실시간 객체 탐지를 위한 인공지능 플랫폼 활용

Roboflow, Ultralytics

Realtime Detection & Tracking

Dataset, Training, Deploy

이 슬라이드에서 사용한 서체 :
- Open Sans(https://ko.cooltext.com/Download-Font-Open+Sans)
- KoPubWorld돋움체(http://www.kopus.org/biz/electronic/font.aspx)

실시간 객체 탐지를 위한 인공지능 플랫폼

Roboflow

컴퓨터 비전 모델을 개발, 훈련 및 배포하는 데 도움을 주는 플랫폼

데이터 어노테이션 및 객체탐지 모델에 맞는 어노테이션 데이터를 내려받을 수 있음

예: 훈련된 모델을 가져와 객체 탐지에 사용할 수 있음

Ultralytics

YOLO(You Only Look Once) 객체 탐지 모델의 최신 버전을 개발 및 유지 관리

YOLOv5, YOLOv8, YOLOv11 등 객체탐지 모델을 쉽게 사용할 수 있음

예: 로보플로우에서 내려받은 데이터를 이용해서 직접 모델을 학습시킬 수 있음

LangChain

언어 모델을 연결하고 결합하여 복잡한 작업을 수행할 수 있도록 하는 프레임워크

OpenAPI를 더 쉽게 사용할 수 있도록 함

예: OpenAI의 Chat GPT API를 사용하여 애플리케이션을 쉽게 개발할 수 있음

허깅페이스는 자연어처리 모델인 Transformer를 제공하는 플랫폼입니다.

- 물론 자연어 처리를 위한 인공지능 모델만 제공하는 것은 아닙니다.
- DETR등 객체 탐지 모델도 제공합니다.

Roboflow(roboflow.com)

실시간 객체 탐지를 위한 인공지능 플랫폼 활용

컴퓨터 비전 모델을 개발, 훈련 및 배포하는 데 도움을 주는 플랫폼

- 객체 탐지, 분류, 세분화 작업에 유용한 도구와 서비스를 제공
- 데이터 준비, 모델 훈련, 성능 평가, 배포 등 컴퓨터 비전 워크플로우의 모든 단계를 지원함

Roboflow의 주요 특징

데이터셋 관리(데이터셋 업로드, 데이터 레이블링, 데이터 증강)

- 다양한 형식의 이미지 데이터를 쉽게 업로드할 수 있음.
- 이미지에 라벨을 추가하거나 수정할 수 있는 인터페이스를 제공함.
- 데이터 증강 기법을 사용하여 훈련 데이터를 늘리고 모델의 일반화 성능을 향상시킴

모델 훈련(자동 모델 빌더, 커스텀 모델 훈련, 프리빌트 모델 사용)

- 사용자가 선택한 데이터셋을 기반으로 최적의 모델을 자동으로 설정하고 훈련함
- 사용자가 원하는 설정과 파라미터로 모델을 훈련할 수 있음
- 사전 훈련된 모델을 사용하여 빠르게 시작할 수 있습니다.

모델 배포(웹 인터페이스, API, 엣지 디바이스 배포)

- 모델을 웹 애플리케이션으로 배포하여 실시간 추론을 수행할 수 있음
- REST API를 통해 모델을 쉽게 통합하고 사용할 수 있음
- Raspberry Pi와 같은 엣지 디바이스에 모델을 배포할 수 있음

성능 평가(평가 도구, 시각화)

- 모델의 성능을 평가하고, 정밀도, 재현율, F1 점수 등을 확인할 수 있음
- 모델의 예측 결과를 시각화하여 분석할 수 있음

협업 기능(팀워크, 프로젝트 관리)

- 팀 단위로 협업하여 데이터셋과 모델을 관리할 수 있음
- 여러 프로젝트를 관리하고, 각 프로젝트별로 진행 상황을 추적할 수 있음.

통합 및 호환성(다양한 포맷 지원, 기존 툴과의 통합)

- COCO, Pascal VOC, YOLO 등 다양한 데이터셋 포맷을 지원
- TensorFlow, PyTorch 등 기존의 AI 프레임워크와 쉽게 통합할 수 있음

Roboflow 시작하기

Roboflow 활용하기

https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw

pip install inference-sdk

모델 가져와 예측하기

```
1. from inference_sdk import InferenceHTTPClient
2. CLIENT = InferenceHTTPClient(
3.     api_url="https://detect.roboflow.com",
4.     api_key="HKU3a7NuEDStiIgEbg9W"
5. )
6.
7. result = CLIENT.infer('sample.png', model_id="rock-paper-scissors-sxsw/14")
8. print(result)
```

```
{'inference_id': '70f00cc2-62c2-4c3e-a9a1-cfd77be70584', 'time': 0.03703791099999876, 'image': {'width': 761,
'height': 568}, 'predictions': [{'x': 526.0, 'y': 445.0, 'width': 204.0, 'height': 198.0, 'confidence': 0.9346
706274185181, 'class': 'Rock', 'class_id': 1, 'detection_id': '983bd847-a2bb-4e3a-9cac-2a7b841c7cb3'}, {'x': 5
': 158.5, 'width': 274.0, 'height': 25
2, 'detection_id':
': 138.0, 'confiden
-ba2d-a41aea63de2c'
'y': 428.5, 'width': 275.0, 'height': 253.0, 'confidence': 0.4923230
1115989685, 'class': 'Scissors', 'class_id': 2, 'detection_id': 'd0675fb1-1f21-47b8-87f2-a06623cdf3d7'}]}
```

탐지한 객체가 리스트로 저장 되어 있음

예측한 클래스 이름

사각형의 중심 위치(x,y)

