

오픈 Datasets과 레이블링 형식 및 파일변환 코드, 레이블링 툴 발표

2022. 5. 18

충북대학교 산업인공지능학과

[21-5조] 방창현, 정원용

목차

1. 오픈 Datasets

2. 레이블링 형식

3. 레이블링 툴

4. 파일변환 코드

1. 오픈 Datasets

- Kaggle

- <https://www.kaggle.com/datasets/andrewmvd/hard-hat-detection?select=annotations>

Safety Helmet Detection

Improve work safety by detecting the presence of people and safety helmets.



Data Code (12) Discussion (6) Metadata

About Dataset



Abstract

Improve workplace safety by detecting people and hard hats on 5k images with bbox annotations.



About this dataset

This dataset, contains 5000 images with bounding box annotations in the PASCAL VOC format for these 3 classes:

- Helmet;
- Person;
- Head.

Data Explorer

Version 1 (1.32 GB)

- ▶  annotations
- ▶  images

1. 오픈 Datasets

- Kaggle

- <https://www.kaggle.com/datasets/vodan37/yolo-helmethead>

YOLO helmet/head

Database with helmet/head labels. In YOLO format. 1- helmet 0 - head.

Data Code (1) Discussion (0) Metadata

About Dataset

Program for TRT. https://github.com/vodan37/yolov5_tensorrt

UPD: add trained weights YOLOv5m and some training data.














UPD2: add test program for TensorRT. Weights included (int8 and fp32). Also add test results for TensorRT.

Source list:

- <https://www.kaggle.com/andrewmvd/hard-hat-detection>
- <https://www.kaggle.com/abhishek4273/helmet-dataset>
- <https://www.kaggle.com/junwide/safetyhelmetwearing>
- <https://public.roboflow.com/object-detection/hard-hat-workers>

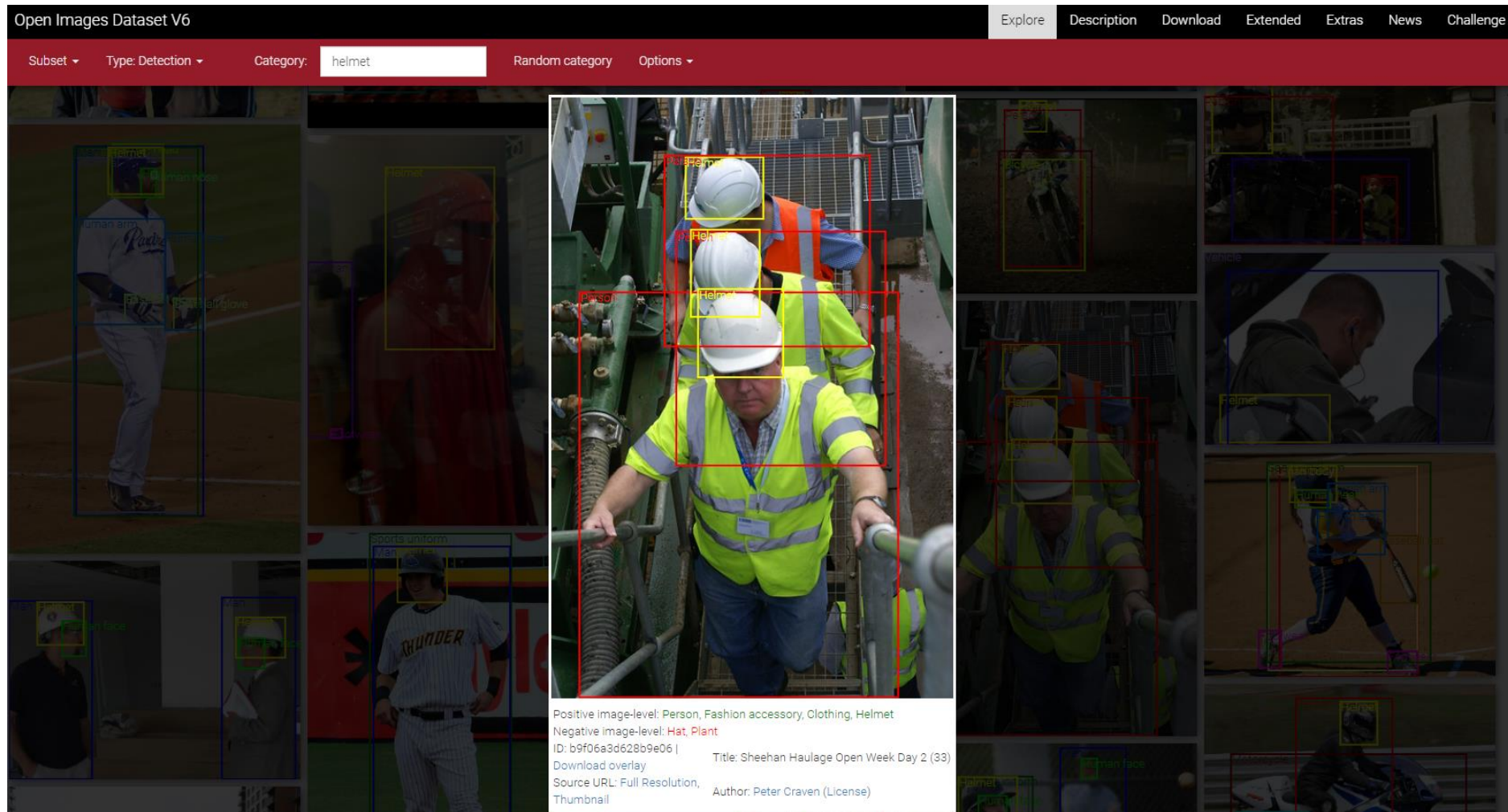
Data Explorer

Version 8 (4.56 GB)

