機器學習系統設計實務與應用

HW3

組別 : 為什麼要醬樣組

組員 : B103012002 林凡皓

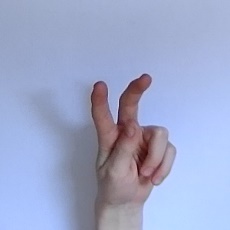
1. Mobile Net
2. 資料準備 :

關於訓練資料的準備，我選擇到kaggle上蒐集資料。會選擇到kaggle上蒐集資料有幾個原因 :

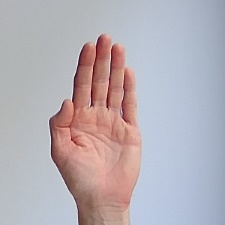
1. 資料中應該要包含多種膚色
2. 需要大量不相同的資料進行訓練
3. 背景需要做些許調整使模型可以適應多種相似的背景
4. 每個人的手勢都會有些許不同

基於以上幾點原因，我認為光是使用我們組員自己的產生的資料是不足以訓練出一個好模型，因此除了使用一些自己拍攝的圖片外，我也有到kaggle上蒐集不同膚色、不同背景、不同手勢形狀的圖片。

訓練資料範例如下 :

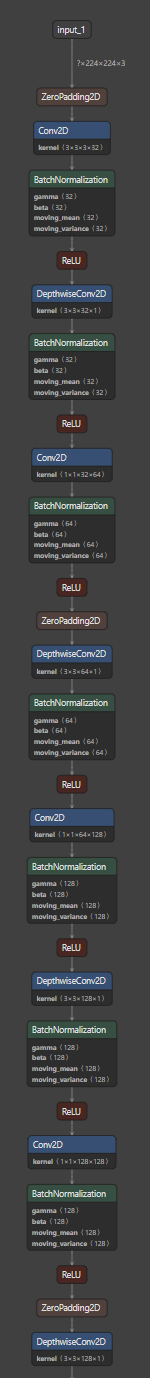
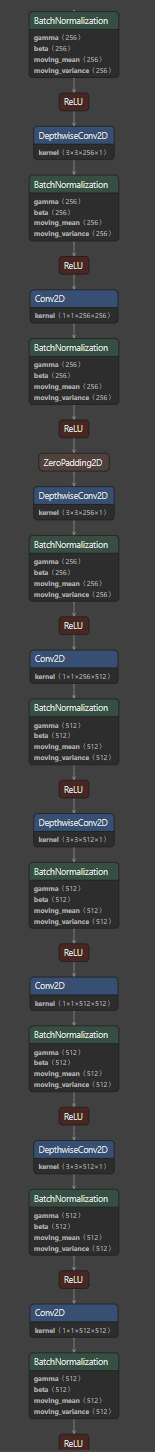
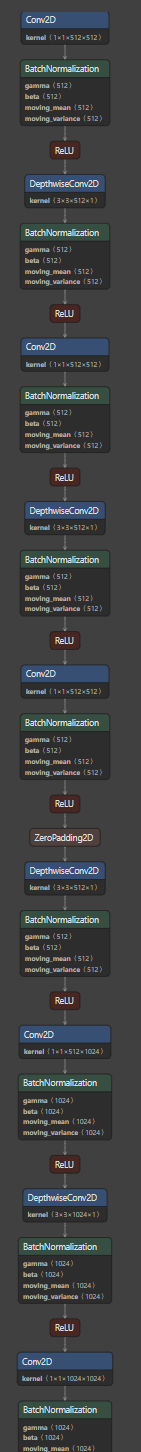
  

訓練資料一共包含2721張。

1. 網路架構 :

這一次訓練使用keras applications中的Mobile Net進行transfer learning。使用transfer learning而不是從頭訓練一個模型的原因在於訓練資料不足，為了確保模型的準確度，因此採用transfer learning做訓練。網路架構如下圖，

   一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

為了防止overfitting的發生，在訓練時有使用callbacks (early\_stopping、reduce\_lr)。此外，我有使用validation set來幫助我確認訓練是否有做起來以及確認overfitting是否有發生，我將validation split設置成0.1。其他參數設置如下 : epochs = 50、batch\_size = 32。

1. 訓練結果 :

