非接觸式體溫量測模組
Non-contact body temperature measurement module

國立台灣科技大學電子系

Department of Electronic and Computer Engineering, NTUST

單元二：熱影像人臉偵測模型訓練實驗

Unit 2: Thermal image face detection model training experiment

國立台灣科技大學電子系

Department of Electronic and Computer Engineering, NTUST

Unit 2: Thermal image face detection model training experiment

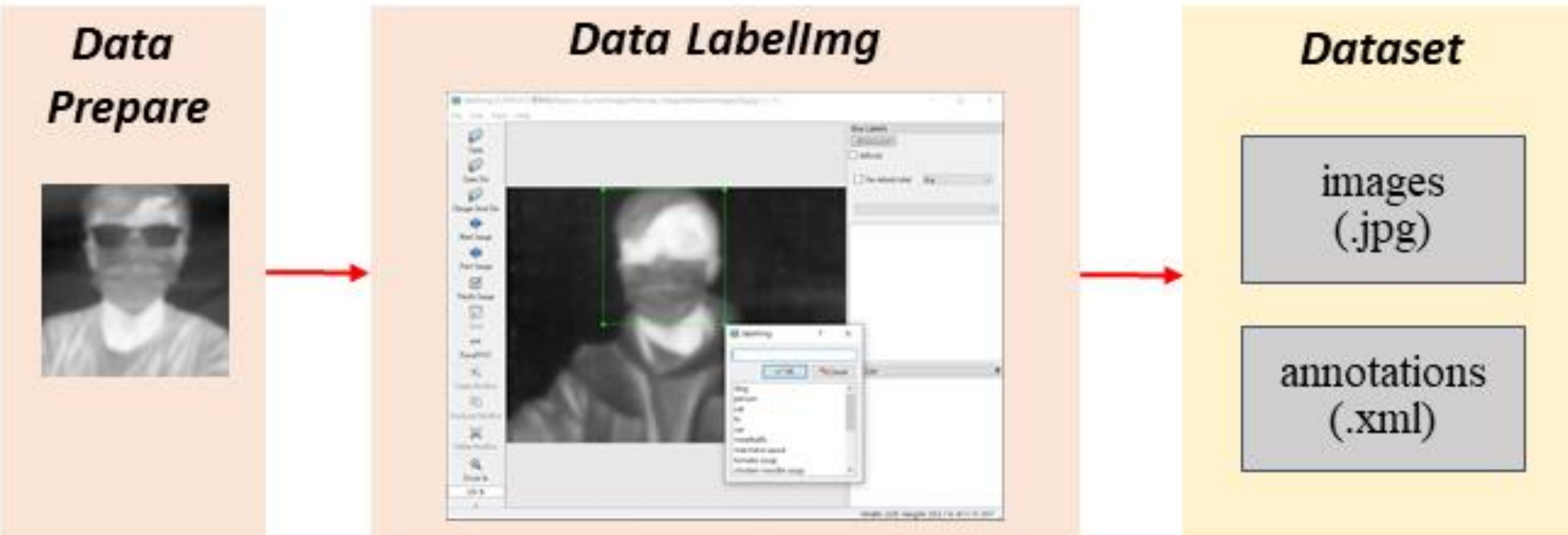
- Model training & data preparation
  - Training procedure
  - Data preparation
- Model architecture & introduction
- Lab2

Model training & data preparation

Training procedure

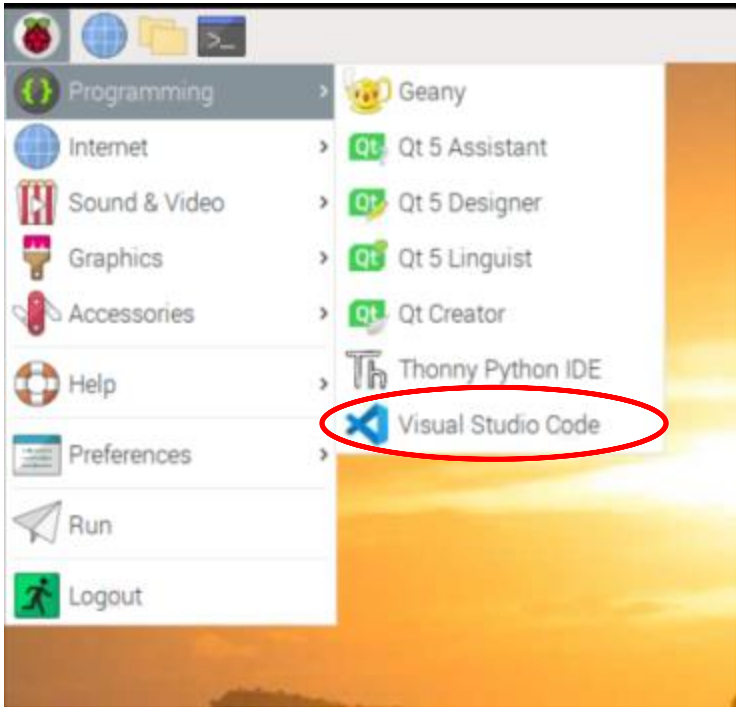
Data preparation

- We need to prepare dataset before training.



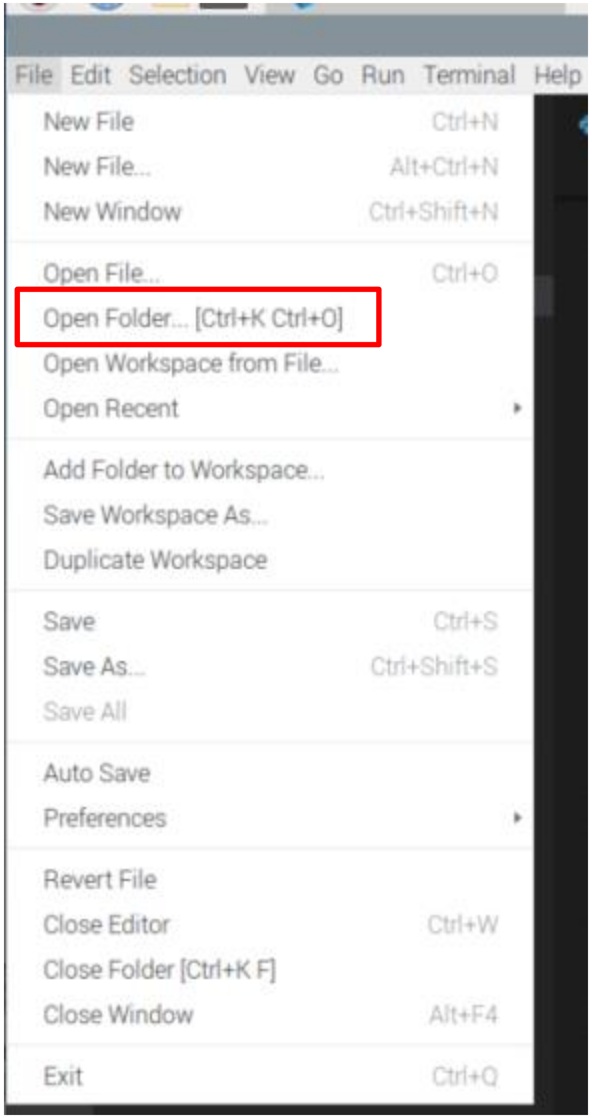
Data preparation

- Press Visual Studio Code.



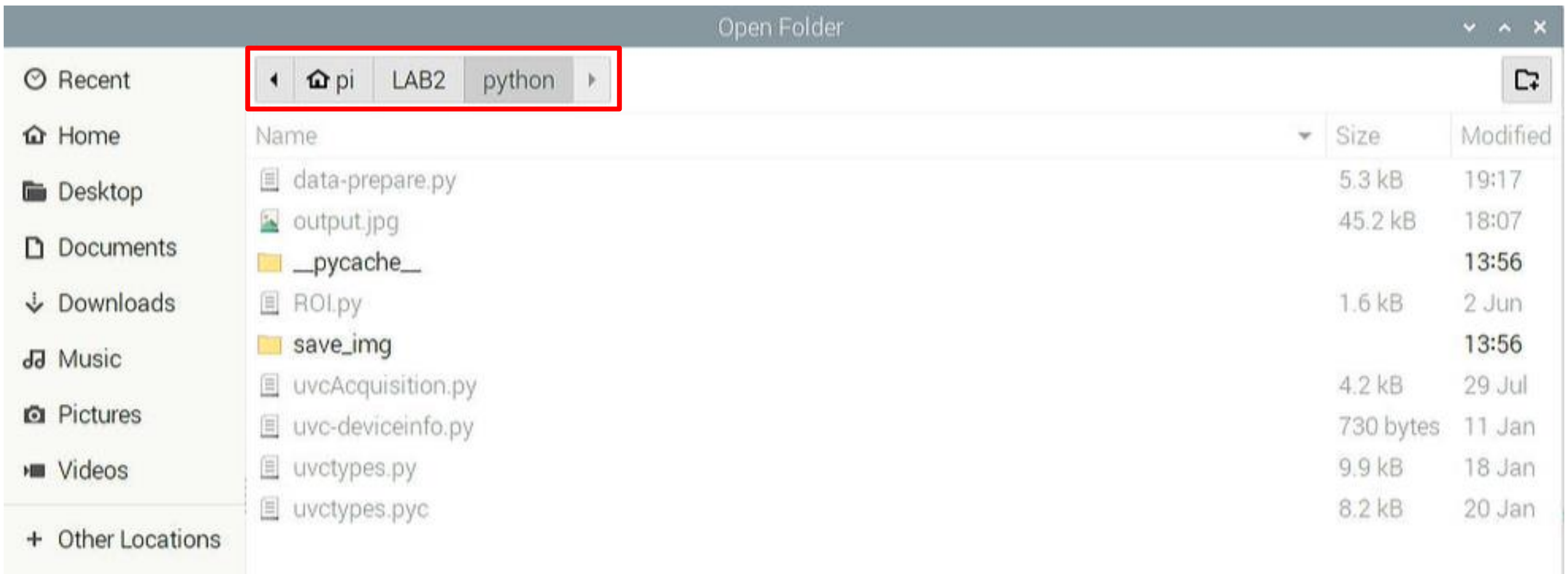
Data preparation

- Press “open folder”.