사각형의 가로 세로 크기

신뢰도

Roboflow 시작하기

Roboflow 활용하기

바운딩박스 그리기

pip install opencv-python

```
1. import cv2
2.
3. filename = 'sample.png'
4. result = CLIENT.infer(filename, model_id="rock-paper-scissors-sxsw/14")
5.
6. img = cv2.imread(filename)
7. for pred in result["predictions"]:
8.     x, y = pred['x'], pred['y'] # 사각형의 중심위치
9.     width, height = pred['width'], pred['height']
10.    conf = pred['confidence']
11.    cls = pred['class']
12.    x1, y1 = int(x-width/2), int(y-height/2)
13.    x2, y2 = int(x+width/2), int(y+height/2)
14.    cv2.rectangle(img, (x1,y1), (x2, y2), (0,0,255), 2)
15.    cv2.putText(img, f'{cls} {conf:.4f}', (x1,y1), cv2.FONT_HERSHEY_PLAIN, 2, (0,0,255))
16.
17.cv2.imshow('image', img)
18.cv2.waitKey()
19.cv2.destroyAllWindows()
```

Rebuild the library with Windows, GTK+ 2.x or Cocoa support. If you are on Ubuntu or Debian, install libgtk2.0-dev and pkg-config, then re-run cmake or configure script in function 'cvShowImage' 오류 발생시Opencv를 재설치 해주세요. 그래도 오류가 발생하면 opencv-python-headless를 삭제해 주세요.

- pip uninstall opencv-python-headless

파이썬에서 Roboflow 모델 사용하기

pip install roboflow supervision

Model 가져오기

모델을 가져와서

```
1. from roboflow import Roboflow
2.
3. rf = Roboflow(api_key="HKU3a7NuEDStiIgEbg9W")
4. project = rf.workspace().project("rock-paper-scissors-sxsw")
5. model = project.version(14).model
```

파이썬에서 Roboflow 모델 사용하기

예측 후 시각화하기

```
1. import supervision as sv
2. import cv2
3.
4. filename = 'sample.png'
5.
6. result = model.predict(filename, confidence=40, overlap=30).json()
7. labels = [item["class"] for item in result["predictions"]]
8.
9. detections = sv.Detections.from_inference(result)
10.
11.label_annotator = sv.LabelAnnotator()
12.bounding_box_annotator = sv.BoundingBoxAnnotator()
13.
14.image = cv2.imread(filename)
15.annotated_image = bounding_box_annotator.annotate(scene=image, detections=detections)
16.annotated_image = label_annotator.annotate(
       scene=annotated_image, detections=detections, labels=labels)
17.
18.sv.plot_image(image=annotated_image, size=(16, 16))
```

객체 탐지 결과를 시각화 하기 위해 supervision 모듈을 사용할 수 있습니다.

파이썬에서 Roboflow 모델 사용하기

Roboflow 활용하기

OpenCV로 보여주기

```
1. import cv2
2.
3. filename = 'sample.png'
4. result = model.predict(filename, confidence=40, overlap=30)
5.
6. img = cv2.imread(filename)
7. for pred in result:
8.     x, y = pred['x'], pred['y']
9.     width, height = pred['width'], pred['height']
10.    conf = pred['confidence']
11.    cls = pred['class']
12.    x1, y1 = int(x-width/2), int(y-height/2)
13.    x2, y2 = int(x+width/2), int(y+height/2)
14.    cv2.rectangle(img, (x1,y1), (x2, y2), (0,0,255), 2)
15.    cv2.putText(img, f'{cls} {conf:.4f}', (x1,y1), cv2.FONT_HERSHEY_PLAIN, 2, (0,0,255))
16.
17.cv2.imshow('image', img)
18.cv2.waitKey()
19.cv2.destroyAllWindows()
```

이 예는 OpenCV를 이용해서
객체탐지 결과를 시각화 합니다.

Ultralytics(ultralytics.com)

실시간 객체 탐지를 위한 인공지능 플랫폼 활용

YOLO(You Only Look Once) 객체 탐지 모델의 최신 버전 개발 및 유지 관리

- Ultralytics의 YOLOv5, YOLOv8, YOLOv11 등 모델은 특히 인공지능 기반 객체 탐지 분야에서 널리 사용되고 있음
- Ultralytics는 모델을 손쉽게 사용할 수 있도록 다양한 기능과 도구를 제공

Ultralytics YOLO의 주요 특징

- **사용자 친화적인 인터페이스**: 모델을 훈련하고 평가할 수 있도록 다양한 기능을 제공하는 라이브러리를 제공. 사용자는 몇 줄의 코드만으로 강력한 객체 탐지 모델을 사용할 수 있음.
- **높은 성능**: YOLO 모델은 높은 속도와 정확도를 자랑함. 특히, YOLOv5, YOLOv8, YOLOv11 등은 실시간 객체 탐지가 가능할 정도로 빠르며, 다양한 환경에서 높은 성능을 발휘함.
- **다양한 모델 버전**: 다양한 크기와 성능의 모델을 제공하여 사용자가 자신의 필요에 맞는 모델을 선택할 수 있도록 함. 예를 들어, YOLOv8은 nano, small, medium, large 등 여러 버전으로 제공됨.
- **풍부한 기능**: 모델 학습을 위한 데이터 증강, 앵커 프리 탐지, 고급 손실 함수 등 다양한 기능이 포함되어 있음. 이러한 기능들은 모델의 성능을 향상시키고 다양한 용도에 적합하게 함.
- **커뮤니티 및 지원**: 강력한 커뮤니티 지원을 제공하며, 다양한 문서와 튜토리얼을 통해 사용자가 모델을 쉽게 이해하고 활용할 수 있도록 도움.

