PORTFOLIO

김덕휘

010 8872 1536

deokhwikim@gmail.com

GitHub

Notion

CONTENTS

About

Project

Warp up

About

김덕휘입니다

저는 경기대학교에서 연기를 전공하고, 약 10년 동안 배우로 활동해왔습니다. 배우로서 무대와 스크린에서 다양한 인물을 연기하며 대중과 소통하는 기회를 가졌고, 이를 통해 창의적인 사고와 공감 능력을 자연스럽게 익힐 수 있었습니다. 다양한 배역을 연구하고 소화하며 스토리를 전달하는 과정은 꽤나 큰 즐거움이었지만, 꾸준히 새로운 도전을 추구하던 제게는 다른 방식으로 세상에 기여할 수 있는 길을 모색하고자 하는 마음이 생겼습니다.

그 과정에서 기술과 데이터를 통해 인간의 삶을 더 풍요롭게 하는 인공지능(AI)이라는 분야에 매료되었습니다. 특히 컴퓨터 비전이라는 분야는 인간의 눈과 같은 역할을 하며 세상을 이해하고 분석하는 데 큰 도움이 되지 않을까? 하는 생각을 하게 되었습니다. 사람의 눈이 단순히 세상을 보는 데 그치지 않고 환경과 상호작용하며 해석하는 것처럼, 컴퓨터 비전을 활용하면 AI도 사람처럼 세상을 보고, 이해하며 더 나아가 무한한 확장 가능성을 제공할 수 있다는 점이 큰 영감을 주었습니다. 저는 이를 통해 AI가 인간의 안전과 편의를 높이고 새로운 창작 기회를 제공하는 데 기여하고자 개발자로서의 길을 선택하게 되었습니다.

About

기획 의도

출산율 감소와 인구 고령화로 인해 노년층의 비율이 점점 증가하고 있다. 그에 따라 노년층의 안전과 건강을 보장하기 위한 서비스 수요의 증가로 이어진다고 판단했다.

특히 낙상과 같은 의도치 않는 사고는 고령자에게 흔하면서도 심각한 사고 중 하나로, 신체적 부상 뿐 아니라 심리적 위축, 의료비 증가 등 다양한 문제를 야기한다.

이에 따라 낙상 사고를 신속히 감지하고 골든 타임을 놓치지 않는 발빠른 대응을 가능하게 함으로써 노년층의 안전과 건강을 지키는 데에 도움이 되고자 하는 것이 이번 프로젝트의 주된 목적이다.

About

개발 환경

CPU Intel i7-14700kf

GPU Nvidia Geforce 4070 SUPER

OS	Windows 11
Cuda	12.5.40
cuDNN	8.9.7

Python 3.10.x

Visual Studio Code
Colab
Jupyter Notebook

Python Libraries
The Ultimate Tool for Developers

Pytorch	2.2.2
Numpy	1.26.4
Mediapipe	0.10.14
Scikit-learn	1.5.2
Ultralytics	8.3.5
OpenCV	4.10.0.84

프로젝트 기간

24.09.30 – 24.10.03	24.10.04 – 24.10.20	24.10.21 – 24.11.23	24.11.24
아이디어 회의 & 주제 선정	데이터 전처리 및 YOLO 학습	데이터 2차 가공, GRU 학습 및 테스트	프로젝트 마무리

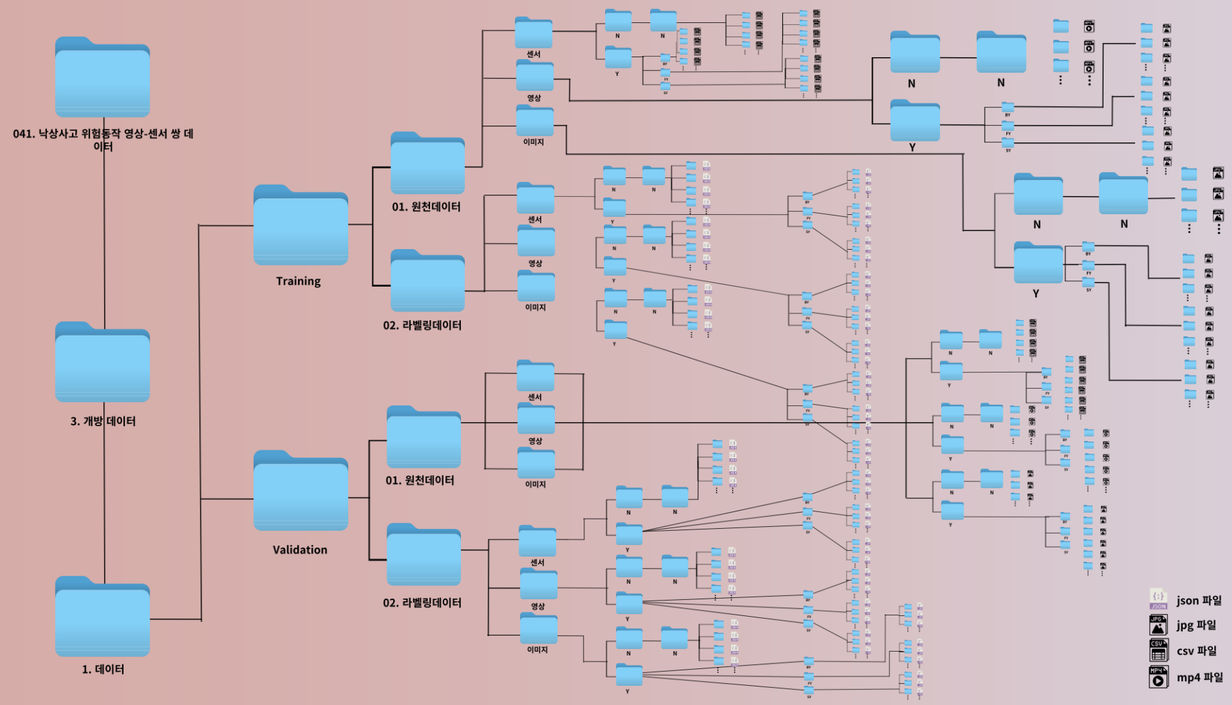
Project

아이디어 회의 및 주제 선정

위 5가지 항목을 토대로 회의를 거쳐 실시간 낙상 감지 프로젝트를 진행하기로 결정했다.

