

dmc 이종진

# 실시간 화재·연기 경고 알림이

YOLO 기반 실시간 화재/연기 탐지 + 경고 조건 감지 + 이메일 알림 발송

2025.06.19

# 목 차

01. 프로젝트 개요
02. 기획 배경 및 동기
03. 타겟 사용자 및 페르소나
04. 서비스 / 기능 소개
05. 개발 내용
06. 어려움과 극복 방법
07. 영상 시연
08. 개선 사항

## 02. 기획 배경 및 동기



### 문제 인식 :

요식업 현장은 주방 화기 사용이 잦아 항상 화재 위험에 노출되어 있음. 실제 현장에서 작은 불씨가 큰 사고로 이어졌던 경험이 있어 예방의 중요성을 크게 느꼈음.



### 선정 이유 :

기존의 화재 경보기는 주로 ‘연기 센서’나 ‘열감지 센서’에 의존하는 구조였음. 이는 상황을 종합적으로 인식하지 못하고, 환경에 따라 오작동하거나 놓치는 경우가 있었음.

따라서 기존 센서 감지에 더해서 ‘카메라 영상 기반’으로 객체 수나 확산 패턴까지 고려해 판단하는 스마트한 시스템이 필요하다고 느꼈음.



### 계기 :

현장 근무 했을 적에 저녁 영업을 하는 업장이 여서 출근을 늦게 하는데 업장은 2층에 위치해 있었고, 점심 쯤에(아무도 출근하지 않은 시간) 밖에서 지나가던 행인이 담배 피우고 담배꽁초를 그냥 던지고 갔는데 그게 밖에 있던 실외기로 불이 들어가면서 실외기가 터지고 2층 가게 까지 다 탔었던 경험이 있음. 상권이 저녁 장사하는 거리라 사람이 거의 없었기 때문에 초기에 불이 커지는 걸 막지도 못해서 피해가 커짐. 이 밖에도 큰 피해로 번지지 않았지만 다양한 사건 사고들이 생겼음.

## 03. 페르소나



이수진

이름 : 이수진

나이 : 28

직업 : 헤어 디자이너

거주지 : 서울 은평구

사례 :

미용실 마감을 하고 퇴근 후 2시간 정도가 지나서 드라이기가 꽂혀있던 멀티탭 합선으로 화재가 나서 미용실이 다 타버림.

새로 오픈한 미용실에는 ‘실시간 화재 경고 알림이’를 설치하고 오픈 3개월 째 되는 날 이번엔 고데기 합선이 일어났는데 실시간으로 경고 알림을 받고 미용실이 다 타는 것을 막을 수 있었음.

## 03. 페르소나



전해원

이름 : 전해원

나이 : 30

직업 : AI 개발자

거주지 : 서울 송파구

사례 :

회사에서 AI 모델을 학습 시켜 놓고 퇴근하던 중 버티지 못한 그래픽카드가 터짐.

하지만 회사에 설치되어있던 ‘실시간 화재 경고 알림이’가 실시간으로 경고를 해준 덕분에 본인의 자리만 전소하고 끝나면서 큰 피해로 번지는 걸 막을 수 있었음.

## 03. 페르소나



teddy

이름 : 테디유

나이 : 35

직업 : 용접공

거주지 : 울산광영시 동구

사례 :

용접하면서 생긴 불티가 옆으로 날라가면서 불이 붙었지만 용접을 하고 있던 테디유씨는 불이 붙은 걸 보지 못했다.

하지만 작업실에 ‘실시간 화제 경고 알림이’가 설치되어 있었고, 애플워치에 울린 경고 알림으로 불이 붙었다는 것을 바로 확인하게 되었고 큰 피해를 막을 수 있었다.

## 03. 페르소나



김하나

이름 : 김하나

나이 : 57

직업 : 가정 주부

거주지 : 화성시 동탄

사례 :

저녁을 만들던 중 깜빡 잠이 들어 냄비가 타고 있는데 연기가 나오기 시작하는 걸 집에 있던 홈캠에 설치된 ‘실시간 화재 경고 알림이’가 알림을 울려서 큰 화재로 번질 수 있었던 사고를 조기에 막을 수 있었음.

## 04. 서비스 / 기능 소개



### 전체 흐름 요약

1. 사용자는 웹캠 혹은 녹화 영상을 선택하여 시스템에 입력
2. YOLOv8 기반의 불/연기 객체 탐지 모델이 실시간 추론 수행
3. 위험 판단 로직에 따라 경고 조건을 만족할 경우 Streamlit UI에 표시 및 Gmail 경고 메일 전송
4. 대시보드 탭에서 전체 탐지 내역과 위험 경고 누적 내역을 실시간 확인 가능



### 구현된 주요 기능

- YOLO 기반 객체 탐지 : fire, smoke 클래스 식별
- 위험 판단 로직 :
  - smoke 객체 3개 이상 → 최초 경고
  - 불 면적 증가율, smoke 마스크 성장률, 농도 변화율 기반 위험 판단
- 이메일 알림 기능 : 경고 발생 시 자동 메일 전송 (이미지 포함)
- Streamlit 대시보드 UI :
  - 탐지 로그 (클래스, 좌표, 신뢰도)
  - 경고 로그 (발생 원인, 위험 레벨, 누적 횟수)
- 입력 선택 기능 : 웹캠/ 영상 업로드 택 1 가능

# 04. 서비스 / 기능 소개

## 🔥 화재 경고 알림 서비스 데모

화재/연기 실시간 탐지 시스템

- 사용 모델: YOLOv8s, YOLO11n-seg

＊ 카메라 위치 선택

주방 (cam01)

