

Deep Learning Term Project

오염 요소 기반 폐의류 재활용 분류 및 이미지 캡셔닝 모델 개발

Image-Based Classification and Captioning Model for Recycling Used Clothing with Contamination Annotations

이화여자대학교 데이터사이언스학과

2392004 김민지

2392038 한가은

*본 팀이 문제를 직접 정의하여 진행한 프로젝트입니다.

목차

01 문제 정의

02 프로젝트 소개

03 데이터 수집 및 적재

04 Object Detection : YOLOv8

05 Image Captioning : BLIP

06 발전가능성 및 적용

01 문제 정의

01 문제 정의

*본 팀이 문제를 직접 정의하여 진행한 프로젝트입니다.



이 중 상당수가
재사용, 재활용 가능함에도 불구하고
단순 폐기되고 있음.

의류 수거 및 분류 작업은 여전히 수작업 중심,
효율성과 정확성의 한계

매년 9,200톤의 섬유 폐기물 발생

01 문제 정의



오염 요소를 기반으로
재활용 가능 여부를 분류할 수 있다면,



효율적인 자원 순환과 폐기물 감소에
기여할 수 있지 않을까?

02 프로젝트 소개

02 프로젝트 소개

- 주제 : 오염 요소 기반 폐의류 재활용 분류 및 이미지 캡셔닝 모델 개발

- **프로젝트 개요**

의류와 오염 요소를 이미지에서 탐지하여 재사용(Reuse)/재활용(Recycle)/폐기(Dispose)로 분류하고,
각 의류에 대한 설명 문장을 자동 생성하는 딥러닝 파이프라인 구축

- **핵심 기술 요소**

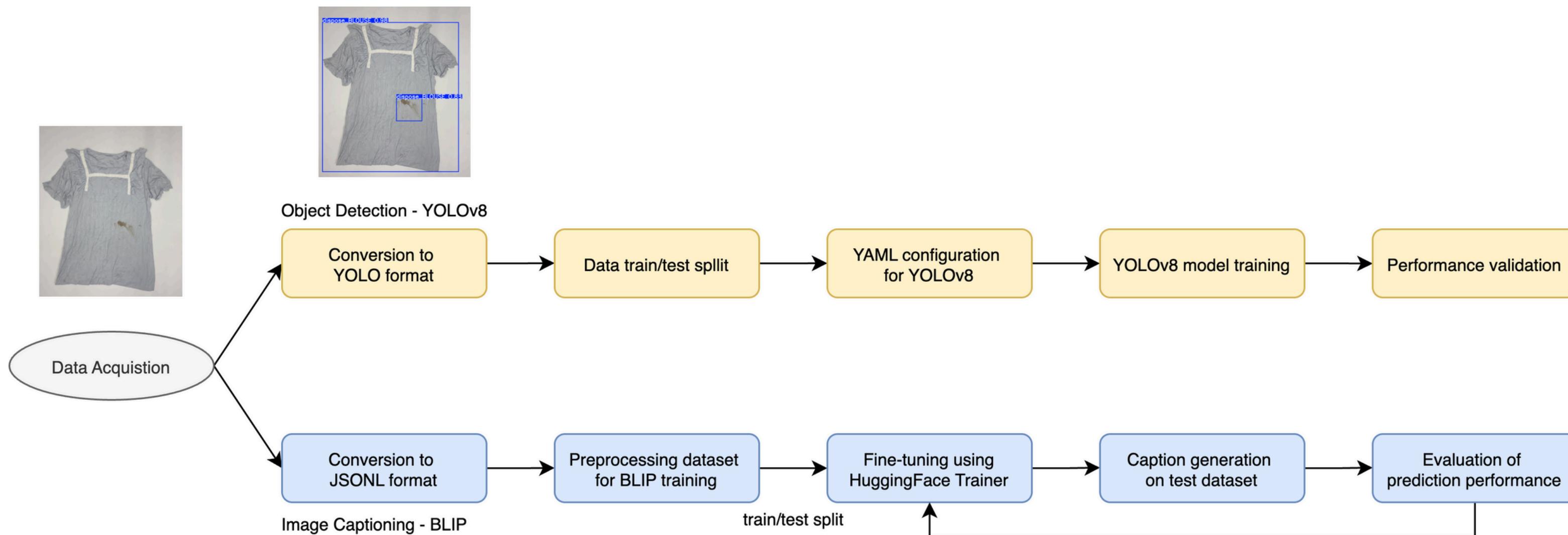
Object Detection : 의류 탐지 및 오염 요소 bounding box로 감지 (YOLOv8 학습)

Classification : 탐지된 오염 상태 기반 폐의류의 3단계 분류 (reuse / recycle / dispose)

Image Captioning : 의류의 시각 정보를 기반으로 자동 설명 문장 생성 (BLIP 학습)

02 프로젝트 소개

Project Workflow



예측된 캡션: 흰색과 스트라이프 패턴이 중앙에 여성용 반팔 블라우스
실제 캡션: 우측 중앙에 외부오염이 있는 하늘색 여성용 반팔 드레스

03 데이터 수집 및 적재

3-1 데이터 수집

데이터셋 소개

- **데이터명:** 폐의류 재활용 분류 및 선별 데이터
- **출처:** AI Hub (한국지능정보사회진흥원)
- **데이터 통계 :**
 - 원천이미지 : 260,000장, 라벨링 데이터 260,000장 (1:1 대응)



메타데이터 구조표			
데이터 영역	재난안전환경	데이터 유형	텍스트, 이미지
데이터 형식	JPG	데이터 출처	직접 촬영
라벨링 유형	바운딩박스(이미지), 질의응답(자연어)	라벨링 형식	json
데이터 활용 서비스	폐의류의 분류 및 손상, 오염, 부착물 탐지	데이터 구축년도/ 데이터 구축량	2024년/260,000set

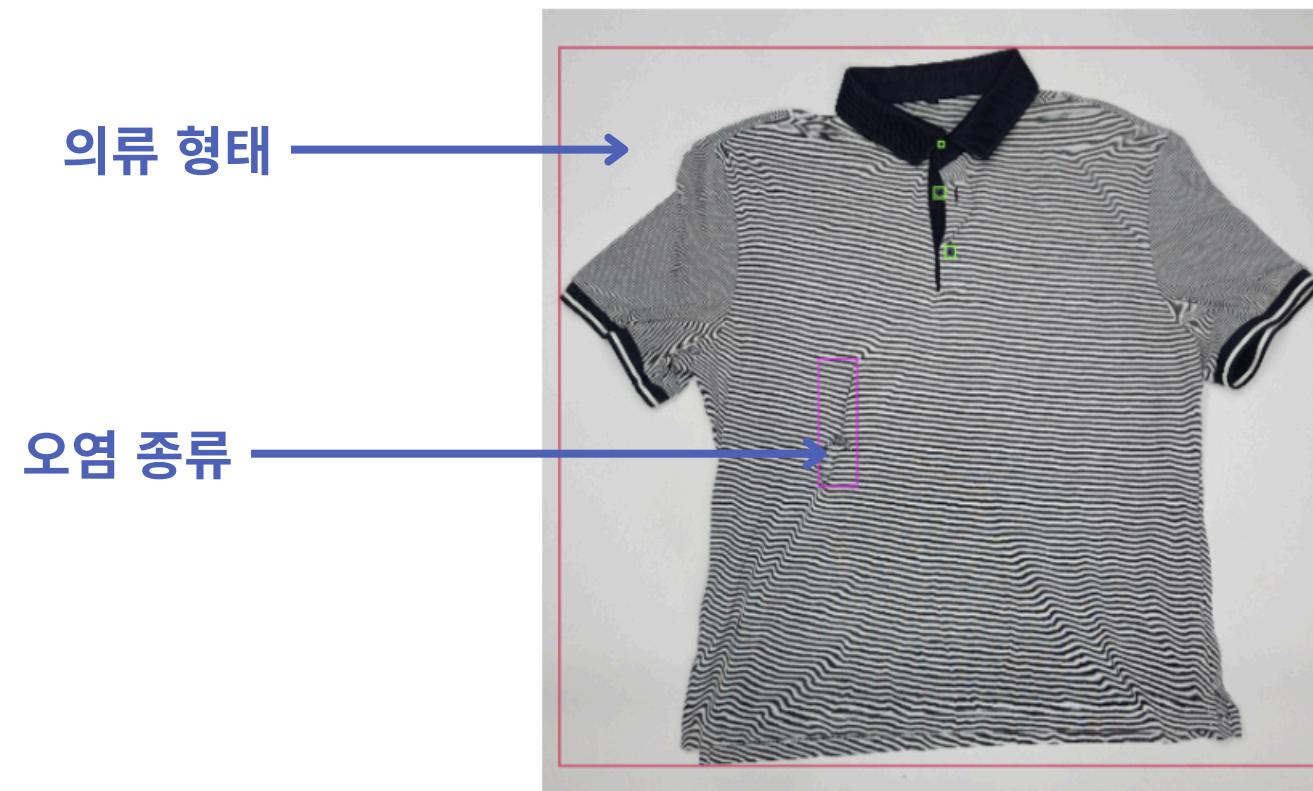
- **데이터 분류 기준 :**
 - **오염 종류** → 재사용 / 재활용 / 폐기로 분류
 - **의류 형태** → 총 11종

3-1 데이터 수집

사용한 데이터 범위

- **문제**: 전체 26만장의 데이터는 용량이 크고 의류 카테고리가 다양하여 전체 학습에는 한계가 존재
→ 본 프로젝트에서는 데이터 처리 효율과 모델 정확도 향상을 위해 **셔츠형 상의(Blouse + Shirts)** 만을 진행

- **선택 이유**
 - 비교적 이미지 수가 많고 균질함
 - 향후 다양한 의류로 확장 시 대표성 존재



[재활용 의류 예시]

