코멘토 직무부트캠프

업무 요청서

comento

시작 전, 눈높이를 맞춰요

멘티님들은

- 실무의 시작은 대부분 구글링이며, A to Z를 알려주지 않습니다
- 멘토님이 전달하는 업무 내용을 잘 메모하고, 파악이 되지 않은 내용들은 적극적으로 질문하면서 확인해주세요
- 이해가 되지 않는 부분들은 꼭 라이브 세션/오픈카톡방에서 질문해주세요

멘토님은

- 업무의 WHY(배경), WHAT(목표), HOW(수행방법)가 최대한 구체적으로 이해되도록 업무를 요청해주세요
- 멘티님들이 이해하지 못한 부분들이 있는지 한 번 더 확인해주세요
- 직무부트캠프에서 멘토님은 친절한 사수 역할이 되어주세요 👩‍🏫

comento

주차 별 일정 안내

*클래스룸 내 '캠프 일정'에서 정확한 일정 확인 가능

멘티님들은

5주간 3번의 실시간 세션 참여와 4번의 업무를 수행합니다.

멘토님은

1주차 실시간 세션에서 강의와 업무 설명을 제공하고,
이후에 멘티님이 수행한 4번의 업무에 대해
2번의 실시간 피드백 세션과 2번의 서면 피드백을 제공합니다.

1주차

1주차 세션 (ZOOM)

- 직무 에센스 강의 90분
- 질의응답, 업무 설명 30분
- 세션 이후 **1차 업무** 수행

2주차

코멘토 클래스룸

- **1차 업무** 제출
- **1차 업무** 서면 피드백 제공
- **2차 업무** 수행

3주차

3주차 피드백 세션 (ZOOM)

- **2차 업무** 제출 및 발표
- **2차 업무** 실시간 피드백 제공
- **3차 업무** 수행

4주차

코멘토 클래스룸

- **3차 업무** 제출
- **3차 업무** 서면 피드백 제공
- **4차 업무** 수행

5주차

5주차 피드백 세션 (ZOOM)

