2 조

캡스톤 디자인 보고서

유지훈 203436

김영빈 193408

서민솔 213418

제출일 2024.12.11

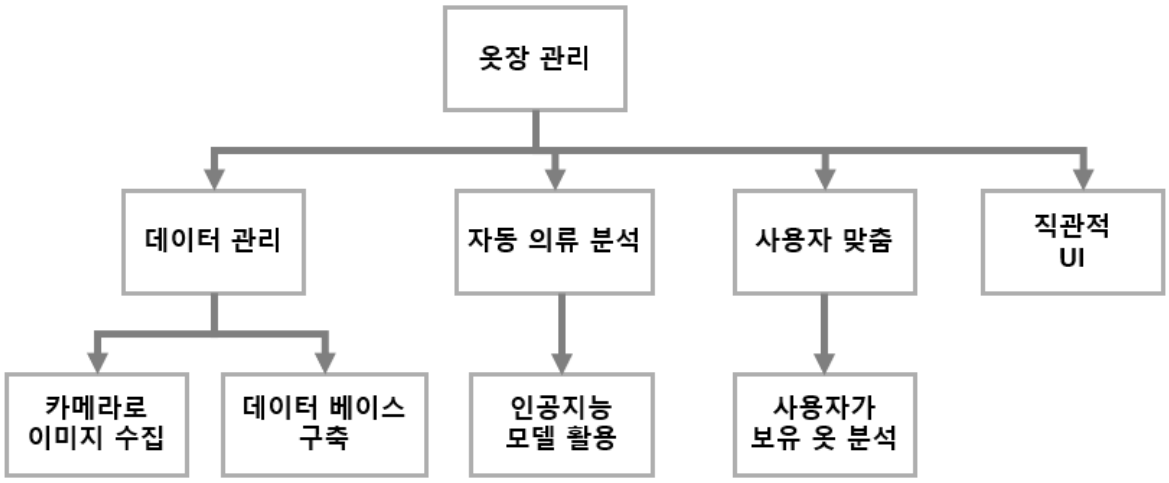
1. 팀원별 역할

- 데이터 저장
- 불러오기 구현
- 모델 학습을 위한 데이터 수집
- 인식 모델 학습을 위한 데이터셋 만들기(바운딩 박스 라벨링)
- 인식 모델 학습
- 분류 모델 학습
- 모델 활용 코드 작성
- 데이터 분류(필터) 구현
- UI 개발
- 전체 시스템 통합

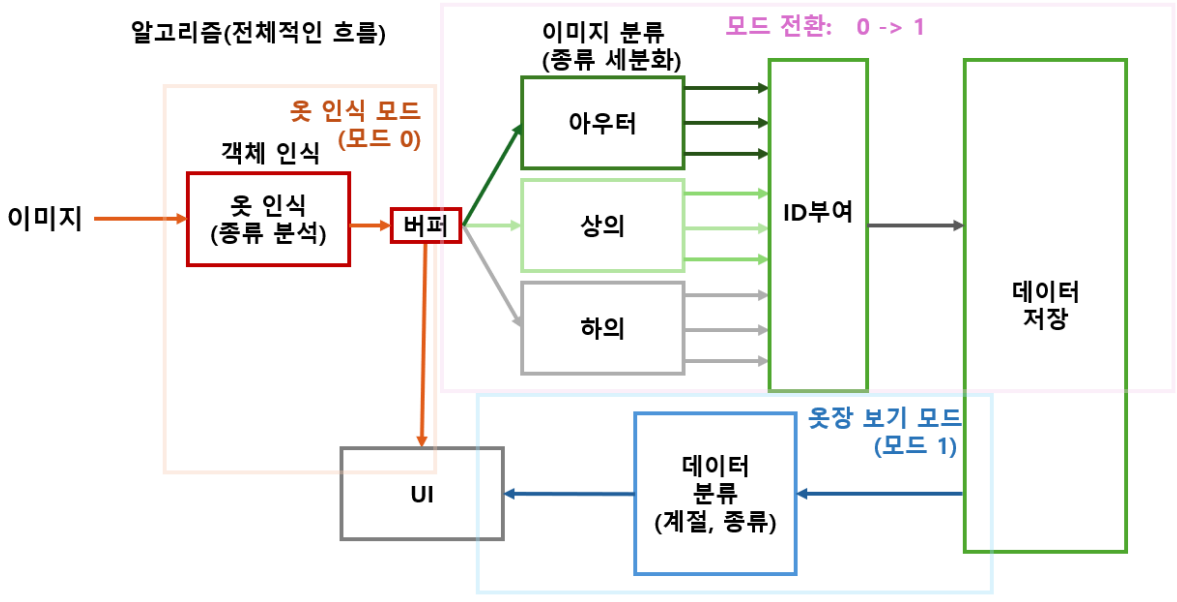
2. 개발 일정

1~7 주차	- 프로젝트 설계, 확정
8 주차	- 제품 구매, 구체적 시스템 설계
9 주차	- 소프트웨어 환경 구축, 필수 라이브러리 설치
10 주차	- 카메라 모듈을 이용해 옷 이미지 수집 구현
11 주차	- 학습을 위한 이미지 데이터 수집 및 데이터셋 생성 - 객체 인식 모델 생성 및 모델 활용 코드 작성
12 주차	- 크롤링으로 이미지 데이터 수집, 분류 모델 학습 - 분류 모델 활용 코드 작성
13 주차	- 데이터 저장 코드 작성(파일 입출력)
14 주차	- UI 개선 및 코드 통합, 오류 수정

3. 목적 계통도



4. 시스템 블록도 및 작품 요약 설명



작품 설명 요약

객체 인식 모드, 모드 전환, 옷장 보기 모드 3가지 상황으로 나누어서 설계했다.

인식 모드: 옷 인식하는 연산이 매우 크기 때문에 과부화가 걸린다. 이를 보완하기 위해 옷 인식 모델로 추론한 옷의 종류와 이미지를 버퍼(list)에 저장해 둔다.

모드 전환: 모드가 전환될 때 버퍼에 있는 데이터들을 한번 더 분류하여 세부 데이터를 얻어낸다. 이미지 분류가 완료된 객체에 ID를 부여하고 세부 데이터들과 함께 리스트에 저장한다.

보기 모드: 저장한 데이터 들은 사용자가 원하는 데이터를 분류하여 볼 수 있게 구현하였다.

5. 설계제한요소

제한요소	내용 (설계사양)
경제성	라즈베리파이 기반으로 개발하여 비용을 최소화.
안전성	하드웨어 모니터링으로 안전성 검증
신뢰성	카메라 모듈과 센서의 데이터 정확성.
미관	직관적인 UI 디자인 및 간결한 하드웨어 구성.
사회적 영향	지속 가능한 패션 소비를 유도하고 의류 관리의 편의성 제공, 의류 소비 실패를 줄여 버려지는 의류 감소
기능성	모든 기능이 유기적으로 연결되어 직관적인 사용자 경험을 제공.

6. 사용 소프트웨어, 하드웨어 목록 및 사용 내용 요약

소프트웨어

Raspbian OS : 라즈베리파이에서 실행되는 기본 운영체제.

Python : 전체적인 프로그램 구현할 언어

Tkinter : UI개발을 위한 파이썬 라이브러리

PyTorch : 욜로 인식 모델 학습과 분류 모델 학습 위한 라이브러리

YOLOV5 : 학습 모델 생성을 위한 워크스페이스

cuda, cudaDNN : 학습할 때 MVIDA GPU사용을 위한 프로그램

OpenCV: 카메라 모듈에서 수집한 이미지를 처리하기 위한 라이브러리, 이미지 전처리에 이용

roboflow annotation: 로보플로우 사이트서 직접 욜로 데이터셋 라벨링을 할 수 있다.