폐의류 이미지 바운딩박스 라벨링 예시

3-2 데이터 적재 : Challenge

데이터 다운로드 초기 시도

- AI Hub의 학습용 데이터를 CLI 환경에서 다운로드할 수 있는 도구 aihubshell 사용 시도
- Google Cloud VM 인스턴스 내에서 .zip 파일 형태로 다운로드 진행 시도

발생한 문제

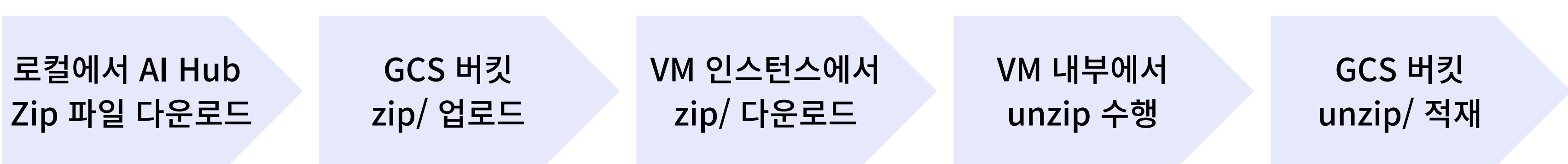
- Google Cloud VM (asia-northeast3)는 한국 리전임에도 AI Hub 측에서 해외 IP로 오인
→ 다운로드 차단 (502 Bad Gateway error) 발생

해결 방법

- 우회 전략 수립 및 적용

3-2 데이터 적재 : Challenge

해결 방법



Google Cloud CLI 기반 데이터 업로드 과정

```
# GCS 버킷에서 라벨링데이터 폴더 전체를 로컬 ~/Training 폴더로 복사  
gsutil -m cp -r gs://deeplearn-bucket-1/Training/라벨링데이터 ~/Training/ ← VM 인스턴스로 다운로드
```

```
# unzip한 뒤, GCS 버킷에 재업로드  
cd ~/Training/라벨링데이터  
for ZIPFILE in *.zip; do  
    FOLDER="${ZIPFILE%.zip}_unzipped"  
    unzip -o "$ZIPFILE" -d "$FOLDER" ← 압축 해제  
    gsutil -m cp -r "$FOLDER" "gs://deeplearn-bucket-1/Training/라벨링데이터/$FOLDER/" ← GCS로 재업로드  
done
```

3-2 데이터 적재

적재한 데이터 리스트

원천 이미지	dispose	TS_task1_yolo Dispose_BLOUSE.zip	951.77 MB	라벨링 데이터	dispose	TL_task1_yolo Dispose_BLOUSE.zip	301.50 KB
		TS_task1_yolo Dispose_SHIRTS.zip	1.29 GB			TL_task1_yolo Dispose_SHIRTS.zip	363.20 KB
		TS_task1_yolo recycle_BLOUSE.zip	2.67 GB		recycle	TL_task1_yolo recycle_BLOUSE.zip	882.49 KB
	recycle	TS_task1_yolo recycle_SHIRTS.zip	3.45 GB			TL_task1_yolo recycle_SHIRTS.zip	1.02 MB
		TS_task1_yolo reusable_BLOUSE.zip	4.50 GB		reusable	TL_task1_yolo reusable_BLOUSE.zip	1.19 MB
	reusable	TS_task1_yolo reusable_SHIRTS.zip	9.49 GB			TL_task1_yolo reusable_SHIRTS.zip	2.39 MB

3-2 데이터 적재

GCS 버킷 내 구조

```
gs://deeplearn-bucket-1/
└── Training/
    ├── 원천데이터/
    │   ├── dispose/
    │   │   ├── BLOUSE/
    │   │   └── SHIRTS/
    │   ├── recycle/
    │   │   ├── BLOUSE/
    │   │   └── SHIRTS/
    └── reusable/
        ├── BLOUSE/
        └── SHIRTS/
    └── 라벨데이터/
        └── (동일 구조로 파일 존재)
```

버킷 > deeplearn-bucket-1 > Training > 01.원천데이터

폴더 만들기 업로드 데이터 이전 기타 서비스

이름 프리픽스로만 필터링 ▾ 필터 객체 및 폴더 필터링

이름	크기	유형	생성
TS_task1_yolo_dispose_BLOUSE_...	-	폴더	-
TS_task1_yolo_dispose_SHIRTS_u...	-	폴더	-
TS_task1_yolo_recycle_BLOUSE_u...	-	폴더	-
TS_task1_yolo_recycle_SHIRTS_un...	-	폴더	-
TS_task1_yolo_reusable_BLOUSE_...	-	폴더	-
TS_task1_yolo_reusable_SHIRTS_...	-	폴더	-

폴더 내부

이름	크기	유형
DP_BL_00001.jpg	3.3MB	image/jpeg
DP_BL_00004.jpg	1.7MB	image/jpeg

⋮

→ 안정적인 구조, 모델 학습 시 빠른 접근 속도 제공

3-2 데이터 적재

이미지 데이터 개수 (.png)

[1] Training/01.원천데이터/TS_task1_yolo Dispose_BLOUSE_unzipped/
총 이미지 개수: 438 (jpg: 438, png: 0)

[2] Training/01.원천데이터/TS_task1_yolo Dispose_SHIRTS_unzipped/
총 이미지 개수: 520 (jpg: 520, png: 0)

[3] Training/01.원천데이터/TS_task1_yolo recycle_BLOUSE_unzipped/
총 이미지 개수: 1184 (jpg: 1184, png: 0)

[4] Training/01.원천데이터/TS_task1_yolo recycle_SHIRTS_unzipped/
총 이미지 개수: 1198 (jpg: 1198, png: 0)

[5] Training/01.원천데이터/TS_task1_yolo reusable_BLOUSE_unzipped
총 이미지 개수: 2074 (jpg: 2074, png: 0)

[6] Training/01.원천데이터/TS_task1_yolo reusable_SHIRTS_unzipped
총 이미지 개수: 4142 (jpg: 4142, png: 0)

라벨 데이터 개수 (.json)

[1] Training/02.라벨링데이터/TL_task1_yolo Dispose_BLOUSE_unzipped/
총 JSON 파일 개수: 438

[2] Training/02.라벨링데이터/TL_task1_yolo Dispose_SHIRTS_unzipped/
총 JSON 파일 개수: 520

[3] Training/02.라벨링데이터/TL_task1_yolo recycle_BLOUSE_unzipped/
총 JSON 파일 개수: 1184

[4] Training/02.라벨링데이터/TL_task1_yolo recycle_SHIRTS_unzipped/
총 JSON 파일 개수: 1198

[5] Training/02.라벨링데이터/TL_task1_yolo reusable_BLOUSE_unzipped
총 JSON 파일 개수: 2074

[6] Training/02.라벨링데이터/TL_task1_yolo reusable_SHIRTS_unzipped
총 JSON 파일 개수: 4142

→ 총 9,556장, 이미지와 라벨이 일대일 대응

04 Object Detection: YOLOv8

04 Object Detection: Yolov8

YOLOv8 개요

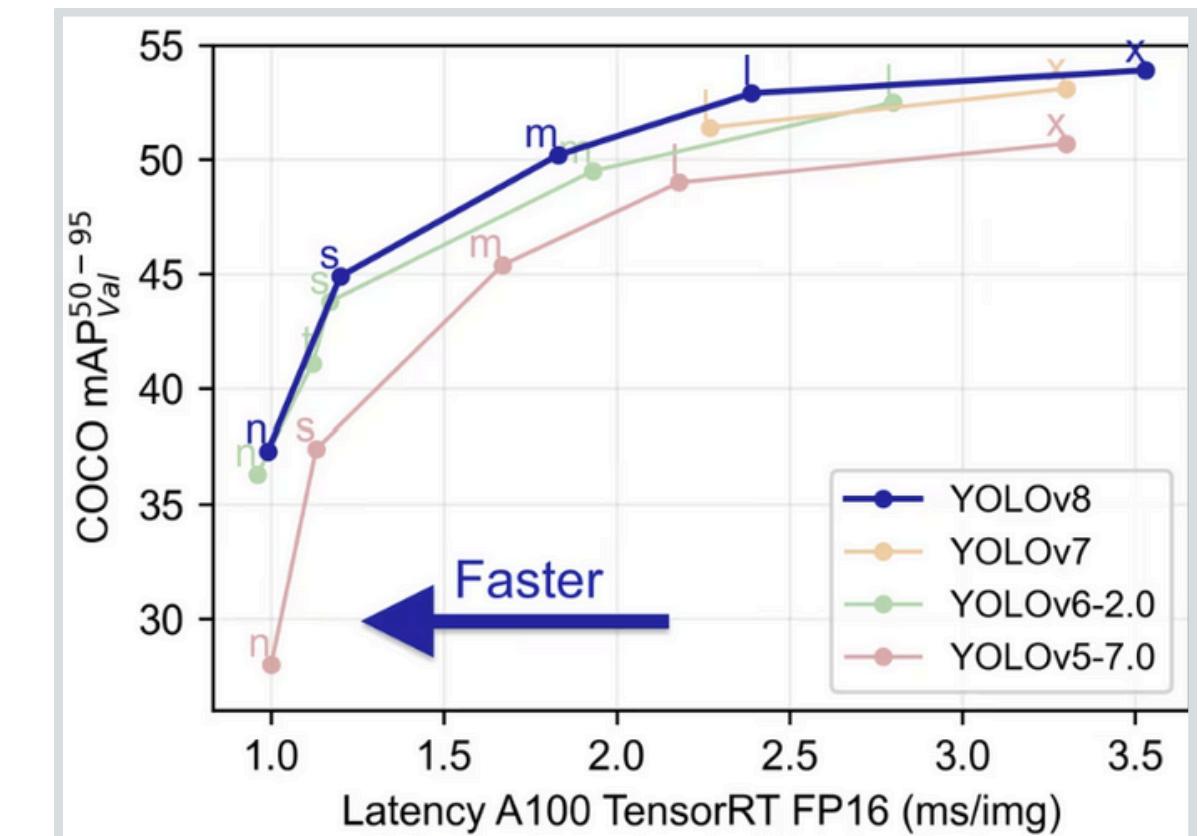
- Ultralytics에서 개발한 최신 Object Detection 모델
- 딥러닝 기술 기반의 속도-정확도 최적화 모델
- 간결한 설계로 다양한 응용 분야에 적합

