과학기술정보통신부 NIA 지능정보원 한국지능정보사회진흥원

DATA CREATOR CAMP

2024 데이터 크리에이터 캠프

대학부 팀명 : 부대찌개

Mission 1-1. 패션 스타일 이미지 분류 통계표 (1/2)

- 이미지 데이터 파일 명 형식을 바탕으로 각각의 데이터를 구분한 table 작성 후, training set과 validation set을 합쳐 하나의 table로 만듦.
- 만든 table을 바탕으로 이미지ID 수 기준 필요한 정보(성별, 스타일, 이미지 수)를 표 형식으로 기입.

데이터셋 이미지 구조

{W/T}_{이미지ID}_{스타일별}_{성별}.jpg

```
T_00456_10_sportivecasual_M.jpg
T_01123_90_hiphop_M.jpg
T_01514_50_ivy_M.jpg
T_06910_50_classic_W.jpg
T_07990_60_mods_M.jpg
T_14538_00_cityglam_W.jpg
T_21986_70_hippie_M.jpg
T_21988_70_hippie_M.jpg
T_21992_70_hippie_M.jpg
W_00004_50_ivy_M.jpg
W_00012_50_ivy_M.jpg
W_00028_50_ivy_M.jpg
W_00033_60_mods_M.jpg
W_00073_50_ivy_M.jpg
W_00103_70_hippie_M.jpg
W_00117_19_normcore_M.jpg
W_00152_50_feminine_W.jpg
```

```
# Train 데이터프레임(training_df)에서 'Sex'와 'style' 컬럼을 기준으로 그룹화하고, 각 그룹의 이미지 파일 수를 계산
# 'file_name' 컬럼의 개수를 세어 그룹별 이미지 파일 수를 나타냄
# 그룹별 파일 수를 기준으로 오름차순 정렬하고, 'count'라는 열 이름으로 데이터프레임 형태로 반환
training_df.groupby(['Sex', 'style'])['file_name'].count().sort_values(ascending=True).to_frame(name='count')
```

제시된 통계표 형식

성별	스타일	이미지 수
여성	feminine	
	classic	
	minimal	
	popart	
	…	
남성	ivy	
	mods	
	hippie	
	bold	
	…	

```
# Validation 데이터프레임(validation_df)에서 'Sex'와 'style' 컬럼을 기준으로 그룹화하고, 각 그룹의 이미지 파일 수를 계산
# 'file_name' 컬럼의 개수를 세어 그룹별 이미지 파일 수를 나타냄
# 그룹별 파일 수를 기준으로 오름차순 정렬하고, 'count'라는 열 이름으로 데이터프레임 형태로 반환
validation_df.groupby(['Sex', 'style'])['file_name'].count().sort_values(ascending=True).to_frame(name='count')
```

{성별}_{스타일별}_{이미지 수}

성별	스타일	이미지 수
M	normcore	364
	sportivecasual	298
	metrosexual	278
	hiphop	274
	mods	269
	bold	268
	hippie	260
	ivy	237
W	sportivecasual	157
	feminine	154
	normcore	153
	minimal	139
	powersuit	120
	bodyconscious	95
	hippie	91

Training, Validation에 대해서 각각 작성함(전체 코드 참고).

Mission 1-1. 패션 스타일 이미지 분류 통계표 (2/2)

Training Image : 4070개

	file_name	W/T	image_id	period	style	Sex
0	W_01752_00_metrosexual_M.jpg	W	01752	00	metrosexual	M
1	W_46417_70_military_W.jpg	W	46417	70	military	W
2	W_01509_00_metrosexual_M.jpg	W	01509	00	metrosexual	M
3	W_18951_50_feminine_W.jpg	W	18951	50	feminine	W
4	W_29485_10_sportivecasual_M.jpg	W	29485	10	sportivecasual	M
...	...	...	...	...	...	...
4065	W_03060_50_feminine_W.jpg	W	03060	50	feminine	W
4066	W_00032_10_sportivecasual_M.jpg	W	00032	10	sportivecasual	M
4067	W_02918_10_sportivecasual_M.jpg	W	02918	10	sportivecasual	M
4068	W_08957_19_normcore_W.jpg	W	08957	19	normcore	W
4069	W_00465_50_classic_W.jpg	W	00465	50	classic	W

4070 rows × 6 columns

Validation Image : 951개

	file_name	W/T	image_id	period	style	Sex
0	W_14468_10_sportivecasual_W.jpg	W	14468	10	sportivecasual	W
1	W_17239_19_normcore_M.jpg	W	17239	19	normcore	M
2	W_06895_00_metrosexual_M.jpg	W	06895	00	metrosexual	M
3	W_13354_50_feminine_W.jpg	W	13354	50	feminine	W
4	W_07018_19_normcore_M.jpg	W	07018	19	normcore	M
...	...	...	...	...	...	...
946	T_21988_70_hippie_M.jpg	T	21988	70	hippie	M
947	W_28022_50_ivy_M.jpg	W	28022	50	ivy	M
948	W_00551_19_normcore_M.jpg	W	00551	19	normcore	M
949	W_03144_50_classic_W.jpg	W	03144	50	classic	W
950	W_28453_10_sportivecasual_M.jpg	W	28453	10	sportivecasual	M

951 rows × 6 columns

{W/T}_{이미지ID}_{스타일별}_{성별}.jpg
을 분리하여 만든 데이터 프레임

Training + Validation 통합 데이터 프레임

	file_name	W/T	image_id	period	style	Sex	T/V
0	W_01752_00_metrosexual_M.jpg	W	01752	00	metrosexual	M	Train
1	W_46417_70_military_W.jpg	W	46417	70	military	W	Train
2	W_01509_00_metrosexual_M.jpg	W	01509	00	metrosexual	M	Train
3	W_18951_50_feminine_W.jpg	W	18951	50	feminine	W	Train
4	W_29485_10_sportivecasual_M.jpg	W	29485	10	sportivecasual	M	Train
...	...	...	...	...	...	...	...
5016	T_21988_70_hippie_M.jpg	T	21988	70	hippie	M	Validation
5017	W_28022_50_ivy_M.jpg	W	28022	50	ivy	M	Validation
5018	W_00551_19_normcore_M.jpg	W	00551	19	normcore	M	Validation
5019	W_03144_50_classic_W.jpg	W	03144	50	classic	W	Validation
5020	W_28453_10_sportivecasual_M.jpg	W	28453	10	sportivecasual	M	Validation

5021 rows × 7 columns

{파일 명}_{W/T}_{이미지ID}_{시대}_{스타일별}_{성별}_{T/V}

최종 '성별 & 스타일' 통계표 (이미지 파일)

Training

Sex	style	count
W	grunge	31
	military	33
	disco	37
	space	37
	popart	41
	lounge	45
	hiphop	48
	lingerie	55
	ecology	64
	punk	65
	cityglam	67
	athleisure	67
	classic	77
	genderless	77
	oriental	78
	hippie	91
	kitsch	91
	bodyconscious	95
	powersuit	120
	minimal	139
	normcore	153
	feminine	154
	sportivecasual	157
M	ivy	237
	hippie	260
	bold	268
	mods	269
	hiphop	274
	metrosexual	278
	sportivecasual	298
	normcore	364

Validation

Sex	style	count
W	lingerie	5
	popart	8
	hiphop	8
	lounge	8
	military	9
	disco	10
	grunge	10
	genderless	12
	punk	12
	hippie	14
	athleisure	14
	space	15
	ecology	17
	cityglam	18
	oriental	18
	normcore	20
	classic	22
	kitsch	22
	bodyconscious	23
	powersuit	34
	minimal	35
	feminine	44
	sportivecasual	48
M	normcore	51
	sportivecasual	52
	bold	57
	metrosexual	58
	hiphop	66
	ivy	79
	mods	80
	hippie	82

{성별}_{스타일별}_{이미지 수}

Mission 1-2. 성별 & 스타일 분류 모델 구현 (1/14)

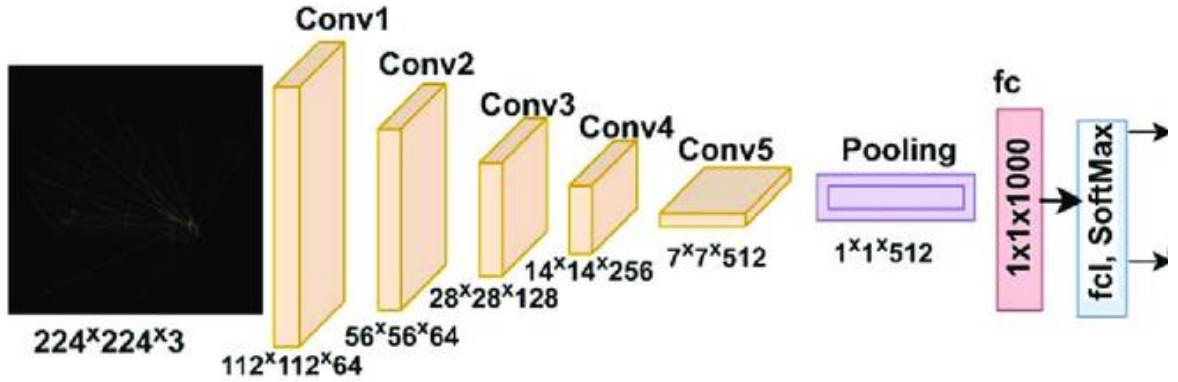
◆ (문제) ResNet-18로 "성별 & 스타일" 단위로 클래스를 분류하고 Validation 데이터에 대한 정확도 제시

- ResNet-18이란?

→ 2015년도 ILSVRC(ImageNet Large Sclae Visual Recognition Challenge)에서 우승한 CNN 네크워크로, Layer가 깊어질수록 성능이 좋아지다가, 떨어지는 현상인 Gradient Vanishing를 이전 층의 출력을 다음 층의 입력에 그대로 추가하는 Skip Connection을 이용한 Residual Learning으로 해결한 모델.

여러구조가존재하는데, 본과제에선 Resnet-18구조를 이용하고 사전가중치는 활용할수없다.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 [3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



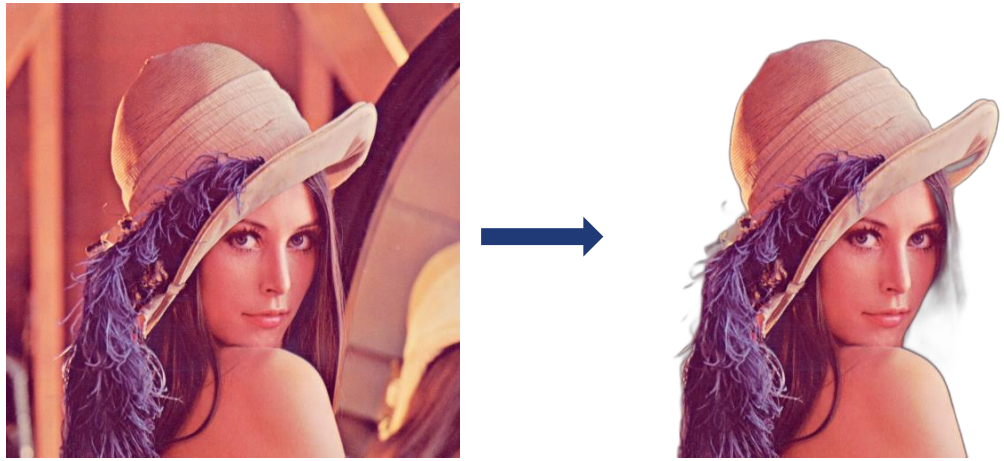
ResNet-18 아키텍처

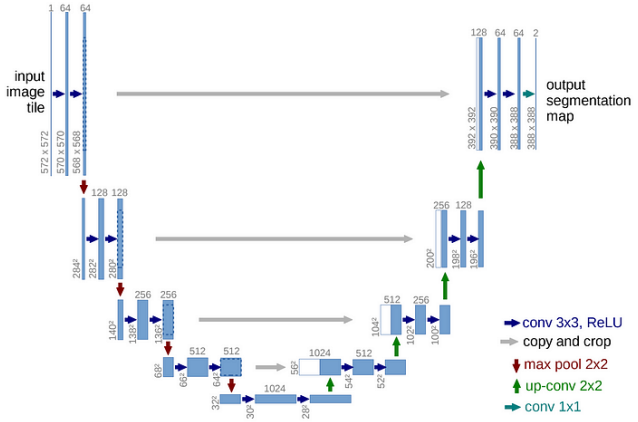
Mission 1-2. 성별 & 스타일 분류 모델 구현 (2/14)

◆ 배경 제거 전처리

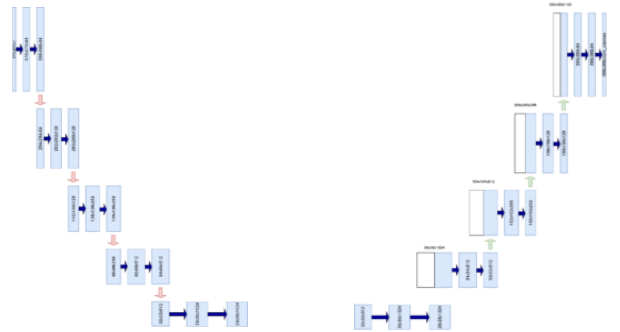
➢ 패션 분류에 혼동을 줄 수 있는 배경 제거
- 파이썬 배경 제거 Rembg 라이브러리 사용
  - U2net_human_seg 모델 사용
  - 일반적인 segmentation 아키텍쳐인 u2net의 사람 분리용 Fine-Tuning모델

➢ 적용한 특징
- 패션이 눈에 띄도록 흰색 배경으로 변경
- multiprocessing 라이브러리를 통한 병렬 처리 (장당 1초->50개씩 처리)





U-Net 아키텍처
Contracting path, Expansive path



Mission 1-2. 성별 & 스타일 분류 모델 구현 (3/14)

