BIGDATA & AI ANALYTICS
EXPERT COMPANY

One-Stage Detection
YOLO

kubwa®

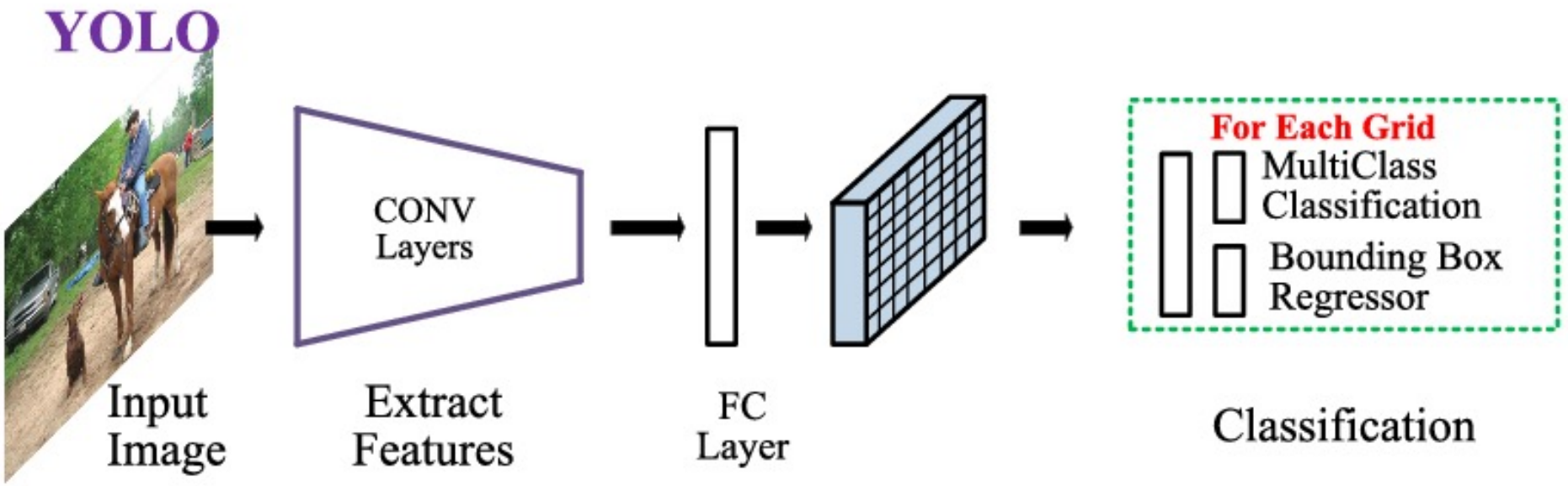
BIGDATA & AI ANALYTICS
EXPERT COMPANY

YOLO 개요

YOLO 개요

You only look once (YOLO)

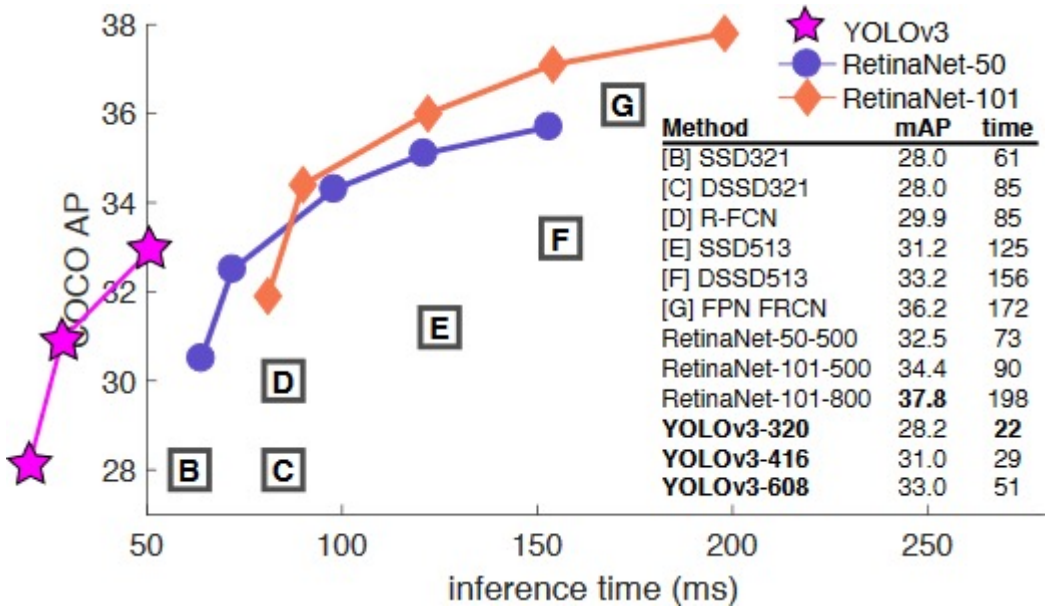
https://pjreddie.com/darknet/yolo/



YOLO V3 성능

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Speed + Accuracy

YOLO – V1, V2, V3 비교

kubwa

항목	V1	V2	V3
원본 이미지 크기	446 X 446	416 X 416	416 X 416
Feature Extractor	Inception 변형	Darknet 19	Darknet 53
Grid당 Anchor Box 수	2개	5개	Output Feature Map당 3개 서로 다른 크기와 스케일로 총 9개
Anchor box 결정 방법		K-Means Clustering	K-Means Clustering
Output Feature Map 크기 (Depth 제외)		13 x 13	13 x13, 26 X 26, 52X52 3개의 Feature Map 사용
Feature Map Scaling 기법			FPN(Feature Pyramid Network)

YOLO V1, V2, V3

kubwa

YOLO V1 빠른 Detection 시간
그러나 낮은 정확도

YOLO V2 수행시간과 성능 모두 개선

YOLO V3 수행시간은 조금 느려졌으나
성능 대폭 개선

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

YOLO V1 특징

kubwa

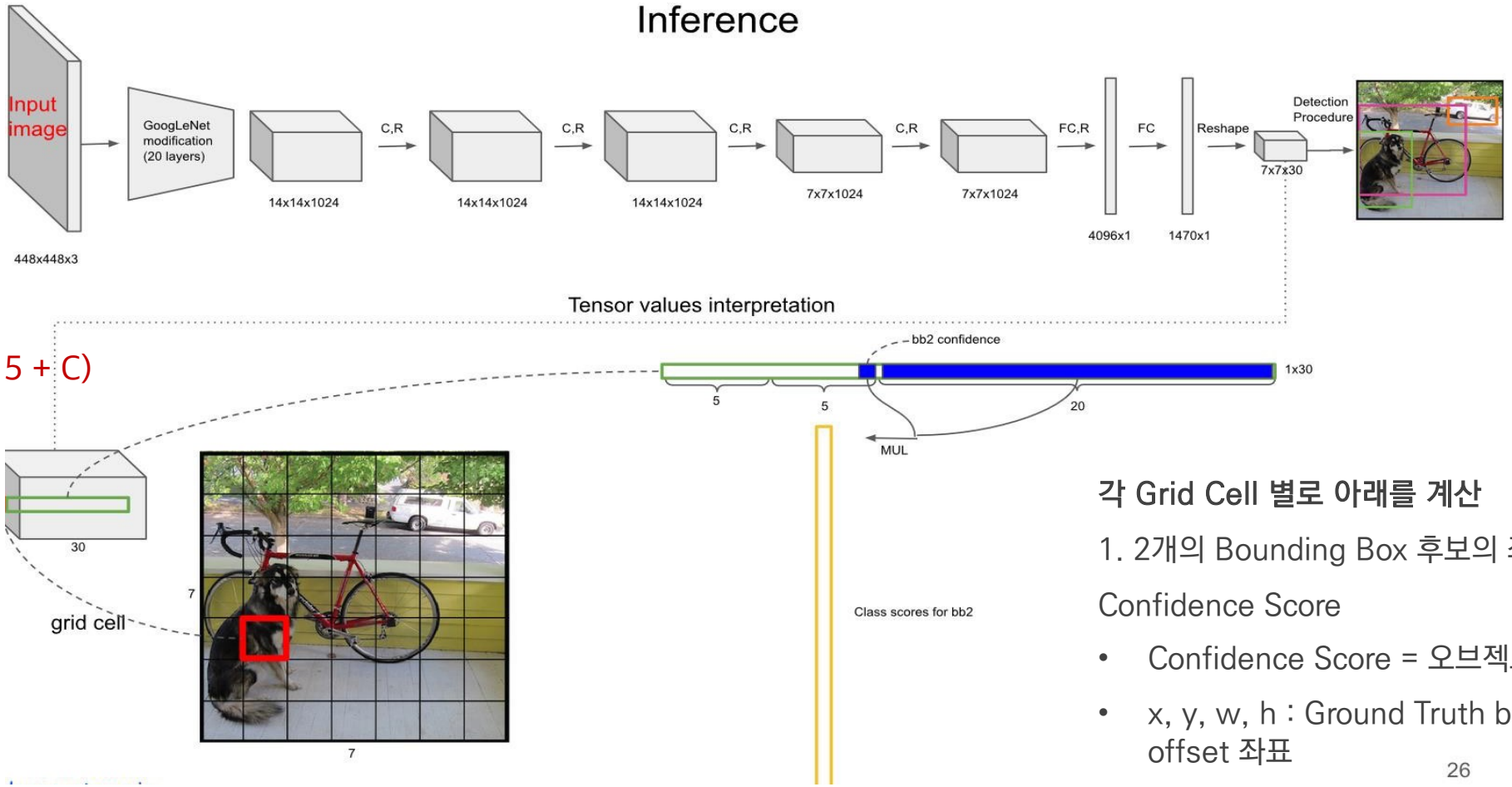
- Yolo V1 은 입력 이미지를 S X S Grid로 나누고 각 Grid의 Cell 이 하나의 Object에 대한 Detection 수행
- 각 Grid Cell 이 2개의 Bounding Box 후보를

