지윤 작업용

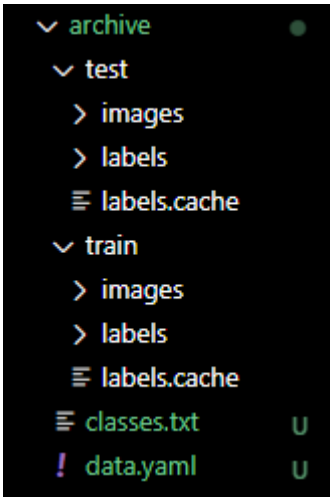
📌 주요 기능 1

1. Yolov5n모델을 이용한 얼굴&손 탐지
2. OpenCV로 웹캠 화면 출력 및 바운딩 박스 표시
3. face 객체에만 가우시안 블러 효과 적용
4. hand 박스의 40% 이상이 face에 3초 이상 겹치는 경우 해당 face객체만 블러 효과 해제

▼ 1차 시도

Dataset 준비 및 모델 학습

- https://www.kaggle.com/datasets/nomihsa965/hand-detection-dataset-vocyolo-format/data 사용 (2052장의 이미지)



- yaml.data

```
train: C:/Users/Admin/Documents/GitHub/JustDance/archive/train/images
val: C:/Users/Admin/Documents/GitHub/JustDance/archive/test/images    #
```

```
nc: 1  # 클래스 개수 (hand만 있기 때문에 1)
names: ['hand']  # 클래스 이름
```

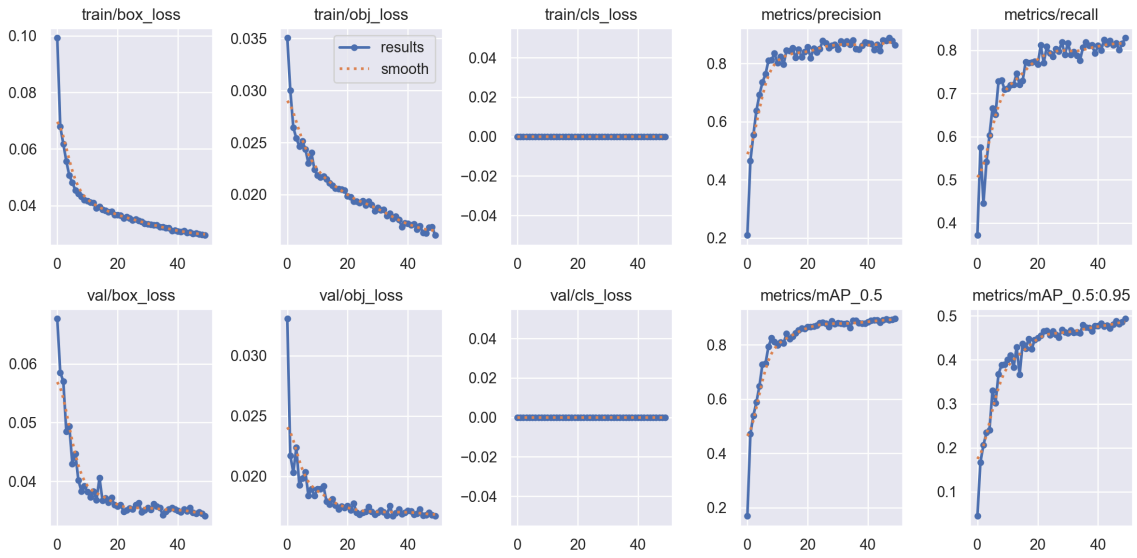
- git clone

```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
pip install -r requirements.txt
```

- 학습 실행 (앱서비스 활용을 고려해 가장 가벼운 모델인 yolov5n을 활용)

```
python train.py --img 640 --batch 16 --epochs 50 --data path/to/data.yam
```

- Results



- 모델 활용

1. Hand 객체 감지 모델 검증 (**best.pt**)

▼ Code

```
import torch

# YOLOv5 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path='C:/U

# 이미지 예측
results = model('C:/Users/Admin/Documents/GitHub/JustDance

# 결과 출력
results.print()  # 텍스트 형식으로 결과 출력
results.show()   # 결과 시각화
```

