# YOLO9000:
# Better, Faster, Stronger

박태우

# Introduction

일반적으로 object detection 데이터 셋은 많이 부족함.

그에 반해 classification 데이터 셋은 상당히 많음.

Classification 데이터 셋을 이용해서 object detection을 학습시키겠다.

방법으로

1. Hierarchical view of object classification
2. Joint training algorithm

두가지를 제안함.

# Better

## Batch Normalization

- 모든 convolutional layer에 batch normalization을 추가시킴으로써 2%의 성능 향상을 시킴.
- Overfitting없이 dropout을 제거할 수 있었음.

## High Resolution Classifier

- 보통 classifier를 ImageNet 기반 pre-trained을 많이 사용함. (224 x 224 네트워크)
- 기존의 YOLO는 detection을 위해 448 x 448 네트워크로 다시 바꿔주는 과정을 사용함.
- 본 논문에서는 448 x 448의 classifier network를 fine tuning을 함으로써 4% 성능 향상을 시킴.

# Better

## Convolutional With Anchor Boxes

- Input image를 448x448에서 416으로 축소시킴. ( Output을 홀수로 만들기 위해 )
- 7x7 grid cell을 사용하던 YOLO와 달리 13x13으로 확장.
- FC를 삭제하고 Anchor box를 사용함.

  (Anchor box의 용도는 초기 학습의 가이드라인으로써 학습의 효율성을 증가시킨다.)
- Anchor box를 사용함으로써 mAP는 0.3% 하락했지만 recall의 경우 7% 증가함.

# Better

## Dimension Clusters

- 어떤 anchor box를 사용할까에 대한 방법으로써 k-means clustering을 사용함.

  ( k가 5일 때 가장 좋은 결과를 얻을 수 있었음 )

- k-means에 Euclidean distance를 사용한다면 박스 크기가 큰 경우 에러가 증가함.

- 크기와 관계없이 IOU를 높게하기 위해 다음과 같은 metric을 사용함.

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

# Better

## Direct Location Prediction.

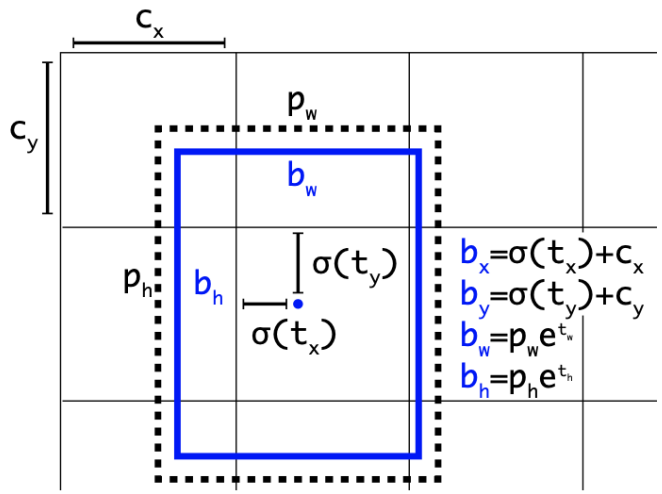
- 초기 anchor box를 이용한 학습에 있어 x, y 좌표는 다음과 같이 계산됨.

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

- 이 부분에 있어 tx는 어떠한 값을 가질 수 있으므로 이미지 내 엉뚱한 위치로 갈 수 있음.

- 그래서 기존 v1 방식을 사용하고 오른쪽 그림과 같이 해당 grid cell 내부에서만 움직일 수 있도록 제한을 함. ( 0~1 사이의 값을 갖도록 함 )

- 그냥 anchor box를 사용하는 것 대비 5% 성능 향상을 얻음.



