대회를 진행하며

0. Overview
(0) 강사 소개

김현우 (Hyeonwoo Kim)

경력:

2020년 8월 ~ : 카이스트 석사과정

2014년 3월 ~ 2020년 8월 : 한양대학교 산업공학과 학사과정

연구분야: Semantic Segmentation

이메일: kimhyeonwoo2431@gmail.com

깃허브 : https://github.com/choco9966/

블로그 : https://eda-ai-lab.tistory.com/

발표자료 : https://github.com/choco9966/Cassava-Leaf-Disease-Classification

0. Overview
(0) 강사 소개

Hyun woo kim

Student at Kaist

Bucheon-si, Gyeonggi-do, South Korea

Joined 3 years ago · last seen in the past day

https://eda-ai-lab.tistory.com/

Followers 147

Following 48

Competitions Expert

Home | Competitions (23) | Datasets (6) | Code (55) | Discussion (454) | Followers (147) | Notifications | Account | **Edit Profile**

Competitions Expert		Datasets Contributor		Notebooks Expert		Discussion Expert	
Current Rank	Highest Rank	Unranked		Current Rank	Highest Rank	Current Rank	Highest Rank
1726 of 158,968	767			506 of 168,489	51	750 of 195,556	88
0	7	0	0	3	5	5	3
2		0		12		124	
IEEE-CIS Fraud ... · 2 years ago · Top 1%	25th of 6351	T Academy Rec... · 8 months ago	7 votes	Santander Light... · 2 years ago	163 votes	Collect all discu... · 3 years ago	109 votes
Santander Cust... · 2 years ago · Top 1%	38th of 8751	T Academy Rec... · 5 months ago	4 votes	PUBG Data Des... · 3 years ago	122 votes	Collect discussi... · 3 years ago	33 votes
Microsoft Malw... · 2 years ago · Top 2%	40th of 2410	ranzcr weight v3 · 4 months ago	0 votes	House Price Pre... · 2 years ago	102 votes	Public 19st Solu... · 2 years ago	20 votes

0. Overview
(0) 강사 소개

2020	**1th Place**, Prediction of the period of sale of used cars	*KB Capital*
2020	**2nd Place**, Winter tire Demand Forecast in Germany	*Hankook Tire*
2020	**2nd Place**, Detect Abnormal Transactions in Mobile Environments	*Pay Letter*
2020	**2nd Place**, AI Based Plant Segmentation with Aerial Photography	*KOFPI*
2020	**3rd Place**, Game Bot Detection in AION	*kisa*
2020	**2nd Place**, Korea Landmark Classification	
2020	**5th Place**, Nowcast Prediction	*Korea Hydro Nuclear Power Co.*
2019	**4th Place**, Fire Prediction in Gimhae	*LH Co.*
2019	**3rd Place**, Prediction Jeju Bus Passengers	*Jeju Technopark*
2019	**5th Place**, Automotive network intrusion detection	*Kisa*
2019	**4th Place**, Brunch Recommendation Competition	*Kakao*
2019	**1st Place**, Prediction the real price of apartments	*Zigbang*
2018	**1st Place**, Data Visualization Challenge of Credit Card transaction	*Banksalad*
2018	**1st Place**, Bigcontest2018	*Shinhan Bank*

0. Overview

(1) Description

아프리카 2위 탄수화물 공급처인 카사바는 열악한 환경에도 견딜 수 있어 소지주 농가들이 재배하는 핵심 식량안보 작물이다. 사하라 이남 아프리카의 가정 농가의 80% 이상이 이 녹말뿌리를 재배하지만 바이러스성 질병은 수확량이 떨어지는 주요 원인이다. 데이터 과학의 도움으로, 치료될 수 있도록 흔한 질병을 식별하는 것이 가능할 수도 있다.

기존의 질병 감지 방법은 농부들이 식물에 대한 육안 검사와 진단을 위해 정부 지원 농업 전문가의 도움을 요청해야 한다. 이것은 노동집약적이고, 공급이 적고, 비용이 많이 드는 것으로 인해 어려움을 겪는다. 아프리카 농부들은 저대역폭의 이동형 카메라에만 접근할 수 있기 때문에, 추가적인 도전 과제로서, 농부들을 위한 효과적인 솔루션은 상당한 제약조건 하에서 잘 수행되어야 한다.

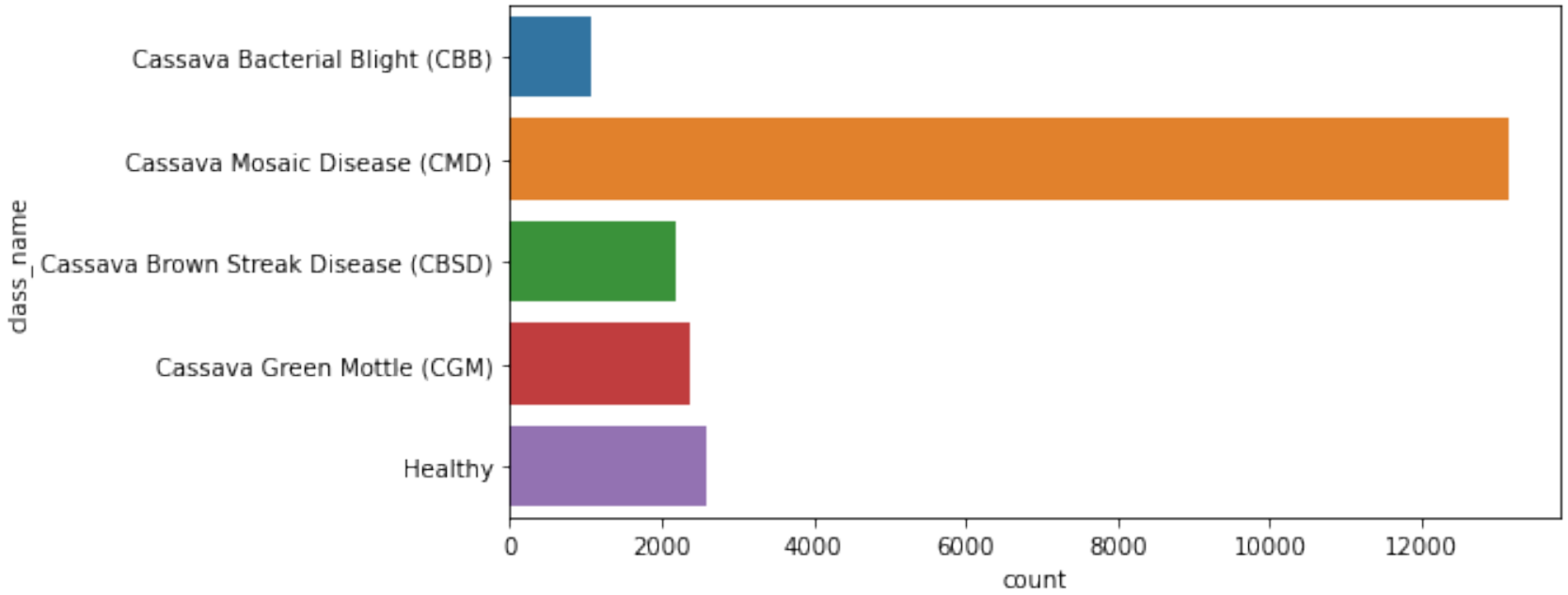
이 대회에서는 우간다에서 정기적으로 설문 조사를 하는 동안 수집된 21,367개의 레이블링된 이미지의 데이터 세트를 소개한다. 대부분의 이미지는 농부들이 그들의 정원을 사진 찍는 것으로부터 크라우드소싱되었으며 캄팔라 마케레 대학의 AI 연구소와 협력하여 국립 작물자원연구소(NaCRRI)의 전문가들이 주석을 달았다. 이것은 농부들이 실제 생활에서 진단해야 할 것을 가장 현실적으로 나타내는 형식이다.

각 카사바 이미지를 네 가지 질병 범주 또는 건강한 잎을 나타내는 다섯 번째 범주로 분류하는 것이 과제입니다. 여러분의 도움으로, 농부들은 질병이 있는 식물을 빠르게 확인할 수 있고, 잠재적으로 그들이 회복할 수 없는 피해를 입히기 전에 그들의 농작물을 구할 수 있을 것입니다.

0. Overview
(2) Dataset

1. Data Exploratory Analysis

- (1) Class Imbalance



- 5개의 Class가 가지고 있는 Imbalance
  - CMD의 비율이 압도적으로 높고 다른 클래스의 비율은 낮음

- CBB : 세균성 질병
- CMD : 모자이크 질병
- CBSD : 갈색 줄무늬병
- CGM : 녹색 반점

⇨ 문제는 눈으로 봐도 전혀 구분이 안감

1. Data Exploratory Analysis

- (2) Zoom

식물의 사진이 Zoom 형식으로 된 경우도 있지만, 전체에 걸쳐져서 나오는 경우도 있음

[확대된 경우]

[멀리서 찍은 경우]

- 사진마다 어떤 식으로 찍은지에 대한 특성이 다름
  - 케이스1. 식물의 잎을 중심으로 포커스해서 사진을 찍은 경우
  - 케이스2. 전체 식물의 이미지가 나오도록 멀리서 찍은 경우

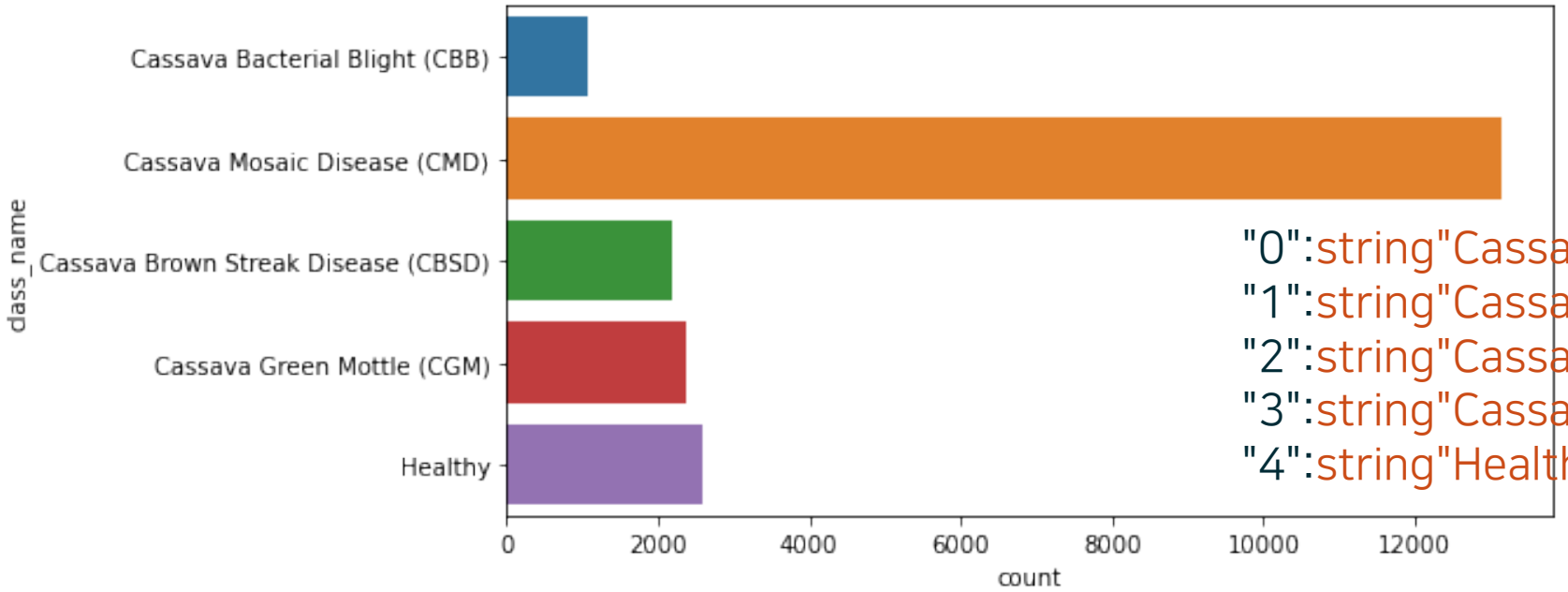
1. Data Exploratory Analysis

- (3) 테스트 이미지의 라벨 비율

Submission을 통해서 테스트 이미지의 라벨 비율을 확인한 결과

- Train과 Test의 Class 비율은 매우 유사
  - *단, 공개 리더보드에 한해서 나온 결과임

Class 0	Class 1	Class 2	Class 3	Class 4
4.8%	10.6%	10.3%	60.2%	14.1%



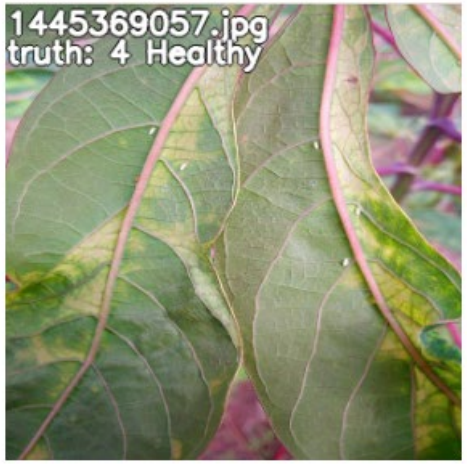
"0":string"Cassava Bacterial Blight (CBB)"
"1":string"Cassava Brown Streak Disease (CBSD)"
"2":string"Cassava Green Mottle (CGM)"
"3":string"Cassava Mosaic Disease (CMD)"
"4":string"Healthy"

1. Data Exploratory Analysis

- (4) 라벨에 노이즈가 발생한 경우

데이터의 Label이 잘못된 경우가 많음

- 육안으로 확인한 결과 (잎의 반점부분이 이상해보이지만 Healthy 여부 맞는거로 언급됨..?) <- 판단하기 어려움



Cassava Brown Streak Disease (CBSD)

Cassava Bacterial Blight (CBB)

1. Data Exploratory Analysis

- (4) 라벨에 노이즈가 발생한 경우

데이터의 Label이 잘못된 경우가 많음

- CleanLab을 통해서 확인해본 결과

[Clean Label]

[Noise Label]

참고자료 : https://www.kaggle.com/telljoy/noisy-label-eda-with-cleanlab

1. Data Exploratory Analysis

- (5) 이미지가 중복이 발생한 경우

데이터의 이미지가 중복된 경우가 있음

[이미지가 중복된 경우]

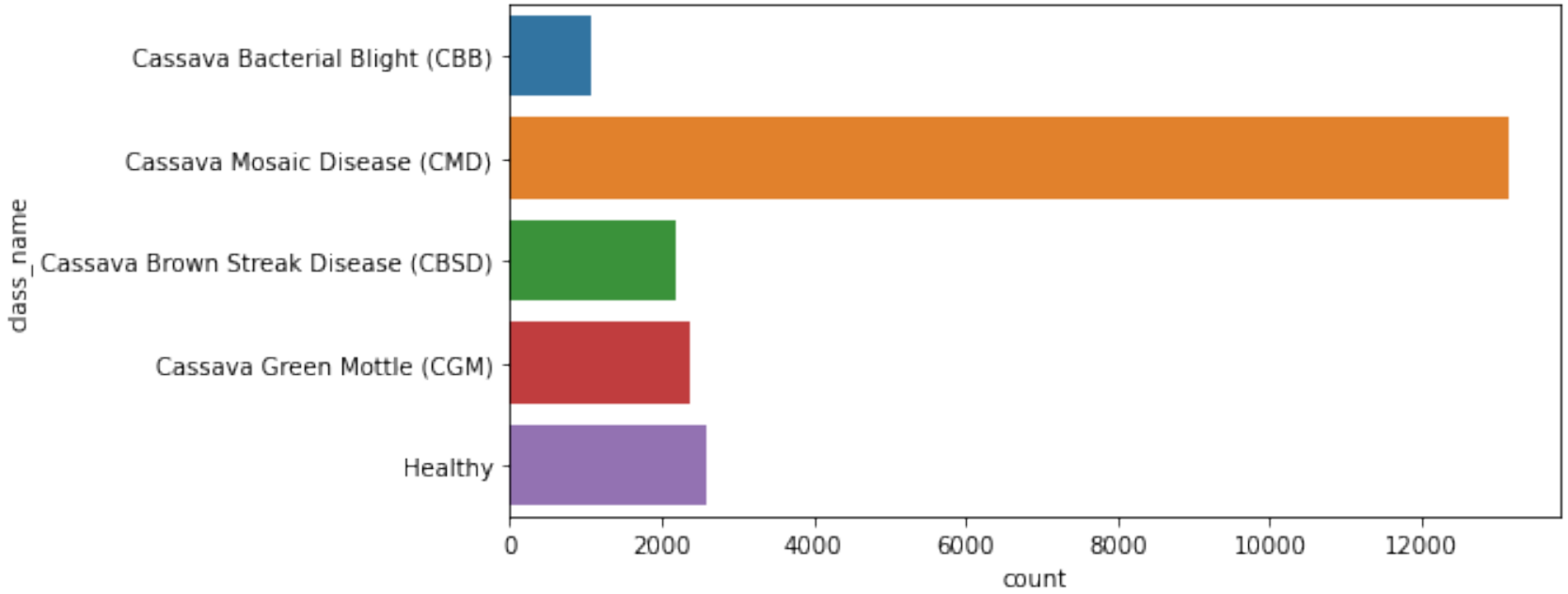
[중복 + 라벨 노이즈인 경우]

참고자료 : https://www.kaggle.com/graf10a/cldc-image-duplicates-with-dbscan

2. Experiments

(1) Class Imbalance

시도1. 외부 데이터를 이용한 클래스 균형 맞추기

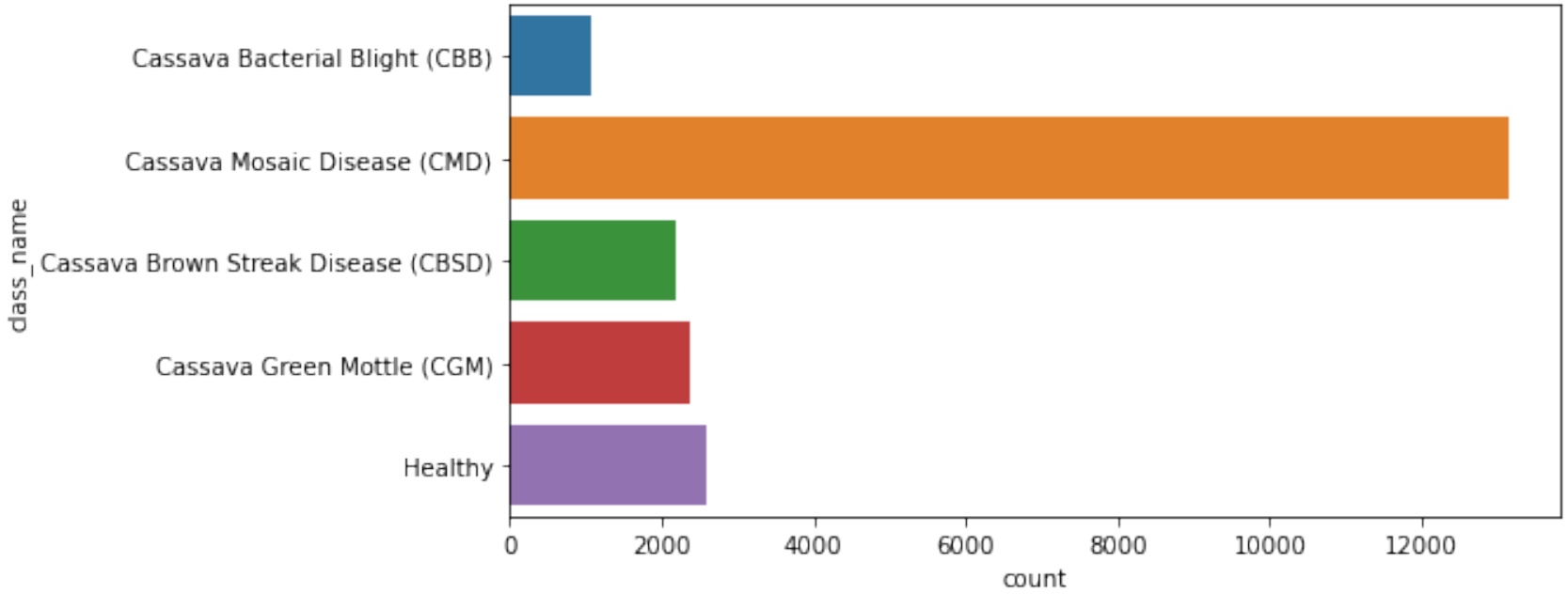


- 외부 데이터를 이용해서 CBB, CBSD, CGM, Healthy 기반의 클래스들만을 가지고 결합해서 Balance를 맞춰주는 방법
  - 2019년도 Cassava 경진대회 데이터 셋
  - In Class Cassava 경진대회 데이터 셋

- CV 는 소폭 상승, LB 는 소폭 하락
  -> 원인 : 다른 데이터 셋에도 Label Noise가 존재

2. Experiments
(1) Class Imbalance

시도2. Focal Loss를 이용해서 클래스 불균등에 강한 로스 사용



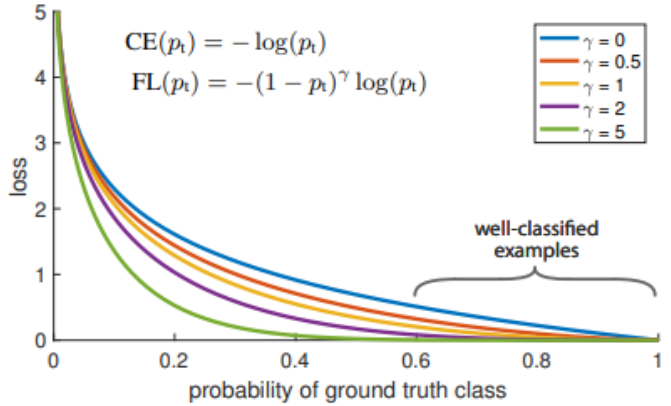


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

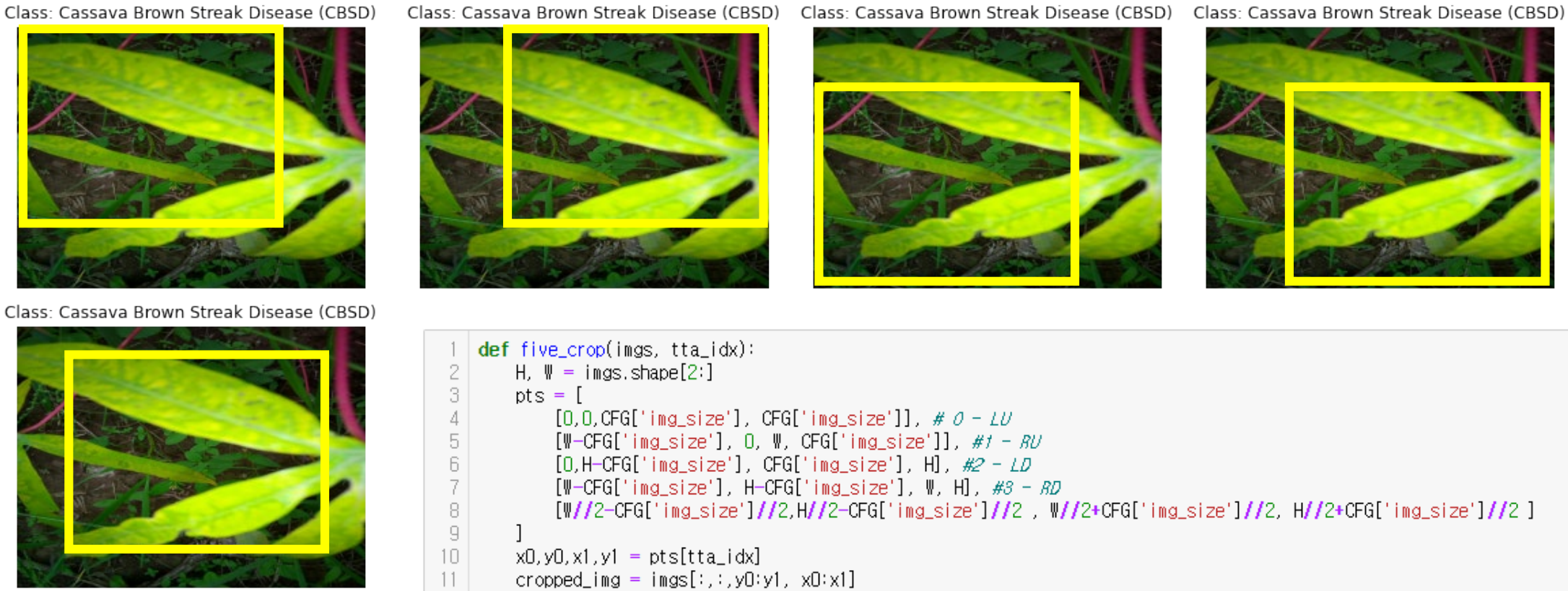
- Loss 함수를 Focal Loss를 사용해서 Class Imbalance에 대해서도 강한 로스를 줬지만, 성능이 오히려 하락했음
  -> 원인 : 데이터에 많이 존재하는 Noise와 연관

참고자료 : Focal Loss for Dense Object Detection

2. Experiments

(2) Zoom

시도1. 5개의 영역에 Crop을 적용해서 전체 영역을 학습할 수 있는 Augmentation 시도




```python
def five_crop(imgs, tta_idx):
    H, W = imgs.shape[2:]
    pts = [
        [0,0,CFG['img_size'], CFG['img_size']], # 0 - LU
        [W-CFG['img_size'], 0, W, CFG['img_size']], #1 - RU
        [0,H-CFG['img_size'], CFG['img_size'], H], #2 - LD
        [W-CFG['img_size'], H-CFG['img_size'], W, H], #3 - RD
        [W//2-CFG['img_size']//2,H//2-CFG['img_size']//2 , W//2+CFG['img_size']//2, H//2+CFG['img_size']//2 ]
    ]
    x0,y0,x1,y1 = pts[tta_idx]
    cropped_img = imgs[:,:,y0:y1, x0:x1]
    return cropped_img
```

- 성능 하락
  -> 원인 : Zoom이 되어있는 데이터를 확대하는 것은 의미 x

2. Experiments

(2) Zoom

시도2. Inference 시의 TTA에서 여러가지의 영역을 볼 수 있도록 사용

```
def get_inference_transforms():
    return Compose([
            OneOf([
                Resize(CFG['img_size'], CFG['img_size'], p=1.),
                CenterCrop(CFG['img_size'], CFG['img_size'], p=1.),
                RandomResizedCrop(CFG['img_size'], CFG['img_size'], p=1.)
            ], p=1.),
            Transpose(p=0.5),
            HorizontalFlip(p=0.5),
            VerticalFlip(p=0.5),
            Resize(CFG['img_size'], CFG['img_size']),
            Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225], max_pixel_value=255.0, p=1.0),
            ToTensorV2(p=1.0),
        ], p=1.)
```

- 902 -> 905로 향상
  - Resize, CenterCrop, RandomResize를 여러 조합으로 사용했지만 OneOf으로 랜덤하게 묶는게 Best였음

