

# Safezone

18 June 2024

CloseAI 이상헌 김유철 박준혁 이정훈

# 목차

table of contents

1	개요	5	핵심코드
2	데이터	6	타임라인
3	모델	7	사용 기술
4	프로토타입	8	레퍼런스
		9	QnA

# 1

## Overview

---

# 프로젝트 목표

특정 상황에서 안전 관리에 도움을 주는 모델 개발

사전에 정의한 각종 위험한 상황을 탐지 및 경고

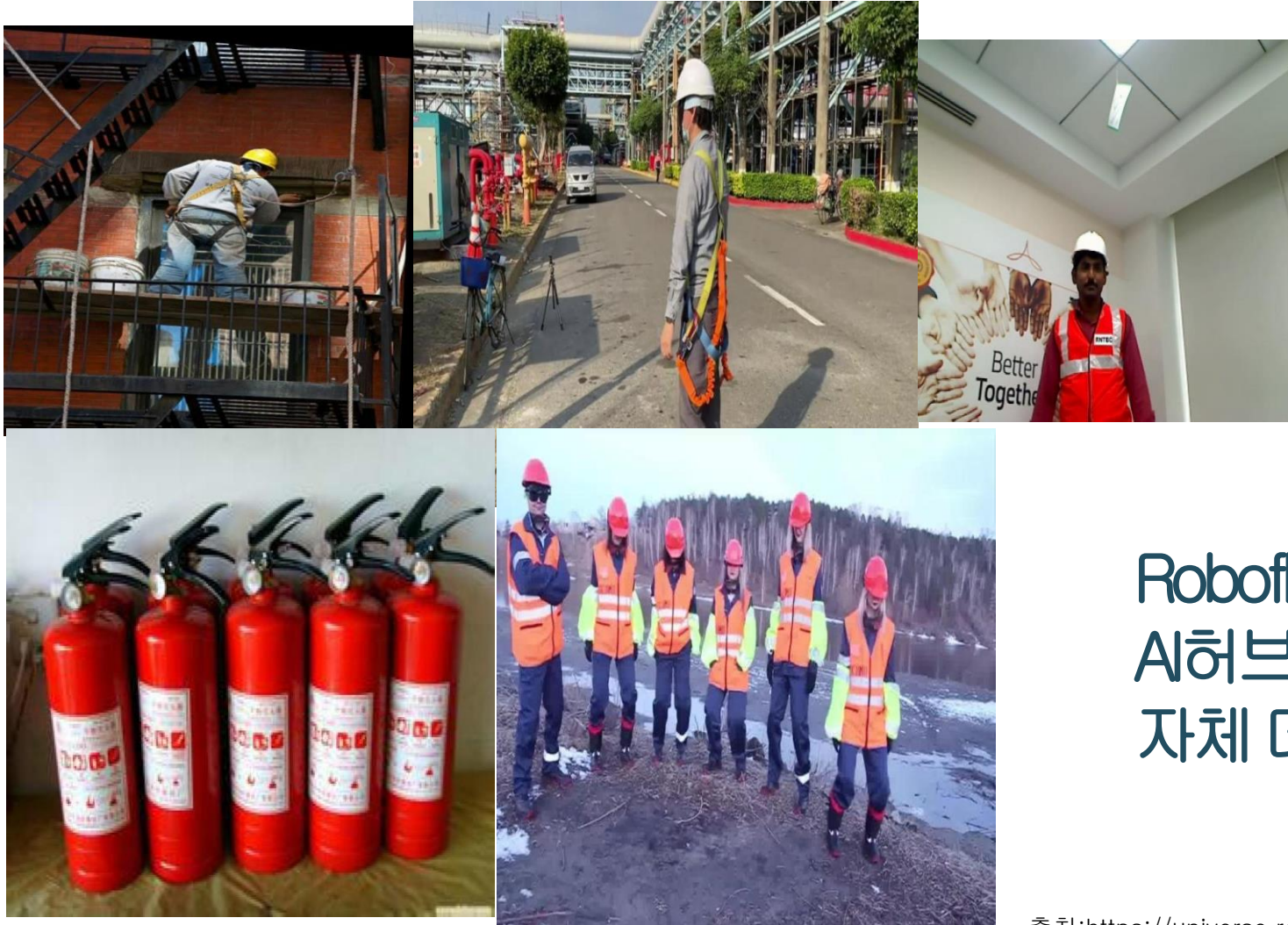
관리자가 모니터링 하고 싶은 객체를 입력 받아 탐지

# 2

## Data

---

# 활용 데이터



Roboflow 데이터셋 수집  
AI허브 안전장비 데이터셋  
자체 데이터셋

출처: <https://universe.roboflow.com/search?q=construction+images%3E5000+model%3A yolov8>

