SUPERB AI.

2,3일차

컨베이어 벨트 객체 인식 딥러닝 모델 최적화

- Vision AI 추론 적용과 모델 성능 개선

김종환 Solutions Engineering Team Lead (jonghwan.kim@superb-ai.com)

ADVANCING HUMANITY WITH AI →

학습 목표

- 학습된 모델에 대해서 추론 결과를 요청 한다.
- 추론 결과를 이미지에 적용하여 확인한다.
- 컨베이어 위에서 운영되는 검사 시스템에 Vision AI 모델을 적용한다.
- 모델의 성능을 확인하고 개선한다.

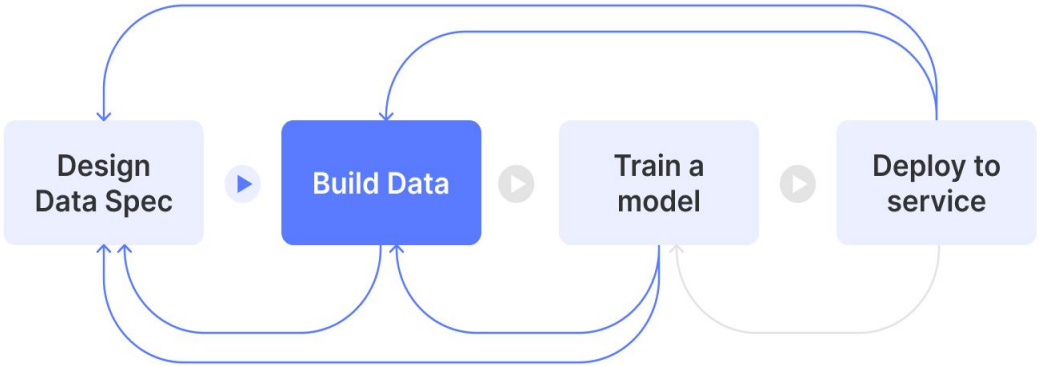
목차

- Vision AI 추론이란?
- 학습된 모델을 이용하여 AI 추론하기
- Vision AI 추론 Application 구현 (Gradio, 동영상)
- 컨베이어 상 검사 시스템 구축
- 모델 개선 과정

과정 구성

이론 강의 / 실습

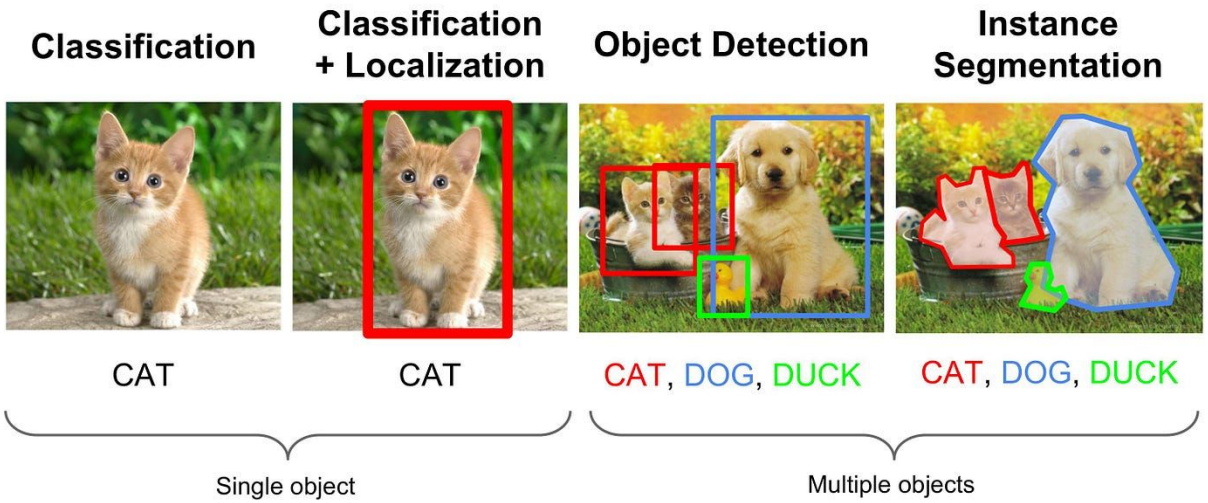
복습 - AI 학습 과정



비전 AI 추론 (Inference)

AI 추론에 대한 개념을 이해한다.

추론은 비전 AI 모델이 어떤 이미지를 보고 그 안에 있는 정보를 해석해내는 과정. 예를 들어, AI가 사진을 보면 "이건 고양이야"라고 알아내는 것. **학습(훈련)이 끝난 모델이 실전에서 데이터를 해석하는 단계.**



비전 AI 추론 사례

AI 추론에 대한 개념을 이해한다.

