

# [Mask classification Competition]

1	CV_9조		0.8007	82.8730
---	-------	---	--------	---------

CV / 9조

부캠 끝난 나      부캠 전의 나

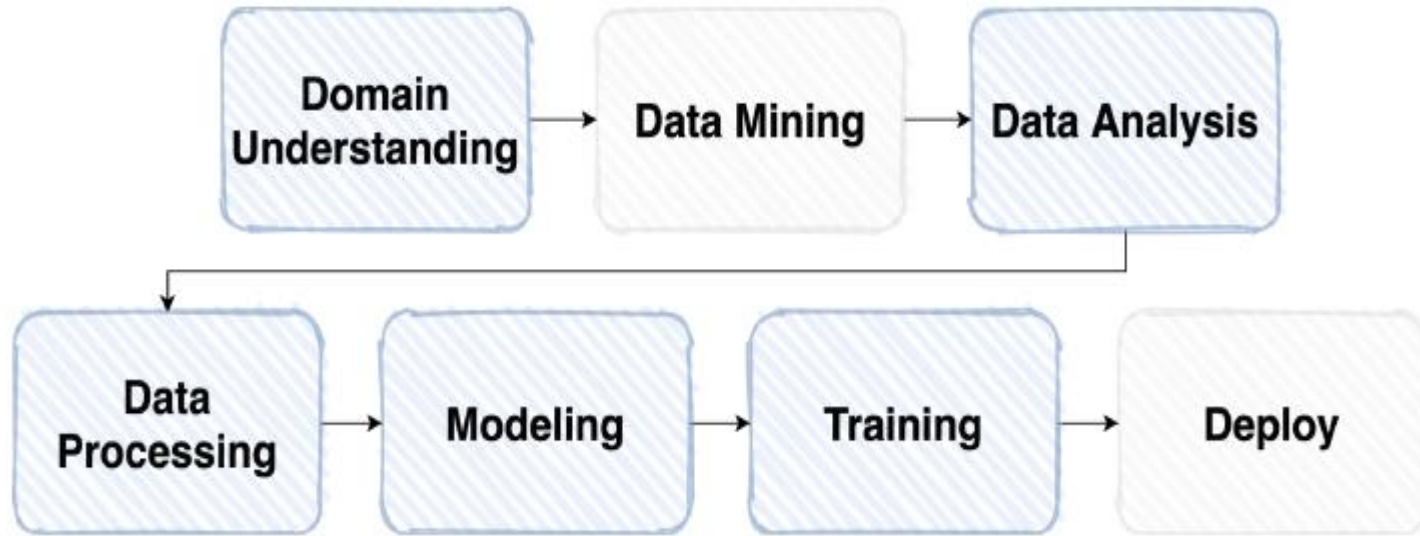


\* 비전? 믿고 맡겨줘!  
\* (각종 조직의 에이스)



\* 비전 그게 뭔데.....  
\* MNIST는 해봤는데.....

# 목차



- [1] 각 단계에서 관찰한 Insight와 Problem을 설명함
- [2] 그에 대한 Approach를 설명하고자 함

차근차근 Step by step으로 이야기해보겠습니다!

# Problem & Solution

Problem	Stage	Approach	Comment
Model architecture	Domain Understanding	[1] Single-way [2] Three-way parallel [3] Three-way head	- 18개의 label을 어떻게 나누어 예측할 것인지에 대한 issue
Data imbalance	EDA(Data Analysis)	[1] Sampler [2] Intra-class Cutmix [3] Data Augmentation	[1] Train/valid의 Data split에 관한 issue [2] CutMix 방법론 [3] 우리의 Data에 적합한 augmentation은?
Face detection (Image Noise)	EDA(Data Analysis)	[1] MTCNN	- TTA 시에 큰 성능 향상을 가져다주었음
Data mislabel	Data Preprocessing	[1] Hands-on correction	-되려 성능이 하락되었음 -직접 Annotate해보는 경험이 인상깊었음
Model selection & Overfitting	Modeling	[1] EfficientB4 [2] Cutmix	- Big model과 Small model 중 무엇을 택할 것인가? - 오버피팅을 막아줄, 데이터셋에 적합한 augmentation
Model-tuning	Training	[1] Val-training [2] Ensemble	- 성능 끌어올리기

# Step 1. Problem definition

## Class Description:

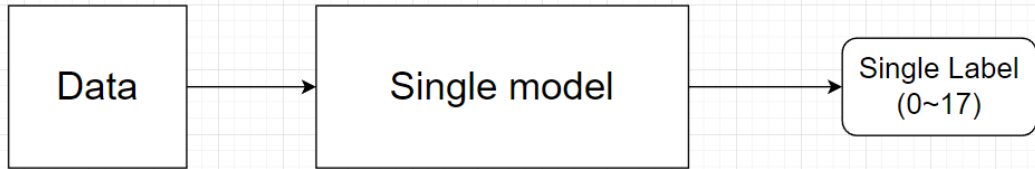
마스크 착용여부, 성별, 나이를 기준으로 총 18개의 클래스가 있습니다.

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

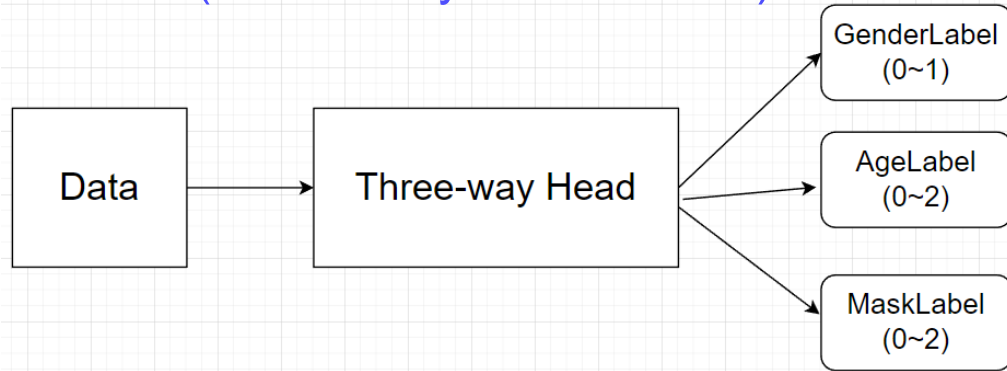
이미지와 메타데이터가 주어졌을 때,  
Mask / Gender / Age의 label을 예측하는,  
multi-label classification

# Comparison

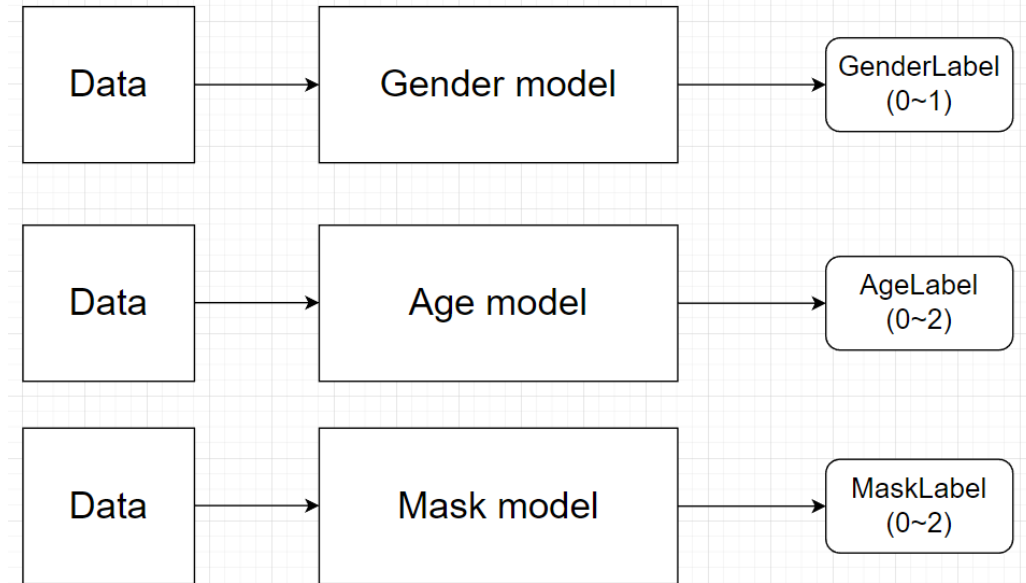
(Single model)



