

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA CÔNG NGHỆ THÔNG TIN**  
**BỘ MÔN NHẬP MÔN HỌC MÁY**

-----oOo-----



**BÁO CÁO**  
**ĐỒ ÁN CUỐI KÌ**

**GV-LT. Ngô Minh Nhựt**  
**GV-TH. Lê Long Quốc**

**Nhóm thực hiện:**  
**Phạm Duy Tiến (19127577)**  
**Trần Nam Khánh (19127441)**

**TP. HỒ CHÍ MINH, NGÀY 03 THÁNG 01 NĂM 2022**

# Mục lục

<b>Ứng dụng web áp dụng yolo có sẵn</b>	<b>3</b>
Front-end:	3
Back-end:	6
<b>Huấn luyện yolo và ứng dụng vào web</b>	<b>9</b>
Các bước train mô hình YOLO:	9
Bước 1. Tải source Darknet về máy.	9
Bước 2. Chuẩn bị dữ liệu train:	10
Bước 3. Chuẩn bị và cấu hình các file cần thiết cho quá trình train dữ liệu.	13
Bước 4: Upload file darknet.zip và tạo folder backup chứa các file weights khi train trên colab.	16
Đánh giá F1- core epoch = 4000	16
<b>Tài liệu tham khảo:</b>	<b>17</b>

## I. Ứng dụng web áp dụng yolo có sẵn

### 1. Front-end:

---

#### Object Detector

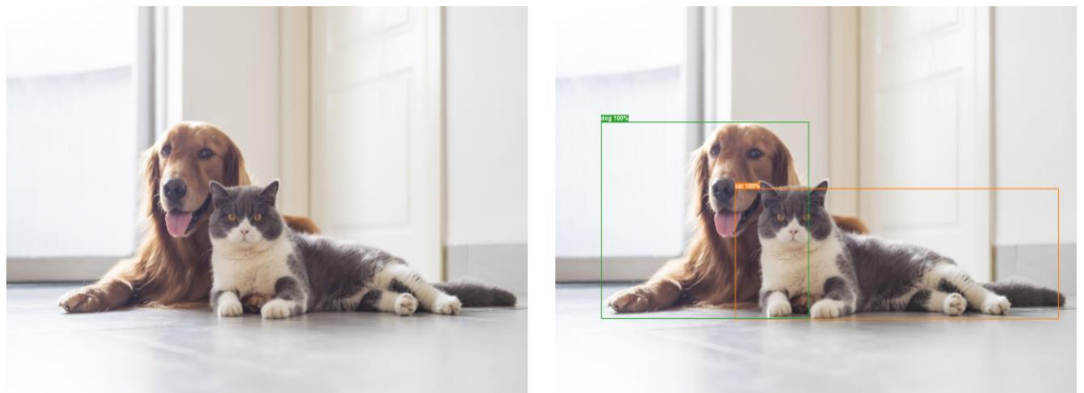
Upload Image

---

KHTN, 2021

#### Object Detector

Reset



KHTN, 2021

### Phần code chính:

```
// Handle event upload button when user press upload button
upload.addEventListener('change', function(e) {
    preview.innerHTML = '';
    var reader= new FileReader();
    reader.onload=function(event){
        if(event.target.result){
            img.onload=onload_func; //call function to pre-process input image
            img.src=event.target.result;
        }
    };
    reader.readAsDataURL(event.target.files[0]);
})
```

Hàm này sẽ handle event mỗi khi user nhấn vào upload button. Khi user upload ảnh, hàm sẽ gọi 1 hàm khác là `onload_func` để pre-process ảnh input trước khi hiện lên UI của user.

```
// Pre process input image then send it to server
function onload_func() {
    // extracting the orientation info from EXIF
    EXIF.getData(img, function () {
        orientation = EXIF.getTag(this, 'Orientation');
        console.log(orientation);
    });
    // resize the sides of the canvas and draw the resized image
    [canvas.width, canvas.height] = reduceSize(img.width, img.height, MAX_SIDE_LEN);
    context.drawImage(img, 0, 0, canvas.width, canvas.height);
    // adds the image that the canvas holds to the source
    resized_img.src = canvas.toDataURL('image/jpeg');
    // clean the result to show before doing anything
    preview.innerHTML = '';
    // append new image
    preview.appendChild(resized_img);
    // send the user image on server and wait for response, and, then, shows the result
    sendtoserver();
}
```

