JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 1, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i1.8545

PENGEMBANGAN APLIKASI ESTIMASI KALORI MAKANAN BERBASIS CITRA DENGAN PENDEKATAN DETEKSI OBJEK MENGGUNAKAN YOLO

Rizqy Supriyadi[1*], Muhamad Irfan[2], Rizki Dwi Hapijar[3], Fadil Abubakar[4], Rendy Saputra[5], Kus Supriyanto[6], Raihan Putra Dwiantara[7], Esron Rikardo Nainggolan[8], Herlambang Brawijaya[9]

[1,2,3,4,5,6,7]Universitas Bina Sarana Informatika; Jl. Kramat Raya No.98, RT.2/RW.9, Kwitang, Kec. Senen, Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta 10450; telp (021) 21231170
[8]Universitas Nusa Mandiri; Jl. Raya Jatiwaringin No.2, RT.8/RW.13, Cipinang Melayu, Kec. Makasar, Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta 13620; telp (021) 28534471

Keywords:
Food calorie estimation; YOLOv8; Object detection; Mobile application.

Corespondent Email:
risqysupriyadi12@gmail.com

Abstrak. Penelitian ini mengembangkan aplikasi estimasi kalori makanan berbasis citra untuk membantu pengguna memantau asupan energi secara praktis melalui foto ponsel. Sistem menggunakan deteksi objek YOLOv8n untuk mengenali makanan Indonesia dan memetakan tiap deteksi ke parameter nutrisi guna menghitung massa dan kalori. Dataset pelatihan berisi 3.772 citra pada 9 kelas makanan (dibagi 80% latih, 10% validasi, 10% uji). Model dilatih selama 100 epoch pada resolusi 640 piksel menggunakan optimizer AdamW dan *early stopping*. *Backend* FastAPI dalam lingkungan Docker menjalankan inferensi dan perhitungan kalori berdasarkan data nutrisi tiap kelas. Aplikasi mobile Flutter mengirim citra ke *endpoint* /predict dan menampilkan makanan terdeteksi beserta *confidence*, estimasi massa, dan total kalori. Hasil uji menunjukkan performa deteksi tinggi dengan mAP@0.5 0,975, sementara kesalahan terbesar terjadi pada kelas yang mirip secara visual atau minim data. Temuan ini menegaskan bahwa sistem *end-to-end* mampu mengestimasi kalori otomatis dari satu foto dan layak dikembangkan lebih lanjut dengan menambah kelas dan menyeimbangkan dataset.

Abstract. *This study develops an image-based food calorie estimation app to help users conveniently monitor energy intake via smartphone photos. It uses YOLOv8n object detection to recognize Indonesian foods and maps each detection to nutritional parameters to estimate mass and calories. The model was trained on 3,772 images across 9 food classes (split 80/10/10 for train/val/test) for 100 epochs at 640px resolution using the AdamW optimizer with early stopping. A FastAPI backend in Docker performs inference and calorie calculation using a nutrition database of caloric density and typical portion sizes. The Flutter mobile app sends images to a /predict API endpoint and displays detected foods with confidence scores, estimated mass, and total calories. Experimental results show high detection performance with mAP@0.5 0.975, while most errors occur in visually similar or underrepresented classes. These findings indicate the end-to-end system can automatically estimate calories from a single photo and is suitable for further development by adding more classes and balancing the dataset.*

1. PENDAHULUAN

Perubahan gaya hidup masyarakat modern ditandai oleh meningkatnya aktivitas harian, pola kerja yang padat, dan kemudahan akses terhadap makanan cepat saji. Remaja cenderung memilih fast food karena praktis, mudah

ditemukan, rasanya disukai, serta didukung kemasan dan promosi yang menarik. Kondisi ini berkontribusi pada meningkatnya kasus obesitas dan penyakit tidak menular lain di kalangan remaja [1]. Di sisi lain, kemampuan masyarakat untuk menghitung dan mengendalikan asupan kalori harian masih terbatas karena perhitungan manual membutuhkan pengetahuan gizi serta waktu yang tidak sedikit, sehingga jarang dilakukan dalam aktivitas sehari-hari.

Perkembangan teknologi informasi membuka peluang untuk menghadirkan solusi yang lebih praktis. Pemanfaatan smartphone yang dilengkapi kamera beresolusi tinggi, dipadukan dengan kemajuan kecerdasan buatan (*artificial intelligence*) dan pembelajaran mesin (*machine learning*), memungkinkan sistem untuk mengenali objek pada citra secara otomatis [2], [3]. Pendekatan deep learning, khususnya algoritma deteksi objek *You Only Look Once* (YOLO), banyak digunakan karena mampu melakukan deteksi secara real time dengan akurasi yang cukup tinggi [4]. Karakteristik tersebut menjadikan YOLO sangat sesuai untuk tugas pengenalan makanan berbasis citra yang membutuhkan respon cepat dan dapat menangani beberapa objek makanan dalam satu gambar.

Berbagai penelitian telah mengkaji deteksi makanan dan estimasi kalori berbasis citra. Penelitian [5] mengembangkan sistem deteksi makanan Indonesia menggunakan YOLOv5 untuk memperkirakan informasi gizi per porsi berdasarkan data FatSecret Indonesia; model yang dihasilkan mencapai akurasi dan F1-score tinggi serta mampu menampilkan energi, protein, lemak, dan karbohidrat untuk setiap makanan yang terdeteksi. Penelitian [6] merancang *Food Image Detection System and Calorie Content Estimation Using YOLO*, yang menghitung estimasi kalori dengan mengalikan nilai kalori per porsi dengan jumlah objek makanan pada citra. Sementara itu penelitian oleh [7] mengusulkan sistem identifikasi makanan dan estimasi kalori berbasis YOLOv5 dan kamera kedalaman untuk memperkirakan kalori berdasarkan volume makanan, dengan tingkat keberhasilan pengenalan di atas 90% dan kesalahan estimasi kalori di bawah 10%. Penelitian lain oleh [8] memanfaatkan YOLOv3 untuk mendeteksi 31 jenis makanan khas Palembang dengan rata-rata akurasi 96% dan kecepatan deteksi sekitar 40 ms per citra, sedangkan [9] memperkenalkan dataset IndianFoodNet dan menunjukkan bahwa YOLOv8 memberikan kinerja terbaik untuk deteksi makanan khas India dengan metrik precision, recall, dan mAP yang tinggi.

