식물병리학
Multiclass Classification

팀원 : 고석주, 남정우, 강동엽

23.01.13

목 차

1 주제설명
2 EDA
3 데이터 전처리
4 모델선정
5 컴파일
6 모델 훈련
7 결과


01
주제 설명

Kaggle 경진대회문제

식물의 여러 사진을 보고 어떠한 질병에 인지
확인하는 문제

예측값이 여러개의 항목으로 이루어져있으며,
예측값의 확률을 계산하는 다중분류문제이다.

02

EDA

EDA

Data Explorer

823.79 MB

- images
- sample_submission.csv
- test.csv
- train.csv

```
train.head()
```

	image_id	healthy	multiple_diseases	rust	scab
379	Train_379	0	0	0	1
1556	Train_1556	1	0	0	0
448	Train_448	1	0	0	0
1202	Train_1202	0	0	0	1
1541	Train_1541	1	0	0	0

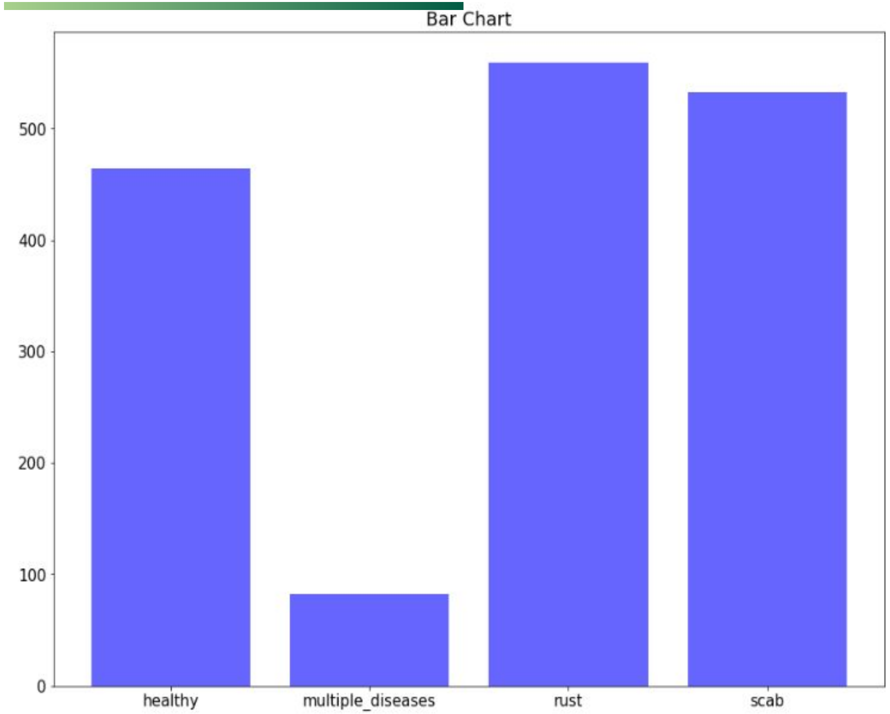
```
test.head()
```

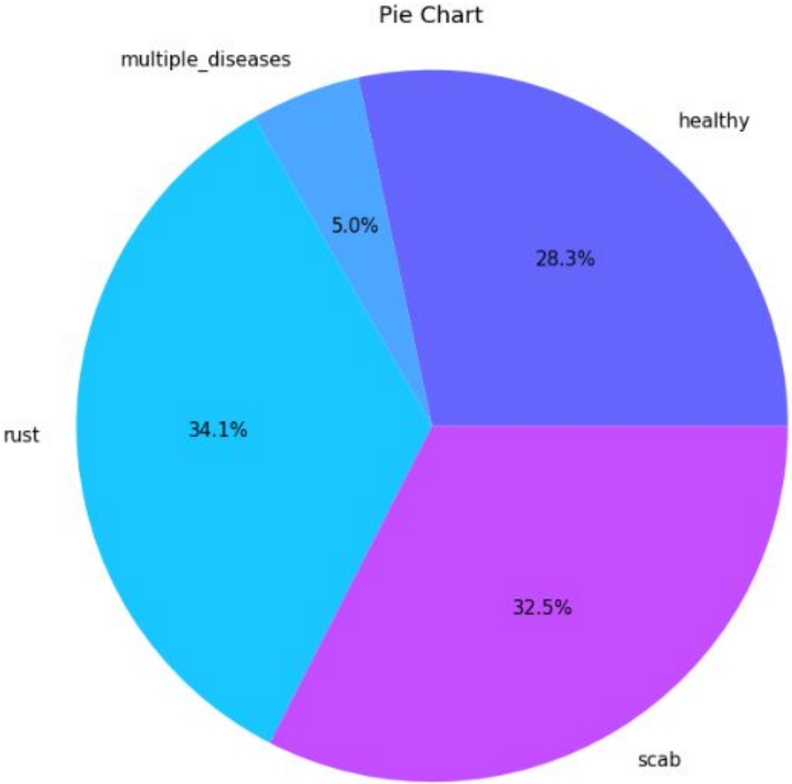
	image_id
0	Test_0
1	Test_1
2	Test_2
3	Test_3
4	Test_4

```
submission.head()
```

	image_id	healthy	multiple_diseases	rust	scab
0	Test_0	0.25	0.25	0.25	0.25
1	Test_1	0.25	0.25	0.25	0.25
2	Test_2	0.25	0.25	0.25	0.25
3	Test_3	0.25	0.25	0.25	0.25
4	Test_4	0.25	0.25	0.25	0.25