Project

Dataset AIHub



Data Preprocessing

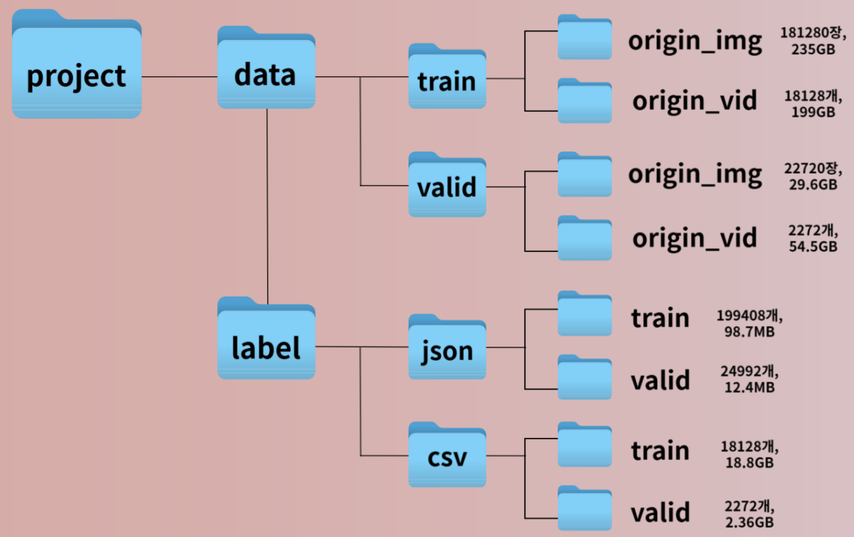
```python
def move_datas(origin_root, target_dir, data_extensions=('.mp4')):
    if not os.path.exists(target_dir):
        os.makedirs(target_dir)

    for dirpath, dirnames, filenames in os.walk(origin_root):
        for filename in filenames:
            if filename.lower().endswith(data_extensions):
                src_path = os.path.join(dirpath, filename)
                target_path = os.path.join(target_dir, filename)

                shutil.copy(src_path, target_path)
                print("완료")

origin_root = 'D:\\041.낙상사고 위험동작 영상-센서 쌍 데이터\\3.개방데이터\\1.데이터\\Training\\01.원천데이터\\TS\\영상\\Y\\'
target_dir = 'D:\\human_fall\\video\\Training\\Y'
val_N_root = 'D:\\041.낙상사고 위험동작 영상-센서 쌍 데이터\\3.개방데이터\\1.데이터\\Validation\\01.원천데이터\\VS\\영상\\N\\W\\'
target_val_N_folder = 'D:\\human_fall\\video\\Validation\\N'
val_Y_root = 'D:\\041.낙상사고 위험동작 영상-센서 쌍 데이터\\3.개방데이터\\1.데이터\\Validation\\01.원천데이터\\VS\\영상\\Y\\'
target_val_Y_folder = 'D:\\human_fall\\video\\Validation\\Y'

move_datas(origin_root, target_dir)
move_datas(val_N_root, target_val_N_folder)
move_datas(val_Y_root, target_val_Y_folder)
```

Python

Total images	204,000
Total videos	20,400
Total jsons	224,400
Total csvs	20,400

데이터셋은 왼쪽과 같은 구조로 되어 있다. 가장 먼저 **YOLO** 모델로 객체 탐지 후 **bbox**의 좌표를 얻어내기로 했기에 데이터 정리부터 시작했다.

Project



1차로 정리한 데이터는 크게 비낙상 데이터(**N**)와 낙상 데이터(**BY, FY, SY**)로 구분 짓고, 3개의 낙상 데이터는 하나(**Y**)로 통합했다.

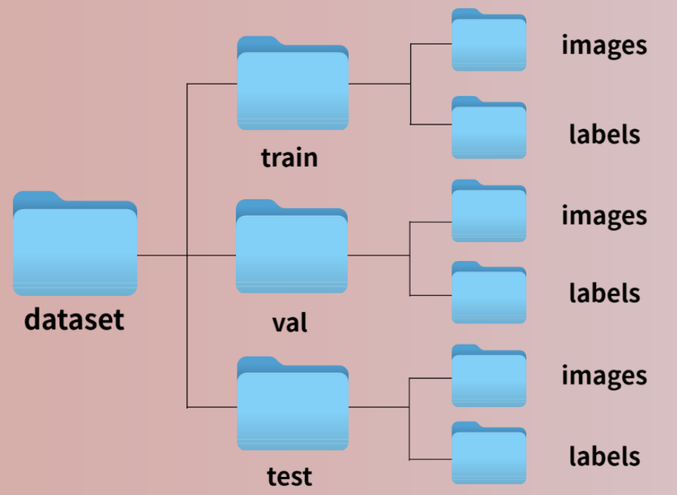
Data Preprocessing

```python
# 소스 폴더와 대상 폴더 경로 설정
source_folder = 'D:\\human_fall\\train_origin\\Y'
train_folder = 'D:\\human_fall\\dataset\\train\\images'
test_folder = 'D:\\human_fall\\dataset\\test\\images'

# 필요한 폴더들이 없으면 생성
for folder in [train_folder, test_folder]:
    if not os.path.exists(folder):
        os.makedirs(folder)

# 파일들을 그룹별로 분류
file_groups = defaultdict(list)
for filename in os.listdir(source_folder):
    if filename.endswith('.jpg'):
        group_key = filename.rsplit('_I', 1)[0]
        file_groups[group_key].append(filename)

# 그룹 목록을 무작위로 섞기
group_keys = list(file_groups.keys())
random.shuffle(group_keys)

# 남은 그룹을 train과 test로 나누기 (각각 1/3씩)
total_groups = len(group_keys)
train_groups = group_keys[:total_groups//3]
test_groups = group_keys[total_groups//3:]

# 파일 이동 함수
def copy_files(groups, destination):
    moved_files = 0
    for group in groups:
        for file in file_groups[group]:
            source_path = os.path.join(source_folder, file)
            dest_path = os.path.join(destination, file)
            shutil.copy(source_path, dest_path)
            moved_files += 1
    return moved_files

# train 폴더로 파일 이동
train_files_copied = copy_files(train_groups, train_folder)
print(f"Moved {train_files_copied} files to train folder")

# test 폴더로 파일 이동
test_files_copied = copy_files(test_groups, test_folder)
print(f"copied {test_files_copied} files to test folder")
```

이후, **YOLO train**을 위한 데이터 재분류 과정.