전체 탐지 로그 경고 대시보드

＊ 입력 속성 선택

웹캠  
 영상 업로드

웹캠 시작



## 🔥 화재 경고 알림 서비스 데모

화재/연기 실시간 탐지 시스템

- 사용 모델: YOLOv8s, YOLO11n-seg

＊ 카메라 위치 선택

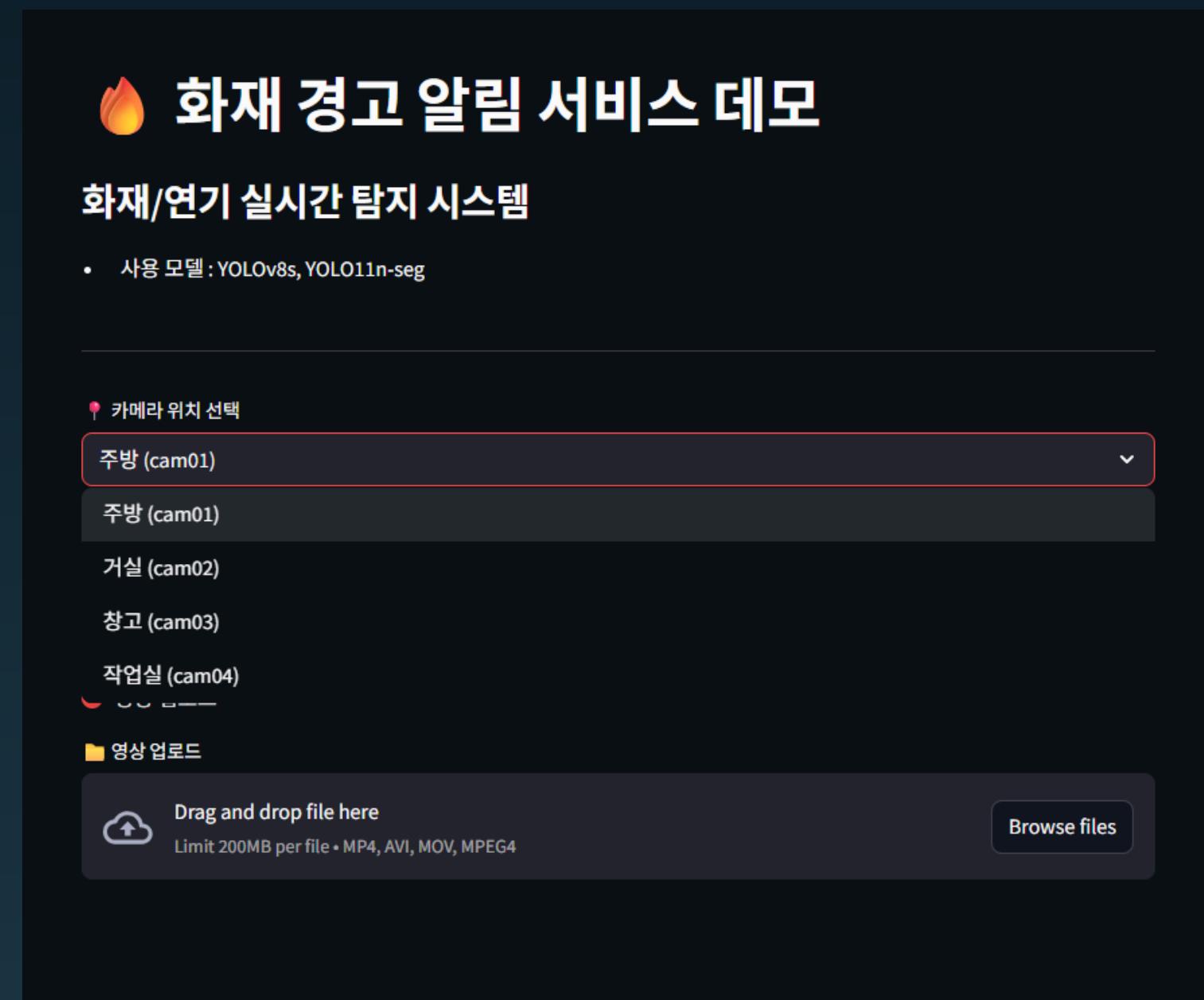
주방 (cam01)

주방 (cam01)  
거실 (cam02)  
창고 (cam03)  
작업실 (cam04)

＊ 영상 업로드

Drag and drop file here  
Limit 200MB per file • MP4, AVI, MOV, MPEG4

Browse files



# 05. 개발 내용



## 사용한 데이터셋 및 평가 지표

- 데이터 셋 : Roboflow - fire/smoke detections & segmentation
- 평가 지표 : mAP@0.5, Recall, IoU, 프레임당 FPS, 최초 경고 시간



## 모델 아키텍처 개요

- YOLOv8s로 불/연기 탐지 → 객체별 박스 및 confidence 획득
- YOLO11n-seg로 연기 mask 추출 → 면적 기반 성장 추적
- 두 모델의 결과를 조합 → 위험 패턴 분석 로직에 적용



## 사용한 기술 스택

- 언어 및 환경 : Python, Streamlit
- 영상 처리 : OpenCV
- AI 모델 : YOLO (Ultralytics)
  - YOLOv8s : Detection
  - YOLO11n-seg : Segmentation
- 후처리 및 양상블 : ensemble-boxes (WBF)
- 데이터 처리 : Pandas, Numpy
- 이메일 알림 : Gmail SMTP

