2020 IIE Individual Research Project

CNN之環境景觀影像分類識別

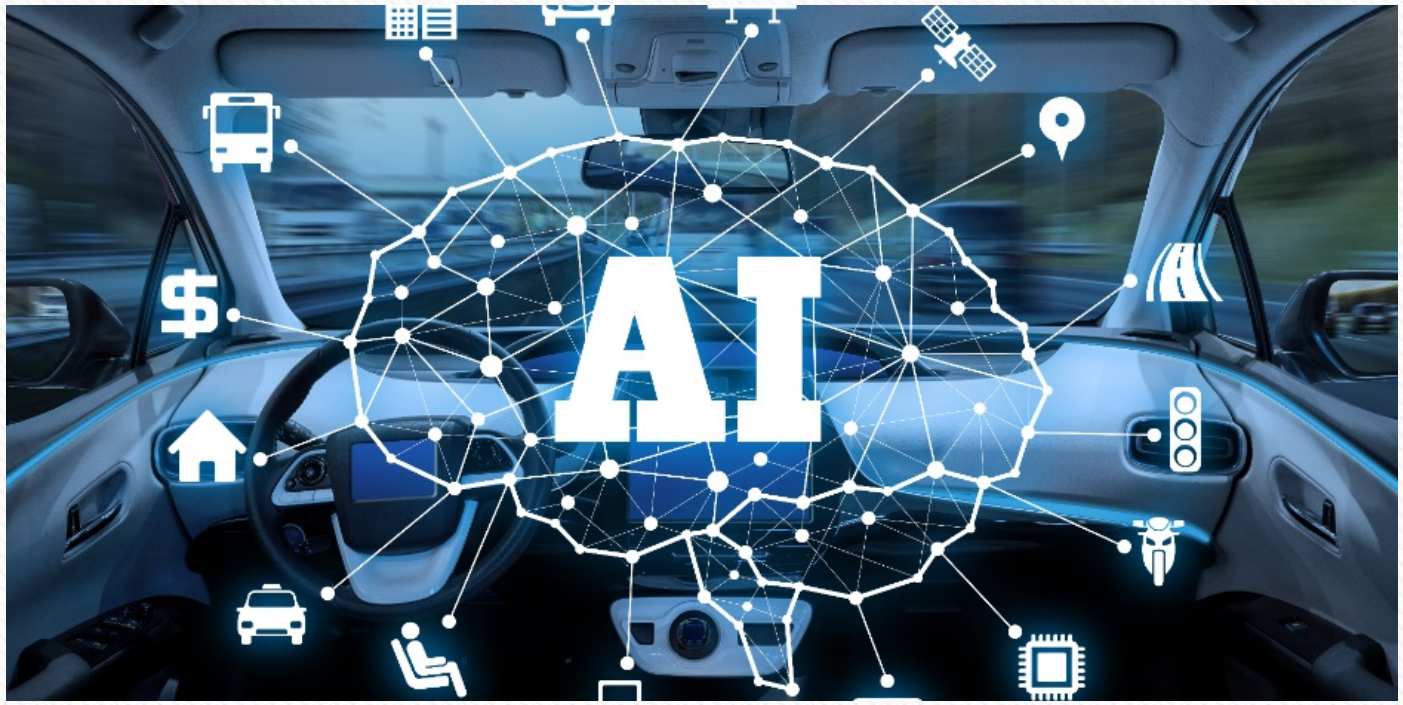
109034548 吳仲人

CONTENT

1 研究背景、動機

研究背景

隨著5G、大數據時代來臨，自駕車已成為近年來全球的顯學，各國政府、各大企業皆以發展自駕車為重要目標。



研究動機

該名特斯拉駕駛以為車輛會偵測到障礙物而降速或停駛，
但結果...

