프로젝트 배경

- 조리 과정 중 발생하는 유해물질
  - 일산화탄소 (기준치 30배)
  - 이산화 탄소 (기준치 9배)
- 조리실 사망사고 및 질환 발생
  - 폐암사망사고
  - 조리실 유행물질 – 폐질환 상관관계 입증
- 기존 점검 시스템의 한계
  - 조리원 Check-list : 정량적 판단 불가
  - 전문 장비 : 한시적인 점검

문제 정의

조리 시설 내 "유해물질 발생 정보"에 대한 접근의 어려움

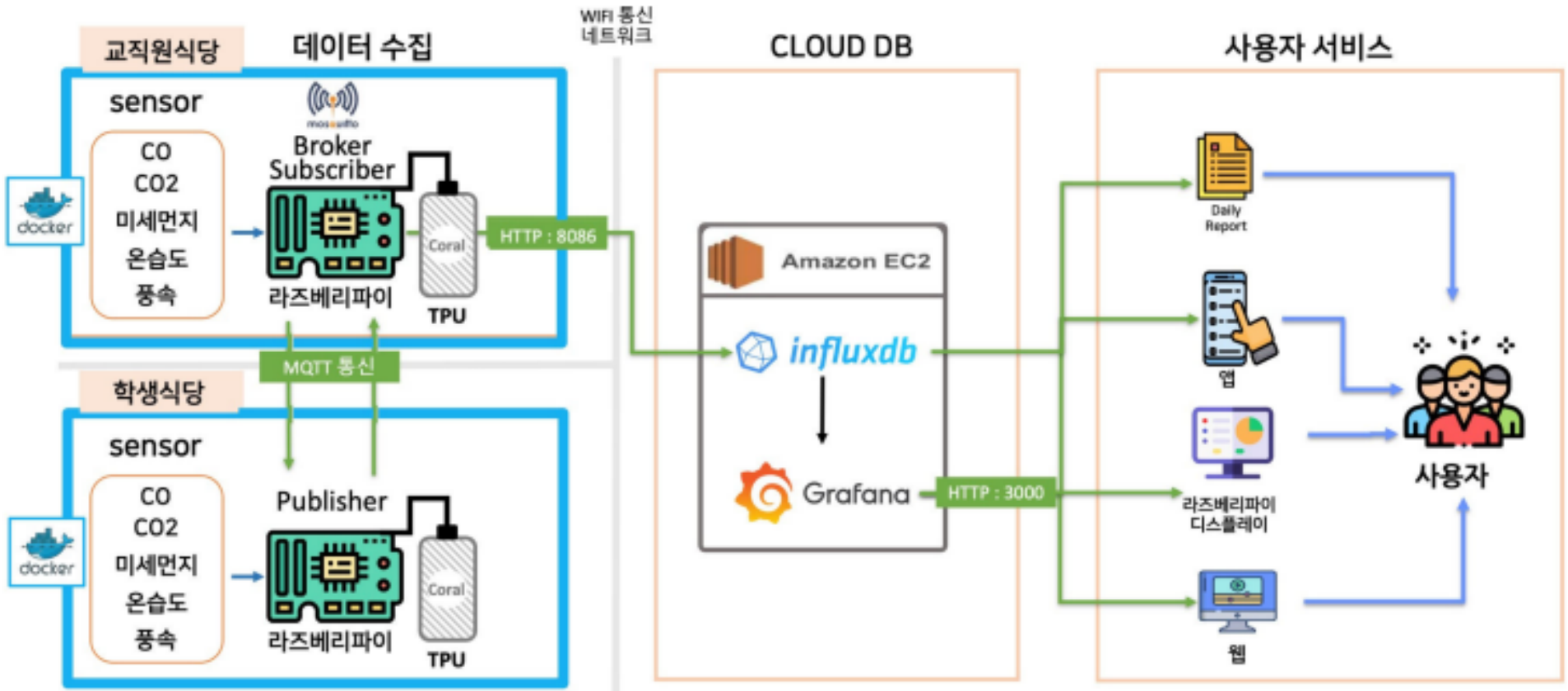
프로젝트 목표

교내 조리실 내 작업 환경 모니터링 시스템 개발

세부 목표

- IoT 네트워크
  - 유해물질 정보 수집 및 모니터링
- 신뢰성 높은 데이터
  - 이상치, 누락 데이터에 처리
- 통합 모니터링
  - 가시성 높은 모니터링 서비스
- 다양한 서비스
  - 사용자 맞춤형 모니터링 서비스