  기반으로 Object의 Bounding Box 를 예측

S x S Grid

448 x 448
image input

YOLO-V1 Detection 모델

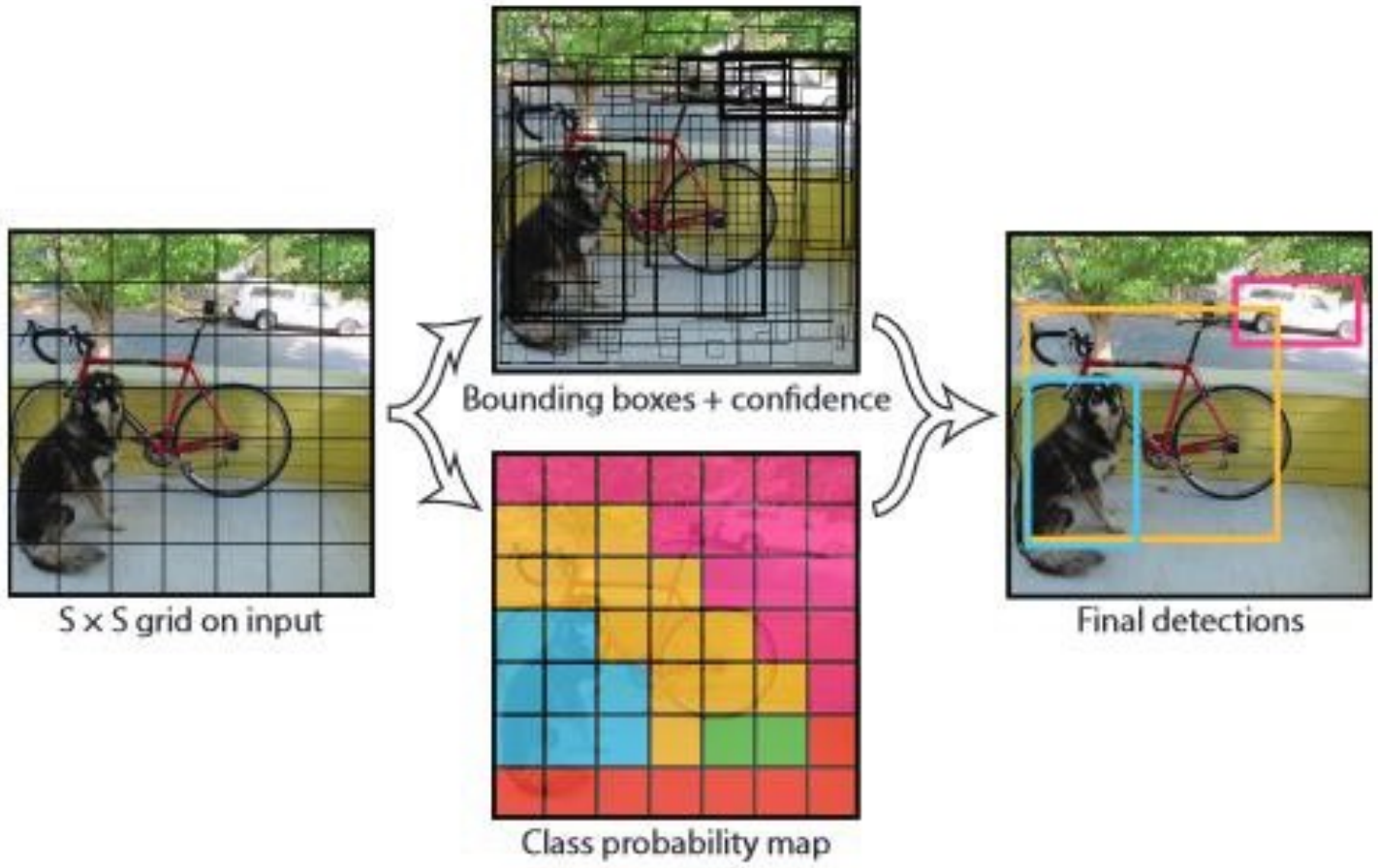


각 Grid Cell 별로 아래를 계산

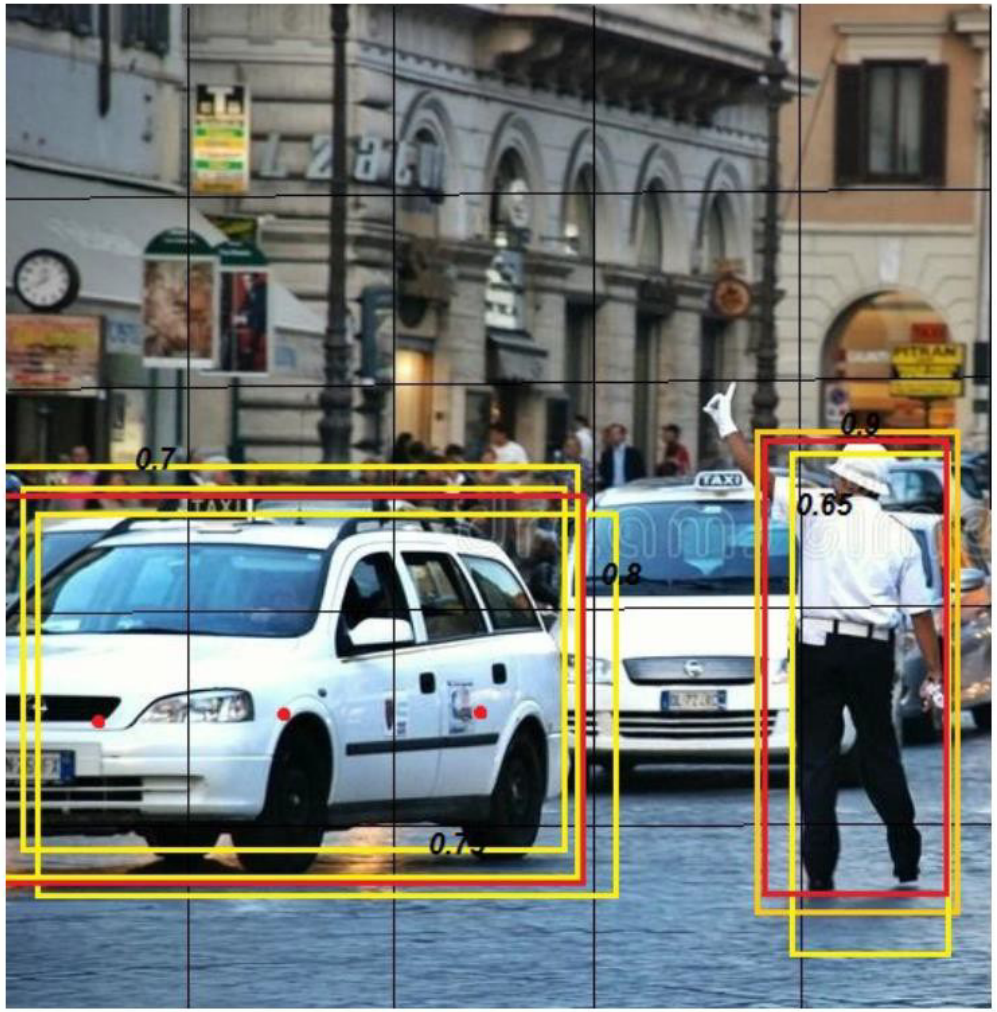
1. 2개의 Bounding Box 후보의 좌표와 해당 Box별 Confidence Score
   - Confidence Score = 오브젝트일 확률 * IOU 값
   - x, y, w, h : Ground Truth box와 후보 Box간 offset 좌표

2. 클래스 확률. Pascal VOC 기준 20개 클래스의 확률

YOLO-V1 Detection 모델



NMS(Non Max Suppression)으로 최종 BBox 예측



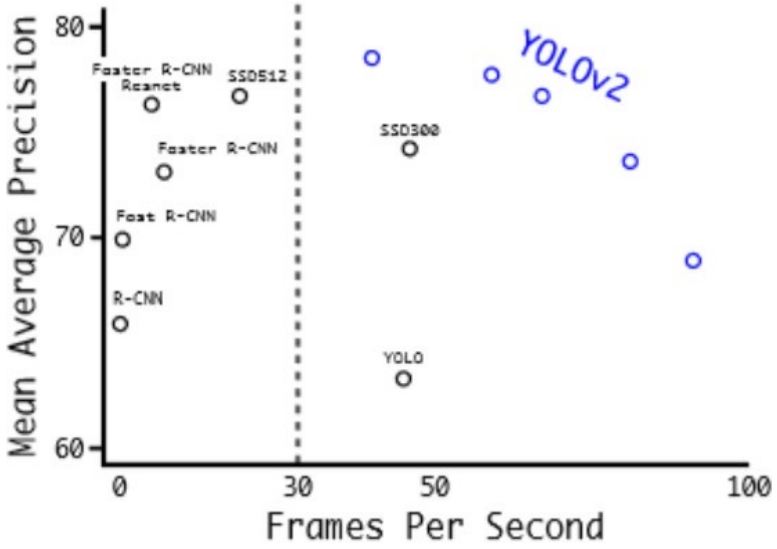
개별 Class 별 NMS 수행

1. 특정 Confidence 값 이하는 모두 제거
2. 가장 높은 Confidence값을 가진 순으로 Bbox 정렬
3. 가장 높은 Confidence를 가진 Bbox와 IOU와 겹치는 부분이 IOU Threshold 보다 큰 Bbox는 모두 제거
4. 남아 있는 Bbox에 대해 3번 Step을 반복