Rangkaian studi tersebut memperlihatkan state of the art penerapan YOLO untuk deteksi makanan dan estimasi kalori, namun sebagian besar masih berfokus pada jenis makanan yang kurang merepresentasikan pola konsumsi masyarakat Indonesia dan berhenti pada level model tanpa integrasi penuh ke dalam aplikasi mobile yang siap pakai.

Beberapa sistem juga tetap mengandalkan input manual jenis dan porsi makanan sehingga kurang praktis dan berpotensi menurunkan kepatuhan penggunaan dalam jangka panjang. Penelitian *NutriChive* [10] membuktikan bahwa integrasi pembelajaran mesin ke dalam aplikasi mobile, dengan model yang dioptimasi menggunakan TensorFlow Lite untuk mendeteksi bahan makanan dan memberikan rekomendasi resep guna mengurangi limbah, dapat mencapai akurasi di atas 90% dengan waktu respon di bawah dua detik. Hal ini menegaskan bahwa pendekatan mobile machine learning efektif untuk menjembatani teknologi deteksi citra dan kebutuhan pengguna sehari-hari.

Berdasarkan kajian tersebut, masih terdapat celah pada pengembangan sistem estimasi kalori berbasis citra yang secara khusus menargetkan makanan yang umum dikonsumsi di Indonesia dan terintegrasi penuh ke dalam aplikasi mobile. Penelitian ini merancang dan membangun aplikasi estimasi kalori yang memanfaatkan YOLOv8 untuk mendeteksi beberapa jenis makanan Indonesia, menghubungkan hasil deteksi dengan basis data nilai kalori, dan menyajikan estimasi kalori total melalui layanan backend. Sistem yang dikembangkan mencakup model deteksi objek, pemetaan hasil deteksi ke nilai kalori, serta implementasi dan pengujian model, backend, dan antarmuka pengguna.

2. TINJAUAN PUSTAKA

2.1. Aplikasi Mobile

Aplikasi mobile merupakan perangkat lunak yang dijalankan pada ponsel pintar dan memanfaatkan mobilitas perangkat, koneksi

internet, serta integrasi dengan sensor dan kamera untuk mendukung berbagai aktivitas pengguna secara praktis dan tidak terikat ruang maupun waktu [11].

2.2. Flutter

Flutter adalah framework open source yang dikembangkan oleh Google untuk membangun antarmuka pengguna lintas platform (Android dan iOS) dengan basis satu kode program [12]. Framework ini menggunakan bahasa pemrograman Dart yang menerapkan konsep static typing sehingga struktur program menjadi lebih terkontrol dan mudah dipelihara [13].

2.3. Kecerdasan Buatan

Kecerdasan buatan (*Artificial Intelligence*) merupakan bidang ilmu yang berupaya membangun sistem komputer dengan kemampuan meniru aspek kecerdasan manusia, seperti penalaran logis, pengambilan keputusan, dan kemampuan belajar dari pengalaman [3]. Berbagai teknik AI meliputi sistem pakar, computer vision, machine learning, pengolahan bahasa alami, hingga deep learning yang memanfaatkan jaringan saraf berlapis untuk mempelajari representasi data yang kompleks.

2.4. Machine Learning

Machine learning atau pembelajaran mesin adalah cabang kecerdasan buatan yang memungkinkan sistem komputer belajar dari data untuk mengenali pola dan menghasilkan prediksi tanpa diprogram secara eksplisit untuk setiap aturan [2]. Pendekatan machine learning mencakup pembelajaran terawasi, tidak terawasi, dan pembelajaran penguatan, yang telah banyak diterapkan di berbagai sektor seperti industri dan layanan publik [14].

2.5. *You Only Look Once* (YOLO)

You Only Look Once (YOLO) merupakan algoritma deteksi objek berbasis deep learning yang dirancang untuk bekerja secara real time dengan pendekatan satu tahap (single-stage), di mana proses lokalisasi dan klasifikasi objek dilakukan dalam satu kali inferensi jaringan saraf [4]. Arsitektur YOLO memproyeksikan citra ke dalam grid dan secara bersamaan memprediksi koordinat kotak pembatas, skor kepercayaaan, serta kelas objek. Varian terbaru seperti YOLOv8 dilaporkan mampu menghasilkan akurasi tinggi dengan waktu respon yang singkat sehingga cocok untuk aplikasi yang menuntut deteksi cepat, termasuk pengenalan makanan [15].

2.6. Application Programming Interface (API)

Application Programming Interface (API) adalah sekumpulan aturan dan protokol yang memungkinkan dua atau lebih komponen perangkat lunak saling berkomunikasi dan bertukar data melalui format standar, biasanya berbasis HTTP dengan payload JSON atau XML [16].

2.7. FastApi

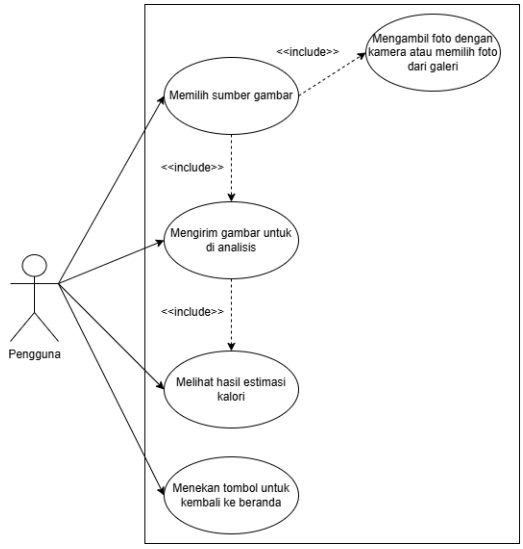
FastAPI adalah framework web modern berbasis Python yang dirancang untuk membangun layanan API berperforma tinggi dengan dukungan pemrograman asinkron dan dokumentasi otomatis melalui standar OpenAPI [17]. Framework ini terintegrasi dengan baik dengan berbagai pustaka Python untuk kecerdasan buatan, sehingga banyak digunakan untuk membungkus model deep learning ke dalam layanan web [18].