1. 자율 주행: 차량이 도로 위 보행자, 차량 등을 인식해 안전하게 주행
2. 의료 영상 분석: X-ray나 MRI에서 암 같은 질병을 감지
3. 무인 점포: 고객이 가져간 상품을 자동 인식해 결제까지 진행
4. 스마트폰 얼굴 인식: 얼굴 인식을 통해 기기 잠금 해제
5. 품질 검사 자동화: 제조업에서 제품의 결함을 자동으로 찾아냄
6. 식물/동물 인식 앱: 사진으로 식물이나 동물을 인식해 정보 제공
7. 보안 감시: CCTV에서 침입자나 이상 행동 감지

비전 AI 추론 방법

AI 추론에 대한 개념을 이해한다.

1. weight 파일 (pt 파일) 을 직접 이용해서 추론
    a. pt 파일은 PyTorch 에서 모델의 가중치(weight)나 전체 모델을 저장하는데 사용하는 파일 형식
    b. 이미지 전처리, 모델 추론, 결과 출력 및 후처리 과정 필요
2. API 서비스를 이용한 추론
    a. 학습한 모델을 파일로 저장한 후, API 서버를 구축
    b. 서버에 이미지를 전송되면, 추론 결과를 전달하도록 구현
    c. 1 의 과정을 Flask 나 FastAPI 같은 프레임워크를 이용하여 API 서비스를 구현

비전 AI 추론 방법 비교

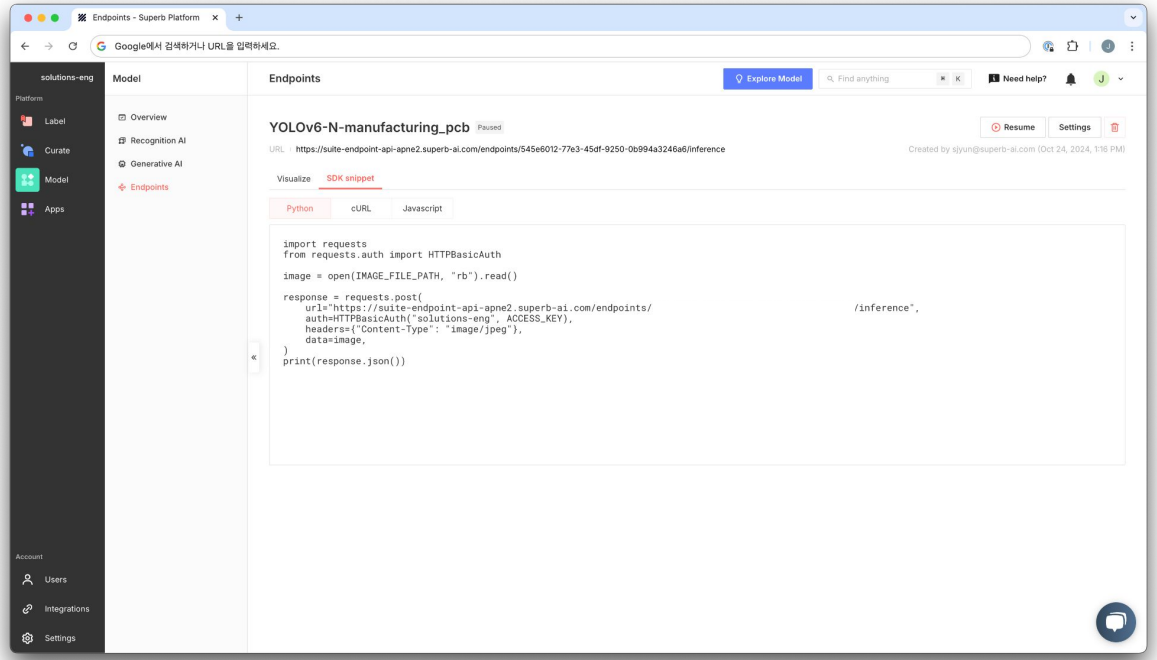
AI 추론에 대한 개념을 이해한다.

모델 파일을 이용한 추론	API 서비스를 이용한 추론
• 빠른 응답 속도 • 네트워크 의존 없음 • 환경 설정 요구 • 엣지 디바이스	• 확장성 • 배포, 유지보수 용이 • 다양한 디바이스와의 호환성 • 클라우드 환경

[실습1] API 를 이용한 추론

실습목표: Superb Platform 의 Endpoint 기능을 이용하여 추론을 요청하고 결과를 확인한다.

Superb Platform Endpoint



API 서비스를 이용한 추론 코드 설명

Superb Platform 에 구현되어 있는 추론 API 서비스를 이용해서 실제로 추론 요청하고 결과를 확인한다.

```
import requests
from requests.auth import HTTPBasicAuth


image = open(IMAGE_FILE_PATH, "rb").read()


response = requests.post(
    url="https://suite-endpoint-api-apne2.superb-ai.com/endpoints/.../inference",
    auth=HTTPBasicAuth("solutions-eng", ACCESS_KEY),
    headers={"Content-Type": "image/jpeg"},
    data=image,
)
print(response.json())
```

API 서비스를 이용한 추론 코드 - Input

Superb Platform 에 구현되어 있는 추론 API 서비스를 이용해서 실제로 추론 요청하고 결과를 확인한다.

