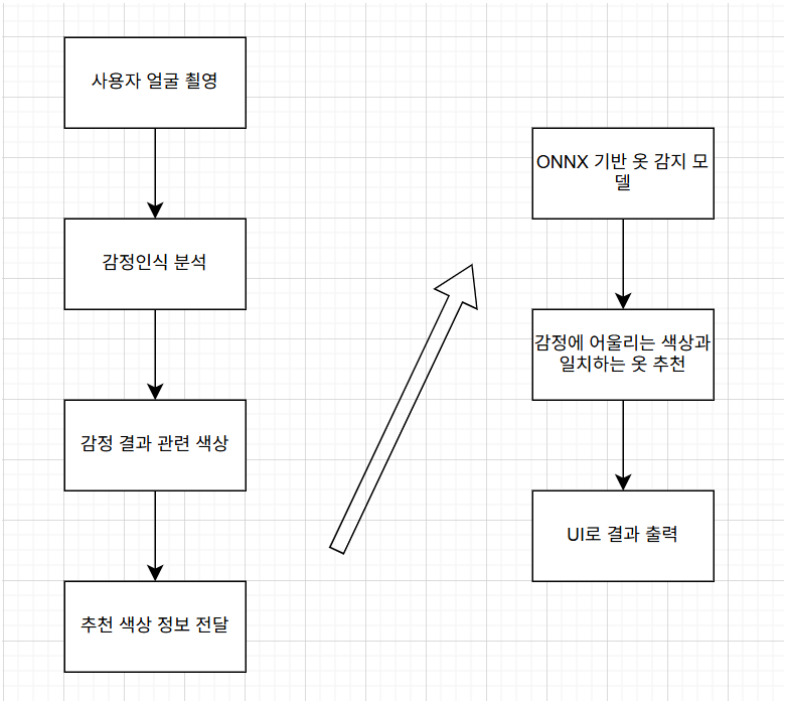
1. 팀원별 역할 (PO6)

팀원 A	감정 인식 모델 학습 및 최종 병합
팀원 B	ONNX 기반 의류 감지, 실시간 영상 처리
팀원 C	감정-의류 매핑, 추천 로직 및 UI 구현

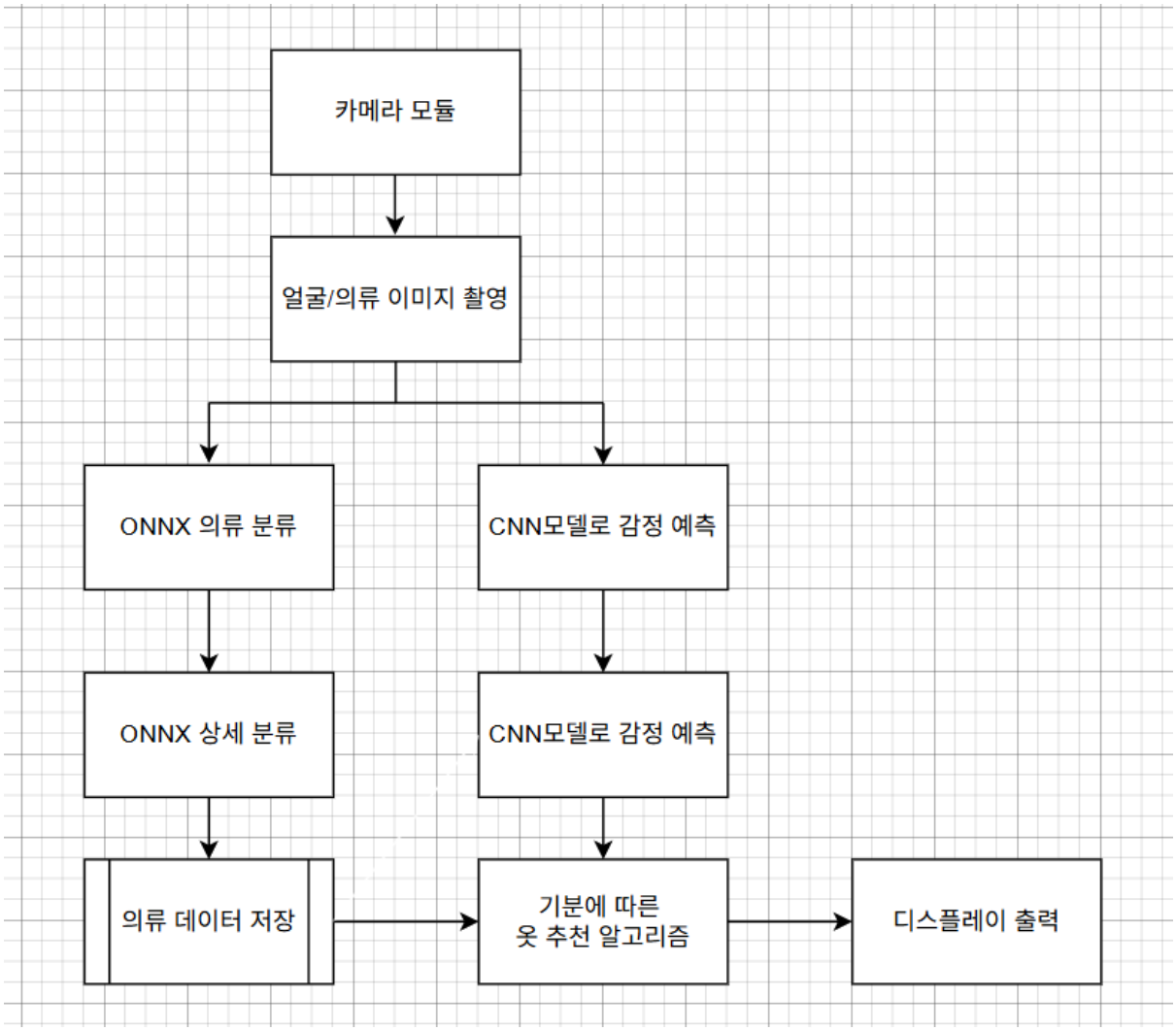
2. 개발일정

1~3주차	프로젝트 목표 수립, 역할 분담, 요구사항 정의
4~6주차	감정 인식 모델 학습 및 의류 감지 모델 준비
7~9주차	감정 인식 결과 → 감정 결과 색상 매핑 → 의류 감지 연동
10~12주차	감정 기반 색상 필터링 기능 완성, UI 정리
13~15주차	프로젝트 마무리, 보고서 및 발표자료 작성

3. 목적계통도



4. 기능블록도 및 작품의 요약 설명



5. 설계제한요소 (PO3)

