

차세대 인공지능 로봇청소기를 위한 소형 객체 탐지 모델

고혜령 문성우 박태희 윤필성용



목 차

1. 프로젝트 배경

- 로봇청소기 시장 동향
- 로봇청소기 장애물 탐지 모듈의 현 위치
- 로봇청소기 시장에서의 딥러닝 전환가치
- 목표 성과

2. 프로젝트

- 데이터
- 모델 아키텍쳐
- 데이터 증강
- 하이퍼파라미터

3. 추후 개선방향 및 시연

- 개선방향
- 시연

프로젝트 진행

일자	내용
23.07.17 ~ 18	주제 선정 및 마일스톤 설정
23.07.19	(파일럿)데이터 수집(촬영) 및 전처리
23.07.20~	데이터 수집(촬영) 방식 통일 및 수행
23.07.24	P&R 및 데드라인 설정
23.07.25	검증 데이터 확보, 기획의도 재점검
23.07.27	수집한 데이터 이슈 공유 및 EDA
23.07.27~	ablation study
23.07.28	Validation 기준 확립
23.08.02	real data에 대한 일반화 전략 논의
23.08.05	ablation study 결과 공유 및 모델 최적화 논의
23.08.06~07	하이퍼파라미터 튜닝
23.08.08	발표준비
총 프로젝트기간	25일(23.07.17 ~ 23.08.09)

팀 내 역할 분배

이름	담당 업무
공통	- 데이터 수집(촬영) 및 annotation 작업 - 모델링, 하이퍼파라미터 튜닝
고혜령	- Color Jittering Augmentation
문성우	- Albumentations 메소드 선별 - Horizontal Mosaic 적용
박태희	- loss 하이퍼파라미터 튜닝 - 마일스톤 관리
윤필성용	- Edge detection, Inverse augmentation 적용

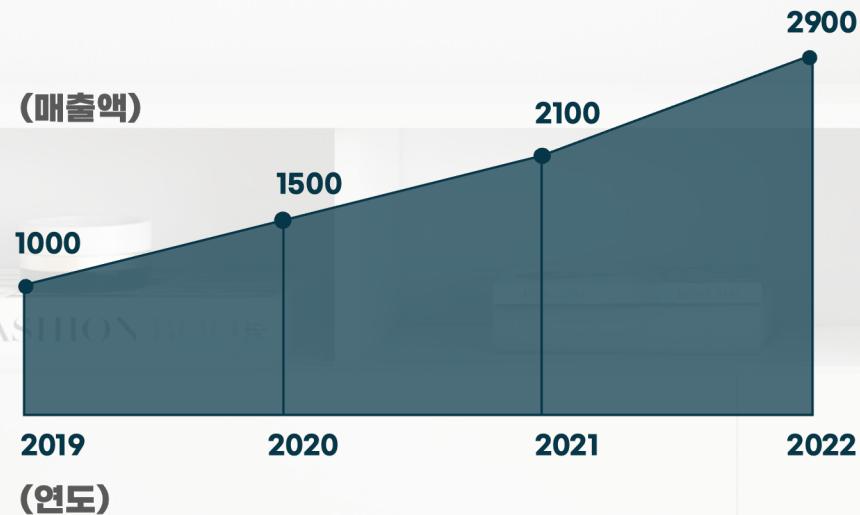
1. 프로젝트 배경 및 기획의도

로봇청소기 시장 동향

SONY

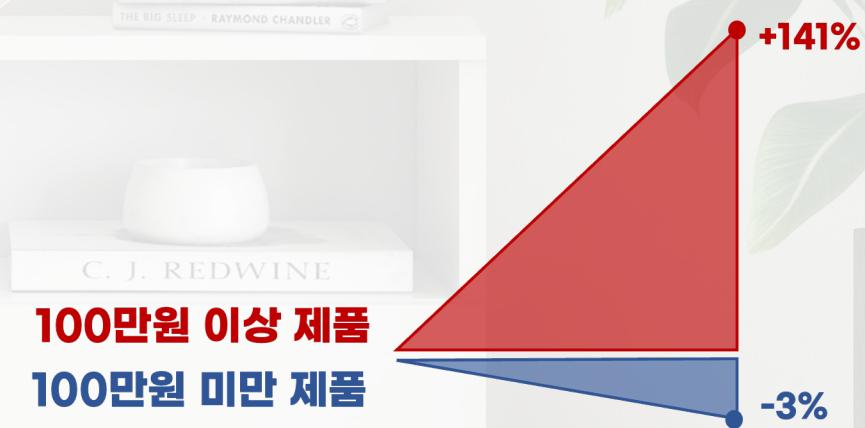
4개년 간 국내 로봇청소기 시장 규모 및 성장률 추이

(단위: 억원, 매출금액 기준)



2022년 국내 로봇청소기 판매량 성장률

(2021년 동기 대비, 판매량 기준)



로봇 청소기 시장의 규모는 2022년 기준 **4년 전 동기 대비 약 3배 성장**

그 중에서도 **하이엔드 제품**에 대한 수요 급증

1. 프로젝트 배경 및 기획의도

하이엔드 제품 수요 증가 요인 - 제품의 마케팅 방향

강조 기능 - 청소능력

1. 자동 먼지 비움
2. 듀얼 메인 브러쉬
3. 듀얼 흡입력
4. 음파 진동 물걸레

강조 기능 - 센서

5. 오토 리프팅
6. Reactive 3D 장애물 회피 시스템
7. IR 적외선 감지
8. PreciSense LIDAR 네비게이션
9. 6배 빨라진 매핑 속도

로보락 로봇청소기 S8 Plus

★★★★★ 381개 상품평

11% ↓ 1,012,000원 ⓘ

947,000원 쿠팡판매가

899,000원 와우쿠폰할인가 ⚡ 로켓배송

1. 프로젝트 배경 및 기획의도

하이엔드 제품 수요 증가 요인 - 이용자의 고충, 주요 고장 원인

삼성전자서비스 홈페이지

로봇청소기 점검코드별 조치 방법

점검코드	03
원인	오른쪽 구동바퀴 이물질(실, 종이, 장난감 등) 걸림
조치	바닥면에 있는 비상스위치를 끈 후 오른쪽 구동바퀴에 걸린 이물질을 제거하여 주세요.

고질적으로 발생하지만 조치가 애매한 점검상황

Dog Poop, Dirty Rugs, and Other Disappointing Truths About Robot Vacuums

PUBLISHED MAY 10, 2019



감지하지 못하는 장애물에 의해 발생하는 사고

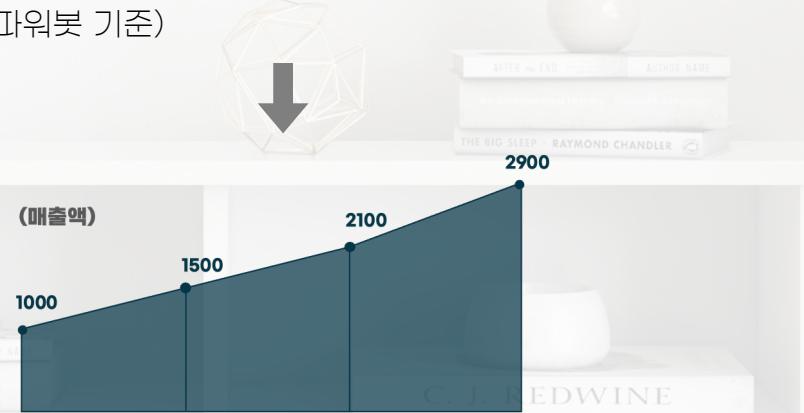
1. 프로젝트 배경 및 기획의도

딥러닝 전환 가치

SONY

풀질보증기간 내 단 한 번이라도 바퀴 한 쪽 고장 시
한 대당 최소 5만원의 손실 발생

(삼성 로봇청소기 파워봇 기준)