# 05. 개발 내용



## 일자별 개발 내용

Day 1

- 🔍 아이템 기획, 선정 이유, 개발 계획

Day 4

- 🧪 실시간 예측 성능 측정, 오탐지 발생 → 개선 방향 도출

Day 7

- 🌟 경고 판단 로직 설계 + 이미지 포함 Gmail 연동

Day 2

- 🛠️ 가상 환경 세팅, Ultralytics 설치, 데이터 구성 및 YOLO 구조 분석

Day 5

- 📈 모델 성능 개선 (증강, 파라미터 조정), segmentation 학습

Day 8

- ✳️ 코드 모듈화 + 기능별 구조화 + 통합 테스트

Day 1 🔍 아이템 기획, 선정 이유, 개발 계획	Day 2 🛠️ 가상 환경 세팅, Ultralytics 설치, 데이터 구성 및 YOLO 구조 분석	Day 3 📊 YOLOv8s 및 YOL011n-seg 학습 시작, 초기 mAP/Recall 평가
Day 4 🧪 실시간 예측 성능 측정, 오탐지 발생 → 개선 방향 도출	Day 5 📈 모델 성능 개선 (증강, 파라미터 조정), segmentation 학습	Day 6 💻 Streamlit 대시보드 구현, 탐지 결과 로그 표시 기능 개발
Day 7 🌟 경고 판단 로직 설계 + 이미지 포함 Gmail 연동	Day 8 ✳️ 코드 모듈화 + 기능별 구조화 + 통합 테스트	Day 9 🎥 데모 영상 제작 및 발표 자료 최종 정리

- Notion link

# 05. 모델 개발 내용



## YOLOv8s

- 모델명 : YOLOv8s
- 용도 : 불/연기 탐지 (Detection)
- 학습 데이터 : [FireSmokeDetection-v3](#)  
링크
- 주요 설정 및 augmentation :
  - roboflow - Dataset, Model 탭 참조

Python ↗

```
from ultralytics import YOLO

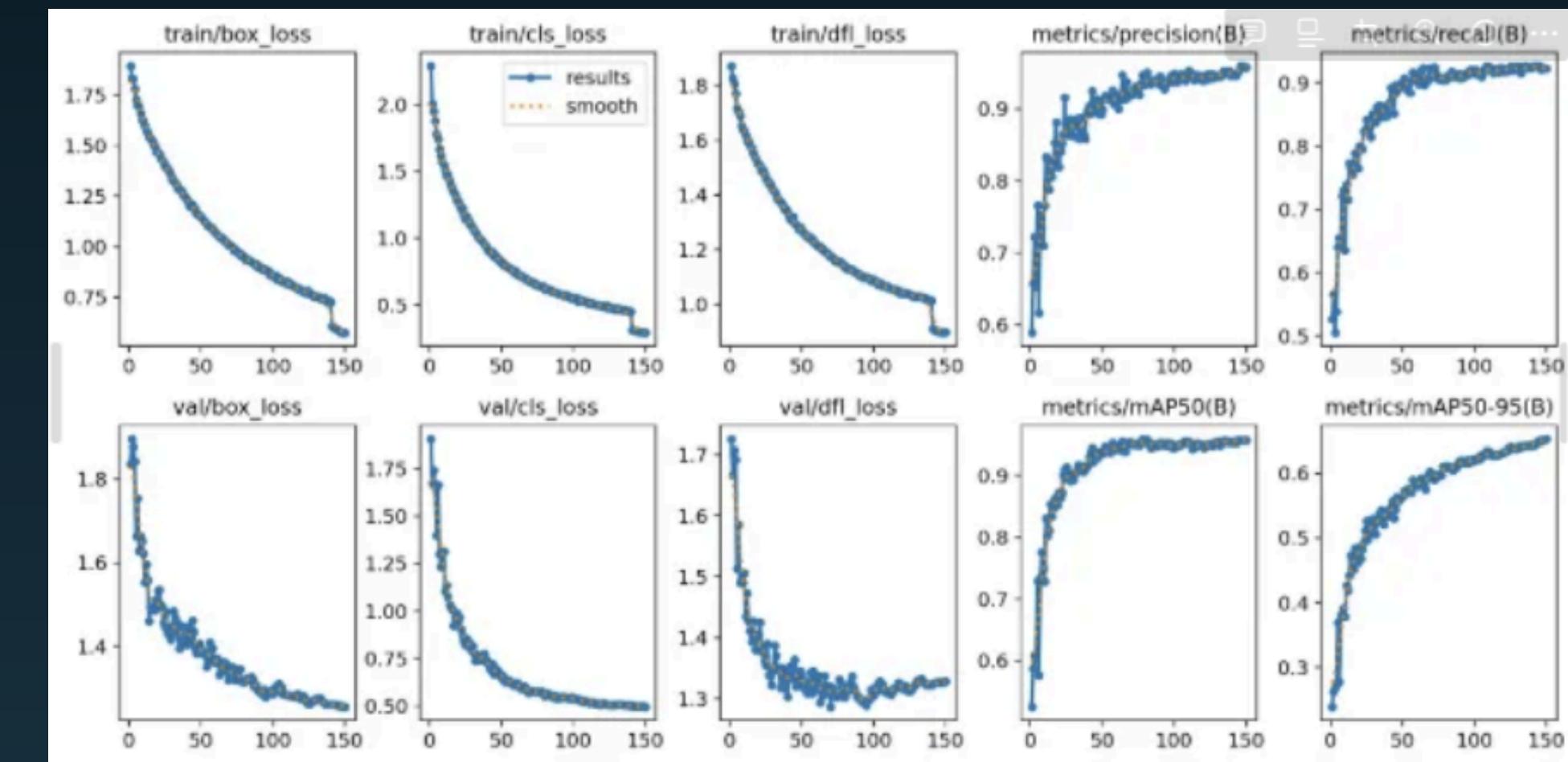
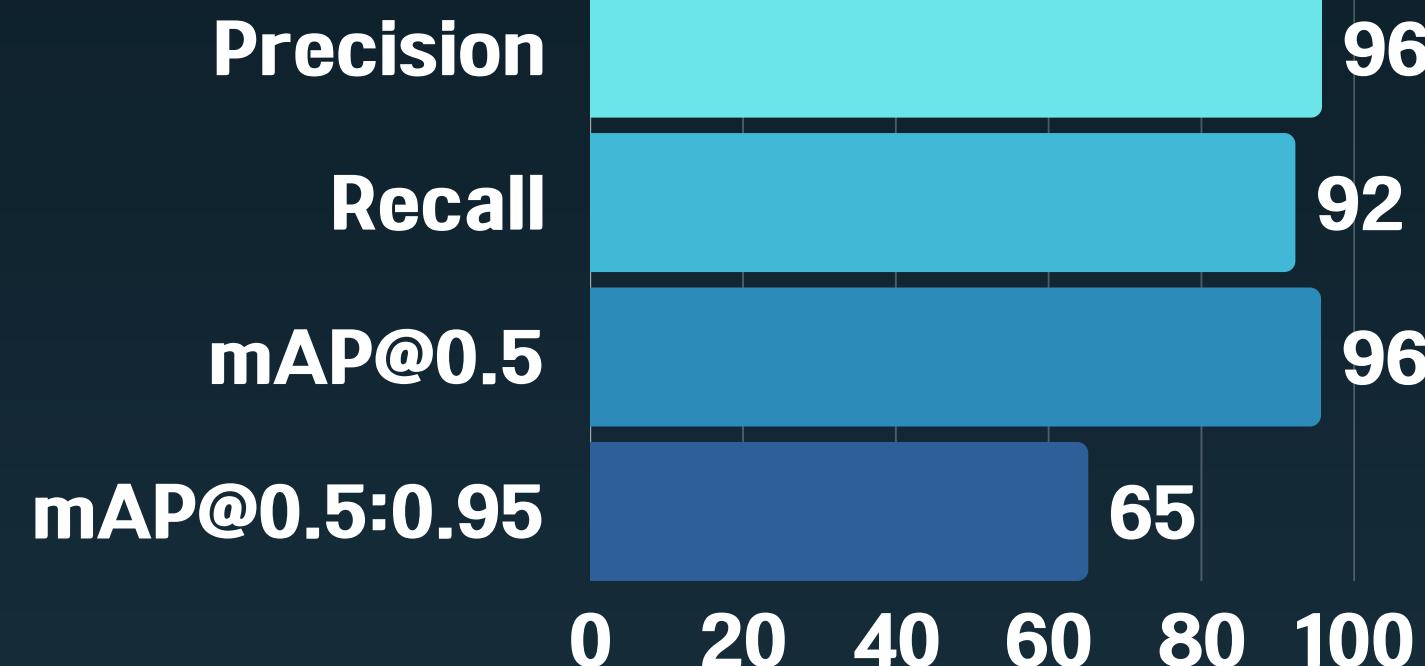
model = YOLO('yolov8s.pt')

model.train(
    data='FireSmokeDetection-v3/data.yaml', # dataset.yaml 경로
    epochs=150,
    imgsz=640,
    batch=16,
    name='firesmoke_detect_v8s',
    optimizer='AdamW',
    lr0=0.001,
    patience=30,
    weight_decay=0.0005,
    hsv_h=0.015,
    hsv_s=0.2,
    hsv_v=0.2,
    degrees=0.0,
    translate=0.1,
    scale=0.15,
    shear=3.0,
    flipud=0.0,
    fliplr=0.5,
    augment=True,
    device=0,
    pretrained=True,
    seed=42
)
```

