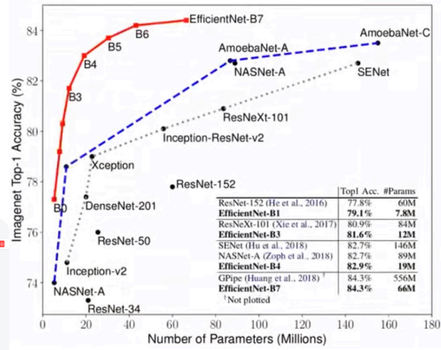
1. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
Google 2019 发表的文章
在论文中提到，本文提出的 EfficientNet-B7 在 Imagenet Top-1 上达到了当年最高准确率 84.3%，与之前准确率最高的 GPip 相比，参数数量仅为其 1/8.4，推理速度提升了 6.1倍。



同时探索输入分辨率，网络的深度、宽度的影响

https://blog.csdn.net/qq_37541097/article/details/114434046

2. · 根据以前的经验，增加网络深度 depth 能够得到更加丰富、复杂的特征并且能够良好的应用到其它任务中。但网络的深度过深会面临梯度消失，训练困难的问题
The intuition is that deeper ConvNet can capture richer and more complex features, and generalize well on new tasks. However, deeper networks are also more difficult to train due to vanishing gradient problem.

· 增加网络的 width 能够获得更高细粒度的特征并且也更容易训练，但对于 width 很大而深度较浅的网络往往很难学习到更深层次的特征
wider networks tend to be able to capture more fine-grained features and are easier to train. However, extremely wide but shallow networks tends to have difficulties in capturing higher level features

· 增加输入网络的图像分辨率能够潜在得获得更高细粒度的特征模版，但对于非常高的输入分辨率，准确率的收益也会越小。并且大分辨率图像会增加计算量
With higher resolution input images, ConvNet can potentially capture more fine-grained patterns. But the accuracy gain diminishes for very high resolutions.

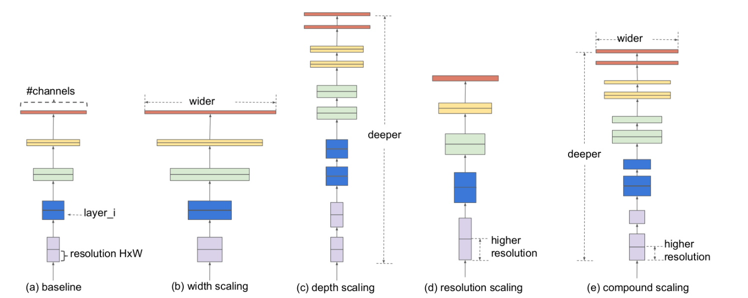
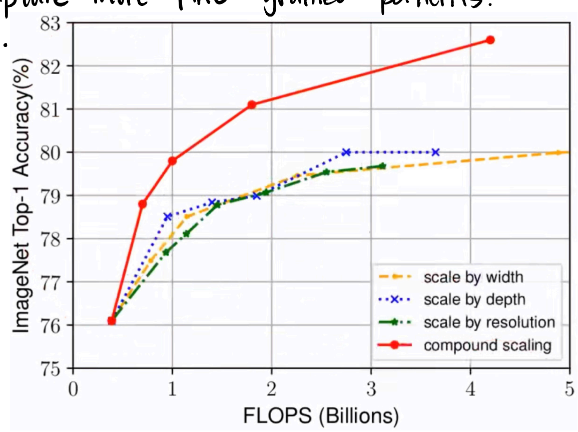


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.



单个维度增加，到 80% acc 就上不去了。一起增加，能突破 80%

3. EfficientNet - B0 baseline network

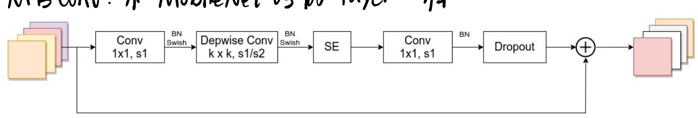
Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equat operator 重复了n次

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

n

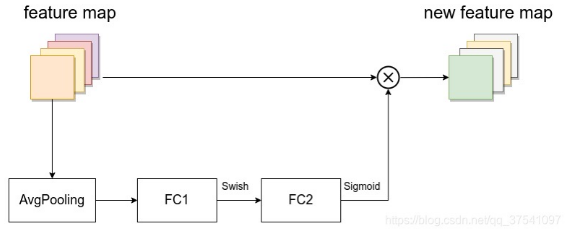
s 针对每个 stage 第1个 operator

MBConv：和 MobileNet v3 的 layer 一样



- 第1个升维的1x1卷积核，#filter = n * #output_c
- n=1时，不要第1个升维的1x1 filter，即 stage2 中的 MBConv 结构中都没有第1个升维的1x1卷积核（和 MobileNet v3 类似）
- shortcut：仅当输入 MBConv 结构的特征矩阵与输出的特征矩阵 shape 相同时才存在。
- 源码中只有使用到 shortcut 连接的 MB Conv 模块才有 Dropout 层

- SE模块:
  第1个全连接层的节点个数是 $\frac{1}{4}$ * # input_C, 且使用 swish AF.
  第2个 FC 的节点个数 == DW Conv # output_C, 且使用 sigmoid AF.



4. 其它版本参数

Model	input_size	width_coefficient	depth_coefficient	drop_connect_rate	dropout_rate
EfficientNetB0	224x224	1.0	1.0	0.2	0.2
EfficientNetB1	240x240	1.0	1.1	0.2	0.2
EfficientNetB2	260x260	1.1	1.2	0.2	0.3
EfficientNetB3	300x300	1.2	1.4	0.2	0.3
EfficientNetB4	380x380	1.4	1.8	0.2	0.4
EfficientNetB5	456x456	1.6	2.2	0.2	0.4
EfficientNetB6	528x528	1.8	2.6	0.2	0.5
EfficientNetB7	600x600	2.0	3.1	0.2	0.5

- `input_size` 代表训练网络时输入网络的图像大小
- `width_coefficient` 代表 `channel` 维度上的倍率因子，比如在 EfficientNetB0中 `Stage1` 的 `3x3` 卷积层所使用的卷积核个数是32，那么在B6中就是 $32 \times 1.8 = 57.6$ 接着取整到离它最近的8的整数倍即56，其它 `Stage` 同理。
- `depth_coefficient` 代表 `depth` 维度上的倍率因子（仅针对 `Stage2` 到 `Stage8`），比如在EfficientNetB0中 `Stage7` 的 $\widehat{L}_i = 4$，那么在B6中就是 $4 \times 2.6 = 10.4$ 接着向上取整即11.
- `drop_connect_rate` 是在 `MBConv` 结构中dropout层使用的 `drop_rate`，在官方keras模块的实现中 `MBConv` 结构的 `drop_rate` 是从0递增到 `drop_connect_rate` 的（具体实现可以看下官方源码，**注意，在源码实现中只有使用shortcut的时候才有Dropout层**）。
- `dropout_rate` 是最后一个全连接层前的 `dropout` 层（在 `stage9` 的Pooling与FC之间）的 `dropout_rate`。

5. Performance

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).