이 때, 단순 판매량 증가 == 기대비용의 증가로 직결

잔고장 요인을 로봇청소기가 판단하고 피할 수 있다면,

풀질 보증 비용 발생 최소화 (기업 입장)
고객센터 방문 빈도 감소 (소비자 입장)

1. 프로젝트 배경 및 기획의도

목표 성과

기존 로봇청소기가 탐지하는 못하는 객체 탐지 후 회피



귀중품 탐지



2. 프로젝트

모델 선정: YOLOv5, YOLOv8n 비교

모델 비교 (출처 Ultralytics)

Model	mAP50-95) (val)	params(M)	FLOPs
YOLOv5n	28.0	1.9	4.5
YOLOv5s	37.4	7.2	16.5
YOLOv8n	37.3	3.2	8.7

선정 기준

임베디드 소프트웨어에 적용될 것을 고려,
성능 대비 용량이 가벼운 최적의 모델

모델 baseline으로 학습 시
mAP 50-95) 기준 하위 10개 라벨의 평균값

Model	mAP 50-95)
YOLOv5s	0.6614
YOLOv8n	0.7373

YOLOv8n

모델 크기, 연산속도, mAP
모든 측면에서 v8n이 더 우수한 성능

2. 프로젝트

데이터: 수집기준 - 수집항목 설정



흡입불가

양말 등 6종



가늘고 긴 물질

신발끈 등 9종



가늘고 길지 않은 물질

풀티슈 등 12종



귀중품

반지 등 13종

2. 프로젝트

데이터: 수집기준 - 촬영기준 설정

카메라

- 위치: 바닥에 바짝 붙여서
- 각도: 바닥에서 수직으로

촬영 기준

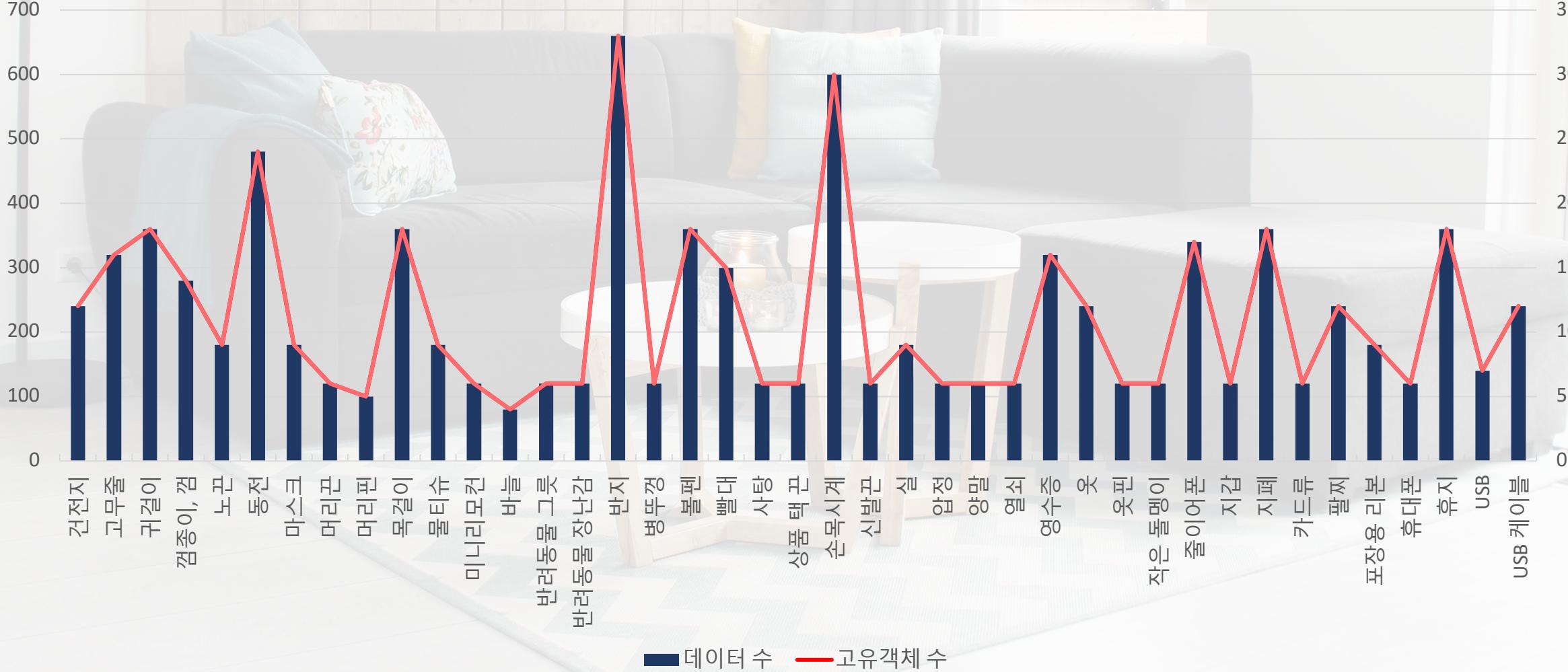
각도	상태	거리
물체를 중심으로 360° 회전	물체를 바르게 놓은 상태	근거리 (물체가 화면에 비치는 길이의 절반)
	물체를 뒤집어놓은 상태	중거리
	기타 다양한 상태	장거리 (약 30cm 지점)

같은 물체여도 상태가 다르면 다른 객체로 간주

2. 프로젝트

데이터: 수집현황

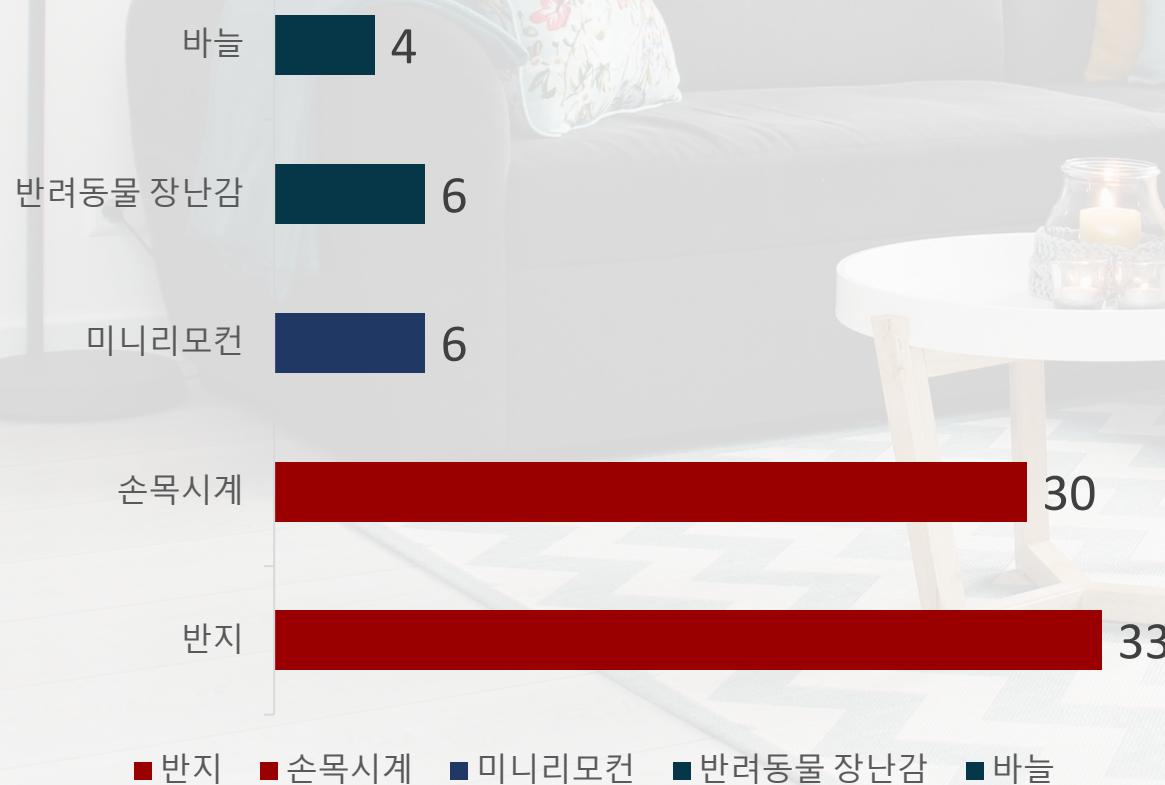
데이터 수집 현황표



2. 프로젝트

데이터: 수집현황

데이터 수집 현황표(최대최소 비교)



