EfficientDet
Scalable and Efficient Object Detection

Introduction

The large model sizes & expensive computation costs

in many real-world applications such as ~~robotics~~  and ~~self-driving cars~~ 

model efficiency becomes increasingly important for object detection

build a *scalable detection architecture*

higher accuracy ✚ better efficiency

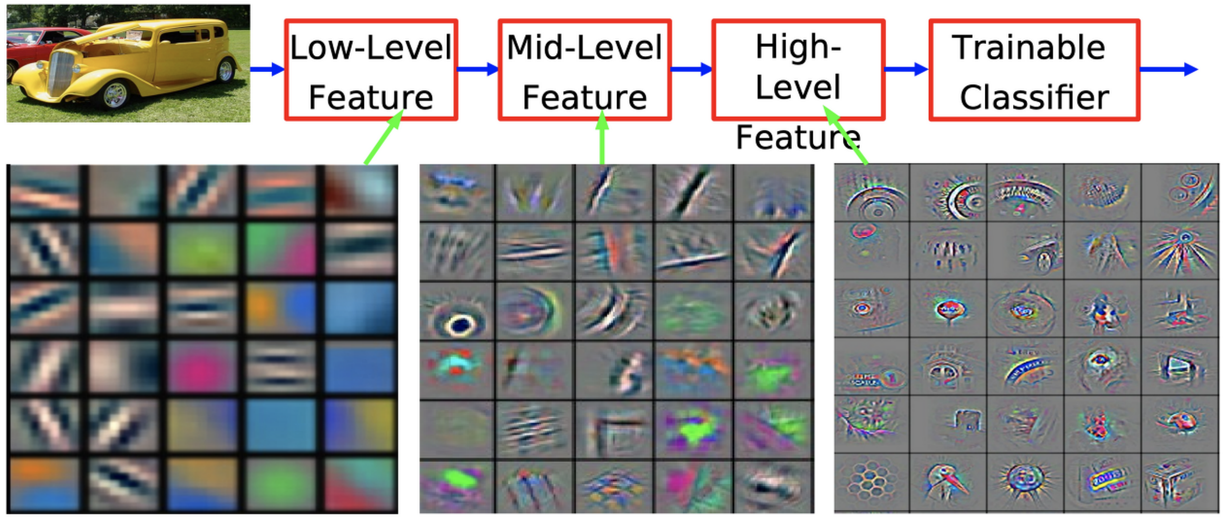
from mobile devices to datacenters

Related Work

Multi-scale feature representations

aggregate features at different resolutions

feature pyramid network (FPN)



