IVPL
Intelligent Vision Processing Lab

Computer Vision

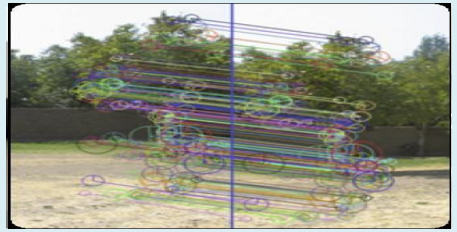
Deep Learning Basics
(#23: Tensorflow 2.6 Object Detection Model Training in Colab)

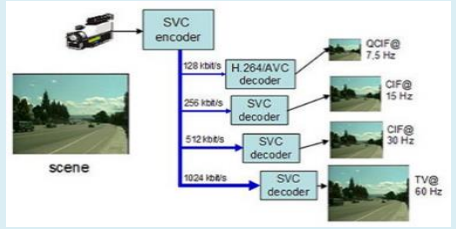
2023 Autumn

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



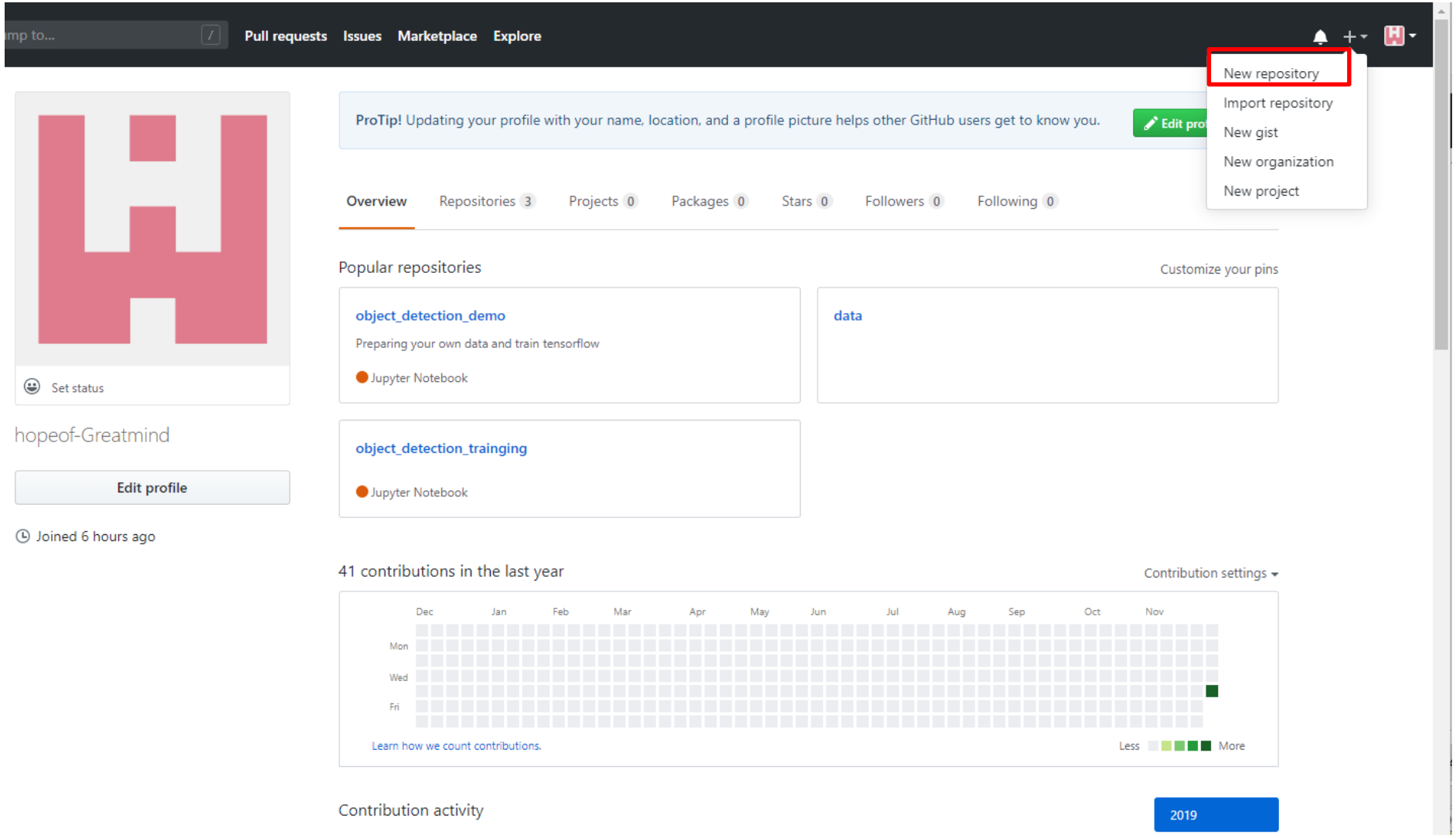


Contents

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

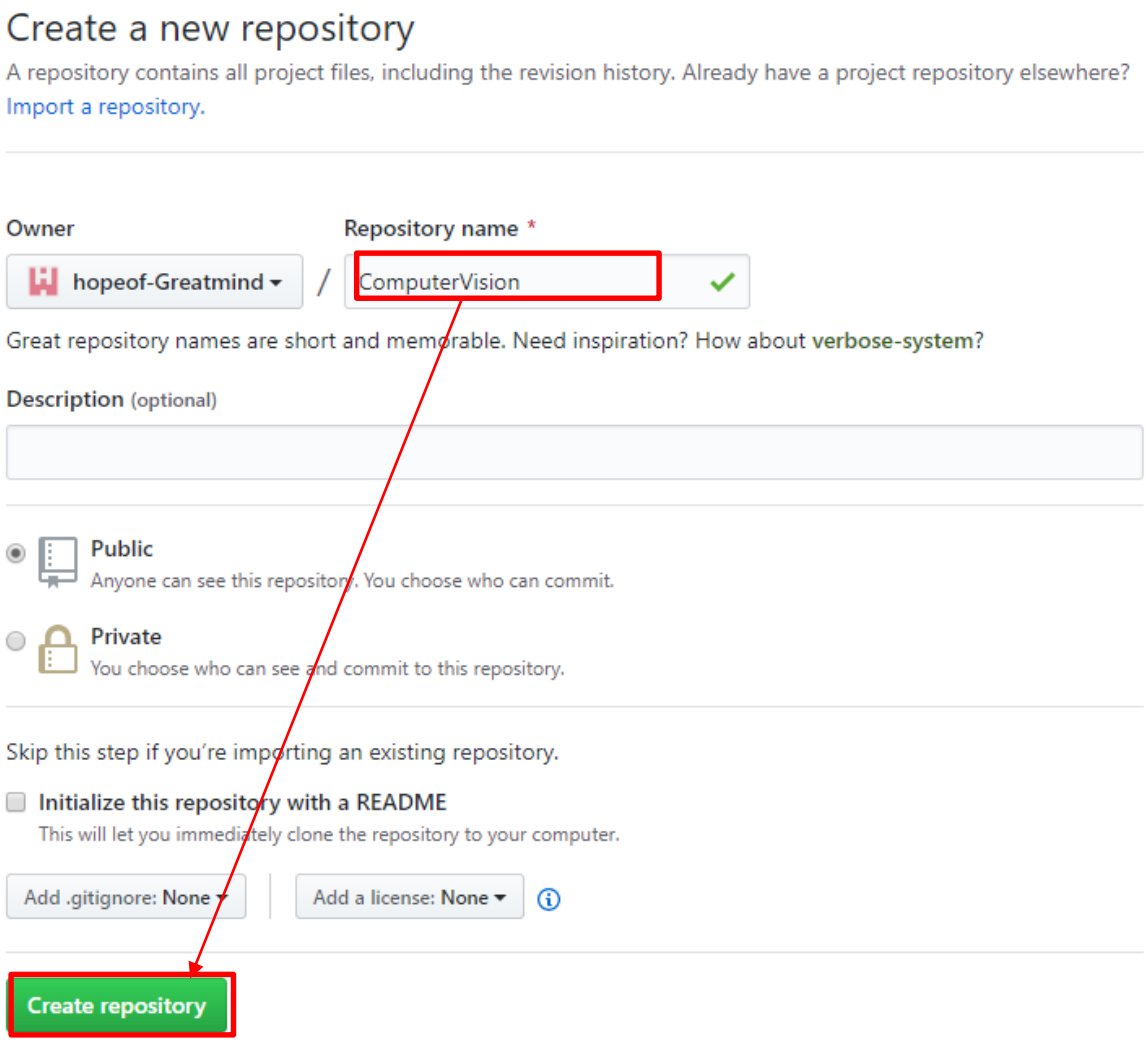
My Github (1)

- Create your account in GitHub
- Click “New repository”



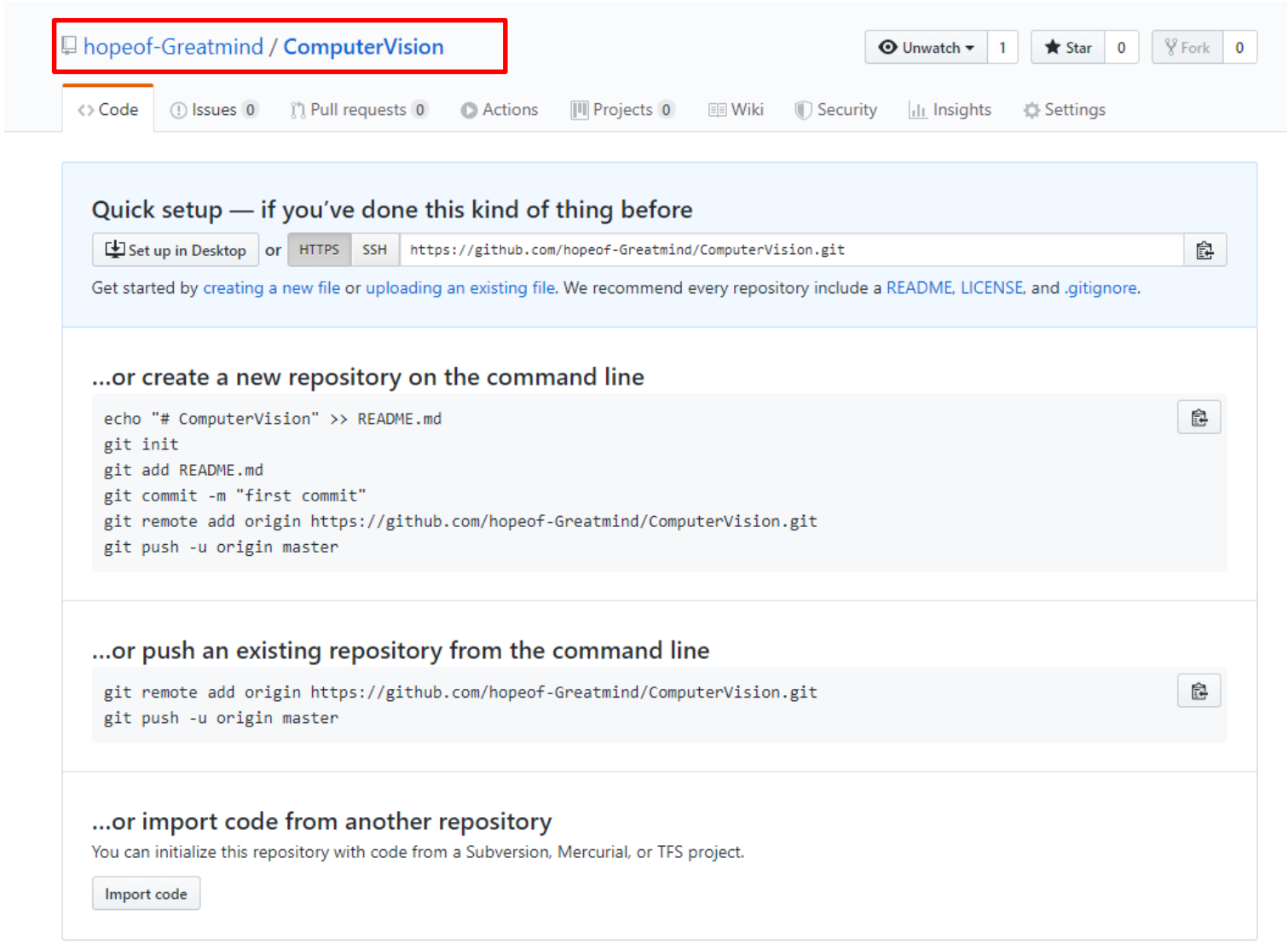
My Github (2)

- Make “your folder” for using and select “Create repository”.



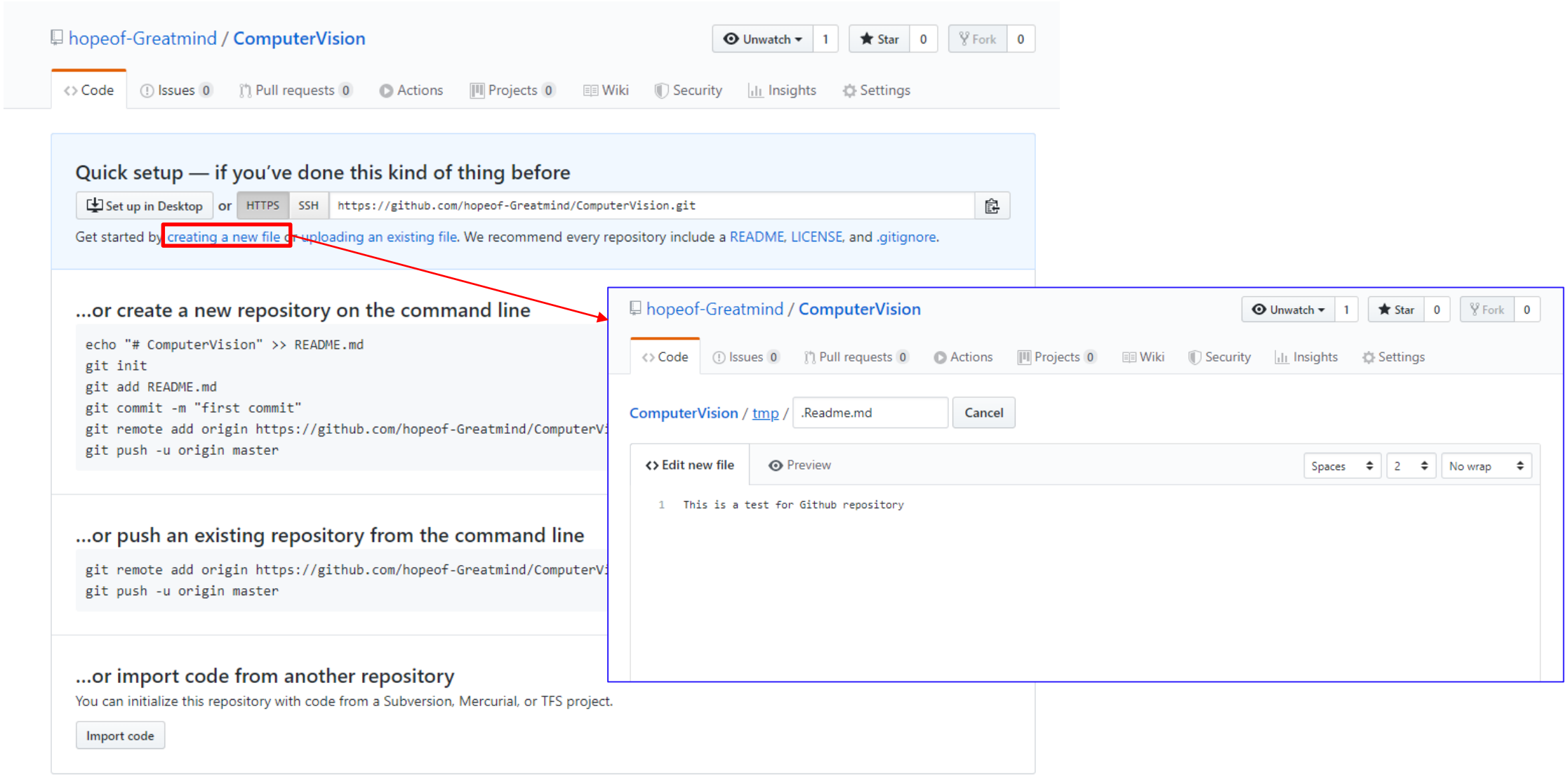
My Github (3)

- You can see your new repository without anything.



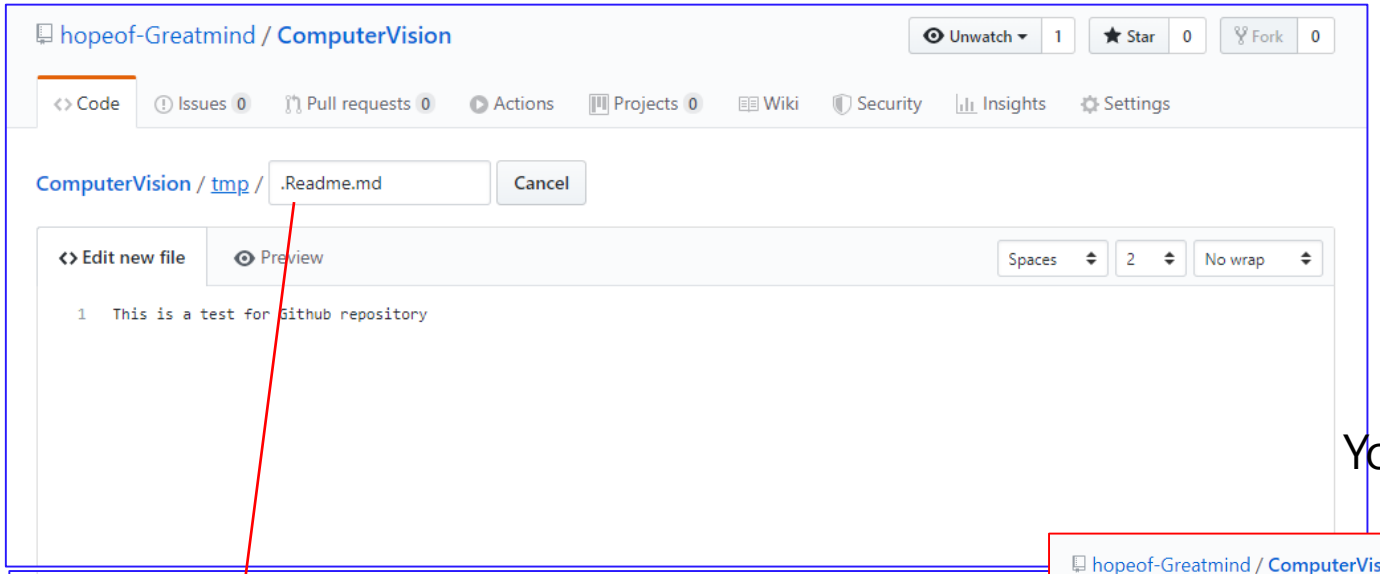
My Github (4)

❖ How to make new folder in my repository?: Click “creating a new file”

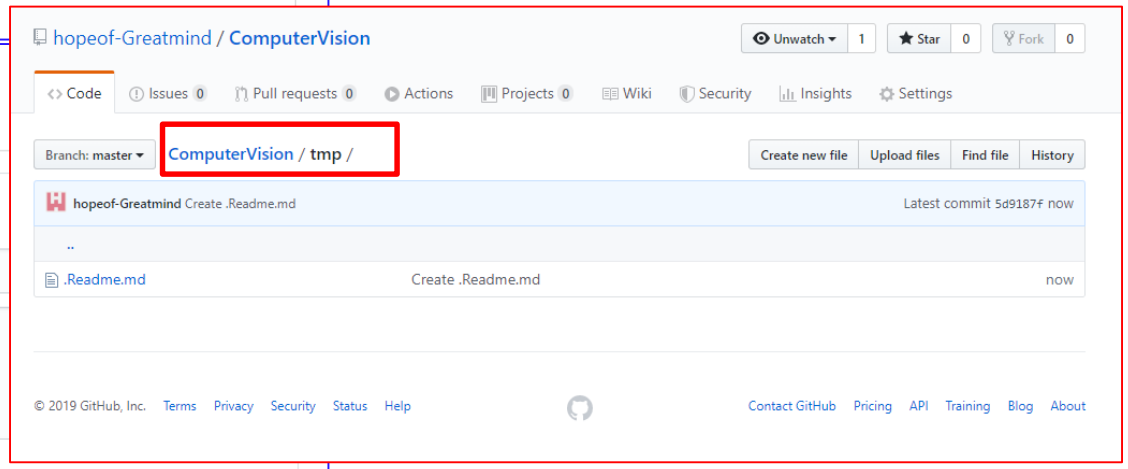


My Github (5)

- If you use "/tmp/", then it will be folder. Otherwise, it will be a file in your folder.



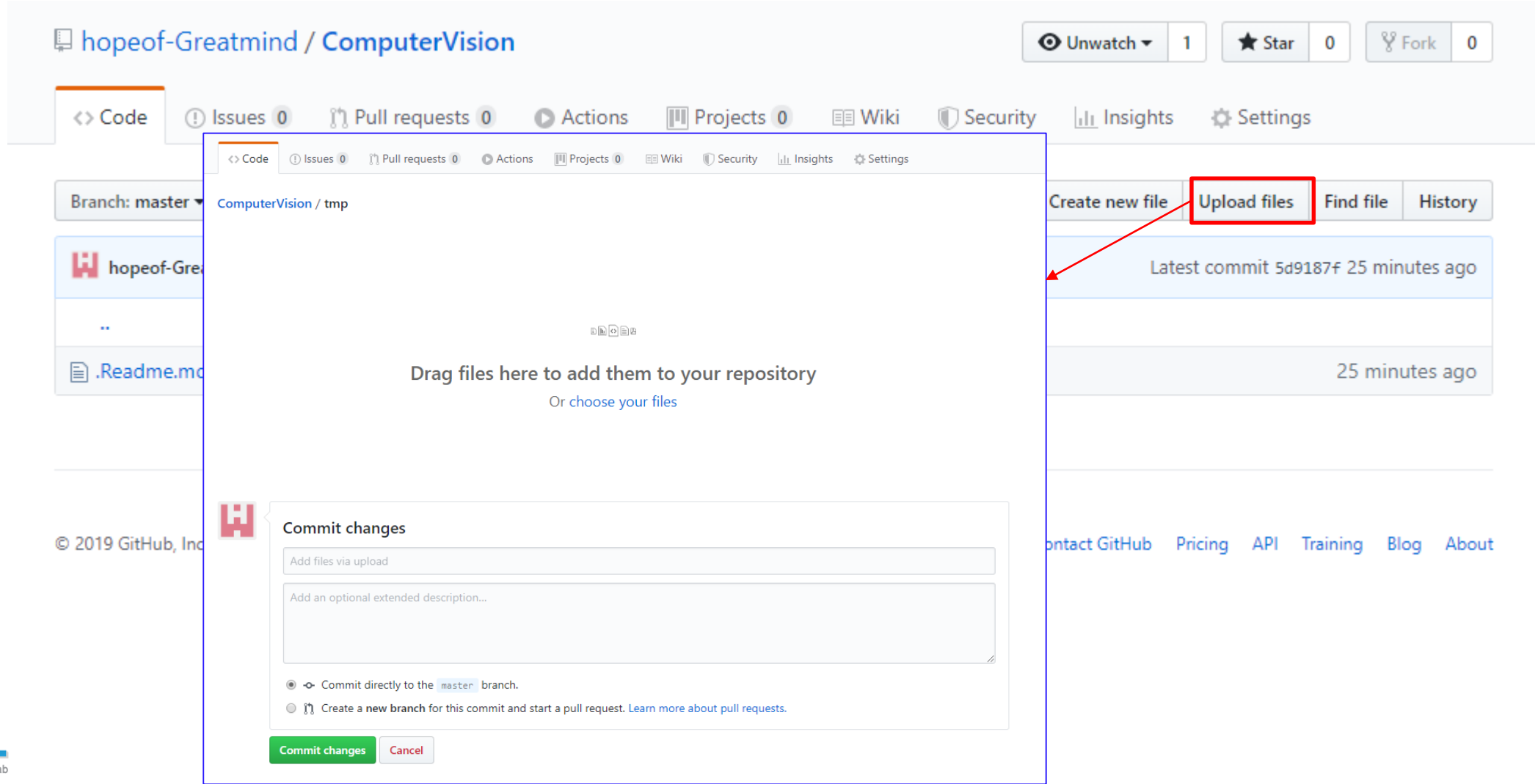
You can see your older



My Github (6)

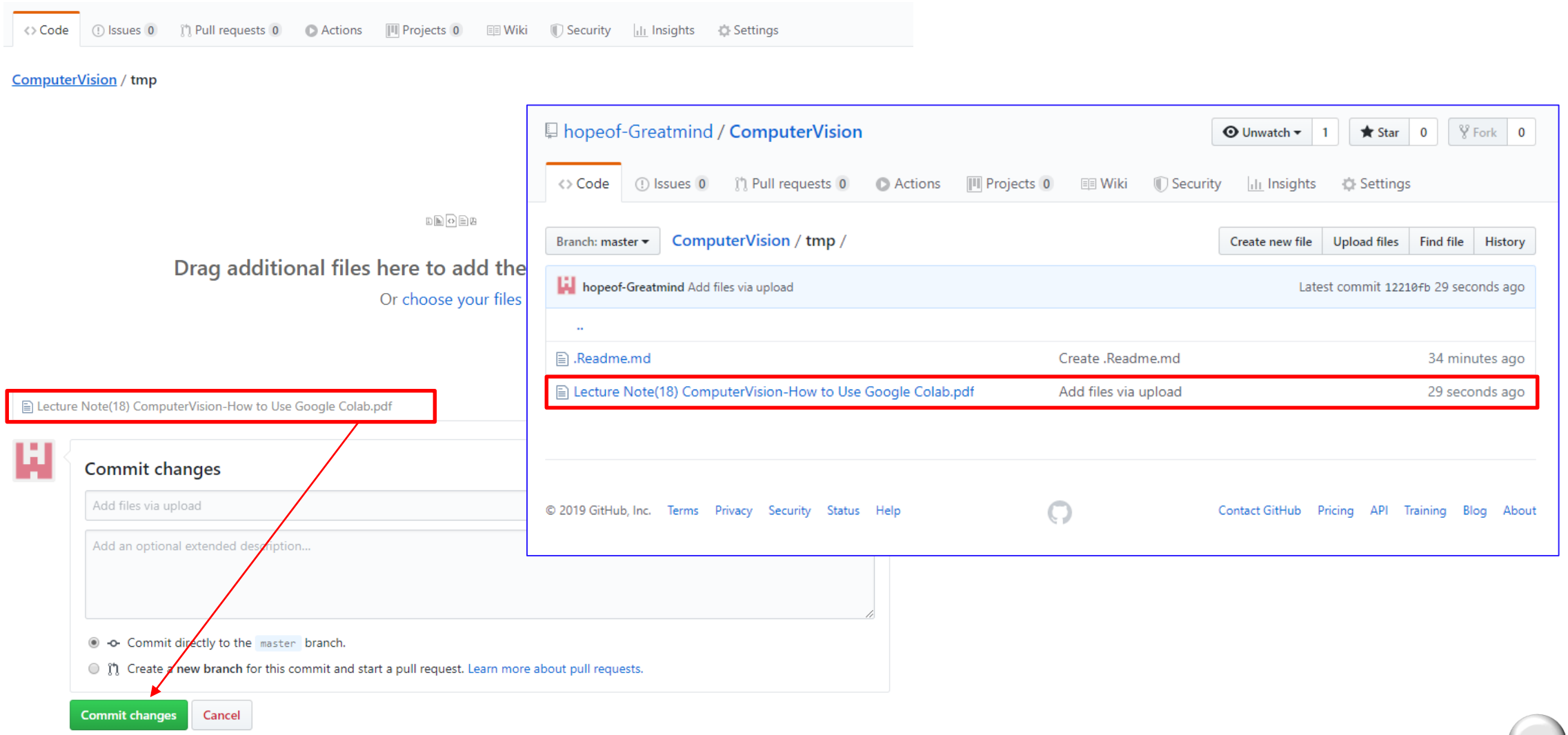
❖ Upload your file to Github.

- Click “Upload files”



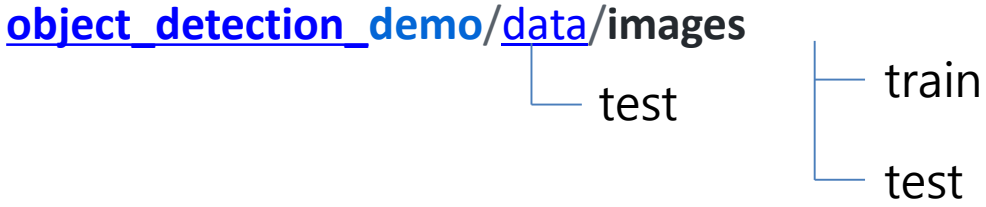
My Github (7)

- Drag & drop...!!! After uploading your files and click "Committ changes".

