

"REAL-TIME OPEN-VOCABULARY DETECTION FOR COMPREHENSIVE SAFETY AND SECURITY APPLICATIONS"

안전 및 보안 응용을 위한 실시간 Open-vocabulary Object Detection

0. Index

Part 1

1

Project Motivation

프로젝트 동기

2

YOLO-World

모델링 과정

3

Demo

데모

Part 2

4

Issue

데모 이후 문제점

5

Solution

해결 방안

6

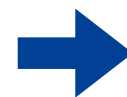
Result & Applications

결과 및 응용

1. Project Motivation

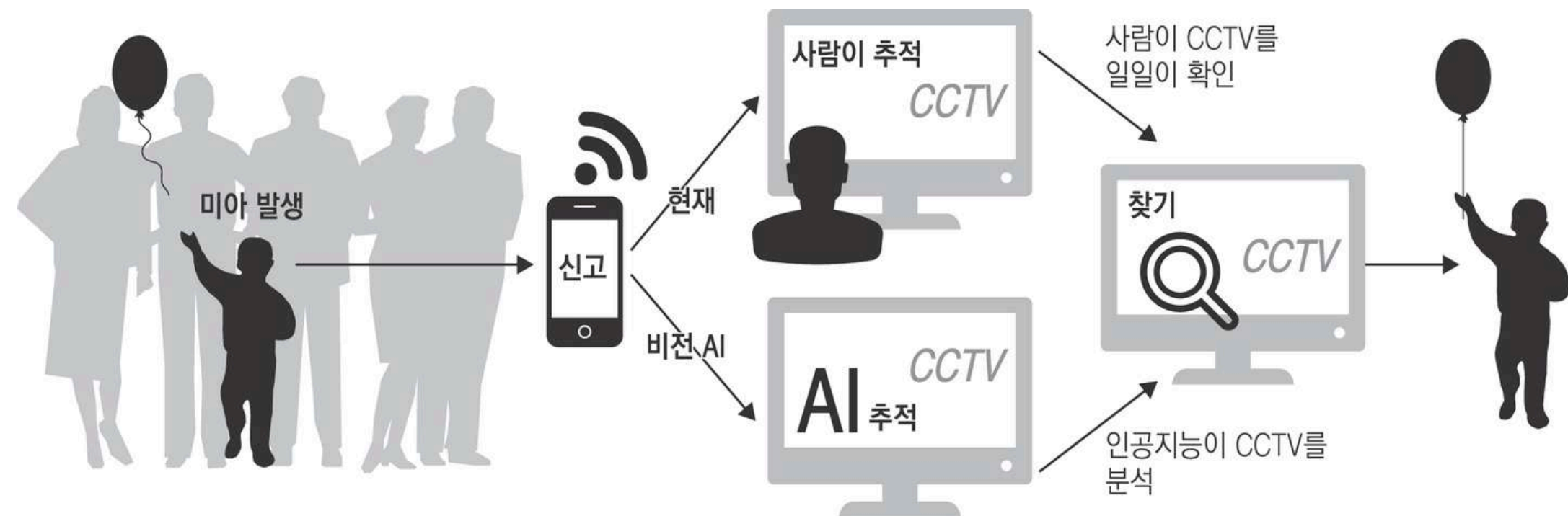
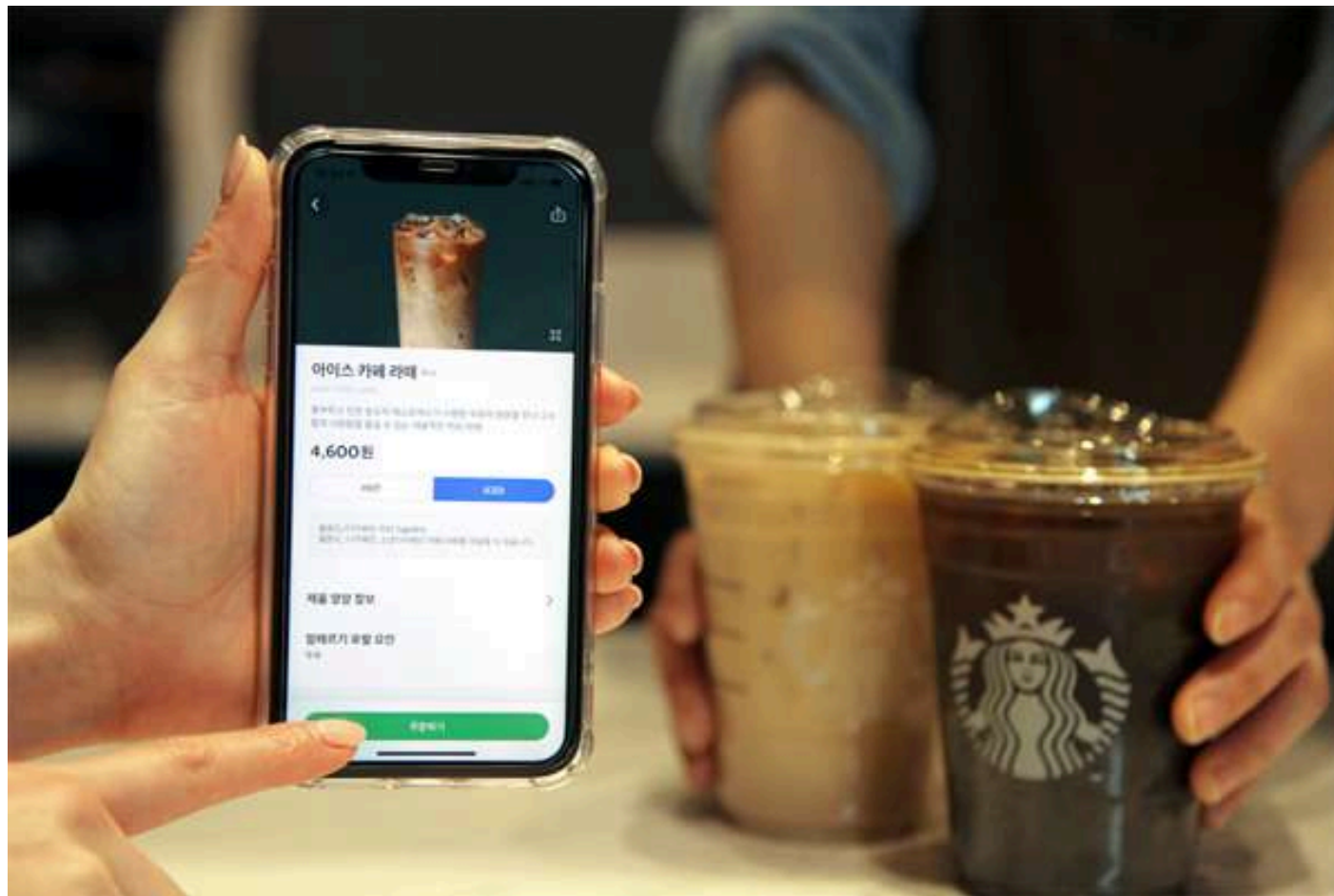
문제 인식

- 사이렌 오더 도난 이슈
- 미아 찾기, 용의자 추적



해결 방안

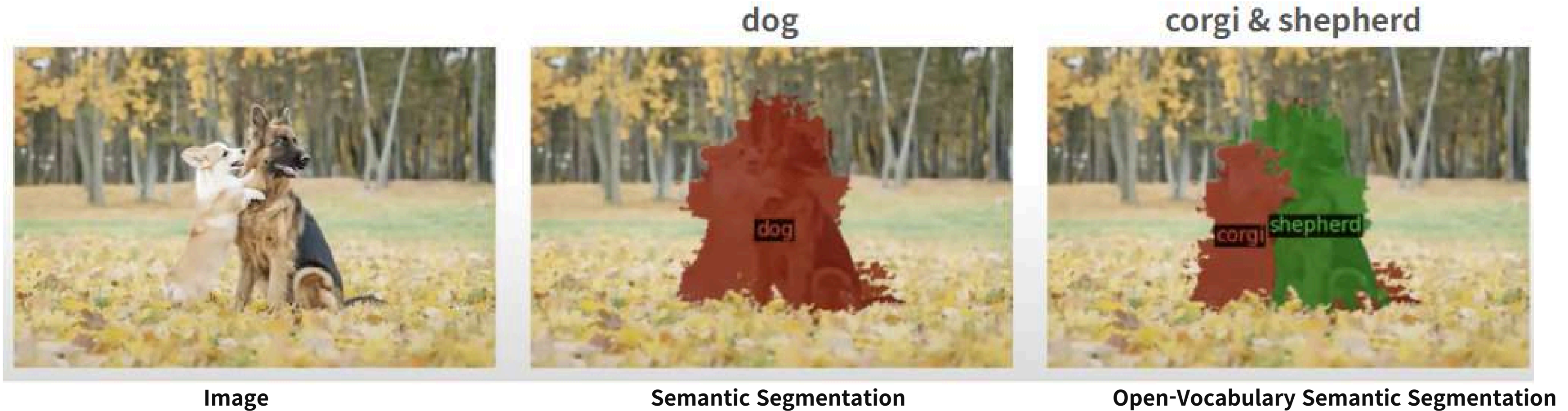
- Cctv에 인상착의를 바탕으로 인물 검거
- 사전 정의된 class만 감지 가능한 object detection으로 불가능
- Open-vocabulary object detection을 통해 해결!!



1. Project Motivation

Open-Vocabulary Setting

- class 개수나 카테고리 등에 제한을 두지 않고 학습시킴으로써 임의의 text description에 따라 이미지를 detection



1. Project Motivation

구현 목표

1) 컴퓨팅 자원이 없는 환경

- Jetson Xavier NX에 Open Vocabulary Object Detection 모델을 배포하여, GPU가 없는 CCTV 환경에서도 작동

2) 사용자 친화적 웹 사이트 제작

- 사용자의 free form text 입력(ex: "person in blue shirt", "person wearing hat")을 통해 간편하게 detection 수행

