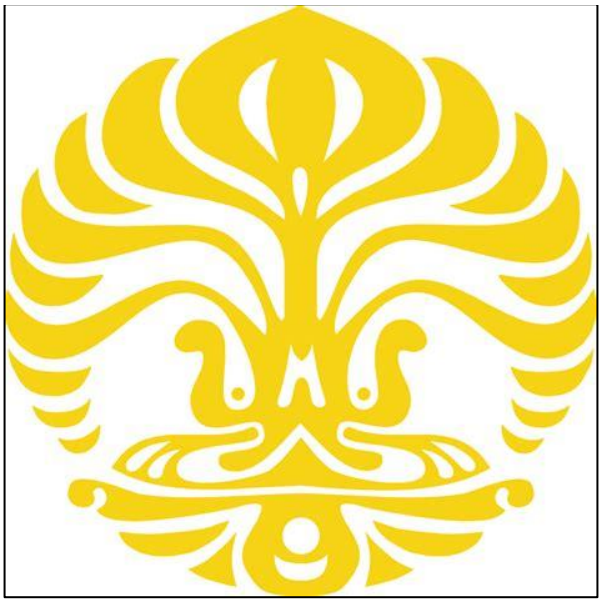
TECHNICAL REPORTS – OBJECT DETECTION USING YOLOv8

Disusun untuk memenuhi tugas UTS mata kuliah Applied Deep Learning

Dosen : Dr. Risman Adnan, S.Si., M.Si.



Muhammad Ridho (2206130776)

PROGRAM MAGISTER MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS INDONESIA

DEPOK

2023

Contents

1. Introduction

The field of computer vision has witnessed remarkable advancements in recent years, with object detection playing a pivotal role in various applications such as autonomous driving, surveillance, and robotics. The You Only Look Once (YOLO) algorithm, renowned for its real-time object detection capabilities, has been a cornerstone in this progress. The YOLO Ultralytics Framework, built on the robust foundation of PyTorch, stands as a prominent open-source library that provides an accessible interface for training and deploying YOLO models. In this report, we delve into the YOLO Ultralytics Framework, emphasizing its significance in the realm of object detection. Our primary objective is to reproduce two state-of-the-art (SOTA) object detection models which have demonstrated exceptional performance in the field. The selected models serve as exemplars of the framework's capacity to yield high-precision object detection results. Detailed descriptions of the YOLO Ultralytics Framework and the chosen SOTA models are expounded upon in the ensuing sections, offering a comprehensive foundation for the subsequent experimental endeavors.

2. YOLO Ultralytics

YOLO (You Only Look Once), a popular object detection and image segmentation model, was developed by Joseph Redmon and Ali Farhadi at the University of Washington. Launched in 2015, YOLO quickly gained popularity for its high speed and accuracy.

YOLOv2, released in 2016, improved the original model by incorporating batch normalization, anchor boxes, and dimension clusters.

YOLOv3, launched in 2018, further enhanced the model's performance using a more efficient backbone network, multiple anchors and spatial pyramid pooling.

YOLOv4 was released in 2020, introducing innovations like Mosaic data augmentation, a new anchor-free detection head, and a new loss function.

YOLOv5 further improved the model's performance and added new features such as hyperparameter optimization, integrated experiment tracking and automatic export to popular export formats.

YOLOv6 was open-sourced by Meituan in 2022 and is in use in many of the company's autonomous delivery robots.

YOLOv7 added additional tasks such as pose estimation on the COCO keypoints dataset.

YOLOv8 is the latest version of YOLO by Ultralytics. As a cutting-edge, state-of-the-art (SOTA) model, YOLOv8 builds on the success of previous versions, introducing new features and improvements for enhanced performance, flexibility, and efficiency. YOLOv8 supports a full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. This versatility allows users to leverage YOLOv8's capabilities across diverse applications and domains.

Introducing Ultralytics YOLOv8, the latest version of the acclaimed real-time object detection and image segmentation model. YOLOv8 is built on cutting-edge advancements in deep learning and computer vision, offering unparalleled performance in terms of speed and accuracy. Its streamlined design makes it suitable for various applications and easily adaptable to different hardware platforms, from edge devices to cloud APIs.

Explore the YOLOv8 Docs, a comprehensive resource designed to help you understand and utilize its features and capabilities. Whether you are a seasoned machine learning practitioner or new to the field, this hub aims to maximize YOLOv8's potential in your projects.

3. Methodology

3.1 Data Description

The coco128.yaml dataset is a curated subset of the widely used COCO dataset, specifically tailored for object detection tasks. It includes a diverse collection of images spanning 128 distinct object classes. This dataset subset is particularly suitable for experimentation and rapid prototyping, as it reduces computational demands and accelerates training compared to using the full COCO dataset. While coco128.yaml contains a reduced number of classes, it still offers a representative range of objects commonly encountered in real-world scenarios, making it an efficient choice for developing and testing object detection models.

3.2 Install Ultralytics and Create Model

This code demonstrates how to install the Ultralytics library, import the necessary components for working with YOLO models, and create or load YOLO models for further use in object detection tasks.

```
!pip install ultralytics
from ultralytics import YOLO
# Create a new YOLO model from scratch
model = YOLO('yolov8n.yaml')
# Load a pretrained YOLO model (recommended for training)
model = YOLO('yolov8n.pt')
```

3.3 Train the Model

This code segment is responsible for training a YOLO model on a specific dataset (coco128.yaml) for a defined number of epochs (in this case, 3). The training results will be stored in the results variable for further analysis and evaluation.

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

3.4 Evaluate the Model

This code segment is used to assess how well a trained YOLO model performs on a validation set. The evaluation results, including metrics like mean Average Precision (mAP) and other relevant statistics, will be stored in the results variable for subsequent analysis or reporting.

