BAB III
METODE PENELITIAN

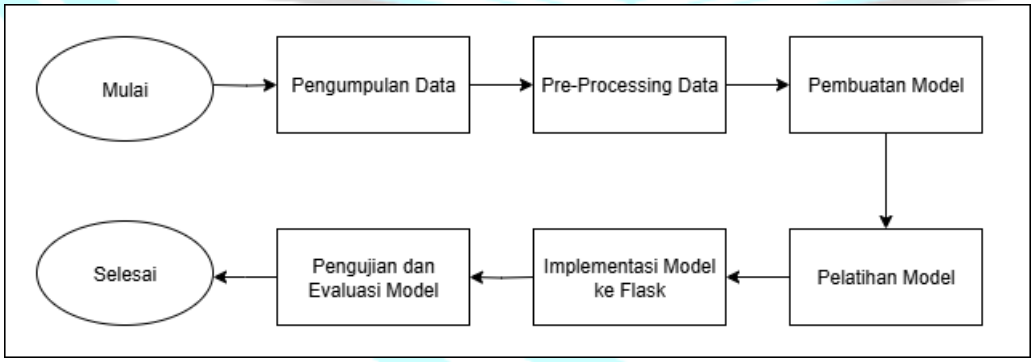
3.1 Objek Penelitian

Objek penelitian dalam tugas akhir ini adalah botol plastik jenis *Polyethylene Terephthalate* (PET) dan *High-Density Polyethylene* (HDPE). Jenis ini dipilih karena merupakan limbah plastik yang paling banyak ditemukan dan memiliki karakteristik fisik yang memungkinkan untuk dibedakan menggunakan teknologi *Computer Vision*. Penelitian ini berfokus pada pengembangan sistem pemilah sampah botol plastik otomatis berbasis teknologi *Computer Vision*.

Lokasi penelitian dilakukan di lingkungan rumah dan lingkungan Universitas Buana Perjuangan Karawang. Penelitian berlangsung selama periode November 2024 hingga April 2025.

3.2 Prosedur Penelitian

Penelitian ini dilaksanakan melalui beberapa tahapan yang sistematis dan berurutan, dimulai dari pengumpulan data, *pre-processing* data, pembuatan model, pelatihan model, implementasi model pada *Flask*, hingga pengujian dan evaluasi model. Rancangan penelitian dapat dilihat pada Gambar 3.1.

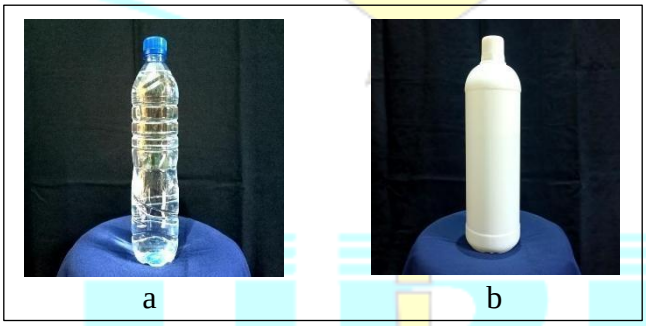


Gambar 3. 1 Rancangan Penelitian

3.2.1 Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil citra botol plastik PET dan HDPE menggunakan kamera *Webcam* yang berukuran 71,5 x 30 x 50 mm dengan resolusi 1920x1080 *pixel*. Proses ini dilakukan di bawah

kondisi pencahayaan yang terkontrol untuk memastikan kualitas gambar yang baik. Target pengumpulan data adalah 490 gambar untuk setiap jenis plastik, sehingga total gambar yang diperoleh adalah 980. Setiap gambar diambil dengan berbagai variasi sudut pandang dan kondisi pencahayaan untuk memperkaya variasi dalam *dataset*. Setelah pengumpulan selesai, data akan dibagi menjadi tiga kelompok, yaitu dataset pelatihan (70%), validasi (20%), dan pengujian (10%). Pembagian ini dilakukan untuk memastikan bahwa model memiliki data yang cukup untuk belajar, divalidasi, dan diuji.



a b

Gambar 3. 2 (a). Botol PET, (b). Botol HDPE

3.2.2 *Pre-Processing* Data

Data yang telah dikumpulkan akan melalui tahap *pre-processing* yang meliputi beberapa langkah penting untuk memastikan kualitas data yang digunakan dalam pelatihan model. Proses *pre-processing* ini bertujuan untuk mempersiapkan data sehingga model dapat memprosesnya dengan baik. Berikut adalah langkah-langkah yang akan dilakukan.

1. *Resize*

   Gambar dalam dataset akan disesuaikan ukurannya menjadi standar, seperti 224x224 piksel, agar kompatibel dengan kebutuhan input. Berikut adalah contoh *pseudocode* untuk proses *resize* (penyesuaian ukuran gambar).

   Pseudocode :

   Mulai

   // Baca gambar dari file atau dataset
   gambar = BacaGambar("path/to/gambar")

// Ubah ukuran gambar menjadi 224x224 piksel

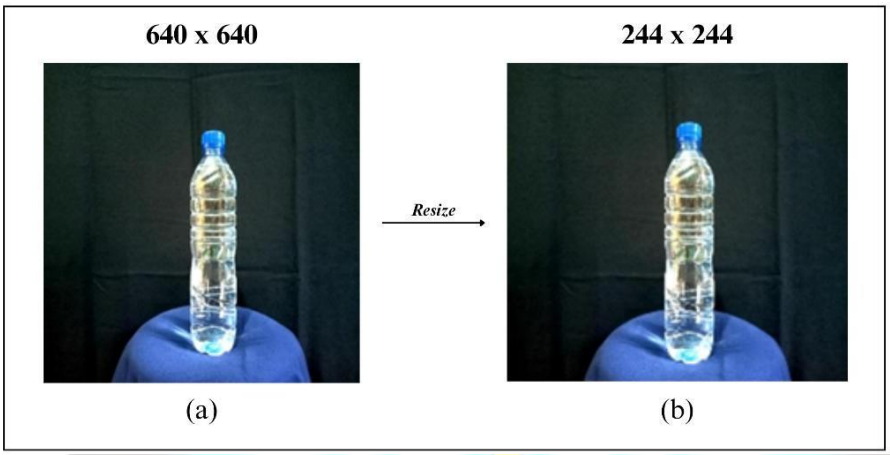
gambar*Resize* = UbahUkuran(gambar, 224, 224)

