

BAB IV

HASIL PEMBAHASAN

4.1 Deskripsi Dataset Penelitian

Dataset dalam penelitian ini merupakan kumpulan gambar buah-buahan yang digunakan untuk melatih dan menguji kinerja model deteksi objek *YOLOv5*.

| Kelas objek | Jumlah gambar | Sumber gambar publik |
|--------------------|---------------|----------------------|
| Pisang Segar | 623 | Roboflow |
| Pisang Tidak Segar | 576 | Roboflow |

- Jumlah total gambar: 2.280 Gambar
- Resolusi gambar 640x480 hingga 1280x750 *pixel*
- Latar belakang gambar: nenerapa gambar memiliki latar belakang kompleks untuk menguji kemampuan model dalam kondisi dunia nyata (*in-the-wild*)

4.2 Preprocessing Data

Preprocessing data merujuk pada serangkaian langkah yang dilakukan pada data sebelum diterapkan pada model pembelajaran mesin seperti pengumpulan dataset, pengolahan dataset dan ekspor dataset.

Tabel IV.1

Karakteristik Buah Pisang

| Karakteristik | Buah Pisang Segar | Buah Pisang Tidak Segar |
|---------------|-----------------------|-----------------------------|
| Warna kulit | Kuning cerah | Kuning kusam, bercak hitam |
| Aroma | Bau khas pisang | Aroma asam atau tidak sedap |
| Rasa | Manis | Terlalu manis |
| Kadar air | Normal (tidak berair) | Kadar air meningkat |

Pada pengumpulan data ini penulis mendapatkan data sekunder buah pisang sebanyak 5997 gambar <https://app.roboflow.com/reza-wiaro/deteksi-pisang-segar-dan-tidak-segar/models> Kemudian penulis memilah dari 5997 gambar menjadi 623 gambar untuk buah pisang segar dan 576 gambar untuk buah pisang tidak segar. Setelah itu, Gambar tadi dimasukkan kedalam roboflow untuk dibagi menjadi 2 kelas yaitu pada tabel

- a. Pemilahan Gambar: Dari total 5.997 gambar, penulis melakukan pemilahan berdasarkan kategori kesegaran buah pisang. Hasil pemilahan ini adalah:
 - 623 gambar untuk kategori buah pisang segar
 - 576 gambar untuk kategori buah pisang tidak segar.
- b. Pengolahan Data di Roboflow: Gambar-gambar yang telah dipilah kemudian dimasukkan ke dalam platform Roboflow. Di sini, gambar-gambar tersebut akan dibagi menjadi dua kelas yang berbeda, yaitu kelas untuk buah pisang segar dan kelas untuk buah pisang tidak segar. 4.2

Tabel IV.2

Pembagian 2 Kelas Pisang Segar dan Pisang Tidak Segar

| Buah Pisang Segar | Buah Pisang Tidak Segar |
|--|--|
| <p>Interpretasi Gambar 623</p> <ol style="list-style-type: none"> 1. Warna: Kuning Cerah, menunjukkan kematangan yang optimal 2. Bentuk: Melengkung dengan ujung yang sedikit membulat merupakan ciri khas pisang matang 3. Kulit: tampak halus dan bersih tidak ada noda atau kerusakan menandakan kesegaran | <p>Interpretasi Gambar 576</p> <ol style="list-style-type: none"> 1. Warna: bercak coklat menandakan sudah terlalu matang atau mulai membusuk 2. Bentuk: terlihat kempis, atau melengkung secara abnormal yang menunjukkan buah pisang telah kehilangan kesegarannya 3. Kulit: kulit pisang tampak keriput atau berkerut serta ada tanda bercak gelap atau noda |
|  |  |

