SUPERB AI.

2,3일차

컨베이어 벨트 객체 인식 딥러닝 모델 최적화

- Vision AI 추론 적용과 모델 성능 개선

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

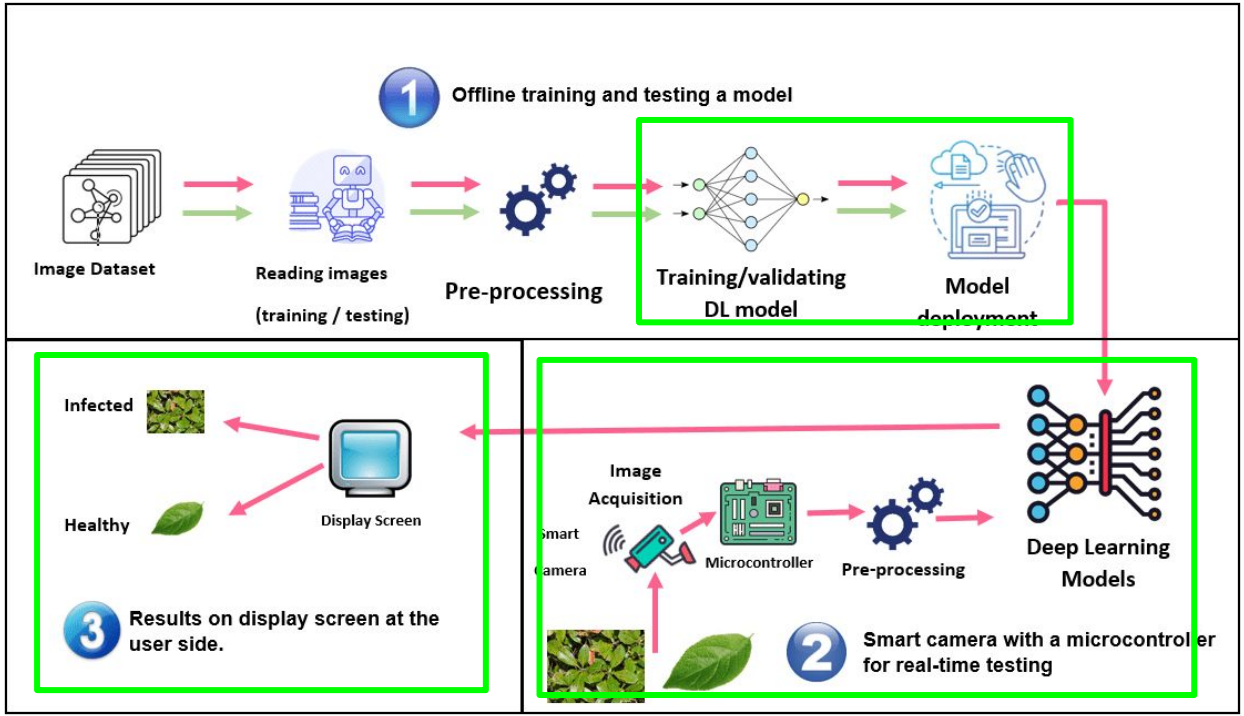
GITHUB
https://github.com/SuperbAI-KDT/vision-ai-inference-practice

python 3.9
pip install -r requirements.txt

0. python-basic	수정	2 months ago
1. vision-ai-inference	init	2 months ago
2. gradio-demos	init	2 months ago
3. yolov8-inference	streamlit 버전 수정	2 months ago
4. video-ai-inference	init	2 months ago
5. conveyor-system	수정	2 months ago
.gitignore	Initial commit	2 months ago
README.md	Initial commit	2 months ago
requirements.txt	streamlit 버전 수정	2 months ago

비전 AI 활용단계

1. 모델 학습과 활용하는 단계



AI 배포 방법

1. Off-line
    a. weight 파일을 직접 이용해서 추론
        i. 독립적인 GPU 서버 or Edge 컴퓨터에 활용
        ii. 빠르게 동작이 가능하며 구성된 시스템 이외의 외부 환경에 영향을 받지 않음
    b. 모델 활용방법
        i. 전처리, 모델 추론, 결과 출력 및 후처리 과정 필요
2. On-line
    a. Online 서비스를 이용한 추론
        i. 학습한 모델을 API, 플랫폼 등을 통해 활용하도록 구현
        ii. 간편하게 사용가능하고 고가의 시스템을 보유하지 않아도 사용 가능
    b. 모델 활용방법
        i. 통신을 통하여 데이터를 전송되면, 추론 결과를 전달

비전 AI 추론 방법 비교

AI 추론에 대한 개념을 이해한다.

