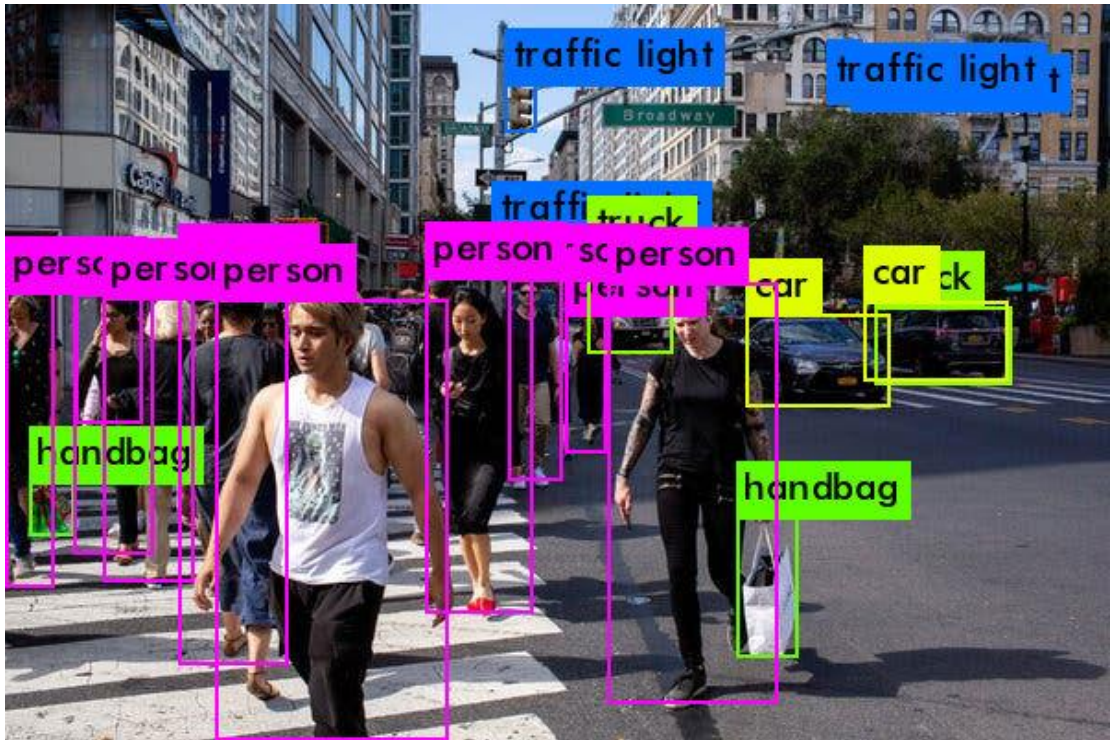
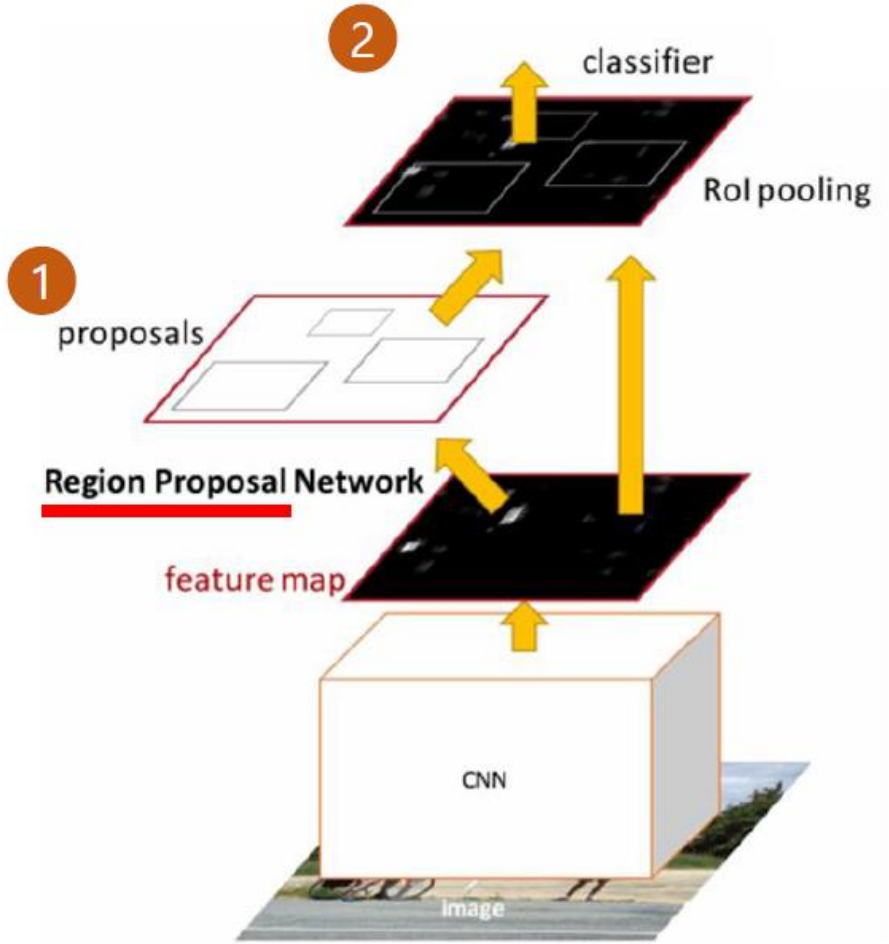
YOLO: Real-time Object detection



