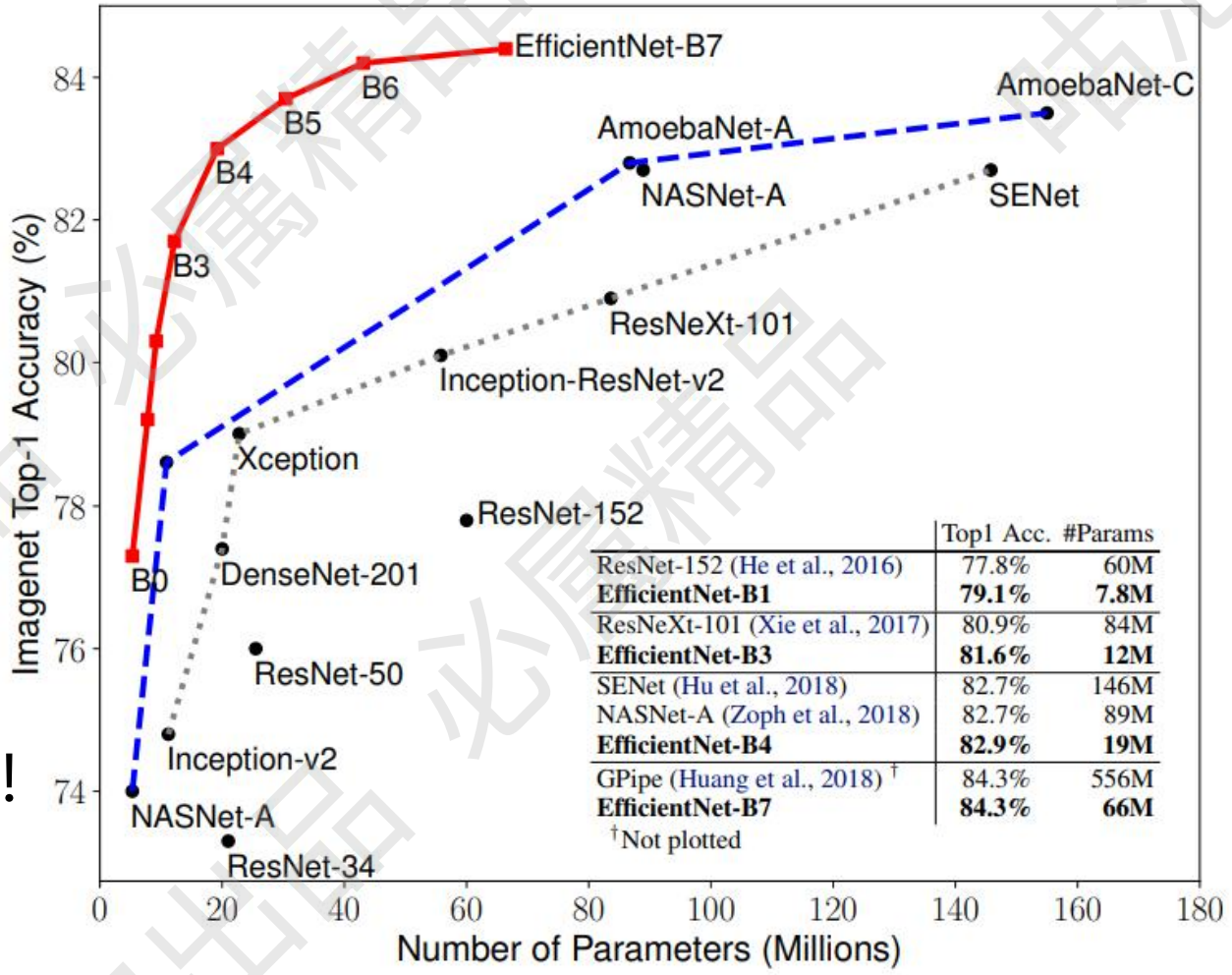
EfficientNet

- 做大事，要成功，三个条件：
  - backbone！backbone！backbone！
  - 相当于特征提取模块，无论啥任务都需要它！
  - 蒋先生告诉浩南的道理，也是深度学习告诉我们的。。。
  - EfficientNet可以说是当下武林比较强悍的backbone！

