PROBLEMS

1/ Install Ubuntu (recommend: version 18.04.6): https://www.youtube.com/watch?v=oZcvqfWf_ps&t=1871s

Note: Phân vùng ít nhất 100GB

2/ Download Anaconda: https://www.anaconda.com/products/distribution

Windows:

B1: Tải file từ trang chủ

B2: Chạy file đã tải

Ubuntu:

B1: Tải file từ trang chủ

B2: chạy lệnh -> bash ~/Downloads/Anaconda3-2020.05-Linux-x86_64.sh

Note: Download Anaconda xong nên tạo môi trường ảo và cài thư viện trên môi trường ảo đó

3/ Sử dụng python 3.7

4/ Tải cuda

B1: Chọn **version** cho phù hợp https://en.wikipedia.org/wiki/CUDA

B2: Cài đặt CUDAToolkit

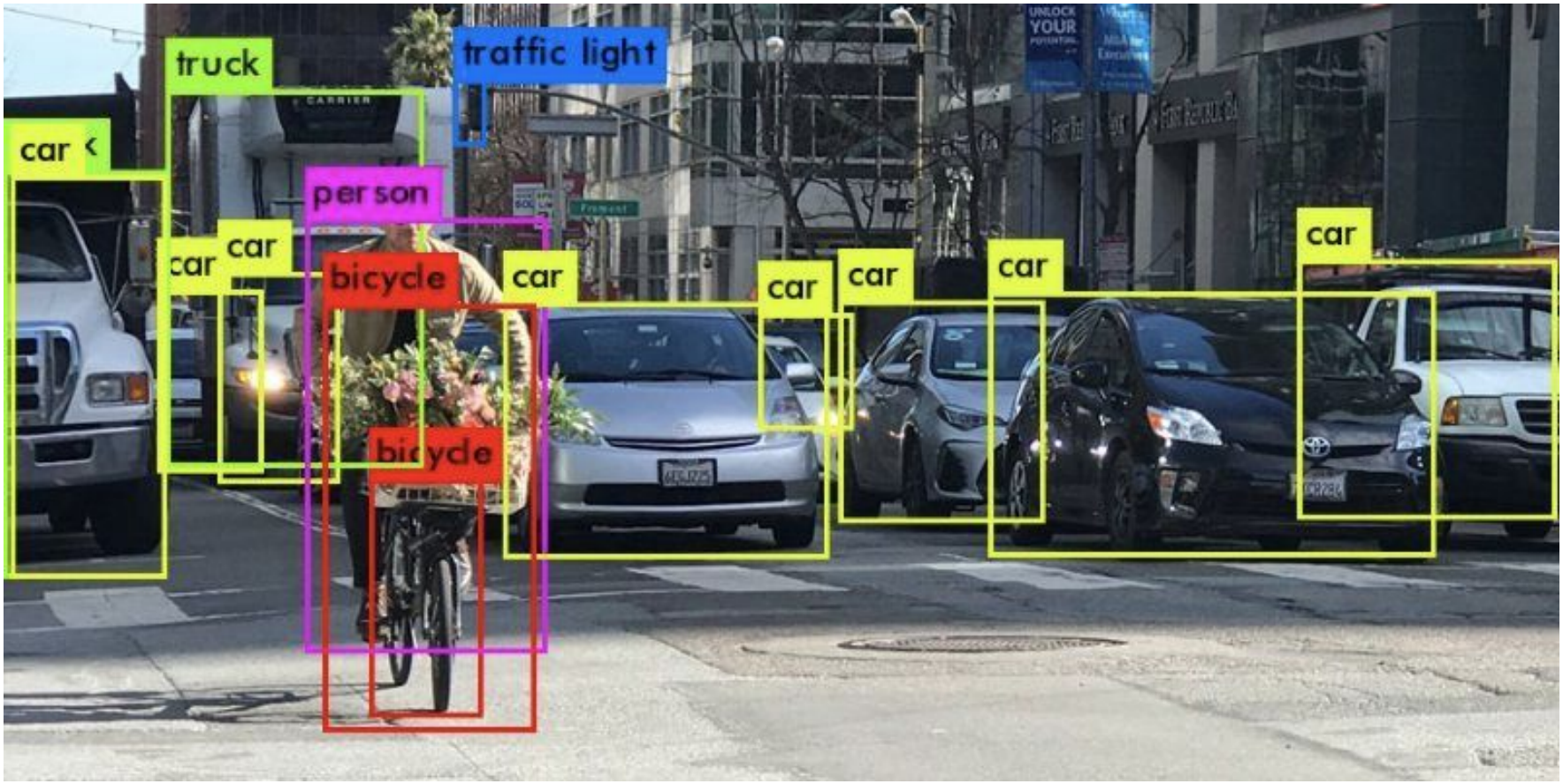
https://thigiacmaytinh.com/build-darknet-voi-cuda-tren-ubuntu-18-04/

https://miai.vn/2021/02/24/cach-cai-dat-cuda-tren-ubuntu-de-tang-hieu-nang-train-model/

CONTENT

1. INTRODUCT TO OBJECT DETECTION

Ứng dụng trong giao thông



1. INTRODUCT TO OBJECT DETECTION

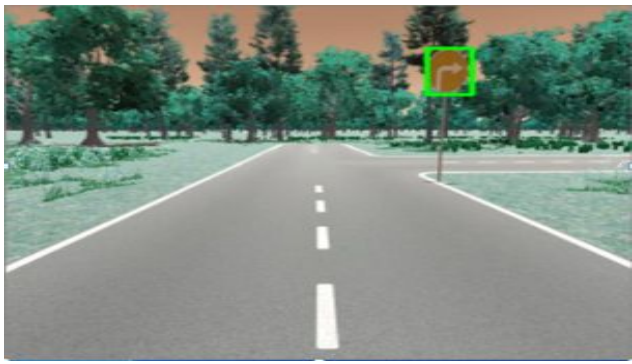
Event Retrieval

2. Using Yolov5 as a blackbox

- What is the object?
- Why use YOLO?

2. Using Yolov5 as a blackbox

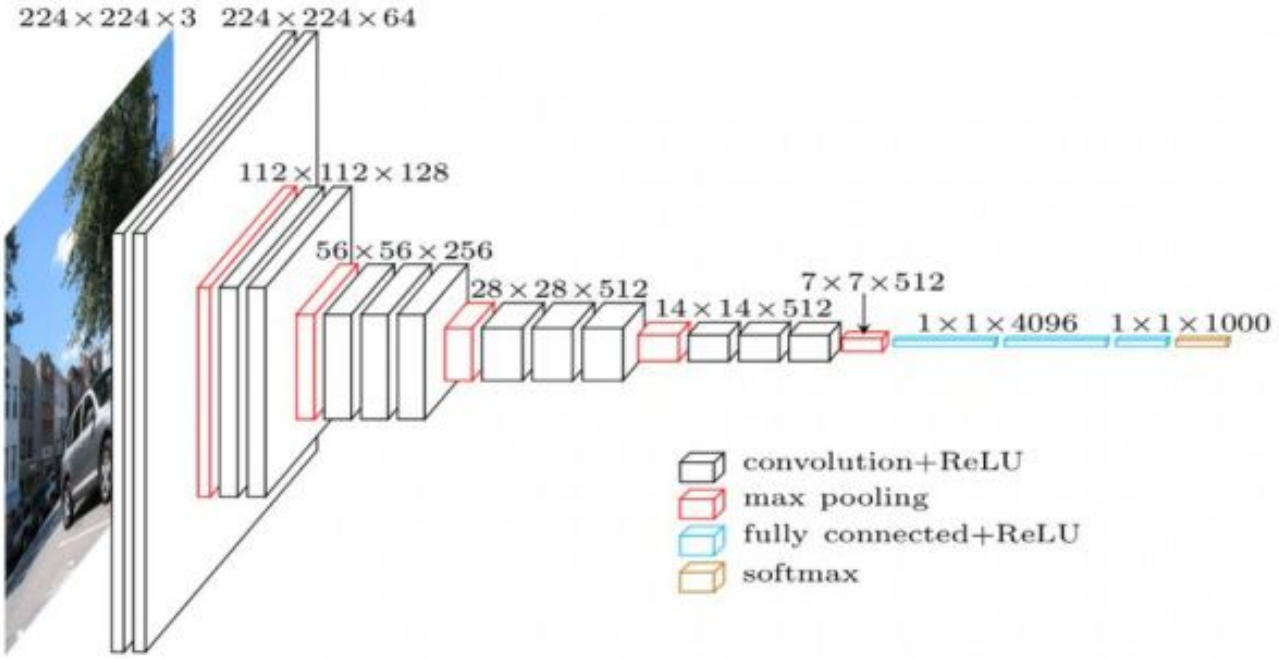
INPUT

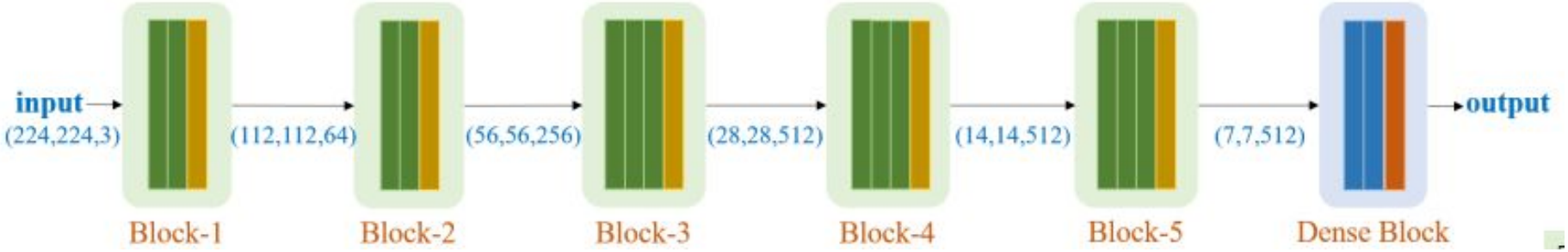


OUTPUT

2. Using Yolov5 as a blackbox

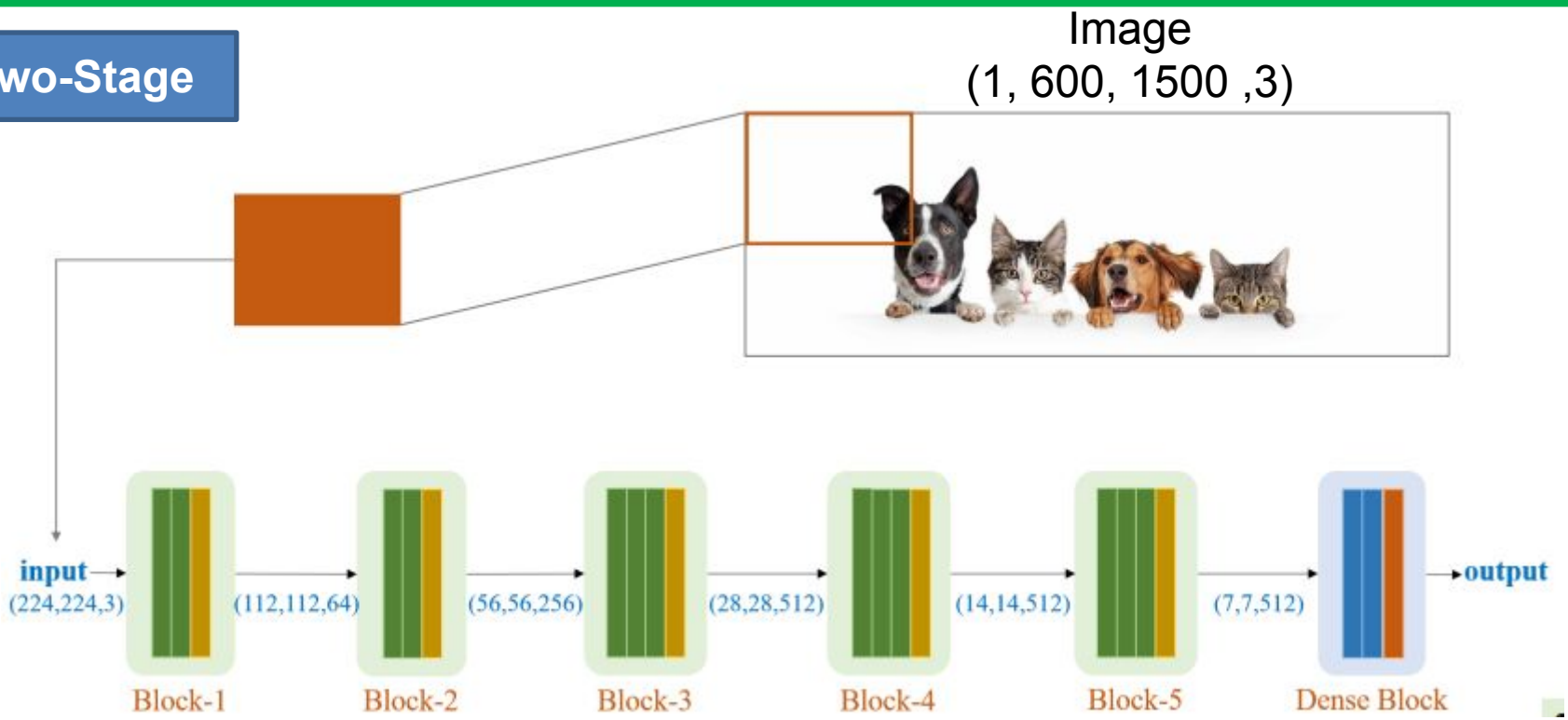
Two-Stage: **R-CNN, FPN, Mask R-CNN**
One-Stage: **Yolo, SSD**





2. Using Yolov5 as a blackbox

2.1 Two-Stage



Problems:
1/ High Cost
2/ Overlap

→

C1: Tăng Stride
C2: Selective Search
C3: Feature Exaction

→

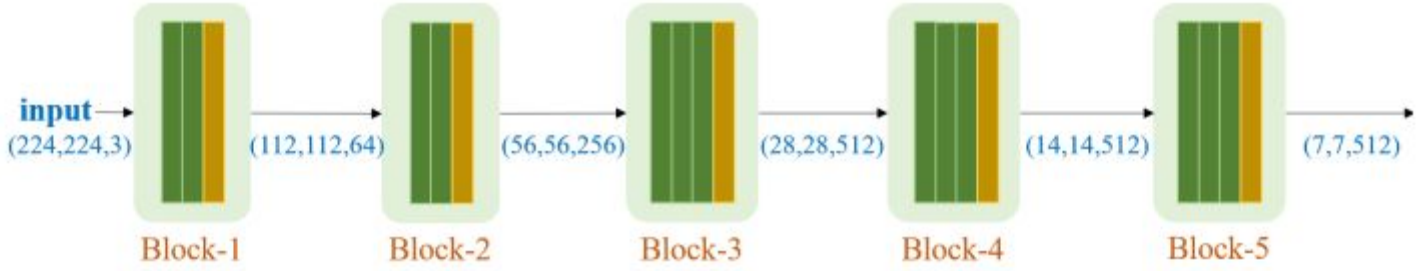
Idea: Model Classification + Selective Search

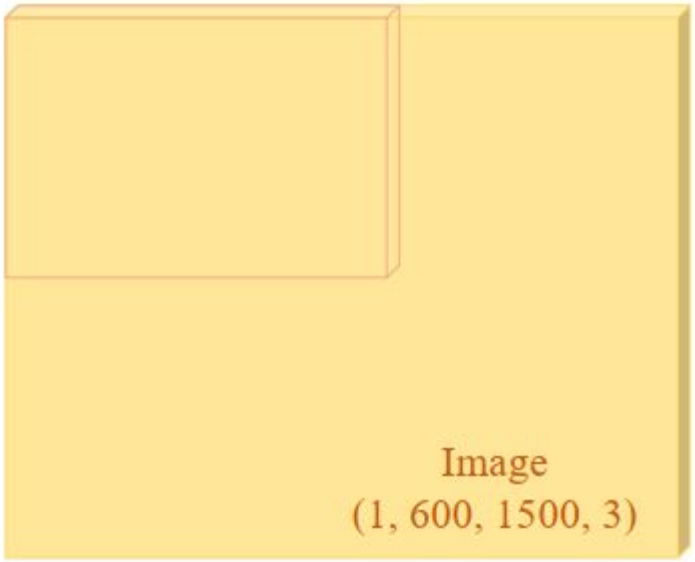
↓

RCNN (AlexNet)