하드웨어

카메라 모듈 : 이미지 데이터 수집

터치스크린 : UI를 통해 사용자와 상호작용하고 서비스 제공

라즈베리파이 : 시스템을 실행시키는 중심 제어 장치

데스크탑 : 학습 모델 생성

7. 알고리즘 설명을 포함한 작품의 상세설명

a. 카메라를 통한 이미지 데이터 수집 – Camera 클래스 (모드 0)

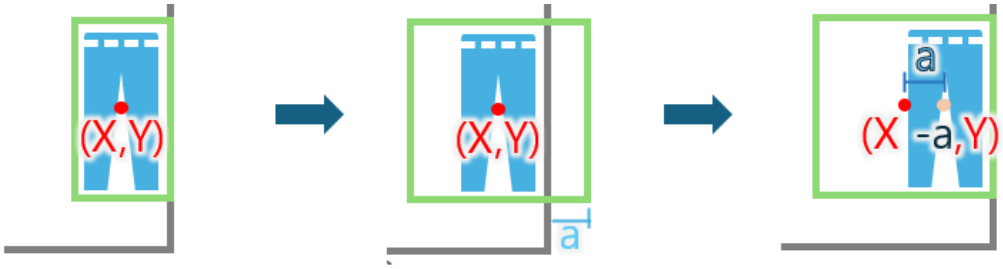
Picamera2라이브러리를 사용하여 실시간으로 카메라의 이미지 데이터를 전달하고, 프레임당 시간(DT)을 계산한다.

b. 객체 인식 – Detector 클래스 (모드 0)

카메라 스트리밍과 실시간 객체인식은 모드 0에서만 동작한다.

이미지를 전달 받아 옷 인식 모델을 사용하여 정보를 추출한다. 인식 모델은 옷의 바운딩 박스와 상의, 하의, 아우터를 구별하는 추론을 수행한다. 크기가 매우 크거나 작은 오브젝트는 제외시키고 통과한 오브젝트들 중에서 신뢰도가 가장 높은 오브젝트와 자른 이미지를 전달한다.

이미지를 자르는 방법 – width와 height중에 큰 값을 한 변으로 하는 사각형으로 자른다. 사각형이 이미지 범위에서 벗어날 경우 오프셋을 조정하는 코드를 작성한다.



자른 이미지는 버퍼에 저장된다. 같은 객체를 연속해서 버퍼에 저장하는 것을 방지하기 위해 저장하는 조건을 설정해 주었다.

Detector의 인식 리턴값은 [정사각형 이미지, 크기, 종류, 신뢰도]이다.

- 인식한 객체가 있을 때

이번 프레임에 Detector가 옷을 인식 했으면 count++해준다. count는 최대 3의 값을 가지며 최대값이 되었을 때 현재 인식하고 있는 옷을 저장할 수 있는 상태가 되고 객체 데이터를 임시변수

에 저장한다

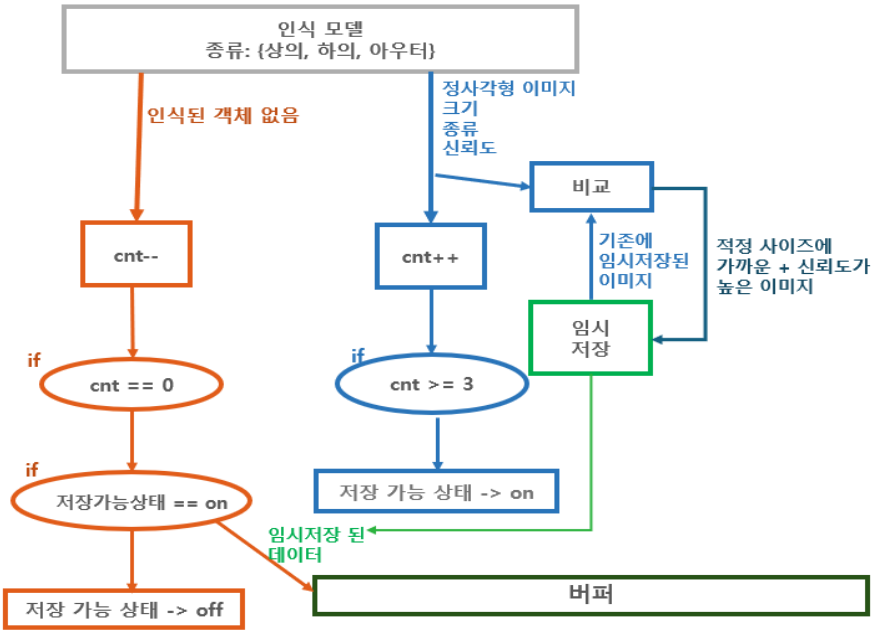
- 인식한 객체가 없을 때

Detector가 인식한 옷이 없으면 count--해준다. count는 최소 0의 값을 가지며 count가 0이고 저장 가능 상태이면 임시변수에 있는 데이터를 버퍼에 저장한다. 그리고 임시변수의 값을 제거한다.

- 객체가 인식된 여러 프레임의 이미지들 중에서 저장할 이미지 선택

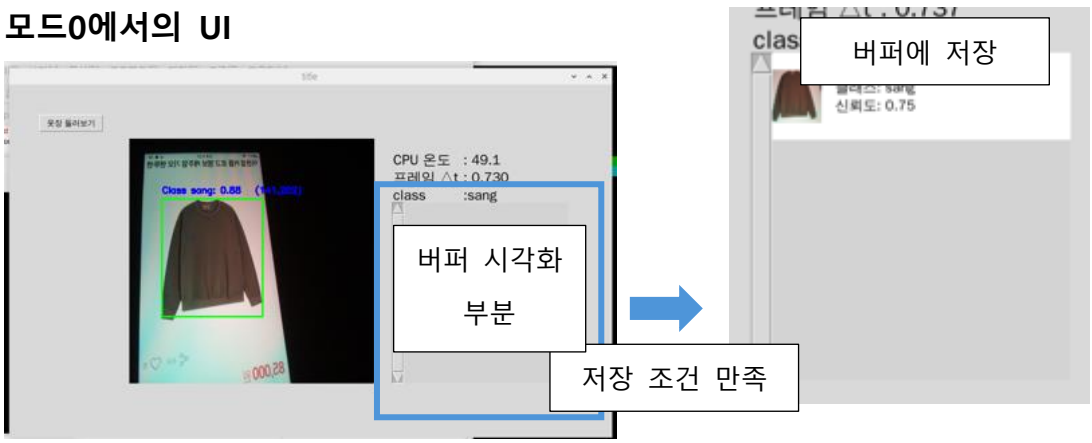
인식된 객체 데이터에는 크기 데이터도 포함되어 있다. 기존의 임시변수에 저장된 데이터와 현재의 데이터를 비교하여 적정 크기와 가까운 이미지 데이터를 임시변수에 저장한다.

아래는 위에서 설명한 알고리즘의 블록도이다.



모드 전환은 UI를 통해 사용자가 전환한다

모드0에서의 UI



c. 이미지 상세 정보 추출 – ClothesSorter 클래스 (모드 0 -> 1)

객체인식 모드일 때 프레임저하가 심하기 때문에 인식된 오브젝트를 버퍼에 저장하고 모드를 전환할때 몰아서 상세 정보를 추출하도록 설계하였다. 모드 전환은 사용자가 UI를 통해 전환한다.

