ResNet strikes back:

An improved training procedure in timm

Tae-hwi Kim

ArXiv, 2021

ResNet strikes back

- ***Introduction***

$$\text{accuracy (model)} = f(\mathcal{A}, \mathcal{T}, \mathcal{N}),$$

딥러닝 모델의 성능은 대표적으로 Model Architecture(A), hyper-parameter(τ), measurement noise(N)에 의존한다.

어떤 아키텍처 A_1에 대한 최적의 τ를 구하더라도, A_2에서는 τ가 최적이라는 보장이 없음.

또한 아키텍처를 비교할 때, Baseline 구현의 Training Recipe(hyper-parameters, seed, GPU.. etc)이 모두 다르기 때문에 최적의 결과가 비교되었다고 보기 어려움.

> 따라서 ResNet-50과 ImageNet-1k(224x224) 데이터에 대한 **최적의 Training Recipe**을 찾고자 함.

ResNet strikes back

- ***Training Procedure***

Training Procedure	Number of epochs	Training resolution	Training time	Peak memory by GPU (MB)	Numbers of GPU	Top-1 accuracy val	real	v2
A1	600	224×224	110h	22,095	4	80.4	85.7	68.7
A2	300	224×224	55h	22,095	4	79.8	85.4	67.9
A3	100	160×160	15h	11,390	4	78.1	84.5	66.1

Table 1: Training resources used for our three training procedures on V100 GPUs and corresponding accuracies at resolution 224×224 on ImageNet1k-val, -V2 and -Real. Note, the top-1 val acc. of pytorch-zoo [1] is 76.1%.

> 기본적으로 구성된 훈련 방법은 총 3가지로, 각각의 목적에 맞게 epochs, Resolution이 다르게 설정됨

> 이 구성을 토대로 여러 Optimizer, Regularization등을 추가하거나 hyper-parameter를 Grid Search 통해 찾는 등의 최적화가 진행됨.

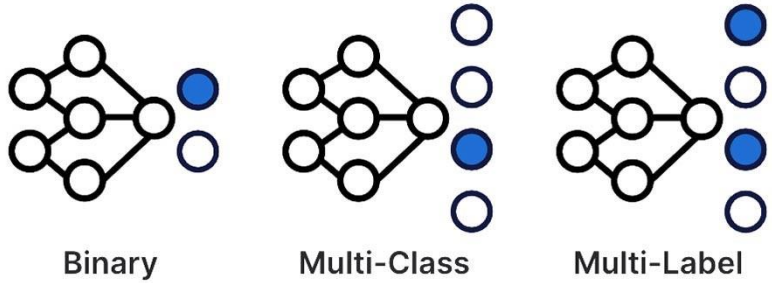
ResNet strikes back

- ***Training Procedure - Components***

Data-Augmentation

GoogleNet부터 사용되었던 Random Resize Crop, Horizontal flip을 사용.

DeiT에서 사용한 RandAugment, Mixup, CutMix (in **timm** library)을 사용.



> 위 Augmentation들을 여러 모델에 따라 조합해가며 사용.

Loss function

Mixup, CutMix를 사용하면서 Distribution Output은 모든 클래스에 대한 확률을 암묵적으로 포함하게 된다.

> Cross-Entropy loss를 사용하게 되면 이 확률이 합성된 이미지에 해당하는 클래스가 아닌 클래스에도 영향을 끼칠 수 있음. (합계가 1이므로)

> Binary Cross-Entropy loss를 사용하여 Multi-label classification 문제로 변형, 합성된 이미지에 각 클래스가 있는지 없는지가 Loss의 기준이 됨.

=> 클래스 간의 독립성을 보장하여 CE에 비해 우수한 성능을 보이는 것을 확인하였음.

ResNet strikes back

- ***Training Procedure - Components***

Regularization

학습 방법 별로 Weight Decay, Label Smoothing, Repeated-Augmentation(RA), Stochastic-Depth(SD)를 적용, Training epoch이 길수록 더 많은 정규화 기법을 사용한다.

Training Procedure	Weight Decay	Label Smoothing	RA & SD
A1	O	O	O
A2	X	X	O
A3	X	X	X

Optimizer

Repeated-Augmentation과 BCE Loss가 LAMB Optimizer와 지속적으로 좋은 결과를 가지는 것을 확인
또한, SGD를 사용했을 때, BCE와 함께 사용 시, 수렴이 잘 안되는 것을 확인

> **기본 Opimizer로 CosineScheduler를 사용하는 LAMB를 사용.**

ResNet strikes back

- ***Experiment Results***

> A2에 비해 A1 방법의 성능이 더 낮은 경우 관찰

정규화가 더 필요한 경우로,

A2 방법에서 81.8%를 달성한 ResNet-152에서

정규화를 더 늘려서 224x224에서 82.4%,

256x256에서 82.7%를 달성하였음

> 즉, 완전한 Training 결과는 아니며 일부의 경우 추가적인 실험을 통해 보완될 수 있음.

