EfficientDet: Scalable and Efficient Object Detection

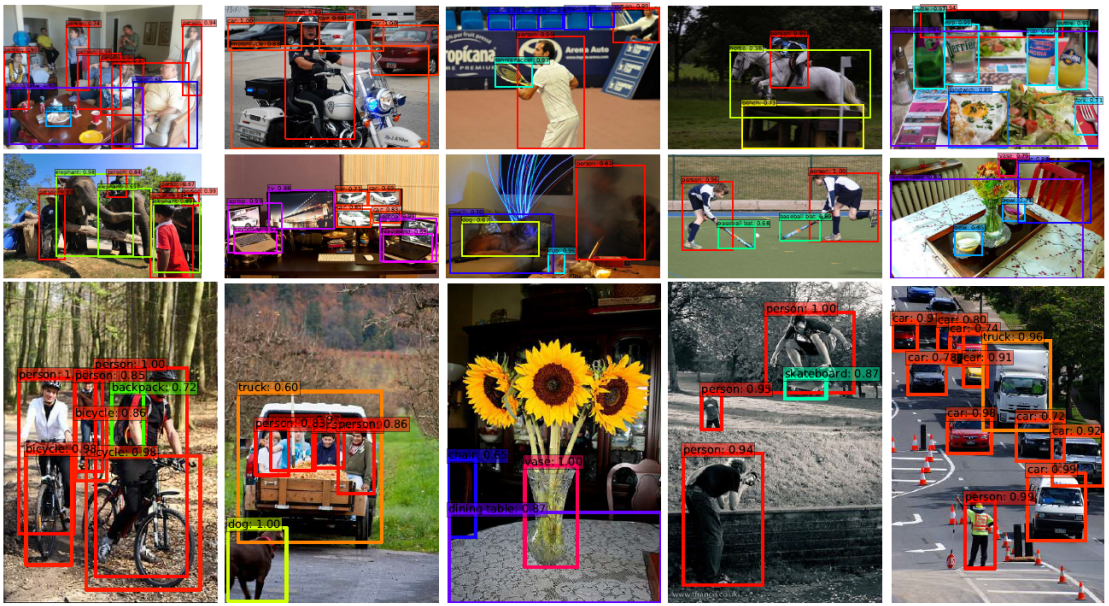
Mingxing Tan, Ruoming Pang, Quoc V. Le

CVPR 2020

2021. 01. 04 Jinhee Kim

1 Introduction

- **Task: object detection**



1 Introduction

- Is it possible to build a scalable detection architecture with both higher accuracy and better efficiency across a wide spectrum of resource constraints (e.g., from 3B to 300B FLOPs)?

  - **Challenge 1: efficient multi-scale feature fusion**
    - Prior work: cross-scale feature fusion with simple summation.
    - Since the different input features are at different resolutions, they usually contribute to the fused output feature **unequally**.

      → **Weighted bi-directional feature pyramid network (BiFPN)**

  - **Challenge 2: model scaling**
    - Prior work: bigger backbone networks or larger input image sizes for higher accuracy.
    - But scaling up feature network and box/class prediction network is **also** critical when taking into account both accuracy and efficiency.

      → **A compound scaling method for object detectors** that jointly scales up the resolution/depth/width for all backbone, feature network, box/class prediction network.

1 Introduction

- **EfficientDet:** a new family of object detectors with several key optimizations to improve efficiency, which is a combination of

  ① **Weighted bi-directional feature pyramid network (BiFPN)** that allows easy and fast multiscale feature fusion,

  ② **Compound scaling method** that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time,

  ③ **EfficientNet** backbone.

