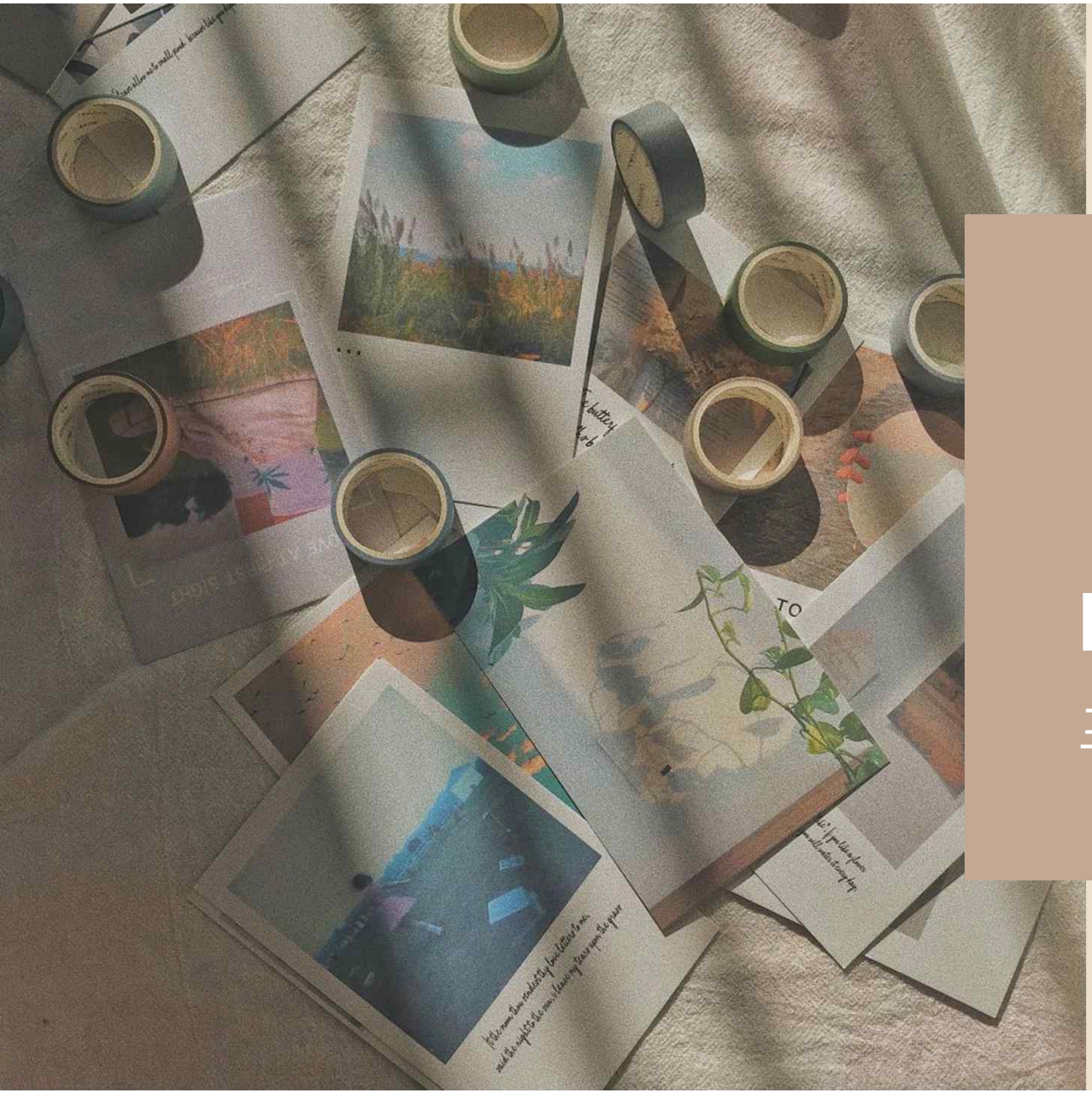
자동 라벨링 프로젝트

한국알파시스템
include <stdio.h>(5조)
김경환, 한세진, 황지원

목차 a table of contents



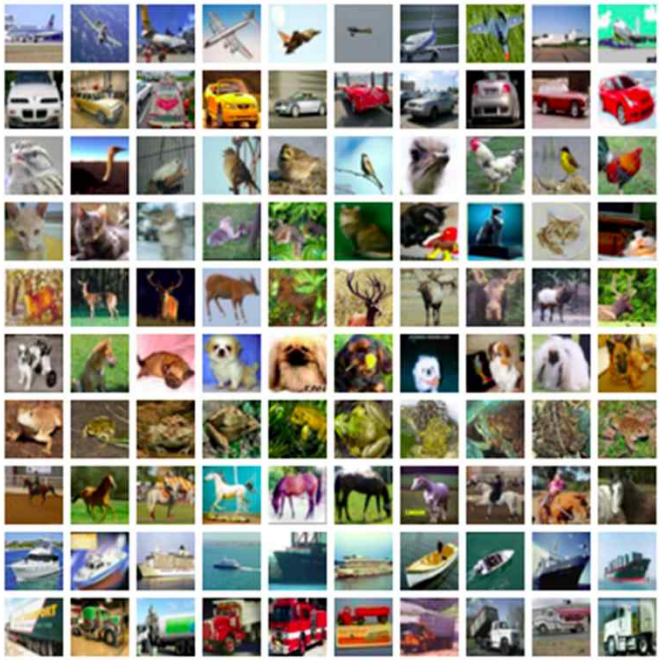
Part 1

프로젝트 개요

Part 1

프로젝트 개요

몇 개의 이미지의 새로운 객체를 학습해서 기존 데이터 셋에서 **새로운 class를 탐지할 수 있는 모델**을 만든다.



Large-scale base dataset

few shot dataset

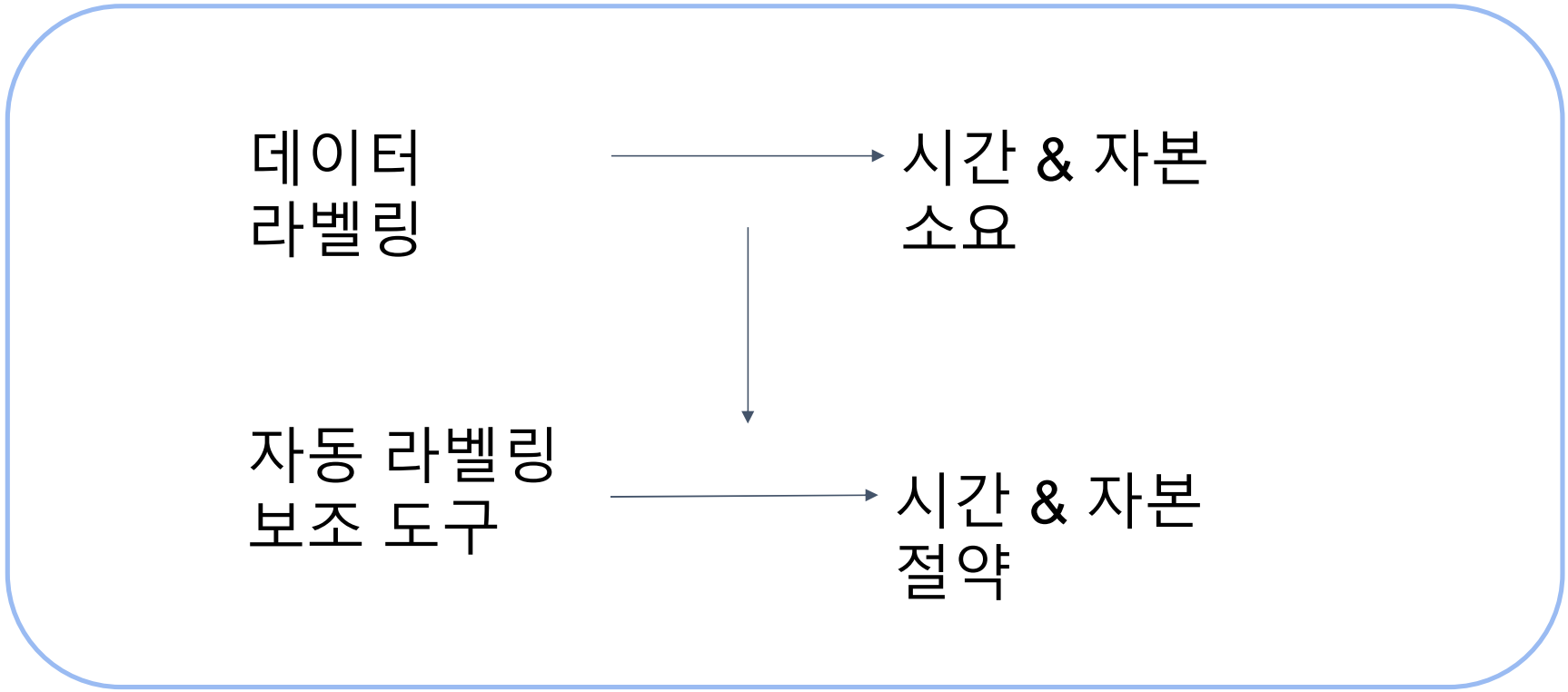
Few-Shot Object Detection(FSOD)

새로운 class에 대해서 visual prompts를 제공함으로써 새로운 class를 탐지할 수 있다.

