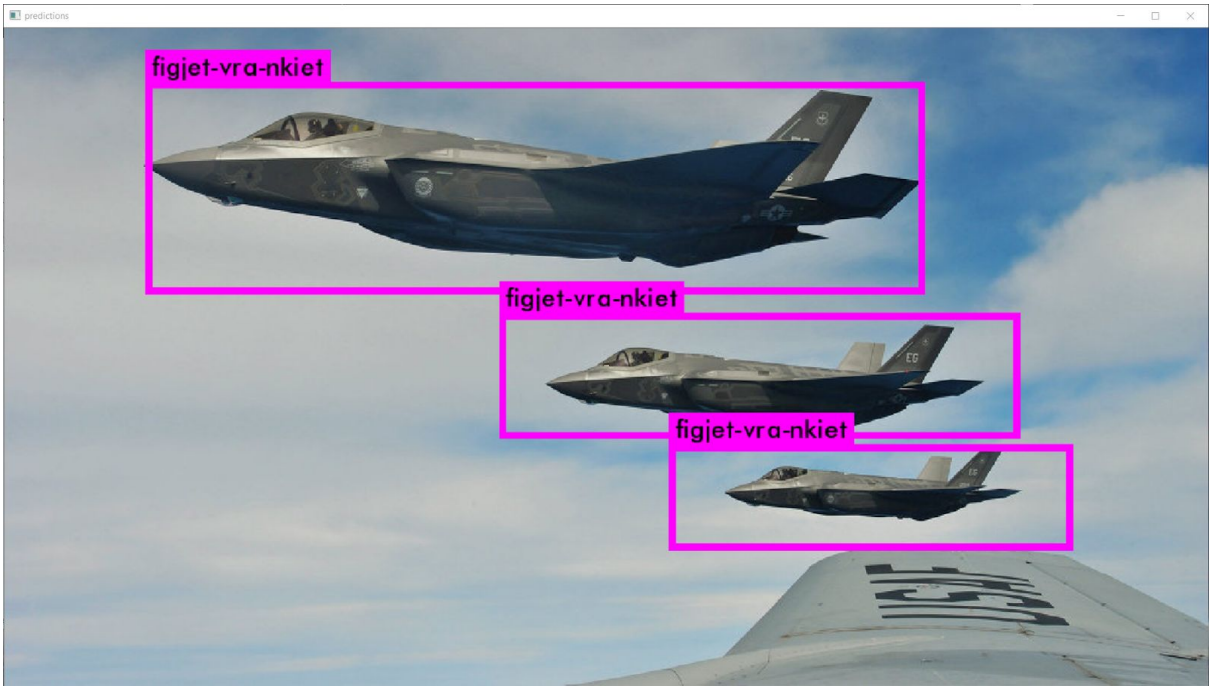
Đồ án cuối kỳ VRA

Người thực hiện: Tô Nhân Kiệt (nhankiet1996 [at] gmail [dot] com)

Dạng đồ án: dạng 3 (sử dụng YOLO và train một đối tượng mới)

Nội dung đồ án: Sử dụng YOLO deep learning framework để train một model nhận dạng máy bay chiến đấu.

(LƯU Ý: MỤC ĐÍCH CỦA ĐỒ ÁN NÀY HOÀN TOÀN VÀ CHỈ CÓ MỘT MỤC ĐÍCH ĐÓ LÀ PHỤC VỤ HỌC TẬP-NGHIÊN CỨU, NGOÀI RA KHÔNG CÒN BẤT CỨ MỤC ĐÍCH GÌ KHÁC. TÁC GIẢ SẼ KHÔNG CHỊU TRÁCH NHIỆM CHO THIỆT HẠI HAY HÀNH VI BẤT CHÍNH NÀO SỬ DỤNG ĐỒ ÁN NÀY)



(Hình lấy từ kết quả thực hiện đồ án)

Khóa học nhận dạng thị giác VRA 2017:
GIẢNG VIÊN: TS. Lê Đình Duy - TS. Nguyễn Tấn Trần Minh Khang

Mục lục

Kết quả đạt được **2**
Link video: 2
Độ chính xác: 2
Kinh nghiệm thu được: 2
Hướng dẫn cài đặt **3**
Cài đặt Microsoft Visual Studio 2015: 3
Cài đặt OpenCV 3.x: 4
Cài đặt CUDA: 4
Cài đặt cuDNN 4
Clone/download repo darknet này: 5
Cài đặt repo darknet: 5
Chuẩn bị hình ảnh: 8
Cài đặt Yolo-mark: 9
Vẽ bounding box cho hình: 10
Tiến hành training 12
Tiến hành test - thử nghiệm kết quả 16
Đồ án này được hỗ trợ bởi **19**

1. Kết quả đạt được

Link video:

https://drive.google.com/file/d/1eLWIU-KDK-2B5uX-gzwOGa3NyX2pu7A0/view?usp=sharing

hoặc

http://www.mediafire.com/file/jr1lma7u4wai9u1/VRA.Final.ResultVideo.mkv

(Có caption trong video)
Trong video bao gồm:

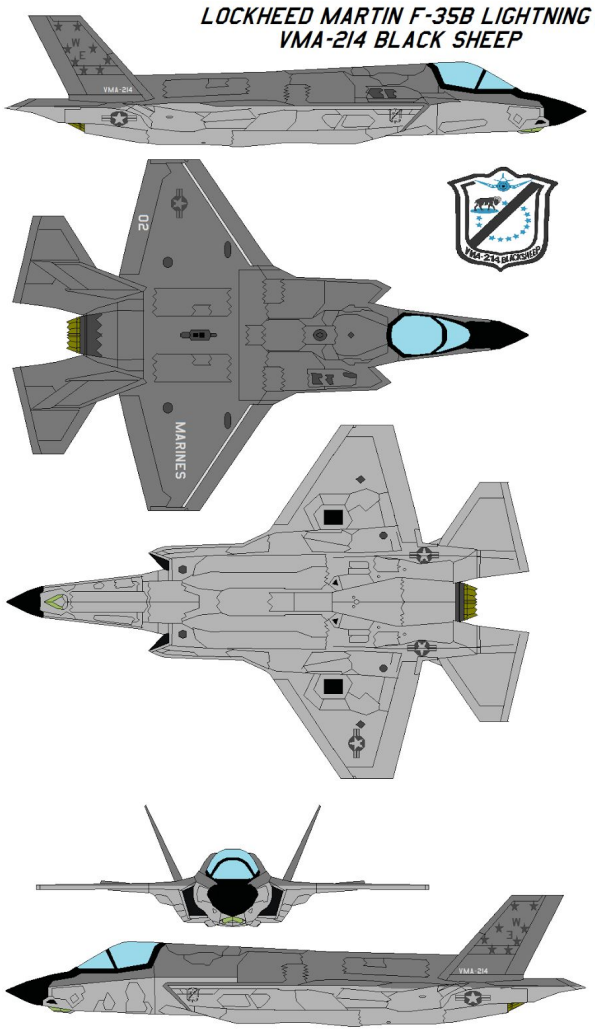
- Duyệt qua ảnh dữ liệu thu thập
- Dùng Yolo_Mark để vẽ bounding box cho ảnh và config các file ở đây
- Đưa ảnh đã có bounding box và các file config qua folder chứa darknet.exe
- Tiến hành train (demo)
- Tiến hành test với 3 ảnh và 3 video

Độ chính xác:

Ở đồ án này, tác giả không đặt nặng về độ chính xác hay có làm thống kê về các lần thử weights hay các bộ dữ liệu, ở một số đoạn trong video hay ảnh khi các máy bay bị chồng chéo lên nhau hay bay ra xa thì không còn nhận diện được. Tuy nhiên độ chính xác vẫn ở mức khá và chấp nhận được và theo tinh thần của môn học thì việc này hoàn toàn ổn.

Kinh nghiệm thu được:

- Cần một lượng lớn dữ liệu để train Deep Learning, và càng nhiều dữ liệu, train càng lâu thì càng cho ra kết quả tốt.
- Đối với việc nhận dạng các vật thể, ta nên thu thập hình ảnh ở nhiều góc độ của vật thể - ở đây ví dụ như chiếc chiến đấu cơ, ta nên thu thập hình ảnh trên, dưới, trái phải, trước, sau của chúng, càng nhiều góc độ càng tốt để khi nhận dạng đạt độ chính xác cao, giả sử như ta chỉ thu thập các hình ảnh phía trước của chúng thì nếu như lúc sau chúng xuất hiện với hình ảnh ở bên trái hay phía sau,...thì ta sẽ không nhận dạng được.



2. Hướng dẫn cài đặt

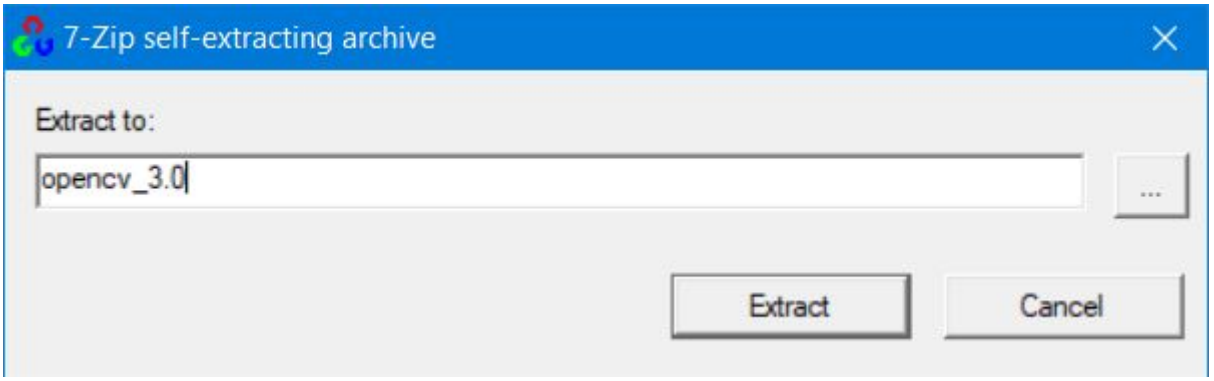
Do tác giả sử dụng **môi trường Windows 10 và GPU của hãng NVIDIA (GTX 1080)** để thực hiện đồ án này, nên dưới đây sẽ là hướng dẫn cài đặt và thực hiện cho môi trường Windows 10 cùng GPU của NVIDIA. Các OS khác hay hãng GPU khác thì mời xem phần cuối "Đồ án này được hỗ trợ bởi" để tìm hướng dẫn phù hợp.