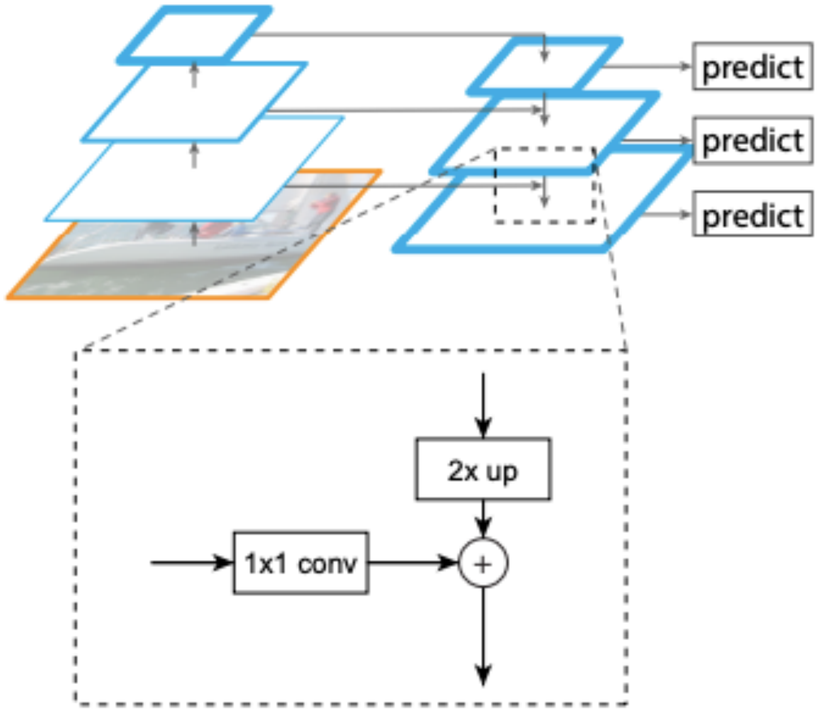
https://herbwood.tistory.com/18

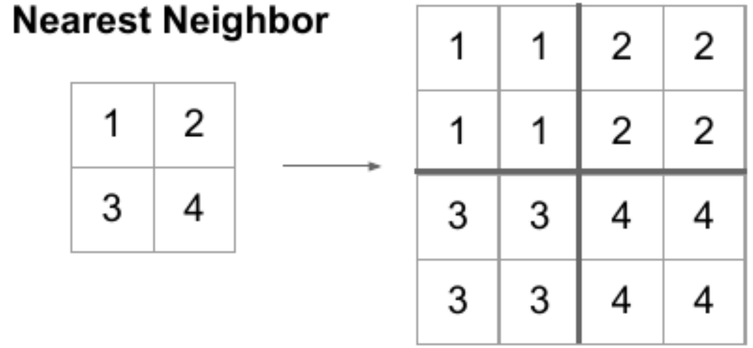
Related Work

Multi-scale feature representations

aggregate features at different resolutions

feature pyramid network (FPN)





