

사이버 물리시스템 보안

유해 동물 감지 시스템 구축

중간 발표

3조

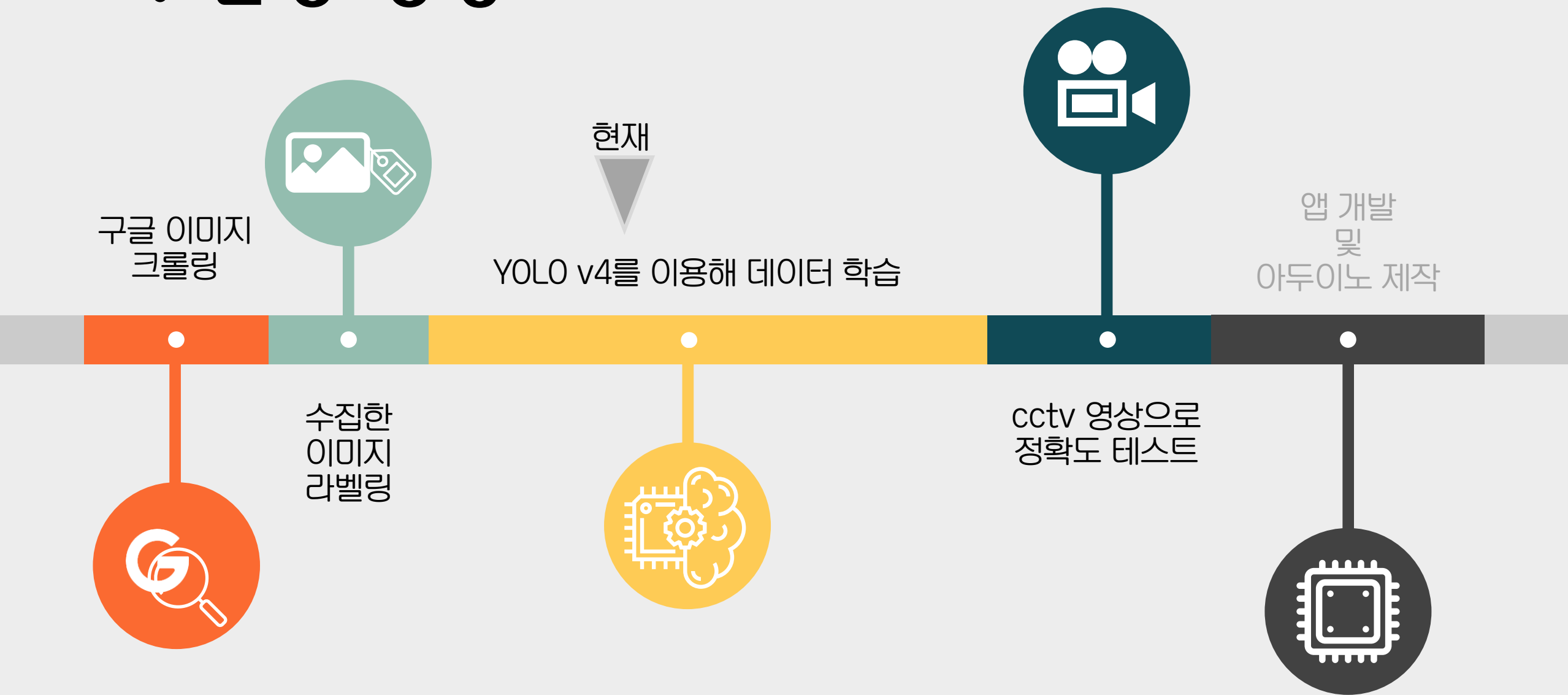
20185636 이은진

20186033 임채원

20180163 최혜정

20184141 황수인

1. 진행 상황



2. 구글 이미지 크롤링

```
animal.py > ...
1 from urllib.request import urlopen
2 from urllib.parse import quote_plus
3 from bs4 import BeautifulSoup
4 from selenium import webdriver
5
6 search= input('검색하고자 하는 사진 입력: ')
7 url = f'https://www.google.com/search?q={quote_plus(search)}&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjV9_Kx-sTpAhVXfd4KHZD9BcsQ_AUoAXoECBAQAw&biw=960&bih=960'
8
9
10 driver = webdriver.Chrome() # 구글 크롤링에서 필요한 코드
11 driver.get(url)
12
13 html = driver.page_source
14 soup = BeautifulSoup(html)
15
16 img = soup.select('.rg_i.Q4LuWd.tx8vtf') #클래스의 값 넣어주기
17
18 n=1
19 imgurl = []
20 for i in img:
21     try:
22         imgurl.append(i.attrs["src"])
23     except KeyError:
24         imgurl.append(i.attrs["data-src"])
25
26 for i in imgurl: #반복문으로 자동으로 이미지 저장
27     urlopen(i, str(n) + '.jpg')
28     n +=1
29
```

