실전기계학습 결과정리 서보민

Guide code의 resnet의 매개변수, 데이터 증강과정 수정

➔ 56% 의 낮은 정확도

Github davda54 유저의 WRN (https://github.com/davda54/sam/tree/main/example)

선정 이유

- 강의에서 들은 resnet과 wideresnet의 개념을 토대로 모델의 활용과 매개변수 선정이 용이할 것으로 생각돼서 선정했습니다.

이론적 배경

- 1. WRN의 깊이 : CONV1에서 1개와 CONV2, CONV3, CONV4에서 Shortcut connection형태로 층이 쌓이게 되어 기본적으로 4개의 layer가 있어야 하며, conv2, conv3, conv4에서 각각 3*3 형태의 2개의 layer가 쌓이므로 깊이는 6n + 4의 값을 가져야 합니다

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Table 1: Structure of wide residual networks. Network width is determined by factor k. Original architecture [13] is equivalent to $k = 1$. Groups of convolutions are shown in brackets where N is a number of blocks in group, downsampling performed by the first layers in groups `conv3` and `conv4`. Final classification layer is omitted for clearance. In the particular example shown, the network uses a ResNet block of type $B(3,3)$.

- (참고 : https://cumulu-s.tistory.com/36)
- 2. 매개 변수의 수와 정확도의 상관 관계

depth	k	# params	CIFAR-10	CIFAR-100
40	1	0.6M	6.85	30.89
40	2	2.2M	5.33	26.04
40	4	8.9M	4.97	22.89
40	8	35.7M	4.66	-
28	10	36.5M	**4.17**	20.50
28	12	52.5M	4.33	**20.43**
22	8	17.2M	4.38	21.22
22	10	26.8M	4.44	20.75
16	8	11.0M	4.81	22.07
16	10	17.1M	4.56	21.59

Table 4: Test error (%) of various wide networks on CIFAR-10 and CIFAR-100 (ZCA preprocessing).

- Depth와 width를 조절하면 매개변수를 조절했을 시, 매개 변수가 많아질수록 정확도는 높아지는 경향을 보임을 알 수 있습니다.

- 따라서 500만 개 이하의 매개 변수에서 가장 최적의 정확도를 가지는 depth와 width를 찾기 위한 test를 진행하였습니다.
- Optim 선택 기준 : 참고
- https://proceedings.neurips.cc/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf
- SGD와 Adam에 대해 고민하였고, Adam에 비해 SGD가 더 나은 일반화 성능을 나타낸다는 결론을 토대로 SGD를 선택하였습니다.

Depth	Width	Lr	Optim	Batch	Train acc	Val acc	Kaggle acc
40	2	0.1	SGD	64	93	91	0.93450
28	3	0.1	SGD	64	93	91	0.92700
10	5	0.1	SGD	64	92	90	0.92700
16	4	0.1	SGD	64	92	93	0.93650

4번의 결과를 통해 모델의 정확도는 매개변수의 수에 비례하여 증가한다는 것을 알 수 있었으며 depth : 16 / width : 4일 때 상대적으로 높은 정확도를 얻을 수 있었습니다.

5번째 WRN (Depth 16 / width 4 고정)

Depth – 16 / width – 4 / learning rate : 0.2 / Optimizer : SGD / batch size : 64

```
class StepLR:
    def __init__(self, optimizer, learning_rate: float, total_epochs: int):
        self.optimizer = optimizer
        self.total_epochs = total_epochs
        self.base = learning_rate

    def __call__(self, epoch):
        if epoch < self.total_epochs * 3/10:
            lr = self.base
        elif epoch < self.total_epochs * 5/10:
            lr = self.base * 0.2
        elif epoch < self.total_epochs * 7/10:
            lr = self.base * 0.2**2
        elif epoch < self.total_epochs * 9/10:
            lr = self.base * 0.2**3
        else:
            lr = self.base * 0.2 ** 4

        for param_group in self.optimizer.param_groups:
            param_group["lr"] = lr

    def lr(self) -> float:
        return self.optimizer.param_groups[0]["lr"]
```

4번의 결과로 얻어진 정확도의 경향 분석을 통하여 초기 learming rate를 높이고 learning rate를 조절하는 lr_scheduler를 커스터마이징하여 정확도 갱신이 잘 되지 않는 epoch에 스케줄러 과정을 추가함으로써 더 높은 정확도를 얻을 수 있었습니다.