이미지 파일을 binary 로 변환

- 이미지를 바이너리로 변환: 이미지를 컴퓨터가 이해하고 처리할 수 있는 0과 1의 조합으로 만드는 과정

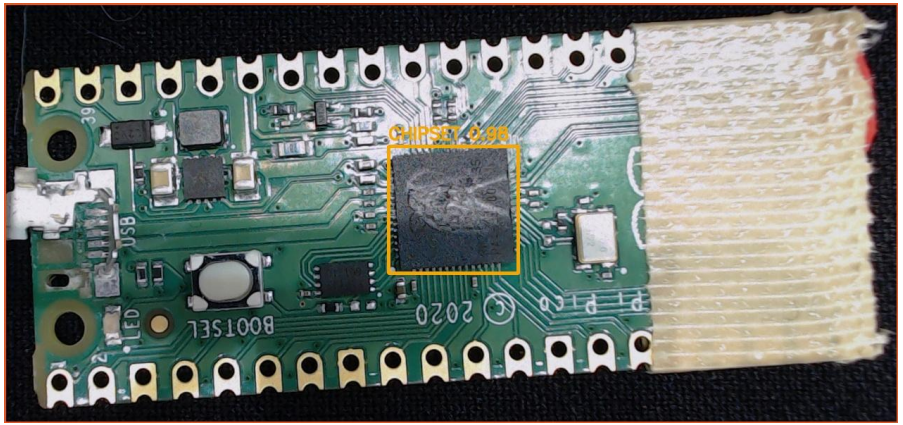
pcb-demo — python3 — python3 — Python — 90×24

```
Python 3.11.7 (main, Dec  4 2023, 18:10:11) [Clang 15.0.0 (clang-1500.1.0.2.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> IMAGE_FILE_PATH = "pico.jpg"
>>> image = open(IMAGE_FILE_PATH, "rb").read()
>>> image
b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x00\x01\x00\x00\xff\xdb\x00C\x00\x0
2\x01\x01\x01\x01\x01\x02\x01\x01\x01\x02\x02\x02\x02\x02\x04\x03\x02\x02\x02\x02\x05\x04\
x04\x03\x04\x06\x05\x06\x06\x06\x05\x06\x06\x06\x07\t\x08\x06\x07\t\x07\x06\x06\x08\x0b\x0
8\t\n\n\n\n\n\x06\x08\x0b\x0c\x0b\n\x0c\t\n\n\n\xff\xdb\x00C\x01\x02\x02\x02\x02\x02\x02\x
05\x03\x03\x05\n\x07\x06\x07\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\xff\xc0\x00\x11\x08\x048\x07\x80\x03\x01"\x00\x02\x
11\x01\x03\x11\x01\xff\xc4\x00\x1f\x00\x00\x01\x05\x01\x01\x01\x01\x01\x01\x00\x00\x00\x00
\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\xff\xc4\x00\xb5\x10\x00\x02\x01\x
03\x03\x02\x04\x03\x05\x05\x04\x04\x00\x00\x01}\x01\x02\x03\x00\x04\x11\x05\x12!1A\x06\x13
Qa\x07"q\x142\x81\x91\xa1\x08#B\xb1\xc1\x15R\xd1\xf0$3br\x82\t\n\x16\x17\x18\x19\x1a%&\'()
*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz\x83\x84\x85\x86\x87\x88\x89\x8a\x92\x93\x94\x95\x
96\x97\x98\x99\x9a\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba
\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xe1\xe2\xe3\xe4\x
e5\xe6\xe7\xe8\xe9\xea\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xff\xc4\x00\x1f\x01\x00\x03
\x01\x01\x01\x01\x01\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x
08\t\n\x0b\xff\xc4\x00\xb5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04\x04\x00\x01\x02
w\x00\x01\x02\x03\x11\x04\x05!1\x06\x12AQ\x07aq\x13"2\x81\x08\x14B\x91\xa1\xb1\xc1\t#3R\xf
0\x15br\xd1\n\x16$4\xe1%\xf1\x17\x18\x19\x1a&\'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz\x
82\x83\x84\x85\x86\x87\x88\x89\x8a\x92\x93\x94\x95\x96\x97\x98\x99\x9a\xa2\xa3\xa4\xa5\xa6
```

API 서비스를 이용한 추론 코드 - Output

Superb Platform 에 구현되어 있는 추론 API 서비스를 이용해서 실제로 추론 요청하고 결과를 확인한다.


```
{
  "objects": [
    {
      "class": "CHIPSET",
      "score": 0.9807657241821289,
      "box": [
        133,
        164,
        221,
        208
      ]
    } ......
  ]
}
```

API 서비스를 이용한 추론

Superb Platform 에 구현되어 있는 추론 API 서비스를 이용해서 실제로 추론 요청하고 결과를 확인한다.