`Onload_func` sẽ lấy thông tin orientation để gửi đến server ( thông tin này để kiểm tra ảnh có bị xoay hay không). Sau đó ảnh sẽ được resize để hiện lên UI của user. Cuối cùng là send ảnh đó đến server.

```

// show to client some processing button then send image to server
function sendtoserver() {
    var element = document.getElementById('upload');
    // disable upload img
    element.parentNode.removeChild(element);
    // show the detect (progress) button
    detect.classList.remove('hide');
    // make the button unresponsive
    detect.classList.add('progress');
    // shows the status notification
    detect.innerHTML = 'Processing...';
    detect.parentNode.removeChild(detect);
    rld.classList.remove('hide');
    //convert to Blob data
    var blob = dataURItoBlob(preview.firstElementChild.src);
    // form a POST request to the server
    var form_data = new FormData();
    form_data.append('file', blob);
    form_data.append('orientation', orientation);
    //send data to server by ajax query
    $.ajax({
        type: 'POST',
        url: 'http://localhost:3000/',
        data: form_data,
        timeout: 1000 * 25, // failed function will call when timeout
        contentType: false,
        processData: false,
        dataType: 'json',
    }).done(function (data, textStatus, jqXHR) {
        // show image with predicted output
        const ig=document.createElement('image');
        document.querySelector("#imgout").src=data['image'];
        detect.parentNode.removeChild(detect);
        //and show the reload button
        rld.classList.remove('hide');
    }).fail(function (data) {
        alert("Something wrong has occurs. Pls try again!");
        // remove the detect button
        detect.parentNode.removeChild(detect);
        // and show the reload button
        rld.classList.remove('hide');
    });
}

```

Dùng ajax query để gửi ảnh tới server theo method post. Nếu gửi thành công hàm done sẽ được gọi còn không thì hàm fail sẽ được gọi. Trước khi gửi data sẽ được giữ trong FormData và data này phải được chuyển sang dạng bolb

## 2. Back-end:

Flask sẽ được dùng cho phía backend.

Model yolo được implement bằng thư viện pytorch

```
app = Flask(__name__)

#LOAD YOLO MODEL WITH CFG AND WEIGHTS WHENEVER SERVER TURN ON
MODEL = Darknet(YOLOV3_608_CFG_PATH)
MODEL.load_weights(YOLOV3_WEIGHTS_PATH)
MODEL.eval()
```

Mỗi lần server chạy model yolo sẽ được khởi động với file cofig và weights của model.

```
@app.route('/', methods=['POST'])
def upload_file():
    """
    Handles a request. If a request to '/' is of the type POST, handle the image
    and add predictions on it.
    """
    # access files in the request.
    files = request.files['file']

    # save the image ('file') to the disk
    files.save(INPUT_PATH)

    #get orientation value from image that user post
    try:
        orientation = request.form['orientation']
        print(f'Submitted orientation: {orientation}')
    except:
        orientation = 'undefined'
    print(vars(request))
```