검색하고자 하는 사진: 멧돼지

```
Python Debug Console
C:\Users\chj09\Desktop\파이썬>python animal.py
검색하고자 하는 사진 입력: 멧돼지

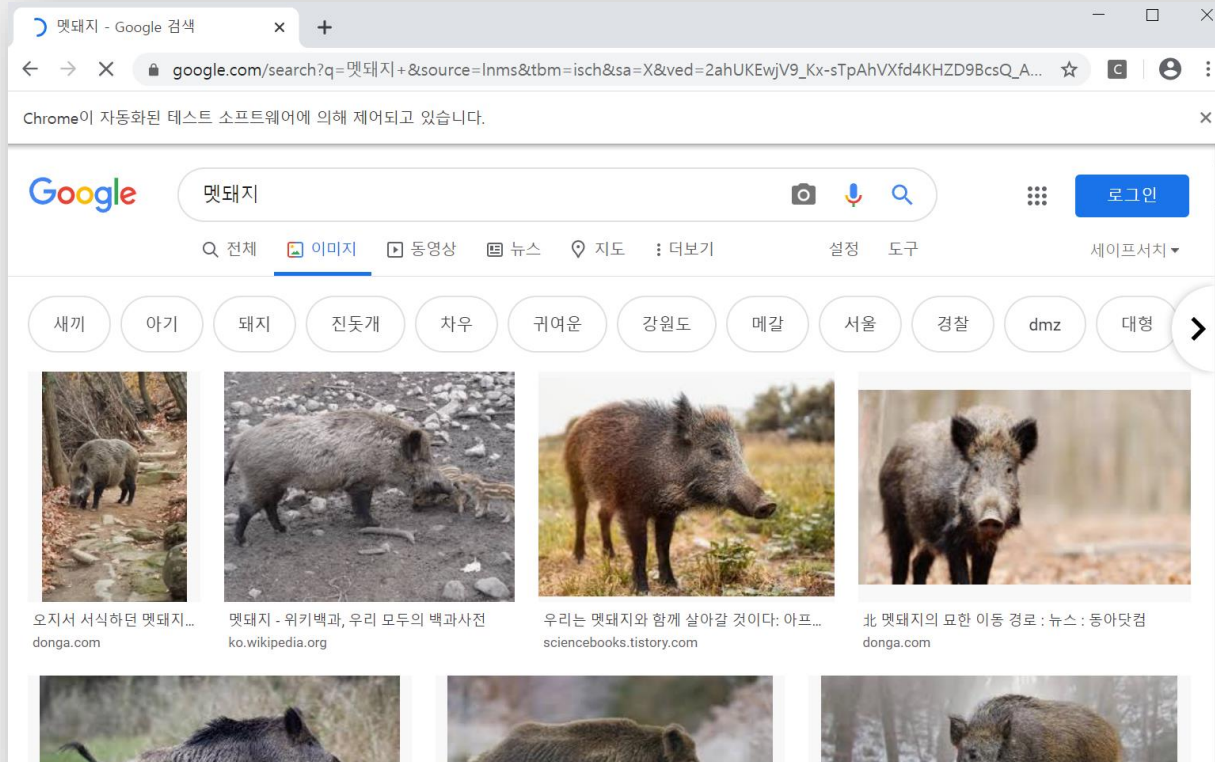
DevTools listening on ws://127.0.0.1:54487/devtools/browser/c00cda4-f95f-43e9-b833-4a05b44dc151
[4876:75388:0521/223131.881:ERROR:browser_switcher_service.cc(238)] XXX Init()
animal.py:14: UserWarning: No parser was explicitly specified, so I'm using the best available HTML parser for this system ("lxml"). This usually is not a problem, but if you run this code on another system, or in a different virtual environment, it may use a different parser and behave differently.

The code that caused this warning is on line 14 of the file animal.py. To get rid of this warning, pass the additional argument 'features="lxml"' to the BeautifulSoup constructor.

soup = BeautifulSoup(html)

C:\Users\chj09\Desktop\파이썬>python animal.py
검색하고자 하는 사진 입력: 멧돼지
```

