

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1 *Spesifikasi Perangkat Keras (Hardware)*

Perangkat keras yang digunakan dalam pembuatan sistem implementasi sampah organik dan non-organik ini adalah sebagai berikut:

- a. Intel® Core™ i5-7200U CPU @ 2.50GHz 2.71 GHz
- b. RAM 4,00 GB
- c. 128 GB SSD dan 1000 GB HDD
- d. kamera.

4.2 **Spesifikasi Perangkat Lunak (Software)**

Perangkat lunak yang digunakan dalam pembuatan sistem implementasi sampah organik dan non-organik ini adalah sebagai berikut:

- a. Sistem operasi Microsoft Windows 10 64bit;
- b. YOLOv8s.pt. model yang digunakan
- c. Google Colab *Research Python* diperlukan untuk pengembangan dan pelatihan model YOLOv8
- d. Roboflow diperlukan untuk pebelan citra gambar
- e. Versi Ultralytics adalah 8.1.14 dan menggunakan Python-3.10.12 dengan library torch-2.1.0+cu121 pada perangkat CUDA:0 (Tesla T4, 15102MiB)

4.3 **Deskripsi Penelitian**

Pada bagian ini, dilakukan evaluasi data berdasarkan pembagian dataset. Jumlah data citra sampah yang digunakan sebanyak 3021 sampah organik dan non organik. Untuk membandingkan akurasi dalam mengklasifikasikan data, penelitian ini melakukan pembagian data dengan rincian: 70% (2115) untuk train set, 20% (604) untuk valid set, dan 10% (302) untuk test set.

Tabel 4.1 Jumlah data organik Train Set

Data Organik	Jenis	Jumlah
Train Set	Kulit pisang	70
	Kulit Jeruk	46
	Sisa Makanan	39
	Daun	811
Total		966

Tabel 4.2 Jumlah data organik Test Set

Data Organik	Jenis	Jumlah
Test Set	Kulit pisang	14
	Kulit Jeruk	11
	Sisa Makanan	7
	Daun	113
Total		145

Tabel 4.3 Jumlah data organik Valid Set

Data Organik	Jenis	Jumlah
Valid Set	Kulit pisang	28
	Kulit Jeruk	15
	Sisa Makanan	21
	Daun	213
Total		277

Tabel 4.4 Jumlah data non organik Train Set

Data Non Organik	Jenis	Jumlah
Train Set	Plastik	39
	Botol	352
	Kertas	394
	Kaleng	364
Total		1.149

Tabel 4.5 Jumlah data non organik Tes Set

Data Non Organik	Jenis	Jumlah
Tes Set	Plastik	7
	Botol	44
	Kertas	57
	Kaleng	49
Total		157

Tabel 4.6 Jumlah data non organik Valid Set

Data Non Organik	Jenis	Jumlah
Valid Set	Plastik	20
	Botol	93
	Kertas	125
	Kaleng	89
Total		327

Tabel 4.7 Jumlah data Train Set

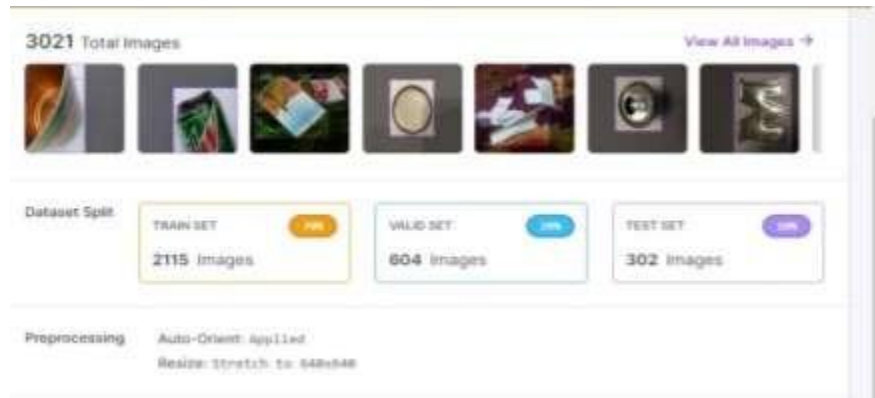
Data	Jumlah
Train Set Organik	966
Train Set Non Organik	364
Total	2115