http://koreascience.or.kr/article/JAKO202117457591215.pdf

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

two main challenges

efficient multi-scale feature fusion

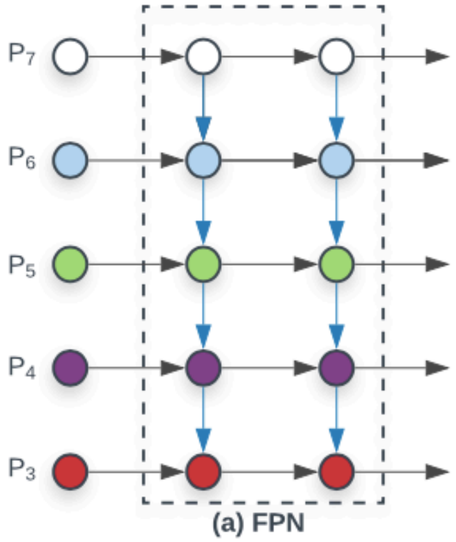
✓ BiFPN

model scaling

✓ Compound Scaling

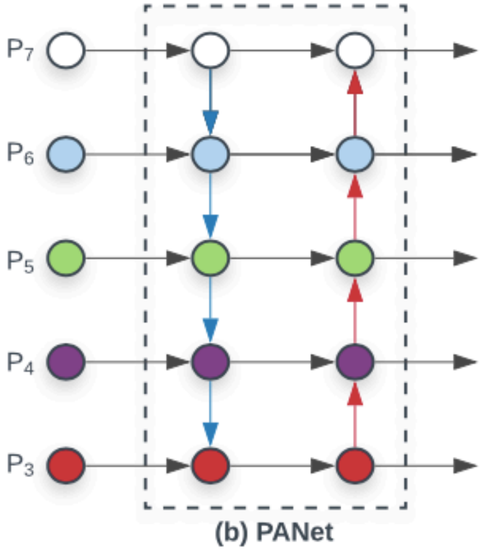
BiFPN

Cross-Scale Connections

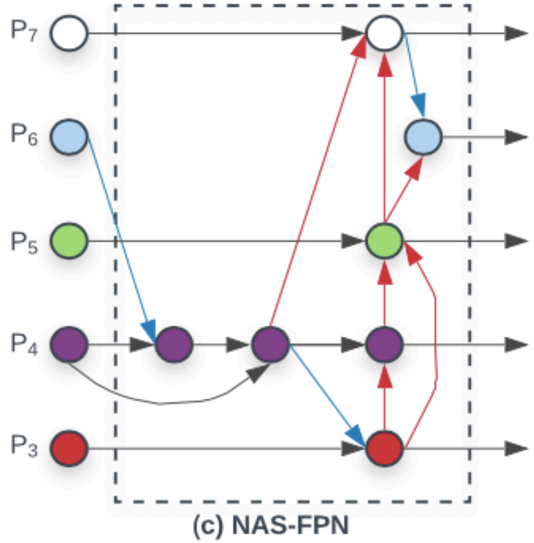


one-way

information flow



extra bottom-up path



neural architecture search

thousands of GPU hours

