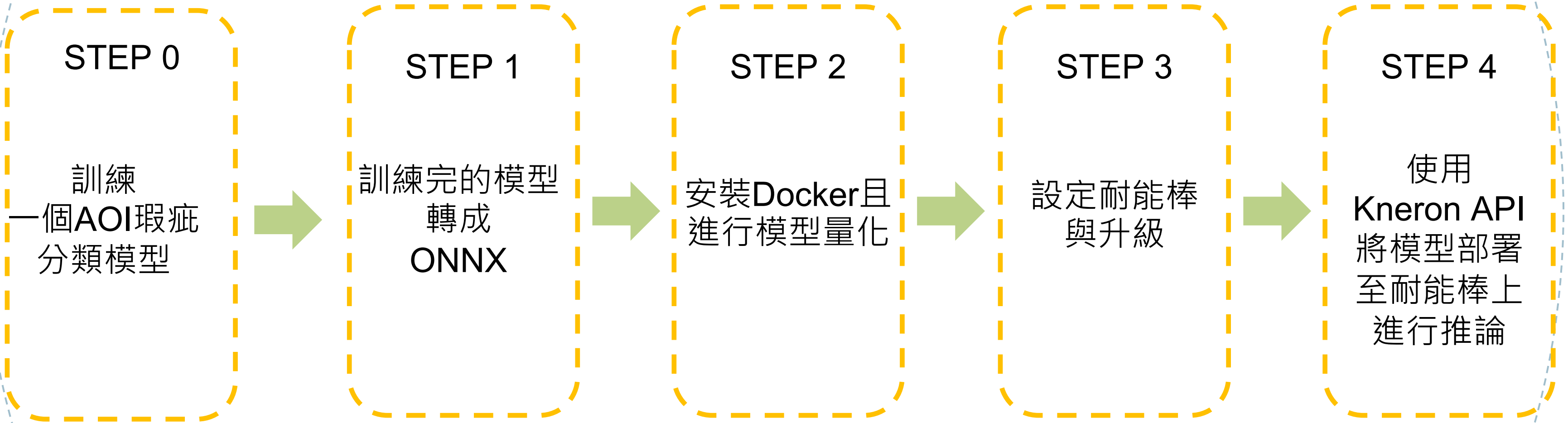
Edge AI
耐能棒使用介紹

2022/05/25

流程步驟



STEP 0

訓練
一個AOI瑕疵
分類模型

STEP 1

訓練完的模型
轉成
ONNX

STEP 2

安裝Docker且
進行模型量化

STEP 3

設定耐能棒
與升級

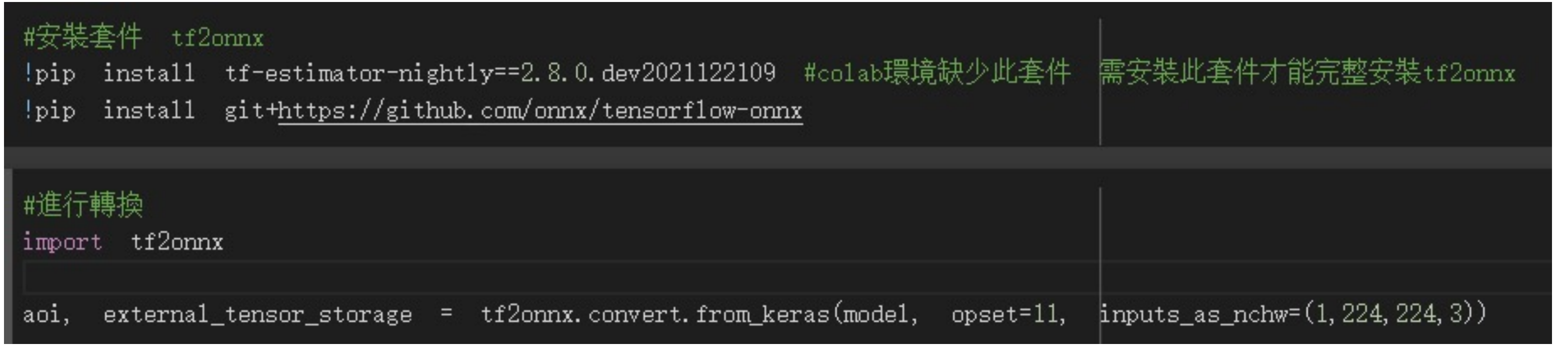
STEP 4

使用
Kneron API
將模型部署
至耐能棒上
進行推論

STEP 1

- 將模型轉成ONNX

(ONNX使不同的人工智慧框架可以採用相同格式存儲模型數據)



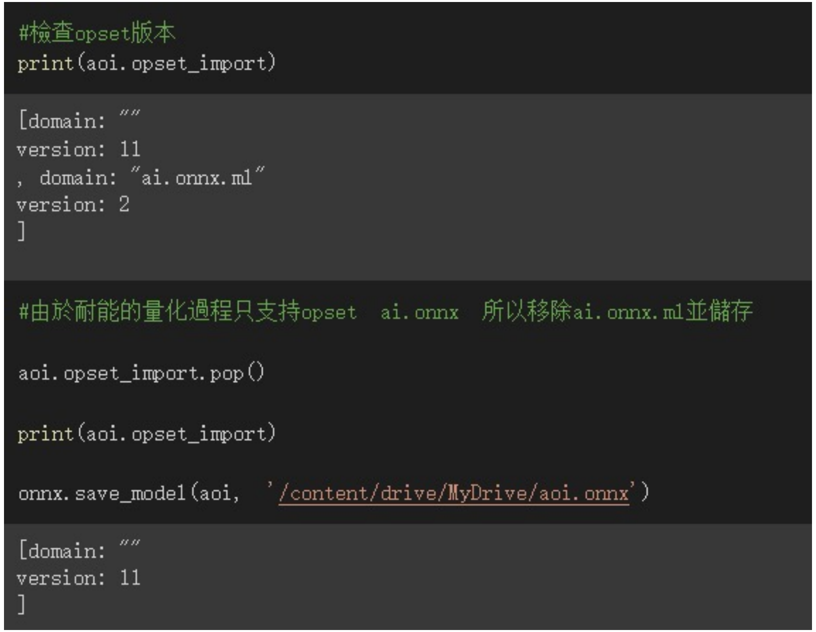
```
#安裝套件 tf2onnx
!pip install tf-estimator-nightly==2.8.0.dev2021122109 #colab環境缺少此套件 需安裝此套件才能完整安裝tf2onnx
!pip install git+https://github.com/onnx/tensorflow-onnx
```

```
#進行轉換
import tf2onnx

aoi, external_tensor_storage = tf2onnx.convert.from_keras(model, opset=11, inputs_as_nchw=(1,224,224,3))
```

STEP 1

- 由於 Keras 轉 ONNX 的 opset 會多 ai.onnx.ml 版本

  ，但耐能不支持 ai.onnx.ml，所以刪掉此版本



```
#檢查opset版本
print(aoi.opset_import)
```

```
[domain: ""
version: 11
, domain: "ai.onnx.ml"
version: 2
]
```

```
#由於耐能的量化過程只支持opset　ai.onnx　所以移除ai.onnx.ml並儲存

aoi.opset_import.pop()

print(aoi.opset_import)

onnx.save_model(aoi, '/content/drive/MyDrive/aoi.onnx')
```

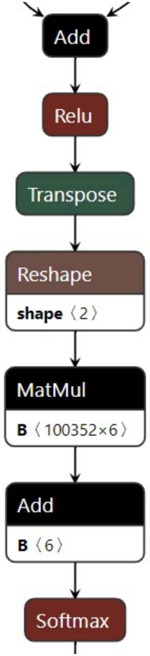
```
[domain: ""
version: 11
]
```

STEP 1

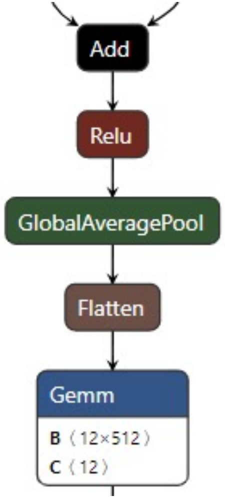
- Keras 轉出的 ONNX 模型，會出現耐能不支持的模型結構，

  但 Pytorch 並不會

Keras



Pytorch

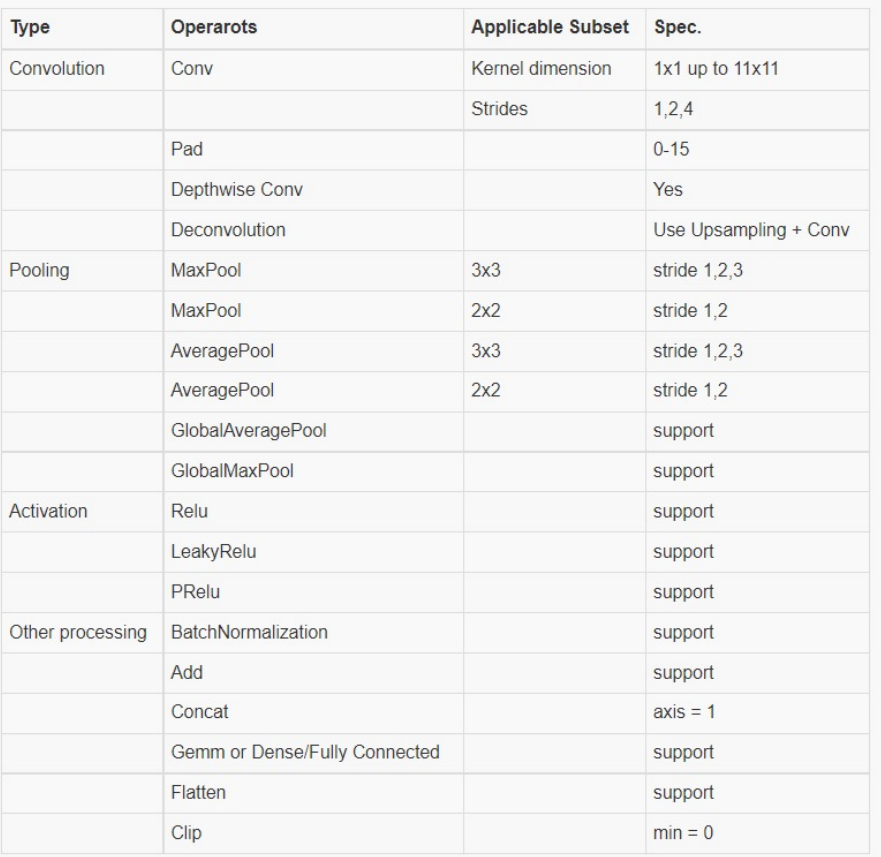


STEP 1

- 不支持模型結構有 mul / div 與 Sigmoid / Softmax，主要出現在 Dence 層，所以可以將 Dence 層移至耐能棒模型推論的後處理
- 為了節省時間，後續耐能示範將會提供從 Pytorch 轉好的 ONNX 模型，以方便同學們操作，有興趣的同學可以自己研究 Keras 版本或詢問耐能論壇，會有工程師解答問題
- 不支持模型結構說明：http://doc.kneron.com/docs/#toolchain/converters/#6-onnx-to-onnx-onnx-optimization

STEP 1

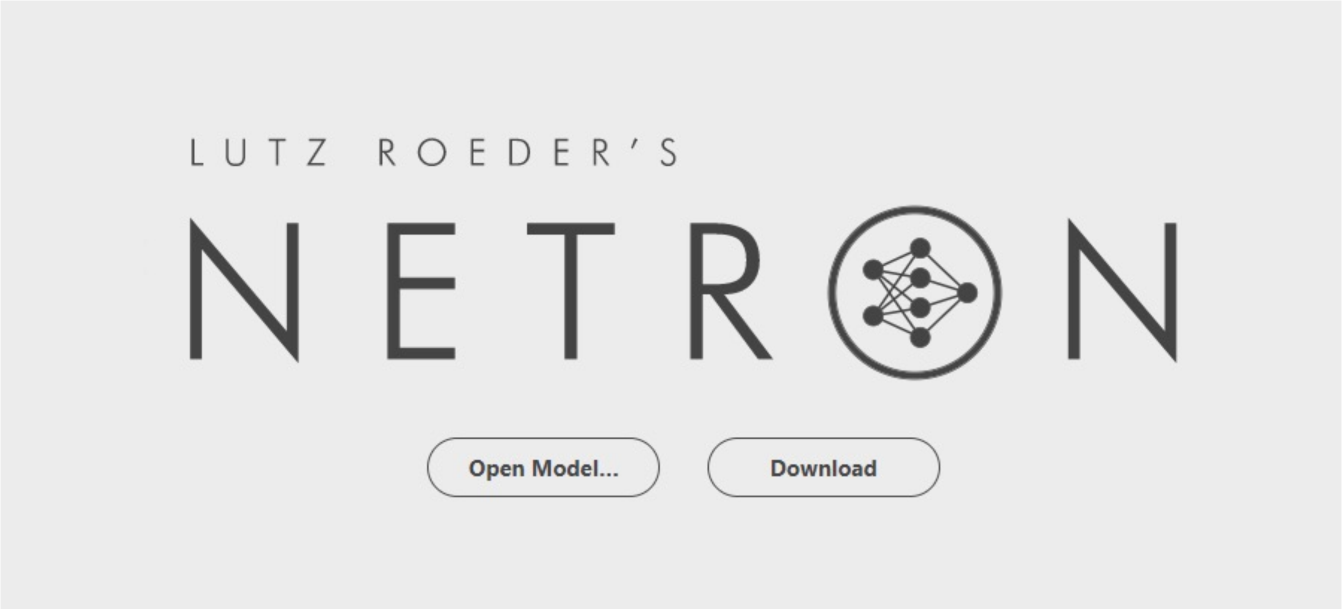
耐能KL520
支持的模型結構



Type	Operarots	Applicable Subset	Spec.
Convolution	Conv	Kernel dimension	1x1 up to 11x11
		Strides	1,2,4
	Pad		0-15
	Depthwise Conv		Yes
	Deconvolution		Use Upsampling + Conv
Pooling	MaxPool	3x3	stride 1,2,3
	MaxPool	2x2	stride 1,2
	AveragePool	3x3	stride 1,2,3
	AveragePool	2x2	stride 1,2
	GlobalAveragePool		support
	GlobalMaxPool		support
Activation	Relu		support
	LeakyRelu		support
	PRelu		support
Other processing	BatchNormalization		support
	Add		support
	Concat		axis = 1
	Gemm or Dense/Fully Connected		support
	Flatten		support
	Clip		min = 0

