EfficientDet:
Scalable and Efficient Object Detection

Mingxing Tan, Ruoming Pang, Quoc V.Le

[Google Research, Brain Team]

arXiv:1911.09070v1

Sungman, Cho.

Introduction

Introduction

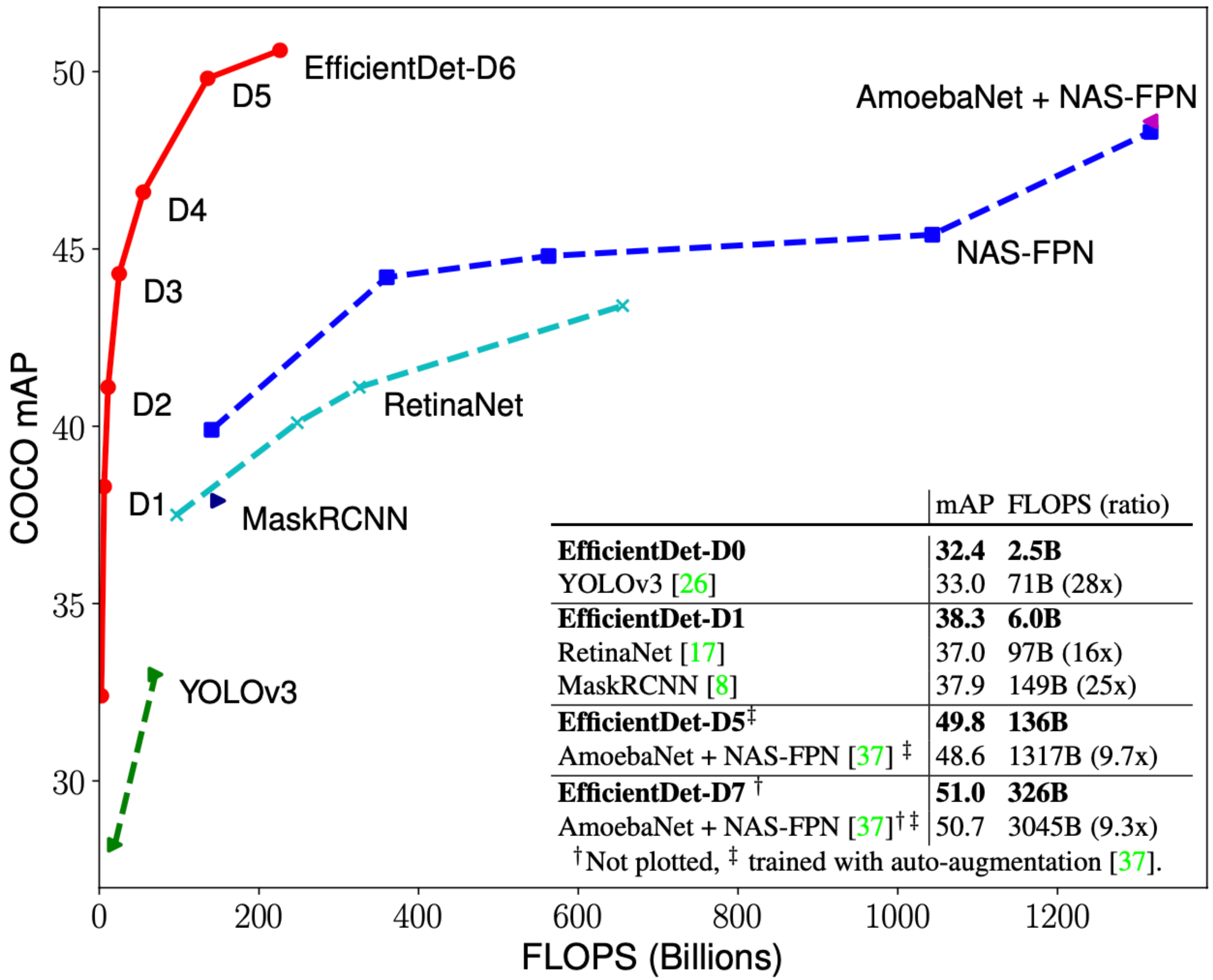
- Propose several key optimizations to improve efficiency.

1. BiFPN (Bi-directional Feature Pyramid Network)

2. Compound scaling method

Achieves **state-of-the-art** 51.0 mAP on COO dataset with 52M parameters and 326B FLOPS,
4x smaller and **9.3x fewer FLOPS** , **more accurate (+0.3% mAP)**

Model FLOPS vs COCO acc.



Introduction

- **Eficient multi-scale feature fusion**
  - PANet, NAS-FPN, …
  - Most previous works simply sum features up without distinction.

- **Model scaling**
  - Accuracy ← Trade Off → Efficiency

Contribution

- We proposed **BiFPN**, a weighted bidirectional feature network for easy and fast multi-scale feature fusion.
- We propose a new **compund scaling method**, which jointly scales up backbone, feature network, box/class network, and resolution, in a principled way.
- Based on BiFPN and compound scaling, we developed EfficientDet.

