

# Setting & Get Pascal VOC Data

## yolo v1 설치 후 데이터 처리

- train dataset: 2007 (train, val), 2012 (train, val)
- test dataset: 2007 (test)
- Makefile: GPU=1 -> GPU = 0으로 변경 ( 개인컴이므로 ! 서버컴에서는 GPU=1로 돌려야 함 )

```
-zsh
C4.jpg      VOCdevkit  obj
Gray.jpg    VOCtest_06-Nov-2007.tar  predictions.jpg
LICENSE     VOCtrainval_06-Nov-2007.tar  python
LICENSE.fuck VOCtrainval_11-May-2012.tar  results
LICENSE.gen  backup      scripts
LICENSE.gpl  cfg         src
LICENSE.meta darknet     voc_label.py
LICENSE.mit  data       yolov1.weights
LICENSE.v1   examples

(base) yeonju52@iyeonjuui-MacBookPro darknet % python voc_label.py
(base) yeonju52@iyeonjuui-MacBookPro darknet % ls
2007_test.txt      LICENSE.meta      examples
2007_train.txt     LICENSE.mit       include
2007_val.txt        LICENSE.v1        libdarknet.a
2012_train.txt     Makefile          libdarknet.so
2012_val.txt        Original.jpg      predictions.jpg
C1.jpg             README.md         python
C2.jpg             VOCdevkit         results
C3.jpg             VOCtest_06-Nov-2007.tar  scripts
C4.jpg             VOCtrainval_06-Nov-2007.tar  src
Gray.jpg           VOCtrainval_11-May-2012.tar  train.txt
LICENSE            backup           voc_label.py
LICENSE.fuck       cfg              yolov1.weights
LICENSE.gen        darknet
LICENSE.gpl        data
(base) yeonju52@iyeonjuui-MacBookPro darknet %

1^C^[k^A
GPU=0
CUDNN=1
OPENCV=1
OPENMP=1
DEBUG=0

ARCH= -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]
#      -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?

# This is what I use, uncomment if you know your arch and want to specify
# ARCH= -gencode arch=compute_52,code=compute_52

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

CC=gcc
NVCC=nvcc
AR=ar
ARFLAGS=rsc
```