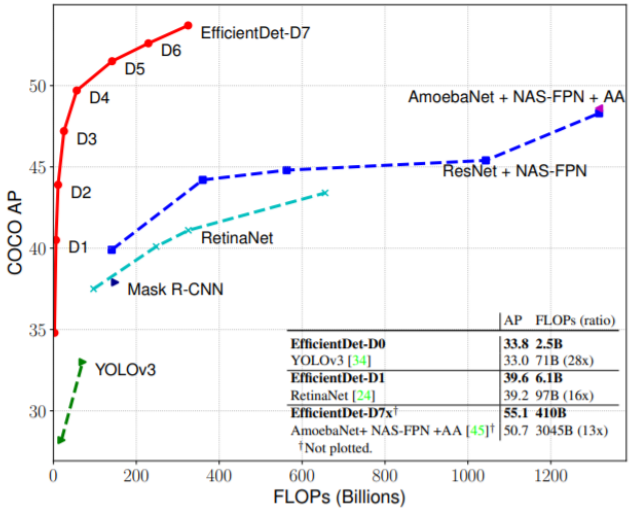
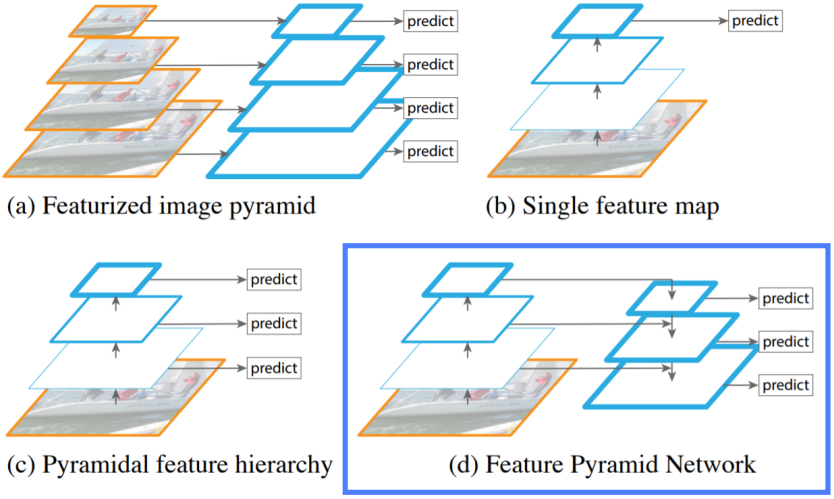


Figure 1: **Model FLOPs vs. COCO accuracy –** All numbers are for single-model single-scale. Our EfficientDet achieves new state-of-the-art 55.1% COCO AP with much fewer parameters and FLOPs than previous detectors. More studies on different backbones and FPN/NAS-FPN/BiFPN are in Table 4 and 5. Complete results are in Table 2.

Feature Pyramid Network (FPN) (Lin et al., CVPR 2017)



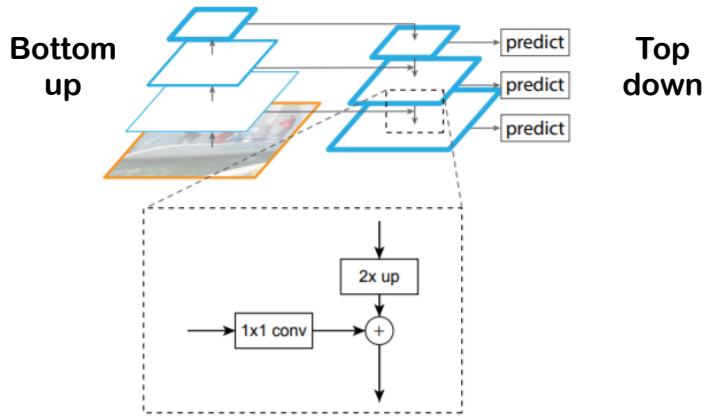
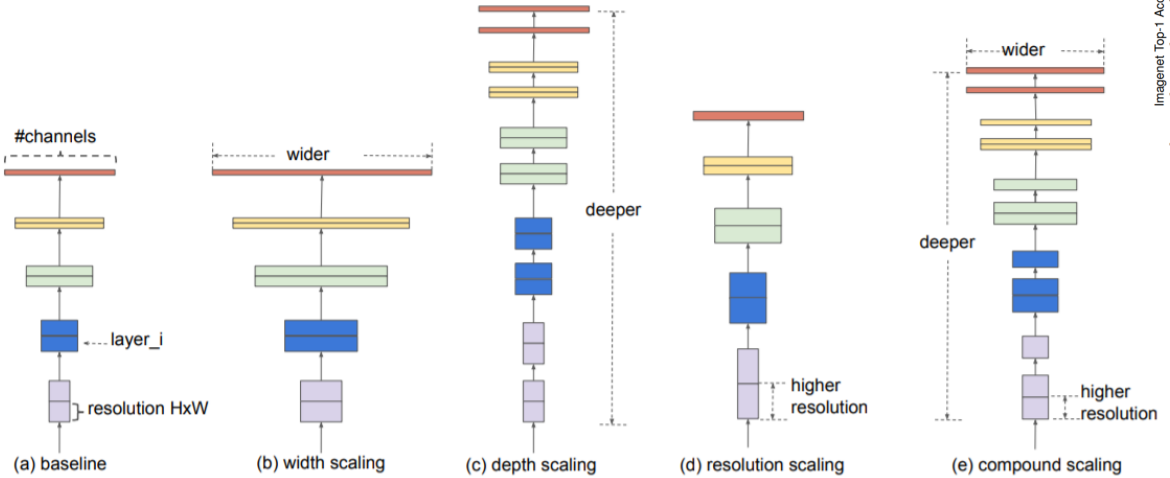
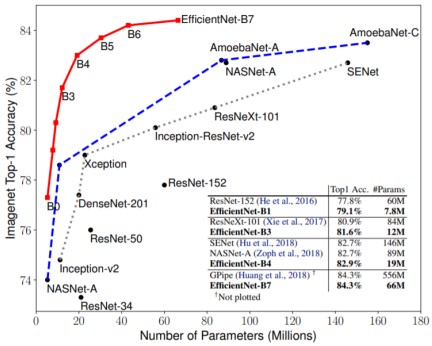