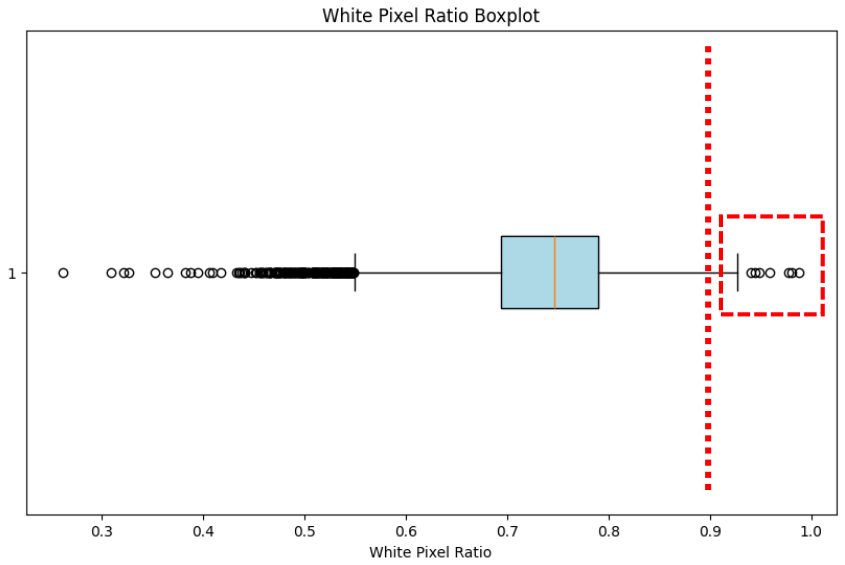
◆ 배경 제거 전처리

➢ U-net기반 모델 배경 제거 결과 확인

- 옷이 지워지는 경우 발생

-〉 이러한 이미지들을 확인하기 위해 IQR을 활용한 손상 이미지 탐지 방식 구현

- 이미지별 white pixel이 차지하는 비율 계산
- 계산된 값을 입력으로 하여 Interquartile Range(IQR) 계산
- 일정 범위를 초과하는 이미지를 손상된 이미지(이상치)로 판단

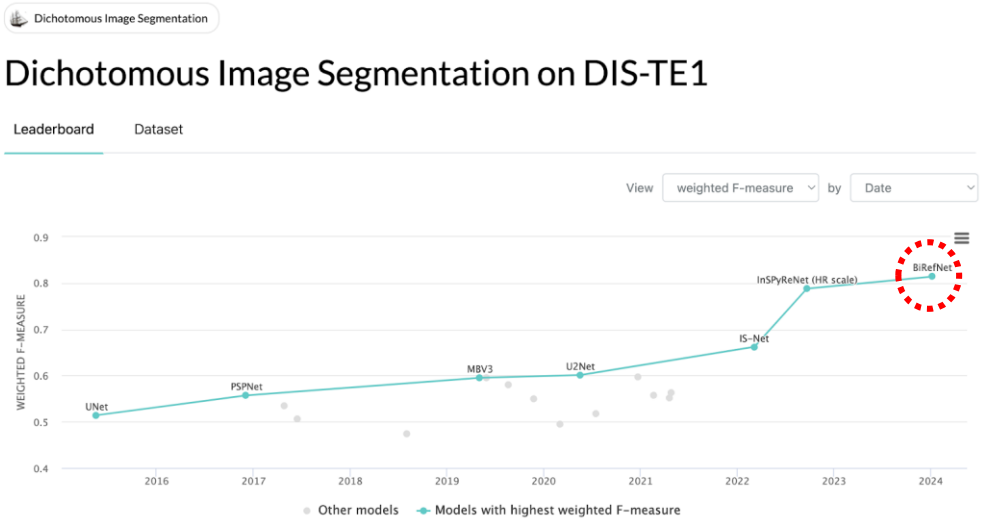


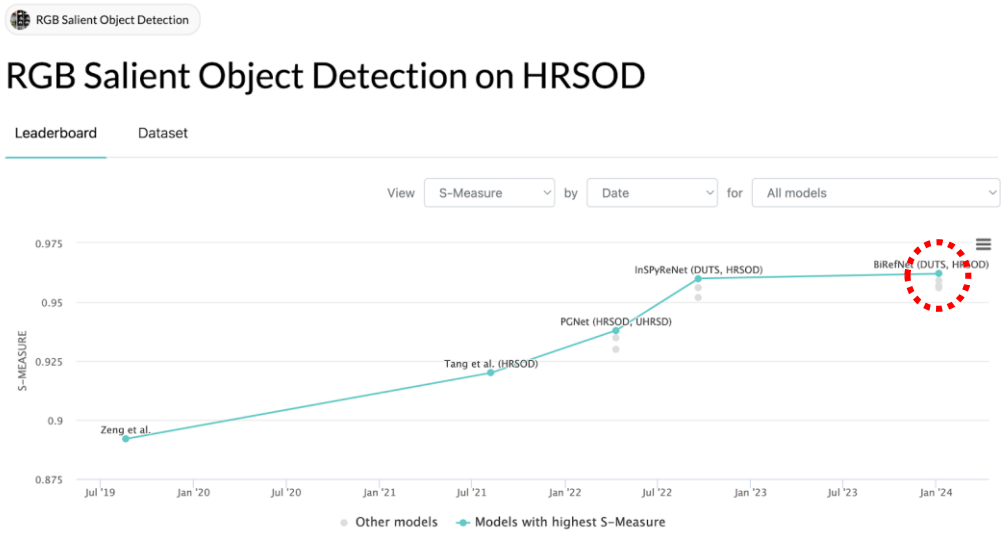
Mission 1-2. 성별 & 스타일 분류 모델 구현 (4/14)

◆ 배경 제거 전처리

➢ U-net기반 모델로 한계가 있는 것 같아 잘못 분리된 이미지에 Image Segmentation 분야 SOTA 모델 적용

- U2net_human_seg → birefnet
- birefnet 은 DIS(Depth Image Segmentation, 깊이 정보를 사용하는 이미지 세분화 작업), COD(Camouflaged Object Detection, 위장된 객체를 탐지하는 작업), HRSOD(High-Resolution Salient Object Detection, 고해상도 이미지를 기반으로 주목할 만한 객체를 탐지하는 작업)에서 우수한 성과로 2024년 SOTA를 달성한 Image Segmentation모델





Mission 1-2. 성별 & 스타일 분류 모델 구현 (5/14)

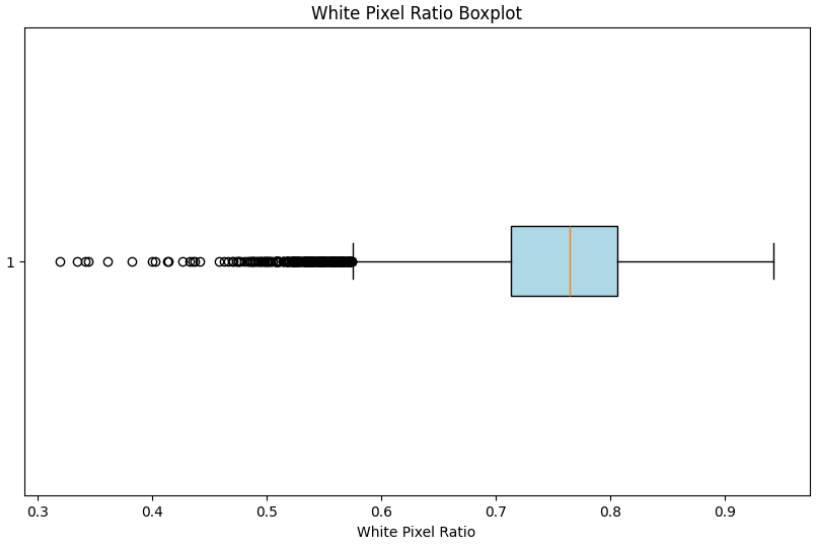
◆ 배경 제거 전처리

➢ u2net_human_seg → birefnet-portrait

- Birefnet의 여러 훈련 버전 중에서도 인물 초상화를 분리하는 것에 특화된 Tuning버전 이용.
- 잘못 분리되었던 이미지들이 모두 개선됨
  - ✓ 처리에 오랜 시간이 걸리더라도 이미지 배경 제거를 Birefnet으로 전처리하기로 결정
  - ✓ 이전 슬라이드와 같은 방식으로 IQR을 계산
  - ✓ 이상치로 판단되는 범위에 있는 이미지도 모두 정상

손상 이미지

개선 이미지

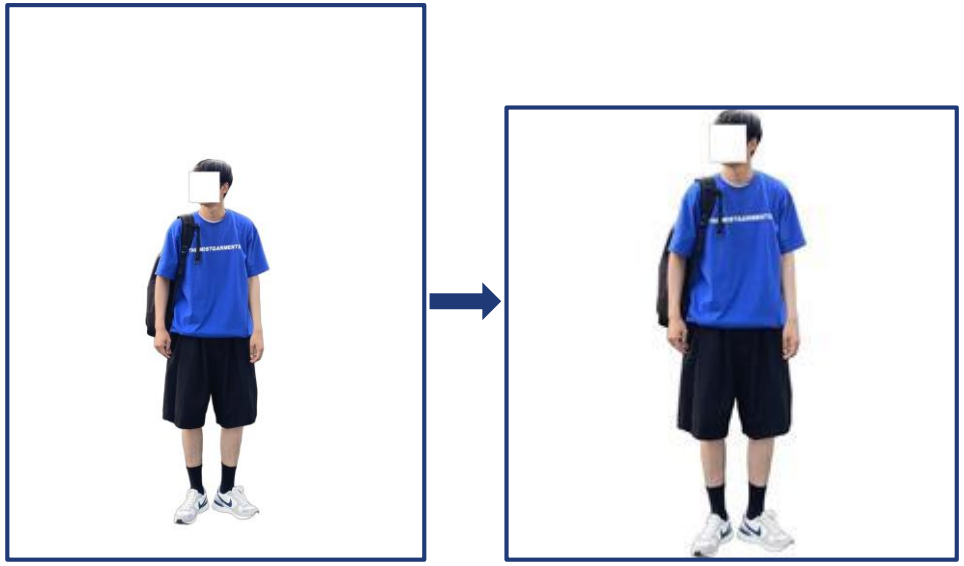


Mission 1-2. 성별 & 스타일 분류 모델 구현 (6/14)

◆ 이미지 관심 영역 추출 및 리사이징

➢ 기존 이미지의 사이즈가(3000x4000) 크므로, 이후 처리 속도를 위해 resize 필요

- 사이즈는 Resnet 기본 구조의 입력인 224x224 로 결정.
- 피사체의 위치가 제각각이라 CV의 윤곽선 검출을 통해 가장 큰 윤곽선을 감싸는 바운딩 박스를 검출하여 Crop 함으로써 모든 피사체가 중앙에 오도록 전처리
- 리사이징 시에는 패션 비율을 유지하도록 Width padding을 통해 height와 동일 비율 유지
- 전처리 작업들은 Validation Image에도 동일하게 적용해준다.

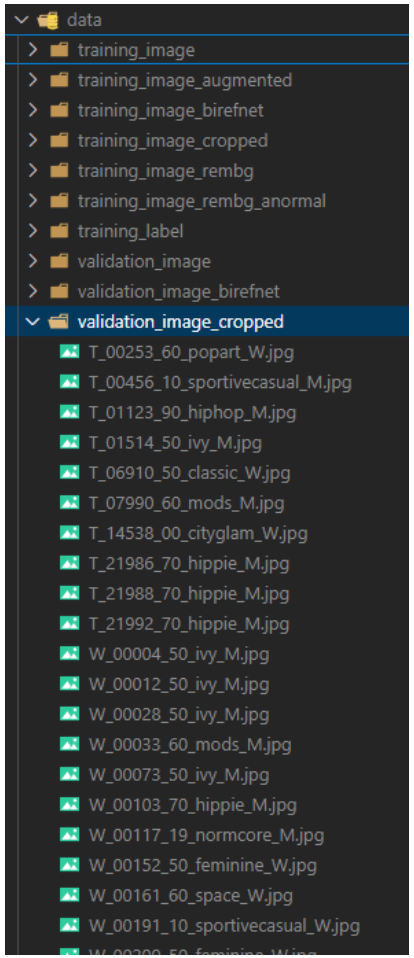


Resize 전 Resize 후

Resize 전 Resize 후

Mission 1-2. 성별 & 스타일 분류 모델 구현 (7/14)

◆ 훈련용 데이터셋 설계



Data폴더 트리

- Class 이름이 폴더 구조가 아닌 이름으로 되어 있고 한 폴더에 있으므로 일반적인 Pytorch 데이터셋 구조가 아닌 커스텀 데이터셋을 구현
- 최종 전처리된 데이터인 training_image_cropped, validation_image_cropped 폴더 이용
- 구현한 데이터셋을 이용하여 이미지를 로드하고, ResNet 입력시 이미지의 정규화를 위해 RGB 채널별 평균과 표준 편차를 구함.

```python
from tqdm import tqdm

# 평균과 표준편차 계산 함수
def calculate_mean_std(dataset):
    mean = 0.0
    std = 0.0
    total_images = len(dataset)
    for img, _ in tqdm(dataset):
        # 이미지의 채널 차원에 대해 계산
        mean += img.mean(dim=[1, 2])  # (C,)
        std += img.std(dim=[1, 2])    # (C,)

    mean /= total_images
    std /= total_images

    return mean, std

# 평균과 표준편차 계산
mean, std = calculate_mean_std(dataset)
print(f"Mean: {mean}, Standard Deviation: {std}")
```

```
100%|██████████| 4070/4070 [07:56<00:00,  8.54it/s]
Mean: tensor([0.8752, 0.8656, 0.8636]), Standard Deviation: tensor([0.2598, 0.2734, 0.2761])
```

training_image_cropped 기준

RGB이미지 평균:
(0.8752, 0.8656, 0.8636)
RGB이미지 표준편차:
(0.2598, 0.2734, 0.2761)

Mission 1-2. 성별 & 스타일 분류 모델 구현 (8/14)