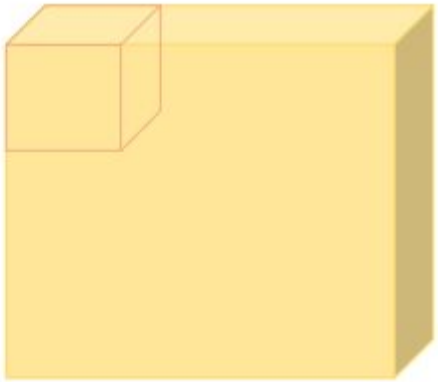
2. Using Yolov5 as a blackbox

2.2 One-Stage



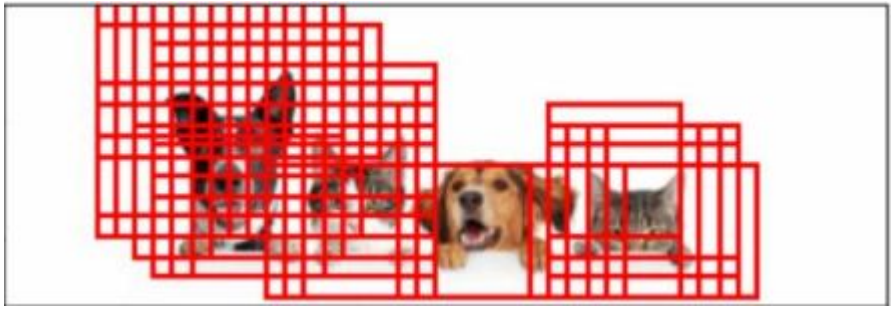


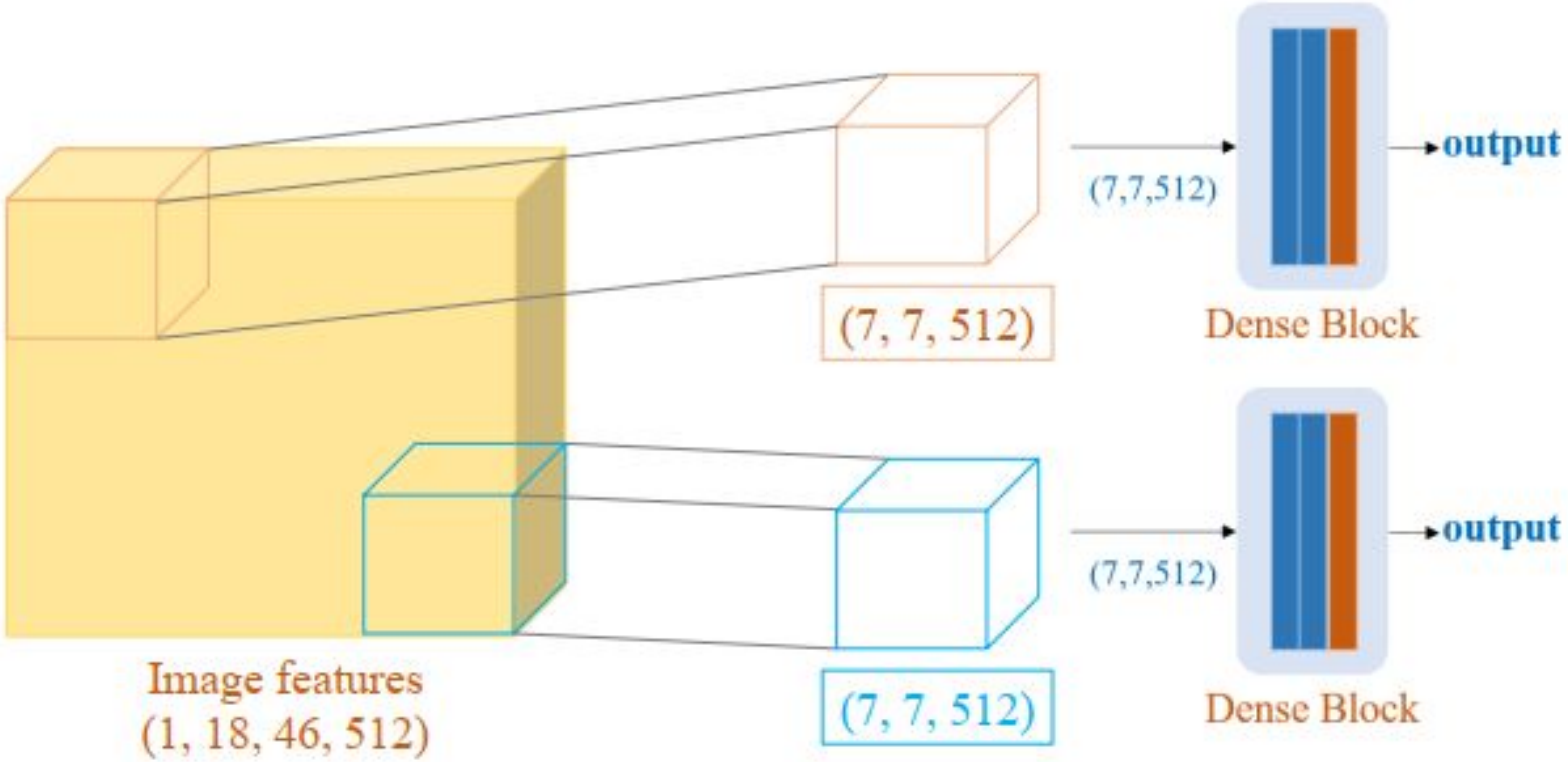
Feature Extraction



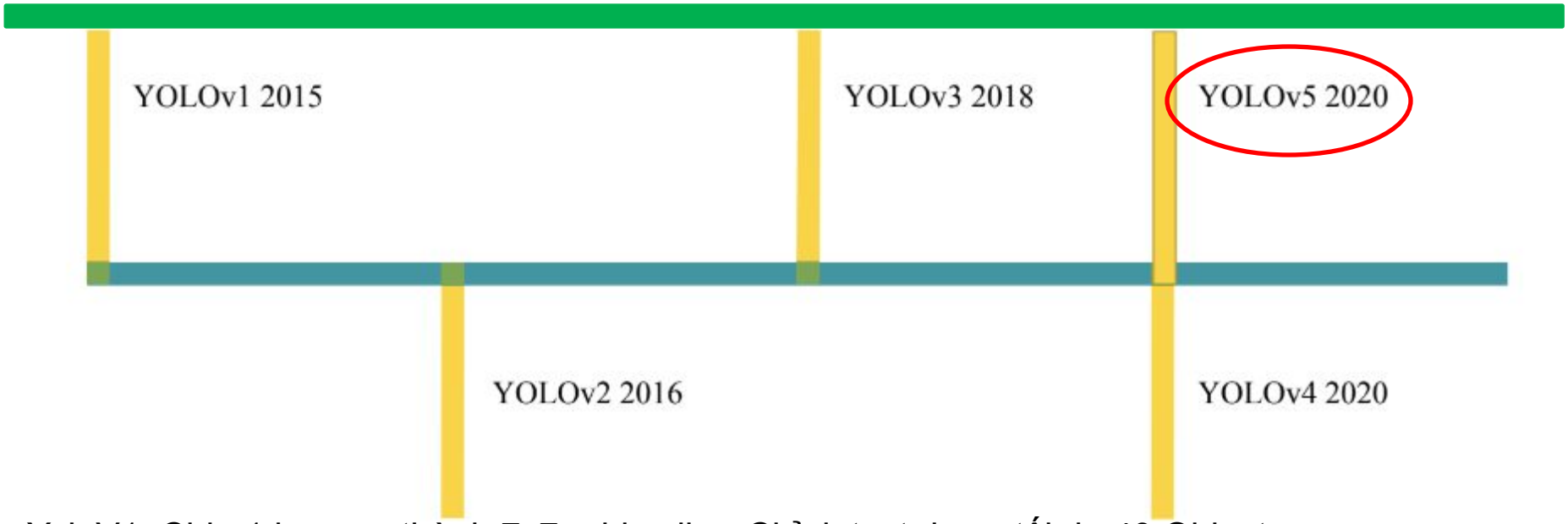
2. Using Yolov5 as a blackbox

2.2 One-Stage





3. PIPELINE



- YoloV1: Chia 1 images thành 7x7 grid cell -> Chỉ detect được tối đa 49 Objects
- YoloV2: Darknet19 và Chỉ detect được 13x13 Object
- YoloV3: DarkNet53 và giải quyết vấn đề small object
- YoloV4: Cải thiện kết quả của Model bằng cách tăng cường dữ liệu như (mixup, mosaic, ...)
- YoloV5: Tương tự như YoloV4 nhưng có **tốc độ nhanh hơn** khi **dự đoán** nhờ vào:
  * Thay đổi kiến trúc mạng (Backbone, Neck)
  * Data Augmentation: Mosaic, Copy-paste, MixUp, ...
  * Thêm hệ số scale cho Objectness Loss
  * Anchor Box: Auto Anchor sử dụng GA

https://viblo.asia/p/tong-hop-kien-thuc-tu-yolov1-den-yolov5-phan-3-63vKjgJ6Z2R#_yolov5-17

3. PIPELINE

Các phiên bản mô hình YOLOv5 gồm
- 5 phiên bản: YOLOv5n, YOLOv5s, ... Với kích thước ảnh **640**
- 5 phiên bản: YOLOv5n6, YOLOv5s6, .. Với kích thước ảnh **1280**

Nano	Small	Medium	Large	XLarge
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB_{FP16} 6.3 ms_{V100} 28.4 mAP_{COCO}	14 MB_{FP16} 6.4 ms_{V100} 37.2 mAP_{COCO}	41 MB_{FP16} 8.2 ms_{V100} 45.2 mAP_{COCO}	89 MB_{FP16} 10.1 ms_{V100} 48.8 mAP_{COCO}	166 MB_{FP16} 12.1 ms_{V100} 50.7 mAP_{COCO}