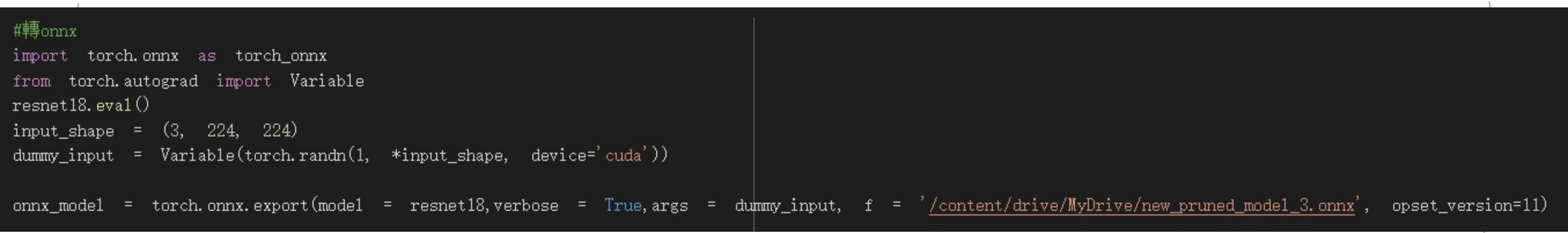
STEP 1

- 查詢模型結構https://netron.app/



STEP 1

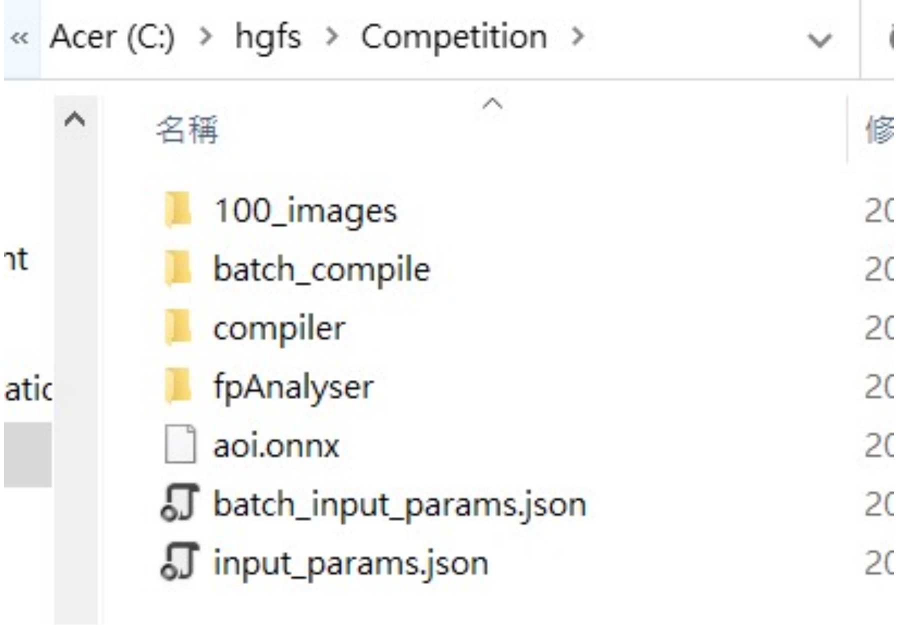
- Pytorch 轉 ONNX (參考)



```
#轉onnx
import torch.onnx as torch_onnx
from torch.autograd import Variable
resnet18.eval()
input_shape = (3, 224, 224)
dummy_input = Variable(torch.randn(1, *input_shape, device='cuda'))

onnx_model = torch.onnx.export(model = resnet18,verbose = True,args = dummy_input, f = '/content/drive/MyDrive/new_pruned_model_3.onnx', opset_version=11)
```

STEP 1

- aoi.onnx檔案移置/c/hgfs/Competition (已放置)



« Acer (C:) › hgfs › Competition ›

名稱

100_images

batch_compile

compiler

fpAnalyser

aoi.onnx

batch_input_params.json

input_params.json

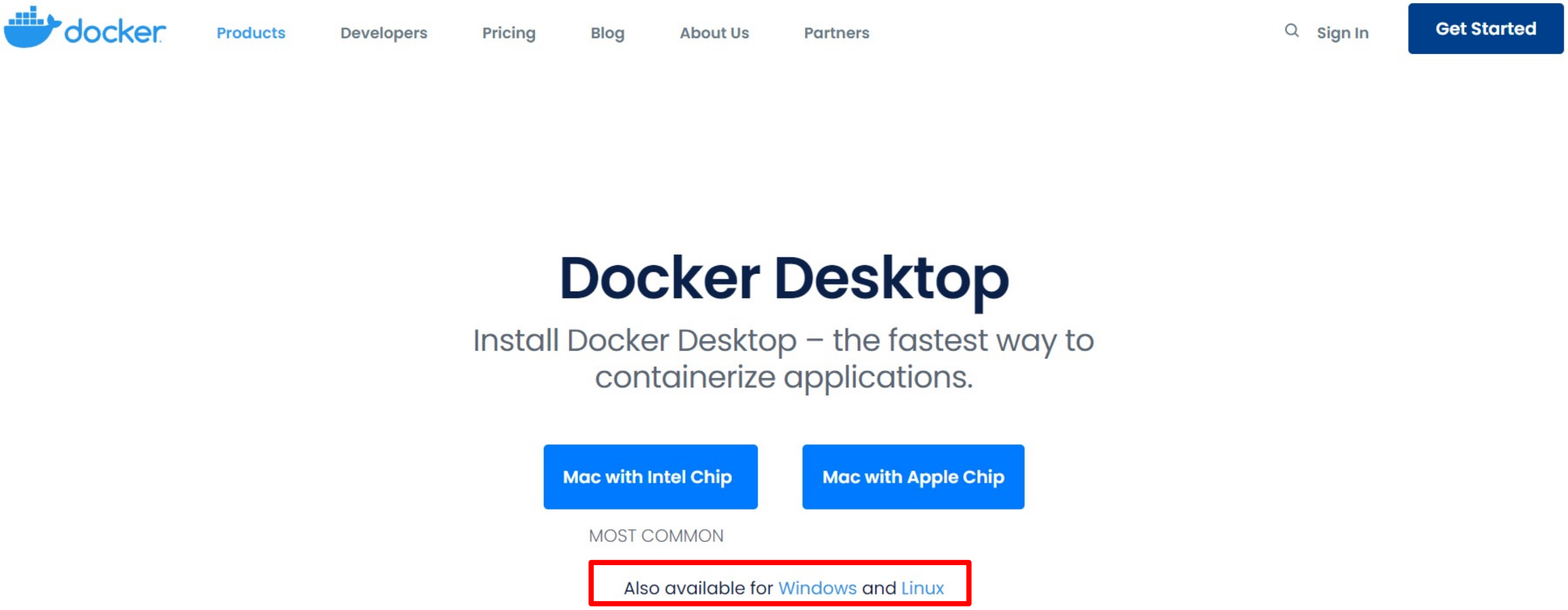
STEP 2

- 安裝 WSL https://docs.microsoft.com/zh-tw/windows/wsl/install
- 舊版 WSL 的手動安裝步驟 https://docs.microsoft.com/zh-tw/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
- 開始使用 Docker https://docs.microsoft.com/zh-tw/windows/wsl/tutorials/wsl-containers

STEP 2

- 安裝 Docker https://www.docker.com/products/docker-desktop/

(整體操作很佔空間，空間小於 50 GB的同學，請先清一下 C 槽)



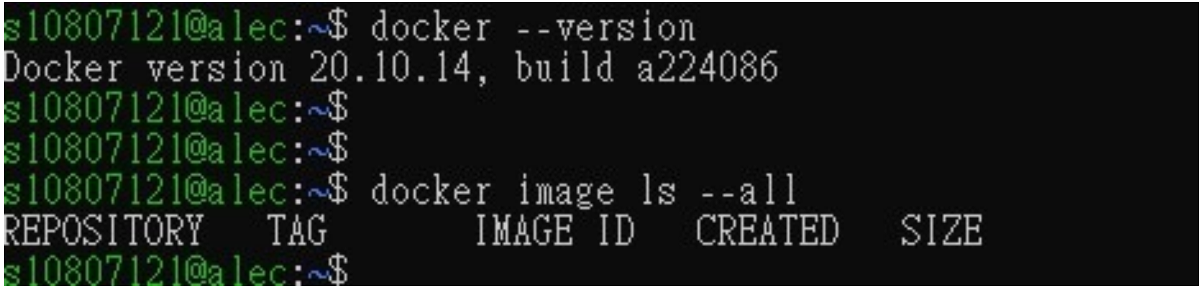
Docker Desktop

Install Docker Desktop – the fastest way to containerize applications.

Also available for Windows and Linux

STEP 2

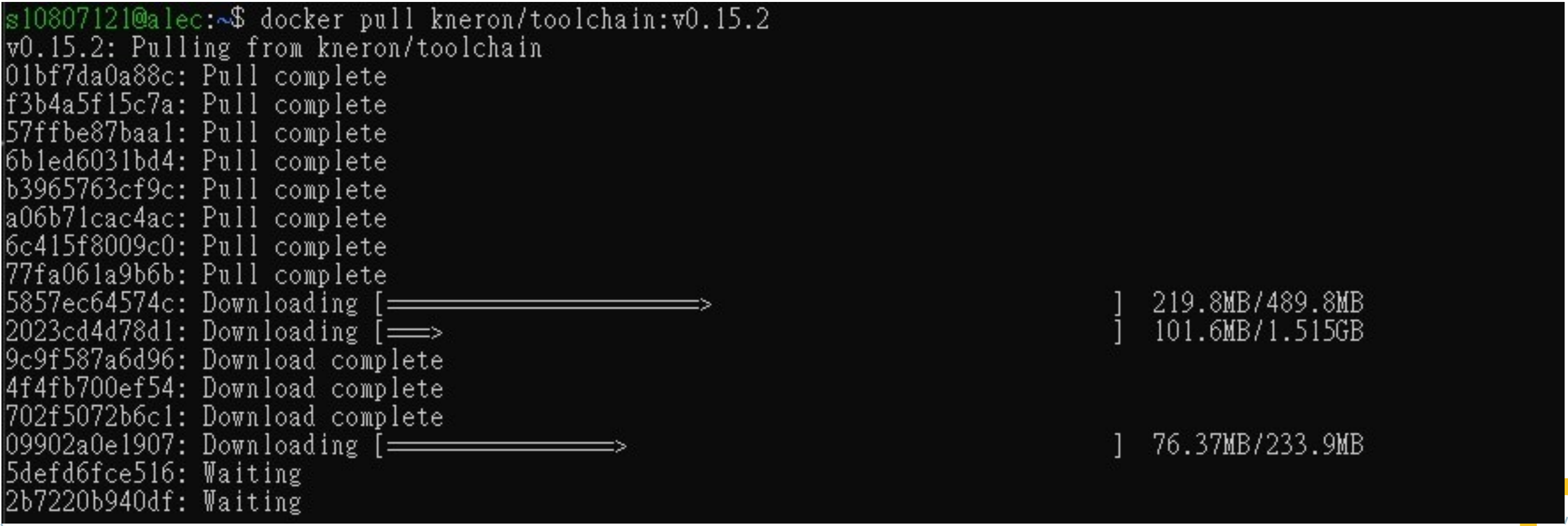
- 打開ubuntu 輸入以下指令，查看是否安裝成功



```
s1080712l@alec:~$ docker --version
Docker version 20.10.14, build a224086
s1080712l@alec:~$
s1080712l@alec:~$
s1080712l@alec:~$ docker image ls --all
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
s1080712l@alec:~$
```

STEP 2

- 下載 image

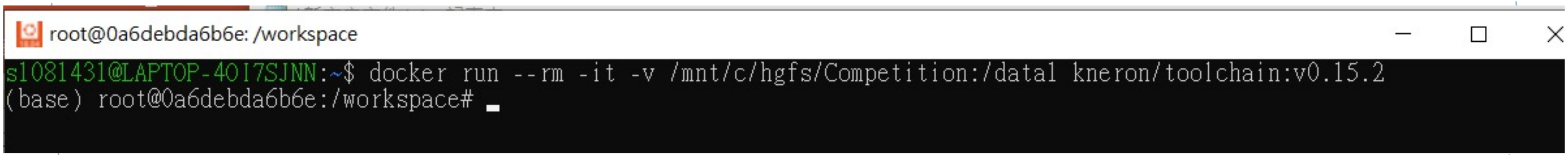


```
s10807121@alec:~$ docker pull kneron/toolchain:v0.15.2
v0.15.2: Pulling from kneron/toolchain
01bf7da0a88c: Pull complete
f3b4a5f15c7a: Pull complete
57ffbe87baa1: Pull complete
6b1ed6031bd4: Pull complete
b3965763cf9c: Pull complete
a06b71cac4ac: Pull complete
6c415f8009c0: Pull complete
77fa061a9b6b: Pull complete
5857ec64574c: Downloading [=====================>                             ]  219.8MB/489.8MB
2023cd4d78d1: Downloading [===>                                               ]  101.6MB/1.515GB
9c9f587a6d96: Download complete
4f4fb700ef54: Download complete
702f5072b6c1: Download complete
09902a0e1907: Downloading [================>                                  ]  76.37MB/233.9MB
5defd6fce516: Waiting
2b7220b940df: Waiting
```

STEP 2

- 打開Ubuntu 18.04.5 LTS

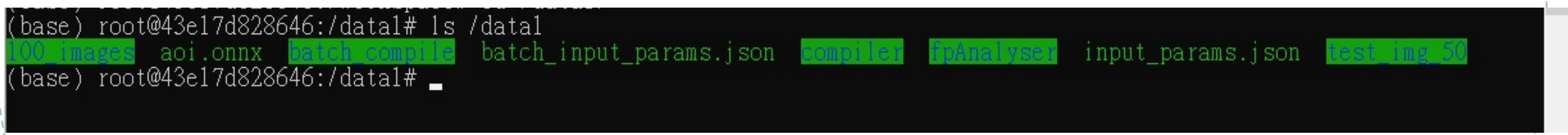
- 透過下面指令將/c/hgfs/Competition掛載到docker的/data1上



```
s1081431@LAPTOP-4017SJNN:~$ docker run --rm -it -v /mnt/c/hgfs/Competition:/data1 kneron/toolchain:v0.15.2
(base) root@0a6debda6b6e:/workspace# 
```

STEP 2

- 進入 /data1
- 查看 /data1 的內容

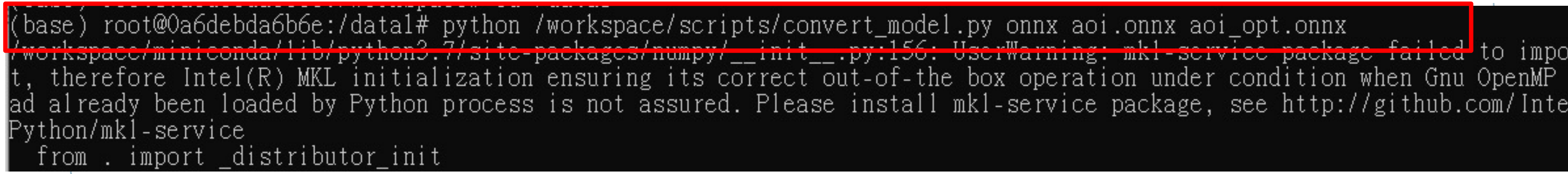


```
(base) root@43e17d828646:/data1# ls /data1
100_images  aoi.onnx  batch_compile  batch_input_params.json  compiler  fpAnalyser  input_params.json  test_img_50
(base) root@43e17d828646:/data1# _
```

STEP 2

- 將模型轉換成 NPU 所需的最佳化 ONNX 架構

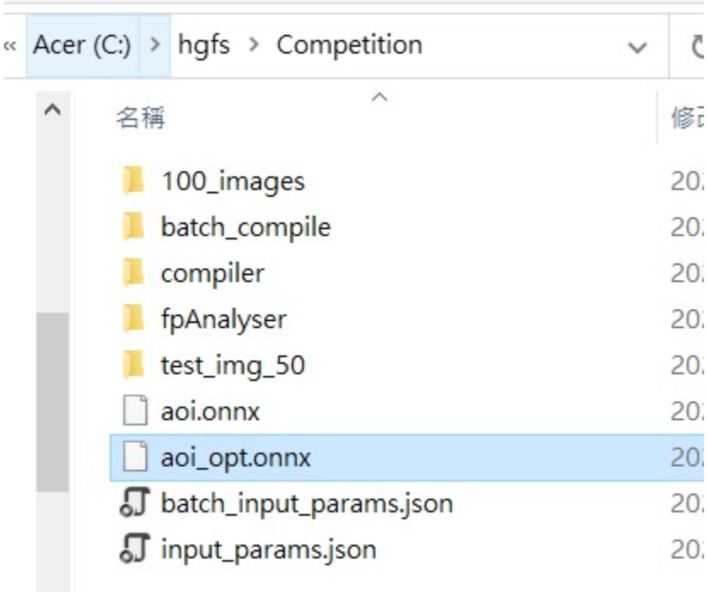
- 執行 onnx optimizator



```
(base) root@0a6debda6b6e:/data1# python /workspace/scripts/convert_model.py onnx aoi.onnx aoi_opt.onnx
/workspace/miniconda/lib/python3.7/site-packages/numpy/__init__.py:156: UserWarning: mkl-service package failed to impo
t, therefore Intel(R) MKL initialization ensuring its correct out-of-the box operation under condition when Gnu OpenMP
ad already been loaded by Python process is not assured. Please install mkl-service package, see http://github.com/Inte
Python/mkl-service
  from . import _distributor_init
```

