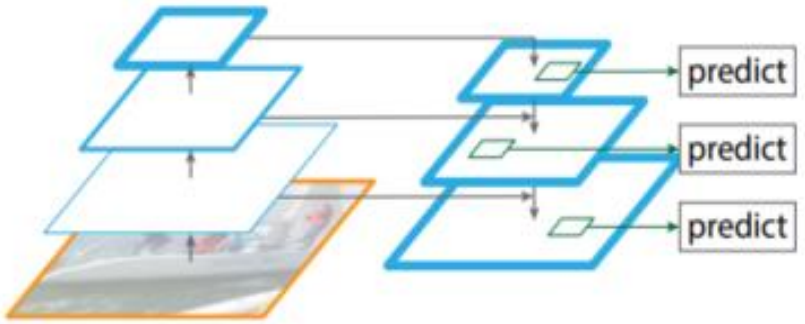
IML 세미나

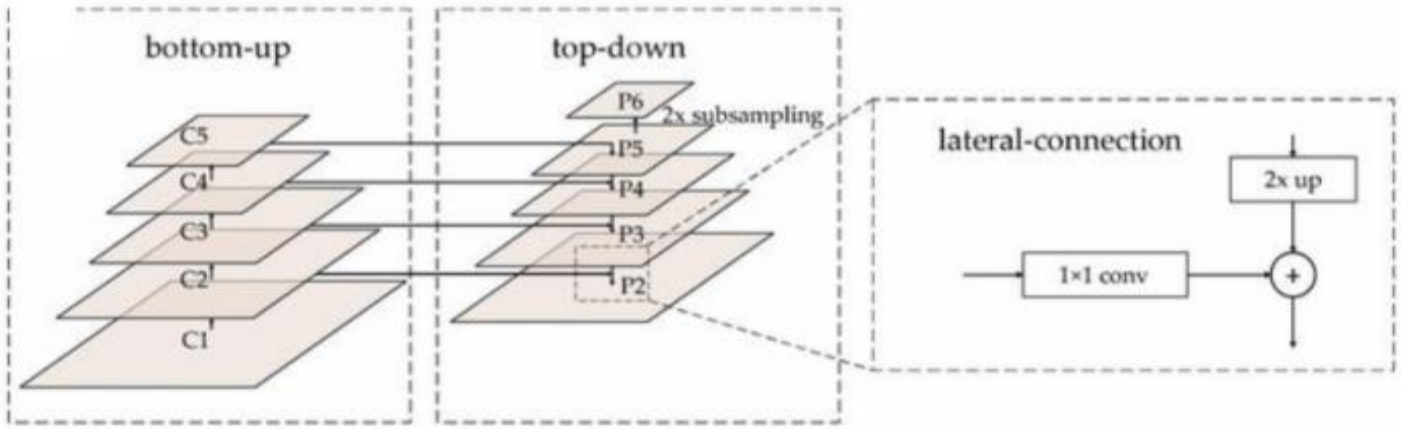
2021-12-28
문구영

Feature Pyramid Networks for Object Detection

Tsung-Yi Lin[1,2], Piotr Dollár[1], Ross Girshick[1],
Kaiming He[1], Bharath Hariharan[1], and Serge Belongie[2]

[1]Facebook AI Research (FAIR)
[2]Cornell University and Cornell Tech



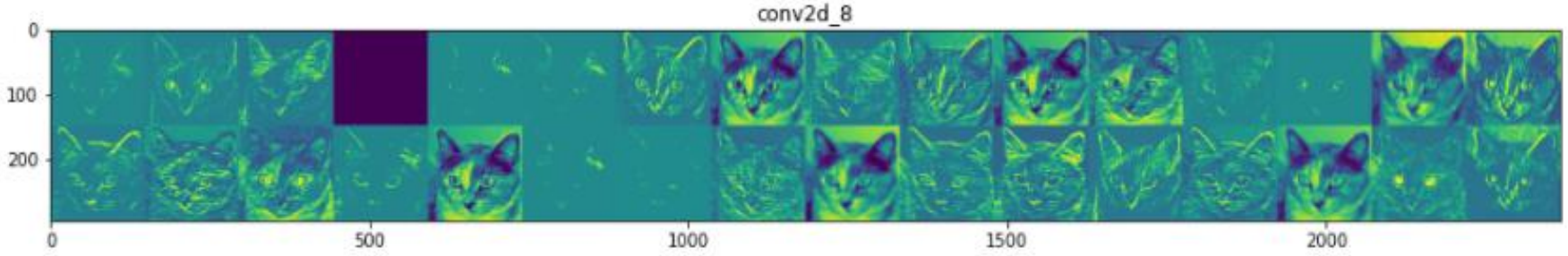


- 컴퓨팅 자원을 적게 차지하면서 **다양한 크기의 객체를 인식**할 수 있도록

- 단일 입력에 대한 **multi scale feature map** 출력 (bottom-up)
- 상위 layer에서 부터 feature map을 결합 (Top-down with lateral connection)

