

Prevent Accidents  
At Construction Sites  
Using Deep Learning



# 딥러닝을 활용한 건설 현장의 사고 예방

최종발표

2017313670 박세연

2017310683 김지수





Prevent Accidents  
At Construction Sites  
Using Deep Learning



# 목차

1. 연구 배경 및 목적
2. 연구 설계 방향
3. 연구 이론적 배경
4. 연구 내용 및 과정
5. 결론 및 한계점
6. 참고 문헌



# **1. 연구 배경 및 목적**



# 연구 배경 및 목적

## 건설산업의 재해 현황 분석

### 후진 이동 중이던 굴삭기와 충돌·협착 사고사례

▲ 헌정안전(주) | ◎ 입력 2020.08.16 16:17 | ◎ 수정 2020.08.21 23:17 | ◎ 댓글 0 |  
◎ 조회수 1508



2020.3.7일 16시 30분경 경기도 평택시 현덕면 소재 토목조경공사 현장에서  
재해자가 굴삭기 바퀴에 깔려 사망한 사고가 발생했다.

이와 유사한 사례들이 자주 발생하고 있어 건설현장에서 주

[사고사례/2019년](#)

### 무안 아파트 공사현장 굴삭기 사고로 작업자 1명 사망

사용자 ulsansafety 2019. 7. 29. 16:32

무안 아파트 공사현장 굴삭기 사고로 작업자 1명 사망

2019년 7월 29일 오전 7시 50분

전남 무안 아파트 신축공사 현장에서 굴착기 운전자가 후진하다 뒤에 있던 작업자가 사고를 당했다. 병원으로 후송하였지만 끝내 사망하였다.

### 도로에서 유도업무 중 후진하는 덤프트럭에 깔려 사망

▲ 김보현 | ◎ 입력 2019.06.07 09:52 | ◎ 수정 2019.06.07 11:00

### 광양 원료부두에서 석탄하역작업 중 불도저에 깔려 2명 사상 (2021.05.03)

사용자 ulsansafety 2021. 5. 4. 17:58

광양 원료부두에서 석탄하역작업 중 불도저에 깔려 2명 사상 (2021.05.03)

2021년 5월 3일 오전 1시 30분경

광양 운수창고 원료부두에서 현장 내 집단작업을 하던 중 불도저에 근로자 2명이 깔려 1명이 사망하고 1명이 중상을 입었다.



## 연구 배경 및 목적

건설 현장 안전관리 현황

### 건설안전

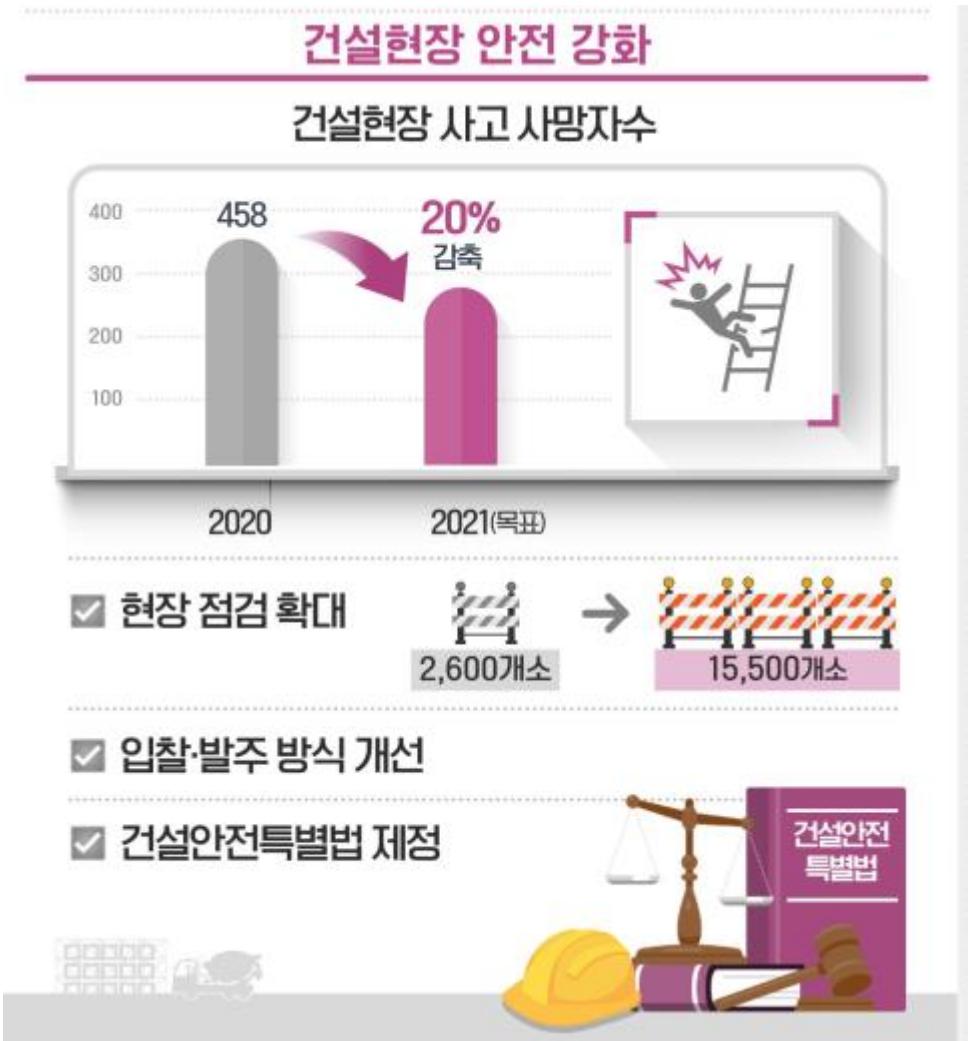
현장점검 확대, 건설기계 관리 강화 등  
**현장 중심의 안전관리 강화로**  
**사망자 수 20% 이상 감축합니다.**

\* 건설사고 사망자 수 : ('20년) 458명(잠정) → ('21년 목표) 350명

**인력 확충 현장 점검 대폭 확대 소규모 민간현장 전수점검** [23년~]  
(2,600→15,500개소)

**건설기계의 검사제도 강화** 검사주기 단축  
검사기관의 인력·장비 확충

국토교통부



**현장 점검 확대** 및 **건설기계 관리 강화** 등 현장 중심의 안전관리 강화로  
건설업의 사망자수를 20% 이상 감축하겠다는 계획



## 연구 배경 및 목적

### 건설산업의 재해 현황 분석

[안전은 권리] 산재 사망사고 절반은 건설현장...  
발판부터 튼튼히 다져라

2017년 사고 사망자 964명...건설업서 절반이상 506명 사망

등록 2019-04-05 오전 6:00:00  
수정 2019-04-05 오전 6:00:00

작년 산재사망자 971명... 절반이 건설현장서  
참변

2018년 산업재해 현황 발표

건설업 사고사망만인율 1.65‰... 평균 3배 이상 높아  
사고사망자 964→971명 증가... 건설업 485명으로 절반

건설업은  
**사망자가 가장 많이 발생**한 업종!

[2019년 산업재해 통계]



사망자는 건설업(517명, 25.6%)에서 가장 많이 발생하였지만,  
전년동기 대비 증감률은 -9.3%로 하락했습니다.

