Mask

■ Wear □ Incorrect □ Wear

Gender

■ Male □ Female

Age

□ <30 ■ >=30 & <60 □ >=60

Naver Boostcamp 2기 Competitions

마스크 착용, 성별, 나이에 따른 Classification Task

23조 Private 순위 11등(총 38조)

0. Index

1 Introduction

2 Data Analysis

3 Baseline

4 Improvement

5 Conclusion

1. Introduction - Task 설명

Input: (384, 512) 이미지

Output: 18개의 class 중 하나

* 마스크 착용여부, 성별, 나이에 따라 18개의 class로 구분된다.

< Class Description >

마스크 착용여부, 성별, 나이를 기준으로 총 18개의 클래스가 있습니다.

Class 1	Mask	Gender	Age
0	Wear	Male	< 30
1	Wear	Male	>= 30 and < 60
2	Wear	Male	>= 60
3	Wear	Female	< 30
4	Wear	Female	>= 30 and < 60
5	Wear	Female	>= 60
6	Incorrect	Male	< 30
7	Incorrect	Male	>= 30 and < 60
8	Incorrect	Male	>= 60
9	Incorrect	Female	< 30
10	Incorrect	Female	>= 30 and < 60
11	Incorrect	Female	>= 60
12	Not Wear	Male	< 30
13	Not Wear	Male	>= 30 and < 60
14	Not Wear	Male	>= 60
15	Not Wear	Female	< 30
16	Not Wear	Female	>= 30 and < 60
17	Not Wear	Female	>= 60

1. Introduction - 일정

▶ 경연 기간

2021.08.22 ~ 2021.09.02

▶ 1주차(2021.08.22 ~ 2021.08.29)

조원 각자 end-to-end 코드 작성 및 여러 실험

▶ 2주차(2021.08.30 ~ 2021.09.02)

