한신대학교

한신대학교 중심대학사업단

캡스톤디자인 결과보고서

프로젝트	제 목	AI를 활용한 자동 코디추천 어플리케이션 개발		
팀장	팀 명	TEAM WD		
	성 명	김민성	학번	201958003
	연락처	010-2268-9396		
	E-MAIL	ava9797@hs.ac.kr		
구분	성 명	학번	E-MAIL	연락처(H.P)
팀원 인적사항	김로은	202258011	kdhok2285@hs.ac.kr	010-3683-2285
	이유빈	202258016	miny2482@hs.ac.kr	010-7566-4295
	정민준	202058010	nicemj0914@hs.ac.kr	010-5364-2817
지도교수	박기홍 (서명)			

본인과 팀원은 2025학년도 1학기 캡스톤디자인 과목의
프로젝트에 대한 결과보고서를 다음과 같이 제출합니다.

2025 년 5 월 30 일

팀 장 : 김민성 (서명)

한신대학교 SW중심대학사업단장 귀하

목 차

1. 서론

가. 연구배경

현대 사람들은 자신의 개성과 라이프스타일을 반영한 맞춤형 패션 코디를 선호하고 있다. 하지만, 많은 사람들이 일상 속에서 어떤 옷을 입어야 할지 결정하는 데 있어 상당한 시간을 소비하고 있다. 실생활 속의 문제를 실증적으로 확인하기 위해 10~20대 남녀 100명을 대상으로 설문조사를 진행하였다.

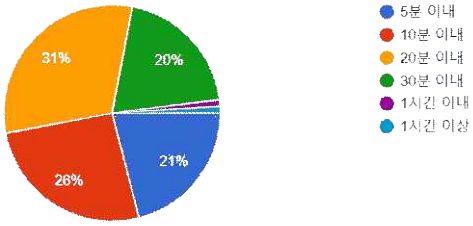

그림 1. '외출 전 옷 고르는 시간은 얼마나 소요되는가?'에 대한 설문 결과

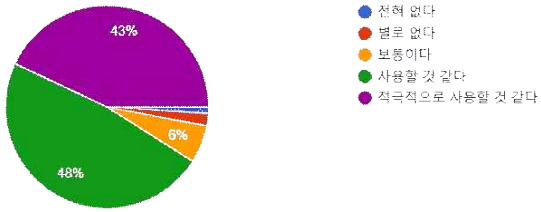

그림 2. '코디하는 것이 어렵다고 느낀 적이 있는가?'에 대한 설문 결과

위 설문조사에 따르면, 응답자들은 외출 전 평균 약 18분을 옷을 고르는 데 소요한다고 응답하였다. 이는 코디 과정에서 반복적인 시간 소모를 유발할 수 있음을 시사한다. 또한, 2번 항목에서 전체 응답자의 72%가 "코디하는 것이 어렵다"고 응답("가끔 있다" 41%, "항상 어렵다" 31%)한 점을 통해 많은 이들이 일상적인 옷 선택 과정에서 어려움을 겪고 있다는 것을 보여준다.

이를 통해 사용자 의류 스타일링을 보조하는 시스템에 대한 필요성을 확인할 수 있으며, 본 프로젝트는 이를 바탕으로 딥러닝 기반 의류 분석 기술을 활용한 시스템 구축을 목표로 한다.

2. 관련 연구

가. Mask R-CNN

Mask R-CNN은 Facebook AI Research(FAIR)의 Kaiming He 등이 2017년에 발표한 Object Detection과 Instance Segmentation을 동시에 수행하는 딥러닝 기반 모델이다. Faster R-CNN에 마스크 분할 branch를 추가한 것이 주요 특징이다. RoI 영역마다 클래스, 위치, 마스크를 병렬로 예측하며, RoIAlign 기법을 통해 feature map과의 공간 정렬 오차를 최소화한다.

본 프로젝트에서는 Mask R-CNN을 활용하여 이미지에서 의류 객체를 감지하고, 의류 카테고리 분류 기능을 수행한다.