pip install ultralytics

Ultralytics의 YOLOv8

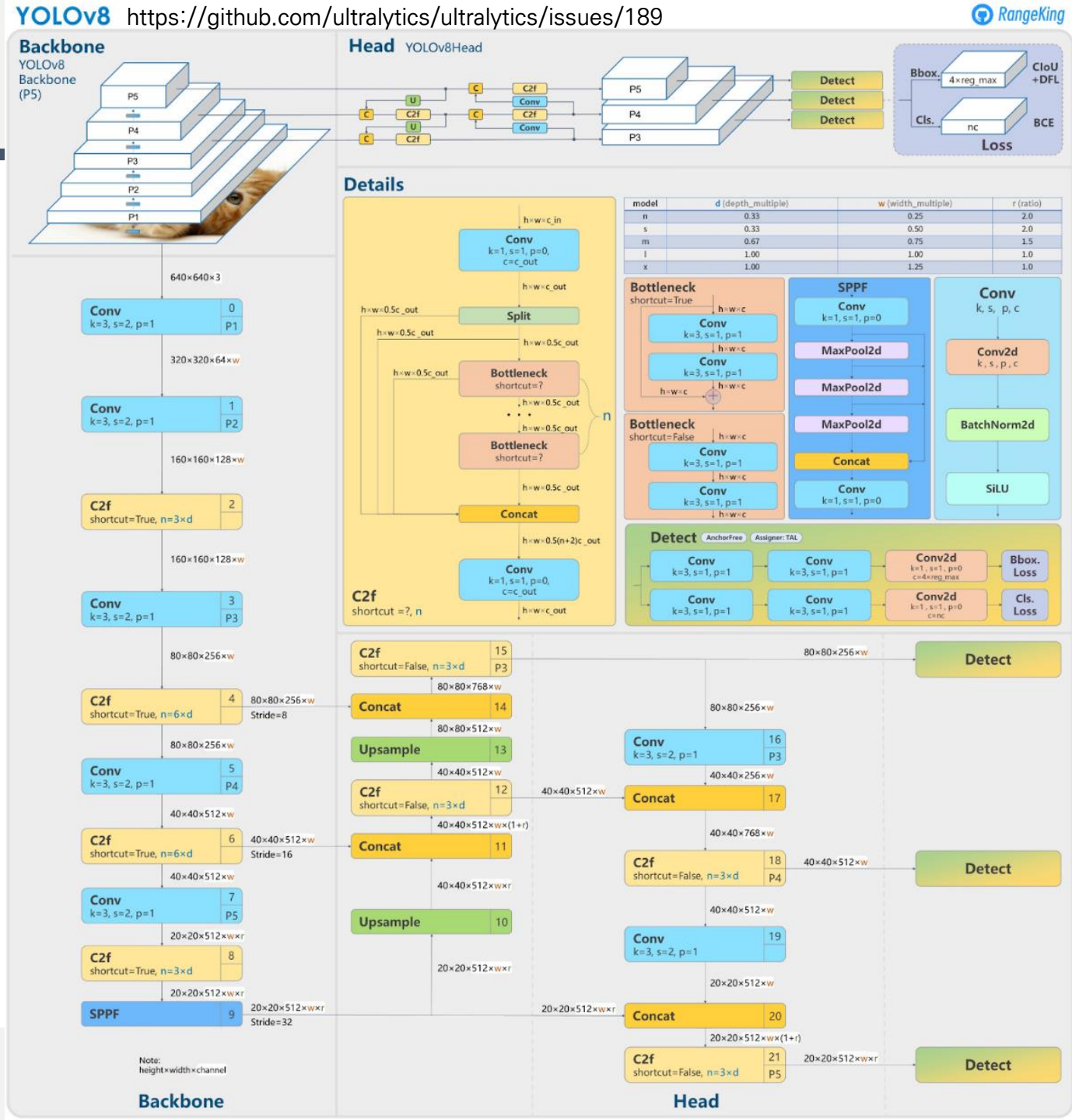
Ultralytics YOLOv8 활용하기

YOLOv8의 주요 기능

- Mosaic Data Augmentation
- Anchor-Free Detection
- Coarse-to-Fine(C2f) Module
- Decoupled Head
- Modified Loss Function

주요 사이트

- https://github.com/ultralytics
- https://docs.ultralytics.com/



YOLOv8의 주요 기능

Ultralytics YOLOv8 활용하기

Mosaic Data Augmentation

훈련 데이터의 다양성을 높이고, 모델의 일반화 성능을 향상시키기 위해 사용되는 기술
이 기법은 네 개의 이미지를 결합하여 하나의 이미지로 만드는 방식으로, 다음과 같은 특징이 있음

- 네 개의 서로 다른 이미지 조각을 합쳐 하나의 큰 이미지를 구성.
- 다양한 배경과 객체 조합을 제공하여 모델이 다양한 상황에 대응할 수 있도록 함.
- 객체의 크기와 위치가 다양해져 모델이 더 일반화된 특징을 학습할 수 있음.

Anchor-Free Detection

기존의 앵커 기반 탐지 방식의 단점을 보완하기 위해 도입된 방식

- 기존 YOLO 모델은 미리 정의된 앵커 박스를 사용하여 객체의 위치와 크기를 예측했음
- 앵커 프리 방식은 특정 앵커 박스에 의존하지 않고, 네트워크가 직접 객체의 중심점과 크기를 예측함.
- 이를 통해 앵커 박스 설정의 복잡성을 줄이고, 다양한 크기와 비율의 객체를 더 효율적으로 탐지할 수 있음.

Coarse-to-Fine(C2f) Module

객체 탐지의 정확성을 높이기 위해 도입된 단계적 접근 방식

- 초기 단계에서 전체 이미지의 거친(대략적인) 특징을 추출하여 큰 객체와 주요 영역을 식별.
- 이후 단계에서 더 세밀한 특징을 추출하여 작은 객체와 디테일을 더 정확하게 탐지함.
- 이 방식은 탐지의 효율성을 높이고, 특히 복잡한 장면에서의 탐지 성능을 향상시킴.

