

# UTS Applied Deep Learning : Pendeteksian Objek Menggunakan Model YOLOv8

Sayyid Abdullah

November 4, 2023

## Abstract

Penelitian ini membahas pengembangan model YOLOv8 (You Only Look Once version 8) menggunakan framework Ultralytics untuk meningkatkan kinerja deteksi objek dalam gambar. YOLOv8 merupakan varian terbaru dari keluarga YOLO yang dikenal dengan akurasi yang tinggi dan kecepatan yang memadai dalam tugas deteksi objek.

Studi ini mengawali dengan menjelaskan konsep dasar YOLOv8 dan mengenalkan framework Ultralytics yang digunakan. Model ini dikembangkan dengan melakukan penyesuaian terhadap arsitektur jaringan, pengoptimalan parameter, serta pengujian yang komprehensif untuk memastikan akurasi dan kecepatan yang maksimal.

Hasil dari penelitian ini adalah model YOLOv8 yang dapat mendeteksi objek dengan akurasi yang tinggi dan menjalankan deteksi objek dalam waktu nyata. Model ini memiliki potensi aplikasi yang luas dalam berbagai bidang, termasuk visi komputer, kendaraan otonom, pengawasan keamanan, dan banyak lagi.

## 1 Pendahuluan

Deteksi objek adalah salah satu tugas fundamental dalam bidang visi komputer yang memiliki aplikasi luas dalam berbagai domain, termasuk kendaraan otonom, pengawasan keamanan, analisis medis, dan banyak lagi. Salah satu pendekatan yang paling populer dalam deteksi objek adalah model YOLO (You Only Look Once). YOLO merupakan singkatan dari "You Only Look Once," yang menggambarkan filosofi dasar model ini, yaitu melakukan deteksi objek dalam satu tahap pengolahan gambar, berbeda dengan metode yang memerlukan beberapa tahap pengolahan.

YOLO bekerja dengan memahami masalah deteksi objek sebagai masalah regresi yang mengidentifikasi koordinat batas-batas (bounding box) objek serta mengklasifikasikan objek tersebut, semuanya dilakukan dalam satu perhitungan. Ini memungkinkan YOLO untuk memberikan hasil deteksi dengan kecepatan tinggi, membuatnya cocok untuk aplikasi real-time. YOLO menggunakan jaringan saraf tiruan konvolusi (CNN) yang mendalam untuk mengambil fitur gambar dan menghasilkan prediksi yang akurat.

Sejak diperkenalkan pertama kali, model YOLO telah mengalami beberapa perkembangan dan variasi. Beberapa jenis model YOLO yang dikenal antara lain:

<b>Classification</b>	<b>Detection</b>	<b>Segmentation</b>	<b>Kind</b>
yolov8n-cls.pt	yolov8n.pt	yolov8n-seg.pt	Nano
yolov8s-cls.pt	yolov8s.pt	yolov8s-seg.pt	Small
yolov8m-cls.pt	yolov8m.pt	yolov8m-seg.pt	Medium
yolov8l-cls.pt	yolov8l.pt	yolov8l-seg.pt	Large
yolov8x-cls.pt	yolov8x.pt	yolov8x-seg.pt	Huge

Figure 1: Tipe dari YOLOv8

1. YOLOv1: Versi pertama YOLO yang memperkenalkan konsep dasar deteksi objek dalam satu tahap.
2. YOLOv2 (YOLO9000): Mengenalkan perbaikan signifikan pada akurasi dengan menggunakan teknik anchor box.
3. YOLOv3: Meningkatkan akurasi lebih lanjut dengan tiga skala deteksi berbeda.
4. YOLOv4: Lebih lanjut meningkatkan akurasi dan kecepatan dengan teknik-teknik canggih seperti pengoptimalan model dan augmentasi data.
5. YOLOv5: Mengoptimalkan arsitektur jaringan untuk kecepatan tinggi dan akurasi yang tetap baik.
6. YOLOv8: Mengoptimalkan arsitektur jaringan untuk kecepatan lebih tinggi dari versi sebelumnya dan akurasi yang lebih baik.

