실전 기계학습
기말 컴피티션

TEAM 2
서보민(팀장) / 김성주 / 최인열

목차

모델 소개

모델 선정

Dataset Notebooks

BIRDS 400

All | Your Work | Shared With You | Bookmarks

400 bird species CNN by Pytorch
Updated 1d ago
0 comments · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

Birds _400_Image_Classification_Densenet
Notebook copied with edits from Harsh Solanki · Updated 13d ago
0 comments · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

ClassificationUsingEfficientNetB0
Updated 1mo ago
0 comments · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

Birds _400_Image_Classification_Densenet
Updated 2mo ago
0 comments · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

MAD4 final
Notebook copied with edits from a private notebook · Updated 2mo ago
0 comments · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

Birds 400 PyTorch | 97%
Updated 3mo ago
1 comment · BIRDS 400 - SPECIES IMAGE CLASSIFICATION

400 Bird Species F1 score=99.4%

Images

ImageNet

Edit

Introduced by Jia Deng et al. in ImageNet: A large-scale hierarchical image database

The **ImageNet** dataset contains 14,197,122 annotated images according to the WordNet hierarchy. Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection. The publicly released dataset contains a set of manually annotated training images. A set of test images is also released, with the manual annotations withheld. ILSVRC annotations fall into one of two categories: (1) image-level annotation of a binary label for the presence or absence of an object class in the image, e.g., "there are cars in this image" but "there are no tigers," and (2) object-level annotation of a tight bounding box and class label around an object instance in the image, e.g., "there is a screwdriver centered at position (20,25) with width of 50 pixels and height of 30 pixels". The ImageNet project does not own the copyright of the images, therefore only thumbnails and URLs of images are provided.

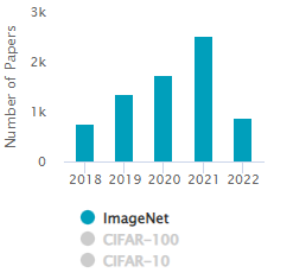
- Total number of non-empty WordNet synsets: 21841
- Total number of images: 14197122
- Number of images with bounding box annotations: 1,034,908
- Number of synsets with SIFT features: 1000
- Number of images with SIFT features: 1.2 million

Source: ImageNet Large Scale Visual Recognition Challenge

Homepage

Source: https://cs.stanford.edu/people/karpa...

