부스트캠프

Wrap-Up 리포트

Stage 1 - Image Classificaiton

이름 : 김동우

캠퍼ID : T1017

◆ 본인의 점수 및 순위

- 리더보드 public accuracy : 80.5873%, f1 : 0.7622, 42등
- 리더보드 private accuracy : 79.8571%, f1 : 0.7403, 62등

◆ 사용한 기능들

- **Model** : ResNet50, ResNet101, EfficientNet b3
- **Loss** : CrossEntropy loss, Focal loss, F1 loss
- **optimizer** : Adam
- **Train, validation split** : stratified split, stratified k-fold
- **metric** : accuracy, f1 score, sklearn f1 score
  (skl f1은 baseline code에서 사용된 f1 score과 다른 값을 출력하여 사용하였다.)
- **Ensemble** : Soft voting
- **Age Filter**
- **Train Time Augmentation (albumentations)** :
  1. Resize(512, 384, p=1.0)
  2. CenterCrop(400, 300, p=1.0)
  3. HorizontalFlip(p=0.5)
  4. ShiftScaleRotate(p=0.7)
  5. RandomBrightnessContrast(brightness_limit=(-0.1, 0.3),
     contrast_limit=(-0.1, 0.3), p=0.5)
  6. GaussNoise(p=0.7)
  7. Normalize(mean=(0.560, 0.524, 0.501), std=(0.233, 0.243, 0.246),
     max_pixel_value=255.0, p=1.0)
  8. ToTensorV2(p=1.0)
- **Test Time Augmentation**
  1. original
  2. HorizontalFlip
  3. ShiftScaleRotate
  4. RandomBrightnessContrast

◆ 사용한 hyper parameters

- **Percentage of Validation set** : 0.3, 0.0 (모든 이미지를 training set으로 사용)
- **K-Fold** : 4
- **Seed** : 7, 33
- **Learning Rate** : 1e-4, 5e-4
- **Batch Size** : 32
- **Age Filter** : 57, 58

◆ Submission Timeline

~ 2021.04.04

평일에는 강의를 따라서 step by step으로 주어진 주제의 코드를 짜서 제출하지 못했고 주말에 만든 코드로 한번 학습해보았다.

제출만 해보자는 식으로 seed 설정도 안하고 [ResNet50, batch size=32, val split=0.3, lr=5e-4, CE Loss]로 하이퍼 파라미터 튜닝을 한 뒤, 위의 Train Time Augmentation에서 CenterCrop을 제외한 transform을 하여 학습하였다.

처음에는 한 사람에 대하여 마스크 쓴 이미지, 잘 못 쓴 이미지, 안 쓴 이미지의 비율이 5:1:1이기 때문에 이 imbalance를 해결하고자 down sampling으로 학습하였다.
(한 사람에 대해서 mask 쓴 이미지는 랜덤으로 하나만 사용하도록 하였다.)

→ **초기 설정 : ResNet50, batch size=32, val split=0.3, lr=5e-4, CE Loss, Train Time Augmentation(CenterCrop 제외), down sampling**

하지만 성능이 좋지 않아 down sampling 없이 학습하여 제출하였고 리더보드의 top f1은 0.72가 나왔다.
(Description에 상세한 설명을 적지 못해 epoch은 모른다.)

→ **설정 변경 : down sampling 삭제**
→ **최고 LB f1 : 0.72**

2021.04.05

이미지 사이즈를 줄여 학습 속도를 증가시키고자 Train Time Augmentation에서 Resize 대신 CenterCrop(400,300)을 사용하였다.

그리고 down sampling을 안쓰기 때문에 data imbalance를 반영하고자 Loss를 Focal loss, F1 loss를 사용하여 학습해보았다.

1. LB f1 score : < 0.5
   a. model : **ResNet101**
   b. Loss : **Focal Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
   f. Hyperparameters : p_val=0.3, lr=5e-4, batch=32, seed=7

2. LB f1 score : < 0.5
   a. model : **ResNet101**
   b. Loss : **F1 Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
   f. Hyperparameters : p_val=0.3, lr=5e-4, batch=32, seed=7

3. LB f1 score : < 0.5
   a. model : ResNet50
   b. Loss : **Focal Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
   f. Hyperparameters : p_val=0.3, lr=5e-4, batch=32, seed=7

→ **설정 변경 : Resize를 CenterCrop으로 변경, ResNet101 추가,**
Focal loss/F1 loss 사용,

→ **최고 LB f1 : 0.72**

CenterCrop을 사용하고 Loss를 바꾸었더니 성능이 매우 안좋아졌다.

