BAB II

KAJIAN PUSTAKA

A. KAJIAN TEORITIS

1. Sistem Deteksi

a. Sistem

Sistem diambil dari kata Latin (*systema*) serta kata Yunani (*sustema*), yang berarti sekumpulan bagian atau elemen yang terhubung satu sama lain untuk memperlancar pergerakan informasi, materi, atau energi (Erkhani & Dianta, 2024). Setiap sistem memiliki masukan, proses, dan keluaran yang saling terhubung. Sistem digunakan dalam bidang teknologi informasi untuk mengubah data menjadi informasi yang ramah pengguna. Pada dasarnya, sistem adalah kumpulan komponen yang saling terhubung dan bekerja sama untuk mencapai tujuan tertentu (Mare & Yana, 2022). Secara umum, sistem terdiri dari beberapa komponen utama, seperti *input*, proses, dan *output*, yang berfungsi secara berurutan untuk menghasilkan data yang dibutuhkan oleh pengguna.

b. Deteksi

Deteksi adalah suatu proses untuk mengidentifikasi atau menemukan keberadaan suatu objek, pola, atau kejadian tertentu berdasarkan data yang diperoleh melalui pengamatan atau sensor. Dalam teknologi informasi, deteksi sering digunakan untuk mengenali objek, menganalisis pola, dan menemukan kondisi tertentu seperti anomali atau klasifikasi data dengan bantuan algoritma tertentu. Pengolahan data,

ekstraksi fitur, dan pengambilan keputusan berdasarkan model yang digunakan biasanya merupakan bagian dari proses deteksi. Deteksi objek menggunakan algoritma dari *Convolution Neural Network*, dan yang paling terkenal saat ini adalah *You Only Look Once (YOLO)* dengan versi terbaru. (Pratama et al., 2022).

Dalam penelitian ini, deteksi objek digunakan untuk mengidentifikasi daging ayam pada citra sekaligus mengklasifikasikan tingkat kesegarannya. Penerapan deteksi objek memungkinkan sistem menampilkan *bounding box* dan label hasil deteksi secara *real-time* sehingga dapat membantu pengguna dalam mengetahui kondisi daging ayam secara cepat dan akurat. Algoritma ini akan memecah gambar menjadi grid dengan ukuran s×s, dan kemudian untuk setiap grid akan memprediksi *boanding box* serta peta kategori untuk setiap grid tersebut. (Sanubari & Puriyanto, 2022).

2. Teori Pengembangan Sistem

a. You Only Look Once (YOLO)

You Only Look Once (YOLO) adalah algoritma deteksi objek berbasis *deep learning* yang memiliki kemampuan untuk mendeteksi dan mengklasifikasikan objek dalam gambar atau video secara *real-time*. Metode ini menggunakan pemrosesan gambar satu kali untuk menentukan lokasi objek dalam bentuk *bounding box* dan kelasnya. *You Only Look Once (YOLO)* populer dalam berbagai aplikasi *vision komputer* karena efisiensi dan kecepatan deteksi objeknya. Sistem yang digunakan untuk mendeteksi adalah dengan memanfaatkan

pengklasifikasi *repurpose classifier* atau *localizer* untuk melakukan proses deteksi (Evita et al., 2022). Misalnya, mereka dapat melihat seluruh gambar saat test dilakukan dengan prediksi global yang diinformasikan pada gambar.

YOLO juga dapat mengenali berbagai objek secara bersamaan dalam satu gambar dengan sangat akurat. Selain itu, algoritma ini terus dikembangkan dari versi *YOLO*v3, *YOLO*v5, dan *YOLO*v8, yang meningkatkan akurasi, kecepatan, dan efisiensi model. *You Only Look Once (YOLO)* sering digunakan dalam penelitian dalam berbagai bidang seperti pengawasan keamanan, kendaraan otonom, dan analisis gambar medis. *You Only Look Once (YOLO)* sangat cocok untuk aplikasi berbasis kamera seperti sistem klasifikasi kesegaran daging ayam karena kemampuannya untuk mendeteksi objek secara *real-time* adalah keunggulannya. Berdasarkan teori yang telah dijelaskan, Pemilihan *YOLOv8n* dilakukan karena merupakan varian *YOLOv8* yang memiliki ukuran model lebih ringan sehingga lebih cocok untuk implementasi pada perangkat *Android* dibandingkan varian *YOLOv8* lainnya. *You Only Look Once (YOLO)* sering disebut sebagai algoritma untuk melakukan tugas deteksi obyek, seperti klasifikasi dan lokalisasi (Pratama et al., 2025). Selain itu, *You Only Look Once (YOLO)* juga dikenal sebagai model deteksi obyek satu tahap.