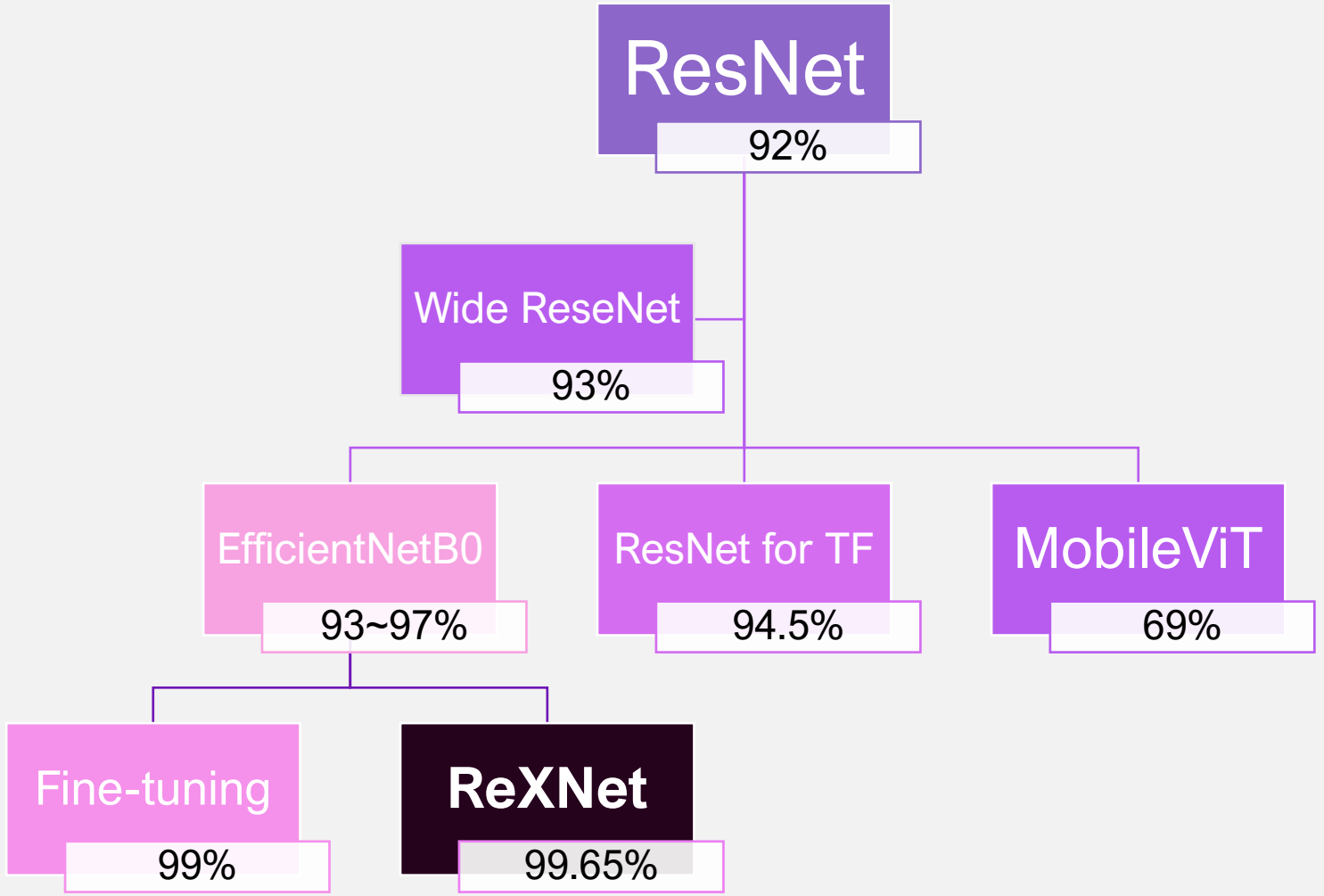
Usage



Part 1,

모델 변천 과정

Model Select



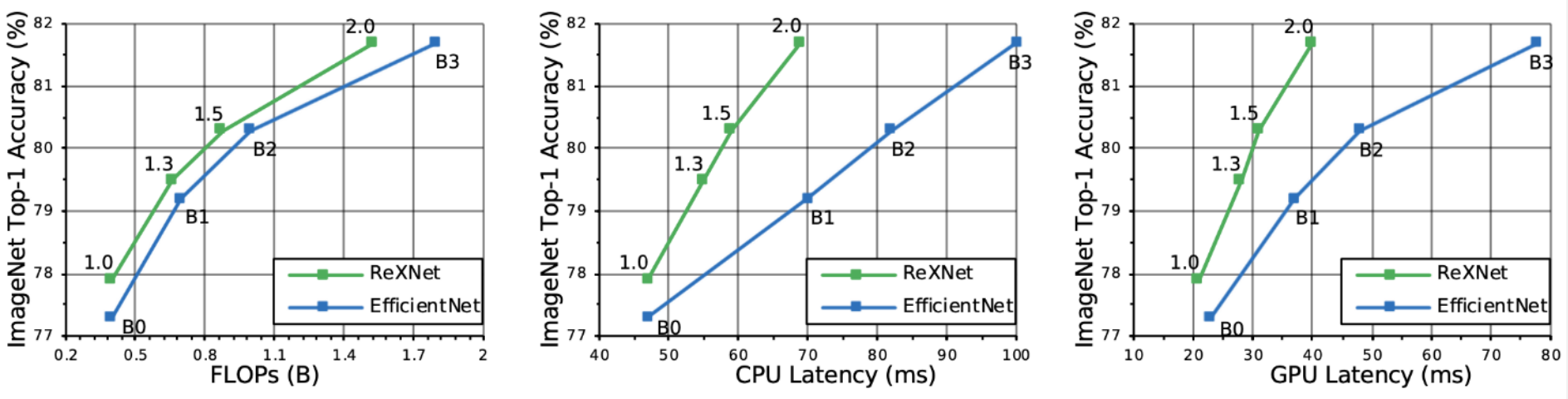
모델 순위

1	1위. ReXNet
2	2위. EfficientB0
3	3위. Resnet / WRN

모델별 학습 과정

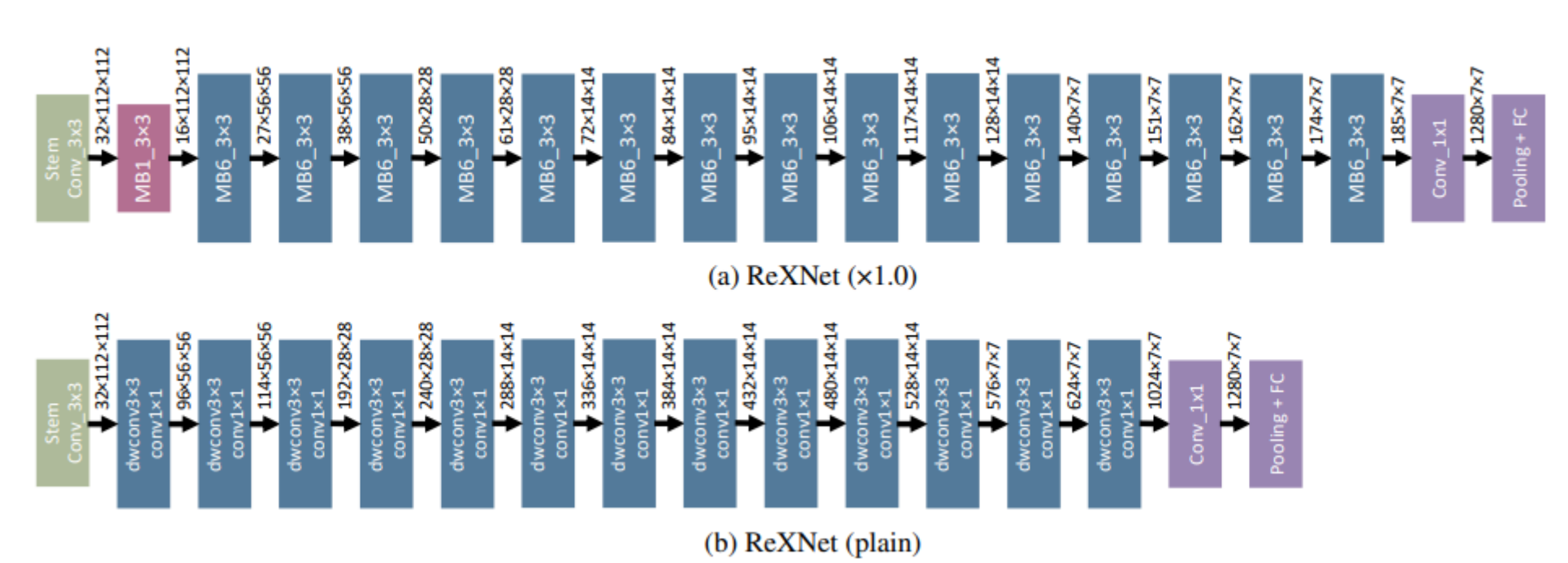
1위. ReXNet

기본 모델 선택: RexNet



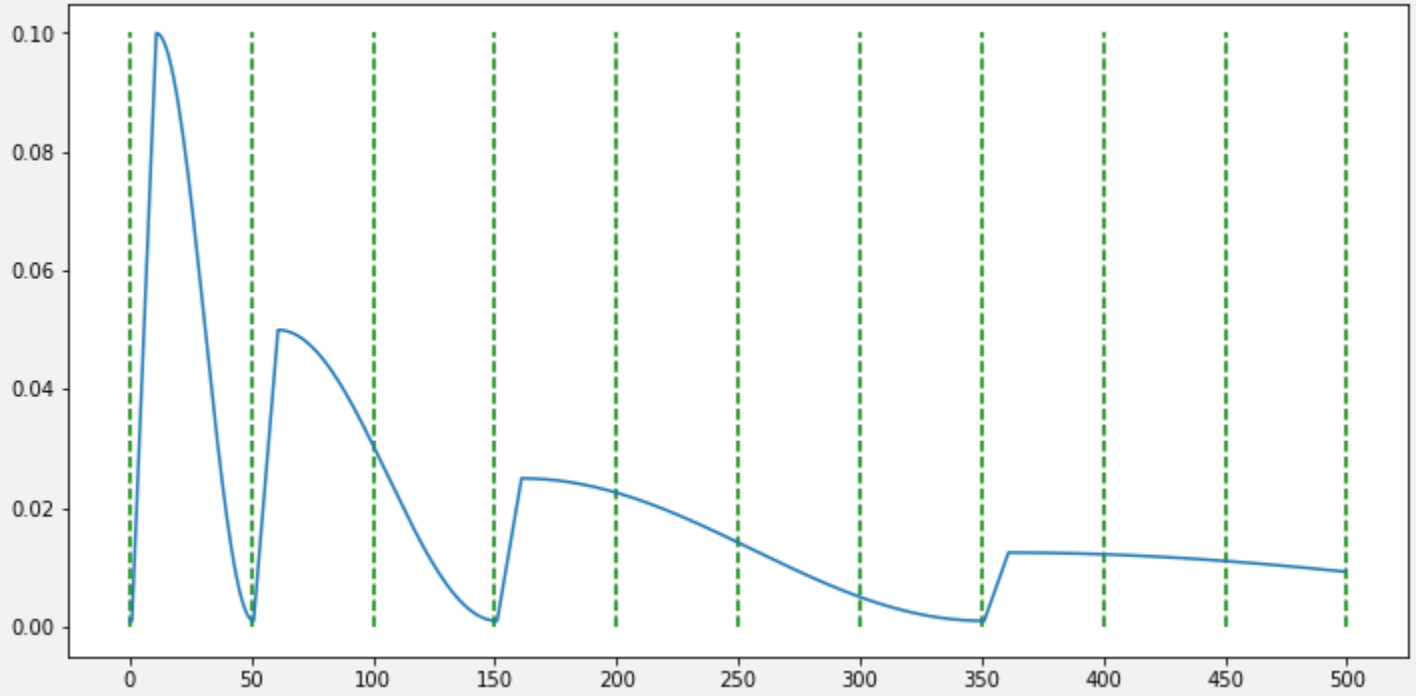
- 대부분의 팀이 ResNet이나 EfficientB0를 이용할 것이라 생각
- ImageNet SOTA 중 파라미터가 5M 미만인 것 중 가장 성능이 높음

기본 모델 선택: RexNet



- **Representation Bottleneck 구조**

기본 모델 선택: RexNet



- CosineAnnealingWarmUpRestarts 활용
- T_0=9, T_mult=2, eta_max=0.01, T_up=2, gamma=0.4

기본 모델 선택: RexNet

	albumentations 0.4.5	imgaug 0.4.0	torchvision (Pillow-SIMD backend) 0.5.0	keras 2.3.1	augmentor 0.2.8	solt 0.1.8
HorizontalFlip	**3066**	1544	1652	874	1658	853
VerticalFlip	**4159**	2014	1427	4147	1448	3788
Rotate	**417**	327	160	29	60	113
ShiftScaleRotate	**703**	471	144	30	-	-
Brightness	**2210**	997	397	210	396	2058
Contrast	**2208**	1023	330	-	331	2059
BrightnessContrast	**2199**	582	190	-	190	1051
ShiftRGB	**2215**	998	-	378	-	-
ShiftHSV	**381**	241	59	-	-	128
Gamma	**2340**	-	686	-	-	951
Grayscale	**4961**	372	735	-	1423	4286
RandomCrop64	**157376**	2560	41448	-	36036	35454
PadToSize512	**2833**	-	478	-	-	2629
Resize512	**952**	595	885	-	873	881
RandomSizedCrop_64_512	**3128**	881	1295	-	1254	2678
Equalize	**760**	399	-	-	666	-
Multiply	**2184**	1059	-	-	-	-
MultiplyElementwise	124	**197**	-	-	-	-

- **Albumentation 활용**

모델 구조 및 학습 과정

Model Structure and Learning Process

학습 결과

Train Score : 95%
Validation Score : 90%
Kaggle Score : 0.909

모델 구조 및 학습 과정

Model Structure and Learning Process

실패 요인 분석

1. 어그멘테이션을 너무 적게 시행하였음.
2. ReXNet 최적의 파라미터를 사용하지 않았음.
3. Epoch 수가 너무 적어 Cosine 스케쥴러의 효율이 떨어짐
4. Pt 저장을 소홀히 하여 이어서 진행할 수 없었음