2.1. Cài đặt Microsoft Visual Studio 2015:

Link: https://go.microsoft.com/fwlink/?LinkId=532606&clcid=0x409
Yêu cầu cài đặt đủ Visual C++.

2.2. Cài đặt OpenCV 3.x:

Link:https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.2.0/opencv-3.2.0-vc14.exe/download
Sau khi download về thì bỏ vào ổ C:// (phải để ở ngay bên ngoài), sau đó click vào và điền tên folder extract là “opencv_3.0” như trong hình.



2.3. Cài đặt CUDA:

Link: https://developer.nvidia.com/cuda-downloads
Lưu ý: Nên chọn phiên bản mới nhất hoặc 8.0.
Sau khi down về cài đặt bình thường, nên chọn Express Installation.

2.4. Cài đặt cuDNN

Link: https://developer.nvidia.com/cudnn
Lưu ý: cần phải tạo tài khoản sau đó mới có thể download, sau khi tạo tài khoản, chọn cuDNN cho phiên bản CUDA mà bạn đã cài đặt sau đó download library cho Windows 10, rồi theo hướng dẫn Installation của phiên bản đó mà tiến hành cài đặt.

(Với tác giả thì sau khi down cái cuDNN về và giải nén, tác giả làm như sau:
a) Copy \cuda\bin\cudnn64_7.dll vào C:\Program Files \NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin.
b) Copy \cuda\ include\cudnn.h vào C:\Program Files \NVIDIA GPU Computing Toolkit\CUDA\v9.0\include.
c) Copy \cuda\lib\x64\cudnn.lib vào C:\Program Files \NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64.)

2.5. Clone/download repo darknet này:

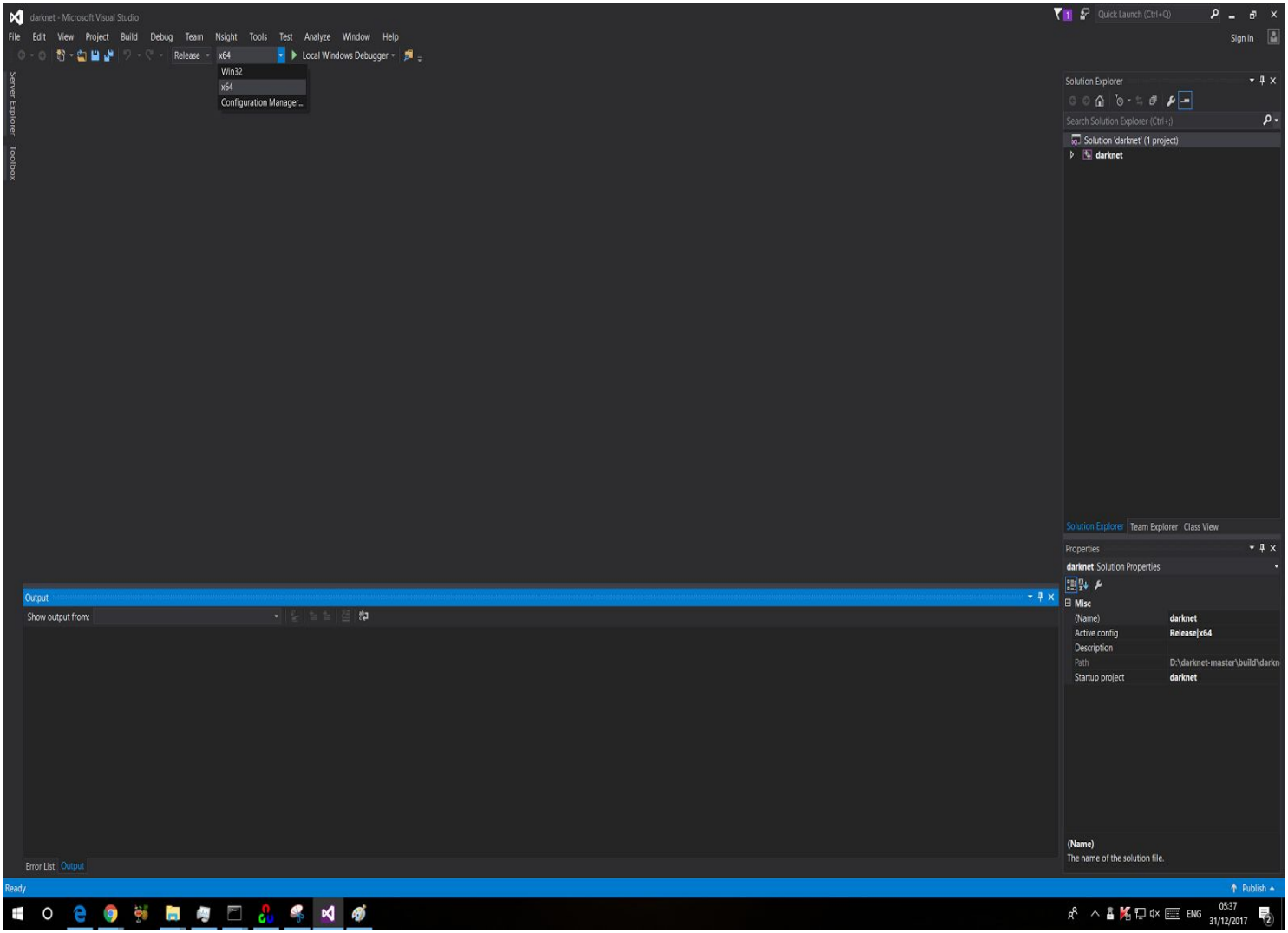
Link: https://github.com/AlexeyAB/darknet
Đây tuy không phải là repo gốc của YOLO nhưnglà repo darknet hỗ trợ Windows lẫn Linux.

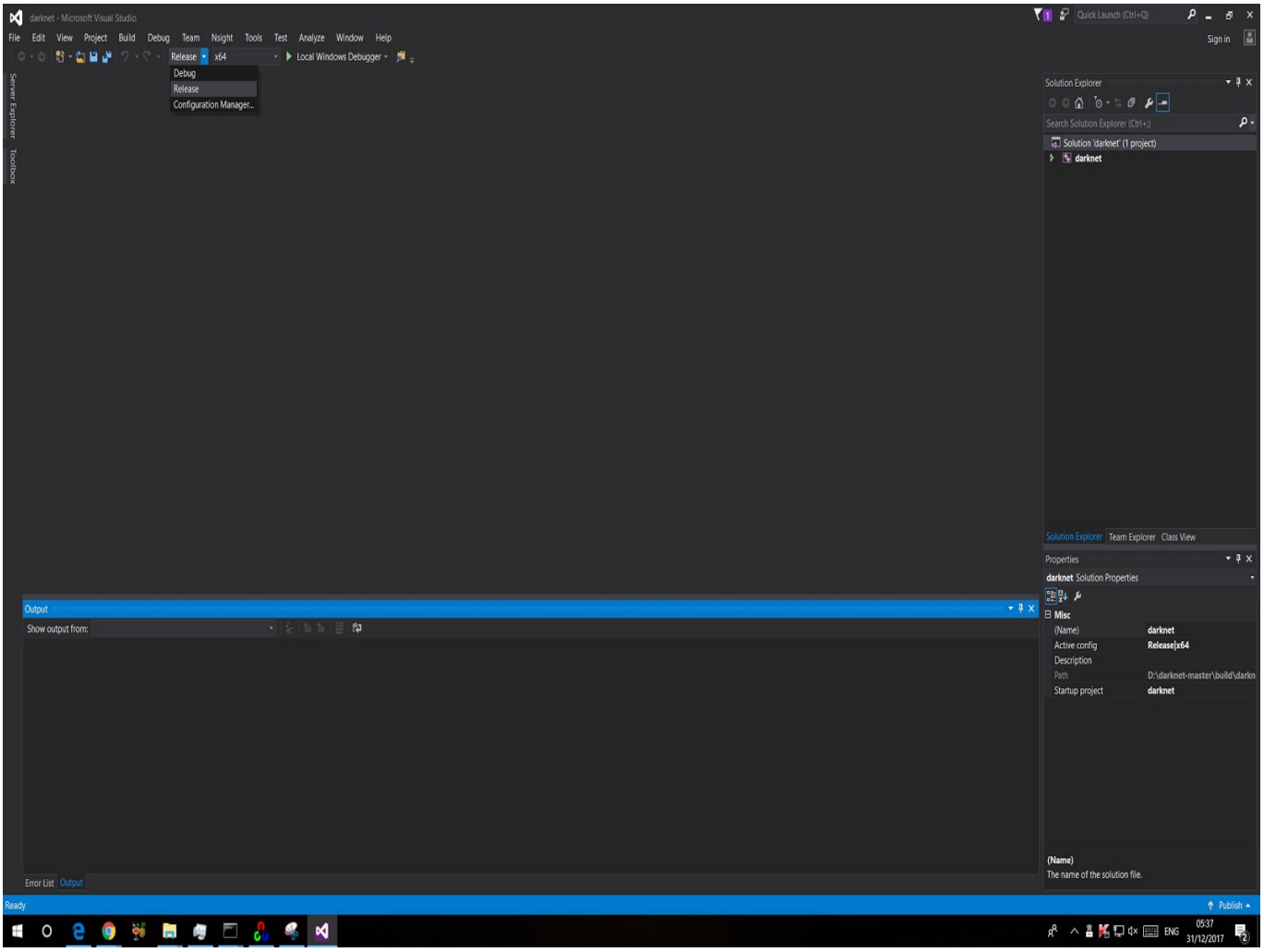
2.6. Cài đặt repo darknet:

B1: Sau khi download repo darknet trên thì vào darknet-master/build/darknet/

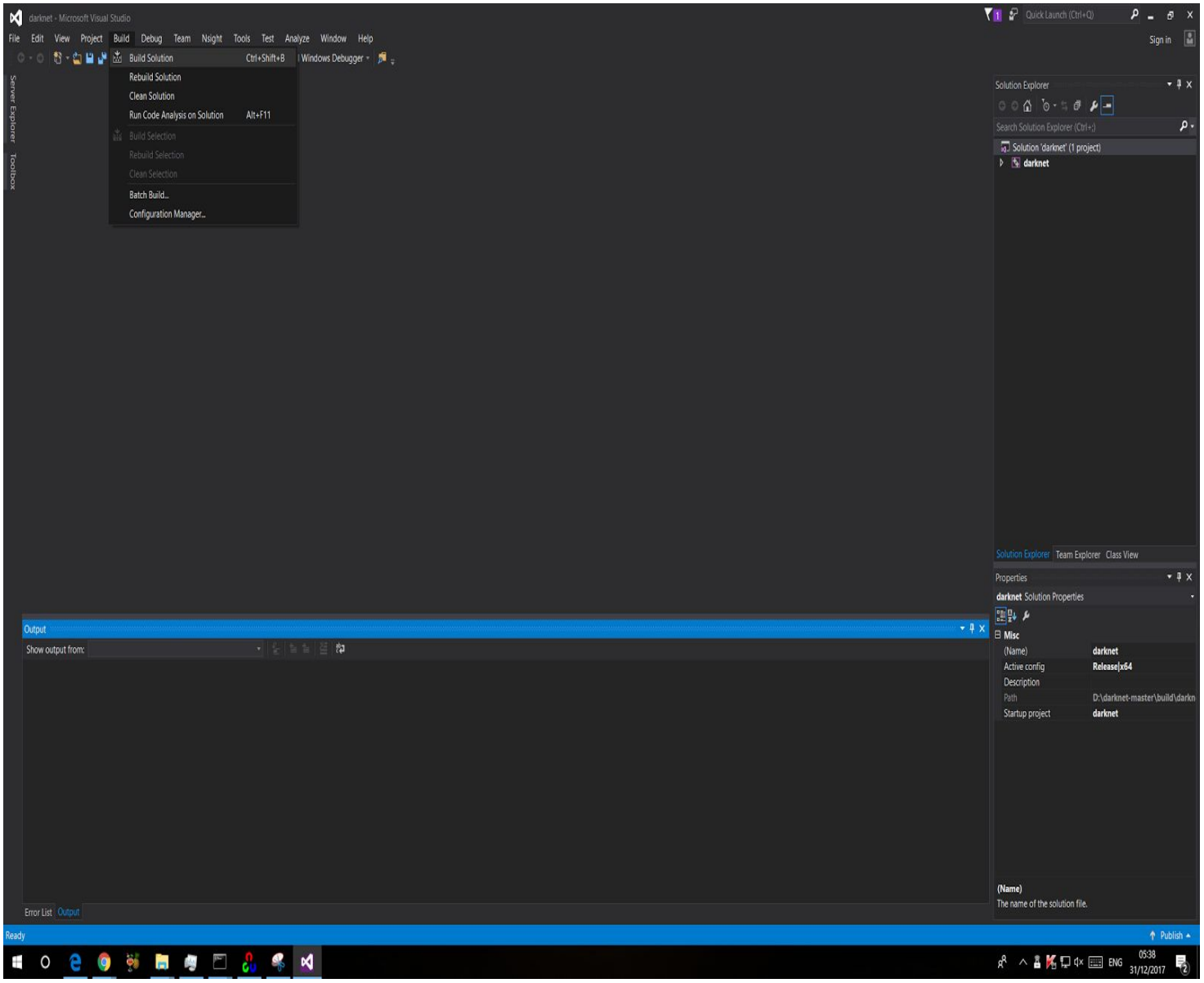
B2: Ở đây nếu như vừa nãy bạn cài đặt phiên bản của CUDA **khác bản 8.0** thì click vào darknet.vcxproj, mở bằng một texteditor nào đó hay notepad, tìm kiếm "**CUDA 8.0**" và thay thế/replace bằng phiên bản CUDA mà bạn cài đặt lúc nãy (ở trường hợp của tác giả là 9.1) sẽ có 2 chỗ matched. Nếu bạn cài đặt CUDA 8.0 thì không cần thực hiện bước này.

B3: Click vào file darknet.sln và mở bằng MS Visual Studio 2015, chỉnh lại thành **Release** và **x64** như hình dưới đây:




darknet - Microsoft Visual Studio
File Edit View Project Build Debug Team Nsight Tools Test Analyze Window Help
Release x64
Local Windows Debugger
Debug
Release
Configuration Manager...
Server Explorer Toolbox
Quick Launch (Ctrl+Q)
Sign in
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'darknet' (1 project)
darknet
Solution Explorer Team Explorer Class View
Properties
darknet Solution Properties
Misc
(Name) darknet
Active config Release|x64
Description
Path D:\darknet-master\build\darkn
Startup project darknet
(Name)
The name of the solution file.
Output
Show output from:
Error List Output
Ready
Publish
ENG
05:37
31/12/2017

Sau đó tiến hành Build



File **darknet.exe** sẽ được tạo ra trong folder x64

Bây giờ bạn qua ổ C:// chỗ bạn extract OpenCV ra, nếu bạn làm đúng như hướng dẫn thì lúc này đường dẫn của OpenCV sẽ là: **C:\opencv_3.0\opencv**.

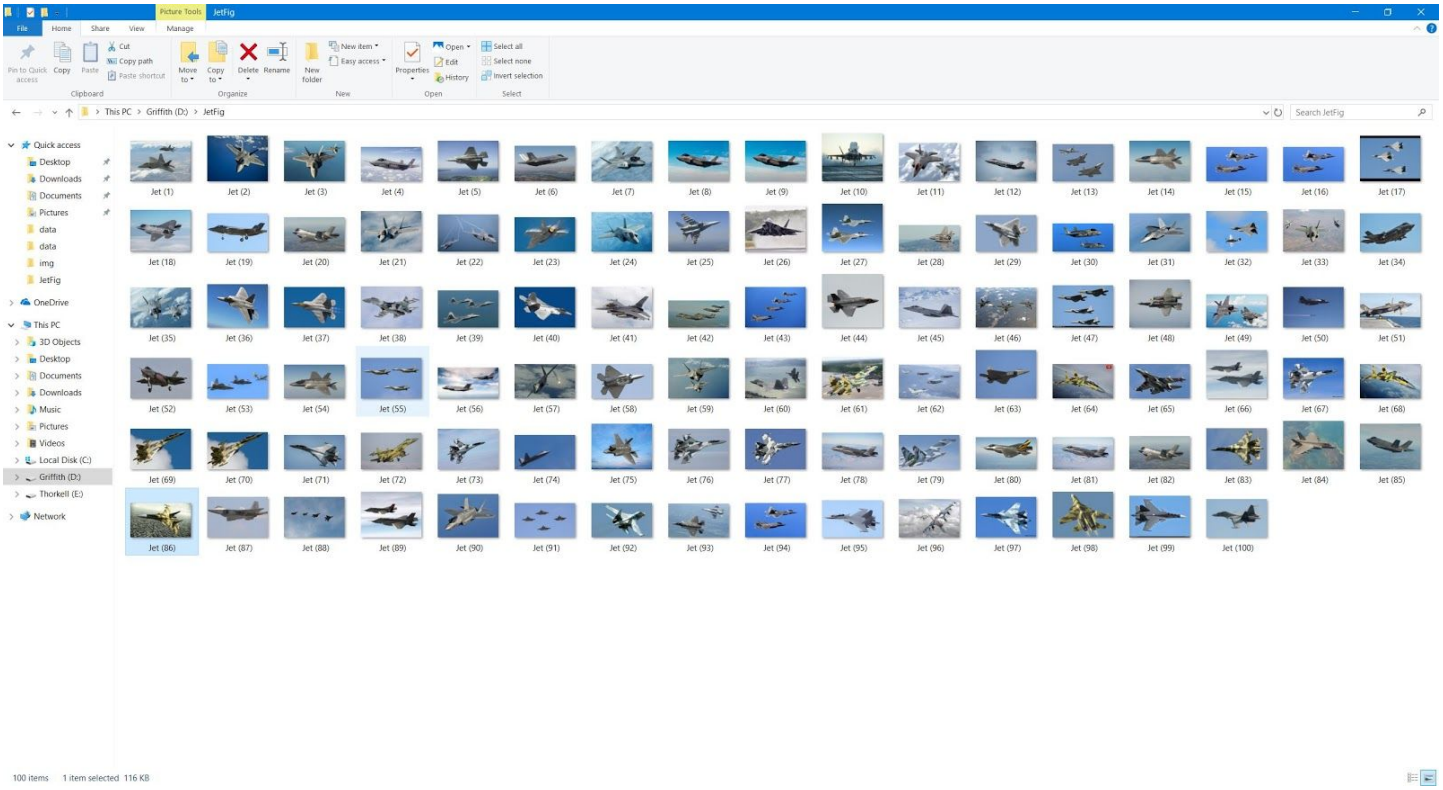
Bạn vào **C:\opencv_3.0\opencv\build\x64\vc14\bin** lấy **opencv_world320.dll** và **opencv_ffmpeg320_64.dll,** sau đó copy 2 file này vào cái folder chứa **darknet.exe** vừa tạo (ở trong darknet/build/darknet/x64)

2.7. Chuẩn bị hình ảnh:

Ở đồ án này tác giả sử dụng 100 hình ảnh (nếu nhiều hơn thì càng tốt), lấy từ Google Images, với đề tài là các máy bay chiến đấu thì nhập vào các từ khóa như: F22, F35, F16, F18, SU 35, Su 27, Su 30. Nên chọn các hình có định dạng đuôi .jpg.