# 05. 모델 개발 내용



YOLOv8s



# 05. 모델 개발 내용



## YOLOv11n-seg

- 모델명 : YOLOv11n-seg
- 용도 : Smoke 영역(Mask) 추출 및 확산 판단용
- 학습 데이터 : [Fire and Smoke Segmentation v5 링크](#)
- 주요 설정 및 augmentation :
  - roboflow - Dataset, Model 탭 참조

```
Python ▾
from ultralytics import YOLO

model = YOLO('yolo11n-seg.pt')

model.train(
    data='Fire_and_Smoke_seg5v11/data.yaml',
    imgsz=768, #  smoke 픽셀 개선
    epochs=100,
    batch=8, # 이미지 크기 증가에 따라 배치 감소
    name='firesmoke_seg_v11_smoke_focused',
    workers=4,
    device=0,
    lr0=0.002,
    lrf=0.01,
    warmup_epochs=3,
    weight_decay=0.0005,
    optimizer='AdamW',
    patience=50, #  더 많은 학습 허용
    augment=True,

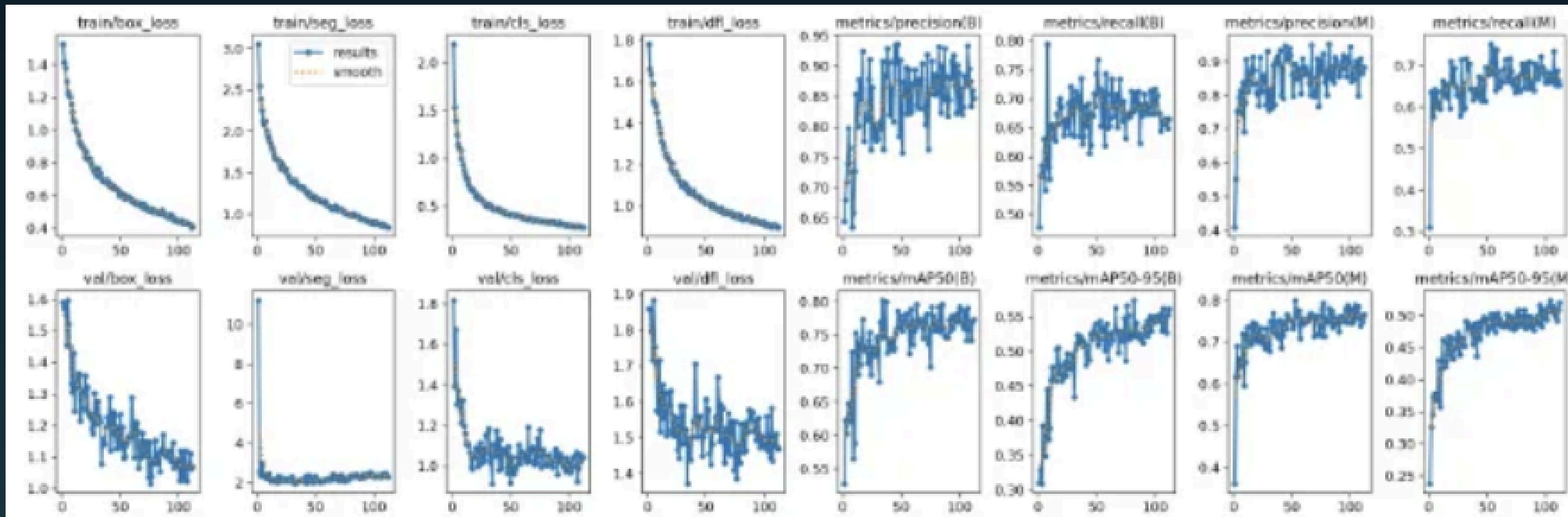
    #  smoke 강조용 증강 추가
    degrees=12,
    shear=3.0,
    perspective=0.0001,
    scale=(0.15, 1.0),
    hsv_h=0.015,
    hsv_s=0.2,
    hsv_v=0.2,
    translate=0.1,
    fliplr=0.5,
    flipud=0.0,
    mosaic=0.0,
    copy_paste=0.0,
    erasing=0.2,
    blur=0.5,
    noise=0.2,
    auto_augment='randaugment',

    # 세그멘테이션 옵션
    overlap_mask=True,
    mask_ratio=4 #  더 고해상도 마스크
)
```

# 05. 모델 개발 내용



YOLOv1n-seg



# 05. 서비스 개발 내용



## 서비스 기능 개발 요약

- 입력 처리 : 웹캠/영상 업로드 선택 기능 구현
- 모델 추론 연결 : 프레임 단위 YOLOv8s/YOLOv11n-seg 추론 적용
- 위험 판단 로직 : 연기 개수, 연기 면적 성장률, 농도 증가율 기반으로 경고 조건 분기 처리
- Streamlit 대시보드
  - 전체 탐지 로그 테이블
  - 위험 경고 발생 내역 테이블
- 최초 경고 및 임계치 도달 시 이미지 포함 Gmail 발송 기능 구현
- 세션 상태 관리 : Streamlit session\_state 활용, 경고 누적 카운팅과 조건별 알림 중복 방지

```
fire-alert-app/
├── app.py                                # Streamlit 메인 실행 앱
├── email_utils.py                         # 이미지 포함 Gmail 전송 유ти
├── requirements.txt                        # 필요한 패키지 목록
└── .env                                    # Gmail 계정 및 비밀번호 등 환경변수

├── utils/
│   ├── detection_utils.py                 # draw_boxes, draw_masks, filter_boxes 등 후처리
│   └── layout.py                          # Streamlit UI 레이아웃 구성 (카메라 선택, 로그인)

└── logic/
    └── alert_logic.py                    # 위험 판단 로직 (최초 경고, 조건분기, 성장률 등)

└── runs/                                   # 모델 weight 저장 폴더
    ├── detect/
    │   ├── firesmoke_detect_v8s/
    │   │   └── weights/best.pt
    │   └── segment/
    │       └── firesmoke_seg_v11_smoke_focus/
    │           └── weights/best.pt
```

# 05. 서비스 개발 내용



## ensemble\_predictions()



## weighted\_boxes\_fusion

- 다수의 박스 예측 결과를 하나의 신뢰도 높은 박스로 통합하는 후 처리 기법
- NMS(Non-Maximum Suppression)와 달리 박스를 제거하지 않고, 위치와 신뢰도를 평균 내어 융합
- 모델 성능이 많이 떨어졌을 때 테스트 성능을 어떻게든 올려보고 싶어서 학습 시켰던 모델들의 결과를 평균을 내어 사용함. (모델 성능이 떨어져서 이렇게 해도 테스트 결과는 좋지 못했음.)
- 현재는 YOLOv8s 단일 모델의 탐지 결과를 WBF로 정제하여 더 정밀한 탐지 결과를 생성하게 함.
- YOLO11n-seg의 결과는 WBF에 포함되지 않으며, 따로 마스크 기반 위험 판단에만 사용됨.

```
def ensemble_predictions(preds, iou_thr=0.5, skip_thr=0.001):
    all_b, all_s, all_l = [], [], []
    for p in preds:
        h, w = p.orig_shape
        b_list, s_list, l_list = [], [], []
        for box, score, lab in zip(
            p.boxes.xyxy.cpu().numpy(),
            p.boxes.conf.cpu().numpy(),
            p.boxes.cls.cpu().numpy(),
        ):
            x1, y1, x2, y2 = box
            b_list.append([x1 / w, y1 / h, x2 / w, y2 / h])
            s_list.append(float(score))
            l_list.append(int(lab))
        all_b.append(b_list)
        all_s.append(s_list)
        all_l.append(l_list)
    fb, fs, fl = weighted_boxes_fusion(
        all_b, all_s, all_l, iou_thr=iou_thr, skip_box_thr=skip_thr
    )
    return filter_boxes(fb, fs, fl)
```