*주의: 작품 설계시에 제약을 받거나 제한을 둔 내용 (즉 설계사양)을 기재. 4개 이상 기재할 것

제한요소	내용 (설계사양)
경제성	일반 PC 기반으로 구현하여 비용 절감
안전성	UI는 감정 결과만 보여주고 사용자 정보는 저장하지 않으며, 버튼 조작은 간단한 터치로 구현되어 사용 중 혼란 없음
신뢰성	감정 인식은 5초 평균 결과로 처리하고, 옷 추천 결과는 터치 UI에서 명확히 확인 가능함
미관	UI는 터치에 맞게 버튼 크기와 텍스트 가독성을 높였고, 직관적이고 깔끔한 배치 적용
윤리성	감정 분석 결과를 판단이 아닌 추천 목적으로만 사용하며, 결과는 사용자에게만 표시됨
사회적 영향	터치 UI까지 통합되어 고령자, 어린이 등 디지털 기기에 익숙하지 않은 사용자도 쉽게 접근 가능
기능성	실시간 감정 분석 + 추천 옷 필터링 + 터치 UI 통합으로 최종 사용자 친화형 시스템 완성

6. 사용할 소프트웨어, 하드웨어 목록 및 사용 내용 요약 (PO5)

구분	항목	사용 목적
소프트웨어	Python 3.12	전체 시스템 개발
소프트웨어	PyTorch	감정 인식 모델 실행
소프트웨어	ONNXRuntime	옷 감지 ONNX 모델 실행
소프트웨어	OpenCV	웹캠 제어 및 이미지 처리
소프트웨어	Tkinter	터치 기반 UI 구현 및 감정 결과 + 추천 옷 표시
하드웨어	웹캠	얼굴 및 옷 실시간 촬영
하드웨어	터치 모니터 UI	결과 확인 및 간단 조작 가능
하드웨어	노트북/PC	전체 시스템 구동

7. 알고리즘 설명을 포함한 작품의 상세설명

---------- 모델 학습 ----------

분류 표정 종류

[default, angry, sad, happy]

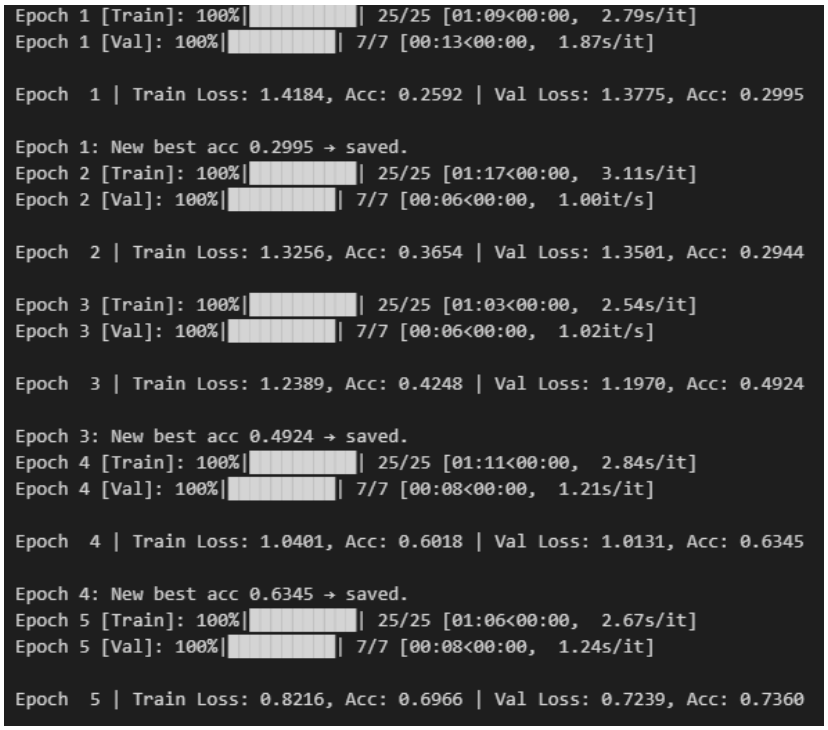
표정 데이터셋 수집

- 구글 이미지 긁어오기
- 직접 촬영한 표정 사진 다수

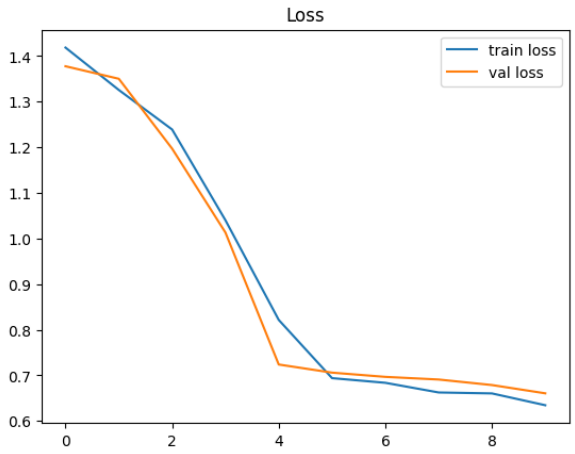
학습 방법

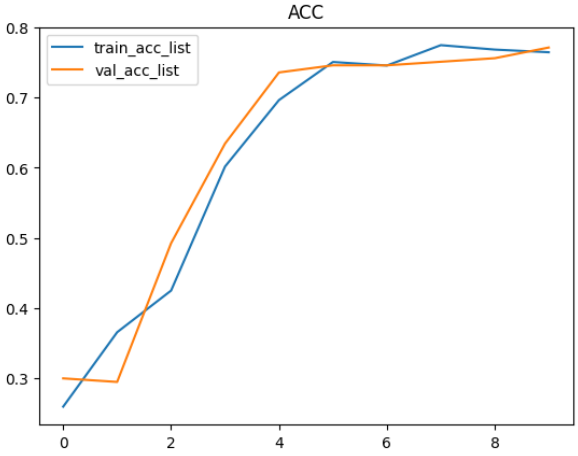
- **데이터 증강:** 회전, 색상/명암/채도/색조, 블러, 노이즈. 수평 플립, 평균 정규화
- **이미지넷을 학습한 resnet18을 불러와 파라미터 재학습**
- **출력층을 수에 맞게 교체**