2.8. Google Colab

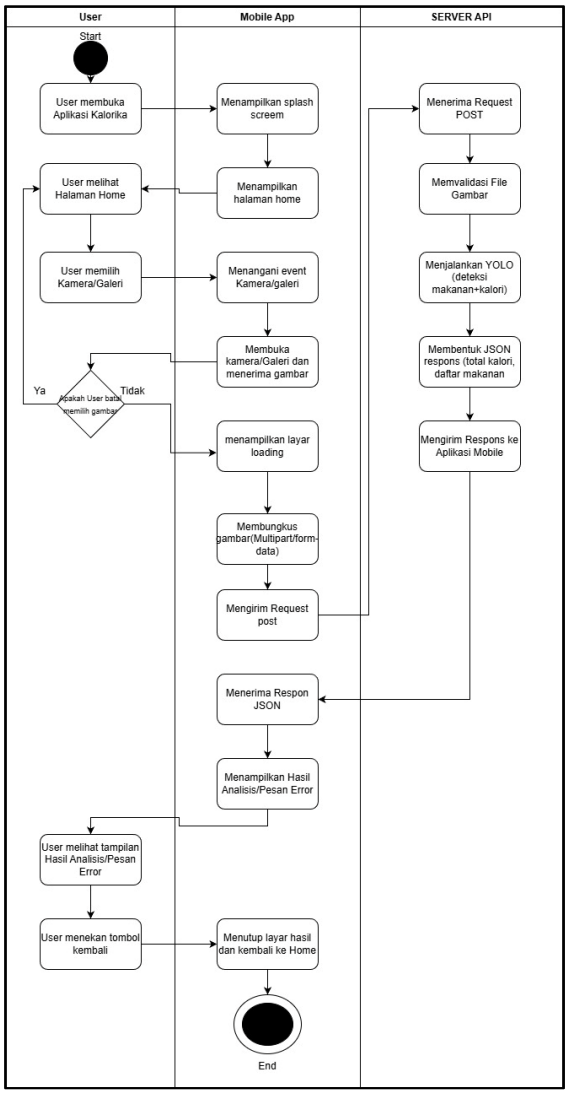
Google Colaboratory (Google Colab) adalah platform komputasi awan yang menyediakan lingkungan eksekusi Python lengkap dengan akses GPU dan TPU yang dapat dimanfaatkan untuk pelatihan model deep learning [19]. Platform ini banyak digunakan oleh peneliti dan mahasiswa karena tidak memerlukan instalasi lokal, sudah terintegrasi dengan berbagai pustaka machine learning, dan mendukung kolaborasi melalui berbagi notebook secara daring [20].

3. METODE PENELITIAN

Penelitian ini mengembangkan sistem estimasi kalori makanan berbasis citra yang bekerja end-to-end, mulai dari deteksi jenis makanan pada foto hingga perhitungan energi (kcal) dan penyajian hasil pada aplikasi mobile. Metode penelitian mencakup beberapa komponen utama: perancangan dan pelatihan model deteksi objek YOLOv8, pengembangan aplikasi mobile sebagai antarmuka pengguna, serta pembangunan backend FastAPI untuk proses inferensi dan perhitungan massa sampai kalori. Gambar 1 dan Gambar 2 berikut menggambarkan alur kerja sistem serta interaksi pengguna melalui Use Case Diagram dan Activity Diagram.



Gambar 1. Use Case Diagram



Gambar 2. Activity Diagram

3.1. Dataset

Dataset yang digunakan diperoleh dari *Roboflow Universe* (proyek "calorease-pxhre") dengan lisensi *public domain*. Dataset ini berisi total 3.772 citra yang telah dianotasi dengan *bounding box* untuk sembilan kelas makanan Indonesia. Daftar kelas makanan dan contoh item tiap kelas ditunjukkan pada Tabel 1. Pemilihan dataset didasarkan pada konsistensi label serta ketersediaan format ekspor YOLOv8 yang mempermudah proses pelatihan menggunakan pustaka Ultralytics.

Tabel 1. Daftar Kelas Makanan

No	Nama Makanan
1	ayam goreng
2	mie ayam
3	nasi putih
4	rendang
5	sambal
6	sate
7	soto
8	tahu goreng
9	tempe goreng

Dataset dibagi menjadi tiga subset dengan rasio 80% data latih, 10% validasi, dan 10% uji. Dari total 3.772 citra, pembagian ini menghasilkan kira-kira 3.021 citra latih, 376 citra validasi, dan 375 citra uji. Rasio tersebut dipilih untuk memberikan porsi pelatihan yang besar tanpa mengurangi kemampuan evaluasi generalisasi pada validasi dan uji, serta selaras dengan praktik umum pada penelitian deteksi objek berbasis YOLO. Distribusi kelas dijaga relatif serupa di setiap subset, dan *data leakage* dicegah dengan memastikan tidak ada citra duplikat lintas subset.

3.2. Pelatihan dan Evaluasi Model

Pelatihan model dilakukan di lingkungan Google Colab dengan memanfaatkan GPU. Arsitektur deteksi yang digunakan adalah YOLOv8 varian *nano* (YOLOv8n) karena berukuran ringan dan cocok untuk inferensi di server CPU dengan latensi rendah. Proses pelatihan dijalankan menggunakan ukuran citra 640×640 piksel, selama 100 epoch, *batch size* 16, serta menerapkan *early stopping* dengan

patience 20 epoch. Optimizer yang digunakan adalah AdamW dengan *learning rate* awal 0,001 dan *weight decay* 0,01 untuk menjaga stabilitas pembelajaran. Augmentasi data bawaan YOLO (misalnya *flip*, HSV, *mosaic*) diaktifkan untuk meningkatkan generalisasi model tanpa memerlukan *pipeline* augmentasi tambahan. Artefak hasil pelatihan (model bobot terbaik best.pt, log metrik, kurva PR, kurva F1-Confidence, serta *confusion matrix*) disimpan otomatis pada direktori hasil pelatihan.

