




PADDY DOCTOR

: Paddy Disease Classification

(Kaggle)



Our team

팀원 소개 및 역할

[팀장]

김찬별

- * Data balancing 문제점을 활용해 sequential한 stepwise 모델 개발중

[부팀장]

박윤수

- * 옵티마이저 비교

최찬혁

- * RESNet으로 자체 개발
 - 자료 조사 및 코드 수정
 - 코드 실행 및 분석
 - 오류에 대한 피드백 및 코드 조정

조기쁨

- * Pytorch를 이용해 RESNet 모델 개발중
- * 최찬혁 코드를 이용하여 RESNet 파라미터 비교

이주행

- * 이미지 선명화
 - 이미지 벡터 부분 검출

Overview

Paddy Doctor : Paddy Disease Classification

■ Problem Statement

- 벼의 수확량은 질병에 의해 70% 가까이 수확량이 떨어짐
- 그리하여 지속적인 관리감독이 필요한데, 인력으로 해결하기에는 기회비용이 너무 많이 듦
- 이러한 점 때문에 컴퓨터 비전을 이용하여 질병 식별 프로세스를 자동화하는 것이 중요해짐

■ Objective

- 10,407개의 학습 데이터가 주어짐
- 이는 벼의 상태를 10가지(9개의 질병과 1개의 정상)로 분류함
- 이 학습 데이터를 이용하여 머신러닝 혹은 딥러닝 모델을 완성하고 3,469개의 벼 이미지를 10가지의 상태로 분류함

