



BÁO CÁO CUỐI KỲ

MÔ HÌNH

PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

NHÓM 13:

1. Dương Thị Kiều Doan
2. Trang Trà My
3. Trần Thị Mỹ

Giảng viên: ThS Trần Văn Lộc



TT. Mỹ





MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

BỐ CỤC



1. GIỚI THIỆU ĐỀ TÀI

2. TỔNG QUAN LÝ THUYẾT

3. TRIỂN KHAI MÔ HÌNH

4. ỨNG DỤNG THỰC TẾ

5. HƯỚNG PHÁT TRIỂN



Hộp giấy

Tái chế

Không
tái chế

Có những người người bỏ vào đúng thùng



NHÓM 13

MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

1. GIỚI THIỆU VỀ MÔ HÌNH

Đặt vấn đề:



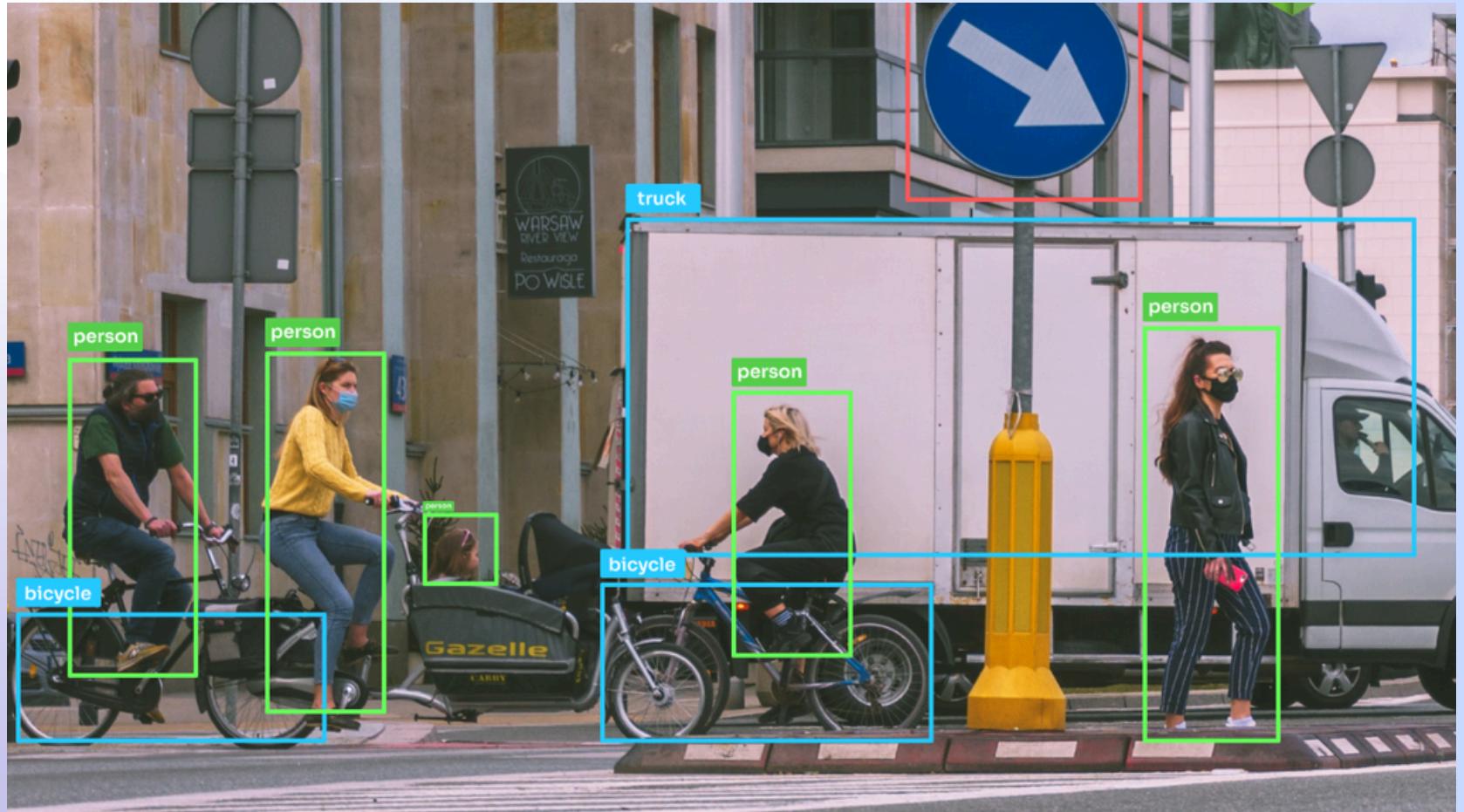
➔ **Mục tiêu của đề tài:** Phát triển mô hình AI giúp tự động phân loại rác tái chế chính xác và hiệu quả góp phần bảo vệ môi trường



MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

2.