2014460 KIM JEONG HYUN

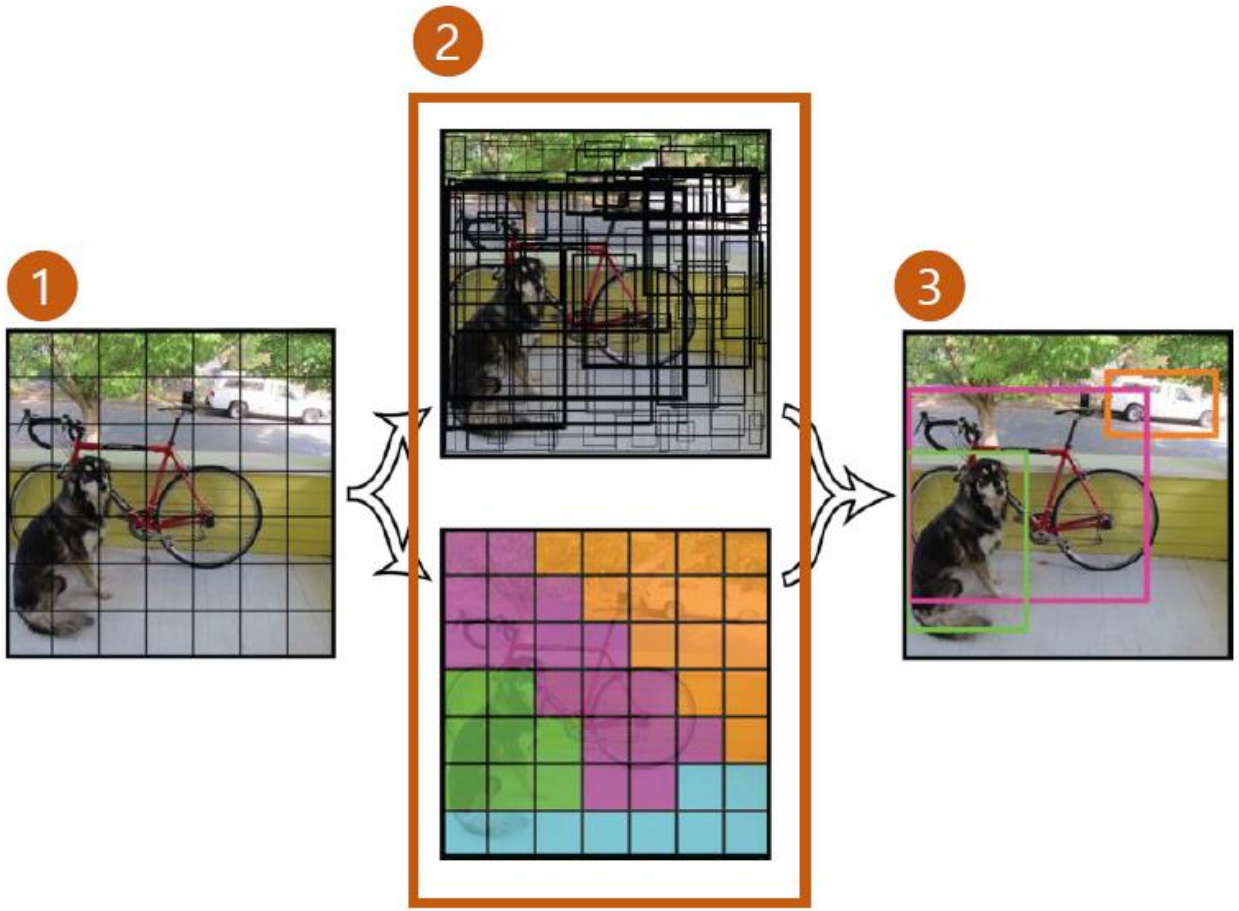


Fast R-CNN

오브젝트가 있을 것 같은 영역을 뽑아서 제안

객체 검출의 정확성은 높지만 신속하지 못함

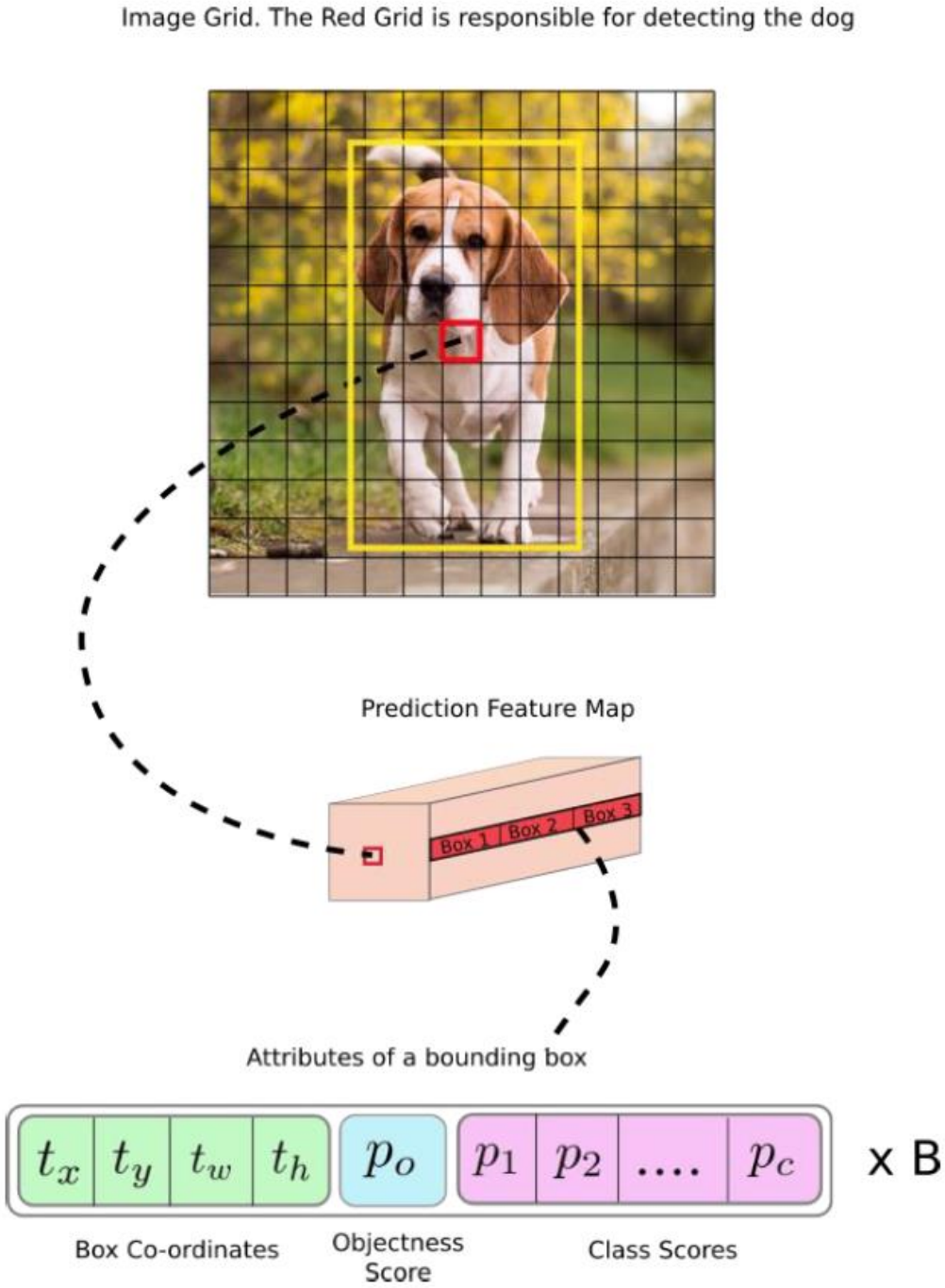
1 2 = **Two-stage Methods**



YOLO (You Only Look Once)

1. 프레임을 7*7 그리드로 분할
2. 각 그리드를 중심으로 Bound box 생성
   1) Bound box를 그리드 셀(7*7)의 2배만큼 생성
      → 프레임 상의 오브젝트 위치 예측