CenterCrop은 필요 없는 배경을 지워주는 것이므로 Loss일 것이라 생각하여 CE loss를 사용해보았다. 그리고 k-fold를 사용하였다.

4. LB f1 score : 0.6223

a. model : ResNet101
b. Loss : **CE Loss**
c. Optimizer : Adam
d. Train, validation split : **k-fold**
e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
f. Hyperparameters : k=4, lr=5e-4, batch=32, seed=7,
g. Fold, Epoch : **fold=0,** epoch=24
h. Metric : train f1 = 0.577, val f1 = 0.582, skl f1 = 0.965

5. LB f1 score : 0.5923

a. model : ResNet101
b. Loss : **CE Loss**
c. Optimizer : Adam
d. Train, validation split : **k-fold**
e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
f. Hyperparameters : k=4, lr=5e-4, batch=32, seed=7,
g. Fold, Epoch : **fold=1**, epoch=26
h. Metric : train f1 = 0.580, val f1 = 0.586, skl f1 = 0.967

6 LB f1 score : 0.6050

a. model : ResNet101
b. Loss : **CE Loss**
c. Optimizer : Adam
d. Train, validation split : **k-fold**
e. Augmentation : Train Time Augmentation에서 Resize만 사용 x
f. Hyperparameters : k=4, lr=5e-4, batch=32, seed=7,
g. Fold, Epoch : **fold=2**, epoch=20
h. Metric : train f1 = 0.573, val f1 = 0.592, skl f1 = 0.986

7. LB f1 score : 0.6237

a. model : ResNet101

b. Loss : **CE Loss**

c. Optimizer : Adam

d. Train, validation split : **k-fold**

e. Augmentation : Train Time Augmentation에서 Resize만 사용 x

f. Hyperparameters : k=4, lr=5e-4, batch=32, seed=7,

g. Fold, Epoch : **fold=3**, epoch=25

h. Metric : train f1 = 0.573, val f1 = 0.587, sk f1 = 0.970

8. LB f1 score : 0.68

a. ensemble result 4~7 (soft vote)

→ **설정 변경 : CE loss 사용, k-fold 사용**

→ **최고 LB f1 : 0.72**

처음으로 metric으로 f1을 사용해보았는데, baseline code의 f1이 너무 낮아 skl f1과 같이 사용하였다. 두 f1 모두 같은 macro f1으로 알고 있는데 값이 너무 달라 혼란스러웠다.

val f1이 가장 높은 것은 항상 epoch이 높았다. LB f1을 보면 분명 과적합인데, 왜 낮을까 생각해보았다.

앙상블의 결과는 좋았다. 앙상블 전 모델들에 비해 f1이 큰 폭 상승하였다.

2021.04.06

과적합 문제인지 확인하기 위해 04.05에서 학습한 모델 4~7의 결과 중 epoch 이 10 이하인 결과 중 가장 좋은 결과로 앙상블 해보았다.

1. LB f1 score : 0.68

a. fold 0:epoch 8 / fold 1:epoch 8 / fold 2:epoch 9 / fold 3:epoch 10

> **ensemble** (soft vote)

→ **설정 변경 : 낮은 epoch으로 fold 앙상블**

→ **최고 LB f1 : 0.72**

너무 깊은 모델 때문이지 않을까란 생각에 ResNet50으로 학습하고 제출하던 도중, 큰 실수를 발견하였다. **Train Time Augmentation은 CenterCrop(400,300)으로 바꾸었지만 Test Time Augmentation에서는 아직도 Resize(512, 384)로 inference를 하고 있었다.**

따라서, 어제 한 학습은 성능이 좋을 수가 없었다.

Tes Time Augmentation을 수정하고 모델만 ResNet50으로 바꾸어 학습하고 fold에서 최선의 모델만 앙상블해보았다.

2. LB f1 score : 0.6696

a. **ensemble** best result of **ResNet50**, each fold

→ **설정 변경 : ResNet50 사용**

→ **최고 LB f1 : 0.72**

fold 중 validation f1이 가장 높은 모델의 LB f1은 0.6777 인 것에 비해 앙상블 모델의 f1은 향상되지 않았다. 앙상블이 항상 좋은 결과를 내는 것은 아니었다.