구분	오프라인 방식	온라인 서비스
특징	로컬 설치를 통해 배포	클라우드 기반으로 인터넷을 통해 사용자에게 접근 및 서비스 제공
장점	- 인터넷 연결 없이 사용 가능 - 데이터 보안성 향상 (로컬 저장으로 인해) - 초기 설치 비용 절감 가능	- 유지보수 및 업데이트 용이 - 다양한 디바이스와 플랫폼에서 접근 가능 - 스케일링 용이
단점	- 업데이트 및 유지보수가 번거로움 - 배포 시간과 비용이 많이 소요될 수 있음	- 인터넷 연결 의존성 - 데이터 보안 우려 (클라우드 기반으로 데이터 노출 가능성)
적합한 상황	- 고립된 환경에서의 사용 - 민감한 데이터가 많아 보안이 중요한 경우	- 지속적인 기능 추가 및 업데이트가 필요한 경우 - 다양한 사용자에게 접근성이 중요한 경우
유지보수 난이도	- 유지보수 시 매번 물리적 배포 필요	- 원격으로 즉각적인 업데이트 및 유지보수 가능
비용 구조	초기 배포 시 비용 집중	구독 기반의 비용 모델 사용 가능, 지속적 운영 비용 발생

[실습1] API 를 이용한 추론

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

REST API

REST API란?

- **REST (Representational State Transfer)**:
  - 자원을 URL로 표현하고, HTTP 프로토콜을 통해 상태를 전송하는 아키텍처 스타일
- **API (Application Programming Interface)**:
  - 소프트웨어 간 상호작용을 가능하게 하는 인터페이스

REST API의 특징

- **Stateless**: 요청 간 서버가 클라이언트 상태를 저장하지 않음
- **Resource 기반**: 모든 데이터는 고유한 URL로 식별
- **표준 메서드 사용**:
  - **GET**: 데이터 조회
  - **POST**: 데이터 생성
  - **PUT**: 데이터 수정
  - **DELETE**: 데이터 삭제
- **JSON 형식**: 요청 및 응답에서 주로 사용 (가볍고 사람이 읽기 쉬움)

REST API의 작동 원리

1. 클라이언트가 특정 **HTTP 메서드**와 **URL**로 요청을 전송.
2. 서버는 요청된 자원을 처리하고 응답을 반환.
3. 응답은 **HTTP 상태 코드**와 함께 데이터(주로 JSON)를 포함.