→ 본 프로젝트에서는,

"객체가 포함된 이미지 + 바운딩 박스"로 구성된 데이터셋을 기반으로

YOLOv8n 모델을 의류 형태 및 오염 탐지 목적에 맞게 fine-tuning하여 사용함

→ 탐지된 오염 종류에 따라 클래스 분류 (재사용 / 재활용 / 폐기)



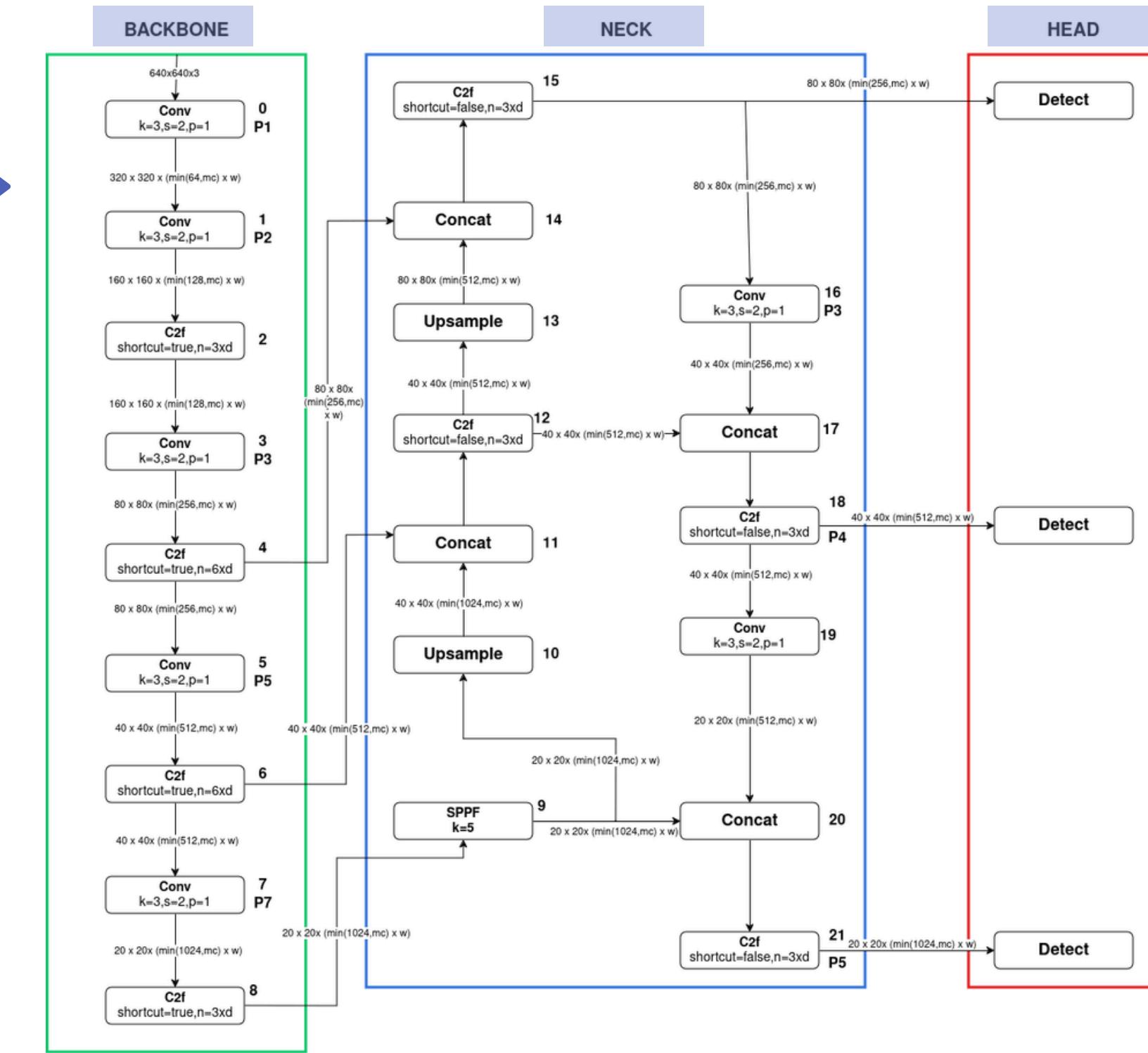
고속·고정확 YOLOv8 모델

04 Object Detection: Yolov8

YOLOv8 구조

Backbone + Neck + Head 구조 →

- **Backbone:** 특징 추출 (CSPDarknet)
- **Neck:** 특징 통합 (FPN+PAN)
- **Head:** 클래스 및 위치 예측



YOLOv8 architecture

04 YOLOv8: 학습 전 준비과정

1. 데이터 수집

- GCS에서 이미지 + JSON 라벨 다운로드
- 파일 매핑: 이미지 ↔ 라벨 연결 확인

2. .JSON → .txt 변환

- YOLO format: class_id x_center y_center width height (normalized)
- 좌표 정규화: 0~1 범위로 변환

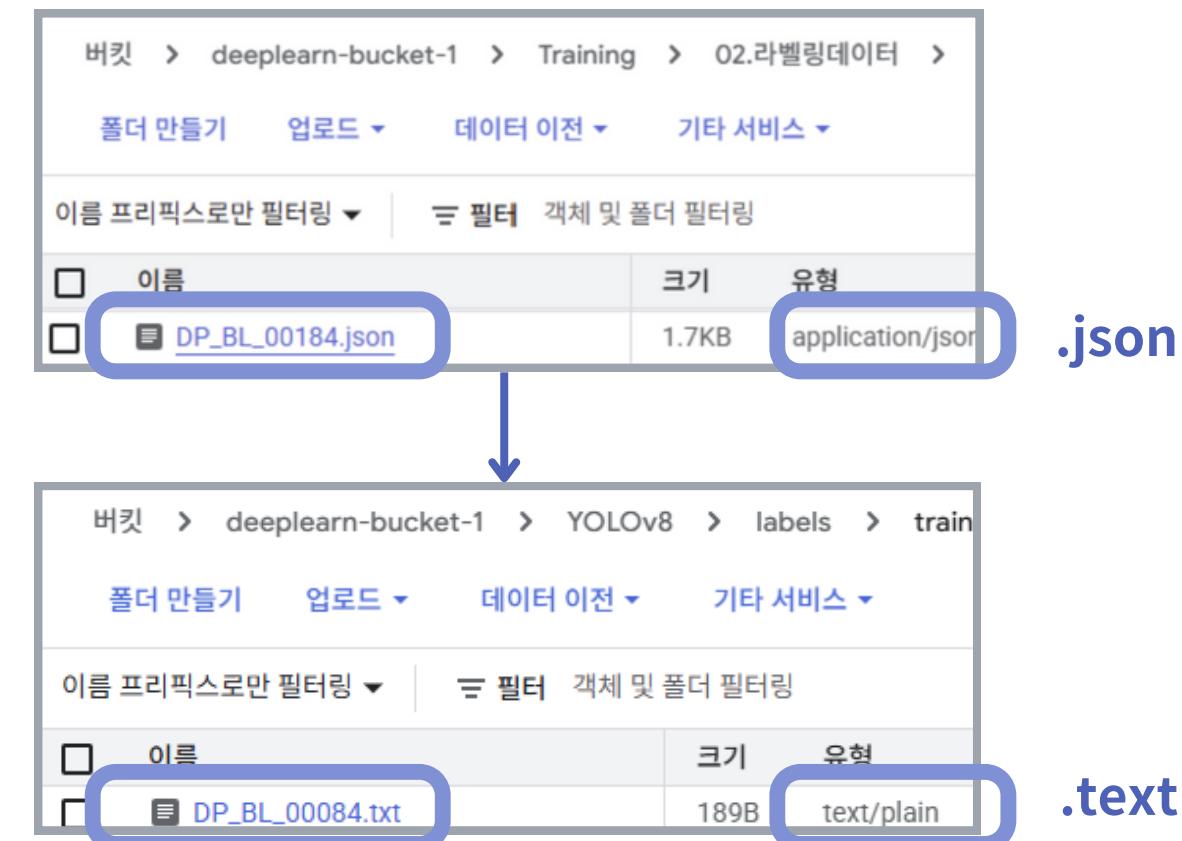
```
def convert_label(json_blob_path, save_txt_path, class_id=0):
    with fs.open(json_blob_path, 'r') as f:
        data = json.load(f)

    annotations = data.get("annotations", [])
    if not annotations:
        return

    txt_lines = []

    for ann in annotations:
        bbox = ann.get("bbox", [])
        if len(bbox) == 4:
            x, y, w, h = bbox
            x_center = x + w / 2
            y_center = y + h / 2
            txt_line = f"{class_id} {x_center:.6f} {y_center:.6f} {w:.6f} {h:.6f}"
            txt_lines.append(txt_line)

    with open(save_txt_path, "w") as out:
        out.write("\n".join(txt_lines))
```



