University of Tehran

College of Engineering

School of Electrical and Computer Engineering (ECE)

School of Mechanical Engineering (ME)

Mechatronics & Robotics

MiniProject 4

Teaching Assistants:
Reihaneh Yourdkhani

Deadline: 21 June 2024 (19 Tir), 23:59

List of Problems

In this project, we are going to use deep learning in the realm of object detection to propose a method to automatically pack custom catering packages. As these packages are really useful and common, this project is a real case scenario of developing a automated system in industry and factories.

In the ever-expanding realm of artificial intelligence (AI), object detection stands out as a foundational pillar with transformative implications. At its core, object detection goes beyond mere image classification—it identifies and locates multiple objects within an image, assigning them precise bounding boxes and categories. This capability has propelled advances in numerous applications, from self-driving cars to smart surveillance systems. As we delve deeper into its nuances and potential, it is crucial to understand the definition and relevance of object detection within the broader scope of AI.

The evolution of object detection has been a captivating journey, reflecting the broader shifts in the world of AI. In its early stages, object detection relied heavily on hand-engineered features—meticulously crafted algorithms that were designed to pick up specific attributes of objects. While innovative for their time, these detectors were often constrained by their rigidity and lack of adaptability to diverse scenarios. However, with the advent of deep learning, the landscape dramatically changed. Neural networks, especially convolutional neural networks (CNNs), introduced a paradigm shift, to learn and extract features from raw data automatically. As we traverse this narrative, we will explore the transformation from the humble beginnings of hand-tuned feature extraction methods to the groundbreaking deep learning techniques that dominate the field today.

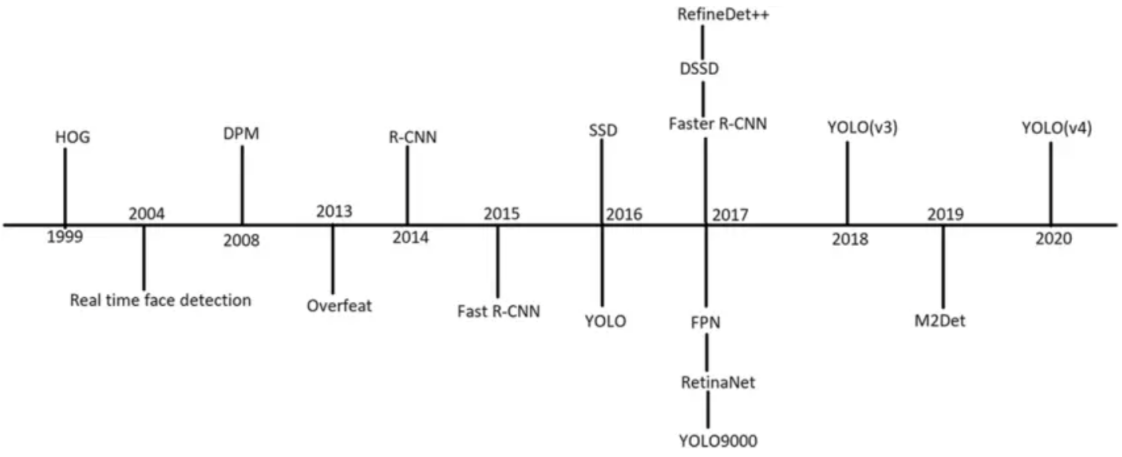


Figure 1: Timeline of object detection throughout time

The state-of-the-art object detection methods can be categorized into two main types: one-stage methods and two stage-methods:

1. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet.

2. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN and Cascade R-CNN.

In this project we are going to take a look at object detection application on a real dataset with practical outputs.

Project description

With 7.88 billion people on earth in 2023 and their in- creasing population, their demand for food has also increased. Global food demand is expected to increase by 35% to 56% between 2010 and 2050. With that in mind, the COVID- 19 pandemic brought much attention to hygiene in the food industry. According to scientists, cleaning, sanitation, good hygienic practices, and active packaging are needed from farm to fork. As a result, 1-increasing demand for food and 2-the importance of hygiene in the food industry made researchers and industries endeavor hand in hand to find new and innovative approaches to make robots facilitate this task in a sanitized environment faster than humans resorting to robotic mechanical systems.