- ▼  helm
 - ▼  helm
 - ▼  images
 - ▶  test
 - ▶  train
 - ▶  valid
 - ▼  labels
 - ▶  test
 - ▶  train
 - ▶  valid
 -  train.cache
 -  helm.yaml
 - ▶  test_results
 - ▶  train_results
 - ▶  yolov5_tensorrt

1. 오픈 Datasets

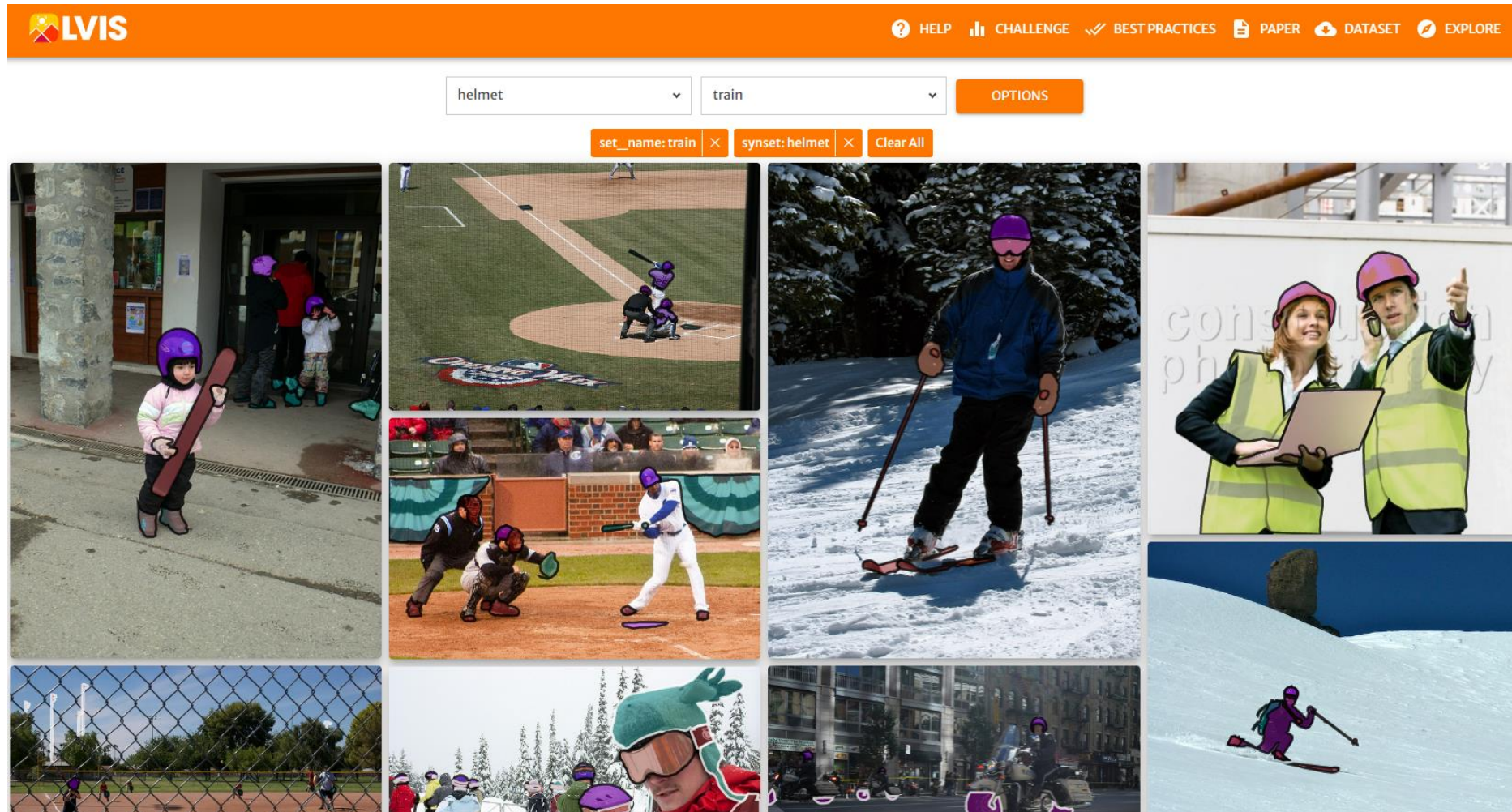
- Open Images Dataset V6 +
- <https://storage.googleapis.com/openimages/web/index.html>



1. 오픈 Datasets

- LVIS


- <https://www.lvisdataset.org/>



2. 레이블링 형식

- Annotation(주석) / label(레이블)

