과학기술정보통신부 NIA 지능정보원 한국지능정보사회진흥원

DATA CREATOR CAMP

2024 데이터 크리에이터 캠프

대학부 팀명 : 아마란스

문제. 인공지능은 사람의 마음을 이해할 수 있을까?

■ 연도별 패션 스타일 이미지 및 추천 데이터

Mission 1 패션 스타일 이미지 분류

Mission 2

Mission 3 패션 스타일 선호 여부 예측

Mission 1
패션 스타일 이미지 분류

Mission 1. 패션 스타일 이미지 분류

■ 1-1. 주어진 이미지 데이터의 파일 명은 아래와 같은 형식이다. “{W/T}_{이미지ID}_{시대별}_{스타일별}_{성별}.jpg”이에 기반하여 “이미지ID” 수 기준으로 “성별 & 스타일” 통계치를 아래 표 형식으로 기입한다.

※ Training, Validation 데이터에 대해서 각각 통계표를 작성한다.

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

```
W_96469_60_minimal_W.jpg
W_96472_19_lounge_W.jpg
W_96487_19_lounge_W.jpg
W_96507_19_lounge_W.jpg
W_96514_60_minimal_W.jpg
W_96538_60_minimal_W.jpg
W_96574_19_normcore_W.jpg
W_96596_60_popart_W.jpg
W_96599_60_popart_W.jpg
W_96600_60_minimal_W.jpg
W_96606_60_popart_W.jpg
W_96607_60_popart_W.jpg
W_96612_60_popart_W.jpg
W_96616_60_minimal_W.jpg
W_96617_60_minimal_W.jpg
W_96619_60_minimal_W.jpg
W_96625_60_minimal_W.jpg
W_96626_60_minimal_W.jpg
W_96632_60_minimal_W.jpg
W_96634_60_minimal_W.jpg
W_96637_60_minimal_W.jpg
W_96643_60_minimal_W.jpg
```

1. Jpg 파일에 기반하여 “이미지ID” 수 기준 분류

```python
def count_images_by_gender_and_style(file_names):
    rows = []  # 결과를 저장할 리스트

    # 파일명 분석 및 카운트
    for filename in file_names:
        # 파일 확장자 체크
        if not filename.endswith('.jpg'):
            continue

        # .jpg 제거
        no_jpg_filename = filename.split('.')[0]
        parts = no_jpg_filename.split('_')

        # 성별, 스타일, 이미지 ID 추출
        gender = '여성' if parts[-1] == 'W' else '남성'
        style = parts[3]
        image_id = parts[1]

        # 유효한 스타일과 이미지 ID일 때만 추가
        if style is not None and image_id is not None:
            rows.append({'성별': gender, '스타일': style, '이미지 ID': image_id})

    # DataFrame 생성
    df = pd.DataFrame(rows)

    # 중복된 행 제거
    df = df.drop_duplicates(subset=['성별', '스타일', '이미지 ID'])

    # 성별과 스타일별로 이미지 수 집계
    result = df.groupby(['성별', '스타일']).size().reset_index(name='이미지 수')
    result = result.sort_values(by='성별')

    return result
```

2. "성별 & 스타일"통계치

train

```
# traininig_image 폴더
train_folder_path = '/content/drive/MyDrive/kict/dataset/training_image'

# 폴더 내의 파일 목록 가져오기
train_file_list = os.listdir(train_folder_path)
train_file_names = [filename for filename in train_file_list]
```

```
# traininig_image 폴더
result_train = count_images_by_gender_and_style(train_file_names)
result_train.to_csv('mission_1-1_train.csv', index=False)
```

```
result_train
```

	성별	스타일	이미지 수
0	남성	bold	268
1	남성	hiphop	274
2	남성	hippie	260
3	남성	ivy	237
4	남성	metrosexual	278
5	남성	mods	269
6	남성	normcore	364
7	남성	sportivecasual	298
21	여성	lounge	45
22	여성	military	33
23	여성	minimal	139
24	여성	normcore	153
28	여성	punk	65
26	여성	popart	41
27	여성	powersuit	120
20	여성	lingerie	55
25	여성	oriental	78
19	여성	kitsch	91
15	여성	genderless	77
17	여성	hiphop	48
16	여성	grunge	31
29	여성	space	37
14	여성	feminine	154
13	여성	ecology	64
12	여성	disco	37
11	여성	classic	77
10	여성	cityglam	67
9	여성	bodyconscious	95
8	여성	athleisure	67
18	여성	hippie	91
30	여성	sportivecasual	157

2. “성별 & 스타일”통계치

validation

```
# validation_image 폴더
val_folder_path = '/content/drive/MyDrive/kict/dataset/validation_image'

# 폴더 내의 파일 목록 가져오기
val_file_list = os.listdir(val_folder_path)
val_file_names = [filename for filename in val_file_list]
```

```
# validation_image 폴더
result_val = count_images_by_gender_and_style(val_file_names)
result_val.to_csv('mission_1-1_val.csv', index=False)
```

```
result_val
```

	성별	스타일	이미지 수
0	남성	bold	57
1	남성	hiphop	66
2	남성	hippie	82
3	남성	ivy	79
4	남성	metrosexual	58
5	남성	mods	80
6	남성	normcore	51
7	남성	sportivecasual	52
21	여성	lounge	8
22	여성	military	9
23	여성	minimal	35
24	여성	normcore	20
28	여성	punk	12
26	여성	popart	8
27	여성	powersuit	34
20	여성	lingerie	5
25	여성	oriental	18
19	여성	kitsch	22
15	여성	genderless	12
17	여성	hiphop	8
16	여성	grunge	10
29	여성	space	15
14	여성	feminine	44
13	여성	ecology	17
12	여성	disco	10
11	여성	classic	22
10	여성	cityglam	18
9	여성	bodyconscious	23
8	여성	athleisure	14
18	여성	hippie	14
30	여성	sportivecasual	48

Mission 1. 패션 스타일 이미지 분류

- **1-2. ResNet-18를 활용하여 "성별 & 스타일" 단위로 클래스 분류를 수행하고 Validation 데이터에 대한 정확도를 제시한다.**

