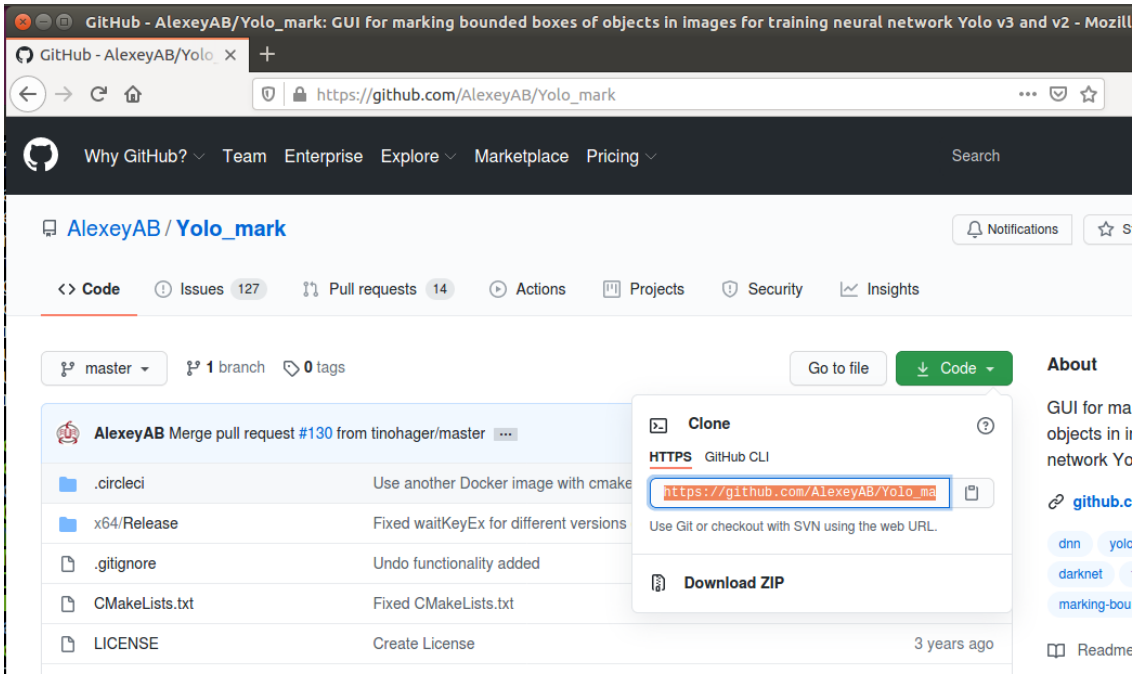
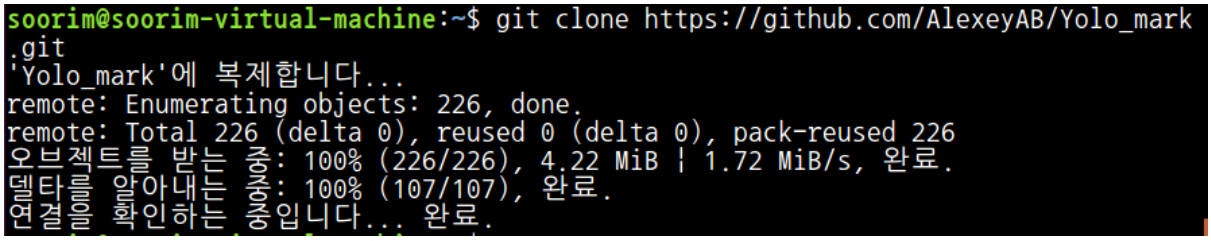
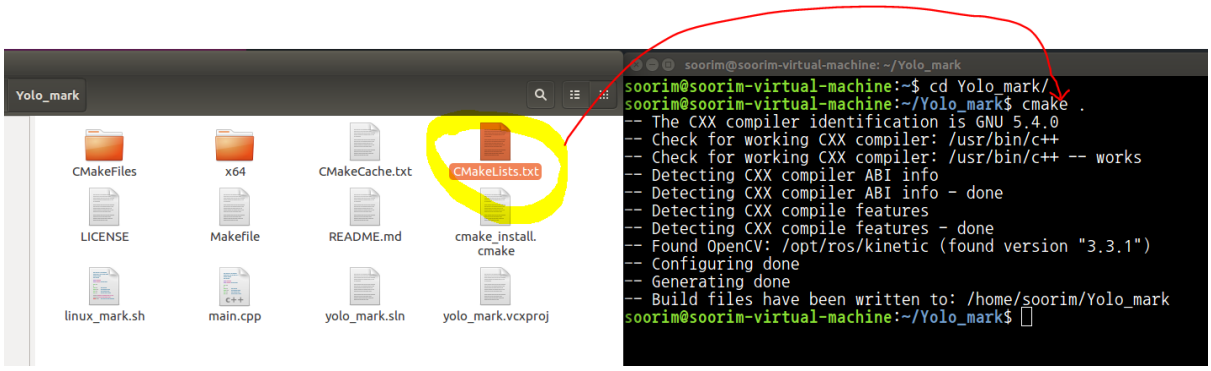
- **YOLO 학습**

