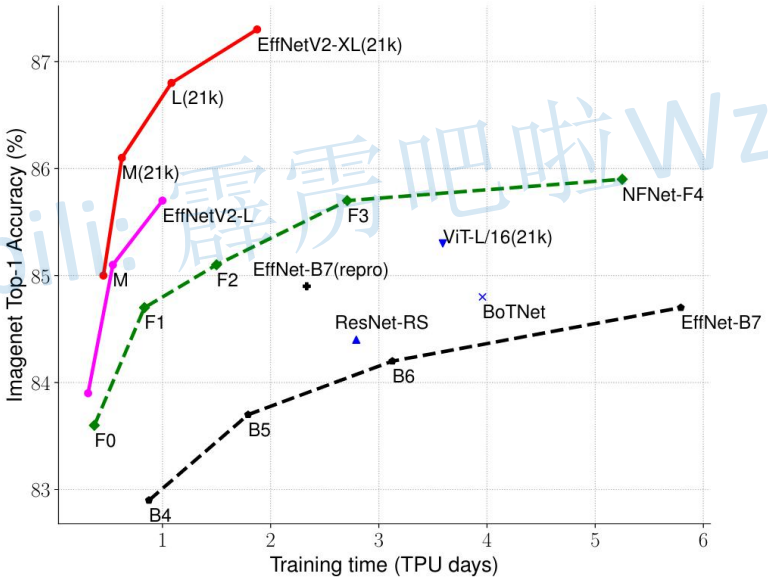
EfficientNetV2

EfficientNetV2: Smaller Models and Faster Training

Mingxing Tan [1] Quoc V. Le [1]

2021 CVPR



- 引入Fused-MBConv模块
- 引入渐进式学习策略(训练更快)

论文下载地址：https://arxiv.org/abs/2104.00298

推荐博文：https://blog.csdn.net/qq_37541097/article/details/116933569

EfficientNetV2

Table 7.EfficientNetV2 Performance Results on ImageNet

	Model	Top-1 Acc.	Params	FLOPs	Infer-time(ms)	Train-time (hours)
ConvNets & Hybrid	EfficientNet-B3 (Tan & Le, 2019a)	81.5%	12M	1.9B	19	10
	EfficientNet-B4 (Tan & Le, 2019a)	82.9%	19M	4.2B	30	21
	EfficientNet-B5 (Tan & Le, 2019a)	83.7%	30M	10B	60	43
	EfficientNet-B6 (Tan & Le, 2019a)	84.3%	43M	19B	97	75
	EfficientNet-B7 (Tan & Le, 2019a)	84.7%	66M	38B	170	139
	RegNetY-8GF (Radosavovic et al., 2020)	81.7%	39M	8B	21	-
	RegNetY-16GF (Radosavovic et al., 2020)	82.9%	84M	16B	32	-
	ResNeSt-101 (Zhang et al., 2020)	83.0%	48M	13B	31	-
	ResNeSt-200 (Zhang et al., 2020)	83.9%	70M	36B	76	-
	ResNeSt-269 (Zhang et al., 2020)	84.5%	111M	78B	160	-
	TResNet-L (Ridnik et al., 2020)	83.8%	56M	-	45	-
	TResNet-XL (Ridnik et al., 2020)	84.3%	78M	-	66	-
	EfficientNet-X (Li et al., 2021)	84.7%	73M	91B	-	-
	NFNet-F0 (Brock et al., 2021)	83.6%	72M	12B	30	8.9
	NFNet-F1 (Brock et al., 2021)	84.7%	133M	36B	70	20
	NFNet-F2 (Brock et al., 2021)	85.1%	194M	63B	124	36
	NFNet-F3 (Brock et al., 2021)	85.7%	255M	115B	203	65
	NFNet-F4 (Brock et al., 2021)	85.9%	316M	215B	309	126
	ResNet-RS (Bello et al., 2021)	84.4%	192M	-	-	61
	LambdaResNet-420-hybrid (Bello, 2021)	84.9%	125M	-	-	67
	BotNet-T7-hybrid (Srinivas et al., 2021)	84.7%	75M	46B	-	95
	BiT-M-R152x2 (21k) (Kolesnikov et al., 2020)	85.2%	236M	135B	500	-
Vision Transformers	ViT-B/32 (Dosovitskiy et al., 2021)	73.4%	88M	13B	13	-
	ViT-B/16 (Dosovitskiy et al., 2021)	74.9%	87M	56B	68	-
	DeiT-B (ViT+reg) (Touvron et al., 2021)	81.8%	86M	18B	19	-
	DeiT-B-384 (ViT+reg) (Touvron et al., 2021)	83.1%	86M	56B	68	-
	T2T-ViT-19 (Yuan et al., 2021)	81.4%	39M	8.4B	-	-
	T2T-ViT-24 (Yuan et al., 2021)	82.2%	64M	13B	-	-
	ViT-B/16 (21k) (Dosovitskiy et al., 2021)	84.6%	87M	56B	68	
	ViT-L/16 (21k) (Dosovitskiy et al., 2021)	85.3%	304M	192B	195	172
ConvNets (ours)	**EfficientNetV2-S**	83.9%	24M	8.8B	24	7.1
	EfficientNetV2-M	85.1%	55M	24B	57	13
	EfficientNetV2-L	85.7%	121M	53B	98	24
	EfficientNetV2-S (21k)	84.9%	24M	8.8B	24	9.0
	EfficientNetV2-M (21k)	86.2%	55M	24B	57	15
	EfficientNetV2-L (21k)	86.8%	121M	53B	98	26
	EfficientNetV2-XL (21k)	87.3%	208M	94B	-	45