   2) 제안된 Bound box의 오브젝트 class 구분
3. 객체별 class와 영역 검출

= Single-stage Methods



Interpreting the output(=featuremap)

Input Image = 416 * 416 size
Stride = 32

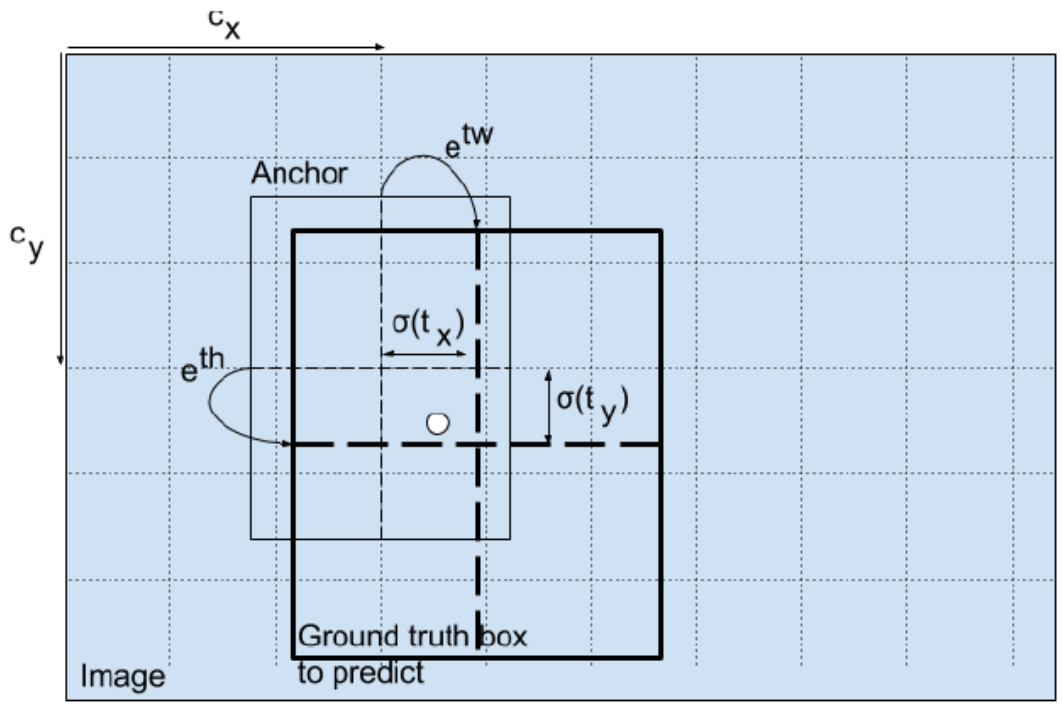
→ cell = 13*13

임의의 cell 하나 당 3개의 bounding box(=B)

B 하나 당 5 + C개의 attr.
(tx, ty = B 의 x, y좌표
tw, th = B 의 크기
Po = 정확도(objectness score)
C = Px = cat, dog, car, ...)