나. OpenCV

OpenCV는 영상 및 이미지 처리에 특화된 오픈소스라이브러리로, 다양한 프로그래밍 언어와 플랫폼을 지원한다. 주요 기능으로는 윤곽선 추출, 특징점 검출 및 객체 추적 등이 있으며, 영상 처리 및 딥러닝 모델 결과의 후처리에 널리 사용된다.

본 프로젝트에서는 OpenCV를 활용하여 의류의 대표 색상을 분석하는 기능을 수행한다.

다. 시스템 구성도

의류 이미지로부터 Mask R-CNN과 OpenCV를 활용하여 의류의 카테고리 및 대표 색상을 분석하고, 이를 통해 의류 상세 데이터를 생성하는 시스템을 설계하였다. 전체 구성도와 그에 대한 설명은 다음과 같다.

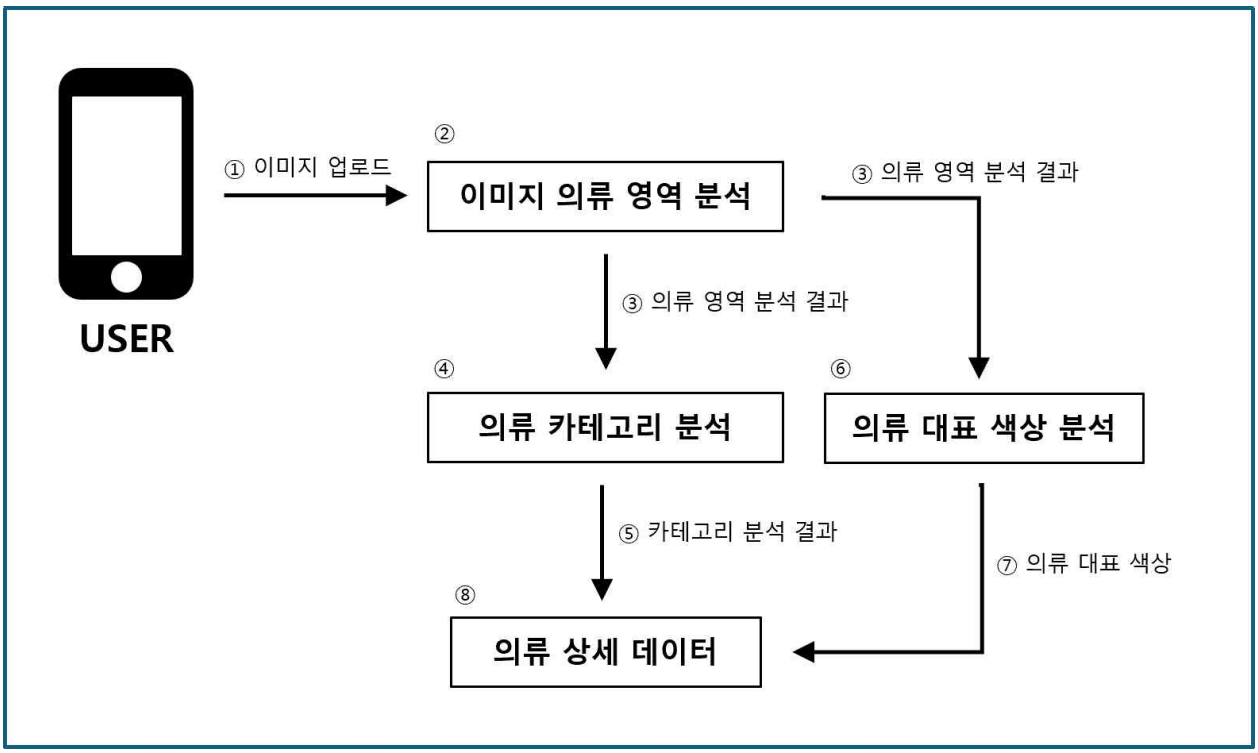

그림 3. 시스템 구성도

구분	설명
① 이미지 업로드	사용자가 의류 이미지를 촬영 및 등록하면 의류 이미지를 Back-end 서버로 전송한다.
② 이미지 의류 영역 분석	학습된 Mask R-CNN 딥러닝 모델을 이용하여 전송받은 의류 이미지를 분석하여 배경과 의류를 구분한다.
③ 의류 영역 분석 결과	예측된 결과를 각각 '의류 카테고리 분석' 모델과 '의류 대표 색상 분석' 모델로 전송한다.
④ 의류 카테고리 분석	전송받은 분석 이미지를 학습된 Mask R-CNN 딥러닝 모델로 분석하여 이 의류가 '티셔츠', '청바지' 등 어떤 카테고리의 의류인지 분석한다.
⑤ 카테고리 분석 결과	예측된 카테고리를 의류 상세 데이터에 저장한다.
⑥ 의류 대표 색상 분석	전송받은 분석 이미지를 OpenCV로 분석하여 해당 의류의 대표 색상이 무엇인지 예측한다.
⑦ 의류 대표 색상	예측된 대표 색상을 의류 상세 데이터에 저장한다.
⑧ 의류 상세 데이터	예측된 데이터를 활용하여 의류 객체를 정의한다.

표 1. 시스템 구성도 상세 설명

3. 설계 내용

가. MobileNetV2 모델을 사용한 의류 분류 모델 프로토타입 설계

본 프로젝트에서 Mask R-CNN 모델을 최종 적용할 예정이지만, 고정밀한 결과에 반해 느리고 무겁기 때문에 빠른 학습이 불가능하다. 이 때문에 초기 개발단계에서 보다 빠른 검증을 위해, 보다 가볍고 빠른 MobileNetV2 모델을 사용하여 의류 분류 모델 프로토타입을 개발하였다. 두 모델의 비교한 표는 아래와 같다.

항목	MobileNetV2	Mask R-CNN