Part 1

프로젝트 개요

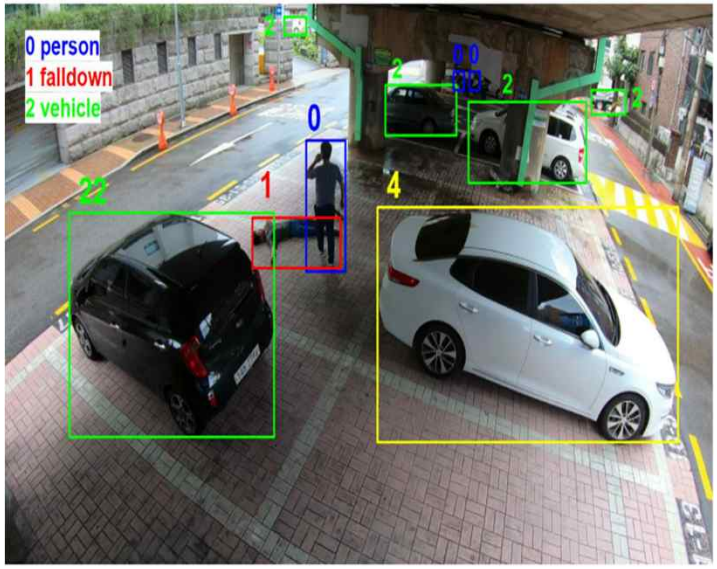
- **Few-Shot Object Detection(FSOD)** 기법을 활용한 이미지 속 **객체 자동 라벨링** 보조 도구 개발



Part 1

프로젝트 배경

- 수동 라벨링의 한계



잘못된 라벨링

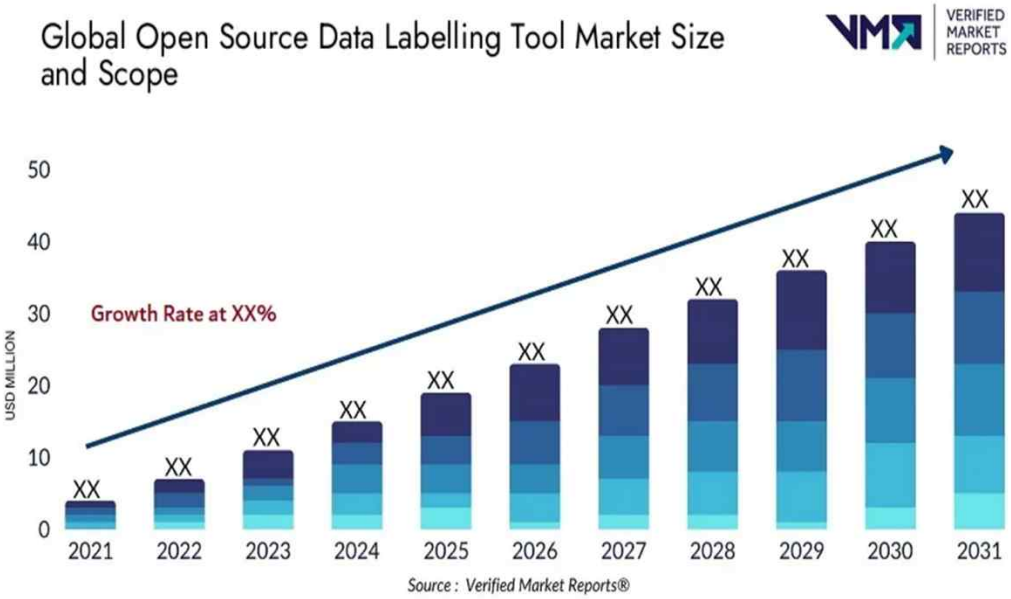
편향된 라벨링

많은 비용 소모

Part 1

프로젝트 배경

- 점점 가치가 높아지는 데이터 라벨링 툴 시장



아시아경제

'인형 눈알 붙이기' 만큼 쉽다…회사 차려 1조원 번 20대 [테크토크]

스케일AI 공동창업자 알렉산더 왕 데이터 라벨링 스타트업 뛰어들어 20대 중반에 개인 자산 '1조' 신화. 인공지능(AI)을 훈련할 때 가장 중요한 요소...

2024. 8. 5.

D 디지털투데이

[테크인사이드] 박사·변호사들이 라벨링을?...AI 데이터셋 경쟁의 세계

[디지털투데이 황치규 기자] AI 학습에 쓰일 수 있도록 데이터에 태그를 달아주는 데이터 라벨링(Data Labeling)을 주특기로 하는 스케일AI가 최근 10...

2024. 7. 2.