- **4차 업무** 제출 및 발표
- **4차 업무** 실시간 피드백 제공

업무 제출 안내

제출 기한

1/3차 업무 : 1/3주차 세션일 기준 5일 이내
2/4차 업무 : 1/3차 업무 제출 마감일 기준 7일 이내

*클래스룸 내 '캠프 일정'에서 정확한 일정 확인 가능

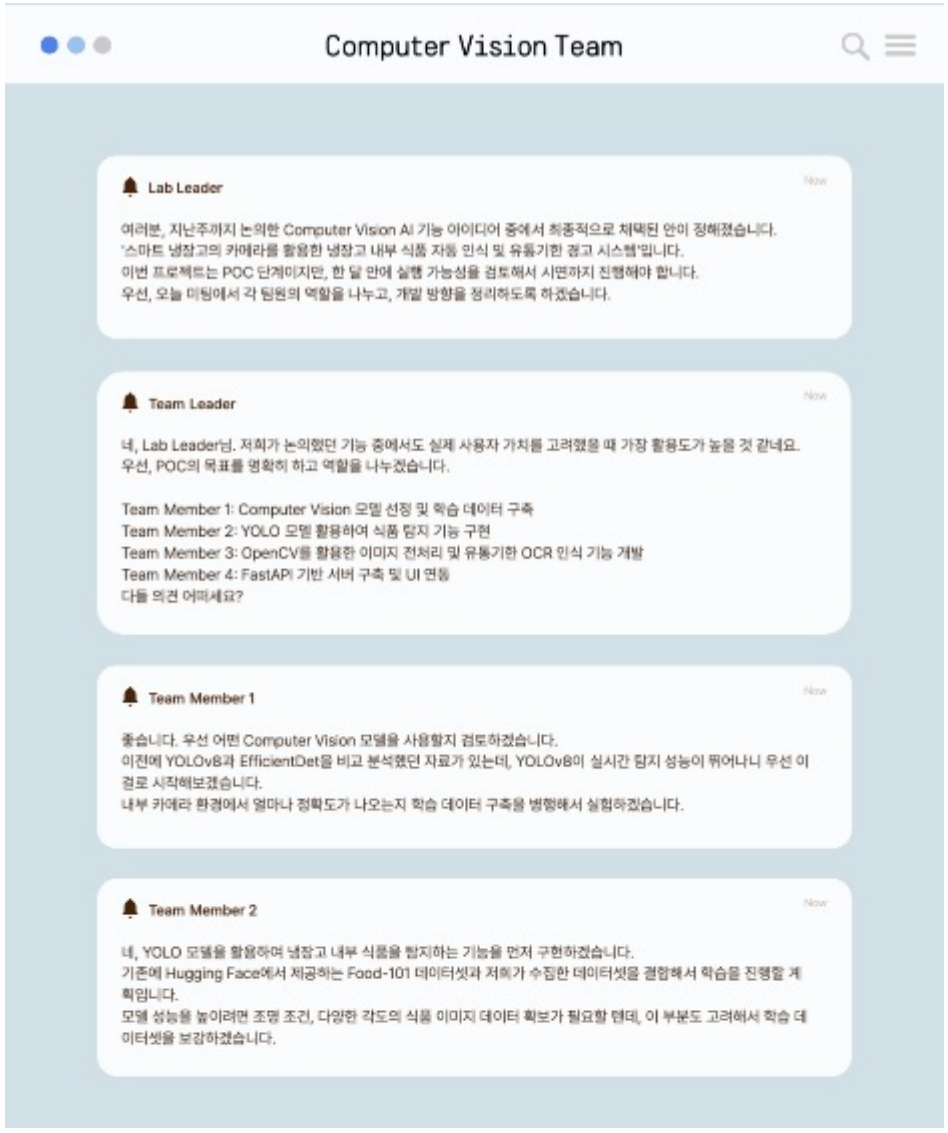
제출 방법

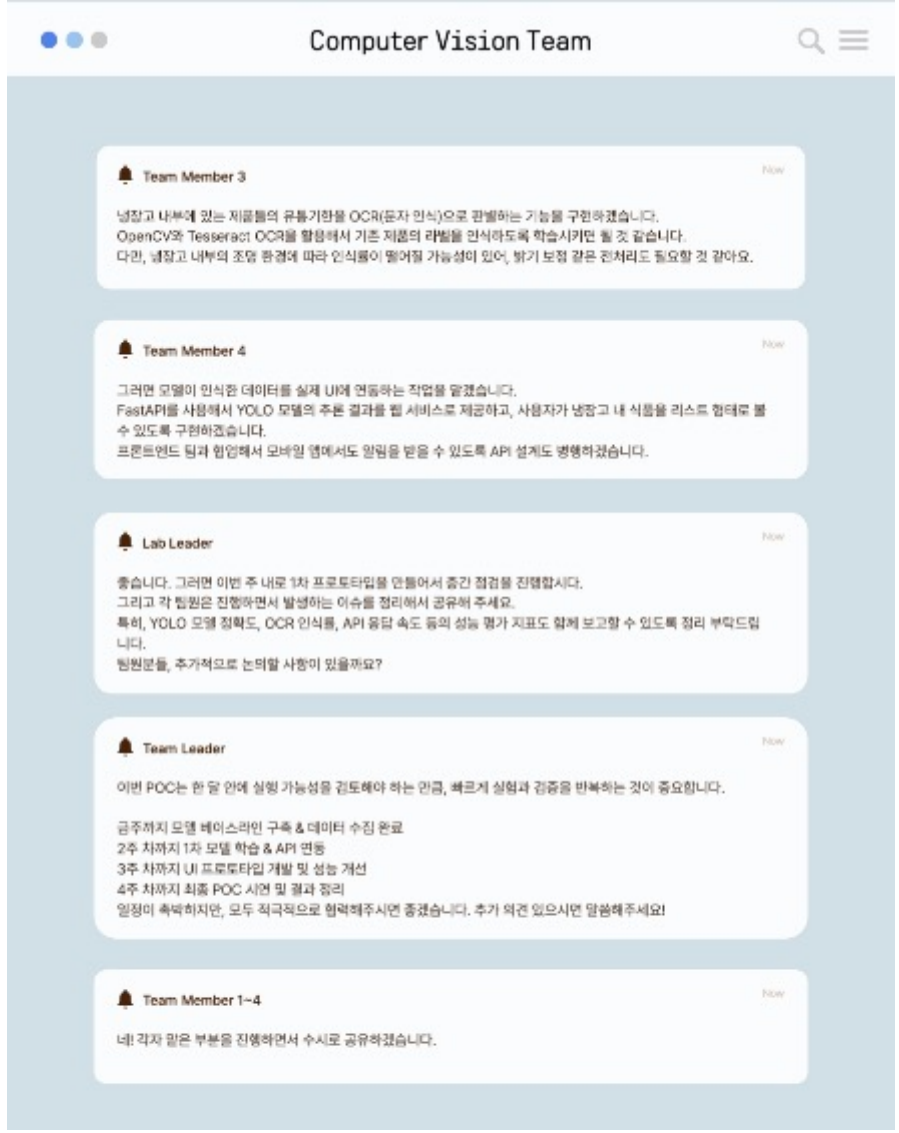
캠프 클래스룸 내 '수강생 업무' 에 업로드

*마이페이지>신청한캠프 에서 클래스룸 입장 가능

1 차 업무

미션 카드 - 업무 메신저





[상황 설정]

- 새로운 AI 기능을 제품에 추가하기 위해 Computer Vision AI 기능을 논의하는 Discussion Meeting이 매일 진행된다.

- 다양한 아이디어 중 하나가 채택되었으며, 한 달 안에 POC(Proof of Concept)를 만들어야 하는 긴급한 프로젝트이다.

- 팀원들은 빠르게 제품 실현 가능성을 검토하고 프로토타입을 개발해야 한다.

- Lab Leader, Team Leader, Team Member(1,2,3,4)가 참석하여 논의한 내용이다.

comento

1 차 업무 안내

[배경]