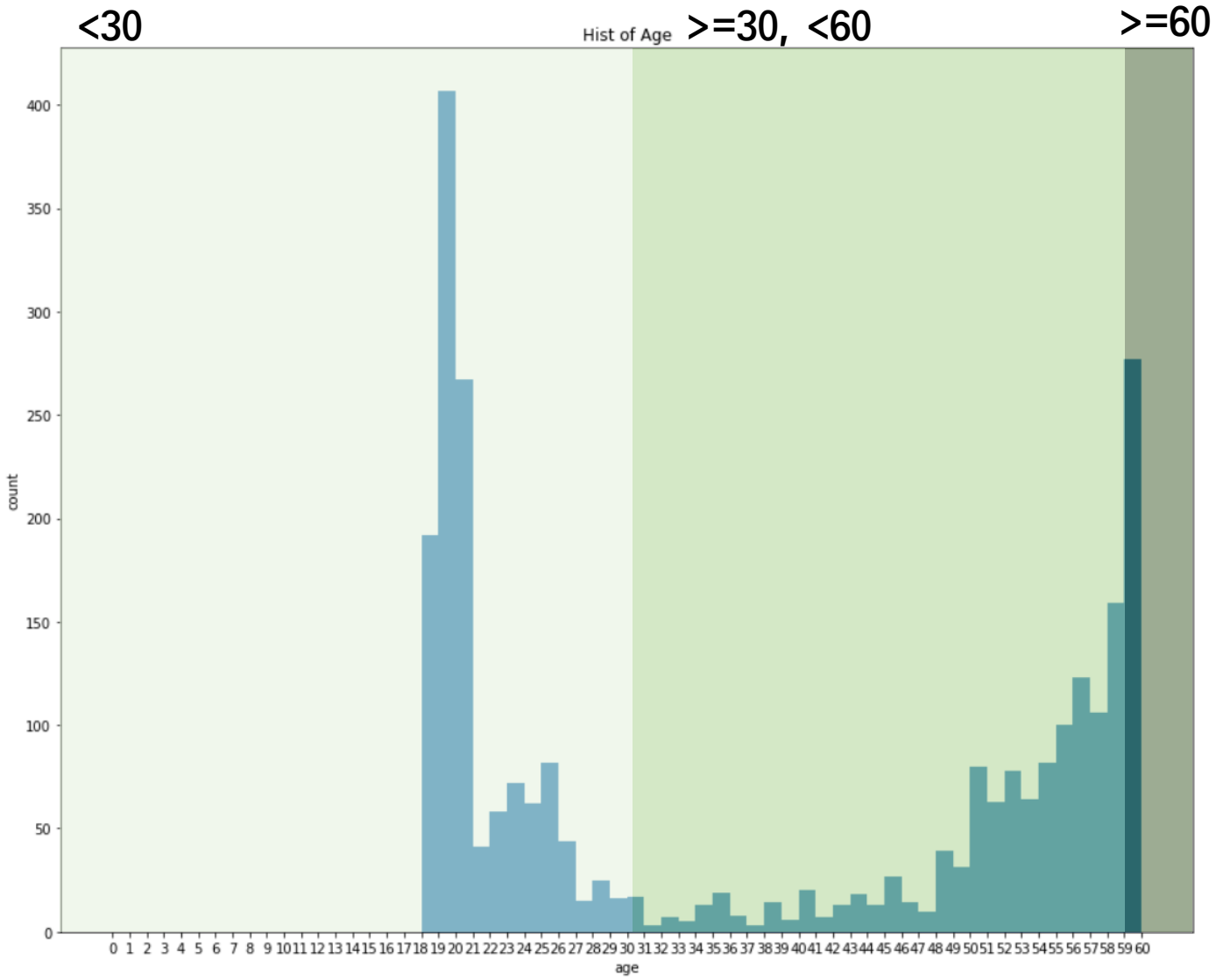
성능 향상을 위한 협업

2. Data Analysis - Age

※ Age에 대해 Inbalance 문제 존재

```
# 노년층의 데이터셋 개수가 다른 것의 약 1/6배
train_csv.age_class.value_counts()
```

```
<30             1281
>=30 and <60    1227
>=60             192
Name: age_class, dtype: int64
```



