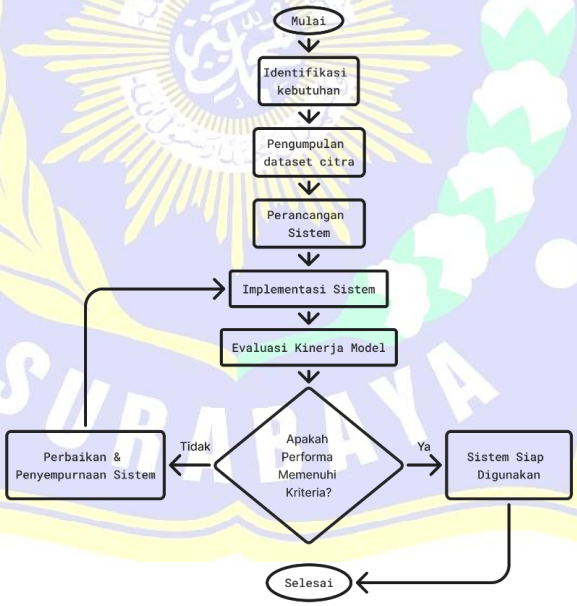
BAB III
METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan pengembangan sistem berbasis Software Development Life Cycle (SDLC) dengan model iteratif. Model ini dipilih karena pengembangan sistem klasifikasi bahan pangan memerlukan proses pelatihan, evaluasi, dan penyempurnaan parameter secara berulang (Ma et al., 2021). Berbeda dengan model linear seperti waterfall yang bersifat sekuensial dan sulit direvisi pada tahap akhir, model iteratif memungkinkan perbaikan bertahap berdasarkan hasil evaluasi setiap siklus pengujian.

Dalam konteks penelitian ini, siklus iteratif diterapkan terutama pada tahap pelatihan dan validasi model klasifikasi. Setiap iterasi melibatkan proses pelatihan model menggunakan data *training*, evaluasi pada data *validation*, analisis kesalahan klasifikasi, serta penyetelan parameter XGBoost hingga diperoleh konfigurasi yang stabil. Proses ini diulang sampai metrik evaluasi yang ditetapkan menunjukkan performa yang konsisten dan tidak mengalami fluktuasi signifikan. Alur pengembangan sistem secara umum ditunjukkan pada Gambar 3.1. Diagram tersebut merepresentasikan tahapan utama pengembangan sistem secara keseluruhan.



Gambar 3. 1 Diagram Model Pengembangan Iteratif

Sistem yang dikembangkan terdiri atas dua komponen utama yang terintegrasi. Komponen pertama adalah modul klasifikasi bahan pangan berbasis citra menggunakan Vision transformer sebagai ekstraktor fitur dan XGBoost sebagai *classifier* multiclass. Komponen kedua adalah modul perhitungan kalori yang menggunakan rumus deterministik berdasarkan nilai kalori per 100 gram dari Tabel Komposisi Pangan Indonesia.

Objek penelitian dibatasi pada sepuluh jenis bahan pangan, yaitu beras putih, telur ayam ras segar, kentang segar, wortel segar, bawang merah segar, bawang putih segar, cabai merah segar, jagung kuning mentah, tomat merah segar, dan terung ungu segar. Alur kerja sistem dimulai dari input citra bahan pangan. Citra tersebut melalui tahap prapemrosesan berupa resize dan normalisasi, kemudian dilakukan augmentasi pada subset data *training* untuk meningkatkan variasi representasi visual. Selanjutnya, citra diproses oleh Vision transformer untuk menghasilkan vektor *embedding* sebagai representasi fitur tingkat tinggi (Dosovitskiy et al., 2021). Vektor *embedding* tersebut digunakan sebagai masukan bagi XGBoost untuk menentukan kelas bahan pangan.

Setelah jenis bahan teridentifikasi, pengguna memasukkan berat bahan dalam satuan gram. Sistem kemudian menghitung estimasi kalori menggunakan rumus proporsional berbasis nilai kalori per 100 gram sesuai TKPI. Evaluasi penelitian difokuskan pada kinerja klasifikasi bahan pangan yang diukur menggunakan metrik *precision*, *recall*, *F1-score*, dan *accuracy*. Perhitungan kalori tidak dievaluasi sebagai model prediktif, melainkan sebagai hasil deterministik dari kelas yang diprediksi dan berat yang dimasukkan.

3.1 Pengumpulan Data

3.1.1 Dataset Citra Bahan Pangan

Dataset citra bahan pangan dalam penelitian ini diperoleh dari dua sumber publik berlisensi terbuka, yaitu Kaggle dan Pixabay API. Penggunaan kedua sumber tersebut dimaksudkan untuk menyediakan jumlah data yang memadai serta meningkatkan keragaman visual pada setiap kelas bahan pangan. Berbeda dengan penelitian sebelumnya yang berfokus pada makanan siap konsumsi atau hidangan multi komponen (J. Louro et al., 2024), penelitian ini membatasi objek pada bahan pangan. Pembatasan tersebut dilakukan untuk menjaga konsistensi pelabelan, mengurangi ambiguitas komposisi objek, serta memastikan keterkaitan langsung antara hasil klasifikasi dan perhitungan nilai kalori berdasarkan referensi Tabel Komposisi Pangan Indonesia (TKPI). Penelitian ini menggunakan sepuluh kelas bahan pangan yang ditampilkan pada Tabel 3.1.

Tabel 3. 1 Daftar objek yang digunakan pada penelitian

No	Nama Kelas
1	Beras putih
2	Telur ayam ras segar
3	Kentang segar
4	Wortel segar
5	Bawang merah segar
6	Bawang putih segar
7	Cabai merah segar
8	Jagung kuning mentah
9	Tomat merah segar
10	Terung ungu segar

Data yang diperoleh selanjutnya melalui proses validasi dan pembersihan, yang mencakup penghapusan citra duplikat menggunakan hash MD5, deteksi file citra yang rusak (corrupt), serta penyaringan gambar yang tidak relevan melalui verifikasi manual. Setelah proses tersebut, Setiap citra diberi label sesuai kelas bahan pangan yang direpresentasikan sebagai ground truth dalam proses pelatihan dan evaluasi model klasifikasi. Target utama sistem adalah mengidentifikasi jenis bahan pangan berdasarkan citra, sedangkan informasi kalori tidak digunakan sebagai variabel target pada tahap pelatihan model. Nilai kalori dirancang untuk dihitung setelah proses klasifikasi menggunakan data referensi Tabel Komposisi Pangan Indonesia (TKPI) Tahun 2020 berdasarkan nilai energi per 100 gram bahan pangan.

3.1.2 Spesifikasi Dataset

Dataset penelitian berupa citra digital bahan pangan dengan resolusi awal yang bervariasi dan diseragamkan melalui tahap prapemrosesan sebelum digunakan dalam pelatihan model. Dataset terdiri atas 5.000 citra yang terbagi ke dalam 10 kelas bahan pangan, masing-masing berjumlah 500 citra. Komposisi yang seimbang ini digunakan untuk mengurangi potensi bias distribusi kelas selama proses pelatihan dan evaluasi. Citra memiliki variasi latar belakang, sudut pengambilan, jarak kamera, serta kondisi pencahayaan alami maupun buatan.

Variasi tersebut dipertahankan secara terkontrol agar model tidak bergantung pada pola latar belakang tertentu dan tetap relevan pada kondisi penggunaan nyata (Fissler et al., 2023).