# 3

## Model

---

# Case 1



## 사전에 정의한 특정 이벤트 발생시 경고

문자 수신  
<인화성 물질에 작업자 접근. 확인 요망>



# Case 2



사용자가 원하는 특정 상황을 입력 받아 탐지 가능  
예시) 입력 : 안전모를 착용하지 않은 사람을 찾아줘

# YOLO v8

## 모델 선정 이유

1. 높은 정확도와 속도
2. 다양한 객체 탐지
3. 간편한 학습 및 튜닝
4. 고해상도 이미지 처리

# 다른 모델과 비교

## YOLO 버전 비교

버전	구조와 효율성	성능
YOLO6	경량화와 속도 최적화에 중점	작은 객체 탐지에서 강점, 빠른 속도
YOLO7	효율적 구조와 훈련기법으로 성능 향상	다양한 벤치마크에 서 높은 성능
YOLO8	최신 기술 통합과 모듈화로 성능 극대화	최신 기술 적용으로 최고성능

# 4

프로토타입

---

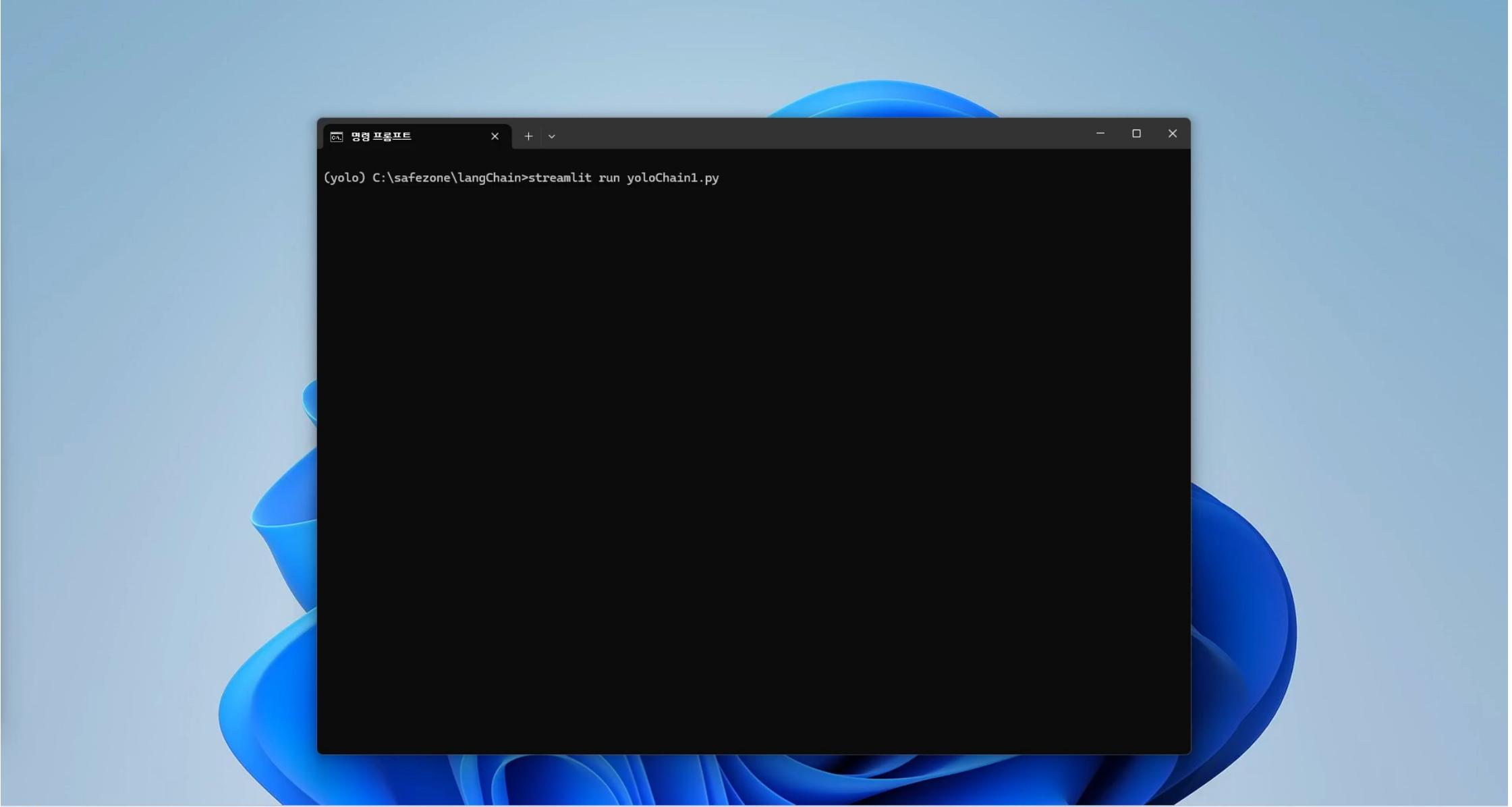
## 프로토타입 구현(CASE 1)