3. PIPELINE

3. PIPELINE

3.2. Labels

Cài đặt công cụ labelImg

```
pip3 install labelImg
labelImg
labelImg [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

Ctrl + u	Load all of the images from a directory
Ctrl + r	Change the default annotation target dir
Ctrl + s	Save
Ctrl + d	Copy the current label and rect box
Ctrl + Shift + d	Delete the current image
Space	Flag the current image as verified
w	Create a rect box
d	Next image
a	Previous image
del	Delete the selected rect box
Ctrl++	Zoom in
Ctrl--	Zoom out
↑→↓←	Keyboard arrows to move selected rect box

Hướng dẫn:

- https://www.youtube.com/watch?v=de3issSnfKI
- https://devai.info/2020/12/13/labelimg-tool/

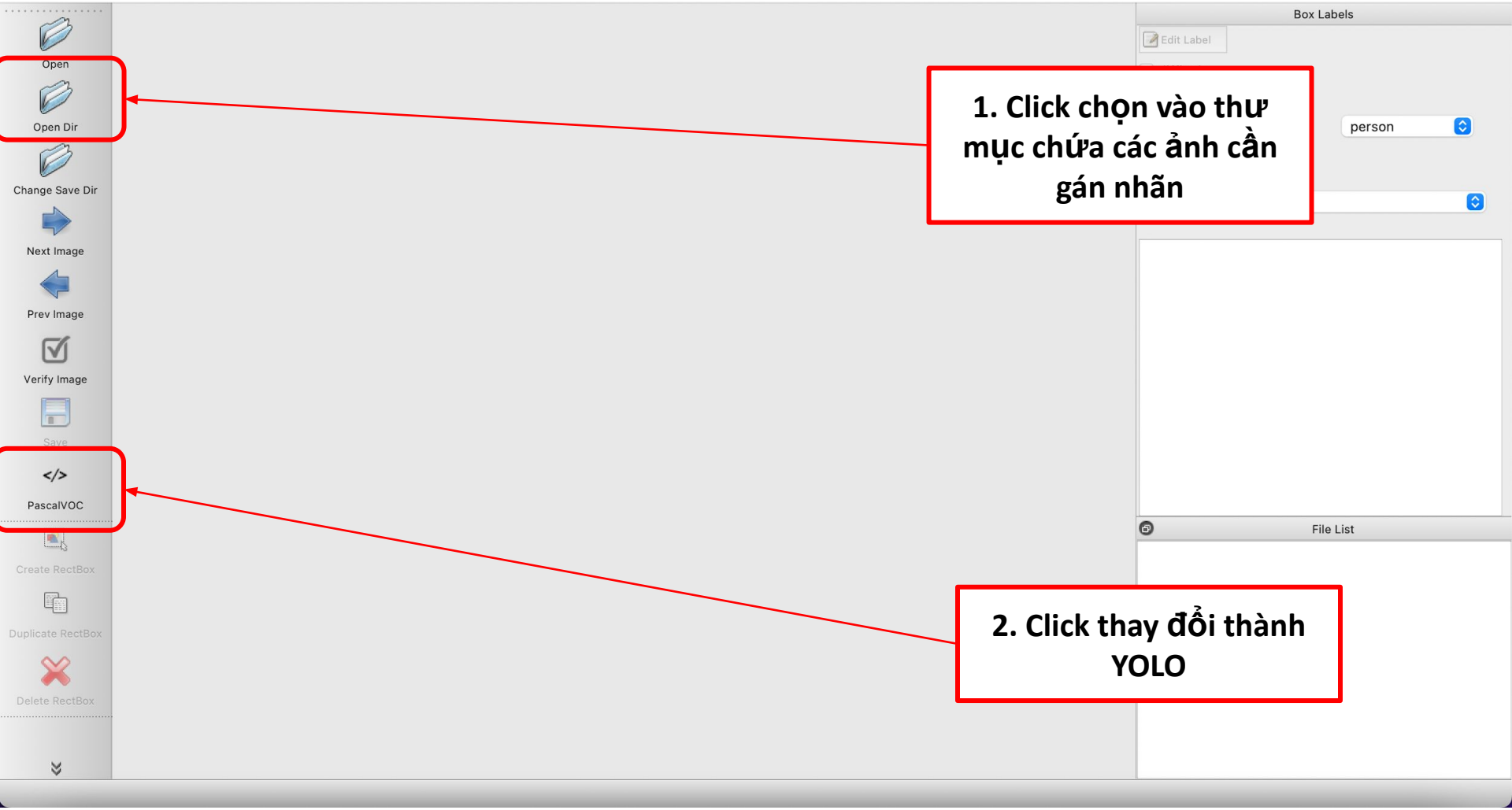
https://github.com/heartexlabs/labelImg

3. PIPELINE

3.2. Labels

3. PIPELINE

3.2. Labels

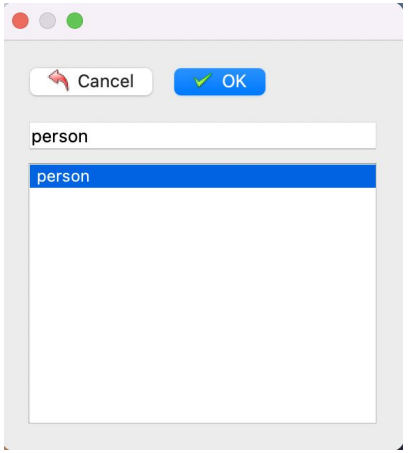


3. PIPELINE

1. Click chọn “Create RectBox”

2. Vẽ hình chữ nhật sao cho khớp với đối tượng cần xác định

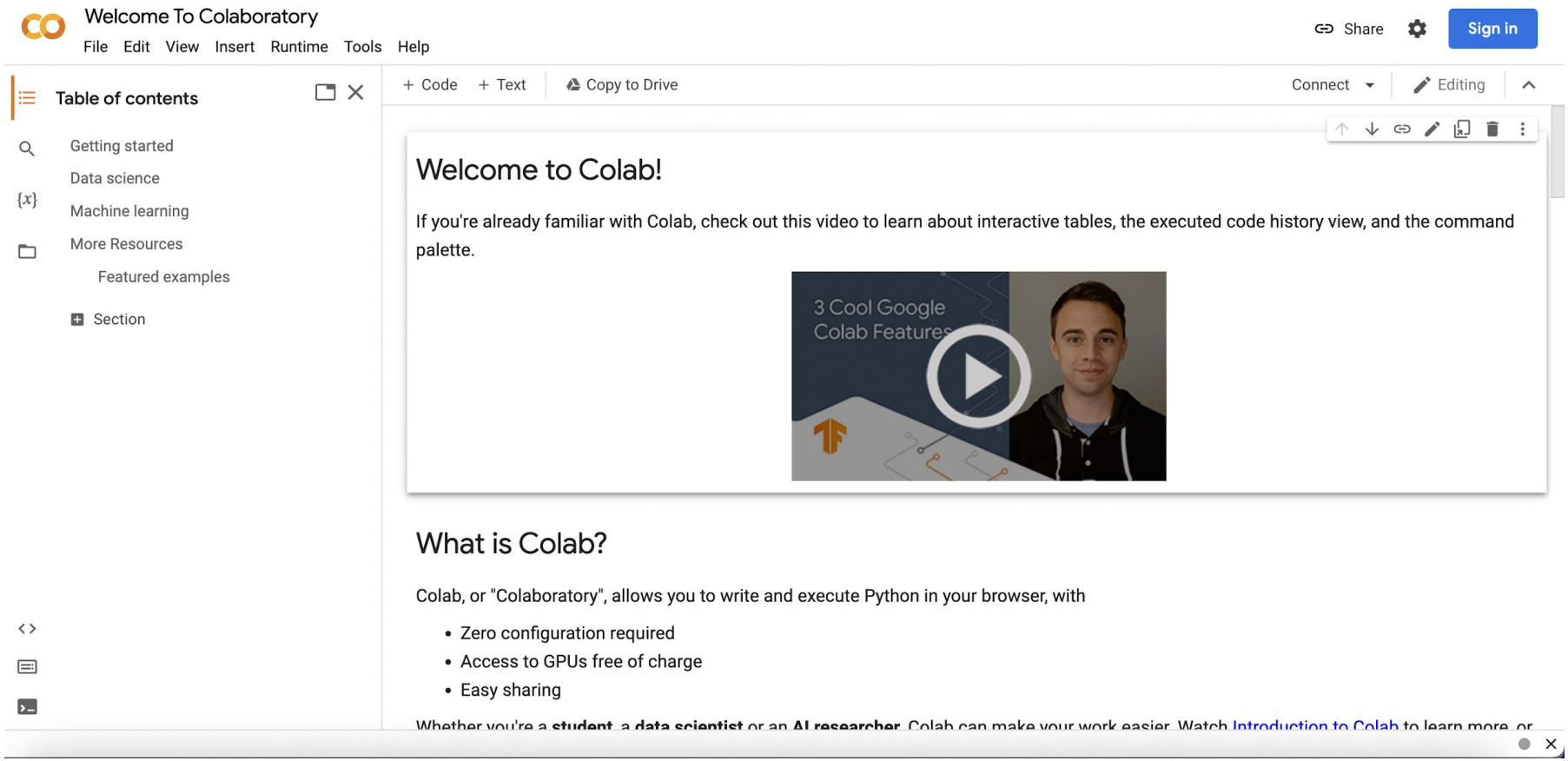
3. Chọn tên class tương ứng với đối tượng



Thực hiện tương tự với các đối tượng còn lại

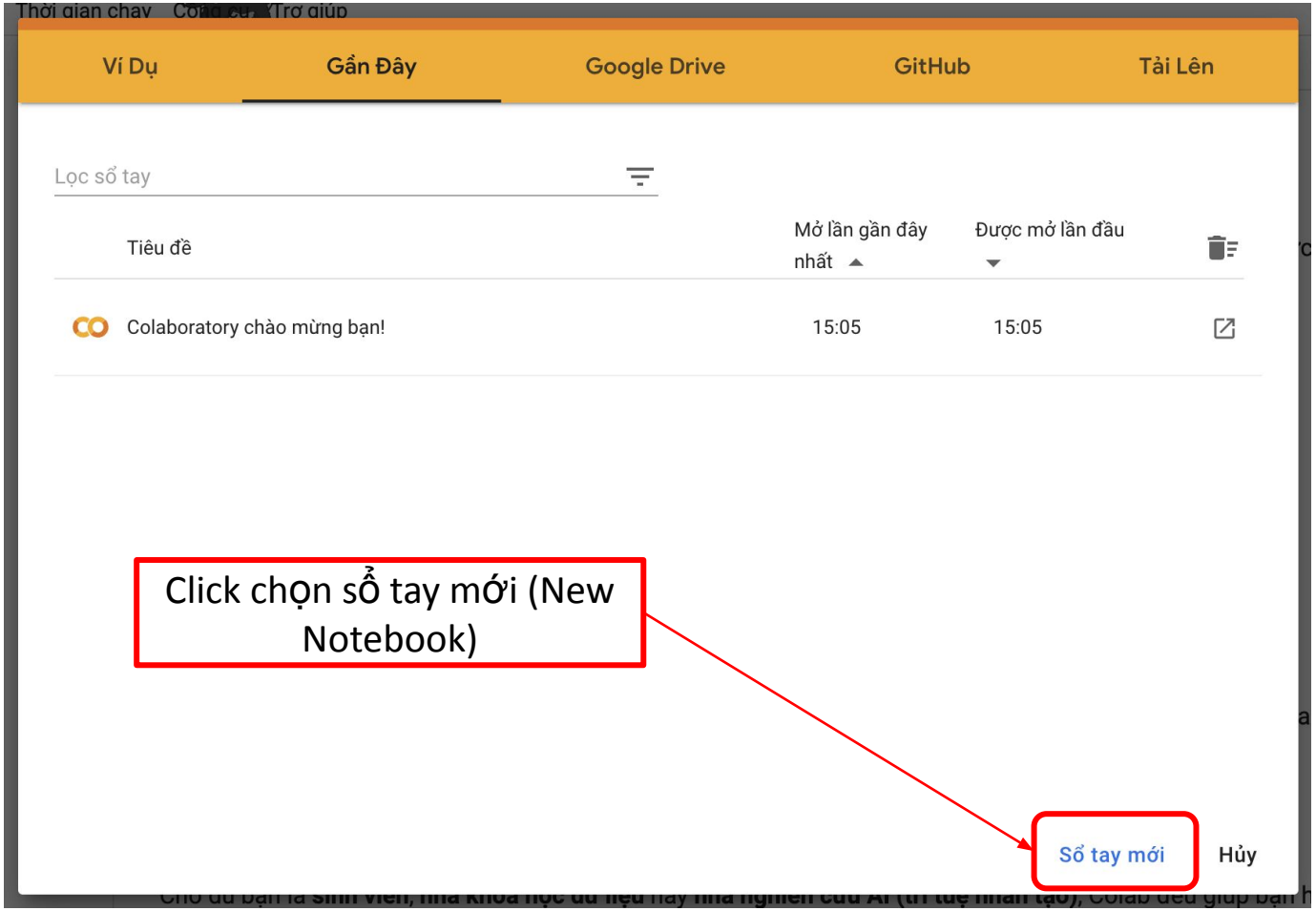
4. HOW TO TRAIN YOLOV5

❖ Google Colab



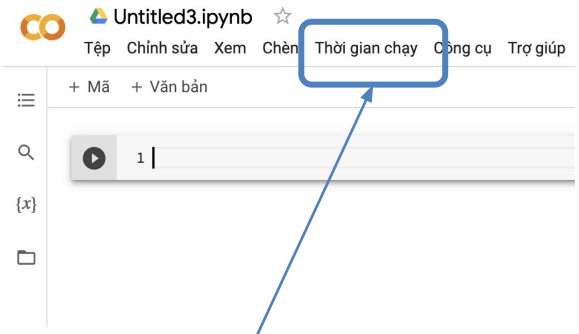
4. HOW TO TRAIN YOLOV5

❖ Google Colab

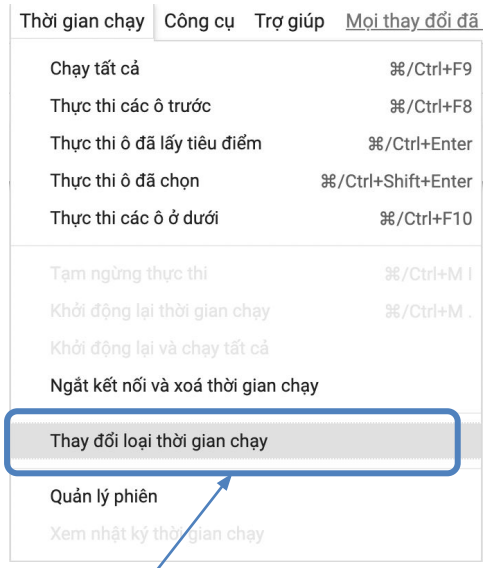


4. HOW TO TRAIN YOLOV5

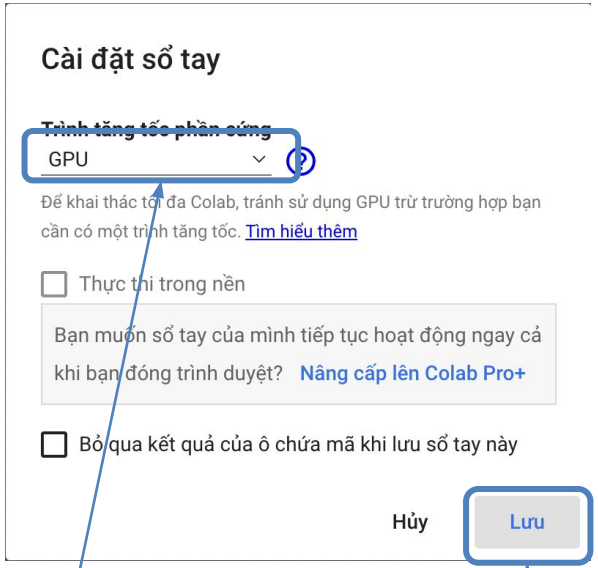
❖ Google Colab



1. Click vào **"Thời gian chạy (Runtime)"**



2. Click vào **"Thay đổi loại thời gian chạy (Change Runtime Type)"**

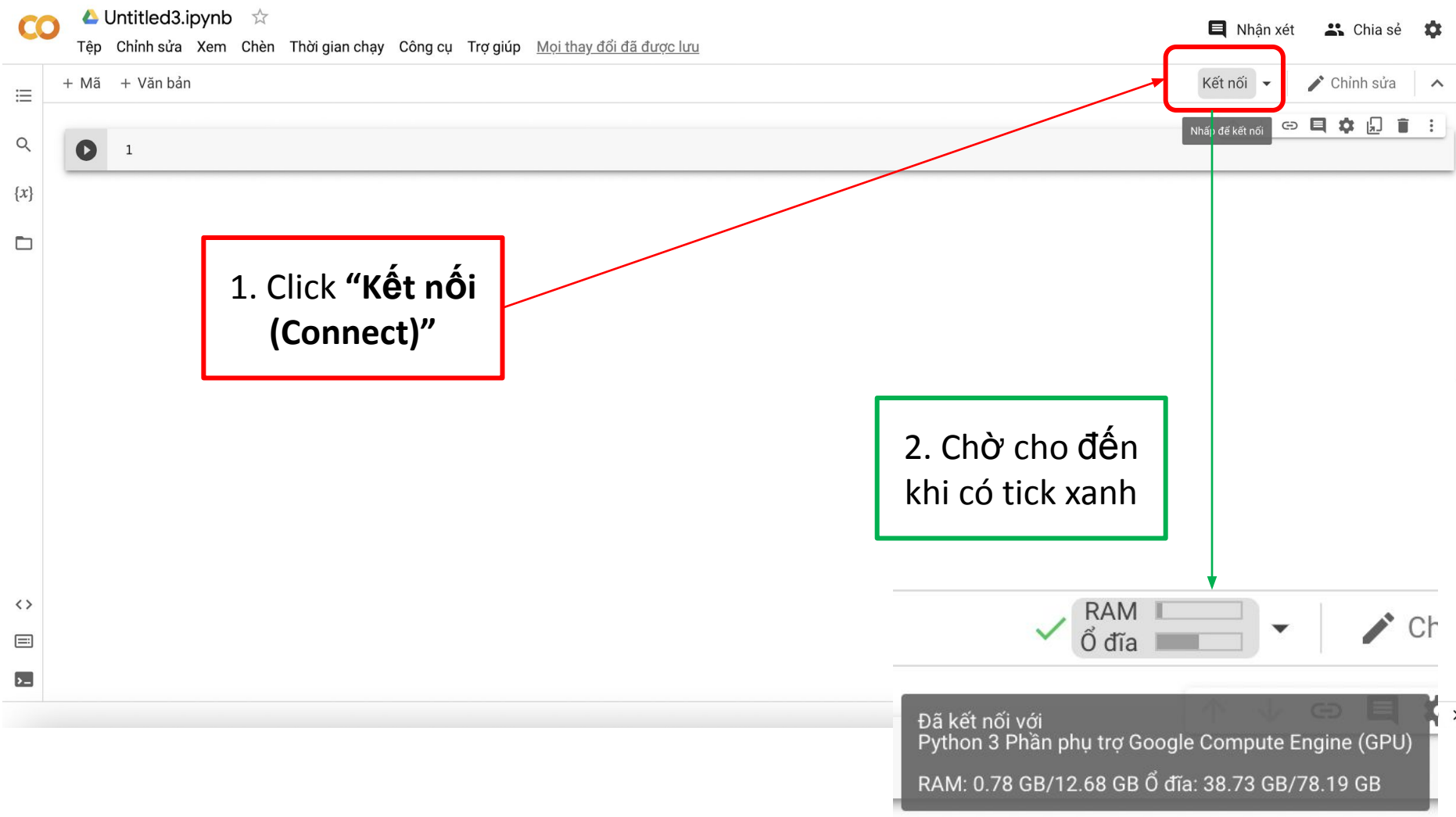


3. Click đổi từ **None sang GPU**

4. Click chọn **"Lưu (Save)"**

4. HOW TO TRAIN YOLOV5

❖ Google Colab



4. HOW TO TRAIN YOLOV5

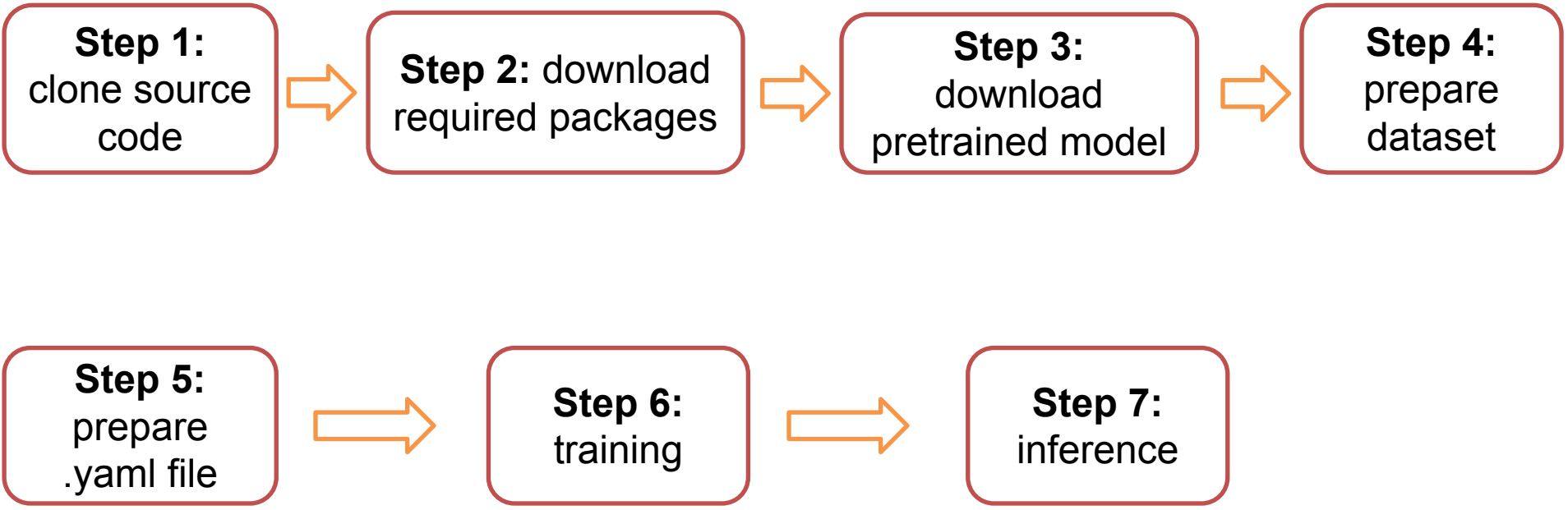
Link Colab:
https://drive.google.com/file/d/1fOKopYn7TqaRJ0pTQN_rgvF3pHKu7ydG/view?usp=share_link

YOLOv5: a family of object detection architectures and models pretrained on the COCO dataset

source: https://github.com/ultralytics/yolov5

4. HOW TO TRAIN YOLOV5

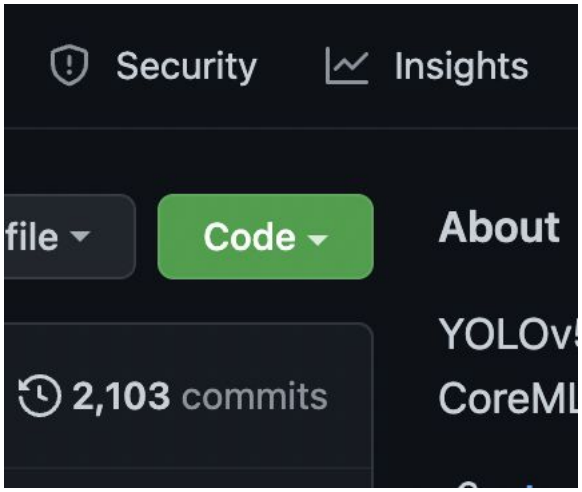
❖ Pipeline Training



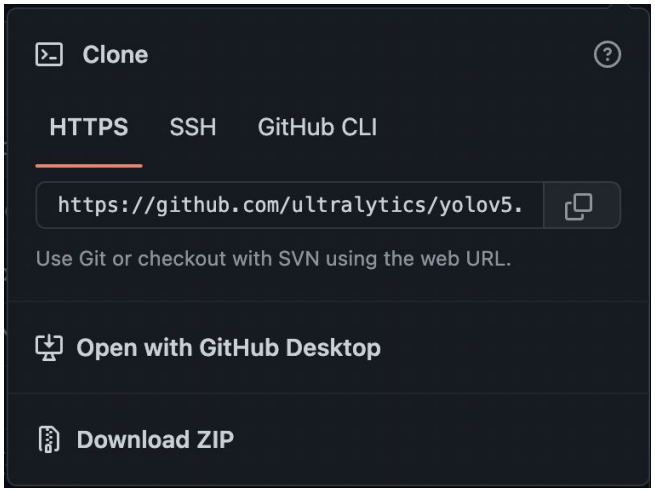
4. HOW TO TRAIN YOLOV5

❖ Step 1: clone YOLOv5 source code

Step 1: Click **Code**



Step 2: Copy link



Step 3: clone source code

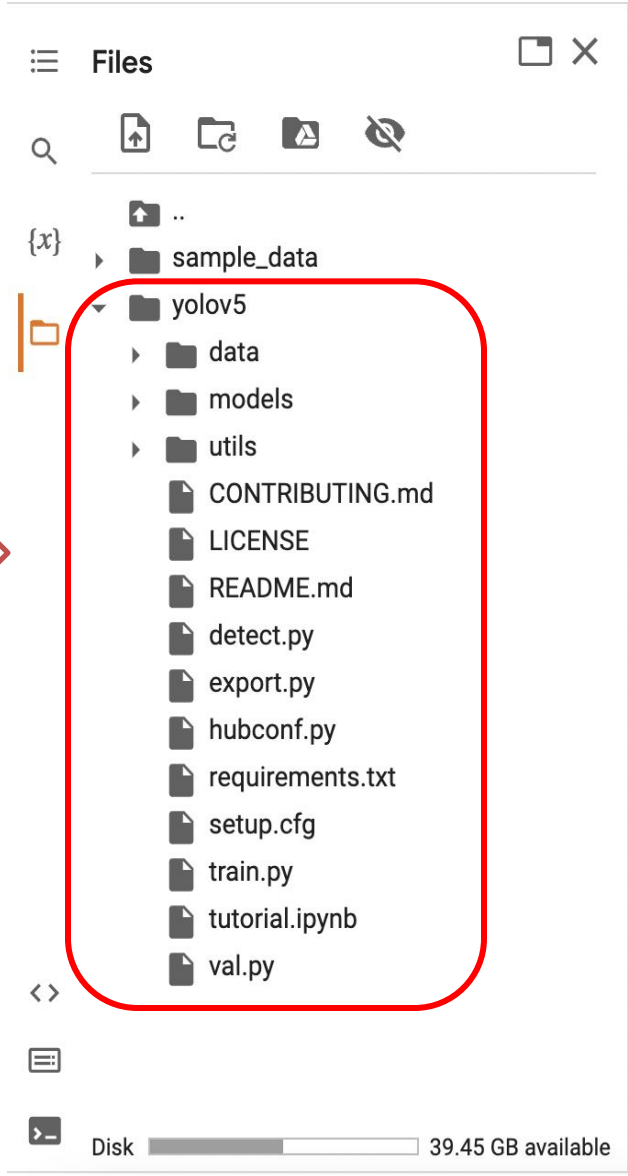
```
1 !git clone https://github.com/ultralytics/yolov5.git
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 12447, done.
remote: Total 12447 (delta 0), reused 0 (delta 0), pack-reused 12447
Receiving objects: 100% (12447/12447), 12.19 MiB | 29.24 MiB/s, done.
Resolving deltas: 100% (8591/8591), done.
```

4. HOW TO TRAIN YOLOV5

❖ Step 1: clone YOLOv5 source code

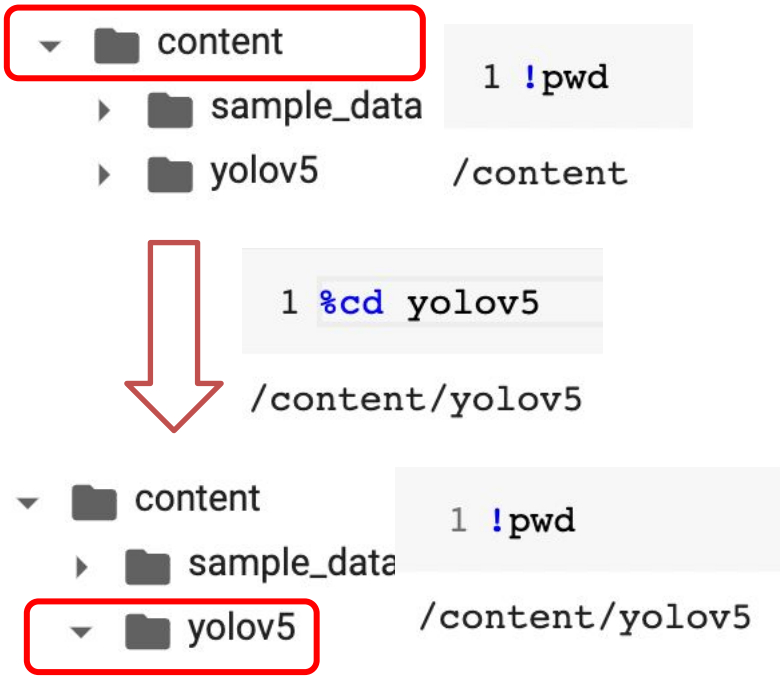
.github	Add `--hard-fail` argument to benchmarks for CI errors (#8513)	2 days ago
data	Dataset autodownload fstring updates	last month
models	Properly expose `batch_size` from OpenVINO similarly to TensorRT (...	2 days ago
utils	Fix `LoadImages()` with dataset YAML lists (#8517)	2 days ago
.gitattributes	git attrib	2 years ago
.gitignore	Ignore `*_openvino_model/` dir (#6180)	6 months ago
.pre-commit-config.yaml	[pre-commit.ci] pre-commit suggestions (#8470)	5 days ago
CONTRIBUTING.md	Add mdformat to precommit checks and update other version (#7529)	3 months ago
LICENSE	Add pre-commit CI actions (#4982)	9 months ago
README.md	Create README_cn.md (#8344)	14 days ago
detect.py	Allow detect.py to use video size for initial window size (#8330)	13 days ago
export.py	XML export `--half` fix (#8522)	2 days ago
hubconf.py	Prefer MPS over CPU if available (#8210)	23 days ago
requirements.txt	Exclude torch==1.12.0, torchvision==0.13.0 (Fix #8395) (#8497)	4 days ago
setup.cfg	Update setup.cfg to `description_file` field (#7248)	3 months ago
train.py	Training reproducibility improvements (#8213)	3 days ago
tutorial.ipynb	Update tutorial.ipynb (#8507)	3 days ago
val.py	Fix AP calculation bug #8464 (#8484)	3 days ago



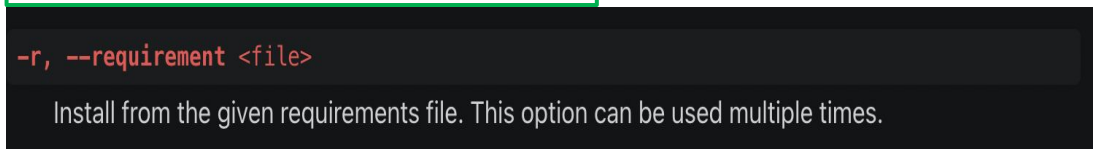
4. HOW TO TRAIN YOLOV5

❖ Step 2: download required packages

Step 1: Move to yolov5



Step 2: Install packages