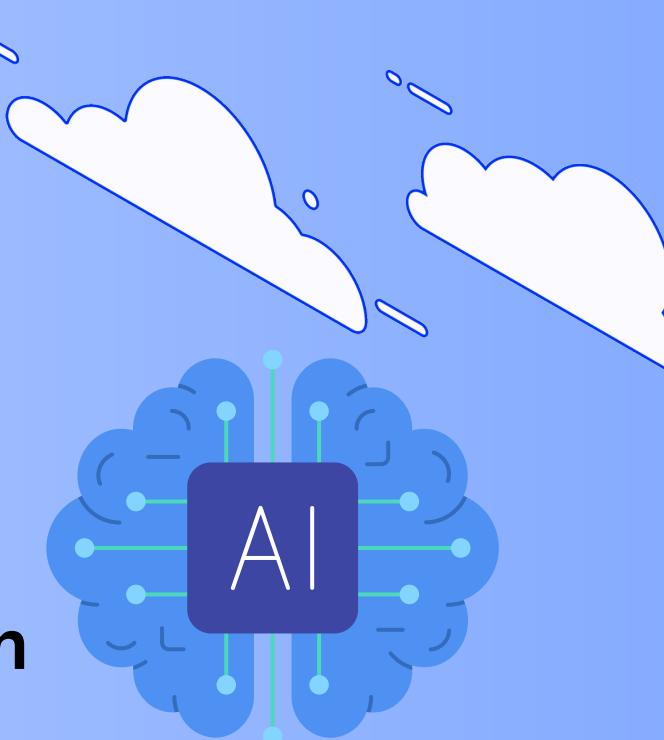
TỔNG QUAN VỀ LÝ THUYẾT



Giới thiệu về AI trong nhận diện hình ảnh

Object Detection: khả năng hệ thống máy tính và phần mềm định vị các đối tượng trong một hình ảnh và xác định từng đối tượng

Object Detection đã được sử dụng rộng rãi để phát hiện khuôn mặt, phát hiện xe cộ, đếm số người đi bộ, nhận diện đèn xanh đèn đỏ, hệ thống xe không người lái.



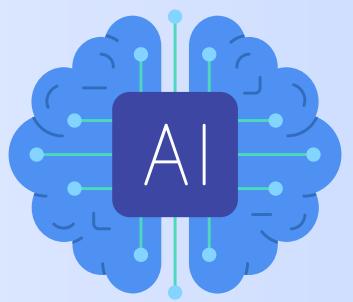


NHÓM 13

MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

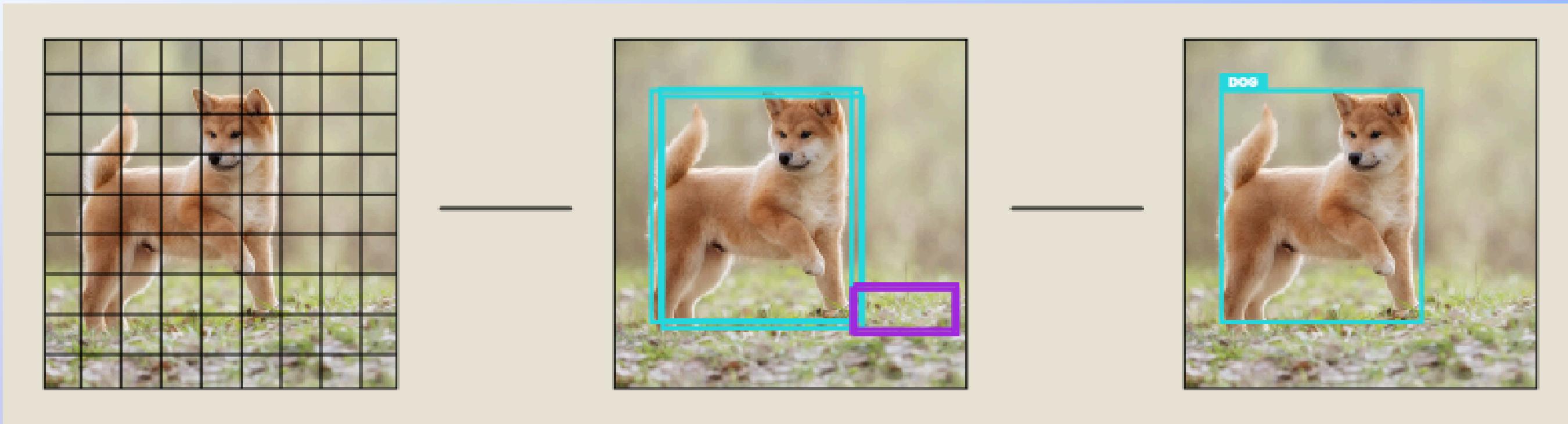
2.

TỔNG QUAN VỀ LÝ THUYẾT



Giới thiệu các công nghệ được ứng dụng trong đề tài nhận diện rác thải bằng Yolo

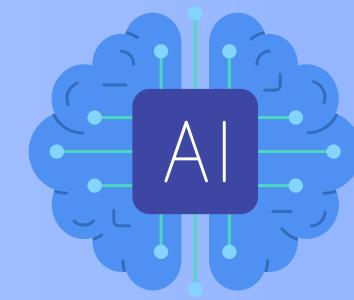
Yolo - You only look once: là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các **convolutional layers** và **connected layers**



