國立清華大學

智慧化企業整合

Project3

交通號誌辨識

指導教授:邱銘傳 教授

110034586 熊傘峻

111/01/07

目錄

第一章、前言

1.1 研究背景

由於近年來人工智慧的崛起，製造業陸續開始使用智慧機器人機器代替傳統人力，以提高產能、降低人資開銷。除了製造業外，各產業也先後導入人工智慧技術。如：手機的人臉辨識功能。透過人工智慧技術的協助，人們將擁有更便利的生活及有效率的工作模式。因此，本次報告將透過深度學習模型「CNN」協助無人車辨識交通號誌。

1.2 5W1H

隨著無人車的發展逐漸成熟，其交通號誌辨識的功能也愈發重要。而無人車辨識交通號誌之準確度越高意味著其發生意外的機率越低。因此，本次報告將以 CNN 模型辨識車道上種種交通號誌，協助無人車提高辨識交通號誌的能力。以下為本次報告所定義的 5W1H。

What：辨識交通號誌

Where：車道上各種交通號誌

Who：協助無人車辨識

When：無人車行駛時

Why：提高無人車判斷交通號誌的能力

How：CNN 模型

第二章、資料前處理

2.1 CNN 模型介紹

- 卷積神經網路（Convolutional Neural Network, CNN）
  卷積神經網路是一種前饋神經網路，它的人工神經元可以回應一部分覆蓋範圍內的周圍單元，其對於大型圖像處理有出色表現。卷積神經網路由一個或多個卷積層和頂端的全連接層組成，同時也包括關聯權重和池化層（pooling layer），這一結構使得卷積神經網路能夠利用輸入資料的二維結構，且其在圖像和語音辨識方面能夠給出更好的結果。卷積神經網路需要考量的參數較少，使得其成為一種頗具吸引力的深度學習結構。
- 卷積層（Convolution Layer）
  卷積層是一組平行的特徵圖（feature map），它透過在輸入圖像上滑動不同 的卷積核，並執行一定的運算而組成。此外，在每一個滑動的位置上，卷積核與 輸入圖像之間會執行一個元素對應乘積並求和的運算，以將感受野內的資訊投影 到特徵圖中的一個元素。這一滑動的過程可稱爲步幅 Z_s，步幅 Z_s 是控制輸出 特徵圖尺寸的一個因素。卷積核的尺寸要比輸入圖像小得多，且重疊或平行地作 用於輸入圖像中，一張特徵圖中的所有元素都是透過一個卷積核計算得出的，也 就是一張特徵圖共享了相同的權重和偏置項。
- 池化層（Pooling Layer）
  池化（Pooling）是一種非線性形式的降採樣，有多種不同形式的非線性池 化函式，而其中「最大池化（Max pooling）」是最為常見的。它是將輸入的圖像 劃分為若干個矩形區域，對每個子區域輸出最大值。 直覺上，這種機制能夠有效地原因在於一個特徵的精確位置遠不及它相對於 其他特徵的粗略位置重要。池化層會不斷地減小資料的空間大小，因此參數的數 量和計算量也會下降，這在一定程度上也控制了過擬合。一般來說，CNN 網路結 構中的卷積層之間都會周期性地插入池化層，而池化操作提供了另一種形式的平 移不變性。由於卷積核是一種特徵發現器，我們透過卷積層可以很容易地發現圖 像中的各種邊緣。然而，卷積層發現的特徵往往過於精確，我們即使高速連拍拍 攝一個物體，相片中的物體的邊緣像素位置也不大可能完全一致，而透過池化層 我們可以降低卷積層對邊緣的敏感

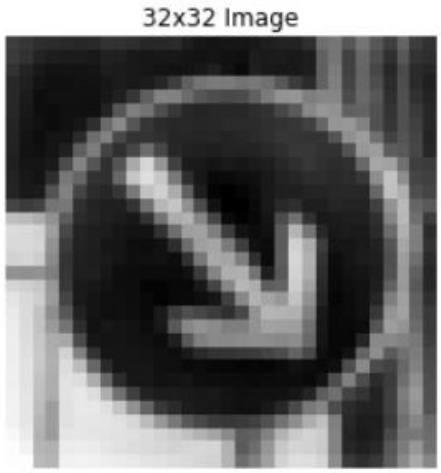
性。

- 全連接層（Fully Connected Layer）
  在經過幾個卷積層和最大池化層之後，神經網路中最後的進階推理透過完全 連接層來完成。就和常規的非卷積人工神經網路中一樣，完全連接層中的神經元 與前一層中的所有啟用都有聯絡。因此，它們的啟用可以作為仿射變換來計算， 也就是先乘以一個矩陣然後加上一個偏差（bias）偏移量（向量加上一個固定的 或者學習來的偏差量）。