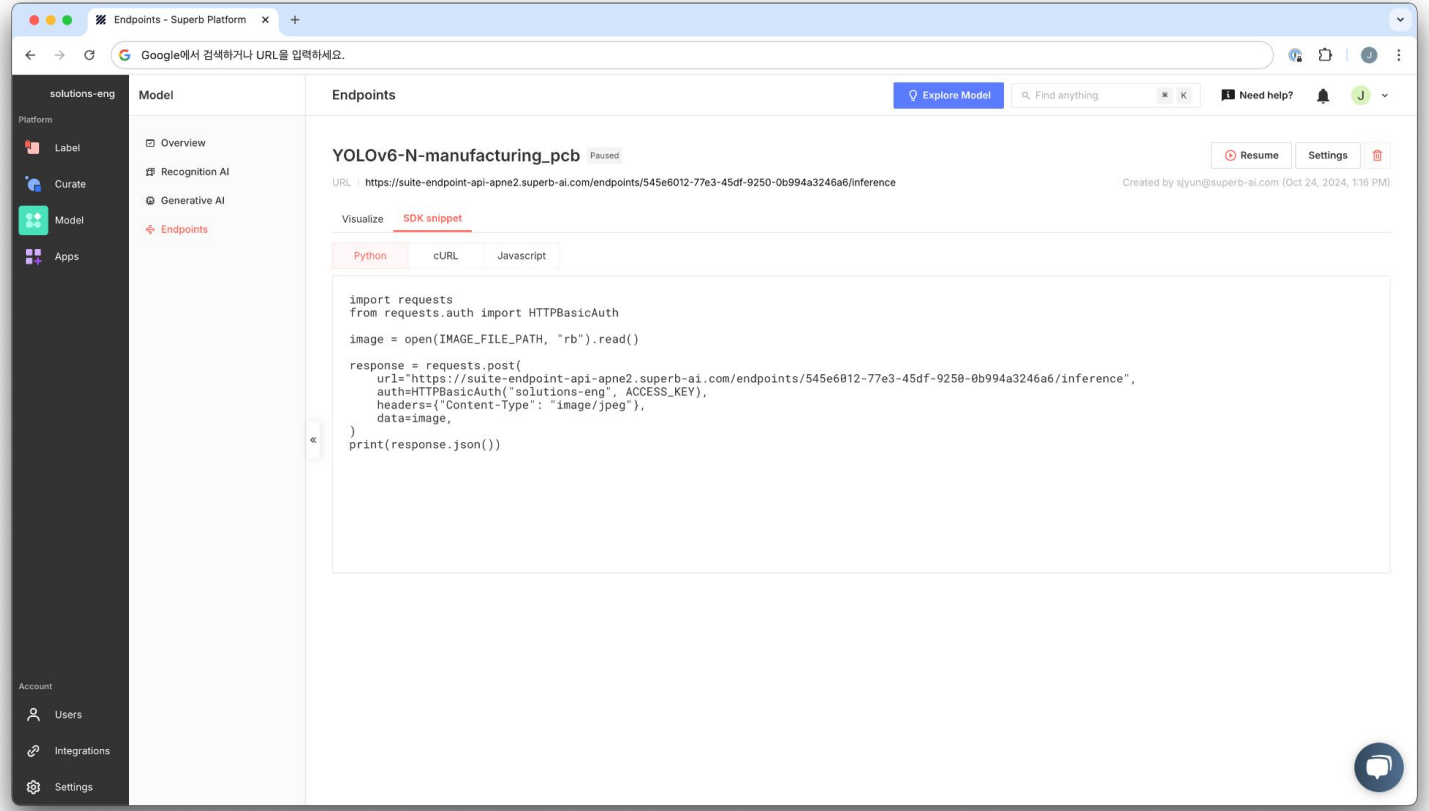
Requests를 이용한 GET, POST

http://192.168.10.58:8080

https://jsonviewer.stack.hu/

```python
import requests

# GET 요청
response = requests.get("http://127.0.0.1:5000/data")

# 응답 출력
print(f"GET 요청 상태 코드: {response.status_code}")
print(f"GET 요청 응답 내용: {response.json()}")

# POST 요청
data = {
    "name": "홍길동",
    "group": "1조"
}

response = requests.post("http://127.0.0.1:5000/data", json=data)

# 응답 출력
print(f"POST 요청 상태 코드: {response.status_code}")
print(f"POST 요청 응답 내용: {response.json()}")
```