Related Work

- **<u>One-Stage Detectors</u>**
  - YOLO v1 ~ v3
  - SSD
  - RetinaNet

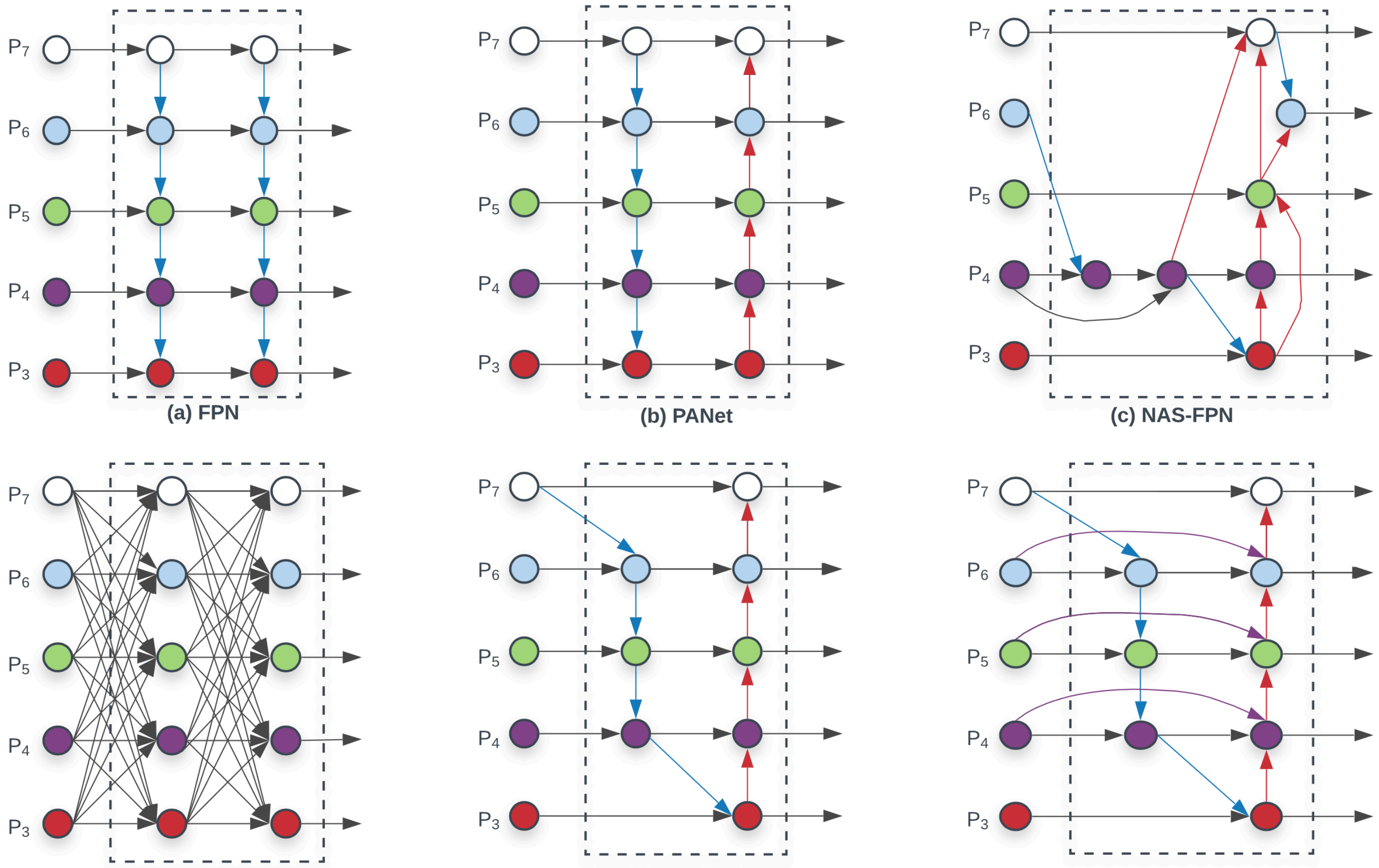
- **<u>Model Scaling</u>**
  - ResNet → ResNeXt → AmoebaNet
  - increasing input image size

- **<u>Multi-Scale Feature Representations</u>**
  - FPN (Feature Pyramid Network)
  - PANet : add an extra bottom-up path aggregation network
  - STDL : propose a scale-transfer module to exploit cross-scale features
  - M2det : poposes a U-shape module to fuse multi-scale features
  - G-FRNet : introduces gate units for controlling inforamtion flow
  - NAS-FPN : leverages NAS to automatically design network topology