EfficientNet

- 这效果有点碾压：
  - 其中B0-B7表示各种参数的版本
  - 效果上基本碾压了前几年的大佬
  - 这事也就谷歌爸爸能干出来
  - 给我们提供了新一代的backbone！



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

EfficientNet

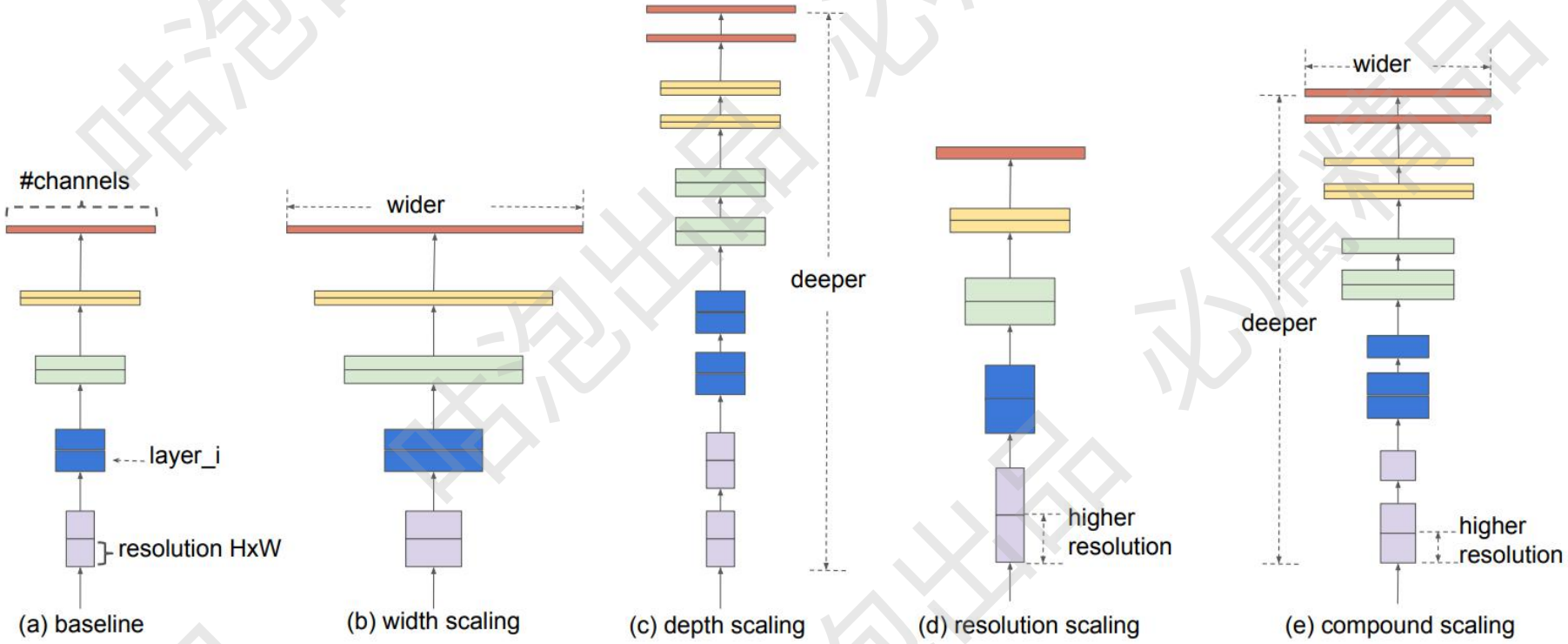
- 整体感觉：

  - 给了我们一代神器，拿来主义就好！直接用就可以，各种任务往里套！
  - 很难解释为什么是这样的组合，每一个参数的设计，调出来的参数！
  - 能在人家基础上基础延伸吗？好像挺难的，这个事一般人还真整不了！
  - 用就得了，根据对速度和精度等指标的要求选择对应版本即可！

