Ministry of Higher Education
Pyramids Higher Institute (PHI) for Engineering and Technology, 6th of October
Electronics and Communications Engineering Department

وزارة التعليم العالي
معهـد الأهرامـات العـالي للهنـدســـة والتكنولوجيا بالسادس من أكتوبر
قسم هندسة الالكترونيات والاتصالات

Design of Autonomous Car with Advanced Driver Assistance System (ADAS)

Students Names:
Abobakr Mostafa Fathy
Ali Osama Ebaid
Eslam Ahmed Ibrahim
Madlen Nady Samir
Mohamed Hany Hassan
Mahmoud Mohamed Elbhrawy
Moaaz Mohamed Mohamed
Youssef Ehab Mohamed

Supervisors:
Dr. Gamal El-Sheikh
Dr. Walaa Omar

You Only Look Once (YOLO) Object detection models

Introduction:

YOLO (You Only Look Once) is a state-of-the-art real-time object detection system that revolutionized the way objects are detected in images and videos. Unlike traditional object detection models that use sliding windows or region proposals to localize objects, YOLO frames object detection as a single regression problem. This approach enables YOLO to predict both bounding boxes and class probabilities directly from full images in a single pass, making it extremely fast and efficient.

YOLO divides the input image into a grid, and for each grid cell, it predicts a predefined number of bounding boxes and the probabilities that each of these boxes contains a specific object. It achieves high accuracy and speed by leveraging deep convolutional neural networks (CNNs) to perform feature extraction and detection simultaneously.

History:

The YOLO series of models has become famous in the computer vision world. YOLO's fame is attributable to its considerable accuracy while maintaining a small model size. YOLO models can be trained on a single GPU, which makes it accessible to a wide range of developers. Machine learning practitioners can deploy it for low cost on edge hardware or in the cloud.

YOLO has been nurtured by the computer vision community since its first launch in 2015 by Joseph Redmond. In the early days (versions 1-4), YOLO was maintained in C code in a custom deep learning framework written by Redmond called Darknet.

YOLOv8 author, Glenn Jocher at Ultralytics, shadowed the YOLOv3 repo in PyTorch (a deep learning framework from Facebook). As the training in the shadow repo got better, Ultralytics eventually launched its own model: YOLOv5.

YOLOv5 quickly became the world's SOTA repo given its flexible Pythonic structure. This structure allowed the community to invent new modeling improvements and quickly share them across repository with similar PyTorch methods.

Along with strong model fundamentals, the YOLOv5 maintainers have been committed to supporting a healthy software ecosystem around the model. They actively fix issues and push the capabilities of the repository as the community demands.

In the last two years, various models branched off of the YOLOv5 PyTorch repository, including Scaled-YOLOv4, YOLOR, and YOLOv7. Other models emerged around the world out of their own PyTorch based implementations, such as YOLOX and YOLOv6. Along the way, each YOLO model has brought new SOTA techniques that continue to push the model's accuracy and efficiency.
Over the last six months, Ultralytics worked on researching the newest SOTA version of YOLO, YOLOv8. YOLOv8 was launched on January 10th, 2023.

YOLO Versions:

The YOLO journey started in 2016. Joseph Redmon and Santosh Divvala introduced a new way of detecting objects in images by processing them in one go, making it both fast and accurate. Here’s how the series has evolved:

- **YOLOv2 (2017):** Known as YOLO9000, this version improved accuracy and performance by adding features like batch normalisation and the Darknet-19 backbone.
- **YOLOv3 (2018):** This version introduced multi-scale predictions, allowing it to detect objects of different sizes more effectively.
- **YOLOv4 (2020):** YOLOv4 introduced new techniques, like CSPDarknet53, making it faster and better at handling complex tasks.
- **YOLOv5 (2020):** Although not officially part of the YOLO series, YOLOv5, developed by Ultralytics, became famous for its ease of use and better training features.
- **YOLOv6 (2022):** Released by Meituan, YOLOv6 made further improvements in speed and efficiency.
- **YOLOv7 (2022):** This version, created by Chien-Yao Wang, brought faster and more accurate object detection through an improved backbone.
- **YOLOv8 (2023):** YOLOv8, created by Glenn Jocher and Ultralytics, is the most advanced version yet. It uses cutting-edge deep learning techniques that make it ideal for tasks like autonomous driving and advanced security systems.

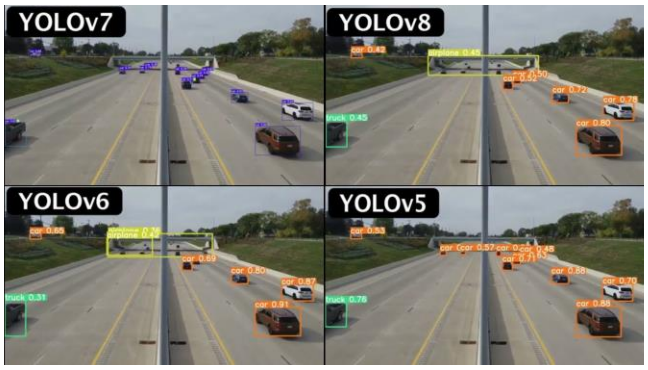

Fig.2. YOLO Versions

Object detection models

Object detection models vary in strengths and are categorized into two types: one-stage models and two-stage models.

