



카사바 나무에 발생하는 질병 분류

채희선 소프트웨어학부 20185542



목차

A table of Contents

#1, 전처리 방법

#2, Algorithm

#3, 결과



전처리

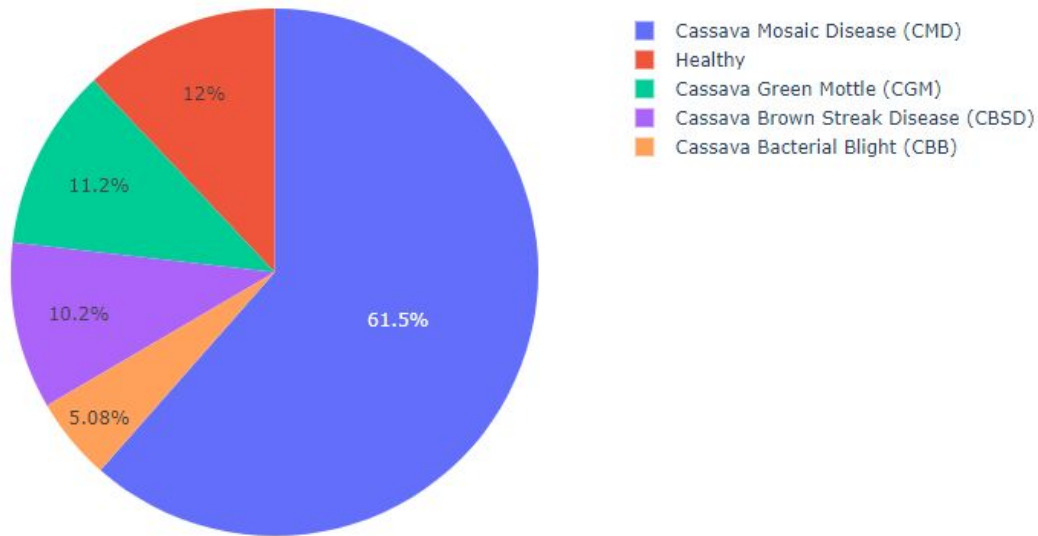
➡ 이미지 넷에서 전이학습 예정이기 때문에 이미지 넷 데이터의 평균과 분산을 이용해 이미지를 정규화 함.

모델링

➡ 빠른 학습과 좋은 성능을 모두 챙기기 위해, **Efficient B4**를 선택함.

imagenet에서 **efficient-net B4**를 이용해 이미지 분류에서 좋은 성능을 보여주고 있는 **noisystudent**를 전이 학습 하였음.

Percentage Distribution of Labels in the Training Dataset



데이터 양의 Unbalance

- ➡ 데이터 셋의 반 이상이 CMD로 구성되어 있는 등
- ➡ 아무런 성능 개선이 없다면?
- ➡ CMD : 약 80%, CBB: 약 60% 적중률
- ➡ 문제점(결론):
데이터가 많은 것은 학습을 잘해 적중도가 높지만,
데이터가 적은 것은 잘 구분을 하지 못해 성능이 낮아짐.
- ➡ 해결(성능 개선 아이디어) : CutMix 사용!
- ➡ cutmix는 정보 손실이 mixup이나 다른 dropout기반 이미지 증강보다 적고 학습과정에서 서로 다른 라벨의 데이터가 섞이게 되면 많은 양의 데이터에 적은 양의 데이터가 희석되면서 불균형 완화에 도움을 줄것으로 기대.

Algorithm

Guess: Cassava Mosaic Disease (CMD)

