



AI Studio 사용자를 위한 데이터 준비 안내

ver. 0.9

최종수정: 2022-05-02

SURROMIND

Making AI, Empowering People

- AI Studio에 데이터 등록하기
- 단계별 안내: AI Studio의 이미지 데이터 등록방법 A
- 단계별 안내: AI Studio의 이미지 데이터 등록방법 B
- AI Studio 이미지 Label의 포맷
- 머신러닝을 위한 데이터 분할 안내

AI Studio에 데이터 등록하기

1. 행과 열로 구성된 표 형식으로 데이터를 정리한다
 1. 첫 행은 header로서 각 열에 대한 '영어' 이름표로 구성 (한글 등 영어 외는 오류)
 2. 각 열은 하나의 변수
 1. 변수의 값은 숫자형(Numeric)와 카테고리형(Categorical) 중 한 형식으로 표현
 1. 날짜형, time series, graph 등의 형식은 미지원
 3. 구축하고자 하는 모델의 입력과 출력 구분: 마지막 열(column)에 모델의 출력변수를, 나머지 열에 모델의 입력변수를 둬
 1. 분류(classification) 문제의 경우 출력변수는 정수 형식의 숫자 또는 label 단어
 2. 회귀(regression) 문제의 경우 출력변수는 실수 형식의 숫자
2. csv 형식으로 파일을 저장한다
 1. csv (comma separated version) 파일: 콤마(,)로 값이 구분된 형태의 파일
3. AI Studio에 New Dataset으로 준비한 데이터를 등록한다

1. 등록방법 A

1. 이미지 데이터를 준비한다
2. 문제 유형에 맞게 어노테이션을 수행한다
3. AI Studio에서 읽을 수 있는 포맷으로 어노테이션 데이터를 정리하여 zip 파일로 묶는다
4. AI Studio에 New Dataset으로 준비한 데이터를 등록한다

2. 등록방법 B

1. 이미지 데이터를 준비한다
2. 이미지 데이터를 zip 파일로 묶어 AI Studio에 등록한다. (Label Upload 생략)
3. AI Studio의 Annotator를 이용하여 annotation을 수행한다
 1. image classification (사진 별로 label 부여),
 2. object detection (탐지 대상 객체에 경계상자(bounding box) 표기)
 3. instance segmentation (탐지 대상 객체에 polygon 경계도형 표기)
4. Annotation 결과를 승인하고 저장하면, 학습 가능한 버전의 dataset이 생성된다

단계별 안내: AI Studio의 이미지 데이터 등록방법 A

1. 이미지 데이터를 준비한다

- 이미지의 포맷
 - png 또는 jpg
- 해상도
 - HD(1280 * 720 픽셀) 이내
 - 권장사항:
 - 가로, 세로 축 중, 긴 축의 길이가 512픽셀 이내이고 짧은 축의 3배 이내

2. 문제 유형에 맞게 어노테이션을 수행한다

- 문제 유형별로 필요한 어노테이션

- image classification: 사진 별로 label 부여
- object detection: 탐지 대상 객체에 경계상자(bounding box) 표기
- instance segmentation: 탐지 대상 객체에 polygon 경계도형 표기 또는 RLE 표기
- semantic segmentation:
 - 이미지 상의 모든 픽셀에 대해 해당 픽셀이 포함되는 사물/영역의 레이블 정보를 레이블별 색상으로 구분. 이미지와 동일한 해상도의 마스크 이미지로 저장(png 포맷)

- 이미지 어노테이션 수행

- 이미지 어노테이션 수행이 가능한 소프트웨어 사용 권장
- 일관된 기준에 따른 적절한 어노테이션 수행
- 인식 대상의 크기
 - Object detection, instance segmentation: 경계상자(bounding box)의 짧은 축의 크기는 최소 30픽셀

