Hyogeun Oh
BOAZ Data Analysis Dept. 19th
ohg3417@gmail.com

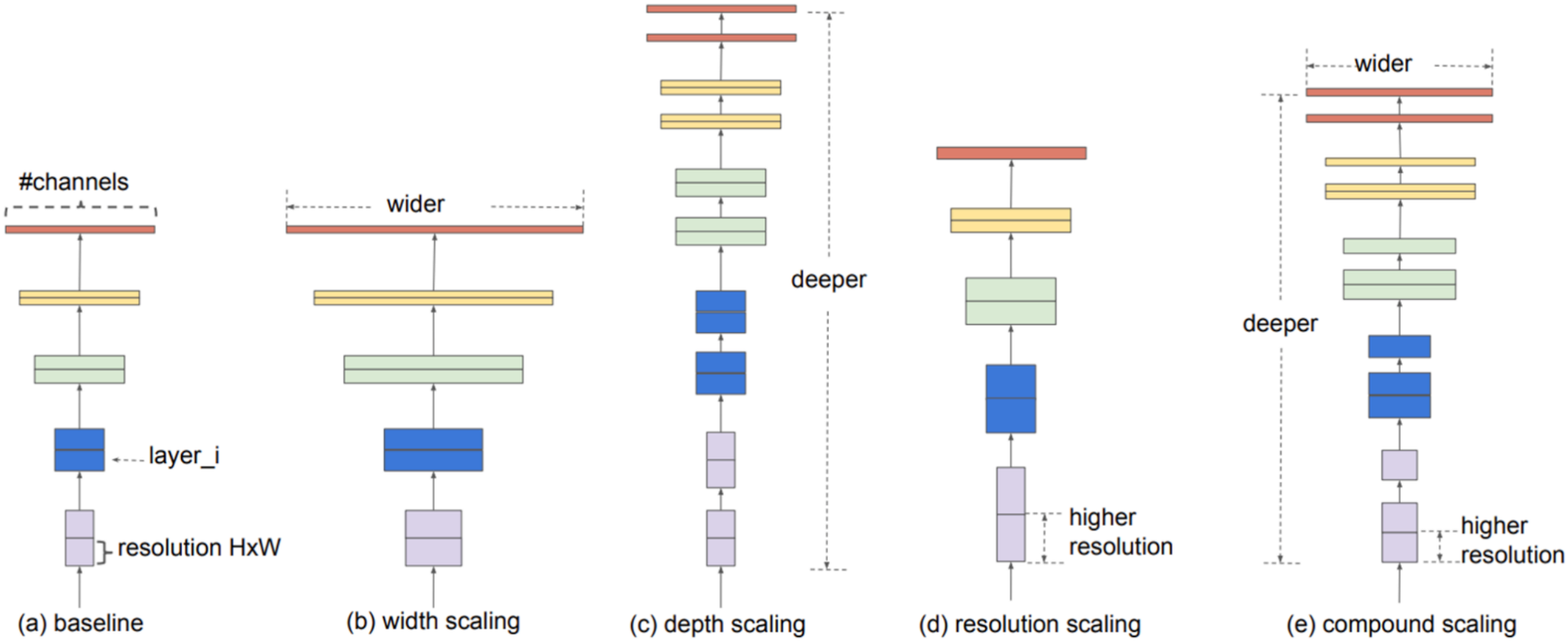
EfficientNet/Det & Advanced Object Detection

2023.02.21

Table of Content

EfficientNet

- 기존 연구의 한계점
  - ConvNet가 더 깊어짐에 따라 높은 정확도를 획득할 수 있음
  - 하지만 실제 real-time 적용에 한계점이 존재하기 때문에 빠르고 작은 모델을 위해 EfficientNet 제안
- Efficient in Object Detection?
  - Width Scaling
  - Depth Scaling
  - Resolution Scaling
  - Compound Scaling



Accuracy & Efficiency

- Accuracy: 최적의 d, w, r 모색
- Efficiency: 모델의 최대 memory 및 flops 한계 설정

$$\max_{d,w,r} \quad Accuracy\left(\mathcal{N}(d,w,r)\right)$$

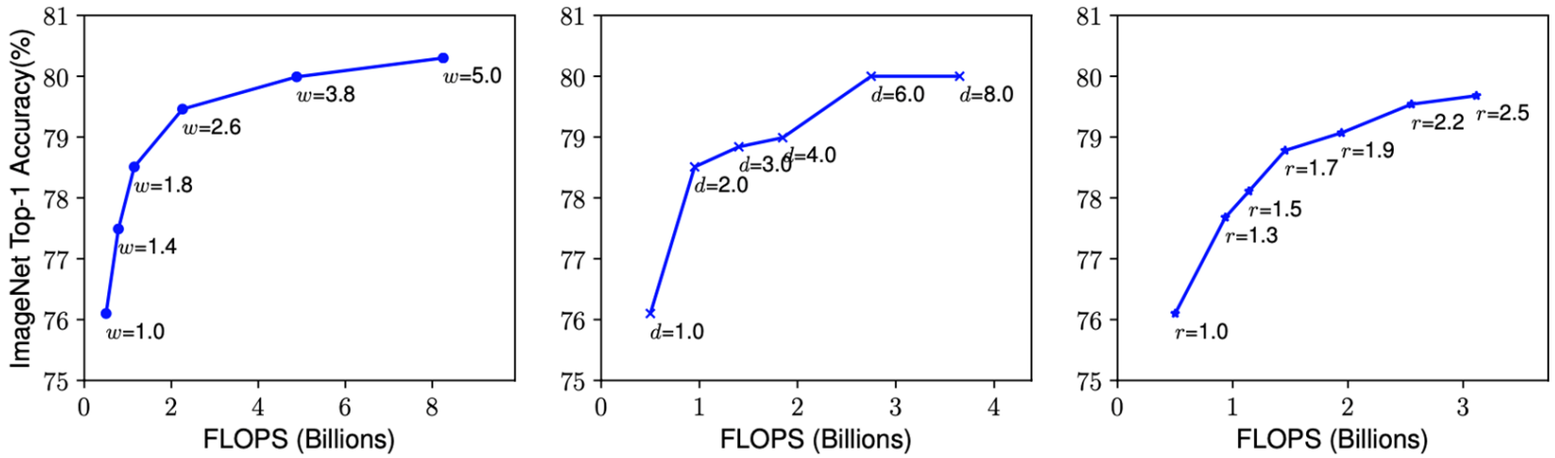
$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$