# 05. 서비스 개발 내용



alert\_logic.py



evaluate\_risks\_from\_masks()

- YOLO11n-seg의 마스크 결과를 기반으로 프레임 간 연기 영역의 변화량을 계산하기 위해 구현한 함수
- 연속 프레임 사이에서 연기 영역의 확장률을 추적하여 위험 판단에 요소로 활용됨
- (연기 면적 성장률 > 1.3배) → 위험 인식

```
# —— smoke mask 기반 확장을 계산 ———
def evaluate_risks_from_masks(result, h, w):
    seg_masks = list(result.masks.data)
    agg = np.zeros((h, w), dtype=np.uint8)
    for m in seg_masks:
        m_np = m.cpu().numpy().astype(np.uint8)
        rm = cv2.resize(m_np, (w, h), interpolation=cv2.INTER_NEAREST)
        agg |= (rm > 0.5).astype(np.uint8)
    curr = agg.sum()
    prev = st.session_state.prev_smoke_mask_area
    growth = curr / (prev + 1e-6)
    st.session_state.prev_smoke_mask_area = curr
    return seg_masks, growth
```

# 05. 서비스 개발 내용



alert\_logic.py



evaluate\_risks()

- 함수 내 연기 농도 증가율 계산
- 연기 농도 = ROI 내 평균 밝기의 감소
- 연기 객체가 탐지된 후, 해당 박스 좌표 영역의 grayscale 밝기 평균값을 추출
- 이전 프레임 대비 얼마나 진해졌는지를 연기 농도 증가율로 정의함.
- 값이 1.1 이상일 경우 위험 판단에 반영되며, 1.5 이상이면 danger로 분기됨.

```
def evaluate_risks():
    growth = fire_area / (st.session_state.prev_fire_area + 1e-6)
    st.session_state.prev_fire_area = fire_area

    alert_result = []
    for b, s, l in zip(boxes, scores, labels):
        cls = "fire" if l == 0 else "smoke"
        coord = tuple(round(x, 2) for x in b)
        ig = 1.0
        if cls == "smoke":
            x1, y1, x2, y2 = [int(v * d) for v, d in zip(b, (w, h, w, h))]
            roi = gray[y1:y2, x1:x2]
            if roi.size > 0:
                mi = float(np.mean(roi))
                prev = st.session_state.prev_smoke_intensity.get(str(coord), mi)
                ig = mi / (prev + 1e-6)
                st.session_state.prev_smoke_intensity[str(coord)] = mi
```

# 05. 서비스 개발 내용



alert\_logic.py



evaluate\_risks()

- 함수 내 fire 객체의 box 면적 합산 후, 이전 프레임 대비 면적 증가율 계산
- 불 면적 =  $(x_2 - x_1) * (y_2 - y_1)$ 의 누적 합
- 불 면적 증가율 = 현재 면적 / 이전 면적
- 증가율이 3배 이상이고, 신뢰도가 0.7 이상인 경우 danger 등급으로 판단

```
# —— 탐지된 박스 기반 위험 판단 ———
def evaluate_risks(
    boxes,
    scores,
    labels,
    gray,
    shape,
    vis,
    smoke_growth,
    cam_id,
    allow_fire,
    selected_display,
    start,
    now,
):
    h, w = shape[:2]
    fire_area = sum(
        (b[2] - b[0]) * (b[3] - b[1]) for b, l in zip(boxes, labels) if l == 0
    )
    growth = fire_area / (st.session_state.prev_fire_area + 1e-6)
    st.session_state.prev_fire_area = fire_area

    alert_result = []
    for b, s, l in zip(boxes, scores, labels):
        cls = "fire" if l == 0 else "smoke"
        if growth >= 3 and s >= 0.7:
```

# 05. 서비스 개발 내용



## alert\_logic.py

- 실시간 위험 판단은 단일 조건이 아닌, 객체 수, 면적 증가율, 밝기 변화율, 누적 경고 수 등
- 다양한 요소를 종합적으로 고려해 다단계 경고 시스템을 설계

### 🔥 실시간 위험 판단 로직: 경고 조건 정리

분류	조건	위험 등급	설명
◆ smoke 객체 수	smoke 객체가 3개 이상 탐지 됨	⚠️ 최초 경고 (warning)	check_first_alert() 에서 판단
🔥 fire 탐지	fire가 허용되지 않은 위치에서 신뢰도 $\geq 0.6$ 로 탐지됨	⚠️ warning	예: 거실, 창고 등에서 불감지 시
🔥 fire 면적 급증	fire box 총 면적이 전 프레임 대비 3배 이상 커짐, 신뢰도 $\geq 0.7$	🔴 danger	불이 급격히 번지고 있다고 판단
🔥 smoke mask 성장	YOLOv11n-seg 마스크 면적이 전 프레임 대비 1.5배 이상 증가	⚠️ warning 또는 🔴 danger	연기 확산으로 판단
🔥 smoke 농도 증가율	ROI 영역 평균 밝기값이 이전 대비 1.5배 이상 증가	🔴 danger	연기 농도가 매우 짙어졌다고 판단
🔥 smoke 농도 증가율	ROI 평균 밝기 증가율이 1.1 이상 & 신뢰도 $\geq 0.7$	⚠️ caution	낮은 단계의 연기 확산으로 간주
🔥 smoke mask 확장	YOLOv11n-seg 마스크 면적이 1.3배 이상 증가	⚠️ warning	낮은 단계의 연기 확산으로 간주
🟦 경고 누적 10회	warning이 누적 10회 발생	⚠️ warning 경고 알림	최초 조건 외 누적 판단 용
🟦 위험 누적 5회	danger가 누적 5회 발생	🔴 danger 위험 알림	심각 위험 누적 시 즉시 전송

