객체 탐지(Object Detection) 기초 및 실습

강의 내용

- YOLO v5를 이용한 실시간 객체 탐지
  - Pre-trained model
  - Custom data training
- dLib face_recognition을 이용한 얼굴 인식
  - face_recognition API
  - facial landmark detection
  - Real-time face detection

딥러닝 비전(Vision) 기술

Image Classification

Object Detection

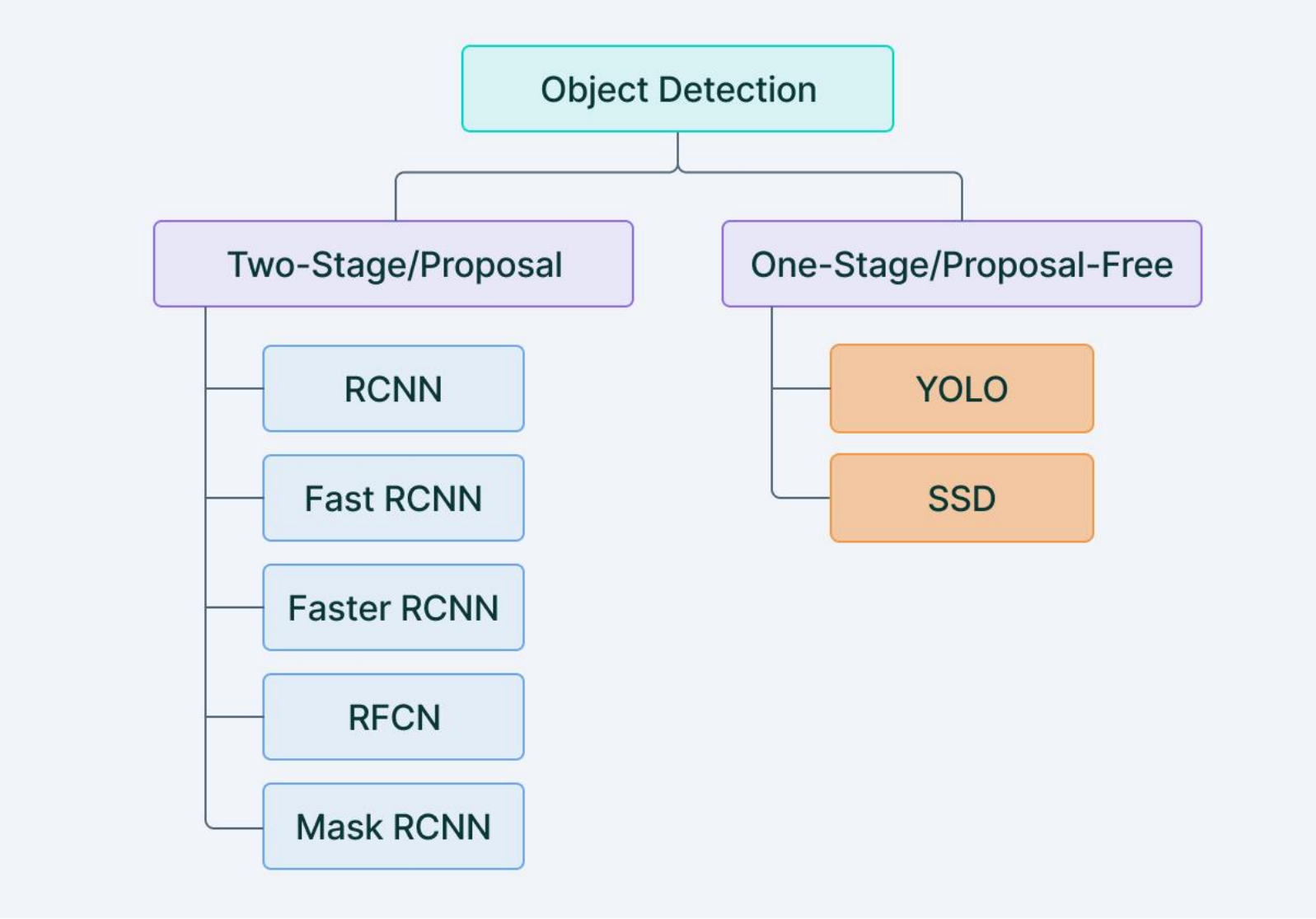
Semantic Segmentation

Pose Estimation

Mono Depth

Image Classfication vs. Object Detection

One and two stage detectors



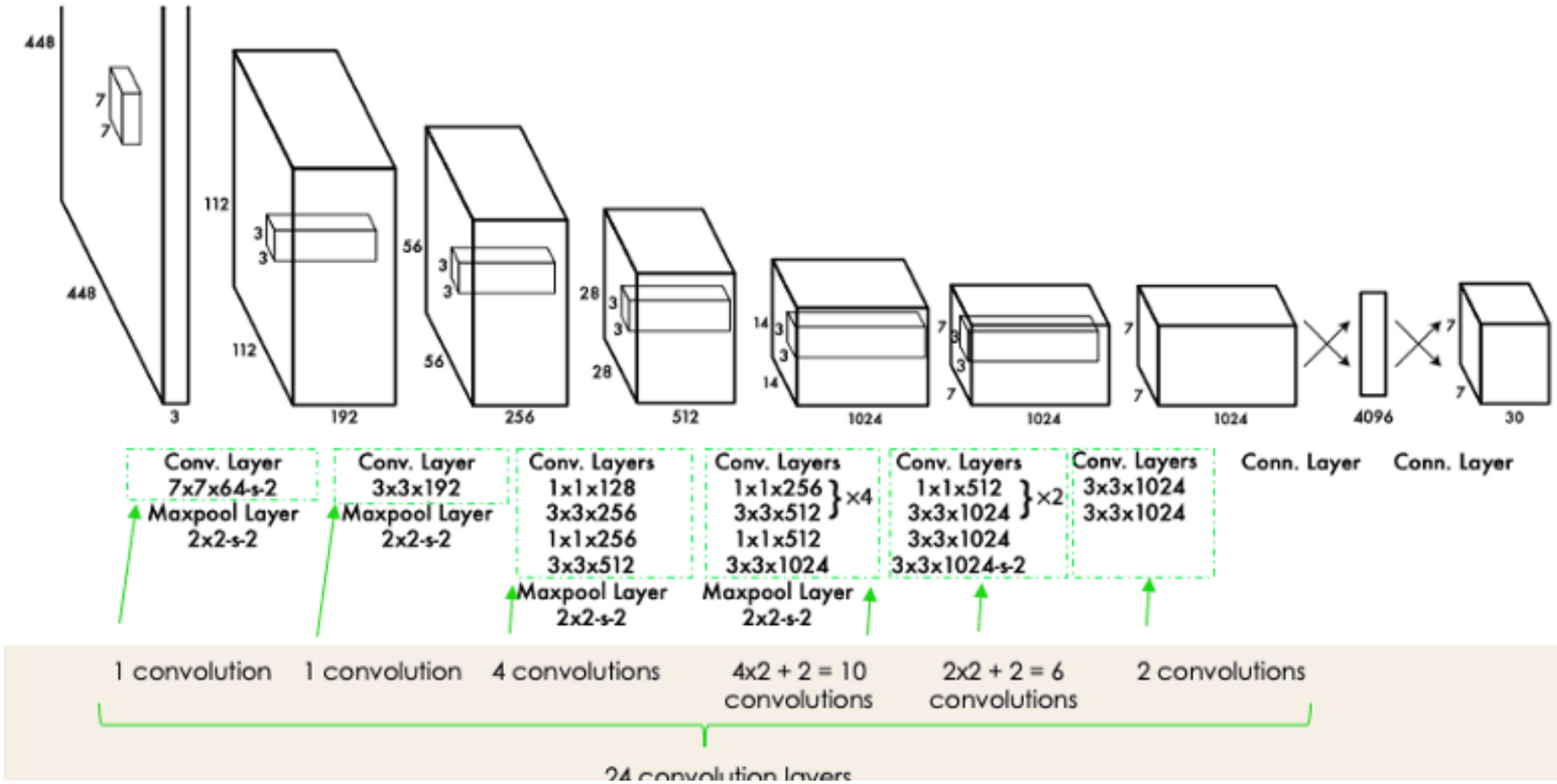
YOLOv5 by

- **이미지 전체를 단 한번만 본다 : You Only Look Once**
  - YOLO 이전의 모델 중 대표적인 R-CNN 처럼 이미지를 여러장으로 분할해 해석하지 않음.
  - YOLO는 이미지 전체를 말 그대로 한 번만 본다.

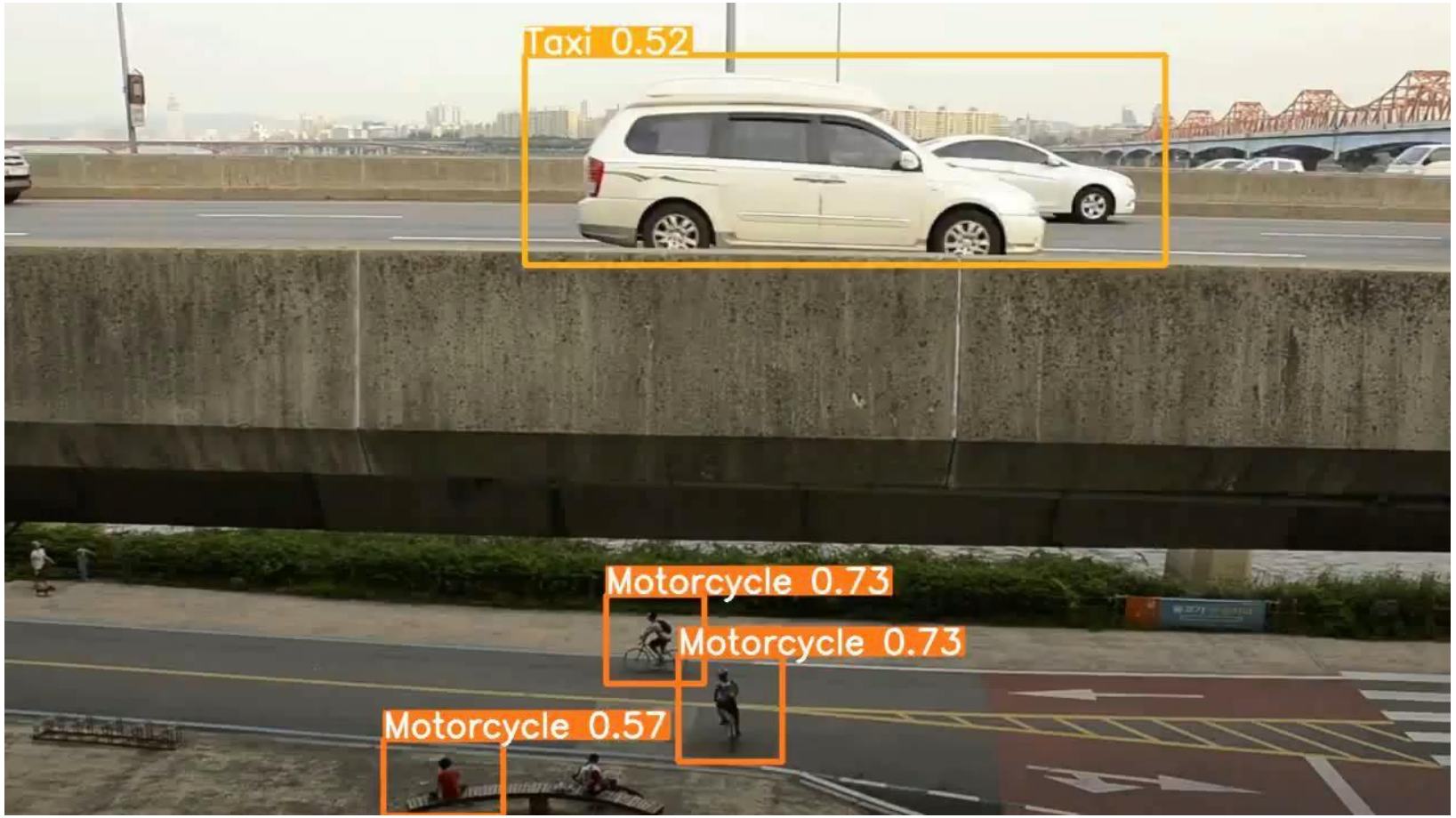
- **통합된 모델을 사용 : one-stage-detection**
  - 다른 객체 탐지 모델들은 다양한 전처리 모델과 인공 신경망을 결합하여 사용한다.
  - YOLO는 단 하나의 인공 신경망을 사용하여 처리한다(simple).

- **실시간 객체 탐지 모델**
  - PyTorch로 구현된 첫번째 YOLO 모델
  - 최상의 정확도(mAP) 보다는 사용의 편리함, 실시간 속도 등에 중점을 둠.
  - 계속 발전하고 있음(YOLO v6, v7, v8)

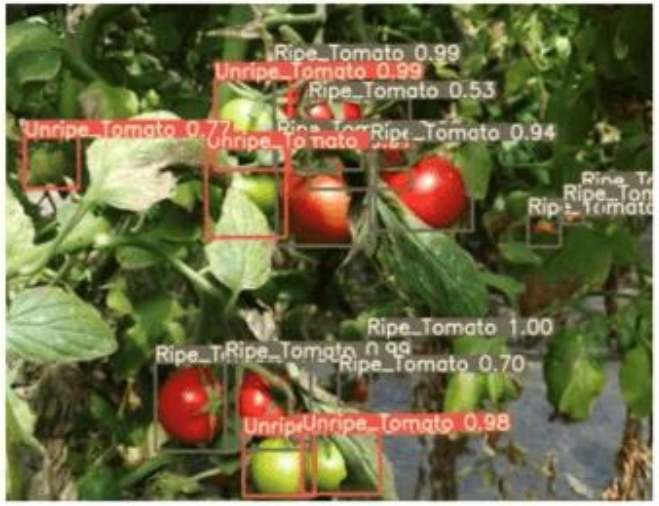
YOLO 구조


448
448
7
7
3
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer
1 convolution
1 convolution
4 convolutions
4x2 + 2 = 10
convolutions
2x2 + 2 = 6
convolutions
2 convolutions
24 convolution layers

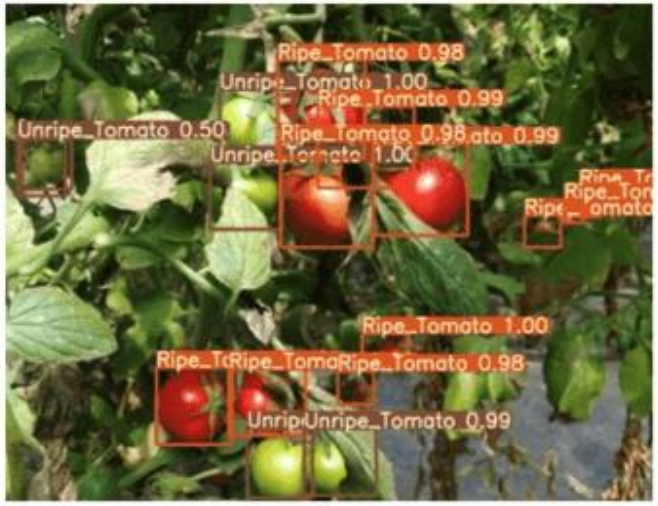
교통



농업


Ripe_Tomato 0.99
Unripe_Tomato 0.99
Ripe_Tomato 0.53
Unripe_Tomato 0.77
Ripe_Tomato 0.94
Ripe_Tomato 1.00
Ripe_Tomato 0.70
Unripe_Tomato 0.98

(a)


Ripe_Tomato 0.98
Unripe_Tomato 1.00
Ripe_Tomato 0.99
Unripe_Tomato 0.50
Ripe_Tomato 1.00
Ripe_Tomato 0.98
Unripe_Tomato 0.99

(b)

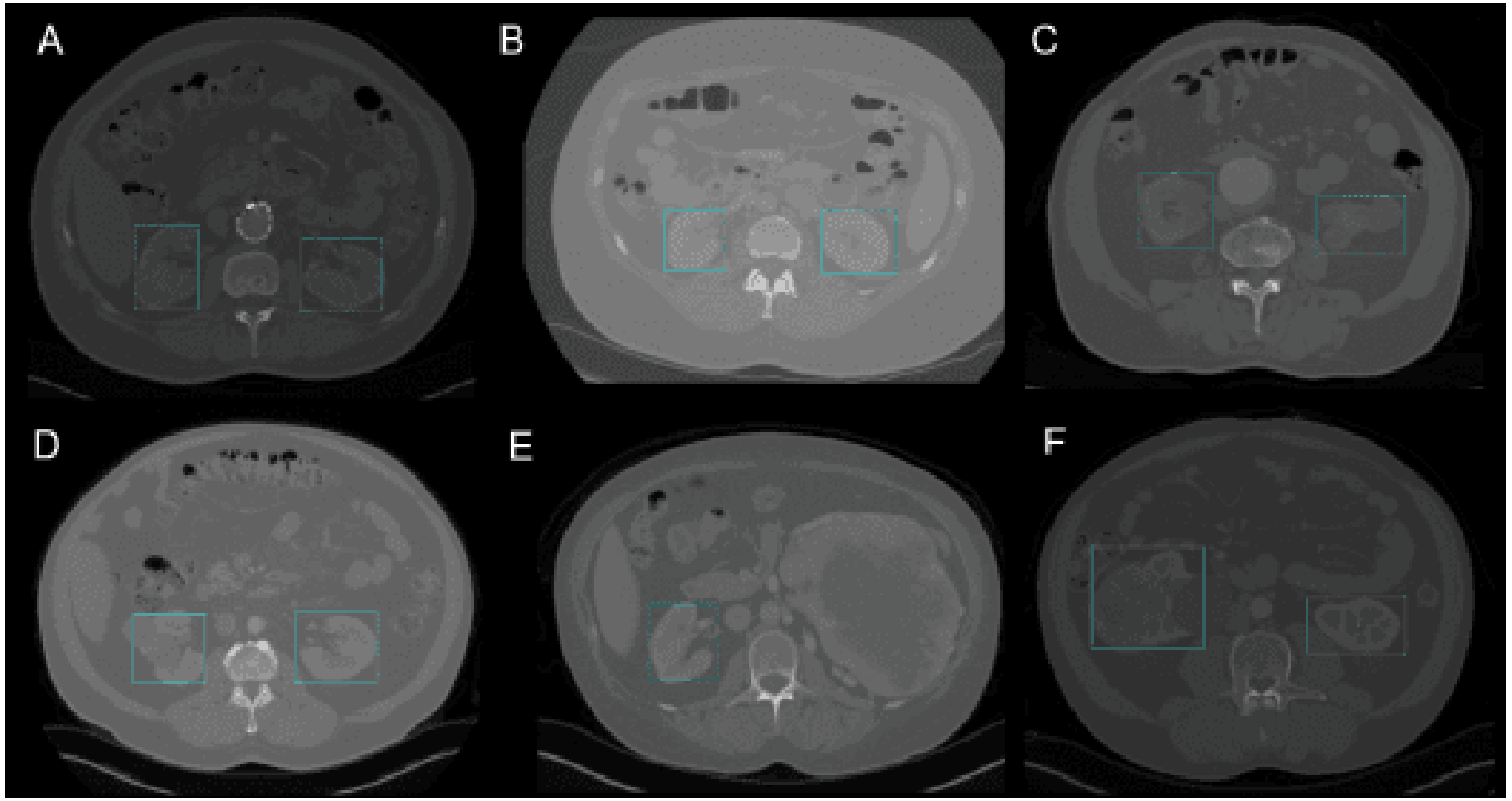

Ripe_Tomato 0.99
Ripe_Tomato 0.99
Unripe_Tomato 0.49
Ripe_Tomato 0.97
Ripe_Tomato 0.99
Ripe_Tomato 0.97
Unripe_Tomato 0.99

(C)


Ripe_Tomato 0.99
Ripe_Tomato 0.78
Ripe_Tomato 0.94
Unripe_Tomato 0.89
Ripe_Tomato 0.99
Ripe_Tomato 0.79
Ripe_Tomato 0.92
Ripe_Tomato 0.99
Unripe_Tomato 0.92
Ripe_Tomato 1.00

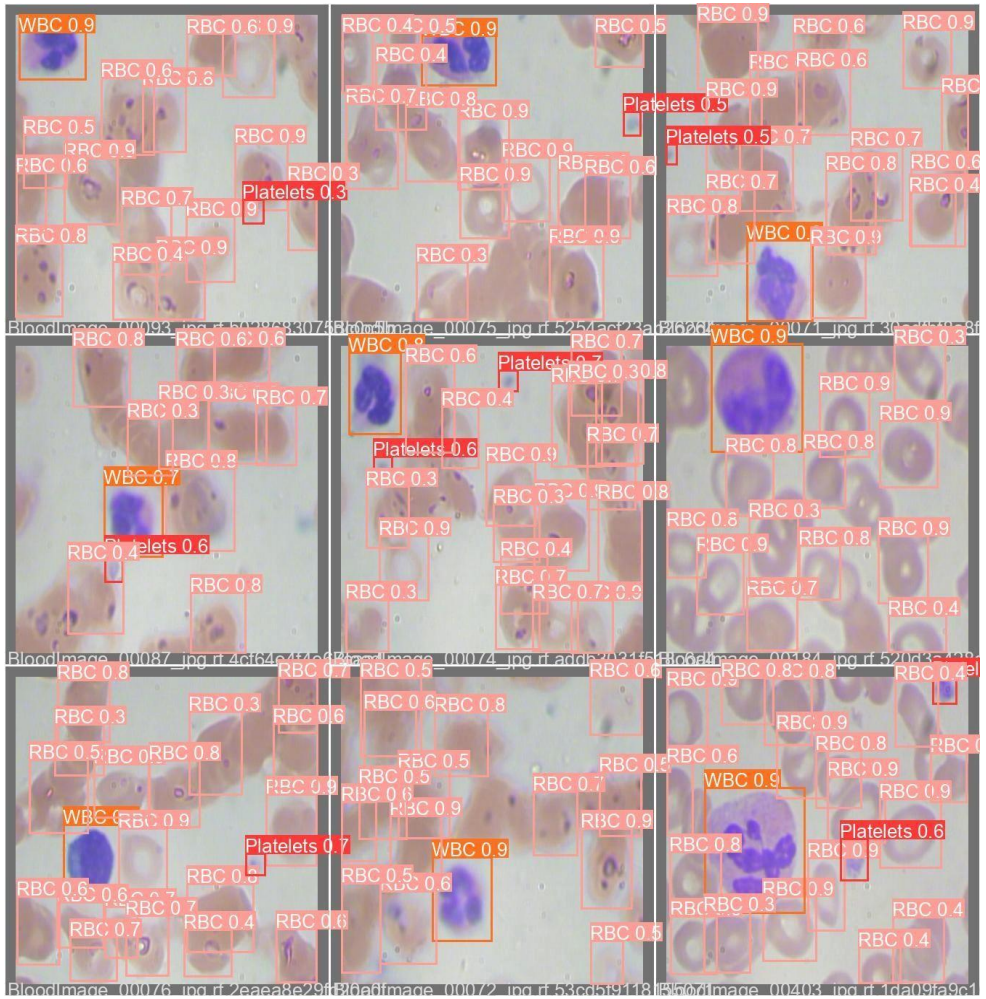
(d)

의료



의료

- 혈액세포 내 백혈구, 적혈구, 혈소판 탐지



Quick Start Examples

- 오픈 소스 저장소
  - https://github.com/ultralytics/yolov5

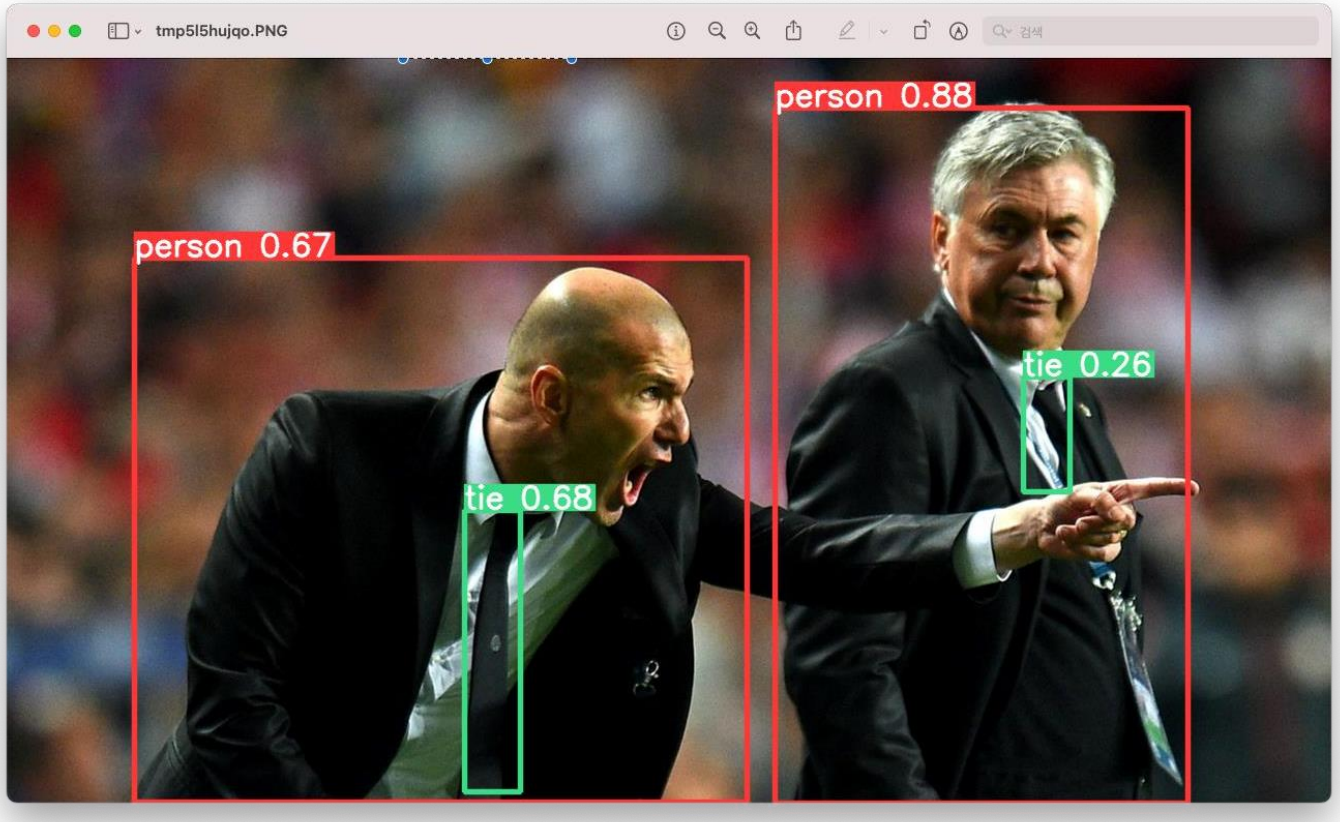
- 설치

```
% git clone https://github.com/ultralytics/yolov5  # clone
% cd yolov5
% pip install -r requirements.txt  # install
```