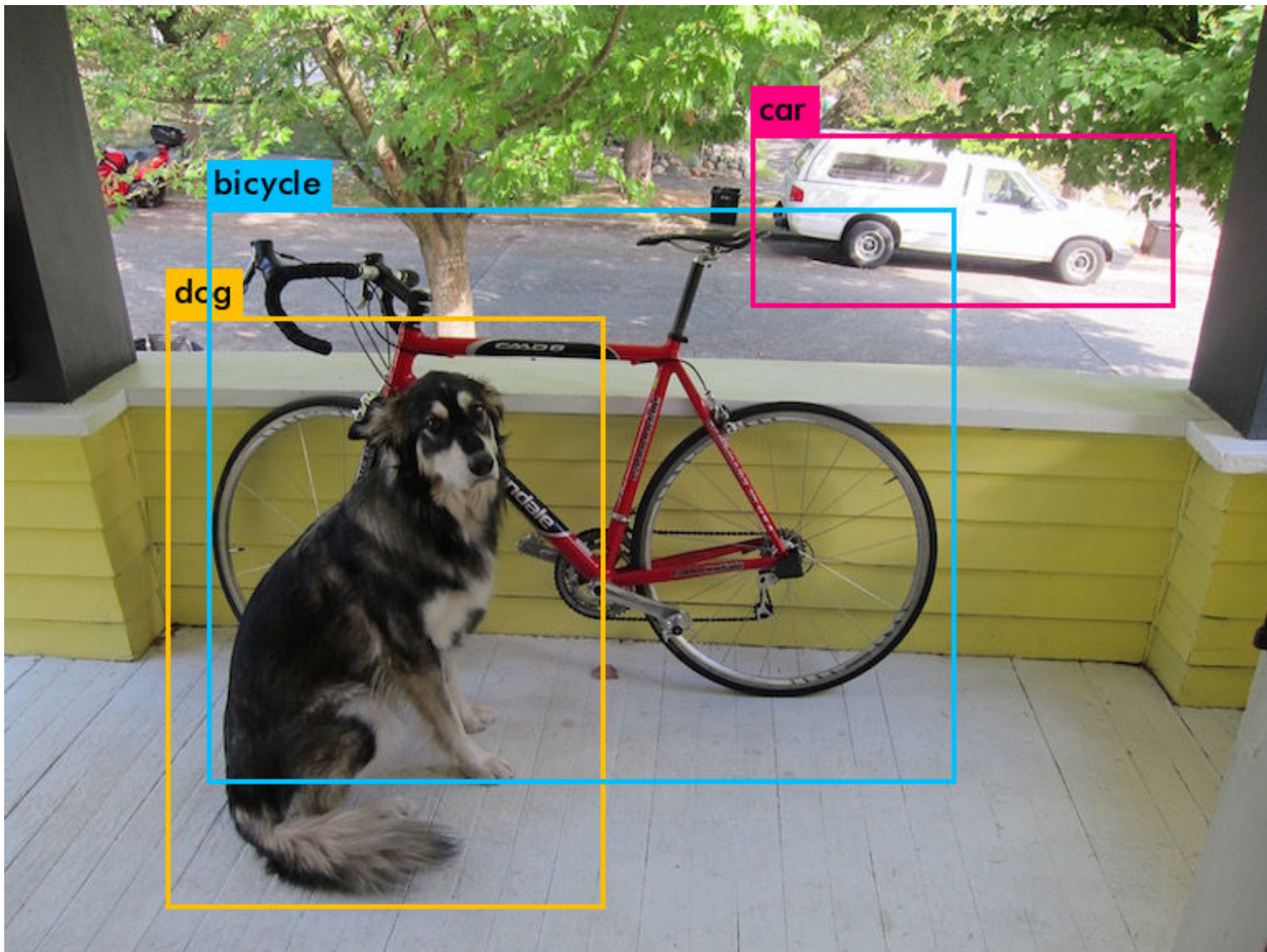
(좌) python voc\_label.py (우) Makefile



# Test Result with pre-trained weight

## yolo v1 test 실행 1

```
./darknet yolo test cfg/yolov1.cfg
yolov1.weights data/dog.jpg
```



```
(base) yeonju52@iyeonjuui-MacBookPro darknet % ./darknet yolo test cfg/yolov1/yolo.cfg yolov1.weights data/dog.jpg
Couldn't open file: cfg/yolov1/yolo.cfg
(base) yeonju52@iyeonjuui-MacBookPro darknet % ./darknet yolo test cfg/yolov1.cfg yolov1.weights data/dog.jpg
layer   filters  size      input              output
0 conv   64        7 x 7 / 2  448 x 448 x 3  -> 224 x 224 x 64  0.944 BFLOPs
1 max    2 x 2 / 2  224 x 224 x 64  -> 112 x 112 x 64
2 conv   192       3 x 3 / 1  112 x 112 x 64  -> 112 x 112 x 192 2.775 BFLOPs
3 max    2 x 2 / 2  112 x 112 x 192 -> 56 x 56 x 192
4 conv   128       1 x 1 / 1  56 x 56 x 192  -> 56 x 56 x 128  0.154 BFLOPs
5 conv   256       3 x 3 / 1  56 x 56 x 128  -> 56 x 56 x 256  1.850 BFLOPs
6 conv   256       1 x 1 / 1  56 x 56 x 256  -> 56 x 56 x 256  0.411 BFLOPs
7 conv   512       3 x 3 / 1  56 x 56 x 256  -> 56 x 56 x 512  7.399 BFLOPs
8 max    2 x 2 / 2  56 x 56 x 512  -> 28 x 28 x 512
9 conv   256       1 x 1 / 1  28 x 28 x 512  -> 28 x 28 x 256  0.206 BFLOPs
10 conv  512       3 x 3 / 1  28 x 28 x 256  -> 28 x 28 x 512  1.850 BFLOPs
11 conv  256       1 x 1 / 1  28 x 28 x 512  -> 28 x 28 x 256  0.206 BFLOPs
12 conv  512       3 x 3 / 1  28 x 28 x 256  -> 28 x 28 x 512  1.850 BFLOPs