AI를 접목한 새로운 기술을 가진 제품을 만들기 위한 조직이 신설되면서, 다양한 요구사항들이 들어오고 있습니다. 이러한 요구사항을 충족시키기 위해 상품기획부터 개발 및 상용화까지 프로젝트를 진행해야 합니다. 현재 속해 있는 팀은 Computer Vision 기술을 활용하여 사용자의 편리성을 증대 시키는 제품을 만드는 것이 주요 목표입니다. 이를 위해, 효율적인 코드 관리와 데이터 처리 방식이 필수적이며, Git을 활용한 협업 및 버전 관리 규칙을 적립 해야합니다. 또한, 픽셀 단위 이미지 처리는 Computer Vision 프로젝트의 기초 역량으로 자리 잡고 있습니다. 따라서, 이번 과제에서는 Git을 활용한 코드 관리와 픽셀 단위의 이미지 처리를 실습하여 실무 환경에 적응할 수 있는 기회를 제공합니다.

[주제]

Git을 활용한 코드 관리 및 픽셀 단위 이미지 처리 실습

[요청내용]

1차 과제에서는 Git을 활용한 코드 관리의 기본 개념을 익히고, 픽셀 단위 이미지 처리 기술을 실습해야 합니다. 이를 위해 다음과 같은 업무를 수행해주세요.

1. Git을 활용한 코드 저장소 구성 및 실습

- GitHub 또는 GitLab에 개인 저장소를 생성하고 로컬 환경과 연동합니다.
- git init, git clone 등의 명령어를 활용하여 Git 환경을 설정합니다.
- 새로운 브랜치를 생성하고(git branch feature/image-processing), 코드 변경 사항을 관리합니다.