Tabel 4.8 Jumlah data Test Set

Data	Jumlah
Test Set Organik	145
Test Set Non Organik	157
Total	302

Tabel 4.9 Jumlah data Valid Set

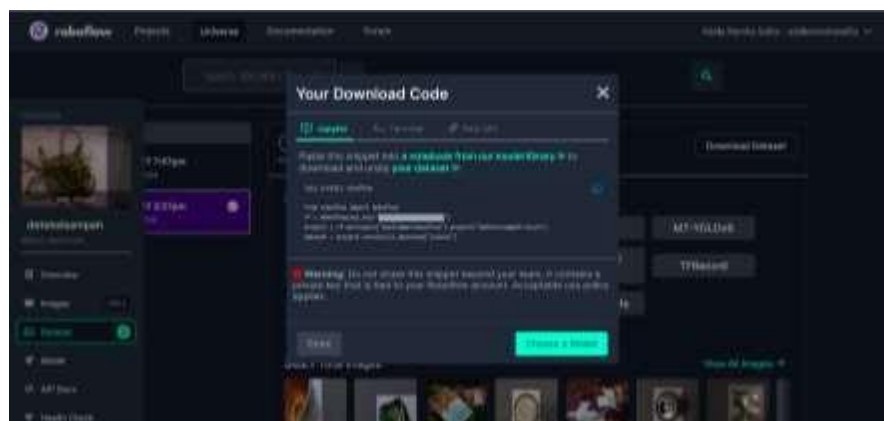
Data	Jumlah
Valid Set Organik	277
Valid Set Non Organik	327
Total	604



Gambar 4.1 Pembagian dataset

4.4 API Roboflow

API Roboflow adalah antarmuka pemrograman aplikasi (*Application Programming Interface*) yang disediakan oleh platform Roboflow. Api ini memungkinkan pengguna untuk berinteraksi dengan fitur- fitur Roboflow melalui kode pemrograman.



Gambar 4.2 Kode API Roboflow

4.5 Graphics Processing Unit (GPU)

Graphics Processing Unit (GPU) adalah jenis prosesor khusus yang dirancang untuk memproses grafik dan tugas komputasi paralel. Dalam konteks penggunaan YOLOv8, GPU memiliki peran krusial dalam meningkatkan kecepatan pelatihan dan inferensi model implementasi objek. Memanggil YOLOv8 menggunakan GPU dapat mempercepat proses implementasi objek, terutama saat berurusan dengan dataset yang besar dan tugas yang memerlukan daya komputasi tinggi seperti penglihatan komputer, untuk memanggil GPU menggunakan kode berikut ini.

Tabel 4.10 Graphics Processing Unit (GPU)

```
!nvidia-smi
```

Berikut ini merupakan *output* :

```
Sat Jan 27 13:00:55 2024
+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name      Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp       Perf         Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+-----+-----+-----+-----+
|   0   Tesla T4          Off          | 00000000:00:04:0 Off |             0         |
| N/A   48C        P8             10W / 70W |  0MiB / 15360MiB |      0%      Default |
|                                           MIG M.         |
+-----+-----+-----+-----+-----+-----+
+-----+
| Processes:                               GPU Memory |
|  GPU   GI    CI          PID    Type    Process name          Usage   |
|  ID   ID             |              |
+-----+-----+-----+-----+-----+-----+
| No running processes found              |
+-----+
```

Gambar 4.3 Output GPU

4.6 Menginstal Pustaka *Ultralytics*

Ultralytics adalah pustaka sumber terbuka yang menyediakan implementasi YOLOv8, sebuah arsitektur implementasi objek yang terkenal dalam bidang penglihatan komputer. Berikut ini merupakan kode.

Tabel 4.11 Menginstal Pustaka *Ultralytics*

```
# Pip install method (recommended)
!pip install ultralytics==8.0.196
from IPython import display
```

```
display.clear_output()
import ultralytics
ultralytics.checks()
```

Pada table di atas digunakan untuk menginstal *Ultralytics*, memberikan output tampilan, dan melakukan beberapa pemeriksaan terkait versi yang terinstal.