(Three-way head model)



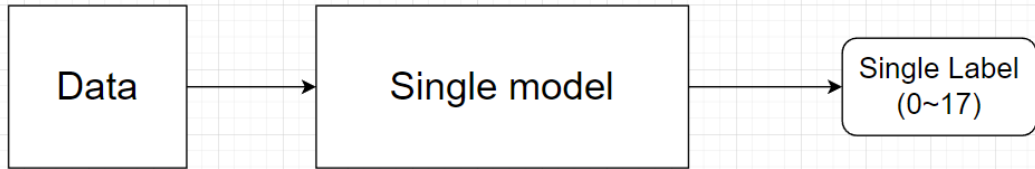
(Three-way model)



# Comparison : Feature correlated?



(Single model)

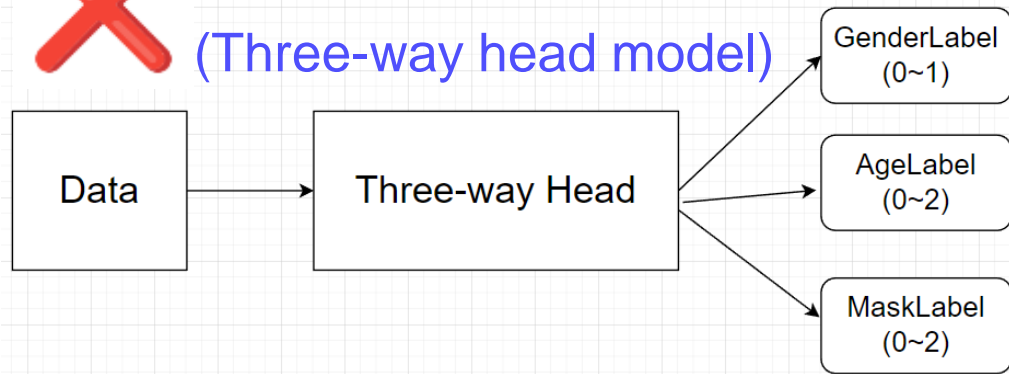


[Single model]

- 구현과 실험관리가 제일 편하다.



(Three-way head model)



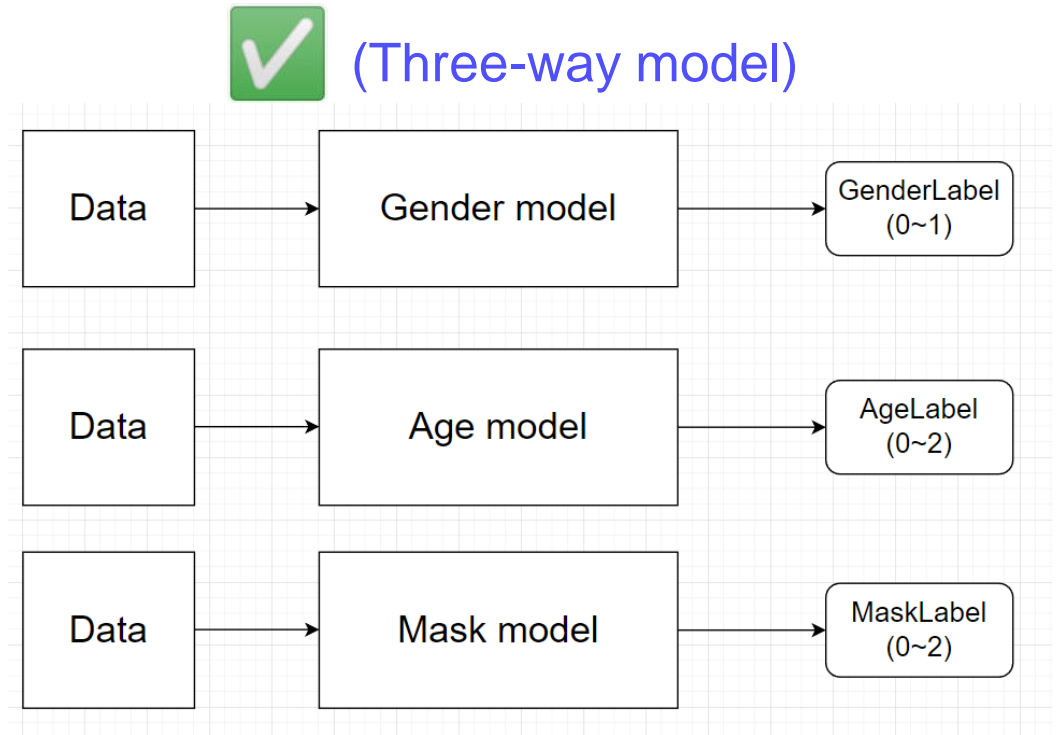
[Three-way head model]

- Correlation을 capture하는 것이 single과 다를 게 없다 생각하였으며, 실험과 구현의 복잡함(Sampler issue)로 인해 기각하였음

# Comparison : Feature Independent?

## [Three-way Model]

- 3개의 model이 전부 옳게 예측해야 실제 정답과 맞음
- (후일담) Age prediction이 이 접근의 bottleneck이었고, 나머지 것들은 90 후반의 정확도였으나 age 예측이 정확도가 많이 낮았다.  
단일 age model을 개선시킬 방법을 찾지 못해 기각.



# Closing Problem definition...

- [1] 총 3개의 접근 중 [Single-model](#)과 [Three-way model](#)의 baseline을 구축하여 진행했다.
- [2] 단일 Age-model의 개선책을 찾지 못해, 실험관리의 용이성과 시간 단축을 위해 [single-model](#)로 방향을 전환했다.



## Step 2. Data Analysis

이미지 분포

- Some insights, and mask

데이터 분포

- Imbalance
- 경계값

데이터 품질

- Image noise
- Label noise

# 이미지 분포 - Some insights

## [1] 마스크를 잘못 쓴 형태가 다양하게 분포해 있었다.

- 코가 보이는 경우와 턱만 보이는 경우는 Incorrect와 normal 간의 분류가 어려울 것 같았다.

## [2] 마스크의 종류도 다양했다.

- 코와 입의 윤곽이 어느 정도 드러나 있는 비말 마스크부터,
- 그보다 얼굴을 더 완벽하게 가리는 KF마스크와, 목까지 가리는 천 마스크도 있다.

### 1. class distinction

#### a. incorrect mask

올바르지 않은 방식으로 mask를 착용한 image를 분석했다.

- 입만 가린 image
- 눈과 코를 가린 image
- 코가 보인 image
- 코만 가리고, 입은 드러난 image
- 아랫입술만 보이는 image
- 턱만 가린 image
- 입과 코를 모두 가렸으나 턱이 보이는 image



코가 보이는 경우와 턱만 보이는 경우는 각각 correct mask, no mask와의 분류에 문제가 있을 수 있다고 분석했다.



# 이미지 분포 - Some insights

[3] 본 데이터셋은 대부분의 마스크가 색상이 있는 마스크였다.

- Mask 또한 불균형 데이터였지만, 마스크가 대부분 유색 마스크였기에 얼굴과 마스크 간의 Color 차이 정보를 모델이 학습하여 잘 분별할 수 있을 것으로 생각하였다.
- 실제로, 학습된 모델은 mask에서 오류율이 제일 적으며, Color 관련 Augmentation을 하면 LB score(리더보드 점수)가 줄어들었다.

```
In [72]: image.open("image_example")
```

```
Out [72]:
```



```
In [71]: show_label("image_example")
```

```
Out [71]: 16
```

**16은 Not-wear로 분류된다.**

- 물론, 모든 투명마스크를 분류 못하는건 아니다.
- 흰색이 일부 섞인 투명마스크의 경우 잘 분류하는 모습을 보였다.