2. 픽셀 단위 이미지 처리 코드 작성

- OpenCV를 사용하여 제공된 이미지를 로드하고 픽셀 값을 분석합니다.
- 특정 색상의 픽셀을 감지하고 필터링하는 기능을 구현합니다.
- cv2.imread(), cv2.cvtColor(), cv2.threshold() 등의 함수를 사용합니다.

3. Git을 활용한 코드 관리 및 제출

- git commit -m "Initial commit" 등을 활용하여 코드 변경 사항을 관리합니다.
- git push origin feature/image-processing 으로 원격 저장소에 업로드합니다.
- Pull Request(PR)를 생성하고 코드 리뷰 및 Merge 과정을 진행합니다.

comento

1 차 업무 안내

[수행 방법]

1. Git 설정 및 저장소 생성

먼저, GitHub 또는 GitLab에 **새로운 저장소(repository)**를 생성하고 로컬 환경과 연결

1.1 GitHub에서 새로운 저장소 생성

GitHub에 로그인하고 **새 저장소(New Repository)**를 만든다.
저장소 이름을 설정하고 "초기화 안 함(No README, No .gitignore 선택)" 상태로 둔다.
생성된 저장소 URL을 복사한다.

1.2 로컬 환경에서 Git 초기화 & 연동

```
# Git 초기화
> git init
# Github 저장소 복제
> git clone [저장소 URL]
# 원격 저장소의 내용을 로컬로 가져오기 (있을 경우)
> git pull origin main
```

2. Branch 및 Commit 실습

업무를 진행할 새로운 브랜치를 만들고, 변경 사항을 관리

```
# [브랜치 이름] 브랜치 생성
> git branch [브랜치 이름]
# 생성한 브랜치로 이동
> git checkout [브랜치 이름]
```


comento

1 차 업무 안내

[수행 방법]

3. 이미지 처리 코드 작성

OpenCV를 사용하여 이미지를 로드하고, 특정 색상의 픽셀을 감지한 후, 필터링하는 기능을 구현

3.1 필요한 패키지 설치

> pip install opencv-python numpy ...