Object Confidence와 IOU Threshold로 Filtering 조절

YOLO-v2 Detection 시간 및 성능

PASCAL VOC 2007 Detection 시간



MS-COCO 기준 Detection 성능

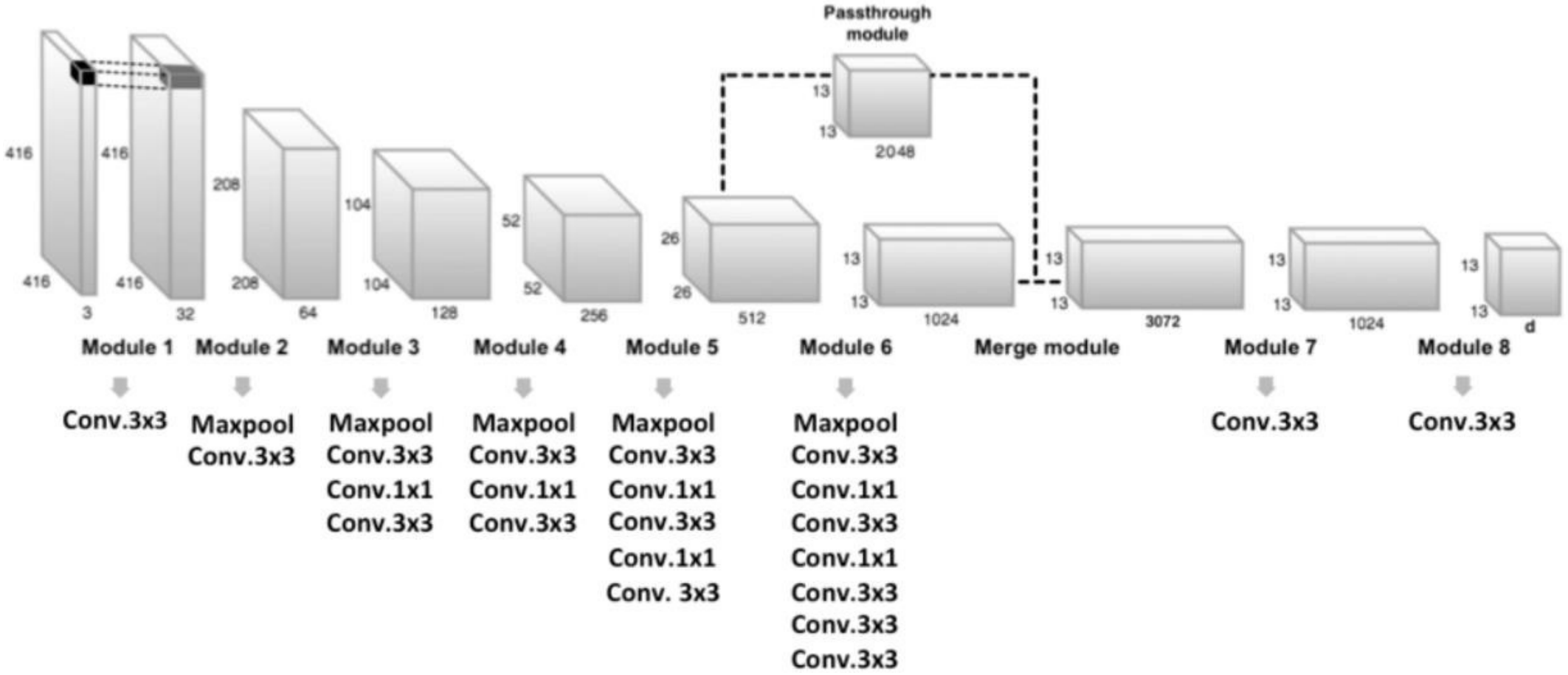
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [5]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN[1]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN[15]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300 [11]	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4

Yolo V2 특징

YOLO V2

- Batch Normalization
- High Resolution Classifier: 네트워크의 Classifier 단을 보다 높은 resolution(448x448)로 fine tuning
- 13 x 13 feature map 기반에서 개별 Grid cell 별 5개의 Anchor box에서 Object Detection
- Darknet-19 Classification 모델 채택
- 서로 다른 크기의 image들로 네트워 학습

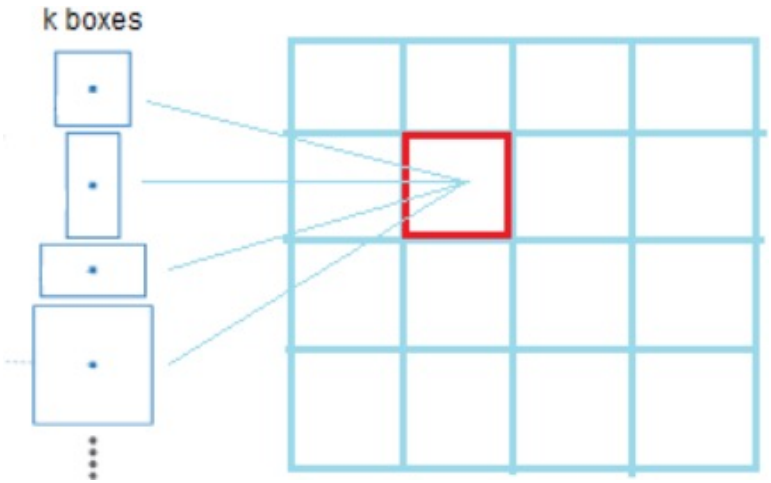
YOLO-v2 Network 구조



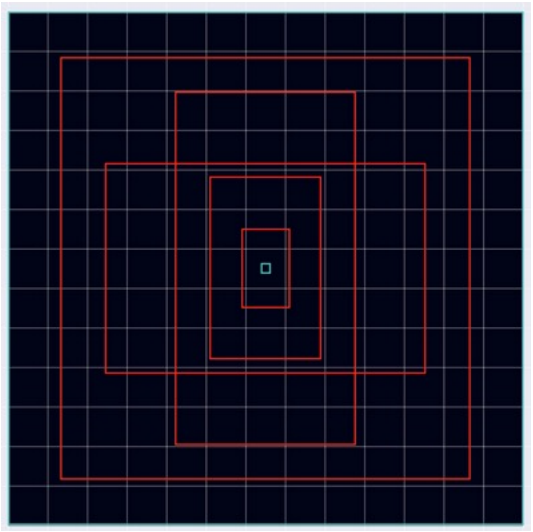
Yolo v2 – Anchor Box

- SSD와 마찬가지로 1개의 Cell에서 여러 개의 Anchor를 통해 개별 Cell에서 여러 개 Object Detection 가능
- K-Means Clustering 을 통해 데이터 세트의 이미지 크기와 Shape Ratio 따른 5개의 군집화 분류를 하여 Anchor Box를 계산

