AI 팀
인턴 실습

SSG.COM
조승진

인턴 과제 정의

Machine Learning Model Training & Inference

- 관심 분야 선정
  - Image – Object Detection
- 모델 선정
  - YOLO(You Only Look Once)
- 모델 학습 진행
- 결과 발표

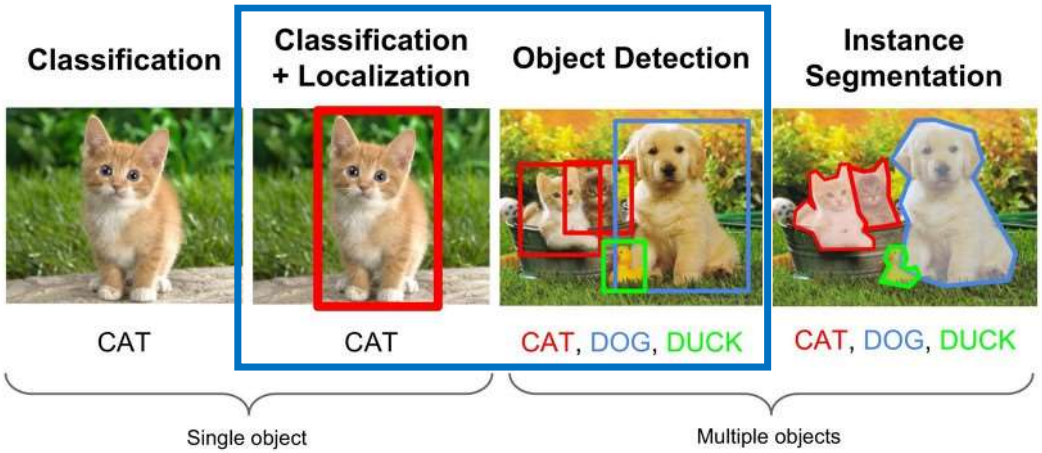
Contents

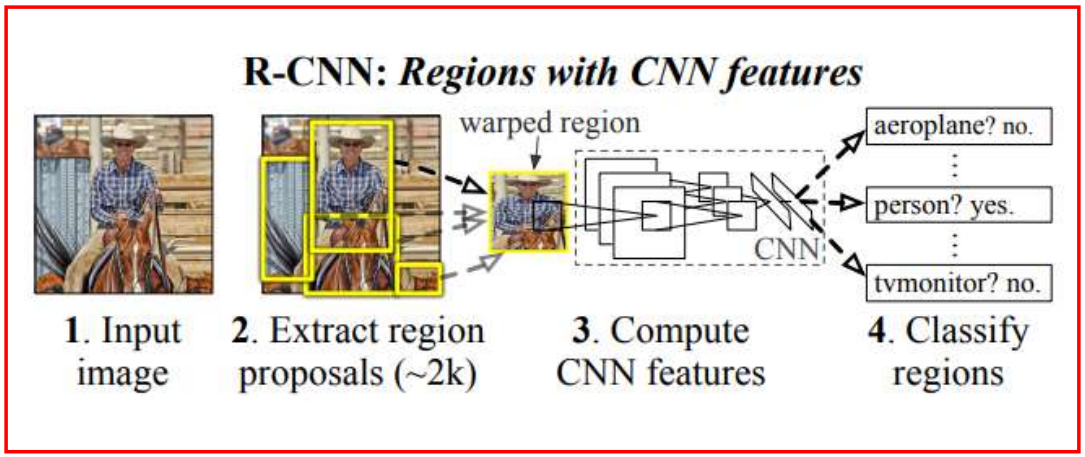
Contents

01 YOLO

- Object Detection
- YOLO v1
- YOLO v2
- YOLO v3

Object Detection





- **Multi-label Classification + Localization**
  - 클래스의 분류
  - 오브젝트 위치 찾기
  - 두가지 문제가 혼합되어 있는 문제

- **2-stage Detector**
  - Region Proposal (후보 박스 선정) + Classification 두 단계
  - 높은 검출 정확도
  - Ex) RCNN (R-CNN, Fast R-CNN, Mask R-CNN...)