연구의 방법론 - 시스템



조리실에서 수집한 여러 유해물질 수치에 대하여 이상치가 제거된 데이터를
활용하여 사용자에게 적합한 형태의 서비스 제공

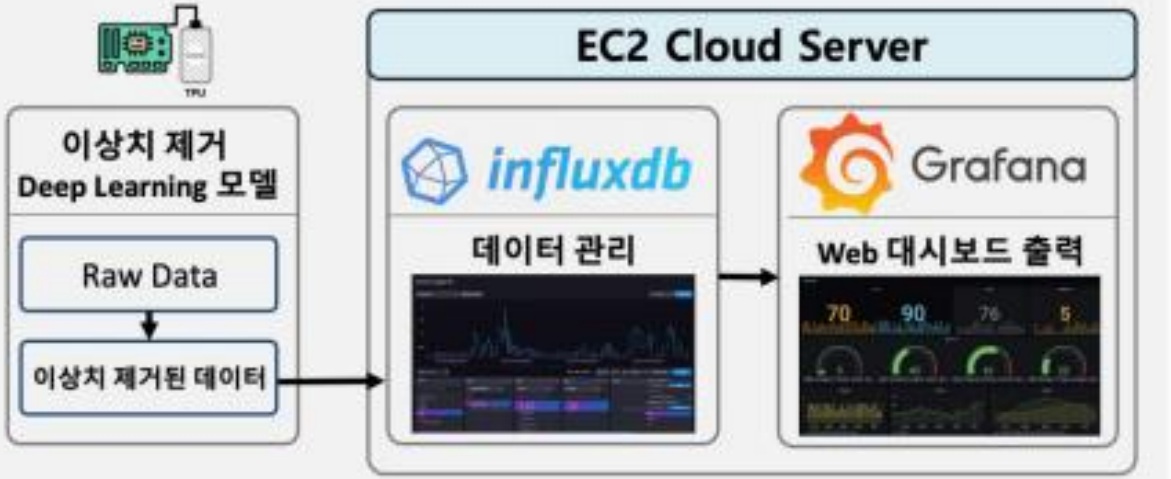
네트워크 및 통신

Device 간 통신

Protocol	특징	사용여부
MQTT	제한된 네트워크에서 경량 M2M 통신을 위해 설계된 발행/구독 메시징 프로토콜	O
CoAP	UDP 사용으로, 패킷 전달에 대한 신뢰성 낮음	X
AMQP	높은 전력 소비와 상대적으로 긴 Latency	X

- MQTT 통신으로 조리실 1, 3층 라즈베리파이 연결
- Broker(3층 라즈베리파이)에 10초 간격 데이터 취합
- 도커를 통해 서비스에 필요한 소프트웨어 배포

IoT 네트워크를 통한 모니터링



- 이상치 제거를 통해 신뢰성 있는 데이터를 클라우드(AWS EC2)에 구축된 InfluxDB에 10초 간격으로 저장
- 유해물질 모니터링을 위한 Grafana 대시보드 UI 커스텀마이징
- Grafana 대시보드를 활용하여 시계열에 따라 저장된 데이터에 대한 모니터링 주기 10초 간격으로 설정

연구의 방법론 - 이상치 제거

1. 데이터 수집

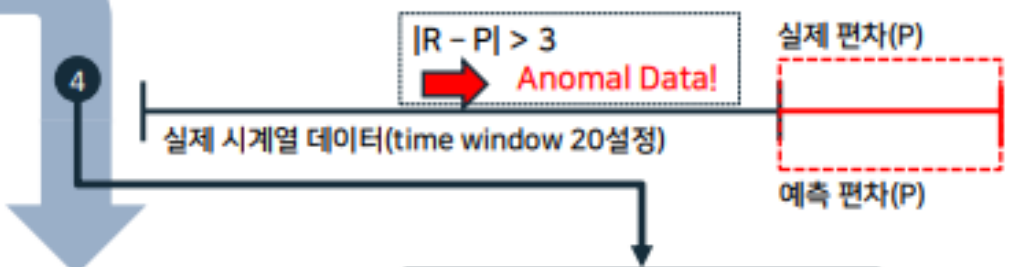
result	table	_time	CO2_ppm	CO_ppm	Humidity_	PM1.0_1f	PM10.0_1f	PM2.5_1f	Tempera.
0_result	0	2021-08-1	5789.65	1.99	58.2248	13	28	24	38.2769
1_result	0	2021-08-1	2319.72	2.72	58.9578	13	27	23	38.2475
2_result	0	2021-08-1	2619.85	2.9	58.2254	13	24	21	38.4157
3_result	0	2021-08-1	1885.82	2.18	57.0443	14	25	21	38.4905
4_result	0	2021-08-1	2124.89	3.26	54.3359	13	28	20	38.6027
5_result	0	2021-08-1	1739.56	2.54	52.8527	15	24	23	38.6294
6_result	0	2021-08-1	5681.53	1.27	52.137	16	32	24	38.6294
7_result	0	2021-08-1	1294.49	3.08	53.5561	15	31	23	38.6294
8_result	0	2021-08-1	1886.14	1.81	54.2992	14	30	22	38.7015
9_result	0	2021-08-1	4893.59	2.36	53.8827	15	21	21	38.7415
10_result	0	2021-08-1	1121.85	2.36	53.6446	14	19	19	38.7415
11_result	0	2021-08-1	1330.86	3.63	53.8781	14	17	17	38.714
12_result	0	2021-08-1	3563.93	1.09	54.931	12	17	17	38.7

2. 데이터 전처리

- 날짜 별 데이터 전 처리
  - CO2, CO, 3개의 미세먼지 농도 데이터 수집
  - 데이터의 비이상적 변동 학습을 위해 편차 데이터 사용
  - 모델을 돌리기 전 정상 데이터 학습을 위해 4분위 계수 이상치 제거 후 학습 데이터 선정

3. 이상치 처리

- 학습 및 검증 방법
  - 기존의 Auto Encoder + 시계열을 다루기 위한 LSTM [1]
  - 편차 데이터 구성(input data)
    ex) 17시 2초 – 17시1초
    17시 21초 – 17시20초
  - 입력 값 : time window 20초 설정 -> 20*5의 데이터
  - 출력 값 : 17시 22초 –17시21초