학습 결과



```
Epoch 6 [Train]: 100%|██████████| 25/25 [01:37<00:00,  3.88s/it]
Epoch 6 [Val]: 100%|██████████| 7/7 [00:07<00:00,  1.11s/it]

Epoch  6 | Train Loss: 0.6940, Acc: 0.7509 | Val Loss: 0.7060, Acc: 0.7462

Epoch 6: New best acc 0.7462 → saved.
Epoch 7 [Train]: 100%|██████████| 25/25 [01:10<00:00,  2.80s/it]
Epoch 7 [Val]: 100%|██████████| 7/7 [00:06<00:00,  1.10it/s]

Epoch  7 | Train Loss: 0.6838, Acc: 0.7459 | Val Loss: 0.6968, Acc: 0.7462

Epoch 8 [Train]: 100%|██████████| 25/25 [00:59<00:00,  2.38s/it]
Epoch 8 [Val]: 100%|██████████| 7/7 [00:06<00:00,  1.03it/s]

Epoch  8 | Train Loss: 0.6626, Acc: 0.7750 | Val Loss: 0.6910, Acc: 0.7513

Epoch 8: New best acc 0.7513 → saved.
Epoch 9 [Train]: 100%|██████████| 25/25 [01:15<00:00,  3.03s/it]
Epoch 9 [Val]: 100%|██████████| 7/7 [00:09<00:00,  1.31s/it]

Epoch  9 | Train Loss: 0.6605, Acc: 0.7686 | Val Loss: 0.6789, Acc: 0.7563

Epoch 9: New best acc 0.7563 → saved.
Epoch 10 [Train]: 100%|██████████| 25/25 [01:04<00:00,  2.58s/it]
Epoch 10 [Val]: 100%|██████████| 7/7 [00:06<00:00,  1.04it/s]

Epoch 10 | Train Loss: 0.6346, Acc: 0.7649 | Val Loss: 0.6607, Acc: 0.7716

Epoch 10: New best acc 0.7716 → saved.
```

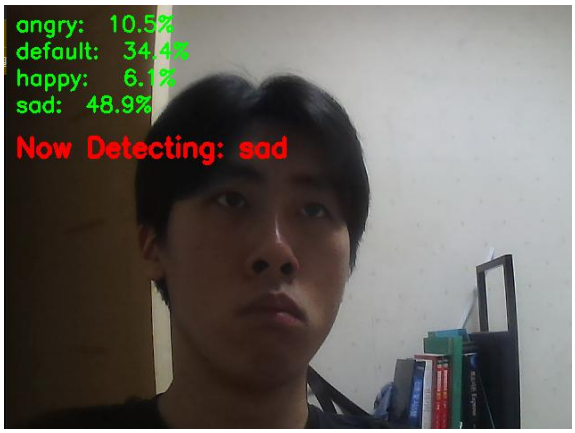




---------- 감정 인식 ----------

- 카메라로부터 실시간 스트리밍 표정 이미지 데이터를 얻는다.
- 5 초 동안 감정분석을 진행한다.

- 가장 높은 정확도의 감정을 선택한다.



---------- 옷 추천 ----------

감정 인식 결과에 따라 사용자에게 맞는 옷을 추천해 준다.

아래는 리스트는 옷 인식 시스템이 인식 가능한 옷 들이다.

상의 종류 - '셔츠', '스웨터, 니트' ,'맨투맨, 후드티', '반팔 티셔츠'

하의 종류 - '청바지', '반바지', '슬렉스, 면바지', '스웨트 팬츠', '카고'

아우터 종류 - '점퍼', '패딩', '바람막이', '코트', '후리스', '자켓', '없음'

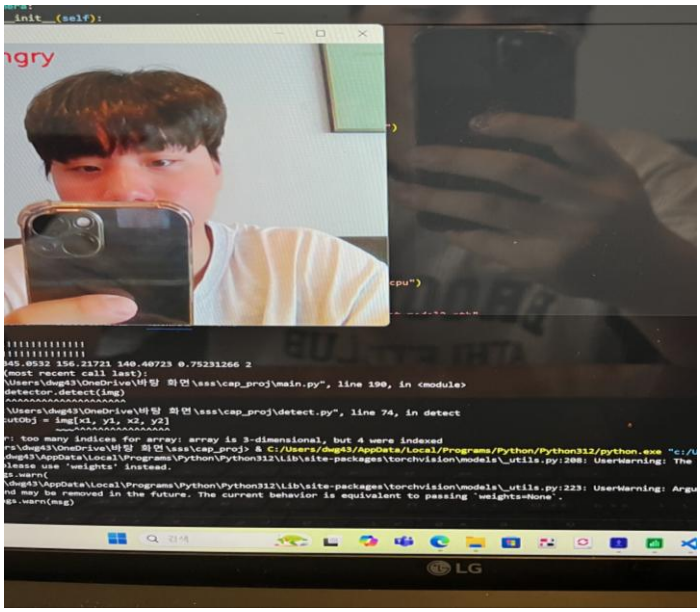
	id	filename	emotion	style	description	in_stock
2	2.41E+11	1.jpg	happy	shorts	연베이지	TRUE
3	2.41E+11	2.jpg	sad	pants	그레이 트	TRUE
4	2.41E+11	3.jpg	angry	outer	가죽 자켓	TRUE
5	2.41E+11	4.jpg	angry	pants	검정 데님	TRUE
6	2.41E+11	5.jpg	happy	top	회색 맨투	TRUE
7	2.41E+11	6.jpg	default	shirt	기본 흰 셔	TRUE
8	2.41E+11	7.jpg	happy	shirt	하늘색 셔	TRUE
9	2.41E+11	8.jpg	sad	shirt	검정 셔츠	TRUE
10	2.41E+11	9.jpg	sad	shirt	네이비 셔	TRUE
11	2.41E+11	10.jpg	default	shirt	크림색 셔	TRUE
12	2.41E+11	11.jpg	sad	top	와인색 맨	TRUE
13	2.41E+11	12.jpg	happy	top	연두색 크	TRUE
14	2.41E+11	13.jpg	angry	outer	블랙 패딩	TRUE
15	2.41E+11	14.jpg	default	knit	브라운 꽈	TRUE
16	2.41E+11	15.jpg	angry	pants	블랙 조거	TRUE
17	2.41E+11	17.jpg	happy	pants	연청 데님	TRUE
18	2.41E+11	18.jpg	default	pants	카키 면바	TRUE
19	2.41E+11	19.jpg	default	outer	그레이 카	TRUE
20	2.41E+11	20.jpg	default	outer	블랙 청자	TRUE
21						

8. 제작과정 (사진첨부)

1. 감정 인식 모델 개발

CNN 기반 감정 분류 모델 직접 설계 및 학습

표정 데이터를 통해 기쁨, 슬픔, 분노 등 감정 분류



2. 의류 분류 시스템 연동

기존 ONNX 기반 의류 분류 모델 재활용

감정에 맞는 의류 스타일 매칭 알고리즘 구현

3. 통합 시스템 구축 (Python 기반)

