BAB II
TINJAUAN PUSTAKA

2.1. Teori Terkait

Dalam penelitian ini, beberapa penelitian terdahulu yang memiliki keterkaitan dengan topik yang dibahas digunakan sebagai referensi dalam proses penyusunan dan perancangan penelitian. Kajian tersebut dirangkum pada Tabel 2.1.

Tabel 2.1. Teori Terkait

No.	Rujukan (Penulis/Tahun)	Data, *Dataset*, Masukan	Model, Algoritma	Capaian / Hasil Evaluasi	*Research* Gap
1	Agarwal, Choudhury, Ahuja, Sarkar (Journal of Food Processing and Preservation, 2023) [46] Hybrid Deep Learning Algorithm-Based Food Recognition and Calorie Estimation.	Dataset: Indian FoodNet-30 (±5.500 Citra, 30 kelas makanan). Citra di-resize menjadi 640×640; split 80% untuk training dan 10% untuk testing 10%.	Arsitektur hybrid: Mask R-CNN untuk segmentasi; fitur dari hasil segmentasi diklasifikasikan memakai YOLOv5. Parameter simulasi: batch size 2; learning rate 0,001; optimizer Adam; epochs 25; device CPU.	Akurasi sistem yang dilaporkan 97,12%. •Performa (Mask R-CNN+YOLOv5): accuracy 0,97125; precision 0,96415; F-score 0,96584; sensitivity 0,96754; specificity 0,96452.	Estimasi volume bergantung pada objek referensi dan asumsi bentuk "bowl-like". Keterbatasan yang diakui variasi tipe/bentuk/ukuran/ presentasi makanan masih menangtang.
2	Ferreira, Cerqueira, Ribeiro	Dataset: 175 citra dasar piring	YOLOv11 untuk deteksi/se	mAP = 0.343; precision	Fokus utama bukan

	(Applied Sciences, 2025) [47] Food Waste Detection in Canteen Plates Using YOLOv11.	kantin, 860 citra setelah augmentasi (zoom/crop/ rotate/flip). Labeling memakai Roboflow.	gmentasi pada piring kantin.	= 0.62; recall = 0.322.	estimasi porsi. Tidak menguji dampak preprocess ing terhadap akurasi & waktu respons.
3	Chrintz-Gath, Daivadanam, Matta, McKeever (JMIR Formative Research, 2025) [48] Image-Based Dietary Assessment Using the Swedish Plate Model: Evaluation of Deep Learning–Based You Only Look Once (YOLO) Models.	Dataset: 3707 citra berlabel, 42 kelas makanan. Sumber data: Roboflow Universe dan Kaggle. Split: 3570 train, 131 validation, 6 test. Preprocessi ng: Auto-Orient, Resizing, Auto-Adjust Contrast; augmentasi: Rotation (±15°), Grayscale (10%), Noise.	Perbandin gan model: YOLOv7 vs YOLOv8 vs YOLOv9. Metrik evaluasi: precision, recall, mAP@0.5 , mAP@0.5 :0.95, F1-score.	Hasil terbaik YOLOv8 : precision 0.82382; recall 0.67553. mAP@0. 5 = 0.70447; mAP@0. 5:0.95 = 0.53606; F1-score 0.63.	Test set sangat kecil = 6 citra Fokus pada identifikas i, belum ke estimasi berat.
4	Gonzalez, Garcia, Velastin, GholamHoss eini, Tejeda, Farias (Sensors, 2024) [49]	Dataset segmentasi: 1025 citra, 14 kelas Estimasi berat nasi: 48 citra porsi nasi,	YOLOv8L -Seg untuk segmentas i piksel pada objek makanan.	Segment asi: mAP@0. 5 = 0.895. Deteksi: mAP@0. 5 = 0.873 .	Pendekata n porsi/berat bergantun g pada depth camera. Belum ada