  - ResNet-18의 parameters는 무작위로 초기화하여 사용한다.
    (즉, pretrained weights는 사용할 수 없음)

  - 성능을 높이기 위해 object detection, image cropping 등의 다양한 데이터 전처리 기법을 활용해도 무방하다.
    (데이터 전처리 단계에 한해서는 외부 라이브러리 활용 가능)


여성 에콜로지룩


여성 오리엔탈룩


여성 놈코어룩

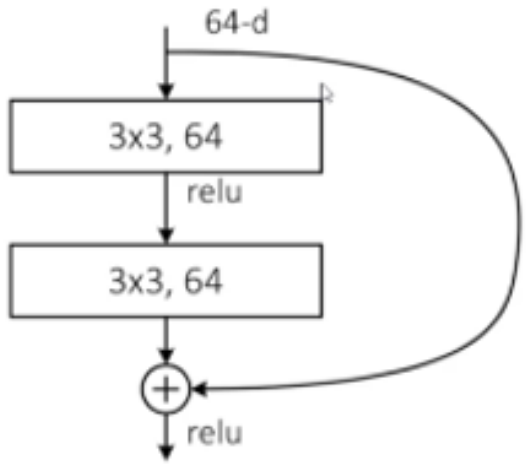

여성 클래식룩

1. Basic block 구현

```python
class BasicBlock(nn.Module):
    def __init__(
        self,
        in_channels: int,
        out_channels: int,
        stride: int = 1,
        expansion: int = 1,
        downsample: nn.Module = None
    ) -> None:
        super(BasicBlock, self).__init__()
        self.expansion = expansion
        self.downsample = downsample
        self.conv1 = nn.Conv2d(
            in_channels,
            out_channels,
            kernel_size=3,
            stride=stride,
            padding=1,
            bias=False
        )
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(
            out_channels,
            out_channels*self.expansion,
            kernel_size=3,
            padding=1,
            bias=False
        )
        self.bn2 = nn.BatchNorm2d(out_channels*self.expansion)
```

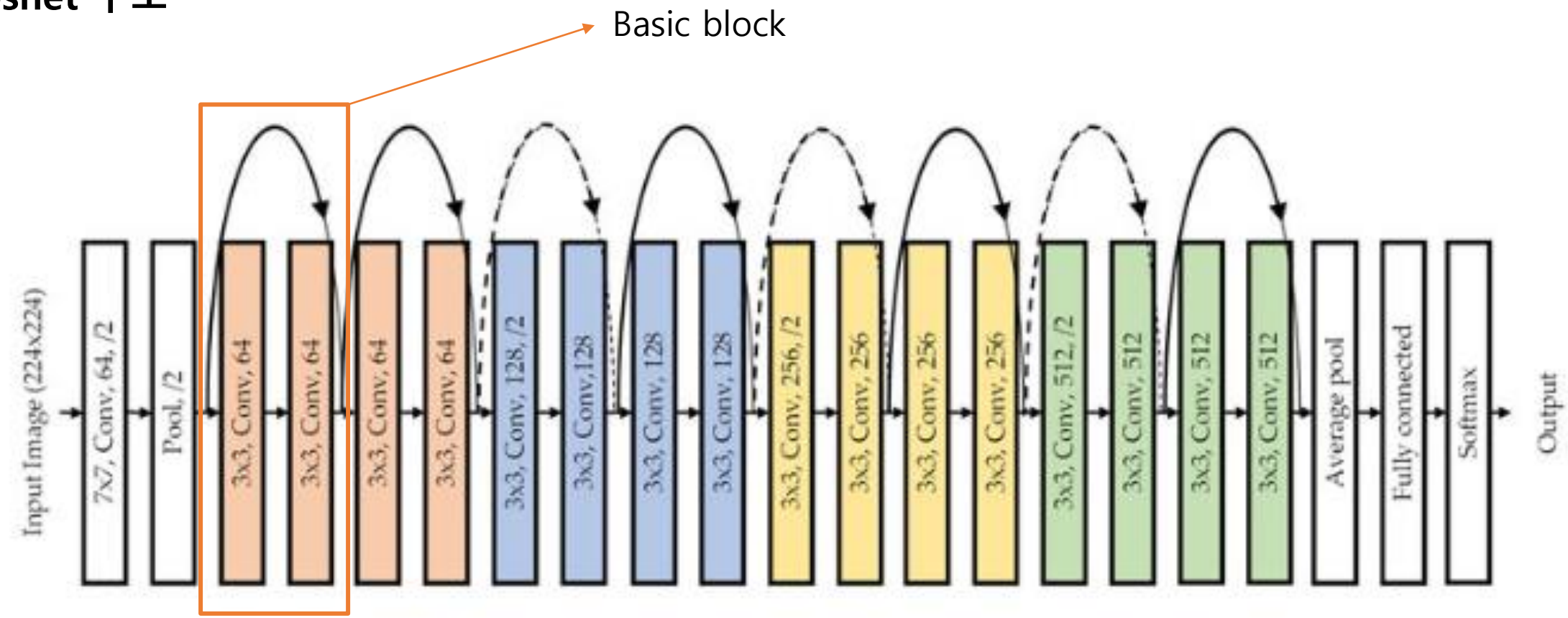
```python
    def forward(self, x: Tensor) -> Tensor:
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            identity = self.downsample(x)

        out += identity
        out = self.relu(out)
        return  out
```

Basic block 구조



BasicBlock

2. Resnet 구조



Structure of the Resnet-18 Model.