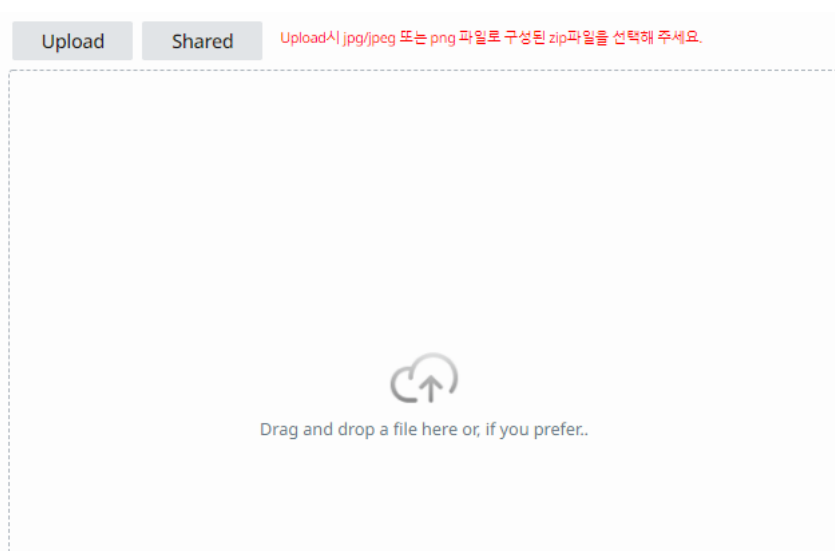
3. AI Studio에서 읽을 수 있는 포맷으로 이미지 데이터와 어노테이션 데이터를 취합한다

• 어노테이션 데이터의 구성 안내

Task	<ul style="list-style-type: none"> Image Classification Object Detection Instance Segmentation 	<ul style="list-style-type: none"> Semantic Segmentation
Upload Input Format	dataset_label_root.zip	dataset_label_root.zip
Label Directory Format	<div> <div>다음은 이름 고정</div> <ul style="list-style-type: none"> - meta.json - 폴더명: annotations </div> <div> <div>dataset_label_root</div> <div> <div>annotations</div> <div>meta.json</div> </div> </div> <div> <div>annotations</div> <div>sample01.json</div> <div>sample02.json</div> <div>meta.json</div> </div> <div> <div>이미지 1개 당 동일한 이름의 하나의 json</div> </div>	<div> <div>dataset_label_root</div> <div> <div>labels</div> <div>meta.json</div> </div> </div> <div> <div>labels</div> <div>sample01.png (or jpg)</div> <div>sample02.png (or jpg)</div> <div>meta.json</div> </div> <div> <div>다음은 이름 고정</div> <ul style="list-style-type: none"> - meta.json - 폴더명: labels </div> <div> <div>json 파일의 구조는 '이미지 데이터 포맷' 참조</div> </div> <div> <div>이미지 1개 당 동일한 이름의 하나의 label (mask) 이미지</div> </div>

4. AI Studio에 취합한 데이터를 등록하여 새로운 dataset으로 설정한다

- New Dataset 페이지로 이동
- 이미지를 취합한 zip 파일을 등록
- 어노테이션 데이터를 취합한 zip 파일을 'Label Upload' 메뉴에서 등록
- 사용자 매뉴얼의 '3. Data - Dataset 생성' 참조



- 이미지를 취합한 zip 파일을 등록.
- drag & drop을 하거나, 화살표 부분을 누른 후 업로드창에서 지정)

- 어노테이션 데이터를 취합한 zip 파일을 등록
- 어노테이션 데이터는 생략 가능. 나중에 업로드하거나 AI Studio의 Annotation을 이용하여 추가 가능.

