嵌入式智慧影像分析與實境界面
Fall 2021

Instructor：Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering

National Taipei University of Technology

Project 8

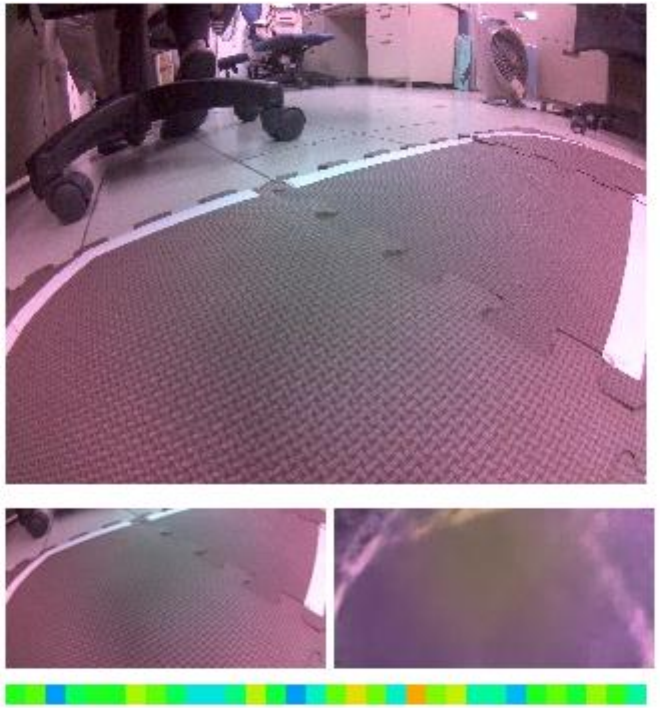
Deep Q-Learning的道路辨識

前言

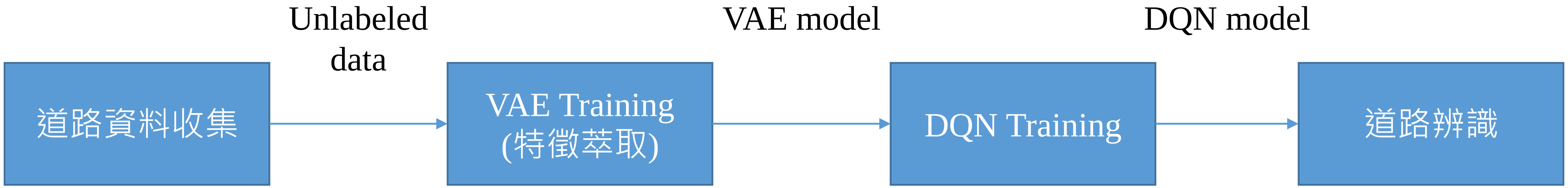
- 多數自駕模型車如Jetbot或是JetRacer進行road following都是使用supervised-learning。但是此方式需要大量資料並且耗費大量人力資源。
- 使用deep reinforcement learning (DRL)可以在行進間與環境資料進行比對調整行為，不需要使用人工標記的數據。
- 使用SAC跟VAE可以使Jetbot在短時間內快速學習，完成road following。

VAE Model

- 左邊是實際影像，右邊是VAE model進行重構後的影像
- 底下的顏色條顯示出VAE的latent variable(z=32 dim)
- 使用VAE model可以更好的得到特徵



道路辨識流程圖



道路資料收集

Unlabeled data

VAE Training (特徵萃取)

VAE model

DQN Training

DQN model

道路辨識

環境安裝

更新ubuntu

- 更新軟體的最新資訊及列表

```
$ sudo apt-get update
```

- 更新目前已安裝的軟體到最新版本

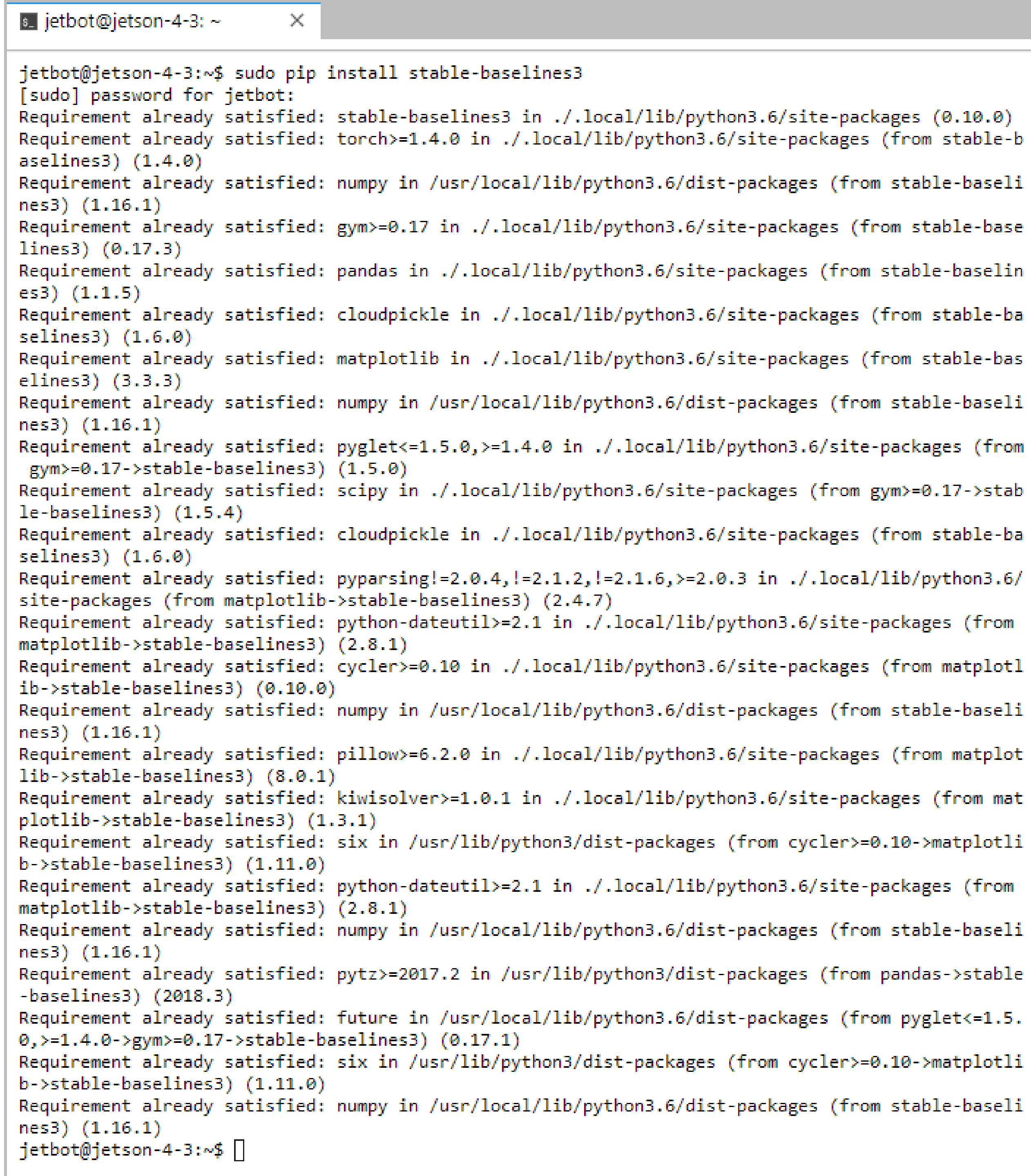
```
$ sudo apt-get upgrade
```

LearningRacer-rl

- $ cd ~/ && git clone https://github.com/masato-ka/airc-rl-agent.git
- $ cd airc-rl-agent
- $ git checkout tags/release-1.5.0
- $ sh install_jetpack.sh
- $ sudo python3 setup.py install

安裝stable-baselines3

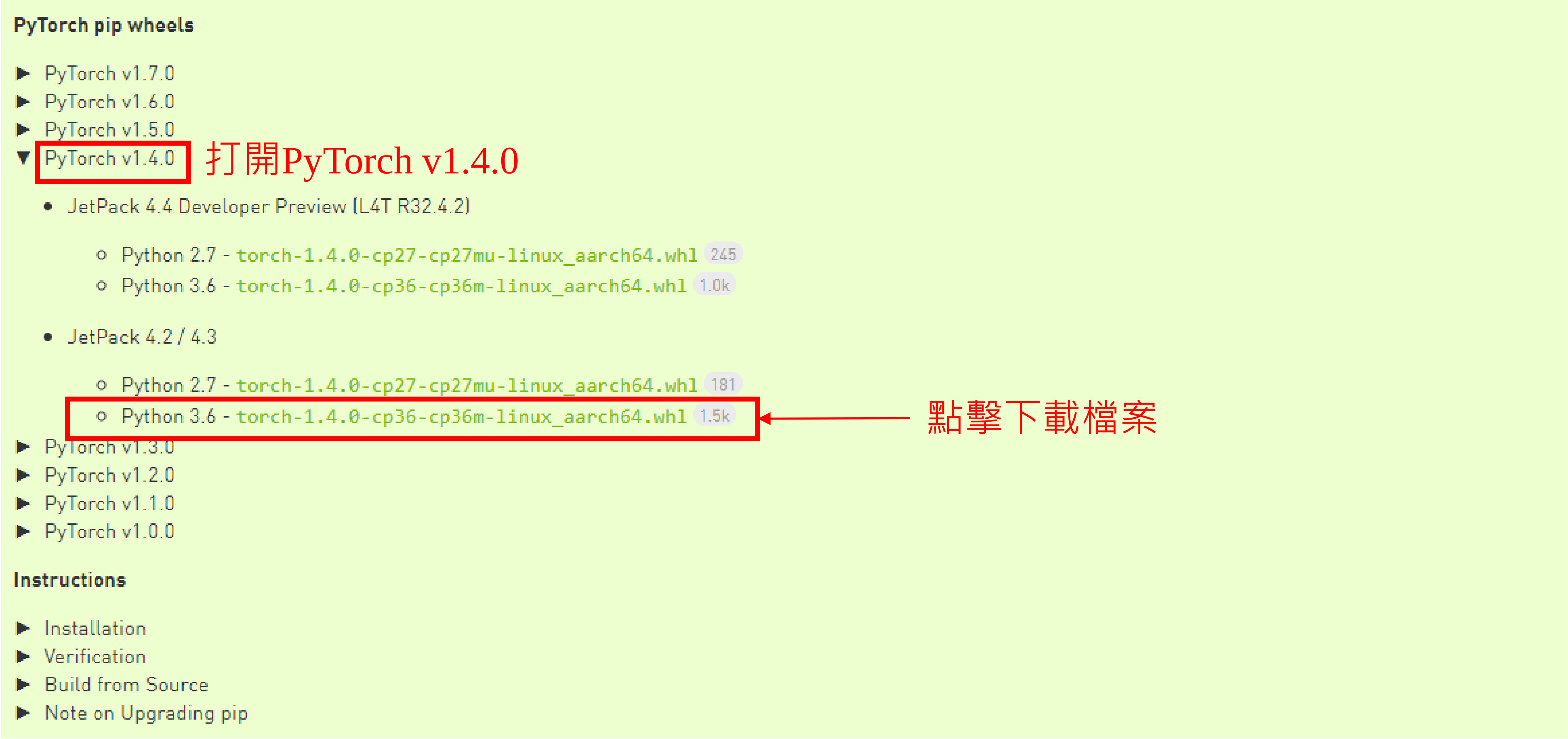
- $ sudo pip uninstall stable-baselines3
- $ sudo pip install stable-baselines3==0.10.0
- 此演算法是基於OpenAI的增強式學習演算法，裡面包含SAC深度的增強式學習系統，且有callback 功能利於實踐Deep Q-Learning。



```
jetbot@jetson-4-3:~$ sudo pip install stable-baselines3
[sudo] password for jetbot:
Requirement already satisfied: stable-baselines3 in ./.local/lib/python3.6/site-packages (0.10.0)
Requirement already satisfied: torch>=1.4.0 in ./.local/lib/python3.6/site-packages (from stable-b
aselines3) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from stable-baseli
nes3) (1.16.1)
Requirement already satisfied: gym>=0.17 in ./.local/lib/python3.6/site-packages (from stable-base
lines3) (0.17.3)
Requirement already satisfied: pandas in ./.local/lib/python3.6/site-packages (from stable-baselin
es3) (1.1.5)
Requirement already satisfied: cloudpickle in ./.local/lib/python3.6/site-packages (from stable-ba
selines3) (1.6.0)
Requirement already satisfied: matplotlib in ./.local/lib/python3.6/site-packages (from stable-bas
elines3) (3.3.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from stable-baseli
nes3) (1.16.1)
Requirement already satisfied: pyglet<=1.5.0,>=1.4.0 in ./.local/lib/python3.6/site-packages (from
 gym>=0.17->stable-baselines3) (1.5.0)
Requirement already satisfied: scipy in ./.local/lib/python3.6/site-packages (from gym>=0.17->stab
le-baselines3) (1.5.4)
Requirement already satisfied: cloudpickle in ./.local/lib/python3.6/site-packages (from stable-ba
selines3) (1.6.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in ./.local/lib/python3.6/
site-packages (from matplotlib->stable-baselines3) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in ./.local/lib/python3.6/site-packages (from
matplotlib->stable-baselines3) (2.8.1)
Requirement already satisfied: cycler>=0.10 in ./.local/lib/python3.6/site-packages (from matplotl
ib->stable-baselines3) (0.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from stable-baseli
nes3) (1.16.1)
Requirement already satisfied: pillow>=6.2.0 in ./.local/lib/python3.6/site-packages (from matplot
lib->stable-baselines3) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in ./.local/lib/python3.6/site-packages (from mat
plotlib->stable-baselines3) (1.3.1)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from cycler>=0.10->matplotli
b->stable-baselines3) (1.11.0)
Requirement already satisfied: python-dateutil>=2.1 in ./.local/lib/python3.6/site-packages (from
matplotlib->stable-baselines3) (2.8.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from stable-baseli
nes3) (1.16.1)
Requirement already satisfied: pytz>=2017.2 in /usr/lib/python3/dist-packages (from pandas->stable
-baselines3) (2018.3)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from pyglet<=1.5.
0,>=1.4.0->gym>=0.17->stable-baselines3) (0.17.1)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from cycler>=0.10->matplotli
b->stable-baselines3) (1.11.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from stable-baseli
nes3) (1.16.1)
jetbot@jetson-4-3:~$
```

安裝Pytorch 1.4.0

下載網址：https://forums.developer.nvidia.com/t/pytorch-for-jetson-version-1-7-0-now-available/72048



PyTorch pip wheels

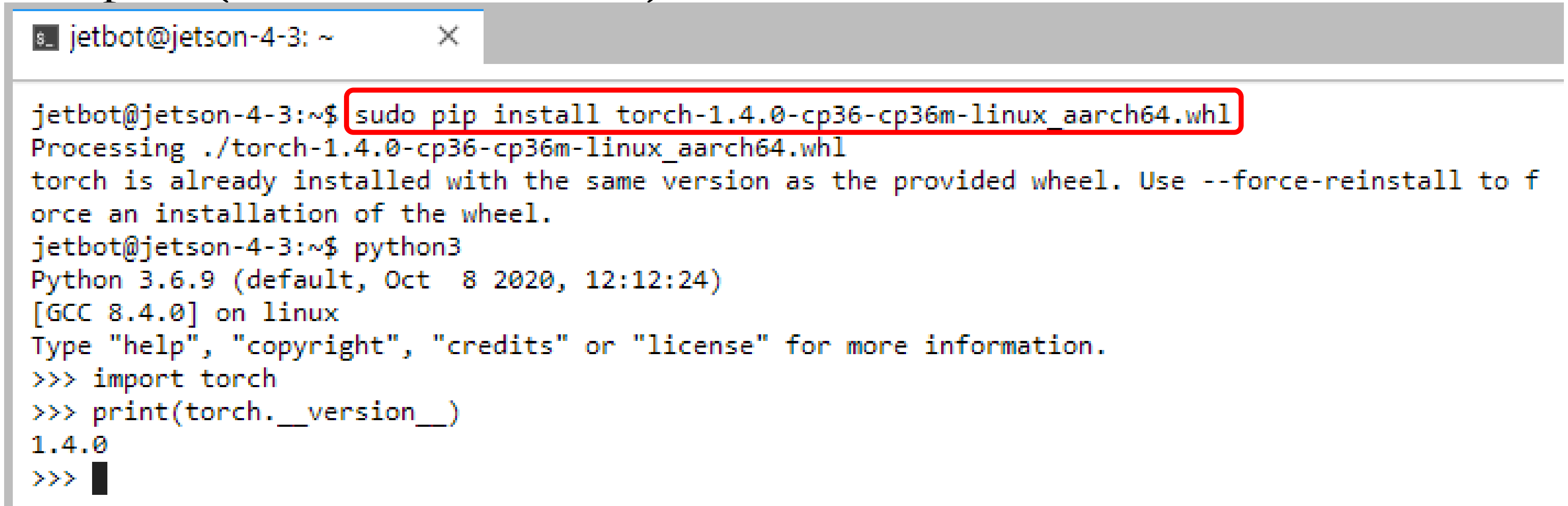
- ▶ PyTorch v1.7.0
- ▶ PyTorch v1.6.0
- ▶ PyTorch v1.5.0
- ▼ PyTorch v1.4.0 打開PyTorch v1.4.0
  - JetPack 4.4 Developer Preview (L4T R32.4.2)
    - Python 2.7 - torch-1.4.0-cp27-cp27mu-linux_aarch64.whl 245
    - Python 3.6 - torch-1.4.0-cp36-cp36m-linux_aarch64.whl 1.0k
  - JetPack 4.2 / 4.3
    - Python 2.7 - torch-1.4.0-cp27-cp27mu-linux_aarch64.whl 181
    - Python 3.6 - torch-1.4.0-cp36-cp36m-linux_aarch64.whl 1.5k ← 點擊下載檔案
- ▶ PyTorch v1.3.0
- ▶ PyTorch v1.2.0
- ▶ PyTorch v1.1.0
- ▶ PyTorch v1.0.0

Instructions

- ▶ Installation
- ▶ Verification
- ▶ Build from Source
- ▶ Note on Upgrading pip

安裝Pytorch 1.4.0

- $ sudo apt-get install python3-pip libopenblas-base
- $ sudo pip install torch-1.4.0-cp36-cp36m-linux_aarch64.whl
- $ python3
- $ import torch
- $ print(torch.__version__)



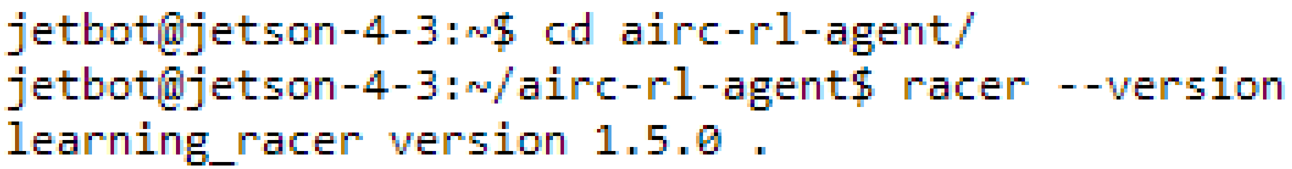
```
jetbot@jetson-4-3:~$ sudo pip install torch-1.4.0-cp36-cp36m-linux_aarch64.whl
Processing ./torch-1.4.0-cp36-cp36m-linux_aarch64.whl
torch is already installed with the same version as the provided wheel. Use --force-reinstall to f
orce an installation of the wheel.
jetbot@jetson-4-3:~$ python3
Python 3.6.9 (default, Oct  8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
1.4.0
>>>
```