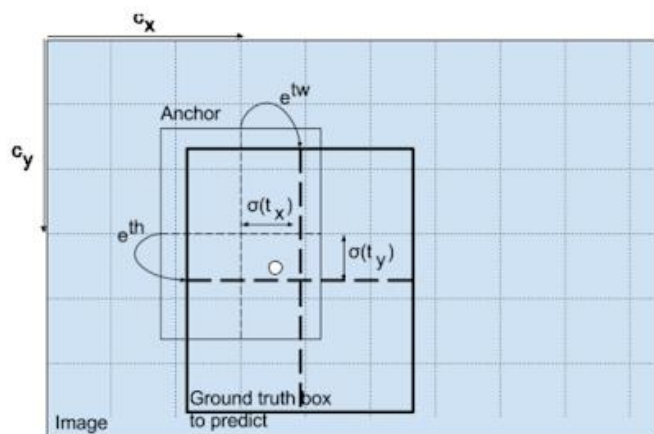
- 영상안의 객체에 대한 위치, 클래스, 속성 등의 정보를 갖고 있는 텍스트 설명 파일

Image	Annotation (PASCAL VOC (*.xml))	Label (YOLOv5 (*.txt))
	<pre><annotation> <folder>images</folder> <filename>hard_hat_workers0.png</filename> <size> <width>416</width> <height>416</height> <depth>3</depth> </size> <segmented>0</segmented> <object> <name>helmet</name> <pose>Unspecified</pose> <truncated>0</truncated> <occluded>0</occluded> <difficult>0</difficult> <bndbox> <xmin>357</xmin> <ymin>116</ymin> <xmax>404</xmax> <ymax>175</ymax> </bndbox> </object> </annotation></pre>	<div><div>helmet head</div><div>classes.txt</div></div> <pre>0 0.044471 0.395433 0.069712 0.084135 1 0.173077 0.378606 0.043269 0.060096 0 0.253606 0.364183 0.031250 0.040865 0 0.311298 0.395433 0.045673 0.069712 1 0.441106 0.412260 0.045673 0.069712 1 0.500000 0.382212 0.033654 0.052885 1 0.560096 0.399038 0.038462 0.057692 0 0.641827 0.385817 0.052885 0.079327 0 0.920673 0.350962 0.105769 0.129808 1 0.403846 0.393029 0.033654 0.050481 0 0.746394 0.391827 0.045673 0.081731</pre>

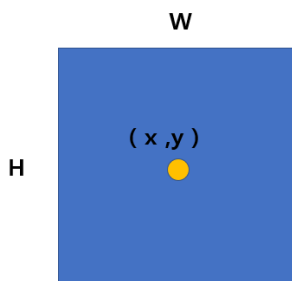
2. 레이블링 형식

- YOLOv5의 주석 파일 형식 (*.txt)

- 5개의 값이 공백으로 분리
- (1열) Class ID : 0~1 (helmet, head)
- (2~5열) Bounding box 좌표 : 0.0~1.0 (center x, center y, weight, height)



경계박스(anchor box={cx, cy, width, height}).
이 경계박스를 클래스별로 입력된 학습데이터
에 맞게 조정해 학습모델을 만든다.



hard_hat_workers0.png

0	helmet
1	head

classes.txt

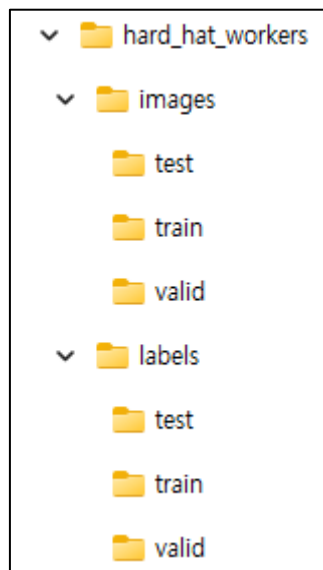
Class ID	center x	center y	weight	height
0	0.044471	0.395433	0.069712	0.084135
1	0.173077	0.378606	0.043269	0.060096
0	0.253606	0.364183	0.031250	0.040865
0	0.311298	0.395433	0.045673	0.069712
1	0.441106	0.412260	0.045673	0.069712
1	0.500000	0.382212	0.033654	0.052885
1	0.560096	0.399038	0.038462	0.057692
0	0.641827	0.385817	0.052885	0.079327
0	0.920673	0.350962	0.105769	0.129808
1	0.403846	0.393029	0.033654	0.050481
0	0.746394	0.391827	0.045673	0.081731

hard_hat_workers0.txt

2. 레이블링 형식

- YOLOv5의 Dataset 구조

- 영상별로 주석파일(*.txt)이 매칭



- 학습용 환경설정파일

- helm.yaml

```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license

# Train/val/test sets as 1) dir: path/to/imgs,
train: ../images/train/
val: ../images/valid/

# number of classes
nc: 2

# class names
names: ['head', 'helmet']
```

3. 레이블링 툴

3.1.데이터 레이블링

• 데이터 레이블링 이란?