Top-1达到87.3%

EfficientNetV2

在EfficientNetV1中作者关注的是准确率，参数数量以及FLOPs（理论计算量小不代表推理速度快），在EfficientNetV2中作者进一步关注模型的训练速度。

Table 10. Comparison with the same training settings – Our new EfficientNetV2-M runs faster with less parameters.

	Acc. (%)	Params (M)	FLOPs (B)	TrainTime (h)	InferTime (ms)
V1-B7	85.0	66	38	54	170
V2-M (ours)	85.1	55 (-17%)	24 (-37%)	13 (-76%)	57 (-66%)

Table 11. Scaling down model size – We measure the inference throughput on V100 FP16 GPU with batch size 128.

	Top-1 Acc.	Parameters	Throughput (imgs/sec)
V1-B1	79.0%	7.8M	2675
V2-7M	78.7%	7.4M	(2.1x) 5739
V1-B2	79.8%	9.1M	2003
V2-8M	79.8%	8.1M	(2.0x) 3983
V1-B4	82.9%	19M	628
V2-14M	82.1%	14M	(2.7x) 1693
V1-B5	83.7%	30M	291
V2-S	83.6%	24M	(3.1x) 901

EfficientNetV2

EfficientNetV1中存在的问题

- 训练图像的尺寸很大时，训练速度非常慢
- 在网络浅层中使用Depthwise convolutions速度会很慢
- 同等的放大每个stage是次优的

EfficientNetV2

训练图像的尺寸很大时，训练速度非常慢。针对这个问题一个比较好想到的办法就是降低训练图像的尺寸，之前也有一些文章这么干过。降低训练图像的尺寸不仅能够加快训练速度，还能使用更大的batch_size.

Table 2. EfficientNet-B6 accuracy and training throughput for different batch sizes and image size.

		TPUv3 imgs/sec/core		V100 imgs/sec/gpu	
	Top-1 Acc.	batch=32	batch=128	batch=12	batch=24
train size=512	84.3%	42	OOM	29	OOM
train size=380	84.6%	76	93	37	52

out of memory

EfficientNetV2

在网络浅层中使用Depthwise convolutions速度会很慢。无法充分利用现有的一些加速器（虽然理论上计算量很小，但实际使用起来并没有想象中那么快）。故引入Fused-MBConv结构。

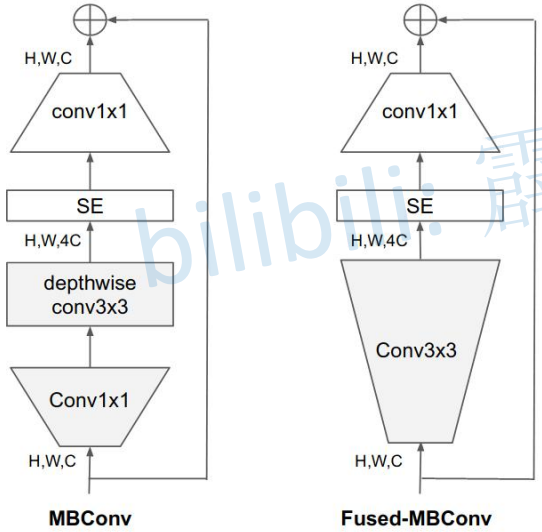


Figure 2. Structure of MBConv and Fused-MBConv.

Table 3. Replacing MBConv with Fused-MBConv. `No fused` denotes all stages use MBConv, `Fused stage1-3` denotes replacing MBConv with Fused-MBConv in stage {2, 3, 4}.

