IVPL
Intelligent Vision Processing Lab

Computer Vision

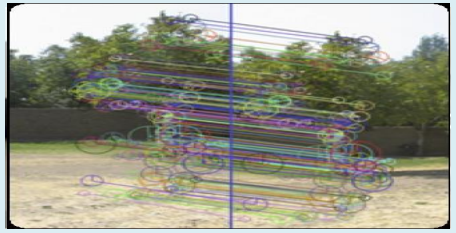
Deep Learning Basics
(#22: Yolo v5 Object Detection Model Training in Colab)

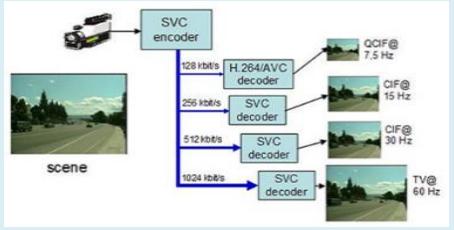
2023 Autumn

Prof. Byung-Gyu Kim
Intelligent Vision Processing Lab. (IVPL)
http://ivpl.sookmyung.ac.kr
Dept. of IT Engineering, Sookmyung Women's University
E-mail: bg.kim@sookmyung.ac.kr

Goal of this lecture

- How to train Yolo object detection API?
  - Preparation of My Dataset
  - Actual Training in Colab
  - How to apply the trained model to my local Tensorflow to detect objects?



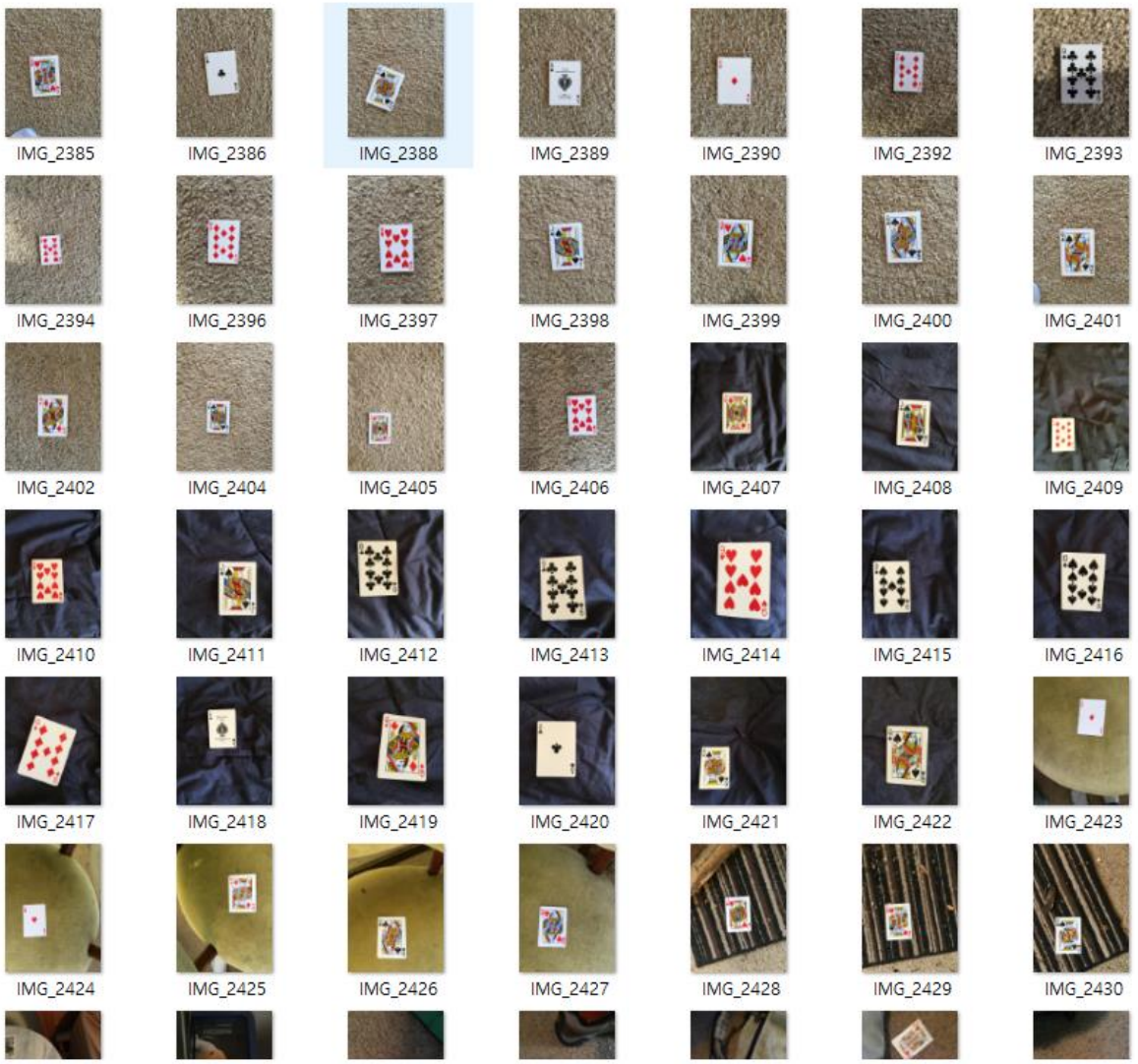


Contents

- My Github
- **Preparation of My Dataset**
- Actual Training Object Detection in Colab
- To my local Tensorflow to detect objects

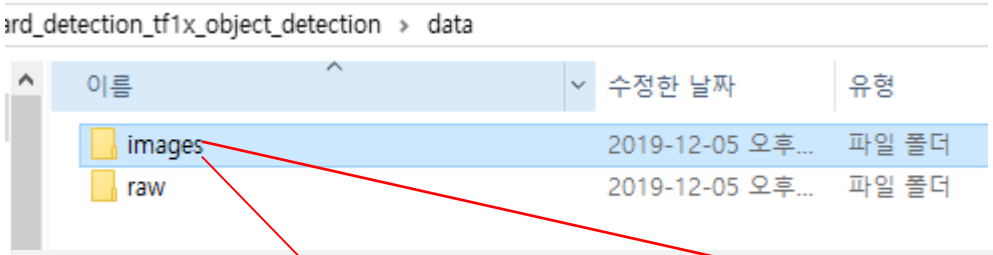
Preparation of My Dataset (1)

❖ **First you'd better to make very large number of photos in "raw" folder.**



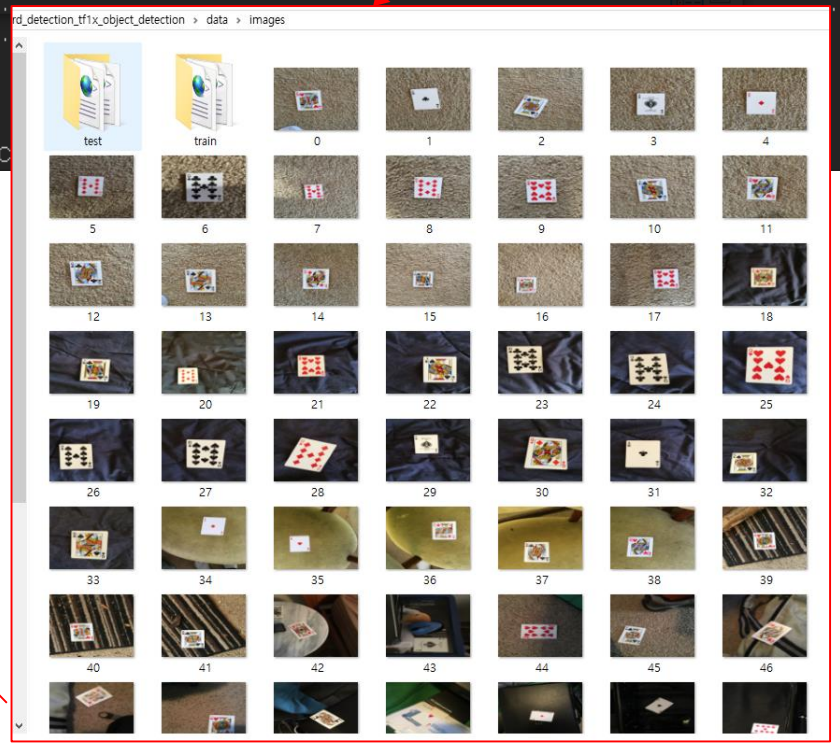
Preparation of My Dataset (2)

- Make all photos into the same size of images **(in Keras)**:



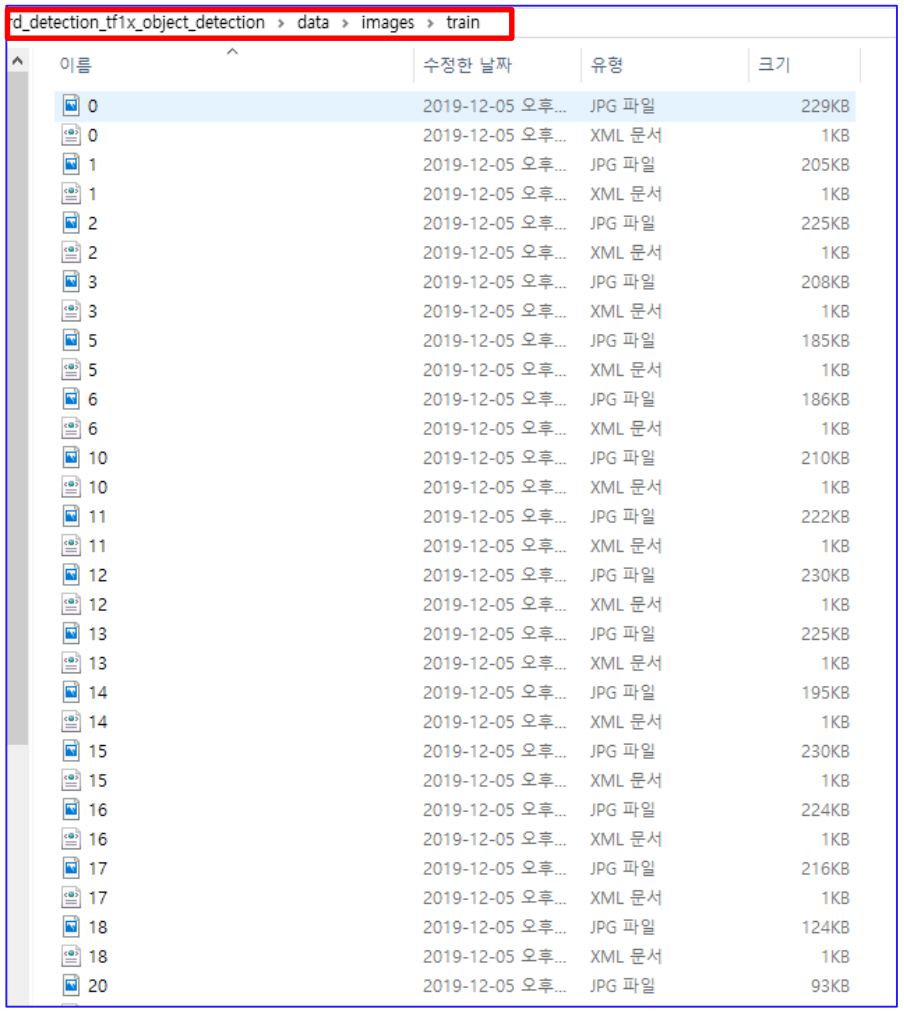
>>python resize_images.py --raw-dir ./data/raw --save-dir ./data/images --ext jpg --target-size "(800, 600)"

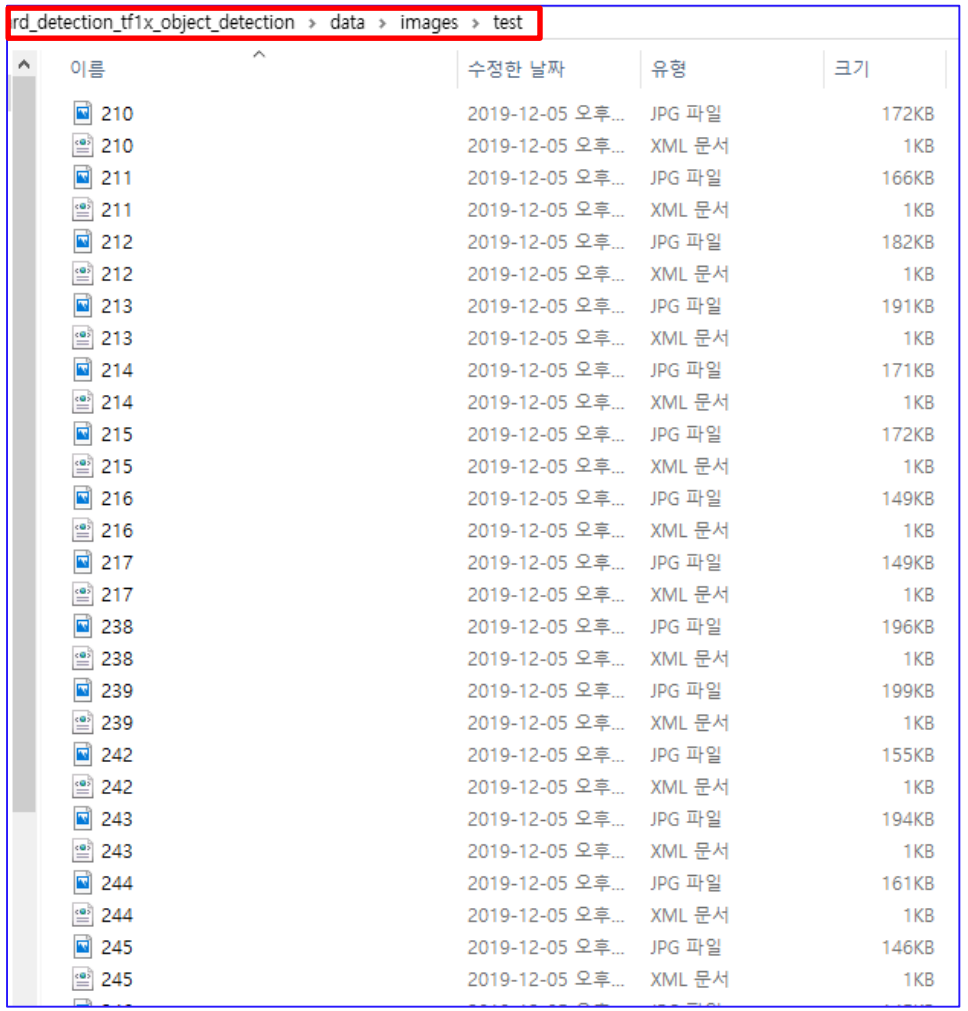
```
(BGKim) C:₩Users₩vicl₩practices₩TesorflowAPI₩card_detection_tf1x_object_detection>python resize_images.py --raw-dir ./data/raw --save-dir
 ./data/images --ext jpg --target-size "(800, 600)"
251 files to resize from directory `./data/raw` to target size:(800, 600)
....................................................................................................
....................................................................................................
Done resizing 251 files.
Saved to directory: `./data/images`

(BGKim) C:₩Users₩vicl₩practices₩TesorflowAPI₩card_detection_tf1x_object_detec
```



Preparation of My Dataset (3)

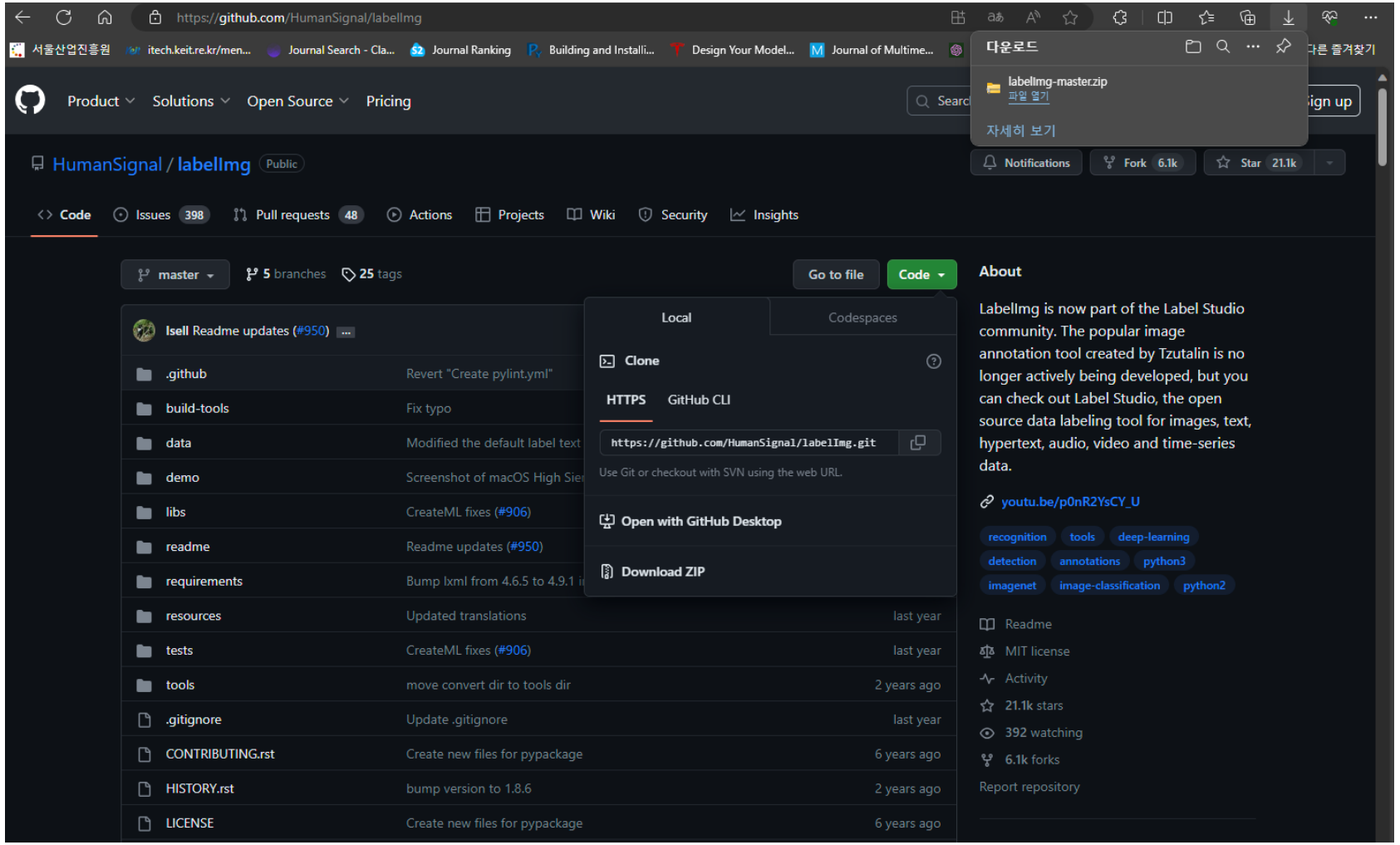
- Split dataset into "train" and "test" folders:
  - Usually, you should give more images into "train"(about 80%).
  - For "test", 20% is enough in usual.