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

EfficientNet (Tan et al., ICML 2019)





depth: $d = \alpha^\phi$

width: $w = \beta^\phi$

resolution: $r = \gamma^\phi$

s.t. $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$

$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$

Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of netw width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

① Weighted bi-directional feature pyramid network (BiFPN)

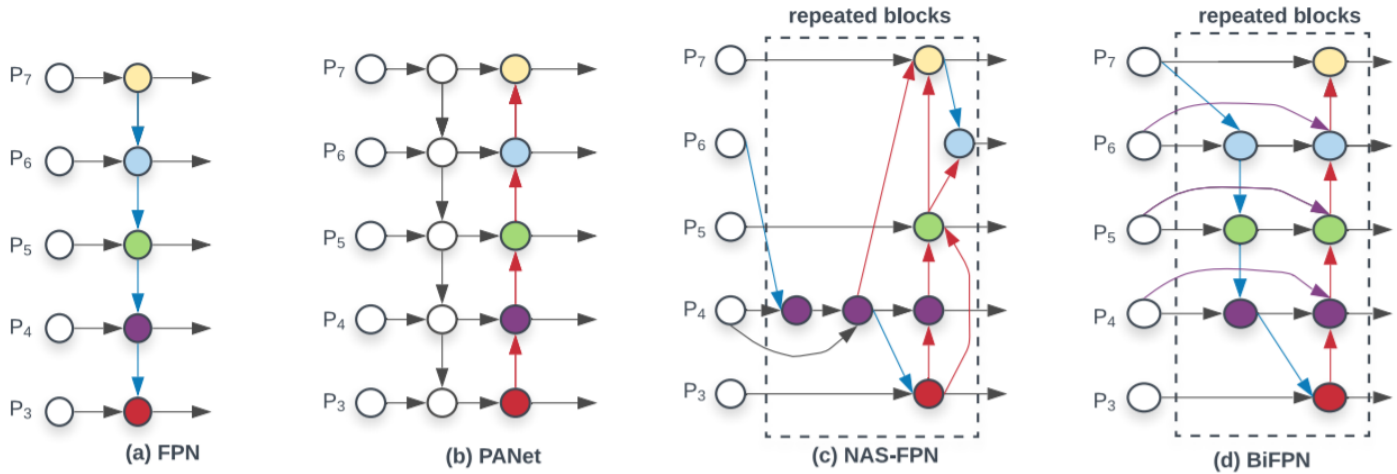


Figure 2: **Feature network design –** (a) FPN [23] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P_3 - P_7); (b) PANet [26] adds an additional bottom-up pathway on top of FPN; (c) NAS-FPN [10] use neural architecture search to find an irregular feature network topology and then repeatedly apply the same block; (d) is our BiFPN with better accuracy and efficiency trade-offs.