➔ Trainset 정확도 : 95% valid set 정확도 92% (점수 0.9365)

6번째 WRN

Depth – 16 / width – 4 / learning rate : 0.2 / Optimizer : Adam / batch size : 16

변경 사항

- 옵티마이저 SGD >> Adam

  ➔ https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

  ➔ Image classfication 문제에 있어 SGD보다 Adam이 더 좋은 성능을 낸다는 이론적 설명을 바탕으로 옵티마이저를 교체하였습니다.

- Batch size : 64 >> 16

- 적은 batch size와 낮은 lr을 가질수록 성능 개선이 이루어졌다는 통계 결과를 보고 배치 사이즈를 64에서 16으로 줄여 진행하였습니다.

- (참고 : https://inhovation97.tistory.com/32)

  ➔ Trainset 정확도 : 95% valid set 정확도 93% (점수 0.937)

그외의 과정

- Lr을 0.2 ~ 0.001을 다양한 값을 할당하여 최적의 학습률을 찾기 위해 노력했으며 NAdam, RMsprop 등으로 optimizier를 교체해보았습니다.

- 같은 높이와 넓이를 가짐에도 비교적 더 적은 매개변수를 가져 높이와 넓이에 더 높은 값을 할당할 수 있는 peter kim의 WRN(https://github.com/PeterKim1/paper_code_review)을 사용하여 측정해봤으며, Dvdda54 유저의 WRN과 peter kim 유저의 WRN의 강점을 취사선택하여 모델을 구현하였으나 뚜렷한 상승폭이 없어 모델을 WRN에서 Effnet으로 교체했습니다.

Effnet – B0 (https://github.com/google-research/sam)

- **선정 이유** : WRN이 93~94%에서 더 좋은 성능을 내기 어려웠으며 이번 프로젝트의 데이터셋과 유사하다고 생각되는 CIFAR100의 데이터셋의 딥러닝에 가장 좋은 성능을 낸 EFFNET-L2 선택하였습니다. 매개변수 5백만 이하의 조건을 맞추기 위하여 EFFNET-B0으로 수정하여 모델을 실행하였습니다.
- **이론적 배경**
- https://dacon.io/en/forum/406054

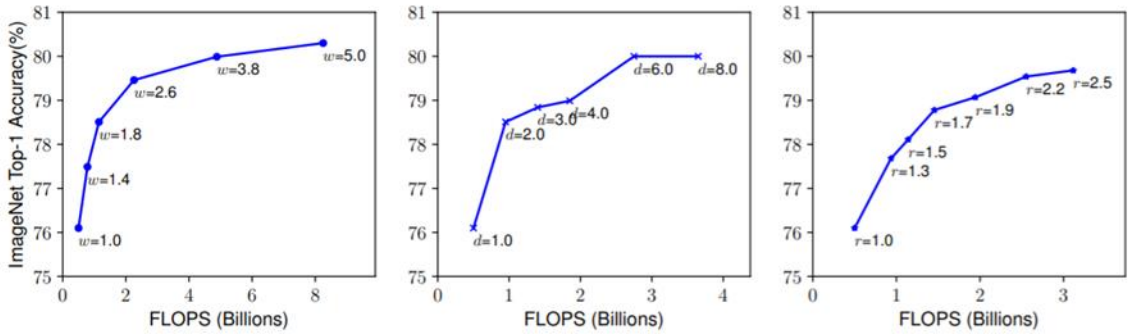


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

-

- effnet에서 depth와 width, 이미지의 해상도가 올라갈수록 높은 정확도를 얻을 수 있습니다.

