

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**NGÀNH KHOA HỌC MÁY TÍNH**



**BÁO CÁO ĐỒ ÁN MÔN**  
**NHẬN DIỆN THỊ GIÁC VÀ ỨNG DỤNG**

**PHÁT HIỆN ĐỐI TƯỢNG TRONG ẢNH VÀ VIDEO SỬ DỤNG  
THU VIỆN YOLO – YOU ONLY LOOK ONCE**

Giảng viên hướng dẫn: TS. Lê Đình Duy

Sinh viên thực hiện: Đặng Lê Bảo Chương MSHV : CH1501021

Lớp KHMT - Khóa 10

7/2017

## Mục lục

I.	Giới thiệu.....	3
II.	Cách cài đặt & biên dịch các thư viện CUDA, OpenCV, Darknet Yolo.....	4
1.	Cài đặt CUDA 8, CUDNN 5.1 .....	5
2.	Cài đặt OpenCV 2.4 .....	6
3.	Cài đặt Darknet YOLO.....	6
4.	Demo với pre-trained weight file .....	7
III.	Áp dụng vào bài toán phát hiện loại đối tượng cụ thể - missile launcher.....	7
1.	Đánh dấu ảnh trong tập dữ liệu train.....	8
2.	Training .....	12
3.	Test training model. ....	16
IV.	Tài liệu tham khảo .....	16

## I. Giới thiệu

Đồ án tìm hiểu về cách sử dụng thư viện object detection Darknet YOLOv2 theo hướng dẫn. Học viện đã tự tìm hiểu cách xây dựng một tập dữ liệu gồm **280** ảnh về 1 lớp đối tượng là Missile Launcher, sau đó đánh dấu/nhấn, tiền xử lý để sau đó dùng thư viện YOLOv2 để training sử dụng 2 GPU Nvidia 1050Ti 4GB.

YOLOv2 là một thư viện object detection tiên tiến nhất hiện nay có khả năng phát hiện các đối tượng trong ảnh hoặc video với tốc độ 40-90 khung hình trên giây trên các tập dữ liệu Pascal VOC 2007+2012.

## II. Cách cài đặt & biên dịch các thư viện CUDA, OpenCV, Darknet Yolo

**Các cấu hình về phần cứng và phần mềm như sau:**

**Hardware:** CPU: Core i3 6100; RAM 8GB; VGA: 2 x GTX 1050Ti 4GB; SSD 128GB

**OS:** Ubuntu 14.04 x64

**Libraries :** CMake 3.2, GCC 4.8, Nvidia driver 375, Nvidia CUDA 8, CuDNN 5.1, OpenCV 2.4



*Hình 1. 2 x GPU GTX 1050Ti dùng để training*

## *1. Cài đặt CUDA 8, CUDNN 5.1*

Lên trang: <https://developer.nvidia.com/cuda-downloads>

Chọn Linux → x86\_64 → Ubuntu → 14.04 → deb(local) – Nhấn Download

Chạy lệnh:

```
1. $ sudo dpkg -i cuda-repo-ubuntu1404-8-0-local-ga2_8.0.61-  
   1_amd64.deb  
2. $ sudo apt-get update  
3. $ sudo apt-get install cuda
```

Thêm các dòng sau vào file .bashrc trong thư mục home ( ~/bashrc)

```
1. export CUDA_HOME=/usr/local/cuda-8.0  
2. export LD_LIBRARY_PATH=${CUDA_HOME}/lib64  
3. PATH=${CUDA_HOME}/bin:${PATH}  
4. export PATH
```

Dùng lệnh để các cấu hình trên có hiệu lực

```
1. $ source ~/.bashrc
```

Sau khi cài xong thử bằng lệnh “nvcc --version”

```
deeppc@deeppc-desktop:~/Downloads$ nvcc --version  
nvcc: NVIDIA (R) Cuda compiler driver  
Copyright (c) 2005-2016 NVIDIA Corporation  
Built on Tue_Jan_10_13:22:03_CST_2017  
Cuda compilation tools, release 8.0, V8.0.61
```

Cài đặt CuDNN 5.1

Vào trang <https://developer.nvidia.com/cudnn> đăng ký một tài khoản Developer (Free). Sau khi đăng ký và đăng nhập thành công thì tải mục sau về:

CuDNN v5.1 (August 10, 2016), for CUDA 8 cuDNN v5.1 Library for Linux

Sau khi tải xong và giải nén file .tar (tar xvf cudnn-8.0-linux-x64-v5.1.tgz ) copy thư viện CuDNN vào các thư mục sau

```
1. $ cd ~/Downloads/cuda  
2. $ sudo cp lib64/* /usr/local/cuda/lib64/  
3. $ sudo cp include/cudnn.h /usr/local/cuda/include/
```

## 2. Cài đặt OpenCV 2.4

Run file script đính kèm trong thư mục install-opencv.sh

<https://gist.github.com/dynamicguy/3d1fce8dae65e765f7c4>

## 3. Cài đặt Darknet YOLO

```
1. git clone https://github.com/pjreddie/darknet  
2. cd darknet
```

Chỉnh các thông số sau trong Makefile để compile darknet với CUDA, CuDNN và Open CV