Table 3: Comparison on ImageNet classification between other architectures trained with our ResNet-50 optimized training procedure **without any hyper-parameters adaptation**. In particular, our procedure must be adapted for deeper/larger models, which benefit from more regularization. For the training cost we report the training time (time) in hours, the number of GPU used (#GPU) and the peak memory by GPU (Pmem) in GB. For A1 and A2, we adopt the same training and test resolution as in the original publication introducing the architecture. For A3 we use a smaller training resolution to reduce the compute-time. †: torchvision [1] results. *: DeiT [45] results.

	A1-A2-org.		A3		Cost							ImageNet-1k-val			
	train	test	train	test	A1	A2	A1-A2		A3			A1	A2	A3	org.
↓ Architecture	res.	res.	res.	res.	time (hour)		# GPU	Pmem	time	# GPU	Pmem	Accuracy(%)			
ResNet-18 [13]†	224	224	160	224	186	93	2	12.5	28	2	6.5	71.5	70.6	68.2	69.8
ResNet-34 [13]†	224	224	160	224	186	93	2	17.5	27	2	9.0	76.4	75.5	73.0	73.3
ResNet-50 [13]†	224	224	160	224	110	55	4	22.0	15	4	11.4	80.4	79.8	78.1	76.1
ResNet-101 [13]†	224	224	160	224	74	37	8	16.3	8	8	8.5	81.5	81.3	79.8	77.4
ResNet-152 [13]†	224	224	160	224	92	46	8	22.5	9	8	11.8	82.0	81.8	80.6	78.3
RegNetY-4GF [32]	224	224	160	224	130	65	4	27.1	15	4	13.9	81.5	81.3	79.0	79.4
RegNetY-8GF [32]	224	224	160	224	106	53	8	19.8	10	8	10.3	82.2	82.1	81.1	79.9
RegNetY-16GF [32]	224	224	160	224	150	75	8	25.6	13	8	13.4	82.0	82.2	81.7	80.4
RegNetY-32GF [32]	224	224	160	224	120	60	16	17.6	12	16	9.4	82.5	82.4	82.6	81.0
SE-ResNet-50 [20]	224	224	160	224	102	51	4	27.6	16	4	14.2	80.0	80.1	77.0	76.7
SENet-154 [20]	224	224	160	224	110	55	16	23.3	12	16	12.2	81.7	81.8	81.9	81.3
ResNet-50-D [14]	224	224	160	224	100	50	4	23.9	14	4	12.3	80.7	80.2	78.7	79.3
ResNeXt-50-32x4d [51]†	224	224	160	224	80	40	8	14.3	15	4	14.6	80.5	80.4	79.2	77.6
EfficientNet-B0 [41]	224	224	160	224	110	55	4	22.1	15	4	11.4	77.0	76.8	73.0	77.1
EfficientNet-B1 [41]	240	240	160	224	62	31	8	17.9	8	8	7.9	79.2	79.4	74.9	79.1
EfficientNet-B2 [41]	260	260	192	256	76	38	8	22.8	9	8	11.9	80.4	80.1	77.5	80.1
EfficientNet-B3 [41]	300	300	224	288	62	31	16	19.5	6	16	10.1	81.4	81.4	79.2	81.6
EfficientNet-B4 [41]	380	380	320	380	64	32	32	20.4	8	32	14.3	81.6	82.4	81.2	82.9
ViT-Ti [45]*	224	224	160	224	98	49	4	16.3	14	4	7.0	74.7	74.1	66.7	72.2
ViT-S [45]*	224	224	160	224	68	34	8	16.1	8	8	7.0	80.6	79.6	73.8	79.8
ViT-B [11]*	224	224	160	224	66	33	16	16.4	5	16	7.3	80.4	79.8	76.0	81.8
timm [50] specific architectures															
ECA-ResNet-50-T	224	224	160	224	112	56	4	29.3	15	4	15.0	81.3	80.9	79.6	-
EfficientNetV2-rw-S [42]	288	384	224	288	52	26	16	16.6	7	16	10.1	82.3	82.9	80.9	83.8
EfficientNetV2-rw-M [42]	320	384	256	352	64	32	32	18.5	9	32	12.1	80.6	81.9	82.3	84.8
ECA-ResNet-269-D	320	416	256	320	108	54	32	27.4	11	32	17.8	83.3	83.9	83.3	85.0

ResNet strikes back

- ***Experiment Results – Significance of measurements***

Hyper-parameter를 고정해도, 여러 단계에서 무작위성에 의존하는 경우가 있음. (e.g. weight normalization, batch 상의 이미지가 네트워크에 들어가는 순서..)