$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$

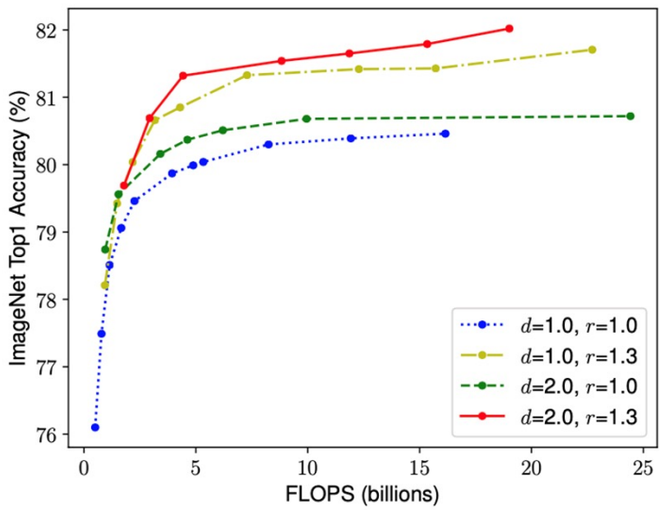
$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

Observation

- Observation 1
  - 네트워크의 d, w, r 증가에 따라 정확도 향상
  - 하지만 일정 수치 이상 증가 시 증가 폭 감소



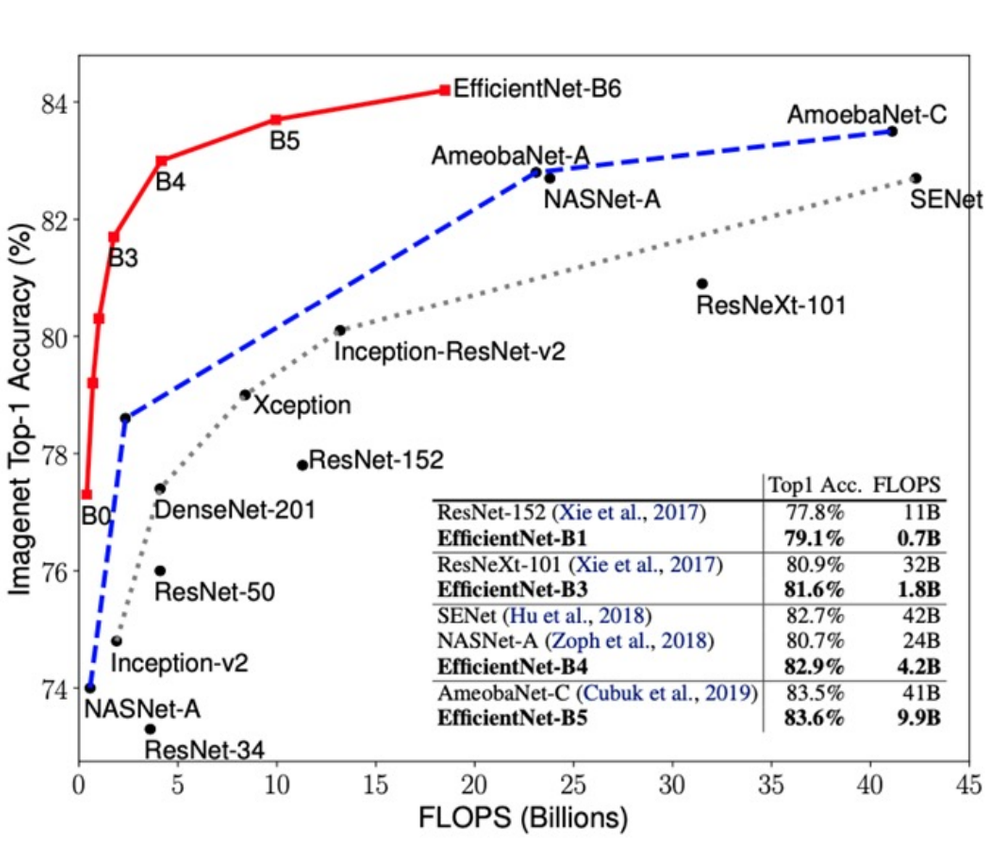
- Observation 2
  - ConvNet 스케일링 과정에서 d, w, r의 최적 균형을 모색해야함



Methodology & Results

- Step 1
  - $\emptyset = 1$ 고정
  - Small grid search 기반 최적 α, β, γ 모색
  - $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$ under constraint of $\alpha^2 \cdot \beta^2 \cdot \gamma^2 \approx 2$
- Step 2
  - α, β, γ 고정
  - $\emptyset$를 통한 scale up