04 YOLOv8: 학습 전 준비과정

3. 클래스 맵핑

- 6개 클래스 (dispose/recycle/reusable + blouse/shirts)에 숫자 ID 부여

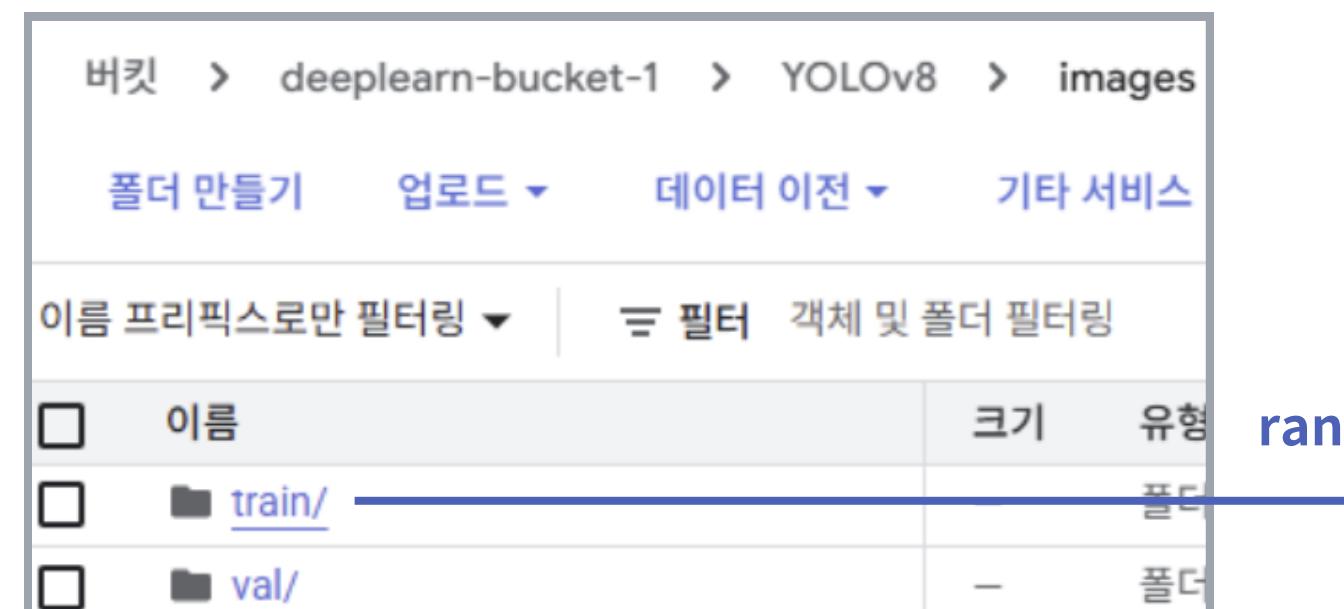
```
# 카테고리 목록
categories = [
    ("dispose_BLOUSE", 0),
    ("dispose_SHIRTS", 1),
    ("recycle_BLOUSE", 2),
    ("recycle_SHIRTS", 3),
    ("reusable_BLOUSE", 4),
    ("reusable_SHIRTS", 5)
]
```

4. 데이터 분할

- train 80%, validation 20%

→ **data.yaml** 생성: 데이터 경로 + 클래스 정보

```
from sklearn.model_selection import train_test_split
train_imgs, val_imgs = train_test_split(image_paths, test_size=0.2, random_state=42)
```



data.yaml: train/val path 설정

```
path: /content/drive/MyDrive/Colab Notebooks/yolo_dataset
train: images/train
val: images/val

names:
  0: dispose_BLOUSE
  1: dispose_SHIRTS
  2: recycle_BLOUSE
  3: recycle_SHIRTS
  4: reusable_BLOUSE
  5: reusable_SHIRTS
```

04 YOLOv8: 1차 파인튜닝 학습

모델 학습 코드

```
model = YOLO("yolov8n.pt") # 경량 모델

model.train(
    data=yaml_path,
    epochs=10, # 학습 반복 횟수
    imgsz=320, # 입력 이미지 크기 (320x320)
    batch=64, # 한 번에 처리할 이미지 수
    workers=4,
    device=0,
    project=os.path.join(root_dir, "runs"),
    name="yolov8_train",
    exist_ok=True,
    save_period=1,
    amp=True,
    pretrained=True,
    patience=10, # early stopping 조건 (10 epoch 동안 성능 개선 없으면 중단)
    cache=True
)
```

`train`: Caching images (1.6GB RAM): 100%|██████████| 7628/7628

1차 학습 결과 평가 (validation)

```
metrics = model.val() # validation set에서 평가
print(metrics.box.map) # box mAP 출력
```

Class	Images	Instances	Box(P)		R	mAP50	mAP50-95
			P	R			
all	1907	2323	0.627	0.652	0.653	0.641	
dispose_BLOUSE	90	278	0.503	0.186	0.225	0.185	
dispose_SHIRTS	104	332	0.48	0.181	0.218	0.188	
recycle_BLOUSE	241	241	0.599	0.925	0.766	0.764	
recycle_SHIRTS	240	240	0.705	0.75	0.825	0.825	
reusable_BLOUSE	416	416	0.554	0.974	0.924	0.922	
reusable SHIRTS	816	816	0.923	0.899	0.961	0.961	

- 가장 빠른 학습 시간
- recycle/reuse 클래스는 성능 우수
- dispose 클래스는 낮은 Recall 및 mAP 성능

→ 이미지 해상도 및 epoch 수 향상 필요성 시사

04 YOLOv8: 성능 개선 시도

파라미터	1차 시도	2차 시도	3차 시도
모델	YOLOv8n		
이미지 크기	320	416	640
배치 크기	64		
epoch	10	40	40
Patience	10	15	10

2차 학습 결과 평가 (validation)

val: Caching images (0.7GB RAM): 100% [██████████] 1907/1907 [00:33<00:00, 56.25it/s]						
Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	1907	2323	0.749	0.713	0.74	0.717
dispose_BLOUSE	90	278	0.571	0.311	0.369	0.3
dispose_SHIRTS	104	332	0.632	0.328	0.37	0.301
recycle_BLOUSE	241	241	0.754	0.9	0.882	0.88
recycle_SHIRTS	240	240	0.798	0.854	0.892	0.892
reusable_BLOUSE	416	416	0.823	0.923	0.954	0.954
reusable_SHIRTS	816	816	0.913	0.96	0.975	0.975

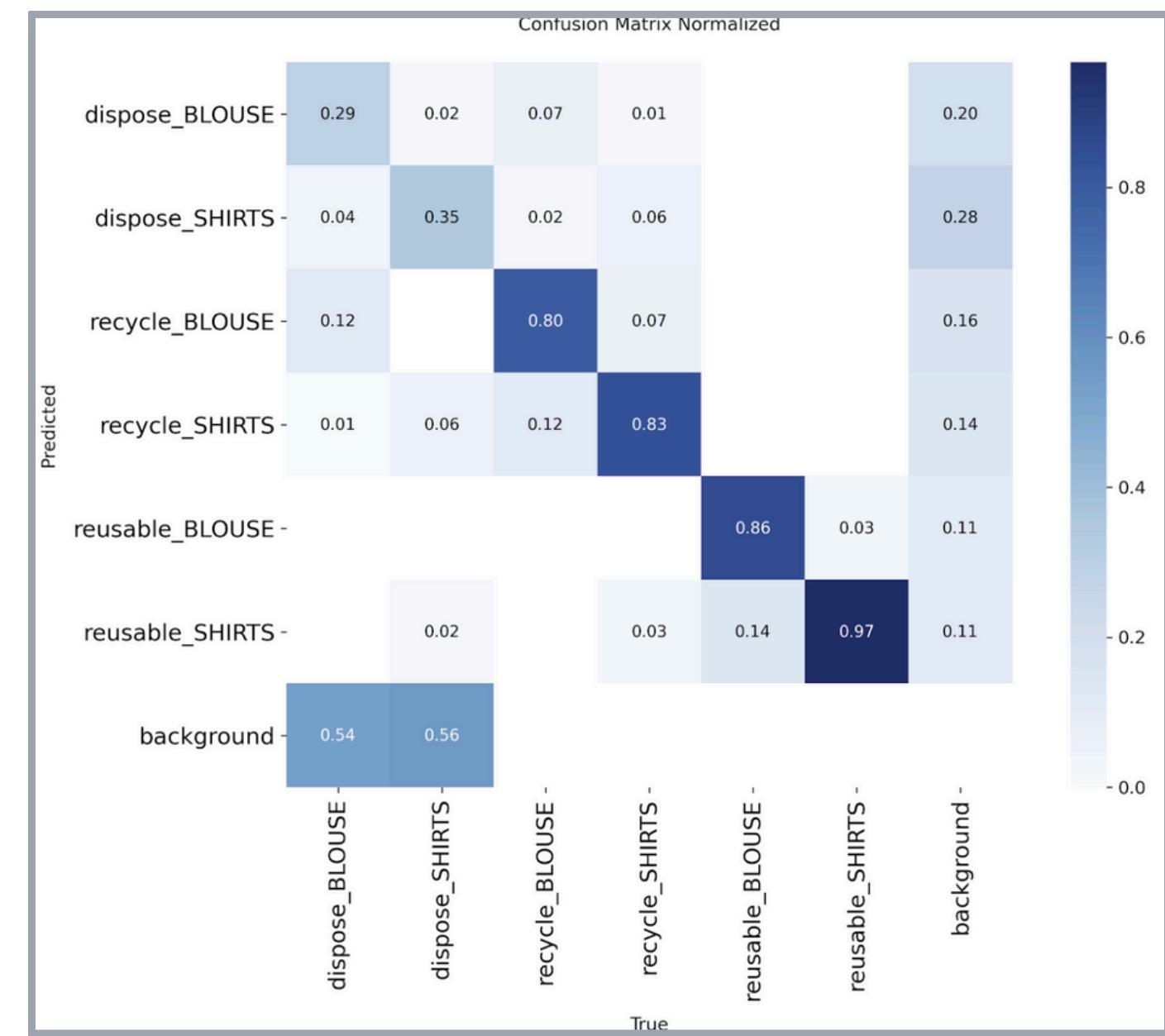
- 1차 대비 전반적인 성능 향상 확인됨
- recycle/reuse 클래스는 성능 매우 우수
- dispose 클래스는 성능 개선이 됐지만, 여전히 낮은 Recall 및 mAP 성능
→ 이미지 해상도 및 epoch 수 향상 필요성 시사