Pre-trained 모델 추론

```
import torch

# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')    # or yolov5n - 
yolov5x6, custom

# Images
img = 'https://ultralytics.com/images/zidane.jpg'  # or file, Path, PIL, 
OpenCV, numpy, list

# Inference
results = model(img)

# Results
results.show()  # or .print(), .save(), .crop(), .pandas(), etc.
```

Pre-trained 모델 추론

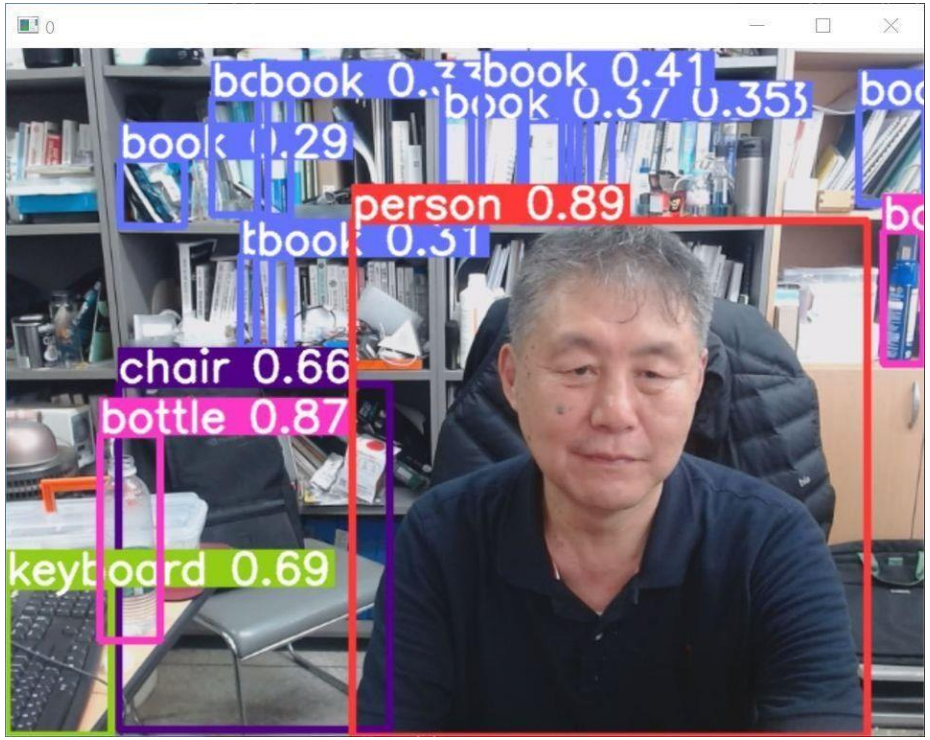


Yolo v5 Pre-trained 모델 : 전이 학습

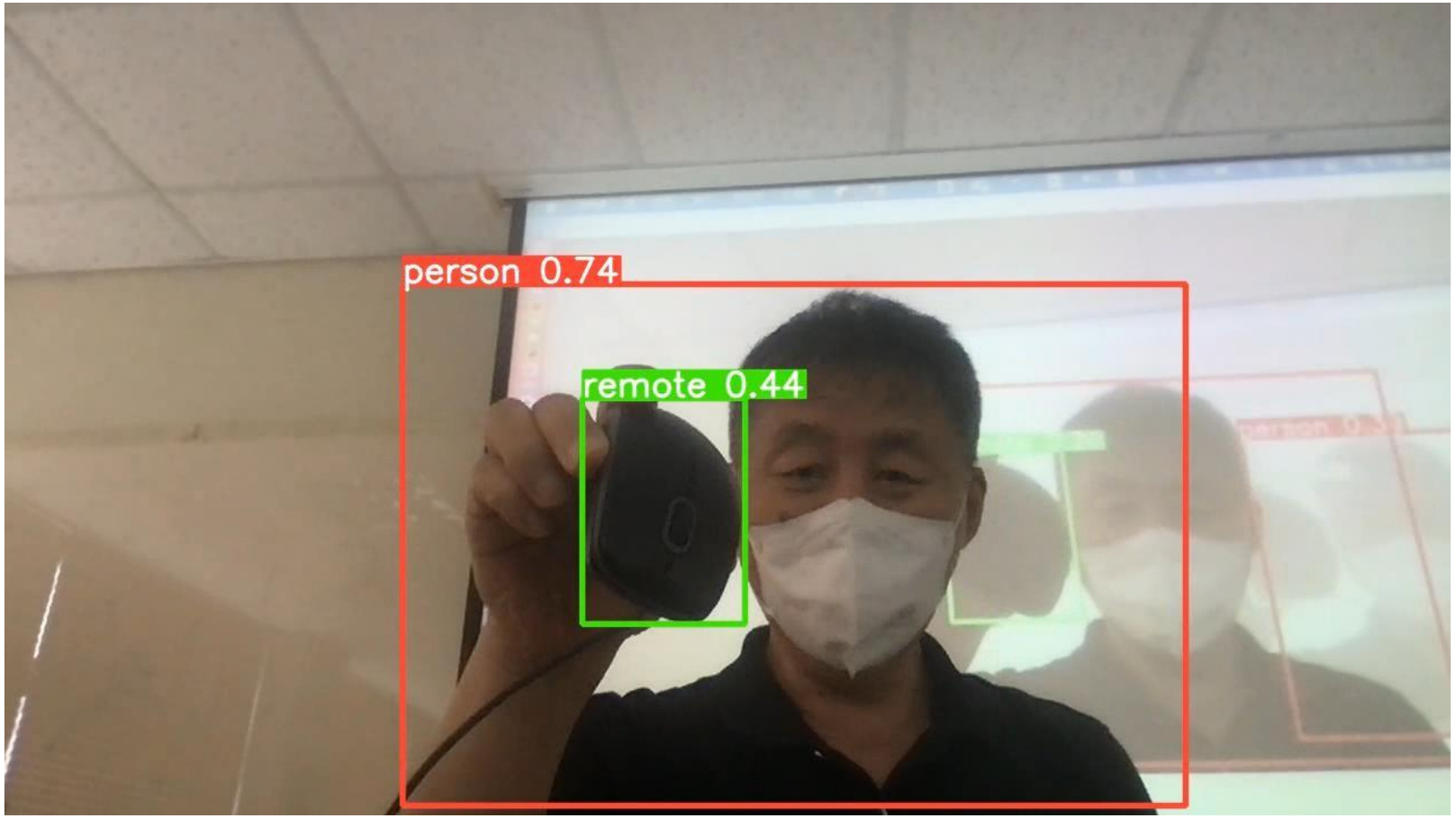
```
nc: 80  # number of classes
names: ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
        'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
        'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
        'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
        'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
        'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
        'hair drier', 'toothbrush']  # class names
```

Pre-trained 모델 추론 : detect.py

```
% python detect.py --source 0  # webcam
                    img.jpg  # image
                    vid.mp4  # video
                    path/  # directory
                    path/*.jpg  # glob
                    'https://youtu.be/Zgi9g1ksQHc'  # YouTube
                    'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP stream
```

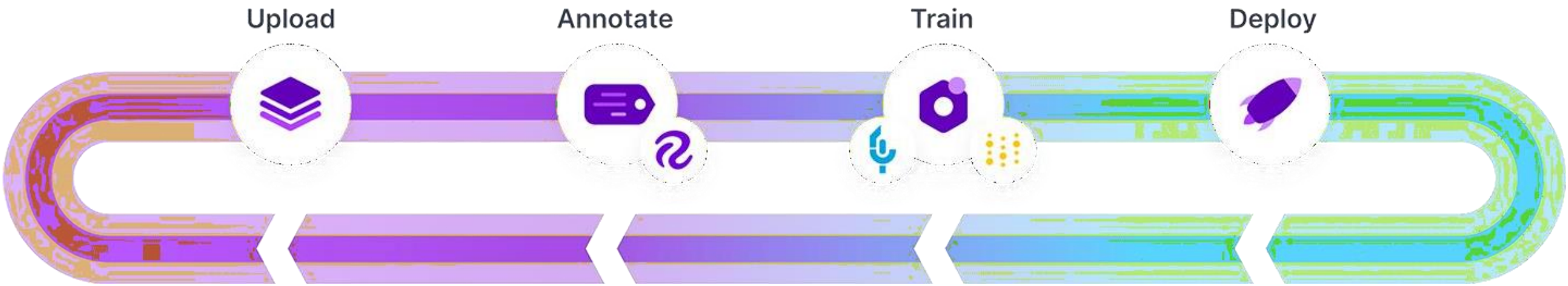


Pre-trained 모델 추론 : detect.py



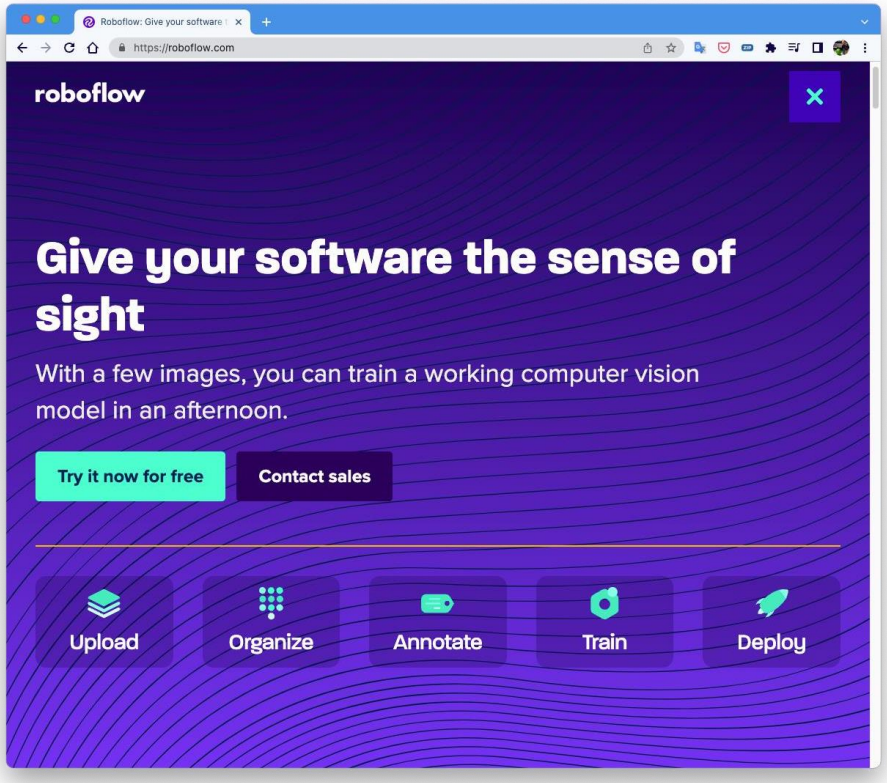
커스텀 데이터 학습 : Custom Data Training

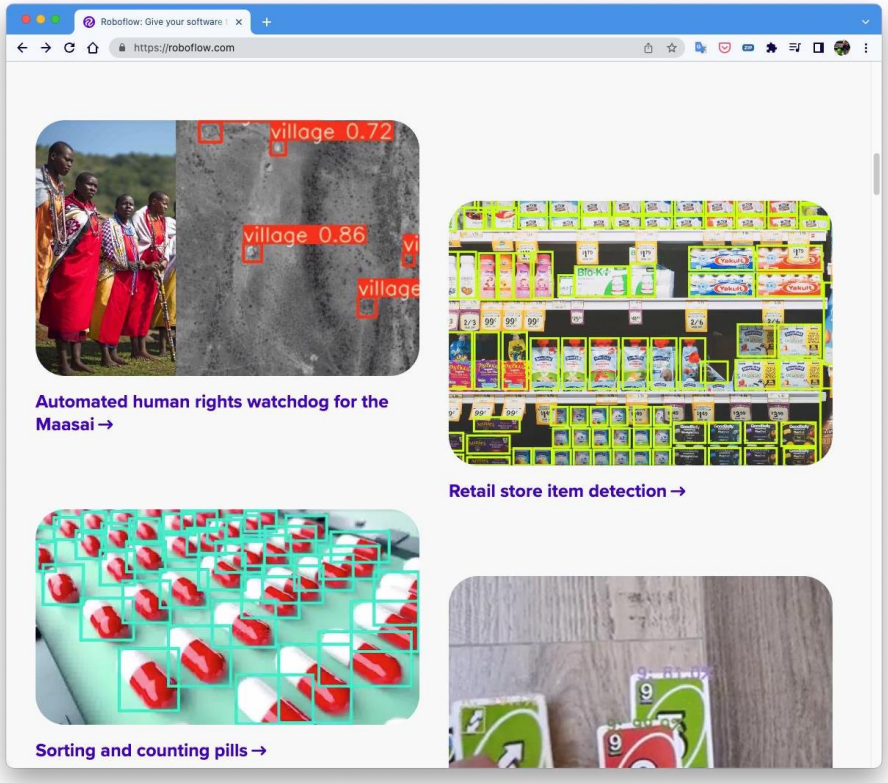
- https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data



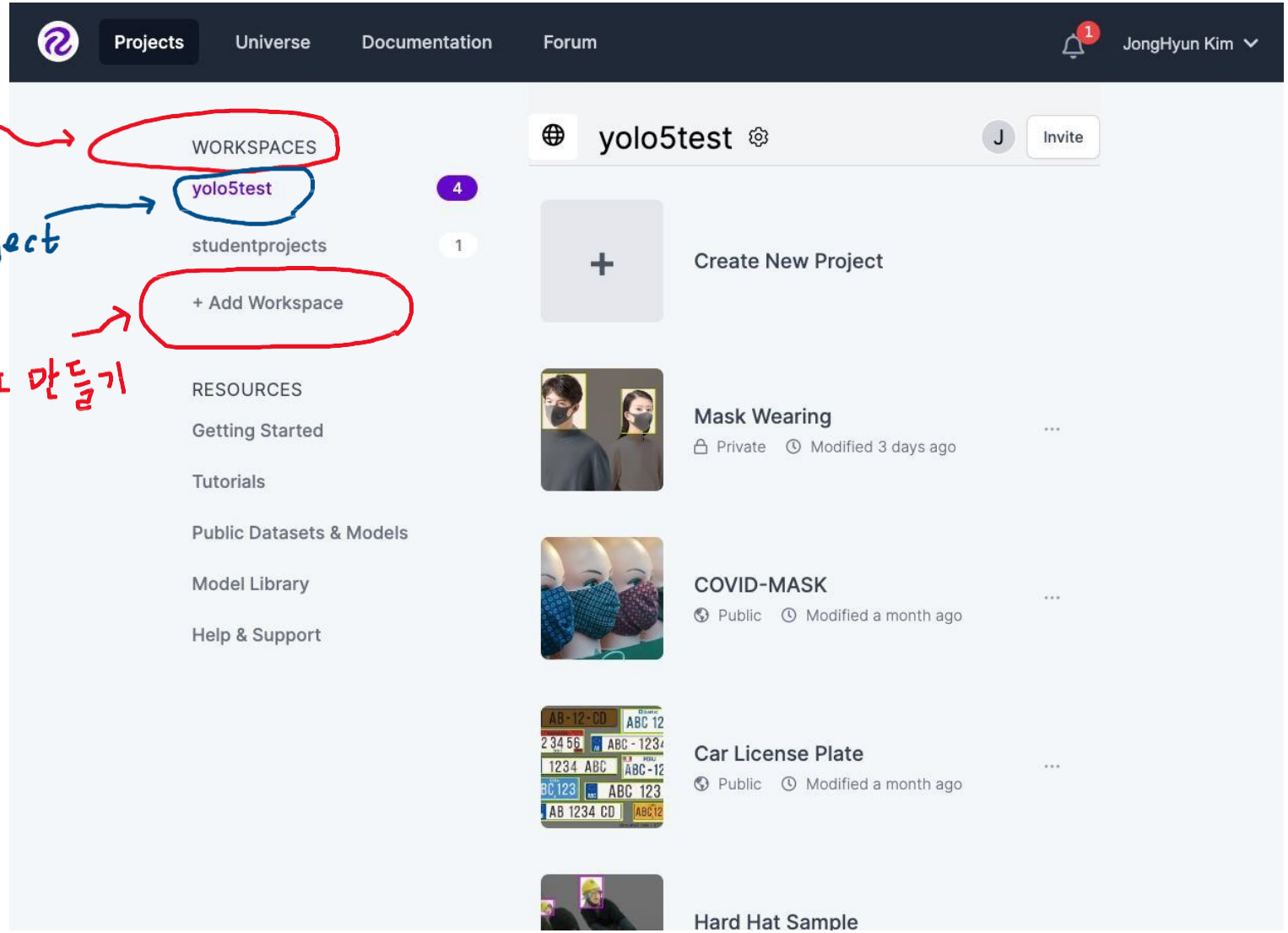
데이터 수집 및 전처리

- roboflow
  - https://roboflow.com
  - 계정 생성 : Sign up for free





Worksapce/ Project 만들기



실습 01: 공개 데이터 셋을 이용한 객체 탐지

- 혈액 세포 탐지 : Platelets, RBC, WBC
  - https://universe.roboflow.com/search?q=BCCD&t=metadata

실습 02 : 공개 데이터 셋을 이용한 객체 탐지

- 자동차 탐지 : bike, cycle, pedestrian, vehicle
- https://universe.roboflow.com/muhammad-tamjid-rahman-
- b6xvp/a4.1/dataset/1

실습 03 : 공개 데이터 셋을 이용한 객체 탐지

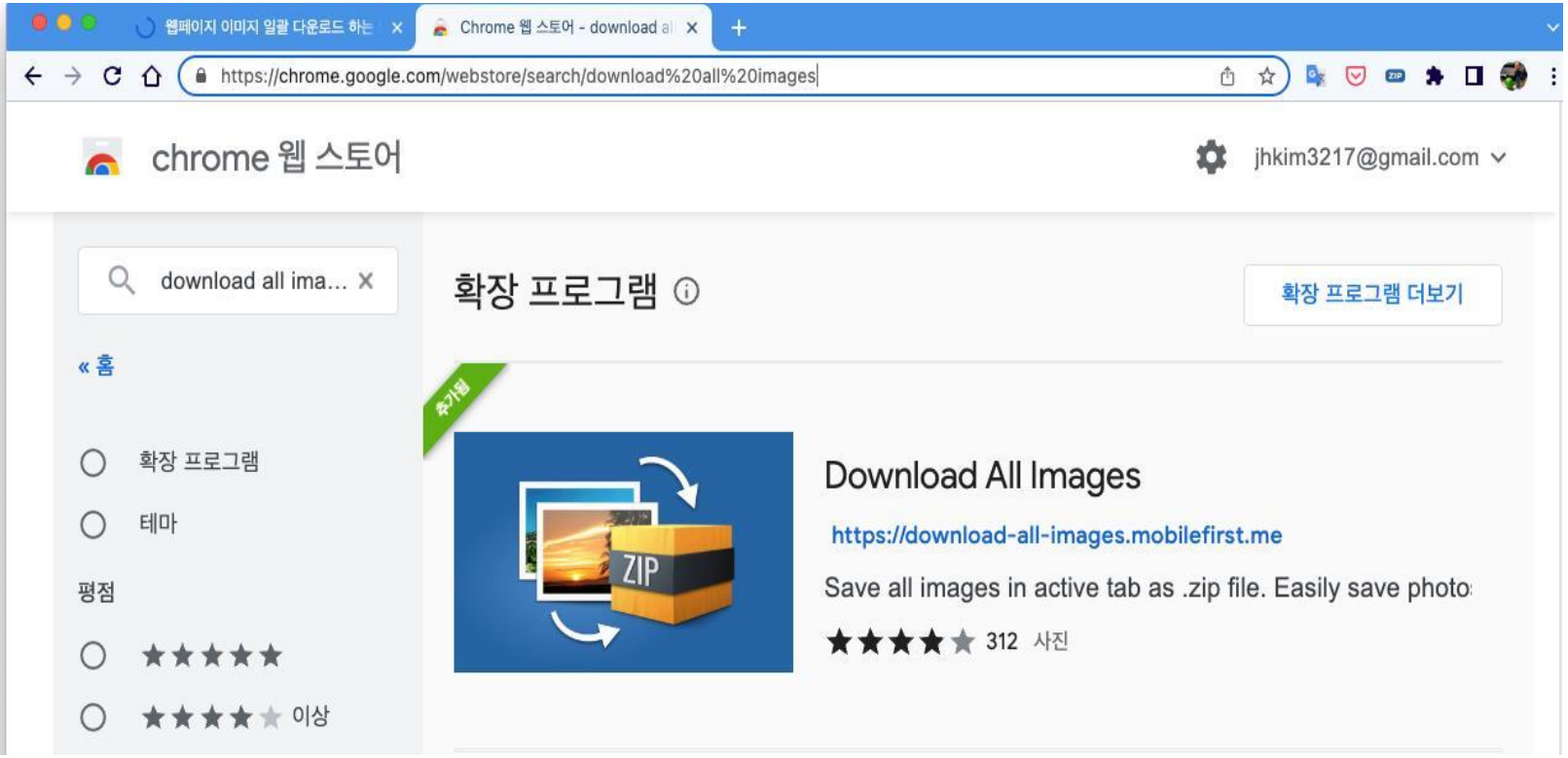
- 권총 탐지 : Pistols
- 데이터 셋 : https://universe.roboflow.com/weapon-detection-using-yolov5/vishnu/dataset/5
- colab 소스: https://colab.research.google.com/drive/18D03OYu8QHNcdc2UNcGUmxh6Vu7uZAKo?usp=sharing

실습 04 : 직접 수집한 데이터 셋을 이용한 객체 탐지

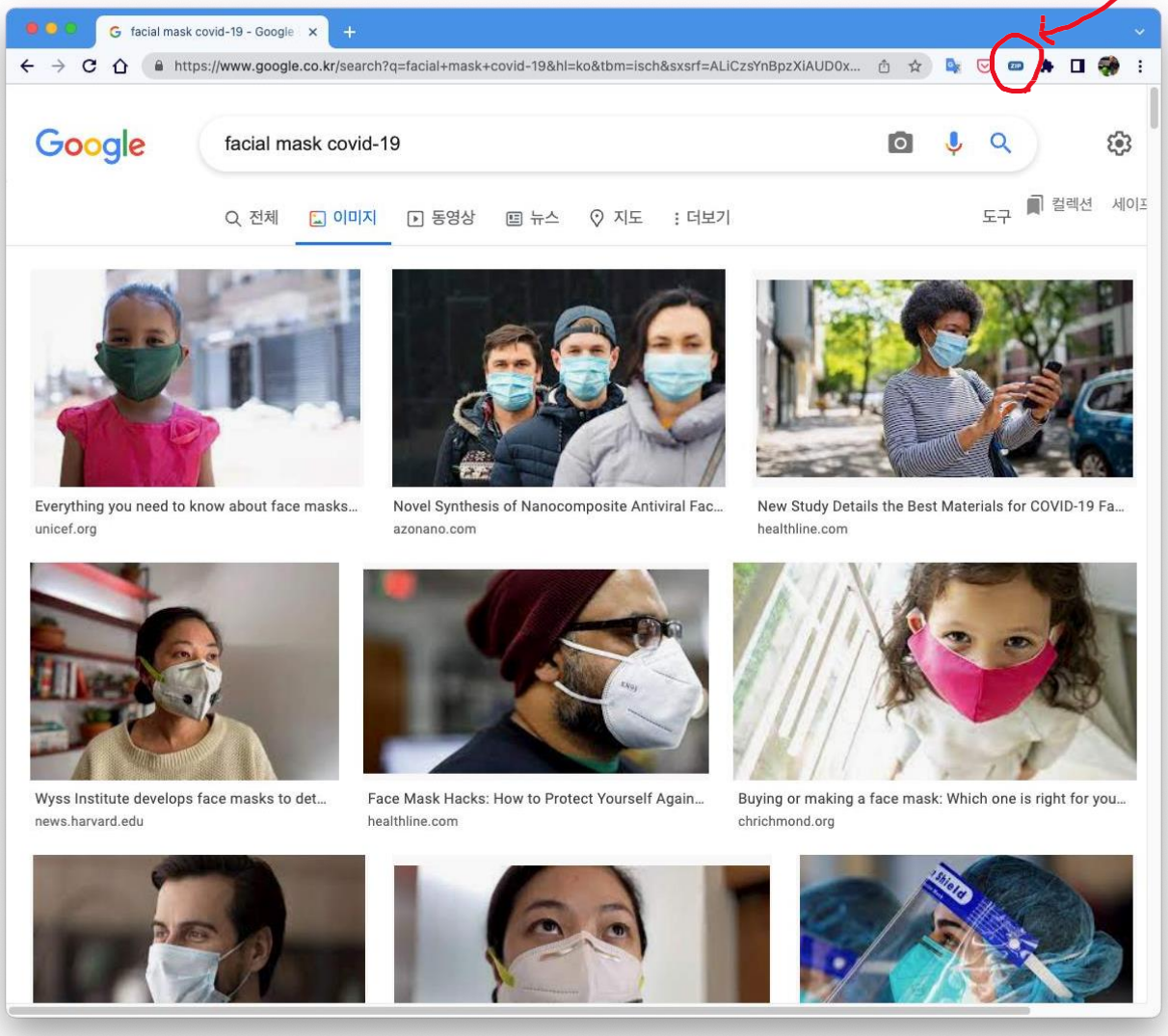
- 실시간 마스크 착용 여부 탐지
  - 데이터 수집
    - 구글 이미지 검색
  - 데이터 전처리, 라벨링, 바운딩
    - Roboflow.com
  - 데이터 학습, 성능평가, 모델 export
    - Google colab
  - AI 서비스 만들기
    - 로컬 PC