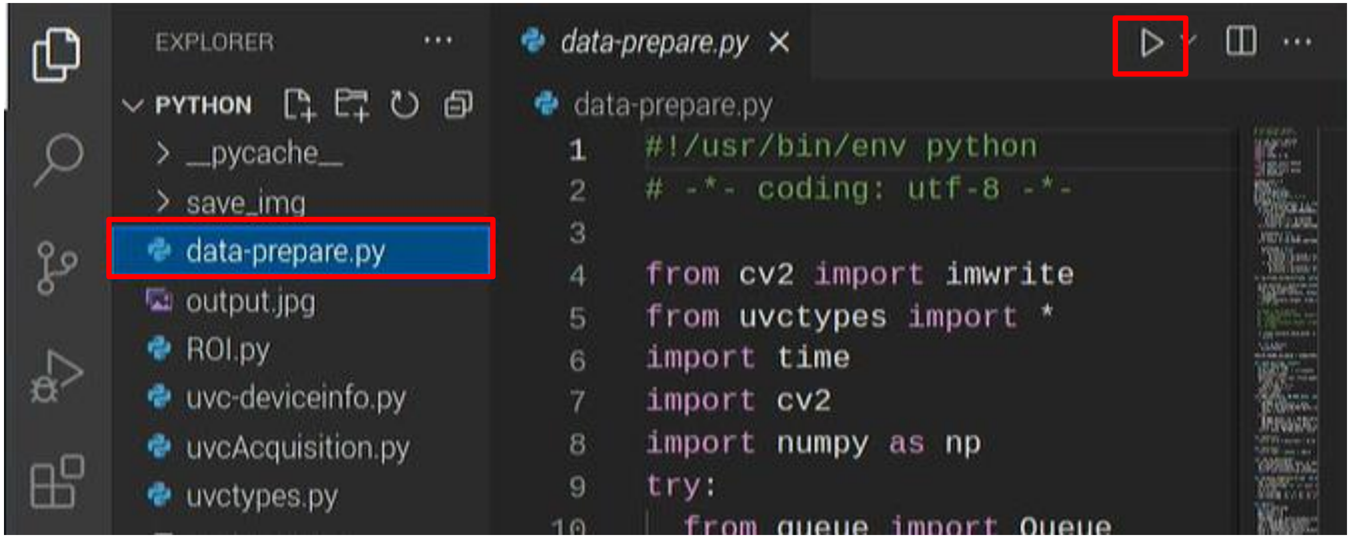
Data preparation

- Select this path.



Data preparation

- Run “data-prepare.py”.



Data preparation

- Rotate the face like this gif.
- Press “b” to save the picture of each frame.

Data preparation

- Use keyboard to control the function.

```
img = raw_to_8bit(data)
if save_flag:
    pic_count = save_pic(img, pic_count)
cv2.namedWindow('Lepton Radiometry')
cv2.setMouseCallback('Lepton Radiometry',OnMouseAction)
cv2.rectangle(img, (x1, y1), (x2, y2), (0,255,0), 1)
maxLoc = (x1+maxLoc[0], y1+maxLoc[1])
# display_temperature(img, minVal, minLoc, (255, 0, 0))
display_temperature(img, maxVal, maxLoc, (0, 0, 255))
cv2.imshow('Lepton Radiometry', img)
key = cv2.waitKey(1)

if key == ord('b'):
    save_flag = True
```

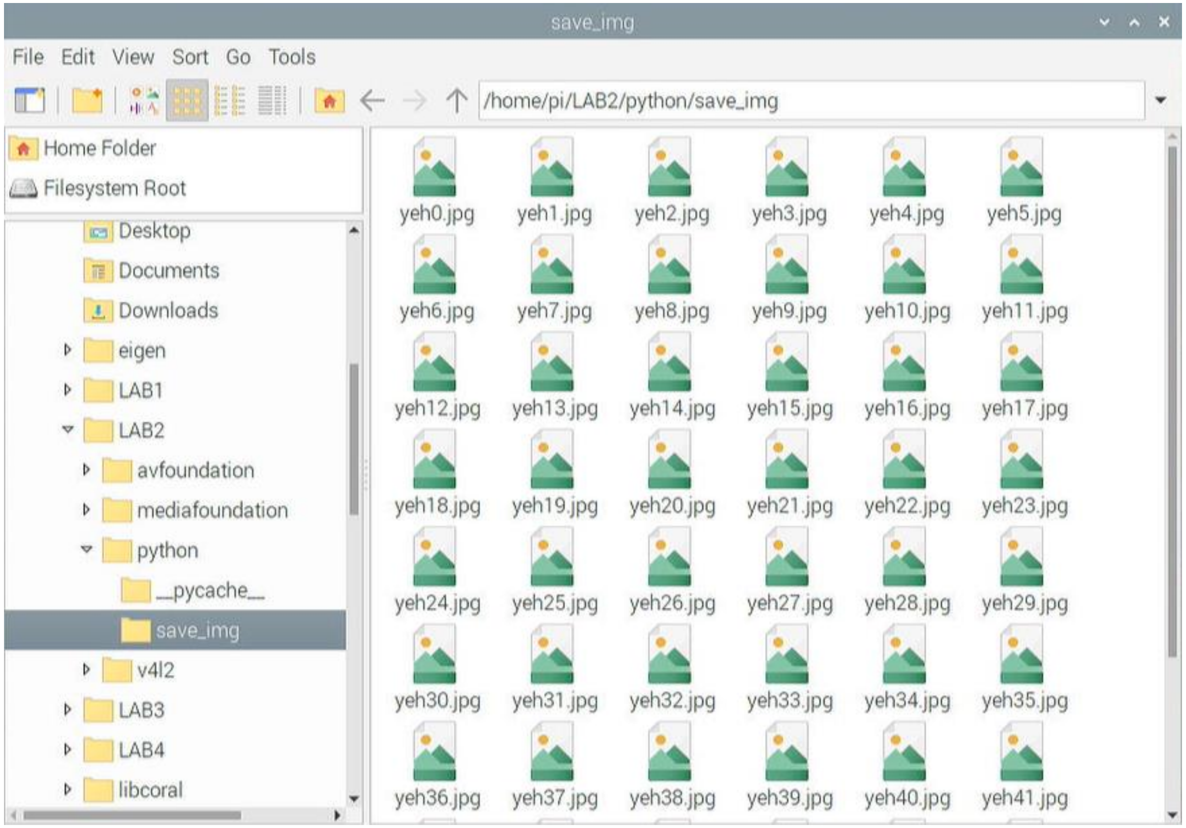
Data preparation

- Save 50 images.

```
def save_pic(img, count):
    global save_flag
    file_name = 'yeh' + str(count)
    print(file_name)
    imwrite(file_name+'.jpg', img)
    if count == 49:
        save_flag = False
    count = count + 1
    return count
```

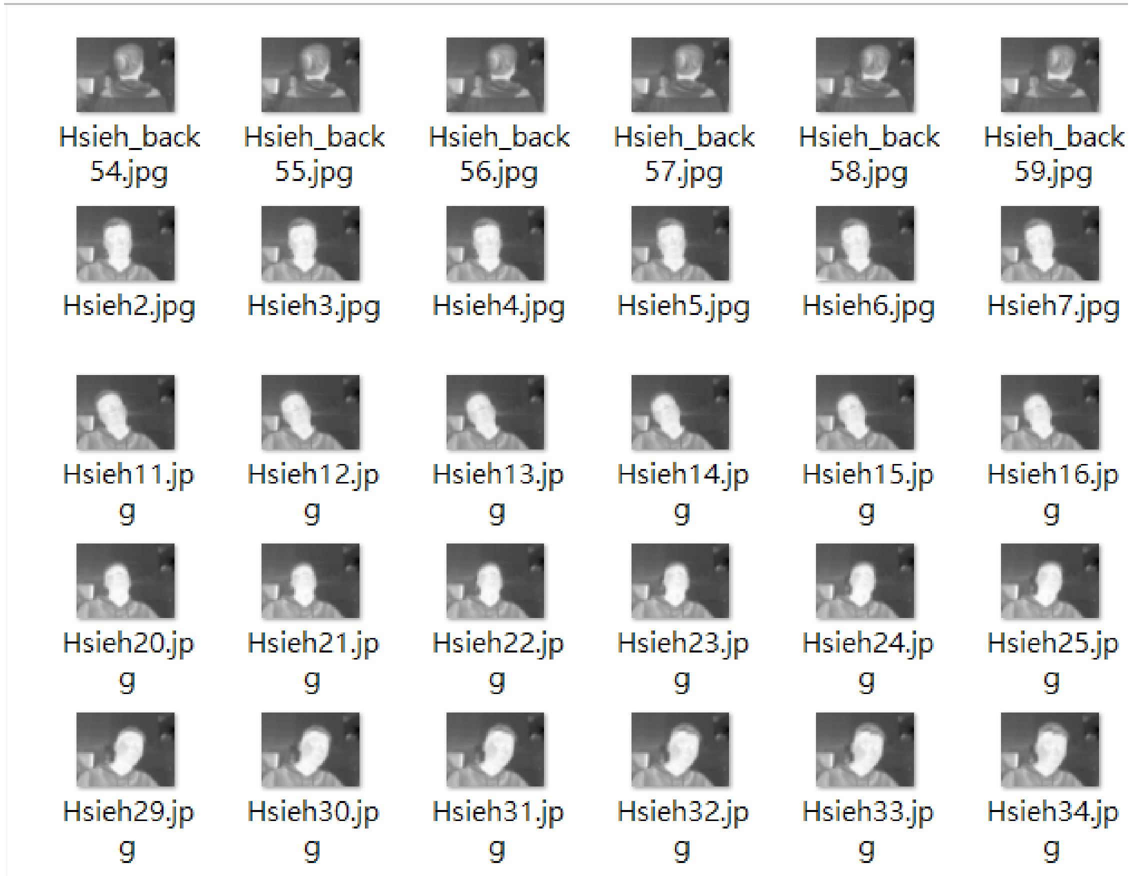
Data preparation

- Find the images in your folder.



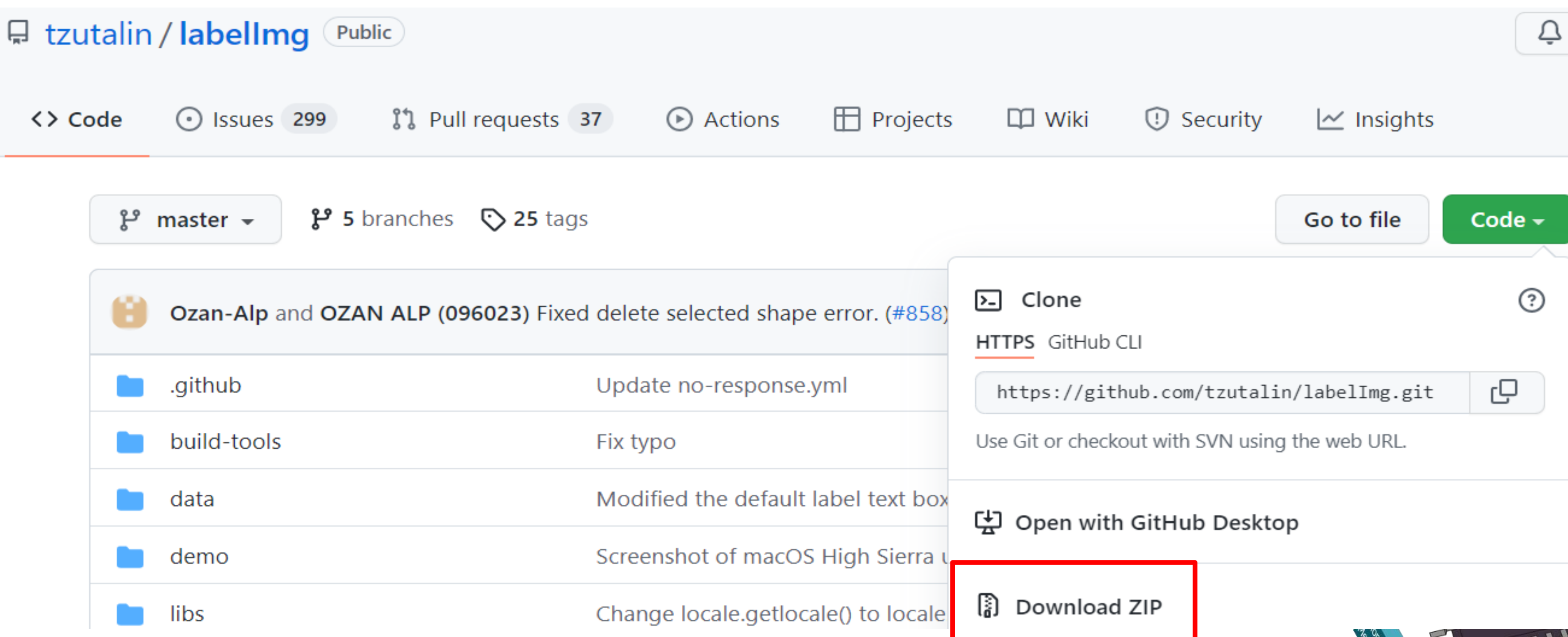
Data preparation

- Move your images to your computer.