Part 1

프로젝트 목표

프로젝트 1차 목표

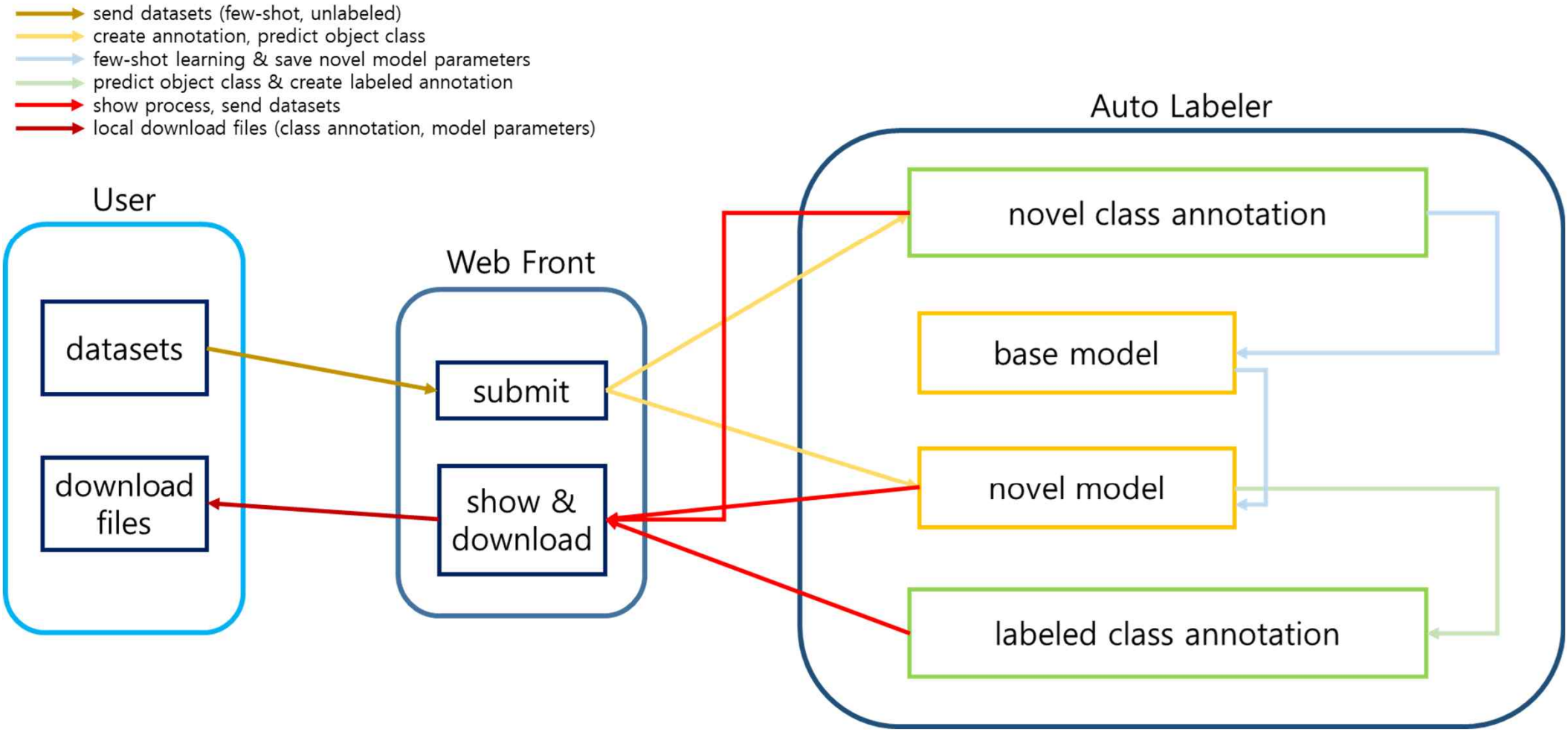
Auto Labeler Model 개발

프로젝트 2차 목표

Web 구축

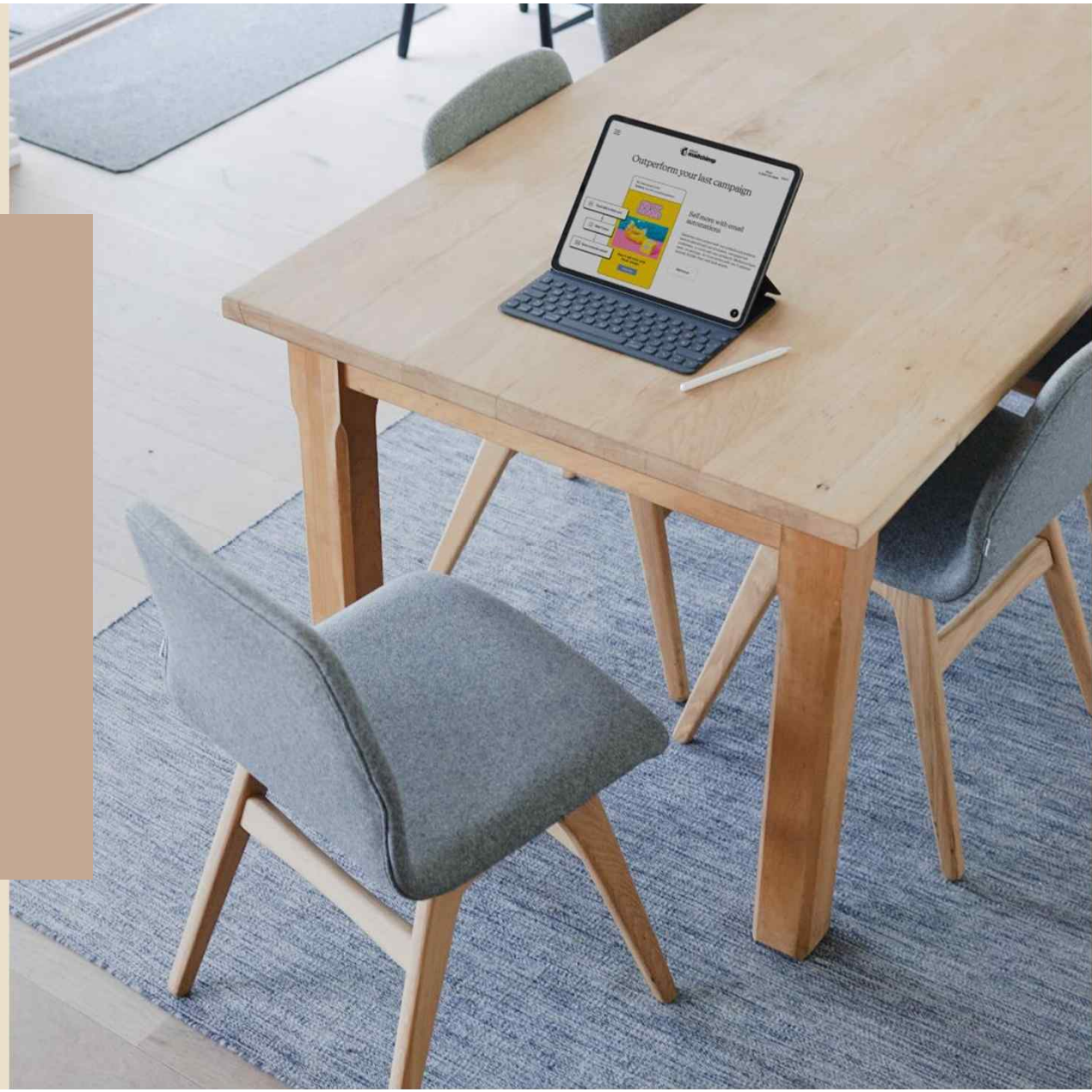
Part 1

시스템 구성도



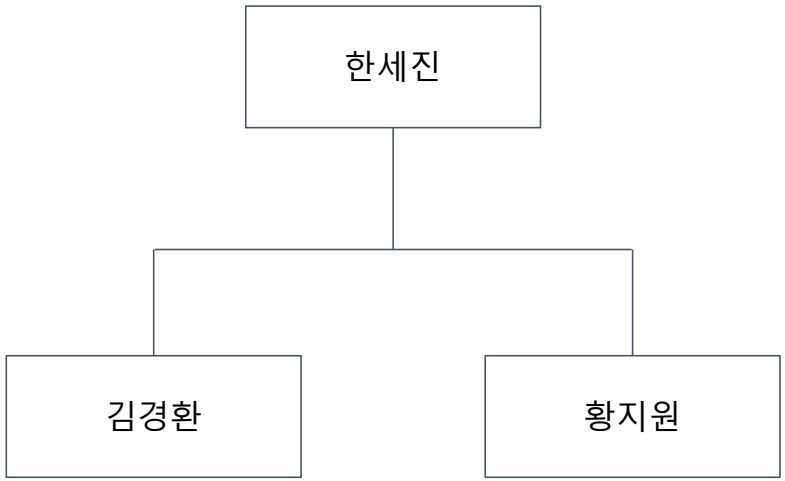
Part 2

프로젝트 팀 구성 및 역할



Part 2

프로젝트 팀 구성 및 역할

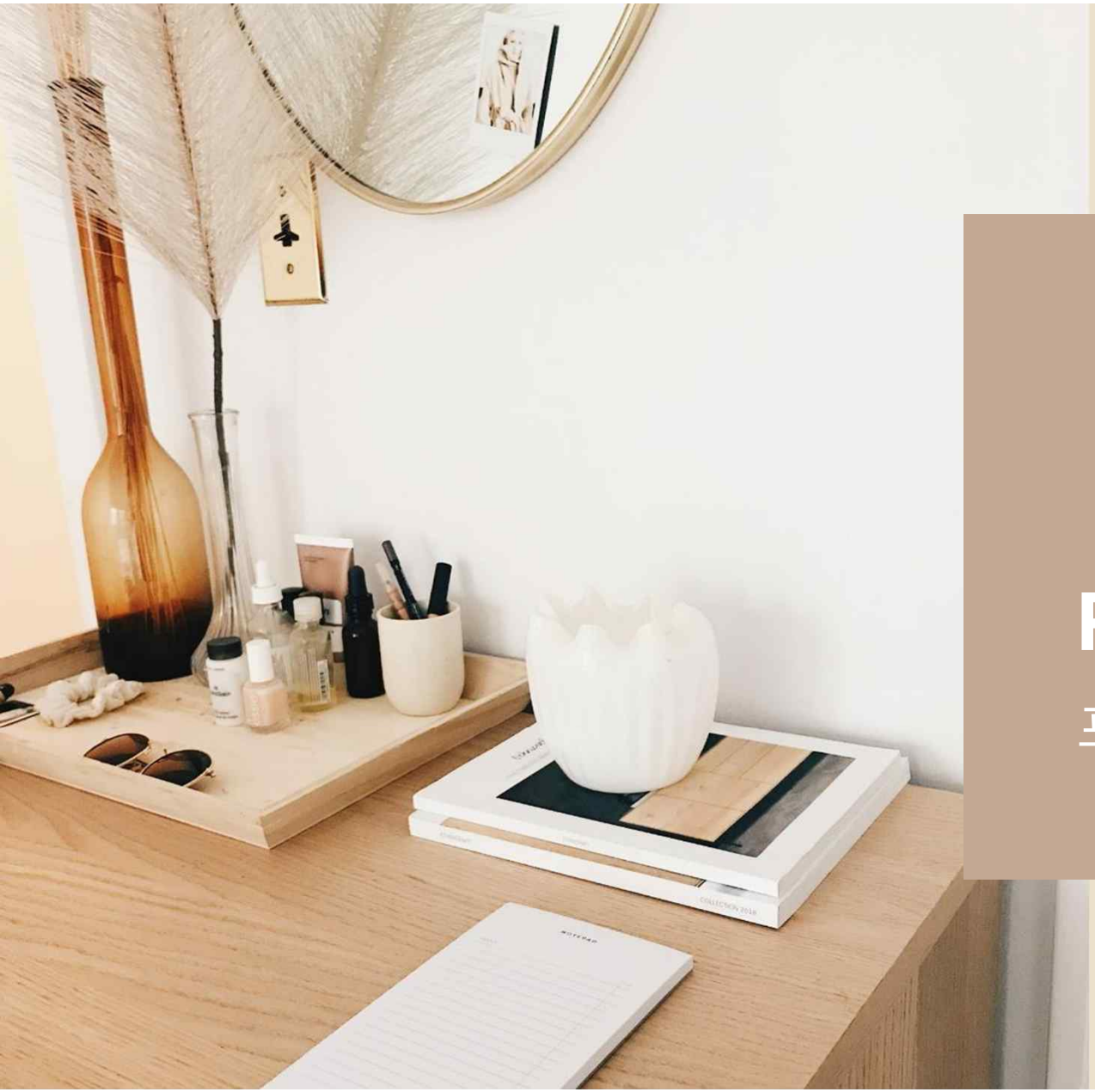


이름	역할	업무
한세진	PM	자료 수집, 논문 리뷰, 모델 개발, 웹 개발, 발표 준비 및 결론 도출
김경환	팀원	자료 수집, 논문 리뷰, 웹 개발, 발표 준비 및 결론 도출
황지원	팀원	자료 수집, 논문 리뷰, 모델 개발, 발표 준비 및 결론 도출

Part 2

프로젝트 진행 일정

단계	테스크	1주차	2주차	3주차	4주차	5주차	6주차	발표	산출물
분석 기획									
	프로젝트 이해 및 범위 설정	→	→						요구사항 정의서
	프로젝트 정의 및 계획 설정	→	→						프로젝트 수행 계획서, WBS
	프로젝트 위험계획 수립	→	→						위험목록 / 위험관리 계획서
논문 리뷰									
	논문 분석		→	→					
	논문 리뷰		→	→					
모델 구현									
	개발 환경 구축		→	→	→				
	모델링			→	→	→	→		모델링 결과 보고서
	모델 평가 및 검증					→	→		모델링 평가 보고서
시스템 구현									
	웹 구현			→	→	→	→		웹 페이지
	설계 및 구현					→	→		구현 시스템
	시스템 테스트					→	→		
평가 및 전개									
	모델 발전 계획 수립						→	→	발전 계획서
	프로젝트 평가 보고						→	→	결과 보고서

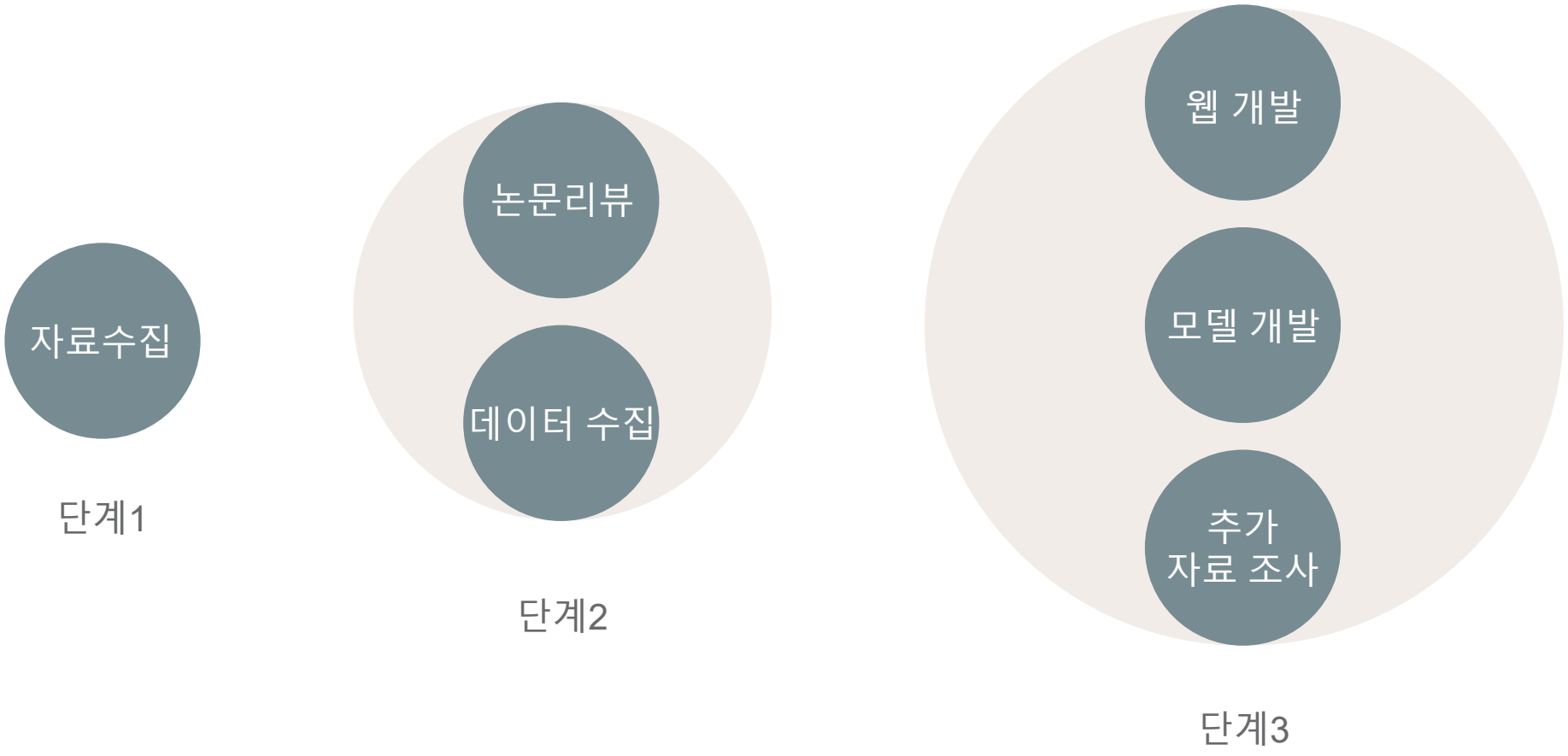


Part 3

프로젝트 수행 절차 및 방법

Part 3

프로젝트 진행 절차



Part 3

참고 논문

Label, Verify, Correct: A Simple Few Shot Object Detection Method

Prannay Kaul[1] Weidi Xie[1,2] Andrew Zisserman[1]
[1]Visual Geometry Group, University of Oxford [2]Shanghai Jiao Tong University
http://www.robots.ox.ac.uk/~vgg/research/lvc/