# 05. 서비스 개발 내용



## email\_utils.py

- Gmail SMTP를 활용하여 경고 발생 시 이미지 포함 이메일 전송 구현
- OpenCV이미지 → JPEG 인코딩 후 이메일 첨부
- Streamlit 내 threading을 사용하여 백그라운드에서 전송처리 → UI 멈춤 없이 실시간 반응 유지

```
# —— 누적 경고 알림 ——
def check_threshold_alerts(vis, selected_display):
    from threading import Thread

    def send(title, msg):
        email_utils.send_alert_email_with_image(title, msg, vis)

    if (
        st.session_state.warning_count == 10
        and not st.session_state.threshold_warning_alerted
    ):
        st.warning("⚠ warning 10회 누적: 경고 알림")
        Thread(
            target=send,
            args=(f"⚠ Warning 10회 누적: {selected_display}", f"현재까지 warning이 10회 누적되었습니다.\n위치: {selected_display}",),
            daemon=True,
        ).start()
        st.session_state.threshold_warning_alerted = True

    if (
        st.session_state.danger_count == 5
        and not st.session_state.threshold_danger_alerted
    ):
        st.error("🔴 danger 5회 누적: 위험 알림")
        Thread(
            target=send,
            args=(f"🔴 Danger 5회 누적: {selected_display}", f"현재까지 danger가 5회 누적되었습니다.\n위치: {selected_display}",),
            daemon=True,
        ).start()
        st.session_state.threshold_danger_alerted = True
```