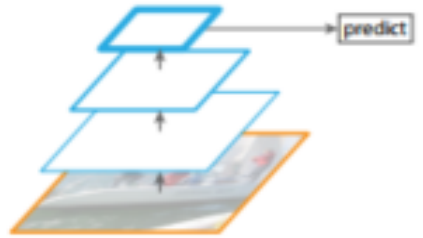
MOTIVATION 1 - CNN의 특징

1. Convolutional layers

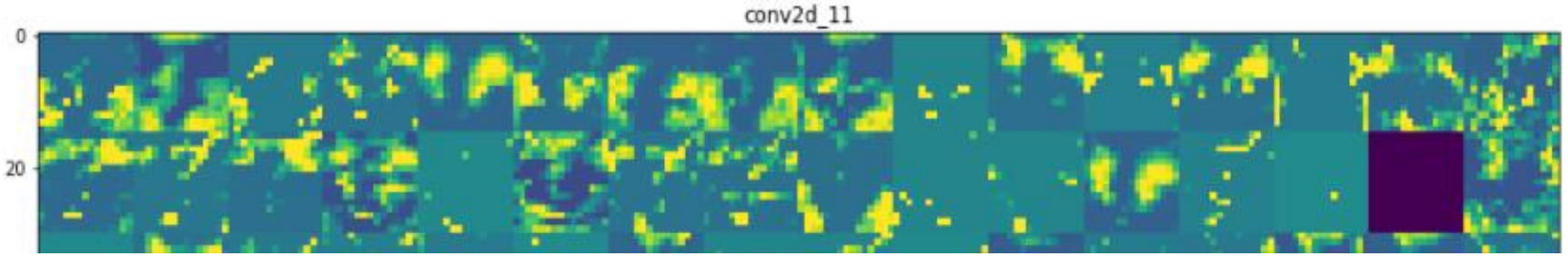


하위 층의 feature map 출력

- 노드, 에지 등의 일반적인 특징, 초기 입력 이미지의 정보를 대부분 보존
- **High resolution & low level features** (low semantic)



(b) Single feature map



상위 층의 feature map 출력

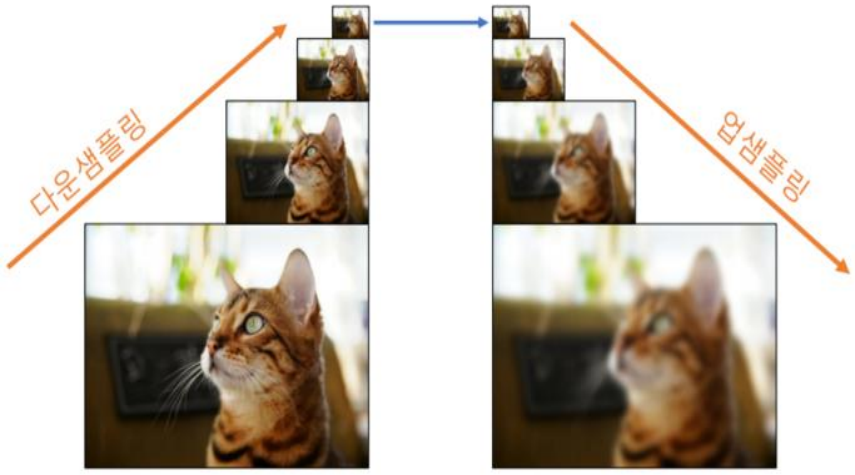
- 고양이 귀, 눈 등 추상화된 특징, 이미지 클래스에 특화된 정보
- **Low resolution & high level features** (high semantic)

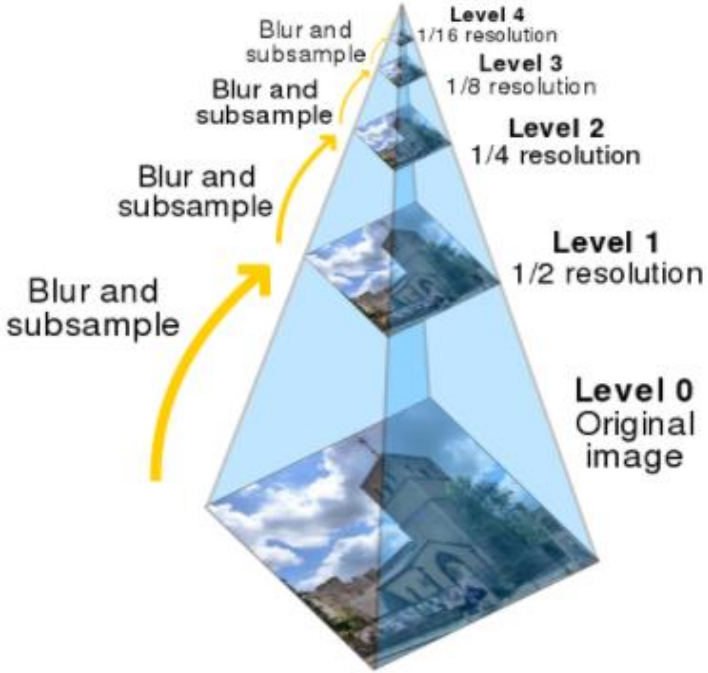
→ 최상위층의 feature map만 사용함으로써 해상도를 포기하게 됨 (작은 객체 인식의 어려움)

MOTIVATION 2 – 이미지 피라미드의 유용성

2. Image Pyramid

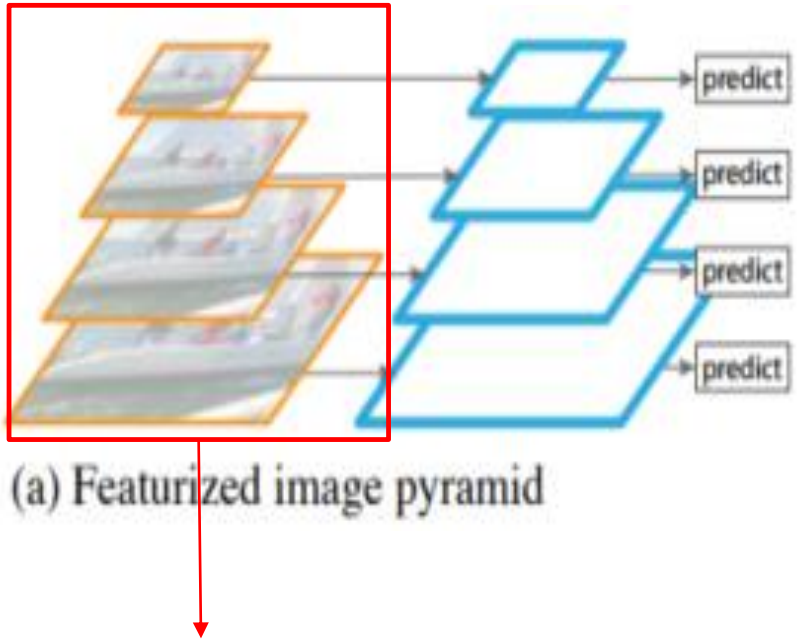
- 하나의 동일한 이미지에 대해 다양한 해상도, 스케일의 이미지 세트를 구성