Sebelum pelatihan, citra diseleksi berdasarkan beberapa kriteria, yaitu objek bahan terlihat jelas sebagai fokus utama, tidak mengalami blur atau distorsi berat, serta tidak mengandung watermark maupun teks yang mengganggu. Citra duplikat atau yang sangat mirip secara visual juga dihapus untuk mengurangi redundansi data.

Pada tahap prapemrosesan, seluruh citra diubah ukurannya menjadi 224 × 224 piksel menggunakan interpolasi bilinear sesuai spesifikasi input Vision Transformer pra latih berbasis ImageNet. Selanjutnya, nilai piksel dinormalisasi ke rentang 0 - 1 dan disesuaikan dengan nilai mean serta standar deviasi ImageNet agar konsisten dengan parameter pra latih (Divyanth et al., 2022). Dengan spesifikasi tersebut, dataset memiliki struktur yang seragam, distribusi kelas seimbang, dan variasi visual yang mendukung proses klasifikasi bahan pangan.

3.1.3 Pembagian Data *Training Validation* Test

Dataset yang telah melalui tahap seleksi dan prapemrosesan dibagi menjadi tiga subset, yaitu *training*, *validation*, dan test. Pembagian ini bertujuan memisahkan proses pelatihan, penyetelan parameter, dan evaluasi model. Data *training* digunakan untuk melatih model dalam mengenali pola visual setiap kelas bahan pangan. Data *validation* digunakan untuk penyetelan parameter dan pemantauan kinerja selama pelatihan tanpa melibatkan data test. Data test digunakan sebagai evaluasi akhir untuk mengukur kinerja model setelah proses pelatihan dan penyempurnaan selesai.

Pembagian dataset dilakukan dengan pendekatan stratified split agar proporsi distribusi kelas tetap relatif seimbang pada setiap subset (S. B. Lee et al., 2019). Dengan cara ini, setiap kelas bahan pangan tetap terwakili secara proporsional pada data *training*, *validation*, dan test. Proporsi pembagian ditetapkan sebesar 70 persen untuk *training*, 15 persen untuk *validation*, dan 15 persen untuk test guna menjaga keseimbangan antara kebutuhan pelatihan dan evaluasi independen (Dalloo & Humaidi, 2024). Berdasarkan total 5.000 citra yang digunakan dalam penelitian, hasil pembagian dataset ditunjukkan pada Tabel 3.2.

Tabel 3. 2 Pembagian Dataset Training, Validation, dan Test

Subset	Persentase	Jumlah Citra
Training	70%	3.500
Validation	15%	750
Test	15%	750
Total	100%	5.000

Teknik augmentasi citra hanya diterapkan pada data training setelah pembagian dataset dilakukan. Data validation dan test tidak mengalami transformasi tambahan selain resize dan normalisasi dasar. Pendekatan ini dilakukan untuk mengurangi risiko kebocoran informasi antara tahap pelatihan dan pengujian, sehingga evaluasi mencerminkan kinerja klasifikasi pada data yang tidak digunakan selama pelatihan. Melalui proses augmentasi, jumlah data pada subset training ditingkatkan dari 3.500 citra menjadi 21.000 citra, sedangkan jumlah data validation dan test tetap masing-masing 750 citra. Dengan demikian, total data yang digunakan pada tahap eksperimen berjumlah 22.500 citra yang terdiri atas 21.000 data training, 750 data validation, dan 750 data test.

Peningkatan jumlah data training menjadi 21.000 citra diperoleh melalui proses augmentasi yang diterapkan setelah pembagian dataset dilakukan. Dari 3.500 citra asli pada subset training, dihasilkan tambahan 17.500 citra hasil transformasi sehingga jumlah data training meningkat enam kali lipat. Pendekatan ini dilakukan untuk memperkaya variasi visual data pelatihan tanpa menambah data pada subset validation dan test. Dengan demikian, proses evaluasi tetap dilakukan menggunakan data yang tidak terlibat dalam pembentukan sampel hasil augmentasi sehingga risiko kebocoran informasi dapat diminimalkan

3.2 Pengolahan Data

3.2.1 Resize dan Normalisasi

Tahap pengolahan data diawali dengan penyesuaian ukuran citra agar sesuai dengan spesifikasi masukan model Vision transformer yang digunakan sebagai ekstraktor fitur. Misalkan citra asli dinyatakan sebagai matriks tiga dimensi:

$$I \in \mathbb{R}^{H \times W \times C} \tag{3.1}$$

dengan Hmenyatakan tinggi citra, Wlebar citra, dan $C = 3$jumlah kanal warna RGB.

Resize

Seluruh citra diubah ukurannya menjadi resolusi tetap 224 × 224 piksel menggunakan interpolasi bilinear. Proses resize dapat direpresentasikan sebagai fungsi transformasi:

$$I' = \mathcal{R}(I) \tag{3.2}$$

dengan:

$$I' \in \mathbb{R}^{224 \times 224 \times 3} \tag{3.3}$$

Interpolasi bilinear dipilih karena memberikan keseimbangan antara efisiensi komputasi dan kualitas hasil transformasi dibandingkan metode nearest neighbor yang lebih kasar. Resolusi 224 × 224 digunakan karena sesuai dengan konfigurasi input standar Vision transformer pra latih berbasis ImageNet (Dosovitskiy, 2020).

Normalisasi

Setelah proses resize, dilakukan normalisasi nilai piksel untuk menyelaraskan distribusi intensitas citra dengan distribusi data pada saat pra pelatihan model. Nilai piksel awal dalam rentang 0 hingga 255 diubah ke dalam skala 0 hingga 1 melalui transformasi:

$$I'' = \frac{I'}{255} \tag{3.4}$$

Selanjutnya dilakukan normalisasi kanal warna menggunakan nilai mean dan standar deviasi ImageNet, yaitu:

$$\mu = (0{,}485,\ 0{,}456,\ 0{,}406) \tag{3.5}$$

$$\sigma = (0{,}229,\ 0{,}224,\ 0{,}225) \tag{3.6}$$

Transformasi normalisasi dilakukan per kanal sebagai berikut:

$$\hat{I}_c = \frac{I_c'' - \mu_c}{\sigma_c} \tag{3.7}$$

dengan $c \in \{R, G, B\}$.

Hasil akhir dari tahap ini adalah tensor terstandarisasi berdimensi 224 × 224 × 3 sebagaimana dinyatakan pada Persamaan (3.3) yang kemudian dikonversi ke format tensor sesuai kebutuhan kerangka kerja *deep learning* sebelum dimasukkan ke dalam Vision transformer. Tahap resize dan normalisasi diterapkan secara konsisten pada seluruh subset data, yaitu *training*, *validation*, dan test. Konsistensi ini penting untuk memastikan bahwa distribusi fitur yang diproses oleh model tetap seragam pada seluruh tahap eksperimen. Dengan prosedur ini, setiap citra memiliki dimensi dan distribusi nilai piksel yang selaras dengan spesifikasi model pra latih, sehingga proses ekstraksi fitur dapat berjalan secara optimal dan stabil.

3.2.2 Teknik Augmentasi

Teknik augmentasi yang digunakan meliputi rotasi, horizontal flip, penyesuaian kecerahan, zoom in, zoom out, dan Gaussian blur tanpa mengubah identitas semantik bahan pangan sehingga label kelas tetap konsisten. Parameter augmentasi ditetapkan menggunakan library torchvision, dengan rotasi maksimum ±15°, horizontal flip sebesar 50%, penyesuaian kecerahan pada rentang 0,8–1,2, faktor skala zoom pada rentang 0,7–1,0, serta Gaussian blur menggunakan kernel berukuran 3 × 3.