One-stage:

- YOLO
- SSD

Two-stages:

- RCNN
- Fact RCNN
- RFCN
- Mask RCNN

After researching and comparing various options, we found that the YOLO (You Only Look Once) model stands out as the perfect choice for our needs. YOLO is known for its balance of speed and accuracy, making it ideal for real-time applications. While other models like R-CNN, Faster R-CNN, SSD (Single Shot Multibox Detector), and RetinaNet offer great performance, YOLO's efficiency and ability to predict bounding boxes and class labels in a single pass through the network make it the most suitable for our project.

Here are a few reasons why YOLO is often considered the best choice for object detection in many scenarios:

- Speed:
  - YOLO is extremely fast because it predicts the bounding boxes and class labels for the entire image in a single forward pass through the neural network. This makes it suitable for real-time applications like video analysis and autonomous driving.
- Accuracy:
  - YOLO's performance is generally good when it comes to detecting objects in images. Recent versions, like YOLOv4 and YOLOv5, have improved accuracy metrics and object localization.
- Single-stage detection:
  - YOLO is a single-stage object detection algorithm, meaning it combines both the region proposal and classification steps into a single neural network process. This makes it simpler and more efficient compared to two-stage detectors like R-CNN and its variants (Fast R-CNN, Faster R-CNN).
- Flexibility:
  - YOLO can generalize across a wide range of object categories, from small objects to large ones, and works well in real-time systems or scenarios with limited computing power (e.g., edge devices).

- **Versions and Support:**
    - Since its inception, YOLO has undergone several updates, with newer versions like YOLOv4, YOLOv5, and YOLOv8 offering even better performance, accuracy, and ease of use.
    - The YOLO ecosystem has wide community support, with pre-trained models and implementations in various frameworks (PyTorch, TensorFlow, etc.).

YOLO Algorithm:

The YOLO algorithm begins by taking an input image, which is resized to a fixed size (for example, 416x416 or 608x608 pixels). This resizing step is essential for standardizing the image format so that it can be processed uniformly by the network. The pixel values are normalized, which speeds up the computation and ensures more stable training. This preprocessing step is relatively straightforward but critical for ensuring that the system can handle images of varying sizes.

Once the image has been prepared, YOLO divides it into a grid of cells. For example, a 416x416 image might be divided into a 13x13 grid, with each cell responsible for detecting objects whose

centers fall within that particular cell. This grid structure is fundamental to YOLO's efficiency, as each grid cell will predict a number of bounding boxes along with confidence scores for each of those boxes.

Each grid cell predicts a fixed number of bounding boxes (typically 2 to 5). For each bounding box, the network predicts the coordinates of the box, which include the x and y coordinates for the center of the box, as well as its width and height. These coordinates are normalized relative to the grid cell and the entire image. In addition to the bounding box coordinates, YOLO also predicts a confidence score for each box, indicating how likely it is that the box contains an object and how accurate the bounding box is. This score represents the confidence that the bounding box encloses an object and is calculated as a product of the probability of an object being present and the accuracy of the predicted box.

After predicting the bounding boxes, YOLO proceeds to predict class probabilities for each detected object. Each grid cell outputs class probabilities for all possible object categories, such as "car," "person," or "dog," depending on the dataset being used. These probabilities indicate how likely it is that the object in the bounding box belongs to each class. Importantly, the final score for each object in a bounding box is the product of the predicted class probability and the confidence score for the bounding box itself. Once the bounding boxes and class probabilities have been predicted, YOLO uses a metric called Intersection over Union (IoU) to determine the overlap between predicted bounding boxes and the ground-truth boxes (the actual location of the objects). The IoU is calculated by dividing the area of overlap between the predicted box and the ground-truth box by the total area of the two boxes combined. If the IoU exceeds a certain threshold (commonly set to 0.5), the box is considered a valid detection. IoU helps YOLO assess the accuracy of its predictions, particularly when multiple boxes overlap or when boxes are too large or too small.

To further refine the predictions, YOLO applies a technique known as Non-Maximum Suppression (NMS). This step is crucial for removing redundant bounding boxes. For example, if multiple bounding boxes are predicted around the same object, NMS ensures that only the box with the highest confidence score is kept, while the others are suppressed.

This step is vital for avoiding duplicate detections of the same object and ensuring that the algorithm outputs only the most accurate bounding box for each object.

Despite its remarkable speed, YOLO has some limitations. For example, it sometimes struggles with detecting small objects, particularly when they occupy a small portion of the grid cell. Additionally, because each grid cell predicts a fixed number of bounding boxes, YOLO may not perform as well when there are multiple overlapping objects in the same grid cell. Nevertheless, YOLO's advantages—its ability to process images in real time and its end-to-end architecture—make it a powerful tool for applications such as autonomous driving, video surveillance, and robotics, where speed is critical.