MOTIVATION 2 – 이미지 피라미드 예시

Featurized Image Pyramid



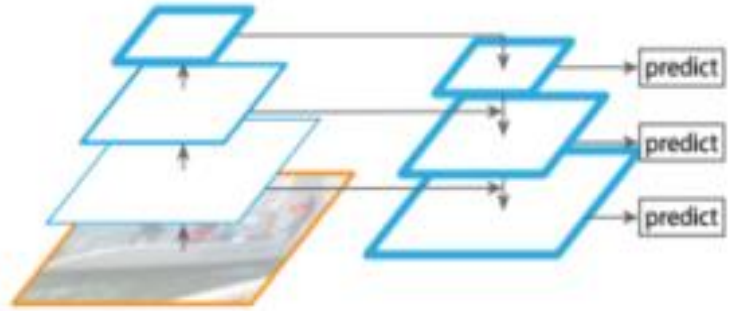
입력 이미지의 resize된 복사본들

- 다양한 해상도, 스케일의 이미지들을 CNN의 입력으로 받아 feature map 획득
  → **다양한 크기의 feature map**을 통해 크고 작은 객체들의 검출이 가능
- 연산량, 시간 소모 큼 (현실적으로 적용하기 어려움)

PROPOSAL

1. Feature pyramid를 사용하여 **작은 객체에 대한 정보를 소실하지 않음**으로써 (고해상도의 feature map 활용), 객체 인식의 성능과 정확도를 높이고

2. 작업 수행에 필요한 **추가적인 시간 비용은 최소한으로** 하는 아키텍쳐 제안



(d) Feature Pyramid Network

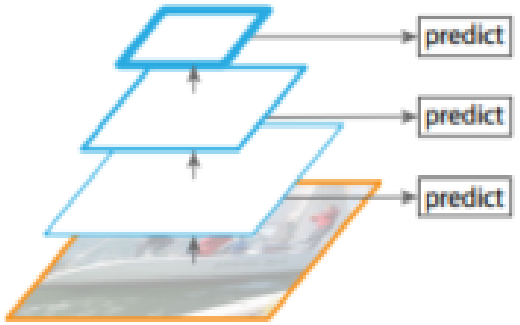
비용 문제를 최소화 하면서 다양한 크기의, 피라미드 계층 구조를 갖는 딥러닝 convolutional 네트워크 제안

→ 다양한 컴퓨터 비전 영역에서 사용되는 feature extractor로써 유의미한 성능 향상을 유도

ex) multi scale object detection, Fast R CNN

NOVELTY – 기존의 피라미드 구조와 비교

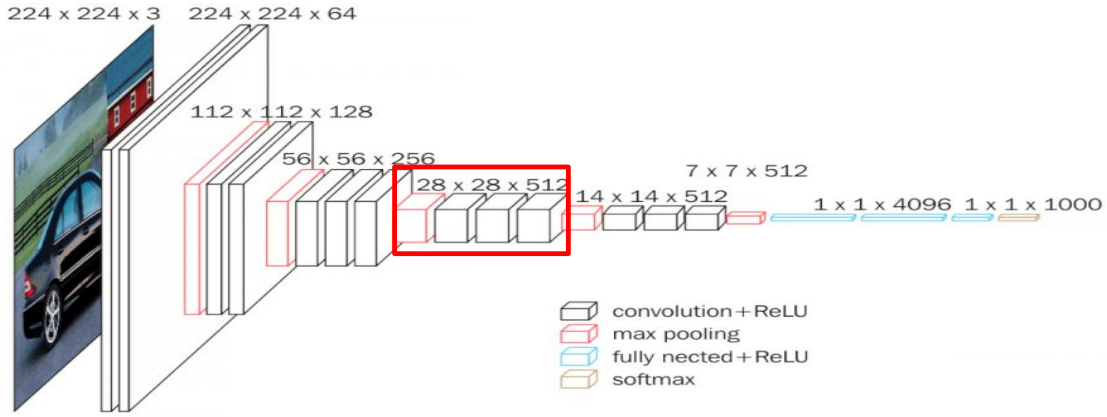
SSD style Pyramid – Pyramidal feature hierarchy



(c) Pyramidal feature hierarchy

- 순전파로 계산된 여러 계층의 **multi scale feature map**을 재사용 (비용 문제 해결)
- 중간 과정에서 생성되는 feature map 각각에 **독립적으로** 객체 인식 수행

- VGG net의 conv4_3 (10층) 이후의 feature map만 사용
  → **고해상도의 feature map이 갖는 정보를 사용하지 못함**
- 각 계층의 feature map들 간의 **semantic gap**이 큼



NOVELTY - FPN

Feature Pyramid Network

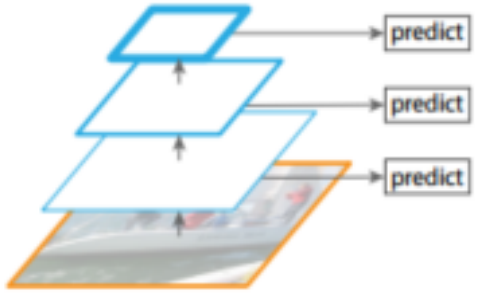
- CNN feature map들의 pyramidal hierarchy를 유지

  → **저해상도, 고수준 feature map과 고해상도, 저수준 feature map 결합** (고해상도 feature map 미사용 문제 해결)