Cassava Green Mottle (CGM)
339278657.jpg



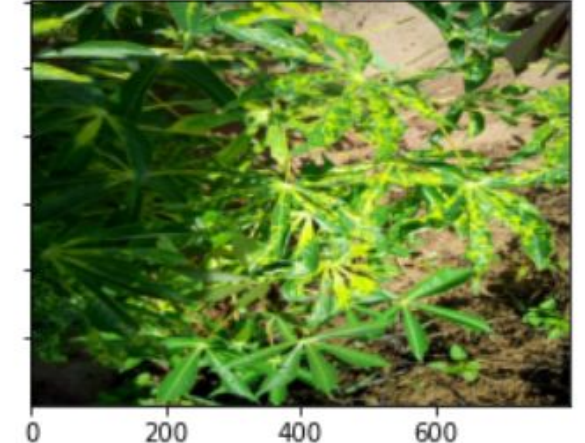
Healthy
832729024.jpg



Healthy
1951968907.jpg



Healthy
3402201686.jpg

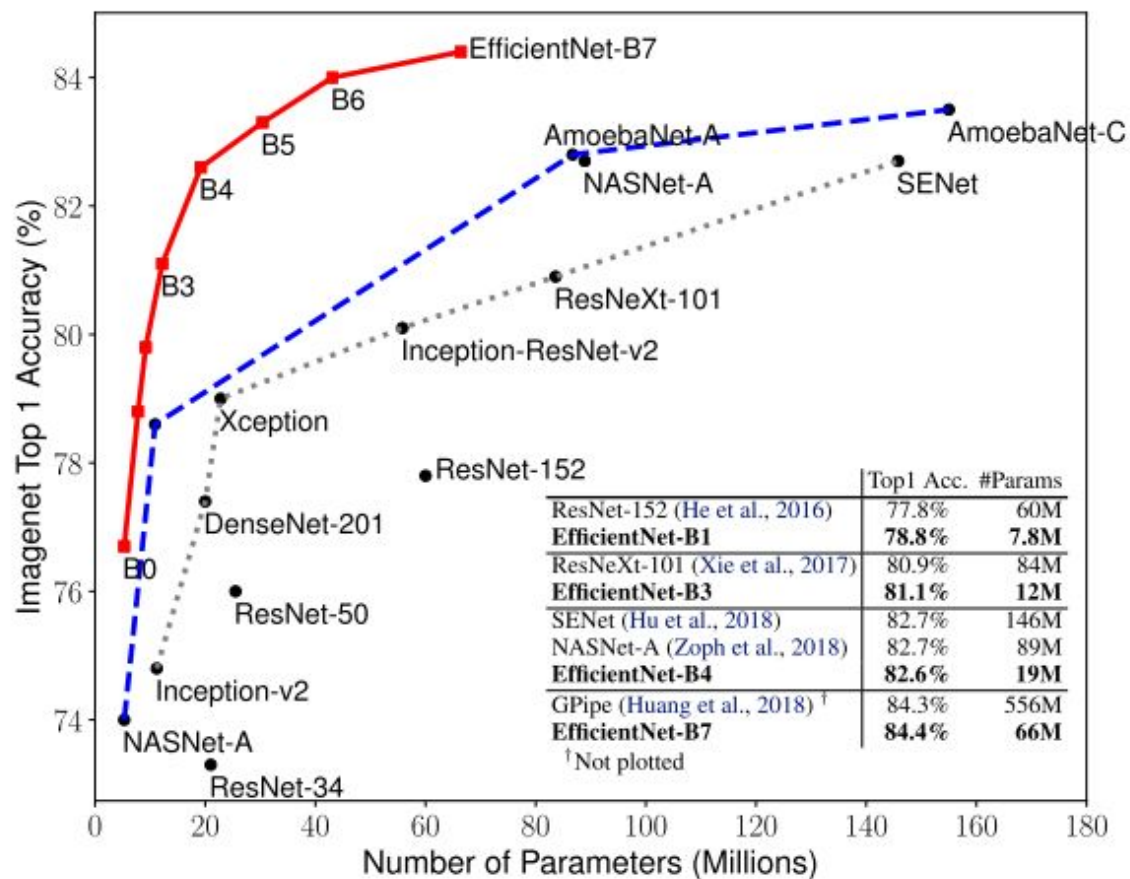


Noise Data

- ➡ train data set에 잘못 label된 데이터가 존재.
- ➡ telljoy EDA
(<https://www.kaggle.com/telljoy/noisy-label-eda-with-cleanlab>)
에서는 noise data가 8%있다고 추정.

- ➡ 아무런 성능 개선이 없다면?
- ➡ 문제점(결론):
학습이 잘못되어 오답을 도출할 수 있음.
- ➡ CutMix 사용!
- ➡ cutmix를 통해 희석되면 긍정적인 영향을 줄 것을 기대.