Methodology

1. BiFPN

BiFPN



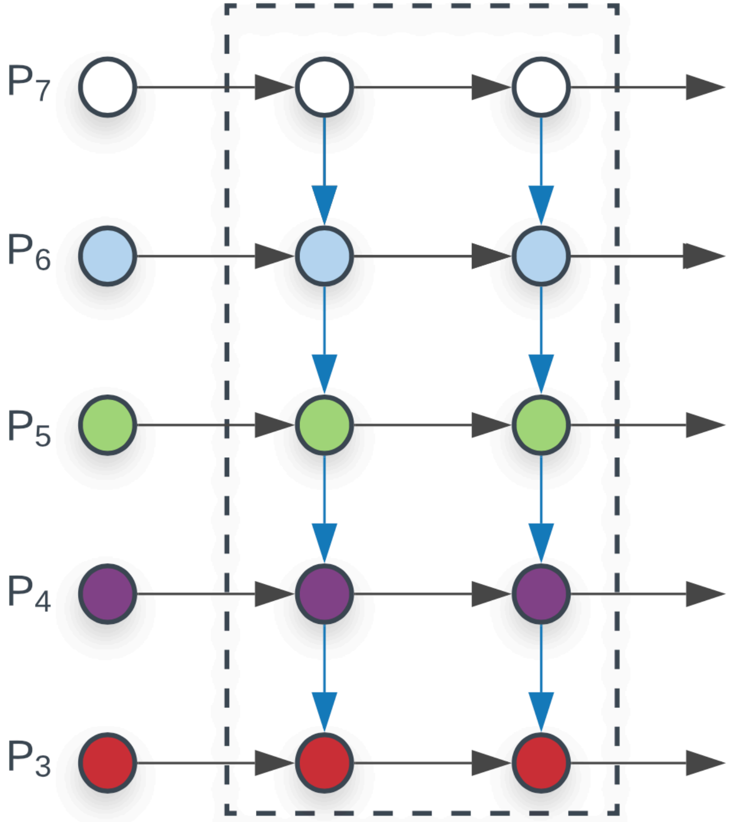
(a) FPN (b) PANet (c) NAS-FPN (d) Fully-connected FPN (e) Simplified PANet (f) BiFPN

$$\vec{P}^{in} = (P^{in}_{l_1}, P^{in}_{l_2}, \ldots)$$
$$\vec{P}^{out} = f(\vec{P}^{in})$$

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

BiFPN

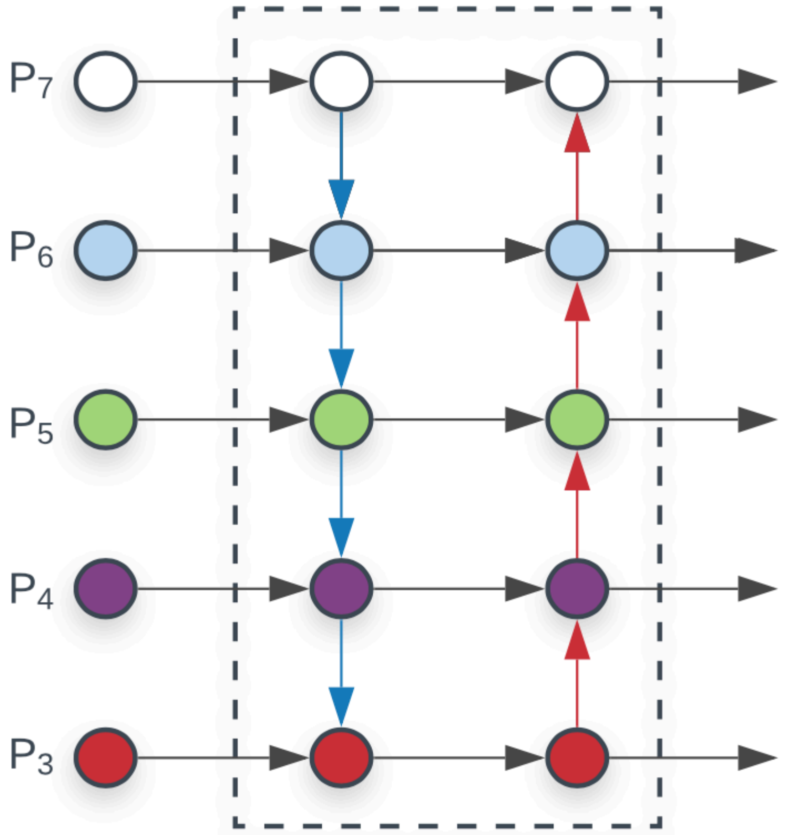
- FPN is inherently limited by the one-way information flow.