訓練完成後，透過自己的筆電的相機進行測試。這邊我會分成兩種情況討論，分別為手勢朝上以及收使朝其他方向。

1. 手勢朝上 : 對於這種情況，模型基本上都可以預測正確，準確度可以說相當高，部分結果如下圖

由上面幾張結果可以發現，只要手勢方向朝上，即便有一些歪掉也能夠成功偵測。預測速度落在28 ~ 42fps之間。

1. 手勢朝其他方向 : 對於這種情況，模型基本上無法正確預測，部分結果如下圖

由上圖結果可以看出，基本上只有石頭可以預測正確，這個現象會在問題論中討論到。

1. 問題討論 :
2. Validation set設置 :

一般在做訓練，validation set基本上都是設置0.2。之所以這次設置為0.1是因為訓練資料過少，如果再將validation set設為0.2，那拿來做訓練的資料就可能會少到讓模型準確度下降。

1. 為甚麼手勢朝其他方向會偵測失敗 :

對於這個問題，我認為是training data的關係。這次拿來做訓練的資料都是朝上的，因此模型對於手勢朝上的圖片有很高的準確度，但是由於沒有拿手勢朝其他方向的圖片做訓練，因此模型無法有效偵測朝其他方向的手勢，唯獨石頭不管朝甚麼方向都是一個拳頭的形狀，因此模型可以成功偵測。接著針對這個問題做改進。

1. 加入轉向圖片
2. 資料準備 :

由於第一次訓練結果中無法成功偵測朝其他方向的手勢，因此這一次訓練前，我在第一次訓練的資料中，利用opencv加入轉向圖片(轉90、180、270度)，讓模型可以更好的去預測朝其他方向的手勢，資料總共4200張。部分訓練資料如下 :

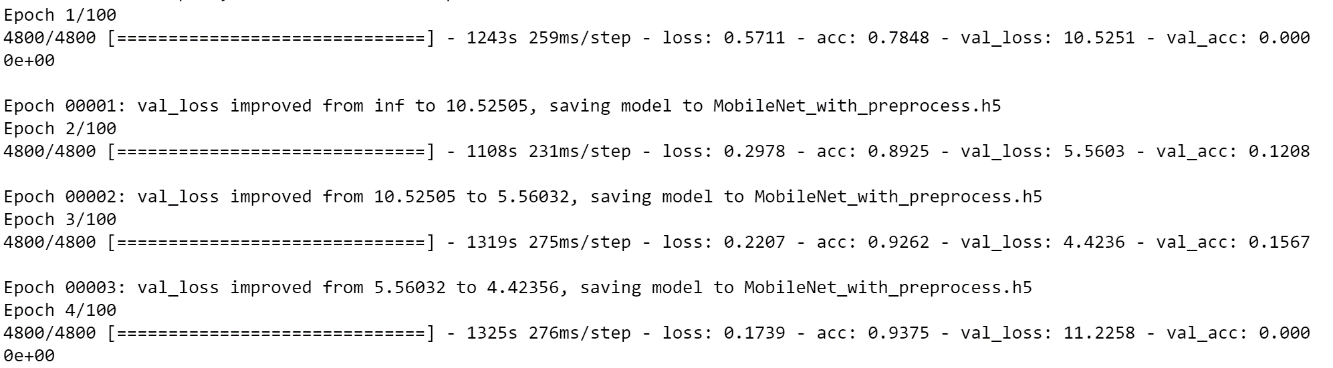
   

1. 網路架構 :

網路架構與第一次訓練模型相同。

1. 訓練過程 :

最初訓練過程如下圖



由上圖我們可以看到一個很奇怪的現象，那就是training accuracy和training loss都有不斷的進步，但是validation accuracy基本上完全沒在進步，或是準確度非常低。造成這樣的原因如下 :

1. 訓練資料分布不平均 :

由於圖片像素的值在0~255之間，這會使未經處理過的資料分布非常不平均，進而造成訓練上gradient descent的困難。為了解決這個問題，我將訓練資料除以225，讓資料分布範圍變成0~1之間。

1. 圖片未經shuffle :

想像一下，訓練模型時，假如一個epoch有30個資料，前十個資料都是0，接著十個資料都是1，最後十個資料都是2。這會造成模型在一開始被訓練成只會回答0的懶惰模型，接著雖然會去學習1特徵，但是沒回答幾題後發現答案都是1，又變回懶惰模型……。這就會造成模型學習不起來。因此我讓rescale後的資料做shuffle後再送進模型訓練。

1. 訓練結果 :