https://github.com/AlexeyAB/Yolo_mark



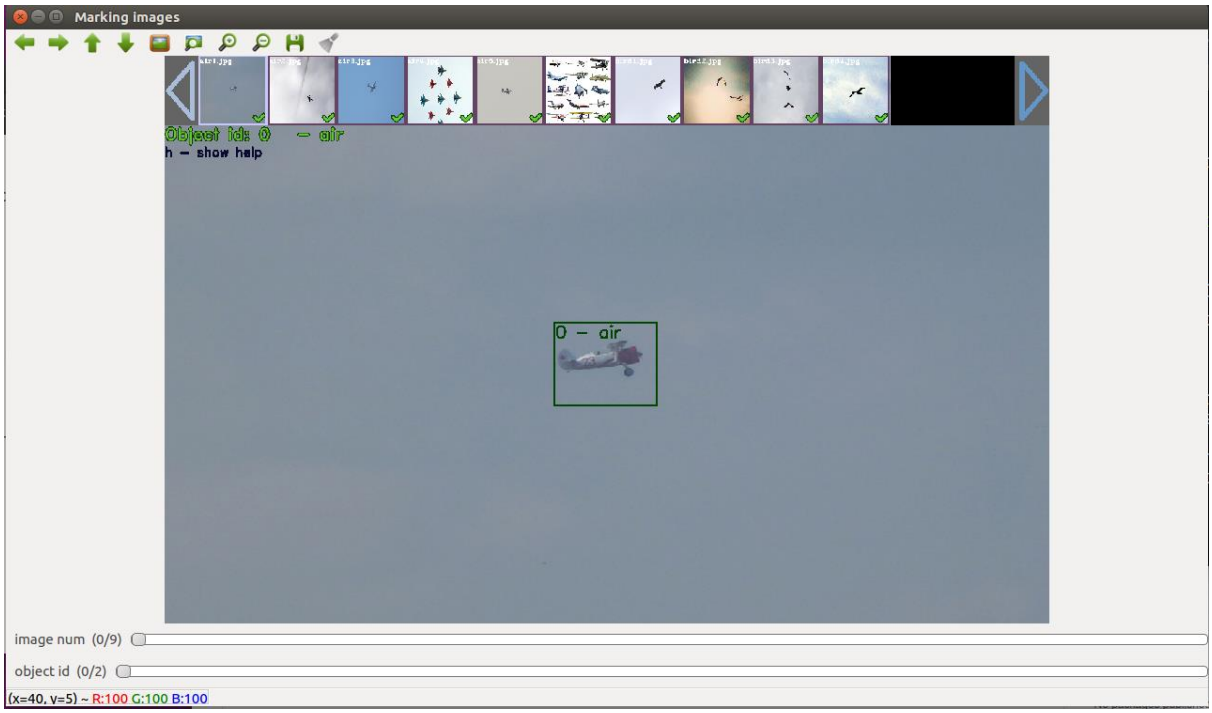




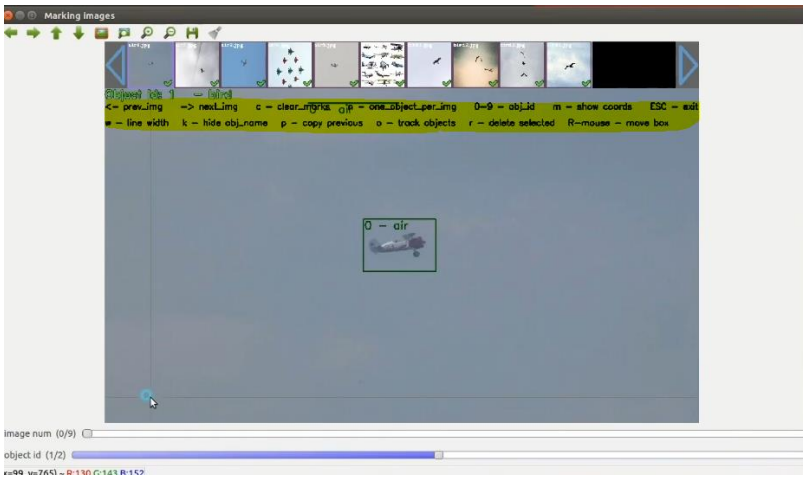
Yolo_make 폴더에 CMake파일이 있으면 cmake . 실행

```
soorim@soorim-virtual-machine:~/Yolo_mark$ make
Scanning dependencies of target yolo_mark
[ 50%] Building CXX object CMakeFiles/yolo_mark.dir/main.cpp.o
[100%] Linking CXX executable yolo_mark
[100%] Built target yolo_mark
soorim@soorim-virtual-machine:~/Yolo_mark$ chmod +x ./linux_mark.sh
soorim@soorim-virtual-machine:~/Yolo_mark$ ./linux_mark.sh
```