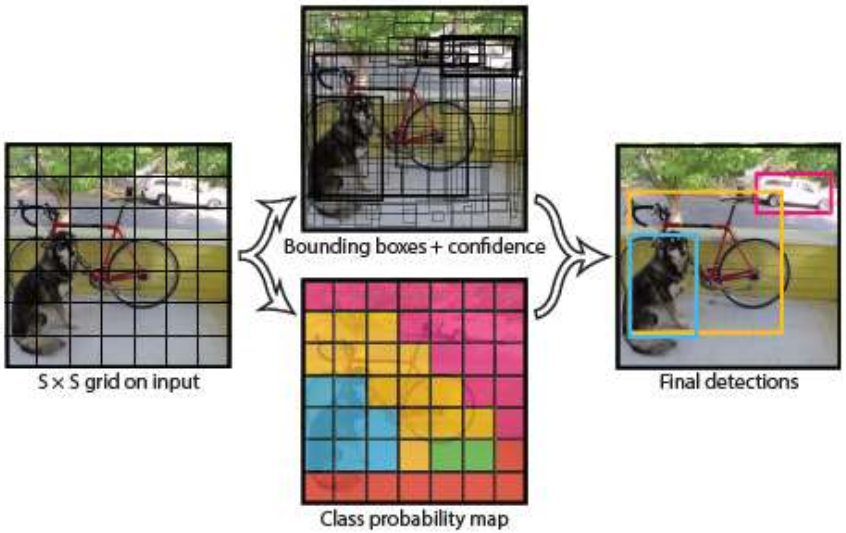
- **1-stage Detector**
  - Single network로 해당 문제를 한 단계로 수행
  - 상대적으로 빠른 검출 속도. 실시간 Object Detection에 유리.
  - Ex) YOLO, SSD ..

YOLO v1

YOLO(You Only Look Once)

사람이 사물을 판단하듯

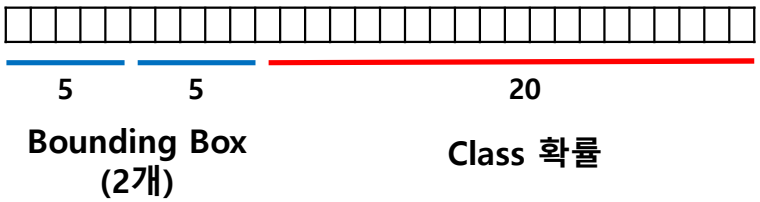
물체를 파악하고, 그 종류를 동시에 파악



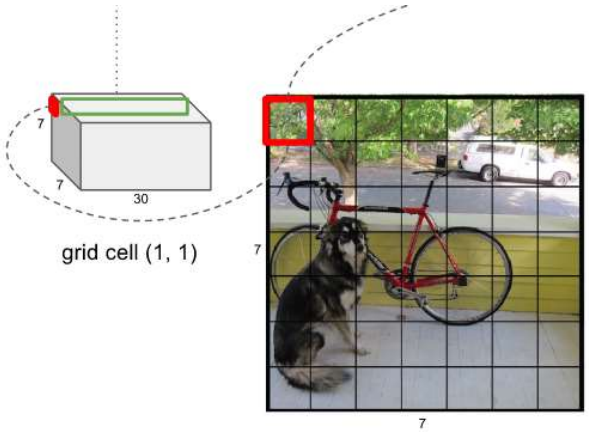
- **Unified Detection**
  - 이미지를 **SxS Grid로 분할**
  - 각 Grid Cell
    - **Bounding Box 정보** (X, Y, W, H)
    - **Confidence Score**
      - 박스내에 객체가 있을 확률 * 실제 객체와의 IOU