// Tampilkan gambar hasil *resize*

TampilkanGambar(gambar*Resize*)

Selesai

Hasil :



Gambar 3. 3 (a). Asli, (b). *Resized Image* (224x224)

2. *Augmentasi Data*

Teknik *augmentasi* yang digunakan mencakup rotasi gambar, pembalikan (*flipping*) baik secara *horizontal* maupun vertikal, serta penyesuaian tingkat kecerahan gambar. Langkah-langkah ini bertujuan untuk meningkatkan kemampuan generalisasi model terhadap variasi dalam kondisi nyata.

a. *Rotate Image*

Rotasi (*Rotated*) adalah teknik untuk memutar gambar pada sudut tertentu, seperti 10°, dengan tujuan meningkatkan variasi dalam data pelatihan. Berikut adalah contoh *pseudocode* untuk melakukan rotasi pada gambar.

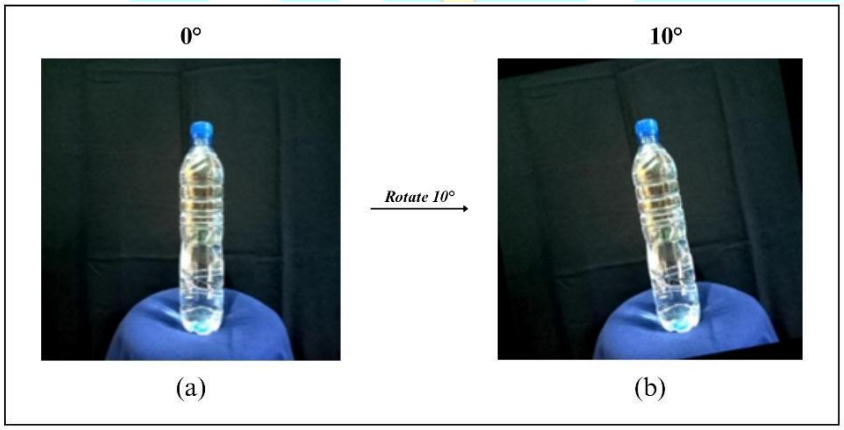
Pseudocode :

Mulai

```
// Baca gambar dari file atau dataset
gambar = BacaGambar("path/to/gambar")
// Tentukan sudut rotasi
sudutRotasi = 10°
// Putar gambar sesuai sudut yang ditentukan
gambarRotasi = PutarGambar(gambar, sudutRotasi)
// Simpan gambar hasil rotasi
SimpanGambar(gambarRotasi, "path/to/output")
```

Selesai

Hasil :



Gambar 3. 4 (a). Asli, (b). *Rotated* (10°)

b. *Flipping*

Flipping merupakan teknik *augmentasi* citra yang dilakukan dengan membalik gambar secara *horizontal* atau *vertikal* untuk menambah variasi data pelatihan dan meningkatkan kemampuan generalisasi model. Berikut adalah contoh *pseudocode* untuk melakukan *flipping* pada gambar.

Pseudocode Flipping Horizontal & Vertikal:

Mulai

```
// Baca gambar dari file atau dataset
gambar = BacaGambar("path/to/gambar")
// Balik gambar secara horizontal
```

gambar*Flip*H = BalikGambar(gambar, "*horizontal*")

// Balik gambar secara vertikal

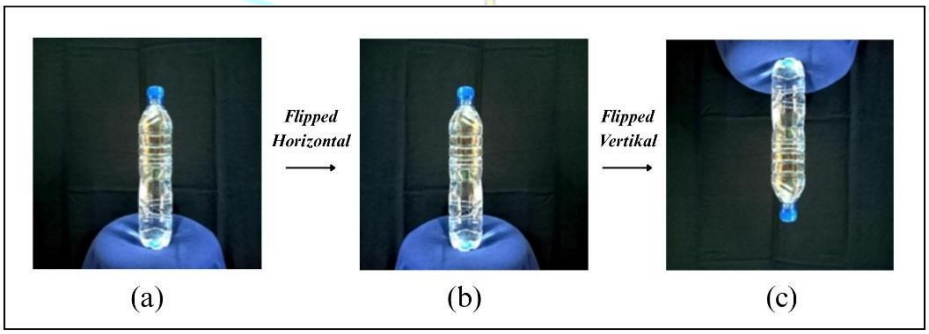
gambar*Flip*V = BalikGambar(gambar, "*vertikal*")

// Simpan gambar hasil *flipping*

SimpanGambar(gambar*Flip*H, "path/to/output")

Selesai

Hasil :



Gambar 3. 5 (a). Asli, (b). *Flipped Horizontal*, (c). *Flipped Vertikal*

c. *Brightness/Contrast Adjustment*

Brightness/Contrast Adjustment adalah teknik yang digunakan untuk mengubah tingkat kecerahan dan kontras suatu gambar, misalnya dengan menambahkan nilai 20 pada kecerahan dan mengalikan kontras dengan faktor 20. Teknik ini bertujuan untuk meningkatkan perbedaan antar piksel dalam gambar. Berikut adalah contoh *pseudocode* untuk melakukan *Brightness/Contrast Adjustment*.