Augmentasi hanya diterapkan pada data *training* setelah pembagian dataset, sedangkan data *validation* dan *testing* hanya melalui proses resize dan normalisasi untuk menjaga konsistensi evaluasi serta menghindari kebocoran informasi. Secara matematis, augmentasi direpresentasikan sebagai fungsi transformasi yang mengubah citra masukan $I(x, y)$, dengan x dan y menyatakan koordinat piksel, maka transformasi menghasilkan citra baru $I'(x', y')$.

Rotasi

Transformasi rotasi dengan sudut θ dapat dinyatakan sebagai:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.8)$$

dengan θ berada pada rentang $-15^\circ \leq \theta \leq 15^\circ$untuk menghindari distorsi berlebihan pada objek.

Horizontal Flip

Transformasi horizontal flip dapat dinyatakan sebagai:

$$x' = W - x \quad (3.9)$$
$$y' = y$$

dengan Wmenyatakan lebar citra.

Penyesuaian Kecerahan

Penyesuaian kecerahan dapat dimodelkan sebagai transformasi linear terhadap intensitas piksel:

$$I'(x, y) = \alpha \cdot I(x, y) \quad (3.10)$$

dengan α merupakan faktor penyesuaian kecerahan pada rentang 0,8 hingga 1,2.

Zoom In

Transformasi zoom in dilakukan dengan memperbesar area tertentu dari citra dan kemudian mengubah ukurannya kembali ke dimensi masukan model. Secara matematis dapat direpresentasikan sebagai:

$$I'(x,y) = I\left(\frac{x}{s}, \frac{y}{s}\right) \quad (3.11)$$

dengan:

$$0.8 \leq s \leq 1.0$$

di mana s merupakan faktor skala pembesaran. Nilai s yang lebih kecil menyebabkan area citra yang dipilih menjadi lebih sempit sehingga objek tampak lebih besar setelah proses resize dilakukan kembali ke ukuran 224 × 224 piksel.

Zoom Out

Transformasi zoom out dilakukan dengan memperkecil ukuran objek pada citra sehingga objek tampak lebih jauh dari kamera. Transformasi ini dapat dinyatakan sebagai:

$$I'(x,y) = I(sx, sy) \quad (3.12)$$

dengan:

$$0.7 \leq s \leq 0.9$$

di mana s merupakan faktor skala pengecilan. Nilai s yang lebih kecil menghasilkan objek yang tampak lebih kecil pada citra hasil transformasi.

Gaussian Blur

Gaussian Blur dilakukan dengan mengonvolusikan citra terhadap kernel Gaussian untuk menghasilkan efek pengaburan. Transformasi ini dapat dinyatakan sebagai:

$$I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} G(i,j)\, I(x-i, y-j) \quad (3.13)$$

dengan fungsi Gaussian:

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \quad (3.14)$$

di mana:

- $G(i,j)$adalah bobot kernel Gaussian
- σadalah standar deviasi Gaussian
- $k$adalah ukuran setengah kernel
- $I(x,y)$adalah intensitas piksel sebelum transformasi
- $I'(x,y)$adalah intensitas piksel setelah transformasi

Pada penelitian ini digunakan kernel berukuran 3 × 3 dengan rentang $\sigma = 0.1$ hingga 1.0.

Seluruh transformasi tersebut tidak mengubah label kelas bahan pangan karena tidak mengubah identitas objek utama pada citra. Dengan pembatasan rentang parameter transformasi, augmentasi bertujuan memperkaya variasi visual tanpa menyebabkan distorsi yang dapat memengaruhi proses klasifikasi.

3.2.3 Analisis Karakteristik Visual Dataset

Sebelum pelatihan model, dilakukan studi pendahuluan untuk menggambarkan karakteristik visual citra menggunakan pustaka OpenCV. Studi ini bersifat deskriptif dengan menganalisis parameter warna, tekstur, dan bentuk objek tanpa melibatkan model pembelajaran mesin. Tujuannya adalah memberikan gambaran karakteristik visual dataset yang digunakan dalam penelitian. Analisis dilakukan pada 90 sampel citra yang terdiri atas 30 citra tomat merah, 30 citra wortel, dan 30 citra telur ayam ras. Ketiga kelas dipilih karena mewakili karakteristik visual yang berbeda. Analisis meliputi warna dan tekstur pada sampel sebagai contoh analisis morfologi objek. Implementasi fungsi analisis ditunjukkan pada Gambar 3.2 dan Gambar 3.3.

```python
def analisis_warna(img_bgr):
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
    img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)

    R = img_rgb[:, :, 0].astype(float)
    G = img_rgb[:, :, 1].astype(float)
    B = img_rgb[:, :, 2].astype(float)
    H = img_hsv[:, :, 0].astype(float)
    S = img_hsv[:, :, 1].astype(float)
    V = img_hsv[:, :, 2].astype(float)

    return {
        "R_min": int(R.min()), "R_max": int(R.max()), "R_mean": round(R.mean(), 2),
        "G_min": int(G.min()), "G_max": int(G.max()), "G_mean": round(G.mean(), 2),
        "B_min": int(B.min()), "B_max": int(B.max()), "B_mean": round(B.mean(), 2),
        "Hue_mean": round(H.mean() * 2, 2),
        "Sat_mean": round(S.mean(), 2),
        "Val_mean": round(V.mean(), 2)
    }

def analisis_tekstur(img_bgr):
    gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
    contrast = gray.std()
    laplacian = cv2.Laplacian(gray, cv2.CV_64F)
    sharpness = laplacian.var()
    return {
        "contrast": round(contrast, 2),
        "sharpness": round(sharpness, 2)
    }
```

Gambar 3. 2 Implementasi Fungsi Citra

Fungsi analisis warna digunakan untuk memperoleh parameter warna pada ruang warna RGB dan HSV, sedangkan fungsi analisis tekstur digunakan untuk menghitung parameter kontras dan ketajaman citra. Seluruh parameter tersebut digunakan sebagai acuan dalam studi pendahuluan untuk mendeskripsikan karakteristik visual dataset. Hasil analisis disajikan pada Bab IV.

```
if is_tomat and len(detections) > 0 and best_box is not None:
    x1, y1, x2, y2 = [int(v) for v in best_box]
    crop_bgr = img_bgr[y1:y2, x1:x2]
    if crop_bgr.size > 0:
        warna = analisis_warna(crop_bgr)

        gray = cv2.cvtColor(crop_bgr, cv2.COLOR_BGR2GRAY)
        _, thresh = cv2.threshold(gray, 40, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        circularity = 0.0
        aspect_ratio = 0.0
        if contours:
            largest = max(contours, key=cv2.contourArea)
            area = cv2.contourArea(largest)
            perimeter = cv2.arcLength(largest, True)
            circularity = (4 * np.pi * area) / (perimeter ** 2) if perimeter > 0 else 0.0
            x, y, w, h_box = cv2.boundingRect(largest)
            aspect_ratio = w / h_box if h_box > 0 else 0.0
```

Gambar 3. 3 Implementasi Fungsi Citra

Proses dilakukan dengan menganalisis area objek hasil deteksi untuk memperoleh karakteristik bentuk. Parameter yang dihitung meliputi circularity dan aspect ratio. Circularity dihitung menggunakan Persamaan, dengan nilai yang mendekati satu menunjukkan bentuk yang semakin menyerupai lingkaran. Sementara itu, aspect ratio diperoleh dari perbandingan lebar dan tinggi objek, di mana nilai yang mendekati satu menunjukkan bentuk yang lebih proporsional. Kedua parameter tersebut digunakan sebagai acuan dalam studi pendahuluan untuk menggambarkan karakteristik bentuk tomat pada dataset. Hasil analisis disajikan pada Bab IV.