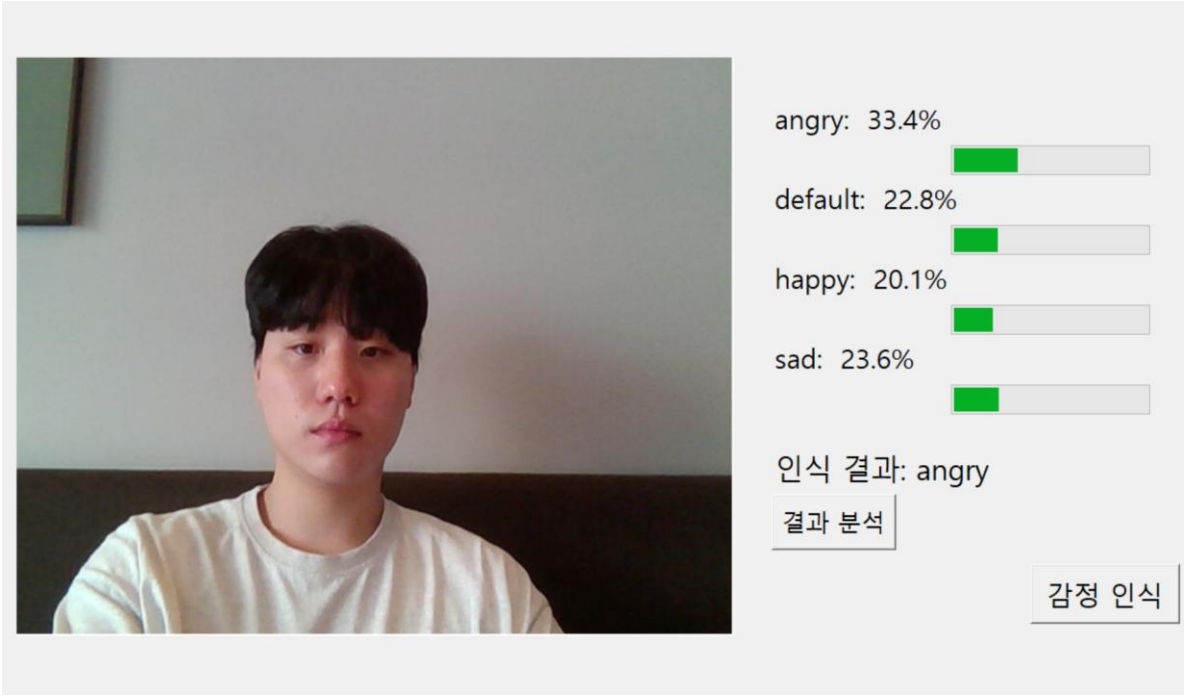
웹캠 입력 → 감정 분석 → 의류 추천까지 자동화

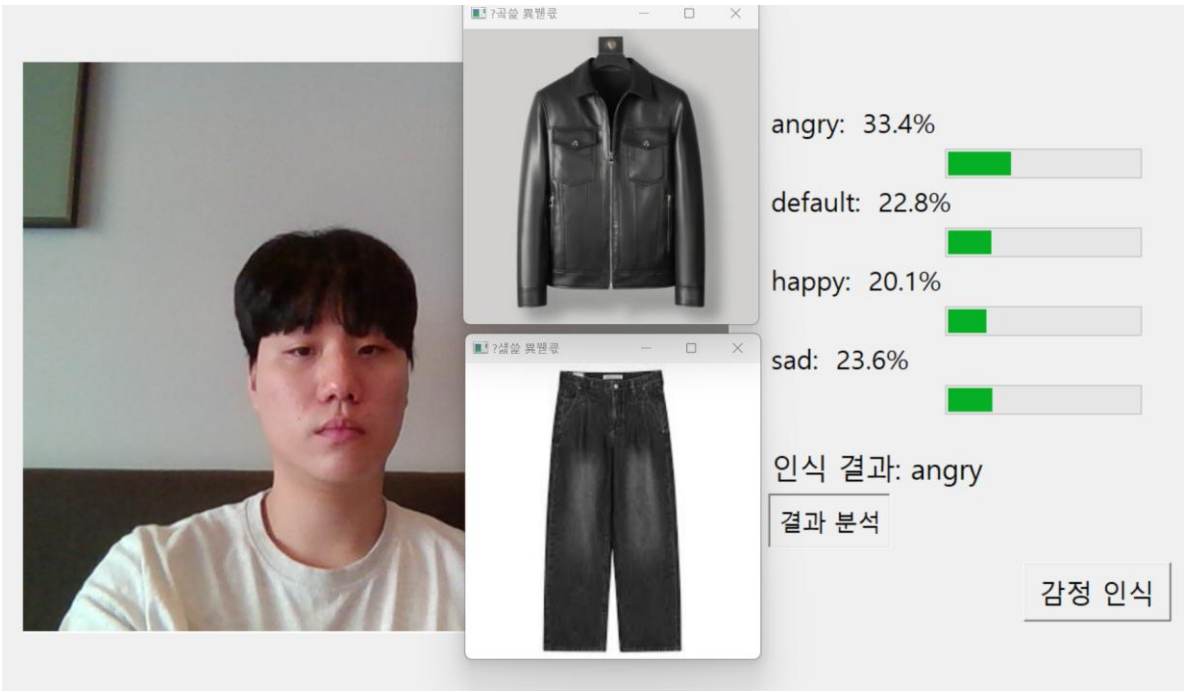
추천 결과 이미지 및 설명 형태로 출력

4. 테스트

다양한 감정 및 상황별 테스트 진행

감정-의류 매칭 정확도 및 사용자 만족도 개선





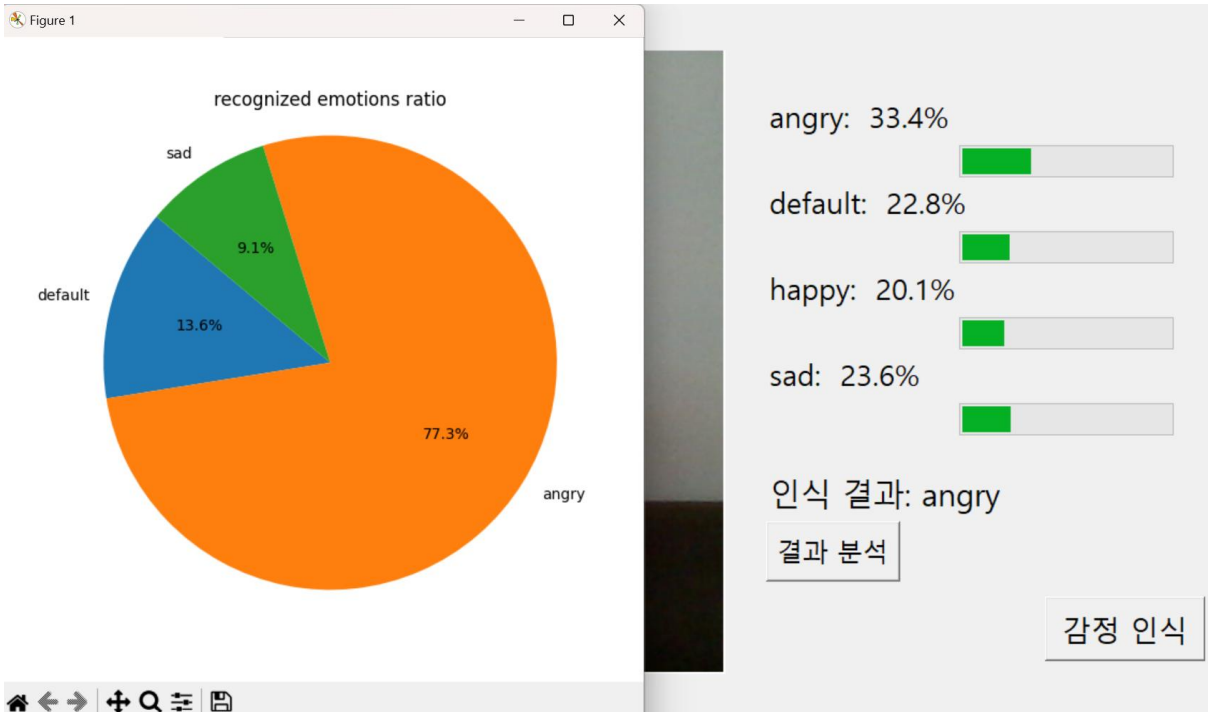
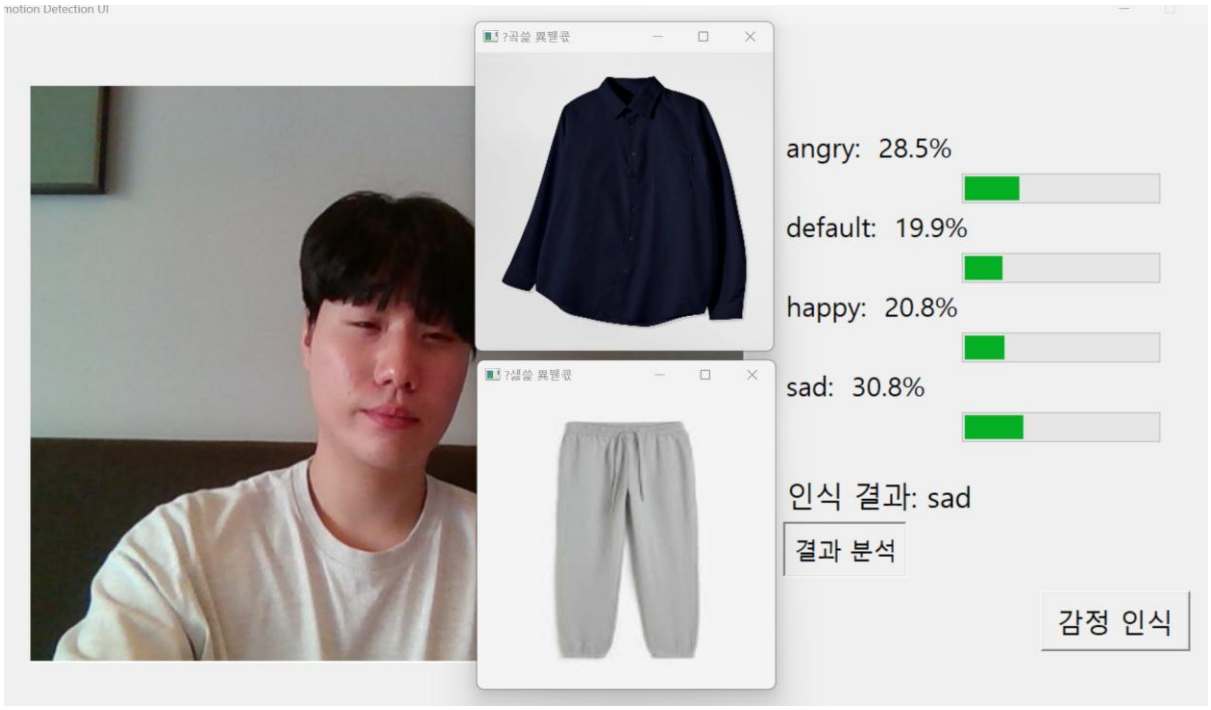
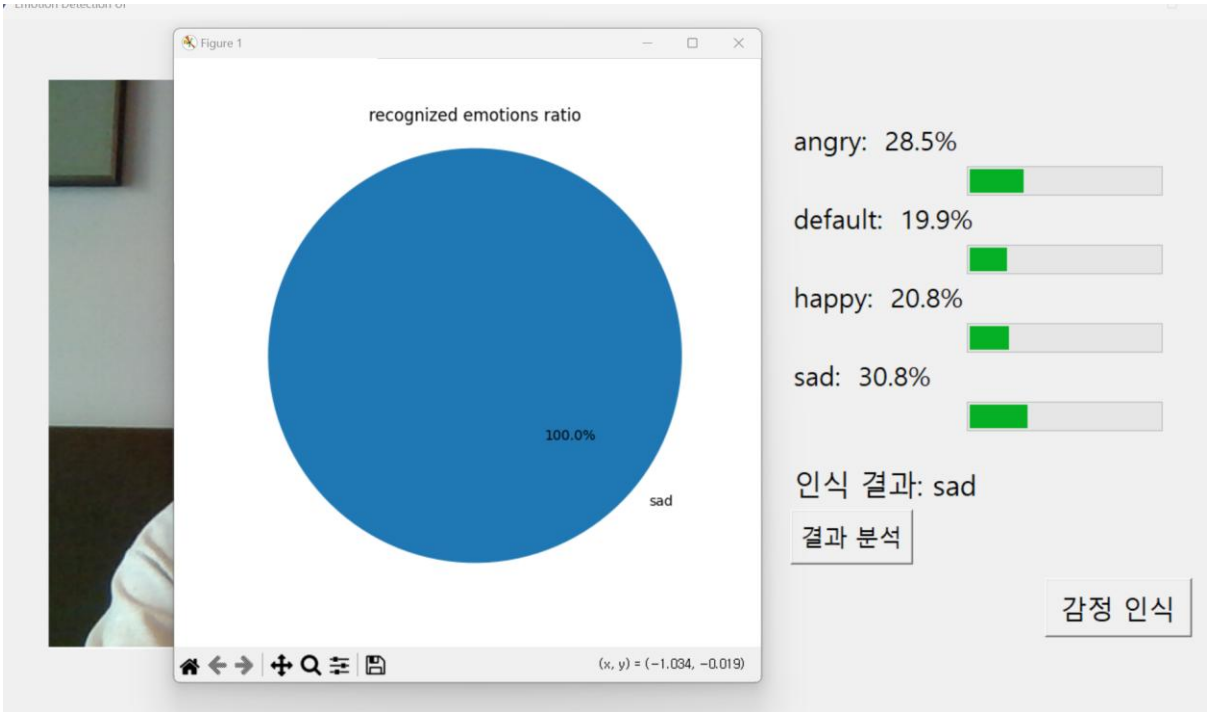


Figure 1
recognized emotions ratio
sad
9.1%
default
13.6%
77.3%
angry
angry: 33.4%
default: 22.8%
happy: 20.1%
sad: 23.6%
인식 결과: angry
결과 분석
감정 인식


angry: 28.5%
default: 19.9%
happy: 20.8%
sad: 30.8%
인식 결과: sad
결과 분석
감정 인식



9. 제작 후 자체평가 및 소감

감정이라는 주관적 요소를 기술적으로 다루는 데 어려움이 있었지만, 직접 감정 인식 모델을 설계하고 추천 알고리즘과 연동하는 과정을 통해 딥러닝과 응용 개발에 대한 실질적인 경험을 쌓을 수 있었습니다. 또한 기존 프로젝트와의 통합을 통해 AI 시스템 개발의 확장성과 응용 가능성을 확인하였으며, 향후 사용자 맞춤형 서비스가 나아갈 방향에 대해 고민해보는 뜻깊은 시간이었습니다.