# 이미지 분포 - Some insights

[4] 하지만... 투명 마스크를 착용한 사람은 잘 탐지를 못할 것 같다는 생각이 들었다.

- 인터넷의 투명 마스크 착용 데이터를 모델에 입력하면, No mask로 분류하는 경우도 종종 있었다..
- 추후 Model의 Robustness를 높이려면, 이런 데이터도 필요하겠다는 내용을 팀원과 토론할 수 있었다.

```
In [72]: image.open("image_example")
```

```
Out [72]:
```



```
In [71]: show_label("image_example")
```

```
Out [71]: 16
```

**16은 Not-wear로 분류된다.**

- 물론, 모든 투명마스크를 분류 못하는건 아니다.
- 흰색이 일부 섞인 투명마스크의 경우 잘 분류하는 모습을 보였다.

Now, problem and solution begins..

# Data Imbalance → Modeling, Training에서 address 예정

[1] Gender : 여성이 남성보다 2배 가량 많다.

[2] Mask : Incorrect와 No mask는 1개씩 있는데, 촬영 시 Normal은 5개씩 있어 1: 1: 5의 비율 차이가 있다.

[3] Age : 60대 이상의 데이터가 현저하게 적다.

---

Q. Imbalance 된 데이터가 어떻게 학습에 악영향을 줄 수 있을까?

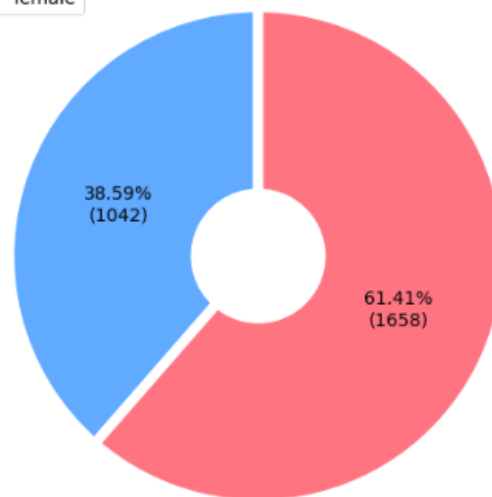
A1. Train/ Valid에 Data가 고르게 분배되지 못할 수 있다.

→ Train에서 학습하지 못한, 확률적으로 드문 label이 Valid에서 나타날 수 있다.

A2. Metric에 따라 Label을 biased된채로 학습할 수 있다.

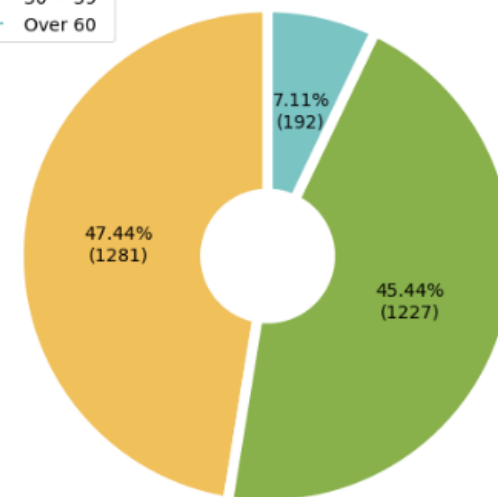
→ 드문 label의 정보를 제대로 학습 못할 수 있다.

— male  
— female



Gender

— Below 30  
— 30 ~ 59  
— Over 60



Age

female data의 양이 male data 보다 약 두배 가까이 많고, 60대 이상의 data 양이 적은 것을 확인했다.

# Data의 경계값 → 제거하거나, 다르게 넣어볼까?

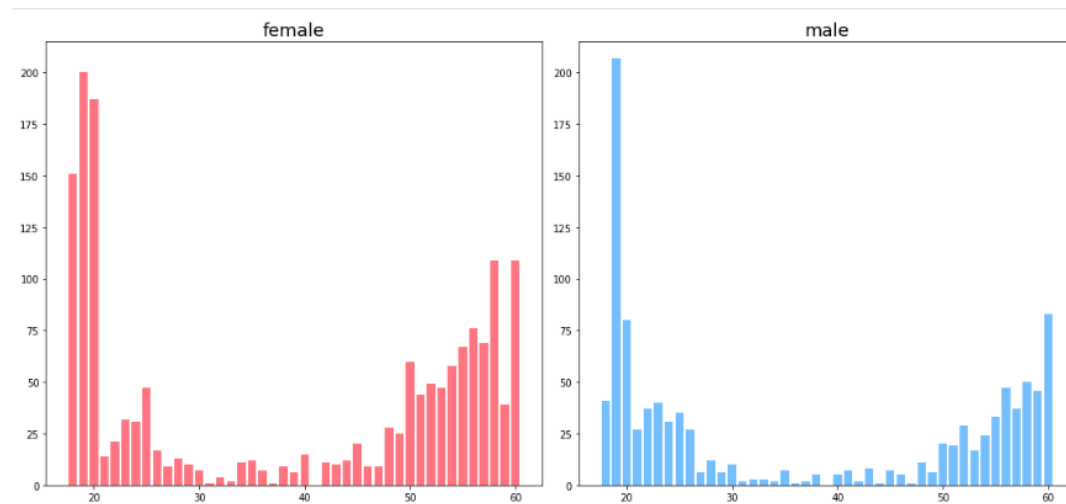
[1] Age의 상한이 60이다.

[2] 57~59살 중 일부 데이터는 육안으로 보기에, 60대 이상과 구분하기가 힘들었다.

[3] 따라서, 학습 시 57 ~ 59살을 제외하거나 / 특정 나이부터는 60대 이상으로 동일하게 취급하자는 이야기가 나왔다.

따라서, 학습의 용이성을 위해 경계값의 데이터를 제거하는 옵션을 `Age_removal`이라 하여

`Age_removal` 옵션이 True라면 27~ 29, 57 ~59의 데이터는 학습에서 제외하고자 하였다.



성별 별로 나이 분포를 확인해본 결과 남녀 모두 20대 이하, 50세 이상의 구간에 data가 편중되어 있었다.

# 데이터 품질 - Image noise

[1] 배경에 글자가 있거나, 다른 인물 사진이 있는 경우도 있었다.

[2] 사진에서 얼굴이 차지하는 비중이 다르다.

- 우측 상단의 사진처럼 크게 차지하는 얼굴도 있는가 하면
- 좌측, 우측 하단의 사진처럼 작거나 중간 사이즈의 얼굴도 있다.

따라서, Augmentation을 할 때  
일반적인 Crop을 하면 모든 얼굴을 다 보지 못하거나  
배경을 완벽히 제거하진 못하겠다는 생각이 들었다.





# 데이터 품질 - Label noise : 사람도 어렵다

Incorrect인데 normal로 분류되어있음

Male? Female?(GT는 Female)

50대 후반? 60대 이상?(GT는 50후반)

헛갈리는 사진이 많았으며,  
잘못 label된 사진이 많았다..정도로 요약하겠습니다.