수집량이 적은 경우

다양한 가짓수가 존재하지 않는 객체 (바늘 등)

주변에서 구하기 어려운 객체(미니리모컨 등)

수집량이 많은 경우

일반화 성능을 확인하기 위해 다양성 확보한 라벨

주변에서 구하기 비교적 쉬웠던 객체(반지 등)

2. 프로젝트

데이터 수집의 한계

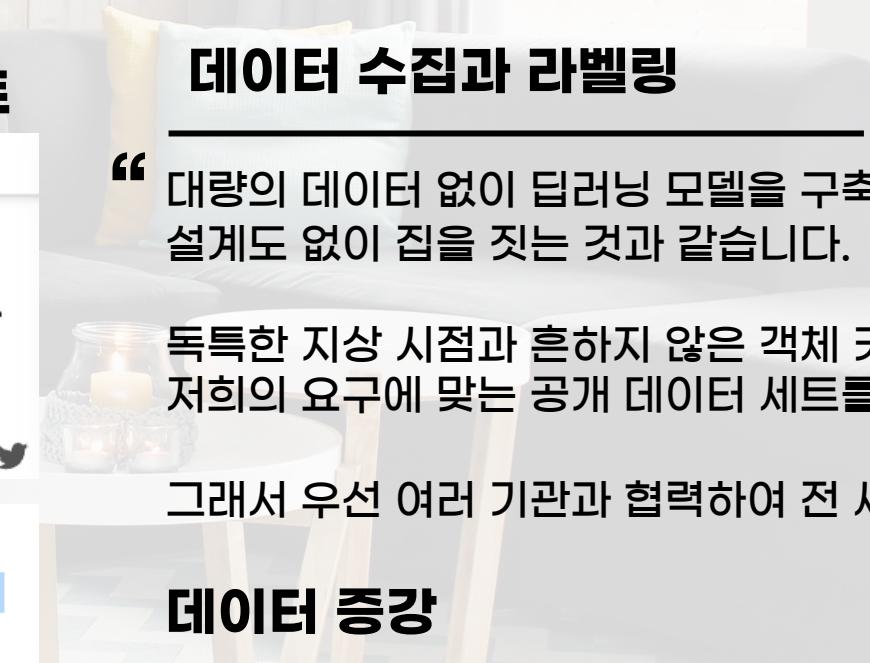
Ecovacs Robotics사의 로봇청소기용 딥러닝 모델 개발 후 포스트

TensorFlow Blog

Community · TensorFlow Lite ·

Ecovacs Robotics: the AI robotic vacuum cleaner powered by TensorFlow

January 07, 2020



Data collection and annotation

Building a deep learning model without large amounts of data is like making a house without blueprints. Due to the unique ground-view perspective and uncommon object categories, we cannot find any public dataset which fit our needs. Therefore, we first cooperated with many institutions to collect data from all over the world. Although manual data collection took up a lot

Data Augmentation

In developing the model, data augmentation played a significant role. For example, by changing the hue values of the image, we can simulate floors and mats of various colors. As mentioned

데이터 수집과 라벨링

“ 대량의 데이터 없이 딥러닝 모델을 구축하는 것은 설계도 없이 집을 짓는 것과 같습니다.

독특한 지상 시점과 흔하지 않은 객체 카테고리로 인해 저희의 요구에 맞는 공개 데이터 세트를 찾을 수 없었습니다.

그래서 우선 여러 기관과 협력하여 전 세계의 데이터를 수집했습니다

데이터 증강

“ 이 모델을 개발할 때 데이터 증강이 중요한 역할을 했습니다. 예를 들어 이미지의 색조 값을 변경하여 다양한 색상의 바닥과 매트를 시뮬레이션할 수 있습니다.

2. 프로젝트

데이터 수집의 한계

YOLOv5와 segmentation 데이터를 이용한 산업용 로봇청소기용 모델 개발 논문

Open Access Editor's Choice Article

A Deep Learning-Based Dirt Detection Computer Vision System for Floor-Cleaning Robots with Improved Data Collection

by  Daniel Canedo ,  Pedro Fonseca ,  Petia Georgieva  and  António J. R. Neves 

Bormann et al. [4] proposed a tool to artificially generate data [12] to tackle the scarcity of data in this area. In

“

보만 등[4]은 이 분야의 데이터 부족 문제를 해결하기 위해 데이터를 인위적으로 생성하는 도구[12]를 제안했습니다.

- Great variations in light intensity.
- Complex floor patterns.
- Lack of enough labelled images covering various dirty scenarios.
- Blurred images due to robot movement.
- Dirt/Clean discrimination.

In order to tackle the challenges mentioned above, our work takes the tool proposed by [4] and adapts it to generate a synthetic dataset that has a great floor variety and dirt samples. This dataset is used to train on the YOLOv5 model to detect dirt in images.

“

위에서 언급한 문제를 해결하기 위해 저희는 [4]에서 제안한 도구를 사용하여 다양한 바닥과 먼지 샘플이 포함된 합성 데이터 세트를 생성할 수 있도록 조정했습니다

2. 프로젝트

Augmentation 개요 - 목적에 따른 구분

부족한 데이터 보강을 위해

Bounding Box Color Jittering

4 Albulmentations

성능 개선을 위해

custom Mosaic

5 Albulmentations

Edge Detection

Image Inverse Augmentation

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering



‘모든 데이터’ 수집의 어려움

모든 색상의 샘플을 수집하여 촬영하는 비용의 문제

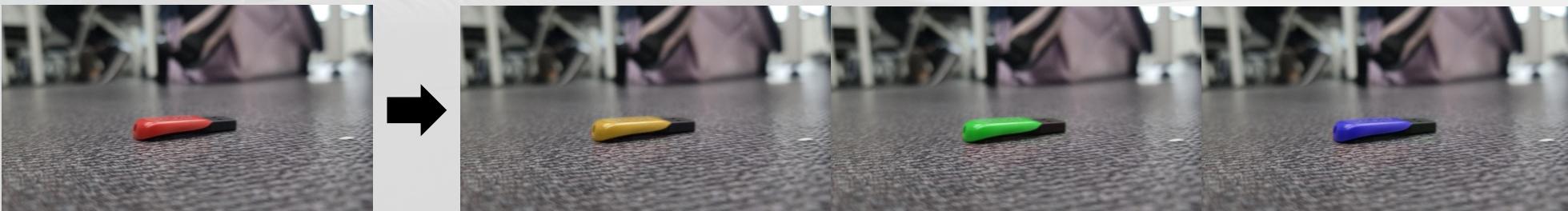
설령 모았다고 한들 이를 ‘모든 색상’이라고 확신할 수 있는가?