◆ 불균형 클래스 처리 및 데이터 증강

```
{'grunge_W': 31,
 'military_W': 33,
 'space_W': 37,
 'disco_W': 37,
 'popart_W': 41,
 'lounge_W': 45,
 'hiphop_W': 48,
 'lingerie_W': 55,
 'ecology_W': 64,
 'punk_W': 65,
 'athleisure_W': 67,
 'cityglam_W': 67,
 'classic_W': 77,
 'genderless_W': 77,
 'oriental_W': 78,
 'kitsch_W': 91,
 'hippie_W': 91,
 'bodyconscious_W': 95,
 'powersuit_W': 120,
 'minimal_W': 139,
 'normcore_W': 153,
 'feminine_W': 154,
 'sportivecasual_W': 157,
 'ivy_M': 237,
 'hippie_M': 260,
 'bold_M': 268,
 'mods_M': 269,
 'hiphop_M': 274,
 'metrosexual_M': 278,
 'sportivecasual_M': 298,
 'normcore_M': 364}
```

```
metrosexual_M: 현재 샘플수: 278,  1722개 추가 샘플 필요, 증강 배율: 6
military_W: 현재 샘플수: 33,  1967개 추가 샘플 필요, 증강 배율: 60
feminine_W: 현재 샘플수: 154,  1846개 추가 샘플 필요, 증강 배율: 12
sportivecasual_M: 현재 샘플수: 298,  1702개 추가 샘플 필요, 증강 배율: 6
hiphop_M: 현재 샘플수: 274,  1726개 추가 샘플 필요, 증강 배율: 6
space_W: 현재 샘플수: 37,  1963개 추가 샘플 필요, 증강 배율: 53
mods_M: 현재 샘플수: 269,  1731개 추가 샘플 필요, 증강 배율: 6
normcore_M: 현재 샘플수: 364,  1636개 추가 샘플 필요, 증강 배율: 4
hippie_M: 현재 샘플수: 260,  1740개 추가 샘플 필요, 증강 배율: 7
sportivecasual_W: 현재 샘플수: 157,  1843개 추가 샘플 필요, 증강 배율: 12
powersuit_W: 현재 샘플수: 120,  1880개 추가 샘플 필요, 증강 배율: 16
normcore_W: 현재 샘플수: 153,  1847개 추가 샘플 필요, 증강 배율: 12
kitsch_W: 현재 샘플수: 91,  1909개 추가 샘플 필요, 증강 배율: 21
ivy_M: 현재 샘플수: 237,  1763개 추가 샘플 필요, 증강 배율: 7
bold_M: 현재 샘플수: 268,  1732개 추가 샘플 필요, 증강 배율: 6
classic_W: 현재 샘플수: 77,  1923개 추가 샘플 필요, 증강 배율: 25
oriental_W: 현재 샘플수: 78,  1922개 추가 샘플 필요, 증강 배율: 25
athleisure_W: 현재 샘플수: 67,  1933개 추가 샘플 필요, 증강 배율: 29
punk_W: 현재 샘플수: 65,  1935개 추가 샘플 필요, 증강 배율: 30
genderless_W: 현재 샘플수: 77,  1923개 추가 샘플 필요, 증강 배율: 25
ecology_W: 현재 샘플수: 64,  1936개 추가 샘플 필요, 증강 배율: 30
bodyconscious_W: 현재 샘플수: 95,  1905개 추가 샘플 필요, 증강 배율: 20
hiphop_W: 현재 샘플수: 48,  1952개 추가 샘플 필요, 증강 배율: 41
disco_W: 현재 샘플수: 37,  1963개 추가 샘플 필요, 증강 배율: 53
minimal_W: 현재 샘플수: 139,  1861개 추가 샘플 필요, 증강 배율: 13
grunge_W: 현재 샘플수: 31,  1969개 추가 샘플 필요, 증강 배율: 64
lounge_W: 현재 샘플수: 45,  1955개 추가 샘플 필요, 증강 배율: 43
hippie_W: 현재 샘플수: 91,  1909개 추가 샘플 필요, 증강 배율: 21
popart_W: 현재 샘플수: 41,  1959개 추가 샘플 필요, 증강 배율: 48
cityglam_W: 현재 샘플수: 67,  1933개 추가 샘플 필요, 증강 배율: 29
lingerie_W: 현재 샘플수: 55,  1945개 추가 샘플 필요, 증강 배율: 35
```

- ➢ Class별 이미지 개수
  - 31 ~ 364
- 모델을 학습하기에 부족함.
- 심각한 클래스 불균형 존재
- 따라서 분포별 전략적 증강 적용
- 클래스당 2000장 정도를 목표로 각각의 클래스에 대해 증강 배율 계산

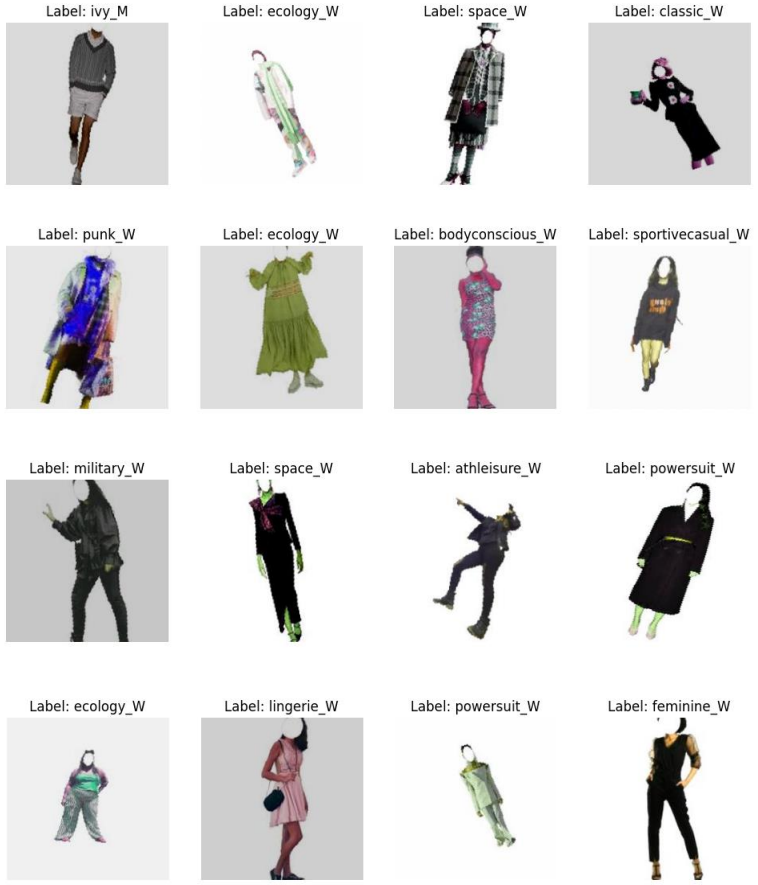
- ➢ 패션에 알맞은 증강 방식 적용
  - ❖ 랜덤 수평 뒤집기
  - ❖ 랜덤 각도, 기울기 변환
  - ❖ 밝기, 대비, 색상 조정
  - ❖ 랜덤 크롭 및 리사이즈

*증강시 패딩이 생기는 부분은 기존 하얀 배경과 통일하기 위해 흰 패딩으로 설정

Mission 1-2. 성별 & 스타일 분류 모델 구현 (9/14)

◆ 불균형 클래스 처리 및 데이터 증강

```
전체 샘플 수: 61582
클래스 개수: 31
1. 클래스: sportivecasual_M, 샘플 수: 2086, 비율: 3.39%
2. 클래스: hippie_M, 샘플 수: 2080, 비율: 3.38%
3. 클래스: sportivecasual_W, 샘플 수: 2041, 비율: 3.31%
4. 클래스: powersuit_W, 샘플 수: 2040, 비율: 3.31%
5. 클래스: oriental_W, 샘플 수: 2028, 비율: 3.29%
6. 클래스: hiphop_W, 샘플 수: 2016, 비율: 3.27%
7. 클래스: punk_W, 샘플 수: 2015, 비율: 3.27%
8. 클래스: grunge_W, 샘플 수: 2015, 비율: 3.27%
9. 클래스: military_W, 샘플 수: 2013, 비율: 3.27%
10. 클래스: athleisure_W, 샘플 수: 2010, 비율: 3.26%
11. 클래스: cityglam_W, 샘플 수: 2010, 비율: 3.26%
12. 클래스: popart_W, 샘플 수: 2009, 비율: 3.26%
13. 클래스: feminine_W, 샘플 수: 2002, 비율: 3.25%
14. 클래스: kitsch_W, 샘플 수: 2002, 비율: 3.25%
15. 클래스: classic_W, 샘플 수: 2002, 비율: 3.25%
16. 클래스: genderless_W, 샘플 수: 2002, 비율: 3.25%
17. 클래스: hippie_W, 샘플 수: 2002, 비율: 3.25%
18. 클래스: space_W, 샘플 수: 1998, 비율: 3.24%
19. 클래스: disco_W, 샘플 수: 1998, 비율: 3.24%
20. 클래스: bodyconscious_W, 샘플 수: 1995, 비율: 3.24%
21. 클래스: normcore_W, 샘플 수: 1989, 비율: 3.23%
22. 클래스: ecology_W, 샘플 수: 1984, 비율: 3.22%
23. 클래스: lounge_W, 샘플 수: 1980, 비율: 3.22%
24. 클래스: lingerie_W, 샘플 수: 1980, 비율: 3.22%
25. 클래스: metrosexual_M, 샘플 수: 1946, 비율: 3.16%
26. 클래스: minimal_W, 샘플 수: 1946, 비율: 3.16%
27. 클래스: hiphop_M, 샘플 수: 1918, 비율: 3.11%
28. 클래스: ivy_M, 샘플 수: 1896, 비율: 3.08%
29. 클래스: mods_M, 샘플 수: 1883, 비율: 3.06%
30. 클래스: bold_M, 샘플 수: 1876, 비율: 3.05%
31. 클래스: normcore_M, 샘플 수: 1820, 비율: 2.96%
```



- ➢ 증강된 수치
  - 전체 4070장 -〉 61582장 (약15배)
- 클래스 불균형 해소됨.
- 전반적으로 증강 적용됨.
- 각각의 클래스 모두 2000장에 가까움

- ➢ 패션에 알맞은 증강 방식 적용
  - ❖ 랜덤 수평 뒤집기
  - ❖ 랜덤 각도, 기울기 변환
  - ❖ 밝기, 대비, 색상 조정
  - ❖ 랜덤 크롭 및 리사이즈

*증강시 패딩이 생기는 부분은 기존 하얀 배경과 통일하기 위해 흰 패딩으로 설정

Mission 1-2. 성별 & 스타일 분류 모델 구현 (10/14)

◆ 샘플링, 인코딩, 정규화

샘플링

- PyTorch의 WeightedRandomSampler를 사용
- 약간 일치하지 않는 클래스 분포를 맞춰주는 역할
- 개수가 부족한 클래스에는 높은 가중치, 개수가 많은 클래스에는 낮은 가중치를 부여함
- 학습시 각 이미지 클래스가 사전에 계산된 가중치대로 모델 학습에 제공됨.

```
전체 샘플 수: 61582
클래스 개수: 31
클래스별 샘플 수:[('sportivecasual_M', 2086), ('hippie_M', 2080), ('sportivecasual_W', 2041), ('powersuit_W', 2040), ('oriental_W', 2028), ('hipho
각 클래스별 적용된 노출 예상 배율:
{'metrosexual_M': 1.0208202101912842, 'military_W': 0.986843581238072, 'feminine_W': 0.9922657987174021, 'sportivecasual_M': 0.952308786688513,
```

클래스 인코딩

- Categorical feature → Numerical feature
- 학습을 가능하게 함
- Label Encoder 사용

```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoder.fit([label for label in concat_dataset.labels])
```

정규화

- 이미지마다 명도, 채도, 대비 등이 다르므로 학습 전에 이미지를 정규화할 필요가 있음
- 이전에 계산한 평균:(0.8752, 0.8656, 0.8636) 표준편차: (0.2598, 0.2734, 0.2761) 이용

```python
# 데이터셋과 데이터로더 생성
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.8752, 0.8656, 0.8636], std=[0.2598, 0.2734, 0.2761])
])
```

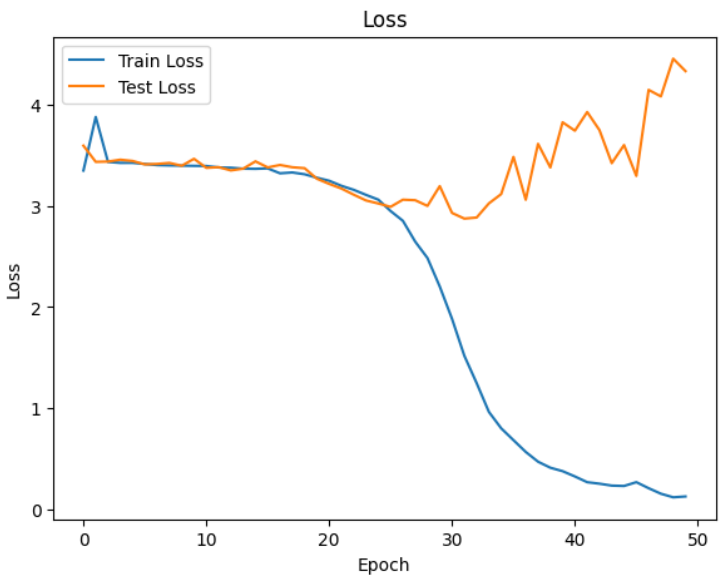
Mission 1-2. 성별 & 스타일 분류 모델 구현 (11/14)