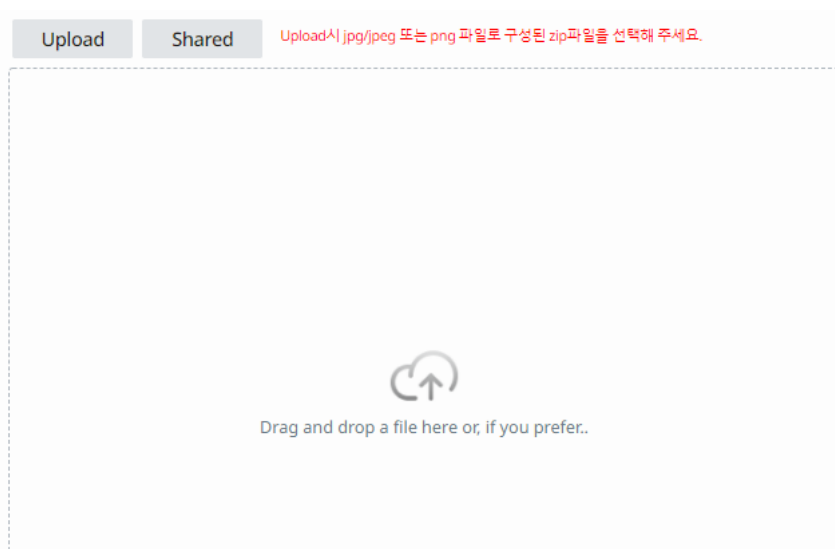
단계별 안내: AI Studio의 이미지 데이터 등록방법 B

1. 이미지 데이터를 준비한다

- 이미지의 포맷
 - png 또는 jpg
- 해상도
 - HD(1280 * 720 픽셀) 이내
 - 권장사항:
 - 가로, 세로 축 중, 긴 축의 길이가 512픽셀 이내이고 짧은 축의 3배 이내

2. 이미지 데이터를 zip 파일로 묶어 AI Studio에 등록한다. (Label Upload 생략)

- New Dataset 페이지로 이동
- 이미지를 취합한 zip 파일을 등록
- 'Label Upload' 는 공란으로 둠
- 사용자 매뉴얼의 '3. Data - Dataset 생성' 참조



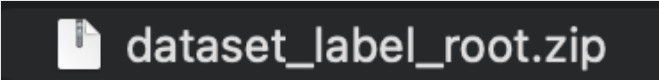
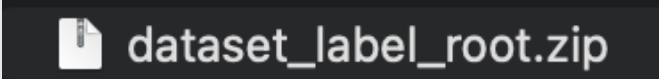
- 이미지를 취합한 zip 파일을 등록.
- drag & drop을 하거나, 화살표 부분을 누른 후 업로드창에서 지정)

- 어노테이션 데이터는 생략 가능. 나중에 업로드하거나 AI Studio의 Annotation을 이용하여 추가 가능.

3. AI Studio의 Annotator를 이용하여 annotation을 수행한다
 4. Annotation 결과를 승인하고 저장하면, 학습 가능한 버전의 dataset이 생성된다
- [AI Studio 이용 가이드](#) 의 'Annotation' 장 참조
 - 이 과정을 거쳐 Label이 없던 원본 데이터(version 1)에서 label이 부여된 데이터(version 2)가 생성된다
 - Label이 부여된 데이터를 이용하여 모델 학습을 위한 실험 단계(experiment)로 진입할 수 있다

AI Studio 이미지 Label의 포맷

Problem Type	Input	Label
Image Classification	- images 폴더안에 이미지의 형태로 존재하는 파일 (ex. images/0001.png, images/0002.png ...)	- annotations 폴더안에 입력 이미지와 이름이 같은 json 형태로 존재하는 파일 (ex. annotations/0001.json, annotations/0002.json ...). - json 파일의 'instances' 키의 값은 각 instance의 정보를 담은 dict의 list를 의미함. - 특정 instance dict의 'category_id' 키의 값은 해당 instance의 class를 나타냄.
Image Object Detection	- images 폴더안에 이미지의 형태로 존재하는 파일 (ex. images/0001.png, images/0002.png ...)	- annotations 폴더안에 입력 이미지와 이름이 같은 json 형태로 존재하는 파일 (ex. annotations/0001.json, annotations/0002.json ...). - json 파일의 'instances' 키의 값은 각 instance의 정보를 담은 dict의 list를 의미함. - 특정 instance dict의 'category_id' 키의 값은 해당 instance의 class를 나타냄. - 특정 instance dict의 'bbox' 키의 값은 해당 instance의 bounding box 좌표를 나타냄.
Image Semantic Segmentation	- images 폴더안에 이미지의 형태로 존재하는 파일 (ex. images/0001.png, images/0002.png ...)	- labels 폴더안에 입력 이미지와 이름이 같은 png 형태로 존재하는 mask image 파일 (ex. labels/0001.png, labels/0002.png ...).
Image Instance Segmentation	- images 폴더안에 이미지의 형태로 존재하는 파일 (ex. images/0001.png, images/0002.png ...)	- annotations 폴더안에 이미지와 이름이 같은 json 형태로 존재하는 파일 (ex. annotations/0001.json, annotations/0002.json ...). - json 파일의 'instances' 키의 값은 각 instance의 정보를 담은 dict의 list를 의미함. - 특정 instance dict의 'segmentation' 키의 값은 해당 instance의 polygon의 좌표 또는 RLE를 값으로 가지는 list를 의미함