2021.04.07

결국 k-fold나 CenterCrop을 사용하지 않고 학습한 모델의 성능이 가장 좋았기 때문에 k-fold와 Train Time Augmentation을 진행하지 않고 Resize와 Normalize만 하여 학습해보기로 했다.

그리고 정교한 수렴을 위해 learning rate도 줄여보았다.

1. LB f1 score : 0.6633

a. model : ResNet50
b. Loss : **focal Loss**
c. Optimizer : Adam
d. Train, validation split : stratified split
e. Augmentation : **Resize, Normalize**
f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7,
g. Fold, Epoch : epoch=13
h. Metric : train f1 = 0.576, val f1 = 0.577, skl f1 = 0.975

2. LB f1 score : 0.7018

a. model : ResNet50
b. Loss : **focal Loss**
c. Optimizer : Adam
d. Train, validation split : stratified split
e. Augmentation : **Resize, Normalize**
f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7,
g. Fold, Epoch : epoch=17
h. Metric : train f1 = 0.579, val f1 = 0.576, skl f1 = 0.977

→ **설정 변경 : Focal loss 사용, CenterCrop 사용 x, Resize/Normalize만 사용, k-fold 사용 x, lr 변경(5e-4 > 1e-4)**

→ **최고 LB f1 : 0.72**

학습 데이터를 늘려서 학습해보고 싶어 위의 hyperparameter을 가지고 validation set split 없이 모든 데이터를 training set으로 하여 학습해보았다.

3. LB f1 score : 0.6821

a. model : ResNet50
b. Loss : focal Loss
c. Optimizer : Adam
d. Train, validation split : **use all data for training**
e. Augmentation : Resize, Normalize
f. Hyperparameters : p_val=0.0, lr=1e-4, batch=32, seed=7,
g. Fold, Epoch : epoch=17

→ **설정 변경 : 모든 데이터로 학습**
→ **최고 LB f1 : 0.72**

같은 epoch 이어도 데이터가 많기 때문에 과적합의 가능성을 생각해두고 피어세션과 토론 게시판에서 성능이 좋다던 efficientnet b3를 사용해보았다. (b4를 사용하고 싶었으나 out of memory로 b3를 사용하였다.)
또한 focal loss의 성능이 좋은 것 같지 않아 다시 CE loss를 사용하였다.

4. LB f1 score : 0.7204
   a. model : **EfficientNet b3**
   b. Loss : **CE Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Resize, Normalize
   f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7,
   g. Fold, Epoch : epoch=5
   h. Metric : train f1 = 0.595, val f1 = 0.592, skl f1 = 0.965

5. LB f1 score : 0.6864
   a. model : **EfficientNet b3**
   b. Loss : **CE Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Resize, Normalize
   f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7,
   g. Fold, Epoch : epoch=6
   h. Metric : train f1 = 0.592, val f1 = 0.590, skl f1 = 0.987

6. LB f1 score : 0.6895
   a. model : **EfficientNet b3**
   b. Loss : **CE Loss**
   c. Optimizer : Adam
   d. Train, validation split : stratified split
   e. Augmentation : Resize, Normalize
   f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7,
   g. Fold, Epoch : epoch=12
   h. Metric : train f1 = 0.602, val f1 = 0.599, skl f1 = 0.979

→ **설정 변경 : CE loss 사용, efficientnet b3 사용**
→ **최고 LB f1 : 0.72 > 0.7204**

이전에 ResNet의 score을 efficientnet이 처음 넘겼다. 따라서 늦었지만 이 테스크에서 efficientnet이 가장 성능이 좋을 것이라 생각하였다.

또한, epoch이 클수록 validation metric은 크지만 LB f1 score는 epoch이 작고 metric도 작은 것이 더 높았다.

왜 validation set에도 과적합이 일어났나를 생각해보았고 피어 세션을 통해서 사람 별로 validation set split을 안했기 때문이라고 짐작하였다.

이 모델도 어제와 같이 모든 데이터로 학습을 진행해보았다.