2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도1. 라벨 스무딩 적용

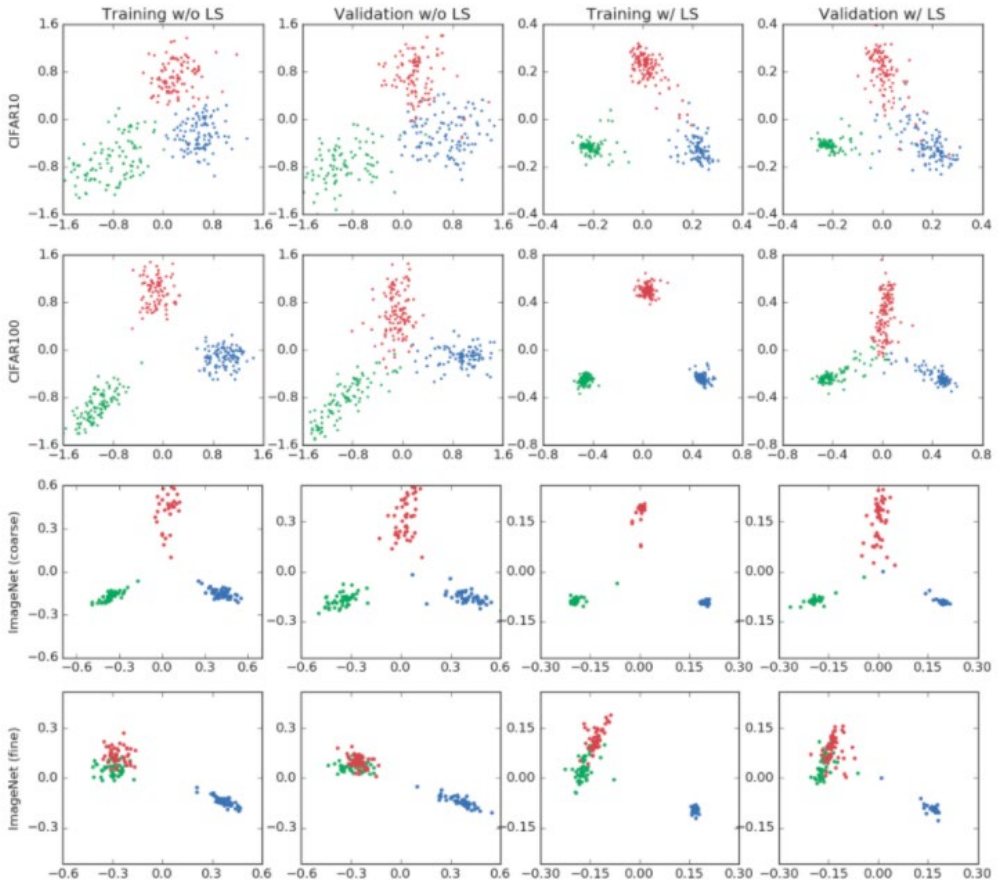
Label Smoothing

$$\mathbf{y}_k^{\mathrm{LS}} = \mathbf{y}_k(1-\alpha) + \alpha/K \qquad H(\boldsymbol{y}, \boldsymbol{p}) = \sum_{k=1}^{K} -y_k^{LS} \log(p_k)$$

$[0, 1, 0, 0]$

$[0.025, 0.925, 0.025, 0.025]$

라벨을 One-Hot Encoding이 아닌 스무딩을 적용한 Soft Target으로 변경



- alpha가 0.2~0.3일때 점수가 가장 좋았음
  - 0.1 미만으로 낮게 주었을때는 오히려 점수가 하락하는 문제가 있었음

- 0.1 / 0.2 / 0.25 / 0.3 4가지 하이퍼파라미터에 대해 LB/CV를 조사
  - 0.10 : 899 -> 897 /// 0.20 : 902
  - 0.25 : 902 /// 0.30 : 902
  - 0.1 < 0.25 < 0.3 < 0.2 순으로 점수가 높았음

2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도2. 노이즈가 강한 로스함수 사용

사용한 로스 목록

- Focal Loss
- Focal Cosine Loss
- Symmetric Cross Entropy Loss
- Taylor Cross Entropy Loss
- 관련 코드 : https://www.kaggle.com/piantic/train-cassava-starter-using-various-loss-funcs

전부 성능향상에 도움이 되지는 않았음

2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도3. OOF 및 CleanLab 기반으로 제거 및 라벨을 수정

- CleanLab 패키지를 통해 노이즈 제거, 변경 -> 점수 하락
- OOF로 노이즈 50개 제거, 변경 -> ViT는 상승, 다른 모델은 하락
- 2019 년도의 노이즈를 OOF로 제거하고 2019+2020년도 모델 사용 -> 점수 하락

2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도3. OOF 및 CleanLab 기반으로 제거 및 라벨을 수정

```python
from cleanlab.pruning import get_noise_indices

ordered_label_errors = get_noise_indices(
    s=numpy_array_of_noisy_labels,
    psx=numpy_array_of_predicted_probabilities,
    sorted_index_method='normalized_margin', # Orders label errors
)
```

[CleanLab]

출처 : https://github.com/cgnorthcutt/cleanlab

2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도3. OOF 및 CleanLab 기반으로 제거 및 라벨을 수정

2020년도의 데이터의 Noise를 KFold OOF 기반으로 제거

이미지1 예측 값 : [0.04, **0.9**, 0.05, 0.005, 0.005]
이미지1 실제 값 : [**1**, 0, 0, 0, 0]

	Train	Valid			
Split 1	Fold1	Fold2	Fold3	Fold4	Fold5
Split 2	Fold1	**Fold2**	Fold3	Fold4	Fold5
Split 3	Fold1	Fold2	**Fold3**	Fold4	Fold5
Split 4	Fold1	Fold2	Fold3	**Fold4**	Fold5
Split 5	Fold1	Fold2	Fold3	Fold4	**Fold5**

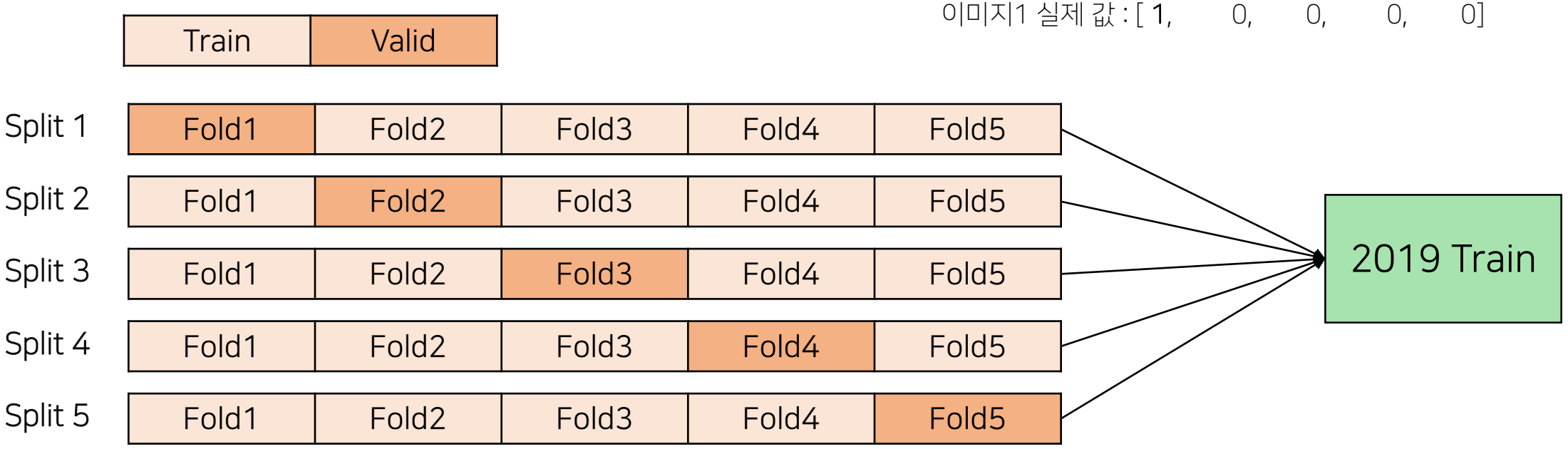
2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도3. OOF 및 CleanLab 기반으로 제거 및 라벨을 수정

2019년도의 데이터의 Noise를 KFold OOF 기반으로 제거

이미지1 예측 값 : [0.04, **0.9**, 0.05, 0.005, 0.005]
이미지1 실제 값 : [**1**, 0, 0, 0, 0]

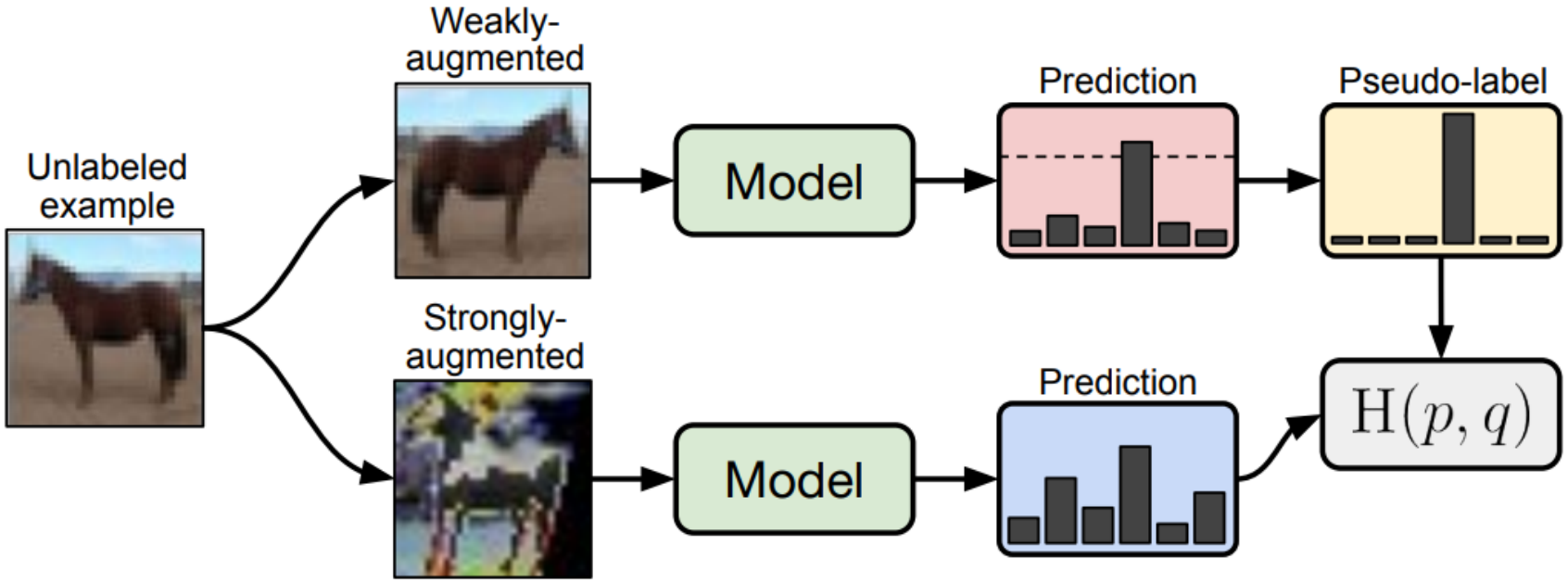


2. Experiments

(4) 라벨에 노이즈가 발생한 경우

시도4. FixMix인 Semi-Supervised Model을 이용

- 2019년도 데이터 셋에도 노이즈가 있는 문제를 이용해서, 2019년도를 UnLabeled 데이터 셋으로 활용하고 2020년도를 라벨이 있는 데이터 셋으로 활용하여 모델을 만듬



- 대회 진행시에는 성능이 낮아서 사용 안했지만, Private LeaderBoard 점수가 가장 높았던 단일 모델이었음

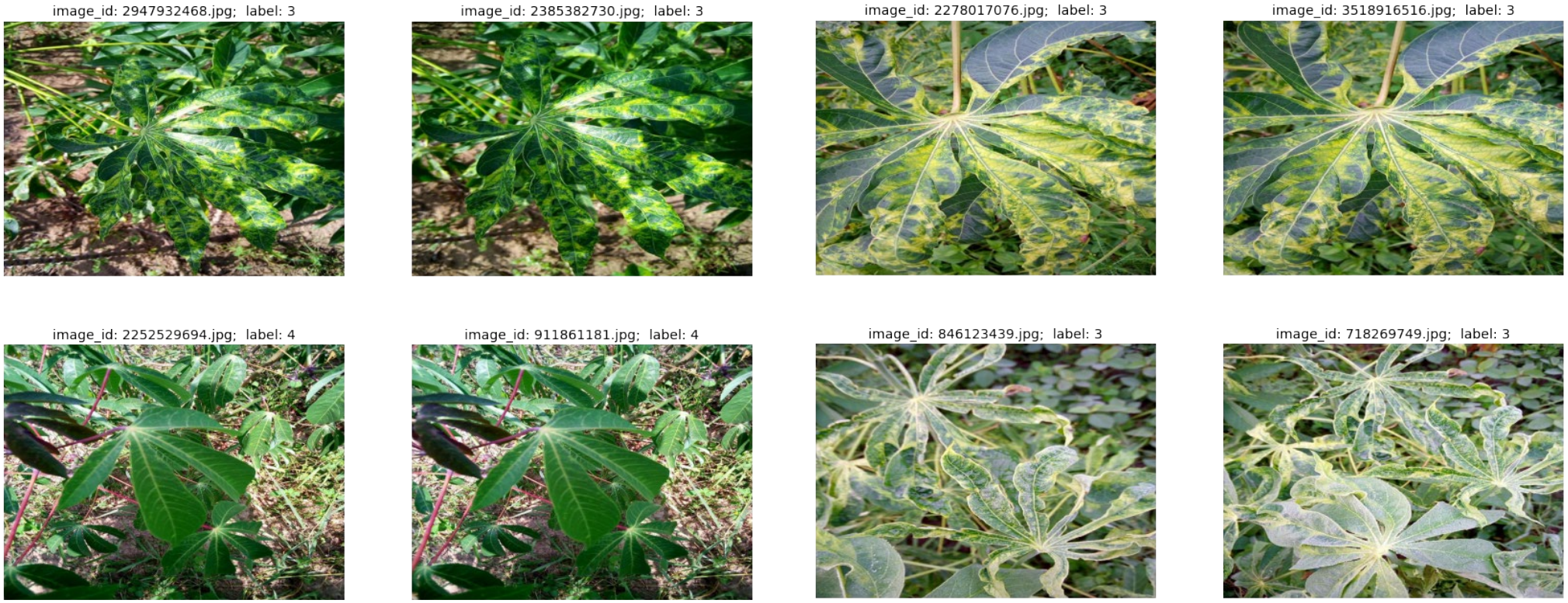
FM2019_fast_thr085_bs9mu2_7ep_CusSwa4 FM2019_fast_thr085_bs9mu2_7ep_CusSwa4 (version 13 days ago by hihunjin From Notebook [FM2019_fast_thr085_bs9mu2_7ep_CusSwa4]	Succeeded	0.8987	0.9005

2. Experiments

(5) 이미지가 중복이 발생한 경우

DBSCAN을 이용한 유사한 이미지 탐색

- Pretrained된 모델에 Global Average Pooling을 이용해서 간단한 2D Embedding을 수행
- Embedding된 벡터들에 DBSCAN을 적용해서 유사도 계산
- 유사도를 기반으로 동일한 이미지 제거 (총 5장)



참고자료 : https://www.kaggle.com/graf10a/cldc-image-duplicates-with-dbscan

2. Experiments

(6) Augmentation 실험

적절한 Augmentation 기법 찾기

Name	테스트 Aug	inference score
aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.881
Augmentation zero(resize)	Randomresizedcrop(512)	0.877
aug1 + Blur	horizontalflip(0.5) Randomresizedcrop(512) Normalize blur	0.888
aug1 + ROTATE	horizontalflip(0.5) Randomresizedcrop(512) Normalize rotate	0.886
aug1 + CLAHE	horizontalflip(0.5) Randomresizedcrop(512) Normalize clahe	0.883
aug1 + brightcontrast	horizontalflip(0.5) Randomresizedcrop(512) Normalize RandomBrightnessContrast	0.886
aug1 + Sharpen	horizontalflip(0.5) Randomresizedcrop(512) Normalize Sharpen	0.886
aug1 + Blur + brightcontrast	horizontalflip(0.5) Randomresizedcrop(512) Normalize RandomBrightnessContrast blur	0.885
Aug1 + Blur + gridmask		0.882
Aug1 + Blur + transpose		0.891
Aug1 + Blur + transpose + vertical		0.888
heroseo Aug	horizontalflip(0.5) Randomresizedcrop(512) Normalize ShiftScaleRotate verticalflip Transpose	0.890
Aug1 + Blur + transpose + coarsedropout		0.891
Aug1 + Blur + transpose + ShiftScaleRotate		0.890
ViT(base+patch16) + aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.886
ViT(base+patch32) + aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.875

2. Experiments

(7) Optimizer 실험

여러가지 Optimizer 실험

- Adam
- AdamW
- AdamP
- RAdam
- Lookahead + Optimizer
- Gradual WarmUp

- Optimizer : Adam to Adamp (EfficientNet has reduced performance)
  - VIT : 901 to 902
  - RegNetY40 : 904 to 905
  - Eff : 905 to 904

2. Experiments

(8) Model 실험

다양한 Model 실험

- EfficientNetB0~B7
- ResNext, RegNetY
- NFNet
- BotNet
- DeiT, ViT
- Distillation
- Fixmatch

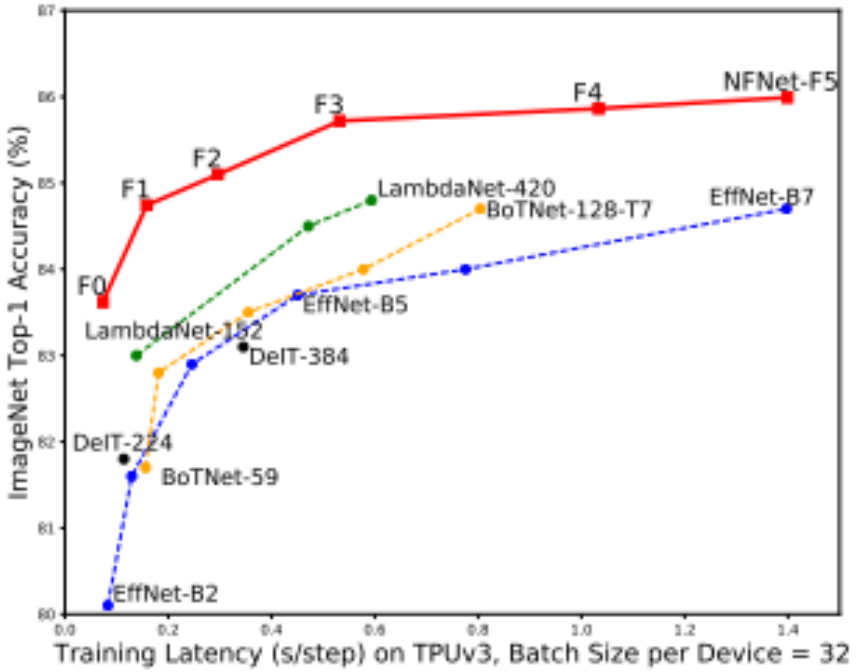
	Public LeaderBoard	Private LeaderBoard
EfficientNetB4 2019 + 2020	0.9027	0.8963
EfficientNetB4 2020	0.9050	0.8960
RegNetY40 2019 + 2020	0.9045	0.8957
RegNetY40 2020	0.9052	0.8957
RegNetY80 2020	0.9030	0.8978
NFNet	0.8993	0.8960
ViT 2019 + 2020	0.9005	0.8970
ViT 2020	0.9014	0.8943
Distillation RegNetY	0.9030	0.8946
Fixmatch	0.9005	0.8987

2. Experiments

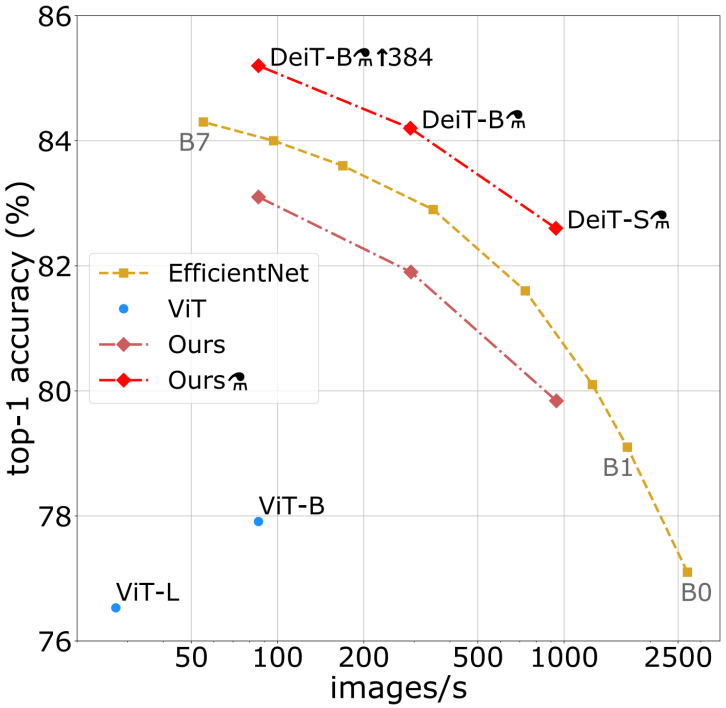
(8) Model 실험

최신 SOTA 모델들도 실험했음 (NFNet : 대회 끝나기 3일전쯤 나온 모델, Deit, ViT, Fixmatch 2020)

- NFNet
- DeiT, ViT
- Fixmatch



[NFNet]



[Deit, ViT]

2. Experiments

(9) Ensemble 실험

모델 / 시드 / TTA / SWA / Epoch Ensemble 실험

- Ensemble (Eff 905 / RegNetY 905 / VIT 902)
  - EFF(2019+2020) + EFF(2020) + EFF(SEED 720) + RegNetY(2019+2020) + RegNetY(2020) + RegNetY(Distillation) : 908
  - EFF(2019+2020) + EFF(2020) + RegNetY(2019+2020) + RegNetY(2020) : 907
  - EFF + RegNetY : 907
  - EFF + VIT : 906
  - EFF + VIT + RegNetY : 906

Avengers Assemble

Preprocessing

- ① 2019+2020 Data
- ② Only 2020 Data
- ③ SEED 719
- ④ SEED 720
- ⑤ Adam
- ⑥ AdamP
- ⑦ Duplicated Image
- ⑧ Relabeling
- ⑨ SWA
- ⑩ No Fold

- EfficientNetB4+②③⑤⑨
- EfficientNetB4+②④⑤⑨
- EfficientNetB4+①③⑤⑦⑨
- RegNetY_40+②③⑥⑨
- RegNetY_40+①③⑥⑦⑨
- RegNetY_40+②③⑥⑨⑩
- VIT+②③⑥⑧⑨
- Distillation+②③⑥⑨
- NFNet+②③⑥⑨
- FixMatch+①③⑥⑨

→ 2 Times TTA

```
OneOf([Resize(p=1.),
       CenterCrop(p=1.),
       RandomResizedCrop(p=1.)
      ], p=1.),
Transpose(p=0.5),
HorizontalFlip(p=0.5),
VerticalFlip(p=0.5),
```

Public : 904 / Private : 899

170	▲173	Welkin	0.8993	28	3mo
171	▼158	Master Breaker	0.8993	203	3mo
172	▲1009	JulienS63	0.8992	8	5mo

LB Best Ensemble

Preprocessing

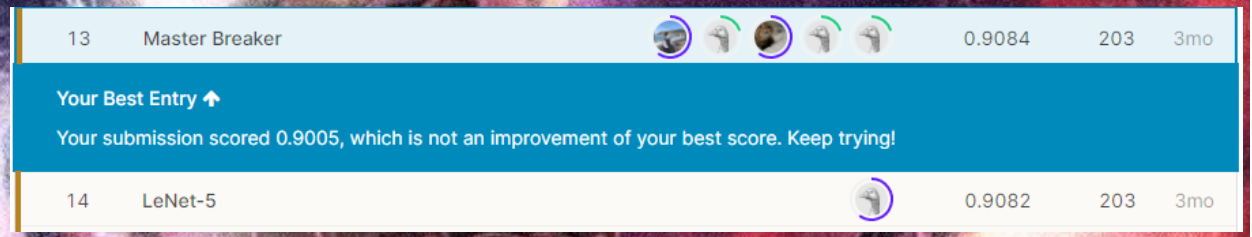
① 2019+2020 Data
② Only 2020 Data
③ SEED 719
④ SEED 720
⑤ Adam
⑥ AdamP
⑦ Duplicated Image
⑧ Relabeling
⑨ SWA
⑩ No Fold

EfficientNetB4+②③⑤⑨
EfficientNetB4+②④⑤⑨
EfficientNetB4+①③⑤⑦⑨
RegNetY_40+②③⑥⑨
RegNetY_40+①③⑥⑦⑨
RegNetY_40+②③⑥⑨⑩

→ 5 Times TTA

```
OneOf([Resize(p=1.),
       CenterCrop(p=1.),
       RandomResizedCrop(p=1.)
      ], p=1.),
Transpose(p=0.5),
HorizontalFlip(p=0.5),
VerticalFlip(p=0.5),
```

Public : 908 / Private : 896



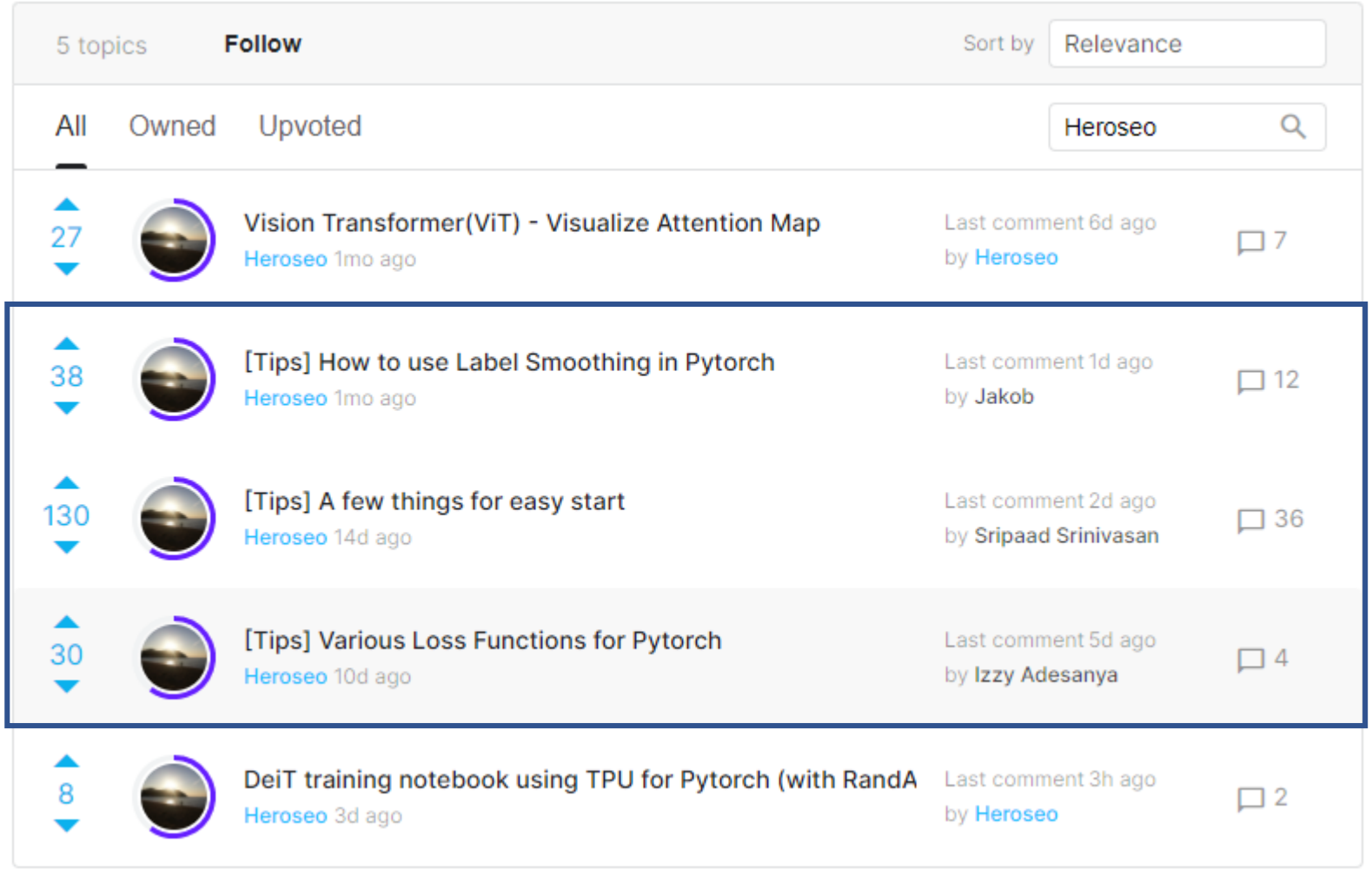
감사합니다.