Sau khi down hình về và để tập trung vào một folder, sau đó rename lại hình như trên (để cho nhanh thì ấn tổ hợp Ctrl + A, rồi ấn chuột phải vào một hình chọn rename rồi điền từ tên vào ví dụ như: Jet, thì Windows sẽ tự rename lại tất cả thành: Jet (1), Jet (2),..., Jet (100).

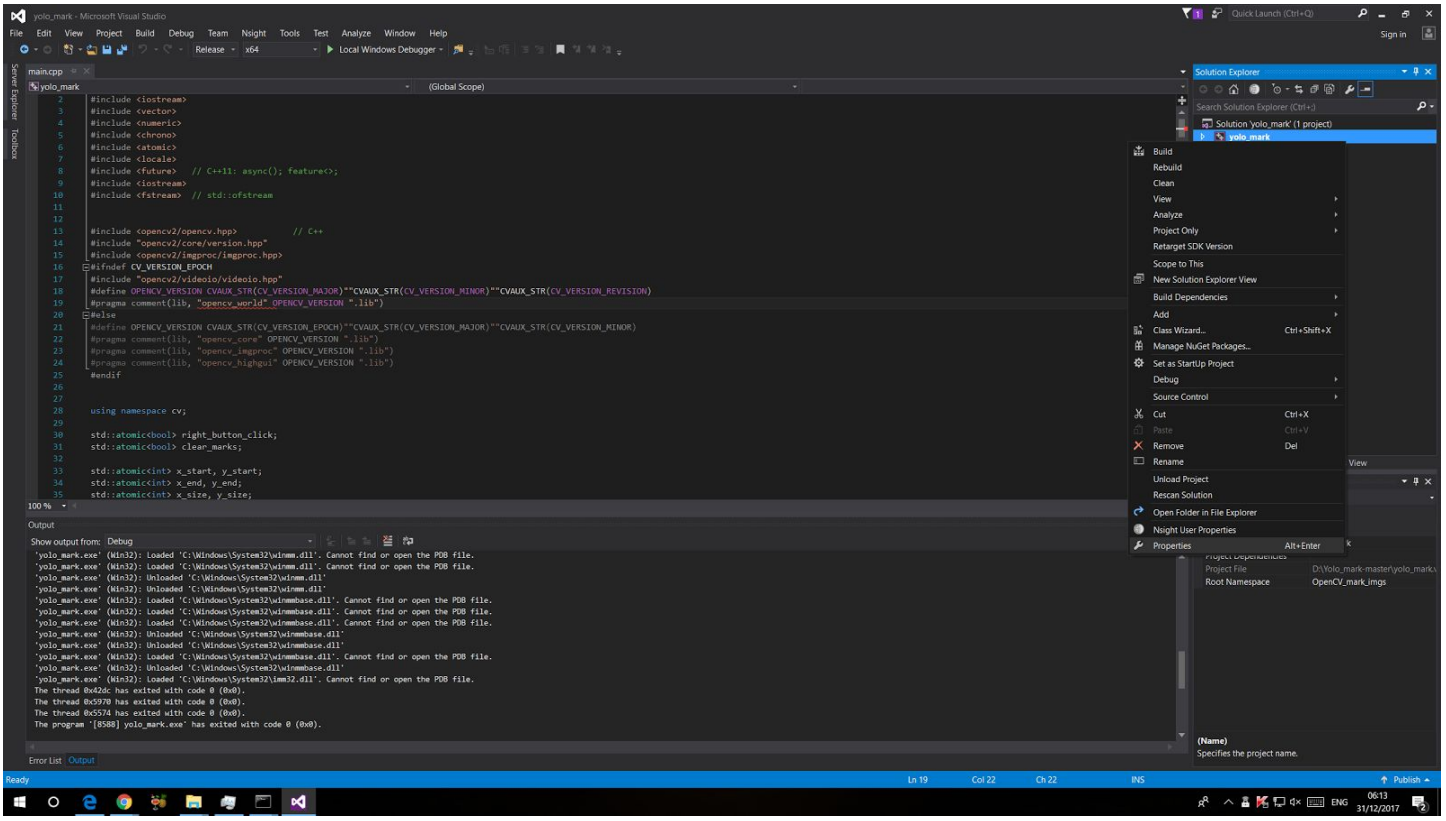
Phía dưới là ví dụ của tác giả:



2.8. Cài đặt Yolo-mark:

Link: https://github.com/AlexeyAB/Yolo_mark

Đây là công cụ giúp ta vẽ các bounding box dùng làm dữ liệu training cho YOLO, sau khi download về rồi bạn mở file “yolo_mark.sln” lên, chỉnh **x64 và Release** như lúc build darknet, sau đó ấn chuột phải vào Project chọn Properties như hình:



Và làm như sau:

B1: Vào: C/C++ -> General -> Additional Include Directories:

Điền vào “C:\opencv_3.0\opencv\build\include;”

Chọn Apply

B2: Vào Linker -> General -> Additional Library Directories:
Điền vào C:\opencv_3.0\opencv\build\x64\vc14\lib;

Chọn Apply

Sau đó tiến hành Build Project

(nếu gặp phải lỗi fopen unsafe thì làm theo hướng dẫn ở đây: https://stackoverflow.com/questions/21873048/getting-an-error-fopen-this-function-or-variable-may-be-unsafe-when-complin)

2.9. Vẽ bounding box cho hình:

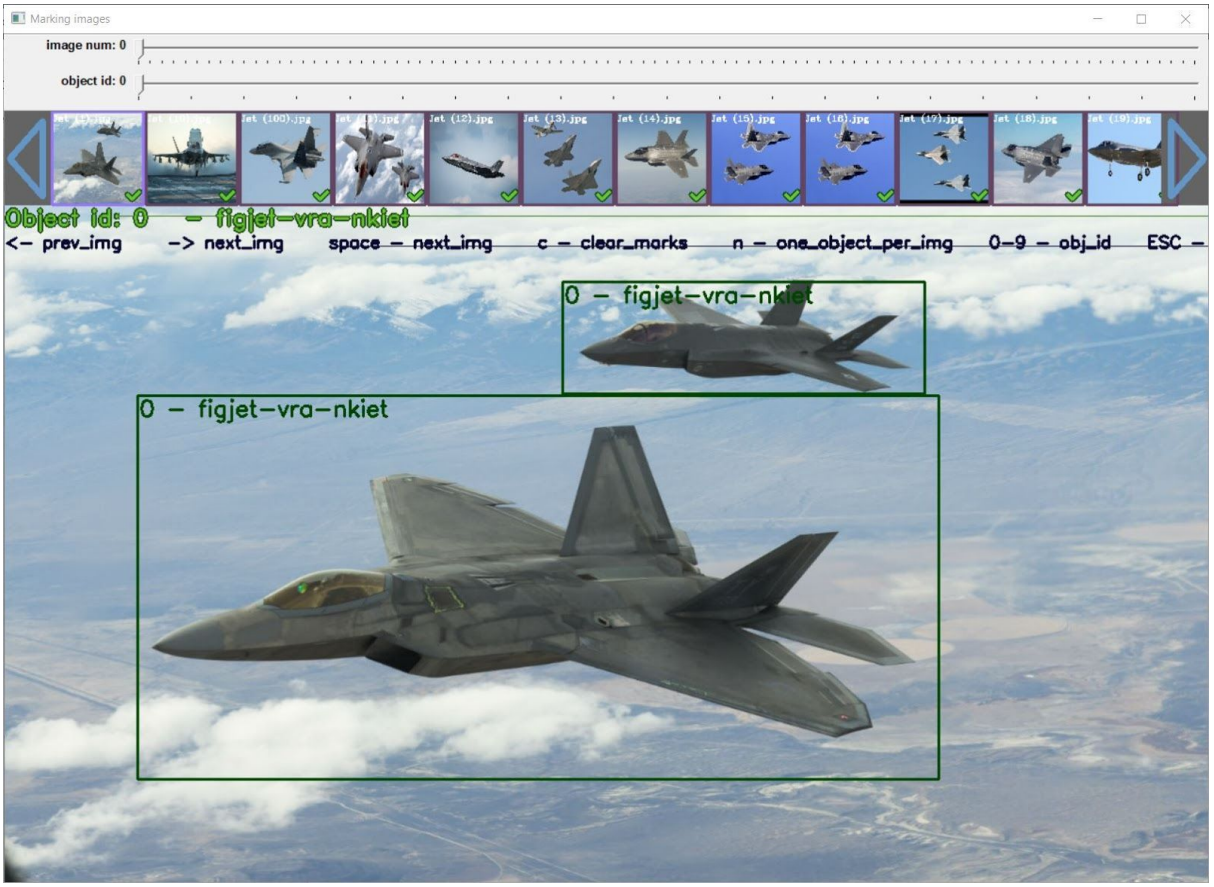
Vào folder x64/Release của yolo_mark và chạy **yolo_mark.cmd** nếu chạy đúng thì lúc này sẽ hiện hình vài cái máy bay.

Kế tiếp, bạn vào folder data/img xóa toàn bộ mọi thứ trong folder này, sau đó mang mớ hình - 100 cái máy bay mà đã down về và copy vào folder data/img.

Ở đây đồ án này chỉ train 1 label, nên bạn dùng một text-editor nào đó hay notepad mở obj.data lên và sửa số classes=1 (bạn train bao nhiêu label thì sẽ có bấy nhiêu class).

Tương tự, mở obj.names lên và điền tên của class vào, ở đây mình đặt tên là figjet-vra-nkiet (viết tắt của fighter-jet-vra-nhankiet) (ở đây lưu ý: không nên để tên có dấu hay có dấu cách hay viết hoa). Bạn có bao nhiêu class thì có bấy nhiêu cái tên.