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

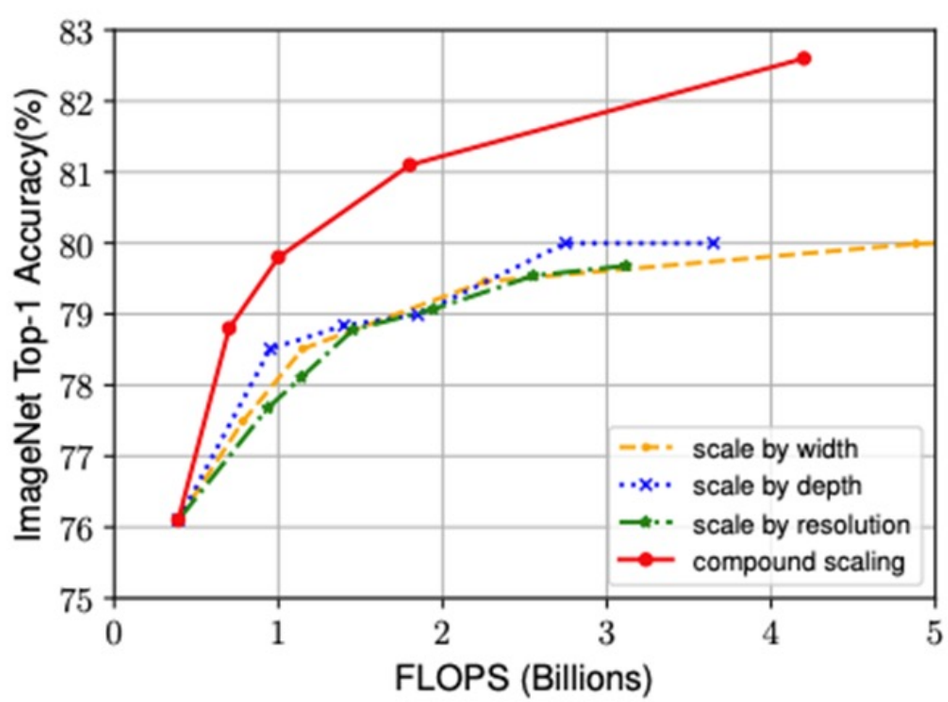
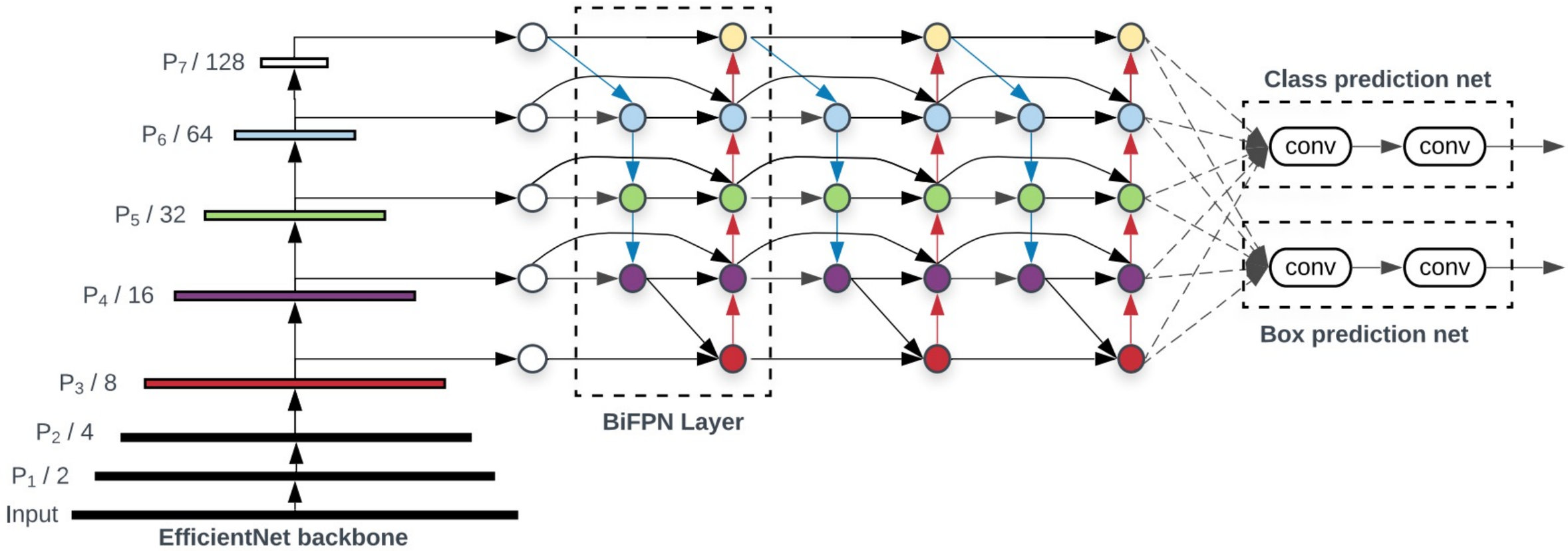


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

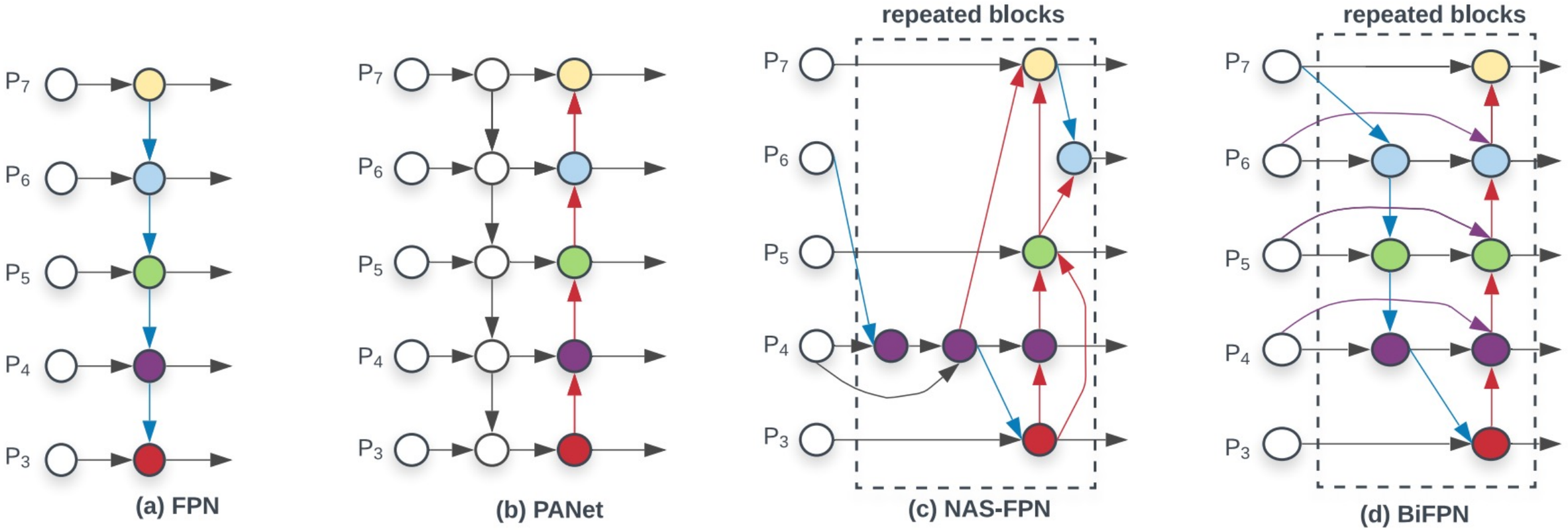
EfficientDet

- 기존 연구의 한계점
  - 속도를 개선하기 위해 다양한 시도가 이루어졌지만 정확도 저조



Efficient Multi-Scale Feature Fusion

- Conventional Multi-Scale Feature Fusion
  - FPN
  - PANet
  - NAS-FPN
- Proposed Method: BiFPN (Bi-directional Feature Pyramid Network)
  - Resolution를 고려하기 위해 학습 가능한 가중치를 정의한 weighted feature fusion 방법
  - 효율성 향상을 위해 cross-scale connections 방법 이용