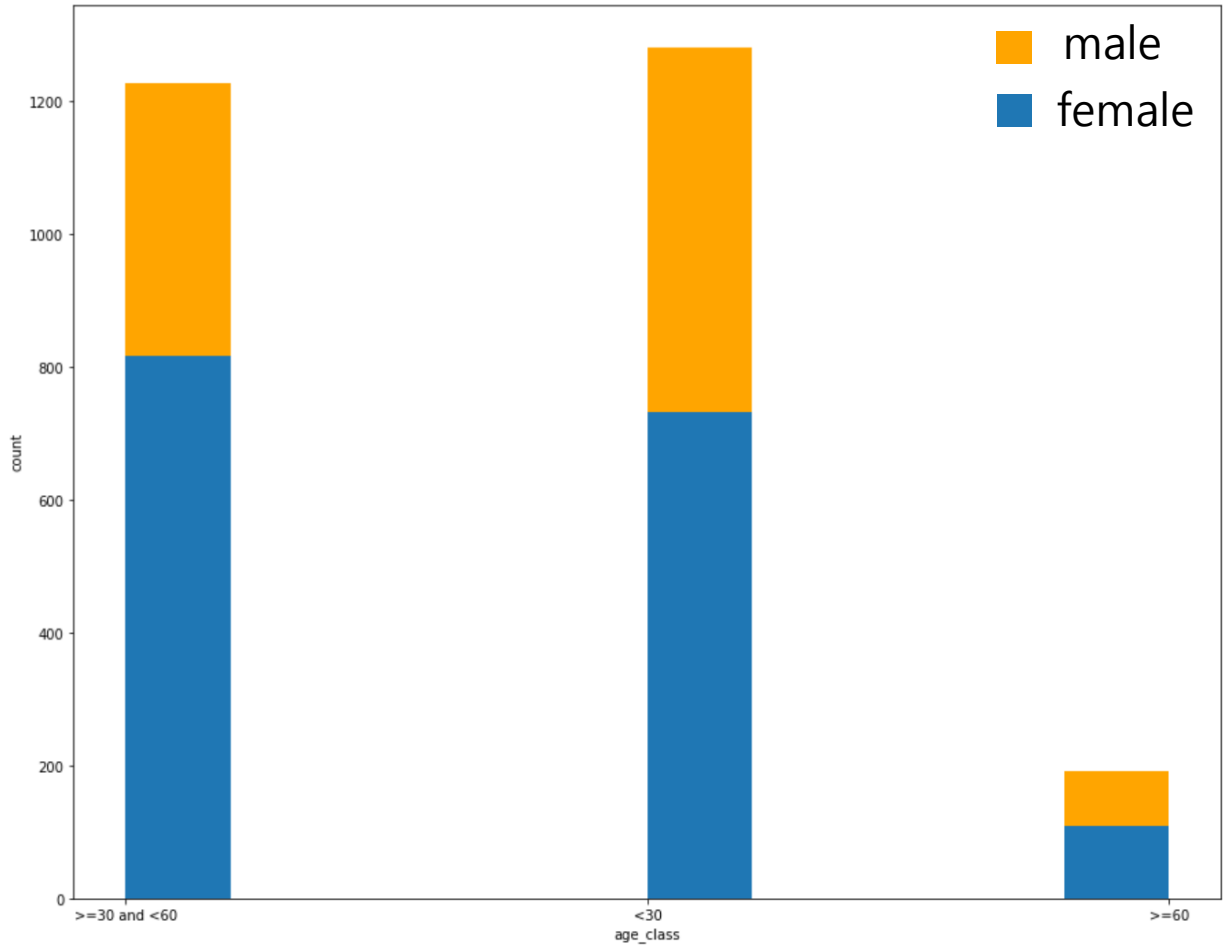
2. Data Analysis - Gender

▶ 성별 분포

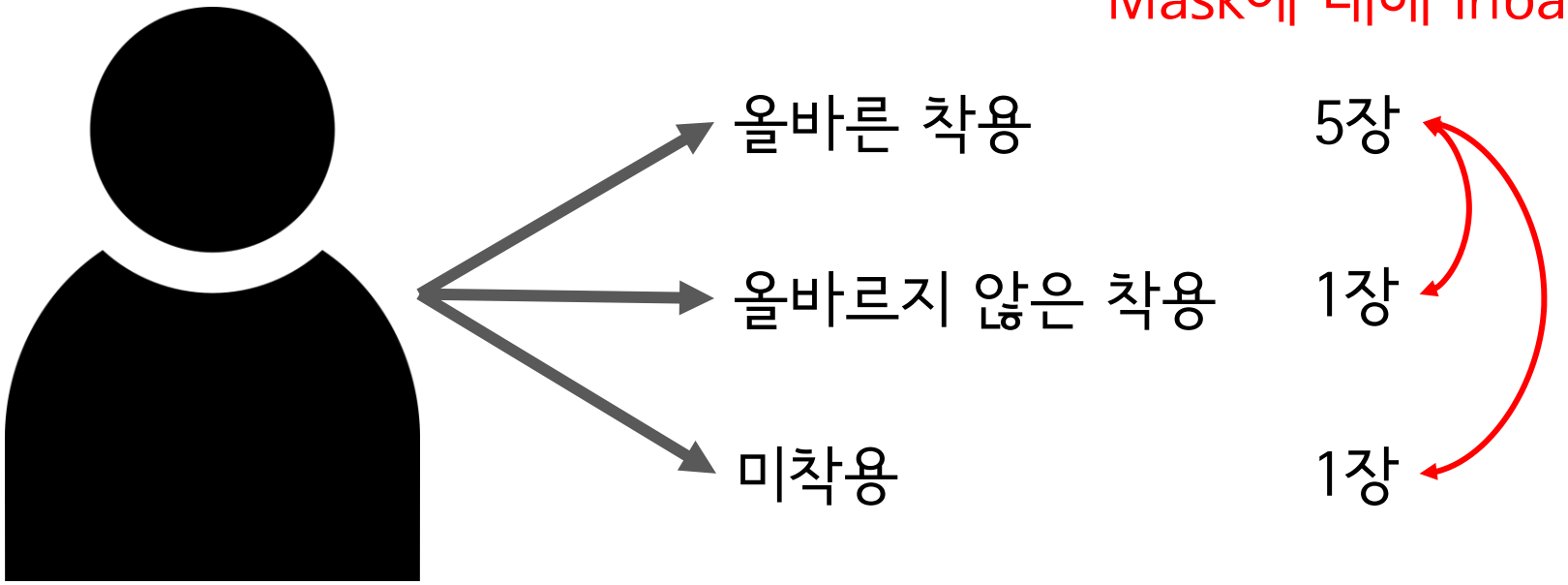
```
# female : male = 1658 : 1042 -> female이 male의 약 1.6배
train_csv.gender.value_counts()
```

```
female    1658
male      1042
Name: gender, dtype: int64
```