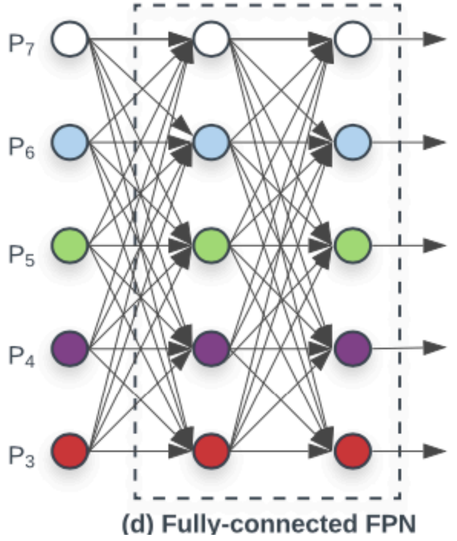
irregular

difficult to interpret or modify

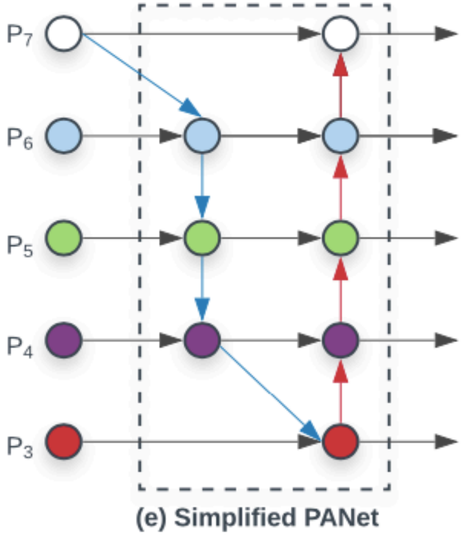
BiFPN

Cross-Scale Connections

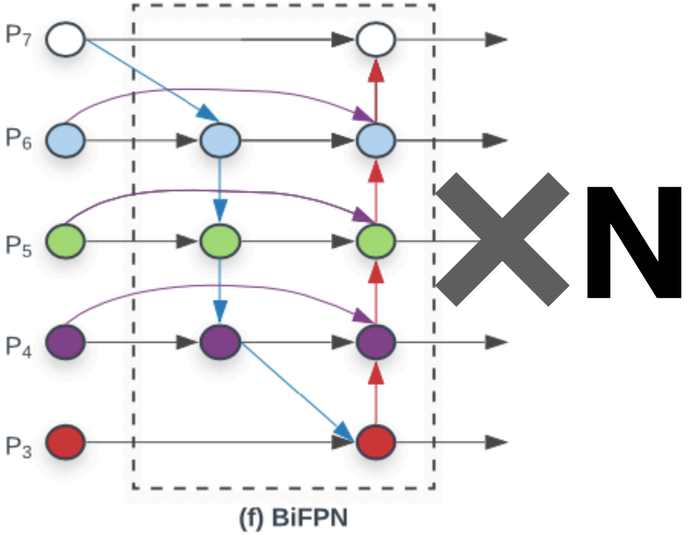
	mAP	#Params ratio	#FLOPS ratio
Top-Down FPN [16]	42.29	1.0x	1.0x
Repeated PANet [19]	44.08	1.0x	1.0x
NAS-FPN [5]	43.16	0.71x	0.72x
Fully-Connected FPN	43.06	1.24x	1.21x
BiFPN (w/o weighted)	**43.94**	**0.88x**	**0.67x**
BiFPN (w/ weighted)	**44.39**	**0.88x**	**0.68x**