3.2 예제 코드: 특정 색상 감지 및 필터링

#빨간색 영역을 감지하고 필터링하는 예제 코드

```
import cv2
import numpy as np
# 이미지 로드
image = cv2.imread('sample.jpg')  # 분석할 이미지 파일
# BGR에서 HSV 색상 공간으로 변환
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
# 빨간색 범위 지정 (두 개의 범위를 설정해야 함)
lower_red1 = np.array([0, 120, 70])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 120, 70])
upper_red2 = np.array([180, 255, 255])
# 마스크 생성
mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask = mask1 + mask2  # 두 개의 마스크를 합침
# 원본 이미지에서 빨간색 부분만 추출
result = cv2.bitwise_and(image, image, mask=mask)
# 결과 이미지 출력
cv2.imshow('Original', image)
cv2.imshow('Red Filtered', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

✅ 실행 결과: 빨간색 영역이 검출되며, 다른 색상은 제거된 상태로 표시됨.

comento

1 차 업무 안내

[수행 방법]

4. Git을 활용한 최종 코드 제출

코드를 작성한 후, Git에 변경 사항을 커밋하고 원격 저장소에 업로드해야 해.

변경 사항 확인
> git status
변경된 파일 추가
> git add .
커밋 메시지 작성
> git commit -m "Added image processing script for red color detection"
원격 저장소로 푸시
> git push origin [브랜치 이름]

5. Pull Request(PR) 생성 & 코드 리뷰

팀원들과 협업하기 위해 PR(Pull Request)을 생성하고 코드 리뷰를 진행
GitHub로 이동하여 [브랜치 이름] 브랜치의 PR을 생성.
리뷰어(멘토 또는 동료)를 추가하여 코드 리뷰 요청.
코드 리뷰 후, Merge(병합) 허가가 나면 main 브랜치에 병합.

main 브랜치로 이동
git checkout main
변경 내용 병합
git merge [브랜치 이름]
최종적으로 원격 저장소에 푸시
git push origin main

comento

1 차 업무 안내

[추가 요청]

[기본 문제]

Hugging Face 데이터셋에서 이미지를 가져와 AI 학습을 위한 전처리를 수행하세요. (ex: https://huggingface.co/datasets/ethz/food101)

- 크기 조정 (224×224)
- 색상 변환 (Grayscale & Normalize 적용)
- 노이즈 제거 (Blur 필터 적용)
- 데이터 증강 (좌우 반전, 회전, 색상 변화)

[심화 문제]

이상치를 탐지하여 필터링하는 알고리즘을 추가하세요.

- 너무 어두운 이미지 제거 (평균 밝기 기준)
- 객체 크기가 너무 작은 이미지 제거

[제출 항목]

- image_preprocessing.py (전처리 코드)
- preprocessed_samples/ (처리된 이미지 5장 저장)
- README.md (전처리 과정 설명)

1 차 업무 안내

[결과물 형식]

- Git 저장소 링크 제출
- 코드 및 문서 정리 후 PR(Pull Request) 생성
- PPT 4페이지 이내로 진행 과정 및 결과 요약

[참고 내용]

- Git 기본 사용법 (Commit, Push, Pull, Branch, Merge)
- OpenCV를 활용한 픽셀 단위 이미지 처리 기초
- 협업을 위한 Git Flow 및 코드 리뷰 방법
- 제공된 이미지 데이터셋 활용 방법

2차 업무 안내

[배경]

AI 기반 제품 개발 과정에서 코드의 안정성을 확보하는 것은 필수적이며, 이를 위해 Unit Test가 중요한 역할을 합니다. 또한, 2D 이미지를 3D로 변환하는 기술은 AR/VR, 자율주행, 제조업 등 다양한 산업에서 활용됩니다. 이번 실습에서는 Unit Test를 구성하여 코드의 품질을 높이고, 2D 데이터를 3D로 변환하는 기술을 익히는 것을 목표로 합니다.

[주제]

Unit Test 구성 및 2D → 3D 변환 실습

[요청내용]

1. Unit Test 작성 및 코드 검증

- Python의 unittest 또는 pytest를 활용하여 Unit Test를 구성합니다.
- 테스트 케이스를 작성하여 코드의 기능을 검증합니다.

2. 2D 이미지를 3D로 변환하는 알고리즘 구현

- OpenCV 및 NumPy를 사용하여 2D 데이터를 3D로 변환합니다.
- 기본적인 깊이 맵(Depth Map) 생성 및 변환 과정을 실습합니다.

3. 코드 검증 및 결과 분석

- 작성한 Unit Test를 실행하여 코드가 정상적으로 동작하는지 확인합니다.
- 변환된 3D 이미지의 시각적 결과를 분석하고 개선점을 도출합니다.

comento

2차 업무 안내

[수행 방법]

1. Unit Test 환경 설정

환경 설정 (필요한 라이브러리 설치)
pip install numpy opencv-python pytest

2. 테스트 케이스 작성

Python의 unittest 또는 pytest를 사용하여
코드의 안정성을 검증하는 테스트 케이스를 작성해야 합니다.
이번 과제에서는 2D → 3D 변환 코드의 주요 기능을 테스트하는 것이 목표입니다.

이 테스트 코드를 실행하면:
generate_depth_map() 함수가 정상적으로 동작하는지 검증
입력 이미지가 없을 경우 예외 처리 확인
출력 이미지 크기 & 타입이 정상적인지 테스트