- 데이터 레이블링 작업은 머신러닝이나 딥러닝 모델링 작업 전에 학습 데이터에 특정 값을 부여 해주는 것이다. 이는, AI 산업의 기본이기도 하지만, 기업의 AI 사업의 획기적 발전을 이루기 위한 핵심 요소로 인식된다. 하지만, 사람이 일일이 하다 보니 비용이 많이 들고 시간이 오래 걸리는 작업이다.
 - . 첫째, 머신러닝을 위해서는 많은 양의 데이터가 필요한데, 이미지를 분류할 때 최소 수천, 수만 장의 이미지가 요구된다.
 - . 둘째, 시간을 포함한 많은 비용이 든다.
 - . 레이블링 작업자가 많아질수록 일관되고 정확한 레이블링 작업이 어려워진다.

<실제 사례>

영국 경제지 파이낸셜타임스(FT)는

"자율주행차 알고리즘이 도로 표지판 등을 학습하려면 수천 시간 분량의 레이블링 된 운전 동영상이 필요하다"며 "1시간 짜리 동영상에 레이블링 하는데 8시간이 걸린다"고 했다.

통상 AI 학습 시간의 약 80~90%를 레이블링 작업이 차지하는 것으로 알려졌다.

- Image, Text, Audio, Video, Time Series 등의 다양한 형태의 데이터에 대한 레이블링 툴이 존재한다.

3. 레이블링 툴

- Image

CVAT (Computer Vision Annotation Tool)	<ul style="list-style-type: none"> 객체 감지, 분할 및 분류와 같은 작업을 위한 이미지 및 비디오용 오픈 소스 주석 도구 응용 프로그램 설치 필요없이 웹 형태로 온라인에서 사용 가능 사전에 훈련된 모델을 사용하여 자동 레이블링 가능(Coco 모델) 데이터 레이블링에 다소 시간 소요 	
labellmg	<ul style="list-style-type: none"> 널리 사용되는 오픈 소스 그래픽 주석 도구 객체 지역화 또는 감지 작업에만 적합하며 객체 주위에 최대한 단순한 Bbox 형태의 라벨 생성 가능 키보드 단축키를 통한 라벨링 편의성 제공하고 PASCAL VOC, YOLO 및 CreateML 3종의 annotation 파일 저장 및 로딩 기능을 제공 	
Labelbox	<ul style="list-style-type: none"> CV(Computer Vision) application 구축을 위한 데이터 annotation에서 가장 빠른 툴임 응용 프로그램 설치 필요없이 웹 형태로 온라인에서 사용 가능 Bbox 외에 폴리곤 형태의 라벨링 가능 이미지 저작권 체크 및 과금이 되는 단점이 존재 데이터 보안 issue가 있는 경우에는 사용하기 어려운 단점 	
LabelMe	<ul style="list-style-type: none"> MIT 컴퓨터 과학 및 인공지능 연구소에서 만듦 Bbox 외에 Polygon, Line, Point 등 다양한 형태의 도형 labeling 가능 설치가 간단하고 손쉬운 라벨링 가능 Png, bmp 등의 다양한 이미지 포맷 외에도 Video annotation 가능 	

3. 레이블링 툴

- Text

YEDDA: A
Lightweight
Collaborative
Text Span
Annotation
Tool

- Text span annotation에 대한 체계적인 솔루션을 제공
- 직접 텍스트에 주석을 달 때 매우 효율적인 short cut key 기능을 지원하는데, 이를 통해사용자는 텍스트 범위만 선택하고 short cut key를 누르면 범위가 자동으로 Annotation 처리
기존의 annotation tool의 낮은 효율성 향상
- Yedda는 또한 최신 주석이 달린 텍스트를 학습하여 지능적인 권장 사항을 제공

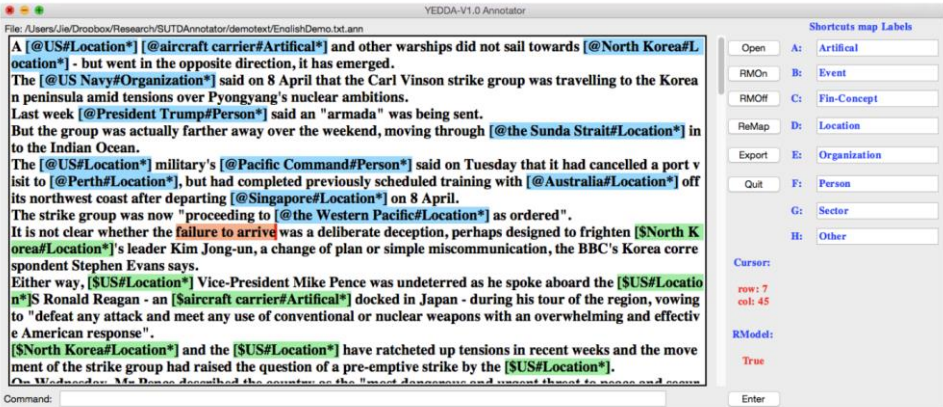
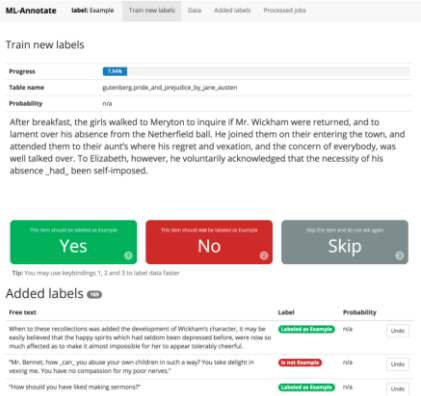


Figure 2: Annotator Interface.