TensorFlow Lite

4. 결과

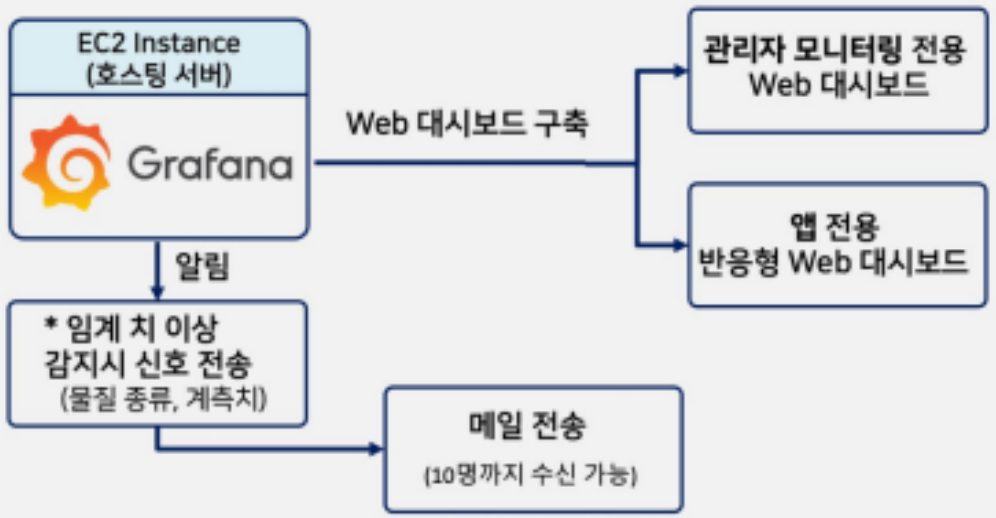
- 이상치 편차 데이터를 해당 열의 중앙값으로 대체
- 서버나 클라우드가 아닌 라즈베리 파이와 같은 물리적인 데이터 저장소에서도 모델이 실행되기 위해 Tensorflow – Lite 를 이용한 CNN 모델 경량화 진행

1- Teng, M. (2010, December). Anomaly detection on time series. In 2010 IEEE International Conference on Progress in Informatics and Computing (Vol. 1, pp. 603-608). IEEE.

연구의 방법론 - 웹, 앱

Web 모니터링

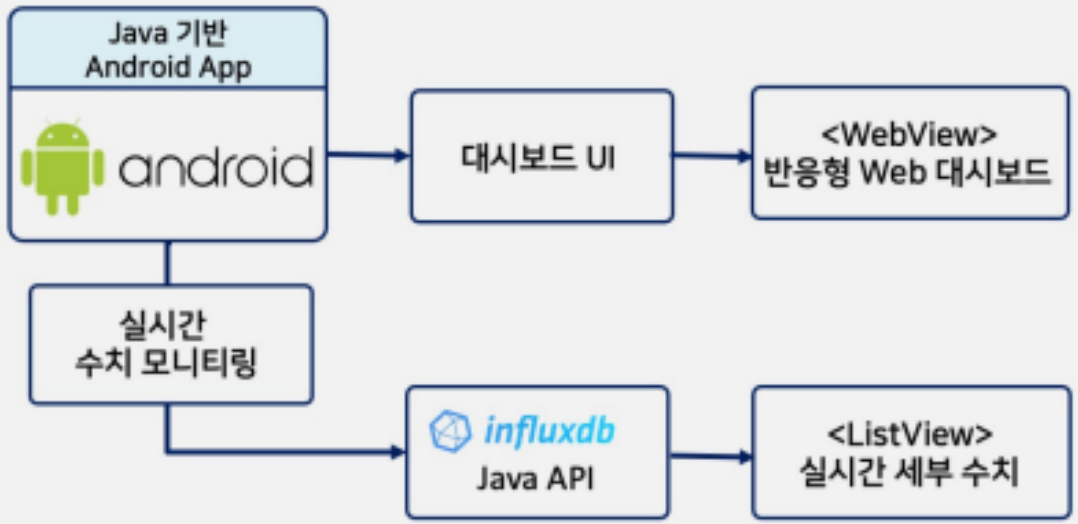
http://<public-ip>:3000



1. EC2 클라우드 서버에 설치된 Grafana를 기반으로 Web 서비스를 제공
   -> web, 앱 전용으로 나누어 대시보드 제작
2. 임계치를 설정하여 이상의 값이 감지될 시, 등록된 관리자 계정에 메일을 전송한다.

* 노동부가 정의한 공기질 기준으로 임계치 설정

App 모니터링



작업자의 입장에서 쉽게 모니터링 할 수 있는 대시보드와 실시간 수치 모니터링 기능을 Java 기반 안드로이드 앱을 통해 구현

➔ Web View를 통해 반응형 Web 대시보드를 출력하고, List View를 통해 실시간 세부 수치를 출력한다.