```
1 !pip install -r requirements.txt -q
     |████████████████████████████████| 596 kB 30.7 MB/s
```

Full commands:

```
1 %cd yolov5
2 !pip install -r requirements.txt -q

/content/yolov5
     |████████████████████████████████| 596 kB 8.1 MB/s
```

4. HOW TO TRAIN YOLOV5

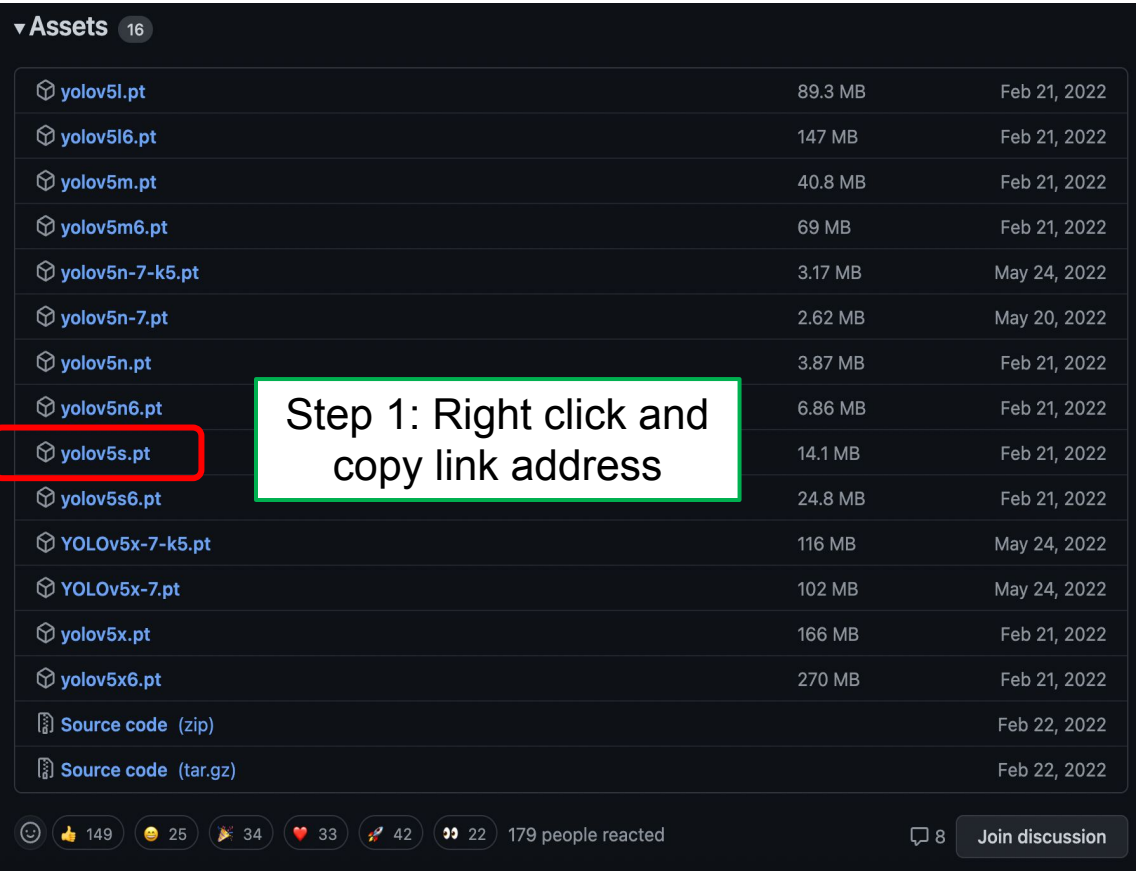
❖ Step 3: download pretrained model

Pretrained Checkpoints

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

4. HOW TO TRAIN YOLOV5

❖ Step 3: download pretrained model



Step 2: Download file using **wget** command

```
1 !wget https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt

--2022-07-10 06:14:50--  https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/
--2022-07-10 06:14:51--  https://objects.githubusercontent.com/github-production-release-asset-2e
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.1
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443.
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to: 'yolov5s.pt'

yolov5s.pt          100%[===================>]  14.12M  39.0MB/s    in 0.4s

2022-07-10 06:14:51 (39.0 MB/s) - 'yolov5s.pt' saved [14808437/14808437]
```

source: https://github.com/ultralytics/yolov5/releases

4. HOW TO TRAIN YOLOV5

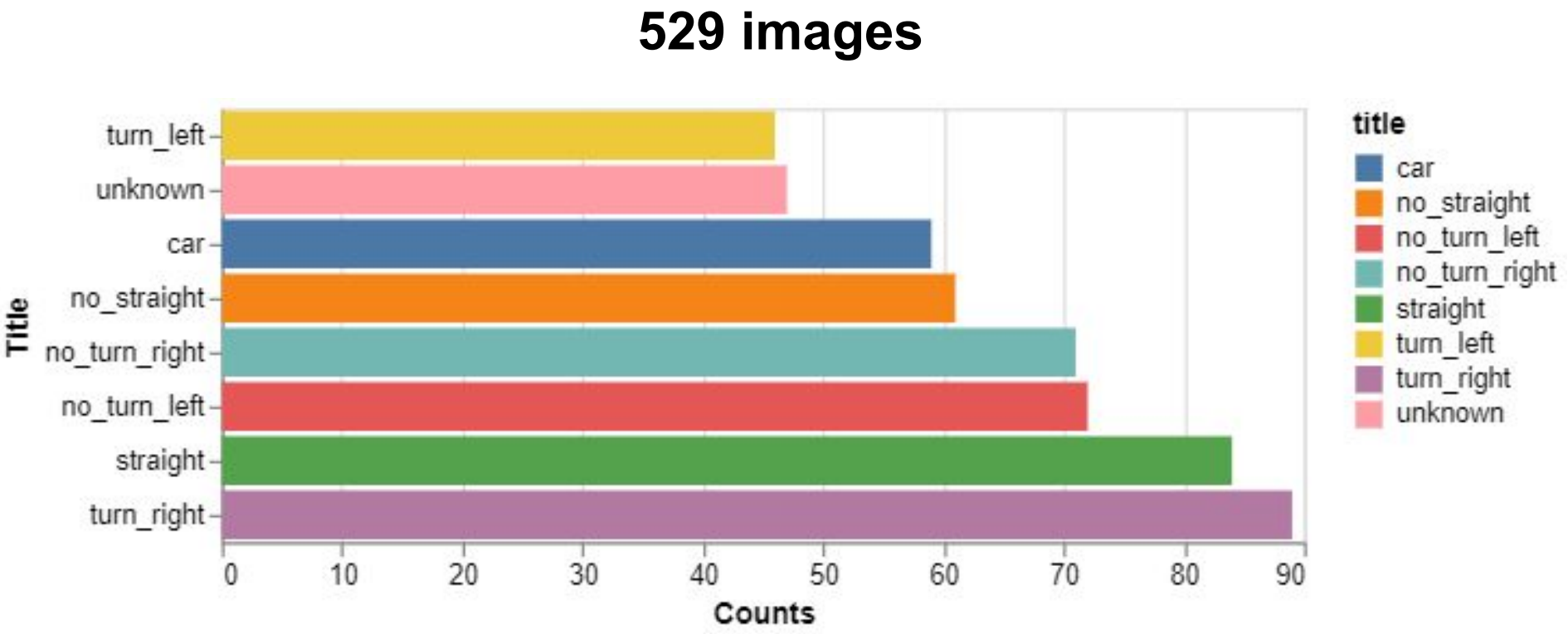
❖ Step 4: Prepare dataset

Bộ dữ liệu gồm:

- 'turn_right‘
- 'straight‘
- 'no_turn_left‘
- 'no_turn_right‘
- 'no_straight‘
- 'car‘
- 'unknown’
- 'turn_left'

4. HOW TO TRAIN YOLOV5

❖ Step 4: Visualize Data



4. HOW TO TRAIN YOLOV5

❖ Step 5: prepare .yaml file

```yaml
import yaml

dataset_info = {
  'path': '/content/train_dataset',
  'train': '/content/train_dataset/train/images',
  'val': '/content/train_dataset/val/images',
  'nc': 8,
  'names': ['turn_right', 'straight', 'no_turn_left', 'no_turn_right', 'no_straight', 'car', 'unknown', 'turn_left']
}

with open('data/sign.yaml', 'w') as f:
  doc = yaml.dump(dataset_info, f, default_flow_style=None, sort_keys=False)
```

Path to dataset root folder

Path to train images folder

Classes's name

Number of classes to detect

Path to val images folder

4. HOW TO TRAIN YOLOV5

❖ Step 5: prepare .yaml file

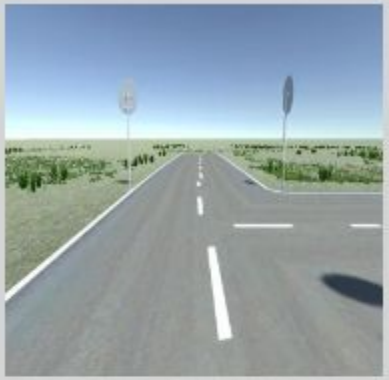
Các Kỹ thuật tăng cường dữ liệu YOLOv5

