

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
Phòng Đào tạo Sau đại học & Khoa học công nghệ



BÀI TOÁN NHẬN DIỆN ĐỐI TƯỢNG DỰA TRÊN BỘ THƯ VIỆN YOLO

Học viên thực hiện:

- CH1701006 – Trần Nguyên Phúc

Giảng viên hướng dẫn:

- TS Lê Đình Duy
- TS Nguyễn Tấn Trần Minh Khang

TP. Hồ Chí Minh – tháng 12/2017

Mục Lục

Lời nói đầu.....	2
I. Giới thiệu bài toán nhận dạng đối tượng.....	3
1. Khái niệm nhận dạng đối tượng	3
2. Một số kỹ thuật và thư viện sử dụng cho bài toán	3
II. Giới thiệu thư viện YOLO	4
1. Giới thiệu thư viện.....	4
III. Cài đặt và chạy thử nghiệm cho bài toán sử dụng YOLO với tập dữ liệu được xây dựng sẵn (Pre-training Model)	7
1. Cài đặt thư viện darknet.....	7
2. Chạy ứng dụng dựa trên bộ dữ liệu YOLO được xây dựng sẵn	8
3. Xây dựng bộ dữ liệu training dựa trên YOLO	11
Bài Đọc Thêm: THUẬT TOÁN TRÍCH RÚT ĐẶC TRƯNG LBP	15
(LOCAL BINARY PATTERN).....	15

Lời nói đầu

Bài toán nhận dạng đối tượng là một trong những bài toán đang được sự chú ý của giới công nghệ cũng như các công ty hàng đầu thế giới trong lĩnh vực công nghệ thông tin – truyền thông như Google, Facebook, Microsoft,... Một hệ thống nhận dạng đối tượng tìm thấy các đối tượng trong thế giới thực từ những hình ảnh đầu vào thông qua mô hình đồ tượng. Những kỹ thuật này khá khó khăn, con người chúng ta có thể dễ dàng nhận biết đối tượng một cách dễ dàng và gần như ngay lập tức. Trong bài đồ án này em xin phép trình bày, thảo luận về những kiến thức cơ bản nhất của bài toán nhận dạng đối tượng mà một hệ thống thị giác thường sử dụng với thư viện YOLO.

Thông qua môn học “nhận dạng thị giác và ứng dụng” em có cơ hội tìm hiểu, thực hiện đồ án với những kiến thức được trang bị từ môn học. Đồng thời, em cũng xin gửi lời cảm ơn đến **TS Lê Đình Duy** và **TS Nguyễn Tấn Trần Minh Khang** đã giảng dạy những kiến thức cần thiết và gợi ý giúp em thực hiện đồ án này.

Link github: https://github.com/nerostamas/uit_var

Link youtube: <https://www.youtube.com/watch?v=bUd4DyOmdvs&feature=youtu.be>

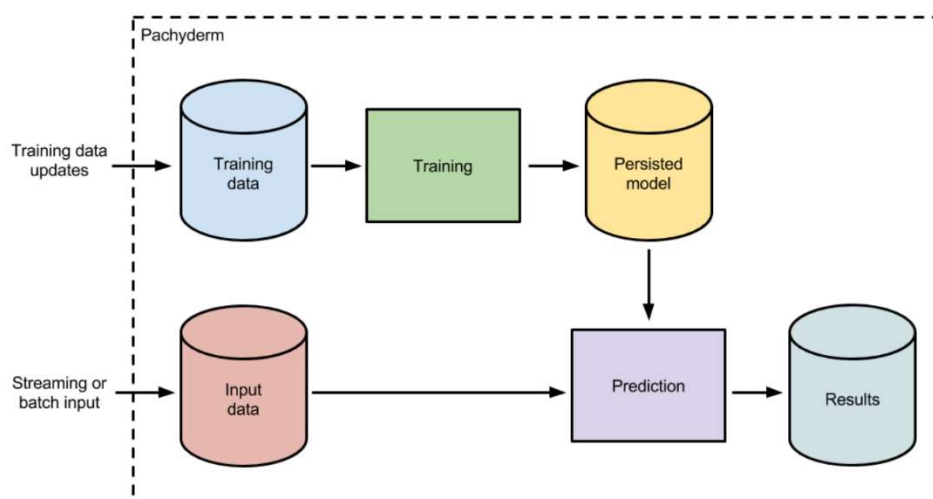
I. Giới thiệu bài toán nhận dạng đối tượng

1. Khái niệm nhận dạng đối tượng

Nhận dạng đối tượng là một nhánh của thị giác máy tính và có thể được hiểu như một bài toán phân lớp, giúp máy tính phân biệt được các lớp khác nhau từ đầu vào (ảnh, video). Mỗi lớp đối tượng đều có các đặc trưng riêng biệt để phân loại lớp, ví dụ như các vòng tròn đều tròn ở 2 điểm bất kỳ trên đường biên. Ta có thể phát hiện đối tượng bằng cách sử dụng các đặc tính riêng biệt này. Ví dụ: tìm kiếm các vòng tròn trong một ảnh, tìm kiếm các đối tượng ở một khoảng cách cụ thể. Tương tự như vậy, khi tìm kiếm các ô vuông, các vật được vuông góc với góc và có độ dài các cạnh bằng nhau là một đặc trưng. Cách tiếp cận tương tự như vậy cũng có thể ứng dụng vào trong bài toán nhận diện khuôn mặt, nơi ta có thể dễ dàng tìm ra các đặc trưng của con người như mũi, mắt và môi dựa trên tỷ lệ khoảng cách, độ đặc trưng màu sắc,...

2. Một số kỹ thuật và thư viện sử dụng cho bài toán

Lợi thế mà chúng ta đang có ở đây là một hình ảnh đầu vào được tạo ra bởi các điểm ảnh. Vì vậy, trong hầu hết các trường hợp chúng ta biết vị trí của điểm tiếp theo, nó sẽ được kết nối với điểm (các điểm) hiện tại của nó thông qua một đặc trưng cụ thể. Lấy ví dụ như hình tròn, sau khi chụp hình chúng ta sẽ chuyển ảnh hình tròn này thành ảnh xám và áp dụng các kỹ thuật dựa vào đặc trưng để phát hiện cạnh của hình tròn. Di chuyển cạnh dọc theo các cạnh và xoay với một góc cố định ta sẽ luôn thu được một đoạn cong cố định. Tương tự như vậy cho hình vuông, mặt người thì sẽ có những đặc trưng khác nhau để áp dụng kỹ thuật khác nhau cho các loại đối tượng.



Hình 1: mô hình máy học áp dụng cho bài toán nhận dạng

Bài toán này hiện nay có rất nhiều ứng dụng trong cuộc sống: Nhận dạng khuôn mặt trong lĩnh vực bảo mật, phân loại sản phẩm tự động trong sản xuất, nhận dạng xe vi phạm giao thông,... đặc biệt là ứng dụng vào công nghệ xe tự lái của rất nhiều hãng lớn đang phát triển hiện nay. Bên cạnh đó, bài toán còn được rất nhiều chuyên gia trong giới nghiên cứu và các công ty công nghệ khổng lồ tham gia phát triển. Ứng dụng cho bài toán này có rất nhiều thuật toán, công nghệ khác nhau như: LBP, HOG, Histogram,... và các thư viện và công cụ: Matlab, Tensorflow, OpenCV, Yolo...

Để minh họa cho bài toán nhận dạng đối tượng này, chúng ta chọn bộ thư viện Yolo để cài đặt và minh họa ứng dụng nhận dạng đối tượng trong ảnh tĩnh và video.

II. Giới thiệu thư viện YOLO

1. Giới thiệu thư viện

YOLO (You only look once) là một thư viện giúp phát hiện đối tượng theo thời gian thực một cách tiên tiến trong hệ thống. Trên một titan X, nó có thể xử lý hình ảnh ở mức 40 - 50 FPS, có một mAP trên VOC 2017 với 78.6% và một mAP với 48.1% trên COCO test-dev.