# Quality problem: 다음 Preprocessing에서 다룰 것

## Data Noise



Face detection - MTCNN



Scene Removal

## Label Noise



외부 API를 이용

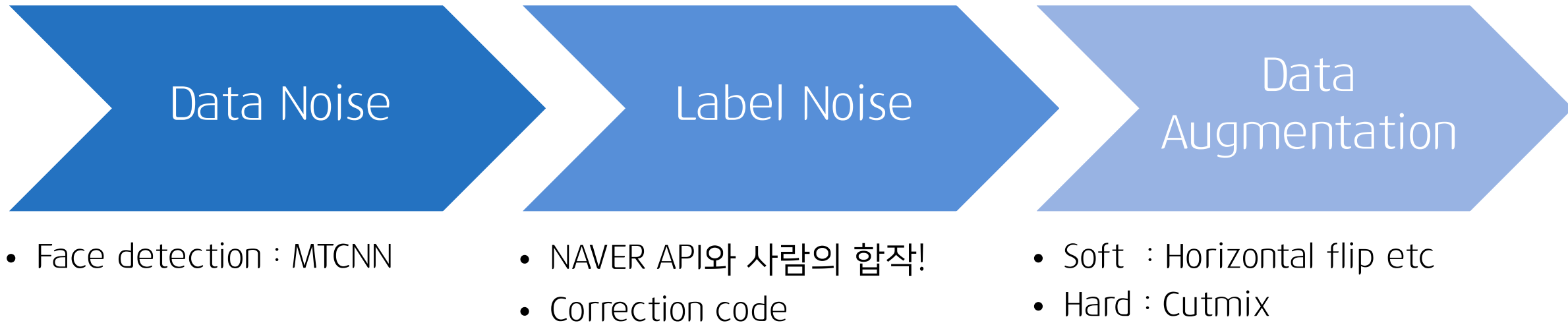


한땀한땀 Annotation



Pseudo-labelling

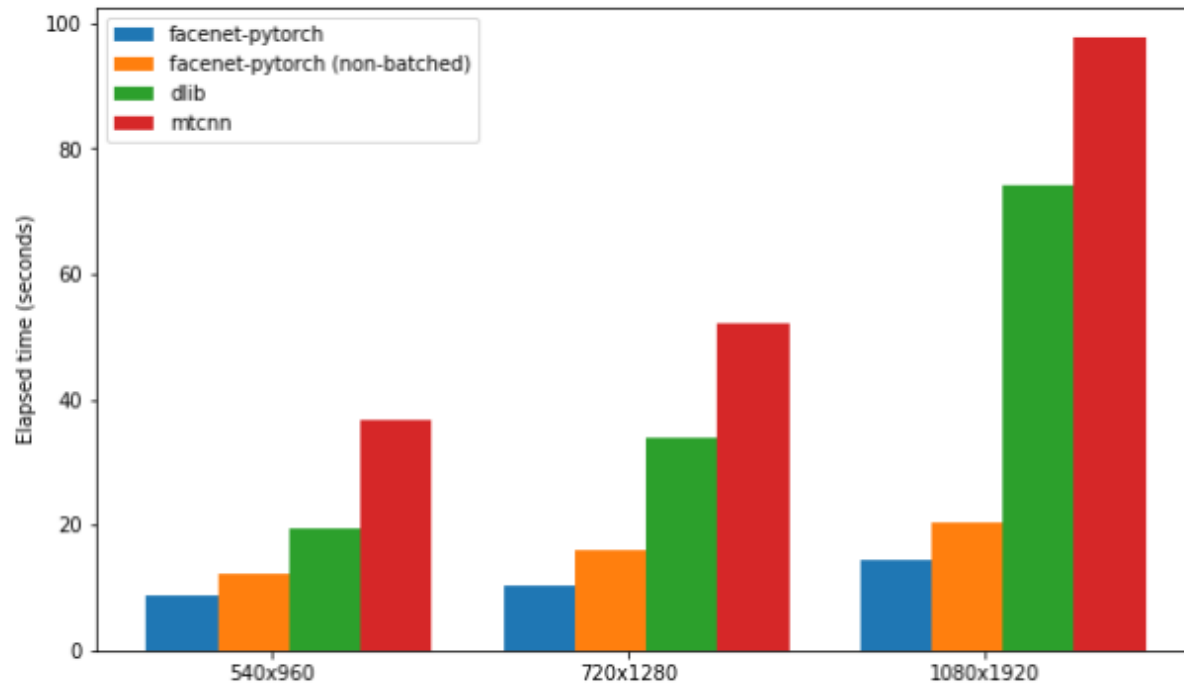
# Step 3. Data Preprocessing



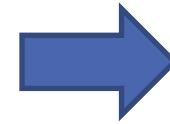
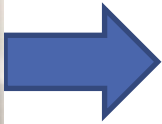
# Face detection : MTCNN

Problem : Dynamic face size, Background noise

- 다른 face-detection 라이브러리 대비 시간이 적게 들고,
- 성능이 준수하다고 하여 이용하였음



# Demo : 제법 괜찮았다.



# MTCNN - Hindsight

[1] CenterCrop보단 훨씬 느리다.

```
▶ %time
  out = facenet(img)
[13] ✓ 0.2s

... CPU times: user 800 ms, sys: 8 ms, total: 808 ms
  Wall time: 136 ms

  %time
  out = prob_facenet(img)
[14] ✓ 0.1s

... 0.9997948
  CPU times: user 772 ms, sys: 32 ms, total: 804 ms
  Wall time: 121 ms

  %time
  out_crop = crop(img)
[15] ✓ 0.4s

... CPU times: user 72 ms, sys: 0 ns, total: 72 ms
  Wall time: 11.3 ms
```

[2] 예상과 다르게 나온 Face들: 학습이 제대로 진행 안되었을 것

22		3	0	Age_GT:0, Age_pred:0	gender_GT:1, gender_pred:0	Mask_GT:0, Mask_pred:0
17		3	0	Age_GT:0, Age_pred:0	gender_GT:1, gender_pred:0	Mask_GT:0, Mask_pred:0

- 눈, 코, 입 자체를 가린 마스크 이미지같은 noise data의 경우에는 MTCNN도 잘 작동하지 못했다.
- 하여, 최종 ensemble에서는 CenterCrop을 이용한 모델도 이용하여 noise를 줄이고자 했다.

<https://www.kaggle.com/code/timesler/comparison-of-face-detection-packages/notebook>

# Label noise – API + Human learning



**NAVER**

Face Recognition API  
(Age, Gender)



- 나이와 성별을 알려주는 네이버 API를 이용하여 mislabel을 추려냄
- 그중에 Gender 580건에 대하여, 이를 교정해주는 GUI 프로그램을 만듦
- 팀원들과 판단하에 이를 고쳐나감

# Label noise - Hindsight



- [1] Re-labeling 시킨 데이터의 비율이 낮아서인지, 적용 전후 **성능 차이가 크게 없었음**
- [2] **Data-Centric AI의 중요성**에 대해 느낄 수 있었다. 데이터가 잘못되었다면 사람이 직접 Annotate를 해야하는 경우가 있겠다고 느꼈다.
- [3] Annotation Tool을 손으로 직접 짜서 비효율적이었는데, 추후 필요하면 **더 좋은 annotate 툴**, 또는 Data fix tool을 찾아보고 싶다 느꼈다.



# Data Augmentation

[1] 18900 장의 데이터는 정말 적은 편

[2] 추후 모델링 부분에서 쓰겠지만, 어떤 모델을 가져다 쓰든 1 epoch 만에 Train이 90%에 가깝게 올라감

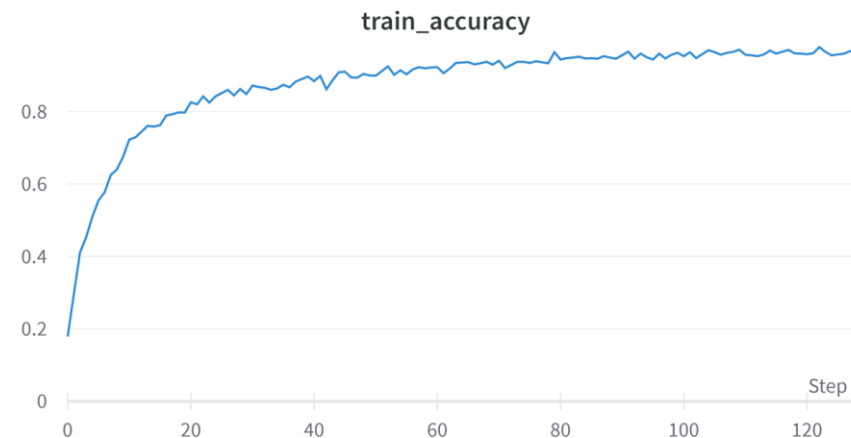
→ [오버피팅에 취약](#)

[3] **Data Augmentation이 없다면**, epochs이 반복될 때마다 모델이 같은 데이터를 여러 번 보게 되는 셈

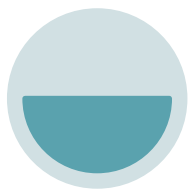
→ 데이터를 사실상 암기하여 일반화 효과가 약해지며, 이는 특히 Few label(60대 이상)에 치명적

→ 모델의 Noise마저 학습할 수 있음.