연구의 방법론 - 메일, 푸시, 보고서

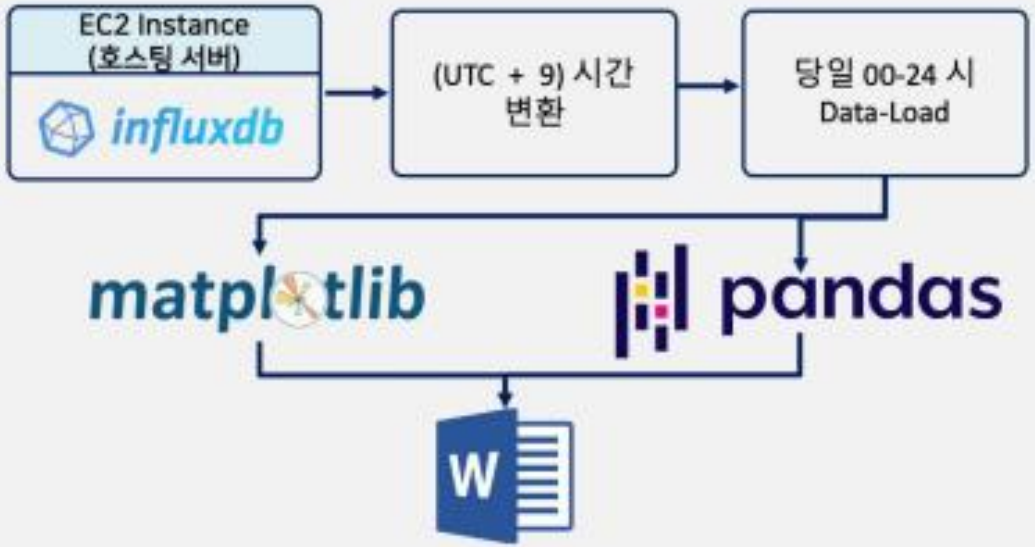
App 푸시 알림

1. 관리자 Application의 암호화된 Token 값에 따라 firebase에 사용자 등록

2. firebase의 Cloud Messaging 기능을 통해 임계치 이상의 유해물질이 발생할 시 종류 및 수치를 push 알림으로 전송

* 노동부가 정의한 공기질 기준으로 임계치 설정

Daily Report



1. Matplotlib으로 임계치 및 일일 누적 데이터 시각화

2. Pandas library를 이용하여 데이터 정제 및 통계 수치 계산

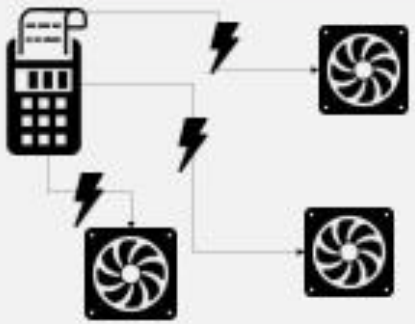
3. Python-docx를 기반으로 그래프 및 통계 자료를 문서로 출력

한계 및 미래연구

한계

- ✓ 모니터링 결과에 대한 사후 조치 부재
- ✓ 교내 조리실 이용객을 위한 모니터링 접근성의 어려움
- ✓ 조리실 내 유해물질 검출 가능 가스 종류의 제한성
- ✓ 선제적 대응이 아닌 단순 모니터링에 제한된 기능

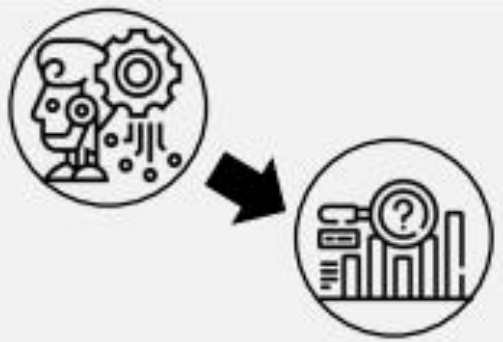
미래 연구



- ✓ Actuator를 이용한 환경 개선 자동화 시스템 구축

- ✓ 교내 조리실 디스플레이 모니터링 대시보드 실시간 송출

- ✓ 잔류 LPG 또는 연기 등 유해물질 검출 센서 추가 설치



- ✓ 경량화 모델을 통한 위험 상황 예측 알고리즘 개발 및 도입

프로젝트 배경

- 잦은 어린이 낙상 사고
  - 미끄러짐, 추락 -> 사고의 40% 이상

- 사고 유형 파악의 필요성
  - 응급 처치 1단계 : 사전 정보 파악
  - 사고 유형 파악 이후 응급 조치 진행

- 현재 낙상 감지 기술 동향
  - 웨어러블 센서
  - 주변 센서
  - 카메라 기반 인식

문제 정의

어린이 낙상 사고 유형 분석 시스템의 부재

프로젝트 목표

놀이터 영상에서 쓰러진 어린이 감지를 통한 낙상 사고 유형 분석 시스템 개발

세부 목표

- 데이터 수집
  - 놀이터 배경
  - 다양한 쓰러짐 자세 및 유형

- 쓰러짐 감지
  - 쓰러짐을 정확하게 감지

- 사교 유형 분석
  - 쓰러짐이 지속되는 경우 사고 원인 분석

- 모니터링 및 알림
  - 쓰러짐 감지 및 유형 분석 통합 모니터링 및 알림 서비스 제공

☑ 연구의 방법론 - App 모니터링 및 Push 알림

App - Push 알림

Input = Realtime Video

YOLOv5 SlowFast Model Load

While YOLOv5

쓰러진 객체 분석

If 쓰러진 객체 탐지

Yes → SlowFast → 사고 유형 분석

No → Pass

SlowFast의 Model Load Latency가 매우 높음.
-) 초기에 model load를 1회 진행하고,
yolo-v5의 Signal에 따라 사고 유형 분석 진행

App 실시간 모니터링

⟨step 1⟩ ⟨step 2⟩ ⟨step 3⟩

RTSP Protocol → [Streaming Server]