EfficientNet

- 出发点：

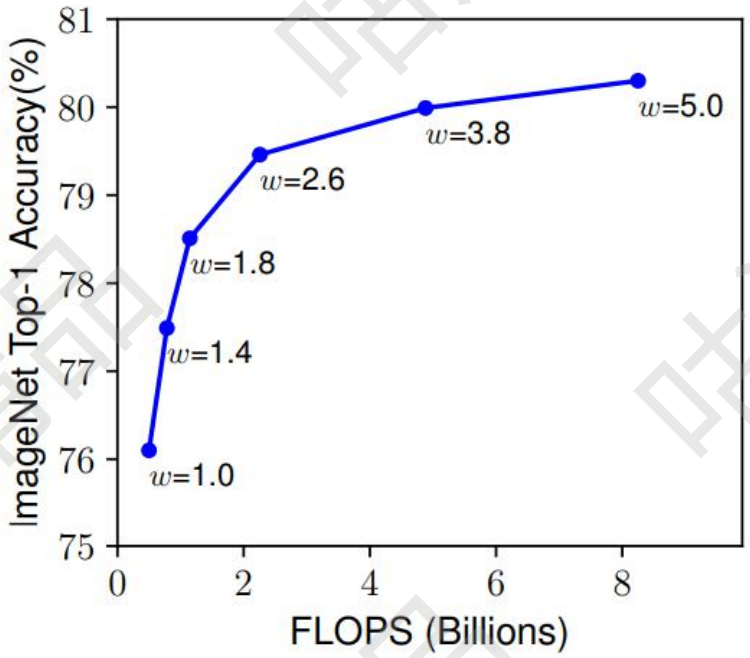
  - 网络的特征图个数，层数，输入分辨率都会对结果产生影响

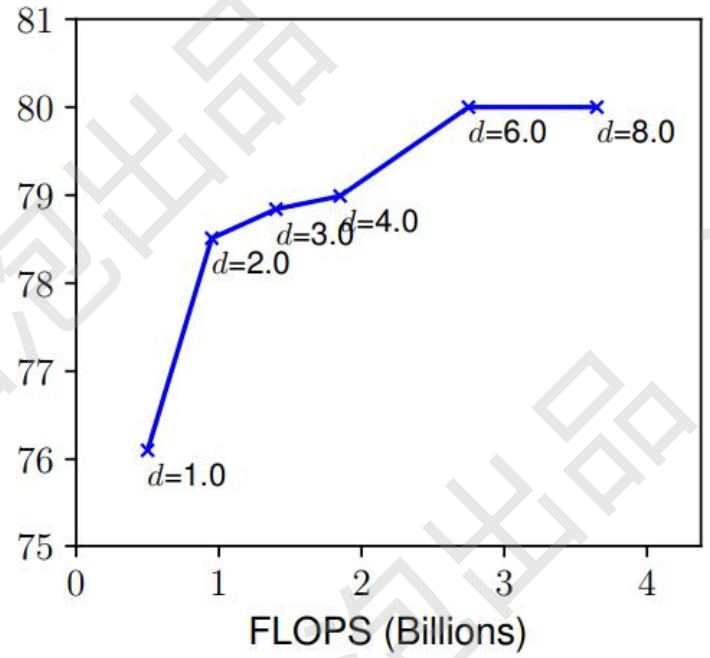


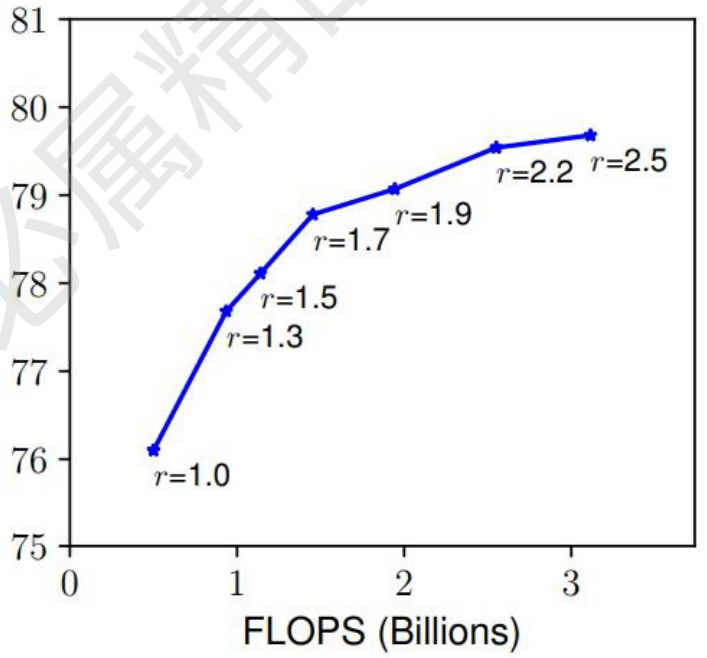
EfficientNet

- 出发点：

  - 单独提升这些指标，都能使得效果有所提升，但是会遇到瓶颈（FLOPS：例如卷积计算量=H*W*K*K*M*N,HW为输出长宽,K是卷积核大小 M为输入特征图的通道数，N为卷积核个数）



