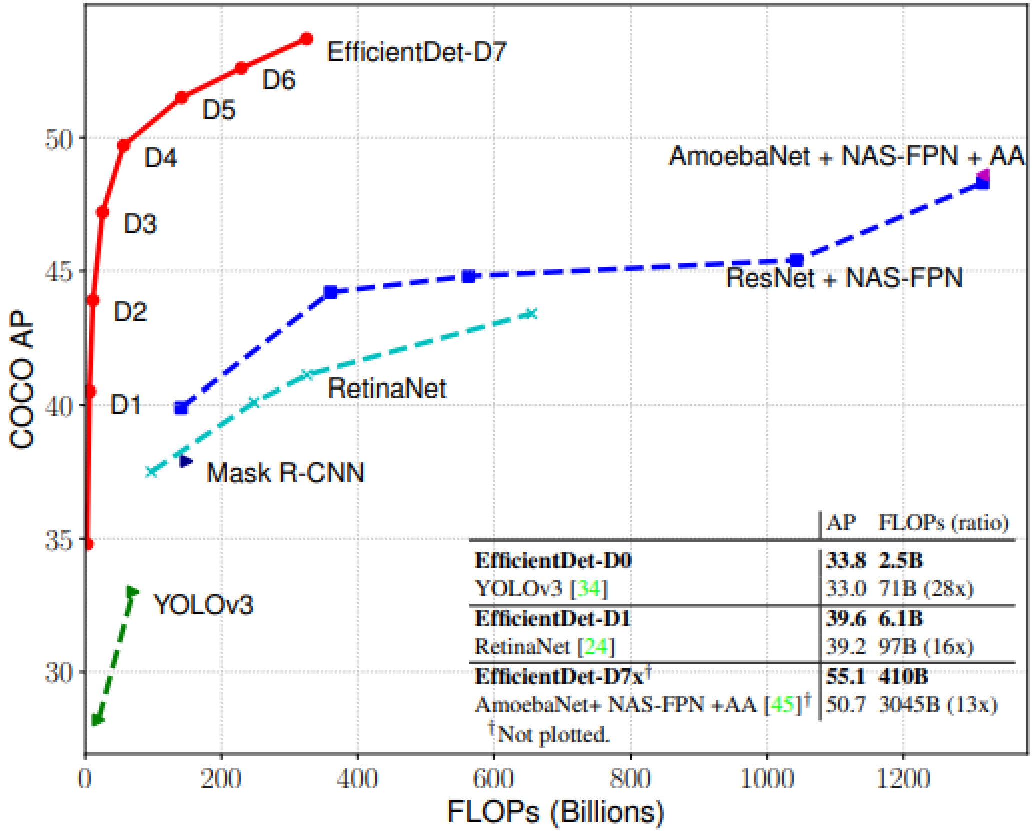
Detection : EfficientDet

Abstract

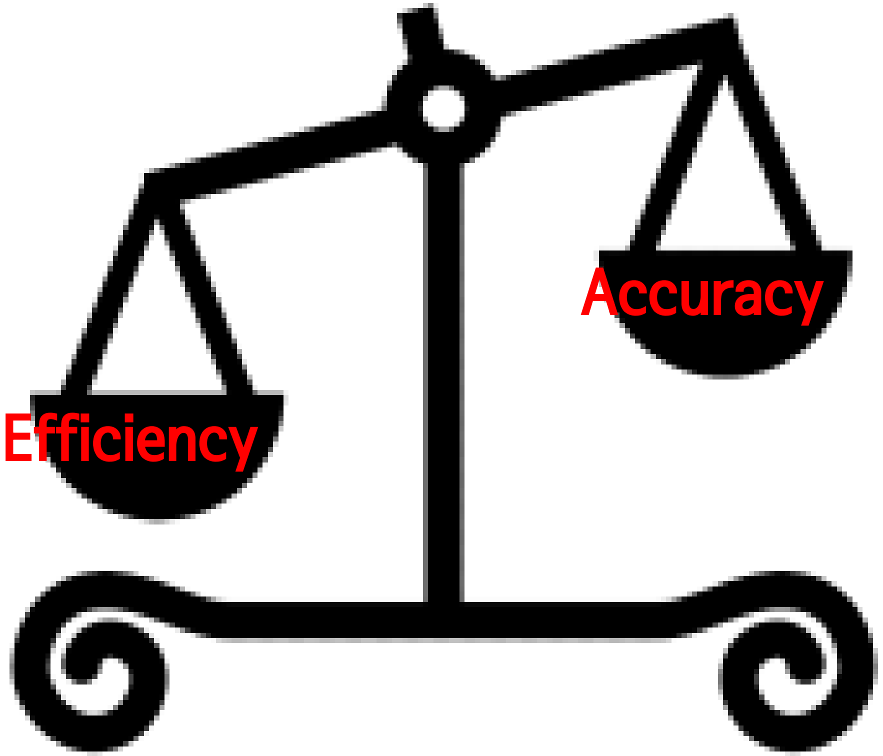
- ▶ EfficientNet: Image classification
- ▶ EfficientDet: object detection



	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.

Introduction



Main challenge and Solution

Two Challenges

1. Efficient multi - scale feature fusion

-> 서로 다른 input feature들을 합칠 때 구분없이 단순히 더하는 방식 지적 문제!

2. Model scaling

-> EfficientNet과 같이 resolution, depth,~~resolution~~ width중 하나만 키우는게 아니라 동시에 compound scaling을 진행해야 함을 제시

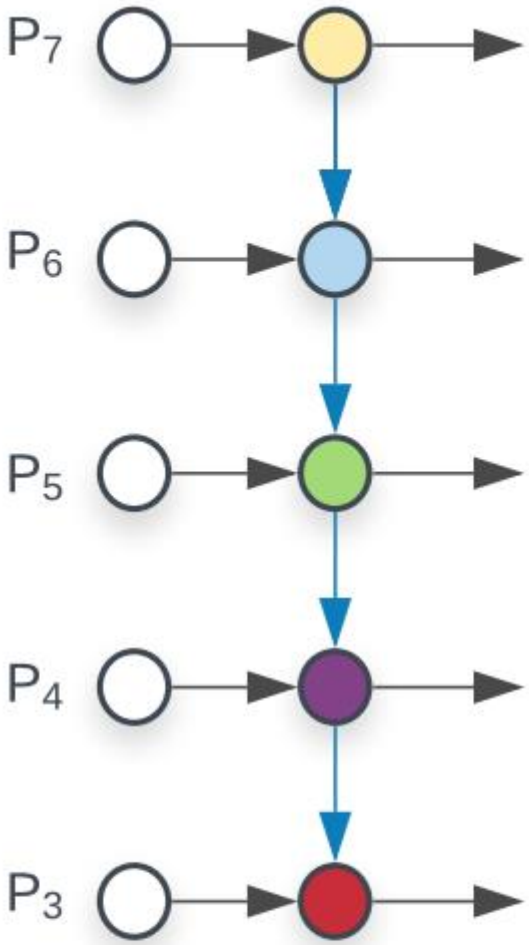
Contribution

- 쉽고 빠른 multi-scale feature fusion을 위한 BiFPN 제시

- Object Detection에도 Compound Scaling을 적용하는 방법을 제안

--> 이 둘을 접목하여 one stage 계열의 detector인데 좋은 accuracy와 efficiency를 보이는 efficientDet알고리즘 제시