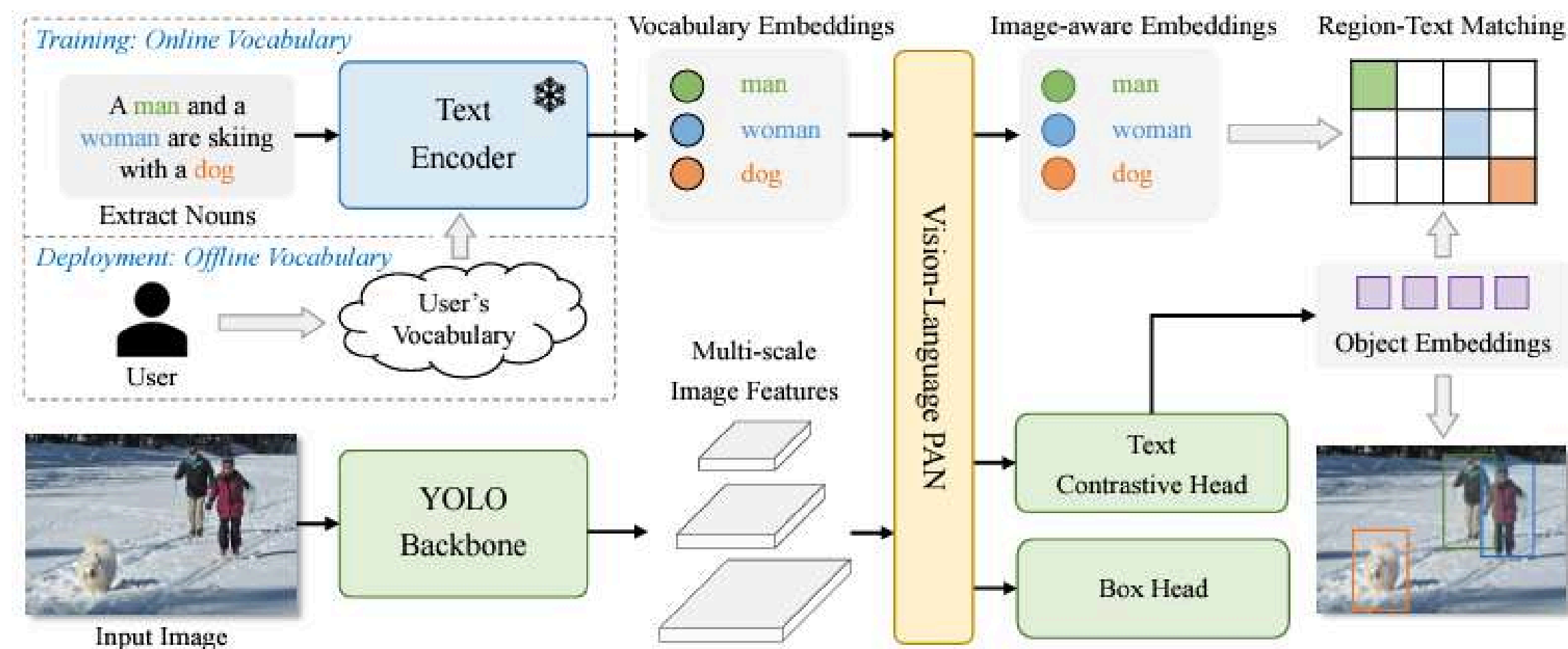
3) 실시간 적용

- 카메라로 입력된 영상을 실시간 처리하여 inference 결과 시각화

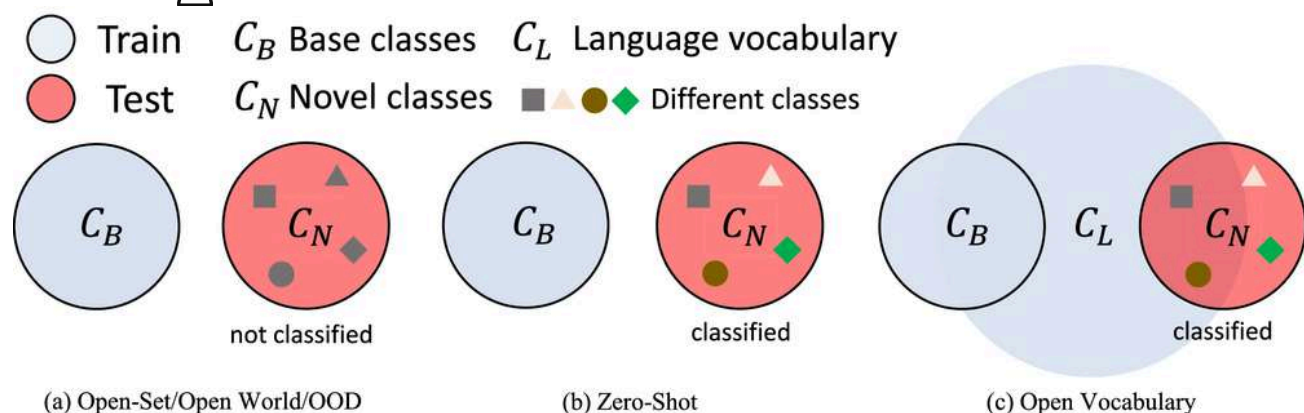


2. YOLO-World: Real-Time Open-Vocabulary Object Detection (CVPR 2024)

• Model Architecture : CLIP + YOLO



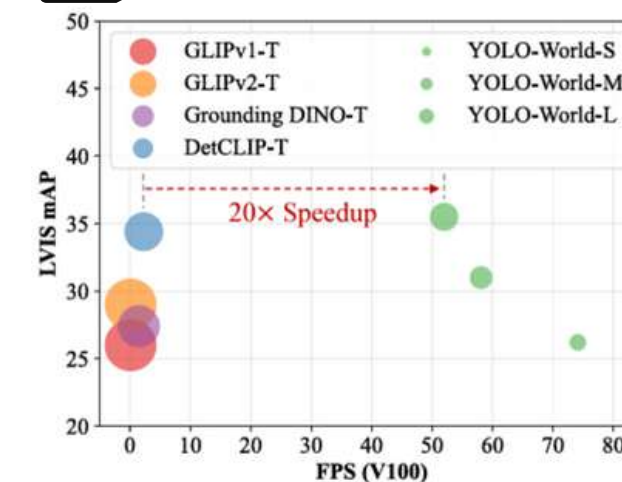
Open-Vocabulary Zero-Shot Detection



Text-Based Flexibility



Real-Time Speed



3. Demo

1. Zero-shot demo on image



{men, women, boy, girl} {elephant, ear, leg, trunk, ivory}



the person in red

the brown animal



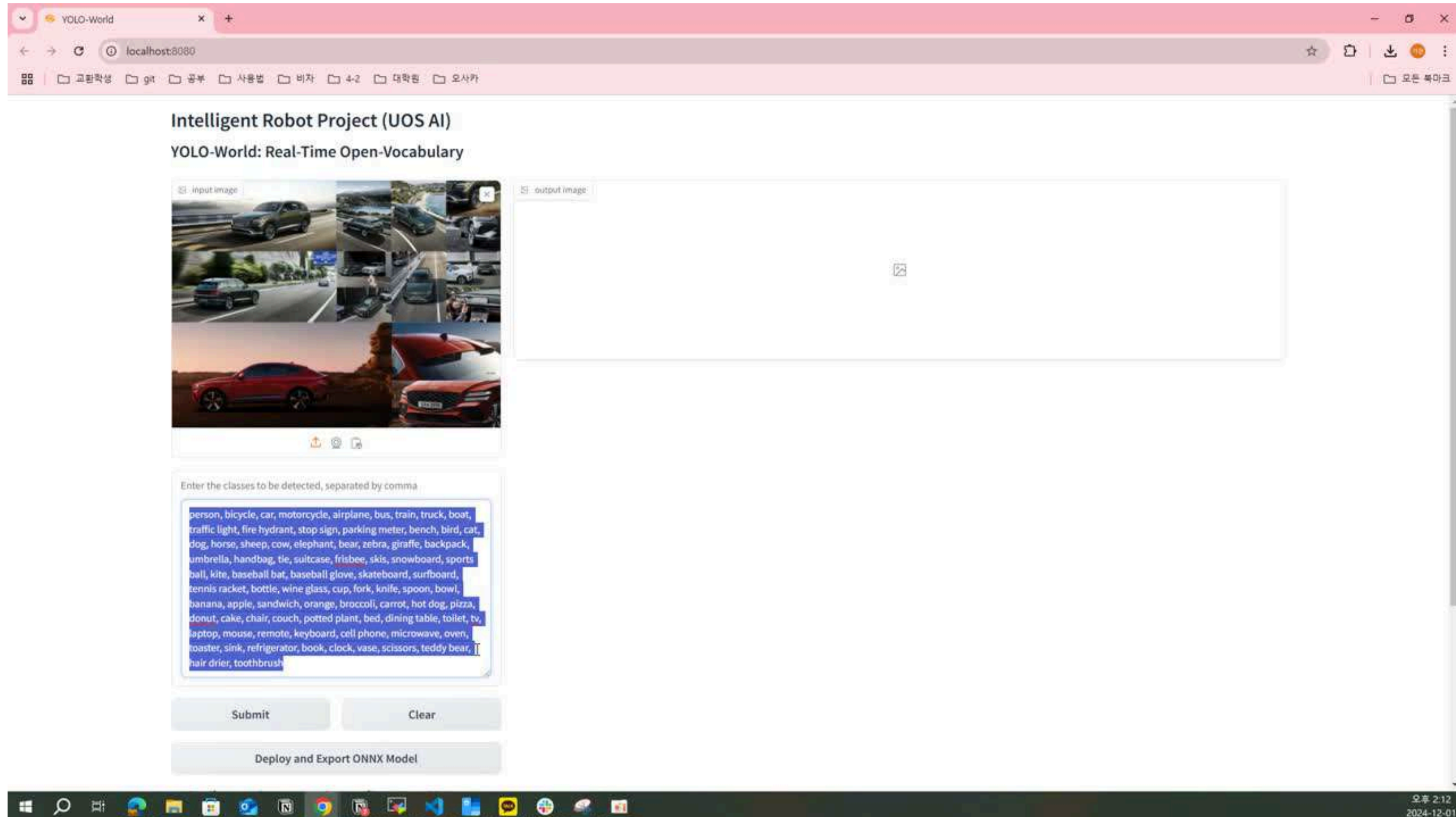
the tallest person

person with a white shirt

2. Real-time demo



3. DEMO (Zero-shot image demo available online)




3. DEMO - Image Demo (1)

- 같은 클래스 내 세부 차이를 구별하며, 사전 정의되지 않은 다양한 객체를 유연하게 인식 가능

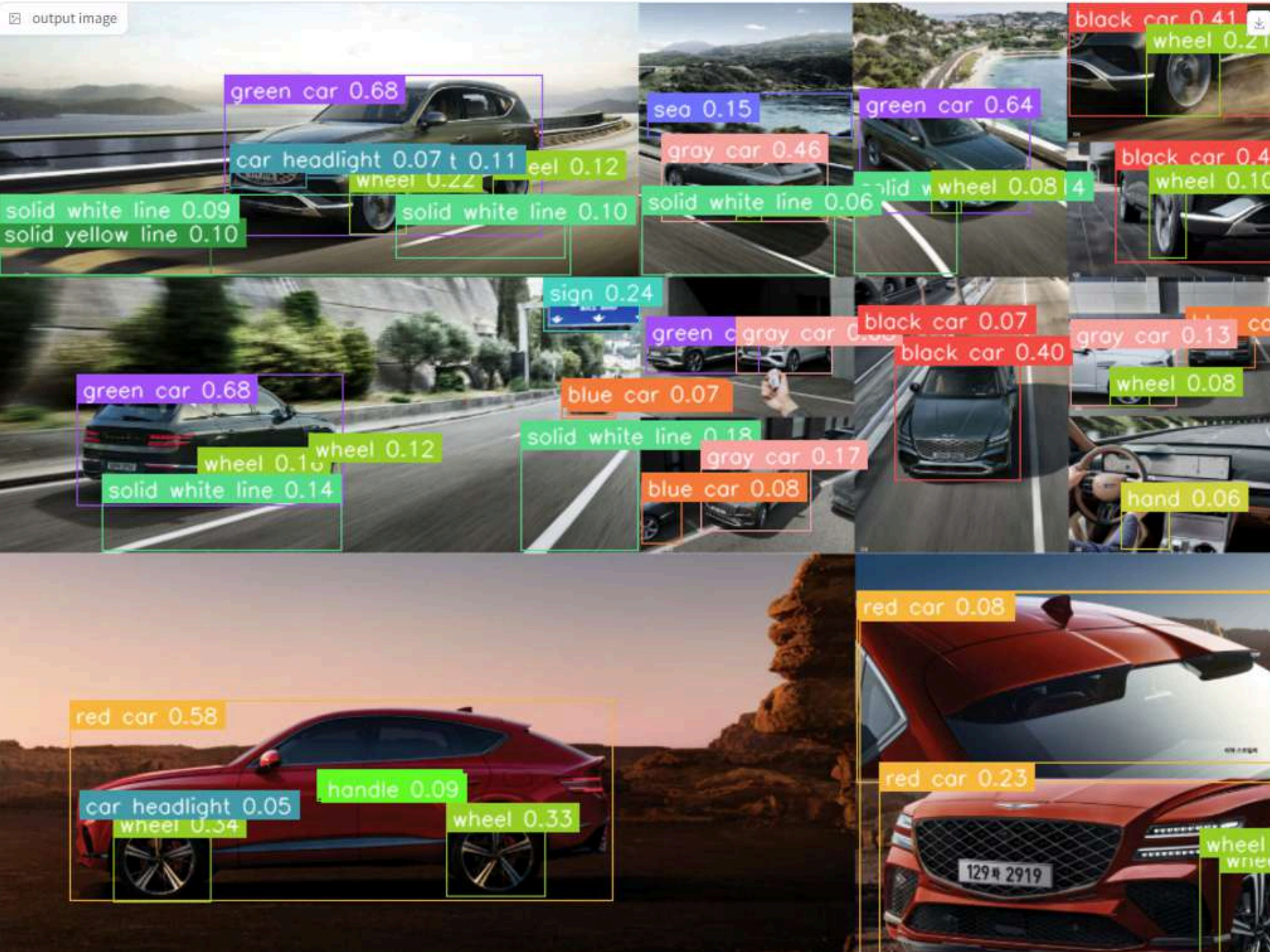
Intelligent Robot Project (UOS AI)