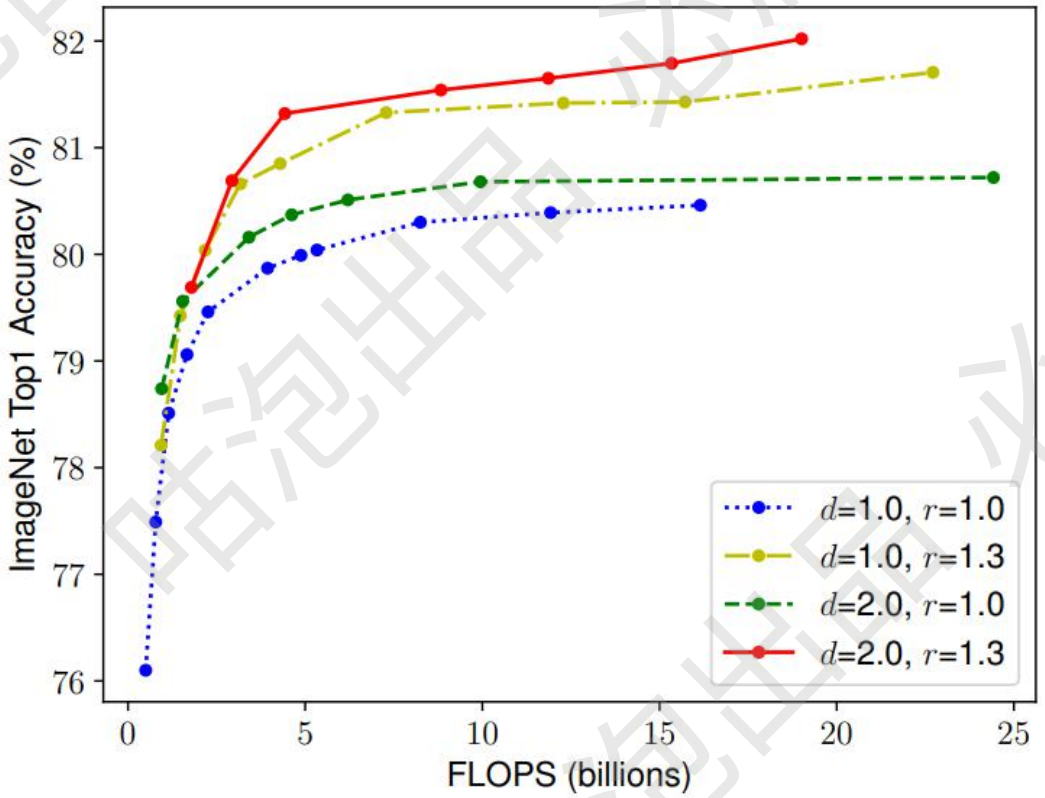




EfficientNet

- 出发点：

  - 综合提升这些指标，用参数搜索的方法（谷歌爸爸专属）来得出结果



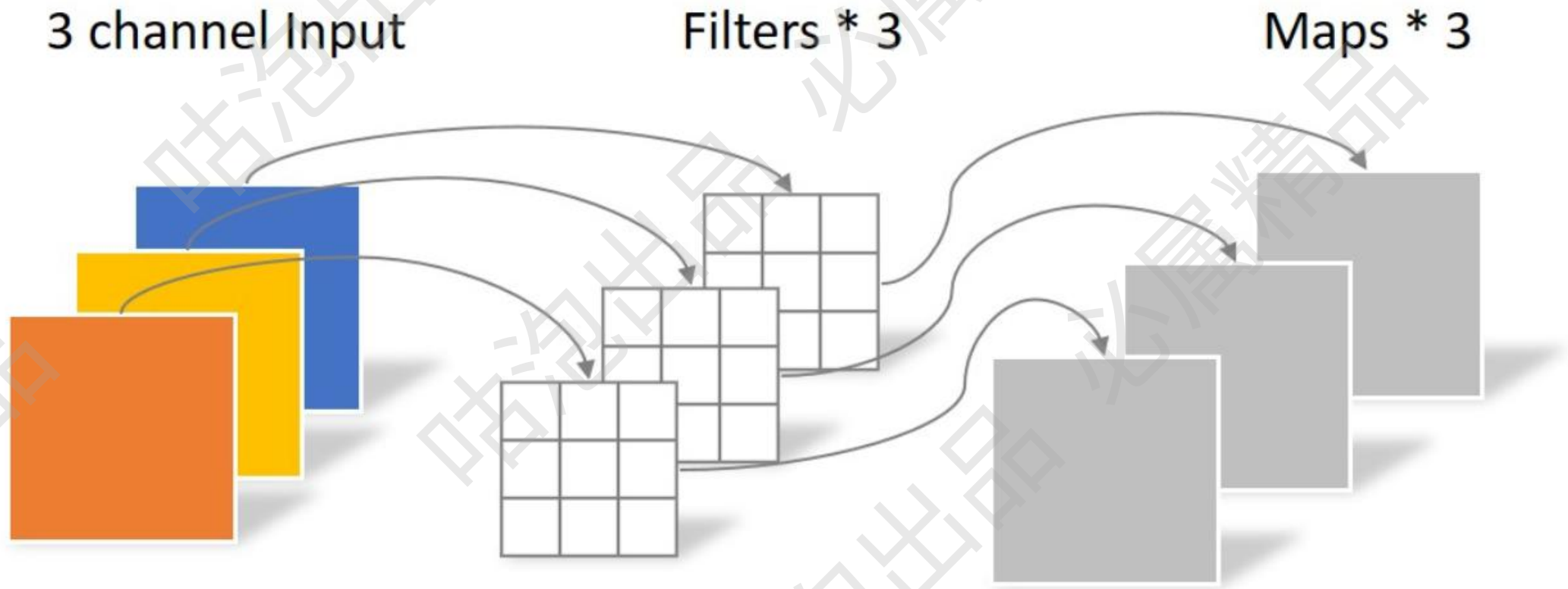
EfficientNet

- 基本网络架构：

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

EfficientNet

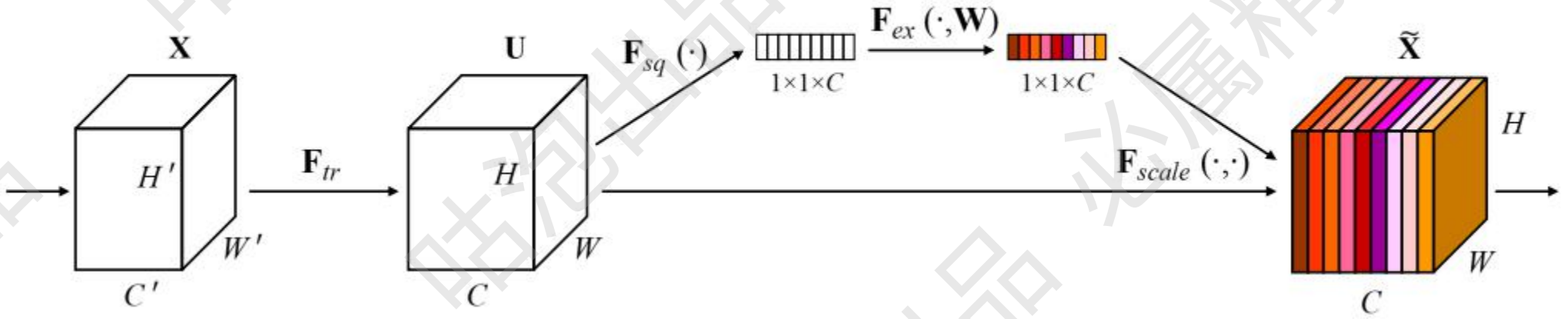
- Depthwise卷积



EfficientNet

- SE模块：

  - 对每个特征图计算其权重（注意力机制）