YOLOv8의 주요 기능

Ultralytics YOLOv8 활용하기

Decoupled Head

분류와 회귀 작업을 분리하여 각각의 작업에 최적화된 출력을 제공하는 구조

- 기존 YOLO 모델에서는 분류와 회귀를 동시에 수행하는 단일 헤드를 사용함
- Decoupled Head는 분류와 회귀를 위한 별도의 헤드를 사용하여 각각의 작업에 맞는 최적의 학습을 진행함.
- 이를 통해 객체의 클래스 예측과 위치 예측의 정확도를 향상시킴

Modified Loss Function

모델의 학습 성능을 높이기 위해 사용

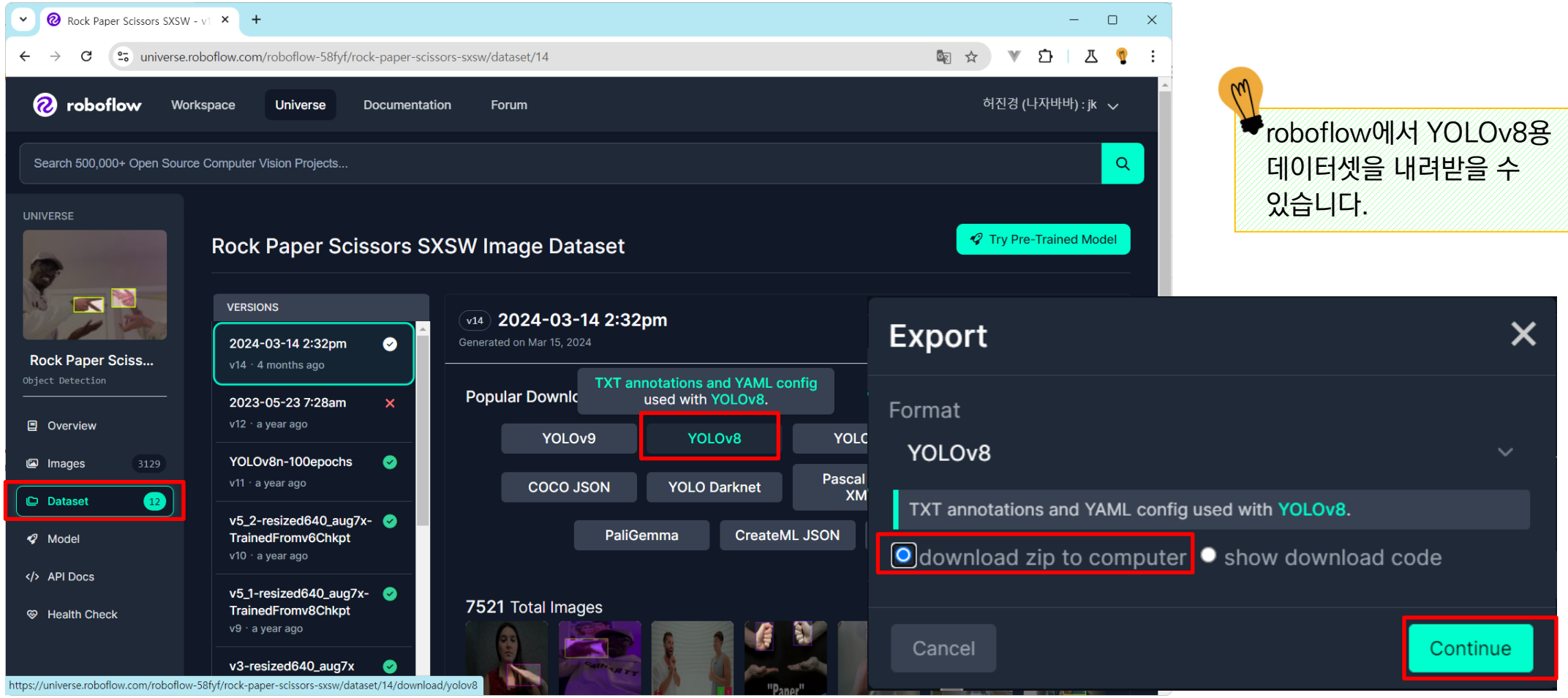
- 기존 손실 함수의 단점을 보완하여 더 효과적으로 객체 탐지를 수행할 수 있도록 설계되었음.
- 예를 들어, 예측된 박스와 실제 박스의 차이를 계산하는 방법이나 클래스 불균형 문제를 해결하는 방법을 개선함.
- 새로운 손실 함수는 모델이 더 빠르고 정확하게 학습할 수 있도록 도움.

Model	Size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

roboflow에서 데이터 내려받기

Ultralytics YOLOv8 활용하기

https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw



roboflow에서 YOLOv8용 데이터셋을 내려받을 수 있습니다.