→ 총 B * (5 * C) 개의 feature map

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$



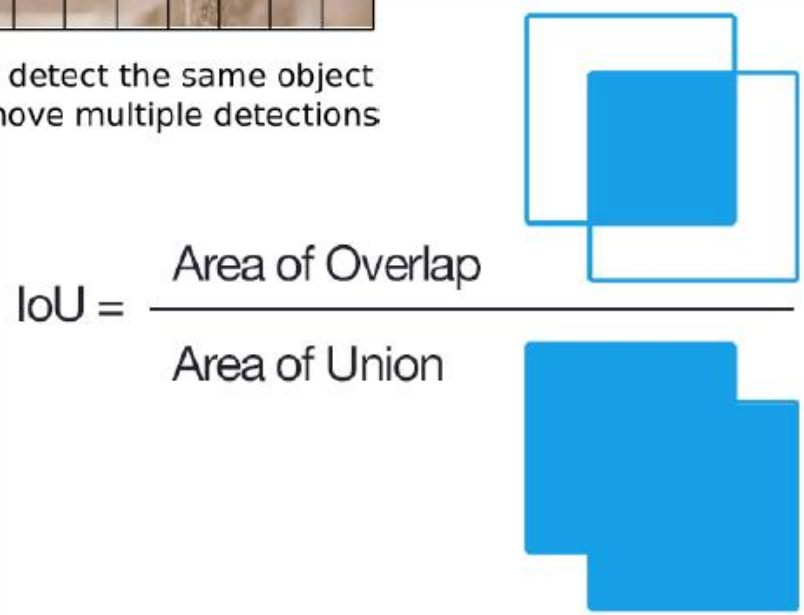
Anchor map

Anchor?: simply offsets to pre-defined default bounding boxes

x, y 좌표를 전체로 확대
: cx, cy = cell의 x, y좌표
→ cx(cy) + tx(ty) = 전체 image에서 B의 좌표

이 때 전체 image의 크기를 벗어나지 않기 위해
tx, ty에 Sigmoid 함수를 사용 → 0~1

Multiple Grids may detect the same object
NMS is used to remove multiple detections



Output Processing

유사한 위치/크기의 feature map이 대량 발생

NMS를 활용
→ IoU를 이용해 겹치는 비율로 유사도 측정
→ 겹치는 Box들의 합집합으로 feature map 확정

YOLO: Real-time Object detection ... / Google Colab

Configuration

```
# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
```

Data(classes)

```
person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
```

Weights

다양한 가중치
파일 제공

YOLO: Real-time Object detection … / Google Colab

Darknet.py
: architecture

- Parse_cfg
- Create_modules
- Forward
- Load_weights

Util.py
: helper functions

- Predict_transform
- Bbox_iou
- Write_results
- Letterbox_image

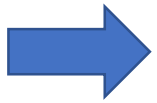
Detector.py
: executor

실행 코드

YOLO: Real-time Object detection … / Google Colab

Colab 에서는 구현 실패

1) 파일 간 연동 어려움
2) Jupitor ???

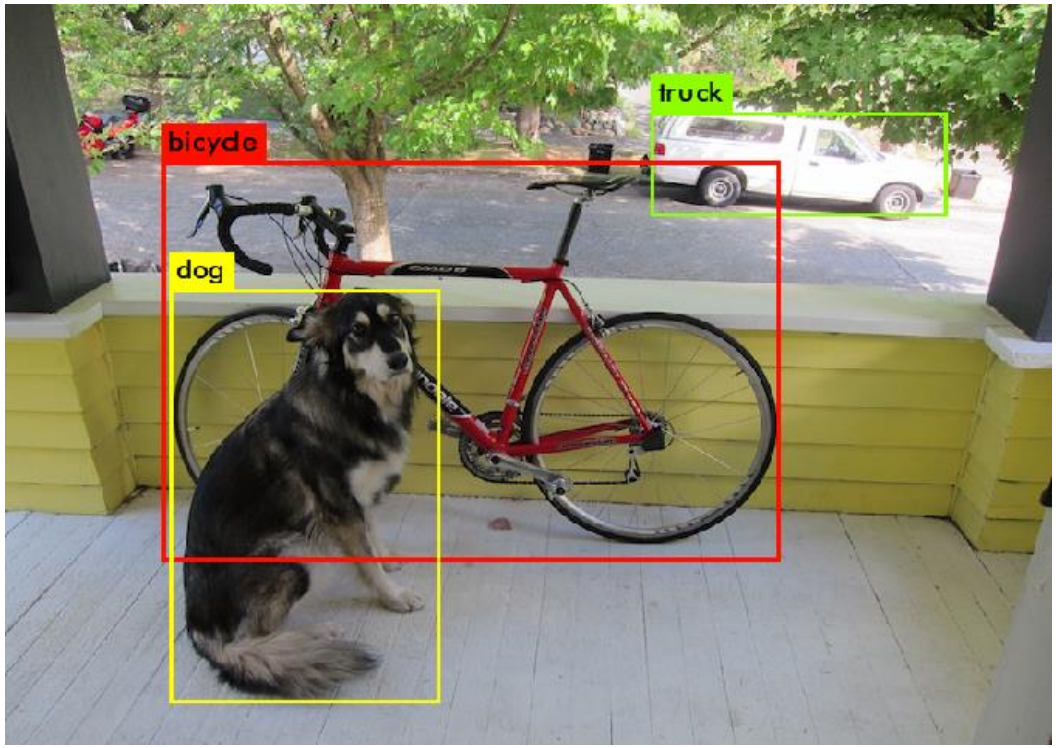


Visual Studio (window)로 구현

1) 매우 복잡
2) CUDA / OpenCV 연동

Results 1

```
darknet detector test data/coco.data cfg/yolov3.cfg weight/yolov3.weights dog.jpg
```