Pseudocode:
Mulai

// Baca gambar dari file atau dataset

gambar = BacaGambar("path/to/gambar")

// Tentukan faktor kecerahan

faktorKecerahan = 20

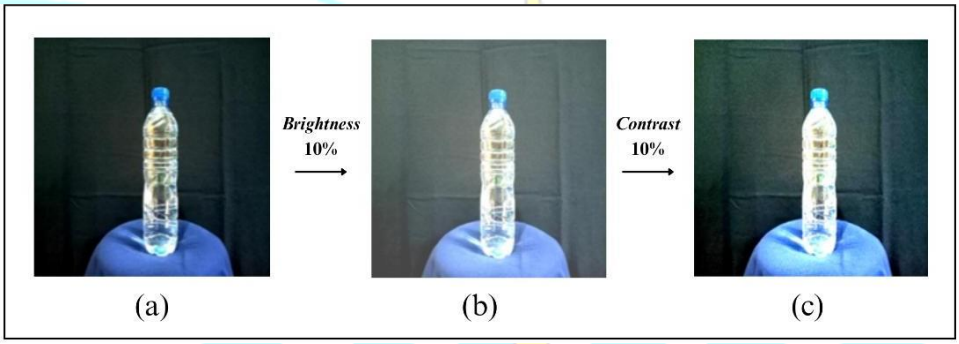
// Tentukan faktor kontras

faktorKontras = 20

// Ubah kecerahan gambar

gambarBrightness = UbahKecerahan(gambar, faktorKecerahan)
// Ubah kontras gambar
gambar*Contrast* = UbahKontras(gambar, faktorKontras)
// Simpan gambar hasil perubahan kecerahan
SimpanGambar(gambar*Brightness*, "path/to/output")
Selesai

Hasil :



Gambar 3. 6 (a). Asli, (b). *Brightness/Contrast Adjustment Image*

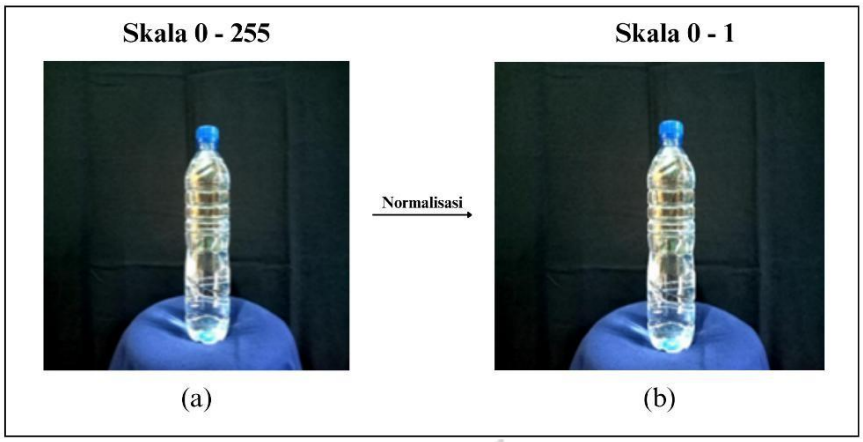
3. Normalisasi

Normalisasi merupakan teknik yang mengonversi nilai piksel gambar ke dalam rentang [0, 1] dengan membagi setiap piksel dengan 255.0. Proses ini bertujuan untuk meningkatkan konsistensi data dan mempercepat pelatihan model. Berikut adalah contoh *pseudocode* untuk melakukan normalisasi pada gambar.

Pseudocode :

Mulai
// Baca gambar dari file atau dataset
gambar = BacaGambar("path/to/gambar")
// Ubah nilai piksel gambar ke dalam rentang 0-1
gambarNormalisasi = BagiSetiapPiksel(gambar, 255.0)
// Simpan gambar hasil normalisasi
SimpanGambar(gambarNormalisasi, "path/to/output")
Selesai

Hasil :



Gambar 3. 7 (a). Asli, (b). *Normalized Image*

3.2.3 Pembuatan Model

Model CNN yang digunakan dalam penelitian ini adalah *MobileNetV2*, yaitu arsitektur *pre-trained* yang diadaptasi dengan menghapus lapisan *fully connected* pada bagian atasnya. Model ini kemudian ditambahkan dengan lapisan *Global Average Pooling*, *Dropout*, dan *Dense* dengan dua output kelas menggunakan fungsi aktivasi *softmax* untuk melakukan klasifikasi dua kelas. Proses pelatihan model diawali dengan warm-up training di mana seluruh lapisan *MobileNetV2* dikunci (*freeze*) agar hanya melatih lapisan tambahan. Setelah itu, dilakukan *fine-tuning* dengan membuka sebagian besar lapisan *MobileNetV2* untuk meningkatkan kemampuan ekstraksi fitur.

Model dikompilasi menggunakan *optimizer Adam* dengan *learning rate* 0.001 pada tahap awal, dan 0.00001 saat *fine-tuning* untuk menjaga stabilitas pembelajaran. Fungsi loss yang digunakan adalah *categorical crossentropy* yang sesuai untuk klasifikasi dua kelas dengan format *one-hot encoding*. Strategi pelatihan juga dilengkapi dengan *EarlyStopping* dan *ReduceLROnPlateau* untuk menghindari overfitting dan mengatur *learning rate* secara adaptif selama proses pelatihan.

Rancangan detail model disajikan pada Tabel 3.1.

Tabel 3. 1 Pembuatan Model CNN

Lapisan	Fungsi	Parameter	Penjelasan
Input Layer	Memasukkan gambar ke model	Ukuran: 224×224×3	Menerima input gambar berwarna berukuran 224x224 piksel dengan 3 channel (RGB).
Convolutional Layer 1	Mengekstraksi fitur awal dari gambar	Filter: 3x3, Stride: 2, Padding: Same	Menghasilkan feature map awal dengan fungsi aktivasi ReLU untuk meningkatkan non-linearitas.
Batch Normalization 1	Menormalkan hasil konvolusi	-	Mempercepat pelatihan dan menstabilkan distribusi data.
ReLU Activation 1	Memberikan non-linearitas	-	Mengaktifkan fitur penting dari hasil konvolusi awal.
DepthwiseConv2D Blocks	Mengekstraksi fitur kompleks	Filter: 3x3, Stride: bervariasi	Menggunakan depthwise separable convolution untuk mengurangi beban komputasi dan parameter model.
Expand Layer (Conv2D)	Memperluas jumlah channel fitur	Filter: 1x1	Menambah jumlah filter untuk meningkatkan kompleksitas representasi fitur.