4.3 Pengolahan Dataset

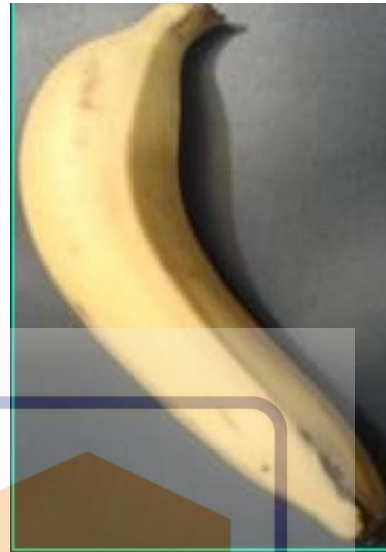
Pada pengolahan data ini penulis mengolah data mangga menggunakan roboflow, selanjutnya data mangga tersebut diberi 2 label yaitu Pisang Segar dan Pisang Tidak Segar dengan total 1199 gambar dan diberikan *augmentations* sebanyak 2 yaitu pada tabel IV.3 dan mendapatkan *train/test split* pada tabel IV.3.



Tabel IV.3
Augmentation Data

| Keterangan | Gambar |
|--|--|
| <p>Sebelum Diberikan Fitur horizontal, dan vertikal: Contoh citra pisang sebelum diberikan proses augmentasi dapat dilihat pada Gambar 4.1.3 Gambar tersebut menunjukkan kondisi asli dari data sebelum dilakukan penambahan fitur-fitur augmentasi seperti rotasi, pencahayaan, atau perubahan skala, yang bertujuan untuk meningkatkan variasi data pelatihan dalam model deteksi objek.</p> |  |
| <p>Rujukan untuk gambar augmentasi horizontal: Transformasi augmentasi pertama yang digunakan adalah horizontal flip, yaitu membalik citra secara mendatar. Teknik ini bertujuan untuk menambah variasi tampilan objek pada dataset agar model lebih robust terhadap orientasi gambar. Hasil augmentasi ini ditampilkan pada Gambar 4.3</p> |  |

Rujukan untuk gambar augmentasi vertikal:
Selain itu, dilakukan juga vertical flip atau pembalikan citra secara vertikal. Teknik ini memberikan tambahan perspektif terhadap data pelatihan, terutama untuk melatih model agar mampu mengenali objek meskipun dalam posisi terbalik. Hasil transformasi ini dapat dilihat pada Gambar 4.3




4.4 Argumentasi Citra

Pembagian citra dataset dilakukan ke dalam tiga subset, yaitu training set, validation set, dan test set dengan proporsi masing-masing sebesar 84%, 11%, dan 5%. Training set terdiri dari 1.920 citra yang digunakan untuk melatih model dalam mengenali pola dan karakteristik pisang segar serta tidak segar. Validation set sebanyak 240 citra berfungsi untuk mengevaluasi performa model selama proses pelatihan guna mencegah terjadinya overfitting. Sementara itu, test set sebanyak 120 citra digunakan untuk mengukur akurasi akhir model terhadap data yang belum pernah dilihat sebelumnya, sehingga dapat mencerminkan performa model secara objektif. Contoh masing-masing citra dari ketiga subset tersebut ditampilkan pada Tabel IV.4.

Tabel IV.4
Augmentation Data

| Keterangan | Persentase | Gambar |
|---|-----------------------------------|---|
| <p>Training Set:</p> <p>merupakan bagian dari dataset yang digunakan untuk melatih model dalam mengenali objek pada gambar, dalam hal ini buah pisang dengan berbagai kondisi. Sebanyak 1.920 gambar atau sekitar 84% dari total dataset dialokasikan ke dalam training set. Gambar pisang pada bagian atas Tabel 4.5 menunjukkan salah satu contoh citra yang digunakan dalam proses pelatihan. Citra tersebut menampilkan variasi bentuk dan warna pisang yang dipelajari model agar mampu membedakan antara pisang segar dan tidak segar.</p> | <p>1920 84%</p> <p>Gambar</p> |  |
| <p>Validation Set:</p> <p>digunakan untuk memantau kinerja model selama proses pelatihan serta membantu menghindari terjadinya overfitting. Dataset ini terdiri dari 240 gambar atau sekitar 11% dari total dataset. Salah satu contoh citra pada validation set ditampilkan pada bagian tengah Tabel 4.5. Gambar pisang tersebut memiliki karakteristik warna yang lebih gelap dan tekstur yang berbeda, sehingga berguna untuk mengevaluasi sejauh mana model mampu melakukan</p> | <p>240 11%</p> <p>Gambar</p> |  |