모델 목적	이미지에서 객체를 탐지 + 분할	이미지에서 객체를 탐지 + 분할
출력	전체 이미지에 대한 하나의 클래스 분류	이미지 속 모든 객체의 위치, 클래스, 분할 마스크 (픽셀 단위)
주요 구성 요소	Depthwise Separable Convolution을 사용한 경량 CNN	Region Proposal Network (RPN) + ROIAlign + FCN 기반 마스크 예측
사용 분야	모바일 환경의 경량 이미지 분류	고정밀 객체 검출 + 분할 (예: 자율주행, 의류 인식 등)
복잡도/속도	매우 가볍고 빠름	느리고 무겁지만 고정밀
입력	하나의 이미지 ((224, 224, 3) 등)	하나의 이미지, 다양한 해상도 지원
프로젝트 적용 가능 범위	옷의 카테고리 분류(상의, 하의 등)	옷 탐지+분할: 어디에 어떤 옷이 있는지 마스크 포함 분석 가능

표 3. MobileNetV2와 Mask R-CNN 모델 비교

MobileNetV2 학습 모델은 사전 학습된 Backbone Model을 활용하여 보다 빠른 학습을 위한 모델로 설계하였다. 총 5개의 Layer로 구성되어 있으며, 각 Layer 계층에 대한 설명과 설계한 python 코드는 다음과 같다.

순서	레이어 이름	설명
1	Input Layer	• 이미지 크기: (224, 224, 3) • RGB 컬러 이미지 입력
2	Backbone Model	• MovileNetV2(include_top=False, weights='imagenet') • ImageNet에서 사전 학습된 CNN으로, 복잡한 이미지 특징을 추출하는 역할 • include_top=False로 설정했기 때문에 fully connected 분류 레이어는 제외됨
3	Global Average Pooling Layer	• 전체 feature map의 평균을 구해 벡터(1차원)로 변환 • 파라미터 수를 줄이고 과적합을 방지
4	Dropout Layer	• Dropout(0.3) • 30%의 뉴런을 무작위로 끔으로써 과적합 방지
5	Output Layer (Dense)	• Dense(num_classes, activation='softmax') • num_classes는 label_encoder.classes_의 길이, 즉 라벨 종류 개수 • softmax를 사용하여 각 클래스에 대한 확률 값 출력