3.3 Rancangan Sistem yang diusulkan

3.3.1 Arsitektur Sistem

Arsitektur sistem dirancang sebagai pipeline terintegrasi yang menghubungkan proses ekstraksi fitur visual, klasifikasi bahan pangan, dan perhitungan kalori berbasis referensi Tabel Komposisi Pangan Indonesia. Struktur sistem secara keseluruhan ditunjukkan pada Gambar 3.4.

Gambar 3. 4 Arsitektur Sistem

Lapisan Masukan

Proses dimulai dari masukan berupa citra bahan pangan yang dapat memuat satu maupun beberapa objek dalam satu gambar. Citra tersebut kemudian diteruskan ke tahap deteksi objek untuk mengidentifikasi lokasi setiap bahan pangan yang terdapat pada gambar.

Deteksi Objek

Citra masukan diproses oleh YOLOv8n untuk mendeteksi lokasi setiap bahan pangan beserta confidence score-nya. Deteksi di bawah threshold tertentu dibuang guna menekan false positive, lalu area yang lolos dipotong (cropping) menjadi citra objek tunggal untuk tahap klasifikasi selanjutnya. Dengan cara ini, beberapa bahan pangan dalam satu foto bisa dikenali sekaligus (Soylu et al., 2024). Pemilihan YOLOv8n sebagai modul deteksi objek didasarkan pada tiga pertimbangan. Pertama, arsitektur single - stage pada YOLO memungkinkan proses inferensi berlangsung cepat sehingga sesuai untuk aplikasi berbasis web yang membutuhkan respons secara real-time. Kedua, varian nano hanya memiliki sekitar 3,2 juta parameter sehingga lebih ringan secara komputasi dibandingkan varian YOLOv8 lainnya dan tetap mampu dijalankan pada perangkat tanpa GPU khusus saat proses inferensi. Ketiga, penelitian ini hanya melibatkan sepuluh kelas bahan pangan dengan kompleksitas objek yang relatif rendah sehingga kapasitas model YOLOv8n dinilai memadai untuk menghasilkan deteksi yang akurat sekaligus efisien.

Tahap Prapemrosesan

Setiap objek hasil cropping kemudian melalui tahap prapemrosesan sebelum dilakukan ekstraksi fitur. Tahap ini meliputi:

- Resize menjadi 224 × 224 piksel
- Normalisasi berdasarkan statistik ImageNet

Tahap prapemrosesan bertujuan menyeragamkan dimensi dan distribusi nilai piksel sehingga sesuai dengan spesifikasi masukan Vision transformer.

Ekstraksi Fitur Menggunakan Vision Transformer

Citra yang telah dipraproses dimasukkan ke dalam model Vision transformer pra latih. Proses internal model meliputi:

- *Patch embedding*
- Mekanisme *self attention*
- Transformasi melalui beberapa encoder layer

Keluaran tahap ini berupa vektor *embedding* berdimensi tetap:

$$z \in \mathbb{R}^d \quad (3.15)$$

dengan $d = 768$untuk konfigurasi ViT Base. Pada penelitian ini, bobot Vision transformer dibekukan sehingga tidak diperbarui selama pelatihan. Model hanya berfungsi sebagai ekstraktor fitur untuk menghasilkan representasi visual tingkat tinggi.

Klasifikasi Menggunakan XGBoost

Vektor *embedding* hasil Vision transformer menjadi masukan bagi model XGBoost *classifier*. XGBoost membangun ensemble pohon keputusan secara bertahap melalui mekanisme gradient boosting. Setiap pohon memperbaiki kesalahan prediksi dari pohon sebelumnya hingga terbentuk model akhir.

Keluaran tahap ini berupa:

- Label kelas bahan pangan
- Probabilitas prediksi untuk setiap kelas

Kelas dengan probabilitas tertinggi dipilih sebagai hasil klasifikasi akhir. Evaluasi kinerja sistem difokuskan pada tahap ini menggunakan metrik *precision*, *recall*, *F1-score*, dan *accuracy*.

Perhitungan Kalori Berbasis TKPI

Setelah jenis bahan teridentifikasi, sistem mengakses basis data internal yang berisi nilai kalori per 100 gram berdasarkan TKPI (Kementerian Kesehatan RI, 2020). Pengguna kemudian memasukkan berat bahan dalam satuan gram.

Perhitungan estimasi kalori dilakukan menggunakan rumus:

$$Kalori = \frac{Berat}{100} \times Kalori_{TKPI} \quad (3.16)$$

Sebagai contoh, apabila bahan yang teridentifikasi adalah beras putih dengan nilai 357 kkal per 100 gram dan pengguna memasukkan berat 200 gram, maka estimasi kalori dihitung sebagai:

$$Kalori = \frac{200}{100} \times 357 = 714 \text{ kkal}$$

Perhitungan ini bersifat deterministik dan tidak melibatkan model. Variabel berat atau gramasi makanan pada penelitian ini diperoleh melalui input manual oleh pengguna (user input) pada antarmuka aplikasi setelah sistem berhasil melakukan lokalisasi dan klasifikasi jenis bahan pangan.

Keterkaitan Antara YOLOv8n, Vision Transformer, dan XGBoost

Ketiga model bekerja secara sekuensial dalam satu pipeline. YOLOv8n bertugas di tahap awal sebagai modul lokalisasi, mendeteksi posisi tiap bahan pangan lalu meng-crop area tersebut menjadi citra individual. Tanpa tahap ini, model klasifikasi hanya bisa memproses gambar yang berisi satu objek saja. Potongan citra dari YOLOv8n kemudian masuk ke ViT-Base/16, yang menghasilkan vektor embedding berdimensi 768 untuk tiap objek. Embedding ini bersifat deterministik karena bobot ViT dibekukan selama pelatihan. Selanjutnya, XGBoost menerima vektor tersebut dan mengklasifikasikan setiap objek ke salah satu dari sepuluh kelas bahan pangan. Hasil klasifikasi digunakan untuk mencocokkan nilai kalori dari TKPI.

Integrasi dan Output Sistem

Sistem menghasilkan keluaran berupa:

- Jenis bahan pangan
- Estimasi kalori berdasarkan berat yang dimasukkan

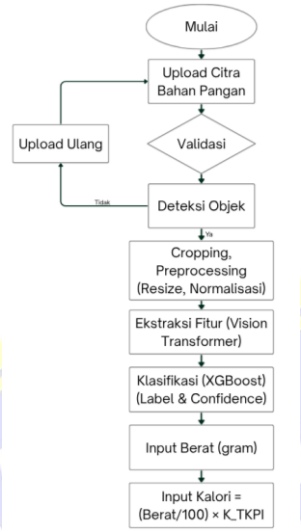
Dengan arsitektur ini, proses klasifikasi dan perhitungan kalori dilakukan secara terpisah namun terhubung dalam satu pipeline terintegrasi. Pemisahan ini meningkatkan transparansi sistem karena kesalahan prediksi dapat dianalisis secara spesifik pada tahap klasifikasi tanpa memengaruhi proses perhitungan kalori yang berbasis standar nasional.