- **실행 결과**

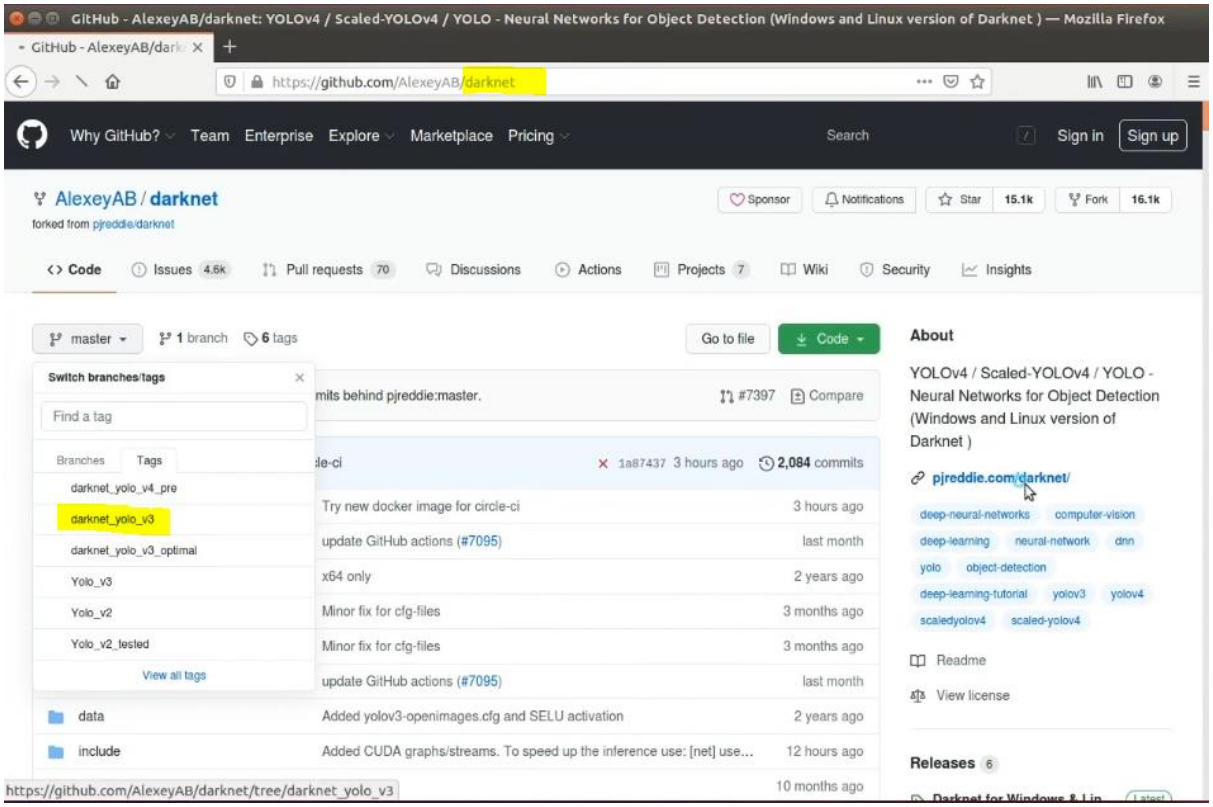


H를 누르면 아래와 같이 됨

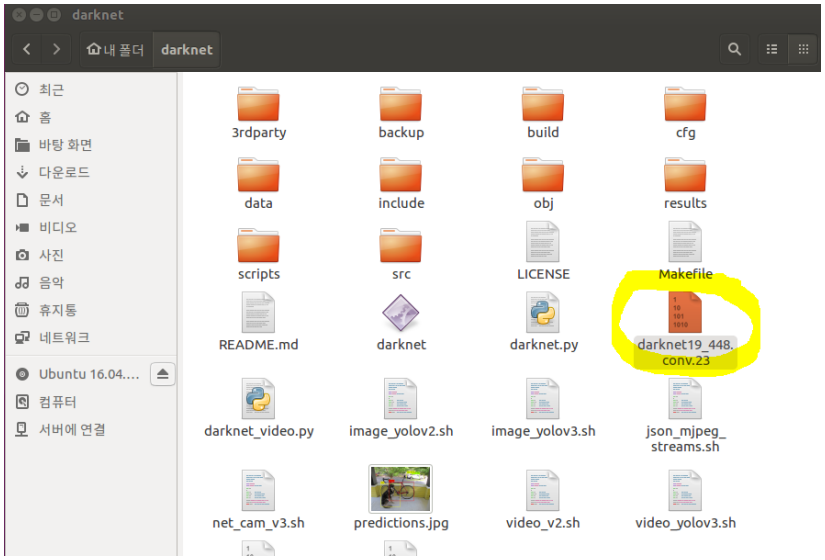


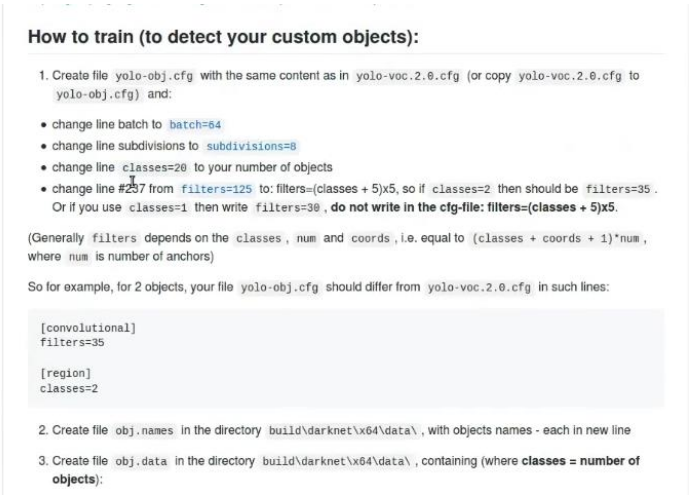
- **학습 모델 고르기**

사전 학습 파일 다운 받기



다운 받은 파일 darknet폴더로 옮기기





How to train (to detect your custom objects):

1. Create file `yolo-obj.cfg` with the same content as in `yolo-voc.2.0.cfg` (or copy `yolo-voc.2.0.cfg` to `yolo-obj.cfg`) and:

- change line batch to `batch=64`
- change line subdivisions to `subdivisions=8`
- change line `classes=20` to your number of objects
- change line #297 from `filters=125` to: filters=(classes + 5)x5, so if `classes=2` then should be `filters=35`. Or if you use `classes=1` then write `filters=30`, **do not write in the cfg-file: filters=(classes + 5)x5**.

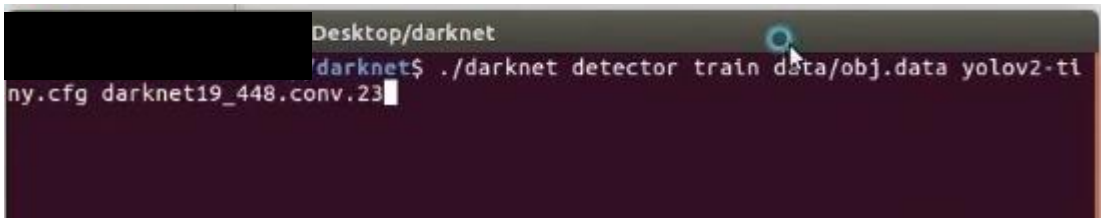
(Generally `filters` depends on the `classes`, `num` and `coords`, i.e. equal to `(classes + coords + 1)*num`, where `num` is number of anchors)