BiFPN



(a) FPN

$$\vec{P}^{in} = (P^{in}_{l_1}, P^{in}_{l_2}, \dots)$$

$$\underline{\vec{P}^{out} = f(\vec{P}^{in})}$$

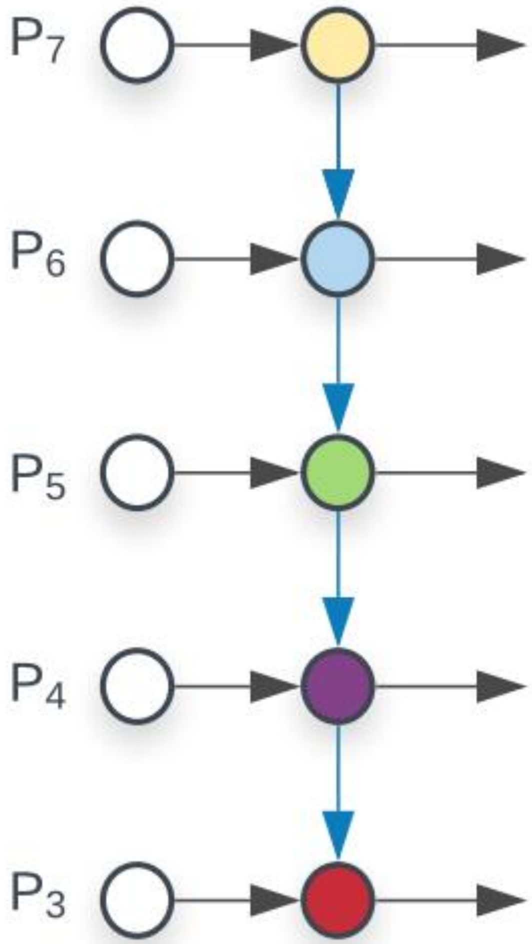
$$\vec{P}^{in} = (P^{in}_3, \dots P^{in}_7)$$

if 640x640 input

$\Rightarrow$ resolution

$640 \times \frac{1}{2^3} = \underline{80 \times 80}$

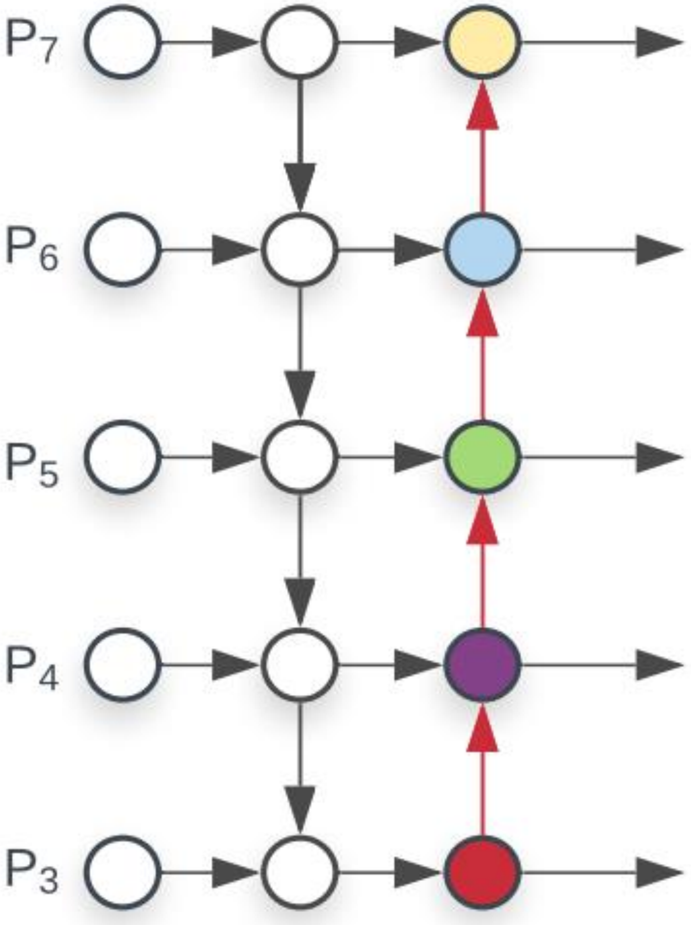
BiFPN



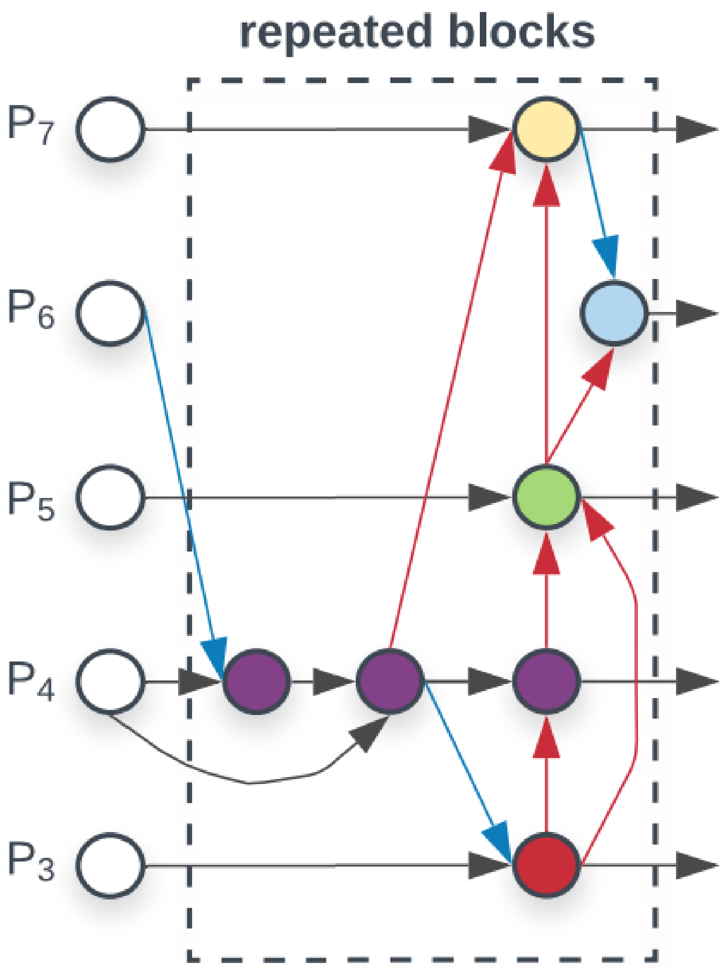
(a) FPN

$$
\begin{aligned}
P_7^{out} &= Conv(P_7^{in}) \\
P_6^{out} &= Conv(P_6^{in} + Resize(P_7^{out})) \\
&\ldots \\
P_3^{out} &= Conv(P_3^{in} + Resize(P_4^{out}))
\end{aligned}
$$