安裝posix_ipc

- $ pip install posix-ipc
- Posix全名為”Portable Operating System Interface”，此介面是由IEEE開發的一個Unix標準。
- 現在大部分Unix和其他流行的版本都是依照POSIX標準，而Linux從一開始就依照POSIX標準來制定。
- Posix IPC包括:Message Queue、Semaphores、Shared Memory。

確認環境

- 環境建置完成後，可輸入指令檢查是否正確建立環境。
- $ cd airc-rl-agent
- $ racer –version
- 會顯示learning_racer version 1.5.0



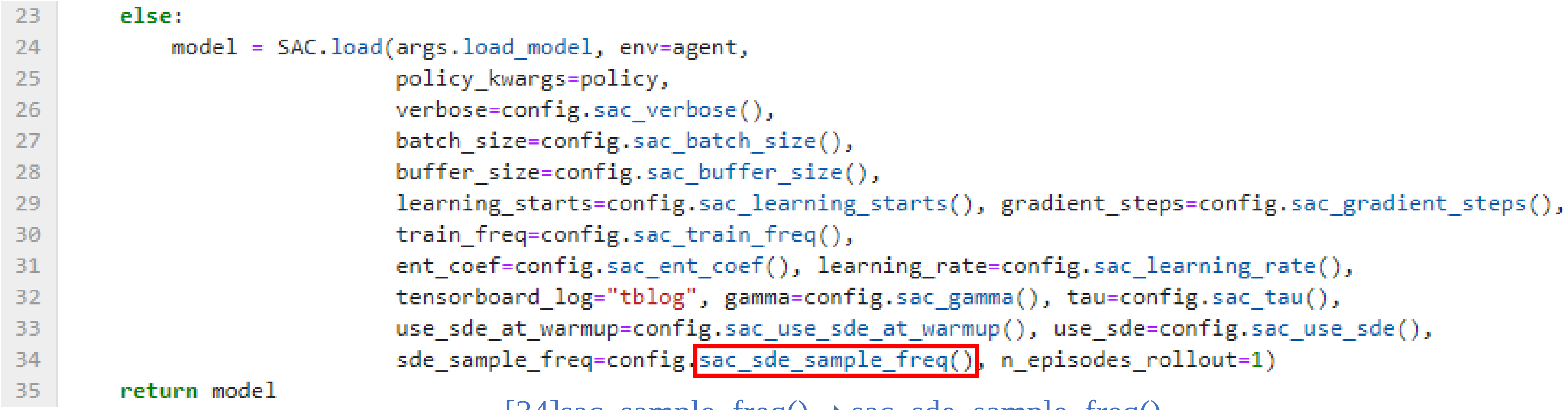
```
jetbot@jetson-4-3:~$ cd airc-rl-agent/
jetbot@jetson-4-3:~/airc-rl-agent$ racer --version
learning_racer version 1.5.0 .
```

複製learning_racer函示庫

- 複製函示庫至指定的目的地，使程式能正常執行引用函示庫。
- $ cd airc-rl-agent
- $ sudo cp -r ./learning_racer ./notebooks/
- $ sudo cp -r ./learning_racer ./notebooks/utility/jetbot/

修改learing_racerc函示庫

- 修改的檔案路徑：/airc-rl-agent/learning_racer/sac/custom_sac.py



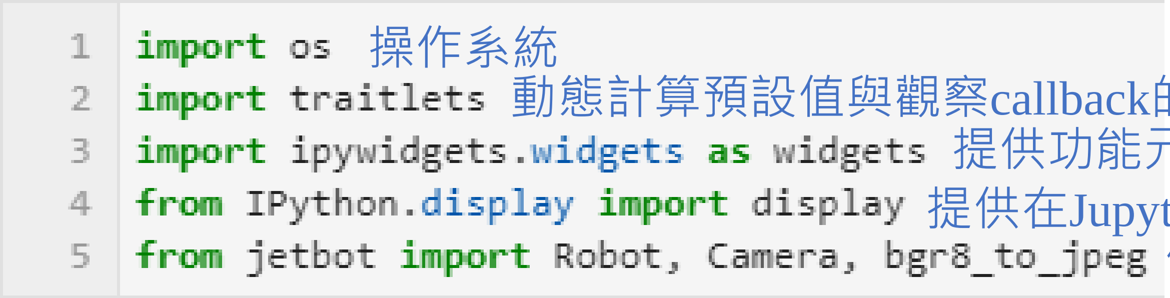
```
23      else:
24          model = SAC.load(args.load_model, env=agent,
25                           policy_kwargs=policy,
26                           verbose=config.sac_verbose(),
27                           batch_size=config.sac_batch_size(),
28                           buffer_size=config.sac_buffer_size(),
29                           learning_starts=config.sac_learning_starts(), gradient_steps=config.sac_gradient_steps(),
30                           train_freq=config.sac_train_freq(),
31                           ent_coef=config.sac_ent_coef(), learning_rate=config.sac_learning_rate(),
32                           tensorboard_log="tblog", gamma=config.sac_gamma(), tau=config.sac_tau(),
33                           use_sde_at_warmup=config.sac_use_sde_at_warmup(), use_sde=config.sac_use_sde(),
34                           sde_sample_freq=config.sac_sde_sample_freq(), n_episodes_rollout=1)
35      return model
```

[34]sac_sample_freq() ➡ sac_sde_sample_freq()

收集影像資料

- 檔案路徑:
  - airc-rl-agent/notebooks/utility/jetbot/data_collection_withoutgamepad.ipynb(無手把版本)

引用函示庫



```
import os
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
from jetbot import Robot, Camera, bgr8_to_jpeg
```

操作系統

動態計算預設值與觀察callback的功能

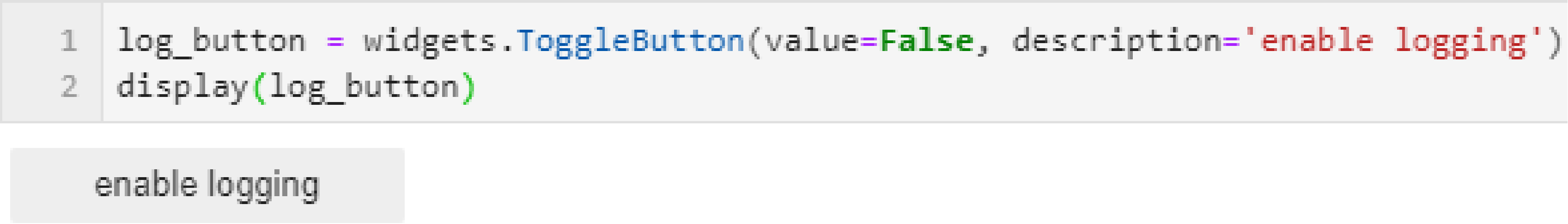
提供功能元件，例如:滑桿、顯示影片等等。

提供在JupyterLab上顯示影像的功能

使用來自jetbot提供的函示庫

產生log按鈕

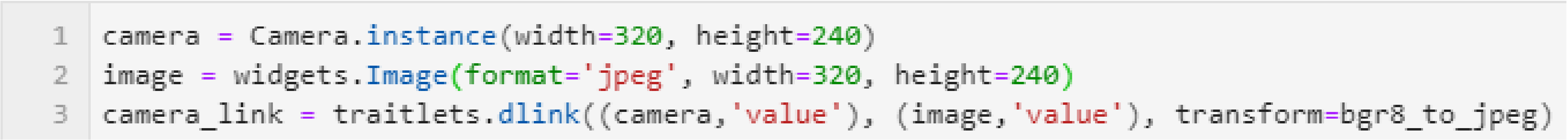
- 當按下按鈕時，將會開始記錄影像。
- 注意!現在按鈕還沒有功能，本範例只先建立了一個按鈕。



```
log_button = widgets.ToggleButton(value=False, description='enable logging')
display(log_button)
```

初始化相機

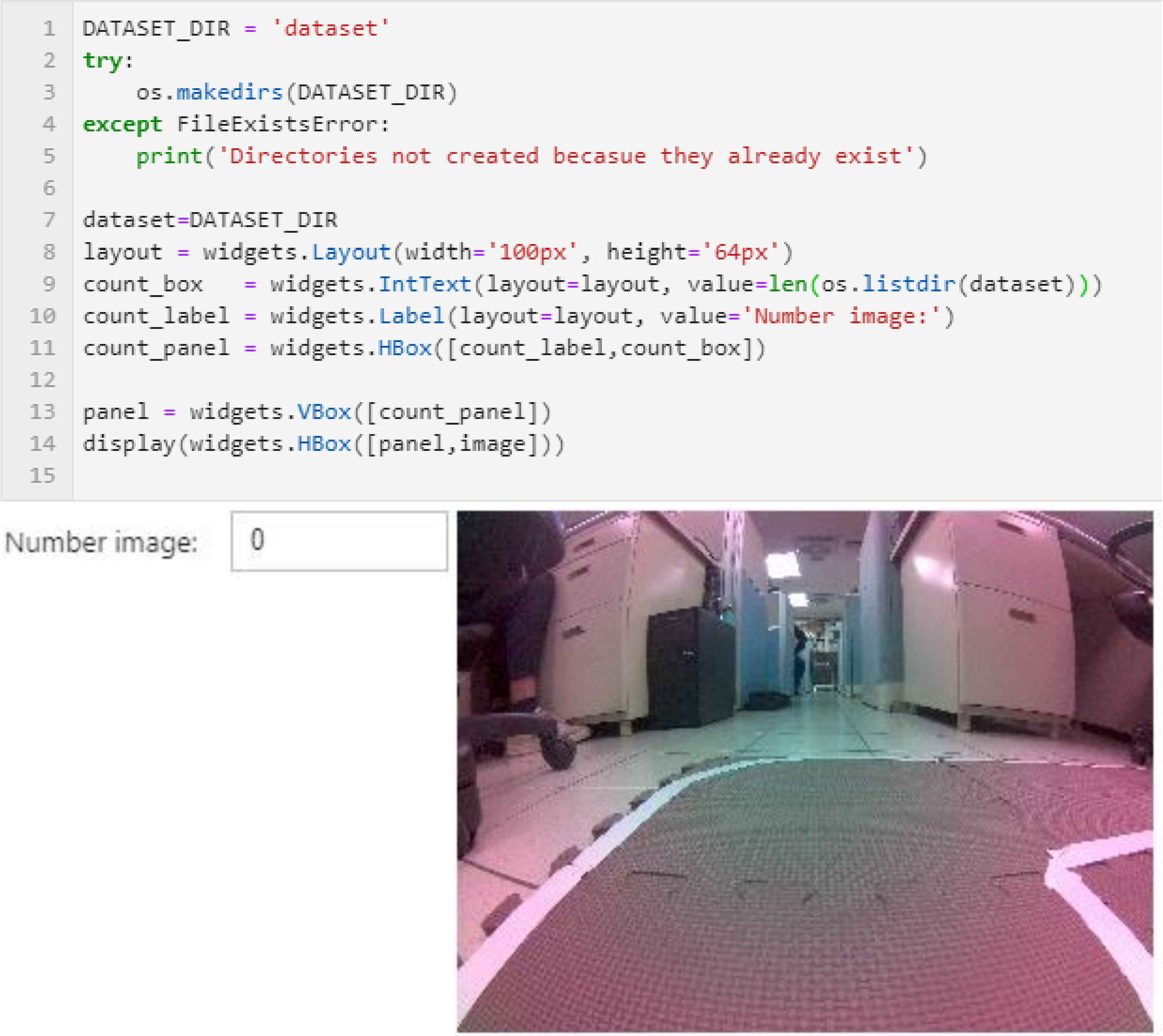
- [1]定義相機尺寸
- [2]定義影像格式
- [3]將前兩者的功能結合，並設定轉換的格式



```
camera = Camera.instance(width=320, height=240)
image = widgets.Image(format='jpeg', width=320, height=240)
camera_link = traitlets.dlink((camera,'value'), (image,'value'), transform=bgr8_to_jpeg)
```

產生UI元件

- [1]資料夾名稱
- [2]~[5]建立儲存影像的資料夾
- [7]~[11]建立顯示的文字框、數字框、數字框數字代表資料夾的影像數量
- [13]使內容垂直排列
- [14]水平顯示元件和影像

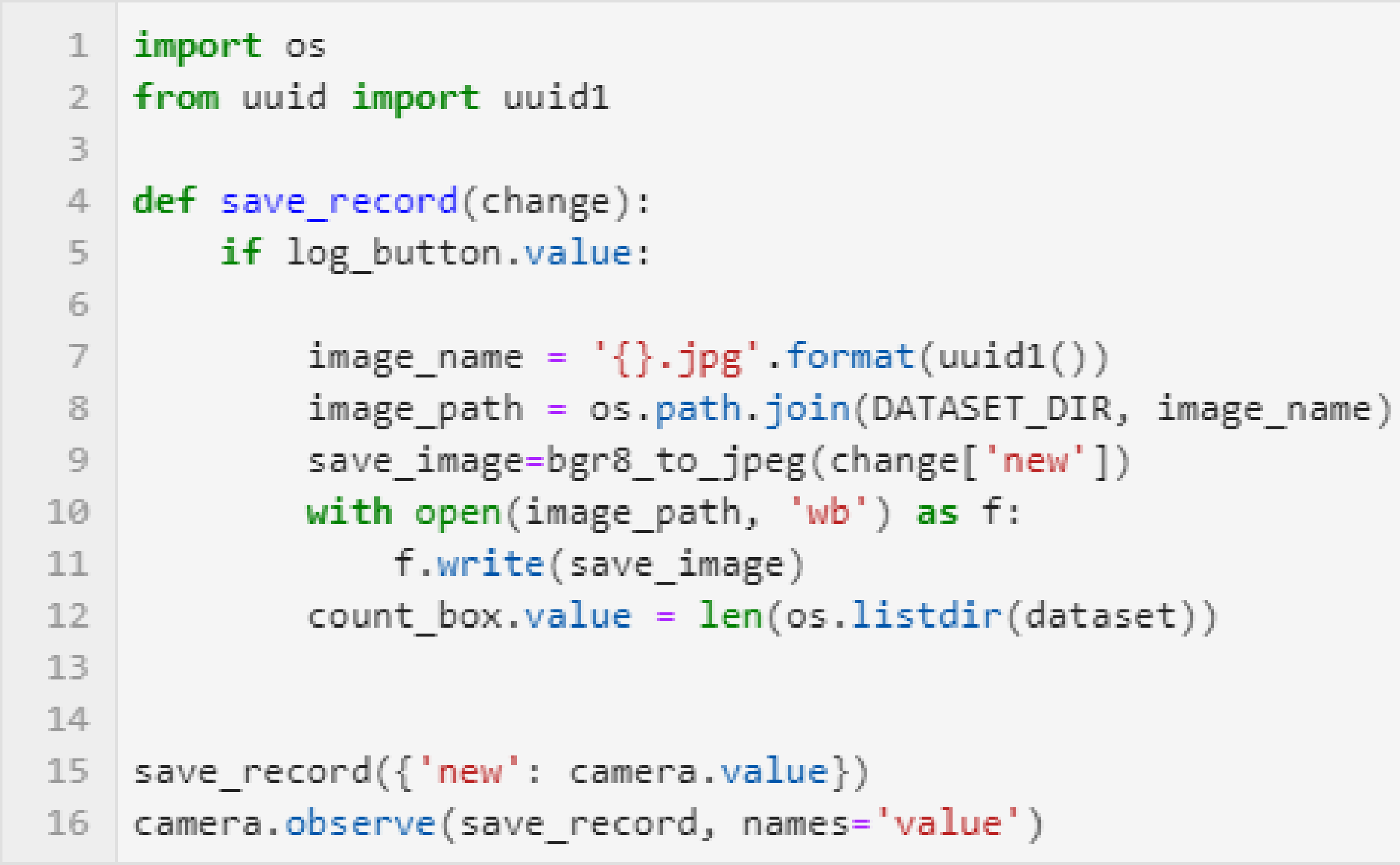


```
 1  DATASET_DIR = 'dataset'
 2  try:
 3      os.makedirs(DATASET_DIR)
 4  except FileExistsError:
 5      print('Directories not created becasue they already exist')
 6
 7  dataset=DATASET_DIR
 8  layout = widgets.Layout(width='100px', height='64px')
 9  count_box   = widgets.IntText(layout=layout, value=len(os.listdir(dataset)))
10  count_label = widgets.Label(layout=layout, value='Number image:')
11  count_panel = widgets.HBox([count_label,count_box])
12
13  panel = widgets.VBox([count_panel])
14  display(widgets.HBox([panel,image]))
15
```

Number image: 0

設定按鈕的callback

- 使用前面範例程式的按鈕參數，設定按鈕功能。
- [4]~[12]當按鈕按下時，使用uuid產生唯一的檔案名稱，且轉換影像格式，最後將影像儲存至資料夾中，回傳資料夾的影像數量。
- [15]將新的影像傳入save_record function中。
- [16]相機資訊訂閱save_record function，給予save_record function 相機數值。



```
 1  import os
 2  from uuid import uuid1
 3  
 4  def save_record(change):
 5      if log_button.value:
 6  
 7          image_name = '{}.jpg'.format(uuid1())
 8          image_path = os.path.join(DATASET_DIR, image_name)
 9          save_image=bgr8_to_jpeg(change['new'])
10          with open(image_path, 'wb') as f:
11              f.write(save_image)
12          count_box.value = len(os.listdir(dataset))
13  
14  
15  save_record({'new': camera.value})
16  camera.observe(save_record, names='value')
```

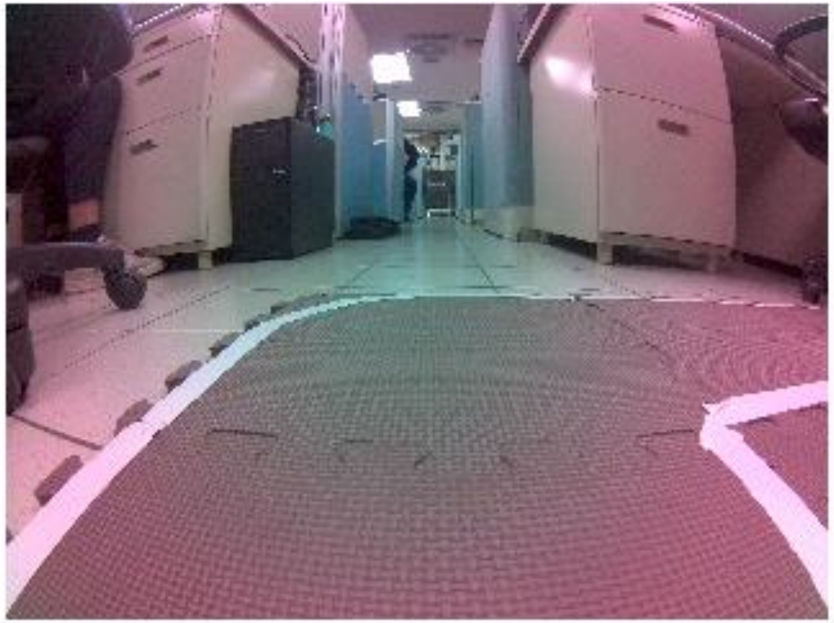
收集影像的方法

- 執行完前面的範例程式碼後，現在可以開始收集影像了。
- 按鈕按下後，請用人工的方式推著Jetbot收集道路中的影像，盡可能的模仿Jetbot在行走時的路線，請收集1000到10000張影像，提供訓練使用。