Batch Normalization	Menormalkan hasil konvolusi	-	Menjaga distribusi data tetap stabil dalam proses pelatihan.
ReLU Activation	Memberikan non-linearitas	-	Mengaktifkan fitur-fitur penting dari hasil depthwise convolution.
Project Layer (Conv2D)	Mengurangi jumlah channel	Filter: 1x1	Mengembalikan jumlah channel ke ukuran yang lebih kecil untuk efisiensi.
Residual Add (Shortcut)	Menghubungkan input dan output blok	-	Mempercepat pelatihan dan mengurangi risiko hilangnya gradien dengan koneksi shortcut.
Zero Padding	Menyesuaikan dimensi	Padding: 1	Digunakan pada beberapa blok untuk memastikan ukuran output tetap sesuai setelah konvolusi.
Global Average Pooling	Merangkum fitur menjadi vektor 1 dimensi	-	Mengambil rata-rata dari setiap peta fitur, mengurangi dimensi data sebelum klasifikasi.
Dropout Layer	Mencegah overfitting	Dropout rate: 0.2	Mengurangi overfitting dengan menghilangkan neuron

Dense Layer (Output)	Melakukan klasifikasi akhir	Unit: 2 Neuron (Softmax Activation)	Menghasilkan probabilitas untuk masing-masing kelas: PET dan HDPE.

3.2.4 Pelatihan Model

Model dilatih menggunakan *dataset training* dalam beberapa *epoch*, di mana setiap *epoch* memproses seluruh data latih untuk memperbarui bobot model dengan tujuan menurunkan nilai *loss* dan meningkatkan akurasi prediksi. Selama proses pelatihan, model menggunakan ukuran batch tertentu yang mempengaruhi kestabilan dan kecepatan pelatihan. Ukuran *batch* yang kecil cenderung memberikan pembaruan bobot yang lebih halus, tetapi waktu pelatihan per *epoch* menjadi lebih lama. Sebaliknya, batch yang besar dapat mempercepat pelatihan, namun berisiko membuat model sulit melakukan generalisasi dan berpotensi mengalami *overfitting*.

Untuk mengoptimalkan proses pembelajaran, digunakan *optimizer Adam* yang efektif dalam mempercepat konvergensi dengan memanfaatkan momentum dan penyesuaian adaptif pada *learning rate*. *Learning rate* yang digunakan dipantau selama pelatihan, dan dapat diturunkan secara bertahap jika diperlukan agar model tetap stabil dan mampu mencapai hasil yang optimal tanpa fluktuasi yang besar dalam pembaruan bobot.

Selama pelatihan berlangsung, model dievaluasi secara berkala menggunakan dataset validasi. Data validasi berfungsi untuk menguji kemampuan model dalam mengenali pola pada data yang tidak dilatih secara langsung, sehingga dapat mengidentifikasi kemungkinan terjadinya *overfitting* atau *underfitting*. Jika model menunjukkan akurasi tinggi pada data *training* namun rendah pada data validasi, maka *overfitting* mungkin sedang terjadi. Oleh karena itu, proses validasi yang berjalan selama pelatihan sangat penting untuk memastikan model tetap mampu melakukan generalisasi.

Selain itu, dalam pelatihan ini diterapkan *callback* yang berfungsi untuk menghentikan pelatihan secara otomatis apabila akurasi model telah

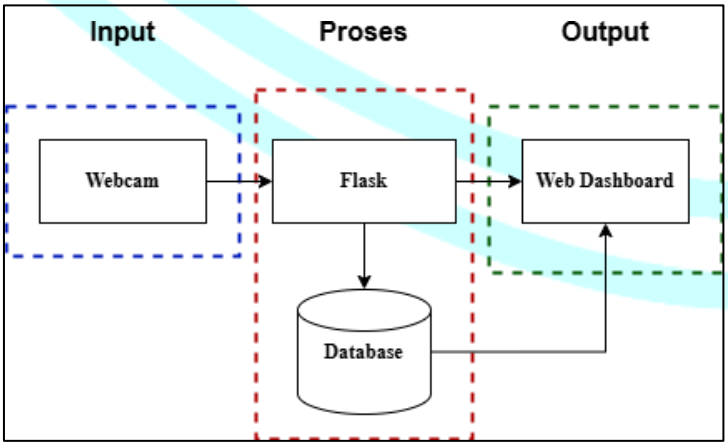
mencapai target tertentu, yaitu 99%. Jika target tersebut tercapai pada salah satu *epoch*, pelatihan akan langsung dihentikan untuk menghemat waktu dan sumber daya komputasi. Strategi ini bertujuan agar model tidak terlalu lama berlatih dan menghindari risiko menghafal data latih secara berlebihan.

Proses pelatihan juga memanfaatkan GPU untuk mempercepat komputasi, mengingat model *MobileNetV2* memerlukan pengolahan matriks dalam jumlah besar, khususnya pada operasi konvolusi dan *backpropagation*.

Sebagai bagian dari evaluasi, grafik riwayat akurasi dan *loss* pada data *training* serta validasi akan divisualisasikan untuk memberikan gambaran perkembangan model selama pelatihan. Melalui grafik ini, dapat diketahui apakah model mengalami *overfitting*, *underfitting*, atau sudah cukup stabil. Misalnya, jika *loss* pada data validasi mulai meningkat sementara akurasi training terus naik, ini menandakan *overfitting*. Dengan memantau grafik tersebut, keputusan seperti penyesuaian *hyperparameter*, perubahan arsitektur model, atau penerapan teknik regularisasi dapat dilakukan secara tepat untuk meningkatkan performa model.

3.2.5 Implementasi Model pada *Flask*

Implementasi model dilakukan dengan arsitektur sistem yang mengintegrasikan beberapa komponen utama yaitu *Flask* sebagai server dan *dashboard web*, *database MySQL* sebagai penyimpanan hasil deteksi, serta laptop/*webcam* sebagai alternatif input gambar. Sistem ini beroperasi secara *real-time*.



Gambar 3. 8 Blok Diagram

1. *Flask* sebagai Web Server

   Flask berfungsi sebagai server utama yang menangani komunikasi data, memproses gambar menggunakan model *Convolutional Neural Network* (CNN), dan mengembalikan hasil prediksi dalam format JSON.

   Selain sebagai API server, Flask juga bertindak sebagai *web dashboard* yang menyajikan riwayat deteksi plastik dalam *database*. *Dashboard* ini memungkinkan pengguna untuk memonitor hasil deteksi secara langsung, termasuk melihat gambar yang telah diproses, jenis plastik yang terdeteksi, serta tingkat kepercayaan (*confidence score*) dari model.