2. Resnet 구조

```python
class ResNet(nn.Module):
    def __init__(
        self,
        img_channels: int,
        num_layers: int,
        block: Type[BasicBlock],
        num_classes: int  = 1000
    ) -> None:
        super(ResNet, self).__init__()
        if num_layers == 18: # ResNet18 만을 본 대회에서 사용함으로 18층만 구현
            layers = [2, 2, 2, 2]
            self.expansion = 1

        self.in_channels = 64
        self.conv1 = nn.Conv2d(
            in_channels=img_channels,
            out_channels=self.in_channels,
            kernel_size=7,
            stride=2,
            padding=3,
            bias=False
        )
```

```python
        self.bn1 = nn.BatchNorm2d(self.in_channels)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        self.layer1 = self._make_layer(block, 64, layers[0])
        self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
        self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
        self.layer4 = self._make_layer(block, 512, layers[3], stride=2)

        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512*self.expansion, num_classes)
```

18 layer

1 layer

Basic block x 2 x 4 = 16 layer

1 layer

2. Resnet 구조

```python
def _make_layer(
    self,
    block: Type[BasicBlock],
    out_channels: int,
    blocks: int,
    stride: int = 1
) -> nn.Sequential:
    downsample = None
    if stride != 1:
        downsample = nn.Sequential(
            nn.Conv2d(
                self.in_channels,
                out_channels*self.expansion,
                kernel_size=1,
                stride=stride,
                bias=False
            ),
            nn.BatchNorm2d(out_channels * self.expansion),
        )
    layers = []
    layers.append(
        block(
            self.in_channels, out_channels, stride, self.expansion, downsample
        )
    )
    self.in_channels = out_channels * self.expansion

    for i in range(1, blocks):
        layers.append(block(
            self.in_channels,
            out_channels,
            expansion=self.expansion
        ))
    return nn.Sequential(*layers)
```

```python
def forward(self, x: Tensor) -> Tensor:
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)
    # print('Dimensions of the last convolutional feature map: ', x.shape)

    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)

    return x
```

3. 데이터 처리

```
import numpy as np
import random

def seed_everything(seed: int = 42):
    random.seed(seed)
    np.random.seed(seed)
    os.environ["PYTHONHASHSEED"] = str(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = True
seed_everything()

# 이미지 파일이 있는 디렉토리 경로
train_image_directory = '/content/training_image'
valid_image_directory = '/content/validation_image'
```

```
# CustomDataset 클래스 정의
class CustomDataset(Dataset):
    def __init__(self, image_directory, transform=None):
        self.image_directory = image_directory
        self.image_files = [f for f in os.listdir(image_directory) if f.endswith('.jpg')]
        self.transform = transform

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, idx):
        file_name = self.image_files[idx]
        image_path = os.path.join(self.image_directory, file_name)
        image = Image.open(image_path).convert('RGB')

        # 이미지 파일명에서 스타일과 성별 정보 추출
        parts = file_name.split('_')
        style_gender = parts[-2] + '_' + parts[-1].split('.')[0]  # 스타일과 성별 정보 추출

        # 스타일과 성별 정보를 레이블로 변환
        label = style_gender
        label_idx = label_to_index[label]

        if self.transform:
            image = self.transform(image)
        return image, label_idx
```

3. 데이터 처리

이미지 데이터 정규화를 위한 평균과 표준편차 계산 수행

```python
# 데이터셋의 평균과 표준편차 계산 함수
def calculate_mean_std(loader):
    mean = 0.0
    std = 0.0
    total_images_count = 0
    for images, _ in loader:
        batch_samples = images.size(0)  # 배치 크기 (이때 마지막 배치는 더 작을 수 있음)
        images = images.view(batch_samples, images.size(1), -1)
        mean += images.mean(2).sum(0)
        std += images.std(2).sum(0)
        total_images_count += batch_samples

    mean /= total_images_count
    std /= total_images_count
    return mean, std

# 임시로 ToTensor 변환만 적용하여 데이터 로더 생성
temp_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
])

temp_train_dataset = CustomDataset(train_image_directory, transform=temp_transform)
temp_train_loader = DataLoader(temp_train_dataset, batch_size=32, shuffle=True)

# 평균과 표준편차 계산
mean, std = calculate_mean_std(temp_train_loader)
print(f"Calculated mean: {mean}")
print(f"Calculated std: {std}")
```

```
Calculated mean: tensor([0.5498, 0.5226, 0.5052])
Calculated std: tensor([0.2600, 0.2582, 0.2620])
```

데이터 전처리 및 로드

```python
# 데이터 전처리 및 로드
transform = transforms.Compose([
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
    transforms.RandomRotation(degrees=15),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.ColorJitter(brightness=0.2,
                           contrast=0.2,
                           saturation=0.2,
                           hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5498, 0.5226, 0.5052], std=[0.2600, 0.2582, 0.2620]),
])

train_dataset = CustomDataset(train_image_directory, transform=transform)
val_dataset = CustomDataset(valid_image_directory, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

4. Resnet 구축

모델 구축

모델 학습

```python
# ResNet-18 모델 생성 함수
def resnet18(img_channels: int, num_classes: int) -> ResNet:
    return ResNet(img_channels, 18, BasicBlock, num_classes) # 18은 ResNet-18을 지정

# 모델 인스턴스 생성
num_classes = len(label_list)  # 레이블의 총 개수
model = resnet18(img_channels=3, num_classes=num_classes) # RGB 이미지: 3개의 채널

# 손실 함수 및 옵티마이저 정의
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 모델을 GPU로 이동
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

# 모델 학습
num_epochs = 100
patience = 5  # Early stopping patience
best_val_loss = float('inf')  # Initialize best validation loss as infinity
counter = 0  # Counter to track patience
```

```python
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        optimizer.zero_grad()
        images, labels = images.to(device), labels.to(device)

        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

        # 정확도 계산
        _, preds = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (preds == labels).sum().item()

    train_loss = running_loss / len(train_loader)
    train_accuracy = correct / total * 100

    print(f'Epoch [{epoch+1}/{num_epochs}], Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.2f}%')
```

4. Resnet 구축

```python
# Validation phase
model.eval()
all_preds = []
all_labels = []
val_running_loss = 0.0
val_correct = 0
val_total = 0

with torch.no_grad():
    for images, labels in val_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        loss = criterion(outputs, labels)
        val_running_loss += loss.item()

        _, preds = torch.max(outputs, 1)
        val_total += labels.size(0)
        val_correct += (preds == labels).sum().item()
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Calculate average validation loss and accuracy
avg_val_loss = val_running_loss / len(val_loader)
val_accuracy = val_correct / val_total * 100

print(f'Epoch [{epoch+1}/{num_epochs}], Validation Loss: {avg_val_loss:.4f}, Validation Accuracy: {val_accuracy:.2f}%')
```