▶ 나이별 성별 분포



2. Data Analysis - Mask



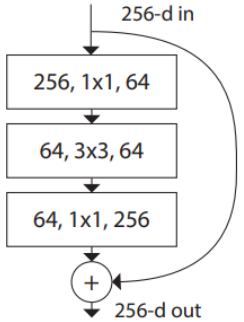
2. Data Analysis - Insight

- Age에 대한 inbalance 문제를 해결해야 한다.
- Mask에 대한 inbalance 문제를 해결해야 한다.
- 18개의 class로 분류하기보다는 마스크 착용/성별/나이 3가지 유형을 따로 다뤄본다.

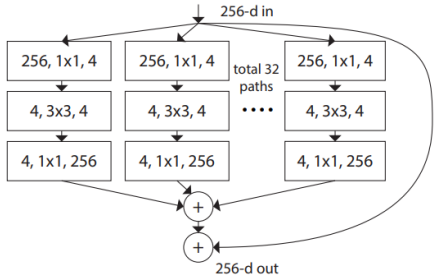
3. Baseline Code

- 주어진 data에 맞게 end-to-end 구현완성
- 18개 class에 대한 image classification을 수행함
- Pretrained model을 이용해 학습을 진행하였음, 성능이 제일 좋았던 EfficientNet-B4 선택

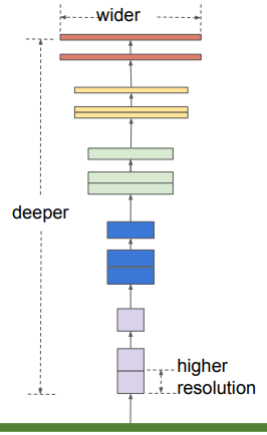
Resnet18, Resnet50



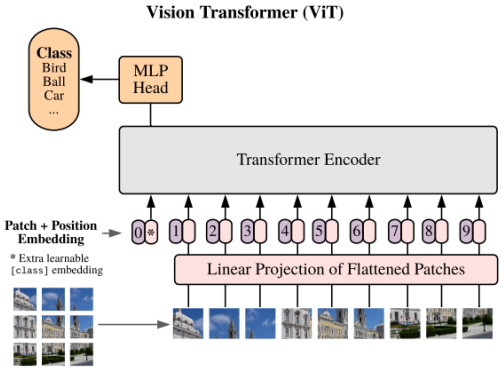
ResNext



EfficientNet-B4



Vision Transformer(ViT)



3. Baseline Code - Insight

- Baseline code 성능을 확인해본 결과 왼쪽과 같은 문제점이 있었고, 이를 위해 오른쪽과 같은 해결방안을 떠올렸다.

1
올바른 마스크 착용 vs 올바르지 않은 마스크 착용의 구분을 완벽하게는 하지 못한다. (Train 성능)

2
성별을 완벽하게는 구분하지 못한다. (Train 성능)

3
Inference 결과 60세 이상의 class 추측이 드물다.
즉 나이에 대한 inbalance 문제를 해결해야 한다.
(Validation 성능)

1 2
18개의 class 학습이 아닌, 마스크/성별/나이 3가지 유형을 나눠 학습을 진행해본다.(multi-head model)

올바른 마스크 착용 데이터를 under sampling 해본다.

데이터 불균형을 해결해줄 수 있는 loss를 적용해본다.

3
Augmentation 단계에서 나이에 대해 mix-up을 시도해본다.

나이에 대해 label smoothing을 적용해본다.

60세 이상의 데이터가 별로 없으니,
Stratified k-fold를 적용해본다.

Labeling을 새로 해본다.(60세 이상 -> 58세이상)

4. Improvement

앞서 생각한 아이디어들을 실제로 구현해보았다.
성능 올리기에 실패한 아이디어는 그 원인을 살펴보고, 성능을 올렸다면 그 구현 방법에 대해 정리해볼 것이다.