표 4. MobileNetV2 학습 모델 Layer 설명

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras import layers, models

base_model = MobileNetV2(input_shape=(224, 224, 3),
                         include_top=False, weights='imagenet')
base_model.trainable = False  # 처음엔 고정

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(len(label_encoder.classes_), activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(train_ds, validation_data=val_ds, epochs=10)
```

코드 1. MobileNetV2를 사용한 의류 분류 학습 코드 설계

나. Mask R-CNN 모델을 사용한 의류 분류 모델 설계

본 프로젝트에서 최종적으로 적용할 Mask R-CNN 모델을 설계하였다. 모델의 Layer 구조는 아래 표 2와 같으며 일반적인 출력층 구조 대신 클래스 분류, 박스 회귀, 마스크 예측을 각각 별도의 출력 모듈에서 병렬로 수행한다. 이러한 병렬 수행으로 멀티태스크 학습의 효율성과 확장성을 동시에 확보 할 수 있는 장점을 가진다.

순서	레이어 이름	설명
1	Input Layer	• 이미지 크기: (224, 224, 3) • RGB 컬러 이미지 입력
2	Backbone Model	• ResNet-50 기반 CNN 구조 • 이미지에서 고수준 특징 추출 • Feature Pyramid Network(FPN)을 통해 다양한 크기의 feature map 생성
3	Region Proposal Network (RPN)	• backbone의 feature map을 기반으로 물체가 있을 법한 영역(bounding box)을 제안 • 다양한 anchor를 사용하여 위치 후보 생성
4	RoIAlign Layer	• 제안된 영역(Region of Interest)을 고정된 크기로 정렬 • 픽셀 정확도를 유지하며 feature를 추출
5	Box Head (Fully Connected)	• RoI에 대한 분류(classification)과 박스 회귀(bbox regression)를 수행 • 2-layer MLP (FC + ReLU × 2)
6	Box Predictor	• cls_score: 각 RoI에 대해 클래스 확률을 출력 • bbox_pred: 각 RoI에 대해 bbox 위치를 조정한 좌표 출력
7	Mask Head (Convolutional)	• 4개의 Conv layer + ReLU • RoI feature를 마스크 feature로 변환
8	Mask Predictor	• 1×1 Conv layer • 각 클래스마다 마스크(28×28 해상도)를 출력 • shape: (N, num_classes, 28, 28)

표 5. Mask R-CNN 학습 모델 Layer 설명

위의 Layer 설계를 바탕으로 Mask R-CNN 모델을 학습시키는 Python 코드를 작성하였다. 학습은 에포크 수 1회, 훈련 데이터 500개로 제한하여 진행하였는데, 이는 Mask R-CNN 모델의 구조가 복잡하고 실행 시간이 길기 때문이다. 본 프로젝트에서는 Google Colab 무료 버전을 사용하여 학습을 수행하였으며, 해당 환경에서는 일반적으로 최대 6시간 이내의 세션 제한이 존재한다. 실제로 에포크 1회 학습하는 데에도 약 4시간 이상이 소요되므로, 현재 환경에서는 더 많은 데이터를 사용하거나 에포크 수를 늘리는 데에 제약이 따른다. 따라서 향후에는 Colab Pro 등의 유료 서비스를 가입하는 등 실행 환경을 개선하여 더 긴 학습 시간 확보가 필요하다. 작성한 학습 코드는 다음과 같다.

```python
import os, torch, tqdm, torchvision, numpy as np, pandas as pd, torchvision.transforms as transforms
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision.models.detection import maskrcnn_resnet50_fpn

# 경로 설정
image_dir = "/kaggle/input/clothing-dataset-full/images_compressed"
csv_file = "/kaggle/input/clothing-dataset-full/images.csv"
class_map = get_class_mapping(csv_file) # 라벨 매핑