그 외

3. Top Kagglers Tips

(1) Heroseo

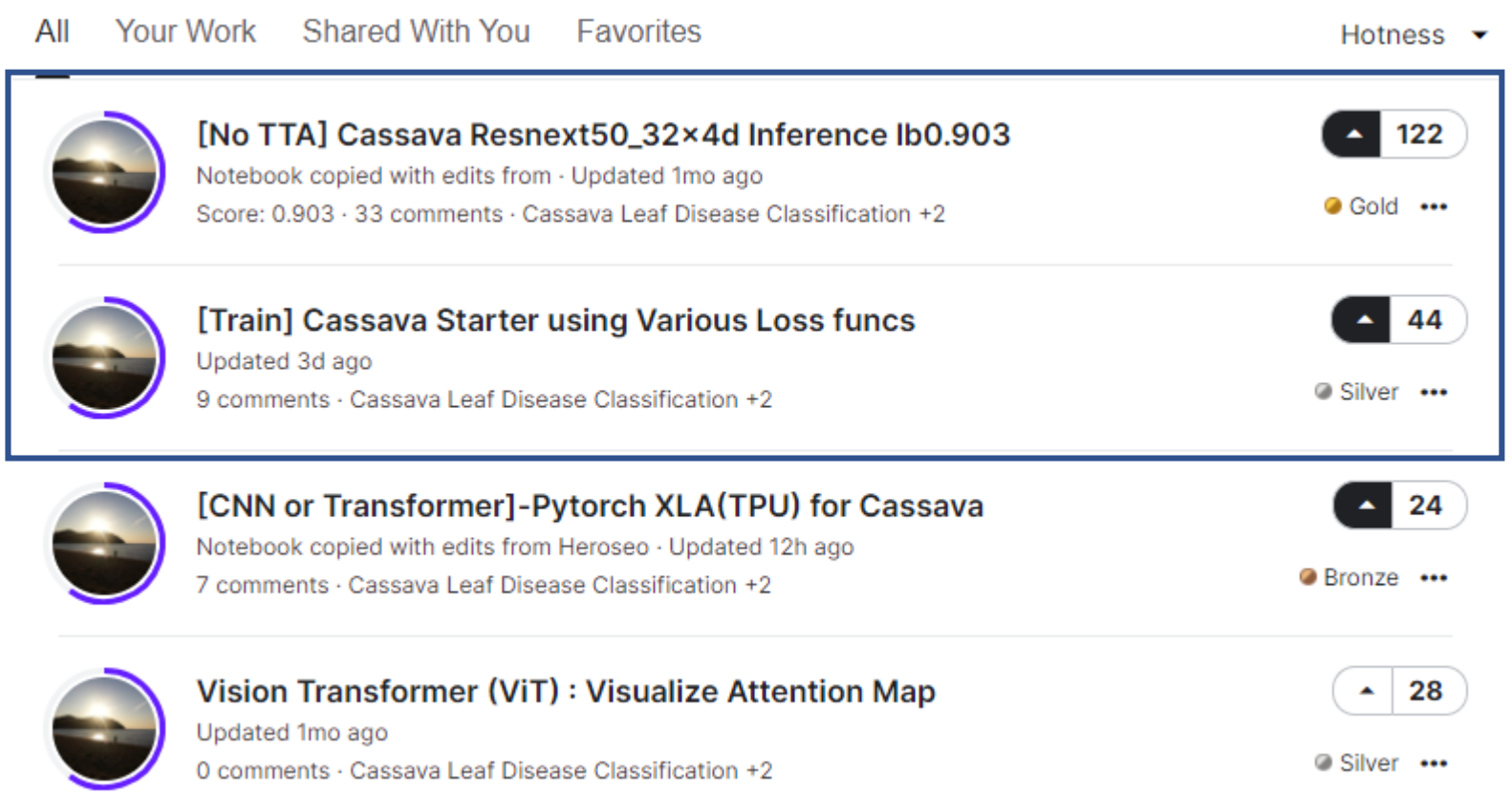
Heroseo Discussion & Notebook



3. Top Kagglers Tips

(1) Heroseo

Heroseo Discussion & Notebook



3. Top Kagglers Tips

• (1) Heroseo

Heroseo Discussion & Notebook

- Noise Label
  - Label Smoothing
  - Focal Loss
  - Bi Tempered Logistic Loss
  - Good Validation Strategy
- Others
  - Image Size (384 ~ 512)
  - Augmentation (Flips + Rotation + ETC)
  - Normalization [0.485, 0.456, 0.406], [0.229, 0.224, 0.255] – 데이터에 맞게 변경하면 조금 상승
  - Resnext50 / efficient-b3
  - Cross validation strategy – 5 fold (seed 바꾸는게 유의미함)
  - Optimizer : Adam / Radam + Lookahead
  - Relabeling and Finetuning
  - TTA (Rotation and simple augmentation) – TTA 횟수도 민감함

https://www.kaggle.com/piantic/train-cassava-starter-using-various-loss-funcs

class LabelSmoothingLoss(nn.Module):

```
def __init__(self, classes, smoothing=0.0, dim=-1):
    super(LabelSmoothingLoss, self).__init__()
    self.confidence = 1.0 - smoothing
    self.smoothing = smoothing
    self.cls = classes
    self.dim = dim
def forward(self, pred, target):
    pred = pred.log_softmax(dim=self.dim)
    with torch.no_grad():
        true_dist = torch.zeros_like(pred)
        true_dist.fill_(self.smoothing / (self.cls - 1))
        true_dist.scatter_(1, target.data.unsqueeze(1), self.confidence)
    return torch.mean(torch.sum(-true_dist * pred, dim=self.dim))
```

참고자료 : https://www.kaggle.com/c/cassava-disease/discussion/94102

4. Similar Competition Solution

(1) Plant Pathology 2020 – FGVC7

1st Solution

- 같은 이미지인데 라벨이 다르게 분류된 경우가 있었음
  - OOF를 방식을 통해서 검증
- Data Augmentation
  - Brightness, Contrast, Blur, Flip, Shift, Scale, Rotate
- Image Size
  - 320 x 512
- SeresnextNet50
- Label Imbalance 때매 CV를 믿는게 더 중요함 (LB를 믿지 말기 – 소수의 Class가 성능을 좌지우지함)
- 5fold + TTA

4. Similar Competition Solution

• (1) Plant Pathology 2020 – FGVC7

2nd Solution

- Pseudo Label (>= 0.95 Confidence)
- Single Fold
- Data Augmentation
  - Horizontal, Vertical Flip, Shift Scale Rotate
  - IAAEmboss, IAASharpen, IAAPiecewiseAffine, Blur
  - Random Erasing
- Back Bone
  - Pnasnet5Large (Private에서 좋았음)
  - Resnext101 (Public에서 좋았음)
- Image Size
  - 545 x 545
- Loss / Learning Rate
  - Cross Entropy Loss
  - Cosine with 30 epoch And Warmup

4. Similar Competition Solution

• (1) Plant Pathology 2020 – FGVC7

3rd Solution

- EfficientB7 with 768 x 768
- Augmentations
  - Rotation, shear, zoom, flip left/right/up/down, random Brightness
- Upsample (x5)
- Ensemble 3 concepts
  - A : Efficient B7 with Augmentation 3개 앙상블
  - B : Efficient B7 with Augmentation 2개 앙상블 (A와 B의 Augmentation 하이퍼파라미터가 다름)
  - 최종 : A + B

4. Similar Competition Solution

(1) Plant Pathology 2020 – FGVC7

4th Solution

[Working]

- EfficientNetB7 , IncepresNetV2 with 800x800
- Increasing Image Size
- noisy student weight (not imagenet)
- only train (no validation)
- Augmentation
  - random flip left, right, up, down

[Not Working]

- Random Crop, rotation
- TTA
- Saturation
- Gaussian blur
- L2 Regularization

4. Similar Competition Solution

• (1) Plant Pathology 2020 – FGVC7

5th Solution

[Working]

- Augmentation
  - VFlip, HFlip, Rotate, RandomResizedCrop
- BackBone
  - ResNet50, DenseNet121, InceptionResNetv2, EfficientNetB7 with noisy student
  - EfficientNet이 다양한 disease에서 잘 작동했음
- Loss
  - Categorical Cross Entropy + Focal Loss
- Image Size
  - 812x812 / 1024 x 1024
- Other
  - Pseudo Labeling
  - SWA
  - Label Smoothing
  - TTA
  - Oversampling

4. Similar Competition Solution

- (1) Plant Pathology 2020 – FGVC7

5th Solution

[Not Working]

- Cutmix
- Mixup
- GridMask
- BlockOut

[Tips]

- Don’t Trust LB

4. Similar Competition Solution

• (1) Plant Pathology 2020 – FGVC7

7th Solution

- TResNet
- Loss
  - Triplet Loss + Cross Entropy Loss
- squish Cropping with 8 rotation (training / inference)
- class weight
- pseudo Label
- EMA
- Ensemble (Different Resolution / Different Augmentation)

4. Similar Competition Solution

• (1) Plant Pathology 2020 – FGVC7

11th Solution (1st 제출안한 버전)

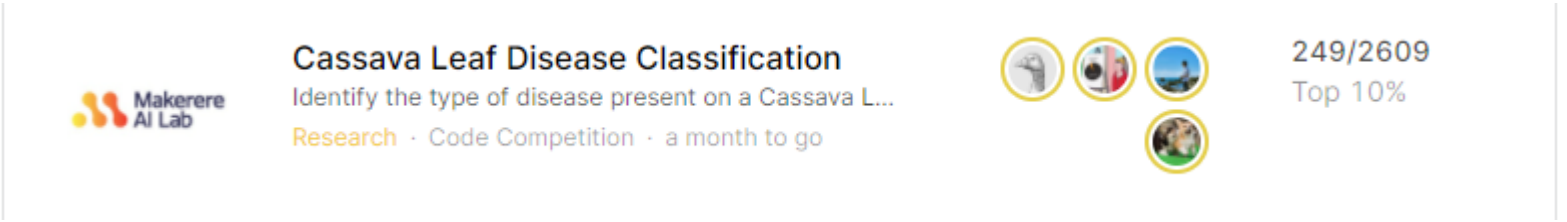
- RegNetY-32GF
  - https://github.com/facebookresearch/pycls
- Augmentation
  - Horizontal, Vertical Flip
  - ShiftScale Rotate
  - Random Brightness
  - IAAEmboss, IAASharpen, BLur
  - ElasticTransform
  - IAAPiecewiseAffine

- Pseudo Labeling
- Focal Loss
- SGD + Momentum
- Cosine Annealing
- One Fold + No TTA

4. Similar Competition Solution

(2) Cassava Disease Classification (2019)

1st Solution



- Se_resnext101
- Augmentation
  - Horizontal, Vertical Flip
  - RandomCrop, RandomRotate
- TTA with 3 times
- Pseudo Labeling
  - Extra image (Trust CV)
- Ensemble
  - Train only train image + Train with Extra image

4. Similar Competition Solution

(2) Cassava Disease Classification (2019)

2nd Solution

- Seresnext101
  - Lr : 2e-4
  - Batch size : 16
  - Input size : 448 x 448 x 3
  - Epochs : 5

- Augmentation
  - standard

4. Similar Competition Solution

• (2) Cassava Disease Classification (2019)

3rd Solution

- Se_resnext50_32x4d (only train images)
  - Lr : 1e-4
  - Batch size : 20
  - Input size : 500x500x3
- Augmentation
  - Horizontal, Vertical Flip
  - Random Crop
  - Random Erasing
  - Random Affine
- Pseudo Extra/Test Images (상승이 컸음)
- TTA
  - 10 times

```
def train_transform(size):
    return T.Compose([
        T.RandomApply([T.RandomAffine(45, shear=15)], 0.8),
        RandomResizedCropV2(size, scale=(0.6, 1.0), ratio=
(3/5, 5/3)),
        T.RandomHorizontalFlip(),
        T.RandomVerticalFlip(),
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
0.224, 0.225]),
        RandomErasing(probability=0.3, sh=0.3),
    ])
```

5. Score

(1) Cassava Disease Classification (2020)

EfficientNetB4
- v1 Baseline
- Augmentation
- StratifiedKFold

→

v2. SWA
- 10 Epoch
- [6, 7, 8, 9, 10] -> SWA

→

v3. Label Smoothing
- Parameter [0.1, 0.2, 0.3]
- 0.25 < 0.1 < 0.3 < 0.2

↓

v4. TTA [902]
- Flip + Transpose
- Test Size

→

v5. BN Freeze [낮은 903]

→

v6. AdamW + BN Freeze [높은 903]
- Adam -> AdamW

↓

v5. TTA 변경 [중간 904]
- CenterCrop + RandomResized

v7. AdamP + BN Freeze [높은 904]

5. Score

(1) Cassava Disease Classification (2020)

v5. TTA 변경 [중간 904]
- CenterCrop + RandomResized

v6. TTA 변경 [상위 906]
- CenterCrop + RandomResized + Resize

v7. 2019 추가 [903]

v7. AdamP + BN Freeze [높은 904]

New ViT

v1. ViT Baseline + BN Freeze
+ AdamP[899]

v2. ViT + 중복제거 [900]

v3. ViT
+ OOF Noise Label Change
+ Epoch 11 [902]

v4. ViT
+ 2020 [900]

New RegNetY_40

v1. Baseline [904]

v2. Baseline + Adamp [905]

v3. Baseline + Adamp + 2019 [904]

v4. Baseline + Adamp + Distilation [903]

Ensemble

v1. EFF + ViT : 낮은 906

v2. EFF + ViT : 높은 906

v3. EFF + RegNetY : 높은 907

v4. EFF x 3 + RegNetY x 3 : 높은 908

대회가 끝난 후

1 Prediction Analysis

- (1) 대회가 끝난 후 기존의 제출물 확인

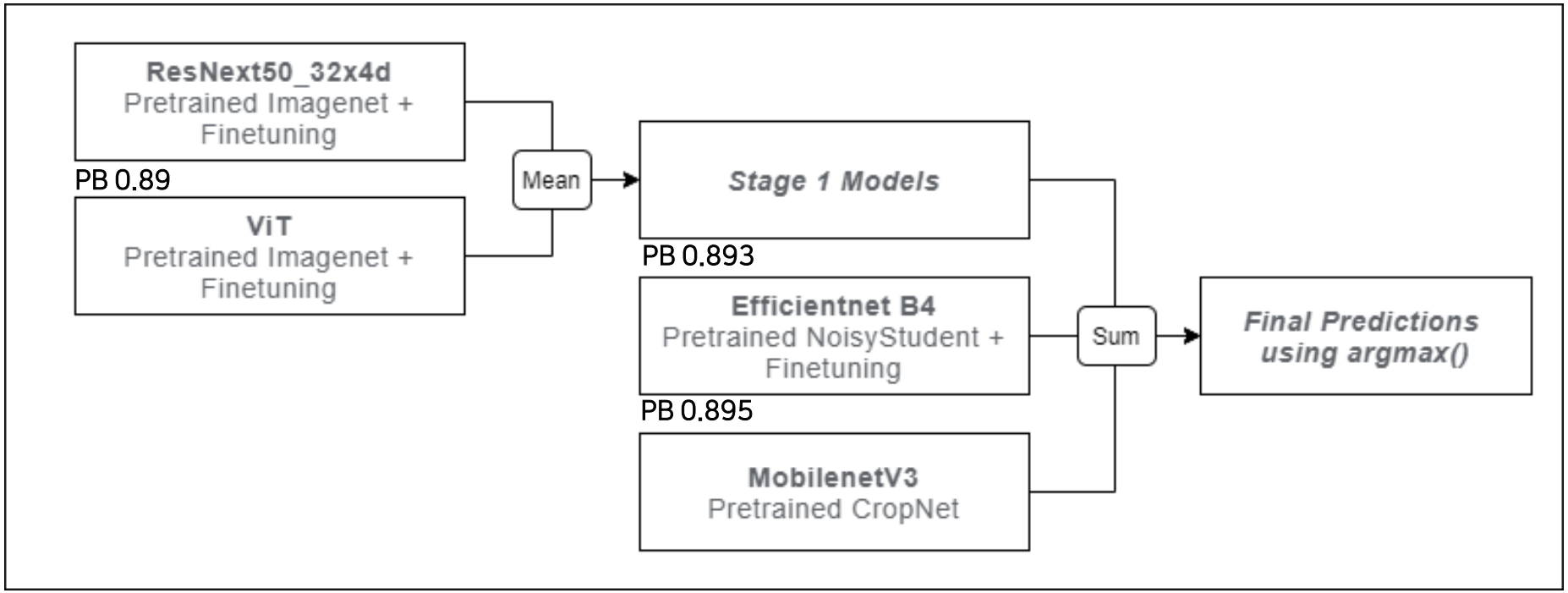
Submission and Description	Status	Private Score	Public Score	Use for Final Score
Fork of Fork of Fork of Fork of ViT 3aug train 0b5 (version 5/5) 2 months ago by Hyun woo kim From Notebook [Fork of Fork of Fork of Fork of ViT 3aug train 0b5]	Succeeded	0.8970	0.9005	☐
2019 EFF (model2) (version 3/3) 2 months ago by Hyun woo kim RegNetY Knowledge Distilation	Succeeded	0.8946	0.9030	☐
2019 EFF (model2) (version 2/3) 2 months ago by Hyun woo kim 2019 + 2020 RegNetY	Succeeded	0.8957	0.9045	☐
2019 EFF (model2) (version 1/3) 2 months ago by Hyun woo kim 2019 + 2020 EfficientNet	Succeeded	0.8963	0.9027	☐
Avengers Assemble#3 | Model10 TTA2 Avengers Assemble#3 | Model10 TTA2 (version 10/ 3 months ago by hihunjin From Notebook [Avengers Assemble#3 | Model10 TTA2]	Succeeded	0.8993	0.9048	☑
sub3. Effx3 + Regx2 (version 3/3) 3 months ago by Hyun woo kim sub3. Effx3 + Regx2 + Reg-Distill	Succeeded	0.8975	0.9082	☑

2. Solution

(1) 1st Solution Summary

Private Score 0.9132 / Public Score 0.9152

- 전략 : 다양한 모델을 다양한 pretrained weight를 가지고 학습
  - Model : EfficientNet, ResNet, ResNext, Xception, ViT, DeiT, Inception and MobileNet
  - Weight : ImageNet, NoisyStudent, Plantvillage, iNaturalist
  - Augmentation, Label Noise, Epoch 등 하이퍼 파라미터를 모델마다 모두 다르게 사용함



2. Solution

(1) 1st Solution Summary

ResNeXt (ResNext50_32x4d) with ImageNet Weight

- Image Size : 512, 512
- CrossEntropyLoss
- Learning Rate 1e-4 with ReduceLROnPlateau (validation Loss))
- Augmentation
  - Train : RandomResizedCrop, Transpose, HorizontalFlip, VerticalFlip, ShiftScaleRotate, Normalize
  - Valid, Test : Resize, Normalize
- 5Fold CV with 15 Epochs (Select Best Accuracy Epoch)

ViT (Vision Transformer Architecture) with ImageNet Weight

- Image Size : 384, 384
- Bi Tempered Logistic Loss (t1=0.8, t2=1.4) and Label Smoothing factor of 0.06
- Cosine annealing warm restarts scheduler (LR = 1e-4 / 7 [7: Warm up factor], T0= 10, Tmult= 1, eta_min=1e-4, last_epoch=-1).
- Batch Accumulation
- Augmentation
  - Train : RandomResizedCrop, Transpose, Horizontal and vertical flip, ShiftScaleRotate, HueSaturationValue, RandomBrightnessContrast, Normalization, CoarseDropout, Cutout
  - Valid : Horizontal and vertical flop, CenterCrop, Resize, Normalization
  - Test : CenterCrop, Resize, Normalization
- 5Fold CV with 10 Epochs (Select Best Accuracy Epoch)

2. Solution

(1) 1st Solution Summary

Stage 1 Models + EfficientNetB4 + CropNet

EfficientNetB4 with NoisyStudent Weights

- Image Size : 512, 512
- Drop Connect Rate 0.4
- DropOut Rate 0.5

Another argument in the model constructor worth noticing is `drop_connect_rate` which controls the dropout rate responsible for stochastic depth. This parameter serves as a toggle for extra regularization in finetuning, but does not affect loaded weights. For example, when stronger regularization is desired, try:

```
model = EfficientNetB0(weights='imagenet', drop_connect_rate=0.4)
```

```
efficientnet = EfficientNetB4(weights=weights, include_top=False,
                              input_shape=(*img_size, 3), drop_connect_rate=dropout[0])
```

```
efficientnet = efficientnet(inputs)
pooling = keras.layers.GlobalAveragePooling2D()(efficientnet)
dropout = keras.layers.Dropout(dropout[1])(pooling)
outputs = keras.layers.Dense(5, activation="softmax")(dropout)
```

- Normalization : Global mean and deviation of the 2020 Cassava dataset
- 5Fold with 20 Epochs with Early Stopping (Select Best Epoch)
- Final model was trained for 14 epochs on whole competition data set
- Augmentation
  - Train : Augmentations (Flip, Transpose, Rotate, Saturation, Contrast and Brightness and some random cropping)
  - Test (TTA) : Flip, Rotate, Transpose

CropNet (MobileNet v3) from Tensorflow Hub : Image Size 224, 244 (Not Fine Tuning)

2. Solution

(1) 1st Solution Summary

Others

- 전략 : Prior optimization of the weights of the softmax logits of each model
- 9개의 모델을 가지고 1개 사용한 경우에서부터 9개 전부 사용한 경우까지 테스트

```
combined = []
for i in range(len(columns)):
    combined.append(list(combinations(columns, i+1)))

def evaluate_ensemble(df, columns):
    return df[[*columns]].apply(lambda x: np.argmax([np.sum(v) for v in zip(*[x[c] for c in columns])]), axis=1).values

results = dict()
with tqdm(total=len(list(chain(*combined)))) as process_bar:
    for c in list(chain(*combined)):
        process_bar.update(1)
        results[c] = accuracy_score(oof_predictions_v3.label.values, evaluate_ensemble(oof_predictions_v3, c))
```

Best Combination

```
# Get top-50 combinations
{k: results[k] for k in sorted(results, key=results.get, reverse=True)[0:50]}
```

```
{('mobilenet', 'b4', 'vit_resnext'): 0.9103145300743095,
 ('mobilenet', 'b3', 'vit_resnext'): 0.9092863485535355,
```

- ResNext와 ViT를 묶은 이유는 기존 실험에서 성능이 좋아다고함

2. Solution

(1) 1st Solution Summary

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

Best Combination

Y_hat과 Y간의 차이를 줄이는 모델들의 가중치 합을 Optimization하는 테크닉을 적용

```python
kfold = KFold(n_splits=4)

yhats = considered_models.iloc[:,2:].values
y = considered_models.label.values
n_models = yhats.shape[1]

accuracy = []
for fold, (train_idx, test_idx) in enumerate(kfold.split(yhats, y)):

    print(f"Iteration {fold+1}")

    weights = np.array([1.0/n_models for _ in range(n_models)])
    bounds = [(0.0, 1.0) for _ in range(n_models)]
    minimizeargs = (np.take(yhats, train_idx, axis=0), np.take(y, train_idx, axis=0))

    def calculate_accuracy(y_true, y_pred):
        return np.average(y_true == y_pred)

    def loss_func(weights, Yhat, Y):
        w = np.mean(weights * Yhat, axis=1)
        return 1 - calculate_accuracy(Y, list(map(lambda x: np.argmax(x), w)))

    sol = differential_evolution(loss_func, bounds, minimizeargs, maxiter=20, tol=1e-5, disp=True, seed=8)


    # Calculate oof accuracy of optimized weights
    oof_accuracy = calculate_accuracy(np.take(y, test_idx, axis=0),
                                      list(map(lambda x: np.argmax(x), np.mean(
                                          np.take(yhats, test_idx, axis=0) * sol.x, axis=1))))

    print(f"{oof_accuracy}")

    accuracy.append((sol.x, oof_accuracy))
```

각 모델별 가중치

```
[(array([0.54944069, 0.31792854, 0.33211889, 0.73793193]), 0.9089719626168224),
 (array([0.29503175, 0.30682115, 0.27844213, 0.67464078]), 0.9113853056646102),
 (array([0.84300212, 0.39496332, 0.00977134, 0.72679045]), 0.9070854365301926),
 (array([0.80643955, 0.37531577, 0.07020489, 0.78758834]), 0.9113853056646102)]
```