問題定義 5W1H

項目	內容
What	自駕車無法辨別某些特定場景
Where	可供車輛行駛之道路（如高速公路、快速道路、一般道路等）
Who	具自駕功能之車輛的駕駛人
When	當駕駛人行駛於道路上時
Why	使配備自駕功能的車輛在行車時更為安全
How	各類基本環境景觀影像的資料前處理及 CNN 模型訓練

2 資料整理

資料整理

資料整理的過程大致可以分為三個步驟：

- 資料集照片來源
- 資料前處理過程
- 以視覺化方式呈現

資料集照片來源

從 Kaggle 公開數據集蒐集環境景觀影像（建築物、森林、冰川、山、海、街道）共24335張。其中包含：

- seg_pred：未分類之環境景觀影像共7301張
- seg_train：已分類之環境景觀影像共14034張
- seg_test：已分類之環境景觀影像共3000張

資料前處理過程


```python
def get_images(directory):
    Images = []
    Labels = []    # 0 for Building , 1 for forest, 2 for glacier, 3 for mountain, 4 for Sea , 5 for Street
    label = 0

    for labels in os.listdir(directory): #Main Directory where each class label is present as folder name.
        if labels == 'glacier': #Folder contain Glacier Images get the '2' class label.
            label = 2
        elif labels == 'sea':
            label = 4
        elif labels == 'buildings':
            label = 0
        elif labels == 'forest':
            label = 1
        elif labels == 'street':
            label = 5
        elif labels == 'mountain':
            label = 3

        for image_file in os.listdir(directory+labels): #Extracting the file name of the image from Class Label folder
            image = cv2.imread(directory+labels+r'/'+image_file) #Reading the image (OpenCV)
            image = cv2.resize(image,(150,150)) #Resize the image, Some images are different sizes. (Resizing is very Important)
            Images.append(image)
            Labels.append(label)

    return shuffle(Images,Labels,random_state=817328462) #Shuffle the dataset you just prepared.

def get_classlabel(class_code):
    labels = {2:'glacier', 4:'sea', 0:'buildings', 1:'forest', 5:'street', 3:'mountain'}

    return labels[class_code]
```


street


sea


street


glacier


mountain


glacier


mountain

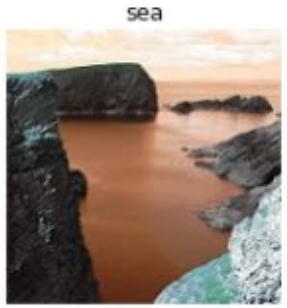

glacier


buildings

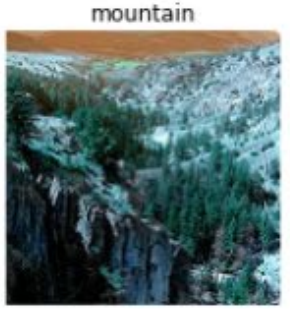

sea


forest


forest


sea

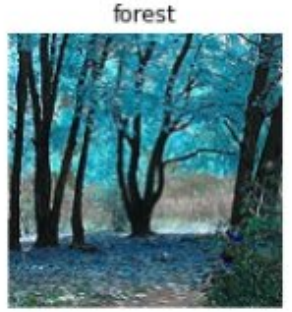

street

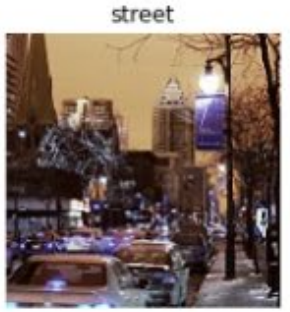

mountain


buildings


glacier


mountain


forest


street

3 模型訓練、驗證、測試

模型設定

CNN parameter	Value
Number of convolution layers	6
Filter size	3×3
Number of pooling layers	2
Pooling filter size	5×5
Number of fully connected layers	4
Activation function	relu
Regularization method	dropout
Classification function of the output layer	sigmoid
Loss function	sparse_categorical_crossentropy
Optimizer	adam
Learning rate	0.001
Epoch	10