3.3.2 Flowchart Sistem

Flowchart sistem menggambarkan alur kerja estimasi kalori bahan pangan berbasis citra mulai dari unggah citra hingga menghasilkan estimasi kalori berdasarkan data TKPI. Proses diawali dengan pengguna mengunggah citra bahan pangan, kemudian sistem melakukan validasi terhadap citra yang diberikan. Apabila citra tidak memenuhi kriteria validasi, pengguna diminta untuk mengunggah ulang citra. Setelah validasi berhasil, sistem melakukan deteksi objek untuk memperoleh area bahan pangan yang akan dianalisis. Objek yang terdeteksi kemudian dipotong (cropping) dan melalui tahap prapemrosesan berupa resize dan normalisasi.

Tahap utama penelitian dilakukan pada proses ekstraksi fitur menggunakan Vision Transformer (ViT). Model ini digunakan untuk memperoleh representasi fitur citra yang mampu menggambarkan karakteristik visual bahan pangan secara lebih komprehensif. Fitur yang dihasilkan selanjutnya menjadi masukan bagi algoritma XGBoost untuk melakukan klasifikasi jenis bahan pangan dan menghasilkan nilai confidence prediksi. Setelah jenis bahan pangan berhasil diidentifikasi, pengguna memasukkan berat bahan pangan dalam satuan gram.

Sistem kemudian mengambil nilai kalori referensi dari TKPI dan menghitung estimasi kalori menggunakan Persamaan (3.16). Hasil akhir yang ditampilkan meliputi jenis bahan pangan, nilai confidence, berat bahan pangan, dan estimasi kalori.



Gambar 3. 5 Flowchart Sistem Estimasi Kalori Bahan Pangan

3.3.3 Ekstraksi Fitur Vision Transformer

Pada tahap ini, citra bahan pangan yang telah melalui proses prapemrosesan dimasukkan ke dalam model Vision transformer pra latih yang digunakan sebagai ekstraktor fitur visual. Model yang digunakan adalah ViT Base dengan konfigurasi input 224 × 224 piksel dan ukuran *patch* 16 × 16 (Han et al., 2023).

Representasi Citra dan *Patch Embedding*

Misalkan citra hasil prapemrosesan dinyatakan sebagai I ∈ R^{H×W×C} sebagaimana didefinisikan pada Persamaan (3.1), dengan H = 224, W = 224, dan C = 3.

Citra dibagi menjadi *patch* berukuran $P \times P$dengan $P = 16$. Jumlah *patch* yang dihasilkan adalah:

$$N = \frac{H \times W}{P^2} \tag{3.17}$$

Sehingga:

$$N = \frac{224 \times 224}{16^2} = 196$$

Setiap *patch* kemudian diratakan (flatten) menjadi vektor berdimensi:

$$P^2 \times C = 16^2 \times 3 = 768 \tag{3.18}$$

Setiap vektor *patch* diproyeksikan ke ruang *embedding* melalui transformasi linear:

$$x_p = E \cdot p \quad (3.19)$$

dengan Eadalah matriks proyeksi linear.

Penambahan Class Token

Sebuah class token $\mathbf{x}_{cls}$ditambahkan ke urutan token *patch*, sehingga urutan masukan menjadi:

$$X_0 = [\mathbf{x}_{cls}; x_1; x_2; \dots ; x_N] \quad (3.20)$$

Token ini berfungsi sebagai representasi global citra setelah diproses oleh seluruh encoder layer.

Mekanisme *Self attention*

Pada setiap blok Transformer encoder, token diproses menggunakan mekanisme multi head *self attention*. Secara umum, *self attention* dihitung sebagai:

$$\text{Attention}(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.21)$$

dengan:

- $Q = XW_Q$
- $K = XW_K$
- $V = XW_V$

di mana W_Q, W_K, dan $W_V$adalah matriks bobot proyeksi, dan $d_k$adalah dimensi key.

Keluaran dari attention kemudian diproses melalui feed forward network dan residual connection pada setiap layer encoder. Untuk konfigurasi ViT Base, terdapat 12 encoder layer.

Representasi *Embedding* Akhir

Setelah seluruh encoder layer selesai diproses, representasi akhir diambil dari class token:

$$\mathbf{z}_i = f_{\text{ViT}}(I_i) \in \mathbb{R}^d \quad (3.22)$$

dengan $d = 768$.

Karena bobot Vision transformer dibekukan selama pelatihan, parameter model tidak diperbarui. Model hanya berfungsi menghasilkan *embedding* visual tingkat tinggi.

Seluruh *embedding* dari dataset kemudian disusun menjadi matriks fitur:

$$Z \in \mathbb{R}^{n \times 768} \quad (3.23)$$

dengan $n$adalah jumlah citra dalam dataset. Matriks ini menjadi masukan bagi model XGBoost pada tahap klasifikasi.

Pendekatan ini memisahkan secara eksplisit proses ekstraksi fitur dan proses klasifikasi, sehingga pelatihan hanya dilakukan pada model XGBoost tanpa memperbarui parameter Vision transformer. Model Vision transformer yang digunakan adalah ViT Base pra latih ImageNet. Konfigurasi teknis model dirangkum pada Tabel 3.3 berikut.

Tabel 3. 3 Spesifikasi Vision Transformer

Parameter	**Nilai**	**Keterangan**
Model	ViT Base	Arsitektur Vision transformer standar
Ukuran Input	224 × 224 piksel	Resolusi citra setelah resize
Jumlah Kanal	3 (RGB)	Representasi warna citra
Patch Size	16 × 16 piksel	Ukuran *patch* pembentuk token
Jumlah *Patch*	196	Hasil pembagian 224×224 dengan *patch* 16×16
Dimensi *Embedding*	768	Dimensi vektor representasi tiap token
Jumlah Encoder Layer	12	Blok Transformer encoder
Jumlah Attention Head	12	Multi head *self attention*
Dataset Pra Latih	ImageNet-21k	Digunakan untuk *transfer learning*

Dalam penelitian ini, model digunakan untuk mengekstraksi fitur visual dari citra, sedangkan proses penyesuaian parameter dilakukan pada XGBoost sebagai *classifier*.

3.3.4 Klasifikasi XGBoost

Tahap klasifikasi dilakukan menggunakan algoritma XGBoost yang berfungsi untuk memetakan vektor *embedding* hasil ekstraksi Vision transformer ke dalam salah satu dari sepuluh kelas bahan pangan. XGBoost merupakan metode gradient boosting berbasis decision tree yang membangun model secara aditif melalui kombinasi beberapa pohon keputusan untuk meminimalkan fungsi objektif (Sagi & Rokach, 2021).

Formulasi Masukan

Setiap citra ke-i direpresentasikan oleh vektor embedding hasil ekstraksi Vision Transformer sebagaimana dirumuskan pada Persamaan (3.22). Seluruh embedding disusun menjadi matriks fitur $Z \in R^{n \times 768}$ sebagaimana didefinisikan pada Persamaan (3.23), dengan n adalah jumlah sampel pada subset training.

Fungsi Objektif Multiclass

Dalam skenario klasifikasi multiclass dengan $C = 10$ kelas, XGBoost meminimalkan fungsi objektif:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.24}$$

dengan:

- $l(y_i, \hat{y}_i)$adalah fungsi loss *softmax* cross entropy,
- $f_k$adalah pohon keputusan ke-k,
- Kadalah jumlah pohon dalam ensemble,
- $\Omega(f_k)$adalah fungsi regularisasi.