## 건설업에서 전체 사고성 사망재해의 51% 발생

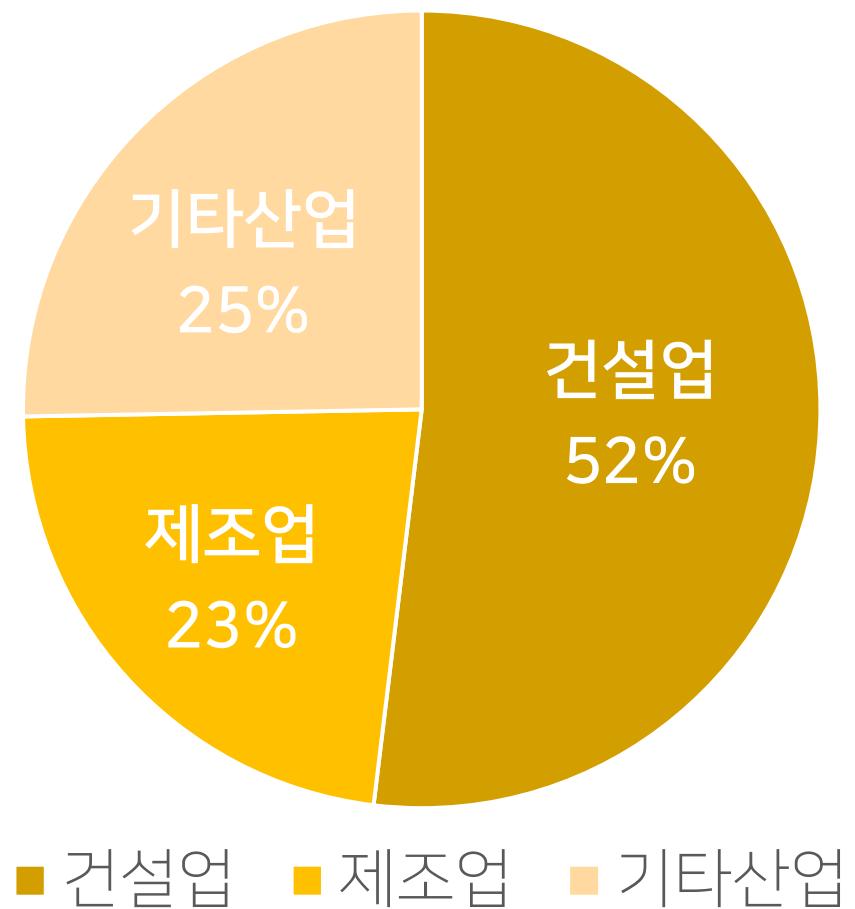
▣ 김성민 기자 | ◎ 입력 2020.10.23 16:22 | ◎ 수정 2020.10.30 09:51



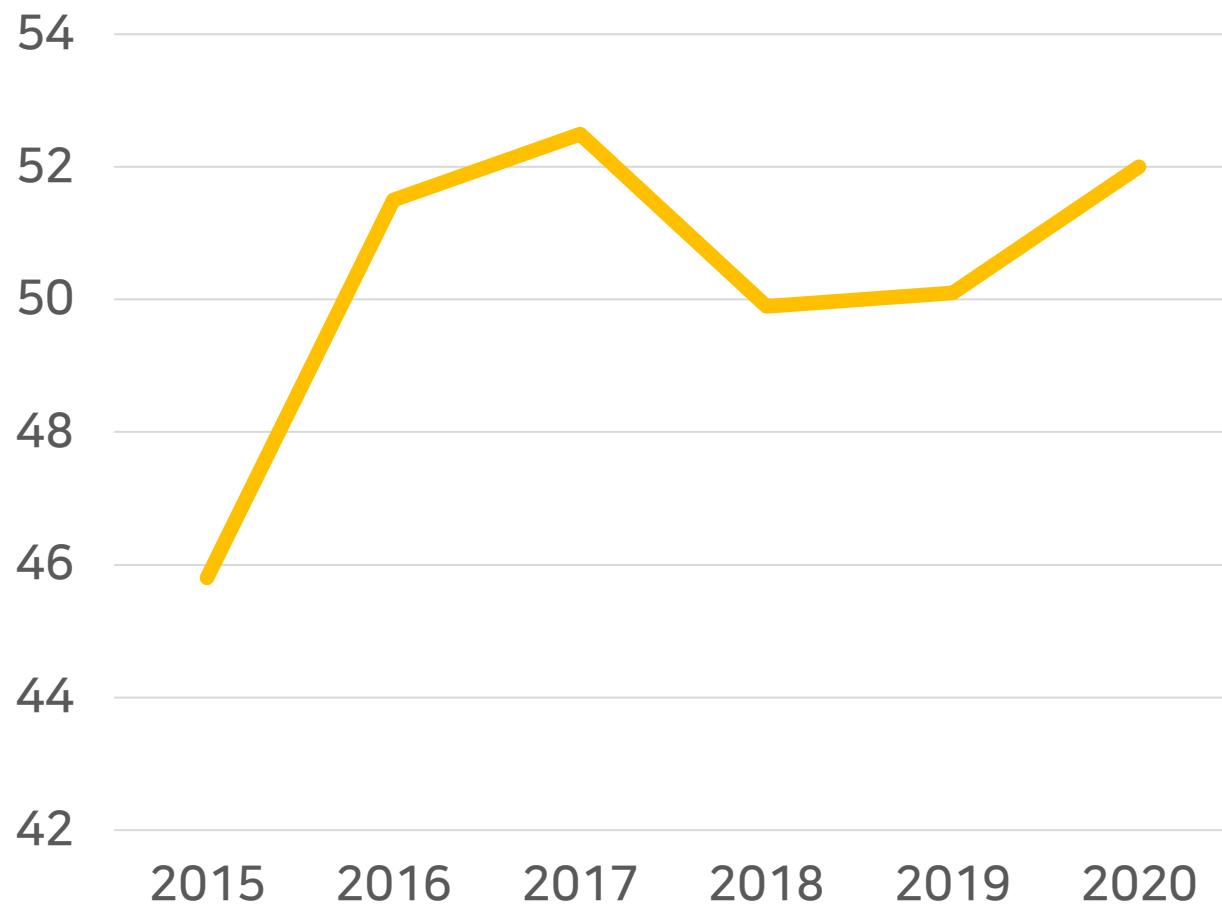
## 연구 배경 및 목적

### 건설산업 재해 현황

### 2020 업종별 산업재해 사고사망



### 산업재해 사망자 중 건설업 비율





## 연구 배경 및 목적

건설산업의 재해 현황 분석

“

연간 산업재해사망자 중 **건설업은 매년 절반 이상을 차지할 정도로 사고사망율이 높다.**

국가에서도 건설 현장의 **안전 관리에 대해 심각성을 인지하고 제도 마련에 관심을 갖고 있다.**

하지만 여전히 사고 발생률이 높은 것으로 보아 **새로운 안전관리 방식의 도입이 필요하다.**

”



## 연구 배경 및 목적

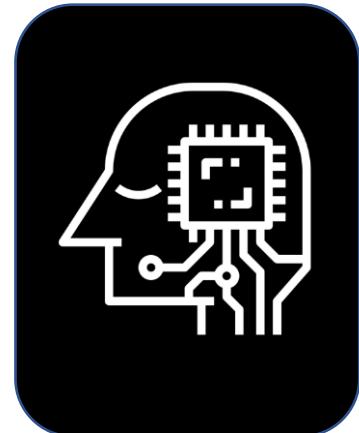
연구 목적

### 문제 인식

“사람의 노력으로 사고를 예방하는 데에는 한계가 있다” 

### 해결 방법

딥 러닝을 활용한 건설 현장 안전 관리 자동화



건설 장비 운전자와 주변 작업자 간의 작업 상황 정보를 공유 및 인지하여 작업자의 안정성 향상

### 목표



안전 관리 자동화를 통해 제도적인 예방에서 나아가 안전이 보장되는 건설 현장을 만들고자 한다.

## **2. 연구 설계 방향**



## 연구 설계 방향

건설 현장 가정



### 재해발생원인

#### ○ 차량계 건설기계 접촉방지 조치 미실시

- 차량계 건설기계를 사용하여 작업을 하는 경우 굴삭기에 접촉되어 부딪칠 위험이 있는 장소에는 근로자 출입을 금지하거나 유도자를 배치하여 굴삭기를 안전하게 유도하여야 하나 미실시

#### ○ 차량계 건설기계 작업계획서 미 작성

- 차량계 건설기계를 사용하여 작업을 하는 경우 해당 굴삭기의 종류 및 성능, 운행경로, 작업방법 등의 내용을 포함한 작업계획서를 작성하고 이에 따라 작업을 진행하여야 하나 미작성