Mỗi khi user upload ảnh. Backend sẽ nhận request đó và extract data của user là lưu image lại image.

```

# run the predictions on the saved image
#INPUT_PATH: path of input img
#OUTPUT_PATH: path of predicted img
#archive_path: path to save each predicted img
#labels_path;: path of obj.names
#font_path: path of font that show on predicted img
#model: input the model that loaded above to predict
#device: using cpu to process. declared above
#orientation: a value to check img's rotation
show_image_w_bboxes_for_server(
    INPUT_PATH, OUTPUT_PATH, ARCHIVE_PATH, LABELS_PATH, FONT_PATH, MODEL, DEVICE, orientation
)
# 'show_image_w_bboxes_for_server' saved the output image to the OUTPUT_PATH
#save predicted img as byte-file, then send it back to user
with open(OUTPUT_PATH, 'rb') as in_f:
    # read an image and decode it into utf-8 string and append it
    # to data:image/jpeg;base64 and then return it.
    img_b64 = b64encode(in_f.read()).decode('utf-8')
    img_b64 = 'data:image/jpeg;base64, ' + img_b64

```

Sau đó gọi hàm để nhận thực hiện predict trên ảnh input đầu vào (INPUT\_PATH) và ảnh được predict sẽ được lưu vào (OUTPUT\_PATH) và các kết quả của mỗi ảnh sẽ được lưu vào (ARCHIVE\_PATH). Đưa vào model yolo đã được khởi chạy ở trên (MODEL), chạy cpu (DEVICE). Các tên class (LABELS\_PATH), FONT\_PATH là chữ dự đoán trên ảnh, orientation giá trị chiều của ảnh.

- Thuật toán dự đoán ảnh:

Khi đưa ảnh vào để predict. Giá trị output sẽ là 1 loạt các giá trị dự đoán khác nhau (các bounding box).

Áp dụng non-max suppression:

1. Nếu kết quả prediction lớn hơn điểm tin cậy thấp nhất (obj\_thresh) thì giữ lại prediction đó.
2. Các prediction được giữ lại sẽ tiếp tục đưa vào thuật toán intersection-over-union (iou). Nếu điểm iou lớn ngưỡng iou mà detector cho phép (nms\_thresh) thì loại ra khỏi list dự đoán.
3. Lặp đến khi chọn được .Nếu không thì sẽ return None.


```

def objectness_filter_and_nms(predictions, classes, obj_thresh=0.5, nms_thresh=0.4):
    ...

```

Thuật toán iou vectorized:

**Intersection over Union** là một số liệu đánh giá được sử dụng để đo độ chính xác của bộ phát hiện đối tượng trên một tập dữ liệu cụ thể.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

```
def iou_vectorized(bboxes1, bboxes2, without_center_coords=False):  
    ...
```



## II. Huấn luyện yolo và ứng dụng vào web

### 1. Các bước train mô hình YOLO:

#### Bước 1. Tải source Darknet về máy.

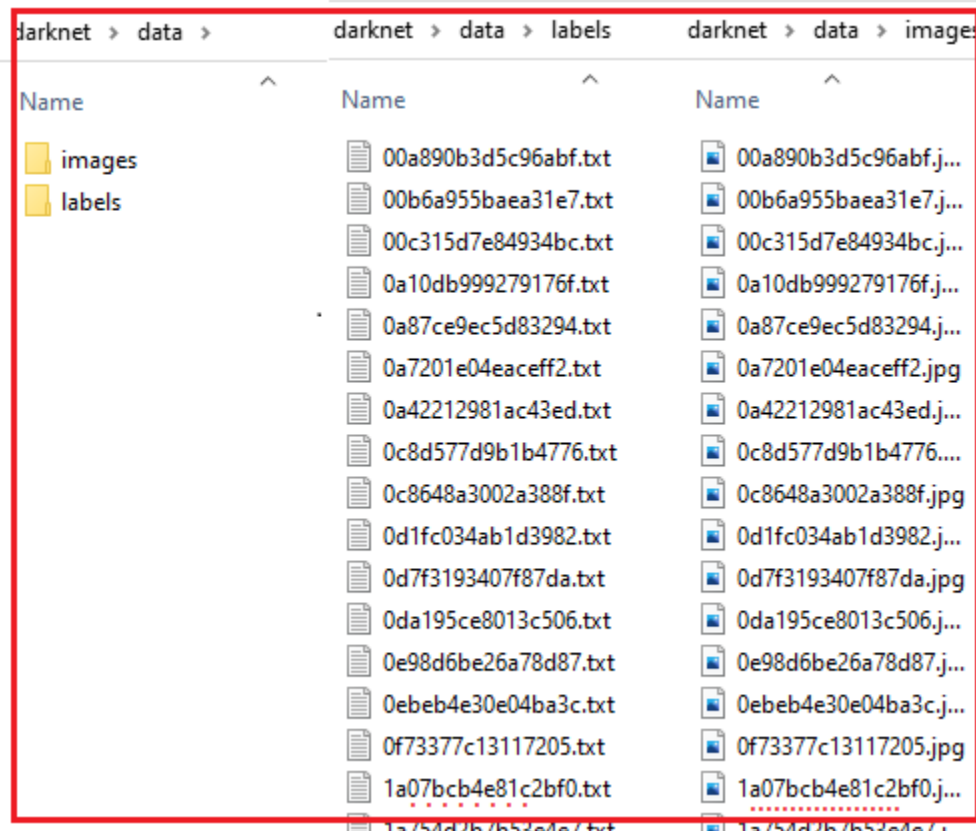
Trên Command Prompt hoặc Terminal, và gõ lệnh:

`git clone https://github.com/pjreddie/darknet`

Name	Date modified	Type	Size
.git	12/22/2021 9:09 PM	File folder	
backup	12/22/2021 9:27 PM	File folder	
cfg	12/22/2021 9:09 PM	File folder	
data	12/22/2021 9:10 PM	File folder	
examples	12/22/2021 9:09 PM	File folder	
include	12/22/2021 9:09 PM	File folder	
python	12/22/2021 9:09 PM	File folder	
scripts	12/22/2021 9:09 PM	File folder	
src	12/22/2021 9:09 PM	File folder	
.gitignore	12/22/2021 9:09 PM	Git Ignore Source ...	1 KB
darknet53.conv.74	12/22/2021 9:14 PM	74 File	158,675 KB
LICENSE	12/22/2021 9:09 PM	File	1 KB
LICENSE.fuck	12/22/2021 9:09 PM	FUCK File	1 KB
LICENSE.gen	12/22/2021 9:09 PM	GEN File	7 KB
LICENSE.gpl	12/22/2021 9:09 PM	GPL File	35 KB
LICENSE.meta	12/22/2021 9:09 PM	META File	1 KB
LICENSE.mit	12/22/2021 9:09 PM	MIT File	2 KB
LICENSE.v1	12/22/2021 9:09 PM	V1 File	1 KB
make_train_val.py	12/20/2021 1:14 AM	Python Source File	1 KB
Makefile	1/2/2022 10:33 AM	File	4 KB
README.md	12/22/2021 9:09 PM	MD File	3 KB

## Bước 2. Chuẩn bị dữ liệu train:

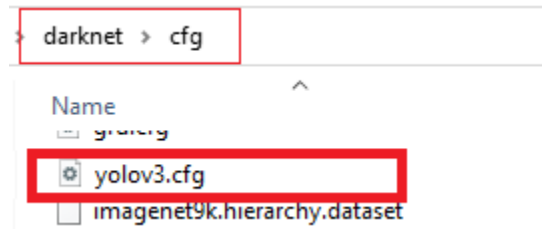
- Dữ liệu có gán nhãn được train chuẩn bị gồm 5 classes: Airplane, Bicycle, Boo, Guitar, Orange và được lấy từ nguồn có sẵn của google: [Open Images Dataset V6 + Extensions](#).
- Bộ dữ liệu huấn luyện của mỗi class là 100 ảnh và với 5 class là 500 được lưu trong folder ở B1 **data/images** và nhãn của dữ liệu lưu trong folder **data/labels**.