(d) Fully-connected FPN



(e) Simplified PANet



(f) BiFPN

Remove nodes
that only have one input edge

less contribution
to feature network

add an extra edge
if they are at the same level

fuse more features
without adding much cost

BiFPN

Weighted Feature Fusion

a common way

resize the same resolution ⟶ sum

treat all input features equally

they contribute to the output feature unequally

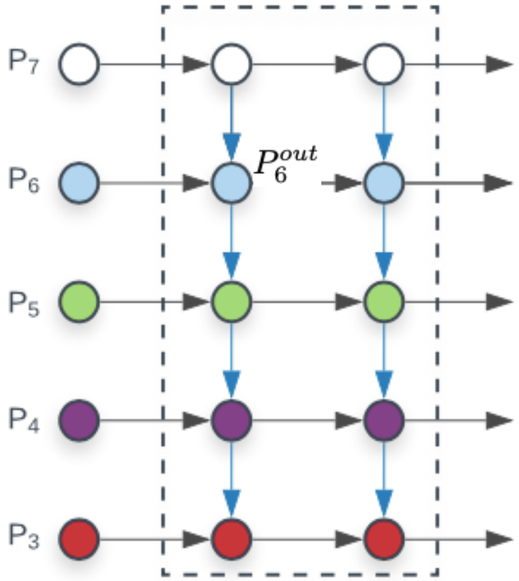
weighted

add an additional weight for each input

let the network to learn the importance of each input feature

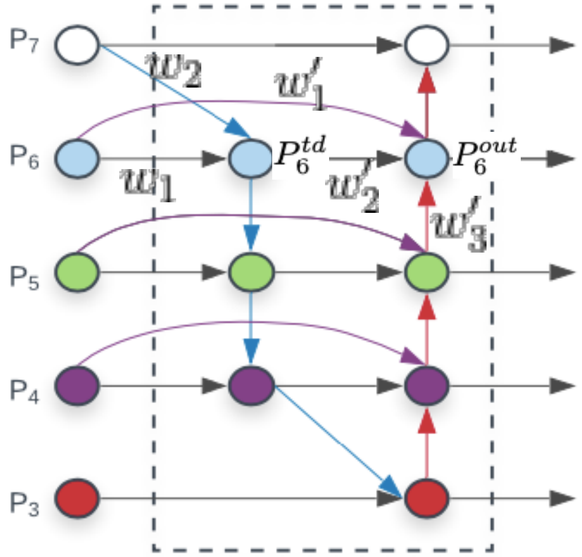
BiFPN

Weighted Feature Fusion



(a) FPN

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



(f) BiFPN

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

BiFPN

Weighted Feature Fusion

Unbounded fusion

$O = \sum_i w_i \cdot I_i$ w_i : learnable weight

scalar weight is unbounded

→ training instability

Softmax-based fusion

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

apply softmax to each weight

normalized to be a probability
with value range from 0 to 1

→ significant slowdown on GPU hardware

BiFPN

Weighted Feature Fusion

Fast normalized fusion

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$$

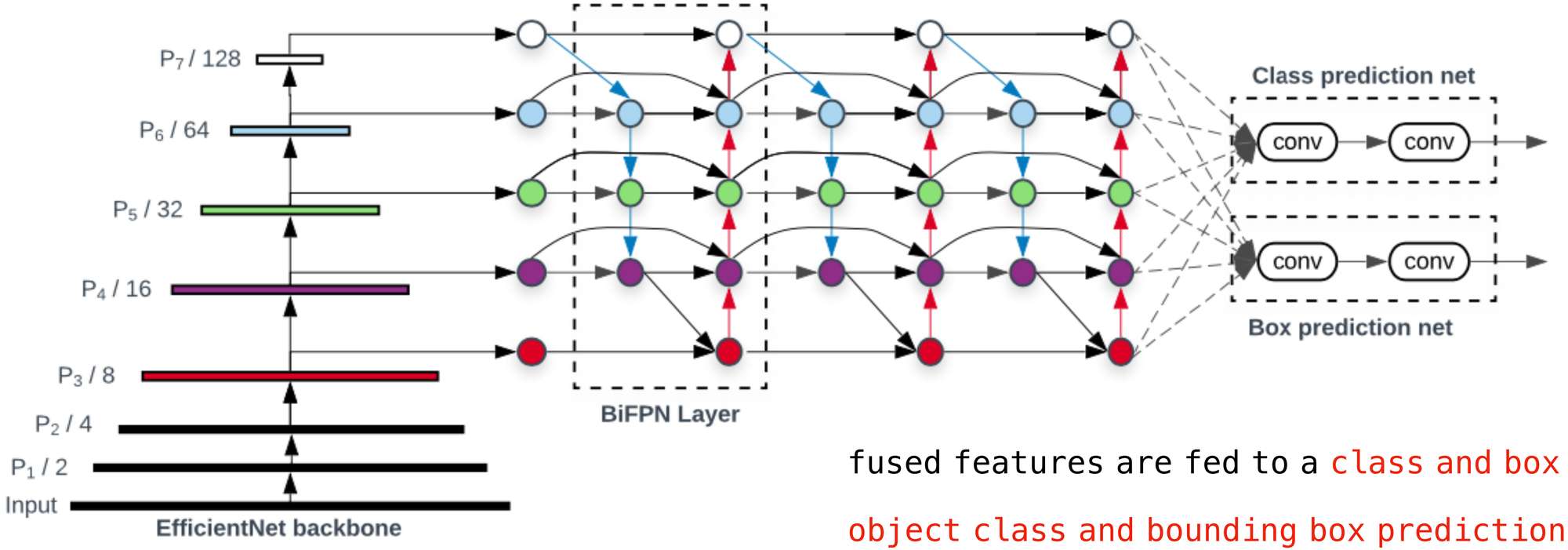
$w_i \geq 0$ is ensured by applying a Relu after each w_i