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 byte 데이터로 전송

- 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\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
```

JPEG 파일의 구조

JPEG 파일은 여러 섹션으로 구성되며, 중요한 정보는 아래와 같습니다:

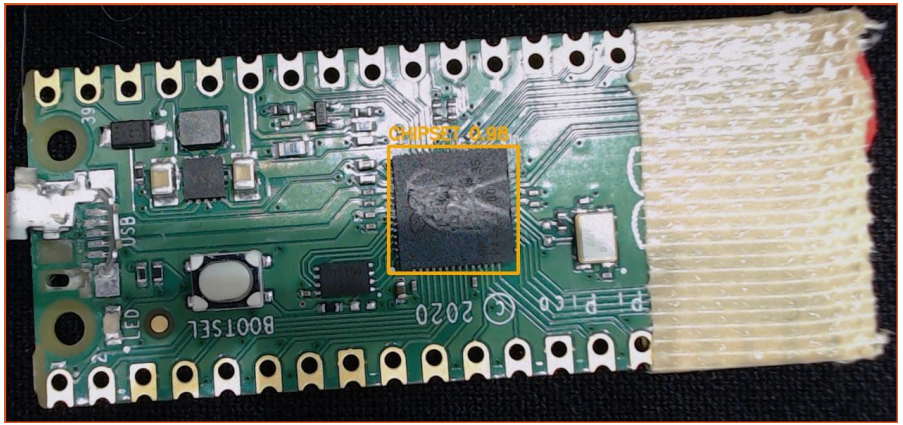
1. **SOI (Start of Image)**:
   - 파일의 시작을 나타내는 2바이트 (`0xFFD8`).
   - 모든 JPEG 파일은 이 값으로 시작합니다.
2. **APPn 마커**:
   - 애플리케이션 세그먼트로, 파일에 포함된 (`0xFFE0`).
     메타데이터(EXIF 등)를 나타냅니다.
   - 예: 카메라 모델, 촬영 시간, GPS 위치 정보.
3. **DQT (Define Quantization Table)**:
   - 이미지 압축에 사용되는 양자화 테이블을 정의.
4. **SOF (Start of Frame)**:
   - 이미지의 해상도, 컬러 채널 등의 정보를 포함.
5. **DHT (Define Huffman Table)**:
   - JPEG 압축에 사용된 허프만 테이블 정의.
6. **SOS (Start of Scan)**:
   - 실제 이미지 데이터가 시작되는 지점 (`0xFFDA`).

SUPERB AI.

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
      ]
    } ......
  ]
}
```

[실습2] 추론 결과 적용

실습목표: 추론 결과를 OpenCV 라이브러리를 이용해 이미지에 출력한다.

OpenCV 라이브러리

OpenCV(Open Source Computer Vision):

- 컴퓨터 비전 및 이미지 처리 작업을 위한 오픈소스 라이브러리.
- 2000년에 인텔(Intel)에서 시작되어 현재는 오픈소스 커뮤니티와 OpenCV.org에 의해 관리.

지원 언어: Python, C++, Java, MATLAB 등.
지원 플랫폼: Windows, macOS, Linux, Android, iOS.

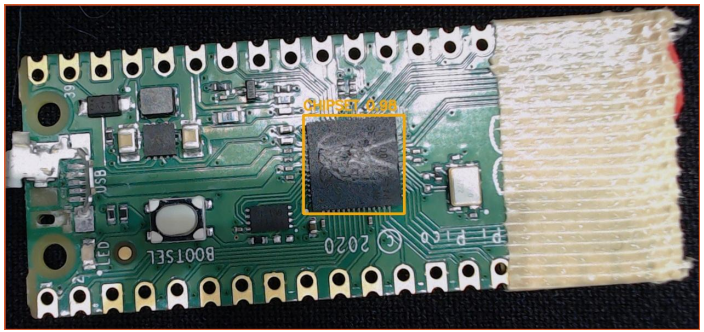
주요 특징

1. **이미지 처리**:
   a. 이미지 필터링, 윤곽선 검출, 히스토그램 계산 및 색상 변환.
2. **객체 탐지 및 추적**:
   a. Haar Cascade, 딥러닝 기반 모델을 활용한 얼굴 및 객체 탐지.
3. **동영상 분석**:
   a. 실시간 비디오 캡처, 배경 제거, 모션 감지.
4. **3D 비전**:
   a. 카메라 캘리브레이션, 스테레오 매칭, 깊이 맵 생성.
5. **머신러닝 지원**:
   a. SVM, KNN, Decision Trees 등 다양한 머신러닝 알고리즘 내장.
6. **딥러닝 프레임워크 통합**:
   a. TensorFlow, PyTorch, Caffe와의 호환성으로 딥러닝 모델 로드 및 실행 지원.