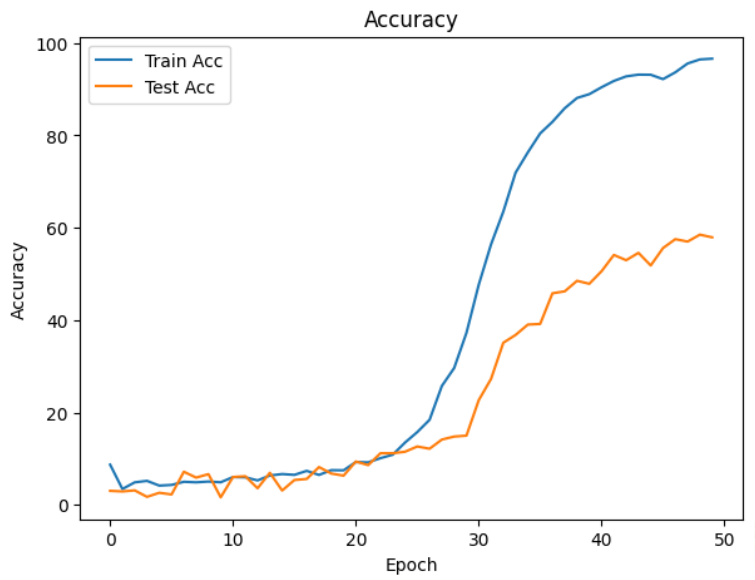
◆ Resnet-18 구현 및 훈련

- PyTorch에 구현된 기본 resnet18 구조 사용(models.resnet18(weights=None))
- 미션 제약에 따라 사전 가중치는 불러오지 않음
- 클래스 개수에 맞게 최종 FC레이어 구조만 변경(final layer dimension = linear layer input x num of classes)
- Loss function: Cross Entropy
- Optimizer: Adam, 따로 스케줄러는 구현하지 않음.
- 먼저 증강하지 않은 데이터를 가지고 훈련하여 베이스라인 정확도를 확보함

```
Epoch 30/50, Train Loss: 2.2032, Train Acc: 37.25%, Test Loss: 3.1937, Test Acc: 15.00%
Epoch 31/50, Train Loss: 1.8840, Train Acc: 47.66%, Test Loss: 2.9292, Test Acc: 22.68%
Epoch 32/50, Train Loss: 1.5200, Train Acc: 56.38%, Test Loss: 2.8733, Test Acc: 27.27%
Epoch 33/50, Train Loss: 1.2505, Train Acc: 63.49%, Test Loss: 2.8836, Test Acc: 35.11%
Epoch 34/50, Train Loss: 0.9657, Train Acc: 71.93%, Test Loss: 3.0233, Test Acc: 36.80%
Epoch 35/50, Train Loss: 0.8040, Train Acc: 76.33%, Test Loss: 3.1152, Test Acc: 39.05%
Epoch 36/50, Train Loss: 0.6868, Train Acc: 80.42%, Test Loss: 3.4825, Test Acc: 39.18%
Epoch 37/50, Train Loss: 0.5707, Train Acc: 82.93%, Test Loss: 3.0593, Test Acc: 45.82%
Epoch 38/50, Train Loss: 0.4742, Train Acc: 85.87%, Test Loss: 3.6109, Test Acc: 46.24%
Epoch 39/50, Train Loss: 0.4150, Train Acc: 88.09%, Test Loss: 3.3779, Test Acc: 48.52%
Epoch 40/50, Train Loss: 0.3807, Train Acc: 88.92%, Test Loss: 3.8236, Test Acc: 47.87%
Epoch 41/50, Train Loss: 0.3286, Train Acc: 90.42%, Test Loss: 3.7407, Test Acc: 50.60%
Epoch 42/50, Train Loss: 0.2720, Train Acc: 91.78%, Test Loss: 3.9251, Test Acc: 54.12%
Epoch 43/50, Train Loss: 0.2572, Train Acc: 92.77%, Test Loss: 3.7464, Test Acc: 52.98%
Epoch 44/50, Train Loss: 0.2384, Train Acc: 93.14%, Test Loss: 3.4204, Test Acc: 54.57%
Epoch 45/50, Train Loss: 0.2346, Train Acc: 93.12%, Test Loss: 3.5992, Test Acc: 51.84%
Epoch 46/50, Train Loss: 0.2724, Train Acc: 92.17%, Test Loss: 3.2946, Test Acc: 55.61%
Epoch 47/50, Train Loss: 0.2128, Train Acc: 93.65%, Test Loss: 4.1436, Test Acc: 57.53%
Epoch 48/50, Train Loss: 0.1582, Train Acc: 95.56%, Test Loss: 4.0791, Test Acc: 57.01%
Epoch 49/50, Train Loss: 0.1230, Train Acc: 96.44%, Test Loss: 4.4514, Test Acc: 58.51%
Epoch 50/50, Train Loss: 0.1311, Train Acc: 96.60%, Test Loss: 4.3289, Test Acc: 57.93%
Training time: 4.0m 56.95s
```

기본 전처리된 데이터의
50 에포크 기준 Best ACC
58.51%





Mission 1-2. 성별 & 스타일 분류 모델 구현 (12/14)

◆ Resnet-18 최종 구현

- 훈련시, 훈련이 진행될때가 있고 안될때가 있어 로컬 미니멈이나 가중치 소실 문제가 발생한다고 생각해 초기 가중치 초기화 방식에 Kaiming Uniform 초기화를 적용하여 무작위 값을 할당하였음.
- Kaiming 초기화(Kaiming Initialization)는 깊은 신경망을 효과적으로 학습시키기 위해 제안된 가중치 초기화 방법으로, 가중치를 정규 분포 또는 균등 분포에서 무작위로 초기화하되, 입력 노드의 수를 전달하는 방식으로 가중치의 크기가 ReLU 비선형 함수와 맞춰 신호가 사라지거나 폭발하지 않고 안정적으로 전달될 수 있도록 돕고 학습의 속도를 빠르게 한다.
- 또한 랜덤성을 제어하기 위해 랜덤 모듈, 넘파이 모듈, 토치 모듈의 시드를 42로 고정함.

```python
# 랜덤 시드 값
seed = 42

# 파이썬의 랜덤 모듈
random.seed(seed)

# 넘파이 랜덤 모듈
np.random.seed(seed)

# 토치 랜덤 모듈
torch.manual_seed(seed)

# CUDA를 사용하는 경우
if torch.cuda.is_available():
    torch.cuda.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
```

```python
# ResNet18 모델 초기화 (사전 가중치 사용 안 함)
model = models.resnet18(weights=None)
model.fc = nn.Linear(model.fc.in_features, len(label_encoder.classes_))  # 마지막 레이어 변경

# kaiming_uniform 초기화 함수 적용
def weights_init_he(m):
    if isinstance(m, nn.Conv2d) or isinstance(m, nn.Linear):
        nn.init.kaiming_uniform_(m.weight, nonlinearity='relu')
        if m.bias is not None:
            nn.init.zeros_(m.bias)

model.apply(weights_init_he)
```

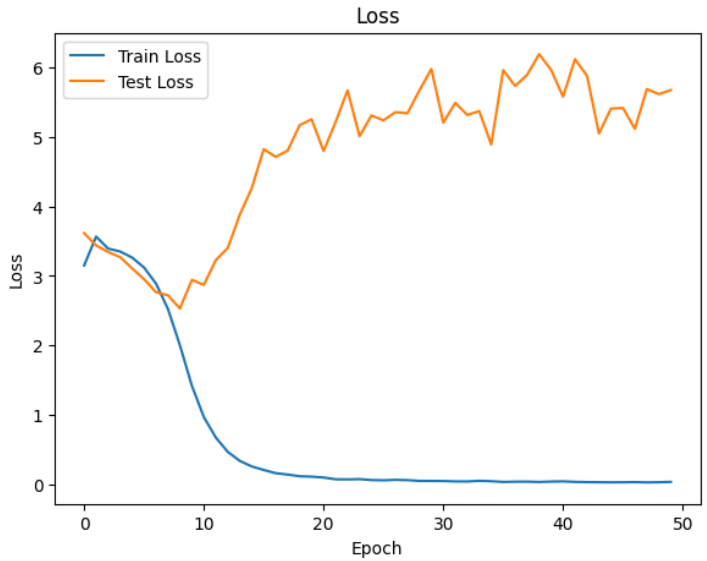
Mission 1-2. 성별 & 스타일 분류 모델 구현 (13/14)

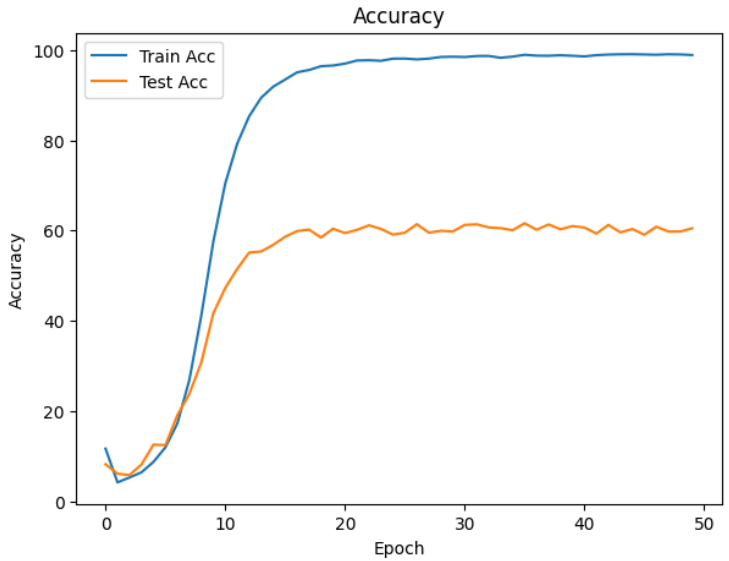
◆ Resnet-18 최종 훈련 및 결과

- 최적의 정확도 모델을 사용하기 위해 CheckPoint로 이전 Epoch보다 높은 성능만 저장되도록 구현함.
- 데이터 증강, 모든 전처리, 모델링의 최종 결과는 다음과 같다.
- 최고 정확도는 61.6%로 그래프로 보면 이미 20에포크 쯤에 수렴하고, 오차는 발산하는 것을 볼 수 있음.
- 멘토님에게 문의해본 결과, 훈련과 검증셋에 중복이 있어 정확도는 유지되도 오차가 커질 수 있다고 한다.
- 증강과 튜닝으로 베이스 라인 모델 정확도 대비, 58.51%->61.6% 3%정도의 향상이 있음.

```
Epoch 35/50, Train Loss: 0.0488, Train Acc: 98.52%, Test Loss: 4.8903, Test Acc: 60.07%
Epoch 36/50, Train Loss: 0.0376, Train Acc: 98.92%, Test Loss: 5.9592, Test Acc: 61.60%
Checkpoint saved at epoch 36
Epoch 37/50, Train Loss: 0.0417, Train Acc: 98.74%, Test Loss: 5.7312, Test Acc: 60.17%
Epoch 38/50, Train Loss: 0.0421, Train Acc: 98.71%, Test Loss: 5.8911, Test Acc: 61.34%
Epoch 39/50, Train Loss: 0.0376, Train Acc: 98.82%, Test Loss: 6.1881, Test Acc: 60.27%
Epoch 40/50, Train Loss: 0.0433, Train Acc: 98.71%, Test Loss: 5.9635, Test Acc: 60.98%
Epoch 41/50, Train Loss: 0.0463, Train Acc: 98.57%, Test Loss: 5.5779, Test Acc: 60.66%
Epoch 42/50, Train Loss: 0.0387, Train Acc: 98.84%, Test Loss: 6.1165, Test Acc: 59.33%
Epoch 43/50, Train Loss: 0.0353, Train Acc: 98.97%, Test Loss: 5.8774, Test Acc: 61.24%
Epoch 44/50, Train Loss: 0.0337, Train Acc: 99.03%, Test Loss: 5.0483, Test Acc: 59.58%
Epoch 45/50, Train Loss: 0.0323, Train Acc: 99.05%, Test Loss: 5.4054, Test Acc: 60.33%
Epoch 46/50, Train Loss: 0.0331, Train Acc: 99.00%, Test Loss: 5.4150, Test Acc: 59.07%
Epoch 47/50, Train Loss: 0.0353, Train Acc: 98.93%, Test Loss: 5.1152, Test Acc: 60.85%
Epoch 48/50, Train Loss: 0.0314, Train Acc: 99.04%, Test Loss: 5.6867, Test Acc: 59.78%
Epoch 49/50, Train Loss: 0.0336, Train Acc: 99.00%, Test Loss: 5.6140, Test Acc: 59.78%
Epoch 50/50, Train Loss: 0.0385, Train Acc: 98.87%, Test Loss: 5.6719, Test Acc: 60.49%
Training time: 48.0m 18.46s
```

최종 구현된 모델의
50 에포크 기준 Best ACC 61.60%

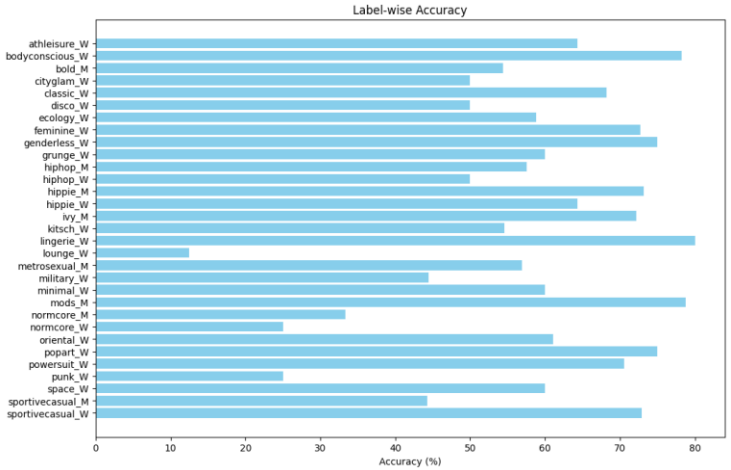




Mission 1-2. 성별 & 스타일 분류 모델 구현 (14/14)

◆ Resnet-18 최종 모델 테스트

- 모델이 예측한 Validation 이미지의 각 라벨별 정확도에선 Lounge_W가 가장 낮고 Lingerie_W가 가장 높다.
- 두 클래스 모두 Training Set에서 증강 전 45, 55개로 매우 적었는데 상반된 정확도 결과가 나왔다.
- 저장된 모델을 로드하여 추론을 실제로 이미지를 보며 테스트 해 본 결과 정확도는 사람의 판단보다 좋은 것 같다.




```
loaded_model = models.resnet18(weights=None)
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, len(label_encoder.classes_))
loaded_model.to(device)
loaded_model.load_state_dict(torch.load('fashion_resnet18.pt', map_location=device, weights_only=False))
```

Mission 2-1. 패션 스타일 선호 여부 예측 통계표 (1/2)

- labeling 데이터의 파일명을 "_" 를 기준으로 구분하기(이때 filename의 원본은 남겨야 함)
- 미션에 주어진 조건에 맞게 이미지 데이터에 존재하는 image_id만을 식별해 유효한 라벨링 데이터만을 남김.
- 유효한 라벨링 데이터만 대상으로 "성별/ 스타일" 통계를 냄.