2. best.pt모델 + OpenCV Cascade모델로 얼굴 감지 & Blur처리

▼ Code

```
########################################
######얼굴 감지&블러 + 손 감지 모델######
########################################

import cv2
import torch
import numpy as np

# YOLOv5 모델 로드 (손바닥 감지용)
model = torch.hub.load('ultralytics/yolov5', 'custom', path='C:/U

# OpenCV Haar Cascade 얼굴 감지 모델 로드
face_cascade = cv2.CascadeClassifier('haarcascade_frontalfac

# 웹캠 실행
cap = cv2.VideoCapture(0)  # 0번 카메라 (웹캠)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
```

```
    # YOLOv5를 사용하여 손바닥 감지
    results = model(frame)
    hand_detections = results.xyxy[0].cpu().numpy()  # 바운딩 박

    # OpenCV로 얼굴 감지
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # 얼굴 검
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1

    # 손바닥 감지 결과 표시
    for x1, y1, x2, y2, conf, cls in hand_detections:
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  # 손바
        cv2.putText(frame, f'Hand {conf:.2f}', (x1, y1 - 10), cv2.FON

    # 얼굴 감지 결과에 블러 적용
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        face_region = frame[y:y+h, x:x+w]
        blurred_face = cv2.GaussianBlur(face_region, (99, 99), 30)
        frame[y:y+h, x:x+w] = blurred_face  # 블러 처리된 얼굴로 교체

    # 결과 출력
    cv2.imshow('Hand Detection and Face Blur', frame)

    # ESC 키로 종료
    if cv2.waitKey(1) & 0xFF == 27:
        break

# 웹캠 종료
cap.release()
cv2.destroyAllWindows()
```

3. hand 객체의 바운딩박스와 blur처리 된 얼굴이 3초 이상 겹치면 blur 해제

- 고려 사항
    1. 앱서비스를 고려했을 때, 얼굴과 손 모두 감지할 수 있는 yolo모델을 학습시켜 사용하는 것이 적합

2. Cascade는 실시간 얼굴 인식의 정확도가 떨어짐

▼ 2차 시도

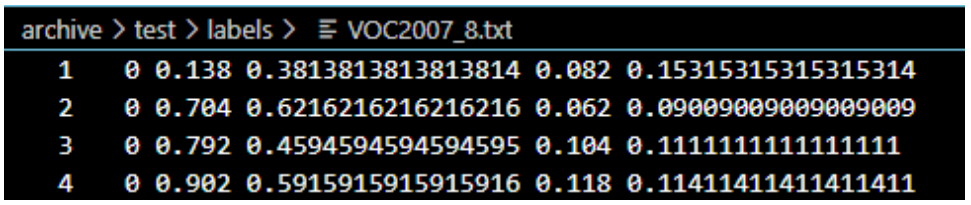
- Hand 학습시킨 kaggle 데이터를 사용
  - OpenCV로 이미지 얼굴 감지 후 yolo 학습용으로 라벨링 및 txt파일 저장
    - hand의 클래스 번호인 0과 중복되지 않게 face 클래스는 1로 지정

▼ Code

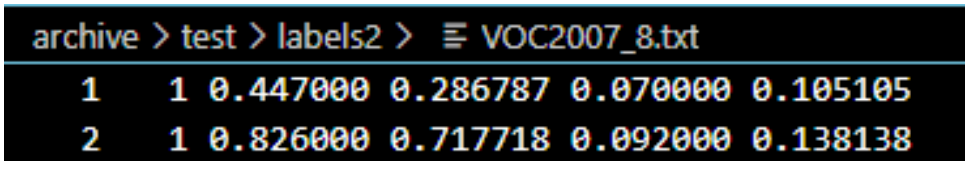
```
################################################
#############이미지 opencv로 얼굴 감지 후 yolo용으로 라벨링#
################################################

import os
import cv2

# Path to dataset
dataset_path = "archive"
subfolders = ["train", "test"]

# Function to detect faces and create YOLO label files
def detect_faces_and_label(subfolder):
    images_path = os.path.join(dataset_path, subfolder, "images
    labels_path = os.path.join(dataset_path, subfolder, "labels2")
    os.makedirs(labels_path, exist_ok=True)

    for image_file in os.listdir(images_path):
        if image_file.endswith((".jpg", ".png", ".jpeg")):
            image_path = os.path.join(images_path, image_file)
            img = cv2.imread(image_path)
            height, width, _ = img.shape

            # Convert to grayscale and detect faces
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            face_cascade = cv2.CascadeClassifier(cv2.data.haarcas
            faces = face_cascade.detectMultiScale(gray, scaleFacto
```

```
            # Create label file
            label_file = os.path.join(labels_path, os.path.splitext(ima
            with open(label_file, "w") as f:
                for (x, y, w, h) in faces:
                    # YOLO format: class_id x_center y_center width he
                    x_center = (x + w / 2) / width
                    y_center = (y + h / 2) / height
                    w_norm = w / width
                    h_norm = h / height
                    f.write(f"1 {x_center:.6f} {y_center:.6f} {w_norm:.6f}

                print(f"Processed {image_file}, found {len(faces)} face(s

# Process each subfolder
for subfolder in subfolders:
    detect_faces_and_label(subfolder)

print("Labeling completed.")
```