13 conv  256       1 x 1 / 1  28 x 28 x 512  -> 28 x 28 x 256  0.206 BFLOPs
14 conv  512       3 x 3 / 1  28 x 28 x 256  -> 28 x 28 x 512  1.850 BFLOPs
15 conv  256       1 x 1 / 1  28 x 28 x 512  -> 28 x 28 x 256  0.206 BFLOPs
16 conv  512       3 x 3 / 1  28 x 28 x 256  -> 28 x 28 x 512  1.850 BFLOPs
17 conv  512       1 x 1 / 1  28 x 28 x 512  -> 28 x 28 x 512  0.411 BFLOPs
18 conv  1024      3 x 3 / 1  28 x 28 x 512  -> 28 x 28 x1024  7.399 BFLOPs
19 max    2 x 2 / 2  28 x 28 x1024  -> 14 x 14 x1024
20 conv   512       1 x 1 / 1  14 x 14 x1024  -> 14 x 14 x 512  0.206 BFLOPs
21 conv  1024      3 x 3 / 1  14 x 14 x 512  -> 14 x 14 x1024  1.850 BFLOPs
22 conv   512       1 x 1 / 1  14 x 14 x1024  -> 14 x 14 x 512  0.206 BFLOPs
23 conv  1024      3 x 3 / 1  14 x 14 x 512  -> 14 x 14 x1024  1.850 BFLOPs
24 conv  1024      3 x 3 / 1  14 x 14 x1024  -> 14 x 14 x1024  3.699 BFLOPs
25 conv  1024      3 x 3 / 2  14 x 14 x1024  -> 7 x 7 x1024  0.925 BFLOPs
26 conv  1024      3 x 3 / 1  7 x 7 x1024    -> 7 x 7 x1024  0.925 BFLOPs
27 conv  1024      3 x 3 / 1  7 x 7 x1024    -> 7 x 7 x1024  0.925 BFLOPs
28 Local Layer: 7 x 7 x 1024 image, 256 filters -> 7 x 7 x 256 image
29 dropout    p = 0.50          12544 -> 12544
30 connected          12544 -> 1715
31 Detection Layer
forced, using default '0'
Loading weights from yolov1.weights...Done!
data/dog.jpg: Predicted in 2.222997 seconds.
dog: 26%
bicycle: 39%
car: 74%
Not compiled with OpenCV, saving to predictions.png instead
```