[STEP 1]

```
[['W_46933_19_genderless_W_167828.json',
  'W',
  '46933',
  '19',
  'genderless',
  'W',
  '167828.json'],
 ['W_30744_19_normcore_M_020980.json',
  'W',
  '30744',
  '19',
  'normcore',
  'M',
  '020980.json'],
 ['W_51820_50_ivy_M_052834.json',
  'W',
  '51820',
  '50',
  'ivy',
  'M',
  '052834.json'],
```

{filename}_ {W/T}_{이미지ID}_{시대별}_{스타일별}_{성별}_{설문ID}

제시된 통계표 형식

성별	스타일	이미지 수
여성	feminine	
	classic	
	minimal	
	popart	
	…	
남성	ivy	
	mods	
	hippie	
	bold	
	…	

```
# drop_training_label_df에서 'Sex'와 'style' 컬럼을 기준으로 그룹화하고,
# 각 그룹에 포함된 'file_name'의 개수를 세어 데이터프레임으로 반환
# 반환된 데이터프레임의 컬럼 이름을 'count'로 지정
drop_training_label_df.groupby(['Sex', 'style'])['file_name'].count().to_frame(name='count')
```

{성별}_{스타일}_{개수}

[출력 예시]

		count
Sex	style	
M	bold	1144
	hiphop	1321
	hippie	1571
	ivy	1719
	metrosexual	1226
	mods	1615
	normcore	1096
	sportivecasual	1657
W	athleisure	424
	bodyconscious	500
	cityglam	329
	classic	504
	disco	141
	ecology	295

Mission 2-1. 패션 스타일 선호 여부 예측 통계표 (2/2)

최종 '성별 & 스타일' 통계표 (라벨링 파일)

	file_name	W/T	image_id	period	style	Sex	survey_id
0	W_32580_90_hiphop_M_081759.json	W	32580	90	hiphop	M	081759
1	W_08785_90_lingerie_W_225338.json	W	08785	90	lingerie	W	225338
2	W_10647_00_ecology_W_123723.json	W	10647	00	ecology	W	123723
3	W_73918_70_hippie_W_019532.json	W	73918	70	hippie	W	019532
4	W_16003_50_ivy_M_014088.json	W	16003	50	ivy	M	014088
...	...	...	...	...	...	...	...
211340	W_67037_19_lounge_W_085333.json	W	67037	19	lounge	W	085333
211341	W_47201_10_sportivecasual_W_033171.json	W	47201	10	sportivecasual	W	033171
211342	T_00366_60_minimal_W_108370.json	T	00366	60	minimal	W	108370
211343	W_48652_60_minimal_W_087178.json	W	48652	60	minimal	W	087178
211344	W_02157_19_normcore_W_115928.json	W	02157	19	normcore	W	115928

211345 rows × columns

	file_name	W/T	image_id	period	style	Sex	survey_id
12	W_03412_50_classic_W_023425.json	W	03412	50	classic	W	023425
19	W_26584_50_ivy_M_071884.json	W	26584	50	ivy	M	071884
22	W_12813_50_ivy_M_044035.json	W	12813	50	ivy	M	044035
25	T_15153_19_normcore_W_227709.json	T	15153	19	normcore	W	227709
42	W_02024_50_classic_W_154035.json	W	02024	50	classic	W	154035
...	...	...	...	...	...	...	...
211313	W_07245_19_normcore_M_110480.json	W	07245	19	normcore	M	110480
211314	T_16092_10_sportivecasual_M_235615.json	T	16092	10	sportivecasual	M	235615
211327	W_12527_70_hippie_M_156484.json	W	12527	70	hippie	M	156484
211333	W_55047_00_oriental_W_200539.json	W	55047	00	oriental	W	200539
211342	T_00366_60_minimal_W_108370.json	T	00366	60	minimal	W	108370

20303 rows × 7 columns

이미지 데이터에 존재하는 image_id만을 식별한 결과 원본 데이터에 비해 유효한 데이터의 수가 줄어든 것을 확인할 수 있음

[STEP 2]

```
drop_training_label_df = training_label_df[training_label_df['image_id'].isin(training_df['image_id'])]
```

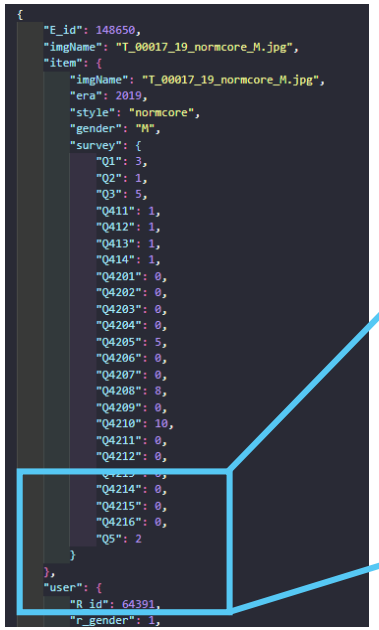
[STEP 3]

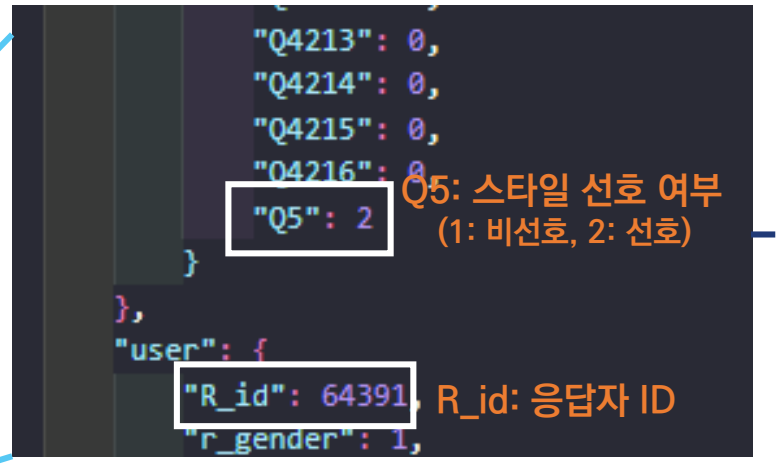
Sex	style	count
M	bold	1144
	hiphop	1321
	hippie	1571
	ivy	1719
	metrosexual	1226
	mods	1615
	normcore	1096
	sportivecasual	1657
W	athleisure	424
	bodyconscious	500
	cityglam	329
	classic	504
	disco	141
	ecology	295
	feminine	762
	genderless	234
	grunge	122
	hiphop	243
	hippie	339
	kitsch	374
	lingerie	188
	lounge	207
	military	115
	minimal	780
	normcore	646
	oriental	369
	popart	195
	powersuit	623
	punk	237
	space	184
	sportivecasual	1143

* training_label에 대한 것만 ppt에 내용을 첨부했는데 validation_label도 과정이 동일함. 코드 참고

Mission 2-2. 패션 스타일 선호 여부 예측 데이터 (1/5)

- Image, Label 둘 다 존재하는 데이터만 분리한 유효 라벨링 데이터의 JSON파일에서 'Q5'(선호/비선호)와 'R_id'(응답자ID)를 추출하여 table에 추가.
- 각각의 라벨에 맞는 jpg 파일 명 더한 후, 응답자ID 수 확인.
- 응답자ID('R_id')를 기준으로 jpg파일 명('jpg_name') 값들을 리스트 형태로 저장 후, Training과 Validation 구분하여 선호/비선호 table 작성.
- 응답 수 상위 100명을 선택하고, Pandas Dataframe을 HTML 형식으로 변환 및 스크롤 스타일 추가.





JSON 파일 읽는 함수 정의

```python
def extract_json_data(file_name):
    with open(file_name, 'r') as f:
        data = json.load(f)

    # 필요한 데이터 추출
    R_id = data['user']['R_id']
    Q5 = data['item']['survey']['Q5']

    return R_id, Q5
```

Mission 2-2. 패션 스타일 선호 여부 예측 데이터 (2/5)

- ‘Q5’, ‘R_id’, ‘jpg_name’ column 추가
  - 유효 라벨링 데이터 24565 개 (train+validation)

	file_name	W/T	image_id	period	style	Sex	survey_id	T/V	R_id	Q5	jpg_name
0	W_03412_50_classic_W_023425.json	W	03412	50	classic	W	023425	Train	18670	1	W_03412_50_classic_W.jpg
1	W_26584_50_ivy_M_071884.json	W	26584	50	ivy	M	071884	Train	52208	2	W_26584_50_ivy_M.jpg
2	W_12813_50_ivy_M_044035.json	W	12813	50	ivy	M	044035	Train	27584	1	W_12813_50_ivy_M.jpg
3	T_15153_19_normcore_W_227709.json	T	15153	19	normcore	W	227709	Train	67689	2	T_15153_19_normcore_W.jpg
4	W_02024_50_classic_W_154035.json	W	02024	50	classic	W	154035	Train	64598	2	W_02024_50_classic_W.jpg
...	...	...	...	...	...	...	...	...	...	...	...
24560	T_12567_80_powersuit_W_102030.json	T	12567	80	powersuit	W	102030	Validation	61503	2	T_12567_80_powersuit_W.jpg
24561	W_04207_50_ivy_M_028693.json	W	04207	50	ivy	M	028693	Validation	20458	2	W_04207_50_ivy_M.jpg
24562	W_33002_60_mods_M_039676.json	W	33002	60	mods	M	039676	Validation	25042	1	W_33002_60_mods_M.jpg
24563	W_01463_60_mods_M_078848.json	W	01463	60	mods	M	078848	Validation	57952	2	W_01463_60_mods_M.jpg
24564	W_00842_10_sportivecasual_M_092976.json	W	00842	10	sportivecasual	M	092976	Validation	60234	2	W_00842_10_sportivecasual_M.jpg

24565 rows × 11 columns

- 응답자 ID 수 확인 (4075개)

```
drop_df['R_id'].nunique()
```

```
4075
```

```
drop_df.groupby('R_id')['jpg_name'].apply(list)
```

```
R_id
12
25
26
27       [W_15212_60_mods_M.jpg, W_17957_80_bold_M.jpg
30

68961
68966
68969
68983
69016
Name: jpg_name, Length: 4075, dtype: object
```

응답자 ID 기준 groupby 실행
➔ 'R_id'에 대해 'jpg_name ' 값 리스트 형태로 저장 (Length: 4075)

Mission 2-2. 패션 스타일 선호 여부 예측 데이터 (3/5)

R_id	Training 스타일 선호	Training 스타일 비선호	Validation 스타일 선호	Validation 스타일 비선호
12	[W_03412_50_classic_W.jpg]	NaN	[W_03412_50_classic_W.jpg]	[W_02651_50_feminine_W.jpg]
25	[T_01475_10_sportivecasual_M.jpg, T_14304_60_mods_M.jpg, T_16423_10_sportivecasual_M.jpg]	[W_12740_00_metrosexual_M.jpg]	NaN	NaN
26	[T_09447_19_lounge_W.jpg]	[W_18990_50_feminine_W.jpg]	NaN	NaN
27	[W_17957_80_bold_M.jpg, W_10810_60_mods_M.jpg, W_17260_19_normcore_M.jpg, W_15268_50_ivy_M.jpg, W_03007_70_hippie_M.jpg, W_16068_80_bold_M.jpg]	[W_15212_60_mods_M.jpg, W_07347_90_hiphop_M.jpg, W_06563_70_hippie_M.jpg, W_10814_50_ivy_M.jpg]	[W_06522_50_ivy_M.jpg, W_07120_19_normcore_M.jpg, W_17697_50_ivy_M.jpg]	NaN
30	[W_18249_50_feminine_W.jpg, W_63984_80_powersuit_W.jpg]	[W_49287_70_punk_W.jpg]	[W_18249_50_feminine_W.jpg]	NaN
...	...	...	...	...
68961	NaN	[T_19177_19_normcore_M.jpg]	NaN	NaN
68966	[T_19158_19_normcore_M.jpg]	NaN	NaN	NaN
68969	[T_16428_10_sportivecasual_M.jpg, T_16439_10_sportivecasual_M.jpg, T_16422_10_sportivecasual_M.jpg, T_16427_10_sportivecasual_M.jpg]	[T_16423_10_sportivecasual_M.jpg, T_16432_10_sportivecasual_M.jpg, T_16414_10_sportivecasual_M.jpg, T_16436_10_sportivecasual_M.jpg]	NaN	NaN
68983	[T_19158_19_normcore_M.jpg]	[T_16151_10_sportivecasual_M.jpg, T_16153_10_sportivecasual_M.jpg, T_19177_19_normcore_M.jpg, T_16136_10_sportivecasual_M.jpg, T_16157_10_sportivecasual_M.jpg]	NaN	NaN
69016	[T_16332_10_sportivecasual_M.jpg]	[T_16570_10_sportivecasual_M.jpg, T_16341_10_sportivecasual_M.jpg, T_16552_10_sportivecasual_M.jpg, T_16323_10_sportivecasual_M.jpg]	NaN	NaN

4075 rows × 4 columns

```
# Total Count 제외하고 상위 100개 선택
top_100_summary_df = summary_df.sort_values(by=('Total', 'Count'), ascending=False).head(100).drop(columns=('Total', 'Count'))
```

Training과 Validation 구분하여 선호/비선호 table 작성
& 상위 100명의 응답자 선택

Mission 2-2. 패션 스타일 선호 여부 예측 데이터 (4/5)

여러 응답이 리스트로 저장된 형태

R_id	Training 스타일 선호
12	[W_03412_50_classic_W.jpg]
25	[T_01475_10_sportivecasual_M.jpg, T_14304_60_mods_M.jpg, T_16423_10_sportivecasual_M.jpg]
26	[T_09447_19_lounge_W.jpg]
27	[W_17957_80_bold_M.jpg, W_10810_60_mods_M.jpg, W_17260_19_normcore_M.jpg, W_15268_50_ivy_M.jpg, W_03007_70_hippie_M.jpg, W_16068_80_bold_M.jpg]
30	[W_18249_50_feminine_W.jpg, W_63984_80_powersuit_W.jpg]
...	...
68961	NaN
68966	[T_19158_19_normcore_M.jpg]
68969	[T_16428_10_sportivecasual_M.jpg, T_16439_10_sportivecasual_M.jpg, T_16422_10_sportivecasual_M.jpg, T_16427_10_sportivecasual_M.jpg]
68983	[T_19158_19_normcore_M.jpg]
69016	[T_16332_10_sportivecasual_M.jpg]

⋮

제시된 '스타일 선호 정보표' 형태

응답자 ID	Training		Validation	
	스타일 선호	스타일 비선호	스타일 선호	스타일 비선호
64747	W_07894_00_cityglam_W.jpg	W_44386_80_powersuit_W.jpg	W_05628_00_cityglam_W.jpg	W_34024_10_sportivecasual_W.jpg
	W_37160_70_punk_W.jpg	W_34573_10_sportivecasual_W.jpg	W_37491_70_military_W.jpg	W_11610_90_grunge_W.jpg
	W_39725_19_normcore_W.jpg	W_40876_70_punk_W.jpg	W_38588_19_genderless_W.jpg	W_47169_70_hippie_W.jpg
	...	...	...	...
...				