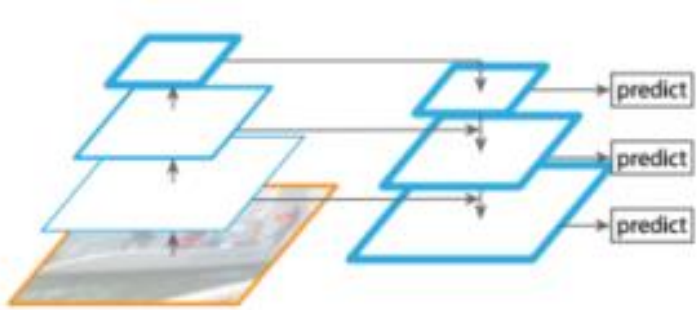
  → 피라미드를 구성하는 모든 feature map들은 **스케일, 단계에 상관없이 semantically strong** (semantic gap 해결)

- Feature fusion을 통해 forward 과정에서 손실된 지역 정보들을 보충

  → **스케일 변화에 강인**



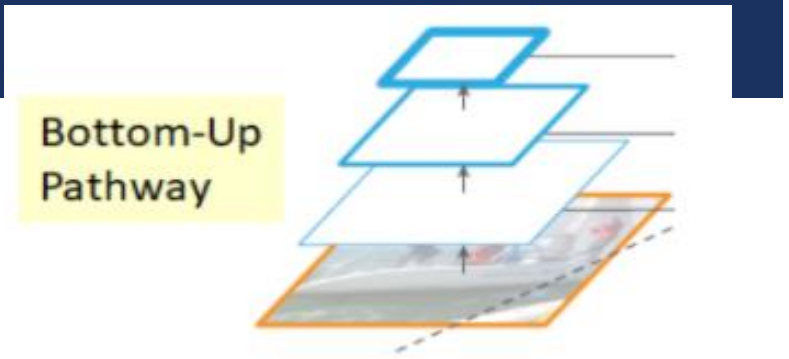
(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

Speed, memory, power 측면에서 효율적

FPN - ARCHITECTURE

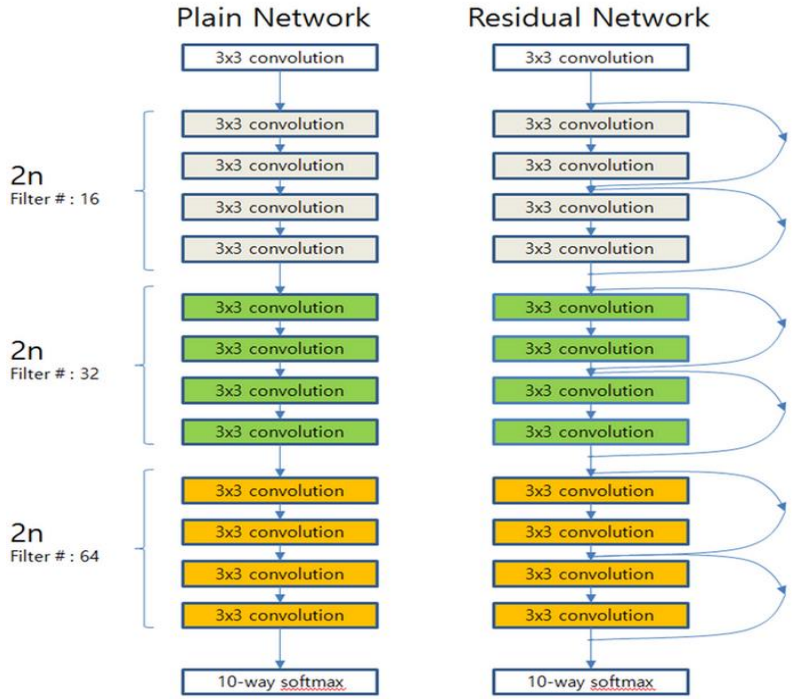


1. Bottom-up pathway

- Backbone CNN의 **순전파** 연산 결과 저장 (**ResNet** 활용)
- 다양한 scale의 feature map들로 구성된 **feature hierarchy** 구축
- 동일한 크기의 출력을 갖는 CNN 연산들을 하나의 stage로 묶고, 각 stage 마지막 layer의 출력 추출

ResNet

- 개별 conv layer의 feature map을 **재사용**
- Feature map 결합 (skip-connection)

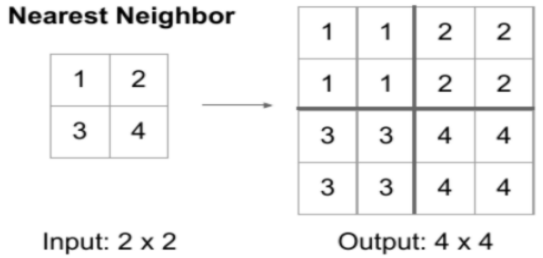


layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2
	1×1	
FLOPs		1.8×10^9

FPN - ARCHITECTURE

2. Top-down pathway

각 계층마다 다른 크기를 갖는 feature map들을 결합하기 위한 처리 과정



① 상위 feature map을 **nearest neighbor upsampling** 기법으로 해상도를 2배씩 증가

② 1x1 conv 연산으로 채널수를 감소 시킴

→ 차원 축소 (중요한 특징에 대한 weight 값을 어느 정도 보존하면서)

50-layer	101-layer
7×7, 64, stride 2	
3×3 max pool, stride 2	
[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4
[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23
[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3