EDA

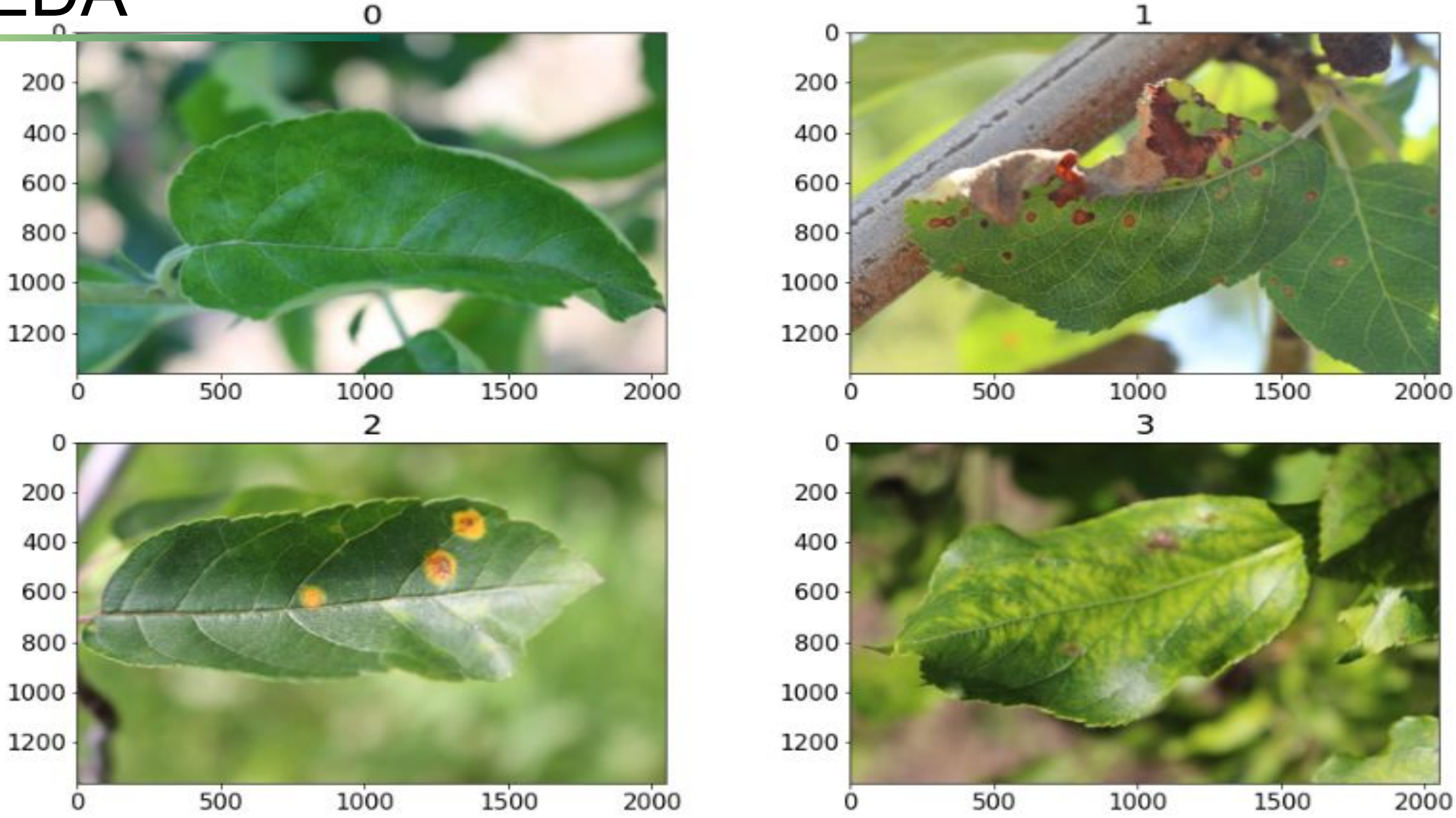




healthy : 건강함
multiple_diseases : 여러 질병
rust : 녹병균
scab : 붉은 곰팡이병

예측값이 4개로 하는 다중분류

EDA



03

전처리

전처리

Albumentation library

- Horizontal
- Vertical
- RandomBright
- Affine
- ShiftscaleRotate
- Normalize
- Blur, Sharpen, Emboss,

```python
import albumentations as A
from albumentations.pytorch import ToTensorV2
```

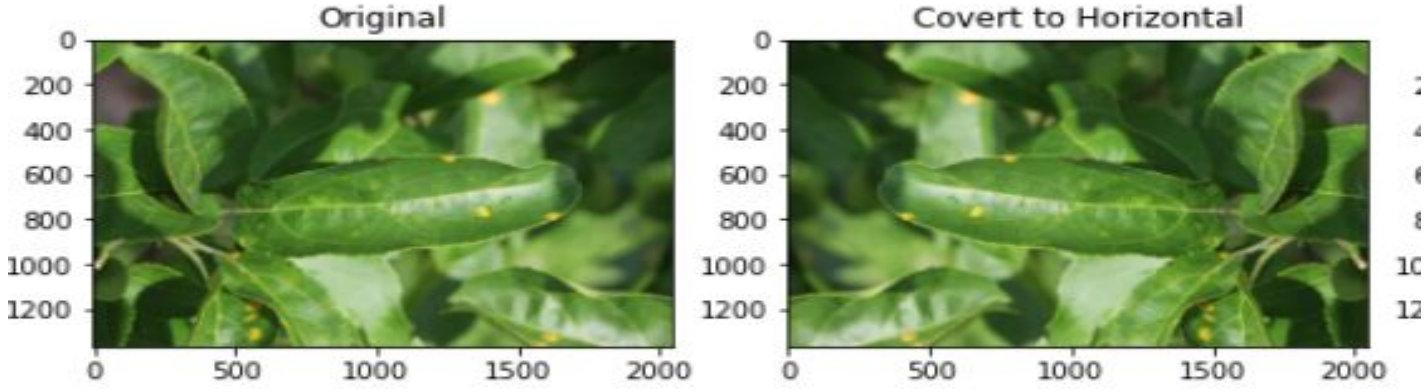
```python
transform_train = A.Compose([
    #A.Resize(450, 850),
    A.Resize(320, 512),
    A.RandomBrightnessContrast(brightness_limit=0.2, # 밝기 대비 조절
                               contrast_limit=0.2, p=0.3),
    A.VerticalFlip(p=0.2),
    A.HorizontalFlip(p=0.5),
    A.ShiftScaleRotate(
        shift_limit=0.1,
        scale_limit=0.2,
        rotate_limit=30, p = 0.3),
    A.OneOf([A.Emboss(p=1), # 양각화, 날카로움, 블러 효과
             A.Sharpen(p=1),
             A.Blur(p=1)], p=0.3),
    A.PiecewiseAffine(p=0.3), # 어파인 변환
    A.Normalize(), # 정규화 변환
    ToTensorV2() # 텐서로 변환
])
```

```python
transform_test = A.Compose([
    #A.Resize(450,850),
    A.Resize(320, 512),
    A.Normalize(),
    ToTensorV2()
])
```