Sau khi hoàn thành, bạn ra ngoài x64/Release chạy file yolo_mark.cmd,

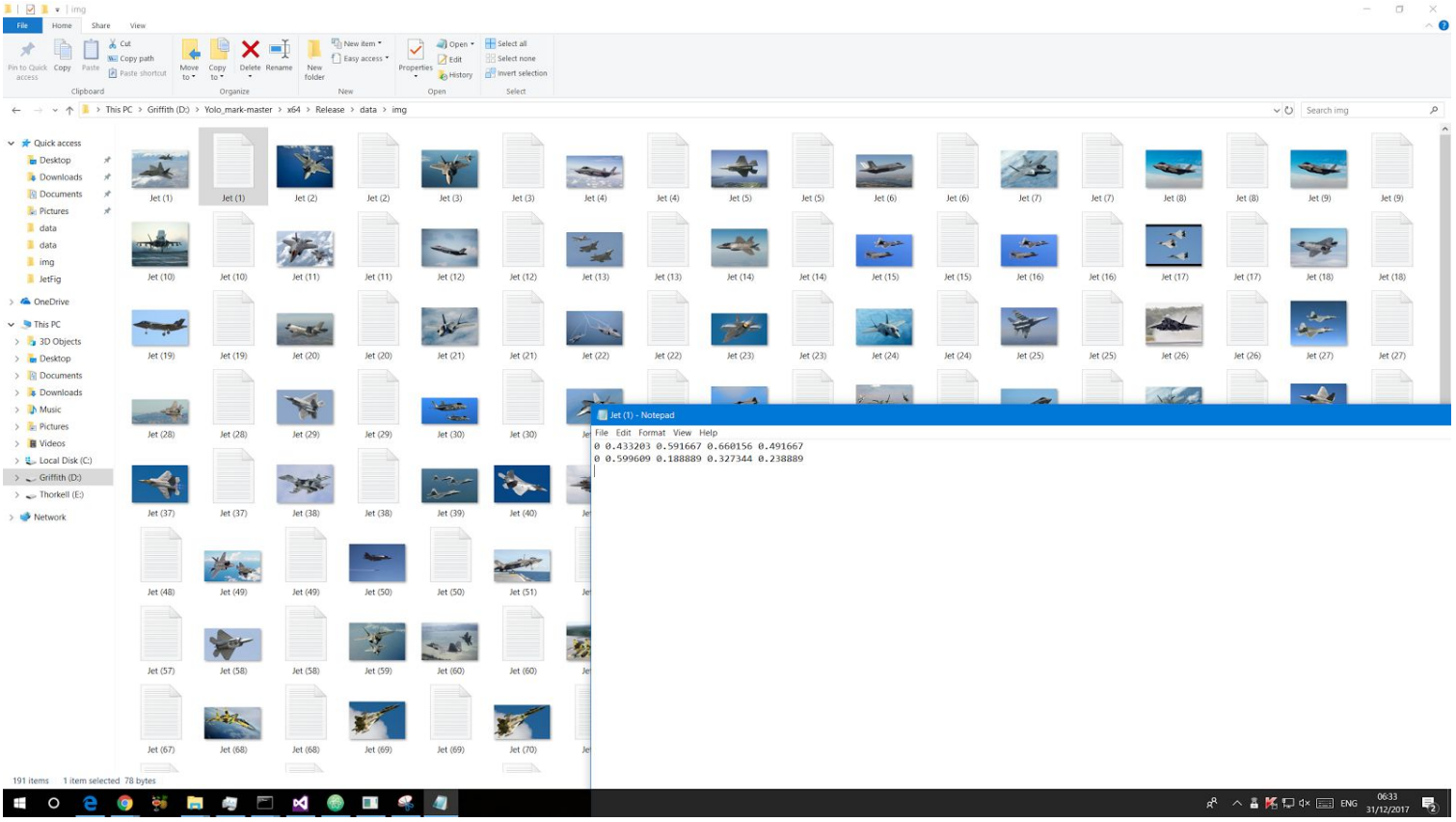


Bây giờ bạn tiến hành vẽ bouding box cho label trên ảnh:

- Click chuột trái và kéo để tạo bounding box.
- Để qua hình kế tiếp click phím Space hoặc mũi tên phải.
- Để qua hình vừa rồi thì click mũi tên trái
- Để xóa bounding box của hình hiện tại ấn phím C
- 0-9 để chọn label (ở đồ án này chỉ có 1 label nên không cần lo về cái này)

Làm lần lượt đến khi xong tất cả các hình, sẽ có một số hình không hiện lên trên yolo_mark này, nhưng bạn không cần lo, vì dù thế lát nữa vẫn có thể chạy được.

Sau khi đánh dấu xong kết quả trong data/img sẽ như sau: file txt sẽ cho biết thông tin của label có trong ảnh, ví dụ: 0 0.433 0.5999 0.777 0.409. Số 0 đầu tiên cho ta biết là loại label gì và 4 số còn lại mô tả vị trí cái bounding box,



2.10. Tiến hành training

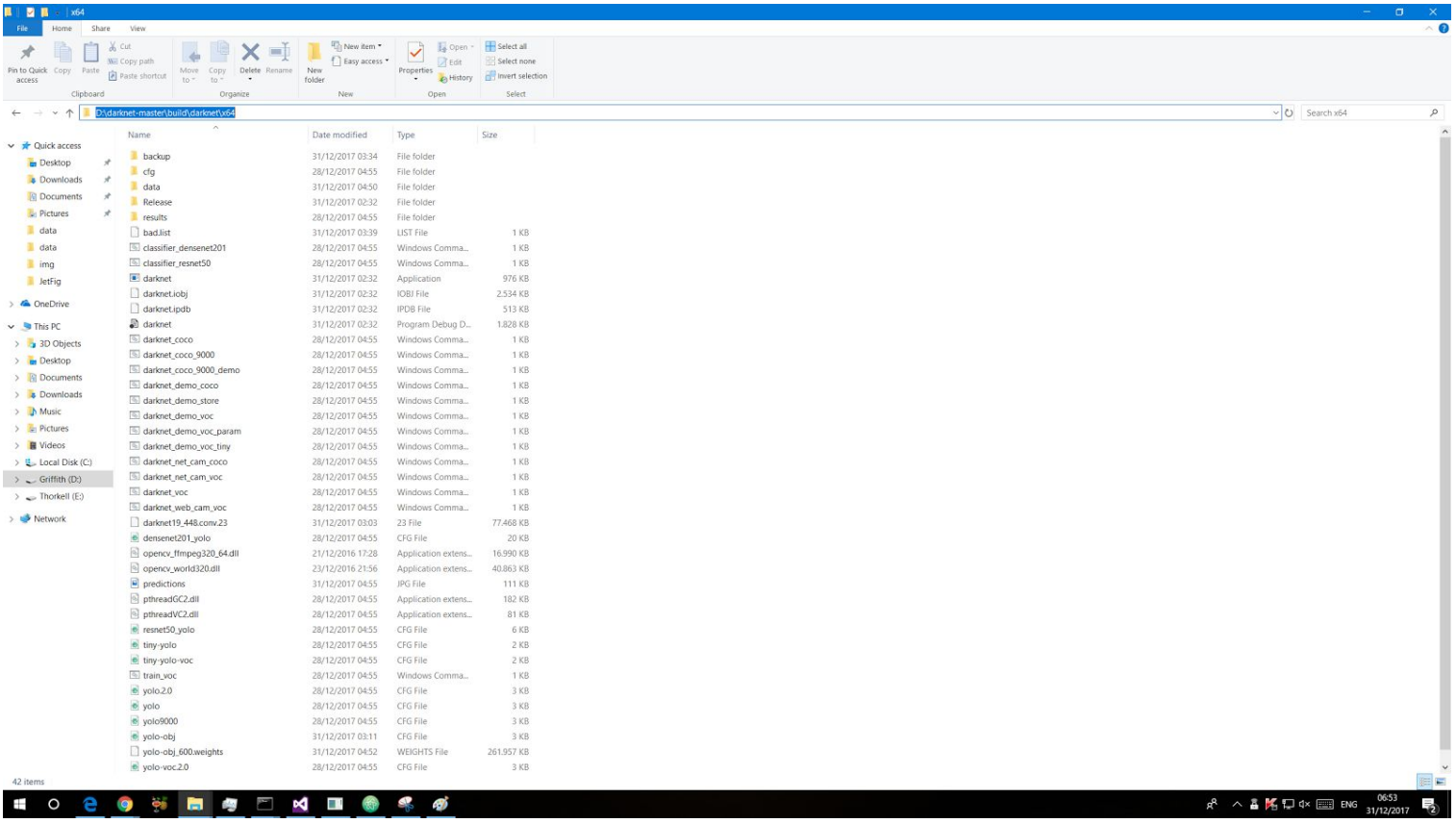
B1: Vào Yolo_mark-master\x64\Release, dùng text editor hay notepad mở yolo-obj.cfg lên, chỉnh classes=1 (số label mà bạn train, ở đồ án này là 1 label) và sau đó chỉnh tới filters với công thức cho filters như sau: **filters=(classes+5)*5**, như vậy với số classes=1 thì filters=30, tương tự với classes=2 thì filters=35 và classes=3 thì filters=40.