BiFPN



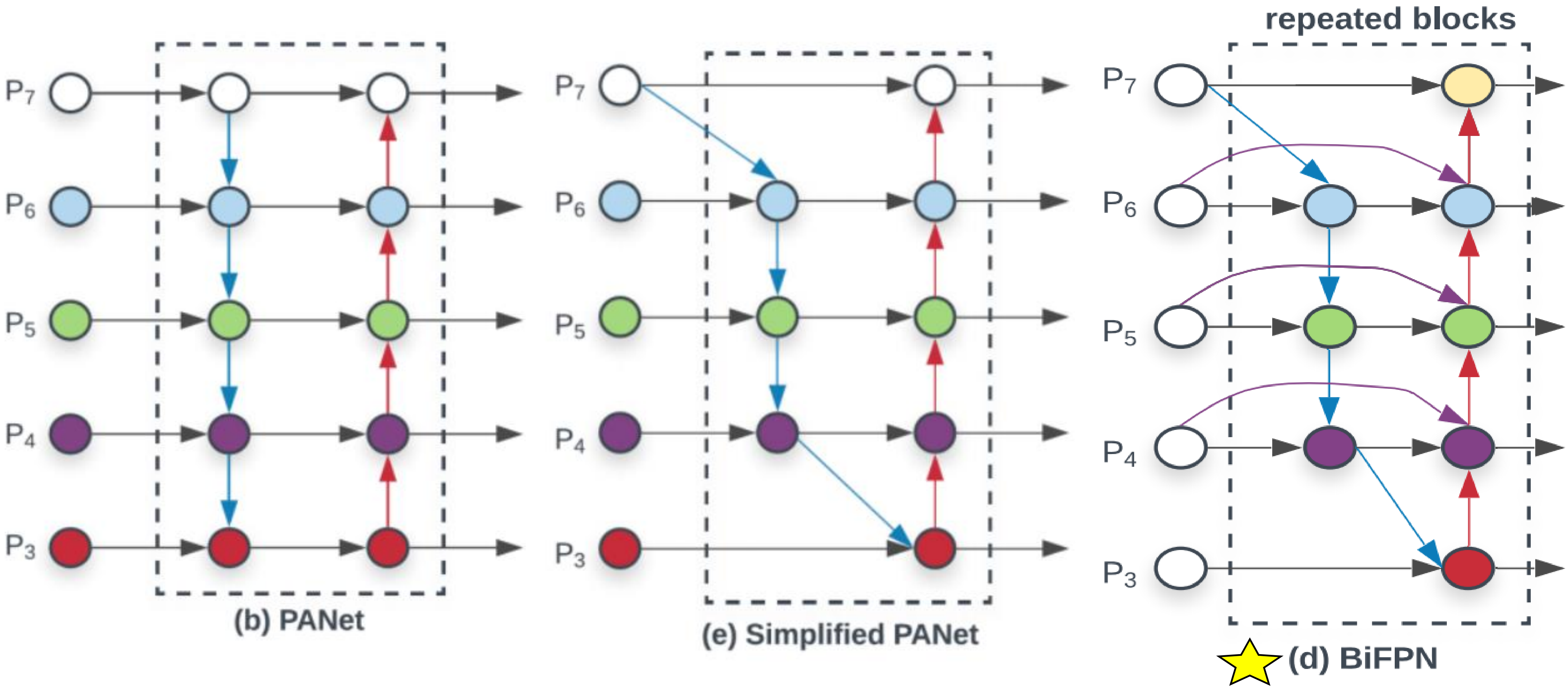
(b) PANet



(c) NAS-FPN

시간 오래 걸림

BiFPN


repeated blocks
P7
P6
P5
P4
P3
(b) PANet
P7
P6
P5
P4
P3
(e) Simplified PANet
P7
P6
P5
P4
P3
(d) BiFPN

BiFPN - weighted Feature Fusion

1. Unbounded fusion

$$\underline{O = \sum_i w_i \cdot \overset{\text{feature map}}{I_i}}$$

scalar 사용

→ I는 feature map이고 이에 weight를 곱해서 sum, 직관적

→Unbounded 되어 있기 때문에 학습에 불안정성을 유발할 수 있다.

BiFPN - weighted Feature Fusion

2. Softmax - based fusion

$$O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i$$

→ 각각을 0에서 1로 값을 변환하여 sum을 해주는 방식

→ 성능은 좋지만 gpu에서 연산을 시도하면 속도를 떨어뜨리는 요인

BiFPN - weighted Feature Fusion

3. Fast Normalized Fusion

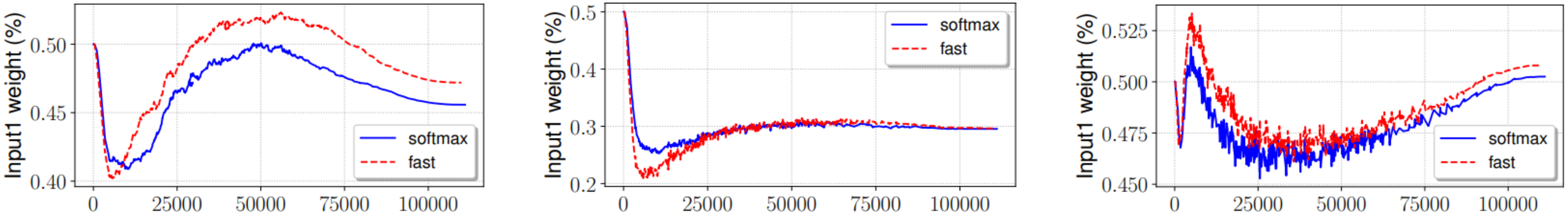
$$O = \sum_i \frac{w_i}{\underset{\substack{\downarrow \\ 1\text{e}-3}}{\tilde{\epsilon}} + \sum_j w_j} \cdot I_i$$

→ relu를 거치기 때문에 weight non-zero 보장

→ 입실론을 넣어주어 분모가 0이 되는 것을 방지

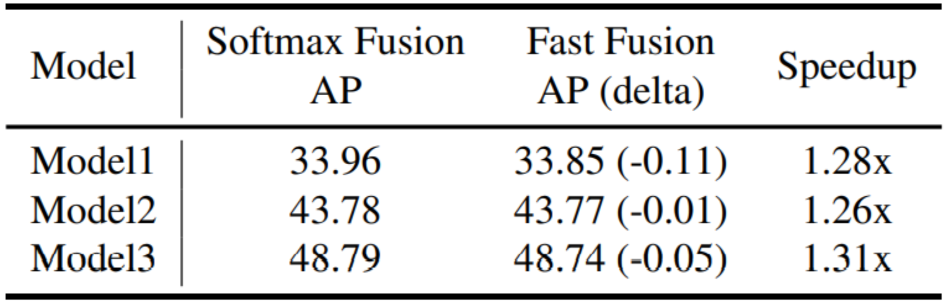
→ softmax based fusion과 성능에 있어서 거의 비슷하지만 속도면에서 30퍼센트 정도 빠름

BiFPN



(a) Example Node 1 (b) Example Node 2 (c) Example Node 3

Figure 5: **Softmax vs. fast normalized feature fusion –** (a) - (c) shows normalized weights (i.e., importance) during training for three representative nodes; each node has two inputs (input1 & input2) and their normalized weights always sum up to 1.



Model	Softmax Fusion AP	Fast Fusion AP (delta)	Speedup
Model1	33.96	33.85 (-0.11)	1.28x
Model2	43.78	43.77 (-0.01)	1.26x
Model3	48.79	48.74 (-0.05)	1.31x

Table 6: **Comparison of different feature fusion –** Our fast fusion achieves similar accuracy as softmax-based fusion, but runs 28% - 31% faster.