# Transform
transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.ToTensor()
])

# Dataset 구성
full_dataset = ClothingDataset(image_dir, csv_file, class_map, transforms=transform)

# 500개 데이터만 사용
subset_size = 500
dataset = torch.utils.data.Subset(full_dataset, range(subset_size))

# 배치 사이즈 1
data_loader = DataLoader(dataset, batch_size=1, shuffle=True, collate_fn=lambda x: tuple(zip(*x)))

# 모델 초기화
device = torch.device('cuda'if torch.cuda.is_available() else 'cpu')
model = maskrcnn_resnet50_fpn(num_classes=len(class_map)+1)
model.to(device)

# Optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.Adam(params, lr=0.0005)

# 학습 루프
num_epochs = 1 # epoch = 1번
for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0
    for images, targets in tqdm.tqdm(data_loader, desc=f"Epoch {epoch+1}"):
        images = list(img.to(device) for img in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())
```

```
47.
48.             optimizer.zero_grad()
49.             losses.backward()
50.             optimizer.step()
51.
52.             epoch_loss += losses.item()
53.         print(f"Epoch {epoch+1} Loss: {epoch_loss:.4f}")
54.
55.     # 모델 저장
56.     torch.save(model.state_dict(), "maskrcnn_clothing.pth")
```

코드 2. Mask R-CNN을 사용한 의류 분류 학습 코드 설계

다. 데이터베이스 설계

