Donkey Car 巡路實作

line 1: 李昱昇
line 2: 元智大學
(of Affiliation)
line 3: 電機乙組
(of Affiliation)
line 4: 桃園市,台灣
line 5: redick7761132@gmail.com

line 1: 王宜婕
line 2: 元智大學
(of Affiliation)
line 3: 電機乙組
(of Affiliation)
line 4: 桃園市,台灣
line 5: michelle379123@gmail.com

line 1: 張友安
line 2: 元智大學
(of Affiliation)
line 3: 電機乙組
(of Affiliation)
line 4: 桃園市,台灣
line 5: honoka0803by16@gmail.com

line 1: 張博崴
line 2: 元智大學
(of Affiliation)
line 3: 電機乙組
(of Affiliation)
line 4: 桃園市,台灣
line 5: chjimmy20@gmail.com

line 1: 劉語恩
line 2: 元智大學
(of Affiliation)
line 3: 電機乙組
(of Affiliation)
line 4: 桃園市,台灣
line 5: s1070430@mail.yzu.edu.tw

***Abstract*— 本次專案以開源的 Donkey Car 為基礎，同時我們結合了 Opencv、Tensorflow 與 keras 進行訓練；使用 Donkey Car 提供的官方程式進行修改和新增，藉由嘗試各式各樣的神經模型與自定義神經網路架構，並在虛擬環境中進行模擬，選擇出效果最佳的模型，使得 Donkey Car 具備最佳的巡路功能，達成自動駕駛之初衷。**

Keywords*— *Donkey Car, 自定義神經網路架構, 自動駕駛

I. 簡介

Donkey Car 驢車起源於美國加州，是一個開源的自走車專案，以 Raspberry Pi 硬體及 Python 程式作為軟體基礎，使用機器學習和電腦視覺技術以驅動遙控車。

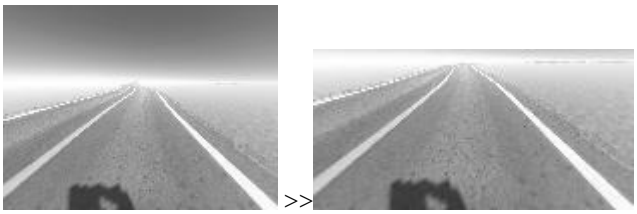
本次專案所使用的 Donkey Car 硬體包括 Raspberry Pi、PWM訊號生成器、馬達控制板、馬達和魚眼相機等。而軟體部分則包括樹莓派的 linux 作業系統加上 Tensorflow 神經網路。

由上述可知，魚眼相機就是本專案實體 Donkey Car 唯一的感測器，在操作 Donkey Car 的過程中，我們可以透過此相機蒐集訓練資料，並使用樹莓派處理資料，之後進行模型選擇與調整、實機訓練、測試驗證等，執行機器學習的完整流程；最後透過車輛的自駕情形，評估實際訓練出來的成果。但由於本次專案僅需要透過模擬軟體進行訓練，因此透過模擬器我們其實也可以輕鬆新增感測器，且本專案也將著重在機器學習的部分。

II. 實驗方法

A. 圖片預處理部分

圖片尺寸是 120*160，因為訓練成效取決於路面狀況，為盡量避免將背景納入模型運算，故將圖片上面 40 pixels 裁掉，做完切割的圖片尺寸變為 80*160，最後將圖片從 RGB 三通道轉為灰階。除此之外嘗試使用經過 Canny 邊緣偵測的圖片進行訓練，試圖取得路線邊緣，原先預期會取得較好成效，但與未使用得到的成效相似。所以最後決定不使用在圖片上。



(圖一 : 裁切前後比較圖)

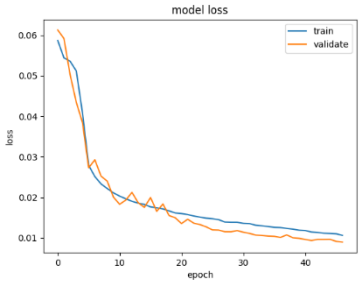
B. 神經網路部分

將進行過預處理的圖片輸入透過 time distribute轉為類似於連續的輸入(時間序列)，以放到後層的 LSTM 進行訓練，並藉由 Dropout 防止過擬合，最後將模型結果輸出。

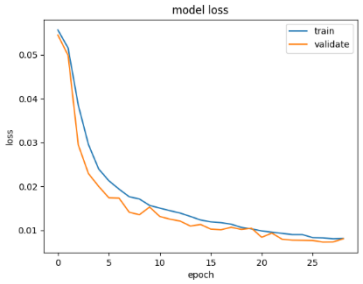
原先我們有使用 transfer learning 來 fine-tuning 模型結果，並分別嘗試了 VGG16、RESNET50 等模型架構。但與原廠模型相比，都未獲得明顯成效，故最後決定使用較為簡單的模型架構。

