

Yolo v1, v2

Object Detection

2023-02-03

Object Detection

이미지 및 비디오 내에서 유의미한 특정 객체를 감지하는 작업

1) Classification

2) Classification
+ Localization

3) Object Detection

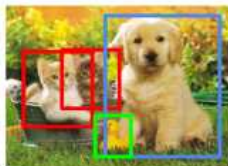
4) Instance
Segmentation



CAT



CAT



CAT, DOG, DUCK



CAT, DOG, DUCK

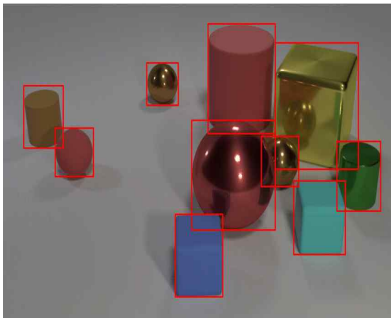
Pixel단위 위치 & 무엇인지

Single object

Multiple objects


Object Detection 관련 용어

1. Bounding Box

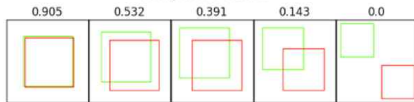


2. IOU(Intersection Over Union)

실제 값과 모델이 예측한 값이 얼마나 겹치는지
(교집합 / 합집합)

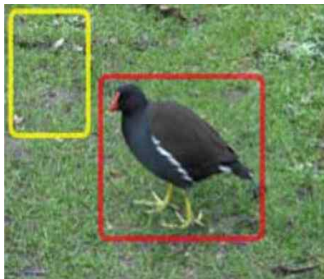
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Sample IoU scores



Object Detection 관련 용어

3. Confidence Score



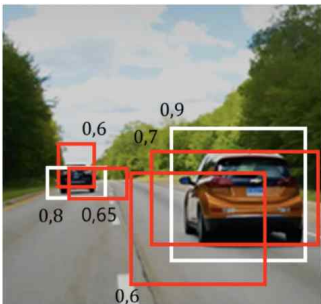
- 물체가 있을 확률
- 물체가 있을 확률 * IOU
- 어떤 물체의 Class일 확률 * IOU

4. NMS(Non-Maximum Suppreslon)



박스의 중복을 제거 및 물체를 가장 잘 나타낸 박스를 남김

4. NMS(Non-Maximum Suppreslon)



1. 특정 **Confidence Score**이하의 **Bounding Box** 제거

2. 남은 **Bounding Box**들을 **Confideence Score**기준으로 내림차순
[0.9, 0.8 ,0.7 ,0.65 ,0.6 ,0.6]

3. 맨 앞 박스부터 기준으로, 이 박스와 **IOU**가 특정 **Threshold** 이상인 박스들은 제거

그 후 2, 3과정을 반복

5. Precision(정밀도), Recall(재현율)

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP	FN
	NEGATIVE (0)	FP	TN

TP : 실제 결과 : True, 모델의 예측 : True

-> 물체가 있는 위치와 Class를 모델이 잘 찾은 경우이다

FP : 실제 결과 : False, 모델의 예측 : True

-> 물체가 있다고 예상했지만, 실제로는 물체가 없는 경우

FN : 실제 결과 : True, 모델의 예측 : False

-> 실제 존재하는 물체를 탐지하지 못한 경우

TN : 실제 결과 : False, 모델의 예측 : False

-> 실제로 그 자리에 물체가 없었고 모델도 그 자리에 Bounding Box를 예측하지 않은 경우이다

Object Detection 관련 용어

5. Precision(정밀도)

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP	FN
	NEGATIVE (0)	FP	TN

맞을 것이라고 예상 한 것중 옳게 검출한 비율

$$\text{정밀도} = \frac{TP}{TP + FP}$$

Recall(재현율)