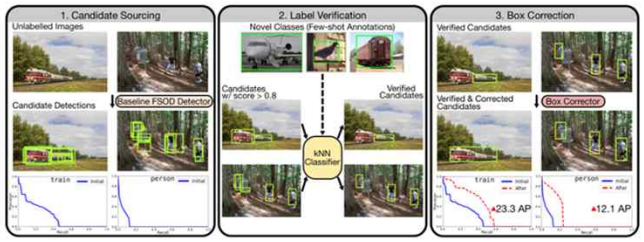


Figure 1. **Label, Verify, Correct:** Object detectors naïvely trained with K-shot annotations perform poorly on novel classes (bottom left). We propose to expand the novel class annotation set and re-train *end-to-end*. 1. Given a baseline few-shot object detector, noisy candidate detections are sourced from unlabelled images (left). 2. Labels for each candidate detection are verified by a kNN classifier, constructed from a self-supervised model using the same few-shot annotations, removing large numbers of false positives (centre). 3. A specialised box corrector, drastically improves the remaining bounding boxes, yielding high-quality pseudo-annotations (right). Re-training end-to-end with our pseudo-labelling method yields a large performance boost on novel class detection, improving precision and recall (bottom right).

Abstract

The objective of this paper is few-shot object detection (FSOD) – the task of expanding an object detector for a new category given only a few instances for training. We introduce a simple pseudo-labelling method to source high-quality pseudo-annotations from the training set, for each new category, vastly increasing the number of training instances and reducing class imbalance; our method finds previously unlabelled instances.

Naïvely training with model predictions yields sub-optimal performance; we present two novel methods to improve the precision of the pseudo-labelling process: first, we introduce a verification technique to remove candidate detections with incorrect class labels; second, we train a specialised model to correct poor quality bounding boxes. After these two novel steps, we obtain a large set of high-quality pseudo-annotations that allow our final detector to be trained end-to-end. Additionally, we demonstrate our method maintains base class performance, and the utility of simple augmentations in FSOD. While benchmarking on PASCAL VOC and MS-COCO, our method achieves state-of-the-art or second-best performance compared to existing approaches across all number of shots.

1. Introduction

Object detection refers to the task of determining if an image contains objects of a particular category, and if so, then localising them. In recent years, the community has seen tremendous successes in object detection by training computational models for a set of pre-defined object classes [8, 16, 32, 36, 40, 46, 55], with large numbers of human annotated labels, *e.g.* MS-COCO [30], and PASCAL VOC [11]. However, such training paradigms have limited the model to only perform well on a closed, small set of categories for which large training data is available.

In contrast, humans can continuously expand their vocabularies, learning to detect a much larger set of categories, even with access to only a few examples [43]. This is also a desirable ability for modern computer vision systems and is studied in the task of few-shot object detection (FSOD) [13, 21, 45, 51, 56]. The goal of our work is FSOD: given an existing object detector that has been

[논문 내용 요약]

1. 연구 목적

- Few-shot Object Detection (FSOD) 성능 개선

- 적은 수의 학습 데이터로 새로운 클래스의 객체를 탐지하는 방법 제안

2. 주요 성과

- MS-COCO와 PASCAL VOC 벤치마크에서 SOTA 또는 차상위 성능 달성

- 기존 클래스에 대한 성능을 유지하면서 새로운 클래스 탐지 성능 향상

- 30-shot MS-COCO 테스트에서 nAP 기준 25.5% 달성

3. 기술적 특징

- 자기지도학습(self-supervised) 모델을 활용한 검증 단계 도입

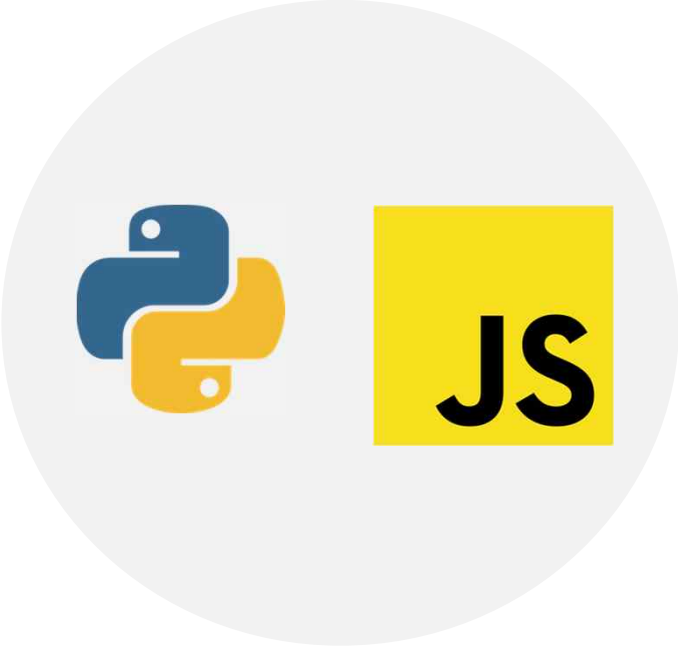
- 데이터 증강 기법의 효과적인 활용

- End-to-end 재학습을 통한 성능 개선

Part 3

관련 기술 사항

Language



Library

Tools

Part 3

프로젝트 변화 (OS 변경)

Window 환경에서
모델 개발

↓

Linux 환경에서
모델 개발

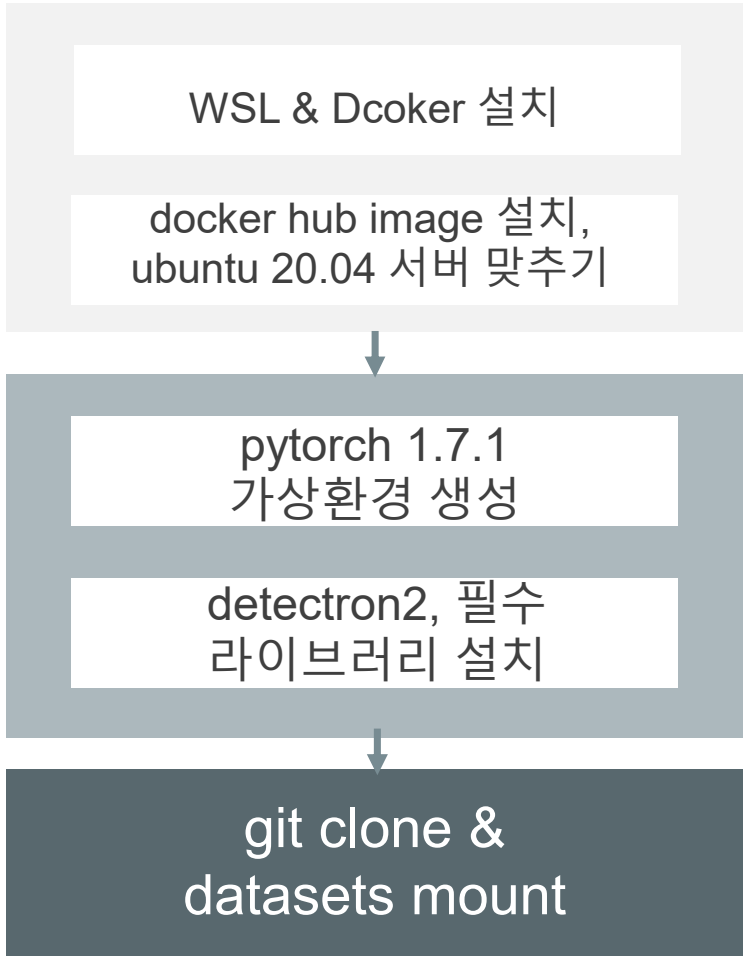
Part 3

사용한 모델

Main Model	Faster R-CNN	YOLO_v5
Backbone Model	ResNet-50	CSP-Darknet53
Label Classifier	k-NN Classifier (k = 30, tuning)	YOLO_v3 head
RPN (Region Proposal Network)	2 Convolution Layers	X

Part 3

모델 환경 구축 절차



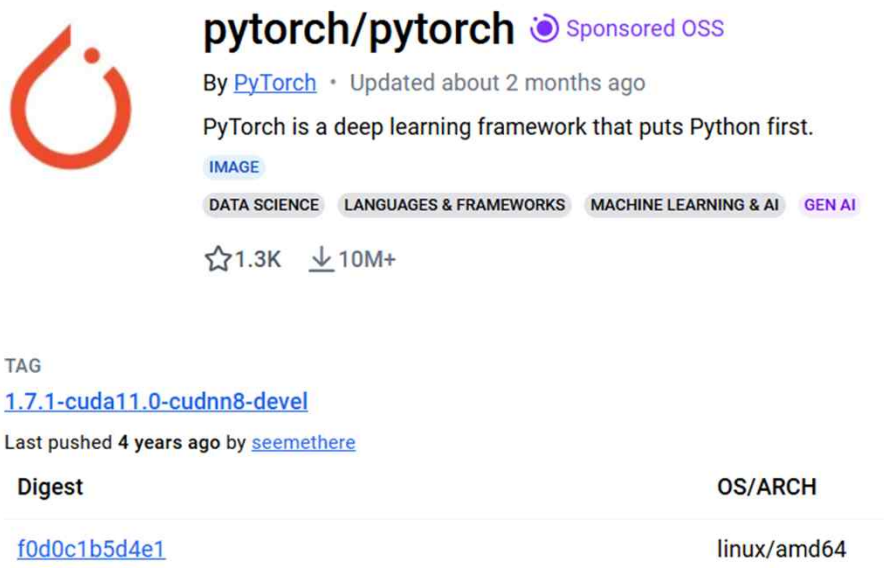
OS 변경으로 인한 환경 재구축

크게 두가지의 과정으로 진행

1. WSL과 Docker Engine을 구축
2. 가상환경 생성과 필수 라이브러리 설치

Part 3

모델 환경 구축



Docker Hub Image

- pytorch/pytorch:1.7.1-cuda11.0-cudnn8-devel

pytorch 버전 1.7.1 버전에 맞춰 갖춰진 docker container image를 사용

python, anaconda, cuda 등등의 설정된 환경에서 ubuntu 버전만 18.04에서 20.04로 맞춤

git, wget, nano, cifs 등등 필요한 리눅스 라이브러리를 추가적으로 설치

Part 3

모델 환경 구축

```
conda create -n lvc python=3.8

conda activate lvc

conda install pytorch=1.7.1 torchvision=0.8.2 \
     torchaudio=0.7.2 cudatoolkit=11.0 -c pytorch
```

```
python -m pip install -e .

python -m pip install -r requirements.txt
```

Conda 가상환경 생성
- docker container 사용

‘instal -e .’ 를 이용하여 custom detectron2를 적용함
‘install -r requiments.txt’로 필수 라이브러리들을 설치.