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

Bounding Box 영역 내 색상 핸들링

다양한 색의 객체(object) 데이터를 학습한 효과를 기대

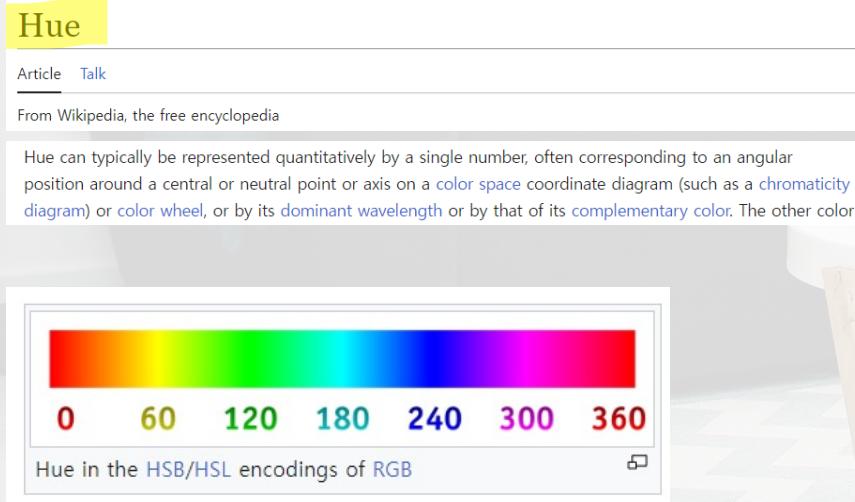


이미지 전체에 대한 색상 변경과 달리 bBox 내 색상만 변경하는 것은
이미지 내 색상 분포를 바꾸는 작업이기 때문에
보다 더 현실에 가까운 다른 색상의 객체로 인식될 가능성을 극대화할 수 있다.

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

Albumentations의 HueSaturationValue와 Wikipedia를 참고, Hue(색조) 핸들링



```
1 hue_shift = [20, 60, 120, 180, 240, 300, 360]
```

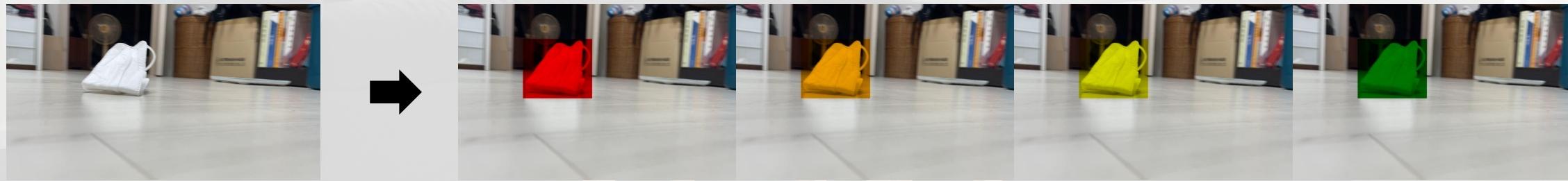
```
1 def hue(img, hue_shift):
2     dtype = img.dtype
3     img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) # 이미지를 HSV로 변환
4     hue, sat, val = cv2.split(img) # 각 채널을 나눔
5
6     lut_hue = np.arange(0, 256, dtype=np.int16)
7     lut_hue = np.mod(lut_hue + hue_shift, 180).astype(dtype)
8     hue = cv2.LUT(hue, lut_hue)
9
10    img = cv2.merge((hue, sat, val)).astype(dtype)
11    img = cv2.cvtColor(img, cv2.COLOR_HSV2BGR)
12    return img
```

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

하지만, 흰색은 hue 조정으로 바꿀 수 없는 문제

Wikipedia를 참고, 단일 색상의 이미지와 혼합(blend)하는 방식 채택



blend 방식 중, multiply는 각 픽셀값이 0 ~ 1 사이인 두 이미지를 element-wise하게 곱하는 방식으로, 밝은 색은 곱하는 색만큼 어두워지고, 검은색은 0이기 때문에 영향이 없다.

따라서 흰색 객체의 색상을 변경하는데에 유효한 방법

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering



```
3 colorchart = {  
4     'red' : (255, 0, 0),  
5     'orange' : (255, 165, 0),  
6     'yellow' : (225, 225, 0),  
7     'green' : (0, 128, 0),  
8     'cyan' : (0, 225, 225),  
9     'blue' : (0, 0, 225),  
0     'purple' : (128, 0, 128),  
1     # 'pink' : (255, 192, 203)  
2 }
```

Multiply [edit]

Multiply blend mode takes the RGB channel values from 0 to 1 of each pixel in the top layer and multiples them with the values for the corresponding pixel from the bottom layer. Wherever either layer was brighter than black, the composite is darker; since each value is less than 1, their product will be less than each initial value that was greater than zero.

$$f(a, b) = ab,$$

where a is the base layer value and b is the top layer value.

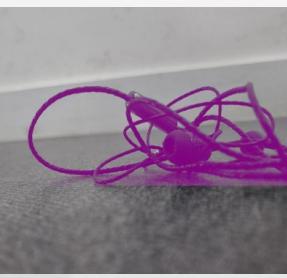
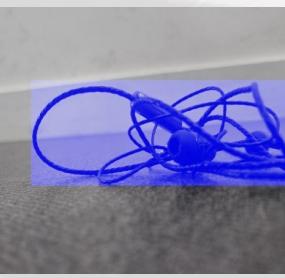
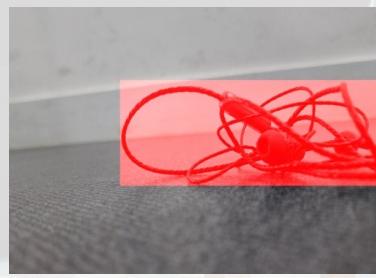
```
1 def multiply(img1, img2): # base, blend  
2     dtype = img1.dtype  
3     size = img1.shape[:2]  
4     if size != img2.shape[:2]:  
5         img2 = img2[:size[0], :size[1], :]  
6  
7     img1, img2 = img1 / 255, img2 / 255 # (0, 1) 사이의 값으로 normalize  
8     result_img = np.multiply(img1, img2)  
9     result_img = result_img * 255 # normalize 했던 것 되돌리기  
10    return result_img.astype(dtype)
```

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

하지만, 검은색은 hue, multiply 둘 다 안되는 문제

Wikipedia를 참고, lighten only 혼합 방식을 채택



lighten only

두 이미지의 픽셀 간 더 밝은 값만 선택하는 방식. 따라서 검은 색은 변경하는데 유효

2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

Lighten Only [edit]

Lighten Only has the opposite action of *Darken Only*. It selects the maximum of each component from the foreground and background pixels. The mathematical expression for *Lighten Only* is:[10]

$$[\max(r_1, r_2), \max(g_1, g_2), \max(b_1, b_2)]$$

```
3 colorchart = {  
4     'red' : (255, 0, 0),  
5     'orange' : (255, 165, 0),  
6     'yellow' : (225, 225, 0),  
7     'green' : (0, 128, 0),  
8     'cyan' : (0, 225, 225),  
9     'blue' : (0, 0, 225),  
10    'purple' : (128, 0, 128),  
11    # 'pink' : (255, 192, 203)  
12}
```

```
1 def lighten_only(img1, img2): # base, blend  
2     dtype = img1.dtype  
3     size = img1.shape[:2]  
4     if size != img2.shape[:2]:  
5         img2 = img2[:size[0], :size[1], :]  
6     result_img = cv2.max(img1, img2.astype(dtype))  
7     return result_img.astype(dtype)
```

데이터 수 증강

6692 -> 20920장

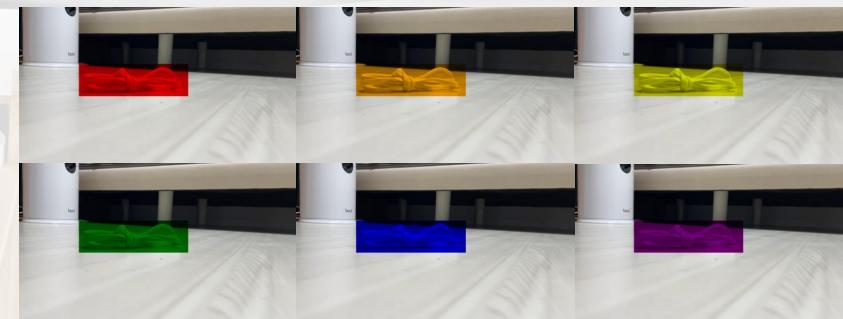
2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