(a) FPN

BiFPN

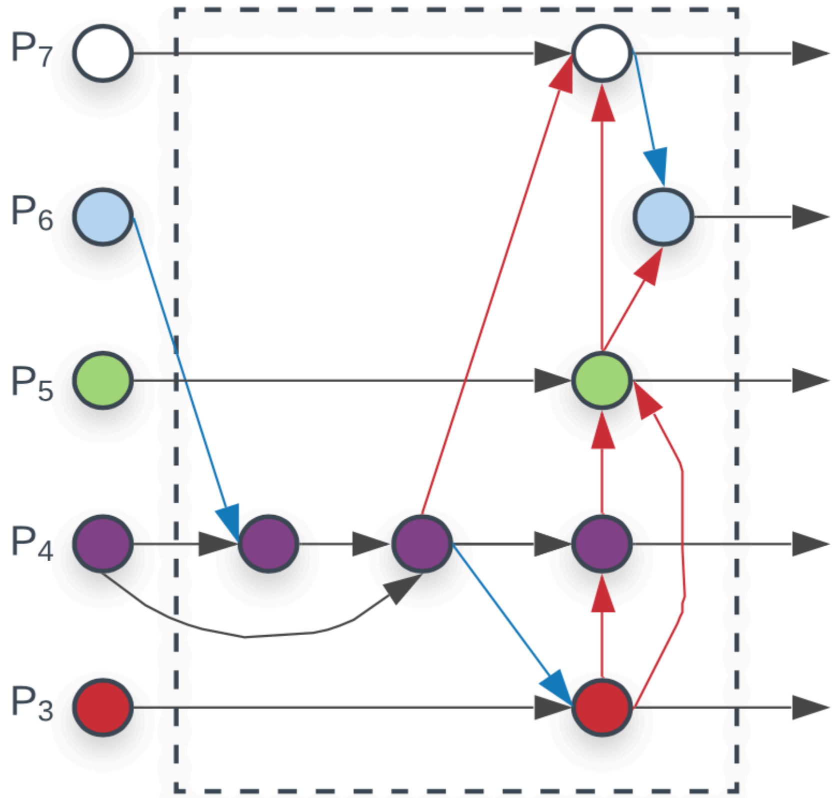
- PANet adds an extra bottom-up path aggregation network.



(b) PANet

BiFPN

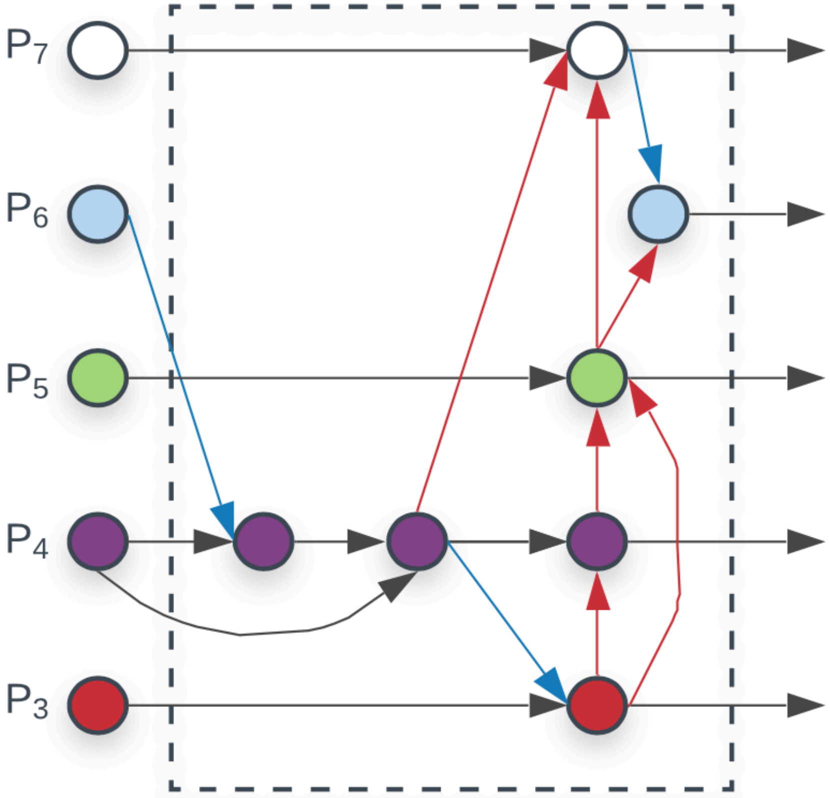
- NAS-FPN employs NAS to search for better cross-scale feature network topology.



(c) NAS-FPN

BiFPN

- NAS-FPN employs NAS to search for better cross-scale feature network topology.