그외의 필요한 라이브러리도 추가

Part 3

모델 학습

```
python -m tools.train_net \
     --config-file configs/COCO-detection/faster_rcnn_R_50_FPN_base.yaml \
     --num-gpus 4

python -m tools.train_net \
     --config-file configs/COCO-detection/faster_rcnn_R_50_FPN_base.yaml \
     --num-gpus 1
```

환경에 맞춰서 명령어 및 설정 파일 최적화

사용할 수 있는 gpu 수에 맞춰 1개로 수정

또한 ‘cuda out of memory’를 해결하기 위해서 *.yaml 파일에서 batch_size를 2로 줄이고 base learning rate를 0.01에서 0.001로 수정해서 모델 학습을 진행

Part 3

모델 학습

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.095
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.223
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.068
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.056
[12/02 04:03:37 lvc.evaluation.coco_evaluation]: Per-category bbox AP50:
| category       | AP50   | category       | AP50   | category      | AP50   |
|:---------------|:-------|:---------------|:-------|:--------------|:-------|
| truck          | 15.286 | traffic light  | 24.348 | fire hydrant  | 41.787 |
| stop sign      | 49.458 | parking meter  | 6.421  | bench         | 6.261  |
| elephant       | 57.566 | bear           | 45.573 | zebra         | 63.290 |
| giraffe        | 59.431 | backpack       | 10.271 | umbrella      | 25.424 |
| handbag        | 4.470  | tie            | 14.566 | suitcase      | 13.990 |
| frisbee        | 32.295 | skis           | 9.356  | snowboard     | 2.562  |
| sports ball    | 47.134 | kite           | 32.084 | baseball bat  | 11.676 |
| baseball glove | 22.633 | skateboard     | 15.212 | surfboard     | 15.564 |
| tennis racket  | 46.184 | wine glass     | 20.670 | cup           | 29.141 |
| fork           | 5.849  | knife          | 2.743  | spoon         | 1.265  |
| bowl           | 32.701 | banana         | 18.836 | apple         | 12.012 |
| sandwich       | 23.955 | orange         | 10.670 | broccoli      | 17.641 |
| carrot         | 11.034 | hot dog        | 7.478  | pizza         | 45.201 |
| donut          | 23.187 | cake           | 24.182 | bed           | 26.553 |
| toilet         | 51.819 | laptop         | 29.866 | mouse         | 35.793 |
| remote         | 10.775 | keyboard       | 24.761 | cell phone    | 17.722 |
| microwave      | 15.896 | oven           | 10.726 | toaster       | 0.000  |
| sink           | 26.254 | refrigerator   | 11.509 | book          | 10.450 |
| clock          | 54.145 | vase           | 22.575 | scissors      | 0.000  |
| teddy bear     | 28.771 | hair drier     | 0.000  | toothbrush    | 0.000  |
```

```
| toilet         | 79.330 | laptop         | 66.949 | mouse         | 71.579 |
| remote         | 37.759 | keyboard       | 68.376 | cell phone    | 47.079 |
| microwave      | 47.727 | oven           | 45.070 | toaster       | 0.000  |
| sink           | 58.768 | refrigerator   | 44.860 | book          | 51.988 |
| clock          | 69.655 | vase           | 54.857 | scissors      | 0.000  |
| teddy bear     | 76.891 | hair drier     | 0.000  | toothbrush    | 0.000  |
```

Base, Novel, Combine model learning - 전체, category 성능지표 도출

학습을 진행하면 Base, Novel, Combine model 각각의 성능이 계산됨

전체 AP값을 보여줌. 이어서 category별 recall, AP50을 보여줌

왼쪽의 사진들은 Base Class에 대해서만 학습을 진행했을 때의 결과

Part 3

모델 학습 실패