실습

[실습2] 추론 결과 적용

실습목표: 추론 결과를 OpenCV 라이브러리를 이용해 이미지에 출력한다.

OpenCV 라이브러리

OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

OpenCV는 **이미지와 비디오 처리에 특화된 오픈소스 컴퓨터 비전 라이브러리**. 컴퓨터 비전과 관련된 다양한 기능을 제공해서, 얼굴 인식, 객체 추적, 이미지 필터링 등 많은 작업에 사용되고 있음

주요 특징

1. **이미지 처리 기능**: 이미지 불러오기, 크기 조정, 자르기, 색상 변환 등 기본적인 이미지 처리가 가능해.
2. **객체 인식 및 추적**: 얼굴, 사람, 자동차 등 다양한 객체를 인식하고 추적하는 데 사용할 수 있어.
3. **동영상 처리**: 실시간 비디오 스트리밍에서 객체 추적, 모션 감지 등을 지원해.

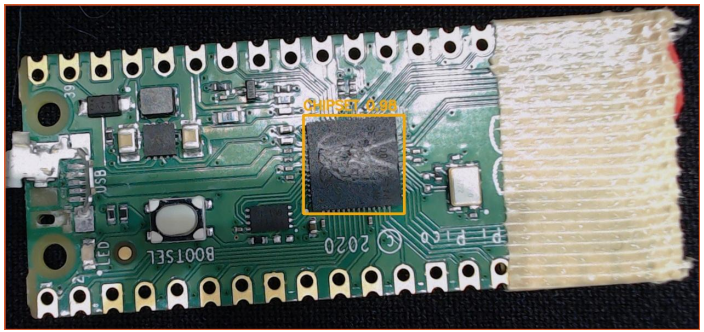
OpenCV 라이브러리 - 박스 그리기

OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

```
# 이미지 불러오기
img_path = 'path_to_your_image.jpg'  # 이미지 파일 경로 설정
img = cv2.imread(img_path)

# 박스 칠 좌표 설정 (예: 좌측 상단 (50, 50), 우측 하단 (200, 200))
start_point = (50, 50)  # 박스 시작 좌표 (x, y)
end_point = (200, 200)  # 박스 끝 좌표 (x, y)
color = (0, 255, 0)  # BGR 색상 (초록색)
thickness = 2  # 박스 선의 두께

# 박스 그리기
cv2.rectangle(img, start_point, end_point, color, thickness)
```

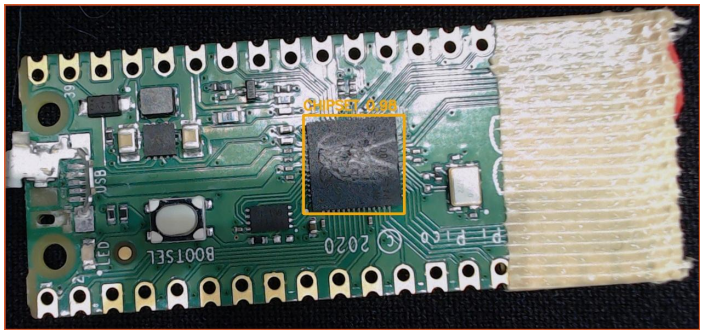


OpenCV 라이브러리 - 텍스트 추가하기

OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

```
# 텍스트 설정
text = "Hello, OpenCV!"  # 추가할 텍스트
position = (50, 50)  # 텍스트 시작 위치 (x, y)
font = cv2.FONT_HERSHEY_SIMPLEX  # 글꼴 설정
font_scale = 1  # 글자 크기
color = (0, 255, 0)  # BGR 색상 (초록색)
thickness = 2  # 글자 두께

# 텍스트 추가
cv2.putText(img, text, position, font, font_scale, color, thickness, cv2.LINE_AA)
```