Model Scaling

- 더 좋은 성능을 위해서는 더 큰 Backbone 모델을 사용하여 Detector의 크기를 키우는 것이 일반적

- 최적의 정확도와 효율을 지닌 모델을 개발하기 위해 EfficientNet과 같은 Compound Scaling 제안
  - Backbone: EfficientNet B0 ~ B6
  - BiFPN Network: 네트워크의 width와 depth를 compound 계수에 따라 증가

$$W_{bifpn} = 64 \cdot \left(1.35^{\emptyset}\right), \qquad D_{bifpn} = 3 + \emptyset$$

  - Box/Class Prediction Network: Width는 고정, depth는 식에 따라 증가

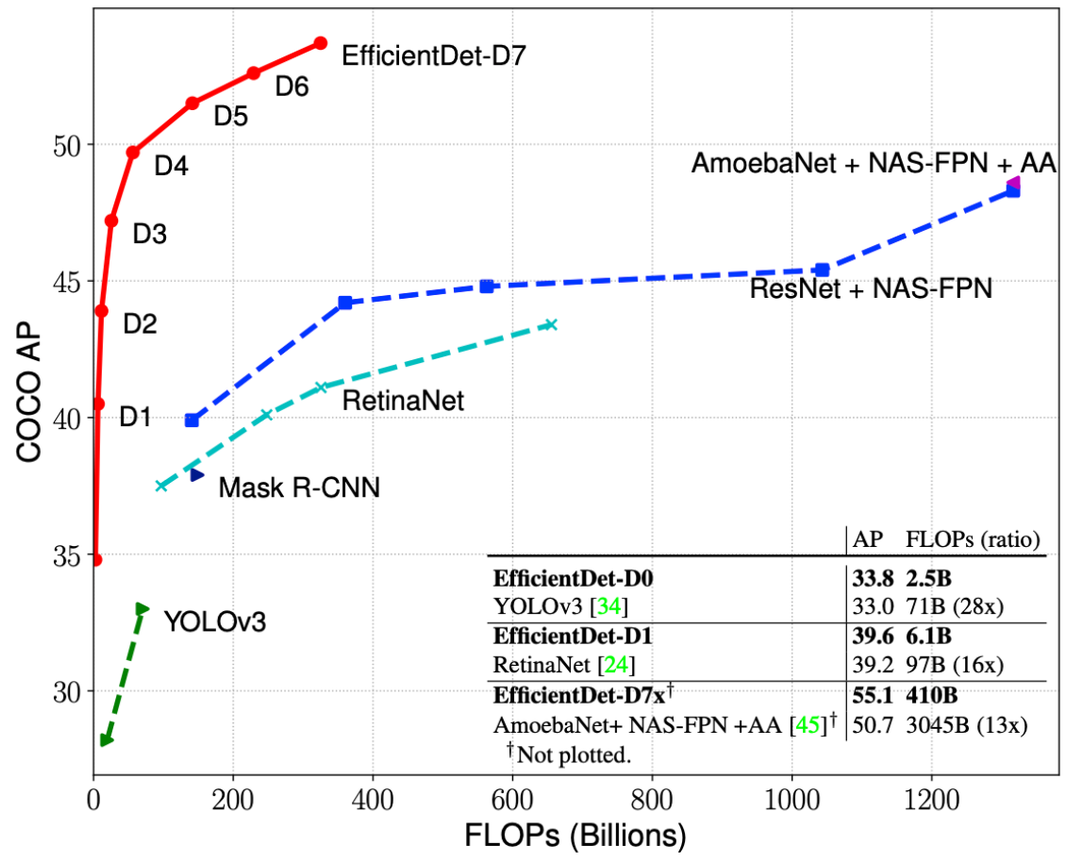
$$D_{box} = D_{class} = 3 + [\emptyset/3]$$

  - Input Image Resolution: 식과 같이 선형적 증가

$$R_{input} = 512 + \emptyset \cdot 128$$

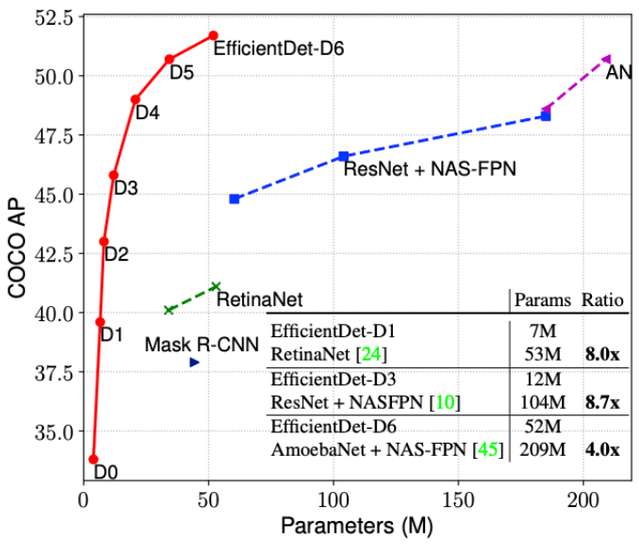
Results

- 기존 모델 대비 동일 조건에서 압도적 성능 보장



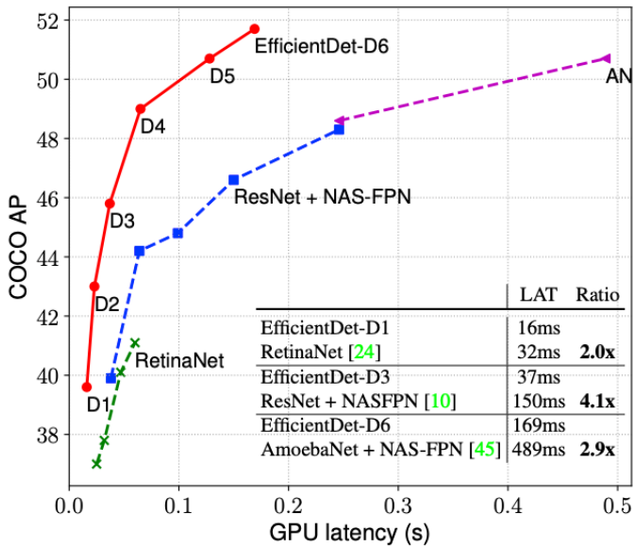
	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.