b. Android

Android adalah sistem operasi untuk perangkat seluler, termasuk tablet dan *smartphone*, yang berbasis pada *Linux*. *Google*

mengembangkan *Android*, yang bersifat *Open Source*, sehingga siapa pun dapat menggunakannya dan mengembangkan perangkat lunak di atasnya. *Android* memprioritaskan konsumen secara umum, meskipun tujuannya adalah *fleksibilitas* (Navintino et al., 2025).

Selain itu, *Android* mendukung berbagai teknologi modern, seperti integrasi dengan kecerdasan buatan, kamera, dan konektivitas internet. Dengan demikian, *Android* sangat cocok untuk pengembangan aplikasi klasifikasi kesegaran daging ayam secara *real-time* berbasis kamera, termasuk penerapan sistem deteksi objek menggunakan metode *deep learning*. *Android* telah menjadi opsi utama, karena saat ini banyak individu mulai beralih menggunakan teknologi ponsel pintar, khususnya sistem operasi *android* (Edy et al., 2022).

c. *Dataset*

Dataset merupakan kumpulan data yang digunakan dalam proses pelatihan *(training)* dan pengujian *(testing)* model *machine learning.* Pada pengolahan citra, *dataset* umumnya berupa kumpulan gambar yang telah diberi label sesuai kategorinya. Data yang diperoleh dari berbagai sumber membantu peneliti dalam proses penelitian, sedangkan kualitas dataset berpengaruh terhadap hasil atau performa sistem. Agar model yang dihasilkan mampu melakukan generalisasi yang baik, *Dataset* harus mencakup berbagai skenario, variasi, dan kompleksitas (Hermawan, 2024). Ini penting untuk data *supervised learning*, deteksi objek seperti klasifikasi kesegaran daging ayam,

Dataset berupa gambar harus mencakup variasi kondisi objek, pencahayaan, dan sudut pengambilan agar model dapat belajar dan membuat prediksi yang akurat.

Dataset pada penelitian ini diperoleh dari *Roboflow* dan hasil pengambilan gambar secara mandiri oleh peneliti. *Dataset Roboflow* digunakan untuk menyediakan data yang telah dianotasi dengan baik, sedangkan *Dataset* hasil pengambilan sendiri digunakan untuk menyesuaikan kondisi data dengan lingkungan penelitian yang sebenarnya.

d. *Google Colab*

Gambar 2. 1 Logo *Google Colab*

Sumber : (Irsabmalfi & Rahmadewi, 2025).

Pada Gambar 2.1 merupakan logo dari *Google Colab* yang akan digunakan dalam penelitian ini. *Google Colaboratory (Google Colab)* merupakan lingkungan pengembangan berbasis *cloud* yang memungkinkan proses penulisan, eksekusi, dan berbagi kode *Python* dilakukan secara daring melalui *web browser*. *Platform* ini banyak digunakan dalam pengembangan *data science* dan *machine learning* oleh peneliti, pengembang, analis data, maupun pendidik karena memberikan akses ke sumber daya komputasi yang fleksibel tanpa

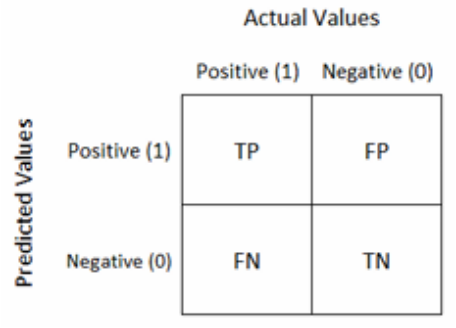
memerlukan instalasi perangkat lunak dan dapat digunakan secara gratis. Alat ini dirancang untuk mendukung peneliti dalam memproses data guna kebutuhan pembelajaran dan melakukan eksperimen melalui pengolahan data, khususnya dalam ranah pembelajaran mesin. Pemakaian alat ini mirip dengan *Jupyter Notebook* (Octarina et al., 2025).