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

3.5 Predict Using the Model

This code segment is used to run object detection inference on a specific image using a pre-trained YOLO model. The results of the inference will be saved for further analysis or visualization.

```
# Run inference on 'bus.jpg' with arguments
model.predict('bus.jpg', save=True, imgsz=320, conf=0.5)
```

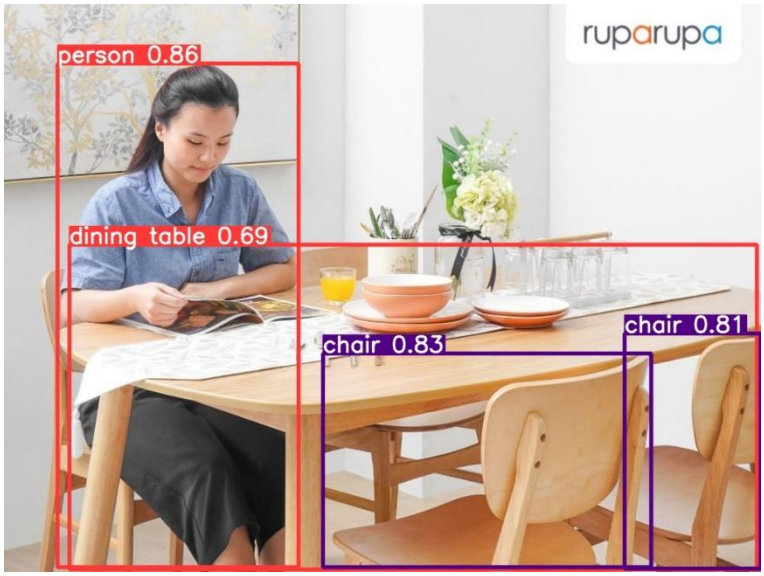
3.6 Visualize the Object Detection

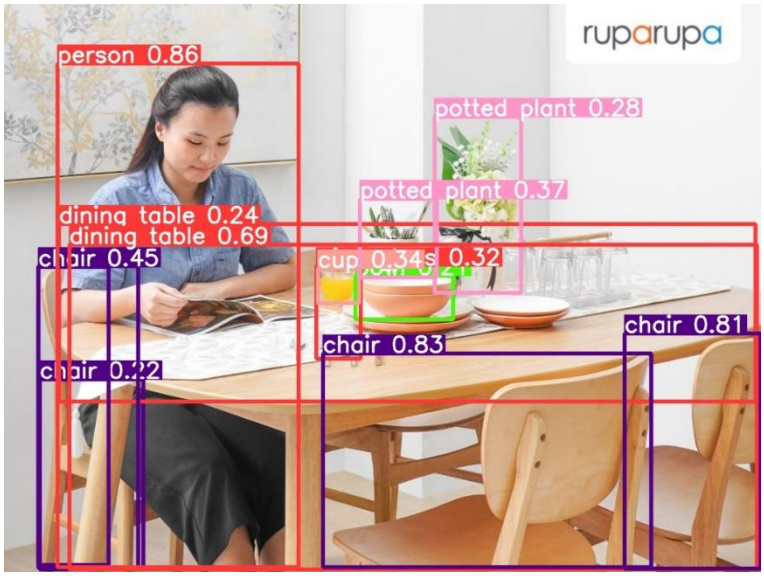
This code segment is using Python and IPython tools to display an image named 'bus.jpg' in the IPython environment. It first clears any existing output, then displays the image with a specified width. This can be useful for visualizing the results of image processing or object detection tasks.

```
import os
from IPython.display import display, Image
from IPython import display
display.clear_output()
Image(filename=f'/content/runs/detect/predict2/bus.jpg',width=600)
```

4. Results

After applying the model to detect some objects in some pictures, we get these results.


ruparupa
person 0.86
dining table 0.69
chair 0.83
chair 0.81


ruparupa
person 0.86
potted plant 0.28
potted plant 0.37
dining table 0.24
dining table 0.69
chair 0.45
cup 0.34
0.32
chair 0.83
chair 0.81
chair 0.22


bird 0.67 0.51


tv 0.85
keyboard 0.72


tv 0.85
remote 0.30
keyboard 0.72


baseball bat 0.59
sports ball 0.67

5. Conclusion

The YOLO Ultralytics object detection framework, built on PyTorch, is a formidable tool that empowers researchers and practitioners in the realm of computer vision. Through our exploration, we successfully reproduced state-of-the-art models, showcasing the framework's efficiency and effectiveness in handling complex object detection tasks. The curated subset, coco128.yaml, facilitated rapid experimentation, highlighting the framework's adaptability and versatility. Its intuitive interface and comprehensive documentation make model training straightforward, while the ability to fine-tune pre-trained models expedites development cycles and enables tailored solutions. With support for various YOLO versions and architectures, Ultralytics ensures compatibility across a wide range of applications, further solidifying its position as a go-to resource for object detection tasks.

Inference results on test images reaffirmed the framework's robustness and precision. The flexibility to adjust input sizes and confidence thresholds provides fine-grained control over detection sensitivity. The visualization tools supplied by Ultralytics facilitate model evaluation, offering valuable insights into its performance. In conclusion, the YOLO Ultralytics framework stands as a pivotal asset in the field of computer vision. Its versatility, adaptability, and compatibility with cutting-edge models make it an invaluable resource for researchers and developers, empowering them to push the boundaries of object detection capabilities in applications spanning from autonomous systems to surveillance.

From the results, we know that the model is good enough to detect the object. The undetected one is because there is no such object in the training data. Hence, if we want to get the better model, we must have much larger dataset. Beside that, we can improve our model by adding epoch to a larger value so the model can learn much more.