ResNet strikes back

Introduction

딥러닝 모델의 성능은 **A(Model Arichtecture), T(hyper-paramters), N(measurement noise, 편차)**로 계산 가능

seed값과 같은 N의 환경에 의해서도 모델의 성능이 바뀔 수 있음. 이에 따라 표준 편차를 구하거나 분리된 검증 데이터셋을 사용하기도 함. N을 빼고 생각하더라도, A,T에도 너무 변수가 많다.

→ (A_1,T)에 대한 최적 파라미터를 찾는다고 하더라도 그 하이퍼 파라미터가 A_2 모델에도 최적일지는 보장할 수 없기 때문임.

또한 대부분의 논문에서 아키텍처를 비교할 때, 이전 논문과 자신들의 결과를 비교

→ 구현 상황에서의 recipe(hyperparamter, seed 등)이 모두 다르기 때문에 최적의 결과끼리 비교된 것이라고 볼 수 없음.

ResNet50의 ImageNet-1k-val 데이터셋의 성능에서 대부분의 논문들은 75.2% ~ 79.5% 성능 기록.

하지만 Baseline에 대한 충분한 연구가 이루어졌는지 확실하지 않음.

> → **따라서 바닐라 ResNet-50(+SE-ResNet-50, ResNet-50-D)과 224x224의 이미지 데이터에 대한 최적의 Training Recipe를 찾고자 함.**
>
> Supposed on.. → 서로 다른 epoch(100, 300, 600)에서의 hyper-parameters and ingredient에 따른 변화 기록 일반적인 CE Loss가 아닌 Mixup, CutMix를 사용한 BCE Loss로 Classification, 모든 synthetizied image가 augmentation을 통해 섞여있다는 것을 가정

Training Procedures

세 가지 다른 Training 방법, ResNet-50, Resolution 224x224, 다양한 변형 in optimizer, regularization, grid search of hyper-parameters

- Procedure A1: 가장 좋은 성능이 목적, 가장 긴 epoch과 Training Time
- Procedure A2: 여타 최신 학습방법(like DeiT)과 비교를 위한 방법으로 batch size와 같은 부분을 빼면 크게 다른 게 없음, 300 epoch을 가짐
- Procedure A3: 100epoch, 2048 batchsize, for exploratory research or studies

Loss

CE와 Mixup, Cutmix을 사용하게 되면, 출력은 이미지에 존재하는 모든 클래스에 대해 mixed concept이 존재할 확률이 된다. (a distribution of concepts that sum to 1)

실제 학습에서는, Augmentation을 통해 concept(class)이 모두 섞여있다고 가정하고, 이에 대한 BCE Loss를 계산한다 (multi-label classification, 1 vs all,)

→각 클래스 별로 합성된 이미지에 존재하는 지 안하는 지만 check, **Independence of ohter classes**

이 방법(BCE)이 Mixup, Cutmix augmentation을 적용할 때 CL보다 더욱 효과적이었다.

Data Augmentation

Random Resized Crop, horizontal flip from GoogleNet,

timm에 구현되어 있는 Augmentation 사용; RandAugment, Mixup, CutMix from DeiT

Random Erasing이 CutMix보다 Regulrizaton을 올려준다? → Appendix A

Regularization

세가지 Training 방법 중에서, 가장 다른 부분이 Regularizaiton임, epcoh이 모두 달라서 그럴지도

일단 Weight decay, label smoothing, Repeated Augmentation, stochastic-depth를 사용한다.

Repeated Augementation, Stochastic depth은 수렴 시의 성능을 올려주는 결과를 보였지만, 초기 학습이 느려지는 현상이 있어 epoch이 짧은 경우에는 사용되지 않았음.

A1 : label smoothing, Repeated Augementation, Stochastic depth

A2 : Repeated Augementation, Stochastic depth,

모델이 더 커질수록 Regularization을 추가하는 게 좋다.

Optimization

AlexNet 이후로 Convolutional Network의 optimizer는 주로 SGD였음.

트랜스포머 모델이나 MLP는 AdamW나 LAMB를 사용하는 경우가 많았다. 하지만 Dosovitskuy에 의하면 AdamW나 SGD나 ResNet-50에서는 비슷하다고 함, 실제로 중간 사이즈의 배치 사이즈에서 그런 경향을 확인함.

→ Repeated Augmentation, 큰 배치사이즈 2048, BCE를 사용하는 상황에서, LAMB가 가장 좋은 결과를 유지하는 데 쉬웠음.

BCE랑 SGD를 같이 쓰면 수렴하는 걸 잘 못봤다. 따라서 LAMB를 optimizer로 쓰고, Cosine scheduler를 쓰는 데에 중심을 뒀다.

⇒ ResNet 모델에는 수정을 가하지 않았음. 최대한 대표성을 나타내게끔 많은 트레이닝 방법을 시도해보았지만 모든 방법을 시도해본 것은 아님, distillation이나 self-supervised 방법은 사용하지 않았음.

Experiment Results

ResNet-50을 학습시키기 위한 하이퍼 파라미터 세팅을 그대로 더 큰 모델의 학습에 그대로 사용,

ResNet-152의 경우, A2 방법을 통해 81.8% 성능 기록

특히 ResNet에 최적화된 세팅답게 ResNet에서 기존보다 더 좋은 성능을 주로 기록했다.

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

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

- Seed experiment

하이퍼 파라미터는 fixed, 몇몇 단계에서 random factor에 의존하는 경우도 있음 (e.g. weight initialization, optimization, 이미지가 네트워크에 들어가는 순서 등..)

→ 정확도에 얼마나 영향을 끼칠까? : 이전 연구 결과에 따르면, outlier의 존재가 큰 outperforming, underperforming에 영향을 끼치는 것으로 밝힘.

	Top-1 accuracy (%)				
dataset ↓	mean	std	max	min	seed 0
ImageNet-val	79.72	0.10	79.98	79.50	79.85
ImageNet-real	85.37	0.08	85.55	85.21	85.45
ImageNet-V2	67.99	0.23	68.69	67.39	67.90

A2 방법으로, 100개의 모두 다른 seed값으로 실험해 본 결과, 오직 훈련 끝 단계에서의 top-1 acc임.

IID하지 않고 시드 값에 따라 정확도가 정규분포를 가지는 것을 확인했다.

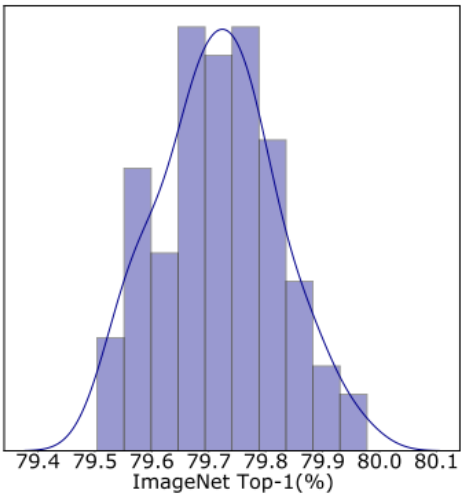

Figure 2: Distribution of the performance on ImageNet-val with the A2 procedure. It is measured with 100 different seeds. We also depict the Gaussian-fit of this distribution.