Penelitian ini akan fokus pada pengembangan model YOLOv8 menggunakan framework Ultralytics, yang bertujuan untuk meningkatkan kinerja deteksi objek dalam gambar dengan memanfaatkan teknologi terbaru dalam bidang visi komputer. Dengan menggabungkan konsep dasar YOLO dan keunggulan Ultralytics, penelitian ini diharapkan dapat menghasilkan model yang memadukan kecepatan dan akurasi dalam tugas deteksi objek, dengan aplikasi yang luas di berbagai sektor.

## 2 Metode

Untuk mengimplementasikan model YOLOv8 menggunakan ultralytics memerlukan beberapa langkah sebagai berikut:

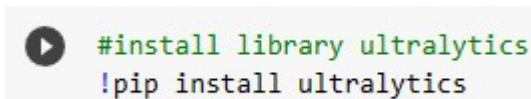
1. Mempersiapkan dataset.

2. Install Ultralytics.
3. Konfigurasi model.
4. Pelatihan model.
5. Pengujian model.
6. Inferensi model yang dilatih.
7. Menyimpan model yang dihasilkan.

## 3 Hasil dan Pembahasan

Pada bagian ini akan dibahas mengenai model YOLOv8 yang digunakan untuk mendeteksi gambar. Pada bagian ini akan dibagi menjadi 2 bagian yaitu model YOLOv8 dengan menggunakan dataset coco128 dari ultralytics dan model YOLOv8 dengan menggunakan dataset dari Roboflow.

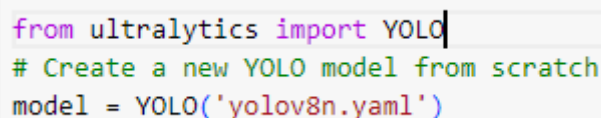
### 3.1 Model YOLOv8 menggunakan dataset yang diberikan di ultralytics



```
#install library ultralytics
!pip install ultralytics
```

Figure 2: Install library Ultralytics

Source code pada bagian ini dapat dilihat pada ultralytics. Banyak bahasa pemrograman yang bisa dipakai untuk membuat model YOLO menggunakan Ultralytics. Bahasa pemrograman yang dipakai pada penelitian ini adalah Python. Notebook yang digunakan adalah google collab. Langkah pertamanya adalah dengan menginstall library Ultralytics seperti pada Gambar 2. Selanjutnya membuat model YOLO baru sesuai dengan kebutuhannya seperti pada Gambar 3. Pada gambar tersebut model yang dibuat adalah model untuk mendeteksi suatu gambar, untuk jenis-jenisnya bisa dilihat pada Gambar 1. 'yolov8n.yaml' adalah file konfigurasi YAML yang digunakan untuk menentukan arsitektur dan parameter model. Selanjutnya adalah dengan melatih model yang dibuat tadi dengan menggunakan dataset yang disediakan oleh ultralytics yaitu coco128.yaml seperti pada Gambar 4. Dataset coco128 ini ada banyak gambar yang masing-masing sudah memiliki label. Dataset ini berupa gambar orang, bus, mobil, sepeda dll yang bisa dilihat pada dokumen dataset ultralytics.



```
from ultralytics import YOLO
# Create a new YOLO model from scratch
model = YOLO('yolov8n.yaml')
```

Figure 3: Membuat model YOLO baru

```
# Train the model using the 'coco128.yaml' dataset for 3 epochs
results = model.train(data='coco128.yaml', epochs=3)
```

Figure 4: Pelatihan model

```
▶ # Evaluate the model's performance on the validation set
results = model.val()
```

Figure 5: Validasi model

Selanjutnya adalah validasi model seperti pada Gambar 5. Setelah validasi dan dirasa model sudah cukup baik, langkah selanjutnya adalah inferensi model. Perintah yang digunakan untuk inferensi model seperti pada Gambar 6. Output pada gambar 6 berupa 1 bus, 4 orang, dan 1 rambu berhenti. Hasil pendeteksiannya bisa ditampilkan seperti pada gambar 7.