YOLO-World: Real-Time Open-Vocabulary

☐ input image



☐ output image



Enter the classes to be detected, separated by comma

green car, black car, gray car, blue car, red car, hand, handle, wheel, solid white line, solid yellow line, sign, car headlight, monitor, hand, sea

Submit Clear

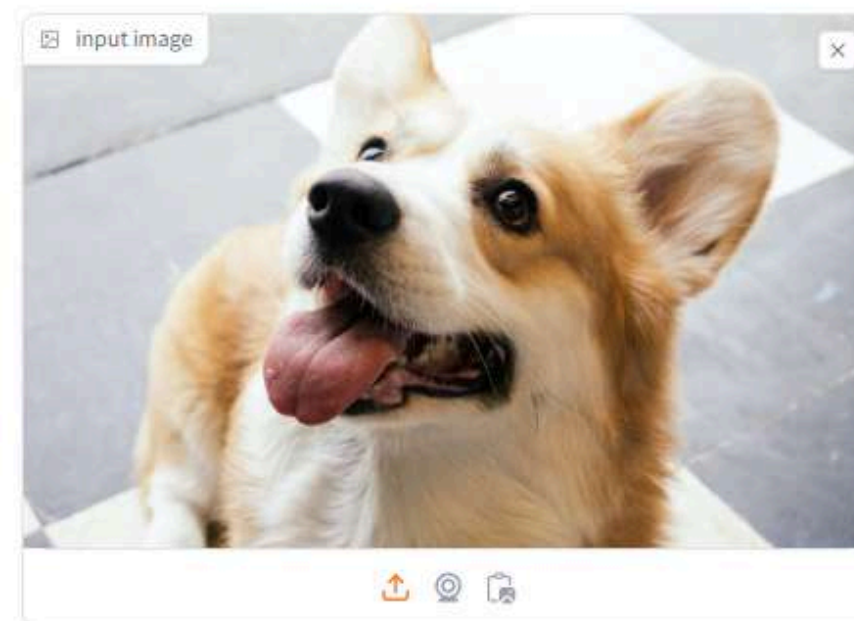
Deploy and Export ONNX Model

It takes a few seconds to generate the ONNX file! YOLO-World-Seg (segmentation) is not supported now

3. DEMO - Image Demo (2)

- 객체의 세부 구조까지 강력하게 탐지 가능

Intelligent Robot Project (UOS AI)
YOLO-World: Real-Time Open-Vocabulary



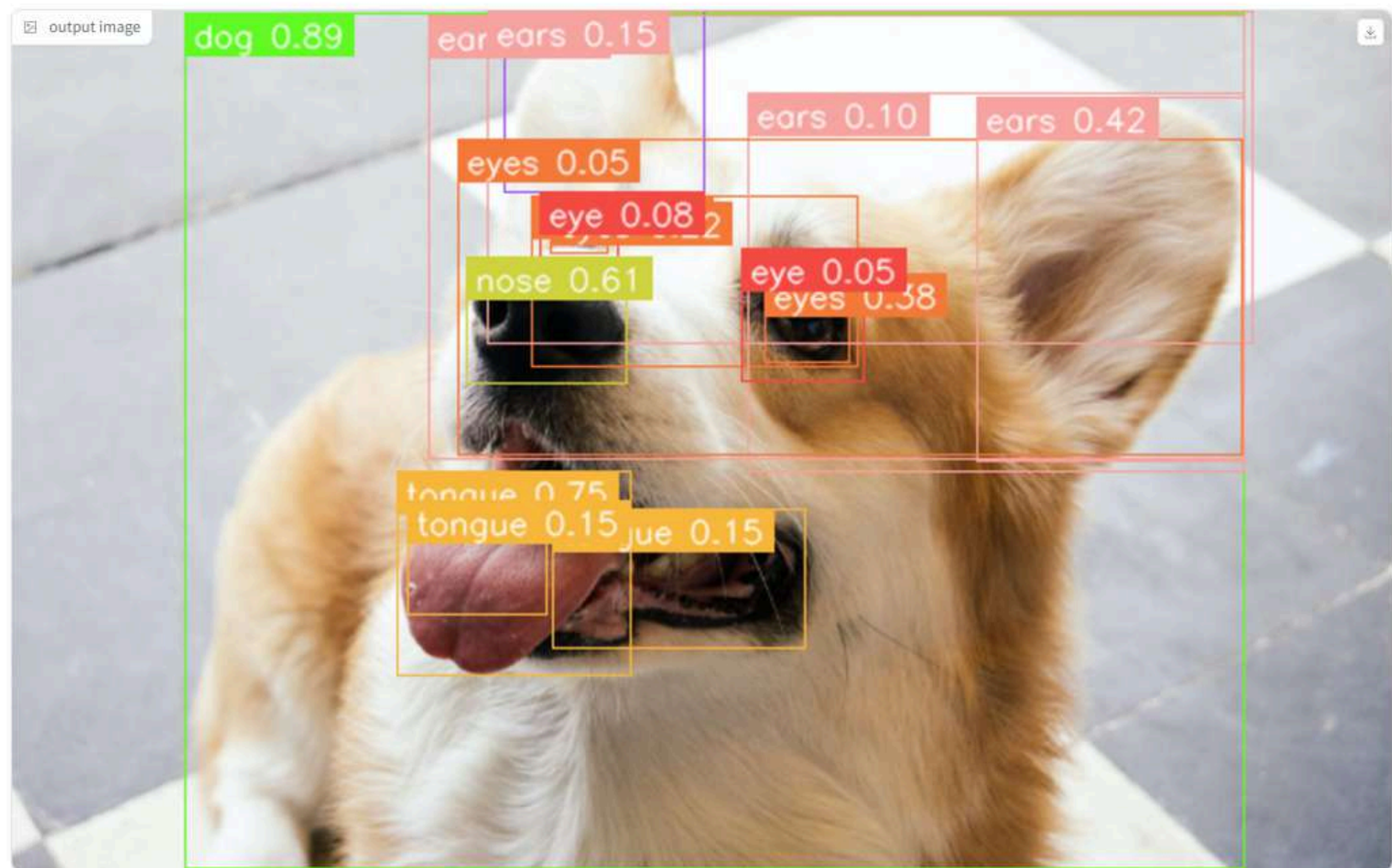
Enter the classes to be detected, separated by comma

ear, eye, ears, eyes, tongue, nose, dog

Submit

Clear

Deploy and Export ONNX Model

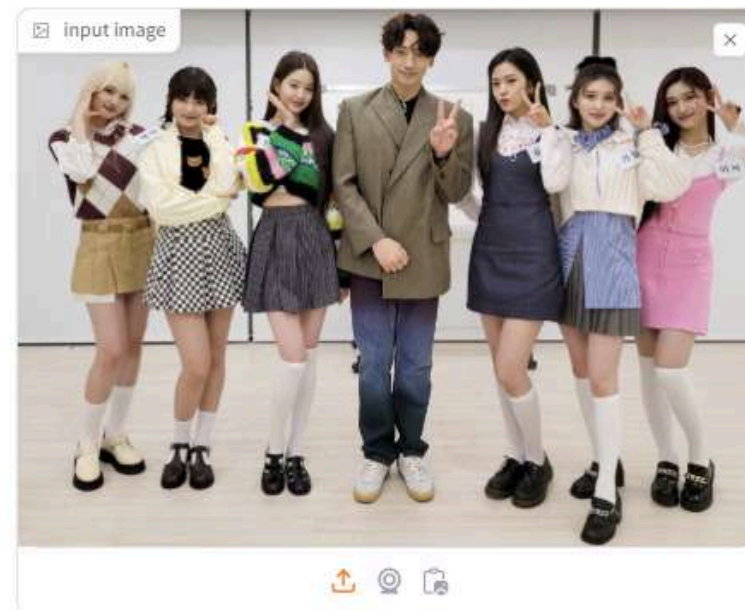


3. DEMO - Image Demo (3)

- 동일한 객체의 키와 같은 세부 특징도 구별 가능

Intelligent Robot Project (UOS AI)

YOLO-World: Real-Time Open-Vocabulary



Enter the classes to be detected, separated by comma

the tallest person

Submit

Clear

Deploy and Export ONNX Model

It takes a few seconds to generate the ONNX file! YOLO-World-Seg (segmentation) is not supported now



It takes a few seconds to generate the ONNX file! YOLO-World-Seg (segmentation) is not supported now

Maximum Number Boxes 100

Score Threshold 0.521

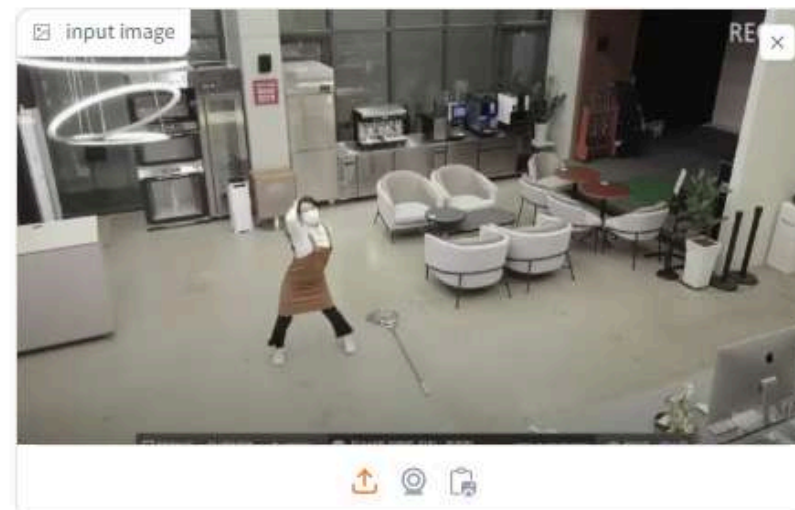
NMS Threshold 0.7

3. DEMO - Image Demo (Our Task)

- Task(사이렌 오더, 미아 방지)에 맞는 CCTV 뷰 이미지에서도 강력한 성능을 발휘
- YOLO와 달리 자유 형식의 텍스트 입력을 통해 사전에 정의된 카테고리에 의존하지 않고 제로샷 추론을 수행할 수 있음

Intelligent Robot Project (UOS AI)