```
[12/03 11:36:59 lvc.evaluation.evaluator]: Inference done 117150/117264. 0.1340 s / img. ETA=0:00:15
[12/03 11:37:05 lvc.evaluation.evaluator]: Inference done 117200/117264. 0.1340 s / img. ETA=0:00:08
[12/03 11:37:12 lvc.evaluation.evaluator]: Inference done 117250/117264. 0.1340 s / img. ETA=0:00:01
[12/03 11:37:14 lvc.evaluation.evaluator]: Total inference time: 4:21:52 (0.133994 s / img per device, on 1
devices)
[12/03 11:37:14 lvc.evaluation.evaluator]: Total inference pure compute time: 4:14:52 (0.130415 s / img per
device, on 1 devices)
Killed
root@docker-desktop:/workspace/lvc#
```

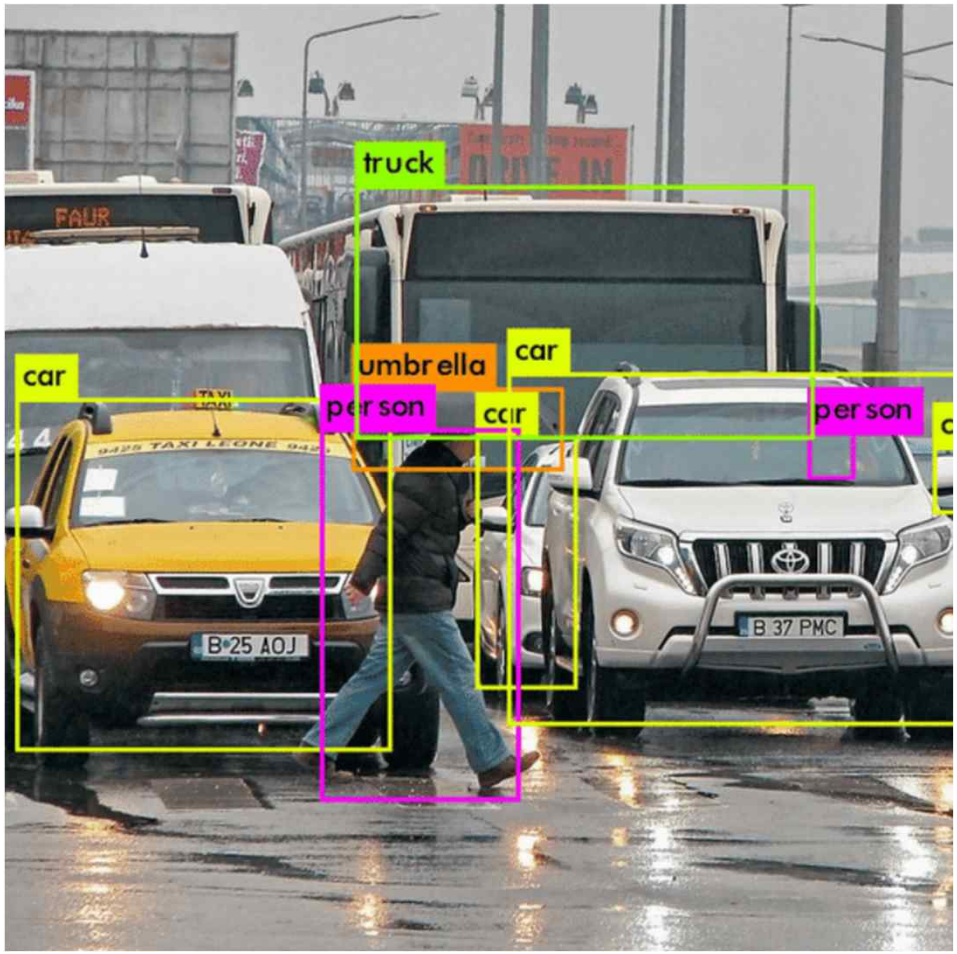
RPN 학습시도 실패

RPN을 학습하기 위해서 학습된 모델에서 bbox 정보를 추출하던 과정에서 docker engine이 멈추는 상황 반복

따라서 최종 모델이 나오지는 못했지만 웹에서 YOLO 모델로 구현

Part 3

YOLO Model



YOLO_v5

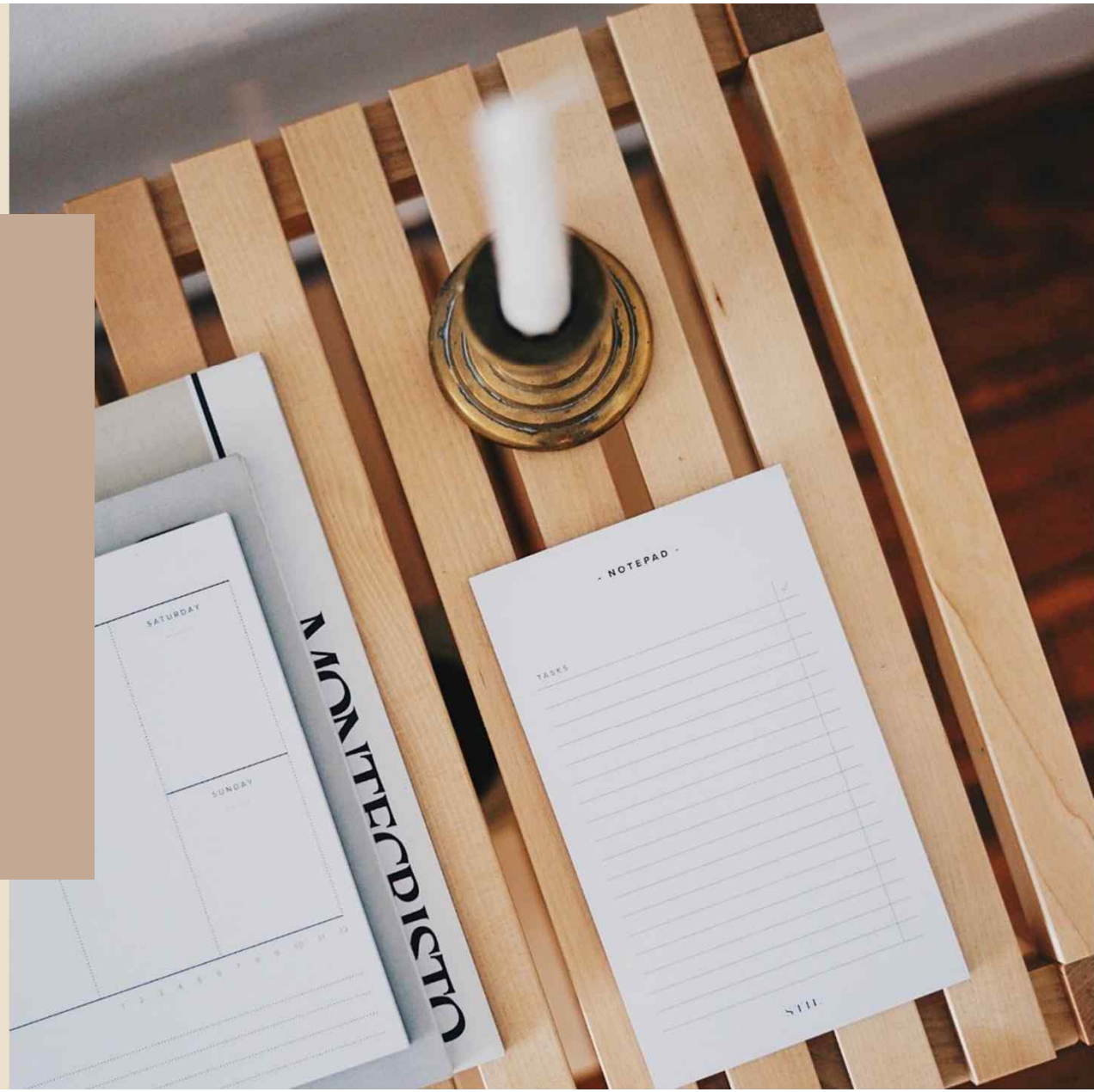