Algorithm



효율적인 학습 efficient net

- ➡ 600*800 정도 크기의 데이터
- ➡ 부족한 시간... 그래도 높은 정확도는 얻고싶은걸...
- ➡ efficient net 최소 파라미터로 높은 정확도 달성 연구
- ➡ efficientnet B4를 사용하고 ImageNet에서 같은 이미지 분류 문제에서 높은 성적을 내는 NoisyStudent를 전이학습하자!
- ➡ 전처리 과정에서 image net 이미지의 평균과 분산을 이용한 정규화

input size

512 X 512 X 3

image

label (one hot)

Efficient net B4 (weight = noisy student)

callback -> lr scheduler

layer마다 Batchnormalization

optimizer = Adam

loss = CategoricalCrossentropy

dropout : 0.45

Global AveragePooling2D()

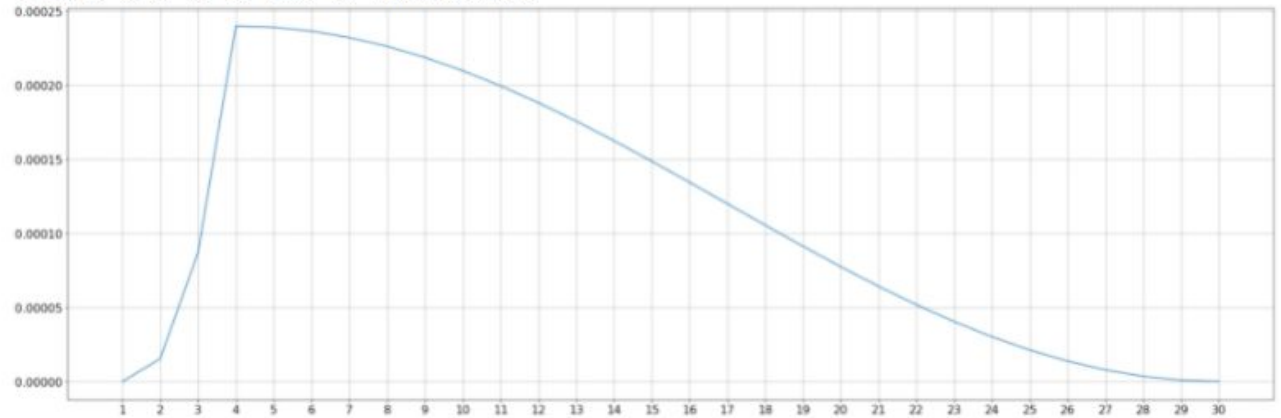
dropout : 0.4

Dense(5, activation = 'softmax', dtype = tf.float32)