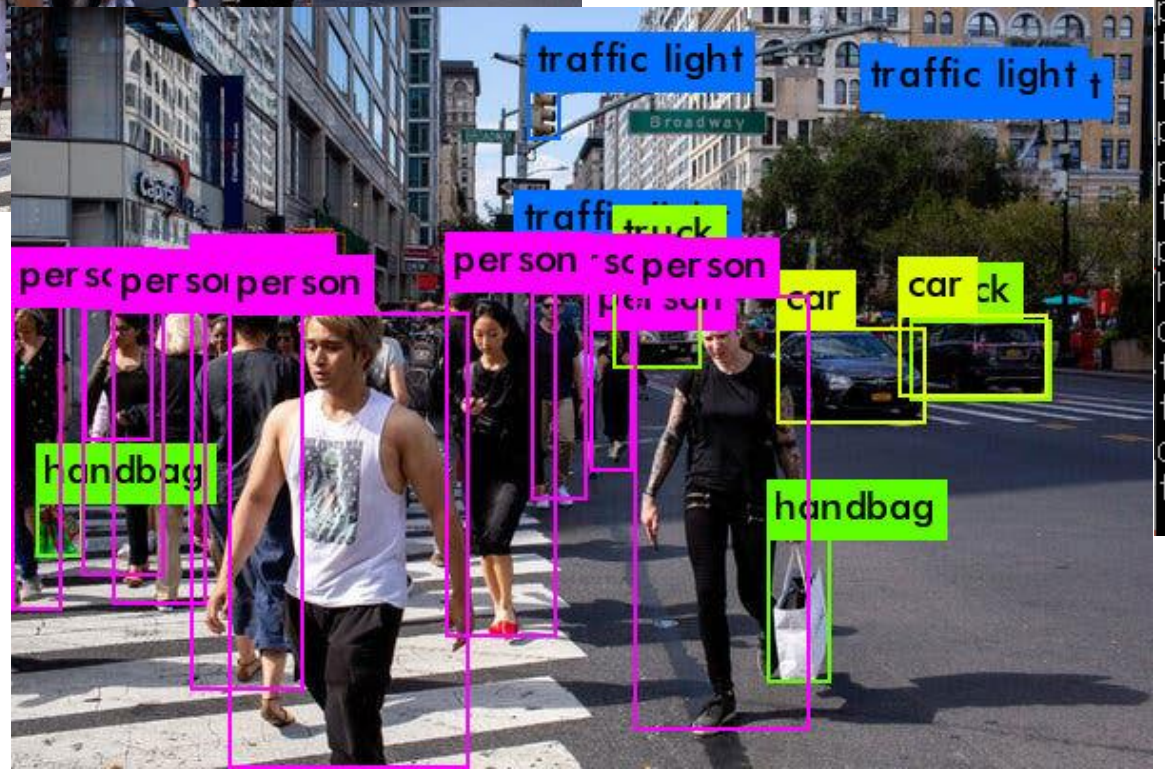



```
 layer   filters  size/strd(dil)      input                output
 0 conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
 1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
 2 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
 3 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
 4 Shortcut Layer: 1
 5 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
 6 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
 7 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
 8 Shortcut Layer: 5
 9 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
10 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
11 Shortcut Layer: 8
12 conv    256       3 x 3/ 2    104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
13 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
14 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
15 Shortcut Layer: 12
16 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
17 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
18 Shortcut Layer: 15
```

```
 91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 93 conv    255       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 255 0.177 BF
 94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
 95 route  91                                     ->   26 x  26 x 256
 96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
 97 upsample                 2x    26 x  26 x 128 ->   52 x  52 x 128
 98 route  97 36                                  ->   52 x  52 x 384
 99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
105 conv    255       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 255 0.353 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.864
```

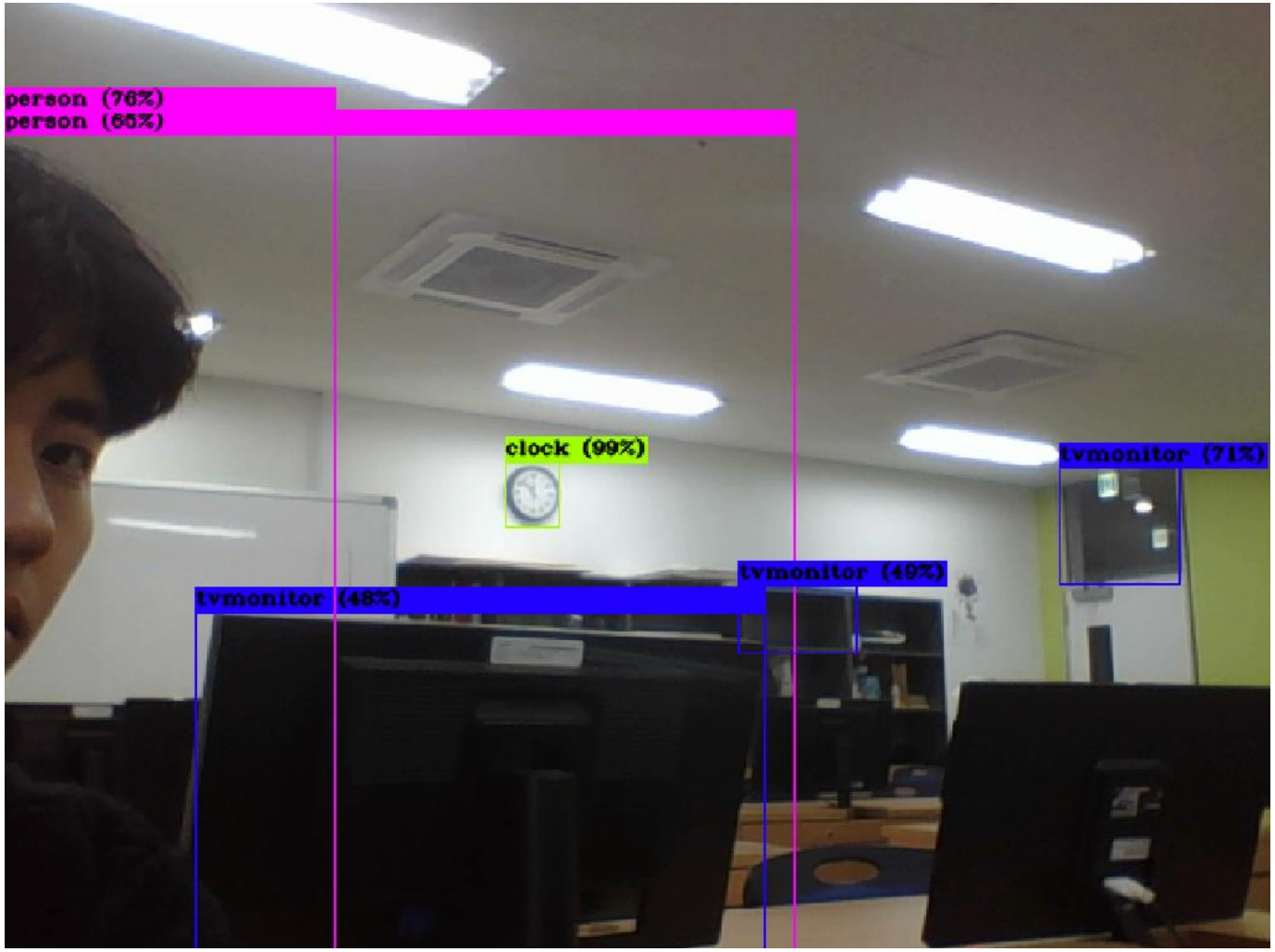
```
dog.jpg: Predicted in 269.237000 milli-seconds.
bicycle: 99%
dog: 100%
truck: 94%
```