OpenCV 라이브러리

OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

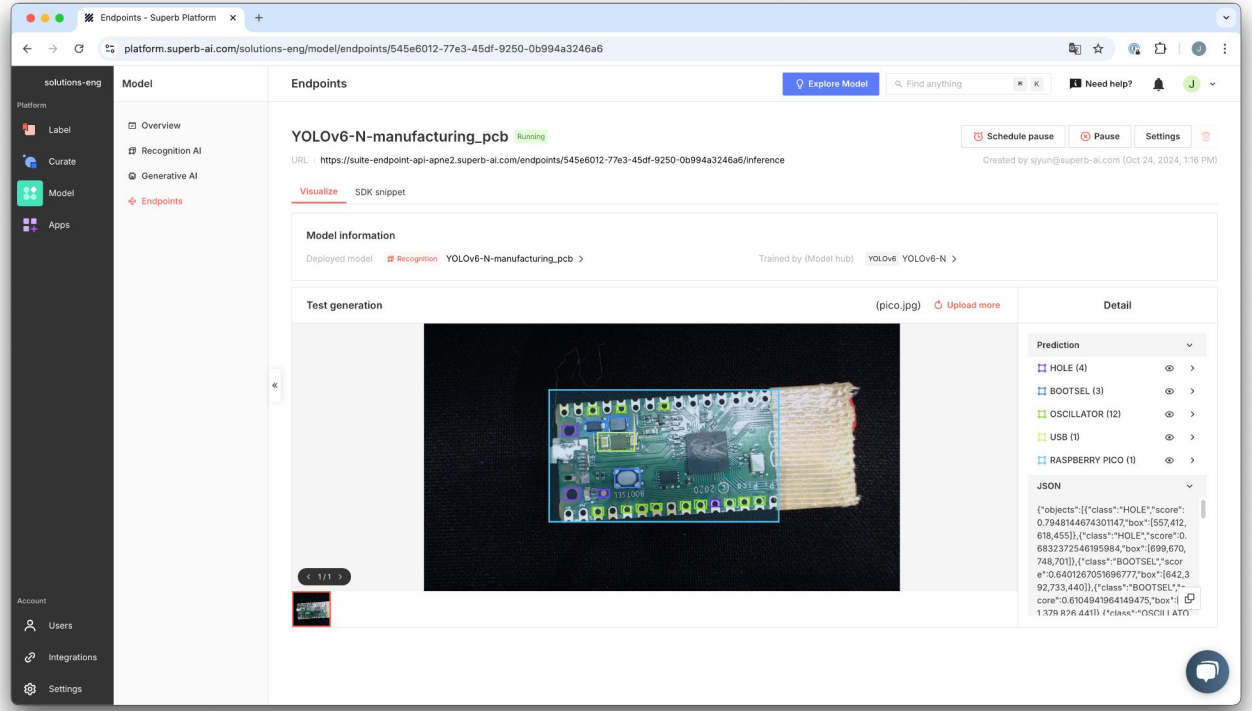
실습

[실습3] 비전 AI 어플리케이션 구현

실습목표: Gradio 프레임워크를 이용하여 비전 AI 를 추론하고 결과를 확인하는 페이지를 구현한다.

비전 AI 추론 어플리케이션 구현

Gradio 프레임워크를 이용해서 비전 AI 를 활용하는 어플리케이션을 구현한다.



비전 AI 추론 어플리케이션 구현

Gradio 프레임워크를 이용해서 비전 AI 를 활용하는 어플리케이션을 구현한다.

Gradio는 **간단한 웹 인터페이스로 머신러닝 모델이나 함수의 데모를 만들 수 있는 오픈소스 라이브러리**. 코드 몇 줄만으로 사용자 친화적인 웹 UI를 생성할 수 있어, 모델을 공유하거나 실험할 때 아주 편리.

주요 특징

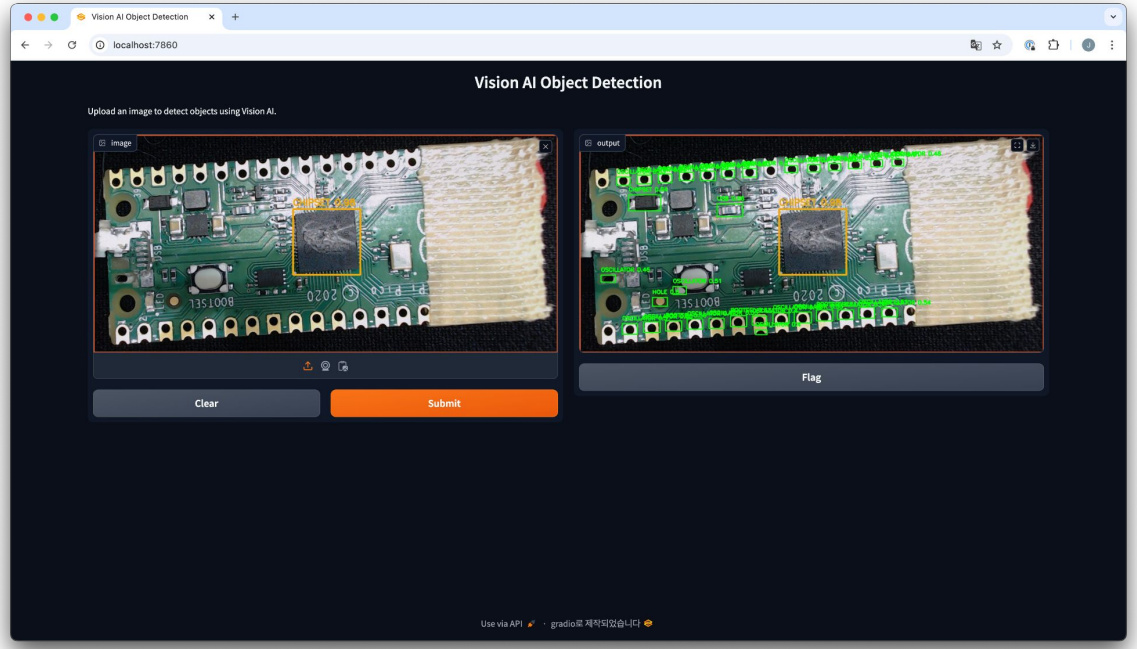
1. **간편한 웹 인터페이스 생성**: 머신러닝 모델, 데이터 처리 함수 등을 웹 브라우저에서 바로 시각화할 수 있음.
2. **다양한 입출력 지원**: 이미지, 텍스트, 오디오, 비디오 등 여러 유형의 데이터로 인터페이스를 생성 가능.
3. **클라우드 배포**: 웹 UI를 Gradio 서버에 바로 업로드해 다른 사람들과 손쉽게 공유할 수 있음.