04 YOLOv8: 최종 모델

최종 모델 성능 평가 (validation)

val: Caching images (1.6GB RAM): 100% [██████████] 1907/1907 [00:35<00:00, 53.10it/s]						
Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	1907	2323	0.752	0.718	0.752	0.723
dispose_BLOUSE	90	278	0.623	0.32	0.401	0.301
dispose_SHIRTS	104	332	0.565	0.337	0.385	0.314
recycle_BLOUSE	241	241	0.755	0.888	0.889	0.887
recycle_SHIRTS	240	240	0.755	0.887	0.904	0.904
reusable_BLOUSE	416	416	0.906	0.883	0.964	0.964
reusable_SHIRTS	816	816	0.906	0.99	0.967	0.967

- 2차 대비 향상된 정밀도와 평균 정밀도 (mAP)
- recycle/reuse 클래스는 0.96 이상 정밀도 유지
- dispose 클래스는 여전히 낮은 Recall 및 mAP이지만, 이전 대비 대선
→ 해상도 640 적용으로, 미세 객체 탐지력 향상 효과



04 Yolov8: Prediction 결과

train_batch.jpg

- 학습용 이미지 배치 시각화
- 학습 중 사용된 이미지 일부와 그 위에 그려진 GT (Ground Truth) 바운딩 박스
- 라벨링 품질이 정상인지 확인 용도



```
# 카테고리 목록  
categories = [  
    ("dispose_BLOUSE", 0),  
    ("dispose_SHIRTS", 1),  
    ("recycle_BLOUSE", 2),  
    ("recycle_SHIRTS", 3),  
    ("reusable_BLOUSE", 4),  
    ("reusable_SHIRTS", 5)  
]
```

dispose_SHIRTS로 올바르게 라벨링된 것을 확인 가능

val_batch_pred.jpg

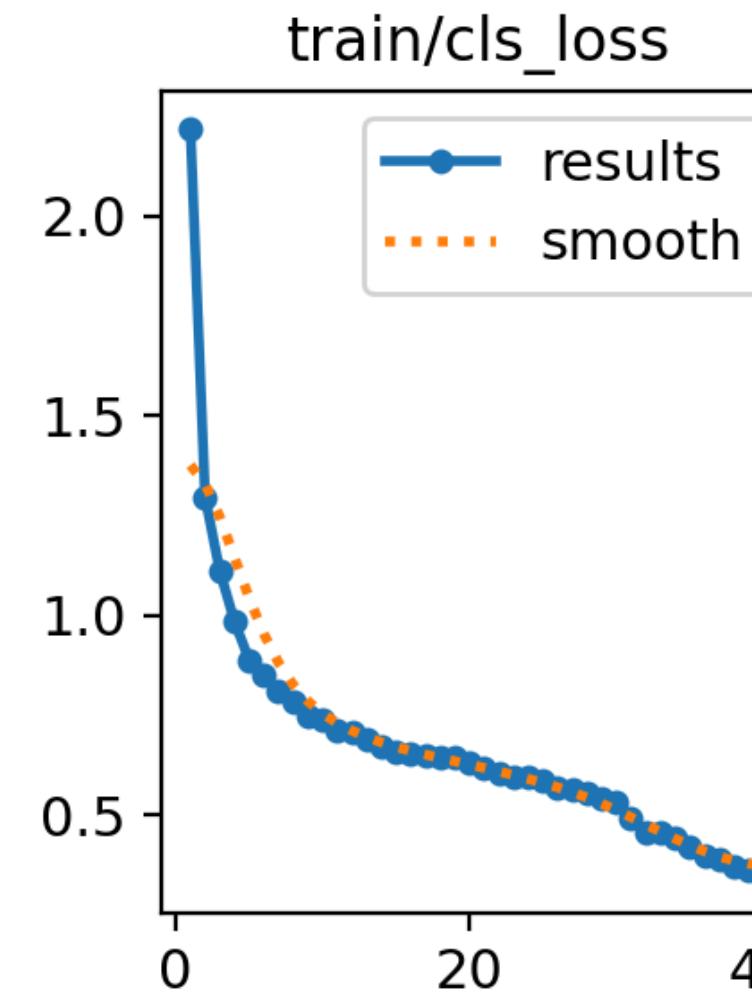
- 검증 배치 예측 시각화
- 검증 데이터셋에서 모델이 예측한 결과
→ 이미지 위에 예측된 바운딩 박스 + 클래스명 + confidence score
- mAP 수치로는 보기 어려운 오분류의 종류나 양상 파악 가능



패턴이 복잡하거나 화려한 옷의 경우,
오염 물질의 시각적 경계가 흐려져 탐지 정확도 저하

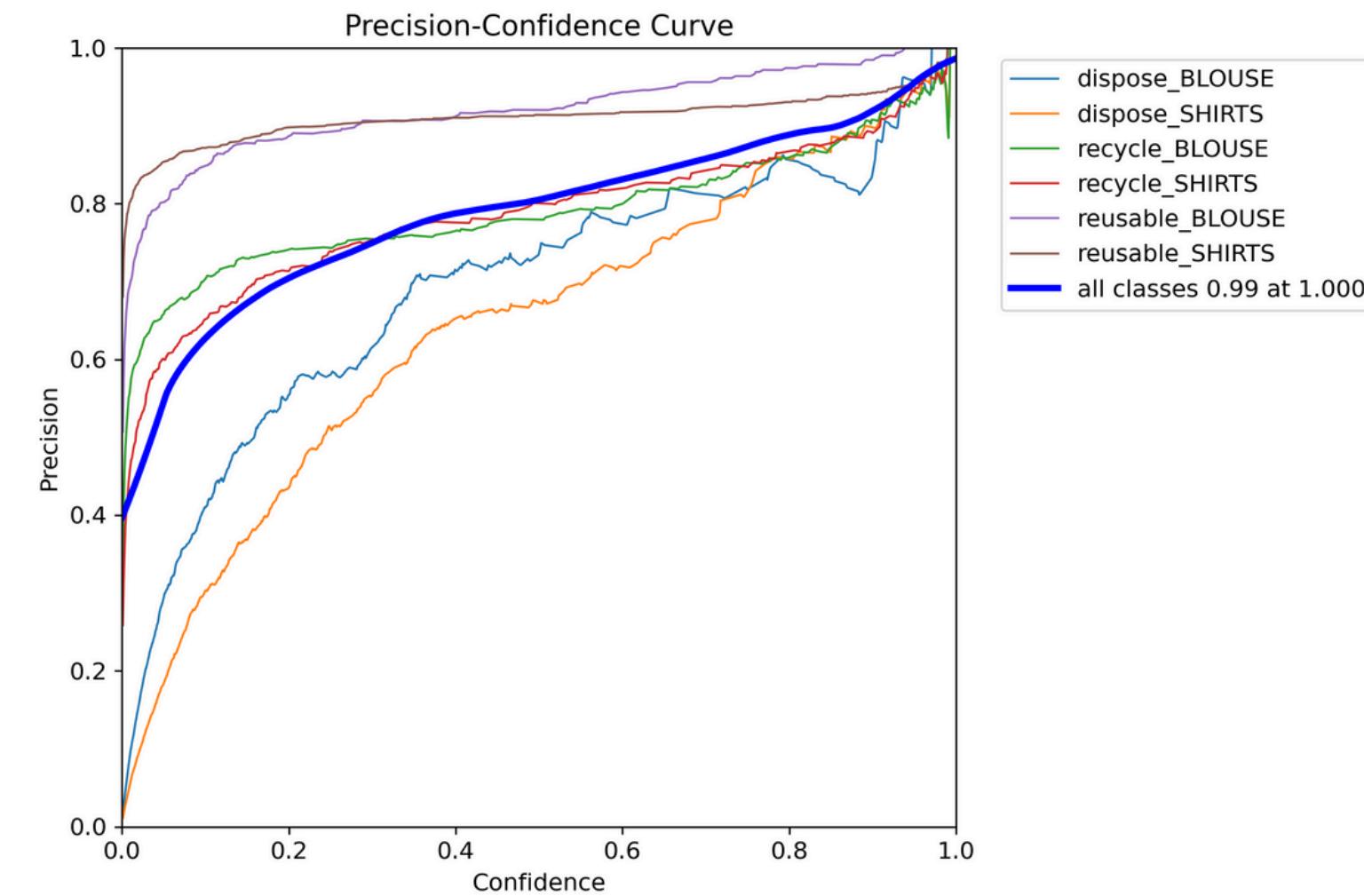
04 Yolov8: 성능

train/cls_loss



- cls_loss가 지속적으로 감소 양상
→ 모델이 점차 정확한 클래스 예측 학습중

Precision-Confidence Curve



- dispose 제외 곡선이 대체로 높게 유지됨
→ 높은 confidence를 가진 예측은 실제로도 정확함

05 Image Captioning : BLIP

05 Image Captioning : BLIP

BLIP : Bootstrapping Language-Image Pretraining

- Salesforce Research에서 제안한 멀티모달 비전 모델
- 텍스트-이미지 동시 이해, 생성에 강력한 성능
- 이미지 캡쳐닝, VQA (시각 질의 응답), 텍스트 기반 이미지 검색에 특화