2.2 資料處理

本次報告的資料來源由 Kaggle 公開數據集取得交通號誌圖像資料。此筆資料的訓練集共有 86989，驗證集共有 4410。

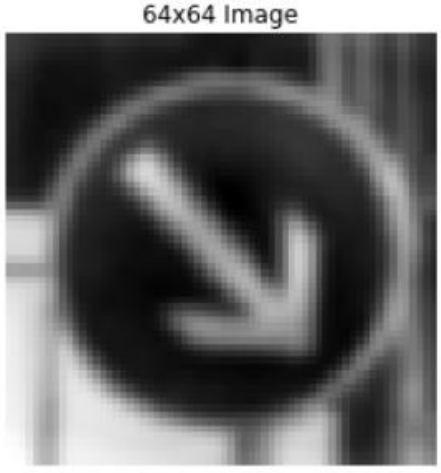
下圖顯示之圖像為 Kaggle 資料中交通號誌圖像。由圖像可明顯看出其解析度並不高。



因此本研究先將交通號誌圖像資料由原本的 32*32 擴增為 64*64。

```
def resize(img):
    numberofImage = img.shape[0]
    new_array = np.zeros((numberofImage,64,64,1))
    for i in range(numberofImage):
        new_array[i] = tf.image.resize(img[i],(64,64))
    return new_array
```

擴增完後之交通號誌圖像顯示如下圖。



2.3 建立模型

建立合適的 CNN 模型如下圖所示。建立 CNN 模型所使用的模型為「Sequential」並使用「relu」作為激活函數。於 Sequential 模型中，本研究分別加入了 4 層卷積層、4 層池化層、1 層扁平層、2 層全連接層及 1 層 dropout 層。

```
model = Sequential()


model.add(Conv2D(filters = 128, kernel_size = (4,4), padding = "Same", activation = "relu", input_shape = (64,64,1)))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 64, kernel_size = (4,4), padding = "Same", activation = "relu" ))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 32, kernel_size = (4,4), padding = "Same", activation = "relu" ))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 16, kernel_size = (4,4), padding = "Same", activation = "relu" ))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Flatten())

model.add(Dense(units = 512, activation = "relu"))

model.add(Dropout(0.6))

model.add(Dense(units = NumberofClass, activation = "softmax"))
```

之後，本研究選用「rmsprop」作為模型優化器。如下圖所示。

```
model.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics = ["accuracy"])
```

最後，本研究設定批輛大小為 512，訓練次數為 10。顯示如下圖。

```
hist = model.fit(x_train_resized, y_train, batch_size = 512,
                 epochs = 10, validation_data = (x_val_resized, y_val))
```