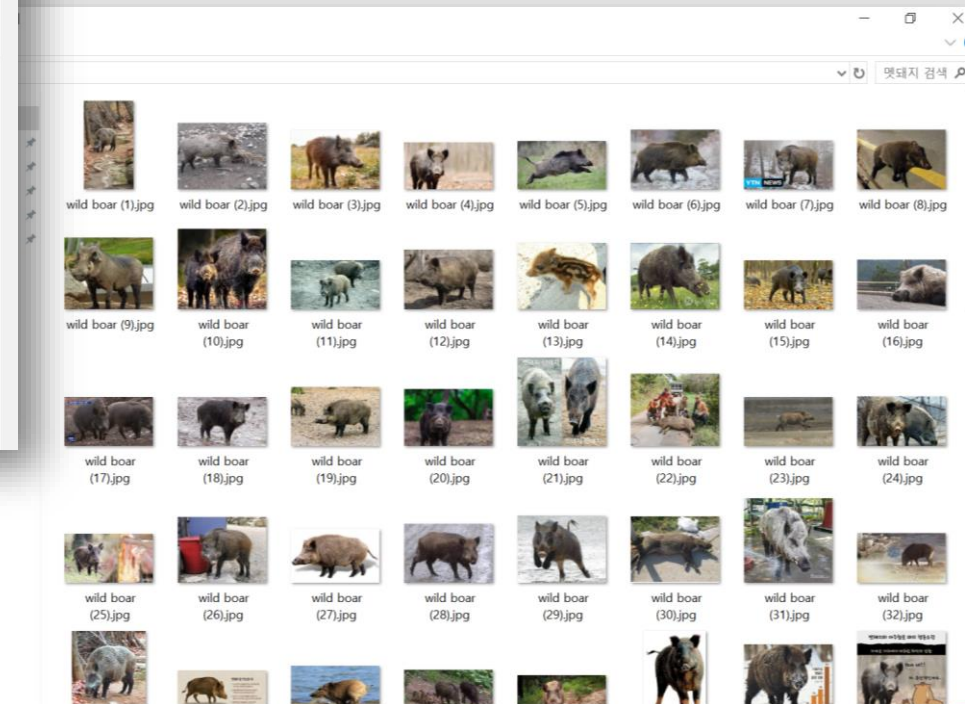
2. 구글 이미지 크롤링



크롤링이 정상적으로 완료되면
'다운로드' 폴더에 지정한 수만큼
사진이 다운로드 된다

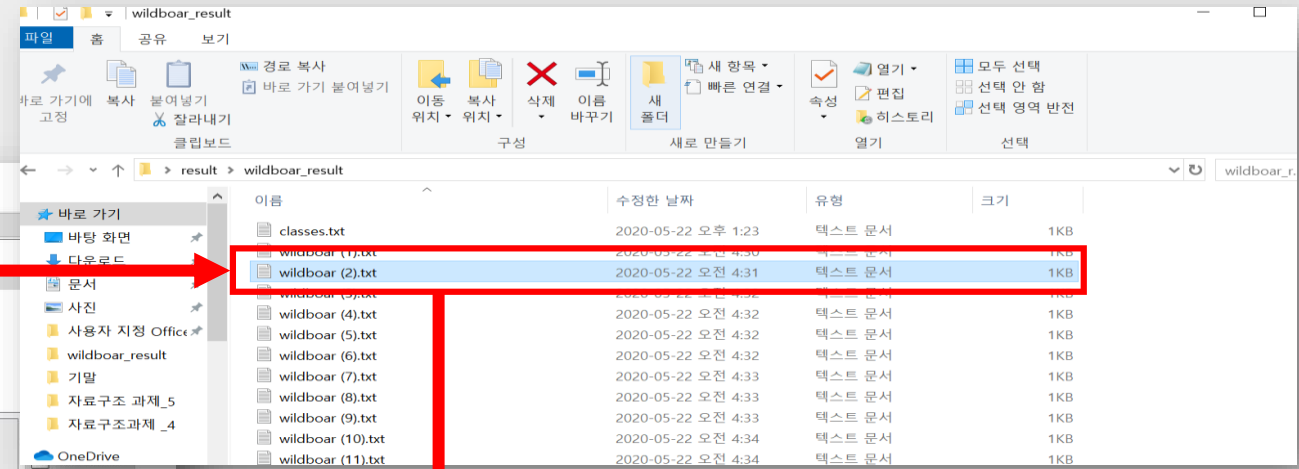
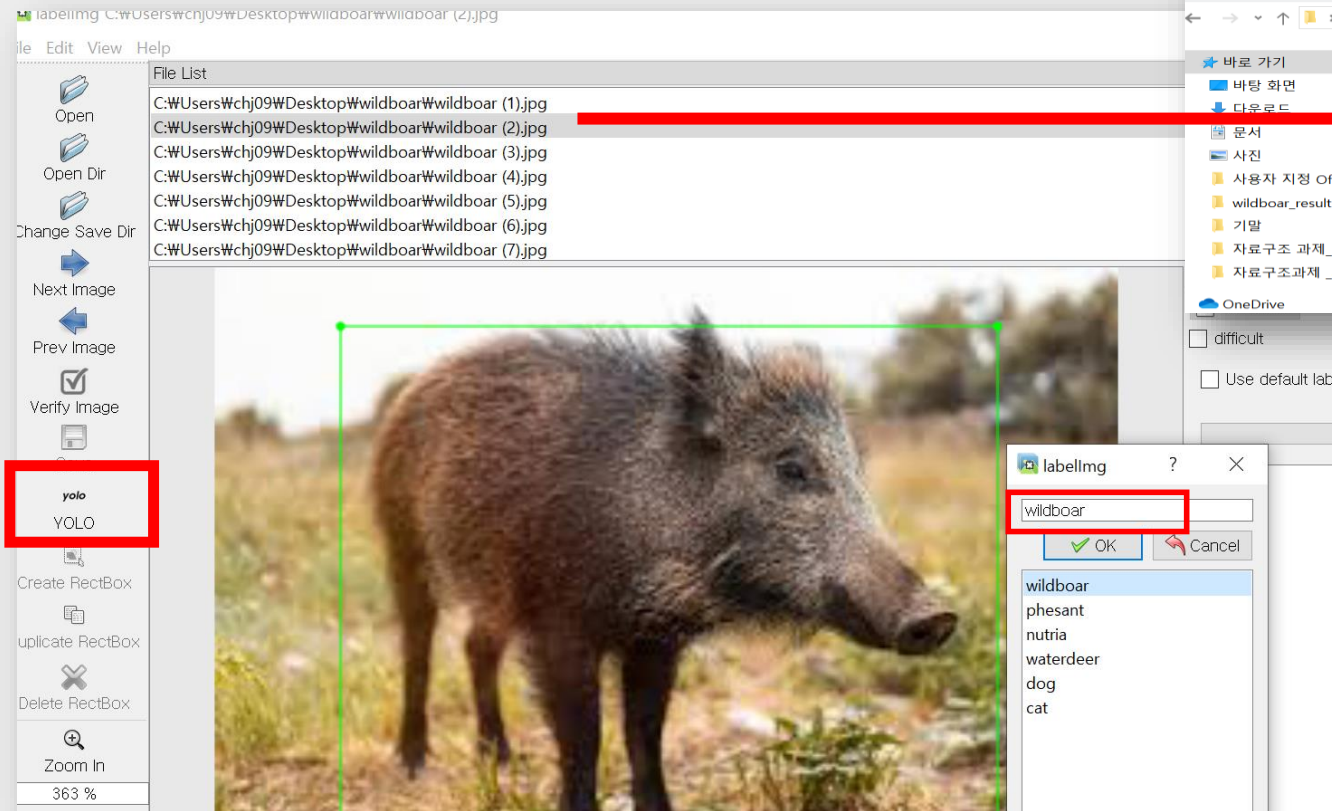
멧돼지 200장
꿩 200장
뉴트리아 200장
고라니 200장
고양이 300장
개 200장

