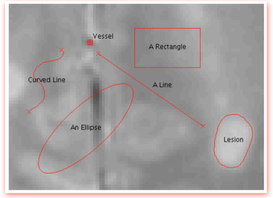
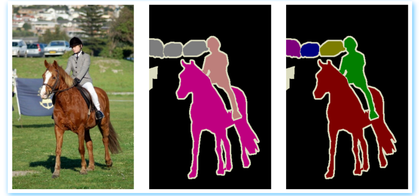
Pascal VOC 2012 dataset

YOLO = a system for detecting objects on the Pascal VOC 2012 dataset



ROI annotations



segmentation ground truth

The [PASCAL](#) Visual Object Classes Homepage

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

Year	Statistics	New developments	Notes
2012	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.	• Size of segmentation dataset substantially increased. • People in action classification dataset are additionally annotated with a reference point on the body.	• Datasets for classification, detection and person layout are the same as VOC2011.

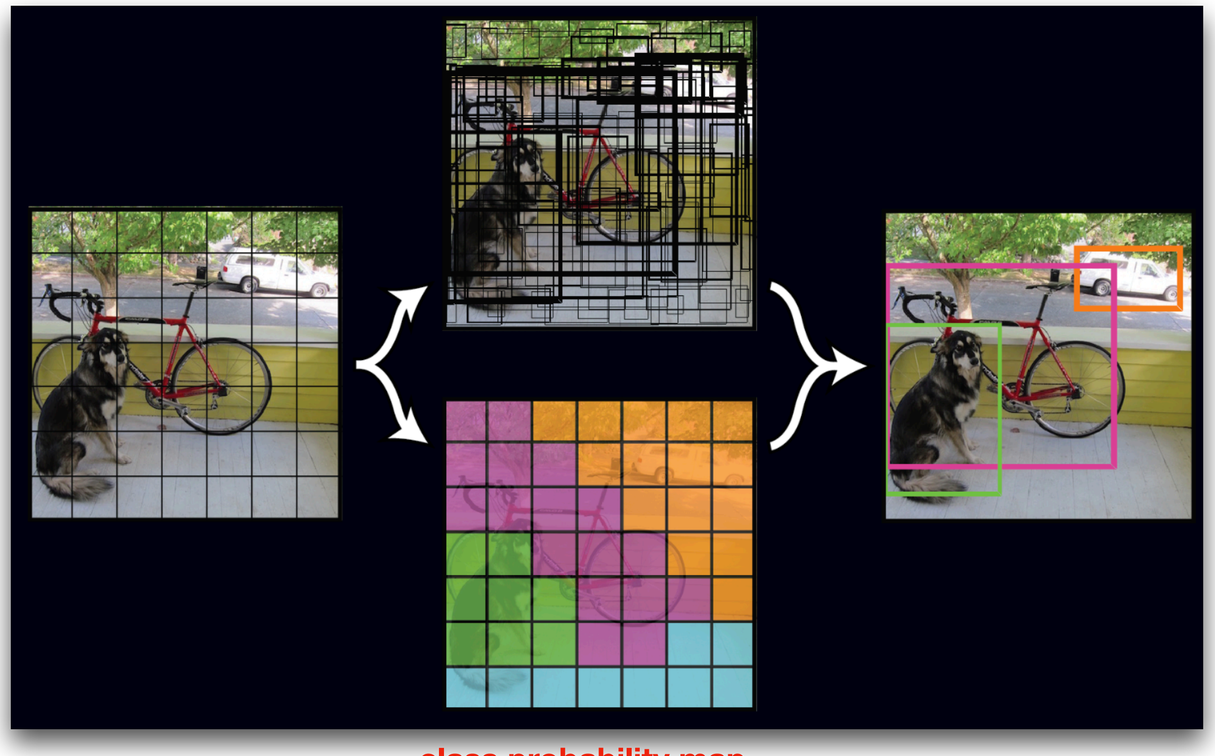
- **20 classes:**
  - (person), (bird, cat, cow, dog, horse, sheep), (airplane, bicycle, boat, bus, car, motorbike, train) 등
- train/val data with annotation & test data without annotation
- ROI = Region of Interest Annotation (= Bounding Box)
- `cat 2007_* 2012_train.txt > train.txt`