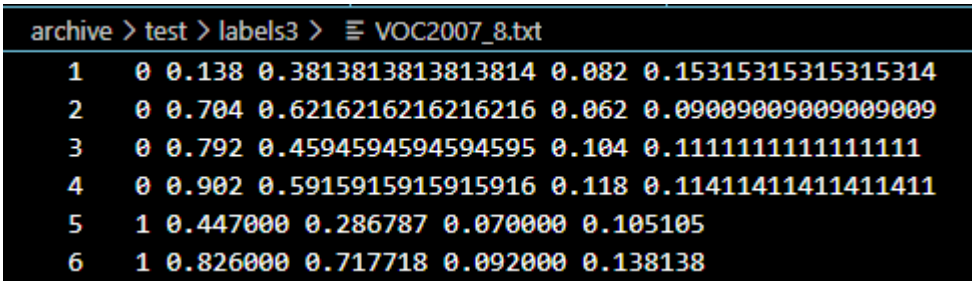
- 라벨 병합 및 새로운 폴더에 저장
  - 기존의 hand(0)라벨값



  - OpenCV로 추출한 face(1) 라벨값



  - 동명의 txt파일 병합



▼ Code

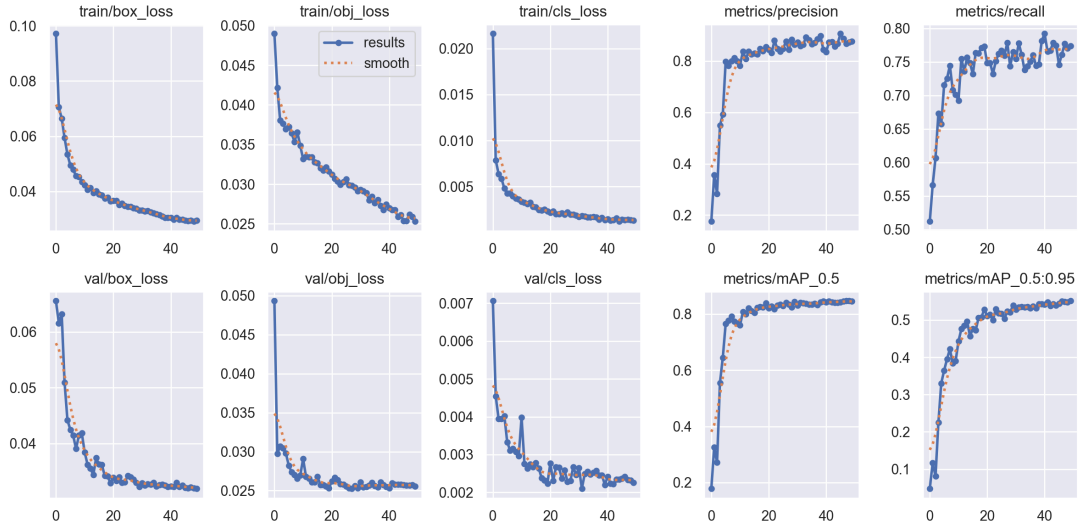
```
################################################
###############라벨 병합 및 새로운 폴더에 저장############
################################################

import os

# Define folder paths
base_path = "archive/train"  # Root directory for test
labels_path = os.path.join(base_path, "labels")
labels2_path = os.path.join(base_path, "labels2")
labels3_path = os.path.join(base_path, "labels3")

# Create labels3 folder if it doesn't exist
os.makedirs(labels3_path, exist_ok=True)

# Iterate through all files in labels folder
for file_name in os.listdir(labels_path):
    # Check if the file exists in labels2 folder
    file1_path = os.path.join(labels_path, file_name)
    file2_path = os.path.join(labels2_path, file_name)

    # Skip if it's not a .txt file
    if not file_name.endswith(".txt"):
        continue

    # Initialize content list
    merged_content = []

    # Read content from labels folder file
    if os.path.exists(file1_path):
```

```
        with open(file1_path, "r") as file1:
            merged_content.extend(file1.readlines())

    # Read content from labels2 folder file
    if os.path.exists(file2_path):
        with open(file2_path, "r") as file2:
            merged_content.extend(file2.readlines())

    # Write merged content to labels3 folder
    output_file_path = os.path.join(labels3_path, file_name)
    with open(output_file_path, "w") as output_file:
        output_file.writelines(merged_content)

    print(f"Merged {file_name} into {output_file_path}")

print("All files have been merged into labels3 folder.")
```