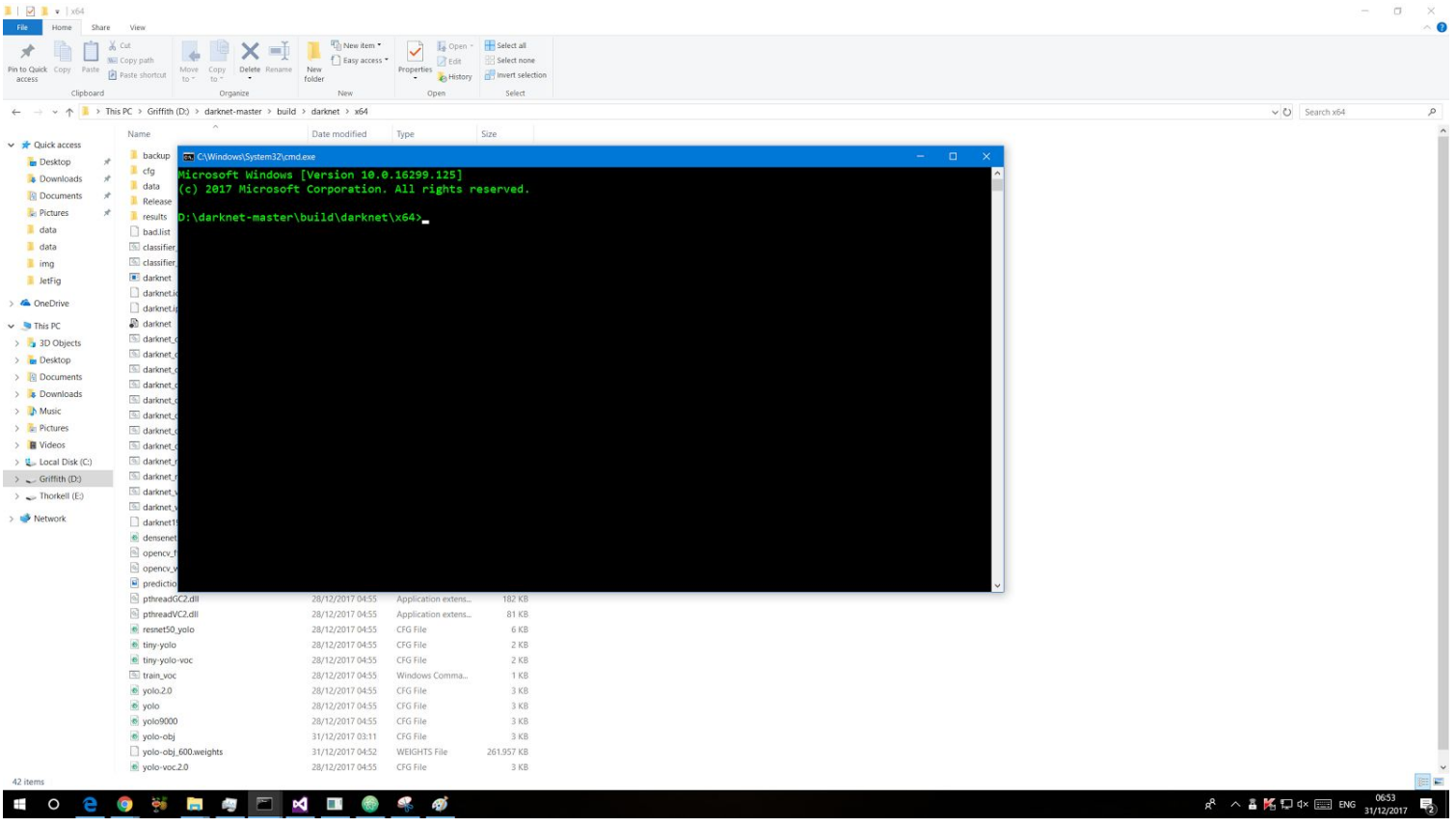
B1*: Ở đây nếu như bạn có một con GPU thuộc loại mạnh, nôm na cỡ GTX 1070 trở lên thì bạn nên chỉnh lại **subdivisions=4** (mặc định là 8) ở yolo-obj.cfg để training nhanh hơn.

B2: Ở **Yolo_mark-master\x64\Release**, bạn copy **file yolo.obj.cfg** và **folder data** sau đó đi qua folder **darknet-master\build\darknet\x64 (**chỗ chứa cái darknet.exe lúc nãy) và paste vào đây, nếu như được hỏi merge hay overwrite thì cứ ok.

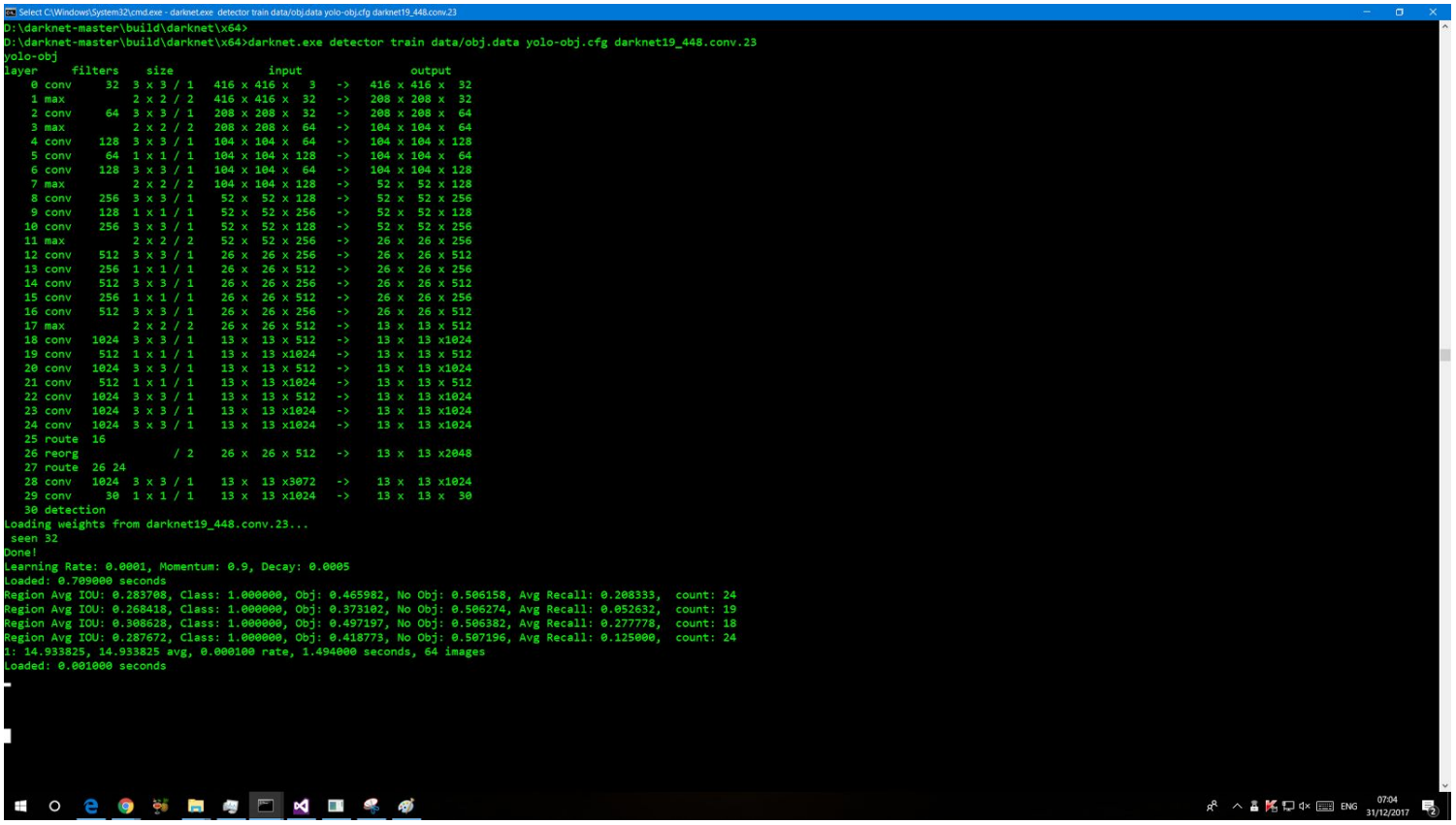
B3: Tải file này về: http://pjreddie.com/media/files/darknet19_448.conv.23 và để cùng folder với darknet.exe (**darknet-master\build\darknet\x64)**

B4: Ở trong folder **darknet-master\build\darknet\x64,** bạn ấn vào cái đường dẫn và nhập vào “cmd” để mở cửa sổ terminal của Windows tới ngay folder này, phía dưới là hình minh họa