C. 優化器與損失函數部分

我們使用過 Adam、SGD、RMSProp 等優化器進行嘗試，最後我們選擇使用收斂速度較快及效果較佳的 Adam 作為最佳化。損失函數在經過嘗試 RMSE 和 MSE 後，發現 RMSE 雖然收斂速度比較快，可是 loss 反而比 MSE 還要來的高一點，但落差並沒有到很大，推測可能是 RMSE 落到區域極值所造成的影響；在評估效能與準確率後，選擇使用 RMSE 作為損失函數。



(a) MSE(final lose:0.0072)



(b) RMSE(final lose:0.0081)

(圖二：MSE、RMSE 的 loss 比較)

III. 提出的機器學習模型

從下圖的 Input Layer 中可以看出，shape 是(None, 3, 80, 160)， (80, 160)為圖片長寬，而其中 3 並非為常見 RGB 圖片的三通道，而是指包含了三張的灰階圖片，目的是為了讓輸入的圖片具有時間關係。在我們進行多次嘗試後，發現一次三張圖片有較好之效果。而為了實現上述目的，我們透過 time distribute 將其轉為類似於連續的輸入(時間序列) ，以放到後層的 LSTM 進行訓練，並藉由 Dropout 適度的放棄部分神經元以防止過擬合，最後將模型結果分別輸出成 angle 和 throttle。

Layer (type)	Output Shape	Param #	Connected to
img_in (InputLayer)	[(None, 3, 80, 160,	0	
time_distributed (TimeDistribut	(None, 3, 38, 78, 24	624	img_in[0][0]
time_distributed_1 (TimeDistrib	(None, 3, 38, 78, 24	0	time_distributed[0][0]
time_distributed_2 (TimeDistrib	(None, 3, 17, 37, 32	19232	time_distributed_1[0][0]
time_distributed_3 (TimeDistrib	(None, 3, 17, 37, 32	0	time_distributed_2[0][0]
time_distributed_4 (TimeDistrib	(None, 3, 13, 33, 64	51264	time_distributed_3[0][0]
time_distributed_5 (TimeDistrib	(None, 3, 13, 33, 64	0	time_distributed_4[0][0]
time_distributed_6 (TimeDistrib	(None, 3, 9, 29, 64)	102464	time_distributed_5[0][0]
time_distributed_7 (TimeDistrib	(None, 3, 9, 29, 64)	0	time_distributed_6[0][0]
time_distributed_8 (TimeDistrib	(None, 3, 3, 13, 64)	102464	time_distributed_7[0][0]
time_distributed_9 (TimeDistrib	(None, 3, 3, 13, 64)	0	time_distributed_8[0][0]
time_distributed_10 (TimeDistri	(None, 3, 1, 6, 64)	0	time_distributed_9[0][0]
time_distributed_11 (TimeDistri	(None, 3, 384)	0	time_distributed_10[0][0]
LSTM_seq (LSTM)	(None, 3, 100)	194000	time_distributed_11[0][0]
dropout_5 (Dropout)	(None, 3, 100)	0	LSTM_seq[0][0]
LSTM_fin (LSTM)	(None, 50)	30200	dropout_5[0][0]
dropout_6 (Dropout)	(None, 50)	0	LSTM_fin[0][0]
angle (Dense)	(None, 1)	51	dropout_6[0][0]
throttle (Dense)	(None, 1)	51	dropout_6[0][0]

Total params: 500,350
Trainable params: 500,350
Non-trainable params: 0

(圖三：自定義的神經網路架構)

IV. 效能評估

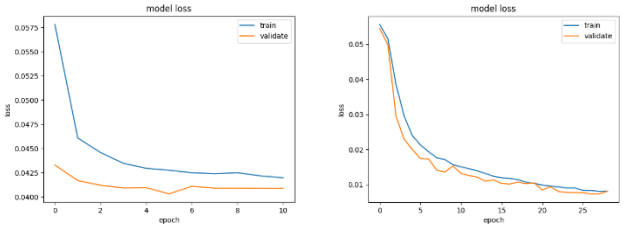
下面我們將比較官方模型架構和我們自定義的模型架構的差異將使用 val loss 圖做呈現及文字說明。

A. 預設模型無考慮時間性

官方預設的模型不會考慮到時間上的對比關係，因此在面臨到急轉彎時，便會因為速度調整不及而超出跑道；但是我們的自定義模型會考慮到前兩幀的瞬時速度，所以在面臨急轉彎時，會預先減速而不致超出彎道。詳細情形可以透過附錄內的影片連結，進行實際情況之觀看。

B. 當訓練資料數量大到一定比例時(圖中為 10000)

官方的模型架構就無法順利收斂， 而我們的模型即使訓練資料較為龐大也可以順利收斂。

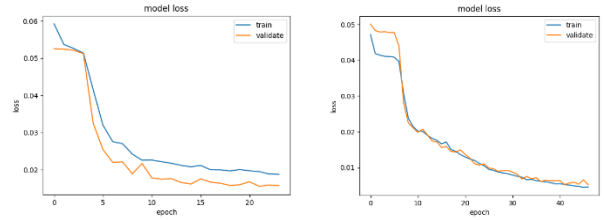


(a) 預設模型　　(b) 我們的模型

(圖四：loss 比較)