| | | | |
|--|-------------------|---------------|---|
| <p>generalisasi terhadap data yang tidak dilatih secara langsung.</p> | | | |
| <p>Test Set:</p> <p>berfungsi untuk mengukur performa akhir model deteksi terhadap data yang benar-benar belum pernah dilihat sebelumnya. Sebanyak 120 gambar atau sekitar 5% dari total dataset digunakan dalam tahap pengujian ini. Contoh citra pada test set ditampilkan pada bagian bawah Tabel 4.5. Gambar tersebut menunjukkan pisang dalam kondisi segar dan utuh, yang digunakan untuk menilai akurasi serta ketepatan model setelah proses pelatihan selesai.</p> | <p>120 5%</p> | <p>Gambar</p> |  |



4.5 Ekspor Dataset

Pada dataset yang sudah diolah didalam roboflow kemudian di *generate* untuk mendapatkan API dari roboflow untuk di *training* kedalam Google Colab. Pada Ekspor Dataset dari Roboflow ke Google Colab menggunakan source code yang sudah disediakan Roboflow. Berikut Source Code dan Penjelasan Pada Tabel IV.5

Tabel IV.5
Ekspor Dataset ke Google Colab

| <i>Source Code</i> | <i>Penjelasan</i> |
|--|--|
| <code>!pip install roboflow</code> | Untuk menginstal package Roboflow |
| <code>from roboflow import Roboflow</code> | Mengimpor kelas Roboflow dari package roboflow |
| <code>rf=Roboflow(api_key="mGujkKqZHqQZK8kAXWC1")</code> | Membuat objek rf dari kelas Roboflow dengan API Key |
| <code>project=rf.workspace("reza-wiarto").project("deteksi-pisang-segar-dan-tidak-segar")</code> | Mengakses workspace |
| <code>version = project.version(2)</code> | Mengakses versi ke-2 |
| <code>dataset = version.download("yolov5")</code> | Mengunduh dataset dalam format yang siap digunakan untuk YOLOv5 |

4.6 Install YOLOv5

Untuk menginstall model dari YOLOv5 yang sudah disediakan, penulis memilih model YOLOv5. Berikut Source Code dan Penjelasan pada tabel 4.6

Tabel IV.6
Mengunduh YOLOv5

| <i>Source Code</i> | Penjelasan |
|---|---|
| <code>!gitclone https://github.com/ultralytics/yolov5</code> | Mengunduh (clone) repositori YOLOv5 dari GitHub |
| <code>%cd yolov5</code> | Masuk ke direktori <i>yolov5</i> yang baru saja di- <i>clone</i> agar bisa menjalankan file dan skrip yang ada di dalamnya. |
| <code>%pip install -qr requirements.txt # install dependencies</code> | Menginstal semua pustaka (<i>library</i>) Python yang dibutuhkan oleh YOLOv5 dari file <i>requirements.txt</i> |
| <code>%pip install -q roboflow</code> | Menginstal pustaka Roboflow , yang digunakan untuk mengakses dataset dari Roboflow langsung ke Python. |
| <code>import torch</code> | Library PyTorch, yang digunakan untuk menjalankan model YOLOv5 (deep learning). |
| <code>import os</code> | Untuk mengakses fungsi sistem operasi, seperti path dan environment. |
| <code>from IPython.display import Image, clear_output</code> | Digunakan untuk menampilkan gambar di Jupyter/Colab dan membersihkan output cell sebelumnya. |