EfficientDet Architecture

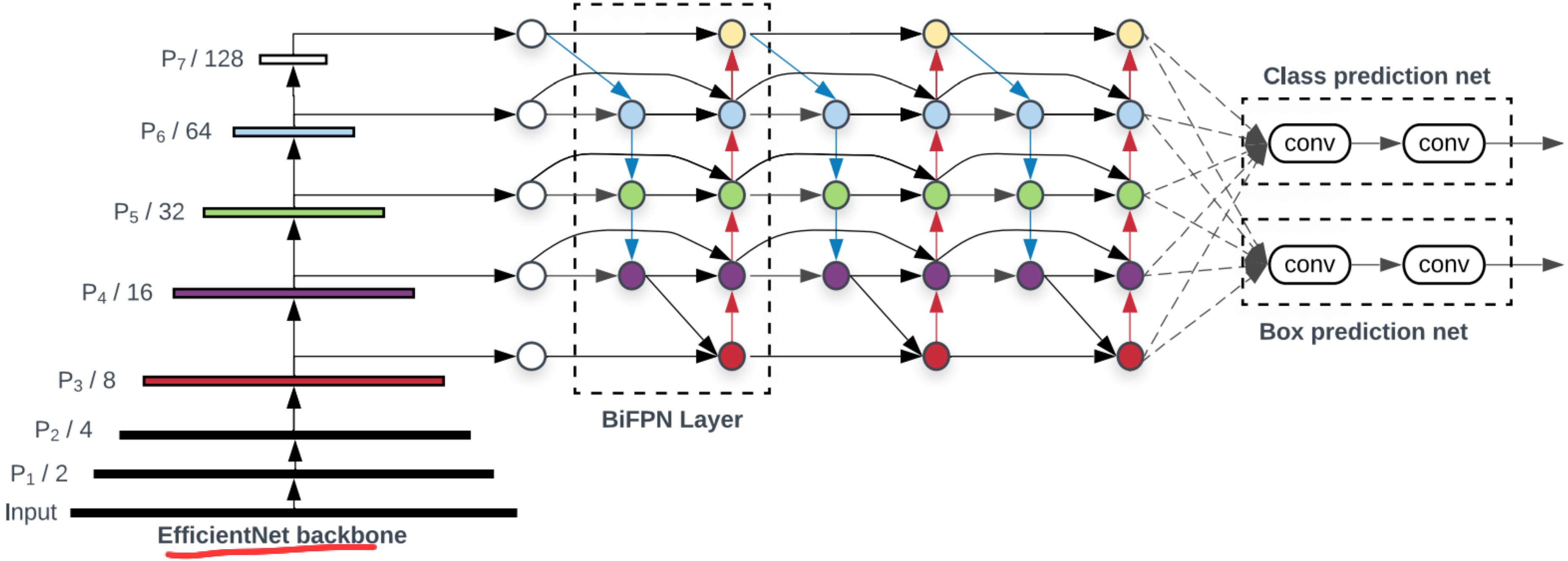
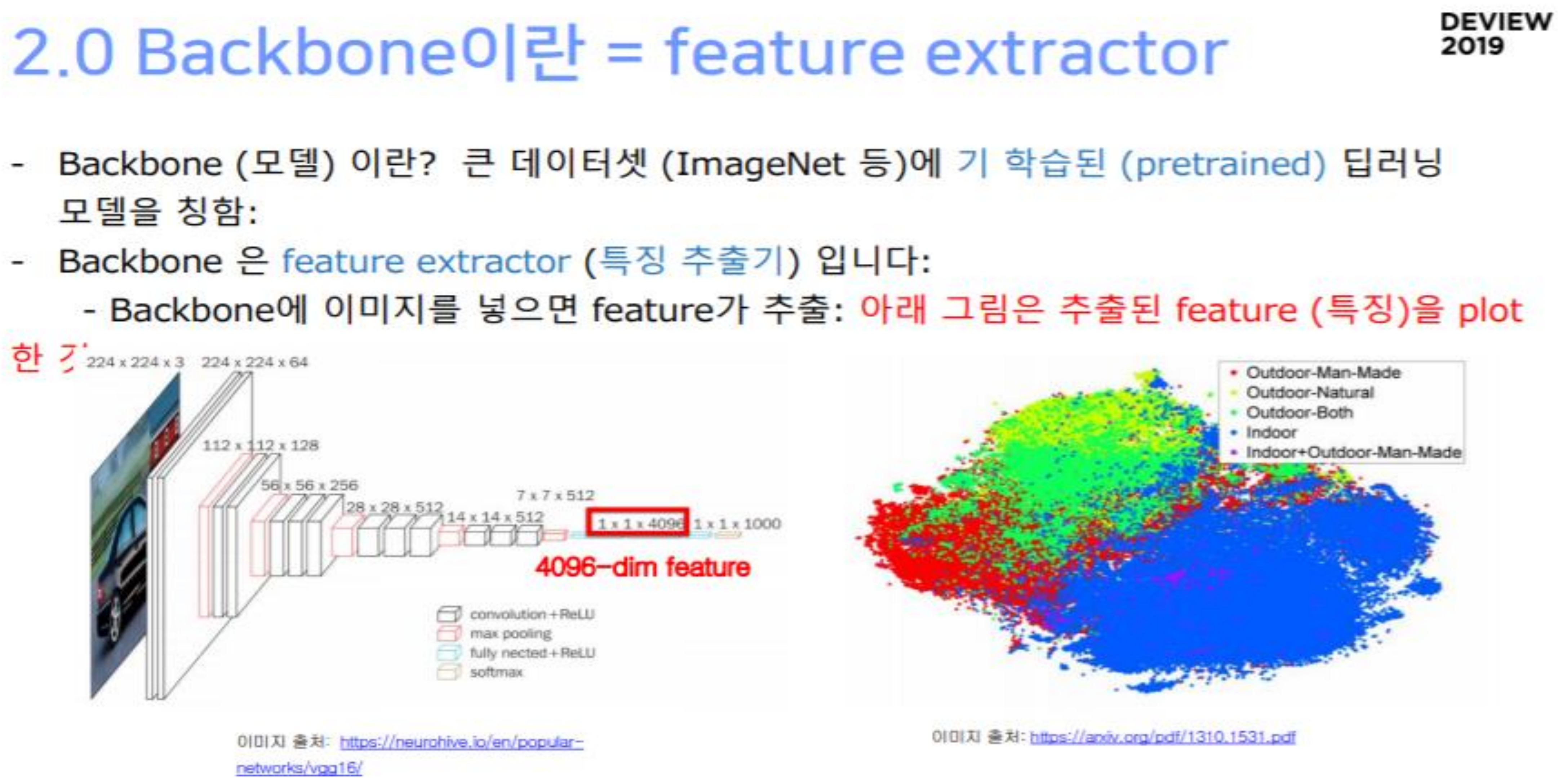


Figure 3: **EfficientDet architecture –** It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