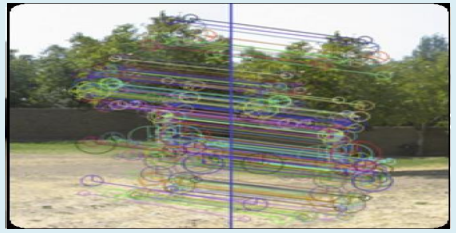


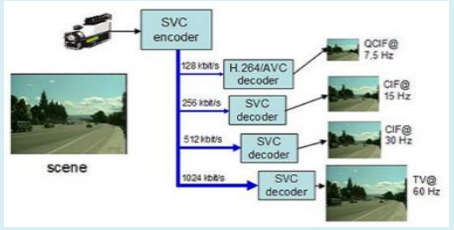
My Github (7): Folder structure of My Github

- Directory structure



- Access you Github: https://github.com/hopeof-Greatmind/object_detection_demo



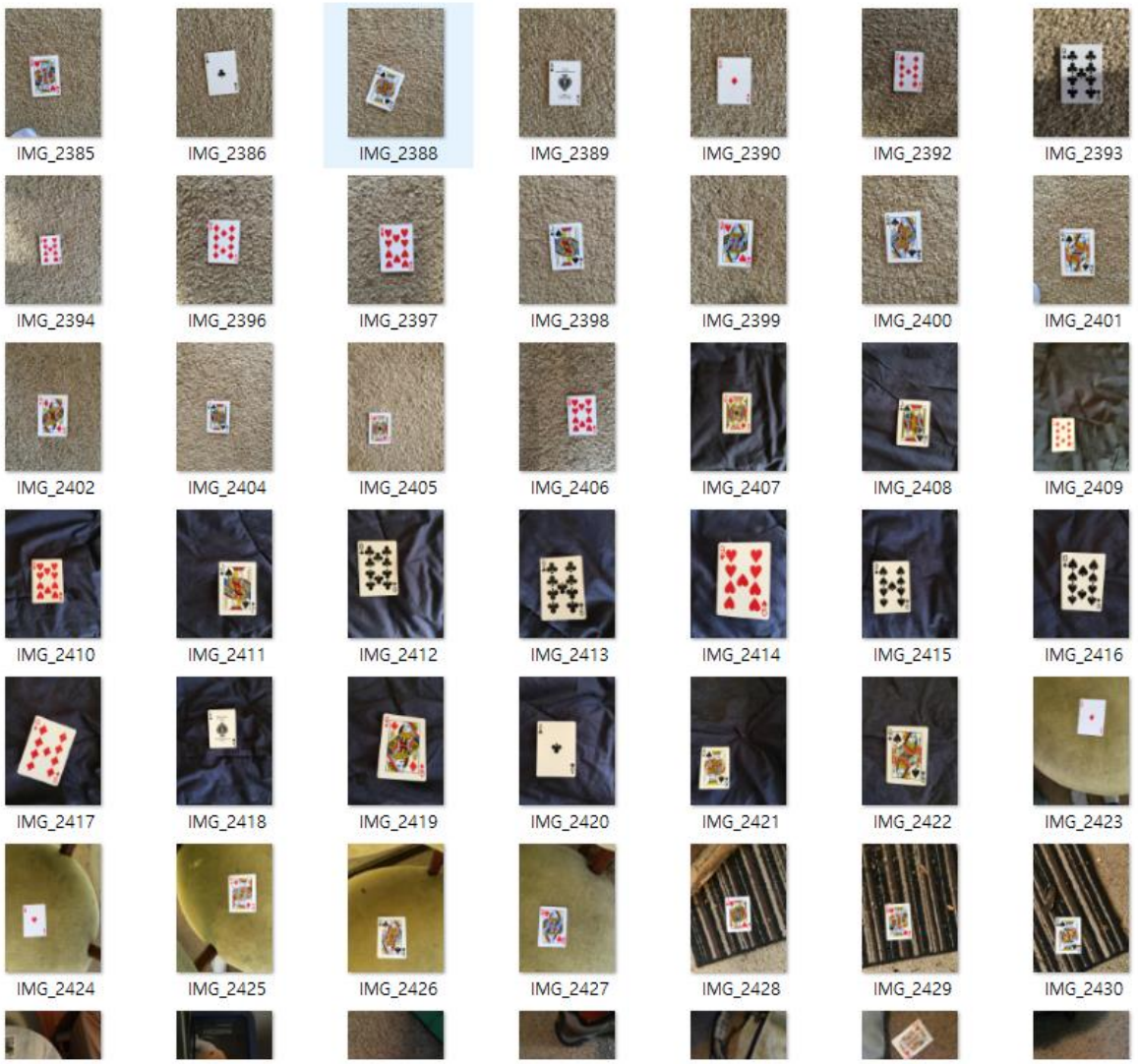


Contents

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

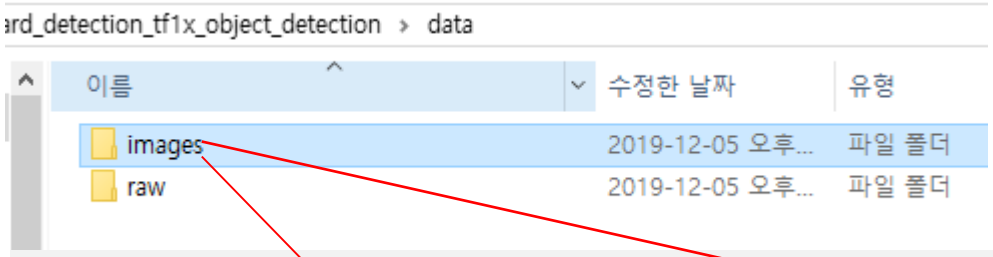
Preparation of My Dataset (1)

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



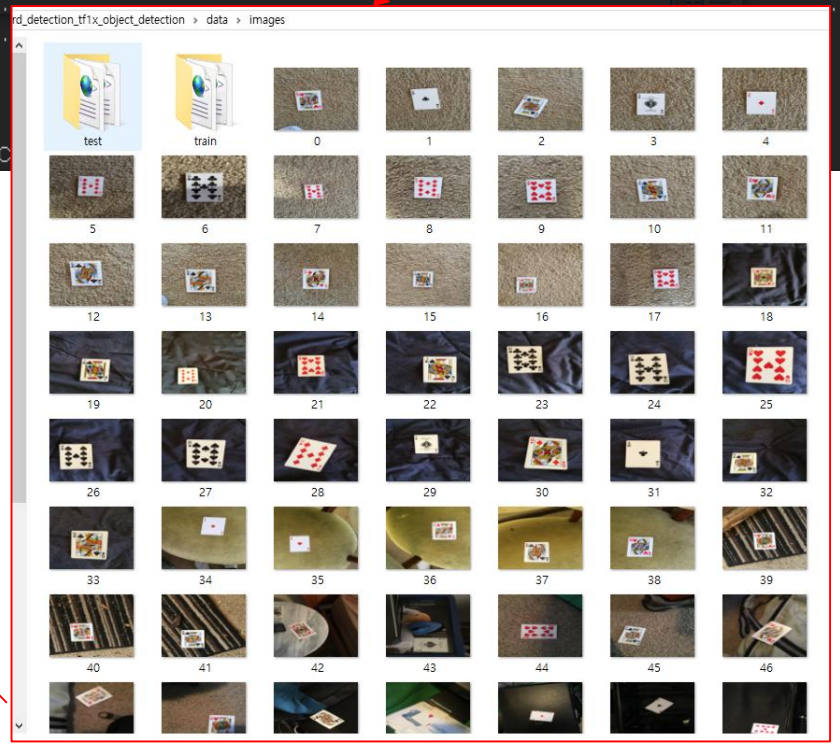
Preparation of My Dataset (2)

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



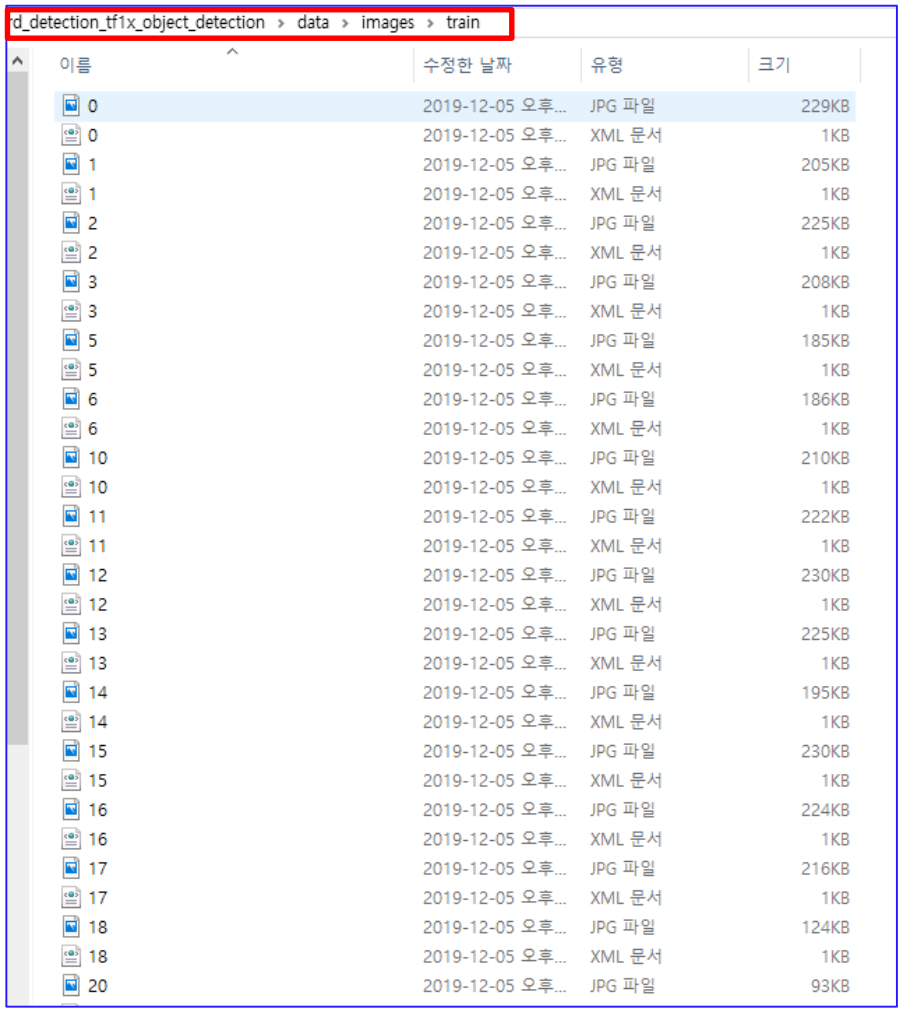
>>python resize_images.py --raw-dir ./data/raw --save-dir ./data/images --ext jpg --target-size "(800, 600)"

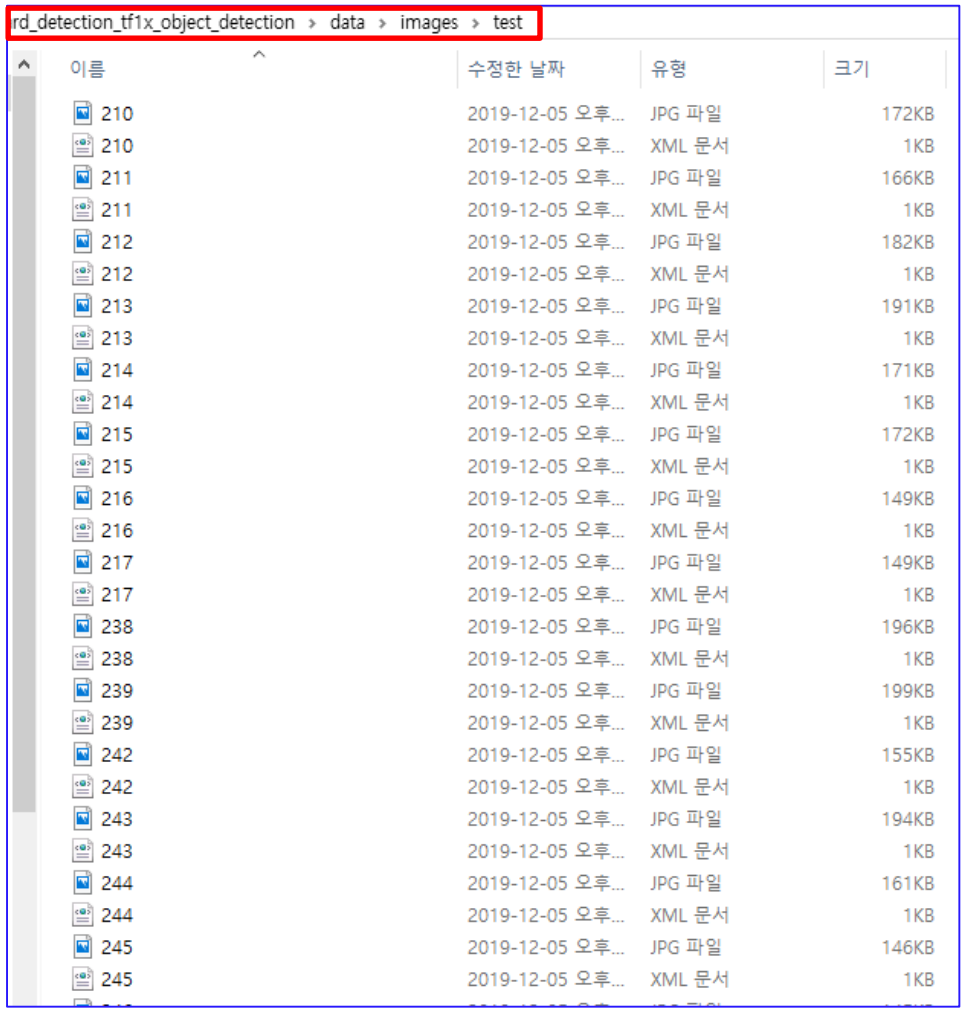
```
(BGKim) C:₩Users₩vicl₩practices₩TesorflowAPI₩card_detection_tf1x_object_detection>python resize_images.py --raw-dir ./data/raw --save-dir
 ./data/images --ext jpg --target-size "(800, 600)"
251 files to resize from directory `./data/raw` to target size:(800, 600)
................................................................................................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

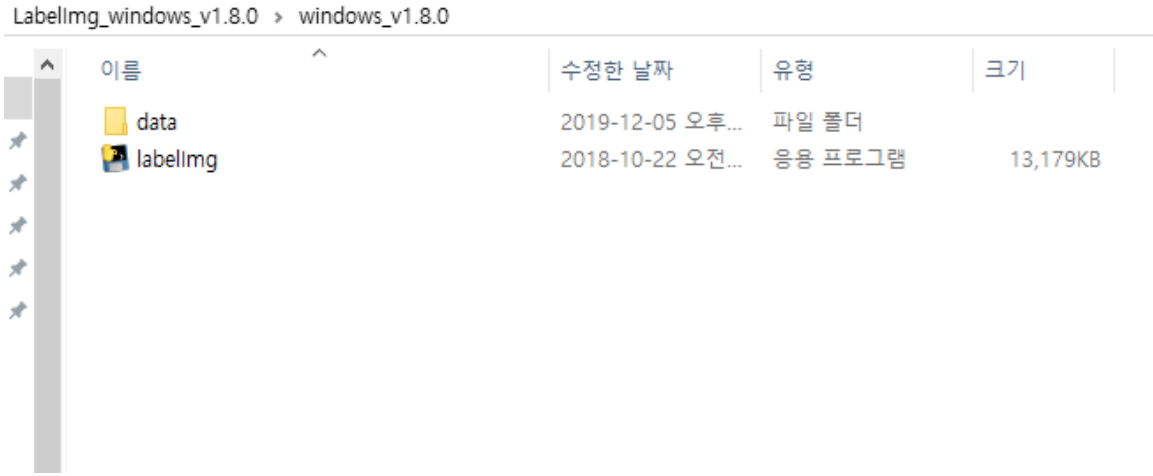




Preparation of My Dataset (4): Data Annotation

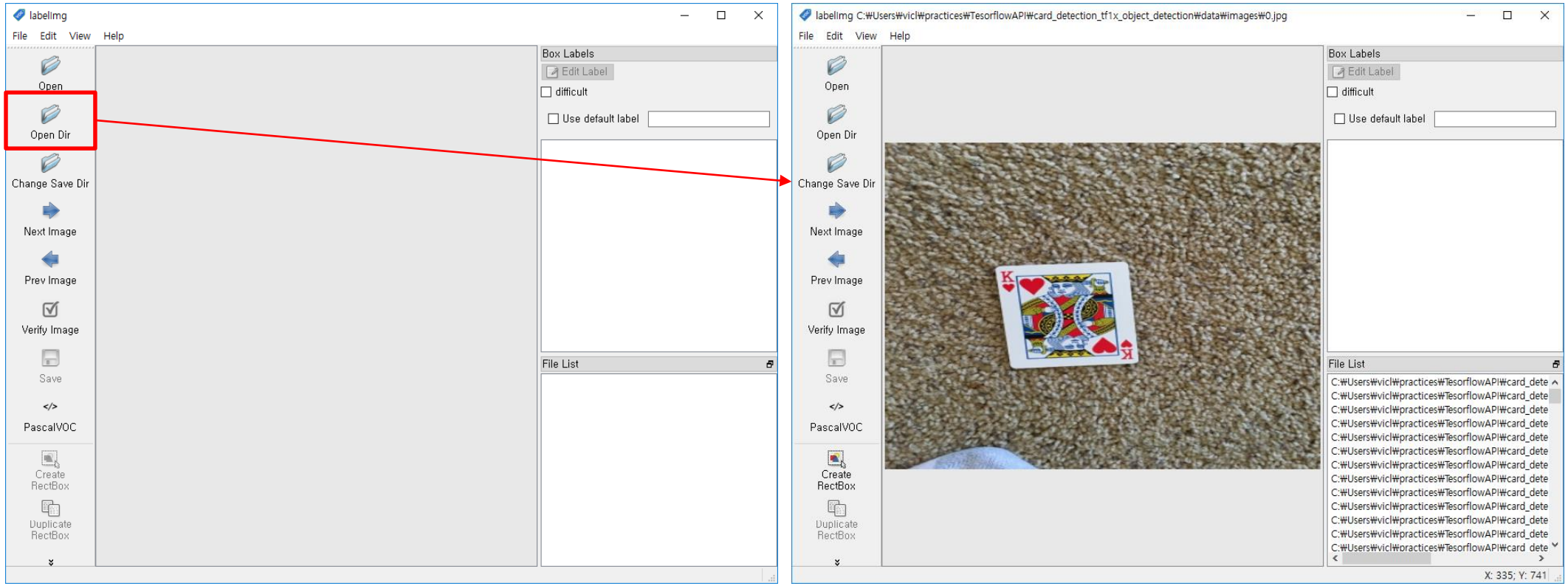
❖ **Data Annotation (object labelling) using "LabelImg" Tool (https://tzutalin.github.io/labelImg/)**

Install (decompress) Windows_v1.8.0



Preparation of My Dataset (5): Data Annotation

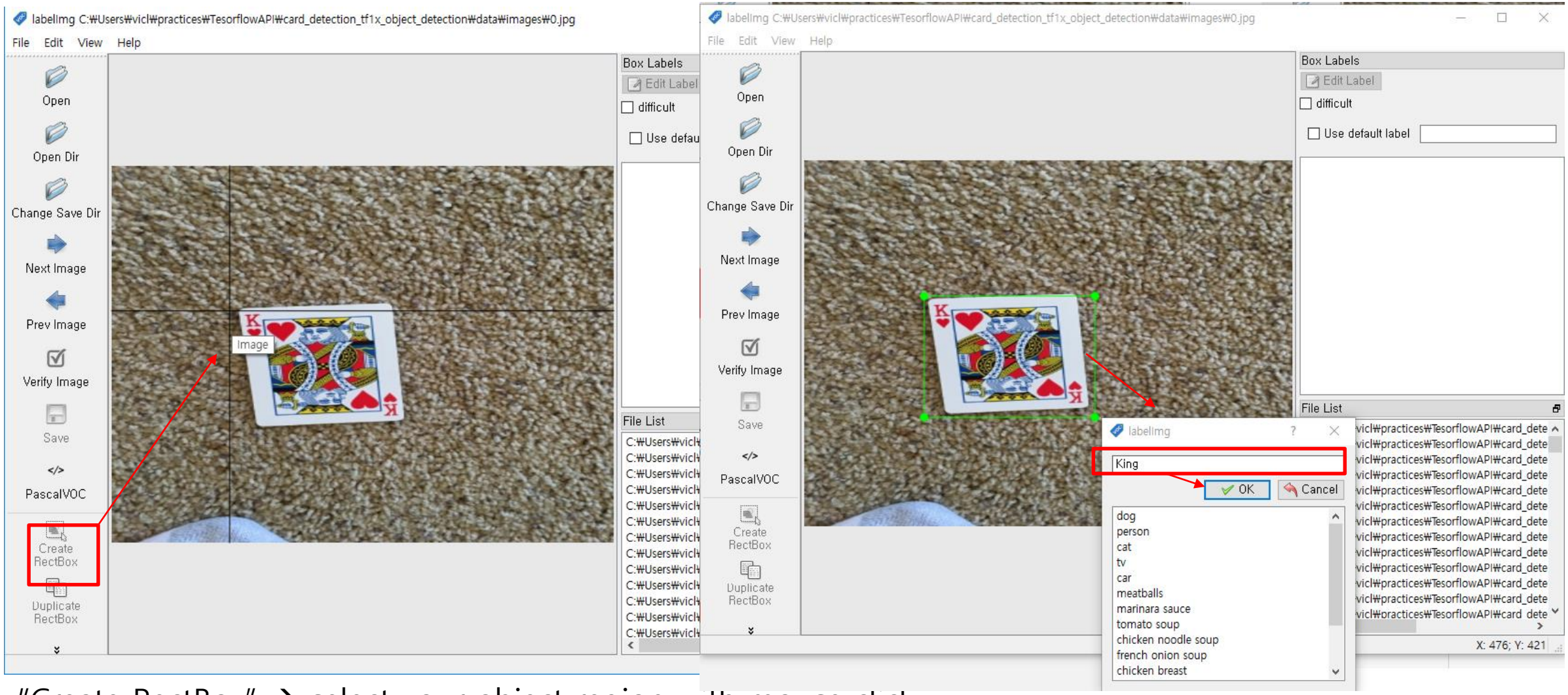
- Run LabelImg.exe



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

Preparation of My Dataset (6): Data Annotation

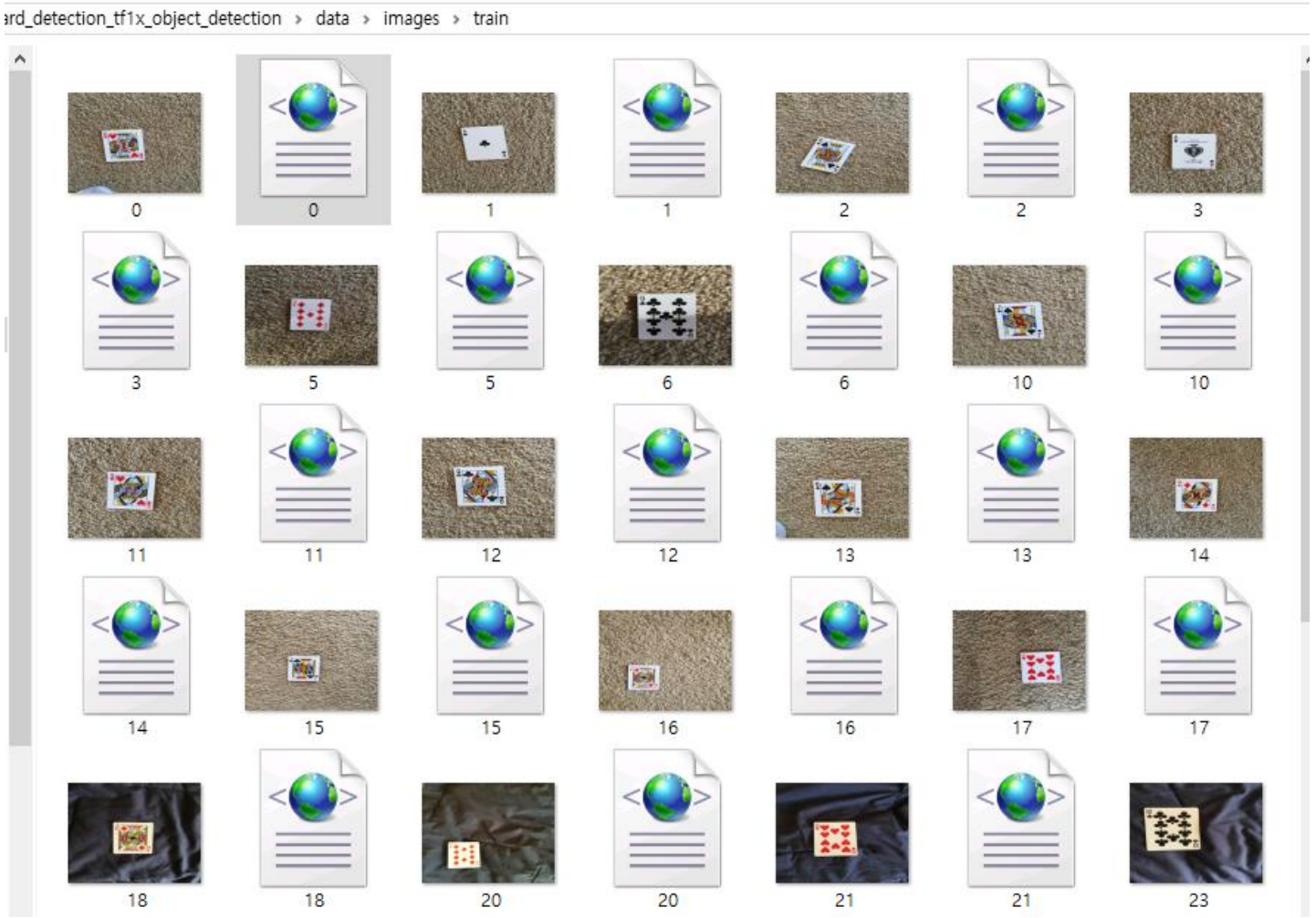
- Image labeling (annotation)



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

Preparation of My Dataset (7): Data Annotation

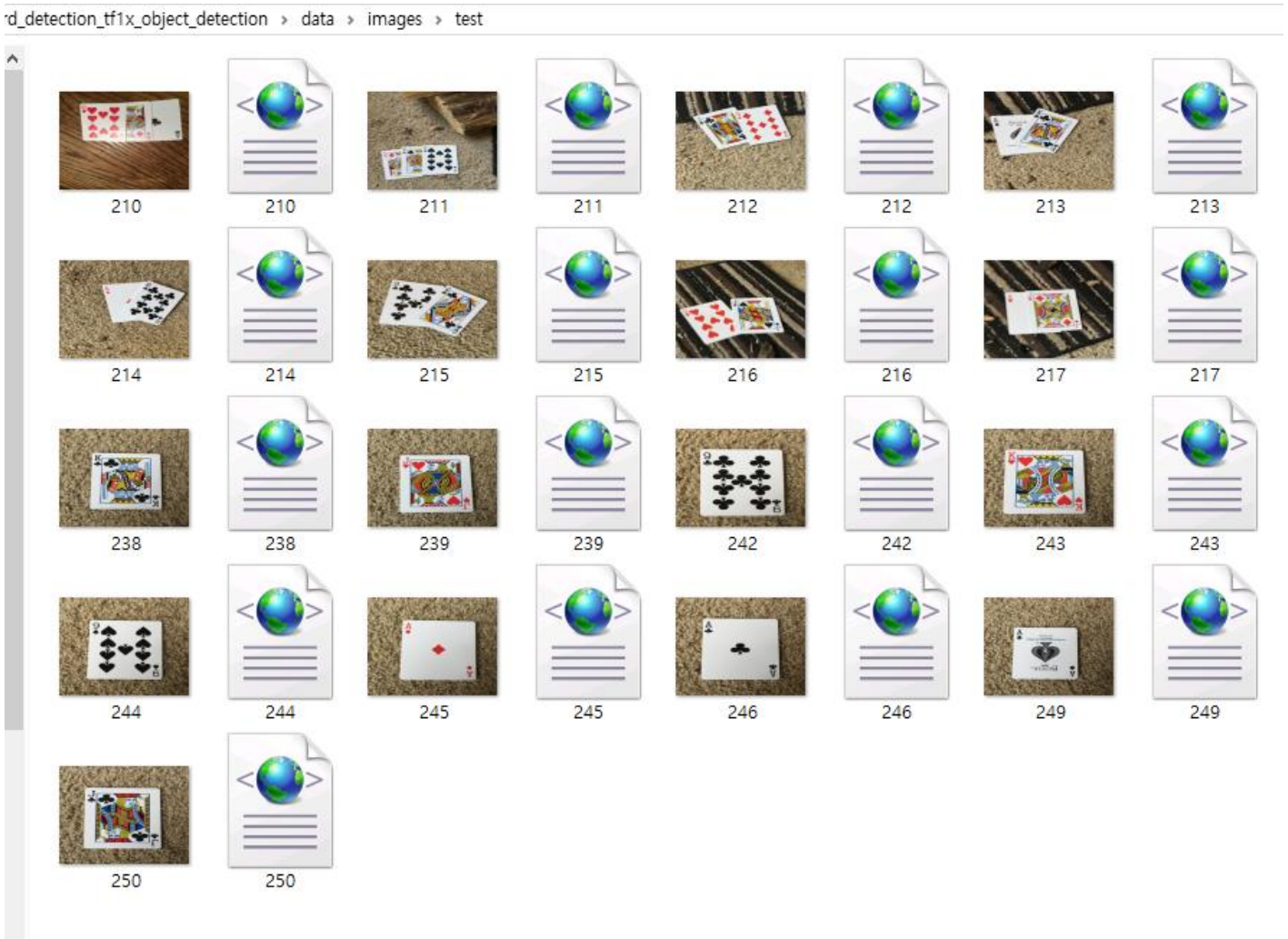
- After clicking "save" button, you can see what is change in "open dir".



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

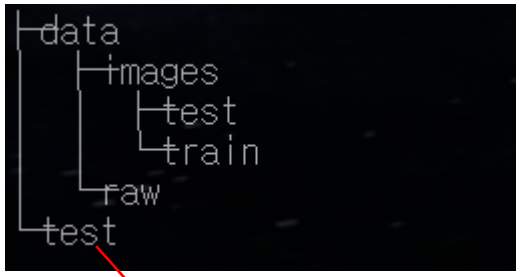
Preparation of My Dataset (8): Data Annotation

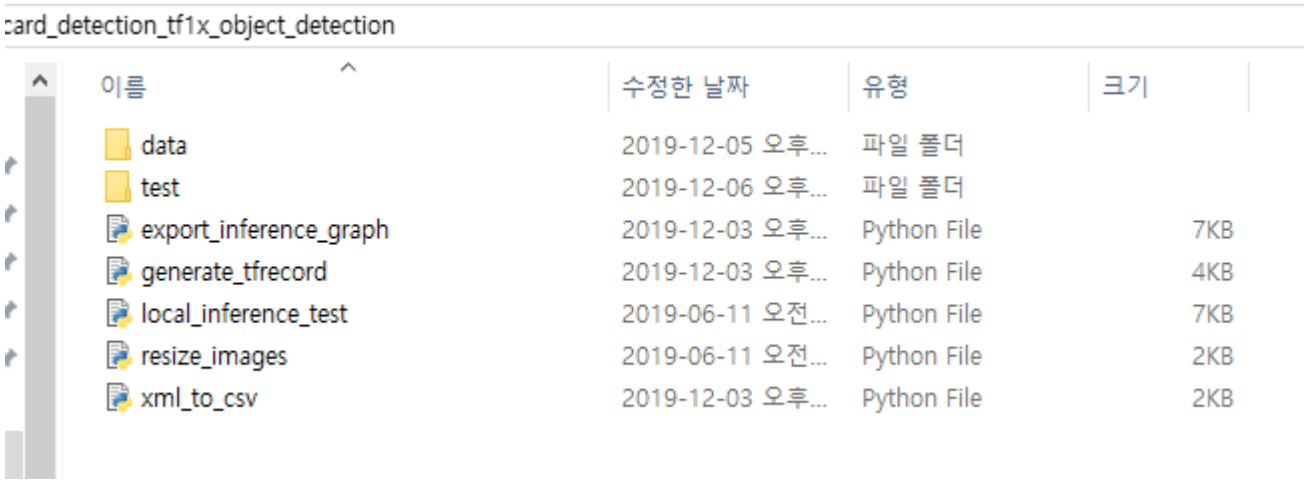
- For all your data, you have to give annotations (both of train and test datasets).

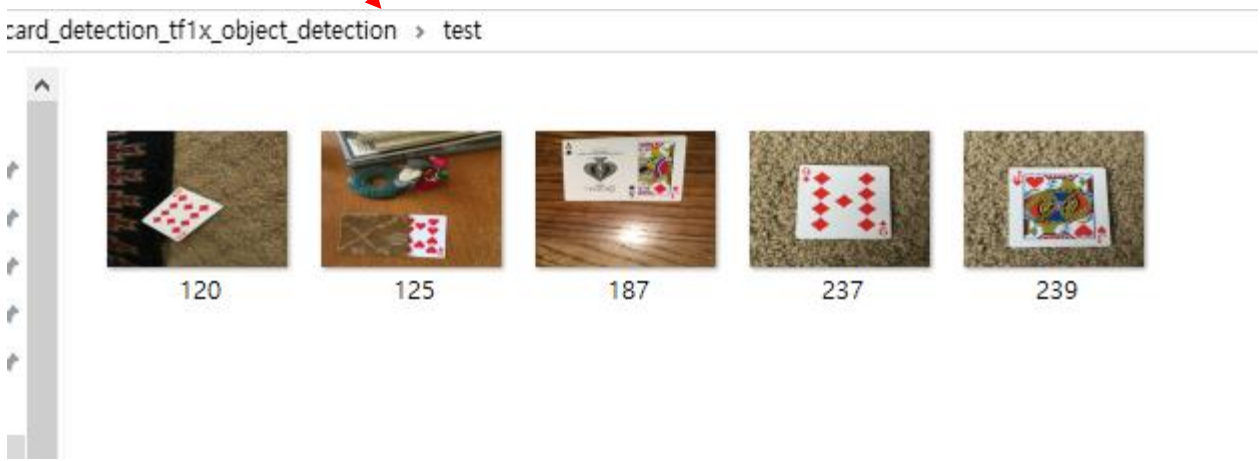


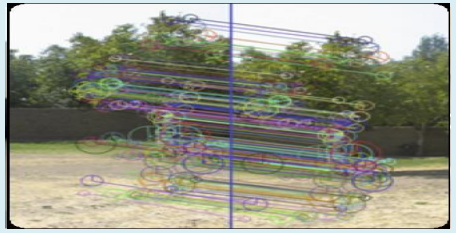
Preparation of My Dataset (9): Uploading your datasets to Github...!

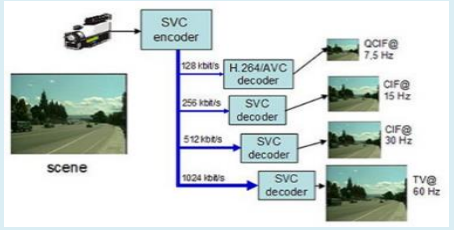
❖ Main folder:











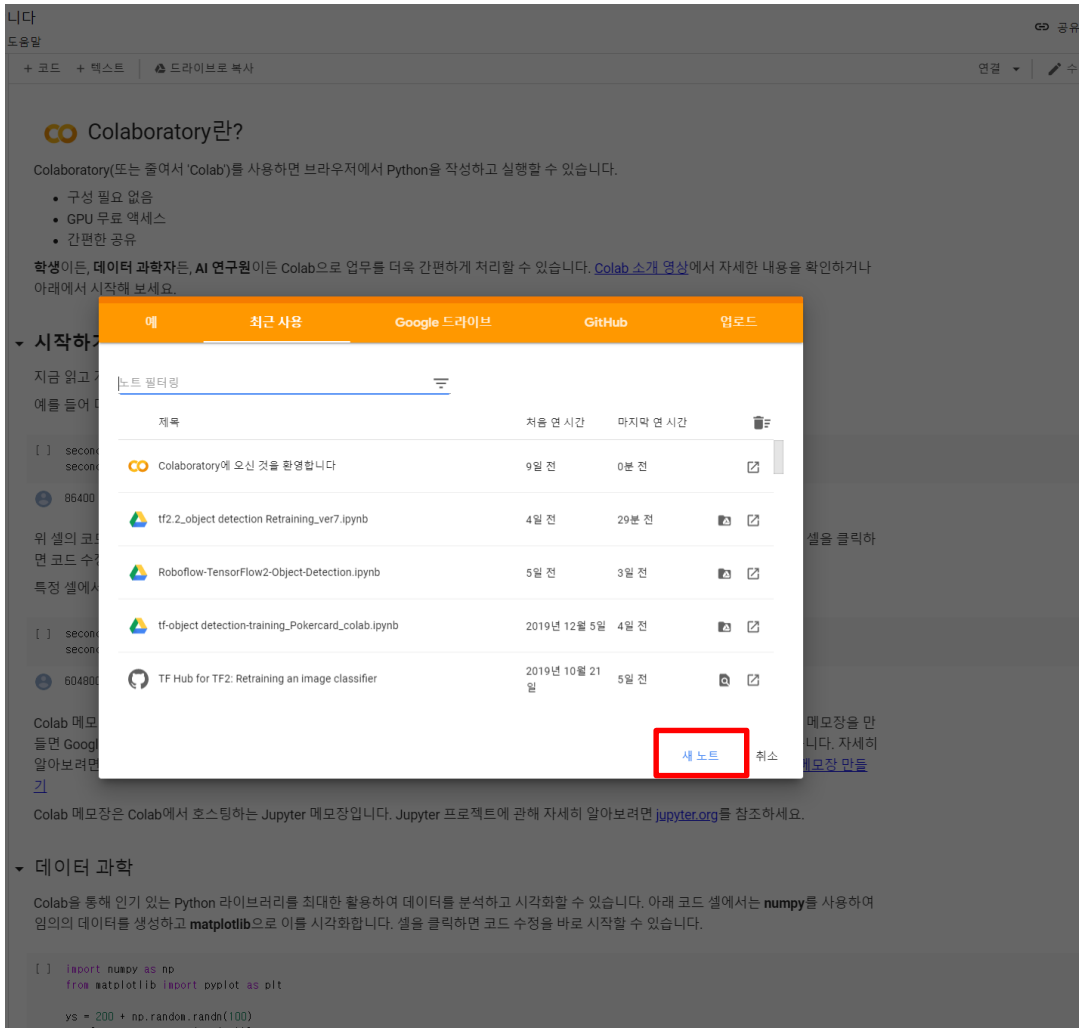
Contents

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

Actual Training in Colab (1)

❖ 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" 선택하면 됨



Actual Training in Colab (1-1)

- From now, you can give your instructions in Colab field.

Actual Training in Colab (2)

❖ Install TensorFlow2 Object Detection Dependencies

- Check on package information