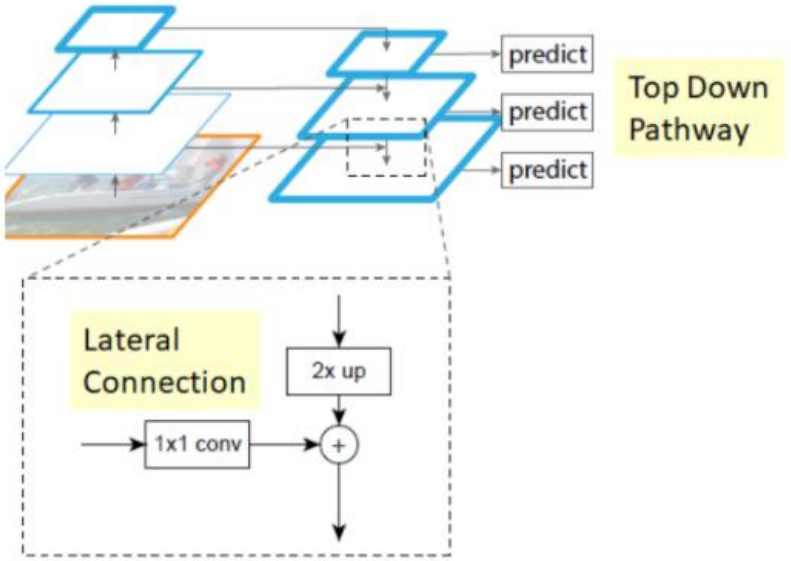
* 1x1 conv

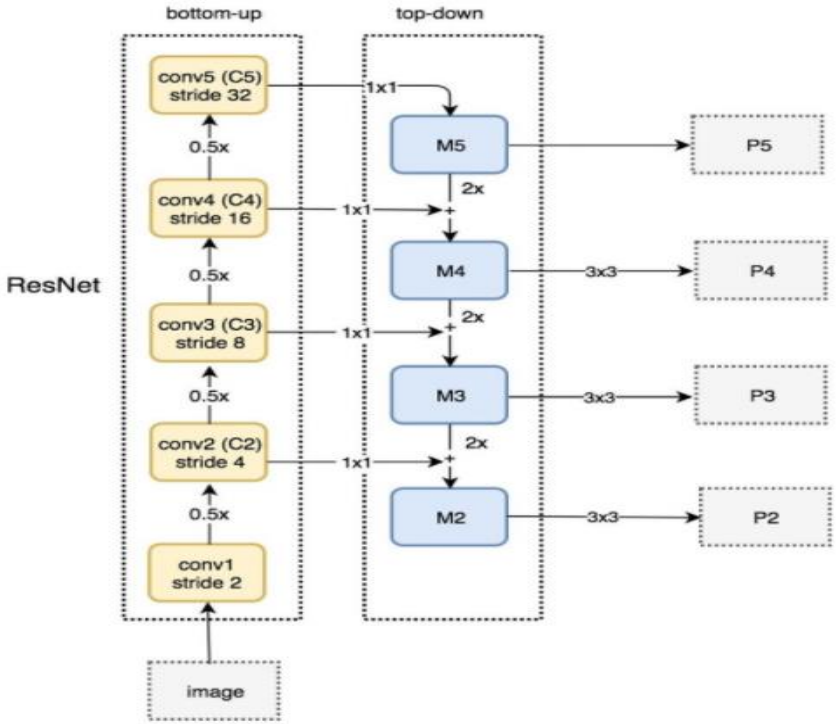
- 1x1x원하는 채널 수
- **차원 축소** : 입력 받는 것에 비해 적은 수의 채널 -> 차원 축소된 정보로 연산량을 감소 (보다 깊은 네트워크 설계 가능)
- **Bottleneck** (> < 모양) : 1x1 conv로 채널을 감소시킨 후, 3x3 conv 연산을 수행하고, 다시 1x1 conv로 차원을 깊게 만듬

FPN - ARCHITECTURE

3. Lateral connections (Feature fusion)

- Bottom-up pathway와 top-down pathway에서 같은 크기의 **feature map을 결합 (Element wise addition, Top down enrichment)**
- 상위 layer의 feature map의 크기를 증가시키고 (up-sampling), 채널은 감소시켜서 (1x1 conv) 결합