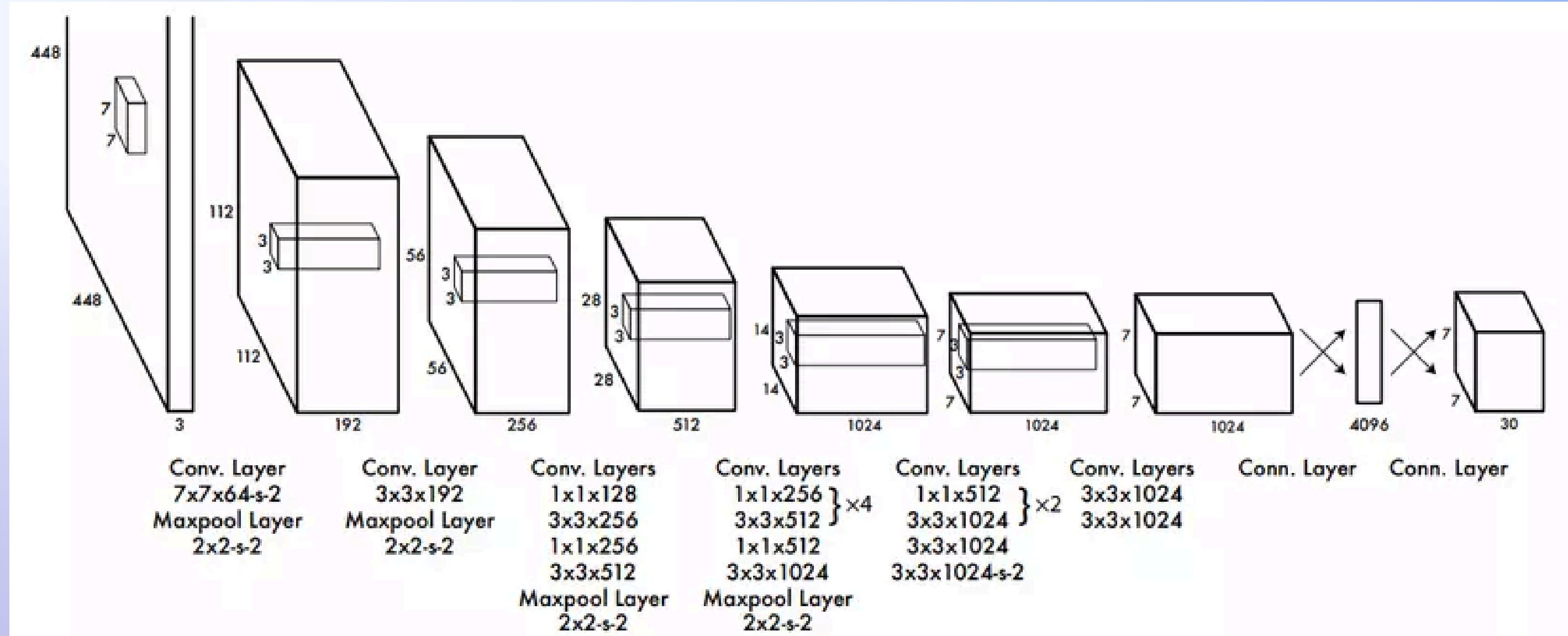


2.