```
import itertools
import os

import matplotlib.pylab as plt
import numpy as np

import tensorflow as tf
import tensorflow_hub as hub

print("TF version:", tf.__version__)
print("Hub version:", hub.__version__)
print("GPU is", "available" if tf.config.list_physical_devices('GP
U') else "NOT AVAILABLE")
```

```
TF version: 2.3.0
Hub version: 0.10.0
GPU is available
```

Actual Training in Colab (3)

- Copy tensorflow models (source) into my colab..!

```
import os
import pathlib

# Clone the tensorflow models repository if it doesn't already ex
if "models" in pathlib.Path.cwd().parts:
  while "models" in pathlib.Path.cwd().parts:
    os.chdir('..')
elif not pathlib.Path('models').exists():
  !git clone --depth 1 https://github.com/tensorflow/models
```

```
Cloning into 'models'...
remote: Enumerating objects: 2797, done.
remote: Counting objects: 100% (2797/2797), done.
remote: Compressing objects: 100% (2439/2439), done.
remote: Total 2797 (delta 563), reused 1406 (delta 322), pack-reused 0
Receiving objects: 100% (2797/2797), 57.74 MiB | 17.88 MiB/s, done.
Resolving deltas: 100% (563/563), done.
```

```
import os
import pathlib

# Clone the tensorflow models repository if it doesn't already e
xist
if "models" in pathlib.Path.cwd().parts:
  while "models" in pathlib.Path.cwd().parts:
    os.chdir('..')
elif not pathlib.Path('models').exists():
  !git clone --depth 1 https://github.com/tensorflow/models
```

```
%ls -al
```

```
total 20
drwxr-xr-x 1 root root 4096 Jul 20 03:46 ./
drwxr-xr-x 1 root root 4096 Jul 20 03:10 ../
drwxr-xr-x 1 root root 4096 Jul 15 16:11 .config/
drwxr-xr-x 8 root root 4096 Jul 20 03:46 models/
drwxr-xr-x 1 root root 4096 Jul 10 16:29 sample_data/
```

Actual Training in Colab (4)

- Install the Object Detection API

```
# Install the Object Detection API
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

```
# Install the Object Detection API
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

```
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (0.3.3)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (0.3
Requirement already satisfied: tensorboard<2.3.0,>=2.2.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (2.2.2)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (1.1.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (3.2.1)
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=2.2.0->tf-models-official->object-detection==0.1) (1.1.2)
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3.6/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-detection==0.
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-detection==0.1)
Requirement already satisfied: google-auth>=1.4.1 in /usr/local/lib/python3.6/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-detection==0.1) (1.17.2
Requirement already satisfied: httplib2<1dev,>=0.17.0 in /usr/local/lib/python3.6/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-detection==0.1) (0
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (4.41.1)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (2.23.0)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (2020.6.20)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (4.0.1)
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.6/dist-packages (from oauth2client>=4.1.2->tf-models-official->object-detection==0.1) (0.4.8)
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.6/dist-packages (from oauth2client>=4.1.2->tf-models-official->object-detection==0.1) (4.6)
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.6/dist-packages (from oauth2client>=4.1.2->tf-models-official->object-detection==0.1) (0.2.8)
Collecting dm-tree~=0.1.1
  Downloading https://files.pythonhosted.org/packages/16/48/10fb7213348100810b7e6eebeba0d12e12126c76993e8c243062d2f56a89f/dm_tree-0.1.5-cp36-cp36m-manylinux1_x86_64.whl (294kB)
Requirement already satisfied: attrs>=18.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-datasets->tf-models-official->object-detection==0.1) (19.3.0)
Requirement already satisfied: dill in /usr/local/lib/python3.6/dist-packages (from tensorflow-datasets->tf-models-official->object-detection==0.1) (0.3.2)
    Successfully uninstalled pyarrow-0.14.1
  Found existing installation: dill 0.3.3
    Uninstalling dill-0.3.3:
      Successfully uninstalled dill-0.3.3
Successfully installed apache-beam-2.25.0 avro-python3-1.10.0 dill-0.3.1.1 fastavro-1.2.1 future-0.18.2 hdfs-2.5.8 lvis-0.5.3 mock-2.0.0 object-detection-0.1 opencv-pytho
ERROR: multiprocess 0.70.11.1 has requirement dill>=0.3.3, but you'll have dill 0.3.1.1 which is incompatible.
ERROR: google-colab 1.0.0 has requirement requests~=2.23.0, but you'll have requests 2.25.0 which is incompatible.
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
ERROR: apache-beam 2.25.0 has requirement avro-python3!=1.9.2,<1.10.0,>=1.8.1; python_version >= "3.0", but you'll have avro-python3 1.10.0 which is incompatible.
```

IVPI Intelligent Vision Processing

Actual Training in Colab (5)

- Import some needed modules:

```
import matplotlib
import matplotlib.pyplot as plt

import os
import random
import io
import imageio
import glob
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display, Javascript
from IPython.display import Image as IPyImage

import tensorflow as tf

from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils
as viz_utils
from object_detection.utils import colab_utils
from object_detection.builders import model_builder

%matplotlib inline
```

Actual Training in Colab (6)

- Check on “tf_util.py” and modify it.

```
#commence temporary keras fix for exporting efficientDet - this will inevitably be fixed and removed from the notebook in the coming days
%cat /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/utils/tf_utils.py
```

```
# Copyright 2018 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================
"""TensorFlow-related utilities."""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import copy
import numpy as np
import six
```

Actual Training in Colab (7)

- Run model builder test

```
#run model builder test
!python /content/models/research/object_detection/builders/model_builder_tf2_test.py
```

```
#run model builder test
!python /content/models/research/object_detection/builders/model_builder_tf2_test.py
```

```
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul): 0.15s
I1203 08:15:14.412472 140073004533632 test_util.py:1973] time(__main__.ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_faster_r
[       OK ] ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul
[ RUN      ] ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul): 0.15s
I1203 08:15:14.566688 140073004533632 test_util.py:1973] time(__main__.ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_mask_rcn
[       OK ] ModelBuilderTF2Test.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
```

```
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I1203 08:15:45.176442 140073004533632 test_util.py:1973] time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I1203 08:15:45.177160 140073004533632 test_util.py:1973] time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I1203 08:15:45.178380 140073004533632 test_util.py:1973] time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
Ran 20 tests in 39.872s

OK (skipped=1)
```

Actual Training in Colab (8)

- Implementation of the basic functions

```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  Puts image into numpy array to feed into tensorflow graph.
  Note that by convention we put it into a numpy array with shape
  (height, width, channels), where channels=3 for RGB.

  Args:
    path: a file path.

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):
  """Wrapper function to visualize detections.

  Args:
    image_np: uint8 numpy array with shape (img_height, img_width, 3)
    boxes: a numpy array of shape [N, 4]
    classes: a numpy array of shape [N]. Note that class indices are
      and match the keys in the label map.
    scores: a numpy array of shape [N] or None.  If scores=None, then
      this function assumes that the boxes to be plotted are groundtr
```

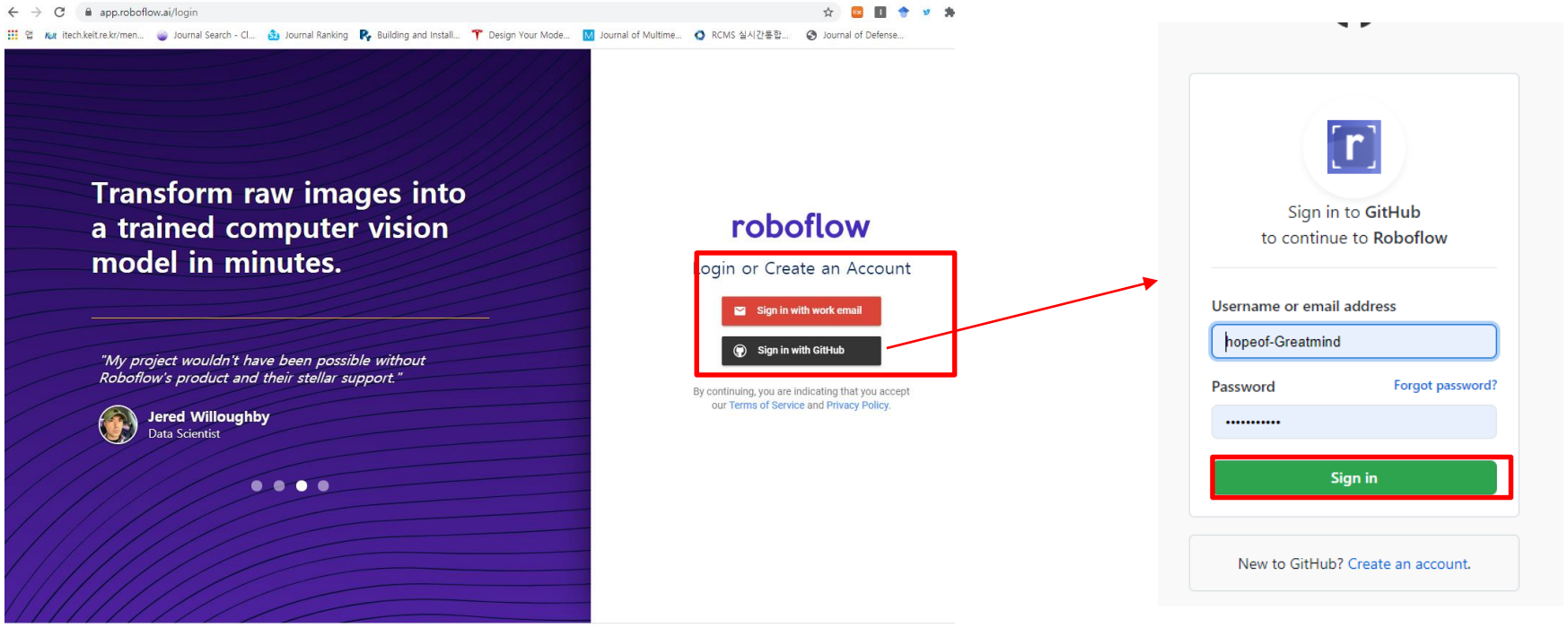
```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  Puts image into numpy array to feed into tensorflow graph.
  Note that by convention we put it into a numpy array with shape
  (height, width, channels), where channels=3 for RGB.

  Args:
    path: a file path.

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):
  """Wrapper function to visualize detections.