經過資料的預處理後，模型訓練結果如下

由上面結果可以看出，這一次的模型對於朝向各種方向的手勢都可以辨別正確。但是嚴格說起來，準確度不如第一個模型，問題討論中會對這樣的現象做討論。此外，偵測速度大約在25~30fps左右。

1. 問題討論 :
2. 第二次訓練的問題 :

第二次訓練結果中，模型預測會與環境有很高的關係，像是光線不足的情況下是無法偵測出paper的，會造成這樣的現象我認為原因在於訓練資料的背景與我測試環境的背景相比明亮許多，再加上模型中參數數量並不是很足夠的關係，導致說為了要偵測出手勢的方向而犧牲掉了對各種環境的適應能力。因此要解決這樣的問題，我認為要使用更大的模型來嘗試看看。

1. 使用ResNet50

由於先前嘗試後得出的結論為可能需要更大的模型，因此我嘗試使用ResNet50對有加入轉向的圖片做偵測。但是結果很不理想，因此我猜想會不會跟訓練資料有關係。我嘗試將經過轉換(.mat檔案)的訓練資料顯示出來，結果如下

一張含有 螢幕擷取畫面, 文字, 手, 人員 的圖片

自動產生的描述 一張含有 螢幕擷取畫面, 文字, 手 的圖片

自動產生的描述 一張含有 螢幕擷取畫面, 手 的圖片

自動產生的描述

由上面結果可以發現到，資料經過處理後背景變得非常白，而且背景非常單一，這導致訓練出來的模型對於環境的適應力非常差，而皆會需要像上面圖片一樣亮的環境才可以成功偵測。因此我嘗試使用更加複雜的背景作訓練。

1. 複雜背景訓練ResNet50
2. 資料準備 :

由先前之教訓，這一次我使用大量複雜背景之照片來做訓練，部分訓練照片如下

總共有2320張訓練資料。

1. 網路架構 :

這一次訓練採用transfer learning的方式進行，base model為keras.applications中的ResNet50，在後面加上一些Dense layer是為了要讓模型可以有一些參數調整的空間來擬和我的訓練資料。架構如下

一張含有 螢幕擷取畫面, 設計 的圖片

自動產生的描述

1. 訓練結果 :

這一次做出來模型準確度很高，不論是朝向何處的手勢，或是背景為何，都可以很好的偵測出手勢，下面放上在我的筆電上測試結果

1. 單純背景 :

1. 有人的背景 :

由上面結果可以看出，各個方向的手勢都可以偵測出來，而且即便背景不是單純的一個顏色，模型也都可以正確偵測。此外，偵測速度大約落在15fps左右，算是非常快速的。

1. 問題討論 :
2. 經過ResNet50的嘗試後，關於使用MobileNet對加入轉向圖片做訓練的結果會與測試環境高度相關是因為MobileNet模型太小而導致underfitting的這個結論，我認為需要更進一步驗證，畢竟單純的加大模型似乎無法解決這個問題，反而是在加入雜亂背景之訓練資料後，情況才得到改善，因此接下來我嘗試使用以雜亂背景的訓練資料來訓練MobileNet。
3. MobileNet對雜亂背景資料訓練
4. 資料準備 :

這次採用有雜亂背景的資料作訓練，資料與上一部分使用ResNet50訓練的資料相同。

1. 網路架構 :

採用keras.application中的MobileNet做transfer learning架構如下圖

一張含有 螢幕擷取畫面, 文字, 設計 的圖片

自動產生的描述

1. 測試結果 :

使用MobileNet在乾淨背景下準確度相當高，但是在有人的背景下對於石頭與布的偵測容易出錯。

1. 乾淨背景 :

由上面結果可以看出，各種方向的手勢都可以成功偵測。

1. 有人的背景 :

由上面四張圖可以看出，對於剪刀來說，即便背景中有人也能夠很順利地偵測出各種方向的手勢。但是對於布和石頭來說，就會很容易偵測錯誤，如下面兩張圖所示

由上面幾張結果也可以看出，偵測速率落在30fps左右。

1. 問題討論 :
2. 會嘗試使用MobileNet來對有雜亂背景之資料作訓練的主要原因在於像要去驗證之前做不起來的原因是甚麼。經過一番嘗試，我認為使用MobileNet是可以成功訓練出能夠偵測各種方向手勢的模型，但是想要做到像ResNet50一樣連在雜亂背景下都可以順利偵測的話就有些強人所難，畢竟對於MobileNet這種比較小的模型來說，能力還是很有限的。
3. 增加更多類別
4. 資料準備 :

