Mengubungkan Kamera dengan Model *YOLO*v8

Gambar 28 menampilkan code yang digunakan untuk menghubungkan kamera dengan model *YOLO*v8.

Selanjutnya, kamera diatur untuk mengirimkan aliran video langsung ke aplikasi deteksi objek yang berjalan di lingkungan Visual Studio Code. Pustaka cv2 dan ultralytics digunakan untuk memproses video secara *real-time* dan menerapkan model deteksi yang telah dilatih.

Gambar 24. Contoh Tampilan Desktop Siste

Gambar 29 menampilkan contoh sistem yang berhasil dijalankan melalui visual studio code. Uji coba dilakukan dengan menggunakan kamera drone secara, gambar 30 menampilkan pengujian yang dilakukan di kebun *strawberry* di Kabupaten Gowa secara realtime.

Gambar 25. Pengujian Secara Realtime