TỔNG QUAN VỀ LÝ THUYẾT



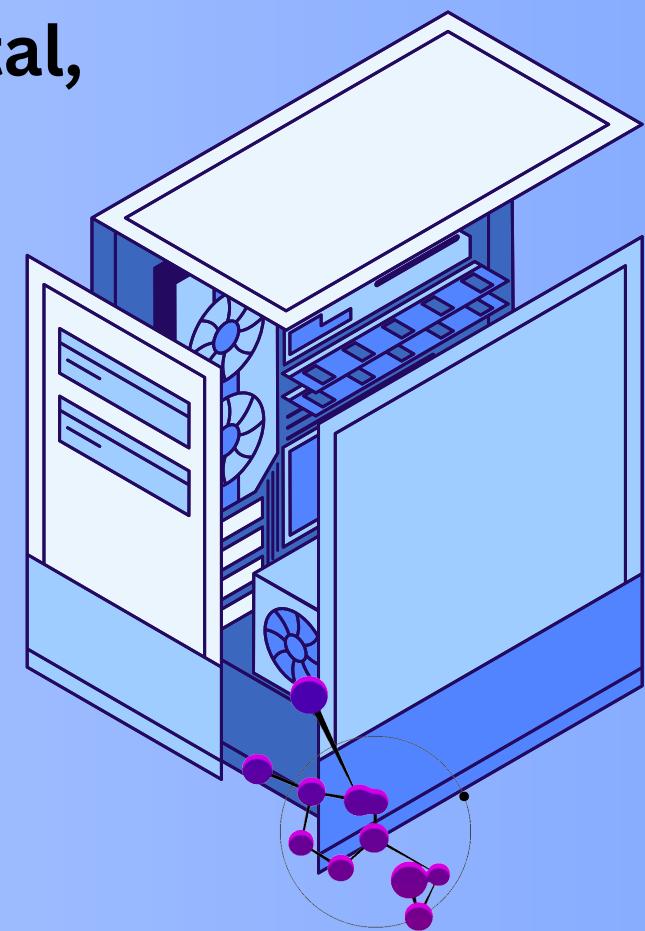
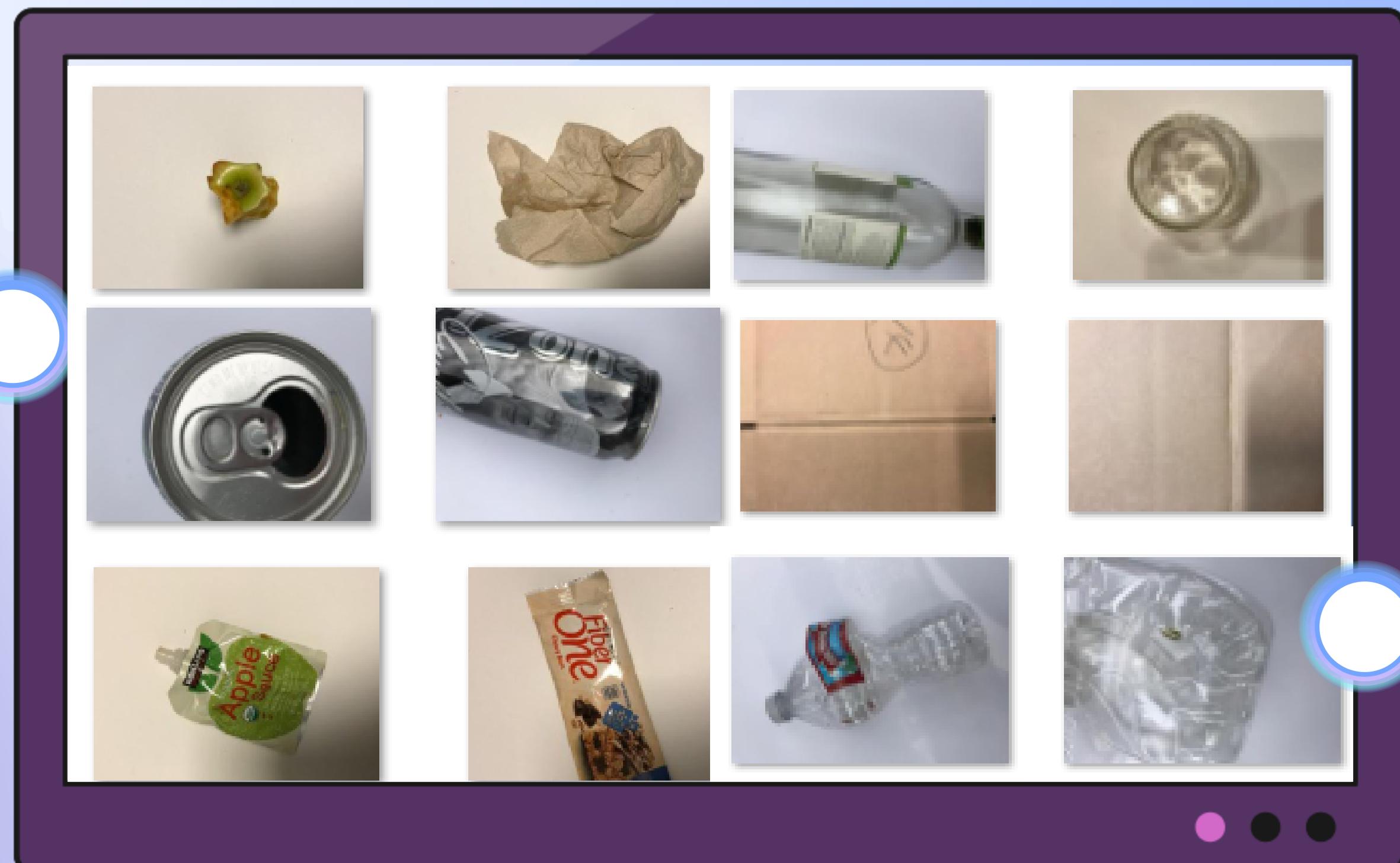
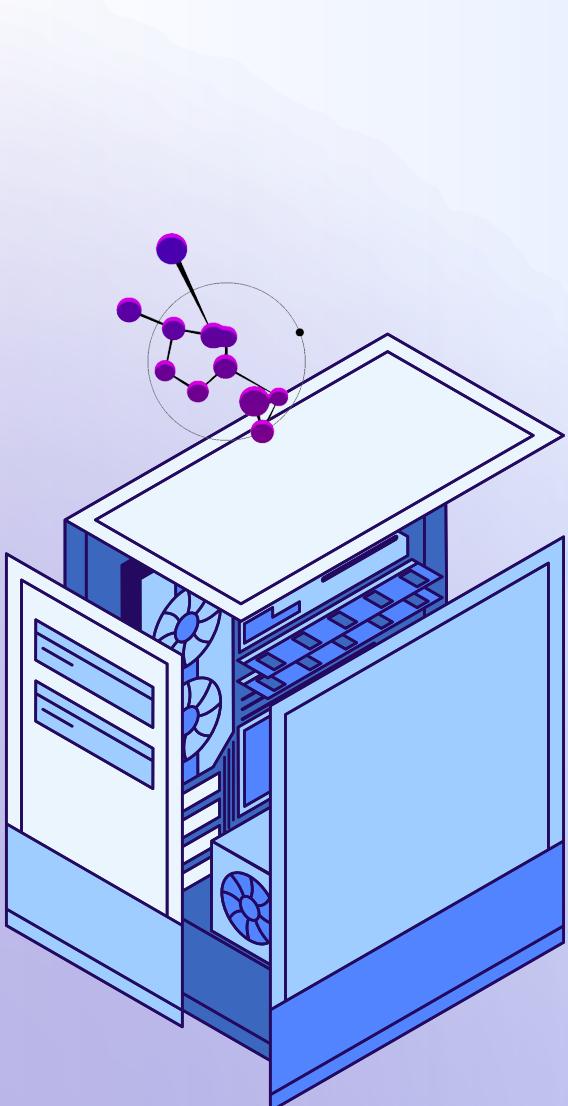
- **Convolutional layers** sẽ trích xuất ra các feature của ảnh
- **Full-connected layers** sẽ dự đoán ra xác suất đó và tọa độ của đối tượng

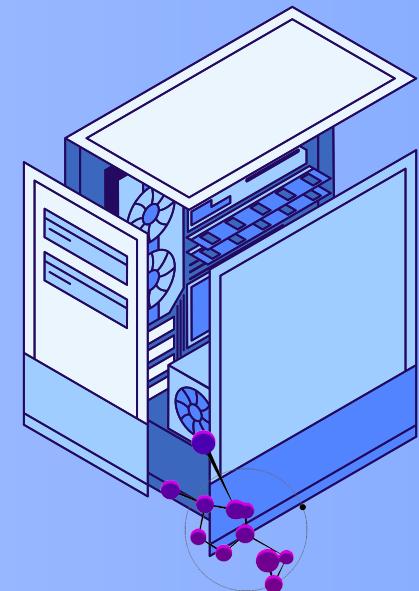




3. TRIỂN KHAI MÔ HÌNH

Bước 1: Thu thập dữ liệu đầu vào: Gồm hình ảnh của 6 loại rác thải: **Cardboard, Glass, Metal, Plastic, Paper, Trash.**