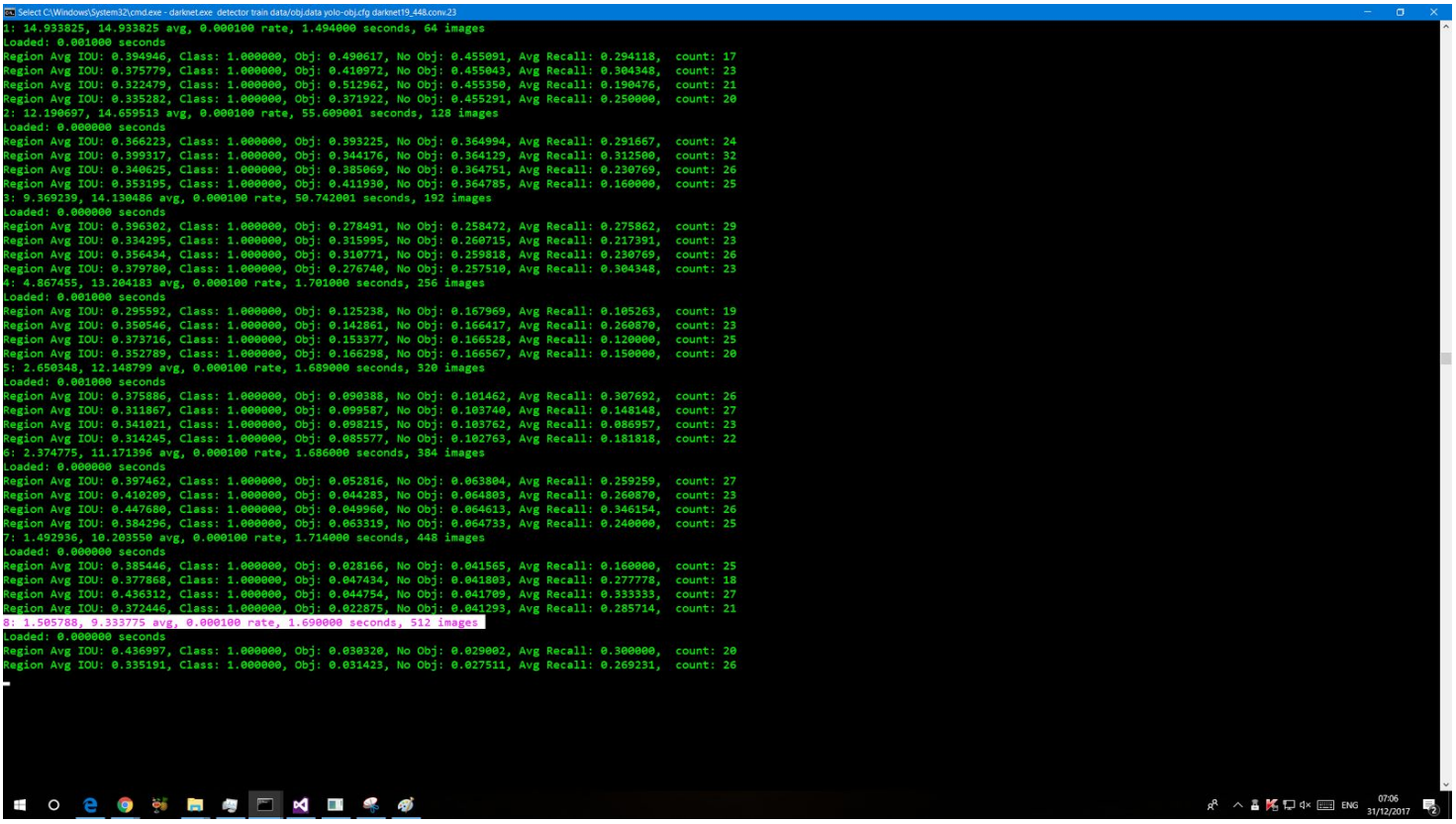




B5: Chạy lệnh: “darknet.exe detector train data/obj.data yolo-obj.cfg darknet19_448.conv.23” và tiến hành train, nếu bạn làm đúng thì sẽ giống như sau:



Ở đây bạn có thể double click chuột trái để pause và để tiếp tục thì có thể ấn Enter.



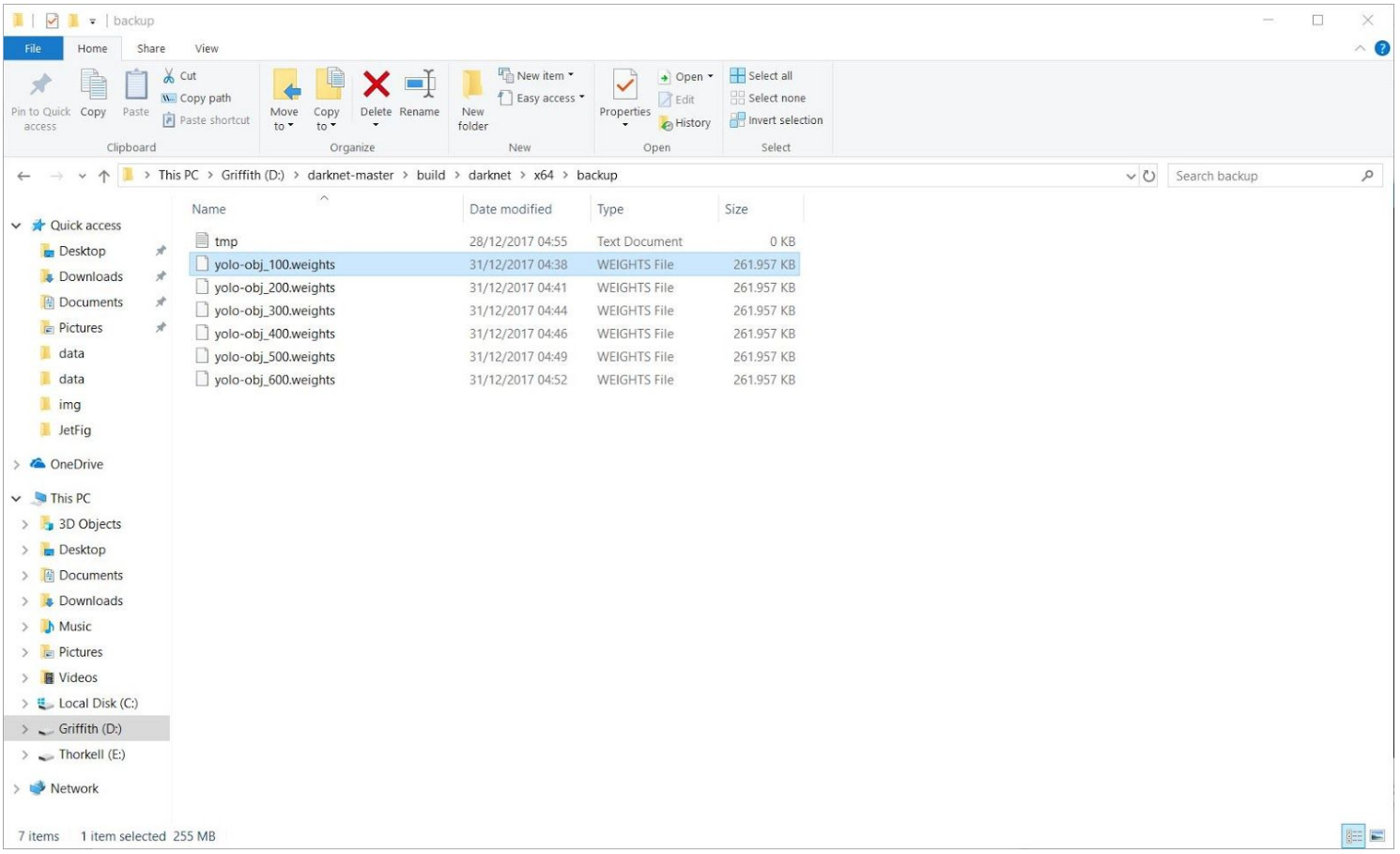
Hình này ở dòng đánh dấu có ý nghĩa như sau:

8: 1.505788, **9.333775 avg**, 0.000100 rate, 1.690000 seconds, 512 images

- Số 8 chỉ số lần iterations
- Số 9.333775 avg, cho ta biết sai số (loss)

Sau mỗi iteration thì sai số sẽ càng giảm đi, ở đây sai số càng nhỏ càng tốt.

Ở đây sau mỗi 100 lần Iterations thì kết quả sẽ được lưu vào darknet-master\build\darknet\x64\backup



Các file **yolo-obj_100.weigths, yolo-obj_200.weigths,...yolo-obj_300.weigths** chính là kết quả train được, càng train lâu càng tốt (ở đây thì mình gợi ý nên dừng khi loss vào khoảng cỡ 0.08 avg).

Để dừng train thì bạn ấn tổ hợp Ctrl + C (tổ hợp này để abort task trong terminal).

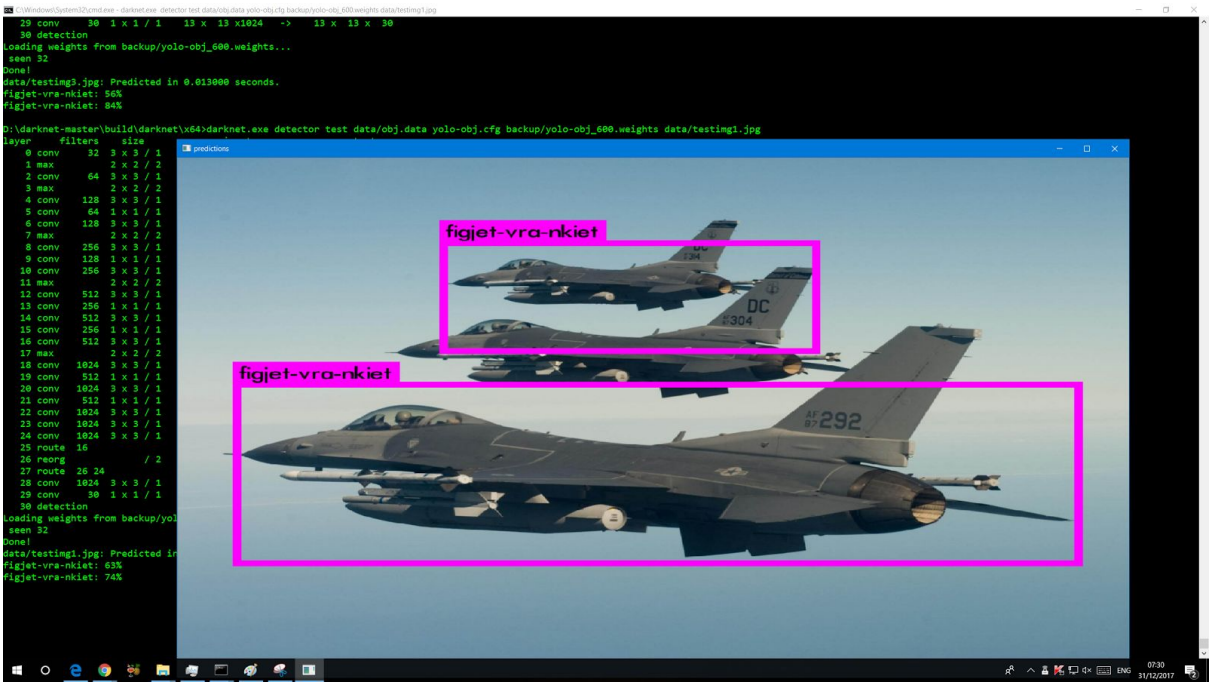
2.11. Tiến hành test - thử nghiệm kết quả