데이터 수집 : 구글 이미지 검색

- Chome 웹 스토어 : Download All Images
  - https://chrome.google.com/webstore/search/download%20all%20images

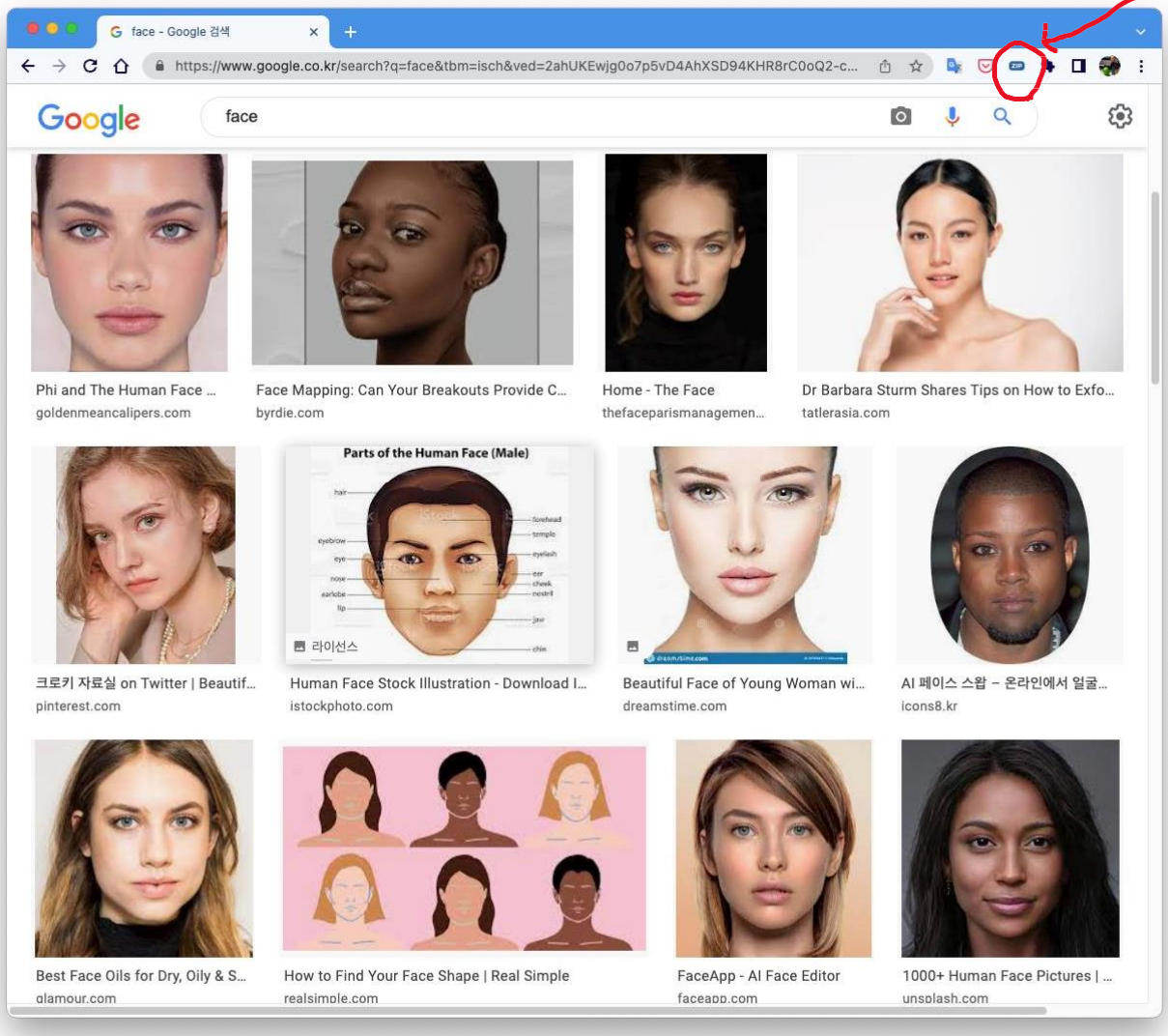


데이터 수집 : 구글 이미지 검색



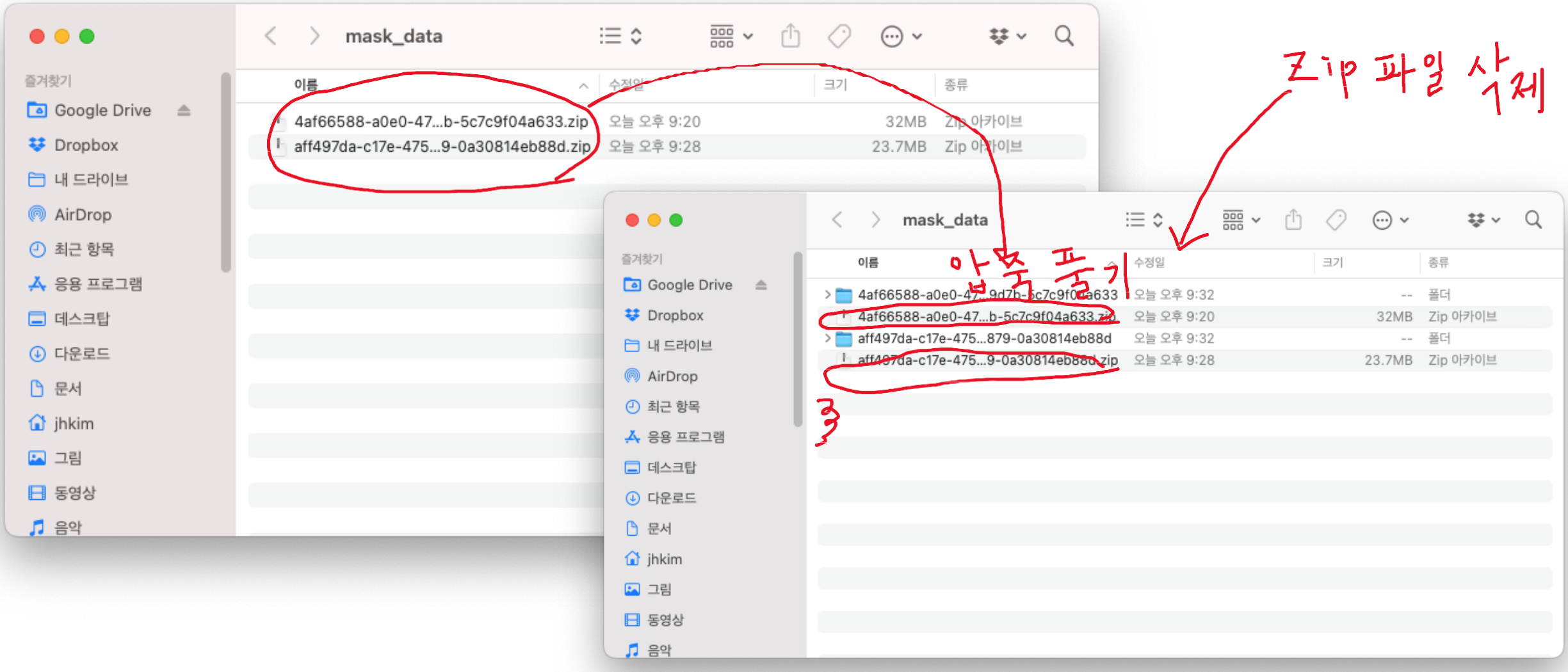
전체 이미지 다운로드 (zip)
클릭

데이터 수집 : 구글 이미지 검색

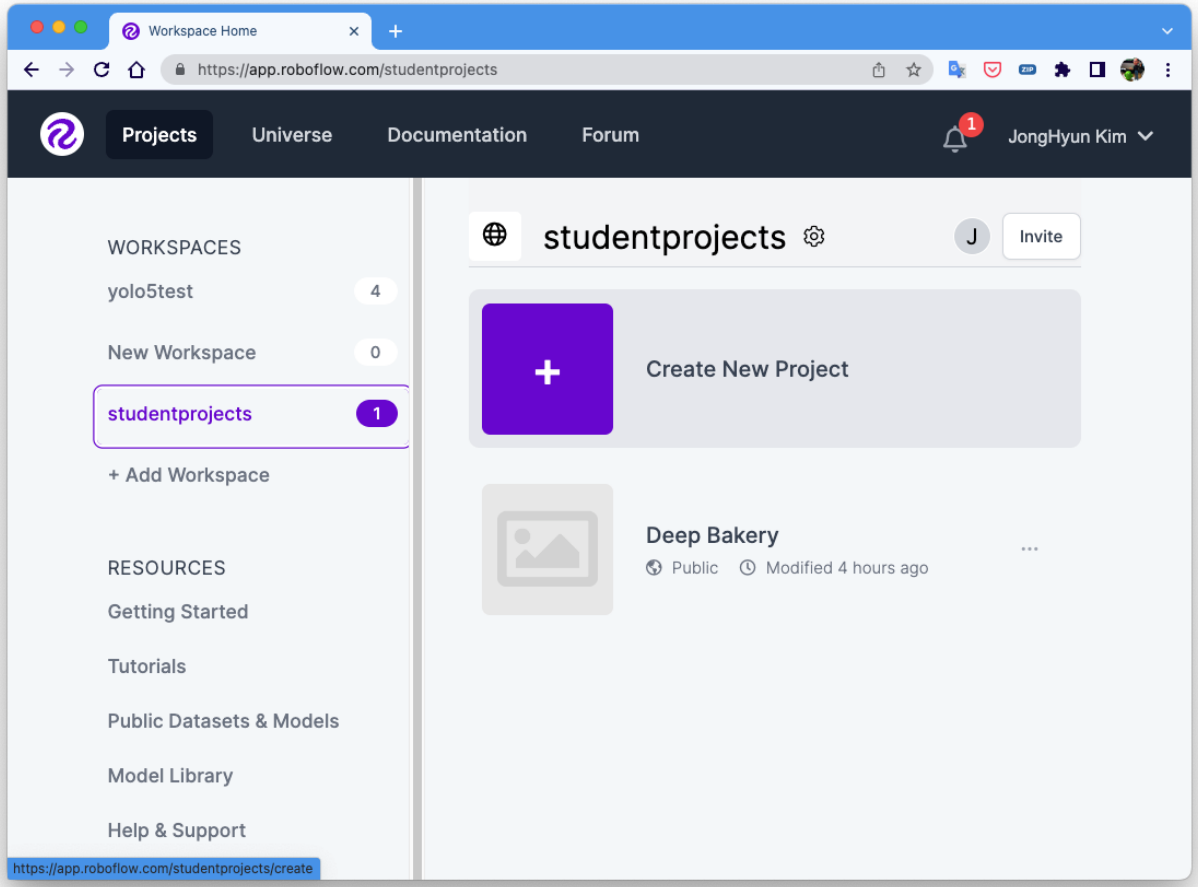


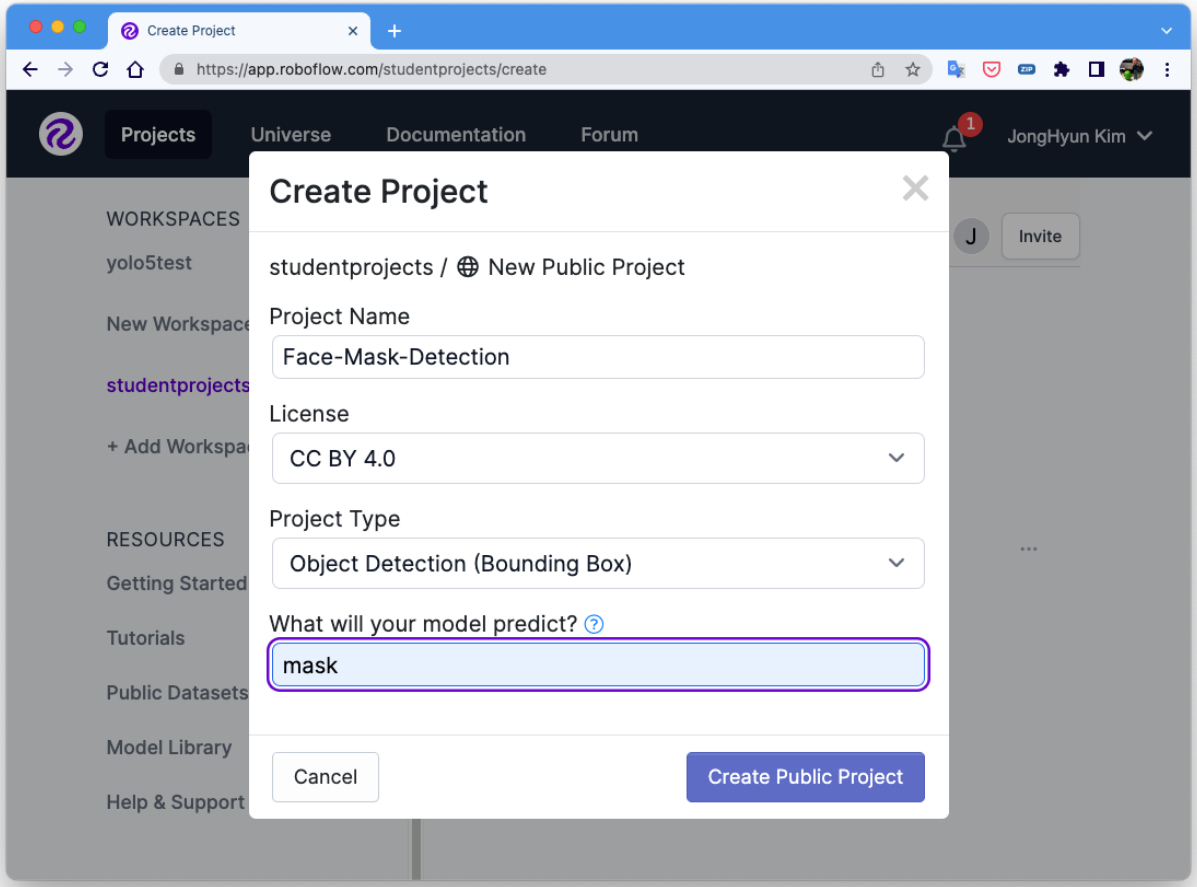
(zip)
전체 이미지 다운로드
클릭

데이터 수집 : 구글 이미지 검색

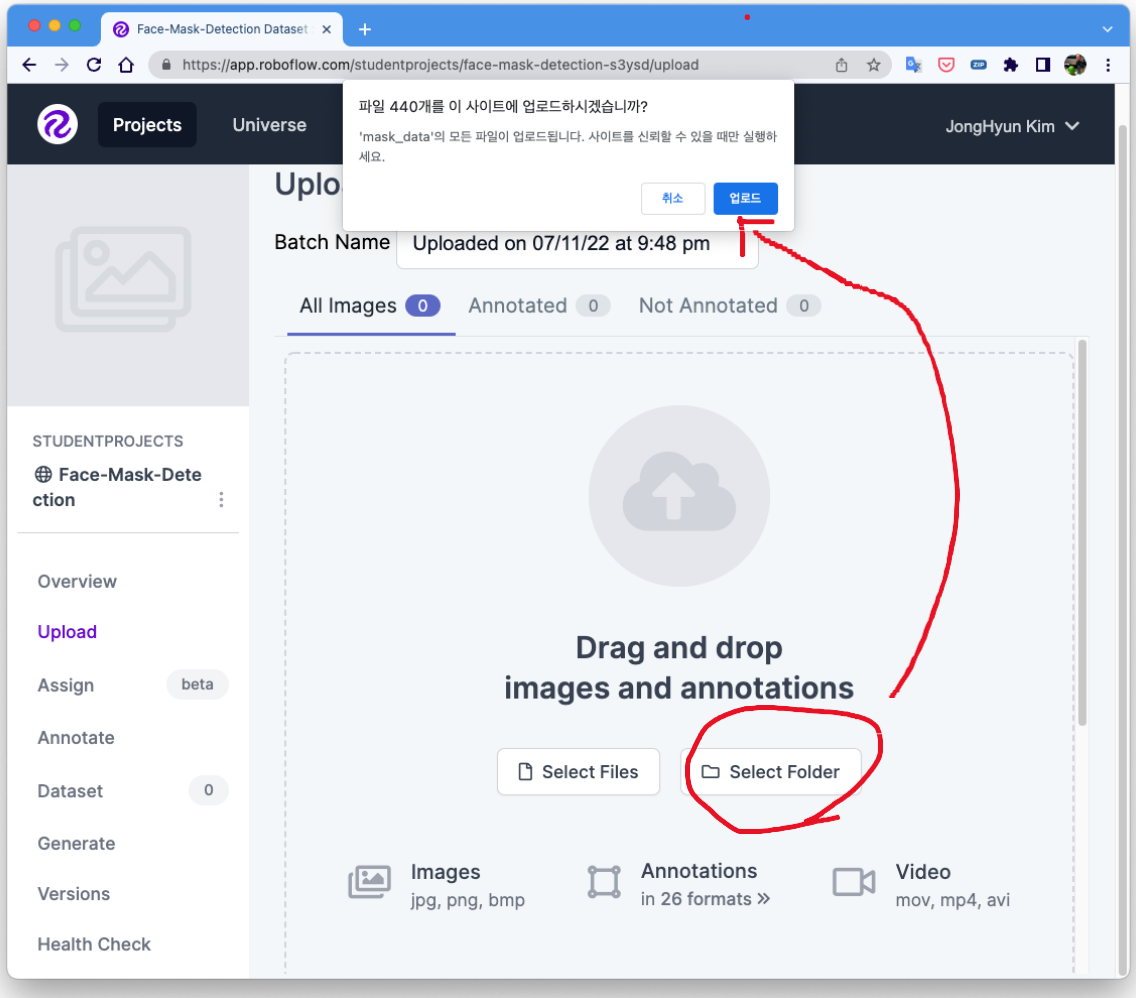


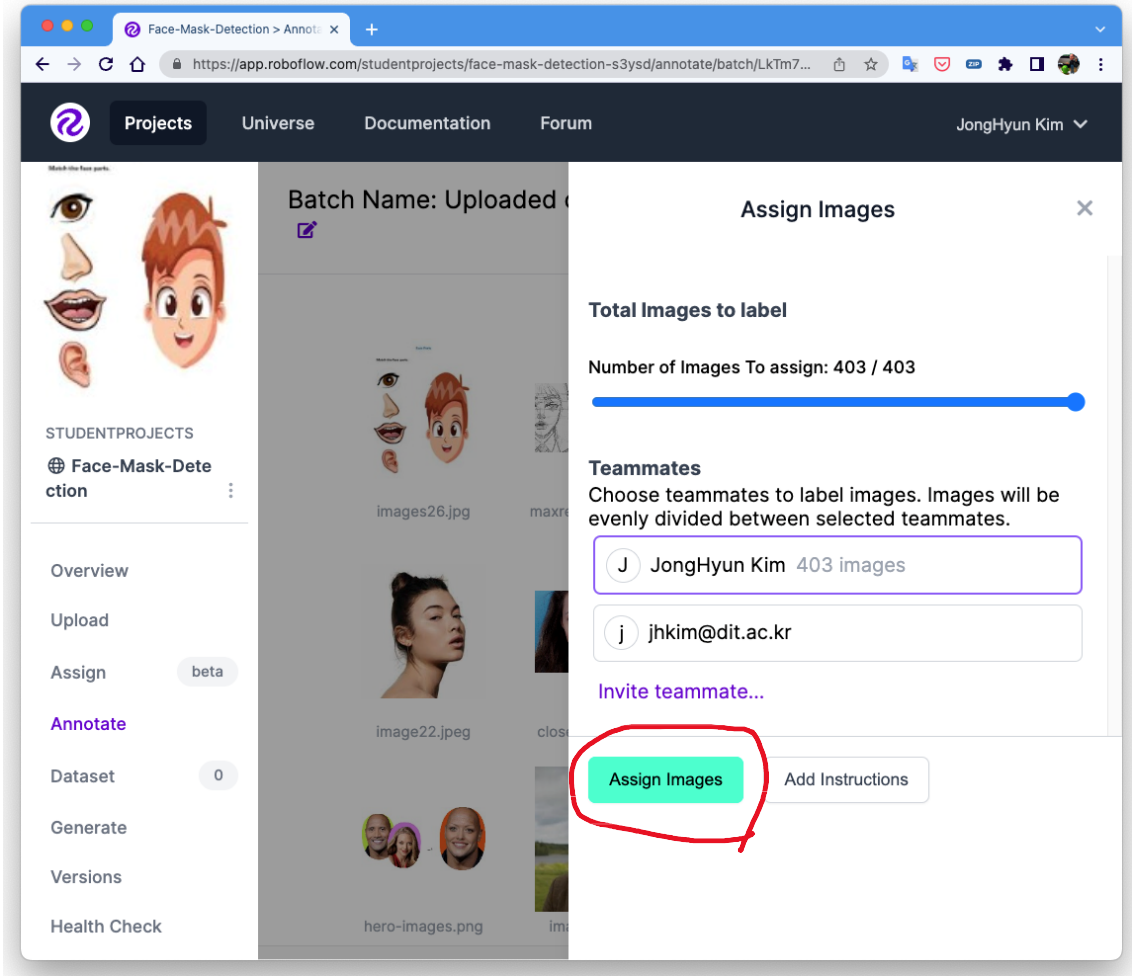
Workspace -> Project 만들기



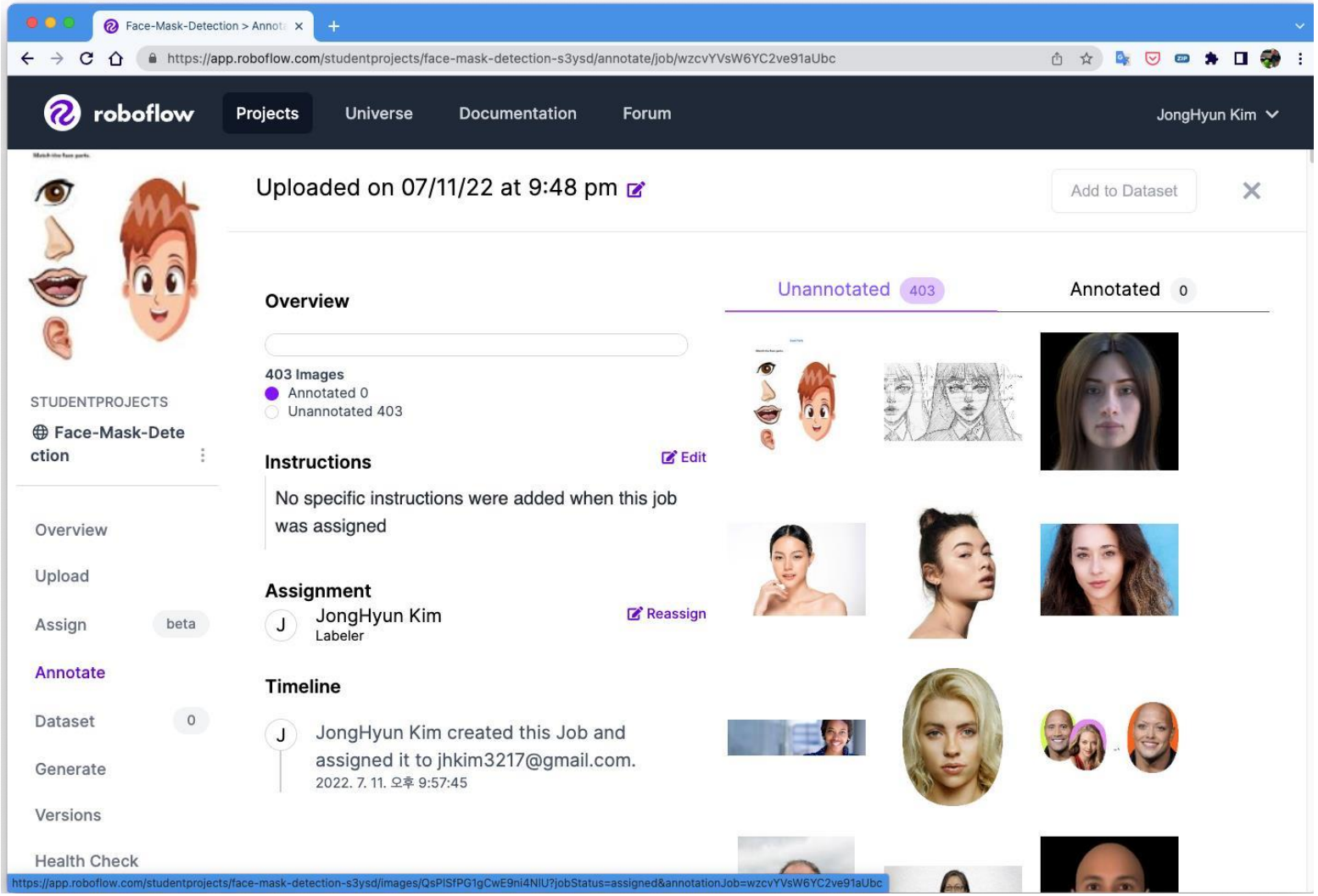


데이터 업로드

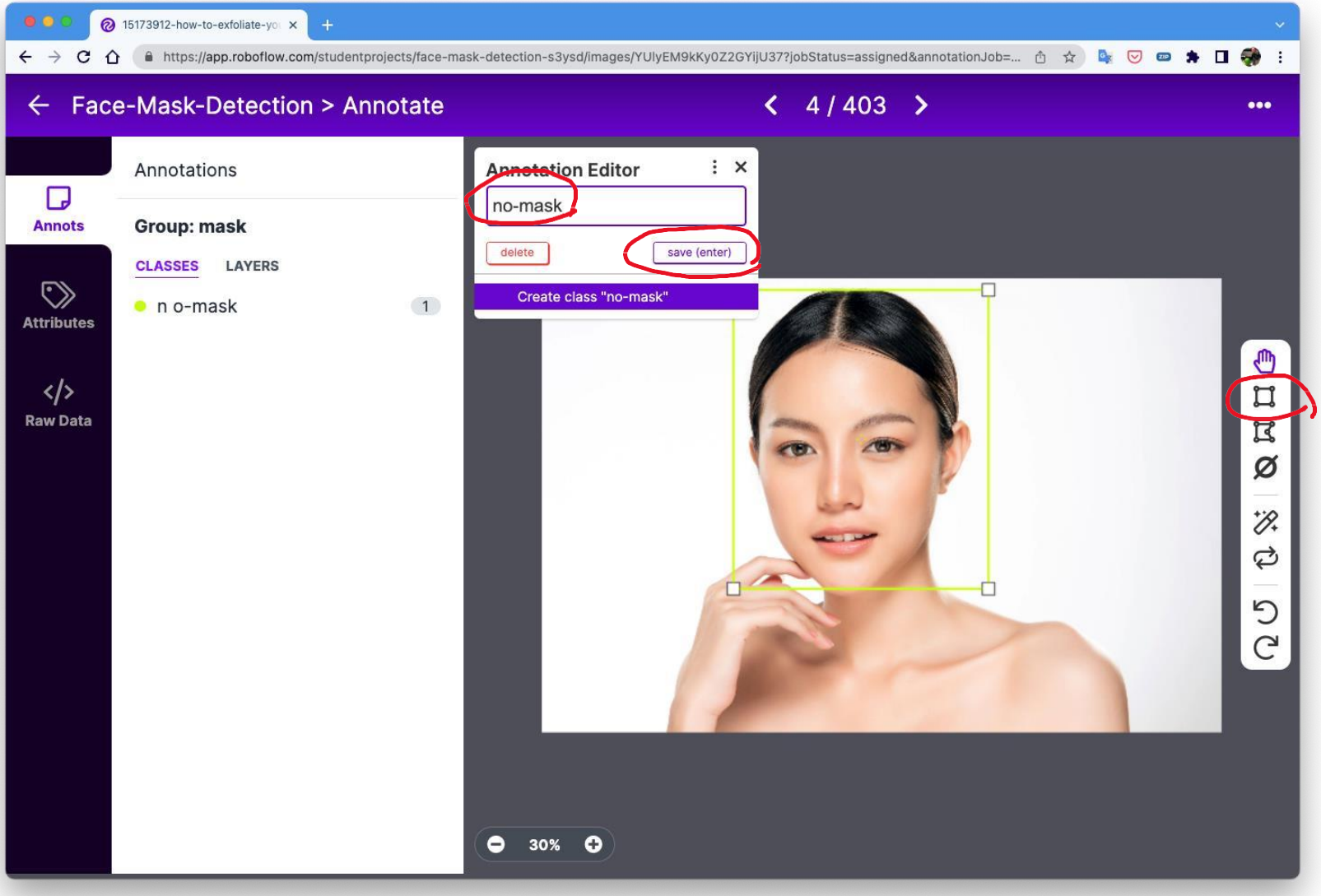

Face-Mask-Detection Dataset
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/upload
Projects
Universe
JongHyun Kim
파일 440개를 이 사이트에 업로드하시겠습니까?
'mask_data'의 모든 파일이 업로드됩니다. 사이트를 신뢰할 수 있을 때만 실행하세요.
취소
업로드
Batch Name
Uploaded on 07/11/22 at 9:48 pm
All Images 0
Annotated 0
Not Annotated 0
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign beta
Annotate
Dataset 0
Generate
Versions
Health Check
Drag and drop images and annotations
Select Files
Select Folder
Images
jpg, png, bmp
Annotations
in 26 formats
Video
mov, mp4, avi


Face-Mask-Detection > Annotate
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/annotate/batch/LkTm7...
Projects
Universe
Documentation
Forum
JongHyun Kim
Batch Name: Uploaded
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign beta
Annotate
Dataset 0
Generate
Versions
Health Check
images26.jpg
image22.jpeg
hero-images.png
Assign Images
Total Images to label
Number of Images To assign: 403 / 403
Teammates
Choose teammates to label images. Images will be evenly divided between selected teammates.
J JongHyun Kim 403 images
j jhkim@dit.ac.kr
Invite teammate...
Assign Images
Add Instructions

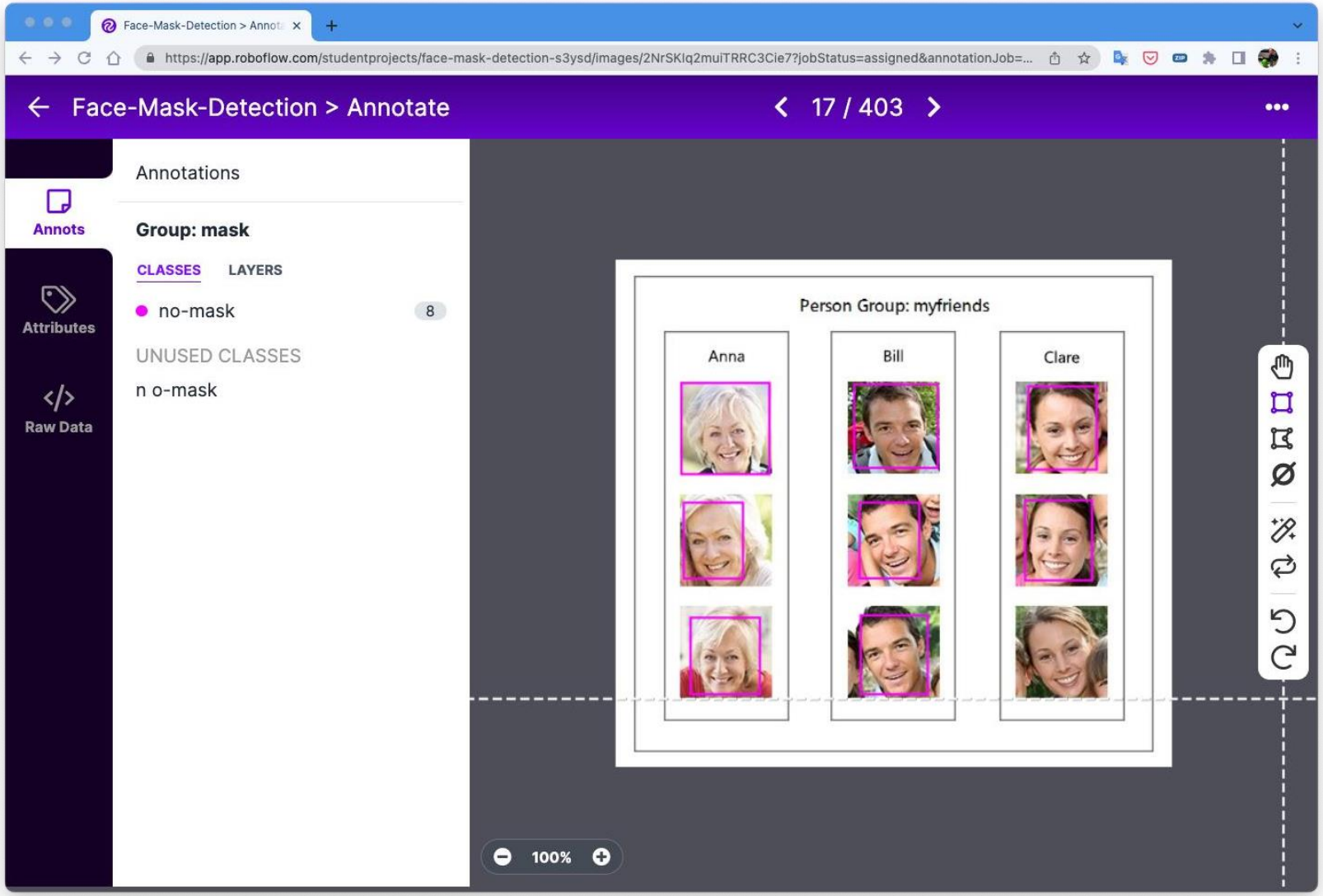
데이터 업로드


Face-Mask-Detection > Annot
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/annotate/job/wzcvYVsW6YC2ve91aUbc
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
Uploaded on 07/11/22 at 9:48 pm
Add to Dataset
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign
beta
Annotate
Dataset
0
Generate
Versions
Health Check
Overview
403 Images
Annotated 0
Unannotated 403
Instructions
Edit
No specific instructions were added when this job was assigned
Assignment
JongHyun Kim
Labeler
Reassign
Timeline
JongHyun Kim created this Job and assigned it to jhkim3217@gmail.com.
2022. 7. 11. 오후 9:57:45
Unannotated 403
Annotated 0
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/images/QsPISfPG1gCwE9ni4NIU?jobStatus=assigned&annotationJob=wzcvYVsW6YC2ve91aUbc