모델 구조 및 학습 과정

Model Structure and Learning Process

2차 시도

0.65 대에서 수렴. Local Optima 발생
부족한 시간과 당시 팀 모델 정확도가 0.95를 넘었기 때문에 포기

Part 2,

모델 구조 및 학습 과정

Model Structure and Learning Process

3차 시도

Opt : SGD, Batch Size : 16, Lr : 0.002, Width_mult = 1.12, dropout_ratio = 0.5

마감 당일 서보민 학우의 EfficieintNetB0(99%) 코드에
"모델만" ReXNet으로 바꾸어서 진행

Width_mult 조정을 통해 파라미터 수 4.97M로 극한까지 끌어올림
Dropout을 통해 과적합 해소

Part 2,

모델 구조 및 학습 과정

Model Structure and Learning Process

3차 시도

Opt : SGD, Batch Size : 16, Lr : 0.002, Width_mult = 1.12, dropout_ratio = 0.5

기존 30시간이었던 총 학습시간이 5시간으로 감소
기존 사용했던 Albumentation이 오히려 더 속도를 저하시켰음

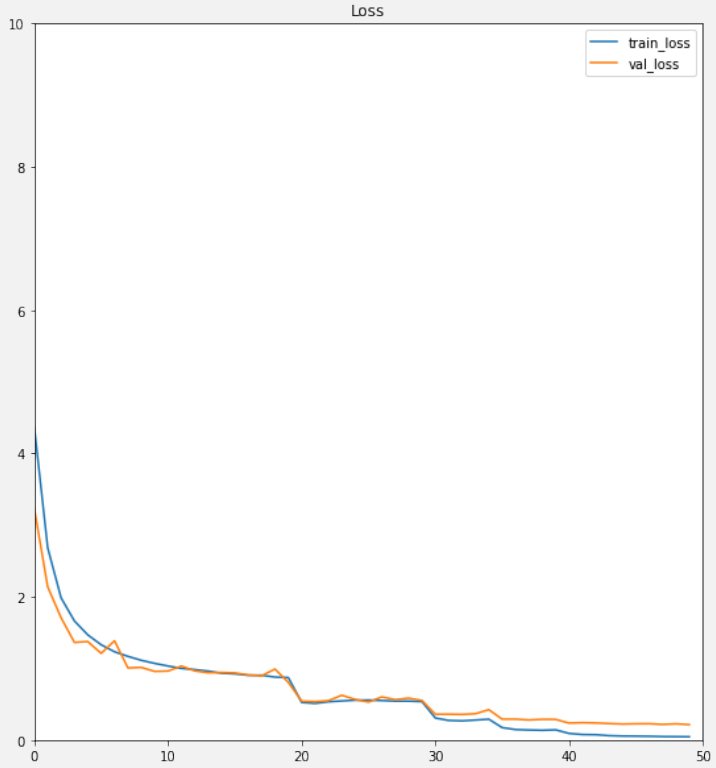
데이터셋이 작고, 클래스를 반복 호출해서 이미지를 읽어야 하다보니 시간적으로 손해였음

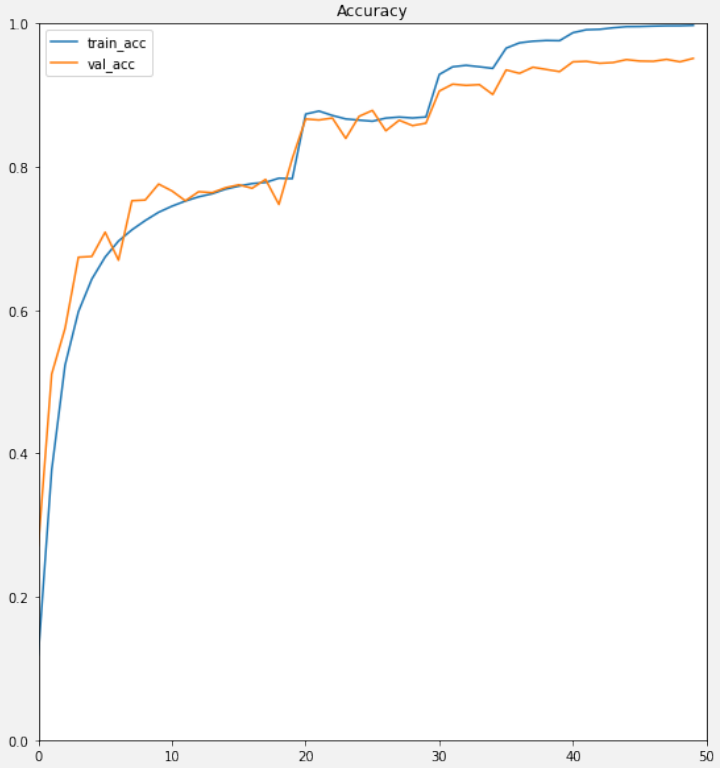
Part 2,

모델 구조 및 학습 과정

Model Structure and Learning Process

학습 결과





Train Score : 99.71%
Validation Score : 95.1%
Kaggle Score : 0.9965

2위. EfficientB0

기본 모델 선택: EfficientB0

- WRN이 93~94%에서 더 이상 좋은 성능을 내지 못했음
- 이번 데이터셋과 유사한 CIFAR100에서 가장 좋은 성능을 낸 모델
- 매개변수 5M 미만의 조건을 만족하기 위하여 B0 모델을 선택

기본 모델 선택: EfficientB0

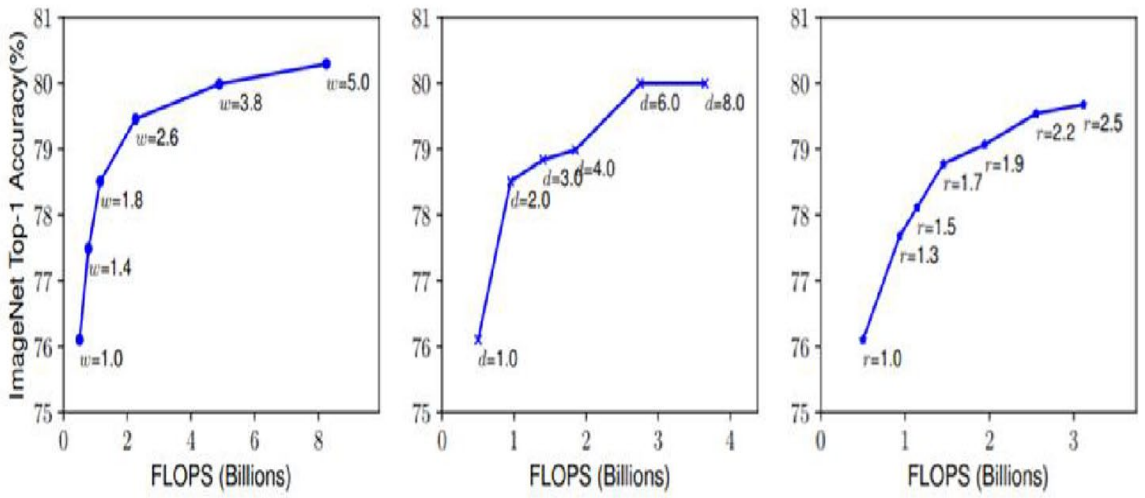


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

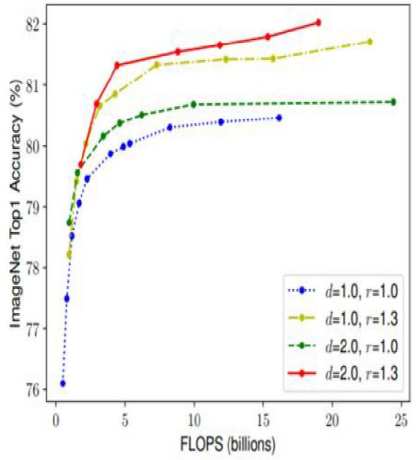


Figure 4. **Scaling Network Width for Different Baseline Networks.** Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline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

$$\text{depth: } d = \alpha^{\phi}$$

$$\text{width: } w = \beta^{\phi}$$

$$\text{resolution: } r = \gamma^{\phi}$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

- 이론적 배경, Depth, Width, Resolution의 상관관계

학습 진행 시도 2회차

- 데이터 증강에서
  색 변환, 상하 반전 제거, Rotate의 각도 범위를 30도로 축소
  Validation data 비율 5%로 수정

- Opt : Adam / Batch Size : 16 / Lr : 0.002
  Depth_coeff = 1 / Width_coeff = 1

- Train Score : 94% / Validation Score : 91%
  Kaggle Score : 0.954%

학습 진행 시도 3회차

- 성능 향상이 뚜렷하지 않은 구간의 Lr 감소 스케쥴러 추가 설정
  Depth_coeff를 1.06으로 증가시켜 매개변수 4.9M으로 증가

- Opt : Adam / Batch Size : 16 / Lr : 0.002
  Depth_coeff = 1.06 / Width_coeff = 1