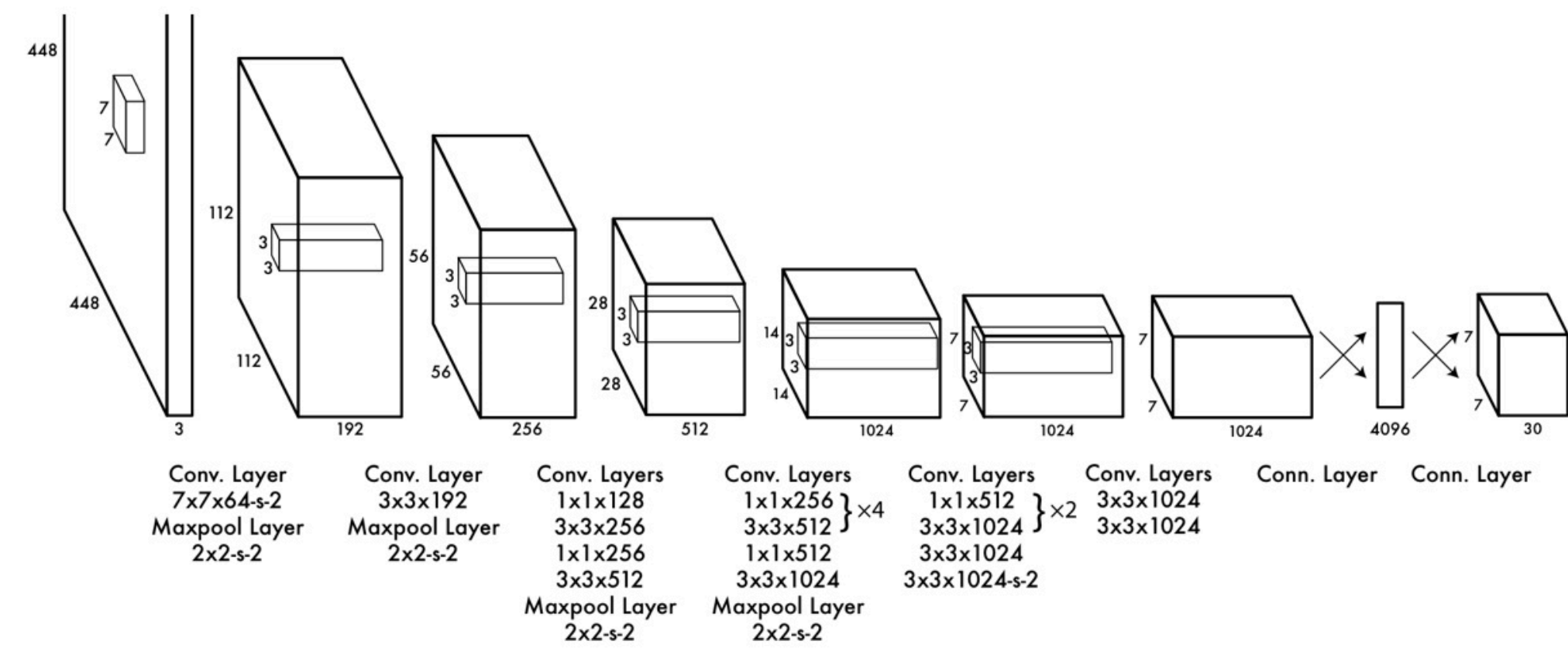


# Test Result with pre-trained weight

## 논문 내 네트워크와 코드 결과 비교

(base) yeonju52@iyeonjuui-MacBookPro darknet % ./darknet yolo test cfg/yolov1.cfg yol