Evaluasi model dilakukan pada subset uji (10% data) yang tidak pernah dilihat model selama pelatihan. Parameter inferensi disamakan dengan konfigurasi *backend*, yaitu ambang kepercayaan (confidence threshold) 0,25 dan ambang IoU untuk NMS 0,45. Metrik utama yang dicatat meliputi *precision*, *recall* dan mAP@0.5. Selain metrik agregat, analisis kesalahan dilakukan dengan meninjau *confusion matrix* untuk mengidentifikasi pasangan kelas yang sering tertukar dan memahami penyebabnya (misalnya kemiripan visual atau ketidakseimbangan jumlah data antar kelas).

3.3. Backend dan Perhitungan Kalori

Sistem *backend* dikembangkan menggunakan FastAPI dan dideploy pada platform Railway dalam bentuk kontainer Docker. Model YOLOv8 dimuat satu kali saat *startup* server untuk menghindari overhead pemuatan berulang. *Endpoint* utama, yaitu POST /predict, menerima berkas citra (melalui *multipart/form-data*), menjalankan inferensi YOLO, kemudian menghitung estimasi massa dan energi (kalori) untuk setiap objek makanan terdeteksi berdasarkan parameter gizi yang terdefinisi pada berkas konfigurasi nutrition.json. Untuk menjaga fleksibilitas *deployment* tanpa perlu mengubah kode, seluruh parameter konfigurasi backend (pemuatan model, proses inferensi, perangkat eksekusi, serta skala konversi luas, massa, dan kalori) didefinisikan sebagai *environment variables* sebagaimana dirangkum pada Tabel 2.

Tabel 2. Environment Variables

NO	Variable	Keterangan
1	MODEL_PATH	Lokasi berkas bobot YOLO yang dimuat saat start. Contoh: 'models/best.pt'.
2	NUTRITION_PATH	Lokasi berkas konfigurasi kalori dan parameter fisik per kelas. Contoh: `nutrition.json`.
3	CONF_THRESHOLD	Ambang kepercayaan deteksi untuk menyaring prediksi ber-noise. Contoh: `0.25`.
4	IOU_THRESHOLD	Ambang Intersection over Union untuk Non-Max Suppression. Contoh: `0.45`.
5	DEVICE	Target perangkat inferensi. Nilai umum: `cpu` atau indeks GPU seperti `0`.
6	PLATE_DIAMETER_CM	Diameter piring default (sentimeter) untuk mengaproksimasi luas proyeksi objek pada perhitungan massa. Contoh: `26`.

Perhitungan massa dan kalori dilakukan dengan memanfaatkan keluaran model YOLO berupa koordinat kotak batas (*bounding box*) untuk setiap objek makanan pada citra. Untuk setiap kelas makanan, berkas nutrition.json menyimpan parameter nutrisi yang dirangkum pada Tabel 3.

Tabel 3. parameter nutrisi

NO	Parameter	Keterangan
1	kcal_per_g	kalori per gram
2	density_g_per_cm3	massa jenis rata-rata
3	thickness_cm	ketebalan efektif hidangan
4	bbox_fill	fraksi kotak yang diisi makanan (sisanya latar)
5	packing_factor	faktor kerapatan penyusunan di piring

Secara konseptual, perhitungan dilakukan melalui beberapa tahap. Pertama, sistem menghitung luas objek pada citra relatif terhadap luas citra atau piring referensi. Kedua,

luas tersebut dikonversi menjadi luas sebenarnya dari objek makanan pada piring dengan mengasumsikan diameter piring (misalnya 26 cm) dan memperhitungkan nilai bbox_fill. Ketiga, volume makanan diperoleh dengan mengalikan luas makanan pada piring dengan ketebalan rata-rata (thickness_cm) dan faktor penataan (packing_factor). Keempat, volume diubah menjadi massa (gram) menggunakan informasi massa jenis (density_g_per_cm3). Terakhir, energi (kalori) dihitung dengan mengalikan massa dengan nilai kalori per gram (kcal_per_g). Seluruh konstanta dan rumus disesuaikan untuk setiap kelas makanan sehingga estimasi kalori yang dihasilkan lebih sesuai dengan karakteristik masing-masing makanan.

3.4. Estimasi Massa dan Kalori

Estimasi kalori diperoleh dengan menggunakan keluaran YOLO berupa kotak pembatas untuk setiap objek makanan pada citra. Untuk setiap kelas makanan, berkas nutrition.json memuat parameter nutrisi sebagaimana dirangkum pada Tabel 3. Secara garis besar, tahapan perhitungan kalori adalah sebagai berikut:

1. Rasio area kotak terhadap citra

$$A_{\text{img}} = \text{W} \times \text{H}, \quad A_{\text{bbox}} = (X_2 - X_1)\,(Y_2 - Y_1),$$

$$\text{r} = \frac{A_{bbox}}{A_{img}}$$

2. Luas Piring Bundar

$$A_{plate} = \pi \left(\frac{d}{2}\right)^2$$

3. Aproksimasi luas makanan pada piring

$$A_{food} = r \times A_{plate}$$

4. Volume

$$V_{food} = A_{food} \times thickness\_cm \times bbox\_fill$$

5. Massa

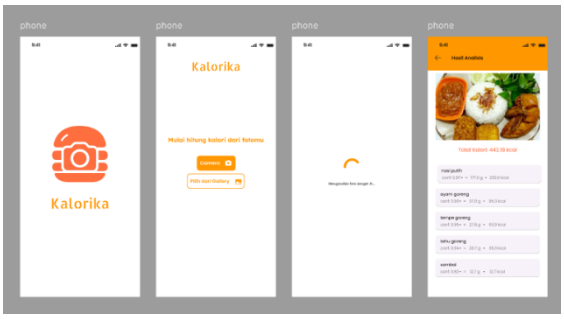
$$grams = V_{food} \times density\_g\_per\_cm^3 \times packing\_factor$$