Mechanism

Mechanism

데이터 불러오기



데이터 증식
및 선명화

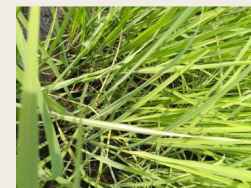


모델 제작 및
데이터 학습

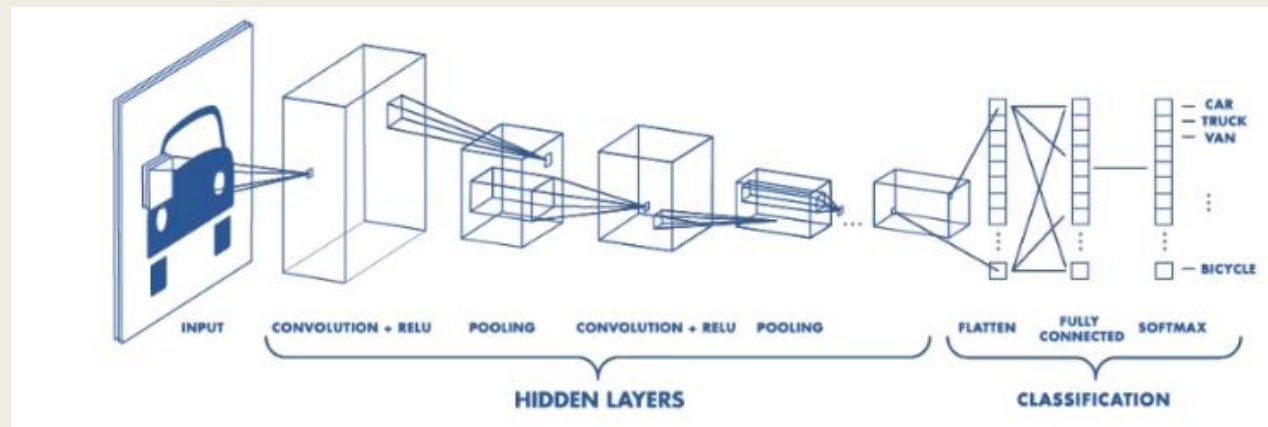


데이터 예측

Imagegenerator



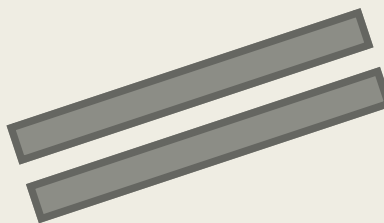
CNN

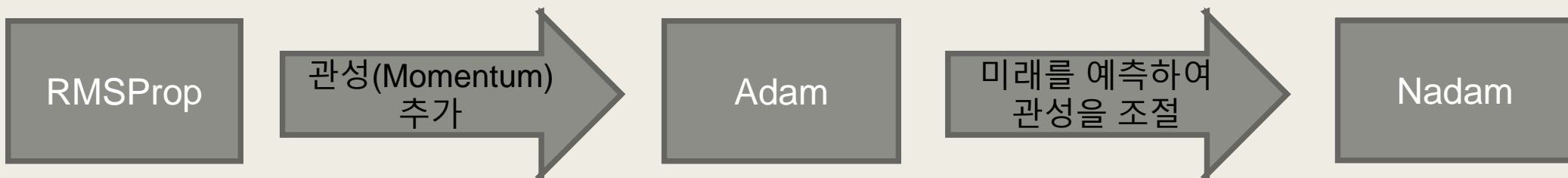


Modeling

옵티마이저(Optimizer) – (Paddy Doctor: Paddy Disease Classification 코드 사용)