- Train Score : 95% / Validation Score : 92%
  Kaggle Score : 0.9545%

학습 진행 시도 4 ~ 8회차

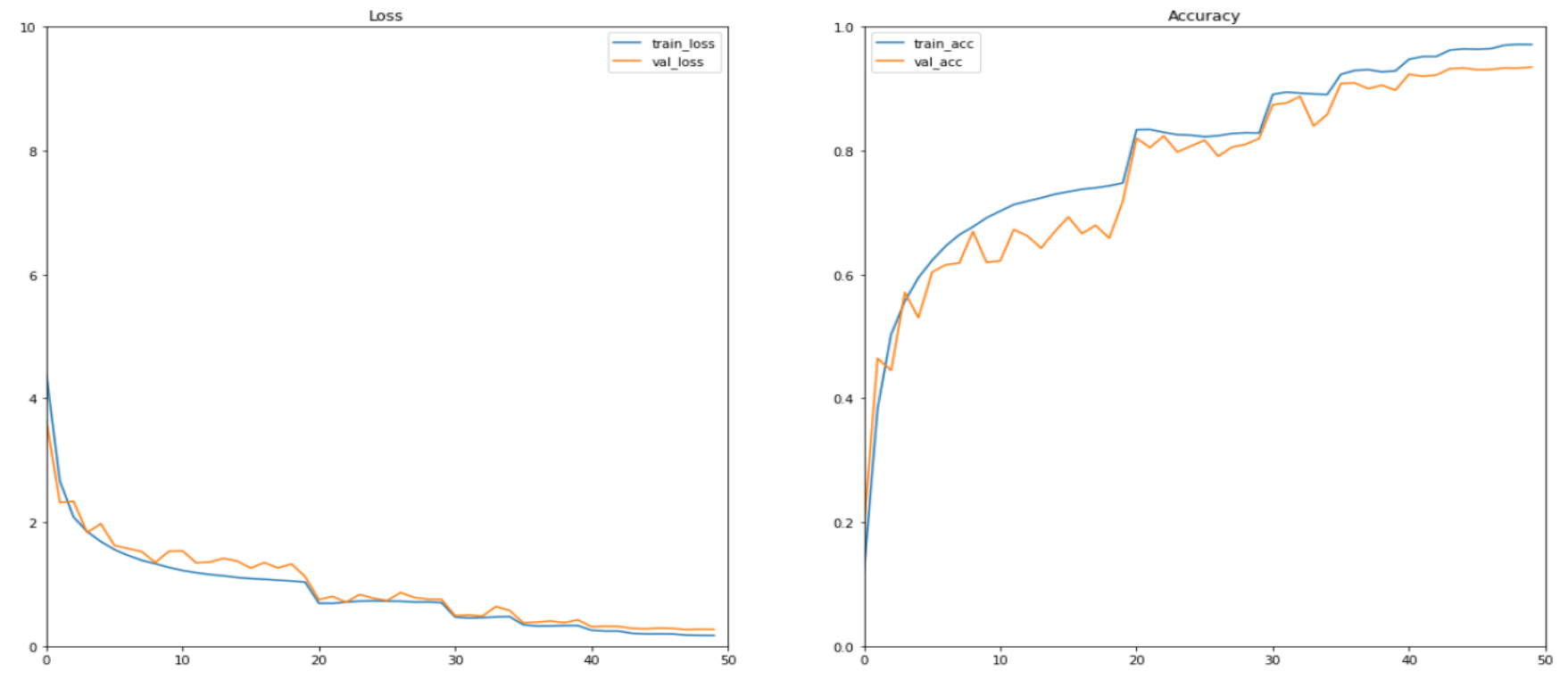
- 초기 학습률을 이전보다 높게 설정한 후 감소되는 구간을 2구간 더 설정
- 오버피팅 문제를 해결하기 위해 Droprate를 0.2에서 0.5로 증가
- Opt를 Nadam으로 교체한 후 더 높은 정확도를 보였으나 오버피팅 문제 발생

- Train Score : 98% / Validation Score : 91%
  Kaggle Score : 0.9725%

학습 진행 시도 9 ~ 11회차

1. 데이터 증강에서 이미지 회전을 30도에서 15도로 감소
2. Drop Rate를 모두 0.8로 증가
3. Epoch 43, 45, 47에서 lr을 세부적으로 조정
4. Drop Connect rate를 0.2에서 0.5로 증가
5. Test Set에서 Drop Rate 0.2에서 0으로 설정
6. 일반화 성능을 위해 NAdam에서 SGD로 교체

학습 진행 시도 9 ~ 11회차



- **Train Score : 97% / Validation Score : 93%**
  Kaggle Score : 0.9850

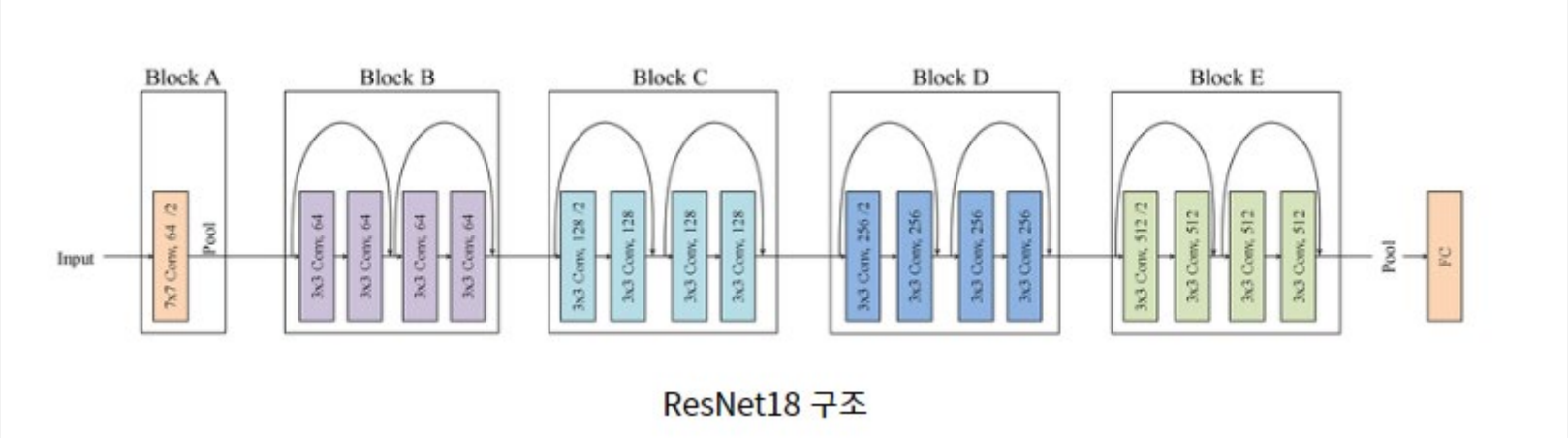
학습 진행 시도 12회차

- 가장 높은 점수를 받았던 모델에서 랜덤하게 추출
  Dropout_rate : 0.18
  Dropout_connect : 0.24
  Batch size : 64