6. Kalori

$$kcal = grams \times kcal\_per\_g$$

3.5. Rancangan Aplikasi

Aplikasi Kalorika dibangun menggunakan Flutter sebagai antarmuka utama pengguna. Fitur utama aplikasi meliputi pengambilan gambar melalui kamera atau pemilihan dari galeri, pengiriman gambar ke *backend*, tampilan *loading* selama proses inferensi, serta halaman hasil yang menampilkan daftar makanan terdeteksi beserta nilai *confidence*, estimasi massa, kalori per item, dan total kalori. Aplikasi berperan sebagai klien ringan yang mengandalkan komputasi di server, sehingga perangkat pengguna hanya menangani antar muka dan komunikasi data. Gambar 3 memperlihatkan salah satu *mockup* antarmuka aplikasi Kalorika.



Gambar 3. Tampilan mockup awal antarmuka aplikasi Kalorika.

Pada Gambar 3 terlihat halaman utama (Home Screen) aplikasi Kalorika dengan judul aplikasi, tagline “Ayo hitung kalori dari fotomu!”, serta dua tombol utama: “Camera” dan “Pilih dari Gallery”. Melalui halaman ini, pengguna dapat memilih untuk mengambil foto makanan secara langsung atau memuat gambar dari galeri perangkat. Jika tombol Camera dipilih, aplikasi akan meminta izin akses kamera dan membuka antarmuka kamera untuk mengambil foto. Sebaliknya, opsi Gallery akan membuka pemilih berkas gambar dengan izin akses penyimpanan. Setelah pengguna memperoleh atau memilih foto makanan, aplikasi membentuk permintaan HTTP POST berisi file image tersebut (multipart/form-data) dan mengirimkannya ke server backend melalui endpoint /predict. Selama proses ini, aplikasi menampilkan layar pemrosesan (Loading Screen) dengan pesan “Menganalisis foto dengan AI…” dan animasi indikator, menandakan bahwa citra sedang diunggah dan dianalisis di server. Setelah server

mengembalikan respons, aplikasi secara otomatis mengarahkan pengguna ke halaman hasil (Result Screen).

4. HASIL DAN PEMBAHASAN

Pada bagian ini memaparkan hasil pelatihan dan pengujian model YOLOv8n untuk deteksi sembilan kelas makanan Indonesia, serta pembahasan kinerja model dan validasi implementasi sistem end-to-end pada aplikasi Kalorika.

4.1. Implementasi Pelatihan Model

Proses pelatihan model YOLOv8n dilaksanakan di Google Colab dengan memanfaatkan GPU Nvidia T4. Setelah pustaka Ultralytics YOLOv8 terinstal, pelatihan dijalankan menggunakan *script* perintah bawaan. Gambar 4 menunjukkan cuplikan konfigurasi pelatihan model yang digunakan pada penelitian ini, termasuk parameter epoch, *batch size*, *optimizer*, *learning rate*, dan augmentasi.

```
!yolo detect train \
  model=yolov8n.pt \
  data="/content/drive/MyDrive/Kalorika/data.yaml" \
  epochs=100 \
  imgsz=640 \
  batch=16 \
  workers=2 \
  patience=20 \
  optimizer=AdamW lr0=0.001 lrf=0.01 weight_decay=0.01
```

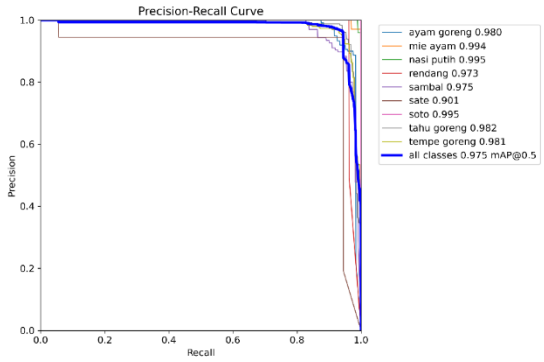
Gambar 4. Source Code Pelatihan Model

Selama proses pelatihan, model secara bertahap mencapai kinerja yang semakin baik. Bobot model terbaik (best.pt) dipilih berdasarkan performa pada data validasi. Selain itu, pelatihan menghasilkan berbagai artefak evaluasi seperti kurva *loss*, *precision-recall curve* (PR curve), kurva F1 terhadap *confidence*, kurva *precision* terhadap *confidence*, kurva *recall* terhadap *confidence*, serta *confusion matrix*. Secara umum, tren *training loss* tercatat menurun stabil dan metrik validasi meningkat seiring bertambahnya epoch, yang mengindikasikan proses pembelajaran berhasil konvergen pada dataset multikelas ini. Pemilihan arsitektur YOLOv8n terbukti mampu mempertahankan efisiensi komputasi, sehingga model dapat dijalankan dengan cepat pada backend berbasis CPU tanpa mengorbankan akurasi.

4.2. Hasil Pengujian Model

1. Precision–Recall Curve

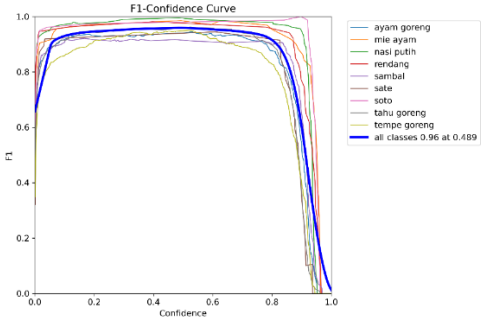
Evaluasi pada subset uji (375 citra) menggunakan model YOLOv8n terbaik menunjukkan hasil deteksi yang sangat baik. Model mencapai mAP@0.5 sebesar 0,975 (97,5%) secara keseluruhan. Kinerja per kelas juga konsisten tinggi, antara lain: *ayam goreng* (mAP 0,980), *mie ayam* (0,994), *nasi putih* (0,995), *rendang* (0,973), *sambal* (0,975), *sate* (0,901), *soto* (0,995), *tahu goreng* (0,982), dan *tempe goreng* (0,981). Gambar 5 memperlihatkan kurva Precision-Recall (PR) untuk seluruh kelas makanan, yang menunjukkan area di bawah kurva yang luas dan menegaskan kualitas deteksi yang tinggi pada data uji.