- 모델 재 학습 (2052개)
  - data.yaml

```
train: C:/Users/Admin/Documents/GitHub/JustDance/archive_merged/t
val: C:/Users/Admin/Documents/GitHub/JustDance/archive_merged/te

nc: 2  # 클래스 개수
names: ['hand', 'face']  # 클래스 이름
```

```
     49/49      2.44G    0.02947    0.02522   0.001289         66        640:  97%|          | 94/97 [00:07<00:00, 12.0
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(amp):
     49/49      2.44G    0.02946    0.02522   0.001297         78        640:  97%|          | 94/97 [00:07<00:00, 12.0
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(amp):
     49/49      2.44G    0.02947    0.02524   0.001296         82        640:  99%|          | 96/97 [00:07<00:00, 12.0
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(amp):
     49/49      2.44G    0.02945     0.0253   0.001299         95        640: 100%|          | 97/97 [00:07<00:00, 12.
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 16/16 [00:02
                   all        510       1520      0.877      0.774      0.845      0.551

50 epochs completed in 0.146 hours.
Optimizer stripped from runs₩train₩exp4₩weights₩last.pt, 3.8MB
Optimizer stripped from runs₩train₩exp4₩weights₩best.pt, 3.8MB

Validating runs₩train₩exp4₩weights₩best.pt...
Fusing layers...
Model summary: 157 layers, 1761871 parameters, 0 gradients, 4.1 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 16/16 [00:03
                   all        510       1520      0.877      0.775      0.845      0.551
                  hand        510       1003       0.88      0.804      0.885       0.48
                  face        510        517      0.873      0.745      0.805      0.621
Results saved to runs₩train₩exp4
```



- 모델 활용
  1. 모델 퍼포먼스 확인

     ▼ Code

```
import torch
import cv2
import time

# YOLOv5 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path='han

# 웹캠 열기
```

```
cap = cv2.VideoCapture(0)  # 0: 기본 웹캠
if not cap.isOpened():
    print("Error: 웹캠을 열 수 없습니다.")
    exit()

# FPS 계산용 변수 초기화
prev_time = 0

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: 프레임을 읽을 수 없습니다.")
        break

    # YOLOv5 모델로 프레임 예측
    results = model(frame)

    # 탐지 결과 정보 가져오기
    detections = results.xyxy[0]  # 탐지 결과 (x1, y1, x2, y2, confid
    for *box, conf, cls in detections:
        # 클래스 이름 가져오기
        cls_name = 'Hand' if int(cls) == 0 else 'Face'
        x1, y1, x2, y2 = map(int, box)  # 좌표값 정수형으로 변환

        # 탐지 결과 박스와 텍스트 표시
        color = (0, 255, 0) if int(cls) == 0 else (255, 0, 0)  # Hand: g
        label = f'{cls_name} {conf:.2f}'
        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)  # 사각형 그
        cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY

    # FPS 계산 및 표시
    curr_time = time.time()
    fps = 1 / (curr_time - prev_time)
    prev_time = curr_time
    cv2.putText(frame, f'FPS: {fps:.2f}', (10, 30), cv2.FONT_HERS

    # 화면에 표시
    cv2.imshow('YOLOv5 Webcam Detection', frame)
```

```
    # ESC 키를 누르면 종료
    if cv2.waitKey(1) & 0xFF == 27:
        break

# 웹캠 및 OpenCV 창 닫기
cap.release()
cv2.destroyAllWindows()
```