### 3.2 Model YOLOv8 dengan menggunakan dataset dari Roboflow

Selanjutnya bagaimana misalkan kita punya dataset sendiri? jika kita ingin menggunakan dataset sendiri, langkah pertama adalah kita harus menyiapkan dataset yang akan kita pakai. Persiapannya dimulai dari mengumpulkan gambar, memberikan label, dan men-split datanya. Pemberian label bisa menggunakan roboflow atau tools sejenis yang bisa digunakan untuk memberikan label pada data yang sudah dikumpulkan. Untuk menggunakan roboflow bisa dengan mengunjungi link ini roboflow. Dataset yang kali ini dipakai adalah dataset yang diperoleh dari roboflow yaitu dataset yang berisi tomat, pasta, susu, dan kentang. Untuk langkah-langkahnya kurang lebih sama seperti bagian sebelumnya.

Dataset yang akan digunakan di load terlebih dahulu dengan menggunakan perintah seperti pada gambar 8. Code tersebut diperoleh dari roboflow untuk ngeloda datasetnya ke google colab. Setelah itu datasetnya akan muncul pada session storage google colab. Kemudian, buat model baru seperti Gambar 3 dan dilanjutkan melatih model tersebut dengan dataset baru seperti pada Gambar 9. setelah itu inferensi modelnya dengan menggunakan gambar tomat yang diperoleh dari google. Hasilnya seperti pada Gambar 10. Jika model yang diperoleh sudah dirasa cukup baik, alangkah baiknya model tersebut disimpan menjadi sebuah file sehingga jika kita ingin menggunakan model tersebut tinggal load modelnya tanpa harus fitting ulang dari awal. Banyak cara untuk menyimpan model yang diperoleh seperti dengan menggunakan onnx, joblib, dan lain-lain. Contohnya

```
# Perform object detection on an image using the model
results = model('/content/bus.jpg')
```

```
image 1/1 /content/bus.jpg: 640x480 4 persons, 1 bus, 1 stop sign, 213.3ms
Speed: 5.2ms preprocess, 213.3ms inference, 1.5ms postprocess per image at shape (1, 3, 640, 480)
```