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

    - **각 Class의 확률(Softmax)**

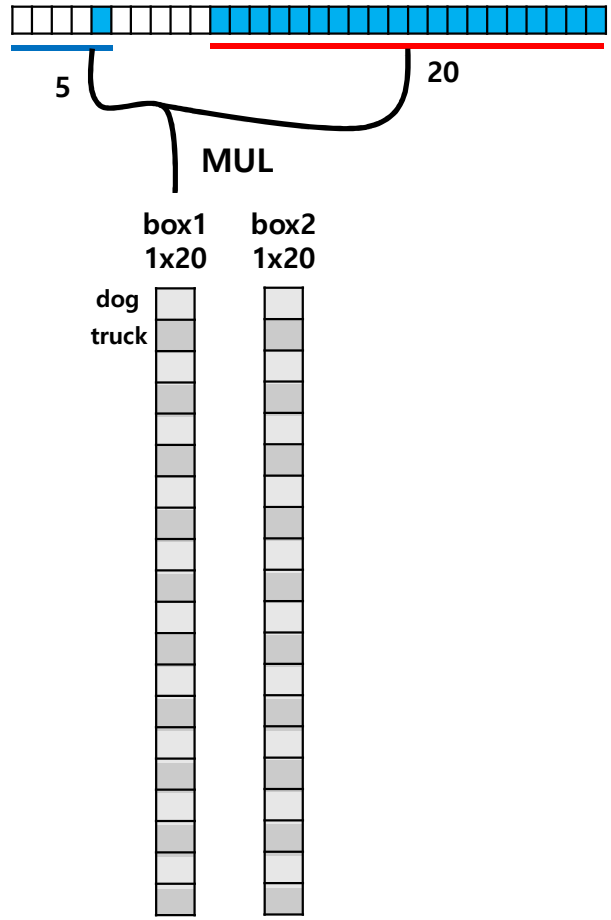


YOLO v1



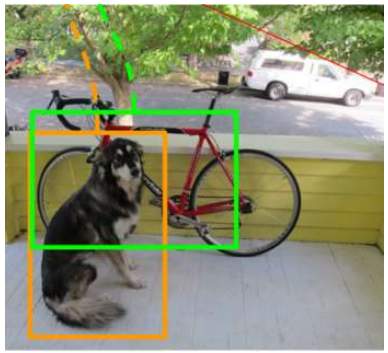
YOLO v1 Bounding Box

7x7x2=98개의 박스 검출

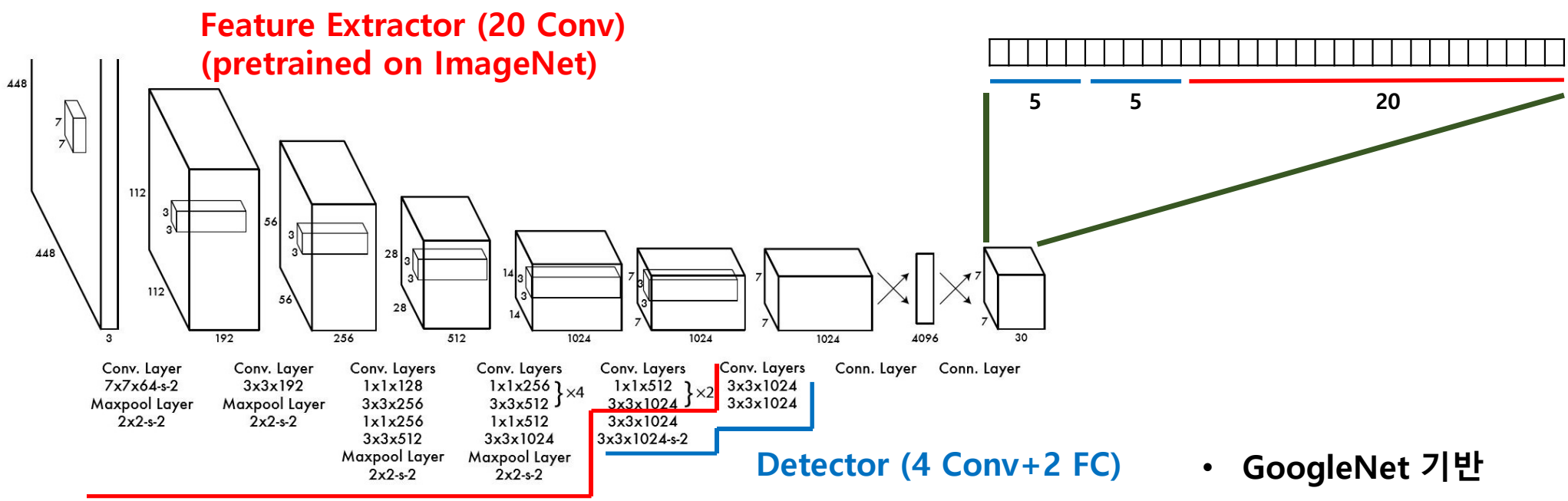


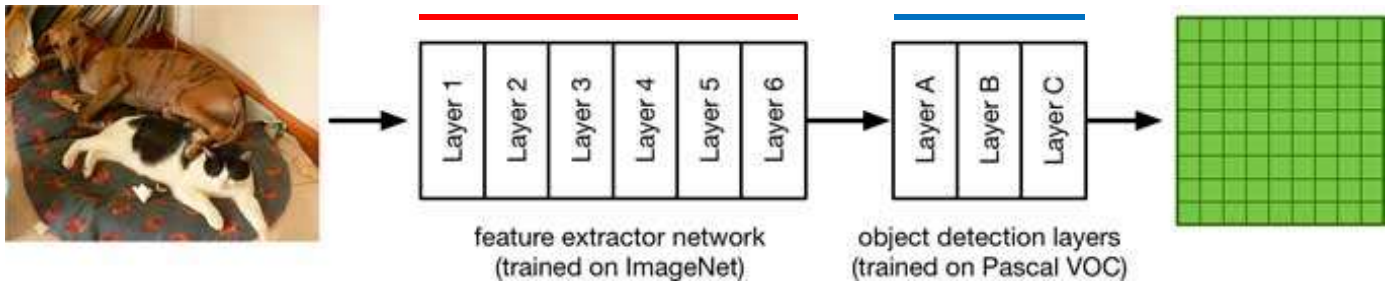
중복 박스 제거

1. Treshold
2. NMS(Non-Maximal Suppresion)



YOLO v1





- **GoogleNet 기반**
- **1x1, 3x3 Conv 위주**

(계산량 줄이기, 층을 더 깊게)

YOLO v1

- **LOSS** $\mathbb{1}_{ij}^{\text{obj}}$ 1) Object의 Center가 속해있는 i번째 cell
  2) Cell의 Bounding box j

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

Localization Loss