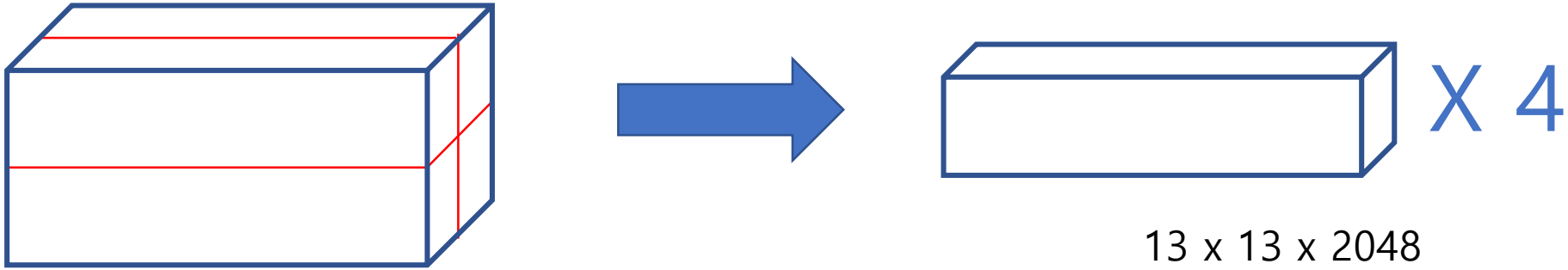
# Better

## Fine-Grained Features

- 기존의 YOLO는 grid cell 방식을 이용하기 때문에 작은 물체에 성능이 낮았음.

- 이를 해결하기 위해 SSD와 비슷한 방식으로 pooling 하기 전 네트워크를 사용.

- Passthrough layer를 이용, 다음과 같이 잘라 최종 네트워크에 붙여줌.




## Multi-Scale Training

- FC를 제거함으로써 사이즈를 조절할 수 있게됨.

- 10 batch마다 416 x 416 뿐만 아니라 다양한 크기에 대해 학습함.

# Faster

## Darknet-19

- Darknet-19라는 새로운 classification model을 만들어 사용.

## Training for classification

- 초기 학습을 224x224에서 448x448로  10 epoch fine tuning 함.
- ImageNet 1000개 class에 대해 학습.

## Training for detection

- 한 grid cell 마다 125개 값을 가짐 ( 5개 bbox X (bbox 정보 5개 + class 20개 ))
- Passthrough layer로부터 얻은 예측 값도 합침.

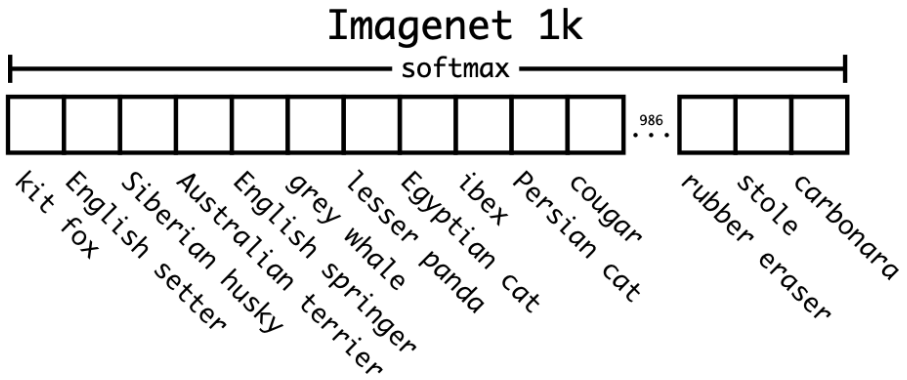
# Stronger

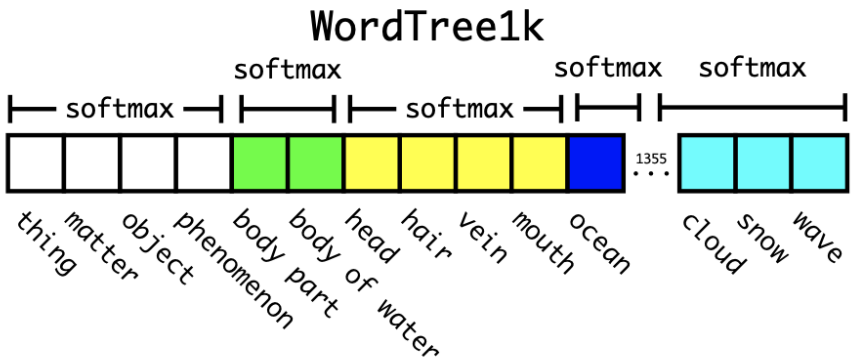
## Hierarchical classification

- WordNet을 이용해 ImageNet 데이터를 hierarchical tree화 시킴.
- 같은 level에 있는 class들끼리 softmax를 구하고 상윗단으로 갈수록 condition probability 적용.
- 최종적으로 구한 condition probability는 정답과 cross entropy를 사용하여 학습.