	Params (M)	FLOPs (B)	Top-1 Acc.	TPU imgs/sec/core	V100 imgs/sec/gpu
No fused	19.3	4.5	82.8%	262	155
Fused stage1-3	20.0	7.5	83.1%	362	216
Fused stage1-5	43.4	21.3	83.1%	327	223
Fused stage1-7	132.0	34.4	81.7%	254	206

EfficientNetV2

同等的放大每个stage是次优的。 在EfficientNetV1中，每个stage的深度和宽度都是同等放大的。但每个stage对网络的训练速度以及参数数量的贡献并不相同，所以直接使用同等缩放的策略并不合理。在这篇文章中，作者采用了非均匀的缩放策略来缩放模型。

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

EfficientNetV2

EfficientNetV2中做出的贡献

在之前的一些研究中，主要关注的是准确率以及参数数量(注意，参数数量少并不代表推理速度更快)。但在近些年的研究中，开始关注网络的训练速度以及推理速度(可能是准确率刷不动了)。

- 引入新的网络(EfficientNetV2)，该网络在训练速度以及参数数量上都优于先前的一些网络。
- 提出了改进的渐进学习方法，该方法会根据训练图像的尺寸动态调节正则方法(提升训练速度、准确率)
- 通过实验与先前的一些网络相比，训练速度提升11倍，参数数量减少为1/6.8

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

EfficientNetV2

EfficientNetV2网络框架

注意，在源码中Stage6的输出Channels是等于256并不是表格中的272，Stage7的输出Channels是1280并不是表格中的1792

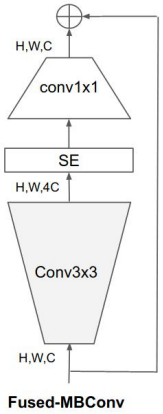
Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

与EfficientNetV1的不同点：

- 除了使用MBConv模块，还使用Fused-MBConv模块
- 会使用较小的expansion ratio
- 偏向使用更小的kernel_size(3x3)
- 移除了EfficientNetV1中最后一个步距为1的stage（V1中的stage8）