Link data của nhóm: <https://shortest.link/2ofj>

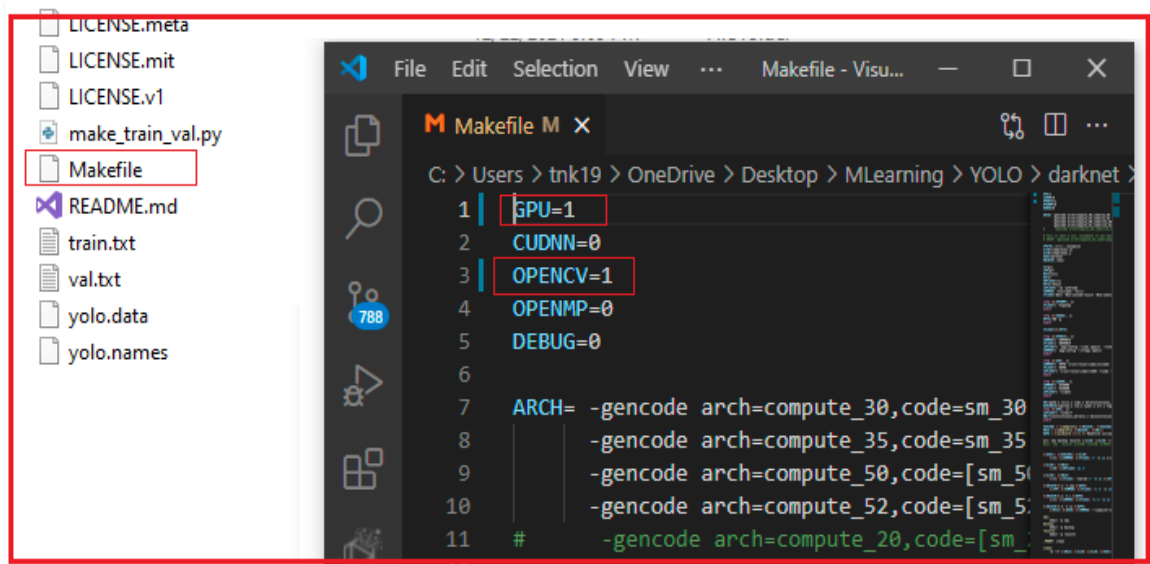
### Bước 3. Chuẩn bị và cấu hình các file cần thiết cho quá trình train dữ liệu.

- Cấu hình file yolov3.cfg tại folder **cfg/**:



- Tại các dòng 610, 696, 783: `classes = 5` là số lượng classes chúng ta huấn luyện.
- Tại các dòng 603, 689, 776: `filters = 30`. Đây chính là layer cuối cùng của base network. Do đó chúng có output shape thay đổi theo số lượng classes theo đúng công thức của bài trước đó là:  $(n\_classes + 5) \cdot 3 = (5+5) \cdot 3 = 30$ .

- Makefile.



- `GPU = 1` : để sử dụng GPU cho quá trình train, ngược lại để `GPU=0`
- `OPENCV = 1` sử dụng thư viện OpenCV ngược lại `OPENCV=0`

Sử dụng make\_train\_val.py tạo file train.txt và val.txt theo tỷ lệ lộn lượt là 80% và 20% của data

```
make_train_val.py
1 import glob
2 import os
3 import shutil
4 import numpy as np
5
6 data_image_path = "data/images/"
7
8 count = 0
9 f = open("train.txt", "w")
10 fv = open("val.txt", "w")
11 print(data_image_path)
12
13 for img_org_path in glob.iglob(data_image_path + '*'):
14     newpath = img_org_path.replace(".PNG", ".png").replace(".JPG", ".jpg").replace(".JPEG", ".jpg")
15     tem = newpath.replace("\\", "/")
16     if count%10<2:
17         fv.write(tem + "\n")
18     else:
19         f.write(tem + "\n")
20     shutil.move(img_org_path, newpath)
21     count += 1
22     print(count)
23 f.close()
24 fv.close()
```

