YOLO(객체 인식 알고리즘)

유튜브: https://www.youtube.com/watch?v=2bHi86HJ7uM

YOLO v1

YOLO v2

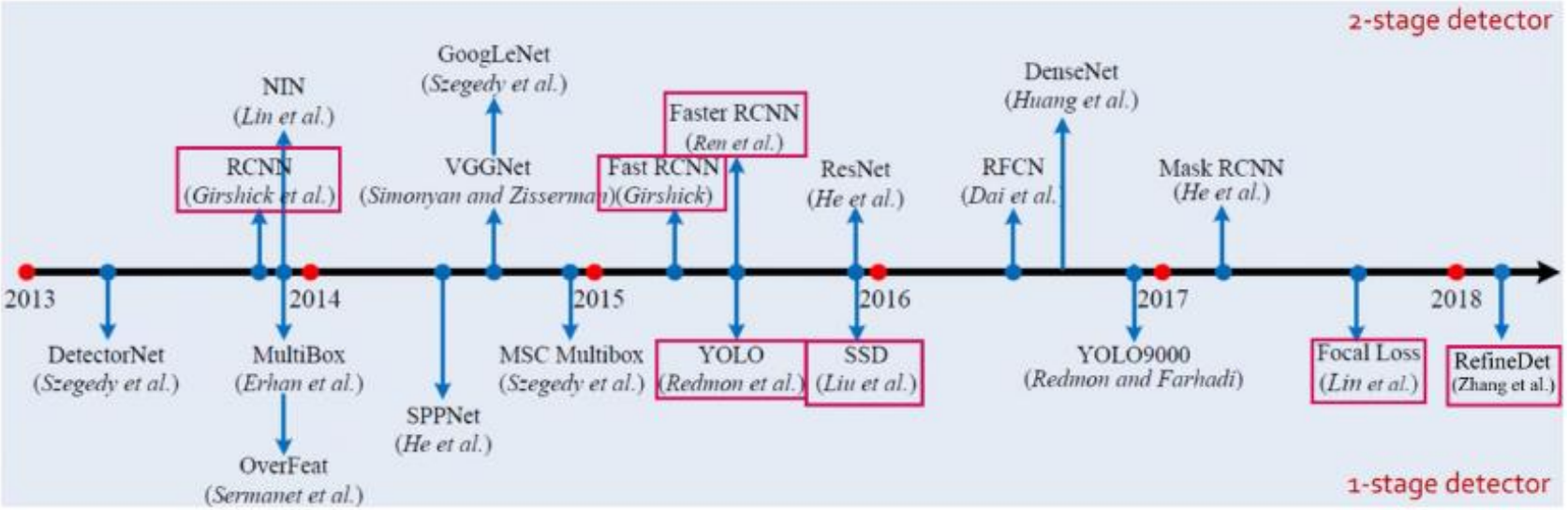
YOLO v3

YOLO v4, v5

YOLO

<2-Stage Detector>

Regional Proposal과 Classification이 순차적으로 이루어진다.



- 속도

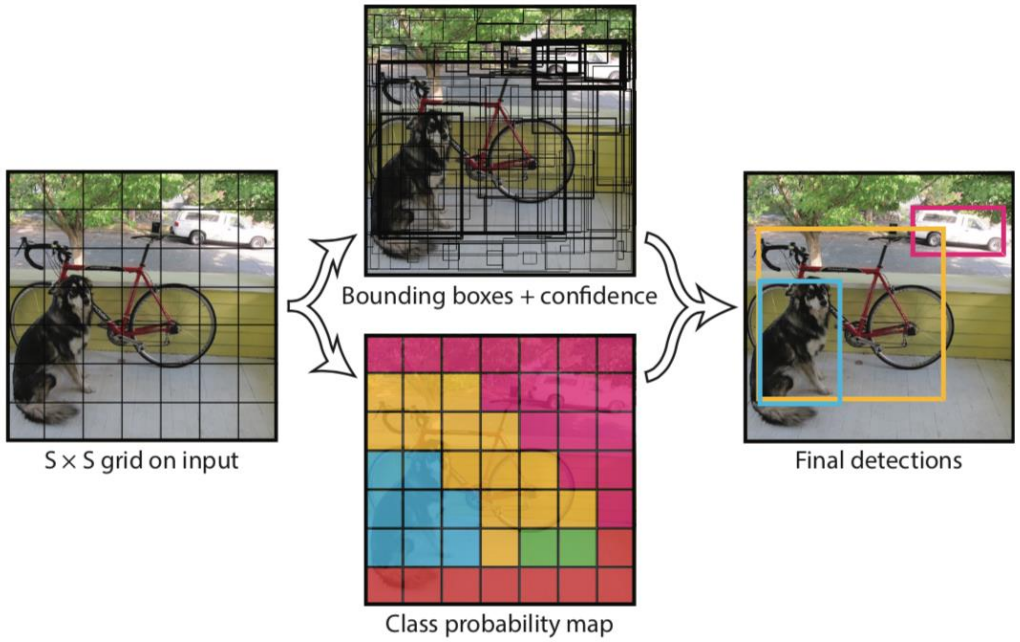
  1-stage detector > 2- stage detector
- 정확도