2. Face 객체에만 Blur처리

▼ Code

```
################################################
########face 객체에만 블러 처리하기#################
################################################


import torch
import cv2
import time

# YOLOv5 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path='han

# 웹캠 열기
cap = cv2.VideoCapture(0)  # 0: 기본 웹캠
if not cap.isOpened():
    print("Error: 웹캠을 열 수 없습니다.")
    exit()

# FPS 계산용 변수 초기화
prev_time = 0

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: 프레임을 읽을 수 없습니다.")
        break
```

```
    # YOLOv5 모델로 프레임 예측
    results = model(frame)

    # 탐지 결과 정보 가져오기
    detections = results.xyxy[0]  # 탐지 결과 (x1, y1, x2, y2, confid
    for *box, conf, cls in detections:
        x1, y1, x2, y2 = map(int, box)  # 좌표값 정수형으로 변환
        cls_name = 'Hand' if int(cls) == 0 else 'Face'

        # Face 객체만 블러 처리
        if int(cls) == 1:  # Face (클래스 ID 1)
            face_roi = frame[y1:y2, x1:x2]  # 얼굴 영역
            blurred_face = cv2.GaussianBlur(face_roi, (51, 51), 30)  #
            frame[y1:y2, x1:x2] = blurred_face  # 블러 처리된 얼굴 다시
        else:
            # Hand 객체는 시각화만 처리
            color = (0, 255, 0)  # Hand: green
            label = f'{cls_name} {conf:.2f}'
            cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)  # 사각형
            cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSH

    # FPS 계산 및 표시
    curr_time = time.time()
    fps = 1 / (curr_time - prev_time)
    prev_time = curr_time
    cv2.putText(frame, f'FPS: {fps:.2f}', (10, 30), cv2.FONT_HERS

    # 화면에 표시
    cv2.imshow('YOLOv5 Webcam Detection with Face Blur', fra

    # ESC 키를 누르면 종료
    if cv2.waitKey(1) & 0xFF == 27:
        break

# 웹캠 및 OpenCV 창 닫기
cap.release()
cv2.destroyAllWindows()
```

3. Face 바운딩박스 위에 hand 박스의 40%가 3초 이상 Overlap되면 Blur 해제

▼ Code

```
################################################
###########3초 후 블러 해제###########
################################################

import torch
import cv2
import time

# YOLOv5 모델 로드 (얼굴과 손 인식)
model = torch.hub.load('ultralytics/yolov5', 'custom', path='han

# 웹캠 열기
cap = cv2.VideoCapture(0)  # 0: 기본 웹캠

# FPS 계산용 변수 초기화
prev_time = 0

# 얼굴 바운딩박스와 손 바운딩박스가 겹친 시간을 추적하기 위한 딕셔너
face_status = {}

# 얼굴과 손이 겹치는 비율 계산 함수
def calculate_intersection_area(box1, box2):
    x1, y1, x2, y2 = box1
    x1_2, y1_2, x2_2, y2_2 = box2

    # 겹치는 영역의 좌상단과 우하단 좌표 계산
    x_overlap = max(0, min(x2, x2_2) - max(x1, x1_2))
    y_overlap = max(0, min(y2, y2_2) - max(y1, y1_2))

    # 겹치는 영역의 면적 계산
    overlap_area = x_overlap * y_overlap
    return overlap_area

while True:
    ret, frame = cap.read()
```

```
    if not ret:
        print("Error: 프레임을 읽을 수 없습니다.")
        break

    # YOLOv5 모델로 프레임 예측
    results = model(frame)

    # 탐지 결과 정보 가져오기
    detections = results.xyxy[0].cpu().numpy()  # 탐지 결과 (x1, y1

    # 현재 프레임의 얼굴 ID를 저장
    current_faces = []

    # 탐지된 객체 처리
    for *box, conf, cls in detections:
        x1, y1, x2, y2 = map(int, box)  # 좌표값 정수형으로 변환

        if int(cls) == 1:  # Face (클래스 ID 1)
            # 얼굴 객체만 블러 처리
            face_id = f"face_{x1}_{y1}"  # 얼굴 ID 생성
            current_faces.append(face_id)
            face_box = (x1, y1, x2, y2)

            # 얼굴 상태 추적
            if face_id not in face_status:
                face_status[face_id] = {"start_time": None, "blurred":

            # 손 객체와 얼굴 바운딩박스 겹침 확인
            for x1_hand, y1_hand, x2_hand, y2_hand, conf_hand, cls_
                if int(cls_hand) == 0:  # 손 (클래스 ID 0)
                    hand_box = (int(x1_hand), int(y1_hand), int(x2_hand
                    overlap_area = calculate_intersection_area(face_bo

                    # 손 바운딩박스의 30% 이상이 얼굴 바운딩박스에 겹칠 경
                    face_area = (x2 - x1) * (y2 - y1)
                    hand_area = (x2_hand - x1_hand) * (y2_hand - y1_h
                    overlap_ratio = overlap_area / hand_area
```