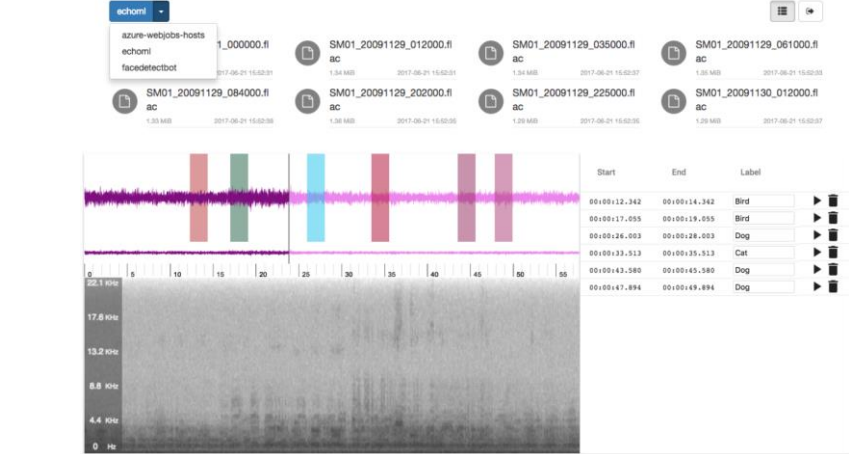
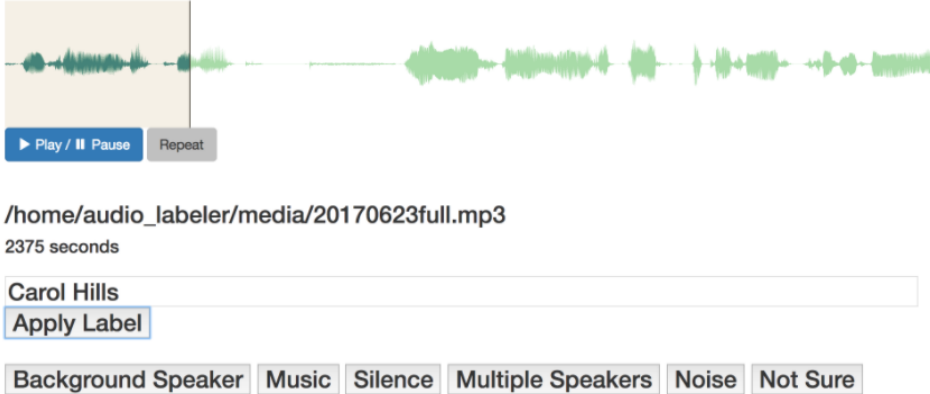
ML-
Annotate

- 기계 학습 목적으로 텍스트 데이터에 레이블링을 서포트
- ML-Annotate는 이진, 다중 레이블 및 다중 클래스 레이블링을 하는데 도움



3. 레이블링 툴

- Audio

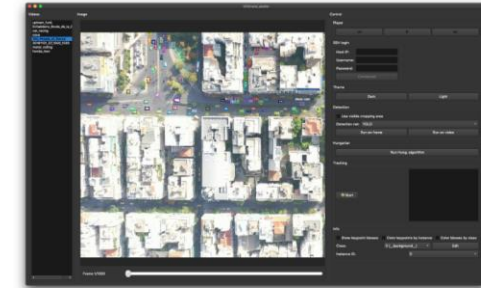
EchoML	<ul style="list-style-type: none">Audio파일을 시각화하고 레이블링 하게 해주는 도구	 <p>The screenshot shows the EchoML web interface. At the top, there's a list of audio files with columns for file name, size, and date. Below this is a spectrogram visualization of an audio file. The spectrogram has a time axis at the bottom (0 to 55 seconds) and a frequency axis on the left (0 Hz to 20.5 kHz). Several colored vertical bars are overlaid on the spectrogram, indicating labeled segments. To the right of the spectrogram is a table with columns for Start, End, and Label, showing specific time intervals and their corresponding labels (e.g., Bird, Dog, Cat).</p>
audio-labler	<ul style="list-style-type: none">Docker와 Flask를 사용하여 오디오 파일을 랜덤으로 레이블링 할 수 있는 in-browser 도구	 <p>The screenshot shows the audio-labler web interface. It features a large green waveform of an audio file. Below the waveform are controls for 'Play / Pause' and 'Repeat'. Underneath, the file path is displayed: '/home/audio_labeler/media/20170623full.mp3' with a duration of '2375 seconds'. There is a text input field containing 'Carol Hills' and an 'Apply Label' button. At the bottom, there are several buttons for selecting labels: 'Background Speaker', 'Music', 'Silence', 'Multiple Speakers', 'Noise', and 'Not Sure'.</p>

3. 레이블링 툴

- Video

UltimateLab eling

- 통합 된 SOTA 검출기 및 추적기를 갖춘 Python의 다목적 비디오 레이블링 GUI
- PyQt5를 사용하여 개발 됨



VATIC - Video Annotation Tool from Irvine, California

- Amazon Mechanical Turk에서 작동하는 컴퓨터 비전 연구를 위한 온라인 비디오 Annotation 도구
- 저렴한 대규모 비디오 데이터 세트를 구축하는 데 용이하다.