무엇보다 사용자 경험의 중요성을 다시 한번 체감할 수 있었던 프로젝트였습니다. 단순히 기술을 구현하는 것에 그치지 않고, 사용자가 실제로 느끼는 감정과 어울리는 스타일을 추천하는 '감성 기반 인터페이스'를 고민해야 했기 때문에, 기술과 감성의 접점을 찾는 과정이 인상 깊었습니다. 감정과 패션이라는 이질적인 두 요소를 하나의 시스템으로 통합하는 과정에서 창의적인 사고와 협업의 중요성을 크게 느꼈습니다.

또한 이번 프로젝트를 통해 머신러닝 모델 학습 과정에 대한 깊은 이해와 모델 성능 개선에 필요한 데이터 구성 및 전처리의 중요성을 체득할 수 있었습니다. 학습 정확도를 높이기 위해 감정 데이터셋을 수집하고 분석하는 과정, 그리고 추천 정확도를 높이기 위한 조건 설계 등의 실험을 반복하면서, AI 개발에 있어 정확한 데이터 설계와 모델 구조의 선택이 얼마나 중요한지를 실감하게 되었습니다.

마지막으로, 기존 캡스톤 프로젝트의 결과물을 재활용하고 발전시켜 새로운 가치를 창출해낸 경험은 매우 의미 있었습니다. 하나의 프로젝트가 끝이 아니라, 다음 프로젝트의 기반이 될 수 있다는 점에서 지속 가능한 개발과 융합적 사고를 배우는 계기가 되었습니다. 앞으로도 기술과 사용자의 삶을 연결할 수 있는, 실질적인 가치를 지닌 AI 시스템을 고민하고 구현해 나가고 싶습니다.

10. 참고자료 목록 (PO2)

11. 회의록

1. 진행 사항 공유

감정 인식 기능: CNN 기반 모델로 감정(기쁨, 슬픔, 분노 등) 분류 구현 완료

의류 추천 기능: 기존 캡스톤 프로젝트의 ONNX 기반 의류 분류 모델 성공적으로 연동

시스템 통합: 웹캠 입력 → 감정 분석 → 의류 추천까지 자동화된 파이프라인 구성

실시간 출력: 추천 결과를 이미지 또는 텍스트로 출력하는 기능 구현 완료

2. 포스터 항목 구성 확정

선정 배경 및 필요성

기술적 필요성 (간결 문단 작성 완료)

경제적 필요성 / 사회적 필요성

작품 설명 (한 문장 요약 + 상세본 별도 정리)

제작 과정 (단계별 기술 흐름 정리)

기대 효과

자체 평가 및 소감

3. 논의된 주요 사항

포스터 문구는 최대한 간결하게 정리하되, 구체적 내용은 부가 자료로 보완

시스템 흐름도를 포스터에 추가하여 시각적으로 설명

추천 결과 예시 이미지 준비 여부 검토

소스코드(파이썬)

데이터 증가

```
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, random_split
train_transform = transforms.Compose([
    transforms.Resize((300, 300)),
    # 2) 회전 및 기하 변형
    transforms.RandomRotation(degrees=15),
    transforms.RandomAffine(degrees=0, translate=(0.1,0.1), scale=(0.9,1.1), shear=10),
    transforms.RandomPerspective(distortion_scale=0.2, p=0.5),
    # 3) 색상/명암/채도/색조 변화
    transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.5),
```

```
    # 4) 블러, 노이즈
    transforms.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
    # 5) 수평/수직 뒤집기
    transforms.RandomHorizontalFlip(p=0.5),

    # 6) 텐서 변환 & 정규화
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485,0.456,0.406],
                         std =[0.229,0.224,0.225]),
    # 7) Random Erasing (텐서 적용 후)
    transforms.RandomErasing(p=0.3, scale=(0.02,0.15), ratio=(0.3,3.3)),
])
val_transform = transforms.Compose([
    transforms.Resize((300, 300)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485,0.456,0.406],
                         std =[0.229,0.224,0.225]),
])
data_dir = "./emotion_data"
full_ds = datasets.ImageFolder(root=data_dir, transform=train_transform)
# 3) Train/Val 분할
val_ratio = 0.2
val_size = int(len(full_ds) * val_ratio)
train_size = len(full_ds) - val_size
train_ds, val_ds = random_split(full_ds, [train_size, val_size])
# val_ds에는 val_transform 적용을 위해 transform 교체
val_ds.dataset = datasets.ImageFolder(root=data_dir, transform=val_transform)
# 4) DataLoader 생성
batch_size = 32
train_loader = DataLoader(train_ds, batch_size=batch_size,
                          shuffle=True,  num_workers=4, pin_memory=True)
val_loader   = DataLoader(val_ds,   batch_size=batch_size,
                          shuffle=False, num_workers=4, pin_memory=True)
from collections import Counter
# 5) 클래스 확인
print("클래스:", full_ds.classes)
```

```
dataset = datasets.ImageFolder(root=data_dir)
# dataset.samples 는 [(파일경로, 클래스인덱스), ...] 리스트

# 2) 클래스 인덱스만 뽑아서 개수 세기
counter = Counter([label for _, label in dataset.samples])

# 3) 클래스명과 함께 출력
for idx, cls_name in enumerate(dataset.classes):
    print(f"{cls_name:4s} : {counter[idx]} 장")
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import models
from tqdm import tqdm  # 진행바
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = models.resnet101(pretrained=True)


print("CUDA available:", torch.cuda.is_available())

for param in model.parameters():
    param.requires_grad = True
# 마지막 FC 레이어 교체
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, len(full_ds.classes))
model = model.to(device)
# 2) 손실함수와 옵티마이저
optimizer = optim.SGD(model.parameters(), lr=0.0009, momentum=0.9)
criterion = nn.CrossEntropyLoss()

# Optional: LR 스케줄러
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
# 3) 학습·검증 루프
num_epochs = 20
train_losses = []
val_losses = []

train_acc_list = []
val_acc_list = []
```

```
best_acc = 0.0

for epoch in range(1, num_epochs+1):
    # — Training
    model.train()
    train_loss = 0
    train_correct = 0
    for images, labels in tqdm(train_loader, desc=f"Epoch {epoch} [Train]"):
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()*images.size(0)
        train_correct += (outputs.argmax(1)==labels).sum().item()
    train_loss /= len(train_loader.dataset)
    train_acc   = train_correct/len(train_loader.dataset)
    # — Va
    model.eval()
    val_loss = 0
    val_correct = 0
    with torch.no_grad():
        for images, labels in tqdm(val_loader, desc=f"Epoch {epoch} [Val]"):
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)

            val_loss += loss.item()*images.size(0)
            val_correct += (outputs.argmax(1)==labels).sum().item()
    val_loss /= len(val_loader.dataset)
    val_acc   = val_correct/len(val_loader.dataset)

    train_acc_list.append(train_acc)
    val_acc_list.append(val_acc)
```

```
        scheduler.step()
        train_losses.append(train_loss)
        val_losses.append(val_loss)

        print(f"₩nEpoch {epoch:2d} | "
              f"Train Loss: {train_loss:.4f}, Acc: {train_acc:.4f} | "
              f"Val Loss: {val_loss:.4f}, Acc: {val_acc:.4f}₩n")



        if val_acc > best_acc:
            best_acc = val_acc
            torch.save(model.state_dict(), 'best_model2.pth')
            print(f"Epoch {epoch}: New best acc {best_acc:.4f} → saved.")

        #print(f"Epoch {epoch}: val_acc = {val_acc:.4f} (best: {best_acc:.4f})")
import matplotlib.pyplot as plt
plt.plot(train_losses, label='train loss')
plt.plot(val_losses,    label='val loss')
plt.legend(); plt.show()
```