멧돼지 152장
꿩 155장
뉴트리아 154장
고라니 165장
고양이 256장
개 164장



3. 이미지 라벨링

labellmg 사용



학습 이미지에서 학습하고자 하는 객체의 좌표 정보 추출

wildboar (2).txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말

0 0.496364 0.521858 0.672727 0.901639

4. 데이터 학습

Yolo-v4 and Yolo-v3/v2 for Windows and Linux

(neural network for object detection) - Tensor Cores can be used on [Linux](#) and [Windows](#)

Paper Yolo v4: <https://arxiv.org/abs/2004.10934>

AlexeyAB

More details: <http://pjreddie.com/darknet/yolo/>

<https://github.com/AlexeyAB/darknet>

Darknet Continuous Integration passing PASSED build passing contributors 56 license Unlicense DOI 10.5281/zenodo.3829035
cs.CV arXiv:2004.10934

기존의 darknet yolo를 커스텀 데이터로 학습을 하거나
학습 중 loss, mAP 값의 변화를 확인하기에 용이

4. 데이터 학습

<darknet detector train data/obj.data cfg/yolo-obj.cfg backup/yolo-obj_last.weights>

detector.c – train_detector

```
void train_detector(char *datacfg, char *cfgfile, ch
{
    option_list.c
    list *options = read_data_cfg(datacfg);
    char *train_images = option_find_str(options, "t
    char *valid_images = option_find_str(options, "v
    char *backup_directory = option_find_str(options
```

7 lines (6 sloc) | 141 Bytes

obj.data

```
1 classes= 6
2 train = data/train.txt
3 valid = data/test.txt
4 #difficult = data/difficult_2007_test.txt
5 names = data/obj.names
6 backup = backup/
```

6 lines (6 sloc) | 41 Bytes

obj.names

```
1 wildboar
2 pheasant
3 nutria
4 waterdeer
5 dog
6 cat
```

1046 lines (1046 sloc) | 81.8 KB

train.txt

```
1 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (1).jpg
2 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (10).jpg
3 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (100).jpg
4 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (101).jpg
5 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (102).jpg
6 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (103).jpg
7 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (104).jpg
8 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (105).jpg
9 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (106).jpg
10 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (107).jpg
11 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (108).jpg
12 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (109).jpg
13 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (11).jpg
```

내 드라이브 > darknet-master > darknet-master > backup

파일



yolo-obj_1000.weights



yolo-obj_2000.weights



yolo-obj_3000.weights

4. 데이터 학습

detector.c – test_detector

```
void test_detector(char *datacfg, char *cfgfile, char
float hier_thresh, int dont_show, int ext_output,
{
    option_list.c
    list *options = read_data_cfg(datacfg);
    char *name_list = option_find_str(options, "names
    int names_size = 0;
    char **names = get_labels_custom(name_list, &name
```