2. *Database* untuk Menyimpan Hasil Deteksi

   Database MySQL (*phpMyAdmin*) digunakan untuk menyimpan setiap hasil prediksi yang dilakukan oleh sistem. Setiap kali model CNN melakukan prediksi, hasilnya akan dicatat dalam tabel *detection_log*, yang berisi informasi seperti :

   a. Jenis plastik yang terdeteksi (PET atau HDPE)
   b. *Confidence score* dari model
   c. Waktu deteksi

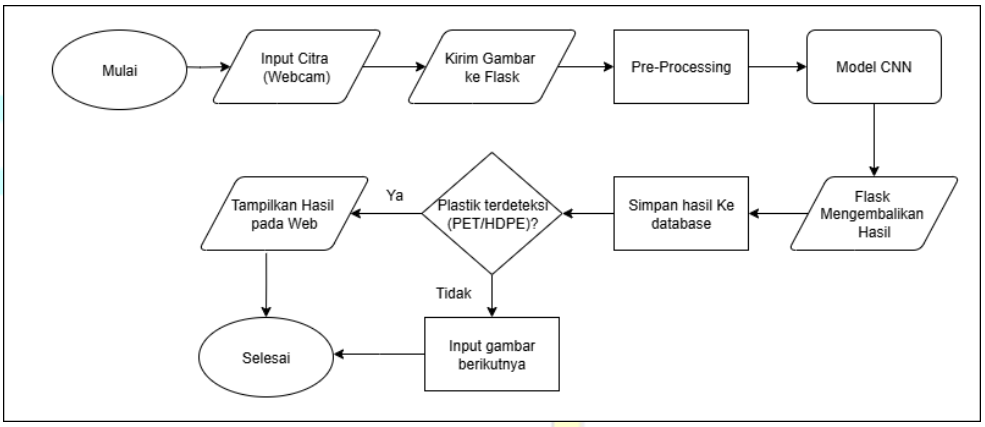
   Penyimpanan hasil deteksi ini memungkinkan analisis data jangka panjang, seperti jumlah plastik yang terdeteksi dalam periode tertentu dan pola deteksi yang dapat digunakan untuk optimasi model.

3. Laptop/*Webcam* sebagai Input Gambar

   Selain menerima input gambar dari *Webcam*, sistem juga memberikan fleksibilitas bagi pengguna dengan memungkinkan mereka mengunggah gambar atau menangkap gambar secara manual.

   Setelah gambar diunggah atau diambil, *Flask* akan langsung memprosesnya dengan model CNN dan menampilkan hasil deteksi di *web dashboard* serta menyimpannya ke *database*.

Alur keseluruhan sistem ini dirancang secara terstruktur untuk memastikan setiap proses berjalan sesuai dengan tujuan yang telah ditetapkan. Berikut adalah tahapan yang menggambarkan alur kerja sistem secara menyeluruh :



Gambar 3. 9 *Flowchart* Keseluruhan Sistem

Gambar 3.9 memperlihatkan alur kerja sistem deteksi botol plastik menggunakan *Computer Vision* berbasis CNN dan *Flask*. Proses dimulai dari tahap inisialisasi sistem, yang menandakan bahwa sistem siap untuk melakukan deteksi. Setelah itu, sistem akan menangkap gambar botol plastik melalui *webcam*, yang kemudian dikirim ke server *Flask* untuk diproses lebih lanjut. *Flask* bertanggung jawab dalam pengolahan gambar dan klasifikasi menggunakan model CNN yang telah dilatih sebelumnya.

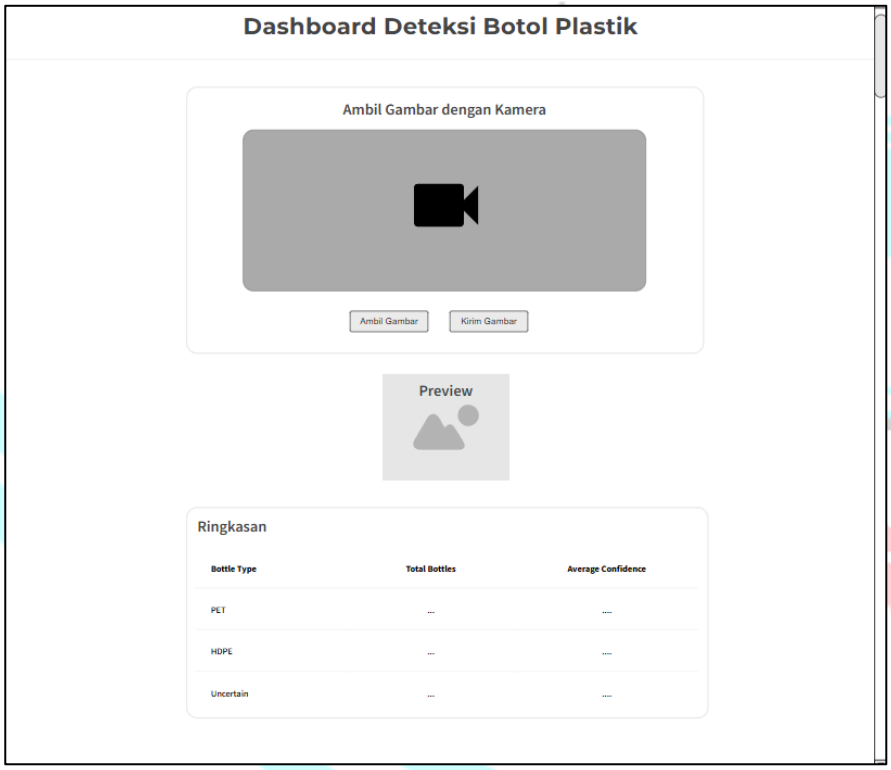
Setelah gambar diterima, sistem akan melakukan tahap *pre-processing* yang mencakup normalisasi warna, *resizing*, dan peningkatan kualitas gambar sebelum dikirim ke model CNN untuk proses klasifikasi. Model CNN kemudian menganalisis gambar dan mengidentifikasi apakah botol plastik yang terdeteksi termasuk dalam kategori PET atau HDPE. Setelah klasifikasi selesai, *Flask* mengembalikan hasil prediksi kepada sistem, yang kemudian disimpan dalam *database* agar dapat digunakan untuk pemantauan lebih lanjut.