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP	FN
	NEGATIVE (0)	FP	TN

검출해내야 하는 물체들 중 제대로 검출된 비율

$$\text{재현율} = \frac{TP}{TP + FN}$$

Object Detection 관련 용어

5. Precision(정밀도), Recall(재현율)

15개의 Object가 검출되어야 하는 이미지에서 아래와 같이

모델이 10개의 객체만 검출했고, 그 때의 Confidence와 TP/FP 여부가 있다고 가정

Detections	confidences	TP or FP
A	57%	TP
B	78%	TP
C	43%	FP
D	85%	TP
E	91%	TP
F	13%	FP
G	45%	TP
H	68%	FP
I	95%	TP
J	81%	TP

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP = 7	FN = 8
	NEGATIVE (0)	FP = 3	TN

Confidence의 threshold = 0

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP = 1	FN = 14
	NEGATIVE (0)	FP = 0	TN

Confidence의 threshold = 95%

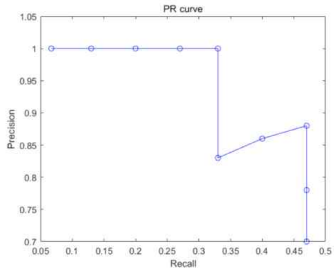
Object Detection 관련 용어

6. AP(Average Precision), mAP(mean Average Precision)

AP : Recall을 0부터 1까지 0.1씩 바꿔가면서 그 때의 Precision을 계산하고 평균낸 값

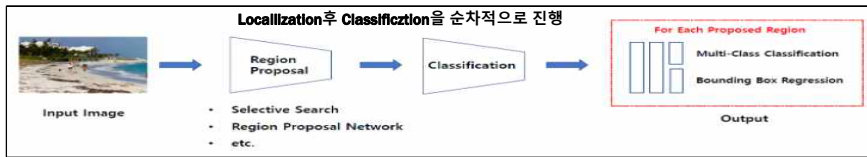
mAP : 각 Class마다 AP를 계산하고 그들을 평균낸 값

Detections	confidences	TP or FP	누적 TP	누적 FP	Precision	Recall
I	95%	TP	1	0	$1/1=1$	$1/15=0.067$
E	91%	TP	2	0	$2/2=1$	$2/15=0.13$
D	85%	TP	3	0	$3/3=1$	$3/15=0.2$
J	81%	TP	4	0	$4/4=1$	$4/15=0.27$
B	78%	TP	5	0	$5/5=1$	$5/15=0.33$
H	68%	FP	5	1	$5/6=0.83$	$5/15=0.33$
A	57%	TP	6	1	$6/7=0.86$	$6/15=0.4$
G	45%	TP	7	1	$7/8=0.88$	$7/15=0.47$
C	43%	FP	7	2	$7/9=0.78$	$7/15=0.47$
F	13%	FP	7	3	$7/10=0.7$	$7/15=0.47$

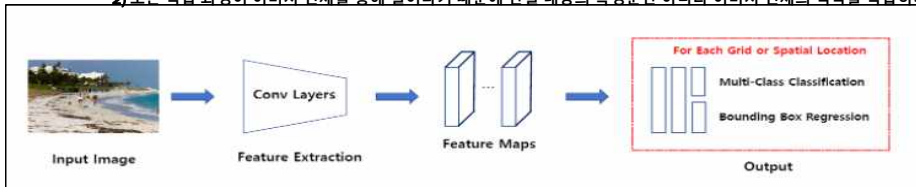


Yolo v1: You Only Look Once

2-Stage Detector - 정확도는 높지만, 학습 및 예측, 최적화속도가 느리다



1-Stage Detector 1) 파이프라인이 간단하기 때문에 학습과 예측속도가 빠르다.
2) 모든 학습 과정이 이미지 전체를 통해 일어나기 때문에 단일 대상의 특징뿐만 아니라 이미지 전체의 맥락을 학습하게 된다.