학습을 위한 내용이 담긴 data 파일
데이터셋의 경로, 클래스 개수 및
이름 정보, 학습의 결과로 나온 가중치
파일을 저장할 경로 등

7 lines (6 sloc) | 141 Bytes

obj.data

```
1 classes= 6
2 train = data/train.txt
3 valid = data/test.txt
4 #difficult = data/difficult_2007_test.txt
5 names = data/obj.names
6 backup = backup/
```

클래스 이름

6 lines (6 sloc) | 41 Bytes

```
1 wildboar
2 pheasant
3 nutria
4 waterdeer
5 dog
6 cat
```

obj.names

데이터셋에 담긴
각 이미지 파일의 경로

.8 KB

train.txt

```
1 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (1).jpg
2 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (10).jpg
3 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (100).jpg
4 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (101).jpg
5 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (102).jpg
6 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (103).jpg
7 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (104).jpg
8 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (105).jpg
9 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (106).jpg
10 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (107).jpg
11 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (108).jpg
12 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (109).jpg
13 /content/gdrive/My Drive/darknet-master/darknet-master/data/obj/cat (11).jpg
```

내 드라이브 > darknet-master > darknet-master > backup

파일



yolo-obj_1000.weights



yolo-obj_2000.weights



yolo-obj_3000.weights

학습에 따른 가중치 → 1000단위 및 가장 최신의 가중치 저장

4. 데이터 학습

<darknet detector train data/obj.data cfg/yolo-obj.cfg backup/yolo-obj_last.weights>

```
1 [net]
2 # Testing
3 #batch=1
4 #subdivisions=1
5 # Training
6 batch=64
7 subdivisions=64
8 width=608
9 height=608
10 channels=3
```

Batch = 모든 데이터세트를
한번의 epoch으로 돌리기에는
메모리의 한계
→ 몇 개로 나누어서(batch)
학습을 돌리게 됨
(batch 마다 학습 = iteration)
Subdivisions = Batch를 얼마나
나눠서 학습을 하는지에 대한
설정 값

```
17
18 learning_rate=0.001
19 burn_in=1000
20 max_batches = 12000
21 policy=steps
22 steps=9600,10800
23 scales=.1,.1
24
25 #cutmix=1
26 mosaic=1
27
```

Max_batches = 언제까지
iteration을 돌 건지에 대한
설정값
Steps = 함수가 최대 얼마나
돌 건지에 대한 설정값
(보통 max_batches의
80%, 90%로 설정)

967 [yolo]	1055 [yolo]	1143 [yolo]
968 mask = 0,1,2	1056 mask = 3,4,5	1144 mask = 6,7,8
969 anchors = 12, 16	1057 anchors = 12, 16, 19	1145 anchors = 12, 16,
970 classes=6	1058 classes=6	1146 classes=6

959 [convolutional]	1047 [convolutional]	1135 [convolutional]
960 size=1	1048 size=1	1136 size=1
961 stride=1	1049 stride=1	1137 stride=1
962 pad=1	1050 pad=1	1138 pad=1
963 filters=33	1051 filters=33	1139 filters=33
964 activation=linear	1052 activation=linear	1140 activation=linear

Classes = 클래스의 개수
Filters = (5+classes) x 3

4. 데이터 학습

Google Colaboratory를 사용하여 학습 실행

```
[ ] 1 from google.colab import drive
    2 drive.mount('/content/gdrive/') 구글드라이브 - colab 연동
```

```
[ ] 1 cd /content/gdrive/My Drive/darknet-master/darknet-master 실행 디렉토리 변경
```

```
[ ] 1 !ls
```

```
▶ 1 !apt install libopencv-dev
   2 !apt install gcc-5 g++-5 -y Opencv, gcc 설치
```

```
[ ] 1 import os
    2 os.environ['PATH'] += ':/usr/local/cuda/bin'
                                           설치한 gcc를 cuda에 링크
```