Gambar 5. Kurva Precision-Recall

2. F1- confidence Curve

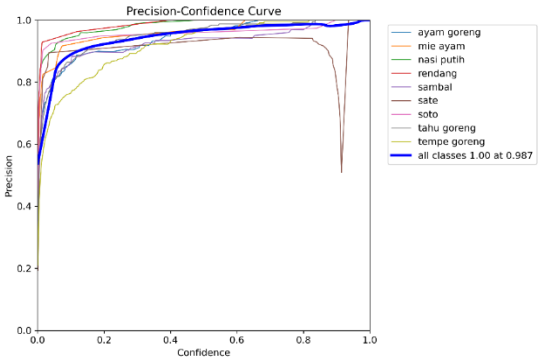
Selain itu, kurva F1 terhadap confidence memperlihatkan titik optimum keseimbangan presisi dan recall berada di sekitar nilai confidence 0,48 dengan F1 agregat ±0,96. Artinya, ambang confidence 0,5 dapat dijadikan acuan awal untuk kompromi terbaik antara ketepatan dan jangkauan deteksi. Secara visual, mayoritas kelas mempertahankan skor F1 tinggi pada rentang confidence menengah, kemudian menurun tajam ketika confidence mendekati 0,9–1,0 akibat semakin sedikitnya deteksi yang lolos ambang tinggi. Gambar 6 menunjukkan kurva F1–Confidence untuk seluruh kelas.



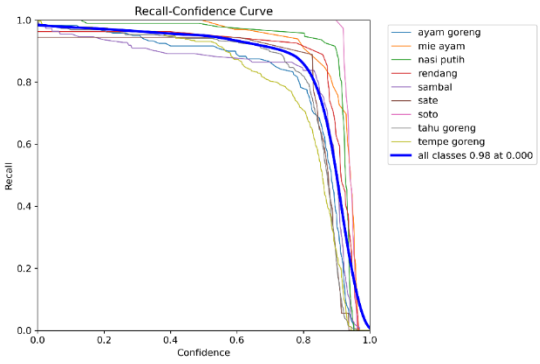
Gambar 6. Kurva F1 terhadap confidence

3. Precision-Confidence Curve Dan Recall-Confidence Curve

Kurva *precision*-confidence dan *recall*-confidence menunjukkan adanya trade-off khas: semakin tinggi ambang *confidence*, precision mendekati 1,00 namun recall menurun signifikan setelah melewati kisaran 0,8–0,9. Hal ini menegaskan bahwa pemilihan ambang *confidence* perlu disesuaikan dengan kebutuhan aplikasi; misalnya, untuk mendeteksi semua objek kecil mungkin ambang diturunkan agar recall tinggi, dengan konsekuensi precision sedikit menurun. Gambar 7 dan Gambar 8 masing-masing menunjukkan kurva precision terhadap confidence dan recall terhadap confidence.



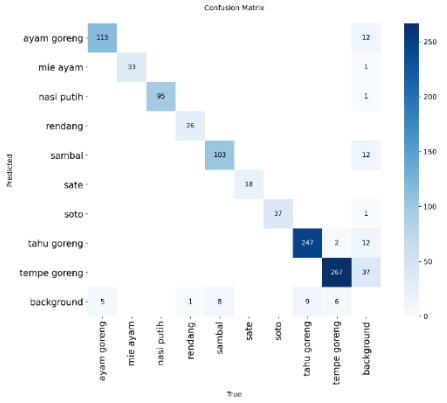
Gambar 7. Precision-Confidence Curve



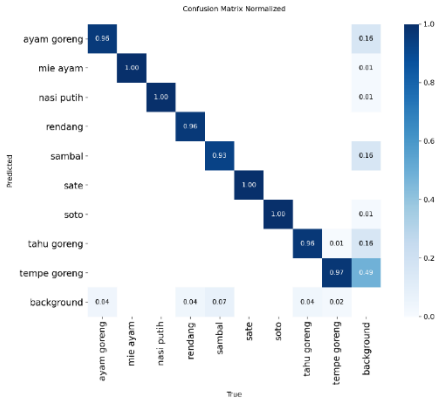
Gambar 8. Recall-Confidence Curve

4. Confusion Matrix

Confusion matrix dalam nilai absolut maupun ternormalisasi memperlihatkan dominasi nilai pada diagonal, menandakan akurasi prediksi yang tinggi di hampir semua kelas. Kesalahan yang tersisa umumnya berupa *false positive* objek *background* yang keliru terdeteksi sebagai kelas makanan tertentu. Paling sering, latar belakang terdeteksi sebagai *tempe goreng*, disusul beberapa kasus sebagai *ayam goreng*, *sambal*, atau *tahu goreng*. Hal ini mengindikasikan adanya kemiripan tekstur/warna latar dengan beberapa makanan gorengan, serta terbatasnya contoh *negative* (latar) dalam data latih. Solusi ke depan adalah menambahkan variasi citra latar belakang dan menyeimbangkan data antar kelas. Gambar 9 menunjukkan *confusion matrix* absolut, sedangkan Gambar 10 menunjukkan versi ternormalisasi yang memperjelas proporsi kesalahan per kelas.