train.txt

```
1 data/images/004b8c5b43c20498.jpg
2 data/images/0055f7e5bac0c8e1.jpg
3 data/images/006ca8fea9211c83.jpg
4 data/images/008c0116989c3edd.jpg
5 data/images/00a890b3d5c96abf.jpg
6 data/images/00b6a955baea31e7.jpg
7 data/images/00c315d7e84934bc.jpg
8 data/images/0108f3fd4b5a6cf4.jpg
9 data/images/0320299003ff65fb.jpg
10 data/images/0328f2c7124c1e3b.jpg
```

```
.....
390 data/images/f5a5fe9e194c3ed1.jpg
391 data/images/f663c82c16f90282.jpg
392 data/images/f6c427772cc0f431.jpg
393 data/images/f734e90785d54649.jpg
394 data/images/f8ebec3f73e1d418.jpg
395 data/images/f9615e322e0f0f96.jpg
396 data/images/fc3b0b4fd04bb7c5.jpg
397 data/images/fdc7a0b9871d9ecf.jpg
398 data/images/fe545aa31a3144d5.jpg
399 data/images/fef29cf2366b7826.jpg
400 data/images/ffd283c172c42dcd.jpg
```

val.txt

```
1 data/images/0026bf4f813772b9.jpg
2 data/images/00396a8814bde4d5.jpg
3 data/images/022cb3b51130c47a.jpg
4 data/images/0285f71d5d3858ea.jpg
5 data/images/065fc991d6d773fb.jpg
6 data/images/06f4ddae5b964094.jpg
7 data/images/0c8d577d9b1b4776.jpg
8 data/images/0d1fc034ab1d3982.jpg
9 data/images/1108d5c218e998cc.jpg
10 data/images/11930f2c976d116c.jpg
```



```
.....
90 data/images/de0f96c0f6e668a1.jpg
91 data/images/e2fe17f2af9c99b7.jpg
92 data/images/e35c2c4b7a0622ad.jpg
93 data/images/e7634b40607116b5.jpg
94 data/images/e7846f0d1b23fd4d.jpg
95 data/images/eaab59fed89e85bf.jpg
96 data/images/eac740e53ce8733a.jpg
97 data/images/f0c48a977312aa4c.jpg
98 data/images/f0f904b1f0ce70ce.jpg
99 data/images/f6e8b13217408704.jpg
100 data/images/f721bc417c652f86.jpg
```

- yolo.data và yolo.names

```
≡ yolo.data
1  classes = 5
2  train = train.txt
3  valid = val.txt
4  names = yolo.names
5  backup = backup
```



```
≡ yolo.names
1  Airplane
2  Bicycle
3  Boot
4  Guitar
5  Orange
```

- yolo.names: chứa tên classes.
- Yolo.data:
  - classes = 5: số lượng classes
  - train = train.txt : file chứa đường dẫn data để train
  - valid = val.txt: file chứa đường dẫn data để validation
  - names = yolo.names: như trên
  - backup = backup: folder chứa các file weights và backup khi train.
- darknet 53.conv.74: được huấn luyện từ bộ dữ liệu ImageNet lưu tại thư mục ở bước B1.

 darknet53.conv.74	12/22/2021 9:14 PM	74 File	158,675 KB
 LICENSE	12/22/2021 9:22 PM	File	4 KB

#### Bước 4: Upload file darknet.zip và tạo folder backup chứa các file weights khi train trên colab.

My Drive > ML ▾ 👤

Name ▾	Owner	Last modified	File size
 backup	me	12:58 AM me	—
 darknet.zip 👤	me	Dec 22, 2021 me	295.2 MB

- Để kết nối với Google Drive.

```
▶ from google.colab import drive  
drive.mount('/content/drive')
```

↳ Mounted at /content/drive

- Lấy và giải nén mã nguồn đã upload trên Drive sang Colab.

```
▶ %cd /content  
!unzip /content/drive/'My Drive'/ML/darknet.zip  
%cd /content/darknet
```

- Biên dịch mã nguồn darknet trên Colab.

```
[ ] %cd /content/darknet  
!make clean  
!make  
!chmod +x ./darknet
```