(c) NAS-FPN

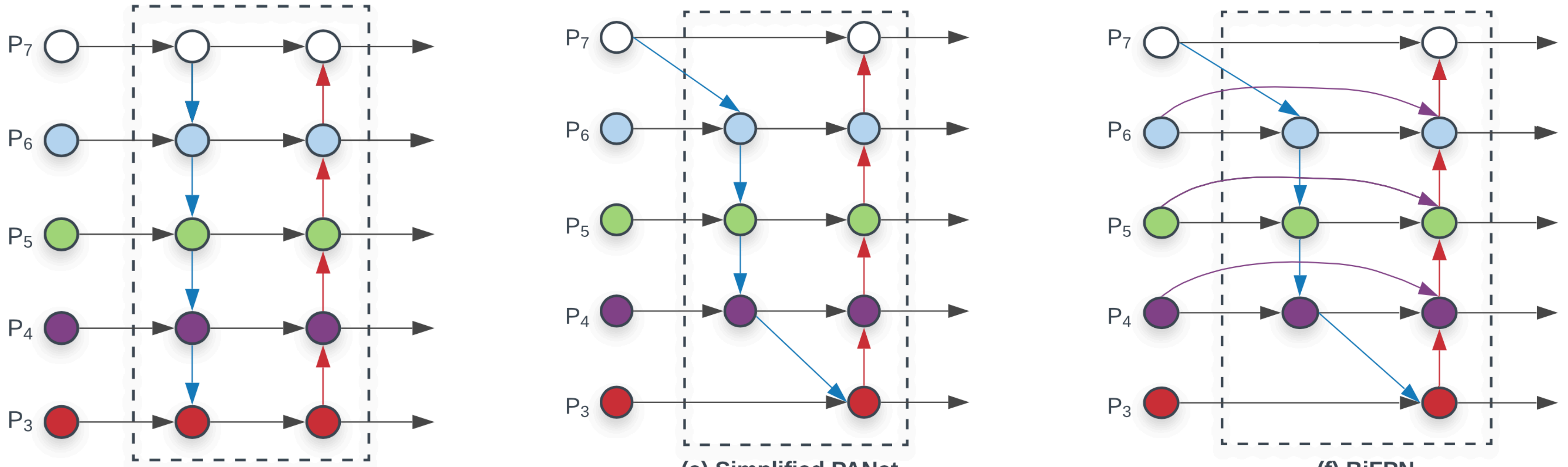
BiFPN

	mAP	#Params ratio	#FLOPS 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**

Table 4: **Comparison of different feature networks –** Our weighted BiFPN achieves the best accuracy with fewer parameters and FLOPS.

PANet is the best accuracy architecutre.

BiFPN



(b) PANet (e) Simplified PANet (f) BiFPN

Weighted Feature Fusion

- Previous feature fusion methods treat all input features equally without distinction.
- However, different input features are at different resoultions, they usually contribute to the output feature unequally.
- Propose to add an additional weight for each input during 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** $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$

Weighted Feature Fusion

- **Unbounded fusion** $O = \sum_i w_i \cdot I_i,$

  since scalar weight is unbounded, it could potentially cause training instability.
  Therefore, we resort to weight normalization to bound the value range of each weight.

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

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

Weighted Feature Fusion

- **Unbounded fusion** $O = \sum_i w_i \cdot I_i,$

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

  Apply softmax to each weight, representing the importance of each input.
  However, it leads to significant slowdown on GPU.

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

Weighted Feature Fusion

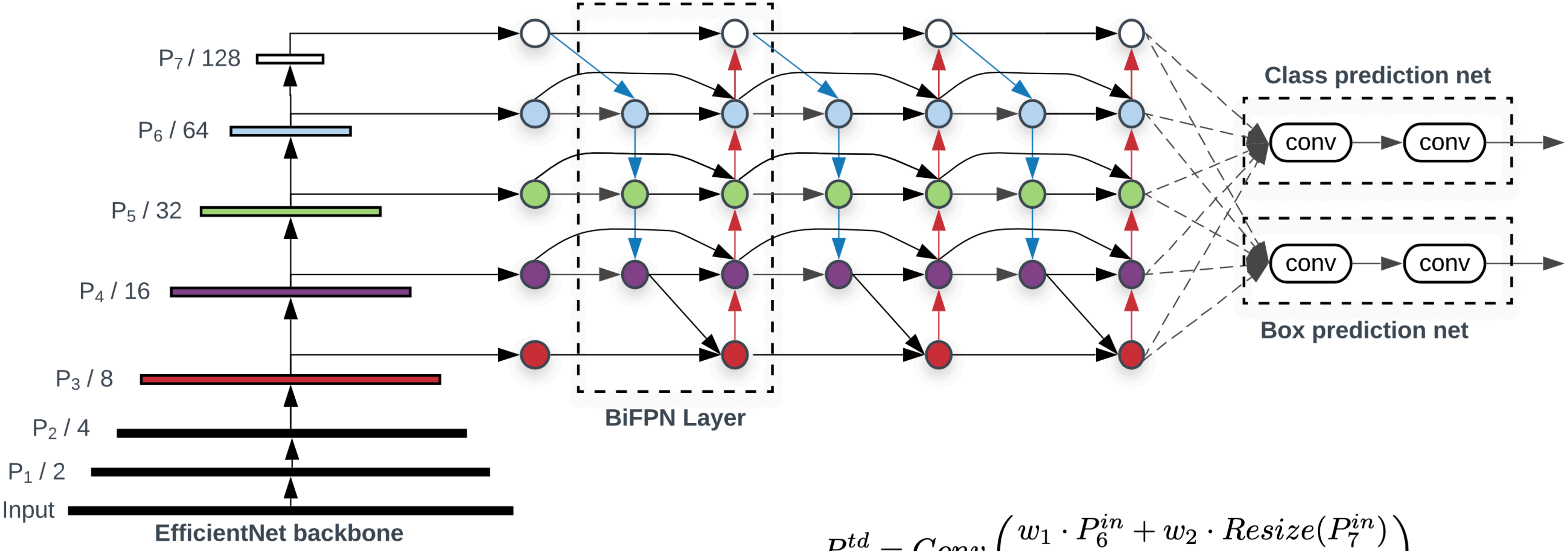
- **Unbounded fusion** $O = \sum_i w_i \cdot I_i,$

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

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

ϵ = 0.0001 is a small value to avoid numerical instability.
fast fusion approach has similar acc. , but runs up to 30% faster on GPU. (vs softmax-based function)

BiFPN



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

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

Methodology

2. Compound Scaling

Compound Scaling

- Previous works mostly scale up a baseline detector by employing bigger backbone networks.