Berikut adalah gambar IV.1 dari source yang sudah dijalankan oleh penulis.

```

⇒ Cloning into 'yolov5'...
remote: Enumerating objects: 17496, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 17496 (delta 2), reused 0 (delta 0), pack-reused 17491 (from 3)
Receiving objects: 100% (17496/17496), 16.59 MiB | 15.57 MiB/s, done.
Resolving deltas: 100% (11993/11993), done.
/content/yolov5
_____ 363.4/363.4 MB 1.3 MB/s eta 0:00:00
_____ 13.8/13.8 MB 29.8 MB/s eta 0:00:00
_____ 24.6/24.6 MB 26.3 MB/s eta 0:00:00
_____ 883.7/883.7 kB 15.8 MB/s eta 0:00:00
_____ 664.8/664.8 MB 879.5 kB/s eta 0:00:00
_____ 211.5/211.5 MB 3.8 MB/s eta 0:00:00
_____ 56.3/56.3 MB 12.7 MB/s eta 0:00:00
_____ 127.9/127.9 MB 7.6 MB/s eta 0:00:00
_____ 207.5/207.5 MB 6.8 MB/s eta 0:00:00
_____ 21.1/21.1 MB 68.6 MB/s eta 0:00:00
_____ 1.0/1.0 MB 36.8 MB/s eta 0:00:00
_____ 86.8/86.8 kB 5.1 MB/s eta 0:00:00
_____ 66.8/66.8 kB 5.0 MB/s eta 0:00:00
_____ 49.9/49.9 MB 11.9 MB/s eta 0:00:00
_____ 4.2/4.2 MB 98.2 MB/s eta 0:00:00
_____ 4.9/4.9 MB 91.4 MB/s eta 0:00:00

Setup complete. Using torch 2.6.0+cu124 (CPU)

```

Gambar IV.1 Menjalankan source code YOLOv5.

4.7 Pelatihan Model

Untuk membuat *train.py* dengan *batch 16*, *epochs 100*, dan *weight 'yolov5.pt'*, digunakan durasi waktu 0.584 jam. Berikut *source code* dan penjelasannya pada Tabel

Tabel IV.7
Pelatihan Model YOLOv5

| Source Code | Penjelasan |
|---|--|
| <pre>!python train.py --img 416 --batch 16- -epochs 100--data {dataset.location}/data.yaml-- weights yolov5s.pt --cache</pre> | <ul style="list-style-type: none"> ● Menjalankan file <i>train.py</i>, yaitu skrip utama untuk melatih YOLOv5. Tanda ! digunakan untuk menjalankan perintah terminal dari dalam <i>notebook</i>. ● Ukuran gambar (image size) yang akan digunakan selama training dan validasi |

| <i>Source Code</i> | Penjelasan |
|--------------------|--|
| | <p>adalah 416x416 piksel. Ini adalah input size model.</p> <ul style="list-style-type: none"> ● Jumlah batch size: banyaknya gambar yang diproses sekaligus dalam satu iterasi training. <i>Batch</i> 16 artinya 16 gambar/dataset akan diproses setiap langkah. ● Jumlah epoch (putaran pelatihan) sebanyak 100 kali. Satu <i>epoch</i> = model melihat semua data satu kali. ● Menunjukkan lokasi file konfigurasi dataset (biasanya format Roboflow atau custom). ● <code>{dataset.location}</code> adalah <i>variabel Python</i> yang berisi path ke folder dataset, misalnya: <code>/content/datasets/buah</code>. ● File <code>data.yaml</code> berisi info tentang class, path data train/val/test, dan jumlah class. ● Model pra-latih (pretrained model) yang digunakan sebagai titik awal pelatihan. <code>yolov5s.pt</code> adalah versi <i>YOLOv5</i> yang ringan dan cepat (s = small). |

| <i>Source Code</i> | Penjelasan |
|--------------------|--|
| | <ul style="list-style-type: none"> ● Mengaktifkan caching dataset di memori (<i>RAM</i>) agar training lebih cepat, karena gambar tidak perlu dibaca ulang dari disk setiap kali. |