C. 當訓練資料數量不大時(圖中為 5000)

從 loss 上來看我們的模型雖只比官方模型略低一些，但實際上執行模擬時，能明顯的看出其差異，可參考附錄連結進行比較。



(a) 預設模型　　(b) 我們的模型

(圖五：loss 比較)

V. 結論

1. 在本次實驗中，我們還嘗試了使用 VGG-16、ResNet50 等常見的模型，但是實際效果並未比原來的模型有明顯的差異，且增加了許多訓練時間，我們認位是巡路功能並不需要太強大的模型即可達成，所以最後我們本次實驗最後以降低訓練及運算所需效能為目標。
2. 降低圖片通道數可大幅降低訓練所需時間，但會使得避障的成功率下降。
3. 訓練資料容易受到人為操作的誤差影響，為改善此問題，我們通過收集 donkey car 自駕時的數據，作為新一批的訓練資料。

VI. 附錄

A. *影片與權重檔連結*

https://drive.google.com/drive/folders/1kaE_blUOqgiUtAaqndxk83kNCW7BBvLK?usp=sharing

B. *資料收集與訓練模型*

資料收集與訓練模型的使用方法部分，跟原來方法一樣，分別為 python manage.py drive 與 python train.py，後者必須指定模型路徑跟圖片路徑，並在 type 指定"my"以使用我們自定義的模型架構，下列為參數說明

```
python train.py
    --model <model path>
    --tub <data path>
    --type <model type>

python manage.py drive
    --model <model path>
--type <model type>
```

C. *程式碼改動*

1. 將用來選擇自定義模型的程式碼

```
elif used_model_type == 'my':
        kl=Mymodel(input_shape=input_shape,
                seq_length=cfg.SEQUENCE_LENGTH)
```

加入於 donkeycar/donkeycar/utils.py 第 485 行後

2. 在 donkeycar/donkeycar/parts/keras.py 新增下列連結中的 class

https://drive.google.com/file/d/1ggWcE1YsI6kVj88I3hUKjdFKqvtDNlya/view

3. 將用來切割圖片的程式碼

```
class Imgcut():
    def __init__(self,x,y,w=None,h=None):
        self.x=x
        self.y=y
        self.w=w
        self.h=h
    def run(self,img_arr):
        if( self.w!=None and  self.h!=None):
            return img_arr[ self.y: self.y+ self.h,self.x: self.
x + self.w]
        else:
            return img_arr[ self.y:,  self.x:]
    def shutdown(self):
        Pass
```

加入於 donkeycar/donkeycar/parts/cv.py

4. 加入圖片預處理

```
gray = dkcv.ImgGreyscale()
cut = dkcv.Imgcut(x=0, y=40)
V.add(gray, inputs=['cam/image_array'],
outputs=['cam/image_array'], threaded=False)
V.add(cut, inputs=['cam/image_array'],
outputs=['cam/image_array'], threaded=False)
```

於 mycar/manage.py 188 行後

D. *Github 連結*

https://github.com/brian-liu-1070430/1092MachineLearning_Group3_finalReport

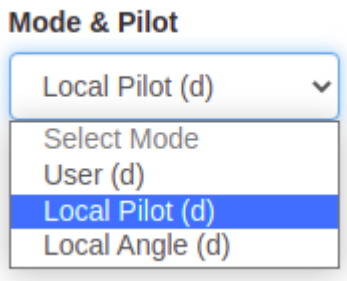
E. *程式使用步驟*

1. 從 github 上取得程式碼(內含 donkeycar 與 mycar 資料夾)
2. 使用 github 中的 donkeycar 資料夾，覆蓋掉原本環境中的 donkeycar 資料夾
3. 請至下列網址下載符合您作業系統之 donkey simulator，並解壓至 mycar 資料夾中

   https://github.com/tawnkramer/gym-donkeycar/releases
4. 啟動 donkeycar 環境，並於 mycar 資料夾中執行下列指令 ：

   ```
   python manage.py drive --model
   models/mypilot_rmae_final.h5 --type my
   ```
5. 開啟瀏覽器，進入下列網址：

   http://localhost:8887/drive
6. 於瀏覽器中直接操作 donkeycar，或是將畫面左側 Mode & Pilot 選至 Local Pilot(d)啟動自駕



(圖六：Mode & Pilot 選單)

7. 如需自行訓練模型請參考附錄資料收集與訓練模型如需訓練資料可至下列連結下載 ：

   https://drive.google.com/file/d/1rHC1DS5gQcKtYKGzAOTpHu2S9X-6WkCS/view?usp=sharing