# 06. 어려운 점 극복 방법



## 모델의 성능 문제

모델의 성능이 너무 낮게 나오는 문제

→ **roboflow** 검증된 데이터셋, 증강 처리, 모델 사용

mAP@0.5 기준 0.95+, smoke segmentation 0.78 달성



## 위험 판단 기준 수립의 어려움

객체 수만 기준으로 경고 시 과도한 오탐 발생

→ 다양한 기준(객체 수 + 면적 + 농도 등)을 조합한 다단계 판단 로직 설계

복합 조건 경고 시스템 설계로 경고 정확도 개선



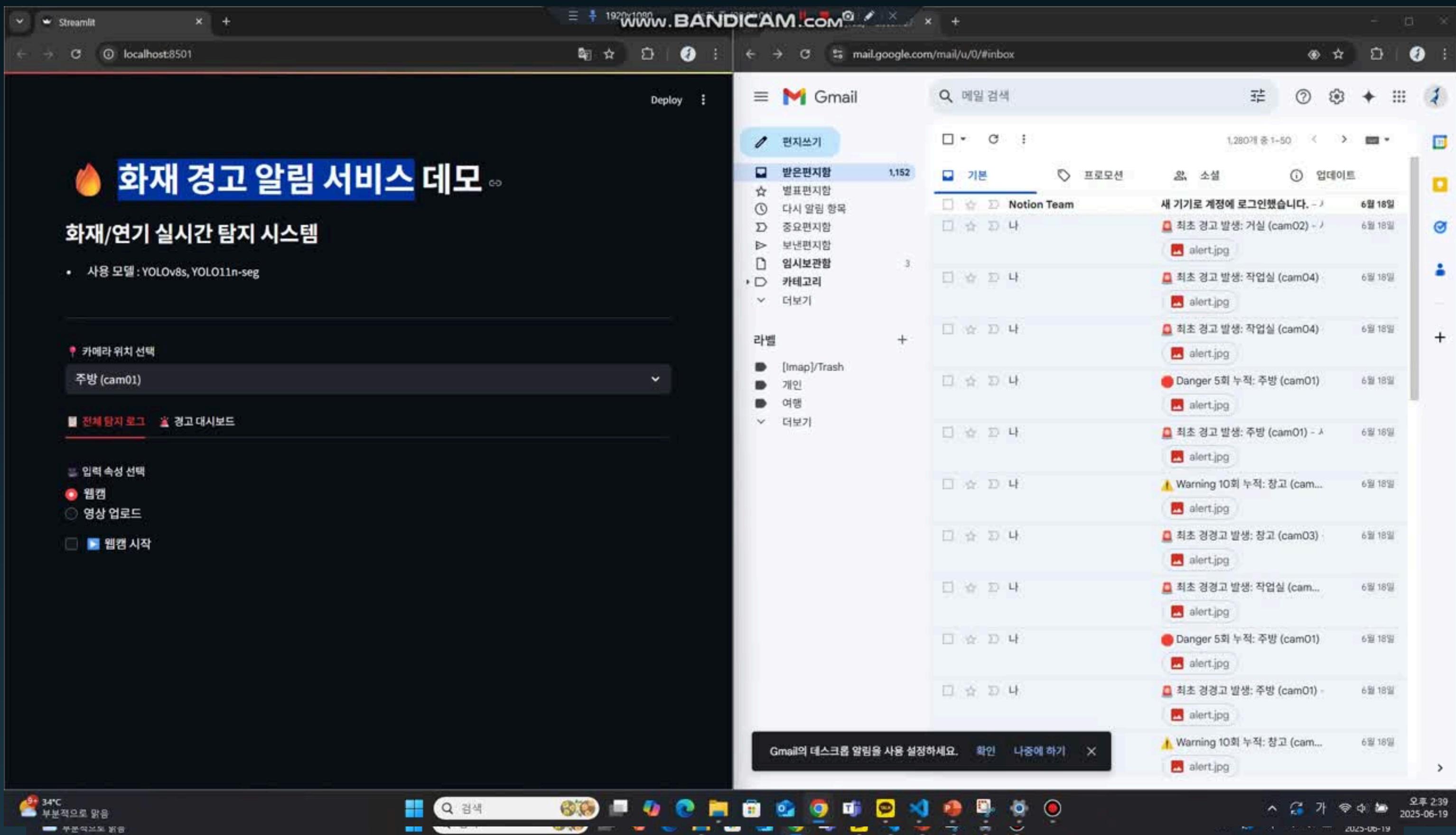
## 실시간 처리 중 UI 멈춤 현상 발생

이메일 전송 과정에서 Streamlit 앱이 일시 정지되는 현상 발견

→ **threading.Thread** 를 활용하여 비동기 전송 처리

사용자 인터페이스가 멈추지 않고 영상 추론이 지속되도록 개선

## 07. 데모 실행 영상



## 08. 느낀점



### 기획 및 개발 측면에서의 배움

이번 10일간의 프로젝트를 통해 “짧은 기간 안에서도 충분히 의미 있는 결과물을 만들 수 있다”는 자신감을 얻었습니다.

실제 홈캠이나 CCTV 화면에 적용해 실환경 테스트까지 해보고 싶었지만, 아직 실력과 구현 여건이 부족해 시도하지 못한 점은 아쉬움으로 남았습니다.

그럼에도 고, Streamlit의 유연성, YOLO의 확장성, Detect + Segment 양상블, 경고 조건 분기 로직, 추론 시간 관리 등 실제 제품화 관점에서 중요한 요소들을 직접 고려하고 해결해 볼 수 있었던 점은 매우 큰 수확이었습니다.

# 08. 느낀점



## 아쉬운 점과 보완하고 싶은 부분

### 1) 데이터 라벨링과 정제의 부족

- 훈련에 사용된 일부 라벨이 부정확하거나 경계가 흐릿해 segmentation 성능이 smoke 클래스에서 제한적이었습니다.
- 다음엔 클래스별로 라벨 품질을 검증하고 보완하는 단계를 먼저 선행하고 싶습니다.

### 2) 모델 성능에 너무 의존한 구조

- 경고 판단 기준이 model output에만 의존하다 보니, false positive/negative에 대한 후처리가 미흡했습니다.
- 이후에는 이상치 필터링이나 이전 프레임 비교 기반의 위험 누적 로직을 보완할 계획입니다.

### 3) 코드 구조화와 재사용성 측면에서의 아쉬움

- 기능 구현에 집중하면서 함수 간 역할 분리나 구조화 설계가 부족했습니다.
- `process_video()` 내에 추론, 시각화, 경고 판단 등 다양한 로직이 혼재되어 디버깅과 유지보수가 어려웠습니다.