  Args:
    image_np: uint8 numpy array with shape (img_height, img_width, 3)
    boxes: a numpy array of shape [N, 4]
    classes: a numpy array of shape [N]. Note that class indices are 1-based,
      and match the keys in the label map.
    scores: a numpy array of shape [N] or None.  If scores=None, then
      this function assumes that the boxes to be plotted are groundtruth
      boxes and plot all boxes as black with no classes or scores.
    category_index: a dict containing category dictionaries (each holding
      category index `id` and category name `name`) keyed by category indices.
    figsize: size for the figure.
    image_name: a name for the image file.
  """
  image_np_with_annotations = image_np.copy()
  viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_annotations,
      boxes,
      classes,
      scores,
      category_index,
      use_normalized_coordinates=True,
      min_score_thresh=0.8)
  if image_name:
    plt.imsave(image_name, image_np_with_annotations)
  else:
    plt.imshow(image_np_with_annotations)
```

Actual Training in Colab (9)

❖ Prepare Tensorflow 2 Object Detection Training Data

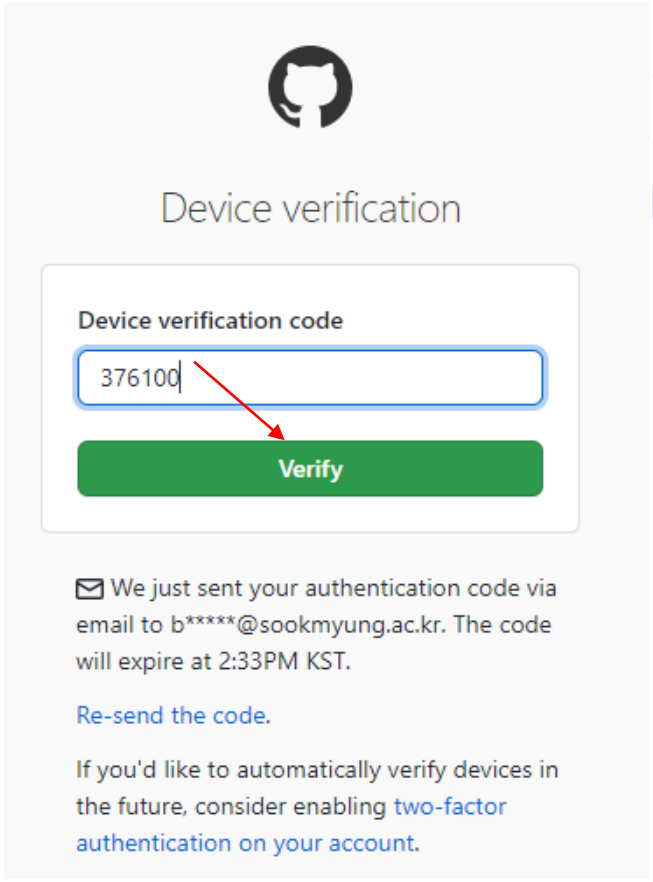
- Basically we need **TFRecord** and **label_map files.**
- **To create these, image and xml data are needed (from labeling process).**
- Roboflow (https://roboflow.com/) automatically creates our **TFRecord** and **label_map files** that we need!

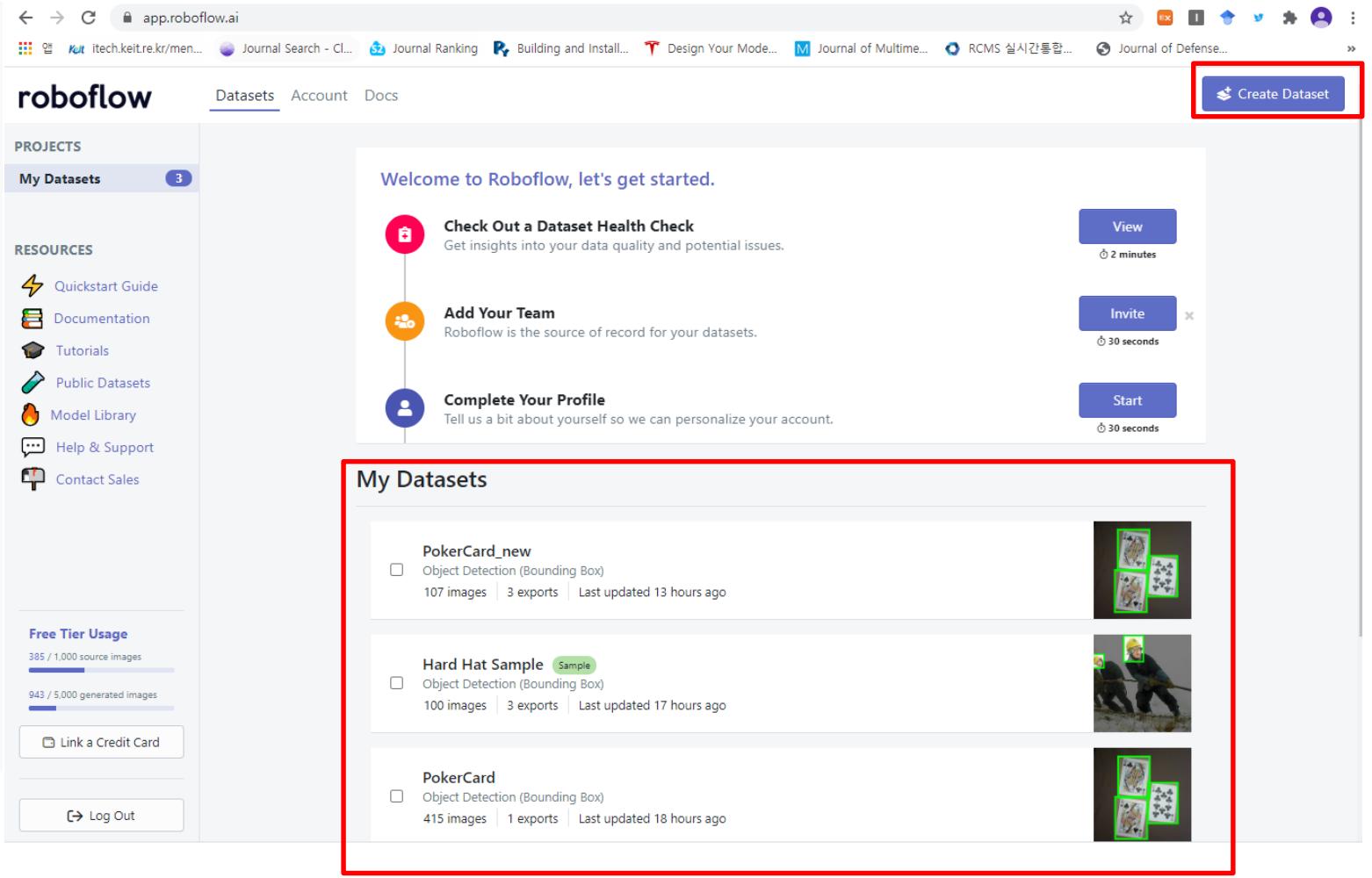
- How to generate **TFRecord** and **label_map files** in Roboflow (https://roboflow.com/)
  - 1] Login Roboflow (https://roboflow.com/).



Actual Training in Colab (10)

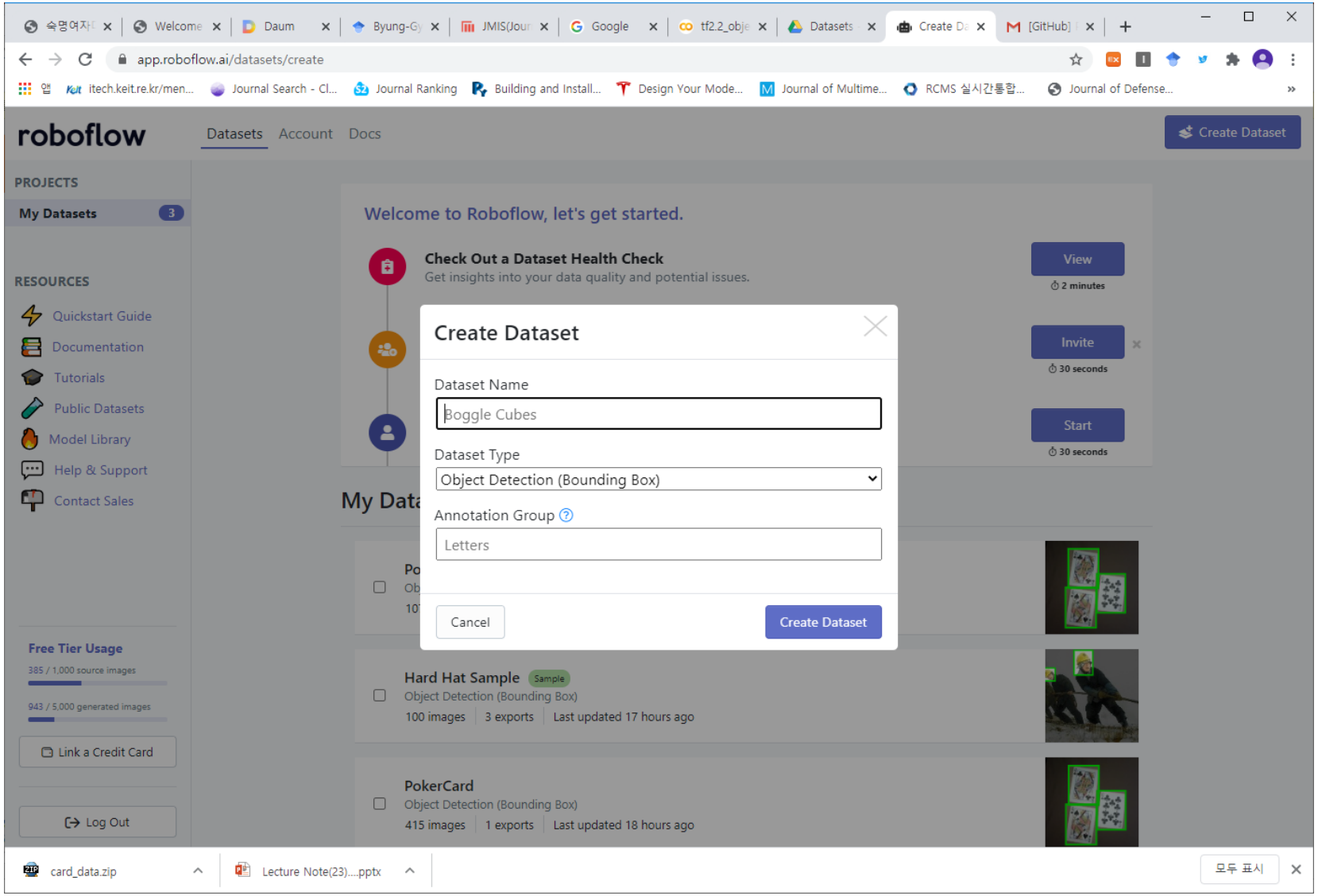
- 2] Verification of your account and Upload your datasets





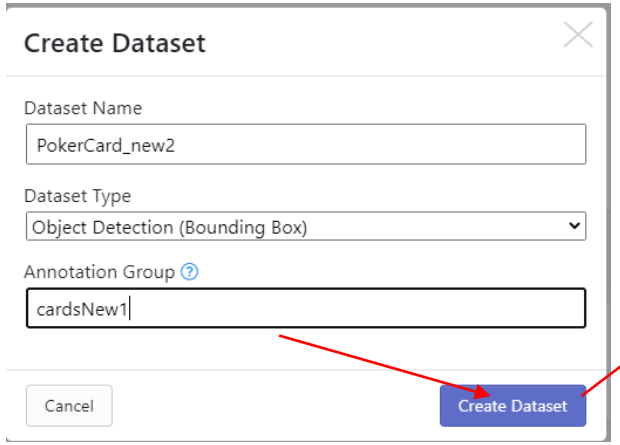
Actual Training in Colab (11)

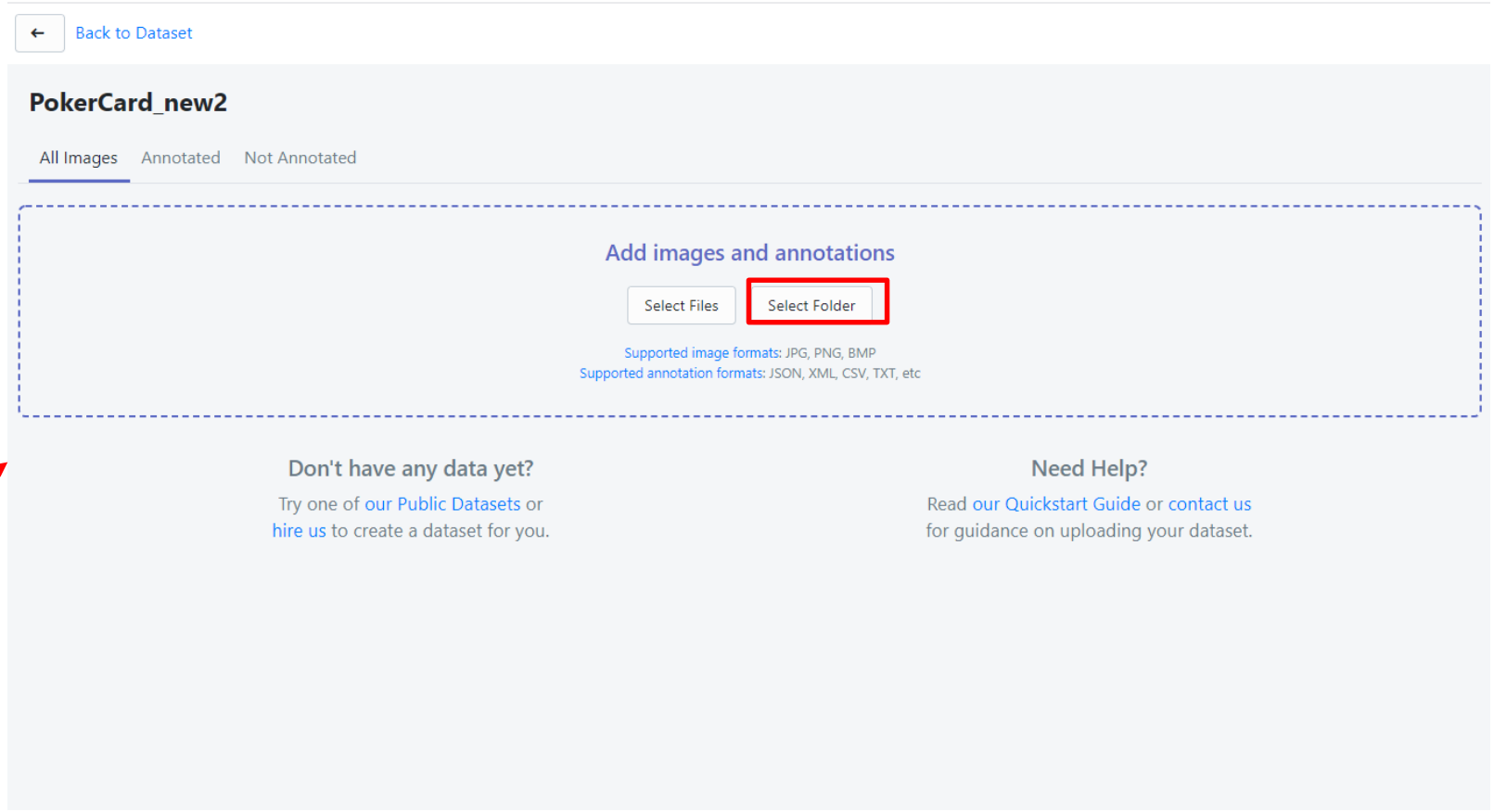
– "Create Dataset" click, the you can see pop-up window for defining my dataset information.



Actual Training in Colab (12)

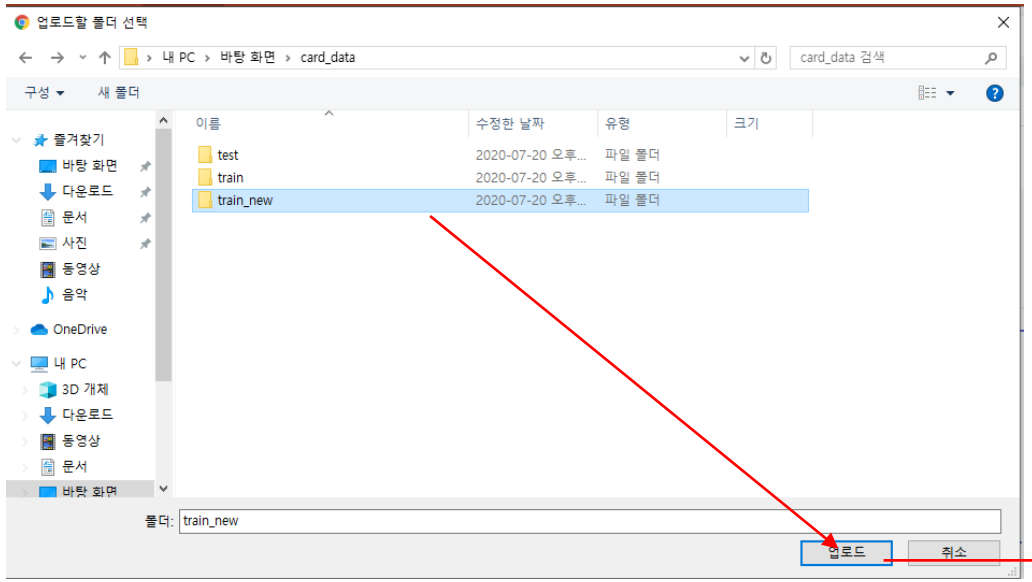
– Fill the name and type, annotation group and click “Create Dataset” .

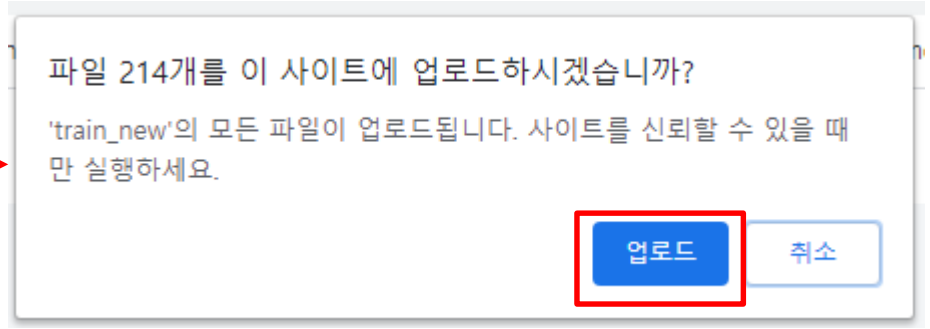




Actual Training in Colab (13)

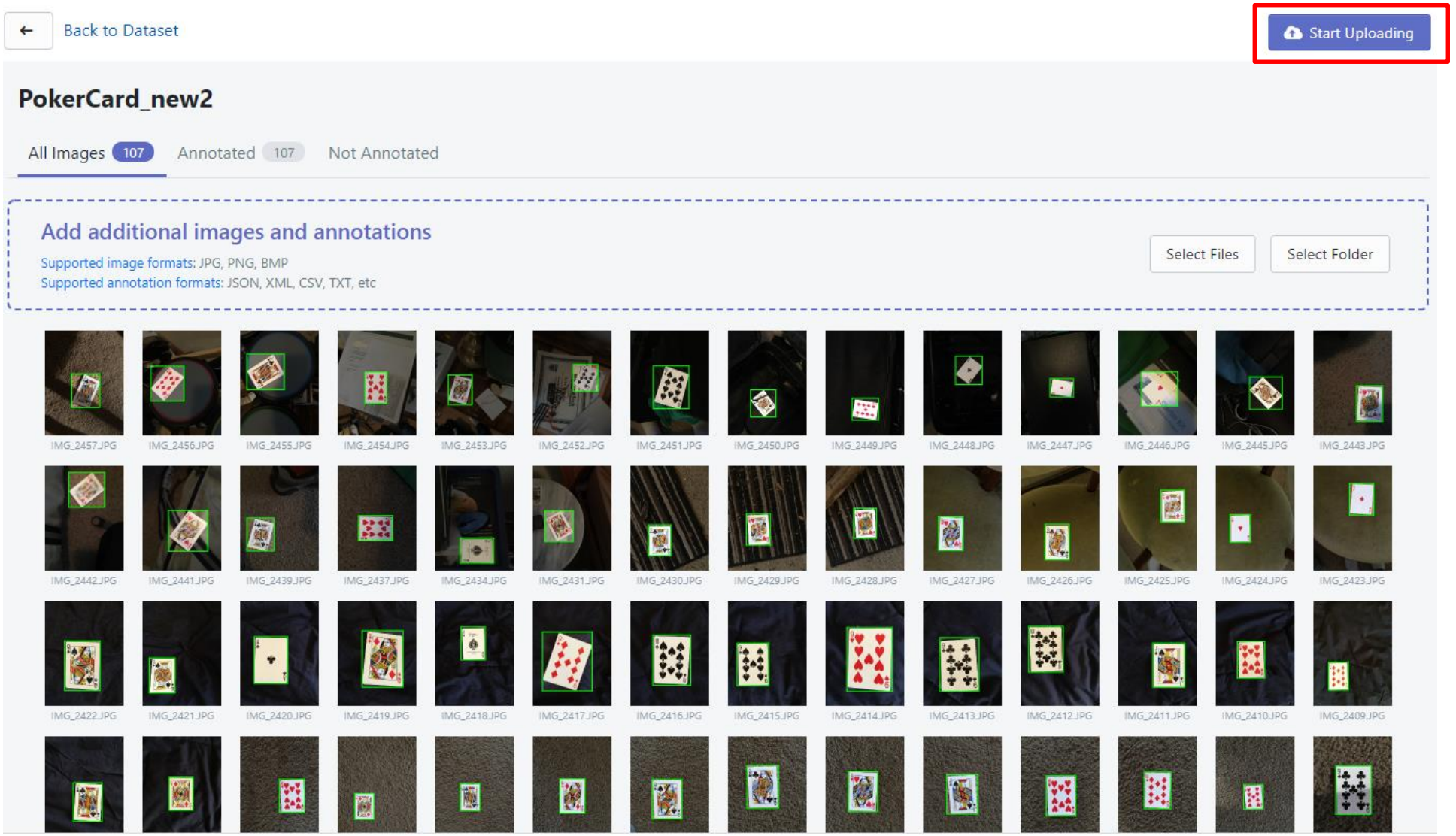
– Select the data folder and upload files.





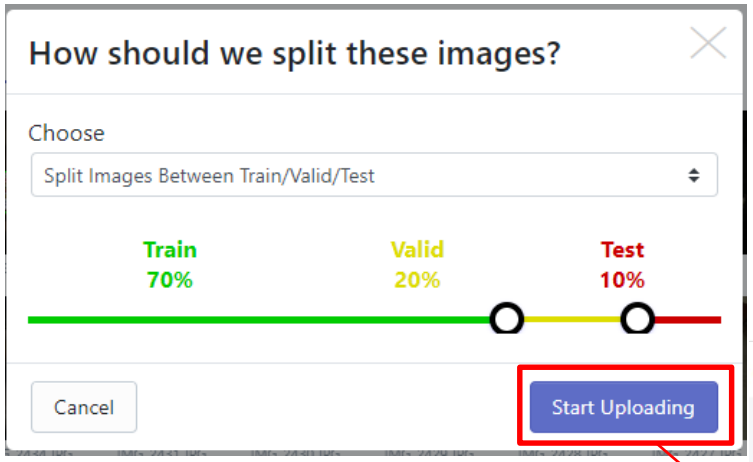
Actual Training in Colab (14)

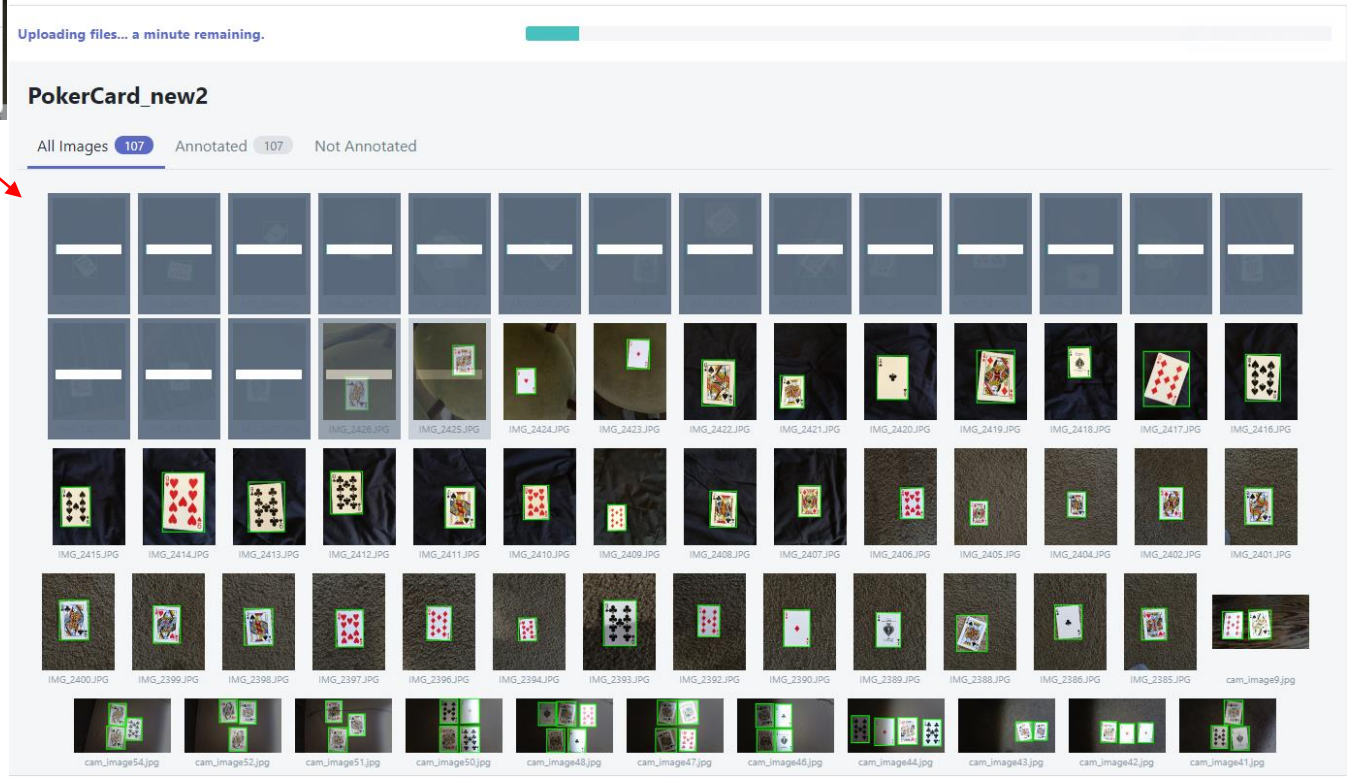
– You can check the selected files in the Browser…!!! Then “start uploading”.



Actual Training in Colab (15)

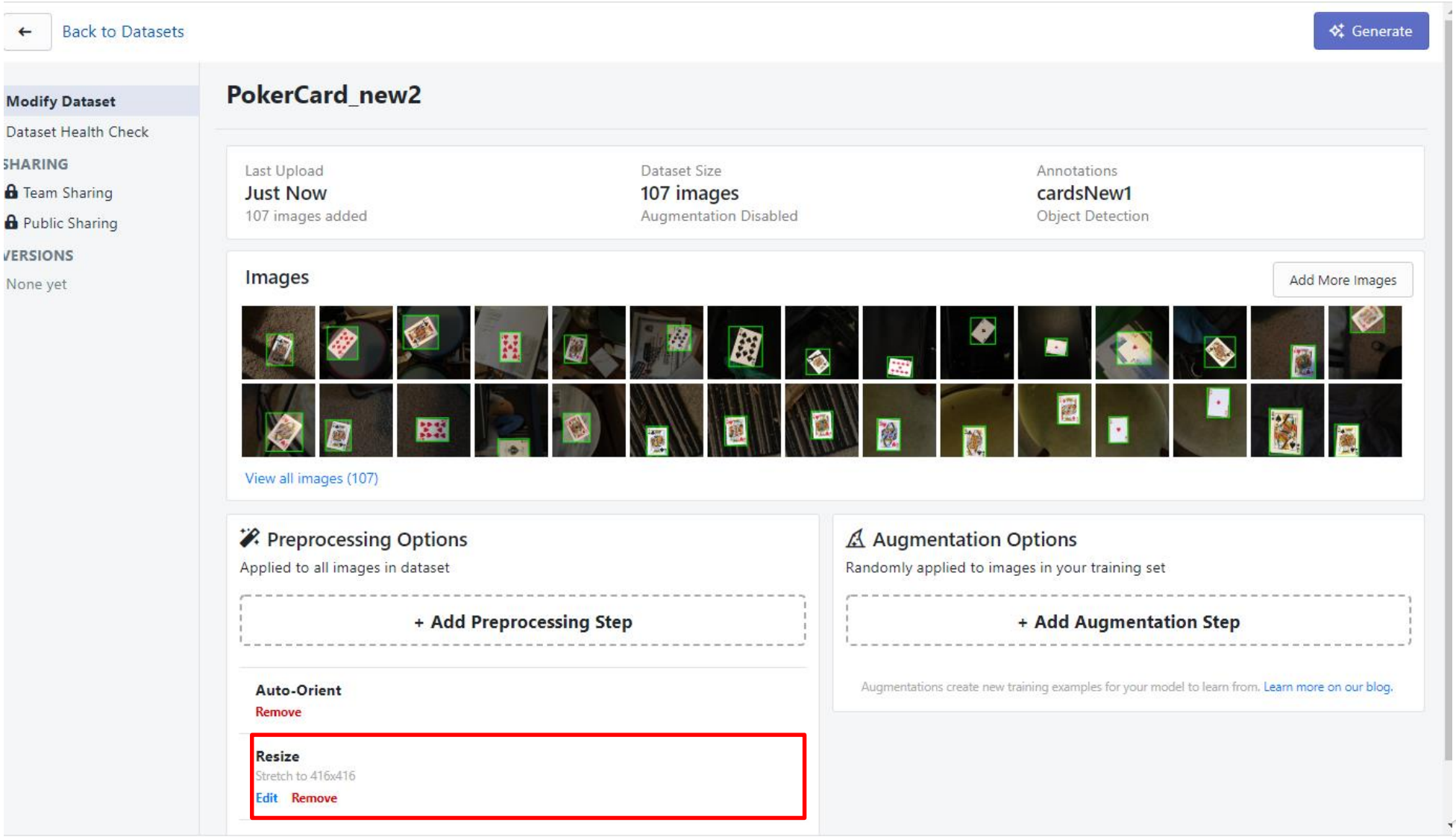
– Option) separate dataset into train, validation, and test sets. After checking and click "Start Uploading".





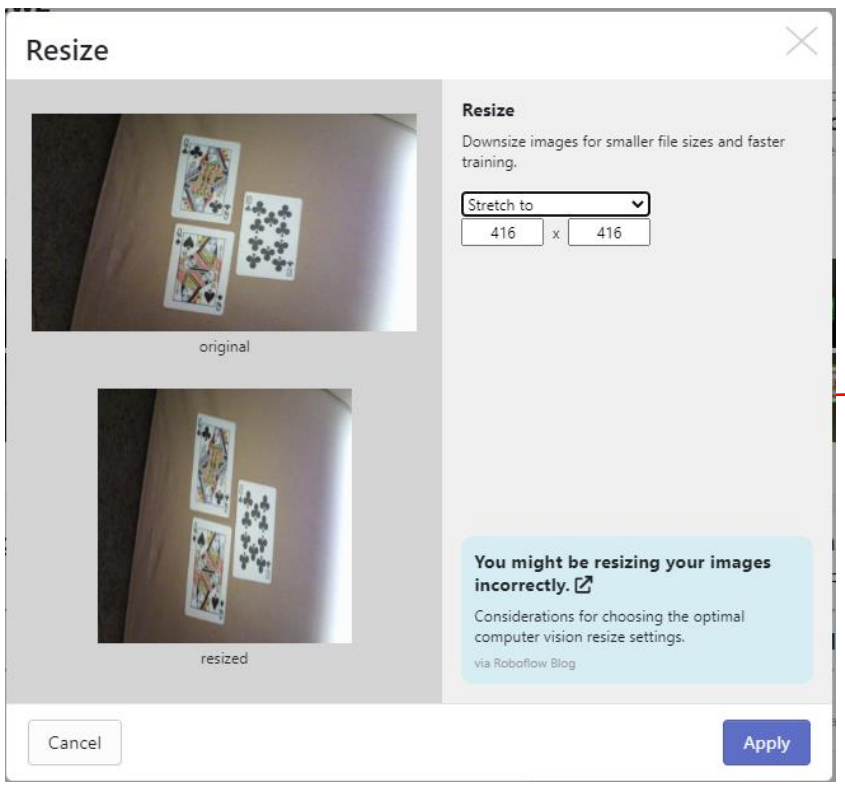
Actual Training in Colab (16)

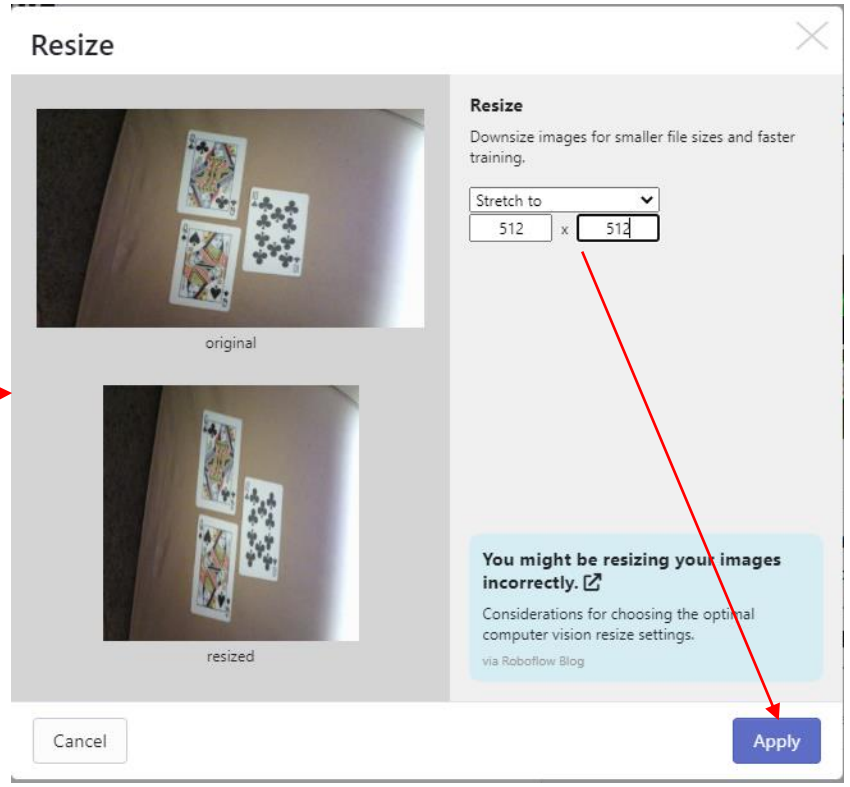
– After uploading your dataset, you can augment your dataset. Here we add "Resize" option as 512x512.



Actual Training in Colab (17)

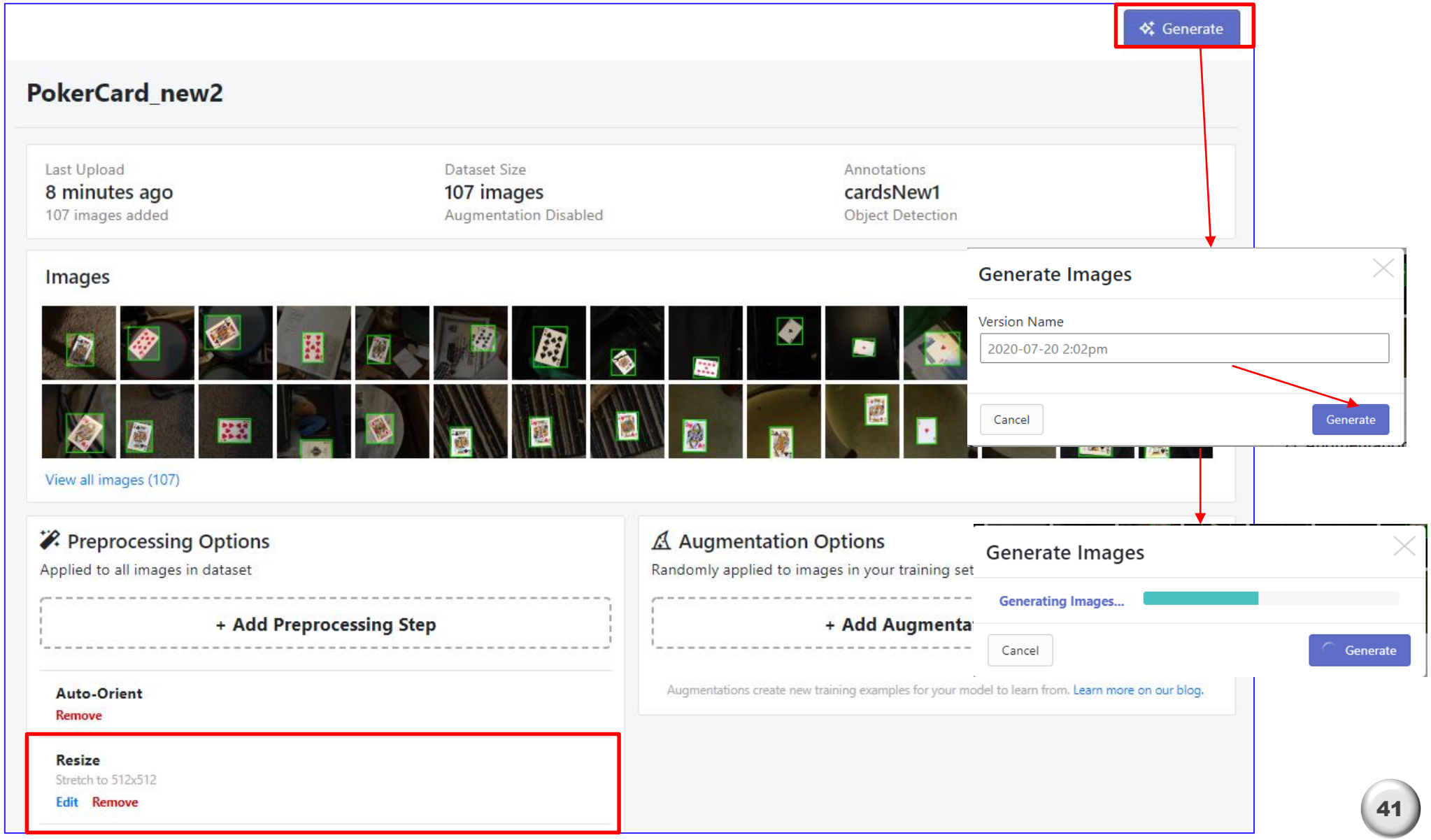
– Click "Resize" item, then you can define image resolution you want.





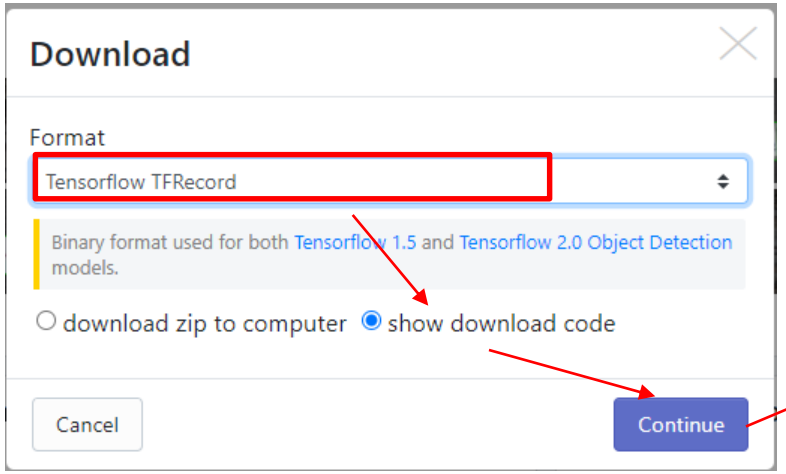
Actual Training in Colab (18)

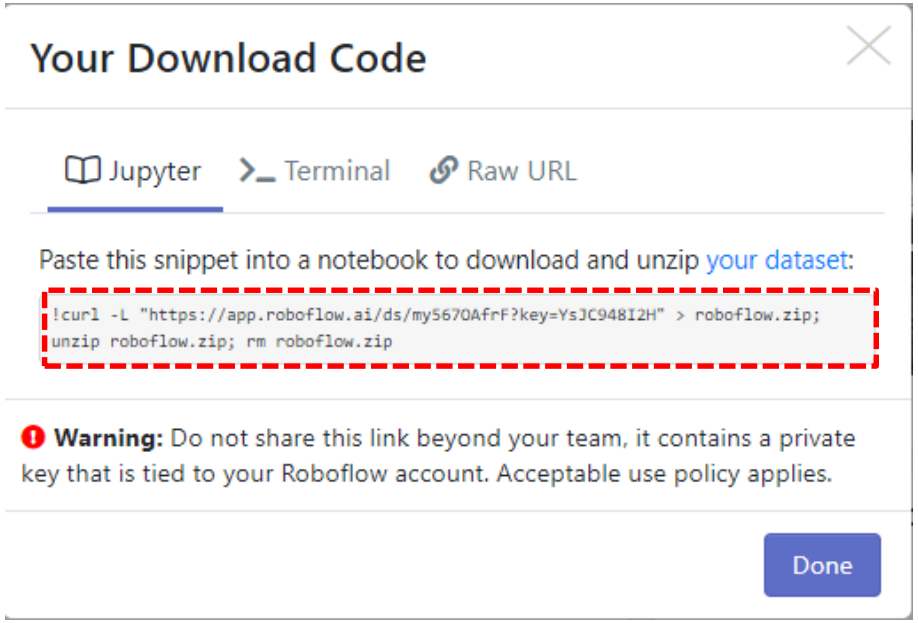
– After checking on the resized info, click "Generate" button.



Actual Training in Colab (19)

– Then download your dataset transformed files as TFRecord. Usually we use "down code" from web.





Actual Training in Colab (20)

- Downloading data into my colab from Roboflow.

```
#Downloading data from Roboflow
#UPDATE THIS LINK - get our data from Roboflow
!pwd
%cd /content
!curl -L "https://app.roboflow.ai/ds/my5670AfrF?key=YsJC948I2H" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
/content
/content
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   891  100   891    0     0    987      0 --:--:-- --:--:-- --:--:--   987
100 4475k  100 4475k    0     0  3638k      0  0:00:01  0:00:01 --:--:--  299M
Archive:  roboflow.zip
 extracting: test/cardsNew1.tfrecord
 extracting: train/cardsNew1.tfrecord
 extracting: valid/cardsNew1.tfrecord
 extracting: test/cardsNew1_label_map.pbtxt
 extracting: train/cardsNew1_label_map.pbtxt
 extracting: valid/cardsNew1_label_map.pbtxt
 extracting: README.roboflow.txt
```

```
!pwd
%cd /content
!curl -L "[your down code]" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
%ls -al train/
```

```
total 3220
drwxr-xr-x 2 root root    4096 Jul 20 05:13 ./
drwxr-xr-x 1 root root    4096 Jul 20 05:13 ../
-rw-r--r-- 1 root root     376 Jul 20 05:08 cardsNew1_label_map.pbtxt
-rw-r--r-- 1 root root 3283585 Jul 20 05:08 cardsNew1.tfrecord
```

Actual Training in Colab (21)

- Configure the parameters….!

```
# NOTE: Update these TFRecord names from "cards" and "cards_label_map" to your files!  or Workers sample
test_record_fname = '/content/valid/cardsNew1.tfrecord'
train_record_fname = '/content/train/cardsNew1.tfrecord'
label_map_pbtxt_fname = '/content/train/cardsNew1_label_map.pbtxt'
```

Actual Training in Colab (22)

❖ Configure Custom TensorFlow2 Object Detection Training Configuration

- In this section you can specify any model in the TF2 OD model zoo and set up your training configuration. **Some case, the batch size is critical in training stage.**

```
##change chosen model to deploy different mode
MODELS_CONFIG = {
    'efficientdet-d0': {
        'model_name': 'efficientdet_d0_coco17_
        'base_pipeline_file': 'ssd_efficientde
        'pretrained_checkpoint': 'efficientdet
        'batch_size': 8 #16
    },
    'efficientdet-d1': {
        'model_name': 'efficientdet_d1_coco17_
        'base_pipeline_file': 'ssd_efficientde
        'pretrained_checkpoint': 'efficientdet
        'batch_size': 8 #16
    },
    'efficientdet-d2': {
        'model_name': 'efficientdet_d2_coco17_
        'base_pipeline_file': 'ssd_efficientde
        'pretrained_checkpoint': 'efficientdet
        'batch_size': 8 #16
    },
        'efficientdet-d3': {
        'model_name': 'efficientdet_d3_coco17_
        'base_pipeline_file': 'ssd_efficientde
        'pretrained_checkpoint': 'efficientdet
        'batch_size': 8 #16
    }
}

#in this tutorial we implement the lightweight
#if you want to scale up tot larger efficientd
chosen_model = 'efficientdet-d0'

num_steps = 40000 #The more steps, the longer
num_eval_steps = 500 #Perform evaluation after
```

```
##change chosen model to deploy different models available in the TF2 object detection zoo
MODELS_CONFIG = {
    'efficientdet-d0': {
        'model_name': 'efficientdet_d0_coco17_tpu-32',
        'base_pipeline_file': 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config',
        'pretrained_checkpoint': 'efficientdet_d0_coco17_tpu-32.tar.gz',
        'batch_size': 16
    },
    'efficientdet-d1': {
        'model_name': 'efficientdet_d1_coco17_tpu-32',
        'base_pipeline_file': 'ssd_efficientdet_d1_640x640_coco17_tpu-8.config',
        'pretrained_checkpoint': 'efficientdet_d1_coco17_tpu-32.tar.gz',
        'batch_size': 16
    },
    'efficientdet-d2': {
        'model_name': 'efficientdet_d2_coco17_tpu-32',
        'base_pipeline_file': 'ssd_efficientdet_d2_768x768_coco17_tpu-8.config',
        'pretrained_checkpoint': 'efficientdet_d2_coco17_tpu-32.tar.gz',
        'batch_size': 16
    },
        'efficientdet-d3': {
        'model_name': 'efficientdet_d3_coco17_tpu-32',
        'base_pipeline_file': 'ssd_efficientdet_d3_896x896_coco17_tpu-32.config',
        'pretrained_checkpoint': 'efficientdet_d3_coco17_tpu-32.tar.gz',
        'batch_size': 16
    }
}

#in this tutorial we implement the lightweight, smallest state of the art efficientdet model
#if you want to scale up tot larger efficientdet models you will likely need more compute!
chosen_model = 'efficientdet-d0'

num_steps = 10000 #The more steps, the longer the training. Increase if your loss function is still decreasing and validatio
n metrics are increasing.
num_eval_steps = 500 #Perform evaluation after so many steps

model_name = MODELS_CONFIG[chosen_model]['model_name']
pretrained_checkpoint = MODELS_CONFIG[chosen_model]['pretrained_checkpoint']
base_pipeline_file = MODELS_CONFIG[chosen_model]['base_pipeline_file']
batch_size = MODELS_CONFIG[chosen_model]['batch_size'] #if you can fit a large batch in memory, it may speed up your trainin
g
```

Actual Training in Colab (23)

- Download pre-trained weights from web...!!!

```
#download pretrained weights
%mkdir /content/models/research/deploy/
%cd /content/models/research/deploy/
import tarfile
download_tar = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/' + pretrained_checkpoint

!wget {download_tar}
tar = tarfile.open(pretrained_checkpoint)
tar.extractall()
tar.close()
```

```
/content/models/research/deploy
--2020-12-03 08:18:31--  http://download.tensorflow.org/models/object_detection/tf2/20200711/efficientdet_d0_coco17_tpu-32.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 172.253.122.128, 2607:f8b0:4004:c09::80
Connecting to download.tensorflow.org (download.tensorflow.org)|172.253.122.128|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30736482 (29M) [application/x-tar]
Saving to: 'efficientdet_d0_coco17_tpu-32.tar.gz'

efficientdet_d0_coc 100%[===================>]  29.31M  55.8MB/s    in 0.5s

2020-12-03 08:18:31 (55.8 MB/s) - 'efficientdet_d0_coco17_tpu-32.tar.gz' saved [30736482/30736482]
```


```
#download pretrained weights
%mkdir /content/models/research/deploy/
%cd /content/models/research/deploy/
import tarfile
download_tar = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/' + pretrained_checkpoint

!wget {download_tar}
tar = tarfile.open(pretrained_checkpoint)
tar.extractall()
tar.close()
```

Actual Training in Colab (24)

- Then download base training configuration file…!!!