OpenCV 라이브러리 - 박스 그리기

OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

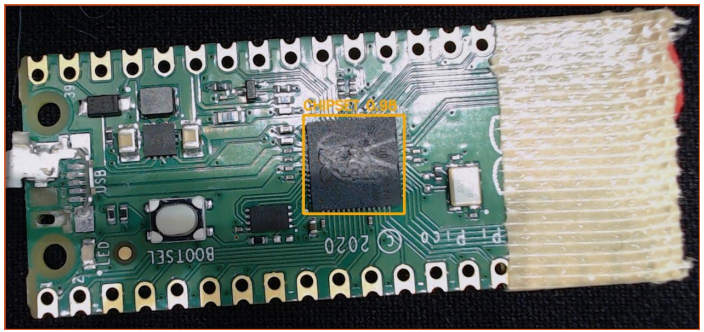
```
# 이미지 불러오기
img_path = 'path_to_your_image.jpg'  # 이미지 파일 경로 설정
img = cv2.imread(img_path)

# 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

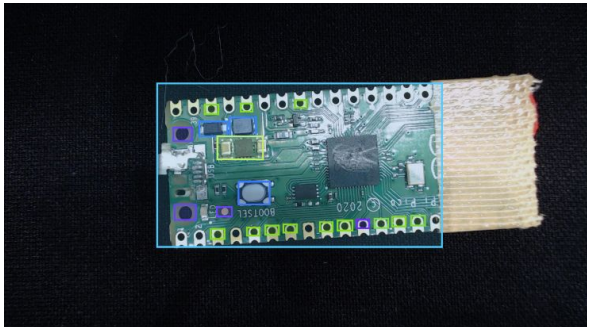
OpenCV 라이브러리에 대해서 이해하고, 이미지에 추론 결과를 출력한다.

```
# 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

```python
import cv2

# 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)

# 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)
```