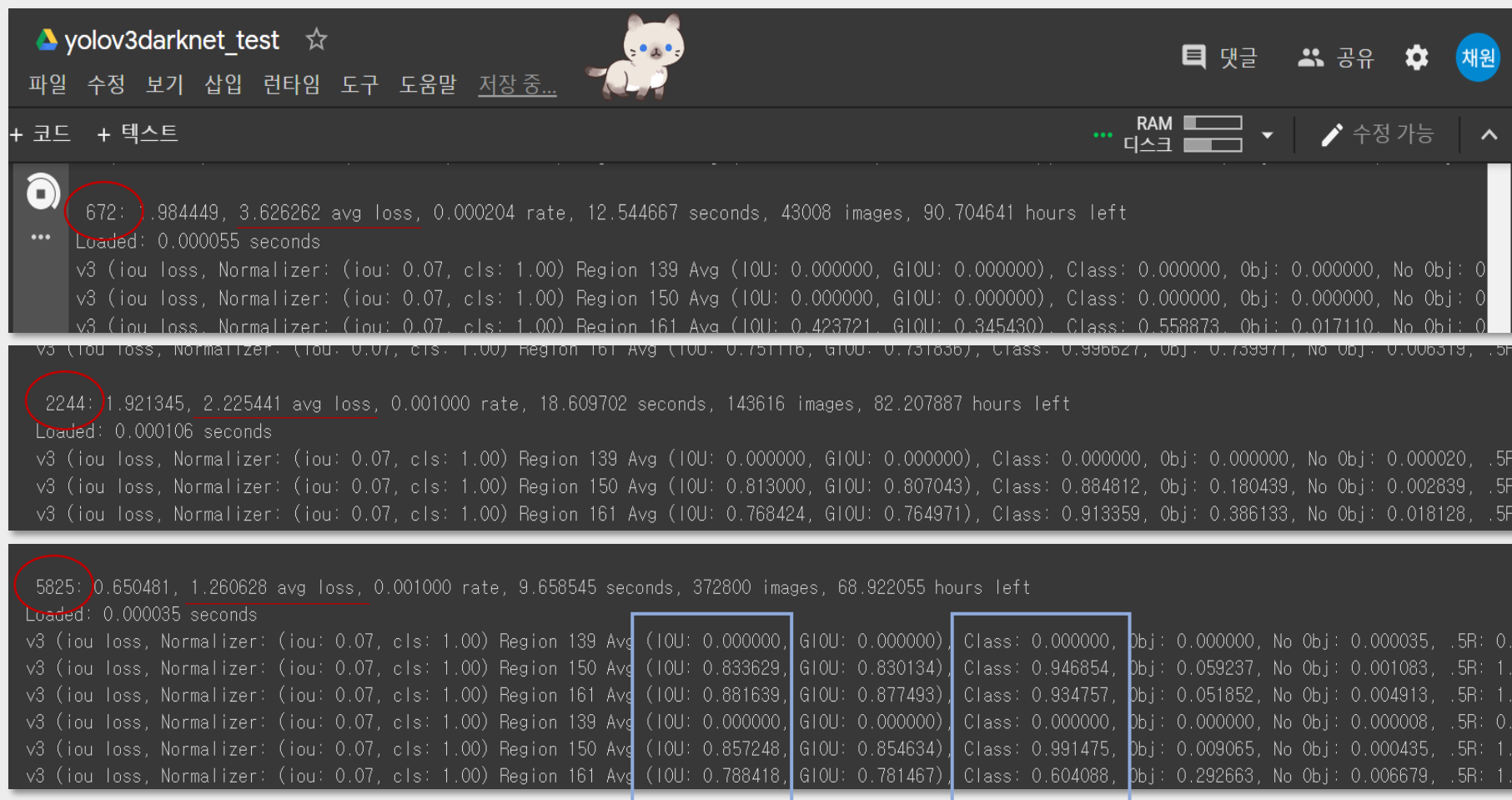
```
[ ] 1 !ln -s /usr/bin/gcc-5 /usr/local/cuda/bin/gcc
    2 !ln -s /usr/bin/g++-5 /usr/local/cuda/bin/g++
```

```
[ ] 1 !sed -i 's/OPENCV=0/OPENCV=1/g' Makefile
    2 !sed -i 's/GPU=0/GPU=1/g' Makefile Opencv, gpu, cudnn 사용으로 옵션 변경
    3 !sed -i 's/CUDNN=0/CUDNN=1/g' Makefile
```

```
[ ] 1 !make 설정사항 반영하여 실행
```

```
▶ 1 !./darknet detector train data/obj.data cfg/yolo-obj.cfg backup/yolo-obj_last.weights -dont_show 학습 시작
```

4. 데이터 학습



```
yolov3darknet_test ☆
파일 수정 보기 삽입 런타임 도구 도움말 저장 중...
+ 코드 + 텍스트
RAM 디스크 수정 가능
672: 0.984449, 3.626262 avg loss, 0.000204 rate, 12.544667 seconds, 43008 images, 90.704641 hours left
Loaded: 0.000055 seconds
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.000000, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.000000, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.423721, GIOU: 0.345430), Class: 0.558873, Obj: 0.017110, No Obj: 0.000000, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
2244: 1.921345, 2.225441 avg loss, 0.001000 rate, 18.609702 seconds, 143616 images, 82.207887 hours left
Loaded: 0.000106 seconds
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.000020, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.813000, GIOU: 0.807043), Class: 0.884812, Obj: 0.180439, No Obj: 0.002839, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.768424, GIOU: 0.764971), Class: 0.913359, Obj: 0.386133, No Obj: 0.018128, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
5825: 0.650481, 1.260628 avg loss, 0.001000 rate, 9.658545 seconds, 372800 images, 68.922055 hours left
Loaded: 0.000035 seconds
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.000035, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.833629, GIOU: 0.830134), Class: 0.946854, Obj: 0.059237, No Obj: 0.001083, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.881639, GIOU: 0.877493), Class: 0.934757, Obj: 0.051852, No Obj: 0.004913, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.000008, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.857248, GIOU: 0.854634), Class: 0.991475, Obj: 0.009065, No Obj: 0.000435, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.788418, GIOU: 0.781467), Class: 0.604088, Obj: 0.292663, No Obj: 0.006679, .5R: 0.000000, .9R: 0.000000, loss: 0.000000)
```