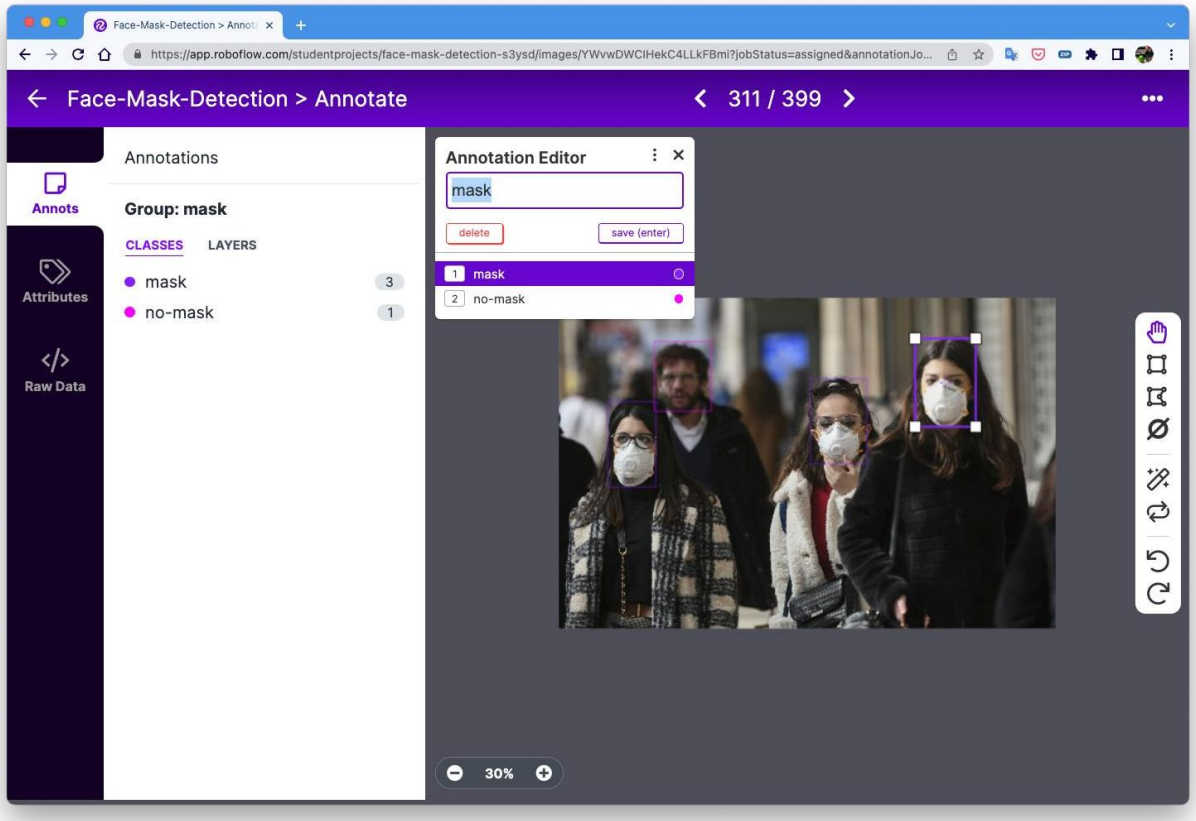
Annotate :데이터 라벨링, 바운딩

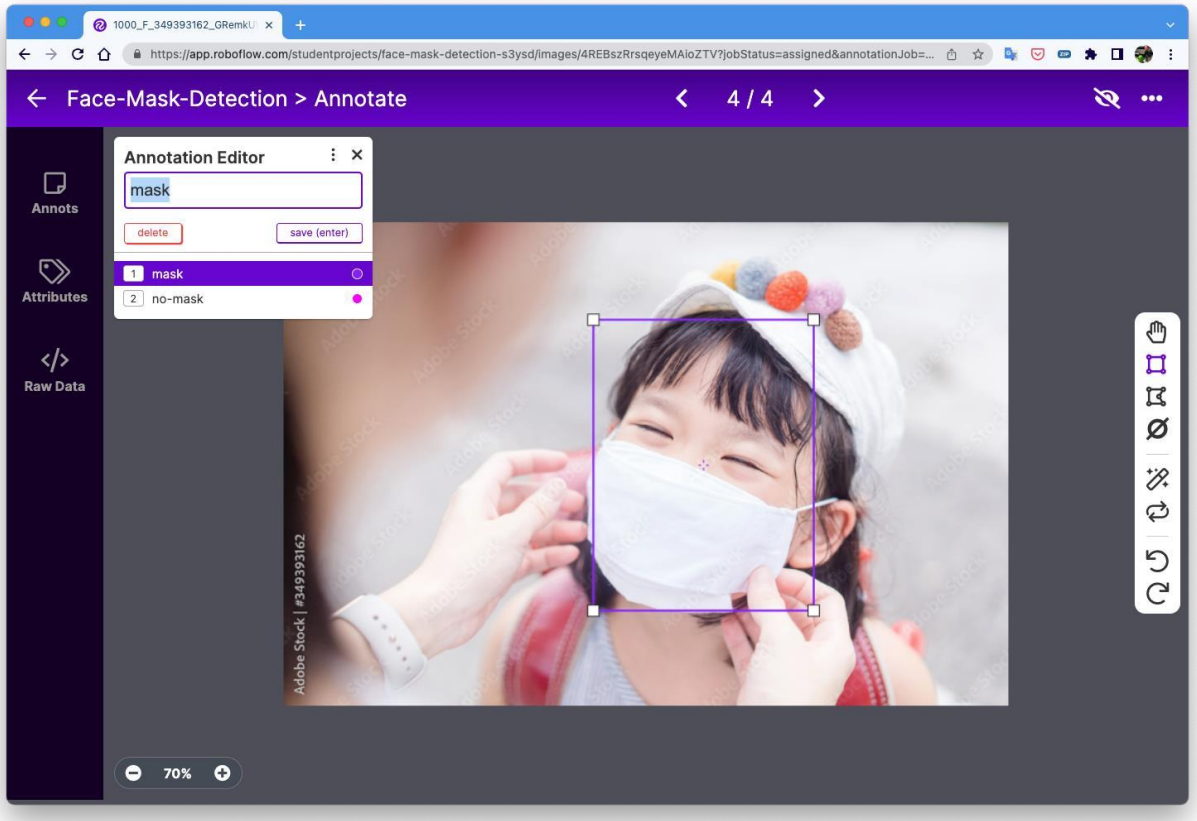


Annotate :데이터 라벨링, 바운딩

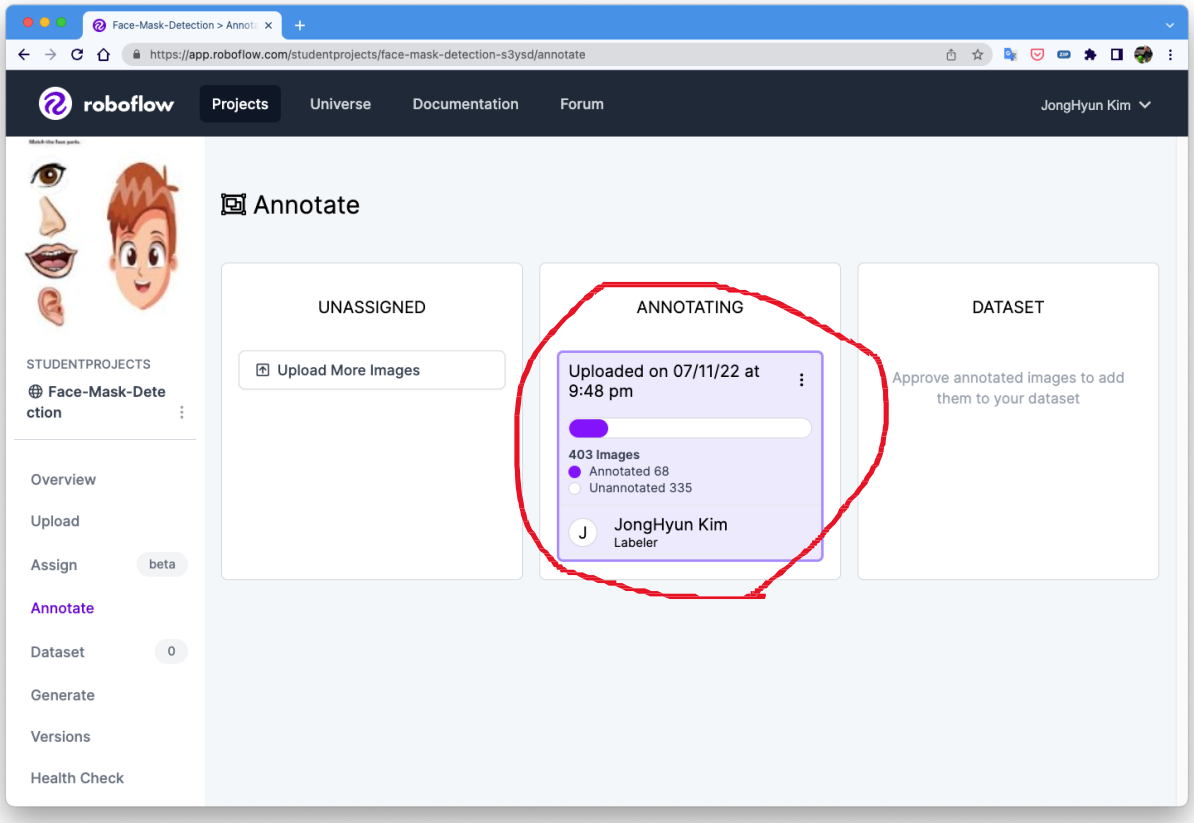


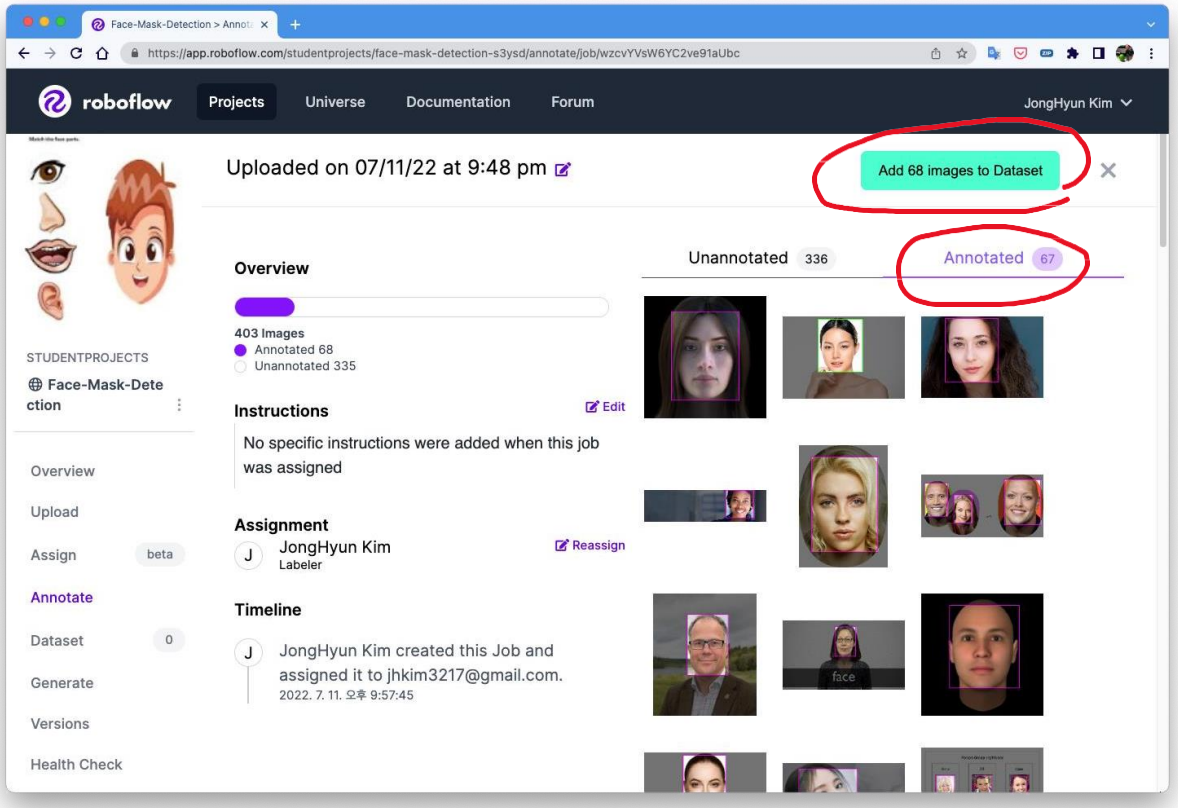
Annotate :데이터 라벨링, 바운딩



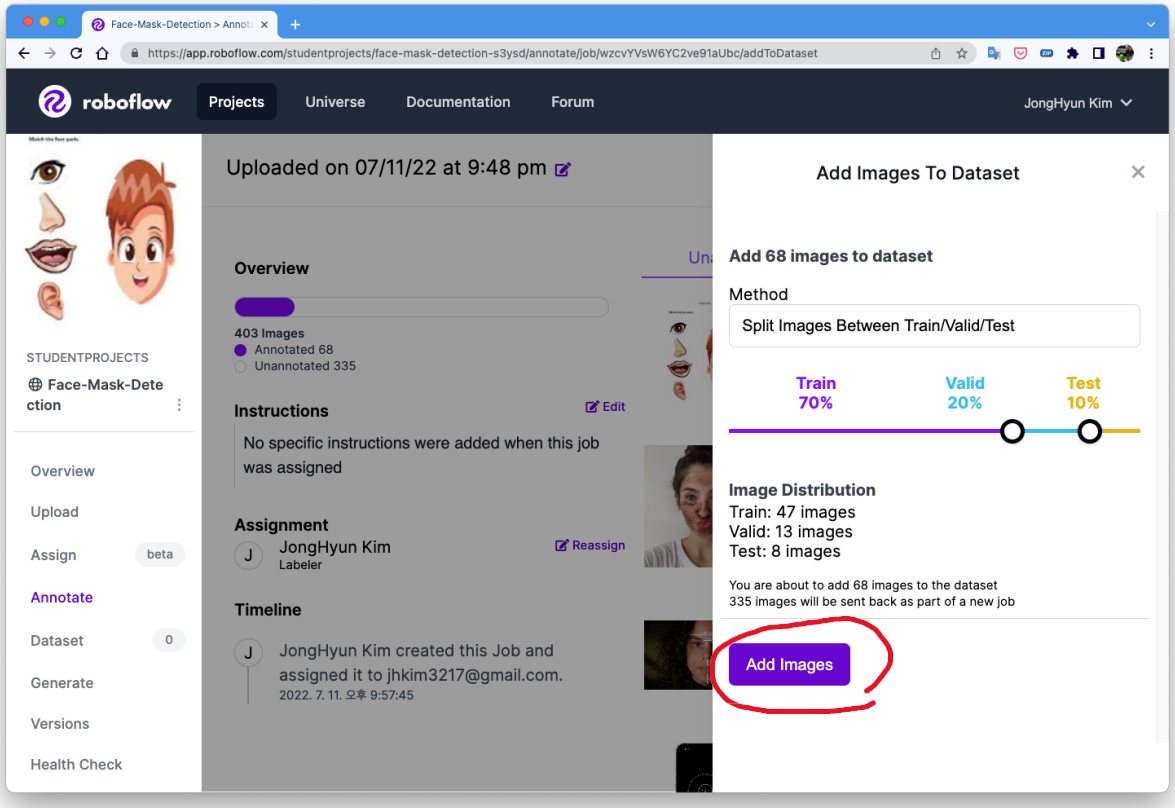


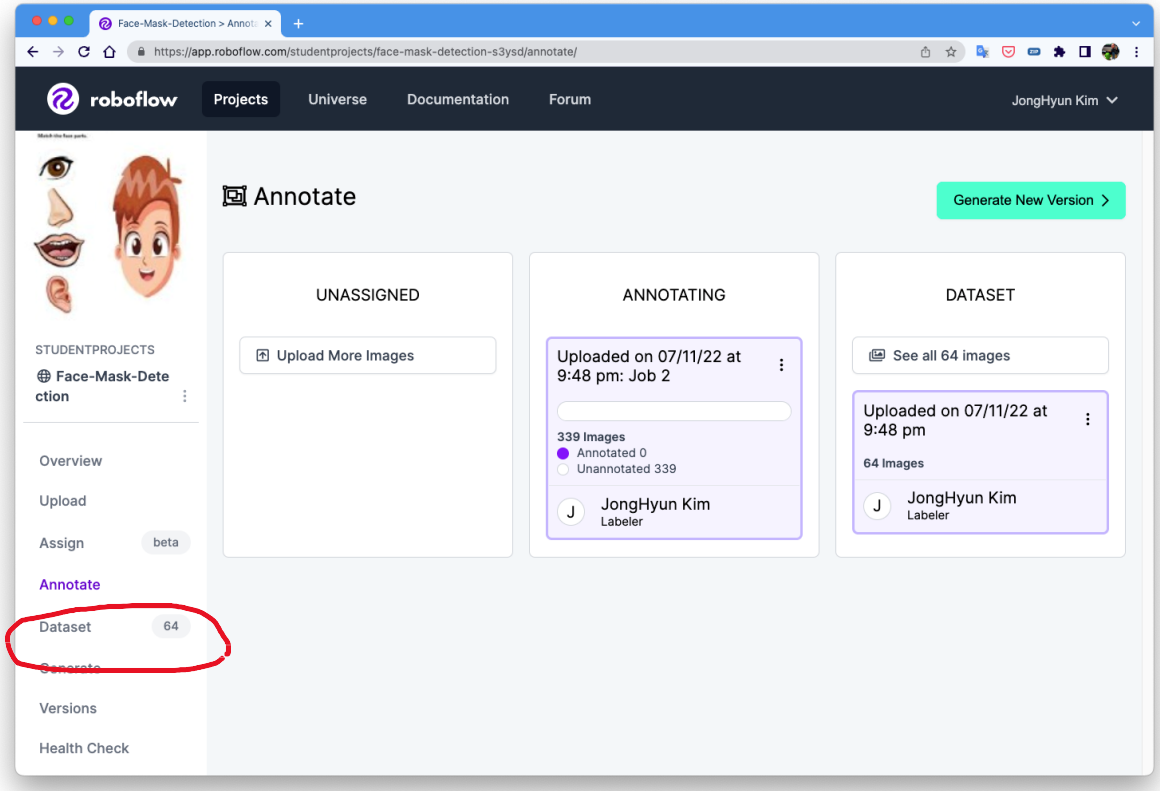
Annotate : 데이터 라벨링, 바운딩



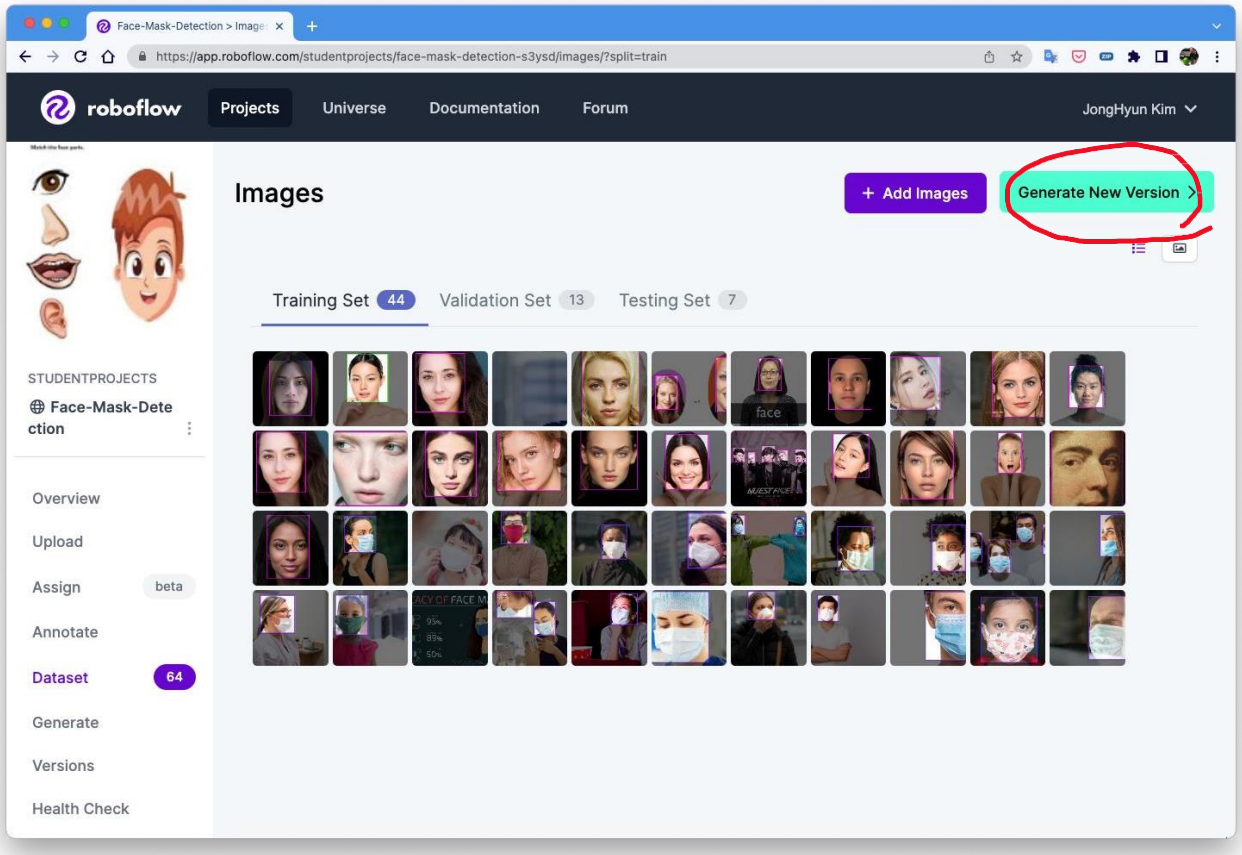


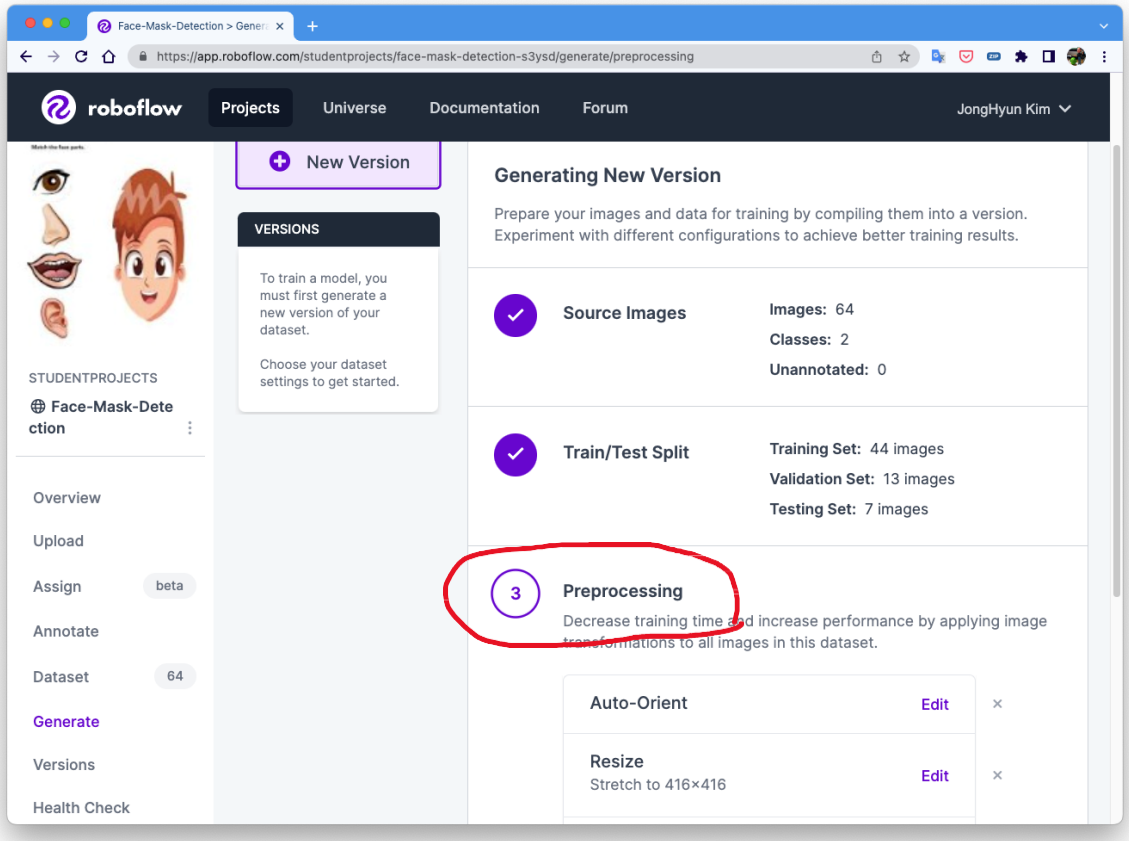
Annotate :데이터 라벨링, 바운딩



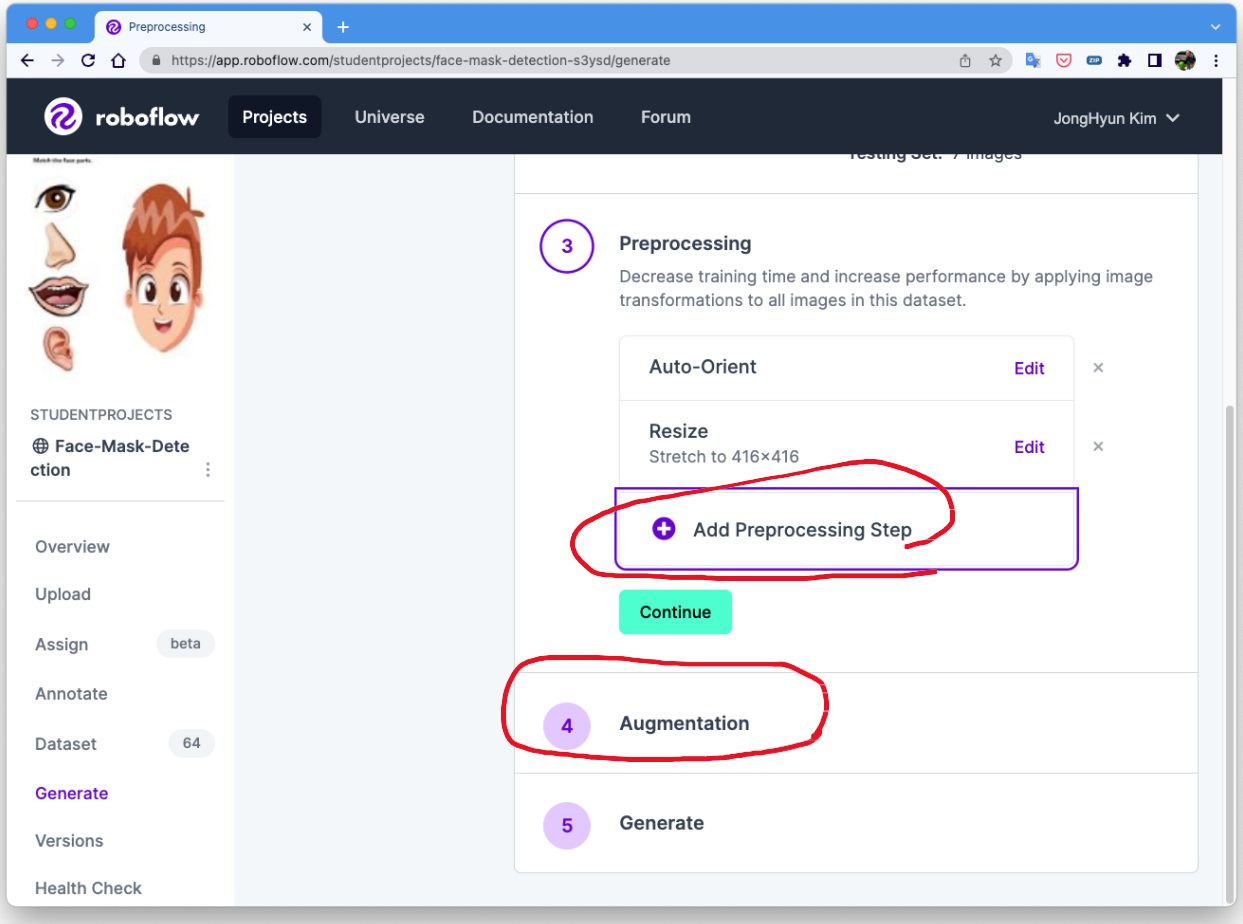


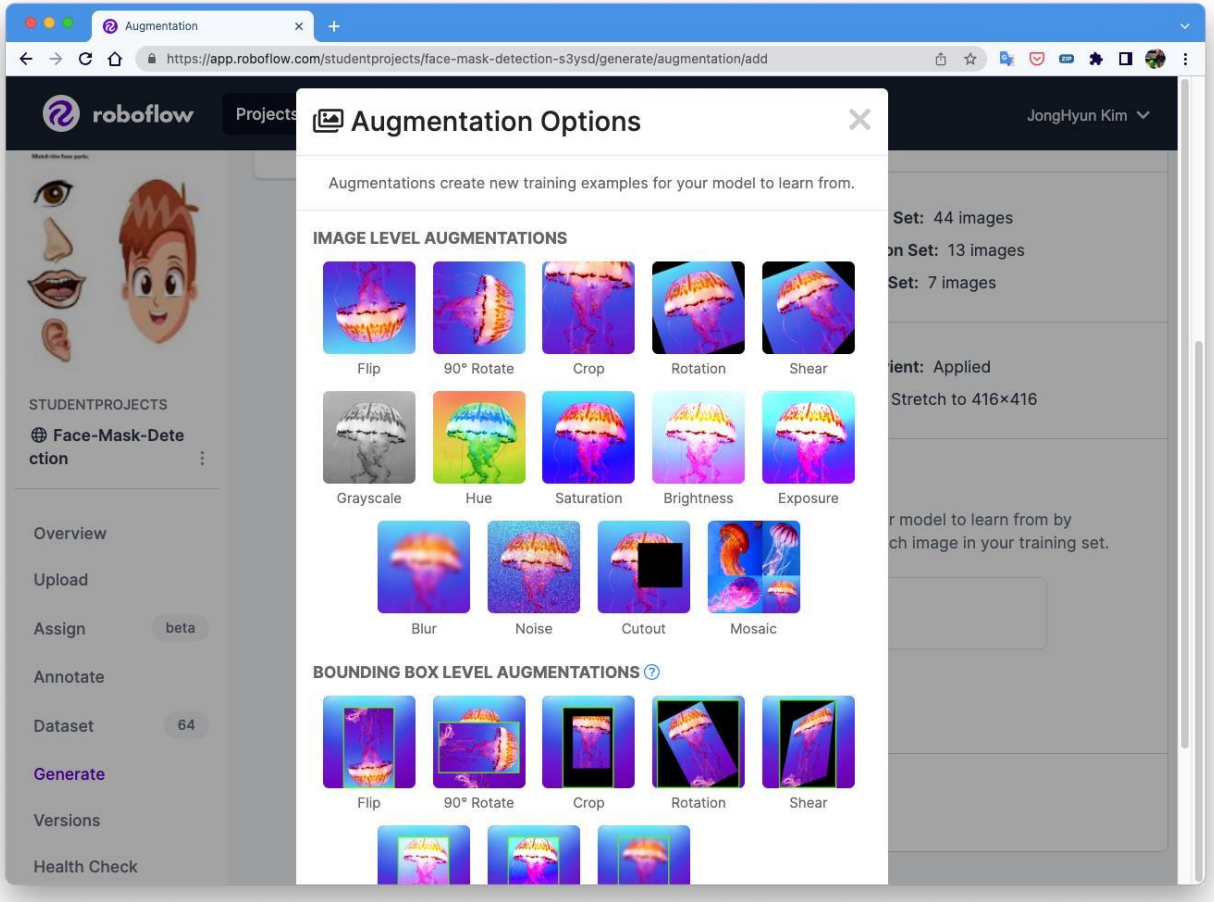
데이터 셋 생성


Face-Mask-Detection > Image
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/images/?split=train
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
Images
+ Add Images
Generate New Version
Training Set 44
Validation Set 13
Testing Set 7
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign beta
Annotate
Dataset 64
Generate
Versions
Health Check

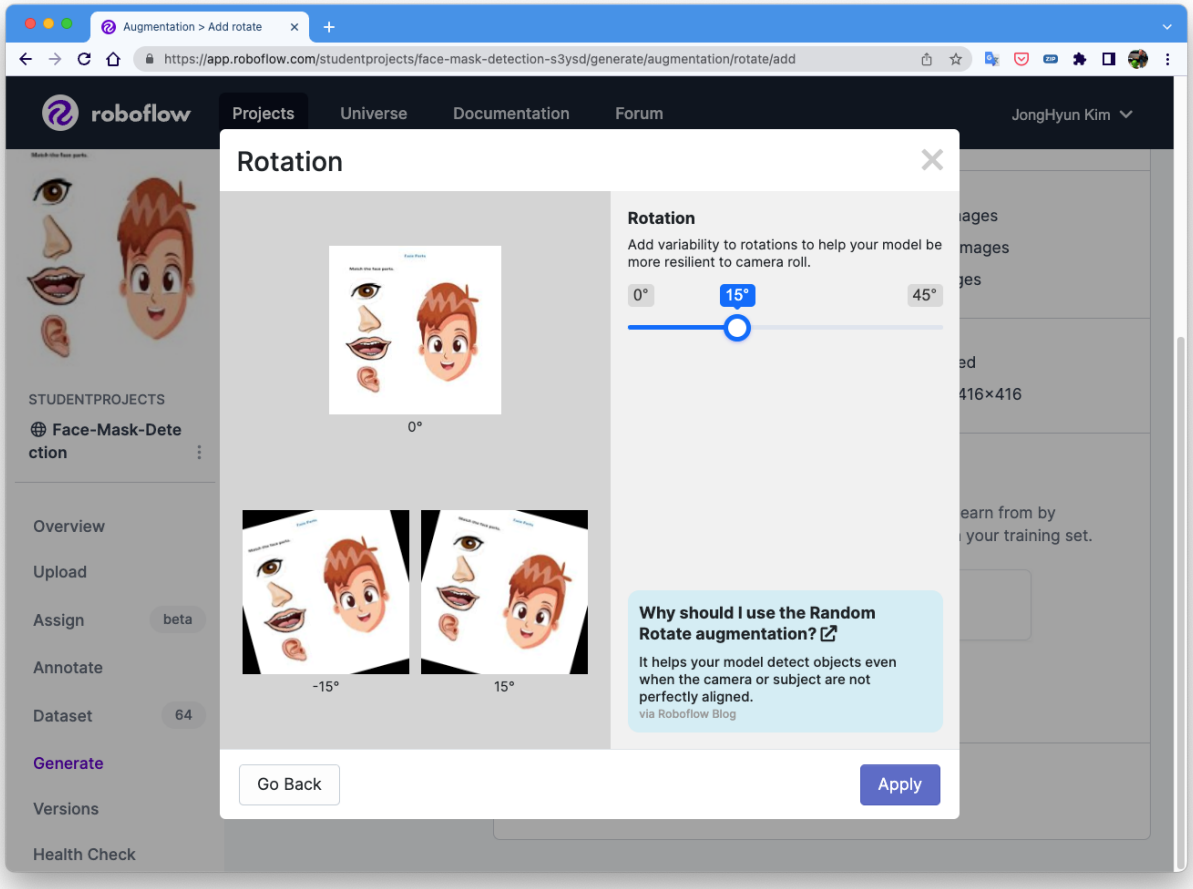

Face-Mask-Detection > Gener
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/generate/preprocessing
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
