EfficientNetV2: Smaller Models and Faster Training

2021-07-25 | JiHyun Lee

Content

개요

1. 딥러닝 모델과 데이터의 사이즈가 점점 커짐에 따라 **학습 효율(training efficiency)**의 중요성 또한 커짐.

2. NFNet (batch norm 제거), RestNet-RS (scaling hyperparameter 최적화), Vision Transformer (Transformer block 사용) 등은 제각기 학습 효율을 개선했으나, **파라미터 효율(모델 크기)이 나빠지는 한계**가 있었음.

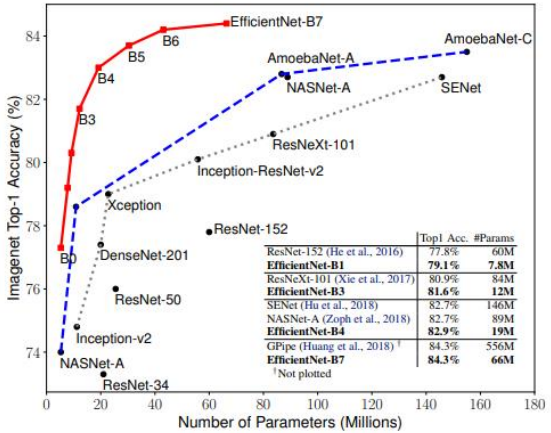
3. 본 연구는 이전 연구(EfficientNetV1), 컨볼루션 커널(Fused-MBConv), 뉴럴넷 설계 방법론(Training-aware NAS), 학습 방식(Progressive Learning) 등 다양한 테크닉을 이용하여 **높은 training efficiency 및 parameter efficiency와 accuracy를 함께 달성.**

EfficientNetV1

1. 사용자의 컴퓨팅 자원에 따라 선택할 수 있는 B0~B7 모델

2. 모델의 **width (채널 수), depth (레이저 개수), resolution (이미지 사이즈)** 을 동시에 조절할 수 있는 **compound scaling method** 적용. B0 기준으로 B7까지 scale up

3. B7의 경우, 당시 SoTA 대비 8.4x smaller & 6.1x faster한 모델로서 ImageNet 84.3% top-1 accuracy 달성



	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

Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

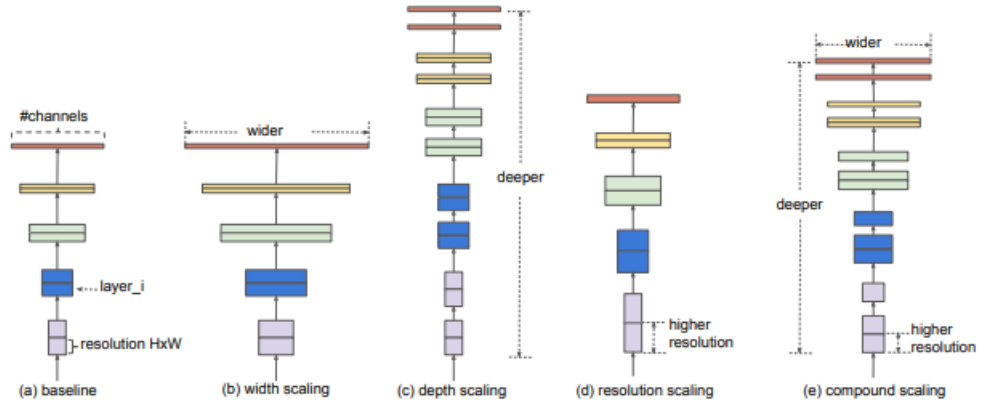


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

모델의 width (채널 수), depth (레이저 개수), resolution (이미지 사이즈) 을 동시에 조절할 수 있는 compound scaling method 적용. B0 기준으로 B7까지 scale up

- IDEA

모델의 depth, width, resolution 중 단 하나만을 조절한 시도들은 많았으나, 이 3가지 facto는 서로 독립적이지 않음.
따라서, 3가지 요소를 동시에 조절할 수 있는 compound scaling 적용.
(e.g. 이미지 사이즈가 커질수록 더 많은 픽셀을 처리하기 위해 레이어 및 채널 개수도 올려야 함)

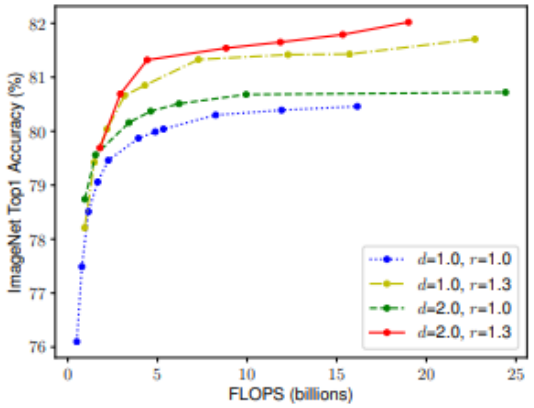


Figure 4. **Scaling Network Width for Different Baseline Networks**. Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline (d=2.0, r=1.3) has 36 layers with resolution 299x299.