- EfficientNet shows remarkable performance on image classification by **jointly scaling up all dimensions** of network width, depth, and input resolution.

- Unlike EfficientNet, **object detectors have much more scaling dimensions** than image classification models, so **grid search for all dimensions is prohibitive expensive**.

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	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

BiFPN network $W_{bifpn} = 64 \cdot \left(1.35^{\phi}\right), \qquad D_{bifpn} = 2 + \phi$

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

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

Experiments

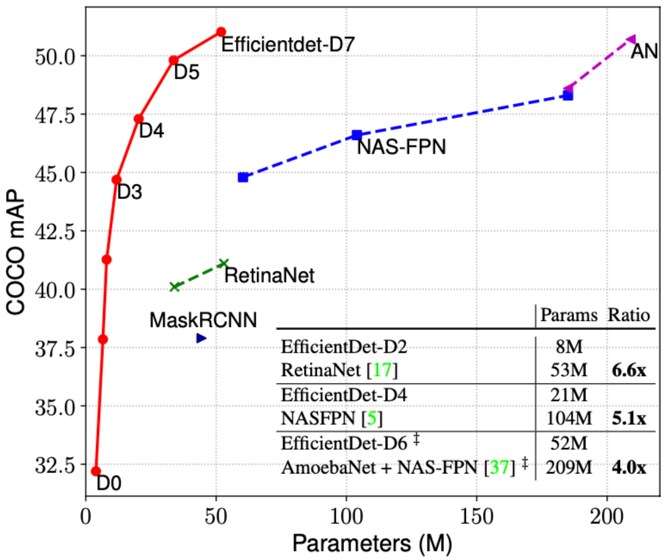
Experiments

Performance on COCO 2017

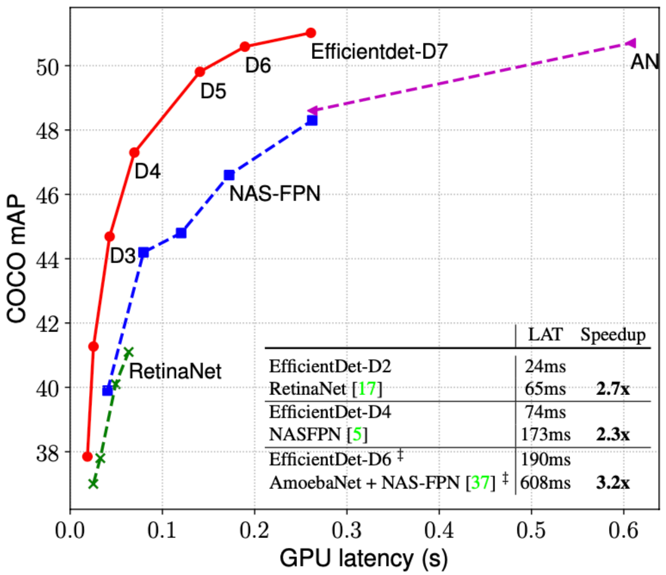
Model	mAP	#Params	Ratio	#FLOPS	Ratio	GPU LAT(ms)	Speedup	CPU LAT(s)	Speedup
EfficientDet-D0	**32.4**	**3.9M**	**1x**	**2.5B**	**1x**	**16** $\pm$**1.6**	**1x**	**0.32** $\pm$**0.002**	**1x**
YOLOv3 [26]	33.0	-	-	71B	28x	$51^\dagger$	-	-	-
EfficientDet-D1	**38.3**	**6.6M**	**1x**	**6B**	**1x**	**20** $\pm$**1.1**	**1x**	**0.74** $\pm$**0.003**	**1x**
MaskRCNN [8]	37.9	44.4M	6.7x	149B	25x	$92^\dagger$	-	-	-
RetinaNet-R50 (640) [17]	37.0	34.0M	6.7x	97B	16x	27 $\pm$1.1	1.4x	2.8 $\pm$0.017	3.8x
RetinaNet-R101 (640) [17]	37.9	53.0M	8x	127B	21x	34 $\pm$0.5	1.7x	3.6 $\pm$0.012	4.9x
EfficientDet-D2	**41.1**	**8.1M**	**1x**	**11B**	**1x**	**24** $\pm$**0.5**	**1x**	**1.2** $\pm$**0.003**	**1x**
RetinaNet-R50 (1024) [17]	40.1	34.0M	4.3x	248B	23x	51 $\pm$0.9	2.0x	7.5 $\pm$0.006	6.3x
RetinaNet-R101 (1024) [17]	41.1	53.0M	6.6x	326B	30x	65 $\pm$0.4	2.7x	9.7 $\pm$0.038	8.1x
NAS-FPN R-50 (640) [5]	39.9	60.3M	7.5x	141B	13x	41 $\pm$0.6	1.7x	4.1 $\pm$0.027	3.4x
EfficientDet-D3	**44.3**	**12.0M**	**1x**	**25B**	**1x**	**42** $\pm$**0.8**	**1x**	**2.5** $\pm$**0.002**	**1x**
NAS-FPN R-50 (1024) [5]	44.2	60.3M	5.1x	360B	15x	79 $\pm$0.3	1.9x	11 $\pm$0.063	4.4x
NAS-FPN R-50 (1280) [5]	44.8	60.3M	5.1x	563B	23x	119 $\pm$0.9	2.8x	17 $\pm$0.150	6.8x
EfficientDet-D4	**46.6**	**20.7M**	**1x**	**55B**	**1x**	**74** $\pm$**0.5**	**1x**	**4.8** $\pm$**0.003**	**1x**
NAS-FPN R50 (1280@384)	45.4	104 M	5.1x	1043B	19x	173 $\pm$0.7	2.3x	27 $\pm$0.056	5.6x
EfficientDet-D5 + AA	**49.8**	**33.7M**	**1x**	**136B**	**1x**	**141** $\pm$**2.1**	**1x**	**11** $\pm$**0.002**	**1x**
AmoebaNet+ NAS-FPN + AA(1280) [37]	48.6	185M	5.5x	1317B	9.7x	259 $\pm$1.2	1.8x	38 $\pm$0.084	3.5x
EfficientDet-D6 + AA	**50.6**	**51.9M**	**1x**	**227B**	**1x**	**190** $\pm$**1.1**	**1x**	**16**$\pm$**0.003**	**1x**
AmoebaNet+ NAS-FPN + AA(1536) [37]	50.7	209M	4.0x	3045B	13x	608 $\pm$1.4	3.2x	83 $\pm$0.092	5.2x
EfficientDet-D7 + AA	**51.0**	**51.9M**	**1x**	**326B**	**1x**	**262** $\pm$**2.2**	**1x**	**24** $\pm$**0.003**	**1x**