New Version
VERSIONS
To train a model, you must first generate a new version of your dataset.
Choose your dataset settings to get started.
Generating New Version
Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.
Source Images
Images: 64
Classes: 2
Unannotated: 0
Train/Test Split
Training Set: 44 images
Validation Set: 13 images
Testing Set: 7 images
3
Preprocessing
Decrease training time and increase performance by applying image transformations to all images in this dataset.
Auto-Orient
Edit
Resize
Stretch to 416×416
Edit
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign beta
Annotate
Dataset 64
Generate
Versions
Health Check

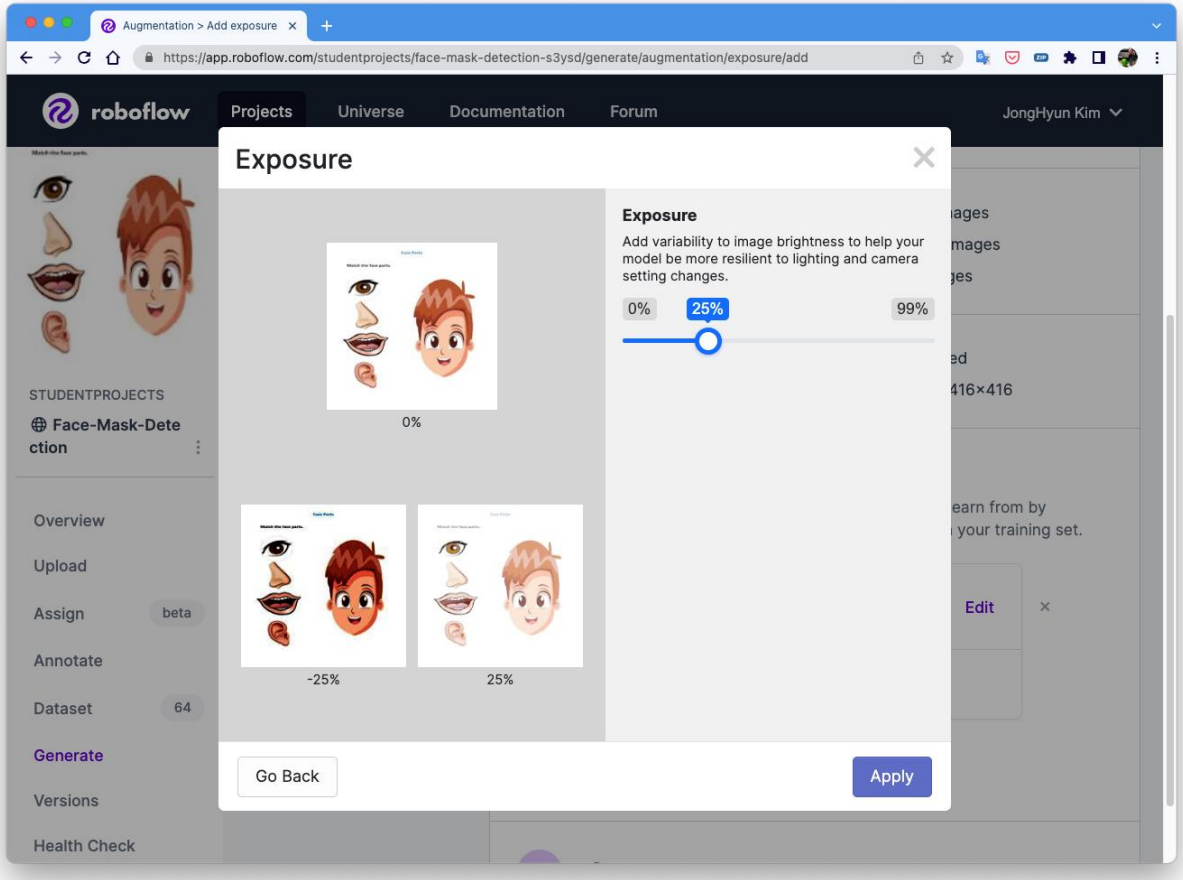
데이터 전처리, 증강


Preprocessing
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/generate
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign
beta
Annotate
Dataset
64
Generate
Versions
Health Check
3
Preprocessing
Decrease training time and increase performance by applying image transformations to all images in this dataset.
Auto-Orient
Edit
Resize
Stretch to 416×416
Edit
Add Preprocessing Step
Continue
4
Augmentation
5
Generate