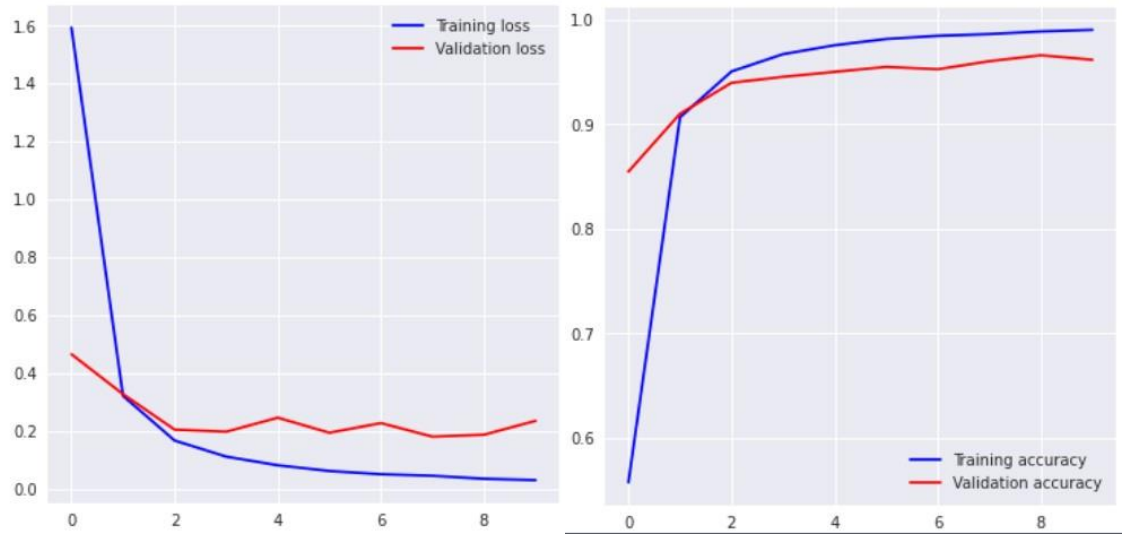
第三章、研究結果

3.1 訓練結果

訓練結果如下圖所示。雖然驗證資料之準確度高達 97.28%，不過運練次數由第 7 次開始，驗證資料之準確度不斷於 96%至 97%之間起伏不定。上述現象代表運練次數過多。此外，於最後一次訓練時，驗證資料的損失函數稍微有點上升。此現象意味著本研究所訓練的模型稍微有點過擬和。

```
Epoch 1/10
170/170 [==============================] - 1575s 9s/step - loss: 2.5515 - accuracy: 0.3184 - val_loss: 0.4649 - val_accuracy: 0.8544
Epoch 2/10
170/170 [==============================] - 1579s 9s/step - loss: 0.3907 - accuracy: 0.8849 - val_loss: 0.3270 - val_accuracy: 0.9098
Epoch 3/10
170/170 [==============================] - 1582s 9s/step - loss: 0.1815 - accuracy: 0.9460 - val_loss: 0.2047 - val_accuracy: 0.9395
Epoch 4/10
170/170 [==============================] - 1589s 9s/step - loss: 0.1196 - accuracy: 0.9645 - val_loss: 0.1973 - val_accuracy: 0.9451
Epoch 5/10
170/170 [==============================] - 1585s 9s/step - loss: 0.0854 - accuracy: 0.9743 - val_loss: 0.2458 - val_accuracy: 0.9499
Epoch 6/10
170/170 [==============================] - 1586s 9s/step - loss: 0.0607 - accuracy: 0.9817 - val_loss: 0.1938 - val_accuracy: 0.9546
Epoch 7/10
170/170 [==============================] - 1593s 9s/step - loss: 0.0500 - accuracy: 0.9848 - val_loss: 0.2271 - val_accuracy: 0.9524
Epoch 8/10
170/170 [==============================] - 1591s 9s/step - loss: 0.0479 - accuracy: 0.9855 - val_loss: 0.1803 - val_accuracy: 0.9601
Epoch 9/10
170/170 [==============================] - 1598s 9s/step - loss: 0.0338 - accuracy: 0.9889 - val_loss: 0.1871 - val_accuracy: 0.9658
Epoch 10/10
170/170 [==============================] - 1606s 9s/step - loss: 0.0299 - accuracy: 0.9901 - val_loss: 0.2350 - val_accuracy: 0.9615
```