- 향후에는 초기부터 기능별 책임을 나누고, 재사용이 가능한 구조로 설계하는 습관을 기르고자 합니다.

## 08. 느낀점



### 가장 기억에 남는 순간

프로젝트 도중 화재 조건 분기 로직에 대해 강사님께 질문을 드렸을 때,  
강사님께서 “전반적으로 불이 나는 영상을 봤을 때, 연기의 농도가 조금씩 짙어지는 것 같아요.”라고 조언해주셨던 순간이 가장 기억에 남습니다.

처음엔 “컴퓨터가 연기의 농도를 어떻게 알 수 있을까?”라는 의문이 들었지만, 이전 수업에서 배운 영상 처리 기법을 떠올리며, 그레이스케일 평균 밝기를 기반으로 농도 계산이 가능하겠다는 실마리를 얻게 되었습니다.

그 한마디 덕분에 연기 농도뿐만 아니라 다른 경고 조건들도 정량적으로 판단할 수 있는 방법을 고민하게 되었고,  
비록 완벽하다고 할 수는 없지만, 지금 이 단계까지 도달할 수 있었던 중요한 전환점이 되었다고 느꼈습니다.

# 08. 개선 사항

## 실시간 스트리밍 영상 연동

현재는 업로드 영상 또는 웹캠 기반으로 작동하지만, 향후 실제 홈캠·CCTV 등의 RTSP 스트리밍을 직접 연동하여 실시간 상황 감지 및 경고 알림 시스템으로 발전 시키고자 합니다.

## 알림 채널 확장 (모바일 앱, 푸시 알림 등)

현재는 이메일 전송 기능만 탑재되어 있으나, 향후 카카오톡/모바일 앱 푸시 알림 등 다양한 실시간 채널로 확장할 예정입니다.

## 지속적인 위험 평가 및 누적 판단

한 순간의 감지가 아닌, 시간에 따른 누적 위험도 분석을 통해 실제 화재 가능성에 더 가까운 정밀한 판단 로직을 구현할 계획입니다.

더 나아가서는 여러 AI 모델 간 협업 추론 구조를 통해, 감지된 영상을 기반으로 화재 여부와 그 원인까지 자동으로 분석하는 시스템에도 도전해보고 싶습니다.

## 탐지 이력 시각화 및 대시보드 구성

탐지 로그 및 경고 이력을 바탕으로 날짜·위치별 발생 현황을 시각적으로 볼 수 있는 모니터링 대시보드도 구상 중입니다.

## 08. 참고 자료/소스 코드



GitHub : <https://github.com/alscom27/firewatch-ai>



참고한 오픈소스 / 기술 자료 :

- [Ultralytics YOLOv8 공식 문서](#)
- [Streamlit Documentation](#)
- [OpenCV Image Processing](#)
- [Gmail SMTP 연동 방법 \(Python email 전송\)](#)
- [Roboflow - FireSmokeDetection Dataset](#)
- [Roboflow - Fire and Smoke Segmentation Dataset](#)

---

# Q&A

질의 응답