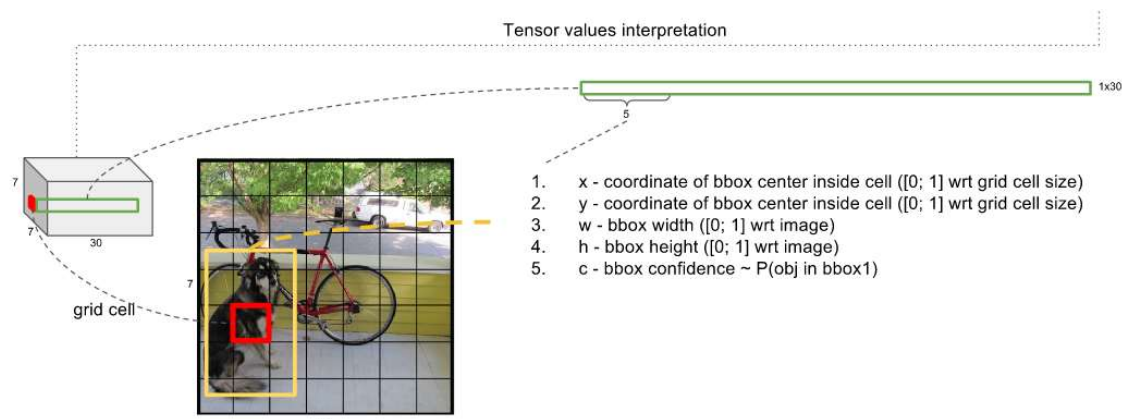
YOLO được xây dựng dựa trên bộ mã nguồn mạng neural Darknet, phát triển bằng ngôn ngữ C và CUDA (bộ thư viện xử lý đồ họa của hãng card đồ họa nổi tiếng Nvidia). Mạng neural này được đánh giá với hiệu suất chạy nhanh, dễ dàng cài đặt và hỗ trợ cả CPU, GPU. Link mã nguồn thư viện darknet neural network: <https://github.com/pjreddie/darknet>

YOLO khác gì với các thư viện/ công cụ nhận dạng đối tượng khác ? YOLO sử dụng một mạng CNN duy nhất để phân loại và định vị đối tượng sử dụng các đường bao quanh (bounding boxes).



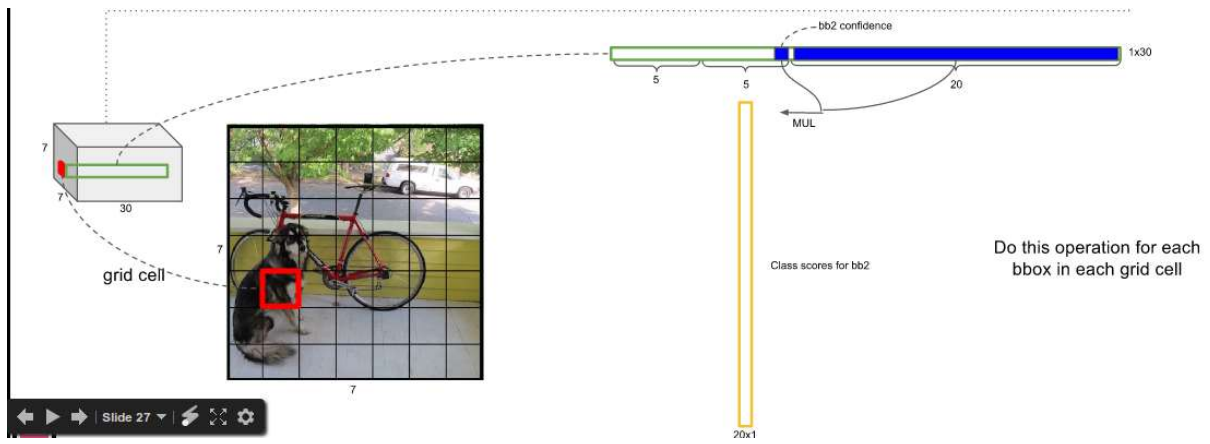
Hình 2: kiến trúc mạng của YOLO

Mỗi hộp ở cuối mô hình được chia thành lưới với kích thước $7*7*30$. Đối với mỗi đơn vị lưới, bạn sẽ nhận được hai hộp bao quanh và nó sẽ đại diện cho 10 giá trị bắt đầu của bộ đếm $1*30$. 20 còn lại đại diện cho số lớp. Các giá trị hiển thị điểm của số lớp là xác suất có điều kiện của đối tượng thuộc lớp i , nếu một đối tượng có trong hộp.



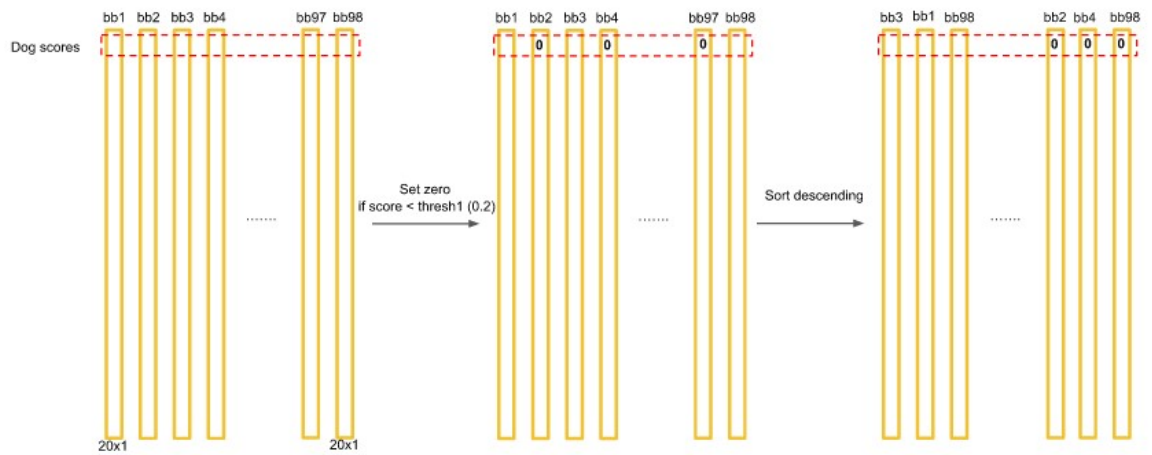
Hình 3: minh họa cách tính của thư viện YOLO

Tiếp theo, chúng ta nhân số lớp với độ confidence của bounding boxes và ta sẽ được các giá trị khác nhau cho các bounding boxes. Thực hiện cho tất cả các ô trong lưới ta được $7*7*2 = 98$ giá trị.