EfficientDet Architecture



2.0 Backbone이란 = feature extractor

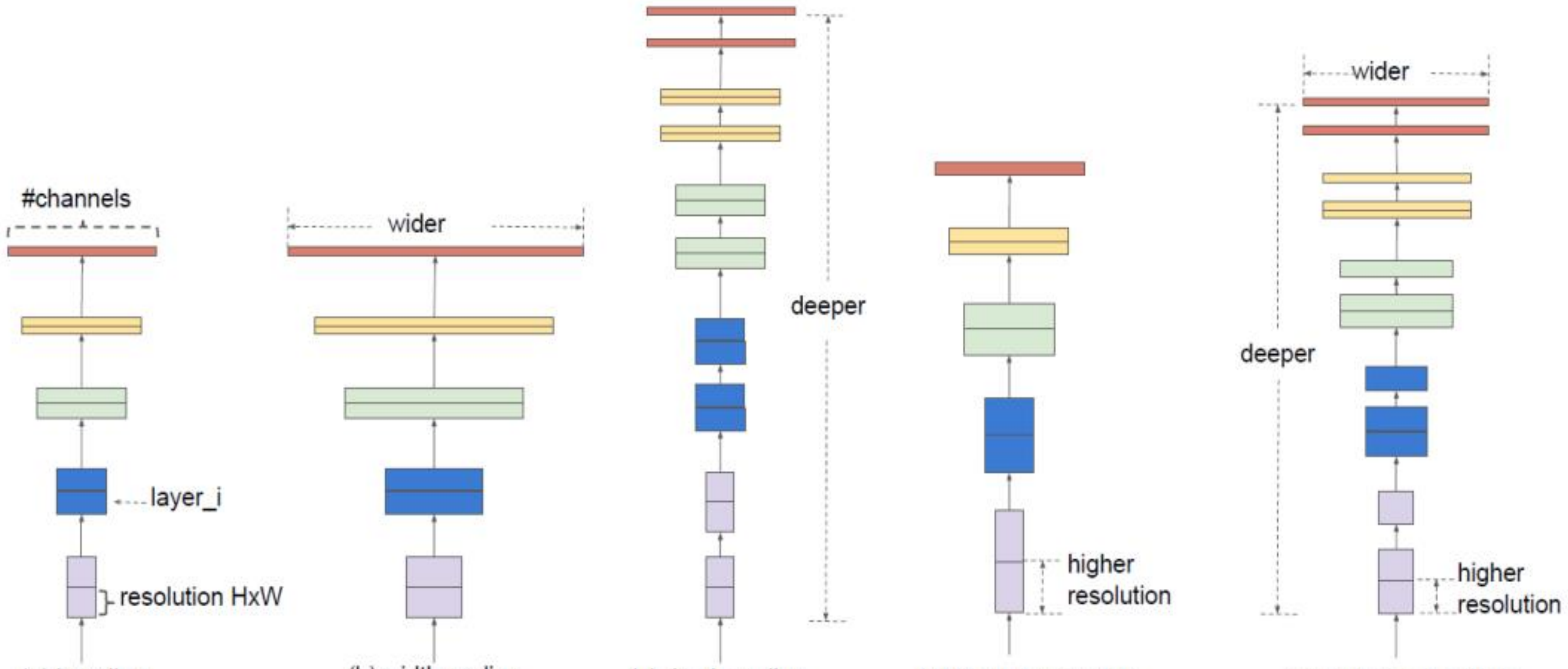
DEVIEW 2019

- Backbone (모델) 이란? 큰 데이터셋 (ImageNet 등)에 기 학습된 (pretrained) 딥러닝 모델을 칭함:
- Backbone 은 feature extractor (특징 추출기) 입니다:
  - Backbone에 이미지를 넣으면 feature가 추출: 아래 그림은 추출된 feature (특징)을 plot 한 것

이미지 출처: https://neurohive.io/en/popular-networks/vgg16/

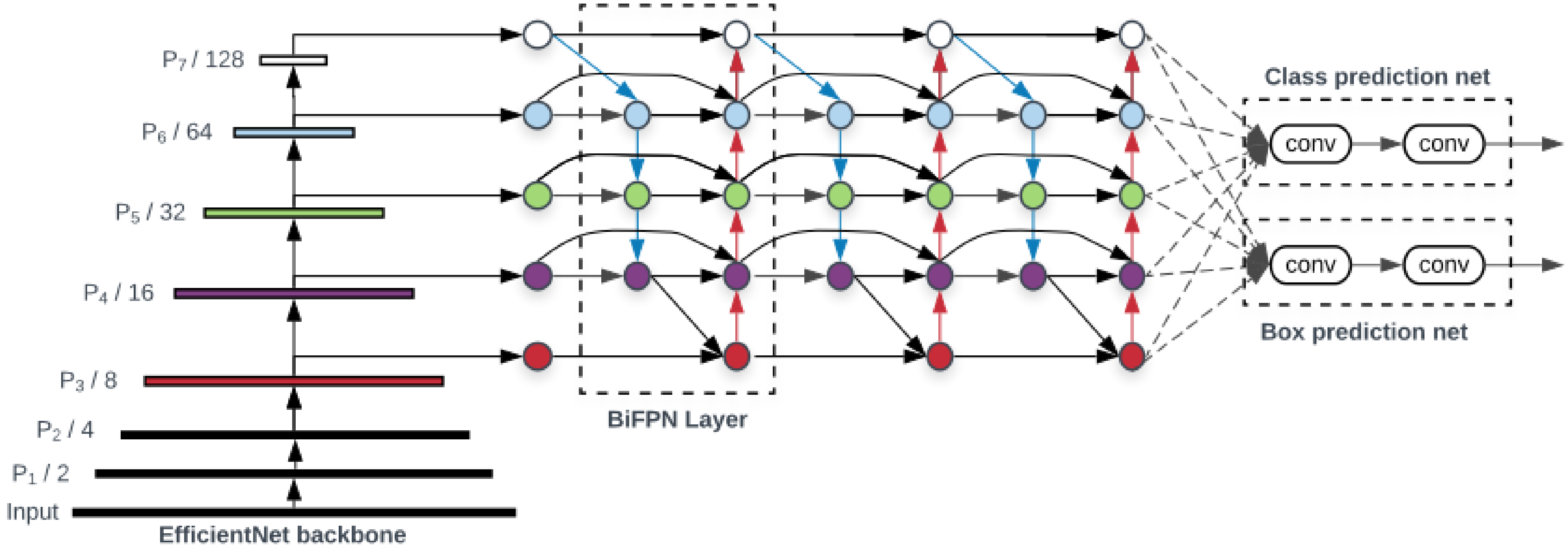
이미지 출처: https://arxiv.org/pdf/1310.1531.pdf