1. 按下按鈕收集數據

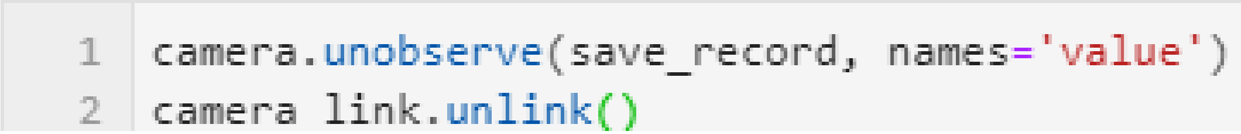
Number image: 0

2. Jetbot會開始收集資料



釋放observe跟camera

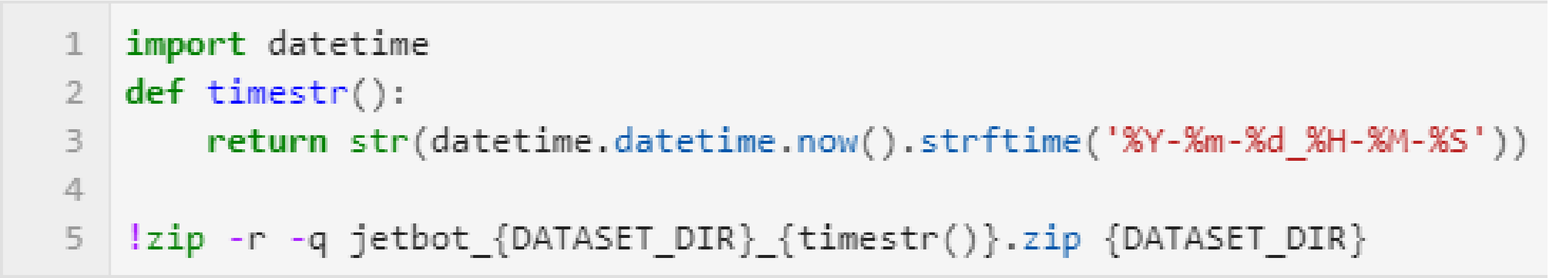
- 當收集了足夠的影像資料後，釋放掉相機的observe訂閱跟相機的記憶體。



```
camera.unobserve(save_record, names='value')
camera_link.unlink()
```

建立dataset.zip

- 將影像資料夾壓縮成zip。



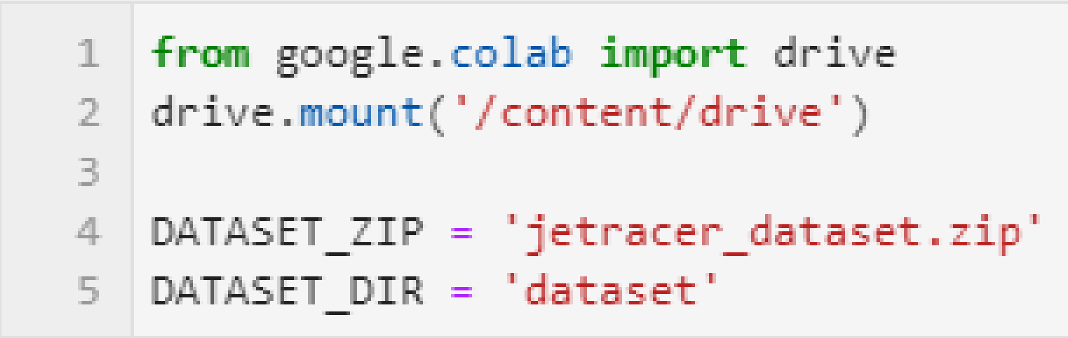
```
import datetime
def timestr():
    return str(datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S'))

!zip -r -q jetbot_{DATASET_DIR}_{timestr()}.zip {DATASET_DIR}
```

訓練資料集

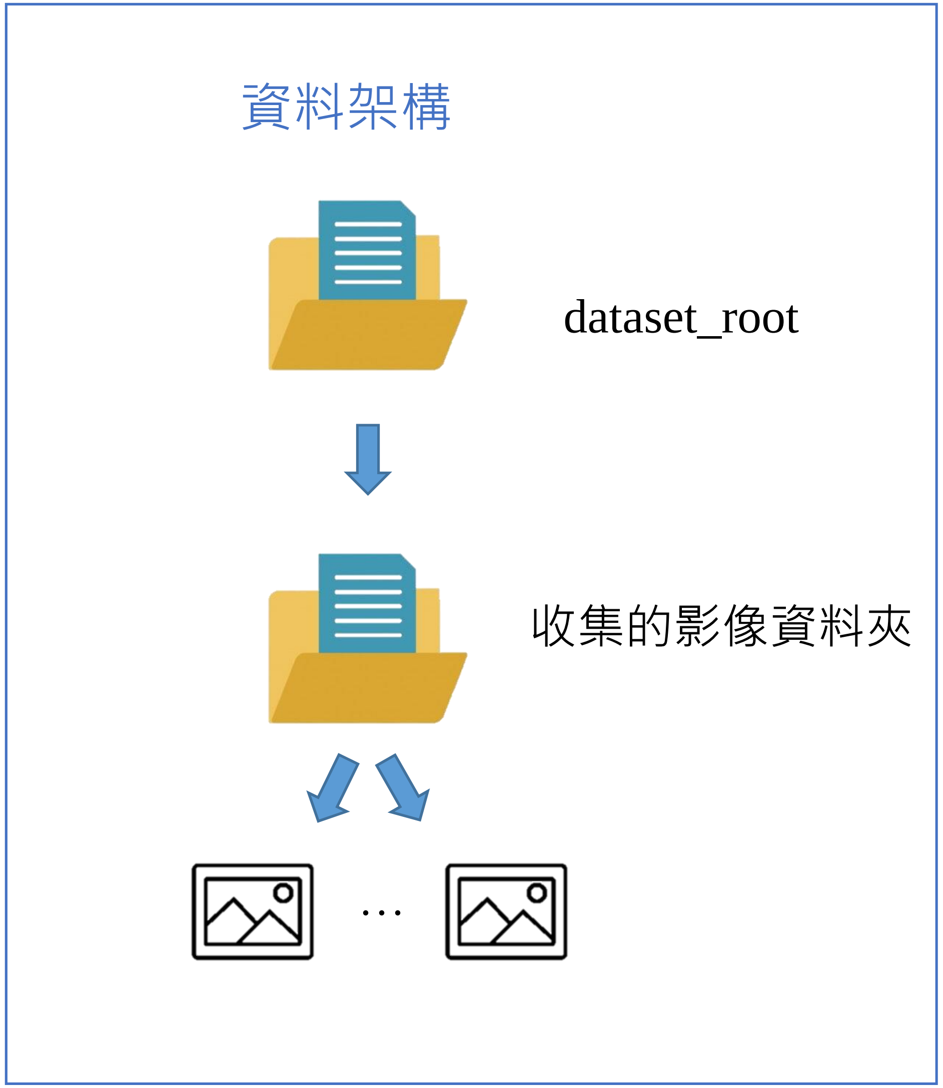
- 執行本範例程式建議使用google colab或者是運算能力較高的設備。
- 檔案路徑 : notebooks/colabo/VAE_CNN.ipynb
- 將VAE_CNN.ipynb上傳至google colab

使用google drive

- 申請使用google drive的權限
- 將影像資料夾上傳至雲端
- [4]壓縮的影像資料夾名稱
- [5]資料夾名稱
- 助教提醒!如果程式碼在執行上有問題，同學請自行更改部分程式碼，或用其他方式讀取到檔案即可。



```
from google.colab import drive
drive.mount('/content/drive')

DATASET_ZIP = 'jetracer_dataset.zip'
DATASET_DIR = 'dataset'
```



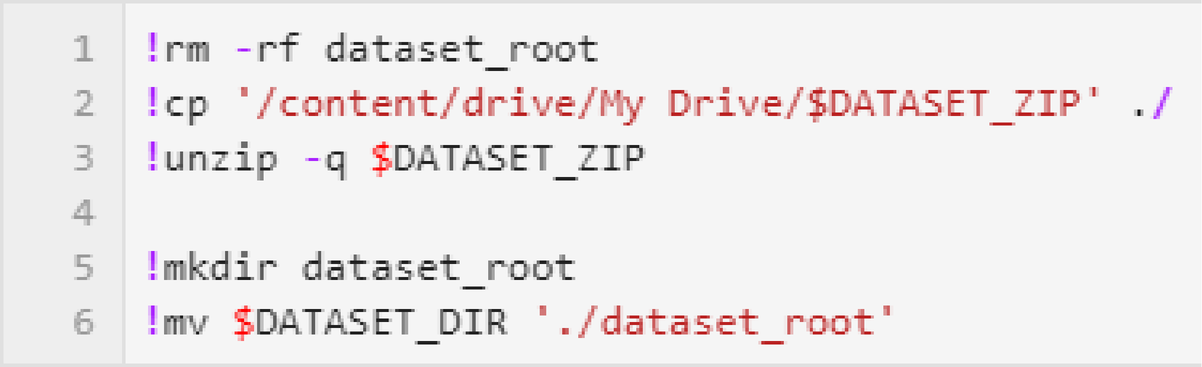
資料架構

dataset_root

收集的影像資料夾

解壓縮資料夾

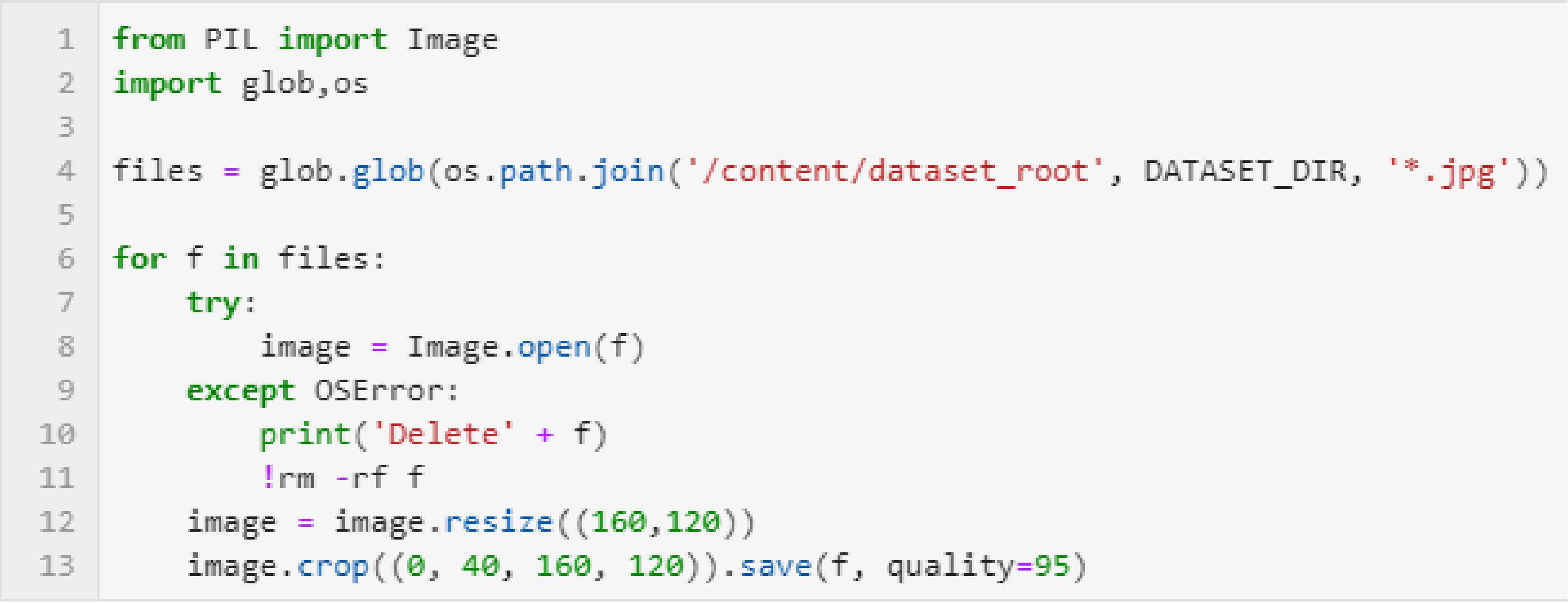
- [1]刪除資料夾
- [2]複製路徑上的檔案
- [3]解壓縮
- [5]建立資料夾
- [6]移動資料夾
- 助教提醒!如果程式碼在執行上有問題，同學請自行更改部分程式碼，或用其他方式讀取到檔案即可。



```
1  !rm -rf dataset_root
2  !cp '/content/drive/My Drive/$DATASET_ZIP' ./
3  !unzip -q $DATASET_ZIP
4
5  !mkdir dataset_root
6  !mv $DATASET_DIR './dataset_root'
```

重新設定影像尺寸

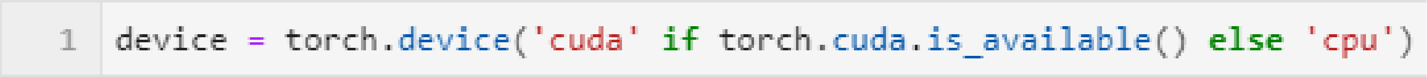
- [4]讀取資料夾底下的.jpg
- [6]~[13]開啟圖片，重新設定影像尺寸為(160, 120)，並裁切影像x軸0~160; y軸為40~120，儲存的品質為95，降低影像的材質與大小，以降低訓練的負擔。



```
 1  from PIL import Image
 2  import glob,os
 3
 4  files = glob.glob(os.path.join('/content/dataset_root', DATASET_DIR, '*.jpg'))
 5
 6  for f in files:
 7      try:
 8          image = Image.open(f)
 9      except OSError:
10          print('Delete' + f)
11          !rm -rf f
12      image = image.resize((160,120))
13      image.crop((0, 40, 160, 120)).save(f, quality=95)
```