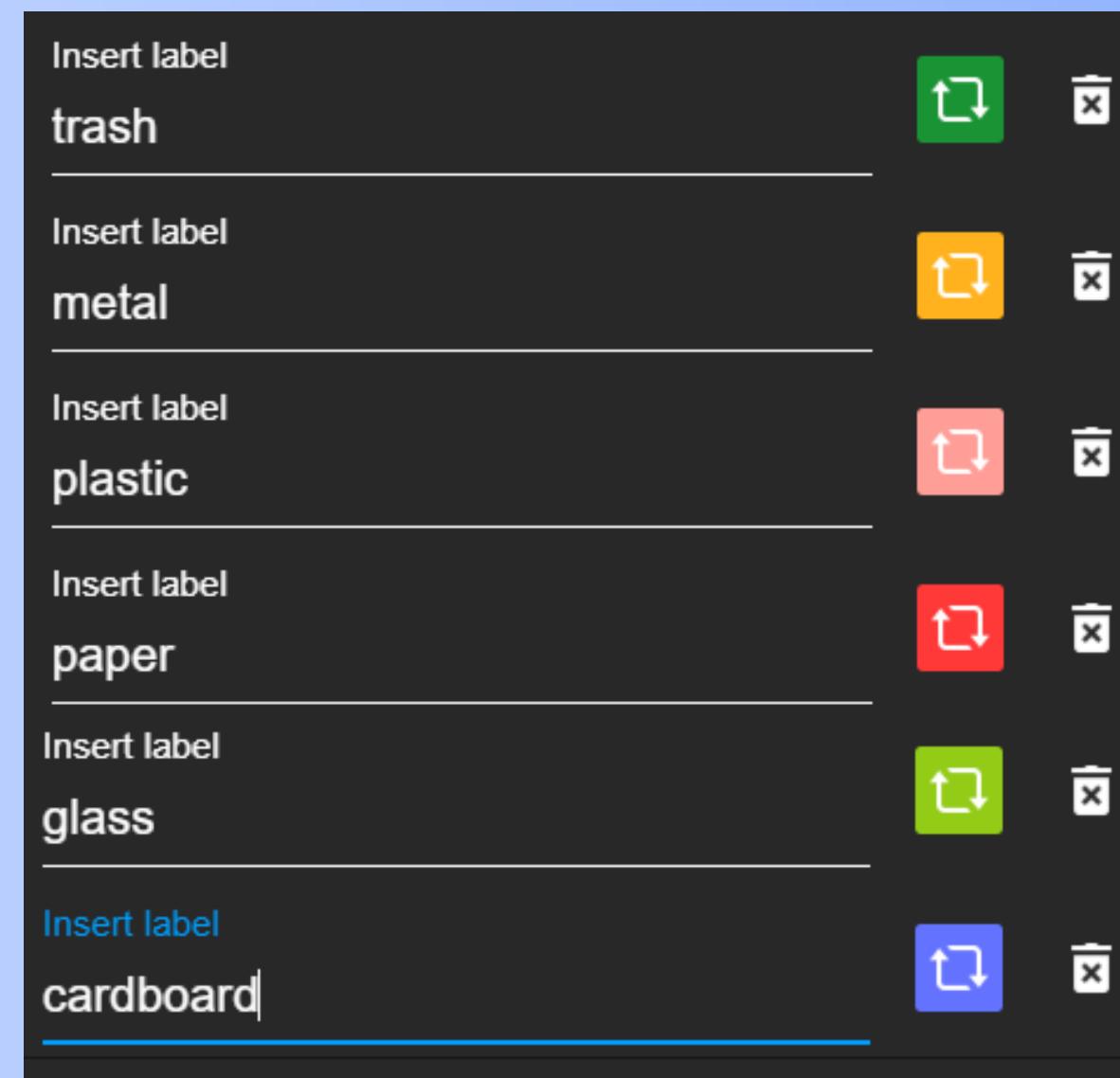
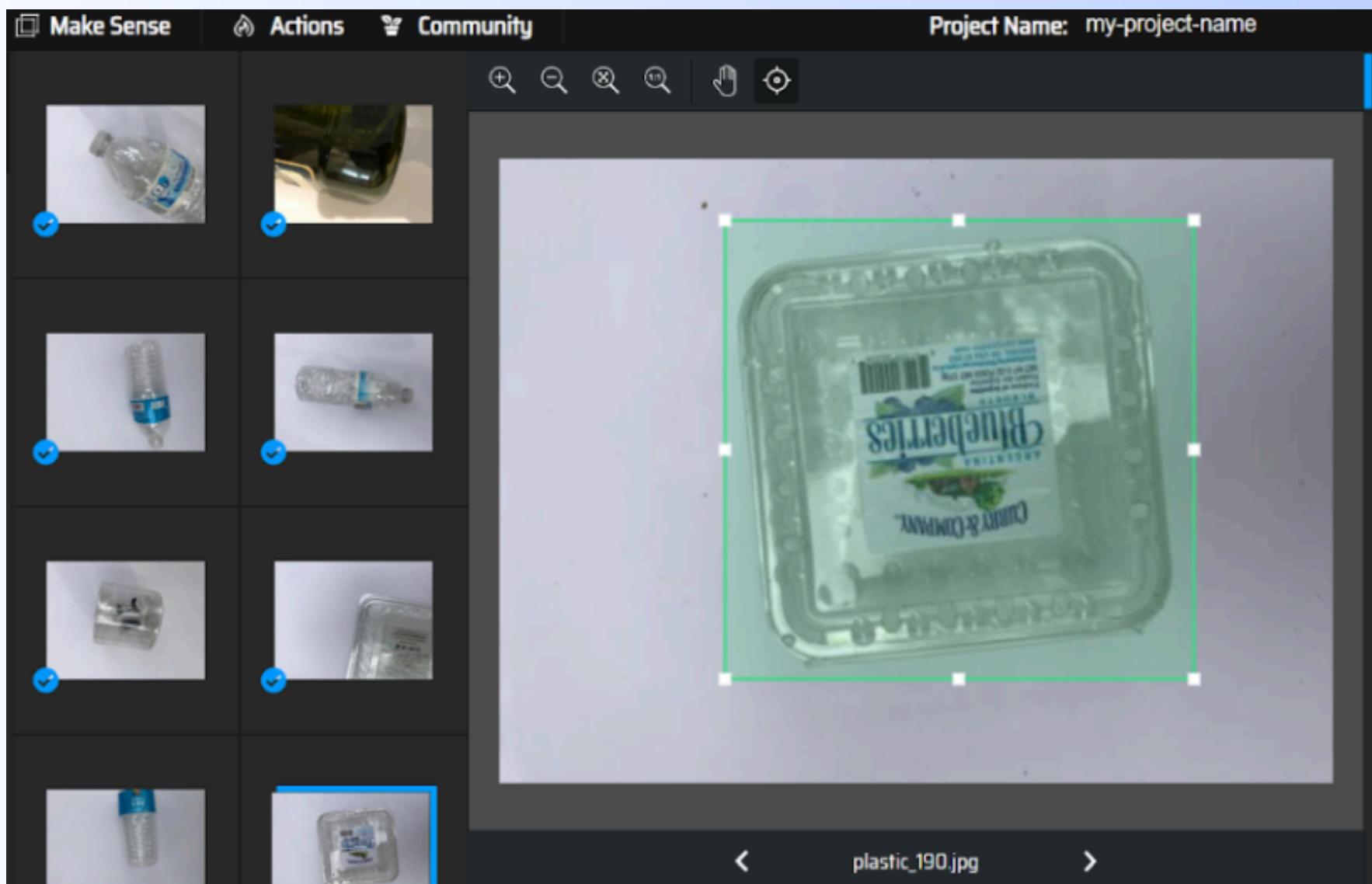