$\epsilon = 0.0001$ is a small value to avoid numerical instability

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

very similar accuracy
but runs up to 30% faster on GPUs

EfficientDet Architecture



fused features are fed to a class and box network

object class and bounding box predictions respectively

ImageNet-pretrained EfficientNets

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

EfficientDet Architecture

Compound Scaling

Previous works

bigger backbone networks
larger input images
stacking more FPN layers

↓

limited scaling dimensions
→ ineffective

Compound scaling

jointly scaling up all dimensions
of network width, depth, and input resolution

EfficientDet Architecture

Compound Scaling

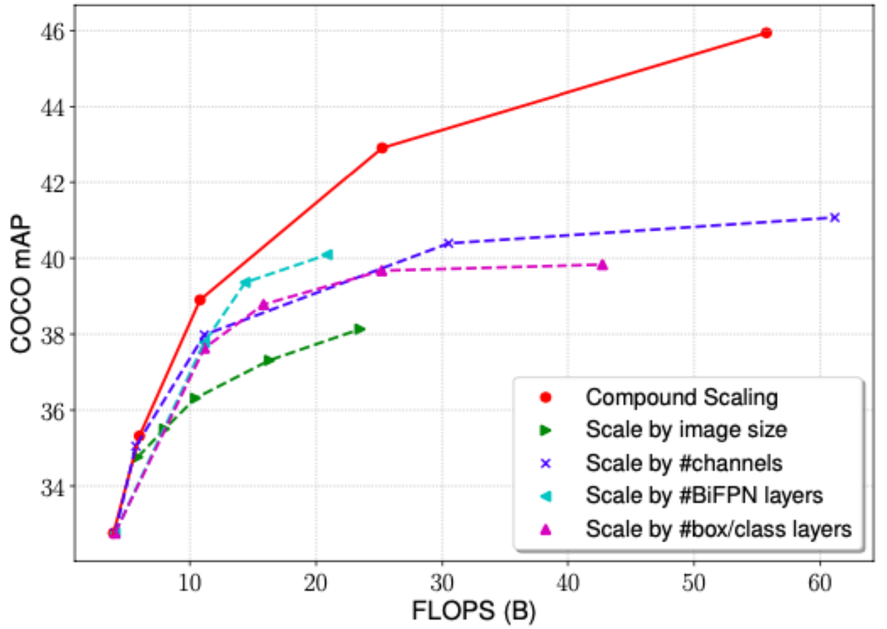
simple compound coefficient φ

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

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

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

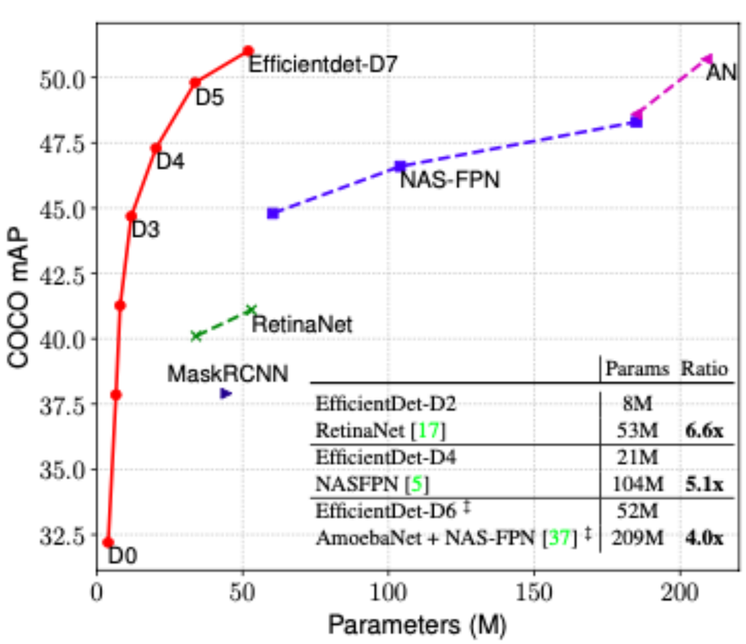
	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5



Conclusion

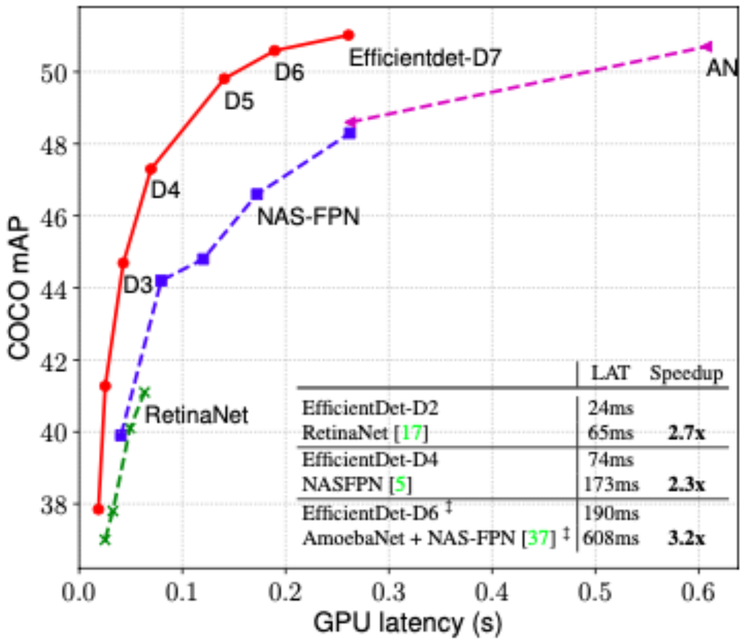
Model	mAP	#Params	Ratio	#FLOPS	Ratio	GPU LAT(ms)	Speedup	CPU LAT(s)	Speedup
EfficientDet-D0	**32.4**	**3.9M**	**1x**	**2.5B**	**1x**	**16** ±1.6	**1x**	**0.32** ±0.002	**1x**
YOLOv3 [26]	33.0	-	-	71B	28x	51†	-	-	-
EfficientDet-D1	**38.3**	**6.6M**	**1x**	**6B**	**1x**	**20** ±1.1	**1x**	**0.74** ±0.003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