본 프로젝트에서 사용자의 정보를 저장하고 사용자 정보, 의류 데이터, 코디 구성, 일정 및 날씨 정보를 효율적으로 관리하기 위해 관계형 데이터베이스를 사용하였다. 관계형 데이터베이스 설계 도구 중 MySQL Workbench를 사용하여 EER 다이어그램을 설계하였으며 총 7개의 테이블로 구성되하였다. 사용자 중심의 추천 기능을 지원하도록 각 테이블 간 연관관계를 설정하였다. 설계한 EER 다이어그램과 각 테이블의 상세 설명은 다음과 같다.

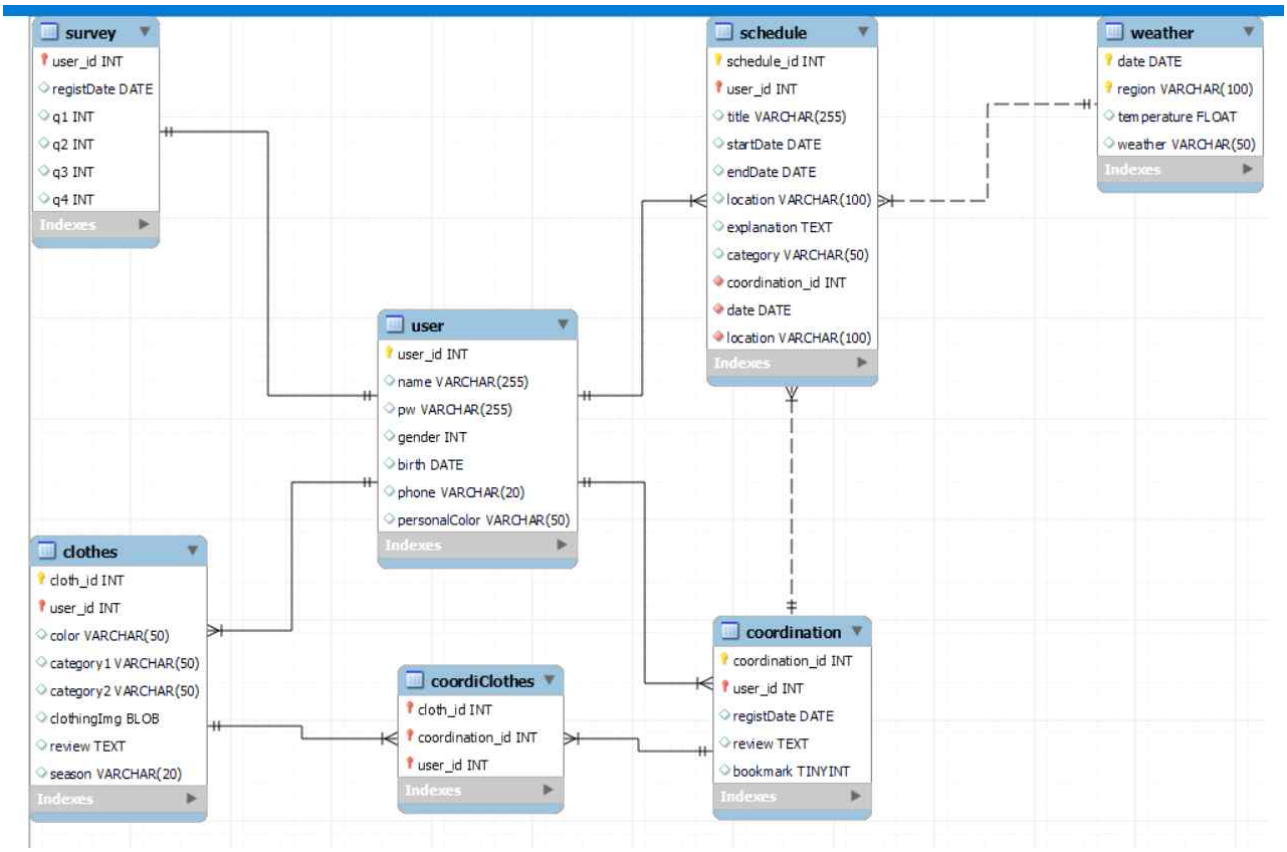

그림 5. EER 다이어그램

테이블	속성	설명
user	user_id, name, pw, birth, phone, personalColor	• 사용자 정보 테이블 • ID, 생년월일, 성별, 퍼스널컬러 등 사용자의 개인정보 저장.
clothes	cloth_id, user_id(FK), color, category1 ~ 2, clothImg, review, season	• 사용자가 등록한 옷의 정보 테이블 • 상의/하의 등 카테고리 및 계절 속성 저장
coordination	coordination_id, user_id(FK), review, bookmark	• 코디 정보 저장. • 사용자가 구성한 코디 정보 및 리뷰, 즐겨찾기 속성 저장
corrdiClothes	coordination_id(FK), cloth_id(FK), user_id(FK)	• 코디에 포함된 옷 정보 연결 테이블 • 코디와 의류 간 다대다 관계 연결
schedule	schedule_id, user_id(FK), startDate, endDate location, explanation, category, oordination_id(FK), date(FK), regon(FK)	• 사용자의 일정 정보. • 코디 추천 시 참고되는 장소, 카테고리 속성 저장.
weather	date, region, weather, temperature	• 날씨 정보 테이블 • 지역, 날짜의 기온과 기상상태를 저장 • 일정의 날씨 데이터를 연동해 코디 추천에 활용 가능.
survey	user_id(FK), q1 ~ q4, registDate	• 사용자 선호 설문 결과 저장. • 맞춤형 코디 추천에 활용. • user 테이블과 연결.

표 6. 데이터베이스 스키마 설계 설명

4. 구현 결과

가. MobileNetV2 모델을 사용한 의류 분류 모델 결과 예측 결과 코드

MobileNetV2를 사용하여 학습하는 코드(코드 1 참조)를 실행하여 이미지와 의류 카테고리의 연관성을 학습시키고 matplotilib 라이브러리를 활용하여 예측한 결과 시각화하였다. 다음은 시각화한 코드와 출력 결과이다.