2. Solution

(1) 2nd Solution Summary

https://tfhub.dev/google/cropnet/classifier/cassava_disease_V1/2

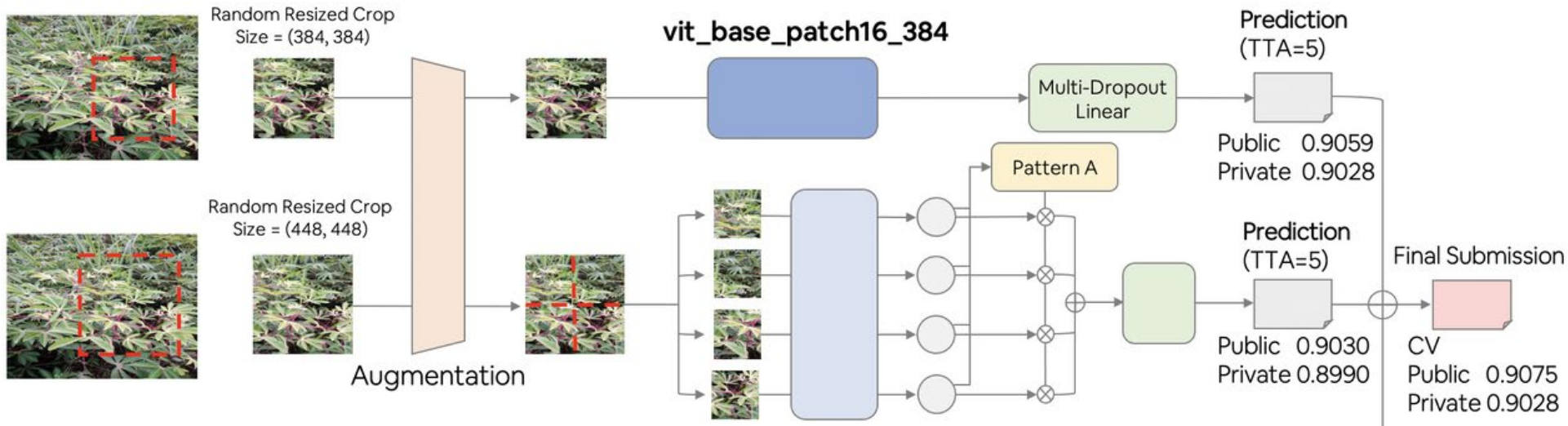
Private Score 0.9043 / Public Score 0.9025

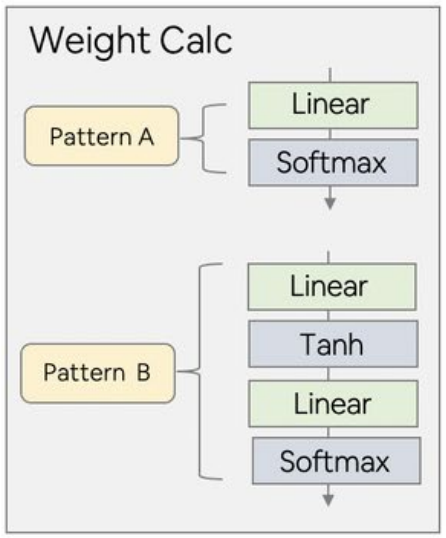
- Model : MobileNetV3 Pretrained Model (Label 6 : 기존 5개 + UnKnown) with Fine Tuning
- Image Size : 224, 224, 3
- Preprocessing
  - Augmentation : Filp, Brightness, Saturation
- Training : EarlyStopping, ReduceLROnPlateau
- Validation : 25% Dataset

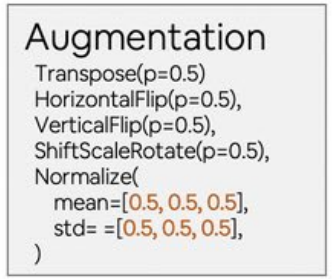
2. Solution

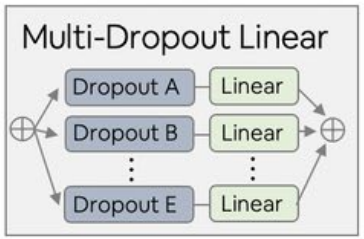
(1) 3rd Solution Summary

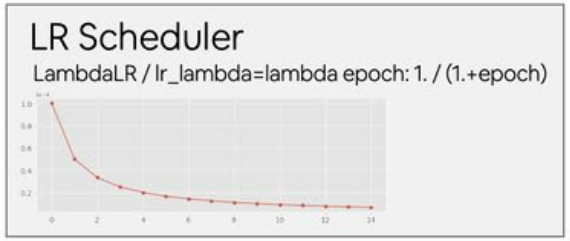
Private Score 0.9028 / Public Score 0.9075











kaggle

Makerere AI Lab

질문

1. Multi-Dropout Linear 어떻게 생각한건지???
생각의 스트림이 궁금!!!

2. Solution

(1) 3rd Solution Summary

Private Score 0.9028 / Public Score 0.9059

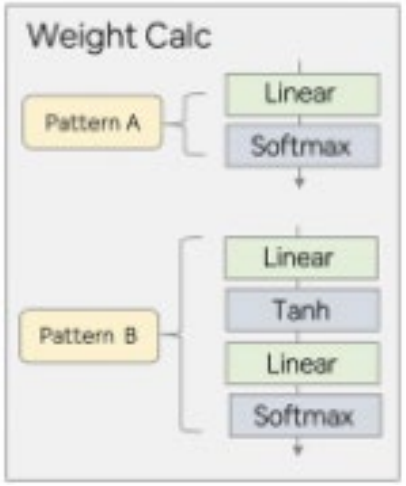
- vit_base_patch16_384
- 8xTTA(transpose, flip)
- ensemble(just average)vit_base_patch16_384
- img_size = 384 x 384
- 5x TTA

Private Score 0.8990 / Public Score 0.9030

- vit_base_patch16_224 - A
- img_size = 448 x 448
- 5x TTA
- weight calculation pattern A

Private Score 0.8952 / Public Score 0.9034

- vit_base_patch16_224 - B
- img_size = 448 x 448
- 5x TTA
- weight calculation pattern B
- label smoothing, alpha=0.01




```
# pattern A
self.att_layer = nn.Linear(n_features, 1)

# pattern B
self.att_layer = nn.Sequential(
    nn.Linear(n_features, 256),
    nn.Tanh(),
    nn.Linear(256, 1),
)
```

2. Solution

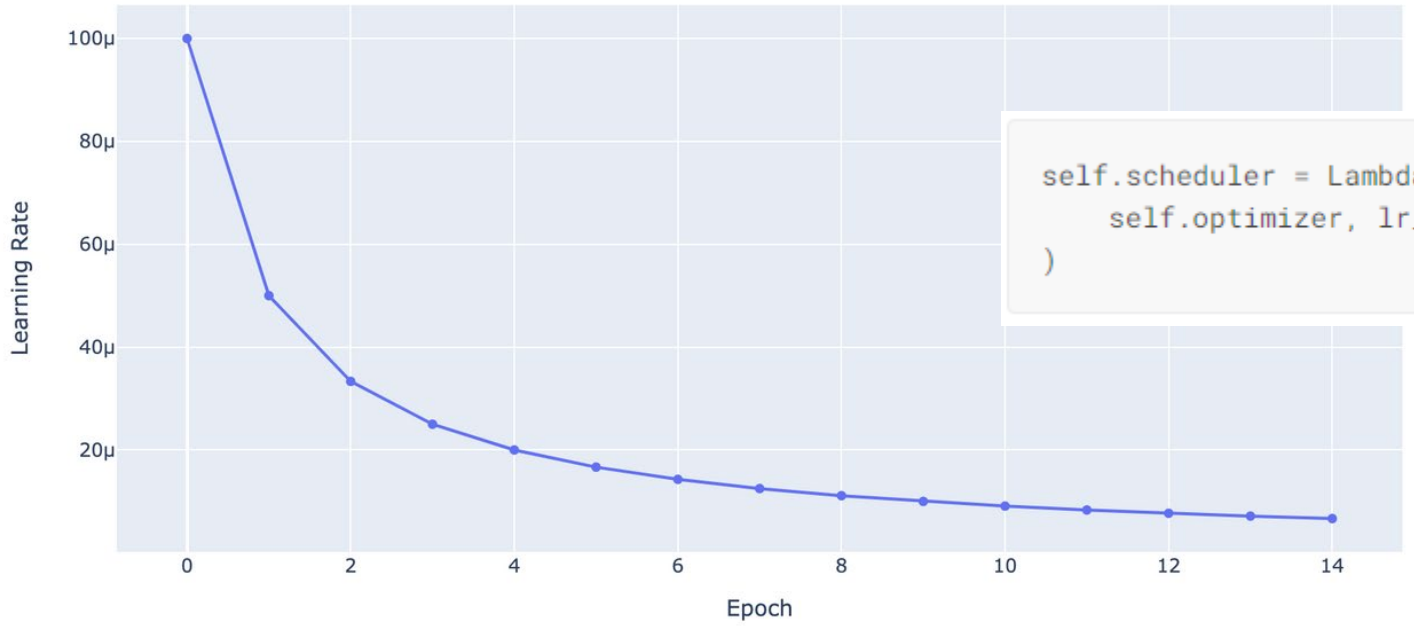
(1) 3rd Solution Summary

Private Score 0.9028 / Public Score 0.9059

- 5fold StratifiedKFold
- 2019 & 2020 Data
- Augmentation
- LambdaLR Scheduler

```
if aug_ver == "base":
    return Compose([
        RandomResizedCrop(img_size, img_size),
        Transpose(p=0.5),
        HorizontalFlip(p=0.5),
        VerticalFlip(p=0.5),
        ShiftScaleRotate(p=0.5),
        Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),
        ToTensorV2(),
])
```

LambdaLRScheduler, LearningRate and Epoch

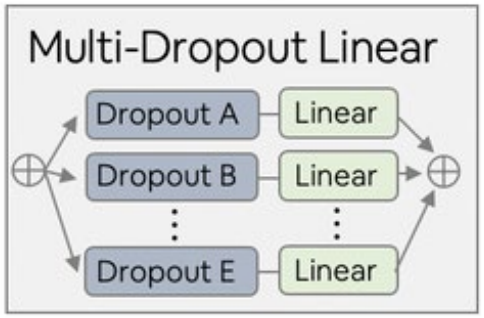



```
self.scheduler = LambdaLR(
    self.optimizer, lr_lambda=lambda epoch: 1.0 / (1.0 + epoch)
)
```

2. Solution

(1) 3rd Solution Summary

Multi DropOut




```
# when setting model
for i in range(5):
    self.head_drops.append(nn.Dropout(0.5))

# when training
for i, layer in enumerate(self.head_drops):
    if i == 0:
        output = self.head(layer(h))
    else:
        output += self.head(layer(h))
output /= len(self.head_drops)
```

2. Solution

• (1) 5th Solution Summary

Private Score 0.9030 / Public Score 0.9019

- Model : MobileNetV3 Pretrained Model (Label 6 : 기존 5개 + UnKnown) with Fine Tuning
- Image Size : 224, 224, 3
- Preprocessing
  - Augmentation : Filp, Brightness, Saturation
- Training : EarlyStopping, ReduceLROnPlateau
- Validation : 25% Dataset

2. Solution

(1) 7th Solution Summary / Public 71st

Private Score 0.901 / Public Score 0.906 / CV 0.90398

- Augmentation : Randomcrop, H/V flip, a large number of RGB transform, cutout, grid distortion

6modelx4TTA:

model	loss	size
efficientnet_b4_ns	bi-tempered	512
seresnext101	bi-tempered	512
seresnext101	focalcos	512
seresnext50	bi-tempered	512
VIT_base16	focalcos	384
VIT_base16	bi-tempered	384

3model and 5xrandomTTA:
CV --- | LB 0.902 | PB 0.902

model	loss	size
efficientnet_b4_ns	bi-tempered	512
seresnext101	bi-tempered	512
VIT_base16	bi-tempered	384

5model and 5xTTA:
CV --- | LB 0.901 | PB 0.902

model	loss	size
efficientnet_b4_ns	bi-tempered	512
efficientnet_b4_ns	focalcos	384
seresnext101	bi-tempered	512
VIT_base16	focalcos	384
regnety120	focalcos	512

2. Solution

(1) 7th Solution Summary / Public 71st

Private Score 0.902 / Public Score 0.900 / CV 0.9063

- Augmentation : Randomcrop, H/V flip, a large number of RGB transform, cutout, grid distortion

6 student model and 4xTTA
CV 0.9079 | LB 0.901 | PB 0.898

model	loss	size
efficientnet_b4_ns	CE	512
seresnext101	CE	512
VIT_base16	CE	384
resnext101	CE	512
resnest101	CE	512

- First Scheme를 기준으로 Knowledge distillatioN을 수행
- Student 0.4 + Teacher 0.6으로 Weighted Sum

2. Solution

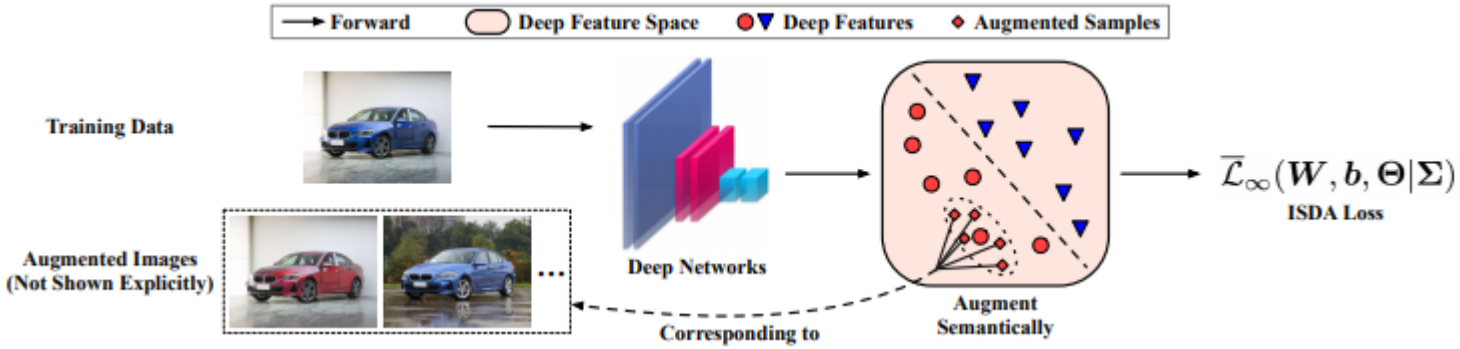
(1) 7th Solution Summary / Public 71st

Trial and Errors

- images with **prediction probability greater than 0.9** and incorrect predictions are **deleted**. PB 0.895
- the labels of images with prediction probability greater than 0.9 and incorrect predictions are **changed**. PB 0.896
- simply aug + **snapmix**, and gradually increase the probability of snapmix by epoch. PB 0.896
- 2019+2020 Dataset, since the 2019 data set contains multiple sizes, and any resize/crop form will affect the final CV, I finally did not use the 2019 data (The data for 2019 was deleted in the verification set) PB 0.895
- focalcos loss will increase CV, but some models will reduce the prediction probability of true label.
- in my submit, when using random TTA, the score of PB and LB are close.
- RegNet PB is higher than LB, but RegNet LB is too low, RegNet is not selected for submit.

2. Solution

(1) 8th Solution Summary



Private 0.9017

- Data : 2020 with Stratified 5fold
- Model : EfficientNetB6, ResNest50, VIT (patch16)
  - EfficientNet : PB 903 LB 894
- Augmentation
  - HorizontalFlip, Transpose, VerticalFlip, HueSaturationValue, CoarseDropout, RandomBrightnessContrast, ShiftScaleRotate, RGBShift, Cutmix, Fmix, Snapmix, **ISDA(implicit semantic data augmentation)**
  - cutmix, fmix가 점수향상에 큰 도움을 주었으며 snapmix의 경우 튜닝이 어려웠음
- Loss : LabelSmoothing, FocalLoss ,FocalCosineLoss, BiTemperedLoss, ClassBlancedLoss(only training)
- Optimizer: Adam, AdamW, AdamP, Ranger
- lr_scheduler: CosineAnnealingWarmRestarts
- 8xTTA(transpose, flip)
- ensemble(just average)

2. Solution

(1) 17th Solution Summary

Private 0.9013 / Public 9039

- Model(시도해본 모든 모델)
  - Ensemble
    - Efficientnet_b3
    - Efficientnet_b4
  - resnext50_32x4d weight사용
    - pretrained weight로 사용
  - Efficientnet_b2
  - resnext50_32x4d
- Dataset
  - 2020
- Augmentation
  - 어떤 증강기법을 사용했는지 명확히 나오지 않는데 위의 resnet 가중치를 활용했다는 것을 봤을 때 이분도 처음 참여하는 것을 미루어보아 아래의 Aug를 사용했을 확률이 가장 높음
  - Fmix는 사용하지 않음

```
RandomResizedCrop(CFG.size, CFG.size),
Transpose(p=0.5),
HorizontalFlip(p=0.5),
VerticalFlip(p=0.5),
ShiftScaleRotate(p=0.5),
Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225],
),
ToTensorV2(),
```

- Final Submission
  - 최종 모델: Effi_b3 + Effi_b4 ensemble(resnext50_32x4d weight사용)
  - Efficientnet_b2의 경우 가장 높은 CV를 보였으나 LB에서 점수가 너무 낮았음
  - resnext50_32x4d의 경우도 점수 향상에 기여를 하지 못함

2. Solution

(1) 18th Solution Summary

https://www.kaggle.com/projdev/base-notebook-cassava-leaf-disease-classification

Private 0.9013 / Public 9054

- Model(시도해본 모든 모델)
  - gluon_seresnext101_32x4d
    - CV: 0.9012; Public LB: 0.9011; Private LB: 0.8993
    - https://github.com/rwightman/pytorch-pretrained-gluonresnet
- tf_efficientnet_b4_ns
  - CV: 0.9013; Public LB: 0.9011; Private LB: 0.8970
- seresnext50_32x4d
  - CV: 0.9007; Public LB: 0.8978, Private LB: 0.8987

- Dataset
  - 2019, 2020

2. Solution

(1) 18th Solution Summary

https://www.kaggle.com/projdev/base-notebook-cassava-leaf-disease-classification

Training (3 Stage Model)

- 1st stage
  - image size: 320x320 or 384x384
  - 10 epochs
  - learning rate: 1e-4
  - train set: 2019 + 2020 data
  - validation set: 2020
  - scheduler: GradualWarmupScheduler
  - criterion: TaylorCrossEntropyLoss with label smoothing=0.2
  - optimizer: SAM + Adam
  - augmentation: RandomResizedCrop, Transpose, HorizontalFlip, VerticalFlip, ShiftScaleRotate, Cutout

2. Solution

(1) 18th Solution Summary

https://www.kaggle.com/projdev/base-notebook-cassava-leaf-disease-classification

Training (3 Stage Model)

- 2nd stage
  - image size: 512x512
  - 25-40 epochs
  - learning rate: 1e-4
  - train set: 2020 data
  - validation set: 2020 data
  - scheduler: GradualWarmupScheduler
  - criterion: Combo Loss (see notebook in detail)
  - optimizer: SAM + AdamP
  - augmentation: RandomResizedCrop, Transpose, HorizontalFlip, VerticalFlip, ShiftScaleRotate, Cutout + Cutmix

```
class ComboLoss(nn.Module):
    def __init__(self):
        super(ComboLoss, self).__init__()

        self.loss_1 = BiTemperedLogisticLoss(t1=0.32, t2=1.0, smoothing=0.0)
        self.loss_2 = TaylorCrossEntropyLoss(smoothing=0.06)
        self.loss_3 = NCEandRCE(alpha=0.17, beta=0.34, num_classes=5)
        #self.loss_4 = FocalCosineLoss(alpha=0.3868, gamma=1.934, xent=0.1)

    def forward(self, preds, labels):
        return self.loss_1(preds, labels) + self.loss_2(preds, labels) + self.loss_3(preds, lab
els) #+ self.loss_4(preds, labels)
```

```
# set optimizer
base_optimizer = AdamP
optimizer = SAM(model_parameters,
                base_optimizer,
                rho=Config['rho'],
                lr=Config['init_lr'],
                weight_decay=weight_decay,
                nesterov=True)
```

2. Solution

(1) 18th Solution Summary

https://www.kaggle.com/projdev/base-notebook-cassava-leaf-disease-classification

Training (3 Stage Model)

- 3rd stage
  - 5 epochs
  - learning rate: 1e-5
  - train set: 2020 data
  - validation set: 2020 data
  - scheduler: no scheduler
  - criterion: Combo Loss (see notebook in detail)
  - optimizer: SAM + AdamP
  - augmentation: hard albumentation-based augmentations

2. Solution

(1) 18th Solution Summary

https://www.kaggle.com/projdev/base-notebook-cassava-leaf-disease-classification

Training (3 Stage Model)

- hard Augmentation + base aug

```
        albumentations.RandomCrop(Config['image_size'], Config['image_size']),
        albumentations.Resize(Config['image_size'], Config['image_size']),
        albumentations.HorizontalFlip(p=0.5),
        albumentations.VerticalFlip(0.5),
        albumentations.Transpose(p=0.5),
        albumentations.Rotate(limit=(-90, 90), p=0.5),
        albumentations.OneOf([
            albumentations.ShiftScaleRotate(),
            albumentations.ElasticTransform(alpha=3)
        ], p=0.5),
        albumentations.OneOf([
            albumentations.OpticalDistortion(distort_limit=1.0),
            albumentations.GridDistortion(num_steps=5, distort_limit=1.0)
        ], p=0.5),
        albumentations.OneOf([
            albumentations.HueSaturationValue(hue_shift_limit=0.2, sat_shift_limit=0.2, val_shi
ft_limit=0.2),
            albumentations.RandomBrightnessContrast(brightness_limit=(-0.1, 0.1), contrast_limi
t=(-0.1, 0.1)),
            albumentations.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),
            albumentations.FancyPCA(),
            albumentations.CLAHE(clip_limit=4.0)
        ], p=0.5),
        albumentations.OneOf([
            albumentations.IAAAffine(),
            albumentations.IAAPerspective(),
            albumentations.IAAPiecewiseAffine(),
            albumentations.IAASuperpixels()
        ], p=0.5),
        albumentations.Cutout(max_h_size=int(Config['image_size'] * 0.375), max_w_size=int(Conf
ig['image_size'] * 0.375), num_holes=1, p=0.5),
        albumentations.Normalize(mean=[0.485,0.456,0.406], std=[0.229,0.224,0.225], max_pixel_v
alue=255, p=1.0),
        ToTensorV2(p=1.0)
    ])
```

- final submission
  - 최종 모델: 위의 세개 모델 simple ensemble(Public LB: 0.905; Private LB: 0.9013)
  - optimizer로 SAM + AdamP를 활용한 것이 robust 한 모델을 만드는데 큰 영향을 미침
    - shakeup에 견고
- 여담으로 GPU문제 때문에 SAM + SGDP and add FocalCosineLoss in Combo loss를 활용해보지 못함

2. Solution

(1) 25th Solution Summary

25	▲19	tom88jerry		0.9011

Private 0.9011 / Public ??

- Efficiencynet b4 + ViT의 stacking ensemble (첫 번째 제출)
- ViT 모델만 (두 번째 제출)
- Efficiennet은 2020 년 데이터만 사용
  - 둘 다 사용하려고했지만 점수가 좋지 않았음
  - 2019년도 이미지 사이즈 때문이라고 생각
- ViT, Deit, R50 + ViT는 2020 + 2019를 사용 (클래스 0,1,2,4에 대한 데이터만)
  - 클래스 3에는 이미 많은 이미지가 있음 (다른 클래스보다 6 배 높음)
- Architectures & Augmentation
  - cutmix, mixup, fmix snapmix를 시도했지만 성능향상 X

Architectures

Model	Public LB	Priave LB
Efficientnetb4	0.9023	0.8977
Deit_base_patch32_38	0.9018	0.8980
Vit_base_patch32_38	0.9061	0.8978
R50+ViT-B	0.9009	0.8975

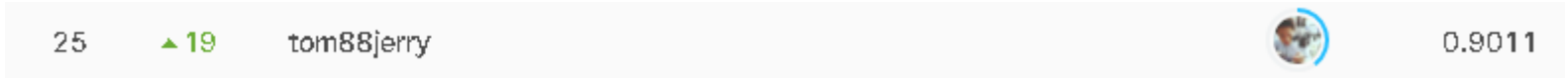
Architectures

- RandomResizedCrop
- Transpose
- HorizontalFlip
- VerticalFlip
- ColorJitter (brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2, always_apply=False, p=0.5),
- OneOf([MotionBlur(blur_limit=3), MedianBlur(blur_limit=3), GaussianBlur(blur_limit=3),], p=0.5,),
- Normalize
- CoarseDropout
- Cutout

Augmentation

2. Solution

• (1) 25th Solution Summary



Private 0.9011 / Public ??

- Training parameters
  - @piantic (<-Heroseo)공유 노트북에서 사용할 수 있는 모든 loss function들을 시도
    - CrossEntropyLoss, LabelSmoothing, FocalLoss, FocalCosineLoss, SymmetricCrossEntropyLoss, BiTemperedLoss, TaylorCrossEntropyLoss … 등

- 또한 손실 함수의 몇 가지 기본 혼합
- 가장 좋은 결과는 Labelsmoothing = 0.3
- 스케줄러 : CosineAnnealingWarmRestarts
- 옵티 마이저 : AdamP train for 10 epoch + 4 epochs of fine tunning

2. Solution

• (1) 25th Solution Summary

25	▲19	tom88jerry		0.9011

Private 0.9011 / Public ??

- Ensemble
  - 2-4 개의 모델에서 다양한 모델 조합을 시도
  - ViT와 EfficientNet 만이 좋은 시너지
  - 1. ViTbase16(10folds) of 2 lr 1e-2과 1e-6
    - public 0.906 private 0.8984.
    - ViT가 높은 public LB를 달성했지만 낮은 CV를 달성

  - 2. Efficientnet (0.4) & ViT (0.6)
    - Public LB = 0.9068, Private LB = 0.9011

  - 최고 private 점수는 0.9017
    - EfficientNet(0.4)& ViT (0.6)의 조합
    - 2020 년 데이터 세트로만 훈련 된 ViT 모델 사용

2. Solution

(1) 28th Solution Summary

Private 0.901 / Public 908

- Model

- EfficientNet B4 with Noisy Student
- SE-ResNeXt50 (32×4d)
- Vision Transformer (base patch16)

Train	Inference	Public LB	Private LB	CV
EfficientNet	EfficientNet-inf	**0.900**	0.891	0.89103
SE-ResNeXt50	SE-ResNeXt50-inf	0.899	0.894	**0.89532**
ViT	ViT-inf	0.899	0.890	0.89220

Inference	Validation	TTA	Public LB	Private LB	CV	TTA weight
inf-no-TTA	val-no-TTA	noTTA	0.905	0.896	0.9429	-
inf-TTA	-	noTTA + TTAx6	0.907	0.899	-	6:6
inf-TTA-weight	-	noTTA + TTAx6	**0.908**	0.900	-	4:6

I decided TTA weight by the public LB score, So I think this may overfit to public LB.
I choose second final submission is average of no TTA and TTA.