Hasil prediksi yang telah disimpan kemudian ditampilkan pada *web*, sehingga pengguna dapat melihat secara langsung jenis plastik yang terdeteksi. Pada tahap ini, sistem akan melakukan pengecekan terhadap hasil klasifikasi. Jika plastik yang terdeteksi adalah PET atau HDPE, maka hasil

klasifikasi akan tampil pada *web dashboard*. Jika tidak ada plastik yang terdeteksi, maka sistem akan kembali ke tahap input gambar berikutnya, memungkinkan pengguna untuk mengambil gambar baru. Jika proses deteksi telah selesai dan tidak ada input gambar tambahan, sistem akan menuju tahap selesai, yang menandakan akhir dari alur kerja sistem.

Secara keseluruhan, *flowchart* ini menggambarkan integrasi antara *Computer Vision* dan *Flask* sebagai *backend* server. Dengan alur ini, sistem dapat secara otomatis mendeteksi jenis plastik, menyimpan hasilnya, serta memberikan notifikasi kepada pengguna.

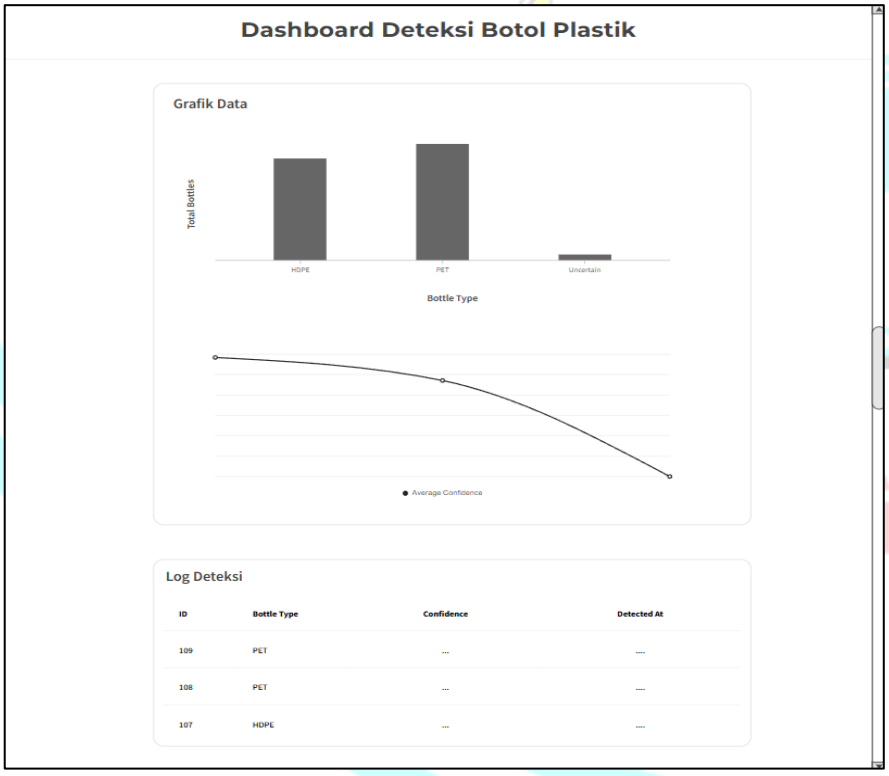
Berikut adalah perancangan antarmuka *Web Dashboard* :



Gambar 3. 10 Antarmuka *Live* Kamera dan Tabel Ringkasan

Gambar 3.10 menampilkan antarmuka halaman utama dari sistem deteksi botol plastik berbasis *Computer Vision*. Pada bagian atas terdapat judul utama "Dashboard Deteksi Botol Plastik" yang menunjukkan bahwa sistem ini digunakan untuk mendeteksi jenis botol plastik. Di bagian tengah, terdapat area tampilan kamera dengan ikon berbentuk kamera yang berfungsi

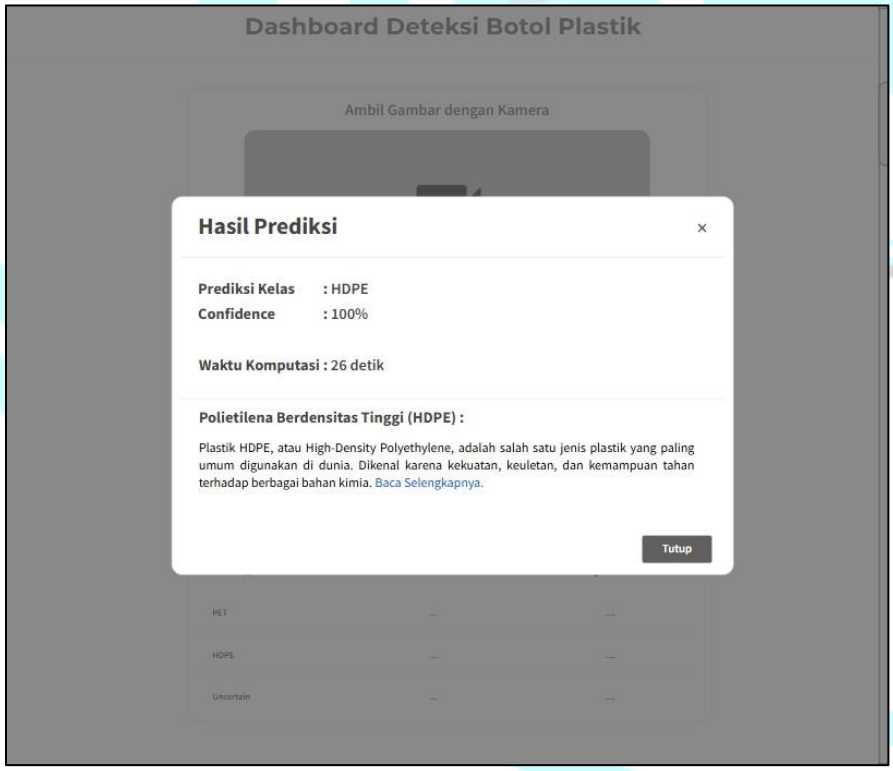
untuk mengambil gambar dari botol plastik yang akan diklasifikasikan. Di bawahnya terdapat dua tombol, yaitu "Ambil Gambar" untuk menangkap gambar menggunakan kamera dan "Kirim Gambar" untuk mengunggah gambar yang telah diambil agar diproses oleh sistem. Setelah gambar berhasil diunggah, hasilnya akan ditampilkan di bagian "*Preview*". Selain itu, terdapat tabel ringkasan hasil deteksi yang berisi informasi mengenai jumlah botol plastik yang terdeteksi berdasarkan jenisnya, yaitu PET, HDPE. Tabel ini juga menampilkan jumlah total botol yang terdeteksi dan rata-rata tingkat kepercayaan model dalam melakukan klasifikasi. Dalam gambar ini, data pada tabel belum diisi dan masih menampilkan *placeholder* berupa tanda titik-titik.