Preparation of My Dataset (4): Data Annotation

- **Data Annotation (object labelling) using "LabelImg" Tool (https://github.com/HumanSignal/labelImg#usage/)**
  - Download zip file to install it.



Preparation of My Dataset (4-1): Data Annotation

- Decompress it to your folder and activate your conda account!!!
- Install the required packages as:

```
>>pip install pyqt5
>>pip install lxml
>>pyrcc5 -o libs/resources.py resources.qrc
```

- Go to your LabelImg folder!!! And run LabelImg.py like:

```
>>python .₩labelImg.py
      or
>>python LabelImg.py
```

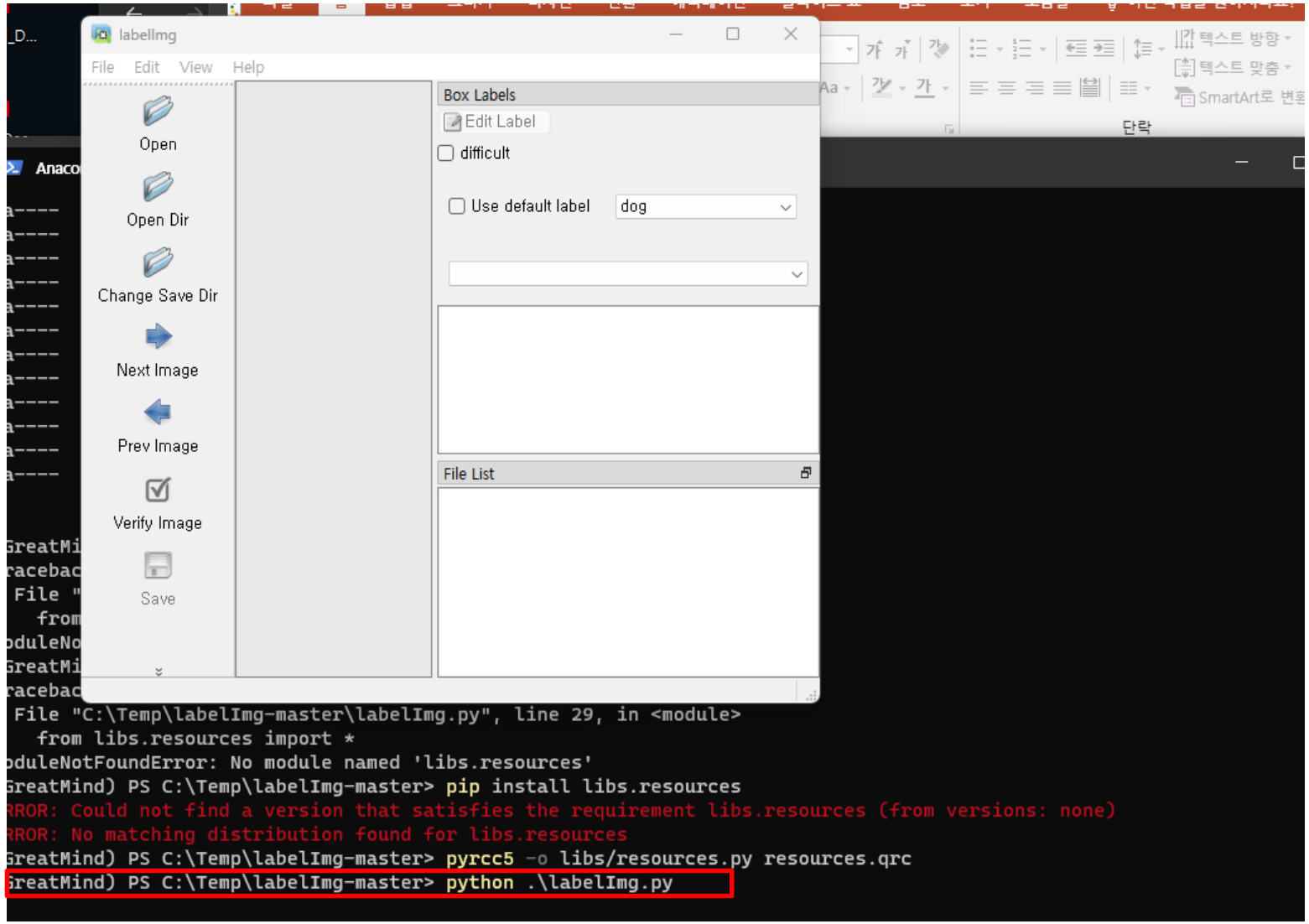
Windows + Anaconda

Download and install Anaconda (Python 3+)