전처리

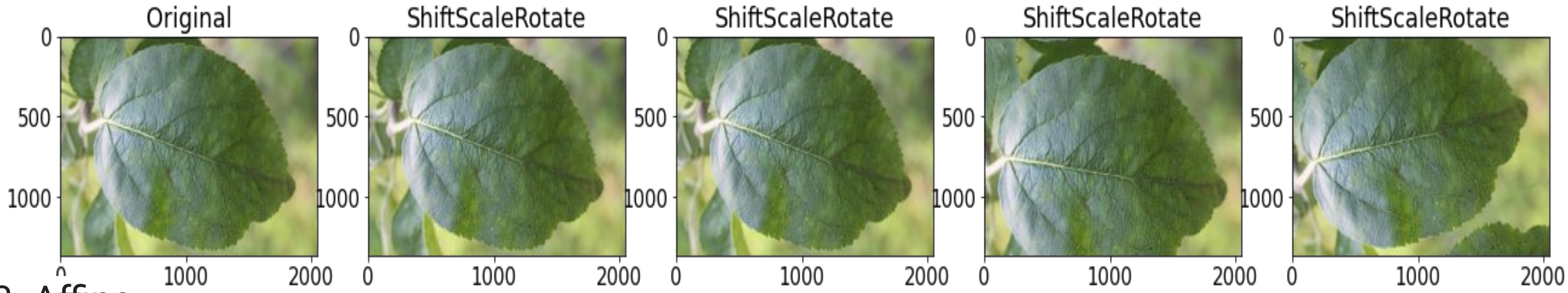
1. Vertical

2. Horizontal

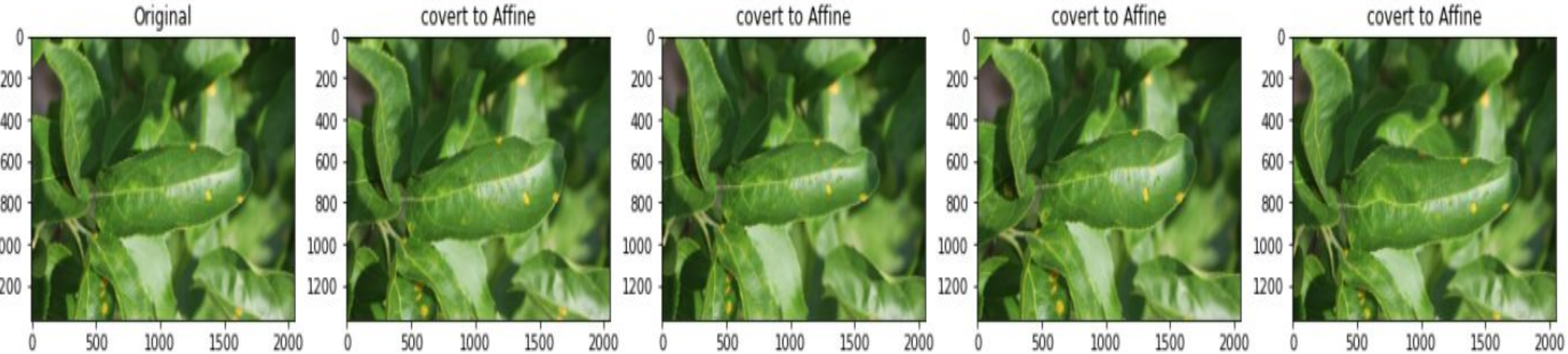


전처리

3. ShiftScaleRotate



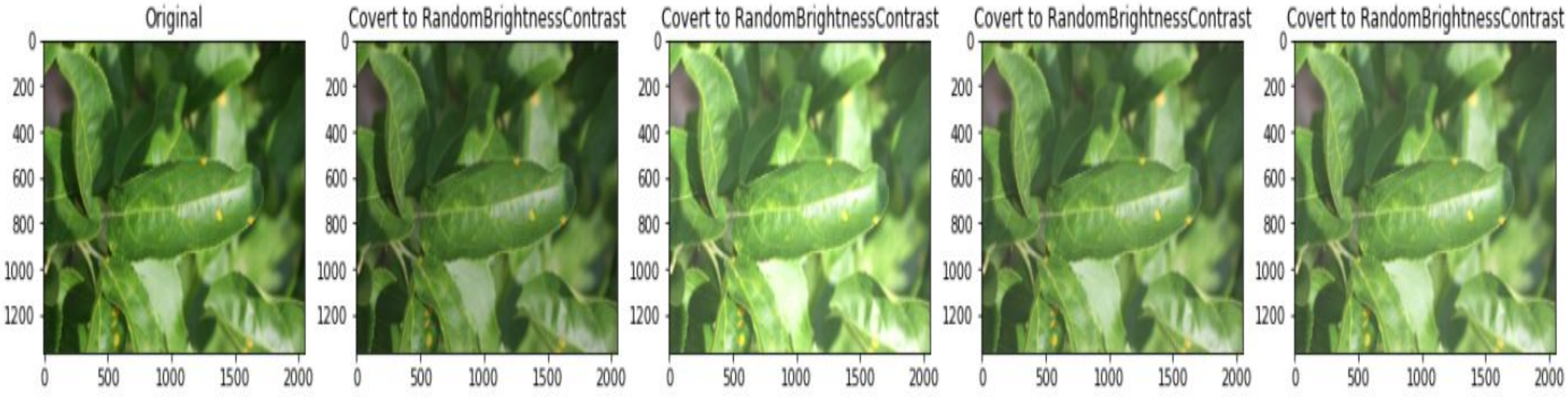
3. Affine



식물들의 자연환경의 따라 잎모양들이 다양하기 때문에 여러가지 변환하였다

전처리

5. RandomBrightneessConstrast



식물들은 대부분 야외에서 자라기때문에 랜덤한 빛의 형태가 필요하다고 생각

04

모델 선정

모델선정

1. Resnet50

2. EfficientNetB0

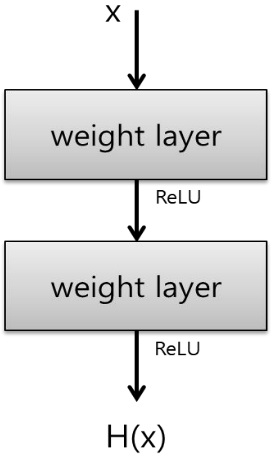
3. EfficientNetB4

3. EfficientNetB7

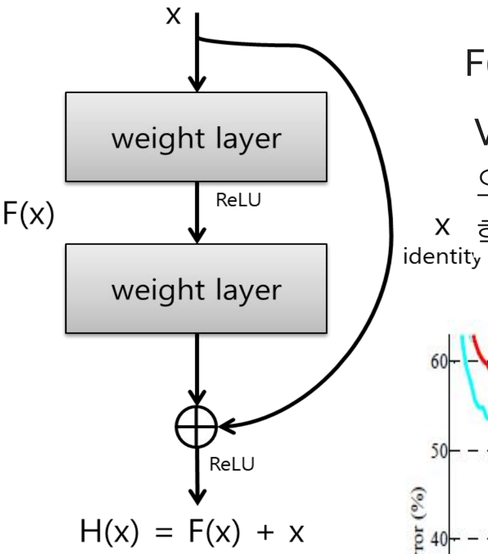
Model 4종류 선정

모델선정

Resnet50



기존 방식

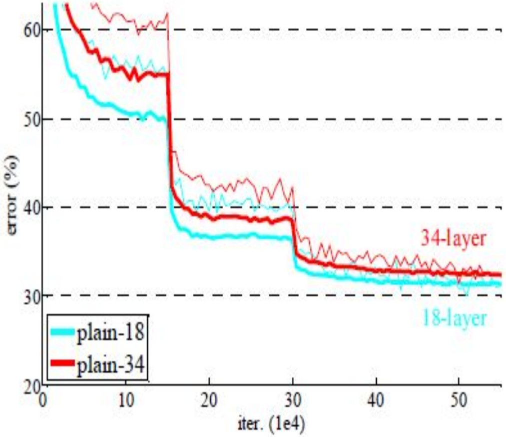


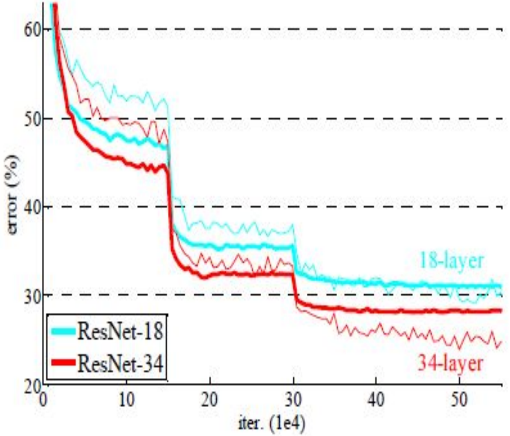
Residual block

~~다들 아시겠지만…~~

F(x) + x

VGG-19 구조로 하며 기존모델은 layer를 깊게하면
오히려 정확도가 낮았지만 shortcut을 사용하여
학습난이도를 낮추고 layer를 깊게하여 성능을 높임