Inference	Validation	TTA	Public LB	Private LB	CV	TTA weight
inf-TTA-avg	-	noTTA + TTAx9	0.908	0.901	-	9:9

2. Solution

(1) 28th Solution Summary

Private 0.901 / Public 908

- Dataset
  - 2020, 2019

- Augmentation
  - Image size 512 ~ 384
  - Additional augumentations
  - No augmentation for first few epochs.
  - Reduce augmentation for last few epochs.
  - Remove augmentation for final epoch.
  - CutMix
  - MixUp

- parameter
  - loss: Bi-Tempered Logistic Loss with label smoothing
  - batch normalization layers frozen for EfficientNet, VIT

- inference
  - TTA
  - light augmentation
- final submission
  - random seed ensemble
  - avg esemble(notta + 6tta)
  - weight esemble(notta + 6tta)

2. Solution

(1) 30th Solution Summary

Private Score 0.9010 / Public Score 0.9018

- Model : EfficientNetB5 + ResNext50(32x4d)
- Dataset : 2020 (Not use 2019)
- Preprocessing

```
Image size 512
Augumentations
 - ResizeCrop
 - Horizontal,VerticalFlip
 - OneOf(RandomBrightness, RandomContrast)
 - OneOf (RandomBlur,MedianBlur,GaussianBlur)
 - ShiftScaleRotate
 - Normalize
```

- Training : Adam / CrossEntropyLoss with StratifiedKFold

EfficientNetB5
- CV average 0.8898
- Public 0.894
- Private 0.897

ResNext50(32x4d)
- CV average 0.8898
- Public 0.896
- Private 0.894

2. Solution

(1) 37th Solution Summary

- https://www.kaggle.com/ttahara/infer-cassava-ens09
- https://www.kaggle.com/ttahara/infer-cassava-ens10

Private Score 0.9010 / Public Score 0.908

- Model(시도해본 모든 모델)
- 1D-CNN
  - weight optimization
- 2D-CNN
  - weight optimization

- small model
  - ResNet50D
  - ResNeXt50_32x4d
  - ECA-ResNet50D
  - ResNeSt50_fast_1s4x24d
  - RegNetY032
- big model
  - ResNet100D
  - ResNeXt101_32x4d
  - ECA-ResNet101D
  - ResNeSt101
  - RegNetY080

- Dataset
  - training: 2020 + 2019
  - validation: 2020

- Augmentation
  - Transpose
  - HorizontalFlip
  - VerticalFlip
  - ShiftScaleRotate
  - RandomResizedCrop
  - HueSaturationValue
  - RandomBrightnessContrast
  - OneOf(RandomErase, CoarseDropout, Cutout)
  - Normalize
  - ToTensorV2

2. Solution

(1) 37th Solution Summary

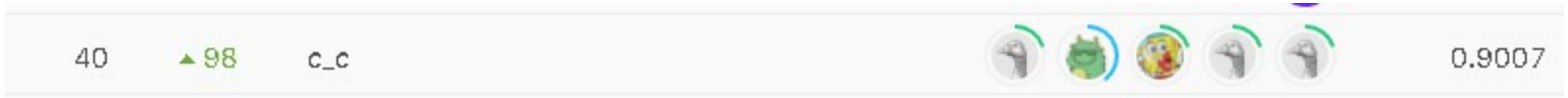
- https://www.kaggle.com/ttahara/infer-cassava-ens09
- https://www.kaggle.com/ttahara/infer-cassava-ens10

Private Score 0.9010 / Public Score 0.908

- parameter
  - loss: Cross Entropy Loss with label Smoothing (alpha=0.3)
  - kfold: 5
  - bs: 64(big model)/32(small model)
  - optimizer: adamw
  - lr: 5e-04(small models), 2.5e-04(big models)
  - scheduler: CosineAnnealingWarmRestarts
  - T_0: 10
  - T_mult: 1
- inference
  - TTA
  - light augmentation
- final submission
  - training 1D-CNN and 2D-CNN using image classification models' outputs as inputs
    - for more details, see my past topic in MoA competition:https://www.kaggle.com/c/lish-moa/discussion/204685
  - applying weight optimization to image classification models' outputs
  - finally averaging these three models' outputs
  - averaging(Public: 0.9018, Private 0.9014)
  - stacking(Public: 0.9014, Private: 0.9008)
  - 자기 현재 모델에선 stacking보다 simple average의 성능이 더 좋았음

2. Solution

(1) 40th Solution Summary



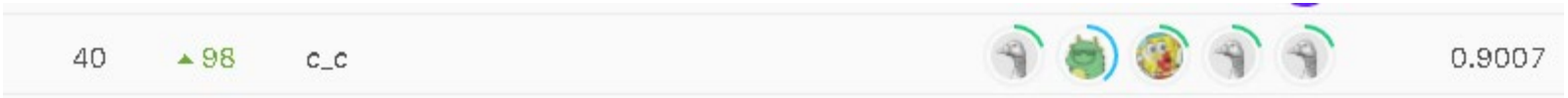
Private Score 0.9007 / Public Score 0.9050

Training Details

- Data: 2020 data with 5fold
- Image_size: 512x512
- Models: efficientnet b4,efficientnet b3,repvgg b2g4,se_resnext50
- Loss function: bi-tempered loss with label smooth
- Optimizer: AdamW
- Lr scheduler: Cosine Annealing with Warm Restarts
- Augmentations: HorizontalFlip, VerticalFlip,ShiftScaleRotate,mixup,snapmix(when we using HueSaturationValue and RandomBrightnessContrast,it improve our cv,but lower lb)
- TTA: ShiftScaleRotate,Flip

2. Solution

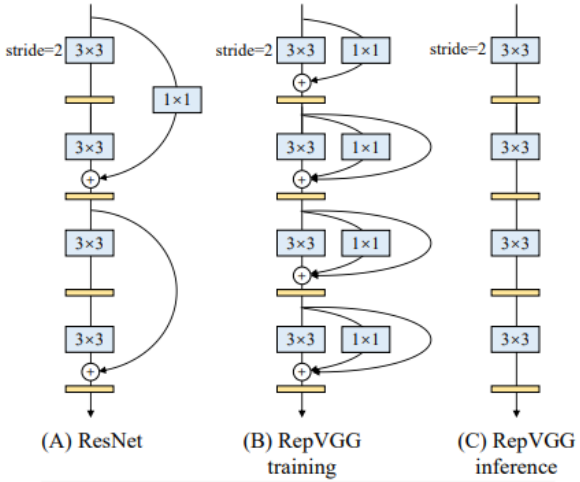
(1) 40th Solution Summary



Private Score 0.9007 / Public Score 0.9050

Ensemble

- 4 가지 모델을 사용 : efficientnet b4, efficientnet b3, repvgg b2g4, se_resnext50
- efficiencynet b3와 se_resnext50 : public lb 0.9000
- RepVGG B2g4 :public lb +0.003



methods that didn't work

- 노이즈 이미지를 제거하고 Phash를 사용하여 동일한 이미지를 찾고 유사한 이미지를 제거하면 cv와 lb가 모두 낮아짐
- Pseudo-Labelling
- concat avgpool feature & maxpool feature

2. Solution

• (1) 45th Solution Summary

45	▲82	Abhinand		0.9007

Private Score 0.9007 / Public Score 0.9050

Ensemble

- 5 fold OOF 예측을 사용하여 “정확도”를 기반으로 모델을 평가
- 공개 리더 보드 점수를 대안으로 처리 (포럼에서 제안한대로)
  - Treated the Public lb score as an alternative fold (as suggested in the forums)

Submission

- EfficientNet-B7 : best single model
- ViT alone was not enough on CV and LB
- 균등 : OOF (CV) 점수가 0.898-> 0.901로 향상, Public LB 0.902-> 0.905도 향상
- 가중 앙상블은 단일 모델보다 약간 더 나은 것처럼 보임
  - This gave me a feeling that an unweighted ensemble can cut it for me so I went with it.

Model	CV	Public LB	Private LB
EfficientNet-B7 NS	0.894	0.900	0.894
ViT Base16	0.888	-	-
EfficientNet-B7 NS 4xTTA	0.896	0.902	0.897
ViT Base16 4xTTA	0.891	0.899	0.896
(0.35 * ViT) + (0.65 * B7) weighted ensemble with 4xTTA	0.898	0.903	0.899
ViT + B7 unweighted ensemble with 4xTTA	0.901	0.905	0.900

2. Solution

(1) 77th Solution Summary

77	▼21	Dave E		0.9002

Private Score 0.9002 / Public Score ?

- 모델 예측을 기반으로 라벨이 잘못 지정되었거나 적어도 잠재적으로 오해의 소지가 있는 healthy 샘플 (약 150개) 중 일부를 삭제
  - 다른 샘플을 많이 없애면, 공개 LB가 떨어질 것 같았고 도움이 된다고 확신 할 수 없었음

- 2019 년 이미지에 대해 사전 학습 (클러스터를 기반으로 50 개의 라벨을 생성해서 pretrained)
  - 2019 년 데이터 사용해서 성능향상 못봄

- adversarial analysis of 2019 vs 2020에서 평균이미지가 상당한 차이를 보임
  - Test set이 train set과 비교하여 약간의 차이를 가질 수도 있겠다고 생각?
  - 이로 인해 CV와 LB를 모두 향상시켜야겠다고 생각함

- Individual classifiers used in blend had CVs in range of 0.905 (EnetB4, VIT) down to around 0.89. The CV numbers are including some TTA (different crops). Eng

- CV : 0.905 (EfficientNetB4, VIT) 범위에서 약 0.89까지

- CV에는 일부 TTA (different crops)가 포함

2. Solution

(1) 77th Solution Summary

77	▼21	Dave E		0.9002

Private Score 0.9002 / Public Score ?

- Trained with some variation in settings: Pretrain (2019) - 3 epochs with lowish / slightly increasing LR. More than this did not really produce any benefit I could see. Eng
- Pretrain (2019) :3에폭은 lr을 낮거나 약간 증가하도록 함
  - 이점을 못봄..
- Training Loss = bitempered('t1' : 0.6, 't2' : 1.4), TaylorCrossEntropyLoss
- Augmentations: CutMix p=0.5 (most of the models), CoarseDropout, Cutout (some models), (flip/rotate/hue/brightness)
- Learning rate: for main training (after pre training) started around 1e-04 and declined over around 8-15 epochs. Think I found that mostly some final epochs with a low LR could squeeze out a little more validation accuracy. Eng
- 학습률 : 메인 학습 (사전학습 후)의 경우 1e-04정도로 시작해서 약 8-15에폭에 걸쳐 감소
- LR이 낮은 일부 최종 에폭은 대부분 val. Acc.를 조금 더 끌어낼 수 있다는 것을 확인

2. Solution

(1) 77th Solution Summary

77	▼21	Dave E		0.9002

Private Score 0.9002 / Public Score ?

- Val. Acc. : 안정적인 검증 추세를 갖기 위해 rand crop->center crop
- train 정확도를 val 정확도보다 작게 유지하며 학습
- Public 0.901 LB (TTA를 사용한 5fold best EffiNet)에서 0.906으로 :
- 1. 블렌딩
  - Analysis of image crops : 무작위 TTA가 최선의 접근 방식이 아닐 수 있다는 것을 다양한 반복에서 깨달음
- 2. Classifier
  - 5 x tf_efficientnet_b4_ns, 512*512 image size
  - 1 x full image weight 0.082631
  - 1 x centre crop weight 0.102575
  - 4 x tta random resized crop weight 0.042924

4 x tf_efficientnet_b3_ns, 384*384 image size
- 1 x full image weight 0.028432

3 x vit_base_patch16_384, 384*384 image size
- 1 x full image weight 0.037909
- 1 x centre crop (512*512 resized) weight 0.126364
- 5 x tta random resized crop weight 0.063182

2 x resnext50_32x4d, 384*384 image size
- 1 x full image weight 0.1
- 1 x centre crop (512*512 resized) weight 0.1
- 5 x tta random resized crop weight 0.078689

2 x vit_small_patch16_224, 224*224image size
- 1 x full image weight 0.075

2. Solution

(1) 77th Solution Summary

77	▼21	Dave E		0.9002

Private Score 0.9002 / Public Score ?

- 모든 TTA는 일부 제한된 CV 실험을 기반으로 약 0.37-0.85 (RandomResizedCrop) 범위로 제한
- Optuna가 계산한 값에 따라 대략적으로 증가
- 최종 순위 변경은 일반적인 셰이크 업에 비해 상대적으로 적었음
- Just the vit_base_patch16_384 / EnetB4 / Resnext blend (no weighting, just unweighted full image+center crop+tta) scored:
  - Priv/Public 0.898 0.905
- Adding crop/tta weightings to the above 3:
  - Priv/Public 0.899 0.903
- Adding in the EnetB3 and vit_small_patch16_224
  - Priv/Public 0.900 0.906 (CV는 떨어짐)

2. Solution

(1) 78th Solution Summary

Private Score 0.9002 / Public Score ?

- 4개의 로컬 앙상블 (2개 자체, 2개 공개)을 포함하여 서로 다른 방법으로 훈련 된 29개의 서로 다른 모델로 구성된 글로벌 앙상블
- 모두 512로 훈련되었지만 다른 기술과 매개 변수를 사용

- 1 - FastAIv2 - 6 models
- 2 - Pytorch - 13 models

1 - FastAIv2 - 6 models
- Used 1-fold of each model from random 10-fold training.
- Ranger or AdamW optimizer
- LabelSmoothingCrossEntropy loss
- 1CycleScheduler
- 시작 3 에폭은 마지막 레이어만 학습하고 모든 레이어를 20 epochs 학습
- 자신의 custom LR finder 사용

1 - FastAIv2 - 6 models
- 2 tf_efficientnet_b5_ns – mixup training
- 2 seresnext50_32x4d – cutmix training
- 1 xception – cutmix training – here I also used upsample with more upsample to class 4 which I visually noticed had more false classifications.
- 1 densenet121 – cutmix training – 여기에서 시각적으로 더 많은 잘못된 분류가 있음을 눈으로 확인한 클래스 4로 더 많은 업샘플을 사용했음
- Aug를 변경하면서 각각 5 개의 TTA로 모델을 동일하게 앙상블
- 크롭하지않고 이미지 크기를 조정하기 위해 다른 3 개의 앙상블과 달리 다른 이미지보기를 가지고 다른 모든 aug를 건너 뛰었습니다. 임의의 밝기, 대비 등만 수행

2. Solution

• (1) 78th Solution Summary

Private Score 0.9002 / Public Score ?

- 4개의 로컬 앙상블 (2개 자체, 2개 공개)을 포함하여 서로 다른 방법으로 훈련 된 29개의 서로 다른 모델로 구성된 글로벌 앙상블
- 모두 512로 훈련되었지만 다른 기술과 매개 변수를 사용

- 1 - FastAIv2 - 6 models
- 2 - Pytorch - 13 models

2 - Pytorch - 13 models
- 8 tf_efficientnet_b1_ns, 5 resnext50-32x4d
- Extra training data
- BiTemperedLoss with Random Noise with 0.5/1.5.
- OneCycleLRWithWarmup from Catalyst which also takes in count momentum and weight_decay in the cycles.
- Used a random mix with almost equal split of all different aug. techniques SnapMix, Fmix, CutMix,CutOut, Mixup and MWH(Mixup-Without-Hesitation).
- AdamP optimizer

2 - Pytorch - 13 models
- 여기에서는 앙상블 대신 5 TTA로 스태킹을 사용하여 글로벌 앙상블에 대한 다른 접근 방식을 만듦
- 마지막 2 개의 로컬 앙상블에 대해 2 개의 공개 학습 모델 5 B4ns 및 5 resnext50-32x4d를 각 모델에 대해 4 개의 TTA와 함께 사용하여 더 큰 앙상블과 자신의 전략에 얽매이지 않도록 훈련
- 모든 4 개의 로컬 앙상블에서 동일한 가중치로 앙상블

2. Solution

• (1) 606th Solution Summary

Private LB: 606th, 0.898 Public LB: 34th, 0.907

- 6 model에 대해 실험
  - ResNeSt50, ResNeSt50 4s2x40d, EffNet-B3, 4, ViT-L/16, DeiT-B/16
  - ResNeSt50 4s2x40d, EffNet-B4 사용 based on LB & CV scores.

arch	CV	Public LB	Private LB
ResNeSt50-fast-4s2×40d (tta4)	0.891	0.898	0.895
EfficientNet-B3 NS (tta4)	0.895	0.896	0.895
EfficientNet-B4 NS (tta4)	0.891	0.900	0.893

- Robutness를 높이기 위해 동일한 아키텍처를 다양한 학습 방법으로 학습
- focal cosine loss & cross-entropy w/ label smoothing 0.2

arch	loss	CV	Public LB	Private LB
EfficientNet-B4 NS (tta3)	focal cosine	0.890	0.900	0.895
EfficientNet-B4 NS (tta4)	cross entropy w/ label smooth	0.891	0.900	0.893

2. Solution

(1) 606th Solution Summary

Private LB: 606th, 0.898 Public LB: 34th, 0.907

- Ensemble
  - CV에서 Optuna & Simple을 사용하여 앙상블 가중치를 조정
  - Simple : Simple is very fast optimizer which likely to converge to global optimum.
  - https://www.kaggle.com/daisukelab/optimizing-ensemble-weights-using-simple

- Combinations
  - best LB : 2 x ResNeSt50-fast-4s2x40d + 2 x EffNet-B4
  - best CV : 3 x ResNeSt50-fast-4s2x40d + 3 x EffNet-B4 + 1 x EffNet-B3

baseline	CV	Public LB	Private LB
best LB	0.9003	0.907	0.897
best CV	0.9051	0.905	0.896

2. Solution

(1) 606th Solution Summary

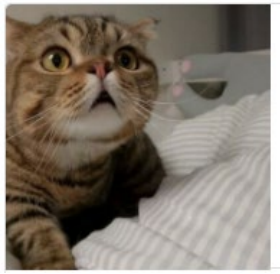
Private LB: 606th, 0.898 Public LB: 34th, 0.907

- TTA
  - Simply using CenterCrop + horizontal & vertical flop achieves better CV & LB scores.
- Training Recipe
  - Data : 2019 + 2020 datasets, only 2020, w/ pseudo labels
  - Augmentations : FMix, CutMix, SnapMix, MixUp, Cutout, etc…
  - Optimizer : AdamW, AdamP, RAdam
  - Loss : focal cosine, cross-entropy w/ label smoothing, bi-tempered
  - Scheduler : Cosine Anealling w/o warm-up (1e-4 to 1e-6)
  - Epochs : 10 ~ 20, Resolution : 512x512
  - Auxiliary : freeze BN layer, TTA : n_iters 4 is best in my case.

- Work
  - Label Smoothing (alpha = 0.2)
  - TTA (n_iters 4 is best)
  - Adam, AdamW to RAdam (slight improvement)
  - external dataset (2019 dataset)
- Not Works
  - noise-tolerant(?) loss like bi-tempered loss

3. 대회를 통해서 배운점

(1) 협업시에 코드를 공유하는 방법



Hyun woo kim

Student at Kaist

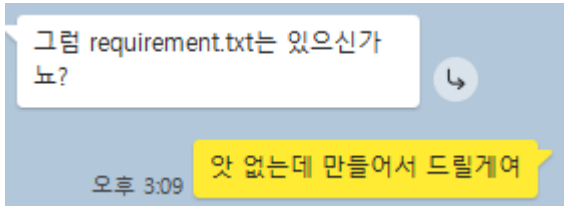
Bucheon-si, Gyeonggi-do, South Korea

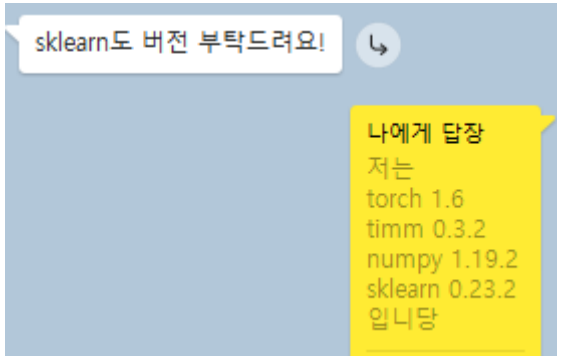
Joined 3 years ago · last seen in the past day

https://eda-ai-lab.tistory.com/

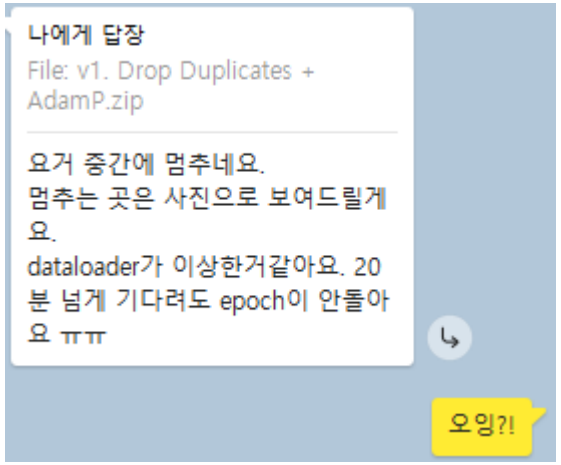
TEAM UP 혹은 코드 공유시에 부족했던 부분

- 처음 팀업시에 requirements.txt가 없고 이제까지 해온 분석에 대한 히스토리가 부족해서 나중에 팀업하신분들이 따라오기 힘들었던 점

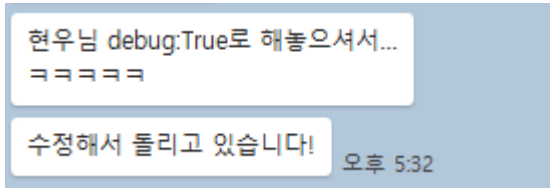


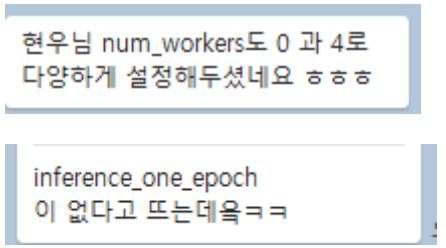


- 코드 공유시에 디버깅코드로 한번 돌아가는지 확인하고 보내줘야한다는 점 (에러 발생 ㅠㅠ)



- + 윈도우랑 리눅스 환경이 달라서 NUMWORKS 부분을 수정하고 보내줘야 했음



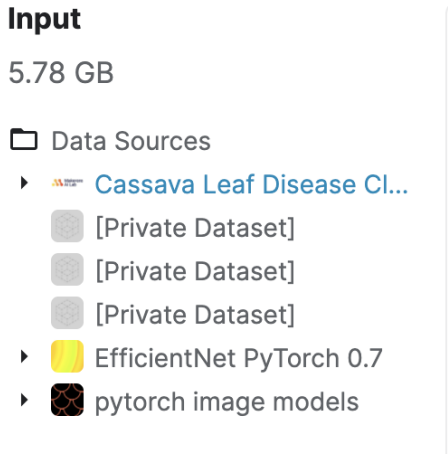


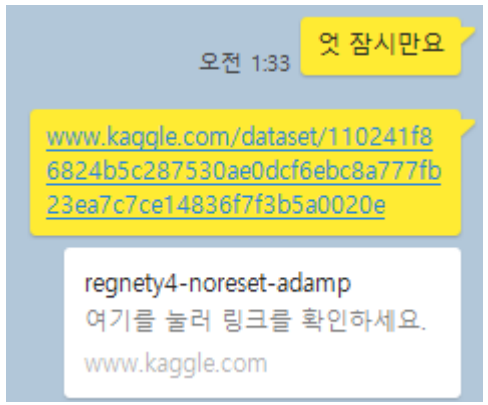
3. 대회를 통해서 배운점

(1) 협업시에 코드를 공유하는 방법

TEAM UP 혹은 코드 공유시에 부족했던 부분

- 웨이트 데이터 셋의 공유를 할때 팀원 모두에게 공유를 해야 바로바로 팀원들이 돌려볼 수 있음





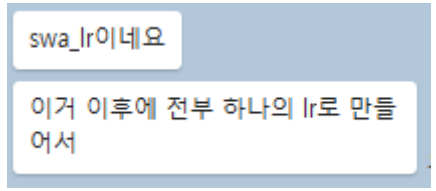
- CV 같은 출력부분을 팀원들이 바로 공유해줄 수 있도록 Log가 떨어지도록 코드를 만들어줘야 함

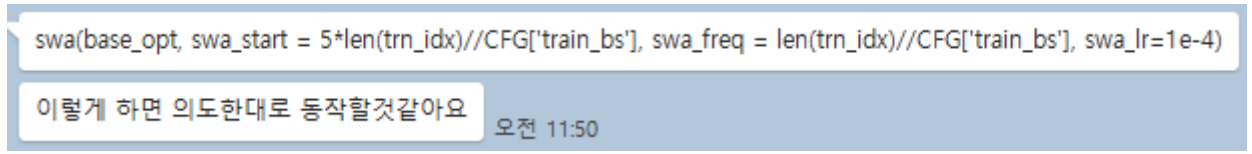
3. 대회를 통해서 배운점

(2) SWA 업데이트 코드 오류

SWA 업데이트 코드에서의 오류

- 현재 코드가 매 Epoch가 아니라 매 Step마다 게산이 되고 있었던 점
- SWA_LR에 의해서 특정 스탭 이후에 LR이 고정된 점





3. 대회를 통해서 배운점

(3) Accumulation과 Batch Size간의 관계

Accumulation과 Batch Size간의 관계

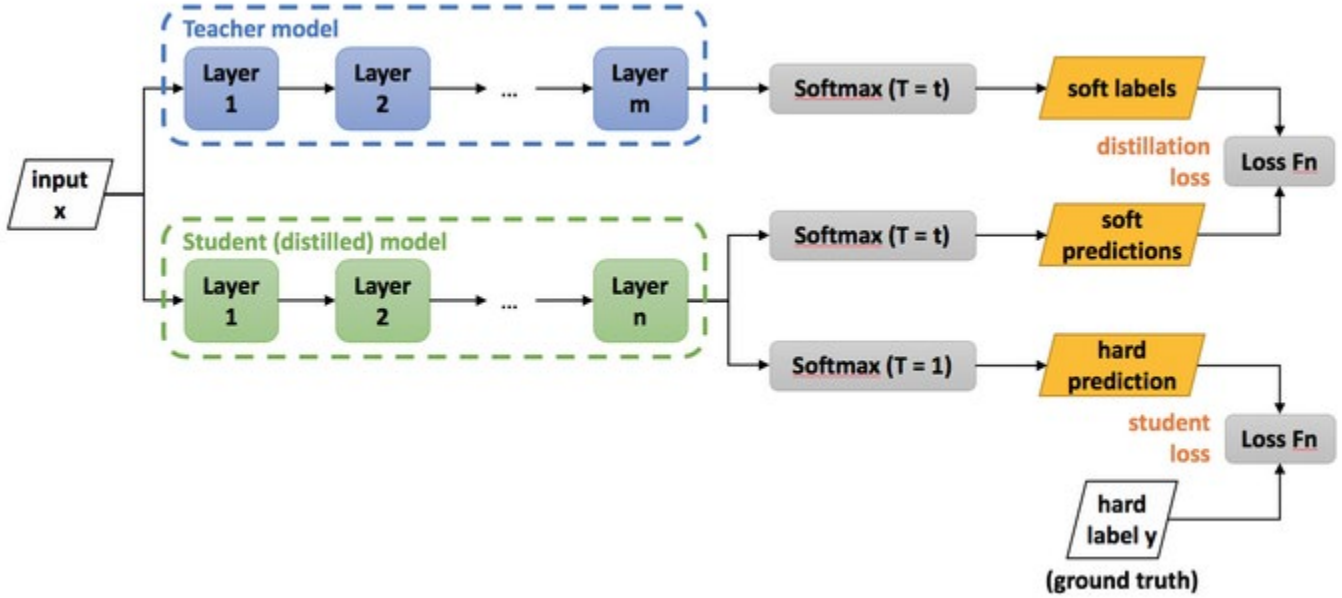
- Accumulation을 사용할 경우 큰 Batch Size와는 같이 사용하면 안좋음
- 배치 Normalization 2번 수행된게 backpropagation 되는거라 batch normalization한 mean, std와 backpropagation 통해서 생기는 역전파값이 조금 관계가 떨어질 수 잇는 문제가 있음