상의 분류:	셔츠	스웨터, 니트	맨투맨, 후드티	반팔 티셔츠	
하의 분류:	청바지	반바지	슬렉스, 면바지	스웨트 팬츠	카고
아우터 분류:	점퍼	패딩	바람막이	후리스	자켓

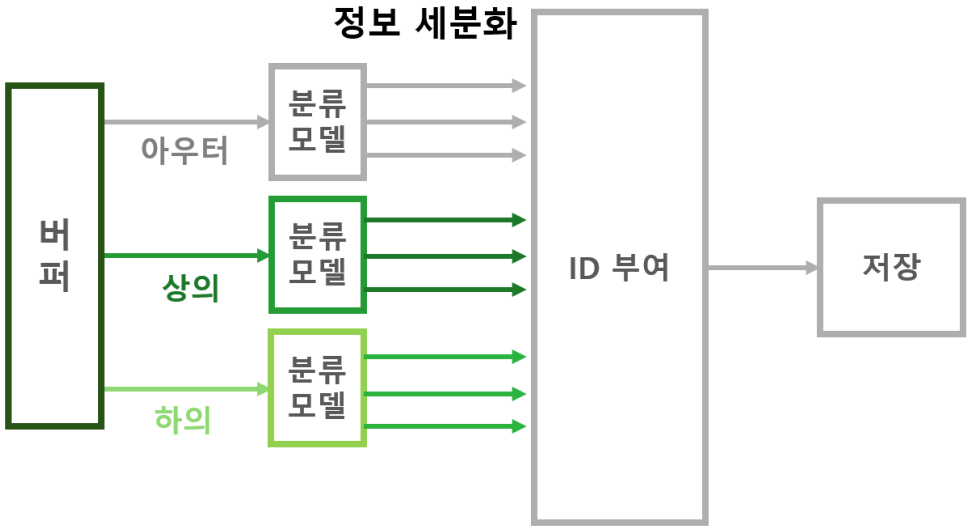
아이디 부여 방법

날짜	시간	종류	세부종류
20241207	134359	1	3

->2024120713435913

현재 날짜와 시간을 아이디 앞부분으로 설정해 중복을 방지한다.
종류는 [상의, 하의, 아우터]가 1 – 3, 세부 종류는 각 분류 클래스의 인덱스 값이다.
아이디를 포함한 옷 객체 정보 리스트를 UI에 넘겨 저장한다.

아래는 위에서 설명한 알고리즘의 블록도이다.



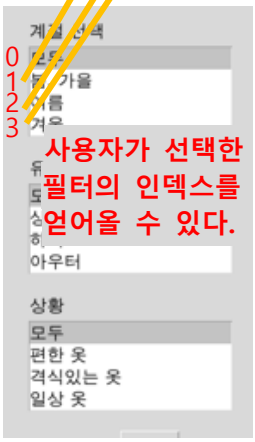
d. 이미지 데이터 분류 – UI 클래스 (모드1)

옷 객체들을 계절, 종류, 상황에 따라 분류한다.
배열을 만들어 각 조건에 맞는 세부 종류들을 저장한다.

```
self.filter_season = [['셔츠', '스웨터, 니트' ,'맨투맨, 후드티','청(데님)바지', '슬랙스, 면바지', '스웨트 팬츠', '카고','점퍼', '바람막이','코트', '자켓' ],
                      ['반팔 티셔츠', '청(데님)바지', '반바지', '슬랙스, 면바지' ],
                      ['셔츠', '스웨터, 니트' ,'맨투맨, 후드티','청(데님)바지', '슬랙스, 면바지', '스웨트 팬츠', '카고', '점퍼', '패딩', '코트', '후리스']]
self.filter_kind = ['sum', 'ha', 'outer']


self.filter_sanghwang = [['맨투맨, 후드티', '반팔 티셔츠', '반바지', '스웨트 팬츠','패딩', '바람막이'],
                         ['셔츠', '슬랙스, 면바지', '코트'],
                         ['스웨터, 니트' ,'맨투맨, 후드티', '반팔 티셔츠', '청(데님)바지', '반바지', '스웨트 팬츠', '카고', '점퍼', '패딩', '자켓']]
```

계절: 봄 + 가을, 여름, 겨울 | 종류: 상의, 하의, 아우터 | 상황: 편한 차림, 격식 차림, 일상 차림



필터를 사용해서 데이터 분류:
옷 객체 리스트의 구조는 [ID, 이미지, 세부종류, 종류, 위치]의 객체들의 정보를 딕셔너리 형태로 담고 있다.

사용자가 선택한 필터의 인덱스를 얻어와 맞는 필터에 접근한다. 주의할 점은 '모두'는 인덱스 값이 0 이고 선택한 필터들은 1 부터 시작하기 때문에 필터 인덱스에 접근할 때에는 +1 을 해주어야 한다.

필터링:
'계절'과 '상황'필터: 옷객체의 세부종류데이터가 필터에 있는지 검사하여 있으면 필터링 리스트에 저장.

종류 필터: 옷 객체의 종류 데이터가 필터값과 일치하면 필터링 리스트에 저장

e. 데이터 저장, 불러오기

프로그램을 시작할 때 파일을 불러오고 종료되기 직전에 파일을 저장한다.

데이터 저장하기:

모든 함수와 루프들이 끝나고 프로그램이 종료되기 바로 전에 수행한다.

UI의 객체 정보 리스트를 얻어온다. 객체 정보 중에서 이미지는 따로 jpg로 저장하고 리스트에서 제거한다. 이미지가 제거된 리스트를 csv형식의 파일로 저장한다.

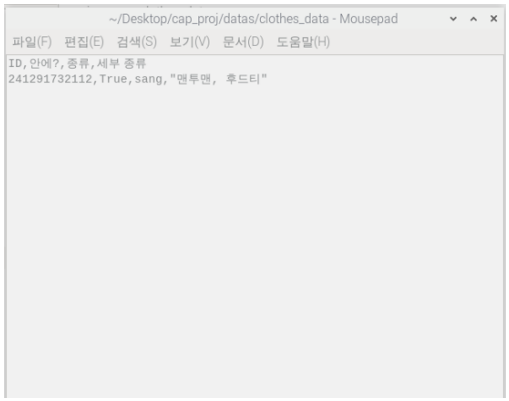
csv위치: ./datas　　파일 이름은 clothes_data.csv

jpg위치: ./datas/imgs　　파일 이름은 아이디.jpg

데이터 불러오기:

csv로 저장된 딕셔너리 형식의 데이터들을 리스트에 불러온다. 딕셔너리의 id값을 이용하여 맞는 이미지를 불러와 딕셔너리에 추가한다.

csv로 저장된 데이터



아이디를 이름으로 저장된 이미지

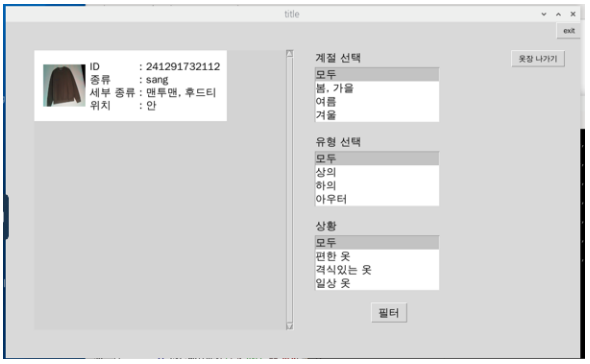
f. 모드에 따라서 UI 분리(tkinter 라이브러리 사용)

이 프로그램은 인식모드인 모드0과 옷 데이터를 확인하는 모드인 모드1이 있다. 모드 별로 다른 화면처럼 보여져야 하기 때문에 두 모드에 따라 UI를 분리하는 것을 구현하였다.

scene[mode]를 하면 모드의 0과 1에 따라 2가지의 인덱스에 접근 할 수 있다. UI 클래스의 생성자에서 두가지 모드의 위젯들을 초기화 한다. 프로그램은 모드0으로 시작한다. 씬에 위젯을 추가할 때는 위젯 객체와 위치 좌표값을 담는다(이런 식으로 [button, x, y]). 모드0의 위젯은 scene[0]에, 모드1의 위젯은 scene[1]에 담는다. 모드가 전환될 때 기존 위젯은 숨기고 바뀐 씬의 위젯은 화면에 그린다.