STEP 2

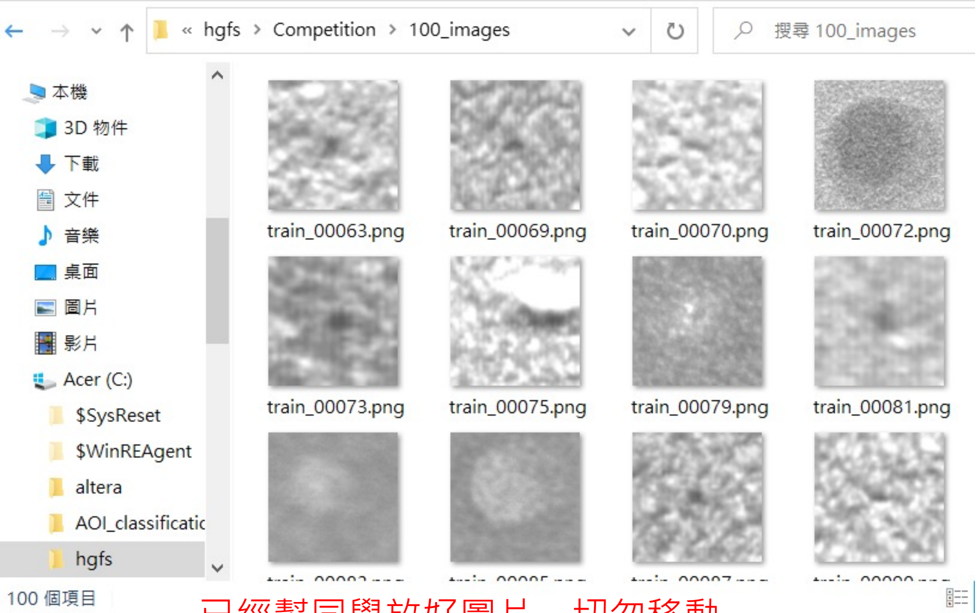
- 將模型轉換成NPU所需的最佳化架構
- 執行 onnx optimizator



名稱	修ā
100_images	20:
batch_compile	20:
compiler	20:
fpAnalyser	20:
test_img_50	20:
aoi.onnx	20:
aoi_opt.onnx	20:
batch_input_params.json	20:
input_params.json	20:

STEP 2

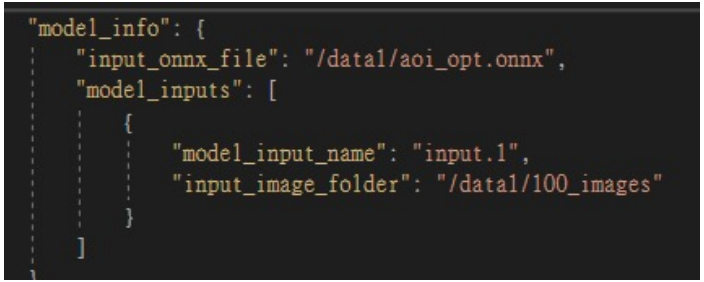
- 轉換模型成量化模型(INT8)前，需先準備好100張與模型相關的圖片



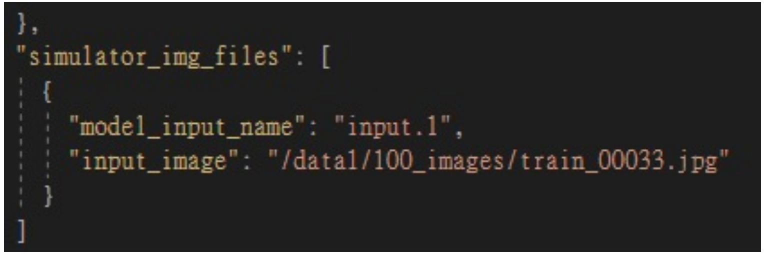
已經幫同學放好圖片，切勿移動

STEP 2

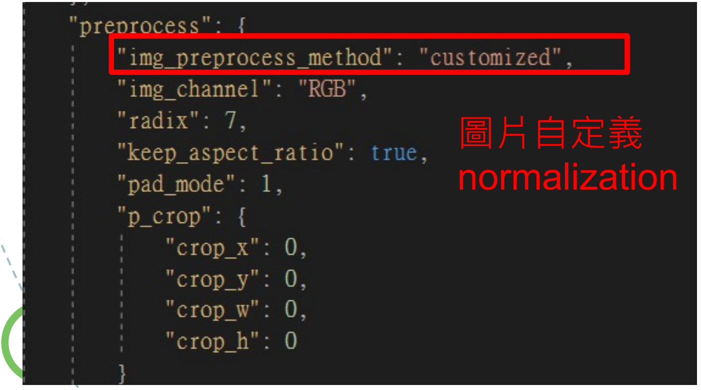
- 修改參數設定檔 input_params.json



```
"model_info": {
    "input_onnx_file": "/data1/aoi_opt.onnx",
    "model_inputs": [
        {
            "model_input_name": "input.1",
            "input_image_folder": "/data1/100_images"
        }
    ]
```

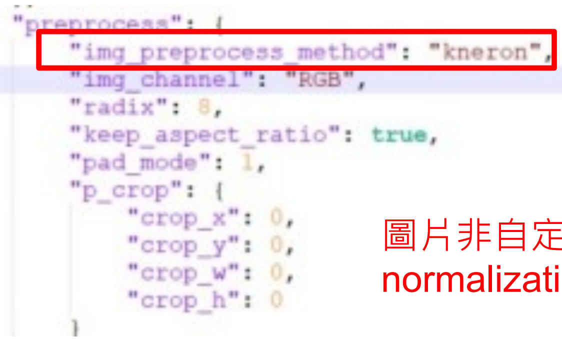


```
},
"simulator_img_files": [
    {
        "model_input_name": "input.1",
        "input_image": "/data1/100_images/train_00033.jpg"
    }
]
```



```
"preprocess": {
    "img_preprocess_method": "customized",
    "img_channel": "RGB",
    "radix": 7,
    "keep_aspect_ratio": true,
    "pad_mode": 1,
    "p_crop": {
        "crop_x": 0,
        "crop_y": 0,
        "crop_w": 0,
        "crop_h": 0
    }
```

圖片自定義
normalization



```
"preprocess": {
    "img_preprocess_method": "kneron",
    "img_channel": "RGB",
    "radix": 8,
    "keep_aspect_ratio": true,
    "pad_mode": 1,
    "p_crop": {
        "crop_x": 0,
        "crop_y": 0,
        "crop_w": 0,
        "crop_h": 0
    }
```

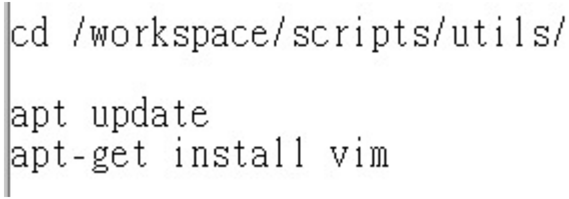
圖片非自定義
normalization

STEP 2

- 圖片自定義 normalization 是為了將100_images正規化0~1

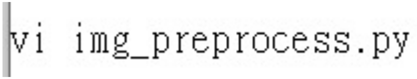
  (原先訓練 tensorflow 模型時，圖片正規化至0~1)

- 先進入資料夾，且安裝 vim



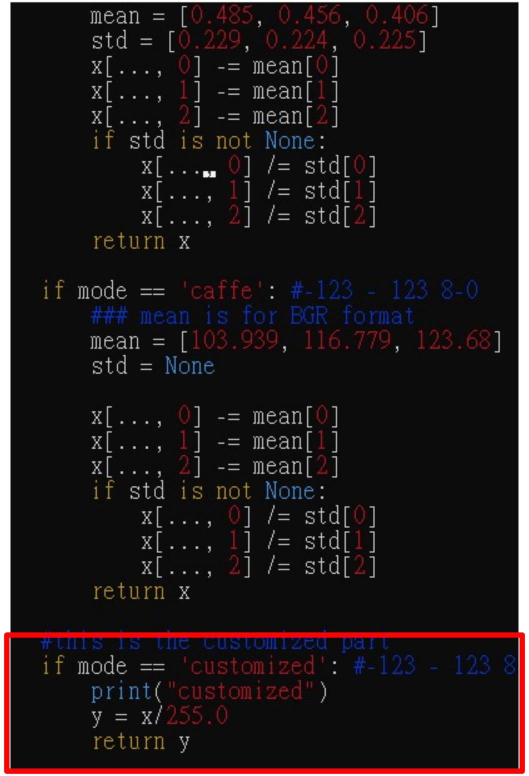
```
cd /workspace/scripts/utils/

apt update
apt-get install vim
```

- 修改 img_preprocess.py



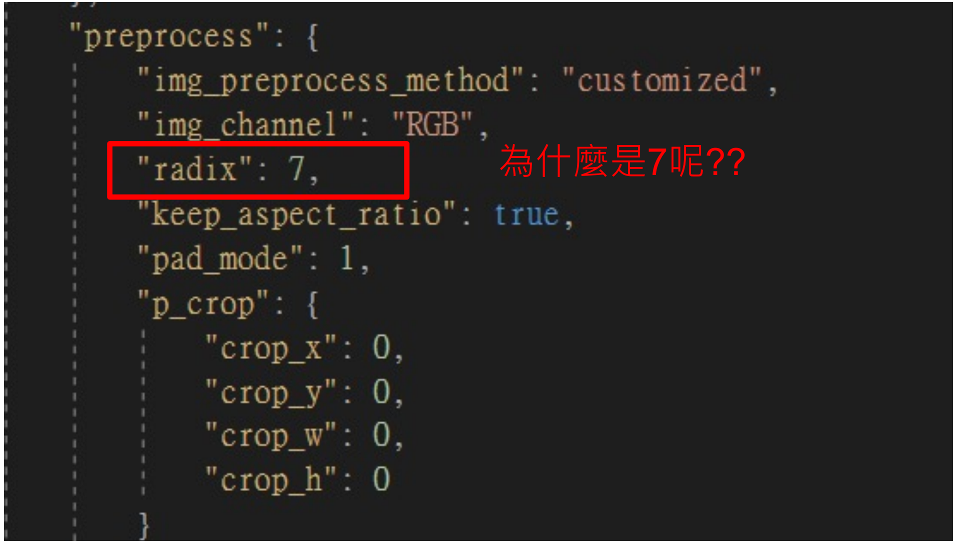
```
vi img_preprocess.py
```

STEP 2



```
        mean = [0.485, 0.456, 0.406]
        std = [0.229, 0.224, 0.225]
        x[..., 0] -= mean[0]
        x[..., 1] -= mean[1]
        x[..., 2] -= mean[2]
        if std is not None:
            x[..., 0] /= std[0]
            x[..., 1] /= std[1]
            x[..., 2] /= std[2]
        return x

    if mode == 'caffe': #-123 - 123 8-0
        ### mean is for BGR format
        mean = [103.939, 116.779, 123.68]
        std = None

        x[..., 0] -= mean[0]
        x[..., 1] -= mean[1]
        x[..., 2] -= mean[2]
        if std is not None:
            x[..., 0] /= std[0]
            x[..., 1] /= std[1]
            x[..., 2] /= std[2]
        return x

    #this is the customized part
    if mode == 'customized': #-123 - 123 8
        print("customized")
        y = x/255.0
        return y
```

STEP 2



```
"preprocess": {
    "img_preprocess_method": "customized",
    "img_channel": "RGB",
    "radix": 7,
    "keep_aspect_ratio": true,
    "pad_mode": 1,
    "p_crop": {
        "crop_x": 0,
        "crop_y": 0,
        "crop_w": 0,
        "crop_h": 0
    }
```

為什麼是7呢??

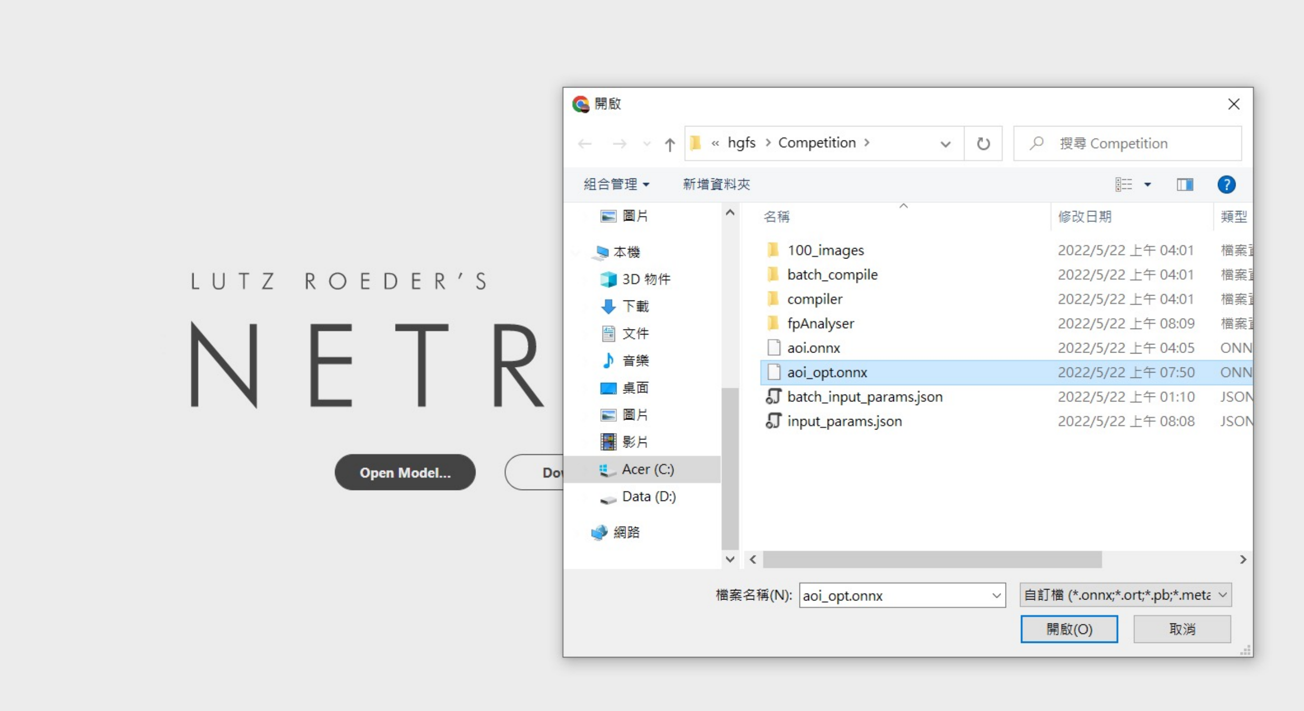
STEP 2

- 量化INT8的世界裡，資料能夠儲存的範圍只有 -128 ~ 127
- 假設浮點狀態下的模型，輸入為 -1.0~1.0，則可以將 radix 設為 7 ，耐能的NPU的量化運算會自動將圖片除以 2^7

- 由於大部分的影像都為 RGB 通道，數值範圍為 0 ~ 255，為了因應量化，所以要在影像前處理時要除以 2.0，將數值範圍變成 0 ~ 127，再透過耐能的量化運算變成 0 ~ 1.0

STEP 2

- 點選 https://netron.app/，查看 aoi_opt.onnx



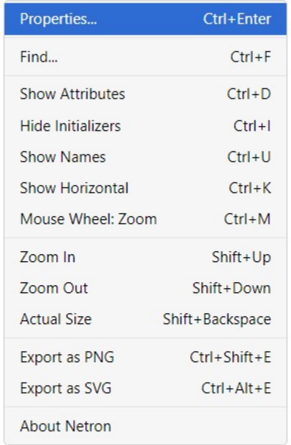
名稱	修改日期	類型
100_images	2022/5/22 上午 04:01	檔案
batch_compile	2022/5/22 上午 04:01	檔案
compiler	2022/5/22 上午 04:01	檔案
fpAnalyser	2022/5/22 上午 08:09	檔案
aoi.onnx	2022/5/22 上午 04:05	ONN
aoi_opt.onnx	2022/5/22 上午 07:50	ONN
batch_input_params.json	2022/5/22 上午 01:10	JSON
input_params.json	2022/5/22 上午 08:08	JSON