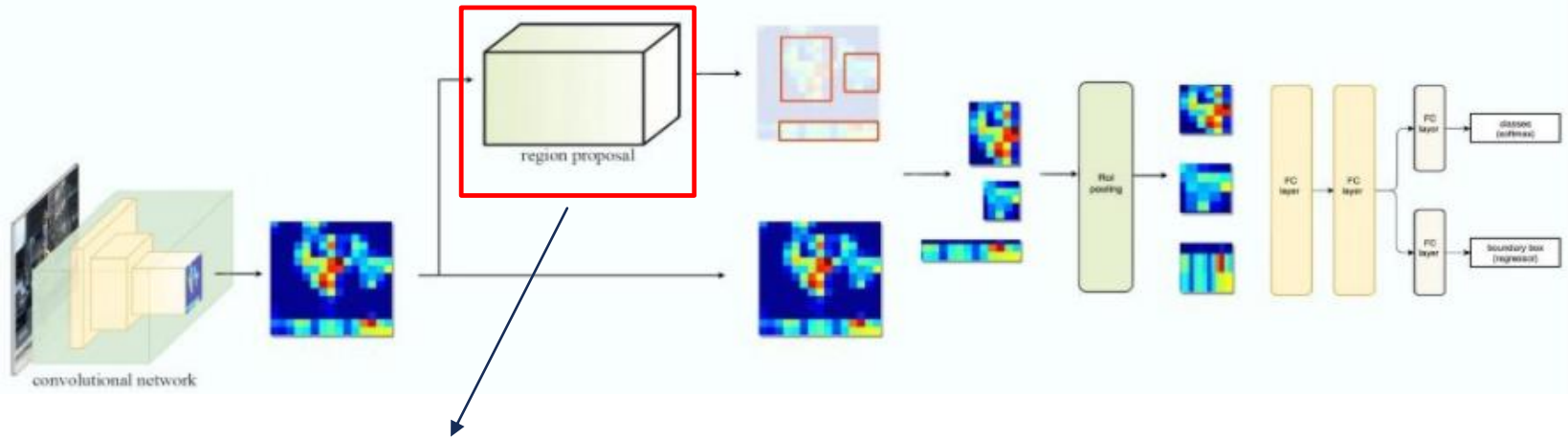




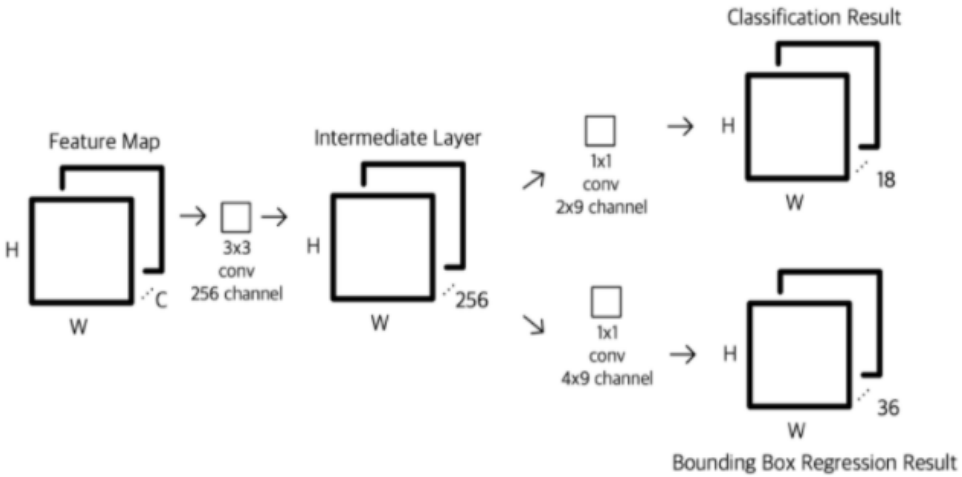
- 결합된 feature map에 3x3 conv를 적용시켜 최종 feature map 출력
(모두 동일한 256 채널)

APPLICATIONS IN RPN

Fast R CNN w/o FPN



단일 feature map에 RPN을 통해 얻은 ROI를 mapping



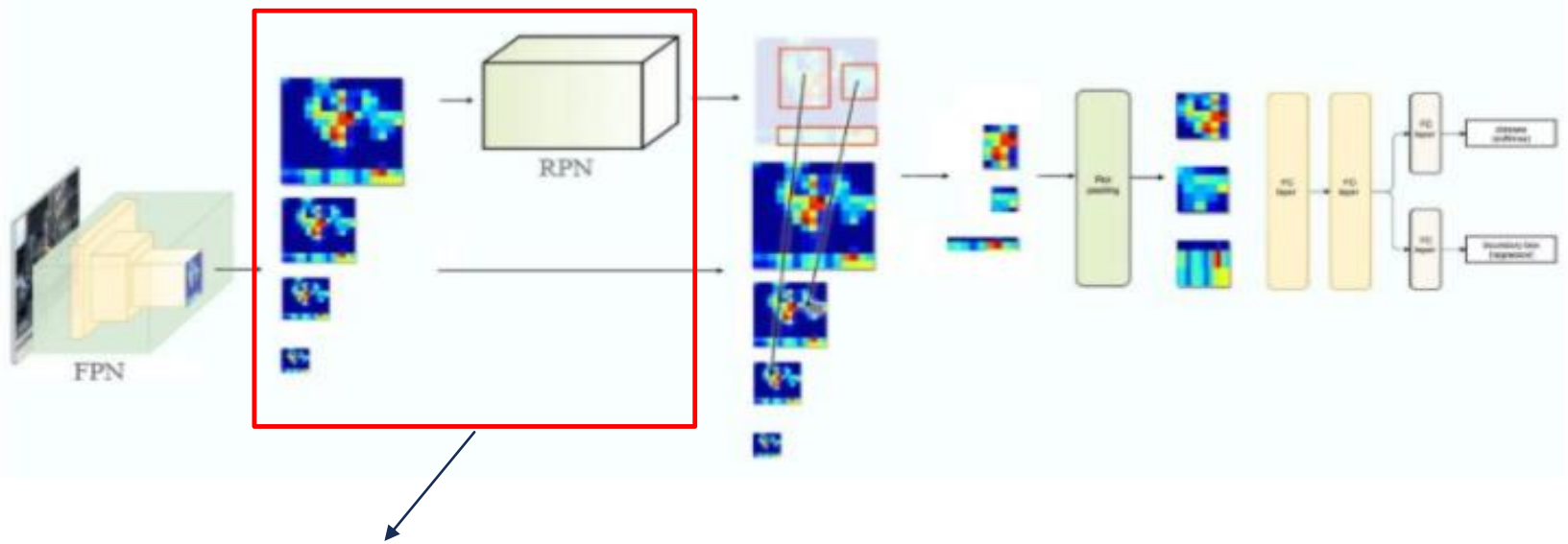
① 마지막 conv layer의 feature map에 3x3x256 conv 연산을 적용시켜 intermediate layer 생성

② Intermediate layer에 두 갈래의 1x1 conv를 적용시켜 이진 분류, bounding box regression 수행

APPLICATIONS IN RPN

Fast R CNN with FPN

- Region Proposal Network 부분에 FPN을 적용



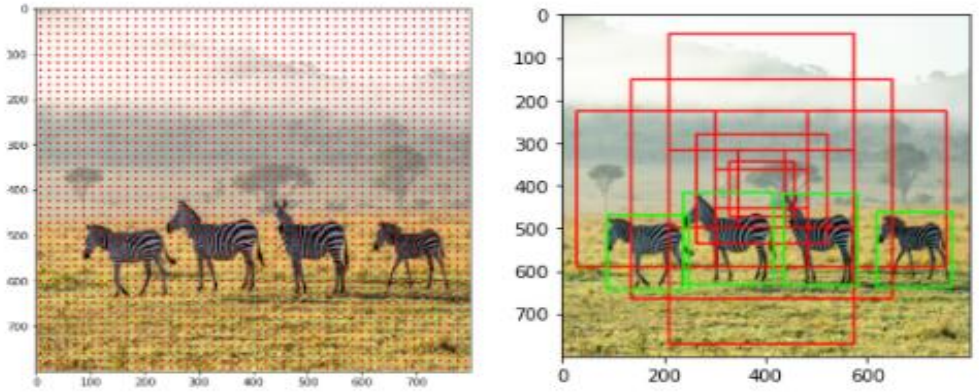
다양한 크기의 feature map 존재
ROI 크기에 따라 mapping할 feature map을 결정하는 수식 제안