Gambar 9. Confusion Matrix Absolut



Gambar 10. Confusion Matrix Ternormalisasi

4.3. Implementasi Aplikasi

Setelah model dan backend teruji dengan baik, tahap berikutnya adalah integrasi ke aplikasi mobile. Aplikasi Kalorika telah diimplementasikan sesuai rancangan pada sub-bab 3.4, dan diuji coba untuk memastikan fungsionalitas end-to-end. Beberapa cuplikan antarmuka kunci dari aplikasi ditunjukkan pada Gambar 11 hingga Gambar 14.

1. Splash Screen

Gambar 11. Splash Screen

Gambar 11 menunjukkan *splash screen* Kalorika yang menampilkan logo aplikasi (ikon kamera berpadu elemen roti berwarna oranye) beserta teks "Kalorika". Layar ini tampil sesaat saat aplikasi dibuka, berfungsi sebagai layar pembuka sekaligus melakukan inisialisasi awal (memuat aset, font, dll.) sebelum masuk ke halaman utama.

2. Home Screen

Gambar 12. Home Screen

Gambar 12 memperlihatkan *home screen* setelah *splash*. Di sini tertera judul Kalorika, tagline "Ayo hitung kalori dari fotomu!", serta dua tombol aksi utama: Camera dan Pilih dari Gallery. Pengguna dapat menekan Camera untuk mengambil foto makanan menggunakan kamera, atau Pilih dari Gallery untuk mengunggah foto makanan yang sudah tersimpan. Mekanisme perolehan gambar dijelaskan pada sub-bab 3.5; aplikasi akan meminta izin yang diperlukan (kamera/penyimpanan) sebelum melanjutkan.

3. Loading Screen

Gambar 13. Loading Screen

Gambar 13 menunjukkan layar *loading* yang muncul setelah pengguna mengirim foto ke server. Pesan "Menganalisis foto dengan AI…" ditampilkan disertai animasi penunjuk proses. Layar ini menandakan aplikasi sedang menunggu respons dari endpoint inferensi di server. Apabila koneksi lambat atau server memerlukan waktu komputasi, pengguna dibiarkan menunggu di layar ini. Jika terjadi kegagalan (misal jaringan putus atau error server), aplikasi akan menampilkan notifikasi kesalahan dan opsi untuk mencoba kembali.

4. Result Screen

Gambar 14. Result Screen

Gambar 14 menunjukkan *result screen* yang menampilkan hasil deteksi dan estimasi kalori. Di bagian atas, ditampilkan pratinjau foto yang dianalisis. Di bawahnya, terdapat ringkasan Total Kalori yang merupakan akumulasi kalori dari semua objek makanan terdeteksi dalam foto. Selanjutnya, aplikasi menampilkan daftar hasil per item makanan dalam bentuk kartu yang dapat digulir. Tiap kartu memuat nama kelas makanan, nilai confidence model, estimasi berat (gram), dan estimasi energi (kcal) untuk item tersebut. Angka-angka estimasi dibulatkan untuk kemudahan pembacaan. Dengan tampilan ini, pengguna dapat dengan mudah melihat kontribusi kalori per item maupun total kalori dalam satu foto makanan.

Secara umum, pengujian *end-to-end* menunjukkan bahwa aplikasi Kalorika berhasil menjalankan skenario yang diharapkan: pengguna cukup mengambil atau memilih foto makanan, lalu sistem mendeteksi makanan di foto dan menampilkan estimasi kalorinya secara otomatis. Waktu respons total (termasuk upload, inferensi, download hasil) rata-rata sekitar 1–2 detik per foto pada koneksi WiFi dan server Railway, yang tergolong cukup cepat dan layak untuk pengalaman pengguna. Dari sisi antarmuka, pengguna uji menilai aplikasi mudah digunakan dan informatif. Dengan demikian, integrasi model YOLOv8n dan komponen backend ke dalam aplikasi mobile dapat dinyatakan berhasil memenuhi tujuan penelitian.

5. KESIMPULAN

Berdasarkan hasil yang diperoleh, dapat disimpulkan beberapa poin sebagai berikut:

a. Penelitian ini berhasil merancang dan mengimplementasikan sistem Kalorika untuk deteksi makanan dan estimasi kalori berbasis citra sesuai tujuan. Model deteksi objek YOLOv8n telah dilatih pada 3.772 citra (9 kelas makanan Indonesia) dan mencapai performa deteksi tinggi (mAP@0.5 97,5%).
b. Analisis hasil menunjukkan model mampu mempertahankan *precision* dan *recall* yang baik pada sebagian besar kelas, sejalan dengan studi terdahulu yang menyatakan YOLOv8 unggul dalam akurasi deteksi makanan. Kesalahan deteksi yang masih terjadi utamanya disebabkan kemiripan visual antar kelas (misal sama-sama gorengan) dan ketidakseimbangan jumlah data, sehingga perbaikan dataset (penambahan sampel kelas minor dan variasi latar) diharapkan dapat lebih meningkatkan kinerja.
c. Sistem *backend* FastAPI yang di-deploy di Railway mampu menjalankan inferensi YOLOv8n dengan efisien (model dimuat satu kali) dan melakukan perhitungan massa serta kalori berdasarkan parameter nutrisi tiap kelas. Arsitektur *environment variables* yang diterapkan membuat sistem fleksibel terhadap perubahan konfigurasi (misal mengganti model atau mengatur ambang deteksi) tanpa modifikasi kode.
d. Aplikasi mobile Flutter Kalorika telah terintegrasi dengan baik, sehingga pengguna dapat memperoleh estimasi kalori makanan secara otomatis hanya dengan mengambil foto. Pengujian *end-to-end* menunjukkan pengalaman penggunaan yang lancar, dengan waktu respon rata-rata 1–2 detik dan antarmuka yang sederhana serta informatif.
e. Ke depan, sistem ini dapat dikembangkan lebih lanjut dengan menambah jenis makanan yang didukung, meningkatkan kualitas dan keseimbangan data training, serta mengeksplorasi teknologi pendukung (misalnya kamera kedalaman atau segmentasi citra) untuk estimasi porsi yang lebih akurat.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada semua pihak yang telah mendukung dan membantu kelancaran penelitian ini, terutama Universitas Bina Sarana Informatika dan komunitas open-source Roboflow Universe yang menyediakan dataset.