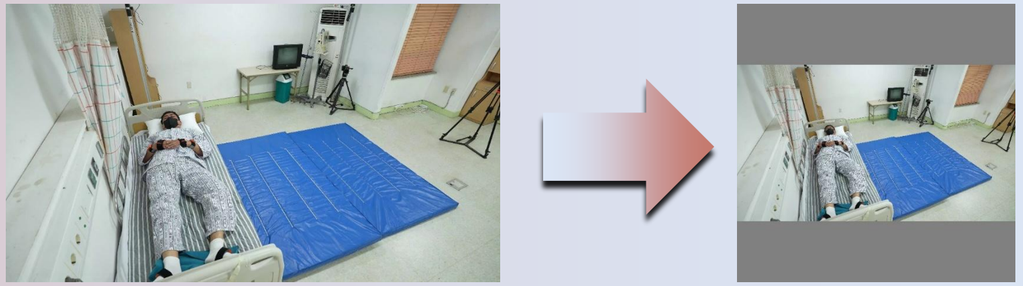
Project

Data Preprocessing



YOLO 모델에 쓸 **img**파일과 **label** 파일만 따로 **copy**해 **YOLO** 형식에 맞춰 2차로 정리했다.
origin_img의 이미지 181,280장만을 가지고 정리했고, 손상된 데이터는 삭제해서
train 87,730 / **val** 42,294 / **test** 42,295 장으로 나눴다.

```python
def letterbox_image(image, target_size=(640, 640)):
    # 원본 이미지 크기
    h, w = image.shape[:2]
    target_w, target_h = target_size

    # 비율 계산
    scale = min(target_w / w, target_h / h)

    # 새로운 이미지 크기
    new_w = int(w * scale)
    new_h = int(h * scale)

    # 이미지 리사이즈
    resized_image = cv2.resize(image, (new_w, new_h), interpolation=cv2.INTER_LINEAR)

    # 패딩 적용
    pad_w = (target_w - new_w) // 2
    pad_h = (target_h - new_h) // 2

    # 이미지를 타겟 크기로 채워서 새로운 이미지를 만듦
    padded_image = cv2.copyMakeBorder(resized_image, pad_h, target_h - new_h - pad_h, pad_w, target_w - new_w - pad_w,
                                      cv2.BORDER_CONSTANT, value=[128, 128, 128])

    return padded_image
```



원본 데이터는 **3840 * 2160**로 16 : 9의 종횡비를 가지고 있다.
training 과정에서도 **resizing**이 가능하지만 데이터의 양이 많아 메모리 누수 발생,
원본 데이터 자체를 **resizing** 및 **padding** 처리해서 640 * 640으로 맞췄다.

Project

Training YOLOv8s

YOLOv8s Training

```
data_root = 'D:\\human_fall\\dataset'
train_root = f'{data_root}\\train\\images'
val_root = f'{data_root}\\val\\images'
class_names = {0 : 'Non_Fall', 1 : 'Fall'}
num_classes = len(class_names)

yaml_info = {
    'path' : data_root,
    'names': class_names,
    'nc': num_classes,
    'train': train_root,
    'val': val_root
}

with open('yaml_info_yolov8s.yaml', 'w') as f :
    yaml.dump(yaml_info, f)
print(f'이 경로에 yaml파일 생성 : {data_root}')
```

```
start_time = time.time()

model = YOLO('yolov8s.pt')
result = model.train(data = 'D:\\project\\prjvenv\\yaml_info_yolov8s.yaml', epochs = 50, batch = 16,
                     imgsz =640, device = device, workers = 20, amp = True, patience = 30, name = 'human_fall_s')

end_time = time.time()
execution_time = end_time - start_time
print(f"실행 시간: {execution_time:.4f} 초") # 약 20시간 소요
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      4.21G     0.2308     0.2552     0.7892          2        640: 100%|██████████| 5484/5484 [18:29<00:00,  4.94it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1322/1322 [04:12<00:00,  5.24it/s]
                   all      42294      42294       0.99      0.996      0.994       0.97

50 epochs completed in 20.185 hours.
Optimizer stripped from runs\detect\human_fall_s30\weights\last.pt, 22.5MB
Optimizer stripped from runs\detect\human_fall_s30\weights\best.pt, 22.5MB

Validating runs\detect\human_fall_s30\weights\best.pt...
Ultralytics YOLOv8.2.101  Python-3.10.6 torch-2.2.1+cu118 CUDA:0 (NVIDIA GeForce RTX 4070 SUPER, 12282MiB)
Model summary (fused): 168 layers, 11,126,358 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1322/1322 [04:48<00:00,  4.59it/s]
                   all      42294      42294       0.99      0.996      0.994       0.97
                  Fall      42294      42294       0.99      0.996      0.994       0.97
Speed: 0.2ms preprocess, 2.8ms inference, 0.0ms loss, 0.4ms postprocess per image
Results saved to runs\detect\human_fall_s30
실행 시간: 74244.0928 초
```

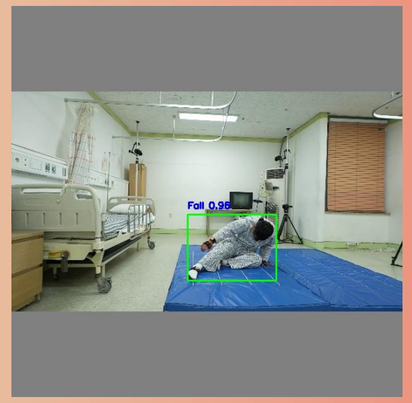
epochs	50
box_loss	0.2308
cls_loss	0.2552
mAP50	0.994
mAP50-95	0.97

87,730장의 이미지를 **training**하는 데 약 20시간이 걸렸다.

Project

Training YOLOv8s

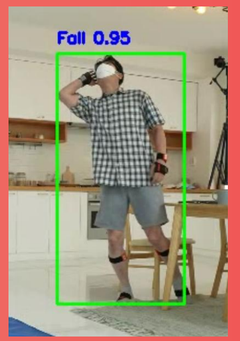
낙상 [참고 영상 링크]



비낙상

비낙상(N)에 분류되어 있는 영상들에는 **Non_Fall** 클래스,
낙상(Y)에 분류되어 있는 영상들에는 **Fall** 클래스를 주고
학습시켰음에도 낙상과 비낙상을 잘 잡아내는 듯이 보인다.