Chỉnh **GPU=1, CUDNN=1, OPENCV=1**

Command dòng Arch có nhiều tuỳ chọn thêm dùng mới với nội dung:

**ARCH= -gencode arch=compute\_61, code=compute\_61** (phụ thuộc vào GPU)

```
GPU=1  
CUDNN=1  
OPENCV=1  
OPENMP=0  
DEBUG=0

#ARCH= -gencode arch=compute_20,code=[sm_20,sm_21] \
#      -gencode arch=compute_30,code=sm_30 \
#      -gencode arch=compute_35,code=sm_35 \
#      -gencode arch=compute_50,code=[sm_50,compute_50] \
#      -gencode arch=compute_52,code=[sm_52,compute_52]

# This is what I use, uncomment if you know your arch and want to
ARCH= -gencode arch=compute_61,code=compute_61

VPATH=./src/./examples
SLIB=libdarknet.so
ALIB=libdarknet.a
EXEC=darknet
OBJDIR=./obj/

CC=gcc
NVCC=nvcc
```

*Hình 2. Darknet build config*

Save lại và gõ lệnh make

```
1. $ cd darknet  
2. $ make
```

#### 4. Demo với pre-trained weight file

Download pre-trained file:

```
$ wget https://pjreddie.com/media/files/yolo.weights
```

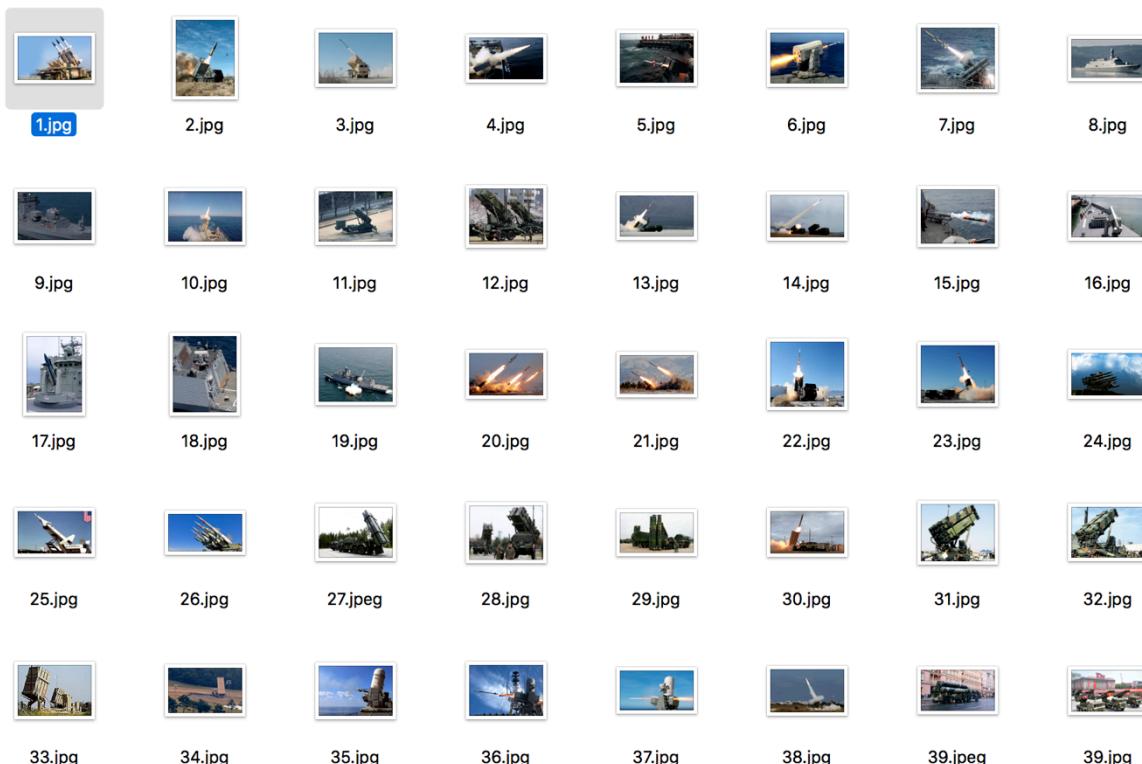
Sau đó chạy thử:

```
$ ./darknet detect cfg/yolo.cfg yolo.weights data/dog.jpg
```

Videoclip demo chạy thử với Webcam:

<https://youtu.be/a9yziZzP5x4>

### III. Áp dụng vào bài toán phát hiện loại đối tượng cụ thể - missile launcher



Hình 3 Dataset Missile launcher

Trong phần này em sẽ trình bày cách để xây dựng tập dữ liệu training theo ý muốn và cách dùng YOLO để train ra

Để giảm thời gian training, em chọn tập dữ liệu của em chỉ chuẩn bị tập dữ liệu có 1 lớp là Missile Launcher (Bệ phóng tên lửa)

Để train dữ liệu liên quan đến thị giác máy tính, việc đầu tiên là cần có hình ảnh. Theo các tài liệu hướng dẫn thì thư viện Darknet **YOLOv2** cần khoảng **300** ảnh cho 1 lớp để đạt hiệu quả. Tập dữ liệu của em gồm **280** ảnh liên quan tới Missile launcher (Nguồn từ Google Image với từ khoá “Missile launcher”, “Missile defense system”)