- **5 main tasks (총 7개)**
  - Classification, Detection, Segmentation, Action Classification, Large Scale Recognition 등
    - ✳ Action Classification 란
      - a specified person's (bounding box로 표시) 10 action classes
      - action: jumping; phoning; running; playing a musical instrument 등
  - 각 task에 따른 Submission of Results의 format이나 Evaluation 등 각각 다르므로 메뉴얼 확인 필요

YOLO v1 (저번주)

How It Works

Bounding box + confidence



class probability map

- network
  - a single neural network to the full image
  - divides the image into regions & predicts bounding boxes and probabilities for each region
  - NMS: only see high scoring detections
- evaluation
  - trained with the IOU prediction (*threshold*) which gives slightly better mAP scores
- ! global context, extremely fast

Darknet
YOLO v1

- **Darknet (Original)** pjreddie / darknet
  - *https://github.com/pjreddie/darknet/*
  - **Joseph Redmon이 YOLO 학습을 위해 개발한 프레임워크**
  - **open source neural network framework written in C and CUDA (supports CPU & GPU)**
  - 

(*https://pjreddie.com/darknet/yolo/*)
- **Alexey** AlexeyAB / darknet forked from pjreddie/darknet
  - *https://github.com/AlexeyAB/yolo-windows*
  - **YOLOv4의 저자인 Alexey Bochkovskiy의 버전**
  - **특징: Darknet을 fork함. 정리가 잘 되어 있음**

prepare Data set

**Generate Labels for VOC

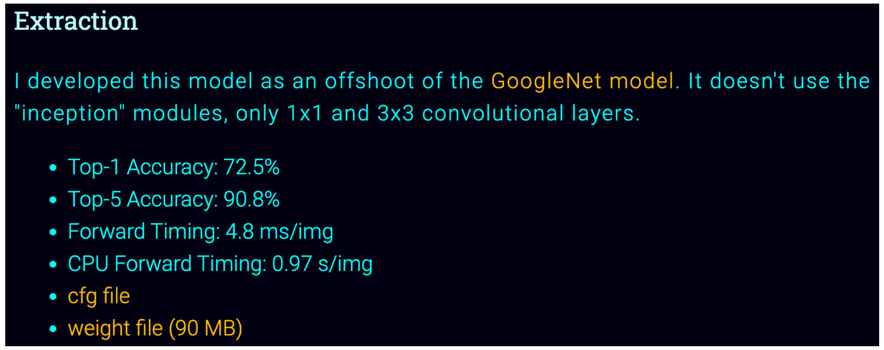
- Get The Pascal VOC Data
- **Generate Labels for VOC: 다크넷의 라벨 형식 (직접 레이블링 시, 그대로 적용 가능)**
  - **.xml to .txt** `<object-class> <x> <y> <width> <height>`
  - **x, y, width, and height are relative to the image's width and height**

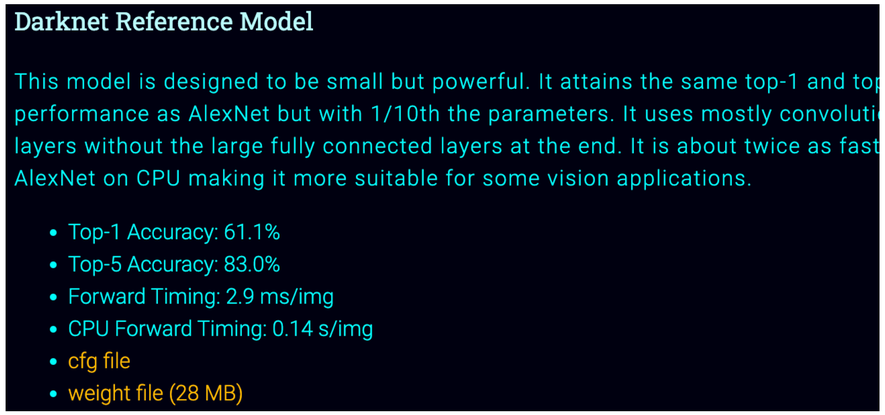
```
sets=[('2012', 'train'), ('2012', 'val'), ('2007', 'train'), ('2007', 'val'), ('2007', 'test')]

classes = ["aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow"


def convert(size, box):
    dw = 1./size[0]
    dh = 1./size[1]
    x = (box[0] + box[1])/2.0
    y = (box[2] + box[3])/2.0
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)
```