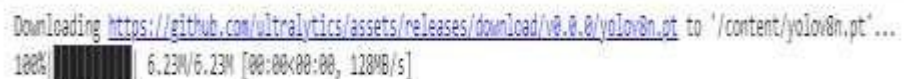
4.7 Implementasi Model

Tabel 4.12 Implementasi Model

```
model = YOLO(f'{HOME}/yolov8n.pt')
```

Berikut ini merupakan penjelasan dari tabel diatas:

Membuat sebuah objek model YOLO menggunakan berkas berat (weights) yang disimpan di jalur file yang diberikan, dengan menggunakan variabel `HOME` untuk menentukan direktori utama. Ini diasumsikan sebagai bagian dari implementasi sebuah model YOLO, di mana kelas `YOLO` mungkin telah didefinisikan sebelumnya untuk menginstansiasi model. Penggunaan `f'{HOME}/yolov8n.pt` digunakan untuk membentuk jalur lengkap ke berkas berat model, yang kemudian digunakan untuk inialisasi objek model YOLO.



```
Downloading https://github.com/ultralytics/assets/releases/download/v8.0.0/yolov8n.pt to '/content/yolov8n.pt' ...
100% [██████████] 6.23M/6.23M [00:00<00:00, 128MB/s]
```

Gambar 4.13 Implementasi Model

4.8 Penyusunan Dataset

Pada tahap ini kode API akan di input ke dalam google colab berikut ini merupakan tampilan koding nya.

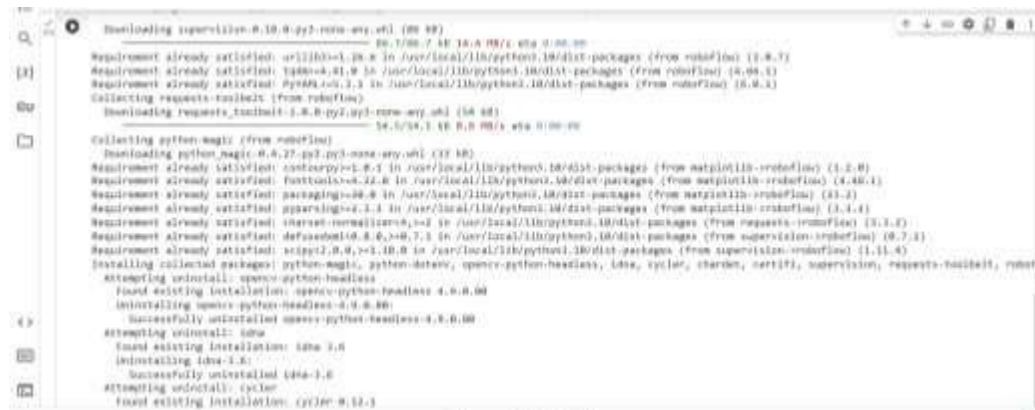
Tabel 4.13 Penyusunan Dataset

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets
```

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="LzcCePwZdtrPbAOcGgMC")
project = rf.workspace("adellamarshasafira").project("deteksisampah-5sx2n")
dataset = project.version(1).download("yolov8")
```

Berikut ini merupakan *output* nya:



Gambar 4.5 Penyusunan dataset

4.9 Custom Training

Kode tersebut merupakan perintah penggunaan YOLOv8 untuk melakukan tugas implementasi objek pada sistem implementasi sampah. Berikut adalah penjelasan untuk setiap bagian dari kode tersebut:

Tabel 4.14 Perintah *Shell*

```
`%cd {HOME}`:
`!yolo task=detect mode=train model=yolov8s.pt
data={dataset.location}/data.yaml epochs=1 imgsz=800
plots=True`:
```

Berikut penjelasan dari table diatas:

Keseluruhan perintah ini digunakan untuk melatih model YOLO versi 8 (yolov8s) dengan menentukan jumlah epochs 10 untuk tugas implementasi objek

menggunakan dataset yang diatur dalam file konfigurasi YAML yang diberikan.