clothes_recommend.py

```
import pandas as pd
import os
import cv2
from PIL import Image
import numpy as np

def recommend_clothes(emotion_input:str):
        script_dir = os.path.dirname(os.path.abspath(__file__))
        csv_path = os.path.join(script_dir, 'clothes_data.csv')
        img_dir = os.path.join(script_dir, 'clothes_style')

        # === CSV 불러오기 ===
        df = pd.read_csv(csv_path)

        # === 감정 입력 받기 ===
        # emotion_input = input("감정을 입력하세요 (happy, sad, angry, default 중 하나): ").strip().lower()
```

```
# valid_emotions = ["happy", "sad", "angry", "default"]
# if emotion_input not in valid_emotions:
#       print("잘못된 감정 입력입니다.")
#       exit()

# === 상의/하의 스타일 정의 ===
top_styles = ['top', 'outer', 'knit','shirt']
bottom_styles = ['shorts', 'pants']

# === 조건에 맞는 옷 필터링 ===
available = df[
    (df['emotion'].str.lower() == emotion_input) &
    (df['in_stock'].astype(str).str.upper() == "TRUE")
]

# === 한글 경로 이미지 열기 함수 ===
def load_image_unicode(path):
    try:
        with Image.open(path) as pil_img:
            return cv2.cvtColor(np.array(pil_img.convert("RGB")), cv2.COLOR_RGB2BGR)
    except Exception as e:
        print(f"이미지 열기 실패: {e}")
        return None

# === 상의 추천 ===
top_candidates = available[available['style'].str.lower().isin(top_styles)]
if not top_candidates.empty:
    top = top_candidates.sample(1).iloc[0]
    top_img_path = os.path.join(img_dir, top['filename'])

    print("₩n상의 추천:")
    print(f"- 파일명: {top['filename']}")
    print(f"- 설명: {top['description']}")
    print(f"- 경로: {top_img_path}")

    img = load_image_unicode(top_img_path)
    if img is None:
        print("상의 이미지 파일을 열 수 없습니다.")
    else:
```

```
            img = cv2.resize(img, (300, 300))
            cv2.imshow("상의 추천", img)
    else:
        print("추천 가능한 상의가 없습니다.")

    # === 하의 추천 ===
    bottom_candidates = available[available['style'].str.lower().isin(bottom_styles)]
    if not bottom_candidates.empty:
        bottom = bottom_candidates.sample(1).iloc[0]
        bottom_img_path = os.path.join(img_dir, bottom['filename'])

        print("₩n하의 추천:")
        print(f"- 파일명: {bottom['filename']}")
        print(f"- 설명: {bottom['description']}")
        print(f"- 경로: {bottom_img_path}")

        img = load_image_unicode(bottom_img_path)
        if img is None:
            print(" 하의 이미지 파일을 열 수 없습니다.")
        else:
            img = cv2.resize(img, (300, 300))
            cv2.imshow("하의 추천", img)
    else:
        print("추천 가능한 하의가 없습니다.")

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

emotionRecognizer.py

```
import torch
import torch.nn as nn
from torchvision import models, transforms
import time
from collections import Counter
import numpy as np
import cv2
class EmotionRecognizer:
    def __init__(self, model_path = "best_model2.pth"):
    #def __init__(self, model_path = r"data₩best_model2.pth"):
```

```
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        print(f"cuda available: {torch.cuda.is_available()}")

        class_names = ["angry", "default", "happy", "sad"]
        m_path = model_path
        model = models.resnet101(weights=None)

        model.fc = nn.Linear(model.fc.in_features, len(class_names))
        model.load_state_dict(torch.load(m_path, map_location=device))
        model.to(device).eval()

        self.preprocess = transforms.Compose([
            transforms.Resize((300, 300)),
            #transforms.Grayscale(num_output_channels=3),
            transforms.ToTensor(),
            transforms.Normalize(
                mean=[0.485, 0.456, 0.406],
                std =[0.229, 0.224, 0.225]
            ),
        ])

        self.model = model
        self.class_names = class_names
        self.device = device
    def recognize_emotion(self, frame: np.ndarray):
        with torch.no_grad():
            pil = transforms.ToPILImage()(frame)

            # 전처리 및 추론
            input_tensor = self.preprocess(pil).unsqueeze(0).to(self.device)
            output = self.model(input_tensor)
            probs = torch.softmax(output, dim=1)[0]

            pred_idx = torch.argmax(probs).item()
            pred_label = self.class_names[pred_idx]

            return pred_label, probs.cpu()

    def getClassNames(self):
```

```
        return self.class_names
```

emotionUI.py

```
import tkinter as tk
from tkinter import ttk
from collections import Counter
from webcam import *
from emotionRecognizer import EmotionRecognizer
from PIL import Image, ImageTk

import matplotlib.pyplot as plt
from collections import Counter

import torch
import clothes_recommend as er

class UI(tk.Tk):
    def __init__(self):
        super().__init__()

        self.cam =Webcam()
        self.ER = EmotionRecognizer("best_model2.pth")
        #self.configure()

        self.img = None
        self.width = 1280
        self.height = 720

        self.geometry(f"{self.width}x{self.height}")

        self.title("Emotion Detection UI")
        self.resizable(False, False)
        #rect(60, 68, 739, 642)
        self.insertUI()

        self.is_recognizing = False
        self.image_on_canvas    = None
        self.emotion = "준비중"
```