Box의 x, y, w, h를 regression

λ_{coord} 다른 loss와의 balancing을 위한 parameter

X_{ij}가 아닌, X_i 에 대한 Loss만을 구한다.
아주 작은 물체의 중심이 grid cell 내부에 2개가 존재한다면?

Confidence Loss

Box에 Object가 존재할 확률(Sigmoid).

Ground Truth의 경우 존재하면 1, 없으면 0

λ_{noobj} Object가 존재하지 않는 셀이 상대적으로 더 많음
0.5 등으로 설정하여 Balancing

Classification Loss

Class별 Softmax 값.

Ground Truth의 경우 1, 틀리면 0

YOLO v2

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

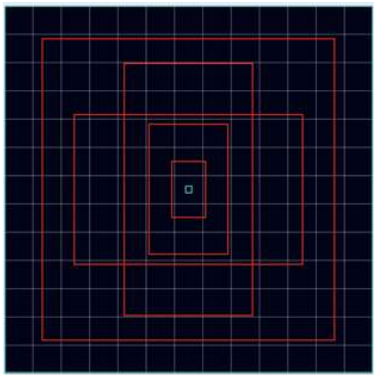
Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

- **Network를 깊게 만들기보단 간소화**
  - **Darknet-19**
  - 빠른 속도의 장점을 유지
  - **Factorization**
    - 3x3 layer를 주로 사용
  - FCL를 Conv로 대체