EfficientNet



(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

EfficientDet compound scaling

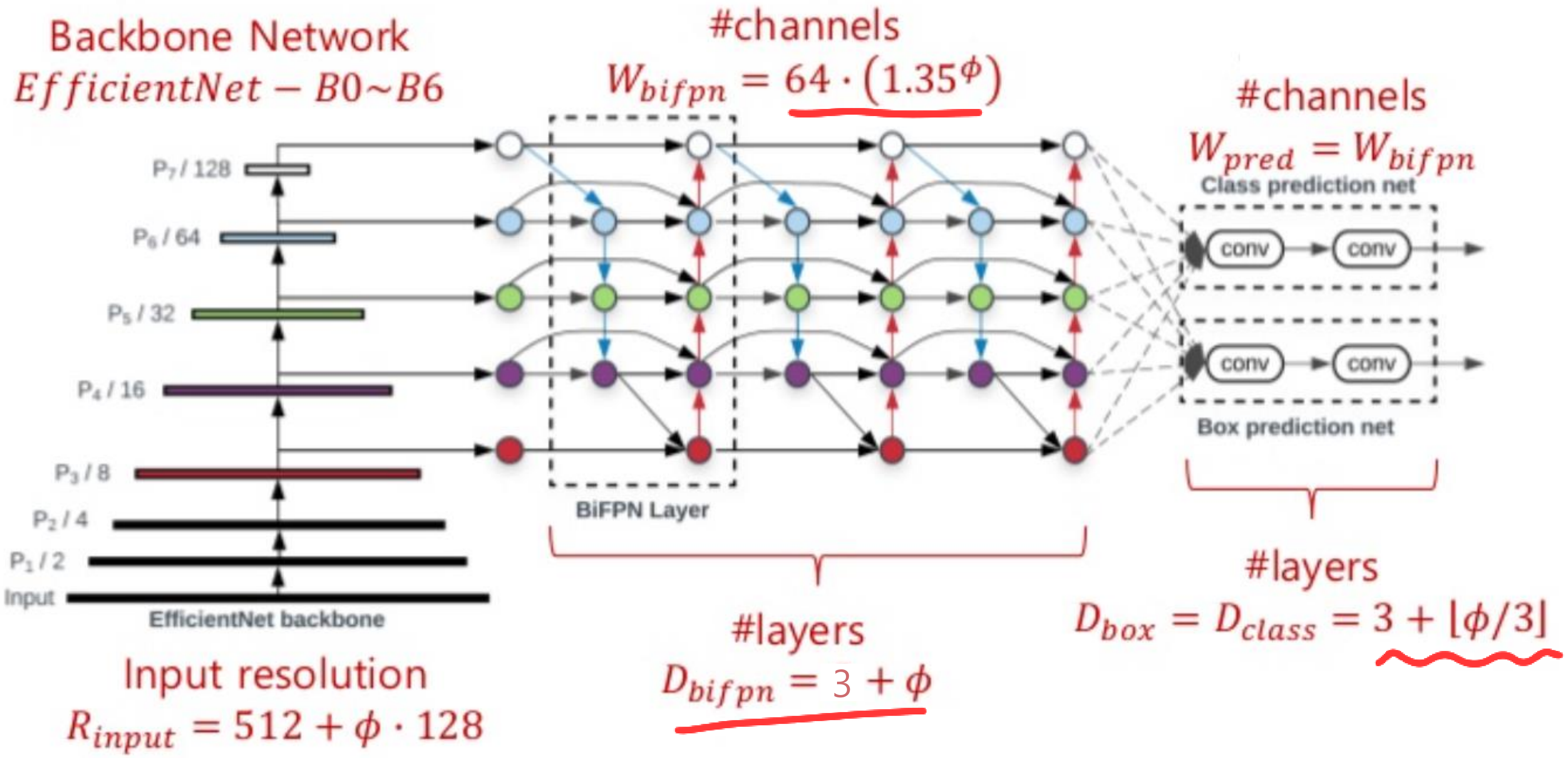


→ Compound Scaling처럼 input의 resolution과 backbone network의 크기를 늘려주었고, BiFPN과 Box/class network 도 동시에 키워줌

→ object detection에서 어떤 dimension으로 scaling해야하는지 정확하게 알기 힘들어 heuristic한 scaling접근방법을 사용했다고 함

휴리스틱이란..?불충분한 시간이나 정보로 인하여 합리적인 판단을 할 수 없거나, 체계적이면서 합리적인 판단이 굳이 필요하지 않은 상황에서 사람들이 빠르게 사용할 수 있게 보다 용이하게 구성된 간편추론의 방법