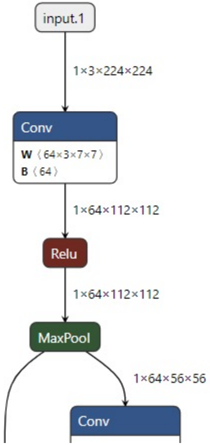
檔案名稱(N): aoi_opt.onnx

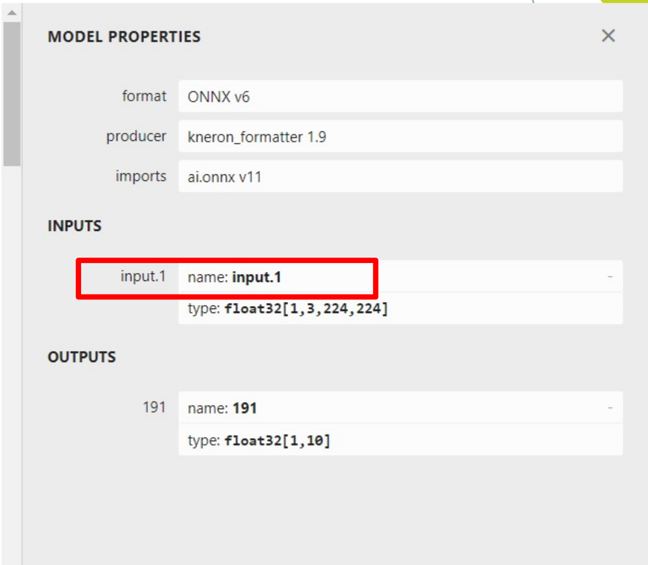
自訂檔 (*.onnx;*.ort;*.pb;*.meta

STEP 2

- 查看 aoi_opt.onnx 的 INPUTS name







MODEL PROPERTIES

format: ONNX v6

producer: kneron_formatter 1.9

imports: ai.onnx v11

INPUTS

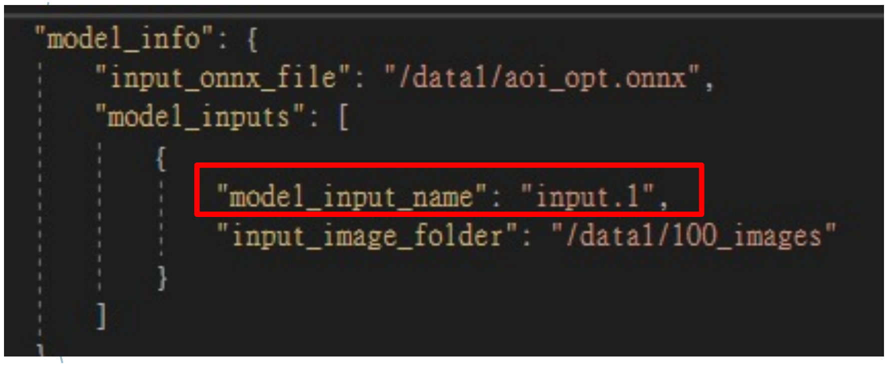
input.1 name: **input.1**
type: `float32[1,3,224,224]`

OUTPUTS

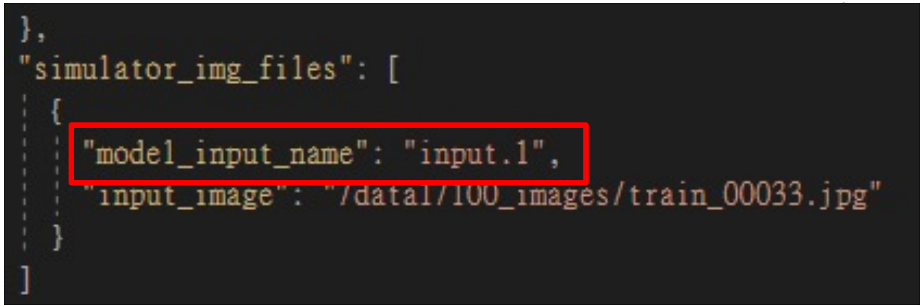
191 name: **191**
type: `float32[1,10]`

STEP 2

- 並將 input.1 填入 model_input_name



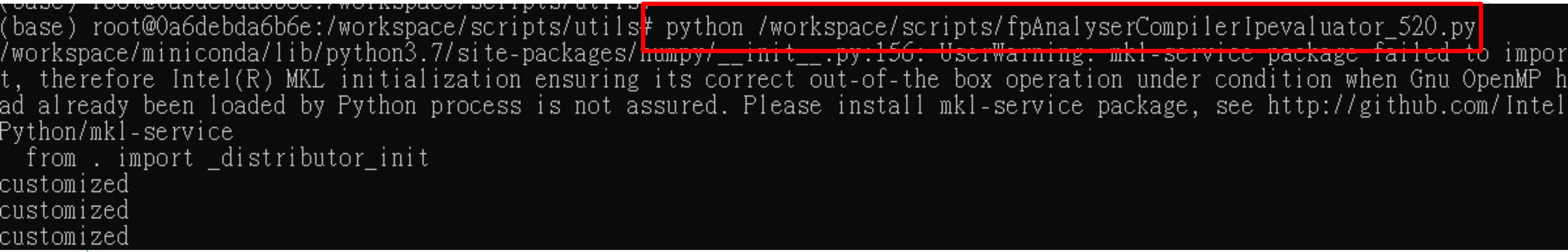
```
"model_info": {
    "input_onnx_file": "/data1/aoi_opt.onnx",
    "model_inputs": [
        {
            "model_input_name": "input.1",
            "input_image_folder": "/data1/100_images"
        }
    ]
```



```
},
"simulator_img_files": [
    {
        "model_input_name": "input.1",
        "input_image": "/data1/100_images/train_00033.jpg"
    }
]
```

STEP 2

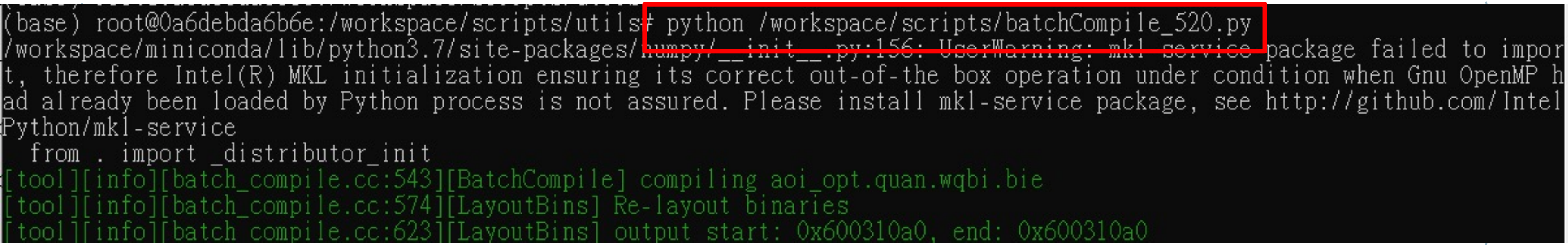
- 執行模型量化(容易使 RAM 不夠，盡量不要開 Google Chrome)



```
(base) root@0a6debda6b6e:/workspace/scripts/utils# python /workspace/scripts/fpAnalyserCompilerIpevaluator_520.py
/workspace/miniconda/lib/python3.7/site-packages/numpy/__init__.py:156: UserWarning: mkl-service package failed to impor
t, therefore Intel(R) MKL initialization ensuring its correct out-of-the box operation under condition when Gnu OpenMP h
ad already been loaded by Python process is not assured. Please install mkl-service package, see http://github.com/Intel
Python/mkl-service
  from . import _distributor_init
customized
customized
customized
```

STEP 2

- 模型編譯

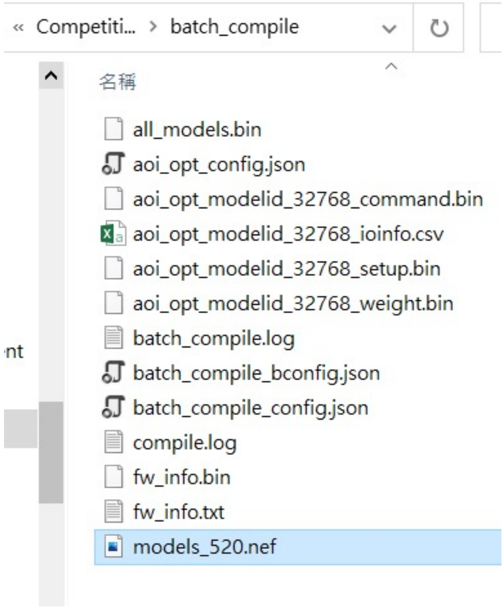


```
(base) root@0a6debda6b6e:/workspace/scripts/utils# python /workspace/scripts/batchCompile_520.py
/workspace/miniconda/lib/python3.7/site-packages/numpy/__init__.py:156: UserWarning: mkl-service package failed to impor
t, therefore Intel(R) MKL initialization ensuring its correct out-of-the box operation under condition when Gnu OpenMP h
ad already been loaded by Python process is not assured. Please install mkl-service package, see http://github.com/Intel
Python/mkl-service
  from . import _distributor_init
[tool][info][batch_compile.cc:543][BatchCompile] compiling aoi_opt.quan.wqbi.bie
[tool][info][batch_compile.cc:574][LayoutBins] Re-layout binaries
[tool][info][batch_compile.cc:623][LayoutBins] output start: 0x600310a0, end: 0x600310a0
```

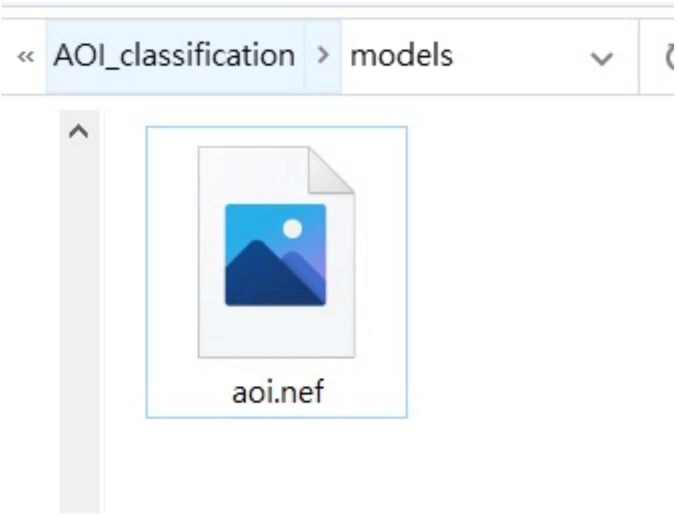
STEP 2

- 將 models_520.nef 移至/c/AOI_classification/models/

  並改名為 aoi.nef



all_models.bin
aoi_opt_config.json
aoi_opt_modelid_32768_command.bin
aoi_opt_modelid_32768_ioinfo.csv
aoi_opt_modelid_32768_setup.bin
aoi_opt_modelid_32768_weight.bin
batch_compile.log
batch_compile_bconfig.json
batch_compile_config.json
compile.log
fw_info.bin
fw_info.txt
models_520.nef

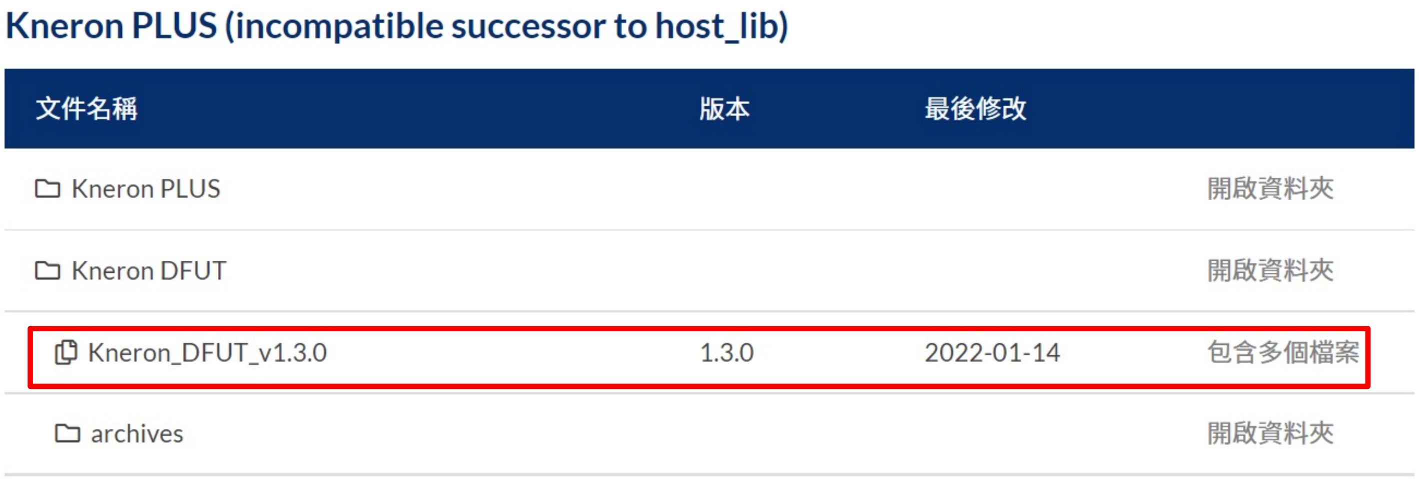


aoi.nef

STEP 3

- 升級耐能棒，下載 Kneron_DFUT

https://www.kneron.com/tw/support/developers/

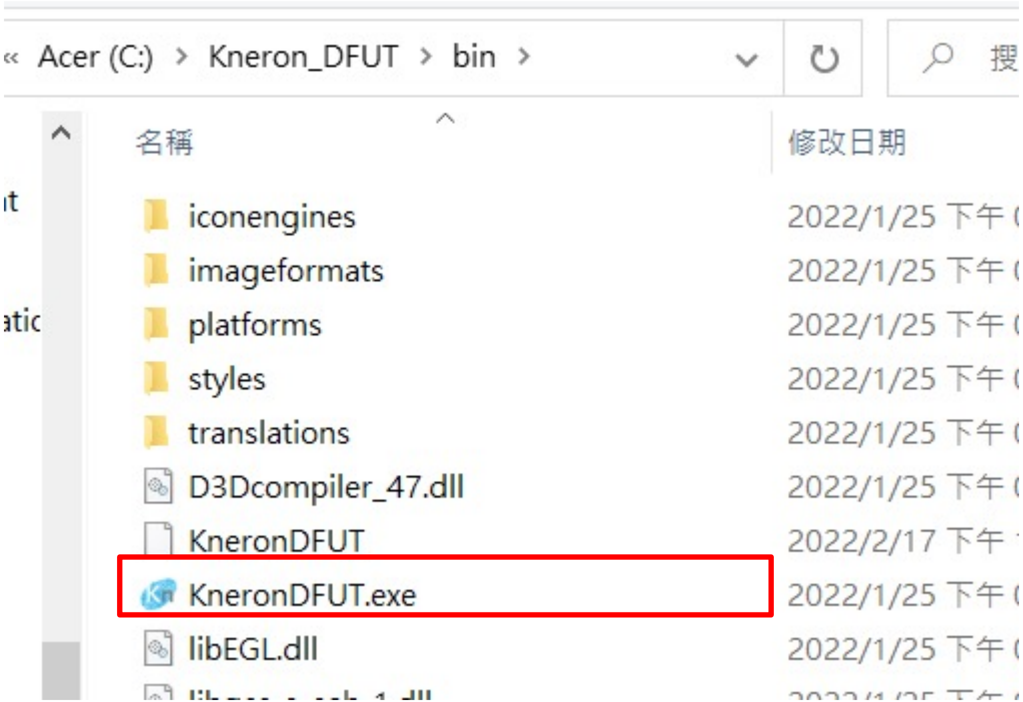


Kneron PLUS (incompatible successor to host_lib)

文件名稱	版本	最後修改	
🗀 Kneron PLUS			開啟資料夾
🗀 Kneron DFUT			開啟資料夾
Kneron_DFUT_v1.3.0	1.3.0	2022-01-14	包含多個檔案
🗀 archives			開啟資料夾