```python
# Early stopping logic
if avg_val_loss < best_val_loss:
    best_val_loss = avg_val_loss
    counter = 0  # Reset counter if validation loss improves
else:
    counter += 1
    if counter >= patience:
        print(f"Early stopping at epoch {epoch+1}")
        break
```

일정성능 변화 x
-> 조기종료

4. Resnet 구축

모델 검증

```
# 정확도 계산
accuracy = accuracy_score(all_labels, all_preds)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')
```

```
# 모델 저장
torch.save(model, '/content/drive/MyDrive/kict/model_path.pth')
```

```
Epoch [1/100], Loss: 3.357233326882124
Epoch [1/100], Validation Loss: 3.270069980621338
Epoch [2/100], Loss: 3.233607143163681
Epoch [2/100], Validation Loss: 3.208656088511149
Epoch [3/100], Loss: 3.2227494679391384
Epoch [3/100], Validation Loss: 3.25605149269104
Epoch [4/100], Loss: 3.195202112197876
Epoch [4/100], Validation Loss: 3.1865834395090737
Epoch [5/100], Loss: 3.180710546672344
Epoch [5/100], Validation Loss: 3.215518109003703
...
Epoch [59/100], Loss: 0.8542025191709399
Epoch [59/100], Validation Loss: 2.2537124395370483
Epoch [60/100], Loss: 0.7428056672215462
Epoch [60/100], Validation Loss: 2.449845727284749
Epoch [61/100], Loss: 0.715821304358542
Epoch [61/100], Validation Loss: 2.3449257055918378
Early stopping at epoch 61
Validation Accuracy: 49.63%
```

Resnet 예측 정확도 : 49.63%

Mission 2
패션 스타일 선호 여부 예측

Mission 2. 패션 스타일 선호 여부 예측

■ 2-1. 주어진 라벨링 데이터의 파일 명은 아래와 같은 형식이다.“{W/T}_{이미지ID}_{시대별}_{스타일별}_{성별}_{설문ID}.json”이에 기반하여 “설문ID” 수 기준으로 “성별 & 스타일” 통계치를 아래 표 형식으로 기입한다.

```
W_96626_60_minimal_W_008455.json
W_96626_60_minimal_W_234988.json
W_96632_60_minimal_W_008463.json
W_96632_60_minimal_W_234973.json
W_96634_60_minimal_W_008471.json
W_96634_60_minimal_W_234981.json
W_96637_60_minimal_W_008472.json
W_96637_60_minimal_W_234989.json
W_96643_60_minimal_W_018557.json
W_96643_60_minimal_W_234974.json
W_96645_60_minimal_W_018565.json
W_96645_60_minimal_W_234982.json
W_96646_60_minimal_W_018573.json
W_96646_60_minimal_W_234990.json
```

※ 이때 주어진 이미지 데이터에 존재하는 “이미지ID”를 식별하여 유효한 라벨링 데이터 대상으로만 통계치를 구해야 한다. (이미지ID 기준으로 라벨링 데이터에는 있지만, 이미지 데이터에는 없는 경우가 있음)

※ Training, Validation 데이터에 대해서 각각 통계표를 작성한다.

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

1. 유효한 라벨링 탐색

라이브러리 불러오기

```
import os
import json
import csv
import pandas as pd
from collections import defaultdict, Counter
```

디렉토리 경로 지정

```
# 파일이 있는 디렉토리 경로
train_label_directory = '../dataset/training_label'
valid_label_directory = '../dataset/validation_label'
train_image_directory = '../dataset/training_image'
valid_image_directory = '../dataset/validation_image'
```

```
def count_images_by_gender_and_style(file_names, valid_image_ids):
    rows = []  # 결과를 저장할 리스트

    # 파일명 분석 및 카운트
    for filename in file_names:
        # 파일 확장자 체크
        if not filename.endswith('.json'):
            continue

        # .json 제거
        no_jpg_filename = filename.split('.')[0]
        parts = no_jpg_filename.split('_')

        # 성별, 스타일, 이미지 ID 추출
        gender = '여성' if parts[4] == 'W' else '남성'
        style = parts[3]
        image_id = parts[1]

        # 유효한 스타일과 이미지 ID일 때만 추가
        if style is not None and image_id is not None and image_id in valid_image_ids:
            rows.append({'성별': gender, '스타일': style, '이미지 ID': image_id})

    # DataFrame 생성
    df = pd.DataFrame(rows)

    # 중복된 행 제거
    df = df.drop_duplicates(subset=['성별', '스타일', '이미지 ID'])

    # 성별과 스타일별로 이미지 수 집계
    result = df.groupby(['성별', '스타일']).size().reset_index(name='이미지 수')
    result = result.sort_values(by='성별')

    return result
```

2. "성별 & 스타일" 통계치

train 데이터

```
# 폴더 내의 파일 목록 가져오기
train_file_list_label = os.listdir(train_label_directory)
train_file_names_label = [filename for filename in train_file_list_label]

# 폴더 내의 파일 목록 가져오기
train_file_image_list = os.listdir(train_image_directory)
train_file_names = [filename for filename in train_file_image_list]

# 이미지 ID 추출
train_image_ID = list(set(name.split('_')[1] for name in train_file_names))

# traininig_image 폴더
result_train = count_images_by_gender_and_style(train_file_names_label, train_image_ID)
result_train.to_csv('mission_2-1_train.csv', index=False)
```

```
result_train
```

	성별	스타일	이미지 수
0	남성	bold	312
1	남성	hiphop	353
2	남성	hippie	274
3	남성	ivy	256
4	남성	metrosexual	332
5	남성	mods	330
6	남성	normcore	754
7	남성	sportivecasual	687
21	여성	lounge	167
22	여성	military	36
23	여성	minimal	188
24	여성	normcore	497
28	여성	punk	74
26	여성	popart	42
27	여성	powersuit	150
20	여성	lingerie	70
25	여성	oriental	119
19	여성	kitsch	147
15	여성	genderless	161
17	여성	hiphop	101
16	여성	grunge	47
29	여성	space	42
14	여성	feminine	163
13	여성	ecology	100
12	여성	disco	41
11	여성	classic	113
10	여성	cityglam	116
9	여성	bodyconscious	109
8	여성	athleisure	82
18	여성	hippie	106
30	여성	sportivecasual	287

2. "성별 & 스타일" 통계치

validation 데이터