### 1. Dánh dấu ảnh trong tập dữ liệu train

Sau khi đã tìm đủ ảnh, bước kế tiếp chúng ta sẽ đánh dấu (annotate) những đối tượng cần phát hiện (detect) trong ảnh, để thực hiện bước này chúng ta sử dụng công cụ BBox Label Tool (viết bằng Python) nằm trong thư mục **darknet/annotate\_tool** đánh dấu:

**Bước 1:** chuyển vào thư mục **annotate\_tool** và copy tất cả các ảnh vào trong thư mục **annotate\_tool/Images/001/** (Tạo thư mục nếu chưa có), các ảnh loại đối tượng sẽ nằm trong từng thư mục riêng biệt, Vd: 002,003,...

```
[→ darknet git:(master) ls
LICENSE           LICENSE.mit      annotate_tool  examples      obj          scripts     training_steps
LICENSE.gen       LICENSE.v1       cfg            include      process.py   src
LICENSE.gpl       Makefile        darknet        libdarknet.a  python      test.txt
LICENSE.meta      README.md      data           libdarknet.so results    train.txt
[→ darknet git:(master) cd annotate_tool
[→ annotate_tool git:(master) python main.py ]
```

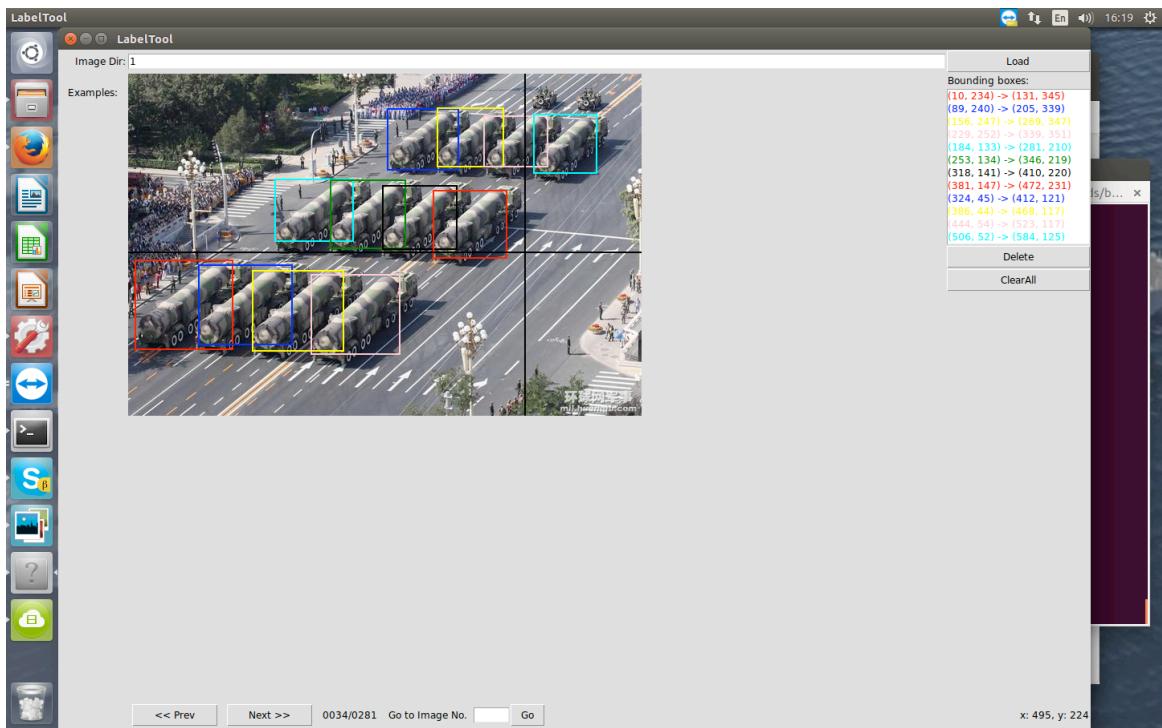
Nếu bị lỗi thì thử cài thêm gói thư viện **python-tk**

```
| 1. $ sudo apt-get install python-tk
```

Trước khi chạy file **main.py** dùng trình text editor chỉnh dòng **128** của file **main.py** thành thư mục hiện tại sau đó save lại và dùng lệnh “**python main.py**” để chạy

```
117
118      # for debugging
119 ##      self.setImage()
120 ##      self.loadDir()
121
122      def loadDir(self, dbg = False):
123          if not dbg:
124              s = self.entry.get()
125              self.parent.focus()
126              self.category = int(s)
127          else:
128              s = r'/Users/chuongdang/Documents/YOLO/darknet/annotate_tool/'
```