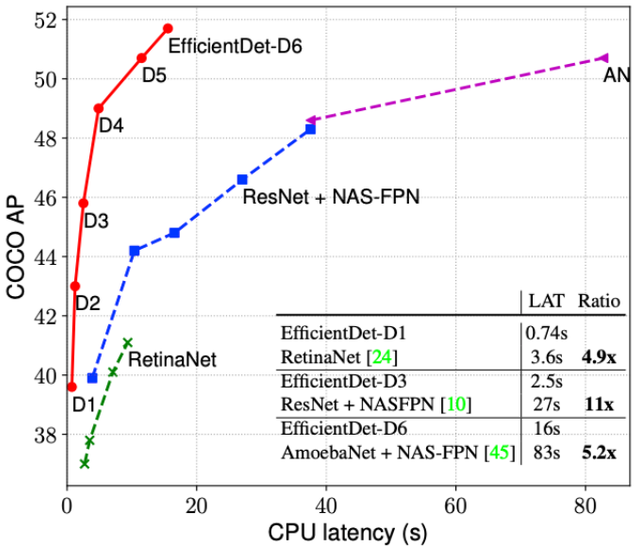
	Params	Ratio
EfficientDet-D1	7M	
RetinaNet [24]	53M	**8.0x**
EfficientDet-D3	12M	
ResNet + NASFPN [10]	104M	**8.7x**
EfficientDet-D6	52M	
AmoebaNet + NAS-FPN [45]	209M	**4.0x**

(a) Model Size



	LAT	Ratio
EfficientDet-D1	16ms	
RetinaNet [24]	32ms	**2.0x**
EfficientDet-D3	37ms	
ResNet + NASFPN [10]	150ms	**4.1x**
EfficientDet-D6	169ms	
AmoebaNet + NAS-FPN [45]	489ms	**2.9x**

(b) GPU Latency

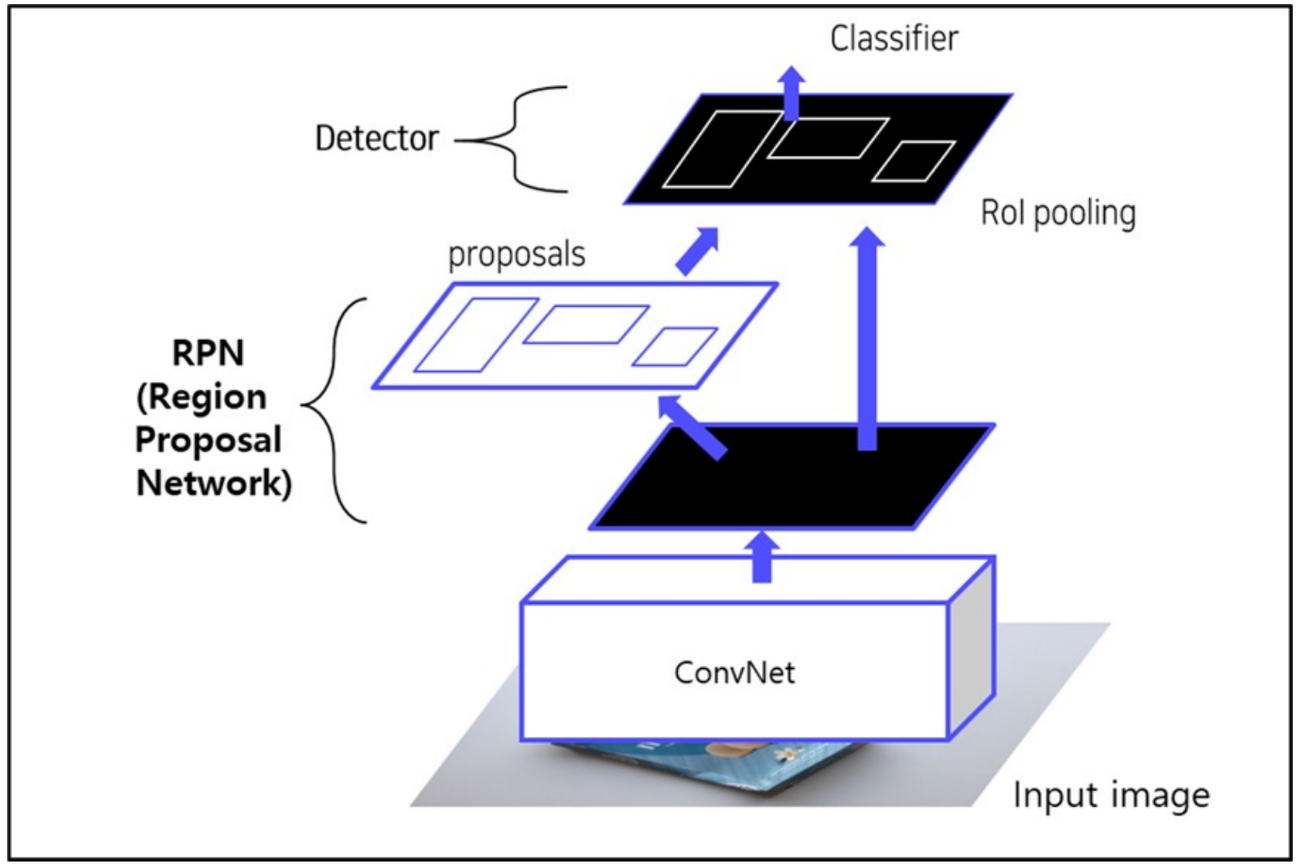


	LAT	Ratio
EfficientDet-D1	0.74s	
RetinaNet [24]	3.6s	**4.9x**
EfficientDet-D3	2.5s	
ResNet + NASFPN [10]	27s	**11x**
EfficientDet-D6	16s	
AmoebaNet + NAS-FPN [45]	83s	**5.2x**