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 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

EfficientNet-B0 baseline network

- **Mobile NasNet의 방법**을 사용하여 탐색한 아키텍처
- Search 방법의 차이 존재.
- 기존 Mobile NasNet 은 속도에 대한 기준으로 실제 디바이스에서의 latency를 측정한 반면,
- EfficientNet의 경우 범용 모델을 목표로 설계한 것이기 때문에 latency 대신 FLOP 사용.

MNasNet: Platform-Aware Neural Architecture Search for Mobile의 **NAS 방법론을** 이용하여 탐색

- 여러가지 방식이 있는데, 여기서는 **강화학습**을 이용하여 **Accuracy (정확도)와 FLOP(속도) 을 NAS의 objective로 설정**
- **이에 대해 보상을 주는 방식 (강화학습) 으로 뉴럴넷 아키텍처를 탐색하여 B0 얻어냄.**

In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

STEP 1
Φ=1로 고정한 뒤, grid search를 이용하여 alpha, beta, gamma 탐색
* Alpha, beta, gamma : depth, width, resolution 을 결정하는 상수

STEP 2
탐색된 alpha, beta, gamma를 고정한 채로, Φ를 2로 증가시킴으로써 B1 획득

Φ를 증가시킴으로써 B2, B3, … , B7 획득
Aplha, beta, gamma 를 다시 구하는 것도 가능한데, 그렇다면 search 비용이 너무 많이 들어서, 저자는 alpha, beta, gamma 고정한 채로 Φ 만 증가시켜 B2, .. , B7 까지 아키텍처 탐색하여 얻어냄.

얻어낸 B0를 기반으로 모델 scale up

Compound scaling method

- Compound coefficient Φ 가 depth, width, resolution 동시에 scale up

EfficientNetV2

EfficientNetV1 문제점

1. **해상도가 높은 이미지로 학습하면 학습 속도가 느림**
- GPU 메모리가 동일함으로, 이미지 사이즈가 클 경우 mini-batch size 감소
- 이에 따라 학습 속도가 크게 느려짐

2. **Depthwise Separable Convolution은 초기 layer 에서 느림**
- Depthwise Separable Conv는 이론적으로 일반 Conv 보다 8~9배 빠르지만, modern accelerator는 그 계산 방식에 최적화 되어 있지 않음

3. **모든 stage를 동일하게 scaling 하는 것은 sub-optimal 함**
- EfficientNetV1은 간단한 compound scaling rule을 이용하여 모든 stage를 동일하게 scale up 하므로, 학습 속도 및 파라미터 효율을 최적화한다고 볼 수 없음

EfficientNetV1 문제점 - 해결방안

1. **해상도가 높은 이미지로 학습하면 학습 속도가 느림**
- **Progressive Learning을 이용**하여 학습 중에 image size 및 regularization 을 동적으로 변경

2. **Depthwise Separable Convolution은 초기 layer 에서 느림**
- 초기 layer 들을 MBConv 대신, **Fused-MBConv 로 대체**

3. **모든 stage를 동일하게 scaling 하는 것은 sub-optimal 함**
- **Non-uniform scaling 전략 적용 (heuristics)**

Progressive Learning
- 학습 중에 학습 설정을 동적으로 바꾸는 것
- Progressive resizing : 기존 여러 연구들이 학습 중 이미지 사이즈를 변경하는 시도를 하였으나, 정확도 감소하였음.

제안 가설
정확도 감소의 원인은 불균형한 regularization
- **즉, 사이즈가 작은 이미지는 상대적으로 약한 regularization 필요**
- **사이즈가 큰 이미지는 상대적으로 강한 regularization 필요**

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

이를 입증하기 위해, RandAugment 라는 regularization 방법으로 실험 진행.
- 이미지가 작은 경우, RandAug 가 작은 편이 정확도가 높았고, 이미지가 큰 경우 RandAug 가 높은 편이 정확도 높음.

Adaptive 한 regularization 적용

- **ViT 는 embedding vector 가 입력 이미지의 사이즈에 의존적인 반면, ConvNet은 이미지 사이즈에 독립적이므로, 서로 다른 사이즈 이미지에서 학습한 weight 를 그대로 사용 가능!**

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

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).