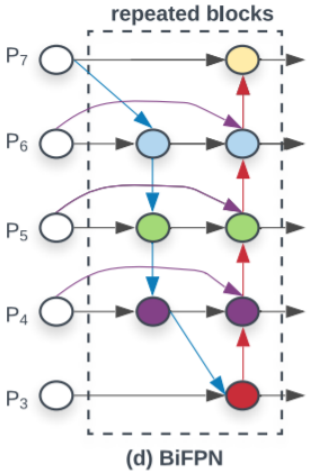
① Weighted bi-directional feature pyramid network (BiFPN)

A. Several optimizations for cross-scale connections

a. They remove the nodes that only have one input edge.
   ✓ It will have less contribution to feature network that aims at fusing different features.

b. They add an extra edge from the original input to output node if they are at the same level, in order to fuse more features without adding much cost.

c. Unlike PANet, they treat each bidirectional (top-down & bottom-up) path as one feature network layer, and repeat the same layer multiple times to enable more high-level feature fusion.



① Weighted bi-directional feature pyramid network (BiFPN)

B. Weighted Feature Fusion

*w: learnable weight that can be a scalar (per-feature), a vector (per-channel), or a multi-dimensional tensor (per-pixel).

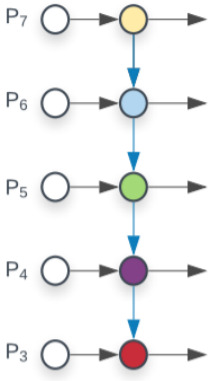
Unbounded fusion: $O = \sum_i w_i \cdot I_i,$ → potential training instability

Softmax-based fusion: $O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i.$ → Significant slowdown on GPU hardware

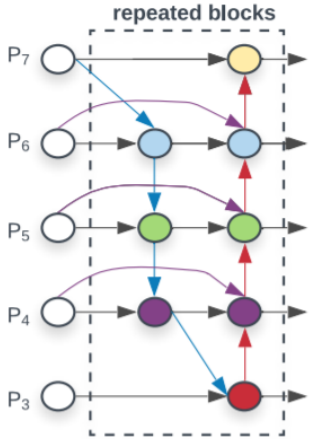
Fast normalized fusion: $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$ $w_i \geq 0$ is ensured by applying a Relu after each w_i

The value of each normalized weight falls between 0 and 1,
And much more efficient since there is no softmax operation.

① Weighted bi-directional feature pyramid network (BiFPN)



(a) FPN



(d) BiFPN

$$P_7^{out} = Conv(P_7^{in})$$
$$P_6^{out} = Conv(P_6^{in} + Resize(P_7^{out}))$$
$$\dots$$
$$P_3^{out} = Conv(P_3^{in} + Resize(P_4^{out}))$$

$$P_6^{td} = Conv\left(\frac{w_1 \cdot P_6^{in} + w_2 \cdot Resize(P_7^{in})}{w_1 + w_2 + \epsilon}\right)$$
$$P_6^{out} = Conv\left(\frac{w_1' \cdot P_6^{in} + w_2' \cdot P_6^{td} + w_3' \cdot Resize(P_5^{out})}{w_1' + w_2' + w_3' + \epsilon}\right)$$