- 18개의 class 학습이 아닌, 마스크/성별/나이 3가지 유형을 나눠 학습을 진행한다.

- 모델의 forward 결과로 3개의 tensor을 return하게 한다.
- Loss는 3가지의 loss(mask, gender, age)의 합으로 계산한다.

```python
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()
        self.model = timm.create_model('tf_efficientnet_b4', pretrained=True)
        self.model.classifier = nn.Linear(1792, 1024)
        self.fc1 = nn.Linear(1024, 3)
        self.fc2 = nn.Linear(1024, 2)
        self.fc3 = nn.Linear(1024, 3)

    def forward(self, x):
        fc_output = self.model(x)
        mask = self.fc1(fc_output)
        gender = self.fc2(fc_output)
        age = self.fc3(fc_output)

        return mask, gender, age
```

```python
loss_masks = criterion(outputs_mask, masks)
loss_genders = criterion(outputs_gender, genders)
loss_ages = criterion(outputs_age, ages)

loss = loss_masks + loss_genders + loss_ages
```

4. Improvement

- 올바른 마스크 착용 데이터를 under sampling 해본다. ⟶ validation 성능이 크게 낮아졌다.

[원인분석]

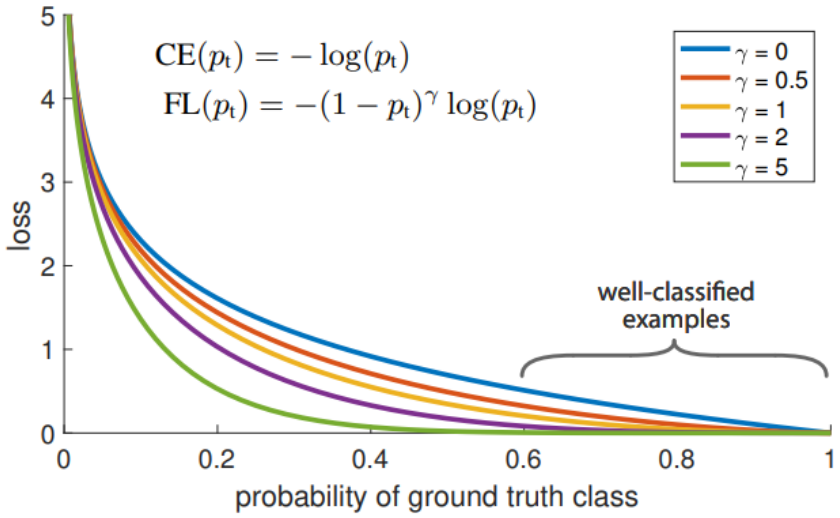
이미 앞선 개선으로 마스크에 대한 validation 정확도가 높아진 상태였다.
데이터 개수만 봤을 때는 마스크 데이터셋이 inbalance 한 것처럼 보이지만(correct:incorrect:not wear=5:1:1)
실제로는 각 클래스에 대한 학습이 잘 이뤄진 상태이므로
굳이 이러한 inbalance 를 해결하기 위해 데이터 개수를 줄일 필요는 없었다.

오히려 데이터 수를 줄임으로써 다른 클래스의 학습(성별, 나이)에 좋지 않은 영향을 끼쳤다.

4. Improvement

- 데이터 불균형을 해결해줄 수 있는 loss를 적용해본다.

  - Focal loss 적용




```python
class FocalLoss(nn.Module):
    def __init__(self, weight=None,
                 gamma=2., reduction='mean'):
        nn.Module.__init__(self)
        self.weight = weight
        self.gamma = gamma
        self.reduction = reduction

    def forward(self, input_tensor, target_tensor):
        log_prob = F.log_softmax(input_tensor, dim=-1)
        prob = torch.exp(log_prob)
        return F.nll_loss(
            ((1 - prob) ** self.gamma) * log_prob,
            target_tensor,
            weight=self.weight,
            reduction=self.reduction
        )
```

4. Improvement

- Augmentation 단계에서 나이에 대해 mix-up을 시도해본다.
- 나이에 대해 label smoothing을 적용해본다.

→ Validation 성능이 거의 변하지 않았다.

- 애초에 60세 이상의 데이터 개수가 매우 작아 이 둘로 성능을 올리기에는 한계가 있었다는 결론을 내렸다.