YOLOv8 사용하기

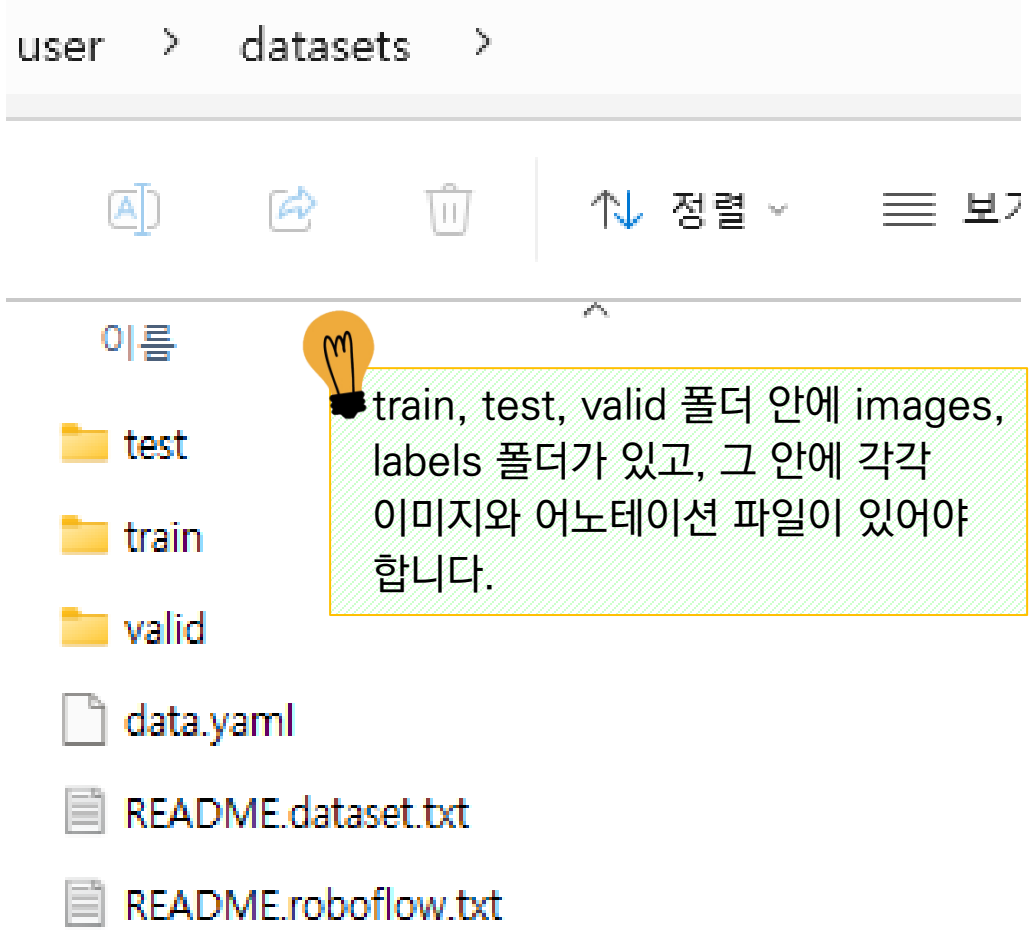
Ultralytics YOLOv8 활용하기

pip install ultralytics

YOLOv8로 학습하기

```
from ultralytics import YOLO

# YOLOv8 모델 로드
model = YOLO("yolov8n.pt")

# 데이터셋 경로 설정
data_path = "datasets/data.yaml"

# 모델 훈련
model.train(data=data_path, epochs=100, imgsz=640)
```

데이터 다운로드: https://universe.roboflow.com/roboflow-58fvf/rock-paper-scissors-sxsw



YOLOv8 사용하기

Ultralytics YOLOv8 활용하기

YOLOv8로 학습하기

```
1. result = model.predict('sample.png')
2. print(result[0].boxes)
```

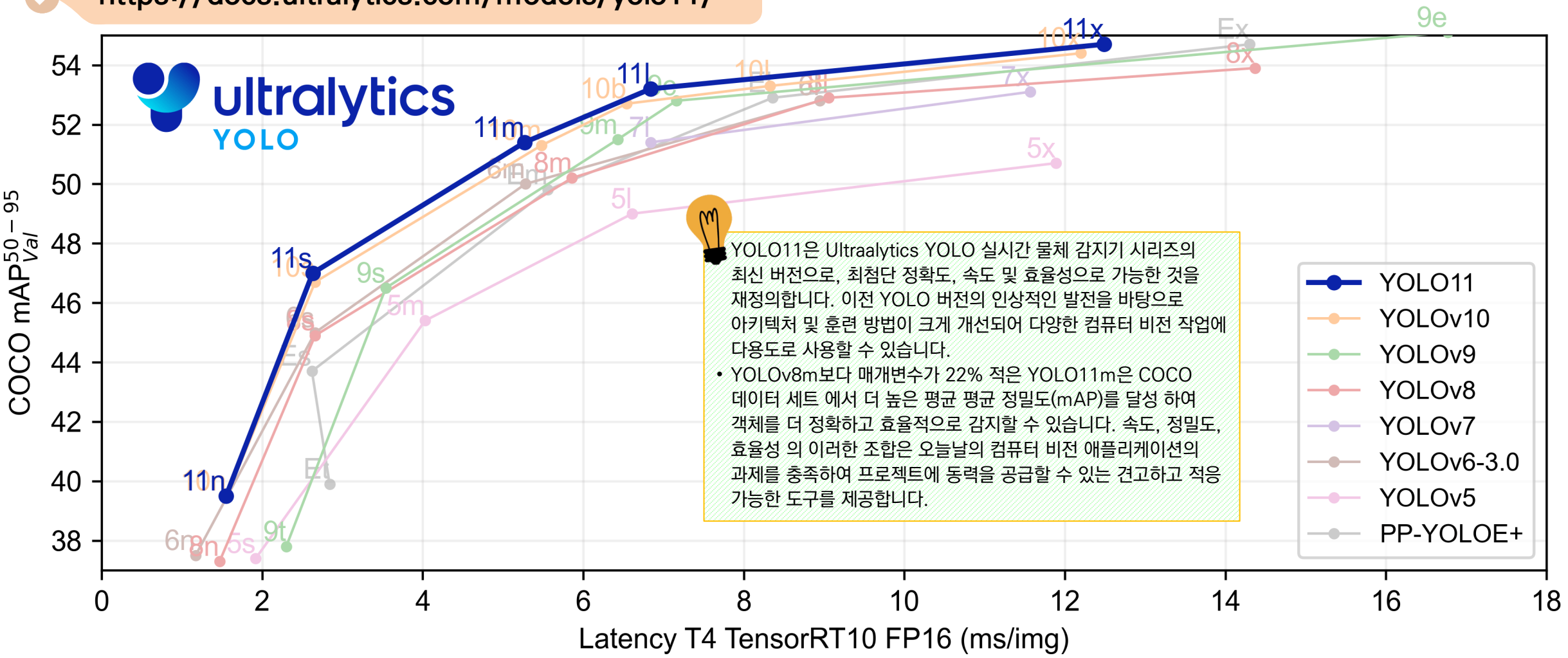
```
ultralytics.engine.results.Boxes object with attributes:

cls: tensor([0.])
conf: tensor([0.8386])
data: tensor([[ 10.4947, 303.9644, 369.7246, 553.6914,   0.8386,   0.0000]])
id: None
is_track: False
orig_shape: (568, 761)
shape: torch.Size([1, 6])
xywh: tensor([[190.1097, 428.8279, 359.2299, 249.7270]])
xywhn: tensor([[0.2498, 0.7550, 0.4720, 0.4397]])
xyxy: tensor([[ 10.4947, 303.9644, 369.7246, 553.6914]])
xyxyn: tensor([[0.0138, 0.5351, 0.4858, 0.9748]])
```

```
1. import cv2
2. import torch
3.
4. filename = 'sample.png'
5. img = cv2.imread(filename)
6.
7. for box in result[0].boxes:
8.     x1, y1, x2, y2 = box.xyxy[0].numpy().flatten().astype(int)
9.     cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
10.    cv2.putText(img, f'{int(box.conf)} {float(box.conf):.4f}', (x1, y1),
   cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 255))
11.
12.cv2.imshow('image', img)
13.cv2.waitKey()
14.cv2.destroyAllWindows()
```