Avg loss

- 평균 손실율
- 알고리즘이 예측한 객체의 위치와 실제 객체의 위치의 차이 = 오차
- 낮을수록 좋음
- 데이터셋이 작을수록 낮은 손실율(0.1~0.05)을 갖도록 학습

IOU

- 실제 객체와 bounding box의 교차율
- 1에 가까울수록 좋음

Class

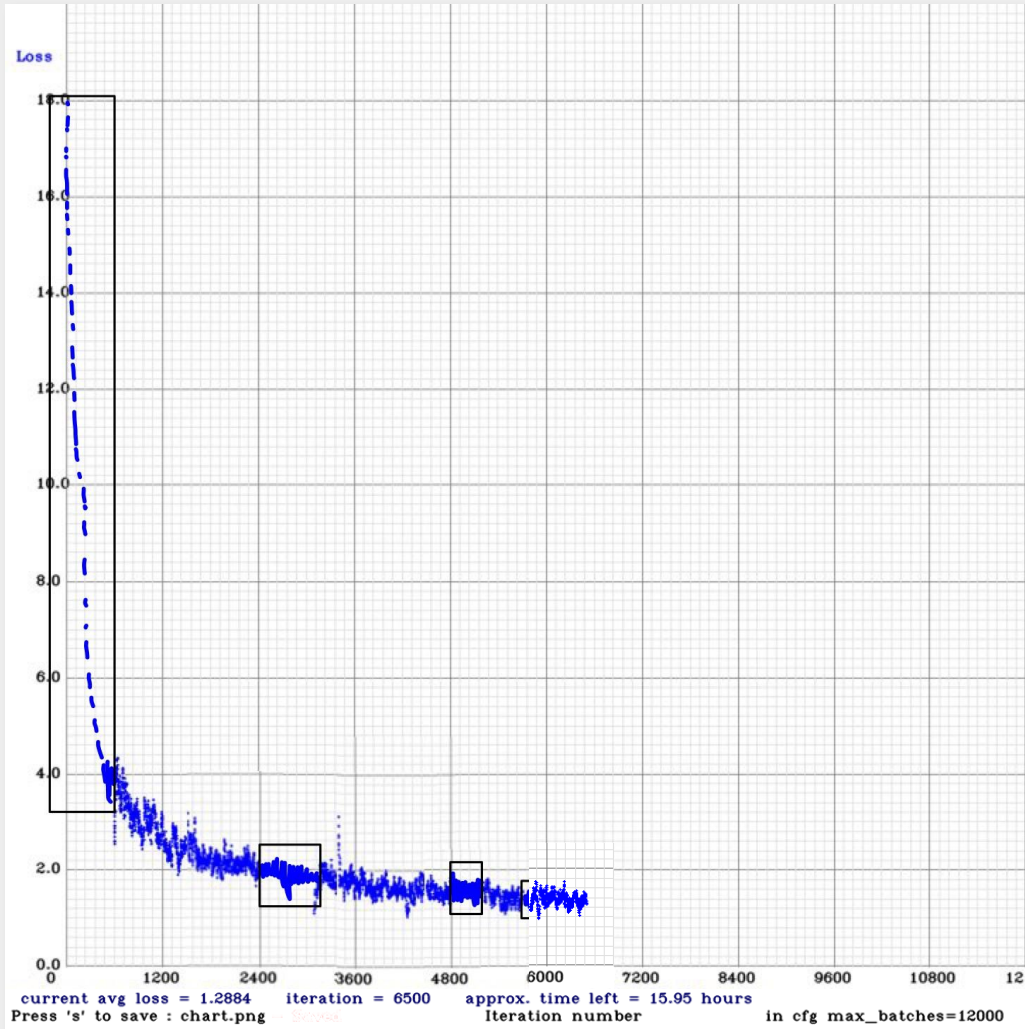
- 1에 가까운 값일수록 학습이 잘 되어가고 있다는 뜻

4. 데이터 학습

현재까지의 학습 결과

Iteration : 6500

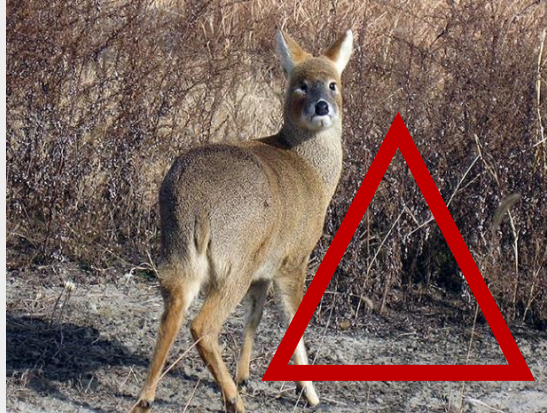
Current avg loss : 1.2884



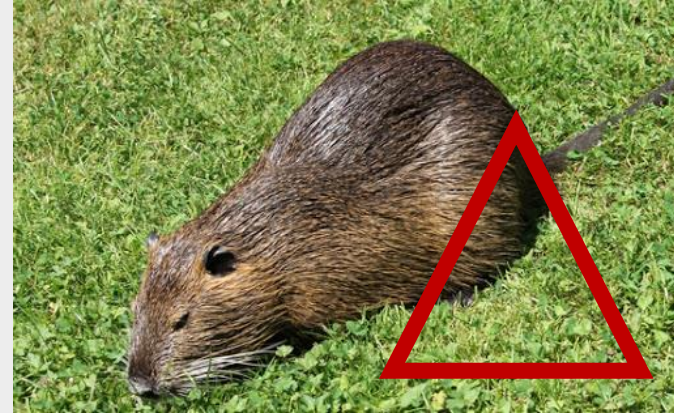
4. 데이터 학습



data/boar.jpg: Predicted in 644.621000 milli-sec