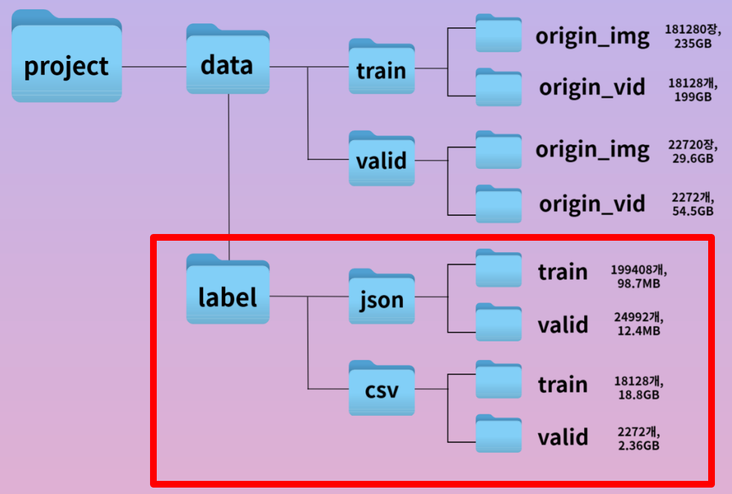
오분류



하지만 데이터에 따라 **낙상하지 않았는데도**
Fall 클래스로 분류하는 케이스가 있었다.

Project

Re-Preprocessing



원본 데이터셋을 1차로 정리할 때, **label** 폴더를 따로 만들어 **json**과 **csv** 폴더로 나눴다. 여기서 **.json** 파일의 정보만 이용해보기로 했다.


```
{
  "metadata": {
    "description": "낙상사고 위험동작 이미지 데이터",
    "scene_id": "00002_H_A_N_C1",
    "scene_format": "MP4",
    "scene_res": "3840 X 2160",
    "creator": "순천향대학교 산학협력단",
    "distributor": "NIA",
    "date": "2023-09-05"
  },
  "scene_info": {
    "scene_loc": "병원",
    "scene_pos": "병실",
    "scene_method": "none",
    "scene_IsFall": "비낙상",
    "scene_cat_name": "비낙상",
    "fall_type": "none",
    "scene_length": 600,
    "cam_num": 1
  },
  "actor_info": {
    "actor_id": "F",
    "actor_age": "adult1(청소년청년)",
    "actor_sex": "m"
  },
  "sensordata": {
    "fall_start_frame": 0,
    "fall_end_frame": 0
  },
  "scene_path": {
    "scene_path": "낙상/N/N/00002_H_A_N_C1"
  }
}
```

```
{
  "metadata": {
    "description": "낙상사고 위험동작 이미지 데이터",
    "scene_id": "00050_H_A_BY_C1",
    "scene_format": "MP4",
    "scene_res": "3840 X 2160",
    "creator": "순천향대학교 산학협력단",
    "distributor": "NIA",
    "date": "2023-09-05"
  },
  "scene_info": {
    "scene_loc": "병원",
    "scene_pos": "병실",
    "scene_method": "none",
    "scene_IsFall": "낙상",
    "scene_cat_name": "후면낙상",
    "fall_type": "중심을 잃고 넘어짐",
    "scene_length": 600,
    "cam_num": 1
  },
  "actor_info": {
    "actor_id": "G",
    "actor_age": "adult2(중장년노년)",
    "actor_sex": "m"
  },
  "sensordata": {
    "fall_start_frame": 226,
    "fall_end_frame": 286
  },
  "scene_path": {
    "scene_path": "낙상/Y/BY/00050_H_A_BY_C1"
  }
}
```

```
{
    "metadata": {
        "description": "낙상사고 위험동작 이미지 데이터",
        "scene_id": "00002_H_A_N_C1",
        "scene_format": "MP4",
        "scene_res": "3840 x 2160",
        "date": "2023-09-05",
        "file_name": "00002_H_A_N_C1_I005.JPG",
        "img_format": "JPG"
    },
    "bboxdata": {
        "bbox_location": "1066.297119140625, 736.6858520507812, 1971.940673828125, 1918.776611328125"
    },
    "img_path": {
        "img_path": "낙상/N/N/00002_H_A_N_C1"
    }
}
```