Yolo v1: You Only Look Once

Detection 과정

1. Input Image를 $S \times S$ 의 Grid로 이미지를 나눠준다

2. 이후 각 Grid Cell에 대해 두 가지 Task 를 동시 진행

① Bounding Box

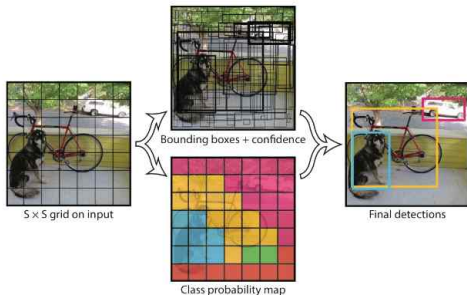
- 각 grid cell은 B개의 Bounding BOX를 가진다
- 박스의 중심 x, y 와 w, h & Confidence score를 가진다

$$confidence = Pr(Object) * IoU(pred, true)$$

② Classification

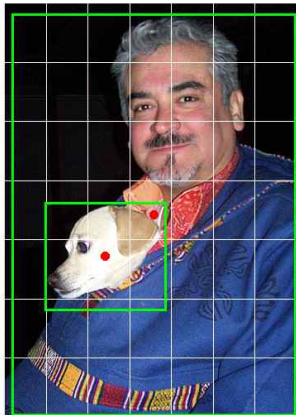
➡ Output Shape = $S * S (B * 5 + C) = 7 * 7 * 30$