Open the Anaconda Prompt and go to the labelImg directory

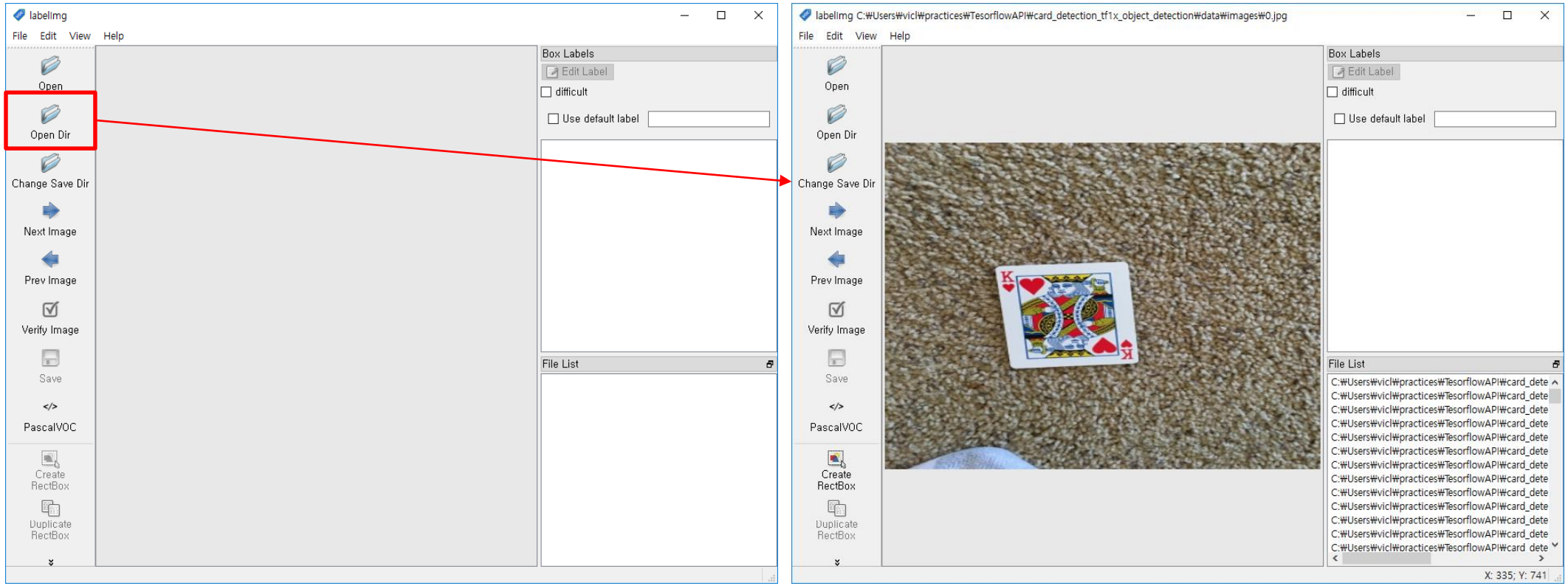
```
conda install pyqt=5
conda install -c anaconda lxml
pyrcc5 -o libs/resources.py resources.qrc
python labelImg.py
python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

Preparation of My Dataset (4-2): Data Annotation



Preparation of My Dataset (5): Data Annotation

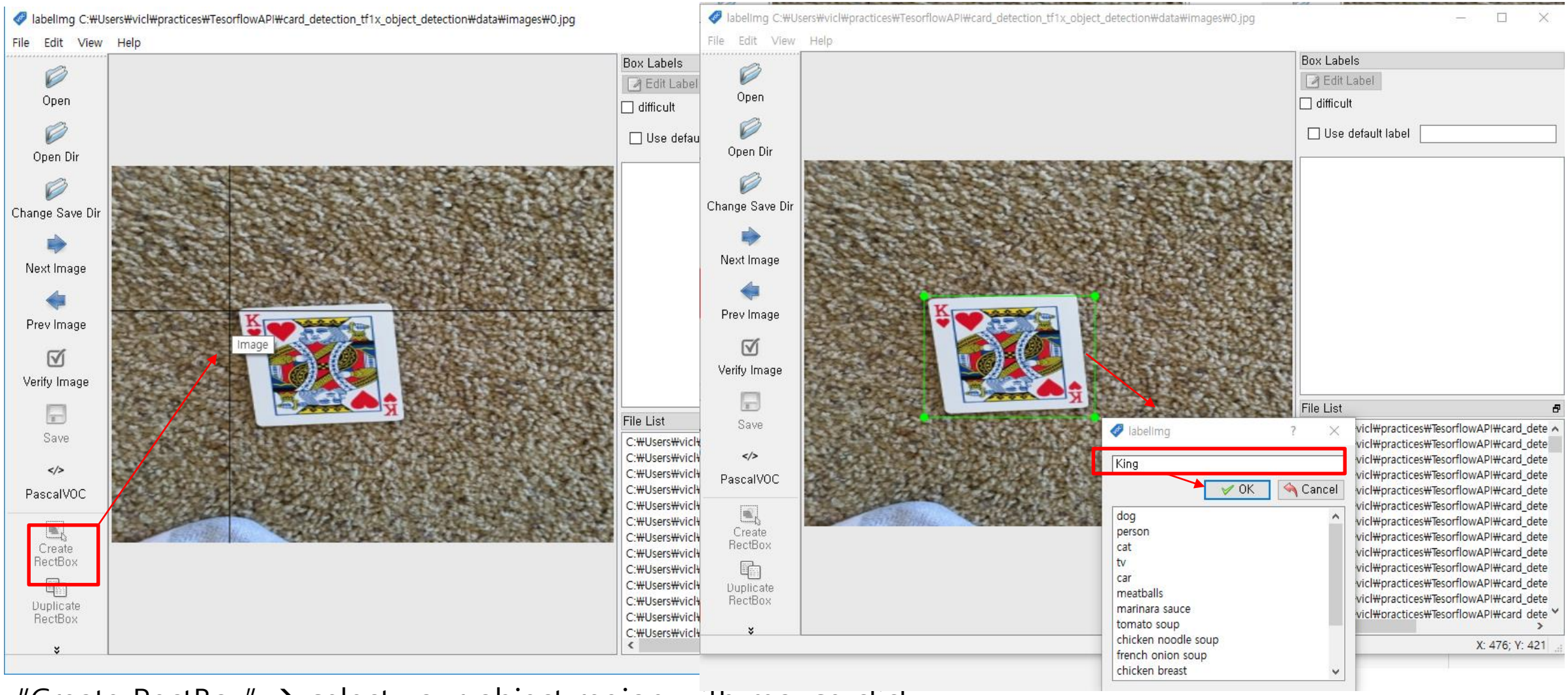
- Run LabelImg.exe



- Next Image: loading the next image
- Prev Image: go to the previous one

Preparation of My Dataset (6): Data Annotation

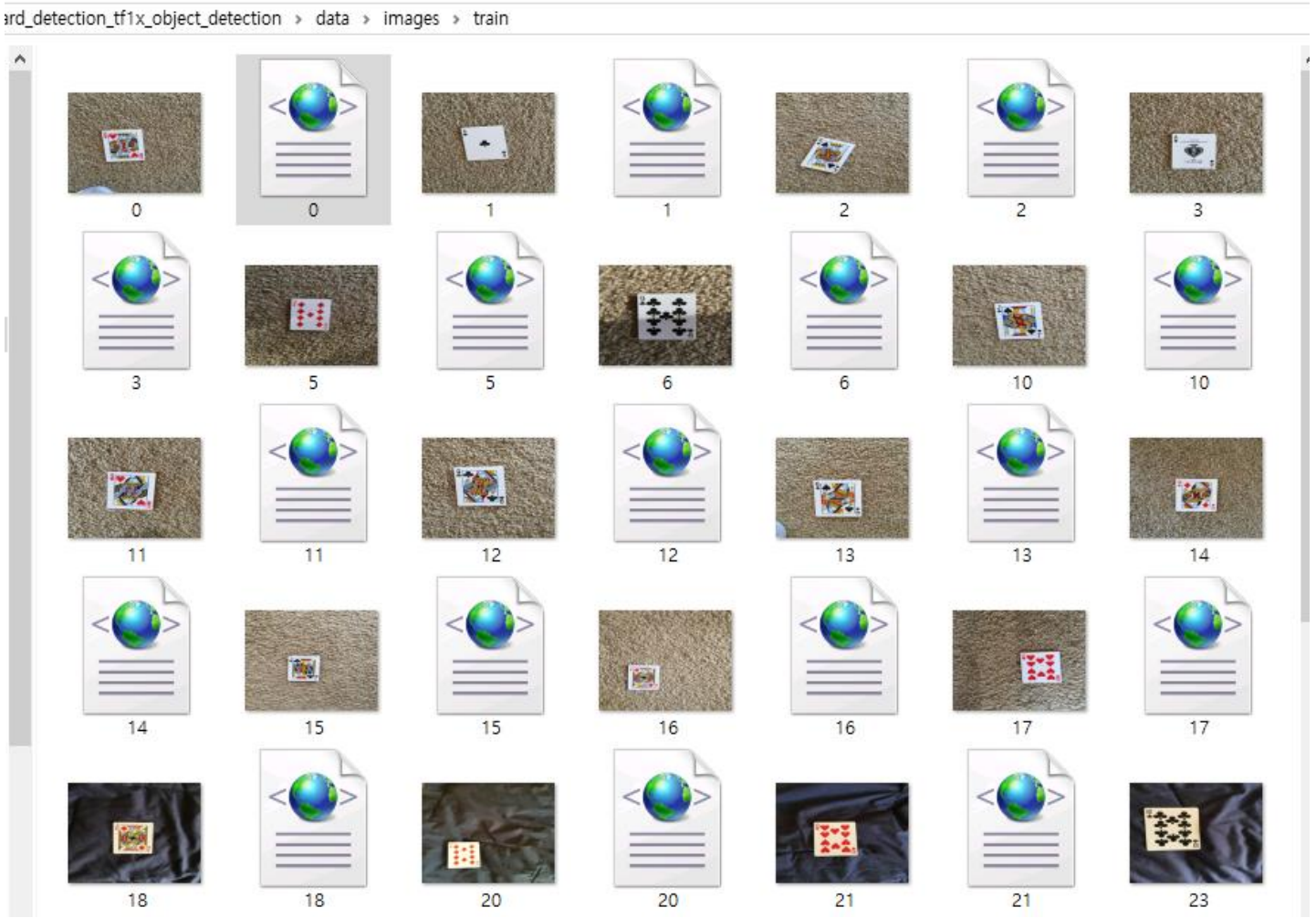
- Image labeling (annotation)



"Create RectBox" → select your object region with mouse click.

Preparation of My Dataset (7): Data Annotation

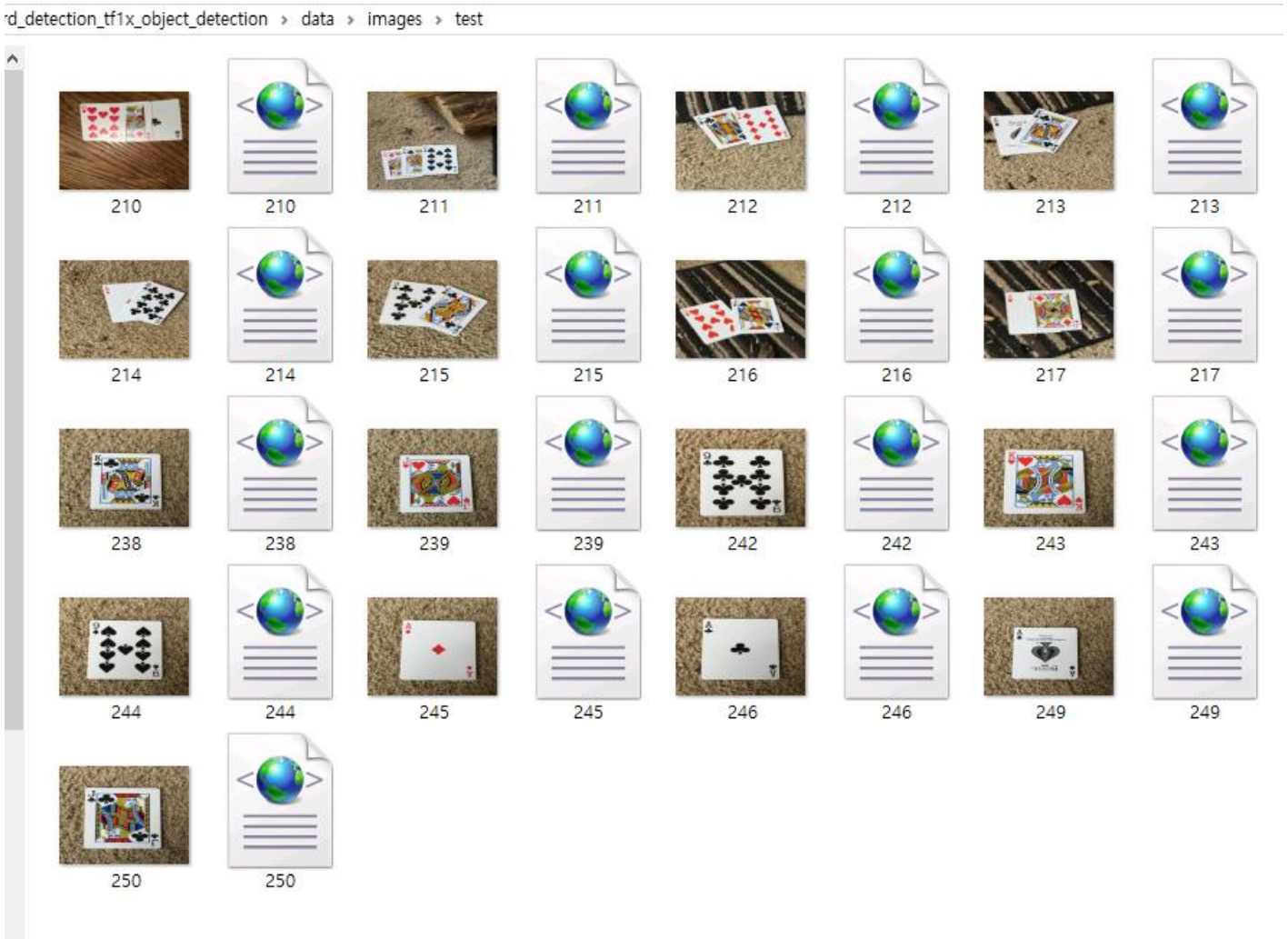
- After clicking "save" button, you can see what is change in "open dir".



For each image, you can save information in each xml file.

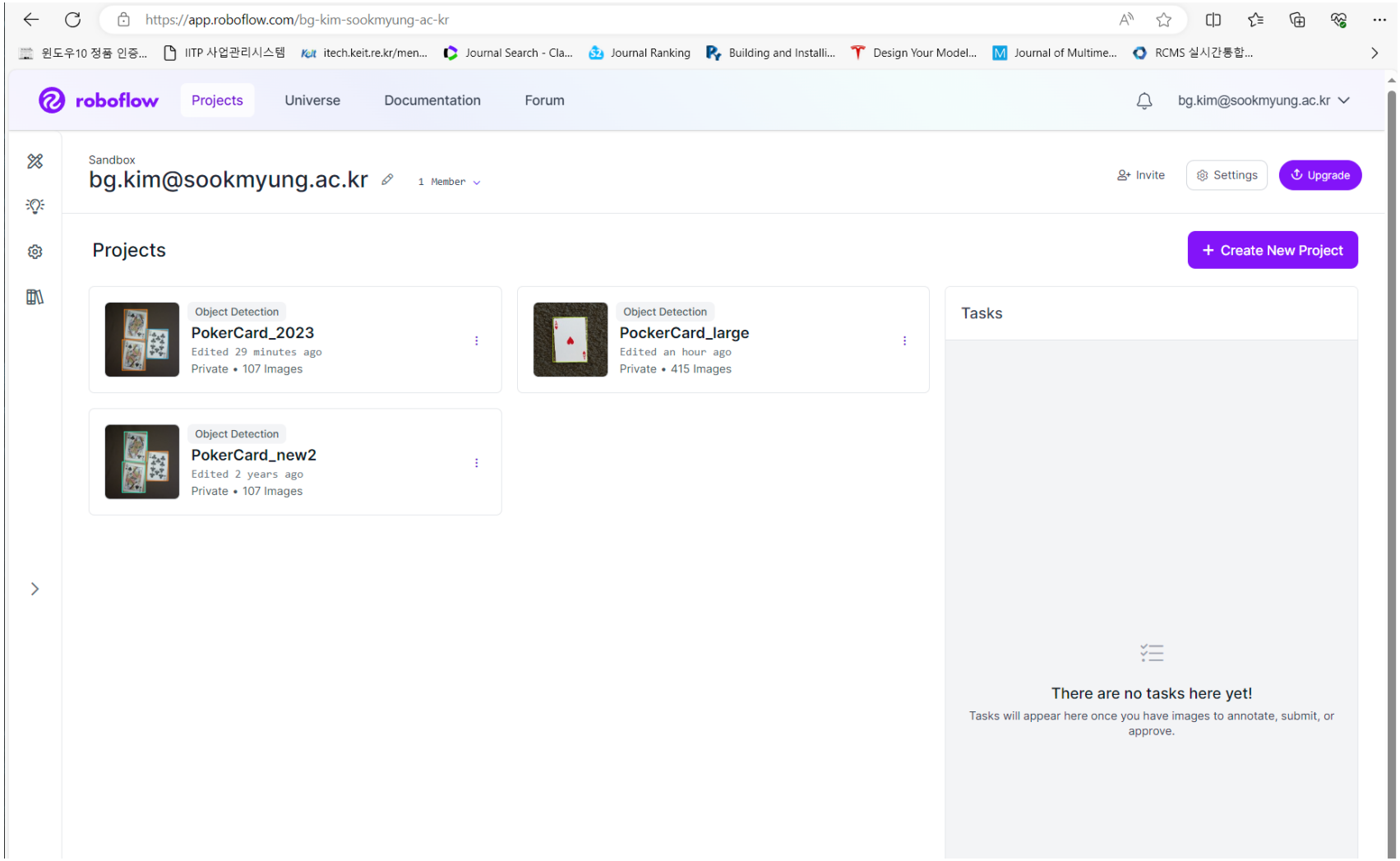
Preparation of My Dataset (8): Data Annotation

- For all your data, you have to give annotations (both of train and test datasets).



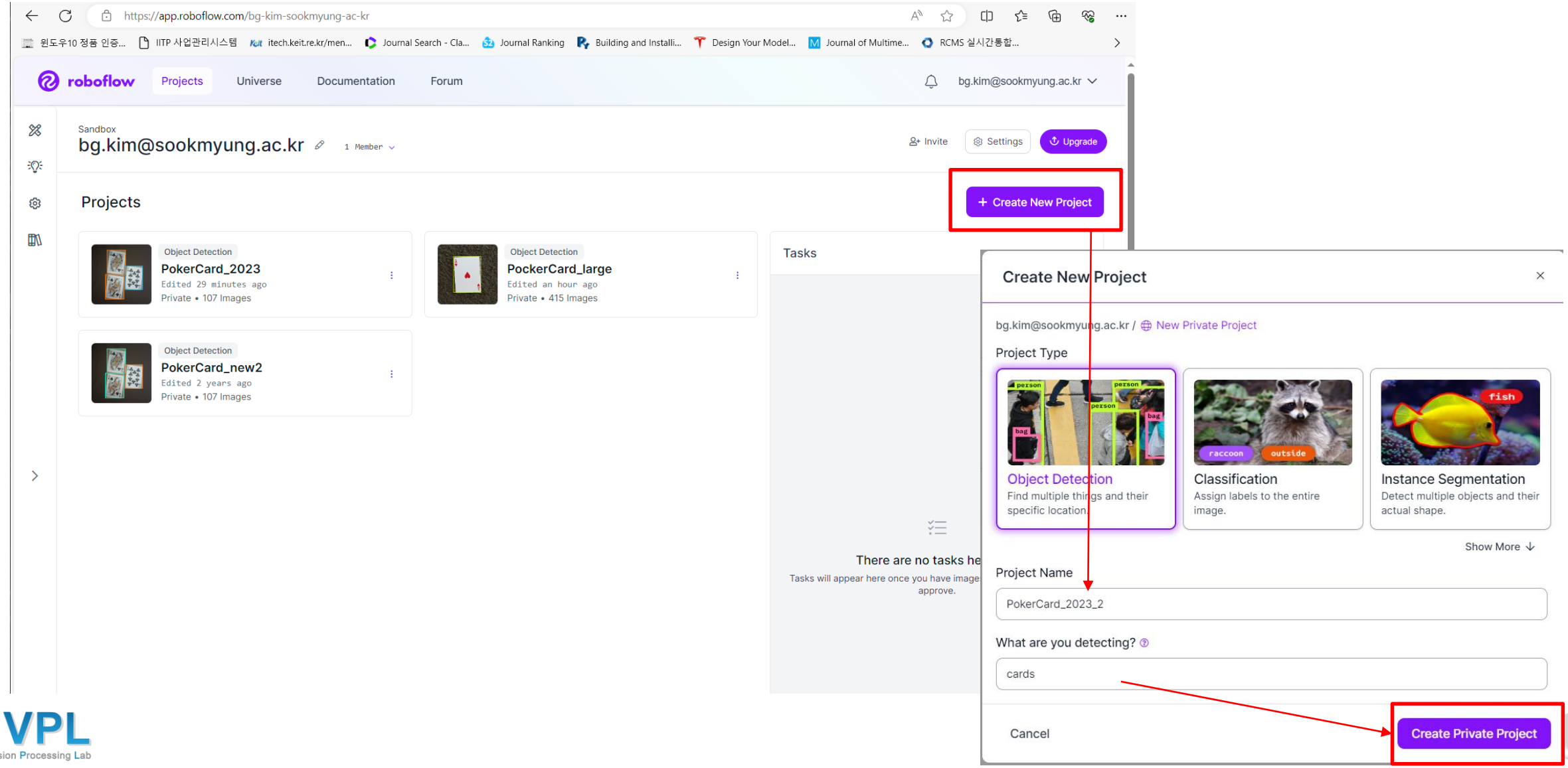
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- Robflow AI (https://app.roboflow.com/)
  - Join (make your account) and login!



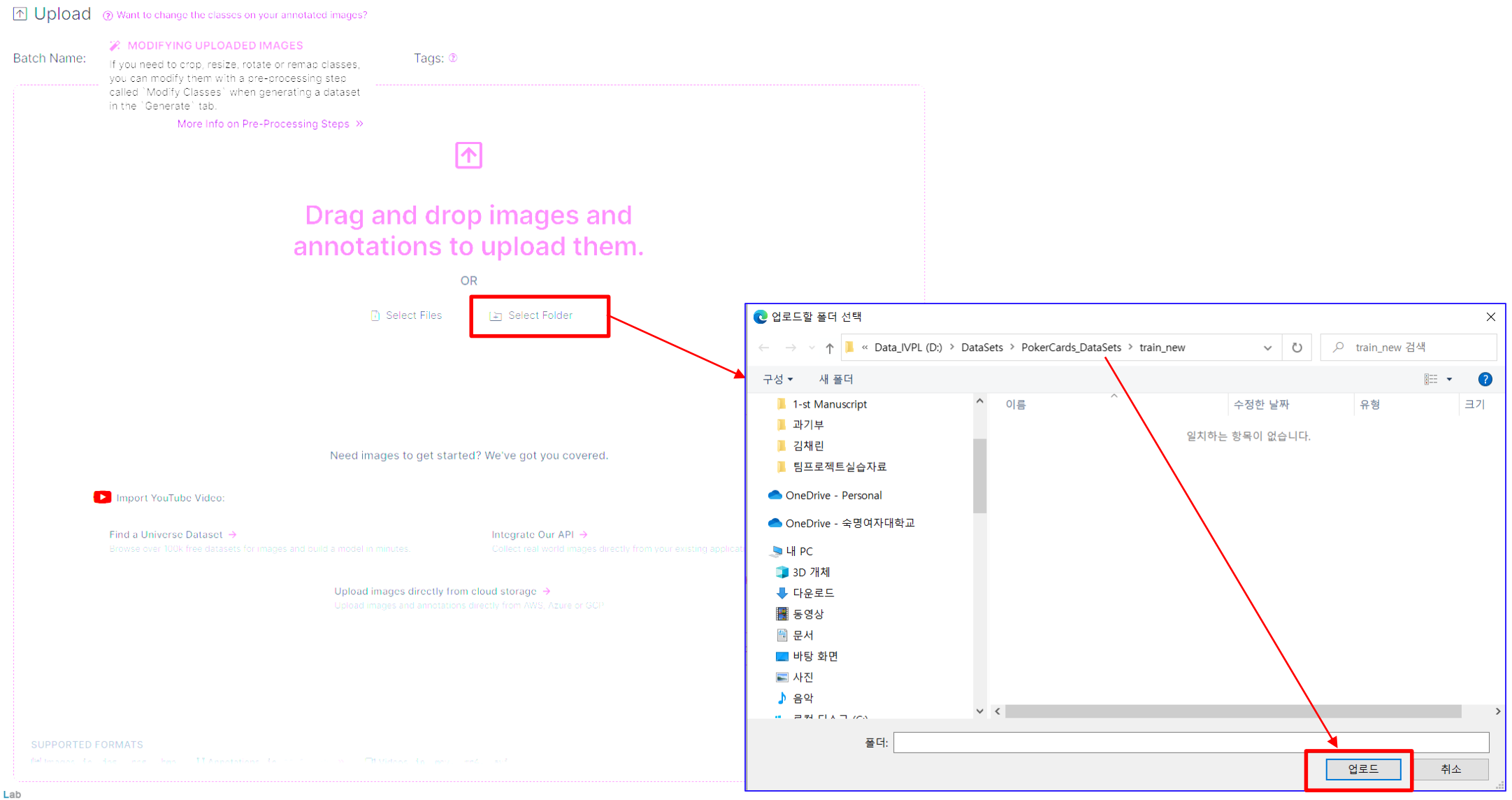
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- Click “Create New Project”



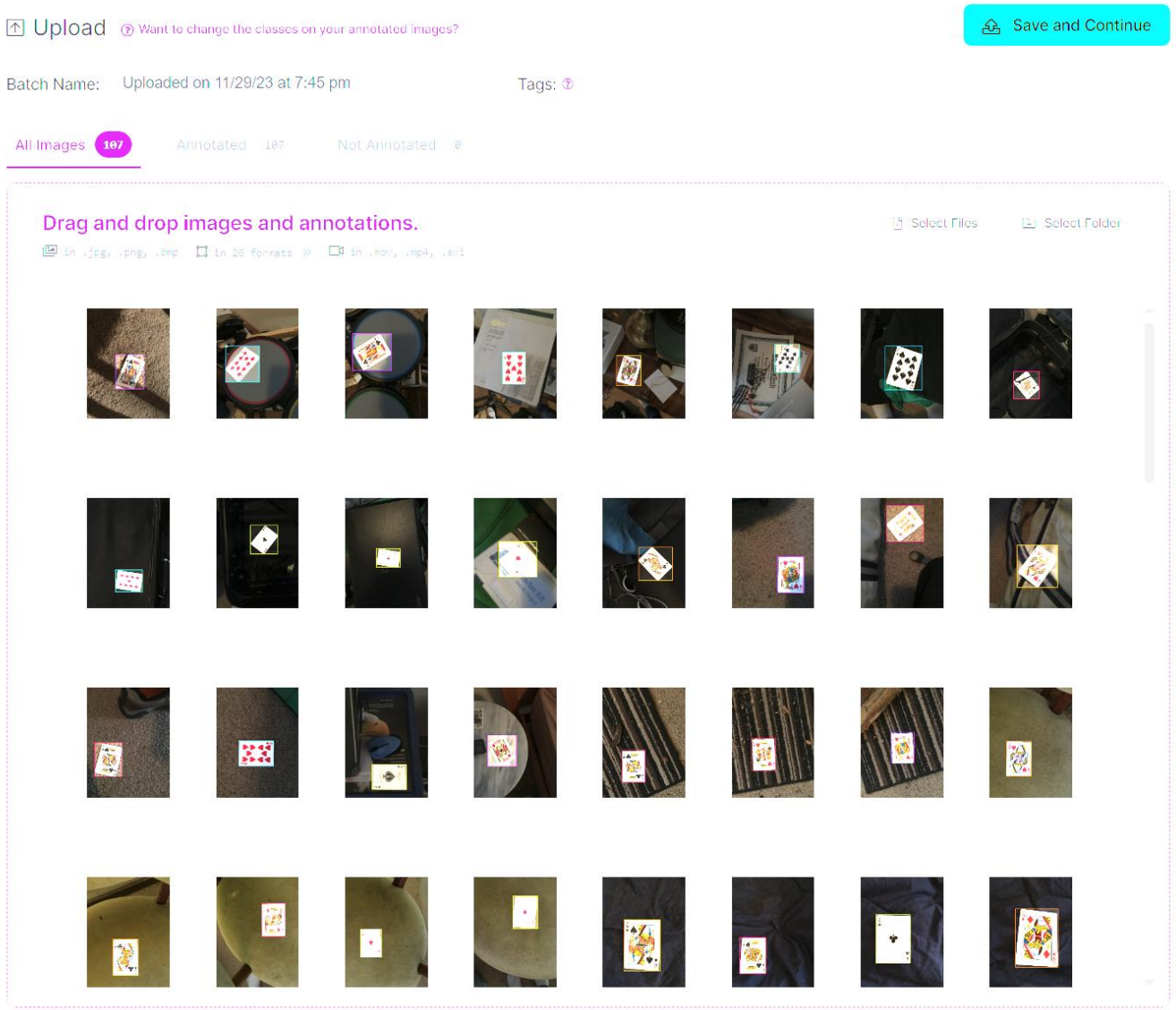
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- Upload your dataset: Use “Select Folder” and upload all data files where you choose.



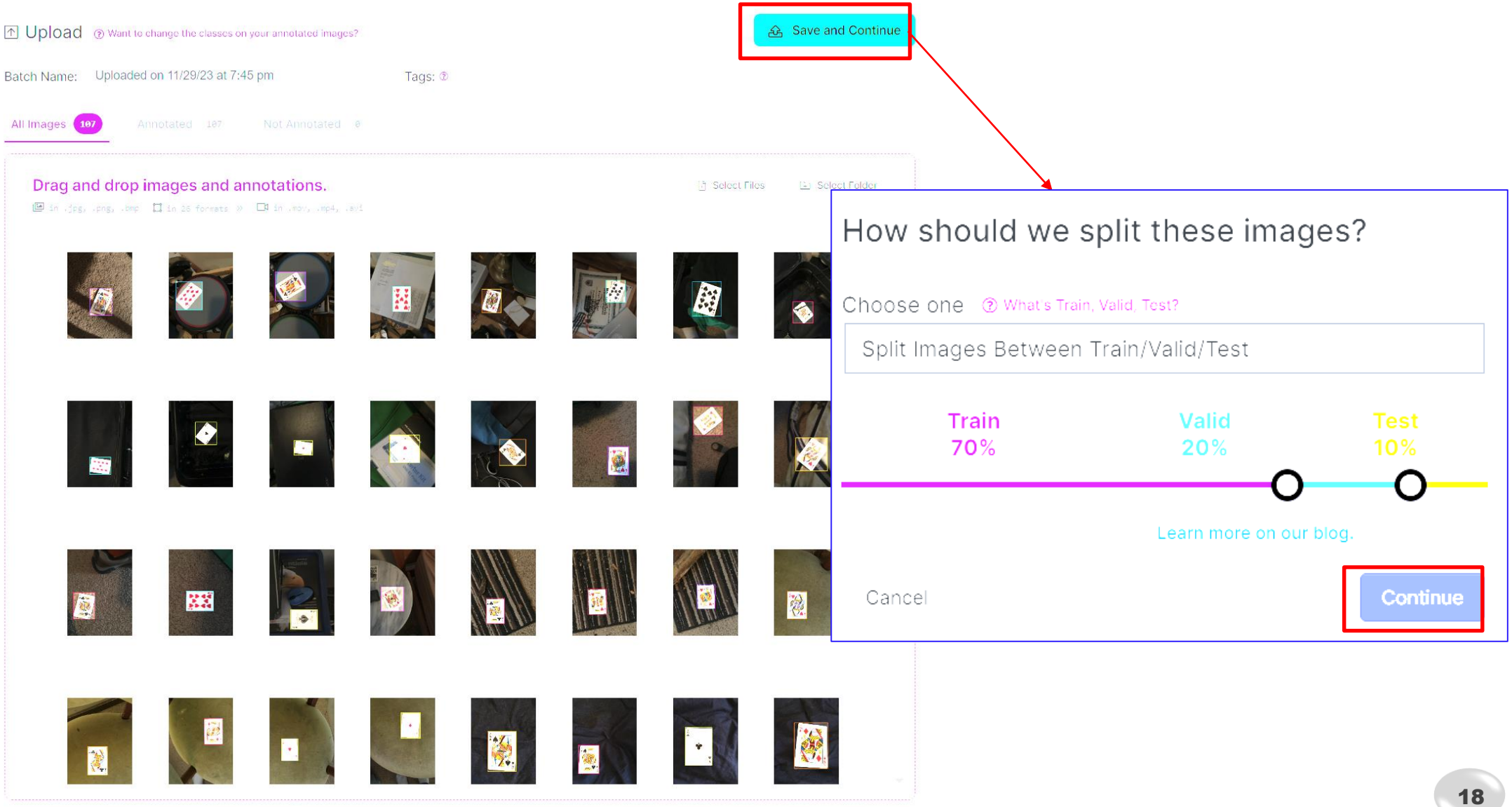
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- Check your uploaded data files



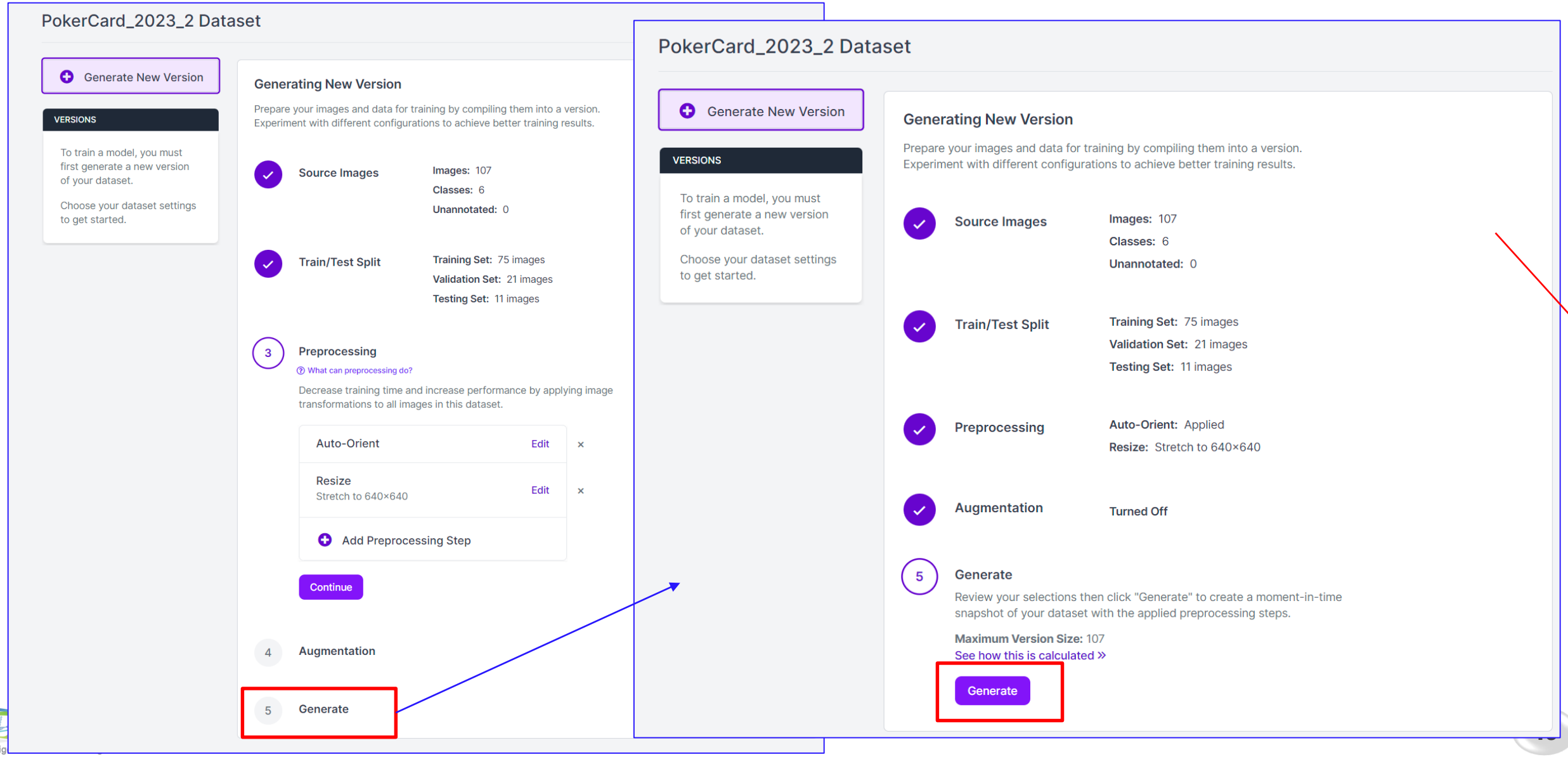