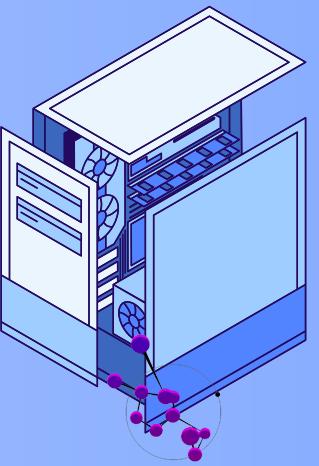


3.

TRIỂN KHAI MÔ HÌNH

Bước 2: Thực hiện gán nhãn cho từng hình ảnh bằng **Make Sense**.
Tổng cộng chia ra làm 6 lớp tương ứng với 6 loại rác thải





3.

TRIỂN KHAI MÔ HÌNH

Bước 3: Huấn luyện mô hình

- Cài đặt môi trường:

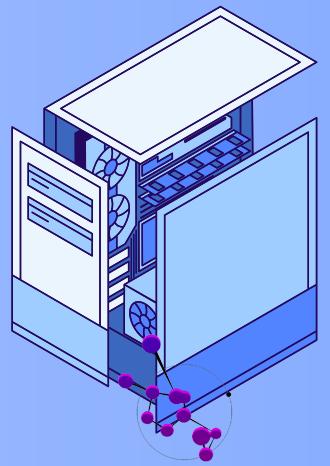
```
✓ [1] !git clone https://github.com/ultralytics/yolov5 # clone
25s   %cd yolov5
      %pip install -qr requirements.txt comet_ml # install

      import torch
      import utils
      display = utils.notebook_init() # checks

→ YOLOv5 🚀 v7.0-389-ge62a31b6 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
      Setup complete ✓ (2 CPUs, 12.7 GB RAM, 32.7/112.6 GB disk)
```

- Tải dữ liệu ảnh từ máy tính lên GG Colab





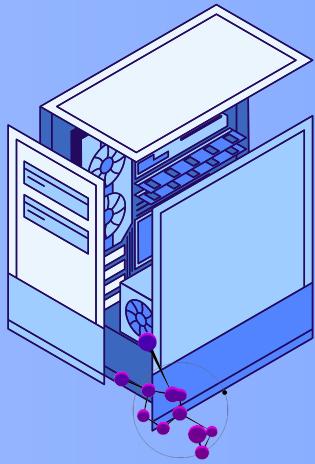
3. TRIỂN KHAI MÔ HÌNH

Bước 3: Huấn luyện mô hình

- Thay đổi đối tượng huấn luyện:

```
9 # Train/val/test sets as
10 # 1) dir: path/to/imgs
11 # 2) file: path/to/imgs.txt
12 # 3) list: [path/to/imgs1, path/to/imgs2, ...]
13 path: ../content/yolov5/coco128 # dataset root dir
14 train: /content/yolov5/coco128/train/images
15 val: /content/yolov5/coco128/validate/images
16 test: /content/yolov5/coco128/test/images/train2017
17
18 # Classes
19 names:
20 0: Trash
21 1: Cardboard
22 2: Glass
23 3: Metal
24 4: Paper
25 5: Plastic
```





3.

TRIỂN KHAI MÔ HÌNH

Bước 3: Huấn luyện mô hình

• Cài đặt tham số chạy mô hình:



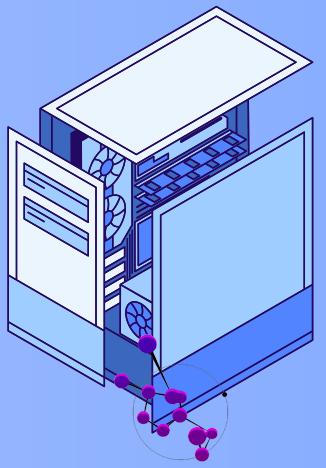
Batch size: Tham số xác định số lượng dữ liệu huấn luyện được sử dụng trong một lần lặp lại thuật toán.