  1-stage detector < 2- stage detector

<1-Stage Detector>

regional proposal와 classification이 동시에 이루어진다.

YOLO v1



Detection 과정

1. Input 이미지를 SxS grid로 분할

2. Grid cell 당 bounding box와 Class probability 예측
 - 각 그리드 셀은 Bounding box B와 해당 box의 confidence score (신뢰도 점수)를 예측

Confidence score = Pr(object) * IOU

*Pr(Object)는 해당 그리드에 물체가 있을 확률

* IOU = 실제 객체 위치와 예측한 객체 위치가 얼마나 일치하는지에 대한 지표

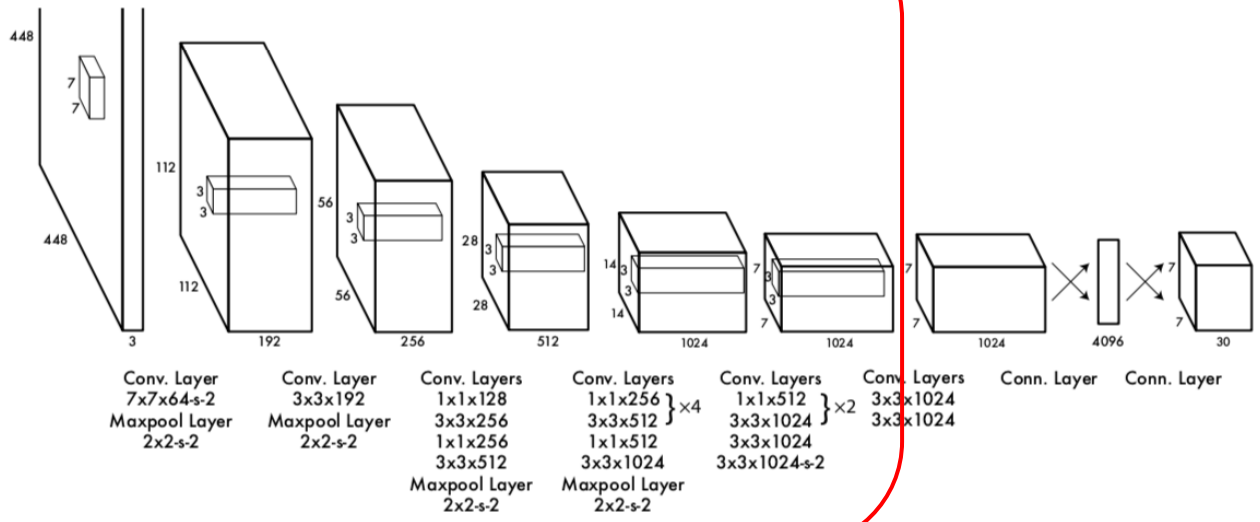
Bounding box ={x,y,w,h,confidence score}