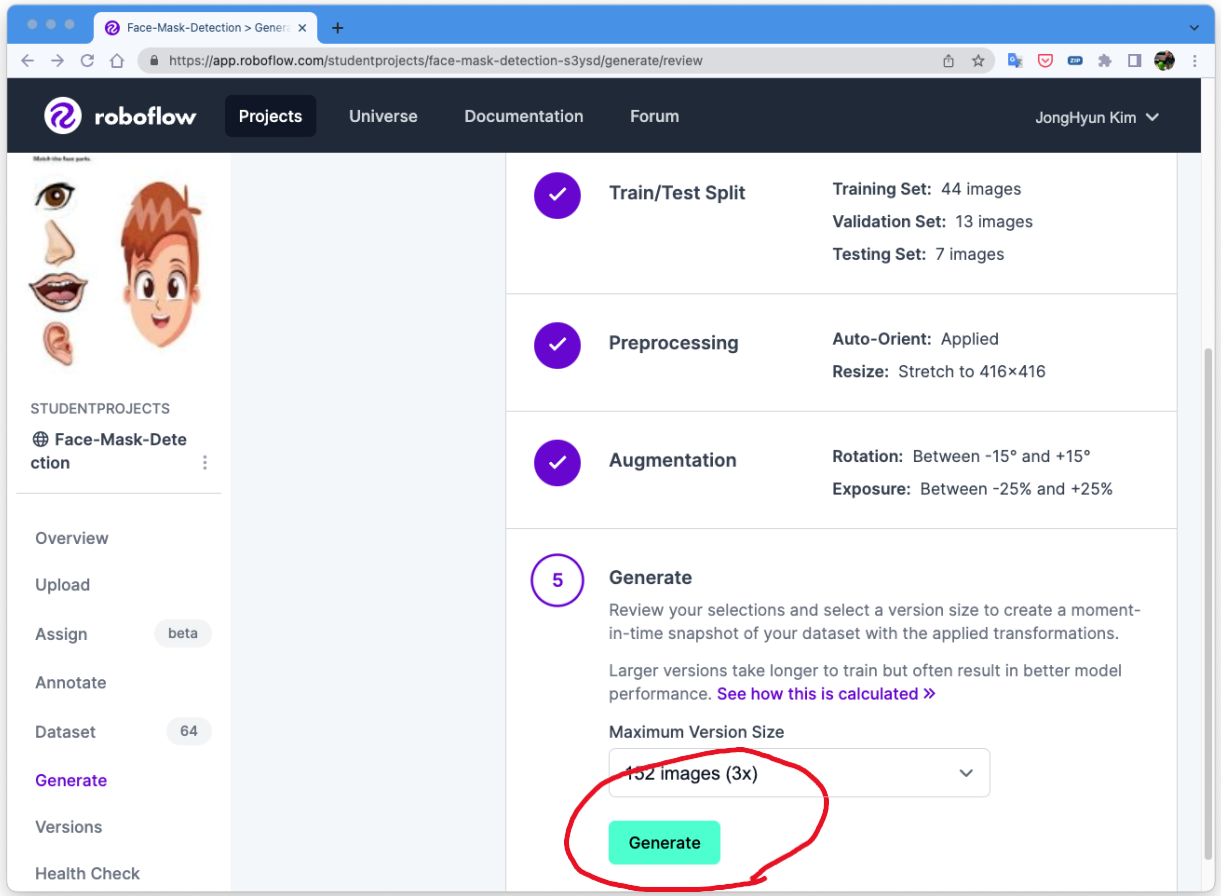

Augmentation
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/generate/augmentation/add
roboflow
JongHyun Kim
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign
beta
Annotate
Dataset
64
Generate
Versions
Health Check
Augmentation Options
Augmentations create new training examples for your model to learn from.
IMAGE LEVEL AUGMENTATIONS
Flip
90° Rotate
Crop
Rotation
Shear
Grayscale
Hue
Saturation
Brightness
Exposure
Blur
Noise
Cutout
Mosaic
BOUNDING BOX LEVEL AUGMENTATIONS
Flip
90° Rotate
Crop
Rotation
Shear
Set: 44 images
Set: 13 images
Set: 7 images
Applied
Stretch to 416×416

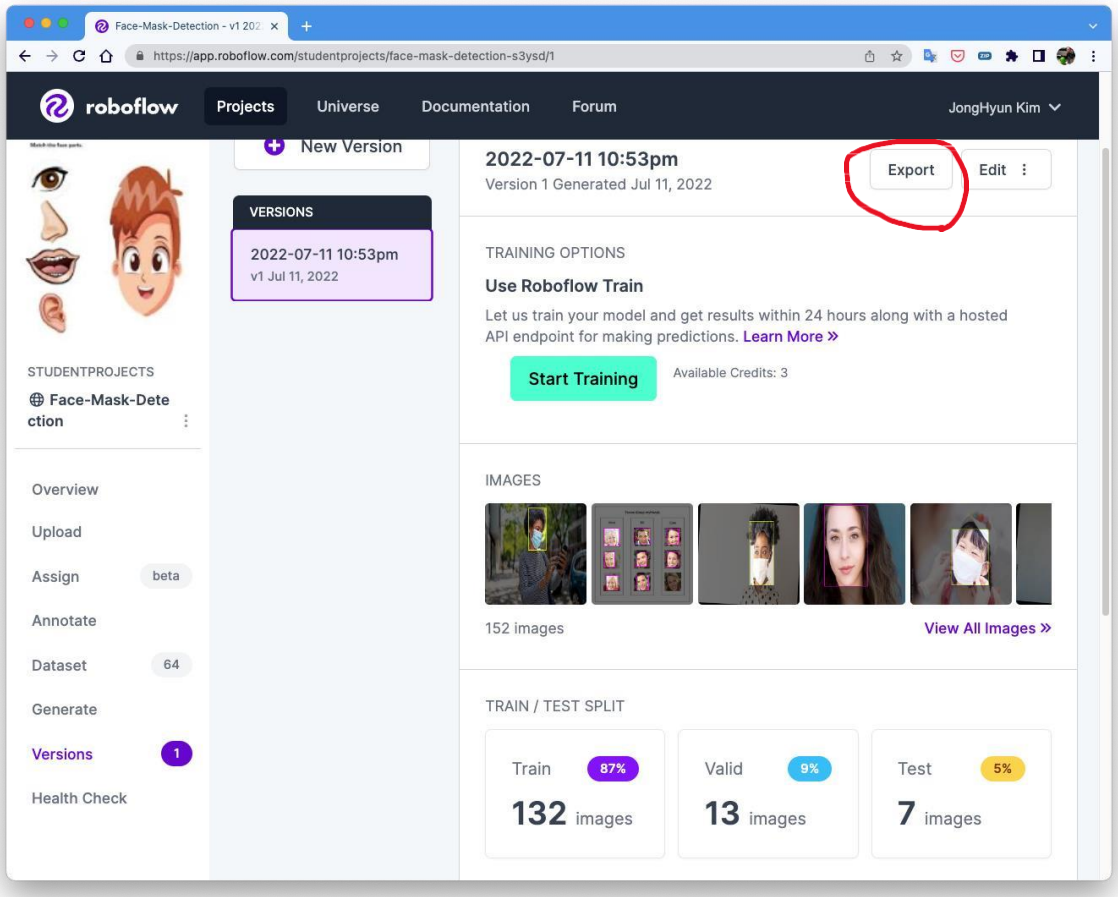
데이터 전처리, 증강


Augmentation > Add rotate
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/generate/augmentation/rotate/add
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
Rotation
0°
-15°
15°
Rotation
Add variability to rotations to help your model be more resilient to camera roll.
0°
15°
45°
Why should I use the Random Rotate augmentation?
It helps your model detect objects even when the camera or subject are not perfectly aligned.
via Roboflow Blog
Go Back
Apply
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign
beta
Annotate
Dataset
64
Generate
Versions
Health Check

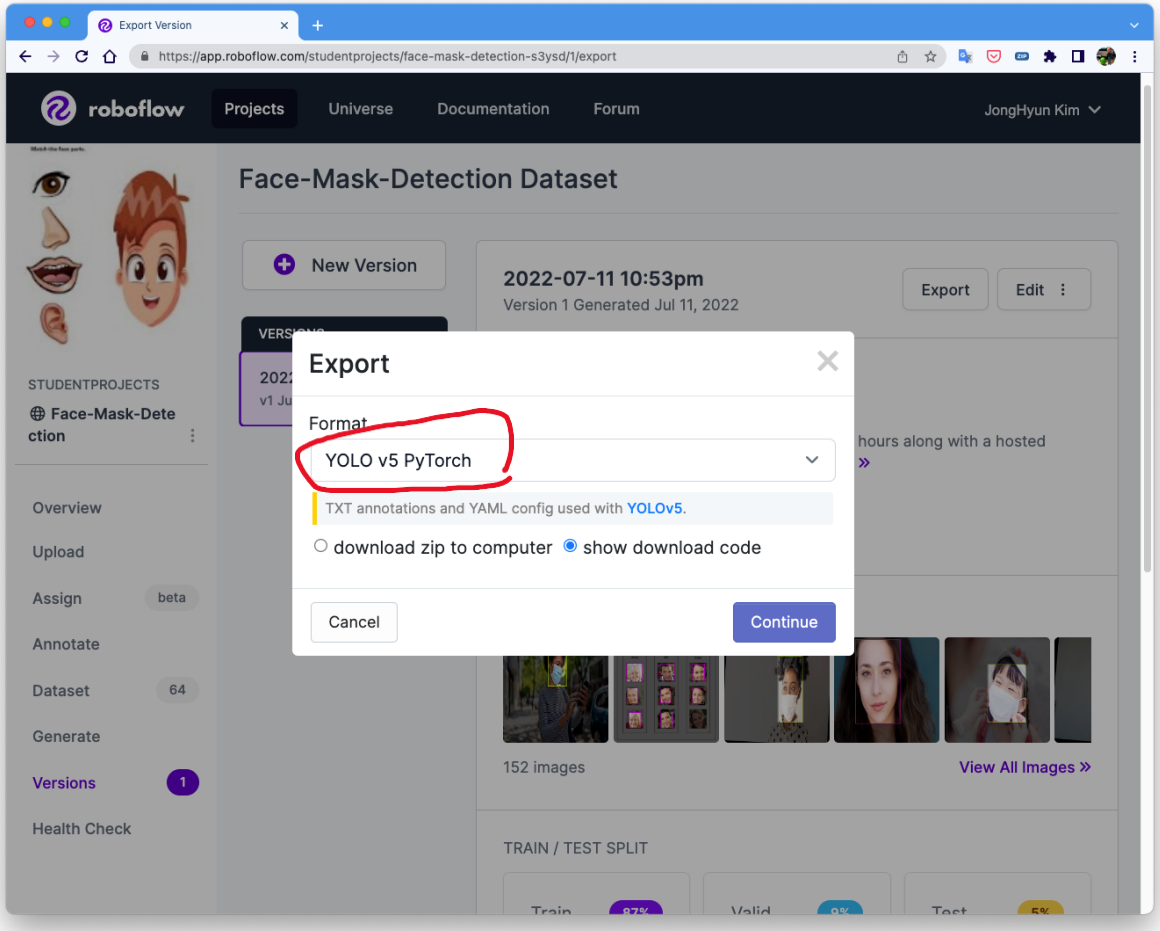

Augmentation > Add exposure
https://app.roboflow.com/studentprojects/face-mask-detection-s3ysd/generate/augmentation/exposure/add
roboflow
Projects
Universe
Documentation
Forum
JongHyun Kim
Exposure
0%
-25%
25%
Exposure
Add variability to image brightness to help your model be more resilient to lighting and camera setting changes.
0%
25%
99%
Go Back
Apply
STUDENTPROJECTS
Face-Mask-Detection
Overview
Upload
Assign
beta
Annotate
Dataset
64
Generate
Versions
Health Check
Edit

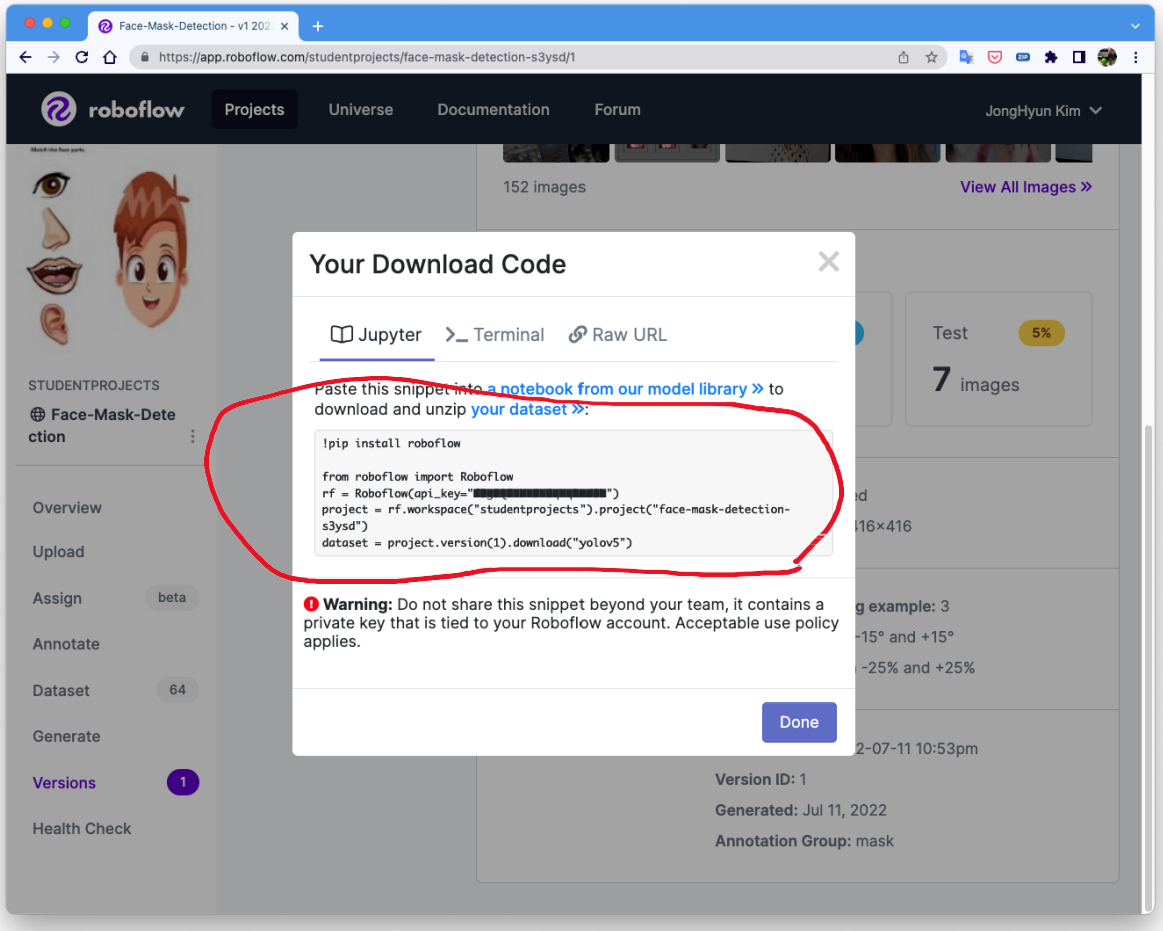
데이터 셋 배포 : Export





데이터 셋 배포 : Export

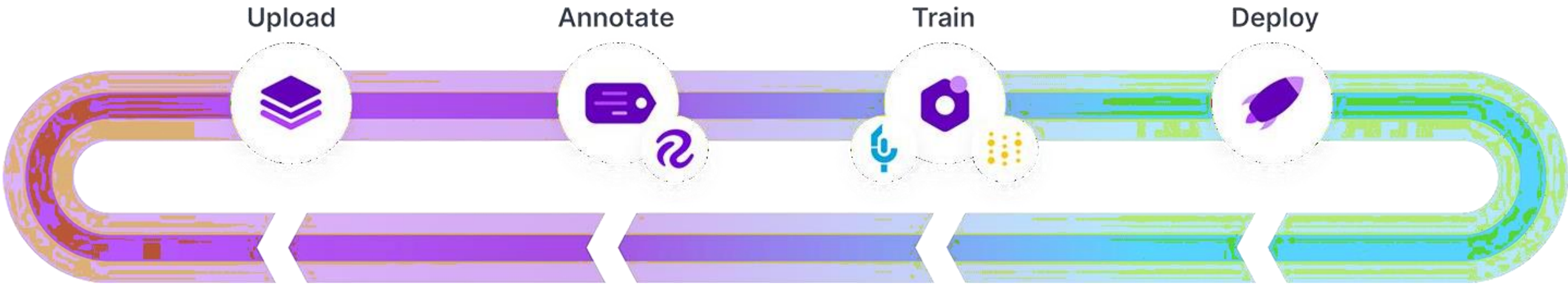




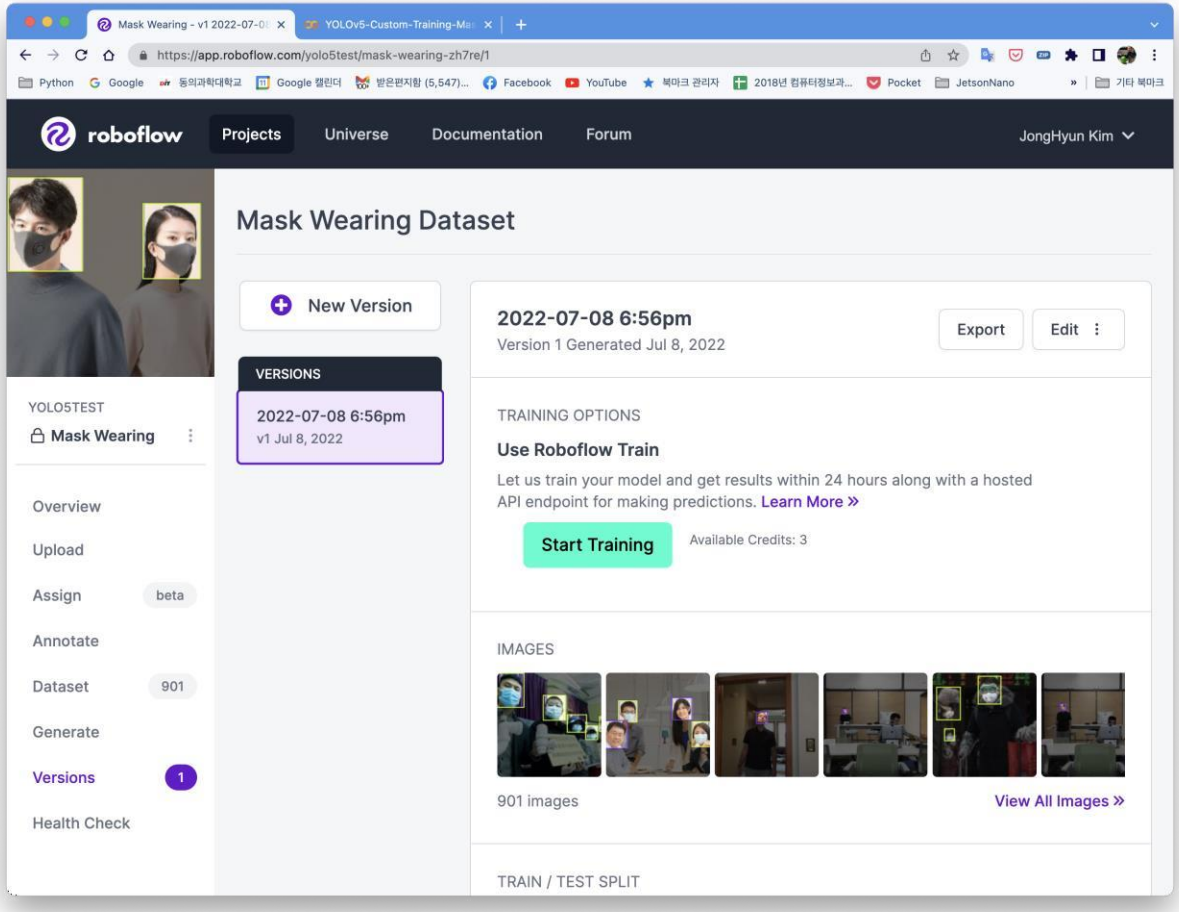
커스텀 데이터 학습 : Custom Data Training

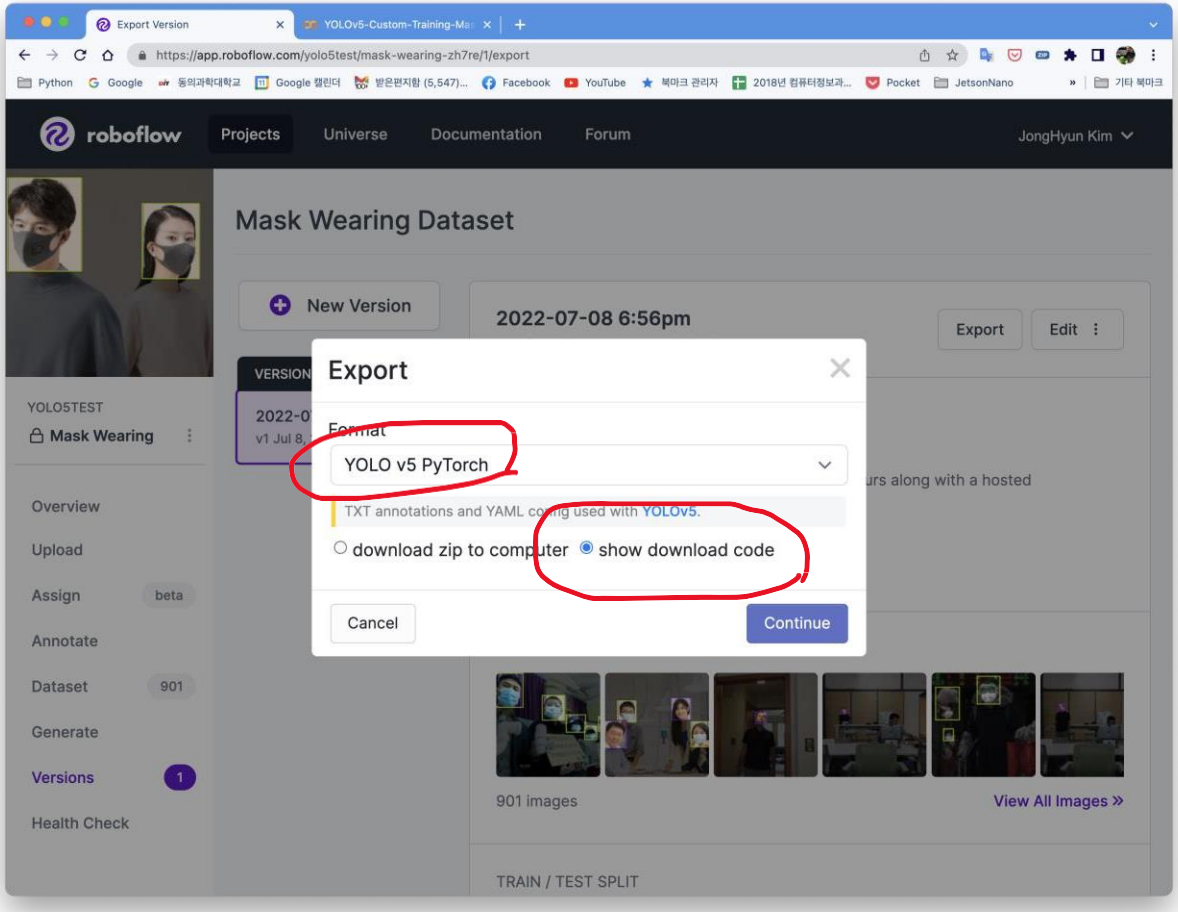
- https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

- Open in Colab
  - https://colab.research.google.com/github/roboflow-ai/yolov5-custom-
  - training-tutorial/blob/main/yolov5-custom-training.ipynb

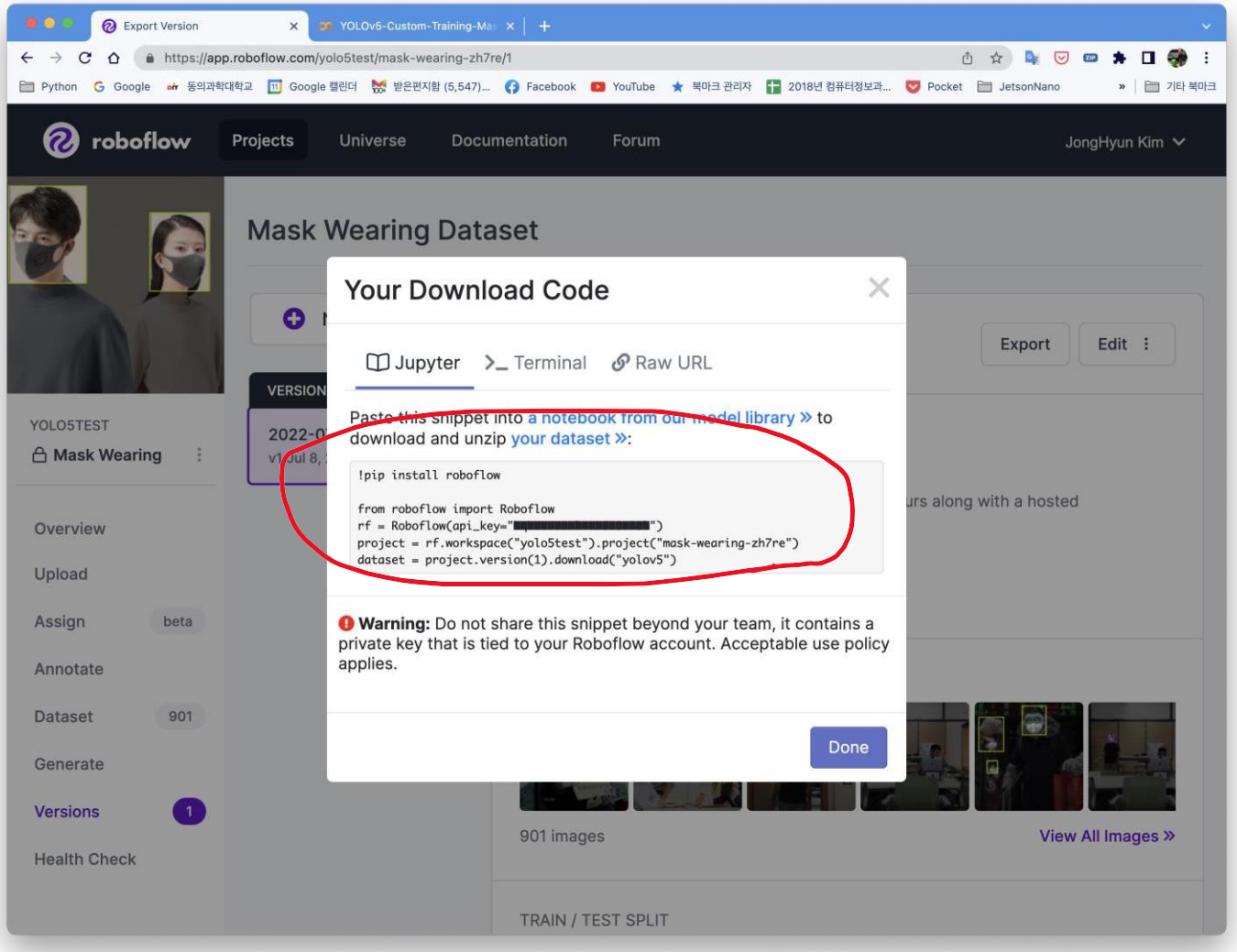


roboflow 데이터 셋 import





roboflow 데이터 셋 import



데이터 학습

```
[ ]  1 from roboflow import Roboflow
     2 rf = Roboflow(api_key="[redacted]")
     3 project = rf.workspace("yolo5test").project("mask-wearing-zh7re")
     4 dataset = project.version(1).download("yolov5")
```

```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in /content/datasets/Mask-Wearing-1 to yolov5pytorch: 100% [135938827 / 135938
Extracting Dataset Version Zip to /content/datasets/Mask-Wearing-1 in yolov5pytorch:: 100%|■■■■■■■■■■| 181
```

```
[ ]  1 !python train.py --img 416 --batch 16 --epochs 100 --data /content/datasets/Mask-Wearing-1/data.yaml --weights yolov5s.pt
```