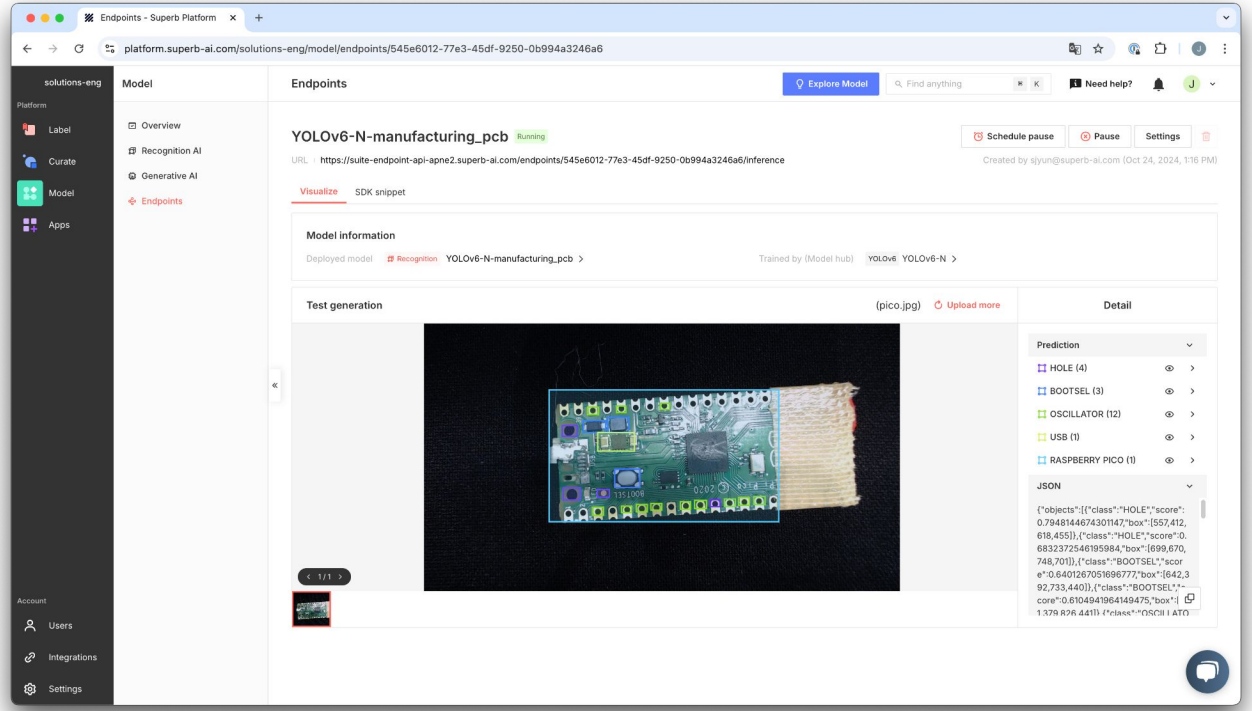


[실습3] 비전 AI 어플리케이션 구현

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

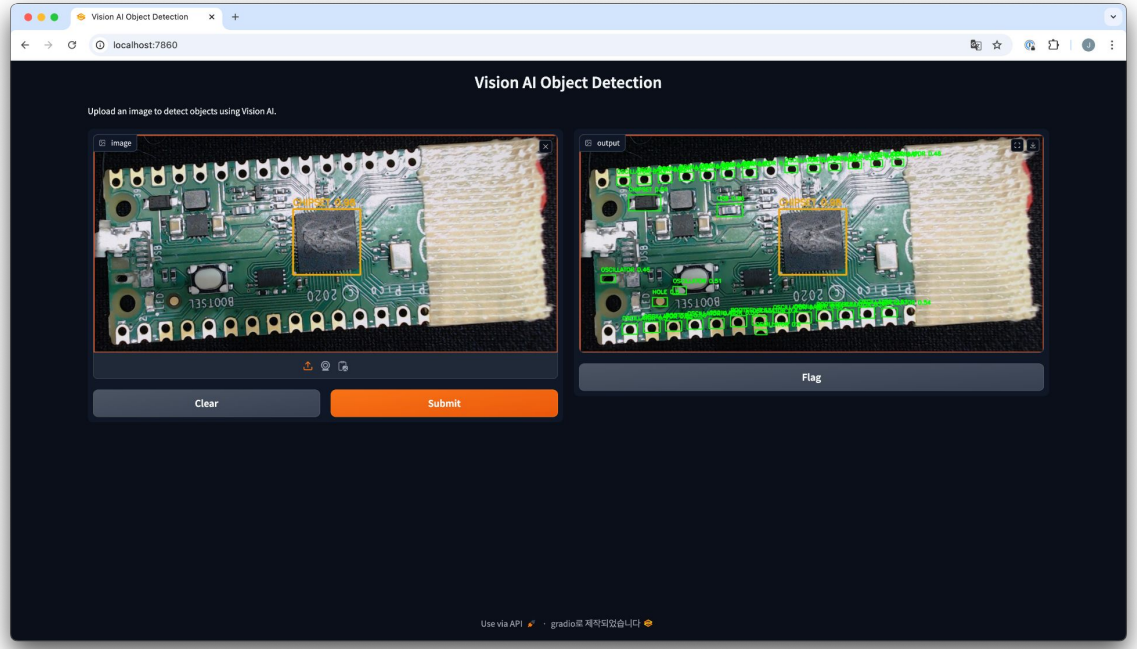
1. **간편한 웹 인터페이스 생성**: 머신러닝 모델, 데이터 처리 함수 등을 웹 브라우저에서 바로 시각화할 수 있음.
2. **다양한 입출력 지원**: 이미지, 텍스트, 오디오, 비디오 등 여러 유형의 데이터로 인터페이스를 생성 가능.
3. **클라우드 배포**: 웹 UI를 Gradio 서버에 바로 업로드해 다른 사람들과 손쉽게 공유할 수 있음.

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.

```python
# 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)
```

```python
# Gradio 인터페이스 설정
iface = gr.Interface(
    fn=process_image,
    inputs=gr.Image(type="pil"),
    outputs="image",
    title="Vision AI Object Detection",
    description="Upload an image to detect
objects using Vision AI.",
)
# 인터페이스 실행
iface.launch()
```

[실습4] 동영상 추론

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

동영상 추론

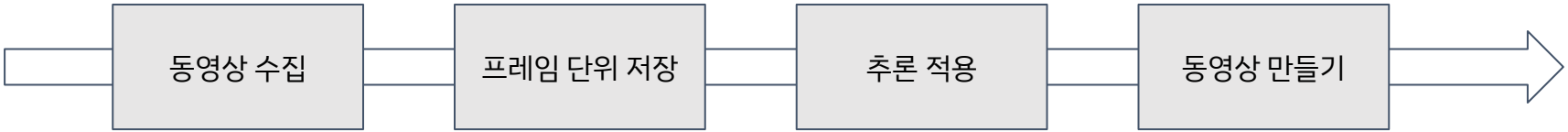
동영상 파일에 Vision AI 를 적용하고, 결과물을 만든다.