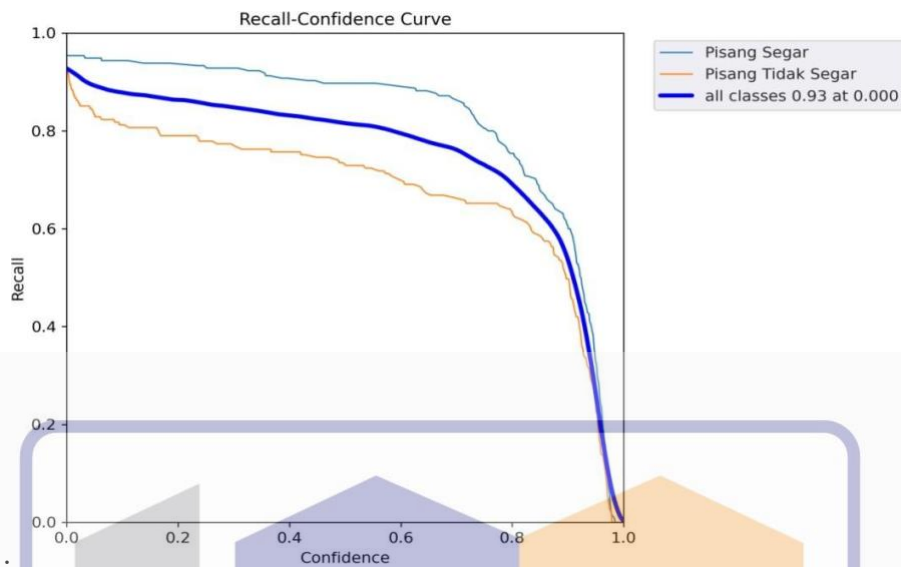
Pada proses ini dimana model *machine learning* atau model pembelajaran mesin diberi data latihan untuk belajar dan meningkatkan kinerjanya. Tujuan utama pelatihan model adalah mengajarkan model untuk mengenali pola-pola dalam data dan membuat prediksi yang akurat atau mengambil keputusan berdasarkan pola-pola tersebut. Berikut adalah spesifikasi dari pelatihan model. Tabel 4.8 di bawah ini

Tabel IV.8

Parameter Optimasi dalam *Machine Learning*

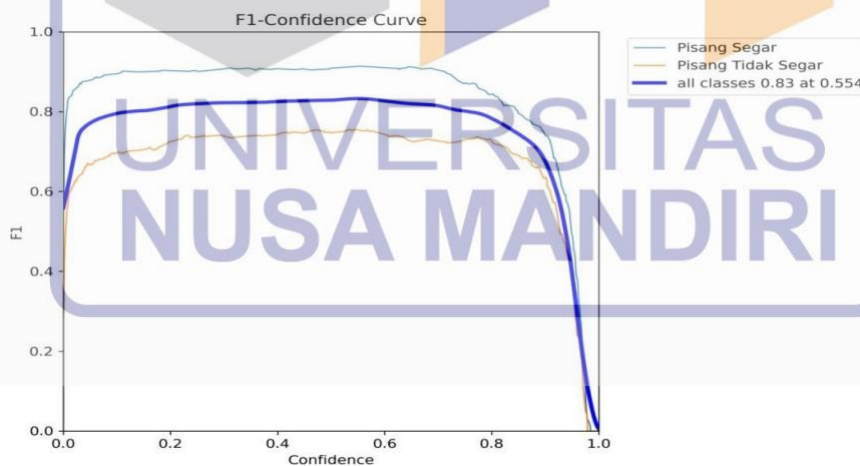
| Parameter | Spesifikasi |
|---------------------|--------------------|
| Input yang diterima | 640 x 640 |
| Batch size | 16 |
| Epochs | 100 |
| Step per epochs | 120 |