Color Augmentation 적용 효과



흰 신발끈 데이터만 주고 학습 시켰을 때

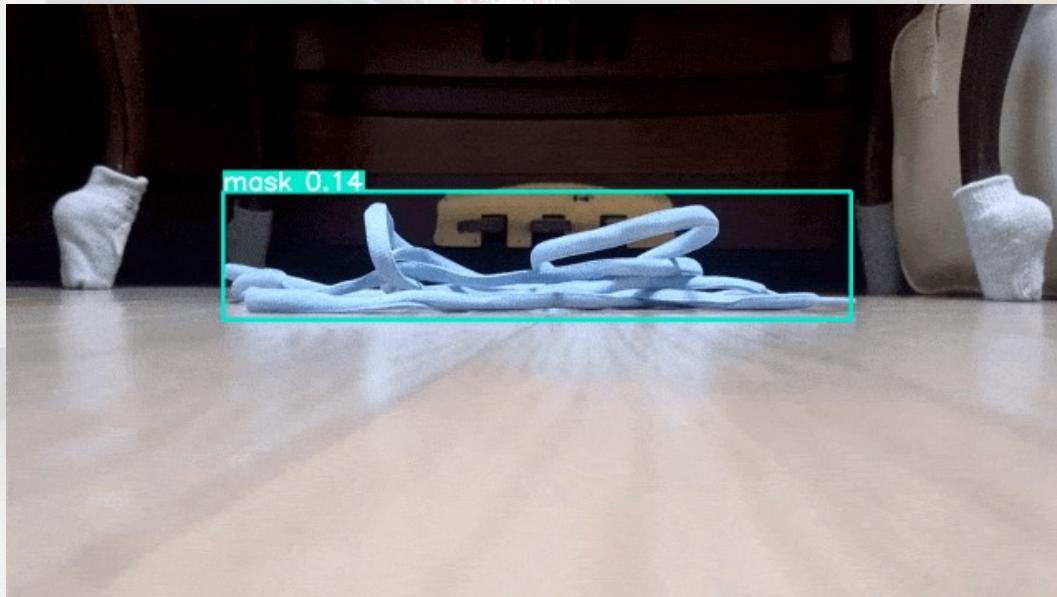


흰 신발끈 데이터에
Color Jittering을 주고 학습시켰을 때

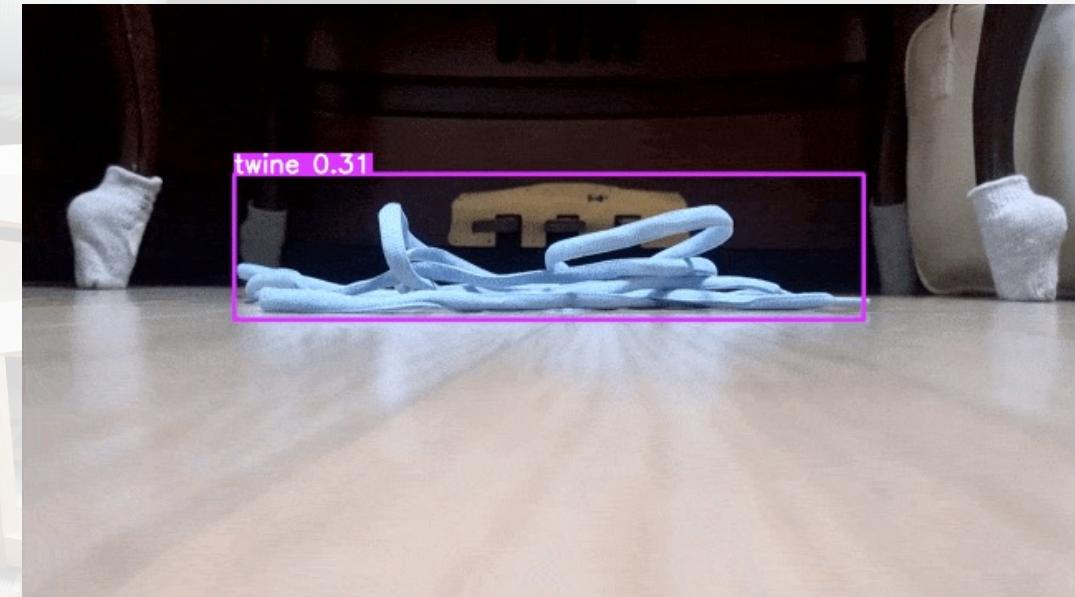
2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