왜 YOLO_v5를 사용하게 되었는가?

결정 사유

1. 새로운 class 학습 & 적은 학습시간 소요
2. 웹 서버 부하 방지를 위한 가벼운 모델
3. 사용자가 다루기 쉬운 모델

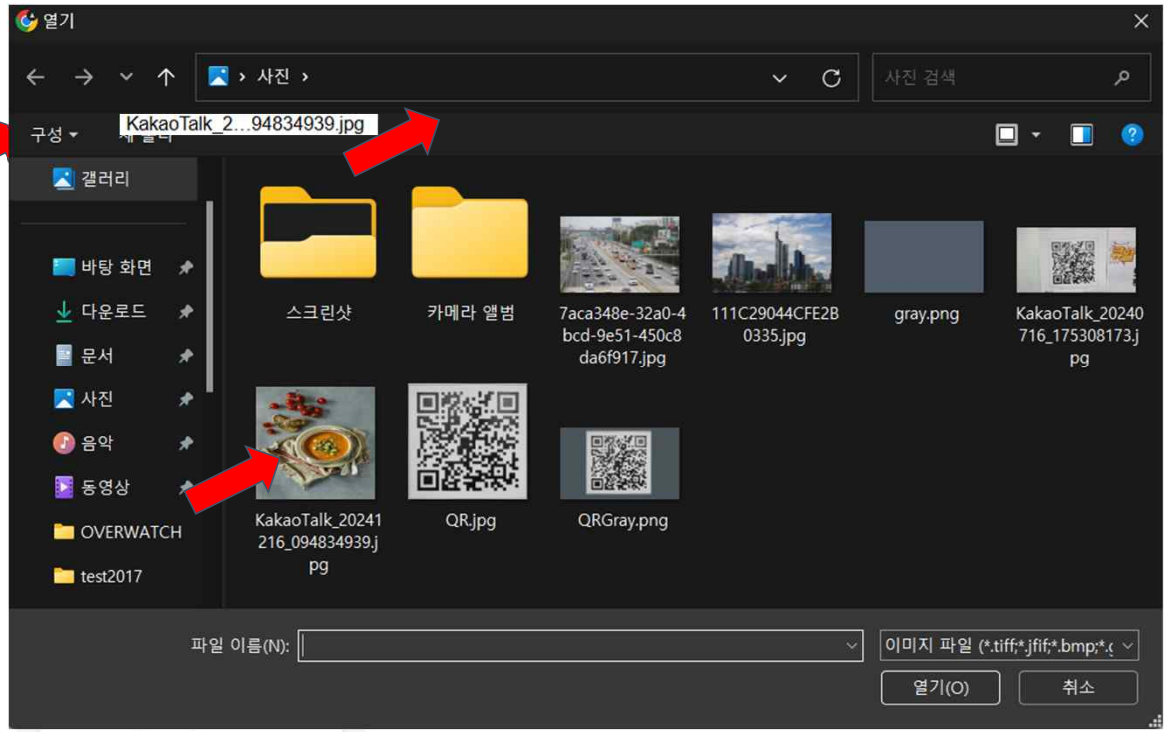
Part 4

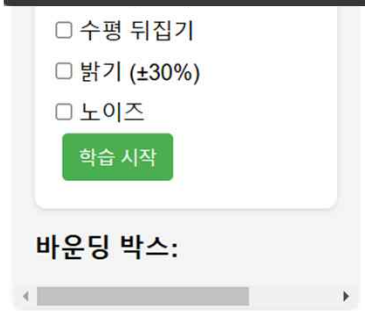
프로젝트 수행 결과



Part 4

웹 구현





Part 4

웹 구현

객체 감지 및 수동 라벨링

파일 선택 KakaoTalk_2...94834939.jpg 업로드 및 감지

업로드된 이미지:
KakaoTalk_20241216_09483493

보기/편집 ×

감지 도구
기본 모델 (YOLOv5)
객체 감지

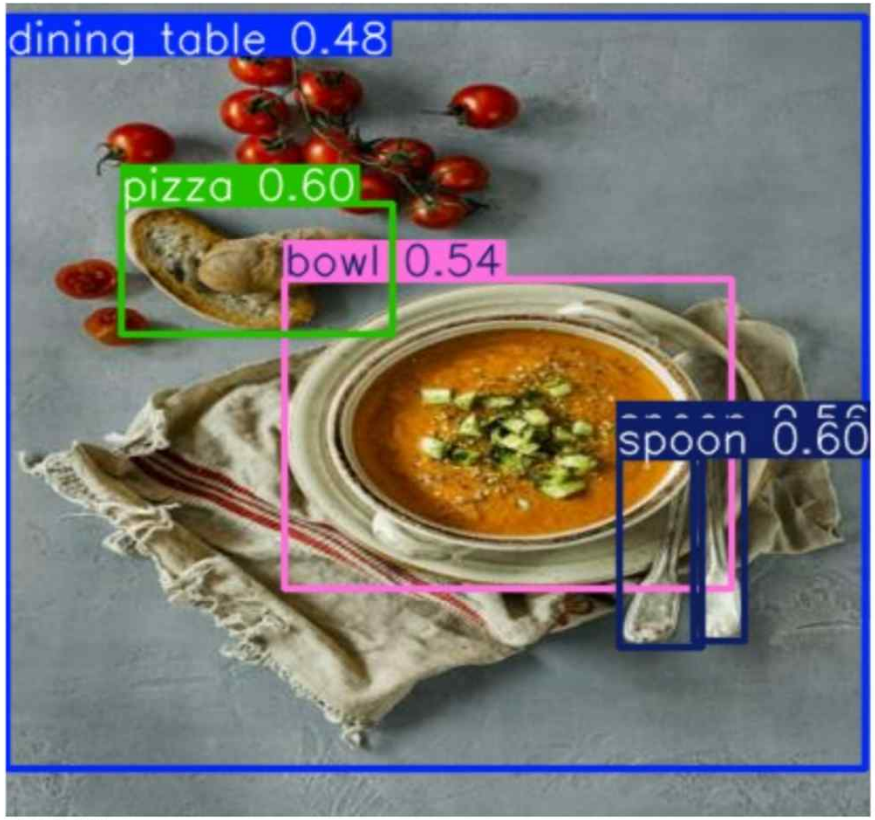
라벨링 도구
클래스 이름 입력
라벨 저장
마지막 라벨 취소

학습 도구
데이터 증강:
- ☐ 회전 (±30°)
- ☐ 수평 뒤집기
- ☐ 밝기 (±30%)
- ☐ 노이즈

학습 시작

바운딩 박스:

원본 이미지



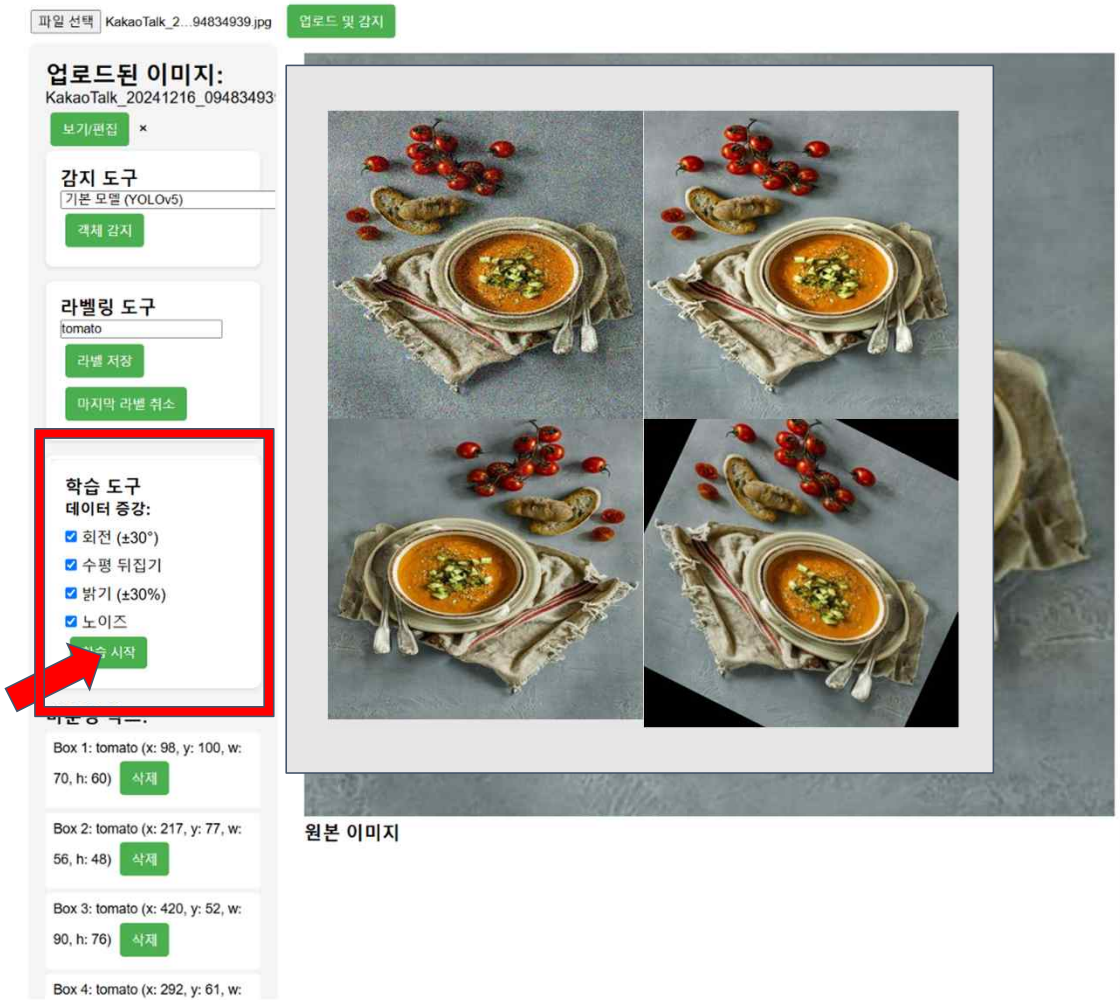
감지된 객체

Point

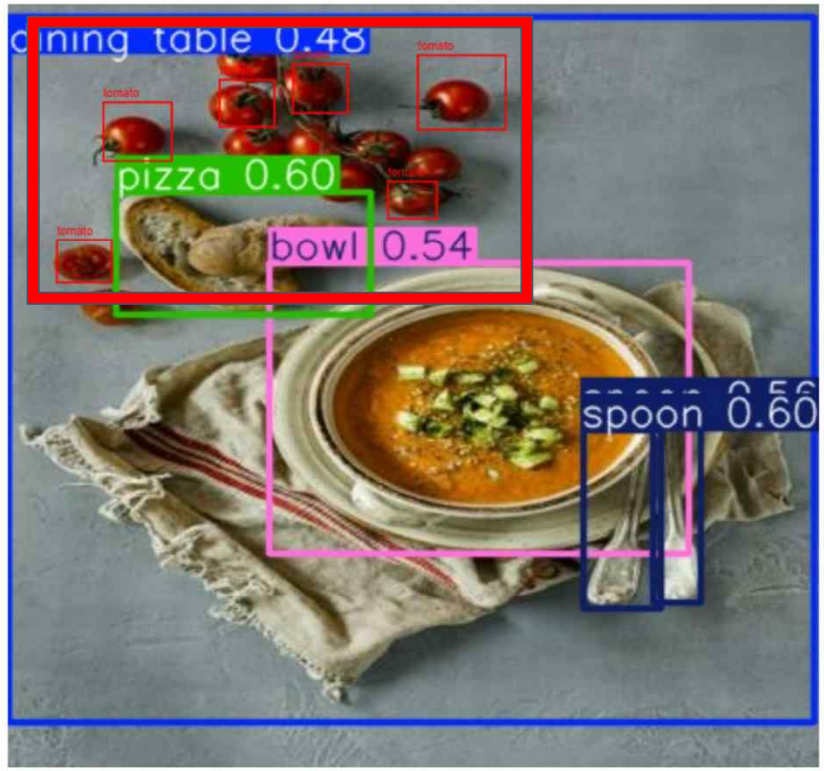
Part 4

웹 구현

객체 감지 및 수동 라벨링



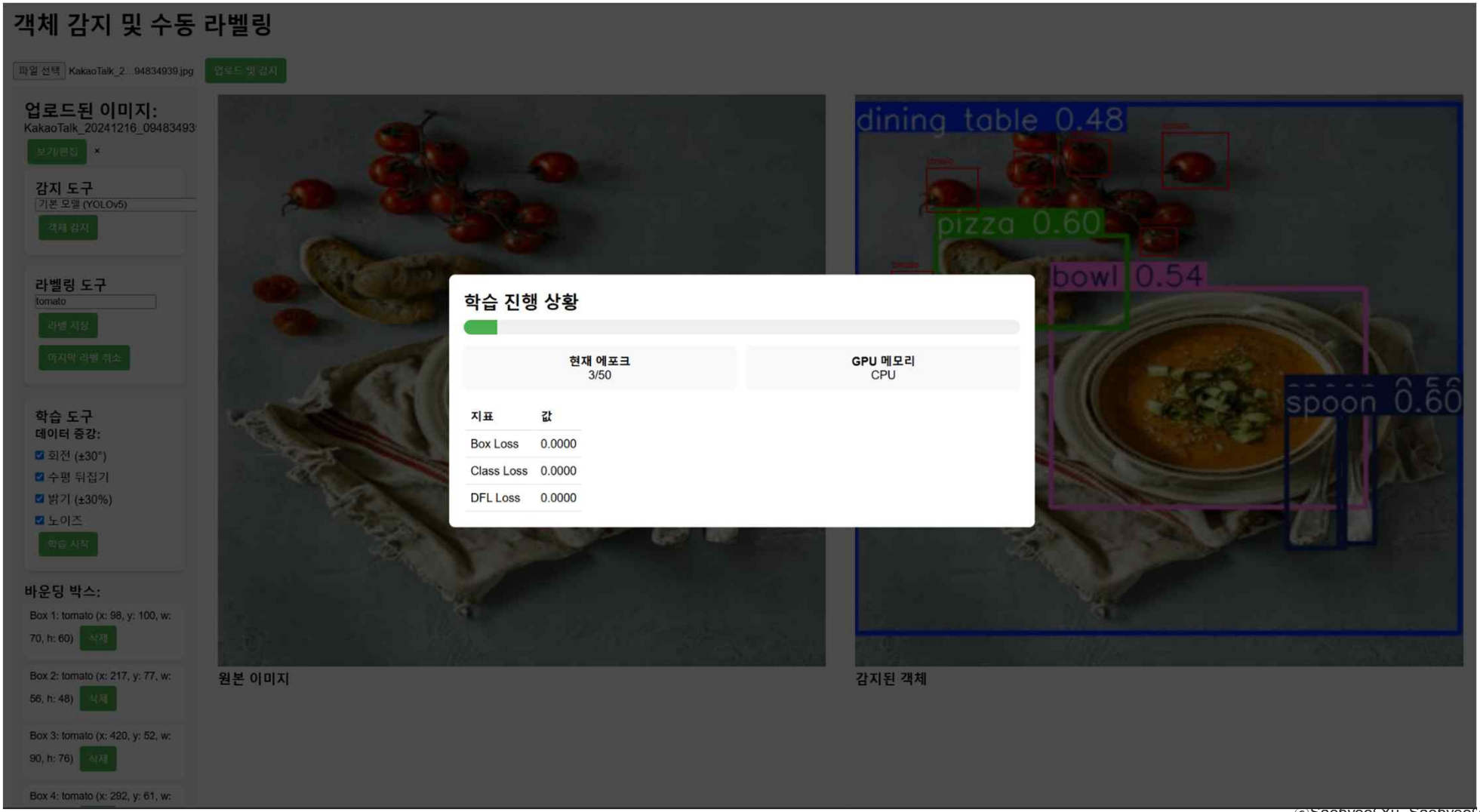
원본 이미지



감지된 객체

Part 4

웹 구현

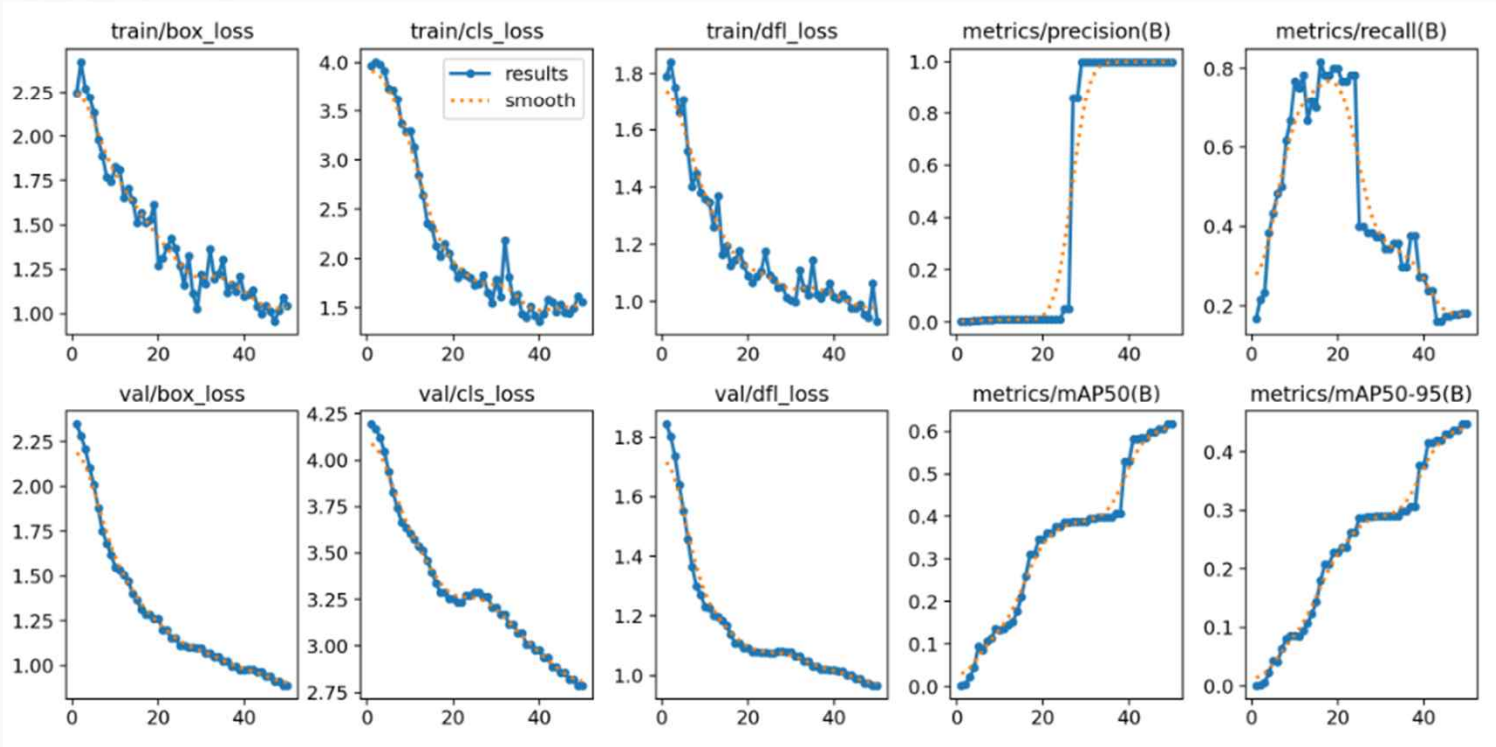


Part 4

학습 결과

결과 창 닫기

학습 결과 그래프



모델 요약