EfficientDet compound scaling



Backbone Network

$EfficientNet - B0{\sim}B6$

Input resolution

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

#channels

$W_{bifpn} = 64 \cdot (1.35^{\phi})$

#layers

$D_{bifpn} = 3 + \phi$

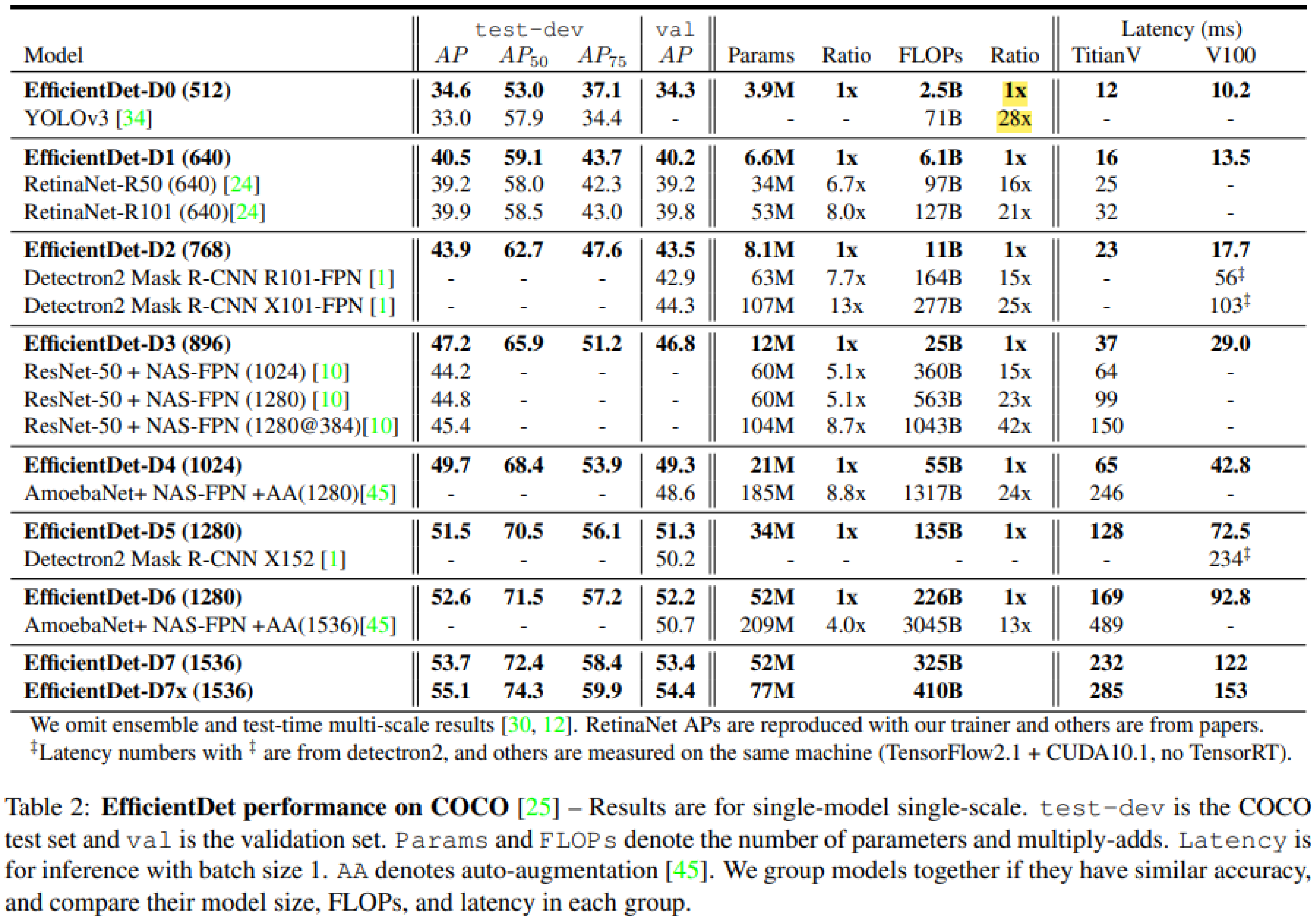
#channels

$W_{pred} = W_{bifpn}$

#layers

$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor$

Experiments

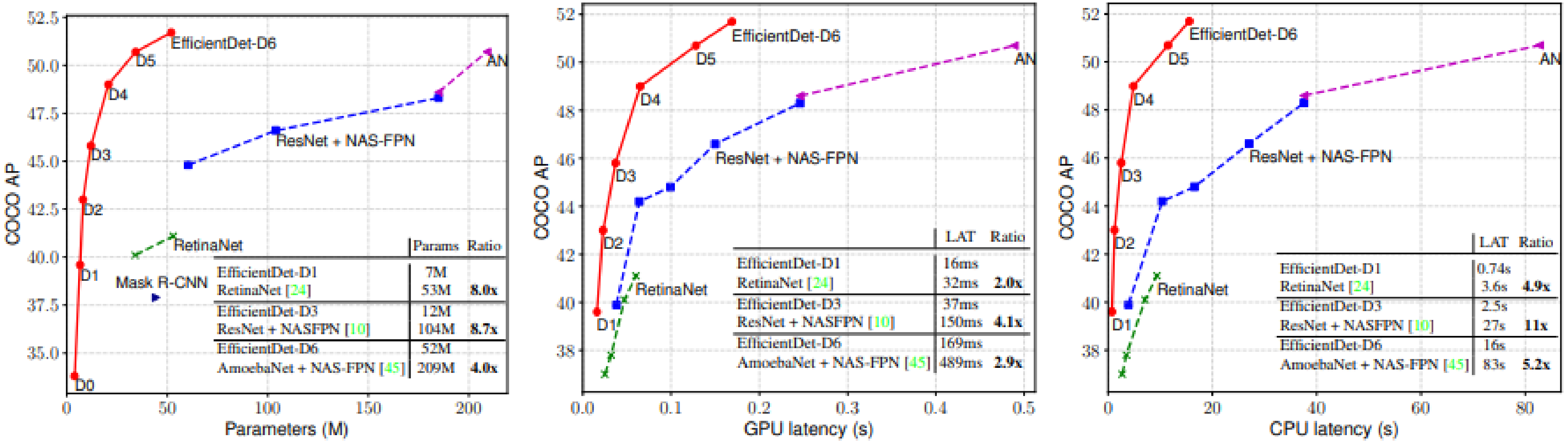


Model	test-dev			val	Params	Ratio	FLOPs	Ratio	Latency (ms)	
	AP	AP_{50}	AP_{75}	AP					TitianV	V100
EfficientDet-D0 (512)	**34.6**	**53.0**	**37.1**	**34.3**	**3.9M**	**1x**	**2.5B**	**1x**	**12**	**10.2**
YOLOv3 [34]	33.0	57.9	34.4	-	-	-	71B	28x	-	-
EfficientDet-D1 (640)	**40.5**	**59.1**	**43.7**	**40.2**	**6.6M**	**1x**	**6.1B**	**1x**	**16**	**13.5**
RetinaNet-R50 (640) [24]	39.2	58.0	42.3	39.2	34M	6.7x	97B	16x	25	-
RetinaNet-R101 (640)[24]	39.9	58.5	43.0	39.8	53M	8.0x	127B	21x	32	-
EfficientDet-D2 (768)	**43.9**	**62.7**	**47.6**	**43.5**	**8.1M**	**1x**	**11B**	**1x**	**23**	**17.7**
Detectron2 Mask R-CNN R101-FPN [1]	-	-	-	42.9	63M	7.7x	164B	15x	-	$56^{\ddagger}$
Detectron2 Mask R-CNN X101-FPN [1]	-	-	-	44.3	107M	13x	277B	25x	-	$103^{\ddagger}$
EfficientDet-D3 (896)	**47.2**	**65.9**	**51.2**	**46.8**	**12M**	**1x**	**25B**	**1x**	**37**	**29.0**
ResNet-50 + NAS-FPN (1024) [10]	44.2	-	-	-	60M	5.1x	360B	15x	64	-
ResNet-50 + NAS-FPN (1280) [10]	44.8	-	-	-	60M	5.1x	563B	23x	99	-
ResNet-50 + NAS-FPN (1280@384)[10]	45.4	-	-	-	104M	8.7x	1043B	42x	150	-
EfficientDet-D4 (1024)	**49.7**	**68.4**	**53.9**	**49.3**	**21M**	**1x**	**55B**	**1x**	**65**	**42.8**
AmoebaNet+ NAS-FPN +AA(1280)[45]	-	-	-	48.6	185M	8.8x	1317B	24x	246	-
EfficientDet-D5 (1280)	**51.5**	**70.5**	**56.1**	**51.3**	**34M**	**1x**	**135B**	**1x**	**128**	**72.5**
Detectron2 Mask R-CNN X152 [1]	-	-	-	50.2	-	-	-	-	-	$234^{\ddagger}$
EfficientDet-D6 (1280)	**52.6**	**71.5**	**57.2**	**52.2**	**52M**	**1x**	**226B**	**1x**	**169**	**92.8**
AmoebaNet+ NAS-FPN +AA(1536)[45]	-	-	-	50.7	209M	4.0x	3045B	13x	489	-
EfficientDet-D7 (1536)	**53.7**	**72.4**	**58.4**	**53.4**	**52M**		**325B**		**232**	**122**
EfficientDet-D7x (1536)	**55.1**	**74.3**	**59.9**	**54.4**	**77M**		**410B**		**285**	**153**

We omit ensemble and test-time multi-scale results [30, 12]. RetinaNet APs are reproduced with our trainer and others are from papers.
‡Latency numbers with ‡ are from detectron2, and others are measured on the same machine (TensorFlow2.1 + CUDA10.1, no TensorRT).

Table 2: **EfficientDet performance on COCO** [25] – Results are for single-model single-scale. `test-dev` is the COCO test set and `val` is the validation set. `Params` and `FLOPs` denote the number of parameters and multiply-adds. `Latency` is for inference with batch size 1. `AA` denotes auto-augmentation [45]. We group models together if they have similar accuracy, and compare their model size, FLOPs, and latency in each group.

Experiments



	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**

	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**

	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**

(a) Model Size

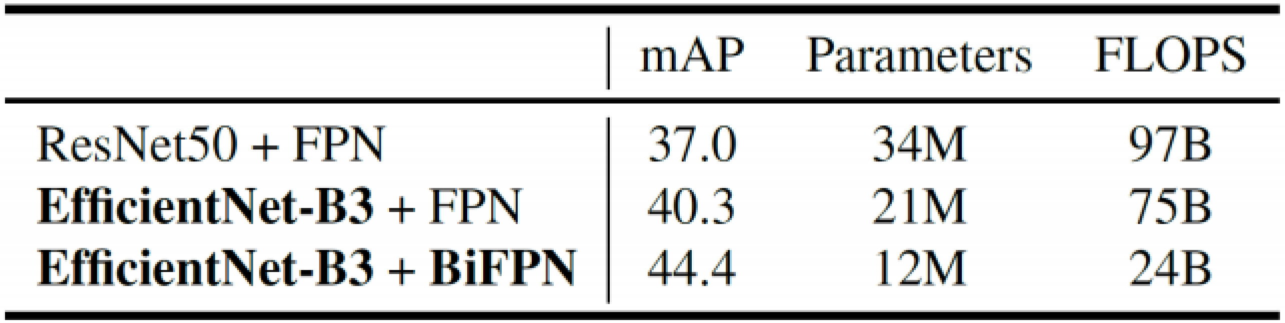
(b) GPU Latency

(c) CPU Latency

Figure 4: **Model size and inference latency comparison** – Latency is measured with batch size 1 on the same machine equipped with a Titan V GPU and Xeon CPU. AN denotes AmoebaNet + NAS-FPN trained with auto-augmentation [45]. Our EfficientDet models are 4x - 9x smaller, 2x - 4x faster on GPU, and 5x - 11x faster on CPU than other detectors.