모드1의 UI

모드2의 UI



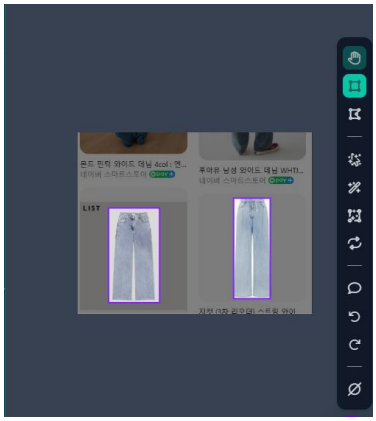
g. 기타 디테일

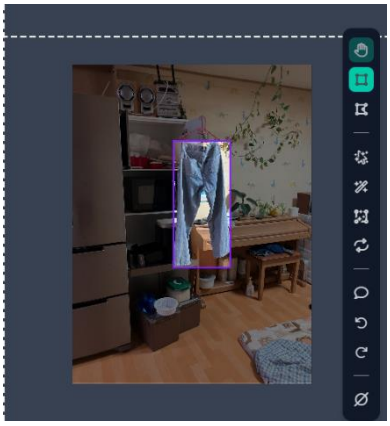
인식 모드에서 인식하는 매우 느리기 때문에 UI 동작이 안될 수 있다. 이를 해결하기 위해 threading 으로 둘의 동작을 분리시켰다.

8. 제작 과정

학습 데이터 생성

-데이터 라벨링





옷 이미지사진을 찍거나 웹사이트에서 캡쳐해서 이미지 데이터를 수집하였다. 인식 모델 데이 수집에는 크롤링을 사용하지 않았다. 그 이유는 이미지 크롤링을 통해 수집한 데이터는 옷 위치가 대부분 가운데에 있어 다양한 위치의 옷 데이터를 학습 할 수 없기 때문이다.

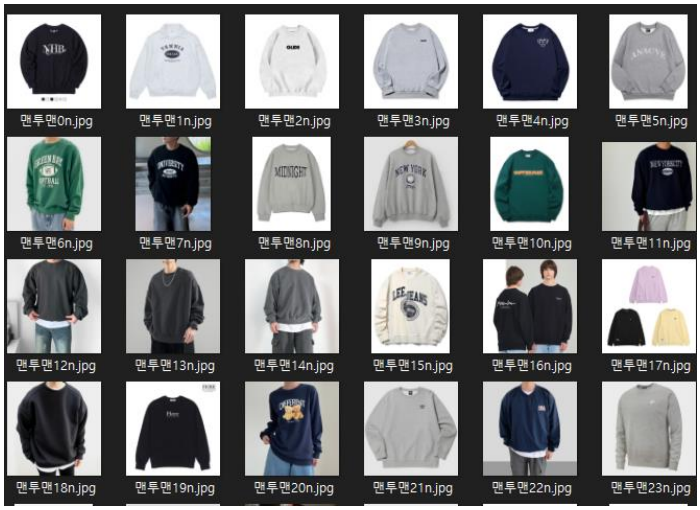
총 401개의 이미지의 라벨링을 하였다. Dataset 401

roboflow에서 제공하는 다양한 옵션으로 데이터를 4364개로 불렸다.

Yolov5 환경에서 epoch과 batch를 다양하게 바꿔보며 학습시켜 성능이 좋은 모델을 찾았다.
성능이 좋은 모델 설정 옵션 -> epochs: 150, batch: 4
모델을 onnx파일로 변환하여 라즈베리파이에 가져왔다.

-데이터 이미지 분류 모델 생성

크롤링을 통해 인터넷에서 이미지 데이터를 긁어온다.



긁어온 이미지들을 pytorch로 분류 모델을 학습시킨 후 이 모델도 onnx파일로 변환 후에 라즈베리파이에 가져온다.

-라즈베리파이와 카메라, 터치스크린 연결

리본모양 케이블을 라즈베리파이에 끼우고 연결한다.

터치스크린에 전원을 공급한다.

8. 제작 후 자체평가 및 소감

하드웨어 성능 한계

라즈베리파이에서 제작한 프로그램을 실행하기에는 모델이 무겁다. 객체 인식 모델을 실행할 경우, CPU 4 개의 사용률이 80%를 초과하며, 1 프레임당 처리 시간이 약 0.7 초 소요된다. 성능 문제를 개선하기 위해서는 서버를 구축하여 연산을 서버 측에서 처리하는 방안을 고려해 볼 수 있다.

최적화된 객체 인식 알고리즘

라즈베리파이의 낮은 성능에 맞춰 객체를 효율적으로 인식하고 저장할 수 있는 알고리즘을 설계했다.

모델 정확도 개선

재 객체 인식 모델의 정확도가 100% 수준까진 아니다. 다양한 환경, 조명, 각도에서 약간의 성능 저하가 발생한다. 이를 해결하기 위해서는 꾸준히 데이터셋을 보강하고, 다양한 조건의 데이터를 수집해야 한다. 객체의 위치와 크기가 균등하게 분포된 데이터셋을 구축하여 모델의 성능을 향상시킬 필요가 있다.

데이터 분류 알고리즘 개선

현재 데이터 분류 알고리즘은 단순 조건 기반으로 설계되어 있다. 이를 머신러닝 기반의 알고리즘으로 개선하면 더 정교한 분류가 가능할 것이다. 추가로, 색상 추출 기능을 구현하여 분류 정확도를 높일 수 있다

실용성 및 사용자 경험 향상

실제 가정에서 사용할 수 있도록 설치 방법의 간소화가 필요하다. 사용자가 쉽게 설치하고 사용할 수 있는 가이드라인을 제공해야 한다.

사용자 맞춤 추천 시스템

사용자 맞춤형 추천 기능을 구현하려면 다양한 정보를 수집하고 분석하는 시스템이 필요하다. 사용자의 스타일, 계절, 상황 등을 고려한 맞춤형 추천을 통해 사용자 경험을 개선할 수 있다.

팀원들의 소감

1) 유지훈:
이번 프로젝트를 통해 하드웨어와 소프트웨어가 통합된 시스템을 설계하면서 많은 것을 배웠습니다. 특히 라즈베리파이의 성능 한계 속에서 객체 인식 모델을 최적화하고 실행 속도를 개선하는 과정이 매우 복잡하다는것을 깨달았습니다. 또한 데이터를 다양하게 수집하면서 머신러닝 모델의 데이터 품질이 얼마나 중요한지 실감했습니다.