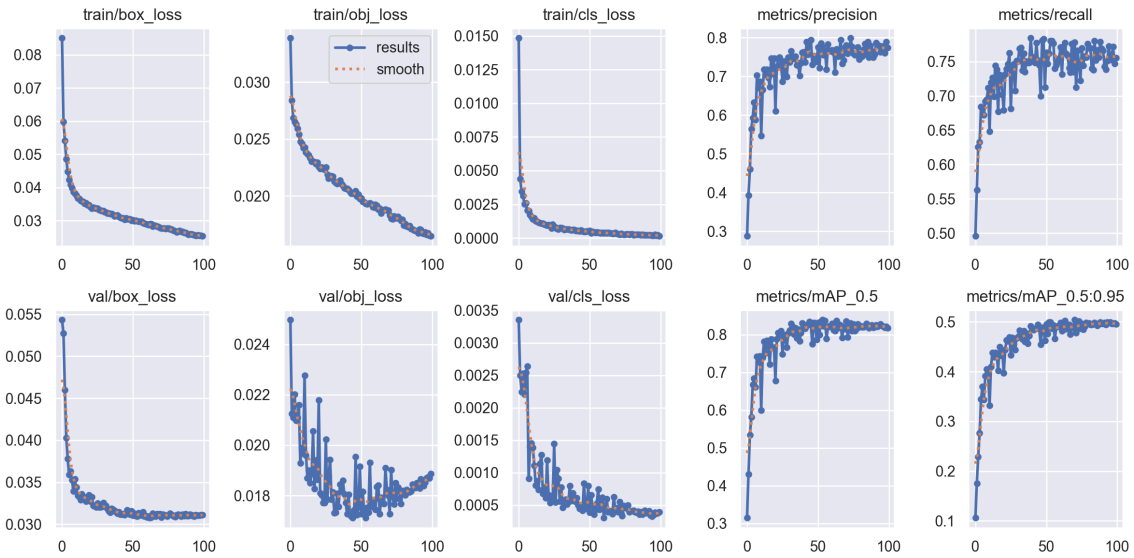
```
                if overlap_ratio >= 0.3:  # 겹침 비율 30% 이상
                    if face_status[face_id]["start_time"] is None:
                        face_status[face_id]["start_time"] = time.time()

                    elapsed_time = time.time() - face_status[face_id]
                    if elapsed_time >= 3:  # 3초 이상 겹쳤을 경우 블러 h
                        face_status[face_id]["blurred"] = False
                        face_status[face_id]["blur_removed"] = True  #

            # 블러 처리 또는 해제
            if face_status[face_id]["blurred"] and not face_status[fa
                # 얼굴에 블러 처리
                face_roi = frame[y1:y2, x1:x2]  # 얼굴 영역
                blurred_face = cv2.GaussianBlur(face_roi, (51, 51), 30)
                frame[y1:y2, x1:x2] = blurred_face  # 블러 처리된 얼굴 다
            else:
                # 블러 해제된 얼굴은 다시 블러 처리하지 않음
                cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)  #

        elif int(cls) == 0:  # Hand (클래스 ID 0)
            # 손 객체 시각화: 경계 상자와 확신도 표시
            color = (0, 255, 0)  # 손 객체는 초록색
            label = f'Hand {conf:.2f}'  # 손 객체 라벨
            cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)  # 경계 상
            cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSH

    # FPS 계산 및 표시
    curr_time = time.time()
    fps = 1 / (curr_time - prev_time)
    prev_time = curr_time
    cv2.putText(frame, f'FPS: {fps:.2f}', (10, 30), cv2.FONT_HERS

    # 화면에 표시
    cv2.imshow('YOLOv5 Webcam Detection with Face Blur and

    # ESC 키를 누르면 종료
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