	Automated Food Weight and Content Estimation Using Computer Vision and AI Algorithms.	tiap porsi ditimbang.			analisis dampak resize/kompresi citra terhadap akurasi dan waktu respons.
5	S., Rajesh Kumar; Prem Kumar, Josephine (Engineering, Technology & Applied Science Research, 2025) [50] Food Object Detection Using Custom-Trained YOLOv8 with Roboflow Integration.	Dataset: 1,800 gambar high-resolution untuk 6 kelas makanan. Pembagian data: train 70% (1,260), validation 15% (270), test 15% (270).	Model: YOLOv8 + Roboflow pipeline. Konfigurasi training: input 416×416; batch size 64, learning rate 0.001, momentum 0.9, max batches 6000, classes = 6.	Validasi: mAP@0.5 dilaporkan > 92% (rentang mAP@0.5 per kelas ~92.2%–97.8%). Kecepatan inferensi: >25 FPS pada GPU NVIDIA T4.	Fokus hanya deteksi bbox Belum membahas estimasi porsi/berat/kalori

2.2. Landasan Teori

Berdasarkan kebutuhan dan ruang lingkup penelitian, landasan teori pada penelitian ini meliputi konsep-konsep dasar yang berkaitan dengan metode dan teknologi yang digunakan, yaitu:

2.2.1. Pembelajaran Mendalam (Deep Learning)

Pembelajaran mendalam (*Deep Learning*) adalah bagian dari *machine learning* yang memanfaatkan jaringan saraf tiruan berlapis-

lapis (*deep neural networks*) untuk mempelajari representasi data secara otomatis [51]. Menurut LeCun, Bengio, dan Hinton, *deep learning* mempunyai kemampuan untuk mengekstraksi fitur kompleks dari data visual, audio, dan teks tanpa membutuhkan ekstraksi fitur manual [52]. Teknologi *deep learning* berkembang pesat seiring meningkatnya kemampuan komputasi *GPU* dan optimalisasi arsitektur model [53].

2.2.2. Komputer Visi (Computer Vision)

Komputer Visi (*Computer Vision*) adalah cabang ilmu komputer yang mempelajari cara agar mesin dapat "melihat" dan menafsirkan informasi visual dari citra atau video menjadi representasi yang bermakna untuk pengambilan keputusan [54]. *Computer vision* merupakan ilmu komputer yang bekerja dengan cara meniru kemampuan visual manusia, yaitu mengembangkan sistem yang sangat mirip dengan fungsi umum sistem visual manusia [55],[56]. Pengolahan citra adalah metode untuk mengatasi kesulitan dalam pemrosesan gambar, sementara *computer vision* bertugas membuat penilaian terhadap objek fisik berdasarkan informasi visual yang diterima [57]. Bidang ini menggabungkan teknik pengolahan citra digital (PCD), pembelajaran mesin, dan kecerdasan buatan (AI) untuk memungkinkan komputer memahami dan mengenali objek visual, seperti pada tugas klasifikasi gambar, deteksi, segmentasi objek, dan

pelacakan. Perkembangan *computer vision* didukung oleh arsitektur jaringan saraf seperti CNN dan model *multimodal* berskala besar, namun tantangan terkait efisiensi komputasi, *interpretabilitas*, dan generalisasi masih menjadi fokus penelitian [58].