Results 2



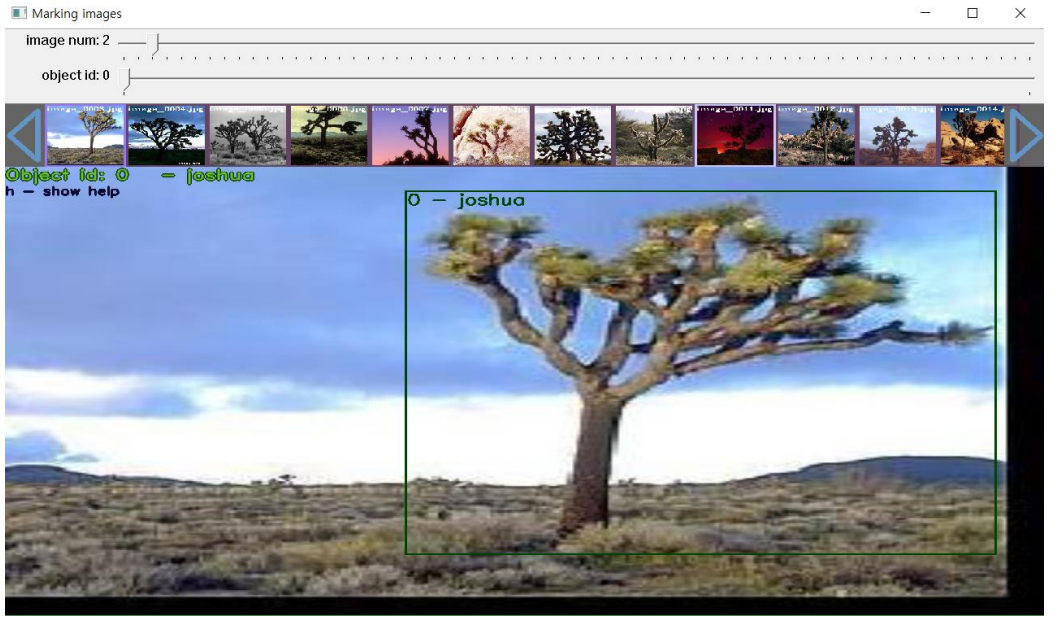

```
Done! Loaded 107 layers from weights-file
street.jpg: Predicted in 269.406000 milli-seconds.
person: 88%
handbag: 26%
person: 73%
person: 37%
person: 98%
person: 97%
person: 100%
person: 100%
traffic light: 30%
traffic light: 81%
person: 98%
person: 94%
truck: 95%
person: 100%
handbag: 86%
car: 99%
traffic light: 87%
traffic light: 75%
car: 83%
truck: 31%
```

Results 3




```
FPS:3.6
Objects:

clock: 99%
chair: 26%
person: 91%
person: 29%
tvmonitor: 42%
chair: 47%
tvmonitor: 69%

FPS:3.6
Objects:

clock: 99%
person: 38%
person: 87%
tvmonitor: 25%
tvmonitor: 47%
chair: 50%
tvmonitor: 70%

FPS:3.7
Objects:

clock: 99%
person: 33%
person: 87%
tvmonitor: 27%
tvmonitor: 32%
chair: 35%
tvmonitor: 75%
```