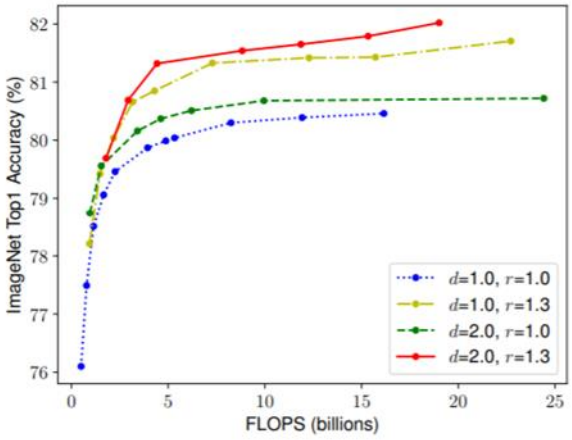


Figure 4. **Scaling Network Width for Different Baseline Networks**. Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline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

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

-

- Effnet은 위의 수식처럼 depth, width, resolution을 scaling합니다.
- Effnet b0는 $\alpha = 1, \beta = 1$을 초기값을 설정하고 있으며, 이에 대응되는 모델의 매개변수의 수는 450만 개입니다.

첫번째 EFFNET

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1 / width_coeff = 1

여러 번의 측정 끝에 LR이 0.002에서 좋은 성능을 낼 수 있음을 알 수 있었습니다.

➔ Trainset 정확도 : 93% valid set 정확도 91% (점수 0.9215)

두번째 effnet

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1 / width_coeff = 1
- 데이터 증강 과정에서 색 변환과 상하 반전을 제거하고 random rotate의 각도를 45도에서 30도로 수정하고, valid data set의 비율을 전체 데이터의 10%에서 5%로 줄임으로써 더 일반화되도록 하였습니다.
- Trainset 정확도 : 94% valid set 정확도 91% (점수 0.95400)

3번째 effnet

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1.06 / width_coeff = 1
- 성능 향상이 뚜렷하지 않은 부분에 대하여 lr 감소 구간을 추가로 설정함으로써 더 좋은 결과를 얻을 수 있었습니다.
- Depth_coeff를 1에서 1.06으로 증가시킴으로써 계산되는 매개변수를 490만개로 만들었습니다.
- Trainset 정확도 : 95% valid set 정확도 92% (점수 0.95450)

4번째 effnet

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.0002 / depth_coeff = 1.06 / width_coeff = 1
- 학습률 스케줄러를 ReduceLROnPlateau / CyclicLR 등 다양한 방법을 시도 하였지만 이전의 결과보다 더 좋은 결과를 얻지 못하였습니다.

5번째 EFFNET

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.003 / depth_coeff = 1.06 / width_coeff = 1
- 초기 학습률을 이전보다 높게 설정한 후 감소되는 구간을 2구간 더 추가함으로써 이전보다 더 좋은 성능을 얻을 수 있었습니다.
- Trainset 정확도 : 95% valid set 정확도 92% (점수 0.96150)

6번째 EFFNET

- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.003 / depth_coeff = 1.06 / width_coeff = 1
- 테스트에 사용되는 모델에 Dropout의 값을 제거하고 test set을 transform하는 과정을 tarin set 과 동일하게 적용함으로써 더 높은 결과를 얻을 수 있었습니다.
- Trainset 정확도 : 96% valid set 정확도 91% (점수 0.96150)

7번째 EFFNET

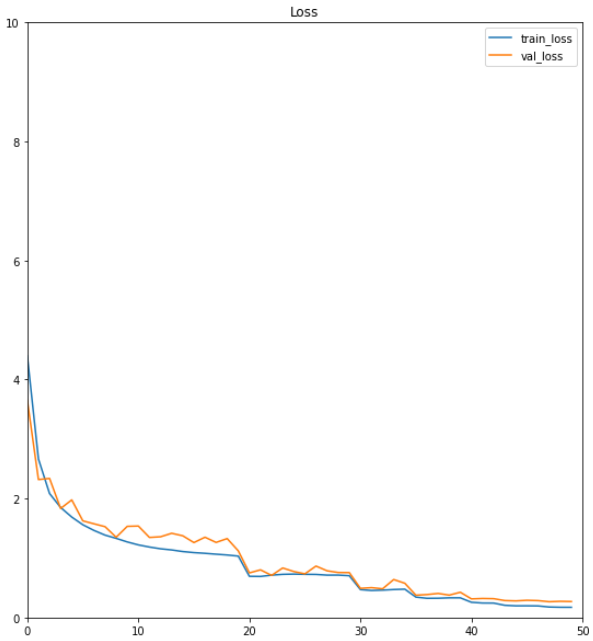
- 옵티마이저 : ADAM / BATCH SIZE : 16 / LR : 0.003 / depth_coeff = 1.06 / width_coeff = 1
- Training set보다 낮은 valid set의 정확도, 오버피팅의 문제를 감소시키기 위하여 droprate를 0.2 에서 0.5로 증가시켰습니다.
- Trainset 정확도 : 96% valid set 정확도 93% (점수 0.96450)