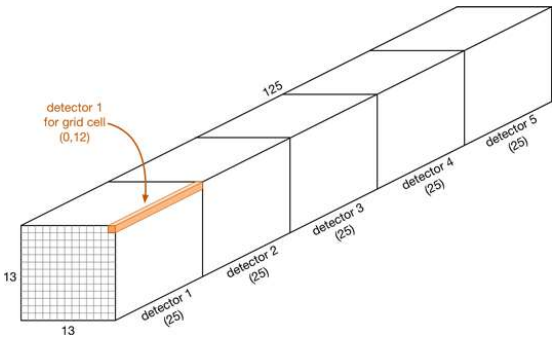
- **YOLO v2 개선점**
  - **Batch Normalization**
    - 각 Conv Layer에 BN Layer추가
    - Internal Covariant Shift 방지
  - **High Resolution Classifier**
    - **Pre-Train on ImageNet**

      224x224 -> 448x448
  - **Fine-Grained Features**
    - 7x7 Grid -> 13x13 Grid
    - **Passthrough Layer**
      - 26x26의 High resolution Feature Concat
      - 13x13x1024 + 13x13x2048 (26x26x512)
      - 다양한 크기의 Object 검출 가능
  - **Multi-Scale Training**
    - FCL -> Conv : 다양한 input size로 학습 가능
    - {320, 352, ..., 608} size 중 임의로 학습
    - 다양한 크기의 image에 대응 가능

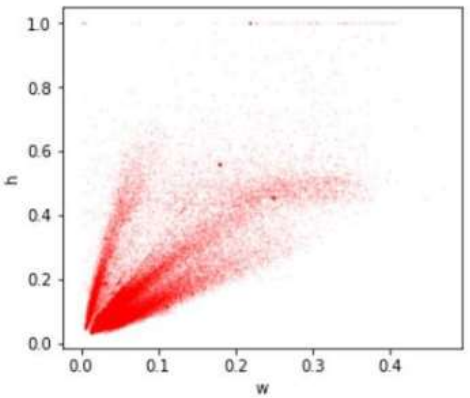
YOLO v2



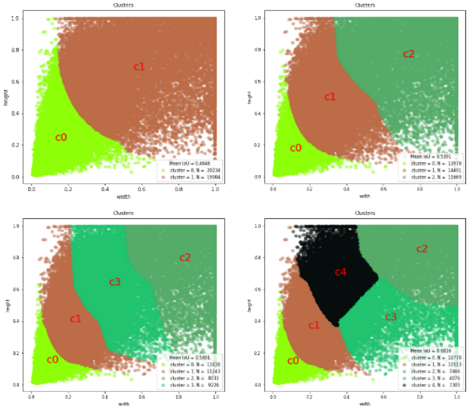
Anchor Box



(x, y, w, h, conf, n class)*5



Dataset 이미지 크기 분포

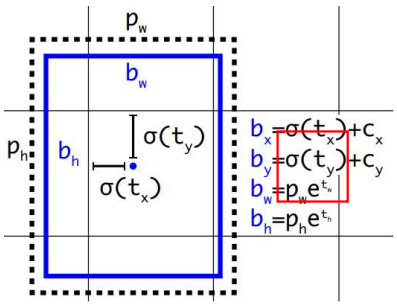


K-means 알고리즘 적용

- **Anchor Box**
  - 사전 정의된 Anchor 활용 Prediction
  - Anchor로부터의 Object 크기로의 Offset 예측
  - 계산량이 늘어나지만, FCL을 Conv로 대체하여 조정
  - **13x13x5 = 845개의 박스 검출**
- **Demension Cluster**
  - Handpick된 Anchor가 아닌, 데이터에 근거 Anchor 선정
  - **K-means clustering**
- **Direct location prediction**