Task	<ul style="list-style-type: none"> Image Classification Object Detection Instance Segmentation 	<ul style="list-style-type: none"> Semantic Segmentation
Upload Input Format		
Label Directory Format	<div data-bbox="351 789 718 911"> <p>다음은 이름 고정</p> <ul style="list-style-type: none"> - meta.json - 폴더명: annotations </div> <div data-bbox="751 743 1319 939"> <p>dataset_label_root</p> <ul style="list-style-type: none"> annotations meta.json </div> <div data-bbox="784 1001 1235 1243"> <pre> . ├── annotations │ ├── sample01.json │ └── sample02.json └── meta.json </pre> </div> <div data-bbox="453 1182 751 1299"> <p>이미지 1개 당 동일한 이름의 하나의 json</p> </div>	<div data-bbox="1319 654 1656 749"> <p>Zip 파일명은 사용자 임의로 설정</p> </div> <div data-bbox="2153 803 2517 918"> <p>다음은 이름 고정</p> <ul style="list-style-type: none"> - meta.json - 폴더명: labels </div> <div data-bbox="1663 743 2224 939"> <p>dataset_label_root</p> <ul style="list-style-type: none"> labels meta.json </div> <div data-bbox="1312 1100 1612 1210"> <p>json 파일의 구조는 '이미지 데이터 포맷' 참조</p> </div> <div data-bbox="1643 1001 2237 1243"> <pre> . ├── labels │ ├── sample01.png (or jpg) │ └── sample02.png (or jpg) └── meta.json </pre> </div> <div data-bbox="2066 1158 2453 1299"> <p>이미지 1개 당 동일한 이름의 하나의 label (mask) 이미지</p> </div>

- meta.json

- 구성요소

- "class_names": 데이터에 포함된 class label을 다음과 같이 [] 안에 쉼표로 구분하여 순서대로 표기 ["name1", "name2", ...]
 - "num_images": 데이터에 포함된 사진의 수를 정수로 표기

- 예시

- {"class_names": ["BenzEClass127", "ToyotaCamry1386", "VolvoXC601706", "BMW3Series68"], "num_images": 600}
 - 설명: class_names에 대해 다음과 같이 정수 index가 순서대로 부여되어 이미지별 label 파일에 반영됨

Class Name (화면표기용)	BenzEClass127	ToyotaCamry1386	VolvoXC601706	BMW3Series68
Index (내부처리용)	0	1	2	3

- 이미지별 label 파일: [이미지명].json

127_684c51007115d7.png



127_684c51007115d7.json

```
{"image": {"id": 41, "width": 512, "height": 352, "file_name": "127_684c51007115d7.png"},  
"instances": [{"category_id": 0}]}
```

설명

- Image: 이미지 정보
 - id: 임의의 정수를 넣어두면 됨(image 별 구분을 위한 고유 id를 옆두에 두고 설계하였으나 현재는 미사용)
 - width: 이미지의 가로 길이. Pixel 단위를 정수로 표기
 - height: 이미지의 세로 길이. Pixel 단위를 정수로 표기
 - file_name: label 파일에 해당하는 원본 이미지 파일명
- Instances: label 정보
 - category_id: 이미지에 부여된 class label의 정수 ID. meta.json의 class_names의 순서대로 0부터 증가하는 정수로 표현됨