영상 **sensor.json**과 이미지**.json**


```
Frame,Segment Acceleration_Pelvis x,Segment Acceleration_Pelvis y,Segment Acceleration_Pelvis z,Segment Acceleration_Head x,Segment Acce
1,0.0626168084444963,-0.0488703530990683,-0.0300440231223576,-0.00925222071787103,0.0125547580385156,-0.0986670932238307,-0.020553670777
2,0.0369928416316919,-0.00819398215932675,-0.0834245643796253,-0.0175151152530468,0.00348149448037331,-0.0380759815494928,-0.02702797174
3,0.0416934662670895,-0.00158303081173435,-0.0774586503396608,-0.00603737640427652,0.0217143077612312,-0.0323165515602044,-0.022667196973
4,0.0540053950767839,-0.0327247283260067,-0.0576254181076959,-0.00376652806883239,-0.00174879832978613,-0.0265012702343458,-0.02188794384
5,0.0400684874229377,0.00960063308827792,-0.0775714167712369,-0.0138961432807132,0.0327092236908126,0.0181938499608532,-0.022561731050966
6,0.0562393123016834,-0.00966976391998368,-0.0685225218391766,-0.00412171757746787,0.0348660312496323,0.00343985937584 29,-0.028657496999 8
7,0.0776490437226663,-0.0378332362548723,-0.0558148628881939,0.00181642577302968,0.0206532952522456,0.0219789678151936,-0.012728226884718
8,0.0371920308831796,0.0166313486265755,-0.0444951800509492,0.0162904771293226,0.0794777142874 09,0.0298184332340081,-0.023105630120856 1,
9,0.0473305323957384,-0.0107012415805885,-0.0299884932710722,0.00405963762 09183,0.0229691219720237,-0.0611277129695 05,-0.010212654923662
10,0.0533270253440 98,-0.0190045864122629,-0.0545585305449102,-0.00742715380722141,0.0465315784003704,-0.0209688589720005,-0.038691601790
11,0.0551056909232181,-0.0239898305275023,-0.0593649797906691,-0.00462903738627205,0.0414543108395673,-0.0143585450736573,-0.03473127516
12,0.0525290341830515,-0.0164869855028374,-0.0623976119133975,0.0270851069068519,0.0218132270259084,-0.0596891201007042,-0.0235283911711
13,0.0572783851502471,-0.0106902628769959,-0.0429999330414316,0.0268856475839015,0.0120362647783 76,-0.0580603895294 1,-0.011683697181269 6
14,0.0772225973768,-0.0391002557749867,-0.0425977229660059,0.0285437426948107,0.0115173881839891,-0.0580606549937091,-0.017541780767223
15,0.0874766659395039,-0.0083176129461360 1,-0.0535211980658002,0.0205090119170001,0.0630727167380 09,-0.0933259706600305,-0.0081597310863
16,0.0526870386082183,-0.0039571475185957,-0.0709768776394726,-0.00391493506204292,0.0216396039517314,-0.0645828976034089,-0.01111949769
17,0.0275065844974168,0.0149406972296007,-0.0584886483750402,-0.0147785173712256,0.0491343916217639,-0.0725100472718222,-0.057479486520 8
18,0.0600402743302993,-0.0217521135718489,-0.0645764851886076,0.0219881787095279,0.0401014901921867,-0.0000403445538111 23,-0.032083964 41
19,0.0398291463077347,-0.0308749701410937,-0.0548405175764201,0.0041391433313144,0.0216276847172412,-0.0117470661437332,-0.023302387366
20,0.0535691518212373,-0.0190240651811177,-0.0537290801812192,0.0076076218287250 5,0.0351554801618517,-0.0047829118395036 5,-0.037998095 80
21,0.0595035889046174,0.00057970606343149 2,-0.0340802607906784,-0.0247238361169259,0.0063894889689539,-0.0175520828879066,-0.0466375806 1
22,0.0623116494411 26,-0.0237693346792248,-0.0595997183308 28,-0.0204445864795163,0.0300426515096323,0.0113683941776847,-0.0310281011584 29
23,0.0467689576852699,-0.0263539353075838,-0.0378505938744213,0.0197029942957417,-0.0003958551940 8928,-0.0456363561745311,-0.0320137838
24,0.0487290892337509,0.0249136118648438,-0.0308521560134183,0.0101741801093913,0.0223485518617948,0.0206833125030443,-0.0464285693565053
25,0.0450032328879692,-0.0065378478310241,-0.0884549464426634,-0.0163736401307464,0.0264607986041729,-0.0469303537282477,-0.039103959571
26,0.0739733711775504,-0.0148919088709325,-0.0187055179811942,-0.00985505039219859,0.0695399046589698,-0.0135568127298588,-0.03978272313
27,0.0655547541668939,-0.0150101467444892,-0.0620575479489013,0.0131151948742774,0.0697890971844076,-0.0199731374918809,-0.040461534018 3
28,0.0657251068729432,-0.0322475439839482,-0.0395848141200428,0.0364230585527305,0.0568495416413349,-0.0564155252894802,-0.021315458026 8
29,0.0626046613771152,-0.0198016496553638,-0.0416636182086215,0.0343924008074563,0.0406849609661144,-0.0757362144392 55,-0.018747081404 21
30,0.0470074726486714,0.0222131891296452,-0.0548960969716399,0.0180520515167407,0.0446939995927721,-0.1576827923172 98,-0.003661944378090
31,0.0601516506712207,0.00027887542666192,-0.0289227855181904,0.0420076947886071,0.0113255009854357,-0.1416922886066174,-0.017831617230 0
32,0.0557064008435 04,-0.0177713528008118,-0.0771033596465643,0.0106646736692838,0.0075767127332581 2,-0.0559977944268528,-0.018499825171 7
33,0.0812067551483411,-0.0114369109976888,-0.0549530368171247,0.0076776032210293 6,0.0427145254630 76,-0.0523342494678092,-0.018488246171 4
34,0.0532018243864897,-0.0058424651539604,-0.0283826796804762,-0.0025984704917813 3,0.0233725194161361,-0.0154141563180386,-0.033659137 75
35,0.0344438101649655,-0.0445194571603658,-0.0789517904717887,-0.0150732124920708,0.0348938627366457,0.0107300185073195,-0.053420018731 6
36,0.0614135477254629,0.00334797060328538,-0.0475611688971563,-0.0352564450373711,-0.0011395099846851 6,0.0407888912270 9631,-0.04112046953
37,0.0423033022108629,-0.0165657823875026,-0.0488944188456189,0.0058021818747748 1,0.0336815836335142,0.0832237541947324,-0.0530306263678
```


영상 **sensor.csv**

Project

Re-Preprocessing

비낙상 sensor data.json

```
  },
  "scene_info": {
    "scene_loc": "병원",
    "scene_pos": "병실",
    "scene_method": "none",
    "scene_IsFall": "비낙상",
    "scene_cat_name": "비낙상",
    "fall_type": "none",
    "scene_length": 600,
    "cam_num": 1
  },
  "actor_info": {
    "actor_id": "F",
    "actor_age": "adult1(청소년청년)",
    "actor_sex": "m"
  },
  "sensordata": {
    "fall_start_frame": 0,
    "fall_end_frame": 0
  },
  "scene_path": {
    "scene_path": "낙상/N/N/00002_H_A_N_C1"
  }
}
```

낙상 sensor data.json

```
  },
  "scene_info": {
    "scene_loc": "병원",
    "scene_pos": "병실",
    "scene_method": "none",
    "scene_IsFall": "낙상",
    "scene_cat_name": "전면낙상",
    "fall_type": "미끄러져 넘어짐",
    "scene_length": 600,
    "cam_num": 2
  },
  "actor_info": {
    "actor_id": "F",
    "actor_age": "adult1(청소년청년)",
    "actor_sex": "m"
  },
  "sensordata": {
    "fall_start_frame": 314,
    "fall_end_frame": 374
  },
  "scene_path": {
    "scene_path": "낙상/Y/FY/00007_H_A_FY_C2"
  }
}
```

```
{
    "metadata": {
        "description": "낙상사고 위험동작 이미지 데이터",
        "scene_id": "00007_H_A_FY_C1",
        "scene_format": "MP4",
        "scene_res": "3840 x 2160",
        "date": "2023-09-05",
        "file_name": "00007_H_A_FY_C1_I001.JPG",
        "img_format": "JPG"
    },
    "bboxdata": {
        "bbox_location": "1393.427978515625, 833.1392211914062, 2280.573486328125, 2143.593994140625"
    },
    "img_path": {
        "img_path": "낙상/Y/FY/00007_H_A_FY_C1"
    }
}
```

이미지.json

이미지.json 파일에는 bbox의 좌표가 들어가 있었지만 sensor data.json에는 bbox의 좌표 대신 fall_start_frame과 fall_end_frame이 적혀 있다.