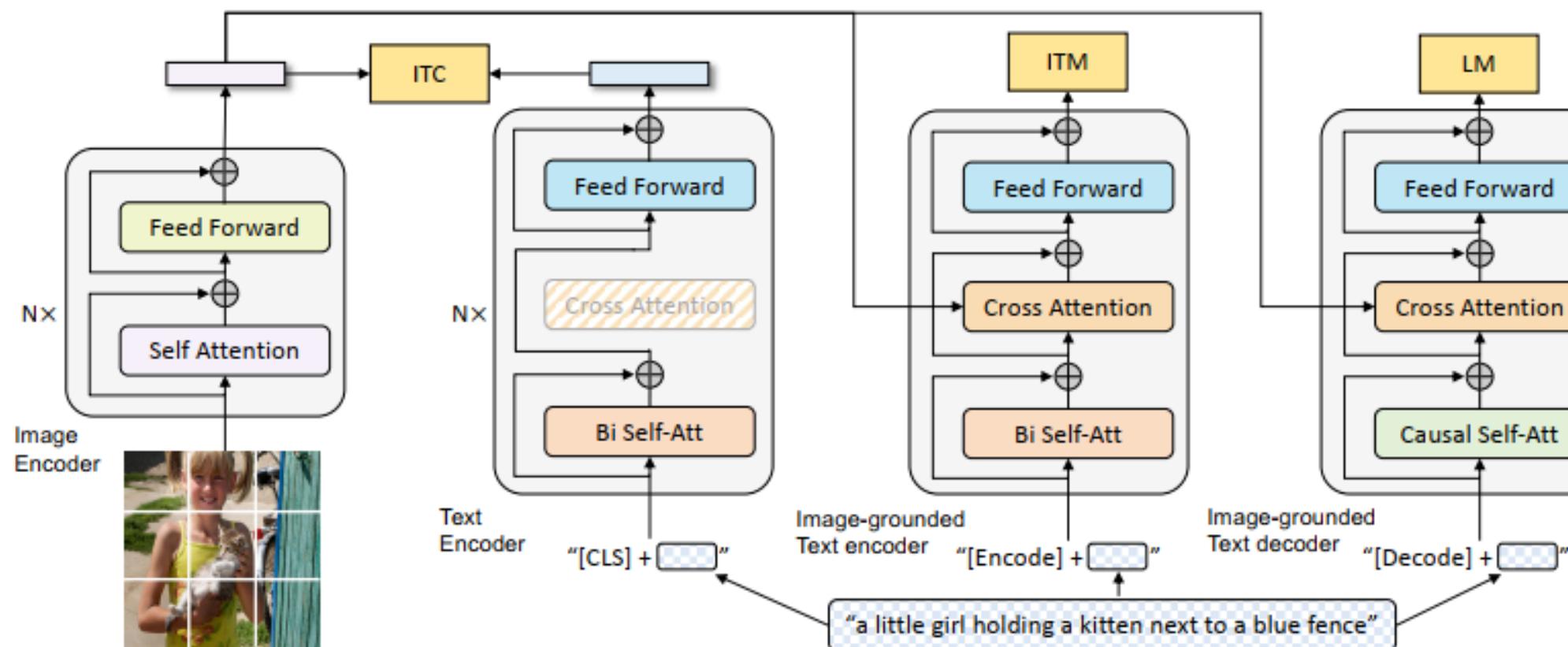
본 프로젝트에서는,

“이미지 + 캡션”으로 구성된 데이터셋 → 학습 → “새로운 이미지” 입력시 적절한 캡션 생성을 위해,
BLIP 모델을 한국어 기반 + 의류 도메인에 맞춰 fine-tuning하여 사용함.

05 Image Captioning : BLIP

BLIP 구조

- Image Encoder + Text Encoder / Decoder
- 이미지와 텍스트 데이터를 각각 ViT, BERT를 이용해 인코딩, transformer 기반 텍스트 decoder



05 Image Captioning : BLIP

1. GCS 로드 및 JSONL 파일 변환

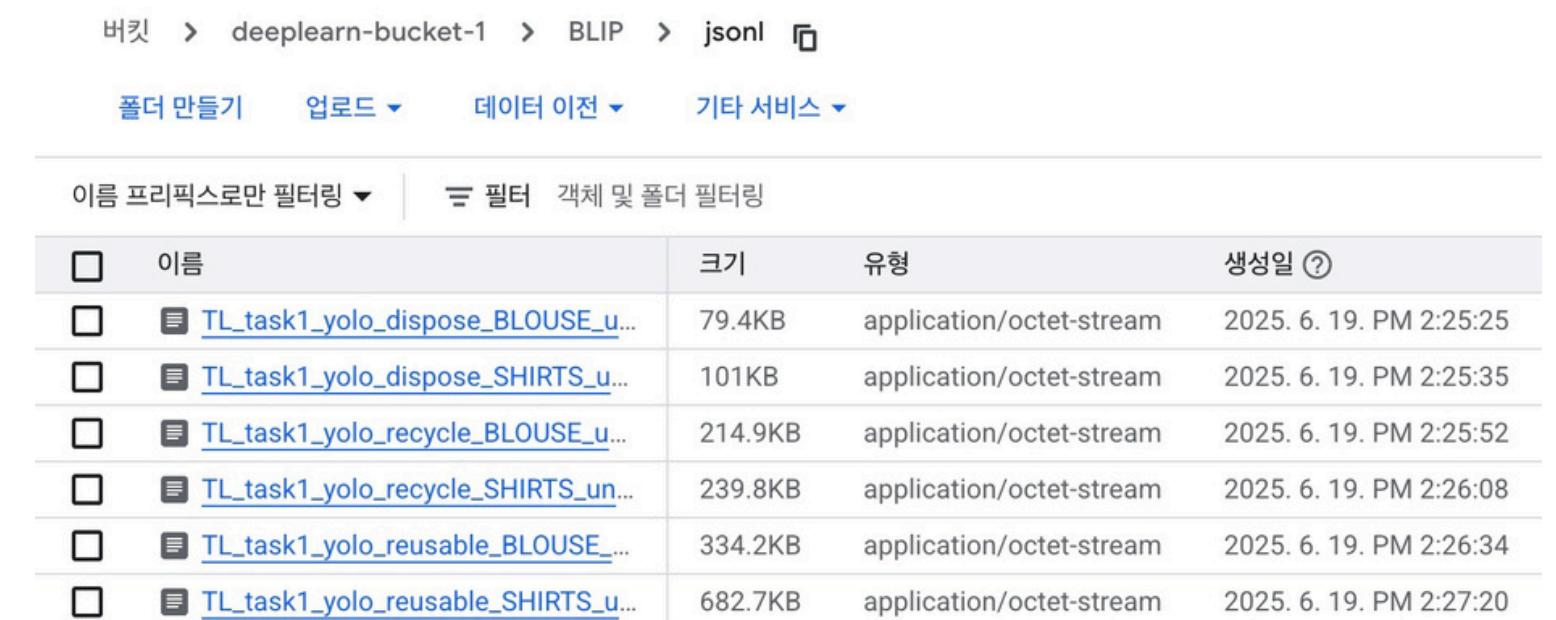
```
# ----- 전처리 및 저장 루프 -----
for prefix in prefixes:
    dir_name = prefix.split("/")[-1]
    local_dir = os.path.join(local_base, dir_name)
    os.makedirs(local_dir, exist_ok=True)

    # 1. GCS → Colab 다운로드
    gcs_full_path = f"gs://{bucket}/{prefix}/*.json"
    print(f"⬇️ Downloading from {gcs_full_path} ...")
    !gsutil -m cp $gcs_full_path $local_dir/

    # 2. JSON → JSONL로 변환
    jsonl_lines = []
    json_files = sorted(glob(os.path.join(local_dir, "*.json")))
    for path in json_files:
        try:
            with open(path, "r", encoding="utf-8") as f:
                data = json.load(f)
            file_name = data.get("meta_information", {}).get("file_name")
            caption = data.get("image_caption", {}).get("caption")
            if file_name and caption:
                jsonl_lines.append(json.dumps({"image": file_name, "caption": caption}, ensure_ascii=False))
        except Exception as e:
            print(f"⚠️ {os.path.basename(path)} 처리 실패: {e}")

    # 3. 로컬에 jsonl 저장
    jsonl_filename = f"{dir_name}.jsonl"
    jsonl_path = os.path.join(local_base, jsonl_filename)
    with open(jsonl_path, "w", encoding="utf-8") as f:
        f.write("\n".join(jsonl_lines))
    print(f"✅ {jsonl_filename} 저장 완료 ({len(jsonl_lines)}개 항목)")
```

각 prefix 안의 .json 파일 모두 로드
.json 파일에서 “file_name”과 “caption” 정보 추출
그 정보를 모아 하나의 .jsonl 파일 생성



The screenshot shows a Google Cloud Storage interface with the following details:

- Bucket: deeapplearn-bucket-1
- Folder: BLIP/jsonl
- File List:

이름	크기	유형	생성일
TL_task1_yolo_dispose_BLOUSE_u...	79.4KB	application/octet-stream	2025. 6. 19. PM 2:25:25
TL_task1_yolo_dispose_SHIRTS_u...	101KB	application/octet-stream	2025. 6. 19. PM 2:25:35
TL_task1_yolo_recycle_BLOUSE_u...	214.9KB	application/octet-stream	2025. 6. 19. PM 2:25:52
TL_task1_yolo_recycle_SHIRTS_u...	239.8KB	application/octet-stream	2025. 6. 19. PM 2:26:08
TL_task1_yolo_reusable_BLOUSE_...	334.2KB	application/octet-stream	2025. 6. 19. PM 2:26:34
TL_task1_yolo_reusable_SHIRTS_u...	682.7KB	application/octet-stream	2025. 6. 19. PM 2:27:20

GCS에 저장된 .jsonl 파일

prefix별 .jsonl 파일

dispose_BLOUSE	dispose_SHIRTS
recycle_BLOUSE	recycle_SHIRTS
reuse_BLOUSE	reuse_SHIRTS