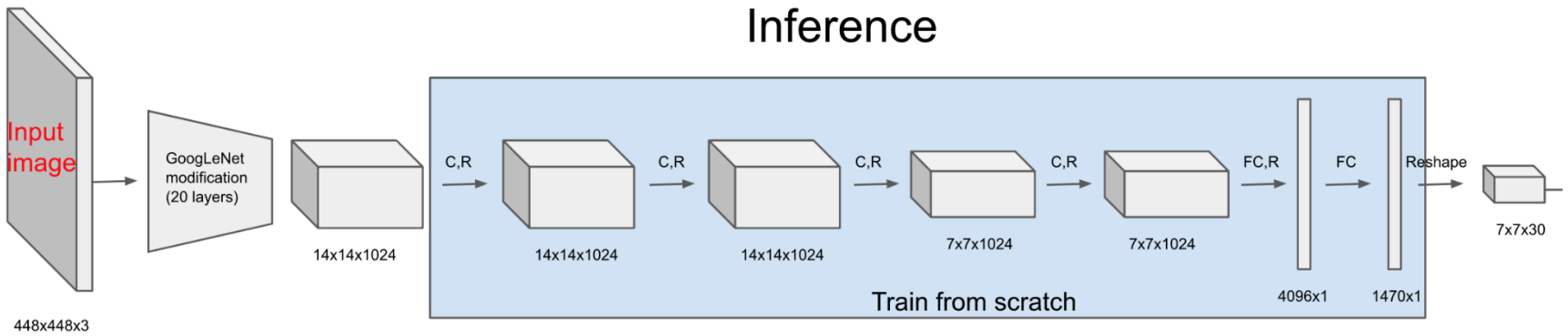
- 각 grid cell은 C개의 class probability예측

Pr(Class i | Object)

*그리드 셀이 객체 포함할 때 해당 객체가 i번째 클래스일 확률

YOLO v1


448
7
7
448
3
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
×4
Conv. Layers
1x1x512
3x3x1024
3x3x1024
3x3x1024-s-2
×2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer


Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Train from scratch

YOLO v2

YOLO 장점
- 간단한 single 구조
- 빠른 속도

YOLO 단점
- 공간적 제약
- Localization 부정확
- 낮은 recall

1.Better(정확성)

Batch Normalization, High Resolution Classifier, Convolutional with Anchor boxes, Dimension Clusters, Direct location prediction, Fine-Grained Features, Multi-Scale Training

2.Faster(속도 향상)

Darknet-19

3.Stronger(class 다양성)

Hierarchical classification, Dataset combination with WordTree, Joint classification and detection

YOLO v2

Better

1. Batch Normalization

기존 모델에서 dropout layer를 제거 하고 batch normalization추가

mAP 2%증가

2. High Resolution Classifier

기존모델 224x224로 학습된 GoogleNet 모델 -> 448x 448 크기로 object detection

-> object detection 학습 전 image classification모델을 큰 해상도 이미지에 대해 fine-tuning , mAP 4%증가

3. Convolutional with Anchor Boxes

앵커박스 도입 -> recall 증가

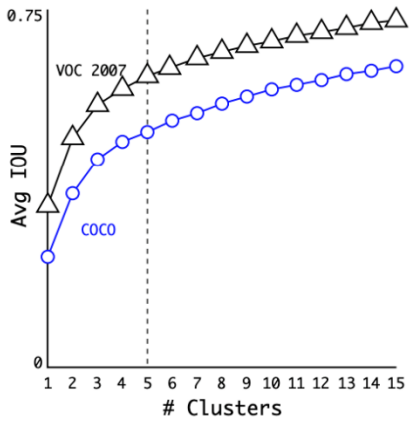
4. Dimension Clusters

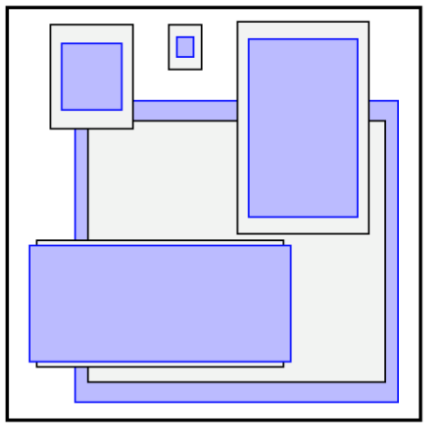
앵커박스 후보군 잡을 때 Kmeans 알고리즘 사용 (거리 IOU 기반)