- Train Score : 97% / Validation Score : 93%
  Kaggle Score : 0.9900

3위. Resnet

기본 모델 선택

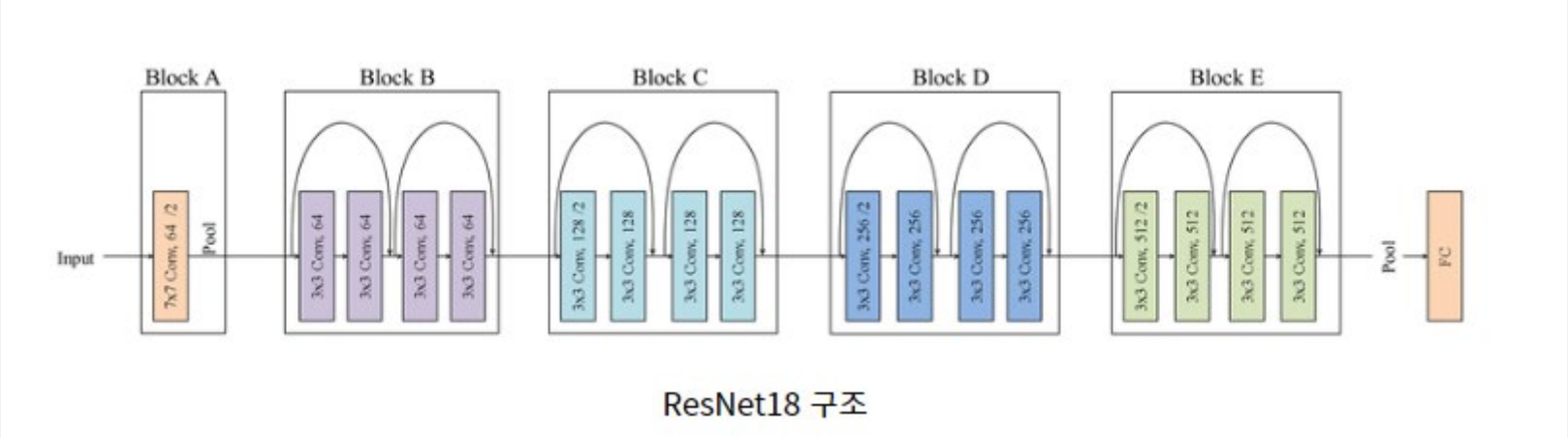


ResNet18 구조

Resnet-18 모델을 기본 모델로 사용

기존 Resnet-18은 매크로 블록이 반복될 때마다
filter의 개수가 2배씩 증가해 파라미터의 개수가 1000만개

기본 모델 커스텀

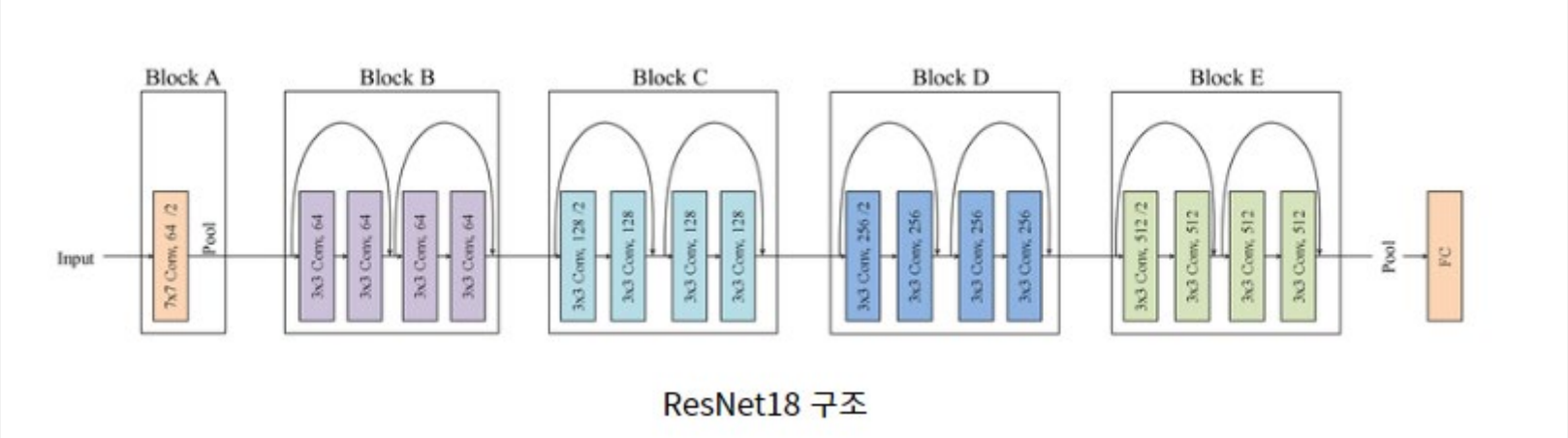


ResNet18 구조

Resnet 모델은 깊이있는 신경망이 가진 문제를 residual 블록을 통해 해결한다는 컨셉

따라서 깊이는 유지하고,
Filter의 개수를 줄여서 모델을 custom

기본 모델 커스텀

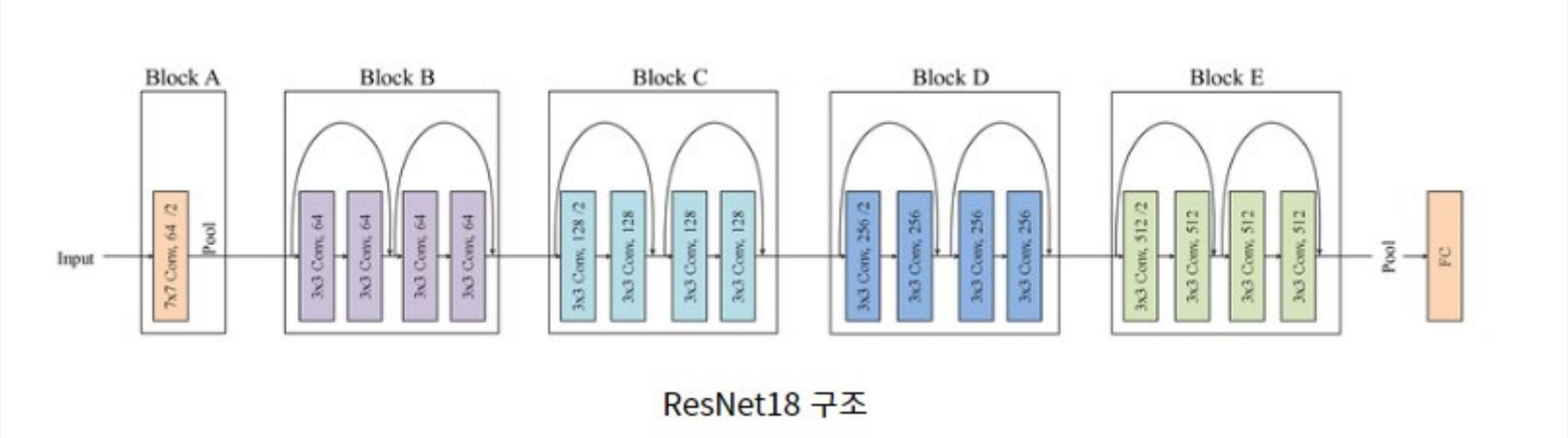


ResNet18 구조

매크로 블록이 반복될 때마다 filter의 개수를 1.5배씩 증가

약 490만개의 파라미터를 가진 모델 생성

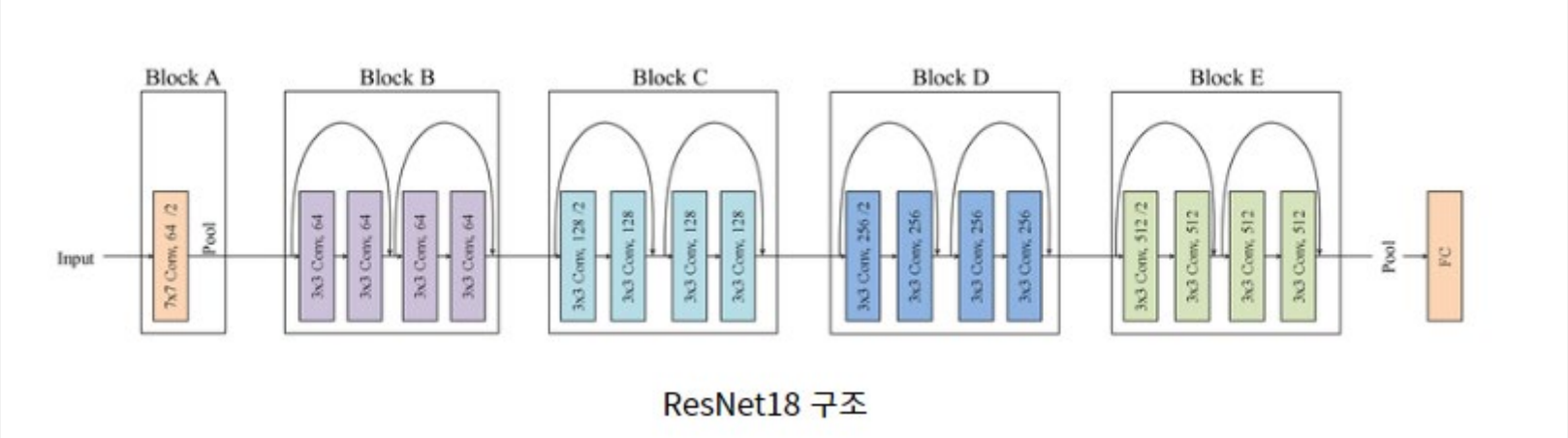
1. 첫 번째 트레이닝 전략



ResNet18 구조

트레이닝 데이터를 가지고 L2 정규화를 적용해서 훈련을 한 결과
트레이닝 정확도는 20 에포크 이내에서 0.99까지 빠르게 상승
밸리데이션 정확도는 0.7에서 더 이상 상승하지 않았음

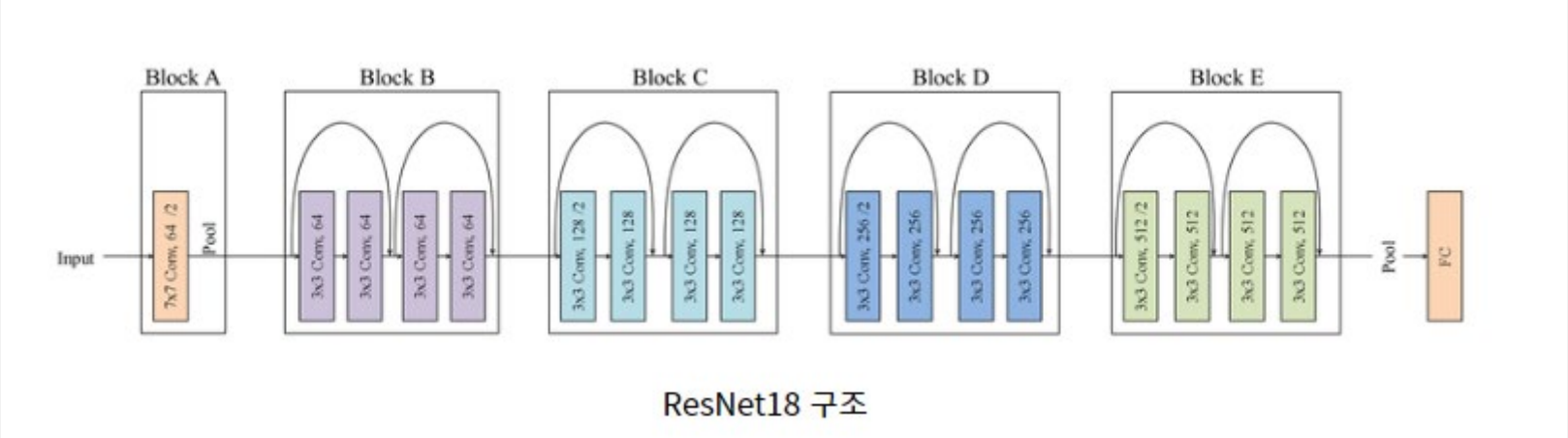
2. 두 번째 트레이닝 전략



ResNet18 구조

러닝 레이트나 최적화 알고리즘을 변화시키지 않고 그대로 사용
대신 데이터의 학습 순서를 조절해서 정확도 향상을 기대
다른 변인을 모두 통제하고 데이터의 학습 순서만 제어