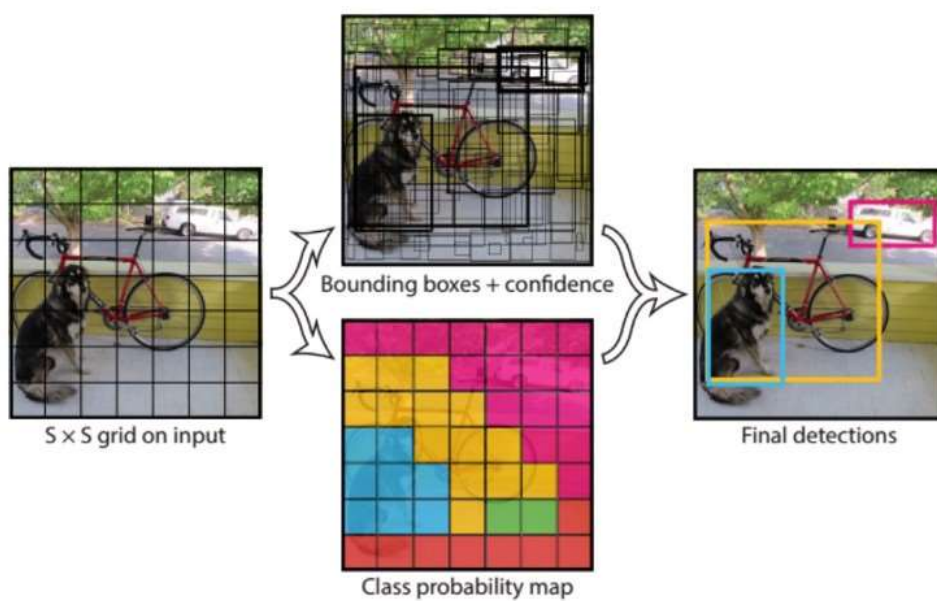
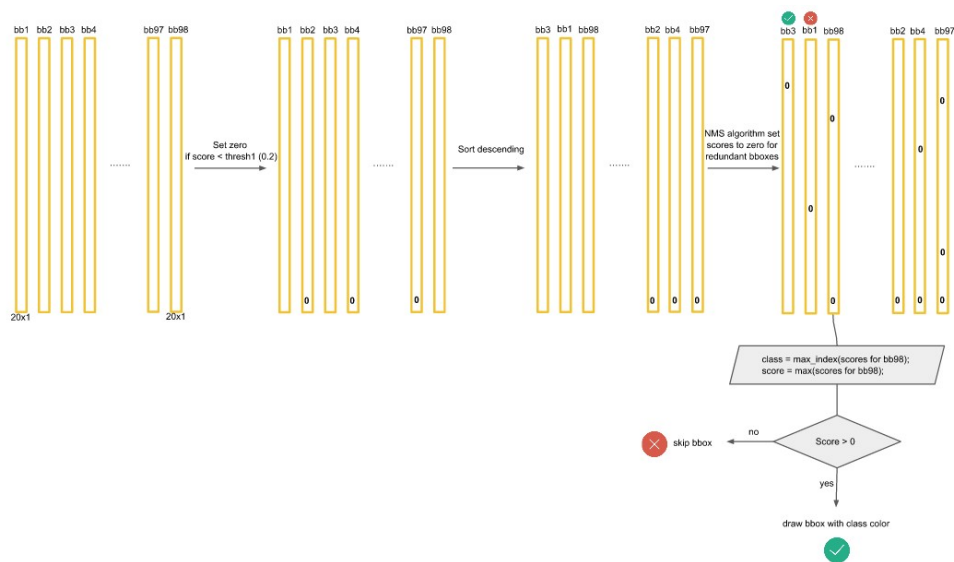
Hình 4: mô tả cách tính giá trị cho mỗi bouding boxes

Như vậy, ta đã thu được các giá trị của lớp cho mỗi bouding boxes (kích thước Tensor là 20×1). Lấy ví dụ đối tượng là con chó trong hình. Giá trị của bouding boxes con chó sẽ được biểu diễn là (1,1) của tensor trong tất cả các giá trị của bouding boxes. Chúng ta sẽ đặt một giá trị ngưỡng (threshold) cho tập giá trị và sắp xếp chúng giảm dần.



Hình 4: mô tả quá trình sắp xếp theo giá trị ngưỡng

Bây giờ chúng ta sử dụng thuật toán chặn không tối đa (Non-max suppression) để gán giá trị 0 cho các ô không cần thiết. Giả sử bạn có giá trị của con chó với boudingbox 1 là 0.5 và đặt nó là giá trị cao nhất và box47 là 0.3. Chúng ta sẽ tính toán các giá trị theo một quy tắc và nếu giá trị lớn hơn 0.5, chúng ta sẽ đặt giá trị cho box2 là 0, ngược lại, chúng ta sẽ tiếp tục cho box kế tiếp. Sau khi tính toán cho tất cả các box, chúng ta sẽ chỉ còn lại 2-3 box. Tất cả những box khác sẽ là 0. Bây giờ, chúng ta chọn bbox để vẽ giá trị cho lớp.



Hình 5: minh họa quá trình tính toán giá trị cho các box

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

Hình 6: Bảng so sánh tốc độ nhận dạng đối tượng của YOLO với một số thuật toán thông dụng

III. Cài đặt và chạy thử nghiệm cho bài toán sử dụng YOLO với tập dữ liệu được xây dựng sẵn (Pre-training Model)

1. Cài đặt thư viện darknet

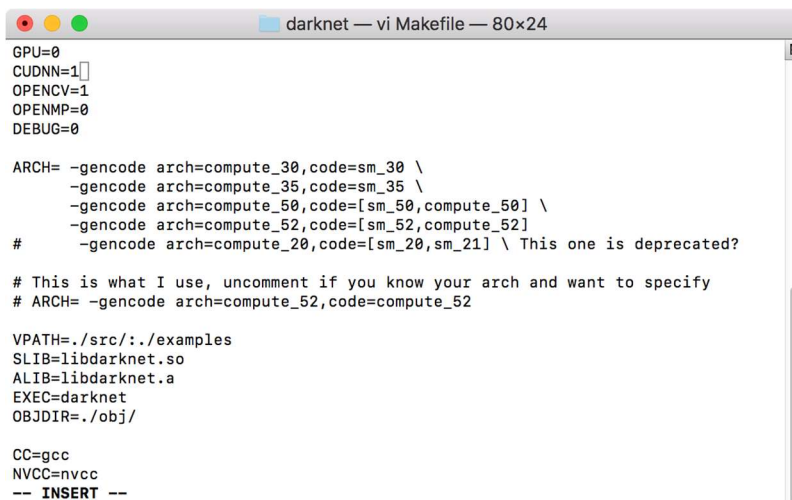
Để có thể chạy được thư viện YOLO, hệ thống của bạn cần phải cài đặt bộ thư viện Darnet (Open Source Neural Networks).

Download bộ mã nguồn darknet thông qua command (linux)
`git clone https://github.com/pjreddie/darknet.git`

Đối với darknet ta có các lựa chọn khi biên dịch như sau:

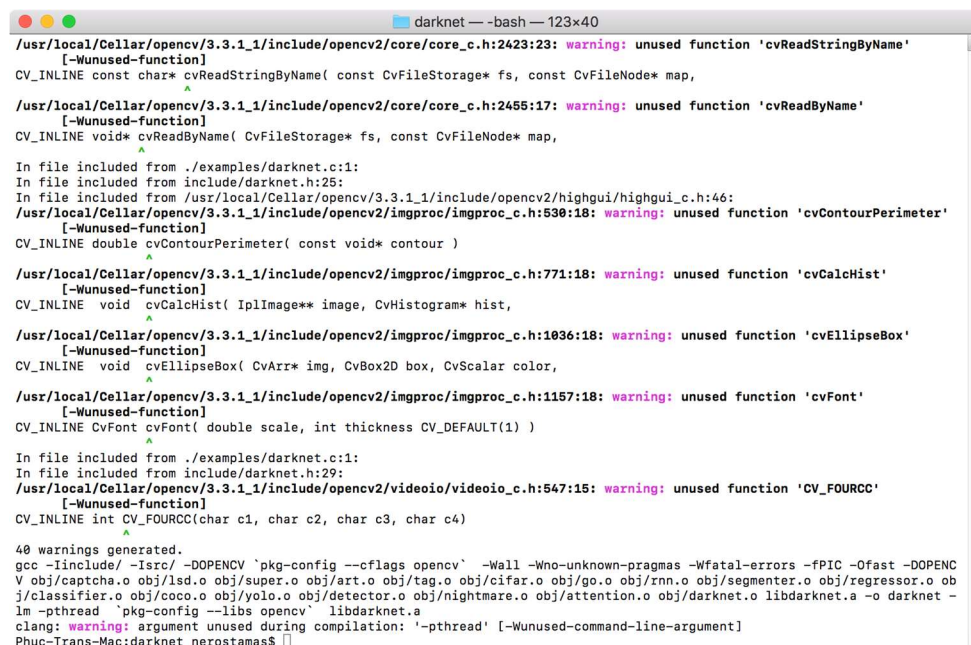
- Lựa chọn build darknet với thư viện CUDA (chạy GPU)
- Lựa chọn build darknet với thư viện OpenCV

Đối với những máy có card đồ họa Nvidia, bạn cần tải bộ thư viện CUDA Toolkit tại địa chỉ <https://developer.nvidia.com/cuda-downloads> . Sau khi download về và cài đặt, mở Makefile trong thư viện darknet lên và bật tùy chọn cho CPU 1 (trong bài báo cáo này chỉ giới hạn ở YOLO với OpenCV do máy tính không có card đồ họa).