layer	filters	size	input	output	
0 conv	64	7 x 7 / 2	448 x 448 x 3	-> 224 x 224 x 64	0.944 BFLOPs
1 max		2 x 2 / 2	224 x 224 x 64	-> 112 x 112 x 64	
2 conv	192	3 x 3 / 1	112 x 112 x 64	-> 112 x 112 x 192	2.775 BFLOPs
3 max		2 x 2 / 2	112 x 112 x 192	-> 56 x 56 x 192	
4 conv	128	1 x 1 / 1	56 x 56 x 192	-> 56 x 56 x 128	0.154 BFLOPs
5 conv	256	3 x 3 / 1	56 x 56 x 128	-> 56 x 56 x 256	1.850 BFLOPs
6 conv	256	1 x 1 / 1	56 x 56 x 256	-> 56 x 56 x 256	0.411 BFLOPs
7 conv	512	3 x 3 / 1	56 x 56 x 256	-> 56 x 56 x 512	7.399 BFLOPs
8 max		2 x 2 / 2	56 x 56 x 512	-> 28 x 28 x 512	
9 conv	256	1 x 1 / 1	28 x 28 x 512	-> 28 x 28 x 256	0.206 BFLOPs
10 conv	512	3 x 3 / 1	28 x 28 x 256	-> 28 x 28 x 512	1.850 BFLOPs
11 conv	256	1 x 1 / 1	28 x 28 x 512	-> 28 x 28 x 256	0.206 BFLOPs
12 conv	512	3 x 3 / 1	28 x 28 x 256	-> 28 x 28 x 512	1.850 BFLOPs
13 conv	256	1 x 1 / 1	28 x 28 x 512	-> 28 x 28 x 256	0.206 BFLOPs
14 conv	512	3 x 3 / 1	28 x 28 x 256	-> 28 x 28 x 512	1.850 BFLOPs
15 conv	256	1 x 1 / 1	28 x 28 x 512	-> 28 x 28 x 256	0.206 BFLOPs
16 conv	512	3 x 3 / 1	28 x 28 x 256	-> 28 x 28 x 512	1.850 BFLOPs
17 conv	512	1 x 1 / 1	28 x 28 x 512	-> 28 x 28 x 512	0.411 BFLOPs
18 conv	1024	3 x 3 / 1	28 x 28 x 512	-> 28 x 28 x1024	7.399 BFLOPs
19 max		2 x 2 / 2	28 x 28 x1024	-> 14 x 14 x1024	
20 conv	512	1 x 1 / 1	14 x 14 x1024	-> 14 x 14 x 512	0.206 BFLOPs
21 conv	1024	3 x 3 / 1	14 x 14 x 512	-> 14 x 14 x1024	1.850 BFLOPs
22 conv	512	1 x 1 / 1	14 x 14 x1024	-> 14 x 14 x 512	0.206 BFLOPs
23 conv	1024	3 x 3 / 1	14 x 14 x 512	-> 14 x 14 x1024	1.850 BFLOPs
24 conv	1024	3 x 3 / 1	14 x 14 x1024	-> 14 x 14 x1024	3.699 BFLOPs
25 conv	1024	3 x 3 / 2	14 x 14 x1024	-> 7 x 7 x1024	0.925 BFLOPs
26 conv	1024	3 x 3 / 1	7 x 7 x1024	-> 7 x 7 x1024	0.925 BFLOPs
27 conv	1024	3 x 3 / 1	7 x 7 x1024	-> 7 x 7 x1024	0.925 BFLOPs
28 Local Layer:	7 x 7 x 1024 image, 256 filters -> 7 x 7 x 256 image				
29 dropout	p = 0.50		12544	->	12544
30 connected			12544	->	1715
31 Detection Layer					