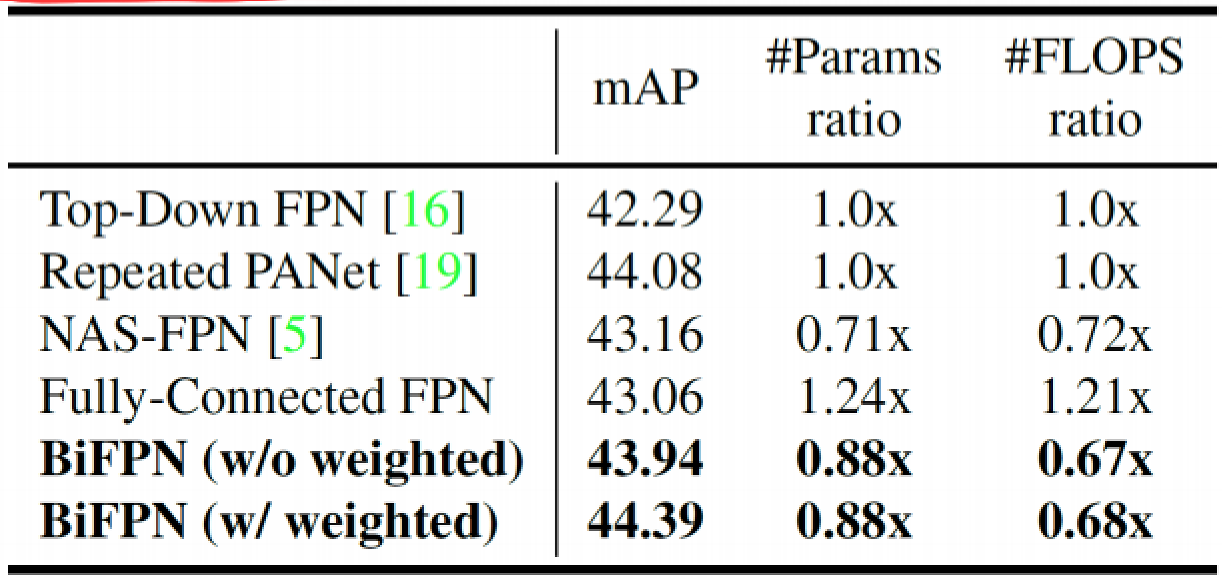
Ablation Study

- Disentangling Backbone and BiFPN



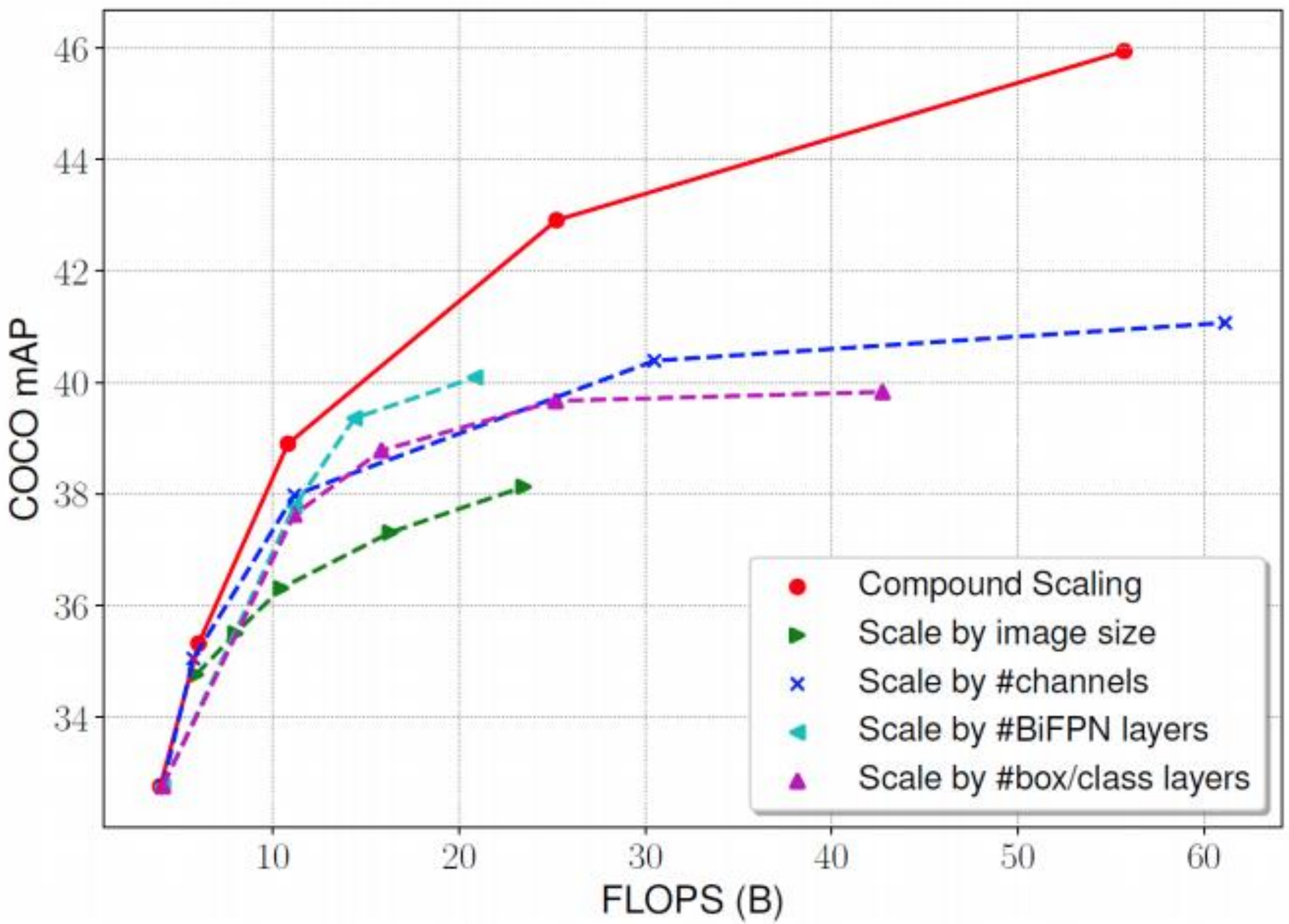
	mAP	Parameters	FLOPS
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + **BiFPN**	44.4	12M	24B

- BiFPN Cross-Scale Connections



	mAP	#Params	#FLOPS
		ratio	ratio
Top-Down FPN [16]	42.29	1.0x	1.0x
Repeated PANet [19]	44.08	1.0x	1.0x
NAS-FPN [5]	43.16	0.71x	0.72x
Fully-Connected FPN	43.06	1.24x	1.21x
BiFPN (w/o weighted)	**43.94**	**0.88x**	**0.67x**
BiFPN (w/ weighted)	**44.39**	**0.88x**	**0.68x**

Ablation Study



Reference

https://ys-cs17.tistory.com/31
https://hoya012.github.io/blog/EfficientDet-Review/

감사합니다! :D