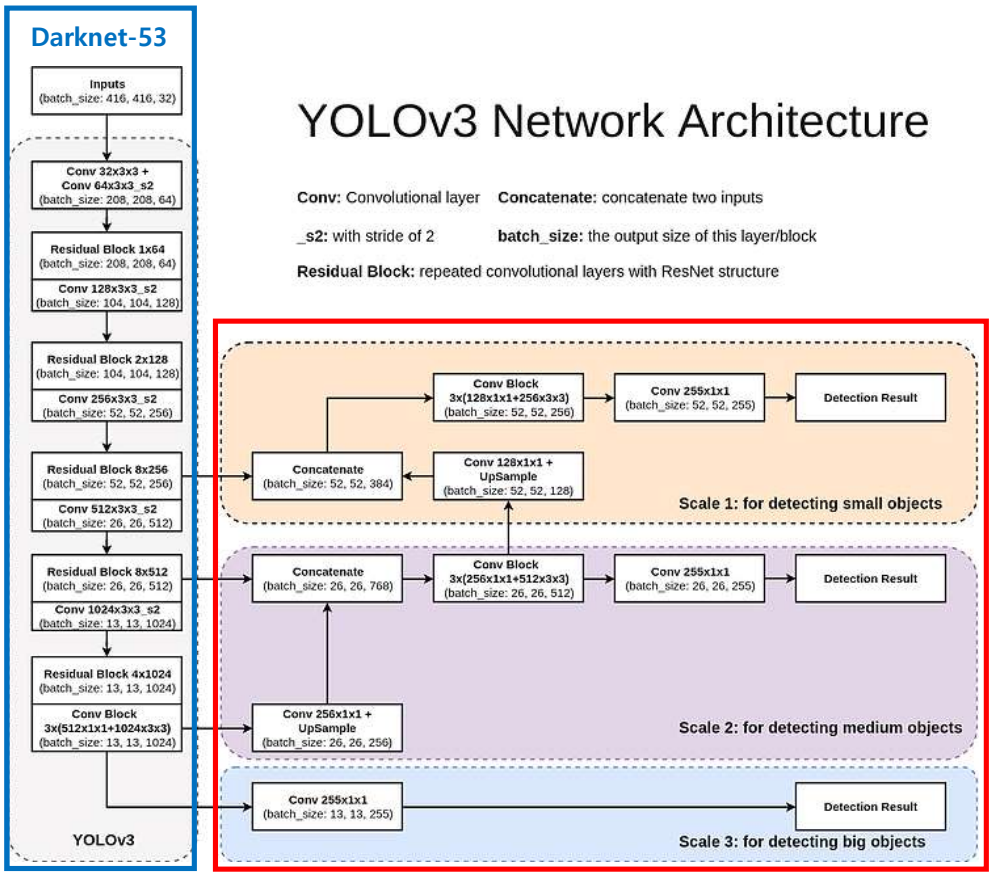
$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

일반적인 방식
Grid Cell을 벗어나 제한없이 이동



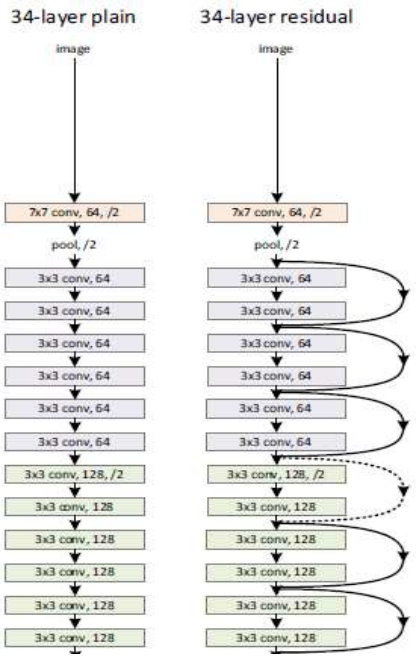
Sigmoid로 0~1로 제한
각 Grid cell을 벗어나지 않음

YOLO v3

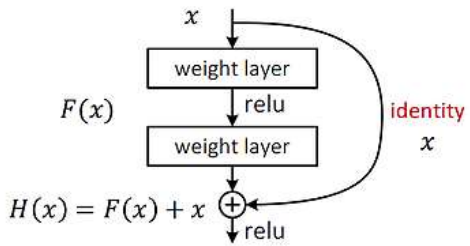


- **Residual Block**
  - Depth가 깊어질 수록, 고차원의 feature
  - 층이 깊어진 만큼, Vanishing or Exploding gradients