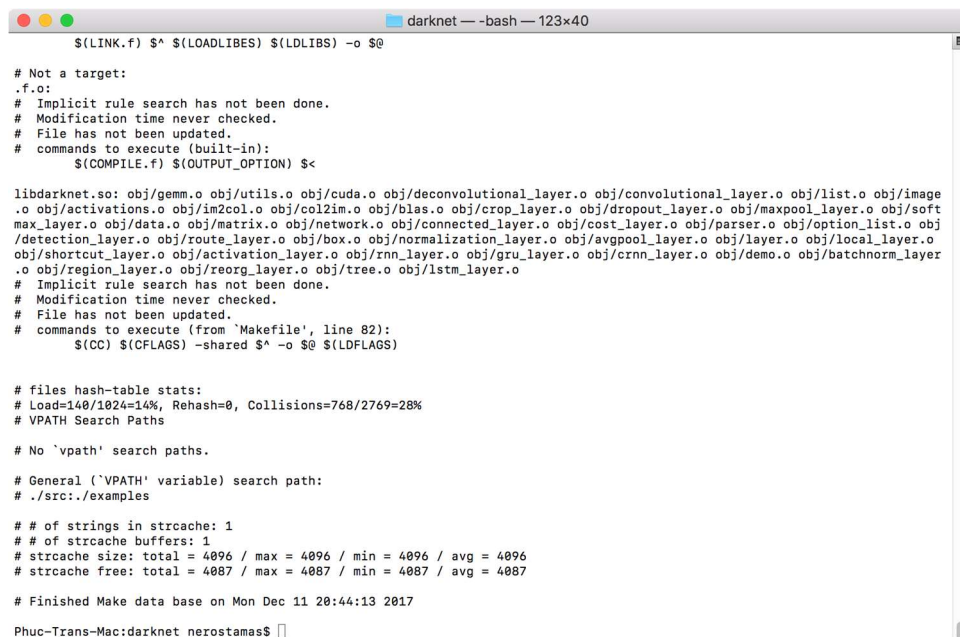
Hình 7: Hình minh họa chỉnh setting cho Makefile build darknet source

Sau khi thiết lập các tùy chỉnh cho darknet. Bạn cần chạy lệnh make để thực hiện build mã nguồn chương trình.



Hình 8: Quá trình build mã nguồn darknet với tùy chọn OpenCV

Sau khi chạy lệnh make, để kiểm tra quá trình build trong darknet gõ lệnh “make -p obj”



```
$(LINK.f) $^ $(LOADLIBES) $(LDLIBS) -o $@

# Not a target:
.f.o:
# Implicit rule search has not been done.
# Modification time never checked.
# File has not been updated.
# commands to execute (built-in):
$(COMPILE.f) $(OUTPUT_OPTION) $<

libdarknet.so: obj/gemm.o obj/utils.o obj/cuda.o obj/deconvolutional_layer.o obj/convolutional_layer.o obj/list.o obj/image.o obj/activations.o obj/im2col.o obj/col2im.o obj/blas.o obj/crop_layer.o obj/dropout_layer.o obj/maxpool_layer.o obj/softmax.o obj/data.o obj/matrix.o obj/network.o obj/connected_layer.o obj/cost_layer.o obj/parser.o obj/option_list.o obj/detection_layer.o obj/route_layer.o obj/box.o obj/normalization_layer.o obj/avgpool_layer.o obj/layer.o obj/local_layer.o obj/shortcut_layer.o obj/activation_layer.o obj/rnn_layer.o obj/gru_layer.o obj/crnn_layer.o obj/demo.o obj/batchnorm_layer.o obj/region_layer.o obj/reorg_layer.o obj/tree.o obj/lstm_layer.o
# Implicit rule search has not been done.
# Modification time never checked.
# File has not been updated.
# commands to execute (from 'Makefile', line 82):
$(CC) $(CFLAGS) -shared $^ -o $@ $(LDFLAGS)

# files hash-table stats:
# Load=140/1024=14%, Rehash=0, Collisions=768/2769=28%
# VPATH Search Paths

# No 'vpath' search paths.

# General ('VPATH' variable) search path:
# ./src:/examples

# # of strings in strcache: 1
# # of strcache buffers: 1
# strcache size: total = 4096 / max = 4096 / min = 4096 / avg = 4096
# strcache free: total = 4087 / max = 4087 / min = 4087 / avg = 4087

# Finished Make data base on Mon Dec 11 20:44:13 2017
Phuc-Trans-Mac:darknet nerostamas$
```

Hình 9: Kết quả quá trình build mã nguồn darknet

Đến bước này bạn đã hoàn thành việc build mã nguồn darknet. Để kiểm tra bạn đã build thành công hay chưa, đơn giản chỉ cần chạy ./darknet . Nếu chương trình hiển thị thông tin như bên dưới là bạn đã build thành công :

```
Phuc-Trans-Mac:darknet nerostamas$ ./darknet
usage: ./darknet <function>
```

2. Chạy ứng dụng dựa trên bộ dữ liệu YOLO được xây dựng sẵn

Để chạy được ứng dụng nhận diện đối tượng, đối với YOLO bạn có thể xây dựng lại model cho ứng dụng hoặc sử dụng model có sẵn do Darknet cung cấp. Ở phần này ta sẽ dùng model được cung cấp bởi Darknet để chạy demo một số ảnh nhận dạng ứng dụng.

Darknet cung cấp một số model đã xây dựng sẵn để có thể chạy ngay mà không cần train lại dữ liệu tại trang <https://pjreddie.com/darknet/yolo/> . Đối với mỗi model sẽ có file weight và file cfg tương ứng.

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
Old YOLO	VOC 2007+2012	2007	63.4	40.19 Bn	45		link
SSD300	VOC 2007+2012	2007	74.3	-	46		link
SSD500	VOC 2007+2012	2007	76.8	-	19		link
YOLOv2	VOC 2007+2012	2007	76.8	34.90 Bn	67	cfg	weights
YOLOv2 544x544	VOC 2007+2012	2007	78.6	59.68 Bn	40	cfg	weights
Tiny YOLO	VOC 2007+2012	2007	57.1	6.97 Bn	207	cfg	weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	-	-	7.07 Bn	200	cfg	weights

Hình 10: Các model được xây dựng sẵn và file cfg tương ứng

Mỗi bộ dữ liệu xây dựng sẵn (weights) thường có một file cfg (config) đi kèm để darknet hiểu thông số của dữ liệu đầu vào. Đối với một số bộ dữ liệu không có file cfg thì thường darknet sẽ lấy giá trị mặc định.

Trong phần này ta sẽ lấy bộ dữ liệu YOLO làm demo. Trước tiên ta cần download bộ dữ liệu YOLO thông qua command sau (khoảng 258 mb):

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

Bây giờ ta chỉ cần chạy darknet với bộ dữ liệu kèm file cấu hình tương ứng để nhận dạng đối tượng với đầu vào là một (hoặc nhiều) ảnh hoặc một video (webcam)

Command chạy dữ liệu với đầu vào là một ảnh:

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

Trong đó:

- ./darknet là file binary sau khi build chương trình darknet
- detect là phương thức gọi làm nhận diện của darknet (function)
- yolo.cfg là nội dung config của model YOLO.weights như: kích thước hình ảnh, threshold,...
- yolo.weights là tập model sau khi đã huấn luyện
- dog.jpg là ảnh đầu vào để chương trình nhận dạng