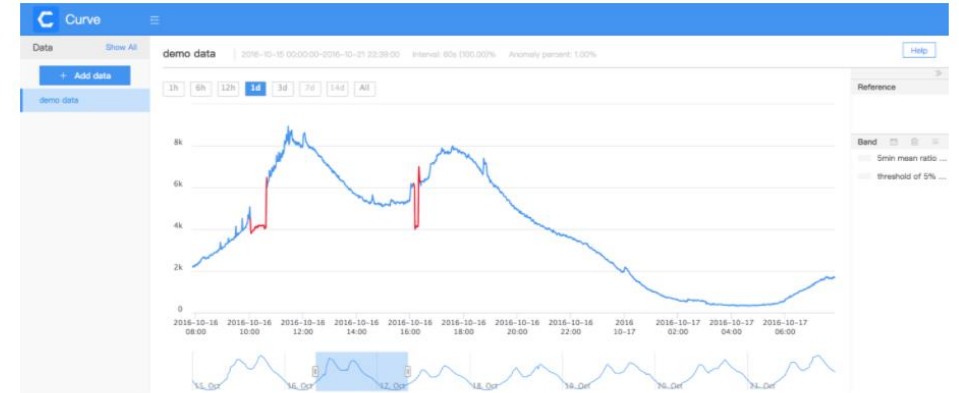


3. 레이블링 툴

- Time Series

Curve

- 시계열 데이터에서 변칙이 보이는 부분을 레이블링 하는데 도움을 주는 도구



taganomaly

- 다중 시계열 데이터에서 보이는 변칙을 탐지하고 레이블링 해주는 도구



3. 레이블링 툴

3.2.레이블링 툴(labelImg) 사용법



- ① 메뉴 :
레이블링 툴(labelImg)의 메뉴
- ② 툴바 :
Open – 이미지 지정
Open Dir – 이미지 폴더 지정
Save – PascalVOC, Yolo, CreateML Annotation 저장
Create RectBox – BoundingBox 생성
- ③ 이미지 패널 :
선택된 이미지 및 Bbox 출력
- ④ Box Labels
Use default label – 라벨 default 값 설정
라벨 목록 표시, 라벨 수정
- ⑤ 라벨 목록
현재 선택된 이미지 라벨 목록 표시
목록 선택 시 해당 라벨 표시
- ⑥ 파일 목록
이미지 폴더 내 파일 목록 표시
파일 선택 시 해당 이미지 및 Bbox를 이미지 패널에 출력

3. 레이블링 툴

3.3. 테스트 데이터셋

- 테스트 데이터 셋
 - 데이터 종류 : 안전모, 사람머리, 사람 구분
 - 데이터 개수 : 이미지(.PNG), xml 각각 5,000건

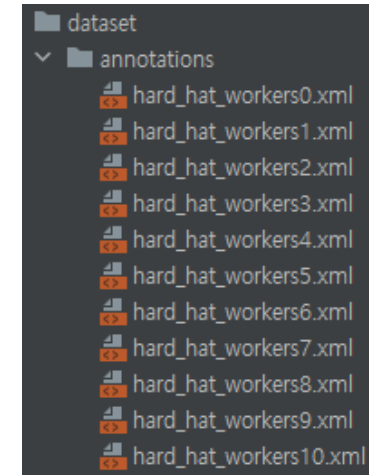
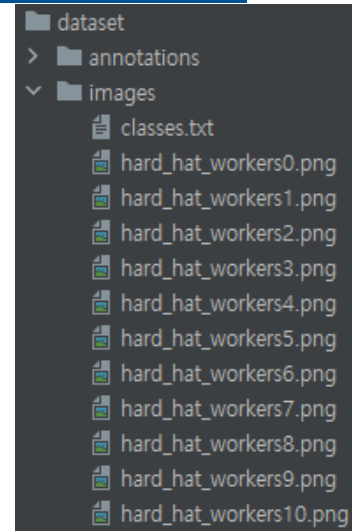
<예시 이미지1>



Case #1

Image name :
hard_hat_workers4496.png
Annotation file :
hard_hat_workers4496.xml

Classification :
helmet : 2
head : 0
person : 0



<예시 이미지2>



Case #2

Image name :
hard_hat_workers43.png
Annotation file :
hard_hat_workers43.xml

Classification :
helmet : 5
head : 2
person : 0

3. 레이블링 툴

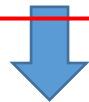
3.4.format 정의

#PASCAL VOC (.xml)

```
<annotation>
  <folder>images</folder>
  <filename>hard_hat_workers4496.png</filename>
  <size>
    <width>416</width>
    <height>415</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>helmet</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
      <xmin>118</xmin>
      <ymin>82</ymin>
      <xmax>197</xmax>
      <ymax>177</ymax>
    </bndbox>
  </object>
  <object>
    <name>helmet</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
      <xmin>219</xmin>
      <ymin>102</ymin>
      <xmax>304</xmax>
      <ymax>191</ymax>
    </bndbox>
  </object>
</annotation>
```