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- Check your uploaded data files



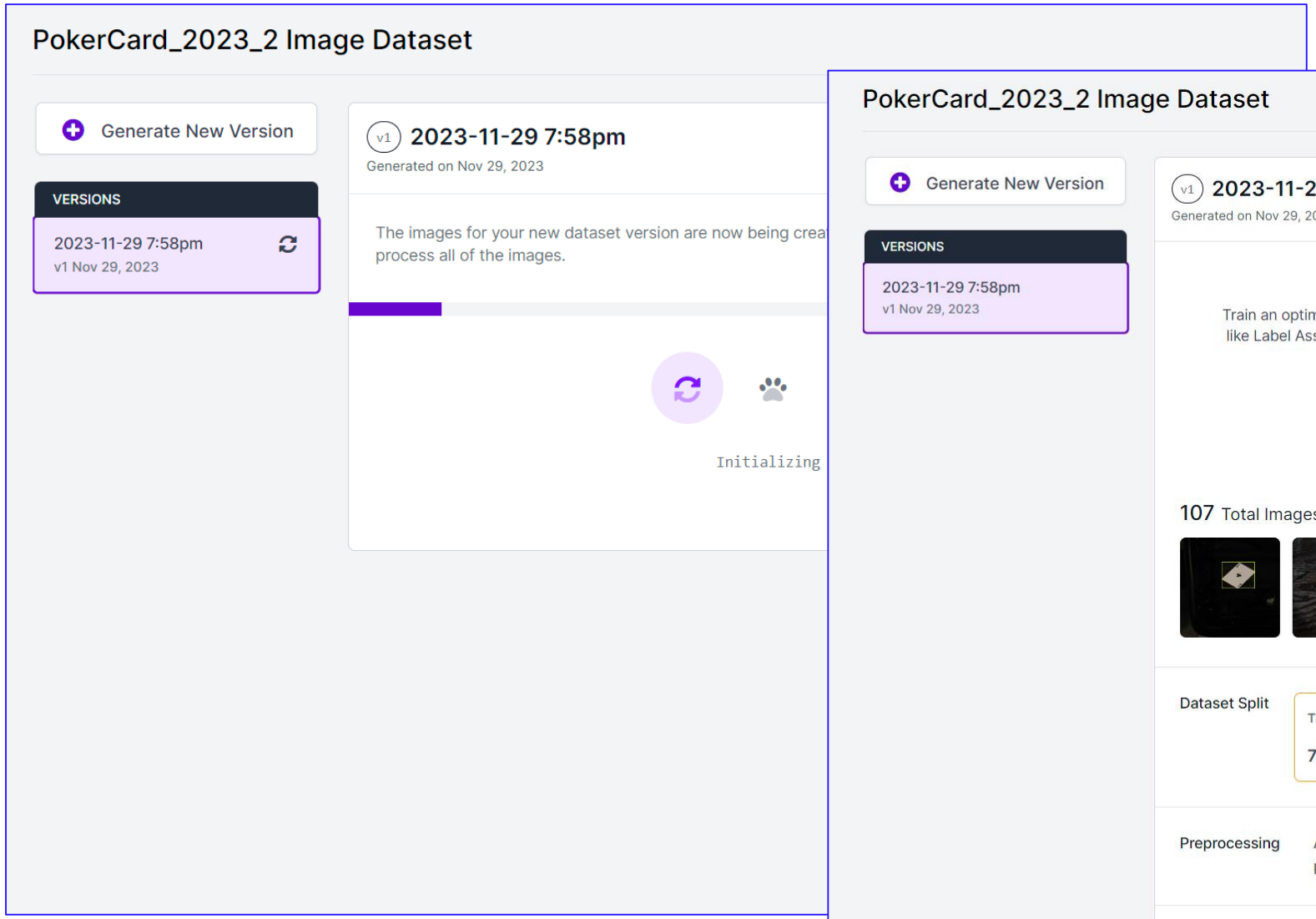
Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

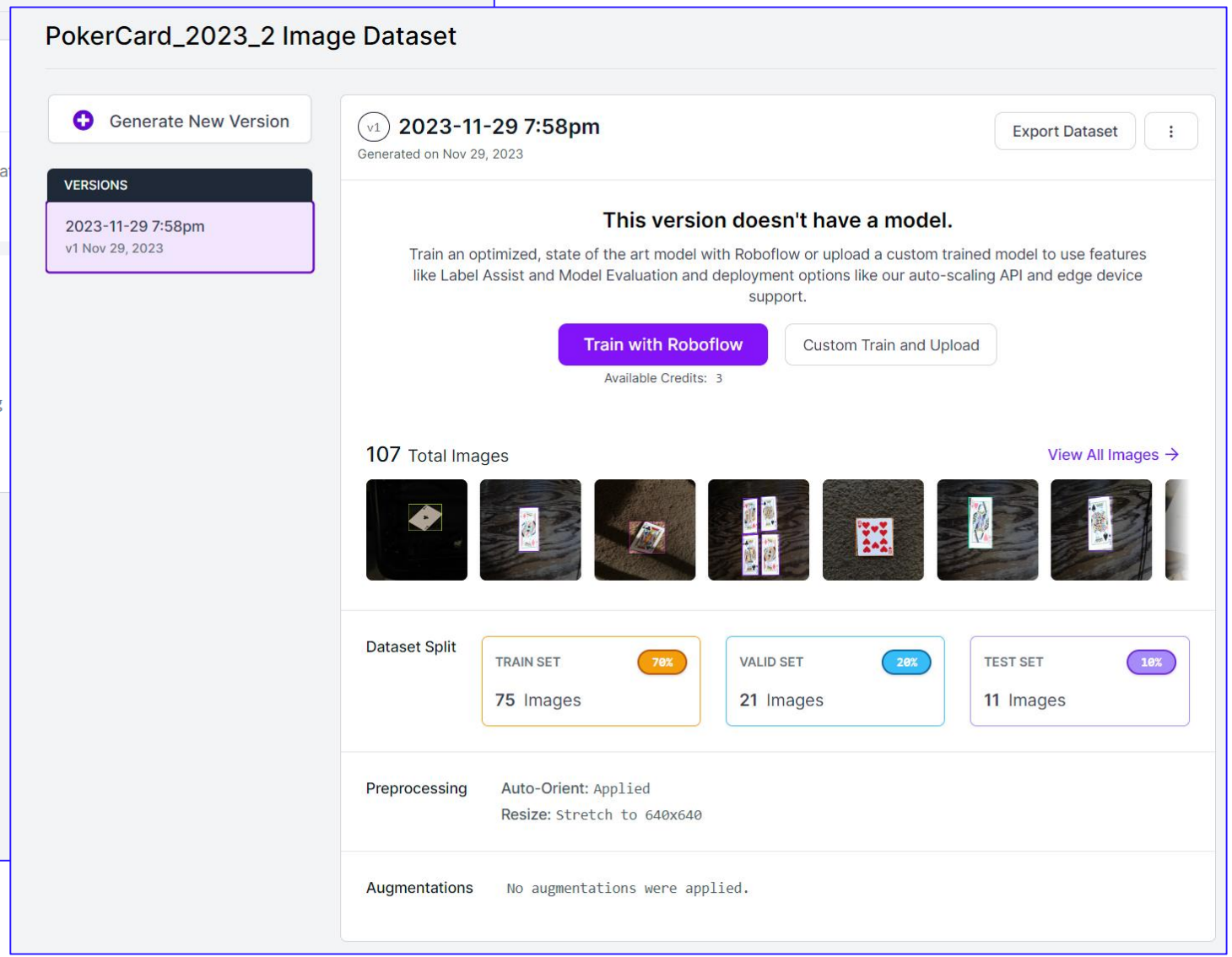
- Finish your dataset generation: Go to “5-Generate”



Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

- You can see the final processing for your dataset.

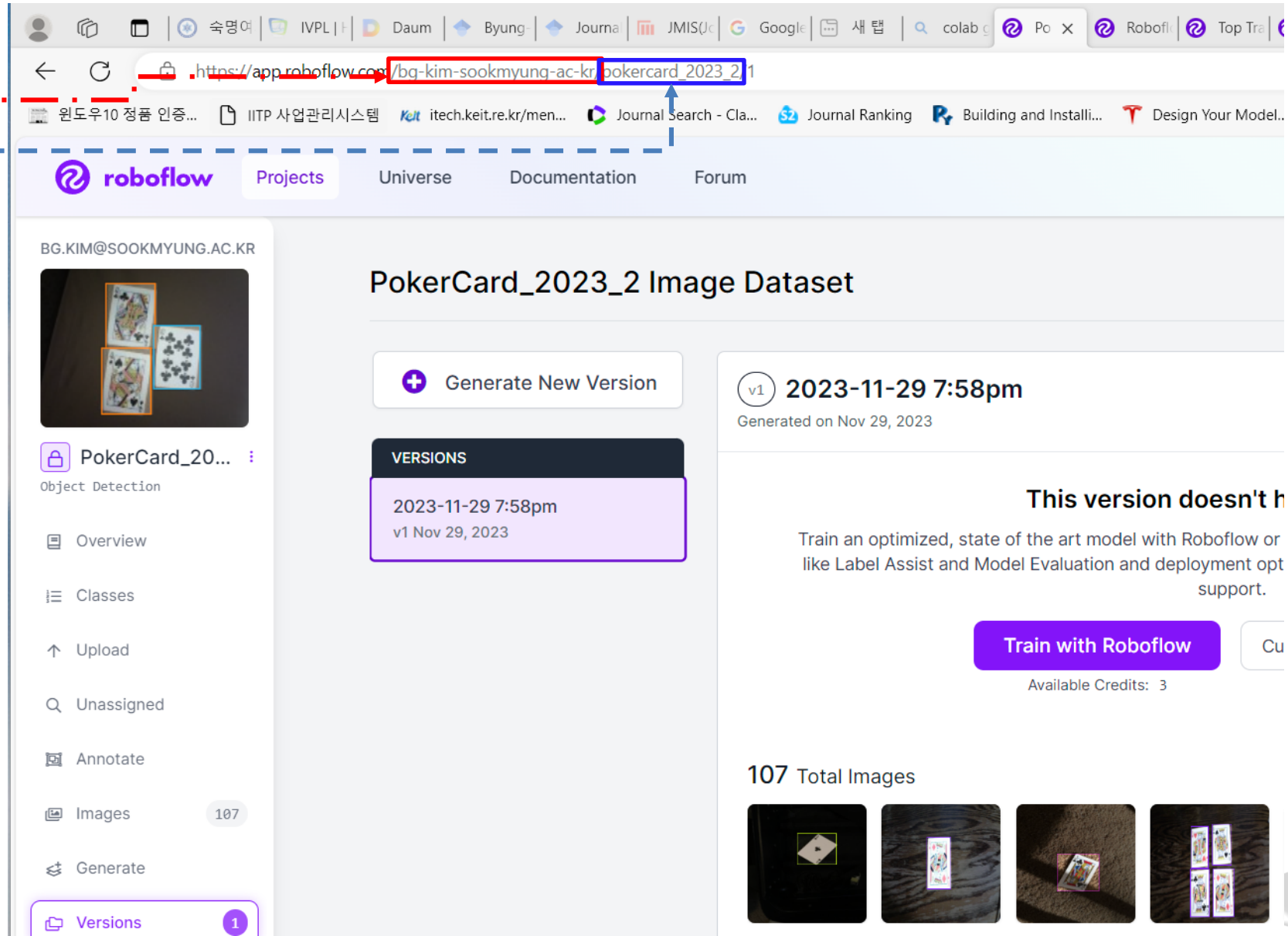




Preparation of My Dataset (8): Upload your dataset (annotated) to Roboflow

❖ Important Variables here….!!!!!

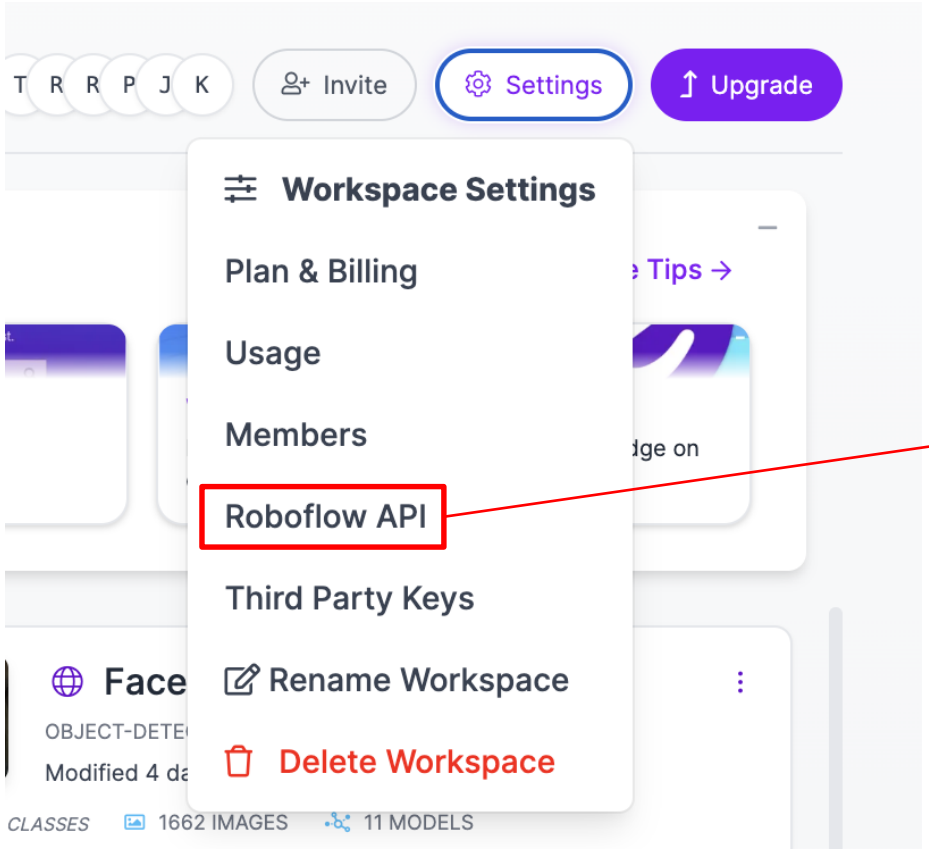
- API_KEY
- Workspace ID
- Project ID

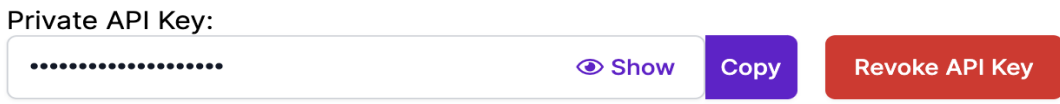


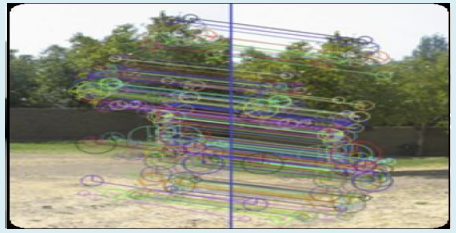
Preparation of My Dataset (9): Upload your dataset (annotated) to Roboflow

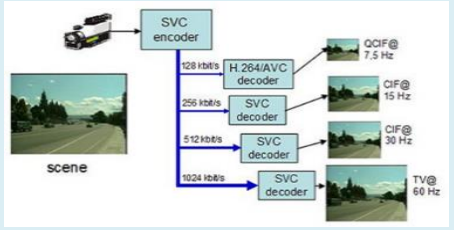
❖ User API Key

- First, go to the Roboflow dashboard. Then, select the Roboflow API tab from the settings page's sidebar navigation:
- Finally, **copy your Private API Key**. Be sure to keep this secret. Treat it like a password; it grants the bearer access to all of your workspace's data and models









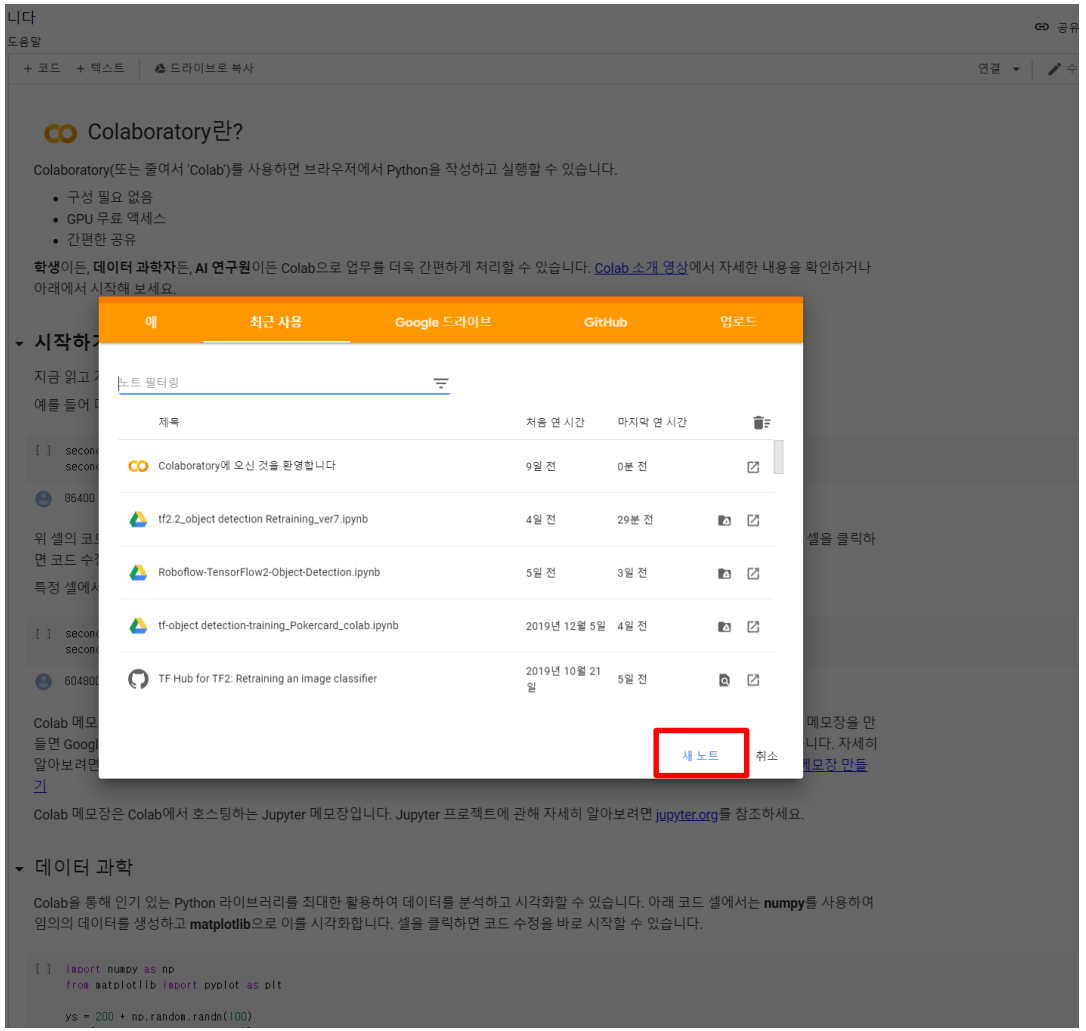
Contents

- My Github
- Preparation of My Dataset
- **Actual Training Object Detection in Colab**
- To my local Tensorflow to detect objects

Actual Training in Colab (1): Yolo v5

❖ Connect your colab (jyputer Notebook) with your account in Google.

- First you have to log-in Google.
- Type "https://colab.research.google.com/" in your broser. → "새노트" or 기존 노트 선택하면 됨



Actual Training in Colab (2): Yolo v5

❖ Download Yolov5 source code in Pytorch platform.

- Check on the download status.

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