Keunggulan utama *Google Colab* adalah kemampuan untuk menyediakan sumber daya komputasi yang cukup tinggi tanpa memerlukan perangkat keras khusus. Ini sangat membantu dalam proses pelatihan model, seperti pelatihan algoritma deteksi objek menggunakan *You Only Look Once (YOLO)*. Dari segi perangkat keras, *Google Colab* menawarkan fasilitas berbentuk ruang penyimpanan yang terhubung langsung dengan *Google Drive*, unit pemroses yang mencakup *CPU*, *GPU*, dan *TPU*, serta memori *RAM* (Guntara, 2023).

e. *Confusion Matrix*

Confusion Matrix adalah sebuah tabel yang digunakan untuk mengukur performa model klasifikasi dengan cara membandingkan hasil prediksi model terhadap nilai sebenarnya dari data yang diuji. *Confusion Matrix* memberikan gambaran yang jelas mengenai seberapa baik model dalam melakukan klasifikasi pada setiap kelas yang ada, sehingga dapat diketahui di mana model melakukan kesalahan prediksi. Di mana dalam *confussion Matrix* pengukuran dilakukan dengan melihat *Mean Average Precision (mAP)*, *Precision*, *Recall* dan *F1 Score* (Rininda et al., 2023). Saat mengevaluasi kinerja menggunakan matriks

konfusi, ada empat istilah yang menunjukkan hasil dari proses klasifikasi, seperti yang ditunjukkan pada gambar 2. 2.



Gambar 2. 2 *Confusion Matrix*

Sumber : (Atmajaya et al., 2023).

Berdasarkan yang terdapat pada gambar 2.2 *Confusion Matrix* tersebut, dapat dihitung beberapa metrik evaluasi yang digunakan untuk mengukur performa model secara lebih menyeluruh, seperti berikut.

1) *Precesion*

Precision mengukur seberapa tepat model dalam memprediksi kelas positif dari seluruh prediksi positif yang dibuat oleh model. Adapun persamaan *Precesion* dapat di lihat pada persamaan 2.1.

$$Precesion = \frac{TP}{TP + FP} \quad (2.1)$$

True Positive (TP) adalah jumlah data positif yang berhasil diprediksi positif dengan benar oleh model. *False Positive (FP)* adalah jumlah data negatif yang salah diprediksi sebagai data positif oleh model. Nilai *Precision* yang tinggi menunjukkan bahwa model

jarang melakukan kesalahan dalam memprediksi suatu data sebagai kelas positif

2) *Recall*

Recall mengukur seberapa baik model dalam menemukan seluruh data yang seharusnya termasuk ke dalam kelas positif dari keseluruhan data positif yang ada. Adapun persamaan *Recall* dapat di lihat pada persamaan 2.2.

$$Recall = \frac{TP}{TP+FN} \quad (2.2)$$

True Positive (TP) adalah jumlah data positif yang berhasil diprediksi positif dengan benar oleh model. *False Negative (FN)* adalah jumlah data positif yang salah diprediksi sebagai data negatif oleh model. Nilai *Recall* yang tinggi menunjukkan bahwa model mampu mendeteksi hampir seluruh data positif tanpa banyak yang terlewat.

3) *F1 Score*

F1 Score merupakan rata-rata harmonik antara *Precision* dan *Recall* yang memberikan keseimbangan penilaian antara kedua metrik tersebut. *F1 Score* sangat berguna digunakan ketika terdapat ketidakseimbangan jumlah data antar kelas. Adapun persamaan *F1 Score* dapat di lihat pada persamaan 2.3.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.3)$$

Adapun untuk keterangan untuk persamaan 2.3, 2 adalah konstanta pengalian yang digunakan dalam perhitungan rata-rata harmonik antara *Precision dan Recall. Precision* adalah tingkat ketepatan model dalam memprediksi kelas *positif. Recall* adalah kemampuan model dalam menemukan seluruh data positif. Nilai *F1 Score* yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara *Precision* dan *Recall*.