**Bước 2:** Annotate



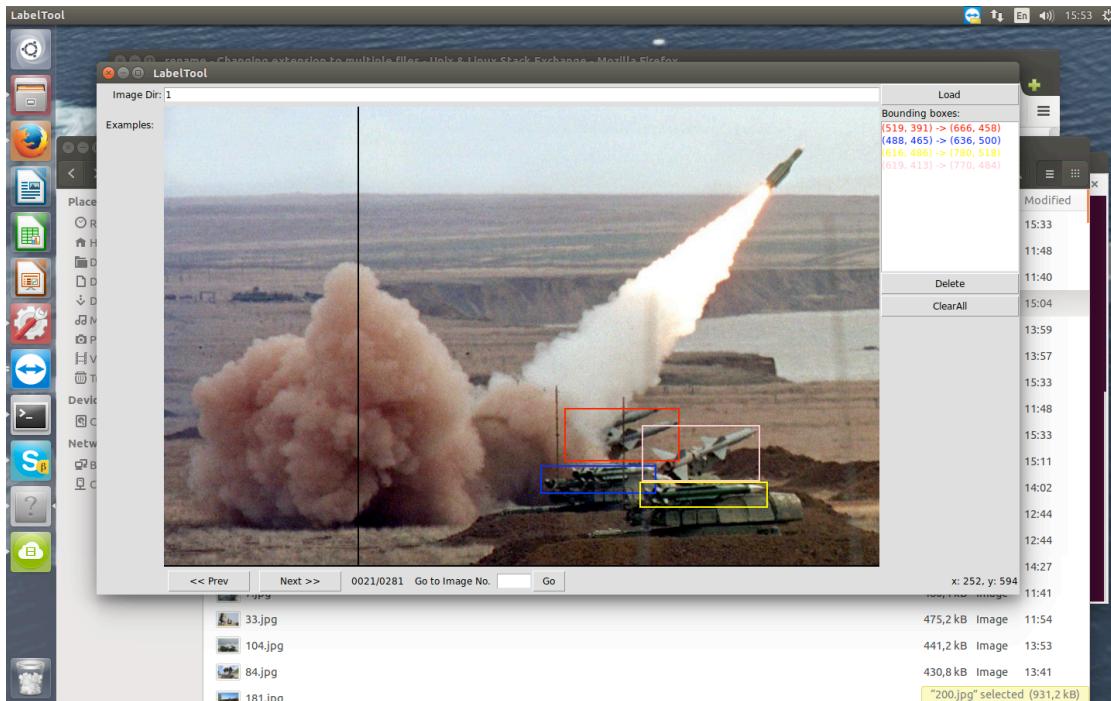
**Hình 4 Đánh dấu các đối tượng trong ảnh**

Rê chuột sao cho các khung bao hình chữ nhật bao lấy đối tượng cần phát hiện rồi nhấp chuột trái để xác định, thông số tọa độ sẽ hiện ở khung bên trái tiếp tục cho tất cả các đối tượng cần nhận diện. Trường hợp muốn xoá các khung bao thì chọn vào dòng tọa độ cùng màu trong khung bounding box → Delete. Chọn Clear All để xoá hết.

Sau khi xong 1 ảnh thì chọn Next >> để tiếp tục cho tới ảnh cuối cùng sau khi xong thì thoát chương trình.

**Lưu ý:** do chương trình đánh dấu ảnh chỉ nhận dạng file mở rộng \*.jpg nên các ảnh PNG, JPEG (.jpeg) cần được chuyển đổi qua jpg trước khi chạy ứng dụng annotate. Có thể dùng các thư viện sau:

- Window: <https://sourceforge.net/projects/bulkimageconverter/>
- Linux/Mac: ImageMagick (sudo apt-get install imagemagick)



*Hình 5 Đánh dấu các đối tượng trong ảnh*

#### Bước 4: Convert label thành định dạng của Darknet YOLO

Sau khi kết thúc việc đánh dấu, chúng ta sẽ thu được các files labels trong thư mục *Annotate\_tool/Labels/001*. Đến đây chúng ta cần một bước xử lý nữa nữa đổi chuyển đổi các label qua định dạng của Darknet YOLO. Một file script đã được chuẩn bị để làm việc này (*convert.py*)

Trước tiên copy thư mục (gồm các ảnh) *Images/001* ra thêm *Images/missile\_launcher* và tạo thư mục *Labels/missile\_launcher*. **Lưu ý:** tên thư mục mới copy ra là tên lớp đối tượng cần nhận dạng, trường hợp chúng ta có nhiều đối tượng cần train chúng ta sẽ lần lượt copy chúng ra các thư mục mới.