- Lưu file weights colab sang cái thư mục Backup trên Drive.

```
[ ] !rm /content/darknet/backup -r  
!ln -s /content/drive/'My Drive'/ML/backup /content/darknet
```

- Train model.

```
%cd /content/darknet
!./darknet detector train yolo.data cfg/yolov3.cfg darknet53.conv.74
```

- Train tiếp, thay file darknet53.conv.74 thành file yolo.backup hoặc file.weights

```
!./darknet detector train yolo.data cfg/yolov3.cfg backup/yolov3.backup
```

```

[14]: !rm /content/darknet/backup -r
      !ln -s /content/drive/My Drive/HL/backup /content/darknet























[ ] %cd /content/darknet
      !./darknet detector train yolo.data cfg/yolov3.cfg darknet53.conv.74

!./darknet detector train yolo.data cfg/yolov3.cfg backup/yolov3.backup

...
Region 82 Avg IOU: 0.815565, Class: 0.983303, Obj: 0.559650, No Obj: 0.002410, .5R: 1.000000, .75R: 1.000000, count: 4
Region 94 Avg IOU: 0.797249, Class: 0.998185, Obj: 0.954735, No Obj: 0.000306, .5R: 1.000000, .75R: 1.000000, count: 1
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000006, .5R: -nan, .75R: -nan, count: 0
Region 82 Avg IOU: 0.812289, Class: 0.999736, Obj: 0.731152, No Obj: 0.001645, .5R: 1.000000, .75R: 0.666667, count: 3
Region 94 Avg IOU: 0.733548, Class: 0.999963, Obj: 0.298320, No Obj: 0.000469, .5R: 1.000000, .75R: 0.000000, count: 1
Region 106 Avg IOU: 0.559649, Class: 0.990310, Obj: 0.899982, No Obj: 0.000223, .5R: 0.600000, .75R: 0.000000, count: 5
Region 82 Avg IOU: 0.860247, Class: 0.996049, Obj: 0.765561, No Obj: 0.001779, .5R: 1.000000, .75R: 1.000000, count: 3
Region 94 Avg IOU: 0.775582, Class: 0.998554, Obj: 0.928020, No Obj: 0.000603, .5R: 1.000000, .75R: 0.666667, count: 3
Region 106 Avg IOU: 0.777258, Class: 0.996469, Obj: 0.901377, No Obj: 0.000047, .5R: 1.000000, .75R: 1.000000, count: 1
Region 82 Avg IOU: 0.819139, Class: 0.997271, Obj: 0.587453, No Obj: 0.001856, .5R: 1.000000, .75R: 1.000000, count: 3
Region 94 Avg IOU: 0.681756, Class: 0.999969, Obj: 0.835930, No Obj: 0.000298, .5R: 1.000000, .75R: 0.000000, count: 1
Region 106 Avg IOU: 0.775442, Class: 0.993301, Obj: 0.005753, No Obj: 0.000013, .5R: 1.000000, .75R: 1.000000, count: 1
Region 82 Avg IOU: 0.775493, Class: 0.999634, Obj: 0.928185, No Obj: 0.001000, .5R: 1.000000, .75R: 1.000000, count: 1
Region 94 Avg IOU: 0.780907, Class: 0.998720, Obj: 0.827150, No Obj: 0.001184, .5R: 1.000000, .75R: 0.666667, count: 6
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000010, .5R: -nan, .75R: -nan, count: 0
Region 82 Avg IOU: 0.898625, Class: 0.988527, Obj: 0.933708, No Obj: 0.000849, .5R: 1.000000, .75R: 1.000000, count: 1
Region 94 Avg IOU: 0.804266, Class: 0.981551, Obj: 0.332381, No Obj: 0.000631, .5R: 1.000000, .75R: 0.666667, count: 3
Region 106 Avg IOU: 0.665641, Class: 0.999534, Obj: 0.775932, No Obj: 0.001294, .5R: 0.944444, .75R: 0.166667, count: 36
Loaded: 0.000057 seconds
Region 82 Avg IOU: 0.786874, Class: 0.999176, Obj: 0.666481, No Obj: 0.002934, .5R: 1.000000, .75R: 0.750000, count: 4
Region 94 Avg IOU: 0.778820, Class: 0.997964, Obj: 0.983853, No Obj: 0.000460, .5R: 1.000000, .75R: 0.500000, count: 2
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000002, .5R: -nan, .75R: -nan, count: 0
Region 82 Avg IOU: 0.871651, Class: 0.994766, Obj: 0.778659, No Obj: 0.003936, .5R: 1.000000, .75R: 0.800000, count: 5
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000048, .5R: -nan, .75R: -nan, count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000011, .5R: -nan, .75R: -nan, count: 0
Region 82 Avg IOU: 0.807633, Class: 0.988968, Obj: 0.634591, No Obj: 0.002585, .5R: 1.000000, .75R: 0.600000, count: 5
Region 94 Avg IOU: 0.569383, Class: 0.995481, Obj: 0.818336, No Obj: 0.000492, .5R: 1.000000, .75R: 0.000000, count: 1
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000006, .5R: -nan, .75R: -nan, count: 0
Region 82 Avg IOU: 0.760588, Class: 0.998863, Obj: 0.914814, No Obj: 0.002578, .5R: 1.000000, .75R: 1.000000, count: 2
Region 94 Avg IOU: 0.664119, Class: 0.997936, Obj: 0.879087, No Obj: 0.000798, .5R: 1.000000, .75R: 0.000000, count: 7
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan, count: 0
Đang thực thi (3 giây) Cell > system() > _system_compat() > _run_command() > _monitor_process() > _poll_process()