4) *Mean Average Precision (mAP)*

Mean Average Precision (mAP) merupakan metrik evaluasi utama yang digunakan dalam sistem deteksi objek seperti *You Only Look Once (YOLO)*. *Mean Average Precision (mAP)* dihitung sebagai rata-rata nilai Average *Precision* (AP) dari seluruh kelas yang ada. *mAP*50 menggunakan *Threshold Intersection over Union* (*IoU*) sebesar 0.50, sedangkan *mAP*50-95 menggunakan rata-rata *Threshold IoU* dari 0.50 hingga 0.95 dengan langkah 0.05. Adapun persamaan *Mean Average Precision (mAP)* dapat di lihat pada persamaan 2.4.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} A\,P_i \qquad (2.4)$$

Adapun untuk keterangan untuk persamaan 2.4, N adalah jumlah kelas yang digunakan dalam model. i adalah indeks kelas yang dihitung, dimulai dari kelas ke-1 hingga kelas ke-N. *Average Precision* (AP_i) adalah nilai *Average Precision* pada kelas ke-i.

Semakin tinggi nilai *mAP* maka semakin baik kemampuan model dalam mendeteksi dan mengklasifikasikan objek secara akurat.

Selain menggunakan metrik evaluasi seperti *Precision, Recall*, *F1-Score*, *Mean Average Precision (mAP)* untuk mengukur performa model *YOLOv8*, penelitian ini juga menggunakan perhitungan tingkat kesesuaian hasil deteksi sistem terhadap penilaian manual. Perhitungan tingkat kesesuaian digunakan untuk mengetahui sejauh mana hasil klasifikasi yang diberikan oleh sistem sesuai dengan hasil pengamatan manual yang dijadikan sebagai data acuan. Semakin tinggi nilai tingkat kesesuaian yang diperoleh, maka semakin baik kemampuan sistem dalam menghasilkan klasifikasi yang sesuai dengan kondisi sebenarnya.

Tingkat kesesuaian dihitung dengan membandingkan jumlah hasil deteksi yang sesuai dengan hasil penilaian manual terhadap seluruh data yang diuji. Adapun rumus yang digunakan untuk menghitung tingkat kesesuaian dapat di lihat pada persamaan 2.5.

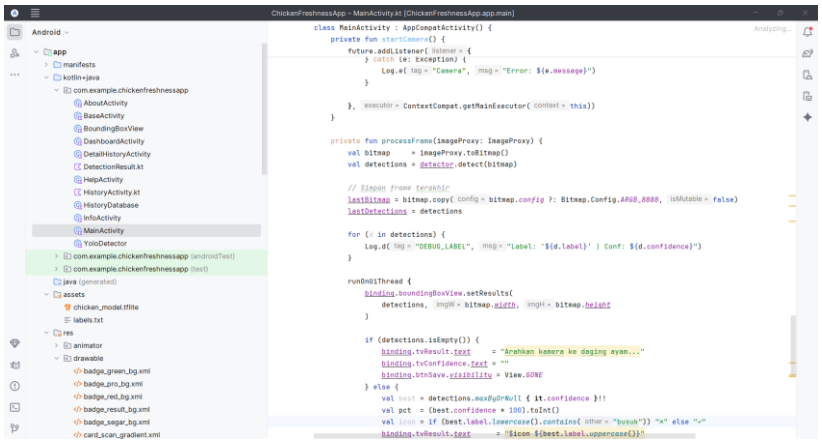
$$\text{Tingkat Kesesuaian} = \frac{\text{Jumlah Data Sesuai}}{\text{Jumlah Seluruh Data Uji}} \times 100\% \qquad (2.5)$$

Adapun untuk keterangan untuk persamaan 2.5, Jumlah Data Sesuai adalah jumlah hasil deteksi sistem yang sama dengan hasil penilaian manual. Jumlah Seluruh Data Uji adalah jumlah keseluruhan data yang digunakan dalam pengujian. Tingkat

Kesesuaian (%) adalah persentase kecocokan antara hasil deteksi sistem dan hasil penilaian manual.