손실함수(loss function)값을 최소화 시키는 알고리즘


$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$



Epoch	24	<	29	<	36
Accuracy	0.872	<	0.954	<	0.966

이미지 선명화 – unsharp mask

처리 전



처리 후



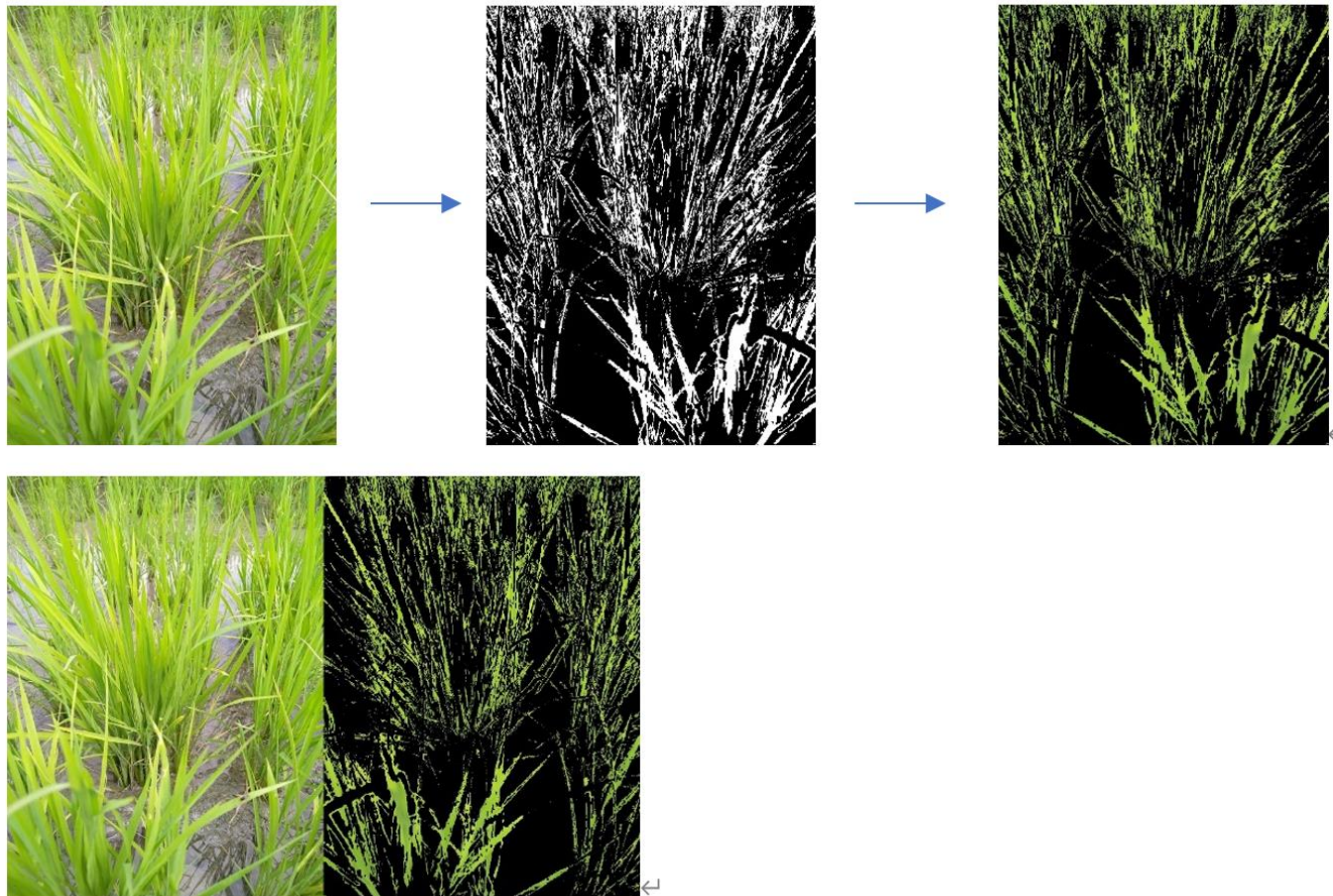
Unsharp mask : 이미지를 blur 처리한 후 원본 이미지에서 빼 흐릿하지 않은 부분을 구한 후 다시 이것을 원본 이미지에 더해서 선명도를 향상시킨다.