Learning rate: Tham số xác định kích thước weight mà tại đó các tham số của mô hình được cập nhật trong quá trình đào tạo. Mặc định learning rate trong YOLOv5 là Initial learning rate: lr0=0.01.

Epochs: Một lần lặp hoàn chỉnh thông qua toàn bộ bộ tập dữ liệu trong quá trình đào tạo

```
[3] # Train YOLOv5s on COCO128 for 100 epochs
!python train.py --img 640 --batch 16 --epochs 100 --data coco128.yaml --weights yolov5s.pt --cache
```





3. TRIỂN KHAI MÔ HÌNH

Bước 3: Huấn luyện mô hình

- **Bắt đầu quá trình huấn luyện mô hình:**

```
Epoch      GPU_mem    box_loss    obj_loss    cls_loss    Instances    Size
0% 0/140 [00:00<?, ?it/s]/content/yolov5/train.py:412: FutureWarning: `torch.cuda.amp.au
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02129      0.01413      0.0008432          89      640:    1% 1/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02529      0.01298      0.001411          63      640:    1% 2/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02559      0.01284      0.00414          65      640:    2% 3/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02682      0.01137      0.003801          36      640:    3% 4/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02596      0.01136      0.003261          60      640:    4% 5/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02606      0.01213      0.003009          88      640:    4% 6/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02459      0.0112      0.002778          36      640:    5% 7/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.02487      0.01185      0.002553          84      640:    6% 8/140
with torch.cuda.amp.autocast(amp):
    95/99      4.18G      0.0245      0.01162      0.0027          54      640:    6% 9/140
```





NHÓM 13

MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

3.

TRIỂN KHAI MÔ HÌNH

Bước 4: Nhận diện đối tượng

```
from IPython.display import Image, display\n\n!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt \\\n--img 640 --conf 0.25 --source /content/test2.jpg\n\n# Đường dẫn tới ảnh kết quả\nimage_path = "runs/detect/exp/test2.jpg"\n\n# Hiển thị ảnh\ndisplay(Image(filename=image_path, width=640))
```

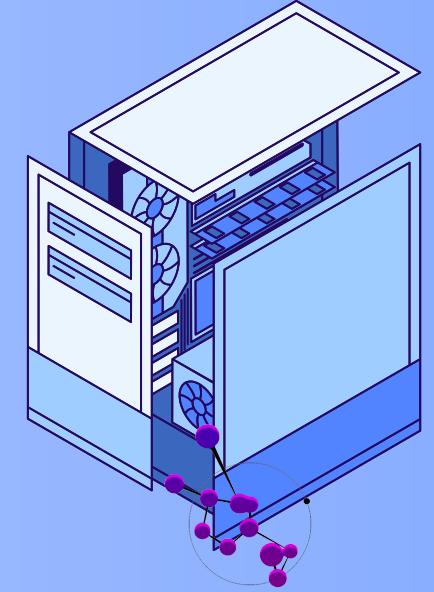


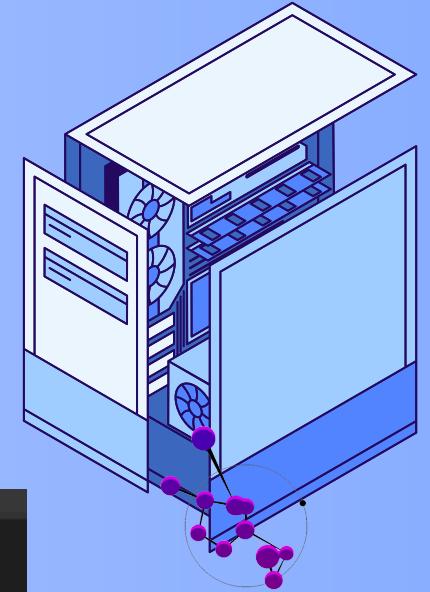
Dữ liệu mới



Sau khi nhận diện

Kiều Doan





3.

TRIỂN KHAI MÔ HÌNH

Bước 4: Nhận diện đối tượng

```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --img 480 --conf 0.25 --source /content/trash_062.jpg
from IPython.display import Image, display

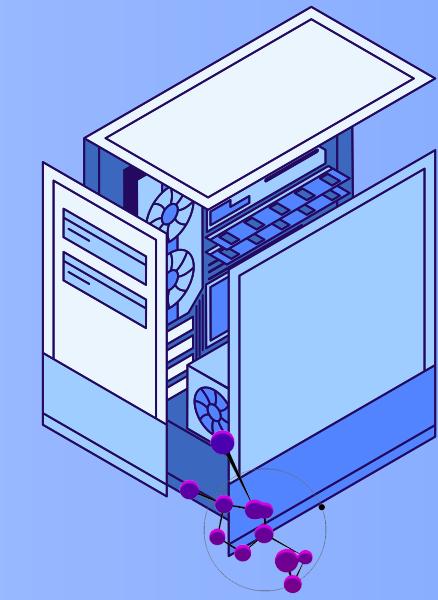
# Đường dẫn tới ảnh kết quả
image_path = "runs/detect/exp3/trash_062.jpg"

# Hiển thị ảnh
display(Image(filename=image_path, width=480))

detect: weights=['/content/yolov5/runs/train/exp/weights/best.pt'], source=/content/trash_062.jpg, data=data/coco128.yaml, imgsz=[480, 480], conf_thres=0.2
YOLOv5 🚀 v7.0-389-ge62a31b6 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
WARNING ⚠️ NMS time limit 0.550s exceeded
image 1/1 /content/trash_062.jpg: 384x480 1 Trash, 30.9ms
Speed: 0.4ms pre-process, 30.9ms inference, 609.2ms NMS per image at shape (1, 3, 480, 480)
Results saved to runs/detect/exp3
```





3.