모델선정

- 빠른 성능으로 ImageNet SOTA 달성
- 적은 parameter로 transfer 또한 잘 됨

EfficientNetB0~B7

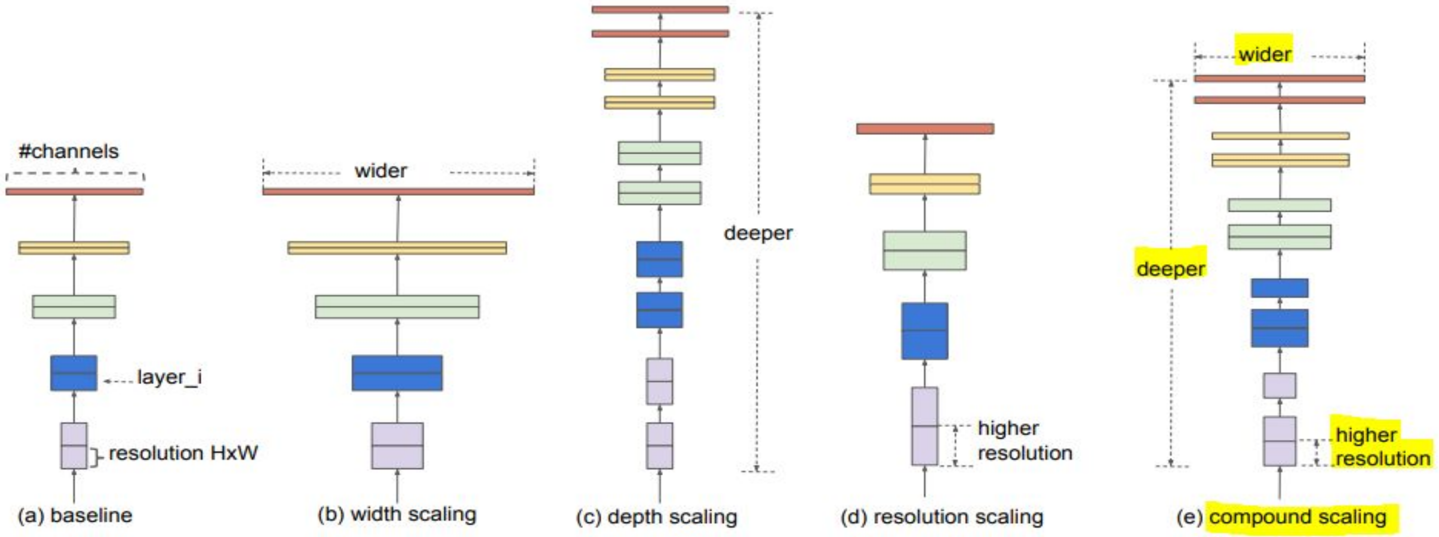
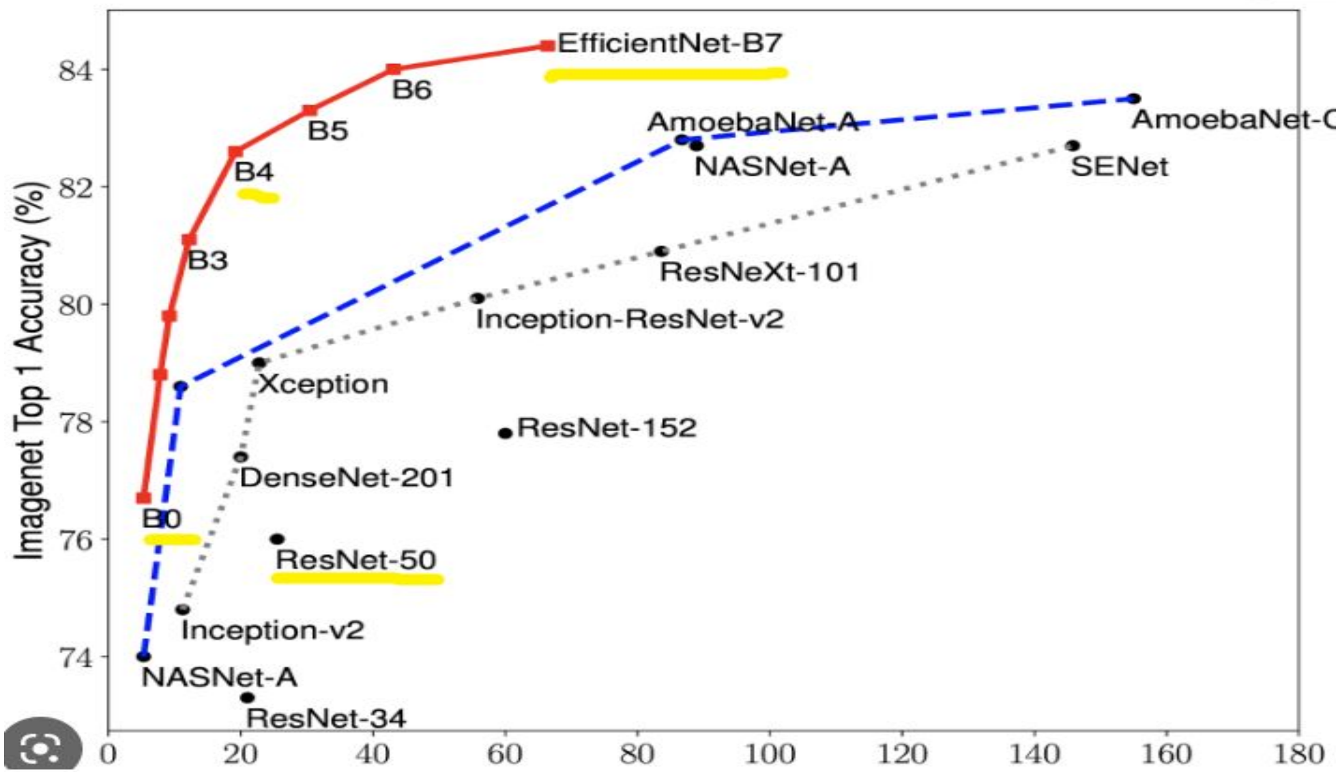


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

기존 모델에선 (width, depth, resolution) 임의로 한가지 요소를 scale

EfficientNet은 세가지 요소를 균일하게 scale

모델선정



Imagenet Top 1 Accuracy (%)
84
82
80
78
76
74
0
20
40
60
80
100
120
140
160
180
EfficientNet-B7
B6
B5
B4
B3
B0
AmoebaNet-A
AmoebaNet-C
NASNet-A
SENet
ResNeXt-101
Inception-ResNet-v2
Xception
ResNet-152
DenseNet-201
ResNet-50
Inception-v2
NASNet-A
ResNet-34

모델선정

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

05

모델 컴파일

Model 컴파일

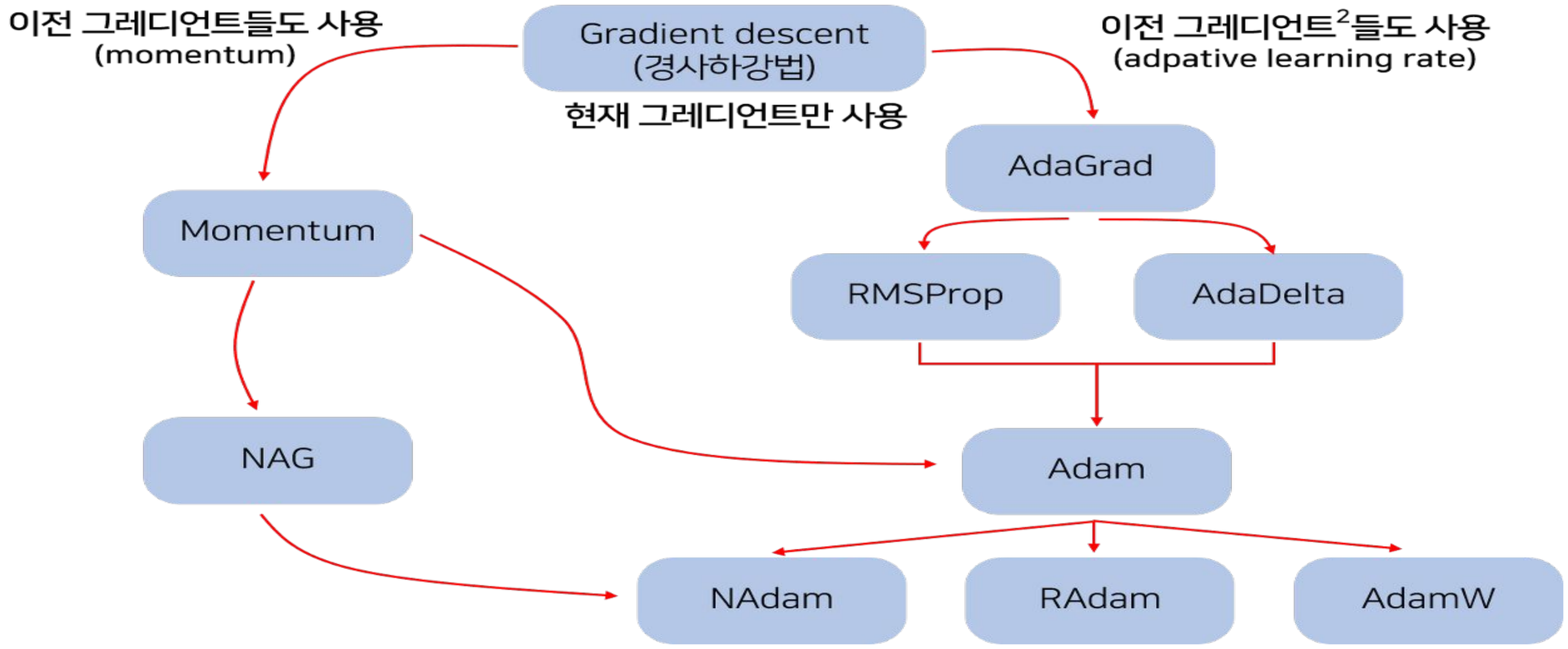
Optimizer

Loss function

Hyper
paramiter

선정이유

모델 컴파일



옵티마이저는 대부분 Adam을 사용하지만, 발전된 모델인 AdamW와 성능을 비교하였음.