이미지 선명화 비교

처리여부	Batch	Epoch	Size	Val_loss	Val_accuracy
처리 안함	32	20	150	0.6906	0.7780
처리 안함	32	30	150	0.5444	0.8450
처리 안함	32	30	256	0.4669	0.8666
선명도 향상	32	30	150	0.6412	0.8146
선명도 향상	32	30	256	0.5504	0.8382
선명도 향상	64	120	256	0.3715	0.9263

벼 부분 검출

이미지 원본 -> 이진 마스크 생성 -> 마스크와 이미지 결합



opencv 히스토그램을 이용
가장 많은 색상을 검출

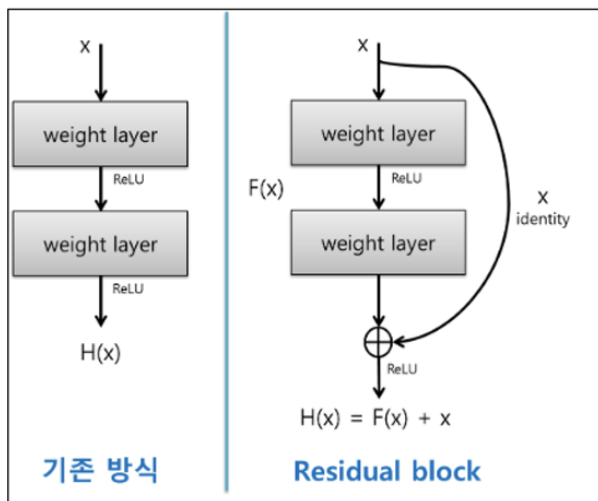
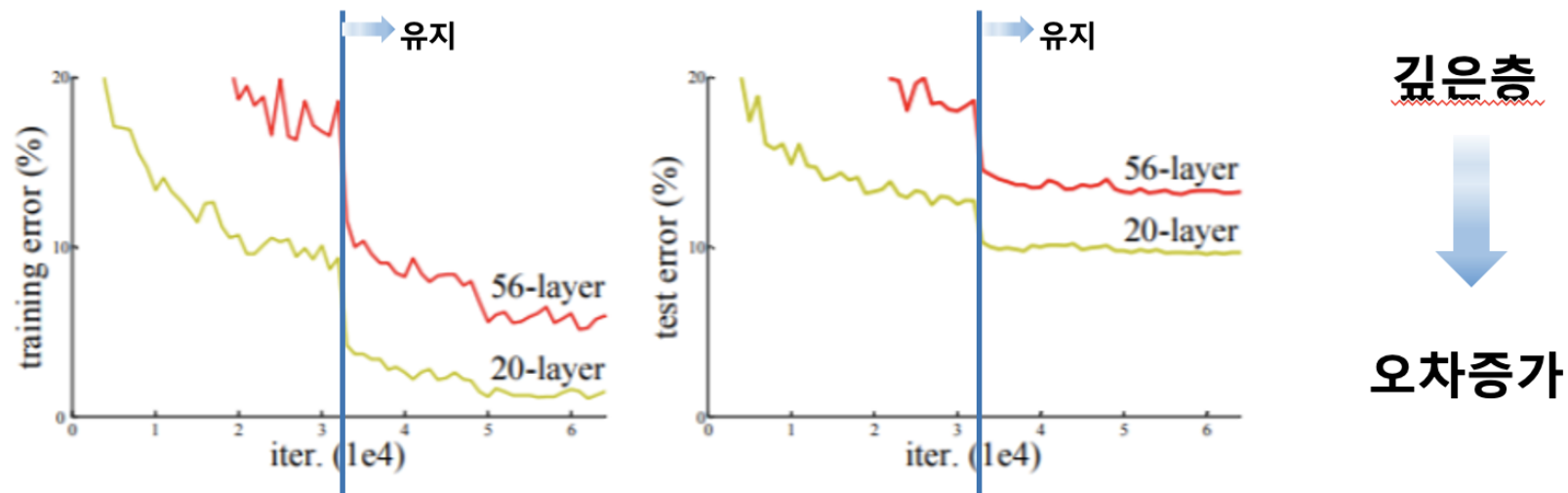
색상정보 바탕으로 마스크
생성하여 벼 부분만 검출

벼 부분이 제대로
검출되지 않음

개선 필요

RESNet

■ 개요



문제
경사소실

해결
Skip Connection

- 기존의 신경망은 입력값 x를 타겟값 y로 매핑하는 함수 $H(x)$ 를 얻는 것이 목적
- ResNet은 $F(x) + x$ 를 최소화하는 것을 목적
- x 는 현시점에서 변할 수 없는 값이므로 $F(x)$ 를 0에 가깝게 만드는 것이 목적
- 즉, 현재 입력값과 출력의 차이만을 학습

■ Basic Architectures

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

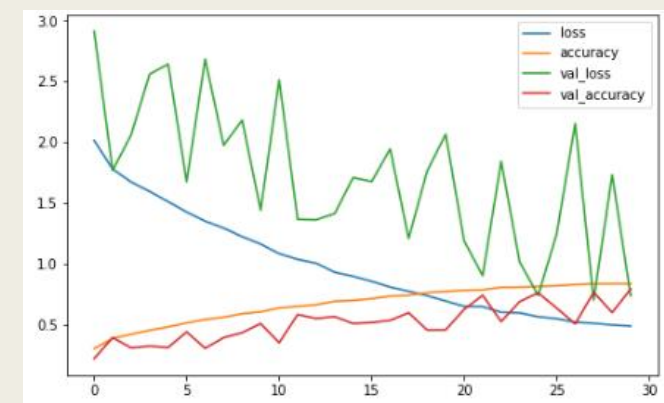
```
1 resnet = ResNet50(weights='imagenet')
2 resnet.summary()
```



7 - 'res2a_branch2a' conv 64, 1x1, 1, valid (6)
 8 - 'bn2a_branch2a' bn (7)
 9 - 'activation_2' relu (8)
 10 - 'res2a_branch2b' conv 64, 3x3, 1, same (9)
 11 - 'bn2a_branch2b' bn (10)
 12 - 'activation_3' relu (11)
 13 - 'res2a_branch2c' conv 256, 1x1, 1, valid (12)
 *14 - 'res2a_branch1' conv 256, 1x1, 1, valid (6)