2) 김영빈:
시스템 통합 과정에서 발생한 예기치 못한 문제들을 해결하며 협업의 중요성을 느꼈습니다. 특히, 데이터 관리 로직과 UI 통합 작업은 처음에는 복잡했지만, 팀원들과 함께 문제를 나누고 해결하면서 최적의 결과를 도출할 수 있었습니다. 추가로 사용자 맞춤형 추천 시스템을 개발하면 더 큰 성과를 낼 수 있을 것 같습니다.

3) 서민솔:
UI 설계와 데이터 필터링 로직을 개선하면서 사용자 경험(UX)이 얼마나 중요한지 깨달았습니다. 초기에 사용성이 떨어지는 디자인에서 최종적으로 직관적이고 간단한 UI 를 완성하게 되어 뿌듯합니다. 앞으로는 사용자 맞춤형 추천 기능이나 색상 추출 기능 등을 추가해 시스템을 더욱 발전시키고 싶습니다.

회의록

1. 프로젝트 완성 상태 요약

- **완료된 기능:**

1. **AI 추론 최적화: ONNX 모델을 활용한 객체 인식 및 분류 기능.**
2. **데이터 구조 개선: 데이터베이스 없이 배열 기반 설계 및 CSV 저장/로드 방식 채택.**
3. **UI 개선: 옷장 재고 시각화 및 필터링 기능.**
4. **토글 방식 도입: 옷장 안/밖 상태 전환과 실시간 데이터 업데이트.**
5. **통합 테스트 완료: 성능 최적화 및 사용자 경험 개선.**

2. 회의 날짜별 논의 내용 및 작업 정리

10 월 16 일, 17 일: 프로젝트 초기 설계 및 AI 추론 논의

1. **주요 논의:**

- **Yolov5 객체 인식 모델의 성능 및 환경 적응성 검토.**
- **라즈베리파이에서 경량화된 모델 필요성 확인.**

2. **작업 결정:**

- **ONNX 포맷으로 변환 후 모델 실행 환경 테스트.**
- **Batch Size 및 Epoch 조정으로 학습 최적화.**

10 월 22 일, 24 일: 데이터 구조 및 재고 관리 방식 설계

1. **주요 논의:**

- **데이터베이스 사용 여부 검토: 성능 부담으로 인해 사용 지양 결정.**
- **배열 기반 데이터 구조 설계 및 CSV 파일 저장/로드 방식 논의.**
- **옷장 재고를 실시간으로 UI 에 표시하는 로직 설계.**

2. **작업 결정:**

- **객체 정보를 배열 형태로 관리하며 프로그램 시작/종료 시 CSV 파일로 처리.**
- **각 객체의 ID, 이미지, 상태 정보를 UI 와 연동.**

11 월 6 일, 10 일: UI 디자인 및 토글 방식 도입

1. **주요 논의:**

- **사용자 친화적인 UI 설계와 필터링 기능 강화.**
- **옷장에 옷을 넣고 빼는 방식을 간단하게 구현하기 위한 토글 버튼 논의.**

2. **작업 결정:**

- **계절, 상황, 종류에 따라 필터링 가능한 UI 설계.**
- **옷 객체의 상태(옷장 안/밖)를 전환하는 토글 버튼 추가.**

11 월 13 일, 14 일: AI 모델 최적화 및 데이터셋 확장

1. 주요 논의:
- 객체 인식 및 분류 모델의 정확도 향상을 위한 데이터셋 강화.
- 크롤링 및 라벨링 작업으로 다양한 환경의 데이터 추가.

2. 작업 결정:
- 라벨링 작업을 통해 401 개의 이미지에서 4364 개의 학습 데이터 생성.
- 다양한 조명과 각도 조건을 반영한 데이터셋 보강.

11 월 27 일, 28 일: 필터링 로직 및 UI 병렬 처리 개선

1. 주요 논의:
- 필터링 속도 및 정확도 개선 방안 논의.
- UI 와 모델 간 병렬 처리(Threading)로 시스템 반응성 향상.

2. 작업 결정:
- UI 전환 시 기존 데이터를 효율적으로 처리하는 로직 최적화.
- 필터링 결과를 사용자 요청에 맞게 즉시 표시하도록 코드 개선.

12 월 4 일, 5 일: 최종 통합 및 테스트

1. 주요 논의:
- 통합 시스템의 안정성 및 최종 버그 수정.
- 라즈베리파이 환경에서 모든 기능이 원활히 작동하는지 확인.

2. 작업 결정:
- 최종 시스템 테스트 후 모든 문제 수정 완료.
- UI 디자인 및 사용자 피드백 반영하여 시스템 완성.

3. 작업 일정 요약

1. 10 월 16 일~24 일:
- AI 추론 알고리즘 설계 및 데이터 구조 초안 작성.

2. 11 월 6 일~14 일:
- UI 디자인 및 데이터셋 강화.
- 옷장 관리 방식을 단순화하고 사용자 중심으로 설계.

3. 11 월 27 일~12 월 5 일:
- 통합 시스템 테스트 및 최종 문제 수정.