```
{
  "parameters": 3011238,
  "gradients": 3011222,
  "layers": 225,
  "gflops": 0
}
```

Part 4

웹 구현

객체 감지 및 수동 라벨링

파일 선택 111C29044CFE2B0335.jpg 업로드 및 감지

업로드된 이미지:
111C29044CFE2B0335.jpg

보기/편집 ×

감지 도구

학습된 모델: custom_model_202412

객체 감지

라벨링 도구

클래스 이름 입력

라벨 저장

마지막 라벨 취소

학습 도구

데이터 증강:

- ☐ 회전 (±30°)
- ☐ 수평 뒤집기
- ☐ 밝기 (±30%)
- ☐ 노이즈

학습 시작

바운딩 박스:

원본 이미지

감지된 객체

Part 4

웹 구현

객체 감지 및 수동 라벨링

파일 선택 111C29044CFE2B0335.jpg 업로드 및 감지

업로드된 이미지:
111C29044CFE2B0335.jpg

보기/편집 ×

감지 도구
학습된 모델: custom_model_202412
객체 감지

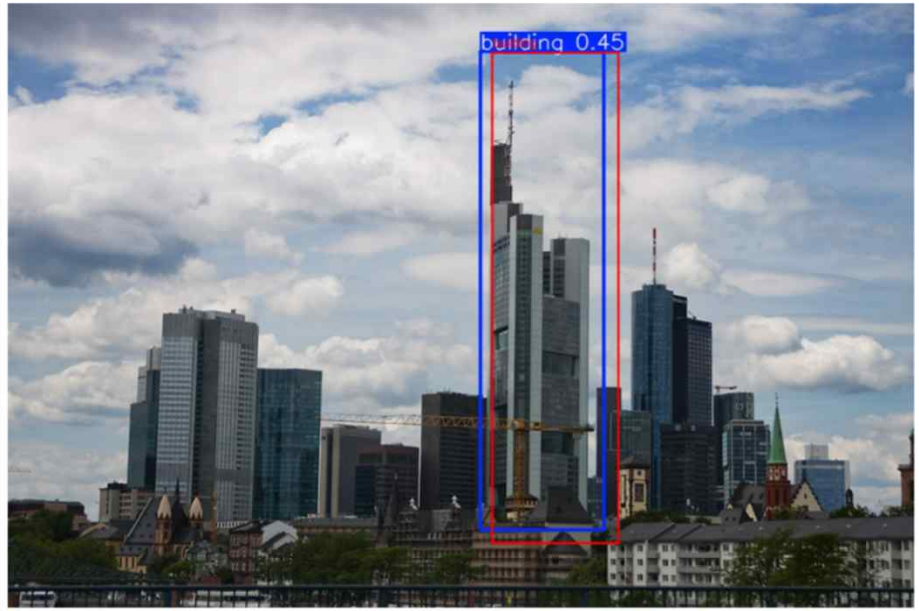
라벨링 도구
클래스 이름 입력
라벨 저장
마지막 라벨 취소

학습 도구
데이터 증강:
- ☐ 회전 (±30°)
- ☐ 수평 뒤집기
- ☐ 밝기 (±30%)
- ☐ 노이즈

학습 시작

바운딩 박스:
Box 1: building (x: 448, y: 46, w: 117, h: 453) 삭제

원본 이미지



감지된 객체

Part 5

자체 평가 의견

Part 5

어려운 점

논문 내용

- 복잡한 개념과 용어
  : 복잡한 공식과 용어의 이해
- 논문 연구의 적용 난이도
  : 제안된 기술을 코드로 구현

개발 환경

- 환경 설정의 복잡성
  : 다양한 라이브러리 설치 및 호환성
- 하드웨어 리소스의 한계
  : GPU 메모리 및 연산 자원의 불충분

모델 개발

- few shot object detection 구현
  : 소량의 객체 학습의 적용이 어려움
- 객체 추가시 학습 안정성
  : 새로운 클래스를 학습할 때 안정성

모델 서빙

- 서빙 환경 구축
  : 예측을 하는데 서버 측 문제 발생
- 모델의 서빙 속도
  : 학습을 하는데 시간이 오래 걸림

Part 5

개선할 점

모델

- 모델 경량화

  : 서빙 속도와 메모리 사용량 줄이기

- 적합한 모델

  : 오토 라벨러 기능이 잘 작동되는 모델

오토 라벨러

- LVC 구조의 심화 학습

  : 논문에 대한 더 깊은 이해

- 오토 라벨러의 기능 학습

  : 오토 라벨러 기능에 대한 더 깊은 이해

모델 서빙

- 서빙 최적화

  : 서빙 속도 및 안정성 높이기

모델 정확도

- 데이터 증강 추가

  : 데이터 다양성 높이고 일반화 능력 향상

- Loss Function 개선

  : 클래스 불균형 문제 해결

Part 5

프로젝트를 진행하면서 느꼈던 점

한세진

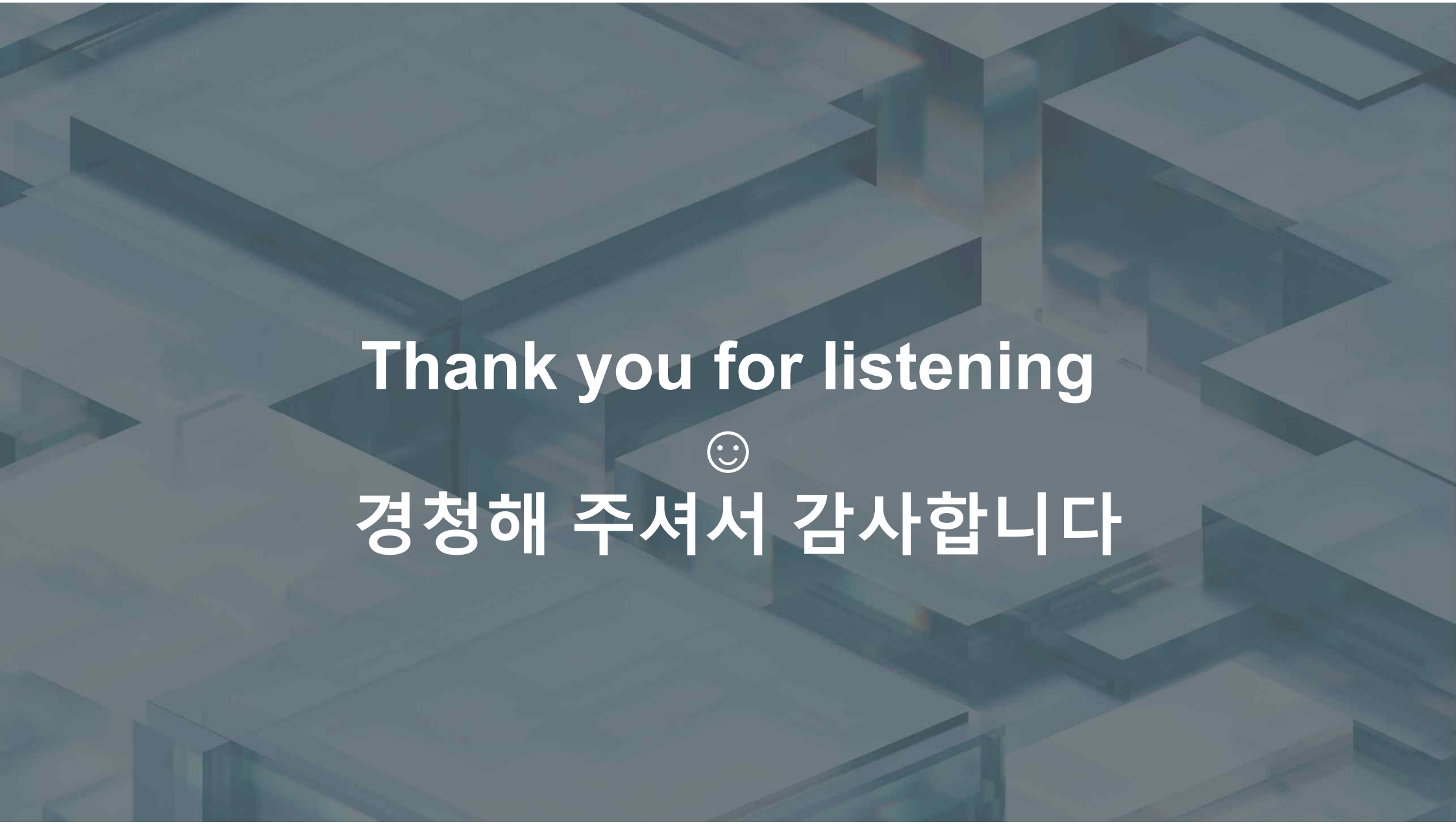
- 팀원들의 소중함
- 의사소통의 중요성
- 업무 자동화의 중요성

김경환

- AI 모델을 웹에 구현함에 어려움
- 서로간에 의사소통의 중요성

황지원

- FSOD 기법 적용의 어려움
- 다양한 OS 사용으로 인한 공부의 즐거움


Thank you for listening
☺
경청해 주셔서 감사합니다

Q & A