Hình 11: Một số nội dung nằm trong file yolo.cfg

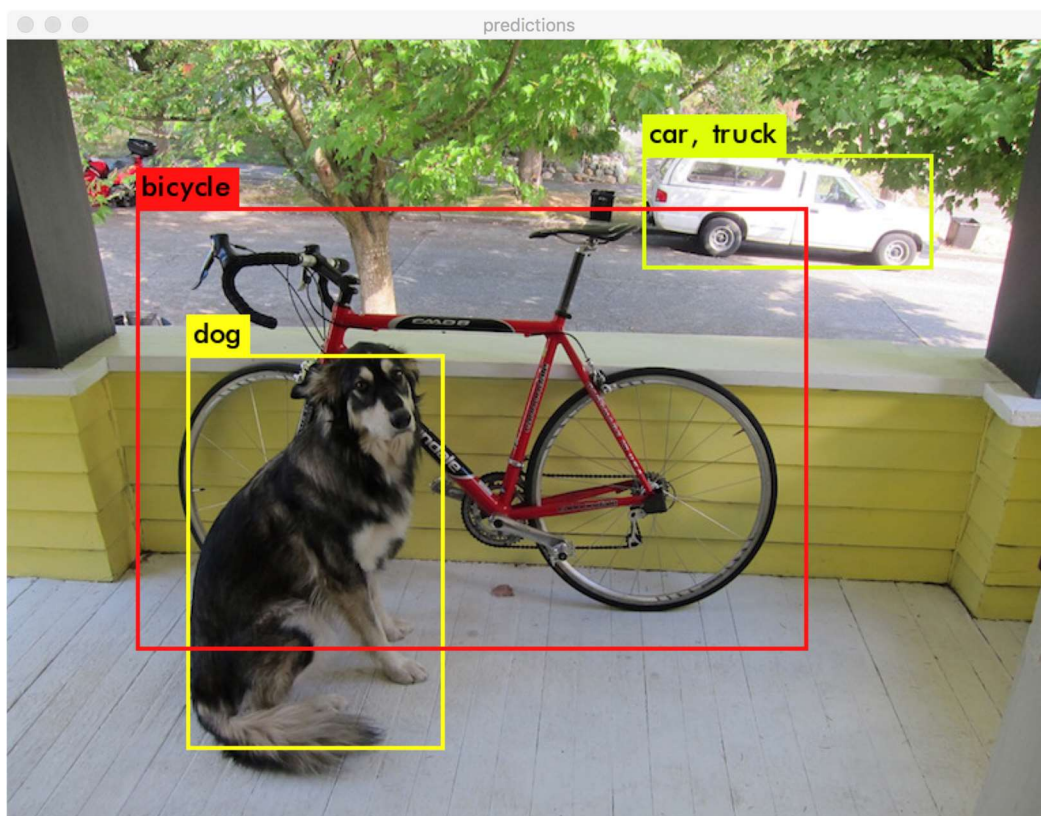
```

darknet — darknet detect cfg/yolo.cfg yolo.weights data/dog.jpg — 123x40
0 conv 32 3 x 3 / 1 608 x 608 x 3 -> 608 x 608 x 32
1 max 2 x 2 / 2 608 x 608 x 32 -> 304 x 304 x 32
2 conv 64 3 x 3 / 1 304 x 304 x 32 -> 304 x 304 x 64
3 max 2 x 2 / 2 304 x 304 x 64 -> 152 x 152 x 64
4 conv 128 3 x 3 / 1 152 x 152 x 64 -> 152 x 152 x 128
5 conv 64 1 x 1 / 1 152 x 152 x 128 -> 152 x 152 x 64
6 conv 128 3 x 3 / 1 152 x 152 x 64 -> 152 x 152 x 128
7 max 2 x 2 / 2 152 x 152 x 128 -> 76 x 76 x 128
8 conv 256 3 x 3 / 1 76 x 76 x 128 -> 76 x 76 x 256
9 conv 128 1 x 1 / 1 76 x 76 x 256 -> 76 x 76 x 128
10 conv 256 3 x 3 / 1 76 x 76 x 128 -> 76 x 76 x 256
11 max 2 x 2 / 2 76 x 76 x 256 -> 38 x 38 x 256
12 conv 512 3 x 3 / 1 38 x 38 x 256 -> 38 x 38 x 512
13 conv 256 1 x 1 / 1 38 x 38 x 512 -> 38 x 38 x 256
14 conv 512 3 x 3 / 1 38 x 38 x 256 -> 38 x 38 x 512
15 conv 256 1 x 1 / 1 38 x 38 x 512 -> 38 x 38 x 256
16 conv 512 3 x 3 / 1 38 x 38 x 256 -> 38 x 38 x 512
17 max 2 x 2 / 2 38 x 38 x 512 -> 19 x 19 x 512
18 conv 1024 3 x 3 / 1 19 x 19 x 512 -> 19 x 19 x 1024
19 conv 512 1 x 1 / 1 19 x 19 x 1024 -> 19 x 19 x 512
20 conv 1024 3 x 3 / 1 19 x 19 x 512 -> 19 x 19 x 1024
21 conv 512 1 x 1 / 1 19 x 19 x 1024 -> 19 x 19 x 512
22 conv 1024 3 x 3 / 1 19 x 19 x 512 -> 19 x 19 x 1024
23 conv 1024 3 x 3 / 1 19 x 19 x 1024 -> 19 x 19 x 1024
24 conv 1024 3 x 3 / 1 19 x 19 x 1024 -> 19 x 19 x 1024
25 route 16
26 conv 64 1 x 1 / 1 38 x 38 x 512 -> 38 x 38 x 64
27 reorg / 2 38 x 38 x 64 -> 19 x 19 x 256
28 route 27 24
29 conv 1024 3 x 3 / 1 19 x 19 x 1280 -> 19 x 19 x 1024
30 conv 425 1 x 1 / 1 19 x 19 x 1024 -> 19 x 19 x 425
31 detection
mask_scale: Using default '1.000000'
Loading weights from yolo.weights...Done!
data/dog.jpg: Predicted in 10.664598 seconds.
dog: 82%
car: 27%
truck: 65%
bicycle: 85%

```

Hình 11: Quá trình load model và kết quả nhận diện từ ảnh đầu vào.

Nếu bạn build mã nguồn darknet với option OPENCV=1 thì sau khi kết thúc quá trình nhận diện, hình ảnh nhận diện tương ứng sẽ được hiển thị (hoặc chương trình sẽ lưu thành file predictions.jpg nếu option này không được bật)



Hình 12: Kết quả nhận diện đối tượng với YOLO

Đối với trường hợp bạn muốn chạy hiệu hình mà không cần load lại model mỗi lần dự đoán thì có thể sử dụng command sau:

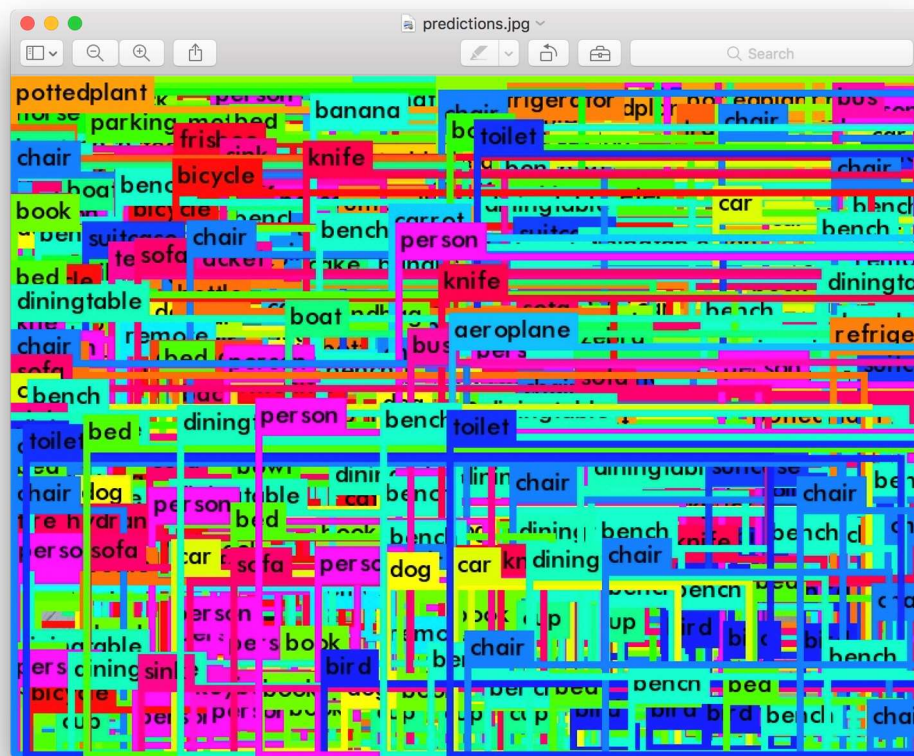
```

./darknet detect cfg/yolo.cfg yolo.weights

```

Trong darknet, mặc định sẽ đọc các thông số cài đặt từ file cfg. Trong một số trường hợp bạn muốn tùy chỉnh thông số, có thể truyền trực tiếp vào đối số khi gọi lệnh detector (một


```
./darknet detect cfg/yolo.cfg yolo.weights data/dog.jpg -thresh 0
```



Hình 13: Kết quả khi thay đổi độ chính xác threshod về 0

Đối với tham số đầu vào là video:

```
./darknet detector demo cfg/coco.data cfg/yolo.cfg yolo.weights <video
file>
```

3. Xây dựng bộ dữ liệu training dựa trên YOLO

Để xây dựng được một model mới dựa trên YOLO, bạn cần phải download tất cả các dữ liệu train VOC data của YOLO từ năm 2012 đến 2017 (tại thời điểm này) về từ trang <https://pjreddie.com/projects/pascal-voc-dataset-mirror/> và công cụ xây dựng dữ liệu training VOC devkit. Bộ dữ liệu VOC bao gồm:

```
VOctrainval_11-May-2012.tar
VOctrainval_06-Nov-2007.tar
VOctest 06-Nov-2007.tar
```

Trong đó tập VOCtest 06-Nov-2007.tar là tập dữ liệu test.