EfficientNetV2

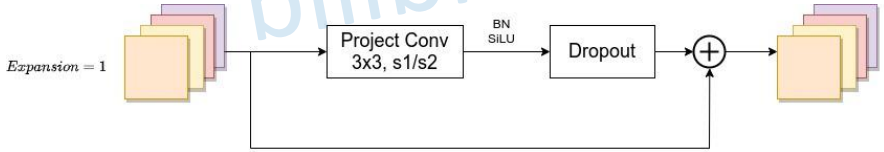


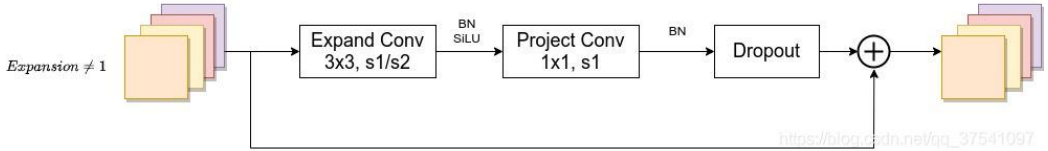
Fused-MBConv

Fused-MBConv模块

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1





注意shortcut连接
注意Dropout层

EfficientNetV2

Deep Networks with Stochastic Depth
https://arxiv.org/pdf/1603.09382.pdf

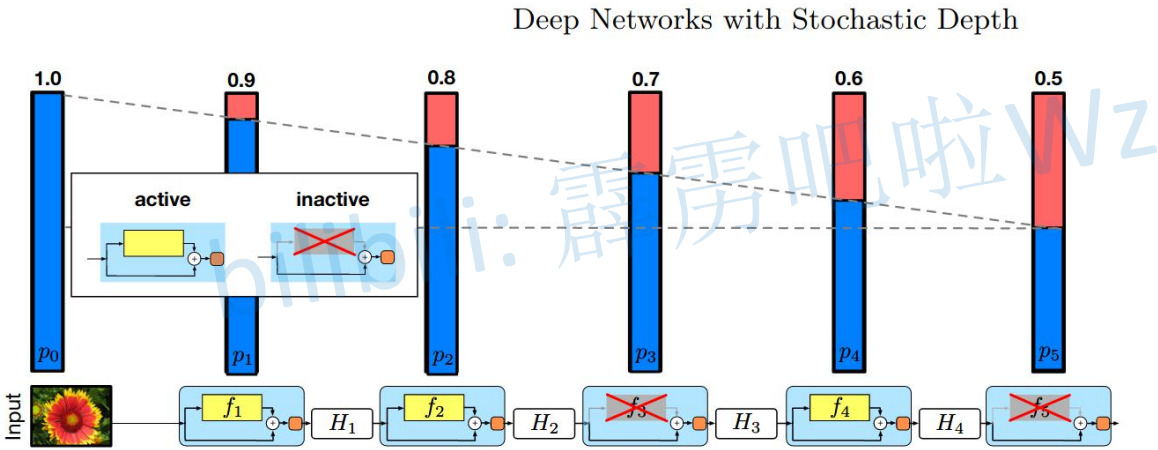


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

提升训练速度
小幅提升准确率

在EfficientNetV2中
drop_prob: 从0-0.2

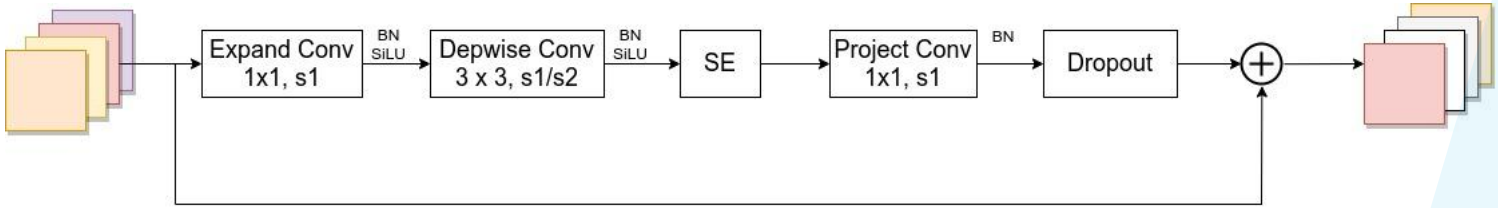
EfficientNetV2

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

MBConv模块

注意shortcut连接
注意Dropout层



EfficientNetV2

effnetv2_configs.py

```
######## EfficientNet V2 configs ########
# The baseline config for v2 models.
v2_base_block = [
    'r1_k3_s1_e1_i32_o16_c1',
    'r2_k3_s2_e4_i16_o32_c1',
    'r2_k3_s2_e4_i32_o48_c1',
    'r3_k3_s2_e4_i48_o96_se0.25',
    'r5_k3_s1_e6_i96_o112_se0.25',
    'r8_k3_s2_e6_i112_o192_se0.25',
]
```

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

- r代表当前Stage中Operator重复堆叠的次数
- k代表kernel_size
- s代表步距stride
- e代表expansion ratio
- i代表input channels
- o代表output channels
- c代表conv_type，1代表Fused-MBConv，0代表MBConv（默认为MBConv）
- se代表使用SE模块，以及se_ratio

EfficientNetV2

EfficientNetV2-S

```
# about base * (width1.4, depth1.8)
v2_s_block = [
    'r2_k3_s1_e1_i24_o24_c1',
    'r4_k3_s2_e4_i24_o48_c1',
    'r4_k3_s2_e4_i48_o64_c1',
    'r6_k3_s2_e4_i64_o128_se0.25',
    'r9_k3_s1_e6_i128_o160_se0.25',
    'r15_k3_s2_e6_i160_o256_se0.25',
]
```

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

- r代表当前Stage中Operator重复堆叠的次数
- k代表kernel_size
- s代表步距stride
- e代表expansion ratio
- i代表input channels
- o代表output channels
- c代表conv_type，1代表Fused-MBConv，0代表MBConv（默认为MBConv）
- se代表使用SE模块，以及se_ratio

EfficientNetV2

EfficientNetV2-M

```
# about base * (width1.6, depth2.2)
 v2_m_block = [
   'r3_k3_s1_e1_i24_o24_c1',
   'r5_k3_s2_e4_i24_o48_c1',
   'r5_k3_s2_e4_i48_o80_c1',
   'r7_k3_s2_e4_i80_o160_se0.25',
   'r14_k3_s1_e6_i160_o176_se0.25',
   'r18_k3_s2_e6_i176_o304_se0.25',
   'r5_k3_s1_e6_i304_o512_se0.25',
]
```

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

- r代表当前Stage中Operator重复堆叠的次数
- k代表kernel_size
- s代表步距stride
- e代表expansion ratio
- i代表input channels
- o代表output channels
- c代表conv_type，1代表Fused-MBConv，0代表MBConv（默认为MBConv）
- se代表使用SE模块，以及se_ratio