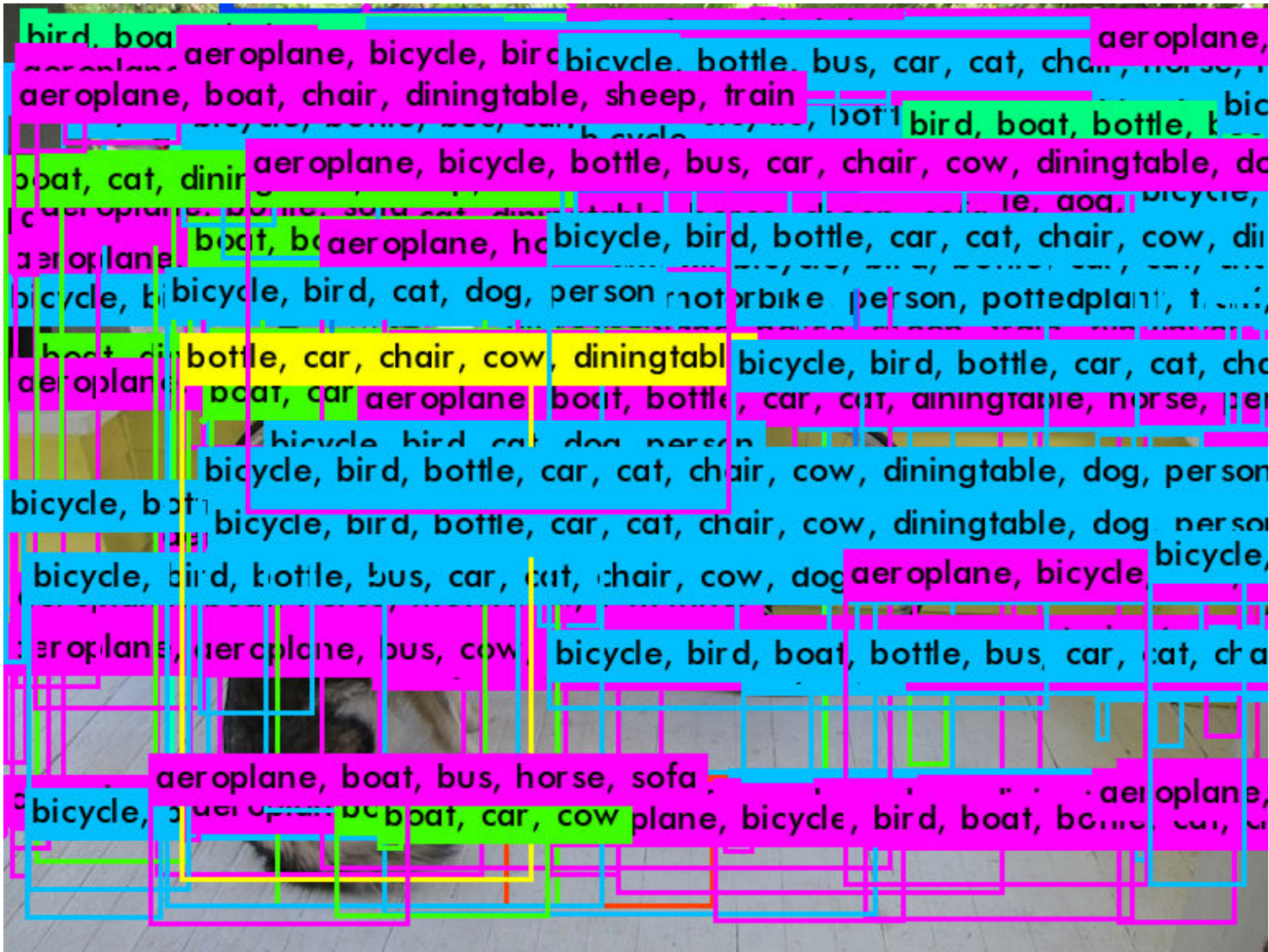




# Test Result (thresh = 0)

## yolo v1 test 실행 2

```
./darknet yolo test cfg/yolov1.cfg
yolov1.weights data/dog.jpg -thresh 0 |
awk 'NR < 7'
```



```
(base) yeoniu52@iveoniuui-MacBookPro darknet % ./darknet yolo test cfg/yolov1.cfg yolov1.weights data/dog.jpg -thresh 0 | awk 'NR < 7'
```