Data preparation

- Download the “labelImg” from github on your computer.



https://github.com/tzutalin/labelImg

Data preparation

- Environment requirement
  - Python
  - PyQt5
  - lxml

Windows

Install Python, PyQt5 and install lxml.

Open cmd and go to the labelImg directory

```
pyrcc4 -o libs/resources.py resources.qrc
For pyqt5, pyrcc5 -o libs/resources.py resources.qrc

python labelImg.py
python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

Python installation

- https://www.python.org/downloads/windows/
- Select the stable releases.

Stable Releases

- Python 3.9.10 - Jan. 14, 2022

  Note that Python 3.9.10 *cannot* be used on Windows 7 or earlier.

  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows help file
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)

PyQt5 installation

- Open CMD and enter the following.

$ pip install PyQt5

```
C:\WINDOWS\system32>pip install PyQt5
Collecting PyQt5
  Downloading PyQt5-5.15.6-cp36-abi3-win_amd64.whl (6.7 MB)
     |████████████████████████████████| 6.7 MB 595 kB/s
Collecting PyQt5-sip<13,>=12.8
  Downloading PyQt5_sip-12.9.1-cp36-cp36m-win_amd64.whl (83 kB)
     |████████████████████████████████| 83 kB 416 kB/s
Collecting PyQt5-Qt5>=5.15.2
  Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
     |████████████████████████████████| 50.1 MB 6.8 MB/s
Installing collected packages: PyQt5-sip, PyQt5-Qt5, PyQt5
Successfully installed PyQt5-5.15.6 PyQt5-Qt5-5.15.2 PyQt5-sip-12.9.1
```

https://www.riverbankcomputing.com/software/pyqt/download

Lxml installation

$ pip install lxml

» Installation

If your system does not provide binary packages or you want to install a newer version, the best way is to get the **pip** package management tool (or use a **virtualenv**) and run the following:

```
pip install lxml
```

If you are not using pip in a virtualenv and want to install lxml globally instead, you have to run the above command as admin, e.g. on Linux:

```
sudo pip install lxml
```

```
C:\WINDOWS\system32>pip install lxml
Requirement already satisfied: lxml in c:\programdata\anaconda3\lib\site-packages (4.2.1)
```

https://lxml.de/installation.html

Requirement

- Cd to labelImg-master.

$ pyrcc5 –o libs/resources.py resources.qrc

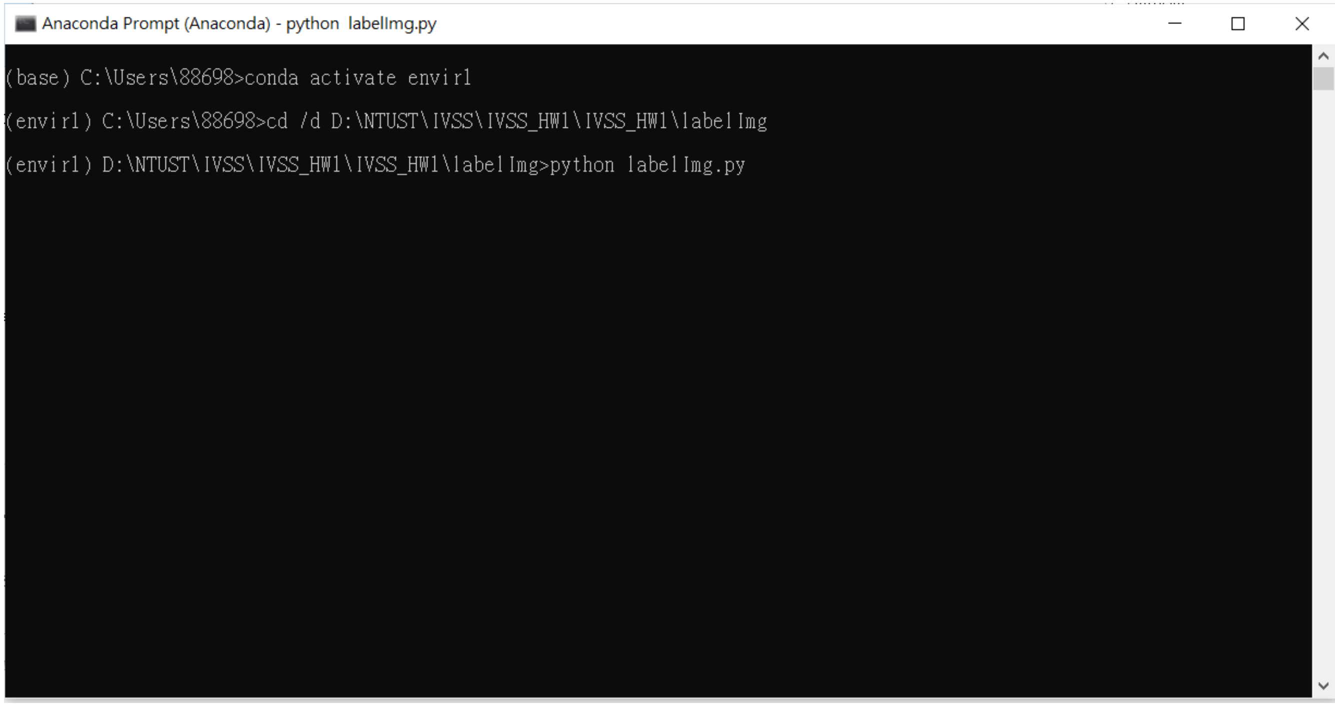
```
pyrcc4 -o libs/resources.py resources.qrc
For pyqt5, pyrcc5 -o libs/resources.py resources.qrc

python labelImg.py
python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

```
E:\labelImg-master>pyrcc5 -o libs/resources.py resources.qrc
```

Data preparation

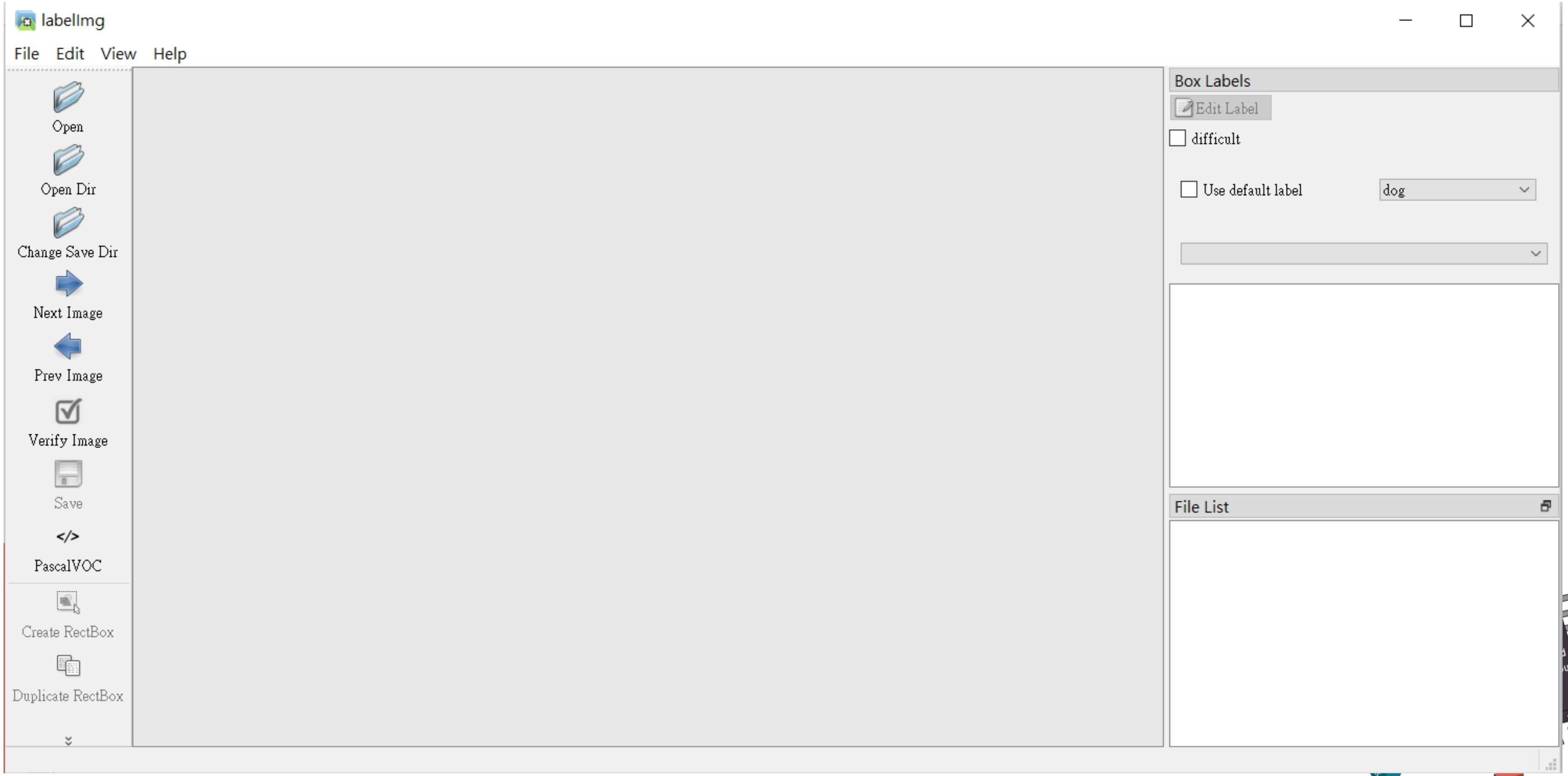
- Open labelImg.