05 Image Captioning : BLIP

2. BLIP 학습을 위한 이미지-텍스트 기반 데이터 전처리

```
# 전처리 함수 정의
def transform(example):
    image_path = os.path.join(image_dir, example["image"])
    try:
        image = Image.open(image_path).convert("RGB")
    except:
        image = Image.new("RGB", (224, 224))
    inputs = processor(images=image, text=example["caption"], return_tensors="pt", padding="max_length")
    return {
        "input_ids": inputs.input_ids[0],
        "attention_mask": inputs.attention_mask[0],
        "pixel_values": inputs.pixel_values[0],
        "labels": inputs.input_ids[0]
    }

# 전처리 적용
dataset = dataset.map(transform, remove_columns=["image", "caption"])

dataset.save_to_disk("/content/preprocessed_blip_dataset")
print("▣ 전처리된 데이터셋 저장 완료")

전체 샘플 수: 9556
/usr/local/lib/python3.11/dist-packages/datasets/table.py:1395: FutureWarning: promote has been super
    block_group = [InMemoryTable(cls._concat_blocks(list(block_group), axis=axis))]
/usr/local/lib/python3.11/dist-packages/datasets/table.py:1421: FutureWarning: promote has been super
    table = cls._concat_blocks(blocks, axis=0)
Map: 100% 9556/9556 [44:36<00:00, 3.42 examples/s]
Saving the dataset (34/34 shards): 100% 9556/9556 [00:48<00:00, 165.32 examples/s]
▣ 전처리된 데이터셋 저장 완료
```

BLIP 학습용 전처리 데이터셋 저장

버킷 > deeplearn-bucket-1 > BLIP > preprocessed_blip_dataset		
폴더 만들기 업로드 데이터 이전 기타 서비스		
이름 프리픽스로만 필터링 ▾ □ 필터 객체 및 폴더 필터링		
□	이름	크기 유형
□	data-00000-of-00034.arrow	500.8MB application/vnd.apache.arrow.file
□	data-00001-of-00034.arrow	500.8MB application/vnd.apache.arrow.file
□	data-00002-of-00034.arrow	499MB application/vnd.apache.arrow.file
□	data-00003-of-00034.arrow	499MB application/vnd.apache.arrow.file
□	data-00004-of-00034.arrow	499MB application/vnd.apache.arrow.file
□	data-00005-of-00034.arrow	499MB application/vnd.apache.arrow.file
□	data-00006-of-00034.arrow	499MB application/vnd.apache.arrow.file
□	data-00007-of-00034.arrow	499MB application/vnd.apache.arrow.file

GCS의 deeplearn-bucket-1/
BLIP/preprocessed_blip_dataset/에 저장

05 Image Captioning : BLIP

3. Trainer 기반 모델 fine-tuning

1차 시도

```
training_args = TrainingArguments(  
    output_dir='./blip_finetuned',  
    per_device_train_batch_size=8,  
    per_device_eval_batch_size=8,  
    num_train_epochs=3,  
    eval_strategy="epoch",  
    save_strategy="epoch",  
    logging_dir='./logs',  
    logging_steps=10,  
    learning_rate=5e-5,  
    remove_unused_columns=False,  
    fp16=True,  
    report_to="none",  
    load_best_model_at_end=True,  
    metric_for_best_model="eval_loss"  
)
```

7. Trainer 정의

```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=train,  
    eval_dataset=eval)
```

8. 학습 시작

```
trainer.train()
```

epoch의 흐름에 따라
validation loss도 감소 양상



2차 시도 (파라미터 변경)

```
training_args = TrainingArguments(  
    output_dir='./blip_finetuned_v2',  
    per_device_train_batch_size=8,  
    per_device_eval_batch_size=8,  
    num_train_epochs=5, # 증가  
    eval_strategy="epoch",  
    save_strategy="epoch",  
    logging_dir='./logs',  
    logging_steps=10,  
    learning_rate=3e-5, # 감소  
    gradient_accumulation_steps=2, # 추가  
    remove_unused_columns=False,  
    fp16=True,  
    report_to="none",  
    load_best_model_at_end=True,  
    metric_for_best_model="eval_loss")
```

Trainer 정의

```
trainer = Trainer(  
    model=mode
```

```
    args=args
```

```
    train_data
```

```
    eval_datas
```

```
)
```

학습 시작

```
trainer.train()
```

Epoch	Training Loss	Validation Loss
1	0.126500	0.157299
2	0.099500	0.157422
3	0.072100	0.160317
4	0.050100	0.168785
5	0.037200	0.175350

num_train_epochs 증가
learning_rate 감소
gradient_accumulation_step 추가

epoch의 흐름에 따라
validation loss 지속적 증가
(높은 가능성 과적합)

05 Image Captioning : BLIP

4. 테스트 데이터에 대한 캡션 생성 (예시)

```
# 실제 캡션 가져오기
with open(jsonl_path, "r", encoding="utf-8") as f:
    for line in f:
        data = json.loads(line)
        if data["image"] == target_image:
            real_caption = data["caption"]
            break

    if real_caption is None:
        raise ValueError(f"{target_image}에 대한 캡션을 찾을 수 없습니다.")

# 이미지 로딩 및 모델 입력
image_path = f"/content/blip_images/{target_image}"
image = Image.open(image_path).convert("RGB")
inputs = processor(images=image, return_tensors="pt").to(model.device)

# 캡션 생성
with torch.no_grad():
    generated_ids = model.generate(**inputs, max_length=128)
    predicted_caption = processor.decode(generated_ids[0], skip_special_tokens=True)

# 이미지 먼저 출력
plt.figure(figsize=(6, 6))
plt.imshow(image)
plt.axis("off")
plt.show()

# 예측 및 실제 캡션 출력
print("예측된 캡션:", predicted_caption)
print("실제 캡션:", real_caption)
```



예측된 캡션: 흰색과 스트라이프 패턴이 중앙에 여성용 반팔 블라우스
실제 캡션: 우측 중앙에 외부오염이 있는 하늘색 여성용 반팔 드레스

예측된 캡션: 카라가 단추가 중앙이 흰색의 남녀 공용 긴팔 셔츠
실제 캡션: 카라가 있고 소매에 단추가 있으며 중앙에 외부오염이 있는 아이보리색 여성용 긴팔 블라우스

05 Image Captioning : BLIP

디렉토리 구조

The screenshot shows a cloud storage interface with the following directory structure:

- ..
- blip_finetuned
- blip_finetuned_v2
- blip_images
- blip_jsonl
- preprocessed_blip_dataset
- sample_data
- service_key.json
- blip_finetuned/ (expanded)
 - checkpoint-1075
 - config.json
 - generation_config.json
 - model.safetensors
 - optimizer.pt
 - rng_state.pth
 - scaler.pt
 - scheduler.pt
 - trainer_state.json
 - training_args.bin
 - checkpoint-2150

체크포인트별 학습 저장
다음 업데이트하는 방식

버킷 > deeplearn-bucket-1 > BLIP

폴더 만들기 업로드 데이터 이전 기타 서비스

이름 프리픽스로만 필터링 ▾ □ 필터 객체 및 폴더 필터링

□ 이름	크기	유형
blip_finetuned/	-	폴더
blip_finetuned_v2/	-	폴더
jsonl/	-	폴더
preprocessed_blip_dataset/	-	폴더

gcs 디렉토리 저장구조

4. 테스트 데이터에 대한 캡션 생성 (평가용)

```
# test 데이터에 대한 정답 캡션 생성
import torch
from tqdm import tqdm

references = []
predictions = []

device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)
model.eval()

for sample in tqdm(eval_dataset):
    pixel_values = torch.tensor(sample["pixel_values"]).unsqueeze(0).to(device)
    with torch.no_grad():
        generated_ids = model.generate(pixel_values=pixel_values, max_new_tokens=50)
    generated_caption = processor.decode(generated_ids[0], skip_special_tokens=True)
    predictions.append(generated_caption)

# 정답 캡션 (input_ids → 텍스트 복원)
label_ids = sample["labels"]
label_text = processor.tokenizer.decode(label_ids, skip_special_tokens=True)
references.append(label_text)
```

100% | 956/956 [18:51<00:00, 1.18s/it]

05 Image Captioning : BLIP

5. 예측 성능 평가 및 개선 시도

1. 예측 평가를 위한 3가지 지표 로드

```
!pip install evaluate nltk --quiet
!pip install rouge_score --quiet

import evaluate
import nltk
nltk.download('punkt')

Preparing metadata (setup.py) ... done
Building wheel for rouge_score (setup.py) ... done
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True
```

```
bleu = evaluate.load("bleu")
rouge = evaluate.load("rouge")
meteor = evaluate.load("meteor")

Downloading builder script: 100% [██████████] 7.02k/7.02k
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

BLEU
ROUGE-L
METEOR

• 1차 시도

```
# BLEU expects list of references per prediction
bleu_result = bleu.compute(predictions=predictions, references=[[ref] for ref
rouge_result = rouge.compute(predictions=predictions, references=references)
meteor_result = meteor.compute(predictions=predictions, references=references)

print("Evaluation Metrics on Validation Set")
print(f"BLEU score: {bleu_result['bleu']:.4f}")
print(f"ROUGE-L: {rouge_result['rougeL']:.4f}")
print(f"METEOR: {meteor_result['meteor']:.4f}")
```

```
Evaluation Metrics on Validation Set
BLEU score: 0.1174
ROUGE-L: 0.0105
METEOR: 0.2810
```

BLEU: 0.1174 , ROUGE-L: 0.0105, METEOR: 0.2810

• 2차 시도

```
# BLEU expects list of references per prediction
bleu_result = bleu.compute(predictions=predictions, references=[[ref] for ref
rouge_result = rouge.compute(predictions=predictions, references=references)
meteor_result = meteor.compute(predictions=predictions, references=references)

print("Evaluation Metrics on Validation Set")
print(f"BLEU score: {bleu_result['bleu']:.4f}")
print(f"ROUGE-L: {rouge_result['rougeL']:.4f}")
print(f"METEOR: {meteor_result['meteor']:.4f}")
```