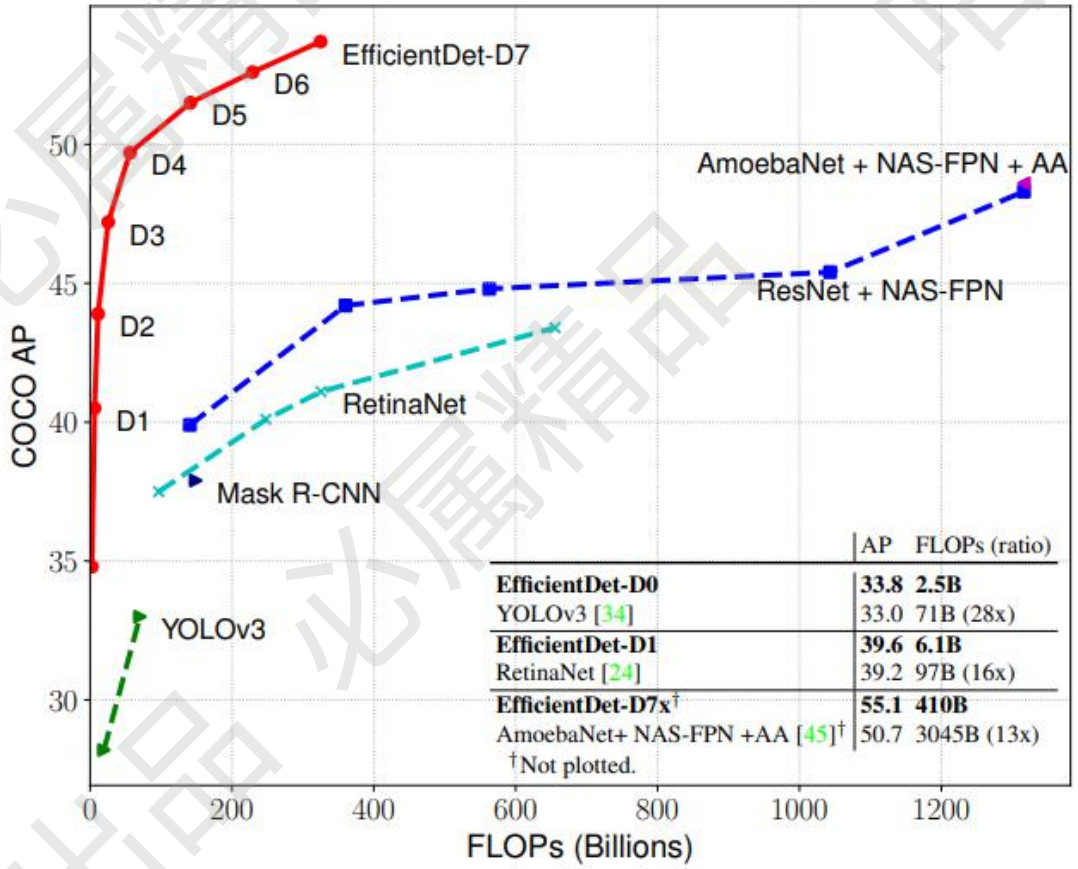
EfficientNet

- 计算流程：

```python
def efficientnet_params(model_name):
    """ Map EfficientNet model name to parameter coefficients. """
    params_dict = {
        # Coefficients:   width,depth,res,dropout
        'efficientnet-b0': (1.0, 1.0, 224, 0.2),
        'efficientnet-b1': (1.0, 1.1, 240, 0.2),