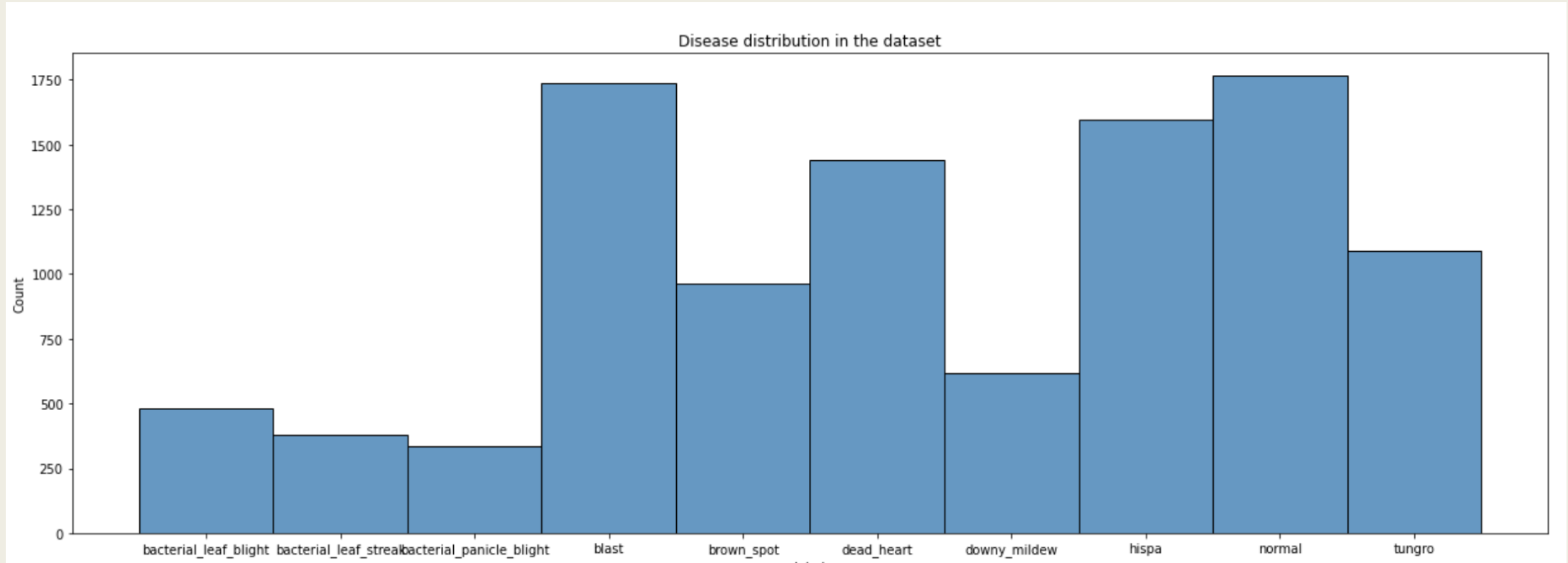
■ Standard parameters & Results

Parameter	Img_size	Batch_size	Epoch	Time(s)	Score
Value	(128,128)	16	30	3352	0.7474