使用GPU

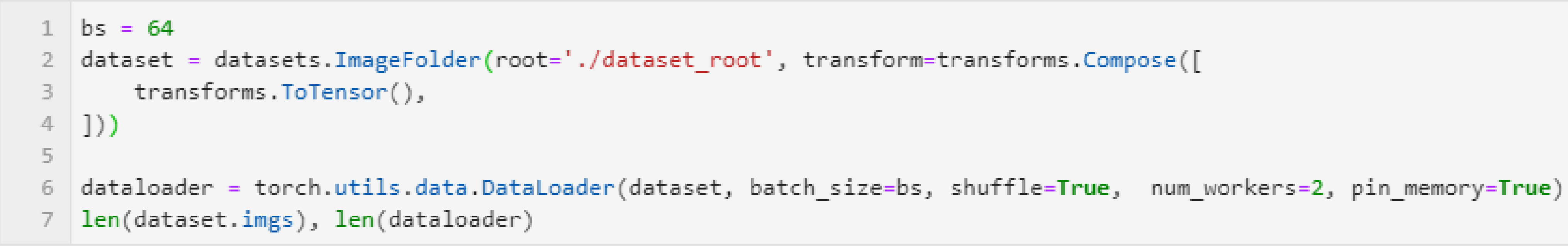
- 使用cuda來加速運算效能



```python
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

讀取資料集(1/2)

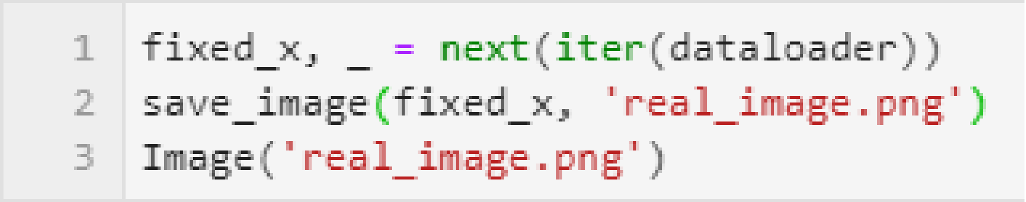
- [2]~[4]將資料集轉成Tensor的格式
- [6]產生要讀取的資料格式，並設定batch_size, shuffle, num_work, pin_memory。
- [7]顯示影像數量與迭代次數



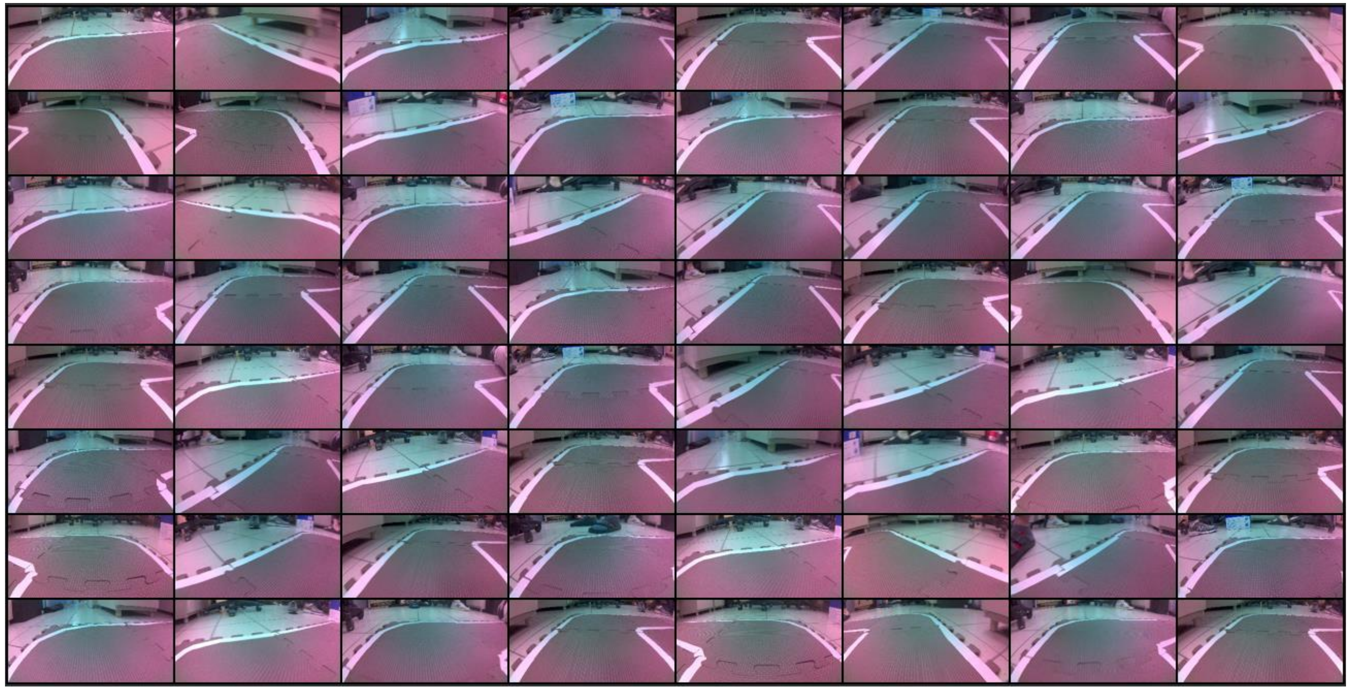
```python
1 bs = 64
2 dataset = datasets.ImageFolder(root='./dataset_root', transform=transforms.Compose([
3     transforms.ToTensor(),
4 ]))
5 
6 dataloader = torch.utils.data.DataLoader(dataset, batch_size=bs, shuffle=True,  num_workers=2, pin_memory=True)
7 len(dataset.imgs), len(dataloader)
```

讀取資料集(2/2)

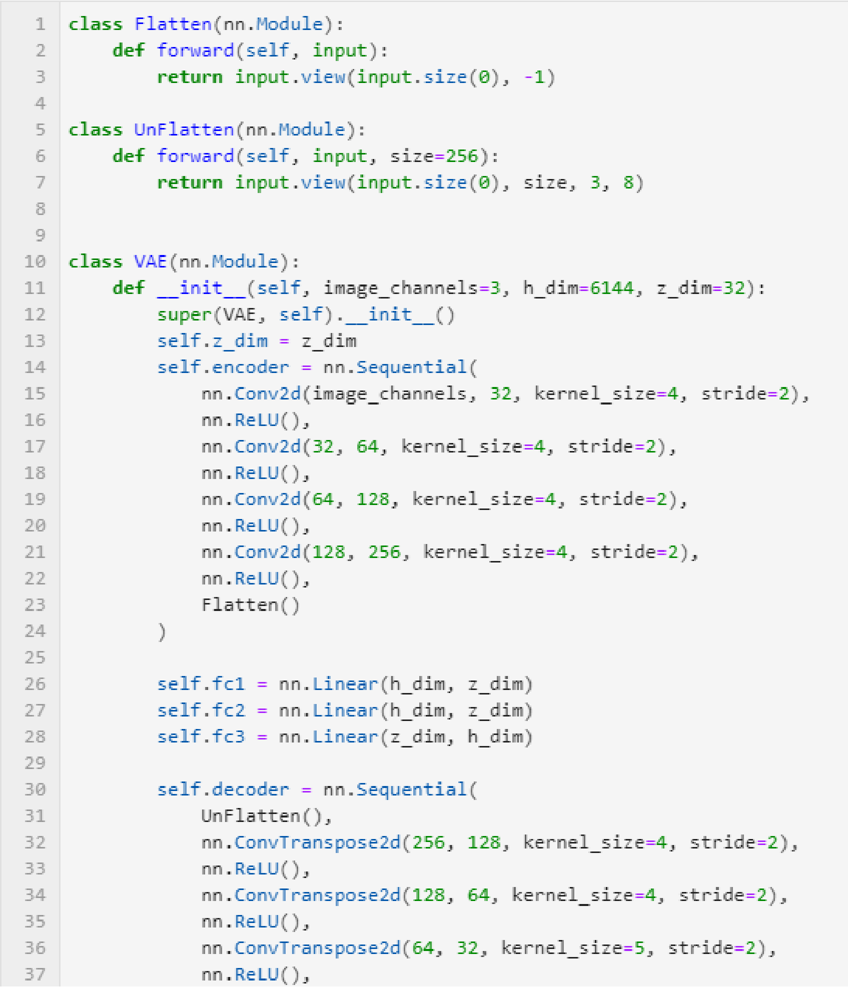
- 儲存每次迭代的影像，並顯示在JupyterLab上。



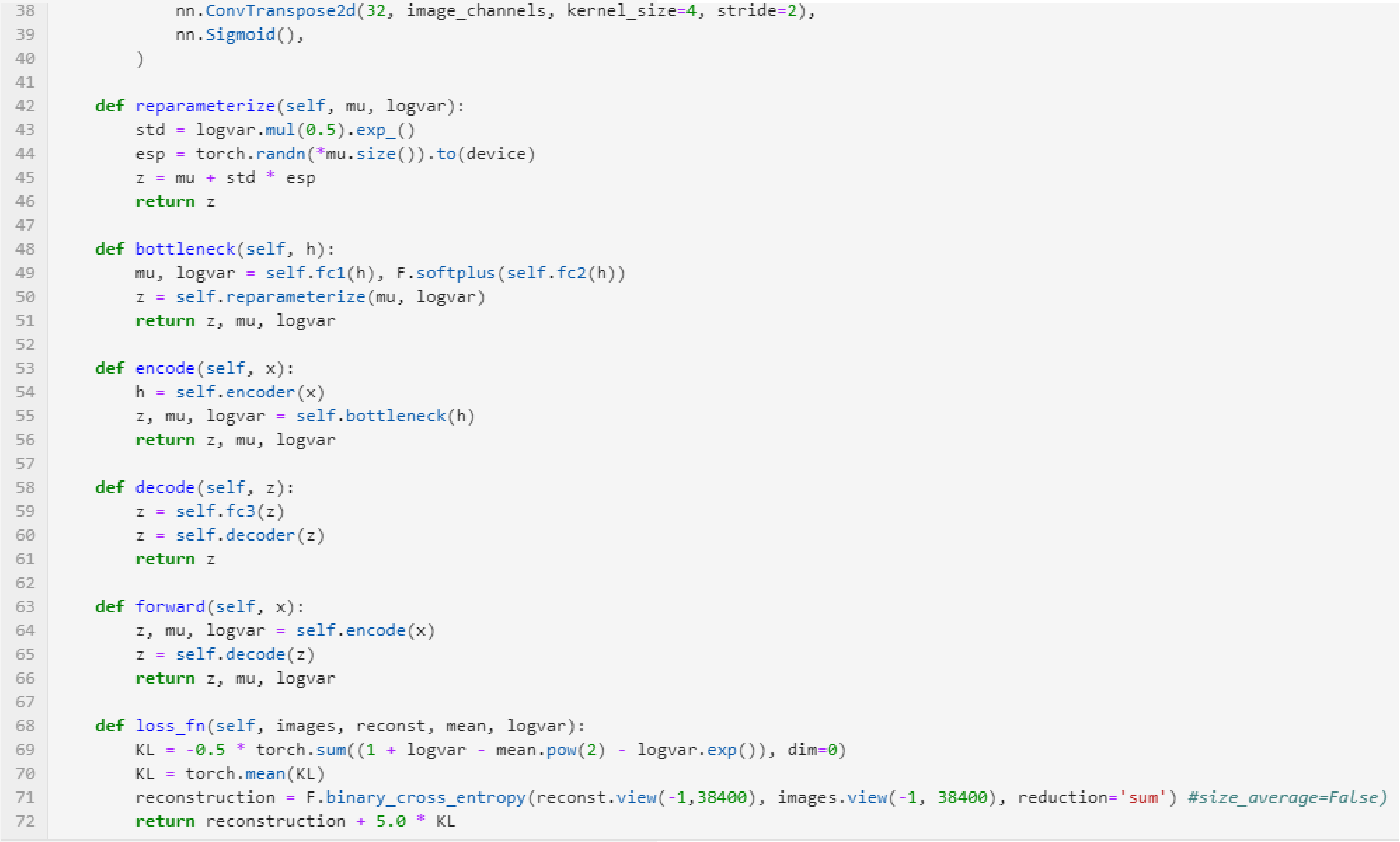
```
fixed_x, _ = next(iter(dataloader))
save_image(fixed_x, 'real_image.png')
Image('real_image.png')
```



定義VAE神經網路架構



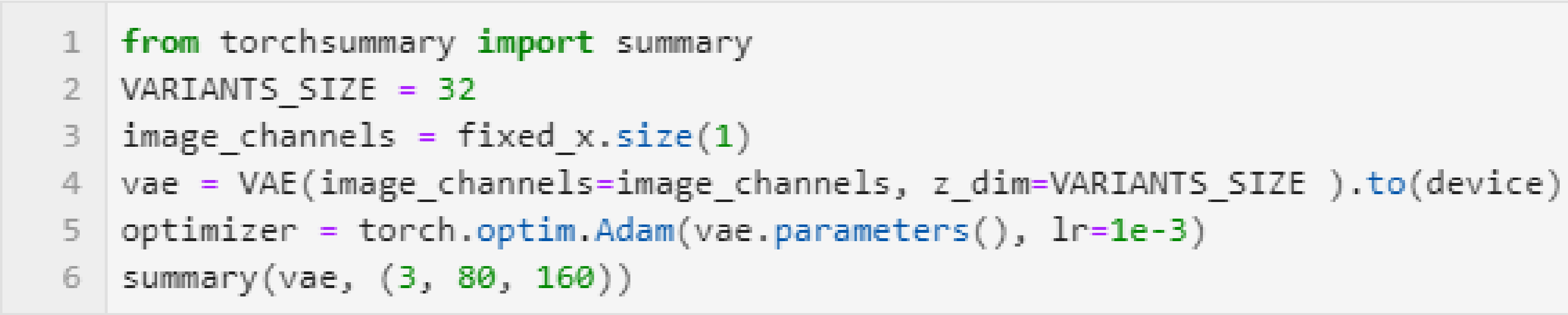
```python
class Flatten(nn.Module):
    def forward(self, input):
        return input.view(input.size(0), -1)

class UnFlatten(nn.Module):
    def forward(self, input, size=256):
        return input.view(input.size(0), size, 3, 8)


class VAE(nn.Module):
    def __init__(self, image_channels=3, h_dim=6144, z_dim=32):
        super(VAE, self).__init__()
        self.z_dim = z_dim
        self.encoder = nn.Sequential(
            nn.Conv2d(image_channels, 32, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.Conv2d(64, 128, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.Conv2d(128, 256, kernel_size=4, stride=2),
            nn.ReLU(),
            Flatten()
        )

        self.fc1 = nn.Linear(h_dim, z_dim)
        self.fc2 = nn.Linear(h_dim, z_dim)
        self.fc3 = nn.Linear(z_dim, h_dim)

        self.decoder = nn.Sequential(
            UnFlatten(),
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, kernel_size=5, stride=2),
            nn.ReLU(),
```



```python
            nn.ConvTranspose2d(32, image_channels, kernel_size=4, stride=2),
            nn.Sigmoid(),
        )

    def reparameterize(self, mu, logvar):
        std = logvar.mul(0.5).exp_()
        esp = torch.randn(*mu.size()).to(device)
        z = mu + std * esp
        return z

    def bottleneck(self, h):
        mu, logvar = self.fc1(h), F.softplus(self.fc2(h))
        z = self.reparameterize(mu, logvar)
        return z, mu, logvar

    def encode(self, x):
        h = self.encoder(x)
        z, mu, logvar = self.bottleneck(h)
        return z, mu, logvar

    def decode(self, z):
        z = self.fc3(z)
        z = self.decoder(z)
        return z

    def forward(self, x):
        z, mu, logvar = self.encode(x)
        z = self.decode(z)
        return z, mu, logvar

    def loss_fn(self, images, reconst, mean, logvar):
        KL = -0.5 * torch.sum((1 + logvar - mean.pow(2) - logvar.exp()), dim=0)
        KL = torch.mean(KL)
        reconstruction = F.binary_cross_entropy(reconst.view(-1,38400), images.view(-1, 38400), reduction='sum') #size_average=False)
        return reconstruction + 5.0 * KL
```

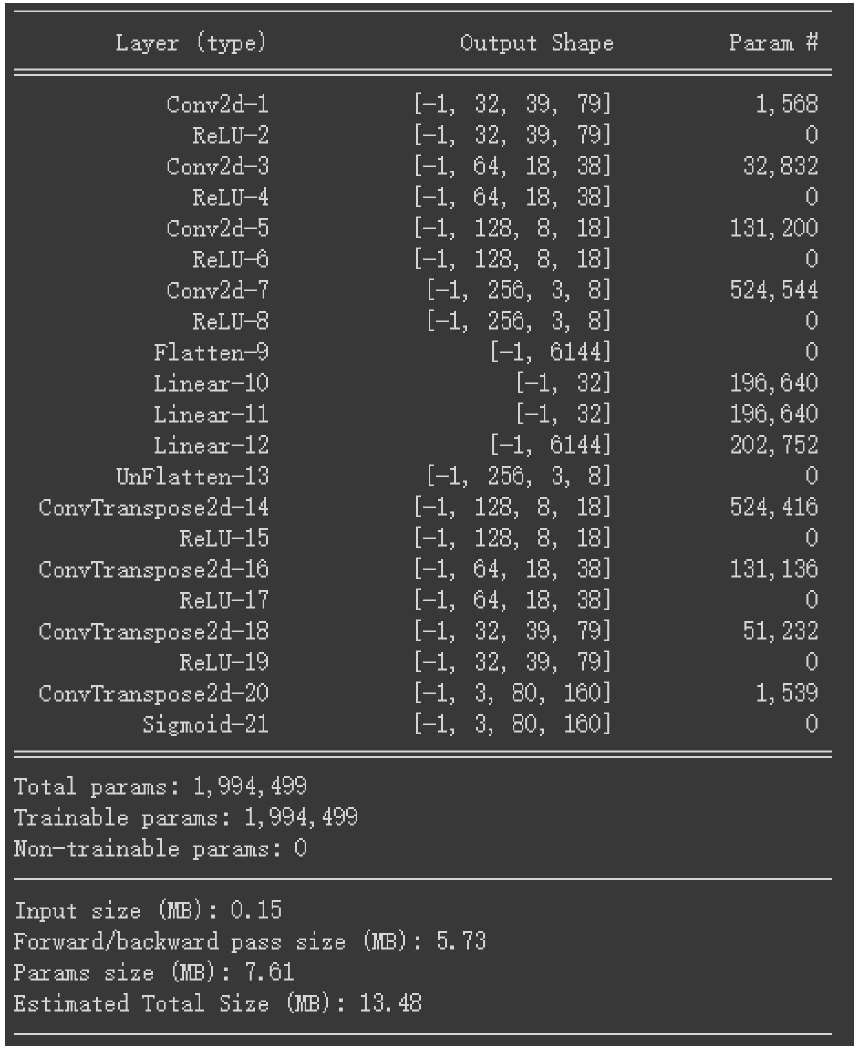
訓練前的準備

- 建立VAE神經網路和初始最佳化模型
- [2]設定色彩深度為32bit
- [3]影像色彩通道設定為1(灰階)。
- [4]定義網路輸入的影像格式，並使用cuda核心。
- [5]使用Adam最佳化模型，並代入模型參數，學習率為1e-3。
- [6]視覺化模型架構，設定輸入的影像格式。



```
1  from torchsummary import summary
2  VARIANTS_SIZE = 32
3  image_channels = fixed_x.size(1)
4  vae = VAE(image_channels=image_channels, z_dim=VARIANTS_SIZE ).to(device)
5  optimizer = torch.optim.Adam(vae.parameters(), lr=1e-3)
6  summary(vae, (3, 80, 160))
```

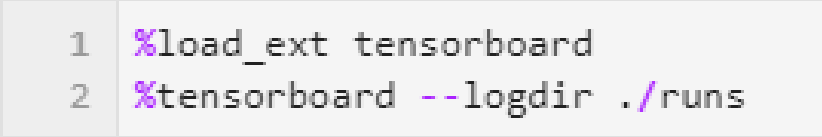
模型架構



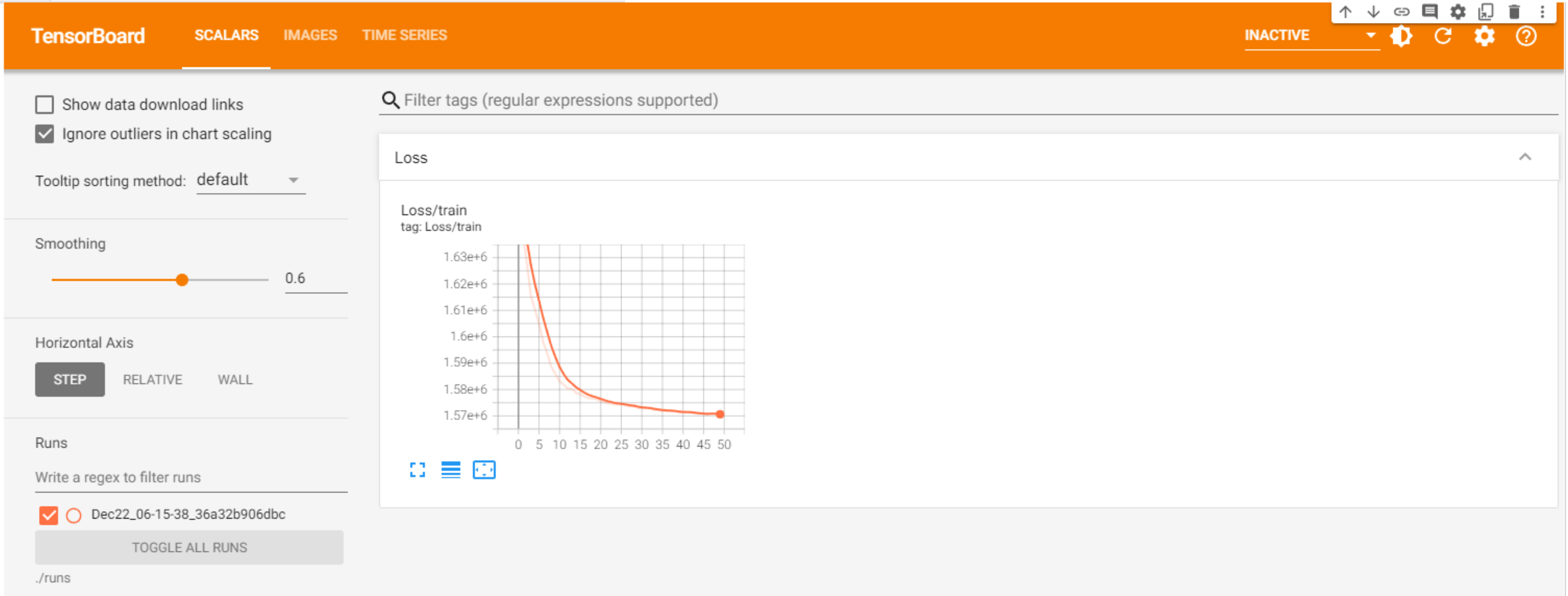
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 39, 79]           1,568
              ReLU-2           [-1, 32, 39, 79]               0
            Conv2d-3           [-1, 64, 18, 38]          32,832
              ReLU-4           [-1, 64, 18, 38]               0
            Conv2d-5           [-1, 128, 8, 18]         131,200
              ReLU-6           [-1, 128, 8, 18]               0
            Conv2d-7            [-1, 256, 3, 8]         524,544
              ReLU-8            [-1, 256, 3, 8]               0
           Flatten-9                 [-1, 6144]               0
           Linear-10                   [-1, 32]         196,640
           Linear-11                   [-1, 32]         196,640
           Linear-12                 [-1, 6144]         202,752
        UnFlatten-13            [-1, 256, 3, 8]               0
  ConvTranspose2d-14           [-1, 128, 8, 18]         524,416
             ReLU-15           [-1, 128, 8, 18]               0
  ConvTranspose2d-16           [-1, 64, 18, 38]         131,136
             ReLU-17           [-1, 64, 18, 38]               0
  ConvTranspose2d-18           [-1, 32, 39, 79]          51,232
             ReLU-19           [-1, 32, 39, 79]               0
  ConvTranspose2d-20           [-1, 3, 80, 160]           1,539
          Sigmoid-21           [-1, 3, 80, 160]               0
================================================================
Total params: 1,994,499
Trainable params: 1,994,499
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.15
Forward/backward pass size (MB): 5.73
Params size (MB): 7.61
Estimated Total Size (MB): 13.48
----------------------------------------------------------------
```