```
# 폴더 내의 파일 목록 가져오기
val_file_list_label = os.listdir(valid_label_directory)
val_file_names_label = [filename for filename in val_file_list_label]

val_file_list_image = os.listdir(valid_image_directory)
val_file_names = [filename for filename in val_file_list_image]

# 이미지 ID 추출
val_image_ID = list(set(name.split('_')[1] for name in val_file_names))

# validation_image 폴더
result_val = count_images_by_gender_and_style(val_file_names_label, val_image_ID)
result_val.to_csv('mission_2-1_val.csv', index=False)
```

```
result_val
```

	성별	스타일	이미지 수
0	남성	bold	59
1	남성	hiphop	67
2	남성	hippie	82
3	남성	ivy	79
4	남성	metrosexual	58
5	남성	mods	81
6	남성	normcore	63
7	남성	sportivecasual	66
21	여성	lounge	12
22	여성	military	9
23	여성	minimal	37
24	여성	normcore	29
28	여성	punk	12
26	여성	popart	8
27	여성	powersuit	37
20	여성	lingerie	6
25	여성	oriental	20
19	여성	kitsch	22
15	여성	genderless	12
17	여성	hiphop	13
16	여성	grunge	10
29	여성	space	16
14	여성	feminine	44
13	여성	ecology	20
12	여성	disco	10
11	여성	classic	26
10	여성	cityglam	20
9	여성	bodyconscious	24
8	여성	athleisure	14
18	여성	hippie	14
30	여성	sportivecasual	57

Mission 2. 패션 스타일 선호 여부 예측

■ 2-2. 2-1에서 구한 유효한 라벨링 데이터만 따로 분리하여 아래와 같이 100명 응답자의 “스타일 선호 정보표”를 구한다. 파일은 json 포맷으로 되어 있으며 json 필드 중, “응답자ID”는 “user>R_id”로 알 수 있고, “스타일 선호 여부”는 “item>survey>Q5”로 알 수 있다.

※ 스타일 선호도 값은 “1: 비선호”, “2: 선호”이다.

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
	…	…	…	…
…				

1. 유효한 라벨링 데이터를 이용하여 스타일 선호 정보표

```python
# 결과를 저장할 딕셔너리
train_result = defaultdict(lambda: {'선호': [], '비선호': []})
valid_result = defaultdict(lambda: {'선호': [], '비선호': []})
response_count = Counter()

# 디렉토리 내의 모든 파일을 순회하며 데이터 추출
def process_directory(directory, result_dict, image_directory):
    for filename in os.listdir(directory):
        if filename.endswith('.json'):
            filepath = os.path.join(directory, filename)
            with open(filepath, 'r', encoding='utf-8') as file:
                data = json.load(file)
                R_id = data['user']['R_id']
                imgName = data['item']['imgName']
                Q5 = data['item']['survey']['Q5']

                # 이미지 파일 경로 생성
                image_path = os.path.join(image_directory, imgName)

                # 이미지 파일이 존재할 때만 처리
                if os.path.exists(image_path):
                    # Q5 값을 선호도 값으로 변환
                    preference = "선호" if Q5 == 2 else "비선호"

                    result_dict[R_id][preference].append(imgName)
                    response_count[R_id] += 1
```

```python
# train과 valid 디렉토리 처리
process_directory(train_label_directory, train_result, train_image_directory)
process_directory(valid_label_directory, valid_result, valid_image_directory)

# 결과를 CSV 파일로 저장
with open('mission2-2_result_all.csv', 'w', newline='', encoding='utf-8-sig') as csvfile:
    fieldnames = ['응답자 ID', 'train 선호', 'train 비선호', 'valid 선호', 'valid 비선호']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
    for R_id in response_count:
        train_preference = ', '.join(train_result[R_id]['선호'])
        train_non_preference = ', '.join(train_result[R_id]['비선호'])
        valid_preference = ', '.join(valid_result[R_id]['선호'])
        valid_non_preference = ', '.join(valid_result[R_id]['비선호'])

        writer.writerow({
            '응답자 ID': R_id,
            'train 선호': train_preference,
            'train 비선호': train_non_preference,
            'valid 선호': valid_preference,
            'valid 비선호': valid_non_preference
        })
```

2. 전체 응답자에 대한 스타일 선호 정보표

전체 응답자에 대한 선호여부

```
import pandas as pd

# 출력 옵션 설정
pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_columns', None)

# CSV 파일 읽기
result = pd.read_csv("mission2-2_result_all.csv")

result.tail()
```

	응답자 ID	train 선호	train 비선호	valid 선호	valid 비선호
3475	61765	NaN	NaN	W_48457_60_minimal_W.jpg	NaN
3476	62013	NaN	NaN	NaN	W_52417_00_metrosexual_M.jpg
3477	66230	NaN	NaN	NaN	W_52521_50_ivy_M.jpg
3478	65680	NaN	NaN	W_53808_80_bold_M.jpg	NaN
3479	67649	NaN	NaN	NaN	W_63188_90_kitsch_W.jpg

3. 응답자 100명에 대한 스타일 선호 정보표

응답자 100명에 대한 선호여부

```
# 상위 100명 데이터 추출
top_100 = result.head(100)