Data balancing - Problem

■ Paddy Disease distribution in the Dataset



=> Imbalance

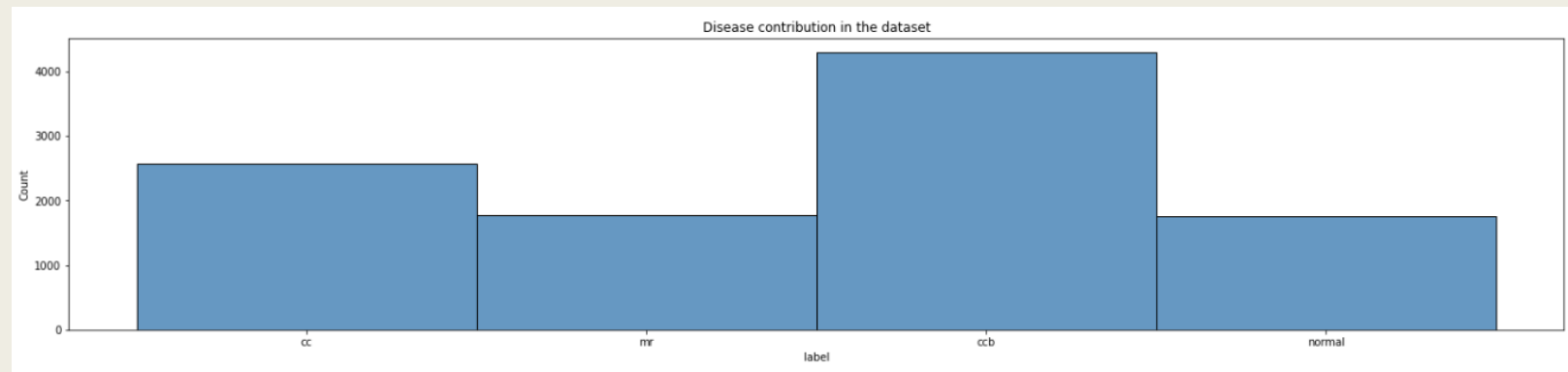
Data balancing – How to improve

* Objective

- 질병에 따른 유사성 구분해서 예측하기
- 데이터 label별로 correlation 구해서 group으로 묶기

* Process

- Model class가 많아지면 예측하기가 어려워짐 -> group 수 줄이기
- cc(color change) – tungro, downy_mildew, bacterial_leaf_blight, bacterial_leaf_streak / mr(middle rice) – dead_heart, bacterial_panicle_blight / ccb(color change + break) – hispa, blast, brown_spot / normal



Data balancing - How to improve

* Process

- `data.replace()` -> 대체
- Path 지정
- `LabelEncoder()`
- `ImageDataGenerator()`
- `flow()`

* Error

- `ValueError: setting an array element with a sequence.`

Discussion

Discussion & Feedback - RESNet

✓ 01

PB : Val_loss 값과 Val_accuracy 값이 위아래로 진동

Cause : 입력과 출력의 차이를 잡아내는 shortcut에 의한 현상으로 파악

Sol : Callback에서 더 좋은 모델을 저장하는 Modelcheckpoint 사용
(기존 : Earlystopping)

Score : 0.80584 & Time : 3364s

✓ 02

Change : FCL (2048)(1024)(128) => (2048)(1024)(512)(256)(128)

Discussion : FCL은 공간정보를 소실한다는 단점 존재

Score : 0.58477 & Time : 3173s

Add : FCL에 Dropout(0.2) 추가

Discussion : 유사성이 많은 데이터이기에 오히려 정확도 하락

Score : 0.6855 & Time : 3215s

✓ 03

Change : Conv층을 50층이 아닌 101층으로 구현(RESnet101)

Discussion : Pooling을 통한 정보요약 증가 (손실 증가)
50층도 깊은 느낌.

Score : 0.60553 & Time : 3918s

Discussion & Feedback

✓ 04

Change : Conv층을 50층이 아닌 18층으로 구현(RESnet18)

Discussion : 구조파악 불가

Score : - & Time : -

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

```
1 resnet = ResNet50(weights='imagenet')
2 resnet.summary()
```

7 - 'res2a_branch2a' conv 64, 1x1, 1, valid (6)
8 - 'bn2a_branch2a' bn (7)
9 - 'activation_2' relu (8)
10 - 'res2a_branch2b' conv 64, 3x3, 1, same (9)
11 - 'bn2a_branch2b' bn (10)
12 - 'activation_3' relu (11)
13 - 'res2a_branch2c' conv 256, 1x1, 1, valid (12)
*14 - 'res2a_branch1' conv 256, 1x1, 1, valid (6)