```
ls -al
```

```
total 376
drwxr-xr-x 9 root root  4096 Nov 30 07:40 ./
drwxr-xr-x 1 root root  4096 Nov 30 07:40 ../
-rw-r--r-- 1 root root  8009 Nov 30 07:40 benchmarks.py
-rw-r--r-- 1 root root   393 Nov 30 07:40 CITATION.cff
drwxr-xr-x 2 root root  4096 Nov 30 07:40 classify/
-rw-r--r-- 1 root root  5009 Nov 30 07:40 CONTRIBUTING.md
drwxr-xr-x 5 root root  4096 Nov 30 07:40 data/
-rw-r--r-- 1 root root 15364 Nov 30 07:40 detect.py
-rw-r--r-- 1 root root  3701 Nov 30 07:40 .dockerignore
-rw-r--r-- 1 root root 41275 Nov 30 07:40 export.py
drwxr-xr-x 8 root root  4096 Nov 30 07:40 .git/
-rw-r--r-- 1 root root    75 Nov 30 07:40 .gitattributes
drwxr-xr-x 4 root root  4096 Nov 30 07:40 .github/
-rwxr-xr-x 1 root root  3998 Nov 30 07:40 .gitignore*
-rw-r--r-- 1 root root  7769 Nov 30 07:40 hubconf.py
-rw-r--r-- 1 root root 34523 Nov 30 07:40 LICENSE
drwxr-xr-x 4 root root  4096 Nov 30 07:40 models/
-rw-r--r-- 1 root root  1815 Nov 30 07:40 .pre-commit-config.yaml
-rw-r--r-- 1 root root 42023 Nov 30 07:40 README.md
-rw-r--r-- 1 root root 41050 Nov 30 07:40 README.zh-CN.md
-rw-r--r-- 1 root root  1563 Nov 30 07:40 requirements.txt
drwxr-xr-x 2 root root  4096 Nov 30 07:40 segment/
-rw-r--r-- 1 root root  1723 Nov 30 07:40 setup.cfg
-rw-r--r-- 1 root root 33959 Nov 30 07:40 train.py
-rw-r--r-- 1 root root 41420 Nov 30 07:40 tutorial.ipynb
drwxr-xr-x 8 root root  4096 Nov 30 07:40 utils/
-rw-r--r-- 1 root root 20586 Nov 30 07:40 val.py
```

Actual Training in Colab (3): Yolo v5

❖ Download your dataset in Roboflow server.

- You need Your API_KEY, Workspace ID, Project ID.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="MMQccvY9HKjgRLQzIwLO")
project = rf.workspace("bg-kim-sookmyung-ac-kr").project("pokercard_2023")
dataset = project.version(1).download("yolov5")

!ls
```

```
    Found existing installation: chardet 5.2.0
    Uninstalling chardet-5.2.0:
      Successfully uninstalled chardet-5.2.0
Successfully installed chardet-4.0.0 cycler-0.10.0 idna-2.10 opencv-python-headless-4.8.0.74 pyparsing-2.4.7 python-dotenv-1.0.0 python-magic-
WARNING: The following packages were previously imported in this runtime:
  [cycler,pyparsing]
You must restart the runtime in order to use newly installed versions.

 RESTART RUNTIME

loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in PokerCard_2023-1 to yolov5pytorch:: 100%|██████████| 14295/14295 [00:00<00:00, 57002.08it/s]

Extracting Dataset Version Zip to PokerCard_2023-1 in yolov5pytorch:: 100%|██████████| 525/525 [00:00<00:00, 5541.03it/s]
benchmarks.py    data        LICENSE           README.zh-CN.md   train.py
CITATION.cff     detect.py   models            requirements.txt  tutorial.ipynb
classify         export.py   PokerCard_2023-1  segment           utils
CONTRIBUTING.md  hubconf.py  README.md         setup.cfg         val.py
```

Actual Training in Colab (4): Yolo v5

❖ Download the pre-trained weight from Guihub.

```
!wget https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt
!ls
```

```
--2023-11-30 07:48:56--  https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/381bd8a8-8910-4e9e-b0dd-2752951ef78
--2023-11-30 07:48:57--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/381bd8a8-8910-4e9e-b
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to: 'yolov5s.pt'

yolov5s.pt          100%[===================>]  14.12M  --.-KB/s    in 0.09s

2023-11-30 07:48:57 (162 MB/s) - 'yolov5s.pt' saved [14808437/14808437]

benchmarks.py    data        LICENSE           README.zh-CN.md   train.py        yolov5s.pt
CITATION.cff     detect.py   models            requirements.txt  tutorial.ipynb
classify         export.py   PokerCard_2023-1  segment           utils
CONTRIBUTING.md  hubconf.py  README.md         setup.cfg         val.py
```

Actual Training in Colab (5): Yolo v5

❖ Download the pre-trained weight from Guihub.

```
!wget https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt
!ls
```

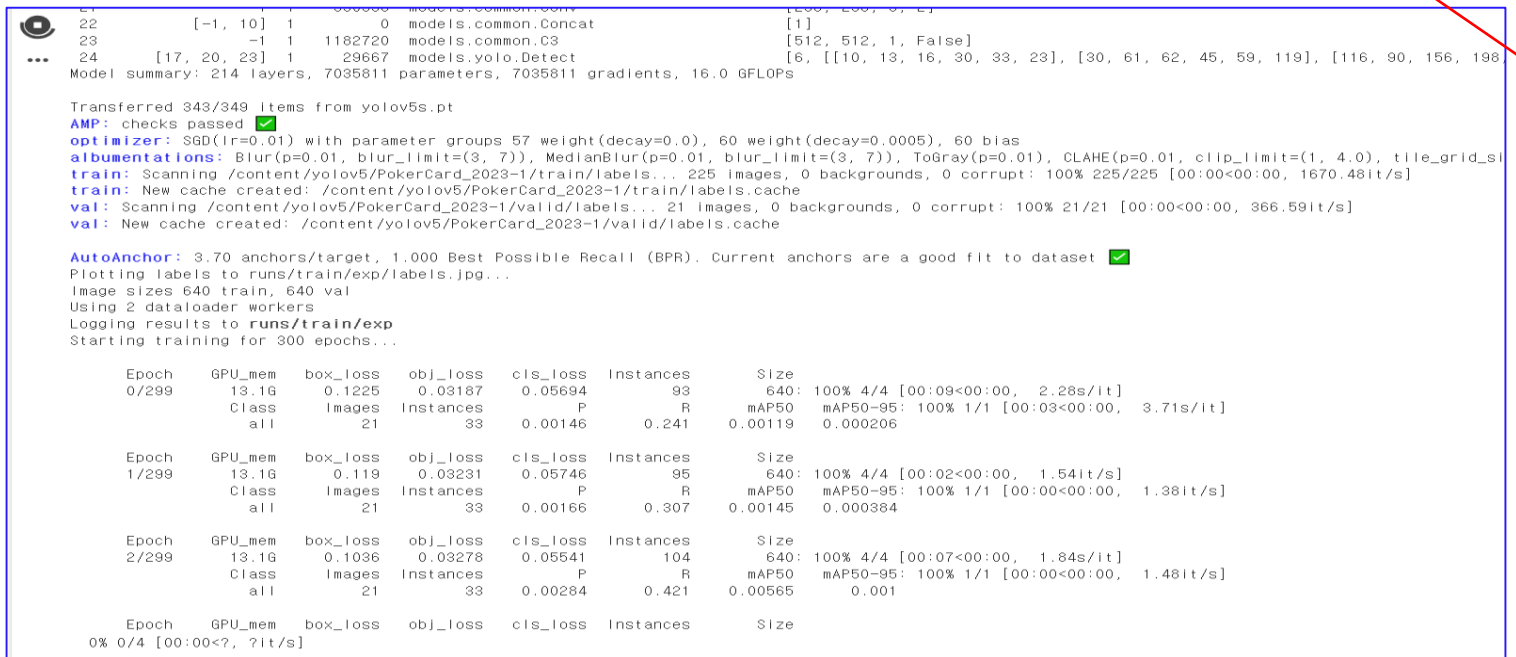
```
--2023-11-30 07:48:56--  https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/381bd8a8-8910-4e9e-b0dd-2752951ef78
--2023-11-30 07:48:57--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/381bd8a8-8910-4e9e-b
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to: 'yolov5s.pt'

yolov5s.pt          100%[===================>]  14.12M  --.-KB/s    in 0.09s

2023-11-30 07:48:57 (162 MB/s) - 'yolov5s.pt' saved [14808437/14808437]

benchmarks.py    data        LICENSE           README.zh-CN.md  train.py        yolov5s.pt
CITATION.cff     detect.py   models            requirements.txt tutorial.ipynb
classify         export.py   PokerCard_2023-1  segment          utils
CONTRIBUTING.md  hubconf.py  README.md         setup.cfg        val.py
```

```
!ls -al
```

Actual Training in Colab (6): Yolo v5

❖ Train Yolo v5 model (training stage).

```
# 학습 단계
# --img      : 이미지 크기
# --batch    : 배치 크기
# --epochs   : 전체 학습 반복 횟수
# --data     : 데이터셋 경로 (data.yaml 파일은 로보플로에서 자동 생성됨)
# --weights  : 이전에 학습된 weight 파일의 경로

#!python train.py --img 640 --batch 64 --epochs 300 --data PokerCard-1/data.yaml --
weights yolov5s.pt
!python train.py --img 640 --batch 64 --epochs 300 --data PokerCard_2023-1/data.yaml --
weights yolov5s.pt
```

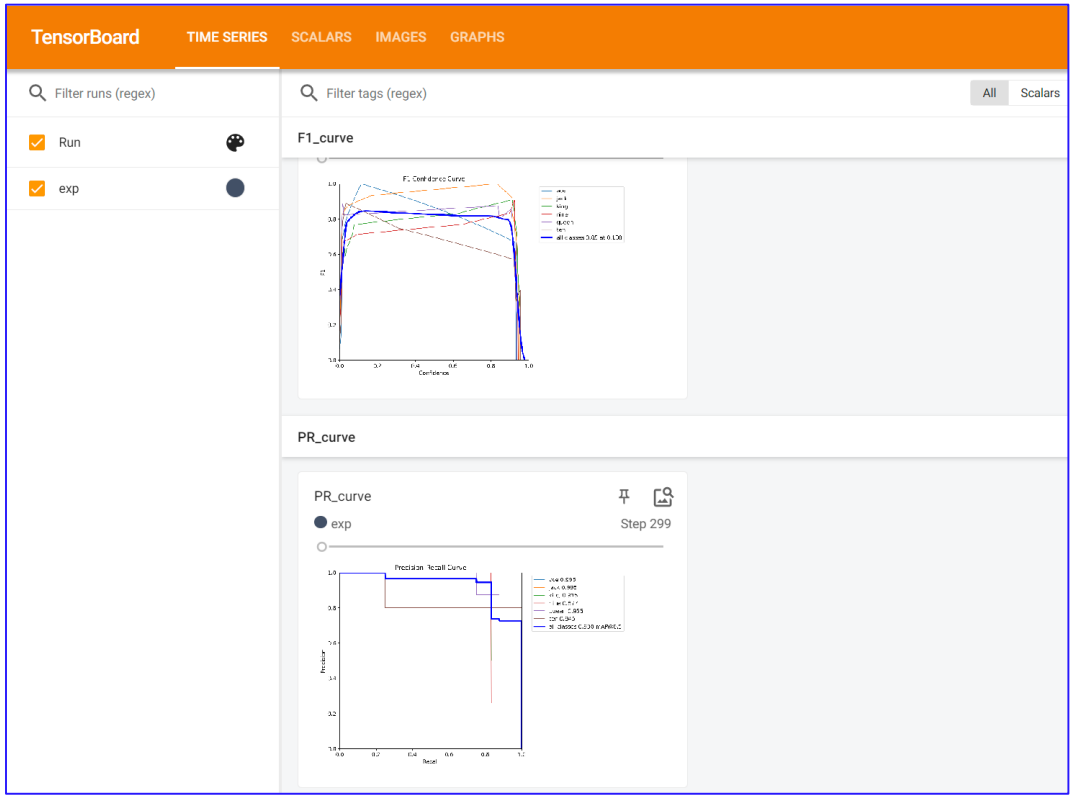


Your data folder

Actual Training in Colab (7): Yolo v5

❖ Checking the training status.

```
# 학습 추세 확인
# 추세 및 가중치 파일은 runs/train/ 경로 아래에 exp 라는 이름의 디렉토리에 생성됨
# 학습을 중단 후 다시 시작하면 exp2, exp3, exp4, ... 와 같이 숫자를 붙여 생성됨

%load_ext tensorboard
%tensorboard --logdir runs/train
```



Actual Training in Colab (8): Yolo v5

❖ Evaluating the trained network.