Project

Re-Preprocessing

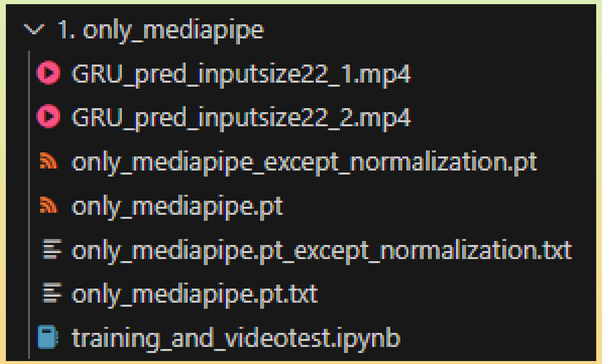
6프레임 간격으로 각 프레임마다 **bbox**의 좌표를 추출하고, **Mediapipe**를 사용해 신체의 각 랜드마크 중 11개(코, 양쪽 어깨, 양쪽 손목, 양쪽 골반, 양쪽 무릎, 양쪽 발목)만 추출

동시에 **sensor data.json**에 명시되어 있던 **fall_start_frame**과 **fall_end_frame**을 기준으로 **Normal, Danger, Fall** 클래스로 재정의했다.

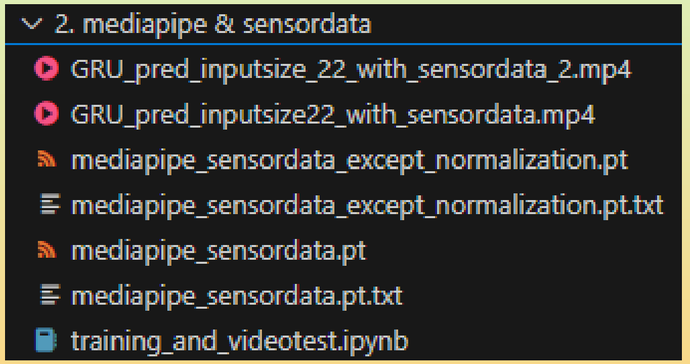
Project

Training_GRU

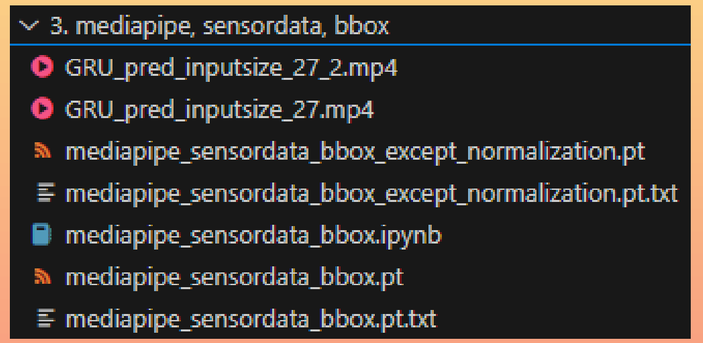
비교를 위해 가장 기본적인 **GRU** 모델만을 사용, 정규화를 시켰을 때와 시키지 않았을 때의 차이도 보고 싶어 총 8개의 가중치 파일(**.pt**)을 만들었고, 가중치 파일에 대응하는 **f1_score, confusion matrix**를 **.txt**로 저장했다.



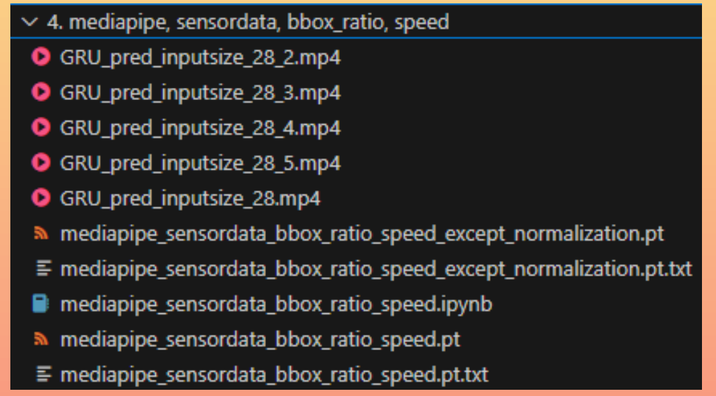
Mediapipe의 각 랜드마크만 학습



Mediapipe의 랜드마크와 **danger** 클래스 추가



Mediapipe의 랜드마크, **danger** 클래스, **bbox**의 좌표 추가



Mediapipe의 랜드마크, **danger** 클래스, **bbox**의 좌표, **bbox**의 비율, 속도 추가

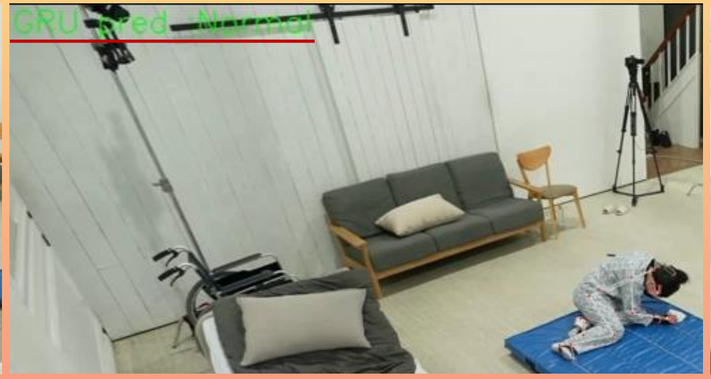
Project

Training_GRU

GRU를 통한 낙상 감지

정규화 하지 않은 모델

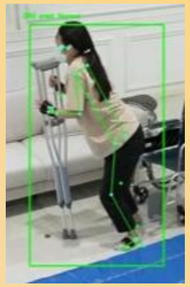
정규화 모델

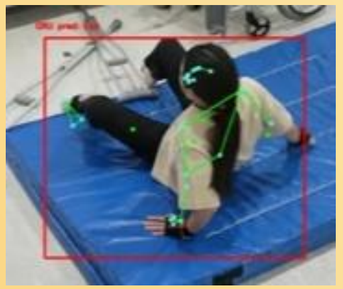


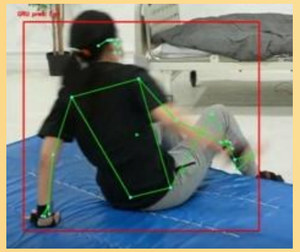
Mediapipe로 추출한 11개의 랜드마크만 학습한 모델.
정규화 모델은 하나의 클래스만 예측하고, 정규화 하지 않은 모델은 정확하진 않지만
2개의 클래스를 예측한다.