STEP 3

- 點選 KneronDFUT.exe

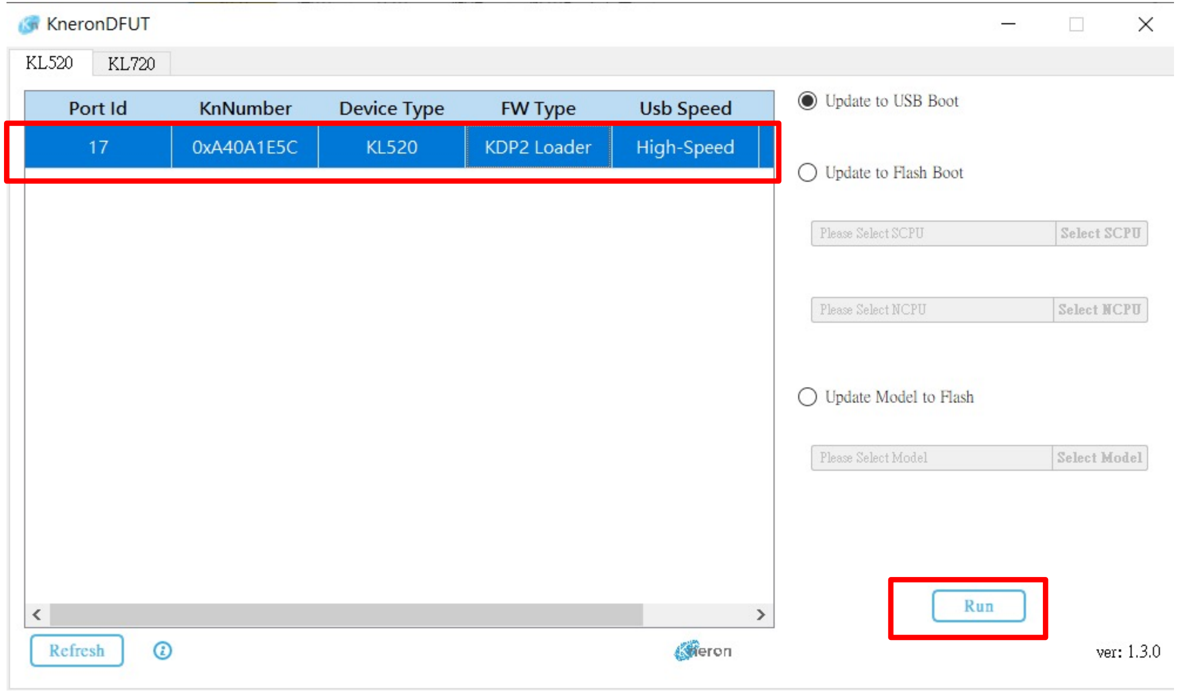


« Acer (C:) › Kneron_DFUT › bin ›

名稱	修改日期
iconengines	2022/1/25 下午
imageformats	2022/1/25 下午
platforms	2022/1/25 下午
styles	2022/1/25 下午
translations	2022/1/25 下午
D3Dcompiler_47.dll	2022/1/25 下午
KneronDFUT	2022/2/17 下午
KneronDFUT.exe	2022/1/25 下午
libEGL.dll	2022/1/25 下午

STEP 3

- 點選 KneronDFUT.exe



Port Id	KnNumber	Device Type	FW Type	Usb Speed
17	0xA40A1E5C	KL520	KDP2 Loader	High-Speed

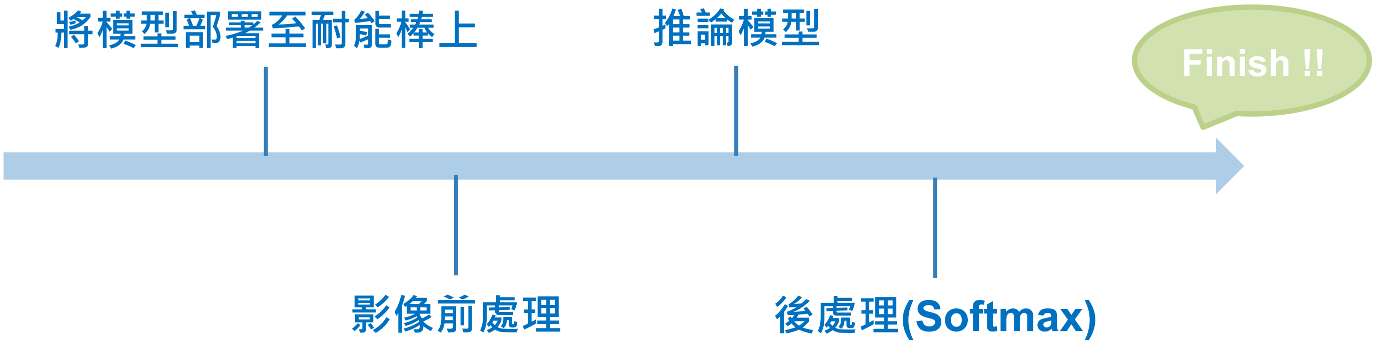
● Update to USB Boot

○ Update to Flash Boot

○ Update Model to Flash

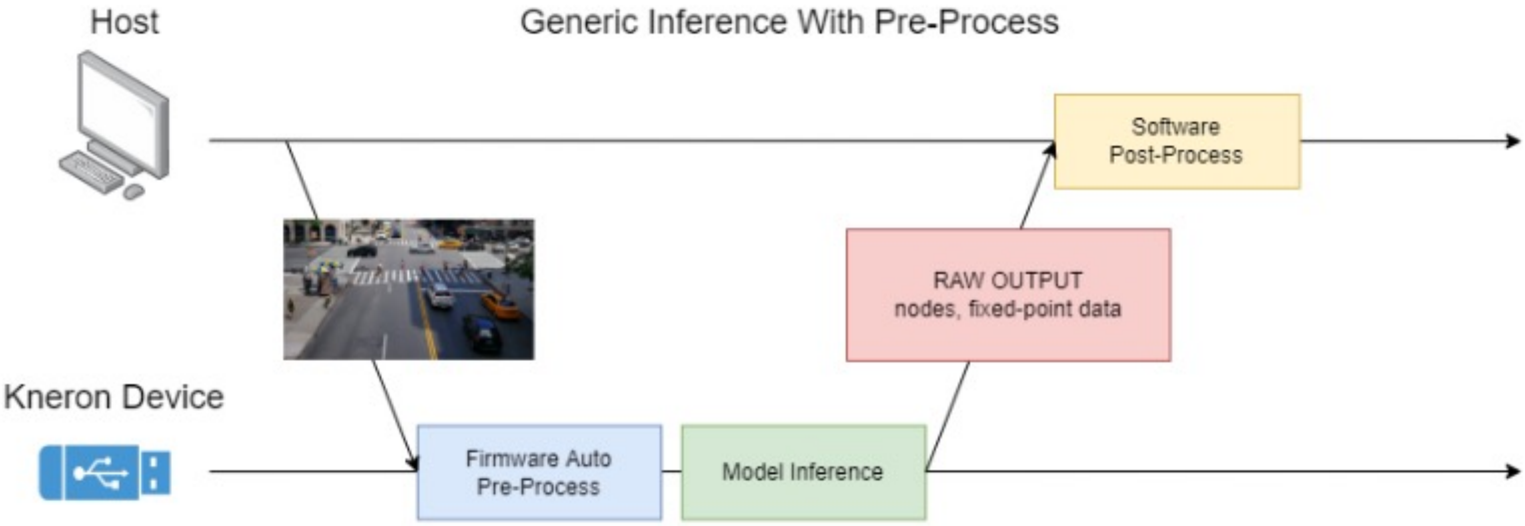
STEP 4

- NPU 推論流程



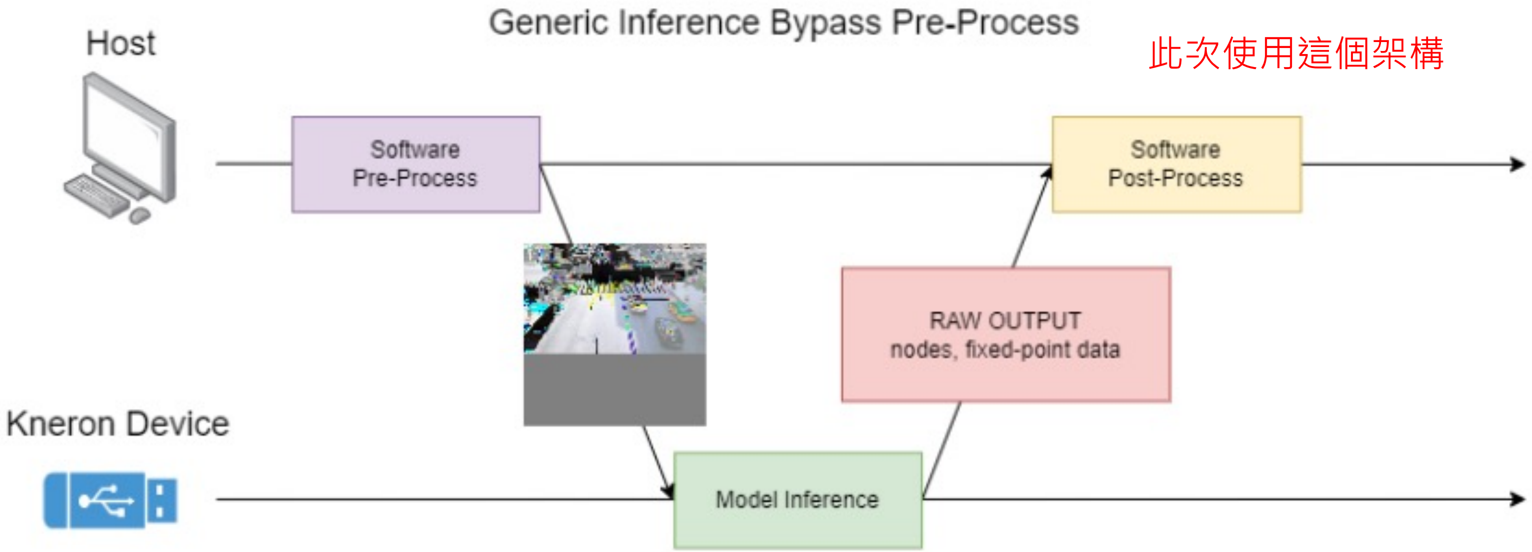
STEP 4

- 可以自由選擇前處理或後處理是否在耐能棒上運行



STEP 4

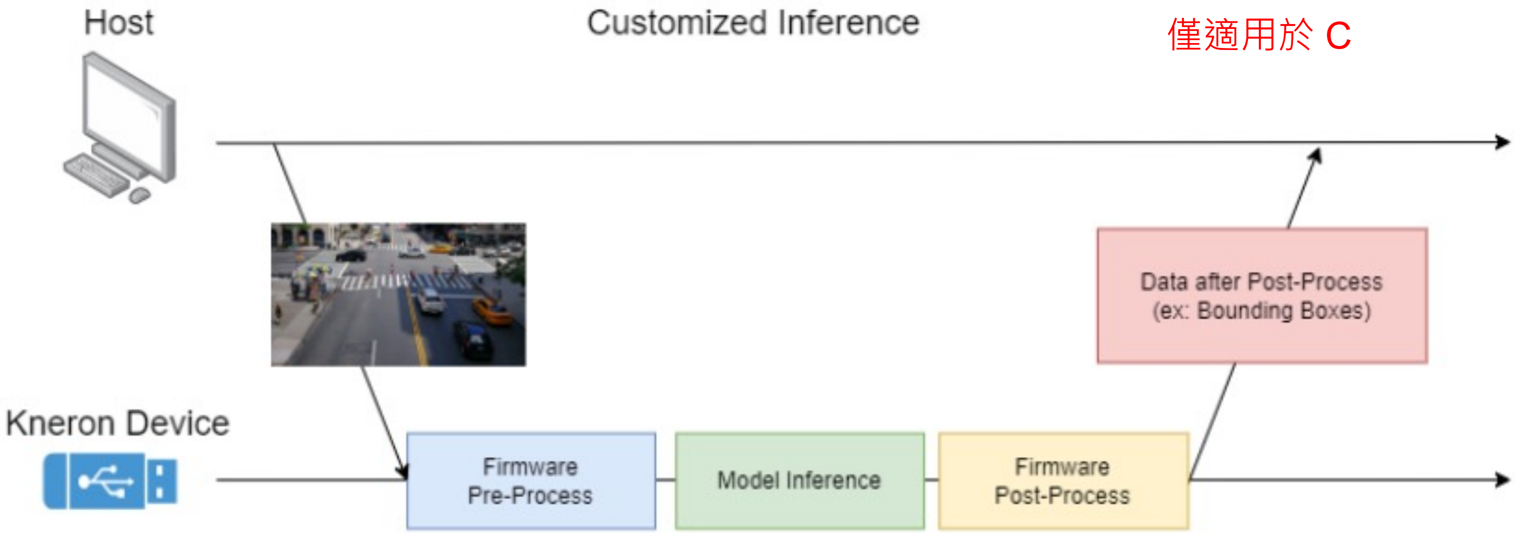
- 可以自由選擇前處理或後處理是否在耐能棒上運行



此次使用這個架構

STEP 4

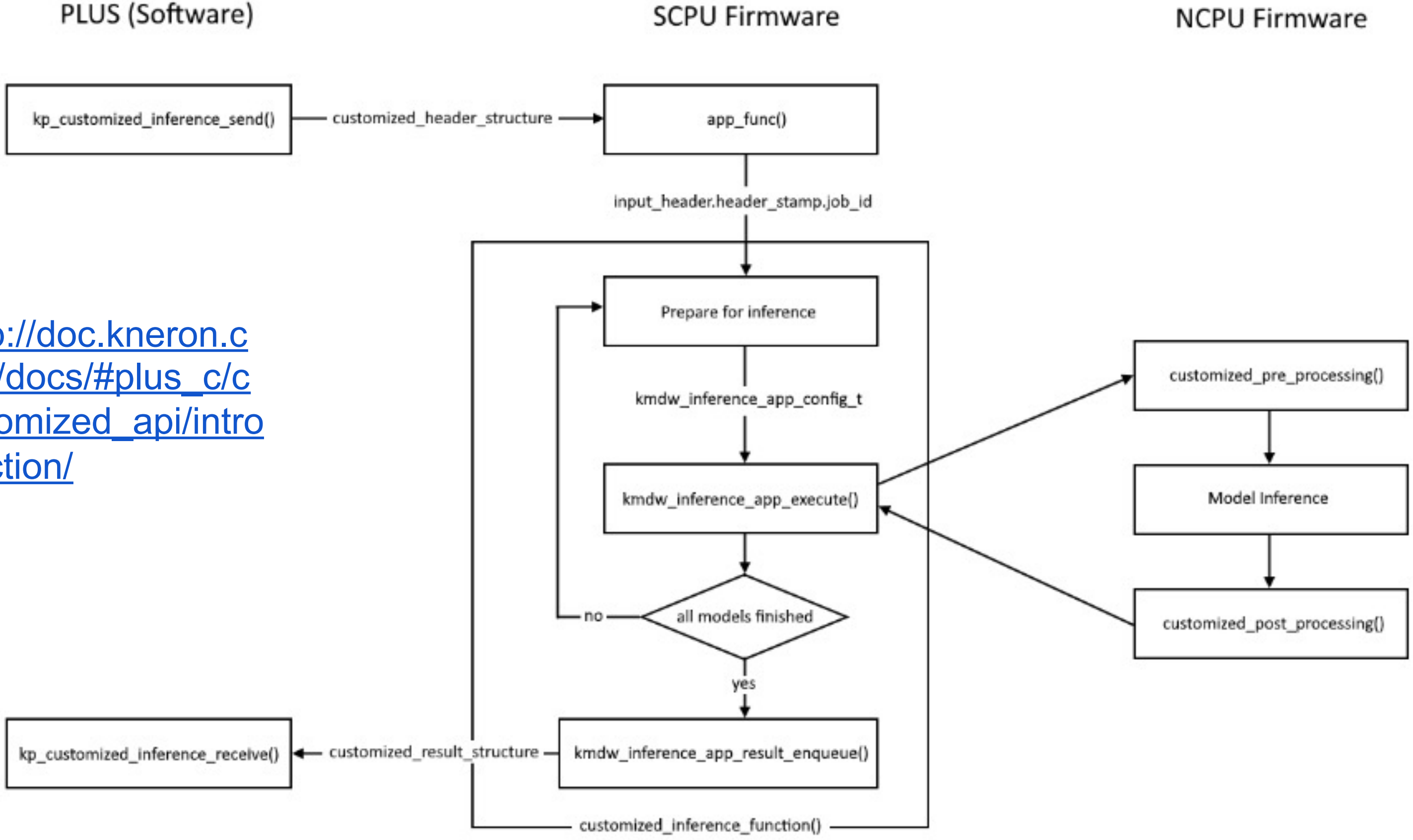
- 可以自由選擇前處理或後處理是否在耐能棒上運行



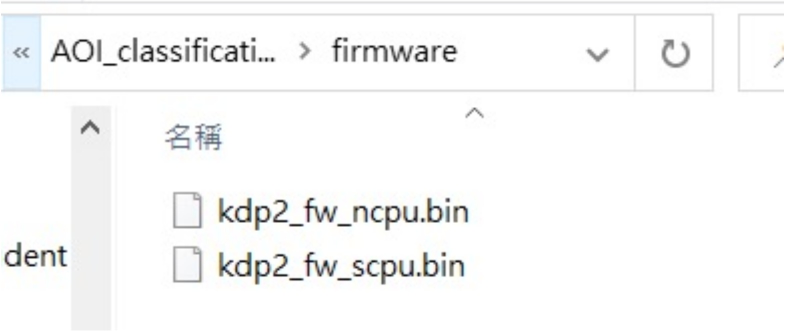
僅適用於 C

STEP 4

http://doc.kneron.com/docs/#plus_c/customized_api/introduction/


PLUS (Software)
SCPU Firmware
NCPU Firmware
kp_customized_inference_send()
customized_header_structure
app_func()
input_header.header_stamp.job_id
Prepare for inference
kmdw_inference_app_config_t
kmdw_inference_app_execute()
all models finished
no
yes
kmdw_inference_app_result_enqueue()
customized_result_structure
kp_customized_inference_receive()
customized_inference_function()
customized_pre_processing()
Model Inference
customized_post_processing()