YOLO-World: Real-Time Open-Vocabulary



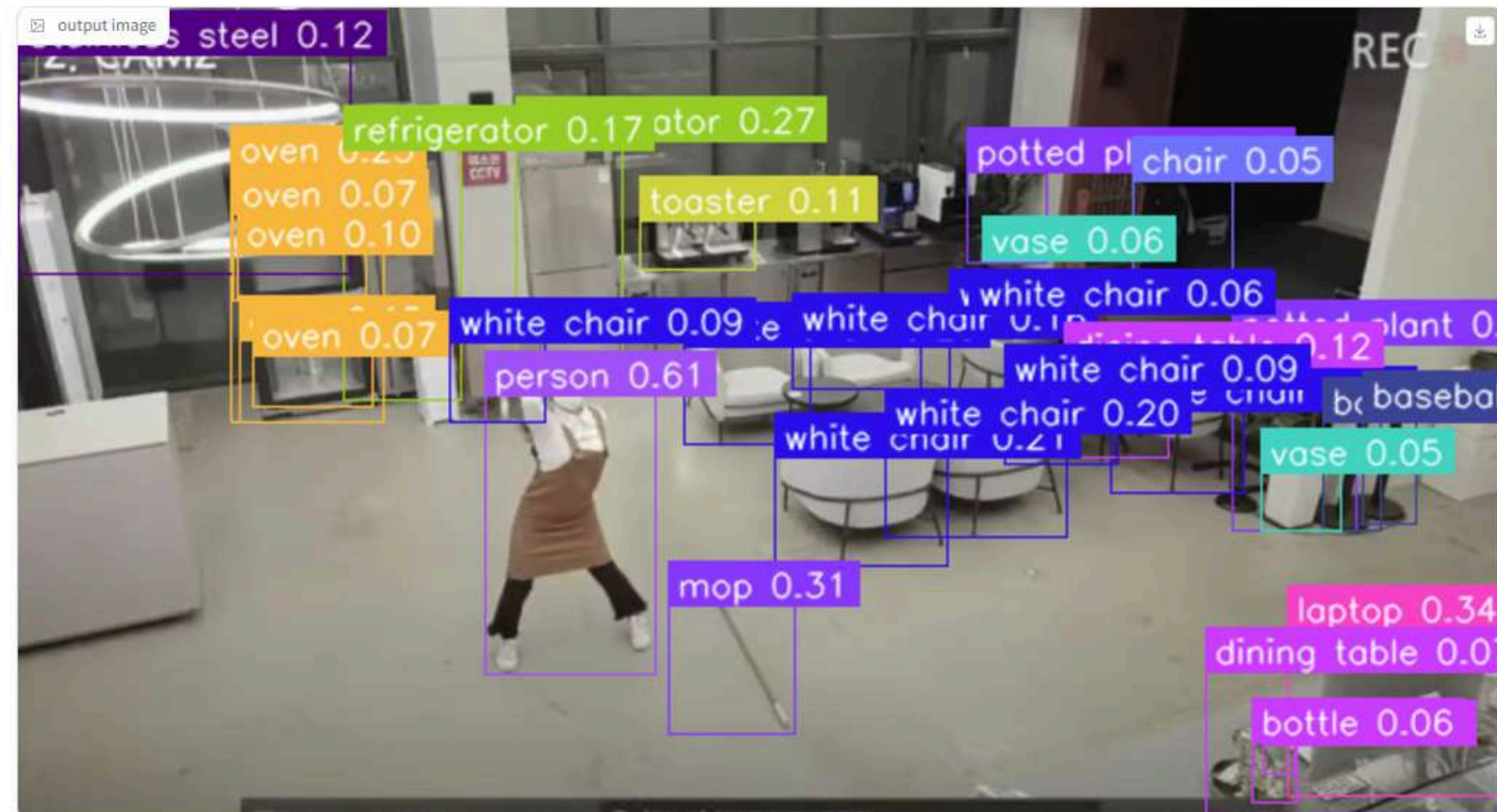
Enter the classes to be detected, separated by comma

person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, laptop, mouse, remote, keyboard, cell phone, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush, white chair, mop, Stainless steel

Submit

Clear

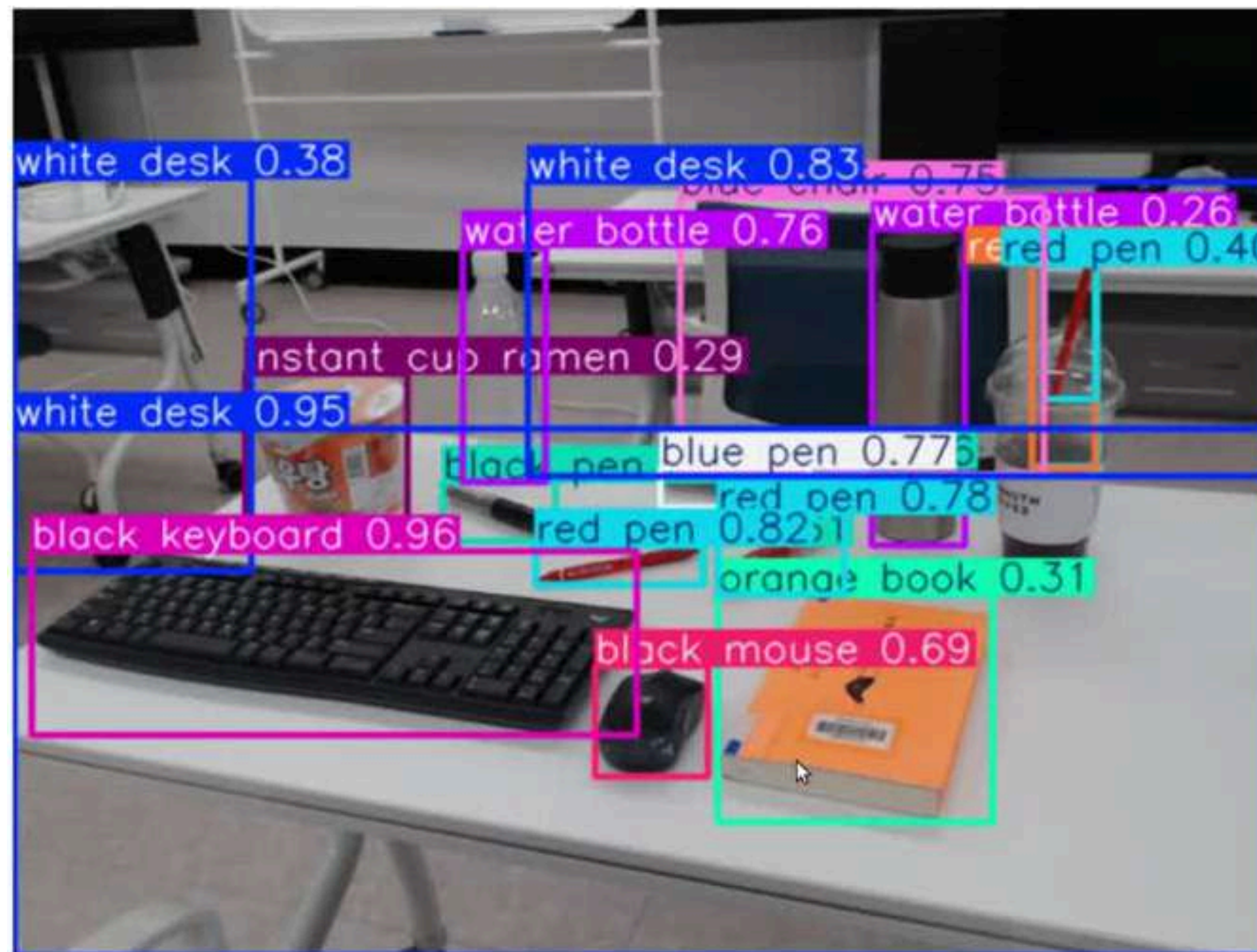
Deploy and Export ONNX Model



3. DEMO - Real Time Demo (1)

Free-form text input (지능형 로봇 연구실 내 모든 객체 입력)

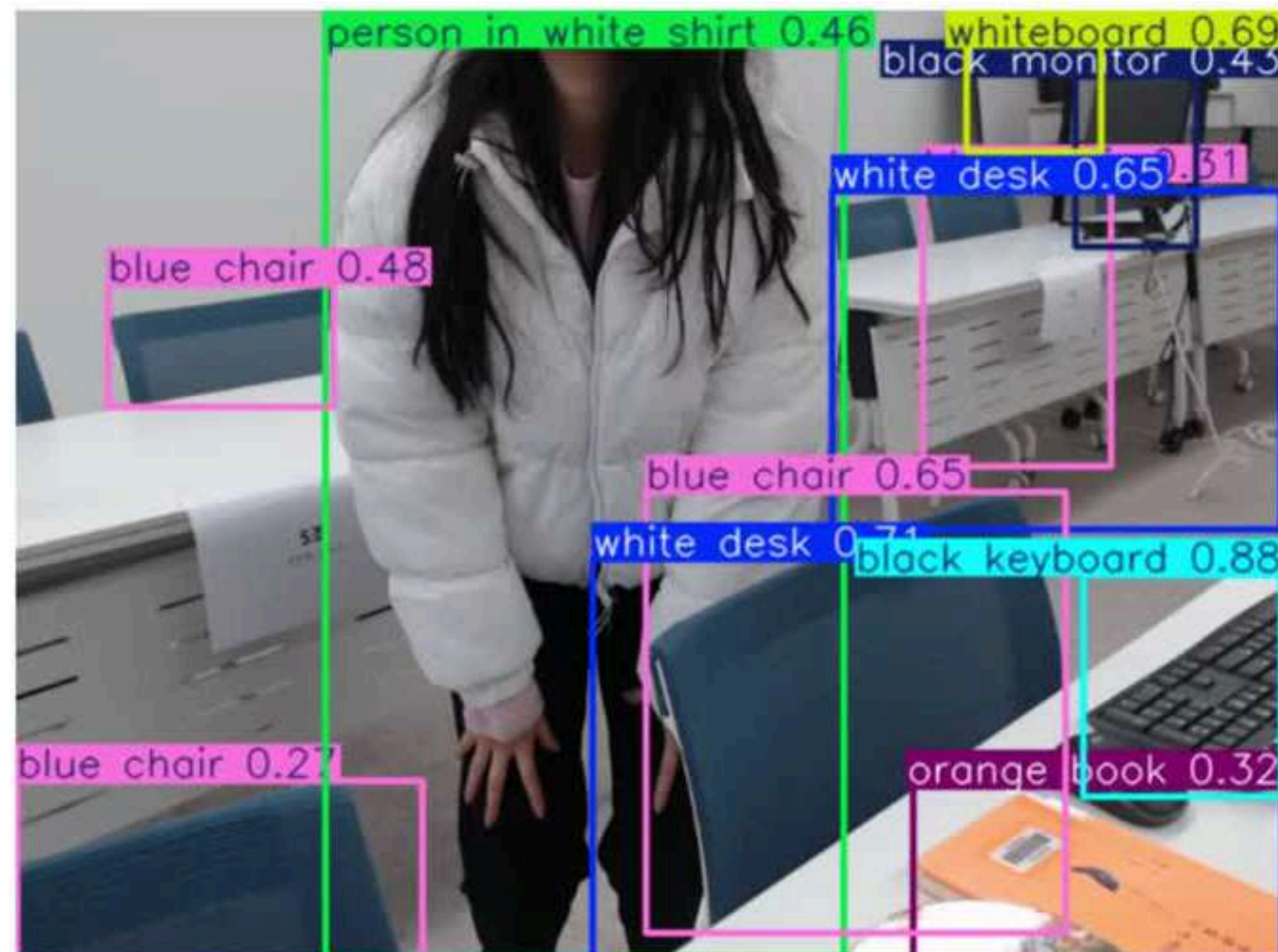
- white desk, red pen, blue pen, black pen, black monitor, blue chair, person in black shirt, whiteboard, person in white shirt, water bottle, cup of coffee, black keyboard, orange book, orange cup, instant cup ramen, red straw