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 200)     5600
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 180)     324180
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 29, 29, 180)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 180)       291780
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 140)       226940
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 100)       126100
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 21, 21, 50)        45050
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 50)          0
_________________________________________________________________
flatten (Flatten)            (None, 800)               0
_________________________________________________________________
dense (Dense)                (None, 180)               144180
_________________________________________________________________
dense_1 (Dense)              (None, 100)               18100
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 306
=================================================================
Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0
```

訓練模型

訓練集準確度為 64.58%，驗證集準確度 70.2%，發現不管是在訓練集或是驗證集資料都無法達到一定的準度。

```
trained = model.fit(Images, Labels, epochs=10, batch_size=32, validation_split=0.30)
```

```
Epoch 1/10
307/307 [==============================] - 91s 267ms/step - loss: 2.1572 - accuracy: 0.1674 - val_loss: 1.7923 - val_accuracy: 0.1643
Epoch 2/10
307/307 [==============================] - 84s 274ms/step - loss: 1.7904 - accuracy: 0.1766 - val_loss: 1.7920 - val_accuracy: 0.1800
Epoch 3/10
307/307 [==============================] - 84s 274ms/step - loss: 1.7899 - accuracy: 0.1835 - val_loss: 1.7917 - val_accuracy: 0.1800
Epoch 4/10
307/307 [==============================] - 83s 269ms/step - loss: 1.8133 - accuracy: 0.1940 - val_loss: 1.7916 - val_accuracy: 0.1802
Epoch 5/10
307/307 [==============================] - 82s 268ms/step - loss: 1.7923 - accuracy: 0.1772 - val_loss: 1.6873 - val_accuracy: 0.3099
Epoch 6/10
307/307 [==============================] - 83s 272ms/step - loss: 1.5666 - accuracy: 0.3266 - val_loss: 1.3700 - val_accuracy: 0.3683
Epoch 7/10
307/307 [==============================] - 83s 270ms/step - loss: 1.2865 - accuracy: 0.4281 - val_loss: 0.9875 - val_accuracy: 0.5728
Epoch 8/10
307/307 [==============================] - 83s 270ms/step - loss: 1.0633 - accuracy: 0.5585 - val_loss: 0.8925 - val_accuracy: 0.6303
Epoch 9/10
307/307 [==============================] - 83s 269ms/step - loss: 0.9714 - accuracy: 0.6160 - val_loss: 0.8874 - val_accuracy: 0.6604
Epoch 10/10
307/307 [==============================] - 83s 269ms/step - loss: 0.9215 - accuracy: 0.6458 - val_loss: 0.7799 - val_accuracy: 0.7020
```

測試模型

測試集準確度為：71%

```
[11] test_images,test_labels  =  get_images('../content/seg_test/seg_test/')
     test_images  =  np.array(test_images)
     test_labels  =  np.array(test_labels)
     model.evaluate(test_images,test_labels,  verbose=1)
```

```
94/94 [==============================] - 8s 81ms/step - loss: 0.7725 - accuracy: 0.7103
[0.7725214958190918, 0.7103333473205566]
```

4 改善方法與過程

改善方法與過程

改善方法1：調整學習率與輸出層分類函數（經實驗發現將學習率設為0.0001，輸出層之分類函數設為softmax表現最佳）

```
trained = model.fit(Images,Labels,epochs=10,batch_size=32,validation_split=0.30)
```