k값(cluster)이 적으면 연산 속도는 올라가지만

Anchor Box 후보군이 제한적이어서 Detection 성능이 떨어지고,

K값을 계속해서 늘리면 성능은 좋아지지만 연산 속도에 대한 trade off





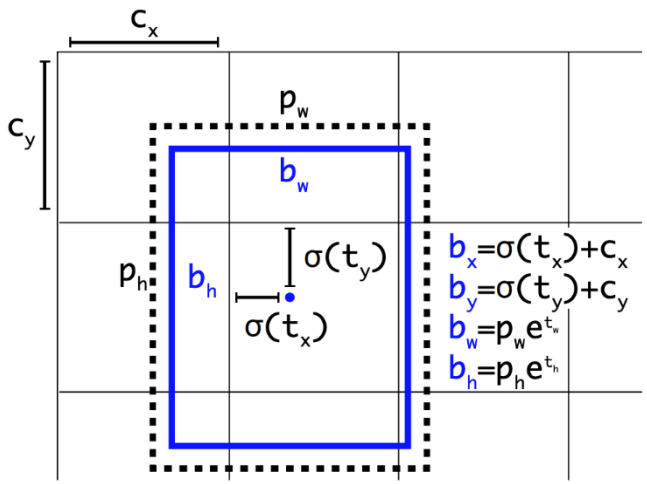
YOLO v2

Better

5. Direct Location Prediction

Anchor box는 초기에 박스(x,y)위치 랜덤하게 예측해서 모델 불안정

Bounding box={tx,ty,tw,th,to}

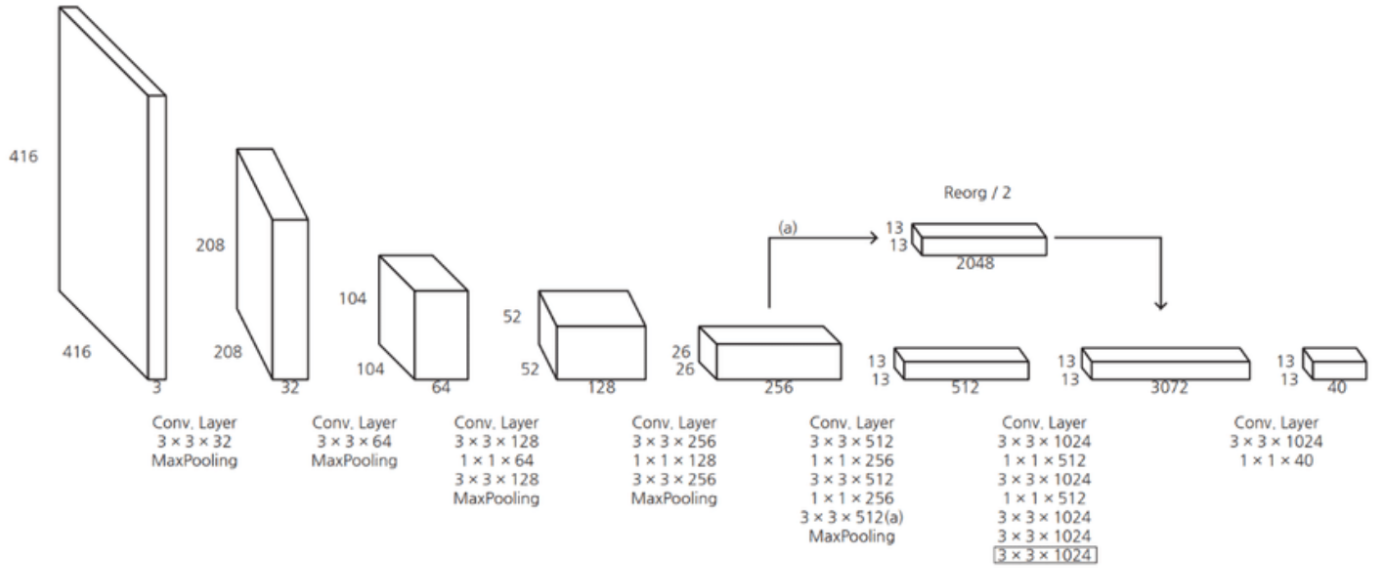


6.Fine-Grained Features

passthrough layer 추가

26 x 26 ->13 x 13 x 2048

+ 13x13 = 13 x 13 x 3072



7.Multi-scare Trainning

모델 학습 시 input size 변경

다양한 스케일에 대해 강인성

YOLO v2

Faster

- Darknet-19

  기존 yolo는 pretrained된 GoogleNet 사용 =너무 크고 복잡

  => 새로운 CNN아키텍쳐인 darknet

3x3 kernel 사용, 1x1 kernel통해 압축

Convolution 연산을 늘리고 Maxpooling을 줄임

Fully Connected Layer를 제거하고 Convolution 연산으로 대체