3. DEMO - Real Time Demo (2)

Free-form text input

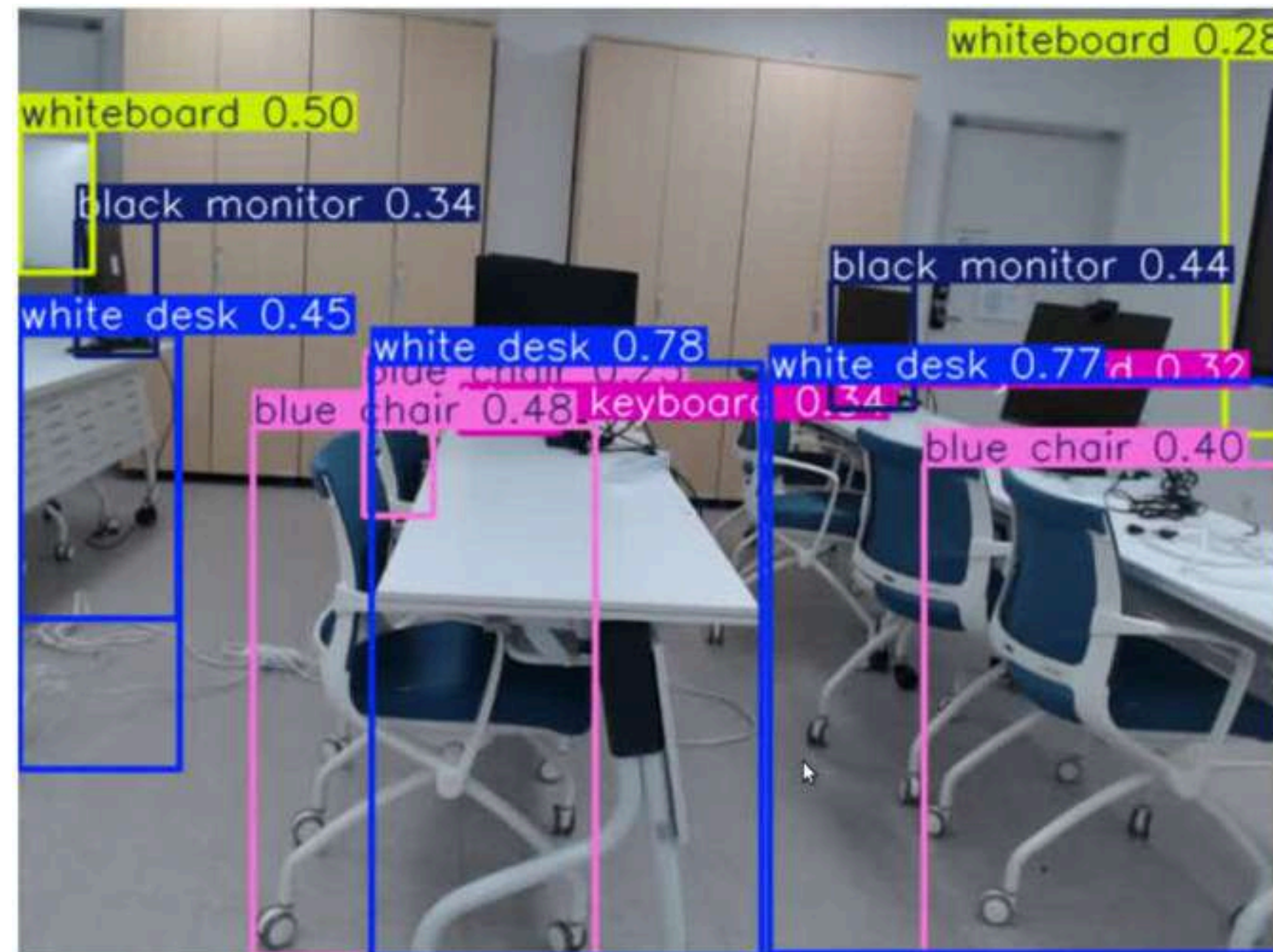
- white desk, red pen, blue pen, black pen, black monitor, blue chair, person in black shirt, whiteboard, **person in white shirt**, water bottle, cup of coffee, black keyboard, orange book, orange cup, instant cup ramen, red straw



3. DEMO - Real Time Demo (3)

Free-form text input

- white desk, red pen, blue pen, black pen, black monitor, blue chair, person in black shirt, whiteboard, person in white shirt, water bottle, cup of coffee, black keyboard, orange book, orange cup, instant cup ramen, red straw




- Strong performance confirmed in real-time scenarios!!

4. Issue

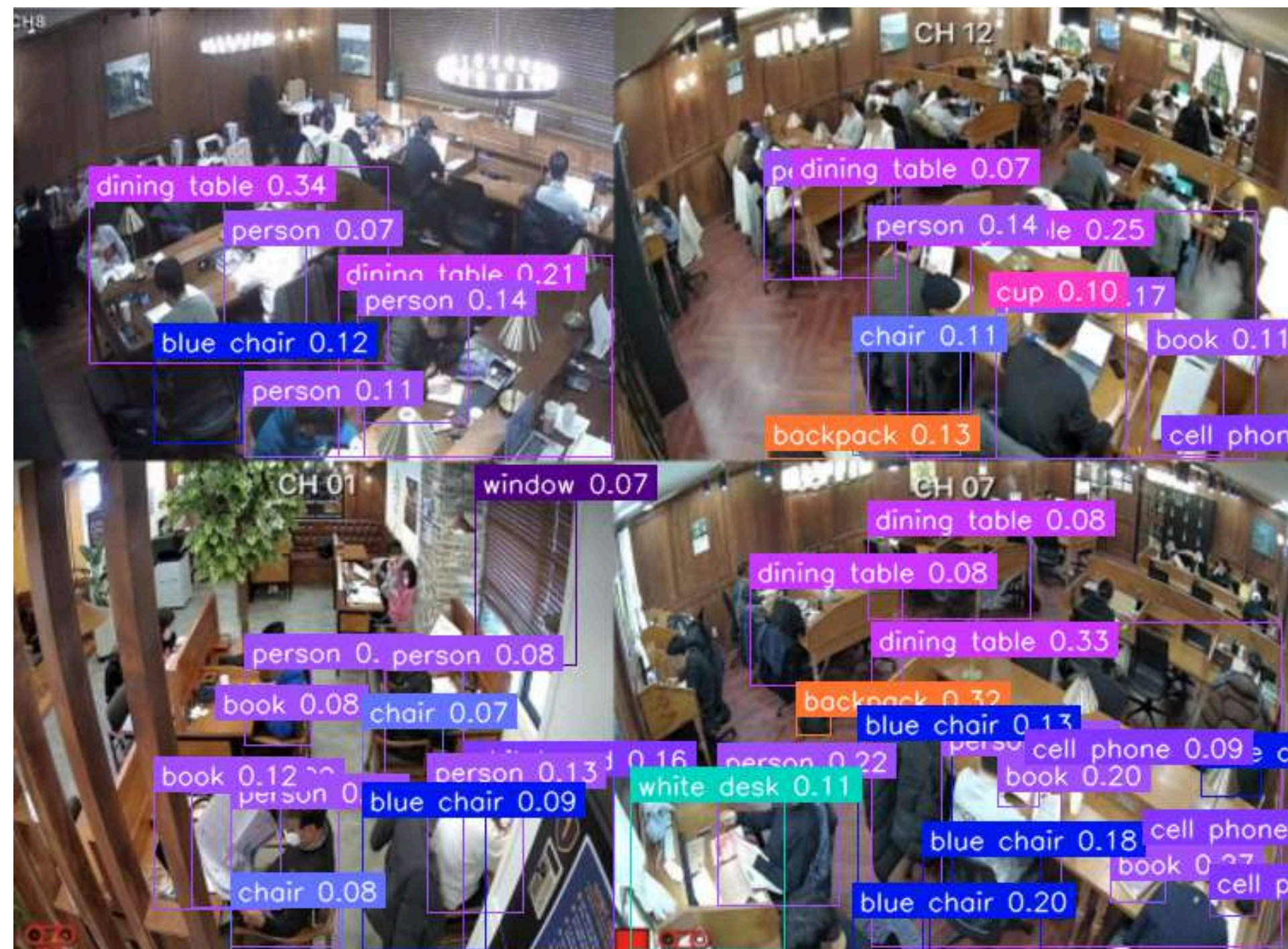
- Issue 1: Performance issues with **Small object detection**
- Issue 2: Performance Issues with **Closely Packed Objects**

☐ input image



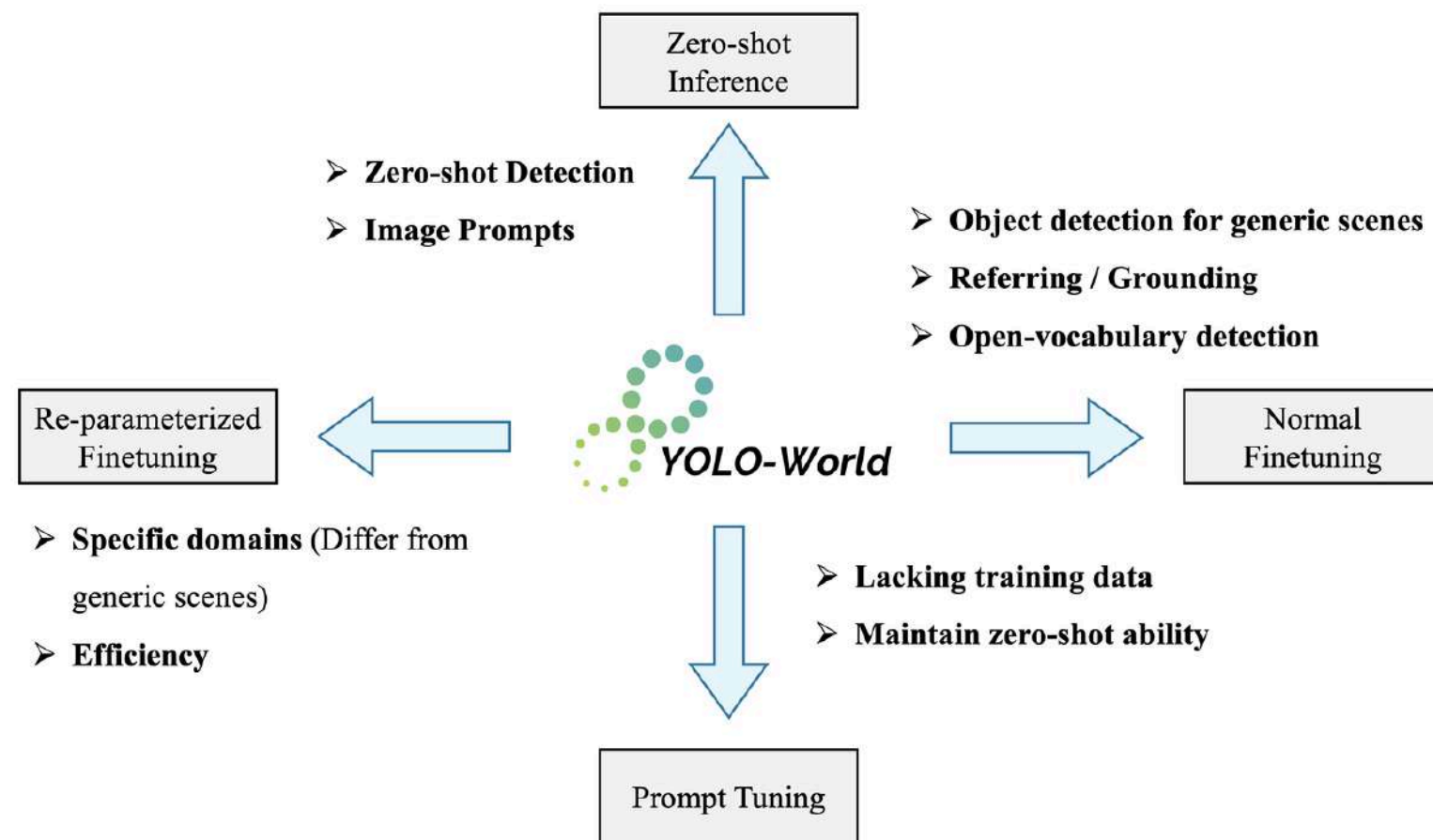
Enter the classes to be detected, separated by comma
 person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, cell phone, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush, white chair, mop, Stainless steel, white desk, red pen, blue pen, black pen, black monitor, blue chair, whiteboard, window, dining table

Submit
 Clear



5. Solution (Fine-Tuning)

Solution: More Fine-tuning!