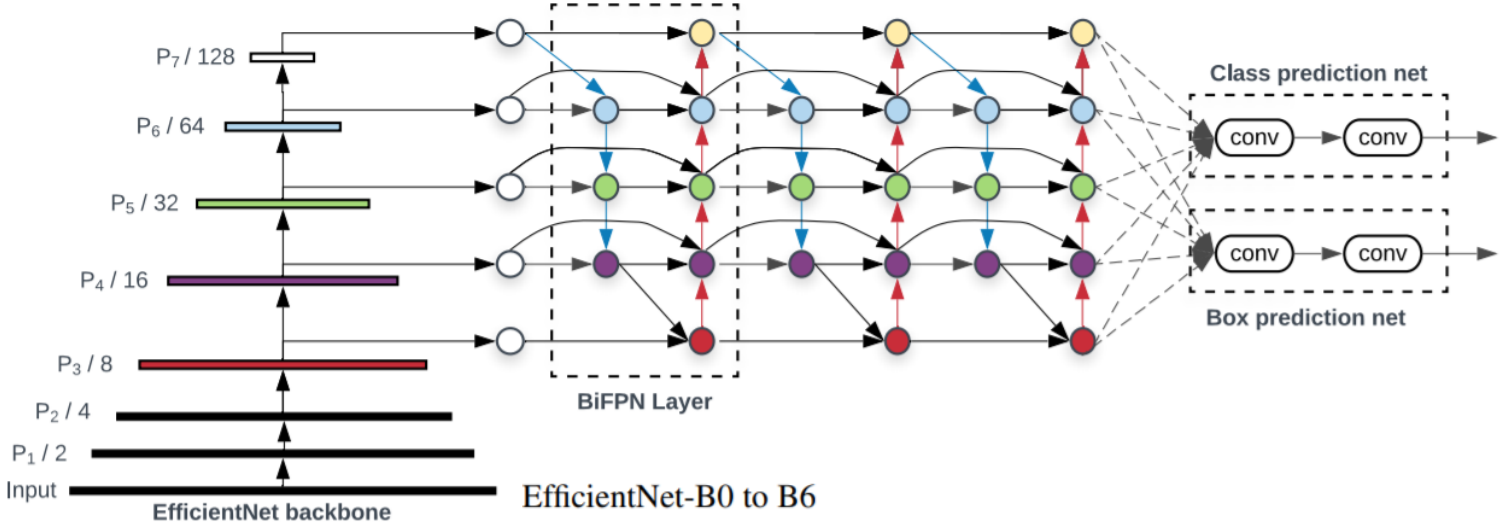


Figure 3: **EfficientDet architecture –** It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

② Compound Scaling and ③ EfficientNet backbone

Backbone network – we reuse the same width/depth scaling coefficients of EfficientNet-B0 to B6 [39] such that we can easily reuse their ImageNet-pretrained checkpoints.

BiFPN network – we linearly increase BiFPN depth D_{bifpn} (#layers) since depth needs to be rounded to small integers. For BiFPN width W_{bifpn} (#channels), exponentially grow BiFPN width W_{bifpn} (#channels) as similar to [39]. Specifically, we perform a grid search on a list of values $\{1.2, 1.25, 1.3, 1.35, 1.4, 1.45\}$, and pick the best value 1.35 as the BiFPN width scaling factor. Formally, BiFPN width and depth are scaled with the following equation:

$$W_{bifpn} = 64 \cdot \left(1.35^{\phi}\right), \qquad D_{bifpn} = 3 + \phi \qquad (1)$$

EfficientNet (Tan et al., ICML 2019)

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

Box/class prediction network – we fix their width to be always the same as BiFPN (i.e., $W_{pred} = W_{bifpn}$), but linearly increase the depth (#layers) using equation:

$$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor \qquad (2)$$

Input image resolution – Since feature level 3-7 are used in BiFPN, the input resolution must be dividable by $2^7 = 128$, so we linearly increase resolutions using equation:

$$R_{input} = 512 + \phi \cdot 128 \qquad (3)$$

② Compound Scaling and ③ EfficientNet backbone

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D7 ($\phi = 7$)	1536	B6	384	8	5
D7x	1536	B7	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D6 –** ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN, box/class net, and input size* are scaled up using equation 1, 2, 3 respectively.