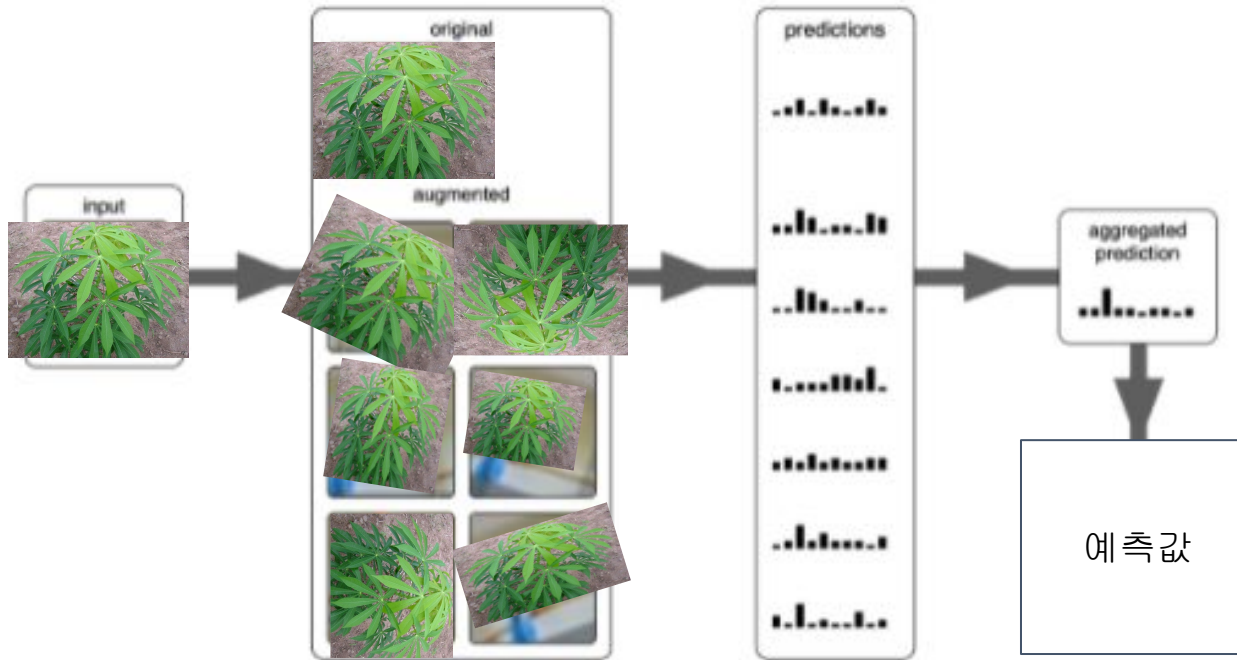
argmax

init lr 1.0e-08 to 2.4e-04 final 1.0e-08



Learning rate Scheduler

- ➡ Parameter가 전이학습을 통해 설정 되는데 image net에서 학습되었기 때문에 기존에 학습하던 것과 다른 카사바 데이터를 학습할때 초기부터 큰 Learning rate(LR)는 학습의 불안정을 초래할 수 있다고 보았다.
- ➡ 초기 4 Epoch 동안 Warm-up 시키는 것이 해결책이라고 보았다.
- ➡ 이후, 완만하게 학습률을 낮춰 최적점을 탐색하는 형태로 Learning rate scheduler를 설정해 주었다.

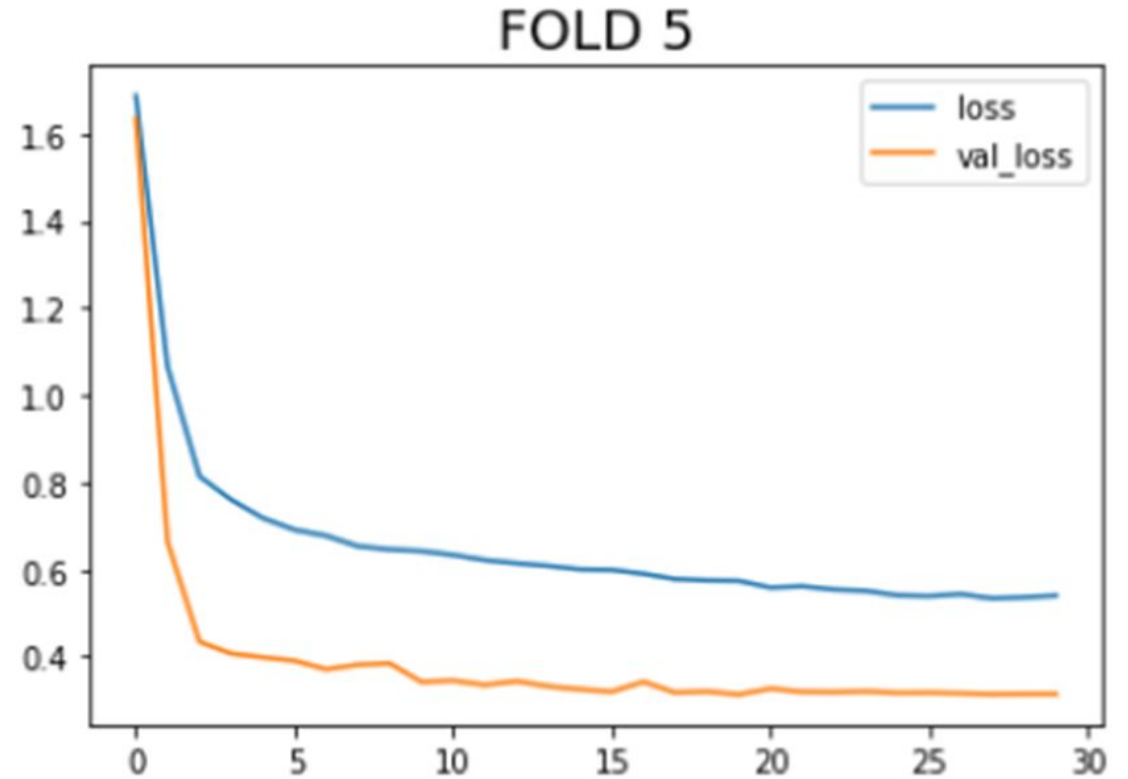
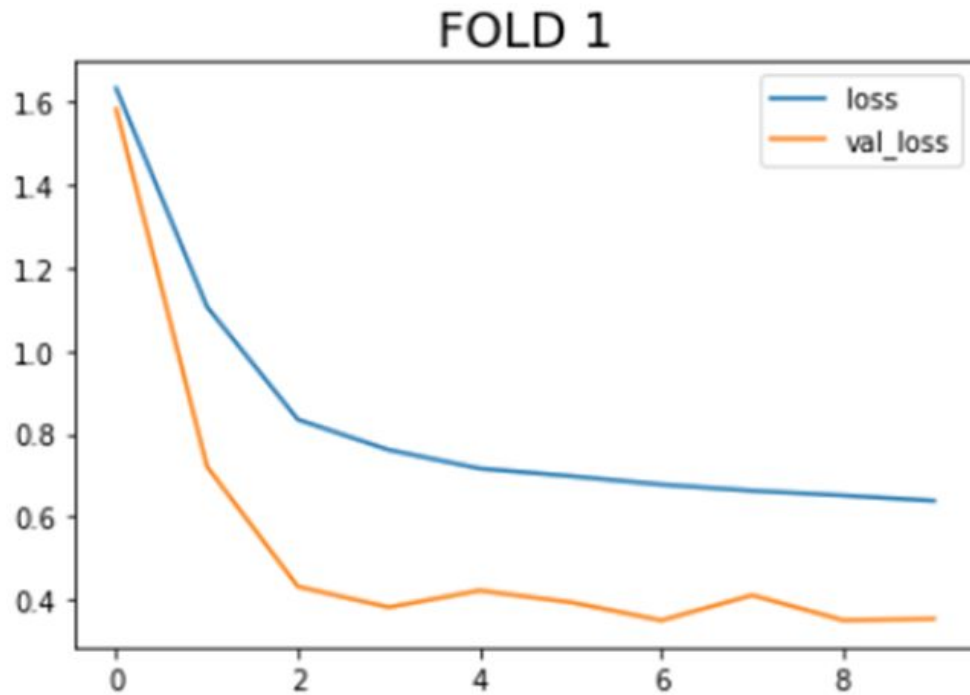