VLC → Streaming Server

Video View
RTSP URL :8554

android

Bounding Box 처리된 실시간 영상 RTSP Protocol로 서버에 전송

VLC를 사용하여 Web 에 RTSP Streaming 서버 구축 -)
실시간 영상 Streaming url 생성

Android Studio의 Video View를 통해 App에서 Streaming Player 구현

App Push 알림

⟨step 1⟩ ⟨step 2⟩ ⟨step 3⟩

2초 = 60frame

Cropped Video

[Fall(head)
Misstep(front)
.
.
Fall(head)
Fall(head)]

최빈값 추출
Data type : String

사고 유형 정보
Firebase

Push
-
사고유형

60 frame 에 대해 각각의 사고 유형중 가장 높은 확률을 사고 유형을 list에 저장

저장된 list에서 (60x1) 빈도가 가장 높은 사고 유형 추출

Firebase Cloud Messaging (FCM)을 통해 String type의 결과를 App에 전달 및 push 알림 실행

☑ 연구의 방법론 - 알고리즘 선정 및 통합 모델 사용 이유

쓰러짐 판별

Model	mAP	FPS	GFLOPS x view
YOLOv5	89.5	140	17
Faster-RCNN	73.5	7	118.61
SSD	72.1	58	88.16

사고유형 판별

Model	top-1	top-5	GFLOPS x view
SlowFast	77.9	93.2	65.7 * 30
R(2+1)D	73.9	90.9	304 * 115
I3D	71.6	90.0	216 * N/A

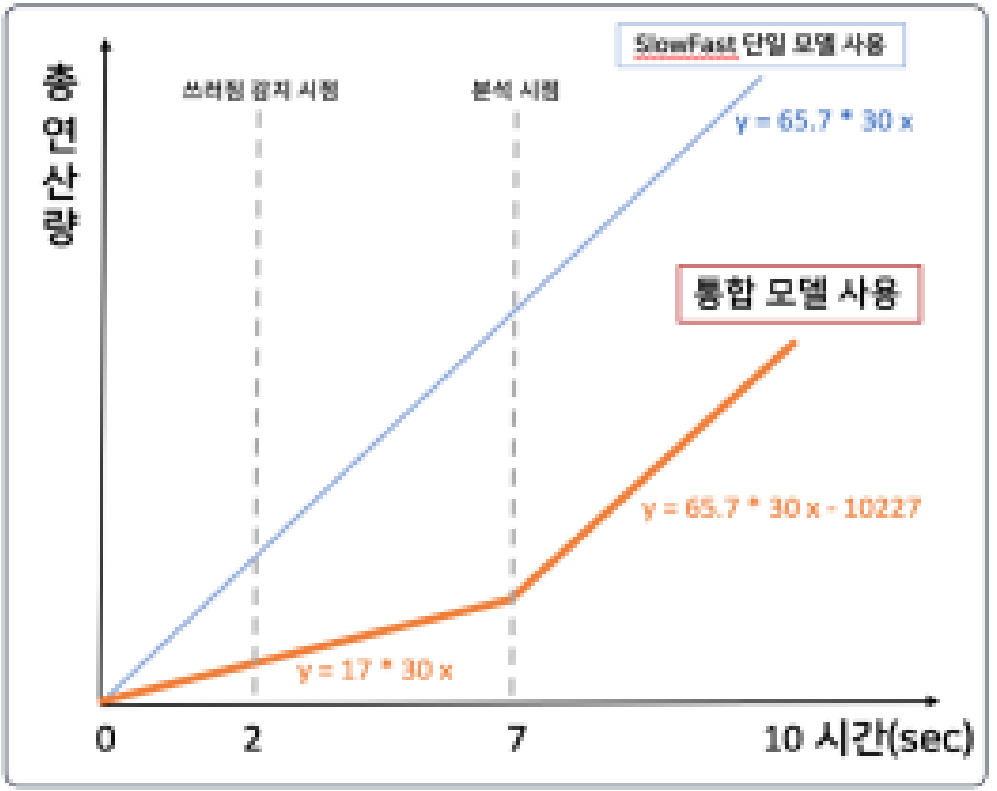
YOLOv5
y = 17 * 30x

SlowFast
y = 65.7 * 30 x

통합모델사용

GFLOPs(Giga **FL**oating point **OP**erations) : 모델의 필요 연산량
-> 연산량이 많아질수록, 학습 시간이 늘어나며 고성능의 전산 자원이 필요하다.

통합 모델 사용 이유



SlowFast 단일 모델로 쓰러짐 감지와 사고 유형 파악 시, 연산량이 많아 성능 저하 문제 발생
통합 모델은 쓰러짐 감지는 YOLO v5, 쓰러짐이 감지되었을 때만 SlowFast를 사용