STEP 4



« AOI_classificati... › firmware

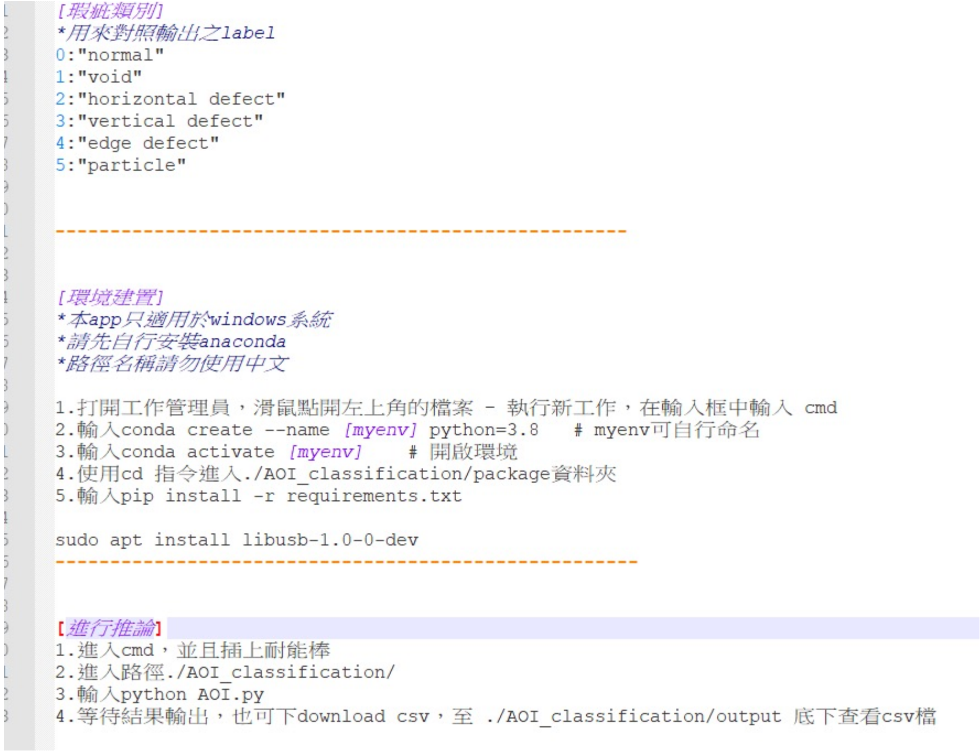
名稱

kdp2_fw_ncpu.bin

kdp2_fw_scpu.bin

STEP 4

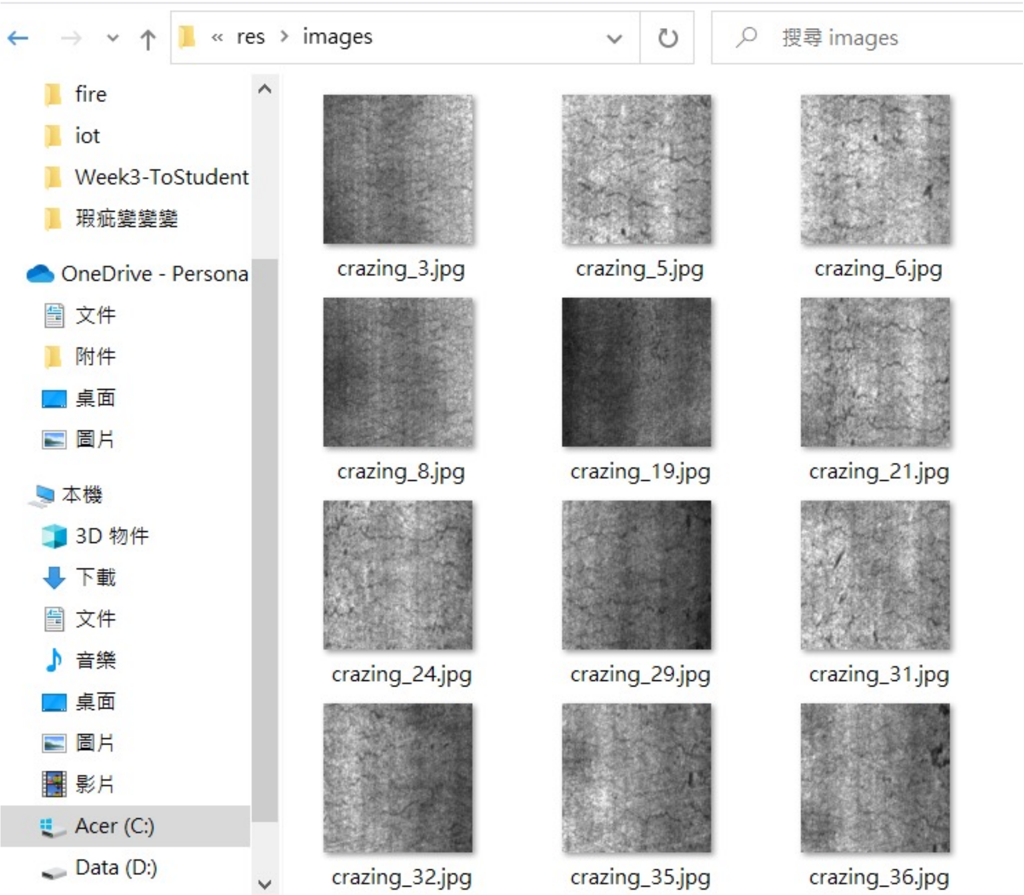
- 點開 AOI_classification，可以查看 README，並照著環境建置



```
[瑕疵類別]
*用來對照輸出之label
0:"normal"
1:"void"
2:"horizontal defect"
3:"vertical defect"
4:"edge defect"
5:"particle"


----------------------------------------------------


[環境建置]
*本app只適用於windows系統
*請先自行安裝anaconda
*路徑名稱請勿使用中文

1.打開工作管理員，滑鼠點開左上角的檔案 - 執行新工作，在輸入框中輸入 cmd
2.輸入conda create --name [myenv] python=3.8   # myenv可自行命名
3.輸入conda activate [myenv]    # 開啟環境
4.使用cd 指令進入./AOI_classification/package資料夾
5.輸入pip install -r requirements.txt

sudo apt install libusb-1.0-0-dev
-----------------------------------------------------


[進行推論]
1.進入cmd，並且插上耐能棒
2.進入路徑./AOI_classification/
3.輸入python AOI.py
4.等待結果輸出，也可下download csv，至 ./AOI_classification/output 底下查看csv檔
```

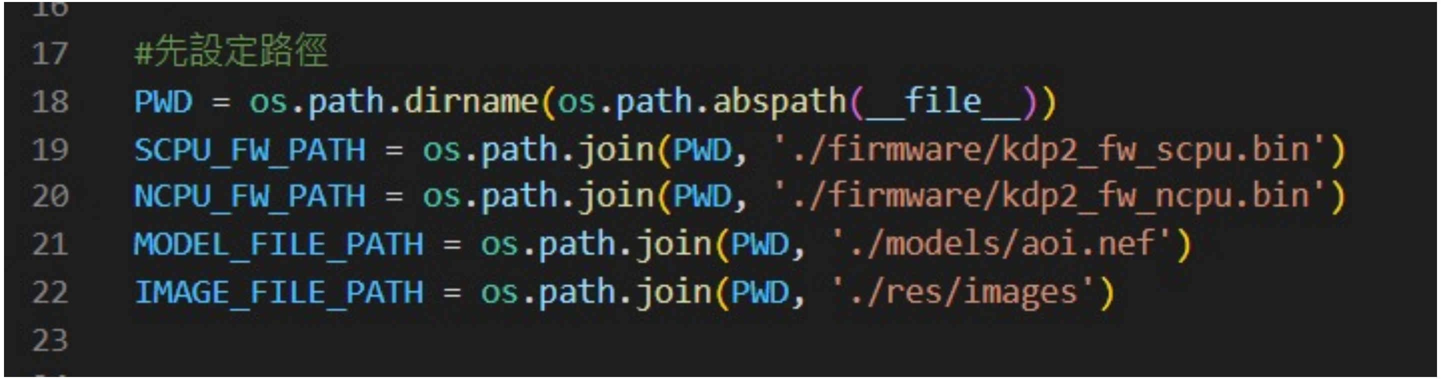
STEP 4

測試圖片已放置
(勿動)



STEP 4

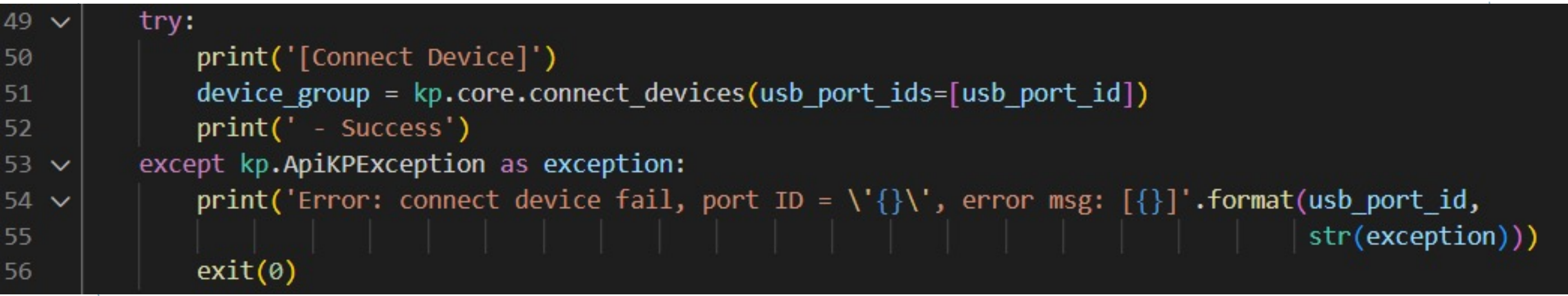
- 先設定路徑



```
#先設定路徑
PWD = os.path.dirname(os.path.abspath(__file__))
SCPU_FW_PATH = os.path.join(PWD, './firmware/kdp2_fw_scpu.bin')
NCPU_FW_PATH = os.path.join(PWD, './firmware/kdp2_fw_ncpu.bin')
MODEL_FILE_PATH = os.path.join(PWD, './models/aoi.nef')
IMAGE_FILE_PATH = os.path.join(PWD, './res/images')
```

STEP 4

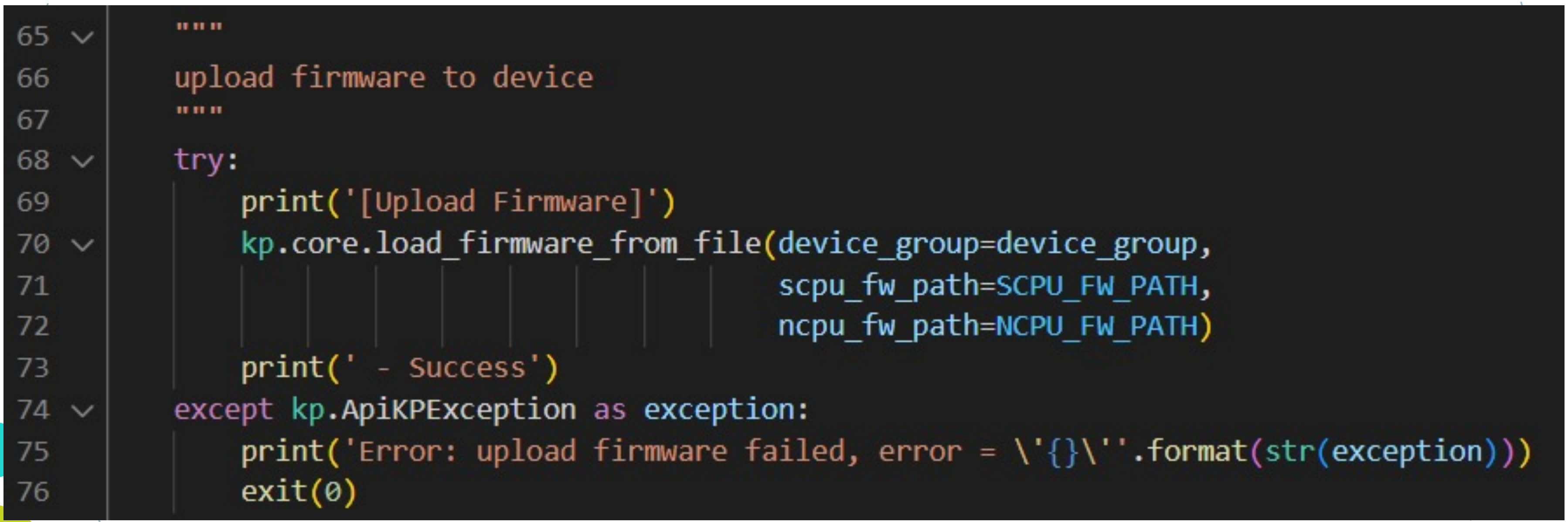
- 連接耐能棒



```python
    try:
        print('[Connect Device]')
        device_group = kp.core.connect_devices(usb_port_ids=[usb_port_id])
        print(' - Success')
    except kp.ApiKPException as exception:
        print('Error: connect device fail, port ID = \'{}\', error msg: [{}]'.format(usb_port_id,
                                                                                     str(exception)))
        exit(0)
```