```
# 웹캠 및 OpenCV 창 닫기
cap.release()
cv2.destroyAllWindows()
```

- Issues
  - 블러 해제 후 유지 X
    - Maybe: 객체에 임의로 고유 번호를 부여해서 다시 작업한다면 가능할지도?

▼ 추가 시도

- OpenCV로 detect한 얼굴 데이터로 학습한 결과 다소 불안정
- coco dataset의 person(0) 데이터를 mediapipe로 얼굴 라벨링 한 후 hand data와 함께 모델 학습



- 2차 시도와 동일 과정 진행

📌 주요 기능 2

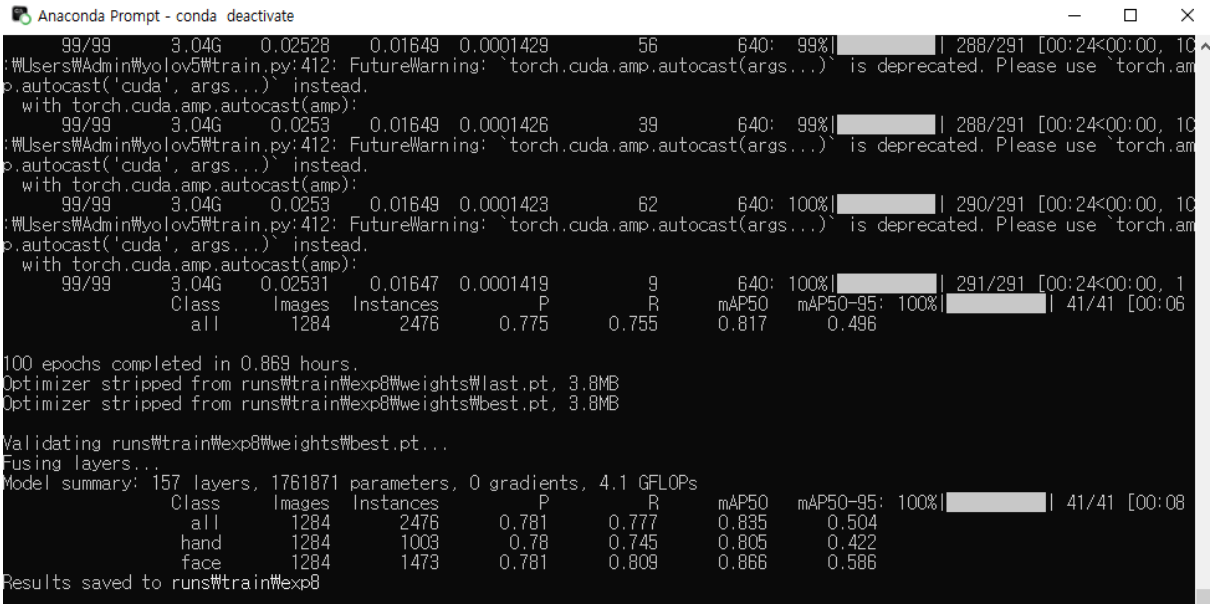
1. Yolov5n모델을 이용한 얼굴&손 탐지
2. OpenCV로 웹캠 화면 출력 및 바운딩 박스 표시
3. hand 박스의 40% 이상이 face에 2초 이상 겹치는 경우 3초 카운트다운 후 스크린샷
4. 지정 파일에 자동 저장

기능 1 에서 학습시킨 모델(hand_face_best.pt) 사용

