EfficientNetV2 : Smaller Models and Faster Training

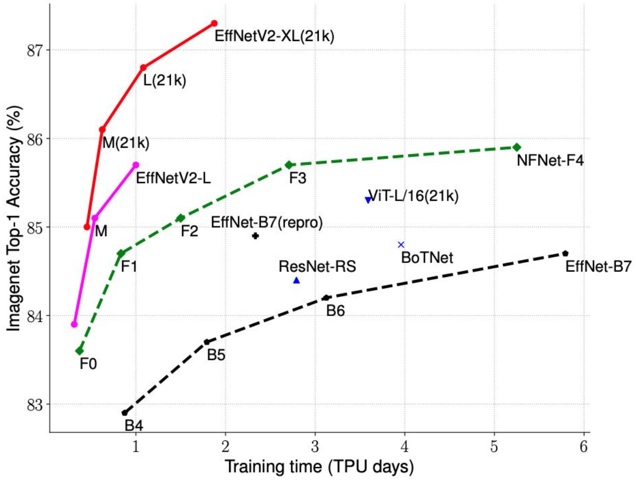
정찬미

Abstract

- image size를 점진적으로 증가시켜 속도를 높임
    - 정확도가 떨어짐
- progressive learning을 사용하여 정확도가 하락하는 것을 방지함
    - 이미지 사이즈와 정규화를 조정함
- 이전 모델보다 training speed가 빨라졌고, parameter efficiency가 더 좋아짐

Introduction

- 모델의 size와 학습데이터의 size가 커짐에 따라 training efficiency이 중요해짐
- 기존의 EfficientNet 관찰 결과
  - Training with very large image size is slow
  - Depthwise convolutions are slow in early layers
  - Equally scaling up every stage is sub-optimal
- 관찰 결과를 기반으로 추가적인 operation을 사용하여 search space를 높이고, training-aware NAS 와 scaling을 사용하여 model accuracy, training speed, parameter size를 최적화함



(a) Training efficiency.

	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 Architecture Design

- Training with very large image size is slow
  - large image size를 사용하면 상당한 양의 메모리를 차지함
  - 메모리 크기가 고정되어 있기 때문에 small batch size를 사용
    - 속도가 느려짐
  - small image size를 사용하는 FixRes를 적용하여 간단하게 개선함

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

EfficientNetV2 Architecture Design

- Depthwise convolutions are slow in early layers
  - Depthwise convolutions는 modern accelerator를 활용하지 못하기 때문에 training speed를 느리게 함
  - MBConv 대신에 Fused-MBConv 사용
    - 모든 stage에 적용하면 training speed가 느려져서 초기의 stage에만 적용함

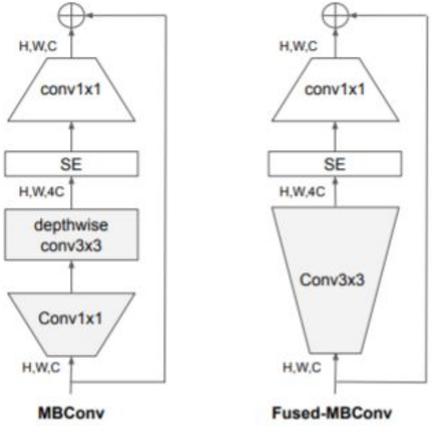


Figure 2. Structure of MBConv and Fused-MBConv.

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 Architecture Design

- Equally scaling up every stage is sub-optimal
    - 기존모델은 compound scaling을 통해 모든 stage에 대해서 균일하게 모델을 scaling함
        - 모든 stage가 동일하게 training speed와 parameter efficiency에 기여하는 것은 아님
    - EfficientNetV2에서는 non-uniform scaling strategy를 사용하여 later stage에 점진적으로 layer를 추가함
    - image size에 따라 training speed가 감소하는 것을 최소화하기 위해 maximum image size를 제한함

Training-Aware NAS and Scaling

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 equation 2.

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

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

Training-Aware NAS and Scaling

- 기존 EfficientNet과의 차이점
  - 초기 layer에서 Fused-MBConv와 MBConv를 모두 사용함
  - memory access overhead를 줄이기 위해 MBConv에 대해 작은 expansion ratio 사용함
  - 3 x 3 의 작은 커널을 사용함
    - 커널의 사이즈가 작아서 receptive field가 감소함
    - 이를 보완하기 위해 더 많은 layer가 사용됨
  - large parameter size와 memory access overhead를 줄이기 위해 기존의 EfficientNet에서 마지막 stride-1 stage를 제거함

Training-Aware NAS and Scaling

- training speed 비교

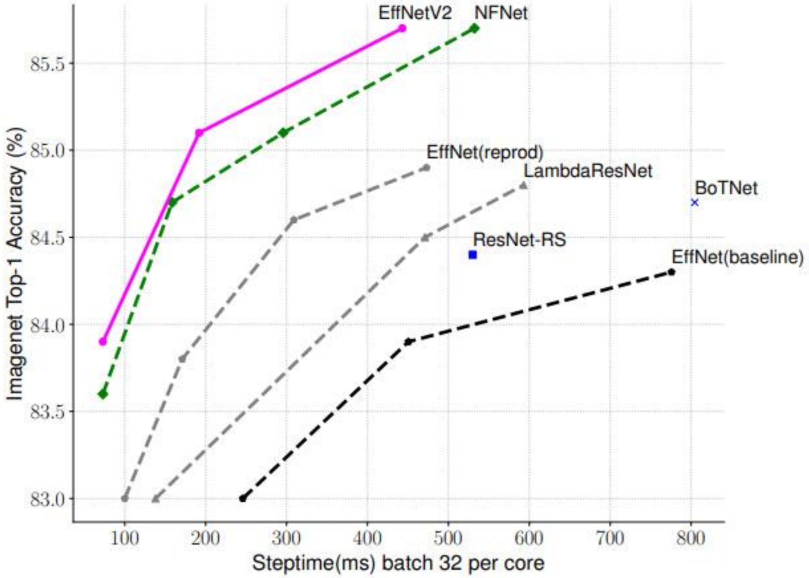


Figure 3. ImageNet accuracy and training step time on TPUv3 – Lower step time is better; all models are trained with fixed image size without progressive learning.

Progressive Learning

- Motivation
  - image size에 대한 불균형한 정규화 때문에 정확도의 저하가 발생했다고 가정
    - 학습에 사용되는 image size에 따라 정규화 강도를 조정
  - 모델에 size가 다른 image와 data augmentation으로 학습시킴

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**

Results

- Comparison EfficientNet

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%)

Results

- Progressive Learning for Different Networks

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%)

Results

- Adaptive Regularization

Table 13. Adaptive regularization – We compare ImageNet top-1 accuracy based on the average of three runs.

	Vanilla	+our adaptive reg
Progressive resize (Howard, 2018)	84.3±0.14	85.1±0.07 (+0.8)
Random resize (Hoffer et al., 2019)	83.5±0.11	84.2±0.10 (+0.7)

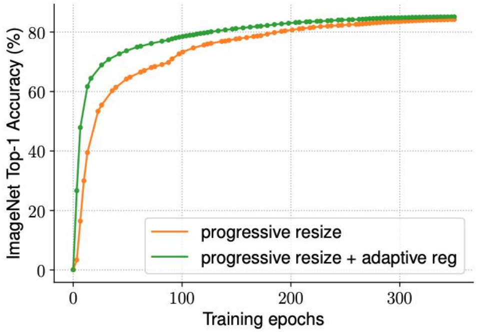


Figure 6. **Training curve comparison** – Our adaptive regularization converges faster and achieves better final accuracy.