3. NMS를 거쳐 겹치는 Box들을 제거 후 최종 결과 도출



Yolo v1: You Only Look Once

① Bounding box - Coordination



Label에서 사용한 이미지를 7x7로 Grid를 나누고

각 Object에 대해 Bounding Box와 Center point를 표시하면,
아래 그림과 같이 표시할 수 있습니다.

```
array([[0., 0., 0., 0., 0., 0., 0.],  
       [0., 0., 0., 0., 0., 0., 0.],  
       [0., 0., 0., 0., 0., 0., 0.],  
       [0., 0., 0., 1., 0., 0., 0.],  
       [0., 0., 1., 0., 0., 0., 0.],  
       [0., 0., 0., 0., 0., 0., 0.],  
       [0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Dog (5,3) Person(4,4)

Yolo v1: You Only Look Once

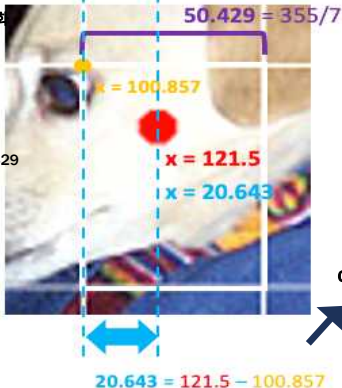
① Bounding box - Coordination(2)

- Goal : grid cell이 전체 이미지, 그 때의 center point를 정규화

x,y는 grid cell에서 어디에 위치하는지 (0~1)
w,h는 grid cell의 w와h 대비 몇배에 해당하는지

Bounding box의 width = 147, grid cell의 width = 50.429
 $w = 147 / 50.429 = 2.915$

dog의 Center point가 포함된 grid cell을 확대



0.409(=20.643 / 50.429)

grid cell내 에서의 좌표로 정규화

Dog의 x, y, w, h 값

[0.34419263456090654, 0.611, 0.4165305949008499, 0.262]

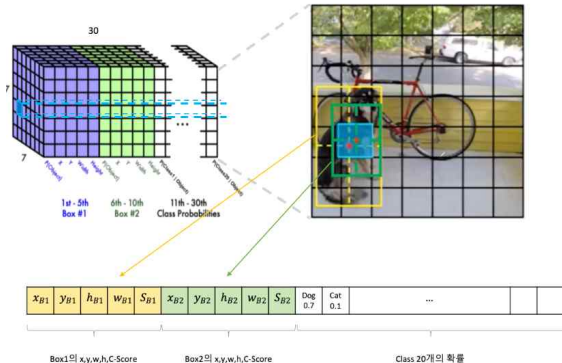


[0.40934837, 0.277, 2.9150143, 1.834]

Yolo v1: You Only Look Once

② Classification

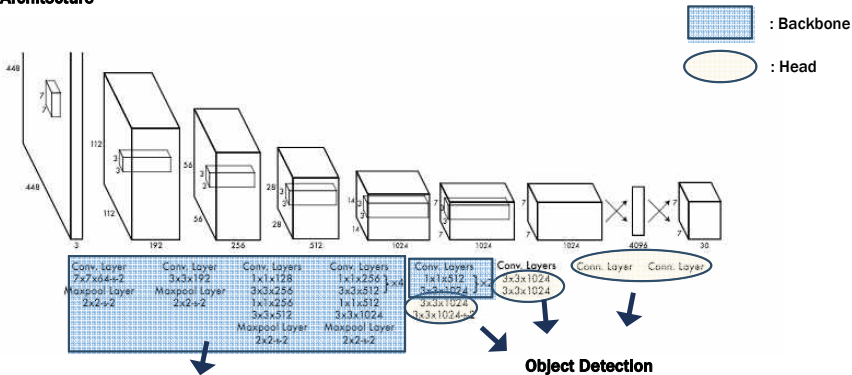
각 grid cell에 대해 수행 C개의 Class에 대해 Class Probabilities 예측 $Pr(class_i|object)$



$$\text{Shape} = S * S * C$$

Yolo v1: You Only Look Once

Network Architecture



Feature Extract

특징을 추출하는 것이 목적이기에
Classificatio 목적으로 만들어진 모델 사용

ImageNet 데이터로 Pre-train 된 모델을 가져와 fine tuning

Yolo v1 - Loss

Grid 별 두 개 Box 중 Ground Truth와 IOU가 더 큰 한 개 Box에 대해서만 Loss를 계산하고 학습

문제 1) 2-stage Detector와 다르게 Localization Error와 Classification Error를 동일하게 가중치준다

문제 2) 객체를 포함하고 있지 않은 grid cell은 confidence 값이 0을 갖는다

=> 모형이 불안정해짐

$$\lambda_{\text{coord}} = 5 \text{ and } \lambda_{\text{noobj}} = .5.$$

Localization loss

grid cell i 의 Bounding box predictor j 에 대해 x 와 y 의 loss를 계산

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