*) mAP : mean average precision, 알고리즘의 성능을 판단하는 척도
FPS : 초당 프레임

☑ 연구의 방법론 - 학습 및 데이터 분할

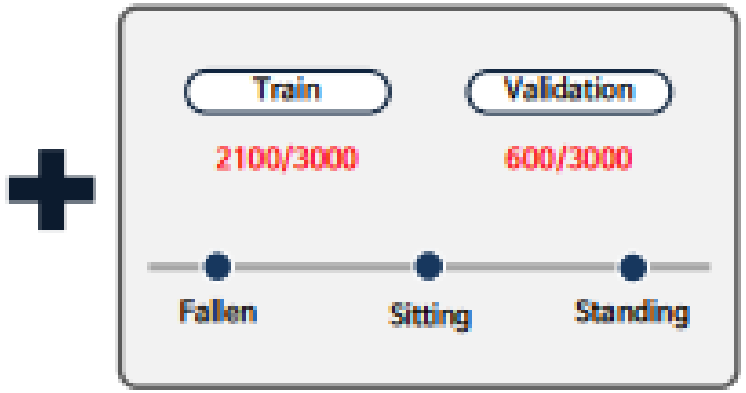
Transfer learning

Pre-trained model dataset — Fine - tuning dataset

YOLOv5

데이터셋 종류	평균 객체 수	특징	사용 여부
COCO	7.3개	일상생활에 가장 근접한 데이터셋	✓
IMAGENET	1.5개	이미지에 인식 가능한 객체 수가 적은 많은 객체가 중앙에 위치	X
PASCAL2	2.4개	카테고리가 20개 정도 밖에 없음	X

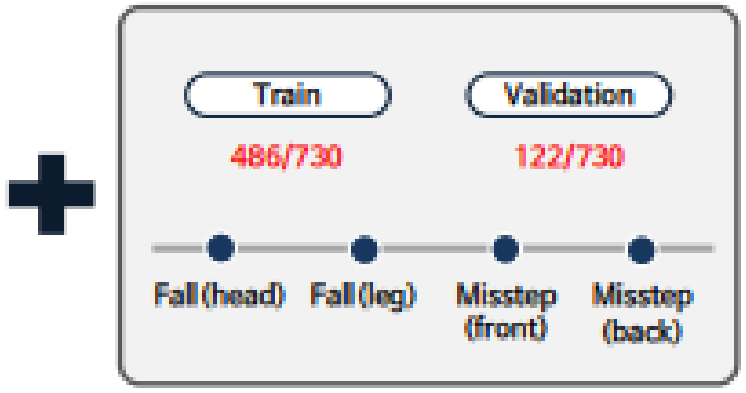
*(단위: 장)



SlowFast

데이터셋 종류	특징	사용 여부
Kinetics-400	일상 생활에 가장 근접한 데이터셋 SlowFast model 학습에 용이	✓
UCF	TV 스포츠 방송 행동 한정	X
hmdb	행동 인식 교육용 목적 데이터셋	X

*(단위: clip)



YOLO v5 Test Dataset *단위: 객체 수

	Fallen	Sitting	Standing
Test Data	300	300	300

SlowFast Test Dataset *1clip=2sec

	Fall (head)	Fall (leg)	Misstep (front)	Misstep (back)
Test Data	122	122	122	122

통합 모델 Test Dataset *1clip=10sec

넘어짐 유형	합계	Fall (head)	Fall (leg)	Misstep (front)	Misstep (back)
테스트 수(장)	100	25	25	25	25

* 10초의 영상에서 3개의 객체 중 한개의 객체만 넘어지는 경우의 데이터 셋

☑ 연구의 방법론 - 통합 모델 알고리즘

과정1 - 쓰러진 객체 Tracking

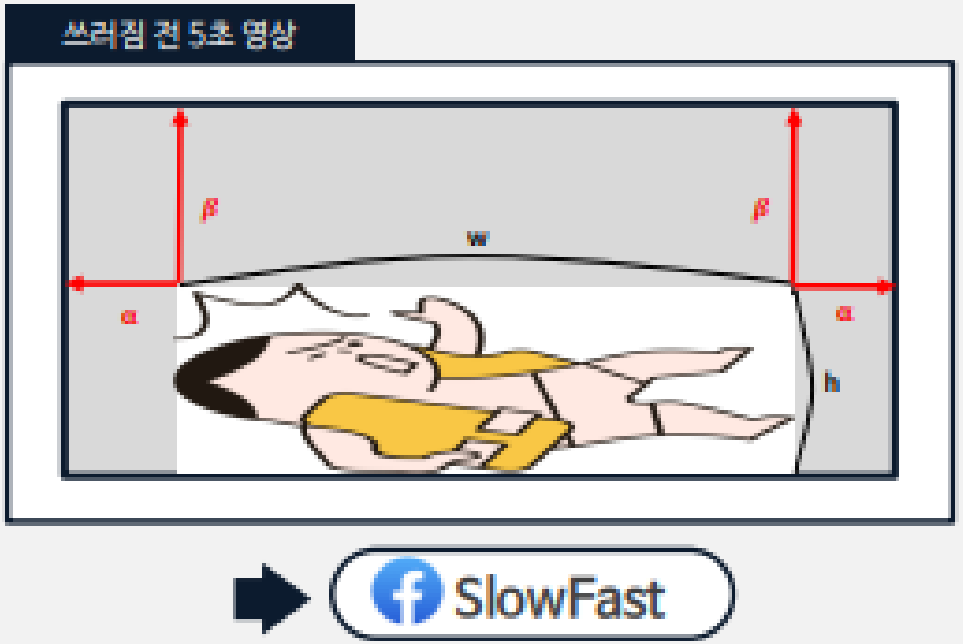
- YoloV5로 쓰러진 객체 탐지 시 Tracking 시작
- MOT tracking algorithm을 통해 동일 객체 인식
- 감지한 객체에게 고유 ID 부여