코드

main.py

```
from camera import *
from detect import *
from sorter import *
import threading
import csv
import os
from ui import *
import time

def getCPU_temp():
temp = os.popen("vcgencmd measure_temp").readline()
return float(temp.replace("temp","").replace("'C₩n", "")[1:])

# ==================데이터 로드, 저장 함수 ===============
```



```
def save_to_file(filename, data):
if not data:
print("No data to save.")
return
header = data[0].keys()
with open(filename, "w", encoding="utf-8-sig", newline="") as f:
writer = csv.DictWriter(f, fieldnames=header)
writer.writeheader()
writer.writerows(data)
print(f"Data saved to {filename}")

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 []

#================================================


camera = Camera()
camera.size = (416, 416)
current_directory = os.getcwd()

onnx_model_path = os.path.join(current_directory, 'models/md_final.onnx')

detector = Detector(onnx_model_path)

window = Window(detector)
sorter = ClothesSorter(detector, window)
window.setSorter(sorter)

#======== load data ========
dic = load_from_file("datas/clothes_data")

for d in dic:
d["이미지"] = cv2.imread(f"datas/imgs/{d['ID']}.jpg")

window.data_arr = dic
#============================
def detec_loof():
while True:
if detector.stop:
if window.isStop: break
time.sleep(0.05)
continue
#호출 순서 - 카메라 - detect - 분류
camera.update()
img = camera.getFrame()
img = detector.detect(img)
sorter.update()
t1 = f"CPU 온도₩t : {getCPU_temp()}"
t2 = f"프레임 △t₩t : {camera.getDT()}"
```

```
t3 = f"class₩t :{detector.getObjClass()}"
window.set_ui_onther_loop(img, t1,t2,t3)
#camera.imgShow("aaaa", img)
if window.isStop:
break
#print(camera.getDT())
if cv2.waitKey(1) & 0xFF == ord('q'):
break
thread1 = threading.Thread(target=detec_loof)
thread1.start()
window.showWin()

thread1.join()
camera.release()
#======== save data ========

for d in window.data_arr: #dic 에서 이미지 빼고 이미지 저장
img = d.pop("이미지")
id_ = d['ID']
path = f"datas/imgs/{id_}.jpg"
cv2.imwrite(path, img)
save_to_file("datas/clothes_data", window.data_arr)
```



```
==========================================================
sorter.py
from detect import *
from onnx_run import *
from datetime import datetime

class ClothesSorter:
        def __init__(self, detector, ui):
                self.sang_class = ['셔츠', '스웨터, 니트' ,'맨투맨, 후드티', '반팔 티셔츠']
                self.ha_class = ['청(데님)바지', '반바지', '슬렉스, 면바지', '스웨트 팬츠', '카고']
                self.outer_class = ['점퍼', '패딩', '바람막이','코트', '후리스', '자켓' ]

                self.sangModel = Model('models/sang_model_final1.onnx', self.sang_class)
```

```
                self.haModel = Model('models/ha_model_final1.onnx', self.ha_class)
                self.outerModel = Model('models/outer_model_final1.onnx', self.outer_class)

                self.ObjImg = np.array([])
                self.detector = detector
                self.obj = None
                self.cnt = 0
                self.doSave = False
                self.bestSize = 350
                self.ui = ui

                self.queue = []

# .qsize, .empty, .qget

        def update(self):

                obj, img = self.detector.getObject() #obj : [x, y, w, h, confidence, class_index]

#img : 자른 정사각형 이미지
                if obj[0]:          #obj 가 None 이면 [0,0,0,0,0,0,0]으로 들어오게 해놓았다.
                        self.obj = obj
                        self.setObj(img)
                        if self.cnt > 2:                                #3 프레임 이상 인식되면
저장 가능 상태로 변환
                                self.print_info()
                                self.doSave = True

                                return 1
                        self.cnt = self.cnt + 1
                else:
                        if self.cnt:
                                self.cnt = self.cnt - 1
                                if not self.cnt and self.doSave: #저장 가능 상태일 때
3 프레임 이상 인식이 없으면 저장
                                        cv2.imwrite("sample.jpg", self.ObjImg)
                                        self.queue.append((self.obj, self.ObjImg))
                                        self.ui.addDataToUI(self.ObjImg, f"클래스:
{self.obj[5]}₩n 신뢰도: {self.obj[4]:.2}")
```

```
                                        self.ObjImg = np.array([])                  #오브젝트
정사각형으로 잘라온 이미지
                                        self.doSave = False
                if self.cnt <= 0 and not self.doSave:
                        self.obj = None
                        cnt = 0

                self.print_info()
#       'ha', 'outer','sang'

        rst = None
        def extractInfo(self):
                obj, img = self.queue.pop(0)
                clth_type = obj[5]
                now = datetime.now()

                obj_id =
f"{str(now.year)[2:]}{now.month}{now.day}{now.hour}{now.minute}{now.second}"

                if clth_type == 'sang':
                        obj_id = obj_id + '1'
                        rst = self.sangModel.useModel(img)
                        for i, class_n in enumerate(self.sang_class):
                                if rst['베스트'] == class_n:
                                        obj_id = obj_id + str(i)

                elif clth_type == 'ha':
                        obj_id = obj_id + '2'
                        rst = self.haModel.useModel(img)
                        for i, class_n in enumerate(self.ha_class):
                                if rst['베스트'] == class_n:
                                        obj_id = obj_id + str(i)

                elif clth_type == 'outer':
                        obj_id = obj_id + '3'
                        rst = self.outerModel.useModel(img)
                        for i, class_n in enumerate(self.outer_class):
                                if rst['베스트'] == class_n:
                                        obj_id = obj_id + str(i)
```

```
                for class_name, prob in rst["모두"].items():
                        print(f"{class_name}: {prob:.4f}")

                best = rst['베스트']
                data = {"ID": obj_id, "안에?":True, "종류": clth_type, "세부 종류": best,
"이미지":img}

                return data

        def makeData(self):

                data_arr = []
                while len(self.queue):
                        data_arr.append(self.extractInfo())

                return data_arr

        #검사 방법: 이전 오브젝트 이미지보다 적합한 크기의 이미지이면 변경, 아니면 넘어감
        def print_info(self):
                for d in self.queue:
                        print(f"queue : {d[0]}")

        def setObj(self, img):

                if self.ObjImg.size == 0:
                        self.ObjImg = img
                        return
                #새로운 이미지
                h_, w_, _ = img.shape
                size_ = abs(1.0 - h_ / float(self.bestSize))

                #기존 이미지
                h, w, _ = self.ObjImg.shape
                size = abs(1.0 - h / float(self.bestSize))
                if size > size_:
                        self.ObjImg = img
```


detect.py

```
import onnxruntime as ort
import numpy as np
import cv2

def preprocess_image(img):
img_resized = np.transpose(img, (2, 0, 1))
img_resized = img_resized / 255.0
img_resized = np.expand_dims(img_resized, axis=0).astype(np.float32)
return img_resized

class Detector:
class_name = []
def __init__(self, model_path):

self.session = ort.InferenceSession(model_path, providers=['CPUExecutionProvider']) #,
sess_options = options)
Detector.class_name = ['ha', 'outer','sang']
self.mostObj = [0, 0, 0, 0, 0, 0]
self.cutObj = self.cutObj = np.array([])
self.stop = False
def detect(self, img):
self.mostObj = [0, 0, 0, 0, 0, 0]
self.cutObj = self.cutObj = np.array([])
origin_img = img.copy()
img_resized = preprocess_image(img)

session = self.session

input_name = session.get_inputs()[0].name
output_name = session.get_outputs()[0].name
output = session.run([output_name], {input_name: img_resized})
output = np.array(output[0])
output = np.array(output[0])
for detection in output:
confidence = detection[4]
if confidence > 0.7:
x, y, w, h = detection[:4]
if h > 250 or w < 10 or h > 250 or h < 10:
break
```

```
if x < 60 or x > 416-60 or y< 60 or y > 416-60:
break

class_index = np.argmax(detection[5:]) # highest class Extract

#Most conf Obj
if self.mostObj[4] < confidence:
self.mostObj = [x, y, w, h, confidence, class_index]

if self.mostObj[0]:
x, y, w, h, conf, idx = self.mostObj
print(x, y, w, h, conf, idx)
x1 = int(x - w / 2)
y1 = int(y - h / 2)
x2 = int(x + w / 2)
y2 = int(y + h / 2)
class_str = Detector.class_name[idx]
self.mostObj[5] = class_str
cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
label = f"Class {class_str}: {conf:.2f} ({int(x)},{int(y)})"
cv2.putText(img, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
if w > h:
y1 = int(y - w / 2)
y2 = int(y + w / 2)
else:
x1 = int(x - h / 2)
x2 = int(x + h / 2)
#자르려는 좌표가 밖일 때 욱여 넣기
if x1 < 0:
x2 = x2 - x1
x1 = 1
elif x2 >= 416:
x1 = x1 - (x2 - 416)
x2 = 415
if y1 < 0:
y2 = y2 - y1
y1 = 1
elif x2 >= 416:
y1 = y1 - (y2 - 416)
```

```
y2 = 415
print(x1, x2, y1, y2)
self.cutObj = origin_img[y1:y2, x1:x2]
#cv2.imshow("title", self.cutObj)
print(self.mostObj[5] )
else:
print("No Object!!!!!!!!!!!!!!!")
return img
def segUIImg(self, UI):
pass
def getObject(self):
return self.mostObj, self.cutObj
def getObjClass(self):
return self.mostObj[5]
```