```
Epoch 1/10
307/307 [==============================] - 91s 296ms/step - loss: 1.9215 - accuracy: 0.2862 - val_loss: 1.0482 - val_accuracy: 0.5868
Epoch 2/10
307/307 [==============================] - 90s 293ms/step - loss: 1.2010 - accuracy: 0.5194 - val_loss: 0.8936 - val_accuracy: 0.6564
Epoch 3/10
307/307 [==============================] - 90s 294ms/step - loss: 1.0321 - accuracy: 0.6025 - val_loss: 0.8417 - val_accuracy: 0.6787
Epoch 4/10
307/307 [==============================] - 90s 293ms/step - loss: 0.9330 - accuracy: 0.6471 - val_loss: 0.7556 - val_accuracy: 0.7324
Epoch 5/10
307/307 [==============================] - 90s 293ms/step - loss: 0.8320 - accuracy: 0.6936 - val_loss: 0.6675 - val_accuracy: 0.7647
Epoch 6/10
307/307 [==============================] - 90s 293ms/step - loss: 0.7403 - accuracy: 0.7344 - val_loss: 0.7271 - val_accuracy: 0.7374
Epoch 7/10
307/307 [==============================] - 90s 294ms/step - loss: 0.7197 - accuracy: 0.7476 - val_loss: 0.5911 - val_accuracy: 0.7858
Epoch 8/10
307/307 [==============================] - 90s 294ms/step - loss: 0.6438 - accuracy: 0.7788 - val_loss: 0.6011 - val_accuracy: 0.7867
Epoch 9/10
307/307 [==============================] - 90s 294ms/step - loss: 0.5905 - accuracy: 0.8009 - val_loss: 0.5970 - val_accuracy: 0.7896
Epoch 10/10
307/307 [==============================] - 90s 293ms/step - loss: 0.5367 - accuracy: 0.8173 - val_loss: 0.5248 - val_accuracy: 0.8233
```

```
test_images,test_labels = get_images('../content/seg_test/seg_test/')
test_images = np.array(test_images)
test_labels = np.array(test_labels)
model.evaluate(test_images,test_labels,batch_size=32,verbose=1)
```

```
94/94 [==============================] - 8s 87ms/step - loss: 0.5127 - accuracy: 0.8363
[0.5127487182617188, 0.8363333344459534]
```

訓練資料集之準確度為81.73%，驗證資料集之準確度為82.33%，測試資料集之準確度為83.63%

改善方法與過程

改善方法2：將訓練資料之batch size調整為64(以改善方法1為基礎進行改善)

```
trained = model.fit(Images, Labels, epochs=10, batch_size=64, validation_split=0.30)
```

```
Epoch 1/10
154/154 [==============================] - 118s 755ms/step - loss: 0.4752 - accuracy: 0.8431 - val_loss: 0.4907 - val_accuracy: 0.8243
Epoch 2/10
154/154 [==============================] - 114s 743ms/step - loss: 0.4345 - accuracy: 0.8556 - val_loss: 0.4930 - val_accuracy: 0.8340
Epoch 3/10
154/154 [==============================] - 115s 747ms/step - loss: 0.4139 - accuracy: 0.8602 - val_loss: 0.5065 - val_accuracy: 0.8262
Epoch 4/10
154/154 [==============================] - 115s 748ms/step - loss: 0.3787 - accuracy: 0.8755 - val_loss: 0.5380 - val_accuracy: 0.8148
Epoch 5/10
154/154 [==============================] - 115s 745ms/step - loss: 0.3678 - accuracy: 0.8788 - val_loss: 0.4791 - val_accuracy: 0.8407
Epoch 6/10
154/154 [==============================] - 115s 746ms/step - loss: 0.3536 - accuracy: 0.8813 - val_loss: 0.5020 - val_accuracy: 0.8342
Epoch 7/10
154/154 [==============================] - 115s 749ms/step - loss: 0.3217 - accuracy: 0.8918 - val_loss: 0.5568 - val_accuracy: 0.8271
Epoch 8/10
154/154 [==============================] - 116s 751ms/step - loss: 0.3013 - accuracy: 0.9015 - val_loss: 0.5183 - val_accuracy: 0.8321
Epoch 9/10
154/154 [==============================] - 115s 749ms/step - loss: 0.2725 - accuracy: 0.9083 - val_loss: 0.5281 - val_accuracy: 0.8428
Epoch 10/10
154/154 [==============================] - 115s 748ms/step - loss: 0.2545 - accuracy: 0.9143 - val_loss: 0.5524 - val_accuracy: 0.8416
```