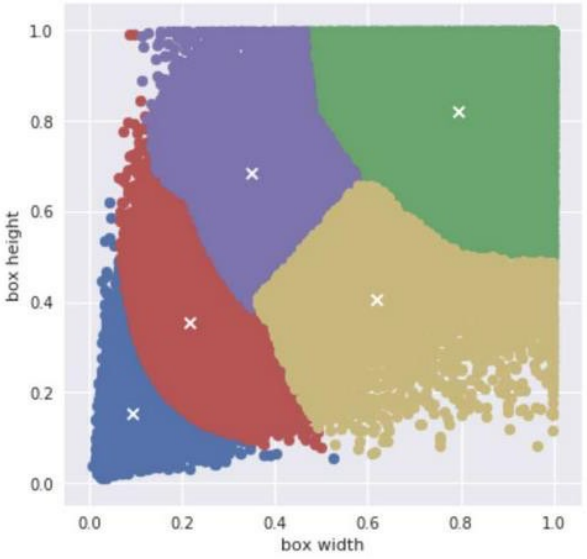
Convolutional With Anchor Boxes



5개 Anchor Box

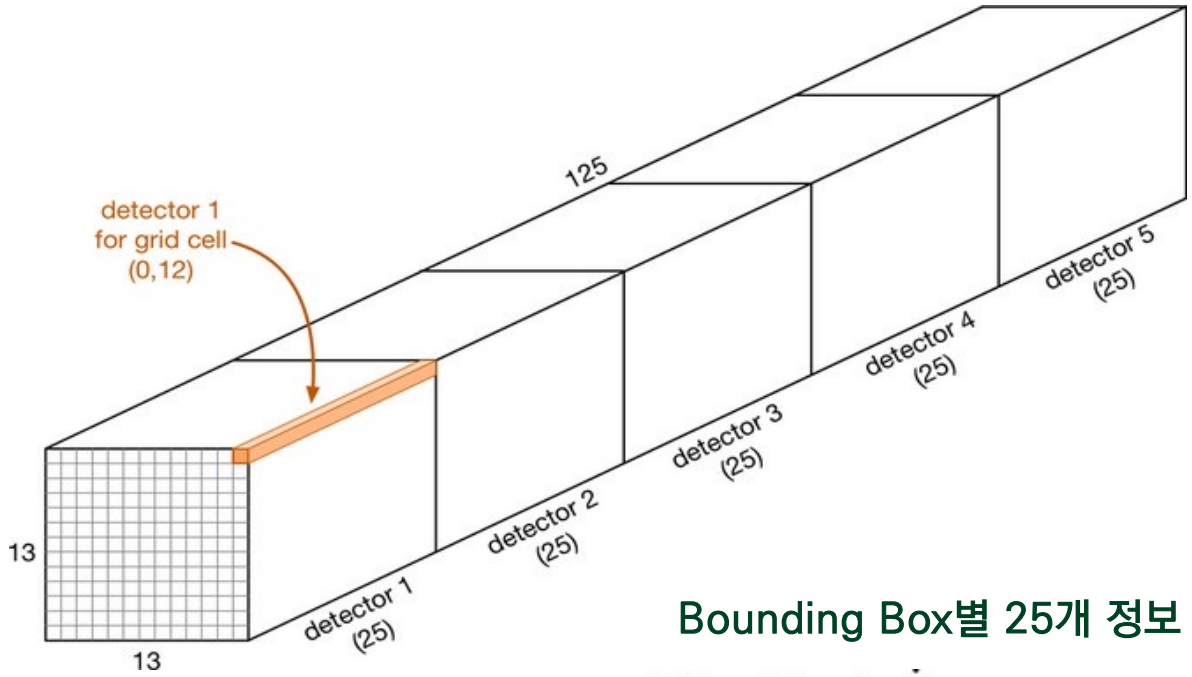


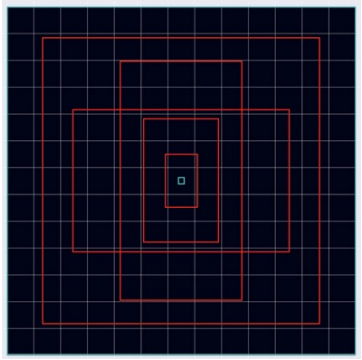
K-Means Clustering



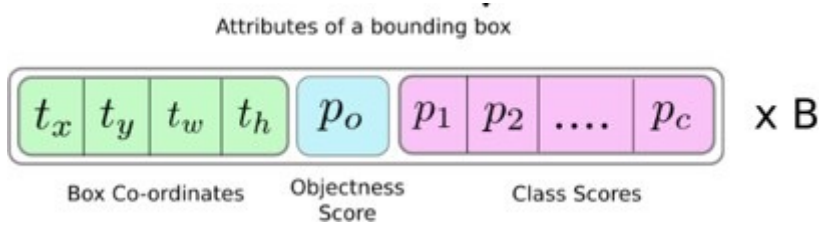
kubwa

YOLO v2 Output Feature Map





Bounding Box별 25개 정보

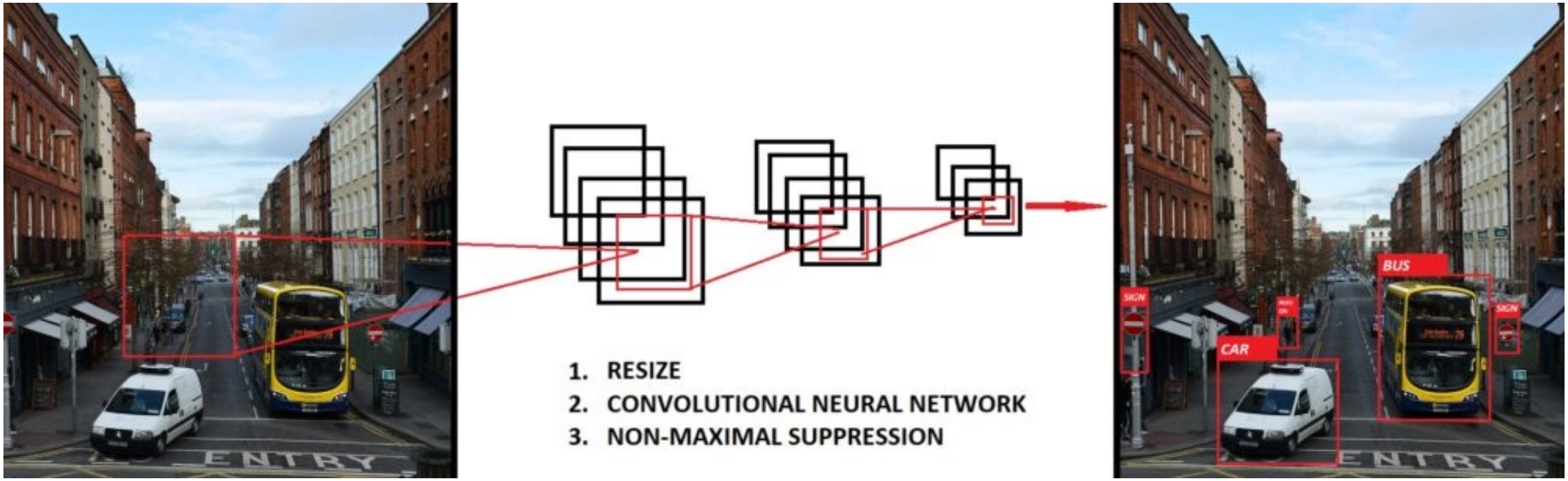


Yolo v3 특징

YOLO V3

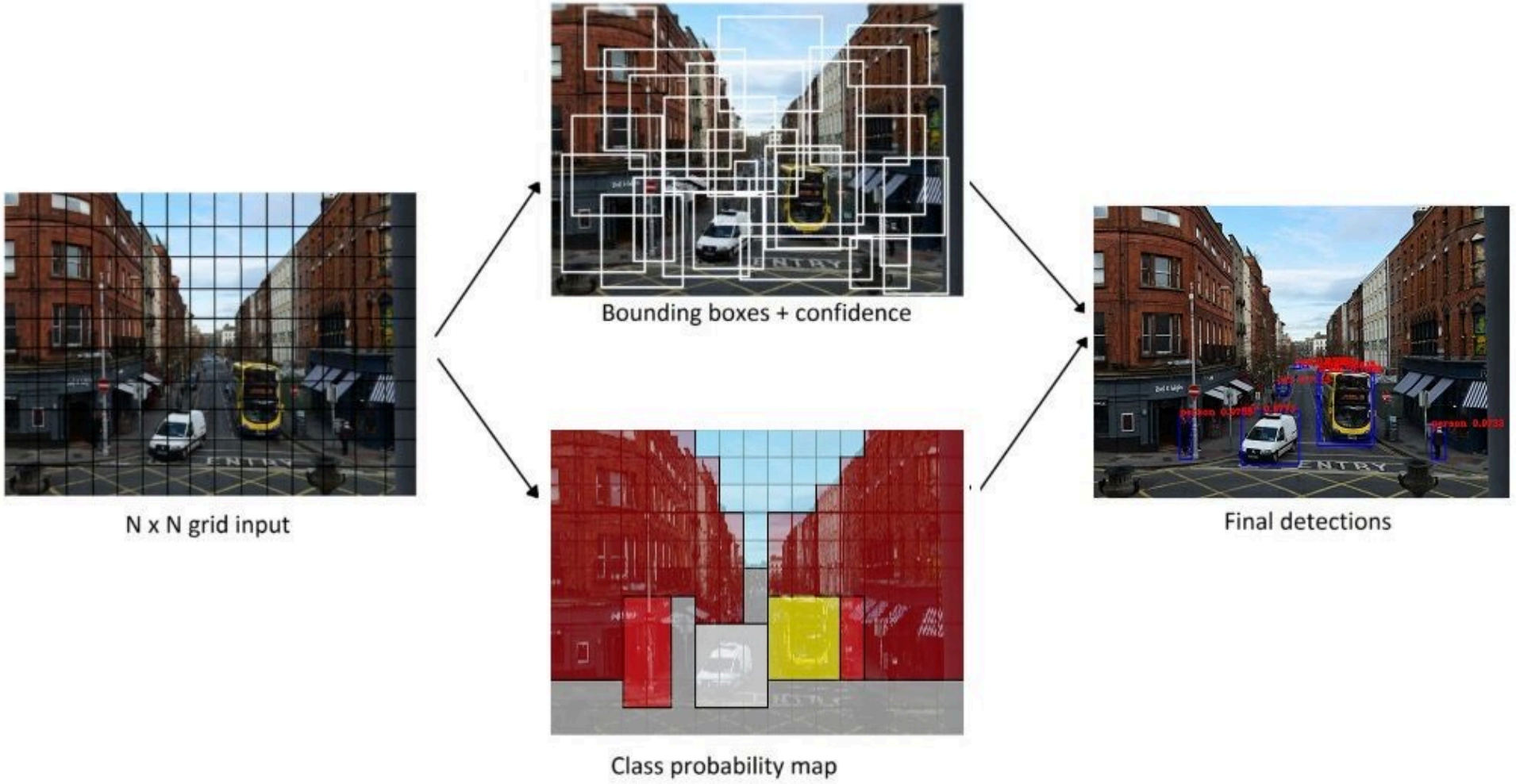
- Feature Pyramid Network 유사한 기법을 적용하여 3개의 Feature Map Output에서 각각 3개의 서로 다른 크기와 scale을 가진 anchor box 로 Detection
- Classification단을 보다 높은 Classification을 가지는 Darknet-53
- Multi Labels 예측: Softmax가 아닌 Sigmoid 기반의 logistic classifier로 개별 Object의 Multi labels 예측