학습 중에 변경할 이미지 사이즈를 최소 크기부터 최대 크기까지 미리 설정한 후,
Stage 1 은 가장 작은 이미지와 그에 따른 regularization 적용 (이미지 상당히 온전)
..
Stage 4 는 가장 큰 이미지와 그에 따른 regularization 적용 (이미지 상당히 변형)

Regularization : RandAugment, mixup, dropout 3가지 사용

2. Depthwise Separable Convolution은 초기 layer 에서 느림

- Depthwise Separable Conv는 이론적으로 일반 Conv 보다 8~9배 빠르지만, modern accelerator는 그 계산 방식에 최적화 되어 있지 않음

Depthwide Separable Conv

- MobileNet 에서 처음 소개.
- 일반 conv 대비 연산량을 8배에서 9배 줄일 수 있는 연산.
- Mobile, edge device 에서 주로 사용되고 있는 연산.

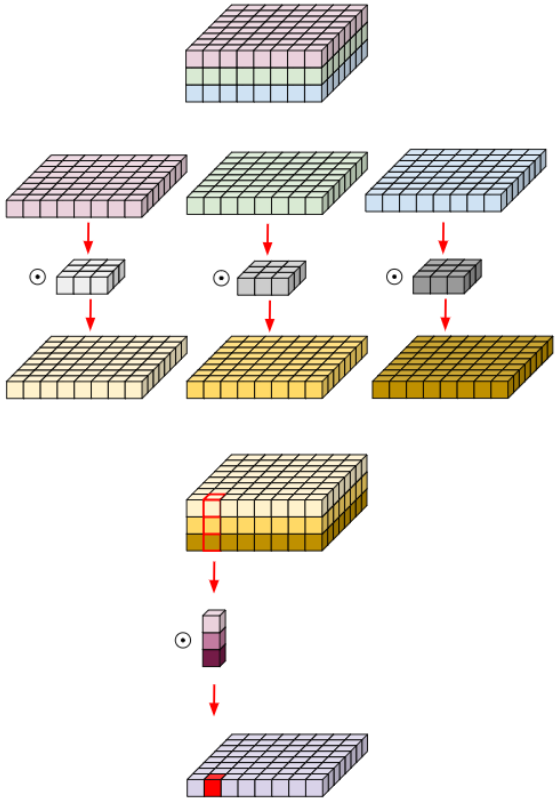
Depthwide Conv

- 단일 채널에 대해서만 수행되는 필터 사용
- 8x8x3 matrix를 Depthwise Conv 하기 위하여 3x3x3 커널 사용.
- 이 때 커널은 채널 방향의 Conv는 진행하지 않고, 공간 방향의 Conv 만을 진행.
- 즉, 각 커널은 하나의 채널에 대해서만 파라미터를 가짐 ➔ 따라서 입력 및 출력 채널의 수 동일. 각 채널 고유의 spatial 정보만을 사용하여 필터 학습.

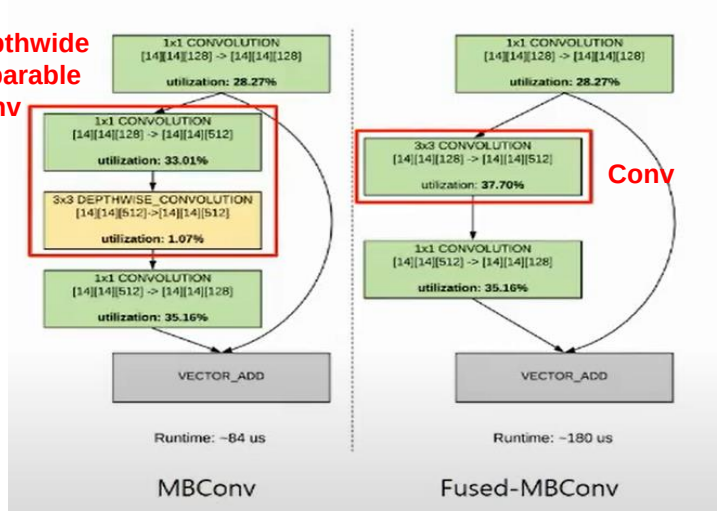
➔ 채널의 출력 값이 하나로 합쳐짐.