과정2 - Bounding Box 색상 변경

- 쓰러짐 감지 시 주황색
- 쓰러짐 감지 5초 유지 시 위급상황 인식 후 빨간색

과정3 - Video Crop 후 SlowFast로 영상 전송

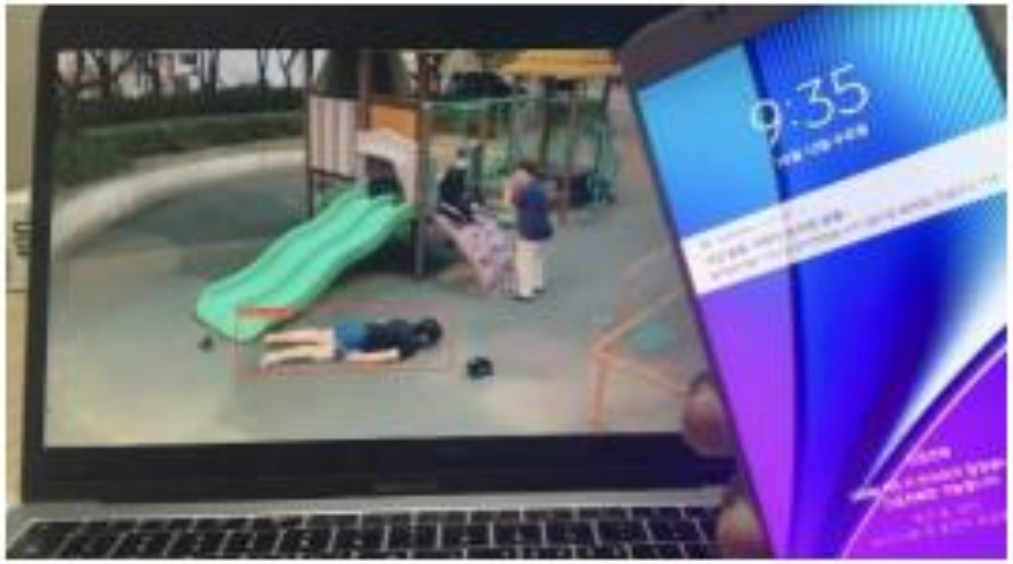
- 위급상황 감지 시 Video Crop
- 유형 분석을 위해 Bounding Box + $\alpha+\beta$ 정보 필요
- α = w*0.2 , β=h
- 원본 영상에서 필요한 부분 Crop하여 SlowFast로 전송



결과 - 평가지표 및 결과

구현 결과

영상



모니터링 영상

Push 알림

최종 결과 정확도

(YOLO v5 평가결과)

	Fallen	Sitting	Standing
AP	85.75	82.62	84.78
Accuracy (%)	87.66		

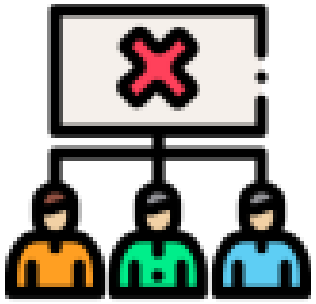
(SlowFast 평가결과)

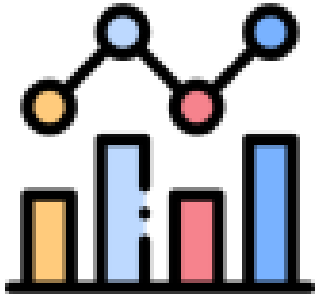
	Misstep (front)	Misstep (back)	Fall (head)	Fall (leg)
F1-Score	0.81	0.83	0.77	0.81
Accuracy (%)	79.73			

(통합 모델 결과) •1clip=10sec

넘어짐 유형	합계	Misstep (front)	Misstep (back)	Fall (head)	Fall (leg)
테스트 수 (clip)	100	25	25	25	25
검출한 클립 수 (clip)	71	19	17	18	17

한계 및 미래연구





한계	모델의 inference을 위한 요구 자원이 큼	실제 어린이 낙상 사고 데이터 수집의 어려움	제한된 촬영 각도로 인한 객체 인식 오류	사고 유형의 다양성 부족
미래 연구	학습된 모델을 pruning 또는 quantization을 통한 경량화	실제 CCTV를 활용한 장기적인 어린이 낙상 사고 영상 데이터 수집	다양한 각도의 영상 데이터 확보	상황별 세분화된 낙상 유형 데이터 수집

개발 환경 (서버 운영)

GPU

Geforce RTX 2080 Ti

Compute Capability

7.5

TensorFlow 2.8.0 CUDA 버전 확인

- 현재 `TensorFlow 2.8.0` 버전을 설치한 상태입니다.
- `TensorFlow 2.8.0` 은 나온지 얼마 안돼서, `2.7.0` 기준에서는 GPU 사용시 cuDNN은 8.1, CUDA는 11.2 버전을 설치해야 한다고 나와있습니다.
  - **CUDA : 11.2**
  - **cuDNN : 8.1**

GPU

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA
tensorflow_gpu-2.7.0	3.7~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.6.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.5.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2