7. LB f1 score : 0.6889

a. model : EfficientNet b3
b. Loss : CE Loss
c. Optimizer : Adam
d. Train, validation split : **use all data for training**
e. Augmentation : Resize, Normalize
f. Hyperparameters : p_val=0.0, lr=1e-4, batch=32, seed=7,
g. Fold, Epoch : epoch=4

8. LB f1 score : 0.6341

a. model : EfficientNet b3
b. Loss : CE Loss
c. Optimizer : Adam
d. Train, validation split : **use all data for training**
e. Augmentation : Resize, Normalize
f. Hyperparameters : p_val=0.0, lr=1e-4, batch=32, seed=7,
g. Fold, Epoch : epoch=5

→ **설정 변경 : 모든 데이터로 학습**
→ **최고 LB f1 : 0.7204**

앞에서 말한 모든 데이터를 학습에 사용하면 split한 모델과 같은 epoch이라도 과적합이 될 수 있다는 가설은 맞았다. 하지만 epoch 수를 적게 해도 split한 모델보다는 성능이 좋지 않았다.

2021.04.08

Competition 마지막 날이므로 seed를 바꾸거나 앙상블을 시도해볼 계획이었고 일단 LB 상에서 가장 score가 높은 9개의 모델을 앙상블 해보았다.

1. LB f1 score : 0.7129

a. Ensemble top 9 models (soft vote)

→ 설정 변경 : top 9 모델 사용

→ 최고 LB f1 : 0.72

가장 성능이 좋은 모델의 score는 0.7204인 것에 반해 성능이 향상되지 않았다.

따라서, 앙상블이 아닌 아예 다른 기법을 시도해보았다.

토론 게시판에 나온 age filter를 사용해보았다. 이는 60세 이상의 이미지가 없는 것을 보완하기 위해 학습할 때, n세 이상의 이미지를 60세 이상 클래스 학습에 사용하는 것이다.

먼저 n을 58로 실험해보았다.

2. LB f1 score : 0.7444

a. model : EfficientNet b3

b. Loss : CE Loss

c. Optimizer : Adam

d. Train, validation split : stratified split

e. Augmentation : Resize, Normalize

f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7, **agefilter=58**

g. Fold, Epoch : epoch=6

h. Metric : train f1 = 0.637, val f1 = 0.634, skl f1 = 0.963

3. LB f1 score : 0.7426

a. model : EfficientNet b3

b. Loss : CE Loss

c. Optimizer : Adam

d. Train, validation split : stratified split

e. Augmentation : Resize, Normalize

f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7, **agefilter=58**

g. Fold, Epoch : epoch=8

h. Metric : train f1 = 0.639, val f1 = 0.638, skl f1 = 0.948

→ **설정 변경 : age filter(58) 사용,**

→ **최고 LB f1 : 0.7204 > 0.7444**

무려 0.02 이상의 성능 향상을 보였다.

age filter를 이용하여 resnet50, resnet101을 학습하여 앙상블 해보았다.

4. LB f1 score : 0.7400

a. ensemble resnet50, resnet101, efficientnet b3 (soft vote)

→ **설정 변경 : resnet50, resnet101도 학습**

→ **최고 LB f1 : 0.7444**

efficientnet만 사용했을 때보다 성능이 하락하였다. 아마 resnet50과 resnet101의 성능이 efficientnet보다 좋지 않았는데, 두 모델이 오히려 성능을 떨어뜨린 것 같다.

age filter를 58로 했을 때, 성능이 큰 폭 향상되었는데 그럼 57로 했을 때는 어떨까 궁금해졌다.

5. LB f1 score : 0.6785

a. model : EfficientNet b3
b. Loss : CE Loss
c. Optimizer : Adam
d. Train, validation split : stratified split
e. Augmentation : Resize, Normalize
f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=7, **agefilter=57**
g. Fold, Epoch : epoch=7
h. Metric : train f1 = 0.646, val f1 = 0.632, skl f1 = 0.947

→ **설정 변경 : age filter 변경 (58 > 57)**
→ **최고 LB f1 : 0.7444**

처참한 결과였다. 위 제출로 제출 기회는 한번이 남았고 이 한번을 특별하게 사용하고 싶었다.

그래서 기존의 성능이 좋았던 efficientnet b3와 age filter=58을 사용하였고 위 Training Time Augmentation에 있는 transform을 모두 사용하였으며 새롭게 seed를 변경하고 TTA를 해보기로 하였다.