# 상위 100명 데이터를 새로운 CSV 파일로 저장
top_100.to_csv("mission2-2_result_top100.csv", index=False, encoding='utf-8-sig')
```

```
import pandas as pd

# CSV 파일 읽기
top_100_result = pd.read_csv("mission2-2_result_top100.csv")

# 모든 행을 생략 없이 출력하도록 설정
pd.set_option('display.max_rows', None)

# 데이터프레임 출력
top_100_result
```

	응답자 ID	train 선호	train 비선호	valid 선호	valid
0	58049	NaN	T_00253_60_popart_W.jpg	NaN	T_00253_60_popar
1	62192	NaN	T_00253_60_popart_W.jpg	NaN	T_00253_60_popar
2	64213	NaN	T_00253_60_popart_W.jpg	NaN	T_00253_60_popar
3	66592	T_00253_60_popart_W.jpg, T_00893_90_hiphop_W.j...	T_07452_50_classic_W.jpg, W_02170_50_feminine_...	T_00253_60_popart_W.jpg, W_10028_50_classic_W....	W_02170_50_feminine W_19352_50_fer
4	66721	W_05960_70_hippie_W.jpg	T_00253_60_popart_W.jpg	NaN	T_00253_60_popar
5	66469	T_00456_10_sportivecasual_M.jpg, T_00588_10_sp...	T_02958_19_normcore_M.jpg, T_06076_60_mods_M.j...	T_00456_10_sportivecasual_M.jpg, T_01123_90_hi...	W_24553_70_hippie W_24647_70_hipp
6	68729	T_00456_10_sportivecasual_M.jpg	NaN	T_00456_10_sportivecasual_M.jpg	
7	66553	W_25518_60_mods_M.jpg	T_00588_10_sportivecasual_M.jpg, T_03772_90_hi...	NaN	W_16104_60_mod
8	28695	T_00770_60_minimal_W.jpg	NaN	NaN	
			...		
92	64035	W_00027_50_ivy_M.jpg, W_01539_60_mods_M.jpg, W...	W_04308_50_ivy_M.jpg, W_09772_70_hippie_M.jpg,...	NaN	
93	64321	W_02769_90_hiphop_M.jpg, W_17221_70_hippie_M.j...	W_00027_50_ivy_M.jpg, W_00031_50_ivy_M.jpg, W_...	W_17221_70_hippie_M.jpg, W_24109_60_mods_M.jpg	
94	837	W_00028_50_ivy_M.jpg, W_00829_10_sportivecasua...	W_07130_19_normcore_M.jpg, W_11107_19_normcore...	W_00028_50_ivy_M.jpg, W_00829_10_sportivecasua...	W_12154_80_bol W_15661_70_hippie
95	23831	W_00028_50_ivy_M.jpg, W_00951_70_hippie_M.jpg,...	W_06629_90_hiphop_M.jpg, W_09777_90_hiphop_M.j...	W_00028_50_ivy_M.jpg, W_12789_00_metrosexual_M...	W_09777_90_hipho