Sửa file *convert.py* như sau

```

convert.py (~/Documents/YOLO/darknet/annotate_tool) - VIM1
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 9 14:55:43 2015
This script is to convert the txt annotation files to appropriate format needed by YOLO
@author: Guanghan Ning
Email: gnxr9@mail.missouri.edu
"""

import os
import re
from os import walk, getcwd
from PIL import Image

classes = ["missile_launcher"]

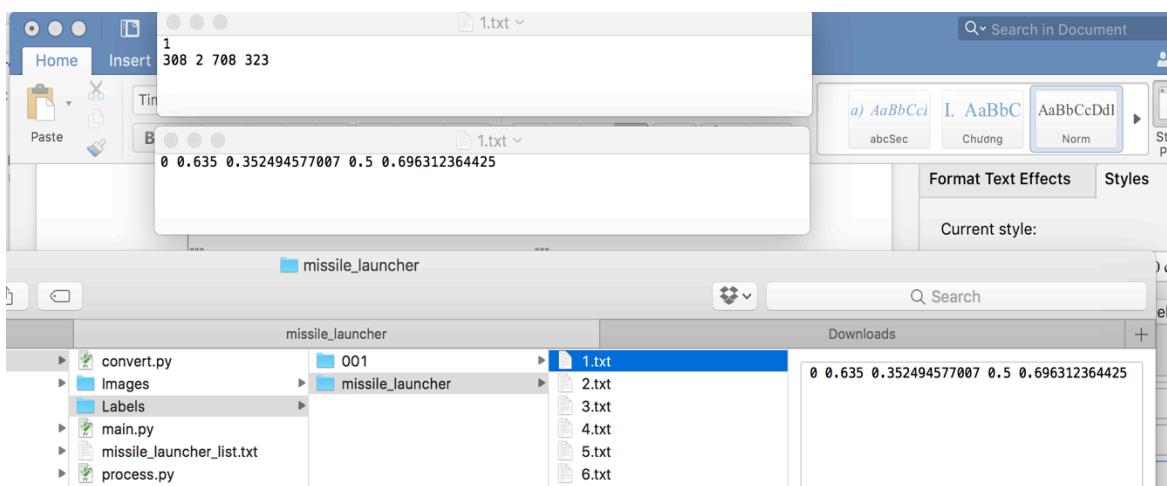
def convert(size, box):
    dw = 1./size[0]
    dh = 1./size[1]
    x = (box[0] + box[1])/2.0
    y = (box[2] + box[3])/2.0
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)

"""
Configure Paths
mypath = "Labels/001/"
outpath = "Labels/missile_launcher/"
"""


```

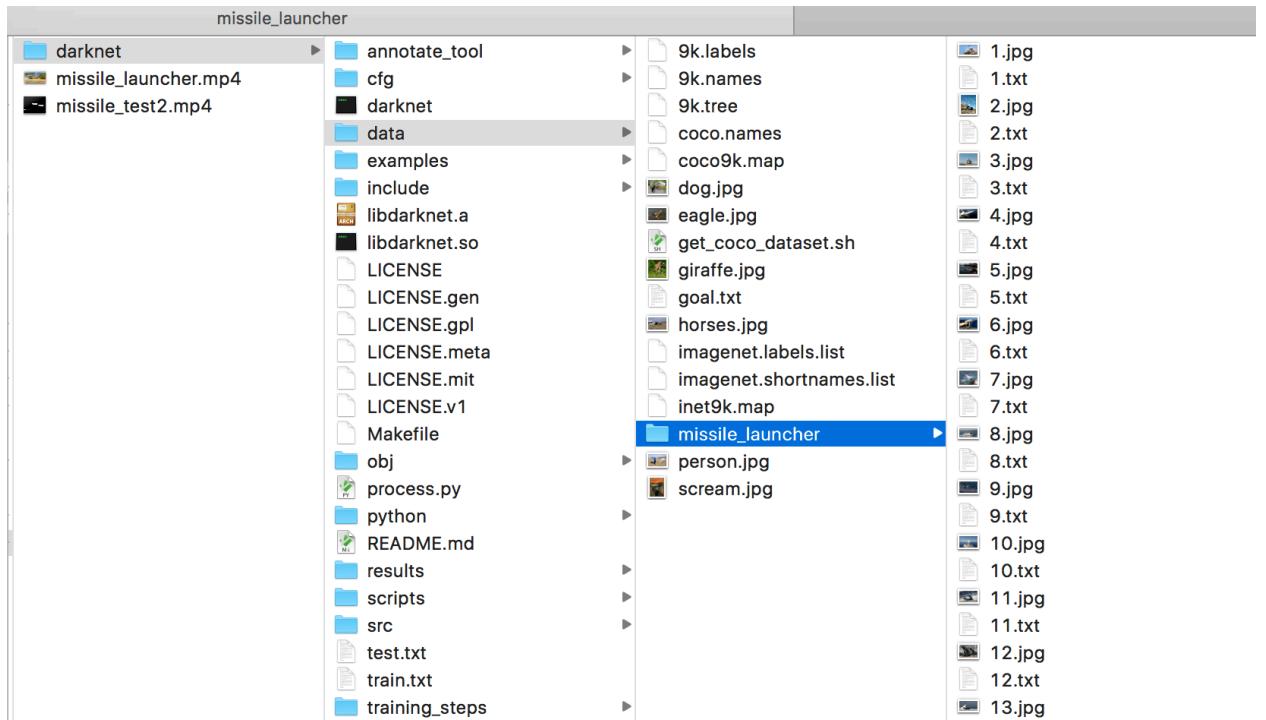
**Hình 6. Chính sửa file convert.py**

Sửa dòng số 16, thành lớp đối tượng cần nhận diện (missile\_launcher) trùng tên với folder ảnh vừa copy ra ở bước trên. Chính lại dòng số 25 cho đúng với các label vừa đánh dấu bằng ứng dụng, dòng 36 thành tên lớp. Sau đó save lại. Vào chạy file convert.py, sau khi chạy xong chúng ta sẽ có các file \*.txt ở thư mục Labels/missile\_launcher/ đã được chuyển đổi