➔ 형태를 맞추기 위해 형식 변환 필요

Mission 2-2. 패션 스타일 선호 여부 예측 데이터 (5/5)

```
html_table = top_100_summary_df.to_html(max_rows=100, max_cols=None)
html_table = f'''
<div style="max-height: 500px; overflow-y: auto; border: 1px solid #ccc;">
    {html_table.replace('<table border="1" class="dataframe">', '<table border="1" class="dataframe" style="text-align: center;">')}
</div>
'''
```

Pandas Dataframe → HTML 변환
(HTML을 보기 좋게 하기 위해 스크롤 스타일 추가)

	Training		Validation	
	스타일 선호	스타일 비선호	스타일 선호	스타일 비선호
R_id				
64747	[W_21483_19_genderless_W.jpg, W_30399_19_genderless_W.jpg, W_04972_90_kitsch_W.jpg, W_38421_10_athleisure_W.jpg, W_07894_00_cityglam_W.jpg, W_46907_80_powersuit_W.jpg, W_39725_19_normcore_W.jpg, W_29783_10_sportivecasual_W.jpg, W_37160_70_punk_W.jpg, W_31416_70_hippie_W.jpg, W_48628_19_genderless_W.jpg, W_30434_60_minimal_W.jpg, W_38629_80_powersuit_W.jpg, W_30454_60_minimal_W.jpg, W_35674_60_minimal_W.jpg, W_03194_50_classic_W.jpg, W_22057_19_genderless_W.jpg, W_05628_00_cityglam_W.jpg, W_34636_00_oriental_W.jpg, W_20598_70_military_W.jpg, W_14901_90_kitsch_W.jpg, W_21223_80_powersuit_W.jpg, W_44330_10_sportivecasual_W.jpg, W_22510_80_powersuit_W.jpg]	[W_33026_90_hiphop_W.jpg, W_48378_90_grunge_W.jpg, W_18951_50_feminine_W.jpg, W_27828_60_minimal_W.jpg, W_14102_50_feminine_W.jpg, W_47169_70_hippie_W.jpg, W_13904_50_feminine_W.jpg, W_02498_50_feminine_W.jpg, W_41633_60_space_W.jpg, W_40690_80_bodyconscious_W.jpg, W_02247_50_classic_W.jpg, W_42595_60_popart_W.jpg, W_03643_00_cityglam_W.jpg, W_44386_80_powersuit_W.jpg, W_34573_10_sportivecasual_W.jpg, W_37025_19_lounge_W.jpg, W_34024_10_sportivecasual_W.jpg, W_11610_90_grunge_W.jpg, W_36644_00_oriental_W.jpg, W_40876_70_punk_W.jpg, W_38771_10_sportivecasual_W.jpg]	[W_30988_90_kitsch_W.jpg, W_39164_00_oriental_W.jpg, W_46907_80_powersuit_W.jpg, W_37491_70_military_W.jpg, W_38588_19_genderless_W.jpg, W_05628_00_cityglam_W.jpg, W_20598_70_military_W.jpg, W_44330_10_sportivecasual_W.jpg, W_22510_80_powersuit_W.jpg]	[W_27828_60_minimal_W.jpg, W_14102_50_feminine_W.jpg, W_47169_70_hippie_W.jpg, W_02498_50_feminine_W.jpg, W_34024_10_sportivecasual_W.jpg, W_11610_90_grunge_W.jpg]
64561	[W_36907_19_genderless_W.jpg, W_36601_19_normcore_W.jpg, W_31157_00_cityglam_W.jpg, W_20369_90_lingerie_W.jpg, W_18759_50_feminine_W.jpg, W_22239_60_space_W.jpg, W_36584_90_kitsch_W.jpg, W_49853_00_oriental_W.jpg, W_43737_80_powersuit_W.jpg, W_34974_60_minimal_W.jpg, W_30671_70_hippie_W.jpg, W_41732_00_cityglam_W.jpg, W_37270_00_cityglam_W.jpg, W_35091_80_powersuit_W.jpg, W_18205_50_feminine_W.jpg, W_34340_80_bodyconscious_W.jpg, W_18066_50_classic_W.jpg, W_06046_10_sportivecasual_W.jpg, W_42314_70_hippie_W.jpg, W_38656_10_sportivecasual_W.jpg, W_45035_10_sportivecasual_W.jpg, W_28013_19_normcore_W.jpg]	[W_19238_90_grunge_W.jpg, W_42868_10_sportivecasual_W.jpg, W_23519_60_minimal_W.jpg, W_48741_00_cityglam_W.jpg, W_41177_10_sportivecasual_W.jpg, W_08584_50_feminine_W.jpg, W_46417_70_military_W.jpg, W_30044_00_ecology_W.jpg, W_20944_80_powersuit_W.jpg, W_32248_80_powersuit_W.jpg, W_26910_19_normcore_W.jpg, W_49287_70_punk_W.jpg, W_13444_50_classic_W.jpg, W_32870_90_hiphop_W.jpg, W_34989_90_hiphop_W.jpg, W_38012_19_normcore_W.jpg, W_32939_70_military_W.jpg, W_27343_90_lingerie_W.jpg, W_22943_10_athleisure_W.jpg, W_41279_19_genderless_W.jpg, W_26946_19_lounge_W.jpg, W_12032_50_feminine_W.jpg, W_33622_60_minimal_W.jpg, W_02232_70_hippie_W.jpg]	[W_22239_60_space_W.jpg, W_33305_60_space_W.jpg, W_30671_70_hippie_W.jpg, W_35091_80_powersuit_W.jpg, W_18205_50_feminine_W.jpg, W_41448_10_sportivecasual_W.jpg, W_06046_10_sportivecasual_W.jpg, W_38656_10_sportivecasual_W.jpg]	[W_23519_60_minimal_W.jpg, W_33240_80_bodyconscious_W.jpg, W_22943_10_athleisure_W.jpg, W_48457_60_minimal_W.jpg]
	[W_16084_80_bold_M.jpg, W_02931_00_metrosexual_M.jpg, W_00555_50_ivy_M.jpg, W_06812_50_ivy_M.jpg, W_02890_19_normcore_M.jpg, W_04684_90_hiphop_M.jpg,	[W_17454_80_bold_M.jpg, W_17219_70_hippie_M.jpg, W_16501_70_hippie_M.jpg, W_15472_70_hippie_M.jpg, W_15782_70_hippie_M.jpg, W_12383_80_bold_M.jpg, W_12904_50_ivy_M.jpg, W_15471_10_sportivecasual_M.jpg,		

최종 '스타일 선호 정보표' 형태

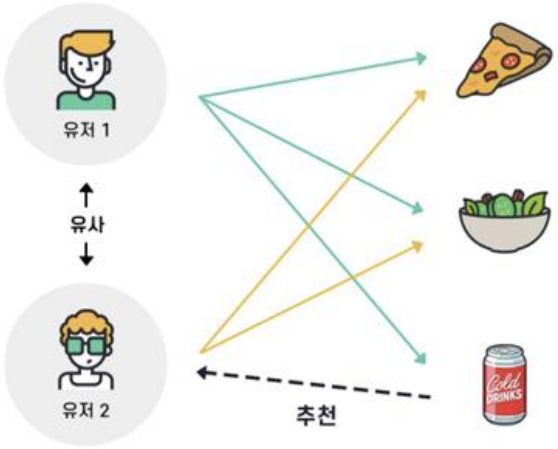
Mission 3-1. 패션 스타일 선호 여부 예측 방안 (1/6)

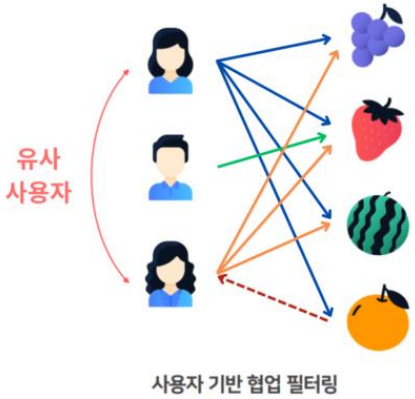
- **협업 필터링이란?**

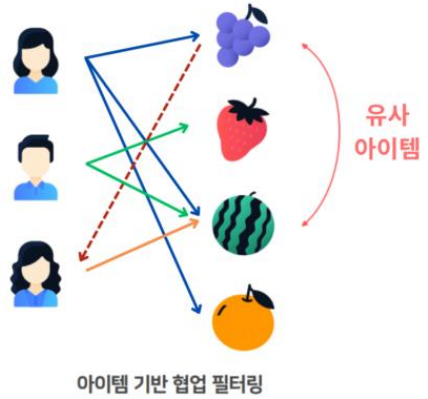
→ 사용자와 아이템 간의 상호작용 데이터를 기반으로 유사한 사용자나 아이템을 찾아 새로운 추천을 생성하는 방법으로 사용자의 과거 행동 데이터나 유사한 사용자 그룹의 데이터를 활용해 개인화된 추천을 제공하는 것

- **협업 필터링의 종류**

→ User-based Filtering (사용자 기반 협업 필터링)
→ Item-based Filtering (아이템 기반 협업 필터링)







Mission 3-1. 패션 스타일 선호 여부 예측 방안 (2/6)

◆ User-based Filtering

- 사용자의 취향이 비슷한 다른 사용자의 선호도를 바탕으로 추천을 진행
- 응답자 간의 유사성을 계산하고, 유사한 응답자가 선호하는 스타일을 현재 응답자에게 추천하거나 선호 여부를 예측하는 방식

- 유사도를 계산하는 방법
  - ❖ 방법은 각 응답자에 대한 선호 스타일 정보를 벡터화 한 뒤 다른 응답자와의 유사도를 측정
  - ❖ 사용자 간의 유사성 평가는 row 기준

장점
- 개인화된 추천을 제공하기 때문에 유사한 응답자들이 선호하는 스타일을 쉽게 예측 가능
- Validation 데이터에 포함된 새로운 스타일이라도 유사한 응답자의 선호를 토대로 예측이 가능
- 추천의 이유를 명확하게 설명할 수 있음

단점
- 사용자가 많아질 수록 유사도 계산 비용이 증가
- 사용자가 선호하는 것이 적은 경우 유사도 계산이 어렵거나 잘못된 결과가 도출

Mission 3-1. 패션 스타일 선호 여부 예측 방안 (3/6)

◆ Item-based Filtering

- 스타일 간의 유사도를 계산하여 추천을 진행하는 방식
- 응답자가 특정 스타일을 선호한다면, 해당 스타일과 유사한 스타일을 선호할 가능성이 높다고 예측

- 유사도를 계산하는 방법
  - 방법은 각 응답자에 대한 선호 스타일 정보를 벡터화 한 뒤 다른 응답자와의 유사도를 측정
  - 이 과정에서 ResNet-18의 중간 레이어에서 추출한 특징 벡터를 이용하여 각 스타일 간의 코사인 유사도를 계산
  - 사용자 간의 유사성 평가는 column 기준

장점
- 스타일 자체를 기반으로 추천을 진행하기 때문에, 응답자의 수가 증가해도 계산 효율이 높음
- 스타일에 대한 취향이 분명한 경우 높은 정확도를 가짐

단점
- 개별 응답자의 특성을 반영하기는 어려워 개인화된 추천에는 한계가 존재
- 이웃 사용자의 정보가 익명으로 처리되기 때문에 설명하기가 어려움

Mission 3-1. 패션 스타일 선호 여부 예측 방안 (4/6)

[예시]

	Feminine	Punk	Sportive Casual	Classic
응답자 1	O	O	X	?
응답자 2	O	X	?	O
응답자 3	X	O	O	?

Classic

* 0: 스타일 선호 / X: 스타일 비선호를 의미함

- User – based filtering

➔ 응답자1과 2는 “feminine” 스타일을 모두 선호하고 있으므로 이 둘의 유사도는 높게 나올 것
➔ 응답자1과 3은 선호하는 스타일이 다르므로 유사도가 낮게 나올 것
➔ 추가적으로 새로운 이미지가 들어왔을 때 “classic”스타일에 속한다면 응답자1에 대한 선호 정보가 없더라도 “classic”을 선호하는 응답자2와의 유사도를 바탕으로 예측 가능

Mission 3-1. 패션 스타일 선호 여부 예측 방안 (5/6)

[예시]

둘 사이의 유사도 파악

	Feminine	Punk	Sportive Casual	Classic
응답자 1	O	O	X	?
응답자 2	O	X	?	O
응답자 3	X	O	O	?

Classic

* 0: 스타일 선호 / X: 스타일 비선호를 의미함

- Item – based filtering

➔ feminine과 classic의 코사인 유사도를 계산하여 스타일 간 유사도를 파악
➔ 유사도가 높게 나온다면 새로운 들어온 이미지가 "classic" 스타일에 속할 때, 이를 선호할 가능성이 높다고 예측

Mission 3-1. 패션 스타일 선호 여부 예측 방안 (6/6)

User-based filtering	Item-based filtering
• 개별 응답자의 선호도를 더 잘 반영할 수 있어 개인화에 유리 • 데이터가 희소할 때 예측 성능이 떨어질 수 있음 • 다른 사용자의 평가를 바탕으로 하기 때문에 예측 성능이 떨어질 수 있음 ➔ 사용자의 개인적인 취향을 더 잘 반영 하고자 하는 경우에 유리	• 응답자 수가 증가해도 효율성이 유지 • 사용자 자신의 평가 데이터를 활용하기 때문에 스타일 예측이 명확함 • 사용자 개인의 특정한 취향보다는 일반적인 아이템 유사성에 의존하기 때문에 개인화된 추천에는 한계가 있을 수 있음 ➔ 새로운 사용자가 많은 시스템이나 사용자 정보가 부족한 경우에 유리

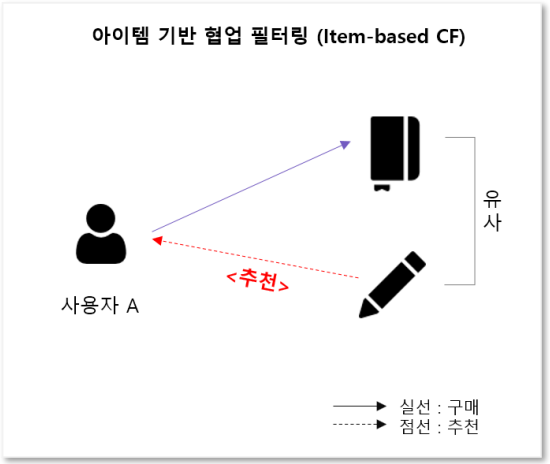
Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (1/8)

◆ 데이터 로드 및 전처리

- (문제) Validation 데이터 내 응답자의 “스타일 선호 여부 예측” 문제를 수행하고 성능을 측정한다.
- 필요한 데이터는 다음과 같다.
- feature_df 데이터프레임으로 Train, Validation 데이터셋 Image들의 특징 벡터를 기록한다.
- user_data에는 각 사용자별 선호 및 비선호 이미지 정보를 담아둔다. (2-2에서 구현한 데이터)
- 위 데이터를 바탕으로, Validation 데이터가 들어왔을때 해당 이미지가 선호/ 비선호 어디에 속할지 Item-Based 기반 협업 필터링 추천 시스템을 구현한다.