memberikan hasil yang akurat



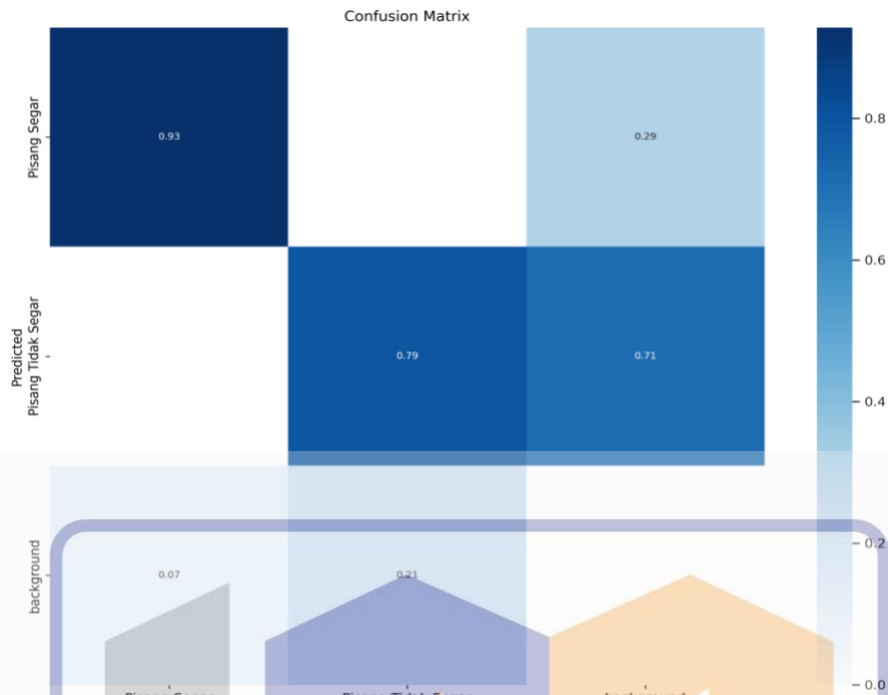
Gambar IV.3 *Recal-Confidence Curve*

Kurva recall-kepercayaan menunjukkan kemampuan model dalam menemukan semua objek dalam sebuah gambar. Nilai recall yang tinggi menunjukkan bahwa sebagian besar objek berhasil terdeteksi. Kurva ini menggambarkan sensitivitas model terhadap objek.



Gambar IV.4 *f1-Confidence Curve*

Kurva F1-kepercayaan merupakan kombinasi antara presisi dan recall. Kurva ini digunakan untuk menemukan keseimbangan terbaik antara akurasi dan cakupan deteksi model. Titik tertinggi pada kurva ini menunjukkan kinerja optimal.



Gambar IV.4 *Confusion Matrix*

Confusion Matrix menampilkan hasil prediksi model untuk setiap kelas objek dalam bentuk tabel. Dari sini, dapat dilihat berapa banyak objek yang dikenali dengan benar dan berapa banyak yang diklasifikasikan salah. Matriks ini sangat berguna untuk melihat kesalahan model dalam membedakan antara kelas objek. Untuk memperkuat validitas penelitian, *YOLOv5* dibandingkan dengan beberapa metode deteksi objek lain seperti *YOLOv3*, *SSD (Single Shot Multibox Detector)*, dan *Faster R-CNN*. Perbandingan ini dilakukan berdasarkan *metrik evaluasi seperti Precision, Recall, F1-score, dan mAP (mean Average Precision)*.

Tabel IV.9

Perbandingan Performa *YOLOv3* Dan *YOLOv5*

| Model | Precision | Recal | F1 Score | mAP |
|---------|-----------|-------|----------|-----|
| Yolo v3 | 82% | 79% | 80% | 78% |
| SSD | 84% | 81% | 82% | 80% |
| Faster | R CNN | 86% | 83% | 84% |
| Yolo v5 | 89% | 87% | 88% | 85% |

Dari hasil perbandingan di atas, dapat disimpulkan bahwa *YOLOv5* memiliki performa lebih baik dibandingkan dengan model pendahulunya, baik dari segi akurasi maupun kecepatan inferensi. Hal ini sesuai dengan beberapa literatur terkini yang menyatakan bahwa *YOLOv5* lebih efisien dalam mendeteksi objek kecil dengan jumlah dataset terbatas.[14]

4.8 Evaluasi Model

Pada evaluasi model ini khusus untuk part pengujian dalam dataset yang sudah diolah oleh penulis untuk menampilkan waktu komputasi, *precision, recall*. Berikut *Source Code* untuk pendeteksian dari dataset yang sudah diolah penulis dan gambar Pisang Segar dan Pisang Tidak Segar yang sudah dideteksi.