```
#download base training configuration file
%cd /content/models/research/deploy
download_config = 'https://raw.githubusercontent.com/tensorflow/models/master/research/object_detection/configs/tf2/' + base_pipeline_file
!wget {download_config}
```

```
/content/models/research/deploy
--2020-07-20 05:24:45--  https://raw.githubusercontent.com/tensorflow/models/master/research/object_detection/configs/tf2/ssd_efficientdet_d0_512x512_coco17_tpu-8.config
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4630 (4.5K) [text/plain]
Saving to: 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config'

ssd_efficientdet_d0 100%[===================>]   4.52K  --.-KB/s    in 0s

2020-07-20 05:24:45 (60.6 MB/s) - 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config' saved [4630/4630]
```


```
#download base training configuration file
%cd /content/models/research/deploy
download_config = 'https://raw.githubusercontent.com/tensorflow/models/master/research/object_detection/configs/tf2/' + base_pi
peline_file
!wget {download_config}
```

Actual Training in Colab (25)

- Prepare some information for training

```
#prepare some information for training
pipeline_fname = '/content/models/research/deploy/' + base_pipeline_file
fine_tune_checkpoint = '/content/models/research/deploy/' + model_name + '/checkpoint/ckpt-0'

def get_num_classes(pbtxt_fname):
    from object_detection.utils import label_map_util
    label_map = label_map_util.load_labelmap(pbtxt_fname)
    categories = label_map_util.convert_label_map_to_categories(
        label_map, max_num_classes=90, use_display_name=True)
    category_index = label_map_util.create_category_index(categories)
    return len(category_index.keys())
num_classes = get_num_classes(label_map_pbtxt_fname)
```

```
#prepare
pipeline_fname = '/content/models/research/deploy/' + base_pipeline_file
fine_tune_checkpoint = '/content/models/research/deploy/' + model_name + '/checkpoint/ckpt-0'

def get_num_classes(pbtxt_fname):
    from object_detection.utils import label_map_util
    label_map = label_map_util.load_labelmap(pbtxt_fname)
    categories = label_map_util.convert_label_map_to_categories(
        label_map, max_num_classes=90, use_display_name=True)
    category_index = label_map_util.create_category_index(categories)
    return len(category_index.keys())
num_classes = get_num_classes(label_map_pbtxt_fname)
```

Actual Training in Colab (26)

- Write custom configuration file by slotti
  arameters into the base pipeline file

```
#write custom configuration file by slotting our dataset, model checkpoint, and

import re

%cd /content/models/research/deploy
print('writing custom configuration file')

with open(pipeline_fname) as f:
    s = f.read()
with open('pipeline_file.config', 'w') as f:

    # fine_tune_checkpoint
    s = re.sub('fine_tune_checkpoint: ".*?"',
               'fine_tune_checkpoint: "{}"'.format(fine_tune_checkpoint), s)

    # tfrecord files train and test.
    s = re.sub(
        '(input_path: ".*?)(PATH_TO_BE_CONFIGURED/train)(.*?")', 'input_path: "
    s = re.sub(
        '(input_path: ".*?)(PATH_TO_BE_CONFIGURED/val)(.*?")', 'input_path: "{}

    # label_map_path
    s = re.sub(
        'label_map_path: ".*?"', 'label_map_path: "{}"'.format(label_map_pbtxt_

    # Set training batch_size.
    s = re.sub('batch_size: [0-9]+',
               'batch_size: {}'.format(batch_size), s)

    # Set training steps, num_steps
    s = re.sub('num_steps: [0-9]+',
```

```
/content/models/research/deploy
writing custom configuration file
```

```
#write custom configuration file by slotting our dataset, model checkpoint, and traini
ng parameters into the base pipeline file

import re

%cd /content/models/research/deploy
print('writing custom configuration file')

with open(pipeline_fname) as f:
    s = f.read()
with open('pipeline_file.config', 'w') as f:

    # fine_tune_checkpoint
    s = re.sub('fine_tune_checkpoint: ".*?"',
               'fine_tune_checkpoint: "{}"'.format(fine_tune_checkpoint), s)

    # tfrecord files train and test.
    s = re.sub(
        '(input_path: ".*?)(PATH_TO_BE_CONFIGURED/train)(.*?")', 'input_path: "{}"'.fo
rmat(train_record_fname), s)
    s = re.sub(
        '(input_path: ".*?)(PATH_TO_BE_CONFIGURED/val)(.*?")', 'input_path: "{}"'.form
at(test_record_fname), s)

    # label_map_path
    s = re.sub(
        'label_map_path: ".*?"', 'label_map_path: "{}"'.format(label_map_pbtxt_fname),
 s)

    # Set training batch_size.
    s = re.sub('batch_size: [0-9]+',
               'batch_size: {}'.format(batch_size), s)

    # Set training steps, num_steps
    s = re.sub('num_steps: [0-9]+',
               'num_steps: {}'.format(num_steps), s)

    # Set number of classes num_classes.
    s = re.sub('num_classes: [0-9]+',
               'num_classes: {}'.format(num_classes), s)

    #fine-tune checkpoint type
    s = re.sub(
        'fine_tune_checkpoint_type: "classification"', 'fine_tune_checkpoint_type: "{}
"'.format('detection'), s)

    f.write(s)
```

Actual Training in Colab (27)

- Check on the my configuration file (pipeline_file.config).

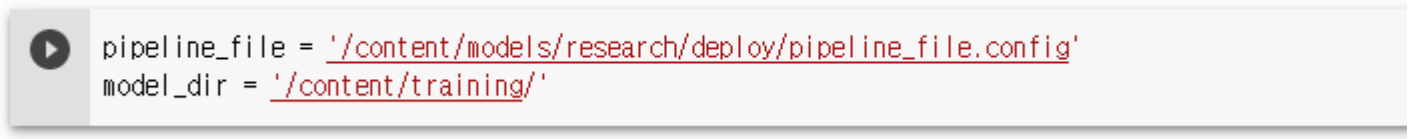
```
%cat /content/models/research/deploy/pipeline_file.config
```

```
# SSD with EfficientNet-b0 + BiFPN feature extractor,
# shared box predictor and focal loss (a.k.a EfficientDet-d0).
# See EfficientDet, Tan et al, https://arxiv.org/abs/1911.09070
# See Lin et al, https://arxiv.org/abs/1708.02002
# Trained on COCO, initialized from an EfficientNet-b0 checkpoint.
#
# Train on TPU-8

model {
  ssd {
    inplace_batchnorm_update: true
    freeze_batchnorm: false
    num_classes: 6
    add_background_class: false
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
```

```
%cat /content/models/research/deploy/pipeline_file.config
```

Actual Training in Colab (28)

- Set configure path and training directory…!!!




```
pipeline_file = '/content/models/research/deploy/pipeline_file.config'
model_dir = '/content/training/'
```

Actual Training in Colab (29)

❖ Training Custom TF2 Object Detector

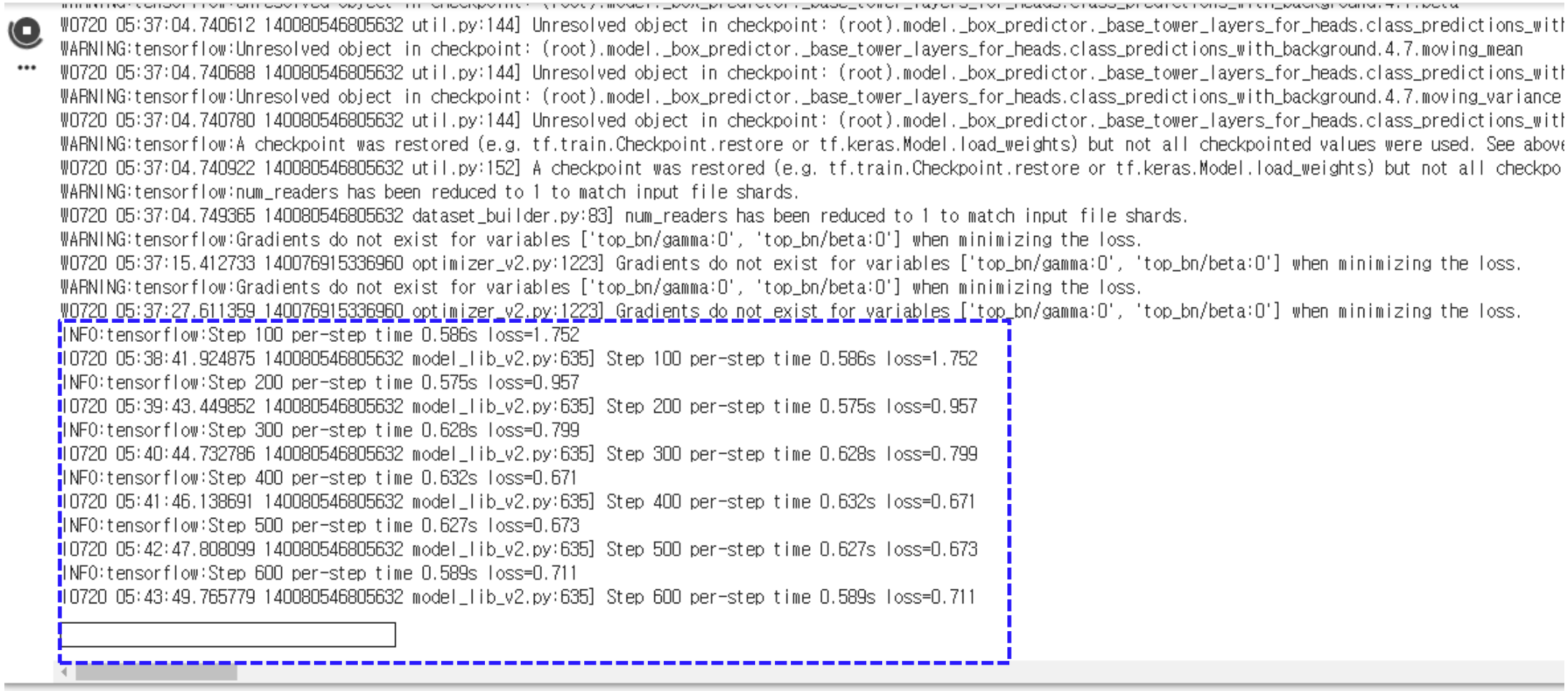
- pipeline_file: defined above in writing custom training configuration
- model_dir: the location tensorboard logs and saved model checkpoints will save to
- num_train_steps: how long to train for
- num_eval_steps: perform eval on validation set after this many steps

```
!python /content/models/research/object_detection/model_main_tf2.py \
    --pipeline_config_path={pipeline_file} \
    --model_dir={model_dir} \
    --alsologtostderr \
    --num_train_steps=10000 \
    --sample_1_of_n_eval_examples=1 \
    --num_eval_steps={num_eval_steps}
```

```
2020-07-20 05:36:01.519593: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519623: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519649: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519673: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519696: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519721: I tensorflow/stream_executor/pla
2020-07-20 05:36:01.519807: I tensorflow/stream_executor/cu
2020-07-20 05:36:01.520422: I tensorflow/stream_executor/cu
2020-07-20 05:36:01.520933: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703] Adding visible gpu devices: 0
2020-07-20 05:36:01.520986: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcuda
2020-07-20 05:36:02.043841: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1
2020-07-20 05:36:02.043923: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]      0
2020-07-20 05:36:02.043947: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] 0:   N
2020-07-20 05:36:02.044214: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node read from SysFS had negati
2020-07-20 05:36:02.044824: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node read from SysFS had negati
2020-07-20 05:36:02.045372: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:39] Overriding allow_growth setting because the T
2020-07-20 05:36:02.045421: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247] Created TensorFlow device (/job:localhost/replica:
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:GPU:0',)
I0720 05:36:02.047699 140080546805632 mirrored_strategy.py:500] Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/d
INFO:tensorflow:Maybe overwriting train_steps: 10000
```

Actual Training in Colab (30)

- Training phase:



For each 100 step, you can see the consumed time of 0.56 min.
Then how much time of 10,000 steps?

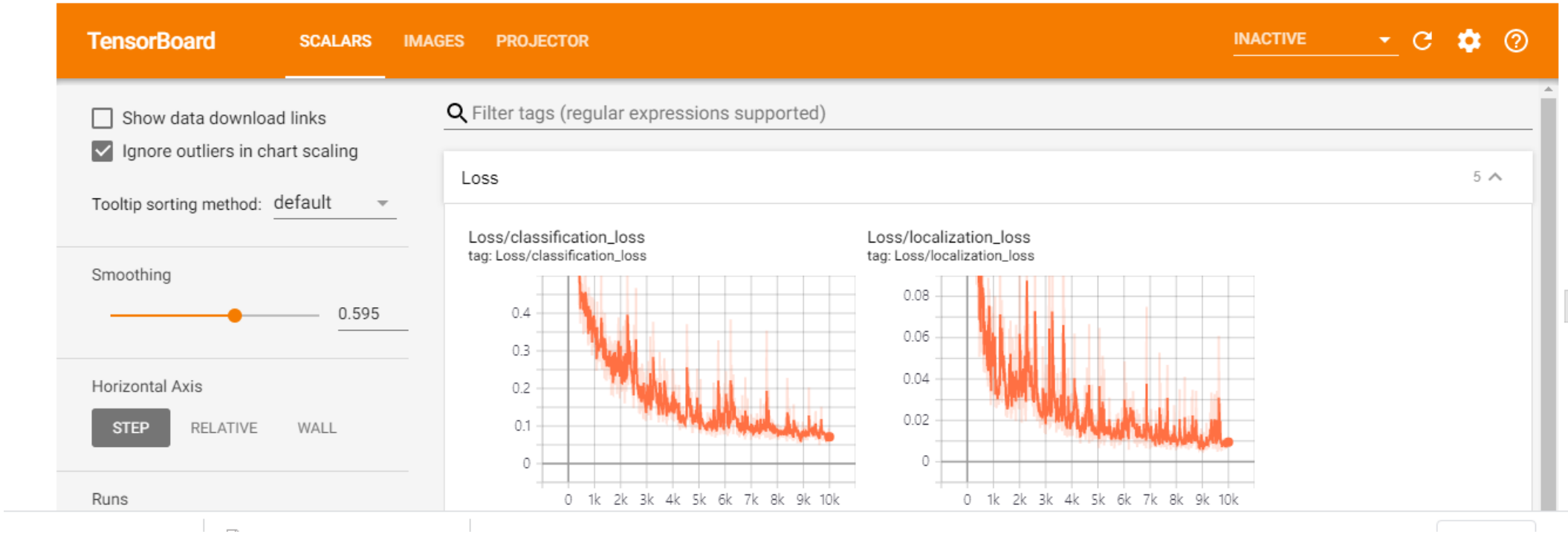
Actual Training in Colab (31)

- **Monitering the training process...!!! Using Tensorboard.**

```
[ ]  %load_ext tensorboard
     %tensorboard --logdir '/content/training/train'
```

```
%load_ext tensorboard
%tensorboard --logdir '/content/training/train'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```



Actual Training in Colab (32)

❖ Exporting a Trained Inference Graph

- Check and see where our model saved weights

```
#see where our model saved weights
%ls '/content/training/'
```

```
checkpoint                  ckpt-6.index
ckpt-3.data-00000-of-00002  ckpt-7.data-00000-of-00002
ckpt-3.data-00001-of-00002  ckpt-7.data-00001-of-00002
ckpt-3.index                ckpt-7.index
ckpt-4.data-00000-of-00002  ckpt-8.data-00000-of-00002
ckpt-4.data-00001-of-00002  ckpt-8.data-00001-of-00002
ckpt-4.index                ckpt-8.index
ckpt-5.data-00000-of-00002  ckpt-9.data-00000-of-00002
ckpt-5.data-00001-of-00002  ckpt-9.data-00001-of-00002
ckpt-5.index                ckpt-9.index
ckpt-6.data-00000-of-00002  eval/
ckpt-6.data-00001-of-00002  train/
```


```
%ls '/content/training/'
```

Actual Training in Colab (33)

- Run the converting (output_dir & model_path)

```
#run conversion script
import re
import numpy as np

output_directory = '/content/fine_tuned_model'

#place the model weights you would like to export here
last_model_path = '/content/training/'
print(last_model_path)
!python /content/models/research/object_detection/exporter_main
    --trained_checkpoint_dir {last_model_path} ₩
    --output_directory {output_directory} ₩
    --pipeline_config_path {pipeline_file}
```

```
#run conversion script
import re
import numpy as np

output_directory = '/content/fine_tuned_model'

#place the model weights you would like to export here
last_model_path = '/content/training/'
print(last_model_path)
!python /content/models/research/object_detection/exporter_main_v2.py \
    --trained_checkpoint_dir {last_model_path} \
    --output_directory {output_directory} \
    --pipeline_config_path {pipeline_file}
```

```
/content/training/
I0719 16:45:00.845904 139782512555904 ssd_efficientnet_bifpn_feature_extractor.py:144] EfficientDet Efficient
I0719 16:45:00.846156 139782512555904 ssd_efficientnet_bifpn_feature_extractor.py:145] EfficientDet BiFPN num
I0719 16:45:00.846297 139782512555904 ssd_efficientnet_bifpn_feature_extractor.py:147] EfficientDet BiFPN num
I0719 16:45:00.856328 139782512555904 efficientnet_model.py:146] round_filter input=32 output=32
2020-07-19 16:45:00.870337: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully oper
2020-07-19 16:45:00.882530: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node
2020-07-19 16:45:00.883257: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with prop
pciBusID: 0000:00:04.0 name: Tesla K80 computeCapability: 3.7
INFO:tensorflow:Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.UnknownArgument object at 0x7f775246f7b8>, TensorS
I1203 12:49:00.714985 140153870641024 def_function.py:1038] Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.Unknow
INFO:tensorflow:Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.UnknownArgument object at 0x7f7751ef2ef0>, TensorS
I1203 12:49:00.715408 140153870641024 def_function.py:1038] Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.Unknow
INFO:tensorflow:Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.UnknownArgument object at 0x7f7751e7b6a0>, TensorS
I1203 12:49:00.715612 140153870641024 def_function.py:1038] Unsupported signature for serialization: ([[(<tensorflow.python.framework.func_graph.Unknow
INFO:tensorflow:Assets written to: /content/fine_tuned_model/saved_model/assets
I1203 12:49:02.216453 140153870641024 builder_impl.py:775] Assets written to: /content/fine_tuned_model/saved_model/assets
INFO:tensorflow:Writing pipeline config file to /content/fine_tuned_model/pipeline.config
I1203 12:49:03.616149 140153870641024 config_util.py:254] Writing pipeline config file to /content/fine_tuned_model/pipeline.config
```

IVPL Intelligent Vision Processing Lab

Actual Training in Colab (34)

- Check on "saved_model" folder.

```
%ls -al '/content/fine_tuned_model/saved_model/'
```

```
[ ] %ls -al '/content/fine_tuned_model/saved_model/'

total 21680
drwxr-xr-x 4 root root     4096 Jul 19 16:47 ./
drwxr-xr-x 4 root root     4096 Jul 19 16:47 ../
drwxr-xr-x 2 root root     4096 Jul 19 16:46 assets/
-rw-r--r-- 1 root root 22183667 Jul 19 16:47 saved_model.pb
drwxr-xr-x 2 root root     4096 Jul 19 16:46 variables/
```

```
%ls -al '/content/fine_tuned_model/saved_model/variables/'
```

```
[ ] %ls -al '/content/fine_tuned_model/saved_model/variables/'

total 22140
drwxr-xr-x 2 root root     4096 Dec  3 12:49 ./
drwxr-xr-x 4 root root     4096 Dec  3 12:49 ../
-rw-r--r-- 1 root root 22621597 Dec  3 12:49 variables.data-00000-of-00001
-rw-r--r-- 1 root root    39311 Dec  3 12:49 variables.index
```

Actual Training in Colab (35)

❖ Download pb and ~label.pbtxt to your local PC..!!!

```
[ ]  import os
     output_directory = '/content/fine_tuned_model/saved_model/'
     pb_fname = os.path.join(os.path.abspath(output_directory), "saved_model.pb")
     assert os.path.isfile(pb_fname), '`{}` not exist'.format(pb_fname)
```

```
[ ]  !ls -alh {pb_fname}
```

```
-rw-r--r-- 1 root root 22M Jul 20 08:00 /content/fine_tuned_model/saved_model/saved_model.pb
```

```
import os
output_directory = '/content/fine_tuned_model/saved_model/'
pb_fname = os.path.join(os.path.abspath(output_directory), "saved_model.pb")
assert os.path.isfile(pb_fname), '`{}` not exist'.format(pb_fname)
```

Actual Training in Colab (36)

```
[ ]  from google.colab import files
     files.download(pb_fname)
```

```
from google.colab import files
files.download(pb_fname)
```

```
[ ]  %ls -al /content/train/
```

```
total 3220
drwxr-xr-x 2 root root    4096 Jul 20 05:13 ./
drwxr-xr-x 1 root root    4096 Jul 20 07:59 ../
-rw-r--r-- 1 root root     376 Jul 20 05:08 cardsNew1_label_map.pbtxt
-rw-r--r-- 1 root root 3283585 Jul 20 05:08 cardsNew1.tfrecord
```

```
[ ]  label_map_pbtxt_fname = '/content/train/cardsNew1_label_map.pbtxt'

     from google.colab import files
     files.download(label_map_pbtxt_fname)
```

```
label_map_pbtxt_fname = '/content/train/cardsNew1_label_map.pbtxt'

from google.colab import files
files.download(label_map_pbtxt_fname)
```

Actual Training in Colab (37)

- Compress the trained model and weights for downloading to your local PC..!!!

```
%pwd
!tar -cvzf trained_model_large_original_20000.tar.gz /content/fine_tuned_model/
```

```
%pwd
!tar -cvzf trained_model_large_original_20000.tar.gz /content/fine_tuned_model/

tar: Removing leading `/' from member names
/content/fine_tuned_model/
/content/fine_tuned_model/pipeline.config
/content/fine_tuned_model/checkpoint/
/content/fine_tuned_model/checkpoint/checkpoint
/content/fine_tuned_model/checkpoint/ckpt-0.index
/content/fine_tuned_model/checkpoint/ckpt-0.data-00000-of-00001
/content/fine_tuned_model/saved_model/
/content/fine_tuned_model/saved_model/saved_model.pb
/content/fine_tuned_model/saved_model/assets/
/content/fine_tuned_model/saved_model/variables/
/content/fine_tuned_model/saved_model/variables/variables.index
/content/fine_tuned_model/saved_model/variables/variables.data-00000-of-00001
```

```
%ls -al

total 3481148
drwxr-xr-x  3 root   root        4096 Dec  3 13:00 ./
drwxr-xr-x 24 root   root        4096 Dec  3 08:18 ../
drwxr-xr--  4 345018 89939       4096 Jul 11 00:12 efficientdet_d0_coco17_tpu-32/
-rw-r--r--  1 root   root    30736482 Jul 11 00:33 efficientdet_d0_coco17_tpu-32.tar.gz
-rw-r--r--  1 root   root        4646 Dec  3 08:18 pipeline_file.config
-rw-r--r--  1 root   root        4630 Dec  3 08:18 ssd_efficientdet_d0_512x512_coco17_tpu-8.config
-rw-r--r--  1 root   root  3503623332 Dec  3 12:15 trained_model_ckpt_large_20000.tar.gz
-rw-r--r--  1 root   root    30297138 Dec  3 13:00 trained_model_large_original_20000.tar.gz
```

Actual Training in Colab (38)

- Download **the compressed file of the trained model and weight**...!!!

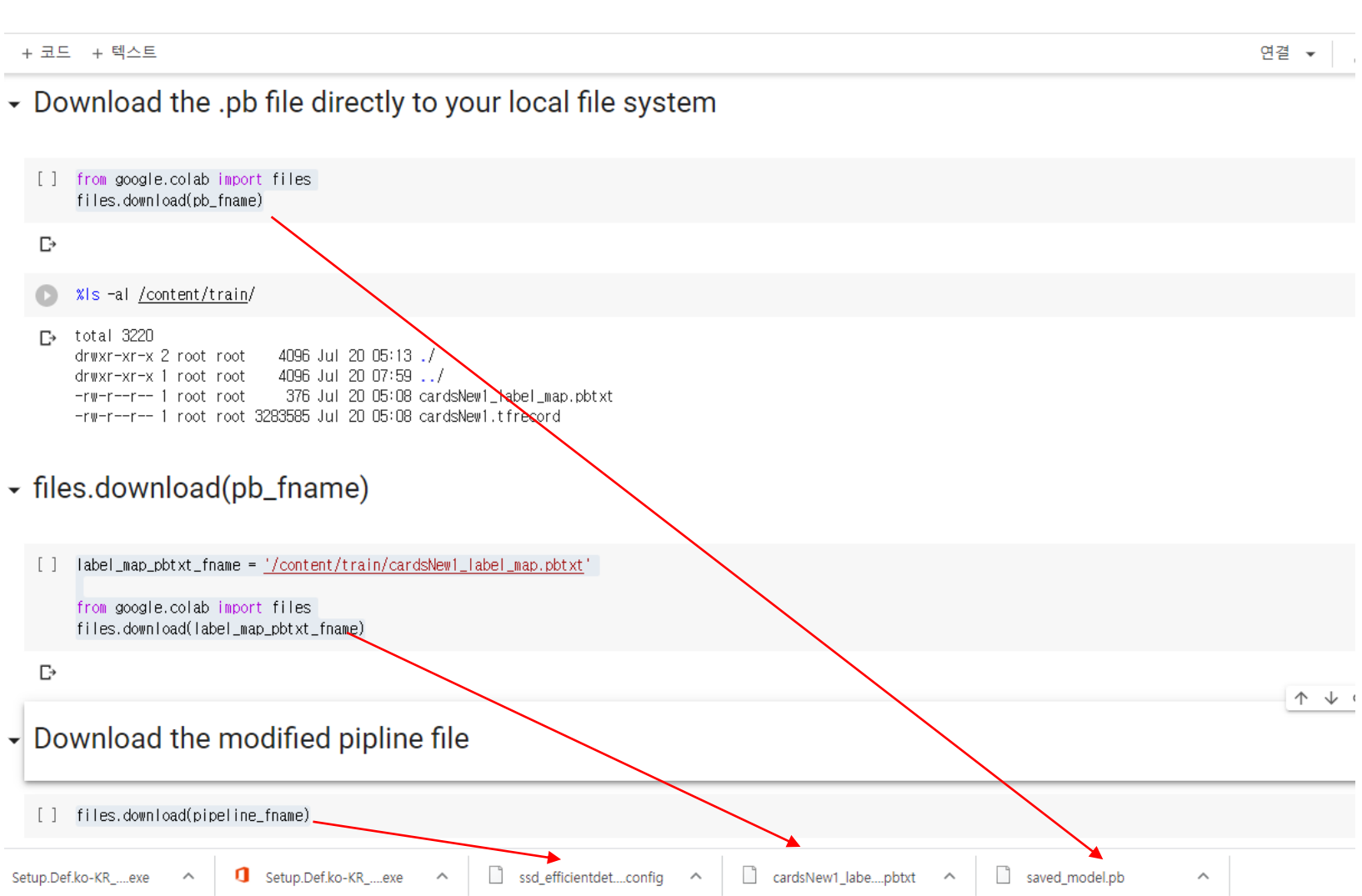
```
from google.colab import files
files.download('trained_model_large_original_20000.tar.gz')
```

Actual Training in Colab (39)

- Download the modified pipline file

```
[ ]  files.download(pipeline_fname)
```

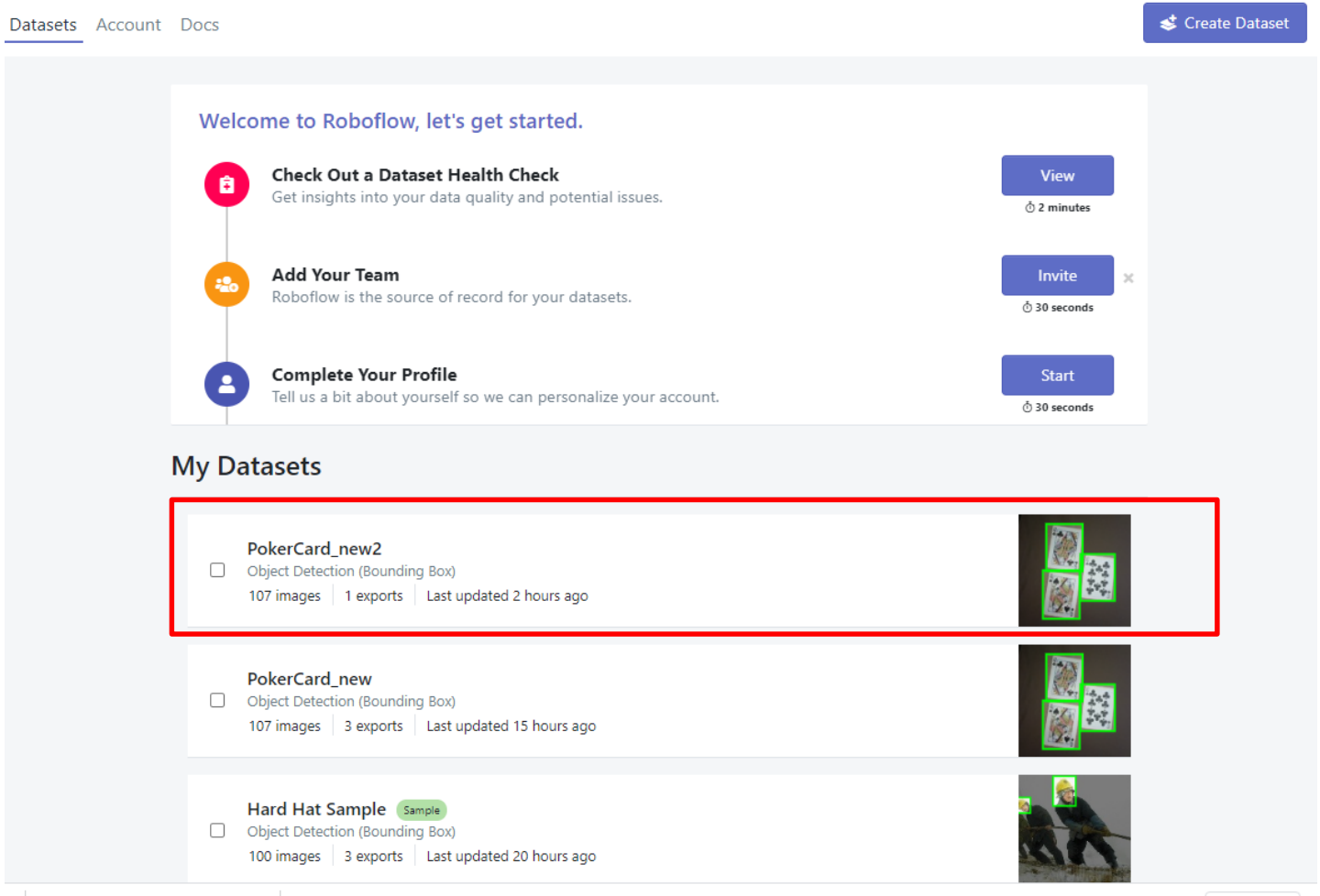
```
files.download(pipeline_fname)
```



Actual Training in Colab (40)

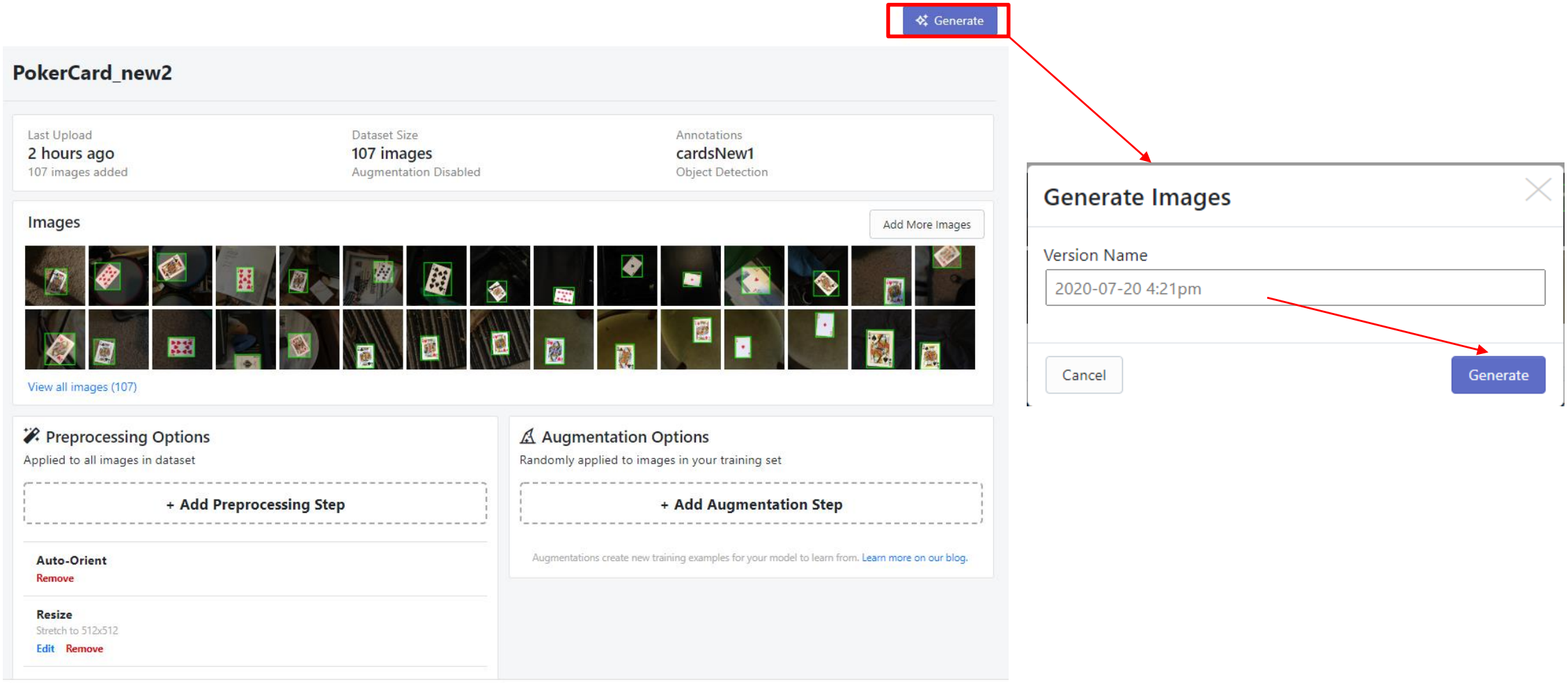
❖ Run Inference on Test Images with Custom TensorFlow2 Object Detector

- Download test images from RoBoflow: CoCo JSON Format
  - How to make CoCo JSON format?
    - 1] Select your dataset in Datasets.