<folder> : 이미지 폴더 경로
<filename> : 이미지 파일 명
<size>
 <width> : 이미지 width 값
 <height> : 이미지 height 값
 <depth> : 이미지 channels 값
<segmented> segmentation 된 이미지 여부
<object>
 <name> : object class 명
 <pose> : object 방향성
 <truncated>: Bbox와 object 전체 영역 일치 여부
 <occluded> : 객체 겹침 부분 여부
 <difficult> : 인식 어려운지 여부
 <bndbox> : BoundingBox

xmin, ymin, xmax, ymax



VOC(Visual Object Classes)

#CreateML (.json)

```
[
  {
    "image": "hard_hat_workers4496.png",
    "annotations": [
      {
        "label": "helmet",
        "coordinates": {
          "x": 157.5,
          "y": 129.5,
          "width": 79.0,
          "height": 95.0
        }
      },
      {
        "label": "helmet",
        "coordinates": {
          "x": 261.5,
          "y": 146.5,
          "width": 85.0,
          "height": 89.0
        }
      }
    ]
  }
]
```

"image" : 이미지 파일 명
"annotations" :
 "label" : classification 명
 "coordinates" : 좌표
 "x" : x 좌표
 "y" : y 좌표
 "width" : 이미지 width 값
 "height" : 이미지 height 값

#Yolo (.txt)

0	0.378606	0.312048	0.189904	0.228916
0	0.628606	0.353012	0.204327	0.214458

①

②

③

④

⑤

① class : classification 배열 Index
② x_center : 라벨 중심 x 좌표
③ y_center : 라벨 중심 y 좌표
④ width : 라벨 width 값
⑤ height : 라벨 height 값

4. 파일변환 코드

4.1. 변환코드 xml(Pascal VOC) -> txt(yolo)

```
import os
import xml.etree.ElementTree as ET
import cv2

labels_path = './dataset/annotations'
images_path = './dataset/images'
classes = ['helmet', 'head', 'person']

def _convertBbox(size, box):
    dw = 1./size[0] # 1/w
    dh = 1./size[1] # 1/h
    x = (box[0] + box[1])/2.0 # The center point of the object in the figure x coordinate
    y = (box[2] + box[3])/2.0 # The center point of the object in the figure y coordinate
    w = box[1] - box[0] # The actual pixel width of the object
    h = box[3] - box[2] # The actual pixel height of the object
    x = round(x*dw, 6) # The center point of the object x Coordinate ratio of ( amount to
    w = round(w*dw, 6) # The width ratio of the object width ( amount to w/ Original pi
    y = round(y*dh, 6) # The center point of the object y Coordinate ratio of ( amount to
    h = round(h*dh, 6) # The width ratio of the object width ( amount to h/ Original pi
    return (x, y, w, h)

def _convert2txt():
    # 1.annotation 파일 디렉토리에서 xml 파일의 변환 대상 정보를 획득
    # 2.라벨링 대상 이미지 파일의 width, height 값 획득
    # 3.Object의 class 이름을 비교하여 class의 index 값 획득
    # 4.Bbox(bndbox) 값 획득
    # 5.annotation 파일과 동일한 이름의 txt 파일을 생성
    # 6.txt 데이터 형태(class, x_center, y_center, width, height) 값 변환 후 txt 파일 저장
```

```
for imgs in os.listdir(images_path):
    file_name = imgs.split('.')[0]
    file_ext = imgs.split('.')[1]

    if file_ext == 'png':
        #pascal VOC 파일
        in_file = open(labels_path + '/%s.xml' % (file_name), encoding='utf-8')
        in_file = open(labels_path + '/' + imgs.strip('.png') + '.xml', encoding='utf-8')
        dom = ET.parse(in_file)
        root = dom.getroot()
        image = cv2.imread(images_path + '/' + imgs)
        #이미지 width, height 획득
        w = int(image.shape[1])
        h = int(image.shape[0])
        #[asis]name, bndbox[xmin,ymin,xmax,ymax]
        #[tobe]class, x_center, y_center, width, height
        for obj in root.findall('object'):
            cls = obj.find('name').text
            cls_id = classes.index(cls)
            bbox = obj.find('bndbox')
            b = (float(bbox.find('xmin').text),
                 float(bbox.find('xmax').text),
                 float(bbox.find('ymin').text),
                 float(bbox.find('ymax').text))
            bb = _convertBbox((w,h),b)

            out_file = open(images_path + '/' + imgs.strip('.png') + '.txt', 'w', encoding='utf-8')
            out_file.write(str(cls_id) + " " + " ".join([str(a) for a in bb]) + '\n')
            out_file.close()
```

[asis]name, bndbox[xmin,ymin,xmax,ymax]
[tobe]class, x_center, y_center, width, height