STEP 4

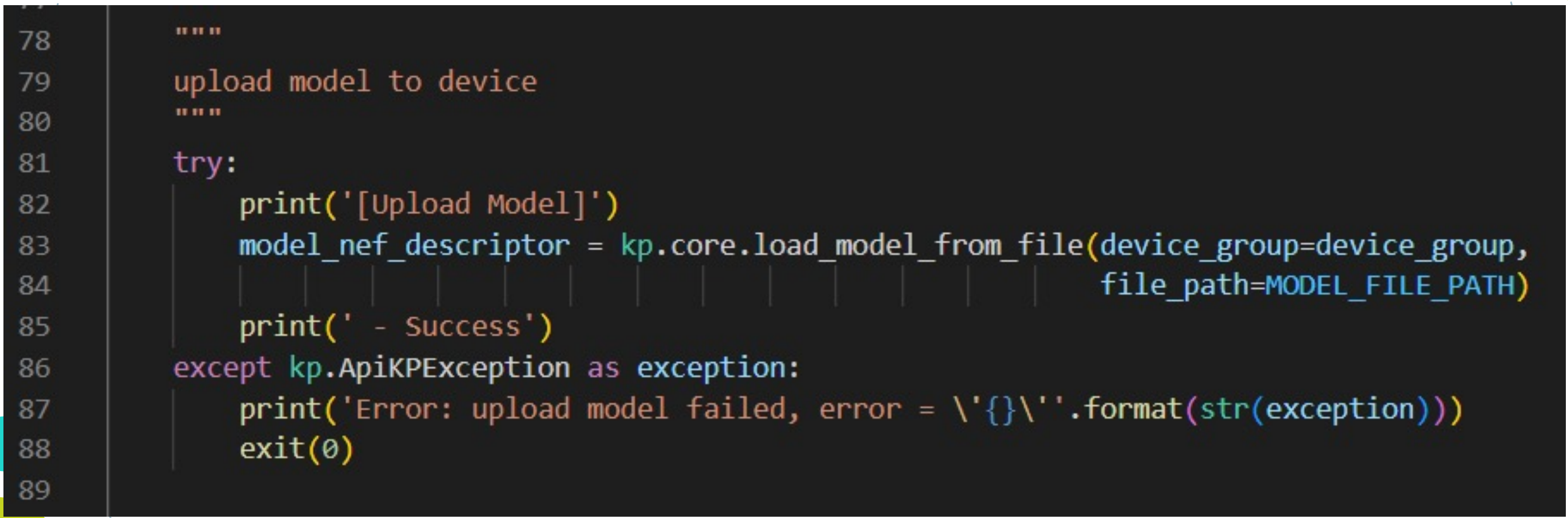
- 將 firmware 上傳至耐能棒



```python
    """
    upload firmware to device
    """
    try:
        print('[Upload Firmware]')
        kp.core.load_firmware_from_file(device_group=device_group,
                                        scpu_fw_path=SCPU_FW_PATH,
                                        ncpu_fw_path=NCPU_FW_PATH)
        print(' - Success')
    except kp.ApiKPException as exception:
        print('Error: upload firmware failed, error = \'{}\''.format(str(exception)))
        exit(0)
```

STEP 4

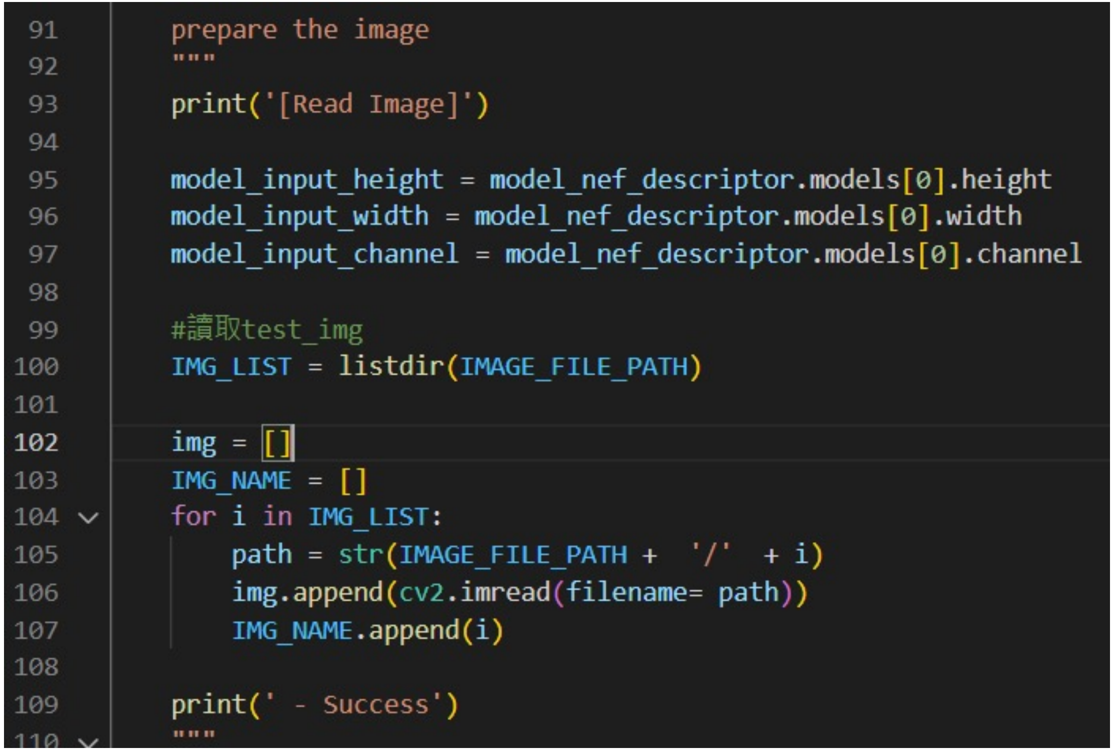
- 將 NEF model 上傳至耐能棒



```python
    """
    upload model to device
    """
    try:
        print('[Upload Model]')
        model_nef_descriptor = kp.core.load_model_from_file(device_group=device_group,
                                                            file_path=MODEL_FILE_PATH)
        print(' - Success')
    except kp.ApiKPException as exception:
        print('Error: upload model failed, error = \'{}\''.format(str(exception)))
        exit(0)

```

STEP 4

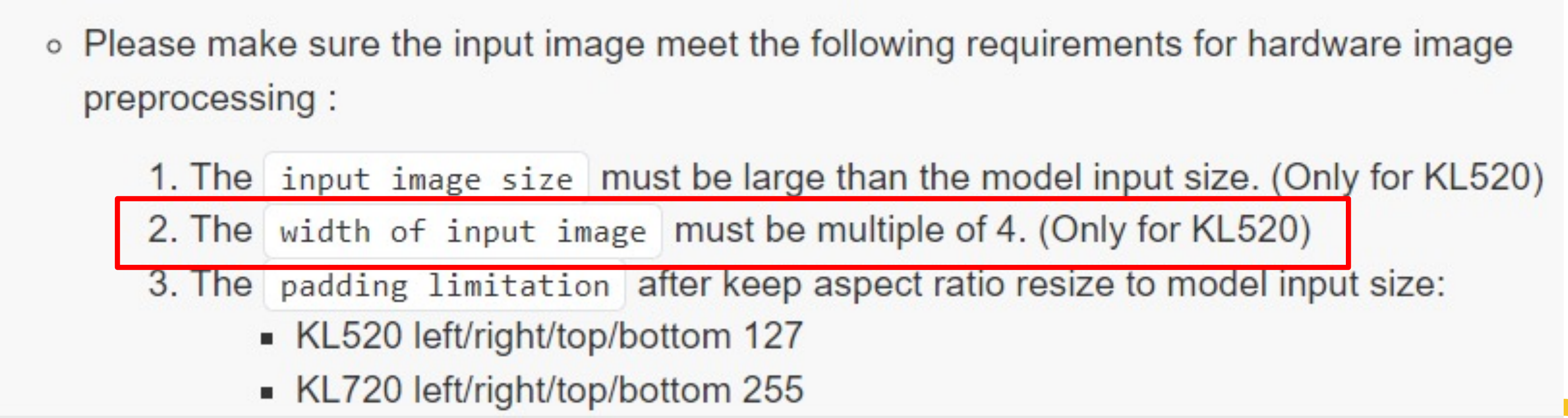
- 讀取圖片



```
prepare the image
"""
print('[Read Image]')

model_input_height = model_nef_descriptor.models[0].height
model_input_width = model_nef_descriptor.models[0].width
model_input_channel = model_nef_descriptor.models[0].channel

#讀取test_img
IMG_LIST = listdir(IMAGE_FILE_PATH)

img = []
IMG_NAME = []
for i in IMG_LIST:
    path = str(IMAGE_FILE_PATH +  '/'  + i)
    img.append(cv2.imread(filename= path))
    IMG_NAME.append(i)

print(' - Success')
"""
```

STEP 4

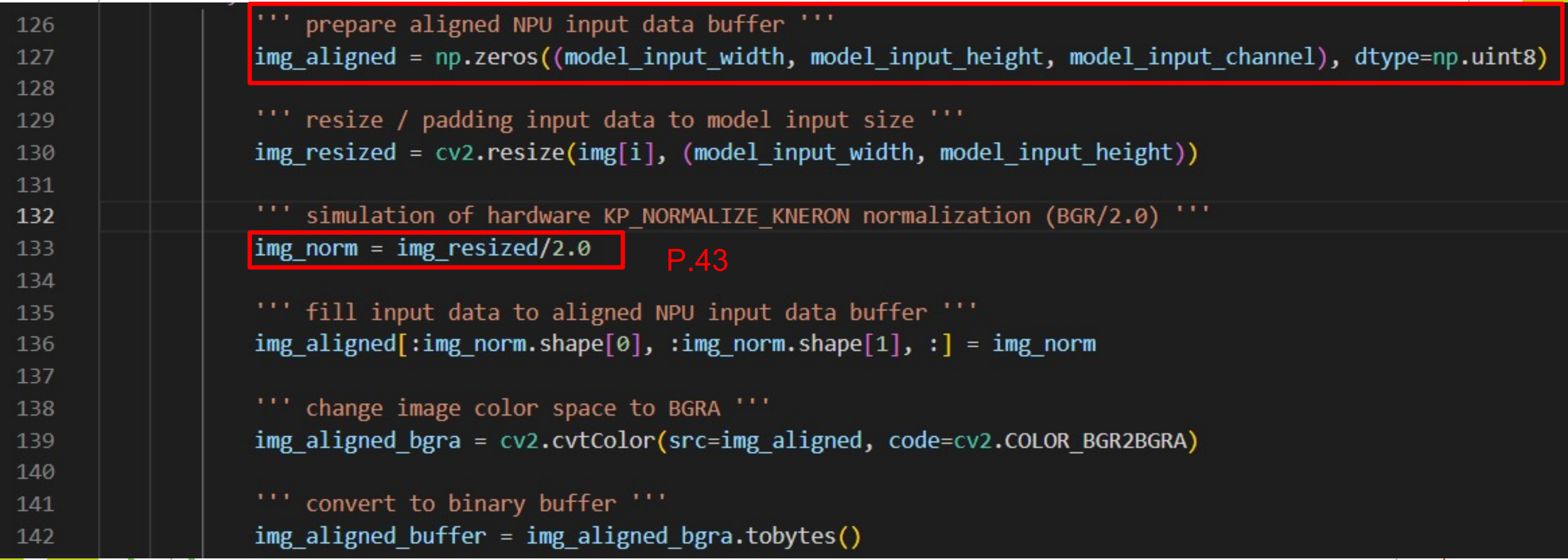
- Kneron PLUS 支持BGR565, BGRA8888, RAW8 (Grayscale)



- Please make sure the input image meet the following requirements for hardware image preprocessing :
  1. The `input image size` must be large than the model input size. (Only for KL520)
  2. The `width of input image` must be multiple of 4. (Only for KL520)
  3. The `padding limitation` after keep aspect ratio resize to model input size:
     - KL520 left/right/top/bottom 127
     - KL720 left/right/top/bottom 255

STEP 4

- 影像前處理



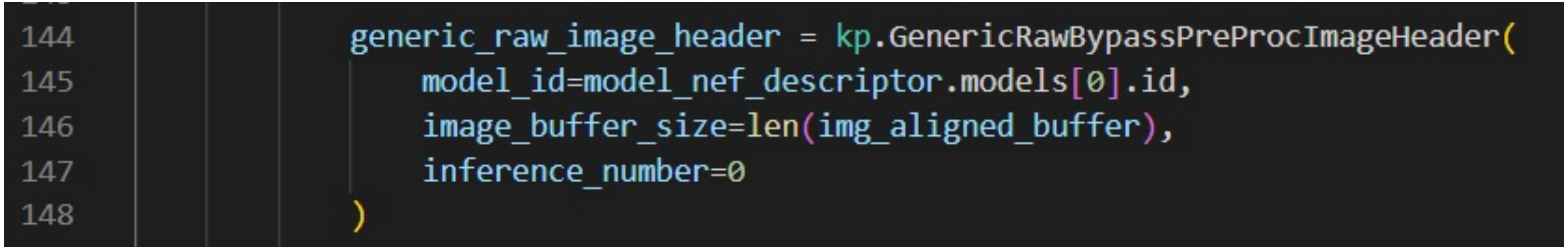
用來放推論圖片的buffer

```python
            ''' prepare aligned NPU input data buffer '''
            img_aligned = np.zeros((model_input_width, model_input_height, model_input_channel), dtype=np.uint8)

            ''' resize / padding input data to model input size '''
            img_resized = cv2.resize(img[i], (model_input_width, model_input_height))

            ''' simulation of hardware KP_NORMALIZE_KNERON normalization (BGR/2.0) '''
            img_norm = img_resized/2.0

            ''' fill input data to aligned NPU input data buffer '''
            img_aligned[:img_norm.shape[0], :img_norm.shape[1], :] = img_norm

            ''' change image color space to BGRA '''
            img_aligned_bgra = cv2.cvtColor(src=img_aligned, code=cv2.COLOR_BGR2BGRA)

            ''' convert to binary buffer '''
            img_aligned_buffer = img_aligned_bgra.tobytes()
```

P.43

STEP 4

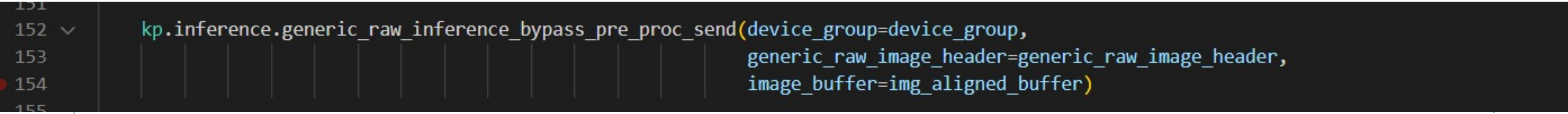
- 準備 generic inference 的配置



```
            generic_raw_image_header = kp.GenericRawBypassPreProcImageHeader(
                model_id=model_nef_descriptor.models[0].id,
                image_buffer_size=len(img_aligned_buffer),
                inference_number=0
            )
```

STEP 4

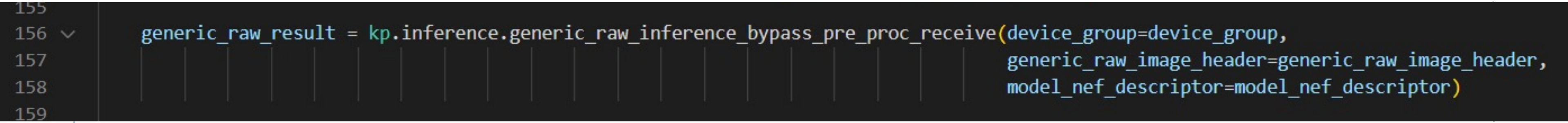
- Send image to connected Kneron devices for inference



```
    kp.inference.generic_raw_inference_bypass_pre_proc_send(device_group=device_group,
                                                            generic_raw_image_header=generic_raw_image_header,
                                                            image_buffer=img_aligned_buffer)
```

STEP 4

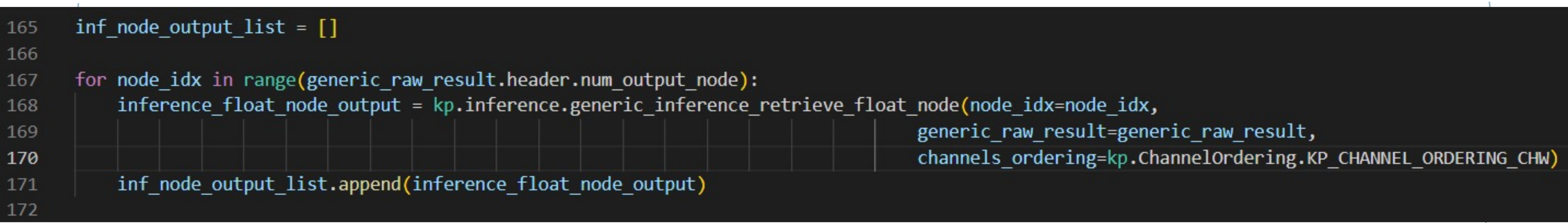
- Receive inference raw result from connected Kneron devices



```python
    generic_raw_result = kp.inference.generic_raw_inference_bypass_pre_proc_receive(device_group=device_group,
                                                                                    generic_raw_image_header=generic_raw_image_header,
                                                                                    model_nef_descriptor=model_nef_descriptor)
```