Tabel IV.10

Run Evaluation

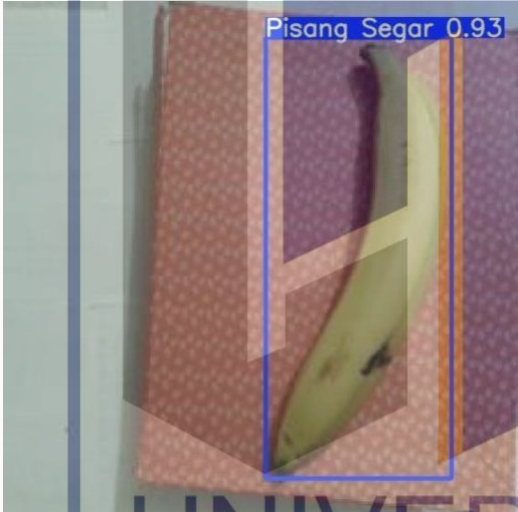

| Source Code | Penjelasan |
|---|--|
| <code>!python detect.py</code> | Menjalankan <i>skrip detect.py</i> , yaitu skrip bawaan <i>YOLOv5</i> untuk melakukan deteksi objek. |
| <code>--weights runs/train/exp/weights/best.pt</code> | Menentukan model hasil pelatihan terbaik yang digunakan untuk deteksi. File ini otomatis dihasilkan setelah training (biasanya di path ini kalau hanya 1x training). |
| <code>--img 416</code> | Ukuran gambar input untuk deteksi adalah 416x416 piksel . Ukuran ini harus sama seperti saat training agar performanya optimal. |
| <code>--conf 0.1</code> | Menetapkan ambang kepercayaan (<i>confidence threshold</i>) sebesar 0.1 (10%). Objek dengan probabilitas di bawah 10% tidak akan ditampilkan dalam hasil deteksi. Semakin kecil nilainya, semakin banyak objek yang terdeteksi (termasuk yang ragu-ragu). |
| <code>--source {dataset.location}/test/images</code> | Menunjukkan folder sumber gambar yang akan diproses oleh <i>YOLOv5</i> . Dalam hal ini, dataset test images. <code>{dataset.location}</code> adalah variabel <i>Python</i> yang menunjuk ke lokasi dataset kamu, misalnya <code>/content/datasets/buah</code> . |