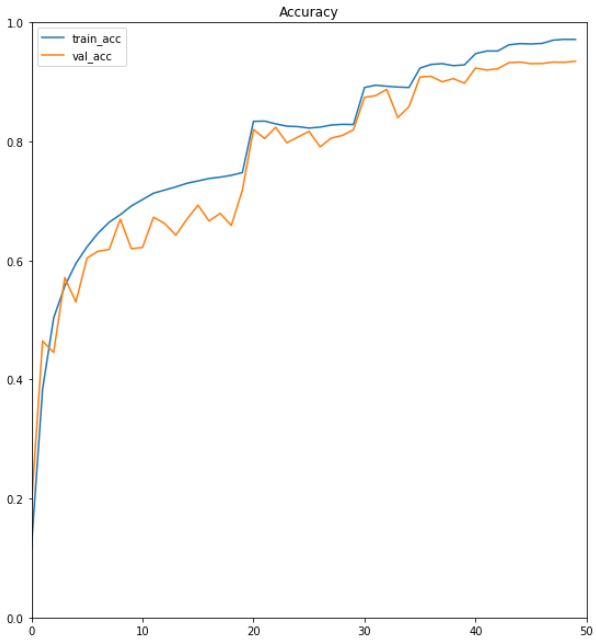
8번째 EFFNET

- 옵티마이저 : NAdam / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1.06 / width_coeff = 1
- 옵티마이저를 NAdam으로 교체한 후 train set에서 더 높은 정확도를 보였지만 오버피팅 문제가 관찰되었습니다.
- Trainset 정확도 : 98% valid set 정확도 91% (점수 0.97250)

9번째 EFFNET

- 옵티마이저 : SGD / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1.06 / width_coeff = 1
- 데이터 증강 과정에서 이미지 회전을 30도에서 15도로 낮추었으며, epoch 43 / 45/ 47 등 훈련 마지막에서 lr을 세분화하여 바꿔주었습니다.
- 또한 drop connect rate를 default 0.2에서 0.5로 증가시키고 테스트셋의 전처리과정에서 dropout rate와 drop connect rate를 모두 0으로 설정함으로써 더 좋은 성능을 보여주었습니다.
- 또한 일반화 성능을 높이기 위해서 옵티마이저를 NAdam에서 SGD로 교체하였습니다.
- Trainset 정확도 : 97% valid set 정확도 93% (점수 0.98500)





10번째 EFFNET

- 옵티마이저 : NADAM / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1.06 / width_coeff = 1
- 오버 피팅 문제를 해결하기 위해 Dropout rate, drop connect rate를 0.8로 올리고 훈련을 진행하였습니다
- Train set은 더 낮은 정확도가 나왔지만 valid set에서는 더 높은 정확도를 도출했습니다.
- 컴페티션 점수는 조금 더 낮은 점수를 받았습니다
- Trainset 정확도 : 96% valid set 정확도 94% (점수 0.97700)

11번째 EFFNET

- 옵티마이저 : SGD / BATCH SIZE : 16 / LR : 0.002 / depth_coeff = 1.06 / width_coeff = 1
- 가장 높은 점수를 받았던 모델에서 dropout_rate를 0.5로 설정해두고 drop connect rate를 0.2로 설정하였습니다.
- Trainset 정확도 : 95% valid set 정확도 90% (점수 0.95700)