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor.$$

- 각 단계 별 feature map에 3x3 conv, 두 갈래의 1x1 conv 적용

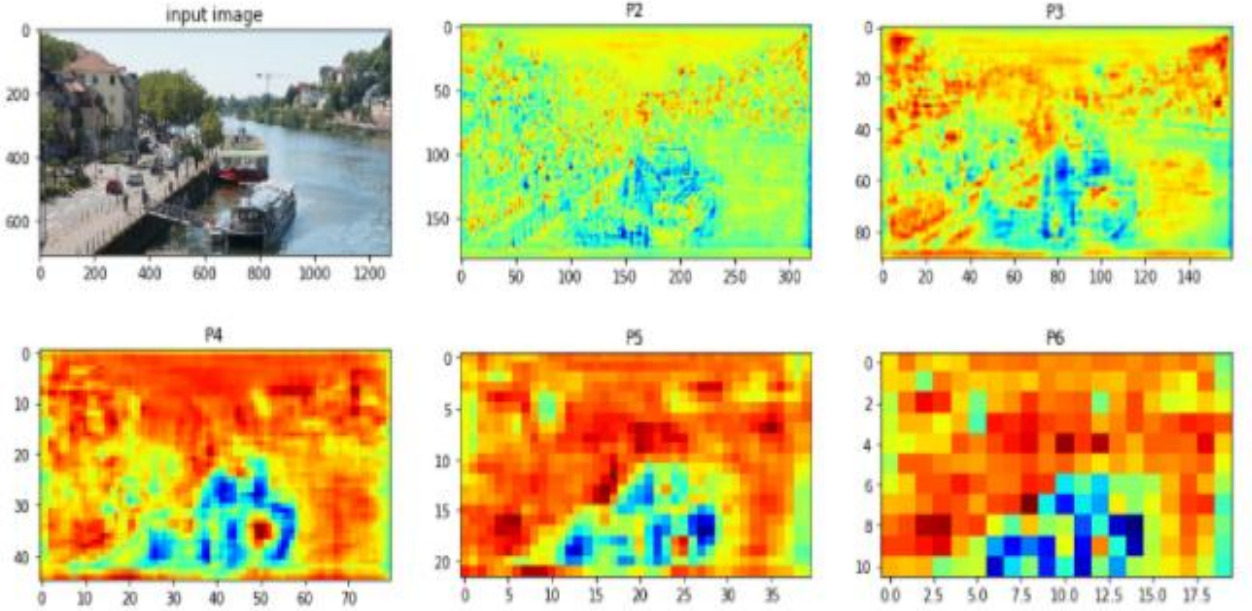
  - 다양한 크기의 feature map이 존재하기 때문에 anchor 사용 x (P2…P5가 각각 특정한 크기의 anchor box를 의미)