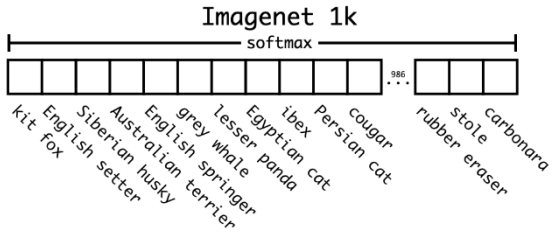
Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	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			

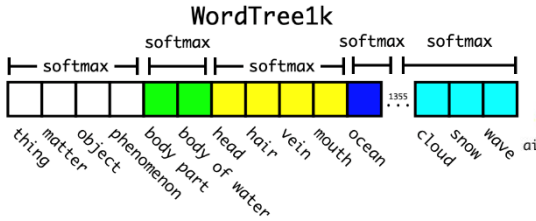
YOLO v2

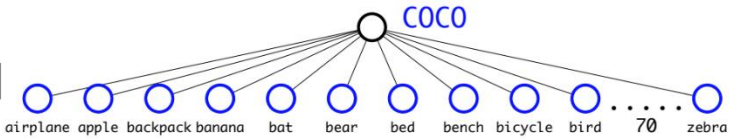
Stronger

- Hierarchical classification

방대한 크기의 class를 계층적으로 분류작업
softmax연산을 수행할 때 전체 클래스에 대해서
한번에 수행하지말고 각 대분류 별로 softmax수행

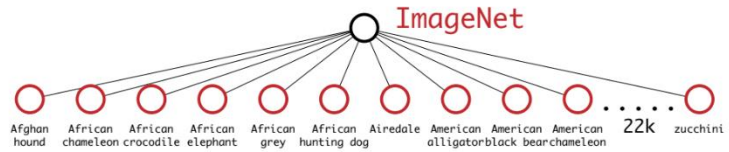


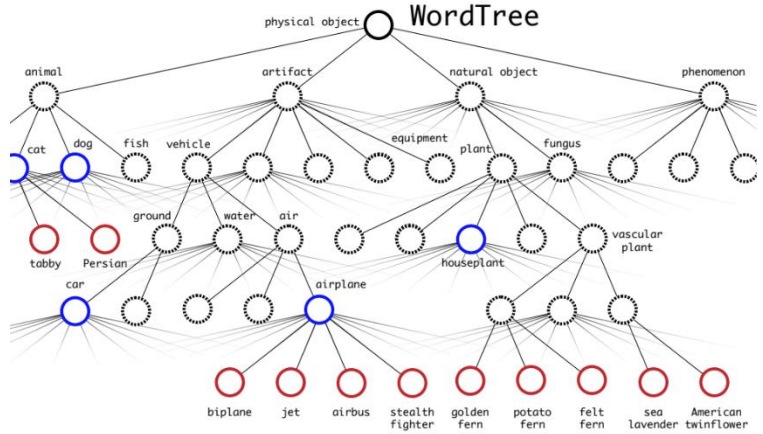




- dataset combination with word Tree

coco 데이터 셋(detection)과 imagenet(classification) 데이터셋의
라벨을 트리구조를 활용하여 섞음





- Joint classification and detection

Detection 용 이미지 – detection losss는그냥 역전파
classification loss는 해당 클래스와 상위레벨에 역전파
Classification용 이미지 – classification loss만 역전파