4.10 Menampilkan File Dalam Train

Tabel 4.15 Menampilkan Daftar File dan Folder

```
!ls {HOME}/runs/detect/train/
```

Berikut penjelasan dari table diatas:

Digunakan untuk menampilkan isi dari direktori atau folder yang terletak di jalur yang didefinisikan oleh variabel HOME dan berada di bawah folder '**runs/detect/train/**'.

Dalam konteks ini, hasil perintah tersebut akan menampilkan daftar file dan folder yang terdapat di dalam direktori '**runs/detect/train/**' setelah proses pelatihan model YOLOv8 untuk implementasi objek pada dataset sampah. Isi dari direktori tersebut mungkin mencakup berbagai file, seperti bobot model yang disimpan, hasil evaluasi, atau file lain yang berkaitan dengan proses pelatihan. Perintah ini berguna untuk melihat hasil-hasil yang dihasilkan selama atau setelah proses pelatihan model.

4.11 Confusion Matrix

Tabel 4.16 Confusion Matrix

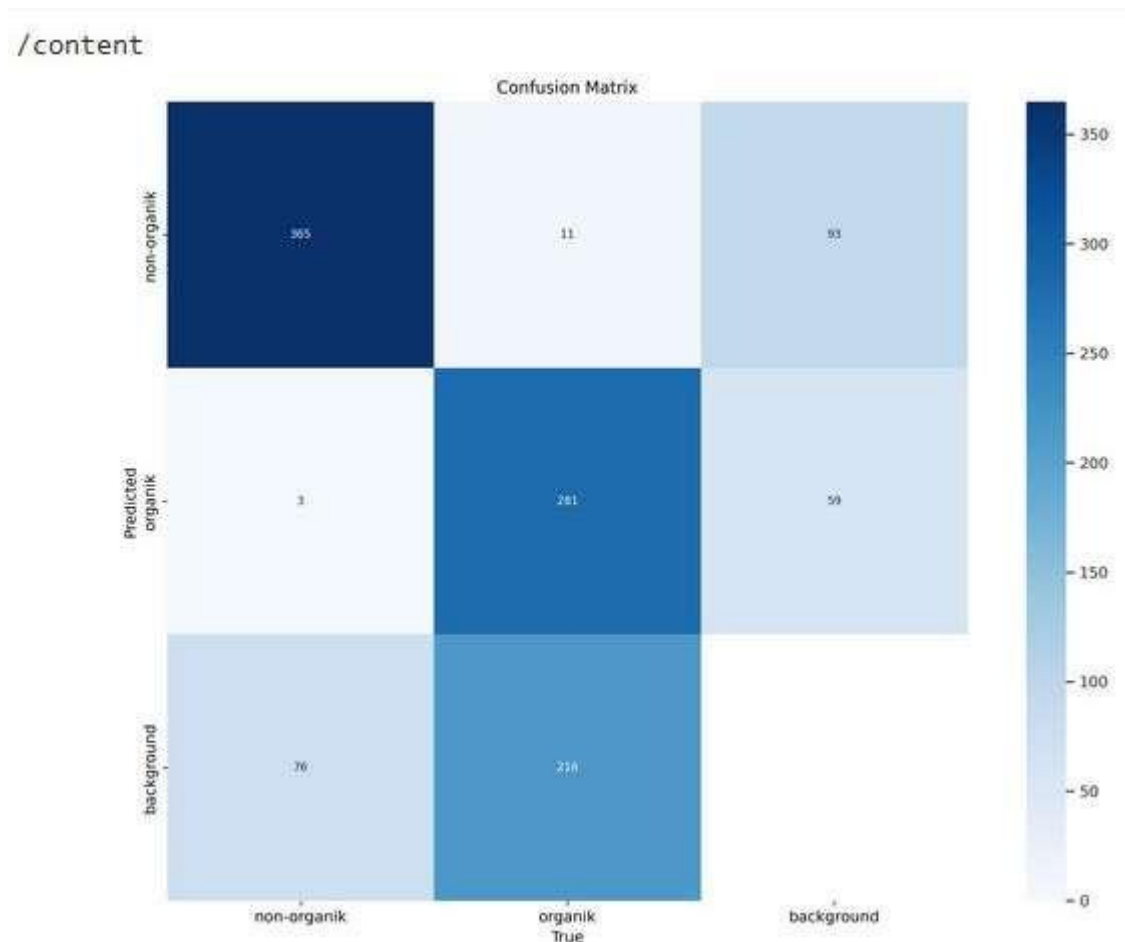
```
%cd {HOME}  
Image(filename=f {HOME}/runs/detect/train2/confusion_matrix.png',  
width=600)
```