$ python labelImg.py



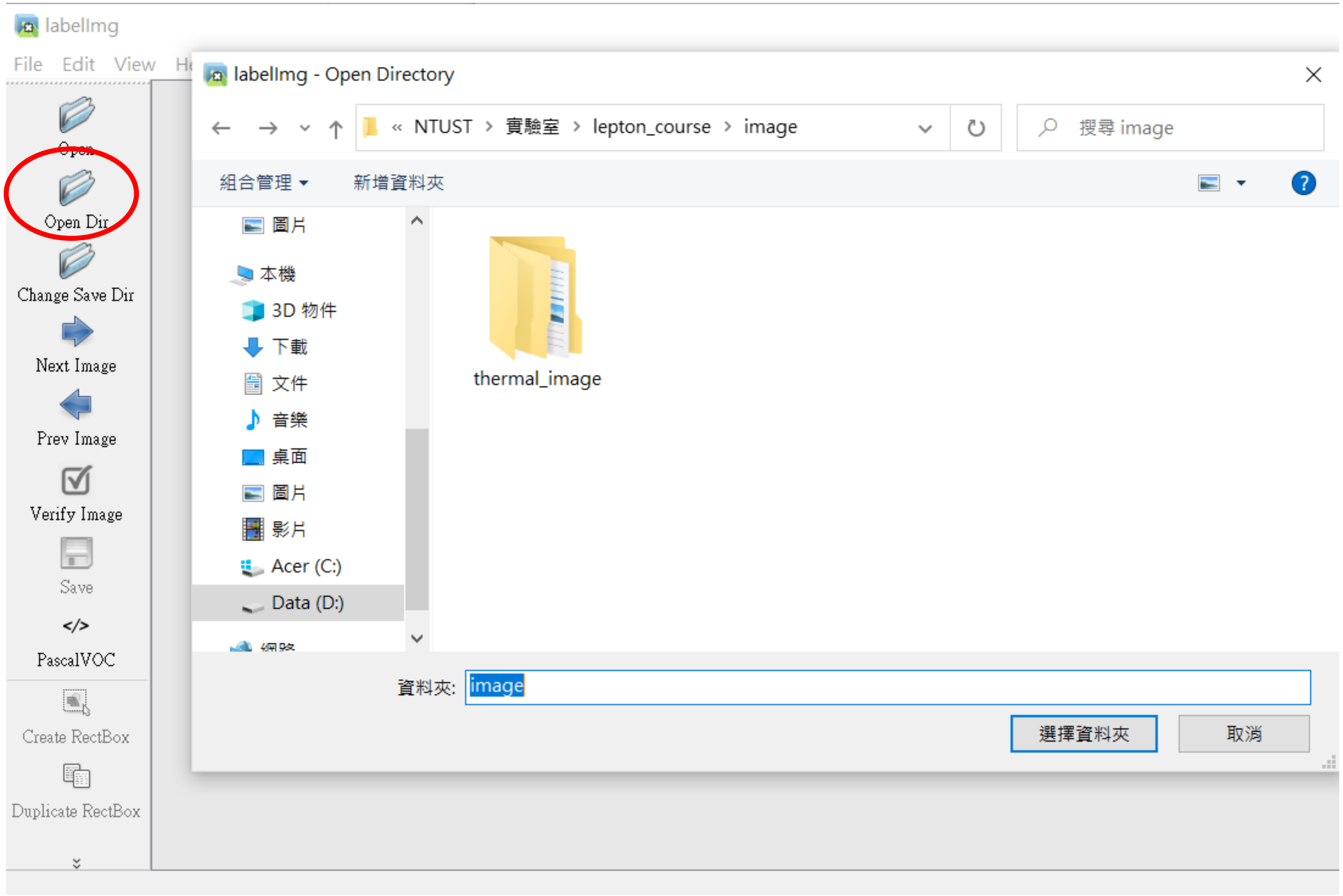
Data preparation

- You will see the GUI as below.



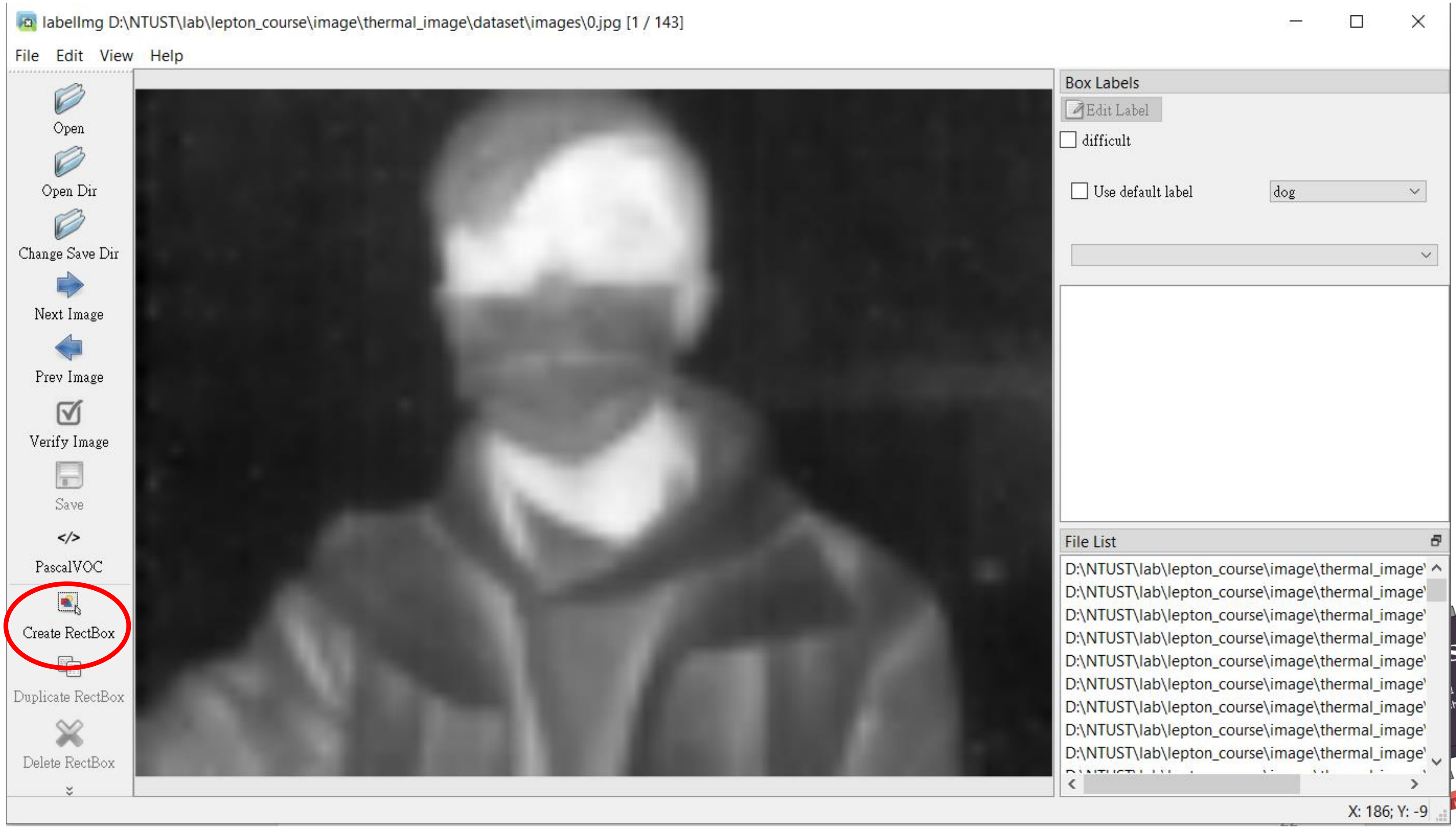
Data preparation

- Select your image folder.



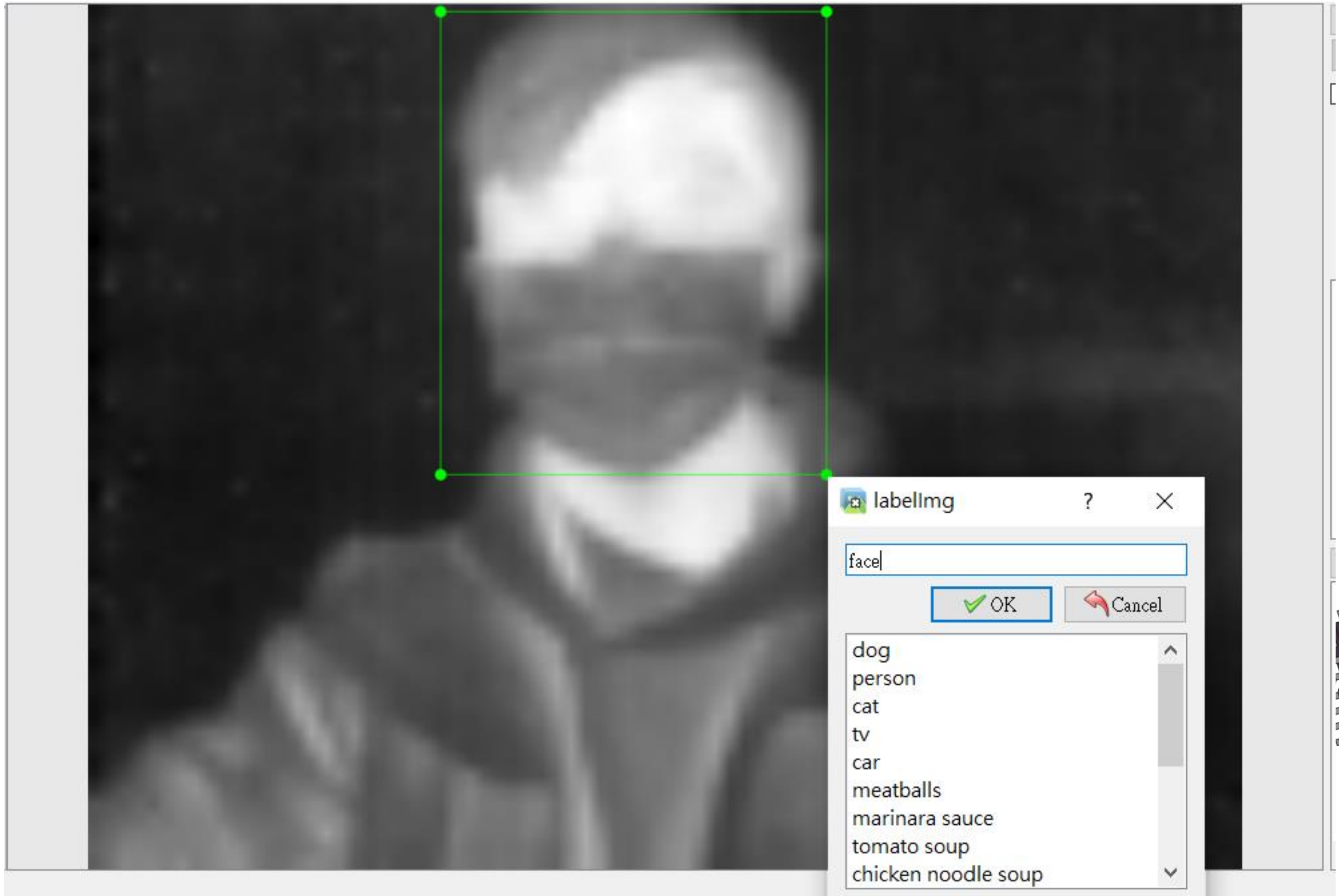
Data preparation

- Create RectBox.