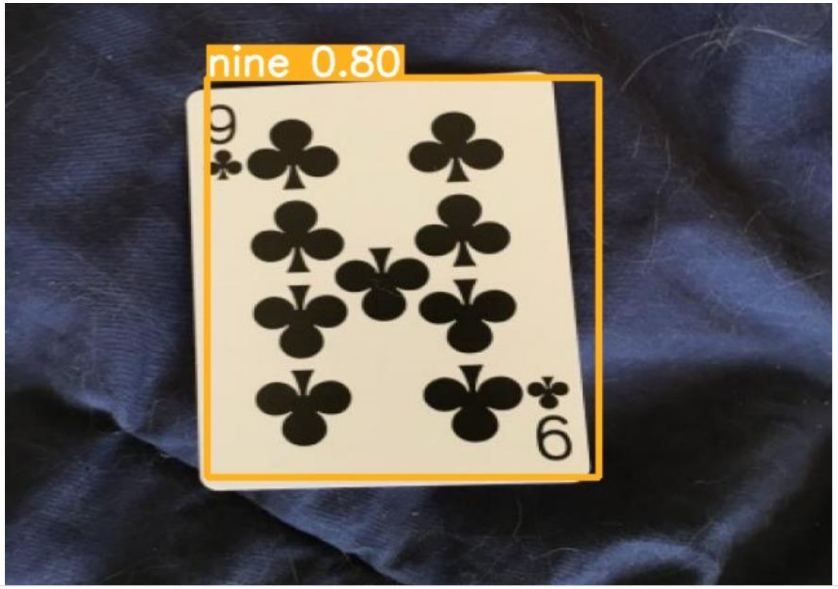
```
# 평가 단계
#!python detect.py --weight runs/train/exp/weights/best.pt --conf
0.1 --source PokerCard-1/valid/images
!python detect.py --weight runs/train/exp/weights/best.pt --conf 0.1
--source PokerCard_2023-1/valid/images
```

```
detect: weights=['runs/train/exp/weights/best.pt'], source=PokerCard-1/valid/images, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.1, iou_thres=0.45
YOLOv5 🚀 v7.0-247-g3f02fde Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
image 1/15 /content/yolov5/PokerCard-1/valid/images/14_jpg.rf.cc24109f8fc6b9ed09c61b3526a04d78.jpg: 640x640 1 nine, 11.6ms
image 2/15 /content/yolov5/PokerCard-1/valid/images/15_jpg.rf.77f89f8dab903e9468a74a0ae5768072.jpg: 640x640 1 nine, 11.6ms
image 3/15 /content/yolov5/PokerCard-1/valid/images/18_jpg.rf.df4563f18597a4515731367bfdb08e41.jpg: 640x640 1 ace, 11.6ms
image 4/15 /content/yolov5/PokerCard-1/valid/images/246_jpg.rf.2978c534b557456fc23c8658d583d0e0.jpg: 640x640 1 ace, 11.6ms
image 5/15 /content/yolov5/PokerCard-1/valid/images/249_jpg.rf.2b887dbda28ac7e5cc4fd0b2a507fd63.jpg: 640x640 1 ace, 11.5ms
image 6/15 /content/yolov5/PokerCard-1/valid/images/2_jpg.rf.e1504b3030dec38ca389d2e6d74da527.jpg: 640x640 1 queen, 11.5ms
image 7/15 /content/yolov5/PokerCard-1/valid/images/35_jpg.rf.0cb327ebfd08127a93b24cb5b6956888.jpg: 640x640 1 ace, 1 ten, 11.5ms
image 8/15 /content/yolov5/PokerCard-1/valid/images/38_jpg.rf.3af00ca6fe19faf2d8e041f72759a5ab.jpg: 640x640 1 ace, 2 jacks, 1 nine, 2 queens, 1 ten, 11.5ms
image 9/15 /content/yolov5/PokerCard-1/valid/images/40_jpg.rf.e80820292b668aaa7f65d5d5c3de25ad.jpg: 640x640 2 nines, 2 queens, 11.5ms
image 10/15 /content/yolov5/PokerCard-1/valid/images/45_jpg.rf.4167b7229ce818cb437f779696c58f67.jpg: 640x640 2 queens, 1 ten, 11.5ms
image 11/15 /content/yolov5/PokerCard-1/valid/images/58_jpg.rf.b7ce2b54996e2344da7f22aef91ad2a0.jpg: 640x640 2 aces, 1 nine, 11.5ms
image 12/15 /content/yolov5/PokerCard-1/valid/images/5_jpg.rf.69f056c6aa72c0ca249d788757af9d58.jpg: 640x640 1 nine, 1 ten, 11.5ms
image 13/15 /content/yolov5/PokerCard-1/valid/images/61_jpg.rf.91d4e2d1921faf8dd72e559e58d484c1.jpg: 640x640 1 queen, 10.8ms
image 14/15 /content/yolov5/PokerCard-1/valid/images/6_jpg.rf.0b24a3a53c2e4d5efdde266045de8f51.jpg: 640x640 1 ten, 10.8ms
image 15/15 /content/yolov5/PokerCard-1/valid/images/7_jpg.rf.1a2eac32811901e5e067b3d9d7b0bd01.jpg: 640x640 1 ten, 10.9ms
Speed: 0.5ms pre-process, 11.4ms inference, 1.1ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp
```

Actual Training in Colab (9): Yolo v5

❖ Visualize the inference results.

```
# 평가 결과를 시각화
import glob
from IPython.display import Image, display

for i, imageName in enumerate(glob.glob('runs/detect/exp/*.jpg')): #assuming
JPG
    if i > 10:
      break
    display(Image(filename=imageName))
    print("\n")
```



실행 중(13분 24초) <cell line: 9> › system() › system_compat() › run_command() › monitor_process() › pol

Actual Training in Colab (10): Yolo v5

❖ Convert the format into C++ libtorch.

```
# c++ libtorch 에서 사용하기 위한 포맷으로 변환
!python export.py --weights runs/train/exp/weights/best.pt --include torchscript
!ls runs/train/exp/weights/
```

```
export: data=data/coco128.yaml, weights=['runs/train/exp/weights/best.pt'], imgsz=[640, 640], batch_size=1, device=cpu, half=False, inplace=Fa
YOLOv5 🚀 v7.0-247-g3f02fde Python-3.10.12 torch-2.1.0+cu118 CPU

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs

PyTorch: starting from runs/train/exp/weights/best.pt with output shape (1, 25200, 11) (13.8 MB)

TorchScript: starting export with torch 2.1.0+cu118...
TorchScript: export success ✅ 3.7s, saved as runs/train/exp/weights/best.torchscript (27.3 MB)

Export complete (5.4s)
Results saved to /content/yolov5/runs/train/exp/weights
Detect:          python detect.py --weights runs/train/exp/weights/best.torchscript
Validate:        python val.py --weights runs/train/exp/weights/best.torchscript
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'runs/train/exp/weights/best.torchscript')
Visualize:       https://netron.app
best.pt  best.torchscript  last.pt
```

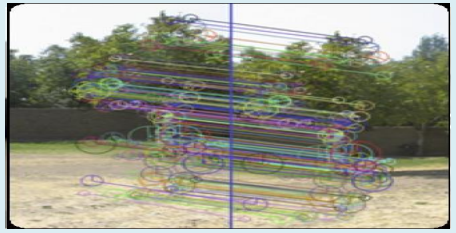
Actual Training in Colab (11): Yolo v5

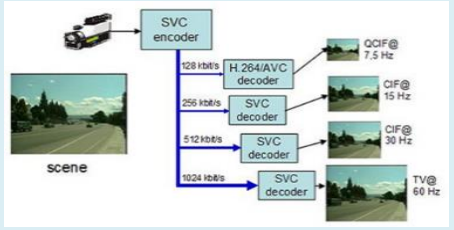
❖ Upload the transformed format (weight) to your Google drive.

```
# 학습된 weight 파일을 구글드라이브에 저장
from google.colab import drive
drive.mount('/content/gdrive')
%cp /content/yolov5/runs/train/exp/weights/best.torchscript
/content/gdrive/My\ Drive/yolov5_best.torchscript
```

```
Mounted at /content/gdrive
```

- ***Then check your google drive. You can see the trained weight for Yolo v5.***





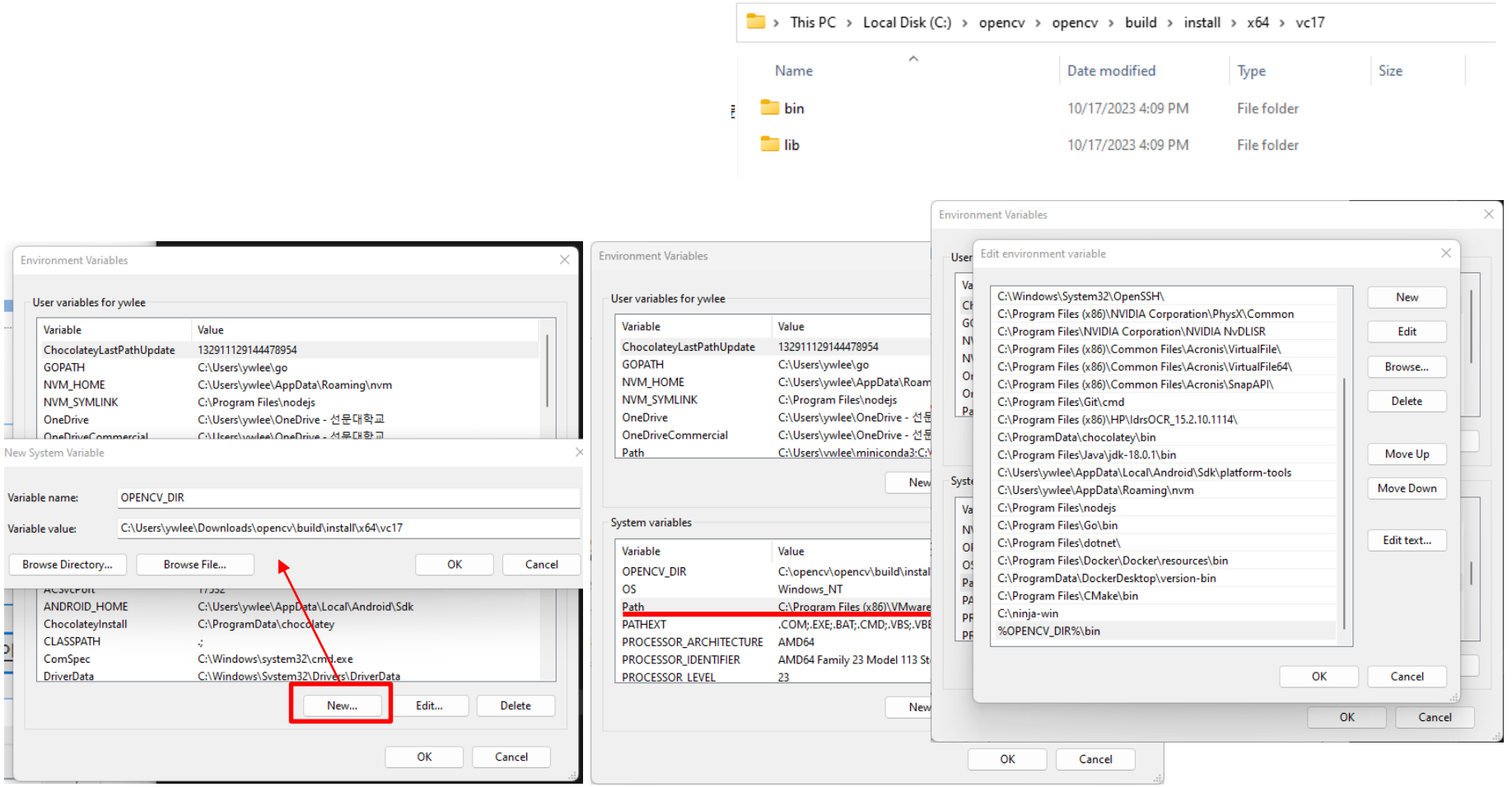
Contents

- My Github
- Preparation of My Dataset
- Actual Training Object Detection in Colab
- **To my local Yolo v5 model to detect objects**

Change your inference model into your local webcam...!!!! (1)

❖ OpenCV Environment Variables (We already set this...!!)

- Add (OPENCV_DIR) in Path environment
- Add %OPENCV_DIR%₩bin in Path variable.



Change your inference model into your local webcam...!!!! (2)

❖ Install LibTorch.

- LibTorch: Pytorch C++ version library
- Download it from https://pytorch.org/get-started/locally/ with following options.
- In this lecture, I explain **Debug version** as reference.

You need to download ver 1.13.0 which has been checked..!

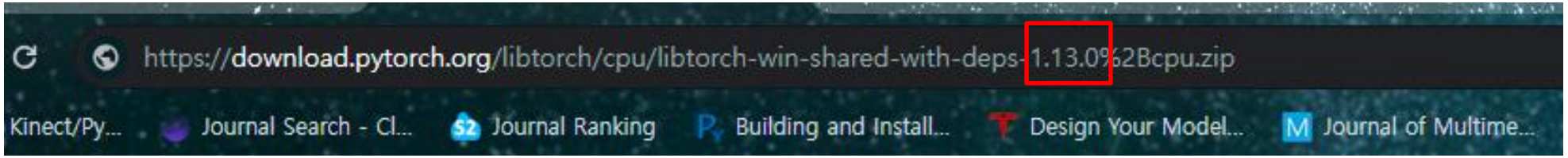
PyTorch Build	Stable (1.13.0)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.6	CUDA 11.7	~~ROCm 5.2~~	CPU
Run this Command:	Download here (Release version): https://download.pytorch.org/libtorch/cpu/libtorch-win-shared-with-deps-1.13.0%2Bcpu.zip Download here (Debug version): https://download.pytorch.org/libtorch/cpu/libtorch-win-shared-with-deps-debug-1.13.0%2Bcpu.zip			

Change your inference model into your local webcam...!!!! (2-1)

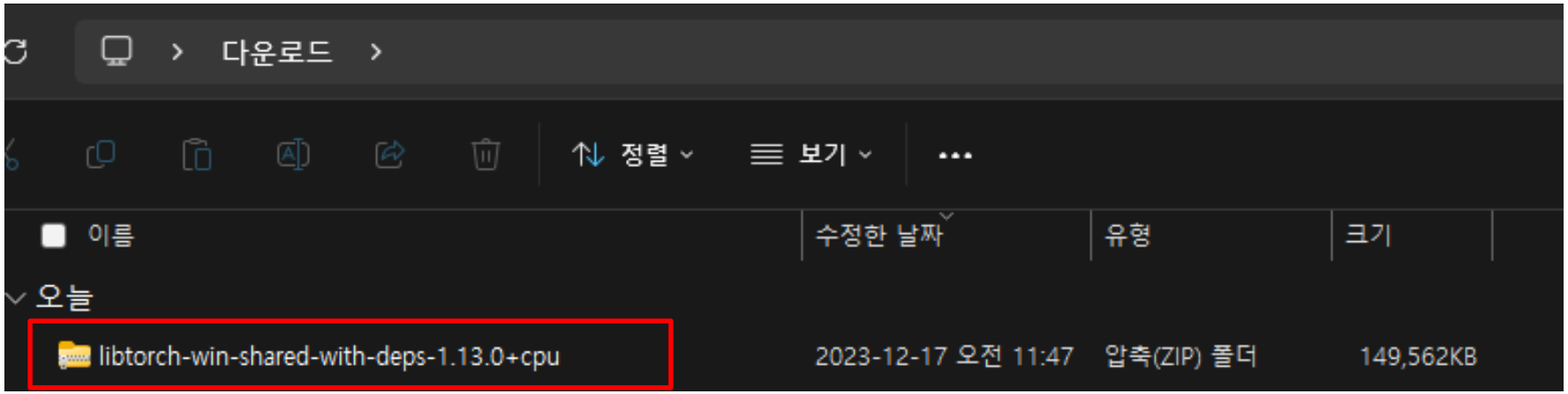
- Copy downlink link to URL of your browser as:

  https://download.pytorch.org/libtorch/cpu/libtorch-win-shared-with-deps-2.1.2%2Bcpu.zip

- Check your version and mode (release or debug mode as your VS project)
  - In url, **you just need to change the version 2.1.2 into 1.13.0**. That is,



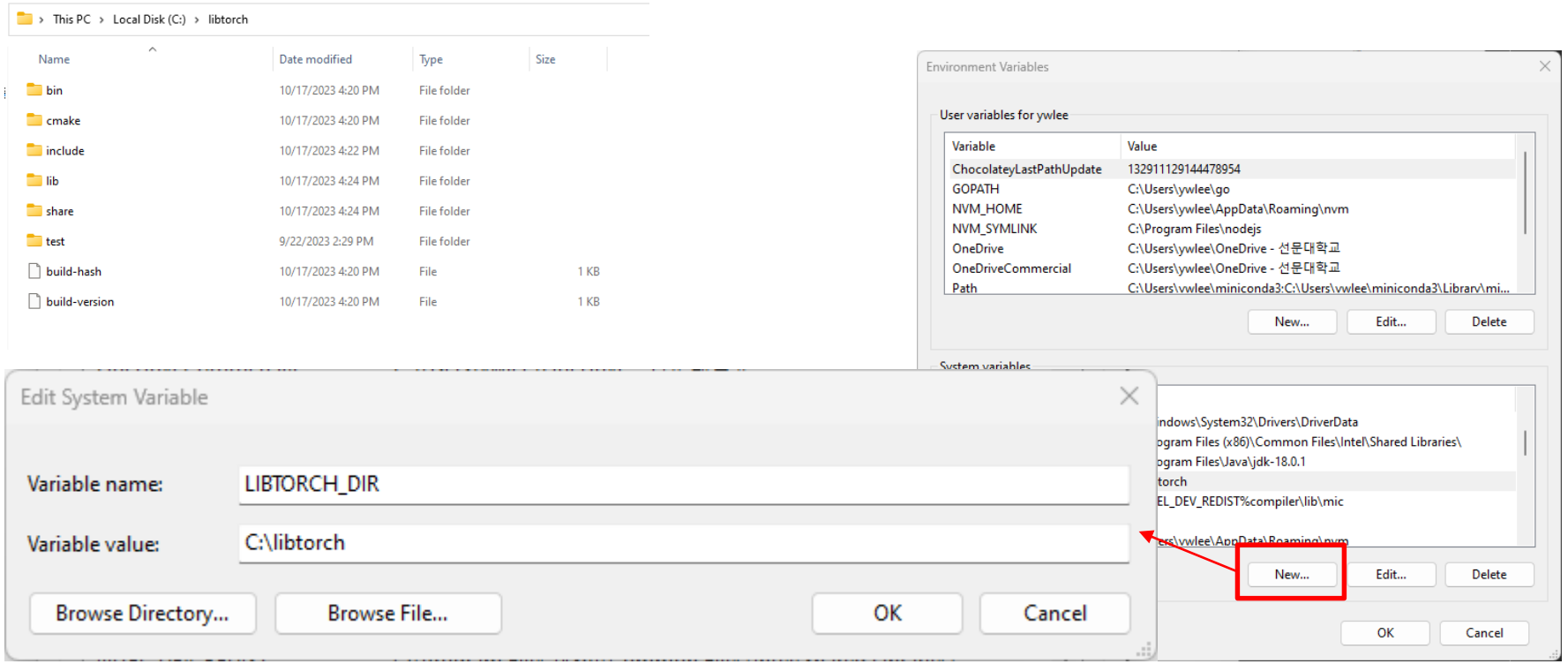
  - To download, you can just "enter".



본인의 프로젝트가 debug 모드인 경우 debug 모드 다운로드 링크를 복사해사 사용해야 함!