Before Train

Detection Using A Pre-Trained Model

- **Pre-Trained Model = pre-trained weight** (정해진 weight 사용 + 학습시 변환..?)
- **YOLO Model Comparison (둘 다 weight file 존재 = trained on 2007 train/val+ 2012 train/val)**
  - **yolo.cfg: extraction network 기반**
    - **yolo.cfg = extraction.cfg + 4 conv layer + 1 fc layer**
  - **tiny-yolo.cfg: much smaller than yolo.cfg & Darknet reference network 기반**
    - ***.cfg:*** 네트워크 구조와 하이퍼파라미터 결정 *(*자신의 데이터에 맞게 조정 필요*)*
    - ***weight file: pre-trained weight***





Train

Detection Using A Pre-Trained Model

- **Crop Layer: 448 x 448 -> 448 x 448 x 3 image**
  - **Crop Layer 이란 ?**
    - 이미지 잘라내기 위한 단계 (일정 이상 크기의 이미지 생성 막음)
- **test's output: confidence & how long it took to find them**

```
./darknet yolo test cfg/yolov1/yolo.cfg yolov1.weights data/dog.jpg
0: Crop Layer: 448 x 448 -> 448 x 448 x 3 image
1: Convolutional Layer: 448 x 448 x 3 image, 64 filters -> 224 x 224 x 64 image
....
27: Connected Layer: 4096 inputs, 1225 outputs
28: Detection Layer
Loading weights from yolov1.weights...Done!
data/dog.jpg: Predicted in 8.012962 seconds.
0.941620 car
0.397087 bicycle
0.220952 dog
Not compiled with OpenCV, saving to predictions.png instead
```

```
1  [net]
2  # Testing
3  batch=1
4  subdivisions=1
5  # Training
6  # batch=64
7  # subdivisions=8
8  height=448
9  width=448
10 channels=3
11 momentum=0.9
12 decay=0.0005
13 saturation=1.5
14 exposure=1.5
15 hue=.1
16
17 learning_rate=0.0005
18 policy=steps
19 steps=200,400,600,20000,30000
20 scales=2.5,2,2,.1,.1
21 max_batches = 40000
22
23 [convolutional]
24 batch_normalize=1
25 filters=64
26 size=7
27 stride=2
28 pad=1
29 activation=leaky
30
31 [maxpool]
```

yolov1.cfg

Train (+option)

Changing The Detection Threshold

- **default: confidence threshold >= 0.2**
- **if confidence threshold == 0,**

- **threshold 설정**

```
./darknet yolo test cfg/yolov1/yolo.cfg yolov1.weights data/dog.jpg -thresh 0
```

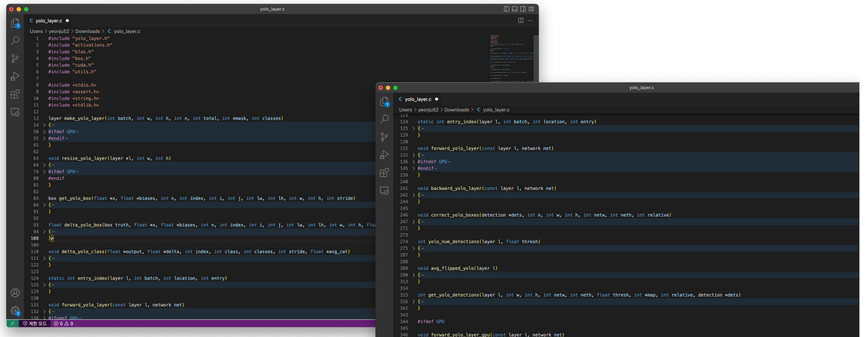
or (추정)

```
247  [detection]
248  classes=20
249  coords=4
250  rescore=1
251  side=7
252  num=3
253  softmax=0
254  sqrt=1
255  jitter=.2
256
257  object_scale=1
258  noobject_scale=.5
259  class_scale=1
260  coord_scale=5
```

yolov1.cfg

다음주 예정

- 서버컴 - 가상환경 세팅 후 코드 실행
- 코드 상세 리뷰 (논문 비교)
  - (예시1) 코드 내 loss 함수 -> 논문과 비교
  - (예시2) yolo_layer.c 함수 기능 분석



(확대) yolo_layer.c: train/valid/test와 관련 함수 (추정)