YOLO v3

YOLO v2 architecture 대부분 사용

- Feature Extractor

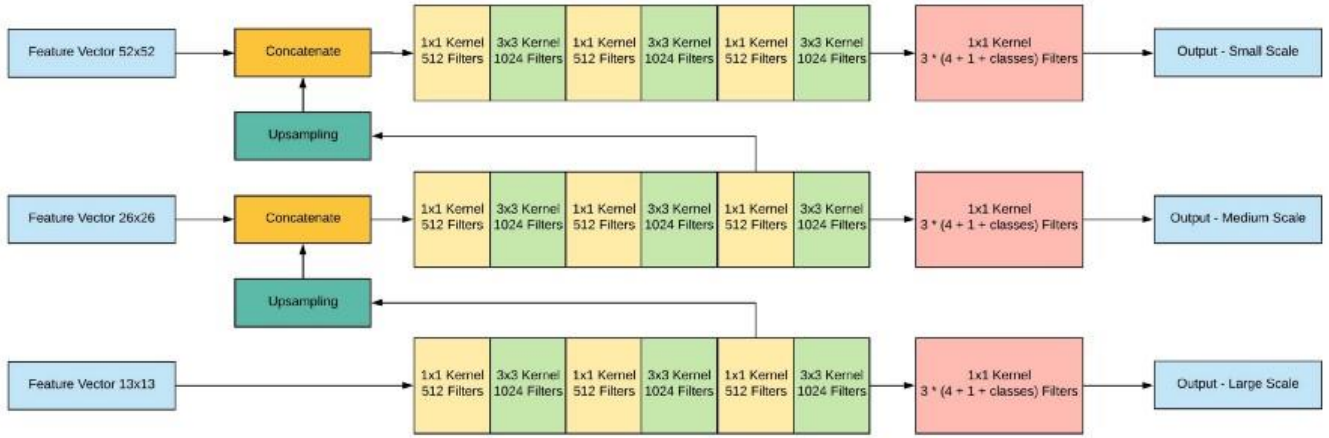
  Darknet-19 -> Darknet-53

- Class Prediction

  softmax -> binary cross-entropy loss

- Prediction Across Scales

  총 3개의 scale에서 3개의 feature map

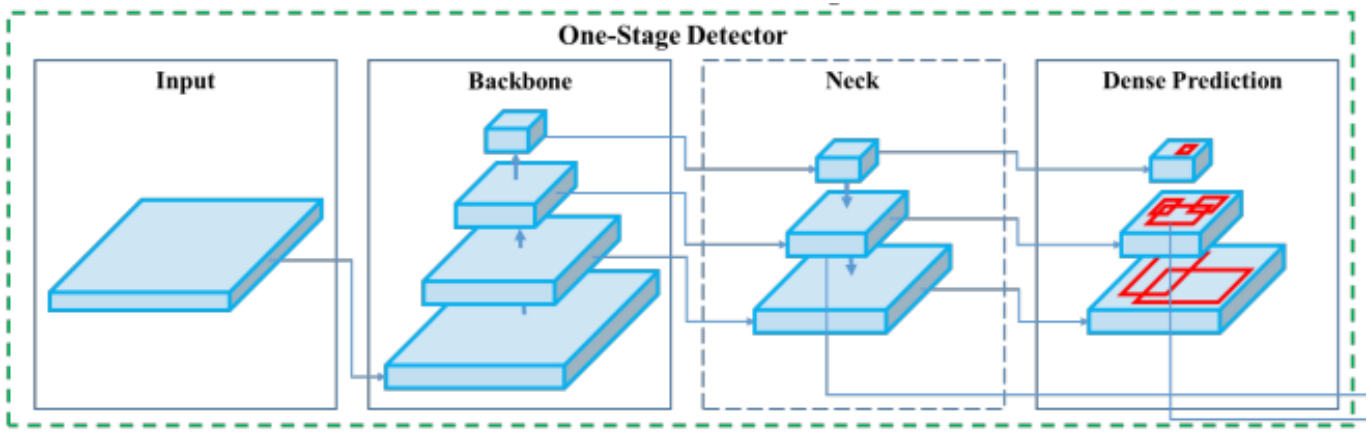


	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 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 v4

YOLO v4= YOLO v3 + CSPDarknet53+SPP,PAN+BoF+BoS

- Backbone: CSPDarknet53
- Neck: SPP, PAN
- Head: YOLO v3



- SPP: conv layer의 마지막 feature map을 고정된 크기의 grid로 분할한 후 평균을 구해 고정된 크기의 representation을 얻음
- PAN: 객체 탐지 시 localization 성능을 향상시킨 네트워크

