심화학습기초 Term-Project

HeeWon-Lee

경북대학교, 전자공학부

saebuk2000@knu.ac.kr

Ⅰ. 서 론

사용된 데이터셋은 CIFAR-10이며, 사용된 모델은 ResNet18 모델이다. Baseline 코드가 공개되기 이전에 실험했던, Single Model에서 성능을 올리기 위해 했던 방법들과 Baseline 코드가 공개되고 난 후 Ensemble 방법을 사용하여 성능을 올리기 위해 했던 방법들을 정리한다.

Ⅱ. Single Model에서의 성능 향상 방법

아래 방법들은 baseline 코드가 공개되기 이전에 수행되었다.

1. optimizer

SAM[1]은 loss value를 낮추고 동시에 sharpness도 최소화함으로써 model generalization을 개선한 최적화 방법이다. SAM은 한 iteration마다 2번의 step을 거친다. 첫 번째 step에서는, parameter에 대한 loss의 gradient를 계산하고 weights를 업데이트한다. 두 번째 step에서는, 새 parameter 값의 인접 관계를 중심으로 gradient를 다시 계산하며, 이 인접 관계에서의 최대 loss를 최소화한다. 주변 인접 관계를 중심으로 gradient를 다시 계산하므로 그림 1에서 볼 수 있듯이 낮은 곡률을 가질 수 있기에, 더 나은 model generalization이 가능하다.

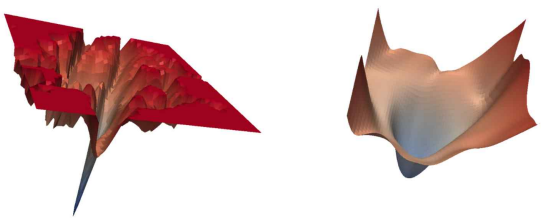

그림 1 SGD(좌)와 SAM(우)의 곡률 비교

ASAM[2]은 SAM에서 발생하는 스케일 의존성 문제를 해결한 최적화 방법이다. SAM은 parameter re-scaling에 민감하다는 단점이 있다. ASAM은 parameter re-scaling에 강한 Adaptive Sharpness Method를 적용한 방법이다.

표 1에서 SGD와 SAM, 그리고 ASAM 최적화를 적용했을 때의 성능을 비교한 ASAM 논문의 실험 결과를 볼 수 있다.

[3]의 논문에서는 SAM 또는 SGD를 사용할 때, 그림 2에서 볼 수 있듯이 batch-size를 16으로 설정하였을 때 성능이 가장 높다고 하였다. [4]의 논문에서는 batch-size가 16일 때, learning rate를 0.0001로 했을 때가 성능이 더 좋았다고 하였다. 이와 같은 이유로 batch-size는 16으로, learning rate는 0.0001로 설정하였다.

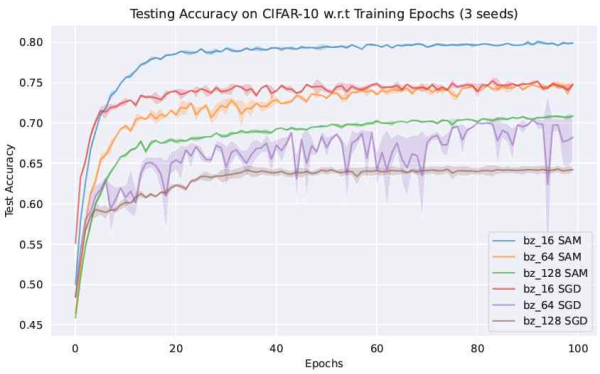

그림 2 batch-size를 다르게 하여 SGD와 SAM으로 CIFAR-10을 학습하였을 때 성능 비교

Model	SGD	SAM	ASAM
DenseNet-121	$91.00_{\pm 0.13}$	$92.00_{\pm 0.17}$	$\mathbf{93.33}_{\pm 0.04}$
ResNet-20	$93.18_{\pm 0.21}$	$93.56_{\pm 0.15}$	$\mathbf{93.82}_{\pm 0.17}$
ResNet-56	$94.58_{\pm 0.20}$	$95.18_{\pm 0.15}$	$\mathbf{95.42}_{\pm 0.16}$
VGG19-BN*	$93.87_{\pm 0.09}$	94.60	$\mathbf{95.07}_{\pm 0.05}$
ResNeXt29-32x4d	$95.84_{\pm 0.24}$	$96.34_{\pm 0.30}$	$\mathbf{96.80}_{\pm 0.06}$
WRN-28-2	$95.13_{\pm 0.16}$	$95.74_{\pm 0.08}$	$\mathbf{95.94}_{\pm 0.05}$
WRN-28-10	$96.34_{\pm 0.12}$	$96.98_{\pm 0.04}$	$\mathbf{97.28}_{\pm 0.07}$
PyramidNet-272†	$98.44_{\pm 0.08}$	$98.55_{\pm 0.05}$	$\mathbf{98.68}_{\pm 0.08}$

표 1 CIFAR-10에서 optimizer 성능 비교

2. scheduler

[5]의 논문에선 SGD를 사용할 때, cosine LR scheduler를 사용하는 것이 성능이 가장 좋다고 하였다. SAM 및 ASAM의 기초 optimizer로 SGD를 사용하고 있기에 scheduler는 cosine LR scheduler를 사용하였다.

3. Loss function

CIFAR-10 데이터셋에서 잘못 labeling 된 데이터 혹은 사람도 구분하기 어려운 데이터가 있지 않을까 싶어 Asymmetric Loss[6]를 적용하였다. 이 Loss function은 Multi-label Classification에서 발생하는 class imbalance와 Mislabeling 문제를 해결하기 위해 제안된 방법이지만, single-label classification에서도 적용할 수 있게 코드가 공개되어 있었기에 사용할 수 있었다.