GPU

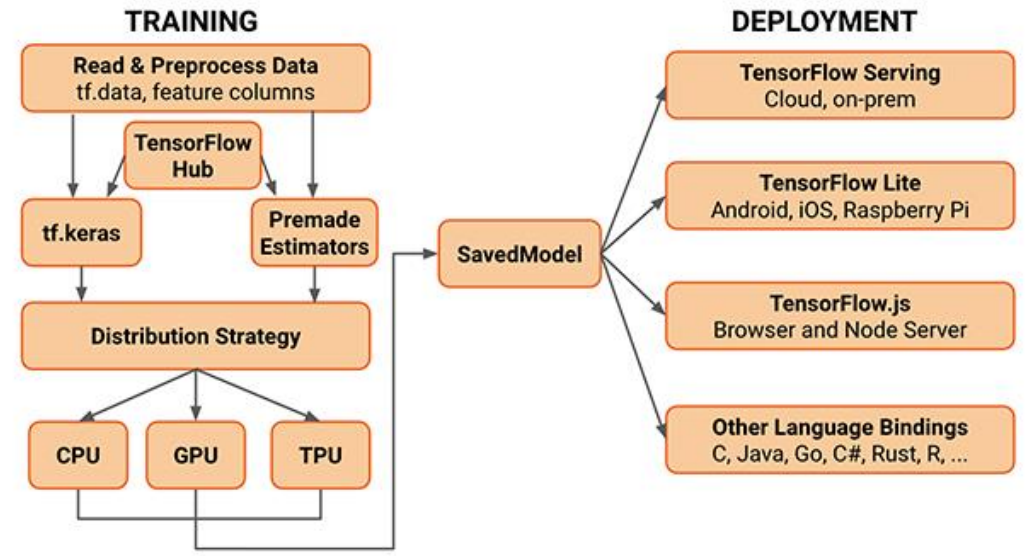
버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA
tensorflow-2.7.0	3.7~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.15.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1	7.4	10.0
tensorflow_gpu-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2	7.4	10.0

CUDA SDK 버전 확인

- https://en.wikipedia.org/wiki/CUDA
- CUDA 11.2 SDK 버전의 capability 확인 결과 `3.5 - 8.6` 까지 사용할 수 있습니다.
- `NVIDIA GeForce GTX 1050 Ti` 그래픽카드의 성능은 확인 결과 `6.1` 이기 때문에 CUDA 11.2 SD
  치 가능합니다.

개발 환경 (서버 운영)

Tensorflow 2.0 ecosystem



ref: https://www.pyimagesearch.com/2019/10/21/keras-vs-tf-keras-whats-the-difference-in-tensorflow-2-0/

1) gpu들 확인

- 내가 어떤gpu를 가지고있으며 해당 gpu번호가 무엇인지 확인한다.

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

2) gpu를 선택하고 변수에 넣어준다.

```
mirrored_strategy = tf.distribute.MirroredStrategy(devices=["/GPU:3","/GPU:4","/GPU:5"
```

3) model에 넣어준다.

- 이때 model 구조 ~ model.compile까지 with mirrored_strategy.scope(): 안에 넣어준다.

```
def transformer_model(X_test, y_test, X_train, y_train, max_len, vocab_size, y_softmax

    with mirrored_strategy.scope():
        inputs = layers.Input(shape=(max_len,))
        embedding_layer = TokenAndPositionEmbedding(max_len, vocab_size, embed_dim)
        x = embedding_layer(inputs)
        transformer_block = TransformerBlock(embed_dim, num_heads, ff_dim)
        x = transformer_block(x)
        x = layers.GlobalAveragePooling1D()(x)
        x = layers.Dropout(0.1)(x)
        x = layers.Dense(128, activation="relu")(x)
    #     x = layers.Dropout(0.1)(x)
        outputs = layers.Dense(y_softmax, activation="softmax")(x)
```

개발 환경 (서버 이전)

https://www.tensorflow.org/install/source#tested_build_configurations
.cuda – gpu 버전 확인 해야함

Quantization – 가능 버전 확인 해보기, 2 점대 이상 가능
Cuda 10.0 부터 가능 하기는 함 (2.4 부터 안정적이기는 해서 2.4 이상 – 즉 cuda 11 이상이면 좋음)

Python – tensrflow 2.x – cudua 10.x, 11.x ,

https://lv99.tistory.com/12
https://velog.io/@hanovator/tensorflow-multi-gpu-%EC%84%A4%EC%A0%95%EB%B0%A9%EB%B2%95
 gpu여러개 활용 (tf – 2.3 버전 이상)

https://ffoorreeuunn.tistory.com/244 코랩 대강 설명 해주기 -> 이걸로 일단은 개발 진행 할거임

차주 게획

CNN 주요 알고리즘 설명 및 실험 진행
경량화 수행 및 성능 비교

https://ffoorreeuunn.tistory.com/244 코랩 대강 설명 해주기 ->
이걸로 일단은 개발 진행 할거임

기존 학습 모델
Imagenet dataset

사용할 데이터
https://public.roboflow.com/classification/rock-paper-scissors

Pre-trained (ImageNet)

이 데이터셋은 1,000개의 클래스로 구성되며 총 백만 개가 넘는 데이터를 포함한다. 약 120만 개는 학습(training)에 쓰고, 5만개는 검증(validation)에 쓴다. 학습 데이터셋 용량은 약 138GB, 검증 데이터셋 용량은 약 6GB이다. 특히 분류(classification) 문제에 관심이 있는 딥러닝 연구자라면 대개 이미지넷 데이터셋을 다운로드하는 편이다. 학습 데이터를 확인해 보면 각 클래스당 약 1,000개가량의 사진으로 구성되어 있다.

모델 이름	Para 개수	기본 정확도 / 속도	경량화 정확도 / 속도	
			Pruning	Post Training Quantizaiton
Vgg 16				
Resnet 50				
Efficientnet_b4				