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.

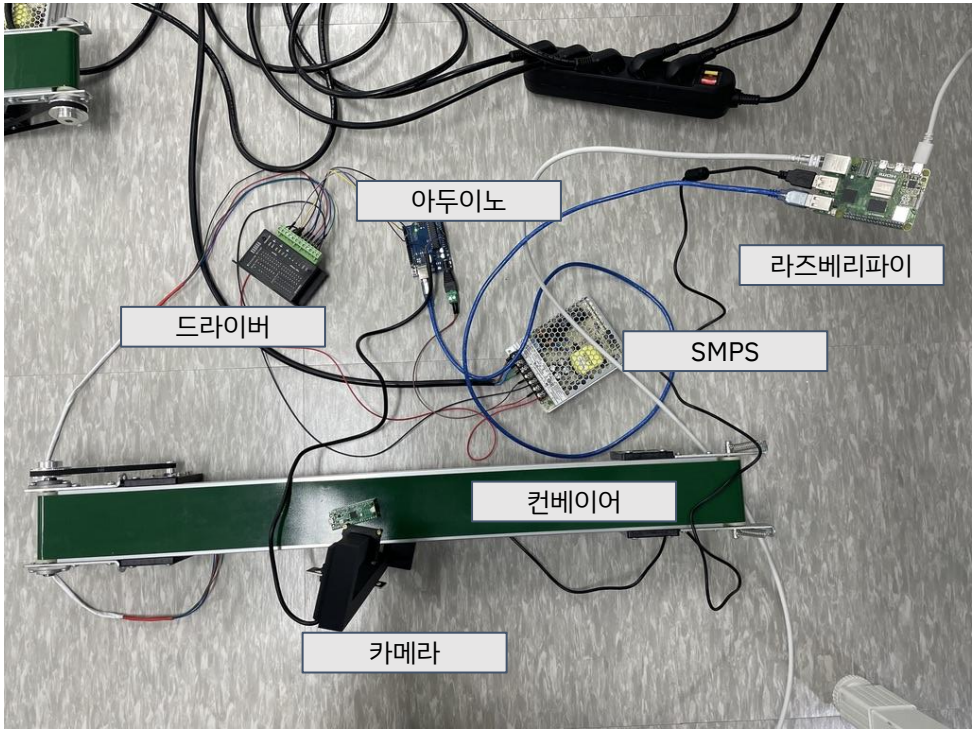
```python
from moviepy.editor import ImageSequenceClip
import os

# 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}")
```