### **3. 연구 이론적 배경**



## 연구 이론적 배경

YOLACT

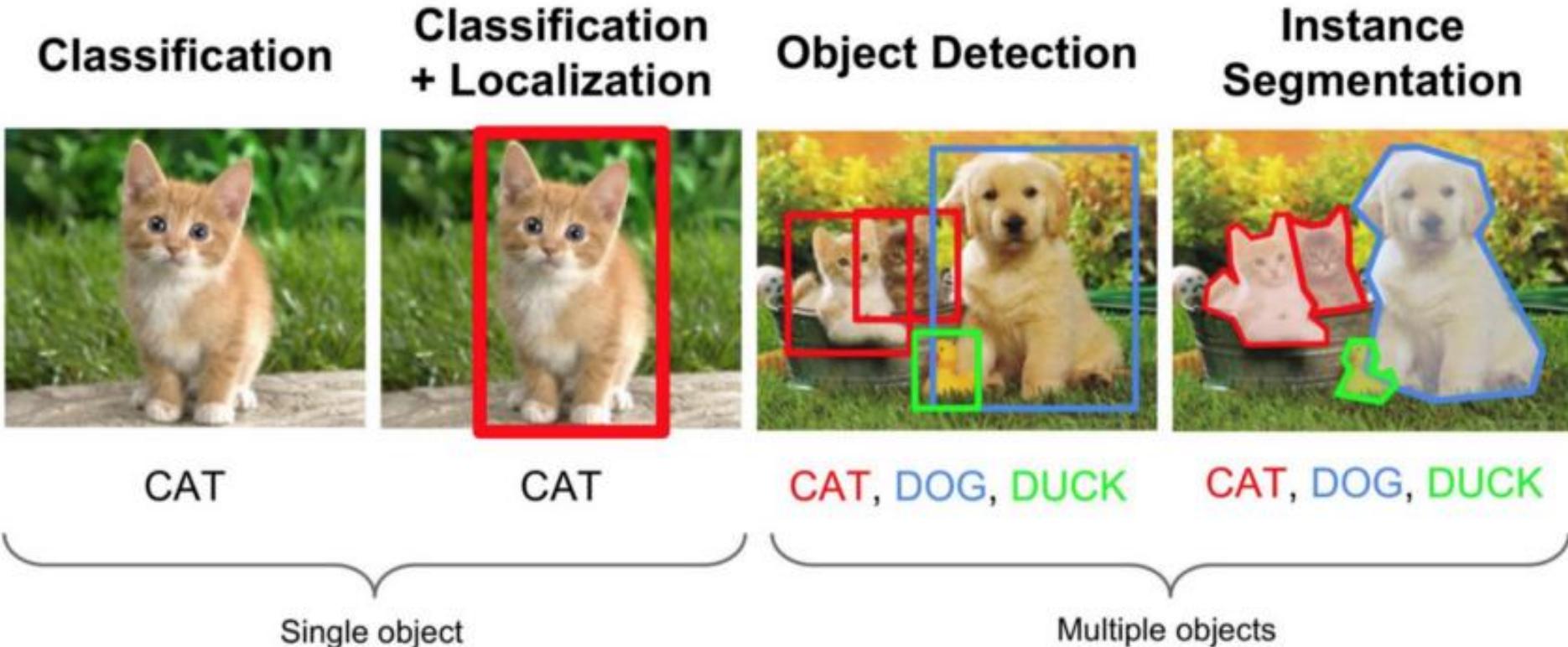
# You Only Look At Coefficients

Instance Segmentation 를 수행하기 위해 단순한 Convolution Model 을 제시한다

Object Detection (객체 탐지)

Instance Segmentation (객체 분할)

Bounding Box로 같은 클래스를 탐지  
객체의 모양 그대로 객체 영역을 탐지



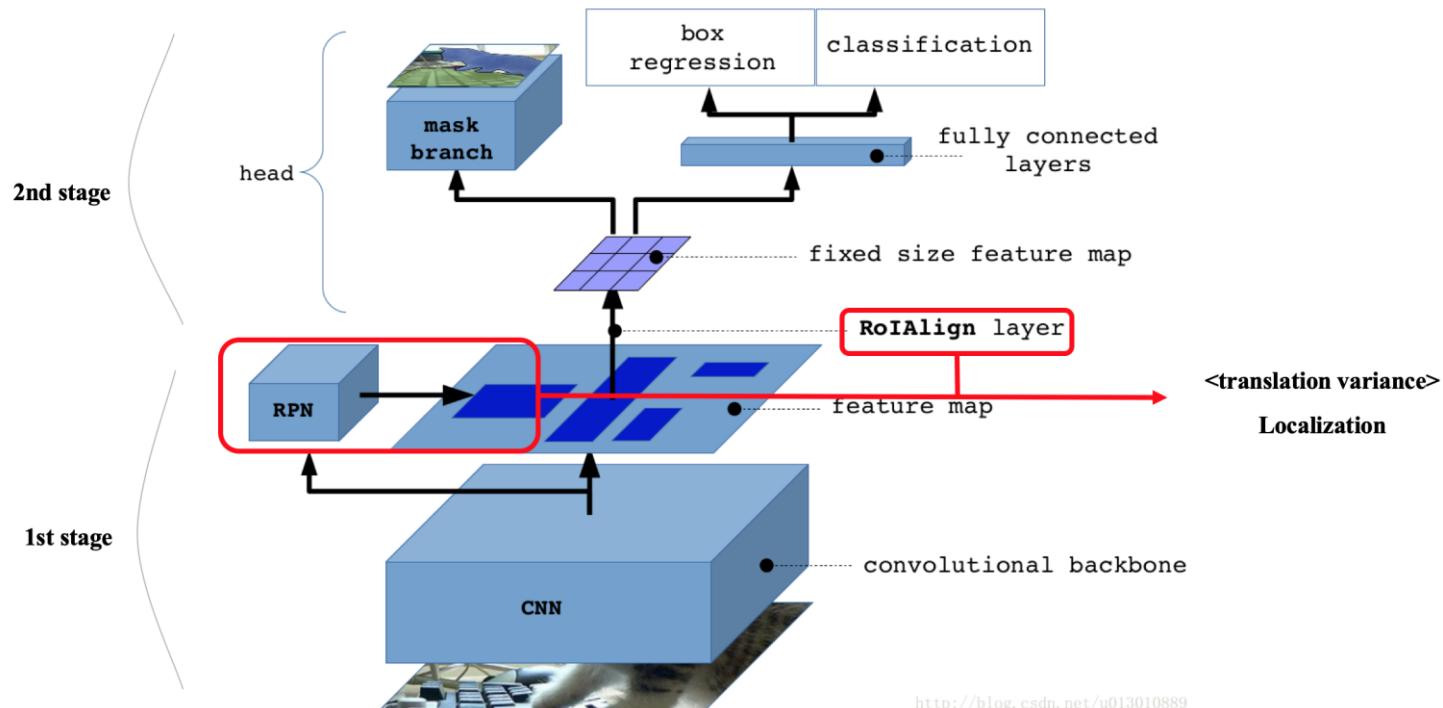


## 연구 이론적 배경 YOLOCT 이전의 기존 방식

## 기존 방식 2-stage detector

1. localization 먼저 수행
  2. classification 및 segmentation 수행

두 단계에 걸쳐서 진행하기에 속도보다는 성능에 중점



## 예) Mask RCNN

1. localization : RPN에서 RoI 영역 추출
  2. segmentation: 각 bounding Box에 대한 segmentation 수행