Confidence loss

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

Classification loss

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

λ : Hyper Parameter $\mathbb{1}_{ij}^{\text{obj}}$: Object가 존재 한다면 $\mathbb{1}_{ij}^{\text{noobj}}$: Object가 존재 하지 않는다면

: Bounding Box 좌표 손실에 대한 파라미터

Yolo v1 - Localization Loss

1. 많은 grid cell은 객체를 포함하지 않음
 2. confidence score가 0이 되어 객체를 포함하는 grid cell의 gradient를 압도하여 모델이 불안정해질 수 있다.
- => coord를 5로 설정 하여 높은

- S^2 : grid cell의 수(=7x7=49)
- B : grid cell별 bounding box의 수(=2)

x_i, y_i, w_i, h_i : ground truth box 값

$\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$: 예측 bounding box 값

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

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



Localization loss

크기가 큰 bounding box의 작은 오류가 크기가 작은 bounding box의 오류보다 덜 중요하다는 것을 반영하기 위해 값에 루트를 씌어주게 됩니다.

Yolo v1 - Confidence Loss

λ_{noobj} : 객체를 포함하지 않는 grid cell에 대한 가중치, 논문에서는 0.5로 설정 coord=5로 설정한것에 비해 상당히 작게 설정하여 객체를 포함하지 않는 grid cell의 영향력을 줄임

C_i : 객체가 포함되어 있을 경우 1, 그렇지 않을 경우 0

\hat{C}_i : 예측한 bounding box의 confidence score

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

Confidence loss

Yolo v1 - Confidence Loss

- $p_i(c)$: 실제 class probabilities
- $\hat{p}_i(c)$: 예측한 class probabilities

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

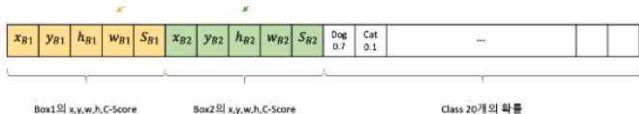
Classification loss

Yolo v1: Experiments

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Yolo v1 한계

1. **Anchor Box**를 사용하지 않고, **Cell** 단위로 **Bounding Box Regressor** 과정을 통해 **Box**를 찾는다
=> **Localization Error**로 인해 성능이 낮아진다.
2. 각 **Grid Cell**에 대해 2개의 **Bounding Box**를 찾지만, **Classification**은 한 개에 대해서만 수행한다.
=> 겹치는 **Object**는 **Detection**하기 어렵다

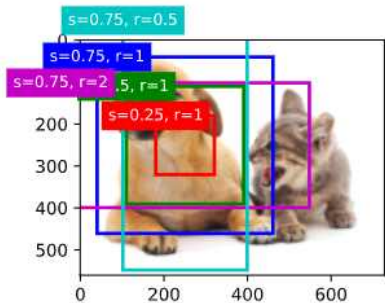


YOLO v1 마지막 Layer

3. 마지막 **Layer Size**가 **7*7**로 매우 작아서 큰 물체는 잘 찾지만, 작은 물체에 대해서는 잘 찾지 못한다.

Anchor Box