참고자료 : https://ai.stackexchange.com/questions/21972/what-is-the-relationship-between-gradient-accumulation-and-batch-size

3. 대회를 통해서 배운점

(4) Knowledge Distillation

Knowledge Distillation



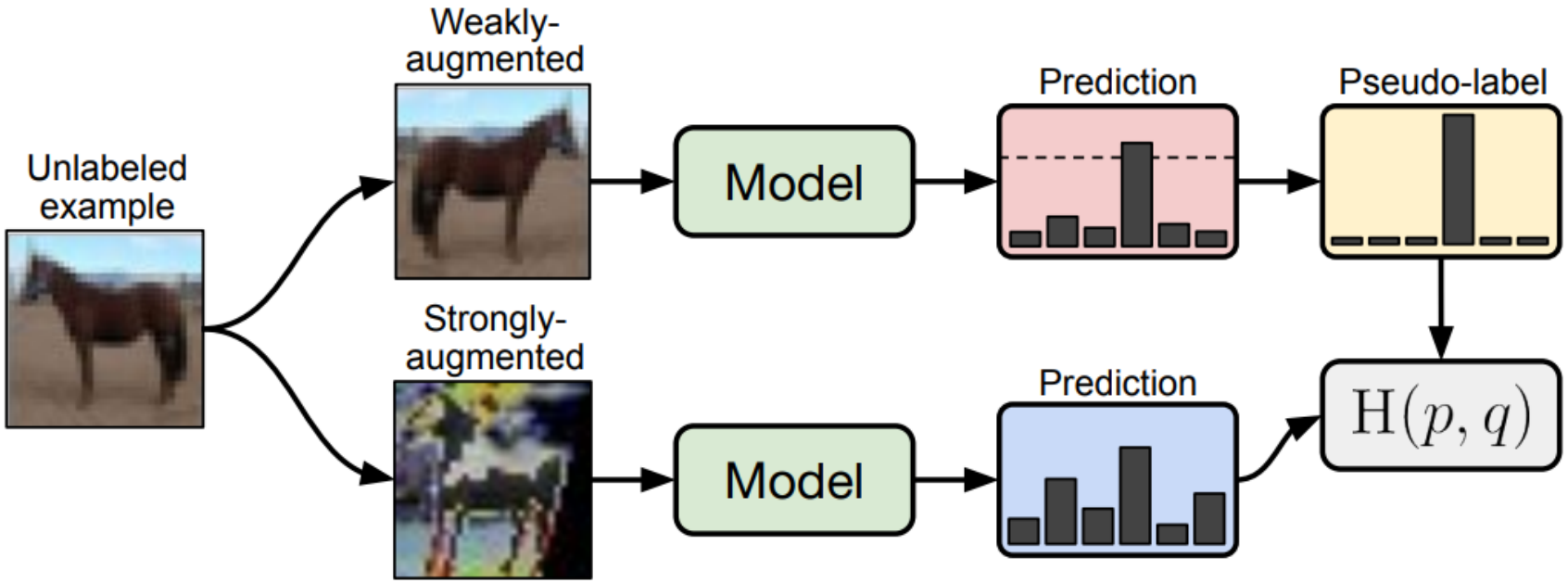
- 2 스테이지의 학습방법으로 먼저 베이스라인 Teacher Model을 생성
- Teacher Model을 기반으로 Student Model을 학습
- 최종적으로 Teacher Model + Student Model을 통해서 2가지의 Loss를 계산
- CV로 확인한 결과 시간은 오래걸리지만 학습이 되게 안정적으로 되는 경향이 있었음

참고자료 : https://light-tree.tistory.com/196

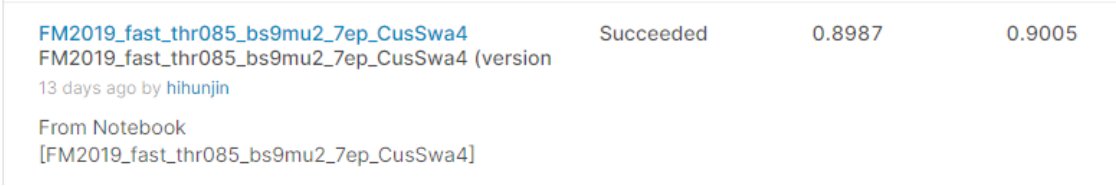
3. 대회를 통해서 배운점

• (5) FixMatch (Semi Supervised Learning)

FixMatch



- Label이 존재하지 않는 이미지들도 학습에 참여시켜서 Pesudo Label을 통해서 같이 학습하는 방법
- 2019년도는 라벨이 존재하지만 신뢰성이 없다고 판단하고 해당 영역을 Unlabel으로 주고 학습시킴
- 단일 모델로는 Private Score가 가장 높은 모델이었음



참고자료 : https://www.youtube.com/watch?v=nSJP7bn2D1U
참고자료 : https://www.youtube.com/watch?v=mXiPbkyGJ9g&feature=youtu.be

3. 대회를 통해서 배운점

- (6) Augmentation 기법들

Mixup Without Hesitation

- First stage: from 1 to pm mini-batches, we train with mixup. Note that here we assume pm is an integer.
- Second stage: from $pm + 1$ to qm mini-batches, we alternate between mixup and basic data augmentation. If the last mini-batch does not apply mixup, the next one will, and vice versa.
- Third stage: from $qm + 1$ mini-batches to the end, we run mixup with probability ϵ, where ϵ decreases *linearly* from 1 to 0.

Table 1: Accuracy (%) on CIFAR-100. Baseline means we train the model with basic data augmentation. Mixup means we apply mixup throughout the training process. First Half Mixup means the first half of epochs apply mixup but the last do not, and similarly, Second Half Mixup means we only apply mixup in the second half of epochs.

Methods	Top1	Top5
Baseline	74.20	92.53
Mixup	75.25	92.40
First Half Mixup	**75.87**	**93.10**
Second Half Mixup	72.50	91.04

```
for batch_idx, (inputs, targets) in enumerate(trainloader):
    if use_cuda:
        inputs, targets = inputs.cuda(), targets.cuda()

    mask = random.random()

    if epoch >= 90:
        # threshold = math.cos( math.pi * (epoch - 150) / ((200 - 150) * 2))
        threshold = (100 - epoch) / (100 - 90)
        # threshold = 1.0 - math.cos( math.pi * (200 - epoch) / ((200 - 150) * 2))
        if mask < threshold:
            inputs, targets_a, targets_b, lam = mixup_data(inputs, targets, args.alpha, use_cuda)
        else:
            targets_a, targets_b = targets, targets
            lam = 1.0
    elif epoch >= 60:
        if epoch % 2 == 0:
            inputs, targets_a, targets_b, lam = mixup_data(inputs, targets, args.alpha, use_cuda)
        else:
            targets_a, targets_b = targets, targets
            lam = 1.0
    else:
        inputs, targets_a, targets_b, lam = mixup_data(inputs, targets, args.alpha, use_cuda)
```

- Mixup의 경우 2가지 단점이 존재
  - 학습이 잘 되려면 충분한 Epoch가 필요함 (기존대비 많은 Epoch가 필요함)
  - 하이퍼파라미터 튜닝이 필요함 (쉽게 찾기 힘듬)
- 해결 : 초기 MixUp으로 Representation을 넓히고 Second에서 mixup의 실행을 하다, 말았다가 함으로써 Local Optimal에 빠지는 것을 방지. 이후, ϵ-greedy알고리즘을 통해서 mixup 확률 계산

참고자료 : https://github.com/yuhao318/mwh
참고자료 : https://arxiv.org/pdf/2101.04342.pdf

3. 대회를 통해서 배운점

(6) Augmentation 기법들

Data Augmentation with CycleGAN

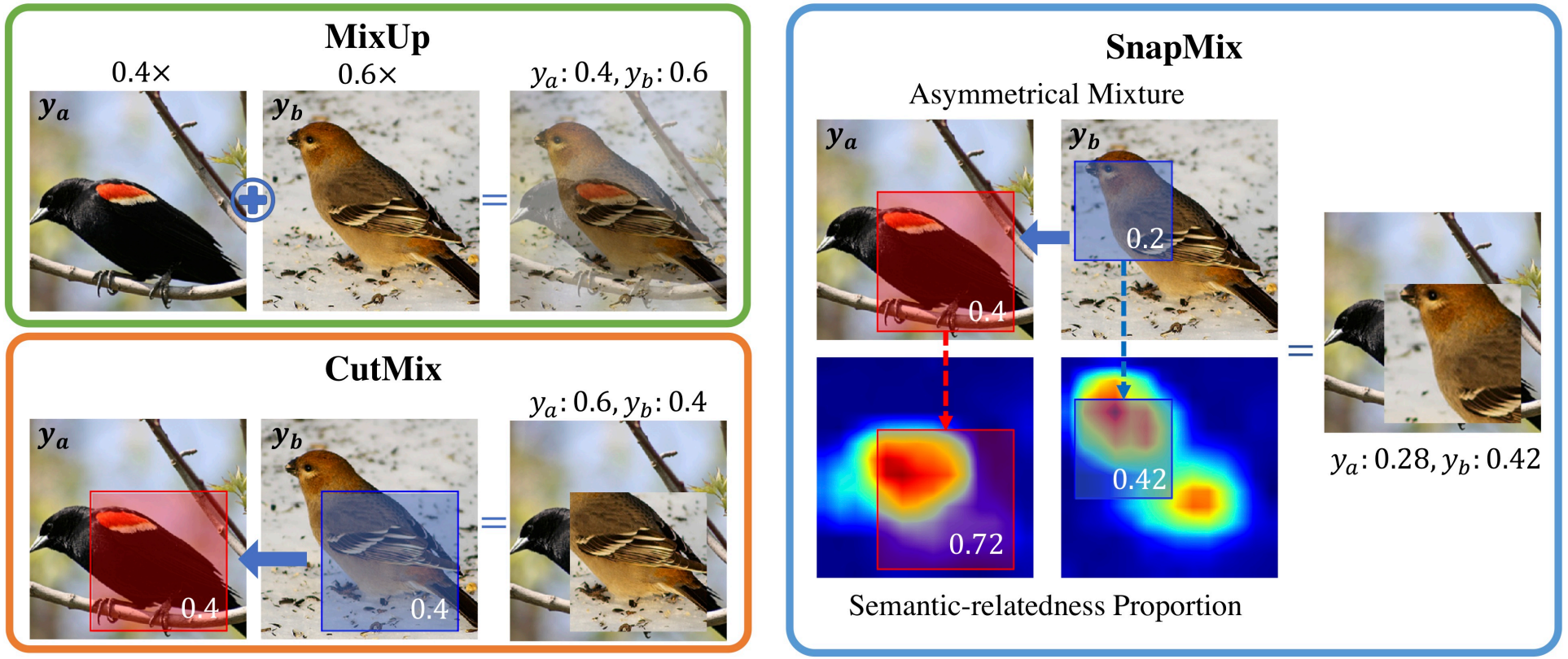
- CycleGAN을 이용해서 부족한 이미지 데이터를 늘리는 방법

참고자료 : https://www.kaggle.com/dimitreoliveira/cassava-leaf-disease-cyclegan-data-augmentation/notebook#Evaluating-generator-models

3. 대회를 통해서 배운점

• (6) Augmentation 기법들

SnapMix



- CAM을 통해서 중요한 부분을 확인하고 해당 부분을 Mix하는 방법 (기존 기법들보다 안정적임)

참고자료 : https://github.com/Shaoli-Huang/SnapMix

3. 대회를 통해서 배운점

• (7) 다양한 Loss Function과 Label Smoothing 기법들

Focal Loss

$$CE(p, y) = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1-p), & \text{otherwise.} \end{cases}$$

$$FL(p_t) = -(1-p_t)^{\gamma} \log(p_t).$$

- p가 1으로 잘 분류된 항목에 대해서는 가중치를 적게 받아서 로스가 작아지는 경향이 있음
- r이 0일때는 같으며 r이 증가할수록 잘분류되는 애들에 대해 패널티가 커짐 (Modulating factor)

- CE의 경우 잘 분류되는 항목들도 로스를 크게 잡는 문제가 있어서 이를 해결한 방법이 Focal Loss

3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

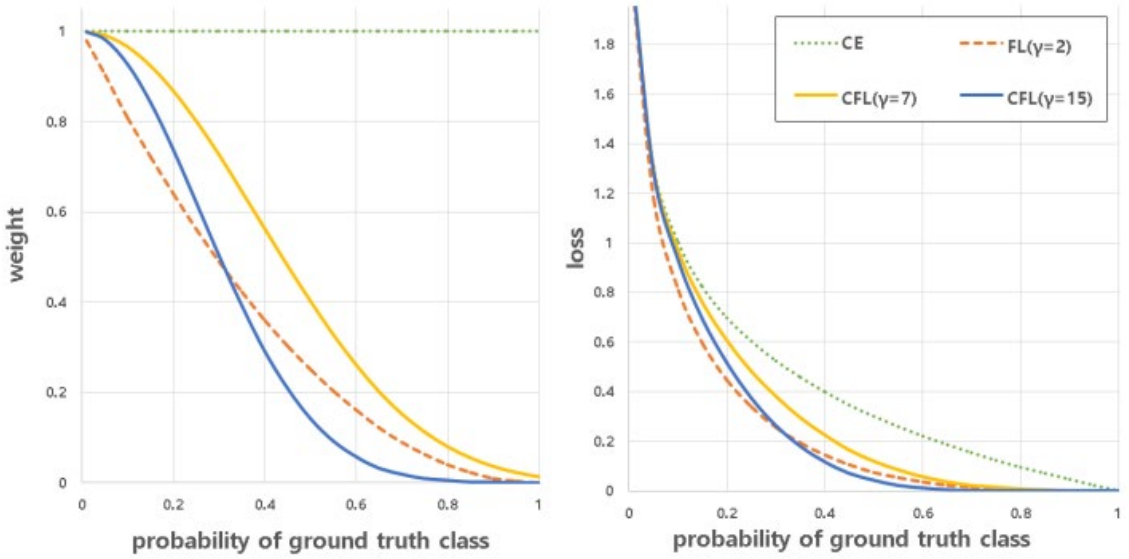
Focal Cosine Loss

$$p_c = \begin{cases} p, & \text{if } y \text{ is foreground} \\ 1-p, & \text{otherwise,} \end{cases}$$

$$CrossEntropy(p_c) = -\log(p_c)$$

$$FocalLoss(p_c) = -\alpha_c(1-p_c)^{\gamma}\log(p_c)$$

$$CosineFocalLoss(p_c) = -\cos^{\gamma} p_c \log(p_c)$$



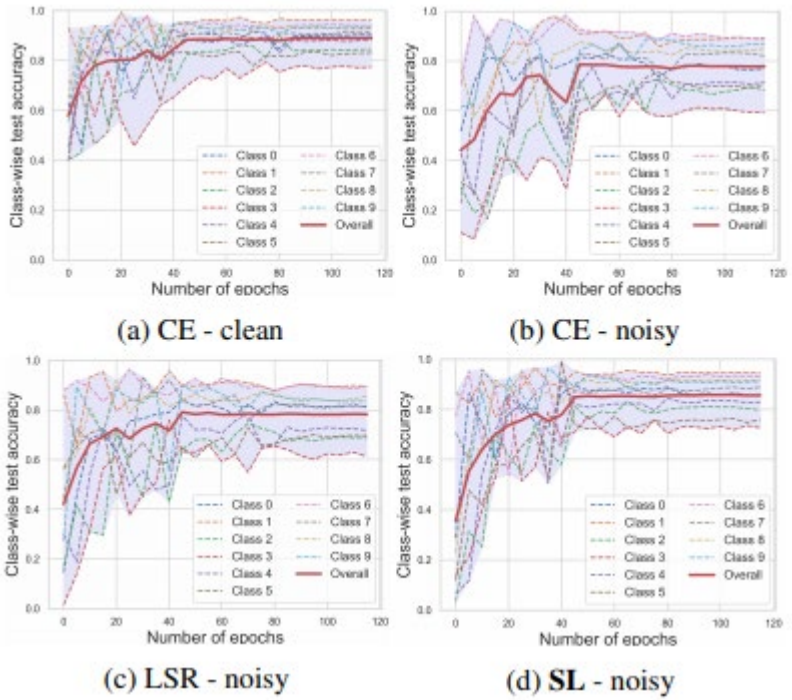
- 기존 Focal Loss는 잘 분류된 데이터만 초점을 맞추는 반면, Cosine Focal Loss는 희귀하게 어려운 데이터들에 대해서도 그래디언트가 잘 보존되도록 설계되었음
- 즉, 초점을 맞춰야 하는 몇가지 중요한 샘플을 잘 학습함
- gamma가 클수록 잘분류된 표본에 대해 가중치가 작음 (중요하게 여기지 않음)
- 왼쪽 그래프를 보면 CFL이 큰 확률로 잘못 분류한 샘플에 대해 가중치가 더 크고, 로스가 큰 것을 볼 수 있음 (잘 분류한 녀석들에 대해서는 반대로 로스와 가중치가 더 작음)

3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

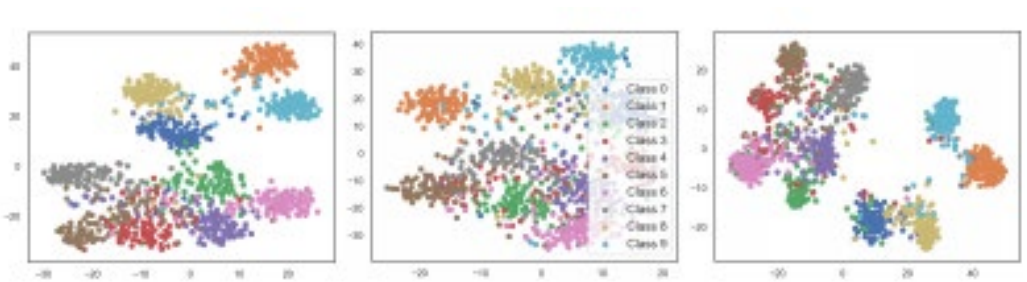
Noisy Label에 대해서 잘 학습하는 로스!!

Symmetric Cross Entropy Loss



(a) CE - clean (b) CE - noisy

(c) LSR - noisy (d) **SL** - noisy



(a) CE - clean (b) CE - noisy (c) SL - noisy

수식이 많아서 아직 이게 왜 잘 작동하는지는 파악하지 못했습니다 ㅠㅠ

출처 : https://arxiv.org/pdf/1908.06112.pdf

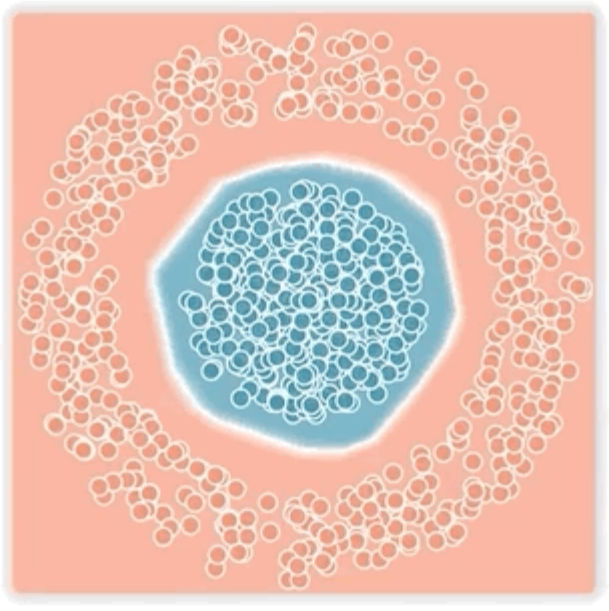
3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

Noisy Label에 대해서 잘 학습하는 로스!!

기존의 Loss Function이 가지고 있는 문제점들

- 이상치에 민감해서 하나의 이상치가 로스의 대부분을 차지할 수 있음. 이상치 하나에 의해서 결정 경계가 많이 나빠지는 문제가 발생함 (외각의 이상치가 추가될수록 결정경계가 많이 넓어지는 것을 볼 수 있음)



```
[63]: input = torch.tensor([[0.94, 0.01, 0.01, 0.02, 0.01]], requires_grad=True)
      target = torch.tensor([2])
      target

Out[63] tensor([2])

      + Code   + Markdown

[64]: loss = nn.CrossEntropyLoss()
      output = loss(input, target)
      output

Out[64] tensor(1.8786, grad_fn=<NllLossBackward>)
```

출처 : https://ai.googleblog.com/2019/08/bi-tempered-logistic-loss-for-training.html

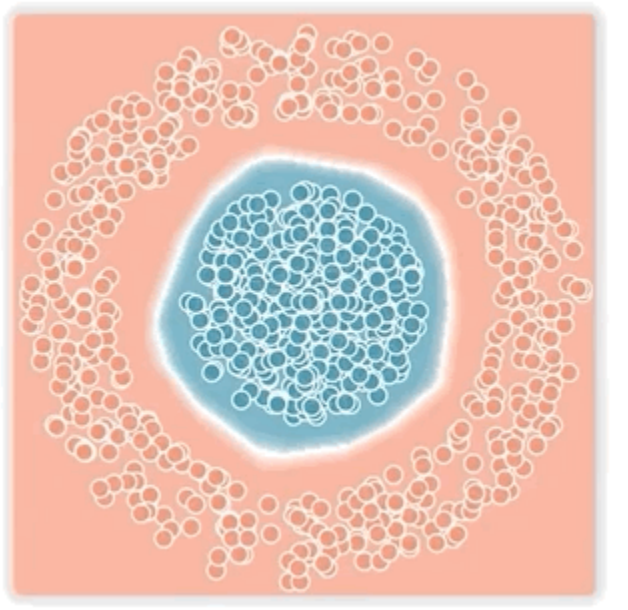
3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

기존의 Loss Function이 가지고 있는 문제점들

- 결정경계의 근처에 잘못 분류된 라벨들에 의해서 결정경계가 확장되는 문제가 발생 (낮은 수준의 노이즈에 의해서 네트워크의 일반화 성능이 낮아짐), 잘못된 라벨도 잘 맞추려고 하려고 경계를 확장하는 문제가 발생

- 이유 : 로지스틱 손실에 대한 이 전송 함수의 꼬리가 **기하급수적으로 빠르게 감소하기 때문에**, 훈련 프로세스는 작은 오류를 보상하기 위해 **경계를 잘못 레이블링된 예제에 더 가깝게 확장**하는 경향이 있음



```
[61]:
input = torch.tensor([[0.94, 0.01, 0.6, 0.02, 0.01]], requires_grad=True)
target = torch.tensor([2])
target

Out[61] tensor([2])
```

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

값이 극단적으로 되어서 오차가 크게 발생

$$Loss = (Y)(-log(Y_{pred})) + (1-Y)(-log(1-Y_{pred}))$$

Remains when Y = 1 / Removed when Y = 0

Remains when Y = 0 / Removed when Y = 1

```
[62]:
loss = nn.CrossEntropyLoss()
output = loss(input, target)
output

Out[62] tensor(1.4045, grad_fn=<NllLossBackward>)
```

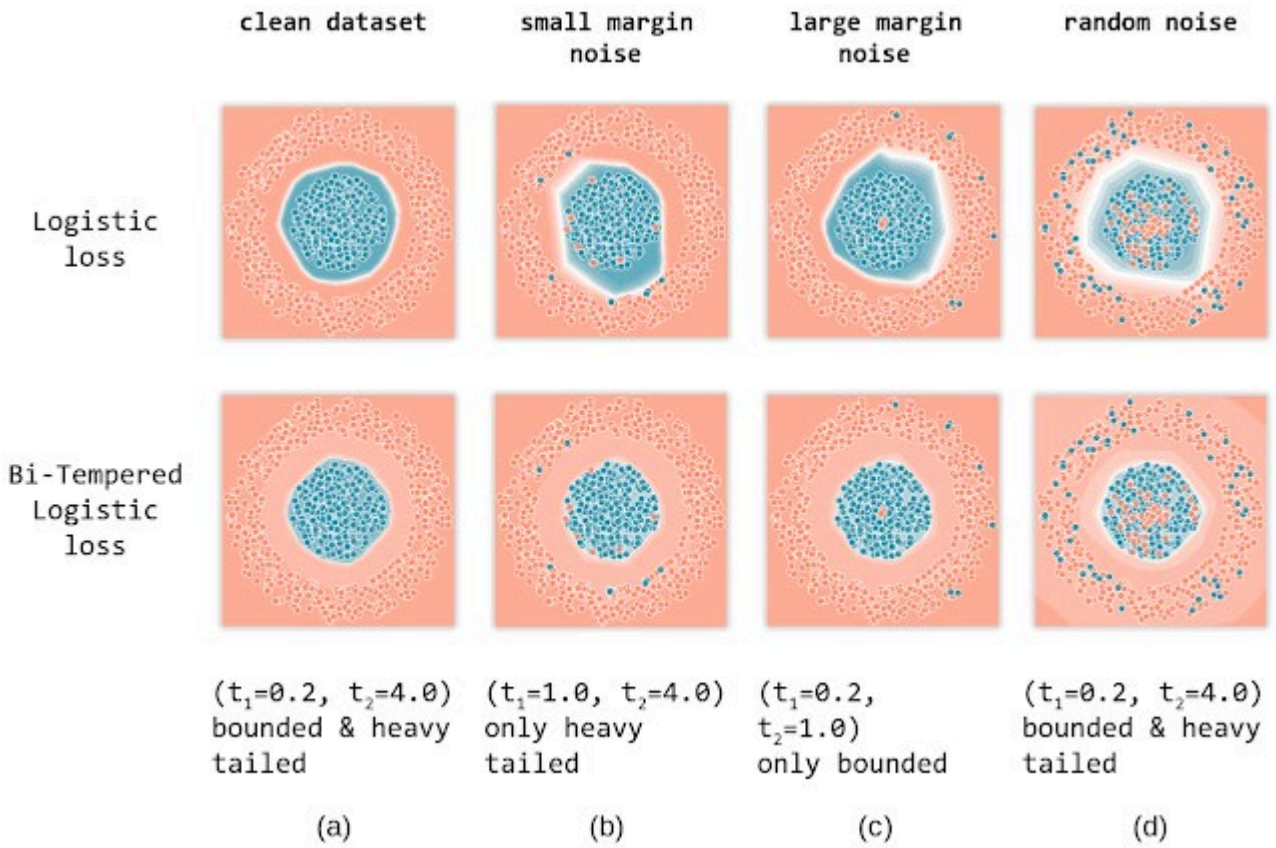
너무 0으로 사라지는 문제가 발생

출처 : https://ai.googleblog.com/2019/08/bi-tempered-logistic-loss-for-training.html

3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

Bi-Tempered Loss



출처 : https://ai.googleblog.com/2019/08/bi-tempered-logistic-loss-for-training.html

3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

Noisy Label에 대해서 잘 학습하는 로스!!