This automated system requires pick-and-place operation. But before initiating pick-and-place operations, object detection is paramount. Leveraging advancements in deep learning, particularly models like RCNN, FastRCNN, and FasterRCNN, has become instrumental. For this project, deep learning methods are employed to integrate detection and segmentation models with conventional approaches. Creating a custom dataset tailored to the robot environment is a prerequisite for developing a specialized object detection model.

As explained in the introduction we have two object detection methods that either prioritize inference speed or detection accuracy. In this project we are going to learn about YOLO(You only look once) which belongs to the first category.

YOLO (You only look once)

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor.Figure 2 shows a timeline showcasing YOLO's development in recent years.

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown in figure 3.

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to

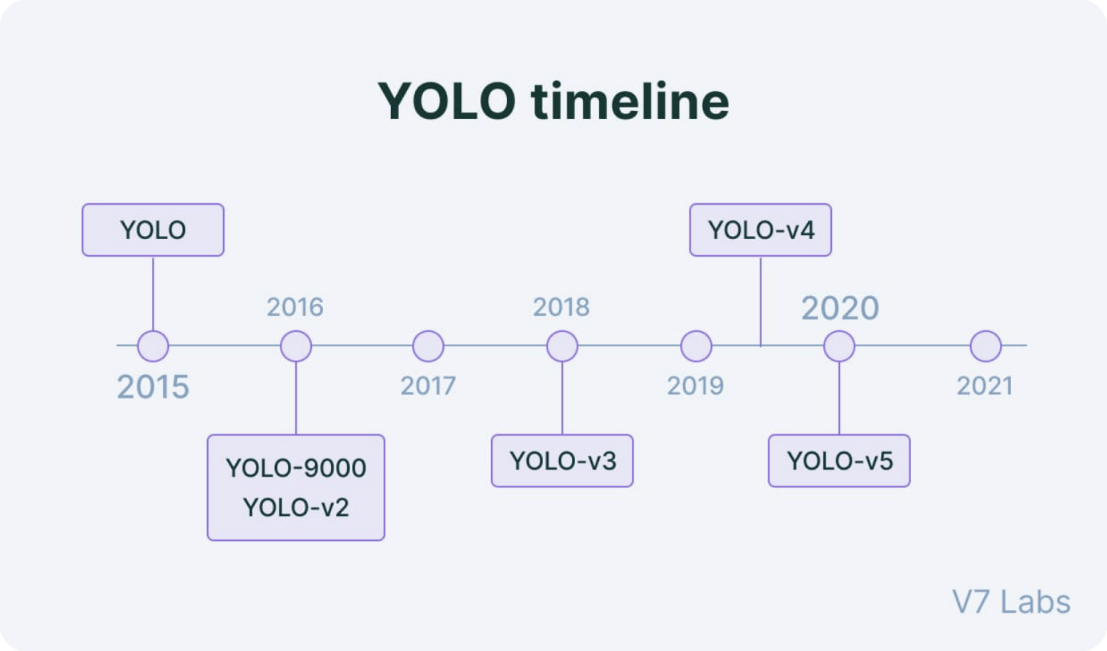
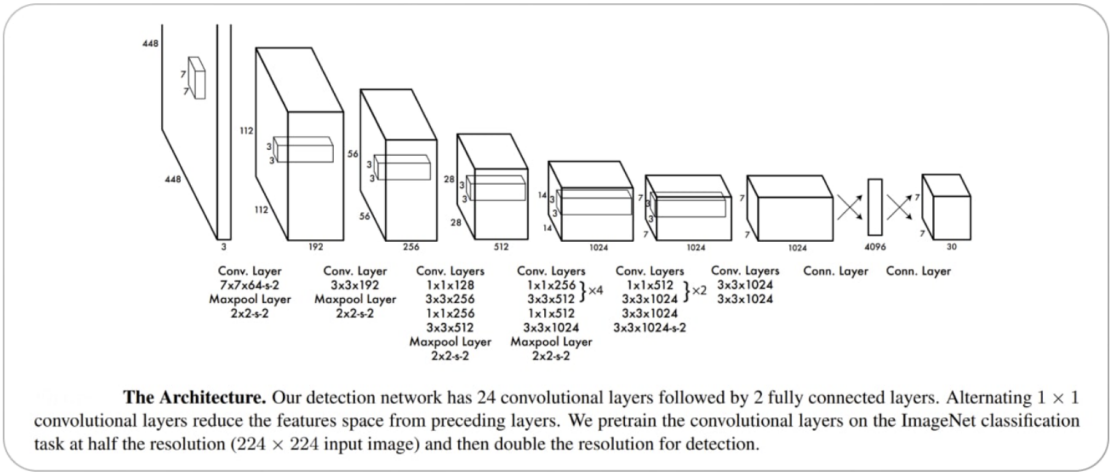