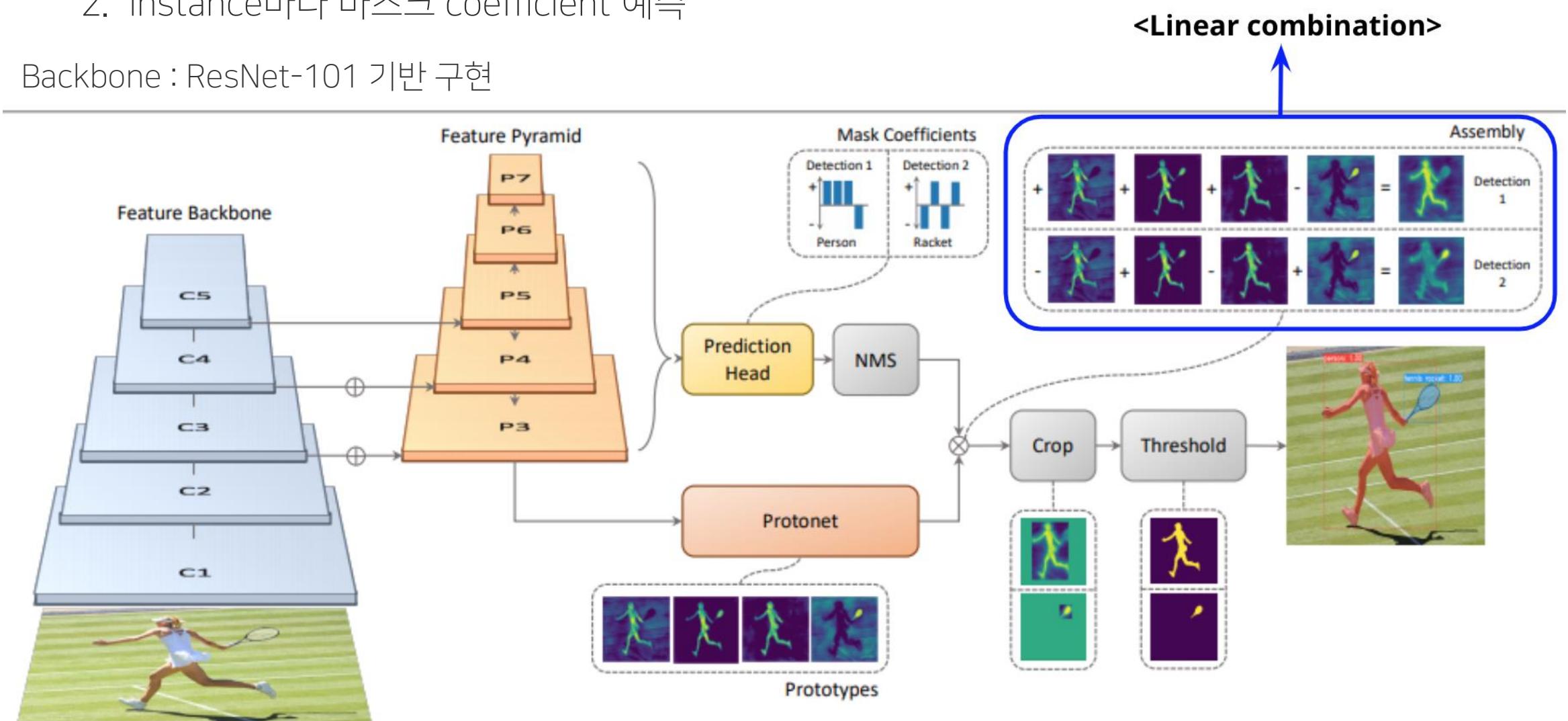
# 연구 이론적 배경

YOLOACT Architecture

## 1, 2 단계를 병렬적으로 실행함으로써 속도 개선

1. Non-local Prototype Mask 생성
2. instance마다 마스크 coefficient 예측

Backbone : ResNet-101 기반 구현



feature map 추출  
C1 ~ C5

feature map 추출  
P3 ~ P7

2 substack 병렬 실행  
- Prediction Head  
- Protonet

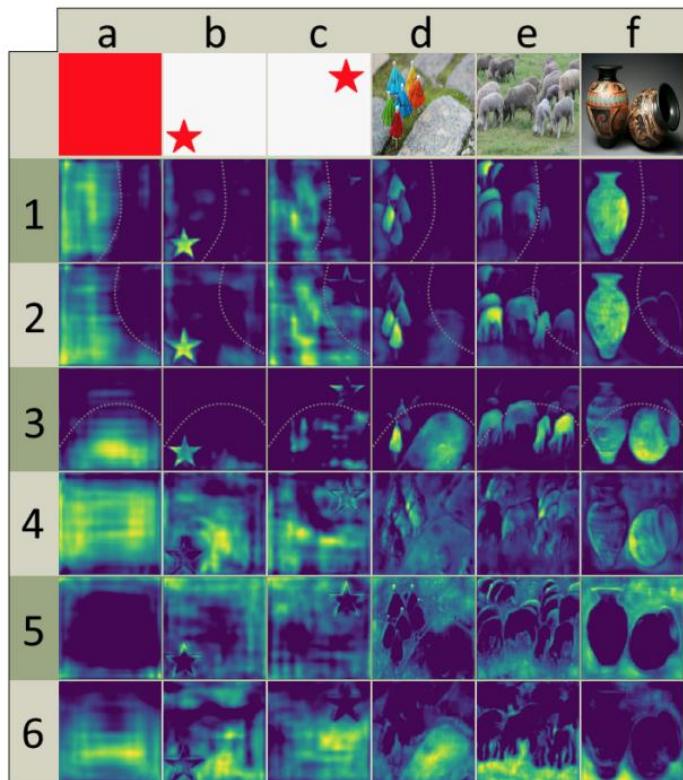
연산을 통해  
병렬 substack 합쳐져서  
final mask 생성



# 연구 이론적 배경

YOLACT Protonet

## Prototype Mask



생성된 mask는  
이미지 개수( $k$ )에 따라 채널  $k$ 개를 생성하는데  
채널마다 이미지의 다른 지역을 activate

- 1~3 : 점선 곡선 안쪽 object activate
- 4 : 이미지 좌하단 activate
- 5 : 배경과 object 간의 edge activate
- 6 : ground activate

## padding을 통한 activate

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0



1	0	0
1	0	0
1	0	0

$$= \begin{array}{|c|c|c|} \hline 0 & 2 & 2 \\ \hline 0 & 3 & 3 \\ \hline 0 & 2 & 2 \\ \hline \end{array}$$

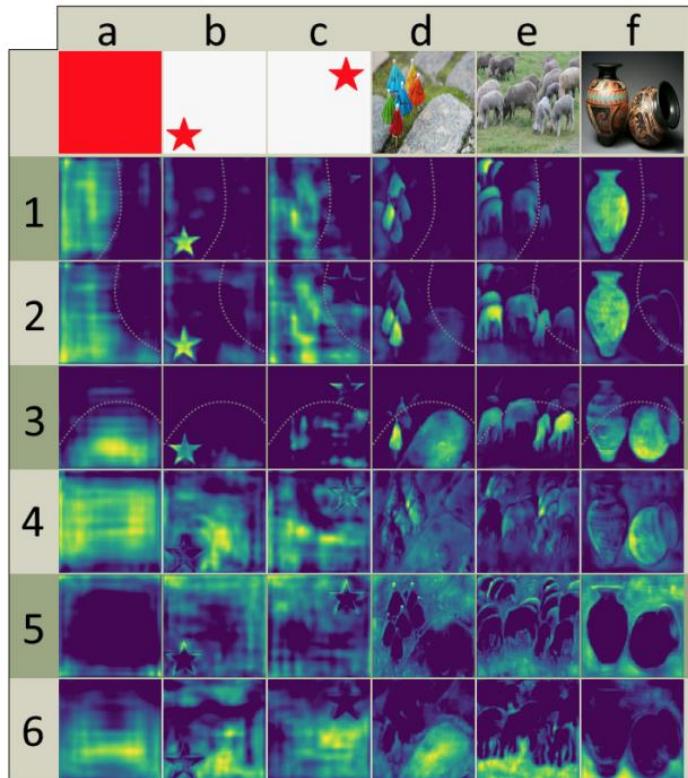
padding 추가  
왼쪽만 1로 활성화된 필터와 convolution 연산  
= 오른쪽만 활성화된 output



# 연구 이론적 배경

YOLACT Prediction Head

## Protonet의 Prototype mask

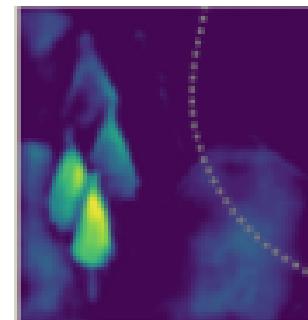


Protonet의 Prototype mask가 특정 지역을 activate하지만 하나의 instance만 activate하고 있지 않다

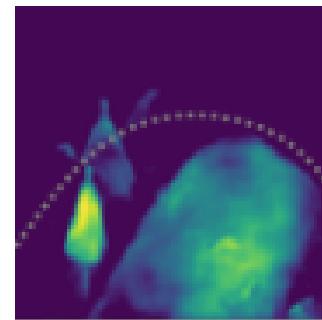
따라서 Prediction Head를 통해 Mask Coefficient 생성 및 적용하여 하나의 instance만 activate