Taylor Cross Entropy Loss (Can Cross Entropy Loss Be Robust to Label Noise?)

출처 : https://www.ijcai.org/Proceedings/2020/0305.pdf

3. 대회를 통해서 배운점

• (7) 다양한 Loss Function과 Label Smoothing 기법들

Noisy Label에 대해서 잘 학습하는 스무딩 기법!!

Label Smoothing

$$\mathbf{y}_k^{\mathrm{LS}} = \mathbf{y}_k(1-\alpha) + \alpha/K \qquad H(\boldsymbol{y}, \boldsymbol{p}) = \sum_{k=1}^{K} -y_k^{LS} \log(p_k)$$

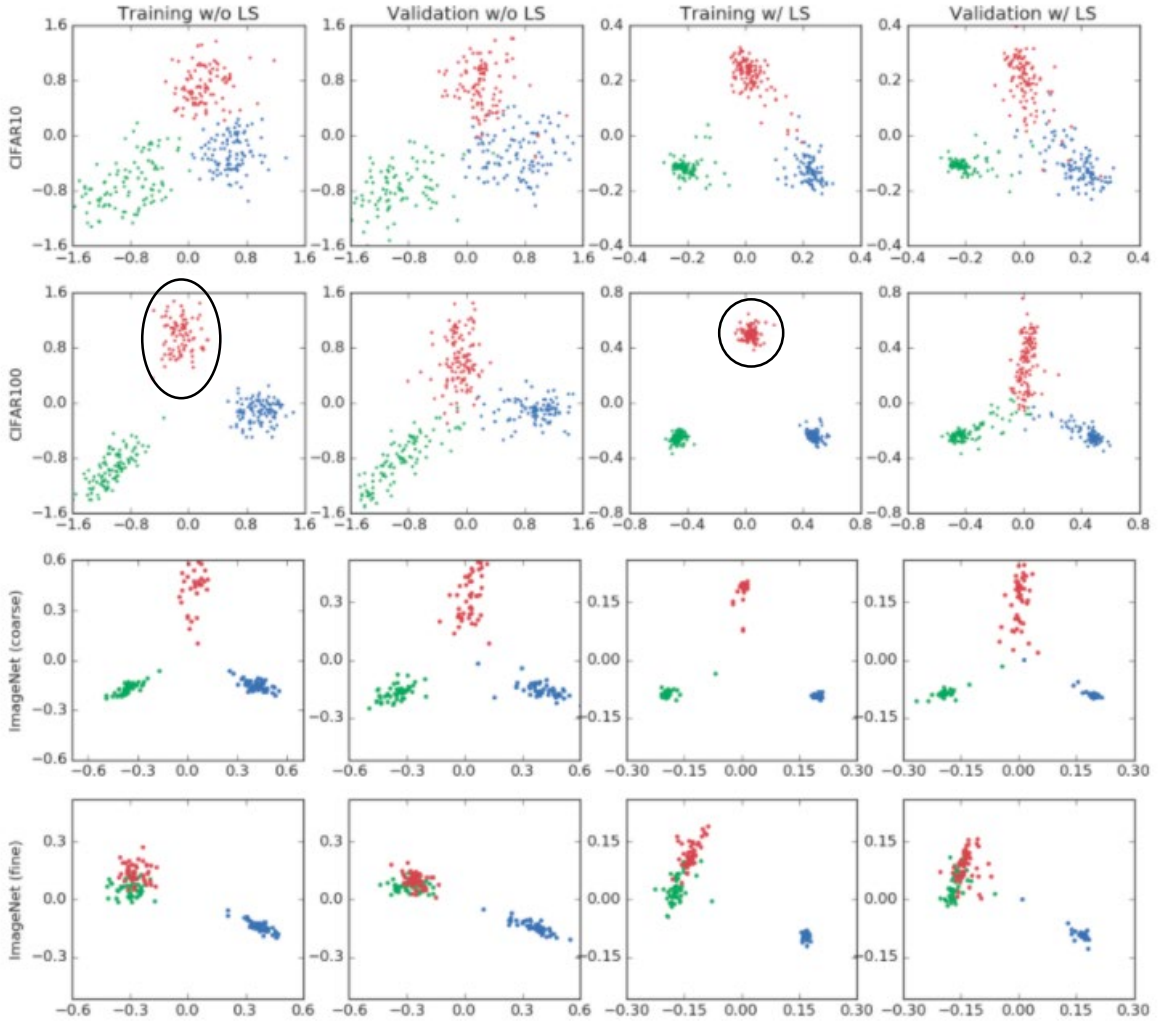
$[0, 1, 0, 0]$

$[0.025, 0.925, 0.025, 0.025]$

라벨을 One-Hot Encoding이 아닌 스무딩을 적용한 Soft Target으로 변경

- 문제 : hard target은 x를 정답과 가깝게 하는 것에만 관심
- 해결 : 스무딩을 줘서 x를 정답과 가깝게 하면서, 오답 템플릿과 동일한 거리에 있도록 멀게 만들어줌
- 시각화된 이미지로 살펴보면 라벨 스무딩을 적용한 경우 같은 라벨들이 묶여있고 라벨들간의 거리가 더 먼 경향이 있음
- 즉, 정답 라벨을 과잉 확신하는 형상을 해소함으로서 최대 로짓이 다른 로짓보다 지나치게 커지는 것을 방지함

그림1 LABEL SMOOTHING 시각화



참고자료 : https://ratsgo.github.io/insight-notes/docs/interpretable/smoothing

3. 대회를 통해서 배운점

(7) 다양한 Loss Function과 Label Smoothing 기법들

Noisy Label에 대해서 잘 학습하는 스무딩 기법!!

Label Smoothing

```python
# ====================================================
# Label Smoothing
# ====================================================
class LabelSmoothingLoss(nn.Module):
    def __init__(self, classes=5, smoothing=0.0, dim=-1):
        super(LabelSmoothingLoss, self).__init__()
        self.confidence = 1.0 - smoothing
        self.smoothing = smoothing
        self.cls = classes
        self.dim = dim
    def forward(self, pred, target):
        pred = pred.log_softmax(dim=self.dim)
        with torch.no_grad():
            true_dist = torch.zeros_like(pred)
            true_dist.fill_(self.smoothing / (self.cls - 1))
            true_dist.scatter_(1, target.data.unsqueeze(1), self.confidence)
        return torch.mean(torch.sum(-true_dist * pred, dim=self.dim))
```

tensor([3])

라벨이 [0, 0, 0, 1, 0]이 아니라 [0.0125, 0.0125, 0.0125, 0.95, 0.0125]으로 변경

tensor([[-0.0189, -0.1481, 0.3698, -0.1186, -0.1509]])

tensor([[0.0125, 0.0125, 0.0125, 0.0125, 0.0125]])

tensor([[0.0125, 0.0125, 0.0125, 0.9500, 0.0125]])

tensor([[0.0125, 0.0125, 0.0125, 0.9500, 0.0125]]) * tensor([[-0.0189, -0.1481, 0.3698, -0.1186, -0.1509]] = tensor([[0.0180, 0.0197, 0.0219, 1.4491, 0.0225]])

참고자료 : https://www.kaggle.com/piantic/train-cassava-starter-using-various-loss-funcs

3. 대회를 통해서 배운점

(8) 이미지의 이상치를 제거하는 방법 – 노이즈

Clean Lab 기반의 노이즈 제거

[Clean Label]

[Noise Label]

참고자료 : https://www.kaggle.com/telljoy/noisy-label-eda-with-cleanlab

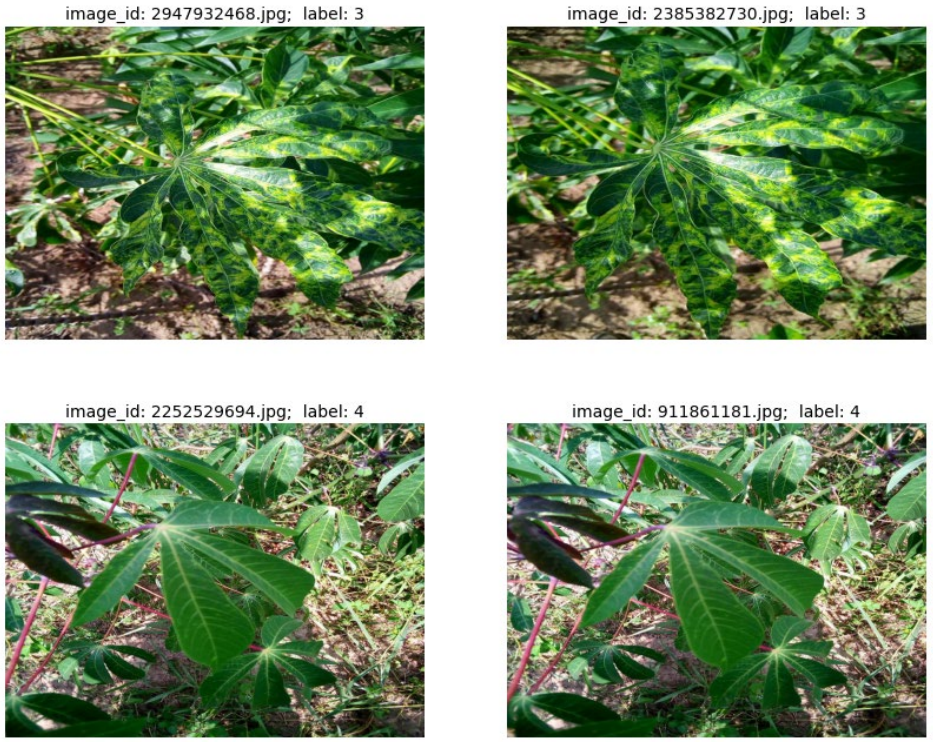
3. 대회를 통해서 배운점

(8) 이미지의 이상치를 제거하는 방법 – 중복 이미지

GlobalAveragePool -> DBSCAN을 통해서 중복 이미지 찾는 방법

```
def build_model():
    with strategy.scope():
        pretrained_model = model_selector[MODEL](input_shape=(*IMAGE_SIZE, 3),
                                                 weights='imagenet',
                                                 include_top=False
                                                 )
        inp = tf.keras.layers.Input(shape=(*IMAGE_SIZE, 3))
        x = pretrained_model(inp)
        out = tf.keras.layers.GlobalAveragePooling2D()(x)
        model = tf.keras.Model(inputs=inp, outputs=out)
        return model
```

```
clusters = defaultdict(list)
for image_name, cluster_id in zip(image_ids, DBSCAN(eps=3.0, min_samples=1, n_jobs=4).fit_predi
ct(embs)):
    clusters[cluster_id].append(image_name)
```

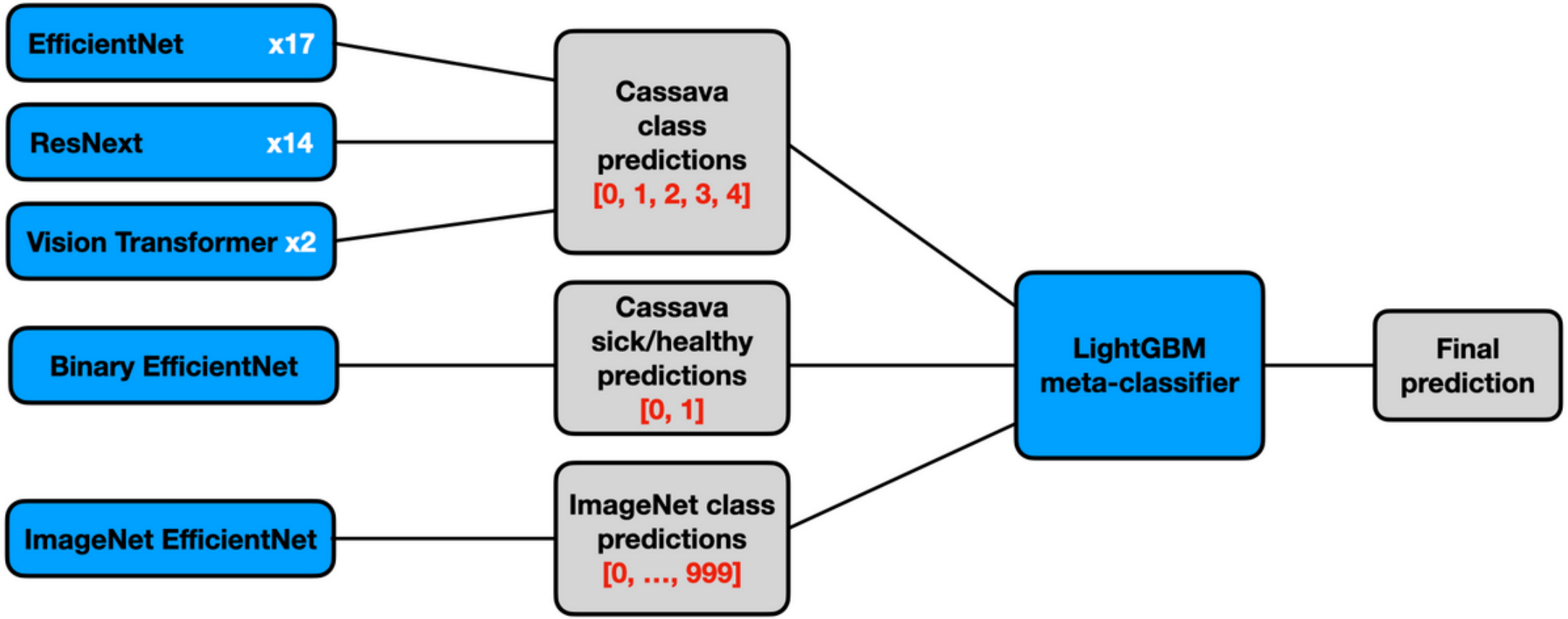


참고자료 : https://www.kaggle.com/graf10a/cldc-image-duplicates-with-dbscan

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

2-Stage GBM Strategy (Stacking) – 14th, 34th, 37th



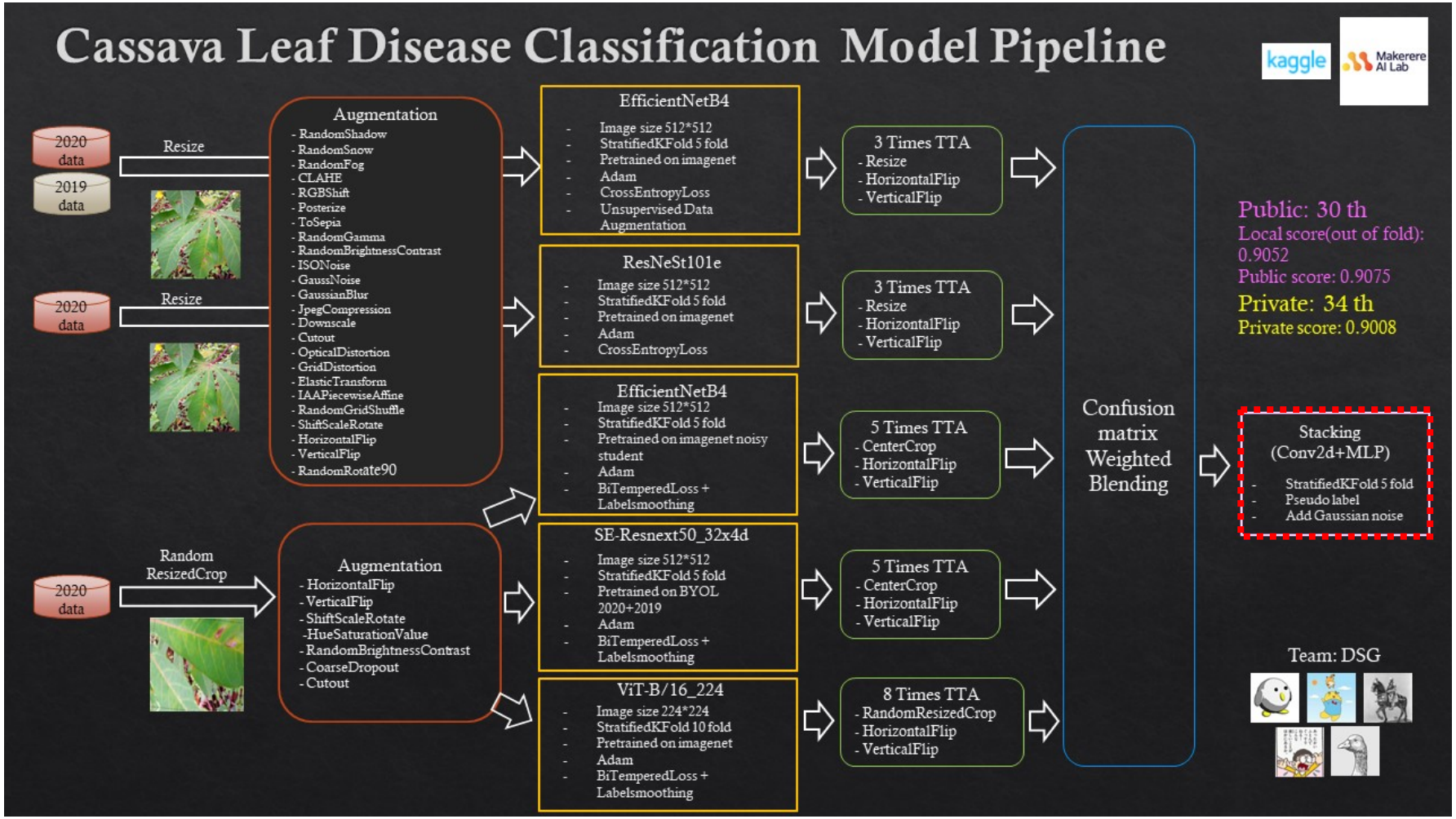
참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/220751

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

출처 : https://www.kaggle.com/anonamename/cassava-emsemble-v2-stacking?scriptVersionId=54065022

2-Stage GBM Strategy (Stacking) – 14th, 34th, 37th



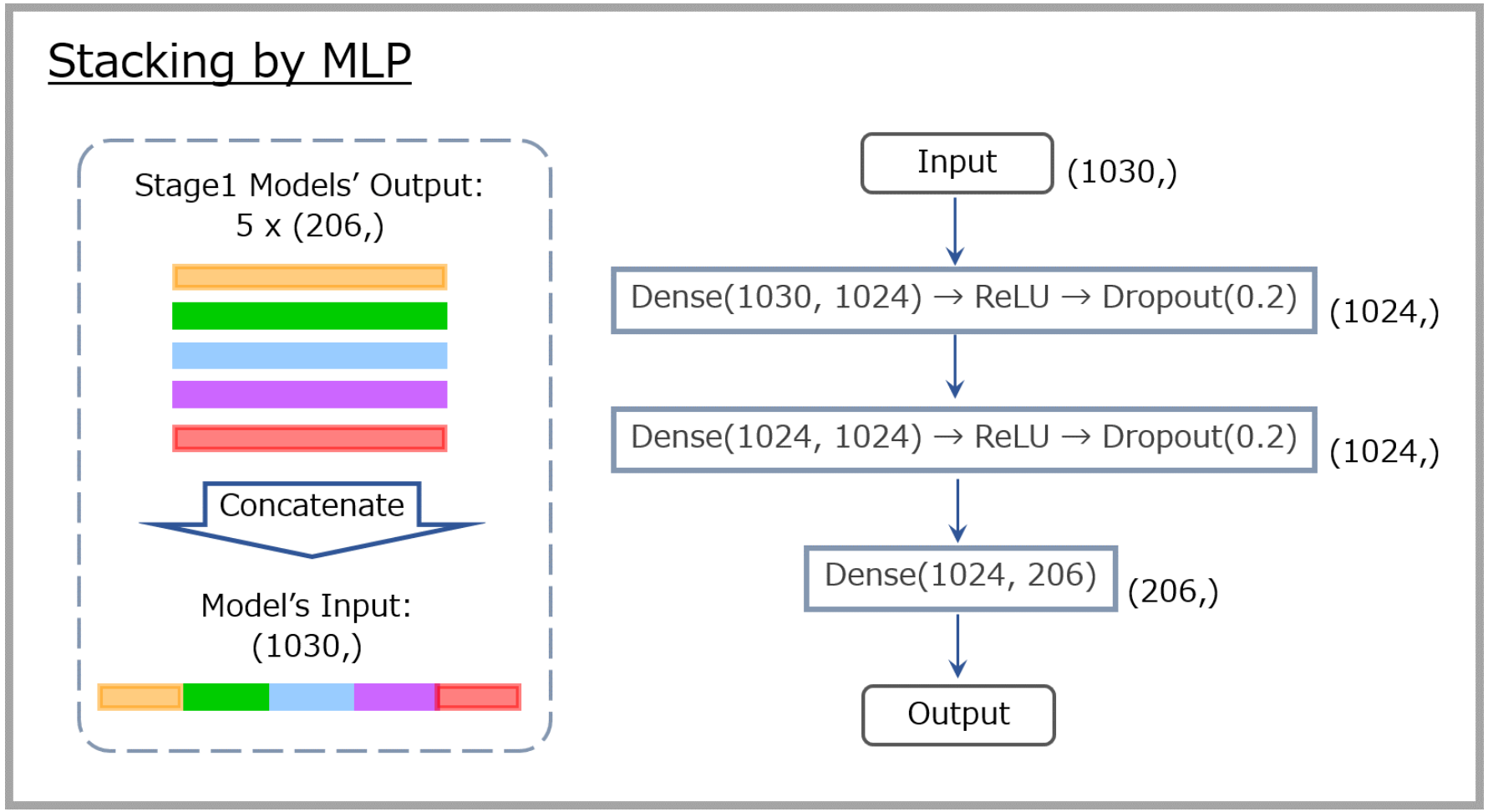
참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/220749

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

출처 : https://www.kaggle.com/ttahara/infer-cassava-ens10

2-Stage GBM Strategy (Stacking) – 14th, 34th, 37th



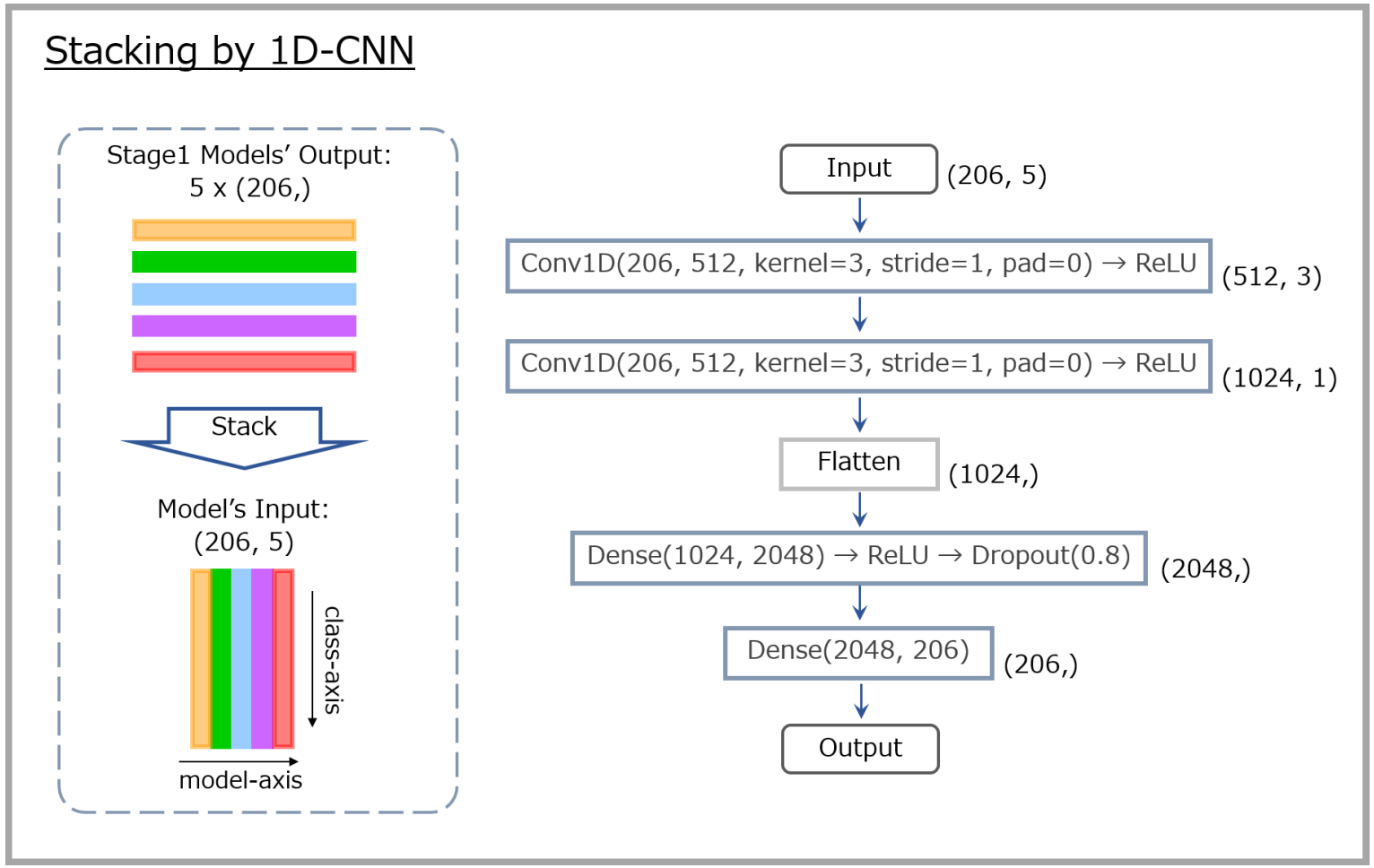
참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/220969