```
        self.elapsed_time = 0.0

        self.update()
    def update(self):

        self.cam.update()
        self.img = self.cam.getImg_tk()
        if self.image_on_canvas is None:
            self.image_on_canvas = self.canvas.create_image(0, 0, anchor=tk.NW, image=self.img)
        else:
            self.canvas.itemconfig(self.image_on_canvas, image=self.img)

        self.UIupdate()

        self.after(1, self.update)

    def insertUI(self):
        # 카메라 프레임 캔버스
        self.canvas   = tk.Canvas(self, width = 720, height = 580, bg = "white")
        self.cam.setSize(720, 580)
        self.canvas.place(x = 60, y = 60)

        btn = tk.Button(self, text = "감정 인식",
                            command=self.start_recognize,
                            font=("맑은 고딕", 20))
        btn.place(x = 1080, y = 570, width = 150, height = 60)
        self.btn = btn


        #실시간 감정 인식 데이터 라벨
        self.emotion_labels: list[tk.Label] = []
        y_positions = [100, 180, 260, 340]
        for y in y_positions:
            label = tk.Label(self, text="감정 인식 준비중", font=("맑은 고딕", 20), anchor="nw")
            label.place(x=820, y=y, width=400, height=60)
            self.emotion_labels.append(label)

        #감정인식 결과 라벨
```

```
        label = tk.Label(self, text="감정 인식 준비중", font=("맑은 고딕", 22), anchor="nw")
        label.place(x = 820, y = 450, width = 400, height = 60)
        self.config_label = label

        #실시간 감정인식 데이터 바
        self.emotion_bars: list[ttk.Progressbar] = []
        for y in y_positions:
            bar = ttk.Progressbar(self, length=200, maximum=100)
            bar.place(x=1000, y=y+50, height=30)  # 라벨과 수직 정렬 보정
            self.emotion_bars.append(bar)

        #감정인식 결과 보기 버튼
        btn = tk.Button(self, text = "결과 분석",
                        command=self.show_pie_chart,
                        font=("맑은 고딕", 18))
        btn.place(x = 820, y = 500, width = 120, height = 40, anchor="nw")
        btn.place_forget()
        self.rst_btn = btn
    def UIupdate(self):
        #print(self.is_recognizing)

        if self.is_recognizing:

            if self.timer():
                #인식 완료
                self.is_recognizing = False
                self.btn.config(state="normal")
                #최빈값
                if self.detected_emotions:
                    self.emotion = Counter(self.detected_emotions).most_common(1)[0][0]



                    print(f"₩n 감정 인식 결과: '{self.emotion}'")
                    self.rst_btn.place(x = 820, y = 500)

                else:
                    #인식된 감정이 없을 때(인식률이 확률이 65 %넘는 감정이 한개도 없을 때)
                    print("₩n 감정 데이터를 수집하지 못했습니다.")
                    self.emotion = "인식 불가"
```

```
            else:
                class_names = self.ER.getClassNames()
                emotion, probs = self.detect_emotion()


                #최빈값 추출을 위한 코드
                pred_idx = torch.argmax(probs).item()
                pred_label = class_names[pred_idx]

                #if probs[pred_idx] >= 0.65:
                self.detected_emotions.append(pred_label)

                #self.detected_emotions.append(pred_label)

                self.config_label.config(text=f"현재 감정: {emotion}")

                for i, prob in enumerate(probs):
                    percent = prob.item() * 100
                    self.emotion_labels[i].config(text=f"{class_names[i]}: {percent:5.1f}%")
                    self.emotion_bars[i]['value'] = percent

                self.btn.config(text = f"인식중{self.elapsed_time:.1f}")
        else:
            self.btn.config(text = f"감정 인식")
            self.config_label.config(text=f"인식 결과: {self.emotion}")


    def show_pie_chart(self):
        counts = Counter(self.detected_emotions)
        labels = list(counts.keys())
        sizes = list(counts.values())
        er.recommend_clothes(self.emotion)
        plt.figure(figsize=(6,6))
        plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
        plt.title("recognized emotions ratio")
        plt.axis('equal')  # 원형 유지
        plt.show()
```

```
    def timer(self):
        dt = self.cam.getDT()
        self.elapsed_time += dt
        if self.elapsed_time > 5:
            return True

        return False

    def start_recognize(self):
        self.btn.config(state="disabled")
        self.is_recognizing = True
        self.elapsed_time = 0.0
        self.detected_emotions = []
        self.rst_btn.place_forget()
    def detect_emotion(self):

        return self.ER.recognize_emotion(self.cam.getFrame())

if __name__ == '__main__':
    ui = UI()
    ui.mainloop()
```

func.py

```
import csv

def load_from_file(filename):
    try:
        with open(filename, "r", encoding="utf-8-sig") as f:
            reader = csv.DictReader(f)
            data = [row for row in reader]
        print(f"Data loaded from {filename}")
        return data
    except FileNotFoundError:
        print(f"{filename} not found.")
        return []
```

webcam.py

```
import cv2
import time
```

```
import numpy as np
import os
from PIL import Image, ImageTk
class Webcam:
	def __init__(self):
		cap = cv2.VideoCapture(0)

		self.cam = cap
		time.sleep(2)
		self.pt = time.time()
		self.ct = 0
		self.dt = 0
		self.size = (640, 480)
	def __str__(self):
		return f"webcam"
	def update(self):

		self.ct = time.time()
		ret, self.frame = self.cam.read()

		if not ret:
			print("E: No Cam")
			return
		#self.preprocess()
		self.frame = cv2.resize(self.frame, self.size)
		self.frame = cv2.flip(self.frame, 1)
		self.dt =   self.ct - self.pt
		self.pt = self.ct

	def saveFrame(self, path):
		ret, buf = cv2.imencode('.jpg', self.frame)
		if not ret:
			raise RuntimeError("JPEG incoding fail")

		buf.tofile(path)
		print("save sucessfully")
		#success = cv2.imwrite(path, self.frame)
		#print(f"save img: {path} → 성공: {success}")
```

```
        def show(self):
                cv2.imshow("Webcam",self.frame)

        def release(self):
                self.cam.release()
                cv2.destroyAllWindows()

        def getFrame(self):
                return self.frame
        def getImg_tk(self):
                frame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
                img_pil = Image.fromarray(frame)
                return ImageTk.PhotoImage(img_pil)

        def setSize(self, x, y):
                self.size = (x, y)

        def getDT(self):
                return self.dt

        def imgShow(self, title, img):
                cv2.imshow(title, img)

        def __del__(self):
                pass


if __name__ == '__main__':
        cam = Webcam()

        while True:
                cam.update()
                cam.show()
                #print(cam.getDT())
                key = cv2.waitKey(1)
                if key & 0xFF == ord('q'):
                        break
        cam.release()
```