wildboar: 93%



data/gorani.jpg: Predicted in 32.468000
waterdeer: 77%



data/nut.jpg: Predicted in 32.365000 milli-sec
nutria: 76%



data/cat.jpg: Predicted in 32.376000 mill
cat: 65%

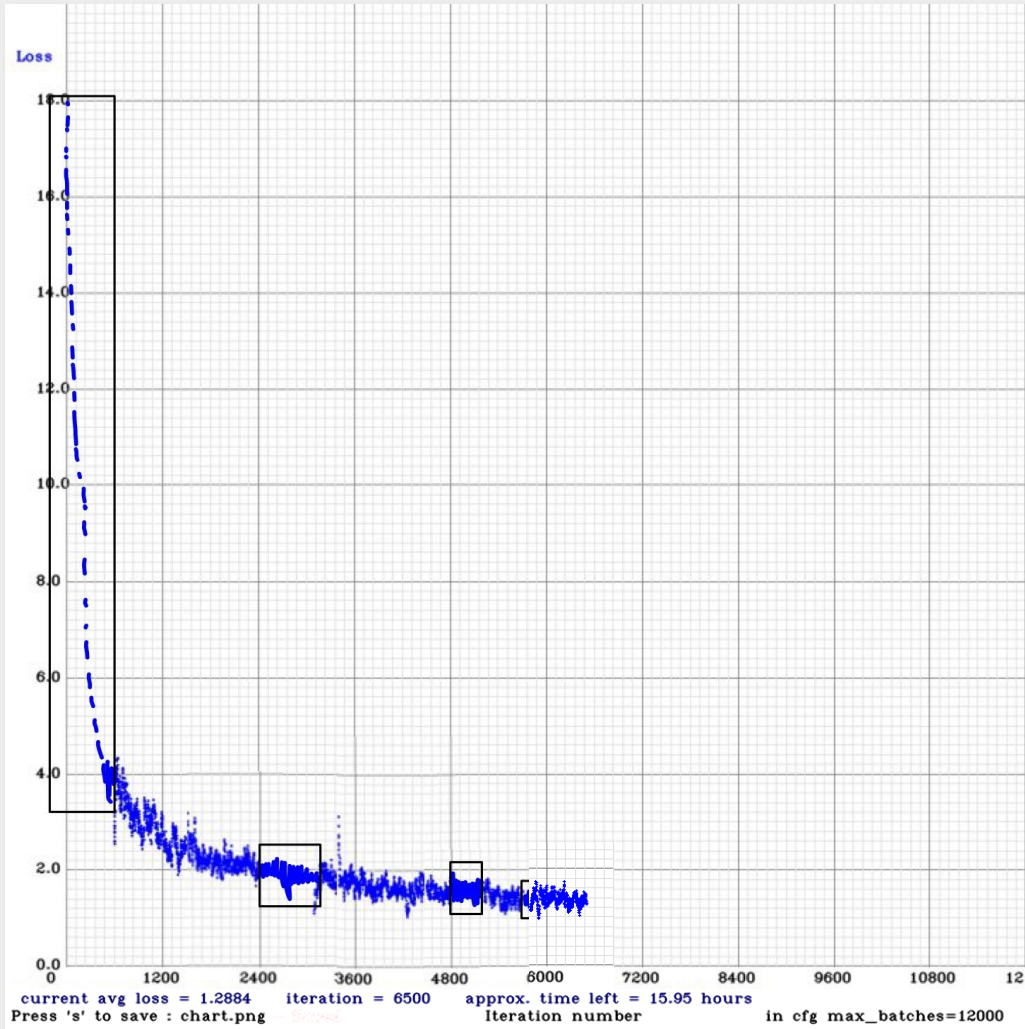


data/phsnt.jpg: Predicted in 32.
phesant: 48%



data/dog.jpg: Predicted in 32.3
cat: 62%

4. 데이터 학습



현재까지의 학습 결과

Iteration : 6500

Current avg loss : 1.2884

목표

Iteration : 대략 12000

Current avg loss : 0.05~0.1

감사합니다