DAFTAR PUSTAKA

[1] Anggie Annisa Permatasari, Fransisca Putri Ardita, Agista Putri Prasetya, Nurul Anggraini, Siti Marpuah, and Erintya Asanti, "Dampak Makanan Cepat Saji Bagi Kesehatan Tubuh Pada Kalangan Remaja," *Jurnal Ventilator: Jurnal riset ilmu kesehatan dan Keperawatan*, vol. 2, no. 2, pp. 110–120, Jun. 2024, doi: 10.59680/ventilator.v2i2.1201.

[2] A. Singh, P. Singh, and A. Kr Tiwari, "A Comprehensive Survey on Machine Learning," *Journal of Manage-ment and Service Science*, vol. 01, no. 003, pp. 1–17, 2021, doi: 10.54060/JMSS/001.01.003.

[3] B. Karyadi, "PEMANFAATAN-KECERDASAN-BUATAN-DALAM-MENDUKUNG-PEMBELAJARAN-MANDIRI," *JURNAL TEKNOLOGI PENDIDIKAN*, vol. 8, pp. 1–7, 2023.

[4] X. Cong, S. Li, F. Chen, C. Liu, and Y. Meng, "Frontiers in Computing and Intelligent Systems A Review of YOLO Object Detection Algorithms based on Deep Learning," *Frontiers in Computing and Intelligent Systems*, vol. 4, pp. 1–4, 2023.

[5] G. C. Utami, C. R. Widiawati, and P. Subarkah, "Detection of Indonesian Food to Estimate Nutritional Information Using YOLOv5," *Teknika*, vol. 12, no. 2, pp. 158–165, Jun. 2023, doi: 10.34148/teknika.v12i2.636.

[6] F. Romadhon *et al.*, "Food Image Detection System and Calorie Content Estimation Using Yolo to Control Calorie Intake in the Body," in *E3S Web of Conferences*, EDP Sciences, Dec. 2023. doi: 10.1051/e3sconf/202346502057.

[7] J.-T. Huang and C.-H. Wang, "Deep Learning-Based Food Identification and Calorie Estimation System," *The Seventh International Conference on Applications and Systems of Visual Paradigms*, pp. 1–4, 2022.

[8] L. Rahma, H. Syaputra, A. H. Mirza, and S. D. Purnamasari, "Objek Deteksi Makanan Khas Palembang Menggunakan Algoritma YOLO (You Only Look Once)," *Jurnal Nasional Ilmu Komputer*, vol. 2, no. 3, pp. 1–20, Aug. 2021.

[9] R. Agarwal, T. Choudhury, N. J. Ahuja, and T. Sarkar, "IndianFoodNet: Detecting Indian Food Items Using Deep Learning," *International Journal of Computational Methods and Experimental Measurements*, vol. 11, no. 4, pp. 221–232, Dec. 2023, doi: 10.18280/ijcmem.110403.

[10] A. M. Albaehaqi, M. I. Andriana, and R. H. Hidayat, "NUTRICHIVE: APLIKASI MOBILE UNTUK DETEKSI BAHAN MAKANAN DAN REKOMENDASI RESEP GUNA MENGURANGI LIMBAH MAKANAN," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 2, no. 1, Jan. 2025, doi: 10.23960/jitet.

[11] N. Aziz, P. Gali, and M. S. Nurcahya, "Analisa dan Perancangan Aplikasi Pembelajaran Bahasa Inggris Dasar Berbasis Android," *Jurnal IKRA-ITH Informatika*, vol. 4, pp. 1–5, Nov. 2020.

[12] P. R. Setiawan, R. A. Ramadhan, D. A. Labellapansa, P. Koresponden, : Panji, and R. Setiawan, "Jurnal Pengabdian Masyarakat dan Penerapan Ilmu Pengetahuan Pelatihan Pemrograman Flutter."

[13] G. S. Chandra and S. Tjandra, "Pemanfaatan Flutter dan Electron Framework pada Aplikasi Inventori dan Pengaturan Pengiriman Barang," *JOURNAL OF INFORMATION SYSTEM, GRAPHICS, HOSPITALITY AND TECHNOLOGY*, pp. 1–6, 2020.

[14] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, "A survey on missing data in machine learning," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00516-9.

[15] N. J. Hayati, D. Singasatia, M. R. Muttaqin, T. Informatika, S. Tinggi, and T. Wastukancana, "OBJECT TRACKING MENGGUNAKAN ALGORITMA YOU ONLY LOOK ONCE (YOLO)v8 UNTUK MENGHITUNG KENDARAAN," *KOMPUTA : Jurnal Ilmiah Komputer dan Informatika*, vol. 12, no. 2, 2023, [Online]. Available: https://universe.roboflow.com/

[16] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences (Switzerland)*, vol. 12, no. 9, May 2022, doi: 10.3390/app12094369.

[17] B. Bednarz and M. Miłosz, "Benchmarking the performance of Python web frameworks Analiza porównawcza wydajności webowych szkieletów programistycznych w języku

Python," *Journal of Computer Sciences Institute*, pp. 1–6, Jun. 2025.

[18] J. Santiago, "TESTING PERFORMA FRAMEWORK EXPRESS.JS, GIN, DAN FASTAPI MENGGUNAKAN API DAN JMETER," *Journal of Information System, Applied, Management, Accounting and Research. Issue Period*, vol. 9, no. 3, pp. 1219–1226, 2025, doi: 10.52362/jisamar.v9i3.1986.

[19] P. Gujjar and N. Kumar, "Google Colaboratory : Tool for Deep Learning and Machine Learning Applications," *Indian Journal of Computer Science*, pp. 1–4, 2021.

[20] R. Gelar Guntara, "Pemanfaatan Google Colab Untuk Aplikasi Pendeteksian Masker Wajah Menggunakan Algoritma Deep Learning YOLOv7," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 5, no. 1, pp. 55–60, Feb. 2023, doi: 10.47233/jteksis.v5i1.750.