2.2.3. Citra Digital

Citra digital merupakan hasil pengolahan citra yang diproses menggunakan komputer dan disajikan dalam bentuk numerik melalui proses digitalisasi data citra analog ke digital, sehingga memungkinkan pemanfaatannya dalam berbagai bidang dan proses pengolahan lanjutan [59]. Citra digital digunakan dalam berbagai disiplin ilmu, seperti seni, penglihatan manusia, astronomi, dan teknik, serta dapat diproses dan dianalisis menggunakan komputer [60]. Citra dalam perwujudan dapat bermacam-macam, mulai dari gambar perwujudannya dapat bermacam–macam, mulai dari gambar putih pada sebuah foto yang tidak bergerak sampai pada gambar warna yang bergerak pada televisi, proses transformasi dari bentuk tiga dimensi ke bentuk dua dimensi untuk menghasilkan citra akan dipengaruhi oleh bermacam-macam faktor yang mengakibatkan penampilan citra suatu benda tidak sama persis dengan bentuk fisik nyatanya [61]. Citra digital dapat dihasilkan melalui kamera, *scanner*, sensor, atau sistem pencitraan medis dan digunakan secara luas dalam fotografi, kedokteran, keamanan, serta bidang *computer vision*. Pada

proses pengolahan citra, citra digital dapat mengalami beberapa tahapan seperti peningkatan kualitas (*image enhancement*), reduksi *noise*, segmentasi objek, transformasi geometris, serta ekstraksi fitur. Tahapan pengolahan citra digital pada umumnya adalah akuisisi citra, peningkatan kualitas citra, segmentasi citra, ekstraksi fitur citra, dan klasifikasi citra [62].

2.2.4. Open CV

Perpustakaan komputer visi OpenCV (*Open Source Computer Vision Library*) adalah perpustakaan perangkat lunak sumber terbuka yang menyediakan lebih dari ribuan fungsi untuk pengolahan citra dan visi komputer, termasuk operasi dasar seperti membaca dan menulis gambar, konversi warna, memfilter, transformasi geometris, deteksi fitur, pelacakan objek, kalibrasi kamera, serta antarmuka untuk menjalankan model *machine learning* dan *deep learning* [63]. OpenCV adalah pustaka pemrograman yang dirancang untuk mendukung berbagai kebutuhan pengolahan citra dan *computer vision*. OpenCV dapat digunakan secara bebas karena didistribusikan dengan lisensi sumber terbuka Berkeley Software Distribution (BSD) [64]. OpenCV dirancang agar efisien dan mampu beroperasi secara *real-time*, sehingga banyak dimanfaatkan dalam berbagai aplikasi industri, kegiatan penelitian, serta pengembangan prototipe akademik [65].

2.2.5. Bahasa Pemrograman Python

Bahasa pemrograman Python merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan dalam pengembangan perangkat lunak dan berbagai bidang komputasi [66]. Bahasa ini diciptakan oleh Guido van Rossum di Stichting Mathematisch Centrum (CWI), Amsterdam pada tahun 1991 [67]. Salah satu karakteristik yang membedakan Python dari bahasa pemrograman lainnya adalah pengembangannya yang melibatkan komunitas global yang sangat luas, terdiri dari *programmer*, peneliti, akademisi, dan berbagai pengguna dari beragam latar belakang, hal ini dimungkinkan karena Python merupakan bahasa pemrograman yang bersifat open *source* [68].

2.2.6. Flask

Dalam pengembangan aplikasi, pemilihan *framework* yang ringan dan fleksibel menjadi salah satu pertimbangan penting. Flask adalah salah satu *microframework* berbasis Python yang digunakan untuk membangun aplikasi *web* secara sederhana, fleksibel, dan modular [69]. *Framework* ini menyediakan fitur inti yang ringan dan mendukung pengembangan aplikasi secara efisien dengan penggunaan sumber daya yang rendah [70]. Berbeda dengan *framework* yang lebih besar seperti Django, Flask tidak menyediakan struktur baku untuk *autentikasi*, ORM, atau manajemen formulir,

sehingga pengembang dapat menyesuaikan aplikasi secara lebih bebas [71]. Flask dikenal sebagai *framework* yang ringan dan memiliki kinerja yang cepat karena dirancang dengan konsep yang sederhana dan minimalis. Dengan slogan "*web development, one drop at a time*", yang memungkinkan pengembang membangun aplikasi *web* secara cepat dan efisien meskipun hanya menggunakan komponen serta pustaka yang sederhana [72].