	Training		Validation	
	스타일 선호	스타일 비선호	스타일 선호	스타일 비선호
R_id				
64747	[W_21483_19_genderless_W.jpg, W_30399_19_genderless_W.jpg, W_04972_90_kitsch_W.jpg, W_38421_10_athleisure_W.jpg, W_07894_00_cityglam_W.jpg, W_46907_80_powersuit_W.jpg, W_39725_19_normcore_W.jpg, W_29783_10_sportivecasual_W.jpg, W_37160_70_punk_W.jpg, W_31416_70_hippie_W.jpg, W_48628_19_genderless_W.jpg, W_30434_60_minimal_W.jpg, W_38629_80_powersuit_W.jpg, W_30454_60_minimal_W.jpg, W_35674_60_minimal_W.jpg, W_03194_50_classic_W.jpg, W_22057_19_genderless_W.jpg, W_05628_00_cityglam_W.jpg, W_34636_00_oriental_W.jpg, W_20598_70_military_W.jpg, W_14901_90_kitsch_W.jpg, W_21223_80_powersuit_W.jpg, W_44330_10_sportivecasual_W.jpg, W_22510_80_powersuit_W.jpg]	[W_33026_90_hiphop_W.jpg, W_48378_90_grunge_W.jpg, W_18951_50_feminine_W.jpg, W_27828_60_minimal_W.jpg, W_14102_50_feminine_W.jpg, W_47169_70_hippie_W.jpg, W_13904_50_feminine_W.jpg, W_02498_50_feminine_W.jpg, W_41633_60_space_W.jpg, W_40690_80_bodyconscious_W.jpg, W_02247_50_classic_W.jpg, W_42595_60_popart_W.jpg, W_03643_00_cityglam_W.jpg, W_44386_80_powersuit_W.jpg, W_34573_10_sportivecasual_W.jpg, W_37025_19_lounge_W.jpg, W_34024_10_sportivecasual_W.jpg, W_11610_90_grunge_W.jpg, W_36644_00_oriental_W.jpg, W_40876_70_punk_W.jpg, W_38771_10_sportivecasual_W.jpg]	[W_30988_90_kitsch_W.jpg, W_39164_00_oriental_W.jpg, W_46907_80_powersuit_W.jpg, W_37491_70_military_W.jpg, W_38588_19_genderless_W.jpg, W_05628_00_cityglam_W.jpg, W_20598_70_military_W.jpg, W_44330_10_sportivecasual_W.jpg, W_22510_80_powersuit_W.jpg]	[W_27828_60_minimal_W.jpg, W_14102_50_feminine_W.jpg, W_47169_70_hippie_W.jpg, W_02498_50_feminine_W.jpg, W_34024_10_sportivecasual_W.jpg, W_11610_90_grunge_W.jpg]
64561	[W_36907_19_genderless_W.jpg, W_36601_19_normcore_W.jpg, W_31157_00_cityglam_W.jpg, W_20369_90_lingerie_W.jpg, W_18759_50_feminine_W.jpg, W_22239_60_space_W.jpg, W_36584_90_kitsch_W.jpg, W_49853_00_oriental_W.jpg, W_43737_80_powersuit_W.jpg, W_34974_60_minimal_W.jpg, W_30671_70_hippie_W.jpg, W_41732_00_cityglam_W.jpg, W_37270_00_cityglam_W.jpg, W_35091_80_powersuit_W.jpg, W_18205_50_feminine_W.jpg, W_34340_80_bodyconscious_W.jpg, W_18066_50_classic_W.jpg, W_06046_10_sportivecasual_W.jpg, W_42314_70_hippie_W.jpg, W_38656_10_sportivecasual_W.jpg, W_45035_10_sportivecasual_W.jpg, W_28013_19_normcore_W.jpg]	[W_19238_90_grunge_W.jpg, W_42868_10_sportivecasual_W.jpg, W_23519_60_minimal_W.jpg, W_48741_00_cityglam_W.jpg, W_41177_10_sportivecasual_W.jpg, W_08584_50_feminine_W.jpg, W_46417_70_military_W.jpg, W_30044_00_ecology_W.jpg, W_20944_80_powersuit_W.jpg, W_32248_80_powersuit_W.jpg, W_26910_19_normcore_W.jpg, W_49287_70_punk_W.jpg, W_13444_50_classic_W.jpg, W_32870_90_hiphop_W.jpg, W_34989_90_hiphop_W.jpg, W_38012_19_normcore_W.jpg, W_32939_70_military_W.jpg, W_27343_90_lingerie_W.jpg, W_22943_10_athleisure_W.jpg, W_41279_19_genderless_W.jpg, W_26946_19_lounge_W.jpg, W_12032_50_feminine_W.jpg, W_33622_60_minimal_W.jpg, W_02232_70_hippie_W.jpg]	[W_22239_60_space_W.jpg, W_33305_60_space_W.jpg, W_30671_70_hippie_W.jpg, W_35091_80_powersuit_W.jpg, W_18205_50_feminine_W.jpg, W_41448_10_sportivecasual_W.jpg, W_06046_10_sportivecasual_W.jpg, W_38656_10_sportivecasual_W.jpg]	[W_23519_60_minimal_W.jpg, W_33240_80_bodyconscious_W.jpg, W_22943_10_athleisure_W.jpg, W_48457_60_minimal_W.jpg]
	[W_16084_80_bold_M.jpg, W_02931_00_metrosexual_M.jpg, W_00555_50_ivy_M.jpg, W_06812_50_ivy_M.jpg, W_02890_19_normcore_M.jpg, W_04684_90_hiphop_M.jpg, W_01670_10_sportivecasual_M.jpg, W_04532_50_ivy_M.jpg, W_12814_10_normcore_M.jpg]	[W_17454_80_bold_M.jpg, W_17219_70_hippie_M.jpg, W_16501_70_hippie_M.jpg, W_15472_70_hippie_M.jpg, W_15782_70_hippie_M.jpg, W_12383_80_bold_M.jpg, W_12904_50_ivy_M.jpg, W_15471_10_sportivecasual_M.jpg]	[W_03677_60_mods_M.jpg, W_04684_90_hiphop_M.jpg]	[W_16501_70_hippie_M.jpg, W_12904_50_ivy_M.jpg]

2-2에서 구한 패션 스타일 선호 여부 데이터



Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (2/8)

◆ 이미지 특징 추출 및 유사도 분석

- 1-2 미션에서 만들었던 패션 분류 Resnet-18의 가중치를 불러와 Feature Extractor를 구현함.
- 구현한 Resnet의 마지막 FC레이어를 Identity로 변경해 Conv층의 결과인 512 특징 벡터만을 꺼냄.
- 해당 벡터에는 각 이미지 패션의 특징이 표현되어있는데, 이를 통해 각 이미지의 유사도를 구할 수 있음.

	0	1	2	3	4	5	6	7	8	9	...	502	503	504	505
W_01752_00_metrosexual_M	1.847668	0.000000	2.458406	0.270229	1.170993	3.836130	0.000000	1.158465	1.753551	4.316210	...	0.000000	1.699600	0.000000	2.604443
W_46417_70_military_W	2.046788	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.133544	0.027279	8.868113	...	0.000000	6.317285	0.000000	0.925806
W_01509_00_metrosexual_M	1.437613	1.770455	2.123551	0.151370	0.000000	1.460724	0.000000	0.000000	2.396101	1.030263	...	6.129653	4.321615	0.000000	0.142883
W_18951_50_feminine_W	0.000000	0.000000	0.663192	0.000000	0.000000	1.108396	0.000000	0.065373	2.407773	1.175754	...	3.787046	0.117699	0.000000	0.000000
W_29485_10_sportivecasual_M	1.839155	0.000000	0.099437	1.552018	0.000000	0.000000	0.000000	1.252575	5.868081	5.809547	...	2.109154	12.650900	0.000000	0.802910
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
T_21988_70_hippie_M	0.360615	0.000000	1.936827	0.657066	0.000000	1.639859	0.000000	0.000000	6.749860	3.557946	...	4.524527	0.000000	2.997836	0.000000
W_28022_50_ivy_M	0.078146	1.574545	0.000000	0.000000	0.992199	7.407101	0.000000	0.287537	0.812477	0.781694	...	0.000000	0.326500	0.000000	0.000000
W_00551_19_normcore_M	0.000000	0.000000	2.552800	5.708521	0.000000	3.025286	1.091456	0.000000	13.621263	1.081994	...	13.015729	3.710604	0.000000	0.000000
W_03144_50_classic_W	0.000000	0.000000	27.429203	9.720285	13.629567	6.878621	0.000000	1.309680	0.000000	1.095730	...	3.294993	1.892139	6.258384	0.087237
W_28453_10_sportivecasual_M	0.000000	1.704645	0.000000	0.000000	0.000000	1.170660	0.000000	0.535658	0.394556	4.417046	...	0.000000	0.000000	0.000000	3.063981

5021 rows × 512 columns

모든 이미지를 Resnet-18기반의 Feature Extractor에 넣어
특징을 출력한 Feature_df
(5021 x 512)

	W_01752_00_metrosexual_M	W_46417_70_military_W
W_01752_00_metrosexual_M	1.000000	0.333978
W_46417_70_military_W	0.333978	1.000000
W_01509_00_metrosexual_M	0.469022	0.280720
W_18951_50_feminine_W	0.267896	0.215525
W_29485_10_sportivecasual_M	0.379189	0.269625
...	...	...
T_21988_70_hippie_M	0.409494	0.232425
W_28022_50_ivy_M	0.344177	0.174615
W_00551_19_normcore_M	0.407112	0.213695
W_03144_50_classic_W	0.376030	0.199068
W_28453_10_sportivecasual_M	0.298720	0.316381

5021 rows × 5021 columns

각 이미지들의 Cosine 유사도를 구해서
계산해본 similarity_df
(5021 x 5021)

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (3/8)

◆ Resnet-18 특징벡터 기반 추천 시스템 구현

```python
# 만약 이미지에 해당하는 특징 벡터가 없을 경우 사용할 기본 특징 벡터 (중앙값)
DUMMY_FEATURES = feature_df.median().values.reshape(1, -1)
len(DUMMY_FEATURES)

# 사용자별 선호도 예측을 위한 딕셔너리 초기화
predictions = {}
for user_id, like_images, dislike_images, val_like_images, val_dislike_images in zip(
        user_data["R_id"], user_data["Training_Like"], user_data["Training_Dislike"],
        user_data["Validation_Like"], user_data["Validation_Dislike"]):

    # 각 사용자의 선호 및 비선호 이미지들의 특징 벡터 추출
    # 만약 선호나 비선호 이미지가 feature_df에 없다면 유효한 이미지 목록으로 필터링
    valid_like_images = [img for img in like_images if img in feature_df.index]
    valid_dislike_images = [img for img in dislike_images if img in feature_df.index]

    # 유효한 선호 이미지가 없으면 더미 특징 벡터 사용, 있으면 해당 특징 벡터 사용
    if not valid_like_images:
        like_features = DUMMY_FEATURES
    else:
        like_features = feature_df.loc[valid_like_images].values

    # 유효한 비선호 이미지가 없으면 더미 특징 벡터 사용, 있으면 해당 특징 벡터 사용
    if not valid_dislike_images:
        dislike_features = DUMMY_FEATURES
    else:
        dislike_features = feature_df.loc[valid_dislike_images].values

    # Validation의 선호 및 비선호 이미지에 대해 평가 수행
    for val_image, label in zip(val_like_images + val_dislike_images, [1] * len(val_like_images) + [0] * len(val_dislike_images)):

        # Validation 이미지의 특징 벡터 가져오기 (feature_df에서 존재하지 않는 경우 더미 특징 벡터 사용)
        if val_image in feature_df.index:
            try:
                val_feature = feature_df.loc[val_image].median().values.reshape(1, -1)
            except:
                val_feature = feature_df.loc[val_image].values.reshape(1, -1)
        else:
            val_feature = DUMMY_FEATURES

        # 코사인 유사도를 통해 선호 유사도 및 비선호 유사도 계산
        like_similarity = cosine_similarity(val_feature, like_features).mean()
        dislike_similarity = cosine_similarity(val_feature, dislike_features).mean()

        # 유사도 출력
        print(f"User {user_id} - Image {val_image}: 선호 유사도 - {like_similarity:.4f}, 비선호 유사도 - {dislike_similarity:.4f}")
```

- User Data에서 각 사용자의 Training_Like(선호 이미지)와 Training_Dislike(비선호 이미지)에 해당하는 이미지들의 특징 벡터를 추출하고, 코사인 유사도를 계산해 선호와 비선호 이미지 간의 유사도를 비교
- Validation 데이터(val_like_images, val_dislike_images)를 사용하여 선호 이미지인지 여부를 예측하며, like_similarity와 dislike_similarity의 차이를 계산하여 예측의 기준으로 삼음.
- 유사도 분석 중 사용자가 선호, 비선호 여부를 선택하지 않은 빈 이미지가 있는 경우, Feature_DF의 중앙값 벡터를 사용해 기본값을 정해 계속 연산할 수 있도록 함.
- 총 4262개의 Validation Image 아이템을 검증

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (4/8)