STEP 4

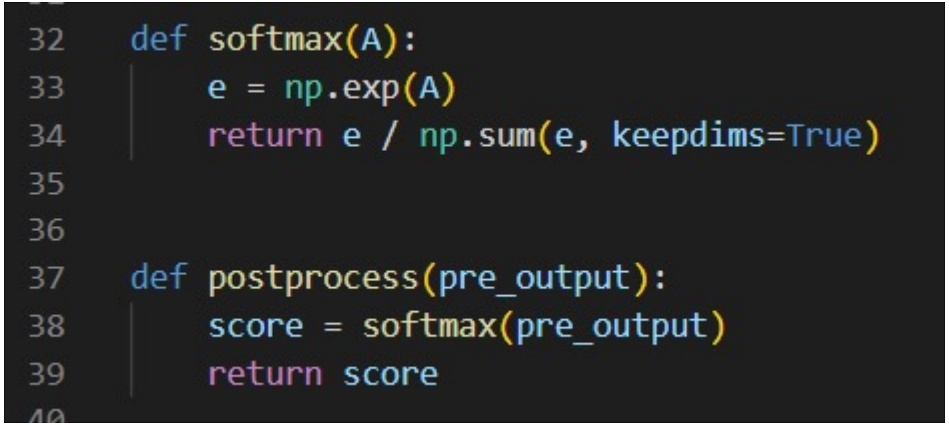
- Retrieve inference node output with floating-point mode



```python
inf_node_output_list = []

for node_idx in range(generic_raw_result.header.num_output_node):
    inference_float_node_output = kp.inference.generic_inference_retrieve_float_node(node_idx=node_idx,
                                                                                generic_raw_result=generic_raw_result,
                                                                                channels_ordering=kp.ChannelOrdering.KP_CHANNEL_ORDERING_CHW)
    inf_node_output_list.append(inference_float_node_output)

```

STEP 4

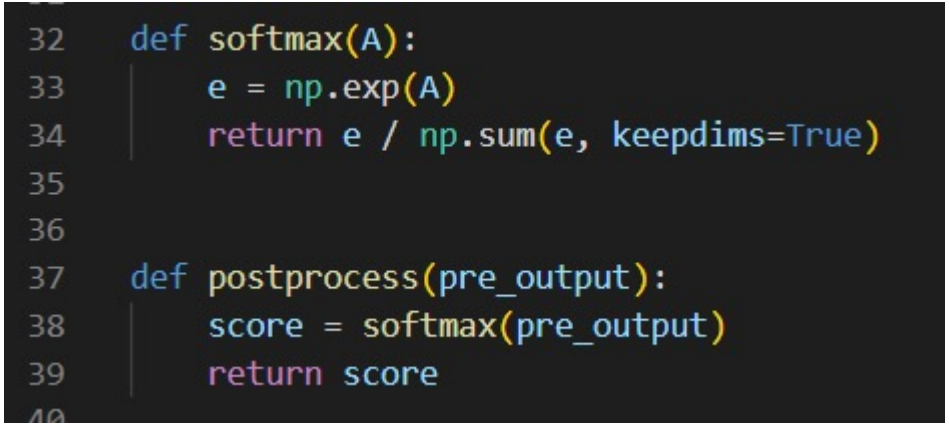
- 後處理



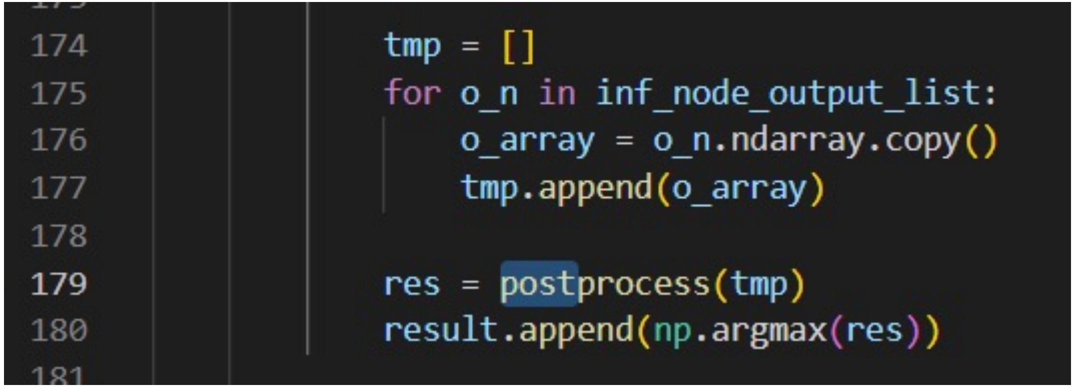
```python
def softmax(A):
    e = np.exp(A)
    return e / np.sum(e, keepdims=True)


def postprocess(pre_output):
    score = softmax(pre_output)
    return score
```

STEP 4

- 後處理



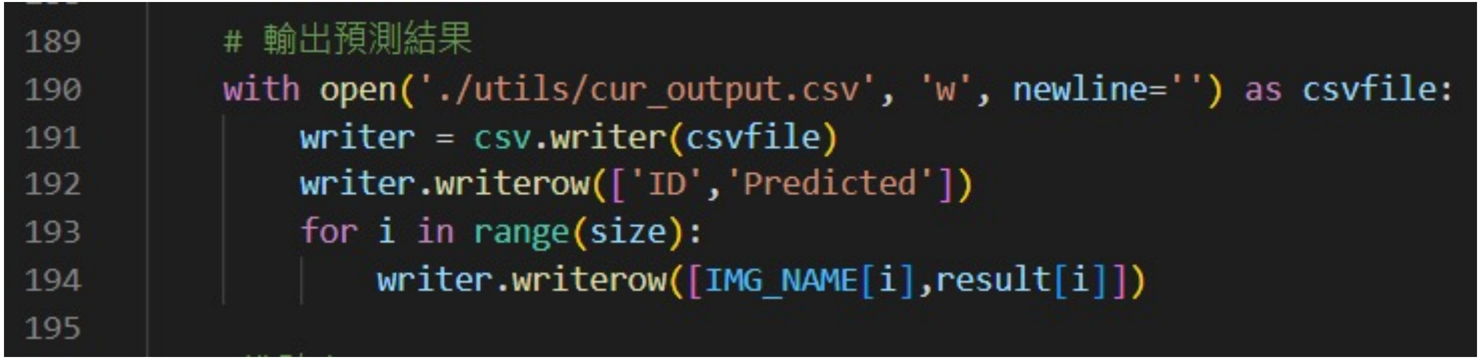
```python
def softmax(A):
    e = np.exp(A)
    return e / np.sum(e, keepdims=True)


def postprocess(pre_output):
    score = softmax(pre_output)
    return score
```



```python
            tmp = []
            for o_n in inf_node_output_list:
                o_array = o_n.ndarray.copy()
                tmp.append(o_array)

            res = postprocess(tmp)
            result.append(np.argmax(res))
```

STEP 4

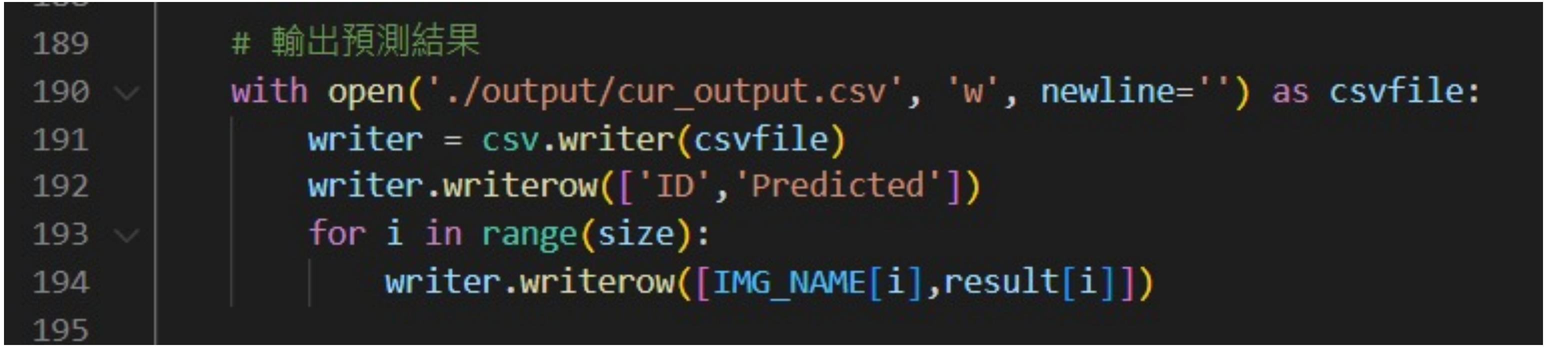
- 輸出預測結果



```python
# 輸出預測結果
with open('./utils/cur_output.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['ID','Predicted'])
    for i in range(size):
        writer.writerow([IMG_NAME[i],result[i]])

```

STEP 4

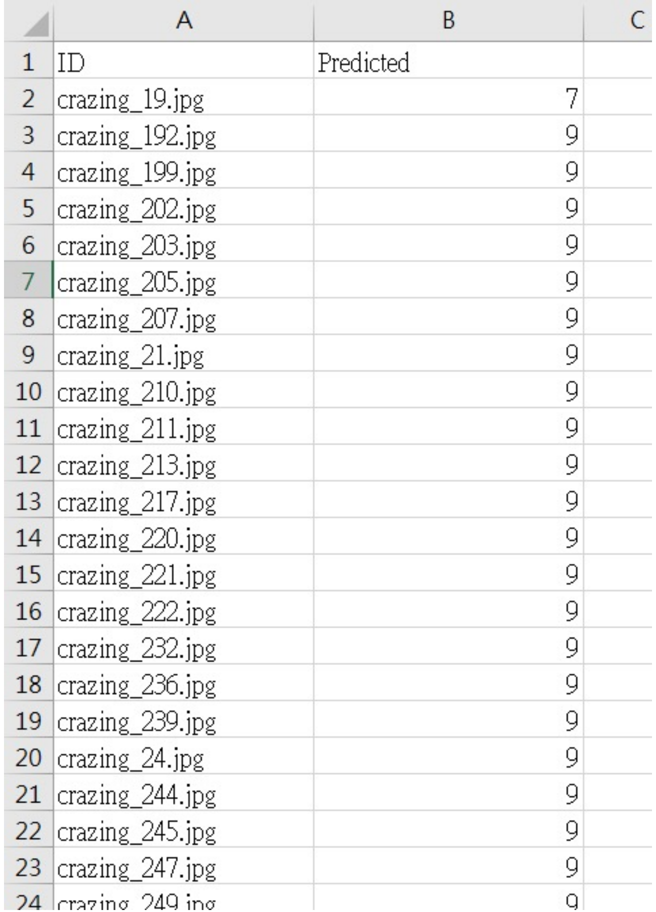
- 輸出預測結果



```
# 輸出預測結果
with open('./output/cur_output.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['ID','Predicted'])
    for i in range(size):
        writer.writerow([IMG_NAME[i],result[i]])

```

STEP 4

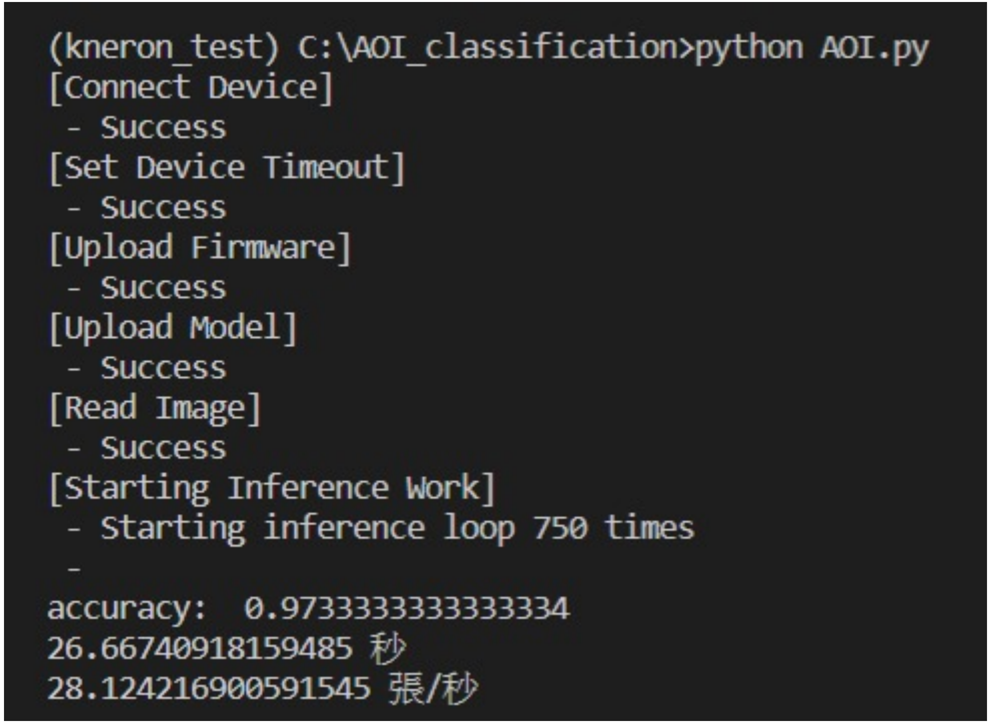
- 輸出預測結果



	A	B	C
1	ID	Predicted	
2	crazing_19.jpg	7	
3	crazing_192.jpg	9	
4	crazing_199.jpg	9	
5	crazing_202.jpg	9	
6	crazing_203.jpg	9	
7	crazing_205.jpg	9	
8	crazing_207.jpg	9	
9	crazing_21.jpg	9	
10	crazing_210.jpg	9	
11	crazing_211.jpg	9	
12	crazing_213.jpg	9	
13	crazing_217.jpg	9	
14	crazing_220.jpg	9	
15	crazing_221.jpg	9	
16	crazing_222.jpg	9	
17	crazing_232.jpg	9	
18	crazing_236.jpg	9	
19	crazing_239.jpg	9	
20	crazing_24.jpg	9	
21	crazing_244.jpg	9	
22	crazing_245.jpg	9	
23	crazing_247.jpg	9	
24	crazing_249.jpg	9	

STEP 4

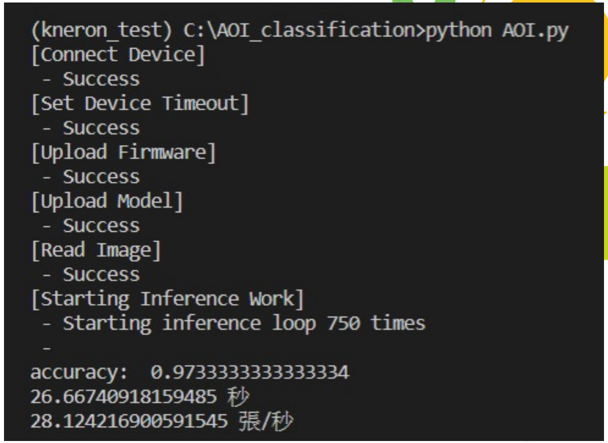
- Finish



```
(kneron_test) C:\AOI_classification>python AOI.py
[Connect Device]
 - Success
[Set Device Timeout]
 - Success
[Upload Firmware]
 - Success
[Upload Model]
 - Success
[Read Image]
 - Success
[Starting Inference Work]
 - Starting inference loop 750 times
 -
accuracy:  0.9733333333333334
26.66740918159485 秒
28.124216900591545 張/秒
```

HW

- 兩人一組完成這份作業，將模型部屬於耐能棒
- 截圖部署成功後，進行inference，如右圖
- 截圖檔名為你與同組同學的學號 (sXXX_sXXX)
- 上傳截圖與預測結果.csv
- Deadline: 2022/6/21 23:59



```
(kneron_test) C:\AOI_classification>python AOI.py
[Connect Device]
 - Success
[Set Device Timeout]
 - Success
[Upload Firmware]
 - Success
[Upload Model]
 - Success
[Read Image]
 - Success
[Starting Inference Work]
 - Starting inference loop 750 times
 -
accuracy:  0.9733333333333334
26.66740918159485 秒
28.124216900591545 張/秒
```

Thanks!

Any questions?

You can find me at @username & user@mail.me