Gambar 3. 11 Antarmuka Grafik dan Tabel Deteksi *Log*

Gambar 3.11 memperlihatkan halaman visualisasi data yang menampilkan hasil deteksi dalam bentuk grafik serta catatan log deteksi. Pada bagian atas, terdapat grafik batang yang menunjukkan jumlah botol plastik yang telah terdeteksi berdasarkan jenisnya. Dari gambar tersebut, terlihat bahwa jumlah botol jenis HDPE dan PET lebih banyak dibandingkan dengan

kategori *Uncertain*. Di bawah grafik batang, terdapat grafik garis yang menampilkan tren rata-rata tingkat kepercayaan model terhadap hasil klasifikasi dari waktu ke waktu. Dari grafik ini, tampak bahwa tingkat kepercayaan model cenderung mengalami penurunan. Di bagian bawah halaman, terdapat tabel log deteksi yang mencatat setiap proses deteksi yang telah dilakukan oleh sistem. Tabel ini terdiri dari beberapa kolom, yaitu ID untuk menandai setiap proses deteksi secara unik, *Bottle Type* yang menunjukkan jenis botol plastik yang terdeteksi (PET atau HDPE), *Confidence* yang seharusnya menampilkan tingkat kepercayaan model terhadap hasil klasifikasi, serta *Detected At* yang menunjukkan waktu terjadinya deteksi. Dalam gambar ini, beberapa data telah tercatat dengan ID deteksi yang berurutan, namun kolom *Confidence* dan *Detected At* masih belum terisi.



Gambar 3. 12 Antarmuka Modal Hasil prediksi

Gambar 3.12 menampilkan modal yang muncul setelah sistem selesai melakukan klasifikasi botol plastik berdasarkan gambar yang diunggah oleh pengguna. Modal ini berjudul "Hasil Prediksi" dan berisi informasi detail

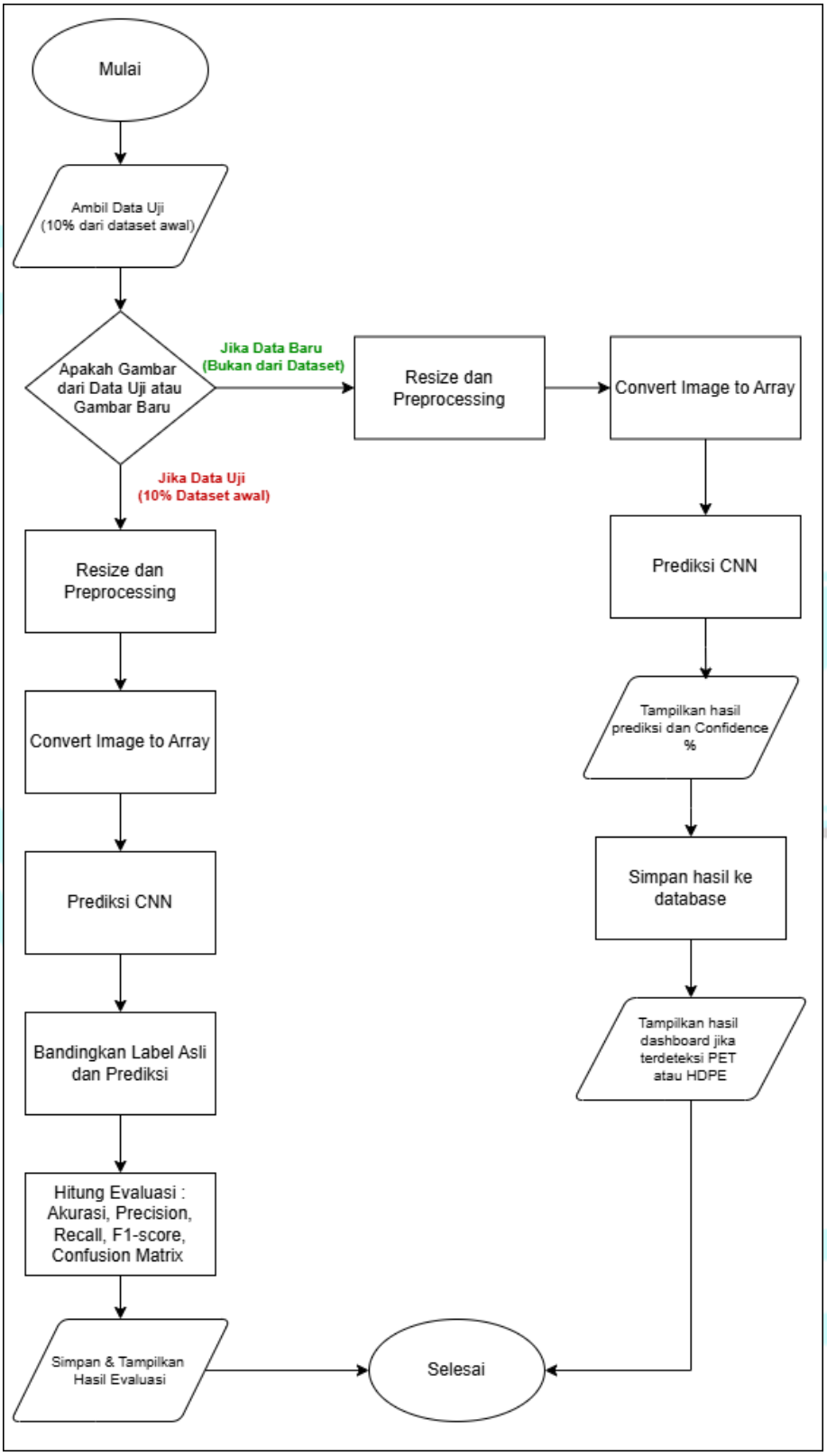
mengenai hasil deteksi yang dilakukan oleh model. Di dalam modal, terdapat beberapa elemen penting, di antaranya Prediksi Kelas yang menunjukkan bahwa sistem mengidentifikasi botol plastik sebagai jenis HDPE. *Confidence* ditampilkan dengan nilai 100%, yang berarti model sangat yakin dengan hasil klasifikasi yang diberikan. Selain itu, terdapat informasi mengenai Waktu Komputasi, di mana proses klasifikasi ini membutuhkan waktu selama 0,5 - 2 detik. Di bawah hasil prediksi, terdapat penjelasan tambahan mengenai *High-Density Polyethylene* (HDPE), yang merupakan jenis plastik yang kuat, fleksibel, serta tahan terhadap berbagai bahan kimia. Teks ini memberikan wawasan tambahan kepada pengguna tentang karakteristik dari jenis plastik yang terdeteksi. Di bagian akhir modal, terdapat tautan "Baca Selengkapnya" yang kemungkinan mengarah ke informasi lebih lanjut tentang plastik HDPE. Pengguna dapat menutup modal ini dengan menekan tombol "Tutup" yang terletak di bagian bawah.