    -> **Training Error 증가 (Degradation)**



- Residual net

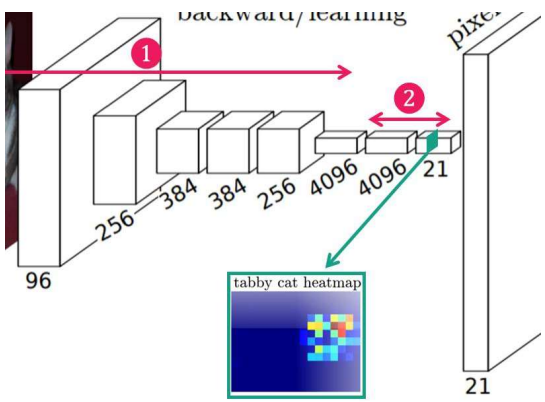


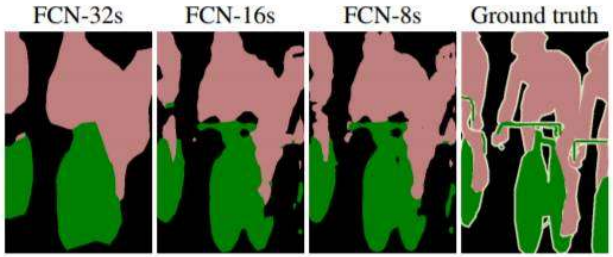
- H(x) 전체를 학습하는 것 보다,

H(x)=F(x)+x의 형태로 F(x)를 0이 되도록 학습

F(x) Block단위로 학습시키는 것이 최적화에 유리

YOLO v3





- **Predictions Across Scales**
  - FPN(Featured Pyramid Network)와 유사
  - **High-resolution map + Upsample Low-resolution map**
  - Low-resolution map의 **Semantic information**
  - High-resolution map의 **Fine-grained information**
  - 작은 Object와 다양한 scale의 Object 검출
  - 13x13
  - 26x26
  - 52x52
  - 3가지 Scale로 Prediction
  - 각 Scale 별 3개의 Anchor
  - **(13x13x3)+(26x26x3)+(52x52x3) = 10647개의 박스 검출**
- **Class Prediction**
  - Softmax -> Logistic
  - Ex) 사람 & 여자 Class가 있는 Dataset 문제
  - Classification Loss 를 SSE -> Binary Cross entropy

Contents

02 Training

- 학습 환경
- 학습 결과

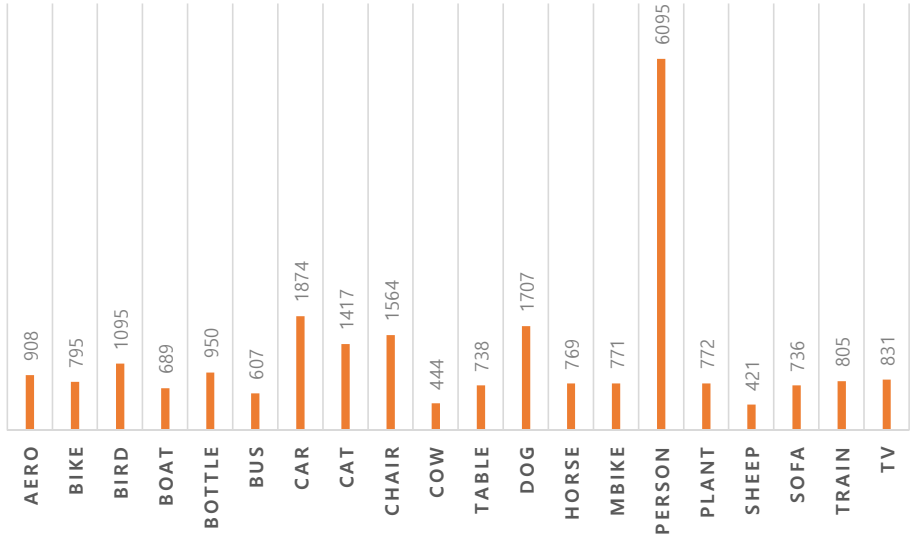
학습 환경

- **Environment**
  - GPU : Quadro P5000
  - Git repo : https://github.com/wizyoung/YOLOv3_TensorFlow
  - Requirement
    - Python version: 2 or 3
    - Packages:

      tensorflow >= 1.8.0

      opencv-python

      tqdm



- **Dataset**
  - Pascal VOC

    -> Train Set (16,551개) : Pascal VOC2012 (train,val) + Pascal VOC2007 (train,val)