f. *Android Studio*

Android Studio adalah *Integrated Development Environment (IDE)* resmi yang dikembangkan untuk mendukung proses pembuatan dan pengembangan aplikasi pada sistem operasi *Android*. *Android Studio* dikembangkan oleh *Google* dan pertama kali diperkenalkan pada tahun 2013 sebagai pengganti *Eclipse Android Development Tools (ADT)*. ide ini dirancang khusus untuk mempermudah proses pengembangan aplikasi *Android* dengan menyediakan berbagai fitur yang terintegrasi dalam satu *Platform*. Berdasarkan *IntelliJ IDEA* dari *JetBrains*, *Android Studio* adalah ide untuk mengembangkan aplikasi *Android* yang tersedia dalam dua versi (Mansawan et al., 2024). Tampilan antarmuka Android Studio dapat dilihat pada Gambar 2.3.



Gambar 2. 3 Tampilan Android Studio

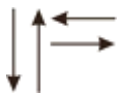
Berdasarkan Gambar 2.3, Android Studio menyediakan berbagai fitur yang mendukung proses pengembangan aplikasi Android.

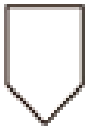
3. Teori Perancangan Sistem

a. *Flowchart*

Flowchart atau diagram alir merupakan representasi grafis yang digunakan untuk menggambarkan urutan langkah, proses, dan aliran logika dalam suatu sistem atau program. *Flowchart* juga dapat diartikan sebagai representasi grafis dari tahapan atau urutan dari suatu prosedur aplikasi yang memiliki tujuan tertentu (Listyoningrum et al., 2023). *Flowchart* memudahkan pengembang dan pengguna dalam memahami alur kerja sistem secara *visual* sehingga proses analisis dan perancangan sistem menjadi lebih terstruktur serta meminimalkan kesalahan pengembangan. Adapun penjelasan *Flowchart* dapat di lihat pada tabel 2. 1.

Tabel 2. 1 *Flowchart*

No	Notasi	Simbol	Keterangan
1	*Flow Direction*		Digunakan untuk menghubungkan antara simbol satu dengan yang lain.
2	*Terminator*		Digunakan untuk menunjukkan awal maupun akhir dari suatu proses atau aktivitas.
3	*Connector*		Digunakan sebagai penghubung alur proses halaman atau lembar kerja yang sama.

4	*Connector*		Berfungsi sebagai penghubung antar proses yang terletak pada halaman yang berbeda dalam suatu diagram alir
5	*Processing*		Menunjukkan pengolahan yang dilakukan oleh komputer
6	Manual *operation*		Menunjukkan pengolahan yang tidak dilakukan oleh komputer
7	*Decision*		Pemilihan proses berdasarkan kondisi yang ada
8	*Input-Output*		Digunakan untuk merepresentasikan proses masukan dan keluaran data.
9	Manual *input*		Pemasukan data secara manual on-line keyboard
10	*Perparation*		Digunakan untuk menunjukkan proses penyiapan media penyimpanan yang akan dimanfaatkan dalam kegiatan pengolahan data.

11	*Predifine proses*		Untuk pelaksanaan suatu bagian (sub-program)/prosedur
12	*Display*		Menyatakan peralatan output yang digunakan yaitu layar, plotter, printer, dan sebagainya
13	*Disk and online storage*		Menyatakan *input* berasal dari disk atau disimpan ke disk
14	*Magnetic tape unit*		Menyatakan *input* berasal dari pita magnetik atau output disimpan ke pita magnetik
15	*Punch card*		Digunakan menunjukkan bahwa data masuk berasal dari kartu atau hasil keluar dihasilkan dalam bentuk kartu.
16	Dokumen		Digunakan untuk menunjukkan bahwa data masukan berasal dari dokumen fisik atau dokumen cetak.

b. Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa pemodelan standar yang dimanfaatkan dalam bidang rekayasa perangkat lunak untuk menggambarkan, merancang, serta mendokumentasikan

struktur dan perilaku suatu sistem secara visual. *Unified Modeling Language (UML)* merupakan bahasa untuk merancang sistem atau perangkat lunak yang berbasiskan objek (Wulandari & Nurmiati, 2022). Dengan adanya pemodelan ini, komunikasi antara analis, desainer, dan programmer dapat berlangsung lebih efektif karena semua pihak memiliki gambaran yang sama mengenai sistem yang akan dibangun.