이미지에서 다양한 형태의 Object를 Detection하기 위한 미리 정해진 크기와 비율을 가진 **Bounding box**



Localization loss

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

Bounding Box Regressor

Yolo v1에서 **Anchor Box**를 사용하지 않고 **Cell** 단위에서 **Bounding Box Regressor** 과정을 통해 **Box**사이즈를 **Object**에 맞도록 조정

Yolo v2에서 개선된 부분

1. 속도 개선

- Backbone으로 Darknet 19 모델을 사용하여 속도를 빠르게 유지 (FPS 45 -> 40)

2. 모든 Conv Layer 뒤에 Batch Normalization 적용 - mAP 약 2% 증가

Vanishing Gradient 문제 해결, Learning Rate를 키울 수 있기 때문에 더 빨리 수렴
Regularization 역할을 하기에 Dropout 기법 사용하지 않아도 된다는 장점

3. High Resolution Classifier - mAP 약 4% 증가

Yolo v1에서는 이미지 사이즈를 224*224로 Pre-train Classifier 모델을 그대로 사용하고,
실제 입력 받을 때는 448*448 사이즈의 고해상도 이미지를 사용하여 해상도가 맞지 않음

=> Yolo v2에서는 Classifier를 똑같이 224*224로 학습하고, 마지막 10 epoch 정도는 448*448의 고해상도
이미지로 Fine tuning 함으로써 해결

4. Convolutional with Anchor Boxes

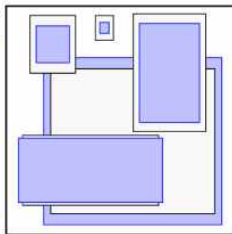
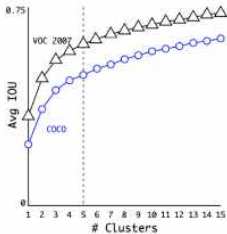
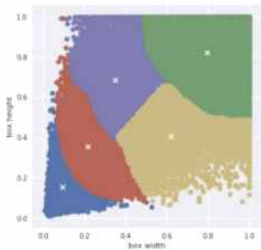
Box는 2개를 예측하고, Classification은 한 번만 수행했던 v1과 달리,
각 Grid Cell에 대해 5개의 Anchor Box를 예측

모든 Anchor Box에 대해 Classification을 수행 즉 Output tensor : 13x13x{(5+C)x5}
=> mAP 69.5 -> 69.2 감소, Recall : 81 -> 88로 상승

Yolo v2에서 개선된 부분

5. Dimension Cluster - Anchor 개수, 크기 탐색

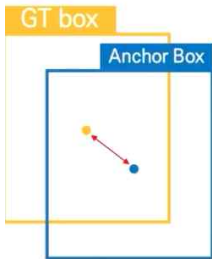
K-Means Clustering



GT들의 width와 height로 K-Means Clustering을 수행하고, 적절한 k개를 탐색
v2 논문에서의 $k = 5 \rightarrow$ Anchor Box 수 = 5
각 Cluster의 Center point = Anchor Box의 사이즈
단, Clustering의 기준을 유클리디안 거리가 아닌 IOU기준으로 한다.

Yolo v2에서 개선된 부분

5. Dimension Cluster(2)

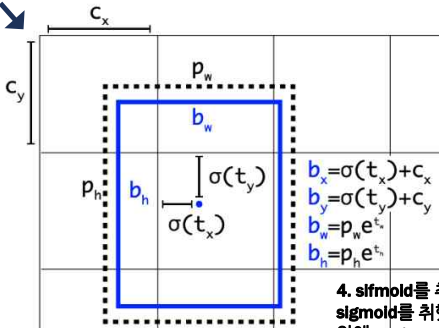


유클리디안 거리를 기준으로 유사도를 측정한다면 실제로는 왼쪽 박스들이 더 유사하지만, 오른쪽 박스들이 더 유사하다고 계산 될 것이기에, 유클리디안 거리가 아닌 **IOU**를 기준으로 **Cluster**를 구한다

Yolo v2에서 개선된 부분 - Direct Location Prediction

Dimension Cluster로 얻은 $(t_x, t_y, t_w, t_h, t_0)$

3. 셀의 시작점



1. : 미리 지정한 Anchor Box

2. : 예측하고자 하는 Box

p_w, p_h : 사전에 정한 Anchor box의 크기