여기서 **큰 문제가 발견**되었다. TTA를 하는 코드를 작성하던 도중, **기존에 Test Time Augmenation이 Resize로 변경되지 않은 것을 발견**하였다.
위에서 **한번 CenterCrop에서 Resize로 변경하였는데, Test time에는 변경하지 않았던 것**이다.
이로 인해 위의 LB f1 score가 많이 떨어졌을 텐데, 이를 지금 알게되어 아쉬웠다.

6. LB f1 score : 0.7622

a. model : EfficientNet b3
b. Loss : CE Loss
c. Optimizer : Adam
d. Train, validation split : stratified split
e. Augmentation : **Train Time Augmentation에서 Resize만 사용 x**
Test Time Augmentation 사용
f. Hyperparameters : p_val=0.3, lr=1e-4, batch=32, seed=33, **agefilter=58**
g. Fold, Epoch : epoch=10
h. Metric : train f1 = 0.610, val f1 = 0.618, skl f1 = 0.969

→ **설정 변경 : age filter 변경 (57 > 58), seed 변경 (7 > 33), Training Time Augmentation 추가(CenterCrop 제외), TTA 사용**
→ **최고 LB f1 : 0.7622**

◆ 시도했던 것들과 잘된 혹은 잘되지 않은 이유

- 잘된 것

1. **이미지 사이즈의 변화 없이 flip, rotate, noise를 준 Training Time Augmentation**
   → 다양한 이미지의 학습으로 일반화에 도움을 주었다.
2. **EfficientNet 사용**
3. **Age filter**
   → 60세 이상 학습에 가중치를 주어 일반화가 잘 일어난 것 같다.
4. **Test Time Augmentation**
   → test set의 많은 경우의 수에 대해 예측하고 이를 앙상블하여 일반화에 도움이 된 것 같다.

- 잘되지 않은 것

1. **Down sampling**
   → 데이터가 충분하지 않았기 때문에 down sampling은 좋지 않았다.
2. **Center crop**
   → 학습에 도움이 되지 않는 배경 부분을 지운 것 뿐인데, 성능이 왜 떨어졌는지 모르겠다.
3. **Focal loss, F1 loss**
   → imbalanced data에 특화된 loss로 알고 있는데, 왜 성능 향상에 도움이 안되었는지 모르겠다.
4. **Ensemble**
   → 성능이 좋지 않은 모델과 함께 앙상블 하여 성능 향상이 되지 않은 것 같다.
5. **Metric**
   → 사람 별로 training, validation set을 나누지 않다보니 training set에 과적합 되는 것이 validation에도 어느정도 과적합을 일으켜 모델이 일반적으로 성능이 좋은 모델인지 판별하기 쉽지 않았다.
   따라서 제출하기 전까지는 이 모델이 일반화가 잘 되었는지 확인할 수 없었다.
6. **k-fold, 모든 데이터를 학습에 사용**
   → 사람 별로 training, validation set을 나누지 않다보니 k-fold를 하든, 모든 데이터로 학습을 하든, split 한 것과 정보량 차이가 크지 않았던 것 같다.

◆ 시도하지 않았던, 못했던 아이디어와 그 이유

1. **wandb 사용, sweep 기능 (autoML)**
   → wandb를 사용하여 코드를 새로 작성해야 하기 때문에, baseline code를 제대로 만들고 추가할 예정이다.

2. **RandAugment()**
   → RandAugment에는 사람의 얼굴을 학습하기에 적합하지 않은 augmentation 방법도 있었다. 일부 augmentation을 지우고 사용할 수도 있지만, 그러면 나의 augmentation과 다를 것이 없다고 생각했다.

3. **Loss를 섞어 사용하기 (CE loss + f1 loss)**
   → 너무 늦게 알게 된 아이디어이다.

4. **낮은 해상도로 모델 전체를 학습 시킨 뒤, 높은 해상도로 fc layer만 학습**
   → 학습 속도는 빠를 수 있으나 결국 모든 모델을 계속 학습하는 것이 더 성능이 좋을 것이라 생각했다.

5. **CutMix**
   → 일반적으로 사진의 구도가 비슷하여 성능이 좋아질 것이라 예상했지만, 피어 세션/토론 게시판에서 성능 향상이 크지 않다고 하여 사용하지 않았다.