학습: YOLO_MARK



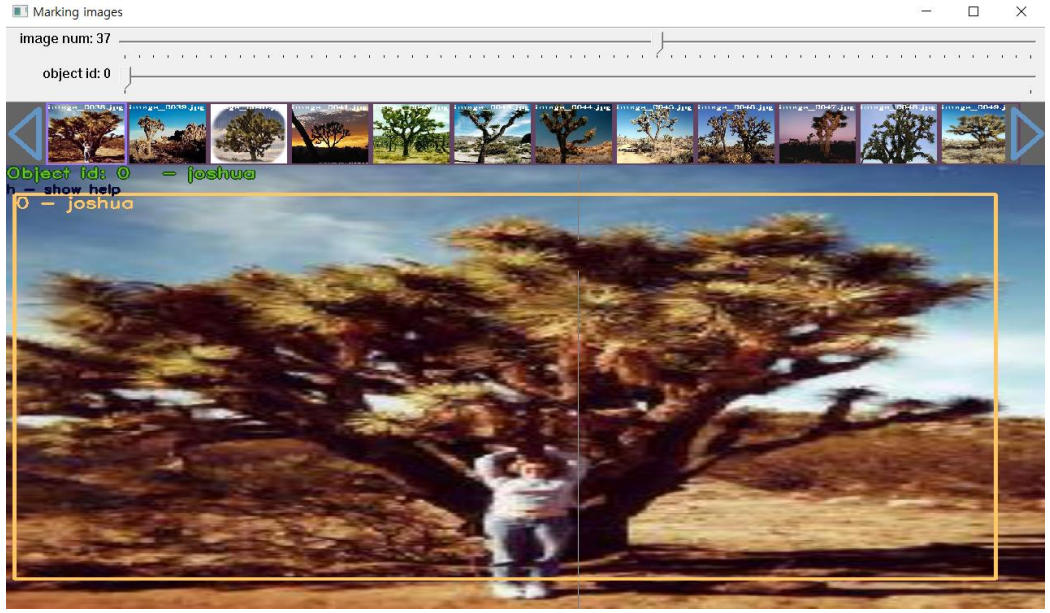


image_0001.jpg	2004-11-09 오후 3...	JPG 파일
image_0001.txt	2019-12-03 오후 9...	TXT 파일
image_0002.jpg	2004-11-09 오후 3...	JPG 파일
image_0002.txt	2019-12-03 오후 9...	TXT 파일
image_0003.jpg	2004-11-09 오후 3...	JPG 파일
image_0003.txt	2019-12-03 오후 9...	TXT 파일
image_0004.jpg	2004-11-09 오후 3...	JPG 파일
image_0004.txt	2019-12-03 오후 9...	TXT 파일
image_0005.jpg	2004-11-09 오후 3...	JPG 파일
image_0005.txt	2019-12-03 오후 9...	TXT 파일
image_0006.jpg	2004-11-09 오후 3...	JPG 파일
image_0006.txt	2019-12-03 오후 9...	TXT 파일
image_0007.jpg	2004-11-09 오후 3...	JPG 파일
image_0007.txt	2019-12-03 오후 9...	TXT 파일

Editing yolo-obj.cfg file

```
[net]
batch=64
subdivisions=8
height=416
width=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0001
max_batches = 45000
policy=steps
steps=100,25000,35000
scales=10,.1,.1
```

```
[net]
batch=8
subdivisions=16
height=416
width=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0001
max_batches = 1200
policy=steps
steps=100,25000,35000
scales=10,.1,.1
```

```
[convolutional]
size=1
stride=1
pad=1
filters=35
activation=linear

[region]
anchors = 1.08,1.19,
bias_match=1
classes=2
coords=4
num=5
softmax=1
jitter=.2
rescore=1
```

```
filters=30
```

```
classes=1
```

Filters
= Gx * Gy * (B * (5 + C))
-
(B = Cell 당 Bounding Box 개수 = num)
(C = classes 종류 수)
-
➔ 5 * (5 + 1) = **30**

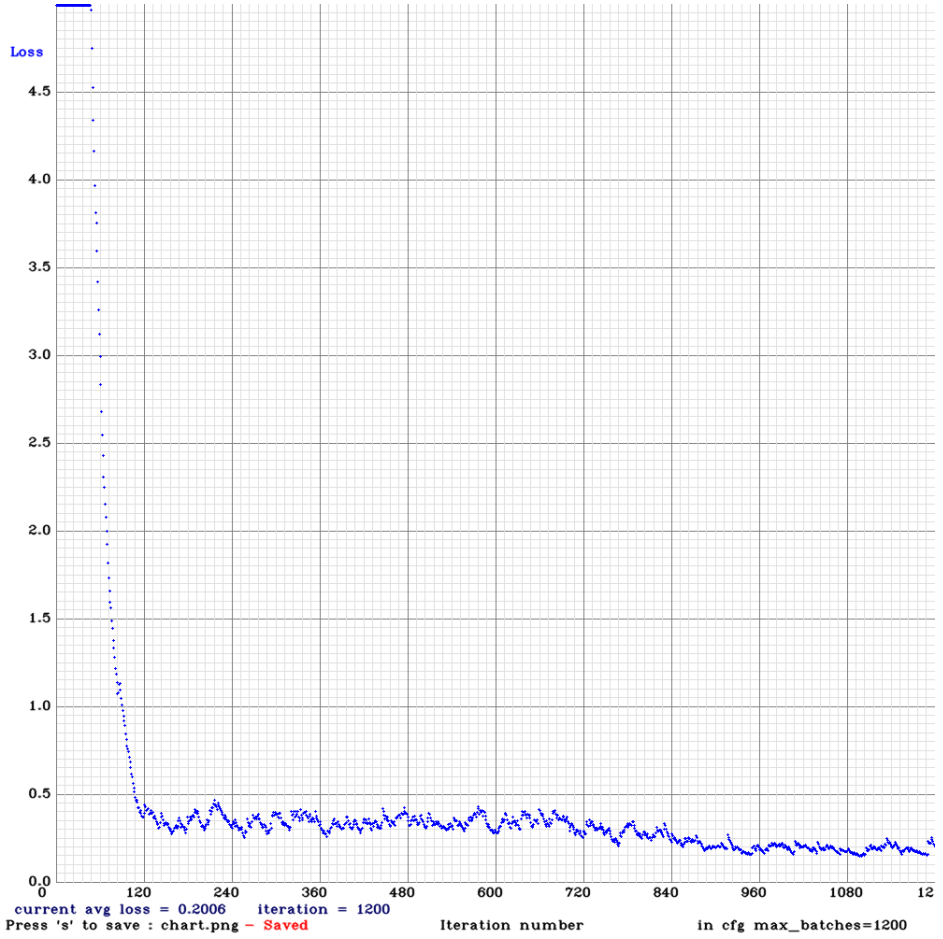
```
>darknet.exe detector train data/obj.data yolo-obj.cfg weight/extraction.conv.weights
```