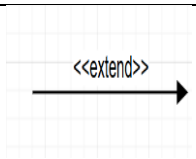
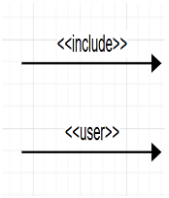
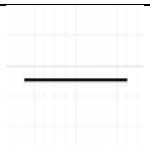
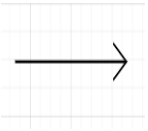
Unified Modeling Language (UML) memiliki berbagai jenis diagram yang digunakan dalam tahapan perancangan sistem, antara lain:

1) *Use Case Diagram*

Use Case Diagram digunakan untuk menggambarkan hubungan antara pengguna sistem dengan fungsi-fungsi *Use Case* yang tersedia di dalam sistem. *Use case diagram* biasanya dipakai untuk menggambarkan jenis aktivitas yang seharusnya dapat dilakukan oleh pengguna terhadap sistem yang dirancang (Binangkit et al., 2023). *Use case* merupakan representasi visual yang menguraikan kegunaan sistem dari perspektif *individu* yang berada di luar sistem atau actor. Adapun penjelasan *Use Case Diagram* dapat di lihat di tabel 2. 2.

Tabel 2. 2 *Use Case Diagram*

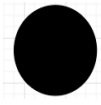
No	Notasi	Simbol	Keterangan
1	*Actor*		Pengguna sistem atau yang berinteraksi secara langsung dengan sistem

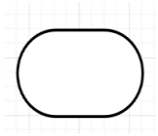
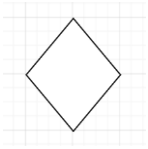
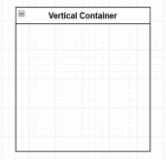
2	*Extend*		Relasi tambahan *Use Case* terhadap *Use Case* lain
3	*User/Include*		Relasi dua *Use Case* yang membutuhkan tambahan untuk menjalankan *Use Case*
4	*Assocation*		Interaksi yang terjadi terhadap *Actor* dan user
5	*Generalizatio n*		Menunjukan hubungan kearah *Use Case* yang lebih umum

2) *Activity Diagram*

Activity Diagram menunjukkan alur kerja dari suatu proses atau *Use Case*, menggambarkan urutan aktivitas, keputusan yang diambil, dan kondisi berpindah dari satu aktivitas ke aktivitas lainnya. Diagram aktivitas menggambarkan aliran kerja atau aktivitas sistem, bukan tindakan yang dilakukan oleh aktor; jadi aktivitas yang bisa dilaksanakan oleh sistem (Prasticha et al., 2022). Adapun penjelasan *Activity Diagram* dapat di lihat pada tabel 2. 3.

Tabel 2. 3 *Activity Diagram*

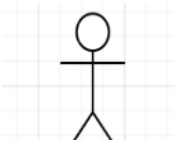
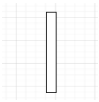
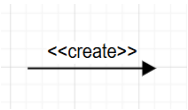
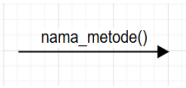
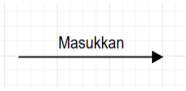
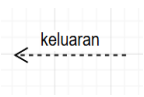
No	Notasi	Simbol	Keterangan
1	Status awal/akhir		Merupakan status awal dan akhir, setiap aktivitas diagram

			memiliki status awal dan status akhir
2	Aktifitas		Sistem melakukan kegiatan yang dimulai dengan kata kerja
3	*Decision*		Hubungan yang terjadi saat kita memiliki beberapa pilihan untuk membuat Keputusan dalam suatu aktivitas
4	*Swimlane*		Merupakan pemisah organisasi bisnis yang memiliki tanggung jawab aktivitas yang terjadi
5	*Join*		Merupakan hubungan jika satu atau lebih aktivitas menjadi satu

3) *Sequence Diagram*

Sequence Diagram menunjukkan interaksi antar objek atau komponen sistem dalam suatu skenario tertentu secara kronologis. *Diagram* sekuen menunjukkan perilaku objek dengan menjelaskan waktu melalui pesan yang dikirim dan diterima di antara objek (Novianto & Purwanto, 2022). Adapun penjelasan *Sequence Diagram* dapat di lihat pada tabel 2. 4.