96	25446	W_00028_50_ivy_M.jpg, W_00866_90_hiphop_M.jpg,...	W_12095_80_bold_M.jpg, W_12214_70_hippie_M.jpg...	W_00028_50_ivy_M.jpg, W_10823_50_ivy_M.jpg, W_...	W_12214_70_hippi W_16747_00_metr
97	62820	W_04377_10_sportivecasual_M.jpg, W_15472_70_hi...	W_00028_50_ivy_M.jpg, W_10060_50_ivy_M.jpg, W_...	NaN	W_00028_50_iv W_17333_19_normcor
98	64345	W_00028_50_ivy_M.jpg, W_00843_10_sportivecasua...	W_06211_10_sportivecasual_M.jpg, W_06570_70_hi...	W_00028_50_ivy_M.jpg, W_12847_19_normcore_M.jp...	W_12730_00_metrosexua W_24923_00
99	64472	W_00028_50_ivy_M.jpg, W_02696_60_mods_M.jpg	W_04307_60_mods_M.jpg, W_04408_60_mods_M.jpg, ...	W_00028_50_ivy_M.jpg, W_00804_50_ivy_M.jpg, W_...	W_16375_80_bol

Mission 3
패션 스타일 선호 여부 예측

Mission 3. 패션 스타일 선호 여부 예측

■ 3-1. 추천 시스템의 기본인 협업 필터링 (Collaborative Filtering)은 크게 user-based filtering, item-based filtering 방식으로 나뉘어져 있다. 각각에 대해서 이해하고, 2-2에서 구해 본 응답자의 “스타일 선호 정보표”를 토대로 Validation 데이터 내 응답자의 “스타일 선호 여부 예측” 문제를 2가지 기법으로 어떻게 적용해 볼 수 있고, 서로 비교하여 어떤 장단점을 갖는지 설명한다.

※ 설명을 용이하게 하기 위해 응답자의 스타일 선호도 예시를 들어서 설명해도 무방하다.

1. 추천시스템 방법

사용자 또는 아이템 필터링?

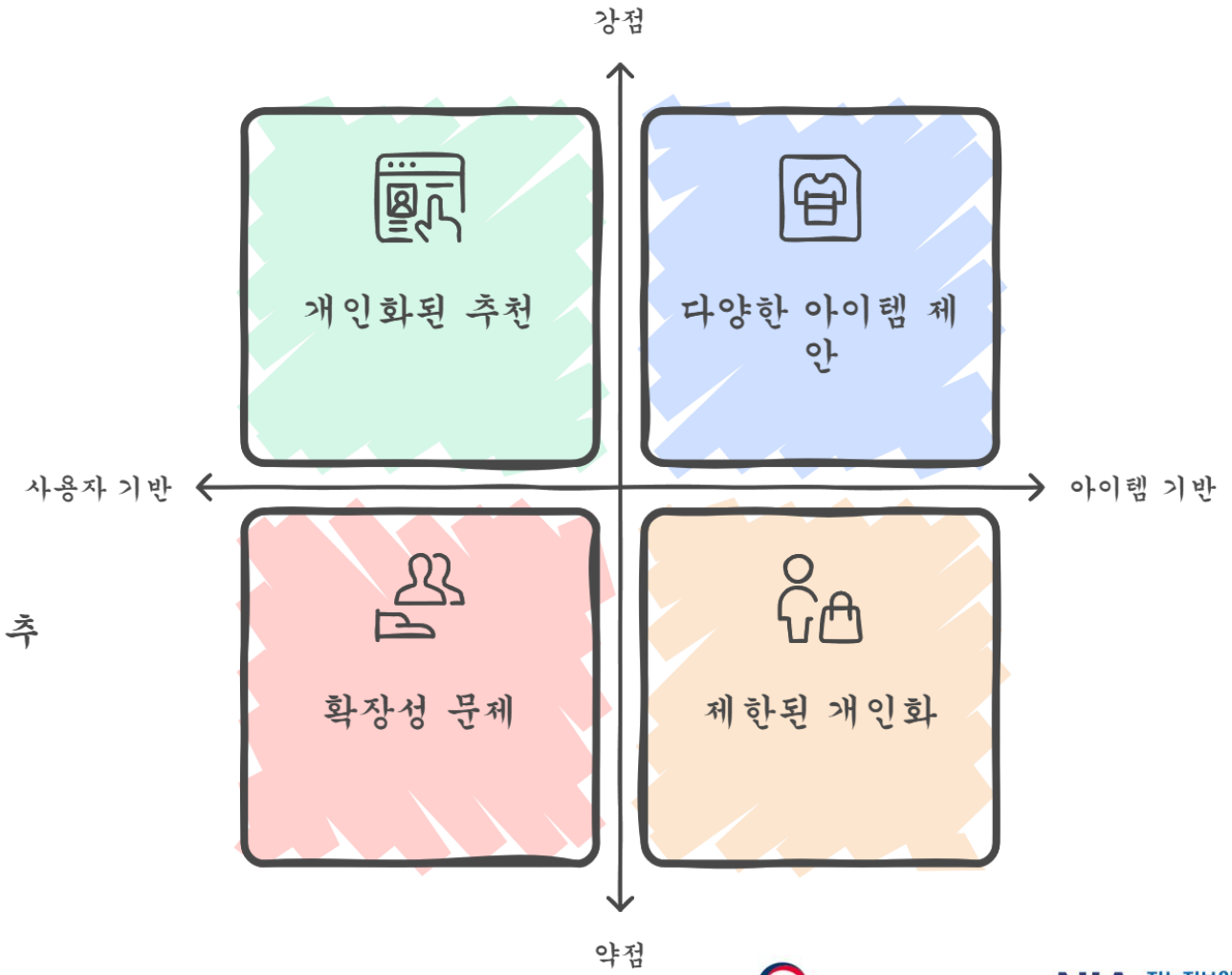
사용자 기반

사용자 선호에 기반한 개인화된 추천.

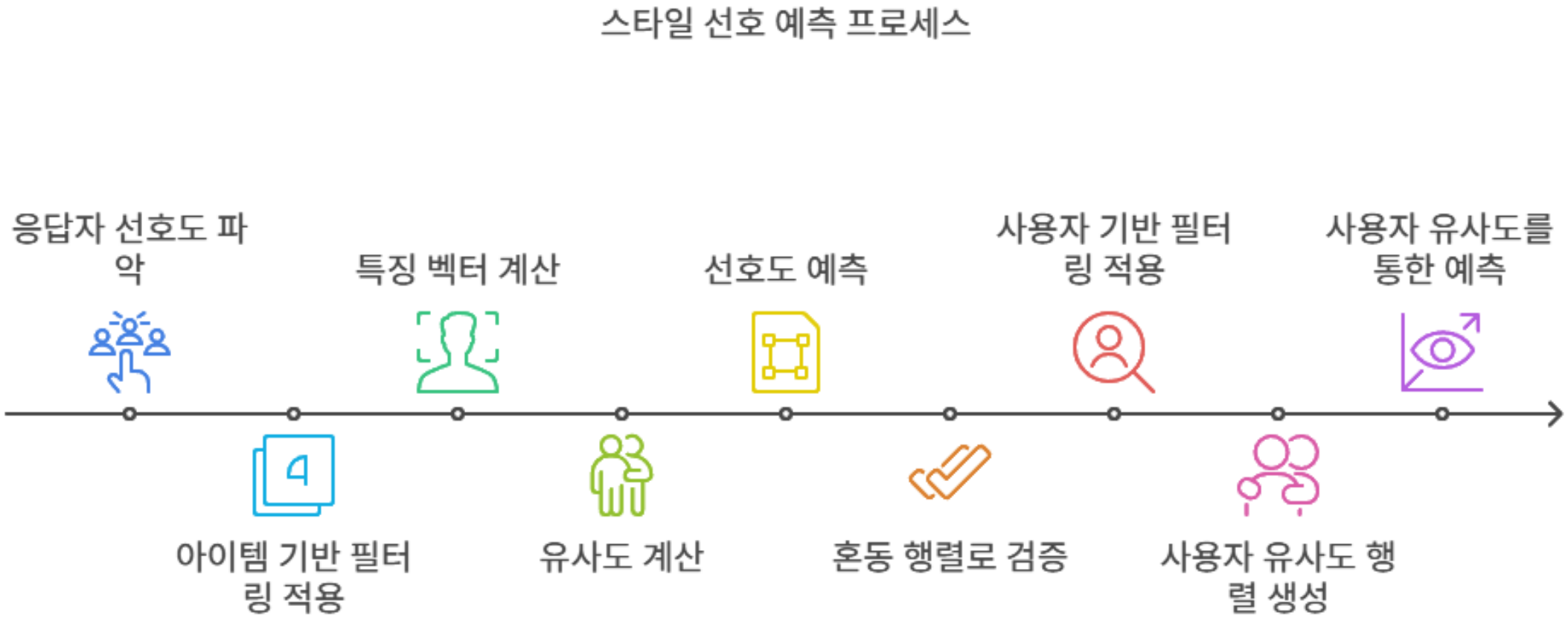
아이템 기반

아이템 유사성에 기반한 추천.

필터링 방법 비교



2. 스타일 선호 정보표 기반 응답자의 스타일 선호 여부를 예측



Mission 3. 패션 스타일 선호 여부 예측

■ 3-2. 3-1에서 살펴 본 기법 중, item-based filtering을 직접 구현해본다. “이미지 간 유사도” (image2image)만을 활용하여 Validation 데이터 내 응답자의 “스타일 선호 여부 예측” 문제를 수행하고 성능을 측정한다.

※ Hint. 1-2에서 학습한 ResNet-18의 중간 layer 값을 활용하여 각 이미지의 feature vector를 구하고, 벡터 연산을 통해 이미지 간 유사도를 구해볼 수 있다.

※ 예측 문제에서 활용한 파라미터 및 임계 값 등의 수치를 정확하게 제시한다.

1. Resnet 모델 불러오기

이미지 전처리

ResNet-18 모델을 사용하여 이미지의 feature vector를 추출 및 저장