적용 전



적용 후



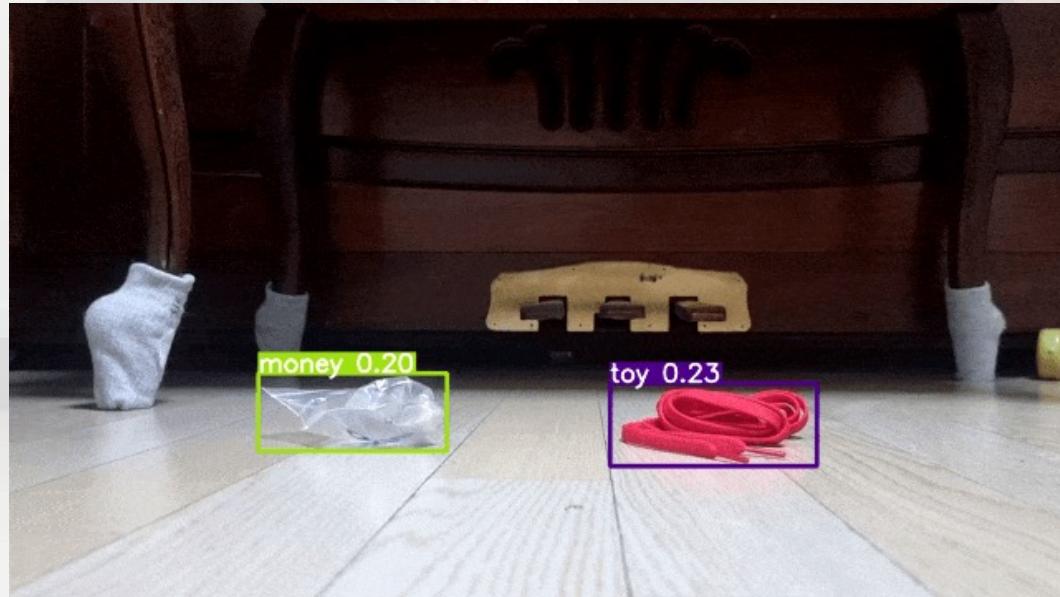
'shoelace'를 detect한 프레임 수 : 61

194

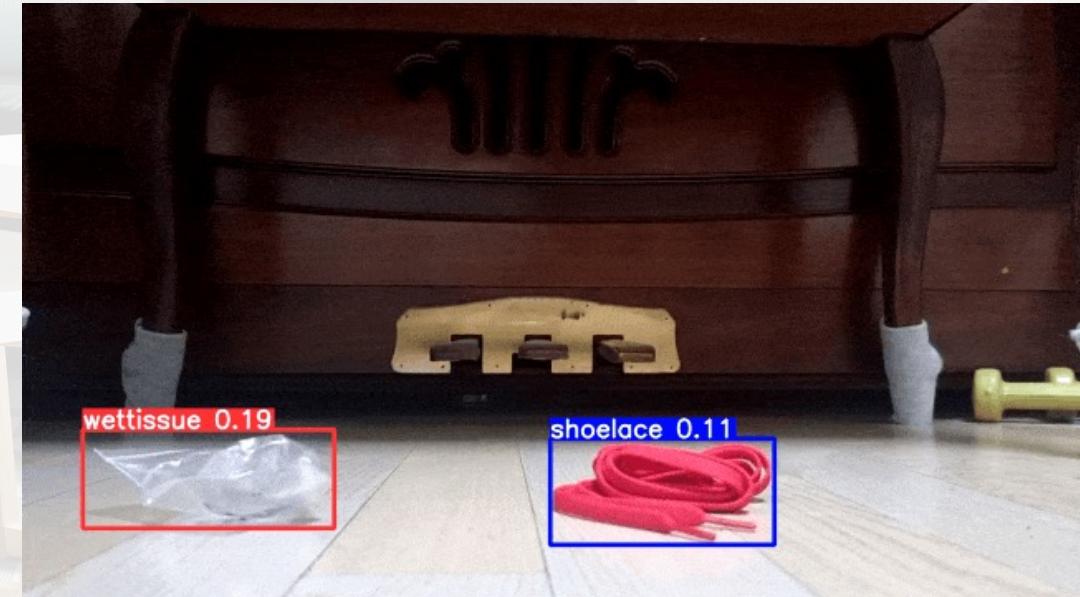
2. 프로젝트

데이터 증강(Augmentation) - Color Jittering

적용 전



적용 후



'shoelace'를 detect한 프레임 수 : 6



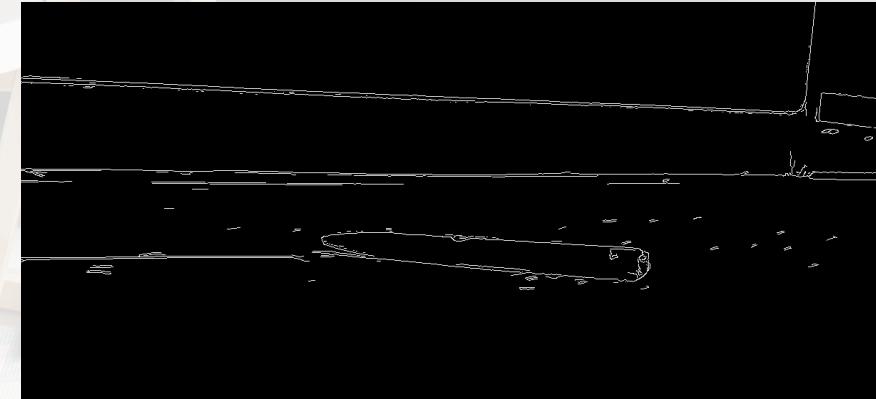
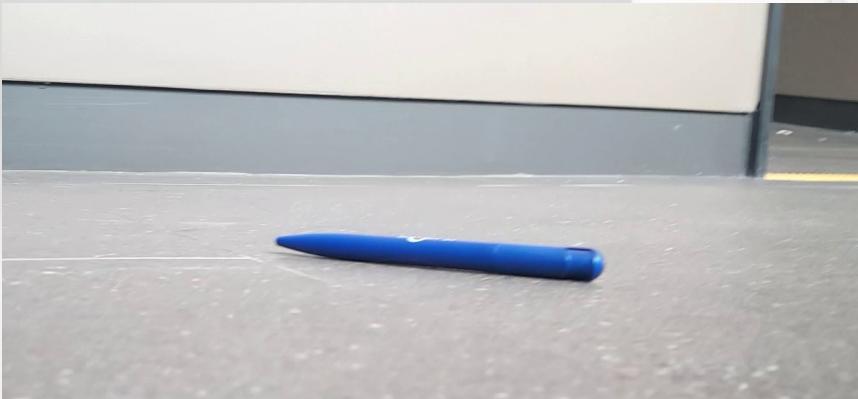
263

2. 프로젝트

데이터 증강(Augmentation) - Edge, Inverse

Edge Detection: 형태만을 유일한 정보로 받아들여보자

윤곽을 통한 형태 정보 외에는 모두 노이즈로 간주, 다른 색상의 데이터가 들어와도 형태/윤곽만 보고 올바른 판단을 할 수 있을 것으로 기대.

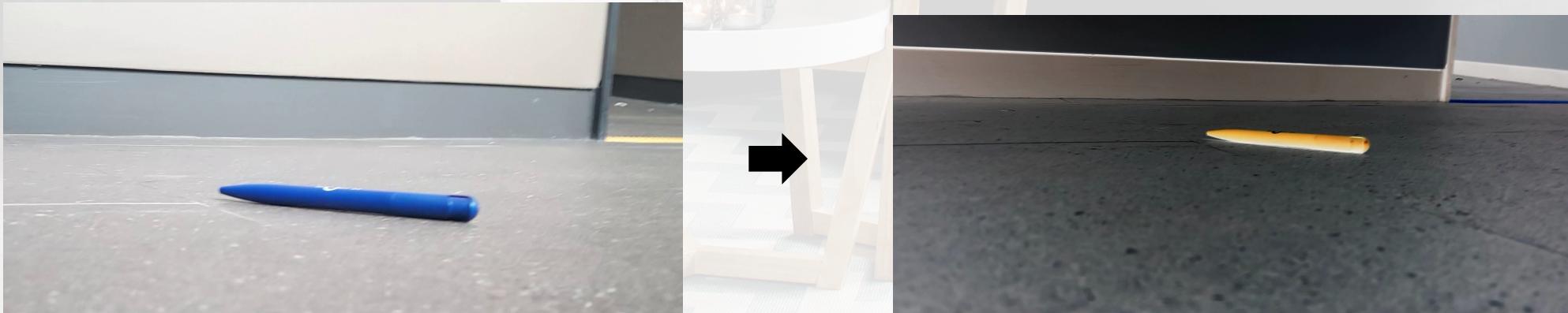


2. 프로젝트

데이터 증강(Augmentation) - Edge, Inverse

Inverse Augmentation:

윤곽을 통한 형태 정보 외에는 모두 노이즈로 간주, 다른 색상의 데이터가 들어와도 형태/윤곽만 보고 올바른 판단을 할 수 있을 것으로 기대.



2. 프로젝트

데이터 증강(Augmentation) - Edge, Inverse

Inverse Augmentation:

InverseAug: Accurate Alignment under Geometric Augmentation

To achieve good performance on existing 3D object detection benchmarks for autonomous cars, most methods require strong data augmentation during training to avoid overfitting. However, the necessity of data augmentation poses a non-trivial challenge in the DeepFusion pipeline. Specifically, the data from the two modalities use different augmentation strategies, e.g., rotating along the z-axis for 3D point clouds combined with random flipping for 2D camera images, often resulting in alignment that is inaccurate. Then the augmented LiDAR data has to go through a **voxelization** step that converts the point clouds into volume data stored in a three dimensional array of **voxels**. The voxelized features are quite different compared to the raw data, making the alignment even more difficult. To address the alignment issue caused by geometry-related data augmentation, we introduce Inverse Augmentation (InverseAug), a technique used to reverse the augmentation before fusion during the model's training phase.

In the example below, we demonstrate the difficulties in aligning the augmented LiDAR data with the camera data. In this case, the LiDAR point cloud is augmented by rotation with the result that a given 3D key point, which could be any 3D coordinate, such as a LiDAR data point, cannot be easily aligned in 2D space simply through use of the original LiDAR and camera parameters. To make the localization feasible, InverseAug first stores the augmentation parameters before applying the geometry-related data augmentation. At the fusion stage, it reverses all data augmentation to get the original coordinate for the 3D key point, and then finds its corresponding 2D coordinates in the camera space.

하이엔드 제품에서의 LiDAR 센서 탑재 경향에 따른 데이터 대응

flip 등 Augment가 진행됨에 따라 3D 데이터와 2D 데이터의 정렬이 부정확해지는 문제 발생.