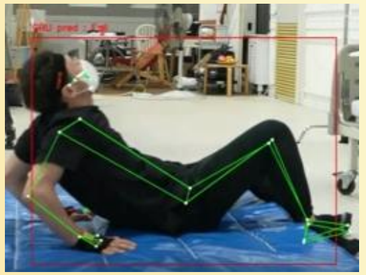
Project

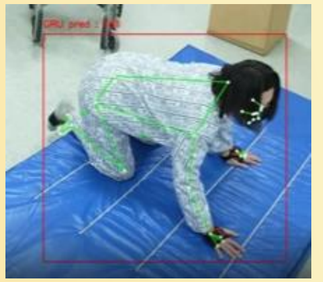
모델 성능 Test 및 비교











Input_size : 22, 정규화 하지 않고
11개의 랜드마크와 **danger** 클래스를 학습한 모델.
정확하진 않지만 일단 낙상 자체는 잘 잡아내는 듯이 보인다.

Input_size : 27, 정규화 하지 않고
11개의 랜드마크, **danger** 클래스, **bbox**의 비율, **bbox**의 좌표를 학습한 모델.
기존 모델보다 조금 더 예민해진 것 같다.

Input_size : 28, 정규화 하지 않고
11개의 랜드마크, **danger** 클래스, **bbox**의 비율, **bbox**의 좌표, **speed**를 학습한 모델.
낙상은 잘 잡아내지만 **Normal 클래스를 잡아내지 못한다.**

Project

모델 성능 Test 및 비교

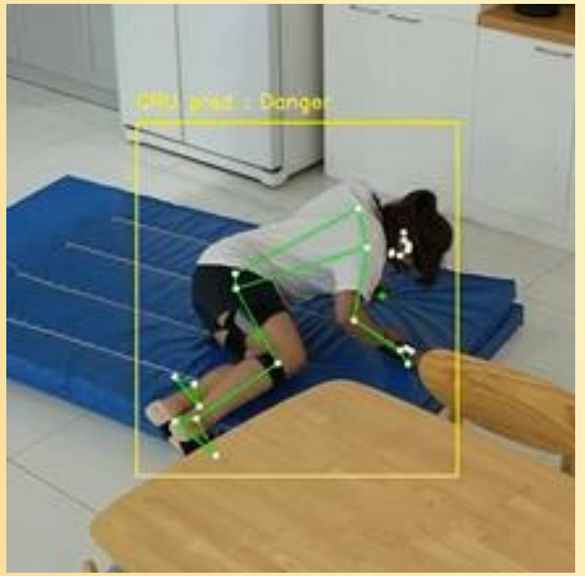
GRU 모델의 성능이 예상보다 좋지 않았다.

그렇다면 **GRU** 모델을 훈련시킬 때 사용했던 방법을 **Mediapipe**에 적용시켜보면 어떨까 하는 생각이 들었다.

1. **랜드마크 추출**
2. **랜드마크 기반 bbox**
3. **이전 프레임과 현재 프레임의 좌표 변화량(속도)**

위 3 항목만 있으면 **mediapipe**만으로도 낙상 감지를 할 수 있을 것 같았다.

속도의 **threshold**값을 정하고 그 임계값을 기준으로 1차 클래스 분류, **bbox**의 비율로 최종 클래스를 결정해보면 어떨까?



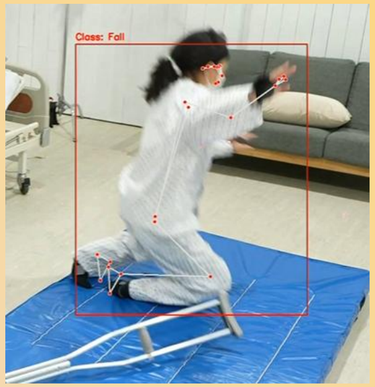
Project

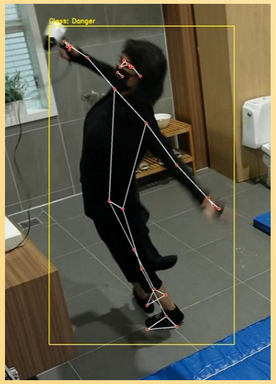
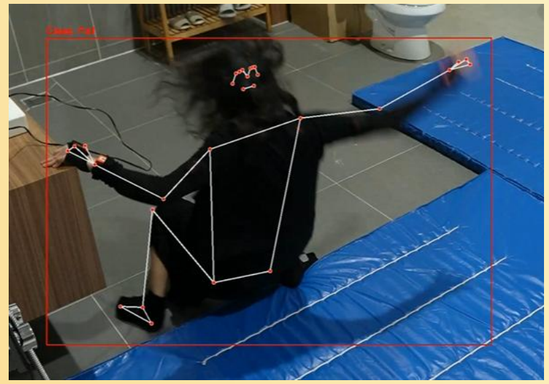
모델 성능 Test 및 비교

[Mediapipe를 이용한 낙상 감지]

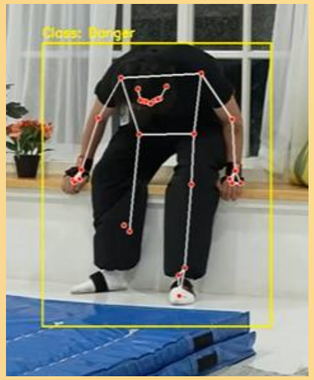
낙상





비낙상


bbox ratio 〈 0.7 :	Normal
0.7 〈= bbox ratio 〈 0.8 :	Danger
bbox ratio 〉= 0.8 :	Fall

Threshold normal = 10.5	Normal
Threshold danger = 15.5	Danger

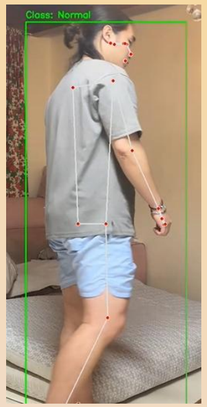
Input_size 27의 GRU 모델과 비슷하거나 더 잘 감지하는 것 처럼 보인다.