2.2.7. Google Collaboratory

Layanan *Google Colab* (Google Colaboratory) merupakan platform berbasis *web* yang disediakan oleh Google untuk menulis, menjalankan, dan membagikan kode Python secara interaktif melalui lingkungan *notebook online* [73]. *Google Colab* menyediakan lingkungan komputasi berbasis *cloud* sehingga pengguna dapat menjalankan kode tanpa perlu melakukan instalasi perangkat lunak di komputer pribadi [74]. Platform ini sangat mendukung berbagai aktivitas seperti pemrosesan data, eksperimen *machine learning*, visualisasi, serta penelitian akademik karena menyediakan akses gratis ke *GPU* dan *TPU* untuk mempercepat komputasi [75]. *Google Colab* sangat menarik perhatian karena memberikan pengguna akses gratis ke layanan *GPU* sebagai *backend* komputasi yang dapat digunakan hingga 12 jam sekaligus, sehingga bagi pengguna yang ingin belajar Python dan *machine learning* tidak perlu membeli

komputer dengan *GPU* tambahan [76]. Selain itu, *Google Colab* memudahkan kolaborasi karena *notebook* dapat dibagikan dan disunting bersama secara *real-time* melalui integrasi dengan Google Drive. *Google Colab* juga menawarkan fleksibilitas dan *skalabilitas* yang lebih baik dalam pemrosesan data besar [77].

2.2.8. Roboflow

Platform Roboflow merupakan layanan berbasis *web* yang menyediakan alat untuk mengelola, memproses, dan mempersiapkan *dataset* gambar yang digunakan dalam pelatihan model *computer vision* [78]. Roboflow merupakan platform yang dirancang khusus untuk membantu pengelolaan *dataset* dan berperan penting dalam tahap persiapan data sebelum proses pelatihan model [79]. Roboflow memungkinkan pengguna membagikan dan mengelola *dataset*, melakukan proses *annotate* atau penandaan objek menggunakan *bounding box*, serta menerapkan *pre-processing* pada *dataset* sebagai bagian dari pengolahan data [80]. Dengan menggunakan antarmuka yang intuitif dan otomatisasi berbasis *cloud*, Roboflow memungkinkan pengguna mengelola ribuan gambar tanpa memerlukan perangkat keras khusus [81]. Selain itu, Roboflow menyediakan alat pendukung pengembangan model serta *API* untuk integrasi aplikasi dan banyak digunakan karena mampu

menyederhanakan alur kerja pengembangan model hingga *deployment*.

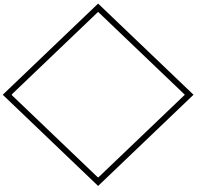
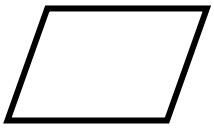
2.2.9. Diagram Alir

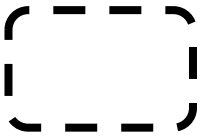
Diagram alir atau *flowchart* merupakan bentuk representasi visual yang digunakan untuk menggambarkan urutan langkah dan logika dari suatu proses atau algoritma menggunakan simbol-simbol tertentu yang saling terhubung melalui garis aliran. *Flowchart* tidak hanya berfungsi sebagai media komunikasi, tetapi juga membantu pemrogram dalam merancang serta mengimplementasikan algoritma secara lebih terstruktur, sehingga dapat mempermudah proses pemecahan masalah, khususnya bagi pemrogram pemula [82]. Setiap tahapan proses digambarkan menggunakan simbol yang memiliki fungsi berbeda dan dihubungkan dengan anak panah untuk menunjukkan arah aliran proses secara berurutan dari awal hingga akhir [83].