YOLO v3 개요


1. RESIZE
2. CONVOLUTIONAL NEURAL NETWORK
3. NON-MAXIMAL SUPPRESSION
BUS
CAR
SIGN
SIGN
ENTRY
ENTRY

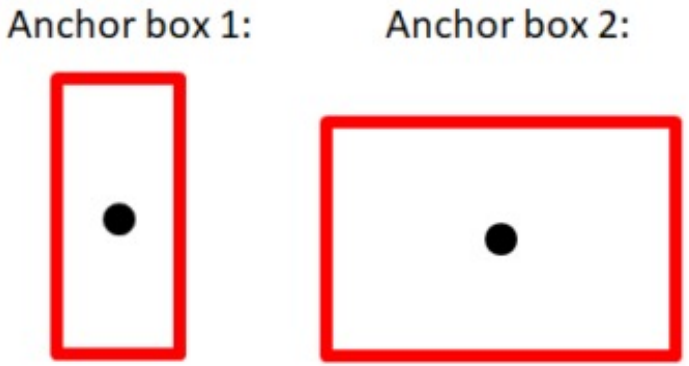
kubwa

Yolo v3 개요



kubwa

Yolo v3 개요



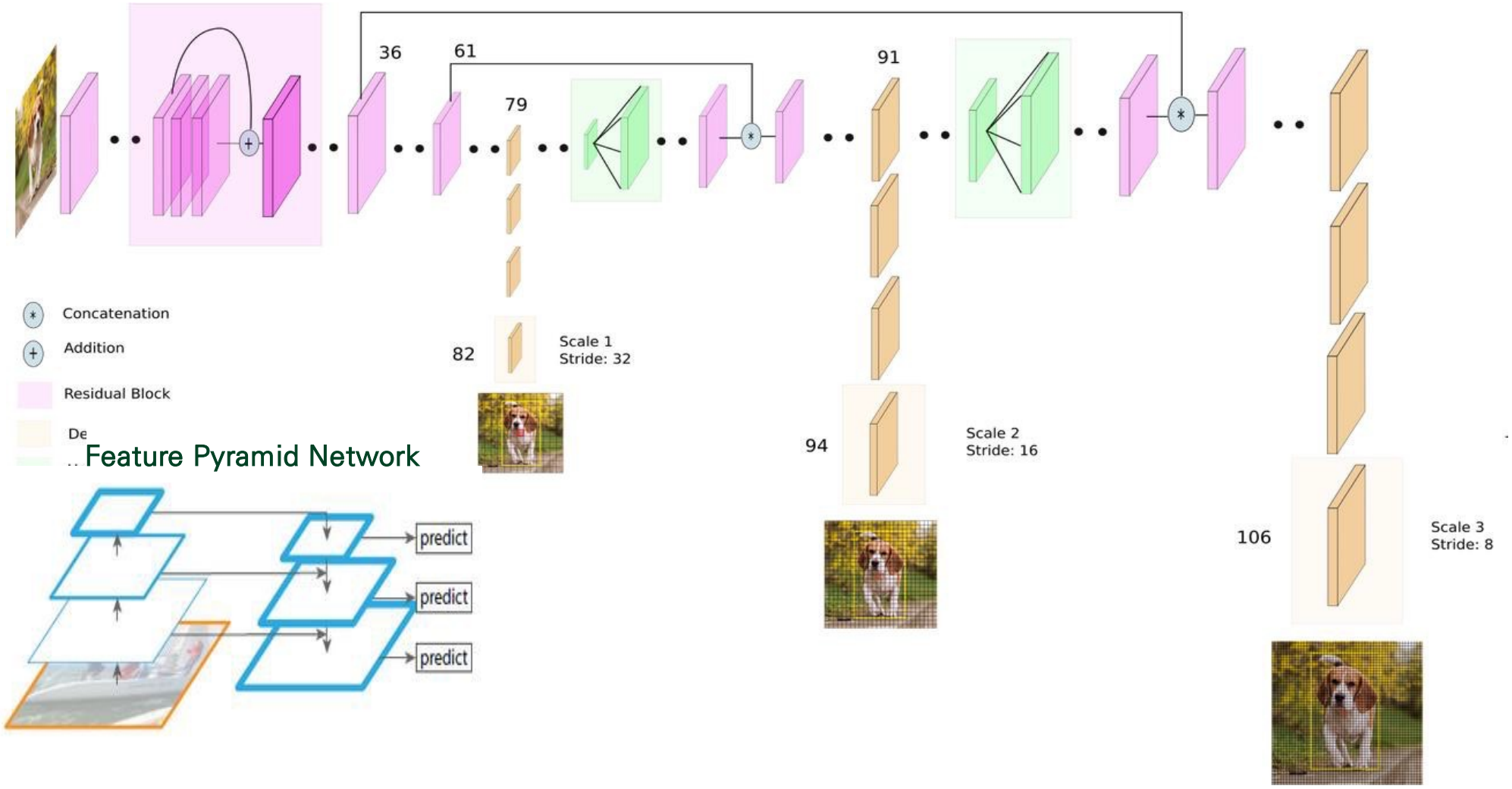
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \cdot \\ \cdot \\ \cdot \\ c_3 \end{bmatrix}$$