    -> Test Set (4,952개) : Pascal VOC2007 test set

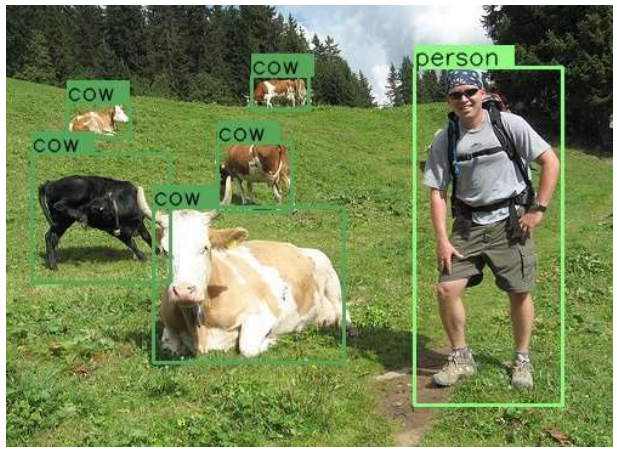
    (Pascal VOC Dataset 2012는 test set에 대한 정답이 공개되어 있지 않기 때문에 구분)
  - 20가지 Class

(aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor)

학습 결과

시연
ssh -L 16006:127.0.0.1:6005 scom@10.203.7.143
http://localhost:16006/

- **Hyperparameter**
  - Image size : 416x416
  - Batchsize : 6
  - nms_threshold : 0.45
  - score_threshold : 0.3



- **Result**
  - **Best mAP**
    - mAP : 76.84 % / Recall : 0.868 / Precision : 0.643

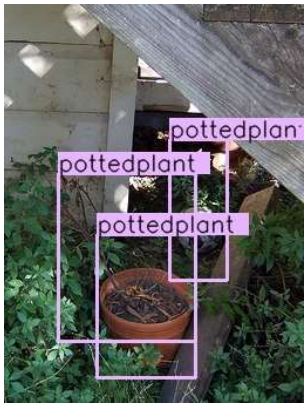
aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
0.8826	0.8575	0.7601	0.6669	0.6711	0.8663	0.9027	0.8581	0.6092	0.7600	0.6939	0.8416	0.8776	0.8072	0.8510	0.4443	0.6961	0.7189	0.8495	0.7543

학습 결과

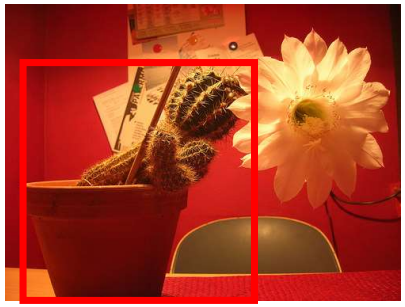
- **Pottedplant**

aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
0.8826	0.8575	0.7601	0.6669	0.6711	0.8663	0.9027	0.8581	0.6092	0.7600	0.6939	0.8416	0.8776	0.8072	0.8510	0.4443	0.6961	0.7189	0.8495	0.7543

- **Result**



- **Train Data**



Contents

03 Conclusion

Conclusion

- **YOLO**
  - Unified Detection이라는 컨셉으로 1-stage detection
  - 빠른 속도 유지에 중점
  - 작은 Object를 검출하기 위한 개선

    Box (98개 -> 845개 -> 10647개)

- **활용가능성**
  - 매장 동선 정보, 행동 분석, 상품 관리 등

- **추가 공부할 부분**
  - GoogleNet (Inception Module)
  - SSD
  - 2-stage detection(F-RCN)
  - ResNet
  - FPN
  - …

- **결과 고찰**

Contents

Appendix

Non-Maximal Suppression