Berikut ini merupakan penjelasan dari table diatas:

Menampilkan gambar matriks (*confusion matrix*) dari hasil implementasi objek menggunakan model YOLOv8 pada dataset sampah. Matriks kebingungan memberikan pandangan tentang sejauh mana model dapat membedakan antara kelas objek yang berbeda. Dengan menampilkan gambar ini, Anda dapat melakukan analisis visual terhadap performa model dalam hal implementasi objek pada data uji. Lebar gambar ditentukan

oleh parameter width yang diatur menjadi 600 piksel agar sesuai dengan tampilan di dalam notebook.

Berikut ini merupakan *output*



Gambar 4.6 Output confusion matrix

Confusion matrix (matriks kebingungan) adalah tabel yang digunakan untuk mengevaluasi kinerja suatu model klasifikasi. Dalam bahasa Indonesia, confusion matrix sering disebut juga sebagai matriks kebingungan. Normalized confusion matrix adalah versi dari confusion matrix yang dinormalisasi, artinya nilai-nilai di dalamnya dibagi dengan jumlah total data pengujian.

Berikut adalah elemen-elemen dalam *confusion matrix*:

1. True Positive (TP): Jumlah data positif yang benar-benar diprediksi dengan benar oleh model.
2. True Negative (TN): Jumlah data negatif yang benar-benar diprediksi dengan benar oleh model.
3. False Positive (FP): Jumlah data negatif yang salah diprediksi sebagai positif oleh model.
4. False Negative (FN): Jumlah data positif yang salah diprediksi sebagai negatif oleh model.

Dengan menggunakan nilai-nilai ini, kita dapat membuat confusion matrix:

TN & FP

FN & TP

Dengan *confusion matrix*, kita dapat menghitung berbagai metrik evaluasi model, seperti akurasi, presisi, recall, dan F1-score.

Normalized confusion matrix adalah confusion matrix yang setiap nilai di dalamnya dibagi dengan jumlah total data pengujian. Dengan kata lain, setiap nilai di dalamnya diukur sebagai persentase dari total data. Hal ini membantu untuk memahami seberapa baik model bekerja relatif terhadap keseluruhan dataset.

4.12 Results

Tabel 4.17 Results

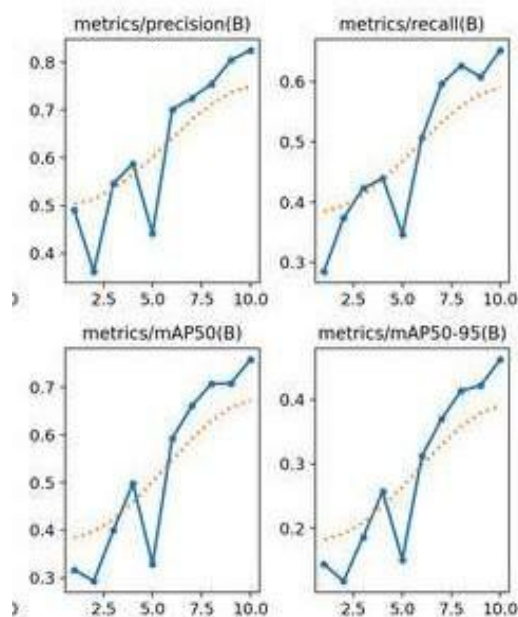
```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train2/results.png',
width=600)
```

Dalam YOLOv8 atau model implementasi objek serupa, memanggil "results" umumnya merujuk pada langkah terakhir dalam proses inferensi atau prediksi, di mana model memberikan hasil implementasi objek pada suatu gambar. Hasil ini dapat berupa sejumlah informasi terkait objek yang terdeteksi, seperti koordinat bounding box (kotak pembatas), kelas objek, dan skor kepercayaan.