Trong darknet, để train và xây dựng được model thì darknet yêu cầu một tập .txt chứa thông tin cho mỗi hình như sau:

```
<object-class> <x> <y> <width> <height>
```

Trong đó:

- Object-class: là tên class của đối tượng (ví dụ: person, cat, dog...)
- x y là tọa độ điểm bắt đầu của box chứa đối tượng
- width và height là kích thước chiều ngang và chiều cao của box chứa đối tượng

Để đơn giản hóa quá trình xây dựng tập dữ liệu training, darkent cung cấp một script python `voc_label.py` trong thư mục `/script` để giúp ta tạo ra những label tự động đơn giản hơn đối với tập dữ liệu VOC data. Nếu chưa có bạn có thể download về tại : https://pjreddie.com/media/files/voc_label.py

Để tinh chỉnh một số thông số cho dữ liệu VOC ở trên, bạn có thể chỉnh sửa tại file `cfg/voc.data`

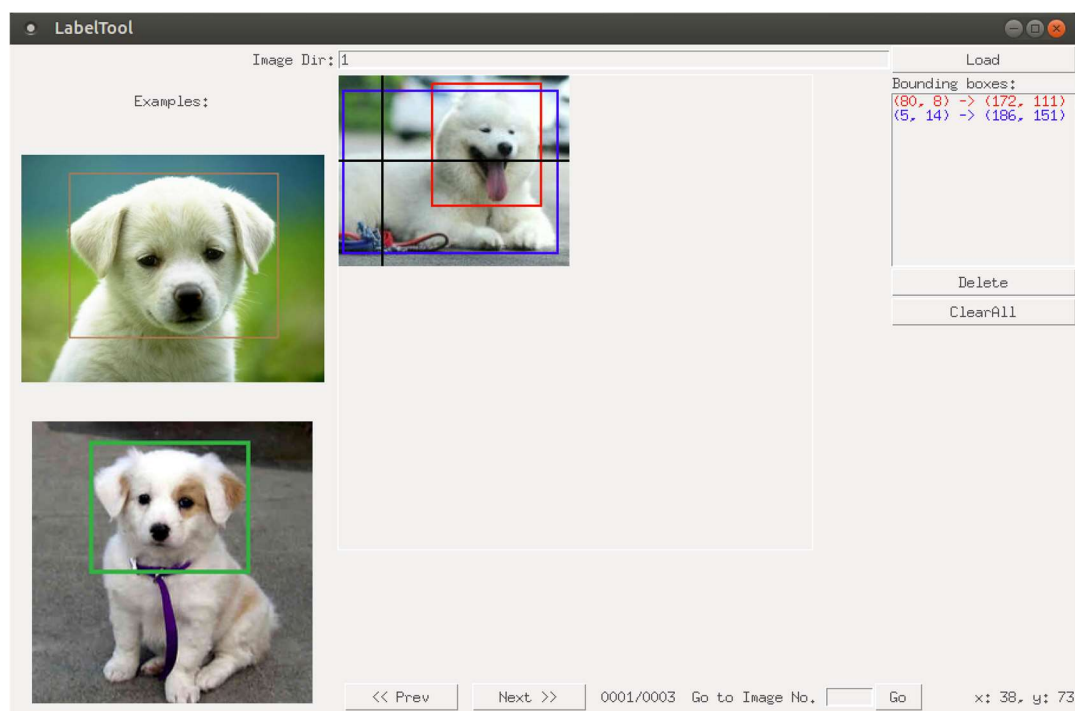
```
1 classes= 20
2 train  = <path-to-voc>/train.txt
3 valid  = <path-to-voc>2007_test.txt
4 names  = data/voc.names
5 backup = backup
```

Hình 14: Một số thông số tinh chỉnh cho VOC

Trong đó:

- Classes là số lượng class training
- Train là đường dẫn chứa label cho tập images
- Valid là đường dẫn chứa label cho tập test
- Backup là đường dẫn chứa thư mục backup cho quá trình training

Đối với dữ liệu của bạn đưa vào training thì cũng phải theo cấu trúc trên, ở phần này ta sẽ tìm một công cụ để chuẩn bị data training với YOLO. Để đơn giản hóa cho việc tạo ra các label như trên, ta sử dụng công cụ Bbox-label (<https://github.com/puzzledqs/BBox-Label-Tool>) được cộng đồng phát triển dựa trên Python Tkinter.



Hình 15: Giao diện chương trình Bbox-label

```

LabelTool
|
|--main.py # source code for the tool
|
|--Images/ # direcotry containing the images to be labeled
|
|--Labels/ # direcotry for the labeling results
|
|--Examples/ # direcotry for the example bboxes

```

Hình 16: Cấu trúc cây thư mục chứa dữ liệu để chạy tool

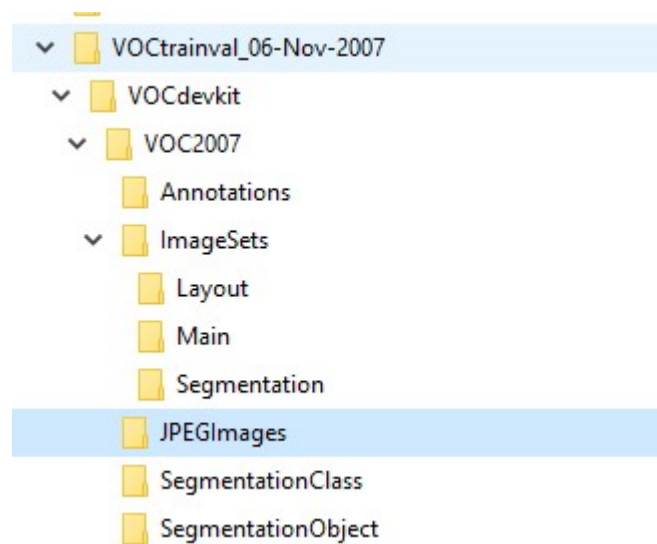
Để sử dụng được bộ công cụ này, yêu cầu máy tính của bạn phải cài Python 2.7 và Python PIL (Pillow). Sau khi chuẩn bị hình ảnh và bỏ vào thư mục Images bạn chạy command sau:

```
python main.py
```

Các bước xử lý của chương trình:

- Chương trình sẽ tìm kiếm hình ảnh và đánh dấu label theo cấu trúc image/001, image/002,... (nếu bạn muốn cấu trúc khác thì bạn có thể chỉnh sửa mã nguồn)
- Nhập vào số thứ tự của thư mục hình ảnh (label) và nhấn load. Các hình ảnh trong thư mục images, cùng với một số kết quả ví dụ sẽ được hiển thị.
- Tạo một boding box cho đối tượng bằng cách click chuột vào điểm đầu tiên của boding box, sau đó vẽ một hình chữ nhật bao đối tượng, và cuối cùng click một lần nữa để kết thúc và vẽ cho đối tượng kế tiếp (nếu có).
- Trong quá trình vẽ nhấn esc để hủy bỏ
- Để xóa bỏ một boding box đã vẽ, chọn thông tin boding box tương ứng trong listbox và nhấn Delete.
- Để xóa tất cả boding box chọn ClearAll.
- Sau khi kết thúc một hình ảnh, bấm Next để tiếp tục cho hình tiếp theo. Và ngược lại Prev để quay lại hình trước.
- Hãy chắc chắn rằng bấm Next sau khi kết thúc một hình để kết quả label được lưu xuống.