**Hình 7. So sánh 2 định dạng, label YOLO format (1.txt bên dưới)**

Đến đây dữ liệu đã sẵn sàng để trên chúng ta copy tất cả các nhãn trong thư mục Labels/missile\_launcher và tất cả các ảnh trong thư mục Images/missile\_launcher vào trong thư mục darknet/data/missile\_launcher



**Hình 8. Cấu trúc thư mục cho dataset chuẩn bị train**

## 2. Training

Trước khi train, chúng ta cần chuẩn bị thêm một số file cấu hình như sau:

- **darknet/cfg/yolo-missile.data**: file này chứa đường dẫn trỏ tới các file chỉ dẫn tới thư mục của tập dữ liệu train (ảnh và label) và tập dữ liệu sẽ được dùng để test (ảnh và label).

```
[→ cfg git:(master) ✘ cat yolo-missile.data
classes = 1
train = train.txt
test = test.txt
names = cfg/obj-missile.names
backup = backup/
```

- **obj-missile.names**: chứa tên các lớp đối tượng cần nhận dạng (tên mỗi đối tượng 1 dòng). Trong trường hợp này, chúng ta chỉ có 1 loại là Missile Launcher

- **train.txt** và **test.txt**: chứa đườ sẽ nằm ngoài thư mục gốc darknet: chúng ta sẽ dùng python script (process.py) để sinh ra 2 file này, lưu ý chỉnh lại đường dẫn trong process.py cho đúng file đường dẫn lưu tập dữ liệu train. Trong file process.py em dùng 10% ảnh trong train để validate trong lúc training (có thể chỉnh lại thông số này bằng biến *percentage\_test*)

```

test.txt
data/missile_launcher/156.jpg
data/missile_launcher/213.jpg
data/missile_launcher/245.jpg
data/missile_launcher/221.jpg
data/missile_launcher/147.jpg
data/missile_launcher/279.jpg
data/missile_launcher/137.jpg
data/missile_launcher/22.jpg
data/missile_launcher/131.jpg
data/missile_launcher/17.jpg
data/missile_launcher/70.jpg
data/missile_launcher/270.jpg
data/missile_launcher/201.jpg
data/missile_launcher/271.jpg
data/missile_launcher/58.jpg
data/missile_launcher/7.jpg
data/missile_launcher/275.jpg
data/missile_launcher/113.jpg
data/missile_launcher/142.jpg
data/missile_launcher/266.jpg
data/missile_launcher/259.jpg
data/missile_launcher/164.jpg
data/missile_launcher/140.jpg
data/missile_launcher/123.jpg
data/missile_launcher/78.jpg
data/missile_launcher/102.jpg

train.txt
data/missile_launcher/240.jpg
data/missile_launcher/217.jpg
data/missile_launcher/186.jpg
data/missile_launcher/1.jpg
data/missile_launcher/180.jpg
data/missile_launcher/263.jpg
data/missile_launcher/4.jpg
data/missile_launcher/3.jpg
data/missile_launcher/90.jpg
data/missile_launcher/54.jpg
data/missile_launcher/144.jpg
data/missile_launcher/216.jpg
data/missile_launcher/232.jpg
data/missile_launcher/121.jpg
data/missile_launcher/200.jpg
data/missile_launcher/115.jpg
data/missile_launcher/163.jpg
data/missile_launcher/215.jpg
data/missile_launcher/238.jpg
data/missile_launcher/92.jpg
data/missile_launcher/69.jpg
data/missile_launcher/182.jpg
data/missile_launcher/187.jpg
data/missile_launcher/257.jpg
data/missile_launcher/141.jpg
data/missile_launcher/71.jpg
data/missile_launcher/104.jpg
data/missile_launcher/135.jpg
data/missile_launcher/101.jpg
data/missile_launcher/274.jpg
data/missile_launcher/96.jpg
data/missile_launcher/80.jpg
data/missile_launcher/260.jpg
data/missile_launcher/253.jpg
data/missile_launcher/82.jpg
data/missile_launcher/168.jpg
data/missile_launcher/242.jpg
data/missile_launcher/244.jpg
data/missile_launcher/208.jpg
data/missile_launcher/122.jpg
data/missile_launcher/84.jpg
data/missile_launcher/102.jpg

process.py (~/Documents/YOLO/darknet) - VIM2
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)
# Directory where the data will reside, relative to 'darknet.exe'
path_data = 'data/missile_launcher/'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('train.txt', 'w')
file_test = open('test.txt', 'w')
# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndfilename in glob.iglob(os.path.join(current_dir, path_data, "*.jpg")):
    title, ext = os.path.splitext(os.path.basename(pathAndfilename))
    print(pathAndfilename)
    if counter == index_test:
        counter = 1
        file_test.write(path_data + title + '.jpg' + "\n")
    else:
        file_train.write(path_data + title + '.jpg' + "\n")
    counter = counter + 1