$$W_{bifpn} = 64 \cdot \left(1.35^{\phi}\right), \qquad D_{bifpn} = 3 + \phi \tag{1}$$

$$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor \tag{2}$$

$$R_{input} = 512 + \phi \cdot 128 \tag{3}$$

4 Experimental results

1. Efficiency and accuracy trade-off for Object Detection

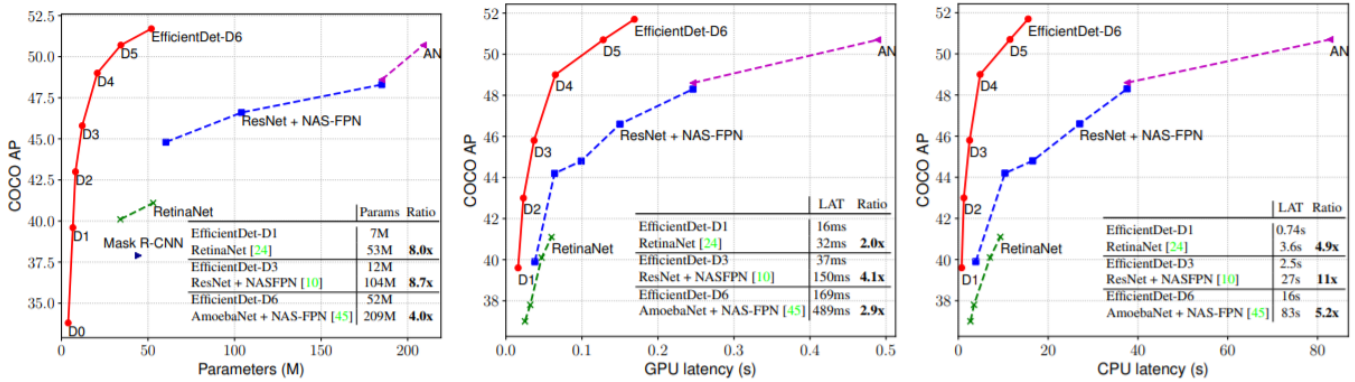
Model	test-dev AP	test-dev AP_{50}	test-dev AP_{75}	val AP	Params	Ratio	FLOPs	Ratio	Latency (ms) TitianV	Latency (ms) V100
EfficientDet-D0 (512)	**34.6**	**53.0**	**37.1**	**34.3**	**3.9M**	**1x**	**2.5B**	**1x**	**12**	**10.2**
YOLOv3 [34]	33.0	57.9	34.4	-	-	-	71B	28x	-	-
EfficientDet-D1 (640)	**40.5**	**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‡
Detectron2 Mask R-CNN X101-FPN [1]	-	-	-	44.3	107M	13x	277B	25x	-	103‡
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‡
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.

4 Experimental results

1. Efficiency and accuracy trade-off for Object Detection



(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.

2. Ablation studies on COCO validation set

2.1. EfficientNet backbone

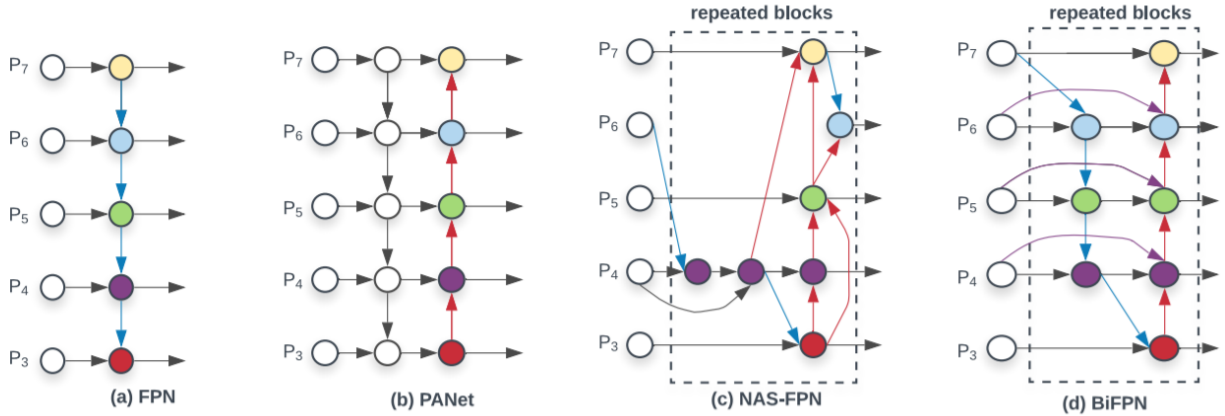
	AP	Parameters	FLOPs
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + **BiFPN**	44.4	12M	24B

Table 4: **Disentangling backbone and BiFPN –** Starting from the standard RetinaNet (ResNet50+FPN), we first replace the backbone with EfficientNet-B3, and then replace the baseline FPN with our proposed BiFPN.