YOLOv11의 새로운 기능

Ultralytics YOLOv11 활용하기

https://docs.ultralytics.com/models/yolo11/



YOLOv11 주요 기능

Ultralytics YOLOv11 활용하기

향상된 특징 추출

- YOLO11은 향상된 백본 및 넥 아키텍처를 사용하여 보다 정확한 객체 감지 및 복잡한 작업 수행을 위해 특징 추출 기능을 향상시킴.

효율성과 속도에 최적화

- YOLO11은 정제된 아키텍처 설계와 최적화된 교육 파이프라인을 도입하여 더 빠른 처리 속도를 제공하고 정확도와 성능 간의 최적의 균형을 유지함.

더 적은 수의 매개변수

- 모델 설계의 발전으로 YOLO11m은 COCO 데이터 세트에서 더 높은 평균 정밀도(mAP)를 달성하는 동시에 YOLOv8m보다 22% 적은 매개변수를 사용하여 정확도를 저하시키지 않으면서 계산 효율성을 높임.

환경 전반에 걸친 적응성

- YOLO11은 엣지 디바이스, 클라우드 플랫폼, NVIDIA GPU를 지원하는 시스템 등 다양한 환경에 원활하게 배포되어 최대한의 유연성을 보장함.

광범위한 지원 작업

- 객체 감지, 인스턴스 분할, 이미지 분류, 포즈 추정 또는 지향적 객체 감지(OBB) 등 다양한 컴퓨터 비전 문제를 충족하도록 설계된 YOLO11은 다양한 컴퓨터 비전 문제를 충족하도록 설계되었음.

YOLOv11 지원 모델

Ultralytics YOLOv11 활용하기

Model	Filenames	Task	Inference	Validation	Training	Export
YOLO11	yolo11n.pt yolo11s.pt yolo11m.pt yolo11l.pt yolo11x.pt	Detection	✅	✅	✅	✅
YOLO11-seg	yolo11n-seg.pt yolo11s-seg.pt yolo11m-seg.pt yolo11l-seg.pt yolo11x-seg.pt	Instance Segmentation	✅	✅	✅	✅
YOLO11-pose	yolo11n-pose.pt yolo11s-pose.pt yolo11m-pose.pt yolo11l-pose.pt yolo11x-pose.pt	Pose/Keypoints	✅	✅	✅	✅
YOLO11-obb	yolo11n-obb.pt yolo11s-obb.pt yolo11m-obb.pt yolo11l-obb.pt yolo11x-obb.pt	Oriented Detection	✅	✅	✅	✅
YOLO11-cls	yolo11n-cls.pt yolo11s-cls.pt yolo11m-cls.pt yolo11l-cls.pt yolo11x-cls.pt	Classification	✅	✅	✅	✅

YOLO11은 YOLOv8에 도입된 다용도 모델 제품군을 기반으로 다양한 컴퓨터 비전 작업에서 향상된 지원을 제공합니다.

YOLOv11 성능 지표

Ultralytics YOLOv11 활용하기

Detection (COCO)

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Segmentation (COCO)

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n-seg	640	38.9	32.0	65.9 ± 1.1	1.8 ± 0.0	2.9	10.4
YOLO11s-seg	640	46.6	37.8	117.6 ± 4.9	2.9 ± 0.0	10.1	35.5
YOLO11m-seg	640	51.5	41.5	281.6 ± 1.2	6.3 ± 0.1	22.4	123.3
YOLO11l-seg	640	53.4	42.9	344.2 ± 3.2	7.8 ± 0.2	27.6	142.2
YOLO11x-seg	640	54.7	43.8	664.5 ± 3.2	15.8 ± 0.7	62.1	319.0

Classification (ImageNet)

Model	size (pixels)	acc top1	acc top5	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B) at 640
YOLO11n-cls	224	70.0	89.4	5.0 ± 0.3	1.1 ± 0.0	1.6	3.3
YOLO11s-cls	224	75.4	92.7	7.9 ± 0.2	1.3 ± 0.0	5.5	12.1
YOLO11m-cls	224	77.3	93.9	17.2 ± 0.4	2.0 ± 0.0	10.4	39.3
YOLO11l-cls	224	78.3	94.3	23.2 ± 0.3	2.8 ± 0.0	12.9	49.4
YOLO11x-cls	224	79.5	94.9	41.4 ± 0.9	3.8 ± 0.0	28.4	110.4

YOLO11은 YOLOv8에 도입된 다용도 모델 제품군을 기반으로 다양한 컴퓨터 비전 작업에서 향상된 지원을 제공합니다.