==========

UI.py

```
from camera import *
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import PhotoImage
```



```


from detect import *
from bunriu import *


def f():
        pass

class Window:
        def __init__(self,detector): # sorter 는 getSorter(0)사용

                root = tk.Tk()
                self.root = root
                self.title = "title"


                self.img = None
                self.width = self.root.winfo_screenwidth()
```

```
                self.height = self.root.winfo_screenheig

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


                self.detector = detector
                self.sorter = None

                self.isStop = False

                self.data_arr = []

                self.scene = [[], []]

                self.mode = 0 # detect, view


                #self.root.attributes("-fullscreen",True)
                self.root.title(self.title)
```



```
                root.protocol("WM_DELETE_WINDOW", f)
                exit_button = tk.Button(root, text = "exit", command = self.exit)
                #exit_button.place(x = 800, y = 40, anchor = "nw")
                self.scene[1].append([exit_button, 980, 0])

#================카메라 모드=============


                self.img_label = tk.Label(root, text = "데이터를 기다리는중....")
                self.img_label.place(x = 1024 / 2 - 100, y = 600 / 2, anchor = "center")
                #self.img_label.place(x = (220), y = (220), anchor = "center")

                label = tk.Label(root, text="데이터를 기다리는중....", font = ("", 18))
                label.place(x = 650, y = 110 ,anchor = "nw")
                self.scene[0].append([label, 650, 110])
                self.cpu_label = label

                label = tk.Label(root, text="데이터를 기다리는중....", font = ("", 18))
                label.place(x = 650, y = 140 ,anchor = "nw")
```

```
self.scene[0].append([label, 650, 140])
self.DT_label = label

label = tk.Label(root, text="데이터를 기다리는중....", font = ("", 18))
label.place(x = 650, y = 170 ,anchor = "nw")
self.scene[0].append([label, 650, 170])
self.Obj_label = label

canvas = tk.Canvas(root,bg = "lightgray")
canvas.place(x = 650, y = 200, width = 300, height = 308, anchor = "nw")

scrollbar = tk.Scrollbar(root, orient="vertical", command=canvas.yview)
scrollbar.place(x = 650, y = 200, width = 20, height = 308, anchor = 'nw')
self.scrollbar = scrollbar


self.scene[0].append([canvas,650, 200])
self.scene[0].append([scrollbar,650, 200])
```



```
#self.scene[0].append([scrollbar, 930, 200])

canvas.configure(yscrollcommand = scrollbar.set)

frame = tk.Frame(canvas, bg = "white")
canvas.create_window((0,0),window=frame, anchor = 'nw')


bt = tk.Button(root, text = "옷장 둘러보기", command= self.changeMode)
bt.place(x = 50, y = 50)

self.scene[0].append([bt,50, 50])


self.frame = frame
self.canvas = canvas
```

```
#=========================================================

#=================둘러보기 모드==============================
                bt = tk.Button(root, text = "옷장 나가기", command= self.changeMode)
                self.scene[1].append([bt, 900, 50])


                canvas = tk.Canvas(root,bg = "lightgray")
                canvas.place(x = 50, y = 50, width = 450, height = 500, anchor = "nw")
                self.scene[1].append([canvas, 50, 50])

                frame = tk.Frame(canvas, bg = "white")
                canvas.create_window((0,0),window=frame, anchor = 'nw')

                canvas.place_forget()

                scrollbar = tk.Scrollbar(root, orient="vertical", command=canvas.yview)
                scrollbar.place(x = 500, y = 50, width = 20, height = 500, anchor = 'nw')
                scrollbar.place_forget()
```



```
                self.scene[1].append([scrollbar, 500, 50])


                self.scrollbar1 = scrollbar

                canvas.configure(yscrollcommand = scrollbar.set)

                frame = tk.Frame(canvas, bg = "white")
                canvas.create_window((0,0),window=frame, anchor = 'nw')

                self.frame1 = frame
                self.canvas1 = canvas

                a = 14

                label = tk.Label(root, text="계절 선택", font = ("", a))
                self.scene[1].append([label, 550, 50])
```

```
                listbox = tk.Listbox(root, height = 4, selectmode = tk.SINGLE,
exportselection = False, font=("", a))
                listbox.insert(tk.END, "모두")
                listbox.insert(tk.END, "봄, 가을")
                listbox.insert(tk.END, "여름")
                listbox.insert(tk.END, "겨울")
                listbox.select_set(0)
                self.scene[1].append([listbox, 550, 80])
                self.listbox1 = listbox

                label = tk.Label(root, text="유형 선택", font = ("", a))
                self.scene[1].append([label, 550, 200])

                listbox = tk.Listbox(root, height = 4, selectmode = tk.SINGLE,
exportselection = False, font=("", a))
                listbox.insert(tk.END, "모두")
                listbox.insert(tk.END, "상의")
                listbox.insert(tk.END, "하의")
                listbox.insert(tk.END, "아우터")
                listbox.select_set(0)
                self.scene[1].append([listbox, 550, 230])
                self.listbox2 = listbox

                label = tk.Label(root, text="상황", font = ("", a))
                self.scene[1].append([label, 550, 350])

                listbox = tk.Listbox(root, height = 4, selectmode = tk.SINGLE,
exportselection = False, font=("", a))
                listbox.insert(tk.END, "모두")
                listbox.insert(tk.END, "편한 옷")
                listbox.insert(tk.END, "격식있는 옷")
                listbox.insert(tk.END, "일상 옷")
                listbox.select_set(0)
                self.scene[1].append([listbox, 550, 380])
                self.listbox3 = listbox

                bt = tk.Button(root, text = "필터", command= self.d_filter, font = ("", a))
                self.scene[1].append([bt, 650, 500])
```

```
                self.filter_season = [['셔츠', '스웨터, 니트' ,'맨투맨, 후드티','청(데님)바지',
'슬렉스, 면바지', '스웨트 팬츠', '카고','점퍼', '바람막이','코트', '자켓' ],
                                                        ['반팔 티셔츠', '청(데님)바지',
'반바지', '슬렉스, 면바지' ],
                                                                ['셔츠', '스웨터,
니트' ,'맨투맨, 후드티','청(데님)바지', '슬렉스, 면바지', '스웨트 팬츠', '카고', '점퍼', '패딩', '코트',
'후리스']]
                self.filter_kind = ['sang', 'ha', 'outer']
                self.filter_sanghwang = [['맨투맨, 후드티', '반팔 티셔츠', '반바지', '스웨트
팬츠','패딩', '바람막이'],
                                                                ['셔츠', '슬렉스, 면바지',
'코트'],
                                                                ['스웨터, 니트' ,'맨투맨,
후드티', '반팔 티셔츠', '청(데님)바지', '반바지', '스웨트 팬츠', '카고', '점퍼', '패딩', '자켓']]
#=======================================


        def changeMode(self):
                self.scrollbar1.place(x = 500, y = 50, height = 500)
                if self.mode: # 1 에서 0 으로가는 순간
                        for wg in self.frame1.winfo_children(): # 자식 위젯 싹 긁어옴
                                wg.destroy()
                        self.detector.stop = False

                else: # 0 에서 1 로가는 순간
                        for wg in self.frame.winfo_children(): # 자식 위젯 싹 긁어옴
                                wg.destroy()
                        self.detector.stop = True
                        datas = self.sorter.makeData()

                        for d in datas:
                                self.data_arr.append(d)


                        for wg in self.frame.winfo_children(): # 자식 위젯 싹 긁어옴
                                wg.destroy()

                        #카메라 스트리밍부분 제거
```

```
                empty = PhotoImage()
                self.img_label.configure(image=empty)
                self.img_label.image = empty
                self.img_label.place_forget()
                self.addData()

        for wg in self.scene[self.mode]:
                wg[0].place_forget()

        self.mode = not self.mode

        for wg in self.scene[self.mode]:
                wg[0].place(x = wg[1], y = wg[2])

        if not self.mode:
                self.canvas.place(x = 650, y = 200, width = 300, height = 308,
anchor = "nw")
                self.scrollbar.place(x = 650, y = 200, width = 20, height = 308,
anchor = 'nw')
                self.img_label.place(x = 1024 / 2 - 100, y = 600 / 2, anchor =
"center")

    def setSorter(self, st):
        self.sorter = st

    def exit(self):
        if not self.mode:
                self.addData()

        self.isStop = True
        self.root.quit()

    def showWin(self):
        self.update()
        self.root.mainloop()

    def addData(self):
        self.scrollbar1.place(x = 500, y = 50, height = 500)
        self.canvas1.place(x = 50, y = 50, width = 450, height = 500)
```

```
                arry = self.data_arr
                for arr in arry:
                        frame = tk.Frame(self.frame1, bg='white', pady=10)

                        img = arr["이미지"]
                        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                        img = cv2.resize(img, (75, 75))
                        img = Image.fromarray(img)
                        img = ImageTk.PhotoImage(img)

                        label = tk.Label(frame, image=img, bg="white")
                        label.image = img # 참조 유지
                        label.pack(side="left", padx=5)



                        t0 = f"ID₩t: {arr['ID']}"
                        t1 = f"₩n 종류₩t: {arr['종류']}"
                        t2 = f"₩n 세부 종류₩t: {arr['세부 종류']}"
                        t3 = f"₩n 위치₩t: {'안' if arr['안에?'] else '밖'}"
                        text = t0 + t1 + t2 + t3
                        print(text)
                        label = tk.Label(frame, text=text, bg="white", anchor="w",
justify="left", font = ("", "14"))
                        label.pack(side="left", fill="both",expand = True)


                        # 프레임 배치
                        frame.pack(fill="x", padx=10, pady=5, expand = True)
                self.frame1.update_idletasks()

                frame_width = self.frame1.winfo_reqwidth()
                frame_height = self.frame1.winfo_reqheight()

                self.canvas1.config(scrollregion=(0, 0, frame_width, frame_height))

        def d_filter(self):
                lst = list(self.data_arr)

                seasion = self.listbox1.curselection()[0]
```

```
                kind = self.listbox2.curselection()[0]
                sanghwang = self.listbox3.curselection()[0]
                if seasion:
                        lst = [d for d in lst if d['세부 종류'] in self.filter_season[seasion - 1]]

                if kind:
                        print(lst[0]['종류'],self.filter_kind[kind - 1])
                        lst = [d for d in lst if d['종류'] == self.filter_kind[kind - 1]]
                if sanghwang:
                        lst = [d for d in lst if d['세부 종류'] in
self.filter_sanghwang[sanghwang - 1]]

                for wg in self.frame1.winfo_children():
                                wg.destroy()
                for arr in lst:
                        frame = tk.Frame(self.frame1, bg='white', pady=10)

                        img = arr["이미지"]
                        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                        img = cv2.resize(img, (75, 75))
                        img = Image.fromarray(img)
                        img = ImageTk.PhotoImage(img)

                        label = tk.Label(frame, image=img, bg="white")
                        label.image = img # 참조 유지
                        label.pack(side="left", padx=5)

                        t0 = f"ID₩t: {arr['ID']}"
                        t1 = f"₩n 종류₩t: {arr['종류']}"
                        t2 = f"₩n 세부 종류₩t: {arr['세부 종류']}"
                        t3 = f"₩n 위치₩t: {'안' if arr['안에?'] else '밖'}"
                        text = t0 + t1 + t2 + t3
                        print(text)
                        label = tk.Label(frame, text=text, bg="white", anchor="w",
justify="left", font = ("", "14"))
                        label.pack(side="left", fill="both",expand = True)


                        # 프레임 배치
```

```
                frame.pack(fill="x", padx=10, pady=5, expand = True)
        self.frame1.update_idletasks()

        frame_width = self.frame1.winfo_reqwidth()
        frame_height = self.frame1.winfo_reqheight()

        self.canvas1.config(scrollregion=(0, 0, frame_width, frame_height))

def addDataToUI(self, img, text):
        self.canvas.place(x = 650, y = 200, width = 300, height = 308)
        self.scrollbar.place(x = 650, y = 200, width = 20, height = 308)
        if self.mode:
                return
        frame = tk.Frame(self.frame, bg='white', pady = 10)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (50, 50))
        img = Image.fromarray(img)
        img = ImageTk.PhotoImage(img)

        label = tk.Label(frame, image = img, bg = "white")
        label.image =img
        label.pack(side = "left", padx = 10)

        label = tk.Label(frame, text=text, bg='white', anchor='w', justify = "left")
        label.pack(side = "left", fill='both', expand = True)

        frame.pack(fill='x', padx=5, pady=5)
        self.frame.pack(fill = "x")
        self.frame.update_idletasks()

        frame_width = self.frame.winfo_reqwidth()
        frame_height = self.frame.winfo_reqheight()
        print(frame_width, frame_height)
        self.canvas.config(scrollregion=(0, 0, frame_width, frame_height))

def set_ui_onther_loop(self, img ,text1, text2,text3):
        if self.isStop:
                return
        if self.mode:
```

```
                return

            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = Image.fromarray(img)
            img = ImageTk.PhotoImage(img)
            self.img = img

            self.img_label.configure(image= self.img)
            self.img_label.image = self.img

            self.cpu_label.configure(text =text1 )
            self.DT_label.configure(text = text2[:15])
            self.Obj_label.configure(text = text3)
```