6. **multilabel 사용**
   (마스크 3개, 나이 3개, 성별 2개를 따로따로 하여 8개의 클래스로 진행)
   → 성능이 좋아졌다고 하지만, 위처럼 할 경우 결국 하나만 틀리면 분류가 틀리게 되는 것이기 때문에 좋을 것이라 생각하지 않았다.

7. **New optimizer : AdamP**
   → optimizer 말고 다른 paramter와 기능들이 성능에 어떻게 영향을 주는지 위주로 보았기 때문에 optimizer는 변경하지 않았다.

8. **Learning rate scheduler**
   → 처음에는 크게 필요하지 않을 것이라 생각했지만, 뒤늦게 수렴하지 않는 상황을 도울 것이라 생각하였다.

9. **사람 기준으로 training, validation set split**
   → 처음에는 사람 기준으로 나누는 것이 무슨 효과가 있을까 생각했지만, 이로 인해 metric이나 k-fold, 모든 데이터를 학습에 사용하는 방법이 잘 안 통했던 것을 뒤늦게 알았다.

◆ 아쉬웠던 점과 보완할 점

1. **Baseline Code는 중요하고 필요하다.**

   → Baseline code가 없어서 새로운 기능을 추가할 때마다 코드를 다시 작성하는 시간이 많이 걸렸던 것 같다.

   또한 수정 사항이 있을 때, 완벽하게 수정하지 못하는 단점이 있다. 이번 stage에서도 벌써 2번이나 test time augmentation을 변경하지 못하여 성능도 하락시키고 제출 기회도 날렸다.

2. **training, validation set split은 신중하게 결정해야 한다.**

   → training set과 validation set을 적절하게 나누지 못해 metric을 보고 모델의 성능을 잘 판단하지 못하였다.

   그 외에도 k-fold나 모든 데이터를 학습하는 방법을 사용할 때, 성능 향상을 보지 못한 것도 이 실수가 한몫하였을 것이다.

3. **학습 과정을 사용한 모델, 파라미터 등을 꼭 기록해둬야 한다.**

   → 중간 중간 사용한 파라미터를 기록하지 못하여 이미 학습한 조합을 다시 학습하고 제출하였다. 시간도 버리고 기회도 버리는 비효율적인 진행을 막기 위해서는 파라미터와 그에 따른 성능 기록이 필요하다.

4. **파라미터를 튜닝하는 순서가 어느 정도 필요할 것 같다.**

   → 이번 stage에서 모델을 제외한 파라미터를 튜닝하다가 뒤늦게 이 테스크에 맞는 모델을 찾았었다. 모델이 바뀌다보니 그 전에 이미 해본 파라미터들을 다시 모델에 맞춰 해보아야 했다.

   따라서, 학습을 진행하고 파라미터를 튜닝하는 순서가 어느정도 있어야 할 것 같다. 이번 **competition을 통해 배운 튜닝 순서**는

   (model > loss > augmentation > lr > TTA이나 다른 기능들 추가, seed 변경)

   이다.

5. **일반화는 중요하다.**

   → 더욱 다양한 일반화 방법들을 적용해보지 못해 아쉽다. 일반화는 결국 LB 상에서 score의 향상이며 private까지 포함했을 때, score의 안정성이다. 하지만 이런 일반화 기법을 많이 적용하지 못했고 그랬기에 public score의 절대적인 값도 높지 않았고 private이 포함되었을 때, 하락폭이 큰 것 같다.

◆ 느낀점

소수점의 score 향상에도 엄청난 시도와 시간이 필요하다는 것을 새삼 느꼈다. 또한, 피어분들과의 공유와 대화로 통해 얻는 인사이트는 정말 끝이 없다는 것에 감탄했다.

짧다면 짧고 길다면 긴 2주 동안 끊임없는 아이디어를 얻을 수 있었고 결국 다 시도하지 못하고 competition이 끝이 났지만, 이런 competition을 임할 때의 마음가짐과 자세에 대해 많이 배울 수 있었다.

일빈화의 중요성과 수많은 일반화 기법들을 배울 수 있었다. 그 외에도 이미지를 학습할 때, 어떤 모델과 어떤 augmentation이 사용될 수 있는지 배운 것이 한 분야의 흐름을 배운 것 같아 크게 느껴졌다.

이제 하나가 끝났고 앞으로의 competition이 겁나면서도 기대된다.