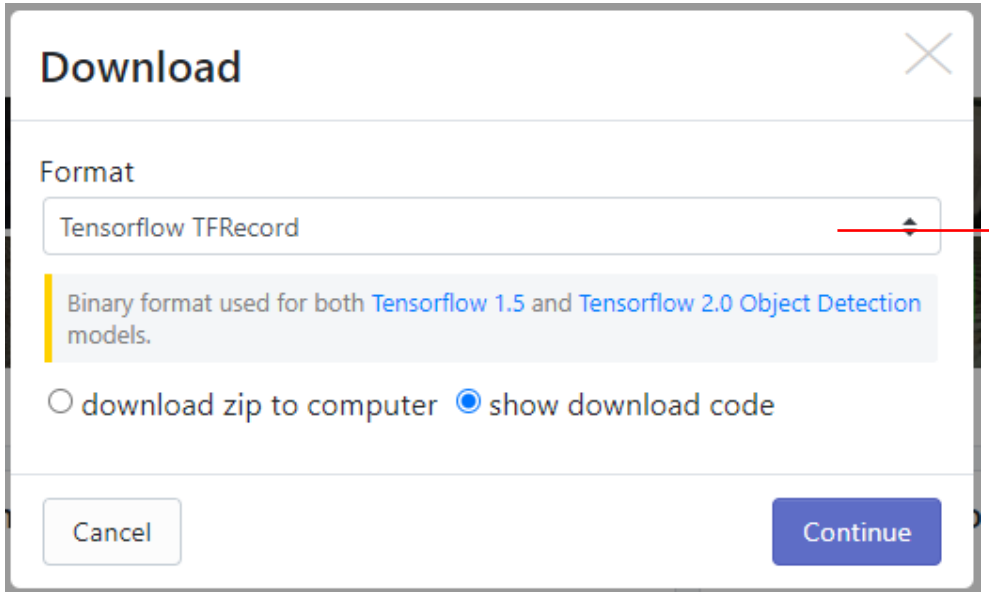
Actual Training in Colab (41)

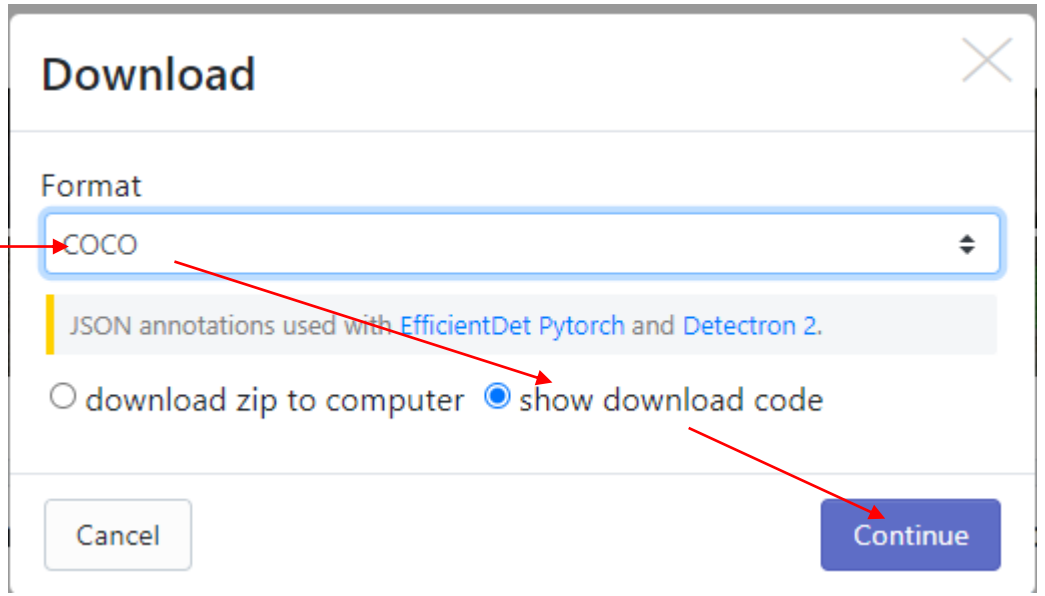
– 2] For your dataset, Click "Generate" button!

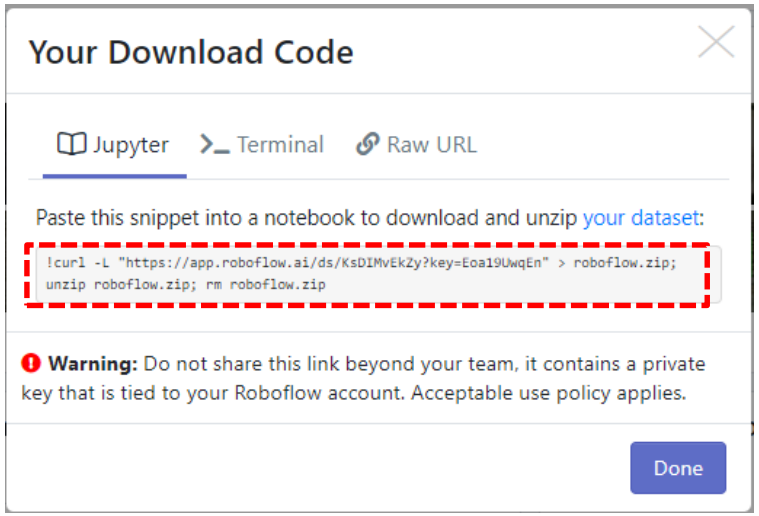


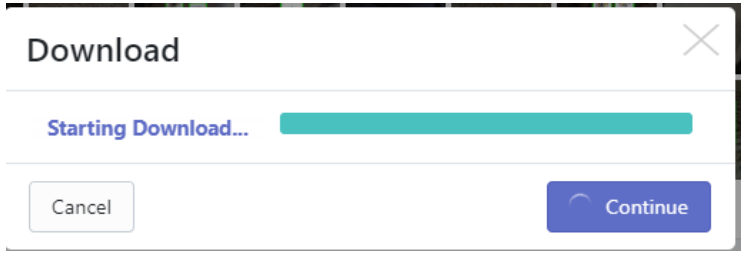
Actual Training in Colab (42)

– 3] In "Download" window, select "JASON-CoCo" type. Then you can see your download code using "curl" command.









Actual Training in Colab (43)

- Download test images from RoBoflow: CoCo JSON Format

```
#downloading test images from Roboflow
#export dataset above with format COCO JSON
#or import your test images via other means.
%mkdir /content/test/
%cd /content/test/
!curl -L "https://app.roboflow.ai/ds/WRFaCYcQsR?key=UiAENpEgms" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
mkdir: cannot create directory ‘/content/test/’ : File exists
/content/test
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   887  100   887    0     0   1107      0 --:--:-- --:--:-- --:--:--  1107
100 4527k  100 4527k    0     0  4710k      0 --:--:-- --:--:-- --:--:-- 4710k
Archive:  roboflow.zip
 extracting: test/IMG_2398_JPG.rf.f87421f41c5ba7993a423813fab08e09.jpg
 extracting: test/cam_image20_jpg.rf.80c13a21b69d5a1c53db71892752b8a8.jpg
 extracting: train/IMG_2426_JPG.rf.066860dd0c43d7385db46dae1cd1b84f.jpg
 extracting: test/cam_image44_jpg.rf.463e4963b3a251da667e0ed7557d9e27.jpg
 extracting: train/IMG_2389_JPG.rf.167d75efb8622fc0a5fc990fb3c6df8a.jpg
 extracting: test/IMG_2456_JPG.rf.6271745927c3badf98d2c5b942ab63da.jpg
 extracting: test/cam_image54_jpg.rf.68d95bae69faf9224740acfb3ccc82ef.jpg
 extracting: test/IMG_2425_JPG.rf.7fb59d723e4863a348e154a3784686d2.jpg
 extracting: train/IMG_2409_JPG.rf.23008a94a730890aed027797f27881be.jpg
 extracting: test/cam_image37_jpg.rf.dba4c9fe775aa6d7a5eb592141c707cd.jpg
 extracting: train/cam_image13_jpg.rf.003e686f07ae19f6aa20b3f0e5b23658.jpg
 extracting: train/cam_image36_jpg.rf.0ab2745372112c34cf73c064d5245803.jpg
```

```
%mkdir /content/test/
%cd /content/test/
!curl -L "https://app.roboflow.ai/ds/KsDIMvEkZy?key=Eoa19UwqEn" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

Actual Training in Colab (44)

- Import modules…!!

```
[23]  import matplotlib
      import matplotlib.pyplot as plt

      import io
      import scipy.misc
      import numpy as np
      from six import BytesIO
      from PIL import Image, ImageDraw, ImageFont

      import tensorflow as tf

      from object_detection.utils import label_map_util
      from object_detection.utils import config_util
      from object_detection.utils import visualization_utils as viz_utils
      from object_detection.builders import model_builder

      %matplotlib inline
```

```
import matplotlib
import matplotlib.pyplot as plt

import io
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont

import tensorflow as tf

from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder

%matplotlib inline
```

Actual Training in Colab (45)

- Define load function

```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  Puts image into numpy array to feed into tensorflow graph.
  Note that by convention we put it into a numpy array with shape
  (height, width, channels), where channels=3 for RGB.

  Args:
    path: the file path to the image

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)
```


```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  Puts image into numpy array to feed into tensorflow graph.
  Note that by convention we put it into a numpy array with shape
  (height, width, channels), where channels=3 for RGB.

  Args:
    path: the file path to the image

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)
```

Actual Training in Colab (46)

- Check on"ckpt-xx" in "training" folder.....!!!

```
%ls  '/content/training/'
```

```
checkpoint                      ckpt-6.index
ckpt-10.data-00000-of-00001     ckpt-7.data-00000-of-00001
ckpt-10.index                   ckpt-7.index
ckpt-11.data-00000-of-00001     ckpt-8.data-00000-of-00001
ckpt-11.index                   ckpt-8.index
ckpt-5.data-00000-of-00001      ckpt-9.data-00000-of-00001
ckpt-5.index                    ckpt-9.index
ckpt-6.data-00000-of-00001      train/
```


```
%ls  '/content/training/'
```

Actual Training in Colab (47)

- Restore our saved model: -model_dir/ckpt.restore(-): Set max ckpt-xx

```
#recover our saved model
pipeline_config = pipeline_file
#generally you want to put the last ckpt from training in here
model_dir = '/content/training/ckpt-11'
configs = config_util.get_configs_from_pipeline_file(pipeline_config)
model_config = configs['model']
detection_model = model_builder.build(
      model_config=model_config, is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(
      model=detection_model)
ckpt.restore(os.path.join('/content/training/ckpt-11'))


def get_model_detection_function(model):
  """Get a tf.function for detection."""

  @tf.function
  def detect_fn(image):
    """Detect objects in image."""

    image, shapes = model.preprocess(image)
    prediction_dict = model.predict(image, shapes)
    detections = model.postprocess(prediction_dict, shapes)

    return detections, prediction_dict, tf.reshape(shapes, [-1])

  return detect_fn

detect_fn = get_model_detection_function(detection_model)
```

```
#recover our saved model
pipeline_config = pipeline_file
#generally you want to put the last ckpt from training in here
model_dir = '/content/training/ckpt-11'
configs = config_util.get_configs_from_pipeline_file(pipeline_config)
model_config = configs['model']
detection_model = model_builder.build(
      model_config=model_config, is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(
      model=detection_model)
ckpt.restore(os.path.join('/content/training/ckpt-11'))


def get_model_detection_function(model):
  """Get a tf.function for detection."""

  @tf.function
  def detect_fn(image):
    """Detect objects in image."""

    image, shapes = model.preprocess(image)
    prediction_dict = model.predict(image, shapes)
    detections = model.postprocess(prediction_dict, shapes)

    return detections, prediction_dict, tf.reshape(shapes, [-1])

  return detect_fn

detect_fn = get_model_detection_function(detection_model)
```

Actual Training in Colab (48)

- Preparing the map labels for inference decoding….!

```
[31]  #map labels for inference decoding
      label_map_path = configs['eval_input_config'].label_map_path
      label_map = label_map_util.load_labelmap(label_map_path)
      categories = label_map_util.convert_label_map_to_categories(
          label_map,
          max_num_classes=label_map_util.get_max_label_map_index(label_map),
          use_display_name=True)
      category_index = label_map_util.create_category_index(categories)
      label_map_dict = label_map_util.get_label_map_dict(label_map, use_display_name=True)
```

```
#map labels for inference decoding
label_map_path = configs['eval_input_config'].label_map_path
label_map = label_map_util.load_labelmap(label_map_path)
categories = label_map_util.convert_label_map_to_categories(
    label_map,
    max_num_classes=label_map_util.get_max_label_map_index(label_map),
    use_display_name=True)
category_index = label_map_util.create_category_index(categories)
label_map_dict = label_map_util.get_label_map_dict(label_map, use_display_name=True)
```

Actual Training in Colab (49)

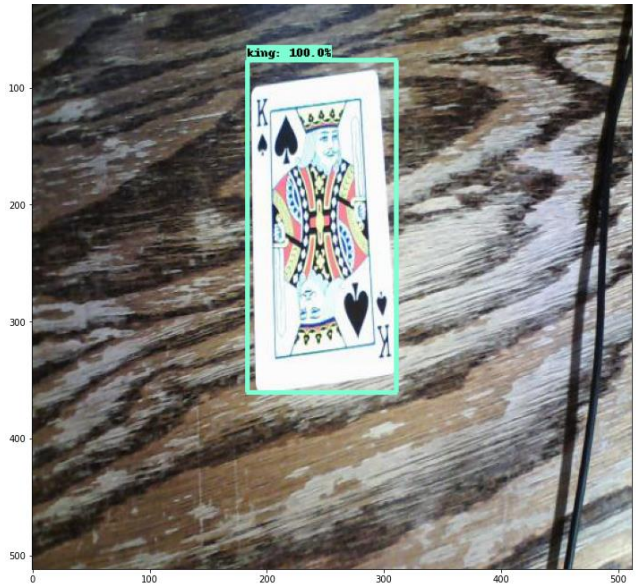
- Run inference with test images.

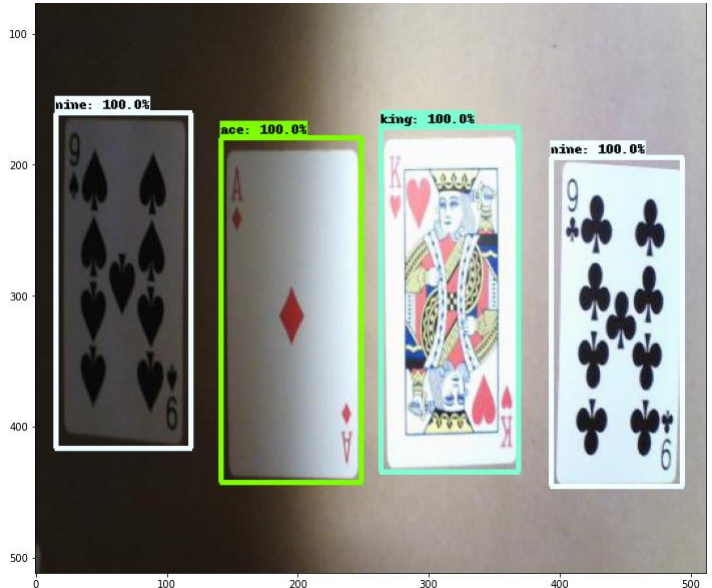
```
#run detector on test image
#it takes a little longer on the first run and then runs at normal speed.
import random

TEST_IMAGE_PATHS = glob.glob('/content/test/test/*.jpg')
image_path = random.choice(TEST_IMAGE_PATHS)
image_np = load_image_into_numpy_array(image_path)

# Things to try:
# Flip horizontally
# image_np = np.fliplr(image_np).copy()

# Convert image to grayscale
# image_np = np.tile(
#     np.mean(image_np, 2, keepdims=True), (1, 1, 3)).astype(np.uint8)

input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np, 0), dtype=tf.float32)
detections, predictions_dict, shapes = detect_fn(input_tensor)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'][0].numpy(),
      (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
      detections['detection_scores'][0].numpy(),
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=200,
      min_score_thresh=.5,
      agnostic_mode=False,
)
```

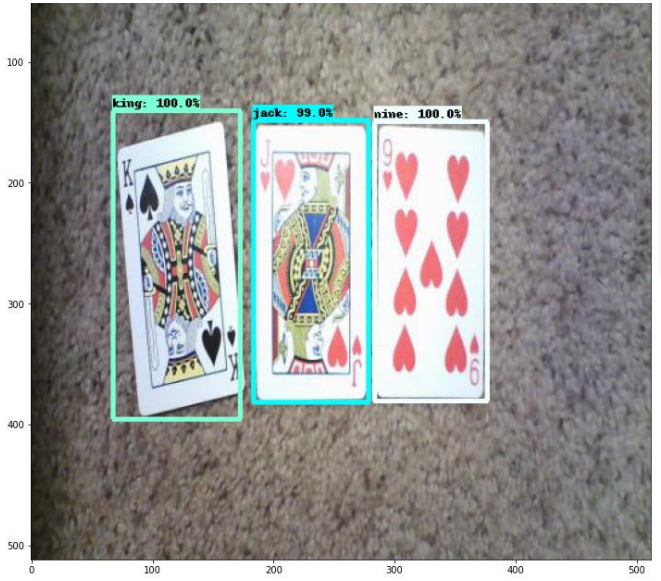
```
#run detector on test image
#it takes a little longer on the first run and then runs at normal speed.
import random
import glob

TEST_IMAGE_PATHS = glob.glob('/content/test/test/*.jpg')
image_path = random.choice(TEST_IMAGE_PATHS)
image_np = load_image_into_numpy_array(image_path)

# Things to try:
# Flip horizontally
# image_np = np.fliplr(image_np).copy()

# Convert image to grayscale
# image_np = np.tile(
#     np.mean(image_np, 2, keepdims=True), (1, 1, 3)).astype(np.uint8)

input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np, 0), dtype=tf.float32)
detections, predictions_dict, shapes = detect_fn(input_tensor)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'][0].numpy(),
      (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
      detections['detection_scores'][0].numpy(),
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=200,
      min_score_thresh=.5,
      agnostic_mode=False,
)