```
Evaluation Metrics on Validation Set
BLEU score: 0.1133
ROUGE-L: 0.0188
METEOR: 0.2779
```

BLEU : 단어 단위 정밀도 평가

ROUGE-L : 문장 간 유사한 시퀀스 길이를 평가

METEOR : 정확도 + 재현율 + 어간/동의어 대응까지 반영

06 확장가능성 및 적용

06 확장가능성 및 적용

상업성 확보 전략 - 중고거래 플랫폼과의 연계 활용

해당 딥러닝 모델을 중고거래 플랫폼에 적용한다면?

- 업로드 이미지 기반 허위 매물 탐지 및 검증 보조

예: '깨끗한 상태'라 기재된 옷 이미지에서 심각한 오염이 탐지될 경우 자동 경고



- 판매자 편의 기능 제공

이미지 기반 자동 설명 생성(Captioning) - 제품명, 색상, 특징 등을 자동 추출하여 게시글 작성 자동화

초보 판매자도 빠르고 정확한 설명 가능 → 사용자 경험(UX) 향상



적용시도 : 중고나라 상품 이미지 기반 의류 분류 및 설명 생성 자동화 시스템

06 확장가능성 및 적용

*본 팀이 데이터를 직접 수집하여 진행한 프로젝트입니다.

중고나라 상품 이미지 기반 의류 분류 및 설명 생성 자동화 시스템

1. 데이터 수집 (Web Crawling)

- 대상: 중고나라 의류 카테고리 제품 (총 1,620건)
- 방식: 저가순 정렬 후 1~20 페이지 크롤링 (한 페이지에 27개)
- 사용 패키지: requests, BeautifulSoup

→ 실사용 데이터 기반 서비스 구축을 위한 크롤링 파이프라인 구축

```
url = f"https://web.joongna.com/search?category=1053&sort=PRICE_ASC_SORT&page={page}"
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, 'html.parser')

for li in soup.find_all('li'):
    a_tag = li.find('a', href=True)
    if a_tag and a_tag['href'].startswith('/product/'):
        product_url = "https://web.joongna.com" + a_tag['href']

        img_tag = a_tag.find('img')
        img_url = img_tag['src'] if img_tag and img_tag.has_attr('src') else None

        title_tag = a_tag.find('h2')
        title = title_tag.text.strip() if title_tag else None
```

일부 크롤링 코드

The screenshot shows the Joongna homepage with a search bar at the top. Below it, there's a navigation bar with links for events, certified sellers, digital gifts, and Jcoupons. A sidebar on the left lists categories like imported goods, fashion accessories, and beauty products. A main content area displays a list of items under the 'Women's Clothing' category, with a specific item for a black and white checkered blazer highlighted. A blue arrow points from this item to the detailed product page shown in the next screenshot.

This screenshot shows a detailed product page for a women's outfit. It features two images: one of a black and white checkered blazer and another of a woman wearing a dark skirt and a matching top. The page includes the brand name 'PRADA' visible on the clothing. At the top, there are sorting options: '추천순' (Recommended), '최신순' (Newest), '낮은가격순' (Lowest price), and '높은가격순' (Highest price). The '낮은가격순' option is highlighted with a blue oval. Below the images, the items are labeled as '셔츠' (Shirt) and '프 플리츠 셋' (Prada Pleated Set). The prices are listed as '15원' (15 won) and '59원' (59 won). The date '2024-11-21' and the status '만년동 | 11시간 전' (Year-round | 11 hours ago) are also displayed.

06 확장가능성 및 적용

수집된 예시 데이터

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1620 entries, 0 to 1619
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          -----          --    
 0   title        1600 non-null   object 
 1   price        1600 non-null   object 
 2   product_url  1620 non-null   object 
 3   img_url      1600 non-null   object 
 4   wish_count   1600 non-null   object 
 5   chat_count   1600 non-null   object 
dtypes: object(6)
memory usage: 76.1+ KB
```

	title	price	image_tag	wish_count	chat_count	price_num
0	자라 크롭탑 오프숄더 블라우스 구해요	1원		0	0	1.0
1	타미힐피거 남성용 슬림핏s 화이트 셔츠	1원		0	0	1.0
⋮						
1599	여성 긴팔 블라우스 새상품	8,000원		0	0	8000.0

상품명 텍스트 분석

```
words = df['title'].dropna().apply(lambda x: re.findall(r'WbWw+Wb', x.lower()))
flat_words = [word for sublist in words for word in sublist]
Counter(flat_words).most_common(40)

[('블라우스', 730),
 ('셔츠', 274),
 ('여성', 185),
 ('새상품', 105),
 ('66', 101),
 ('반팔', 97),
 ('55', 90),
 ('남방', 85),
 ('77', 56),
 ('티셔츠', 55),
 ('m', 52),
 ('체크', 50),
 ('여름', 48),
 ('쉬폰', 47),
 ('시스루', 45),
 ('레이스', 45),
 ('화이트', 44),
 ('free', 44),
 ('새것', 42),
 ('크롭', 41),
 :]
```

→ 일반적으로 제목에 의류 형태나 사이즈가 포함됨

06 LiClo 서비스 구축

Gradio 기반 자동화 시스템

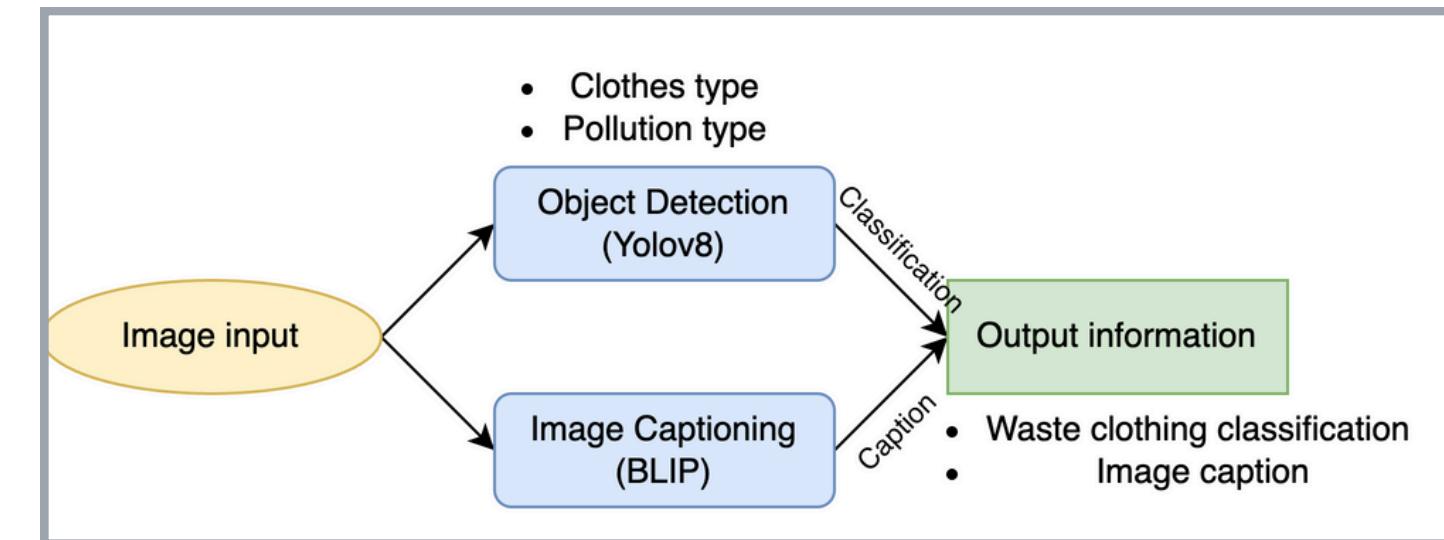
- **LiClo (Recycle + Clothing)**: 본 팀이 구축한 자동화 서비스

주요 구성 요소

- **YOLOv8**: 의류 형태 및 오염 객체 탐지 + 분류 (dispose/recycle/reuse)
- **BLIP**: 이미지 캡션 생성 (의류 설명 자동 생성)
- **Gradio**: 사용자 이미지 업로드 시 실시간 예측 결과 제공

예측 결과

- Object Detection Bounding Box 시각화
- 탐지된 클래스 이름 (dispose/recycle/reuse 중 하나)
- 각 클래스별 confidence score
- BLIP 기반 자연어 이미지 설명 생성



→ 사용자가 이미지를 업로드하면
탐지 + 분류 + 캡션 생성까지 자동 수행

06 LiClo 서비스 시연

Gradio 기반 자동화 시스템

Recycle + Clothing 시스템
YOLOv8로 객체 및 오염 부분 감지를 통한 dispose/recycle/reuse 분류 + BLIP 이미지 캡션 생성까지 자동으로 실행됩니다.

LiClo 서비스

The screenshot shows a user interface for the LiClo service. On the left, there is a file upload area with an 'img' placeholder and a file icon. Below it are 'Clear' and 'Submit' buttons. On the right, there is a results panel with a title ' 객체 감지 결과 이미지' (Object Detection Result Image) showing the blouse with a red bounding box around the neckline. Below this is a section titled 'dispose / reuse / recycle 확률' (Probability of Disposal / Reuse / Recycling) displaying the following data:
Dispose: 0.0%
Recycle: 0.0%
Reuse: 90.7%
Underneath, there is a '이미지 캡션' (Image Caption) section with the text: '이미지 캡션: 넥으로 장식이 흰색 여성용 반팔 블라우스'. At the bottom, there is a '클래스별 감지 결과' (Class-wise Detection Results) section with the text: 'reusable_BLOUSE (0.91)'.

Gradio 시연영상

<https://drive.google.com/file/d/1Yqv9p-DIfS3nolljoPeBOxzUN9bnHKmF/view?usp=sharing>

06 결론

1

폐의류의 재활용 가능성 제고라는 점에서 환경 문제 해결에 대한 시도

2

YOLO의 객체 탐지와 BLIP의 자연어 생성 능력을 결합한
딥러닝 파이프라인 구축

3

공적 분야 뿐만 아니라 상업화에 대한 가능성 제시

THANK YOU

감사합니다

이화여자대학교 데이터사이언스학과
2392004 김민지
2392038 한가은