[4] 우리의 Dataset의 속성에 걸맞는 Augmentation 기법을 골라야 한다고 생각하였다.



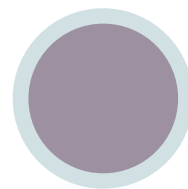
# Data Augmentation



## Soft Augmentation

이미지의 원본을  
손상시키지 않는 선에서  
noise를 적용

Ex) Flip, Rotate



## Hard Augmentation

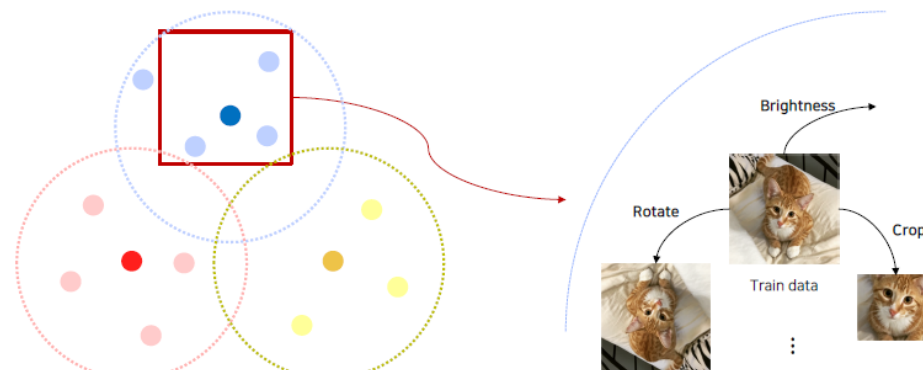
이미지의 원본에  
강하게 noise를 주는 기법

Ex) MixUp, CutMix

# Data Augmentation에게 우리가 기대하는 것

- [1] Augmentation을 통해 데이터의 일반화 성능을 확보하면서 + Overfitting을 방지하는 효과를 거두어야 한다.
- [2] 데이터의 개수가 전반적으로 적기 때문에, Oversampling과 비슷한 효과를 내는 Hard Augmentation을 이용하면 좋을 것 같다.
- [3] 데이터셋의 속성을 고려한 Augmentation을 하여, TTA때도 일반화될 수 있는 방법론을 찾자.

Examples of augmentations to make a dataset denser



And many other augmentations available!

-(출처) 오태현 마스터님 CV이론 2강, P.9

# Soft Augment 관련 실험

[1] ColorJitter의 경우 어떤 transforms와 조합해도 좋은 결과가 나왔다.  
이때, ColorJitter의 값을 0이 아닌 최소로 주는 것이 좋은 결과를 낸다.

[2] 너무 많은 transform이 추가되자 모델의 성능이 매우 낮아졌다.

→ 실험군이 다 밝기 관련 변환 + Geometric 변환이었는데,  
이런 변환 중 일부는 Data의 일반화 가능성을 떨어뜨리는  
augment 기법이었기 때문으로 추정된다.

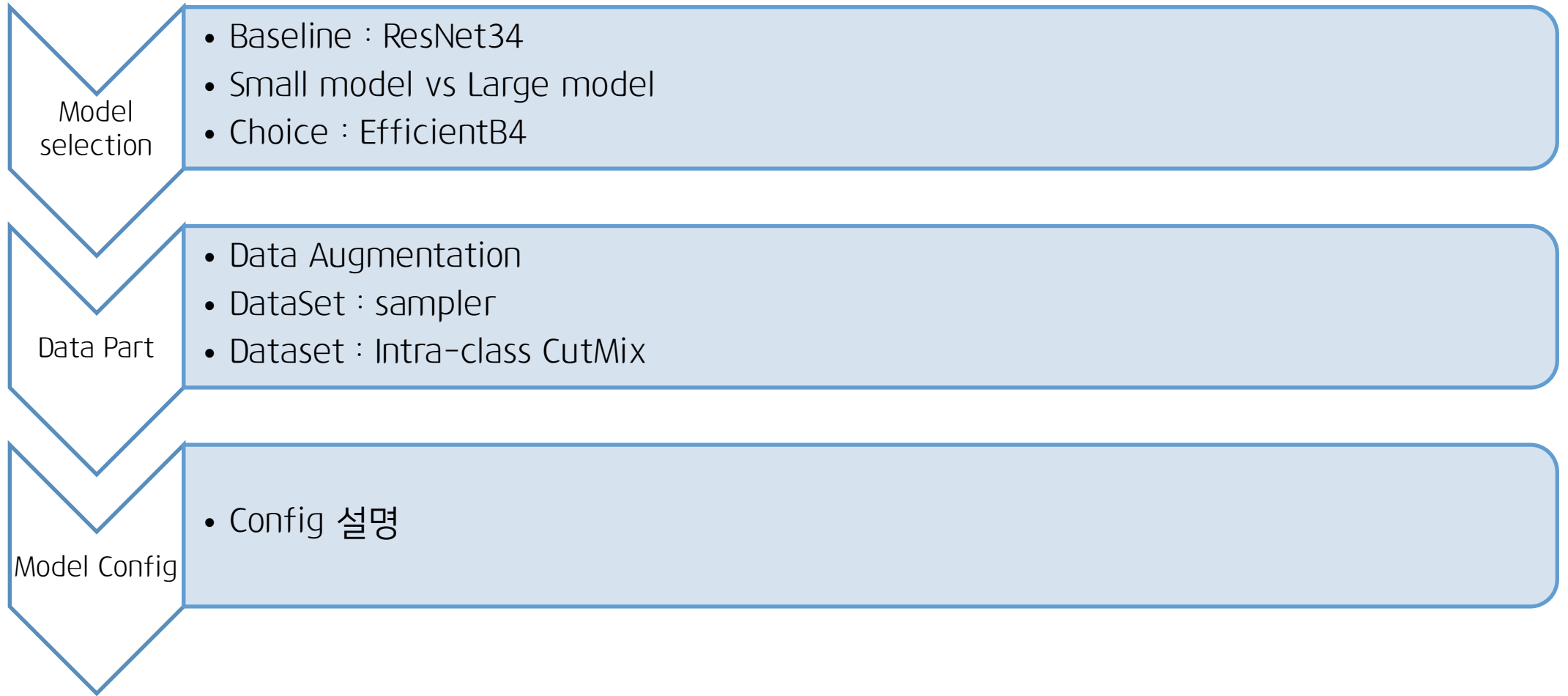


# 우리가 택한 Augmentation

종류	구현 여부	사용 여부	의의(우리에게 적합한지)
Horizontal Flip	0	0	적합하다. 카메라의 경우 좌우반전이 될 수 있고, 충분히 있을 법한 사진이라고 느꼈다.
CenterCrop(MTCNN)	0	0	필요하다. 이는 원활한 Inference를 위함이다.
ColorJitter	0	X	- 초기 실험에서는 성능을 높여줬지만, 대회 후반부에서는 사용하지 않기로 했다. - ColorJitter는 유색 마스크와 인물 얼굴 간의 밝기차 정보를 어느정도 왜곡시킬 수 있다고 생각하였기 때문이다.
RandomErasing	0	0	불필요한 배경정보를 삭제해주거나, 경우에 따라 얼굴 정보를 삭제해 robust한 학습이 가능할 것이라 생각했다.
Intra-class CutMix	0	0	(설명) getitem 시에, 같은 label 끼리 cutmix해서 data를 load (의의) 각 Data가 위치한 manifold를 조금 더 연속적으로 학습할 수 있을거라고 생각했다.
Age-transfer GAN	X	X	(설명) 젊은 사람들의 데이터에 StyleGAN을 적용시켜, 60대 이상의 데이터를 확보하고자 했음 (한계) 학습의 어려움 + 60살이 경계값인데 데이터를 오염시키는 행동일 것 같았다.
Scene Removal	X	X	(설명) 사람을 제외한 배경을 제거하여, 인물 사진만 남기는 방법 (한계) 시간 문제 상 구현하지 못하였으며, CenterCrop이 이를 해결해 줄 수 있을 것이라고 여겼다.