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

출처 : https://www.kaggle.com/ttahara/infer-cassava-ens10

2-Stage GBM Strategy (Stacking) – 14th, 34th, 37th



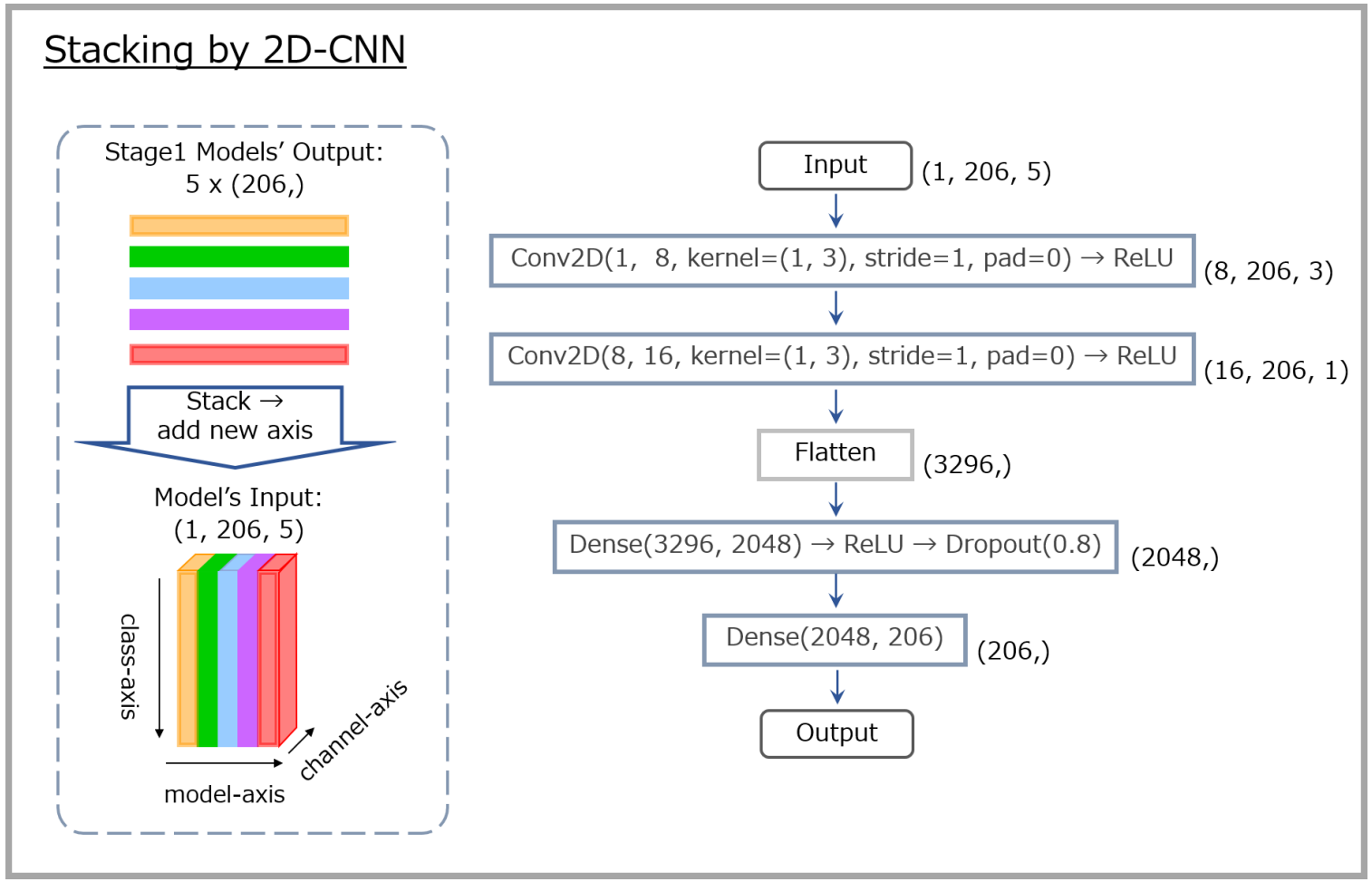
참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/220969

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

출처 : https://www.kaggle.com/ttahara/infer-cassava-ens10

2-Stage GBM Strategy (Stacking) – 14th, 34th, 37th



참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/220969

3. 대회를 통해서 배운점

(9) OOF를 활용한 모델링

Model Ensemble Strategy (1등 솔루션)

- 전략 : Prior optimization of the weights of the softmax logits of each model
- 9개의 모델을 가지고 1개 사용한 경우에서부터 9개 전부 사용한 경우까지 테스트

```
combined = []
for i in range(len(columns)):
    combined.append(list(combinations(columns, i+1)))

def evaluate_ensemble(df, columns):
    return df[[*columns]].apply(lambda x: np.argmax([np.sum(v) for v in zip(*[x[c] for c in columns])]), axis=1).values

results = dict()
with tqdm(total=len(list(chain(*combined)))) as process_bar:
    for c in list(chain(*combined)):
        process_bar.update(1)
        results[c] = accuracy_score(oof_predictions_v3.label.values, evaluate_ensemble(oof_predictions_v3, c))
```

Best Combination

```
# Get top-50 combinations
{k: results[k] for k in sorted(results, key=results.get, reverse=True)[0:50]}
```

```
{('mobilenet', 'b4', 'vit_resnext'): 0.9103145300743095,
 ('mobilenet', 'b3', 'vit_resnext'): 0.9092863485535355,
```

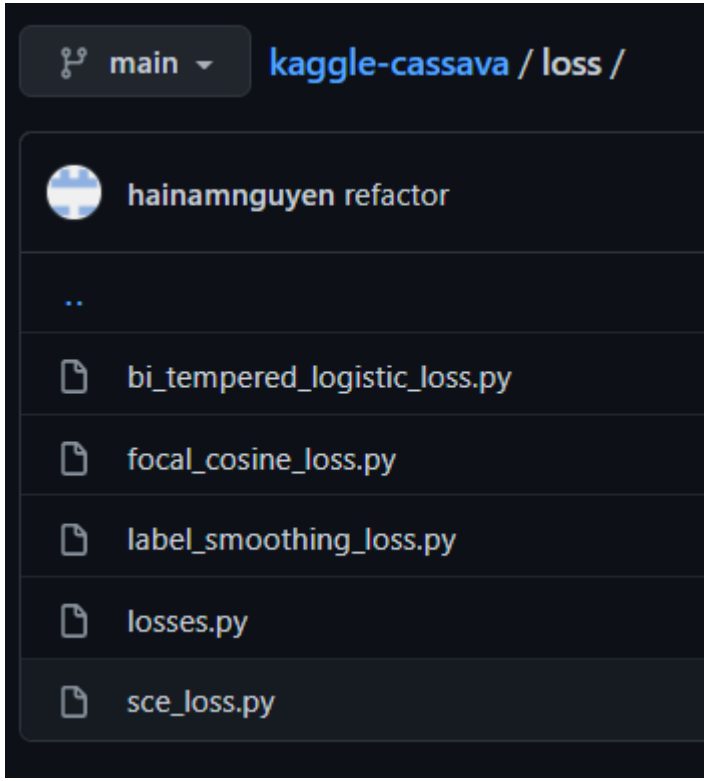
- ResNext와 ViT를 묶은 이유는 기존 실험에서 성능이 좋아다고함

3. 대회를 통해서 배운점

(10) 코드 정리하기

대회 끝난 후에 코드 정리하기

출처 : https://github.com/freedom1810/kaggle-cassava/



참고자료 : https://www.kaggle.com/c/cassava-leaf-disease-classification/discussion/221113

3. 대회를 통해서 배운점

(11) 다양한 기법들

Minyong Shin

Data Scientist at NRISE

Seoul, Seoul, South Korea

Joined 3 years ago · last seen in the past day

https://shinminyong.tistory.com/

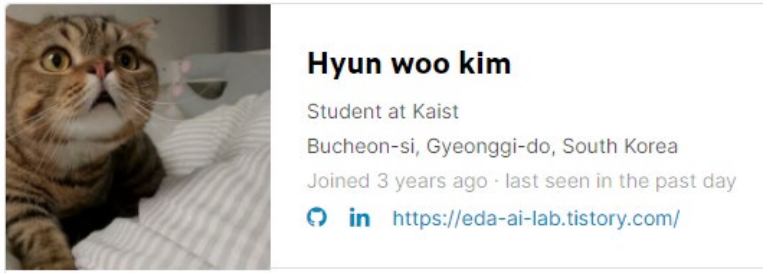
다양한 Augmentation 기법들과 방법들

- 여러가지 방면으로 실험해보면서 Best Augmentation을 탐색하는 과정
- 여러 optimizer, loss function, 스케쥴러에 대해서 공부한점
  - 아직 부족한게 너무 많음
- Machine learning과 약간 다른 서로 다른 모델에 대한 앙상블 기법
- TTA
- 기존에 접하거나 사용해보지 못했던 모델에 대한 공부
  - VIT
  - resnet
  - efficientnet
- image model training ~ inference의 전과정을 배우게 된 것

Name	테스트 Aug	inference score
aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.881
Augmentation zero(resize)	Randomresizedcrop(512)	0.877
aug1 + Blur	horizontalflip(0.5) Randomresizedcrop(512) Normalize blur	0.888
aug1 + ROTATE	horizontalflip(0.5) Randomresizedcrop(512) Normalize rotate	0.886
aug1 + CLAHE	horizontalflip(0.5) Randomresizedcrop(512) Normalize clahe	0.883
aug1 + brightcontrast	horizontalflip(0.5) Randomresizedcrop(512) Normalize RandomBrightnessContrast	0.886
aug1 + Sharpen	horizontalflip(0.5) Randomresizedcrop(512) Normalize Sharpen	0.886
aug1 + Blur + brightcontrast	horizontalflip(0.5) Randomresizedcrop(512) Normalize RandomBrightnessContrast blur	0.885
Aug1 + Blur + gridmask		0.882
Aug1 + Blur + transpose		0.891
Aug1 + Blur + transpose + vertical		0.888
heroseo Aug	horizontalflip(0.5) Randomresizedcrop(512) Normalize ShiftScaleRotate verticalflip Transpose	0.890
Aug1 + Blur + transpose + coarsedropout		0.891
Aug1 + Blur + transpose + ShiftScaleRotate		0.890
ViT(base+patch16) + aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.886
ViT(base+patch32) + aug1	horizontalflip(0.5) Randomresizedcrop(512) Normalize	0.875

4. 아쉬웠던 점

(1) GPU 부족 및 생각의 부족



GPU 부족 및 생각의 부족으로 코드는 만들었지만 실험은 하지 못했었던 내용들

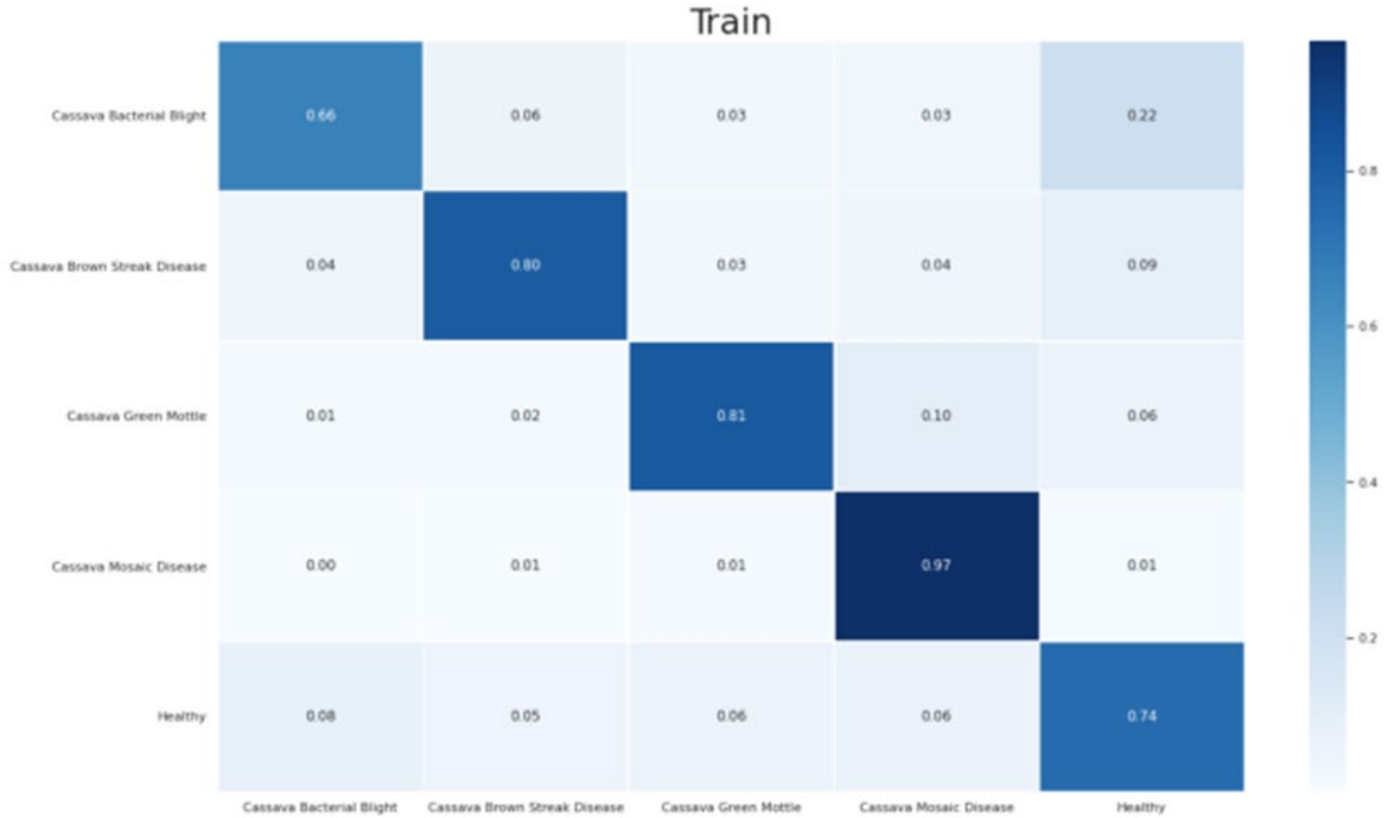
- NIPA 지원을 받고있었는데 1월부터 지원이 종료되었음 (ㅠㅠ)
- 실험1. 초기 몇개의 에폭은 간단한 Augmentation을 진행하고 후반의 에폭은 mixup이나 fmix와 같은 복잡한 Augmentation을 수행하는 방법
- 실험2. OOF를 이용해서 LGBM과 같은 머신러닝 모델 돌리는 방법 : 2019년도가 같이 사용되는 모델도 있고 그렇지 않은 모델도 있어서 Validation이 다르니 따로 OOF를 저장안했는데, 다른 솔루션 중에 2020년도만 Validation으로만 사용해서 OOF를 모으고 해당 값들로 CV 체크와 머신러닝 모델 적용한 솔루션이 있었음
- 실험3. 실험2에서 Validation에 대한 생각을 잘못해서 현재 모델에 대한 분석을 제대로 하지 못한 점

4. 아쉬웠던 점

(1) GPU 부족 및 생각의 부족

GPU 부족 및 생각의 부족으로 코드는 만들었지만 실험은 하지 못했었던 내용들

- 실험3. 실험2에서 Validation에 대한 생각을 잘못해서 현재 모델에 대한 분석을 제대로 하지 못한 점 : OOF를 저장하지 않아서 아래의 사진처럼 현재 모델에 대한 분석과 앙상블시에 어떻게 해야하는지 정확한 의미를 파악하지 못하고 사용했던 문제가 있음



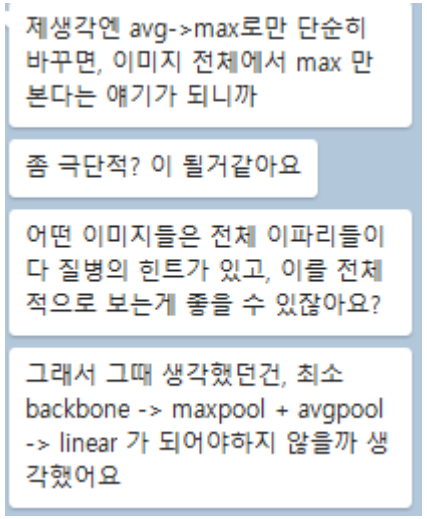
4. 아쉬웠던 점

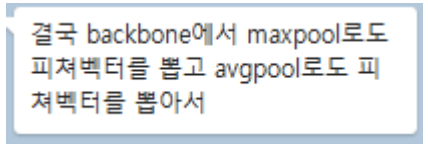
(1) GPU 부족 및 생각의 부족

GPU 부족 및 생각의 부족으로 코드는 만들었지만 실험은 하지 못했었던 내용들

- 실험4. 모델에 대한 이해를 통해서 모델을 직접 수정을 많이 못해본 점
  - 이미지의 특성상 중앙에만 나오는게 있고 전체 걸쳐서 나오는게 있어서 Avg + Max Pooling을 통해서 실험해봤지만 점수는 하락했음

  - 점수는 하락했지만 이미지의 특성을 모델에 반영해봤던 좋은 아이디어였음






```
[14] # print(model)
     # print(vars(model).keys())
     timm.list_modules()
     model.global_pool = SelectAdaptivePool2d(pool_type='max', flatten=True)
     print(model)

[16] # Concate two vectors, maxpool + avgpool
     model.global_pool = nn.Identity()
     model.classifier = nn.Identity()
     print(model)

[24] Input = torch.rand(5,3,512,512)
     out = model(Input)
     print(out.size())

     torch.Size([5, 1792, 16, 16])

[25] max_out = SelectAdaptivePool2d(pool_type='max', flatten=True)(out)
     avg_out = SelectAdaptivePool2d(pool_type='avg', flatten=True)(out)
     max_and_avg = torch.cat((max_out, avg_out), 1)
     print(max_and_avg.size())

     torch.Size([5, 3584])

[26] n_features = max_and_avg.size(1)
     n_class=5
     out = nn.Linear(n_features, n_class)(max_and_avg)
     print(out.size())

     torch.Size([5, 5])
```

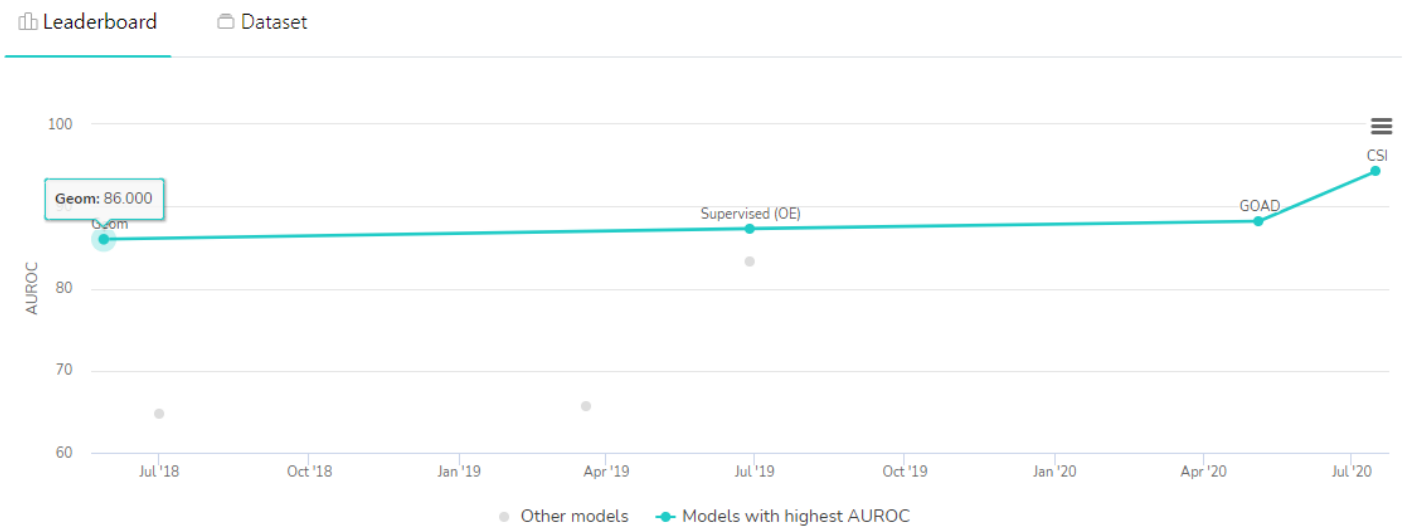
4. 아쉬웠던 점

(2) 하지 못했던 점

https://www.kaggle.com/hihunjin

Label noise가 있는 data를 anomaly로!

- https://paperswithcode.com/sota/anomaly-detection-on-one-class-cifar-10



View AUROC All models Edit

RANK	MODEL	AUROC	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	CSI	94.3	×	CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances			2020
2	GOAD	88.2	×	Classification-Based Anomaly Detection for General Data			2020
3	Supervised (OE)	87.3	✓	Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty			2019

4. 아쉬웠던 점

(2) 하지 못했던 점

Learning with Label Noise

- 논문 기법들 적용해볼껄 ㅠㅠ
  - https://arxiv.org/pdf/2012.04193.pdf
- anomaly detection wrong labeled data 으로 구글 검색

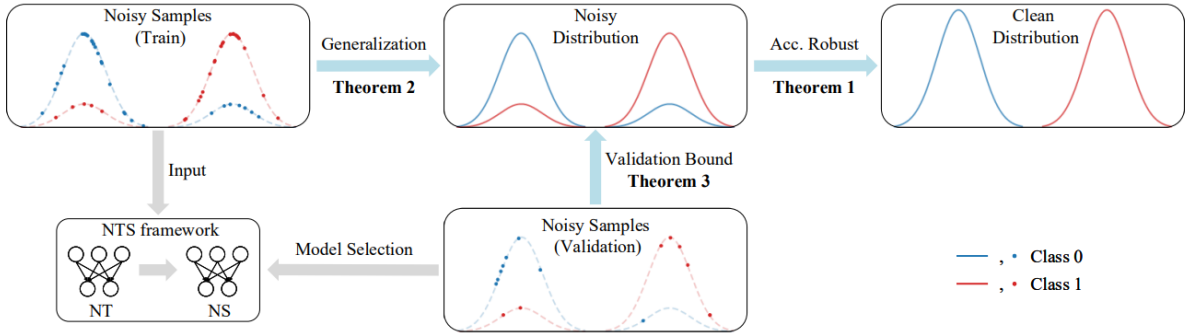
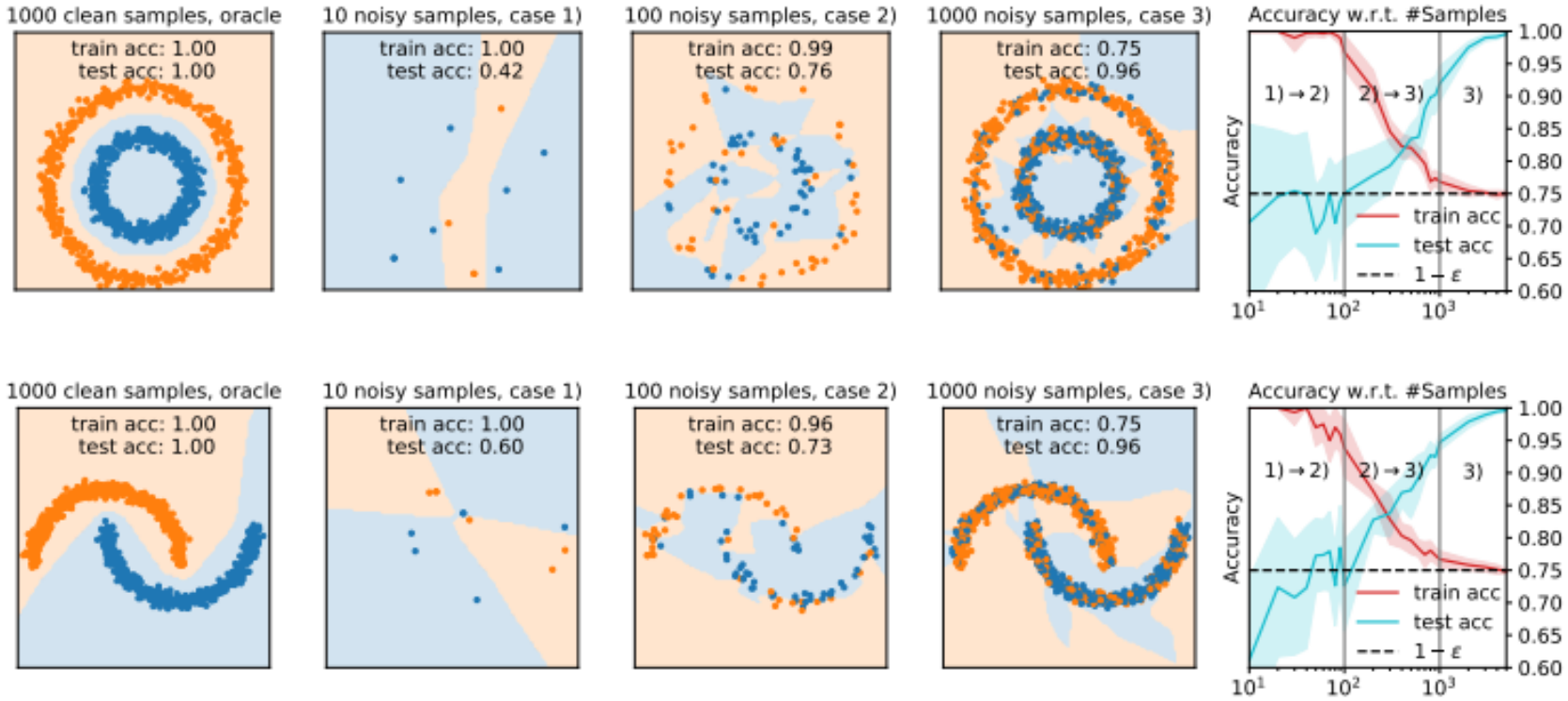


Figure 1: A systematical illustration of learning with noisy labels.



4. 아쉬웠던 점

(2) 하지 못했던 점

다양한 Metric에서 성능 확인

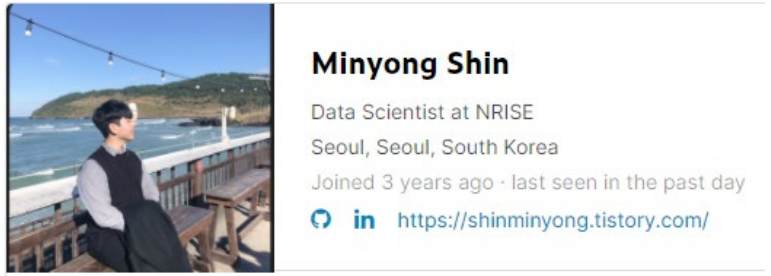
- ROC-AUC
- F1
- precision/recall
- 등등..

그 외

- 1 Epoch을 Private에서 수도 라벨링하는 전략 (회의때 얘기나왔는데 해보지 않았음)
- Under Sampling

4. 아쉬웠던 점

(3) 부족했던 점



여러가지 부족했던 점들

- 여러 테스트(모델, Augmentation, parameter)에 대한 로깅
- 초반 GPU 리소스 문제를 적극적으로 해결하지 못했던 것
  - model 및 weight를 캐글 노트북 커널로 체크포인트 형식처럼 저장하고 불러오는 식으로 하면 시간은 꽤 소요되겠지만 부분학습이 아닌 전체를 모두 학습할 수 있었을 듯
- 좀 더 deep하게 parameter을 수정하면서 대회에 참가하지 못한 점
- 건강상의 문제 + 회사 채용으로 본 스터디 중반~막판에 참여하지 못한 부분