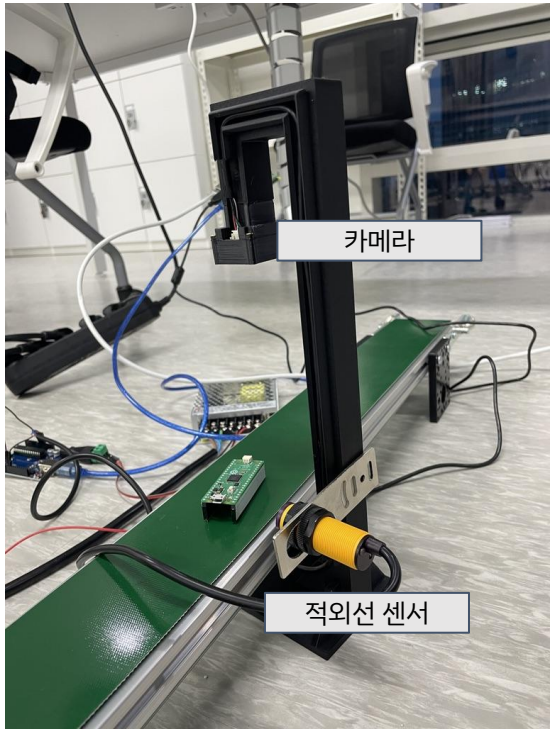
[실습5] Vision AI 를 이용한 외관 검사

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

Vision AI 를 이용한 외관 검사

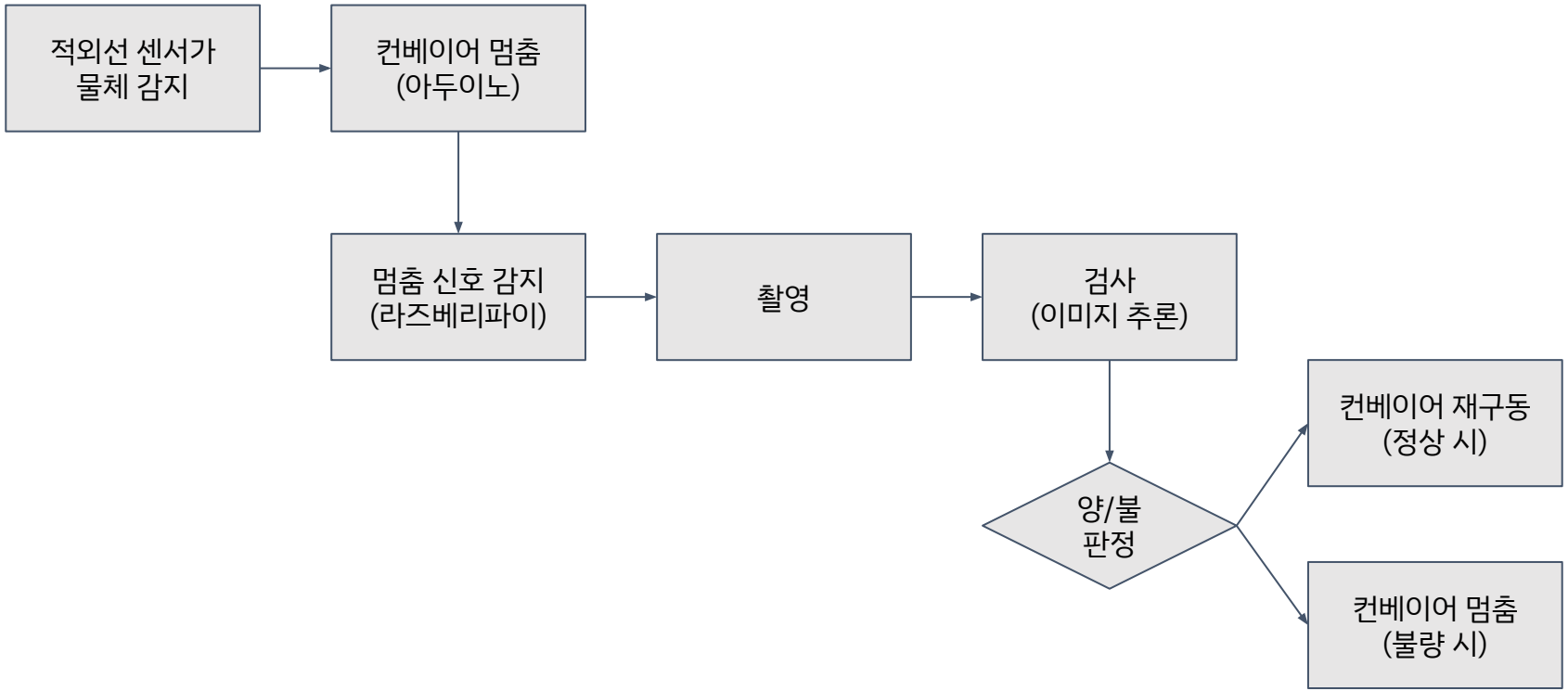
실습 환경





Vision AI 를 이용한 외관 검사

실습 환경



Vision AI 를 이용한 외관 검사

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. 양품과 불량을 샘플을 컨베이어에서 검사하기
3. 검사 결과를 Confusion matrix 로 만들기
4. 정확도 등 지표 계산
5. 모델의 정확도를 높일 수 있는 방법 토의