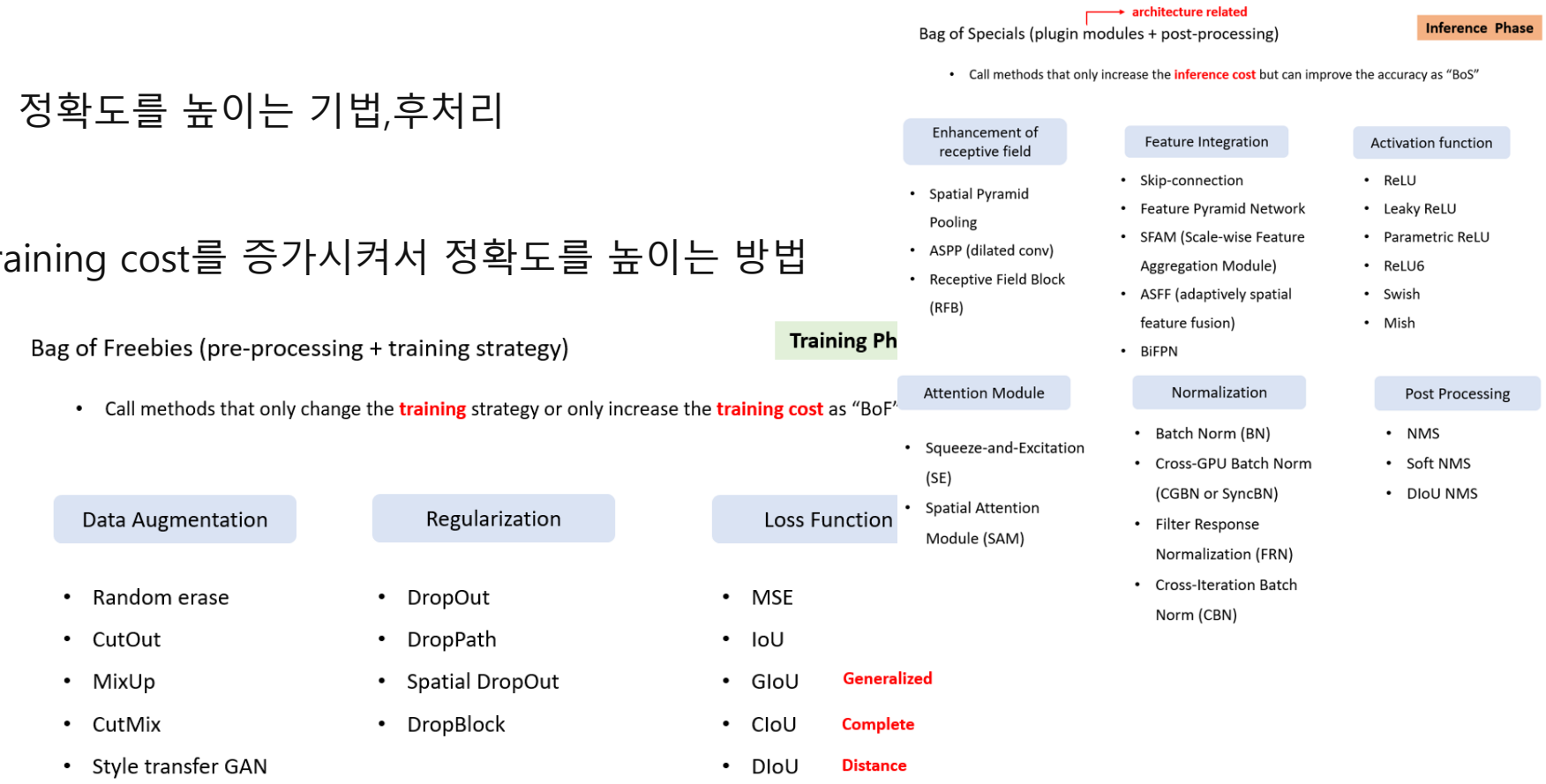
YOLO v4, v5

- BoS(Bag of Specials)

  inference cost만 증가시켜서 정확도를 높이는 기법,후처리

- BoF(Bag of Freebies)

  학습에 관여하는 요소로, training cost를 증가시켜서 정확도를 높이는 방법



➢ YOLO v5

파이토치로 구현

Q & A