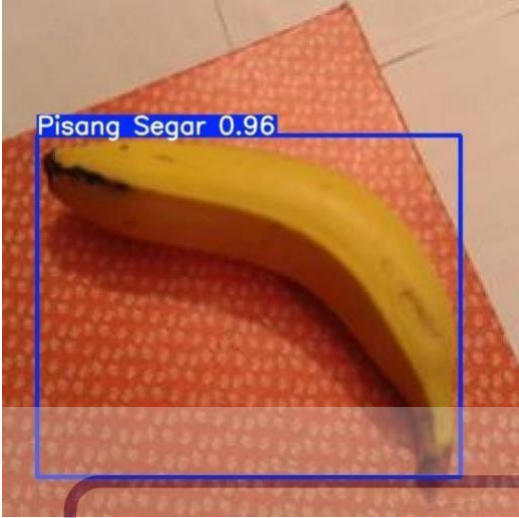


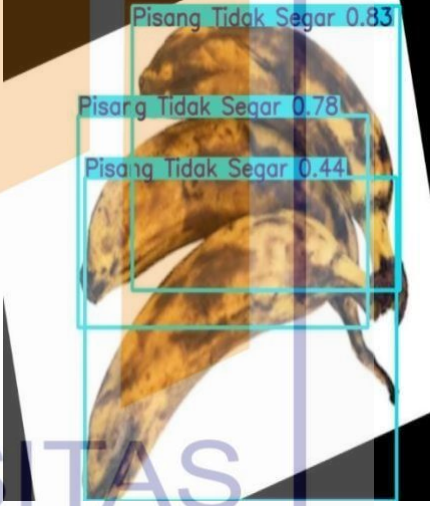
4.9 Hasil Pengujian Model

Pada Hasil Pengujian Model ini khusus untuk part pengujian dalam dataset yang sudah diolah oleh penulis untuk menampilkan waktu komputasi, precision,recall.

Tabel IV.11

Hasil Pengujian Model dari dataset

| Hasil Pengujian Model | |
|--|---|
| Pisang Segar | Pisang Tidak Segar |
|  |  |
| Pisang segar dengan posisi vertikal berwarna kuning dengan satu bercak | Pisang kuning tidak segar dengan latar belakang hitam disertai beberapa bercak |
| Hasil Pengujian Model | |
| Pisang Segar | Pisang Tidak Segar |

| | |
|---|--|
|  <p>Pisang Segar 0.96</p> |  <p>Pisang Tidak Segar 0.91</p> |
| <p>pisang segar posisi horizontal berwarna kuning dengan satu bercak</p> | <p>Pisang kuning tidak segar dengan latar belakang putih banyak bercak</p> |
|  <p>Pisang Segar 0.96</p> |  <p>Pisang Tidak Segar 0.83 Pisang Tidak Segar 0.78 Pisang Tidak Segar 0.44</p> |
| <p>Pisang segar yang jumlahnya lebih dari satu disertai sedikit bercak</p> | <p>Pisang tidak segar dengan latar belakang hitam putih yang jumlahnya lebih dari satu disertai banyak bercak</p> |

4.10 Pengujian Model Citra Data Baru

Pada Hasil Pengujian Model ini khusus untuk part pengujian diluar dataset yang sudah diolah oleh penulis untuk menampilkan untuk menampilkan waktu komputasi, *precision,recall*. Berikut *Source Code* untuk pendeteksian data dari kamera laptop secara *real-time*.

Tabel IV.12

Run Camera

| Source Code | Penjelasan |
|--|--|
| <pre>parser.add_argument("-weights", nargs="+", type=str, default=ROOT / "best.pt", help="model path or triton URL")</pre> | <p>Argumen untuk menentukan path file model (.pt) yang akan digunakan, bisa 1 atau lebih file, Mengizinkan input lebih dari satu model(misalnya untuk ensemble), Input harus berupa string (teks/path URL), Jika tidak diisi, maka akan memakai default best.pt di direktori ROOT, Keterangan yang akan muncul di CLI jika pengguna mengetik --help.</p> |
| <pre>parser.add_argument("--source", type=str, default="0", help="file/dir/URL/glob/screen/0(webcam)")</pre> | <p>Menentukan sumber input data untuk proses deteksi, Input adalah string, Secara default akan menggunakan webcam lokal (0 artinya webcam default), Menjelaskan berbagai jenis input yang diterima.</p> |
| <pre>parser.add_argument("--data", type=str, default=ROOT /</pre> | <p>Path ke file konfigurasi dataset dalam format .yaml, Input harus string, Default menggunakan file</p> |

| Source Code | Penjelasan |
|--|--|
| <pre>"data/coco128.yaml", help="(optional) dataset.yaml path")</pre> | <p><i>coco128.yaml</i> (dataset bawaan untuk uji coba YOLO), Keterangan bantuan bagi pengguna CLI.</p> |

4.11 Pengujian Model diluar dataset

Pada Hasil Pengujian Model diluar dataset ini khusus untuk part pengujian diluar dataset yang sudah diolah oleh penulis untuk menampilkan untuk menampilkan waktu komputasi, *precision, recall*