        'efficientnet-b2': (1.1, 1.2, 260, 0.3),
        'efficientnet-b3': (1.2, 1.4, 300, 0.3),
        'efficientnet-b4': (1.4, 1.8, 380, 0.4),
        'efficientnet-b5': (1.6, 2.2, 456, 0.4),
        'efficientnet-b6': (1.8, 2.6, 528, 0.5),
        'efficientnet-b7': (2.0, 3.1, 600, 0.5),
        'efficientnet-b8': (2.2, 3.6, 672, 0.5),
        'efficientnet-l2': (4.3, 5.3, 800, 0.5),
    }
    return params_dict[model_name]
```

EfficientDet

- 效果依旧有点碾压：
  - 对比了一些传统算法，效果还是可以的
  - 同样是多个版本，跟backbone类似
  - 相同重量级的优势比较明显
  - 整体=EfficientNet+BiFPN



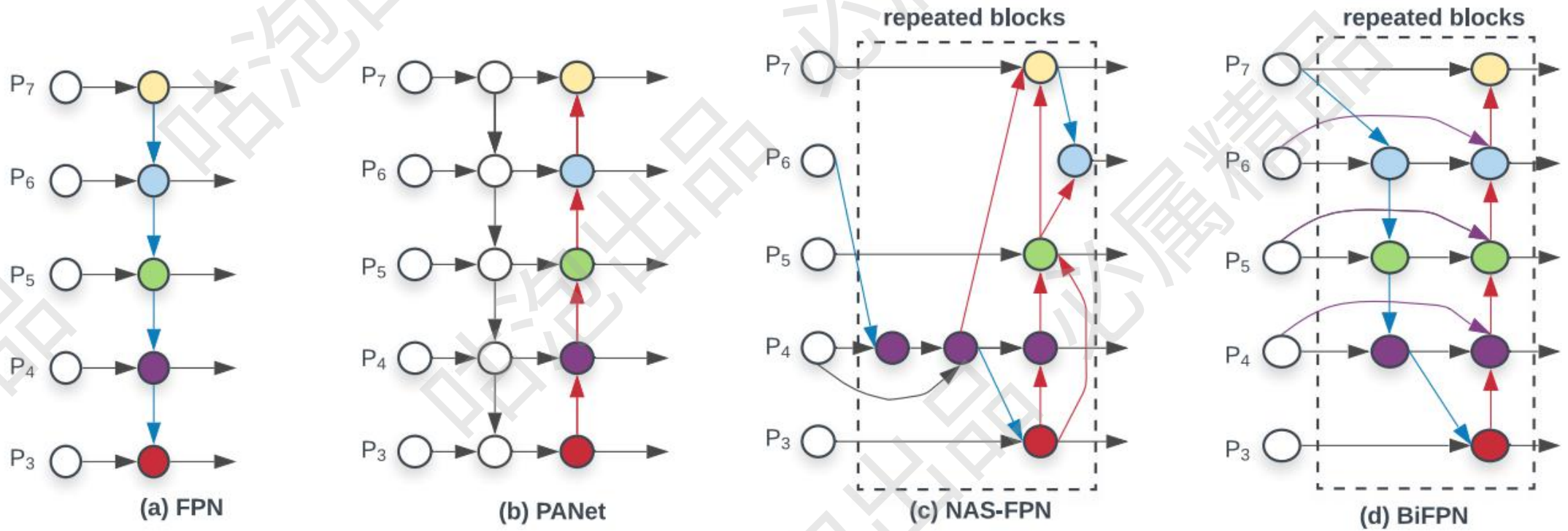
	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