==========

camera.py

```
import cv2
from picamera2 import Picamera2
import time
```



```
class Camera:
    def __init__(self):
        self.cam = Picamera2()
        config = self.cam.create_video_configuration(main={"size":
self.cam.sensor_resolution, "format":"RGB888"})

        self.cam.configure(config)

        self.cam.start()
        time.sleep(2)
        self.pt = time.time()
        self.ct = 0
        self.dt = 0
        self.size = (416, 416)
    def update(self):
        self.ct = time.time()
        self.frame= self.cam.capture_array()
        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 show(self):
            cv2.imshow("stream",self.frame)
    def release(self):
            self.cam.stop()
            cv2.destroyAllWindows()

    def getFrame(self):
            return self.frame

    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
```


==========

(이 코드는 라이브러리 사용, 직접 짠부분 없음.)

onnx_run.py

```
import onnxruntime as ort
import numpy as np
import cv2
from scipy.special import softmax

class Model:
def __init__(self, model_path, class_names):
```

```
self.model_path = model_path
self.class_names = class_names
self.session = ort.InferenceSession(model_path)

def useModel(self, image):
# 전처리
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = cv2.resize(image, (64, 64))
image = image.astype(np.float32) / 255.0 # 정규화
image = (image - 0.5) / 0.5 # 정규화
image = np.transpose(image, (2, 0, 1)) # (H, W, C) -> (C, H, W)
image = np.expand_dims(image, axis=0)
iname = self.session.get_inputs()[0].name
oname = self.session.get_outputs()[0].name
logits = self.session.run([oname], {iname: image})[0]
probabilities = softmax(logits[0])

# 가장 높은 확률의 클래스 선택
predicted_index = np.argmax(probabilities)
predicted_class = self.class_names[predicted_index]

# 클래스별 확률 및 예측 클래스 반환
return {
"모두": {self.class_names[i]: prob for i, prob in enumerate(probabilities)},
"베스트": predicted_class
}
```