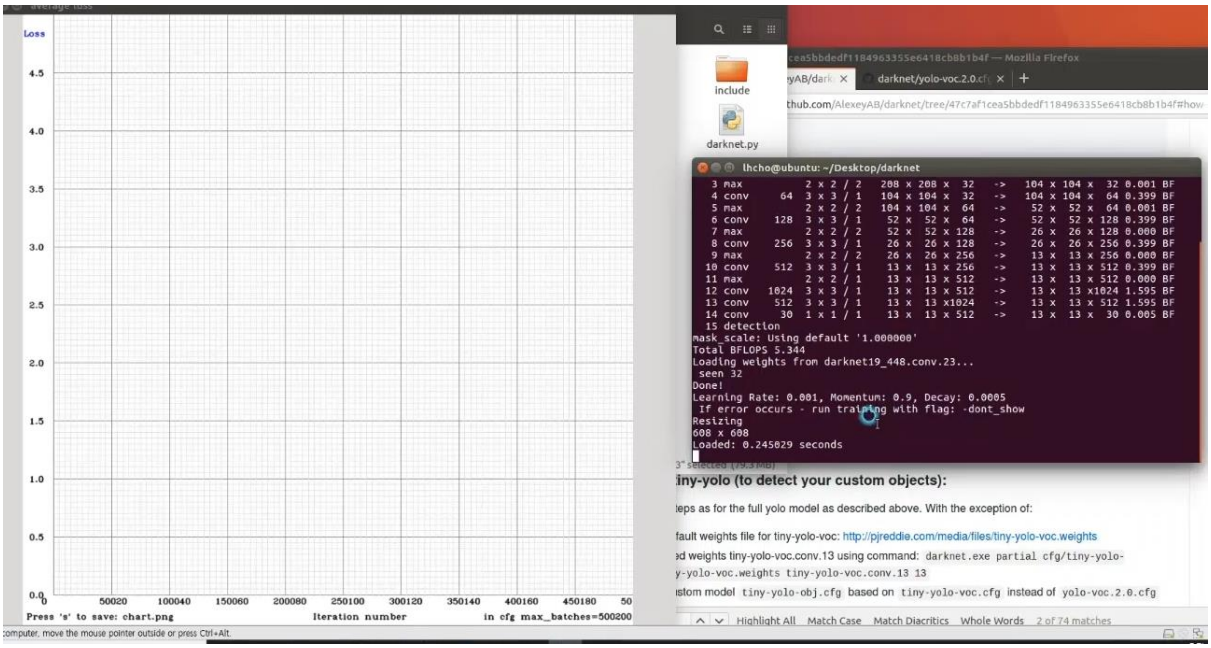
So for example, for 2 objects, your file `yolo-obj.cfg` should differ from `yolo-voc.2.0.cfg` in such lines:

```
[convolutional]
filters=35

[region]
classes=2
```

2. Create file `obj.names` in the directory `build\darknet\x64\data\`, with objects names - each in new line
3. Create file `obj.data` in the directory `build\darknet\x64\data\`, containing (where **classes = number of objects**):




Loss
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50020
100040
150060
200080
250100
300120
350140
400160
450180
Press 's' to save: chart.png
Iteration number
in cfg max_batches=500200
include
darknet.py
lhcho@ubuntu: ~/Desktop/darknet
3 max 2 x 2 / 2 208 x 208 x 32 -> 104 x 104 x 32 0.001 BF
4 conv 64 3 x 3 / 1 104 x 104 x 32 -> 104 x 104 x 64 0.399 BF
5 max 2 x 2 / 2 104 x 104 x 64 -> 52 x 52 x 64 0.001 BF
6 conv 128 3 x 3 / 1 52 x 52 x 64 -> 52 x 52 x 128 0.399 BF
7 max 2 x 2 / 2 52 x 52 x 128 -> 26 x 26 x 128 0.000 BF
8 conv 256 3 x 3 / 1 26 x 26 x 128 -> 26 x 26 x 256 0.399 BF
9 max 2 x 2 / 2 26 x 26 x 256 -> 13 x 13 x 256 0.000 BF
10 conv 512 3 x 3 / 1 13 x 13 x 256 -> 13 x 13 x 512 0.399 BF
11 max 2 x 2 / 1 13 x 13 x 512 -> 13 x 13 x 512 0.000 BF
12 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x1024 1.595 BF
13 conv 512 3 x 3 / 1 13 x 13 x1024 -> 13 x 13 x 512 1.595 BF
14 conv 30 1 x 1 / 1 13 x 13 x 512 -> 13 x 13 x 30 0.005 BF
15 detection
mask_scale: Using default '1.000000'
Total BFLOPS 5.344
Loading weights from darknet19_448.conv.23...
seen 32
Done!
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
If error occurs - run training with flag: -dont_show
Resizing
608 x 608
Loaded: 0.245829 seconds
iny-yolo (to detect your custom objects):
eps as for the full yolo model as described above. With the exception of:
fault weights file for tiny-yolo-voc: http://pjreddie.com/media/files/tiny-yolo-voc.weights
d weights tiny-yolo-voc.conv.13 using command: darknet.exe partial cfg/tiny-yolo-
y-yolo-voc.weights tiny-yolo-voc.conv.13 13
stom model tiny-yolo-obj.cfg based on tiny-yolo-voc.cfg instead of yolo-voc.2.0.cfg
Highlight All Match Case Match Diacritics Whole Words 2 of 74 matches
computer, move the mouse pointer outside or press Ctrl+Alt.