Figure 6: Inferensi Model

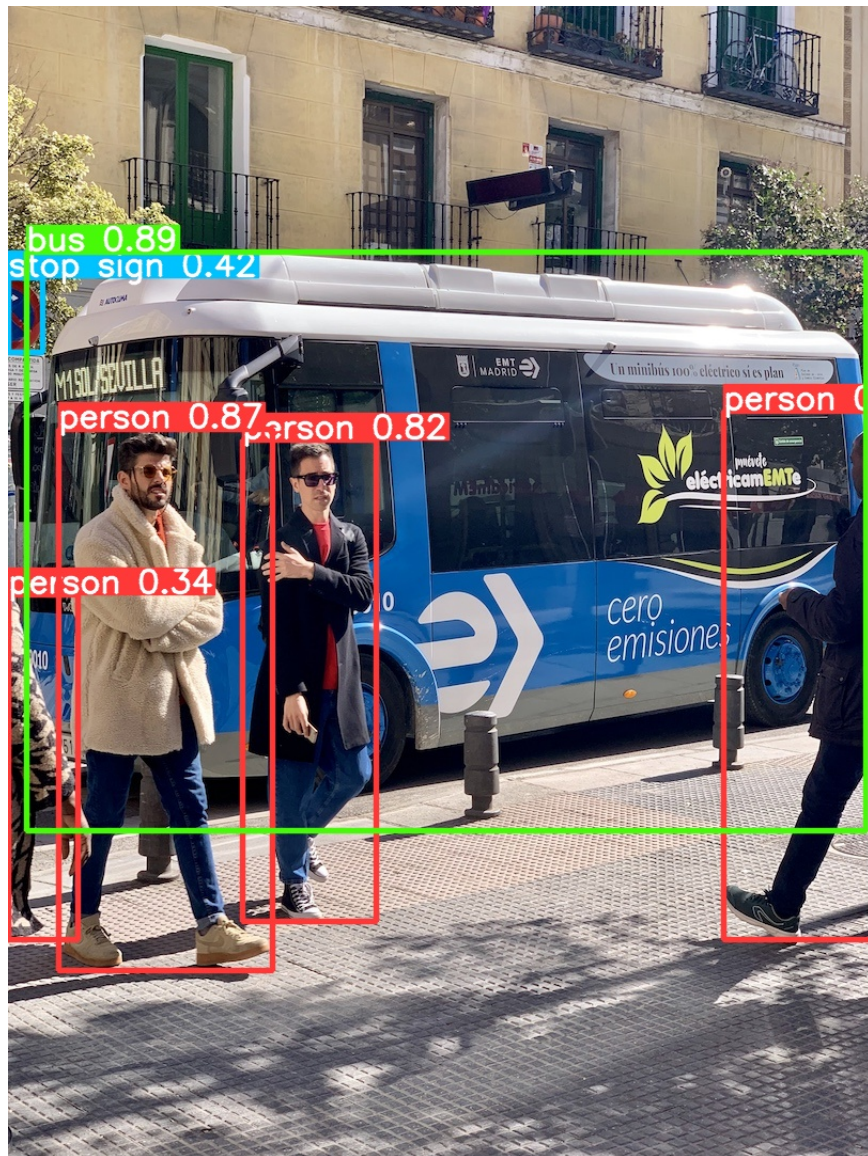


Figure 7: Hasil pendeteksian

seperti pada Gambar 11.

Dari percobaan-percobaan yang dilakukan, diperoleh bahwa epoch yang digunakan jangan terlalu sedikit. Jika epoch yang digunakan terlalu sedikit, maka model yang diperoleh kurang baik. Pada contoh yang menggunakan dataset sendiri, awalnya menggunakan 3 dan 5 epoch, namun hasilnya adalah no detection, Sehingga dinaikkan epochnya menjadi 100 dan bisa mendeteksi 3 tomat seperti pada Gambar 10.



```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="nBWWcXSg2urfLx1WVJfL")
project = rf.workspace("hochschule-munchen").project("groceryclassification")
dataset = project.version(5).download("yolov8")
```

Figure 8: Caption

```
# Train the model using the 'data.yaml' dataset for 100 epochs
results = model.train(data='/content/GroceryClassification-5/data.yaml', epochs=100)
```

Figure 9: Pelatihan model dengan dataset dari roboflow

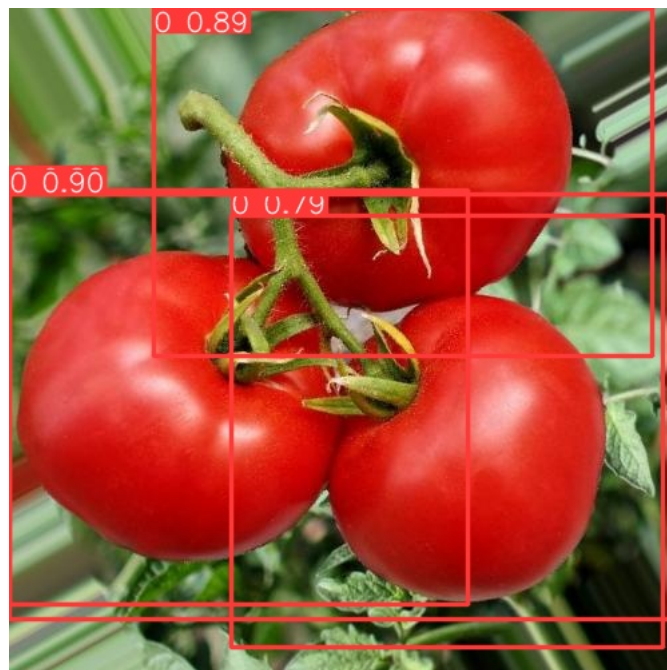


Figure 10: Caption

```
# Export the model to ONNX format
success = model.export(format='onnx')
```

Figure 11: Menyimpan Model