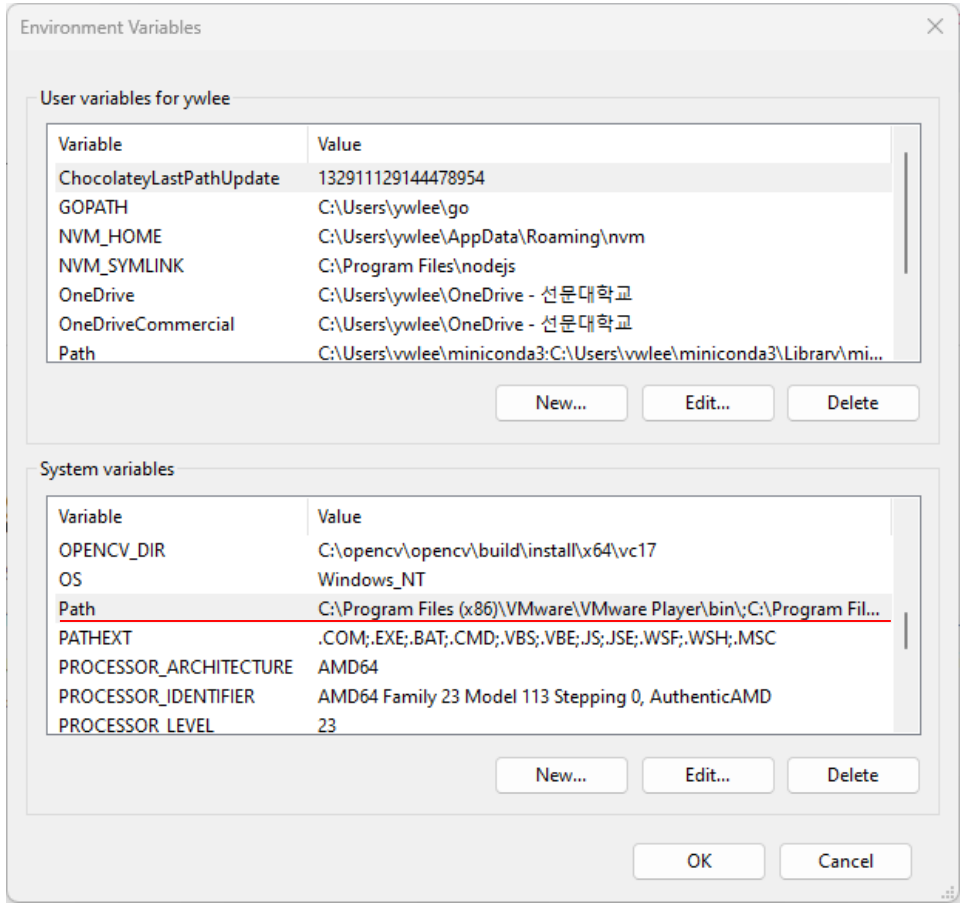
Change your inference model into your local webcam...!!!! (3)

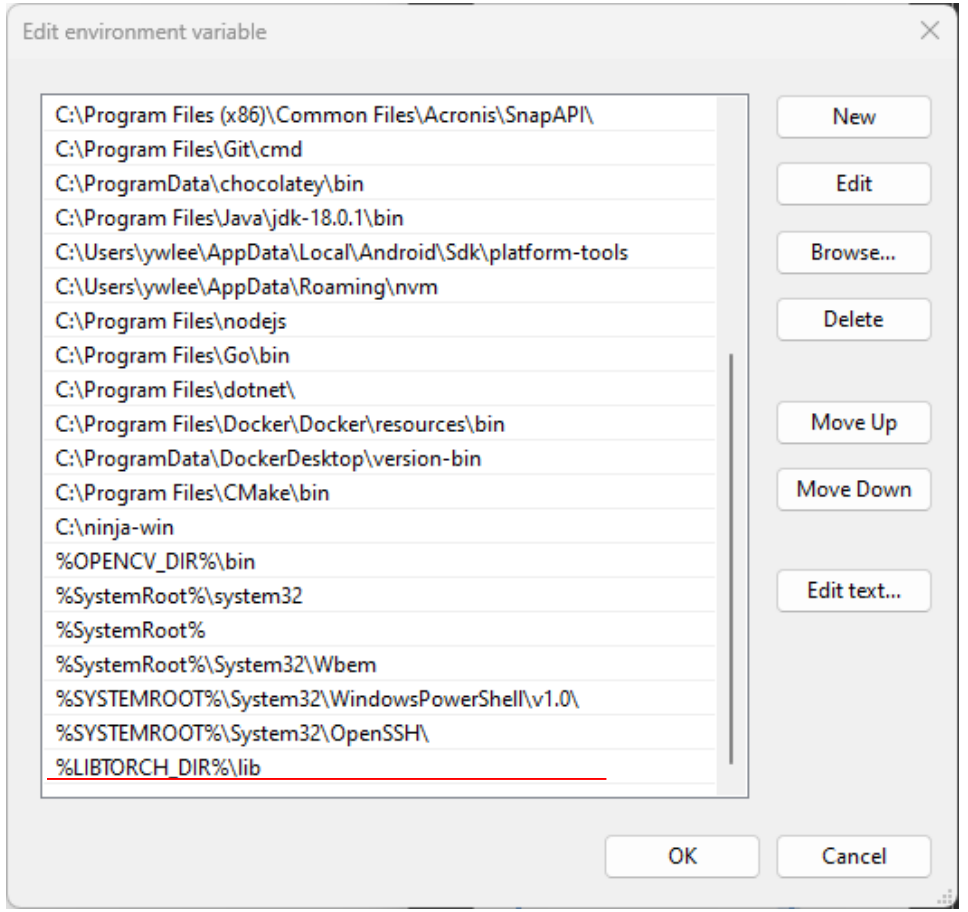
- Decompress the file and locate the folder at roor C:/ (C:/libtorch).
  - You can see the following figure.
- Then add “LIBTORCH_DIR” in the environment (system) variable.



Change your inference model into your local webcam...!!!! (4)

- Add "%LIBTORCH_DIR%₩lib" in system variables: path.

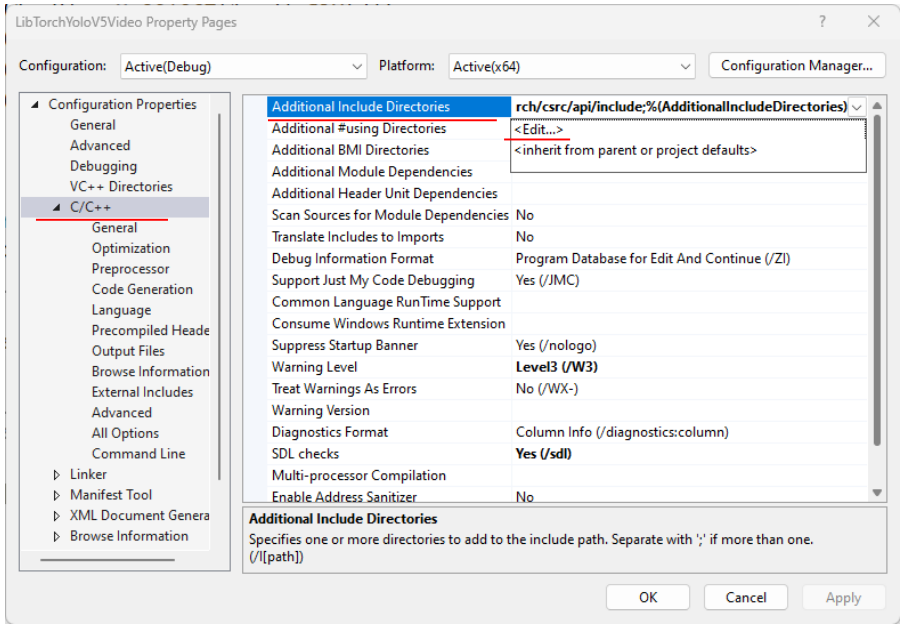


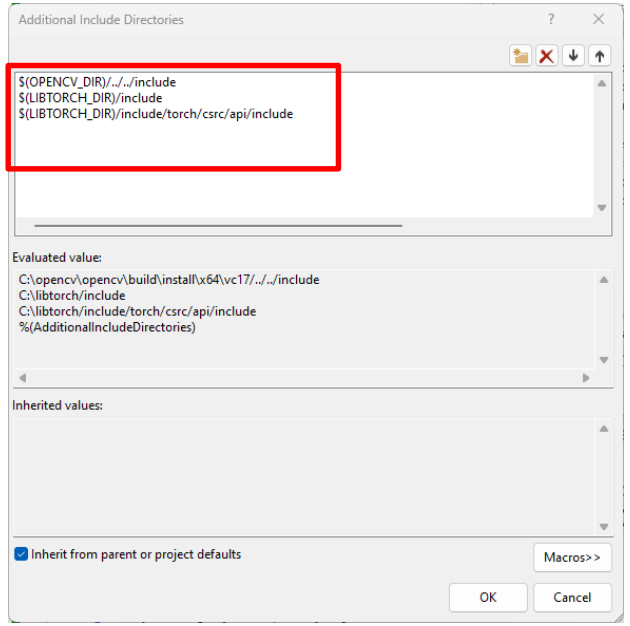


Change your inference model into your local webcam...!!!! (5)

❖ **LibTorch** Configuration in **Visual Studio 2022**.

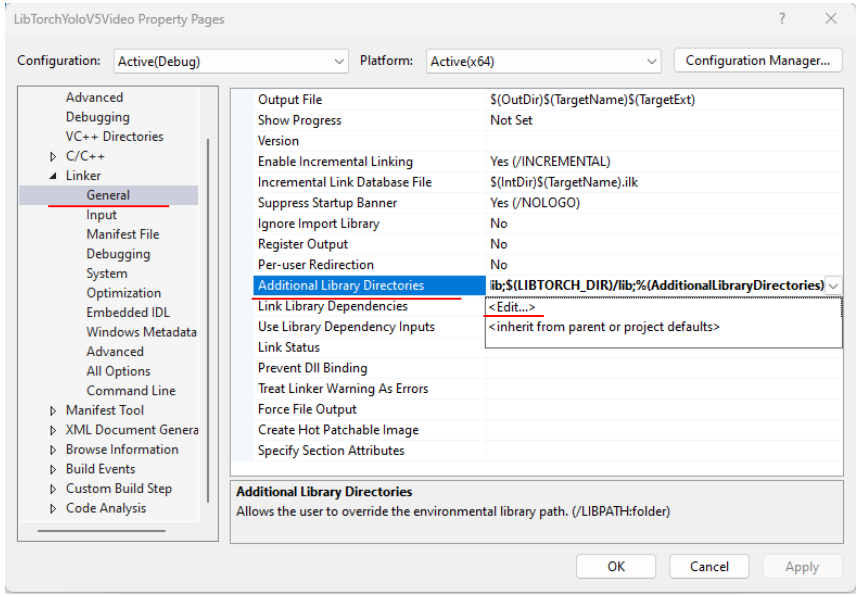
- Visual Studio → New project → add (.cpp) file.
- Change "Properties" of my project (solution)
  - Set the OpenCV & LibTorch path.
- Item: **C/C++ → Additional Include Directories**
  - $(OPENCV_DIR)/../../include ← **Already set...!!!**
  - $(LIBTORCH_DIR)/include
  - $(LIBTORCH_DIR)/include/torch/csrc/api/include

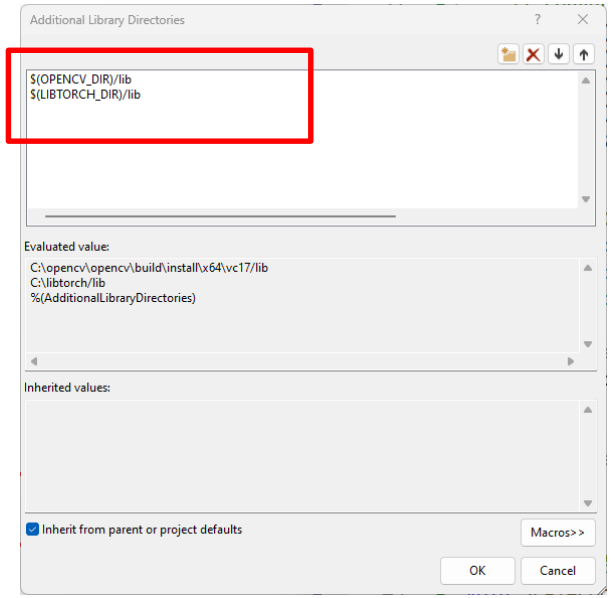




Change your inference model into your local webcam...!!!! (6)

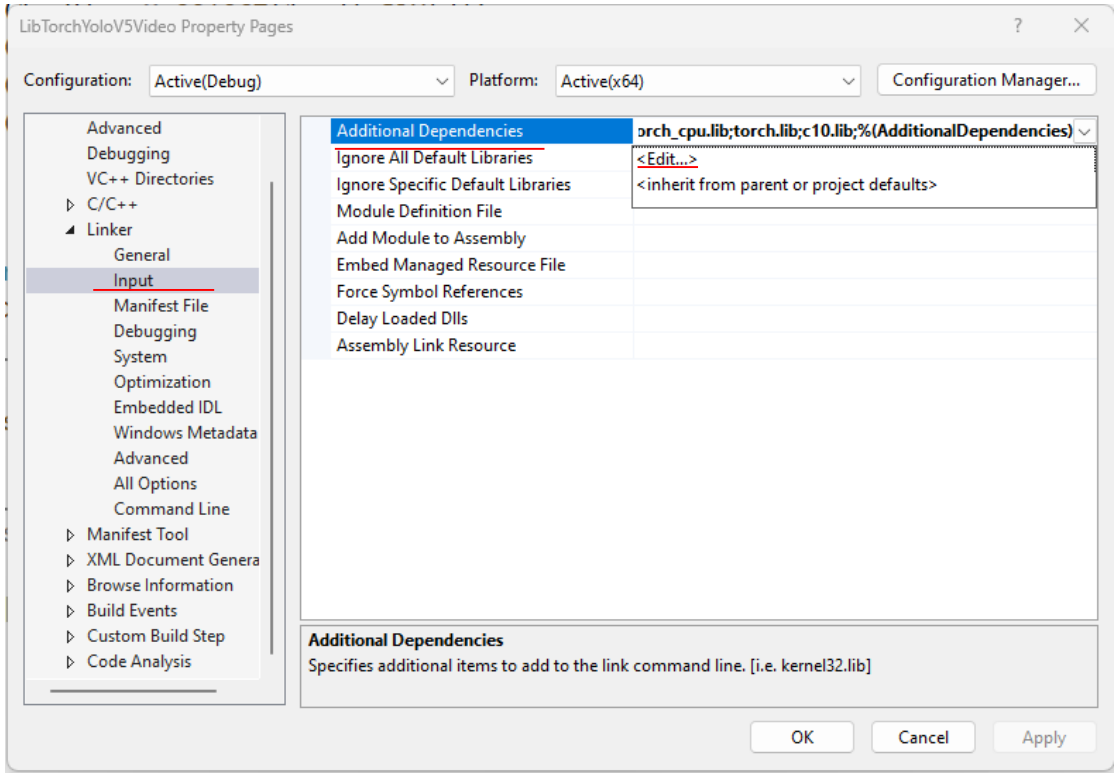
- Linker → General → Additional Library Directories
  - $(OPENCV_DIR)/lib ← **Already set...!!!**
  - $(LIBTORCH_DIR)/lib

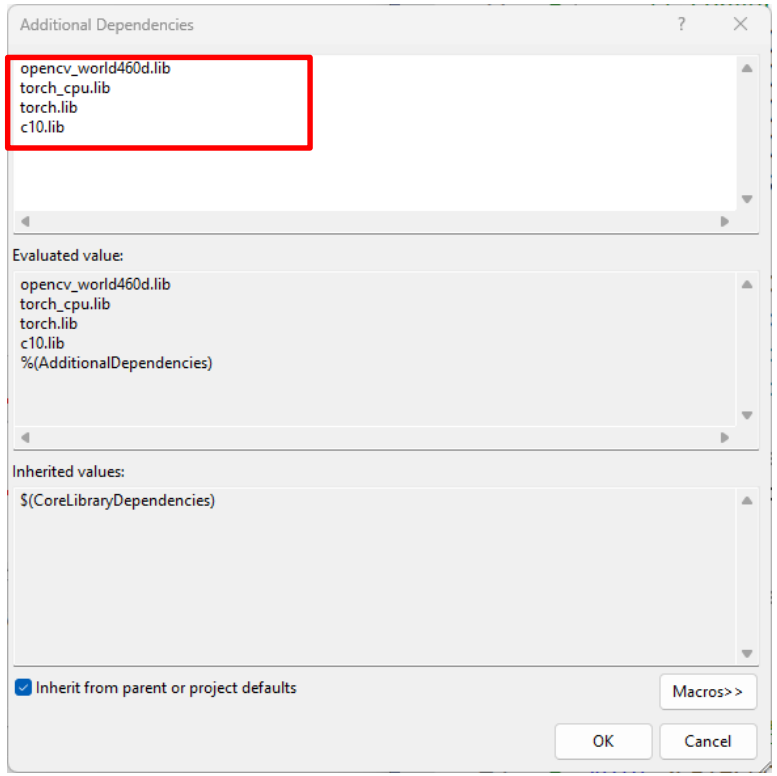




Change your inference model into your local webcam...!!!! (7)

- Linker → Input → Additional Dependencies
  - opencv_worldXXXd.lib (XXX is your version number) ← **Already set...!!!**
  - torch_cpu.lib
  - torch.lib
  - c10.lib





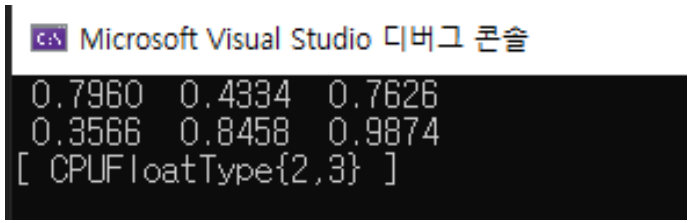
Change your inference model into your local webcam...!!!! (8)

❖ LibTorch Test

▪ Test Code sample

An example for displaying some random tensors in the screen...!!!!

```
#include <torch/script.h>
#include <iostream>

using namespace std;

int main()
{
	at::Tensor tensor = torch::rand({ 2,3 });
	cout << tensor << endl;

	return 0;
}
```

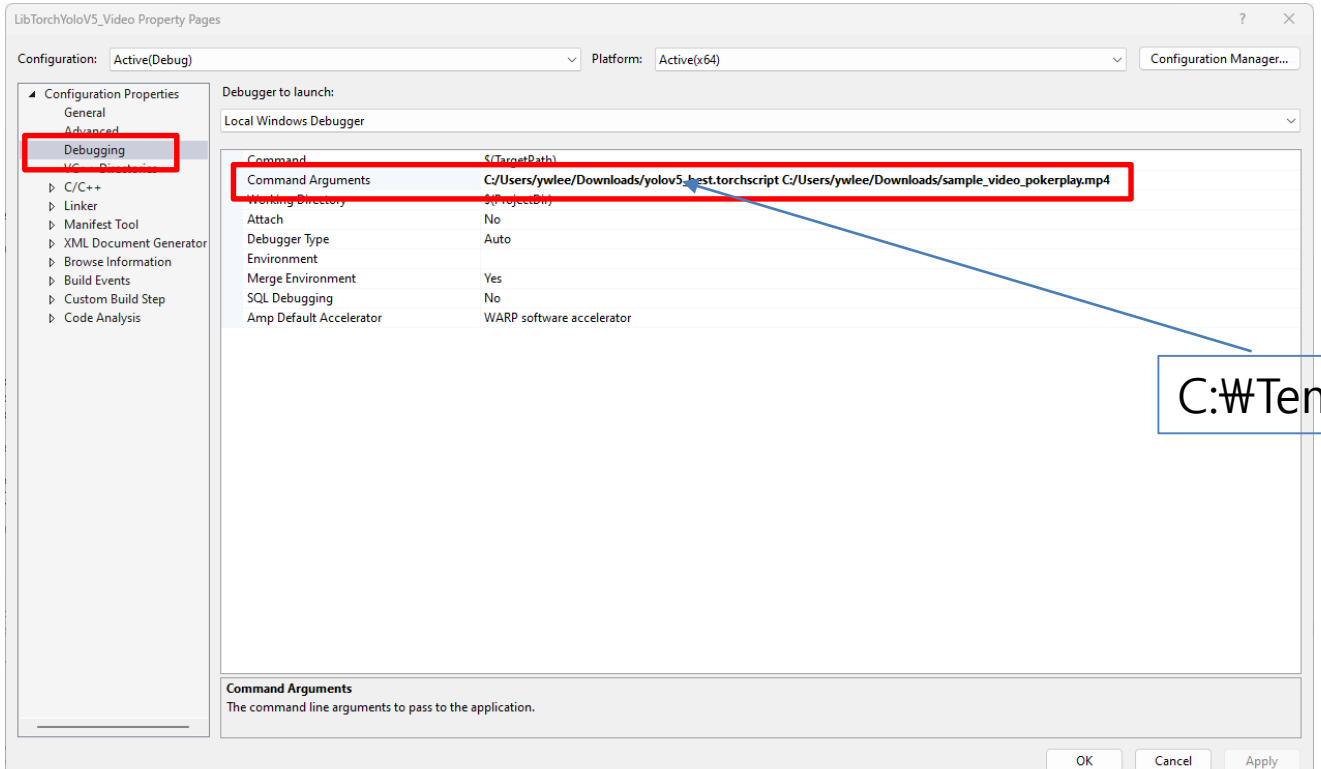


If you got the result, your torch lib. works well..!!