```
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

4. HOW TO TRAIN YOLOV5

Các kỹ thuật tăng cường dữ liệu YOLOv5

HUE



original

-25°

25°

SATURATION

original

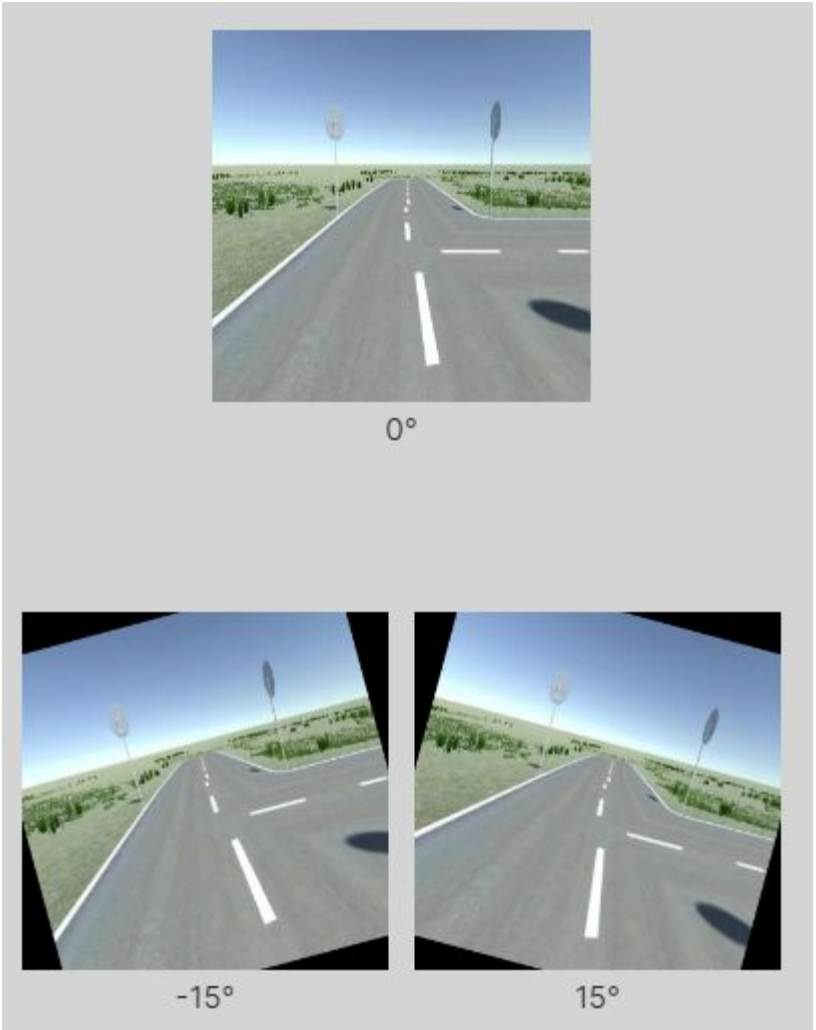
-25%

25%

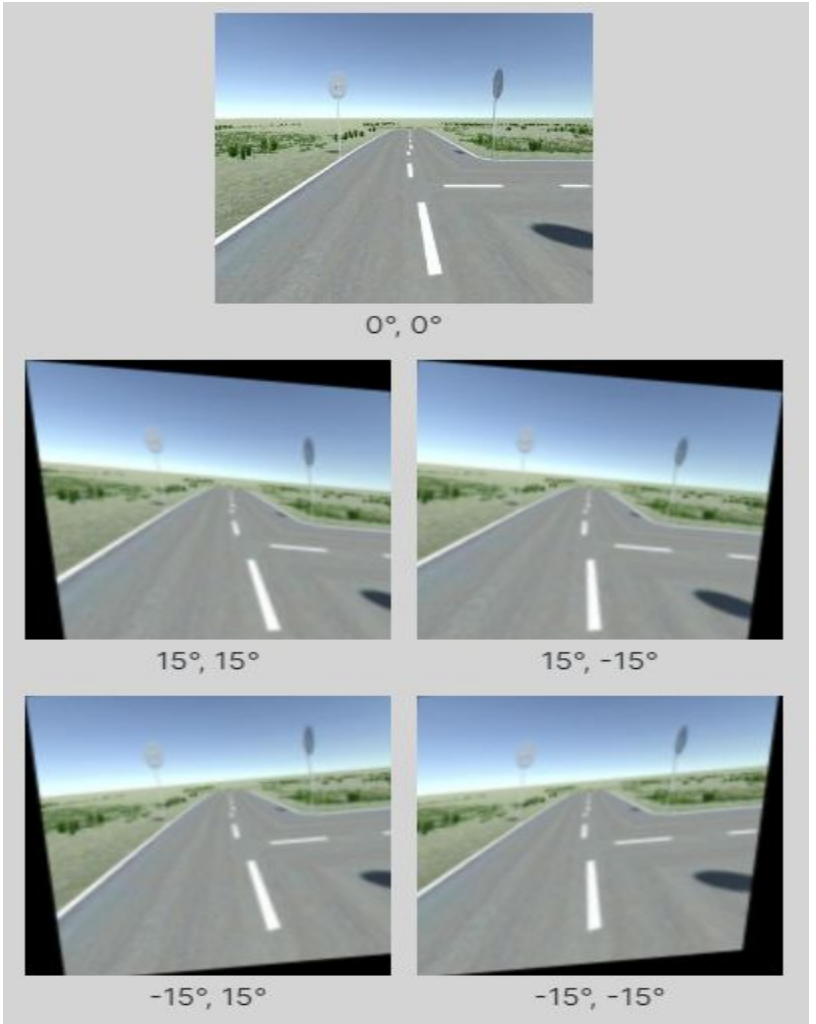
4. HOW TO TRAIN YOLOV5

Các kỹ thuật tăng cường dữ liệu YOLOv5

Rotation



Shear



4. HOW TO TRAIN YOLOV5

Các kỹ thuật tăng cường dữ liệu YOLOv5

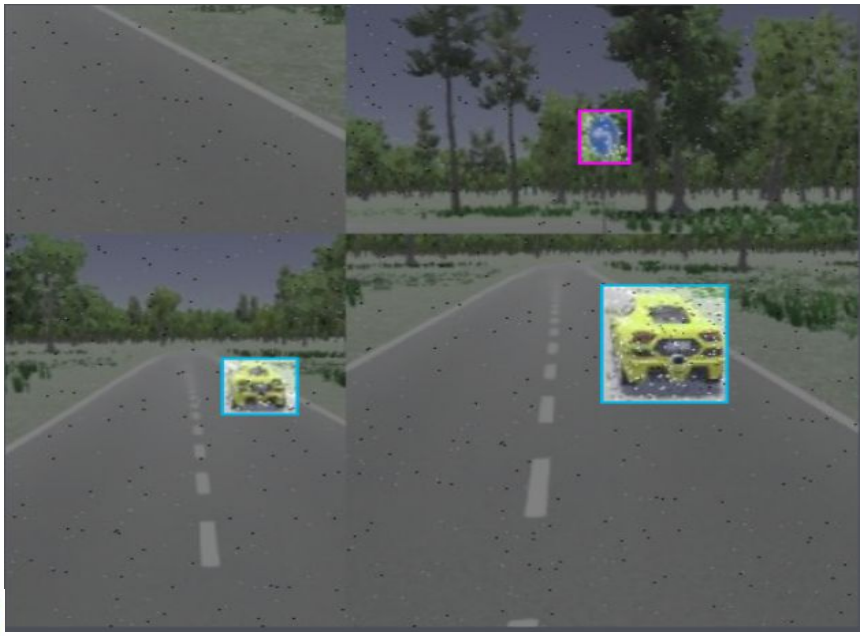
Flip

Original

Vertical

Horizontal

Mosaic



Mixup

* 0.5 +

* 0.5 =

Cat

Dog

50%: Cat, 50%: Dog

4. HOW TO TRAIN YOLOV5

Các kỹ thuật tăng cường dữ liệu YOLOv5

	xmin	ymin	xmax	ymax	confidence	class
0	484.594696	27.074352	548.712097	92.111092	0.905416	8
1	212.767197	86.766312	233.222260	117.577248	0.819874	7
2	486.016510	27.753481	547.728210	92.036285	0.677914	1

	name
0	turn_left
1	unknown
2	turn_right

4. HOW TO TRAIN YOLOV5

❖ Step 5: prepare .yaml file

Link Download:
https://drive.google.com/file/d/1qSssMxlTehct-_6glLDpSM9dRn__4TyS/view?usp=share_link

```
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.0  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

4. HOW TO TRAIN YOLOV5

❖ Step 6: Training

Training command

```
!python train.py --img 640 --batch 16 --epochs 10 --data helmet.yaml --weights yolov5s.pt --cache
```

```
train: weights=yolov5s.pt, cfg=, data=helmet.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=10, batch_size=16,
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8,
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 🚀 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 26.7MB/s]
Overriding model.yaml nc=80 with nc=3
```

```
      Epoch    gpu_mem       box       obj       cls    labels  img_size
        9/9      4.64G   0.03089   0.03128  0.002934       141       640: 100% 250/250 [01:03<00:00,  3.91it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 32/32 [00:09<00:00,
                   all       1000       5104      0.947      0.597      0.628      0.403
```

Log when training is end

```
10 epochs completed in 0.206 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.5MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 32/32 [00:13<00:00,
                   all       1000       5104      0.947      0.597      0.628      0.403
                helmet       1000       3762       0.94      0.904      0.961      0.619
                  head       1000       1201        0.9      0.888      0.913      0.586
                person       1000        141          1          0     0.0101    0.00452
Results saved to runs/train/exp
```

4. HOW TO TRAIN YOLOV5

❖ Step 6: Resume (Optional)

Resume command

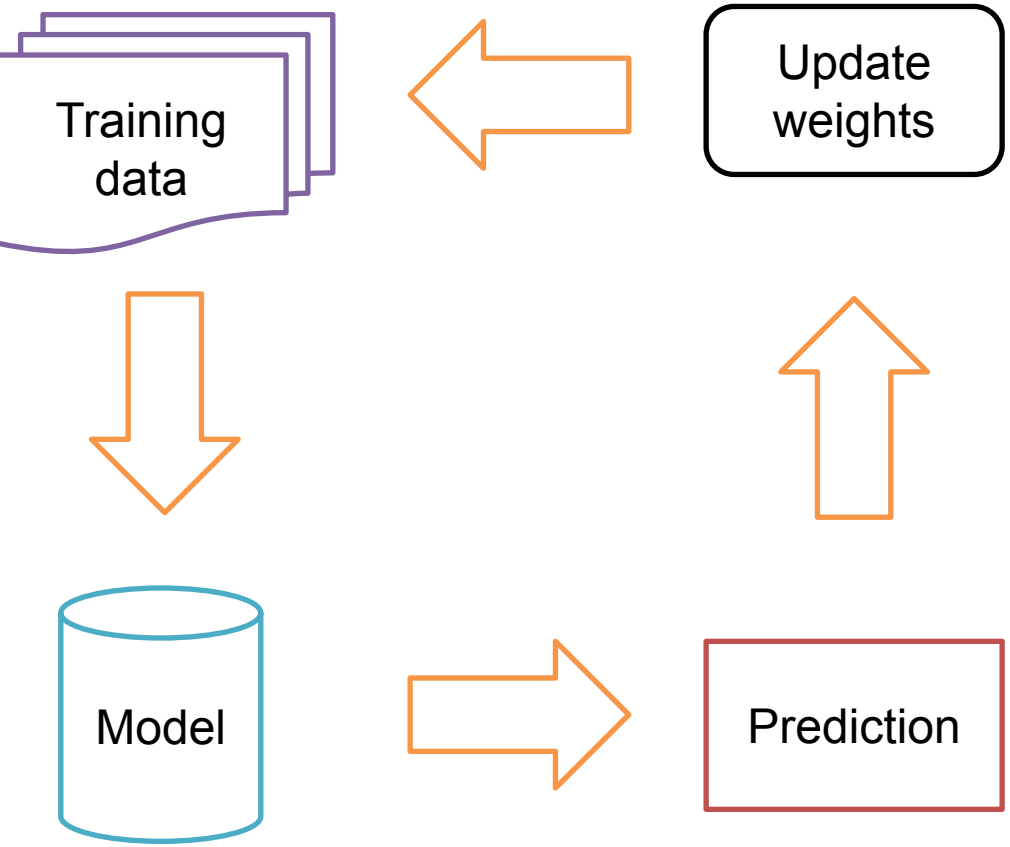
```
!python train.py --resume path/to/last.pt --cache
```

Log when Resume

```
Transferred 349/349 items from /content/drive/MyDrive/UTE2023/train/exp3/weights/
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 60 weight(dec
Resuming training from /content/drive/MyDrive/UTE2023/train/exp3/weights/last.pt
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3
train: Scanning /content/train_dataset/train/labels.cache... 370 images, 0 backgr
train: Caching images (0.4GB ram): 100% 370/370 [00:01<00:00, 278.17it/s]
val: Scanning /content/train_dataset/val/labels.cache... 106 images, 0 backgroun
val: Caching images (0.1GB ram): 100% 106/106 [00:01<00:00, 87.49it/s]
Plotting labels to /content/drive/MyDrive/UTE2023/train/exp3/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to /content/drive/MyDrive/UTE2023/train/exp3
Starting training for 300 epochs...
```

4. HOW TO TRAIN YOLOV5

❖ Training command flags



Flag	Meaning
img	The size of training image
batch	Number of data samples to read in 1 epoch
epochs	Number of iteration (number of times model ”see” training data)
data	Path to .yaml file
weights	Path to pretrained model
hyp	hyperparameters path
project	save to project/name
resume	resume most recent training

4. HOW TO TRAIN YOLOV5

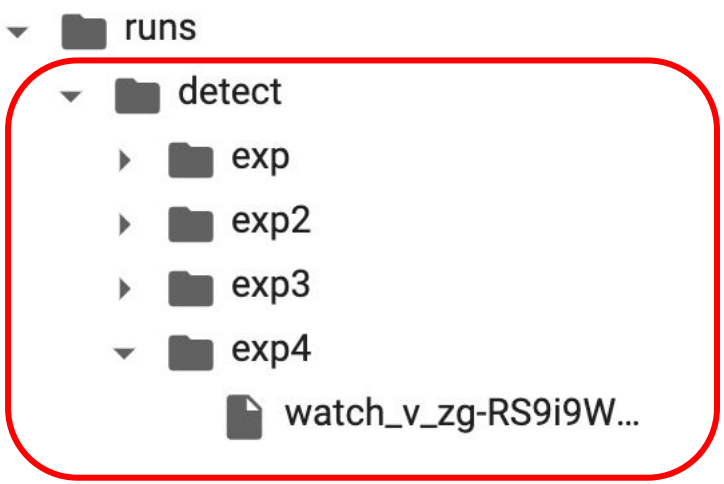
❖ Step 7: inference

Inference command

```
1 !python detect.py --weights runs/train/exp/weights/best.pt --source /content/yolov5/working.png
```

```
detect: weights=['runs/train/exp/weights/best.pt'], source=/content/yolov5/working.png, data=data/coco128.yaml,
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)
```

Saved



4. HOW TO TRAIN YOLOV5

❖ Step 7: inference

Input source	Command
Single image	`!python detect.py --weights "weight.pt" --source "image.jpg"`
Folder of images	`!python detect.py --weights "weight.pt" --source "root/images"`
Video	`!python detect.py --weights "weight.pt" --source "video.mp4"`
Youtube link	`!python detect.py --weights "weight.pt" --source "https://youtube.com"`
Webcam	`!python detect.py --weights "weight.pt" --source 0`

model = torch.hub.load('ultralytics/yolov5', 'custom', path='path/to/best.pt') # local model

4. HOW TO TRAIN YOLOV5

❖ Step 7: inference

Load YOLOv5 from PyTorch Hub
(Recommend)

```
1 model = torch.hub.load('ultralytics/yolov5', 'custom', path='path/to/best.pt', device=0)  # local model
2
3 #### Inference Settings ####
4 model.conf = 0.90
5
6 #### Inference ####
7 img = '/content/train_dataset/test/images/Anh_Bien_Nguoc_181_png_jpg.rf.e1646e4bee8504d057cab93fb0707d84.jpg'
8 results = model(img)
9
10 #### Results ####
11 print(results.xyxy[0])
```