TRIỂN KHAI MÔ HÌNH

Bước 5: Đánh giá kết quả:

```
[3] # Train YOLOv5s on COCO128 for 100 epochs
!python train.py --img 640 --batch 16 --epochs 100 --data coco128.yaml --weights yolov5s.pt --cache

→ 100 epochs completed in 1.505 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
  Class   Images  Instances      P      R    mAP50    mAP50-95: 100% 20/20
    all     636     1838    0.866    0.866    0.874    0.534
    Trash    636      160    0.818    0.844      0.8      0.48
  Cardboard  636      170    0.918    0.918    0.958      0.72
    Glass    636      175    0.853    0.795    0.843      0.523
    Metal    636      545    0.841    0.897    0.882      0.484
    Paper    636      510    0.927    0.886    0.913      0.528
    Plastic   636      278    0.84     0.856    0.849      0.47
```

✓ [4] !python val.py --data coco128.yaml --task test --weights /content/yolov5/runs/train/exp/weights/best.pt

→ val: data=/content/yolov5/data/coco128.yaml, weights=['/content/yolov5/runs/train/exp/weights/best.pt'],
YOLOv5 🚀 v7.0-389-ge62a31b6 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
test: Scanning /content/yolov5/coco128/test/labels/train2017... 317 images, 1 backgrounds, 0 corrupt: 10
test: New cache created: /content/yolov5/coco128/test/labels/train2017.cache

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 10/10 [00]
all	318	326	0.867	0.857	0.89	0.724
Trash	318	85	0.836	0.661	0.784	0.522
Cardboard	318	50	0.933	1	0.994	0.959
Glass	318	30	0.644	0.967	0.739	0.492
Metal	318	102	0.987	0.759	0.963	0.845
Paper	318	59	0.933	0.898	0.972	0.802

Speed: 0.2ms pre-process, 6.4ms inference, 3.3ms NMS per image at shape (32, 3, 640, 640)

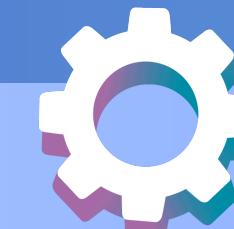


4.

KẾT LUẬN



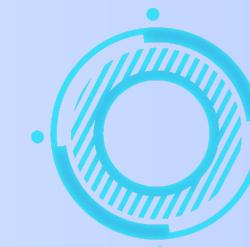
- **Mô hình cân bằng giữa Precision và Recall**
- **Precision (độ chính xác)** khá tốt ở tất cả các lớp thấp nhất là lớp Glass
- **Recall (độ nhạy)** ổn định với tất cả các lớp
- Giá trị **Mean Average Precision** tại IoU 50% cao
- Giá trị giảm khi mở rộng mức IoU từ 50% đến 95%, nhưng vẫn được đánh giá là khá tốt.



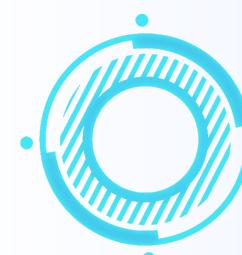


4.

KẾT LUẬN

 HẠN CHẾ CỦA MÔ HÌNH

- Dữ liệu chưa đa dạng
- Hiệu suất của mô hình trong môi trường thực tế (nhiều, ánh sáng, vật cản) chưa được kiểm nghiệm đầy đủ.

 HƯỚNG PHÁT TRIỂN

- **Cải thiện dữ liệu**
- **Tối ưu hóa mô hình:** Sử dụng các kiến trúc tiên tiến hơn như YOLOv8, EfficientDet hoặc Transformer-based models. Điều chỉnh hyperparameters để đạt được độ hội tụ tốt hơn.
- **Ứng dụng công nghệ mới:** Tích hợp các phương pháp học tăng cường (Reinforcement Learning) để giúp mô hình học từ môi trường thực tế.



MÔ HÌNH PHÂN LOẠI RÁC THẢI BẰNG THUẬT TOÁN YOLO

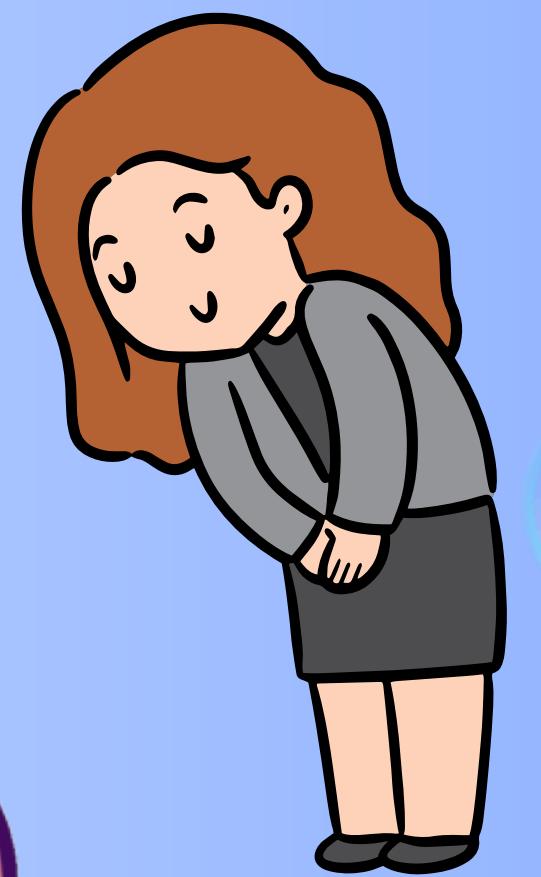


5.

ỨNG DỤNG THỰC TẾ

- Tích hợp mô hình AI vào hệ thống với camera.
- Ứng dụng trên các dây chuyền xử lý rác để tự động phân loại trước khi tái chế.





NHÓM 13