Sau khi đã chuẩn bị đầy đủ dữ liệu bạn cần đặt vào các thư mục trong tập train VOC tương ứng.



Hình 17: Cấu trúc thư mục tập dữ liệu train VOC của YOLO

Một số lưu ý:

- Thông tin label (bao gồm các bounding box) của tập dữ liệu sẽ được lưu trữ trong ImageSets/Main
- Hình ảnh của tập train sẽ được lưu trữ trong JPEGImages
- Thư mục layout sẽ chứa dữ liệu về tên các label được đánh số từ 0001

Sau chuẩn bị dữ liệu xong, bạn cần tạo ra các **Annotation** cho model bằng cách sử dụng script **voc_label.py** đã đề cập ở trên.

Tiếp theo, để training được bộ dữ liệu này trên nền tập dữ liệu YOLO, darknet yêu cầu phải chạy lệnh train với file **Pretrained Convolutional** (https://pjreddie.com/media/files/darknet19_448.conv.23) thông qua command:

```
./darknet detector train cfg/voc.data cfg/yolo-voc.cfg
darknet19_448.conv.23
```







Sau khi quá trình train kết thúc, bạn sẽ thu được một file weight tương với với model đã train ở trên. Lưu ý khi sử dụng model này để predict, bạn cần sử dụng tập dữ liệu config với thông số khi train (trường hợp này là file **cfg/yolo-voc.cfg**)

Bài Đọc Thêm: THUẬT TOÁN TRÍCH RÚT ĐẶC TRƯNG LBP

(LOCAL BINARY PATTERN)

LBP là viết tắt của Local Binary Pattern hay là mẫu nhị phân địa phương được Ojala trình bày vào năm 1996 như là một cách đo độ tương phản cục bộ của ảnh. Phiên bản đầu tiên của LBP được dùng với 8 điểm ảnh xung quanh và sử dụng giá trị của điểm ảnh ở trung tâm làm ngưỡng. Giá trị LBP được xác định bằng cách nhân các giá trị ngưỡng với trọng số ứng với mỗi điểm ảnh sau đó cộng tổng lại. Hình dưới minh họa cách tính độ tương phản trực giao (C) là hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng.

Kể từ khi được đưa ra, theo định nghĩa là bất biến với những thay đổi đơn điệu trong ảnh đen trắng. Để cải tiến phương pháp, bổ sung thêm phương pháp tương phản trực giao địa phương. Hình dưới minh họa cách tính độ tương phản trực giao (C) là ký hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng. Phân phối hai chiều của mã LBP và độ tương phản cục bộ được lấy làm đặc trưng gọi là LBP/C.

Vi dụ	Lấy ngưỡng	Trọng số																											
<table><tr><td>6</td><td>5</td><td>2</td></tr><tr><td>7</td><td>6</td><td>1</td></tr><tr><td>9</td><td>8</td><td>7</td></tr></table>	6	5	2	7	6	1	9	8	7	<table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td></td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	1		0	1	1	1	<table><tr><td>1</td><td>2</td><td>4</td></tr><tr><td>128</td><td></td><td>8</td></tr><tr><td>64</td><td>32</td><td>16</td></tr></table>	1	2	4	128		8	64	32	16
6	5	2																											
7	6	1																											
9	8	7																											
1	0	0																											
1		0																											
1	1	1																											
1	2	4																											
128		8																											
64	32	16																											
Pattern = 11110001																													
LBP = $1 + 16 + 32 + 64 + 128 = 241$																													
C = $(6+7+8+9+7)/5 - (5+2+1)/3 = 4.7$																													

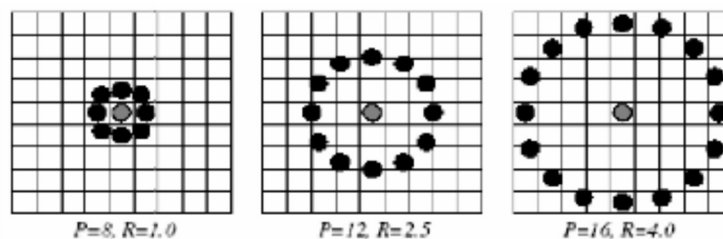
Vi dụ về LBP và độ tương phản cục bộ C.

Nguồn gốc:

Dãy LBP được Ojala trình bày vào năm 2002. Định nghĩa một cấu trúc điểm ảnh T là một phân phối đại số của cấp độ xám của $P+1$ ($P > 0$) điểm ảnh.

$$T = t(gc, g_0, \dots, g_{p-1})$$

Với gc ứng với cấp độ xám của điểm ảnh trung tâm P_{tt} , g_p ($p = 0, \dots, P$) tương ứng với P điểm ảnh xung quanh, P điểm ảnh này nằm trên đường tròn bán kính R và tâm là P_{tt} .



Tập hợp các điểm xung quanh P_{tt} .

Không mất thông tin, có thể trừ gc đi một lượng là gc

$$T = t(gc, g_0 - gc, \dots, g_{p-1} - gc)$$

Giả sử sự sai số giữa g_p và g_c là độc lập với g_c , ta có thể nhân tử hóa g_c như sau:

$$T = t(g_c)t(g_0 - g_c, \dots, g_{p-1} - g_c)$$

$t(g_c)$ biểu thị xu hướng độ sáng tối của cả bức ảnh nên không liên quan đến kết cấu của ảnh cục bộ do đó có thể bỏ qua

$$T \sim t((g_0 - g_c), \dots, (g_{p-1} - g_c))$$

Mặc dù tính bất biến ngược với độ thay đổi tỷ lệ xám của điểm ảnh, sự khác biệt ảnh hưởng bởi tỷ lệ. Để thu được đặc điểm bất biến với bất kỳ một sự thay đổi nào của ảnh đen trắng (gray scale) chỉ quan tâm đến dấu của độ lệch:

$$T \sim t(s(g_0 - g_c), \dots, s(g_{p-1} - g_c))$$

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Với s là hàm dấu:

Trọng số 2^p được dùng cho các hàm dấu, $s(g_p - g_c)$ để chuyển sự khác biệt giữa các điểm ảnh bên cạnh về một giá trị duy nhất.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) * 2^p$$

Với P pixel thì có 2^P giá trị $LBP_{P,R}$ trong khoảng $[0, 2^P - 1]$ nhưng để đơn giản ta có thể chọn một số giá trị trong 2^P giá trị ký hiệu là $LBP_{P,R}^{u2}$

Thuật toán LBP:

Thông tin LBP của pixel tại trung tâm của mỗi khối ảnh sẽ được tính dựa trên thông tin của các pixel lân cận. Có thể tóm tắt các bước tiến hành như sau:

Bước 1: Xác định bán kính làm việc.

Bước 2: Tính giá trị LBP cho pixel ở trung tâm (x_c, y_c) khối ảnh dựa trên thông tin của các pixel lân cận:

Trong đó, (g_p) là giá trị grayscale của các pixel lân cận, (g_c) là giá trị grayscale của các trung tâm và (s) là hàm nhị phân được xác định như sau: $s(z) = 1$ nếu giá trị $z \geq 0$.

Ví dụ:

47	51	65
62	70	70
80	83	78

-23	-19	-5
-8	*	0
10	13	8

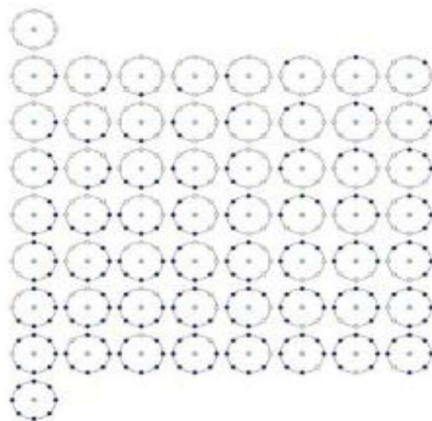
0	0	0
0	*	1
1	1	1

$$1*2^0 + 1*2^1 + 1*2^2 + 1*2^3 + 0*2^4 + 0*2^5 + 0*2^6 + 0*2^7 = 15.$$

Các biến thể của LBP LBP đồng dạng.