Diagram alir banyak dimanfaatkan dalam berbagai bidang, seperti pengembangan perangkat lunak, rekayasa sistem, hingga penelitian ilmiah, karena mampu menyederhanakan proses yang kompleks agar lebih mudah dipahami secara visual [84]. Dalam penelitian ini, *flowchart* digunakan untuk menggambarkan alur kerja sistem, mulai dari proses pengembangan model, proses deteksi objek,

hingga proses estimasi berat nasi. Simbol-simbol yang digunakan pada *flowchart* disajikan pada Tabel 2.2.

Tabel 2.2. Simbol Flowchart

No.	Gambar	Simbol	Keterangan
1.		Proses/Langkah	Menyatakan kegiatan yang akan ditampilkan dalam diagram alir.
2.		Titik Keputusan	Proses/langkah dimana perlu adanya keputusan atau adanya kondisi tertentu. Di titik ini selalu ada dua keluaran untuk melanjutkan kondisi berbeda.
3.		Masukkan/Keluaran Data	Digunakan untuk mewakili data masuk atau data keluar dari suatu proses.
4.		Terminasi	Menunjukkan awal atau akhiran sebuah proses.

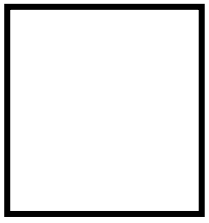
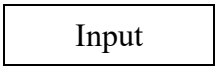
5.		Garis Aliran	Menunjukkan arah aliran proses dari satu langkah ke langkah berikutnya.
6.		Batas Subproses	Menandai batas wilayah logis dari suatu subproses atau kelompok proses dalam diagram alir untuk memperjelas tahapan yang saling berkaitan.

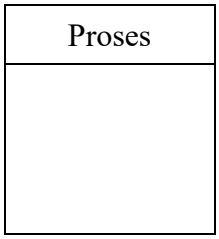
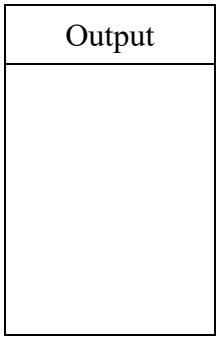
2.2.10. Diagram Blok

Diagram blok merupakan diagram yang tersusun dari beberapa kotak (blok) dan garis penghubung yang digunakan untuk menjelaskan alur kerja suatu sistem dalam bidang teknik. Diagram ini menggambarkan hubungan antar komponen sistem sehingga proses kerja sistem dapat dipahami dengan lebih mudah. Selain itu, diagram blok juga berfungsi sebagai panduan dalam menjelaskan mekanisme kerja sistem yang akan dibuat serta menjadi rancangan awal dalam proses pengembangan sistem [85].

Dalam penelitian ini, diagram blok digunakan untuk menunjukkan alur kerja sistem estimasi berat dan perhitungan kalori nasi berbasis YOLO26 *segmentation* dan *REST API* Flask, mulai dari tahap *input*, proses, hingga *output* sistem. Adapun penjelasan masing-masing komponen pada diagram blok sistem disajikan pada Tabel 2.3.

Tabel 2.3. Komponen Diagram Blok

No.	Simbol	Nama Komponen	Keterangan
1.		Blok	Merepresentasikan sebuah subsistem, modul, fungsi, atau komponen dalam sistem. Diberi label nama fungsi/komponen di dalamnya.
2.		Garis Penghubung	Menunjukkan arah aliran sinyal, data, atau energi antara dua blok. Arah panah menunjukkan arah transmisi informasi.
3.		*Input Block*	Blok yang menerima sinyal atau data dari luar sistem (misal: sensor

			MPU6050, PT100, GPS). Biasanya terletak di sisi paling kiri diagram.
4.		*Processing Block*	Blok yang melakukan operasi atau transformasi terhadap data (misal: ESP32-S3, FFT, model TinyML). Berada di tengah alur data sistem.
5.		*Output Block*	Blok yang menghasilkan keluaran akhir sistem (misal: hasil klasifikasi kondisi mesin, data MQTT ke platform monitoring). Terletak di sisi paling kanan diagram.