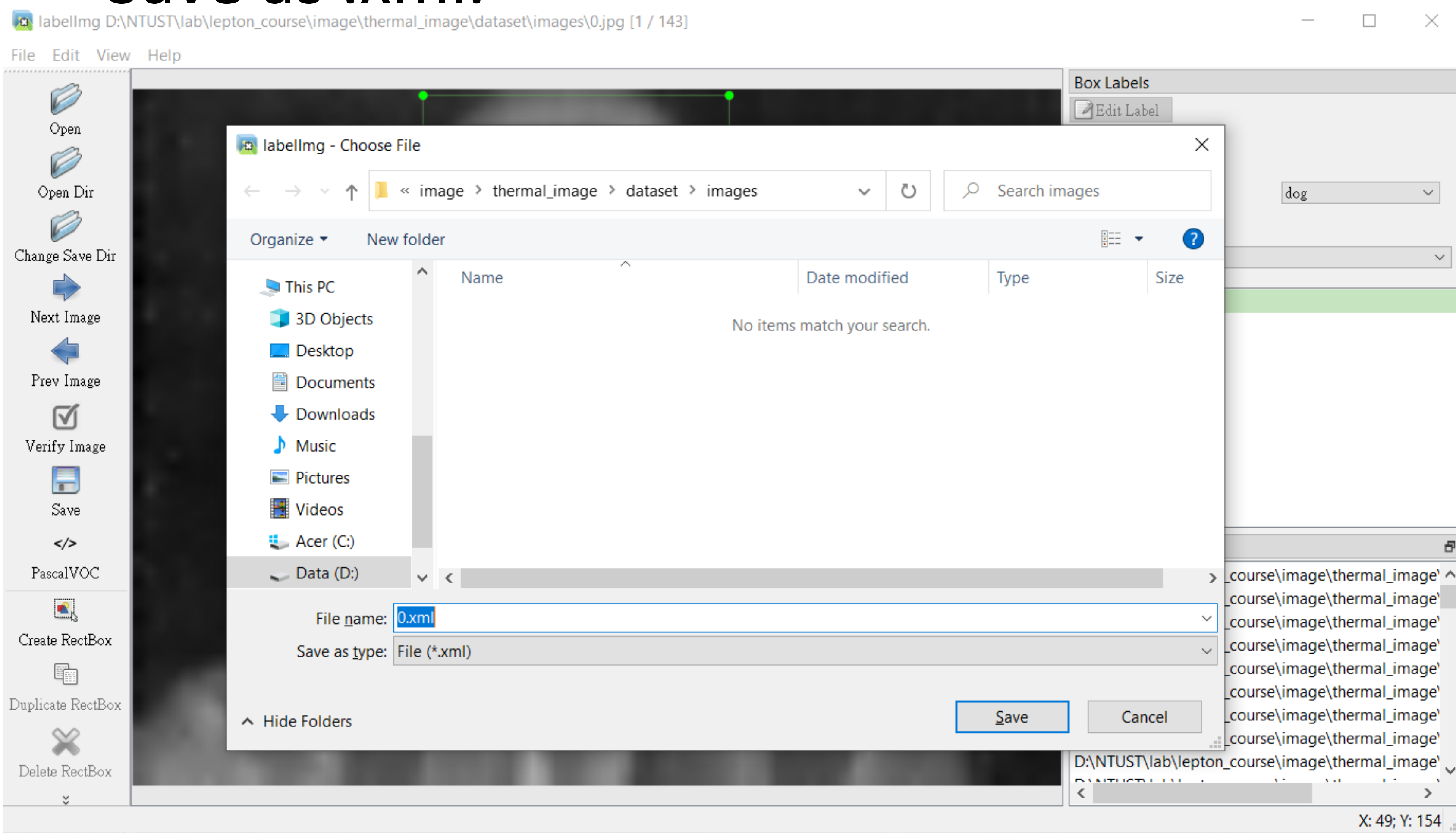
Data preparation

- Label and name the label as face.



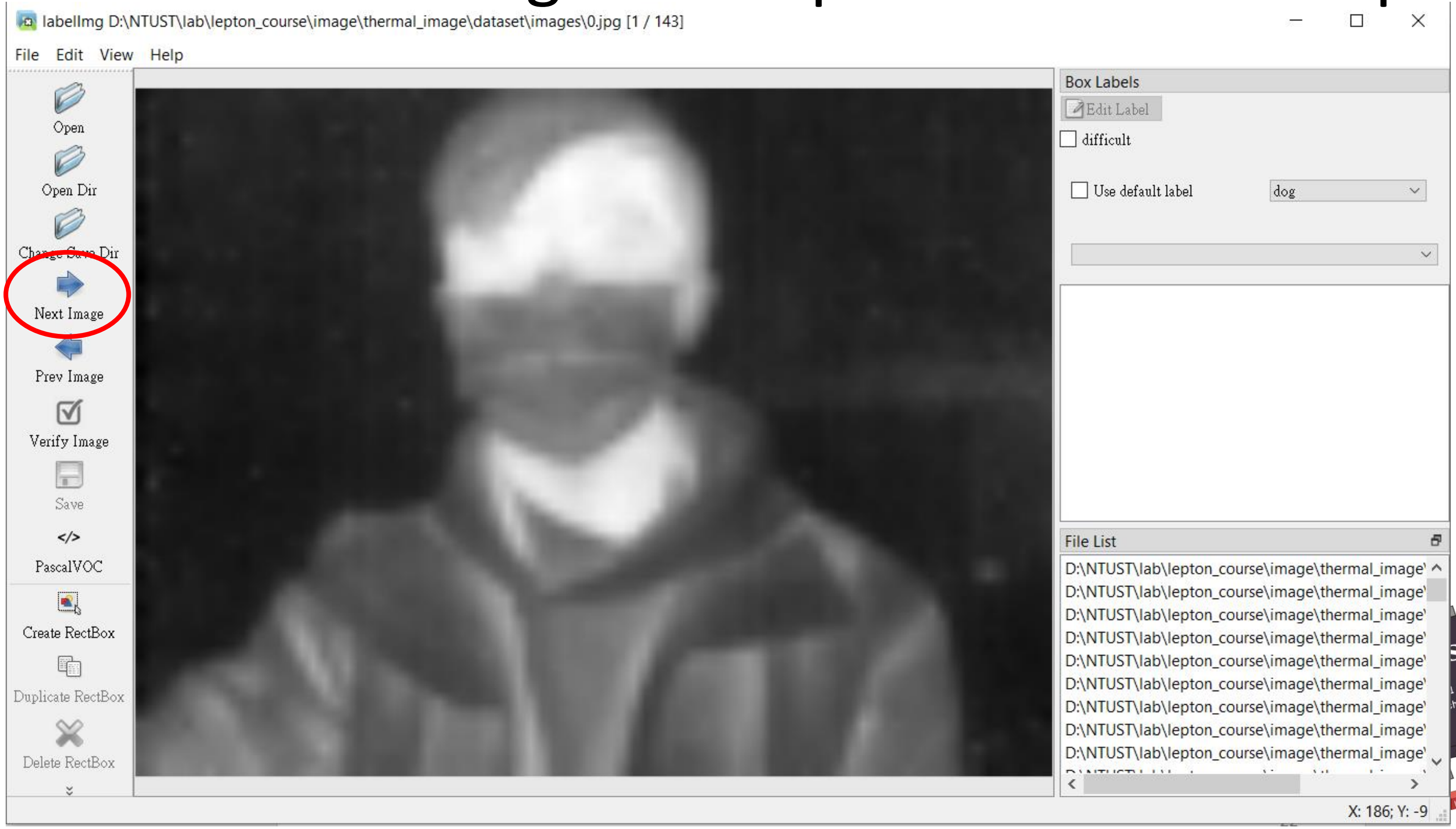
Data preparation

- Save as .xml.



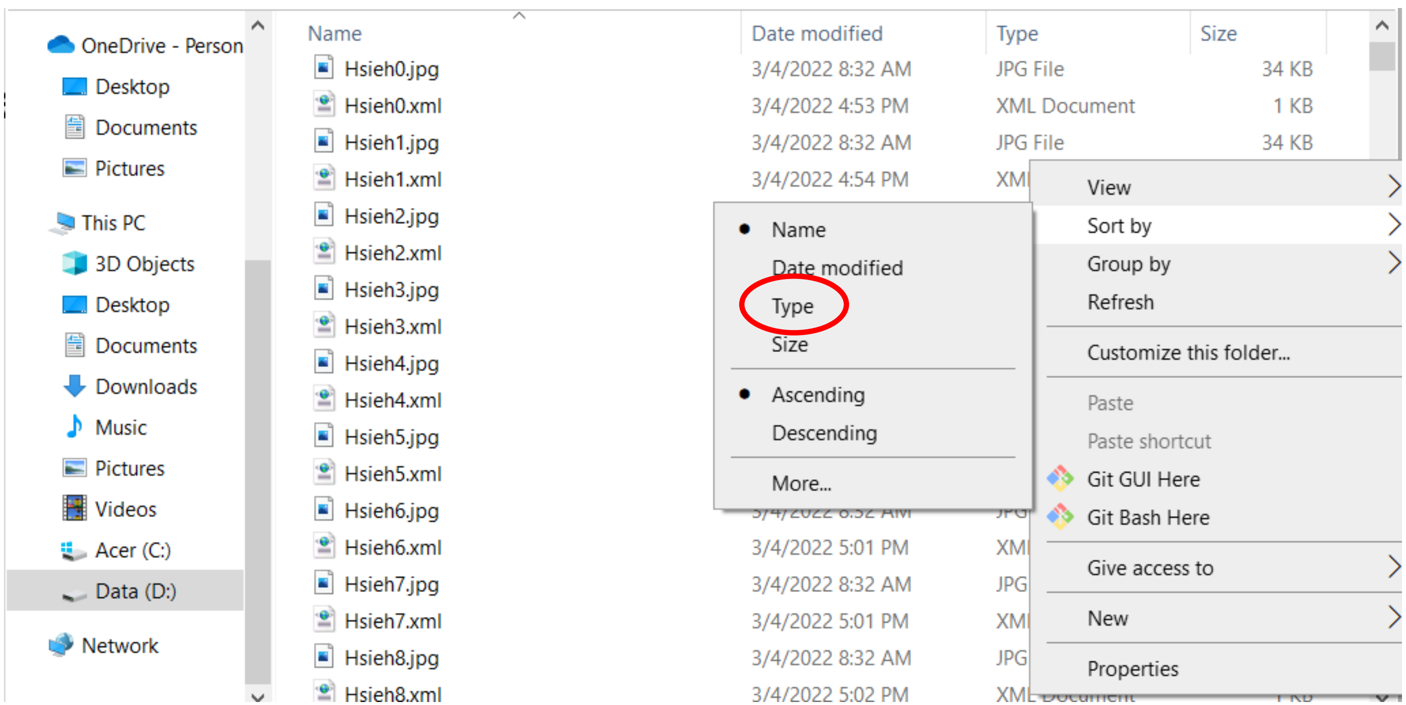
Data preparation

- Press next image and repeat the above steps.