```

**Hình 9. Sinh ra đường dẫn train data và test data**

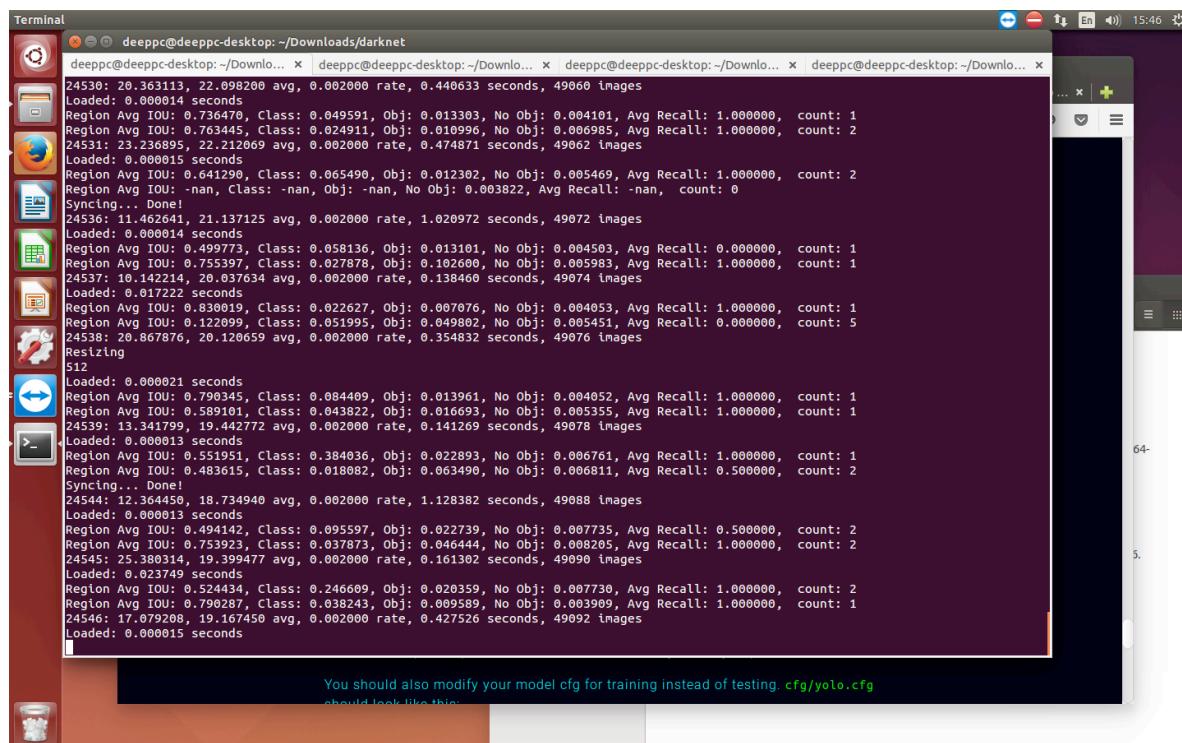
- **darknet/cfg/yolo-missile.cfg**: đây là file cấu hình các layer trong mô hình Convolutional Neural Network. Để tạo file này, em copy nội dung từ file cấu hình có sẵn là **yolo-voc.cfg** và thay đổi một số thông số sau
  - Dòng 6: gán batch=64, chúng ta sẽ sử dụng 64 ảnh cho mỗi bước training

- Dòng **7**: gán subdivisions=8, batch sẽ được chia cho 8 để giảm lượng VRAM sử dụng bởi GPU, nếu chúng ta có GPU mạnh với nhiều VRAM (11GB) thì số này có thể giảm và có thể tăng batch để tăng tốc độ train).
  - Dòng **244**: chỉnh lại số class là 1 (chúng ta chỉ có 1 class)
  - Dòng **237**: chỉnh filters=(số class + 5)\*5 (chúng ta có 1 nên số này là 30).

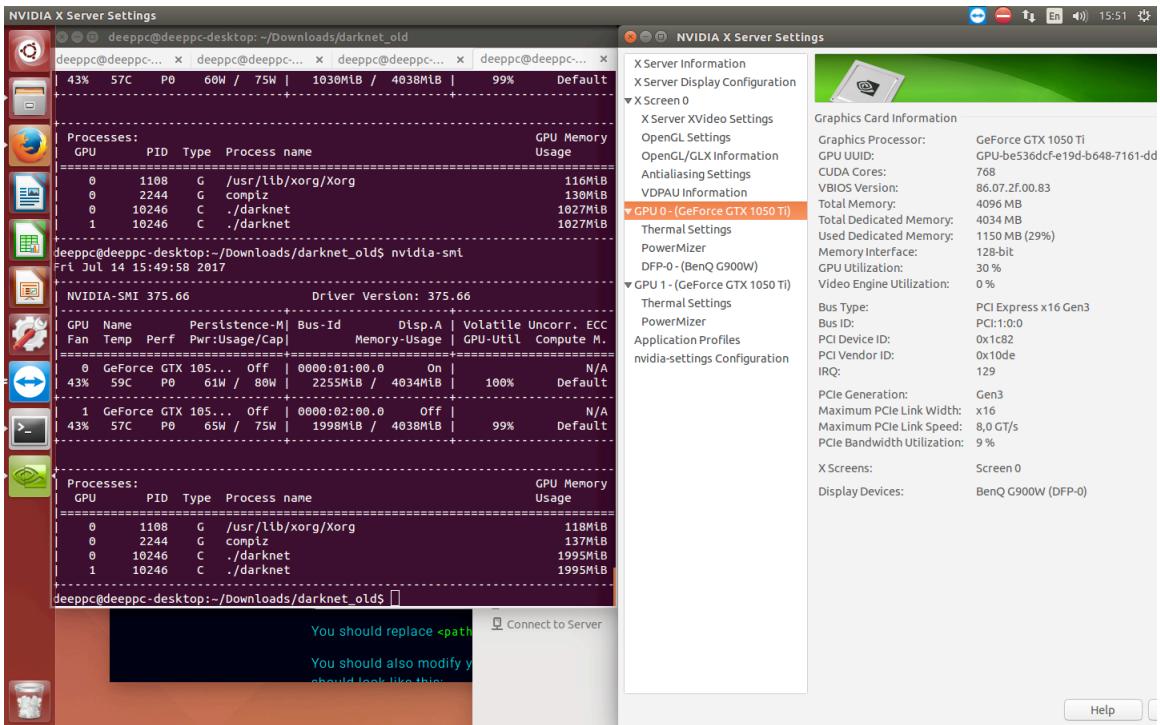
Để train YOLOv2 cần một tập các convolutional weights đã được train trước trên ImageNet ([https://pjreddie.com/media/files/darknet19\\_448.conv.23](https://pjreddie.com/media/files/darknet19_448.conv.23))