訓練次數與準確率、損失函數的關係圖顯示如下。



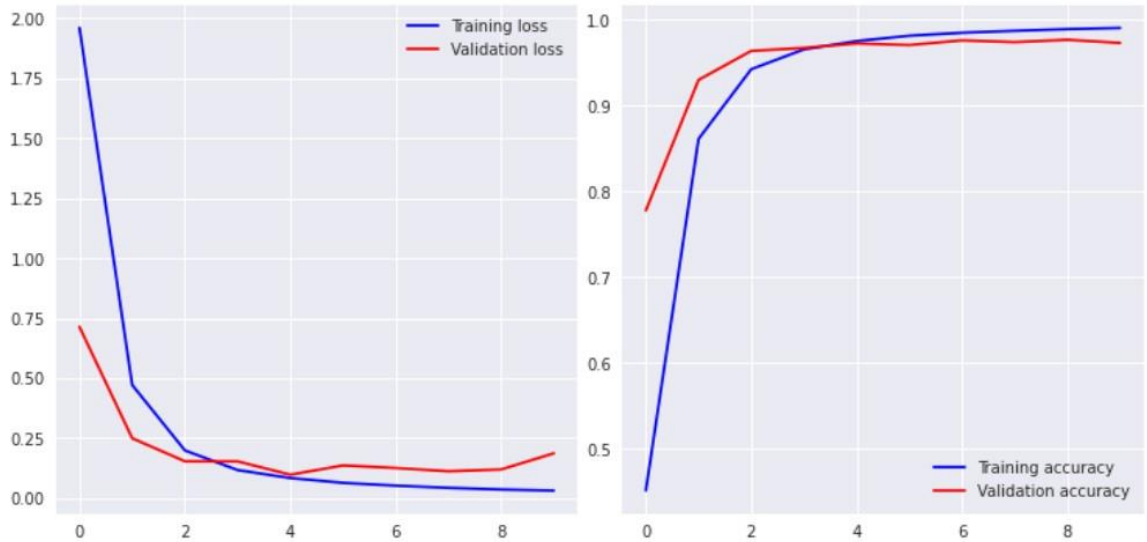
第四章、參數優化

4.1 調整參數

為提升模型準確度，本研究首先調整 CN 模型中的 optimizer。將原本的 adam 調整為 rmsprop。而訓練結果如下圖。由下圖可以看出模型之準確度由本的原本的 96.15%升高為 97.26%

```
Epoch 1/10
170/170 [==============================] - 1507s 9s/step - loss: 2.7626 - accuracy: 0.2586 - val_loss: 0.7138 - val_accuracy: 0.7776
Epoch 2/10
170/170 [==============================] - 1497s 9s/step - loss: 0.6114 - accuracy: 0.8179 - val_loss: 0.2492 - val_accuracy: 0.9295
Epoch 3/10
170/170 [==============================] - 1475s 9s/step - loss: 0.2336 - accuracy: 0.9326 - val_loss: 0.1531 - val_accuracy: 0.9633
Epoch 4/10
170/170 [==============================] - 1478s 9s/step - loss: 0.1228 - accuracy: 0.9636 - val_loss: 0.1528 - val_accuracy: 0.9667
Epoch 5/10
170/170 [==============================] - 1471s 9s/step - loss: 0.0864 - accuracy: 0.9735 - val_loss: 0.0971 - val_accuracy: 0.9719
Epoch 6/10
170/170 [==============================] - 1467s 9s/step - loss: 0.0658 - accuracy: 0.9802 - val_loss: 0.1359 - val_accuracy: 0.9701
Epoch 7/10
170/170 [==============================] - 1476s 9s/step - loss: 0.0514 - accuracy: 0.9842 - val_loss: 0.1251 - val_accuracy: 0.9755
Epoch 8/10
170/170 [==============================] - 1471s 9s/step - loss: 0.0403 - accuracy: 0.9872 - val_loss: 0.1116 - val_accuracy: 0.9735
Epoch 9/10
170/170 [==============================] - 1467s 9s/step - loss: 0.0342 - accuracy: 0.9890 - val_loss: 0.1192 - val_accuracy: 0.9762
Epoch 10/10
170/170 [==============================] - 1464s 9s/step - loss: 0.0289 - accuracy: 0.9906 - val_loss: 0.1865 - val_accuracy: 0.9726
```