```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀 v7.0-4-g7398d2d Python-3.7.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
Model summary: 157 layers, 7031701 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
tensor([[100.87380,  59.78672, 135.83234, 121.57002,   0.94173,   6.00000]], device='cuda:0')
```

https://github.com/ultralytics/yolov5/issues/36

4. HOW TO TRAIN YOLOV5

❖ **Step 7: inference**

Inference Settings

YOLOv5 models contain various inference attributes such as **confidence threshold**, **IoU threshold**, etc. which can be set by:

```
model.conf = 0.25  # NMS confidence threshold
      iou = 0.45  # NMS IoU threshold
      agnostic = False  # NMS class-agnostic
      multi_label = False  # NMS multiple labels per box
      classes = None  # (optional list) filter by class, i.e. = [0, 15, 16] for COCO persons, cats and dogs
      max_det = 1000  # maximum number of detections per image
      amp = False  # Automatic Mixed Precision (AMP) inference

results = model(im, size=320)  # custom inference size
```

https://github.com/ultralytics/yolov5/issues/36

4. HOW TO TRAIN YOLOV5

- **Step 7: inference**

```
1 results.pandas().xyxy[0]
```

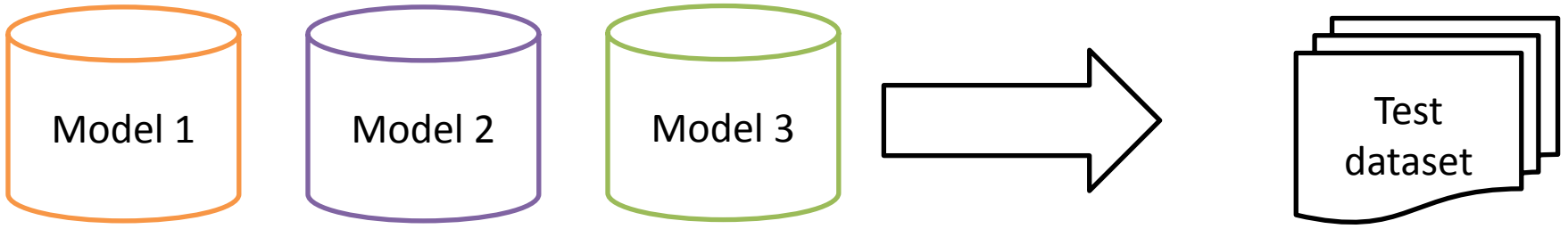
	xmin	ymin	xmax	ymax	confidence	class	name
0	100.873802	59.786716	135.832336	121.570023	0.941726	6	unknown

4. HOW TO TRAIN YOLOV5

❖ Evaluate model

Which one is better?

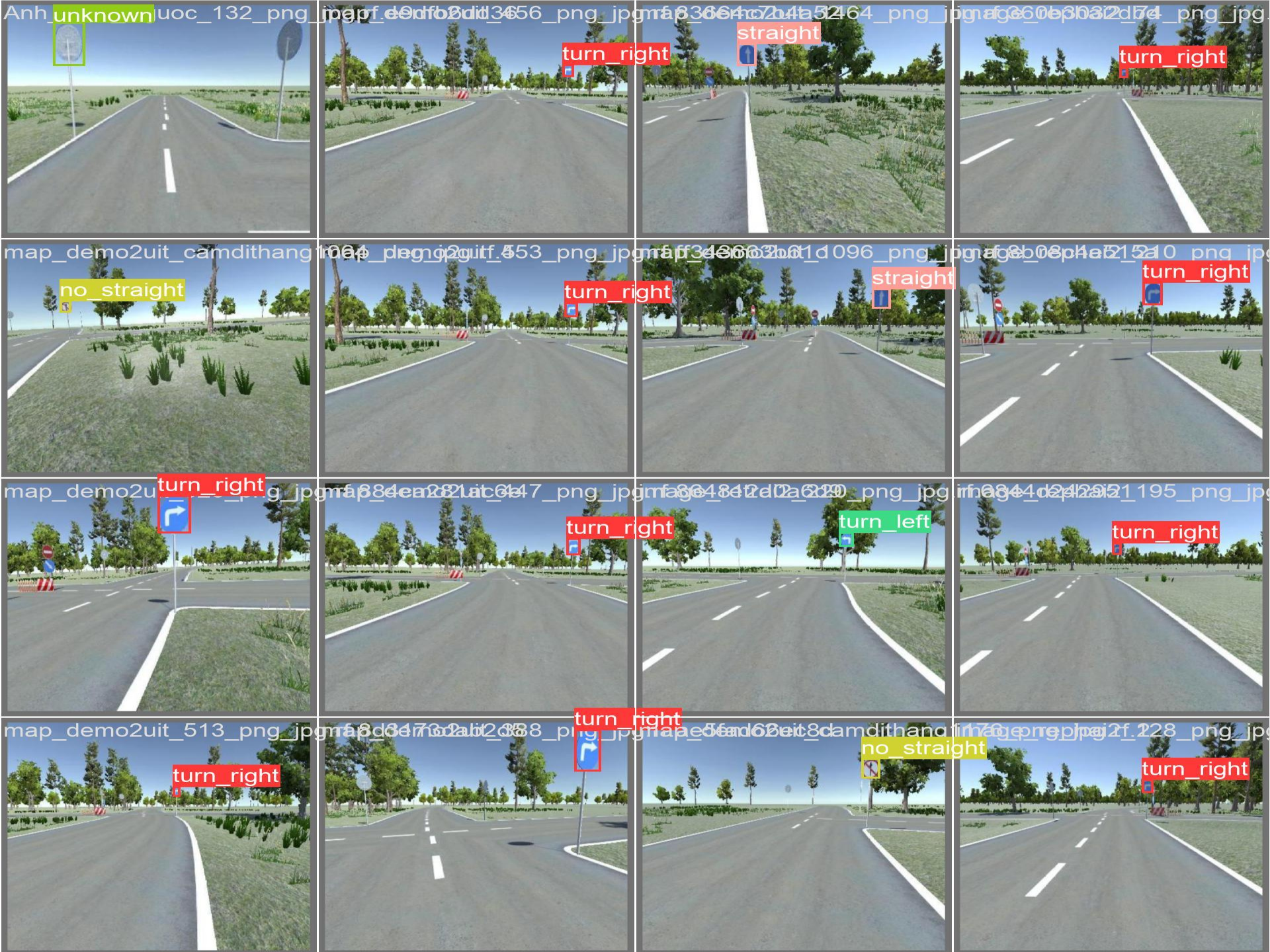
Evaluate on new dataset (test dataset)




```
1 !python val.py --weights runs/train/exp2/weights/best.pt --data helmet.yaml --img 640 --iou 0.65 --half
```

```
val: data=/content/yolov5/data/helmet.yaml, weights=['runs/train/exp2/weights/best.pt'], batch_size=32, imgsz=640,
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)