모델 컴파일

손실함수로 교차엔트로피 선정

$$\mathcal{L} = \sum_{i=1}^{n} \mathrm{CrossEntropy}(\hat{y}_i, y_i)$$

$$\text{where } \hat{y}_i \in \mathbb{R}^3, y_i \in \{0,1\}^3 \text{ and } |y_i| = 1.$$

$$\mathrm{CrossEntropy}(\hat{y}i, y_i) = -y_i \times \log \hat{y}_i$$

```
train.head()
```

	image_id	healthy	multiple_diseases	rust	scab
379	Train_379	0	0	0	1
1556	Train_1556	1	0	0	0
448	Train_448	1	0	0	0
1202	Train_1202	0	0	0	1
1541	Train_1541	1	0	0	0

Label이 one-hot 벡터로 구성되어있고, 클래스가 4개, CrossEntopy가 적합하다고 판단.

모델 컴파일

하이퍼파라미터

Epoch, Batchsize, Learning Rate

모델 컴파일

ICT Express
Volume 6, Issue 4, December 2020, Pages 312-315

The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset

Ibrahem Kandel, Mauro Castelli

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.icte.2020.04.010
Get rights and content

실험 결과

Test AUC		
Batch size	Adam LR = 0.0001	Adam LR = 0.001
16	**0.9677**	0.9144
32	0.9636	0.9332
64	0.9616	0.9381
128	0.9567	0.9432
256	0.9585	**0.9652**

Test AUC		
Batch size	SGD LR = 0.0001	SGD LR = 0.001
16	0.9555	0.9461
32	**0.9570**	0.9521
64	0.9512	0.9545
128	0.9302	0.9567
256	0.9077	**0.9579**

06

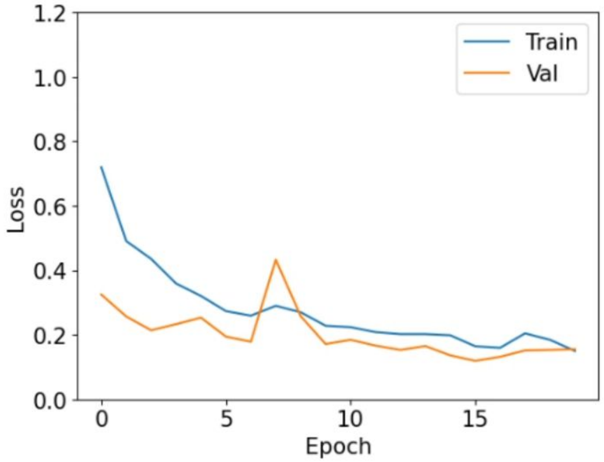
모델 훈련

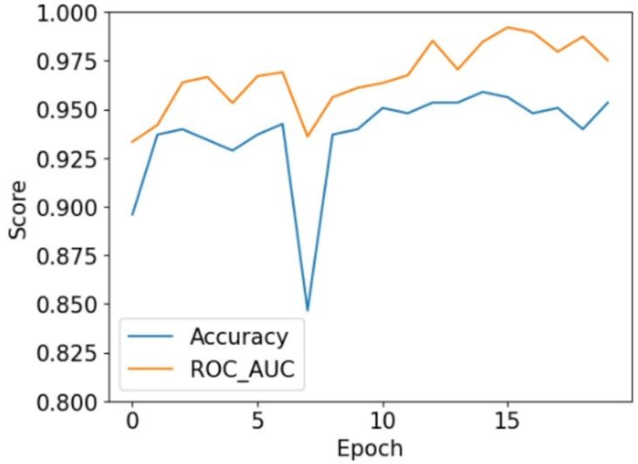
모델 실행

Resnet50

0.946

```
Epoch : 11/20... Train Loss : 0.224 /  Valid Loss : 0.184 /  Valid AUC : 0.964 /  Valid Accuracy : 0.951
trigger :  1

100% 364/364 [02:07<00:00, 2.90s/it]

Epoch : 12/20... Train Loss : 0.208 /  Valid Loss : 0.166 /  Valid AUC : 0.967 /  Valid Accuracy : 0.948

100% 364/364 [02:04<00:00, 2.88s/it]

Epoch : 13/20... Train Loss : 0.202 /  Valid Loss : 0.153 /  Valid AUC : 0.985 /  Valid Accuracy : 0.953

100% 364/364 [02:06<00:00, 2.88s/it]

Epoch : 14/20... Train Loss : 0.202 /  Valid Loss : 0.165 /  Valid AUC : 0.970 /  Valid Accuracy : 0.953
trigger :  1

100% 364/364 [02:07<00:00, 2.92s/it]

Epoch : 15/20... Train Loss : 0.198 /  Valid Loss : 0.136 /  Valid AUC : 0.985 /  Valid Accuracy : 0.959

100% 364/364 [02:07<00:00, 2.88s/it]

Epoch : 16/20... Train Loss : 0.164 /  Valid Loss : 0.119 /  Valid AUC : 0.992 /  Valid Accuracy : 0.956

100% 364/364 [02:04<00:00, 2.87s/it]

Epoch : 17/20... Train Loss : 0.160 /  Valid Loss : 0.132 /  Valid AUC : 0.990 /  Valid Accuracy : 0.948
trigger :  1

100% 364/364 [02:06<00:00, 2.88s/it]

Epoch : 18/20... Train Loss : 0.204 /  Valid Loss : 0.151 /  Valid AUC : 0.980 /  Valid Accuracy : 0.951
trigger :  2

100% 364/364 [02:05<00:00, 2.83s/it]

Epoch : 19/20... Train Loss : 0.184 /  Valid Loss : 0.153 /  Valid AUC : 0.987 /  Valid Accuracy : 0.940
trigger :  3

100% 364/364 [02:04<00:00, 2.84s/it]

Epoch : 20/20... Train Loss : 0.150 /  Valid Loss : 0.155 /  Valid AUC : 0.975 /  Valid Accuracy : 0.953
trigger :  4
```