```
#pytest를 활용한 기본 Unit Test

import numpy as np
import pytest
import cv2
# 샘플 함수: 가짜 깊이 맵 생성
def generate_depth_map(image):
    if image is None:
        raise ValueError("입력된 이미지가 없습니다.")
    grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 가짜 깊이 맵 적용
    depth_map = cv2.applyColorMap(grayscale, cv2.COLORMAP_JET)
    return depth_map
# 테스트 코드
def test_generate_depth_map():
    image = np.zeros((100, 100, 3), dtype=np.uint8)  # 검정색 빈 이미지
    depth_map = generate_depth_map(image)

    assert depth_map.shape == image.shape, "출력 크기가 입력 크기와 다릅니다."
    assert isinstance(depth_map, np.ndarray), "출력 데이터 타입이 ndarray가 아닙니다."
# pytest 실행
if __name__ == "__main__":
    pytest.main()
```


comento

2차 업무 안내

[수행 방법]

3. 2D → 3D 변환 알고리즘 구현

2D → 3D 변환을 위해 Depth Map을 생성하고 이를 기반으로 3D 포인트 클라우드를 만드는 과정을 실습합니다.

```
#### 기본적인 Depth Map 생성 코드 (OpenCV 활용)
import cv2
import numpy as np
# 이미지 로드
image = cv2.imread('sample.jpg')
# 그레이스케일 변환
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 깊이 맵 생성 (가상의 깊이 적용)
depth_map = cv2.applyColorMap(gray, cv2.COLORMAP_JET)
# 결과 출력
cv2.imshow('Original Image', image)
cv2.imshow('Depth Map', depth_map)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#### 심화 코드: Depth Map을 기반으로 3D 포인트 클라우드 생성
import cv2
import numpy as np
# 이미지 로드
image = cv2.imread('sample.jpg')
# 그레이스케일 변환
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Depth Map 생성
depth_map = cv2.applyColorMap(gray, cv2.COLORMAP_JET)
# 3D 포인트 클라우드 변환
h, w = depth_map.shape[:2]
X, Y = np.meshgrid(np.arange(w), np.arange(h))
Z = gray.astype(np.float32)  # Depth 값을 Z 축으로 사용
# 3D 좌표 생성
points_3d = np.dstack((X, Y, Z))
# 결과 출력
cv2.imshow('Depth Map', depth_map)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. 결과 검증 및 개선

실제로 pytest를 실행하여 Unit Test가 정상적으로 동작하는지 확인합니다.

pytest test_3d_processing.py

📌 **테스트 결과 예시**

```
============================= test session starts ==============================
collected 2 items
test_3d_processing.py::test_generate_depth_map PASSED [100%]
============================== 1 passed in 0.15s ===============================
```


comento

2차 업무 안내

[결과물 형식]

- Git 저장소 링크 제출
- Unit Test 코드 및 실행 결과 문서화
- 2D → 3D 변환 결과 이미지 첨부
- PPT 4페이지 이내로 과정 및 결과 요약

[참고 내용]

- pytest 및 unittest 활용법
- OpenCV 및 NumPy를 이용한 이미지 변환 기법
- 2D → 3D 변환 알고리즘 및 관련 논문 참고

3차 업무 안내

[배경]

AI와 OpenCV는 Computer Vision에서 핵심적인 도구로 활용되며, 데이터 분석부터 시각화까지 다양한 단계에서 필수적인 역할을 합니다. 특히, AI 기반 모델링은 여러 산업에서 수요가 높은 기술입니다. 이번 과제에서는 AI 모델을 활용하여 데이터를 분석하고, OpenCV를 이용하여 결과물을 시각화하는 실습을 진행합니다.

[주제]

AI 기반 데이터 모델링 및 OpenCV를 활용한 결과 시각화

[요청내용]

1. AI 모델을 활용한 데이터 분석

- 제공된 데이터셋을 활용하여 AI 모델을 학습시키고, 예측 결과를 분석합니다.
- 머신러닝/딥러닝 프레임워크(TensorFlow, PyTorch)를 사용합니다.

2. 이미지에서 객체 탐지 및 패턴 분석

- OpenCV를 사용하여 이미지 내 특정 객체를 탐지하고, 패턴을 분석하는 기능을 구현합니다.
- YOLO, ResNet, ViT 등의 기법을 적용할 수 있습니다.