```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# 테스트 이미지 경로
test_image_path = '/content/drive/MyDrive/Colab Notebooks/캡스톤디자인/캡처.JPG'
predicted_label = predict_image(test_image_path)

# 이미지 출력
img = mpimg.imread(test_image_path)

plt.imshow(img)
plt.axis('off')
plt.title(f"Predicted: {predicted_label}")
plt.show()

print("예측된 의류 종류:", predicted_label)
```

코드 3. MobileNetV2를 사용한 의류 분류 모델 예측 결과 시각화 코드

그림 6. MobileNetV2를 사용한 의류 분류 모델 예측 결과

위 예측 결과(그림 6 참조)를 보면 왼쪽부터 각각 스커트(Skirt), 청바지(Pants), 반팔 티셔츠(T-Shirt) 이미지를 입력으로 사용하였으며, 모델은 각 이미지에 대해 해당 의류 카테고리를 정확하게 예측한 것을 알 수 있다. 이러한 경량화 모델을 활용하여 초기 개발 단계에서 보다 빠르고 정확한 검증을 통해 의류 분류 기능의 기본 구조를 테스트할 수 있고, 복잡하고 연산량이 많은 고정밀 모델(Mask R-CNN)을 적용하기 전, 모델 설계 방향성과 데이터셋 구성의 적절성을 확인하는 데 중요한 기반이 되었다.

나. Mask R-CNN 모델을 사용한 의류 분류 모델 결과 예측 결과 코드

Mask R-CNN을 사용하여 학습하는 코드(코드2 참조)를 실행하여 이미지와 의류 카테고리의 연관성을 학습시키고 matplotilib 라이브러리를 활용하여 예측한 결과 시각화하였다. 다음은 시각화한 코드와 출력 결과이다.

```
import torch, torchvision, cv2, os, numpy as np, matplotlib.pyplot as plt
from torchvision.transforms import functional as F
from collections import OrderedDict

# 설정: 경로 및 클래스 매핑
model_path = "maskrcnn_clothing.pth"  # 학습된 모델 경로
test_image_path = "(예측할 이미지 경로)"

# 학습에 사용된 라벨 맵
class_map = {
    'Blazer': 0, 'Blouse': 1, 'Body': 2, 'Dress': 3, 'Hoodie': 4,
    'Jacket': 5, 'Jeans': 6, 'Pants': 7, 'Polo': 8, 'Shorts': 9,
    'Skirt': 10, 'Sweater': 11, 'Tank': 12, 'Tee': 13, 'Top': 14,
    'Tshirt': 15
}
idx_to_class = {v: k for k, v in class_map.items()}

# 모델 정의 및 weight 불러오기
device = torch.device('cuda'if torch.cuda.is_available() else 'cpu')
num_classes = len(class_map) + 1  # +1 for background

# 현재 클래스 수에 맞는 모델 구조
model = torchvision.models.detection.maskrcnn_resnet50_fpn(num_classes=num_classes)

# 체크포인트 전체 로드
checkpoint = torch.load(model_path, map_location=device)

# head 제외하고 backbone만 로드
filtered_state_dict = OrderedDict()
for k, v in checkpoint.items():
    if not (
        "roi_heads.box_predictor"in k or
        "roi_heads.mask_predictor"in k
    ):
        filtered_state_dict[k] = v
model.load_state_dict(filtered_state_dict, strict=False)
model.to(device)
model.eval()

```

```
# 이미지 로드 및 예측
image = Image.open(test_image_path).convert("RGB")
image_tensor = F.to_tensor(image).to(device)

with torch.no_grad():
    prediction = model([image_tensor])[0]

# 결과 시각화
image_np = np.array(image).copy()
for i in range(len(prediction["boxes"])):
    score = prediction["scores"][i].item()
    if score < 0.5:
        continue  # 낮은 확률은 건너뜀
    box = prediction["boxes"][i].cpu().numpy().astype(np.int32)
    label_id = prediction["labels"][i].item()
    mask = prediction["masks"][i, 0].cpu().numpy()

    # 경계 상자 그리기
    cv2.rectangle(image_np, (box[0], box[1]), (box[2], box[3]), (0, 255, 0), 2)
    class_name = idx_to_class.get(label_id - 1, "Unknown")  # background 제외
    cv2.putText(image_np, f"{class_name}{score:.2f}", (box[0], box[1] - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # 마스크 시각화
    mask_bin = (mask > 0.5).astype(np.uint8)
    color_mask = np.zeros_like(image_np)
    color_mask[mask_bin == 1] = [0, 255, 0]
    image_np = cv2.addWeighted(image_np, 1.0, color_mask, 0.4, 0)