Anchor box 1

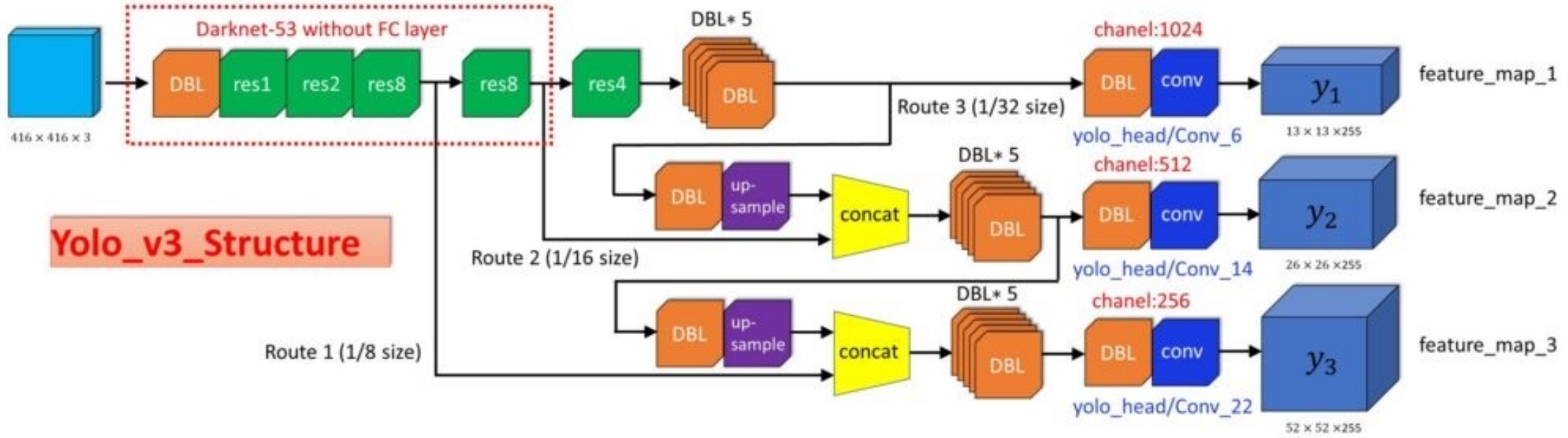
Anchor box 2

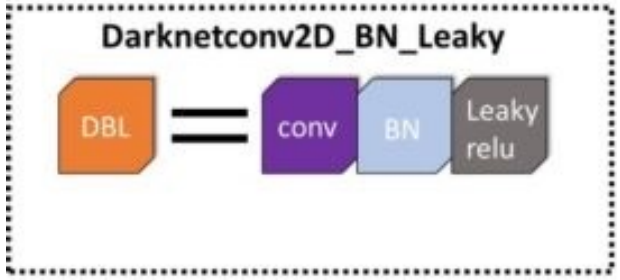
kubwa

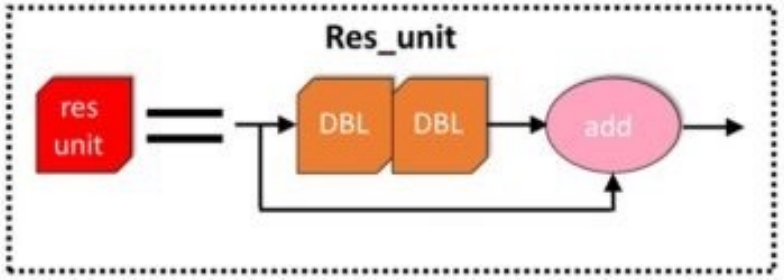
YOLO v3 Network 구조

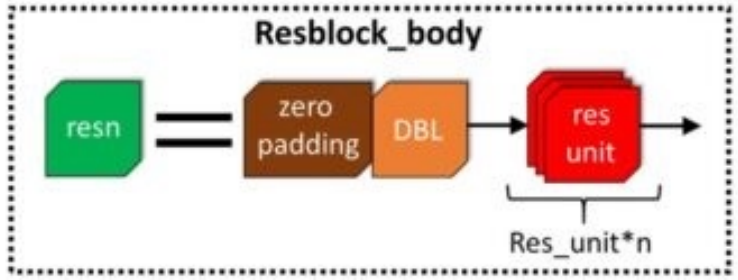


Yolo V3 Network 구조

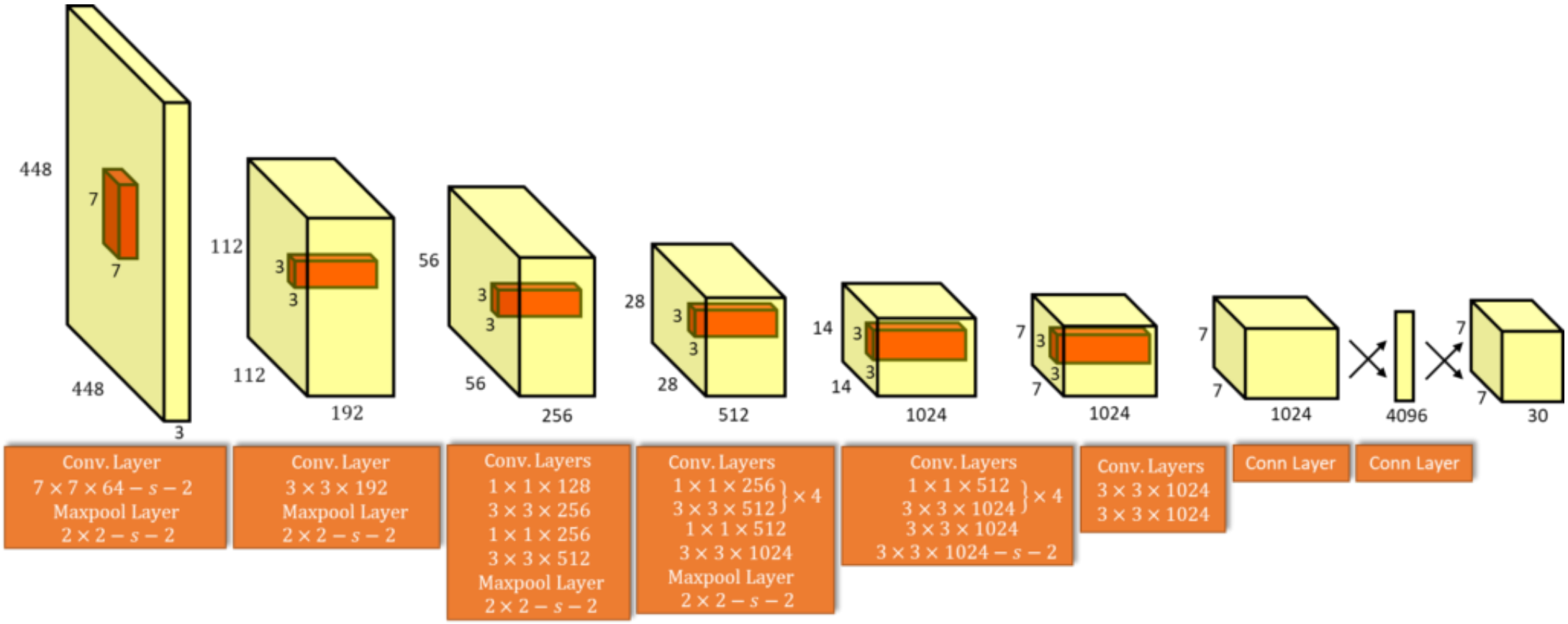




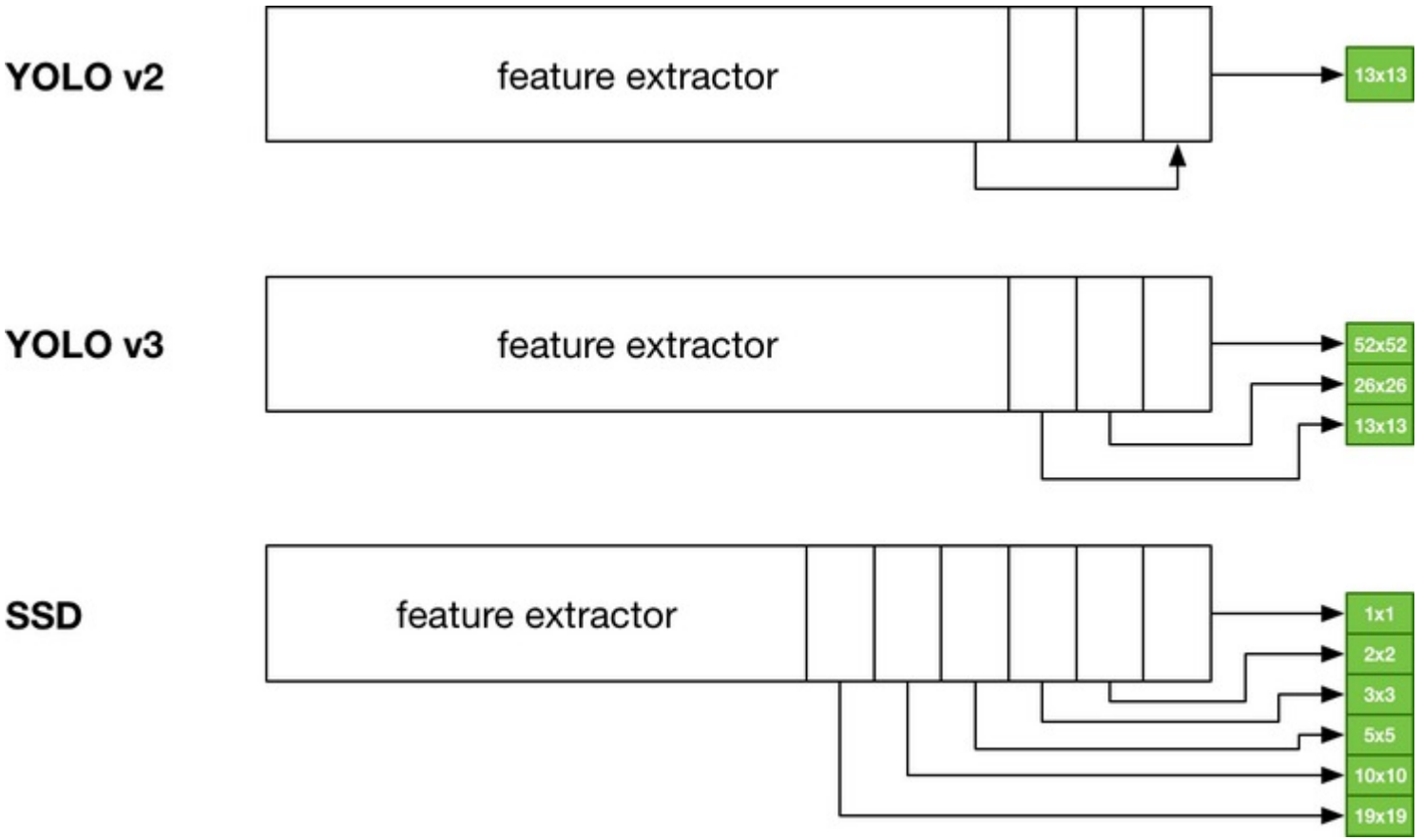




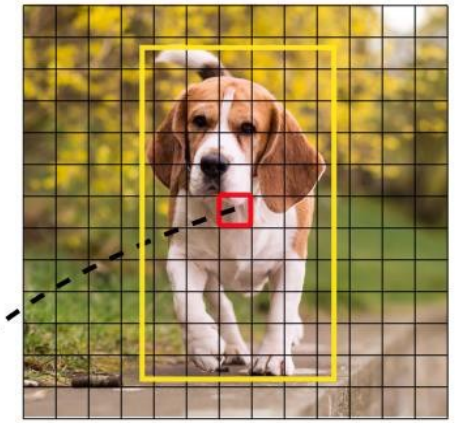
Yolo v3 Classification 구조



YOLO와 SSD의 비교



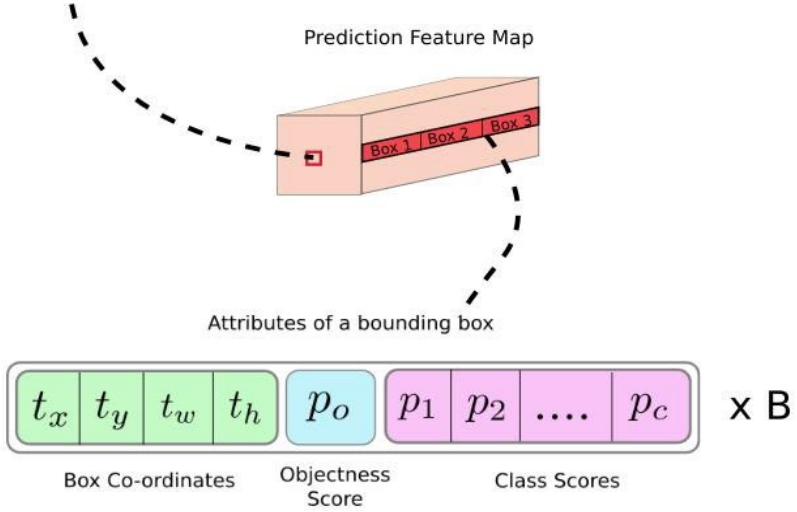
YOLO v3 Output Feature Map



13 x 13

26 x 26

52 x 52



Darknet-53 특성

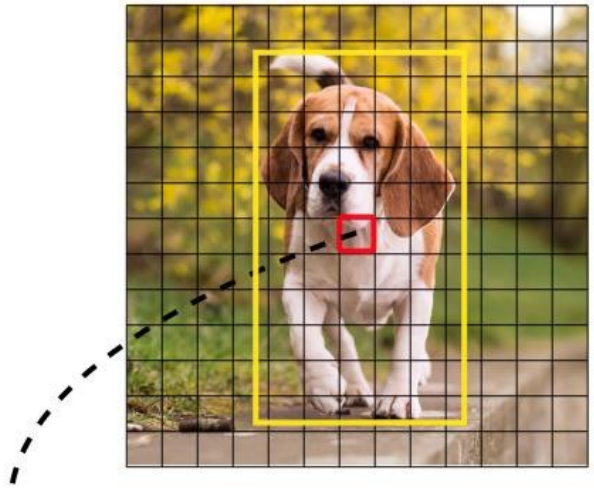
kubwa

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLO v3 Training

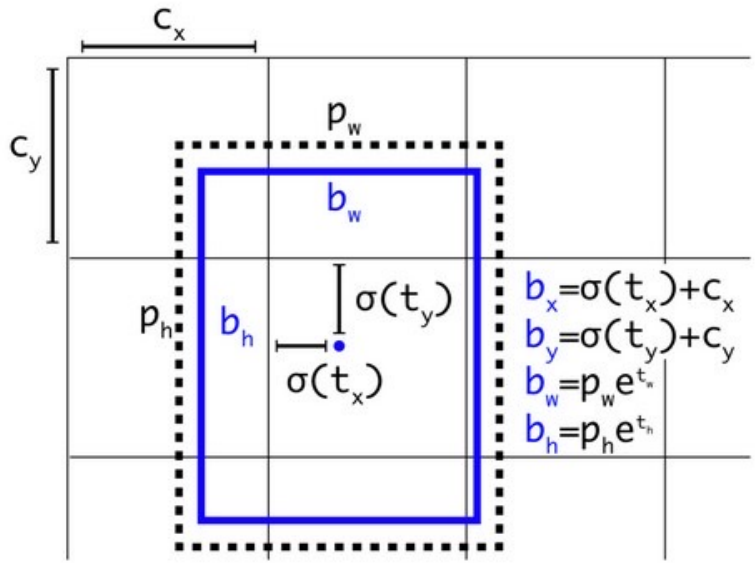
```
<annotation>
    <folder>VOC2007</folder>
    <filename>003585.jpg</filename>
   <size>
        <width>333</width>
        <height>500</height>
        <depth>3</depth>
    </size>
    <object>
        <name>person</name>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>138</xmin>
            <ymin>183</ymin>
            <xmax>259</xmax>
            <ymax>411</ymax>
        </bndbox>
    </object>
    <object>
        <name>motorbike</name>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>89</xmin>
            <ymin>244</ymin>
            <xmax>291</xmax>
            <ymax>425</ymax>
        </bndbox>
    </object>
    . . .
```



multi-scale training,
Data augmentation

Location Prediction

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$\Pr(\text{object}) \cdot \text{IoU}(b, \text{object}) = \sigma(t_o)$$