Yolo v5 Application Implementation (1)

❖ 실습에 필요한 파일
- Colab 을 통해 학습된 weight 파일 (yolov5_best.torchscript)
- 테스트용 포커이미지

❖ Visual Studio 프로젝트 속성에서 매개변수 전달 방법
- Configuration Properties → Debugging → Command Arguments
- 매개변수로 전달할 경로를 입력, 2개 이상일 경우 공백(스페이스바) 로 구분



C:₩Temp₩yolov5_best_2023.torchscript

Yolo v5 Application Implementation (1)

❖ **Make your C++ Code as:**

- **Define header files.**

```
#include <opencv2/opencv.hpp>
#include <torch/script.h>
#include <iostream>
#include <vector>

using namespace std;
using namespace cv;
```

Yolo v5 Application Implementation (2)

- Structure for object recognition results (객체인식 결과 정보를 담는 구조체 정의)

```
enum Det {
        tl_x = 0,
        tl_y = 1,
        br_x = 2,
        br_y = 3,
        score = 4,
        class_idx = 5
};

struct Detection {
        cv::Rect bbox;
        float score;
        int class_idx;
};
```

Yolo v5 Application Implementation (3)

- Function for changing the coordinate to OpenCV format.
  (YOLO 형식의 좌표를 OpenCV 형식의 좌표로 변환하는 함수)

```
torch::Tensor xywh2xyxy(const torch::Tensor& x) {
        auto y = torch::zeros_like(x);
        y.select(1, Det::tl_x) = x.select(1, 0) - x.select(1, 2).div(2);
        y.select(1, Det::tl_y) = x.select(1, 1) - x.select(1, 3).div(2);
        y.select(1, Det::br_x) = x.select(1, 0) + x.select(1, 2).div(2);
        y.select(1, Det::br_y) = x.select(1, 1) + x.select(1, 3).div(2);
        return y;
}
```

Yolo v5 Application Implementation (4)

- Function for changing the detection results to Detection structure.
  (텐서 자료형의 객체인식 결과를 Detection 구조체로 변환하는 함수)

```
void Tensor2Detection(const at::TensorAccessor<float, 2>& offset_boxes,
        const at::TensorAccessor<float, 2>& det,
        std::vector<cv::Rect>& offset_box_vec,
        std::vector<float>& score_vec)
{
        for (int i = 0; i < offset_boxes.size(0); i++) {
                offset_box_vec.emplace_back(
                        cv::Rect(cv::Point(offset_boxes[i][Det::tl_x],
                                offset_boxes[i][Det::tl_y]),
                                cv::Point(offset_boxes[i][Det::br_x],
                                        offset_boxes[i][Det::br_y])));
                score_vec.emplace_back(det[i][Det::score]);
        }
}
```

Yolo v5 Application Implementation (5)

- Function for changing the object detection results to the original image coordinate. (객체인식 결과의 좌표를 원본 이미지의 좌표로 환산하는 함수)

```
void ScaleCoordinates(std::vector<Detection>& data, float pad_w, float pad_h,
        float scale, const cv::Size& img_shape)
{
        auto clip = [](float n, float lower, float upper)
        {
                return std::max(lower, std::min(n, upper));
        };
        std::vector<Detection> detections;
        for (auto& i : data) {
                float x1 = (i.bbox.tl().x - pad_w) / scale;  // x padding
                float y1 = (i.bbox.tl().y - pad_h) / scale;  // y padding
                float x2 = (i.bbox.br().x - pad_w) / scale;  // x padding
                float y2 = (i.bbox.br().y - pad_h) / scale;  // y padding

                x1 = clip(x1, 0, img_shape.width);
                y1 = clip(y1, 0, img_shape.height);
                x2 = clip(x2, 0, img_shape.width);
                y2 = clip(y2, 0, img_shape.height);

                i.bbox = cv::Rect(cv::Point(x1, y1), cv::Point(x2, y2));
        }
}
```

Yolo v5 Application Implementation (6)

- Function that converts input images to the fixed input size of the YOLO model and calculates the difference.
  (입력 이미지를 YOLO 모델의 고정된 입력 크기로 변환 후 차이 값을 계산하는 함수)

```
std::vector<float> LetterboxImage(const cv::Mat& src,
          cv::Mat& dst, const cv::Size& out_size)
{
          auto in_h = static_cast<float>(src.rows);
          auto in_w = static_cast<float>(src.cols);
          float out_h = out_size.height;
          float out_w = out_size.width;

          float scale = std::min(out_w / in_w, out_h / in_h);

          int mid_h = static_cast<int>(in_h * scale);
          int mid_w = static_cast<int>(in_w * scale);

         int top = (static_cast<int>(out_h) - mid_h) / 2;
          int down = (static_cast<int>(out_h) - mid_h + 1) / 2;
          int left = (static_cast<int>(out_w) - mid_w) / 2;
          int right = (static_cast<int>(out_w) - mid_w + 1) / 2;

          cv::copyMakeBorder(dst, dst, top, down, left, right,
                    cv::BORDER_CONSTANT, cv::Scalar(114, 114, 114));

          std::vector<float> pad_info{ static_cast<float>(left),
                    static_cast<float>(top), scale };
          return pad_info;
}
```

Yolo v5 Application Implementation (7)

- Write main () part...!!! (메인함수 작성)

```
int main(int argc, char** argv) {
        if (argc < 3) {
                cout << "Wrong path of weight and image" << endl;
                return -1;
        }

        float conf_thres = 0.25; // 검출 정확도 기준
        float iou_thres = 0.45;  // BBox 정확도 기준

        // 클래스 이름
        std::vector<std::string> class_names = {
                "ace",
                "jack",
                "king",
                "nine",
                "queen",
                "ten"
        };
```

Yolo v5 Application Implementation (8)

- (continue)

```
// 파이토치 모델 객체
torch::jit::script::Module model;
try {
        // 첫번째 인자 (argv[1]) 는 weight 파일 경로
        model = torch::jit::load(argv[1]);
        cout << "YOLO model loaded" << endl;
}
catch (const c10::Error& e) {
        cerr << "cannot load the weight file" << endl;
        cerr << e.backtrace() << endl;
        return -1;
}
// 두번째 인자 (argv[2]) 는 이미지 파일 경로
cv::Mat img = imread(argv[2], IMREAD_COLOR);
cv::Mat img_input = img.clone(); // 복사

// 이미지 크기를 640 x 640 크기로 일괄 변경
std::vector<float> pad_info = LetterboxImage(
        img_input, img_input, cv::Size(640, 640));
const float pad_w = pad_info[0];
const float pad_h = pad_info[1];
const float scale = pad_info[2];
```

Yolo v5 Application Implementation (9)

- (continue)

```
cv::resize(img_input, img_input,
        cv::Size(640, 640)); // 크기 변경
cv::cvtColor(img_input, img_input,
        cv::COLOR_BGR2RGB); // BGR -> RGB
img_input.convertTo(img_input, CV_32FC3,
        1.0f / 255.0f); // normalization 1/255

// 텐서 객체 생성
auto tensor_img = torch::from_blob(img_input.data,
        { 1, img_input.rows, img_input.cols,
        img_input.channels() }, torch::kFloat);
// BHWC -> BCHW (Batch, Channel, Height, Width)
tensor_img = tensor_img.permute({ 0, 3, 1, 2 }).contiguous();
std::vector<torch::jit::IValue> inputs;
inputs.emplace_back(tensor_img);
```

Yolo v5 Application Implementation (10)

- (continue)

```
// 추론
torch::jit::IValue output = model.forward(inputs);
auto detections = output.toTuple()->elements()[0].toTensor();
constexpr int item_attr_size = 5;
int batch_size = detections.size(0); // 배치 크기
auto num_classes = detections.size(2)
        - item_attr_size; // 클래스 수

// confidence > threshold 조건에 해당하는 후보만 남김
auto conf_mask = detections.select(2, 4).ge(conf_thres).unsqueeze(2);
auto det = torch::masked_select(detections[0], conf_mask[0]).view(
        { -1, num_classes + item_attr_size });

// det 크기가 0 이면 검출된 결과가 존재하는 않는 것
if (0 == det.size(0)) {
        return -1;
}
```

Yolo v5 Application Implementation (11)

- (continue)

```
std::vector<std::vector<Detection>> result;
result.reserve(batch_size);

// 스코어 계산 = obj_conf * cls_conf, similar to x[:, 5:] *= x[:, 4:5]
det.slice(1, item_attr_size, item_attr_size + num_classes)
        *= det.select(1, 4).unsqueeze(1);

// 좌표 계산 (center x, center y, width, height) to (x1, y1, x2, y2)
torch::Tensor box = xywh2xyxy(det.slice(1, 0, 4));

// 클래스 스코어가 가장 높은 것만 남김
std::tuple<torch::Tensor, torch::Tensor> max_classes =
        torch::max(det.slice(1, item_attr_size,
                item_attr_size + num_classes), 1);
```

Yolo v5 Application Implementation (12)

- (continue)

```
// class score
auto max_conf_score = std::get<0>(max_classes);
// index
auto max_conf_index = std::get<1>(max_classes);

max_conf_score = max_conf_score.to(torch::kFloat).unsqueeze(1);
max_conf_index = max_conf_index.to(torch::kFloat).unsqueeze(1);

// shape: n * 6, top-left x/y (0,1),
// bottom-right x/y (2,3), score(4),
// class index(5)
det = torch::cat({ box.slice(1, 0, 4),
        max_conf_score, max_conf_index }, 1);

// NMS 계산을 하기 위해 검출된 박스 결과를 배치 단위화
constexpr int max_wh = 4096;
auto c = det.slice(1, item_attr_size,
        item_attr_size + 1) * max_wh;
auto offset_box = det.slice(1, 0, 4) + c;

std::vector<cv::Rect> offset_box_vec;
std::vector<float> score_vec;
```

Yolo v5 Application Implementation (13)

- (continue)

```
// copy data back to cpu
auto offset_boxes_cpu = offset_box.cpu();
auto det_cpu = det.cpu();
const auto& det_cpu_array = det_cpu.accessor<float, 2>();

// 텐서 객체를 Detection 객체로 변경
Tensor2Detection(offset_boxes_cpu.accessor<float, 2>(),
        det_cpu_array, offset_box_vec, score_vec);

// NMS: Non-Maximum Suppression 실행
std::vector<int> nms_indices;
cv::dnn::NMSBoxes(offset_box_vec, score_vec,
        conf_thres, iou_thres, nms_indices);
```

Yolo v5 Application Implementation (14)

- (continue)

```
// NMS 계산 후의 박스들의 위치 좌표 저장
std::vector<Detection> det_vec;
for (int index : nms_indices) {
        Detection t;
        const auto& b = det_cpu_array[index];
        t.bbox = cv::Rect(cv::Point(b[Det::tl_x], b[Det::tl_y]),
                cv::Point(b[Det::br_x], b[Det::br_y]));
        t.score = det_cpu_array[index][Det::score];
        t.class_idx = det_cpu_array[index][Det::class_idx];
        det_vec.emplace_back(t);
}

// 원본 이미지 크기에 맞추어 좌표를 재계산
ScaleCoordinates(det_vec, pad_w, pad_h, scale, img.size());
result.emplace_back(det_vec);
```

Yolo v5 Application Implementation (15)

- (continue)

```
// 이미지에 검출 결과를 표시
for (const auto& detection : result[0]) {
        const auto& box = detection.bbox;
        float score = detection.score;
        int class_idx = detection.class_idx;

        cv::rectangle(img, box, cv::Scalar(0, 0, 255), 2);
        std::stringstream ss;
        ss << std::fixed << std::setprecision(2) << score;
        std::string s = class_names[class_idx] + " " + ss.str();

        auto font_face = cv::FONT_HERSHEY_DUPLEX;
        auto font_scale = 1.0;
        int thickness = 1;
        int baseline = 0;
        auto s_size = cv::getTextSize(s, font_face,
                font_scale, thickness, &baseline);
        cv::rectangle(img,
                cv::Point(box.tl().x, box.tl().y - s_size.height - 5),
                cv::Point(box.tl().x + s_size.width, box.tl().y),
                cv::Scalar(0, 0, 255), -1);
        cv::putText(img, s, cv::Point(box.tl().x, box.tl().y - 5),
                font_face, font_scale,
                cv::Scalar(255, 255, 255), thickness);
}
```

Yolo v5 Application Implementation (16)

- (continue)

```
        // 화면 출력
        cv::namedWindow("Result", WINDOW_AUTOSIZE);
        cv::imshow("Result", img);
        cv::waitKey(0);

        return 0;
} //← 메인 함수의 끝
```

실행 방법: CVApp.exe C:₩Temp₩yolov5_best_2023.torchscript IMG_2384.JPG
(CV.exe Yolov5 weight 경로 테스트이미지)

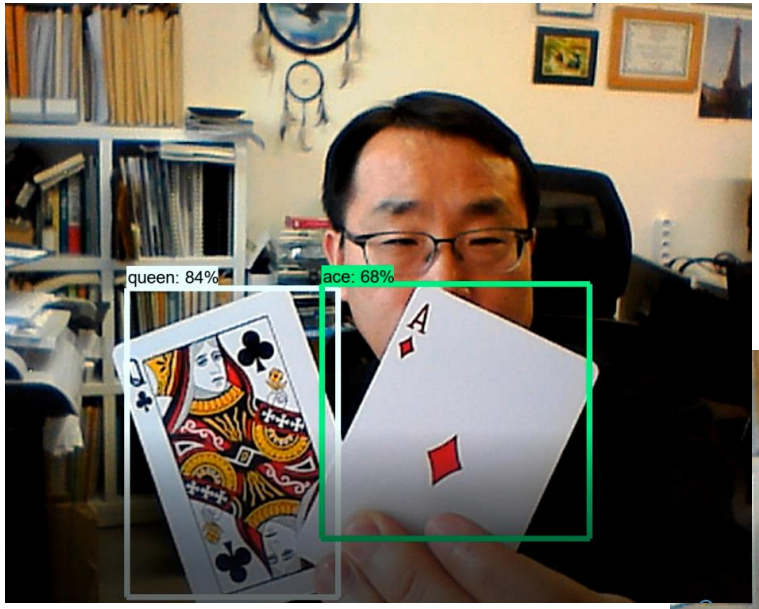
Yolo v5 Application Implementation (17)

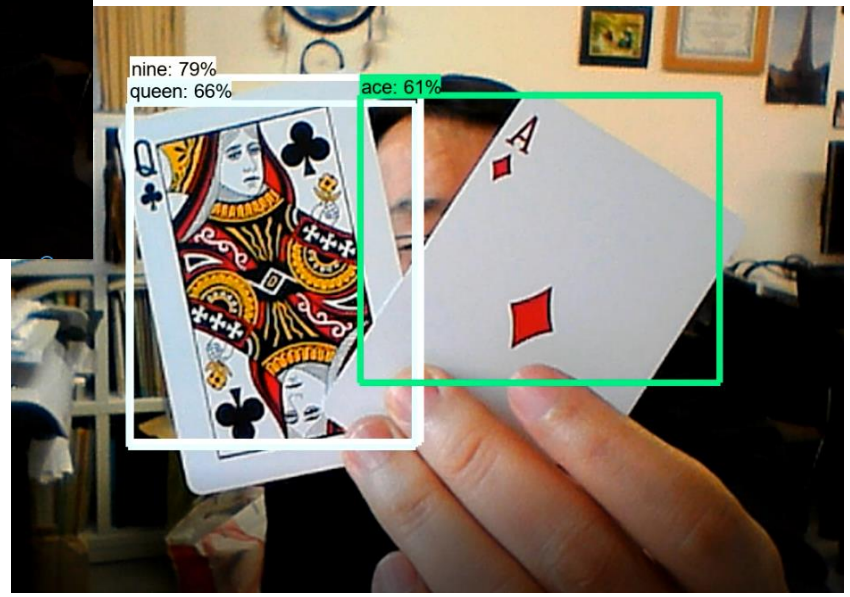
❖ Run your Yolo v5 model to infer your input image as:

```
C:\Users\SM-PC\Lectures\2023\ComputerVision\CVApp_Yolov5ObjectDetection\x64\Release>CVApp.exe C:\Temp\yolov5_best_2023.t
orchscript IMG_2391.JPG
YOLO model loaded
```

Yolo v5 Application Implementation (18): With your local webcam...!!!!

❖ Change the input as your webcam. Then run your inference as:





Yolo v5 Application Implementation (19): With your local webcam...!!!!

Thank you for your attention.!!!
QnA

http://ivpl.sookmyung.ac.kr