Experiments

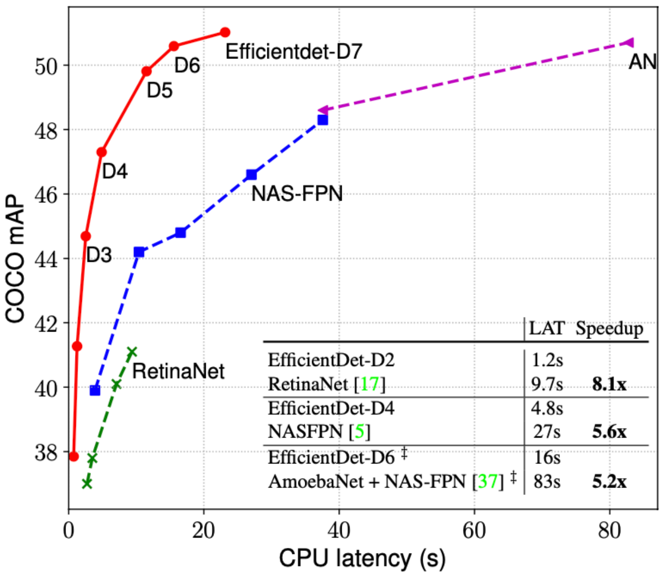
Model size and inference latency



(a) Model Size



(b) GPU Latency



(c) CPU Latency

Experiments

Disentangling backbone and BiFPN

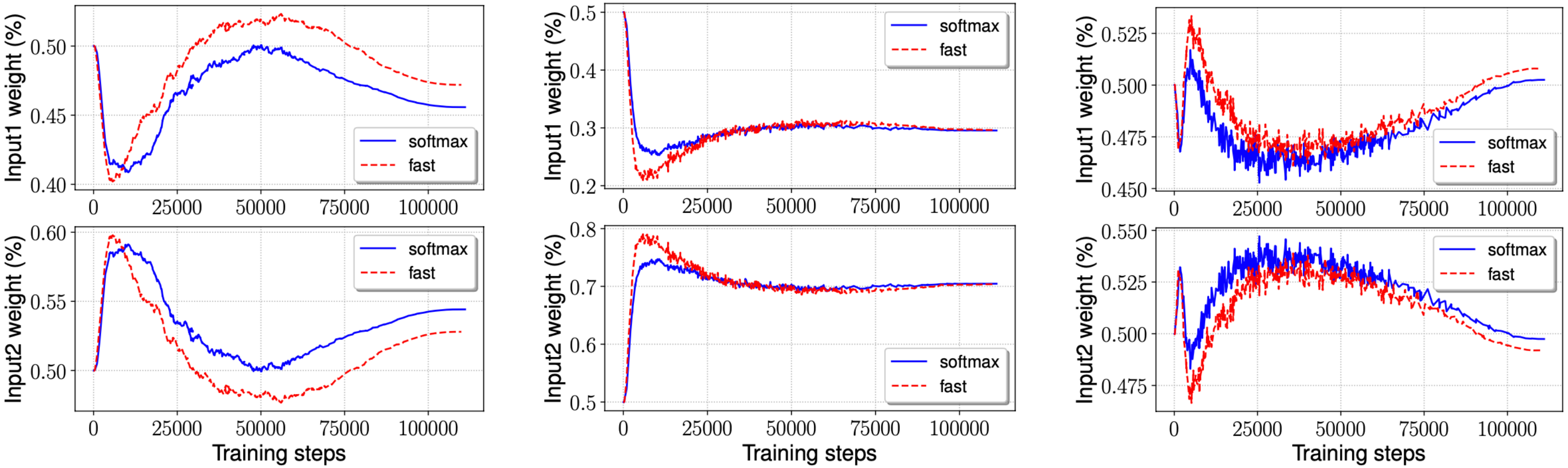
	mAP	Parameters	FLOPS
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + BiFPN	44.4	12M	24B