Discussion : 유사한 데이터를 정리하면 더욱 uniform한 결과 기대가능

```

tqdm_dataset = tqdm(enumerate(val_dataloader))
print(tqdm_dataset)
training = False
for batch, batch_item in tqdm_dataset:
    batch_loss = train_step(batch_item, epoch, batch, training)
    total_val_loss += batch_loss

    tqdm_dataset.set_postfix({
        'Epoch': epoch + 1,
        'Val Loss': '{:06f}'.format(batch_loss.item()),
        'Total Val Loss' : '{:06f}'.format(total_val_loss/(batch+1))
    })
val_loss_plot.append(total_val_loss/(batch+1))

if val_loss_plot[-1]<0.04:
    torch.save(model, 'models/model'+str(epoch+1)+str(val_loss_plot[-1])+'.pt')
if min(val_loss_plot) == val_loss_plot[-1]:
    torch.save(model, 'model.pt')

```

```

it [00:02, 6.21it/s, Epoch=1, Loss=0.000000, Total Loss=0.216538]
t [00:00, ?it/s]
t [00:00, ?it/s]

```

```

error: Caught error in DataLoader worker process 1.
Original Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/torch/utils/data/_utils/worker.py", line 287, in _worker_loop
    data = fetcher.fetch(index)
  File "/opt/conda/lib/python3.7/site-packages/torch/utils/data/_utils/fetch.py", line 49, in fetch
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "/opt/conda/lib/python3.7/site-packages/torch/utils/data/_utils/fetch.py", line 49, in <listcomp>
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "/tmp/ipykernel_33/3260224000.py", line 15, in __getitem__
    img = cv2.resize(img, dsize=(128, 128)) #interpolation=cv2.INTER_AREA
cv2.error: OpenCV(4.5.4) /tmp/pip-req-build-jpmv6t9/_opencv/modules/imgproc/src/resize.cpp:4051: error: (-215





```

pytorch를 이용해 Resnet 모델 작성 하였으나, 1 epoch 수행 이후 error 발생

-> CustomDataset을 통해 cv2.imread() 이미지를 읽어오는 과정

-> 프로젝트 끝난 이후 지속적으로 error 수정할 예정

-> 찬혁님이 만드신 resnet 모델로 파라미터 변경해봄.

<div data-bbox="310 274 496 359">  01 </div>	<ul style="list-style-type: none"> - epoch = 30, 소요 시간 : 3364s, valid_acc = 0.7963 - 27번째에서 가장 높은 val_accuracy를 보임.
<div data-bbox="310 531 802 616">  02 epoch = 30 -> 60 </div>	<ul style="list-style-type: none"> - epoch = 60, 소요 시간 : 5041s, valid_acc = 0.889로 이전보다 상승 - 48번째에서 0.8890으로 가장 높은 val_accuracy를 보임. Learning curve에서 훈련 셋, 검증 셋의 Accuracy는 1을 향해 상승하고 있어 안정적으로 학습이 된 것으로 파악됨.
<div data-bbox="310 802 871 888">  03 batch_size = 16 -> 64 </div>	<ul style="list-style-type: none"> - epoch = 30, 소요 시간 : 3132s, valid_acc = 0.78 - val_loss가 이전보다 커지고 훈련 셋과 검증 셋의 Accuracy 상승도는 이전보다 하락함. val_loss가 비정상적으로 큰 문제를 해결하기 위해 learning_rate나 옵티마이저를 바꿔 봐야 할 듯.
<div data-bbox="310 1073 835 1159">  04 optimizer Adam -> RMSprop </div>	<ul style="list-style-type: none"> - epoch = 30, 소요 시간 : 3074s, valid_acc = 0.8127 - Adam optimizer의 val_loss보다 확연히 줄어듦. - Resnet 18, 34 모델도 돌려보면 좋을듯.

Discussion & Feedback - Data balancing

* Error

- ValueError: setting an array element with a sequence.

* How to solve?

- Array, dimension 맞춘 후, train model building
- Test model building
- Accuracy 보고 다시 class 분류

Thank you