본격적인 Augment 이전에 기하학적으로 정렬 문제를 해소하는 전처리 방법론으로써 접근

2. 프로젝트

데이터 증강(Augmentation) - Edge, Inverse

```
edge.py
1 import cv2 as cv
2 import numpy as np
3
4 obj = "blue_pen_1.mp4"
5 base_dir = "augmentation/"
6 video_path = base_dir + obj
7 save_path = base_dir
8 cap = cv.VideoCapture(video_path)
9
10 count = 0
11
12 while cap.isOpened():
13
14     ret, frame = cap.read()
15
16     if not ret:
17         print("* No new frame! \n"); break
18
19     ext = cv.Canny(frame, 50, 150)
20     inversed = ~frame ### inversed:
21
22     cv.imwrite(base_dir + f'inversed/inv_{count}.jpg', inversed)
23
24     count += 1
25
26     if cv.waitKey(10) == 27: break
27
28 cv.waitKey(0); cv.destroyAllWindows()
```

Open CV 라이브러리

- edge 추출 : 19번줄(`cv.Canny`)
- Inversed: 20번줄

2. 프로젝트

데이터 증강(Augmentation) - Horizontal Mosaic



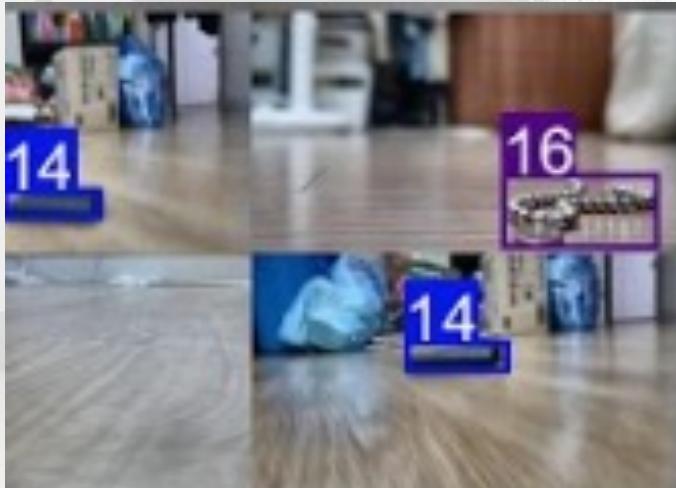
문제 상황

real data의 객체는 땅, 바닥 위에 존재하는 반면에, 기존의 mosaic을 통해 생성되는 이미지는 격자무늬로 이미지를 합치는 과정에서 바닥이 아닌 위치에서 데이터가 만들어지는 현상이 발생.

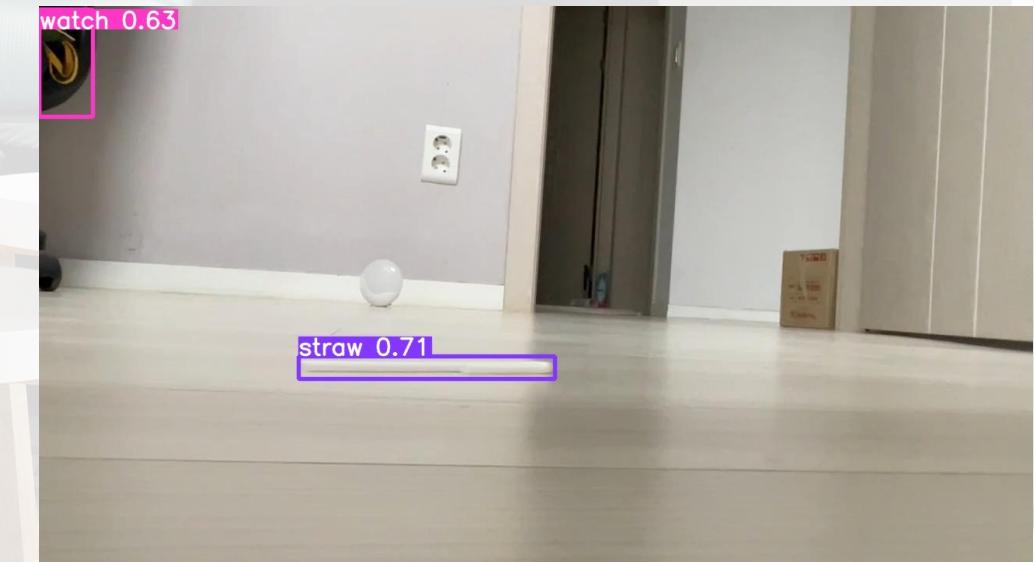
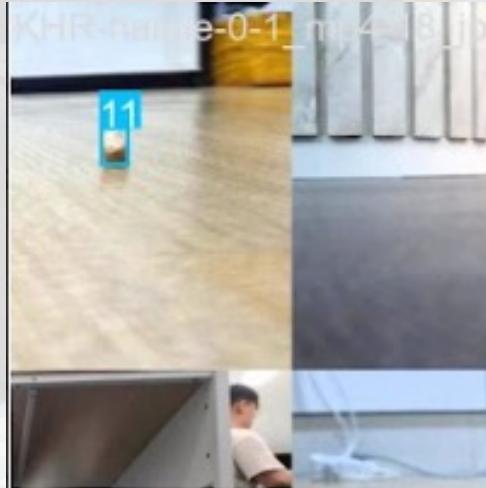
2. 프로젝트

데이터 증강(Augmentation) - Horizontal Mosaic

문제 상황



학습 시 mosaic4 (2, 2)가 적용된 모습



inference 때,
학습데이터에는 없던 위치에서 watch가 나타나는 현상

2. 프로젝트

데이터 증강(Augmentation) - Horizontal Mosaic

높이 변형 요소 제거

노이즈 제거를 위해 (1, 3) Mosaic으로 커스텀



학습 시 horizontal mosaic (1, 3)이 적용된 모습

```
def _mosaic3(self, labels):
    """Create a 1x3 image mosaic."""
    mosaic_labels = []
    s = self.imgsz
    hp, wp = -1, -1 # height, width previous
    for i in range(3):
        labels_patch = labels if i == 0 else labels['mix_labels'][i - 1]
        # Load image
        img = labels_patch['img']
        h, w = labels_patch.pop('resized_shape')

        # Place img in img3
        if i == 0: # center
            img3 = np.full((s * 3, s * 3, img.shape[2]), 114, dtype=np.uint8) # base image with 3 tiles
            h0, w0 = h, w
            c = s, s, s + w, s + h # xmin, ymin, xmax, ymax (base) coordinates
        elif i == 1: # right
            c = s + w0, s, s + w0 + w, s + h
        elif i == 2: # left
            c = s - w, s + h0 - h, s, s + h0

        padw, padh = c[:2]
        x1, y1, x2, y2 = (max(x, 0) for x in c) # allocate coords

        img3[y1:y2, x1:x2] = img[y1 - padh:, x1 - padw:] # img3[ymin:ymax, xmin:xmax]
        hp, wp = h, w # height, width previous for next iteration

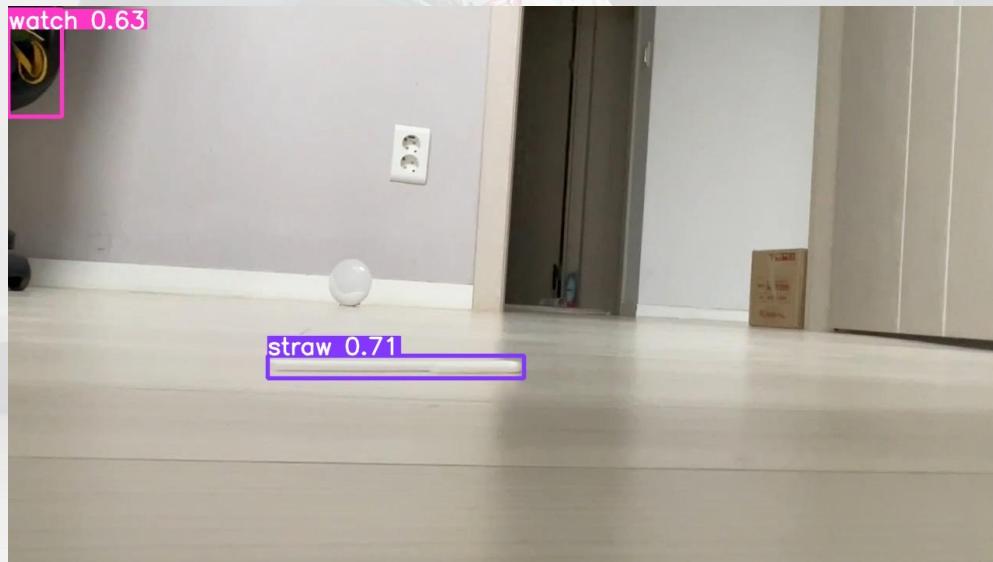
        # Labels assuming imgsz*2 mosaic size
        labels_patch = self._update_labels(labels_patch, padw + self.border[0], padh + self.border[1])
        mosaic_labels.append(labels_patch)
    final_labels = self._cat_labels(mosaic_labels)

    final_labels['img'] = img3[-self.border[0]:self.border[0], -self.border[1]:self.border[1]]
    return final_labels
```