Training

Dataset = 64, Batch = 8, Epoch = 1200 (5 hours)

```
 1: 11.856052, 11.856052 avg loss, 0.000100 rate, 14.364000 seconds, 16 images
Loaded: 0.000000 seconds
Region Avg IOU: 0.315456, Class: 1.000000, Obj: 0.570019, No Obj: 0.403891, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.416143, Class: 1.000000, Obj: 0.536885, No Obj: 0.404072, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.307929, Class: 1.000000, Obj: 0.660938, No Obj: 0.403797, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.530347, Class: 1.000000, Obj: 0.472383, No Obj: 0.403872, Avg Recall: 0.500000,  count: 2
Region Avg IOU: 0.595959, Class: 1.000000, Obj: 0.726969, No Obj: 0.404875, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.660955, Class: 1.000000, Obj: 0.355971, No Obj: 0.405684, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.171451, Class: 1.000000, Obj: 0.380485, No Obj: 0.403649, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.390751, Class: 1.000000, Obj: 0.534355, No Obj: 0.405844, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.433815, Class: 1.000000, Obj: 0.356365, No Obj: 0.404095, Avg Recall: 0.666667,  count: 3
Region Avg IOU: 0.590513, Class: 1.000000, Obj: 0.521027, No Obj: 0.404295, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.287756, Class: 1.000000, Obj: 0.397978, No Obj: 0.405545, Avg Recall: 0.166667,  count: 6
Region Avg IOU: 0.454096, Class: 1.000000, Obj: 0.199343, No Obj: 0.404597, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.577058, Class: 1.000000, Obj: 0.469409, No Obj: 0.404965, Avg Recall: 0.500000,  count: 2
Region Avg IOU: 0.381875, Class: 1.000000, Obj: 0.465776, No Obj: 0.403929, Avg Recall: 0.500000,  count: 2
Region Avg IOU: 0.365735, Class: 1.000000, Obj: 0.413922, No Obj: 0.403910, Avg Recall: 0.000000,  count: 2
Region Avg IOU: 0.545493, Class: 1.000000, Obj: 0.556809, No Obj: 0.403207, Avg Recall: 1.000000,  count: 1

 2: 9.698283, 11.640276 avg loss, 0.000100 rate, 15.198000 seconds, 32 images
Loaded: 0.001000 seconds
Region Avg IOU: 0.423900, Class: 1.000000, Obj: 0.338053, No Obj: 0.310912, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.431652, Class: 1.000000, Obj: 0.396238, No Obj: 0.309303, Avg Recall: 0.333333,  count: 3
Region Avg IOU: 0.275678, Class: 1.000000, Obj: 0.363837, No Obj: 0.310092, Avg Recall: 0.000000,  count: 1
Region Avg IOU: 0.303211, Class: 1.000000, Obj: 0.560663, No Obj: 0.311937, Avg Recall: 0.000000,  count: 1
```

```
 1199: 0.163459, 0.210698 avg loss, 0.001000 rate, 15.137000 seconds, 19184 images
Loaded: 0.001000 seconds
Region Avg IOU: 0.606155, Class: 1.000000, Obj: 0.010319, No Obj: 0.007853, Avg Recall: 1.000000,  count: 2
Region Avg IOU: 0.791877, Class: 1.000000, Obj: 0.019231, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.438305, Class: 1.000000, Obj: 0.012407, No Obj: 0.007853, Avg Recall: 0.500000,  count: 2
Region Avg IOU: 0.851613, Class: 1.000000, Obj: 0.013014, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.769861, Class: 1.000000, Obj: 0.014925, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.671525, Class: 1.000000, Obj: 0.010627, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.749163, Class: 1.000000, Obj: 0.013686, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.654242, Class: 1.000000, Obj: 0.027353, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.644621, Class: 1.000000, Obj: 0.029380, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.802083, Class: 1.000000, Obj: 0.013686, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.888562, Class: 1.000000, Obj: 0.023396, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.858748, Class: 1.000000, Obj: 0.013014, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.772261, Class: 1.000000, Obj: 0.018886, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.848371, Class: 1.000000, Obj: 0.013093, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.854500, Class: 1.000000, Obj: 0.013686, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1
Region Avg IOU: 0.736559, Class: 1.000000, Obj: 0.027306, No Obj: 0.007853, Avg Recall: 1.000000,  count: 1

 1200: 0.110179, 0.200646 avg loss, 0.001000 rate, 15.085000 seconds, 19200 images
Saving weights to backup//yolo-obj_last.weights
Saving weights to backup//yolo-obj_final.weights
```



Problems/Improvements

1)

```
#Sigmoid the  centre_X, centre_Y. and object confidencce
prediction[:,:,0] = torch.sigmoid(prediction[:,:,0])
prediction[:,:,1] = torch.sigmoid(prediction[:,:,1])
prediction[:,:,4] = torch.sigmoid(prediction[:,:,4])
```

x, y 좌표에 대해서는 sigmoid 함수 필요(전체 이미지 크기를 벗어나지 않기 위해)
하지만 Object confidence에는 ReLU 함수를 사용하는 것이 더 효과적이지 않을까...

2)

학습을 위한 Epoch 크기가 너무 크다
혹은 Epoch 크기에 따른 학습 소요시간/정확도를 예측할 수 있으면 좋을 것