- Geometric 변환은, 좌우반전을 해주는 Horizontal Flip만이 적합하다고 생각했다.
- 그 외의 Affine transformation이나 color 관련 transformation은 데이터셋의 특성에 적합하지 않다고 결론지었다.  
(ex. 데이터셋 수집 특성상, 사진이 회전된 채로 있을것같진 않았다.)

# Step 4. Modeling



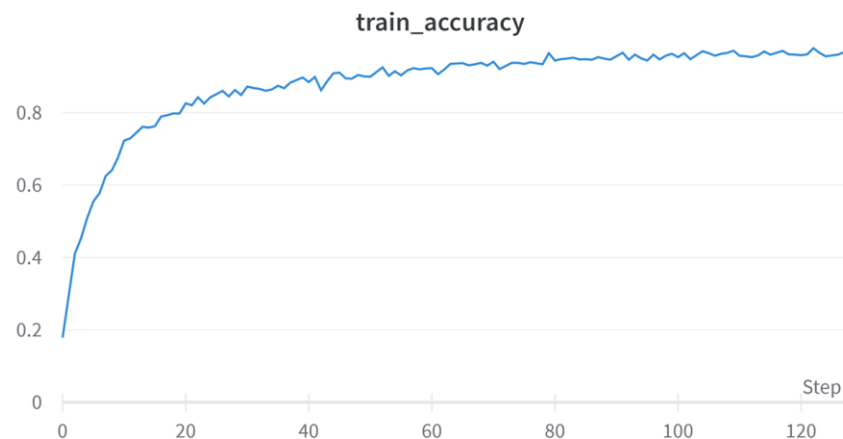
# Model selection – Baseline… Overfitting!!

(Baseline) ResNet18 & ResNet34

Baseline은 SOTA 모델보다는 , 단순한 모델부터 시작하는게 좋을 것 같아 ResNet18, ResNet34를 이용하였다.

➔ 학습의 용이성 때문.

그러나 ResNet34에서도 , Augmentation 없이 1 epoch만에 Train Acc이 90%를 넘어가 오버피팅이 생긴다고 생각했다.



# Model selection – Small model & Big model

## Small Model



☐ CNN 계열

☐ 학습이 빠른 편

☐ 파라미터 수가 적으면 오버피팅을 피할 수 있음

## Big Model



☐ ViT 계열

☐ 표현력 High, 요구되는 데이터 High

☐ Inductive bias로 인해 그만큼 데이터가 필요함

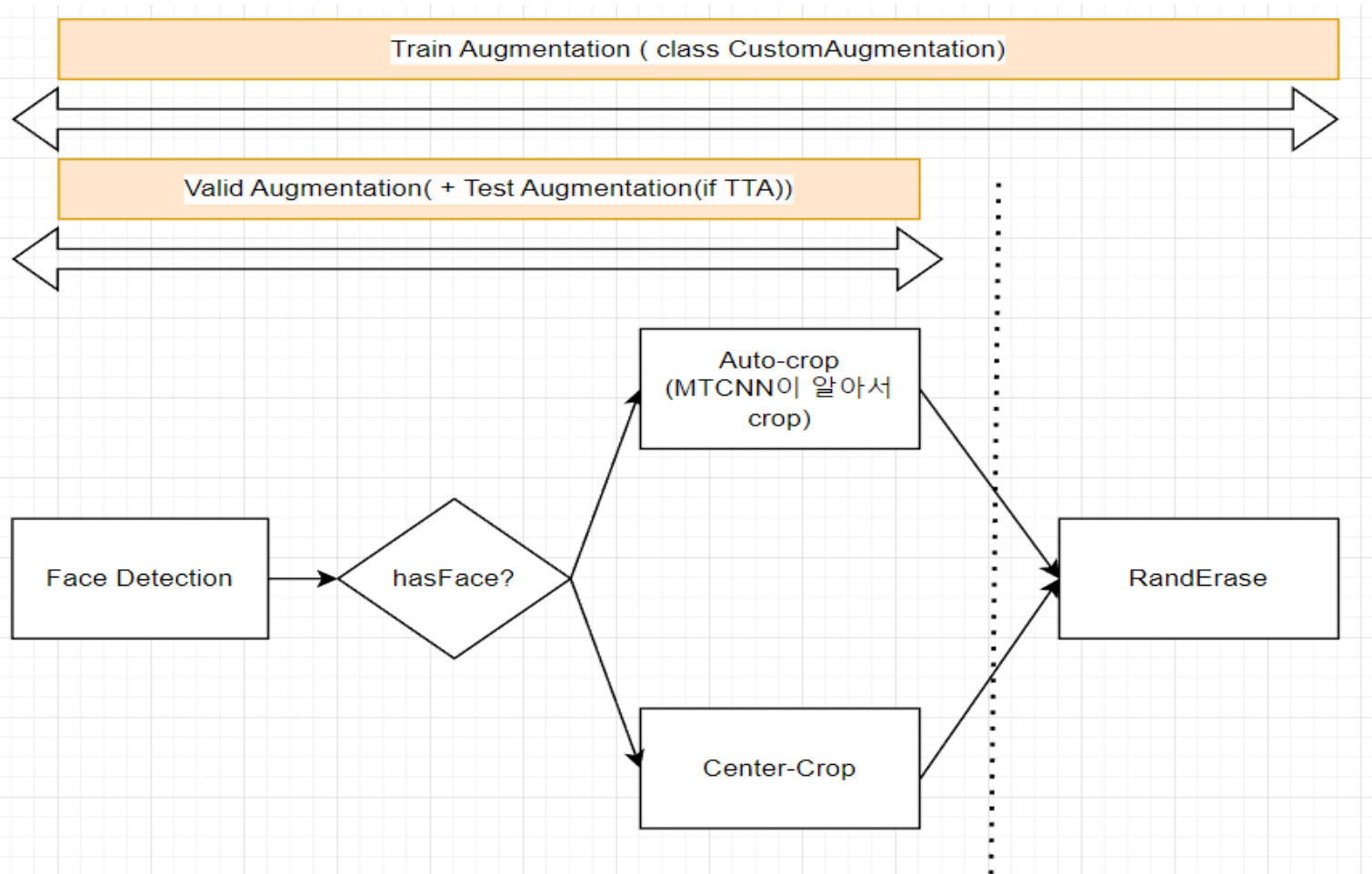


# Model selection

- [1] ResNet34로도 Train accuracy가 높다고 생각했기에, 모델이 주어진 데이터를 표현하기에는 표현력이 이미 충분하다고 생각했다.
- [2] 파라미터 수가 ResNet34보다는 적어야 overfitting을 방지할 수 있을 거라고 생각하여, ResNet34보다 파라미터 수가 적으면서 ImageNet 기준 Accuracy가 더 높은 모델을 고르기로 하였다.
- [3] Efficient\_v2\_s와 Efficient\_B4였고, torchvision의 버전 문제로 인해 Efficient\_B4를 택하였다.
- [4] ViT를 사용하려는 시도도 있었으나, 학습의 어려움과 시간문제로 포기하였다.

	Weight	Acc@1	Acc@5	Params	Numbers
	EfficientNet_V2_S_Weights.IMAGENET1K_V1	84.228	96.878	21.5M	2,150,000
✓	EfficientNet_B4_Weights.IMAGENET1K_V1	83.384	96.594	19.3M	1,930,000
	EfficientNet_B3_Weights.IMAGENET1K_V1	82.008	96.054	12.2M	1,220,000
	RegNet_Y_3_2GF_Weights.IMAGENET1K_V2	81.982	95.972	19.4M	1,940,000

# Data Part 1 : Augmentation



- 얼굴 탐지 성공하면 그대로, 실패 시 centercrop
- Train에는 추가로 RandErase 적용
- Test때 valid와 동일한 augmentation을 적용시킬 수 있다(TTA)

# Data Part 2: WeightedRandomSampler

## [1] Sampler??

- DataLoader에서 데이터를 가져올 때, 그 기준이 되는 idx를 sampler로 조절 가능

## [2] WeightedRandomSampler

- 가중치(확률)를 기반으로 주어진 data에서의 sample을 뽑아냄
- 가중치가 클수록 자주 뽑히고, 적을수록 적게 뽑힘
- 각 label 의 분포확률을 이용하면, 모든 label이 비슷한 비율로 뽑히게 조절할 수 있다.
- 많은 label은 더 적게, 적은 label은 더 많이 뽑히도록

CLASS torch.utils.data.WeightedRandomSampler(weights, num\_samples, replacement=True, generator=None) [SOURCE]

Samples elements from [0, ..., len(weights)-1] with given probabilities (weights).