```
test_images, test_labels = get_images('../content/seg_test/seg_test/')
test_images = np.array(test_images)
test_labels = np.array(test_labels)
model.evaluate(test_images, test_labels, batch_size=32, verbose=1)
```

```
94/94 [==============================] - 8s 80ms/step - loss: 0.5375 - accuracy: 0.8453
[0.5375233292579651, 0.8453333377838135]
```

訓練資料集之準確度為91.43%，驗證資料集之準確度為84.16%，測試資料集之準確度為84.53%

改善方法與過程

改善方法3：將卷積層數量改為7(以改善方法2為基礎進行改善)

```
trained = model.fit(Images,Labels,epochs=10,batch_size=32,validation_split=0.20)
```

```
Epoch 1/10
351/351 [==============================] - 162s 455ms/step - loss: 1.6032 - accuracy: 0.3518 - val_loss: 1.0363 - val_accuracy: 0.6067
Epoch 2/10
351/351 [==============================] - 159s 454ms/step - loss: 1.1315 - accuracy: 0.5618 - val_loss: 0.8519 - val_accuracy: 0.6879
Epoch 3/10
351/351 [==============================] - 159s 454ms/step - loss: 0.9905 - accuracy: 0.6291 - val_loss: 0.7833 - val_accuracy: 0.7246
Epoch 4/10
351/351 [==============================] - 160s 455ms/step - loss: 0.8524 - accuracy: 0.6840 - val_loss: 0.6735 - val_accuracy: 0.7481
Epoch 5/10
351/351 [==============================] - 160s 455ms/step - loss: 0.7452 - accuracy: 0.7341 - val_loss: 0.6290 - val_accuracy: 0.7795
Epoch 6/10
351/351 [==============================] - 159s 454ms/step - loss: 0.7089 - accuracy: 0.7644 - val_loss: 0.5510 - val_accuracy: 0.8119
Epoch 7/10
351/351 [==============================] - 159s 454ms/step - loss: 0.6226 - accuracy: 0.7937 - val_loss: 0.6200 - val_accuracy: 0.7813
Epoch 8/10
351/351 [==============================] - 159s 453ms/step - loss: 0.5802 - accuracy: 0.8074 - val_loss: 0.5434 - val_accuracy: 0.8090
Epoch 9/10
351/351 [==============================] - 160s 455ms/step - loss: 0.5332 - accuracy: 0.8211 - val_loss: 0.5357 - val_accuracy: 0.8212
Epoch 10/10
351/351 [==============================] - 159s 454ms/step - loss: 0.5022 - accuracy: 0.8329 - val_loss: 0.4949 - val_accuracy: 0.8422
```

```
test_images,test_labels = get_images('../content/seg_test/seg_test/')
test_images = np.array(test_images)
test_labels = np.array(test_labels)
model.evaluate(test_images,test_labels,batch_size=32,verbose=1)
```

```
94/94 [==============================] - 12s 129ms/step - loss: 0.4691 - accuracy: 0.8457
[0.46908560395240784, 0.8456666469573975]
```

訓練資料集之準確度為83.29%，驗證資料集之準確度為84.22%，測試資料集之準確度為84.57%

改善方法與過程

	原型	方法一	方法二	方法三
Number of convolution layers	6	6	6	7
Number of pooling layers	2	2	2	2
Number of fully connected layers	4	4	4	4
Activation function	Relu	Relu	Relu	Relu
Classification function of the output layer	Sigmoid	Softmax	Softmax	Softmax
Optimizer	Adam	Adam	Adam	Adam
Learning rate	0.001	0.0001	0.0001	0.0001
Batch size	32	32	64	64
Test accuracy	71%	83.63%	84.53%	84.57%

5

結論

Thanks For Your Watching

谢　谢　聆　听