Fusing layers...
Model summary: 213 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning '/content/yolov5/helmet/val/labels.cache' images and labels... 1000 found, 0 missing, 0 empty, 0 corr
               Class      Images      Labels           P           R      mAP@.5 mAP@.5:.95: 100% 32/32 [00:17<00:00,  1
                 all        1000        5104       0.943       0.594       0.626       0.408
              helmet        1000        3762       0.943       0.896       0.959       0.624
                head        1000        1201       0.884       0.885        0.91       0.596
              person        1000         141           1           0     0.00967     0.00441
Speed: 0.3ms pre-process, 4.1ms inference, 1.8ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp4
```


unknown
turn_right
straight
turn_right
no_straight
turn_right
straight
turn_right
turn_right
turn_right
turn_left
turn_right
turn_right
turn_right
no_straight
turn_right


unknown 0.9
turn_right 1.0
straight 0.9
turn_right 0.9
no_straight 0.9
turn_right 0.9
straight 0.9
turn_right 1.0
turn_right 1.0
turn_right 1.0
turn_left 0.9
turn_right 0.9
turn_right 0.9
turn_right 1.0
no_straight 0.9

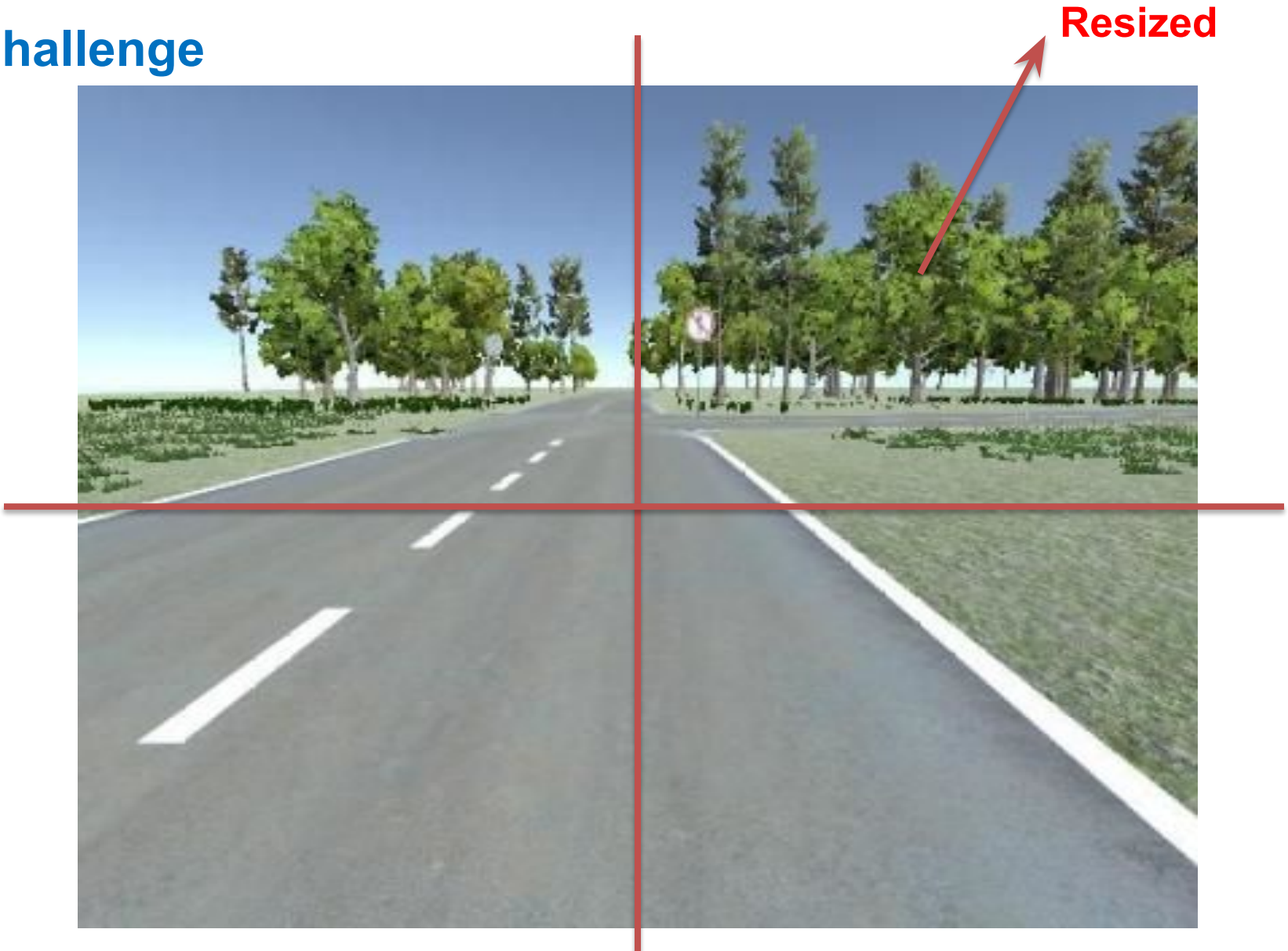
4. HOW TO TRAIN YOLOV5

❖ **Challenge**

Small Object

4. HOW TO TRAIN YOLOV5

- **Challenge**



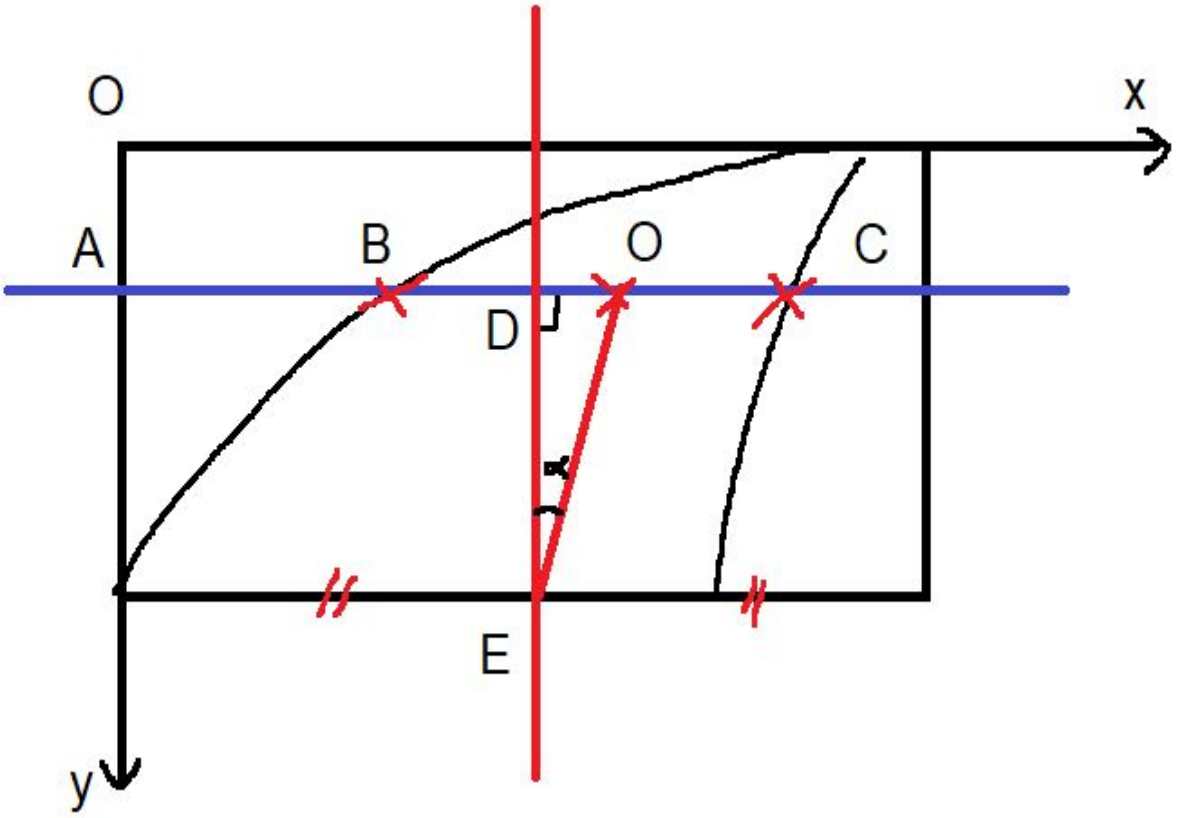
4. HOW TO TRAIN YOLOV5

- **Challenge** Model đã rất dễ nhầm lẫn các loại biển báo như: trái-phải-đi thẳng, cấm trái-cấm phải.

Thêm class None để tránh tình trạng model cố dự đoán ảnh ROI lỗi (xe, người, đường, cây cối,..)

Lý thuyết:
https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2?fbclid=IwAR2JPWGycM8hPTiwNjluWbbbGKgGv66lH8tyfQcMRpgn8dLXnXFA6AqTHow
Model:

5. CONTROL CAR BY CALCULATE ANGLE



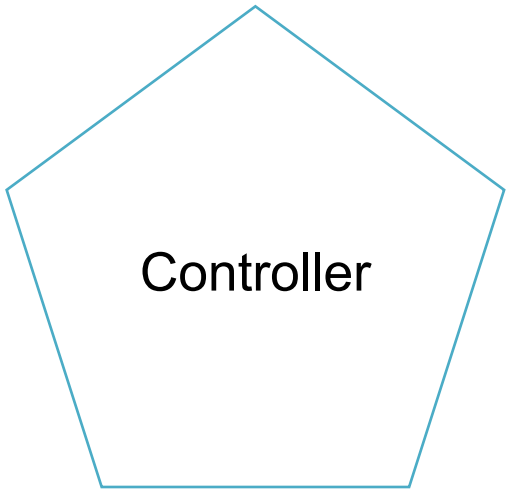
Ta có:
* AD = Width/2
* OA = 30 => DE = Height – OA

* Từ làn đường nhận diện được, từ đó ta tìm được AB và AC
=> AO = (AB + AC)/2
=> DO = AO - AD

* Alpha = tan(DO/DE)

https://www.youtube.com/watch?v=uDkqiuMYZm8&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H&index=1

6. CONTROL CAR BY PID CONTROLLER



Angle

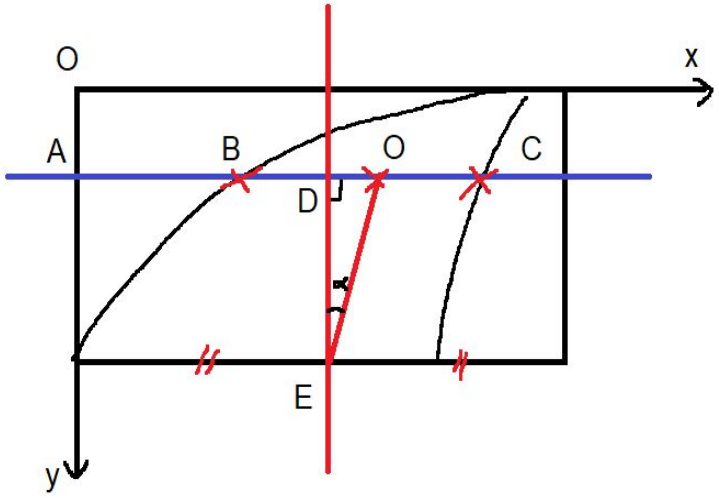
Model Predictive Control

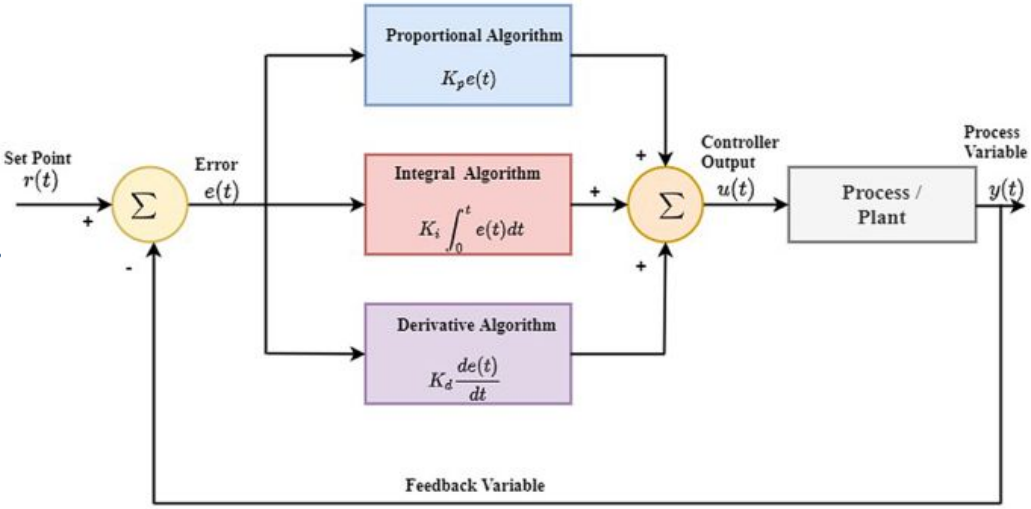
PID

Fuzzy

- PID cho phép vọt lố => Rất tốt cho trường hợp cần rẽ gấp
- Dễ thực hiện
- Dễ điều chỉnh

6. CONTROL CAR BY PID CONTROLLER





Set_Point = Width/2

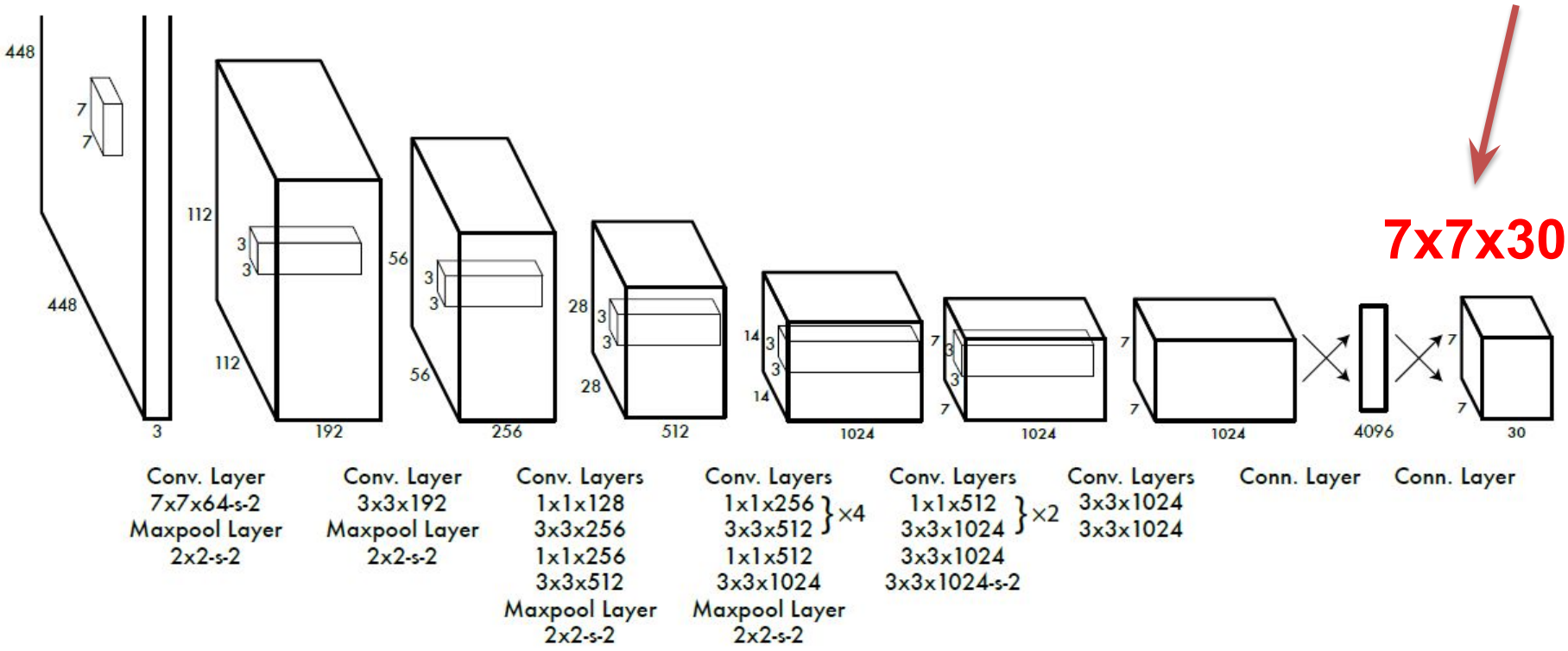
Current_Point = AO

⇒ Error = Set_Point - Current_Point

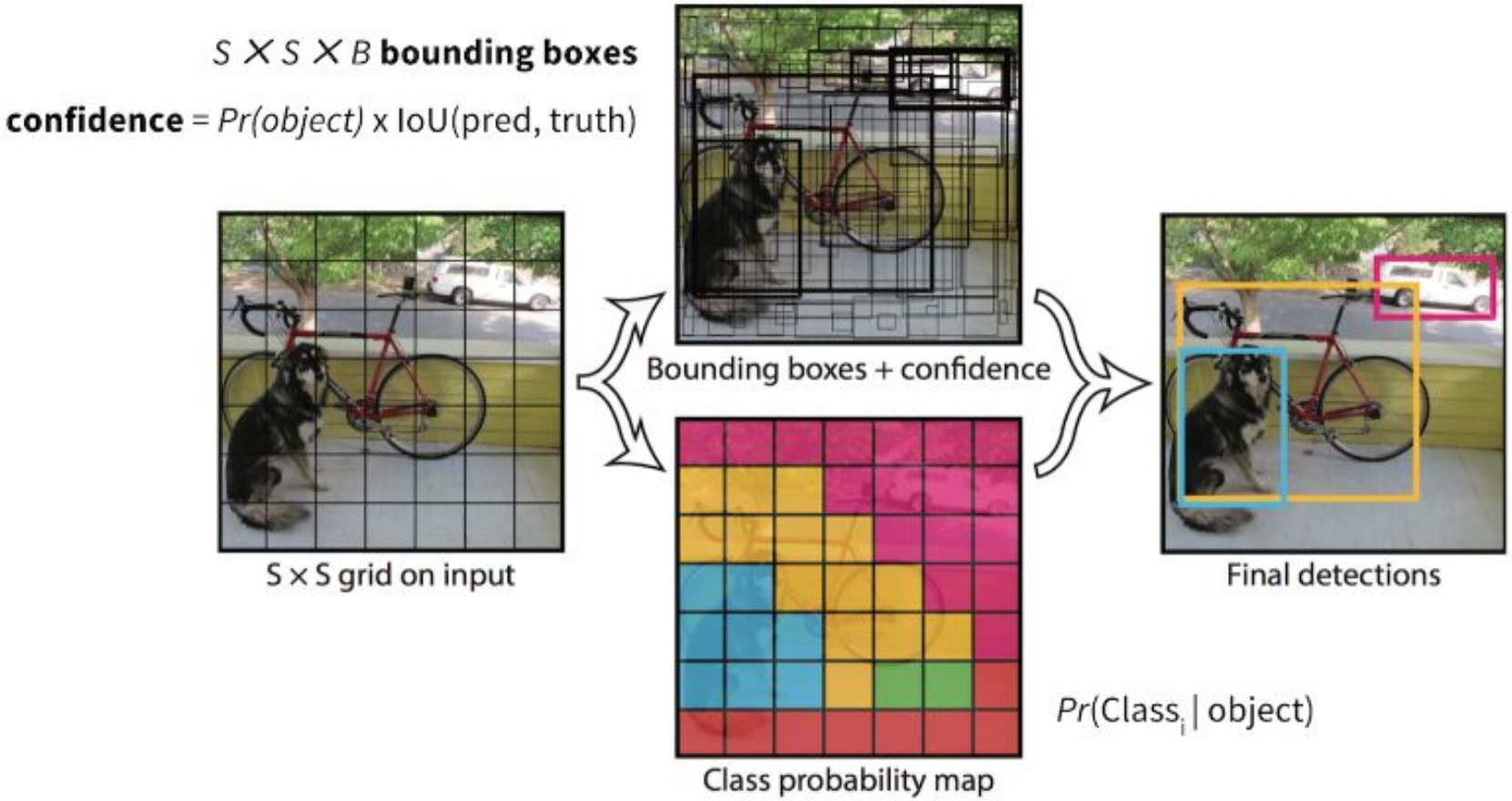
⇒ Angle = PID(Error)

7. YOLOV1 (OPTIONAL)

Model YoloV1



7. YOLOV1 (OPTIONAL)



7. YOLOV1 (OPTIONAL)

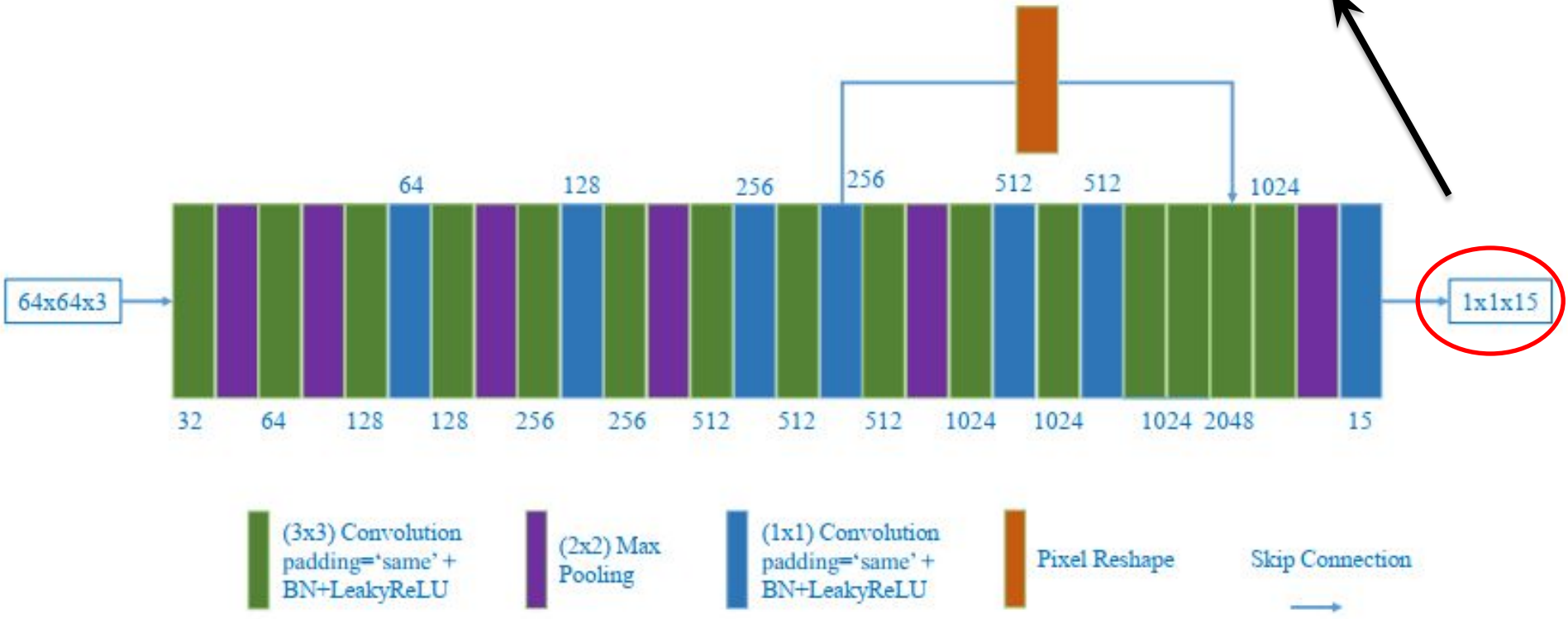
B1: Chia ảnh **448x448** thành **7x7** ảnh => được **7x7** ảnh **64x64**