Comparison of different feature networks

	mAP	#Params ratio	#FLOPS 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**

Experiments

Softmax vs Fast Normalized Fusion

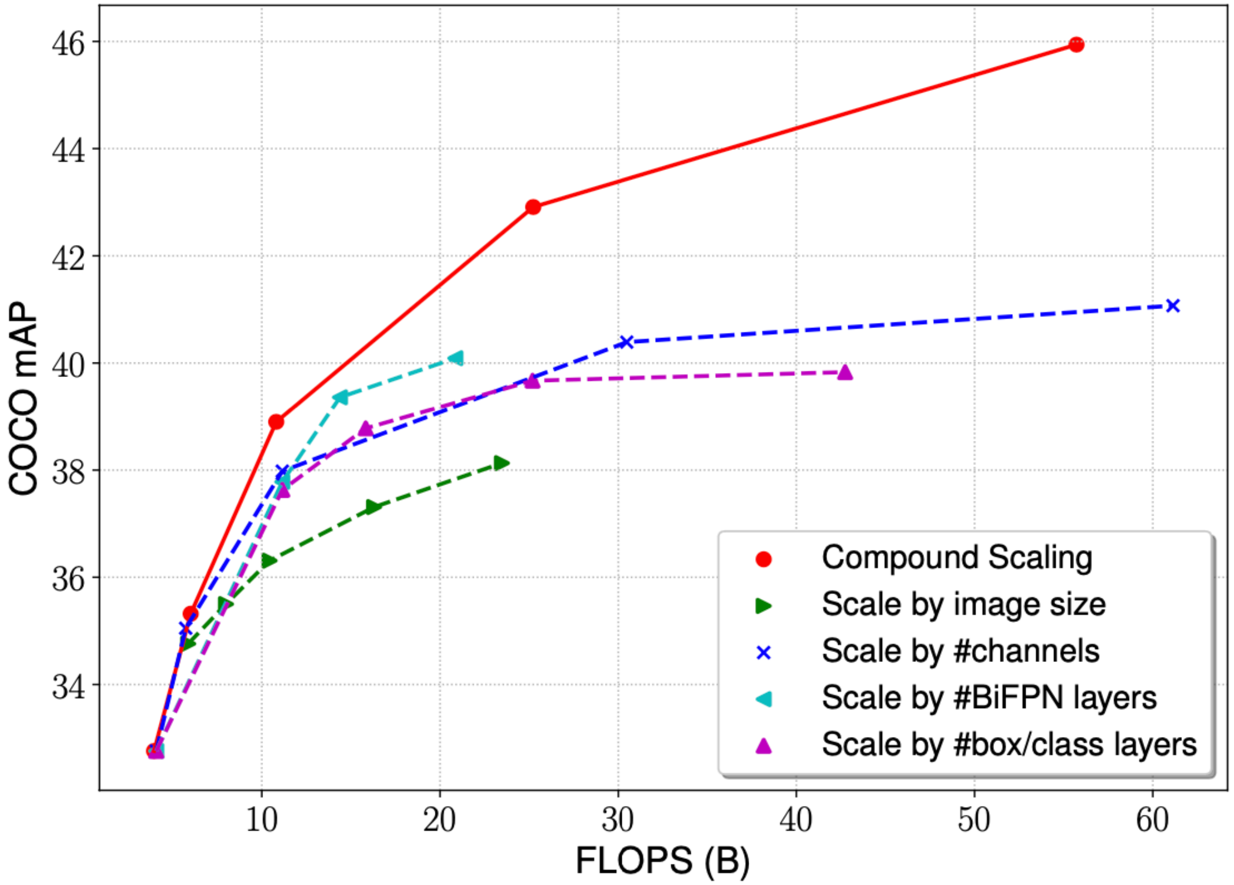


(a) Example Node 1 (b) Example Node 2 (c) Example Node 3

Model	Softmax Fusion mAP	Fast Fusion mAP (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

Experiments

Comparison of different scaling methods



Conclusion

- Propose a weighted bidirectional feature network and a customized compound scaling method, in order to improve accuracy and efficiency.

- EfficientDet 3.2x faster on GPUs and 8.1x faster on CPUs.

Thank You.