3. 결과 시각화 및 보고서 작성

- AI 모델의 예측 결과를 시각적으로 표현하고 분석합니다.
- OpenCV의 cv2.imshow() 등을 활용하여 결과를 출력합니다.

comento

3차 업무 안내

[수행 방법]

1. AI 모델 개발 환경 구성

YOLO, PyTorch, OpenCV 등을 사용하여 객체 탐지 모델을 학습시키고 결과를 분석합니다.

> pip install torch torchvision opencv-python matplotlib ultralytics

2. 데이터셋을 활용한 모델 학습

```
#### 예제 코드: YOLOv8을 활용한 객체 탐지 모델 학습
from ultralytics import YOLO
# YOLOv8 모델 로드
model = YOLO("yolov8n.pt")  # YOLOv8 기본 모델 사용
# 사용자 데이터셋으로 학습 (data.yaml 파일 필요)
model.train(data="data.yaml", epochs=10, imgsz=640)
```

```
data.yaml
# 데이터셋 정보 설정
train: ./datasets/train/images  # 학습 데이터 경로
val: ./datasets/valid/images    # 검증 데이터 경로
test: ./datasets/test/images    # 테스트 데이터 경로 (선택 사항)

# 클래스 개수 설정
nc: 3  # 클래스 개수 (예: 3개 클래스)

# 클래스 이름 정의
names: ['person', 'car', 'dog']
```

3. 객체 탐지 및 패턴 분석 코드 구현

모델 학습이 완료되면 실제 이미지에서 객체를 탐지하는 실습을 진행합니다.

```
#### 예제 코드: 학습된 YOLOv8 모델을 사용한 객체 탐지
import cv2
# 학습된 YOLO 모델 로드
model = YOLO("runs/train/exp/weights/best.pt")
# 테스트할 이미지 불러오기
image_path = "test_image.jpg"
image = cv2.imread(image_path)
# 객체 탐지 실행
results = model(image)
# 탐지된 객체 시각화
for result in results:
    for box in result.boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])  # 좌표값 변환
        label = result.names[int(box.cls[0])]  # 클래스 라벨
        confidence = box.conf[0]  # 신뢰도

        # 객체 경계 상자 그리기
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(image, f"{label} ({confidence:.2f})", (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
# 결과 출력
cv2.imshow("YOLO Object Detection", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


comento

3차 업무 안내

[수행 방법]

4. 결과 시각화 및 평가

4-1. 객체 탐지 모델의 성능을 평가하고, 성능 향상을 위한 전략을 적용합니다.

```
# 모델 평가 방법
metrics = model.val()
print(metrics)

# 이 코드 실행 시:
모델의 정확도(AP, Recall, Precision) 출력
```

4-2. 모델 학습 및 예측 결과를 효과적으로 표현하기 위해 시각화 작업을 수행합니다.

```
#### Matplotlib을 활용한 성능 평가 시각화
import matplotlib.pyplot as plt
# 모델 평가 결과
metrics = model.val()
# Precision, Recall 그래프 출력
plt.plot(metrics['precision'], label="Precision")
plt.plot(metrics['recall'], label="Recall")
plt.xlabel("Epochs")
plt.ylabel("Score")
plt.legend()
plt.title("Model Performance")
plt.show()
```

```
# 성능 향상을 위한 방법
데이터 증강(Augmentation): 이미지 회전, 밝기 조절, 노이즈 추가
Hyperparameter Tuning: 학습률 조정, Batch Size 조정
더 깊은 모델 사용: yolov8s.pt, yolov8m.pt 등 더 큰 모델 활용