# 출력
plt.figure(figsize=(8, 8))
plt.imshow(image_np)
plt.axis('off')
plt.title("Predicted Clothing with Mask R-CNN")
plt.show()
print("📢 예측된 클래스:")
for i in range(len(prediction["boxes"])):
    score = prediction["scores"][i].item()
    if score < 0.5:
        continue  # 낮은 확률은 제외

    label_id = prediction["labels"][i].item()
    class_name = idx_to_class.get(label_id - 1, "Unknown")  # backgrou코드 3. Mask R-CNN을 사용한 의류 분류 모델 예측 결과 시각화 코드nd 제외
    print(f"- {class_name} (score: {score:.2f})")
```

코드 4. Mask R-CNN을 사용한 의류 분류 모델 예측 결과 시각화 코드

그림 7. Mask R-CNN을 사용한 의류 분류 모델 예측 결과

그림 7의 결과를 보면, 원본 이미지는 'Shirts'임에도 불구하고 'T-shirts'로 잘못 분류된 것을 확인할 수 있다. 이는 설계 단계에서 실행 시간 단축을 위해 학습 데이터를 500개로 제한하고 에포크를 1회만 수행한 결과로, 충분한 학습이 이루어지지 않아 예측 정확도가 낮은 것으로 판단된다.

향후 실행 시간 문제를 해결한 후, 학습 데이터의 양과 에포크 수를 증가시켜 모델의 예측 성능을 개선할 계획이다.

5. 결론

본 프로젝트는 딥러닝 기반 기술을 활용하여 사용자에게 상황에 맞는 코디를 추천하는 어플리케이션을 개발하는 것을 목표로 수행되었다. 이를 통해 외출 전 의류 선택에 대한 고민을 줄이고, 사용자 스타일 만족도를 높이며, 의류 소비의 효율성을 향상시키고자 하였다.

초기 개발 단계에서는 빠른 기능 검증을 위해 MobileNetV2를 활용한 경량 모델로 의류 분류 기능을 구현하였으며, 이후 최종 적용할 학습 모델인 Mask R-CNN과 OpenCV를 활용하여 고정밀 객체 인식 및 색상 분석이 가능한 구조를 시험적으로 적용하였다. 그러나 Mask R-CNN 모델의 높은 연산량으로 인해 실행 시간이 길어졌고, Google Colab 무료 버전의 실행 시간 제한으로 인해 학습 데이터 양과 에포크 수를 제한적으로 적용할 수밖에 없었다. 이로 인해 예측 정확도가 낮은 결과가 도출되었으며, 이는 모델 성능 개선을 위한 주요 과제로 확인되었다. 향후에는 보다 효율적인 학습 알고리즘 설계와 안정적인 학습 환경 확보를 통해, 학습 데이터와 에포크 수를 충분히 늘려 모델의 예측 성능을 고도화할 예정이다.

이처럼 의류 분류 모델을 고도화시키고 이를 활용하여 일정 및 날씨 정보와 사용자 피드백 기반의 맞춤형 추천 기능을 추가로 구현할 계획이다. 특히 사용자가 착용한 코디에 대한 평가 데이터와 설문 결과를 반영하여 개인의 취향을 학습하는 방식으로, 더욱 정교하고 사용자 맞춤화된 코디 추천이 가능하도록 시스템을 고도화할 예정이다. 또한, 현재 데스크탑 기반에서 개발된 구조를 모바일 환경에서도 활용할 수 있도록 경량화 및 연산 최적화를 적용하여, 일상 속에서 실시간으로 활용 가능한 실용적인 패션 추천 서비스로 확장하는 것을 궁극적인 목표로 한다.

본 프로젝트를 통해 구축된 딥러닝 기반 의류 분류 및 자동 코디 추천 시스템은 사용자 중심의 의류 관리 및 추천 서비스의 가능성을 제시하며, 향후 지속적인 기능 확장과 성능 개선을 통해 패션 분야의 디지털 전환을 이끄는 핵심 기술로 발전할 수 있을 것으로 기대된다.

6. 참고 문헌

[1] He, K., Gkioxari, G., Dollar, P., and Girshick, R., "Mask R-CNN," IEEE International Conference on Computer Vision (ICCV), pp. 2961-2969, Oct. 2017.

[2] Bradski, G., "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.