Pseudocode for YOLO Algorithm:

```
1. Initialize the YOLO model
2. Preprocess the input image
3. Pass the preprocessed image through the YOLO network
4. Get the model output
5. Post-process the predictions
6. Output the final detections
7. (Optional) Repeat steps 2-6 for a video stream or multiple images
```

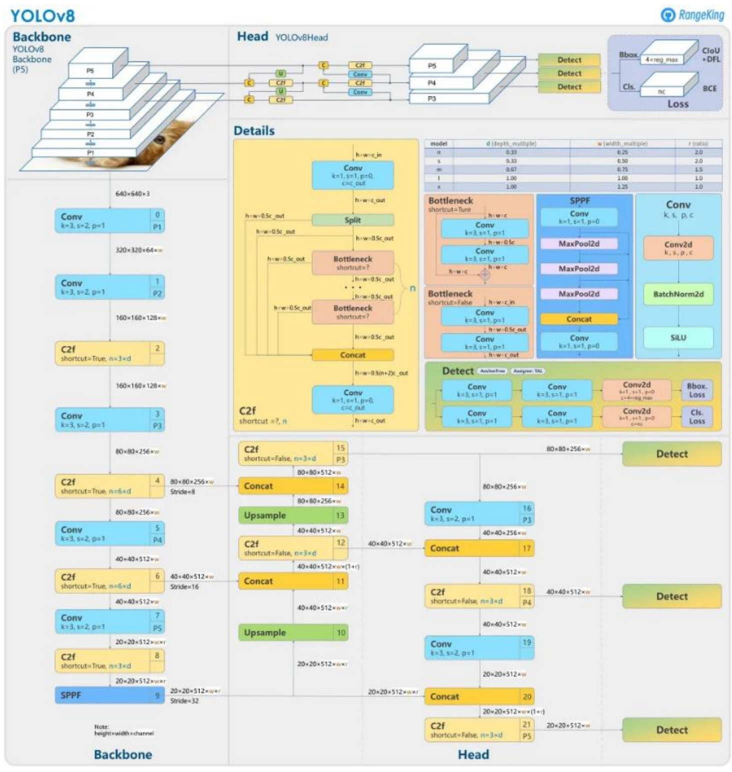

Fig.1. YOLOv8 Architecture

The overall process can be summarized as follows:

1. **Input Image:** YOLO receives the input image for object detection.
2. **Grid Division:** The image is divided into a grid of cells.
3. **Bounding Box Prediction:** YOLO predicts multiple bounding boxes for each grid cell.
4. **Class Probability Prediction:** Class probabilities are assigned to each bounding box based on the objects within.
5. **IoU Evaluation:** Intersection over Union (IoU) is used to evaluate the accuracy of each predicted bounding box.
6. **Non-Maximum Suppression (NMS):** NMS is applied to remove duplicate or overlapping boxes, keeping only the most accurate ones.
7. **Final Output:** A set of bounding boxes, each with a class label and confidence score, is returned as the detected objects.

Advantages of YOLOv8:

YOLOv8 is the latest version in the YOLO object detection series, and it comes with several advantages over previous versions and other object detection models:

1. **Accuracy and Speed**: It offers higher precision and faster real-time detection with better feature extraction and optimized neural structures.
2. **Advanced Architecture**: YOLOv8 uses a more efficient backbone and improved neck/head design for better multi-scale object detection.
3. **Better Generalization**: It performs well in diverse real-world conditions like varying lighting and backgrounds.
4. **Model Flexibility**: Provides different model sizes (nano to large) to balance speed and accuracy based on deployment needs.
5. **Data Augmentation**: Uses modern techniques like Mosaic and Mixup for better training and optimized anchors for accurate bounding boxes.
6. **Ease of Use**: YOLOv8 integrates well with deep learning frameworks like PyTorch and is beginner-friendly while supporting advanced customizations.
7. **Edge Optimization**: Smaller models are highly efficient for mobile and edge devices.
8. **Multi-task Support**: YOLOv8 handles object detection, segmentation, and instance segmentation, making it versatile.

Disadvantages of YOLOv8:

1. **Speed vs .Accuracy Trade-off**: Smaller models prioritize speed over accuracy, and real-time performance can be challenging on low-power devices.
2. **Resource-Intensive**: Training and running larger YOLOv8 models require powerful hardware like GPUs, limiting accessibility.
3. **Difficulty with Small Objects**: YOLOv8 struggles to detect very small objects due to its grid-based architecture.
4. **Challenges with Dense Object Detection**: It may miss objects in crowded or overlapping scenarios.
5. **Anchor Dependency**: Predefined anchor boxes may require complex tuning for variable object sizes.
6. **Limited Performance in Specialized Tasks**: YOLOv8 may underperform in tasks like segmentation or non-standard detection compared to specialized models.
7. **Inaccuracy with Irregular Objects**: Rectangular bounding boxes can lead to inaccuracies with irregularly shaped objects.
8. **Not Ideal for Fine-Grained Detection**: It may not handle subtle object distinctions or defects well.
9. **Risk of Overfitting**: Small datasets can lead to overfitting without proper augmentation.
10. **Customization Limitations**: Significant modifications are needed for highly specialized applications.