$$Pr(\text{Norfolk terrier}|\text{terrier})$$
$$Pr(\text{Yorkshire terrier}|\text{terrier})$$
$$Pr(\text{Bedlington terrier}|\text{terrier})$$

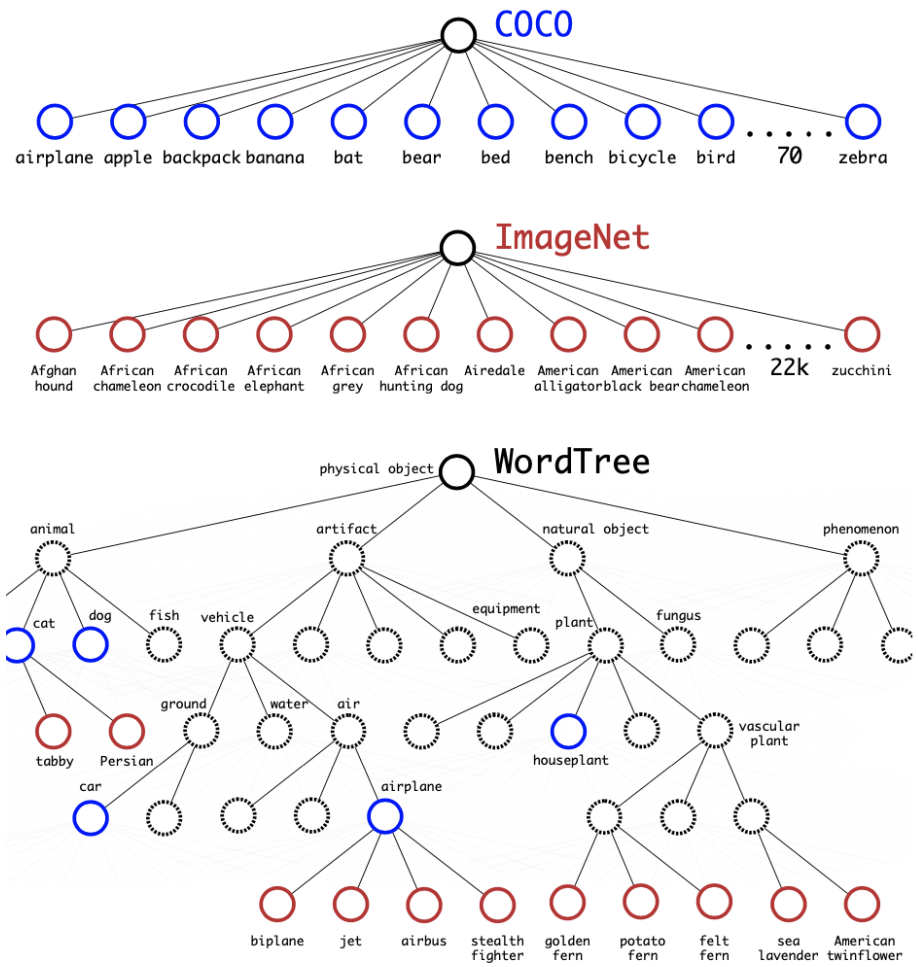
$$Pr(\text{Norfolk terrier}) = Pr(\text{Norfolk terrier}|\text{terrier})$$
$$*Pr(\text{terrier}|\text{hunting dog})$$
$$*\ldots*$$
$$*Pr(\text{mammal}|Pr(\text{animal})$$
$$*Pr(\text{animal}|\text{physical object})$$



# Dataset combination with WordTree

- ImageNet 데이터와 COCO 데이터 셋을 합쳐 WordTree 구조 사용.



## Joint Classification and Detection

- 계산량 등의 이유로 학습할 때는 3개의 anchor box를 사용.

- Detection 데이터와 classification 데이터가 같이 들어오는데 classification 데이터가 들어왔을 때는 classification loss만 backpropagation 한다.

- 그리고 hierarchical tree 구조에서 예측한 값 기준으로 상위 계층만 backpropagation 진행함.

# 정리

## 특징

- 본 논문은 YOLO v2, 9000에 대한 내용을 포함하고 있음.
- 기존 YOLO의 grid cell 기반 detection을 유지한 채 성능을 높임.
- Classification, detection 데이터를 사용하면서 학습을 시킴.
- Hierarchical classification을 이용해 수 많은 class들을 예측함.