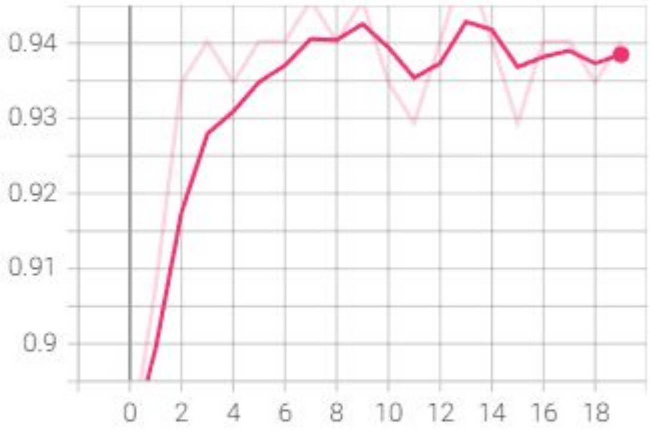


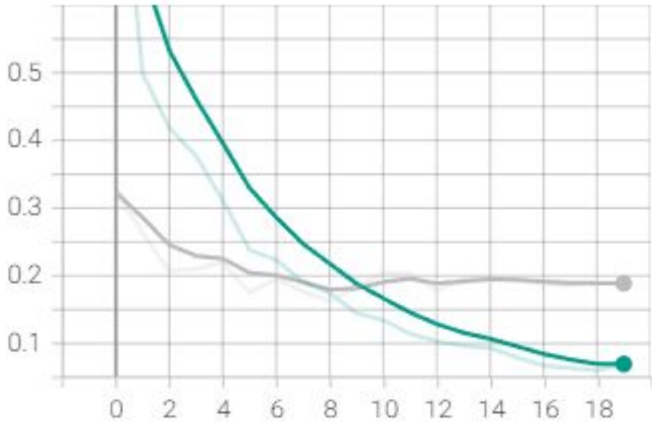
모델 실행

EfficientNet B0

0.925

```
Epoch : 1/39..... Train Loss : 0.905 Valid Loss : 0.324 Valid Accuracy : 0.886
Epoch : 2/39..... Train Loss : 0.499 Valid Loss : 0.263 Valid Accuracy : 0.908
Epoch : 3/39..... Train Loss : 0.419 Valid Loss : 0.207 Valid Accuracy : 0.935
Epoch : 4/39..... Train Loss : 0.377 Valid Loss : 0.210 Valid Accuracy : 0.940
trigger :  1
Epoch : 5/39..... Train Loss : 0.311 Valid Loss : 0.220 Valid Accuracy : 0.935
trigger :  2
Epoch : 6/39..... Train Loss : 0.238 Valid Loss : 0.176 Valid Accuracy : 0.940
Epoch : 7/39..... Train Loss : 0.223 Valid Loss : 0.195 Valid Accuracy : 0.940
trigger :  1
Epoch : 8/39..... Train Loss : 0.191 Valid Loss : 0.177 Valid Accuracy : 0.946
trigger :  2
Epoch : 9/39..... Train Loss : 0.175 Valid Loss : 0.163 Valid Accuracy : 0.940
Epoch : 10/39..... Train Loss : 0.145 Valid Loss : 0.185 Valid Accuracy  0.946
trigger :  1
Epoch : 11/39..... Train Loss : 0.135 Valid Loss : 0.204 Valid Accuracy  0.935
trigger :  2
Epoch : 12/39..... Train Loss : 0.115 Valid Loss : 0.203 Valid Accuracy  0.929
trigger :  3
Epoch : 13/39..... Train Loss : 0.103 Valid Loss : 0.178 Valid Accuracy  0.940
trigger :  4
Epoch : 14/39..... Train Loss : 0.097 Valid Loss : 0.196 Valid Accuracy  0.951
trigger :  5
Epoch : 15/39..... Train Loss : 0.093 Valid Loss : 0.200 Valid Accuracy  0.940
trigger :  6
Epoch 00015: reducing learning rate of group 0 to 6.0000e-06.
Epoch : 16/39..... Train Loss : 0.079 Valid Loss : 0.193 Valid Accuracy  0.929
trigger :  7
Epoch : 17/39..... Train Loss : 0.068 Valid Loss : 0.188 Valid Accuracy  0.940
trigger :  8
Epoch : 18/39..... Train Loss : 0.064 Valid Loss : 0.186 Valid Accuracy  0.940
trigger :  9
Epoch : 19/39..... Train Loss : 0.060 Valid Loss : 0.189 Valid Accuracy  0.935
trigger :  10
Epoch : 20/39..... Train Loss : 0.069 Valid Loss : 0.188 Valid Accuracy  0.940
```



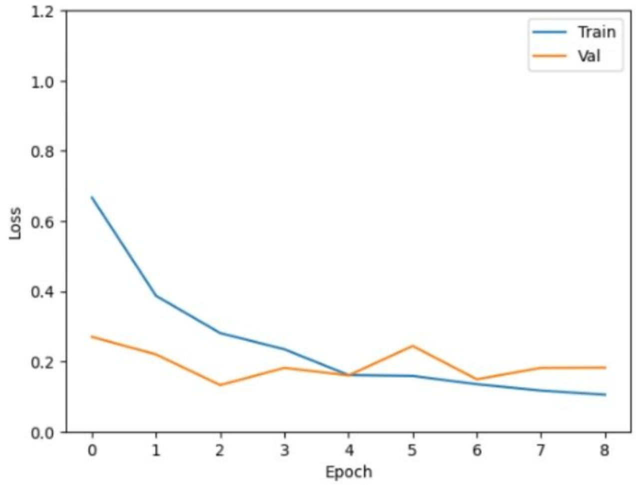


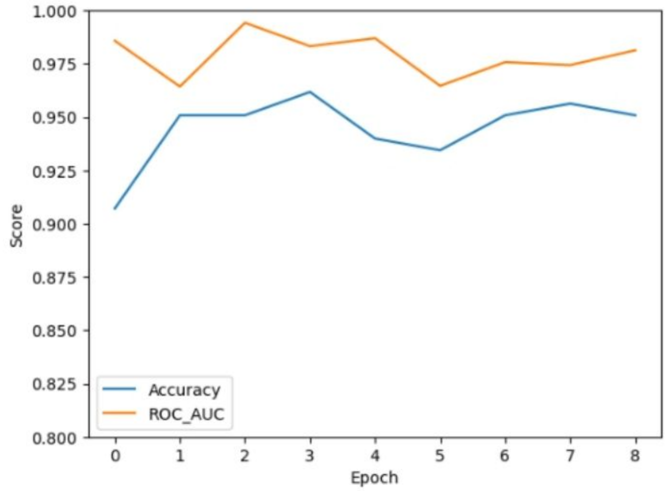
모델 실행

EfficientNet B4

0.942

```
100% 410/410 [02:04<00:00, 1.57s/it]
Epoch : 1/15... Train Loss : 0.667 /  Valid Loss : 0.270 /  Valid AUC : 0.986 /  Valid Accuracy : 0.907
100% 410/410 [01:57<00:00, 1.62s/it]
Epoch : 2/15... Train Loss : 0.387 /  Valid Loss : 0.220 /  Valid AUC : 0.964 /  Valid Accuracy : 0.951
100% 410/410 [01:59<00:00, 1.51s/it]
Epoch : 3/15... Train Loss : 0.281 /  Valid Loss : 0.132 /  Valid AUC : 0.994 /  Valid Accuracy : 0.951
100% 410/410 [02:00<00:00, 1.55s/it]
Epoch : 4/15... Train Loss : 0.234 /  Valid Loss : 0.182 /  Valid AUC : 0.983 /  Valid Accuracy : 0.962
trigger :  1
100% 410/410 [01:58<00:00, 1.47s/it]
Epoch : 5/15... Train Loss : 0.161 /  Valid Loss : 0.160 /  Valid AUC : 0.987 /  Valid Accuracy : 0.940
trigger :  2
100% 410/410 [01:59<00:00, 1.44s/it]
Epoch : 6/15... Train Loss : 0.159 /  Valid Loss : 0.244 /  Valid AUC : 0.965 /  Valid Accuracy : 0.934
trigger :  3
100% 410/410 [01:59<00:00, 1.44s/it]
Epoch : 7/15... Train Loss : 0.135 /  Valid Loss : 0.149 /  Valid AUC : 0.976 /  Valid Accuracy : 0.951
trigger :  4
100% 410/410 [02:00<00:00, 1.52s/it]
Epoch : 8/15... Train Loss : 0.117 /  Valid Loss : 0.182 /  Valid AUC : 0.974 /  Valid Accuracy : 0.956
trigger :  5
100% 409/410 [01:58<00:00, 3.16it/s]
Epoch : 9/15... Train Loss : 0.105 /  Valid Loss : 0.182 /  Valid AUC : 0.981 /  Valid Accuracy : 0.951
trigger :  6
Early Stopping!!!
Traning step is finished!!
```