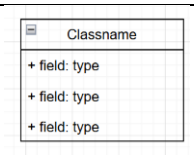
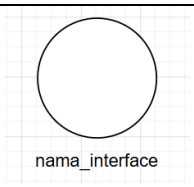
Tabel 2. 4 Sequence Diagram

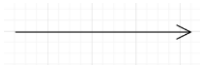
No	Notasi	Simbol	Keterangan
1	*Actor*		Merupakan orang yang berhubungan dengan sistem yang di buat
2	*Lifeline*		Garis hidup objek yang menerangkan kehidupan dari objek tersebut
3	Objek		Merupakan interaksi pesan yang dilakukan oleh objek
4	Waktu Aktif		Menandakan interaksi objek
5	Pesan tipe *create*		Peryataan suatu objek membuat objek lain
6	Pesan tipe *call*		Sebuah objek memanggil metode atau proses pada objek lain
7	Pesan tipe *send*		Sebuah objek mengirimkan informasi,masukan, atau data ke objek lain
8	Pesan tipe *return*		Pernyataan objek menjalankan suatu perintah dan memberi keluaran ke objek

4) *Class Diagram*

Class Diagram menggambarkan struktur statis sistem, termasuk kelas, atribut, metode, dan relasi antar kelas. Ini merupakan fondasi dari pemrograman berorientasi objek (OOP). *Diagram Class* berfungsi untuk menggambarkan struktur sistem yang mencakup kemampuan untuk mengelola data dan melaksanakan Tindakan (Wayahdi & Ruziq, 2023). Adapun penjelasan *Class Diagram* dapat di lihat pada tabel 2. 5.

Tabel 2. 5 *Class Diagram*

No	Notasi	Simbol	Keterangan
1	*Class*		Kelas pada sistem dan memiliki atribut dan operasi.
2	*Association*		Merupakan relasi antarkelas, yang dilengkapi dengan multiplicity
3	Generalisasi		Merupakan relasi kelas dengan bermakna generalisasi
4	*Dependency*		Merupakan relasi keberuntungan antar kelas
5	*Aggregation*		Merupakan relasi kelas bermakna semua bagian
6	*Interface*		Meruapakan bagian yang ditampilkan namun tidak ada isi.

7	*Directed Association*		Hubungan antar dua kelas di mana satu kelas digunakan oleh kelas lain dan memiliki multiplicity.

4. Pengujian Sistem

Black Box Testing pada penelitian ini diterapkan untuk menguji seluruh fungsi yang tersedia pada aplikasi Sistem Deteksi Tingkat Kesegaran Daging Ayam Menggunakan *You Only Look Once (YOLO)* Berbasis *Android*. Pengujian dilakukan terhadap setiap fitur utama aplikasi, meliputi proses membuka aplikasi, mengakses kamera, mengambil citra secara *real-time*, melakukan deteksi tingkat kesegaran daging ayam menggunakan model *YOLOv8*, menampilkan hasil deteksi, serta memastikan seluruh fitur aplikasi dapat berfungsi sesuai dengan kebutuhan fungsional yang telah dirancang. Metode ini bertujuan untuk menguji fungsionalitas sistem dengan memanfaatkan analisis terhadap *input* dan *output* tanpa melakukan pengecekan kode sumbernya (Saputra & Mardianti, 2025).