2. 두 번째 트레이닝 전략

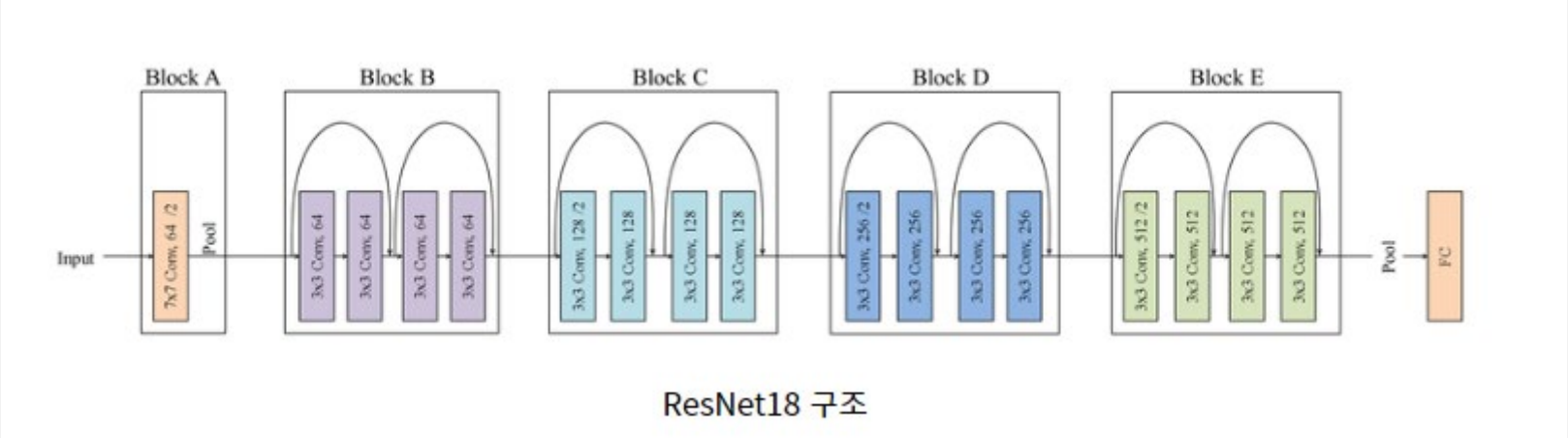


ResNet18 구조

- 정규화만으로는 오버피팅을 막을 수 없었기 때문에
  데이터 어그멘테이션을 통해서 트레이닝

- 어그멘테이션에 걸리는 시간의 단점을 해결하기 위해
  첫 10 에포크는 어그멘테이션 없이 트레이닝

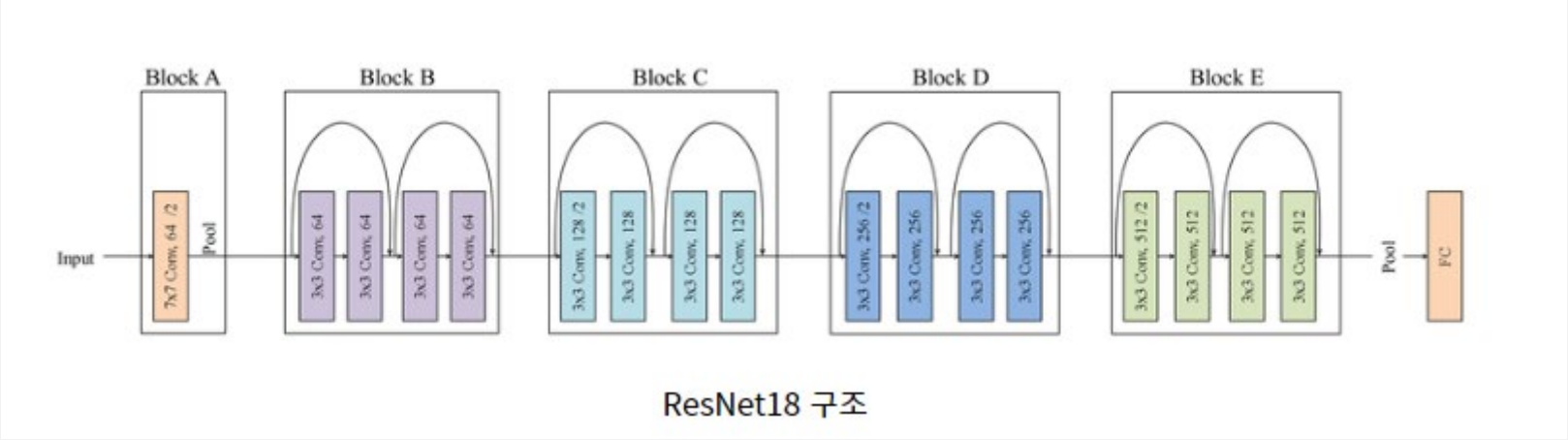
2. 두 번째 트레이닝 전략



ResNet18 구조

약 5분 이내에 10 에포크로
트레이닝 정확도 90%벨리데이션 정확도 60%에 도달

2. 두 번째 트레이닝 전략



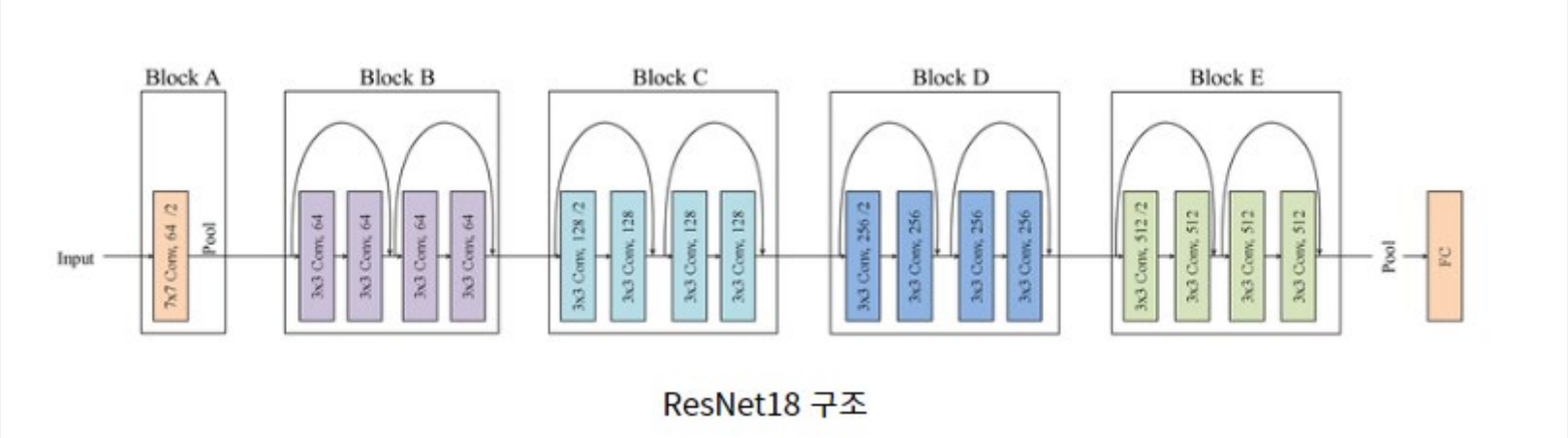
ResNet18 구조

문제는 남은 40 에포크 이내에 일반 트레이닝과 어그멘테이션 트레이닝을

어떤 비율로 번갈아가면서 시행해야

50 에포크 내에 최고의 결과를 얻을 수 있을지 찾아내는 것