EfficientDet

- FPN层：

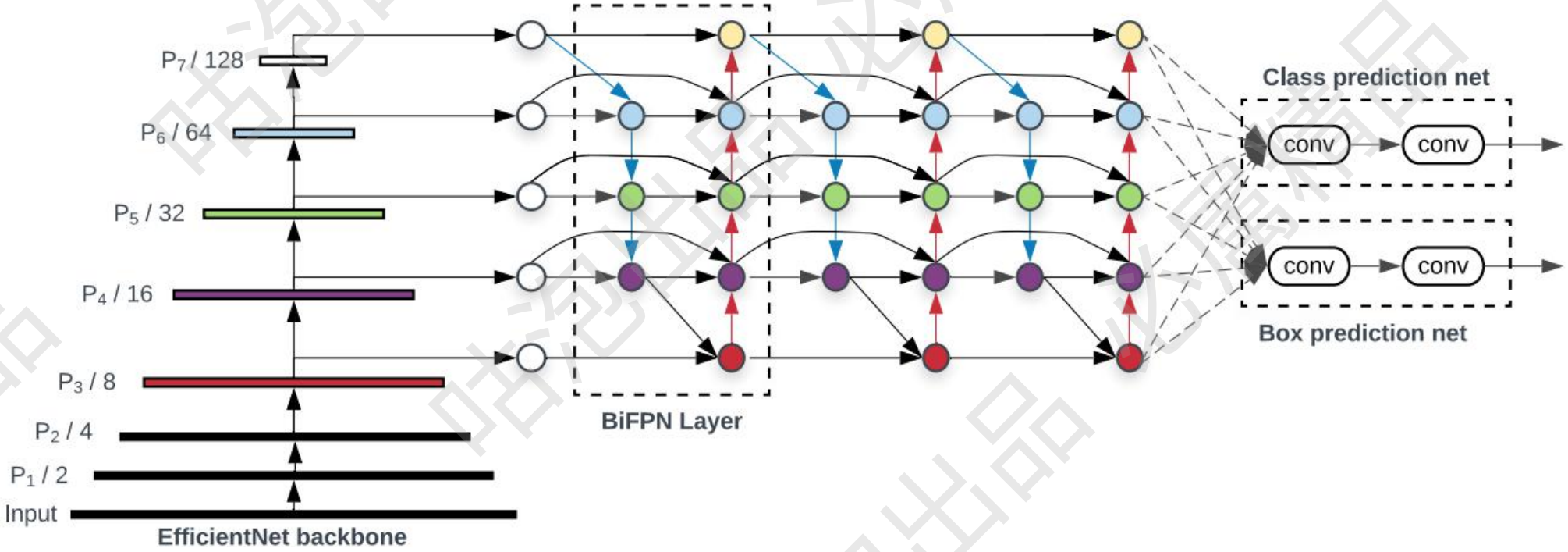
现在你说啥领域能不提及特征融合呢？检测方法也是一样的：



EfficientDet

- BiFPN：

  - 可以重复多次，基础结构就是虚线框，堆叠多少次可以选择



EfficientDet

- BiFPN：

  - 不同版本的BiFPN网络层数，需要根据精度和速度要求来选择

	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