## Mask Coefficient 적용 방법

Prototype mask끼리 연산함으로써 하나의 instance 만 activate (localization) 한다



Channel 2  
R, G 강조



Channel 3  
R 강조

= G  
Localization

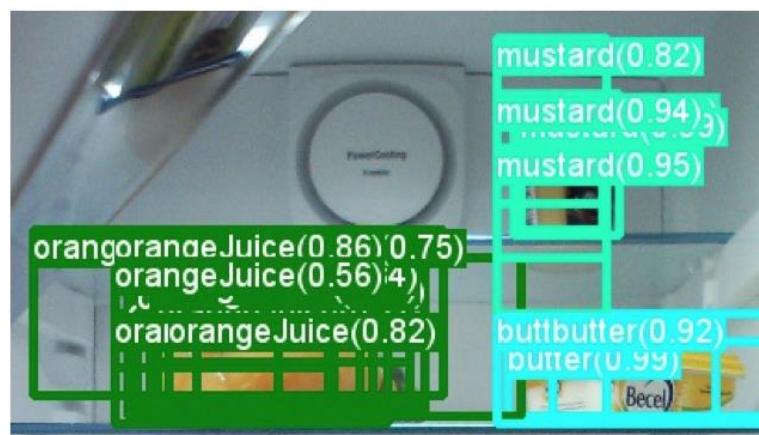


# 연구 이론적 배경

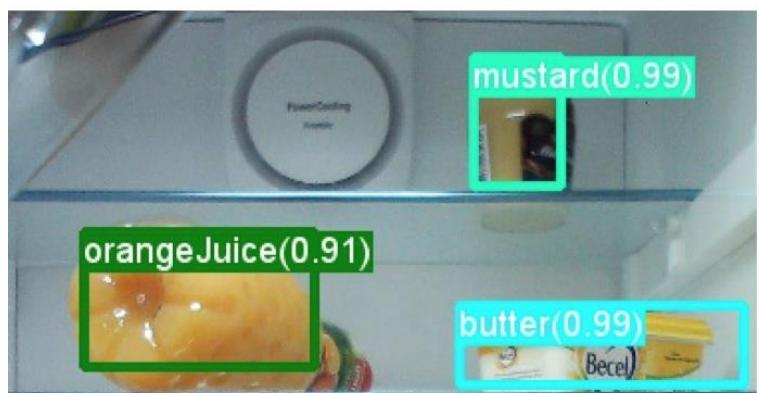
## YOLOACT Fast NMS

Substack을 병렬실행한 결과  
각 픽셀마다 앵커박스를 생성하는데  
여러 앵커박스가  
하나의 인스턴스를 focus하고 있기에 겹침

기존에는 NonMaximumSupperssion(NMS)를 적용하여  
신뢰도가 가장 높은 앵커박스만 남겨 Object Detection을 한다



NMS 적용 전



NMS 적용 후

행렬연산으로  
1. GPU로 연산 수행  
2. for문을 3번 → 1번

이에 따라 기존의 NMS보다 속도 향상

Sort by score							
0	$IoU_{12}$	$IoU_{13}$	$IoU_{14}$	$IoU_{15}$	$IoU_{16}$	$IoU_{17}$	
0	0	$IoU_{23}$	$IoU_{24}$	$IoU_{25}$	$IoU_{26}$	$IoU_{27}$	
0	0	0	$IoU_{34}$	$IoU_{35}$	$IoU_{36}$	$IoU_{37}$	
0	0	0	0	$IoU_{45}$	$IoU_{46}$	$IoU_{47}$	
0	0	0	0	0	$IoU_{56}$	$IoU_{57}$	
0	0	0	0	0	0	$IoU_{67}$	
0	0	0	0	0	0	0	0

If the largest IoU value compared with higher scores is under threshold, there is no need to suppress.

$$K = \begin{matrix} 0 & IoU_{12} & IoU_{13} & IoU_{24} & IoU_{25} & IoU_{56} & IoU_{57} \end{matrix}$$

$K < t?$

$$\begin{matrix} 0 & IoU_{12} & 0 & IoU_{24} & IoU_{25} & 0 & IoU_{57} \end{matrix}$$

Keep Box with score #1, #2, #4, #5, #7



## 연구 이론적 배경

### YOLACT 사용 이유

Method	Backbone	FPS	Time	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
PA-Net [29]	R-50-FPN	4.7	212.8	36.6	58.0	39.3	16.3	38.1	53.1
RetinaMask [14]	R-101-FPN	6.0	166.7	34.7	55.4	36.9	14.3	36.7	50.5
FCIS [24]	R-101-C5	6.6	151.5	29.5	51.5	30.2	8.0	31.0	49.7
Mask R-CNN [18]	R-101-FPN	8.6	116.3	35.7	58.0	37.8	15.5	38.1	52.4
MS R-CNN [20]	R-101-FPN	8.6	116.3	<b>38.3</b>	58.8	41.5	17.8	40.4	54.4
<b>YOLACT-550</b>	<b>R-101-FPN</b>	<b>33.5</b>	<b>29.8</b>	29.8	48.5	31.2	9.9	31.3	47.7
YOLACT-400	R-101-FPN	45.3	22.1	24.9	42.0	25.4	5.0	25.3	45.0
YOLACT-550	R-50-FPN	45.0	22.2	28.2	46.6	29.2	9.2	29.3	44.8
YOLACT-550	D-53-FPN	40.7	24.6	28.7	46.8	30.0	9.5	29.6	45.5
YOLACT-700	R-101-FPN	23.4	42.7	31.2	50.6	32.8	12.1	33.3	47.1

기존에 Object Detection 모델로 많이 쓰이는 Mask RCNN과 비교했을 때  
우월한 FPS와 Time을 보여주고 있다

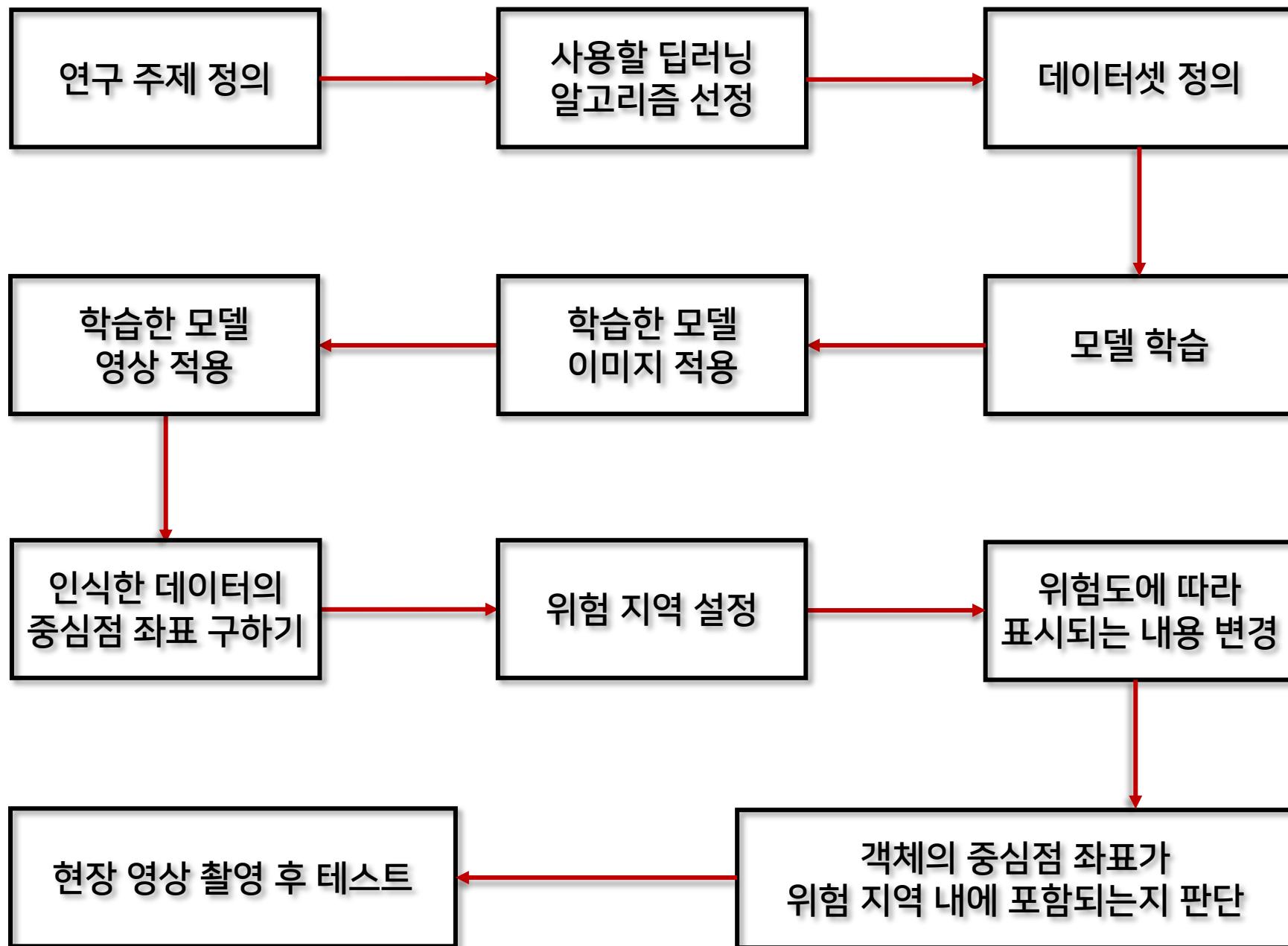
건축 현장 장비에 카메라를 설치하여 객체를 인식해 현장의 안전관리를 자동화하는 것을 목표로 함