Fungsi regularisasi dinyatakan sebagai:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.25}$$

di mana:

- Tadalah jumlah daun pada pohon,
- $w_j$adalah bobot daun ke-j,
- γdan λadalah parameter regularisasi.

Probabilitas prediksi untuk kelas ke-c dihitung menggunakan fungsi *softmax*:

$$P(y = c \mid \mathbf{z}_i) = \frac{e^{s_c}}{\sum_{j=1}^{C} e^{s_j}} \tag{3.26}$$

dengan $s_c$adalah skor logit yang dihasilkan model untuk kelas ke-c.

Kelas akhir ditentukan melalui:

$$\hat{y}_i = \arg\max_c P(y = c \mid \mathbf{z}_i) \tag{3.27}$$

Penyetelan Hiperparameter

Pada penelitian ini, penyetelan hiperparameter dilakukan menggunakan metode *Grid Search* pada data *validation*. Parameter utama yang disesuaikan meliputi:

- Jumlah pohon (n_estimators)
- Kedalaman maksimum pohon (max_depth)
- *Learning rate*
- Subsample
- Colsample_bytree

3.3.5 Perhitungan Estimasi Kalori

Setelah tahap klasifikasi selesai dan jenis bahan pangan berhasil diidentifikasi, sistem melanjutkan ke tahap perhitungan estimasi kalori. Berbeda dengan pendekatan regresi yang memprediksi nilai kalori langsung dari citra, penelitian ini menggunakan perhitungan deterministik berdasarkan referensi resmi Tabel Komposisi Pangan Indonesia (TKPI) yang diterbitkan oleh Kementerian Kesehatan Republik Indonesia. Nilai energi dalam TKPI dinyatakan dalam satuan kilokalori per 100 gram bahan pangan (Kementerian Kesehatan RI, 2020). Oleh karena itu, estimasi kalori untuk berat bahan tertentu dihitung secara proporsional berdasarkan berat aktual yang dimasukkan pengguna.

Formulasi Matematis

Misalkan:

- B= berat bahan dalam gram
- K_{TKPI}= nilai kalori per 100 gram berdasarkan TKPI
- E= estimasi kalori dalam kilokalori

Estimasi kalori pada penelitian ini dihitung secara proporsional berdasarkan berat bahan yang dimasukkan pengguna. Perhitungan dilakukan menggunakan formulasi yang telah dijelaskan pada Persamaan (3.16). Formulasi tersebut bersifat linear dan tidak melibatkan proses pembelajaran parameter tambahan karena seluruh nilai kalori diperoleh dari referensi resmi Tabel Komposisi Pangan Indonesia (TKPI).

Contoh Perhitungan

Sebagai ilustrasi, apabila bahan yang teridentifikasi adalah beras putih dengan nilai kalori 357 kkal per 100 gram dan pengguna memasukkan berat 200 gram, maka estimasi kalori dihitung sebagai:

$$E = \frac{200}{100} \times 357 = 714 \text{ kkal}$$

Hasil tersebut menunjukkan estimasi kalori total untuk berat bahan yang dimasukkan.

Integrasi dengan Sistem

Setelah model XGBoost menghasilkan label kelas, sistem melakukan pencocokan dengan basis data internal yang memuat nilai kalori per 100 gram berdasarkan Tabel Komposisi Pangan Indonesia. Nilai tersebut kemudian digunakan untuk menghitung estimasi kalori sesuai dengan berat bahan yang dimasukkan pengguna. Proses perhitungan dilakukan secara

deterministik menggunakan data referensi resmi TKPI. Nilai kalori masing-masing bahan pangan yang digunakan dalam penelitian ini disajikan pada Tabel 3.4.

Tabel 3. 4 Nilai Kalori Bahan Pangan Berdasarkan TKPI

Jenis Bahan	**Kalori TKPI (kkal/100g)**
Beras putih	357
Telur ayam ras segar	154
Kentang Segar	62
Wortel Segar	36
Bawang Merah Segar	46
Bawang Putih Segar	112
Cabai Merah Segar	36
Jagung Kuning Mentah	147
Tomat Merah Segar	24
Terung Ungu Segar	25

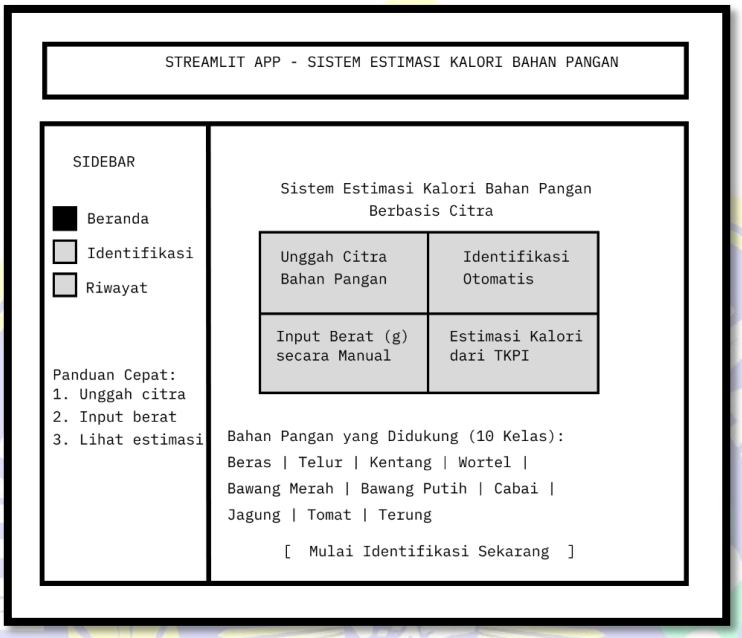
3.3.6 Perancangan Antarmuka Sistem

Antarmuka sistem dirancang sebagai media interaksi antara pengguna dan modul klasifikasi bahan pangan serta modul perhitungan estimasi kalori berbasis Tabel Komposisi Pangan Indonesia. Sistem diimplementasikan dalam bentuk aplikasi prototipe berbasis web menggunakan framework Streamlit. Pemilihan Streamlit didasarkan pada kemampuannya dalam mengintegrasikan model dengan antarmuka interaktif secara efisien, sehingga mendukung proses demonstrasi dan pengujian fungsional sistem.

Dalam arsitektur keseluruhan, Streamlit berfungsi sebagai lapisan presentasi yang menghubungkan proses unggah citra, pemanggilan model klasifikasi Vision transformer dan XGBoost, serta perhitungan estimasi kalori berbasis TKPI. Framework ini tidak memengaruhi proses pelatihan model, melainkan berperan dalam menampilkan hasil klasifikasi dan perhitungan secara terstruktur kepada pengguna. Struktur antarmuka terdiri atas *sidebar* navigasi dan area konten utama. *Sidebar* menyediakan menu utama berupa Beranda, Identifikasi, Riwayat, dan Tentang Sistem. Desain ini bertujuan menjaga konsistensi navigasi dan memudahkan pengguna berpindah antarfitur tanpa mengganggu alur sistem.

a. Halaman Beranda

Halaman beranda berfungsi sebagai titik awal penggunaan sistem. Pada halaman ini ditampilkan judul sistem, deskripsi singkat mengenai fungsi aplikasi, serta daftar bahan pangan yang didukung. Tersedia pula tombol untuk memulai proses identifikasi. Penyajian informasi awal ini bertujuan memberikan orientasi penggunaan sebelum pengguna memasuki tahap unggah citra.