활용 예시

- **모델 데모**: 이미지 분류, 텍스트 생성 등 모델을 간단히 테스트할 수 있는 웹 인터페이스 생성.
- **실험 공유**: 연구나 프로젝트 결과물을 웹에서 누구나 체험할 수 있도록 배포.
- **피드백 수집**: 사용자와 협력하여 모델에 대한 피드백을 실시간으로 수집

비전 AI 추론 어플리케이션 구현

Gradio 프레임워크를 이용해서 비전 AI 를 활용하는 어플리케이션을 구현한다.



비전 AI 추론 어플리케이션 구현

Gradio 프레임워크를 이용해서 비전 AI 를 활용하는 어플리케이션을 구현한다.

```
# 가상의 비전 AI API URL (예: 객체 탐지 API)
VISION_API_URL = ""
TEAM = ""
ACCESS_KEY = ""


def process_image(image):
    # 이미지를 OpenCV 형식으로 변환
    image = np.array(image)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    # 이미지를 API에 전송할 수 있는 형식으로 변환
    _, img_encoded = cv2.imencode(".jpg", image)

    # API 호출 및 결과 받기 - 실습1

    # API 결과를 바탕으로 박스 그리기 - 실습2

    # BGR 이미지를 RGB로 변환
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return Image.fromarray(image)
```

비전 AI 추론 어플리케이션 구현

Gradio 프레임워크를 이용해서 비전 AI 를 활용하는 어플리케이션을 구현한다.

실습

[실습4] 동영상 추론

실습목표: 동영상을 프레임 단위로 나누고, 프레임 별로 추론을 적용하고, 다시 동영상으로 만든다.

동영상 추론

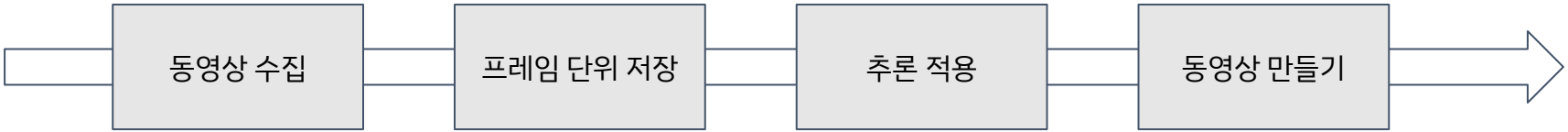
동영상 파일에 Vision AI 를 적용하고, 결과물을 만든다.

동영상 추론

동영상 파일에 Vision AI 를 적용하고, 결과물을 만든다.

동영상 추론

동영상 파일에 Vision AI 를 적용하고, 결과물을 만든다.



동영상 추론

pytube 라이브러리를 이용해서 YouTube 영상 다운로드 받기

```
import yt_dlp

# 다운로드할 YouTube URL
url = "https://www.youtube.com/shorts/JRMEpi-4U2Y"

# yt-dlp 설정
ydl_opts = {
    "format": "best",
    "outtmpl": "downloaded_video.%(ext)s",
}

# 다운로드
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download([url])
```

동영상 추론

OpenCV 라이브러리를 이용하여 동영상을 프레임 별로 저장

```python
import cv2
import os

video_path = "downloaded_video.mp4"

# 2. cv2를 이용해 동영상 파일 열기
cap = cv2.VideoCapture(video_path)

# 프레임을 저장할 디렉토리 생성
output_dir = "frames"
os.makedirs(output_dir, exist_ok=True)

# 3. 프레임 단위로 이미지로 저장
frame_count = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    # 이미지 파일로 프레임 저장
    frame_path = os.path.join(output_dir, f"frame_{frame_count:04d}.jpg")
    cv2.imwrite(frame_path, frame)
    frame_count += 1

# 동영상 파일 닫기
cap.release()
print(f"총 {frame_count}개의 프레임이 저장되었습니다.")
```

동영상 추론

Ultraytics YOLOv8 pre-trained 모델 추론

```
from ultralytics import YOLO
import os

# Load a model
model = YOLO("yolov8n.pt")  # pretrained YOLO8n model

frames = os.listdir("frames/")

for frame in frames:
    # YOLOv8 모델을 사용하여 객체 감지 수행
    results = model("frames/" + frame, device="mps")

    # Process results list
    for result in results:
        result.save(filename="results/" + l)  # save to disk
```