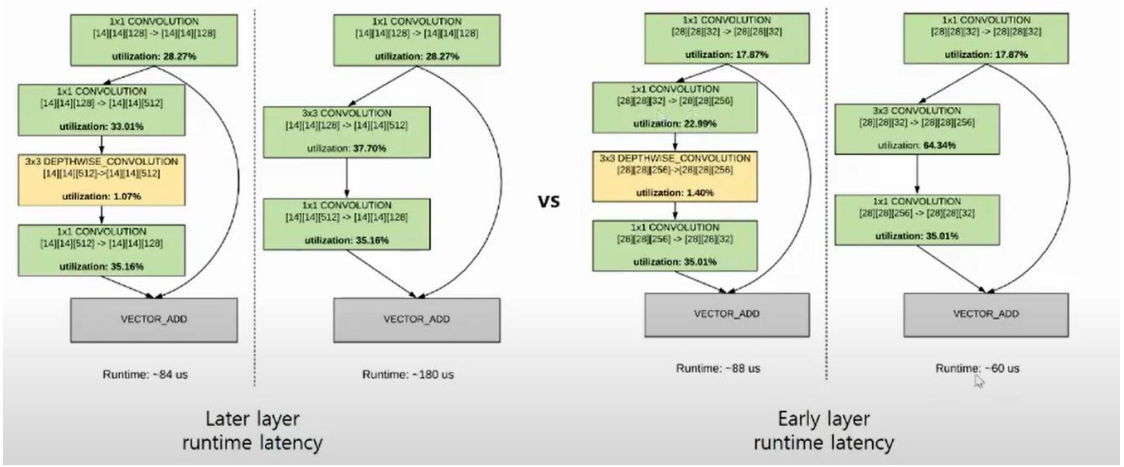
spatial feature 와 channel-wise feature를 모두 고려하여 네트워크 경량화.
기존의 convolution 연산과 거의 유사하게 동작하지만, 파라미터 수와 연산량은 훨씬 적음.

기존 depthwise convolution을 진행한 결과물에 각 채널을 1개의 채널로 압출할 수 있는 추가적인 convolution 진행. => 결과물이 굉장히 간략하게 나옴.



EfficientNetV2 – ② Fused-MBConv





이론적으로는 depthwise 가 빠른데..
Depthwise conv가 초기 layer에 있을수록 연산 속도가 느림을 실험적으로 입증.
➔ Modern hardware가 depthwise conv를 효율적으로 구현하지 못하기 때문.

EfficientNetV2 – ② Fused-MBConv

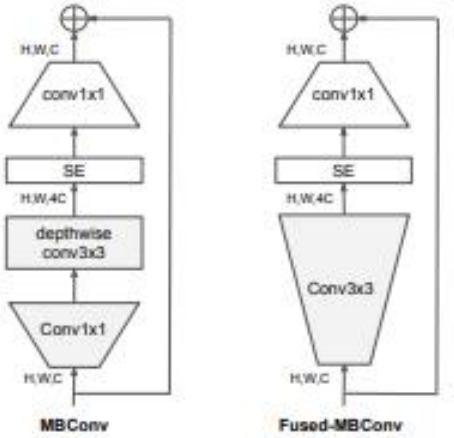


Figure 2. Structure of MBConv and Fused-MBConv.

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

Params 및 FLOP 자체는 단순 MBConv보다 많지만,
실제 throughput은 Fused-MBConv를 조합한 것이 더 빠름.

EfficientNetV2 – ③ Training-aware NAS & non-uniform scaling

EfficientNetV1
- 정확도 & FLOP 을 최적화

EfficientNetV2
- 정확도, 학습 효율 및 파라미터 효율을 NAS의 objective 로 설정하여 최적화
- 학습 효율을 objective로 잡았기 때문에 training-aware..

NAS 설정 값

1) Search Reward: (Accuracy A) * (normalized training step time S^w) * (parameter size P^v)
2) w = -0.07, v = -0.05

Design choices:

1) Conv ops type: {MBConv, Fused-MBConv}
2) Number of layers
3) Kernel size: {3 x 3, 5 x 5}
4) Expansion ratio: {1, 4, 6}

EfficientNetV2 – ③ Training-aware NAS & non-uniform 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	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

EfficientNetV2 – ③ Training-aware NAS & non-uniform scaling

EfficientNetV1과 동일한 방법으로 EfficientNetV2-S를 EfficientNetV2-M/L로 scale up

Additional Optimizations:

- 지나치게 큰 이미지 사이즈는 메모리/학습 속도에 부하를 줄 수 있으므로, inference시 최대 이미지 사이즈를 480으로 제한.
- 즉, 기존의 compound scaling method 대로라면, 더 커질 수 있는 이미지 사이즈를 480으로 제한했기 때문에 non-uniform 함.

- 조금 명확하지 않은 부분은 heuristic하게 runtime 에 부하를 주지 않는 선에서 network capacity (네트워크 용량)를 늘리기 위하여, later stage에 점진적으로 레이어 추가.