APPLICATIONS IN RPN



- 다양한 스케일의 객체를 검출하기 위해 비율, 스케일이 서로 다른 9개의 anchor box 이용

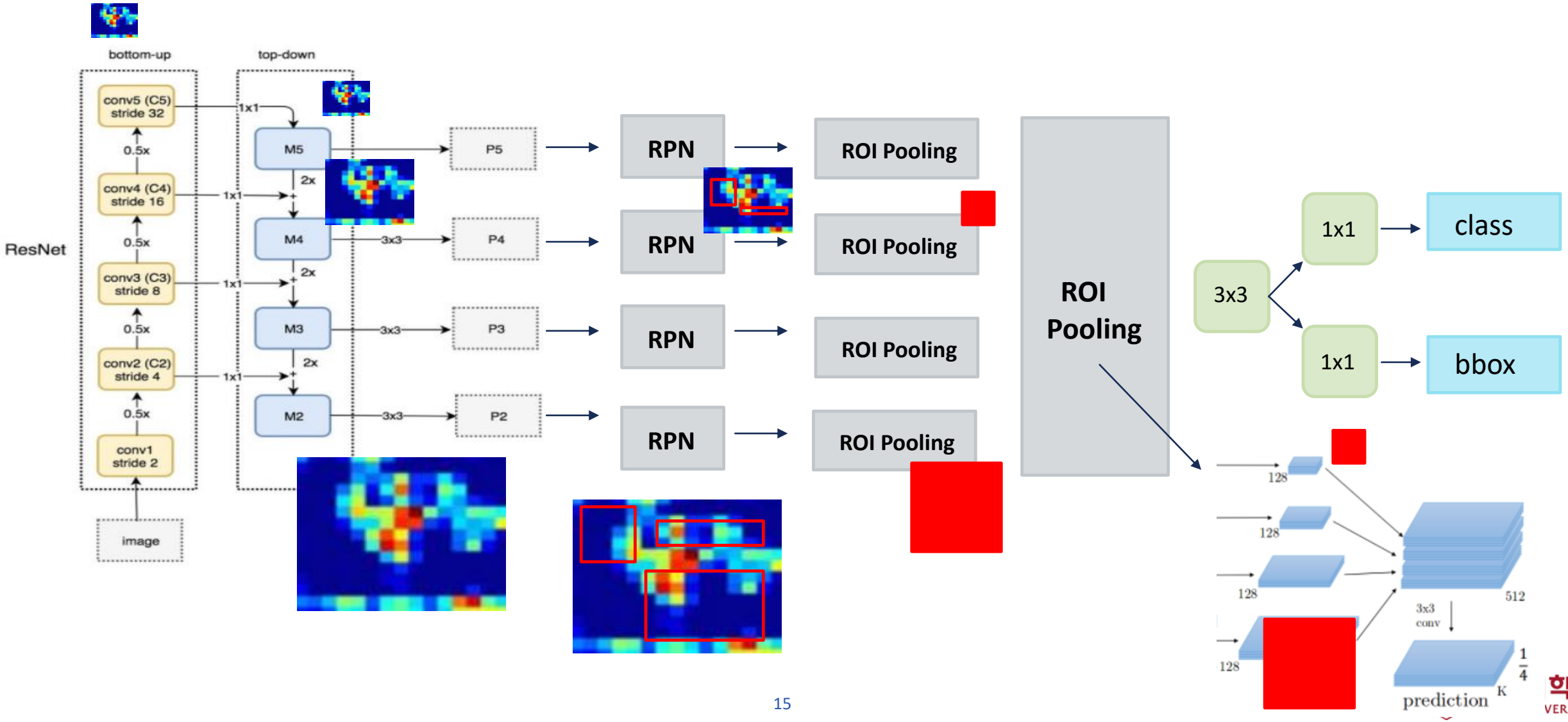
Output features of FPN



- 상위 feature map P5은 크기가 큰 객체에 대한 정보 보유 (큰 anchor box 역할)
- 하위 feature map P2는 작은 객체에 대한 정보 보유 (작은 anchor box 역할)

→ **FPN에서는 multi scale feature map이 anchor 역할 대체**

APPLICATIONS – FAST R CNN



EXPERIMENTAL RESULTS

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

Fast R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(a) baseline on conv4	RPN, $\{P_k\}$	C_4	conv5			54.7	31.9	15.7	36.5	45.5
(b) baseline on conv5	RPN, $\{P_k\}$	C_5	*2fc*			52.9	28.8	11.9	32.4	43.4
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	**45.8**
Ablation experiments follow:										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	*2fc*	✓	✓	56.3	33.4	17.3	37.3	45.6

Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16]†	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	*2fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

SUMMARY

Overview

- **접근 방식** : 순전파의 중간 단계 feature map 결과들을 활용하여 pyramidal hierarchy 구축
- **목표** : **다양한 크기의 객체를 보다 정확히 인식**할 수 있도록 (시간 비용을 최소화 하면서)

Key insight

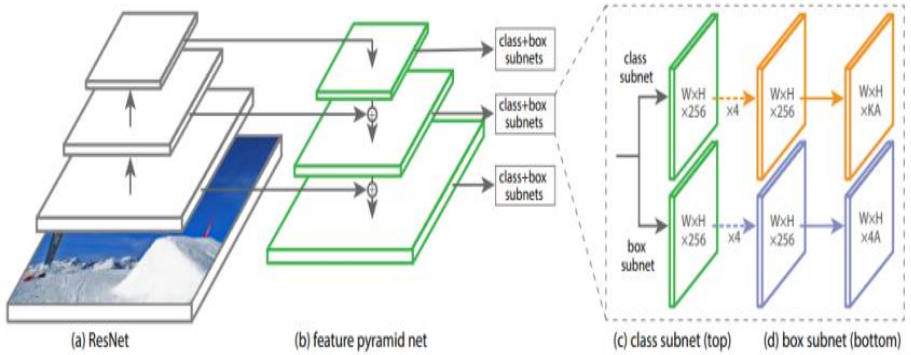
1. **Top down enrichment** : 상위 layer의 feature map을 결합하는 방식을 통한 feature fusion & enrichment
2. RPN : 객체 검출 모델의 RPN 단계에서 유용하게 활용
3. Anchor box 대체 : 재사용되는 다양한 크기의 feature map들이 anchor box 대체

Contributions

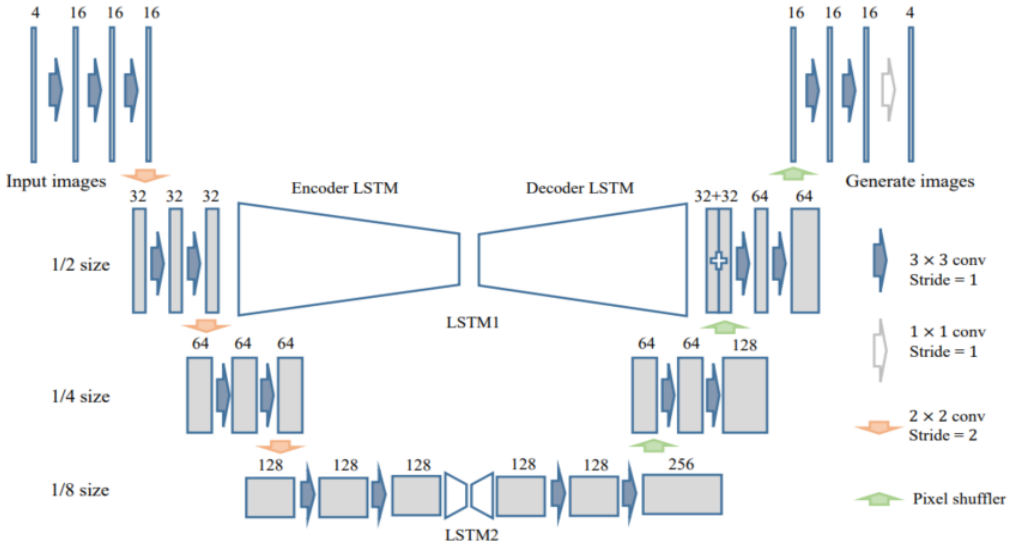
1. 직관적이면서도 범용적으로 적용 가능한 기법
2. 이후 많은 모델들에게 채택 받아 사용 ex) YOLO v3, U-Net, PVT

SUMMARY

YOLO v3



U-Net



Pyramidal Vision Transformer