(c) CPU Latency

Cascade R-CNN

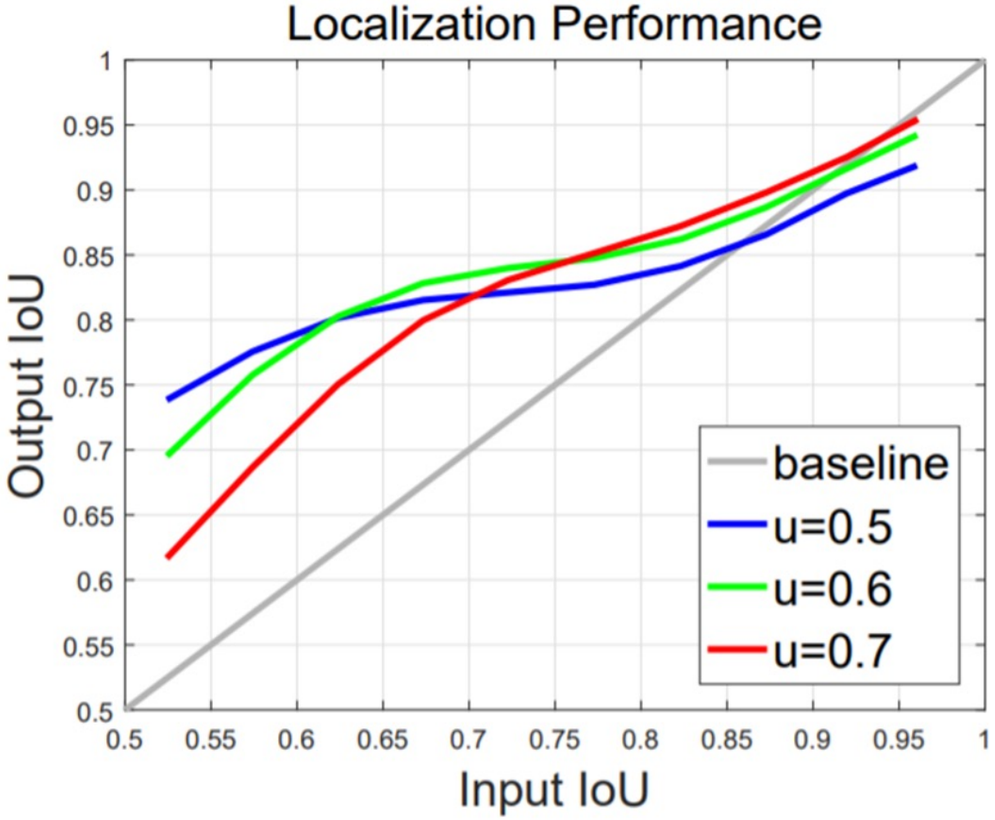
- 기존 R-CNN
  - Pipeline
    1. ConvNet: 입력 이미지에 대해 feature map 생성
    2. RPN Network: RoI 추출
    3. RoI Pooling: RoI를 feature map에 projection 후 최종 head에 입력될 feature map 생성
    4. Classification Head & Box Coordinate Head: 해당 RoI에 대해 객체의 종류 분류 및 위치 예측
  - Dataset
    - RPN: IoU threshold 기준으로 positive/negative 분류
    - Head Network: RoI threshold 기준으로 positive/negative 분류

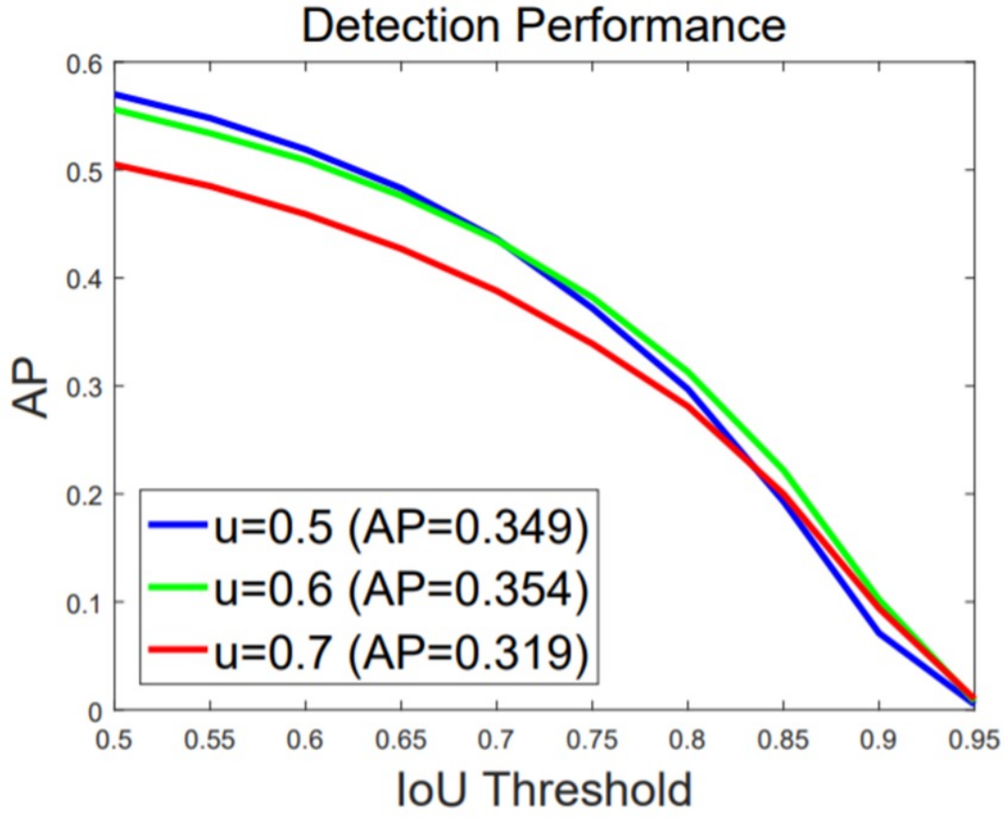


Performance

- Localization Performance
  - 높은 IoU threshold에서 학습된 model이 더 좋은 결과
- Detection Performance
  - AP의 IoU threshold 증가 시 0.6, 0.7로 학습된 model의 성능 우위