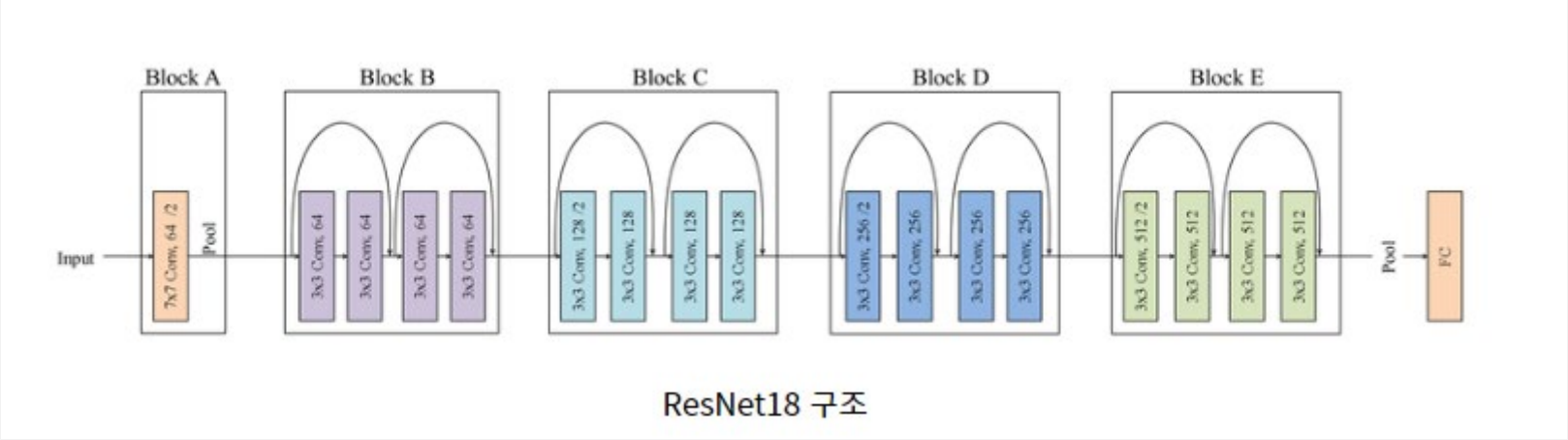
2. 두 번째 트레이닝 전략



ResNet18 구조

시간 절약을 위해 절반의 데이터를 가지고 100 에포크를 진행했을 때 트레이닝 정확도 및 벨리데이션 정확도 모두 98.5%가 달성되는 것을 확인

3. 세 번째 트레이닝 전략

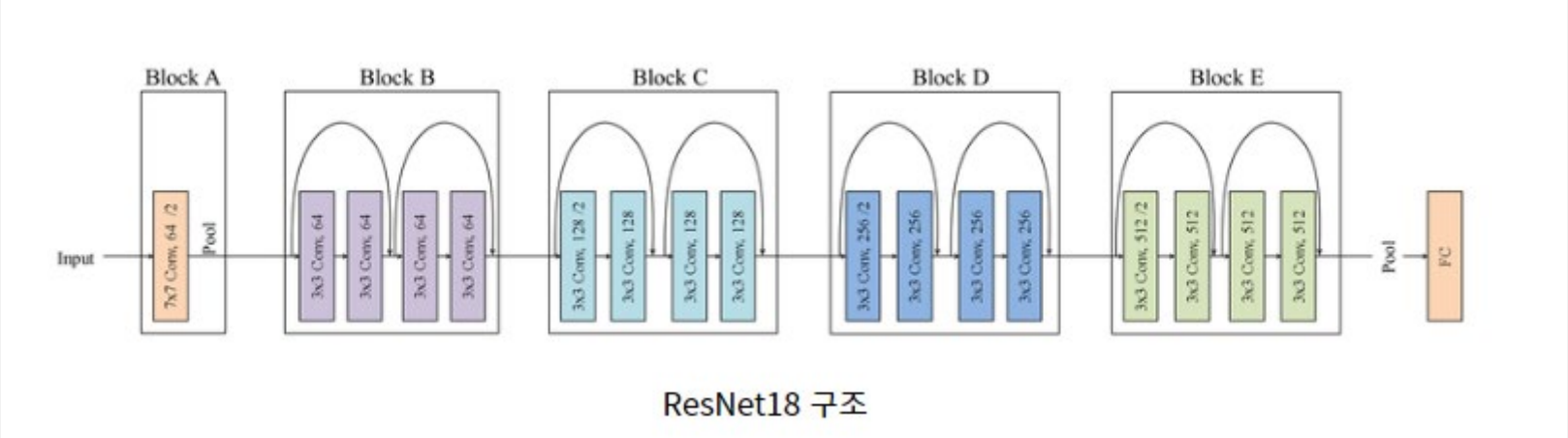


ResNet18 구조

초기 어그멘테이션은 상하 반전부터 사진 일그러짐까지 다양하게 생성

주어진 제한 조건 내에서 모든 변화를 다 파악하는 모델을 만들기에는 모델의 수용량이 낮다는 것을 파악

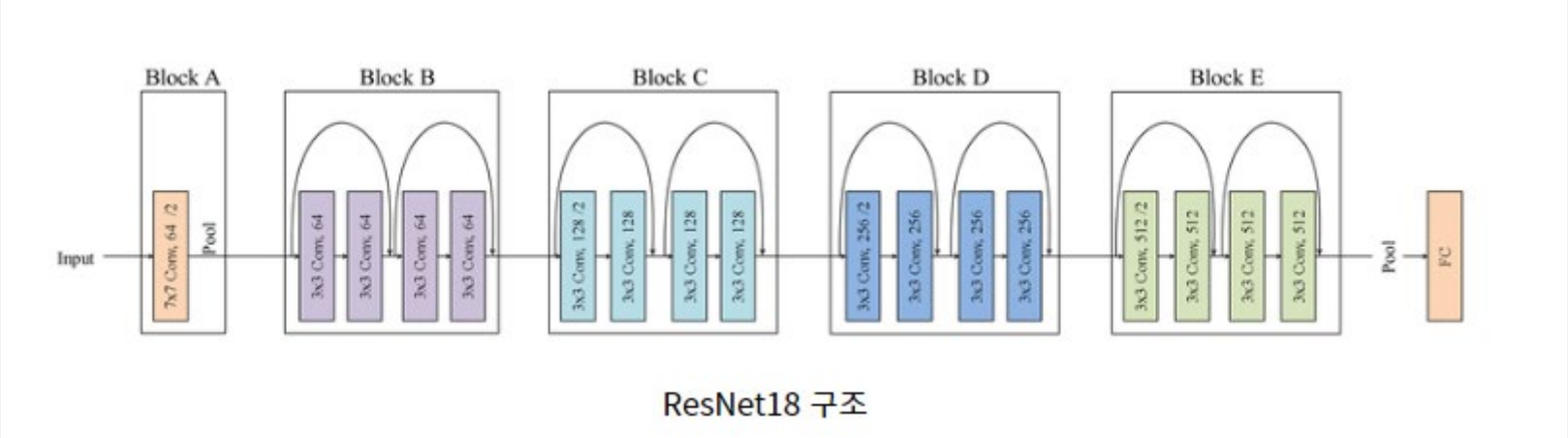
3. 세 번째 트레이닝 전략



ResNet18 구조

밸리데이션 셋과 테스트 셋에 주어진 대부분의 사진에 상하 반전이 존재하지 않는다는 사실을 알게 되어 어그멘테이션에서 상하 반전을 제거
밸리데이션 정확도가 상승