Tensorboard

- 視覺化訓練過程，可從統計圖中得知訓練次數、Loss、訓練結果等等。在訓練的過程中統計圖會不斷變化。



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



□ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ Dec22_06-15-38_36a32b906dbc

TOGGLE ALL RUNS

./runs

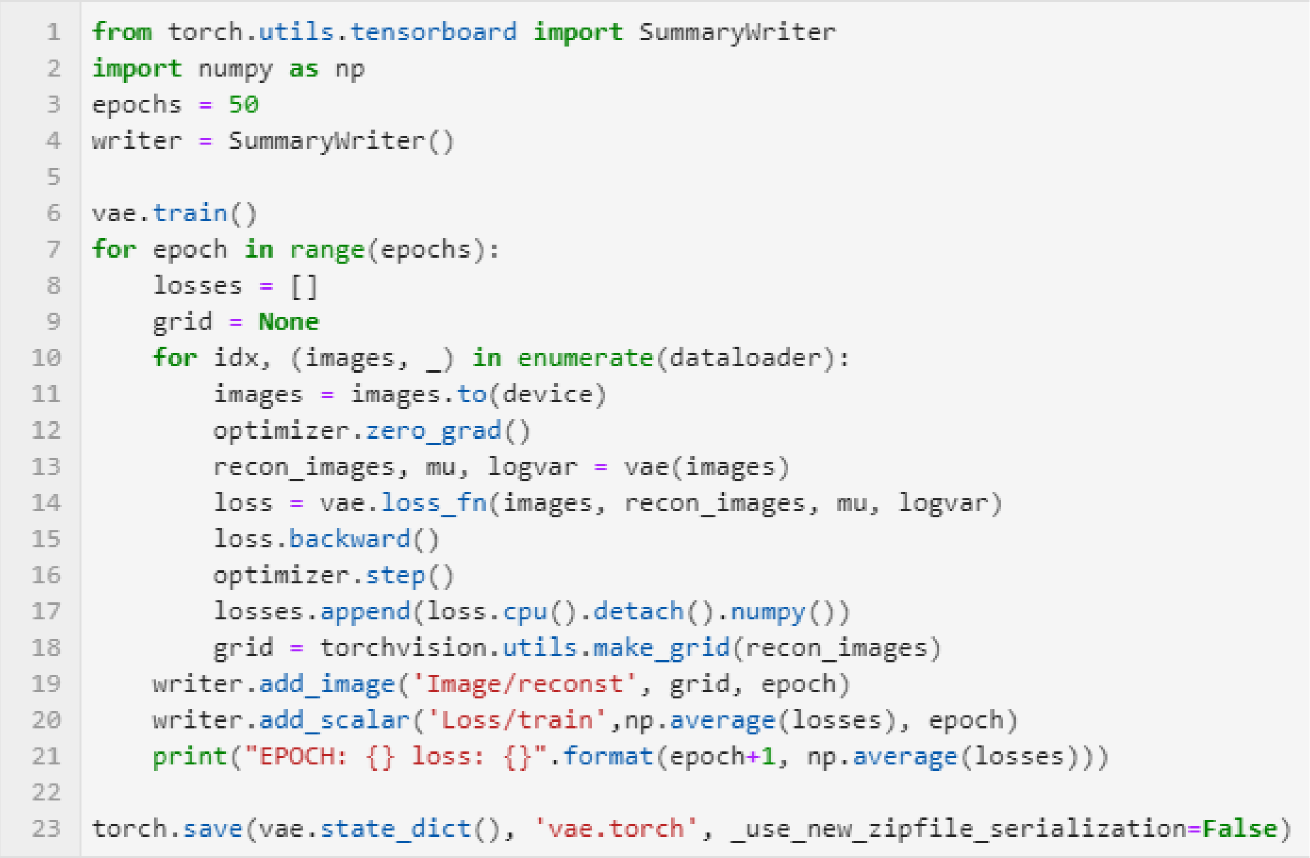
Filter tags (regular expressions supported)

Loss

Loss/train

tag: Loss/train

開始訓練

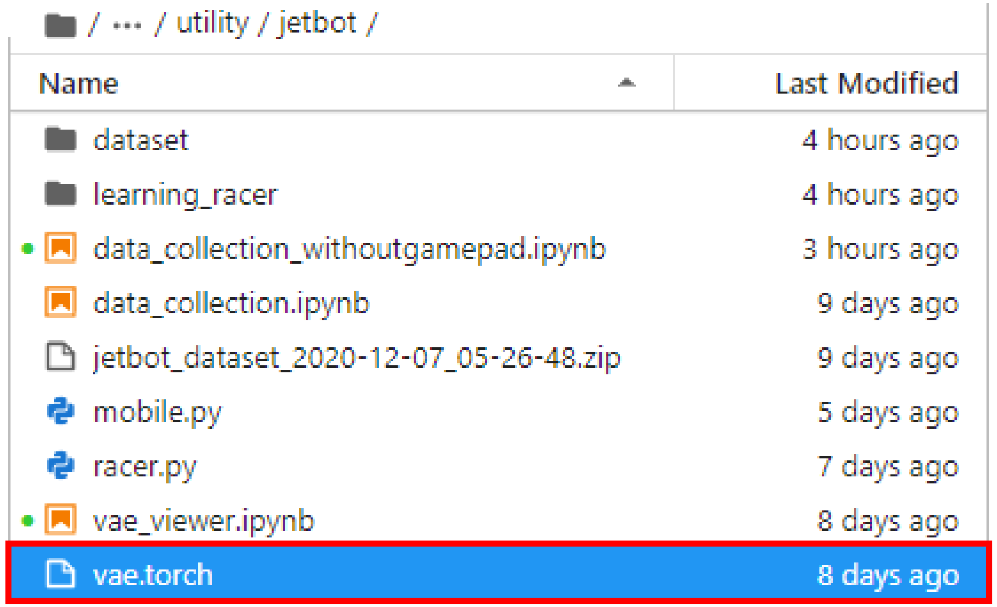


```
 1  from torch.utils.tensorboard import SummaryWriter
 2  import numpy as np
 3  epochs = 50
 4  writer = SummaryWriter()
 5  
 6  vae.train()
 7  for epoch in range(epochs):
 8      losses = []
 9      grid = None
10      for idx, (images, _) in enumerate(dataloader):
11          images = images.to(device)
12          optimizer.zero_grad()
13          recon_images, mu, logvar = vae(images)
14          loss = vae.loss_fn(images, recon_images, mu, logvar)
15          loss.backward()
16          optimizer.step()
17          losses.append(loss.cpu().detach().numpy())
18          grid = torchvision.utils.make_grid(recon_images)
19      writer.add_image('Image/reconst', grid, epoch)
20      writer.add_scalar('Loss/train',np.average(losses), epoch)
21      print("EPOCH: {} loss: {}".format(epoch+1, np.average(losses)))
22  
23  torch.save(vae.state_dict(), 'vae.torch', _use_new_zipfile_serialization=False)
```

- 訓練數次模型，請同學自行調整到最佳的訓練次數，完成模型訓練。
- [6]~[18]模型訓練及更新權重
- [19]新增影像資料到summary
- [20]新增loss與訓練次數到summary。
- [23]儲存權重到vae.torch模型中
- 在路徑下會產生一個vae.torch模型檔案

下載vae.torch至目錄中

- 請將vae.torch複製到airc-rl-agent/notebooks/utility/jetbot/

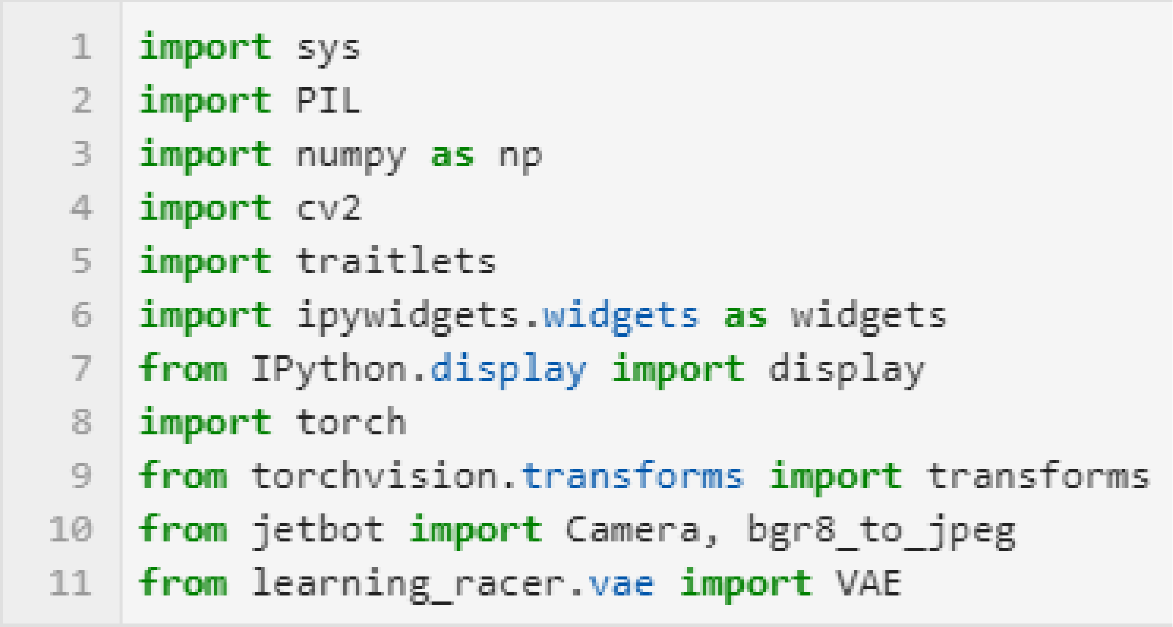


/ ••• / utility / jetbot /

Name	Last Modified
dataset	4 hours ago
learning_racer	4 hours ago
data_collection_withoutgamepad.ipynb	3 hours ago
data_collection.ipynb	9 days ago
jetbot_dataset_2020-12-07_05-26-48.zip	9 days ago
mobile.py	5 days ago
racer.py	7 days ago
vae_viewer.ipynb	8 days ago
vae.torch	8 days ago

引用函示庫

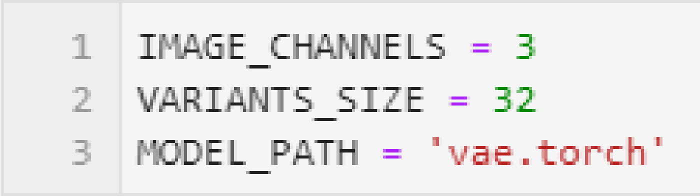
- 執行檔案路徑:airc-rl-agent/notebooks/utility/jetbot/vae_viewer.ipynb



```python
import sys
import PIL
import numpy as np
import cv2
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
import torch
from torchvision.transforms import transforms
from jetbot import Camera, bgr8_to_jpeg
from learning_racer.vae import VAE
```

設定參數

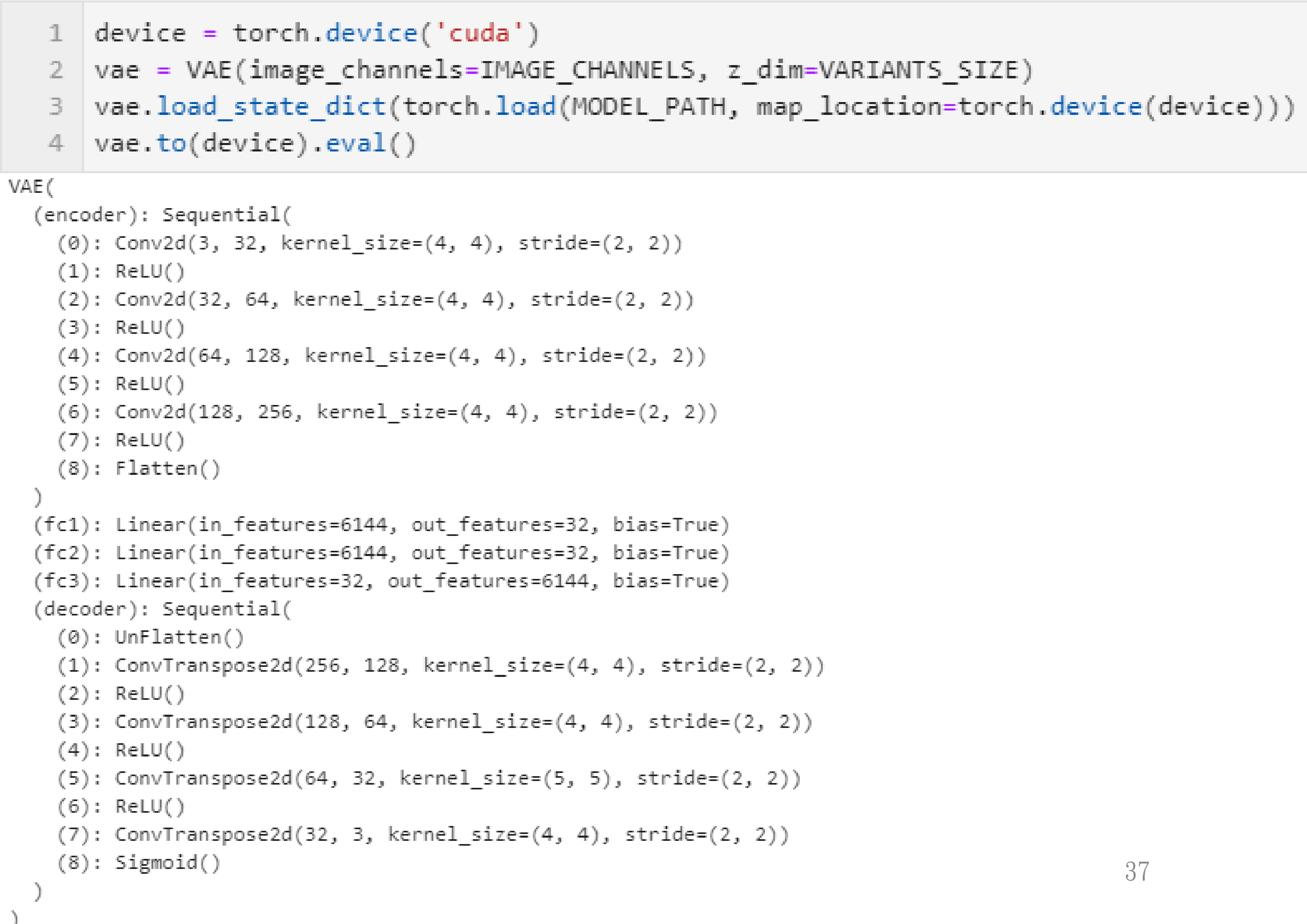
- [1]設定影像通道數
- [2]設定影像色彩深度
- [3]已訓練模型路徑



```python
IMAGE_CHANNELS = 3
VARIANTS_SIZE = 32
MODEL_PATH = 'vae.torch'
```

讀取已訓練模型

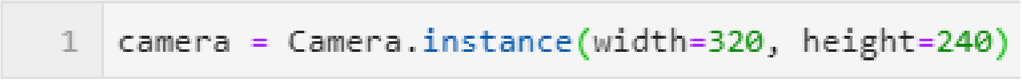
- [1]使用cuda核心加速運算
- [2]模型參數設定與訓練時相同
- [3]讀取模型
- [4]讓模型使用評估模式



```
device = torch.device('cuda')
vae = VAE(image_channels=IMAGE_CHANNELS, z_dim=VARIANTS_SIZE)
vae.load_state_dict(torch.load(MODEL_PATH, map_location=torch.device(device)))
vae.to(device).eval()
```

```
VAE(
  (encoder): Sequential(
    (0): Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
    (3): ReLU()
    (4): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2))
    (5): ReLU()
    (6): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2))
    (7): ReLU()
    (8): Flatten()
  )
  (fc1): Linear(in_features=6144, out_features=32, bias=True)
  (fc2): Linear(in_features=6144, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=6144, bias=True)
  (decoder): Sequential(
    (0): UnFlatten()
    (1): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2))
    (2): ReLU()
    (3): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2))
    (4): ReLU()
    (5): ConvTranspose2d(64, 32, kernel_size=(5, 5), stride=(2, 2))
    (6): ReLU()
    (7): ConvTranspose2d(32, 3, kernel_size=(4, 4), stride=(2, 2))
    (8): Sigmoid()
  )
)
```

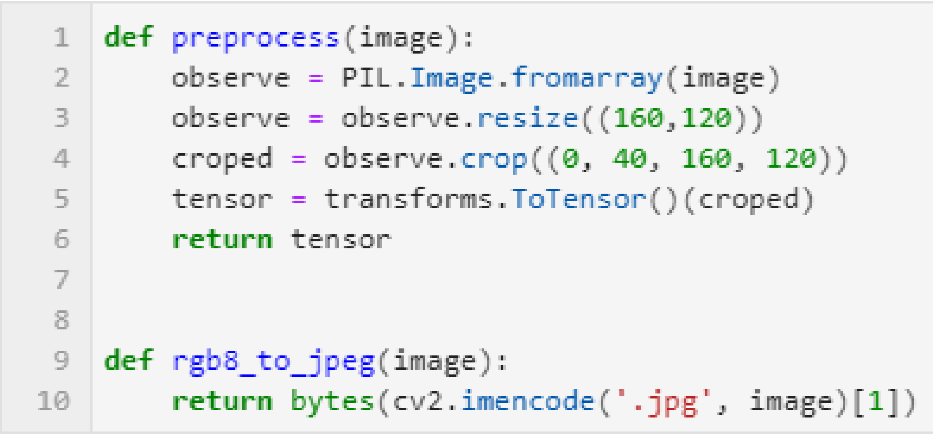
設定相機尺寸



```python
camera = Camera.instance(width=320, height=240)
```