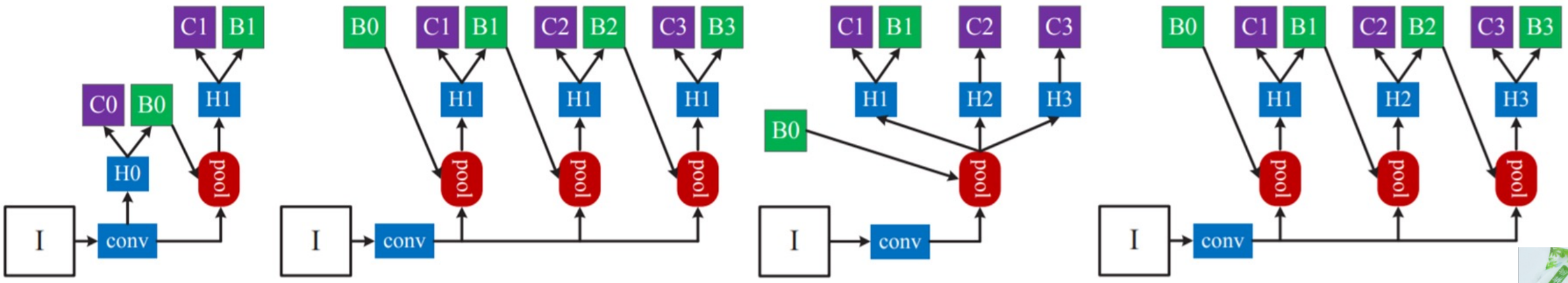
→ 학습되는 IoU에 따라 대응 가능한 IoU box가 다름





Methodology

- Faster R-CNN
  - I: Input image
  - conv: Backbone network
  - H0: Result of RPN
  - H1: Result of pooling
  - C0: Classification result (Background vs. Object)
  - C1: Object prediction result
- Iterative Bbox at Inference
  - B1: Result of Head
  - B2: B1을 다시 projection한 RoI로부터 head를 통과한 결과
  - Inference 단계에서 head로 예측된 Bbox를 다시 RoI proejcetion
  - RPN에 비해 더 정확한 박스로부터 head를 통과할 수 있지만 큰 성능 향상 X
- Integral Loss
  - IoU threshold가 다른 classifier 학습
  - Loss: Classifer들의 loss 합
  - Inference: Classifier들의 confidence 평균
  - 동일한 RPN 결과를 사용하여 성능 향상 X
- Cascade R-CNN
  - 여러 개의 RoI head 학습
  - 각 head의 IoU threshold를 다르게 설정
  - 최종 결과: C3, B3



(a) Faster R-CNN (b) Iterative BBox at inference (c) Integral Loss (d) Cascade R-CNN

Results

- Bbox Pooling 반복 수행 시 성능 향상 (Iterative)
- IoU Threshold가 다른 Classifier가 반복될 시 성능 향상 (Integral)
- IoU Threshold가 다른 RoI Head를 Cascade로 쌓을 시 성능 향상 (Cascade)

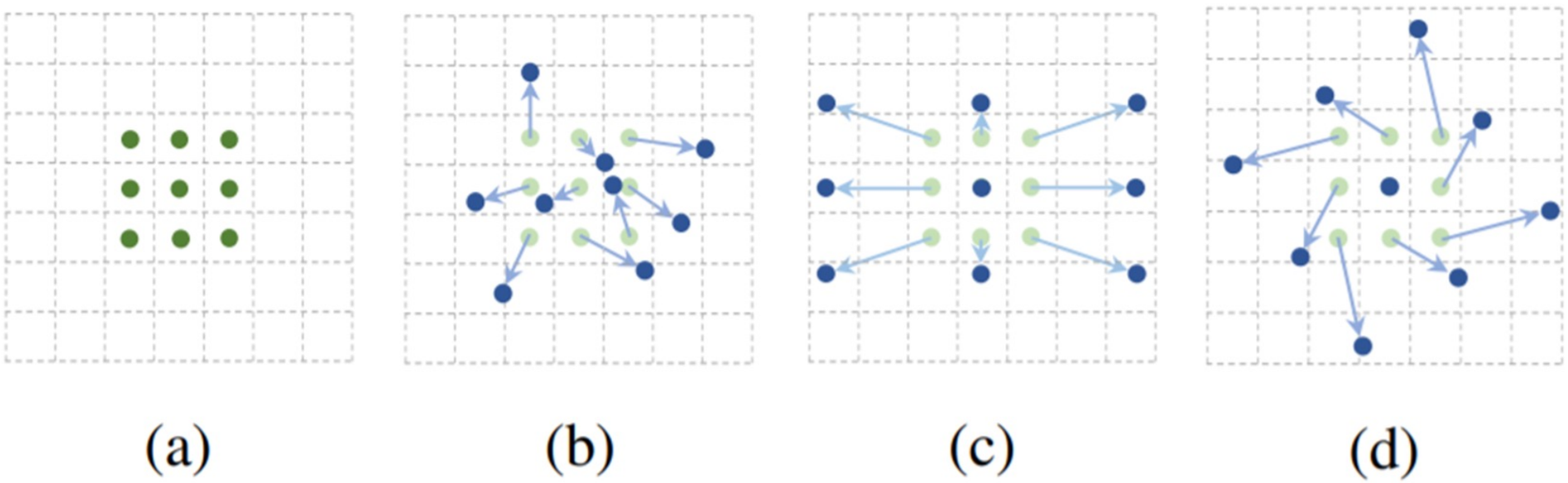
	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
FPN+ baseline	34.9	57.0	51.9	43.6	29.7	7.1
Iterative BBox	35.4	57.2	52.1	44.2	30.4	8.1
Integral Loss	35.4	57.3	52.5	44.4	29.9	6.9
Cascade R-CNN	**38.9**	**57.8**	**53.4**	**46.9**	**35.8**	**15.8**

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv2 [29]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [25]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet [24]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
Faster R-CNN+++ [18]*	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [23]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w FPN+ (ours)	ResNet-101	38.8	61.1	41.9	21.3	41.8	49.8
Faster R-CNN by G-RMI [19]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Deformable R-FCN [5]*	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN [16]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
AttractioNet [11]*	VGG16+Wide ResNet	35.7	53.4	39.3	15.6	38.0	52.7
Cascade R-CNN	ResNet-101	**42.8**	**62.1**	**46.3**	**23.7**	**45.5**	**55.2**