b_x, b_y : ground truth에 가까워지도록 계속해서 학습되는 trained anchor box의 중심 좌표

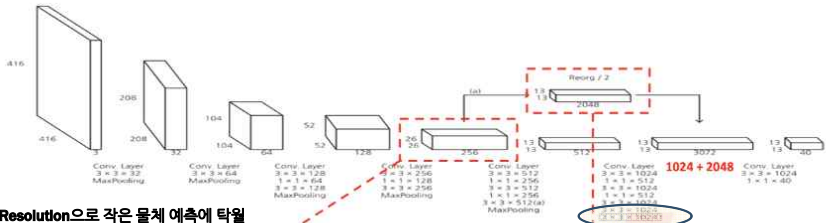
4. **slfmold**를 취한 t_x, t_y 만큼 움직여 center point를 조정 : **sigmold**를 취했으므로 0~1사이만큼만 움직이기에 cell 안에 center point가 항상 존재

- Box가 Cell을 벗어나 아무 위치에나 존재 -> 학습 초기 iteration시 모델이 불안정

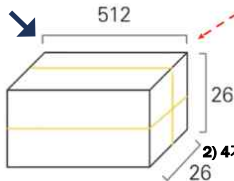
=> **Bounding Box Regression**을 통해 얻은 t_x, t_y 에 **Sigmold** 함수를 적용하여 Box의 중심점이 Cell 내에 존재하도록 한다.

Yolo v2에서 개선된 부분

7. Fine-Grained Features - Feature map이 작을 때 큰 물체는 잘 예측하지만, 작은 물체는 예측하기 어려움

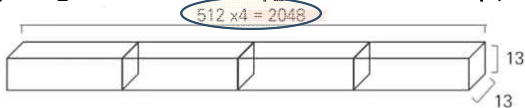


1) High Resolution으로 작은 물체 예측에 탁월



2) 4개로 나누어 Concat

3) Concat한 13 X 13 X 2048 Feature map을 13 x 13 x 1024 feature map과 concat



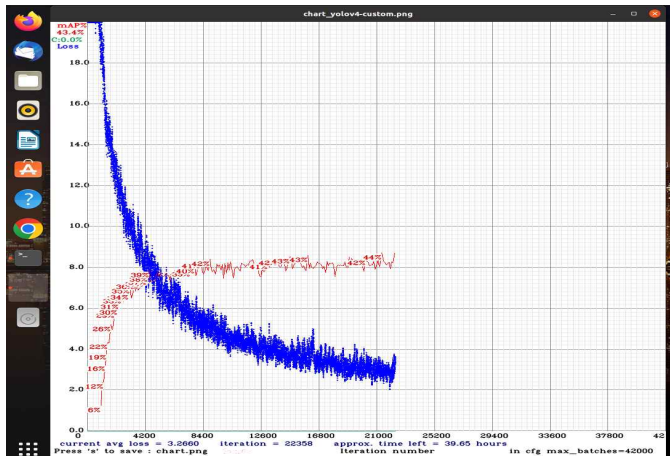
Yolo v2에서 개선된 부분

8. Multi - Scale Training

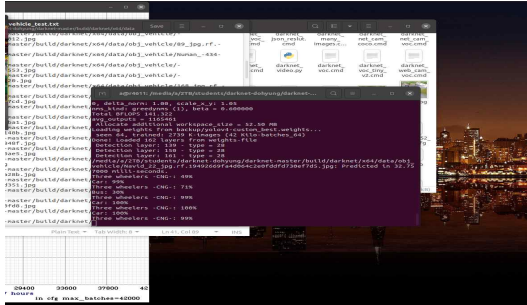
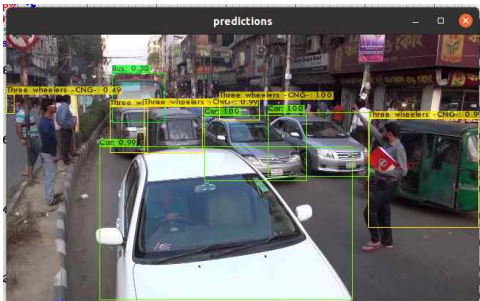
- Anchor Box를 이용해 Bounding Box를 예측하게 되면서 네트워크 구조가 conv and pooling layer로만 구성
- Fully connected layer가 없으므로 입력의 크기를 즉석에서 변경 가능
- 가장 작은 크기는 320x320, 가장 큰 크기는 608 x 608 (subsampling 비율이 32이기 때문에 32의 배수로 설정)

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

학습 진행 상황



학습 진행 상황



To do List

1. 학습 데이터 **accuracy** 개선
2. 현재까지 나온 **yolo version** 별 한계점 및 개선 사항, 구조 파악
3. **Action Reognition** 개념 및 알고리즘 파악

Reference

<https://herbwood.tistory.com/17>

<https://m.blog.naver.com/sogangori/221011203855>

<https://leedakyeong.tistory.com/entry/Object-Detection-YOLO-v1v6-%EB%B9%84%EA%B5%90>

<https://yeomko.tistory.com/47>

<https://blog.naver.com/intelliz/221709190464>

[https://velog.io/@skhim520/YOLO-v2-%EB%85%BC%EB%AC%B8-%EB%A6%A
C%EB%B7%B0](https://velog.io/@skhim520/YOLO-v2-%EB%85%BC%EB%AC%B8-%EB%A6%A
C%EB%B7%B0)

<https://bokonote.tistory.com/11>

<https://arxiv.org/abs/1506.02640>

<https://arxiv.org/abs/1612.08242>

<https://arxiv.org/abs/1804.02767>

<https://arxiv.org/abs/2004.10934>

<https://herbwood.tistory.com/13>