◆ Resnet-18 특징벡터 기반 추천 시스템 구현 결과

```
User 12 - Image W_03412_50_classic_W: 선호 유사도 - 1.0000, 비선호 유사도 - 0.4004
User 12 - Image W_02651_50_feminine_W: 선호 유사도 - 0.4011, 비선호 유사도 - 0.4429
User 27 - Image W_06522_50_ivy_M: 선호 유사도 - 0.3213, 비선호 유사도 - 0.3188
User 27 - Image W_07120_19_normcore_M: 선호 유사도 - 0.3292, 비선호 유사도 - 0.3412
User 27 - Image W_17697_50_ivy_M: 선호 유사도 - 0.3404, 비선호 유사도 - 0.3447
User 30 - Image W_18249_50_feminine_W: 선호 유사도 - 0.7835, 비선호 유사도 - 0.3391
User 133 - Image W_10073_70_hippie_M: 선호 유사도 - 0.2675, 비선호 유사도 - 0.4467
User 133 - Image W_17697_50_ivy_M: 선호 유사도 - 0.3553, 비선호 유사도 - 0.3590
User 140 - Image W_07121_80_bold_M: 선호 유사도 - 0.5408, 비선호 유사도 - 0.3283
User 179 - Image W_14147_70_disco_W: 선호 유사도 - 0.2829, 비선호 유사도 - 0.6993
User 196 - Image W_01549_50_ivy_M: 선호 유사도 - 0.5407, 비선호 유사도 - 0.3509
User 196 - Image W_12803_70_hippie_M: 선호 유사도 - 0.5085, 비선호 유사도 - 0.2723
User 289 - Image W_06448_10_sportivecasual_W: 선호 유사도 - 0.8020, 비선호 유사도 - 0.3564
User 289 - Image W_11495_19_genderless_W: 선호 유사도 - 0.2636, 비선호 유사도 - 0.5671
User 368 - Image W_01703_00_metrosexual_M: 선호 유사도 - 0.4894, 비선호 유사도 - 0.4014
User 368 - Image W_12817_50_ivy_M: 선호 유사도 - 0.4102, 비선호 유사도 - 0.3651
User 368 - Image W_00551_19_normcore_M: 선호 유사도 - 0.4001, 비선호 유사도 - 0.4125
User 368 - Image W_06864_10_sportivecasual_M: 선호 유사도 - 0.3359, 비선호 유사도 - 0.3568
User 368 - Image W_04622_60_mods_M: 선호 유사도 - 0.3872, 비선호 유사도 - 0.2976
User 368 - Image W_04678_50_ivy_M: 선호 유사도 - 0.3975, 비선호 유사도 - 0.3547
User 368 - Image W_15791_70_hippie_M: 선호 유사도 - 0.2856, 비선호 유사도 - 0.3678
User 368 - Image W_15340_50_ivy_M: 선호 유사도 - 0.3646, 비선호 유사도 - 0.4033
User 368 - Image W_06551_60_mods_M: 선호 유사도 - 0.4343, 비선호 유사도 - 0.4343
User 368 - Image W_16848_19_normcore_M: 선호 유사도 - 0.3390, 비선호 유사도 - 0.3564
User 368 - Image W_16034_80_bold_M: 선호 유사도 - 0.3407, 비선호 유사도 - 0.3451
User 525 - Image W_04629_90_hiphop_M: 선호 유사도 - 0.4629, 비선호 유사도 - 0.3603
User 525 - Image W_06889_90_hiphop_M: 선호 유사도 - 0.3381, 비선호 유사도 - 0.4537
User 525 - Image W_15766_00_metrosexual_M: 선호 유사도 - 0.2905, 비선호 유사도 - 0.4220
User 559 - Image W_05876_70_hippie_M: 선호 유사도 - 0.4942, 비선호 유사도 - 0.3239
User 677 - Image W_24152_60_mods_M: 선호 유사도 - 0.6639, 비선호 유사도 - 0.3945
User 677 - Image W_10823_50_ivy_M: 선호 유사도 - 0.6731, 비선호 유사도 - 0.3834
User 677 - Image W_24747_60_mods_M: 선호 유사도 - 0.4005, 비선호 유사도 - 0.5305
User 681 - Image W_37014_60_minimal_W: 선호 유사도 - 0.4069, 비선호 유사도 - 1.0000
User 681 - Image W_22783_70_hippie_W: 선호 유사도 - 0.4301, 비선호 유사도 - 0.3747
User 791 - Image W_30591_50_ivy_M: 선호 유사도 - 1.0000, 비선호 유사도 - 0.4663
User 837 - Image W_00829_10_sportivecasual_M: 선호 유사도 - 0.3416, 비선호 유사도 - 0.2773
User 837 - Image W_00028_50_ivy_M: 선호 유사도 - 0.3218, 비선호 유사도 - 0.3034
User 837 - Image W_06590_90_hiphop_M: 선호 유사도 - 0.3595, 비선호 유사도 - 0.3256
User 837 - Image W_17305_70_hippie_M: 선호 유사도 - 0.3589, 비선호 유사도 - 0.3160
User 837 - Image W_10104_60_mods_M: 선호 유사도 - 0.3791, 비선호 유사도 - 0.3233
```

- 결과는 이처럼 각 사용자의 Train Image에서 선호 이미지, 비선호 이미지들의 코사인 유사도를 계산해 나온 값으로 Valid Image의 선호 유사도/ 비선호 유사도를 출력함.
- 그리고 이들의 **차이 값**으로 먼저 임계값 없이 비교해서 예측값을 만드는 (양수면 선호, 음수면 비선호) 형태로 정확도를 측정함.

```
# 정확도 계산
accuracy = accuracy_score(y_true, [1 if score > 0 else 0 for score in y_scores])
print(f"임계값 찾기 이전의 정확도: {accuracy:.4f}")
```

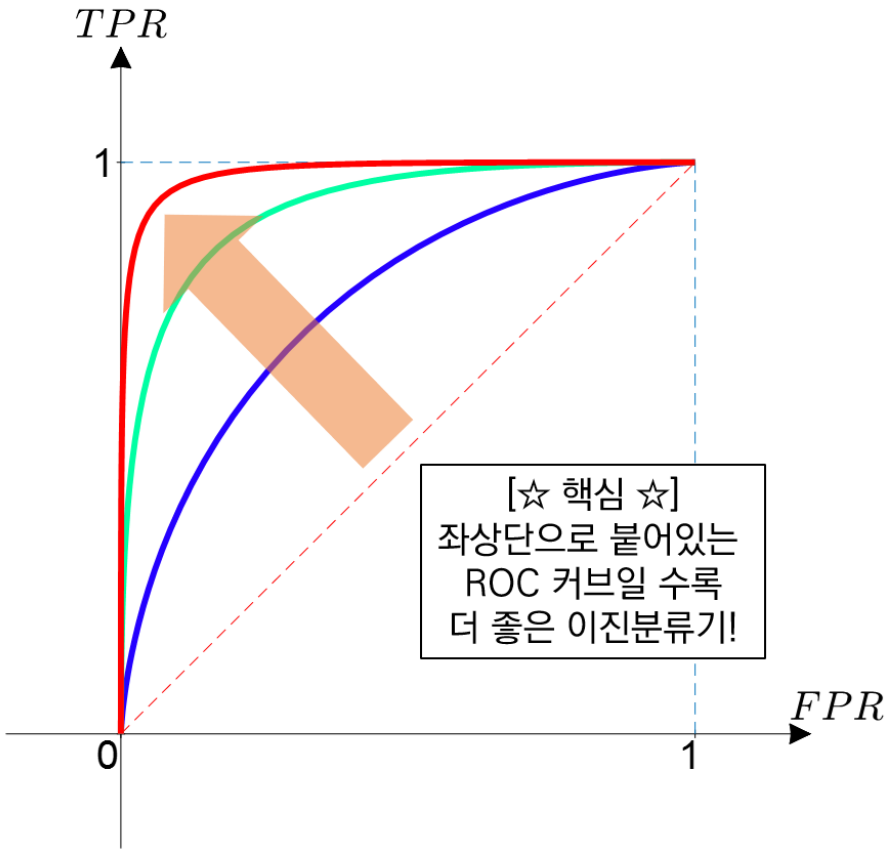
✓ 0.0s

```
임계값 찾기 이전의 정확도: 0.7705
```

따로 선호, 비선호 결정 임계값 없이 음수 양수 정확도
0.7705 = 약 77%

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (5/8)

◆ Resnet-18 특징벡터 기반 추천 시스템 구현 고도화



➢ ROC 커브와 최적 임계값 결정

- 예측된 like_similarity와 dislike_similarity 차이 값을 ROC 커브를 통해 분석하고, 최적의 tpr – fpr 값이 최대가 되는 지점에서 최적 임계값(optimal_threshold)을 결정한다.
- 이 임계값은 유사도 차이에 적용되어 최종적으로 이미지를 선호 또는 비선호로 분류하는 기준이 된다.

최적 임계값
0.0051981241362478725

```
# 정확도 계산
accuracy = accuracy_score(y_true, [1 if score > 0 else 0 for score in y_scores])
print(f"임계값 찾기 이전의 정확도: {accuracy:.4f}")

# 5. ROC 커브를 통해 최적 임계값 찾기
# y_true와 y_scores를 기반으로 ROC 커브 계산
fpr, tpr, thresholds = roc_curve(y_true, y_scores)

# tpr - fpr이 최대인 지점을 최적 임계값으로 선택
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]

# 최적 임계값 출력
print(f"최적 임계값: {optimal_threshold}")
```

✓ 0.0s

```
임계값 찾기 이전의 정확도: 0.7705
최적 임계값: 0.0051981241362478725
```

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (6/8)

◆ Resnet-18 특징벡터 기반 추천 시스템 구현 고도화 결과

```python
# 코사인 유사도를 통해 선호 유사도 및 비선호 유사도 계산
like_similarity = cosine_similarity(val_feature, like_features).mean()
dislike_similarity = cosine_similarity(val_feature, dislike_features).mean()

# 유사도 차이 계산
score_diff = like_similarity - dislike_similarity

# 최적 임계값을 기반으로 최종 예측 결정
final_prediction = "선호" if score_diff >= optimal_threshold else "비선호"
```

➢ 최적 임계값 정확도

- 구한 최적값으로 이전에 수행했던 Validation 검증을 진행하면 됨

```
User 68366 - Image W_24543_70_hippie_M: 예측 - 비선호, 실제 - 비선호
User 68393 - Image W_04836_50_feminine_W: 예측 - 선호, 실제 - 선호
User 68393 - Image W_04840_50_feminine_W: 예측 - 비선호, 실제 - 비선호
User 68396 - Image W_24535_70_hippie_M: 예측 - 선호, 실제 - 비선호
User 68507 - Image T_19945_19_normcore_M: 예측 - 비선호, 실제 - 비선호
User 68574 - Image T_03587_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68729 - Image T_00456_10_sportivecasual_M: 예측 - 선호, 실제 - 선호
User 68747 - Image T_02908_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68808 - Image T_06739_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68813 - Image T_06590_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68826 - Image T_07226_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68883 - Image T_15661_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68891 - Image T_15440_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호

최적 임계값 적용 후 정확도: 77.59%
```

- 최적 임계값을 적용한 Validation Image의 선호/비선호 예측값, 실제값으로 추론한 결과
- Best 정확도 77.59%
- 이전 77.05 대비 0.54% 상승하였다.

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (7/8)

◆ 오토인코더 특징벡터 기반 추천 시스템 구현 테스트

```python
# PyTorch 오토인코더 모델 정의
class Autoencoder(nn.Module):
    def __init__(self, input_dim):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 128),  # 인코딩 단계
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Linear(64, 128),  # 디코딩 단계
            nn.ReLU(),
            nn.Linear(128, input_dim),
            nn.Sigmoid()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

간단한 AutoEncoder 아키텍처

➢ 512개 값은 너무 복잡하지 않을까?

- Autoencoder는 입력 데이터를 압축한 후 다시 복원하는 방식으로 작동하는 신경망 구조이다.

•이 과정을 통해 중요한 특징을 추출하고 불필요한 정보를 제거할 수 있음.

➢ 오토인코더 구현 절차

•특징 벡터에 대해 Autoencoder를 학습.
•학습된 Autoencoder를 사용하여 이미지 특징 벡터를 압축.
•유사도 계산에 압축된 벡터 사용
•간단하게 512->128->64->128->512 형태로 인코더, 디코더를 구성하였다.

Mission 3-2. 패션 스타일 선호 여부 예측 추천 모델 구현 (8/8)

◆ 오토인코더 특징벡터 기반 추천 시스템 구현 테스트

```
Epoch [10/200], Loss: 11.9475
Epoch [20/200], Loss: 11.9180
Epoch [30/200], Loss: 11.8983
Epoch [40/200], Loss: 11.8931
Epoch [50/200], Loss: 11.8845
Epoch [60/200], Loss: 11.8763
Epoch [70/200], Loss: 11.8755
Epoch [80/200], Loss: 11.8724
Epoch [90/200], Loss: 11.8647
Epoch [100/200], Loss: 11.8691
Epoch [110/200], Loss: 11.8631
Epoch [120/200], Loss: 11.8613
Epoch [130/200], Loss: 11.8578
Epoch [140/200], Loss: 11.8576
Epoch [150/200], Loss: 11.8577
Epoch [160/200], Loss: 11.8529
Epoch [170/200], Loss: 11.8543
Epoch [180/200], Loss: 11.8558
Epoch [190/200], Loss: 11.8526
Epoch [200/200], Loss: 11.8566
```

```
# 정확도 계산
accuracy = accuracy_score(y_true, [1 if score > 0 else 0 for score in y_scores])
print(f"임계값 찾기 이전의 정확도: {accuracy:.4f}")

# 5. ROC 커브를 통해 최적 임계값 찾기
# y_true와 y_scores를 기반으로 ROC 커브 계산
fpr, tpr, thresholds = roc_curve(y_true, y_scores)

# tpr - fpr이 최대인 지점을 최적 임계값으로 선택
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]

# 최적 임계값 출력
print(f"최적 임계값: {optimal_threshold}")
```

✓ 0.0s

```
임계값 찾기 이전의 정확도: 0.7220
최적 임계값: 0.014079570770263672
```

```
User 68574 - Image T_03587_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68729 - Image T_00456_10_sportivecasual_M: 예측 - 선호, 실제 - 선호
User 68747 - Image T_02908_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68808 - Image T_06739_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68813 - Image T_06590_10_sportivecasual_M: 예측 - 비선호, 실제 - 비선호
User 68826 - Image T_07226_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68883 - Image T_15661_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호
User 68891 - Image T_15440_10_sportivecasual_M: 예측 - 비선호, 실제 - 선호

최적 임계값 적용 후 정확도: 73.51%
```

- 오토인코더 방식은 잘 동작했지만, 최적 임계값을 적용했음에도 오히려 정확도가 4%나 떨어지고 말았다.
- Resnet-18에서 출력된 512개의 특징벡터 하나하나가 압축이 따로 필요 없이 중요하다는 것을 의미한다.

- 따라서 512개 벡터는 PCA, LDA 등의 일체의 차원 축소 가공 없이 해당 데이터를 그대로 이용한 추천 모델을 최종으로 제시한다.

AutoEncoder 기반 특징압축 결과 테스트

DATA CREATOR CAMP

2024 데이터 크리에이터 캠프

감 사 합 니 다

과학기술정보통신부 NIA 지능정보원