訓練次數與準確率、損失函數的關係圖顯示如下。由損失函數途中可發現驗證曲線尾端明顯上翹。此現象意味著有過擬合的情形發生。

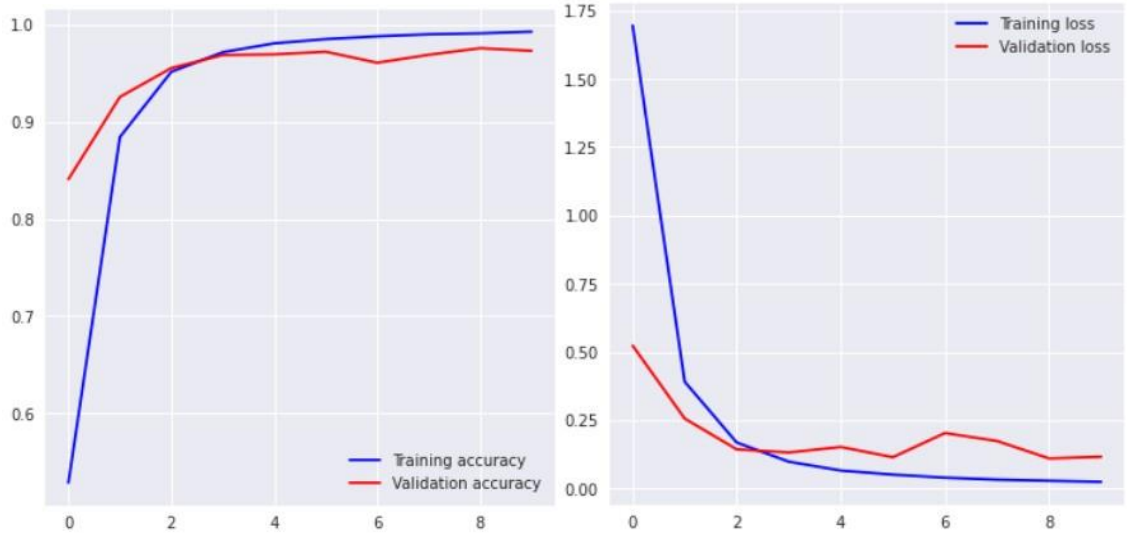


為解決過擬合的問題，本研究將將 dropout 層係數由原本的 0.6 改為 0.5。而訓練結果如下

圖。

```
Epoch 1/10
170/170 [==============================] - 1507s 9s/step - loss: 2.5748 - accuracy: 0.3123 - val_loss: 0.5229 - val_accuracy: 0.8408
Epoch 2/10
170/170 [==============================] - 1476s 9s/step - loss: 0.4925 - accuracy: 0.8529 - val_loss: 0.2557 - val_accuracy: 0.9254
Epoch 3/10
170/170 [==============================] - 1476s 9s/step - loss: 0.1922 - accuracy: 0.9448 - val_loss: 0.1429 - val_accuracy: 0.9551
Epoch 4/10
170/170 [==============================] - 1472s 9s/step - loss: 0.1027 - accuracy: 0.9698 - val_loss: 0.1315 - val_accuracy: 0.9685
Epoch 5/10
170/170 [==============================] - 1470s 9s/step - loss: 0.0670 - accuracy: 0.9796 - val_loss: 0.1518 - val_accuracy: 0.9692
Epoch 6/10
170/170 [==============================] - 1477s 9s/step - loss: 0.0513 - accuracy: 0.9842 - val_loss: 0.1141 - val_accuracy: 0.9719
Epoch 7/10
170/170 [==============================] - 1492s 9s/step - loss: 0.0393 - accuracy: 0.9875 - val_loss: 0.2027 - val_accuracy: 0.9605
Epoch 8/10
170/170 [==============================] - 1487s 9s/step - loss: 0.0299 - accuracy: 0.9905 - val_loss: 0.1735 - val_accuracy: 0.9687
Epoch 9/10
170/170 [==============================] - 1483s 9s/step - loss: 0.0268 - accuracy: 0.9911 - val_loss: 0.1097 - val_accuracy: 0.9755
Epoch 10/10
170/170 [==============================] - 1463s 9s/step - loss: 0.0233 - accuracy: 0.9929 - val_loss: 0.1158 - val_accuracy: 0.9728
```

訓練次數與準確率、損失函數的關係圖顯示如下。調整 dropout 層係數後及解決過擬合的問題。



4.2 比較結果

將原本的 optimizer 由 adam 調整為 rmsprop，並將 dropout 層係數由原本的 0.6 改為 0.5。即可得最佳結果。

第五章、結果與討論

5.1 結論

本研究所訓練之 CNN 模型辨識交通號誌的準確度接近 97.28%，大幅降低無人車交通號誌判斷錯誤的考能性。而無人車擁有高辨識交通號誌能力即可降低意外的發生。本研究欲訓練出 100%準確度之交通號誌辨識能力，以提高無人車使用者的便利性及安全性。

5.1 未來展望

除了繼續提高辨識交通號誌之準確度外，考量交通號誌圖示及交通號誌上的文字皆因國籍不同而有所不同。因此，本研究欲新增辨識各國交通號誌之能力，使無人車得以在國際之間穿梭自如。

此外，除了豎立在路上的看板式交通號誌外，道路上也存在了許多路面交通號誌。因此本研究亦希望擴充辨識路面交通號誌的功能，使無人車行駛更加便利。