또한 이전 연구*(David Picard. torch.manual seed(3407) is all you need)*에서, 성능에 큰 영향을 주는 이상값 Seed가 있다는 결론이 존재함.

> Seed 값이 정확도에 끼치는 영향이 어느 정도인지 확인해 볼 필요가 있음.

100개의 고유한 시드(seed: 0~100)를

A2, Resnet-50, ImageNet 데이터셋에 대해 정확도를 측정.

> Seed값에 따라 정확도가 정규 분포를 가짐을 확인

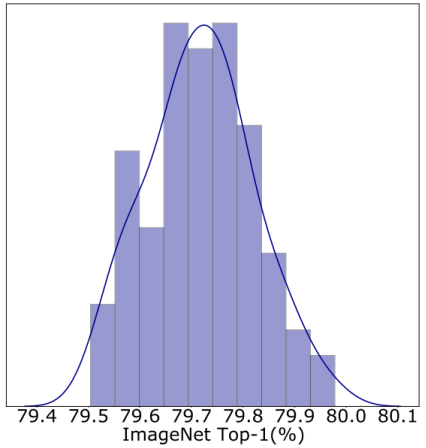


Figure 2: Distribution of the performance on ImageNet-val with the A2 procedure. It is measured with 100 different seeds. We also depict the Gaussian-fit of this distribution.

ResNet strikes back

- ***Experiment Results – Transfer Learning***

Table 5: Performance comparison on transfer-learning tasks for different pre-training recipes.

Dataset	Train size	Test size	#classes	Pytorch [1]	A1	A2	A3
ImageNet-val [36]	1,281,167	50,000	1000	76.1	**80.4**	79.8	78.1
iNaturalist 2019 [18]	265,240	3,003	1,010	73.2	73.9	**75.0**	73.8
Flowers-102 [29]	2,040	6,149	102	**97.9**	**97.9**	**97.9**	97.5
Stanford Cars [24]	8,144	8,041	196	92.5	**92.7**	92.6	92.5
CIFAR-100 [25]	50,000	10,000	100	86.6	**86.9**	86.2	85.3
CIFAR-10 [25]	50,000	10,000	10	98.2	**98.3**	98.0	97.6

서로 다른 3개의 방법을 통해 Pre-train(ImageNet)한 모델에 대한 각 데이터셋에 대한 Fine-tuning 결과

ResNet strikes back

- ***Ablation Study – Main ingredients***

A2에 대한 Learning rate에 대한 Ablation 결과
> 0.005가 최적임을 확인함

Loss를 CE로 사용하면서 성능이 하락,
하지만 모든 CE가 BCE에 비해 낮은 것은 아님.

RA의 경우 A3와 같이 짧거나 Mixup의 alpha값이
높은 경우에는 좋지 않은 결과가 나옴을 확인.
> A1, A2에만 적용.

loss	LR	WD	RA	A2
BCE	2×10^{-3}	0.02	✓	78.24
BCE	2×10^{-3}	0.03	✓	78.47
BCE	3×10^{-3}	0.02	✓	79.16
BCE	3×10^{-3}	0.03	✓	79.28
BCE	5×10^{-3}	0.01	✓	79.66
BCE	5×10^{-3}	0.02	✓	79.85
BCE	5×10^{-3}	0.03	✓	79.73
BCE	8×10^{-3}	0.02	✓	79.63
BCE	3×10^{-3}	0.02	✗	78.74
BCE	5×10^{-3}	0.02	✗	79.57
BCE	5×10^{-3}	0.03	✗	79.58
CE	2×10^{-3}	0.02	✓	77.37
CE	3×10^{-3}	0.02	✓	78.22
CE	5×10^{-3}	0.02	✓	79.18
CE	5×10^{-3}	0.03	✓	79.23
CE	5×10^{-3}	0.05	✓	79.31
CE	8×10^{-3}	0.03	✓	79.12
CE	3×10^{-3}	0.02	✗	77.71
CE	5×10^{-3}	0.01	✗	78.93
CE	5×10^{-3}	0.02	✗	79.00
CE	5×10^{-3}	0.03	✗	78.62
CE	8×10^{-3}	0.02	✗	78.72

Table 6: Main ablation table with A2 procedure. We compare BCE vs CE, including repeated augmentation or not, and vary the learning rate LR and weight decay WD in ranges that our exploration phase has identified as being the most adapted. All results are reported with Seed 0 and therefore all the ResNet-50 are initialized with the same weights when the training starts. The highlighted row corresponds to our A2 procedure.