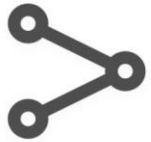
```
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)
```

모델 크기, 속도

Nano

YOLOv5n

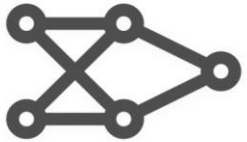
4 MB_{FP16}
6.3 ms_{V100}
28.4 mAP_{COCO}



Small

YOLOv5s

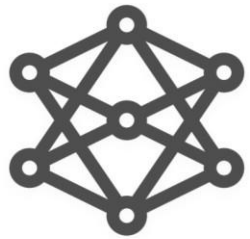
14 MB_{FP16}
6.4 ms_{V100}
37.2 mAP_{COCO}



Medium

YOLOv5m

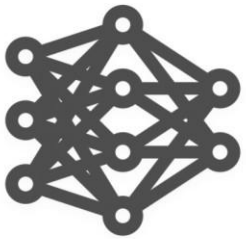
41 MB_{FP16}
8.2 ms_{V100}
45.2 mAP_{COCO}



Large

YOLOv5l

89 MB_{FP16}
10.1 ms_{V100}
48.8 mAP_{COCO}

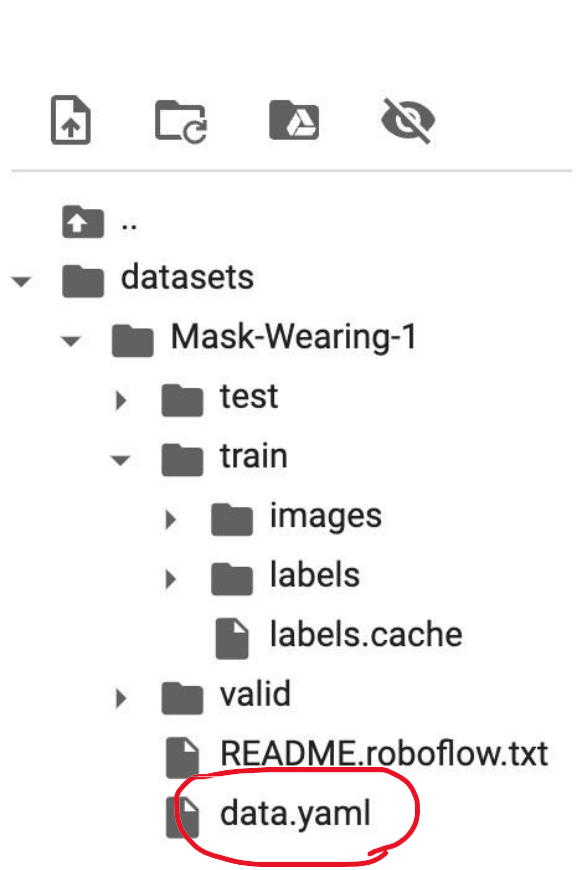


XLarge

YOLOv5x

166 MB_{FP16}
12.1 ms_{V100}
50.7 mAP_{COCO}

data.yaml



노트북 data.yaml

```
1 names:
2 - mask
3 - no_mask
4 nc: 2
5 train: /content/datasets/Mask-Wearing-1/train/images
6 val: /content/datasets/Mask-Wearing-1/valid/images
7
```

ultralytics/yolov5: YOLOv5 🚀 | YOLOv5-Custom-Training-Mas

https://colab.research.google.com/drive/1UsOIV2Ys9V2t3NUWumPqSOHfqOXIQcT6?usp=drive_fs#scrollTo=ZbUn4_b9GCKO

YOLOv5-Custom-Training-Mask-detection.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트

RAM 디스크 수정 가능

노트북 data.yaml

파일

- data
- models
- runs
  - train
    - exp
      - weights
        - best.pt
        - last.pt
      - F1_curve.png
      - PR_curve.png
      - P_curve.png
      - R_curve.png
      - confusion_matrix.png
      - events.out.tfevents.1657612581.aa4c24f77...
      - hyp.yaml
      - labels.jpg
      - labels_correlogram.jpg
      - opt.yaml
      - results.csv
      - results.png
      - train_batch0.jpg
      - train_batch1.jpg
      - train_batch2.jpg
      - val_batch0_labels.jpg
      - val_batch0_pred.jpg
      - val_batch1_labels.jpg
      - val_batch1_pred.jpg
      - val_batch2_labels.jpg
      - val_batch2_pred.jpg
- utils
- CONTRIBUTING.md
- LICENSE

디스크 39.02 GB 사용 가능

- **img:** define input image size
- **batch:** determine batch size
- **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
- **data:** Our dataset locaiton is saved in the `dataset.location`
- **weights:** specify a path to weights to start transfer learning from. Here we choose the generic COCO pretrained checkpoint.
- **cache:** cache images for faster training

[6] 13분

```
!python train.py --img 416 --batch 16 --epochs 100 --data /content/datasets/Mask-Wearing-1/
```

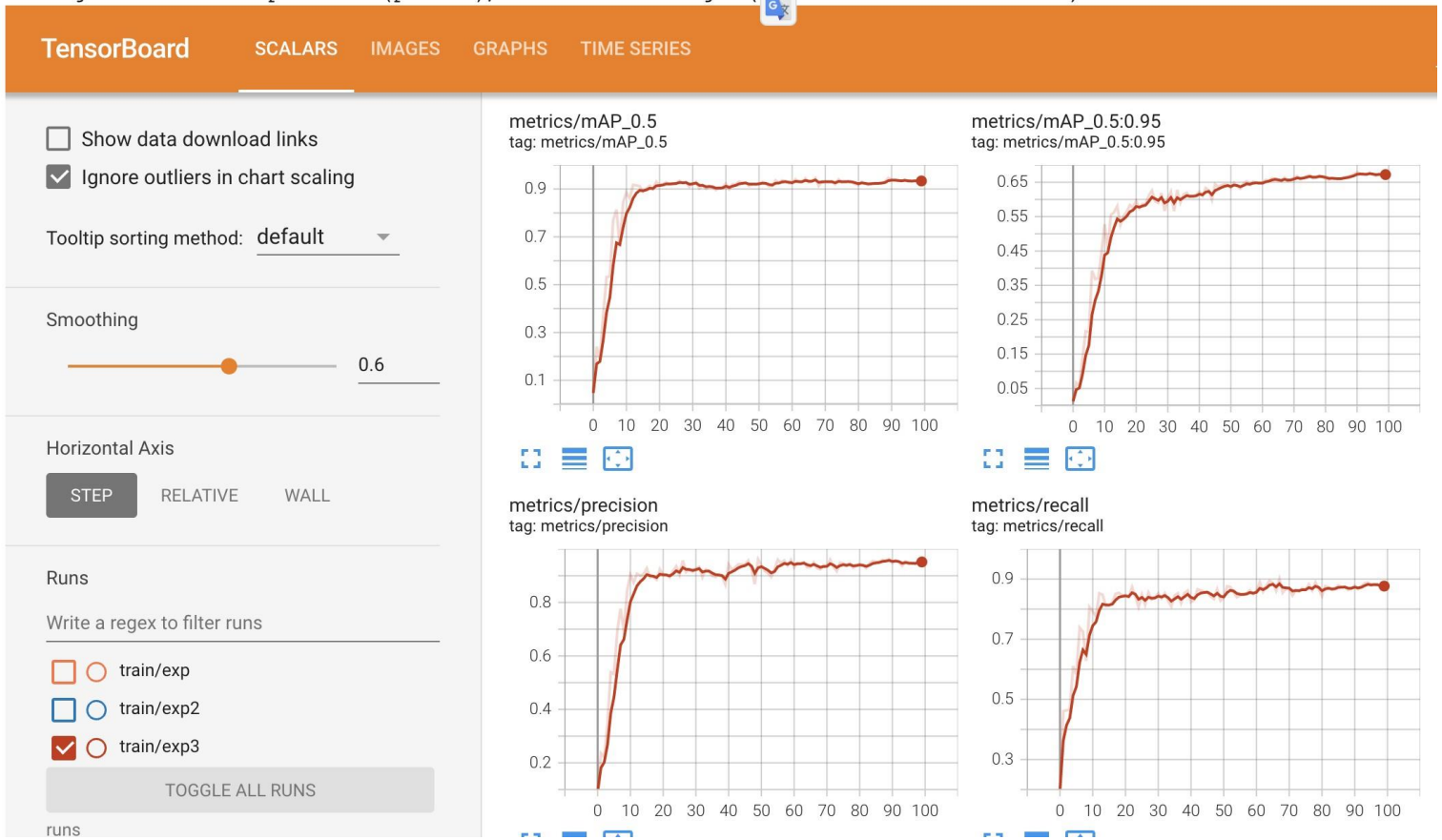
```
Logging results to runs/train/exp
Starting training for 100 epochs...

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      0/99     1.57G     0.103   0.03618   0.02813        61       416: 100% 41/41 [00:08<00:00,  4.60it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:01<00:00,  4.82it/s]
                  all        168        539      0.107      0.159     0.0522     0.0133

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      1/99     2.19G   0.07933   0.03735   0.02305       100       416: 100% 41/41 [00:06<00:00,  6.66it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:01<00:00,  5.45it/s]
                  all        168        539      0.395       0.35      0.294     0.0918

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      2/99     2.19G   0.07563   0.02815    0.0219        34       416: 100% 41/41 [00:06<00:00,  6.73it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:01<00:00,  5.90it/s]
                  all        168        539      0.373      0.391      0.375      0.161

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      3/99     2.19G   0.06802   0.02601   0.01726        63       416: 100% 41/41 [00:06<00:00,  6.78it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:00<00:00,  6.06it/s]
                  all        168        539      0.512      0.484      0.459      0.227

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      4/99     2.19G   0.05546   0.02543   0.01228        63       416: 100% 41/41 [00:06<00:00,  6.73it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:01<00:00,  5.98it/s]
                  all        168        539      0.599      0.633      0.645      0.328

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      5/99     2.19G   0.05031   0.02352  0.009031        58       416: 100% 41/41 [00:07<00:00,  5.82it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 6/6 [00:00<00:00,  6.26it/s]
                  all        168        539      0.731      0.692      0.757      0.385
```

0초 오후 5:15에 완료됨

Custom YOLOv5 Detector 성능 검증

```
[ ]  1 # Start tensorboard
     2 # Launch after you have started training
     3 # logs save in the folder "runs"
     4 %load_ext tensorboard
     5 %tensorboard --logdir runs
```

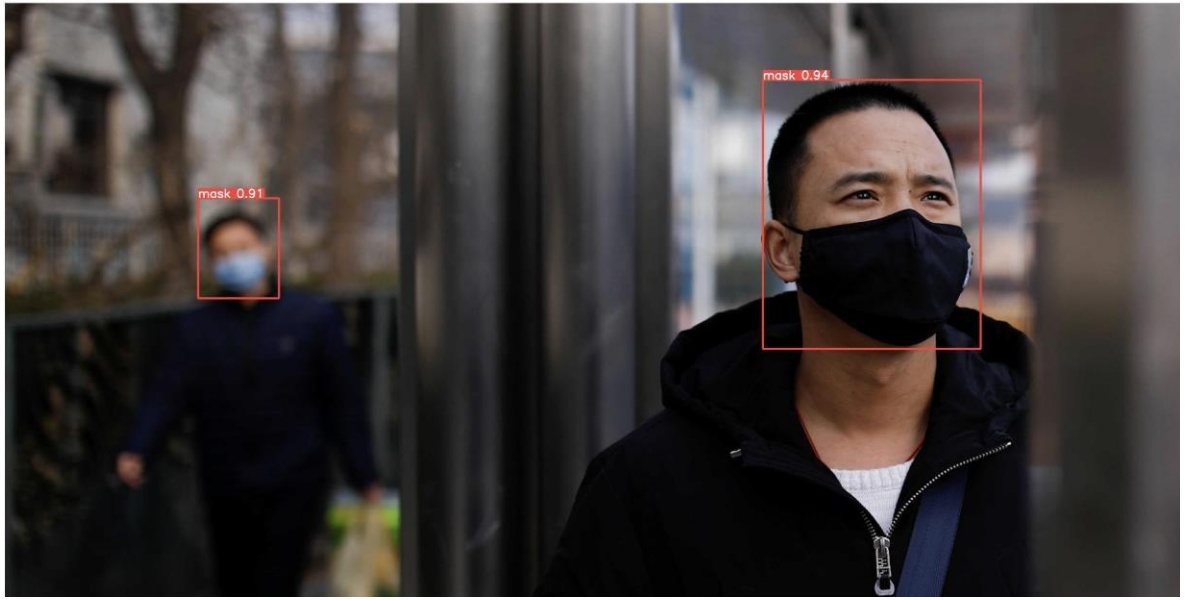
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 344), started 1:08:12 ago. (U... '!kill 344' to kill it.)
```



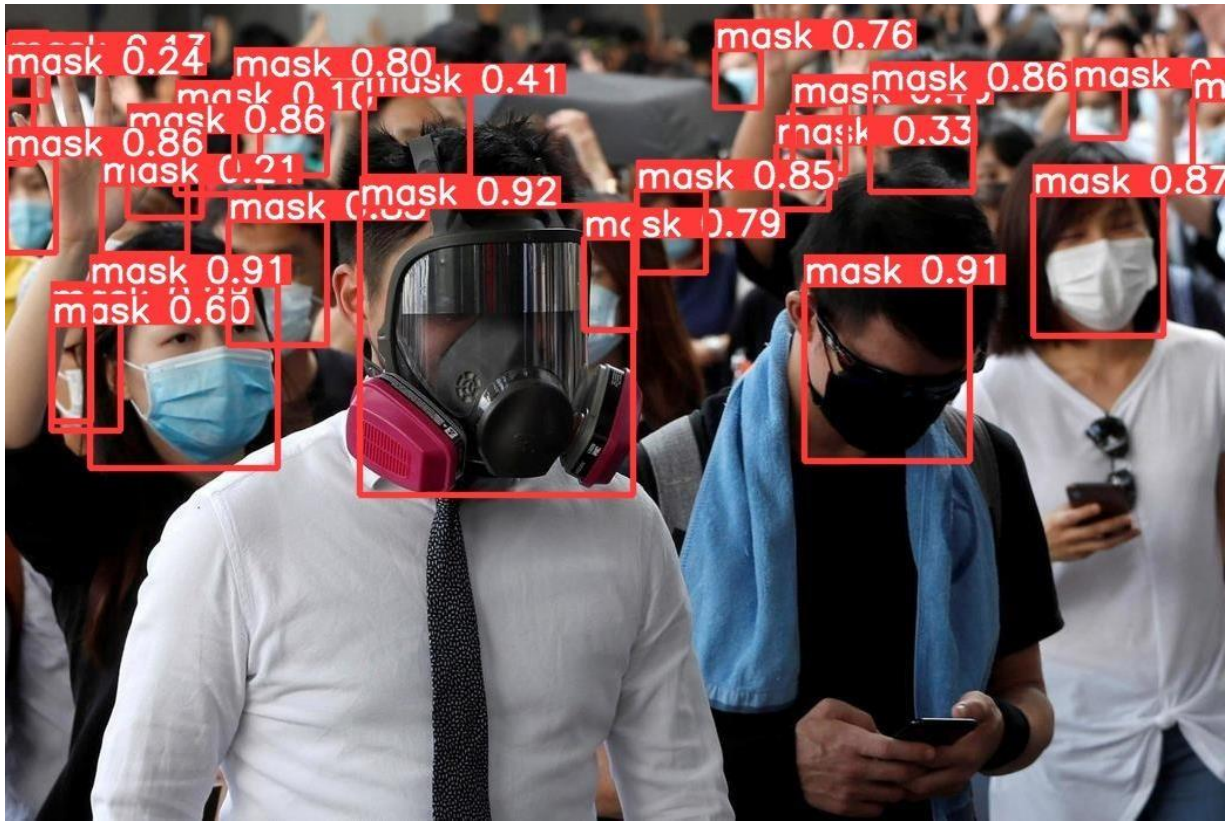
Trained Weights 추론

```
[ ]  1 !python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source /content/datasets/Mask-Wearing-1/test/images
```

```
ct: weights=['runs/train/exp3/weights/best.pt'], source=/content/datasets/Mask-Wearing-1/test/images, data=data/coco128.yaml, imgsz=[416, 416], conf_thres=0.1, iou_thres=0.45, max_det
v5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

ng layers...
l summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
e 1/83 /content/datasets/Mask-Wearing-1/test/images/0209-00176-076b1_jpg.rf.b5879cb0c604e946fc66e05808863ce1.jpg: 288x416 3 masks, 1 no_mask, Done. (0.014s)
e 2/83 /content/datasets/Mask-Wearing-1/test/images/0450908675_50159485_mutation-virus-chine-inquietude_jpg.rf.78f7187e33599ea007dfd483d8ffe775.jpg: 288x416 3 masks, 1 no_mask, Done.
e 3/83 /content/datasets/Mask-Wearing-1/test/images/0_Concern-In-China-As-Mystery-Virus-Spreads_jpg.rf.157ced3ca3a3d948313050b484add80e.jpg: 224x416 4 masks, Done. (0.015s)
e 4/83 /content/datasets/Mask-Wearing-1/test/images/0_Concern-In-China-As-Mystery-Virus-Spreads_jpg.rf.c7348a3dc8df48c724735c797e86bfe1.jpg: 224x416 4 masks, Done. (0.009s)
e 5/83 /content/datasets/Mask-Wearing-1/test/images/1224331650_g_400-w_g_jpg.rf.98ae7debb9e5ebc91b3ce4c56d2b2e30.jpg: 416x416 3 masks, Done. (0.011s)
e 6/83 /content/datasets/Mask-Wearing-1/test/images/1224331650_g_400-w_g_jpg.rf.9fe767c9d05d662400b9bd1b9422e3af.jpg: 416x416 3 masks, Done. (0.009s)
e 7/83 /content/datasets/Mask-Wearing-1/test/images/126202-untitled-design-13_jpg.rf.065fbc4190cc207a91a46b9fe13e7ea9.jpg: 256x416 5 masks, Done. (0.013s)
e 8/83 /content/datasets/Mask-Wearing-1/test/images/126202-untitled-design-13_jpg.rf.41757e0c41a5770e63d886b291d4d275.jpg: 256x416 5 masks, Done. (0.009s)
```



Trained Weights 추론



학습 모델 다운로드

```
[ ]  1 #export your model's weights for future use
     2 from google.colab import files
     3 files.download('/content/yolov5/runs/train/exp3/weights/best.pt')
```

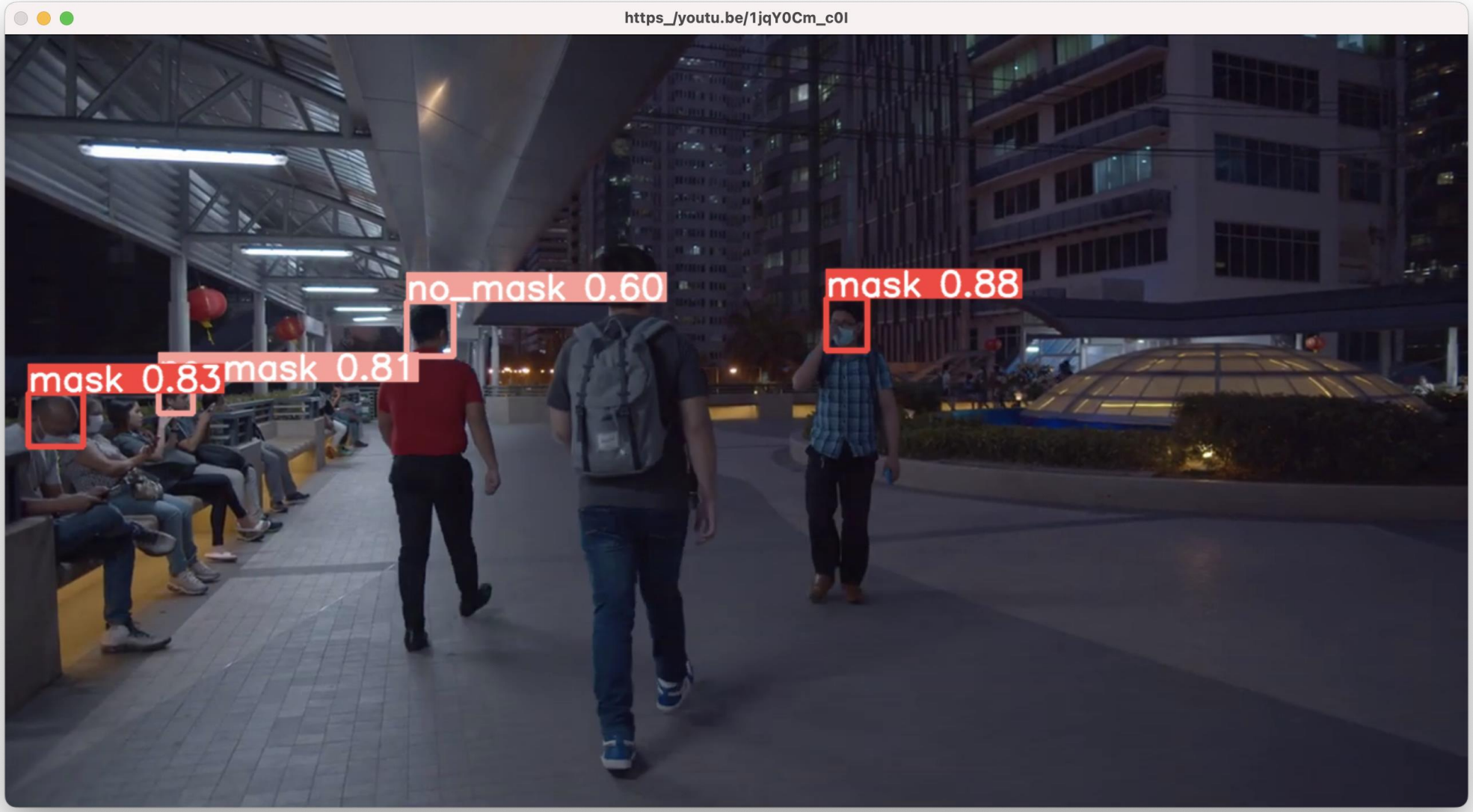
모델 사용하기

- yolov5를 clone 한 local PC 디렉토리에 모델(best.pt) 저장 및
- 이동
  - git clone https://github.com/ultralytics/yolov5 # clone
  - cd yolov5
  - pip install -r requirements.txt # install
- detection 실행

```
% python detect.py --source 0  # webcam
                            img.jpg  # image
                            vid.mp4  # video
                            path/  # directory
                            path/*.jpg  # glob
                            'https://youtu.be/Zgi9g1ksQHc'  # YouTube
                            'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP stream
```

실시간 객체(얼굴 마스크 착용) 탐지

실시간 객체(얼굴 마스크 착용) 탐지



미니 프로젝트

- 객체 탐지 주제 및 아이디어 도출
  - 자신의 객체 탐지 주제를 생각해 보자
  - 자신의 전공 분야 관련이면 좋음
- 마스크 착용 탐지
- 빵집 알파고 : 제과점 빵 종류 탐지
- 썩은 과일, 농작물 등 탐지
- 도둑 탐지
- 낙상 사고 탐지
- 댐 및 건물 파손 탐지
- 산불 탐지
- 의료 분야(CT 데이터) 탐지 …

참고

- https://github.com/ultralytics/yolov5
- https://roboflow.com/
- https://www.v7labs.com/blog/object-detection-guide

Face Recognition을 이용한 AI 프로젝트

- dLib
  - http://dlib.net/
- face_recognition
  - https://face-recognition.readthedocs.io/en/latest/readme.html
  - https://github.com/ageitgey/face_recognition
  - https://github.com/SKT-FLY-AI-2022/face_recognition_examples

face_recognition

- 라이브러리 설치(PC)