### Parameters:

- **weights** (sequence) – a sequence of weights, not necessary summing up to one
- **num\_samples** (int) – number of samples to draw
- **replacement** (bool) – if True, samples are drawn with replacement. If not, they are drawn without replacement, which means that when a sample index is drawn for a row, it cannot be drawn again for that row.
- **generator** (Generator) – Generator used in sampling.

### Example

```
>>> list(WightedRandomSampler([0.1, 0.9, 0.4, 0.7, 3.0, 0.6], 5, replacement=True))
[4, 4, 1, 4, 5]
>>> list(WightedRandomSampler([0.9, 0.4, 0.05, 0.2, 0.3, 0.1], 5, replacement=False))
[0, 1, 4, 3, 2]
```

## Data Part 2: 진짜 고르게 뽑힐까? ➔ YES, but..

[1] 샘플링 결과 각 label이 700~800 개로 고르게 배분된 것을 알 수 있지만,

- Frequent label에서 뽑히지 않은 데이터가 있음
- Rare label에서 같은 데이터가 반복적으로 뽑혔음

이런 사실을 알 수 있다.

[2] Frequent label의 문제는 Epochs이 반복되면서 해결할 수 있겠지만, Rare label의 문제는 Data Augmentation이 필요하겠다고 생각했다.

각 Label이 700~800개로 고르게 배분된 것을 알 수 있음

- 달리 말하면, Frequent Label은 적게 뽑혔고, Rare label은 많이 뽑혔단 뜻
- Rare label은 epoch이 반복될수록 overfitting 가능성
- Frequent label은 epoch이 반복되면서, 그 전에 보지 못한 데이터를 볼 수 있다.
- 어느 형태든 Data의 다양성 측면에서 hard-augmentation이 필요하다고 생각했다.

```
#Loader 정의 train_sampler
train_loader = DataLoader(
    data,
    512, # batch size
    num_workers=0,
    shuffle=False,
    drop_last=True,
    sampler = data.get_sampler("train") # WeightedRandomSampler
)

label_count = list(range(18))

for data, label in tqdm(train_loader):
    data = data.to(device)
    label = label.to(device)

    label = label.cpu().numpy()
    for i in label:
        label_count[i] += 1

print(label_count)
```

[illegible]

[748, 759, 720, 740, 722, 781, 722, 757, 767, 700, 748, 747, 753, 754, 780, 736, 758, 773]

# Data Part 3: Intra-class CutMix (SMOTE)

## [1] SMOTE

- 낮은 비율로 존재하는 클래스의 데이터를 kNN 알고리즘을 활용하여 새롭게 생성

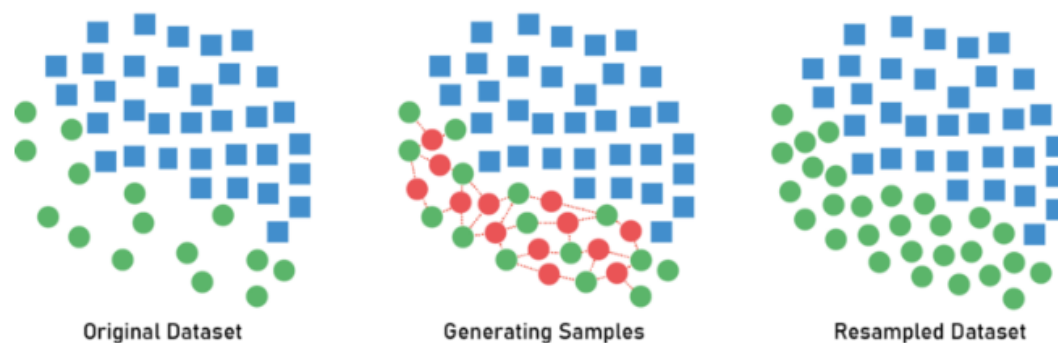
## [2] SMOTE를 이미지에서 구현하려면 어떻게 할 수 있을까?

- 동일한 Label의 이미지를 뽑아 MixUp 또는 CutMix 등을 하면
- 두 이미지 사이의 평균 데이터 포인트를 오버샘플링하는 것과 동일한 효과를 낸다고 생각했다.

## [3] Dataset에서 \_\_getitem\_\_을 통해 CutMix를 구현 가능하게끔 하였다.

## 1. SMOTE란

### Synthetic Minority Oversampling Technique



SMOTE의 동작 방식은 데이터의 개수가 적은 클래스의 표본을 가져온 뒤 임의의 값을 추가하여 새로운 샘플을 만들어 데이터에 추가하는 오버샘플링 방식이다.

# Data Part : Hindsight

Intra-class Cutmix은 좋은 결과를 냈다.

구상모  
\_T4008

0.7786 →  
0.7817

81.0159 →  
80.9365

상세  
보기

2022-11-02  
01:05



- 다만, 최종 제출에서는 Intra-class Cutmix대신 기존의 Cutmix를 이용하였다.

➔ (a) 성능이 더 잘나왔기 때문도 있고

➔ (b) 오버샘플링보단 각 Label 간의 경계를 더 robust하게 학습하는 기존의 CutMix가 더 일반화 성능이 높을 것이라고 생각했기 때문이다.

# Model Part :: Config Part 1

```
{
  "dataset": "MaskSplitByProfileDataset",
  "model": "EfficientB4",
  "augmentation": "CustomAugmentation",
  "optimizer": "Adam",
  "criterion": "f1",
  "seed": 7,
  "epochs": 10,
  "resize": [380, 380],
  "batch_size": 32,
  "valid_batch_size": 256,
  "lr": 0.0001,
  "val_ratio": 0.2,
  "lr_decay_step": 5,
  "log_interval": 20,
  "name": "exp",
  "cutmix": "yes",
  "cutmix_prob": 0.7,
  "cutmix_lower": 0.46,
  "cutmix_upper": 0.54,
  "val_train": "true",
  "val_epochs": 3,
  "age_removal": true,
}
```

(설명)

[1] dataset : SOTA에 쓰인 것은, 결론적으로는 MaskSplitbyProfileDataset.  
CutMixDataset은 앙상블에 이용하지 않았다.

[2] Model : EfficientB4

[3] Augmentation : CustomAugmentation

[4] Resize : [380, 380]

➔ EfficientB4가 이미지넷에서 학습되었던 데이터 사이즈인 [380,380]을 맞추어줬다.

[5] Batch\_size : 32

➔ 그 이상 64, 128을 넣으면 OOM이 일어났다.(이미지 해상도가 높아졌기 때문)

[6] age\_removal : True

➔ 경계값의 데이터를 제외하여 보다 뚜렷하게 구분할 수 있게끔 하였다.

# Model Part :: Config Part 2

```
{
  "dataset": "MaskSplitByProfileDataset",
  "model": "EfficientB4",
  "augmentation": "CustomAugmentation",
  "optimizer": "Adam",
  "criterion": "f1",
  "seed": 7,
  "epochs": 10,
  "resize": [380, 380],
  "batch_size": 32,
  "valid_batch_size": 256,
  "lr": 0.0001,
  "val_ratio": 0.2,
  "lr_decay_step": 5,
  "log_interval": 20,
  "name": "exp",
  "cutmix": "yes",
  "cutmix_prob": 0.7,
  "cutmix_lower": 0.46,
  "cutmix_upper": 0.54,
  "val_train": "true",
  "val_epochs": 3,
  "age_removal": true,
}
```

(설명)

[7] optimizer : Adam을 썼다.