Datasets for Issue 1: Performance Issues with Small Object Detection



VisDrone example

Datasets for Issue 2: Performance Issues with Closely Packed Objects



SKU-110k example

- Purpose: To address issues with small object detection and closely packed objects.

5. Solution (Fine-Tuning)

• Fine -Tuning : SKU-110k Dataset

Epochs: 20

Batch Size: 16

Image Size: 640 px

Optimizer: Auto-selected (SGD/Adam based on configuration)

Learning Rate:

Initial: 0.01

Final: $0.01 \times 0.01 = 0.0001$ (using cosine scheduler)

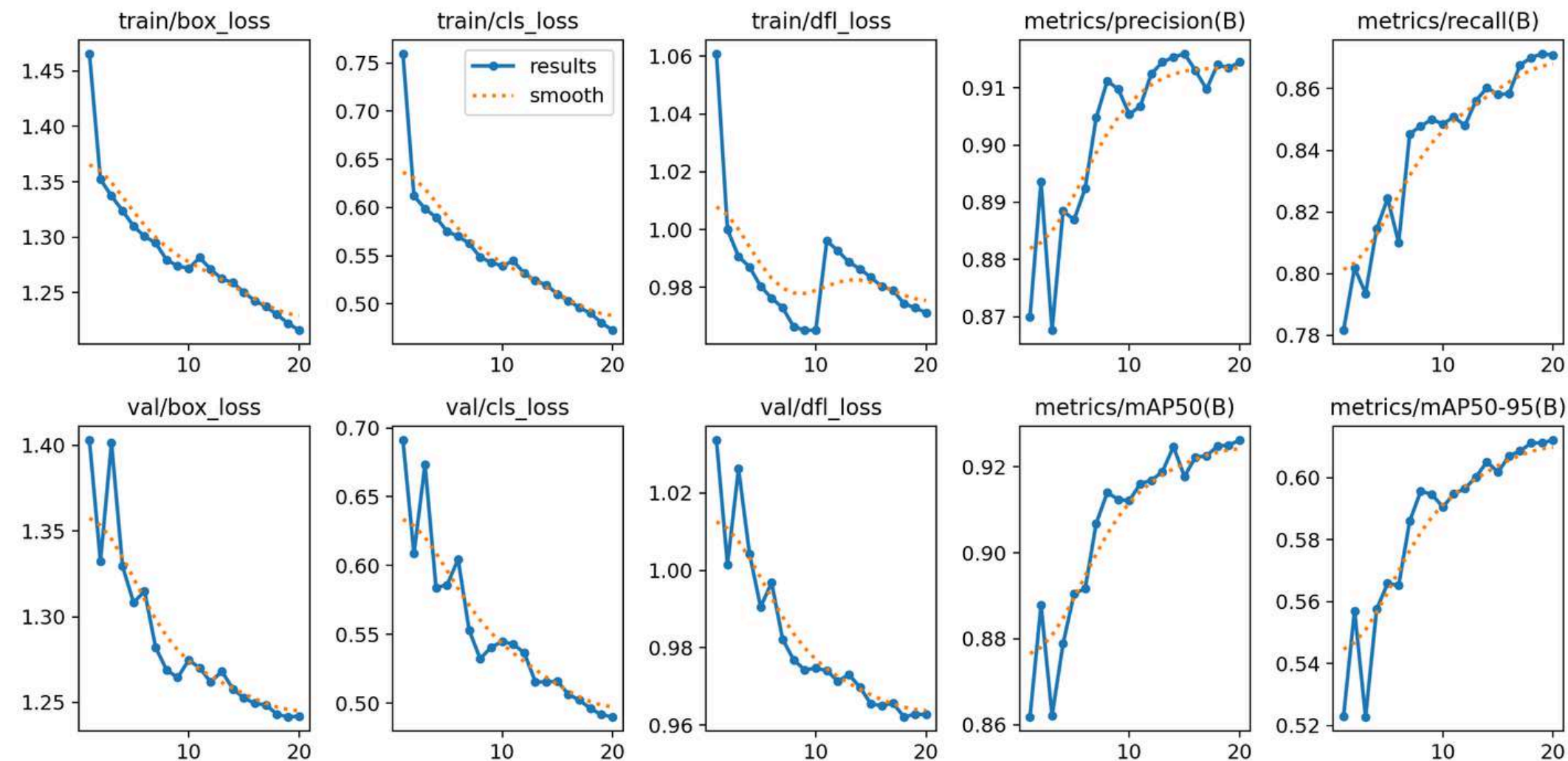
```
20 epochs completed in 1.445 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 147.8MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 147.8MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.39 Python-3.8.18 torch-2.4.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3090, 24260MiB)
YOLOv8x-world summary (fused): 311 layers, 73,660,057 parameters, 0 gradients, 277.1 GFLOPs

```

| | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | |
|---------|-------|--------|-----------|--------|-------|-------|----------|------|
| r 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | 26% |
| er 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | 47% |
| er 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | 58% |
| er 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | 68% |
| | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95 | 100% |
| | all | 588 | 90968 | 0.914 | 0.872 | 0.926 | 0.612 | |

Speed: 0.1ms preprocess, 6.0ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/train2

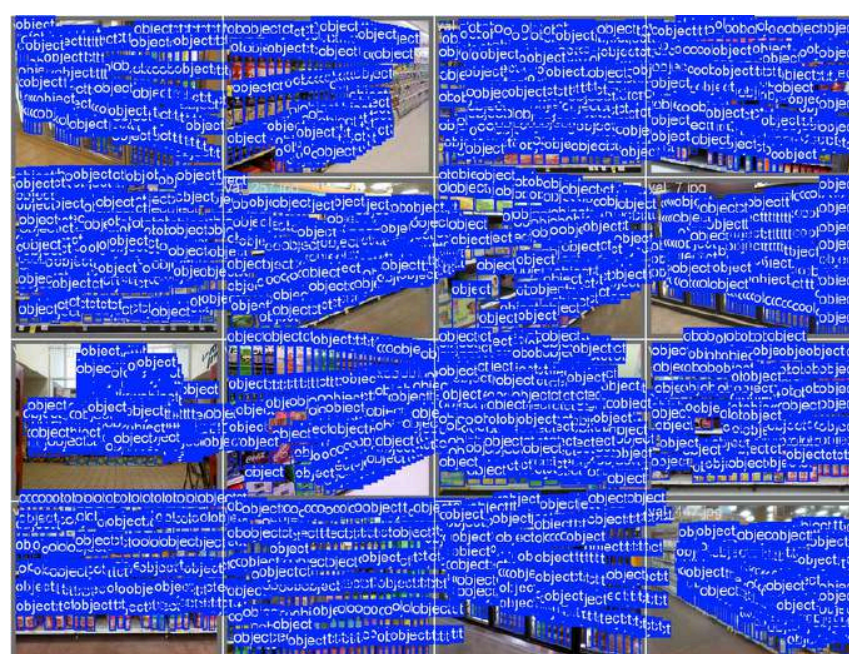


5. Solution (Fine-Tuning)

- Quantitative Evaluation (on SKU-110k validation dataset)

| Metrics | Before Fine-Tuning | After Fine-Tuning |
|-----------|--------------------|-------------------|
| Precision | 0.1921 | 0.914 |
| Recall | 0.0106 | 0.872 |
| mAP@50 | 0.0973 | 0.926 |
| mAP@50-95 | 0.0467 | 0.612 |

- Qualitative Evaluation



GroundTruth



Prediction



GroundTruth



Prediction

5. Solution (Fine-Tuning)

• Fine -Tuning : VisDrone Dataset

Epochs: 20

Batch Size: 16

Image Size: 640 px

Optimizer: Auto-selected (SGD/Adam based on configuration)

Learning Rate:

Initial: 0.01

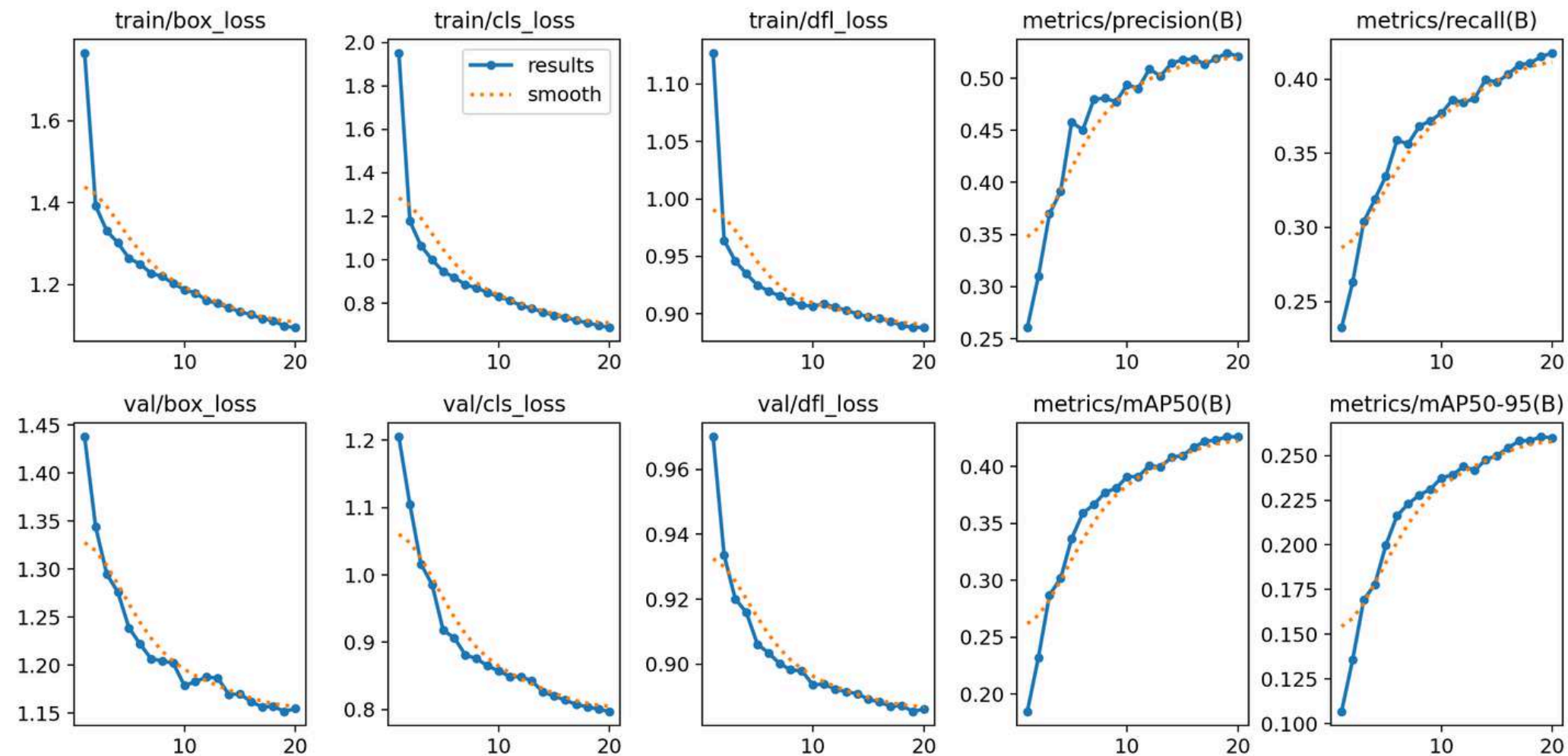
Final: $0.01 \times 0.01 = 0.0001$ (using cosine scheduler)

```
20 epochs completed in 1.175 hours.
Optimizer stripped from runs/detect/train3/weights/last.pt, 147.8MB
Optimizer stripped from runs/detect/train3/weights/best.pt, 147.8MB

Validating runs/detect/train3/weights/best.pt...
Ultralytics 8.3.39 Python-3.8.18 torch-2.4.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3090, 24260MiB)
YOLOv8x-world summary (fused): 311 layers, 73,660,057 parameters, 0 gradients, 283.1 GFLOPs
```

| Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95): 100% |
|-----------------|--------|-----------|--------|-------|-------|-----------------|
| all | 548 | 38759 | 0.527 | 0.414 | 0.426 | 0.261 |
| pedestrian | 520 | 8844 | 0.604 | 0.402 | 0.463 | 0.219 |
| people | 482 | 5125 | 0.609 | 0.266 | 0.35 | 0.137 |
| bicycle | 364 | 1287 | 0.273 | 0.179 | 0.15 | 0.0662 |
| car | 515 | 14064 | 0.763 | 0.774 | 0.813 | 0.59 |
| van | 421 | 1975 | 0.468 | 0.511 | 0.481 | 0.344 |
| truck | 266 | 750 | 0.514 | 0.419 | 0.418 | 0.285 |
| tricycle | 337 | 1045 | 0.496 | 0.328 | 0.325 | 0.184 |
| awning-tricycle | 220 | 532 | 0.274 | 0.242 | 0.176 | 0.116 |
| bus | 131 | 251 | 0.728 | 0.558 | 0.617 | 0.45 |
| motor | 485 | 4886 | 0.542 | 0.456 | 0.467 | 0.213 |

Speed: 0.1ms preprocess, 4.7ms inference, 0.0ms loss, 0.9ms postprocess per image



5. Solution (Fine-Tuning)

- Quantitative Evaluation (on VisDrone Dataset)

| Metrics | Before Fine-Tuning | After Fine-Tuning |
|-----------|--------------------|-------------------|
| Precision | 0.0708 | 0.527 |
| Recall | 0.0793 | 0.414 |
| mAP@50 | 0.0520 | 0.426 |
| mAP@50-95 | 0.0268 | 0.261 |

- Qualitative Evaluation



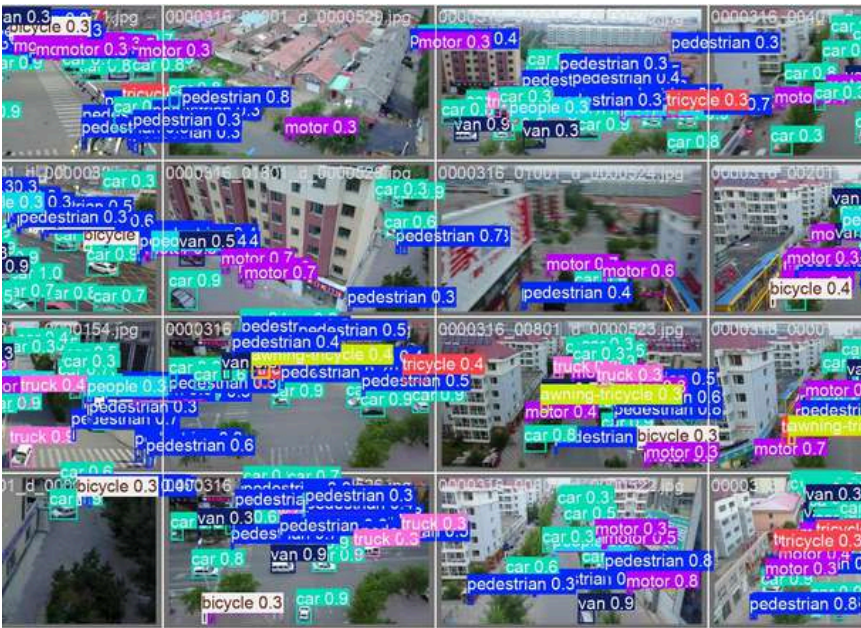
GroundTruth



Prediction



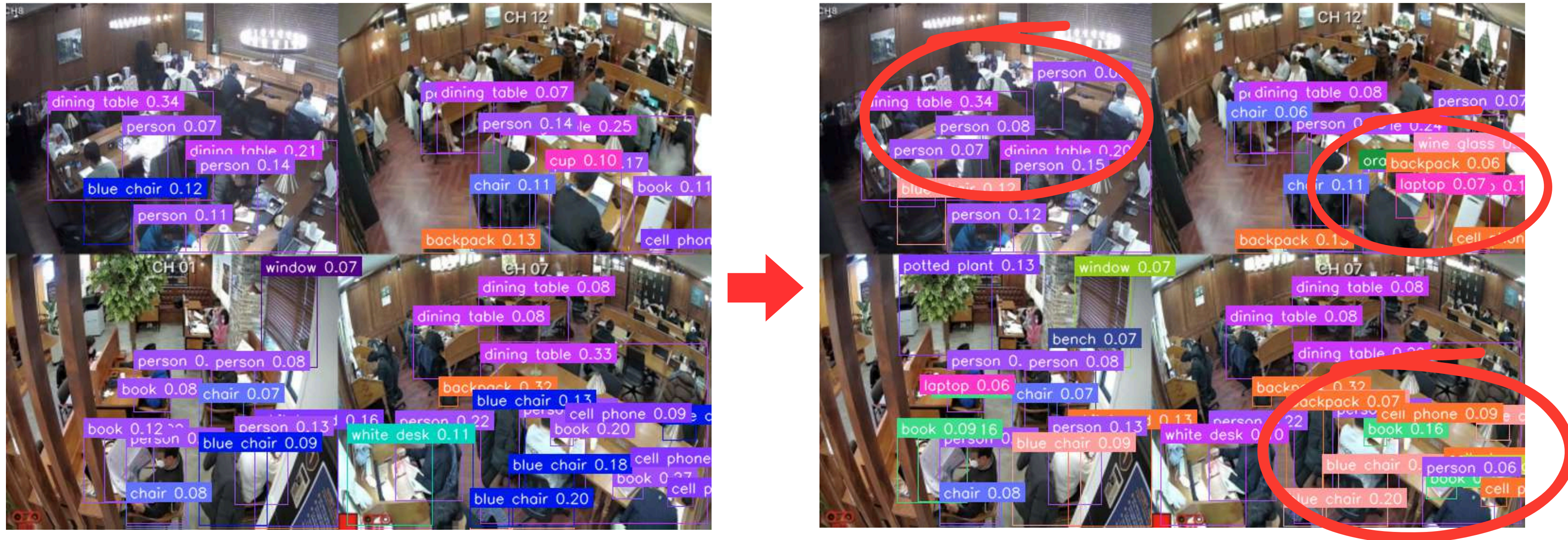
GroundTruth



Prediction

6. Result & Applications

• Final Qualitative Result

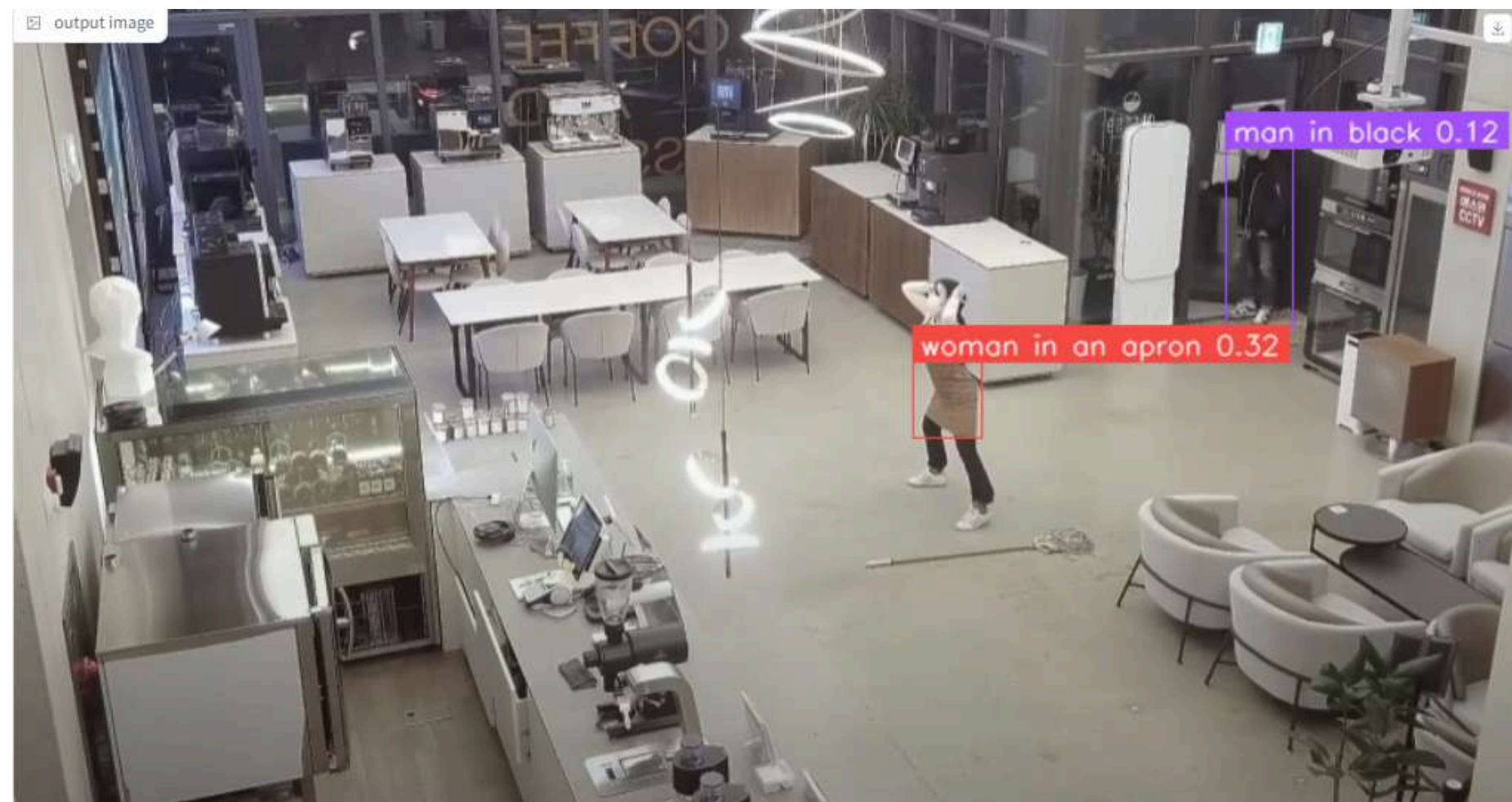
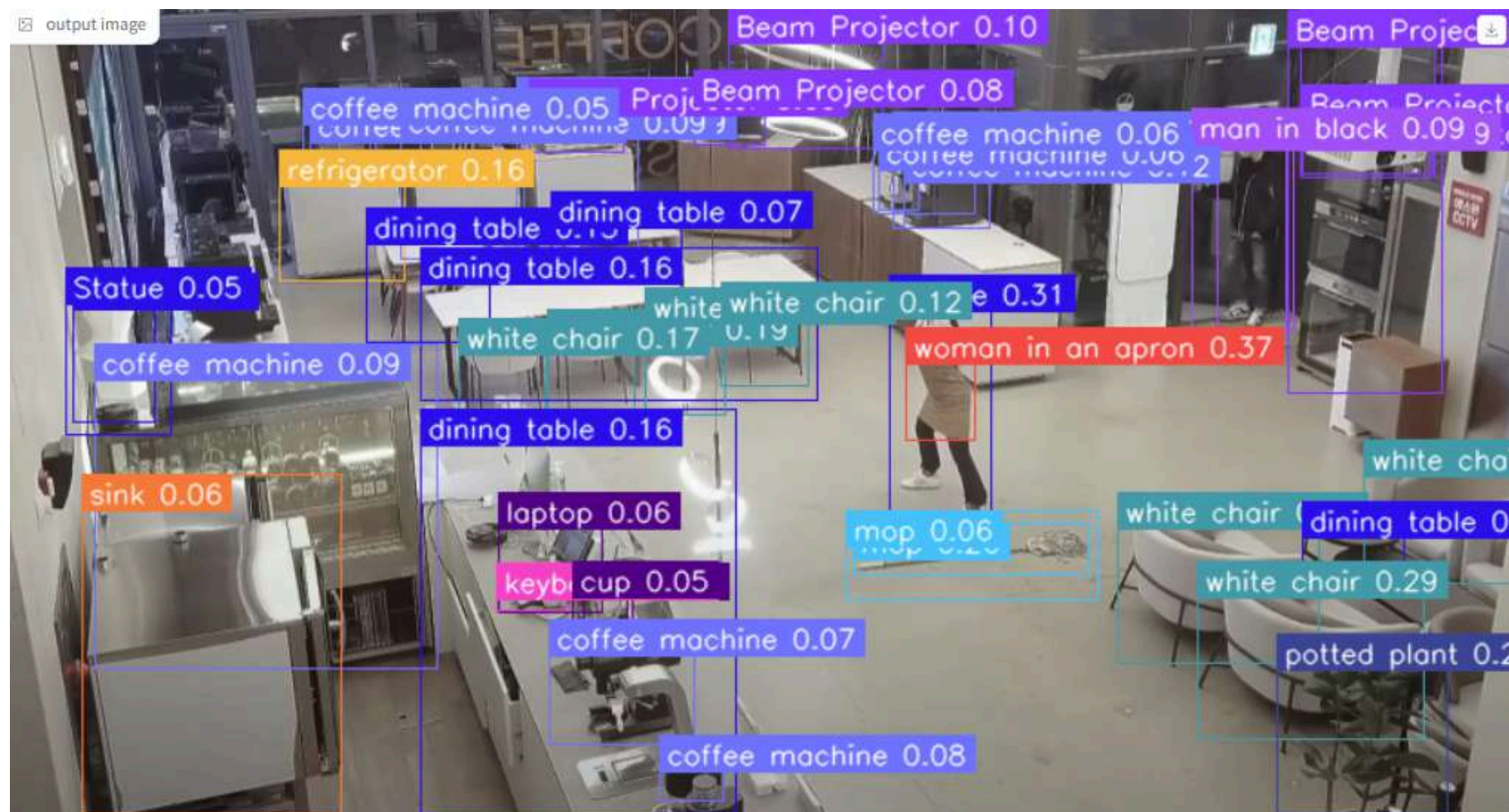