Một mẫu nhị phân được gọi là đồng dạng khi xét chuỗi bit xoay vòng thì có nhiều nhất là 2 lần thay đổi (transitions) từ giá trị bit 0 sang 1 hoặc từ giá trị bit 1 sang 0. Ví dụ: 00000000 có 0 transitions, 01110000 có 2 transitions, 11001111 có 2 transitions nên đây là uniform LBP. 11001001 có 4 transitions, 01010011 có 6 transitions nên không phải là uniform LBP.

Dựa trên định nghĩa này, bảng ánh xạ cho bán kính làm việc P -neighbours sẽ có $P(P-1) + 3$ nhãn. Có nghĩa là có 59 nhãn trong trường hợp làm việc với 8-neighbour. Hình vẽ sau đây thể hiện 59 nhãn (mẫu) và minh họa về histogram của đặc trưng LBP đồng dạng.



Bảng thống kê các mẫu của uniform LBP

Nguyên lý phân lớp không tham biến:

Trong phân lớp, sự khác biệt giữa mẫu và mô hình phân phối LBP được đánh giá bởi kiểm tra thống kê không tham biến. Phương pháp tiếp cận này có ưu điểm là không cần phải có những giả thiết về phân phối của các đặc trưng. Thông thường, những kiểm tra thống kê được chọn cho mục đích là nguyên lý crossentropy được giới thiệu bởi Kullback (1968). Sau đó, Sokal và Rohlf (1969) gọi cách đo này là thống kê G.

$$G(S,M) = 2 * \sum_{b=1}^B S_b \log \frac{S_b}{M_b} = 2 \sum_{b=1}^B [S_b * \log S_b - S_b * \log M_b]$$

Với S, M kí hiệu phân phối mẫu và mô hình mong muốn. S_b và M_b là xác suất để b thuộc vào phân phối mẫu hoặc mô hình. B là số phần tử trong phân phối. Thống kê G sử dụng trong phân lớp có thể viết lại như sau:

$$L(S,M) = - \sum_{b=1}^B S_b \log M_b$$

Kiến trúc mô hình có thể xem như xử lý ngẫu nhiên có đặc tính có thể xác định bởi phân phối LBP. Trong một phân lớp đơn giản, mỗi lớp được biểu diễn bởi một mô hình phân phối đơn giản M_i . Tương tự, một kiến trúc mẫu không xác định có thể miêu tả bởi phân phối S. L là một giả ma trận đo khả năng mẫu S có thể thuộc lớp i.

Lớp C của một mẫu không xác định có thể được xác định bởi luật “hàng xóm gần nhất”:
 $C = \text{argmin}_i L(S, M_i)$

Bên cạnh đó, một thống kê log-likelihood có thể xem như đơn vị đo sự khác biệt và có thể sử dụng để liên kết nhiều bộ phân lớp giống như bộ phân lớp k-NN hoặc self-organizing map (SOM). Log-likelihood đúng trong một số trường hợp nhưng không ổn định khi mà cỡ mẫu nhỏ. Trong trường hợp này Chi-square- distance thường cho kết quả tốt hơn:

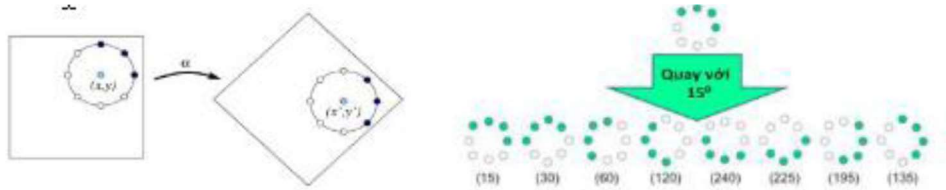
$$\chi^2(S,M) = \sum_{b=1}^B \frac{(S_b - M_b)^2}{S_b + M_b}$$

Để đạt được độ chính xác cao sử dụng giao histogram

$$H(S,M) = \sum_{b=1}^B \min[Sb, Mb]$$

Phép quay bất biến

LBP bất biến với phép quay. Giả sử $I_\alpha(x, y)$ là ảnh quay góc (α) của ảnh $I(x, y)$. Với phép quay này điểm ảnh (x, y) sẽ nằm tại vị trí (x', y') như hình vẽ sau đây (hình trái). Trong ví dụ này (hình phải): tất cả 8 mẫu LBP bên dưới được ánh xạ về mẫu LBP đầu tiên vì mẫu đầu tiên cho giá trị nhỏ nhất.



Minh họa các trường hợp của LBP khi quay với góc 15 độ

LBP đồng dạng có khả năng bất biến với phép quay Kết hợp của mẫu LBP đồng dạng và LBP bất biến với phép quay có thể tạo nên một dạng biến thể khác của LBP (uniform rotation – invariant LBP). Dựa trên định nghĩa này, bảng ánh xạ cho bán kính làm việc P-neighbors sẽ có P + 2 nhãn (label). Có nghĩa là có 10 nhãn trong trường hợp làm việc với 8-neighbour.

Để không bị ảnh hưởng bởi sự quay, mỗi giá trị LBP cần quay ngược lại về vị trí ban đầu, cách tốt nhất là tạo ra tất cả các trường hợp quay của một mẫu, sự quay có thể định nghĩa như sau:

$$LBP_{R,I}^{ri} = \min \{ \text{ROR}(LBP_{P,R}, i) \mid i=0,1,\dots,P-1 \}$$

Trong đó ri là viết tắt của rotation invariant (quay bất biến), ROR(x,i) dịch vòng tròn số nhị phân P - bit (x) i lần theo chiều kim đồng hồ.

Độ tương phản và kết cấu mẫu:

Kết cấu có thể được coi là một hiện tượng hai chiều được đặc trưng bởi hai đặc tính trực giao: cấu trúc không gian (mô hình) và độ tương phản (độ mạnh của mô hình). Quay bất biến tương phản địa phương có thể được đo trong một hình tròn đối xứng xung quanh giống như LBP:

$$\text{VAR}_{P,R} = \frac{1}{P} \sum_{p=0}^{P-1} (g_p - \mu)^2$$

Trong đó:
$$\mu = \frac{1}{P} \sum_{p=0}^{P-1} g_p$$

Tổng hợp lại ta có : $LBP_{P1,R1}^{ri} / \text{VAR}_{P2,R2}$

Ví dụ về trích rút đặc trưng LBP trên ảnh số :

Ví dụ với một ảnh có kích thước 4x4 :

23	27	33	64
35	29	15	65
6	72	11	30
1	31	3	90

Chúng ta sẽ tính được giá trị đặc trưng LBP như sau:

23	27	33	64
35	29	15	65
6	72	11	30
1	31	3	90



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	23	27	0	23	27	33	0	27	33	64	0	33	64	0	0
0	35	29	0	35	29	15	0	29	15	65	0	15	65	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	23	27	0	23	27	33	0	27	33	64	0	33	64	0	0
0	35	29	0	35	29	15	0	29	15	65	0	15	65	0	0
0	6	72	0	6	72	11	0	72	11	30	0	11	30	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	35	29	0	35	29	15	0	29	15	65	0	15	65	0	0
0	6	72	0	6	72	11	0	72	11	30	0	11	30	0	0
0	1	31	0	1	31	3	0	31	3	90	0	3	90	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	6	72	0	6	72	11	0	72	11	30	0	11	30	0	0
0	1	31	0	1	31	3	0	31	3	90	0	3	90	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	1	1	1	1	0	1	0	0	0
0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0
0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



56	104	24	32
16	164	223	2
30	0	223	34
14	2	143	0

Ví dụ về quá trình tính toán đặc trưng

Ưu điểm: Thuật toán trích rút đặc trưng LBP cài đặt đơn giản, thời gian tính toán giá trị đặc trưng nhanh vì nó làm việc với giá trị nguyên.

Nhược điểm: Tuy nhiên độ chính xác không cao bằng thuật toán Haar-like.

Ứng dụng: Được ứng dụng trong bài toán phát hiện mặt người.