사용 방법

Ultralytics YOLOv11 활용하기

```
from ultralytics import YOLO

# Load a COCO-pretrained YOLO11n model
model = YOLO("yolo11n.pt")

# Train the model on the COCO8 example dataset for 100 epochs
results = model.train(data="coco8.yaml", epochs=100, imgsz=640)

# Run inference with the YOLO11n model on the 'bus.jpg' image
results = model("path/to/bus.jpg")
```

```
@software{yolo11_ultralytics,
  author = {Glenn Jocher and Jing Qiu},
  title = {Ultralytics YOLO11},
  version = {11.0.0},
  year = {2024},
  url = {https://github.com/ultralytics/ultralytics},
  orcid = {0000-0001-5950-6979, 0000-0002-7603-6750, 0000-0003-3783-7069},
  license = {AGPL-3.0}
}
```

YOLOv5, YOLOv8, YOLOv11은 AGPL-3.0라이선스와 Ultralytics Enterprise 라이선스하에 제공됩니다.

- AGPL-3.0 라이선스(Affero General Public License v3.0)는 SaaS(Software as a Service)와 같은 형태로 소프트웨어를 제공하는 경우에도 그 소스 코드를 공개해야 하는 의무를 추가합니다. 따라서 AGPL은 특히 서버 기반 애플리케이션이나 웹 애플리케이션에서 사용될 때 그 소스 코드의 공개를 보장합니다.
- AGPL-3.0 라이선스는 네트워크를 통해 제공되는 서비스로 사용될 가능성이 있는 소프트웨어에 적합하며, 이 라이선스를 따르는 소프트웨어를 사용할 경우 해당 소프트웨어를 수정하여 내부 서비스로 사용하더라도, 서비스 사용자가 요청하면 소스 코드를 제공해야 할 수 있습니다. 이는 기업이나 서비스 제공자가 사용 시 중요한 법적 고려 사항이 될 수 있습니다.
- 엔터프라이즈 라이선스는 상업적 사용을 위해 설계되었으며, 이 라이선스는 AGPL-3.0 의 오픈 소스 요건을 우회하여 Ultralytics 소프트웨어 및 AI 모델을 상업용 상품 및 서비스에 원활하게 통합할 수 있도록 허용합니다.
- https://www.ultralytics.com/license

YOLO 버전별 특징 및 라이선스

Ultralytics YOLOv11 활용하기

버전	출시 연도	라이선스	특징	
YOLOv3	2018년	MIT 라이선스 GNU AGPL 3.0	- Darknet-53 백본 사용 - 잔차 연결과 업샘플링 도입 - 3개의 다른 스케일에서 객체 검출	https://pjreddie.com/darknet/yolo/ https://github.com/ultralytics/yolov3
YOLOv4	2020년	YOLO	- CSPDarknet53 백본 사용 - Bag of Freebies와 Bag of Specials 개선 - 검출 성능 향상	https://arxiv.org/abs/2004.10934 https://github.com/AlexeyAB/darknet
YOLOv5	2020년	GNU AGPL 3.0	- 모델 크기 축소 - 학습과 배포의 용이성 개선 - 속도와 정확도 향상	Ultralytics
YOLOv6	2022년	GNU GPL 3.0	- Meituan에서 개발 - 산업용 애플리케이션에 초점 - 속도와 정확도 측면에서 성능 개선	https://arxiv.org/abs/2301.05586 https://github.com/meituan/YOLOv6
YOLOv7	2022년	GNU GPL 3.0	- 가장 빠르고 정확한 실시간 객체 검출기 주장 - E-ELAN 구조 도입 - 아키텍처 개선으로 성능 향상	https://arxiv.org/pdf/2207.02696 https://github.com/WongKinYiu/yolov7
YOLOv8	2023년	GNU AGPL 3.0	- 검출, 분류, 분할 작업 지원 - 아키텍처 개선으로 성능 및 사용성 향상	Ultralytics
YOLOv9	2024년	GNU GPL 3.0	- 주로 사전 훈련된 모델로 성능을 강화하는 방식. - 트랜스포머 기반 백본 도입 가능성.	https://arxiv.org/pdf/2402.13616 https://github.com/WongKinYiu/yolov9
YOLOv10	2024년	GNU AGPL 3.0	- 하이브리드 모델로 CNN과 트랜스포머의 장점을 결합. - 보다 더 복잡한 객체 및 세밀한 객체 탐지 성능 개선.	https://arxiv.org/abs/2405.14458 https://github.com/THU-MIG/yolov10
YOLOv11	2024년	GNU AGPL 3.0	- 멀티 모달 학습 도입. - 다양한 센서 및 데이터 소스를 결합한 모델. - 다중 도메인 객체 탐지를 위한 더욱 강력한 성능 제공.	Ultralytics

GNU GPL 3.0은 소프트웨어를 외부에 공개할 경우 소스코드를 공개할 의무가 있지만, 내부에서만 사용할 경우 또는 서비스 형태로 제공하는 경우는 소스코드를 공개할 의무가 없습니다.