- 작은 객체 검출 및 밀집된 객체 문제를 해결하려는 파인 튜닝 효과 정성적으로 검증

6. Result & Applications

• Applications

Case 1. 사이렌 오더 도난 방지

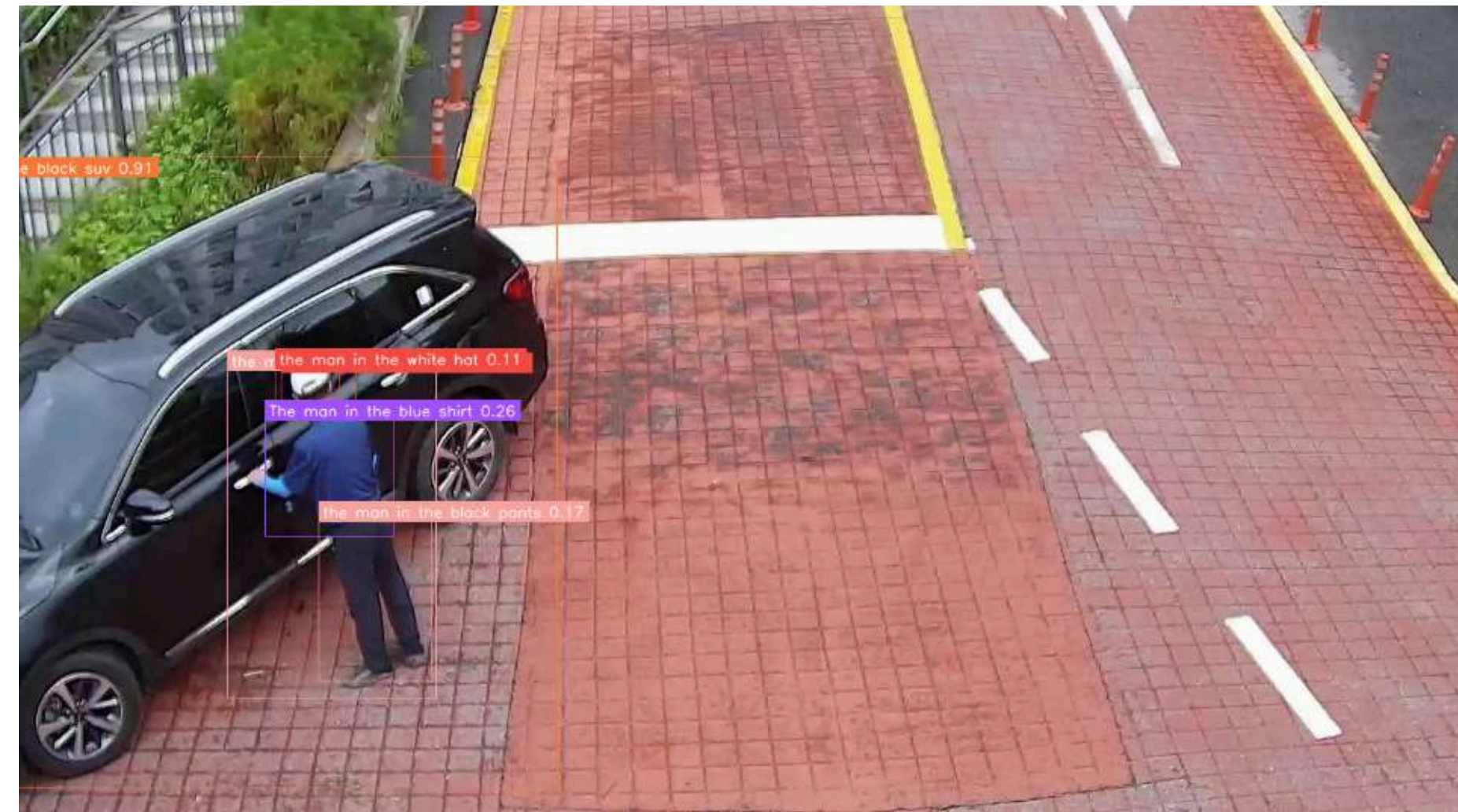


- Open Vocabulary Detection 기술을 활용해 카페 내 모든 객체를 실시간으로 검출할 수 있음
- 사이렌 오더 도난 방지를 위해 고객의 인상착의 기반으로 특정 인물을 탐지 가능

6. Result & Applications

- Applications

Case 2. 용의자 추적, 미아 방지



- 검은색 차량을 타고, 흰색 모자에 파란 셔츠와 검은 옷을 입은 범인을 탐지하는 예시
- 인상착의, 성별, 주변 물체(차량) 등을 고려한 타겟 인물(용의자 or 미아) 탐지 가능

Thank You!

Open Vocabulary Object Detection

- Qualitative Evaluation



GroundTruth



Prediction



GroundTruth



Prediction

0. Index

1. Project Motivation

2. YOLO-World

3. Demo

4. Issue

5. Solution

6. Result & Applications

Open Vocabulary Object Detection

- Quantitative Evaluation (on SKU-110k validation dataset)

```

20 epochs completed in 1.445 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 147.8MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 147.8MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.39 Python-3.8.18 torch-2.4.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3090, 24260MiB)
YOLOv8x-world summary (fused): 311 layers, 73,660,057 parameters, 0 gradients, 277.1 GFLOPs

```

| | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95) | | |
|---------|-------|--------|-----------|--------|-------|-------|-----------|------|--|
| r 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95) | 26% | 5/19 [00:36<01:38, 7.00s/it]Corrupt JPEG data: premature end of data segment |
| | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95) | 47% | 9/19 [00:39<00:20, 2.01s/it]Corrupt JPEG data: 305 extraneous bytes before marker |
| er 0xd9 | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95) | 58% | 11/19 [00:40<00:10, 1.31s/it]Corrupt JPEG data: 786 extraneous bytes before marker |
| | Class | Images | Instances | Box(P) | R | mAP50 | mAP50-95) | 68% | 13/19 [00:42<00:05, 1.00it/s]Corrupt JPEG data: 366 extraneous bytes before marker |
| | all | 588 | 90968 | 0.914 | 0.872 | 0.926 | 0.612 | 100% | 19/19 [00:46<00:00, 2.43s/it] |

Speed: 0.1ms preprocess, 6.0ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/train2

Metrics (Before Fine-Tuning)

Precision: 0.1921

Recall: 0.0106

mAP@50: 0.0973

mAP@50-95: 0.0467

Metrics (After Fine-Tuning)

Precision: 0.914

Recall: 0.872

mAP@50: 0.926

mAP@50-95: 0.612

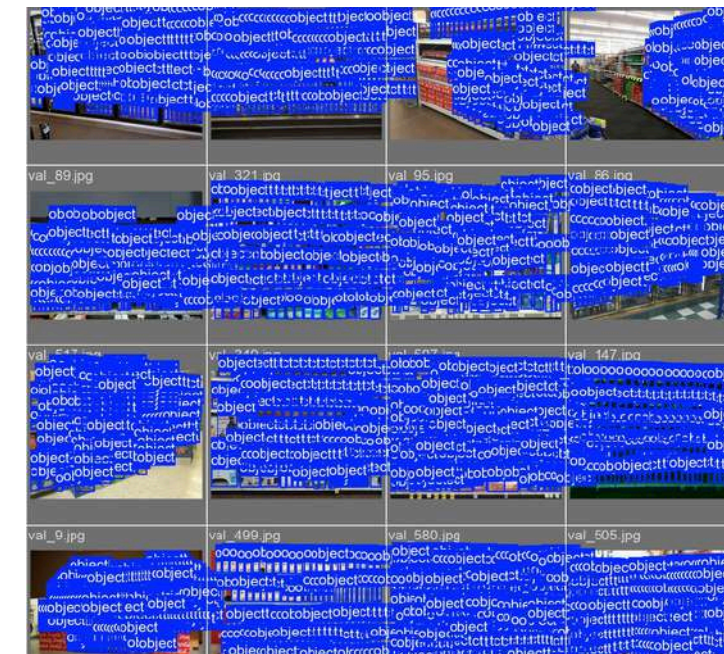
- Qualitative Evaluation



GroundTruth



Prediction



GroundTruth



Prediction