```
############################################################
##############손 감지(2초) 후 이미지 저장 (countdown)#############
############################################################


import torch
import cv2
import time

# YOLOv5 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path='hand_face_best.pt

# 웹캠 열기
cap = cv2.VideoCapture(0)  # 0: 기본 웹캠
if not cap.isOpened():
    print("Error: 웹캠을 열 수 없습니다.")
    exit()

# 이미지 저장 폴더 설정 (폴더가 없으면 생성)
output_dir = "captured_images"
```

```
# 캡처된 이미지 저장을 위한 변수 설정
capture_triggered = False
capture_time = 0
capture_index = 1  # 저장할 이미지 번호 초기화
hand_detection_time = 0  # 손이 감지된 시간 추적용 변수

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: 프레임을 읽을 수 없습니다.")
        break

    # YOLOv5 모델로 프레임 예측
    results = model(frame)

    # 탐지 결과에서 손 객체 감지 확인
    detected_objects = results.pandas().xywh[0]  # 'xywh' 컬럼을 가진 데이터프러
    hand_detected = False
    for _, obj in detected_objects.iterrows():
        if obj['name'] == 'hand':  # 객체 이름이 'hand'인 경우
            hand_detected = True
            break

    # 손 객체가 2초 이상 감지되면 카운트다운 시작
    if hand_detected:
        if hand_detection_time == 0:
            hand_detection_time = time.time()  # 손이 처음 감지된 시간 기록
        elif time.time() - hand_detection_time >= 2:  # 손이 2초 이상 감지된 경우
            if not capture_triggered:
                capture_triggered = True
                capture_time = time.time()
                print("손 객체가 2초 이상 감지됨. 3초 카운트다운 시작!")

    else:
        hand_detection_time = 0  # 손이 감지되지 않으면 시간 초기화

    if capture_triggered:
```

```
            # 3초 카운트다운
            countdown_time = 3 - int(time.time() - capture_time)
            if countdown_time > 0:
                cv2.putText(frame, f"Capture after {countdown_time}sec", (50, 50), cv
            else:
                # 순차적인 파일 이름으로 저장
                capture_filename = f"captured_images/captured_image_{capture_inde
                cv2.imwrite(capture_filename, frame)
                print(f"이미지 저장됨: {capture_filename}")

                # 파일 번호 증가
                capture_index += 1
                capture_triggered = False  # 카운트다운이 끝났으므로 리셋

        # 탐지된 결과를 이미지에 시각화
        annotated_frame = results.render()[0]  # YOLOv5의 렌더링된 결과 가져오기

        # 화면에 표시
        cv2.imshow('YOLOv5 Webcam Detection', annotated_frame)

        # ESC 키를 누르면 종료
        if cv2.waitKey(1) & 0xFF == 27:
            break

# 웹캠 및 OpenCV 창 닫기
cap.release()
cv2.destroyAllWindows()
```


Anaconda Prompt - conda deactivate
99/99 3.04G 0.02528 0.01649 0.0001429 56 640: 99%| | 288/291 [00:24<00:00, 1C
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
with torch.cuda.amp.autocast(amp):
99/99 3.04G 0.0253 0.01649 0.0001426 39 640: 99%| | 288/291 [00:24<00:00, 1C
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
with torch.cuda.amp.autocast(amp):
99/99 3.04G 0.0253 0.01649 0.0001423 62 640: 100%| | 290/291 [00:24<00:00, 1C
:₩Users₩Admin₩yolov5₩train.py:412: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.am
p.autocast('cuda', args...)` instead.
with torch.cuda.amp.autocast(amp):
99/99 3.04G 0.02531 0.01647 0.0001419 9 640: 100%| | 291/291 [00:24<00:00, 1
Class Images Instances P R mAP50 mAP50-95: 100%| | 41/41 [00:06
all 1284 2476 0.775 0.755 0.817 0.496
100 epochs completed in 0.869 hours.
Optimizer stripped from runs₩train₩exp8₩weights₩last.pt, 3.8MB
Optimizer stripped from runs₩train₩exp8₩weights₩best.pt, 3.8MB
Validating runs₩train₩exp8₩weights₩best.pt...
Fusing layers...
Model summary: 157 layers, 1761871 parameters, 0 gradients, 4.1 GFLOPs
Class Images Instances P R mAP50 mAP50-95: 100%| | 41/41 [00:08
all 1284 2476 0.781 0.777 0.835 0.504
hand 1284 1003 0.78 0.745 0.805 0.422
face 1284 1473 0.781 0.809 0.866 0.586
Results saved to runs₩train₩exp8