Bắt đầu train với câu lệnh:

```
Darknet$ ./darknet detector train cfg/yolo-missile.data  
cfg/yolo-missile.cfg darknet19_448.conv.23 -gpus 0,1
```

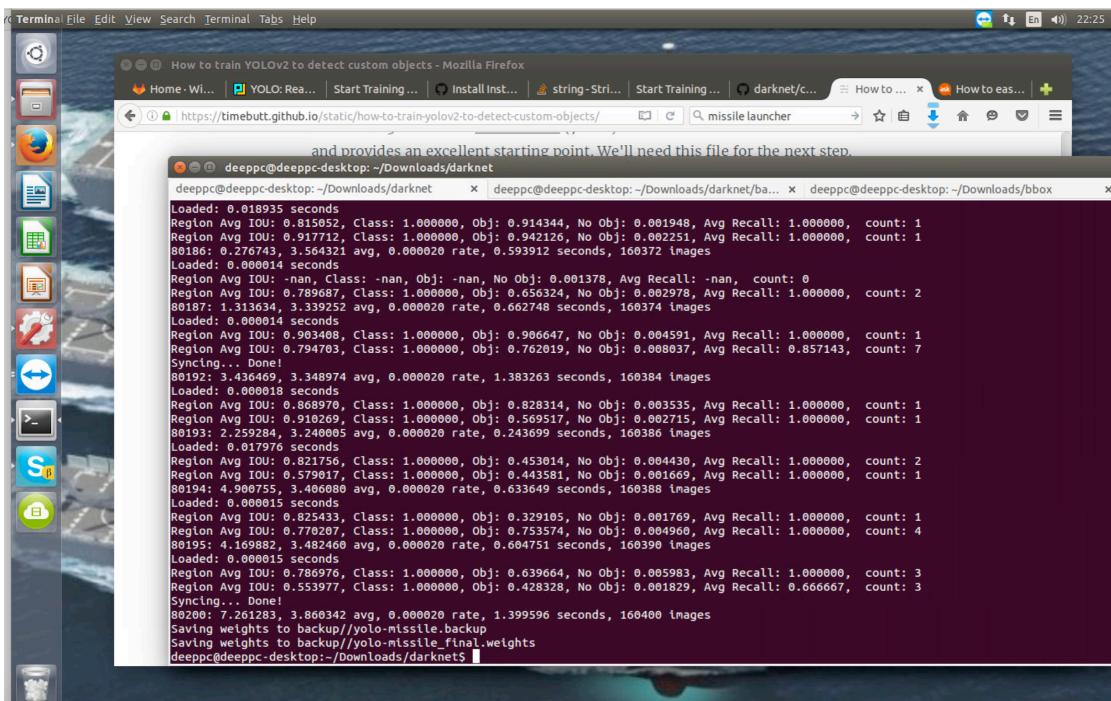


**Hình 10. Start training**



**Hình 11. Tải GPUs trong lúc training**

Sau khi training xong chúng ta sẽ thu được file yolo-missile.weights → đây là file chứa training model sẽ được sử dụng cho quá trình phát hiện (detection).



**Hình 12. Hoàn tất quá trình training sau khoảng 3 tiếng**

### *3. Test training model.*

Đoạn video clip được sử dụng để test có thể tải ở [đây](#).

Câu lệnh để test:

```
Darknet$ ./darknet detector train cfg/yolo-missile.data  
cfg/yolo-missile.cfg missile_launcher
```

**Link Video demo:**

Full: <https://youtu.be/WfGBcwG0lLU>

With command: <https://youtu.be/gmU3ubYl9vc>

## IV. Tài liệu tham khảo

- [1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).
- [2] <https://pjreddie.com/darknet/yolo/>
- [3] <https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/>