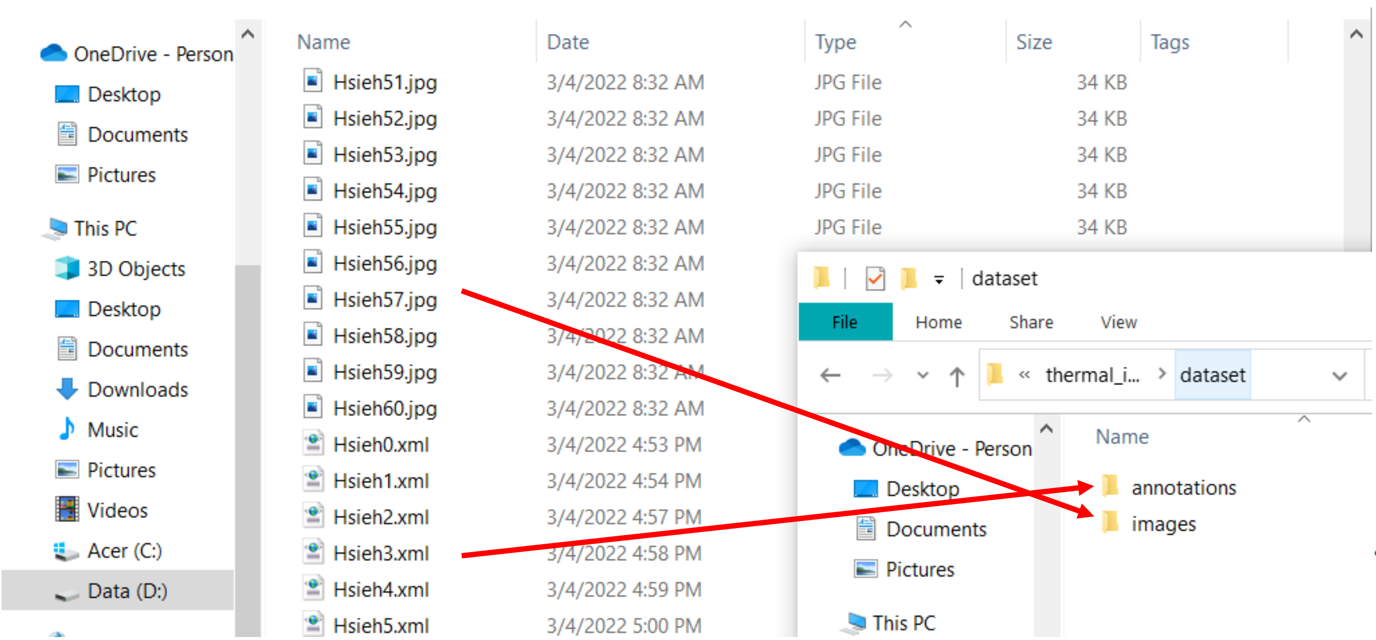
Data preparation

- Prepare dataset.
- Sort the file by type.



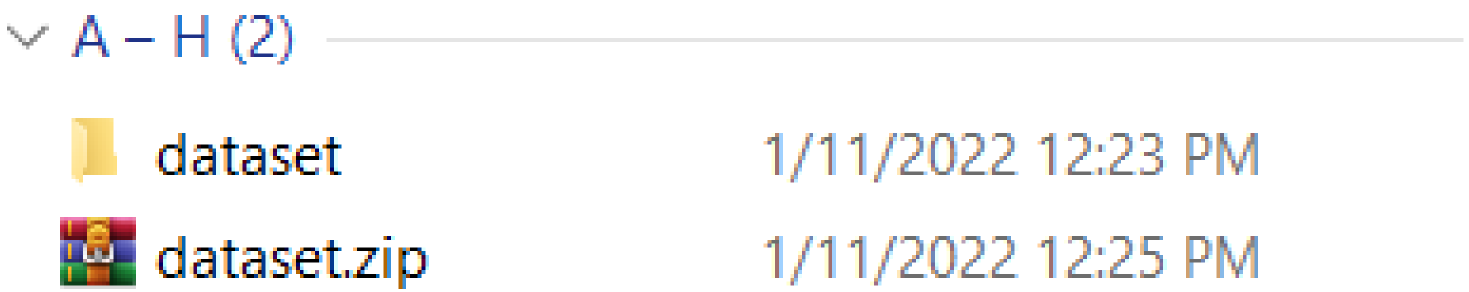
Data preparation

- New a dataset folder shown as below.
- Put .xml into annotations folder and put .jpg into images folder.



Data preparation

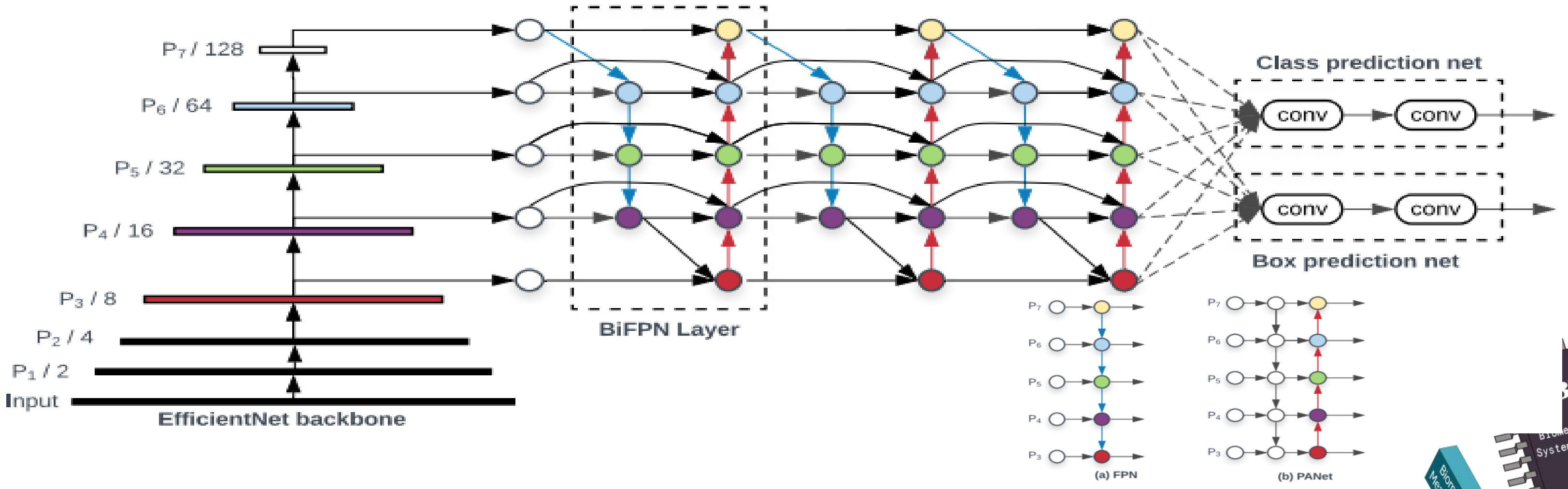
- Compress the dataset folder into “dataset.zip”.



Model architecture & introduction

Model architecture

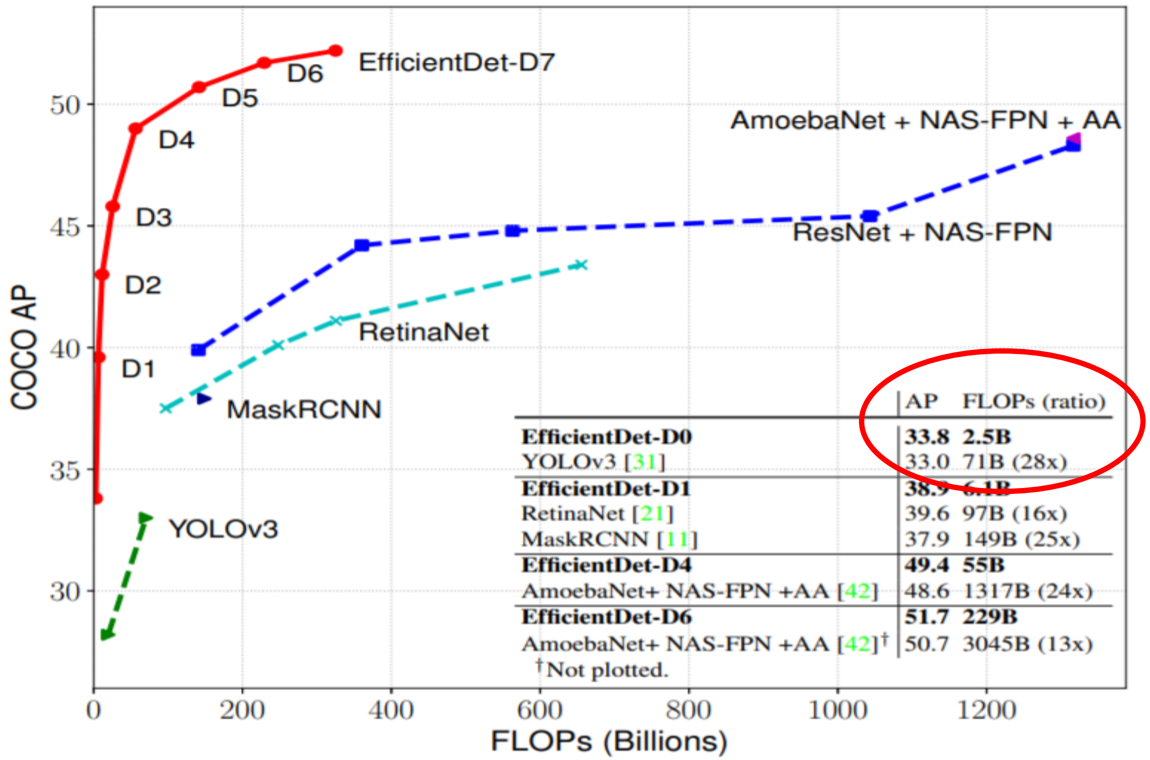
- Use EfficientDet architecture.
- It employs EfficientNet as the backbone network, BiFPN as the feature network , and shared class/box prediction network.