(pw,ph): anchor box size

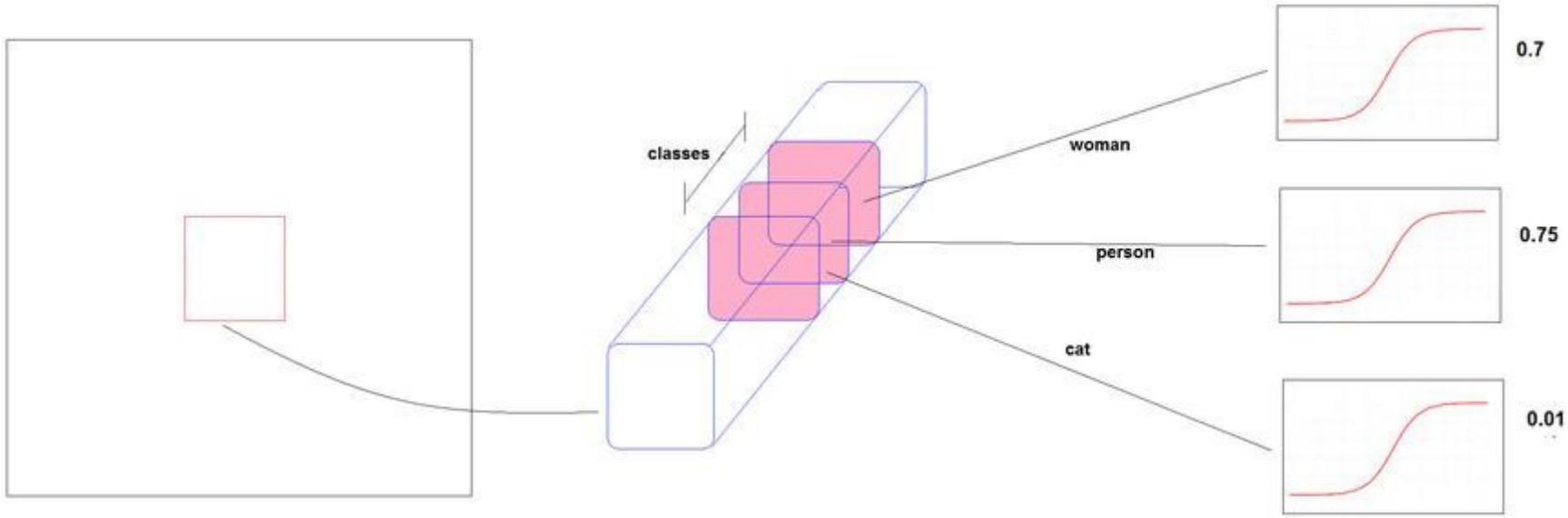
(tx,ty,tw,th) : 모델 예측 offset 값

(bx,by), (bw,bh): 예측 Bounding box 중심 좌표와 Size

Center 좌표가 Cell 중심을 너무 벗어나지 못하도록
0 ~ 1 사이의 시그모이드 값으로 조절

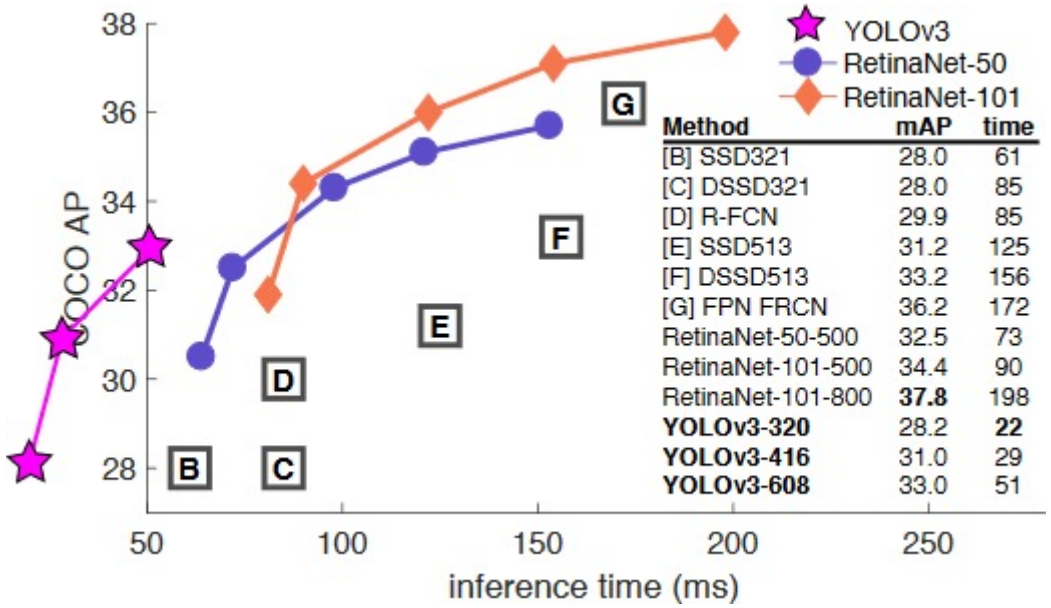
Multi Labels 예측

여러 개의 독립적인 Logistic Classifier 사용

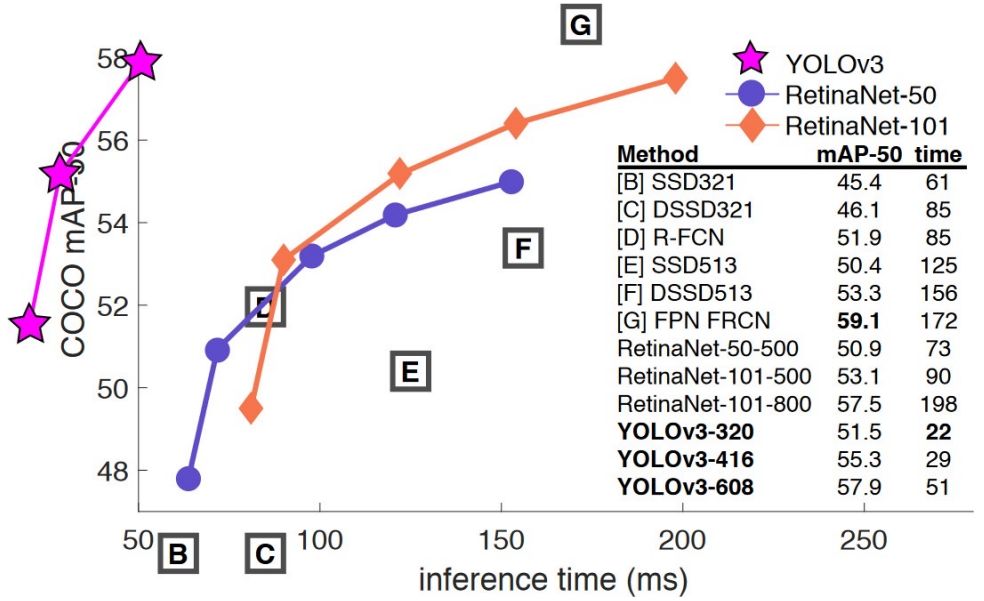


kubwa

YOLO v3 성능 비교



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

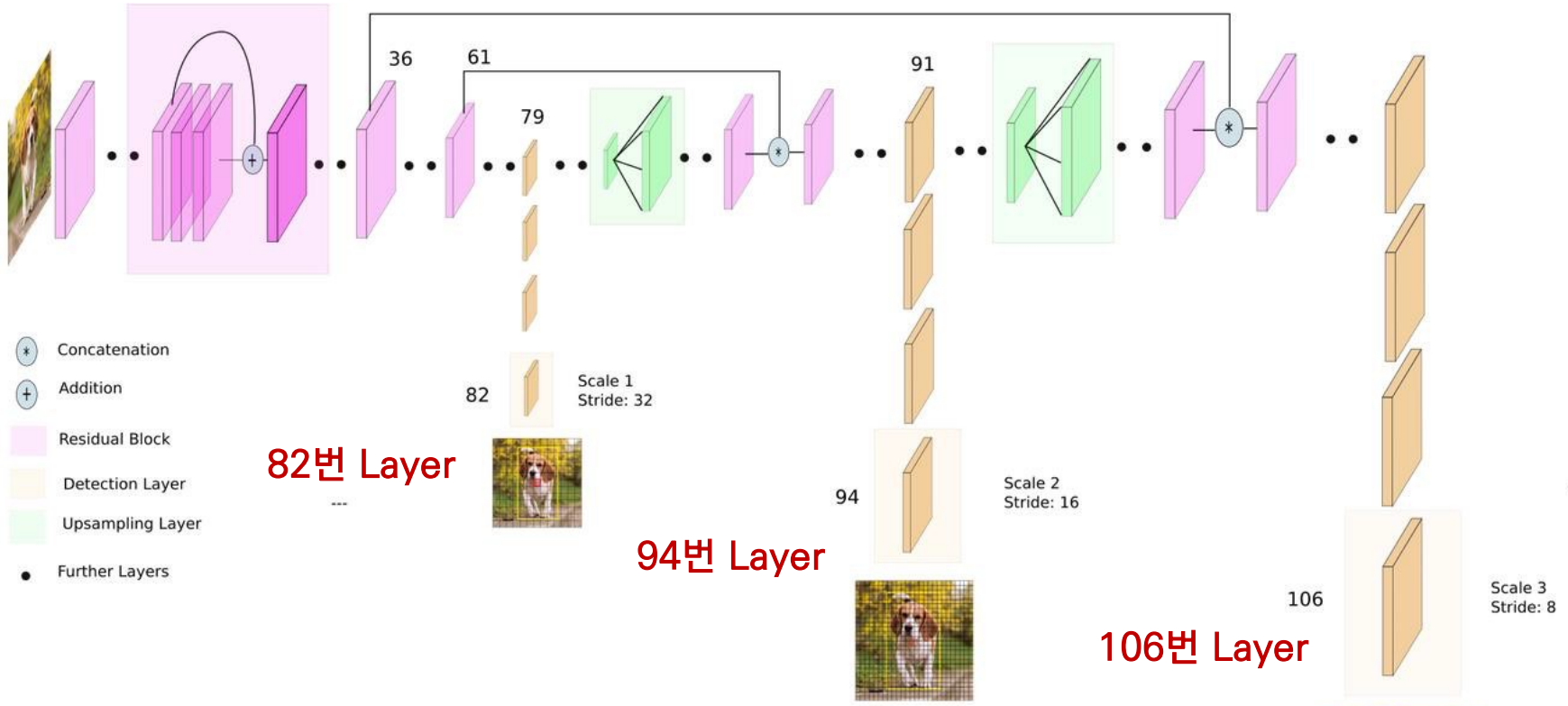
BIGDATA & AI ANALYTICS
EXPERT COMPANY

OpenCV Inference

Open CV DNN으로 Yolo inference

- OpenCV Yolo inference 코드는 기존 OpenCV inference코드와 다름.
- Darknet 구성 환경 및 YOLO 아키텍처에 따라 사용자가 직접 Object Detection 정보 추출
- Weight 모델 파일과 config 파일은 Darknet 사이트에 Download 가능.
- cv2.dnn.readNetFromDarknet(config 파일, weight 모델 파일)으로 pretrained된 inference 모델 로딩.
- readNetFromDarket(config 파일, weight 모델 파일)에서 config 파일 인자가 weight 모델 파일 인자보다 먼저 위치함에 유의

3개의 scale Output Layer에서 직접 Detection 결과 추출

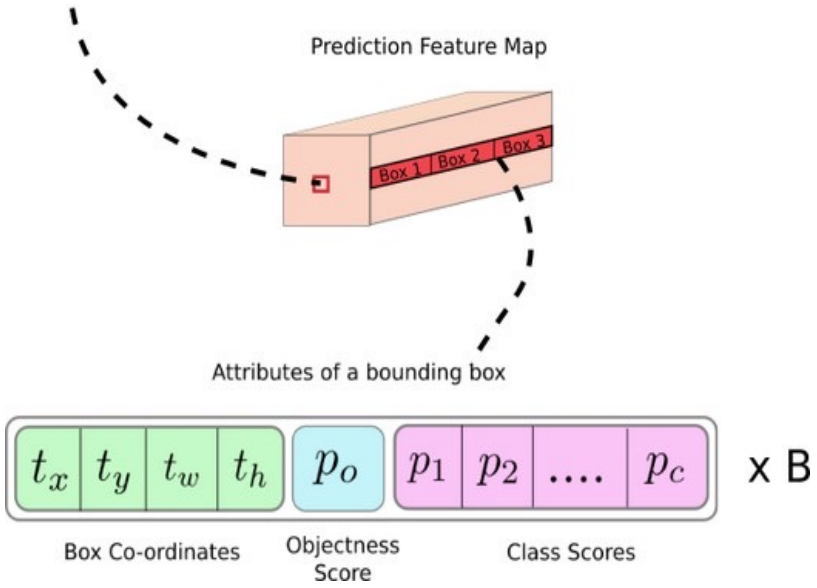


사용자가 직접, 3개의 다른 Scal별로 구성된 output layer에서 Object Detect 결과를 추출

사용자가 직접, NMS(Non Maximum Suppressing)로 최종 결과 필터링

Bounding box 정보추출 시 직접 85개의 구성에서 추출

Coco 데이터 세트로 Pretrained 된 모델에서 bounding box 정보 추출하기



- Bounding Box 정보를 4개의 Box 좌표, 1개의 Object Score, 80개의 Class score(Coco는 80개의 Object Category임)로 구성된 총 85개의 정보 구성에서 정보 추출 필요.
- Class id와 class score는 이 80개 vector에서 가장 높은 값을 가지는 위치 인덱스와 그 값임.

추출 좌표의 변환

- OpenCV Yolo로 추출한 좌표는 Detected된 Object의 center와 width, height 값이므로 이를 좌상단, 우하단 좌표로 변경 필요

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

OpenCV Inference 구현

cv2.dnn.readNetFromDarknet(config 파일, weight 모델 파일)으로 pretrained된 inference 모델 로딩.

- config 파일 위치 주의

사용자가 3개의 다른 Scal별로 구성된 output layer에서 Object Detect 결과 추출.

- Detected된 Object당 85개의 vecto를 가짐. 4개의 위치 정보는 center 점의 x,y 좌표와 width, height이므로 이를 좌상단, 우하단 좌표로 변경 필요.
- Class id와 class score는 이 80개 vector에서 가장 높은 값을 가지는 위치 인덱스와 그 값.

사용자가 직접, NMS(Non Maximum Suppressing)로 최종 결과 필터링 해야 함.

- OpenCV에서 제공하는 NMS 함수로 NMS 최종 필터링 된 Object 시각화

BIGDATA & AI ANALYTICS
EXPERT COMPANY

YOLO Keras 학습

YOLO-Keras기반 Open source 패키지

kubwa

- 보다 쉬운 환경 설정
- Keras의 Callbacks, Tensorboard, Preprocessing등의 다양한 기능 활용
- 소스코드가 보다 이해하기 쉬우며 Customization도 쉽게 가능.
- Darknet C/C++로 구현된 Original Yolo 패키지 보다는 처리 속도가 느림

Keras YOLO Open source 패키지

kubwa

qqwweee keras-yolo3

https://github.com/qqwweee/keras-yolo3

- 학습, Inference가능하나 Evaluation 지원하지 않음.
- CSV 포맷의 데이터 입력 지원(VOC, COCO를 CSV로 변환하는 유틸리티 지원)

Experiencor keras-yolo3

https://github.com/experiencor/keras-yolo3

- 학습, Inference, Evaluation이 가능
- XML 포맷의 데이터로만 입력 가능(CSV나 JSON을 XML 포맷으로 변경 필요)