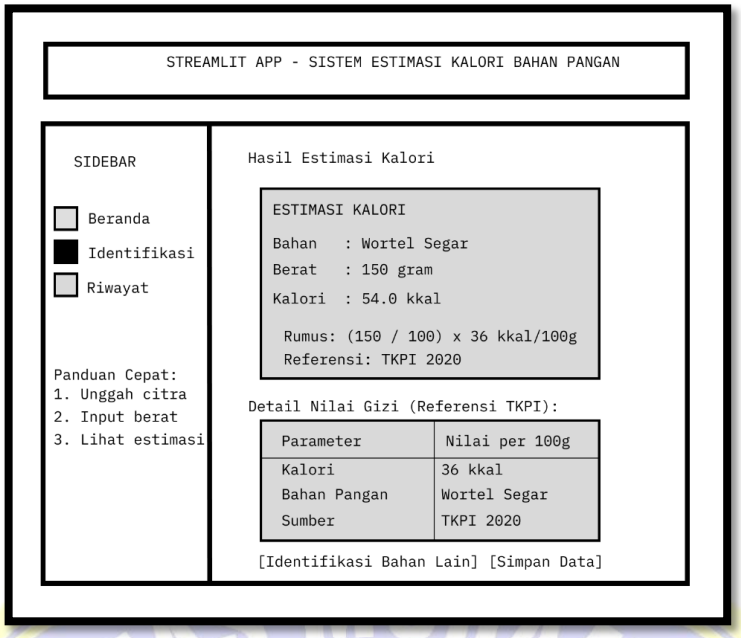
Gambar 3. 6 Halaman Beranda

b. Halaman Identifikasi Bahan Pangan

Halaman identifikasi merupakan inti interaksi sistem. Pengguna dapat mengunggah citra bahan pangan melalui fitur unggah file dengan format JPG atau PNG. Sistem menampilkan pratinjau citra dan melakukan proses resize otomatis menjadi 224 × 224 piksel sebelum dikirim ke model klasifikasi.

Setelah proses inferensi selesai, sistem menampilkan hasil prediksi berupa:

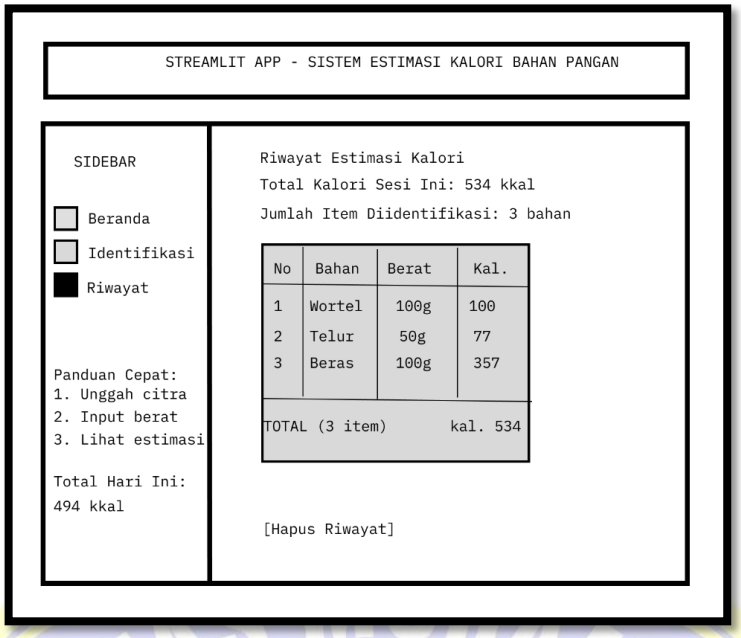
- Label jenis bahan pangan
- Estimasi kalori berdasarkan berat yang dimasukkan.



Gambar 3. 7 Halaman Identifikasi Bahan Pangan

c. Input Berat dan Perhitungan Estimasi Kalori

Setelah bahan pangan teridentifikasi, pengguna memasukkan berat bahan dalam satuan gram melalui kolom input numerik. Sistem melakukan validasi sederhana untuk memastikan nilai yang dimasukkan berupa angka positif. Nilai kalori per 100 gram berdasarkan TKPI ditampilkan secara eksplisit sebelum proses perhitungan dilakukan. Setelah pengguna menekan tombol perhitungan, sistem menghitung estimasi kalori menggunakan Persamaan (3.16) yang telah dijelaskan pada bagian Perhitungan Kalori Berbasis TKPI. Hasil estimasi kalori kemudian ditampilkan bersama informasi berat bahan, nilai kalori referensi TKPI, serta rincian hasil perhitungan. Penyajian informasi tersebut bertujuan meningkatkan transparansi sistem sehingga pengguna dapat memahami dasar perhitungan estimasi kalori yang dihasilkan.



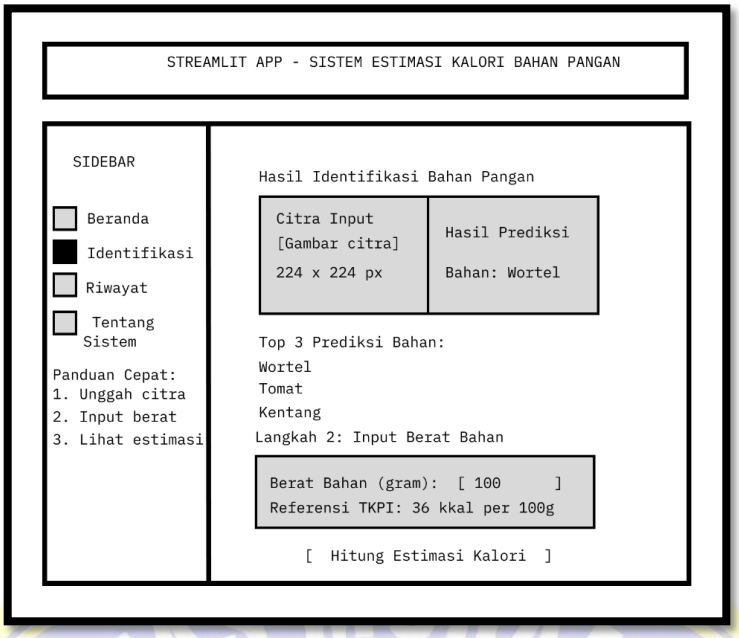
Gambar 3. 8 Input Berat dan Perhitungan Estimasi Kalori

d. Halaman Hasil Estimasi

Halaman hasil menampilkan ringkasan informasi berupa:

- Nama bahan pangan
- Berat yang dimasukkan
- Estimasi kalori total
- Referensi nilai kalori

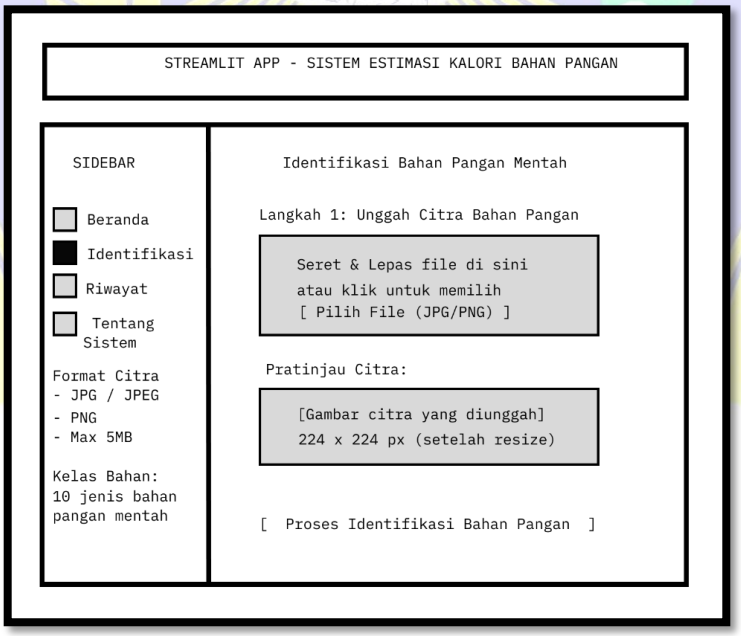
Penampilan rumus secara eksplisit bertujuan menunjukkan bahwa estimasi dilakukan secara deterministik berdasarkan standar nasional, bukan melalui prediksi regresi.