M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, arXiv:1911.09070. [Online]. Available: http://arxiv.org/abs/1911.09070

Model introduction

- EfficientDet performance



	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [31]	33.0	71B (28x)
EfficientDet-D1	**38.9**	**6.1B**
RetinaNet [21]	39.6	97B (16x)
MaskRCNN [11]	37.9	149B (25x)
EfficientDet-D4	**49.4**	**55B**
AmoebaNet+ NAS-FPN +AA [42]	48.6	1317B (24x)
EfficientDet-D6	**51.7**	**229B**
AmoebaNet+ NAS-FPN +AA [42]†	50.7	3045B (13x)

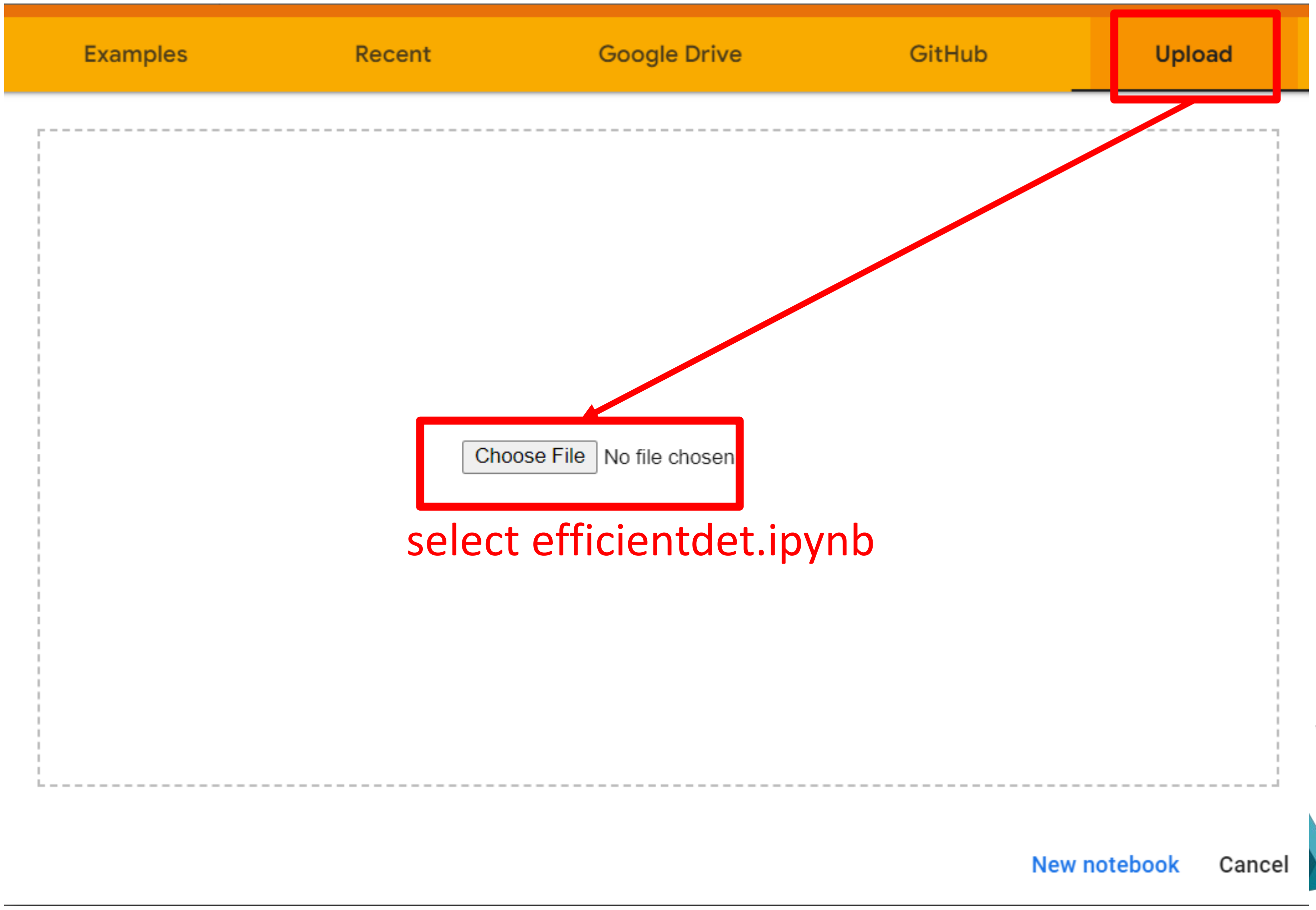
†Not plotted.

https://openaccess.thecvf.com/content_CVPR_2020/html/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.html

Lab 2
Thermal image face detection model training experiment

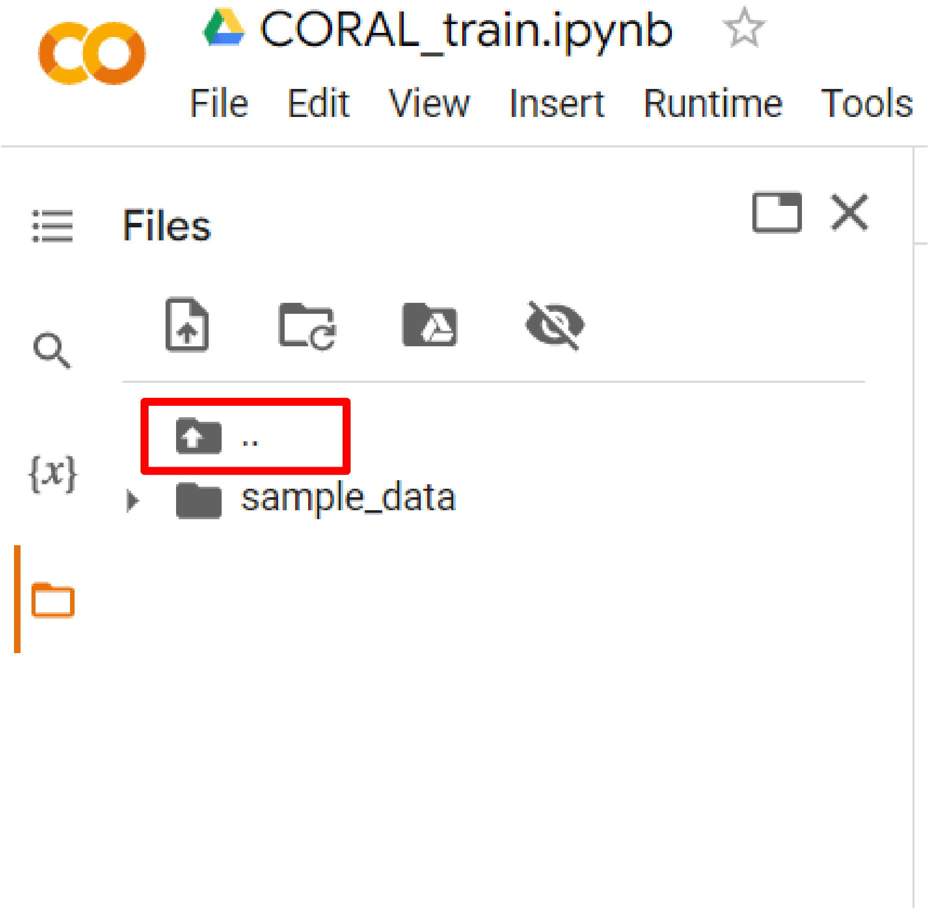
Colab

https://colab.research.google.com/?utm_source=scs-index



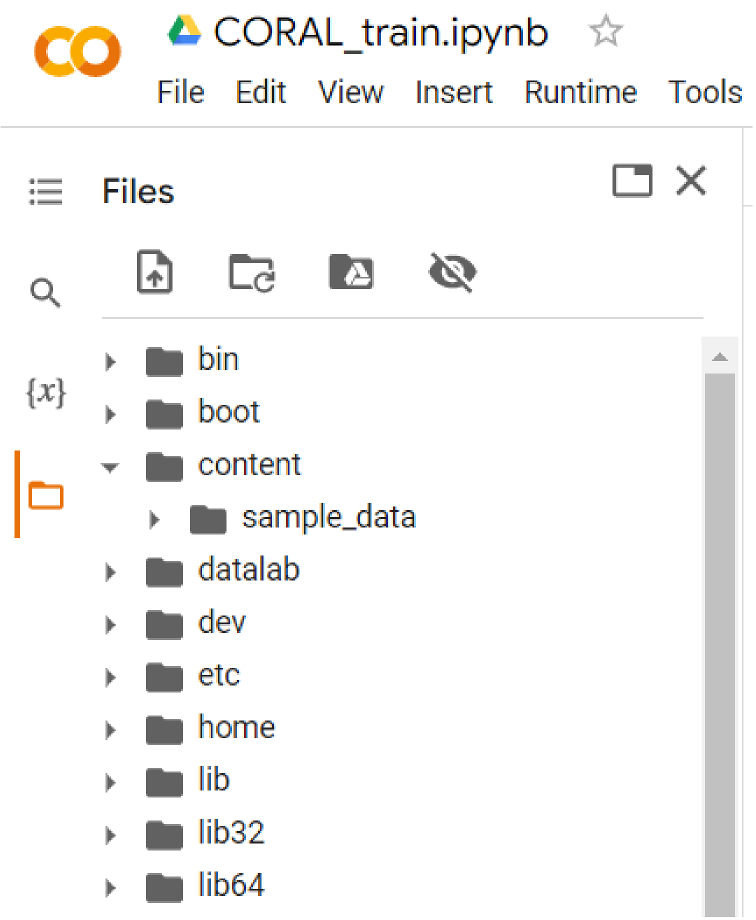
Upload dataset

- Press ".." Folder.



Upload dataset

- Upload dataset.zip to the content folder.



Import packages

- Import the required packages.

import the required packages

```
[ ] !pip install -q tflite-model-maker
```

```
[ ] import numpy as np
    import os

    from tflite_model_maker.config import ExportFormat
    from tflite_model_maker import model_spec
    from tflite_model_maker import object_detector

    import tensorflow as tf
    assert tf.__version__.startswith('2')

    tf.get_logger().setLevel('ERROR')
    from absl import logging
    logging.set_verbosity(logging.ERROR)
```

Set options

- Set dataset options.

Set dataset options

```
[ ]  use_custom_dataset = True
     dataset_is_split = False
```

Split function

- Split dataset function

```
[ ] import os
    import random
    import shutil

    def split_dataset(images_path, annotations_path, val_split, test_split, out_path):
        """Splits a directory of sorted images/annotations into training, validation, and test sets.

        Args:
            images_path: Path to the directory with your images (JPGs).
            annotations_path: Path to a directory with your VOC XML annotation files,
                with filenames corresponding to image filenames. This may be the same path
                used for images_path.
            val_split: Fraction of data to reserve for validation (float between 0 and 1).
            test_split: Fraction of data to reserve for test (float between 0 and 1).
        Returns:
            The paths for the split images/annotations (train_dir, val_dir, test_dir)
        """
```

Load dataset

- Load dataset and split.