13



# 프로토타입 구현(CASE 2)

14



The screenshot shows a Windows 11 desktop with a blue abstract background. A black command prompt window is open in the center. The window's title bar is dark gray with the text '명령 프롬프트' (Command Prompt) and standard window controls. The command prompt shows the command 'streamlit run yoloChain1.py' being executed. The taskbar at the bottom is light blue and contains the Start button, a search bar with the text '검색', and several application icons including File Explorer, Microsoft Edge, and VS Code. The system tray on the right shows the date and time as '오후 5:19 2024-06-13'.

```
(yolo) C:\safezone\langChain>streamlit run yoloChain1.py
```

# 5

핵심 코드

---

# 핵심 코드

```
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8l.yaml") # build a new model from scratch
model = YOLO("yolov8l.pt") # load a pretrained model (recommended for training)

# Use the model
model.train(data="data.yaml", epochs=500, batch=-1, patience=100) # train the model
metrics = model.val() # evaluate model performance on the validation set
# results = model("https://ultralytics.com/images/bus.jpg") # predict on an image
# results = model("/home/saka/바탕화면/123.jpg") # predict on an image

path = model.export(format="onnx") # export the model to ONNX format
```



# 핵심 코드

```
class YOLODetectionTool:
    def __init__(self, model_path):
        self.model = YOLO(model_path)
        # self.font = ImageFont.truetype("/usr/share/fonts/truetype/nanum/NanumGothic.ttf", 24)
        np.random.seed(0)
        self.colors = np.random.randint(0, 255, size=(len(class_names), 3), dtype=np.uint8)
        self.class_id = None

    def set_class_name(self, class_name):
        if class_name in class_dict:
            self.class_id = class_dict[class_name]
        else:
            raise ValueError(f"클래스 이름 '{class_name}'에 해당하는 ID를 찾을 수 없습니다.")

    def run(self, frame):
        if self.class_id is None:
            raise ValueError("탐지할 클래스가 설정되지 않았습니다.")

        results = self.model(frame)
        predictions = results[0].boxes

        frame_pil = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        draw = ImageDraw.Draw(frame_pil)

        for pred in predictions:
            cls = int(pred.cls[0])
            if cls == self.class_id:
                x1, y1, x2, y2 = map(int, pred.xyxy[0])
                conf = pred.conf[0]
                label = f"{class_names[cls]} {conf:.2f}"
                color = self.colors[cls].tolist()
                draw.rectangle(((x1, y1), (x2, y2)), outline=tuple(color), width=2)
                draw.text((x1, y1 - 50), label, fill=tuple(color) + (255,))

        return cv2.cvtColor(np.array(frame_pil), cv2.COLOR_RGB2BGR)
```

# 핵심 코드

18

```
class CustomLLMChain:
    def __init__(self, llm, prompt, yolo_tool):
        self.llm_chain = LLMChain(llm=llm, prompt=prompt)
        self.yolo_tool = yolo_tool
        self.current_description = ""
        self.current_class_name = ""

    def __call__(self, inputs):
        description = inputs['description']
        frame = inputs['frame']

        # 입력된 설명이 변경되었을 때만 LLM을 호출하여 클래스 이름 추출
        if description != self.current_description:
            response = self.llm_chain.run({"description": description})
            class_name = response.strip()
            self.current_class_name = class_name
            self.current_description = description

            # YOLO Tool에 클래스 이름 설정
            self.yolo_tool.set_class_name(self.current_class_name)

        # YOLO Tool을 사용하여 객체 탐지 및 주석 추가
        annotated_frame = self.yolo_tool.run(frame)
        return {"annotated_frame": annotated_frame}
```

# 6

## Time Line

---

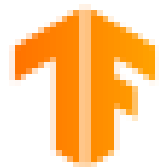
	6 월			
	1 주차	2 주차	3 주차	4 주차
주제 선정	<div></div>			
데이터 수집	<div></div>	<div></div>		
YOLO	<div></div>	<div></div>	<div></div>	
LLM(랭체인)		<div></div>	<div></div>	
모델 고도화			<div></div>	
마무리				<div></div>

# 7

사용 기술

---

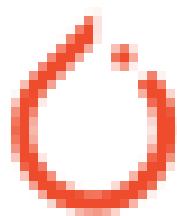
# 사용 기술



TensorFlow



LangChain



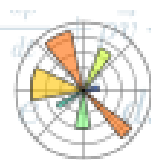
PyTorch



OpenCV



ultralytics  
YOLOv8



matplotlib



# 8

## Reference

---

# Reference

**Real-Time Flying Object Detection with YOLOv8** ( Dillon Reis, 22 May 2024

<https://arxiv.org/pdf/2305.09972>)

**You Only Look Once: Unified, Real-Time Object Detection** ( Joseph Redmon, 9 May 2016

<https://arxiv.org/pdf/1506.02640>)



# 9

QnA

---