2. Ablation studies on COCO validation set

2.2. BiFPN cross-scale connections

	AP	#Params ratio	#FLOPs ratio
Repeated top-down FPN	42.29	1.0x	1.0x
Repeated FPN+PANet	44.08	1.0x	1.0x
NAS-FPN	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**



2. Ablation studies on COCO validation set

2.3. BiFPN fast normalized feature fusion

Unbounded fusion: $O = \sum_i w_i \cdot I_i,$

Softmax-based fusion: $O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i.$

Fast normalized fusion: $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i.$

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.

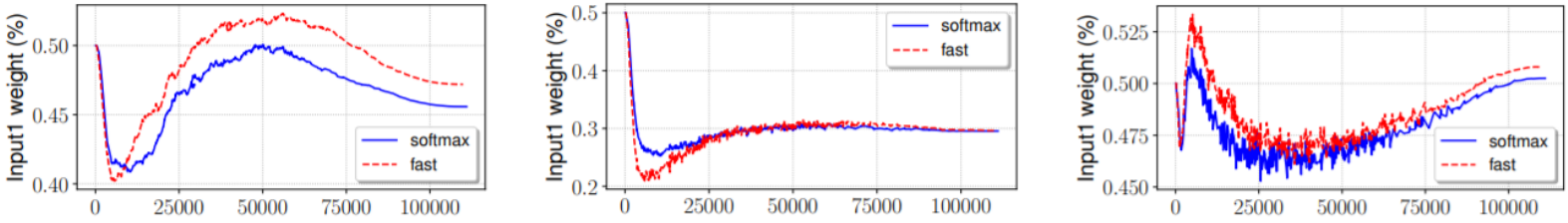


Figure 5: **Softmax vs. fast normalized feature fusion –** (a) - (c) shows normalized weights (i.e., importance) during training for three representative nodes; each node has two inputs (input1 & input2) and their normalized weights always sum up to 1.

2. Ablation studies on COCO validation set

2.4. Compound scaling

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D7 ($\phi = 7$)	1536	B6	384	8	5
D7x	1536	B7	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D6 –** ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN, box/class net, and input size* are scaled up using equation 1, 2, 3 respectively.

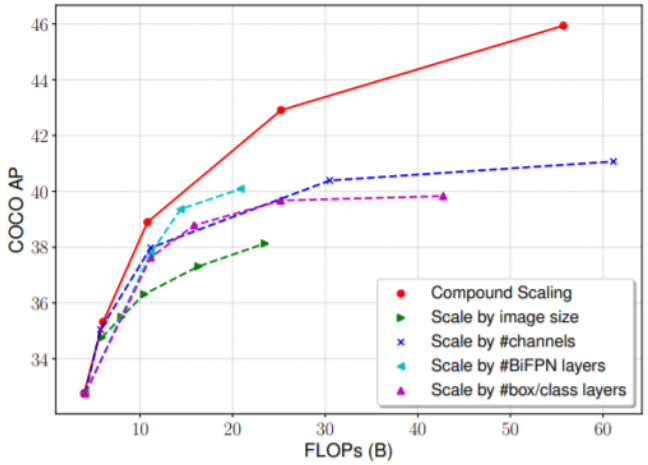


Figure 6: **Comparison of different scaling methods –** compound scaling achieves better accuracy and efficiency.

3. EfficientDet for Semantic Segmentation

Model	mIOU	Params	FLOPs
DeepLabV3+ (ResNet-101) [6]	79.35%	-	298B
DeepLabV3+ (Xception) [6]	80.02%	-	177B
Our EfficientDet†	**81.74%**	**17M**	**18B**

†A modified version of EfficientDet-D4.

Table 3: **Performance comparison on Pascal VOC semantic segmentation**.

Q&A