Sebagai contoh, jika Anda menggunakan YOLOv8 untuk implementasi sampah organik dan non-organik dalam pelatihan, Anda akan mengambil hasil prediksi dari model setelah melakukan inferensi pada gambar-gambar pelatihan. Hasil ini mungkin berisi informasi seperti:

1. **Koordinat Bounding Box:** Koordinat titik awal dan titik akhir dari kotak pembatas yang mengelilingi objek sampah.
2. **Kelas Objek:** Label yang menunjukkan apakah objek yang terdeteksi adalah sampah organik atau non-organik.
3. **Skor Kepercayaan:** Sebuah skor yang mencerminkan sejauh mana model yakin bahwa implementasi objek tersebut benar.

Dengan menggunakan hasil tersebut, Anda dapat melakukan evaluasi performa model, menghitung metrik seperti keakuratan (accuracy), presisi (precision), recall, dan lainnya. Selain itu, hasil ini juga dapat digunakan untuk memvisualisasikan implementasi objek pada gambar-gambar pelatihan



Gambar 4.7 *Output Results*

4.13 Validasi Prediksi

Tabel 4.18 Kode Prediksi

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred
```

```
.jpg', width=600)
```

Kode ``%cd {HOME}`` digunakan untuk mengganti direktori kerja saat ini ke direktori yang ditentukan oleh variabel ``HOME``. Kemudian, ``Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg', width=600)`` digunakan untuk menampilkan gambar hasil prediksi yang terletak pada jalur file ``val_batch0_pred.jpg``. Gambar ini kemungkinan menunjukkan hasil prediksi dari model terhadap data validasi, yang berisi klasifikasi antara organik dan non-organik. Namun, tanpa melihat isi gambar tersebut, tidak mungkin menentukan dengan pasti apa yang menjadi penyebab kesalahan prediksi. Kesalahan yang disebutkan, yaitu memprediksi organik sebagai non-organik, bisa disebabkan oleh beberapa faktor seperti kurangnya representasi data organik dalam set pelatihan, kompleksitas model yang tidak sesuai, atau penyesuaian parameter yang dibutuhkan pada fase pelatihan. Diperlukan analisis lebih lanjut terhadap data, model, dan parameter untuk mengidentifikasi akar masalah dan meningkatkan performa prediksi.



Gambar 4.8 Hasil Validasi Prediksi1

4.14 Hasil Label Memanggil

Tabel 4.19 Kode Label Memanggil

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_labels.
jpg', width=600)
```

Kode ini menggunakan perintah `%cd {HOME}` untuk mengubah direktori kerja saat ini ke direktori yang diidentifikasi oleh variabel `HOME`. Selanjutnya, dengan menggunakan modul `Image` (mungkin dari library seperti PIL atau IPython.display), sebuah gambar hasil prediksi dari sistem pengenalan gambar organik dan non-organik ditampilkan. Gambar yang ditampilkan diberikan oleh berkas dengan nama `val_batch0_labels.jpg` yang berlokasi di dalam direktori `runs/detect/train/` yang terdapat di direktori utama yang telah diubah sebelumnya. Gambar ini mungkin memvisualisasikan hasil prediksi sistem terhadap suatu dataset validasi (val_batch0) dengan bounding box atau label untuk membedakan antara objek organik dan non-organik. Lebar gambar ditentukan sebagai 600 piksel.