定義預處理和後處理

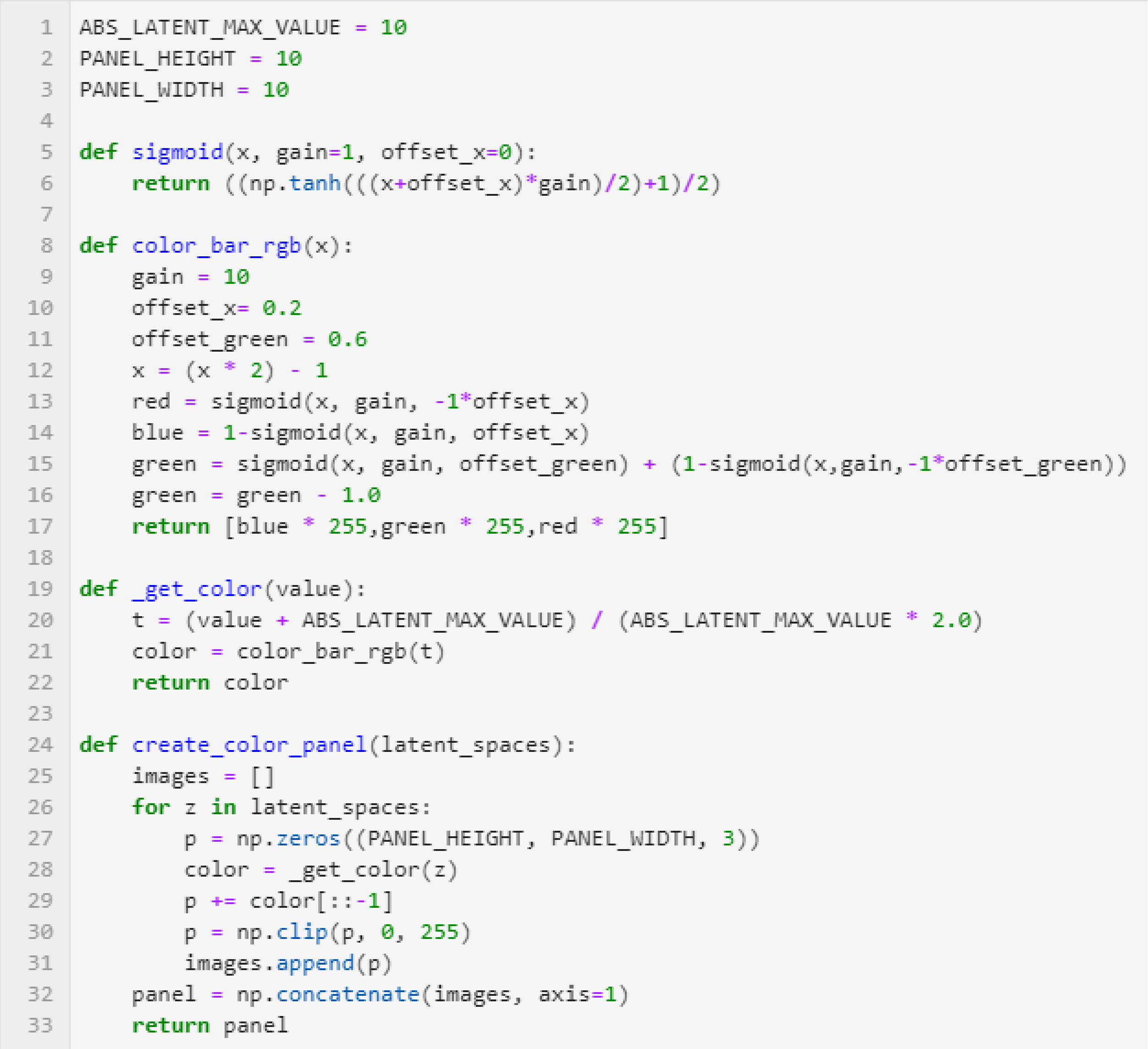
- 預處理和後處理都與訓練時相同



```python
def preprocess(image):
    observe = PIL.Image.fromarray(image)
    observe = observe.resize((160,120))
    croped = observe.crop((0, 40, 160, 120))
    tensor = transforms.ToTensor()(croped)
    return tensor


def rgb8_to_jpeg(image):
    return bytes(cv2.imencode('.jpg', image)[1])
```

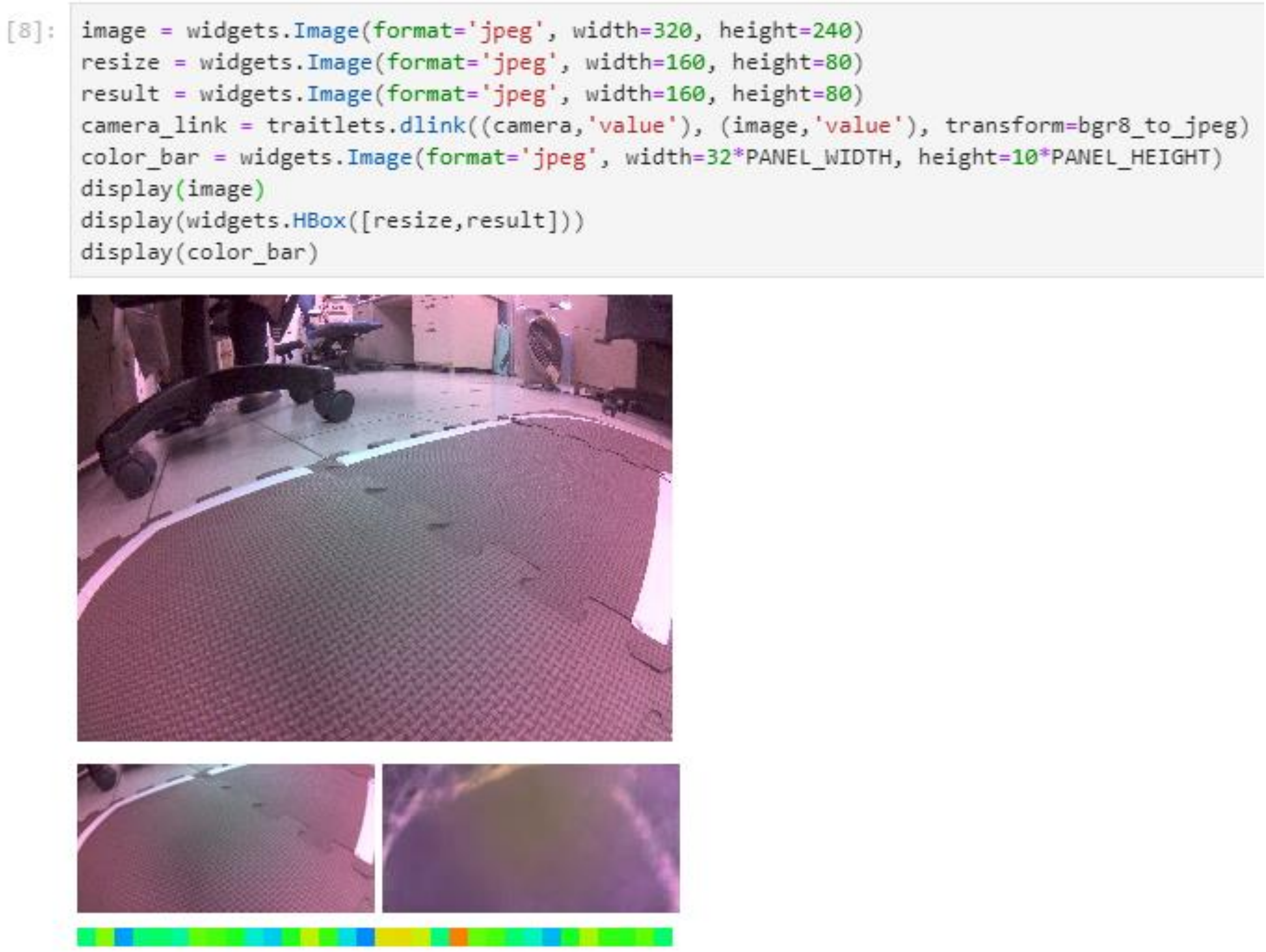
定義視覺化空間函數

- [5]~[6]定義sigmoid
- [8]~[17]定義顏色的RGB
- [19]~[22]計算顏色數值
- [24]~[33]定義顯示的面板尺寸



```
 1  ABS_LATENT_MAX_VALUE = 10
 2  PANEL_HEIGHT = 10
 3  PANEL_WIDTH = 10
 4
 5  def sigmoid(x, gain=1, offset_x=0):
 6      return ((np.tanh(((x+offset_x)*gain)/2)+1)/2)
 7
 8  def color_bar_rgb(x):
 9      gain = 10
10      offset_x= 0.2
11      offset_green = 0.6
12      x = (x * 2) - 1
13      red = sigmoid(x, gain, -1*offset_x)
14      blue = 1-sigmoid(x, gain, offset_x)
15      green = sigmoid(x, gain, offset_green) + (1-sigmoid(x,gain,-1*offset_green))
16      green = green - 1.0
17      return [blue * 255,green * 255,red * 255]
18
19  def _get_color(value):
20      t = (value + ABS_LATENT_MAX_VALUE) / (ABS_LATENT_MAX_VALUE * 2.0)
21      color = color_bar_rgb(t)
22      return color
23
24  def create_color_panel(latent_spaces):
25      images = []
26      for z in latent_spaces:
27          p = np.zeros((PANEL_HEIGHT, PANEL_WIDTH, 3))
28          color = _get_color(z)
29          p += color[::-1]
30          p = np.clip(p, 0, 255)
31          images.append(p)
32      panel = np.concatenate(images, axis=1)
33      return panel
```

建立GUI

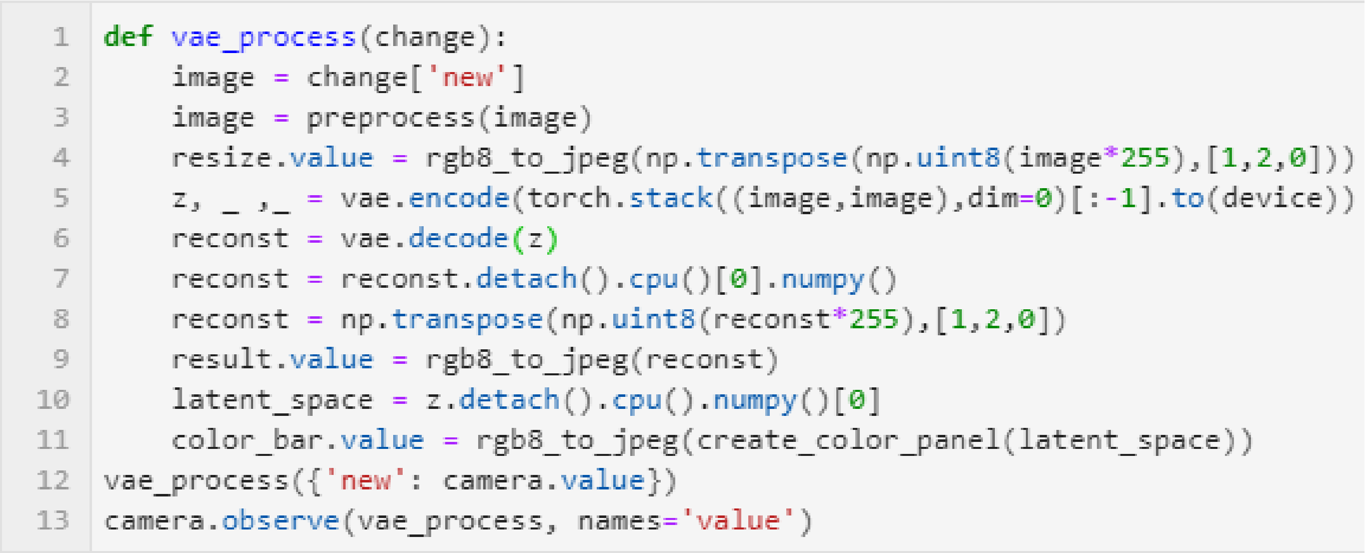


```
[8]: image = widgets.Image(format='jpeg', width=320, height=240)
     resize = widgets.Image(format='jpeg', width=160, height=80)
     result = widgets.Image(format='jpeg', width=160, height=80)
     camera_link = traitlets.dlink((camera,'value'), (image,'value'), transform=bgr8_to_jpeg)
     color_bar = widgets.Image(format='jpeg', width=32*PANEL_WIDTH, height=10*PANEL_HEIGHT)
     display(image)
     display(widgets.HBox([resize,result]))
     display(color_bar)
```

- [1]原始影像
- [2]重新設定尺寸的原始影像
- [3]模型重新產生的影像
- [5]設定顏色條
- [6]~[8]顯示[1]~[5]建立的影像視窗
- 需要執行接下來的程式來訂閱相機資訊

模型產生影像

- 利用訓練好的模型產生影像，並訂閱相機資訊。



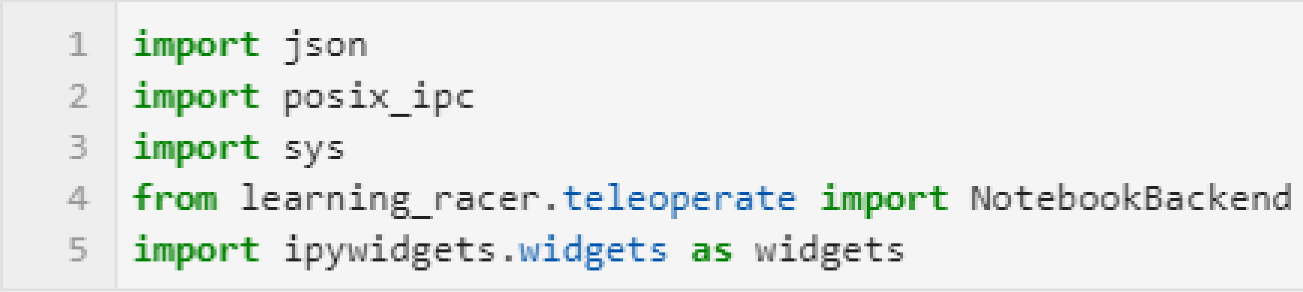
```python
def vae_process(change):
    image = change['new']
    image = preprocess(image)
    resize.value = rgb8_to_jpeg(np.transpose(np.uint8(image*255),[1,2,0]))
    z, _ ,_ = vae.encode(torch.stack((image,image),dim=0)[:-1].to(device))
    reconst = vae.decode(z)
    reconst = reconst.detach().cpu()[0].numpy()
    reconst = np.transpose(np.uint8(reconst*255),[1,2,0])
    result.value = rgb8_to_jpeg(reconst)
    latent_space = z.detach().cpu().numpy()[0]
    color_bar.value = rgb8_to_jpeg(create_color_panel(latent_space))
vae_process({'new': camera.value})
camera.observe(vae_process, names='value')
```

Start learning

user_interface_without_gamepad.ipynb

引用函示庫

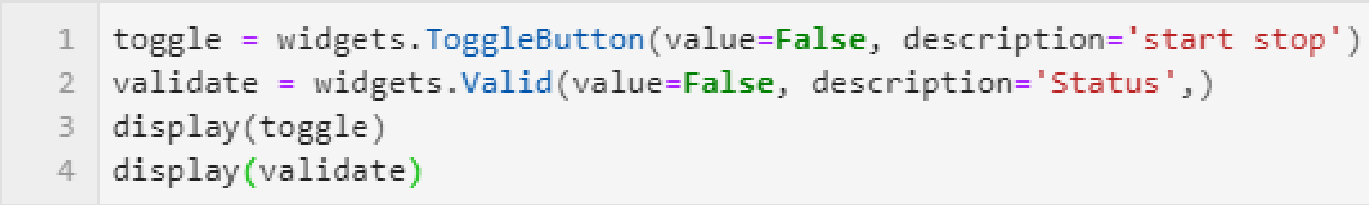
- 檔案路徑: airc-rl-agent/notebooks/user_interface_without_gamepad.ipynb
- [1]使用json來編碼與解碼
- [2]用於unix、linux的通訊標準
- [3]用於系統環境的建置
- [4]自訂義用於通訊的函示庫，包含posix的通訊協定
- [5]提供功能元件，例如:滑桿、顯示影片等等。



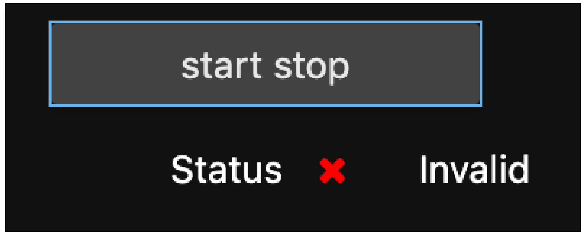
```python
import json
import posix_ipc
import sys
from learning_racer.teleoperate import NotebookBackend
import ipywidgets.widgets as widgets
```

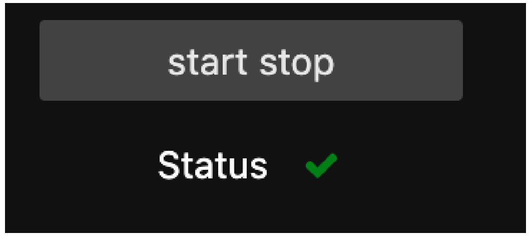
設定學習狀態的按鈕

- [1]定義bool值的按鈕，預設為False。
- [2]表示狀態的元件。當學習完成時，狀態(Status)會變成綠勾勾，表示可以繼續下一次的學習。
- [3]~[4]顯示按鈕與狀態框。

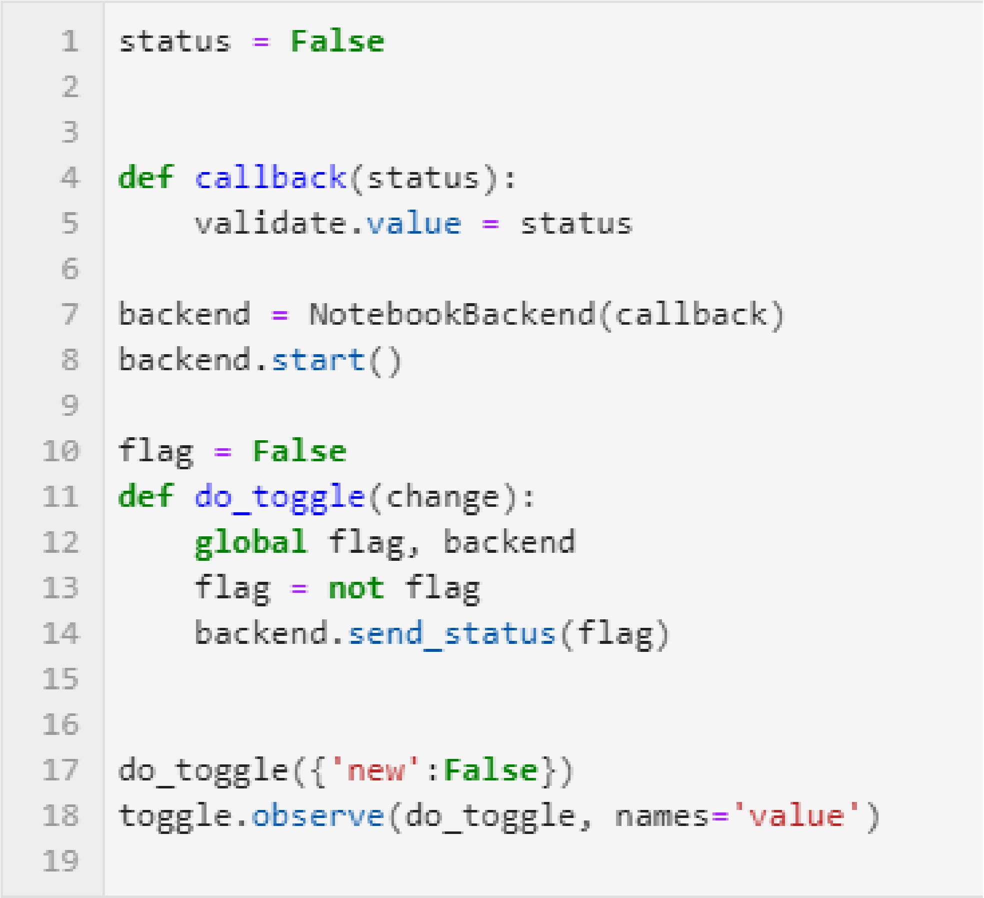


```
1  toggle = widgets.ToggleButton(value=False, description='start stop')
2  validate = widgets.Valid(value=False, description='Status',)
3  display(toggle)
4  display(validate)
```