TTA(Test Time Augmentation)

- ➡ 모델에 한가지 이미지를 주는 것보다 여러가지 변형된 이미지를 주어 평가하면 발생하는 오차가 작아질 것이므로 **Test** 데이터의 오차를 줄일 수 있고 모델이 가지고 있을 편향된 학습결과 또한 벗어날 수 있을 것이라고 기대함
- ➡ 뒤집기, 자르기, **transpose**, **resize** 등의 가벼운 증강을 적용, 8번의 예측을 합하여 각 모델당 최종 결과를 냄
5fold * 8TTA
- ➡ LB에 큰 차이는 없었지만 이 모델을 활용할 때 편향된 데이터 등에서 보다 나은 성능을 낼 것으로 기대함

Loss

validation loss가 계속 train loss가 낮은 것을 확인할 수 있음.
→오버피팅이 발생하지 않음.



OOF validation accuracy: 0.899370527267456


Part 3 실행 방법

Add Data Upload X

Datasets **Competition Data** **Notebook Output Files** Sort by **Relevance** Q

추가1 Featured Getting started Entered

Cassava Leaf Disease Classification
Identify the type of disease present on a Cassava Leaf image
Research · Code Competition · a year ago \$18,000 3,900 teams Add

추가2  **Cassava Leaf Disease TFRecords 600x600**
Mark Wijkhuisen updated a year ago (Version 12) food Other 15 15.71GB 0 Other 515 Add

학습된 모델
불러오기

(ipynb)파일 다운 받아서 캐글에서 오픈. TPU 설정




<https://github.com/chaehuseon/CAU-PE/blob/main/notebook472516c786.ipynb>

데이터 링크

추가1,2 => **cass**라고 검색했을때 다음과 같은 데이터 추가
학습된 모델 불러오기 : **search** 부분에 붙여넣기 해서
모델 추가

Data + **Add data** ^

Input

-  cassava-leaf-disease-classific...
-  cassava-leaf-disease-tfrecords...
-  wqdasdasfwadasasd