plt.figure(figsize=(12,16))
plt.imshow(image_np_with_detections)
plt.show()
```

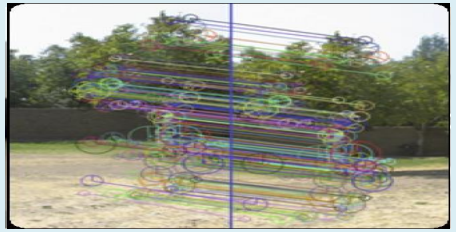
Actual Training in Colab (50)

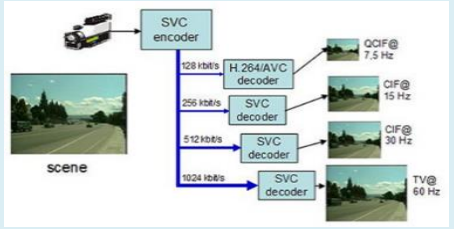
- You can see some inference results....!!! Congrat..!!!











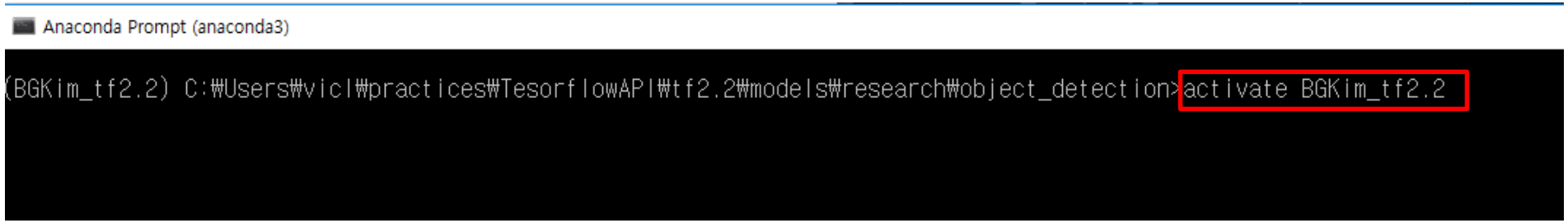
Contents

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

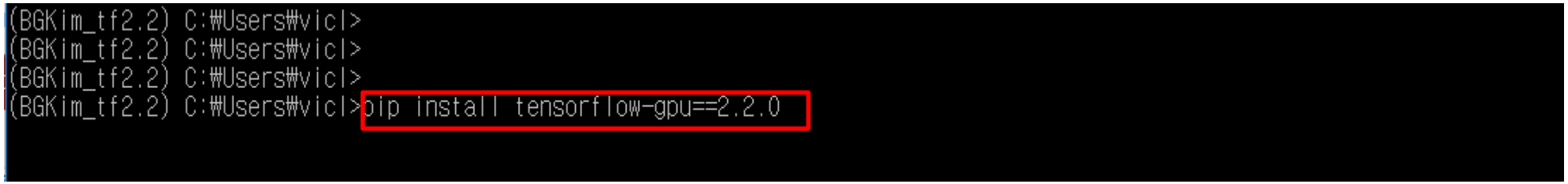
Change your inference model into your local webcam...!!!! (1)

❖ Install tensorflow 2.2 (latest version)

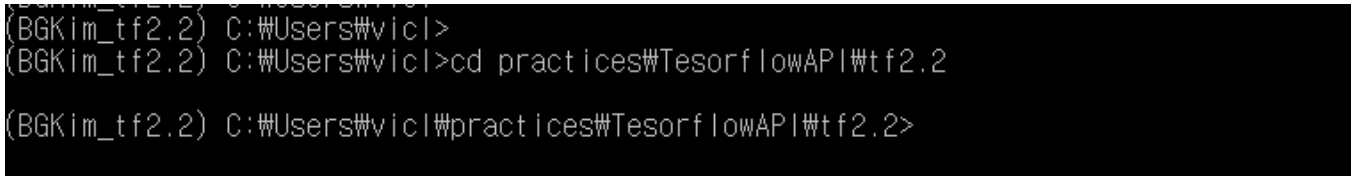
- 1] Run Anaconda prompt window.
- 2] Activate your env. (>>activate “your account name” (enter))



- 3] Install tensorflow2.2 API

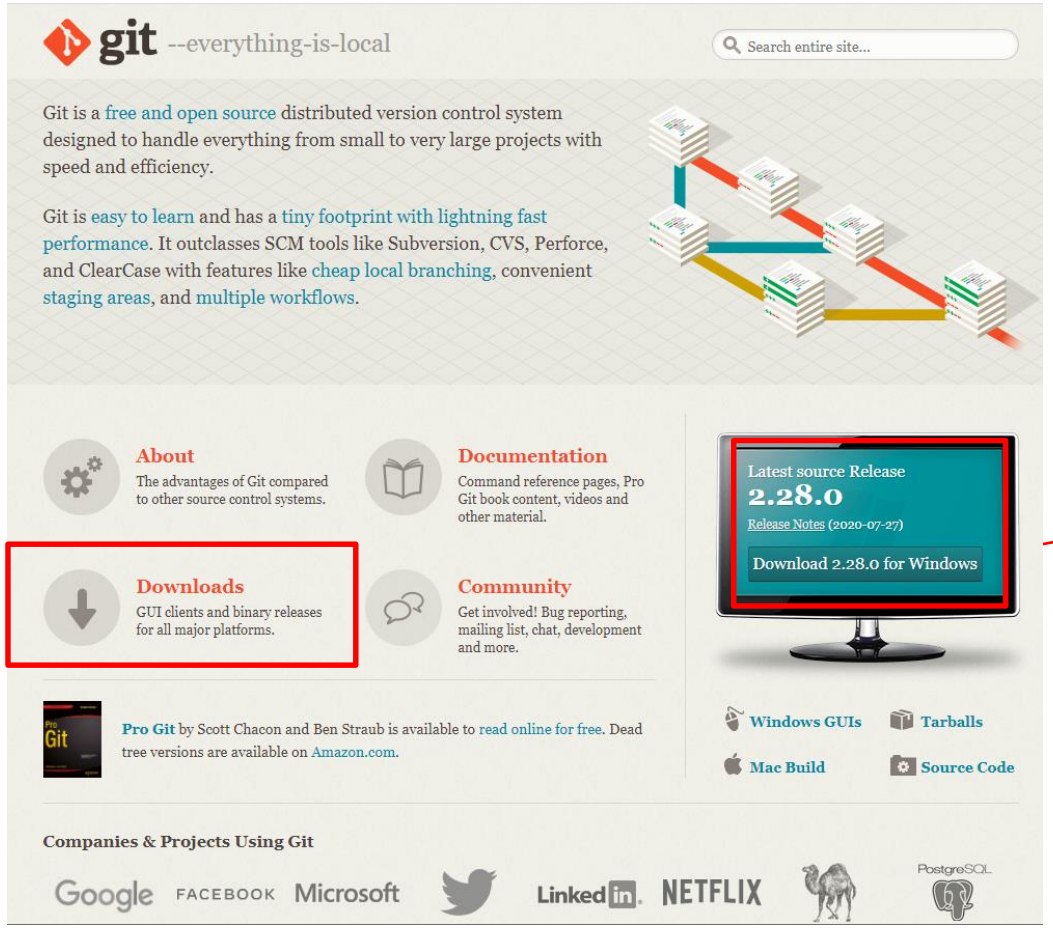


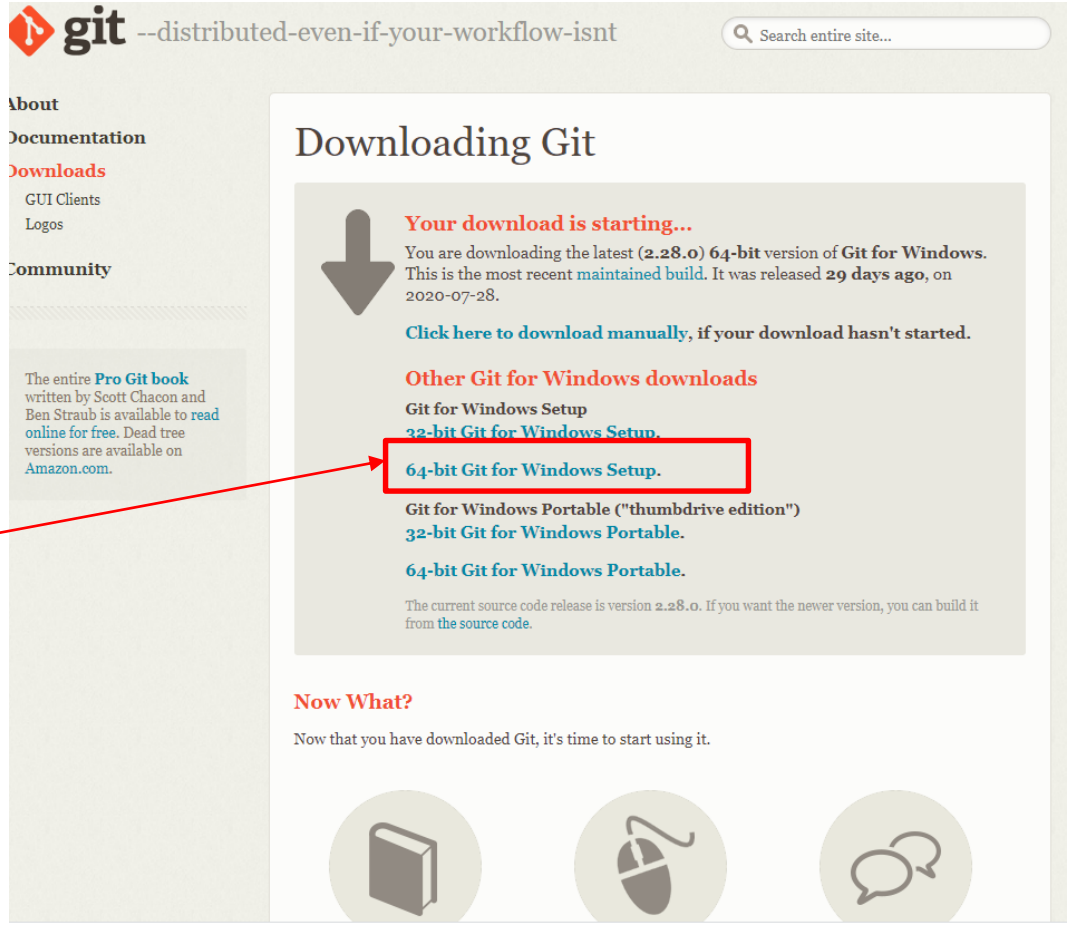
- 4] Go to your folder where you want to make the source building



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

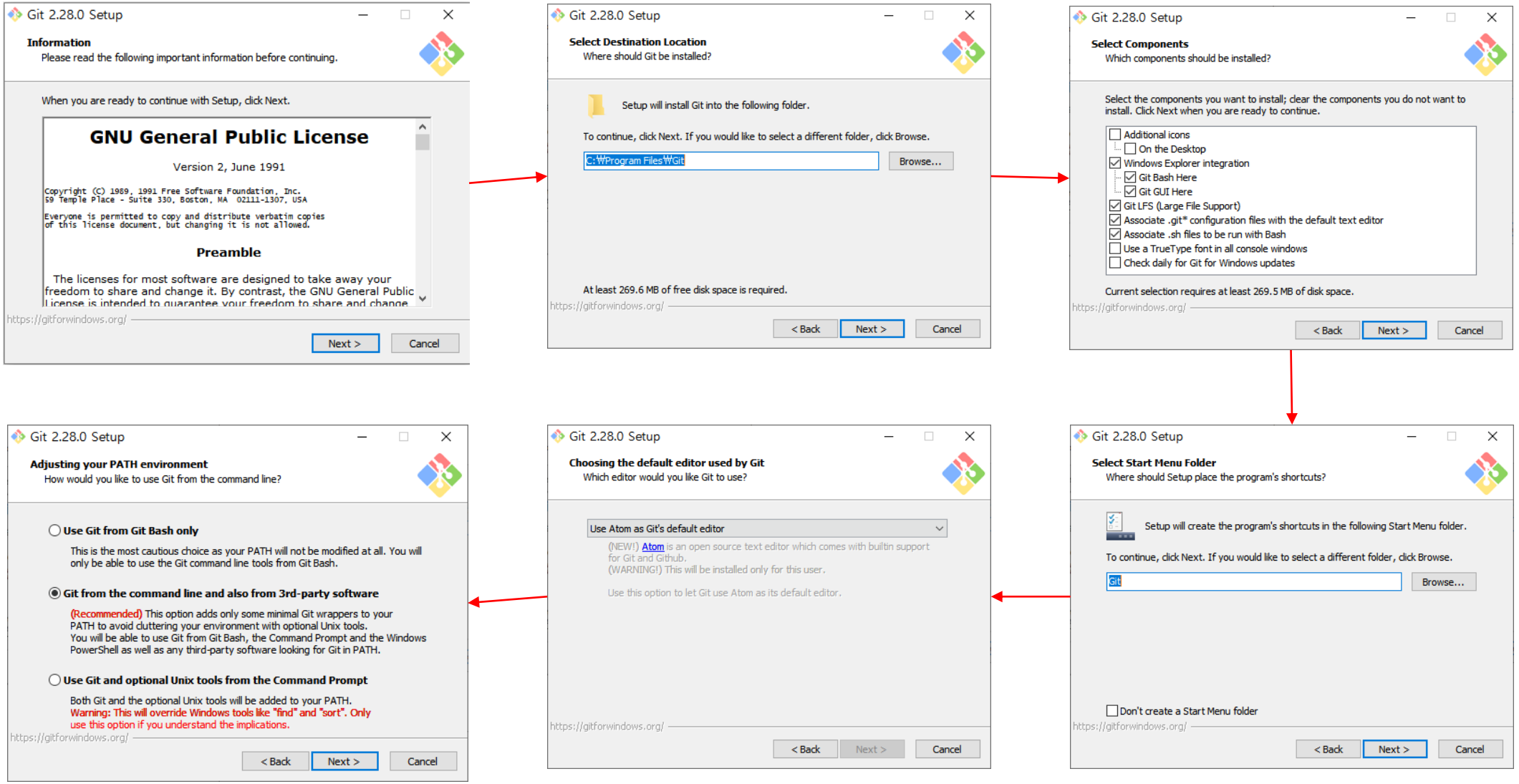
- Windows 10 git installation.
  - 1] connect to https://git-scm.com/ and download install exe file.



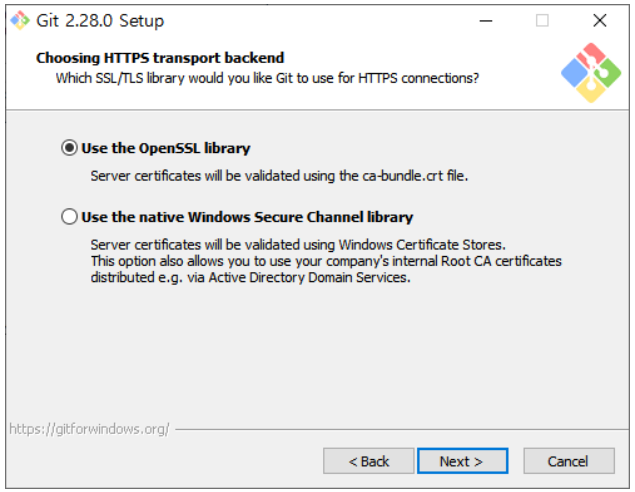


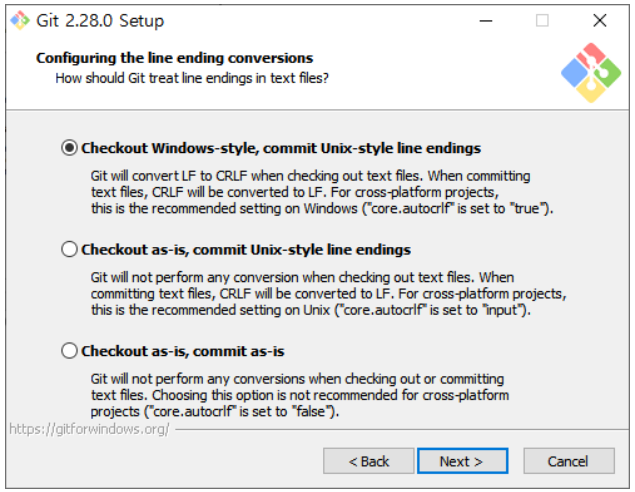
Change your inference model into your local webcam...!!!! (1-2)

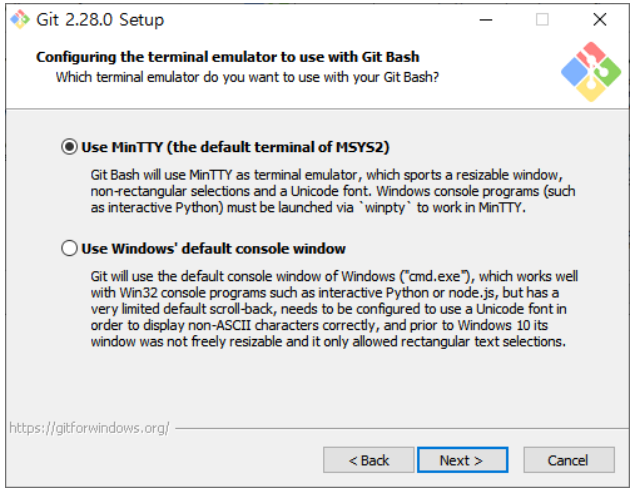
- 2] Run the install exe file as:

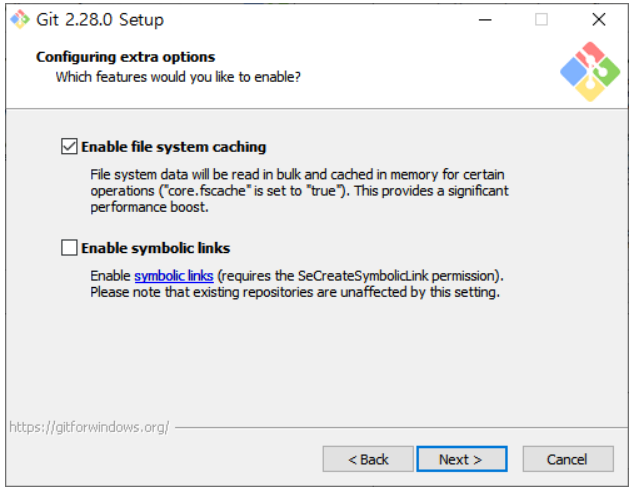


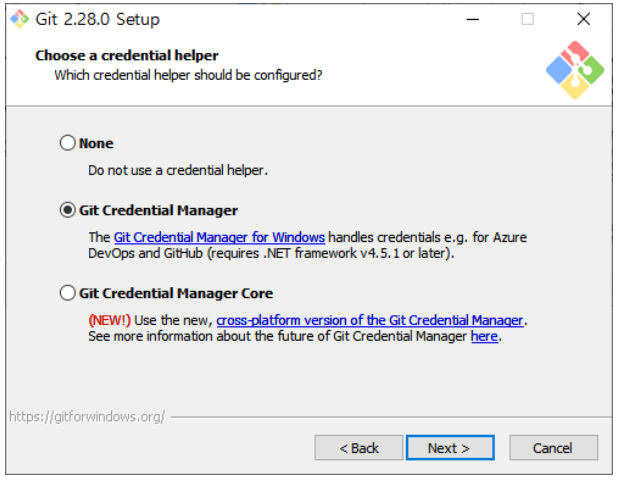
Change your inference model into your local webcam...!!!! (1-3)

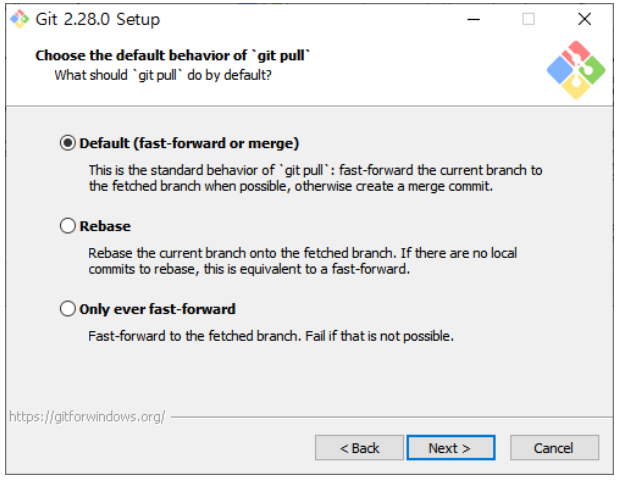




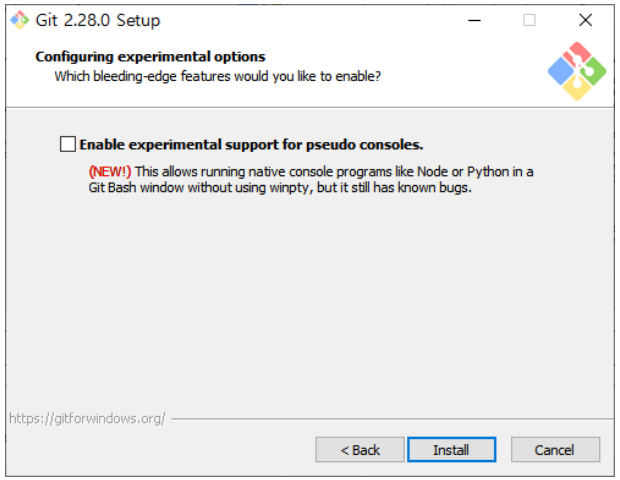


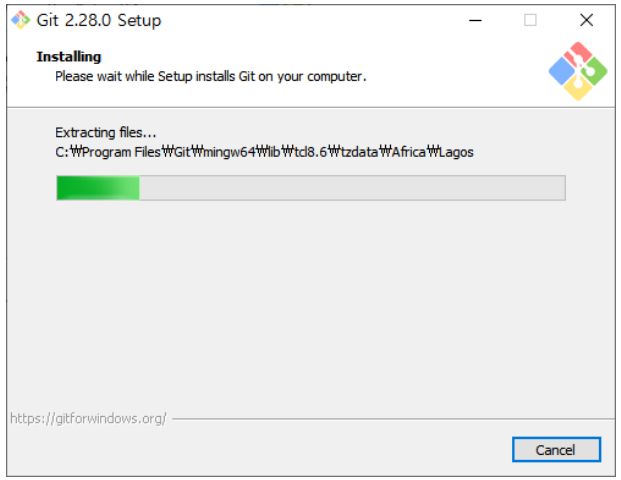


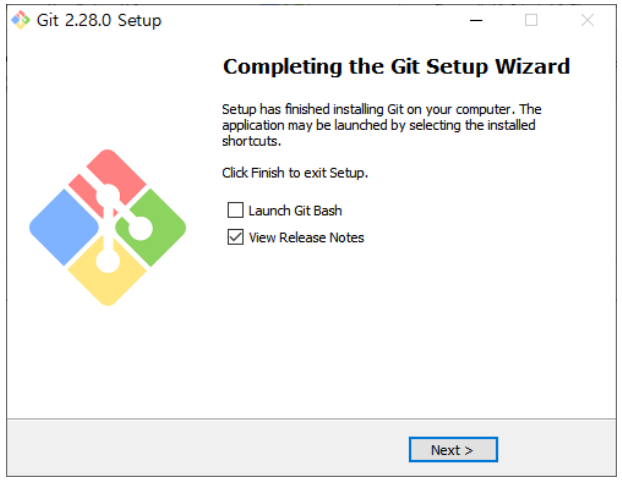




Change your inference model into your local webcam...!!!! (1-4)





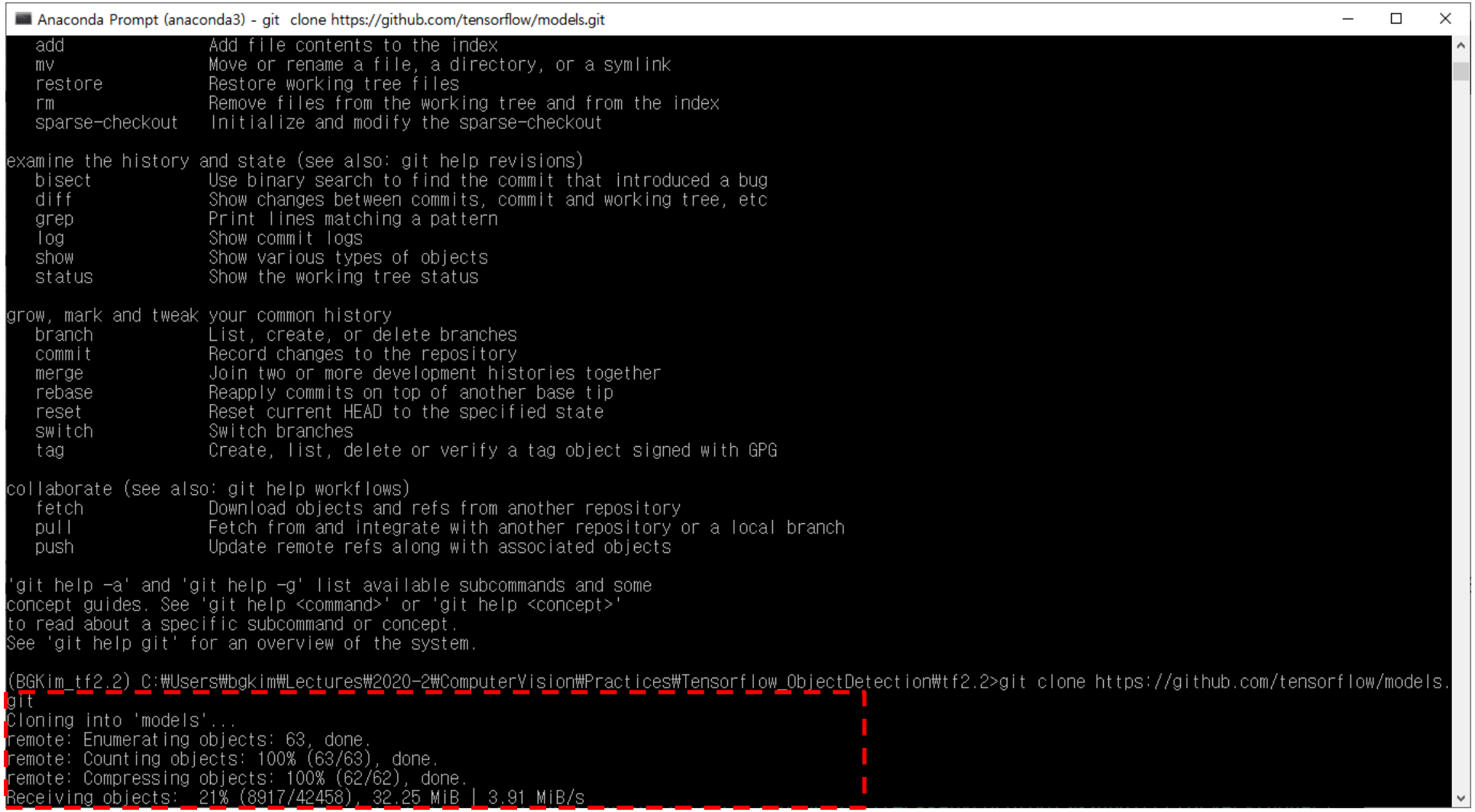


Install completed..!!

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

- 5] Cloning the repository and installing dependencies

```
git clone https://github.com/tensorflow/models.git
```



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

- 5-1] Verify new folder “models” in the current directory.

```
Cloning into 'models'...
remote: Enumerating objects: 63, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 42458 (delta 31), reused 33 (delta 1), pack-reused 42395
Receiving objects: 100% (42458/42458), 549.63 MiB | 3.80 MiB/s, done.
Resolving deltas: 100% (28881/28881), done.
Updating files: 100% (1826/1826), done.

(BGKim_tf2.2) C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AA75-BD71

 C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2 디렉터리

2020-08-26  오후 06:25    <DIR>          .
2020-08-26  오후 06:25    <DIR>          ..
2020-08-26  오후 06:27    <DIR>          models
               0개 파일                   0 바이트
               3개 디렉터리  311,692,173,312 바이트 남음
```

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

- 5-2] Cloning the repository and installing dependencies

```
pip install --user Cython
pip install --user contextlib2
pip install --user pillow
pip install --user lxml
pip install --user jupyter
pip install --user matplotlib
```

- 6] Install COCO API
  - COCO is a large image dataset designed for object detection, segmentation, person keypoints detection, stuff segmentation, and caption generation.

Linux

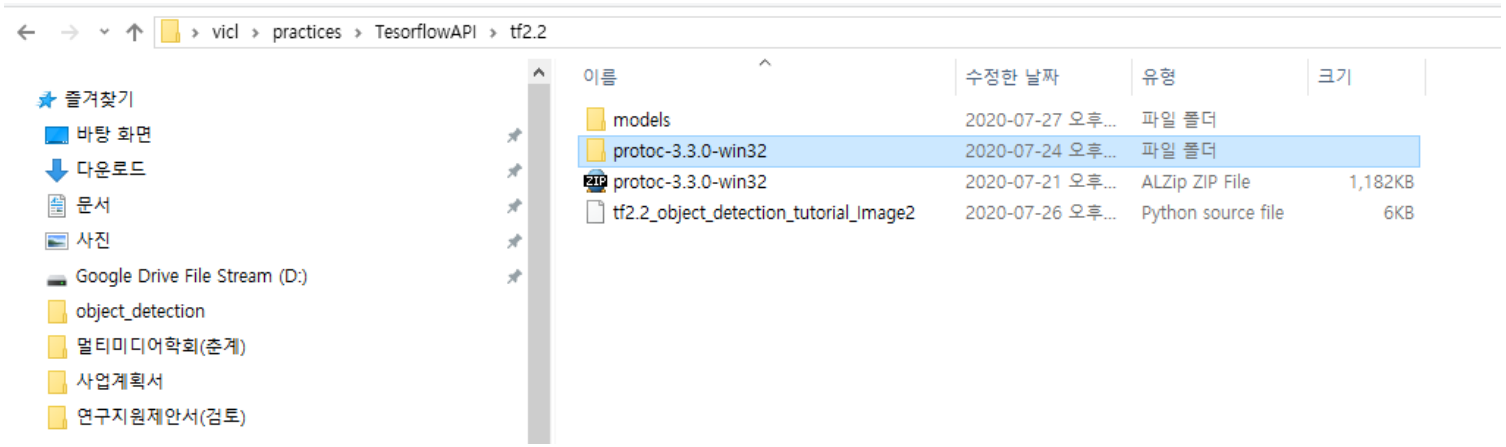
```
git clone https://github.com/cocodataset/cocoapi.git
cd cocoapi/PythonAPI
make cp -r pycocotools <path_to_tensorflow>/models/research/
```

```
pip install
"git+https://github.com/philferriere/cocoapi.git#egg=pycocotools&subdirectory=PythonAPI"
```

Windows(Anaconda3)

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

- 7] Protobuf Installation/Compilation
  - Protobuf can be downloaded from https://github.com/protocolbuffers/protobuf/releases. After downloading you can extract the folder in a directory of your choice **(v3.3.0 is recommended).**



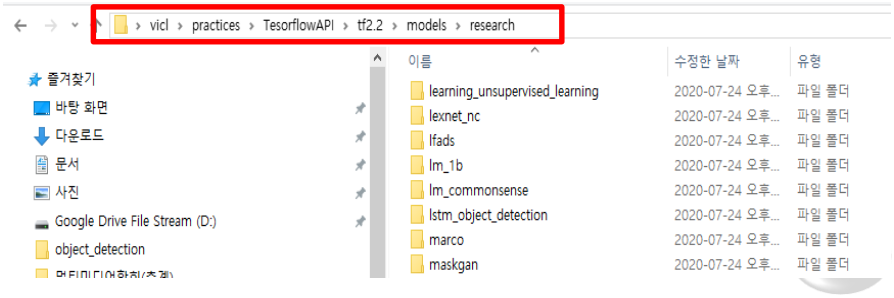
  - Compile Proto buffer and installation

```
./bin/protoc object_detection/protos/*.proto --python_out=.
```

or

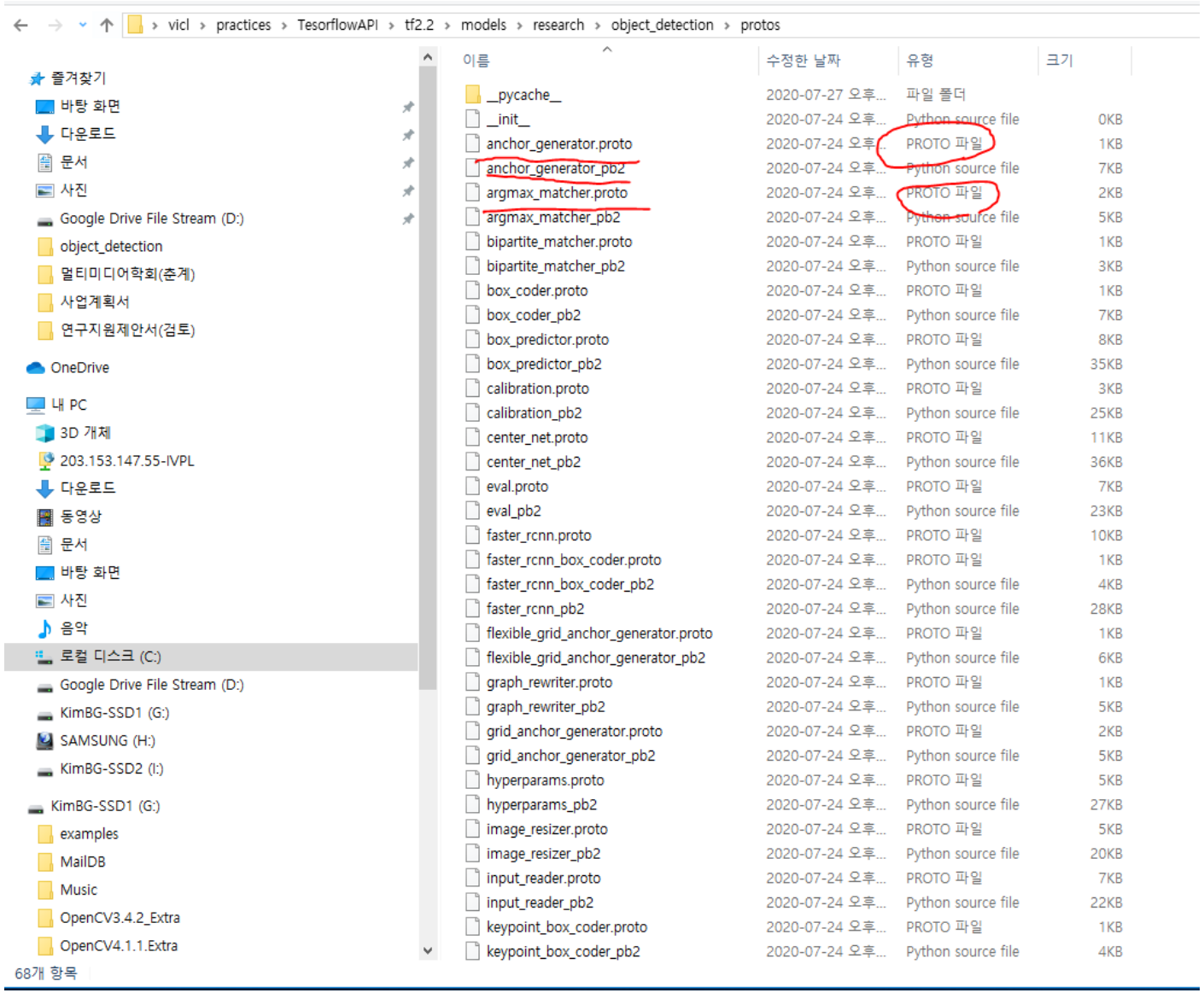
- **Copy "protoc.exe" to "models/research" folder.**
- **Run**

```
Protoc.exe object_detection/protos/*.proto --python_out=.
```



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

After installation properly, you can see some generated pb files in “protos” folder.

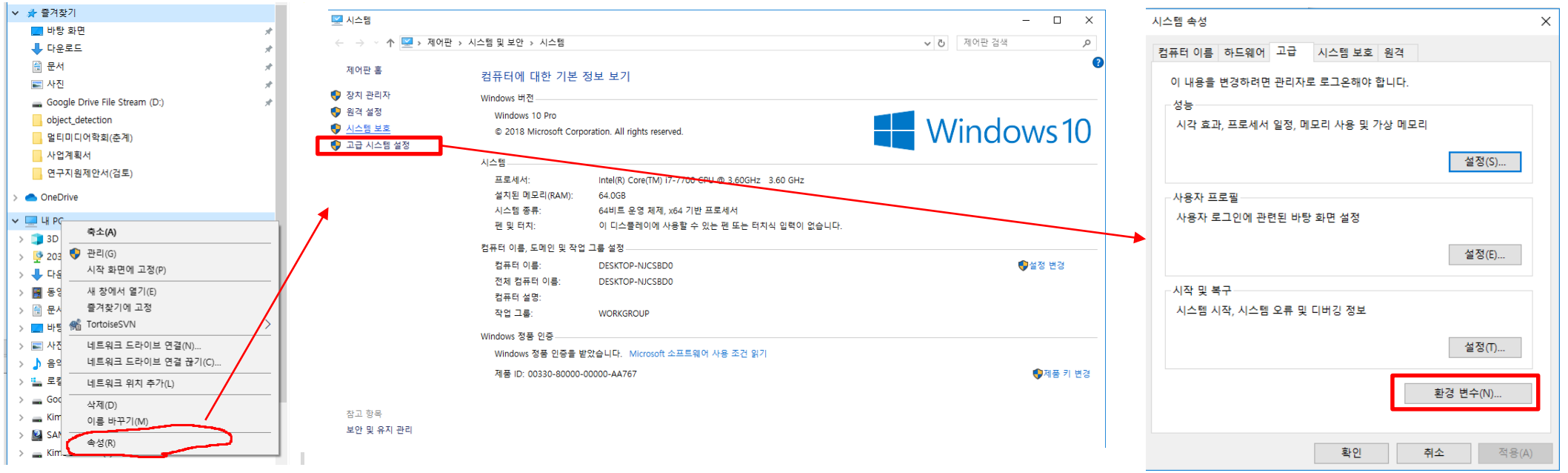


Change your inference model into your local webcam…!!!! (5)

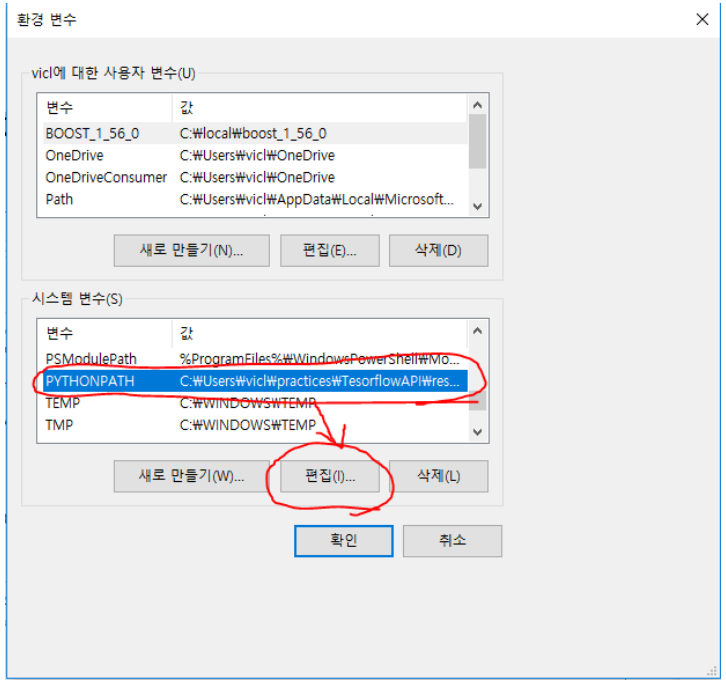
- 8] Add necessary environment variables and finish Tensorflow Object Detection API installation.
  - On **Linux**

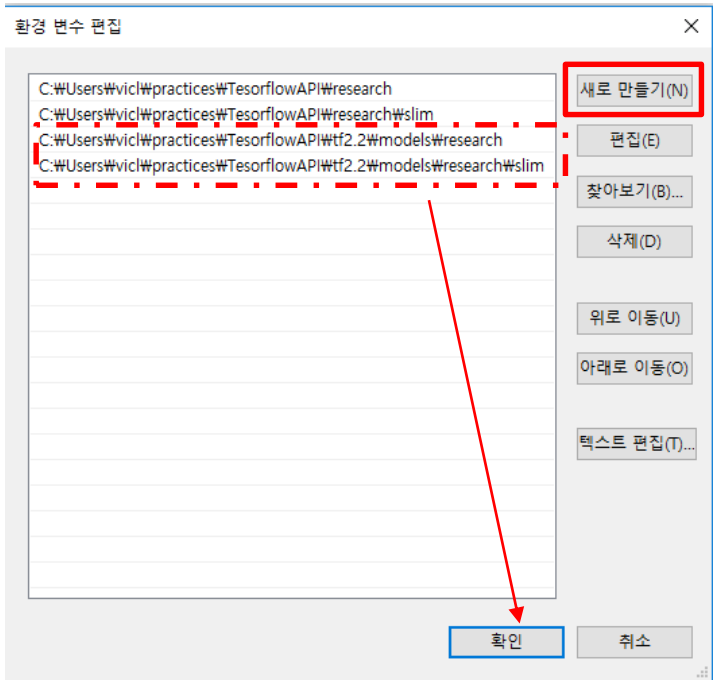
```
export PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research export
PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research/object_detection export
PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research/slim
```

  - On **Windows**
    - you need to at the path of the "research" folder and the "research/slim" to your **PYTHONPATH** environment variable **(if you don't have, make this parameter using "새로만들기".)**



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





C:₩Users₩vicl₩practices₩TesorflowAPI₩tf2.2₩models₩research
C:₩Users₩vicl₩practices₩TesorflowAPI₩tf2.2₩models₩research₩slim

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

- 9] Final Install and Setup of Object detection API
  - 1) Go to "models/research" folder.
  - 2) Run the following step by step:

```
cd models\research
python setup.py build
```

```
python setup.py install
```

Anaconda Prompt (anaconda3)

```
copying object_detection\models\keras_models\inception_resnet_v2_tf2_test.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\mobilenet_v1.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\mobilenet_v1_tf2_test.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\mobilenet_v2.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\mobilenet_v2_tf2_test.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\model_utils.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\resnet_v1.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\resnet_v1_tf2_test.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\test_utils.py -> build\lib\object_detection\models\keras_models
copying object_detection\models\keras_models\__init__.py -> build\lib\object_detection\models\keras_models
creating build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\box_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\box_head_tf1_test.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\class_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\class_head_tf1_test.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_box_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_box_head_tf2_test.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_class_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_class_head_tf2_test.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_mask_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keras_mask_head_tf2_test.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keypoint_head.py -> build\lib\object_detection\predictors\heads
copying object_detection\predictors\heads\keypoint_head_tf1_test.py -> build\lib\object_detection\predictors\heads
ors\heads\mask_head.py -> build\lib\object_detection\predictors\heads
ors\heads\mask_head_tf1_test.py -> build\lib\object_detection\predictors\heads
ors\heads\__init__.py -> build\lib\object_detection\predictors\heads
ion\tpu_exporters\testdata
orters\testdata\__init__.py -> build\lib\object_detection\tpu_exporters\testdata

fo
o\PKG-INFO
ct_detection.egg-info\dependency_links.txt
detection.egg-info\requires.txt
t_detection.egg-info\top_level.txt
tection.egg-info\SOURCES.txt'
tection.egg-info\SOURCES.txt'
tection.egg-info\SOURCES.txt'

tures\2020-2\ComputerVision\Practices\Tensorflow_ObjectDetection\tf2.2\models\research>
```

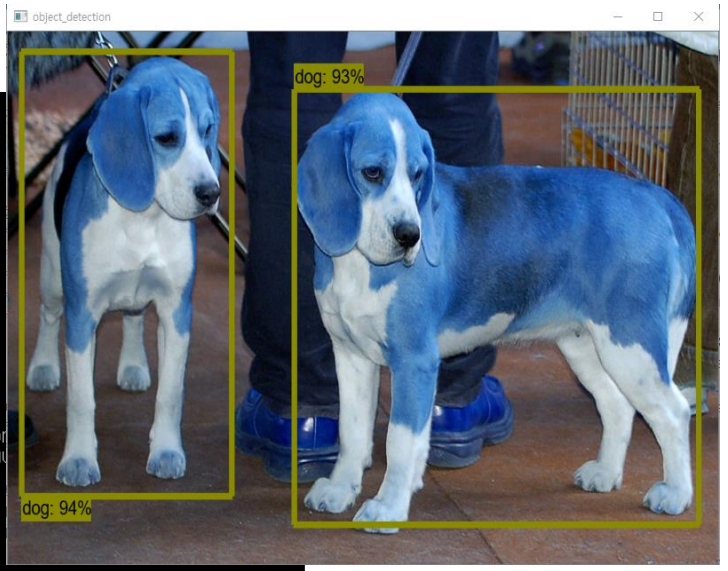
Anaconda Prompt (anaconda3) - python setup.py install

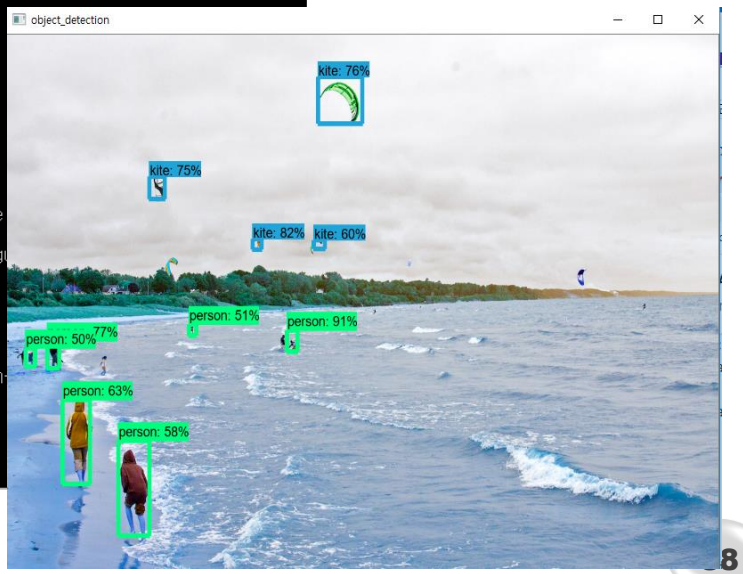
```
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\create_cococameratraps_tfexample_main.py to create_cococameratrap
s_tfexample_main.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\create_cococameratraps_tfexample_tf2_test.py to create_cococamera
traps_tfexample_tf2_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\generate_detection_data.py to generate_detection_data.cpython-37.
pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\generate_detection_data_tf2_test.py to generate_detection_data_tf
2_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\generate_embedding_data.py to generate_embedding_data.cpython-37.
pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\generate_embedding_data_tf2_test.py to generate_embedding_data_tf
2_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\context_rcnn\__init__.py to __init__.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_coco_tf_record.py to create_coco_tf_record.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_coco_tf_record_test.py to create_coco_tf_record_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_kitti_tf_record.py to create_kitti_tf_record.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_kitti_tf_record_test.py to create_kitti_tf_record_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_oid_tf_record.py to create_oid_tf_record.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_pascal_tf_record.py to create_pascal_tf_record.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_pascal_tf_record_test.py to create_pascal_tf_record_test.cpython-37.pyc

byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\create_pet_tf_record.py to create_pet_tf_record.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\oid_hierarchical_labels_expansion.py to oid_hierarchical_labels_expansion.cpyt
hon-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\oid_hierarchical_labels_expansion_test.py to oid_hierarchical_labels_expansion
_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\oid_tfrecord_creation.py to oid_tfrecord_creation.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\oid_tfrecord_creation_test.py to oid_tfrecord_creation_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\seq_example_util.py to seq_example_util.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\seq_example_util_test.py to seq_example_util_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\tf_record_creation_util.py to tf_record_creation_util.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\tf_record_creation_util_test.py to tf_record_creation_util_test.cpython-37.pyc

byte-compiling build\bdist.win-amd64\egg\object_detection\dataset_tools\__init__.py to __init__.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\data_decoders\tf_example_decoder.py to tf_example_decoder.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\data_decoders\tf_example_decoder_test.py to tf_example_decoder_test.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\data_decoders\tf_sequence_example_decoder.py to tf_sequence_example_decoder.cpython-37.pyc
byte-compiling build\bdist.win-amd64\egg\object_detection\data_decoders\tf_sequence_example_decoder_test.py to tf_sequence_example_decoder_test.cpytho
n-37.pyc
```

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

❖ To test if everything is working correctly, run the **object_detection_tutorial.py** from the object_detection folder.

```
(BGKim_tf2.2) C:₩Users₩vicl₩practices₩TesorflowAPI₩tf2.2₩models₩research₩object_detection>python tf2.2_object_detection_tutorial_Image.py
2020-07-28 17:05:57.144940: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
[INFO] TF verion =  2.2.0
[INFO] Downloading model and loading to network : ssd_mobilenet_v1_coco_2017_11_17
2020-07-28 17:06:03.972264: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-07-28 17:06:04.184317: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth: 238.66GiB/s
2020-07-28 17:06:04.198188: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
2020-07-28 17:06:04.938533: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_10.dll
2020-07-28 17:06:06.007306: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cufft64_10.dll
2020-07-28 17:06:06.380861: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library curand64_10.dll
2020-07-28 17:06:07.682430: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusolver64_10.dll
2020-07-28 17:06:08.346073: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusparse64_10.dll
2020-07-28 17:06:11.700117: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-07-28 17:06:11.705709: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703] Adding visible gpu devices: 0
2020-07-28 17:06:11.710814: I tensorflow/core/platform/cpu_feature_guard.cc:143] Your CPU supports instructions that this TensorFlow binary was not cor
2020-07-28 17:06:11.723463: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1e49269b330 initialized for platform Host (this does not gu
). Devices:
2020-07-28 17:06:11.732201: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
2020-07-28 17:06:11.737579: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth: 238.66GiB/s
2020-07-28 17:06:11.748818: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
2020-07-28 17:06:11.754513: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_10.dll
2020-07-28 17:06:11.759799: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cufft64_10.dll
2020-07-28 17:06:11.765536: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library curand64_10.dll
2020-07-28 17:06:11.770614: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusolver64_10.dll
2020-07-28 17:06:11.776567: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusparse64_10.dll
2020-07-28 17:06:11.783529: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-07-28 17:06:11.788763: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703] Adding visible gpu devices: 0
2020-07-28 17:06:12.269435: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-07-28 17:06:12.275384: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]      0
2020-07-28 17:06:12.279525: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] 0:   N
2020-07-28 17:06:12.283103: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247] Created TensorFlow device (/job:localhost/replica:0/task:0/device
 physical GPU (device: 0, name: GeForce GTX 1070 Ti, pci bus id: 0000:01:00.0, compute capability: 6.1)
2020-07-28 17:06:12.297597: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1e4bfd62930 initialized for platform CUDA (this does not gu
). Devices:
2020-07-28 17:06:12.304557: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): GeForce GTX 1070 Ti, Compute Capability 6.1
[<tf.Tensor 'image_tensor:0' shape=(None, None, None, 3) dtype=uint8>]
test_images₩image1.jpg is being processed...!!!
2020-07-28 17:06:19.478835: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-07-28 17:06:20.189026: W tensorflow/stream_executor/gpu/redzone_allocator.cc:314] Internal: Invoking GPU asm compilation is supported on Cuda non-
Relying on driver to perform ptx compilation.
Modify $PATH to customize ptxas location.
This message will be only logged once.
2020-07-28 17:06:20.216362: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_10.dll
```





Change your inference model into your local webcam...!!!! (8-1)

❖ If you have the following error (**No module named 'object_detection'**),

```
(BGKim_tf2.2) C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection>python tf2.2_o
bject_detection_tutorial_Image.py
2020-08-26 18:50:46.049579: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
Traceback (most recent call last):
  File "tf2.2_object_detection_tutorial_Image.py", line 16, in <module>
    from object_detection.utils import ops as utils_ops
ModuleNotFoundError: No module named 'object_detection'

(BGKim_tf2.2) C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection>
```

than you can reboot your system. This error usually occurs if you fail to set the PYTHONPATH in your system. Than try again.

❖ IF you have another error, you have to install tf-slim package.

```
(BGKim_tf2.2) C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection>python tf2.2_o
bject_detection_tutorial_Image.py
2020-08-26 19:04:02.123745: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
Traceback (most recent call last):
  File "tf2.2_object_detection_tutorial_Image.py", line 16, in <module>
    from object_detection.utils import ops as utils_ops
  File "C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection₩utils₩ops.py", line
28, in <module>
    import tf_slim as slim
ModuleNotFoundError: No module named 'tf_slim'
```

```
>>pip install --upgrade tf-slim
```

❖ One more point…!!! In “object_detection_tutorial.py ”, “PATH_TO_LABELS” setting should be pointed correctly. Otherwise, you can see the following error:

```
56  PATH_TO_LABELS = os.path.join('C:/Users/bgkim/Lectures/2020-2/ComputerVision/Practices/Tensorflow_ObjectDetection/tf2.2/models/research/object_detection/data','mscoco_label_map.pbtxt')
57  category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS)
```

```
(BGKim_tf2.2) C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection>python tf2.2_o
bject_detection_tutorial_Image.py
2020-08-26 19:07:22.778578: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
[INFO] TF verion =  2.2.0
Traceback (most recent call last):
  File "tf2.2_object_detection_tutorial_Image.py", line 57, in <module>
    category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS)
  File "C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection₩utils₩label_map_util
.py", line 315, in create_category_index_from_labelmap
    categories = create_categories_from_labelmap(label_map_path, use_display_name)
  File "C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection₩utils₩label_map_util
.py", line 295, in create_categories_from_labelmap
    label_map = load_labelmap(label_map_path)
  File "C:₩Users₩bgkim₩Lectures₩2020-2₩ComputerVision₩Practices₩Tensorflow_ObjectDetection₩tf2.2₩models₩research₩object_detection₩utils₩label_map_util
.py", line 156, in load_labelmap
    label_map_string = fid.read()
  File "C:₩Users₩bgkim₩anaconda3₩envs₩BGKim_tf2.2₩lib₩site-packages₩tensorflow₩python₩lib₩io₩file_io.py", line 116, in read
    self._preread_check()
  File "C:₩Users₩bgkim₩anaconda3₩envs₩BGKim_tf2.2₩lib₩site-packages₩tensorflow₩python₩lib₩io₩file_io.py", line 79, in _preread_check
    self._name, 1024 * 512)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc1 in position 149: invalid start byte
```

Change your inference model into your local webcam...!!!! (9)

- Source insight: object_detection_tutorial.py

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
from IPython.display import display
import pathlib

from object_detection.utils import ops as utils_ops
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util

import cv2
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW) # or cap =
cv2.VideoCapture("<video-path>")

# patch tf1 into `utils.ops`
utils_ops.tf = tf.compat.v1

# Patch the location of gfile
tf.gfile = tf.io.gfile

print("[INFO] TF verion = ",tf.__version__)
```

```
#--- Model Preparation ----#
def load_model(model_name):
    #--- 1) tf 1.x model base url
    #base_url = 'http://download.tensorflow.org/models/object_detection/'

    #--- 2) tf 2.2 model base url
    base_url = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/'

    model_file = model_name + '.tar.gz'
    model_dir = tf.keras.utils.get_file(
        fname=model_name,
        origin=base_url + model_file,
        untar=True)

    model_dir = pathlib.Path(model_dir)/"saved_model"

    model = tf.saved_model.load(str(model_dir))

    return model

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS =
os.path.join('C:/Users/vicl/practices/TesorflowAPI/tf2.2/models/research/object_detection/dat
a','mscoco_label_map.pbtxt')
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS)

# If you want to test the code with your images, just add path to the images to the
TEST_IMAGE_PATHS.
#PATH_TO_TEST_IMAGES_DIR = pathlib.Path('models/research/object_detection/test_images')
PATH_TO_TEST_IMAGES_DIR = pathlib.Path('test_images')
TEST_IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
TEST_IMAGE_PATHS
```

Change your inference model into your local webcam...!!!! (10)

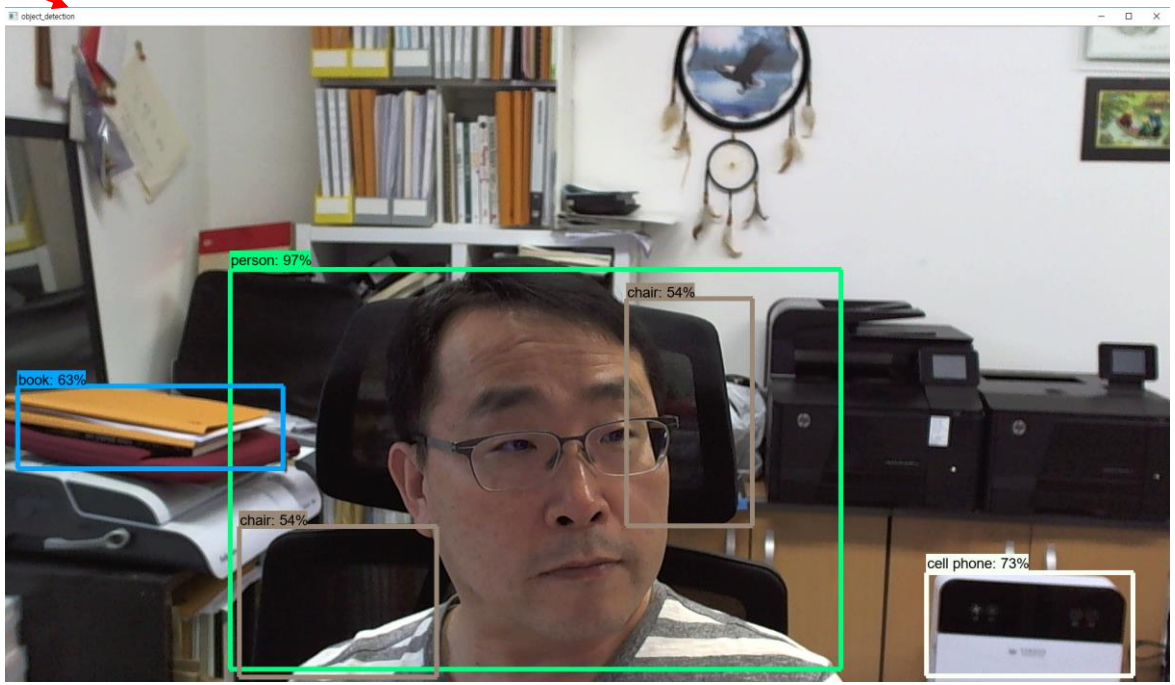
```
#--- Detection --#
model_name = 'ssd_mobilenet_v1_coco_2017_11_17'
print('[INFO] Downloading model and loading to network : '+ model_name)
detection_model = load_model(model_name)
print(detection_model.signatures['serving_default'].inputs)
detection_model.signatures['serving_default'].output_dtypes
detection_model.signatures['serving_default'].output_shapes

def run_inference_for_single_image(model, image):
  image = np.asarray(image)
  # The input needs to be a tensor, convert it using `tf.convert_to_tensor`.
  input_tensor = tf.convert_to_tensor(image)
  # The model expects a batch of images, so add an axis with `tf.newaxis`.
  input_tensor = input_tensor[tf.newaxis,...]

  # Run inference
  model_fn = model.signatures['serving_default']
  output_dict = model_fn(input_tensor)

  # All outputs are batches tensors.
  # Convert to numpy arrays, and take index [0] to remove the batch dimension.
  # We're only interested in the first num_detections.
  num_detections = int(output_dict.pop('num_detections'))
  output_dict = {key:value[0, :num_detections].numpy()
                 for key,value in output_dict.items()}
  output_dict['num_detections'] = num_detections

  # detection_classes should be ints.
  output_dict['detection_classes'] = output_dict['detection_classes'].astype(np.int64)

  # Handle models with masks:
  if 'detection_masks' in output_dict:
    # Reframe the the bbox mask to the image size.
    detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
              output_dict['detection_masks'], output_dict['detection_boxes'],
               image.shape[0], image.shape[1])
    detection_masks_reframed = tf.cast(detection_masks_reframed > 0.5,
                                       tf.uint8)
    output_dict['detection_masks_reframed'] = detection_masks_reframed.numpy()

  return output_dict
```

```
def show_inference(model, image_path):
  # the array based representation of the image will be used later in order to prepare the
  # result image with boxes and labels on it.
  image_np = np.array(Image.open(image_path))
  # Actual detection.
  output_dict = run_inference_for_single_image(model, image_np)
  # Visualization of the results of a detection.
  vis_util.visualize_boxes_and_labels_on_image_array(
      image_np,
      output_dict['detection_boxes'],
      output_dict['detection_classes'],
      output_dict['detection_scores'],
      category_index,
      instance_masks=output_dict.get('detection_masks_reframed', None),
      use_normalized_coordinates=True,
      line_thickness=8)

  #display(Image.fromarray(image_np))    #<-- this got an error....!!!
  #--- Using openCV, we want to show the results.... #
  cv2.imshow('object_detection', cv2.resize(image_np, (800, 600)))

def run_inference(model, cap):
    while cap.isOpened():
        ret, image_np = cap.read()
        # Actual detection.
        output_dict = run_inference_for_single_image(model, image_np)
        # Visualization of the results of a detection.
        vis_util.visualize_boxes_and_labels_on_image_array(
            image_np,
            output_dict['detection_boxes'],
            output_dict['detection_classes'],
            output_dict['detection_scores'],
            category_index,
            instance_masks=output_dict.get('detection_masks_reframed', None),
            use_normalized_coordinates=True,
            line_thickness=8)
        cv2.imshow('object_detection', cv2.resize(image_np, (800, 600)))

        if cv2.waitKey(25) & 0xFF == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
```

Change your inference model into your local webcam...!!!! (11)

```
#-- Run it on each test image and show the results: --#
for image_path in TEST_IMAGE_PATHS:
    print(str(image_path)+' is being processed...!!!')
    show_inference(detection_model, image_path)
    cv2.waitKey(0)
```

```
#-- Web camp-based detection run...!!! ---#
#run_inference(detection_model, cap)
```



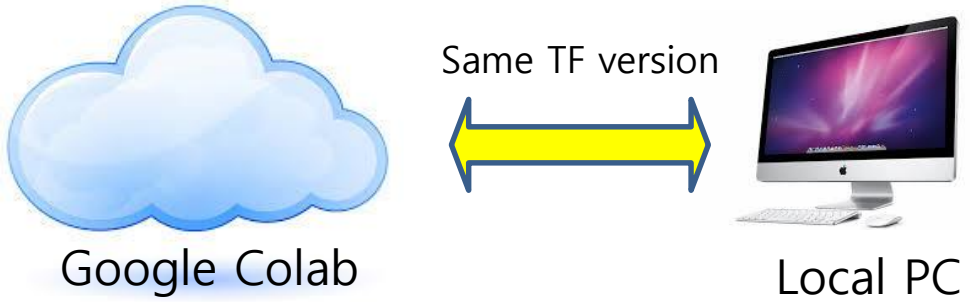
Applying your trained model into your local webcam...!!!! (1)

❖ Check on your local Tensorflow version and one in Google Colab:

- You should verify your local Tensorflow version as the same one of Colab's

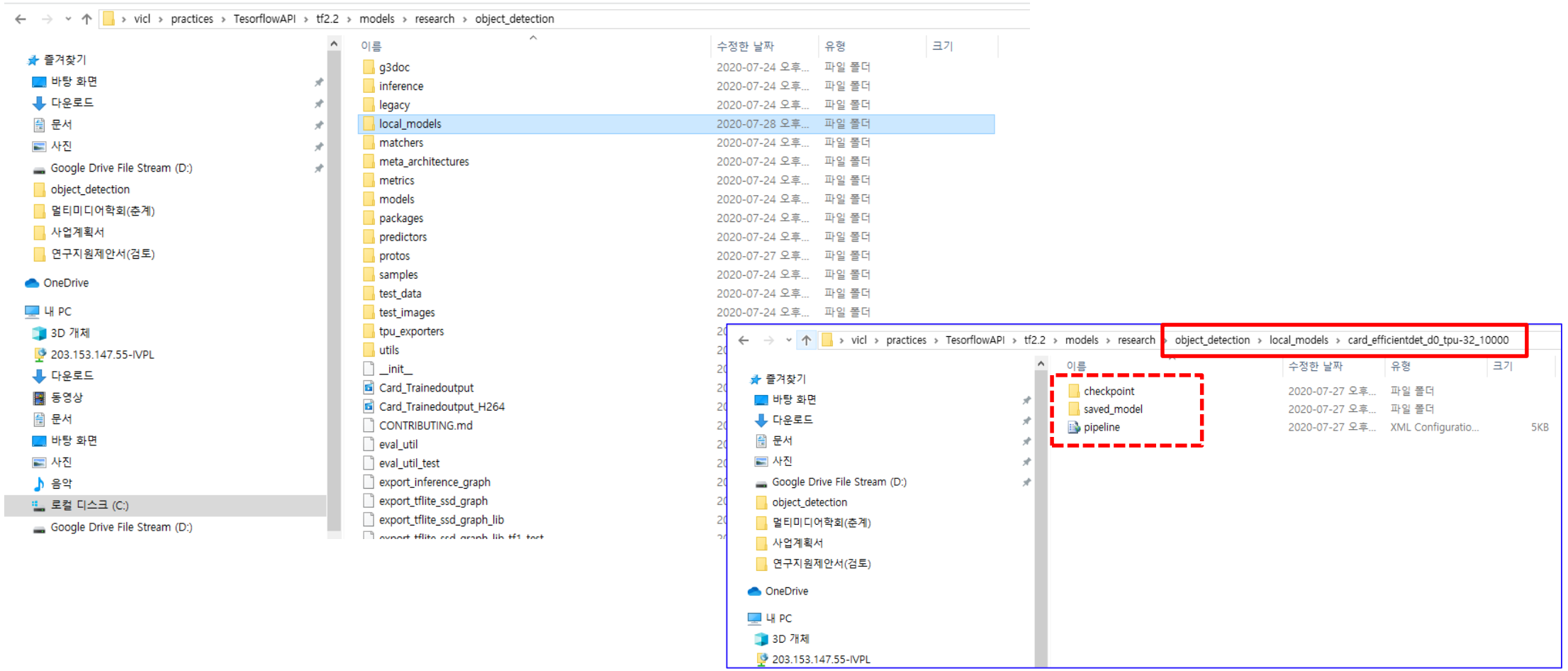
```
[ ]  import itertools
     import os

     import matplotlib.pylab as plt
     import numpy as np

     import tensorflow as tf
     import tensorflow_hub as hub

     print("TF version:", tf.__version__)
     print("Hub version:", hub.__version__)
     print("GPU is", "available" if tf.config.list_physical_devices('GPU') else "NOT AVAILABLE")
```

```
TF version: 2.2.0
Hub version: 0.8.0
GPU is available
```



Applying your trained model into your local webcam...!!!! (2)

❖ Make “local_models” and copy your inference files (**trained tar.gz** file) into the “local_models”. Decompress xxx.tar.gz file.

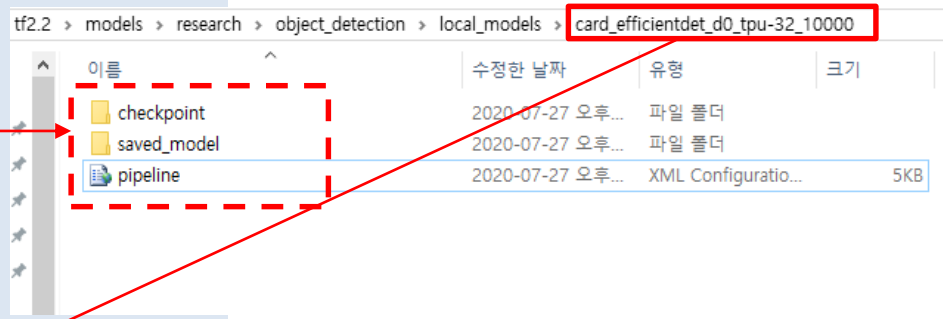


Applying your trained model into your local webcam...!!!! (3)

❖ Then change your source file to load your trained model data as the following:

In my case,

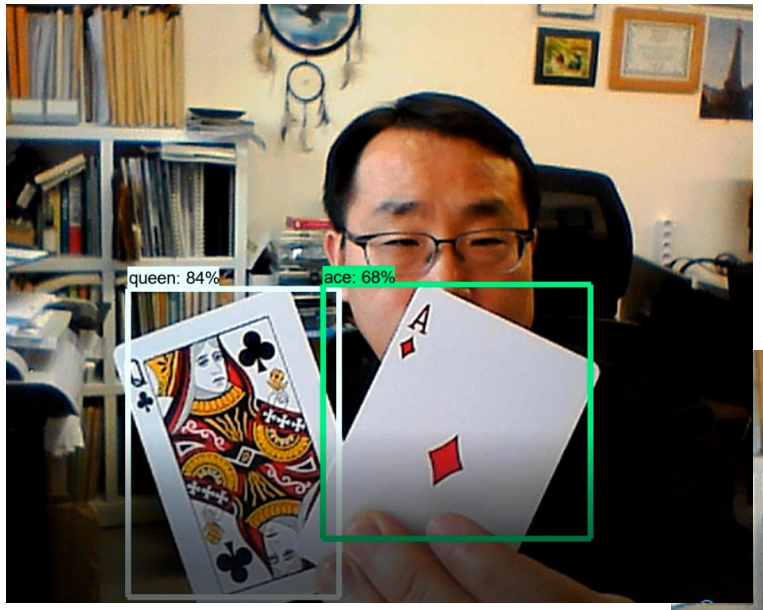
```
def load_model(model_name):
    ~~~~~~

  #--- 2) tf 2.2 model base url
  #base_url = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/' #not working....!!!
  #model_dir = 'local_models/card_efficientdet_d0_tpu-32'  # efficientdet_d1
  model_dir = 'local_models/'+model_name  # efficientdet_d1

  model_dir = pathlib.Path(model_dir)/"saved_model"
  print('[INFO] Loading the modle from '+ str(model_dir))
  model = tf.saved_model.load(str(model_dir))

  return model

# List of the strings that is used to add correct label for each box.
#-- my trainging data label loading
PATH_TO_LABELS =  'data/cardsNew1_label_map.pbtxt' #'data/mscoco_label_map.pbtxt'
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS)

# If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
#PATH_TO_TEST_IMAGES_DIR = pathlib.Path('models/research/object_detection/test_images')
PATH_TO_TEST_IMAGES_DIR = pathlib.Path('test_images')
TEST_IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
TEST_IMAGE_PATHS

#--- Detection --#
model_name =  'card_efficientdet_d0_tpu-32_10000 '  #'centernet_resnet50_v1_fpn_512x512_kpts_coco17_tpu-8'
#'efficientdet_d0_coco17_tpu-32' #'ssd_mobilenet_v1_coco_2017_11_17'
print('[INFO] Downloading model and loading to network : '+ model_name)
detection_model = load_model(model_name)
```

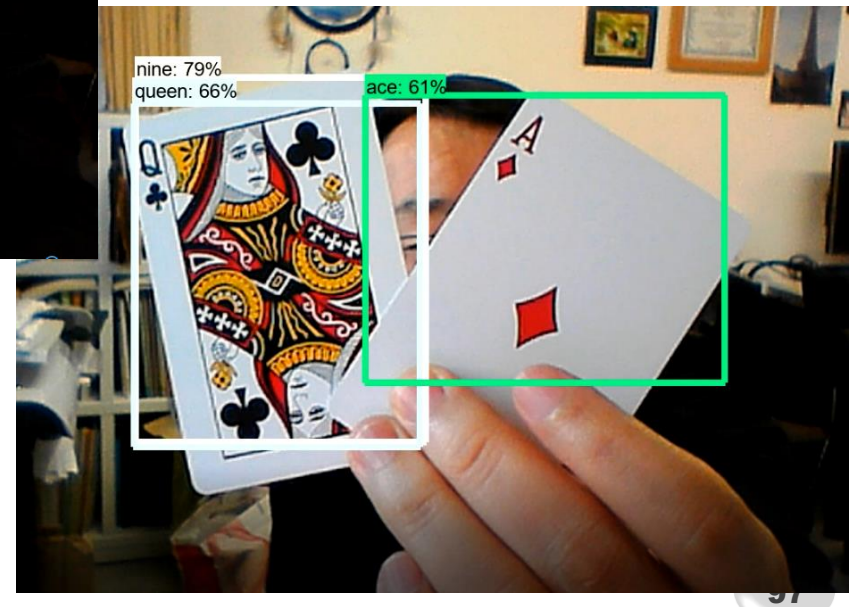


Applying your trained model into your local webcam...!!!! (4)

- Save the changed code as "tf2.2_object_detection_tutorial_WebcamLive_PokerCard.py"
- Then run your inference python script as:

```
>>python tf2.2_object_detection_tutorial_WebcamLive_PokerCard.py
```





Applying your trained model into your local webcam...!!!! (5)

Thank you for your attention.!!!
QnA

http://ivpl.sookmyung.ac.kr