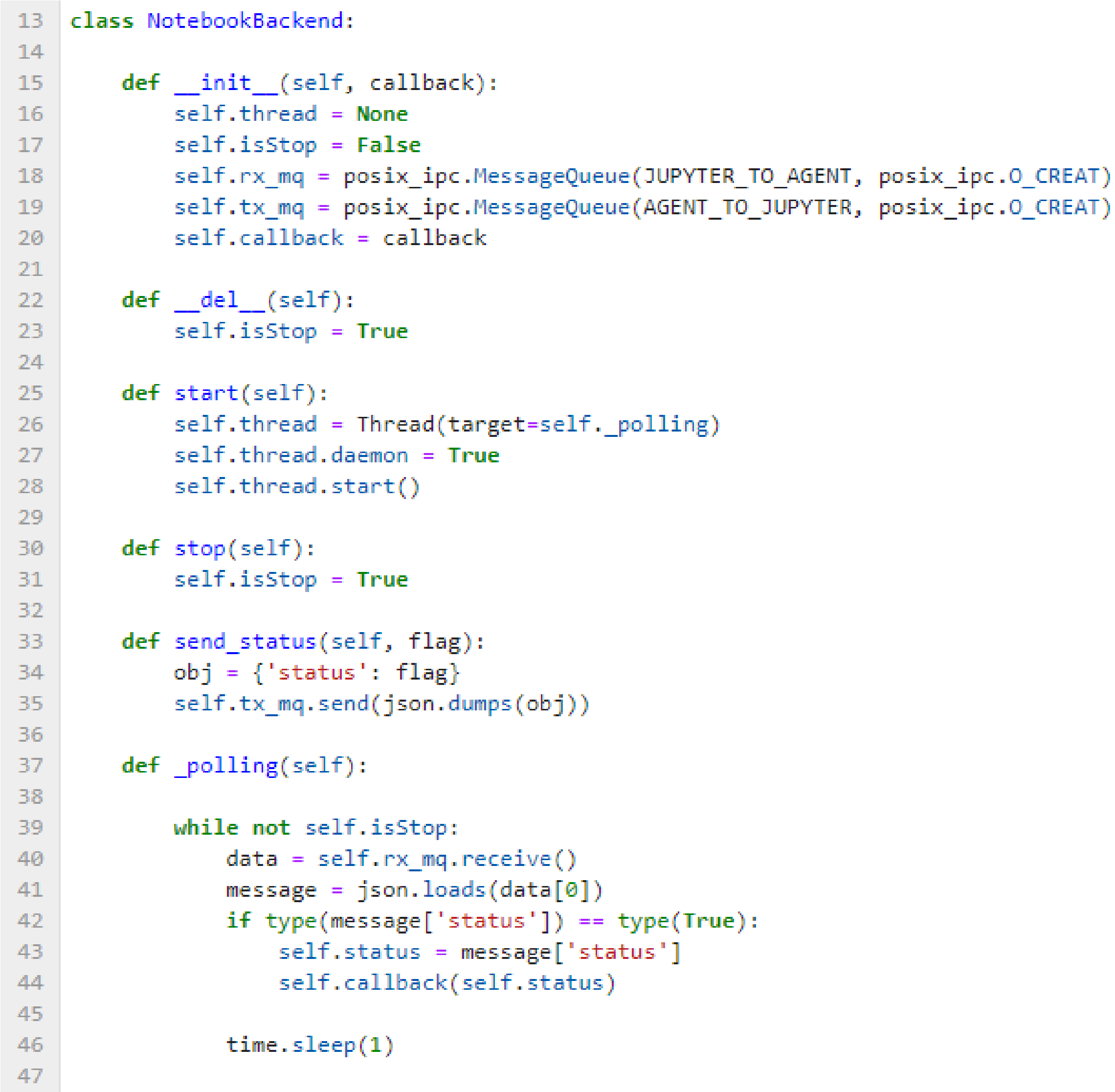
設定按鈕的callback



```
 1  status = False
 2
 3
 4  def callback(status):
 5      validate.value = status
 6
 7  backend = NotebookBackend(callback)
 8  backend.start()
 9
10  flag = False
11  def do_toggle(change):
12      global flag, backend
13      flag = not flag
14      backend.send_status(flag)
15
16
17  do_toggle({'new':False})
18  toggle.observe(do_toggle, names='value')
19
```

- [1]預設狀態為False。
- [4][5]回傳狀態。
- [7]給予NotebookBackend狀態。
- [8]啟動執行緒。
- [10]~[14]定義觸發事件，定義全域變數，flag設為相反，傳送狀態給backend。
- [17]初始化do_toggle。
- [18]按鈕訂閱do_toggle function。

learning_racer/teleoperate/message_queue.py



```
13  class NotebookBackend:
14
15      def __init__(self, callback):
16          self.thread = None
17          self.isStop = False
18          self.rx_mq = posix_ipc.MessageQueue(JUPYTER_TO_AGENT, posix_ipc.O_CREAT)
19          self.tx_mq = posix_ipc.MessageQueue(AGENT_TO_JUPYTER, posix_ipc.O_CREAT)
20          self.callback = callback
21
22      def __del__(self):
23          self.isStop = True
24
25      def start(self):
26          self.thread = Thread(target=self._polling)
27          self.thread.daemon = True
28          self.thread.start()
29
30      def stop(self):
31          self.isStop = True
32
33      def send_status(self, flag):
34          obj = {'status': flag}
35          self.tx_mq.send(json.dumps(obj))
36
37      def _polling(self):
38
39          while not self.isStop:
40              data = self.rx_mq.receive()
41              message = json.loads(data[0])
42              if type(message['status']) == type(True):
43                  self.status = message['status']
44                  self.callback(self.status)
45
46              time.sleep(1)
47
```

[26]target為新執行緒(thread)所要實作的function。

[27]將執行緒(thread)設定為daemon。Daemon thread是一種在背景執行的執行緒。

[28]啟動thread。

[37]~[46]當狀態不等於isStop時，rx接收資料，使用json格式讀取出訊息。當訊息狀態等於True時，將狀態等於訊息狀態，回傳狀態(True)。

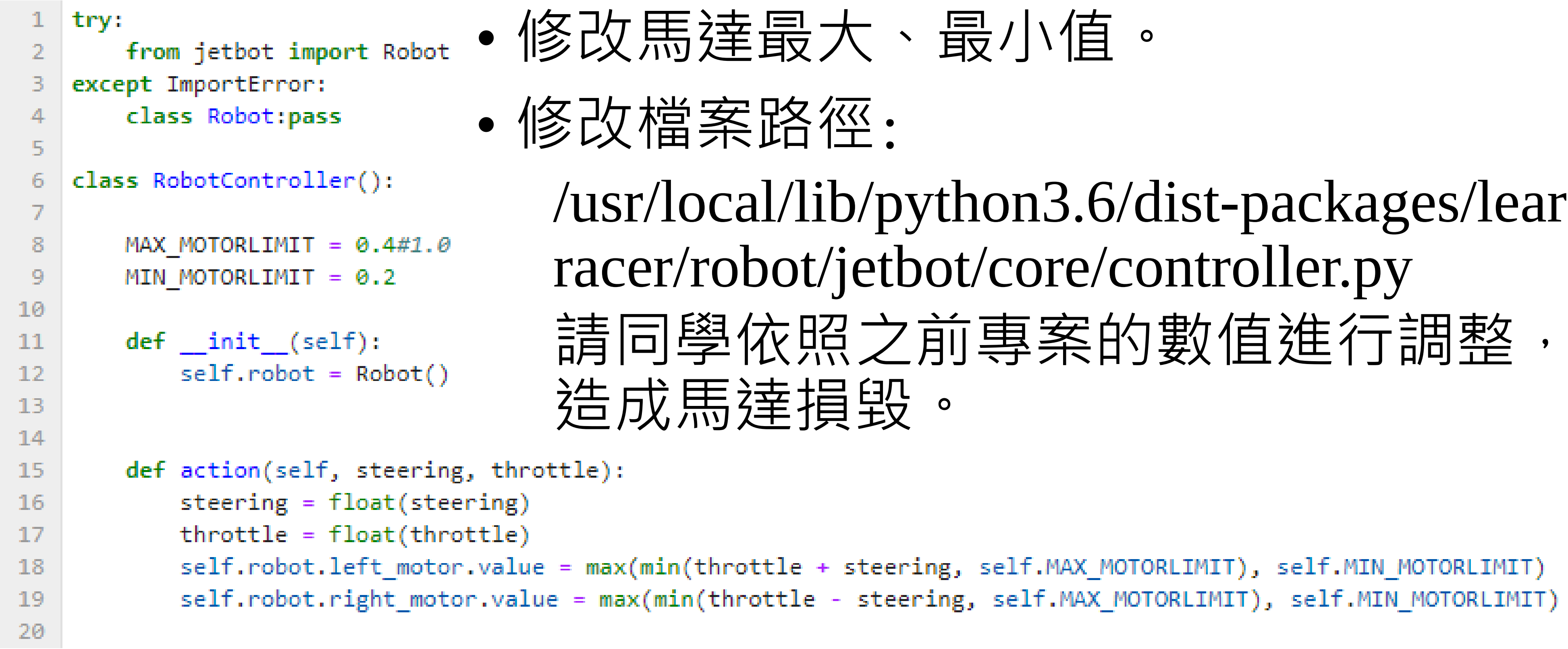
[33]~[35]使用tx傳送str格式的狀態

修改馬達參數

- 修改馬達最大、最小值。
- 修改檔案路徑:

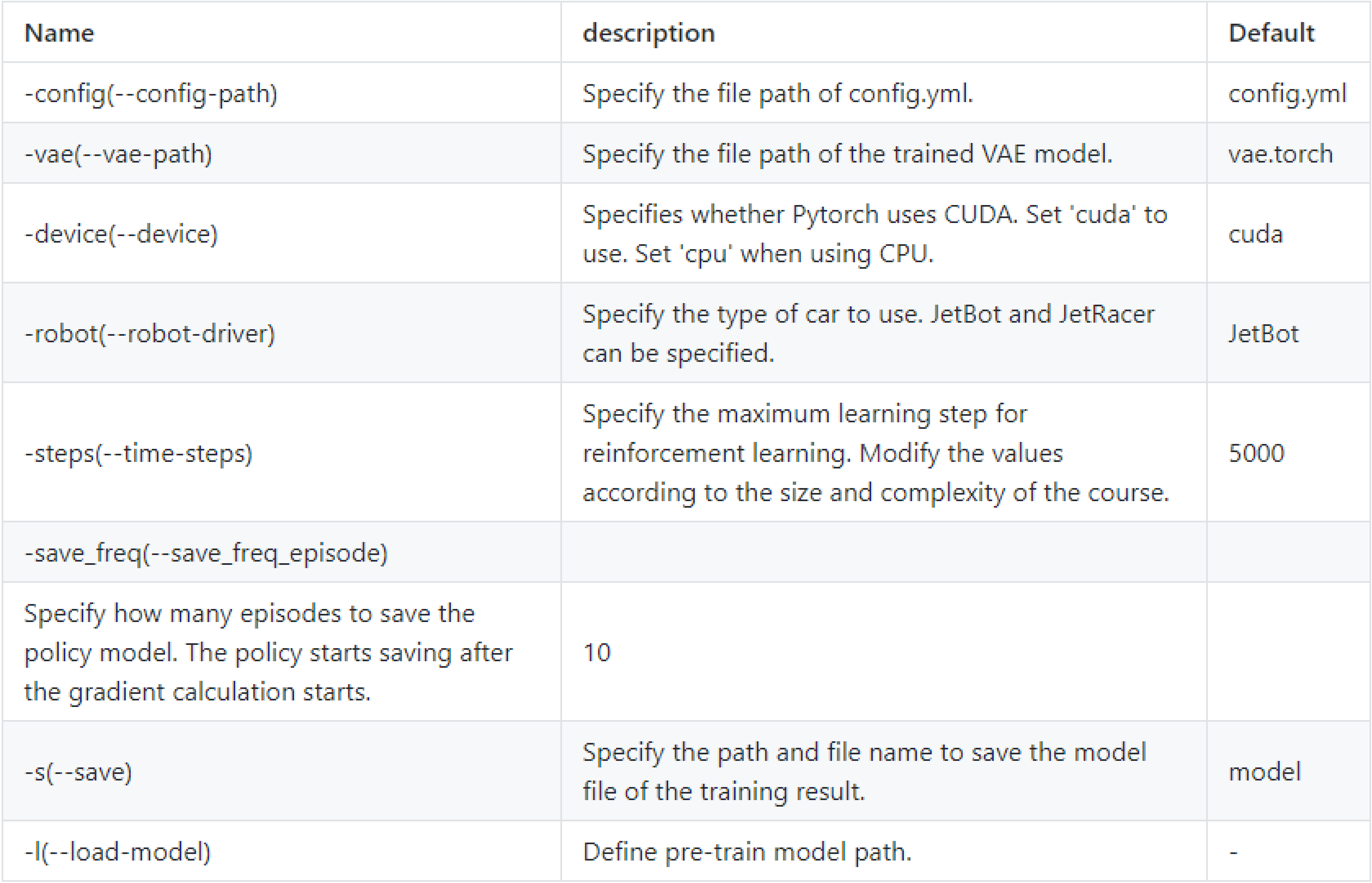
  /usr/local/lib/python3.6/dist-packages/learning-racer/robot/jetbot/core/controller.py

  請同學依照之前專案的數值進行調整，避免過大的輸出造成馬達損毀。



```python
try:
    from jetbot import Robot
except ImportError:
    class Robot:pass

class RobotController():

    MAX_MOTORLIMIT = 0.4#1.0
    MIN_MOTORLIMIT = 0.2

    def __init__(self):
        self.robot = Robot()


    def action(self, steering, throttle):
        steering = float(steering)
        throttle = float(throttle)
        self.robot.left_motor.value = max(min(throttle + steering, self.MAX_MOTORLIMIT), self.MIN_MOTORLIMIT)
        self.robot.right_motor.value = max(min(throttle - steering, self.MAX_MOTORLIMIT), self.MIN_MOTORLIMIT)

```

訓練階段可用參數

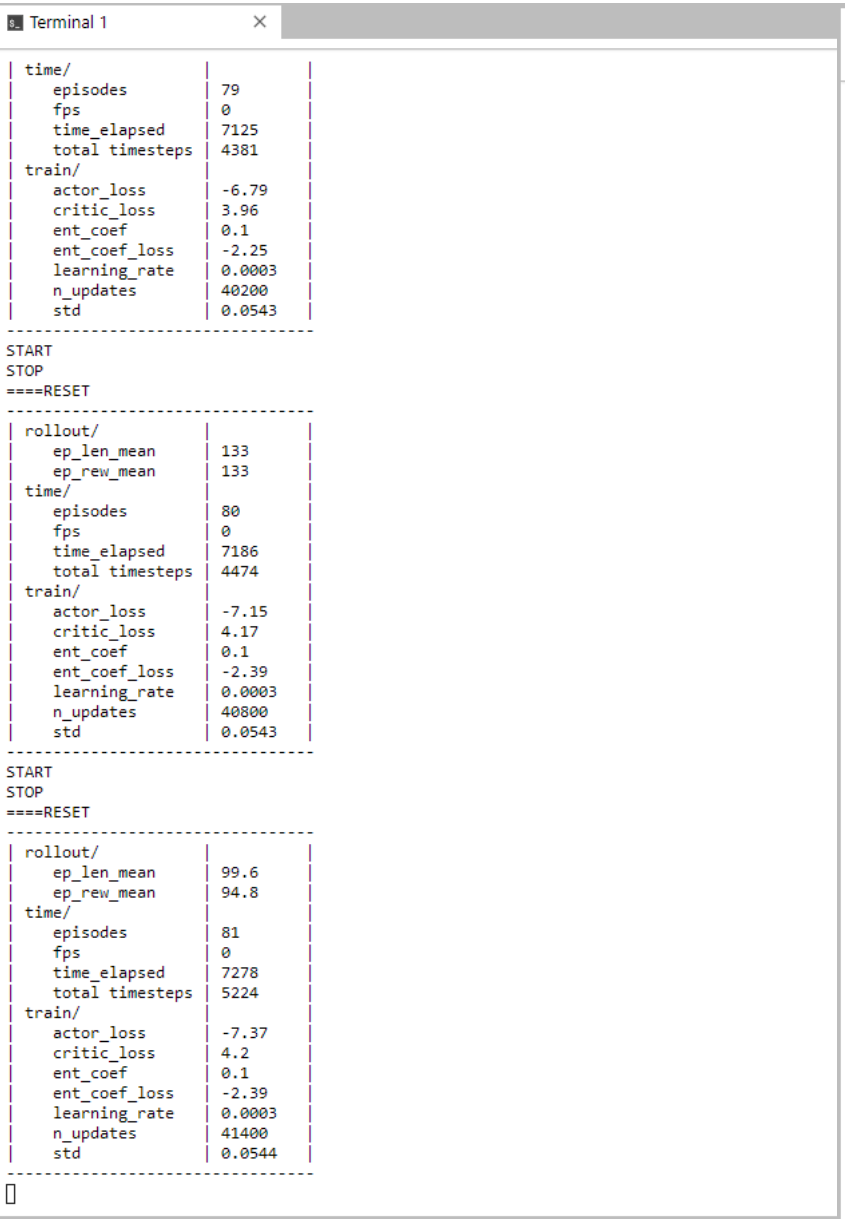


Name	description	Default
-config(--config-path)	Specify the file path of config.yml.	config.yml
-vae(--vae-path)	Specify the file path of the trained VAE model.	vae.torch
-device(--device)	Specifies whether Pytorch uses CUDA. Set 'cuda' to use. Set 'cpu' when using CPU.	cuda
-robot(--robot-driver)	Specify the type of car to use. JetBot and JetRacer can be specified.	JetBot
-steps(--time-steps)	Specify the maximum learning step for reinforcement learning. Modify the values according to the size and complexity of the course.	5000
-save_freq(--save_freq_episode)		
Specify how many episodes to save the policy model. The policy starts saving after the gradient calculation starts.	10	
-s(--save)	Specify the path and file name to save the model file of the training result.	model
-l(--load-model)	Define pre-train model path.	-