ResNet strikes back

- ***Ablation Study – Regularization & Augmentation***

Smoothing의 경우 A1에서 성능 향상,

Stochastic Depth와 같은 경우에는
학습 시간이 짧은 A3에는 효과적이지 못했음

Crop-ratio를 결정하기 위해
10개의 Seed값을 통해 최적의 값을 구한 건
0.95였으나, 큰 차이를 보이지 못했고
0.9가 A1에서 가장 높은 정확도를 보임.

drop-factor	A1	A2	A3
0	79.94	79.79	78.06
0.05	80.38	79.85	77.57
0.1	80.12	79.62	77.32
smoothing			
✗	80.22	79.85	78.06
✓	80.38	79.58	77.99

	A1			A2			A3		
crop-ratio	mean (*std*)	max – min	seed 0	mean (*std*)	max – min	seed 0	mean (*std*)	max – min	seed 0
0.875	80.18 (*0.14*)	80.45 – 79.90	80.14	79.67 (*0.08*)	79.91 – 79.59	79.91	77.69 (*0.10*)	77.85 – 77.48	77.69
0.9	80.22 (*0.15*)	**80.54** – 79.98	80.25	79.73 (*0.09*)	79.89 – 79.56	79.75	77.86 (*0.09*)	78.01 – 77.62	77.83
0.95	80.24 (*0.14*)	80.49 – 79.91	80.38	79.68 (*0.09*)	79.85 – 79.57	79.85	78.00 (*0.09*)	78.09 – 77.83	78.06
1.0	80.15 (*0.11*)	80.15 – 79.66	80.19	79.58 (*0.13*)	79.88 – 79.32	79.88	78.02 (*0.10*)	78.16 – 77.83	77.93

Table 9: Ablation of the crop-ratio when training with A1. We compute the Imagenet-val top-1 accuracy as a function of this parameter for 10 different seeds, for ResNet50 trained with our procedures. Our selection of 0.95 was based on Seed 0 in early experiments. It is comparable but not statistically better than the standard 0.875. Note that we have one A1 seed that leads to a top 80.54% top-1 accuracy at crop-ratio 0.9. We regard it as being overfit and therefore we do not recommend to report this number.

ResNet strikes back

- ***Ablation Study – Resolution***

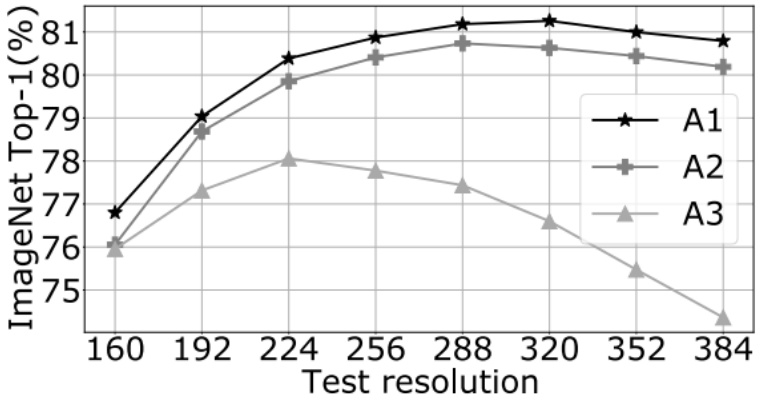


Figure 4: We compare ImageNet Top-1 accuracy according to the test resolution for our three training procedures A1, A2 and A3 with ResNet-50 architecture. Our training procedure and models also benefit from the FixRes effect [48]: the performance increases when using a larger image at test time for the procedures A1 and A2. This observation is not true for A3, which is expected since this procedure was already relying on feeding smaller images at train time, so as to maximize the accuracy at test resolution 224×224.

A1과 A2에 대해서 해상도가 높아지더라도 견고한 성능을 보임을 확인함.

ResNet strikes back

- ***Main Contiributions***

: Vanilla ResNet-50을 학습하는 최적의 성능과 효율성을 내는 여러 학습 방법 제시,
224x224 ImageNet-eval 데이터셋에 대해 80.4% Top-1 Accuracy (SOTA) 달성
(no extra data or distillation)

- ***Pros***

: 오픈 소스인 timm을 사용해 hyper-parameter, augmentation만으로 더 우수한 Baseline을 쉽게 구성 가능,
단순히 성능만이 아닌 리소스를 고려하면서도 성능을 유지하는 학습 방법 제시.

- ***Cons***

: 최적화 범위가 ResNet-50, ImageNet으로 한정되어 있어 다양한 모델에 실용적으로 적용하기에는 어려움.
사용된 Augmentation 기법이 다양하지 않고 이전 연구에 의존적: 보편성을 보장하기 어려움.