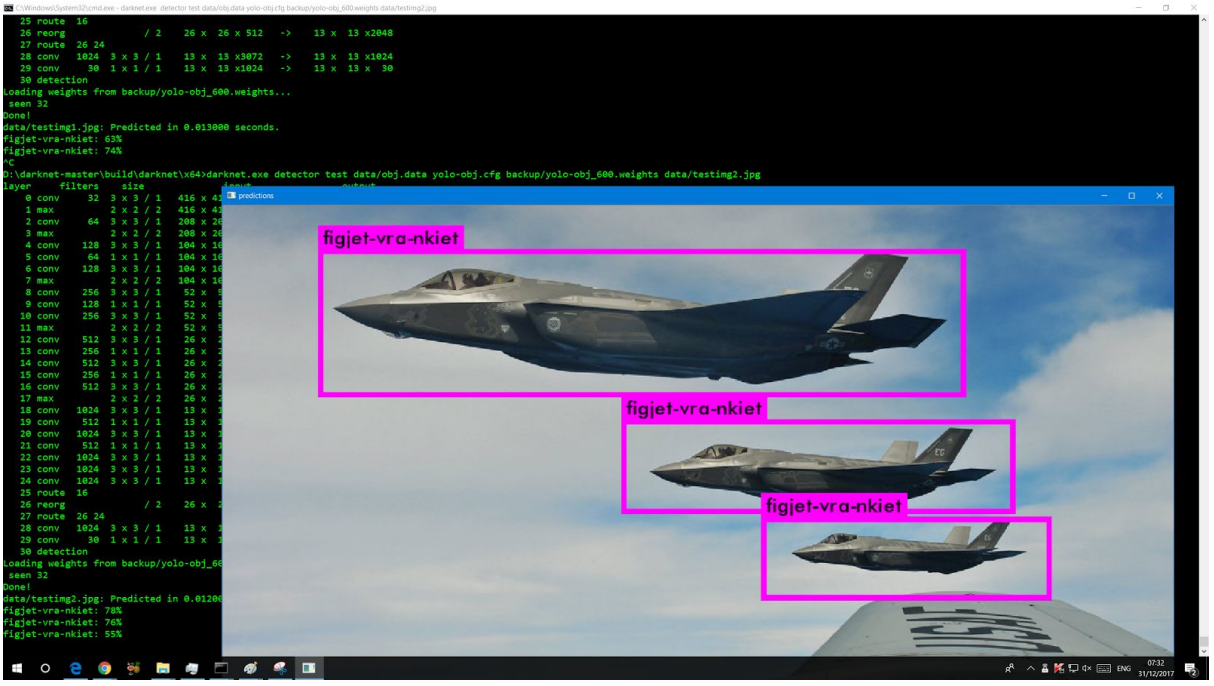
Để tiến hành test, bây giờ bạn tìm thêm một vài cái ảnh nào đó có chứa label của bạn (ở đồ án này cụ thể là cái ảnh nào đó chứa máy bay chiến đấu, mình lên Google Images và tìm vài cái chưa có trong tập 100 cái training) và nếu được thì bạn tìm thêm vài cái video có chứa cái label nữa.

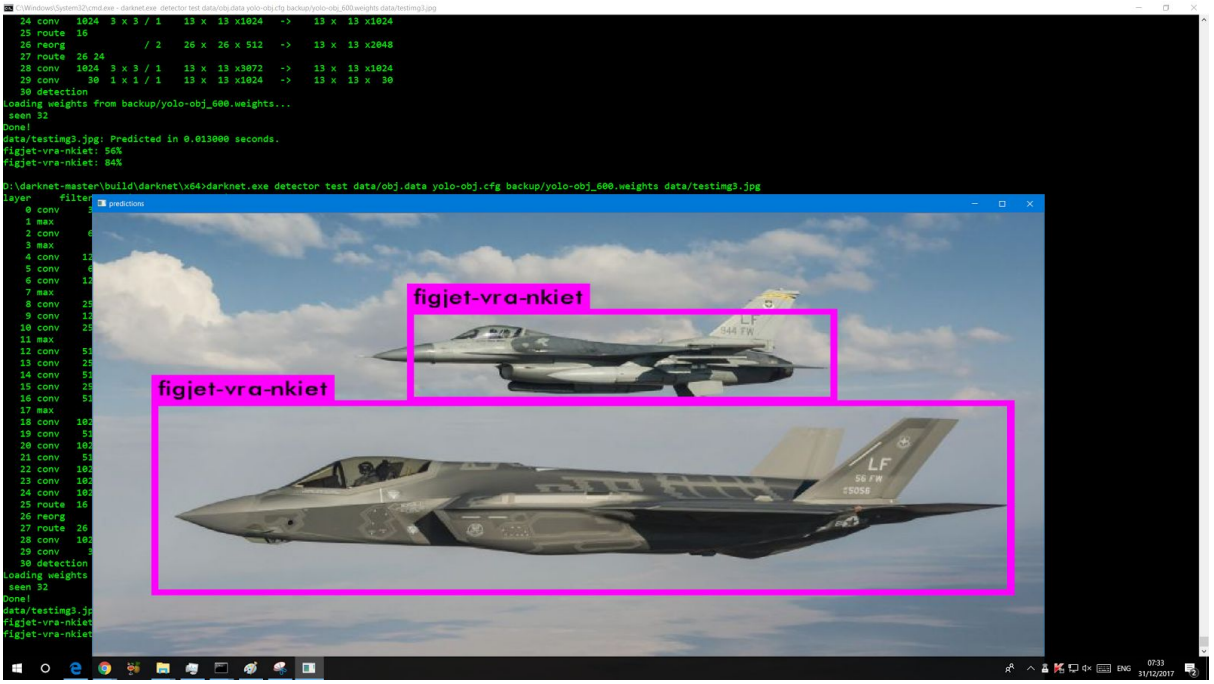
Ở đây mình tìm thêm 3 cái ảnh, đặt tên là: testimg1.jpg, testimg2.jpg, testimg3.jpg và 3 cái video mình down về: testvid1.mp4, testvid2.mp4, testvid3.mp4.

Cho mấy cái ảnh và video này vào **darknet-master\build\darknet\x64\data.** Sau đó chạy lệnh sau (cũng cái terminal mở ở trong folder này):

“**darknet.exe detector test data/obj.data yolo-obj.cfg backup/yolo-obj_600.weights data/testimg1.jpg**” - Test cái hình testimg1.jpg, dùng weight 600, ở đây bạn cứ xài cái weight lớn nhất, để đổi ảnh thì bạn viết câu lệnh tương tự nhưng thay vào là testimg2 hay testimg3. Đây là vài kết quả của tác giả:

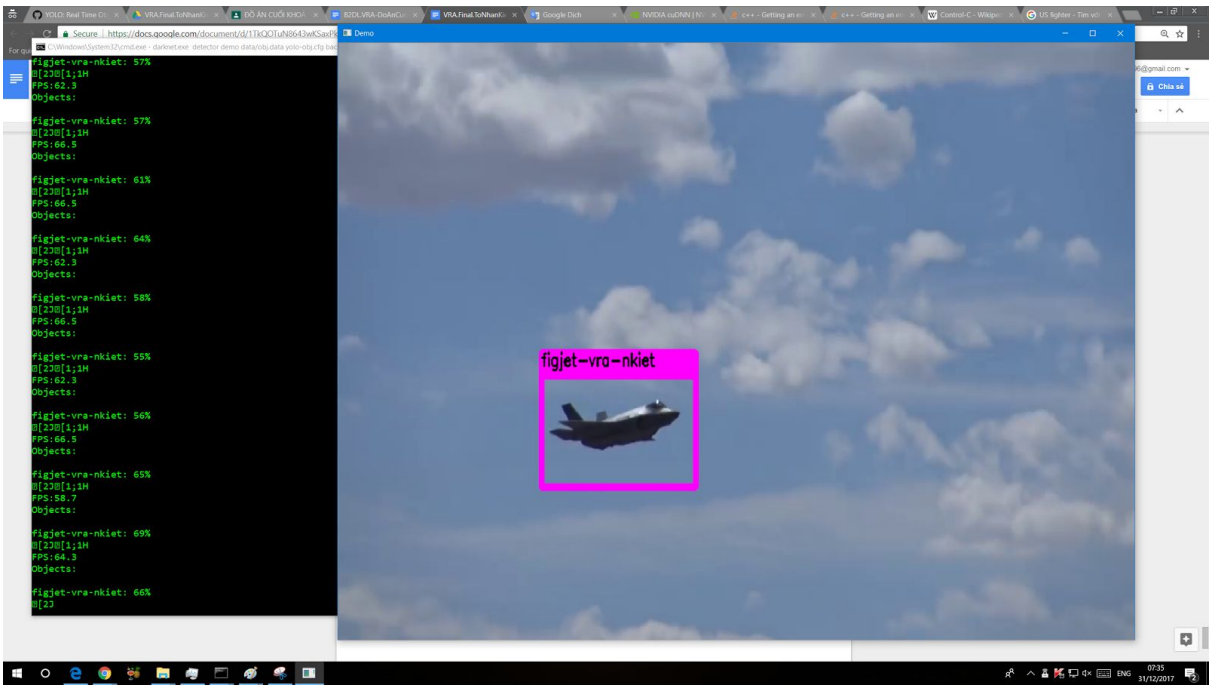






Để chạy test với video, bạn dùng lệnh sau:

“**darknet.exe detector demo data/obj.data yolo-obj.cfg backup/yolo-obj_600.weights data/testvid1.mp4**” - Ở đây bạn vẫn dùng cái weight to nhất là 600, dùng detector demo thay vì detector test, và tương tự ở dưới điền đường dẫn tới video, với các video còn lại thì cứ thay đường dẫn. Kết quả:



3. Đồ án này được hỗ trợ bởi

Nguồn tham khảo (theo APA Style):

- Tijtgat, N. (2017, May 16). How to train YOLOv2 to detect custom objects. Retrieved from https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/
- A. (n.d.). Yolo_mark. Retrieved from https://github.com/AlexeyAB/Yolo_mark
- A. (n.d.). Darknet. Retrieved from https://github.com/AlexeyAB/darknet#you-only-look-once-unified-real-time-object-detection-version-2
- J. (n.d.). *How to train YOLOv2 on custom dataset*. Retrieved March 4, 2017, from https://jumabek.wordpress.com/2017/03/04/how-to-train-yolov2-on-costum-dataset/.

Video được lấy từ: www.youtube.com
Hình ảnh được lấy từ: Google Images

Đặc biệt cảm ơn thầy Lê Đình Duy và Nguyễn Tấn Trần Minh Khang.

HẾT