3.2.6 Pengujian dan Evaluasi Model

Setelah Implementasi selesai, model akan diuji untuk menilai performanya. Pengujian dilakukan menggunakan dua jenis input: data uji dan gambar baru. Dataset yang digunakan pada tahap pengujian berasal dari pembagian awal dataset dengan proporsi 70% data latih, 20% data validasi, dan 10% data uji, yang tidak dilibatkan dalam proses pelatihan maupun validasi model. Tujuannya agar hasil evaluasi benar-benar menggambarkan performa model terhadap data yang belum pernah dilihat sebelumnya.

Selain itu, dilakukan juga pengujian menggunakan gambar baru (*real testing*) yang diunggah secara manual melalui *dashboard* atau diambil secara langsung melalui *webcam*. Pengujian ini bertujuan untuk mengevaluasi performa model secara langsung di lingkungan nyata, di luar dataset pelatihan awal. Hasil prediksi dari gambar baru ini akan langsung ditampilkan di *dashboard*.

Berikut adalah *Flowchart* Alur Pengujian :



Gambar 3. 13 Alur Pengujian

Beberapa pengujian yang dilakukan untuk mengevaluasi model adalah:

1. Tes Akurasi pada Data Uji

   Model akan dievaluasi menggunakan dataset uji untuk menghitung akurasi. Akurasi dihitung dengan membandingkan jumlah prediksi yang benar dengan total jumlah prediksi yang dilakukan oleh model. Hasil ini memberikan gambaran umum tentang seberapa baik model dalam mengklasifikasikan botol plastik antara PET dan HDPE.

2. *Confusion Matrix*

   Untuk memahami lebih dalam bagaimana model membuat keputusan, digunakan *confusion matrix*. *Confusion matrix* menunjukkan jumlah prediksi yang benar dan salah untuk masing-masing kelas (PET dan HDPE). Dengan analisis ini, dapat diketahui apakah model lebih cenderung salah mengklasifikasikan satu kelas dibandingkan kelas lainnya, serta area yang perlu diperbaiki.

3. Tes *Precision*, *Recall*, dan *F1-Score*

   a. *Precision* mendeskripsikan nilai presisi yang didapatkan dari pembagian jumlah total contoh positif, saat benar diklasifikasikan melalui model (Fitrah et al., 2025). misalnya berapa banyak prediksi PET yang benar dari semua yang diprediksi sebagai PET. Rumus *precision* yang digunakan untuk kalkulasi nilai *precision* ditulis melalui Persamaan 2.

   $$Precision = \frac{TP}{TP+FP} \quad (2)$$

   b. *Recall* mengukur seberapa banyak sampel dari kelas tertentu yang berhasil dikenali dengan benar oleh model, misalnya berapa banyak botol PET yang terdeteksi dengan benar dari semua botol PET yang ada di dataset uji. Rumus *recall* yang digunakan untuk kalkulasi nilai *recall* ditulis melalui Persamaan 3.

   $$Recall = \frac{TP}{TP+FN} \quad (3)$$

c. *F1-Score* adalah kombinasi dari *precision* dan *recall*, yang berguna terutama jika terdapat ketidakseimbangan jumlah data antara kelas PET dan HDPE. Rumus *FI-Score* yang digunakan ditulis melalui Persamaan 4.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

4. Visualisasi Hasil Pelatihan

Selama proses pelatihan, grafik akurasi dan *loss* digunakan untuk memantau performa model pada data latih dan validasi.

a. Grafik akurasi menunjukkan bagaimana akurasi model meningkat seiring waktu pada data latih dan validasi.

b. Grafik *loss* menunjukkan bagaimana *error* model menurun selama pelatihan.

c. Jika terjadi *overfitting*, akurasi data latih mungkin tinggi, tetapi akurasi validasi rendah atau *loss* validasi meningkat.

5. Pengujian Model dengan Gambar Baru

Setelah model dilatih dan dievaluasi pada data uji, model akan diuji dengan gambar baru yang diunggah oleh pengguna.

a. Gambar akan diproses dan dikonversi menjadi format yang sesuai sebelum dimasukkan ke dalam model.

b. Model akan memberikan prediksi jenis plastik (PET atau HDPE) serta tingkat kepercayaan (%) dari prediksi tersebut.

c. Hasil prediksi akan ditampilkan di layar dengan anotasi untuk menunjukkan apakah gambar diklasifikasikan sebagai PET atau HDPE.