```

- Kết quả folder backup với epoch = 4000.

 yolov3.backup 	me
 yolov3_4000.weights 	me
 yolov3_3000.weights 	me
 yolov3_2000.weights 	me
 yolov3_900.weights 	me
 yolov3_800.weights 	me
 yolov3_700.weights 	me
 yolov3_600.weights 	me
 yolov3_500.weights 	me
 yolov3_400.weights 	me
 yolov3_300.weights 	me

## 2. Đánh giá F1- core epoch = 4000

```

4000: 1.112405, 0.958609 avg loss, 0.001000 rate, 32.873148 seconds, 256000 images, 68.841687 hours left
Resizing to initial size: 608 x 608 try to allocate additional workspace_size = 106.46 MB
CUDA allocate done!

calculation mAP (mean average precision)...
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
100
detections_count = 1013, unique_truth_count = 224
class_id = 0, name = Airplane, ap = 60.10% (TP = 11, FP = 0)
class_id = 1, name = Bicycle, ap = 38.37% (TP = 14, FP = 5)
class_id = 2, name = Boat, ap = 33.74% (TP = 12, FP = 6)
class_id = 3, name = Guitar, ap = 56.98% (TP = 15, FP = 2)
class_id = 4, name = Orange, ap = 47.31% (TP = 40, FP = 55)

for conf_thresh = 0.25, precision = 0.57, recall = 0.41, F1-score = 0.48
for conf_thresh = 0.25, TP = 92, FP = 68, FN = 132, average IoU = 41.51 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.472987, or 47.30 %
Total Detection Time: 12 Seconds

```



### III. Tài liệu tham khảo:

<https://shortest.link/2v83>

<https://shortest.link/2v88>

[How to develop a HTML5 Image Uploader - Mozilla Hacks - the Web developer blog](#)

[Khoa học dữ liệu \(phamdinhhkhanh.github.io\)](https://phamdinhhkhanh.github.io)

[Anchor Boxes — The key to quality object detection | Towards Data Science](#)

[Intersection over Union \(IoU\) for object detection - PyImageSearch](#)

[Tìm hiểu và triển khai thuật toán Non Maximum Suppression \(viblo.asia\)](https://viblo.asia)

[AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection \(Windows and Linux version of Darknet \) \(github.com\)](#)