- 그이 비해서 GNU AGPL 3.0라이선스는 소프트웨어를 배포하지 않고 네트워크 서비스로 제공하더라도, 서비스 형태로 외부 사용자에게 제공되면 소스코드를 공개해야 합니다.
- https://opensource.com/article/17/1/providing-corresponding-source-agplv3-license

LangChain(langchain.com)

실시간 객체 탐지를 위한 인공지능 플랫폼 활용

인공지능 및 자연어 처리 분야에서 언어 모델을 연결하고 결합하여 복잡한 작업을 수행할 수 있도록 하는 프레임워크

- 다양한 언어 모델, 데이터 소스, 툴 및 알고리즘을 결합하여 더 강력하고 유연한 시스템을 구축할 수 있도록 설계되었음.
- 언어 모델의 강력한 기능을 최대한 활용할 수 있도록 돕는 도구로, 다양한 NLP 응용 프로그램을 더 효과적으로 개발할 수 있게 해줌

LangChain의 주요 특징

모듈화된 설계

- 렝체인은 모듈화된 구조로 되어 있어 사용자가 필요에 따라 다양한 언어 모델과 컴포넌트를 쉽게 연결하고 조합할 수 있음.

다양한 언어 모델 지원

- GPT-3, BERT, T5 등 다양한 최신 언어 모델을 지원하며, 이들을 조합하여 더 복잡한 작업을 수행할 수 있음.

데이터 소스 통합

- 텍스트 데이터뿐만 아니라 구조화된 데이터, 데이터베이스, API 등을 통해 다양한 데이터 소스를 통합하여 사용 가능.

확장성과 유연성

- 사용자가 필요에 따라 새로운 모델이나 알고리즘을 추가하거나 기존의 것들을 수정할 수 있는 높은 확장성과 유연성을 제공.

LangChain의 응용 사례

질의응답 시스템

- 여러 언어 모델을 결합하여 더 정확하고 풍부한 답변을 제공하는 질의응답 시스템 구축.

텍스트 요약

- 다양한 문서로부터 중요한 정보를 추출하고 요약하는 작업.

대화형 에이전트

- 사용자와의 대화를 통해 다양한 정보를 제공하고, 복잡한 작업을 수행하는 대화형 인공지능 에이전트 개발.

텍스트 생성 및 변환

- 주어진 텍스트를 다양한 형식으로 변환하거나 새로운 텍스트를 생성하는 작업.

LangChain 시작하기(OpenAI)

LangChain 활용하기

pip install langchain-openai

OpenAI 채팅 모델 생성

```
import os
os.environ["OPENAI_API_KEY"] = 'sk-proj-w9TVnvFtfdYDSFTJmgFrT3BlbkFJqBFNAcGGDmrWHEr4UXDh'

from langchain_openai import ChatOpenAI
model = ChatOpenAI(model="gpt-4")
```

요청하고 응답 받기

```
from langchain_core.messages import HumanMessage, SystemMessage
messages = [
    SystemMessage(content="다음 한국어를 일본어로 번역해줘"),
    HumanMessage(content="나는 어제 아무것도 먹지 않았다."),
]

result = model.invoke(messages)
print(result)
```

```
content='私は昨日何も食べませんでした。' response_metadata={'token_usage':
{'completion_tokens': 17, 'prompt_tokens': 49, 'total_tokens': 66}, 'model_name':
'gpt-4-0613', 'system_fingerprint': None, 'finish_reason': 'stop', 'logprobs': None}
id='run-6305b04e-0018-447b-8b8b-979350964f2c-0' usage_metadata={'input_tokens': 49,
'output_tokens': 17, 'total_tokens': 66}
```

응답 메시지 출력하기

```
print(result.content)
```

私は昨日何も食べませんでした。

Content 속성을 이용하거나
랭체인의 StrOutputParser를
사용할 수 있습니다.

```
from langchain_core.output_parsers import StrOutputParser
```

```
parser = StrOutputParser()
print(parser.invoke(result))
```

私は昨日何も食べませんでした。

```
messages = [
    SystemMessage(content="다음 한국어를 일본어로 번역해줘(한자 옆에 히라가나도 표기해줘)"),
    HumanMessage(content="나무 위에 원숭이가 한 마리 있습니다."),
]
chain = model | parser
print(chain.invoke(messages))
```

木(き)の上(うえ)に猿(さる)が一匹(いっぴき)います

LangChain 서비스 만들기

LangChain 활용하기

pip install "langserve[all]"

브라우저에서 확인: http://localhost:8000/chain/playground/

```
%%writefile 'serve.py'
#!/usr/bin/env python
import os
os.environ["OPENAI_API_KEY"]='sk-proj-w9TVnvFtfdYDSFTJmgFrT3BlbkFJqBFNAcGGDmrWHEr4UXDh'

from typing import List
from fastapi import FastAPI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_openai import ChatOpenAI
from langserve import add_routes

# 1. Create prompt template
system_template = "Translate the following into {language}:"
prompt_template = ChatPromptTemplate.from_messages([
    ('system', system_template),
    ('user', '{text}')
])

# 2. Create model
# model = ChatOpenAI()
model = ChatOpenAI(model="gpt-4")

# 3. Create parser
parser = StrOutputParser()

# 4. Create chain
chain = prompt_template | model | parser

# 5. App definition
app = FastAPI(
  title="LangChain Server",
  version="1.0",
  description="A simple API server using LangChain's Runnable interfaces",
)

# 6. Adding chain route
add_routes(
    app,
    chain,
    path="/chain",
)

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="localhost", port=8000)
```

LangChain과 FastAPI를 이용해서 쉽게 웹기반 AI 서비스를 만들 수 있습니다.

http://localhost:8000/chain/playground/

LangChain 활용하기