B2: **Giả sử** ảnh 64x64 chỉ có 2 trường hợp:

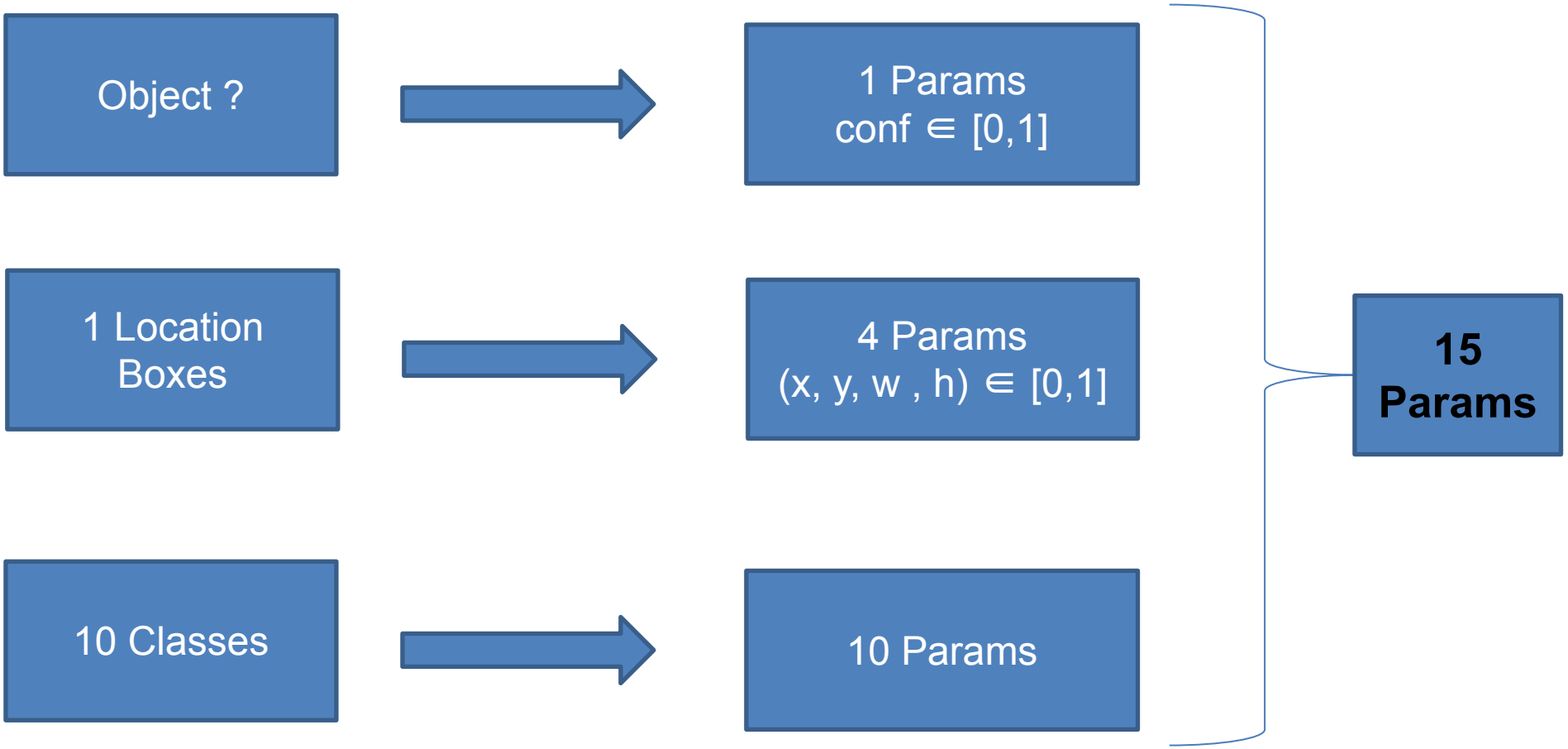
TH1: Có Object **(conf = 1)**

TH2: Không có Object **(conf = 0)**

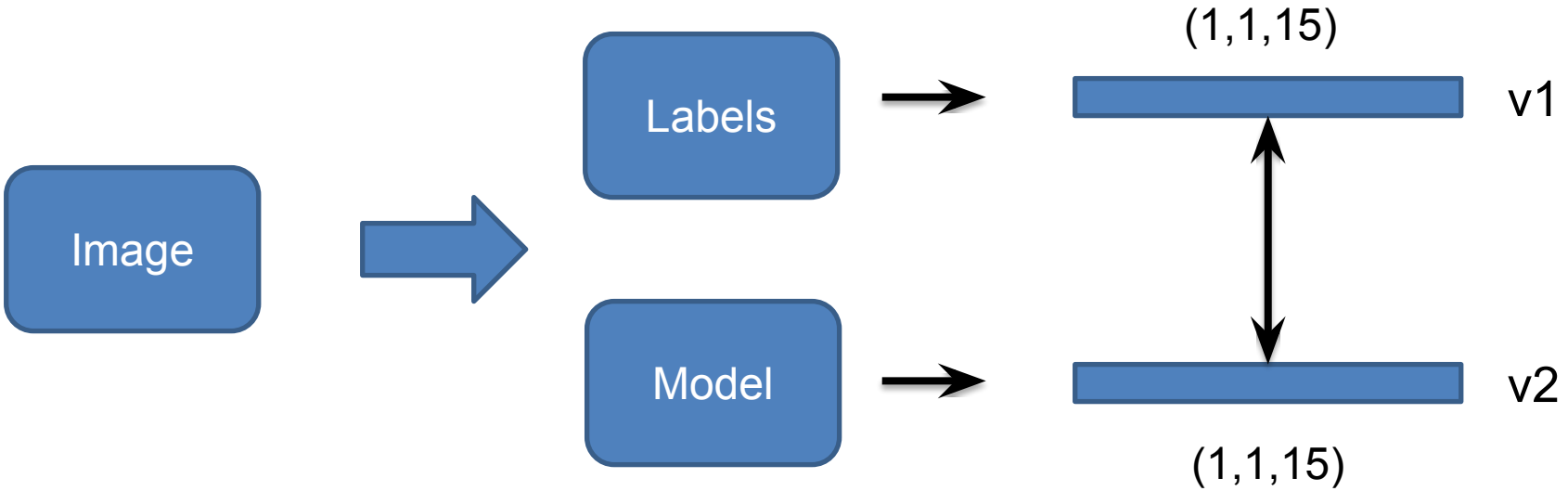
B3: Thực hiện trên tập MNIST chỉ có **10** class



7. YOLOV1 (OPTIONAL)



7. YOLOV1 (OPTIONAL)



Loss = (v2-v1)^2

7. YOLOV1 (OPTIONAL)

$$L_{loc} = 1^{obj} * [(x - \hat{x})^2 + (y - \hat{y})^2] + 1^{obj} * [(\sqrt{w} - \sqrt{\hat{w}})^2 + (\sqrt{h} - \sqrt{\hat{h}})^2]$$

$$L_{obj} = 1^{obj} * (C - \hat{C})^2, with\ C = 1$$
$$L_{no-obj} = 1^{no-obj} * (C - \hat{C})^2, with\ C = 0$$

$$L_{class} = 1^{obj} * \sum_{c=0}^{\#classes} (p(c) - \hat{p}(c))^2$$

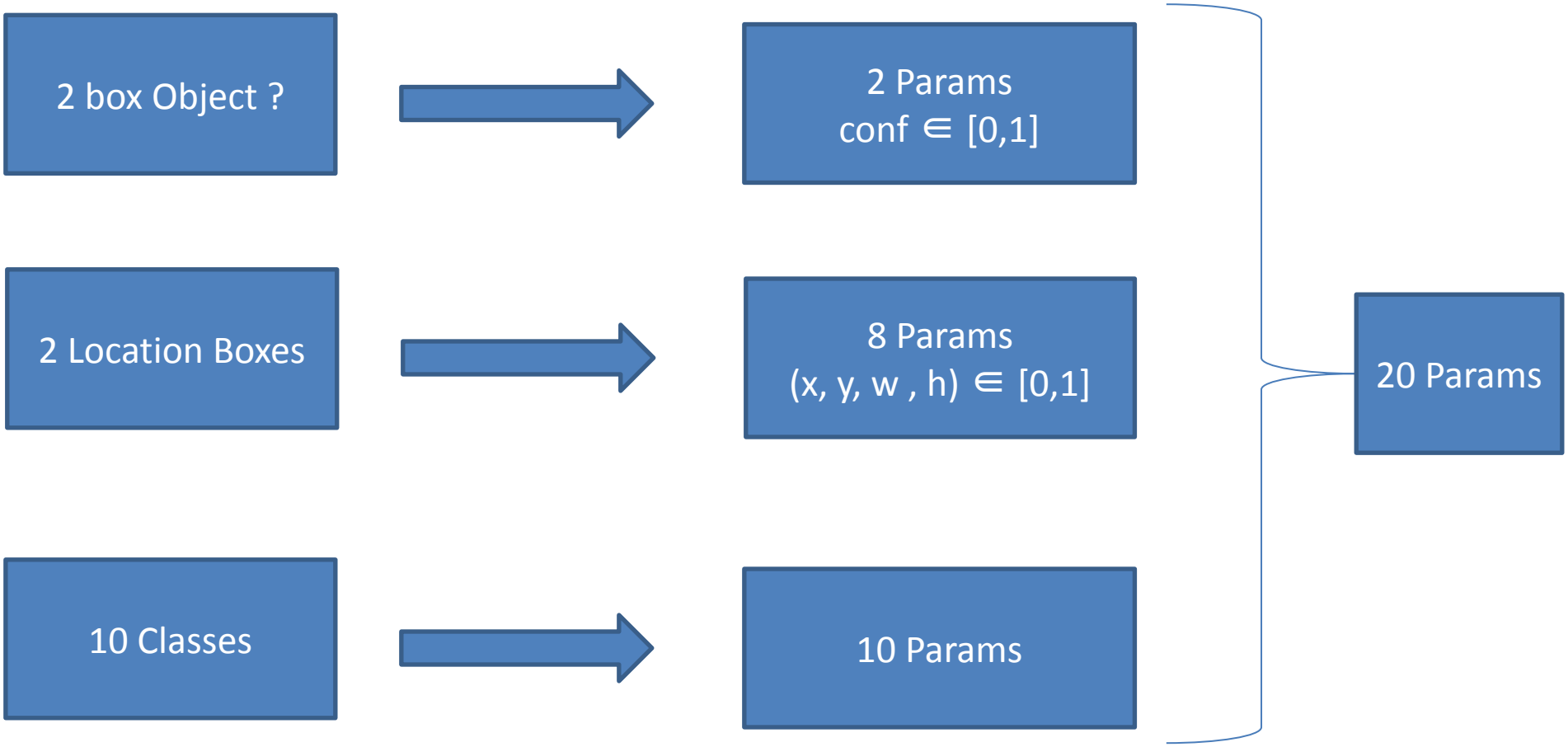
$$L = \lambda_{loc} * L_{loc} + \lambda_{obj} * L_{obj} + \lambda_{no-obj} * L_{no-obj} + \lambda_{class} * L_{class}$$

1^{obj}: Cell có chứa object (dựa vào target)

x,y: center của bbox
w,h: width và heigh của bbox
C: 0 không có object, 1 có object
$\hat{C}$: xác suất có object do model predict
p(c): xác suất của class
$\widehat{p(c)}$: xác suất của class do model preidct

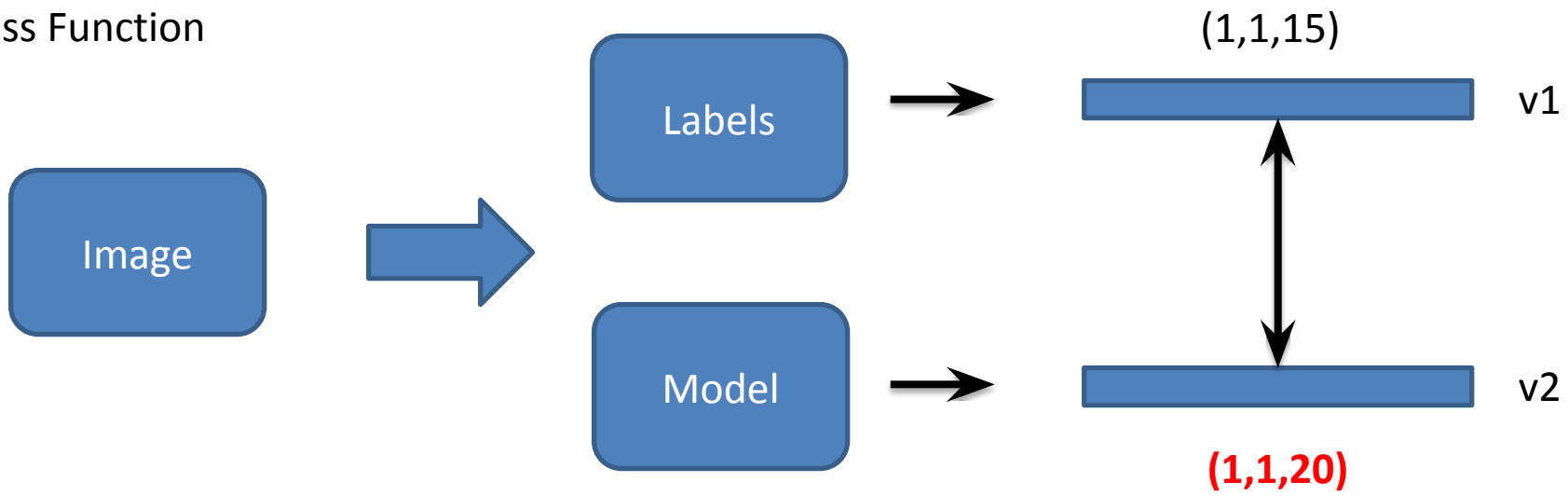
7. YOLOV1 (OPTIONAL)

How to Improve ???



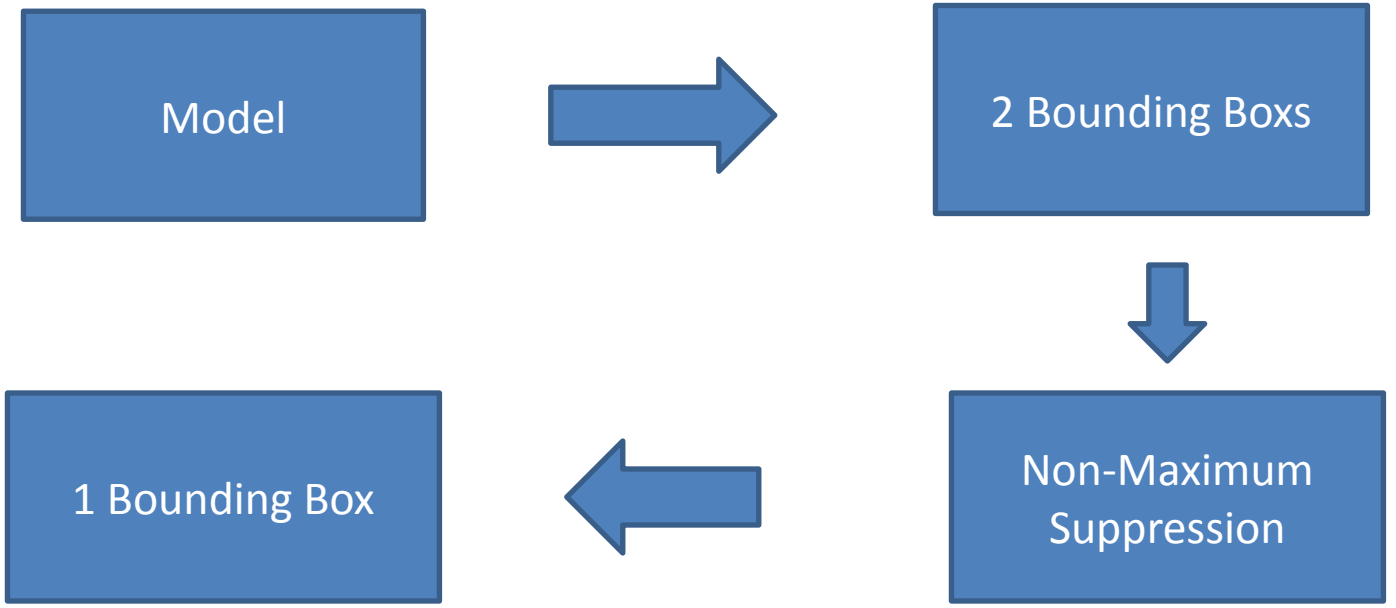
7. YOLOV1 (OPTIONAL)

Loss Function



Loss = (v2-v1)^2

7. YOLOV1 (OPTIONAL)



Non-Maximum Suppression:

B1: Có [bbox1, bbox2]

B2: Remove những đối tượng có confidence C < C-threshold (ngưỡng này chúng ta có thể re setup).

B3: Sắp xếp confidence score giảm dần

B4: Lấy những bbox có confidence cao nhất.

B5: Remove những box có IoU < threshold

B6: Quay lại bước B4

7. YOLOV1 (OPTIONAL)

Location loss

$$\mathcal{L}_{\text{loc}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Object loss

$$\mathcal{L}_{\text{obj}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_{ij} - \hat{C}_{ij})^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_{ij} - \hat{C}_{ij})^2$$

Classification loss

$$\mathcal{L}_{\text{cls}} = \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

Loss

$$L = \lambda_{loc} * L_{loc} + \lambda_{obj} * L_{obj} + \lambda_{no-obj} * L_{no-obj} + \lambda_{class} * L_{class}$$

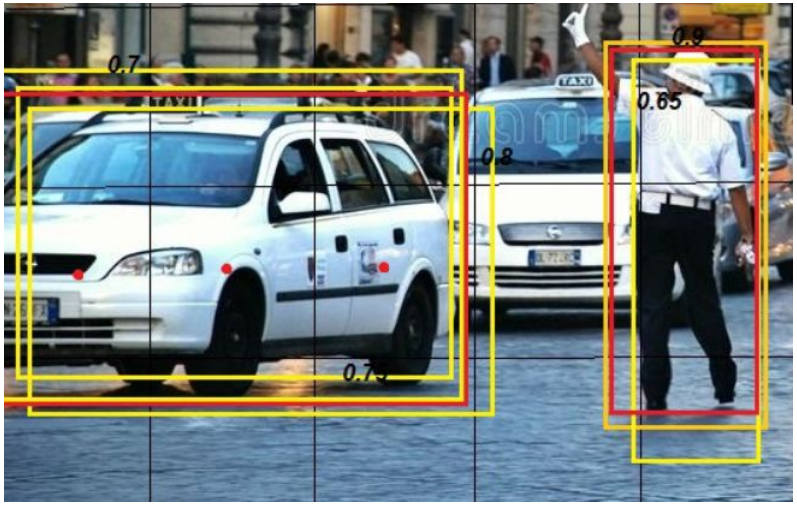
7. YOLOV1 (OPTIONAL)

❖ YOLOV1

- Với mỗi ô lưới thì tối đa sẽ chỉ phát hiện được 1 đối tượng
- Phát hiện được tối đa 49 OBJECT(nếu thuật toán chia ảnh thành 7×7).
- Model chỉ phát hiện được một đối tượng có độ chính xác cao nhất trong một ô lưới . (hình 1)
- Nếu vật thể nào đó quá lớn so với bức ảnh, đồng nghĩa với việc vật thể đó sẽ có nhiều trọng tâm ở nhiều ô lưới khác nhau. Vậy nên vật thể đó sẽ được phát hiện nhiều lần. (hình 2) => Giải quyết bằng **Non-Maximum Supression**

- Hạn chế về việc xác định vị trí của vật thể
- Giá trị recall luôn thấp

Hình 1



Hình 2