EfficientNetV2 – Experiment

Training setups

- EfficientNetV1 과 유사
- RMSProp, decay 0.9, Momentum 0.0
- Batch norm momentum 0.99
- Weight decay 1e-5
- 350 epochs
- Batch size 4096
- Learning rate 0~0.256 (decay: 0.97 every 2.4 epochs)
- Exponential moving average with 0.9999 decay rate

Progressive Learning setups

- 87 epoch 을 1 stage 로 설정, 4 stage 로 분할
- 최대 이미지 사이즈를 inference 사이즈의 약 20% 로 설정

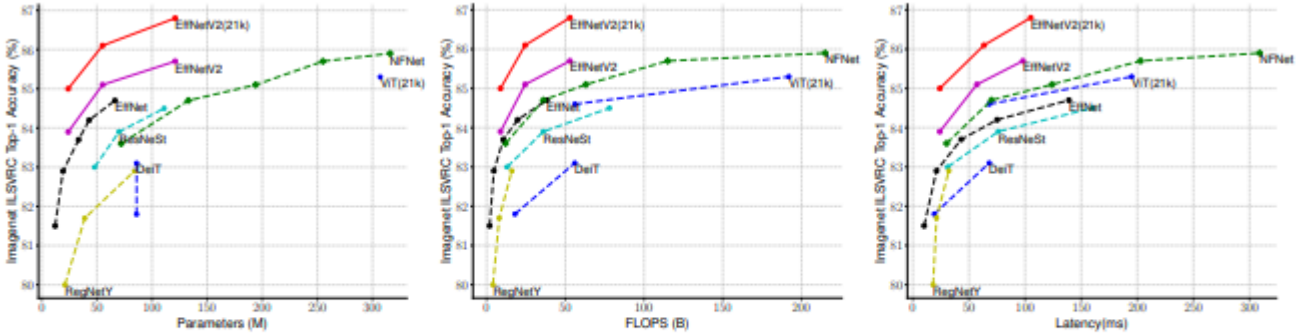
Table 6. Progressive training settings for EfficientNetV2.

	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

Results - ImageNet

	ViT-L/16 (21k) (Dosovitskiy et al., 2021)	85.3%	304M	192B	193	172
ConvNets (ours)	**EfficientNetV2-S**	83.9%	22M	8.8B	24	7.1
	EfficientNetV2-M	85.1%	54M	24B	57	13
	EfficientNetV2-L	85.7%	120M	53B	98	24
	EfficientNetV2-S (21k)	84.9%	22M	8.8B	24	9.0
	EfficientNetV2-M (21k)	86.2%	54M	24B	57	15
	EfficientNetV2-L (21k)	86.8%	120M	53B	98	26
	EfficientNetV2-XL (21k)	87.3%	208M	94B	-	45

We do not include models pretrained on non-public Instagram/JFT images, or models with extra distillation or ensemble.



(a) Parameters (b) FLOPs (c) GPU V100 Latency (batch 16)

Figure 5. **Model Size, FLOPs, and Inference Latency** – Latency is measured with batch size 16 on V100 GPU. `21k` denotes pretrained on ImageNet21k images, others are just trained on ImageNet ILSVRC2012. Our EfficientNetV2 has slightly better parameter efficiency with EfficientNet, but runs 3x faster for inference.

EfficientNetV2 – Ablation Studies

1. 동일한 학습 방법을 EfficientNetV1에 적용한 경우에도, EfficientNetV2가 나은 성능을 보임.

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%)

2. 다른 네트워크에 progressive Learning을 적용하였을 때, 학습 시간이 줄어들고 정확도가 향상됨.

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

3. 모든 이미지 사이즈에 대해 동일한 regularization을 적용한 기존 방법들보다 나은 정확도를 보임

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)

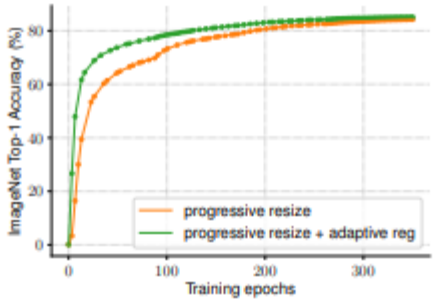


Figure 6. **Training curve comparison** – Our adaptive regularization converges faster and achieves better final accuracy.

EfficientNetV2 – Conclusion

1. 이전 연구 결과 및 EfficientNetV1 의 단점에 대한 분석을 통해 네트워크 아키텍처 개선
2. Adaptive regularization을 동반한 progressive learning 제안
3. 학습 효율 & 파라미터 효율 & 정화도 up

End of the Document