Mix-up

```python
def mixup_data(x, y1, y2, y3, alpha=1.0, use_cuda=True):
    '''Returns mixed inputs, pairs of targets, and lambda'''
    if alpha > 0:
        lam = np.random.beta(alpha, alpha)
    else:
        lam = 1

    batch_size = x.size()[0]
    if use_cuda:
        index = torch.randperm(batch_size).cuda()
    else:
        index = torch.randperm(batch_size)

    mixed_x = lam * x + (1 - lam) * x[index, :]
    y1_a, y1_b = y1, y1[index]
    y2_a, y2_b = y2, y2[index]
    y3_a, y3_b = y3, y3[index]

    return mixed_x, y1_a, y1_b, y2_a, y2_b, y3_a, y3_b, lam # (mixed_x, y_b) => 새로운 set [1-


def mixup_criterion(criterion, pred1, pred2, pred3, y1_a, y1_b, y2_a, y2_b, y3_a, y3_b, lam):
    loss = 0.0
    loss += lam * criterion(pred1, y1_a) + (1 - lam) * criterion(pred1, y1_b)
    loss += lam * criterion(pred2, y2_a) + (1 - lam) * criterion(pred2, y2_b)
    loss += lam * criterion(pred3, y3_a) + (1 - lam) * criterion(pred3, y3_b)

    return loss
```

Label smoothing

```python
class LabelSmoothingLoss(nn.Module):
    def __init__(self, classes=3, smoothing=0.2, 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))
```

4. Improvement

- 60세 이상의 데이터가 별로 없으니, Stratified k-fold를 적용해본다.

```
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=None)
k_idx = 1
for train_index, validate_index in stratified_kfold.split(np.zeros(len(train_ages)), train_ages):
    print(f'## Stratified_K-Fold :: {k_idx}')
    k_idx += 1
```

4. Improvement

- 나이에 대해 Labeling을 새로 해본다.

  - 기존에는 30세 미만 / 30세 이상 60세 미만 / 60세 이상으로 클래스가 나뉘었다.
    하지만 60세이상의 클래스는 61세 이상 데이터가 존재하지 않았다.
    또한 데이터 분석 챕터에서 보듯 초반, 중반대의 데이터 개수의 1/6정도였다.

    다른 class와는 달리 나이에 따른 외형은 완벽한 구분이 어렵다.
    따라서 나이 labeling 자체를 재정의 하는 것이 좋겠다는 판단 하에
    60세 이상을 58세 이상으로 변경하여 학습을 진행하였다.
    그 결과 f1 score가 0.05 향상되었다.

```
if 30 <= age < 58:
    age_label += 1
elif 58 <= age:
    age_label += 2
```

5. Conclusion – 최종 모델

New Age Labeling

Baseline: EfficientNet-B4

Optimizer: AdamP

Loss: Focal Loss

Multi-head Model

Stratified 5-fold

Train Augmentation

```
CenterCrop(400,200),
RandomBrightnessContrast(
  brightness_limit=(-0.1, 0.1),
  contrast_limit=(-0.1, 0.1), p=0.5
),
Normalize(
  mean=(0.548, 0.504, 0.479),
  std=(0.237, 0.247, 0.246), p=1.0
),
ToTensorV2(p=1.0)
```

5. Conclusion – 아쉬웠던 점

- 추가적인 외부 데이터를 활용하지 못했던 점
- 모델 성능 비교를 위한 Wandb, TensorBoard를 활용하지 못했던 점
- 결과물별 성능을 체계적으로 정리하지 못했던 점
- ipynb 파일이 아닌 py파일로 실험을 진행하지 않았던 점
- 각 개념에 대한 깊은 이해를 하지 못하고 썼던 점
- Github을 이용하여 팀원간 소스공유를 하지 않았던 점

5. Conclusion – 느낀점

- Validation Set과 Test Set과의 일치성
- 결과물별 성능을 체계적으로 정리해야 함
- 문제 파악과 이를 해결하기 위한 insight 떠올리기
- insight를 실현시키기 위한 코딩 능력
- insight를 떠올리기 위해서는 기본적으로 탄탄한 기초지식이 갖춰져야 함
- 팀원간의 idea 및 진행상황 공유