```
import os
import torch
from torchvision import transforms
from PIL import Image

# 이미지 전처리
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5498, 0.5226, 0.5052], std=[0.2600, 0.2582, 0.2620]),
])

# 이미지 feature vector 추출 함수
def extract_features(image_path, model, transform):
    if not os.path.exists(image_path):
        print(f"File not found: {image_path}")
        return None
    image = Image.open(image_path).convert('RGB')
    image = transform(image).unsqueeze(0)  # Add batch dimension
    with torch.no_grad():
        features = model(image)
    return features.numpy().flatten()
```

```
# ResNet-18 모델 초기화
class ResNet18FeatureExtractor(torch.nn.Module):
    def __init__(self):
        super(ResNet18FeatureExtractor, self).__init__()
        self.resnet18 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', pretrained=False)
        self.resnet18.fc = torch.nn.Identity()  # 마지막 FC 레이어를 Identity로 대체

    def forward(self, x):
        return self.resnet18(x)

# 저장된 모델 전체 로드
model_path = r'../model/model_path.pth'
model = torch.load(model_path, map_location=torch.device('cpu'))
model.eval()
```

1. Resnet 모델 불러오기

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=31, bias=True)
)
```

2. 이미지 간 유사도를 계산

```python
import os
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# 학습 이미지와 검증 이미지의 디렉토리 경로 설정
train_image_directory = '../dataset/training_image'
valid_image_directory = '../dataset/validation_image'

# 유사도 계산 함수
def calculate_similarity(feature1, feature2):
    return cosine_similarity([feature1], [feature2])[0][0]
```

```python
# 응답자별 학습 이미지와 검증 이미지의 feature vector 추출 및 저장
def extract_respondent_features(mission2_result, model, transform):
    respondent_train_features = {}
    respondent_valid_features = {}

    for index, row in mission2_result.iterrows():
        respondent_id = row['응답자 ID']
        train_images = []
        valid_images = []

        if not pd.isna(row['train 선호']):
            train_images.extend([(img, '선호') for img in row['train 선호'].split(', ')])
        if not pd.isna(row['train 비선호']):
            train_images.extend([(img, '비선호') for img in row['train 비선호'].split(', ')])
        if not pd.isna(row['valid 선호']):
            valid_images.extend(row['valid 선호'].split(', '))
        if not pd.isna(row['valid 비선호']):
            valid_images.extend(row['valid 비선호'].split(', '))

        respondent_train_features[respondent_id] = {}
        respondent_valid_features[respondent_id] = {}

        for train_image, preference in train_images:
            image_path = os.path.join(train_image_directory, train_image).replace('₩₩', '/')
            if not os.path.exists(image_path):
                print(f"File not found: {image_path}")
                continue
            image_id = train_image.split('_')[1]
            feature = extract_features(image_path, model, transform)
            if feature is not None:
                respondent_train_features[respondent_id][image_id] = (feature, preference)

        for valid_image in valid_images:
            image_path = os.path.join(valid_image_directory, valid_image).replace('₩₩', '/')
            if not os.path.exists(image_path):
                print(f"File not found: {image_path}")
                continue
            image_id = valid_image.split('_')[1]
            feature = extract_features(image_path, model, transform)
            if feature is not None:
                respondent_valid_features[respondent_id][image_id] = feature

    return respondent_train_features, respondent_valid_features
```

3. Validation 데이터 내 응답자의 스타일 선호 여부를 예측

```python
# Validation 데이터 내 응답자의 스타일 선호 여부 예측
def predict_preference(respondent_train_features, respondent_valid_features, threshold=0.5):
    predictions = {}
    for respondent_id, valid_features in respondent_valid_features.items():
        respondent_predictions = {}
        for valid_image_id, valid_feature in valid_features.items():
            similarities = {'선호': [], '비선호': []}
            for train_image_id, (train_feature, preference) in respondent_train_features[respondent_id].items():
                similarity = calculate_similarity(valid_feature, train_feature)
                similarities[preference].append(similarity)
            # 모든 유사도를 사용하여 평균 계산
            if len(similarities['선호']) == 0 and len(similarities['비선호']) == 0:
                preference_score = 0  # 기본값 설정
            else:
                preference_score = (sum(similarities['선호']) - sum(similarities['비선호'])) / (len(similarities['선호']) + len(similarities['비선호']))
            respondent_predictions[valid_image_id] = '선호' if preference_score > threshold else '비선호'
        predictions[respondent_id] = respondent_predictions
    return predictions
```

```python
predictions = predict_preference(respondent_train_features, respondent_valid_features, threshold=0.5)
```

4. 스타일 선호 여부 최종 예측 성능

성능 확인

```
# 성능 측정 (예시로 정확도 계산)
def calculate_accuracy(predictions, mission2_result):
    correct = 0
    total = 0
    for index, row in mission2_result.iterrows():
        respondent_id = row['응답자 ID']
        if respondent_id not in predictions:
            continue
        for valid_image in row['valid 선호'].split(', ') if not pd.isna(row['valid 선호']) else []:
            valid_image_id = valid_image.split('_')[1]
            if valid_image_id in predictions[respondent_id]:
                total += 1
                if predictions[respondent_id][valid_image_id] == '선호':
                    correct += 1
        for valid_image in row['valid 비선호'].split(', ') if not pd.isna(row['valid 비선호']) else []:
            valid_image_id = valid_image.split('_')[1]
            if valid_image_id in predictions[respondent_id]:
                total += 1
                if predictions[respondent_id][valid_image_id] == '비선호':
                    correct += 1
    return correct / total if total > 0 else 0

# 정확도 계산
accuracy = calculate_accuracy(predictions, mission2_result)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.55
```

최종 예측 정확도 : 0.55

DATA CREATOR CAMP

2024 데이터 크리에이터 캠프

감 사 합 니 다

과학기술정보통신부 NIA 지능정보원