Gambar 3. 9 Halaman Hasil Estimasi

e. Halaman Riwayat Estimasi

Fitur riwayat memungkinkan pengguna melihat daftar bahan yang telah diidentifikasi dalam satu sesi penggunaan. Informasi yang ditampilkan meliputi nama bahan, berat, dan estimasi kalori. Sistem juga menghitung total kalori kumulatif dari seluruh bahan yang diidentifikasi dalam sesi tersebut. Fitur ini bersifat opsional dan tidak memengaruhi evaluasi kinerja model klasifikasi, melainkan ditujukan untuk meningkatkan fungsionalitas prototipe.



Gambar 3. 10 Halaman Riwayat Estimasi

3.4 Rencana Evaluasi

Evaluasi kinerja model dalam penelitian ini difokuskan pada klasifikasi bahan pangan multiclass. Prosedur evaluasi menerapkan metode validasi machine learning yang mengintegrasikan strategi pembagian dataset dan metrik evaluasi. Pembagian data menggunakan pendekatan stratified split (training, validation, dan test) untuk menjaga proporsi sampel pada setiap tahap. Selanjutnya, performa model dievaluasi menggunakan metrik precision, recall, F1-score, dan accuracy guna menggambarkan hasil klasifikasi pada setiap kelas.

a. *Confusion matrix*

Confusion matrix merupakan metode evaluasi yang digunakan untuk mengukur performa model klasifikasi dengan membandingkan hasil prediksi terhadap label sebenarnya. Pada penelitian ini, model melakukan klasifikasi multi kelas terhadap sepuluh jenis bahan pangan. Secara umum, *confusion matrix* untuk klasifikasi multi kelas direpresentasikan dalam bentuk matriksberukuran $C \times C$, dengan $C = 10$ kelas. Setiap elemen matriks merepresentasikan jumlah prediksi yang benar maupun salah antar kelas. Setiap elemen matriks dinotasikan sebagai n_{ij}, yang menyatakan jumlah data dengan kelas sebenarnya ke-i yang diprediksi sebagai kelas ke-j.

Untuk setiap kelas c, didefinisikan:

- True Positive (TP_c)
- False Positive (FP_c)
- False Negative (FN_c)
- True Negative (TN_c)

Nilai ini menjadi dasar perhitungan metrik evaluasi. Contoh struktur umum *confusion matrix* dalam penelitian ini ditunjukkan pada Tabel 3.5.

Tabel 3. 5 Contoh Format Tabel Confusion matrix

True \ Predicted	**Beras**	**Telur**	**Kentang**	**…**	**Terung**
Beras	n1,1	n1,2	n1,3	…	n1,10
Telur	n2,1	n2,2	n2,3	…	n2,10
Kentang	n3,1	n3,2	n3,3	…	n3,10
…	…	…	…	…	…
Terung	n10,1	n10,2	n10,3	…	n10,10

Baris menunjukkan kelas sebenarnya (true label), sedangkan kolom menunjukkan kelas hasil prediksi model (predicted label). Elemen diagonal n_ii merepresentasikan jumlah prediksi yang benar untuk masing-masing kelas.

b. Precision

Precision untuk kelas ke-c didefinisikan sebagai:

$$Precision_c = \frac{TP_c}{TP_c + FP_c} \tag{3.28}$$

Precision mengukur proporsi prediksi positif yang benar. Nilai *precision* tinggi menunjukkan bahwa ketika model memprediksi suatu bahan, kemungkinan prediksi tersebut benar relatif besar.

c. *Recall*

Recall untuk kelas ke-c dinyatakan sebagai:

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \tag{3.29}$$

Recall mengukur kemampuan model dalam menemukan seluruh sampel yang benar pada kelas tersebut. Nilai *recall* tinggi menunjukkan bahwa model jarang melewatkan sampel yang seharusnya termasuk dalam kelas tersebut.

d. *F1-score*

F1-score merupakan rata rata harmonik antara *precision* dan *recall*:

$$F1_c = 2 \times \frac{Precision_c \times Recall_c}{Precision_c + Recall_c} \tag{3.30}$$

F1-score digunakan untuk menyeimbangkan *precision* dan *recall*, terutama ketika terdapat perbedaan distribusi kesalahan antar kelas.

e. *Accuracy*

Accuracy mengukur proporsi prediksi yang benar terhadap seluruh sampel:

$$Accuracy = \frac{\sum_{c=1}^{C} T\,P_c}{N} \tag{3.31}$$

dengan *N*adalah jumlah total sampel pada data test.

Accuracy memberikan gambaran umum performa model, namun tidak selalu mencerminkan performa tiap kelas secara detail, terutama pada dataset yang tidak sepenuhnya seimbang.

Tabel 3. 6 Contoh Tabel Performa Model

Nama Kelas	***Precision***	***Recall***	***F1-score***	**Support**
Beras putih				
Telur ayam ras segar				
Kentang Segar				
Wortel Segar				
Bawang Merah Segar				
Bawang putih Segar				
Cabai Merah Segar				
Jagung Kuning Mentah				
Tomat Merah Segar				
Terung Ungu Segar				
***Macro* Average**				
***Micro* Average**				

f. *Macro* dan *Micro* Average

Karena penelitian ini menggunakan klasifikasi multiclass, maka nilai *precision*, *recall*, dan *F1-score* dihitung dalam dua pendekatan agregasi:

***Macro* Average**

Menghitung rata rata metrik dari seluruh kelas tanpa mempertimbangkan jumlah sampel per kelas:

$$Precision_{macro} = \frac{1}{C}\sum_{c=1}^{C} Precisio\, n_c \quad (3.32)$$

Macro average memberikan bobot yang sama untuk setiap kelas.

***Micro* Average**

Menggabungkan seluruh nilai TP, FP, dan FN dari semua kelas sebelum menghitung metrik:

$$Precision_{micro} = \frac{\sum TP}{\sum TP + \sum FP} \quad (3.33)$$

Micro average lebih mencerminkan performa keseluruhan sistem. Dalam penelitian ini, nilai *macro F1-score* digunakan sebagai metrik utama karena memberikan representasi yang lebih adil terhadap seluruh kelas bahan pangan.

g. Support

Support menunjukkan jumlah sampel setiap kelas pada data uji dalam confusion matrix, sehingga membantu interpretasi nilai precision, recall, dan F1-score. Hasil evaluasi kemudian diimplementasikan dalam prototipe berbasis Streamlit yang memungkinkan pengguna mengunggah citra, memasukkan berat, serta memperoleh hasil klasifikasi dan estimasi kalori. Kinerja model hibrida Vision Transformer–XGBoost juga dibandingkan dengan model CNN menggunakan data yang sama berdasarkan precision, recall, F1-score, dan accuracy.