這次使用的資料為在先前有雜亂背景之資料中再加入三個類別，分別為call、mute、rock\_and\_roll，部分資料如下

1. Call

1. Mute

1. Rock\_and\_roll

1. 網路架構 :

採用與先前成功偵測雜亂背景資料之ResNet50相同。

1. 測試結果 :

在這一次次試結果中，scissor、mute、rock、rock\_and\_roll有不錯的準確度，但是幾乎偵測不出paper和call。測試速度大約落在10fps左右。測試結果如下

1. 成功偵測 :

1. 失敗偵測 :

1. 問題討論 :
2. 為甚麼會偵測失敗?

我認為會失敗偵測的可能原因如下 :

* 資料過於雜亂 : 在這次的資料中，除了有六個類別需做偵測之外，每一張資料都相當的複雜，這就導致神經網路無法順利學習有效特徵，因此部分類別無法成功偵測。
* Underfitting : 神經網路不夠大也是原因之一，由先前嘗試結果可以得知，加大模型確實可以增加偵測更加複雜圖片的正確率，因此我認為，改用其他更深的神經網路可以有效改善這個問題。

1. 增加類別的影響

增加三個類別後準確度下降不少，我認為原因在於神經網路會需要將參數分配一些給偵測額外的類別，這會導致說平均每一個類別可以得到的參數量會下降，而在沒有overfitting的情況下，準確度會下降。基於同樣的網路架構與同樣的資料複雜度的情況下，想要不修改模型或是不經過一些預處理，像是二值化、邊緣偵測……等便能成功偵測所有類別是不可行的。

1. 加大模型對增加類別後資料做測試

為了解決增加類別後underfitting的問題，我選擇加大模型來嘗試解決這樣的問題。

1. 網路架構 :

我選擇在原本的ResNet架構的尾端再加上一些Dense layer，讓模型有更多參數空間去抓取新增的類別的特徵。可訓練的參數從原本的27,306,091增加到28,705,006。模型架構如下

一張含有 螢幕擷取畫面, 藝術, 春季, 設計 的圖片

自動產生的描述

1. 測試結果 :

這一次模型可以順利將所有類別都偵測出來，但是能然會因為一些動作上的差異而偵測錯誤，不過整體來說準確度不錯。以下為一些測試結果

由上方的結果可以看到，偵測速度為10fps左右。

1. 問題討論 :
2. 加大模型後的效果 :

由這次嘗試可以看出，原先偵測不出來的類別paper和call在更新模型後都可以成功偵測了，代表說增加模型參數確實可以有效提升模型性能。目前都只對準確度方面做討論，接著做預測速度的討論。

1. 預測速度討論 :

這次主要針對兩個模型做訓練，分別為MobileNet和ResNet50。這兩個模型中MobileNet可訓練參數為5,954,475，ResNet50可訓練參數為27,306,091(或是28,705,006)。從參數量就可以明顯看出ResNet50比MobileNet大上許多，這也反映在預測速度上。經過多種情境下的統計，ResNet50的預測速度落在15fps左右(加入新類別後甚至降到10fps)，而MobileNet的速度落在30fps左右(在比較簡單的資料集中甚至可以到40fps)。會有這樣的現象很合理，畢竟ResNet50有將近5倍多的參數需要計算，在偵測花費多一些時間也是合情合理。但是也因為這樣的trade off才造成ResNet50可以在更複雜的背景下有很高的準確度，甚至再加入一些很複雜的類別也可以有很不錯的表現。

1. Reference

[1] Kaggle “Rock Paper Scissors dataset” <https://www.kaggle.com/datasets/sanikamal/rock-paper-scissors-dataset>

[2] Kaggle “Rock-Paper-Scissor dataset” <https://www.kaggle.com/datasets/alexandredj/rock-paper-scissors-dataset>

[3] CSDN “十個原因可能導致訓練集明明很高，驗證集、測試集validation accuracy stuck 準確率很低 損失不減小” <https://blog.csdn.net/u013249853/article/details/89393982>

[4] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/>

[5] Kaggle “Rock Scissor Paper” <https://www.kaggle.com/datasets/alishmanandhar/rock-scissor-paper>