Conclusion



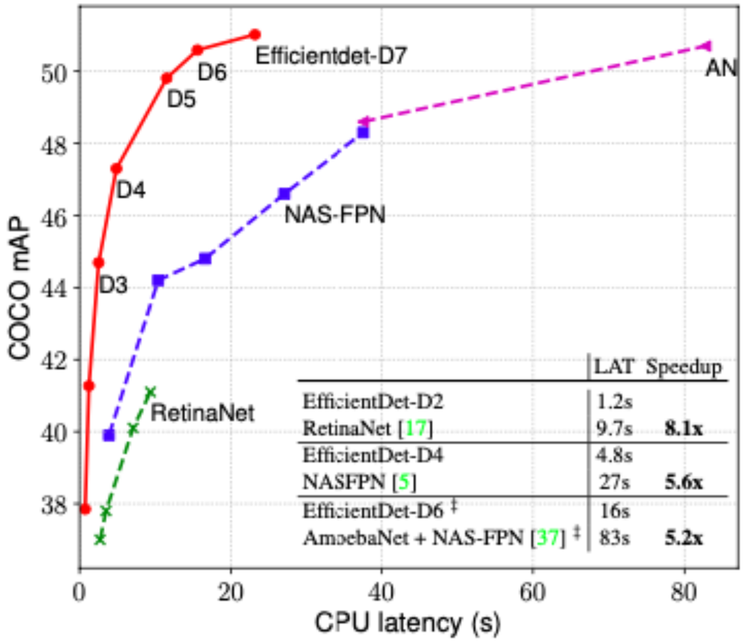
	Params	Ratio
EfficientDet-D2	8M	
RetinaNet [17]	53M	**6.6x**
EfficientDet-D4	21M	
NASFPN [5]	104M	**5.1x**
EfficientDet-D6 ‡	52M	
AmoebaNet + NAS-FPN [37] ‡	209M	**4.0x**

(a) Model Size



	LAT	Speedup
EfficientDet-D2	24ms	
RetinaNet [17]	65ms	**2.7x**
EfficientDet-D4	74ms	
NASFPN [5]	173ms	**2.3x**
EfficientDet-D6 ‡	190ms	
AmoebaNet + NAS-FPN [37] ‡	608ms	**3.2x**

(b) GPU Latency



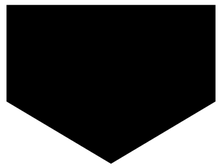
	LAT	Speedup
EfficientDet-D2	1.2s	
RetinaNet [17]	9.7s	**8.1x**
EfficientDet-D4	4.8s	
NASFPN [5]	27s	**5.6x**
EfficientDet-D6 ‡	16s	
AmoebaNet + NAS-FPN [37] ‡	83s	**5.2x**

(c) CPU Latency

Conclusion

Weighted bidirectional feature network

Customized compound scaling method



EfficientDet

achieve better accuracy and efficiency