3. 세 번째 트레이닝 전략

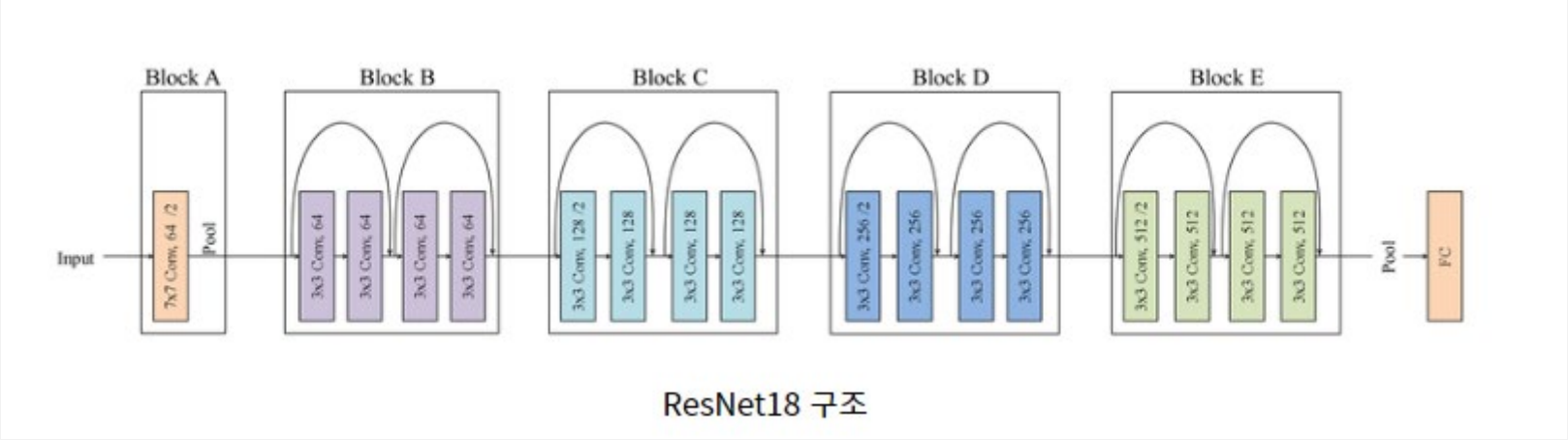


ResNet18 구조

이는 주어진 조건 하에서 최대 정확도를 얻으려면
어그멘테이션의 변화 정도를 감소시켜야 한다는 뜻

상하 반전을 제거하고 이동 및 축소도 15% 내의 작은 범위로 축소

3. 세 번째 트레이닝 전략

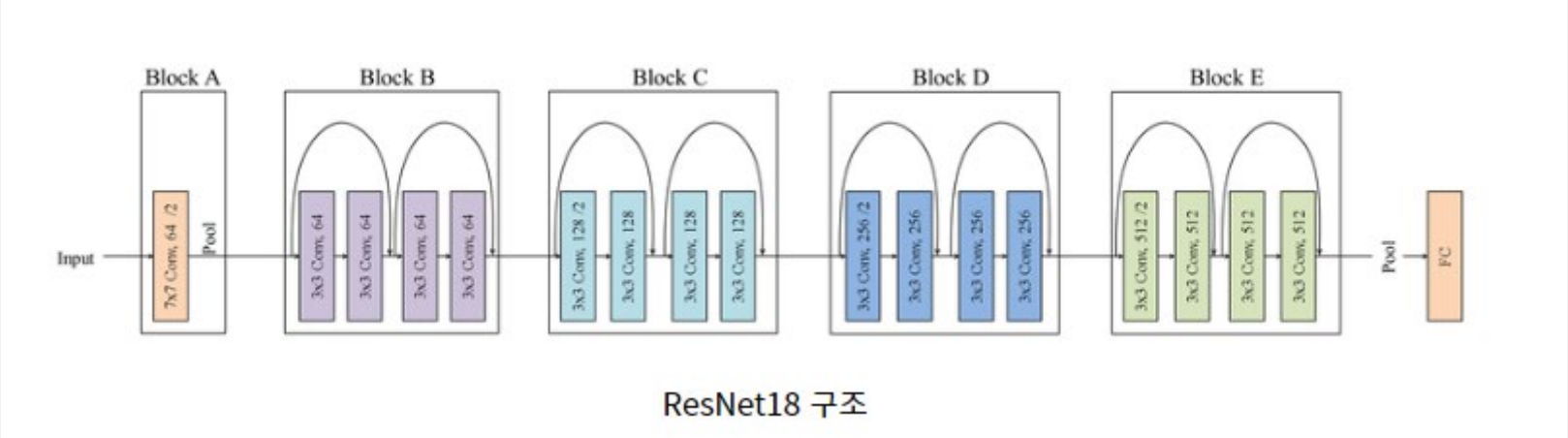


ResNet18 구조

트레이닝 셋 acc 96.5%
밸리데이션 셋 acc 94.5%
테스트 셋 acc 94.5%

GPU 사용시간 제한으로 최적의 비율을 찾는 것에는 실패

아쉬운 점



ResNet18 구조

GPU 사용 제한으로
10 에포크는 일반 트레이닝
40 에포크는 어그멘테이션 트레이닝만 진행

서로 다른 방식의 훈련을 섞어가면서 진행해보지 못함

4위. WRN

기본 모델 선택: WRN

- 가이드 코드 Resnet의 매개변수 및 데이터 증강 과정을 수정
- 56%의 정확도 획득
- 더 높은 정확도를 위해 Wide Resnet을 고려

기본 모델 선택: WRN

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 × 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 $B(3,3)$.

- **WRN 모델의 깊이**

기본 모델 선택: WRN

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 × 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 $B(3,3)$.

- **WRN 모델의 매개변수의 수와 정확도의 상관 관계**

기본 모델 선택: WRN

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

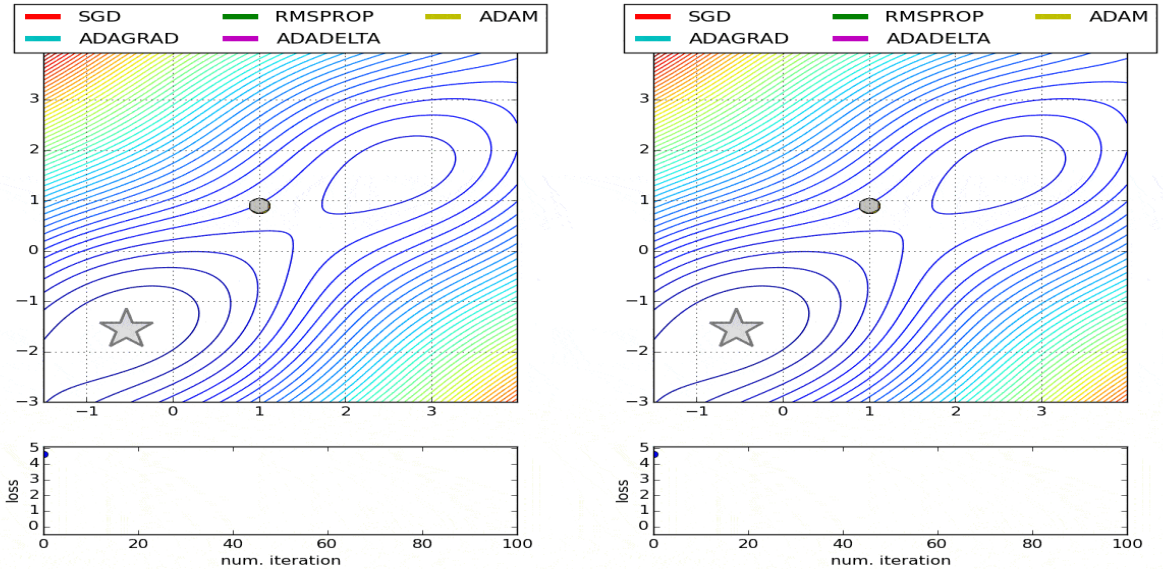
- 옵티마이저의 경우 논문을 참고
- Adam, SGD를 후보로 두고 실험 결과에 따라 SGD를 선택

학습 진행 시도 5회차

```python
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"]
```

- 4번째 학습의 depth 16, width 4 고정 후 LR을 조절
- LR: 0.2 / Opt: SGD / Batch size: 64
- Train Score: 95% / Validation Score: 92%
- Kaggle Score: 0.9365

학습 진행 시도 6회차

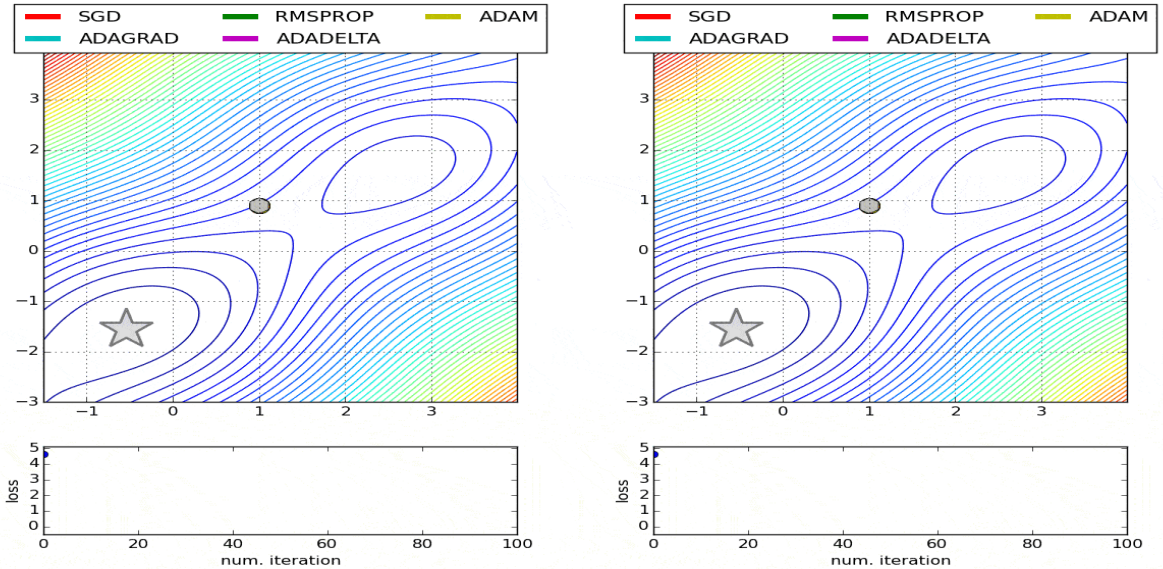


Test AUC

Batch size	Adam LR = 0.0001	Adam LR = 0.001
16 →	0.9677 Best 조합 1	0.9144 Worst 조합 1
32	0.9636	0.9332
64	0.9616	0.9381
128	0.9567	0.9432
256 →	0.9585 Worst 조합 2	0.9652 Best 조합 2

- 이미지 분류에서 Adam의 성능이 더 좋다는 레퍼런스
- 적은 Batch size와 낮은 LR의 조합이 성능을 개선한다는 통계 자료

학습 진행 시도 6회차



Test AUC

Batch size	Adam LR = 0.0001	Adam LR = 0.001
16 →	0.9677 Best 조합 1	0.9144 Worst 조합 1
32	0.9636	0.9332
64	0.9616	0.9381
128	0.9567	0.9432
256 →	0.9585 Worst 조합 2	0.9652 Best 조합 2

- 두 자료를 바탕으로 파라미터를 변경
- LR: 0.2 / Opt: Adam / Batch size: 16
- Train Score : 95% / Validation Score : 93%
- Kaggle Score : 0.937

학습 진행 시도 N회차

- Lr을 0.2 ~ 0.001 사이에서 다양한 값으로 바꾸어 가며 시도
- Nadam, Rmsprop 등 Opt 교체
- 다른 유저에 의해(Peter Kim)이 구현된 WRN 사용
- Dvdda54와 Peter Kim의 WRN을 어셈블

역할 분담 및 레퍼런스

역할분담

TEAM Leader
EfficientNet 모델(99%)
팀 내 최다 제출
다양한 Case 실험 진행

서보민
(contribute 40%)

ResNet 모델(94.5%)
PPT 발표

김성주
(contribute 30%)

MobileViT 모델(69%)
ReXNet 모델(99.65%)
SOTA 모델 Fine-Tuning
PPT 제작

최인열
(contribute 30%)

레퍼런스

MobileViT
https://youtu.be/1vZ2wEsUdZc
https://arxiv.org/abs/2110.02178
https://bigwaveai.tistory.com/25

EfficientNet
https://greeksharifa.github.io/computer%20vision/2022/03/01/EfficientNet/
https://github.com/google research/sam
https://dacon.io/en/forum/406054

ResNet
https://github.com/davda54/sam/tree/main/example
https://cumulu s.tistory.com/36
https://proceedings.neurips.cc/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf
https://towardsdatascience.com/optimizers for training neural network 59450d71caf6
https://github.com/PeterKim1/paper_code_review

ReXNet
https://github.com/clovaai/rexnet
https://skyil.tistory.com/92
https://gaussian37.github.io/dl-pytorch-lr_scheduler/

감사합니다