동영상 추론

추론된 프레임을 다시 mp4 로 만든다.

```python
from moviepy.editor import ImageSequenceClip
import os

# 이미지 파일들이 저장된 디렉토리 경로
image_folder = "results"  # 이미지 파일들이 있는 폴더 경로
fps = 30  # 초당 프레임 수 설정

# 이미지 파일 리스트 가져오기 (예: jpg, png 형식)
image_files = [
    os.path.join(image_folder, img)
    for img in sorted(os.listdir(image_folder))
    if img.endswith((".jpg", ".png"))
]

# 이미지 시퀀스를 이용해 클립 생성
clip = ImageSequenceClip(image_files, fps=fps)

# 동영상 파일로 저장
output_path = "output_video.mp4"
clip.write_videofile(output_path, codec="libx264")

print(f"동영상이 생성되었습니다: {output_path}")
```

동영상 추론

동영상 파일에 Vision AI 를 적용하고, 결과물을 만든다.

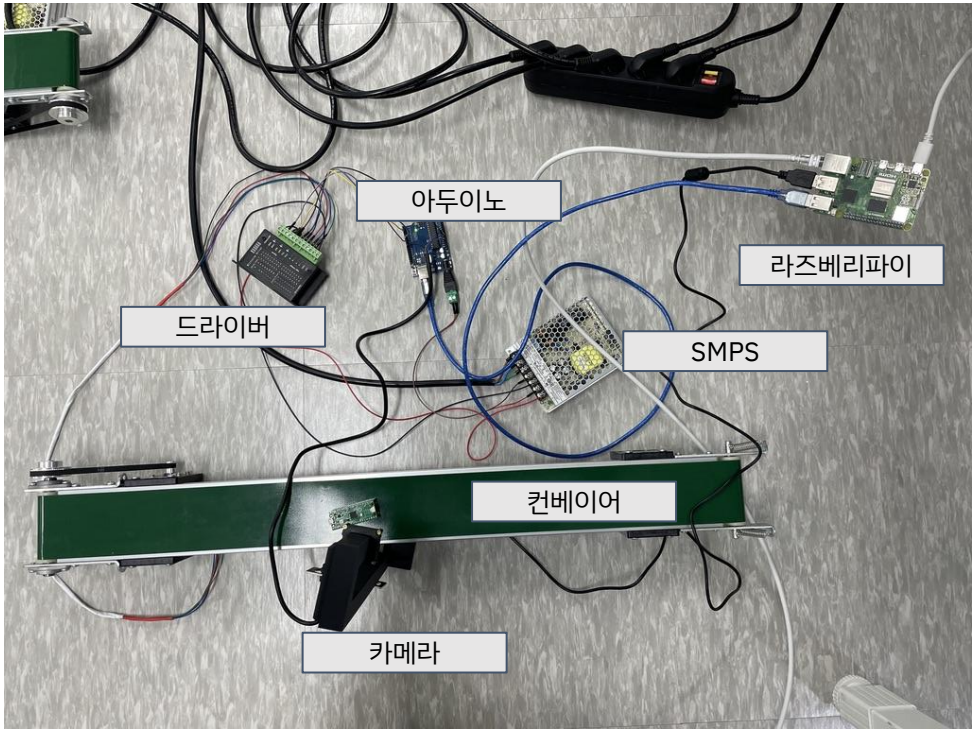
실습

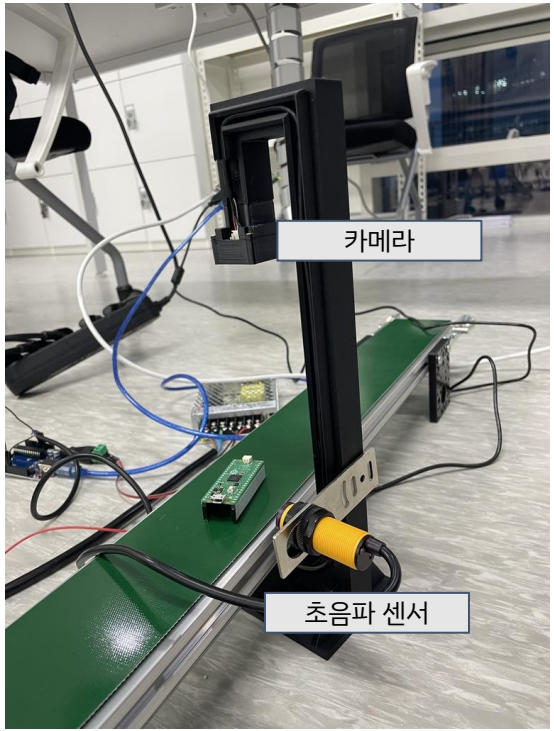
[실습5] Vision AI 를 이용한 외관 검사

실습목표: 컨베이어 상에서 이동되는 PCB 제품에 대해서 외관 검사하고, 모델을 개선한다.