따라서 실시간으로 보장된 segmentation 성능을 보이는 YOLACT 모델이 더 적합할 것이라 예상

## **4. 연구 내용 및 과정**



# 연구 진행 절차





# 데이터셋 생성

## 중장비 데이터 선정

[사고사례/2019년](#)

### 무안 아파트 공사현장 **굴삭기** 사고로 작업자 1명 사망

사용자 ulsansafety 2019. 7. 29. 16:32

### 도로에서 유도업무 중 후진하는 **덤프트럭**에 깔려 사망

♪ 김보현 | ◎ 입력 2019.06.07 09:52 | ◎ 수정 2019.06.07 11:00

### 광양 원료부두에서 석탄하역작업 중 **불도저**에 깔려 2명 사상 (2021.05.03)

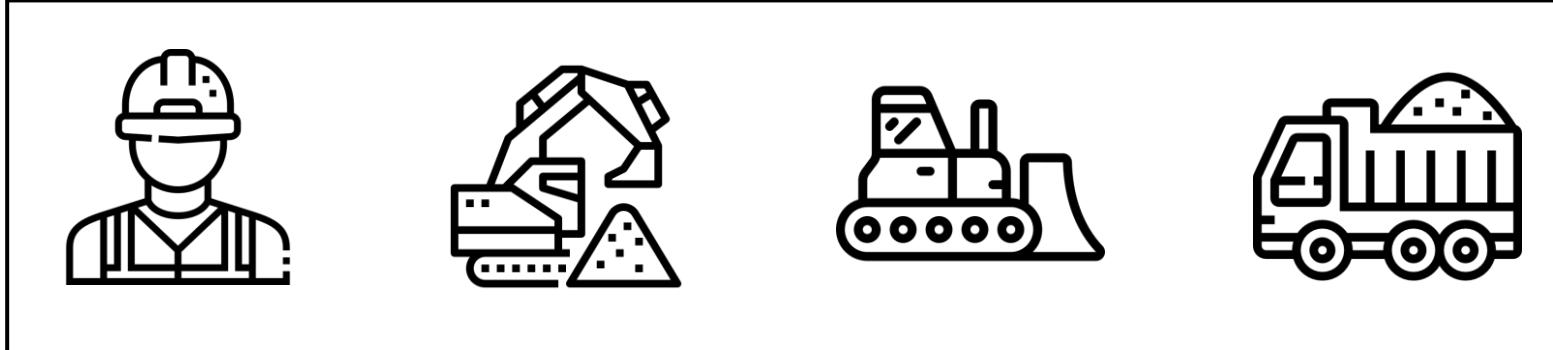
사용자 ulsansafety 2021. 5. 4. 17:58





# 데이터셋 생성

## 중장비 데이터 선정



labelme



Labelme2coco.py





# 데이터 라벨링

Labelme 프로그램 사용

labelme - C:/Users/PSY/Desktop/Window\_Main/2021/건축종합설계/이미지데이터/0.jpg

File Edit View Help

Open  
Open Dir  
Next Image  
Prev Image  
Save  
Delete File

Create Polygons

Edit Polygons

Duplicate Polygons

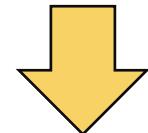
Delete Polygons

Undo

Brightness Contrast

162 %

Fit Width

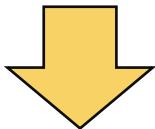




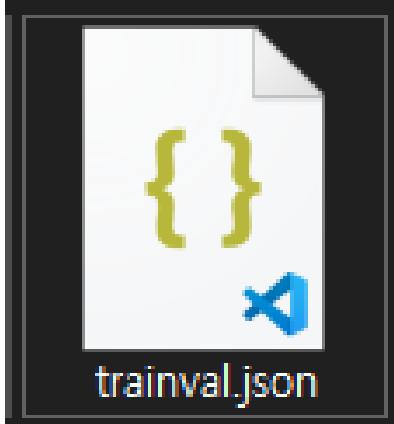
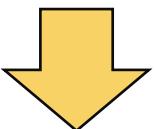
# 데이터 라벨링

Labelme2coco 사용 (코코데이터셋으로 변환)

이름	날짜	유형	크기
1.jpg	2008-10-01 오후 2:41	JPG 파일	377KB
1.json	2021-03-31 오전 12:37	JSON 원본 파일	118KB
2.jpg	2021-03-31 오전 12:38	JPG 파일	69KB
2.json	2021-03-31 오전 12:43	JSON 원본 파일	60KB
3.jpg	2021-03-31 오전 12:38	JPG 파일	59KB
3.json	2021-03-31 오전 12:55	JSON 원본 파일	62KB
4.jpg	2021-05-11 오후 5:01	JPG 파일	67KB



```
# Labelme2coco>python labelme2coco.py images
```



```
"categories": [  
    {  
        "supercategory": "bulldozer",  
        "id": 0,  
        "name": "bulldozer"  
    },  
    {  
        "supercategory": "excavator",  
        "id": 1,  
        "name": "excavator"  
    },  
    {  
        "supercategory": "person",  
        "id": 2,  
        "name": "person"  
    },  
    {  
        "supercategory": "truck",  
        "id": 3,  
        "name": "truck"  
    }],
```



# 모델 학습

config.py 파일 수정

```
187 √ equipment_dataset = dataset_base.copy({  
188     'name': 'Equipment',  
189     'train_info': '/content/drive/MyDrive/YOLCAT/Data/train/trainval_0519.json', → 학습용 데이터셋  
190     'train_images': '/content/drive/MyDrive/YOLCAT/Data/train/images',  
191     'valid_info': '/content/drive/MyDrive/YOLCAT/Data/val/val.json', → 검증용 데이터셋  
192     'valid_images': '/content/drive/MyDrive/YOLCAT/Data/val/images',  
193     'class_names': ('bulldozer', 'excavator', 'person', 'truck'),  
194     'label_map': { 0: 1, 1: 2, 2: 3, 3:4 } } ) → 생성한 데이터셋 Class
```

```
800 √ yolact_resnet50_equipment_config = yolact_resnet50_config.copy({  
801     'name': 'yolact_plus_resnet50_equipment',  
802     # Dataset stuff  
803     'dataset': equipment_dataset,  
804     'num_classes': len(equipment_dataset.class_names) + 1,  
805  
806     'max_size': 512,  
807 })
```



# 모델 학습

모델학습방법&모델정밀도

```
[ ] !python /content/drive/MyDrive/YOLCAT/yolact/train.py --config=yolact_resnet50_equipment_config
```

```
loading annotations into memory...
Done (t=0.95s)
creating index...
index created!
loading annotations into memory...
Done (t=0.50s)
creating index...
index created!
Initializing weights...
Begin training!
```

학습에 사용할 데이터 셋

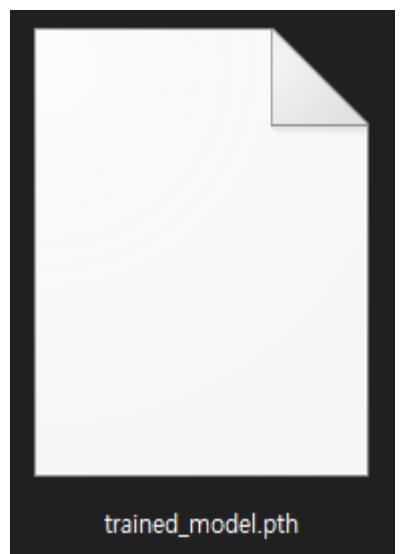
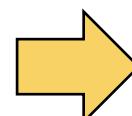
```
tcmalloc: large alloc 1657511936 bytes == 0x557b330c8000 @ 0x7fdc35808001 0x7fdc32e8054f 0x7fdc32ed0b58 0x7fdc3
[236] 1890 || B: 0.359 | C: 0.231 | M: 0.765 | S: 0.056 | T: 1.412 || ETA: 21 days, 12:45:01 || timer: 1.753
```

```
Computing validation mAP (this may take a while)...
```