```
% pip install opencv-python
% pip install face_recognition
```

- 윈도우즈 10에 face_recognition 설치

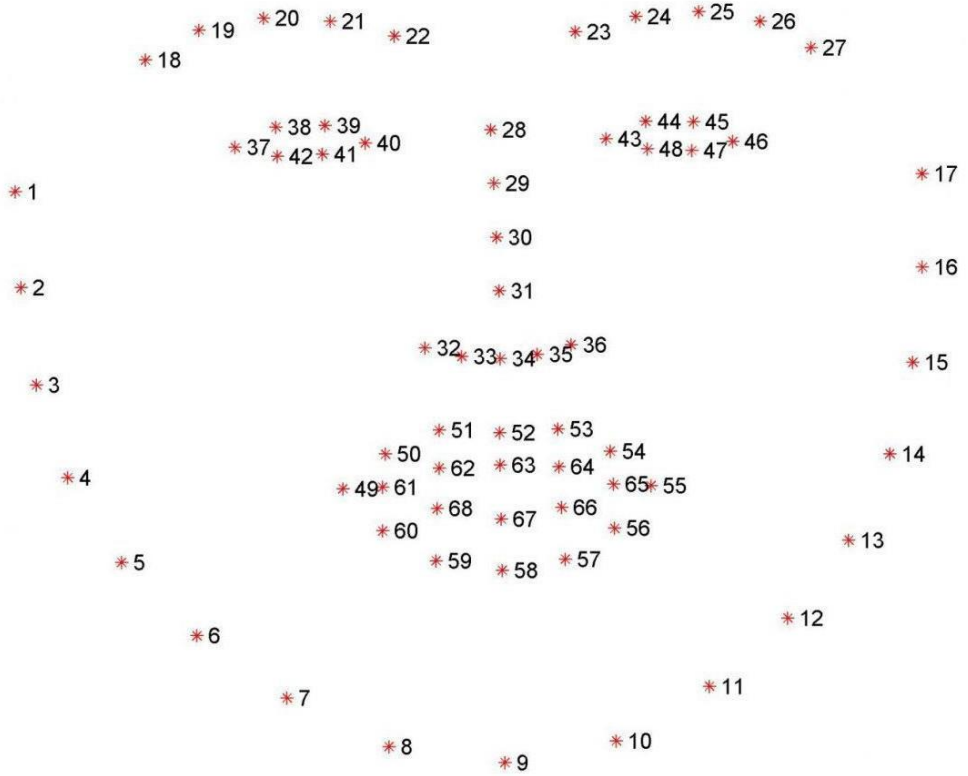
```
% pip install cmake
% conda install –c conda-forge dlib
```

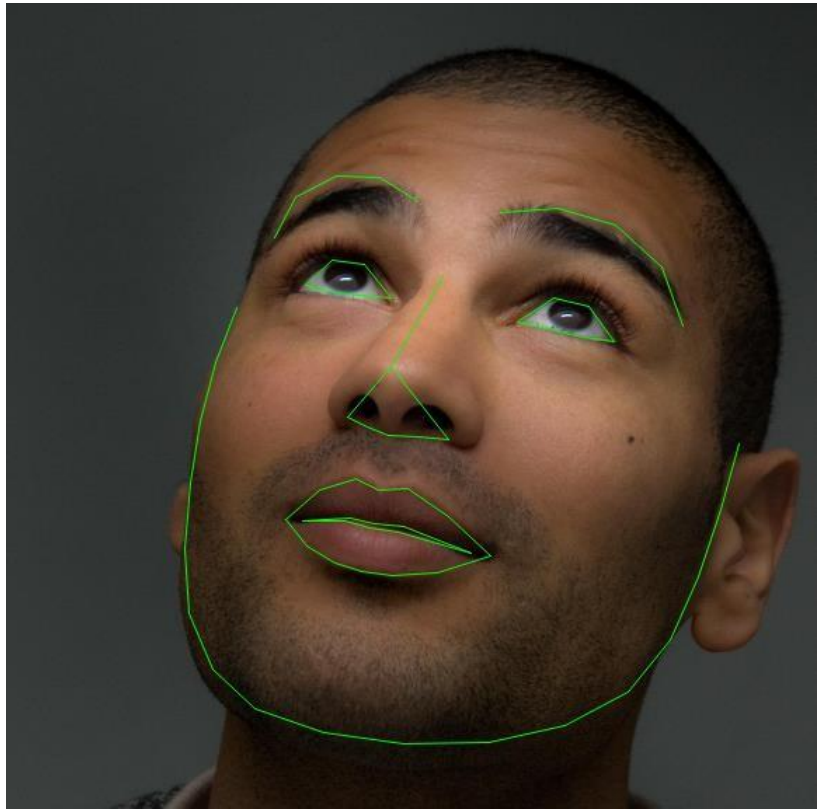
face_recognition API

- load_image_file()
- face_locations()
- batch_face_locations()
- face_landmarks()
- face_encodings()
- compare_faces()
- face_distance()

dLib facial landmark

- pre trained model
  - https://github.com/davisking/dlib-models
  - https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks.dat.bz2





face landmark 출력하기

```
import cv2
import face_recognition

image = cv2.imread('img/unknown/me.jpg')
results = face_recognition.face_landmarks(image)
face_location = face_recognition.face_locations(image)
print(results)

chin = results[0]['chin']
right_eyebrow = results[0]['right_eyebrow']
left_eyebrew = results[0]['left_eyebrow']
right_eye   = results[0]['right_eye']
left_eye     = results[0]['left_eye']
nose_tip = results[0]['nose_tip']
bottom_lip = results[0]['bottom_lip']
```

```
while True:
    for (top, right, bottom, left) in face_location:
    # Draw a box around the face
    cv2.rectangle(image, (left, top), (right, bottom), (0, 0, 255), 1)

    for chins in chin:
    # Draw a box around the face
    cv2.circle(image, chins, 5, (0,255,0), 1)
    for     r_eb in right_eyebrow:
    cv2.circle(image, r_eb, 5, (0,255,0), 1)

    for l_eb in left_eyebrow:
    cv2.circle(image, l_eb, 5,          (0,255,0), 1)

    for l_eb in left_eyebrow:
    cv2.circle(image, l_eb, 5, (0,255,0), 1)

    for r_e in right_eye:
                                5, (0,255,0), 1)
    cv2.circle(image, r_e,
    for l_e in left_eye:
    cv2.circle(image, l_e,          5, (0,255,0), 1)

    for l_e in nose_tip:
    cv2.circle(image, l_e,          5, (0,255,0), 1)

    for b_l in bottom_lip:
    cv2.circle(image, b_l,          5, (0,255,0), 1)

    cv2.imshow('img', image)
    cv2.waitKey(5)
```


img

얼굴 찾기

```
import face_recognition

image = face_recognition.load_image_file('./img/groups/team2.jpg')
face_locations = face_recognition.face_locations(image)

# Array of coords of each face
print(face_locations)

print(f'There are {len(face_locations)} people in this image')
```

```
[(89, 79, 132, 36), (94, 429, 137, 386), (70, 371, 113, 328), (65, 487, 108, 443), (74, 655, 118, 612), (84, 573, 127, 530),
(65, 213, 108, 170), (79, 295, 122, 252), (94, 157, 130, 121), (78, 121, 114, 85), (94, 245, 130, 209)]
There are 11 people in this image
```

https://github.com/SKT-FLY-AI-2022/face_recognition_examples

얼굴 비교하기

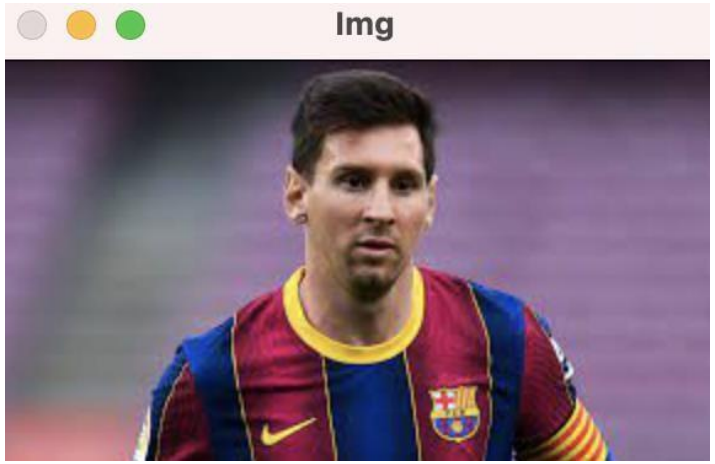
```
import face_recognition

image_of_bill = face_recognition.load_image_file('./img/known/Bill Gates.jpg')
bill_face_encoding = face_recognition.face_encodings(image_of_bill)[0]

unknown_image = face_recognition.load_image_file('./img/unknown/d-trump.jpg')

unknown_face_encoding = face_recognition.face_encodings(unknown_image)[0]

# Compare faces
results = face_recognition.compare_faces([bill_face_encoding], unknown_face_encoding, 0.6)

if results[0]:
print('This is Bill Gates')
else:
print('This is NOT Bill Gates')
```

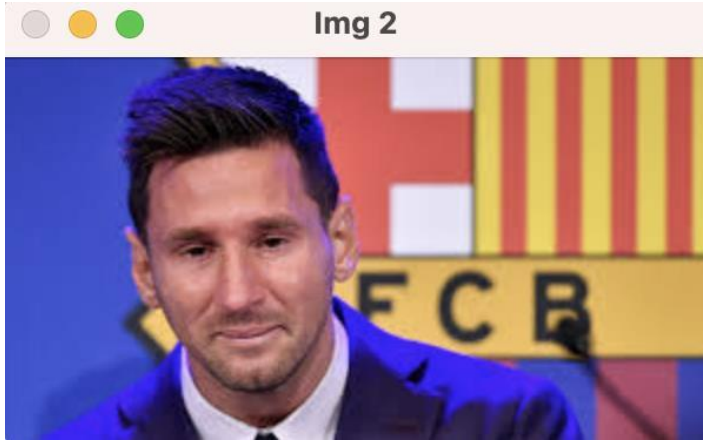
```
This is NOT Bill Gates
```

얼굴 비교하기

```python
import cv2
import face_recognition

img = cv2.imread("Messi1.jpg")
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_encoding = face_recognition.face_encodings(rgb_img)[0]

img2 = cv2.imread("images/Messi.jpg")
rgb_img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
img_encoding2 = face_recognition.face_encodings(rgb_img2)[0]

result = face_recognition.compare_faces([img_encoding], img_encoding2)

print("Result: ", result)

cv2.imshow("Img", img)

cv2.imshow("Img 2", img2)

cv2.waitKey(0)
```

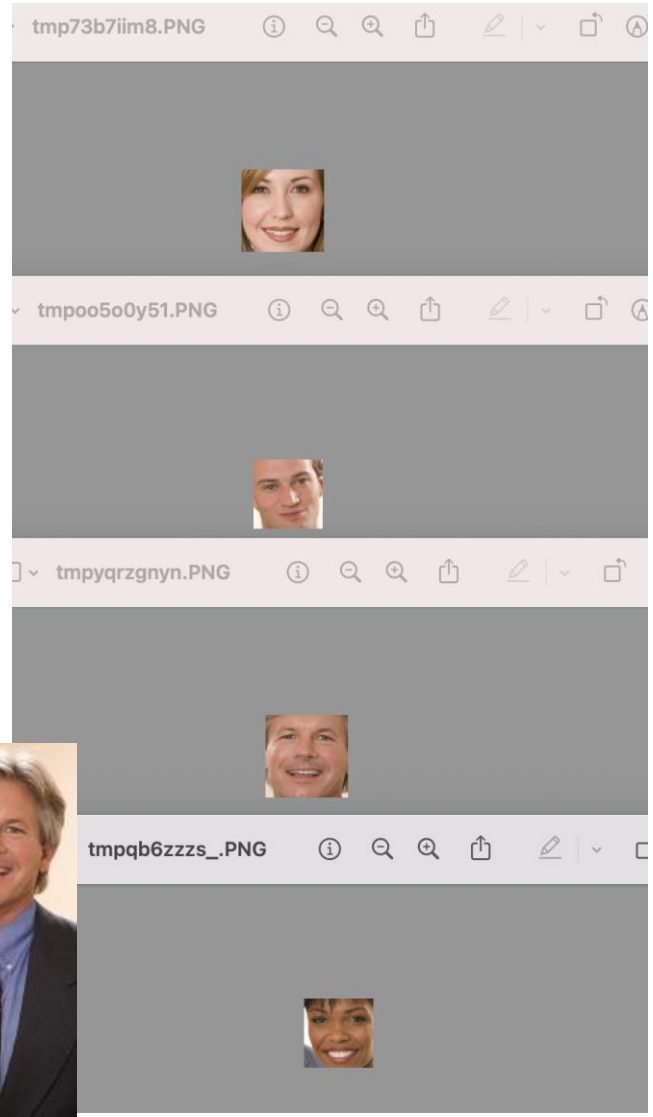
```
Result:  [True]
```





얼굴 가져오기

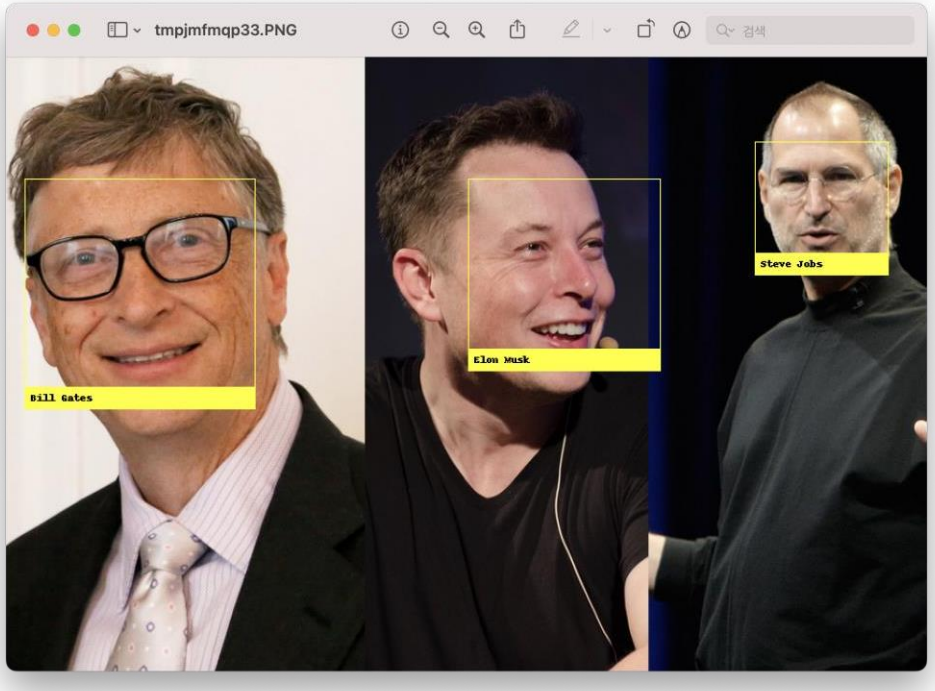
```
from PIL import Image
import face_recognition

image = face_recognition.load_image_file('./img/groups/team1.jpg')
face_locations = face_recognition.face_locations(image)

for face_location in face_locations:
    top, right, bottom, left = face_location

    face_image = image[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)
    pil_image.show()
    pil_image.save(f'{top}.jpg')
```



얼굴 인식하기

```
import face_recognition
from PIL import Image, ImageDraw

image_of_bill = face_recognition.load_image_file('./img/known/Bill Gates.jpg')

bill_face_encoding = face_recognition.face_encodings(image_of_bill)[0]

image_of_steve = face_recognition.load_image_file('./img/known/Steve Jobs.jpg')

steve_face_encoding = face_recognition.face_encodings(image_of_steve)[0]

image_of_elon = face_recognition.load_image_file('./img/known/Elon Musk.jpg')
elon_face_encoding = face_recognition.face_encodings(image_of_elon)[0]

# Create arrays of encodings and names
known_face_encodings = [
  bill_face_encoding,
  steve_face_encoding,
  elon_face_encoding
]
known_face_names = [
"Bill Gates",
"Steve Jobs",
"Elon Musk"
]
```

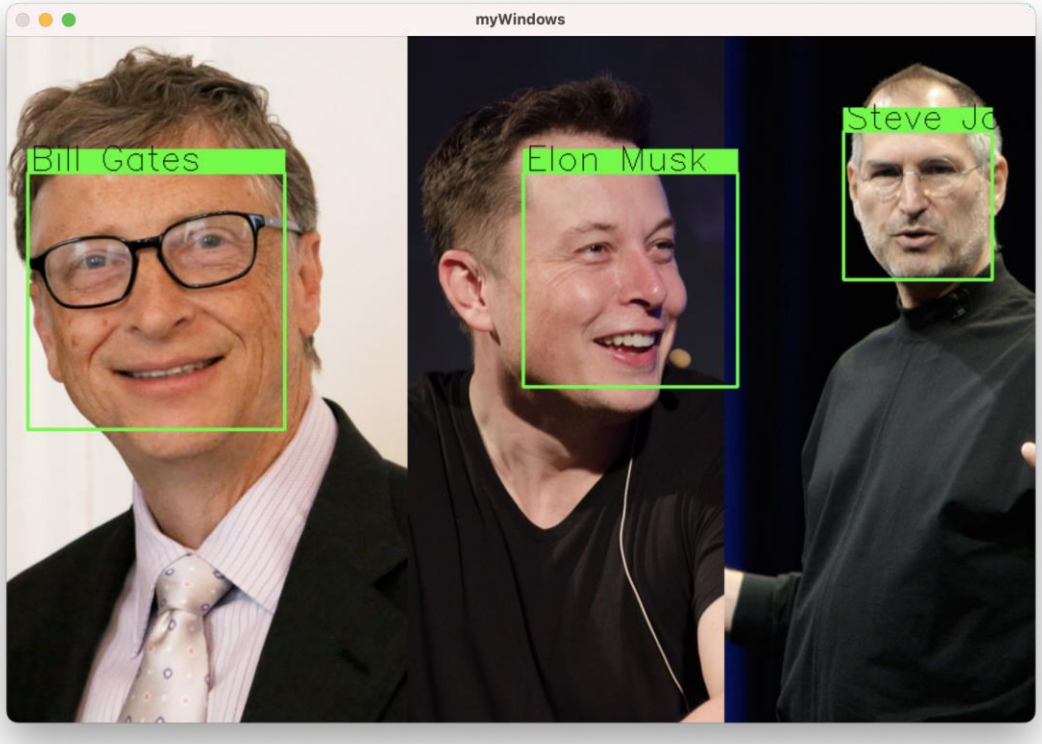
Pillow(PIL) 구현



```
# Load test image to find faces in
test_image = face_recognition.load_image_file('./img/groups/bill-steve-elon.jpg')

# Find faces in test image
face_locations = face_recognition.face_locations(test_image)
face_encodings = face_recognition.face_encodings(test_image, face_locations)

# Convert to PIL format
pil_image = Image.fromarray(test_image)

# Create a ImageDraw instance
draw = ImageDraw.Draw(pil_image)

# Loop through faces in test image
for(top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
matches = face_recognition.compare_faces(known_face_encodings, face_encoding)

name = "Unknown Person"

# If match
if True in matches:
   first_match_index = matches.index(True)
   name = known_face_names[first_match_index]

# Draw box
draw.rectangle(((left, top), (right, bottom)), outline=(255,255,0))

# Draw label
text_width, text_height = draw.textsize(name)
draw.rectangle(((left,bottom - text_height - 10), (right, bottom)), fill=(255,255,0), outline=(255,255,0))
draw.text((left + 6, bottom - text_height - 5), name, fill=(0,0,0))

del draw

# Display image
pil_image.show()
```

```
import face_recognition
import cv2

image_of_bill = face_recognition.load_image_file('./img/known/Bill Gates.jpg')
bill_face_encoding = face_recognition.face_encodings(image_of_bill)[0]

image_of_steve = face_recognition.load_image_file('./img/known/Steve Jobs.jpg')
steve_face_encoding = face_recognition.face_encodings(image_of_steve)[0]

image_of_elon = face_recognition.load_image_file('./img/known/Elon Musk.jpg')
elon_face_encoding = face_recognition.face_encodings(image_of_elon)[0]

# Create arrays of encodings and names
known_face_encodings = [
        bill_face_encoding,
        steve_face_encoding,
        elon_face_encoding
]

known_face_names = [ "Bill Gates", "Steve Jobs", "Elon Musk "  ]

# Load test image to find faces in
test_image = face_recognition.load_image_file('./img/groups/bill-steve-elon.jpg')

# Find faces in test image
test_face_locations = face_recognition.face_locations(test_image)
test_face_encodings = face_recognition.face_encodings(test_image, test_face_locations)

test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)

for(top, right, bottom, left), test_face_encoding in zip(test_face_locations, test_face_encodings):
name = "Unknown Person"
        matches = face_recognition.compare_faces(known_face_encodings, test_face_encoding, 0.6)
        if True in matches:
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]
        cv2.rectangle(test_image, (left, top), (right, bottom), (0,255,0), 2)
        cv2.rectangle(test_image, (left, top - 20), (right, top), (0, 255, 0), cv2.FILLED)
        cv2.putText(test_image, name, (left+2, top-2), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)

cv2.imshow('myWindows', test_image)
cv2.moveWindow('myWindows', 0, 0)
if      cv2.waitKey(0) == ord('q'):
        cv2.destroyAllWindows()
```

Opencv 구현

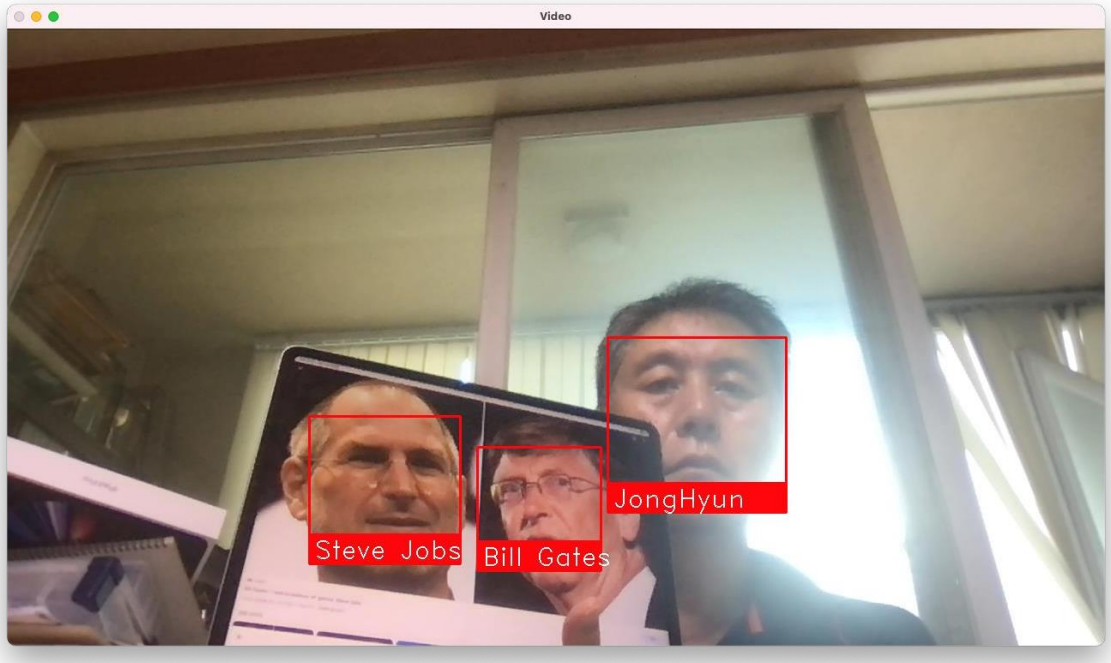


실시간 얼굴 인식하기(웹캠)

```
import face_recognition
import cv2
import numpy as np

video_capture = cv2.VideoCapture(1)

# Load a sample picture and learn how to recognize it.
bill_image = face_recognition.load_image_file("img/known/BillGates.jpg")
bill_face_encoding = face_recognition.face_encodings(bill_image)[0]

steve_image = face_recognition.load_image_file("img/known/SteveJobs.jpg")
steve_face_encoding = face_recognition.face_encodings(steve_image)[0]

jhkim_image = face_recognition.load_image_file("img/known/jhkim.jpg")
jhkim_face_encoding = face_recognition.face_encodings(jhkim_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    bill_face_encoding,
    steve_face_encoding,
    jhkim_face_encoding
]
known_face_names = [
"Bill Gates",
    "Steve Jobs",
     "JongHyun"
]
```

```
# Initialize some variables
face_locations = []
face_encodings    = []
face_names      = []
process_this_frame  = True

while True:
        # Grab a single frame of video
        ret, frame = video_capture.read()
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        rgb_small_frame = small_frame[:, :, ::-1]
        # Find all the faces and face encodings in the current frame of
        video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
        face_locations)
        face_names = []
        for face_encoding in face_encodings:
                # See if the face is a match for the known face(s)
                matches =
                face_recognition.compare_faces(known_face_encodings,
                face_encoding)
                name = "Unknown"
                face_distances =
                face_recognition.face_distance(known_face_encodings,
                face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                name = known_face_names[best_match_index]
                face_names.append(name)
```

```
        # Display the results
        for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

        # Display the resulting image

        cv2.imshow('Video', frame)

        # Hit 'q' on the keyboard to quit!

        if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release handle to the webcam

video_capture.release()

cv2.destroyAllWindows()
```



참고 자료

- https://face-recognition.readthedocs.io/en/latest/readme.html
- https://medium.com/@ageitgey/build-a-hardware-based-face-recognition-
- system-for-150-with-the-nvidia-jetson-nano-and-python-a25cb8c891fd
- https://www.width.ai/post/facial-detection-and-recognition-with-dlib
- https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-
- dlib/