Gambar 4.9 Hasil Label Memanggil

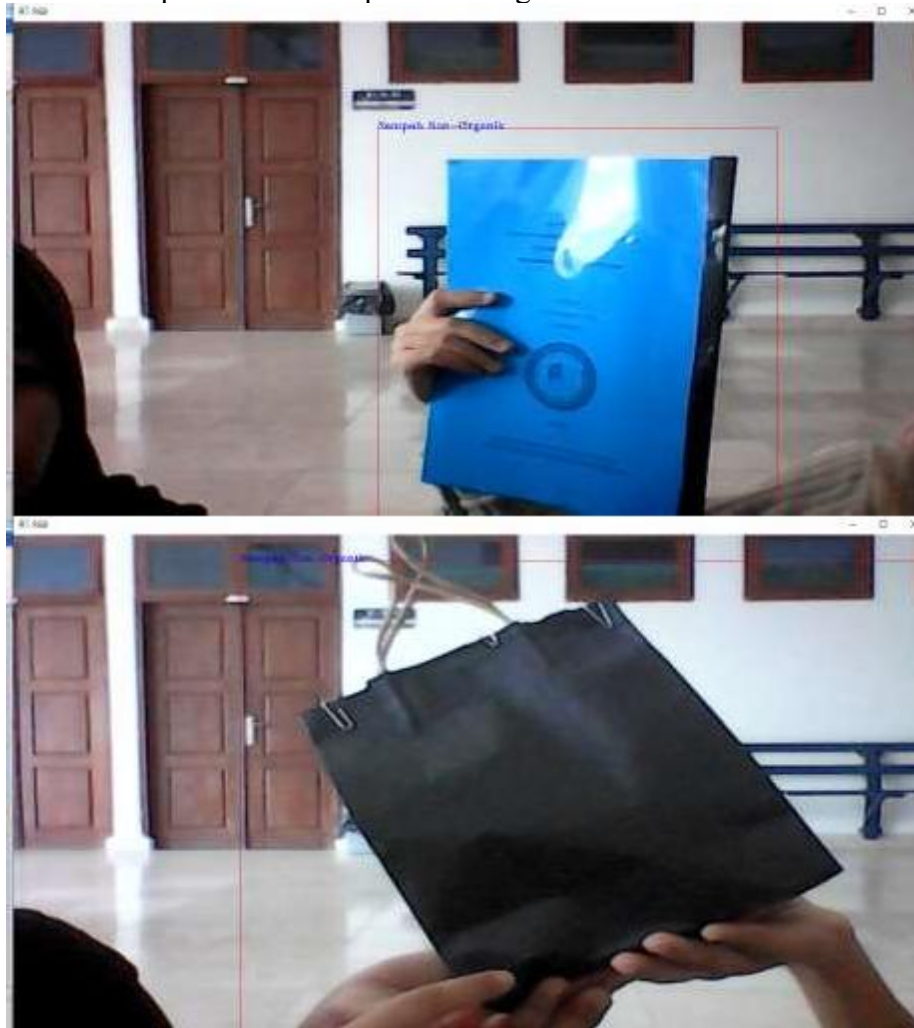
4.15 Hasil Implementasi Sampah Organik



Gambar 4.10 Daun

Dalam konteks implementasi gambar di atas, tampak jelas bahwa hasilnya mencerminkan batasan yang dapat menditek dari gambar daun dengan menggunakan kelas organik. Melalui analisis tersebut, sistem berhasil mengidentifikasi dan mengelompokkan gambar daun ke dalam kategori organik.

4.16 Hasil Implementasi Sampah Non Organik



Gambar 4.11 Kertas

Dalam konteks implementasi gambar yang terlihat pada gambar di atas, terlihat dengan jelas bahwa hasilnya merefleksikan batasan yang dapat implementasi dari gambar buku dan paperbag dengan penerapan kelas non-organik. Melalui analisis mendalam tersebut, sistem berhasil mengidentifikasi dan mengelompokkan gambar buku dan paperbag ke dalam kategori non organik.



Gambar 4.12 Case Hp

Dalam konteks implementasi gambar yang terlihat pada gambar di atas, terlihat dengan jelas bahwa hasilnya merefleksikan batasan yang dapat implementasi dari gambar case Hp dengan penerapan kelas non-organik. Melalui analisis mendalam tersebut, sistem berhasil mengidentifikasi dan mengelompokkan gambar case Hp ke dalam kategori non organik.