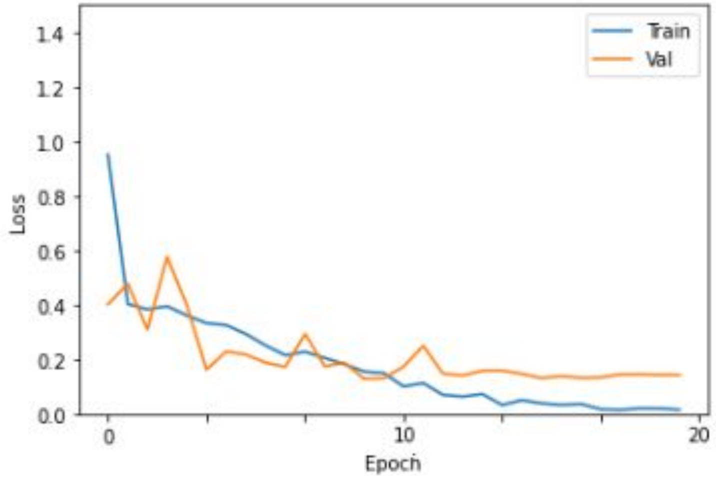


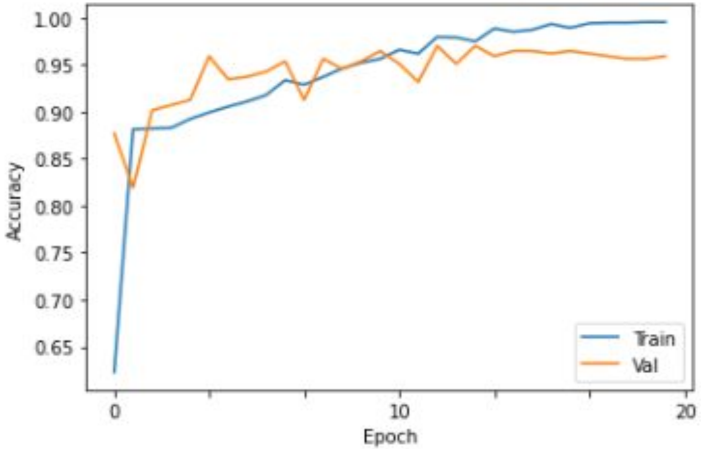
모델 실행

EfficientNet B7

0.945

```
100%|███████████▌| 409/410 [07:51<00:01,  1.13s/it]
Epoch : 1/20..... Train Loss : 0.670 Valid Loss : 0.207 Valid Accuracy : 0.929
100%|████████████| 410/410 [08:07<00:00,  1.19s/it]
100%|███████████▌| 409/410 [07:51<00:01,  1.15s/it]
Epoch : 2/20..... Train Loss : 0.368 Valid Loss : 0.204 Valid Accuracy : 0.962
100%|████████████| 410/410 [08:06<00:00,  1.19s/it]
100%|████████████| 410/410 [08:05<00:00,  1.19s/it]
Epoch : 3/20..... Train Loss : 0.272 Valid Loss : 0.181 Valid Accuracy : 0.935
100%|████████████| 410/410 [08:04<00:00,  1.18s/it]
Epoch : 4/20..... Train Loss : 0.202 Valid Loss : 0.188 Valid Accuracy : 0.946
trigger :  1
100%|████████████| 410/410 [08:03<00:00,  1.18s/it]
Epoch : 5/20..... Train Loss : 0.149 Valid Loss : 0.254 Valid Accuracy : 0.940
trigger :  2
100%|████████████| 410/410 [08:07<00:00,  1.19s/it]
Epoch : 6/20..... Train Loss : 0.119 Valid Loss : 0.242 Valid Accuracy : 0.946
trigger :  3
100%|████████████| 410/410 [08:05<00:00,  1.18s/it]
Epoch : 7/20..... Train Loss : 0.111 Valid Loss : 0.242 Valid Accuracy : 0.940
trigger :  4
100%|████████████| 410/410 [08:04<00:00,  1.18s/it]
Epoch : 8/20..... Train Loss : 0.094 Valid Loss : 0.158 Valid Accuracy : 0.962
100%|████████████| 410/410 [08:07<00:00,  1.19s/it]
Epoch : 9/20..... Train Loss : 0.083 Valid Loss : 0.244 Valid Accuracy : 0.951
trigger :  1
100%|████████████| 410/410 [08:08<00:00,  1.19s/it]
Epoch : 10/20..... Train Loss : 0.057 Valid Loss : 0.316 Valid Accuracy : 0.918
trigger :  2
100%|████████████| 410/410 [08:06<00:00,  1.19s/it]
Epoch : 11/20..... Train Loss : 0.061 Valid Loss : 0.204 Valid Accuracy : 0.951
trigger :  3
100%|████████████| 410/410 [08:05<00:00,  1.18s/it]
Epoch : 12/20..... Train Loss : 0.049 Valid Loss : 0.341 Valid Accuracy : 0.924
trigger :  4
100%|████████████| 410/410 [08:06<00:00,  1.19s/it]
Epoch : 13/20..... Train Loss : 0.054 Valid Loss : 0.282 Valid Accuracy : 0.935
trigger :  5
100%|███████████▌| 409/410 [07:52<00:01,  1.13s/it]
Epoch : 14/20..... Train Loss : 0.043 Valid Loss : 0.202 Valid Accuracy : 0.967
```






07
결과

결과

ResNet50

submission_resnet50_e15_rrp.csv Complete (after deadline) · 1s to go · resnet50 e15 rrp	0.95265	0.96257

EfficientNet B0

EfficientNet_B0 - Version 5 Complete (after deadline) · 1s to go	0.92975	0.93912	☐

EfficientNet B4

subm_effnetb4.csv Complete (after deadline) · 1s to go · effnetb4 rrp	0.9303	0.94252

EfficientNet B7

Submission and Description	Private Score	Public Score	Selected
submission_test_Eff_B07.csv Complete (after deadline) · now · Eff_B07 submission	0.94344	0.95338	☐

08

성능개선

모델 성능개선

옵티마이저 변경

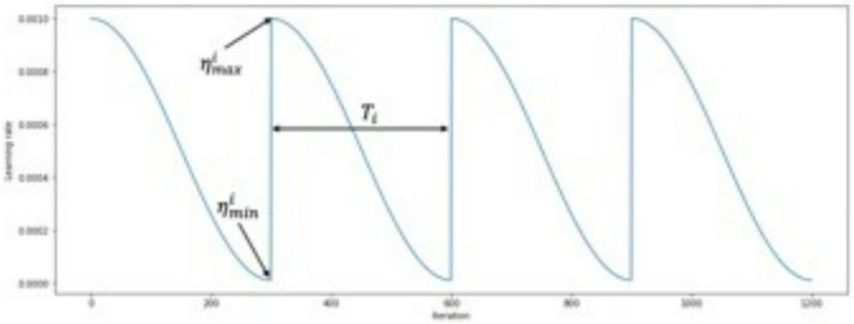
최적의 파라미터 선정

(epoch, scheduler, batch_size..)