Figure 2: Timeline of YOLO



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 3: YOLO architecture

perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

Problem 1: (YOLO different versions) 10 points

YOLO has published different versions throughout time, desrcibe one difference in each version in comparison with the older version. (Getting help from ChatGPT is valid for this question.)

Problem 2: (mAP score) 10 points

YOLO uses mAP(mean average percision) score as an evaluation metrics. Describe how this score is calculated. (Getting help from ChatGPT is valid for this question.)

Problem 3: (Project dataset) 20 points

In order to gather comprehensive data for catering pack- ages, a diverse set of objects crucial to these packages was considered. As these objects, mainly manufactured products, are influenced by brand characteristics like shape and color, a specific dataset, Catering Packages Objects (CPO), was created for this study. Tailored to the robotic environment, this dataset factors in conditions such as lighting and shadows. Additionally, two augmentation steps were applied to this dataset. This dataset has been preserved and labeled using Roboflow platform for transparency and reproducibility.

Dataset Sources: The CPO dataset is divided into two segments, enhancing its robustness. Approximately 90% of the dataset comprises images captured by the authors of this paper, ensuring a controlled and consistent environment. The remaining 10% is sourced from online shopping platforms through web scraping, introducing diversity and resilience to varied environments. This subset includes images obtained directly from manufacturers, characterized by their cleanliness and clarity, as well as images contributed by consumers in the comment sections—typically unclean and noisy, reflecting real-world scenarios. This dual sourcing strategy fortifies the dataset's adaptability and relevance across different contexts. The CPO dataset images exhibit two distinct groups based on capture devices: 1) iPhone camera images focusing on the test environment background, and 2) 720p webcam images taken under the robot, considering background, lighting (DPR LEDs and room lighting), and shadows. The latter set, pivotal for the experimental stage, introduces unique challenges explored in subsequent sections.

This is a link to this dataset.

1. Open the link to the dataset in roboflow platform. Analyze this dataset regarding its even distribution throughout different classes. Report the number of images and instances of each class in this dataset.

2. Why do we use augmentation steps in the process of developing a dataset?

3. What are the augmentation steps used in this dataset? How are they useful for this project?

(Help: You can use health check and other options in roboflow platform to answer this question. Although it is recommended to go through the options of this platform once, since it is an important dataset storage tool.)

Problem 4: (Object detection) 40 points

In this part of the project we are going to go through the actual object detection process step by step. Do this question in a python notebook. (Since this project needs GPU, it is recommended to use google colab)

1. Import the dataset using the link to the dataset that has been provided in the last section. You can export a link to the dataset regarding to which versions of YOLO want to use in the next sections.

2. Install and import YOLO in your notebook. Although its up to use which version you want to use, it is highly recommended to use versions above v5.

3. Go through the training process of training this dataset on the first version of YOLO you chose. After you're done training, you get a report of your training process. Report this table and discuss it. (Since you have limited GPU sources 50 epochs of training is enough for this project)

4. Report the training time needed per epoch for this version.

5. You have been provided with a folder of test images. Upload this folder to your colab and test the trained model on these folders. Talk about the results. Why does the model get some outputs wrong?

Problem 5: (Object segmenation) 20 points

By applying Object Detection models, we will only be able to build a bounding box corresponding to each class in the image. But it will not tell anything about the shape of the object as the bounding boxes are either rectangular or square in shape. Image segmentation will create pixel-wise masks for each object hence it will be useful to understand granular details about the object.

Figure 4: An example of segmentation

Fast Segment Anything (FastSAM) is an image segmentation model developed by Meta AI. This model can identify the precise location of either specific objects in an image or every object in an image. SAM utilizes user prompts, such as bounding boxes, text, points, or segment- everything inputs. In this problem you will use SAM to segment images using the bounding boxes outputs of problem 4.

1. Import FastSAM to your colab.
2. You have tested your fine-tuned model in previous question on the test folder. Using FastSAM which is a zero-shot segmentation model and the bounding box outputs from YOLO for these images, visualize the output of SAM on these pictures.