Asymmetric Loss와 CrossEntropy Loss의 성능 차이 비교는 실험하지 못하였다.

4. Data Augmentation And DataLoader

Data Augmentation 방법은 [7]의 방법을 참고하였다. 또한 학습 속도를 향상하기 위해 Albumentation[8]을 적용하였다.

그리고 DataLoader에서 pin_memory를 True로 설정하고, num_workers를 32로 설정하여 학습 속도를 더욱 빠르게 하였다.

5. Model EMA

학습 과정에서 model generalization을 극대화하기 위해 exponential moving average[9]를 적용하였다. EMA라고 불리는 이 방법은 기존 가중치의 이동 평균을 계산하고 새로운 가중치 업데이트에 대한 평균값을 사용하여 가중치를 계산하기에, 모델 파라미터의 변동성이 줄어들고, 수렴이 더 빠르고 안정적으로 이루어질 수 있다고 한다.

6. 실험

위와 같은 방법을 적용하여 학습을 진행하였다. 학습 결과 Train Loss는 0.001918, Validation Loss는 0.07747, Train accuracy는 0.9999, Validation accuracy는 0.9741가 나왔다. Accuracy 그래프는 그림 3에서 볼 수 있다.

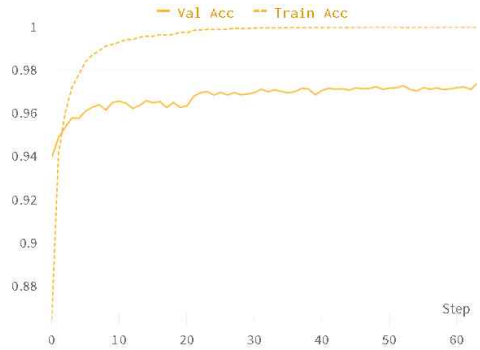

그림 3 epoch에 따른 Accuracy 그래프

Ⅲ. Ensemble Model에서 성능 향상

1. Methods change

이전 chapter의 방법으로 모델을 여러 개 학습한 뒤 Ensemble 하여 성능을 평가하였을 때, 성능이 많이 증가하지 못하였다.

ASAM을 적용하여 Ensemble 하였을 때보다 적용하지 않았을 때가 성능이 더 증가하였기에 ASAM을 적용하지 않고, [4]를 참고하여 Adam을 사용하고 파라미터를 설정하였다.

scheduler는 optimizer를 Adam으로 변경하였기에 [5]의 방법을 적용할 수 없어, baseline의 StepLR을 사용하였다.

Data Augmentation은 Albumentation으로 학습하고 baseline의 inference 코드로 추론했을 때 성능이 많이 감소하여, 학습과 추론 모두 transforms를 사용하였다.

Loss function은 그대로 사용하였으며, Model EMA는 적용하였을 때보다 적용하지 않았을 때 성능이 높아 EMA 방법은 사용하지 않았다.

2. 실험

random seed를 0부터 10까지 바꿔가며 학습하였다. 모델의 개수를 달리하여 추론하였을 때, 6개의 모델로 Ensemble 한 결과가 가장 높게 나왔다.

Inference 할 때, 각 데이터의 예측값을 모아 빈도가 가장 높은 label을 최종 예측값으로 정하였다. 하지만 이 방법은 빈도가 가장 높은 label이 여러 개일 때, 잘못 예측할 수 있는 문제점이 있어, soft voting 방법을 적용하여 문제를 해결하였고, 좀 더 높은 성능을 달성할 수 있었다.

위의 방법들을 통해 모델의 정확도는 97.68%를 달성하였다.

Ⅳ. 결 론

모델의 성능을 올리기 위해 수많은 논문을 찾아보며, 나의 연구 역량을 늘릴 수 있었던 뜻깊은 시간이었다.

하지만 모델이 제한되어 있었고, 데이터셋의 난이도가 다른 데이터셋에 비해 높지 않았기 때문에, 변별력이 낮아 0.01% 차이로 순위가 뒤바뀌는 상황이 자주 발생하였다.

모델은 ResNet18을 그대로 사용하더라도, 난이도가 좀 더 높지만 크기는 크지 않은 데이터셋으로 선정하면 좀 더 변별력이 생기지 않을까 생각해본다.

References

〔1〕 Foret, Pierre, et al. "Sharpness-aware minimization for efficiently improving generalization." arXiv preprint arXiv:2010.01412 (2020).

〔2〕 Kwon, Jungmin, et al. "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks." International Conference on Machine Learning. PMLR, 2021.

〔3〕 Coldenhoff, Jozef Marus, Chengkun Li, and Yurui Zhu. "Model Generalization: A Sharpness Aware Optimization Perspective." arXiv preprint arXiv:2208.06915 (2022).

〔4〕 Kandel, Ibrahem, and Mauro Castelli. "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset." ICT express 6.4 (2020): 312-315.

〔5〕 Moreau, Thomas, et al. "Benchopt: Reproducible, efficient and collaborative optimization benchmarks." Advances in Neural Information Processing Systems 35 (2022): 25404-25421.

〔6〕 Ridnik, Tal, et al. "Asymmetric loss for multi-label classification." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

〔7〕 https://www.kaggle.com/code/ayushnitb/cifar10-custom-resnet-cnn-pytorch-97-acc

〔8〕 Buslaev, Alexander, et al. "Albumentations: fast and flexible image augmentations." Information 11.2 (2020): 125.

〔9〕 Cai, Zhaowei, et al. "Exponential moving average normalization for self-supervised and semi-supervised learning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.