이미지 크기 조절

모델 성능개선

스케쥴러 교체



[그림2] Cosine annealing

```
optimizer = torch.optim.AdamW(model.parameters(), lr=0.0001)
scheduler = CosineAnnealingWarmRestarts(optimizer,T_0=5, T_mult=1, eta_min=0.00001)
```

모델 성능개선

ResNet50

subm_resnet50_cosineAlr.csv 0.95198 0.96613

Complete (after deadline) · 18h ago · plant-pathology-ResNet50_e15_AdamW_CosineALR_bsize4_rl0001

EfficientNet B0

subm_effnetb0_cosine.csv 0.92897 0.94362

Complete (after deadline) · 1s to go · b0 cosine

EfficientNet B4

subm_effnetb4_cosineAlr.csv 0.9303 0.94252

Complete (after deadline) · 13h ago · plant-pathology-EfficientNetB4_e30_AdamW_CosineALR

EfficientNet B7

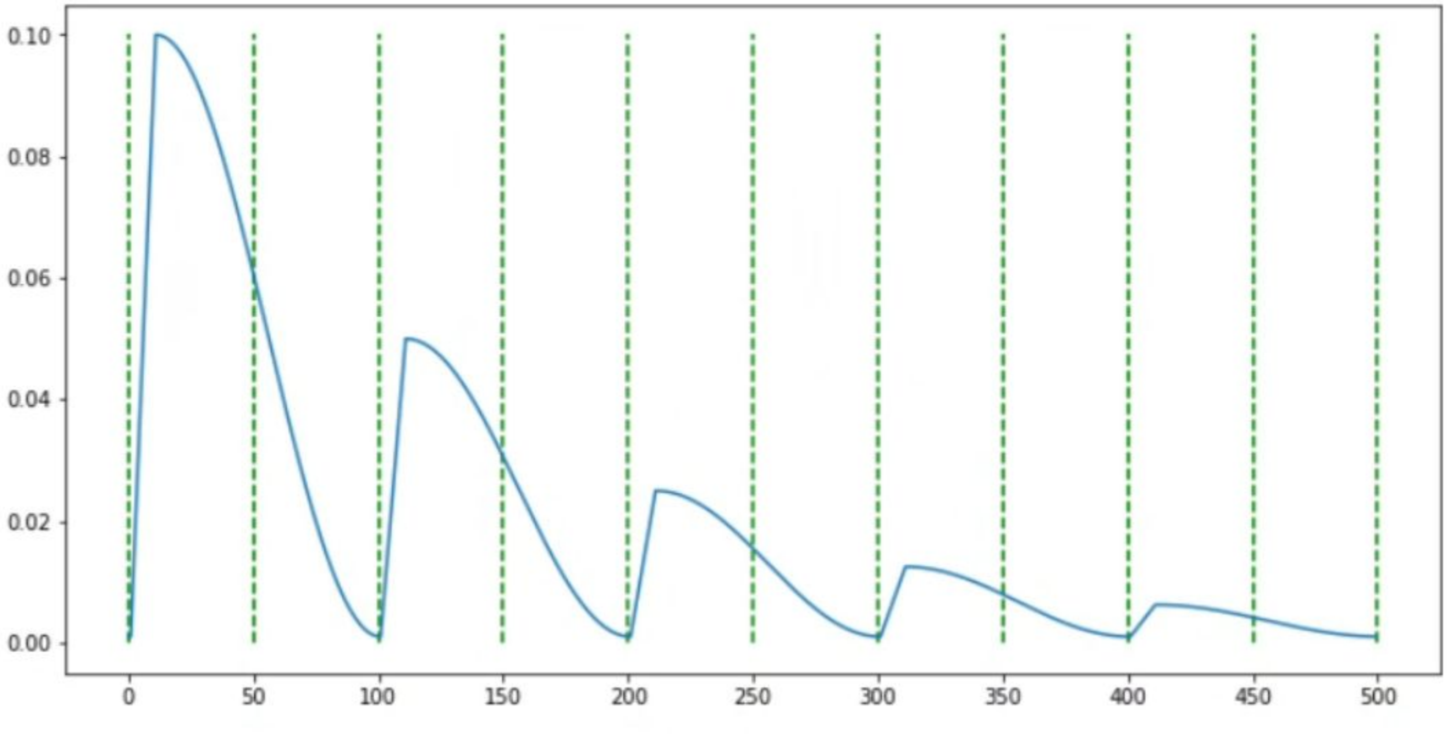
subm_effnetb7_cosineAlr.csv 0.95086 0.95878

Complete (after deadline) · 14h ago · plant-pathology-EfficientNetB7_e15_AdamW_CosineALR

모델 성능개선

추가 개선사항

Custom CosineAnnelingWarmRestart



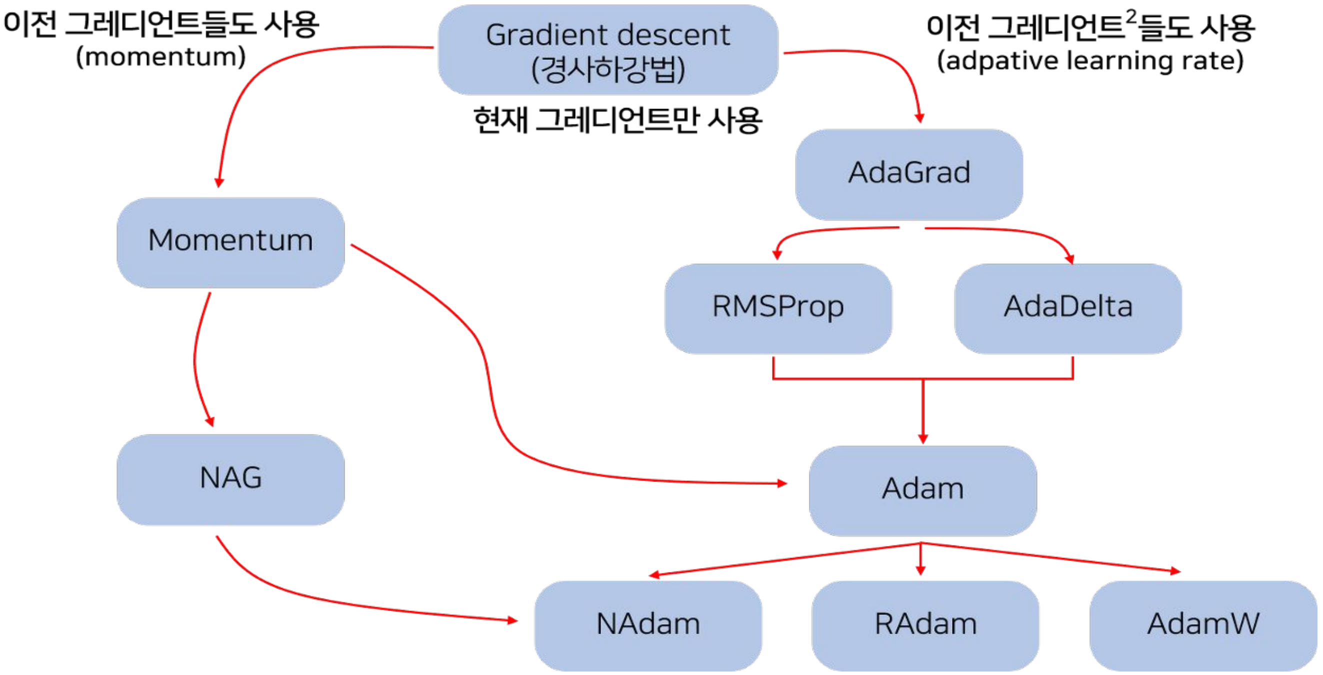
모델 성능개선

추가 개선사항

모델별 권장 이미지 크기

	Model	Recommended Image Size
EfficientNet V1	B0	224
	B1	240
	B2	260
	B3	300
	B4	380
	B5	456
	B6	528
	B7	600
EfficientNet V2	S	384
	M 한글	480
	L	480
	B0	224
	B1	240
	B2	260
	B3	300

모델 성능개선



이전 그레디언트들도 사용
(momentum)
Gradient descent
(경사하강법)
현재 그레디언트만 사용
이전 그레디언트²들도 사용
(adpative learning rate)
AdaGrad
Momentum
RMSProp
AdaDelta
NAG
Adam
NAdam
RAdam
AdamW

THANK YOU!