Vision AI 를 이용한 외관 검사

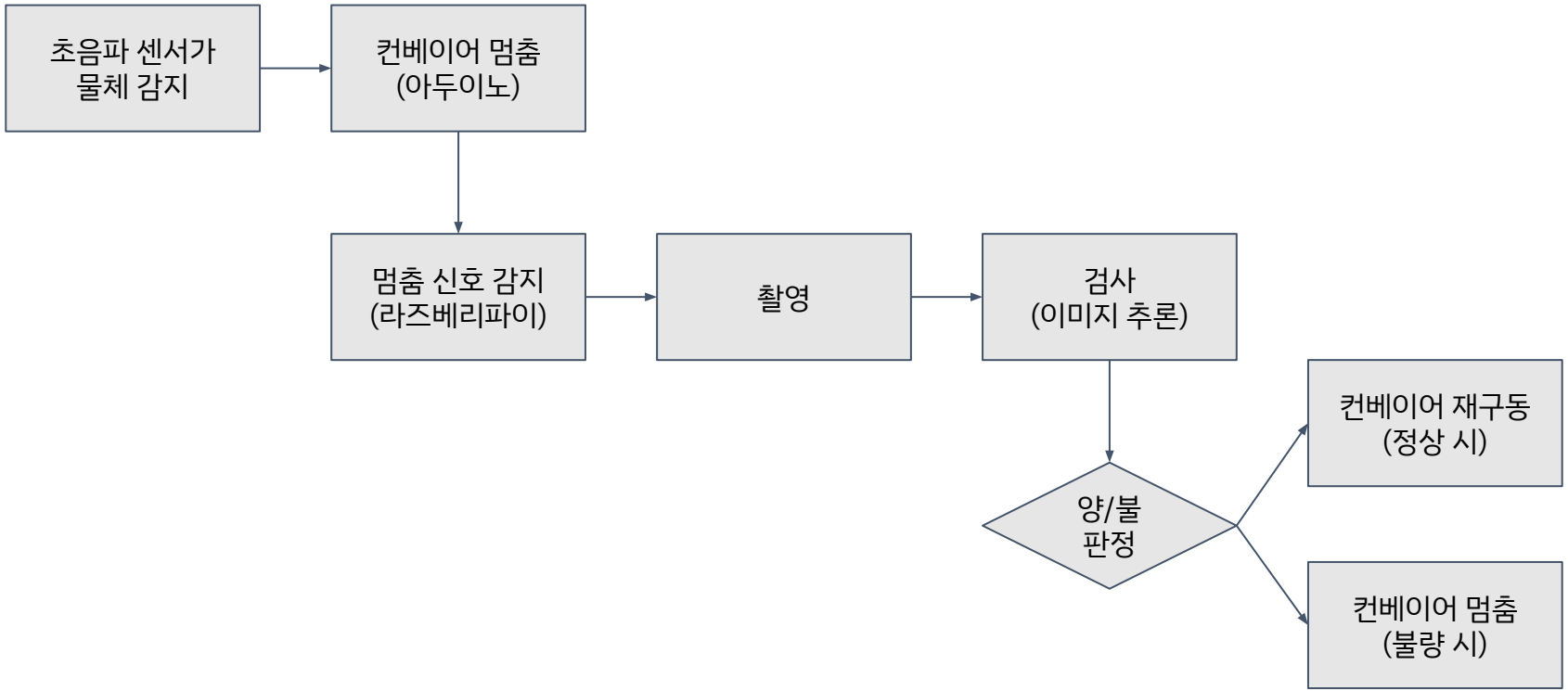
실습 환경





Vision AI 를 이용한 외관 검사

실습 환경



Vision AI 를 이용한 외관 검사

실습 환경

실습

Vision AI 를 이용한 외관 검사

Confusion Matrix 의 이해

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Vision AI 를 이용한 외관 검사

모델의 정확도

Vision AI 추론 결과와 실제 결과를 비교해서 모델의 성능을 측정한다.

- 정확도(True Positive): 모델이 전체 데이터에서 얼마나 정확하게 예측했는지 보여주는 지표

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{True Negative (TN)}}{\text{전체 예측 수 (TP + TN + FP + FN)}}$$

특히, 제조 분야에서는

- 과검(False Positive, 1종 오류) : 제품이 실제로는 정상인데 품질검사 결과는 불량이라고 예측
- 미검(False Negative, 2종 오류) : 제품이 실제로는 불량인데 품질검사 결과는 정상이라고 예측
- 과검과 미검은 Trade-off 관계
- 주로 제조에서는 미검을 0% 로 만들면서 과검을 최대한 줄이는 방향을 원함

Vision AI 를 이용한 외관 검사

컨베이어 실습

1. 어떤 시스템인지 설계하기 (양품/불량에 대한 정의)
2. 일부러 불량 만들기
3. 양품과 불량을 샘플을 컨베이어에서 검사하기
4. 검사 결과를 Confusion matrix 로 만들기
5. 정확도 등 지표 계산
6. 모델의 정확도를 높일 수 있는 방법 토의

Vision AI 를 이용한 외관 검사

모델 개선

실습