- # 1.annotation 파일 디렉토리에서 xml 파일의 변환 대상 정보를 획득
- # 2.라벨링 대상 이미지 파일의 width, height 값 획득
- # 3.Object의 class 이름을 비교하여 class의 index 값 획득
- # 4.Bbox(bndbox) 값 획득
- # 5.annotation 파일과 동일한 이름의 txt 파일을 생성
- # 6.txt 데이터 형태(class, x_center, y_center, width, height) 값 변환 후 txt 파일 저장

4. 파일변환 코드

4.2.변환코드 txt(yolo) -> xml(Parscal VOC)

```
import os
import xml.etree.ElementTree as ET
from PIL import Image
import numpy as np

#constant
labels_path = './dataset/annotations'
images_path = './dataset/images'
classes = ['helmet', 'head', 'person']

def generate_xml(imgname, sw, sh, sd, filepath, label_dicts):

    # xml 파일의 root 노드 설정
    root = ET.Element('annotation')

    # 자식 노드 설정
    ET.SubElement(root, 'folder').text = 'images'
    ET.SubElement(root, 'filename').text = str(imgname)
    sizes = ET.SubElement(root, 'size')
    ET.SubElement(sizes, 'width').text = str(sw)
    ET.SubElement(sizes, 'height').text = str(sh)
    ET.SubElement(sizes, 'depth').text = str(sd)
    ET.SubElement(root, 'segmented').text = '0'

    for label_dict in label_dicts:
        objects = ET.SubElement(root, 'object')
        ET.SubElement(objects, 'name').text = label_dict['name']
        ET.SubElement(objects, 'pose').text = 'Unspecified'
        ET.SubElement(objects, 'truncated').text = label_dict['truncated']
        ET.SubElement(objects, 'occluded').text = label_dict['occluded']
        ET.SubElement(objects, 'difficult').text = label_dict['difficult']
        bndbox = ET.SubElement(objects, 'bndbox')
        ET.SubElement(bndbox, 'xmin').text = str(int(label_dict['xmin']))
        ET.SubElement(bndbox, 'ymin').text = str(int(label_dict['ymin']))
        ET.SubElement(bndbox, 'xmax').text = str(int(label_dict['xmax']))
        ET.SubElement(bndbox, 'ymax').text = str(int(label_dict['ymax']))

    tree = ET.ElementTree(root)
    tree.write(filepath, encoding='utf-8')
```

```
def _convert2xml():
    # 1.annotation 파일 디렉토리에서 txt 파일의 변환 대상 정보를 획득
    # 2.라벨링 대상 이미지 파일의 width, height, depth 값 획득
    # 3.Parscal VOC xml bndbox[xmin,ymin,xmax,ymax] 계산
    # 4.tree 형태의 element 구성 요소 값 설정 후 xml 파일 저장

    # txt(yolo) -> xml(Parscal VOC)
    for label in os.listdir(labels_path):
        with open(labels_path + '/' + label, 'r') as f:
            contents = f.readlines()
            label_dicts = []

            # [axis]class, x_center, y_center, width, height
            # [tobe]name, bndbox[xmin,ymin,xmax,ymax]

            for content in contents:
                image = np.array(Image.open(images_path + '/' + label.strip('.txt') + '.png'))

                sh, sw, sd = image.shape[0], image.shape[1], image.shape[2]
                content = content.strip('\n').split()

                x = float(content[1]) * sw
                y = float(content[2]) * sh
                w = float(content[3]) * sw
                h = float(content[4]) * sh

                # 좌표 변환 x_center y_center width height -> xmin ymin xmax ymax
                obj_dict = {
                    'name': classes[int(content[0])],
                    'truncated': '0',
                    'occluded': '0',
                    'difficult': '0',
                    'xmin': x + 1 - w / 2,
                    'ymin': y + 1 - h / 2,
                    'xmax': x + 1 + w / 2,
                    'ymax': y + 1 + h / 2
                }
                label_dicts.append(obj_dict)

    generate_xml(label, sw, sh, sd, images_path + '/' + label.strip('.txt') + '.xml', label_dicts)
```

- # 1.annotation 파일 디렉토리에서 txt 파일의 변환 대상 정보를 획득
- # 2.라벨링 대상 이미지 파일의 width, height, depth 값 획득
- # 3.Parscal VOC xml bndbox[xmin,ymin,xmax,ymax] 계산
- # 4.tree 형태의 element 구성 요소 값 설정 후 xml 파일 저장

5. 기타

대부분의 툴이 오픈소스 형이다보니 매뉴얼이 부족함.

점점 다양해지는 Dataset 레이블링 포맷에 대한 지원이 한계가 있음.

크게 xml, txt, json 타입이지만 사용하는 AI엔진에 따라 attribute가 차이가 있음.

이러한 문제점으로 인해 온라인형의 과금 서비스가 제공되고 있는 상황임.

여러가지 툴을 사용하지는 못했지만, 레이블 변환 후, 변환 전후의 결과물 간의 비교 기능이 있으면 데이터 검증에 시간을 줄일 수 있을 것으로 보임.

6. 질문&답변

감사합니다.