모델 정밀도

```
Calculating mAP...
```

	.all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95	
box	59.47	69.62	69.62	69.62	69.62	69.62	69.62	69.62	62.44	33.66	11.29	
mask	45.76	69.62	69.62	51.53	51.53	50.25	50.00	50.00	39.73	25.33	0.00	



trained\_model.pth



# 학습 결과 확인

이미지 인식 확인





# 학습 결과 확인

영상인식 확인





# 객체의 중심 좌표 표시

## eval.py

```
if args.display_text or args.display_bboxes:
    for j in reversed(range(num_dets_to_consider)):
        x1, y1, x2, y2 = boxes[j, :]
        color = get_color(j)
        score = scores[j]

        if args.display_bboxes:
            cv2.rectangle(img_numpy, (x1, y1), (x2, y2), color, 1)

        if args.display_text:
            _class = cfg.dataset.class_names[classes[j]]
            text_str = '%s: %.2f' % (_class, score) if args.display_scores else _class

            font_face = cv2.FONT_HERSHEY_DUPLEX
            font_scale = 0.6
            font_thickness = 1

            text_w, text_h = cv2.getTextSize(text_str, font_face, font_scale, font_thickness)[0]

            text_pt = (x1, y1 - 3)
            text_color = [255, 255, 255]

            m1 = int((x1 + x2) / 2) # 가로 중심
            m2 = int((y1 + y2) / 2) # 세로 중심

            text_text = "%s %s" % (m1, m2)

            cv2.rectangle(img_numpy, (x1, y1), (x1 + text_w, y1 - text_h - 4), color, -1)
            cv2.putText(img_numpy, text_str, text_pt, font_face, font_scale, text_color, font_thickness, cv2.LINE_AA)
            # cv2.putText(img_numpy, text_text, text_pt, font_face, font_scale, text_color, font_thickness, cv2.LINE_AA)
            cv2.circle(img_numpy, (m1, m2), 5, (0, 0, 255), -1)
```



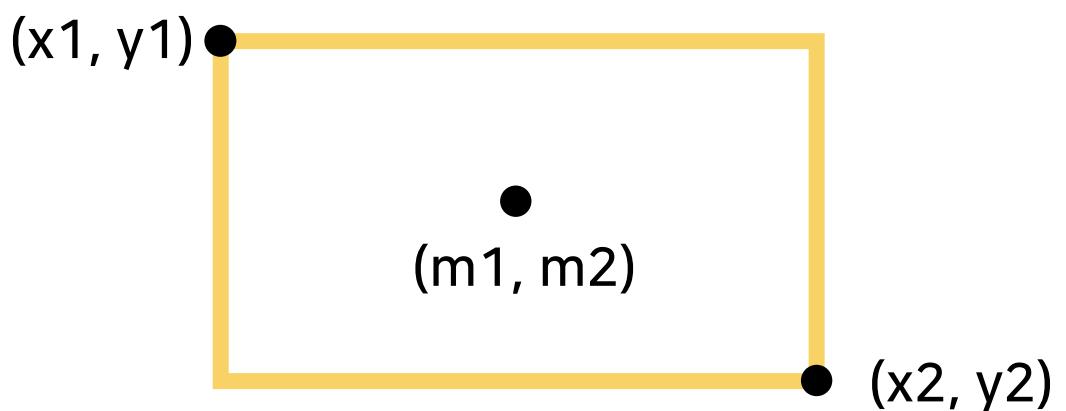
## 객체의 중심 좌표 표시

eval.py

```
if args.display_bboxes:  
    cv2.rectangle(img_numpy, (x1, y1), (x2, y2), color, 1)
```



```
m1 = int((x1 + x2) / 2) # 가로 중심  
m2 = int((y1 + y2) / 2) # 세로 중심
```



```
cv2.circle(img_numpy, (m1, m2), 5, (0,0,255), -1)
```