layer	filters	size	input	output	
0 conv	64	7 x 7 / 2	448 x 448 x 3	224 x 224 x 64	0.944 BFLOPs
1 max		2 x 2 / 2	224 x 224 x 64	112 x 112 x 64	
2 conv	192	3 x 3 / 1	112 x 112 x 64	112 x 112 x 192	2.775 BFLOPs
3 max		2 x 2 / 2	112 x 112 x 192	56 x 56 x 192	
4 conv	128	1 x 1 / 1	56 x 56 x 192	56 x 56 x 128	0.154 BFLOPs
5 conv	256	3 x 3 / 1	56 x 56 x 128	56 x 56 x 256	1.850 BFLOPs
6 conv	256	1 x 1 / 1	56 x 56 x 256	56 x 56 x 256	0.411 BFLOPs
7 conv	512	3 x 3 / 1	56 x 56 x 256	56 x 56 x 512	7.399 BFLOPs
8 max		2 x 2 / 2	56 x 56 x 512	28 x 28 x 512	
9 conv	256	1 x 1 / 1	28 x 28 x 512	28 x 28 x 256	0.206 BFLOPs
10 conv	512	3 x 3 / 1	28 x 28 x 256	28 x 28 x 512	1.850 BFLOPs
11 conv	256	1 x 1 / 1	28 x 28 x 512	28 x 28 x 256	0.206 BFLOPs
12 conv	512	3 x 3 / 1	28 x 28 x 256	28 x 28 x 512	1.850 BFLOPs
13 conv	256	1 x 1 / 1	28 x 28 x 512	28 x 28 x 256	0.206 BFLOPs
14 conv	512	3 x 3 / 1	28 x 28 x 256	28 x 28 x 512	1.850 BFLOPs
15 conv	256	1 x 1 / 1	28 x 28 x 512	28 x 28 x 256	0.206 BFLOPs
16 conv	512	3 x 3 / 1	28 x 28 x 256	28 x 28 x 512	1.850 BFLOPs
17 conv	512	1 x 1 / 1	28 x 28 x 512	28 x 28 x 512	0.411 BFLOPs
18 conv	1024	3 x 3 / 1	28 x 28 x 512	28 x 28 x 1024	7.399 BFLOPs
19 max		2 x 2 / 2	28 x 28 x 1024	14 x 14 x 1024	
20 conv	512	1 x 1 / 1	14 x 14 x 1024	14 x 14 x 512	0.206 BFLOPs
21 conv	1024	3 x 3 / 1	14 x 14 x 512	14 x 14 x 1024	1.850 BFLOPs
22 conv	512	1 x 1 / 1	14 x 14 x 1024	14 x 14 x 512	0.206 BFLOPs
23 conv	1024	3 x 3 / 1	14 x 14 x 512	14 x 14 x 1024	1.850 BFLOPs
24 conv	1024	3 x 3 / 1	14 x 14 x 1024	14 x 14 x 1024	3.699 BFLOPs
25 conv	1024	3 x 3 / 2	14 x 14 x 1024	7 x 7 x 1024	0.925 BFLOPs
26 conv	1024	3 x 3 / 1	7 x 7 x 1024	7 x 7 x 1024	0.925 BFLOPs
27 conv	1024	3 x 3 / 1	7 x 7 x 1024	7 x 7 x 1024	0.925 BFLOPs
28 Local Layer:			7 x 7 x 1024 image, 256 filters	-> 7 x 7 x 256 image	
29 dropout		p = 0.50	12544	-> 12544	
30 connected			12544	-> 1715	
31 Detection Layer					

```
forced: Using default '0'
```

```
Loading weights from yolov1.weights...Done!
```

```
Not compiled with OpenCV, saving to predictions.png instead
```

```
data/dog.jpg: Predicted in 2.223863 seconds.
```

```
aeroplane: 1%
```

```
bicycle: 11%
```

```
car: 74%
```

```
cat: 1%
```

```
cow: 2%
```



# Train 실행 시도 -> 실패★

- pre-trained model 말고 직접 test를 시도

```
./darknet yolo train cfg/yolov1/yolo.train.cfg extraction.conv.weights
```

- 실패: extraction.conv.weights 찾을 수 없음 > 버전이 업데이트 되면서 주소가 달라지거나 아예 삭제된 것으로 추정
- yolo v3 이나 yolo v5를 재설치 시도 중
  - yolo v3 설치 시도  $\Rightarrow$  make 명령어 오류

```
clang: fatal error: unsupported option '-fopenmp'
```

## Anchor Box

- 종횡비(w, h)가 정해진 bounding box
- b.w, b.h: 사전에 크기와 비율이 모두 결정되어 있는 박스임을 알 수 있음 (Anchor Box)
  - 이 박스를 전제로, 학습을 통해서 이 박스의 위치나 크기를 세부 조정함
- Anchor Box는 object detection에 큰 영향을 끼침 -> 다음주 계산식 해석

```
83  box get_yolo_box(float *x, float *biases, int n, int index, int i, int j, int lw, int lh, int w, int h, int stride)
84  {
85      box b;
86      b.x = (i + x[index + 0*stride]) / lw;
87      b.y = (j + x[index + 1*stride]) / lh;
88      b.w = exp(x[index + 2*stride]) * biases[2*n] / w;
89      b.h = exp(x[index + 3*stride]) * biases[2*n+1] / h;
90      return b;
91  }
```