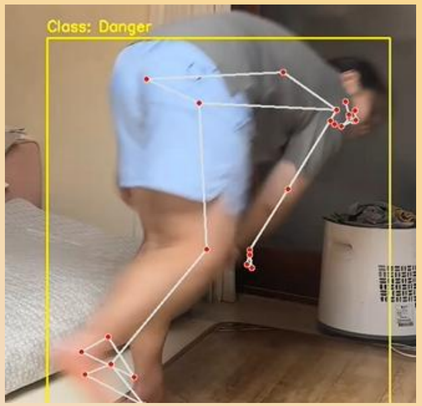
Project

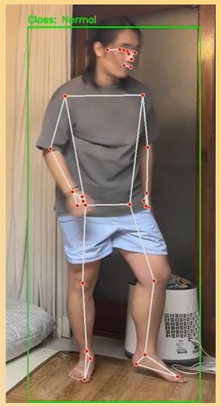
모델 성능 Test 및 비교

[웹캠을 이용한 실시간 낙상 감지]

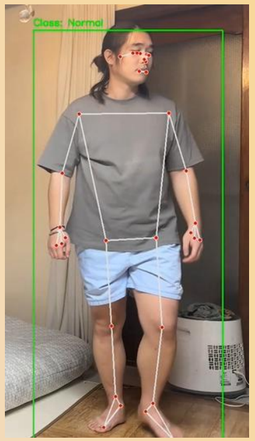
1. 비낙상

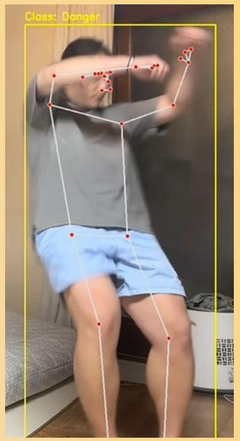


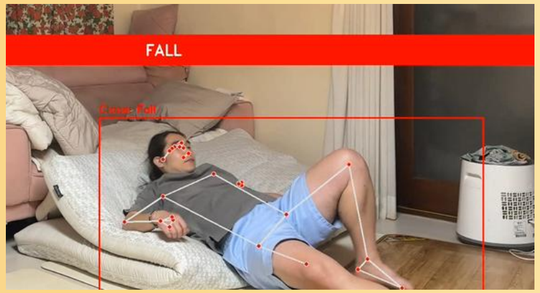




2. 낙상

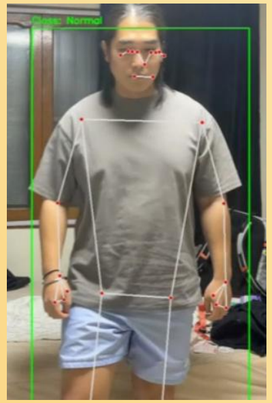


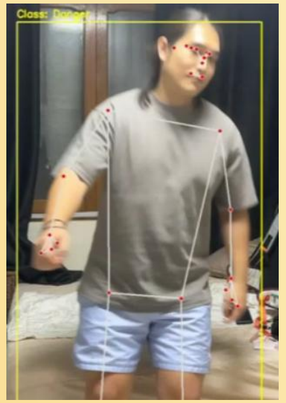




*Fall 클래스가 15프레임 이상 지속되면 완전한 낙상이라고 간주하고 경고문을 띄우는 형식

3. 비낙상





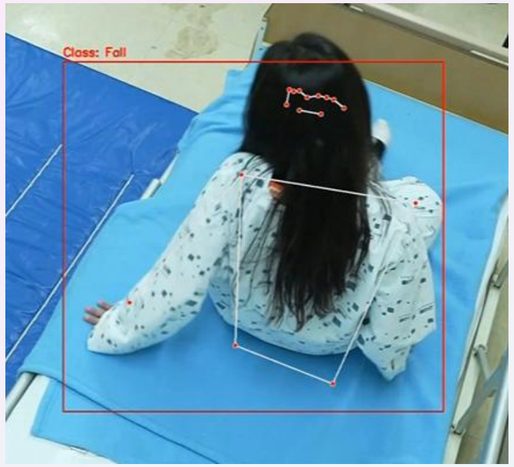
*속도가 임계값 이하이면 bbox의 비율이 초과되어도 Normal 클래스

Wrap up

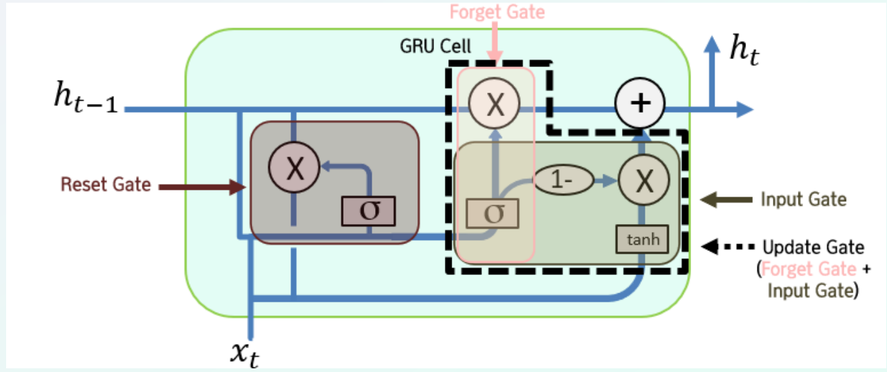
아쉬운 점 및 개선할 점

1. 우리 프로젝트에서 가장 근본적인 단점은 mediapipe가 객체 인식을 하지 못하면 GRU 모델로 낙상 감지를 하던, mediapipe로만 낙상 감지를 하던간에 제대로 동작하기가 쉽지 않다. mediapipe는 휠체어 뒤에 있는 사람의 상태를 예측하지 못하고 이불을 덮고 침대에 누워있는 환자의 상태를 알지 못한다.

3. 완전 낙상을 감지했을 때 화면에 표시만 하는 것이 아닌, 보호자에게 알림 메시지가 간다던가 연동된 앱에 신호를 준다던가 하는 시도도 해보고 싶었는데 여건상 하지 못했다. 개인적으론 앱을 해보고 싶은 마음도 있어서 공부하면서 만들어볼 생각이다.



2. 카메라의 각도와 사람 사이의 거리도 중요한 변수 중 하나라고 생각된다. 카메라와의 거리에 따라 모델이 인식하는 사람의 속도도 달라질 것이고, 그 거리에 맞춰 속도 임계값을 재정의 해줘야 하는 번거로움이 생길 여지가 있다. 또한 카메라의 각도에 따라서 bbox의 비율도 천차만별로 달라질 테니 이러한 부분들에 대해 고민이 더 필요할 듯 싶다.



4. GRU 모델의 성능도 개선해보고 싶다. 입력 feature만 다양하게 시도해봤지 하이퍼 파라미터나 레이어 부분도 비교해보면서 우리가 가진 데이터셋을 어떻게 학습시켜야 성능이 개선되는지 눈으로 보고 싶다.

Thank You