# 객체의 중심 좌표 표시

영상인식 결과





## 위험 지역 설정 설정 방법

~~선서~~



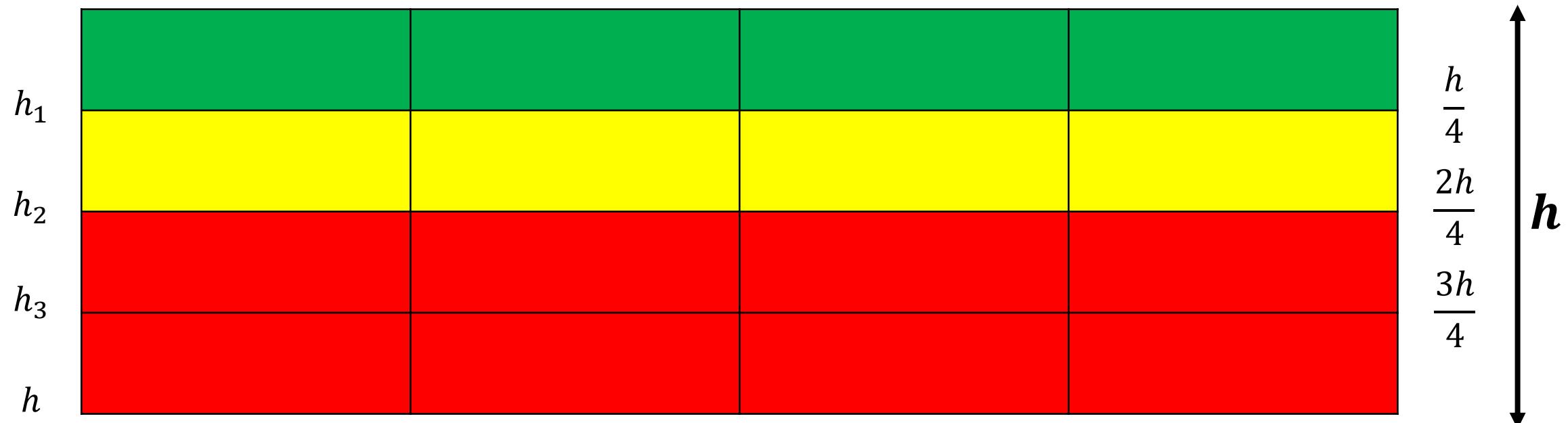
객체 간 거리  
측정 불가





# 위험 지역 설정

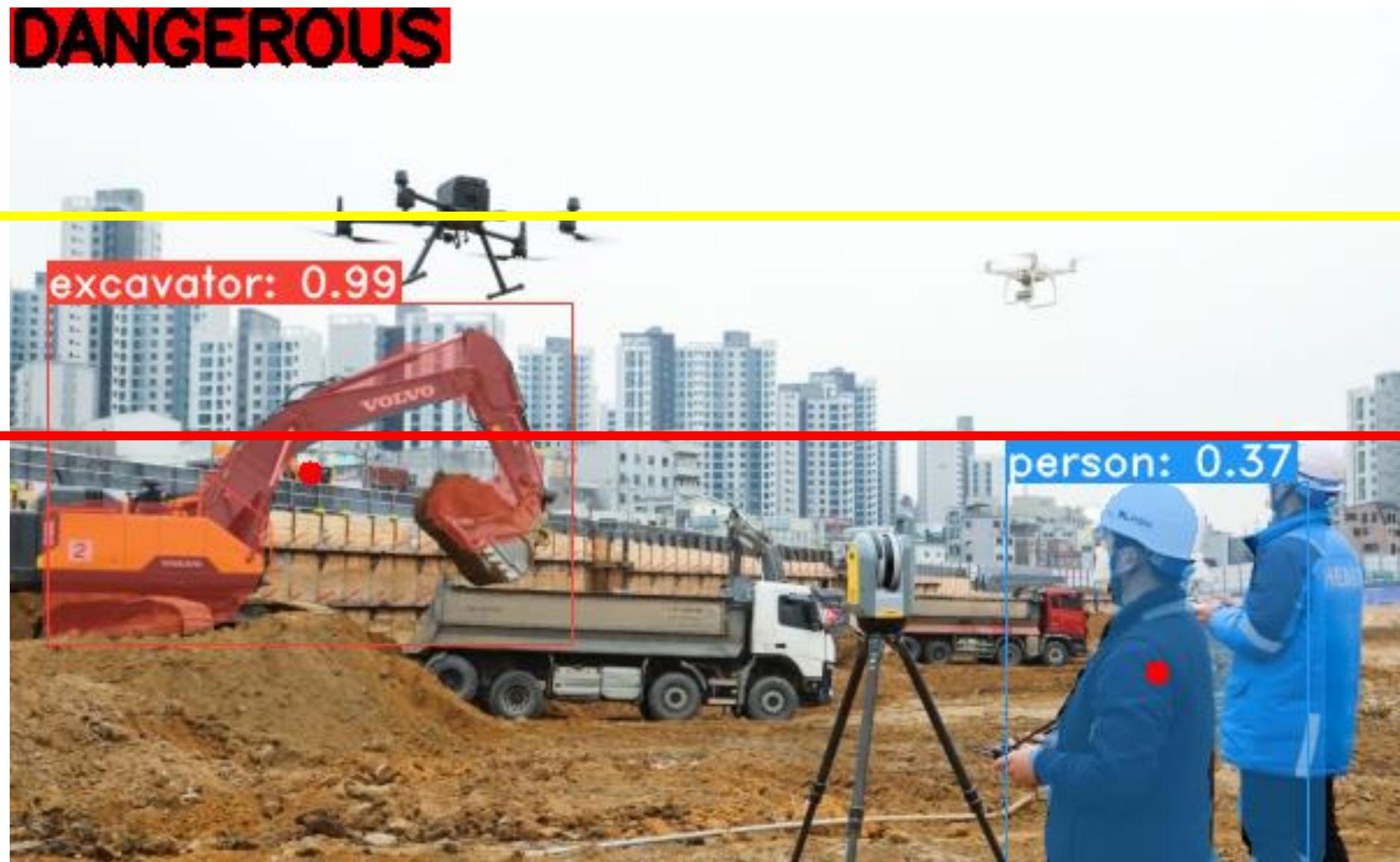
설정 방법





## 위험 지역 설정

이미지 결과





## 위험 지역 설정

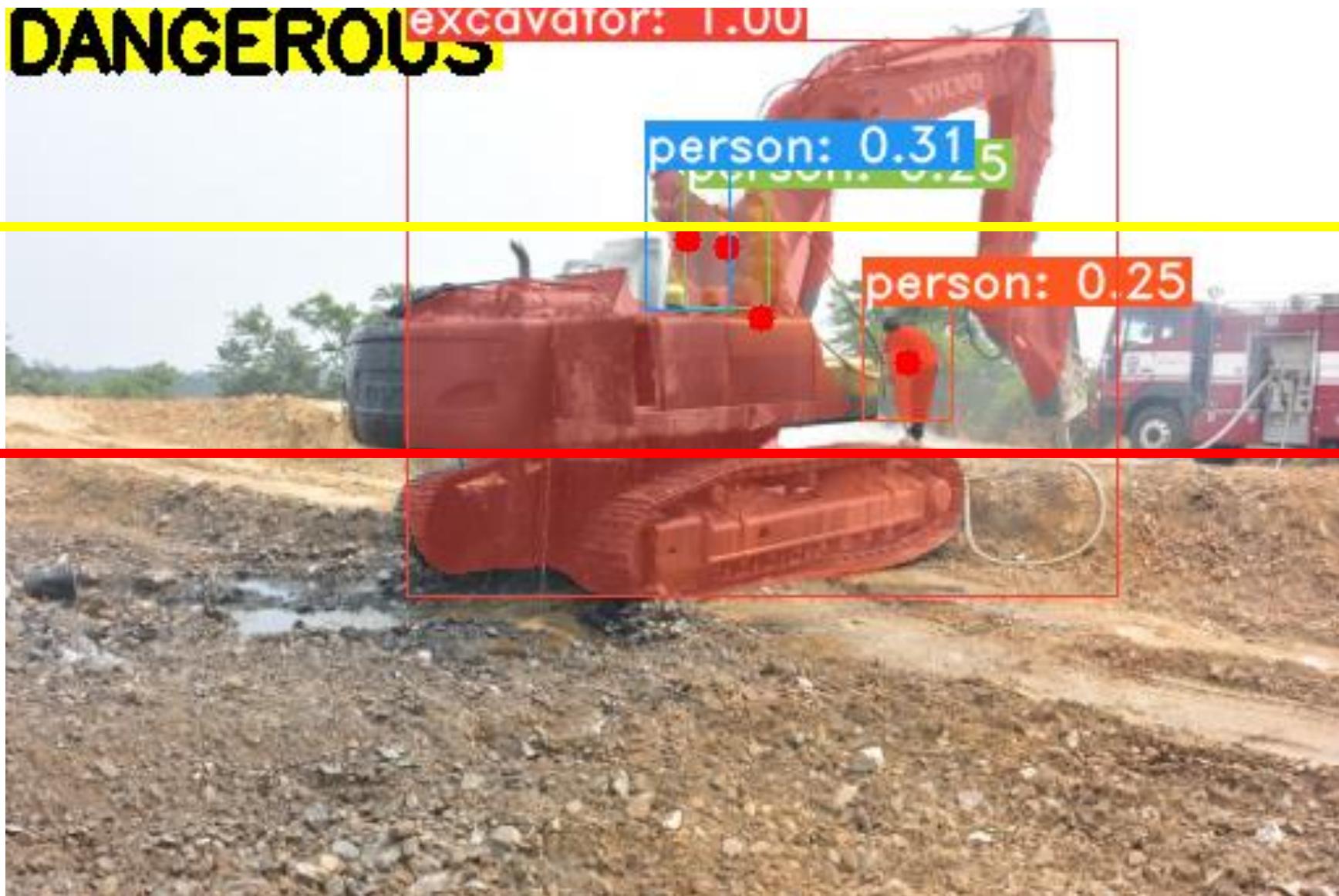
이미지 결과

**DANGEROUS**

excavator: 1.00

person: 0.31

person: 0.25





## 최종 결과

현장 영상에 최종모델 적용

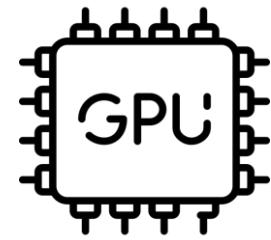
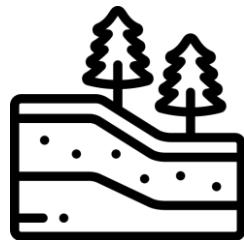
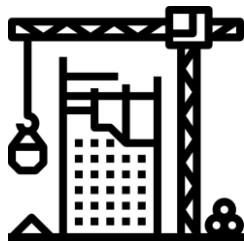


## **5. 결론 및 한계점**



## 결론 및 한계점

연구한계 및 후속 연구 방향



사운드 출력 하드웨어

실제 현장 적용

건설 지형 고려

CUDA GPU 사용



# 결론 및 한계점

연구 결론



산업 재해 중  
높은 비율을 차지하는  
건설 현장 사고

딥러닝 모델

YOLACT

알고리즘

객체 중심점 도출  
영상 위험 지역 설정



딥러닝을 활용한  
건설 현장  
안전관리 자동화

## **6. 참고문헌**



## 참고문헌

심승보, 최상일, “YOLO-v3을 활용한 건설 장비 주변 위험 상황 인지 알고리즘 개발(2019)”

Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee,  
“YOLACT: Real-time Instance Segmentation(2019)”

YOLACT github, “Installation YOLACT”,  
<https://github.com/dbolya/yolact>

Immersive Limit , “Train YOLACT with a Custom COCO Dataset”,  
<https://www.immersivelimit.com/tutorials/train-yolact-with-a-custom-coco-dataset>

핸즈온민러닝, “Make COCO dataset”,  
<https://hansonminlearning.tistory.com/52>

프라이데이, “YOLACT 설명 및 정리”,  
<https://ganghee-lee.tistory.com/42>



---

감사합니다