2. 프로젝트

데이터 증강(Augmentation) - Horizontal Mosaic

mosaic 변경 전후 비교



Mosaic4



Horizontal Mosaic

2. 프로젝트

데이터 증강(Augmentation) - Horizontal Mosaic

mosaic 변경 전후 비교



Mosaic4

Horizontal Mosaic

2. 프로젝트

데이터 증강(Augmentation) - Albumentations

부족한 데이터 보강을 위해

Defocus

아웃포커싱된 데이터 기대

MotionBlur

움직이면서 생기는 잔상 노이즈 기대

ZoomBlur

줌 인/아웃 시 생기는 잔상 노이즈 기대

RandomBrightnessContrast

빛이 더 밝은/적은 환경에서의 데이터 기대

성능개선을 위해

MedianBlur

ToGray

CLAHE

Emboss

RandomGamma

2. 프로젝트

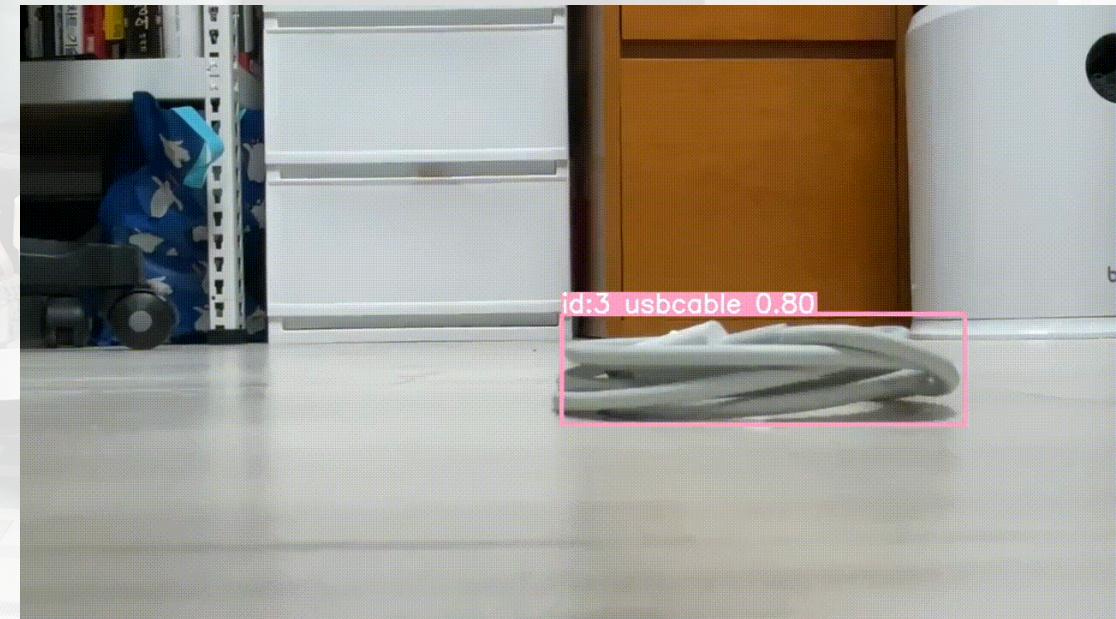
Loss 하이퍼파라미터 튜닝

문제 상황

Object 위치, Bounding Box 크기는 잘 추론하지만, Class에 대한 성능이 그만큼 확보되지 않는 현상 발생.

문제 접근

cls_loss gain, box_loss gain, dfl_loss gain 중
목적에 부합하는 적절한 cls_loss gain의 값 탐색



2. 프로젝트

Loss 하이퍼파라미터 튜닝

cls_loss 튜닝

alulation study를 통해
타 loss와의 trade-off를 최소화시키면서
class loss를 최소화시키는 값 탐색

값 최적화: (default) 0.5 -> 0.9

특이사항	metrics/mAP50(B)
Agumaintation : mosaic= 1.0, mixup = 0.3, scale=0.5	0.82702
Augmentation mixup = 0.3, scale=0.5, perspective=0.001, translate=0.1	0.80165
cls loss gain = 0.9 해당 값을 상승 조정하면 base line 대비 작은 객체에 대한 정확도가 상승함 확인결과 작은 객체에 가중치를 더 부여하며 탐지가 용이하도록함 (작은 객체를 탐지할때 조금더 관대하게(?) 라벨을 붙이는 것 같음) 정확도가 올라간다기 보단 작은 객체에 대해 더 러프하게 탐지하여 예측하는 것 같음	-
box loss gain 9.0 (baseline 대비 1.5 상승) 해당 값을 상승 조정하면 baseline 대비 작은 객체에 대한 예측 확률이 낮아짐 확인 결과 box loss gain은 bbox에 대한 디텍팅을 조금 덜 tight하게 잡는 파라미터로 확인됨 그럼에도 불구하고 작은 객체에대한 예측 확률이 낮아지는 점은 확인이 필요해 보임	-
box loss gain 1.0 (baseline 대비6.5 하락) box loss gain을 낮췄을때, 작은 객체의 정확도는 baseline 대비 올라갔지만, 얇게 펴져있는 객체에 대한 정확도는 baseline 대비 낮아짐	-
box=1, cls=0.9, perspective=0.001, mixup=0.5 이전보다 작은 bbox를 더 탐지하지 못하는 현상 발생...	0.78822
box=3.0 cls=1.0	0.85148

2. 프로젝트

Loss 하이퍼파라미터 튜닝



cls_loss : 0.5



0.9



3. 추후 개선방향

데이터

저조도 환경에서의 데이터 학습 필요

실제로 로봇청소기가 활동할 수도 있는 암전 상황 하에서는 카메라 센서만으로 객체 탐지가 어려웠음.

따라서 IR 센서를 이용해 저조도/암전 환경에 대비할 필요가 있어보임.

다양한 객체 확보의 어려움

색상의 다양성은 augmentation으로 일부 해소하였으나,
형태의 다양성은 이 방법으로 해소하기 어려웠음.

Predict 단계에서의 부정확성

obj, box 추론은 잘 하지만, 라벨 혼동이 잦게 일어남.

cls_loss gain 값 조정을 통해 일부 해소하였으나, 여전히 class 오판 문제가 남아있다.
보여지는 형태에 따라 라벨 재정의를 고려해볼 필요가 있음.

3. 시연