EfficientNetV2

EfficientNetV2-L

```
# about base * (width2.0, depth3.1)
v2_l_block = [
    'r4_k3_s1_e1_i32_o32_c1',
    'r7_k3_s2_e4_i32_o064_c1',
    'r7_k3_s2_e4_i64_o096_c1',
    'r10_k3_s2_e4_i96_o192_se0.25',
    'r19_k3_s1_e6_i192_o224_se0.25',
    'r25_k3_s2_e6_i224_o384_se0.25',
    'r7_k3_s1_e6_i384_o640_se0.25',
]
```

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

- r代表当前Stage中Operator重复堆叠的次数
- k代表kernel_size
- s代表步距stride
- e代表expansion ratio
- i代表input channels
- o代表output channels
- c代表conv_type，1代表Fused-MBConv，0代表MBConv（默认为MBConv）
- se代表使用SE模块，以及se_ratio

EfficientNetV2

EfficientNetV2其他训练参数

```
efficientnetv2_params = {

    # (block, width, depth, train_size, eval_size, dropout, randaug, mixup, aug)
    'efficientnetv2-s':  # 83.9% @ 22M
        (v2_s_block, 1.0, 1.0, 300, 384, 0.2, 10, 0, 'randaug'),

    'efficientnetv2-m':  # 85.2% @ 54M
        (v2_m_block, 1.0, 1.0, 384, 480, 0.3, 15, 0.2, 'randaug'),

    'efficientnetv2-l':  # 85.7% @ 120M
        (v2_l_block, 1.0, 1.0, 384, 480, 0.4, 20, 0.5, 'randaug'),
}
```

EfficientNetV2

Progressive Learning渐进学习策略

Table 5. ImageNet top-1 accuracy. We use RandAug (Cubuk et al., 2020), and report mean and stdev for 3 runs.

	Size=128	Size=192	Size=300
RandAug magnitude=5	**78.3** ±0.16	81.2 ±0.06	82.5 ±0.05
RandAug magnitude=10	78.0 ±0.08	**81.6** ±0.08	82.7 ±0.08
RandAug magnitude=15	77.7 ±0.15	81.5 ±0.05	**83.2** ±0.09

EfficientNetV2

训练早期使用较小的训练尺寸以及较弱的正则方法weak regularization，这样网络能够快速的学习到一些简单的表达能力。接着逐渐提升图像尺寸，同时增强正则方法adding stronger regularization。这里所说的regularization包括**Dropout**，**RandAugment**以及**Mixup**。

Figure 4. Training process in our improved progressive learning – It starts with small image size and weak regularization (epoch=1), and then gradually increase the learning difficulty with larger image sizes and stronger regularization: larger dropout rate, RandAugment magnitude, and mixup ratio (e.g., epoch=300).

EfficientNetV2

Algorithm 1 Progressive learning with adaptive regularization.

Input: Initial image size S_0 and regularization $\{\phi_0^k\}$.
Input: Final image size S_e and regularization $\{\phi_e^k\}$.
Input: Number of total training steps N and stages M.
for $i = 0$ **to** $M - 1$ **do**
  Image size: $S_i \leftarrow S_0 + (S_e - S_0) \cdot \frac{i}{M-1}$
  Regularization: $R_i \leftarrow \{\phi_i^k = \phi_0^k + (\phi_e^k - \phi_0^k) \cdot \frac{i}{M-1}\}$
  Train the model for $\frac{N}{M}$ steps with S_i and R_i.
end for

EfficientNetV2

Table 6. Progressive training settings for EfficientNetV2.

	S		M		L	
	min	max	min	max	min	max
Image Size	128	300	128	380	128	380
RandAugment	5	15	5	20	5	25
Mixup alpha	0	0	0	0.2	0	0.4
Dropout rate	0.1	0.3	0.1	0.4	0.1	0.5

EfficientNetV2

Table 12. Progressive learning for ResNets and EfficientNets – (224) and (380) denote the targeted inference image size. Our progressive training improves both the accuracy and training time for all different networks.

	Baseline		Progressive	
	Acc.(%)	TrainTime	Acc.(%)	TrainTime
ResNet50 (224)	78.1	4.9h	78.4	3.5h (-29%)
ResNet50 (380)	80.0	14.3h	80.3	5.8h (-59%)
ResNet152 (380)	82.4	15.5h	82.9	7.2h (-54%)
EfficientNet-B4	82.9	20.8h	83.1	9.4h (-55%)
EfficientNet-B5	83.7	42.9h	84.0	15.2h (-65%)

沟通方式

1.github

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

2.bilibili

https://space.bilibili.com/18161609/channel/index

3.CSDN

https://blog.csdn.net/qq_37541097/article/details/103482003

感谢各位的观看！