- meta.json

- 구성요소

- "class_names": 데이터에 포함된 object의 label을 다음과 같이 [] 안에 쉼표로 구분하여 순서대로 표기 ["name1", "name2", ...]
 - "num_images": 데이터에 포함된 사진의 수를 정수로 표기

- 예시

- {"class_names": ["clock", "person", "car", "train", "TV"], "num_images": 126}
 - 설명: class_names에 대해 다음과 같이 정수 index가 순서대로 부여되어 이미지별 label 파일에 반영됨

Class Name (화면표기용)	clock	person	car	train	TV
Index (내부처리용)	0	1	2	3	4

- 이미지별 label 파일: [이미지명].json

000000000009.png

000000000009.json



```
{"image": {"id": 9, "width": 640, "height": 480, "file_name": "000000000009.jpg"},  
"instances": [{"category_id": 12, "bbox": [1.0816, 187.6896, 612.672, 473.5296]}, {"category_id": 12, "bbox": [311.7312, 4.3104, 631.008, 232.992]}, {"category_id": 2, "bbox": [249.6, 229.272, 565.8432, 474.3504]}, {"category_id": 12, "bbox": [0.0, 13.512, 434.4832, 388.632]}]}
```

설명

- Image: 이미지 정보
 - id: 임의의 정수를 넣어두면 됨(image 별 구분을 위한 고유 id를 염두에 두고 설계하였으나 현재는 미사용)
 - width: 이미지의 가로 길이. Pixel 단위를 정수로 표기
 - height: 이미지의 세로 길이. Pixel 단위를 정수로 표기
 - file_name: label 파일에 해당하는 원본 이미지 파일명
- Instances: instance별 label 정보. [] 안에 다음의 정보를 instance별로 표기
 - category_id: 이미지에 부여된 class label의 정수 ID. meta.json의 class_names의 순서대로 0부터 증가하는 정수로 표현됨
 - bbox: object에 대한 경계상자(bounding box)의 좌표 정보(float). 좌상단의 x, y좌표, 우하단의 x, y 좌표

- meta.json

- 구성요소

- "class_names": 데이터에 포함된 object label을 다음과 같이 [] 안에 쉼표로 구분하여 순서대로 표기 ["name1", "name2", ...]
 - "num_images": 데이터에 포함된 사진의 수를 정수로 표기

- 예시

- {"class_names": ["Marge_Simpson", "Bart_Simpson", "Homer_Simpson", "Lisa_Simpson"], "num_images": 12}
 - 설명: class_names에 대해 다음과 같이 정수 index가 순서대로 부여되어 이미지별 label 파일에 반영됨

Class Name (화면표기용)	Marge_Simpson	Bart_Simpson	Homer_Simpson	Lisa_Simpson
Index (내부처리용)	0	1	2	3

- 이미지별 label 파일: [이미지명].json

pic_0007.png



pic_0007.json

```
{"image": {"id": 6, "width": 640, "height": 480, "file_name": "pic_0007.jpg"}, "instances": [{"category_id": "Homer_Simpson", "segmentation": [[267.9, 78.9, 265.8, 80, 264.2, 84.8, 261.5, 86.9, 258.8, 91.8, 257.2, 96.6, 254.5, ..., 74.1, 273.3, 74.6, 271.1, 78.4]], "bbox": [186.3, 61.2, 565.4, 479.5]}]}
```

설명

- Image: 이미지 정보
 - Id: 임의의 정수를 넣어두면 됨(image 별 구분을 위한 고유 id를 옆두에 두고 설계하였으나 현재는 미사용)
 - width: 이미지의 가로 길이. Pixel 단위를 정수로 표기
 - height: 이미지의 세로 길이. Pixel 단위를 정수로 표기
 - file_name: label 파일에 해당하는 원본 이미지 파일명
- Instances: instance별 label 정보. [] 안에 다음의 정보를 instance별로 표기
 - category_id: 이미지에 부여된 class label의 정수 ID. meta.json의 class_names의 순서대로 0부터 증가하는 정수로 표현됨
 - Segmentation: objec의 윤곽선에 대한 polygon의 좌표 정보를 x좌표,y좌표의 순서로 표기
 - bbox: object에 대한 경계상자(bounding box)의 좌표 정보(float). 좌상단의 x, y좌표, 우하단의 x, y 좌표

pic_0007.json의 정보 가시화



- meta.json

- 구성요소

- "class_names": 데이터에 포함된 class label을 다음과 같이 [] 안에 쉼표로 구분하여 순서대로 표기 ["name1", "name2", ...]
 - "num_images": 데이터에 포함된 사진의 수를 정수로 표기