Deformable Convolution Networks (DCN)

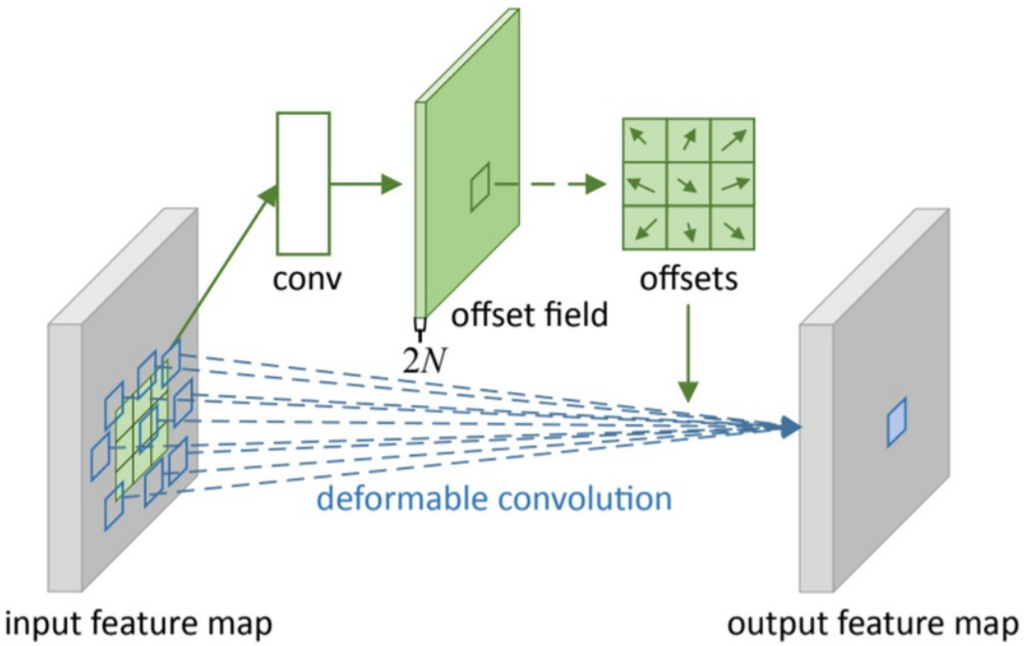
- Convolution Neural Network (CNN)의 한계점
  - 일정 패턴을 지니기 때문에 geometric transformations에 한계 존재
  - 기존 해결 방법
    - Geometric augmentation
    - Geometric invariant feature engineering

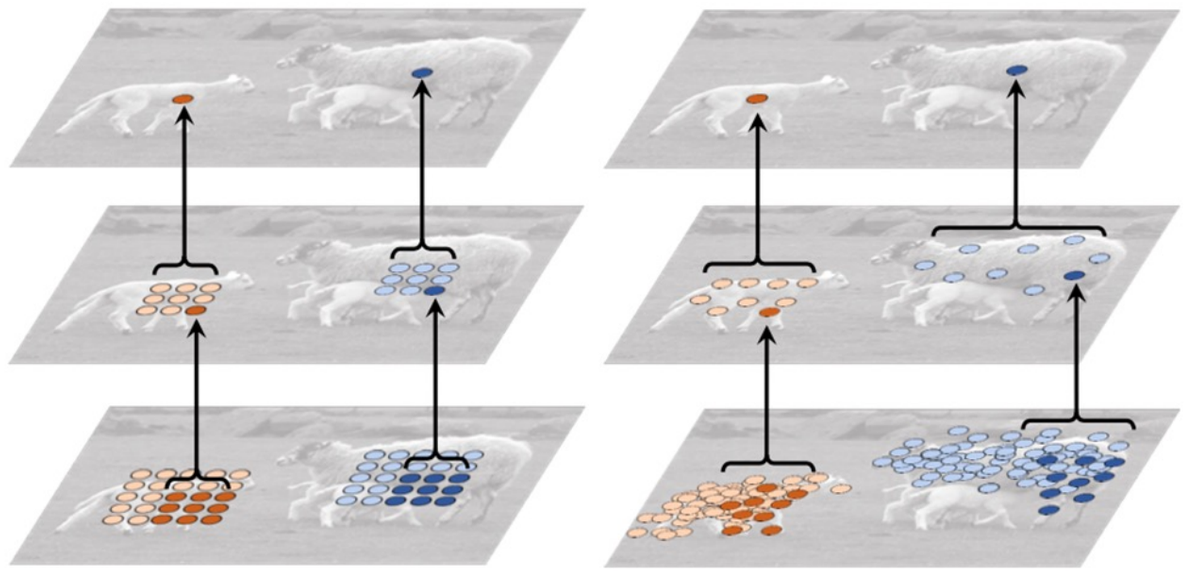
- Deformable Convolution
  - 기존의 고정된 사이즈인 정사각형 convolution을 벗어나는 연산



(a) 기존의 convolution 연산 (b) 랜덤 오프셋 추가 (c) 4 방향 확장 오프셋 추가 (d) 45도 회전 오프셋 추가

Architecture & Results





(a) standard convolution (b) deformable convolution

Figure 2: Illustration of 3×3 deformable convolution.

usage of deformable convolution (# layers)	DeepLab		class-aware RPN		Faster R-CNN		R-FCN	
	mIoU@V (%)	mIoU@C (%)	mAP@0.5 (%)	mAP@0.7 (%)	mAP@0.5 (%)	mAP@0.7 (%)	mAP@0.5 (%)	mAP@0.7 (%)
none (0, baseline)	69.7	70.4	68.0	44.9	78.1	62.1	80.0	61.8
res5c (1)	73.9	73.5	73.5	54.4	78.6	63.8	80.6	63.0
res5b,c (2)	74.8	74.4	74.3	56.3	78.5	63.3	81.0	63.8
res5a,b,c (3, default)	**75.2**	**75.2**	74.5	57.2	78.6	63.3	81.4	64.7
res5 & res4b22,b21,b20 (6)	74.8	75.1	**74.6**	**57.7**	**78.7**	**64.0**	**81.5**	**65.4**

Thank you