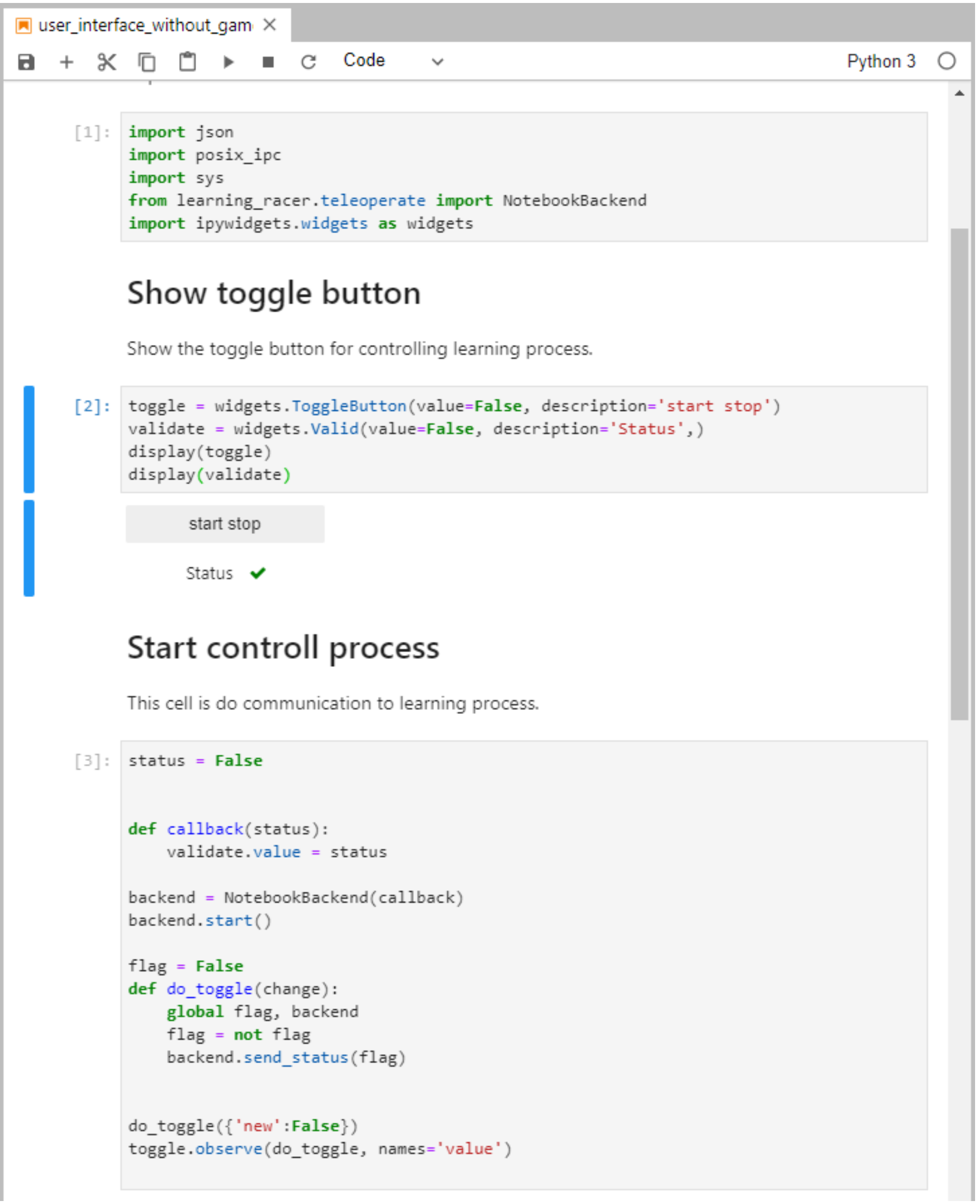
開始學習

- On terminal:
  - $ cd ~/airc-rl-agent/notebooks/utility/jetbot
  - $ racer train -robot jetbot -config ../../../config.yml -vae vae.torch -device cuda -robot jetbot -steps 5000 -save_freq 10 -s modelname
- 指令解析:
  - 執行racer
  - train 表示在學習模式
  - -config 在airc-rl-agent路徑下的config.yml(在這邊使用相對路徑)
  - -vae 訓練完成的模型
  - -device 使用cuda核心
  - -robot 本自走車為jetbot
  - -steps 指定增強式學習的最大學習次數。根據訓練集的大小和複雜程度修改。
  - -save_freq 學習多少次儲存一次模型
  - -s 儲存在model_log的模型檔案名稱，需自行填上名稱。
  - -l 可以選擇要讀取之前訓練的模型檔案名稱，使用範例: -l modelanme_xxx_steps

開始學習



```
| time/                |          |
|    episodes          | 79       |
|    fps               | 0        |
|    time_elapsed      | 7125     |
|    total timesteps   | 4381     |
| train/               |          |
|    actor_loss        | -6.79    |
|    critic_loss       | 3.96     |
|    ent_coef          | 0.1      |
|    ent_coef_loss     | -2.25    |
|    learning_rate     | 0.0003   |
|    n_updates         | 40200    |
|    std               | 0.0543   |
-------------------------------------
START
STOP
====RESET
-------------------------------------
| rollout/             |          |
|    ep_len_mean       | 133      |
|    ep_rew_mean       | 133      |
| time/                |          |
|    episodes          | 80       |
|    fps               | 0        |
|    time_elapsed      | 7186     |
|    total timesteps   | 4474     |
| train/               |          |
|    actor_loss        | -7.15    |
|    critic_loss       | 4.17     |
|    ent_coef          | 0.1      |
|    ent_coef_loss     | -2.39    |
|    learning_rate     | 0.0003   |
|    n_updates         | 40800    |
|    std               | 0.0543   |
-------------------------------------
START
STOP
====RESET
-------------------------------------
| rollout/             |          |
|    ep_len_mean       | 99.6     |
|    ep_rew_mean       | 94.8     |
| time/                |          |
|    episodes          | 81       |
|    fps               | 0        |
|    time_elapsed      | 7278     |
|    total timesteps   | 5224     |
| train/               |          |
|    actor_loss        | -7.37    |
|    critic_loss       | 4.2      |
|    ent_coef          | 0.1      |
|    ent_coef_loss     | -2.39    |
|    learning_rate     | 0.0003   |
|    n_updates         | 41400    |
|    std               | 0.0544   |
-------------------------------------
```



```
[1]: import json
     import posix_ipc
     import sys
     from learning_racer.teleoperate import NotebookBackend
     import ipywidgets.widgets as widgets
```

Show toggle button

Show the toggle button for controlling learning process.

```
[2]: toggle = widgets.ToggleButton(value=False, description='start stop')
     validate = widgets.Valid(value=False, description='Status',)
     display(toggle)
     display(validate)
```

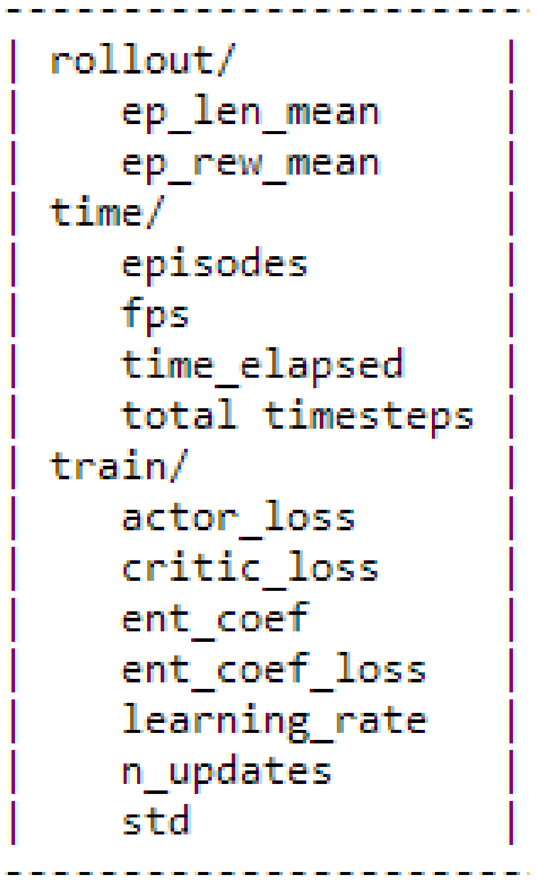
Start controll process

This cell is do communication to learning process.

```
[3]: status = False


     def callback(status):
         validate.value = status

     backend = NotebookBackend(callback)
     backend.start()

     flag = False
     def do_toggle(change):
         global flag, backend
         flag = not flag
         backend.send_status(flag)


     do_toggle({'new':False})
     toggle.observe(do_toggle, names='value')
```

1. 先執行user_interface_without_gamepad.ipynb
2. 再執行racer.py
3. 每次按下按鈕Jetbot會開始移動，當Jetbot走出道路時按下按鈕，讓模型儲存該次的reward，學習直到Jetbot能夠在道路中行駛。當狀態從紅色叉叉轉為綠色勾勾表示模型訓練完成。
4. 如果要停止學習，再terminal按下ctrl+c即可。

學習階段的參數介紹



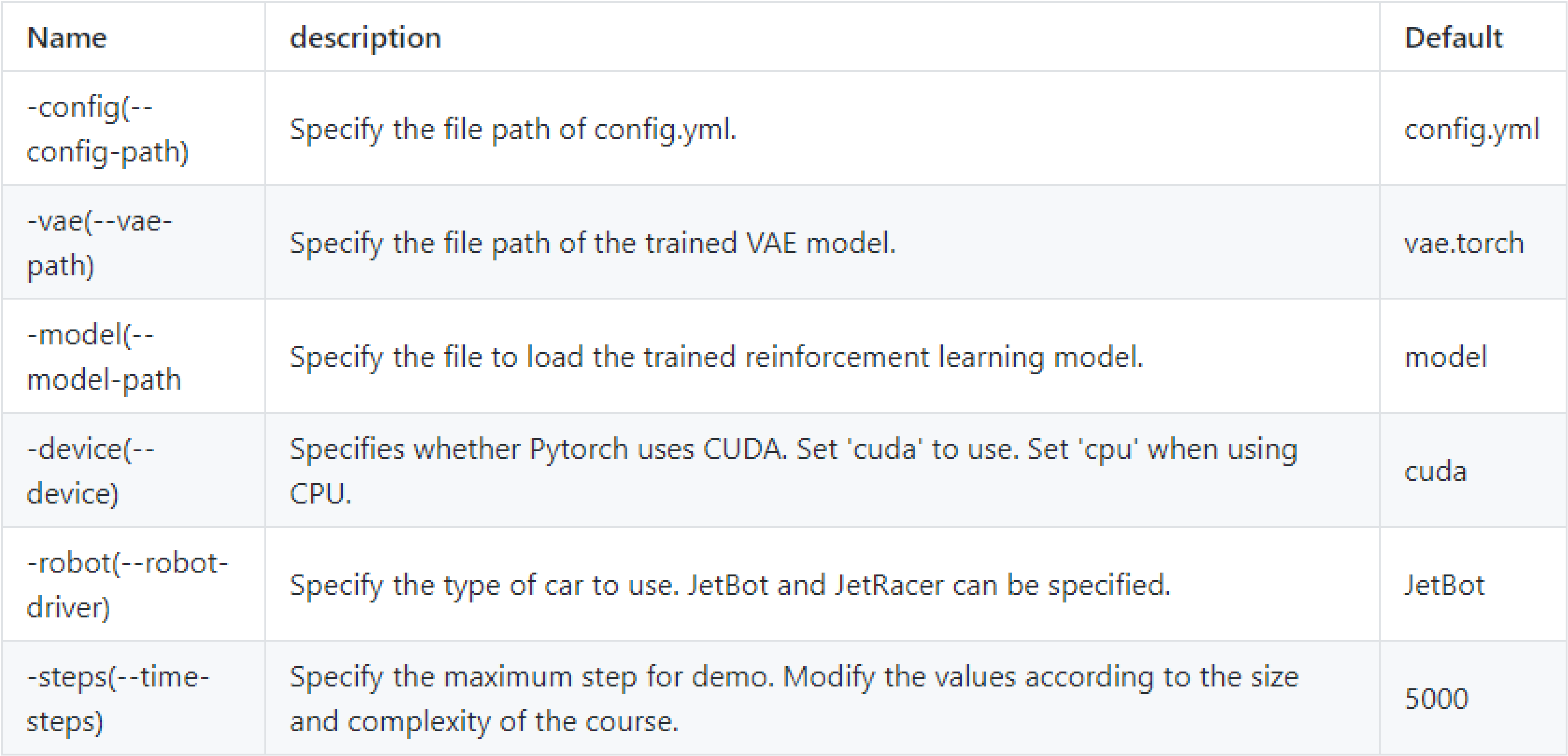
```
---------------------------
| rollout/                |
|    ep_len_mean          |
|    ep_rew_mean          |
| time/                   |
|    episodes             |
|    fps                  |
|    time_elapsed         |
|    total timesteps      |
| train/                  |
|    actor_loss           |
|    critic_loss          |
|    ent_coef             |
|    ent_coef_loss        |
|    learning_rate        |
|    n_updates            |
|    std                  |
---------------------------
```

ep_len_mean:表示本次模型展開地的水平長度
ep_rew_mean:表示本次訓練的獎勵
episodes:訓練的次數
time_elapsed:實際執行時間(ms)
total timesteps:在任何環境下程式總共要執行的次數
actor_loss : 依照Critic給予的數值，指引policy function變數Θ進行更新
critic_loss : 更新action-value函數的變數w

$$\nabla_\theta J(\theta) \approx E_{\pi\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)$$

DEMO階段可用參數

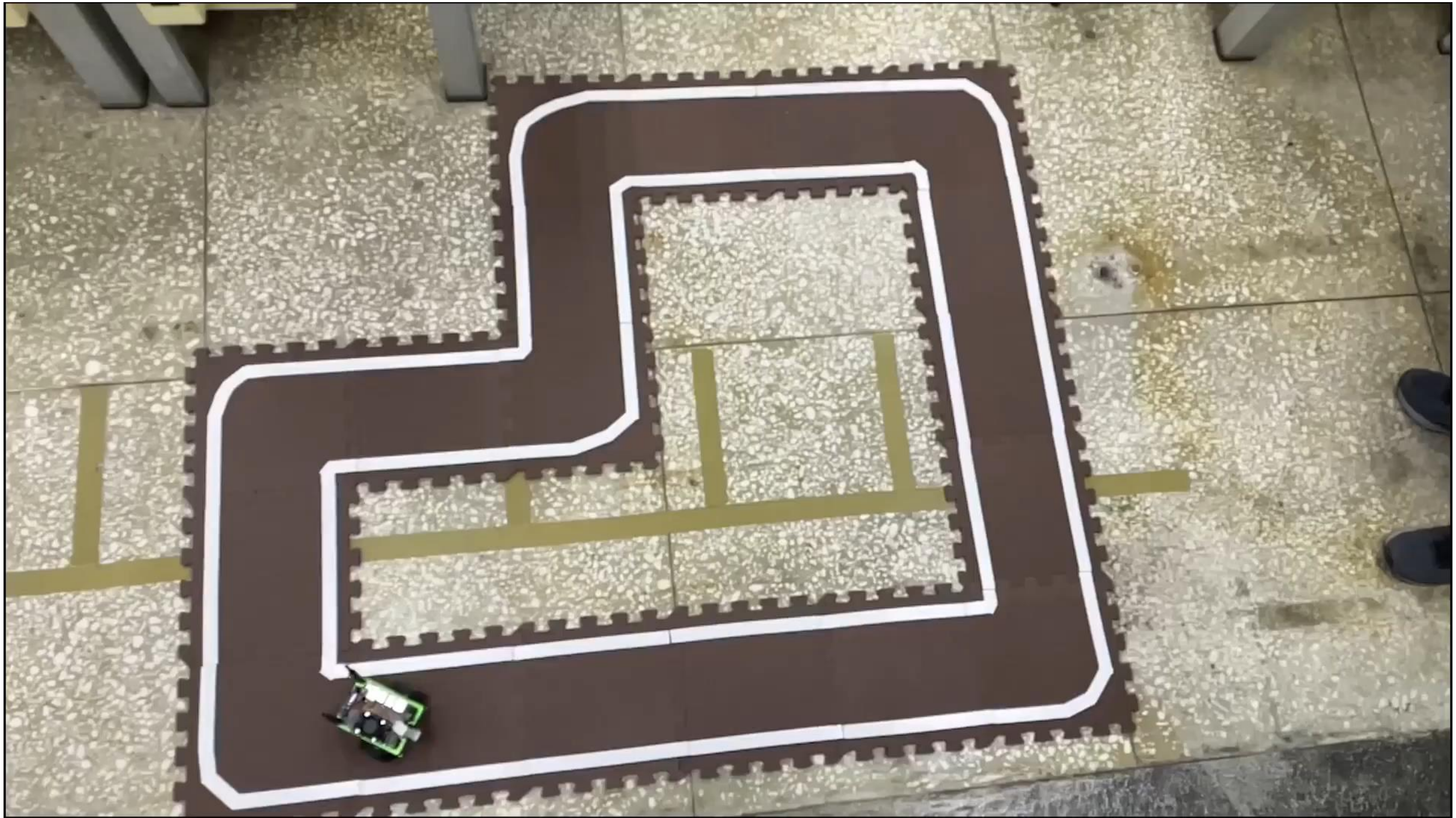


Name	description	Default
-config(--config-path)	Specify the file path of config.yml.	config.yml
-vae(--vae-path)	Specify the file path of the trained VAE model.	vae.torch
-model(--model-path	Specify the file to load the trained reinforcement learning model.	model
-device(--device)	Specifies whether Pytorch uses CUDA. Set 'cuda' to use. Set 'cpu' when using CPU.	cuda
-robot(--robot-driver)	Specify the type of car to use. JetBot and JetRacer can be specified.	JetBot
-steps(--time-steps)	Specify the maximum step for demo. Modify the values according to the size and complexity of the course.	5000

DEMO

- $ racer demo -robot jetbot -step 1000 -model modelname
- 指令解析:
  - Demo 為展示模式
  - -robot 本教學使用的自走車為jetbot
  - -step 可以指定要展示的模型訓練次數
  - -model 填上要讀取的模型檔案名稱

DEMO 實際成果



小組報告格式規定

- 專案情境
  - 小組所討論出來的議題，並簡明扼要描述議題的情境。
- 定義問題
  - 將議題中的問題定義出來，並收斂問題方向。
- 方案構思
  - 簡單描述如何解決定義好的問題，並預計使用的技術。
- 解決方法
  - 說明實際上如何完成此議題的方案構思。
- 分工
  - 說明小組成員分工內容與比例。

個人報告內容

- 個人報告內容須要有以下內容：
    - 你一開始所提出的議題是?你的議題是否有被選為小組議題候選?
    - 你在小組議題中，提出了那些問題與解決方案?是否有被小組接受?
        - 如個人所提出的方案沒被接受，是因為那些原因?
        - 為了這個議題，你去找了那些資料?你是如何分析找到的資料?
        - 其他小組成員所提出的提議有哪些?而你對於其他人的提議意見如何?
    - 在小組決定小組議題過程中，你對於小組最後提出的議題討論是否能接受?接受理由為何?不接受理由為何?
    - 你是否能接受最後的議題與方案?如接受請說明接受與否的理由?
    - 本次專案個人的心得
    - 本次你認為小組成員的貢獻比例及理由

專案繳交規則

- 專案成果實體驗收請於111/1/7課程結束前找助教檢查
- 小組報告繳交期限:111/1/7 23:59(以I學園上傳時間為基準)
- 個人報告繳交期限:111/1/7 23:59(以I學園上傳時間為基準)
- 補交規則
  - 超過正常繳交期限兩周內成績**打8折**
  - 超過正常期限兩周後不接受補交