Penerapan metode Black Box Testing pada penelitian ini bertujuan untuk memastikan bahwa aplikasi dapat digunakan dengan baik dari sudut pandang pengguna. Keunggulan dari pengujian ini adalah bahwa penguji tidak diharuskan memiliki pemahaman mendalam mengenai bahasa pemrograman maupun pengetahuan tentang penerapan (Putri et al., 2024). Dengan demikian, hasil pengujian ini dapat menjadi indikator bahwa sistem

yang dikembangkan telah memenuhi kebutuhan fungsional, serta siap digunakan sebagai media untuk membantu proses deteksi tingkat kesegaran daging ayam secara cepat dan akurat menggunakan perangkat Android. Perhitungan persentase keberhasilan dilakukan dengan membandingkan jumlah test case yang berhasil dijalankan dengan jumlah seluruh test case yang diuji. Adapun rumus yang digunakan dapat dilihat di persamaan 2.6.

$$\text{Persentase Keberhasilan} = \frac{\sum \text{Test Case Berhasil}}{\sum \text{Test Case}} \times 100 \quad (2.6)$$

B. Kajian Empiris

Penelitian yang dilakukan oleh Priana & Karyawati (2023) membahas penerapan *You Only Look Once (YOLO)* untuk mendeteksi sampah secara *real-time*. Hasil penelitian menunjukkan bahwa metode *You Only Look Once (YOLO)* mampu mengidentifikasi jenis dan lokasi objek dengan baik berdasarkan *Dataset* yang telah dilatih, sehingga efektif digunakan dalam sistem deteksi objek berbasis citra.

Selanjutnya, penelitian oleh Harahap & Muliantara (2024) mengimple mentasikan model *You Only Look Once (YOLO)* untuk deteksi objek pada citra digital. Hasil penelitian menunjukkan bahwa *You Only Look Once (YOLO)* mampu menghasilkan nilai akurasi yang tinggi dengan nilai *mAP* mendekati 0,8 serta mampu mendeteksi objek secara efektif dalam berbagai kelas.

Penelitian yang dilakukan oleh Setiawan et al. (2024) mengimplementa sikan *Convolutional Neural Network (CNN)* untuk mengklasifikasikan kesegar an daging ayam berdasarkan citra digital. *Dataset* gambar daging ayam segar dan tidak segar diperoleh melalui dokumentasi manual.

Berdasarkan penelitian tersebut, dapat disimpulkan bahwa metode *You Only Look Once (YOLO)* memiliki performa yang baik dalam deteksi citra, khususnya pada objek daging ayam. Namun, sebagian penelitian masih terbatas pada model saja dan belum banyak diimplementasikan dalam aplikasi *Android*. Oleh karena itu, penelitian ini mengembangkan sistem deteksi tingkat kesegaran daging ayam berbasis *Android* untuk meningkatkan kemudahan penggunaan serta akurasi hasil.

C. Kerangka Berpikir

Penelitian ini diawali dari permasalahan yang terdapat pada usaha peternakan dan penjualan daging ayam yang dikelola oleh peneliti, di mana proses penentuan tingkat kesegaran daging ayam masih dilakukan secara manual dengan mengandalkan pengamatan visual seperti warna, tekstur, dan kondisi fisik daging. Metode tersebut sangat bergantung pada pengalaman penjual sehingga hasil penilaian cenderung subjektif, kurang konsisten, serta berpotensi menimbulkan kesalahan dalam menentukan kualitas daging ayam yang layak dikonsumsi.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pengembangan sistem deteksi tingkat kesegaran daging ayam berbasis *Android* dengan menggunakan *You Only Look Once (YOLO)*. Sistem ini dirancang untuk memanfaatkan kamera *smartphone* sebagai alat untuk mengambil citra daging ayam, yang kemudian akan diproses secara otomatis oleh model *You Only Look Once (YOLO)*. Citra yang diperoleh terlebih dahulu melalui tahapan *preprocessing*, kemudian diekstraksi fitur-fiturnya oleh *You Only Look Once (YOLO)* untuk mengenali pola visual seperti warna dan tekstur daging ayam.

Selanjutnya, hasil pengolahan citra tersebut diklasifikasikan ke dalam beberapa kategori tingkat kesegaran, yaitu segar dan busuk. Model *You Only Look Once (YOLO)* yang telah dilatih menggunakan *Dataset* citra daging ayam akan digunakan untuk melakukan prediksi terhadap citra baru secara cepat dan akurat. Hasil klasifikasi kemudian ditampilkan melalui aplikasi berbasis *Android* sehingga pengguna dapat dengan mudah mengetahui tingkat kesegaran daging ayam hanya dengan mengambil gambar menggunakan *smartphone*.

Gambar 2. 4 Kerangka Berpikir.