- 예시

- {"class_names": ["background", "aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike", "person", "pottedplant", "sheep", "sofa", "train", "tvmonitor"], "num_images": 2913}
 - 설명: class_names에 대해 다음과 같이 정수 index가 순서대로 부여되어 이미지별 label 파일에 반영됨

Class Name (화면표기용)	background	aeroplane	bicycle	bird	...	tvmonitor
Index (내부처리용)	0	1	2	3	...	20

- 이미지별 label 파일: [이미지명].png (명칭: mask 이미지)

2007_000129.png



2007_000129.png



- 다음과 같이 class label의 index 별로 순서대로 해당 픽셀에 RGB 색상 부여

VOC_COLORMAP = [[0, 0, 0], [128, 0, 0], [0, 128, 0], [128, 128, 0], [0, 0, 128], [128, 0, 128], [0, 128, 128], [128, 128, 128], [64, 0, 0], [192, 0, 0], [64, 128, 0], [192, 128, 0], [64, 0, 128], [192, 0, 128], [64, 128, 128], [192, 128, 128], [0, 64, 0], [128, 64, 0], [0, 192, 0], [128, 192, 0], [0, 64, 128],]

- 보통 백그라운드는 첫 번째 label, 검정색 [0,0,0] 으로 고정함

- 같은 class label을 가진 pixel은 같은 RGB 색상으로 표기. 즉, 사물의 종류별로 색상을 통일하며, 개별 사물(instance)은 구분하지 않음

머신러닝을 위한 데이터 분할 안내

데이터는 취득된 데이터를 다음과 같이 세 그룹으로 구분하여 활용하며, 일반적으로 구분 비율은 학습 80%, 검증 10%, 시험 10%이다.
 데이터 구분시 각 그룹에 포함된 차량 종류의 비율이 모두 유사해야 한다(stratified 요건이라고 함).

	학습(Training)	검증(Validation)	시험(Test)
개요	<ul style="list-style-type: none"> - 모델의 성능 지표를 올리기 위해 입력한 사진과 이에 해당하는 정답 출력값으로 구성된 데이터를 반복적으로 학습하는 과정 • 일반적으로 데이터를 2의 제곱 단위로 조금씩 묶어서 (mini batch, 단위는 4,8,16, 32, ...) 학습에 적용하며, 학습에 사용하는 계산 자원이 클수록 큰 단위를 적용 • 반복 학습을 통해 성능 개선 • 딥러닝 모델은 일반적으로 데이터가 많을수록 성능이 개선됨 	<ul style="list-style-type: none"> • 학습 도중 모델 성과 평가 및 비교 • 모델의 성과 지표는 알고리즘별로 설정된 학습 가이드 함수(loss function)과 실제 응용에서 고려할 평가 척도를 함께 살펴본다 • 모델의 학습이 더 필요한지(과소적합), 너무 학습을 많이 하였는지(과적합)를 성과 지표들을 통해 확인 • 일반적으로 과적합이 시작되는 시점 또는 성과 지표가 수렴하기 시작하는 시점의 모델을 선택 	<ul style="list-style-type: none"> • 학습이 완료된 모델의 성능 시험 • 학습에 사용하지 않은 별개의 데이터를 적용 • 검증 단계에서 확인한 모델의 성과 지표뿐만 아니라, 실제 응용 단계에서 고려할 다양한 특성을 만족하는지 확인
필요 데이터	취득 데이터의 80%	취득 데이터의 10% 차량 종류의 비율이 학습 데이터와 유사해야 함	취득 데이터의 10% 차량 종류의 비율이 학습 데이터와 유사해야 함

End of Document