- 초기 실험은 SGD로 진행했는데, Adam이 hyopt에 상관없이 빠르게 수렴한다고 하여 Adam을 택했다.

[8] lr : 0.0001로 진행하였다.

그 이상의 lr에서는 Adam의 학습 속도가 너무 빨라, 이를 0.0001로 낮추었다.

---

(Train part에서 얘기하고자 하는 것들)

[9] loss : cross\_entropy와 F1 loss를 동시에 진행하였는데, 최종 제출 모델은 F1 loss로 학습시켰음

[10] Cutmix : train 시에 cutmix를 적용하였으며, cutmix\_lower ~ cutmix\_upper를 통해 cutmix 되는 비율을 지정하였다.

[11] val\_train : validation data도 학습에 이용하였다.

**TLDR:** Adam makes it easy to converge quickly and on the first try. SGD+momentum takes more work to tune, but tends to reach better optima.

18.2K views · View 115 upvotes · View 4 shares · Answer requested by Brando Miranda



# Step 5. Training

Train  
strategy

- Custom cutmix(세로 방향)
- Validation as training





Error case  
Analysis

- Hintsights on data ; model debugging

Others

- Ensemble strategy & TTA etc...
- Metric learning

# Training part - CutMix

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	<b>78.6</b> (+2.3)
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	<b>47.3</b> (+1.0)
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	<b>76.7</b> (+1.1)

- 서로 다른 Label의 이미지 patch를 일부 섞는 기법
- 좌측의 예를 들면, 강아지를 0.6, 고양이를 0.4 섞었다고 알려주는 것  
(본래의 구현)
- Random하게 patch를 섞어야 함
- 그런데, 랜덤하게 섞는 것이 우리의 데이터셋에 걸맞는 방법일까???
- 마스크라는 특수성으로 인해, patch를 정말 random하게 가져가면 label의 정보를 상당 부분 잃을 수 있다.

상단의 사람의 마스크착용 여부를 아시겠습니까?  
또한, 하단의 사람의 성별과 나이를 아시겠습니까?



# Training part – CutMix :: 절반에 가깝게 세로로

(우리의 구현)

따라서, 이미지의 46~ 54%를 세로 방향으로 CutMix하고자 하였다.

이렇게 CutMix를 하는 것이 random하게 이미지를 쪼개는 것보단

label의 정보를 보존하는 것이라고 생각했고,

이를 통해 모델 성능을 끌어올릴 수 있었다.



# Validation data as Training

(우리의 구현)

학습이 끝난 후 Validation data를 Training 데이터로 사용하였다.

val\_ratio가 20%이므로, 20%의 데이터에서 더 많은 표현을 학습할 수 있으리라고 생각했다.

대회의 후반부에서 사용하였다.

# Model Debugging – wandb Table을 보면...

+ Add Panel

Table"]

label	pred	Age	Gender	Mask
4	1	Age_GT: 1, Age_pred: 1	gender_GT: 1, gender_pred: 0	Mask_GT: 0, Mask_pred: 0
16	15	Age_GT: 1, Age_pred: 0	gender_GT: 1, gender_pred: 1	Mask_GT: 2, Mask_pred: 2
4	3	Age_GT: 1, Age_pred: 0	gender_GT: 1, gender_pred: 1	Mask_GT: 0, Mask_pred: 0
5	4	Age_GT: 2, Age_pred: 1	gender_GT: 1, gender_pred: 1	Mask_GT: 0, Mask_pred: 0
3	0	Age_GT: 0, Age_pred: 0	gender_GT: 1, gender_pred: 0	Mask_GT: 0, Mask_pred: 0
6	9	Age_GT: 0, Age_pred: 0	gender_GT: 0, gender_pred: 1	Mask_GT: 1, Mask_pred: 1

← < 13 - 18 of 34 > →

Export as CSV Columns... Reset Table

[1] Mask가 틀렸다면 대체로 face detection 탓이다  
(17번 데이터의 경우)

→ 하여, ensemble 시 centercrop을 시킨 model도 포함시켰다.

[2] Age를 종종 틀린다.

→ 사람이 봐도 어렵다.

→ 이를 위해 metric-learning 등을 생각했지만  
준수한 성능이 나오지 않아 관뒀다.

# Others : Ensemble strategy and TTA

## [1] Thresholding

ImageID	best1	best2	my_best1	my_best2	my_val1	my_val2	similarity
c5dae7f9a9	1	0	2	0	1.2727816	1.0330424	true
71e1de7f2	2	1	2	1	1.4967771	0.4716686	false
482571b69	14	17	14	13	1.8219097	0.9414834	false
3ea8be82c	13	14	13	14	2.4922321	0.9524657	false
cad9b1b8e	2	1	2	1	1.3073969	1.238693	true
f381c7e47	10	11	10	11	2.7999167	0.8872521	false
b87147080	10	11	10	11	2.4261386	1.0741038	false
e6efc6c416	1	0	4	1	1.3562051	1.2281513	true
e48be606f	5	4	2	1	1.1659666	1.1174465	true

- Top-2 model이 다른 label을 가질 경우, 예측 값의 차이가 threshold 이하인 label을 더 데이터가 많은 쪽의 label로 바꿔줬음.
- 모델의 나이 예측이 정확하지 않았기 때문
- F1 : 0.002, acc : 0.2로 약간의 상승이 있었음.

## [2] Hard voting

- 최종 제출 시에는 Top-3를 hard-voting하였음.

## [3] TTA

- test-time 때 MTCNN을 쓰니, F1-score가 0.1 상승했는데, face부분을 잘 crop해주었기 때문이라 생각합니다.

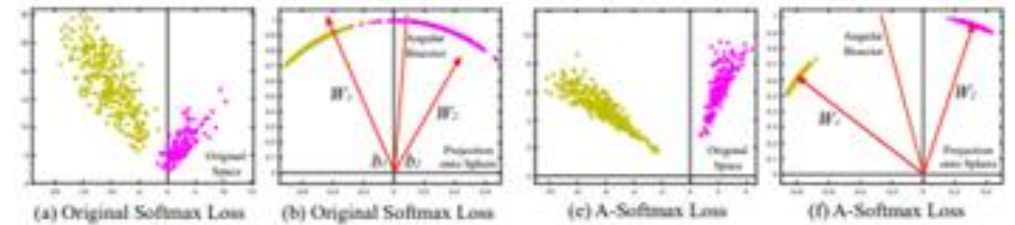
# Others : Metric learning for age

(Metric learning)

- 일반적인 classification → Feature 간의 decision boundary를 찾도록 학습
- Metric learning → 비유사한 Feature를 멀리 떨어지도록 학습

모델 디버깅 결과, 잘 구분이 안되는 age label 0, 1, 2에 대해 metric learning을 하면 어떨까?란 생각을 하였다.

그래서 Arcface를 도입시켰지만, 성능이 잘 나오지 않았다.



# Recap : Overall Problem & Solution

Problem	Solution
Image noise : 배경 정보, 얼굴의 크기 차이	MTCNN을 통한 processing
Label noise : 잘못된 label	외부 API + human annotation
Data value : 경계값에 있던 데이터	Age-removal :: 경계값(27~29, 57~59) 제외 학습
Data imbalance : 60대가 너무 적었음	[1] Sampler를 통해 Label을 고르게 load [2] Intra-class cutmix를 통해 oversampling을 시도해봤음 [3] Training 시 cutmix를 통해 Label 간의 경계를 더 robust하게 학습했음
Model overfitting : 수렴이 너무 빠름	[1] 파라미터가 적은 EfficientB4 [2] Dataset의 특성에 적합한 Augmentation을 통해 overfitting 방지 - Soft : RandErase, HorizontalFlip(그 외 변환은 마스크 정보 등을 왜곡)
Techniques : 성능을 올리기	[1] Training CutMix [2] Use val data for training [3] Model Ensemble



# Thank you.

로켓만큼 빠르게 성장하되, 뜻깊고 몸/마음이 건강한 부캠 기간을 보내시길.  
들어주셔서 진심으로 감사합니다.