Load your Pascal VOC dataset

```
if use_custom_dataset:

    # The ZIP file you uploaded:
    !unzip dataset.zip

    # Your labels map as a dictionary (zero is reserved):
    label_map = {1: 'face'}

    if dataset_is_split:
        # If your dataset is already split, specify each path:
        train_images_dir = 'dataset/train/images'
        train_annotations_dir = 'dataset/train/annotations'
        val_images_dir = 'dataset/validation/images'
        val_annotations_dir = 'dataset/validation/annotations'
        test_images_dir = 'dataset/test/images'
        test_annotations_dir = 'dataset/test/annotations'
    else:
        # If it's NOT split yet, specify the path to all images and annotations
        images_in = 'dataset/images'
        annotations_in = 'dataset/annotations'
```

Split dataset

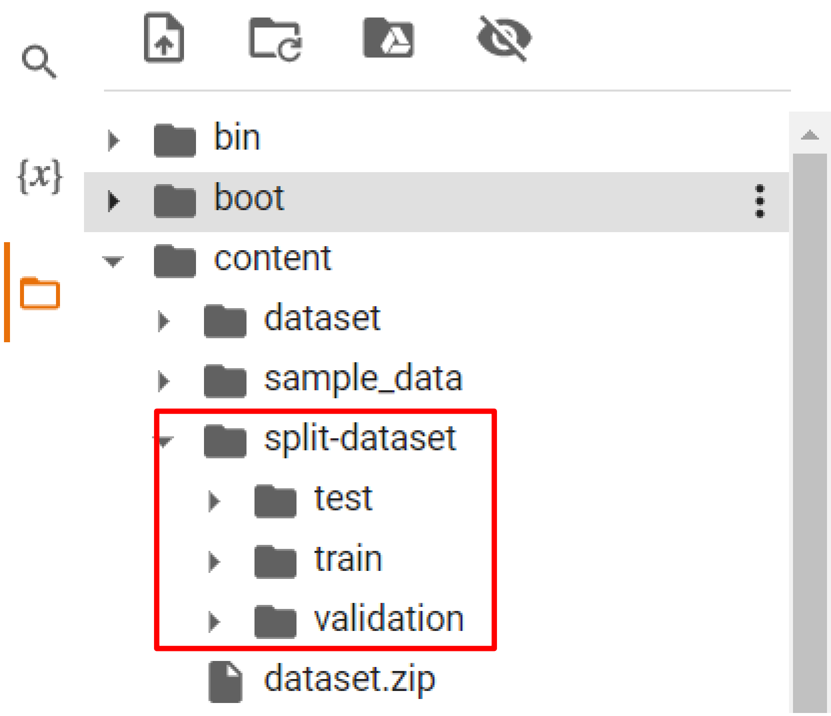
- Split dataset (train, validation, test).

Split the dataset

```
# We need to instantiate a separate DataLoader for each split dataset
if use_custom_dataset:
  if dataset_is_split:
    train_data = object_detector.DataLoader.from_pascal_voc(
        train_images_dir, train_annotations_dir, label_map=label_map)
    validation_data = object_detector.DataLoader.from_pascal_voc(
        val_images_dir, val_annotations_dir, label_map=label_map)
    test_data = object_detector.DataLoader.from_pascal_voc(
        test_images_dir, test_annotations_dir, label_map=label_map)
  else:
    train_dir, val_dir, test_dir = split_dataset(images_in, annotations_in,
                                                 val_split=0.2, test_split=0.2,
                                                 out_path='split-dataset')
    train_data = object_detector.DataLoader.from_pascal_voc(
        os.path.join(train_dir, 'images'),
        os.path.join(train_dir, 'annotations'), label_map=label_map)
    validation_data = object_detector.DataLoader.from_pascal_voc(
        os.path.join(val_dir, 'images'),
        os.path.join(val_dir, 'annotations'), label_map=label_map)
    test_data = object_detector.DataLoader.from_pascal_voc(
        os.path.join(test_dir, 'images'),
        os.path.join(test_dir, 'annotations'), label_map=label_map)
```

Split dataset

- You can see split-dataset folder in content folder.



Model selection

- Select EfficientDet model spec.
- Here, we'll use Lite0.

Select model spec

Model architecture	Size(MB)*	Latency(ms)**	Average Precision***
EfficientDet-Lite0	5.7	37.4	30.4%
EfficientDet-Lite1	7.6	56.3	34.3%
EfficientDet-Lite2	10.2	104.6	36.0%
EfficientDet-Lite3	14.4	107.6	39.4%

** File size of the compiled Edge TPU models.*
*** Latency measured on a desktop CPU with a Coral USB Accelerator.*
**** Average Precision is the mAP (mean Average Precision) on the COCO 2017 validation dataset.*

```
[ ]  spec = object_detector.EfficientDetLite0Spec()
```

Train model

Create and train model

Now we need to create our model according to the model spec, load our dataset into the model, specify training parameters, and begin training.

Using Model Maker, we accomplished all of that with `create()`:

```
[ ]  model = object_detector.create(train_data=train_data,
                                    model_spec=spec,
                                    validation_data=validation_data,
                                    epochs=20,
                                    batch_size=10,
                                    train_whole_model=False)
```

Args	
`train_data`	Training data.
`model_spec`	Specification for the model.
`validation_data`	Validation data. If None, skips validation process.
`epochs`	Number of epochs for training.
`batch_size`	Batch size for training.
`train_whole_model`	Boolean, False by default. If true, train the whole model. Otherwise, only train the layers that are not match `model_spec.config.var_freeze_expr`.

Model evaluation

Predicted label

True label

	P	N
P	TP	FN
N	FP	TN

Evaluate the model

Now we'll use the test dataset to evaluate how well the model performs with data it has never seen before.

The `evaluate()` method provides output in the style of COCO evaluation metrics:

```
[ ]  model.evaluate(test_data)
```

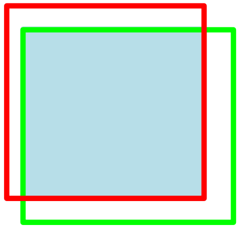
```
{'AP': 0.7412693,
 'AP50': 1.0,
 'AP75': 0.88781303,
 'AP_/face': 0.7412693,
 'APl': 0.7412693,
 'APm': -1.0,
 'APs': -1.0,
 'ARl': 0.8214286,
 'ARm': -1.0,
 'ARmax1': 0.76428574,
 'ARmax10': 0.8214286,
 'ARmax100': 0.8214286,
 'ARs': -1.0}
```

IOU = 0.5 : 0.05 : 0.95

IOU = 0.5

IOU = 0.75

IOU = 0.7



$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

Export model

- Export the model to tflite.

Export to TensorFlow Lite

```
[ ] TFLITE_FILENAME = 'efficientdet-lite-thermal-face.tflite'
    LABELS_FILENAME = 'thermal-face.txt'
```

```
[ ] model.export(export_dir='.', tflite_filename=TFLITE_FILENAME, label_filename=LABELS_FILENAME,
                 export_format=[ExportFormat.TFLITE, ExportFormat.LABEL])
```

Test model

- Pick a random picture of test dataset to test.

Test the model

[12] 0 秒

```
import random

# If you're using a custom dataset, we take a random image from the test set:
if use_custom_dataset:
    images_path = test_images_dir if dataset_is_split else os.path.join(test_dir, "images")
    filenames = os.listdir(os.path.join(images_path))
    random_index = random.randint(0,len(filenames)-1)
    INPUT_IMAGE = os.path.join(images_path, filenames[random_index])
```

Test model

- Load model, resize input image and inference.

```
# Load the TF Lite model
labels = read_label_file(LABELS_FILENAME)
interpreter = tflite.Interpreter(TFLITE_FILENAME)
interpreter.allocate_tensors()

# Resize the image for input
image = Image.open(INPUT_IMAGE)
_, scale = common.set_resized_input(
        interpreter, image.size, lambda size: image.resize(size, Image.ANTIALIAS))

# Run inference
interpreter.invoke()
objs = detect.get_objects(interpreter, score_threshold=0.4, image_scale=scale)
```

Test model

```
display_width = 500
scale_factor = display_width / image.width
height_ratio = image.height / image.width
image = image.resize((display_width, int(display_width * height_ratio)))
draw_objects(ImageDraw.Draw(image), objs, scale_factor, labels)
image
```

Compilation

- Download the compiler and compile.

Download the edge TPU compiler

```
[ ] ! curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
    ! echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
    ! sudo apt-get update
    ! sudo apt-get install edgetpu-compiler
```

```
[ ] NUMBER_OF_TPUS = 1
    !edgetpu_compiler $TFLITE_FILENAME -d --num_segments=$NUMBER_OF_TPUS
```

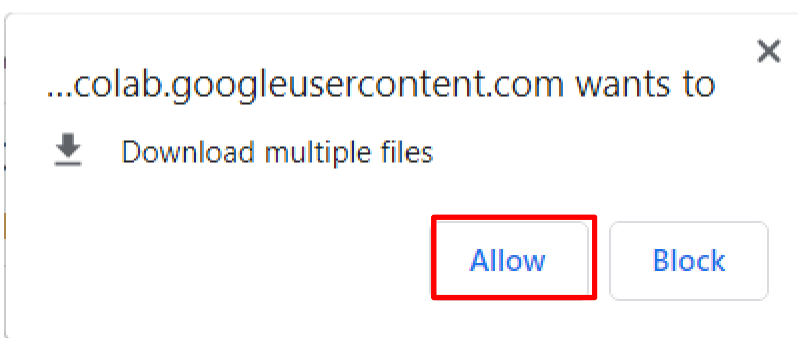
Model architecture	Minimum TPUs	Recommended TPUs
EfficientDet-Lite0	1	1
EfficientDet-Lite1	1	1
EfficientDet-Lite2	1	2
EfficientDet-Lite3	2	2
EfficientDet-Lite4	2	3

Download model

- Download and press Allow.

Download the model

```
from google.colab import files

files.download(TFLITE_FILENAME)
files.download(TFLITE_FILENAME.replace('.tflite', '_edgetpu.tflite'))
files.download(LABELS_FILENAME)
```



Download model

- Find your model in Downloads folder.

This PC > Downloads

Search Downloads

Name	Date modified	Type
Today (3)		
thermal-face.txt	5/30/2022 7:47 PM	Text Document
efficientdet-lite-thermal-face.tflite	5/30/2022 7:48 PM	TFLITE File
efficientdet-lite-thermal-face_edgetpu.tflite	5/30/2022 7:48 PM	TFLITE File

Assignment 2

- Train your own thermal image face detection model.
- You can also use other models.
- Your own model will be tested by TAs.
  - Must be able to label TAs' face stably and measure the temperature every frame.

Submission

- Hierarchy :
  - Dataset folder
  - Efficientdet.ipynb
  - Report.pdf
- Compress all above files in a single zip file named StudentID_lab2.zip.

References

- M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, arXiv:1911.09070. [Online]. Available: http://arxiv.org/abs/1911.09070
- Coral, "Retrain EfficientDet-Lite detector for the Edge TPU." [Online]. Available: https://colab.research.google.com/github/google-coral/tutorials/blob/master/retrain_efficientdet_model_maker_tf2.ipynb.