Problem 6: (Grasp point generation) 20 points(Bonus)

Upon reaching this stage, it becomes imperative to compute certain geometrical parameters before proceeding to the grasping phase. These calculations encompass determining the precise coordinates, orientation, and width of each object. These matters are elaborated in details in the remaining parts of this section.

The FastSAM algorithm produces masks that highlight each object in an image. These masks can be converted into outlines, allowing to fit a rotated rectangle around each object. This rectangle is defined by four corners, numbered 0 to 3 in a clockwise direction, with the first corner having the smallest x-value. The algorithm also provides an "angle" output, representing the tilt of the rectangle. This angle is calculated by measuring the rotation from the x-axis. In simpler terms, it helps understand how each object is oriented in the image.

You can use MinAreaRect function in OpenCV with some modifications in order to obtain a rotated rectangle for each object and calculate the rotation angle of each object.

Using the rectangle obtained in the previous step, the center of the rotated rectangle can be easily found. By considering the object's boundary polygon and a line that crosses the center of the rotated rectangle at its rotation angle, two points that are likely to be the best places to grasp the object can be identified.

This line represents the shorter side of the object, crossing its center. The point where this line intersects with the object's boundary polygon gives the solution that is needed. In this part you are expected to get a result like this:

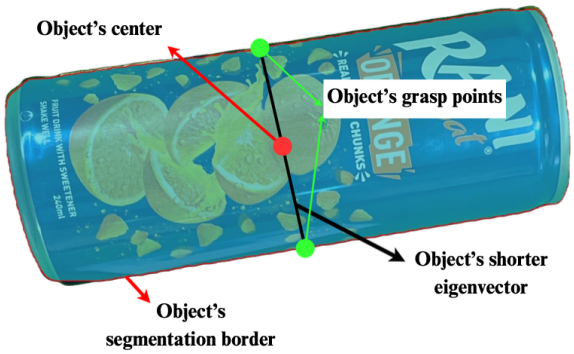


Figure 5: An example of grasp point generation

Please note that is not necessary to visualize everything on your output picture and only visualizing the grasp points are enough.

1. Using MinAreaRect and after converting FastSam segmentations into contours, obtain the rotated rectangle around the object. You should convert FastSAM's segmentations to binary masks for this step of the question. Then with getting help from the commands provided at the end of the question, you can output your desired results.

2. Using data from MinAreaRect obtain the rotation angle of the object. For this part you would have to understand what is the angle output of MinAreaRect. You can use this link

3. By using the explained algorithm above, obtain the grasp points using the rotation angle from the previous question, the center of rotated rectangle and FastSam's segmentation.

4. Visualize the output of your algorithm testing it on one picture.

Useful commands while working with OpenCV:

```
(center(x, y), (width, height), angle of rotation) = cv2.minAreaRect(points)

cv2.line(image, start point, end point, color, thickness)


from shapely import box, LineString, normalize, Polygon
line = LineString([(0, 0), (2, 2)])
intersection(line, LineString([(1, 1), (3, 3)]))
```

Homework Guidelines and Instructions

- The deadline for sending this exercise will be until the end of Tuesday, March 12.
- This time cannot be extended and you can use time grace if needed.
- The implementation must be in Python or Matlab programming language and your codes must be executable and uploaded along with the report.
- This exercise is done by one person.
- If any similarity is observed in the work report or implementation codes, this will be considered as fraud for the parties.
- Using ready-made codes without mentioning the source and without changing them will constitute cheating and your practice score will be considered zero.
- If you do not follow the format of the work report, you will not be awarded the grade of the report.
- Handwritten exercise delivery are acceptable as long as they are legible and clean.
- Please note that for the code exercise you MUST upload your code files and write a report for it as well. If one is missing, your answer won't be graded.
- A large part of your grade is related to the work report and problem solving process.
- Please upload the report, code file and other required attachments in the following format in the system: `HW1_[Lastname]_[StudentNumber].zip`
  For example, the: `HW1_Ezati_12345678.zip`
- If you have questions or doubts, you can contact the assistants through the following e-mail with the subject HW1-Mechatronics. Stay in touch educationally:
  - `r.yourdkhani@gmail.com` (Reihaneh Yourdkhani)
- Be happy and healthy