# 데이터 증강 적용 후 학습
model.train(data="data.yaml", epochs=20, imgsz=640,
augment=True)
→ 20 Epoch 동안 증강된 데이터로 학습 진행
```

3차 업무 안내

[결과물 형식]

- Git 저장소 링크 제출
- AI 모델 학습 코드 및 실행 결과 문서화
- OpenCV를 활용한 결과물 시각화 및 분석 보고서 (PPT 4페이지 이내)

[참고 내용]

- TensorFlow/PyTorch 기반 모델링 기법
- OpenCV를 활용한 객체 탐지 및 이미지 분석 기법
- AI 모델 평가 방법 및 시각화 기법

4차 업무 안내

[배경]

특정 제품이나 소프트웨어를 개발하는 과정은 실제 프로젝트와 가장 유사한 경험을 제공하며, 문제 정의부터 기술 선택, 개발, 결과물 도출까지의 전 과정을 경험할 수 있습니다. 이번 과제에서는 멘티가 원하는 제품 또는 소프트웨어를 선정하여 직접 개발하며, 앞서 배운 Git, Unit Test, AI 모델링 및 OpenCV 활용 기술을 종합적으로 적용합니다.

[주제]

희망하는 제품/SW 선정 및 개발 프로젝트 수행

[요청내용]

1. 개발할 제품 또는 소프트웨어 주제 선정

- AI/Computer Vision을 활용한 기능을 포함하는 주제를 선정합니다.

2. 프로젝트 기획 및 개발 진행

- 요구사항 분석 및 개발 일정 수립
- 프로젝트에 필요한 기술 스택 결정 및 개발 진행

3. 결과물 구현 및 발표

- 개발된 기능을 시연하고 프로젝트 결과물을 발표합니다.
- Git을 활용하여 코드 관리 및 최종 문서 작성

comento

4차 업무 안내

[수행 방법]

1. 개발할 주제 선정 및 기획 문서 작성

- 프로젝트 목표, 요구사항, 기술 스택 정리

2. Git을 활용한 코드 개발 및 관리

- 기능 구현 후 Unit Test 작성 및 실행

3. 최종 결과물 개발 및 테스트

- AI 모델 및 OpenCV 기능을 활용한 소프트웨어 구현

4. 결과물 발표 및 보고서 제출

- 프로젝트 수행 과정과 결과를 PPT로 정리하여 발표

[결과물 형식]

- Git 저장소 링크 제출
- 개발된 소프트웨어 및 실행 가능 파일 제공
- 프로젝트 기획서 및 결과 보고서 (PPT 4페이지 이내)

[참고 내용]

- 프로젝트 기획 및 요구사항 정의 방법론
- Git을 활용한 협업 및 코드 관리
- AI 및 Computer Vision을 활용한 소프트웨어 개발 사례

참고 자료

주제	설명	링크
Git & GitHub 기초	Git 기본 사용법, 브랜치, PR, 협업 방법	Pro Git Book
OpenCV 기본 문서	OpenCV 함수 설명, 이미지 처리 기법	OpenCV 공식 문서
YOLO 객체 탐지 기초	YOLOv8 모델 개요 및 사용법	Ultralytics YOLO Docs
데이터 증강 기법	AI 모델 학습을 위한 데이터 증강 기법	Albumentations
Python Unit Test	unittest & pytest 활용법	pytest 공식 문서
OpenCV로 Depth Map 생성	2D 이미지를 3D로 변환하는 과정	OpenCV Depth Estimation
Point Cloud 데이터	3D 포인트 클라우드 데이터 구조	Open3D 공식 문서
Depth Map을 활용한 3D 변환	Stereo Vision & Depth Map 활용 기법	Stereo Vision with OpenCV
YOLO 모델 학습 및 추론	YOLOv8 모델 사용법	YOLOv8 Ultralytics Docs
PyTorch 기반 객체 탐지	YOLO, Faster R-CNN 등 다양한 모델 활용	Torchvision Object Detection
ResNet & ViT 기반 분류 모델	Vision Transformer 기본 개념	Hugging Face ViT
OpenCV 객체 탐지	Haar Cascade & CNN 활용	OpenCV Object Detection
Git 협업 & PR 리뷰	실무 Git 브랜치 전략 및 협업 방식	Git Flow
AI 기반 프로젝트 기획	AI 제품 개발 프로세스 및 요구사항 분석	AI Project Planning
Flask/FastAPI 백엔드 구축	AI 모델 배포를 위한 API 개발	FastAPI 공식 문서