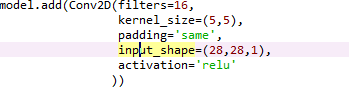
105650036 光電四 李振宣

題目:模擬卷積

動機:

在現今的時代，人工智慧已是一個家喻戶曉的名詞，其中對於圖像的處理如:物件辨識、物件偵測、影像還原……等皆是經常使用的領域，但像是物件辨識要如何去做辨識，即能夠找出不同物件的特徵，便能解決此問題，要如何取得特徵值便可以使用卷積的方式取得，目前的機器學習框架如:keras 內部要使用卷積都非常的簡單，但並不知道內部是如何運作，因此選擇了此題目來了解內部實際的狀況。



以上為keras使用卷積的程式碼 可以看到非常簡易 但實際在寫程式時若要使用GPU做卷積運算少說要快100行

目的:

以機器學習做物件辨識的領域中，卷積運算以可以算是家常便飯，然而在機器學習的領域中要計算的數據都非常大量，很容易輸入的筆數就上萬，輸入夠大的情況下平行發揮的效益就極大，且GPU跟CPU的核心數量差距極大，使得現今大多機器學習的訓練都要使用GPU來做運算，因此想要來簡單比較同樣用CPU及GPU來運算會有多少的加速。

平行/研究方法:

平行/研究方法如下

1.先去寫一個C的程式 也就是用CPU計算卷積的程式

2.由於工作內容沒有互相依賴可以直接把迴圈拆開做平行完成簡易版GPU運算

3.分配一塊共享記憶體讓讀取的速度加快做出共享記憶體版本的GPU運算

4.分別做計時去比較速度及加速效益

5.使用另一個輸出比較是否是限定特定輸入還是皆可

結果與檢討:

影片部分同樣放於壓縮檔內

比較部分如下

1. 卷積核7X7 shared memory為32X32

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 輸入\大小 | 256 | 512 | 1024 | 2048 | 4096 |
| rand() | 31 | 62 | 296 | 1200 | 4852 |
| srand(10) | 31 | 93 | 312 | 1201 | 4821 |

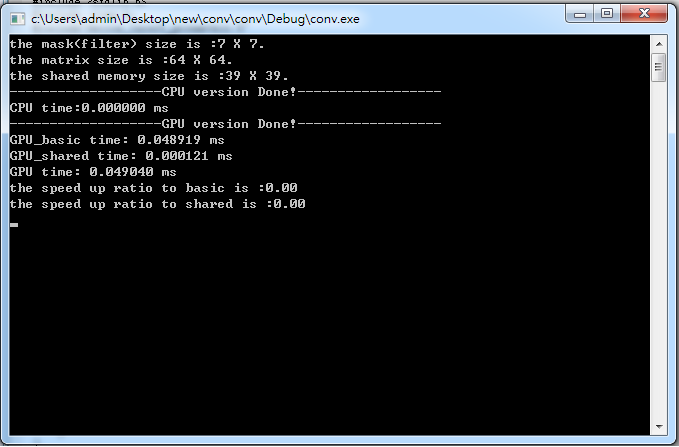
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 輸入\大小 | 256 | 512 | 1024 | 2048 | 4096 |
| basic1 | 0.025865 | 0.022888 | 0.023636 | 0.023 | 0.027 |
| shared1 | 0.000136 | 0.000417 | 0.001351 | 0.0048 | 0.019 |
| basic2 | 0.019049 | 0.018497 | 0.018272 | 0.018482 | 0.022184 |
| shared2 | 0.000115 | 0.000309 | 0.001037 | 0.003939 | 0.015578 |

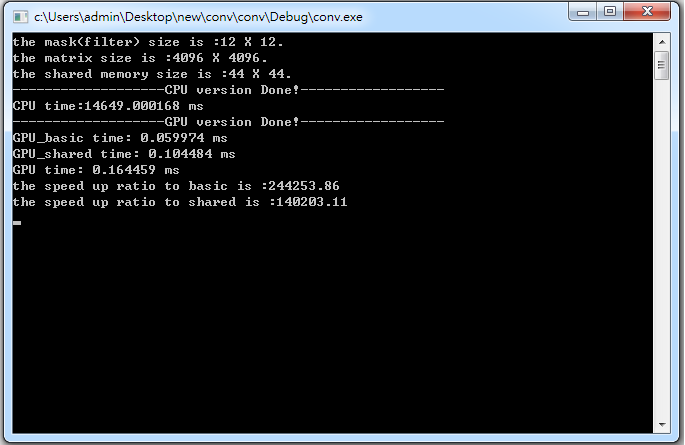
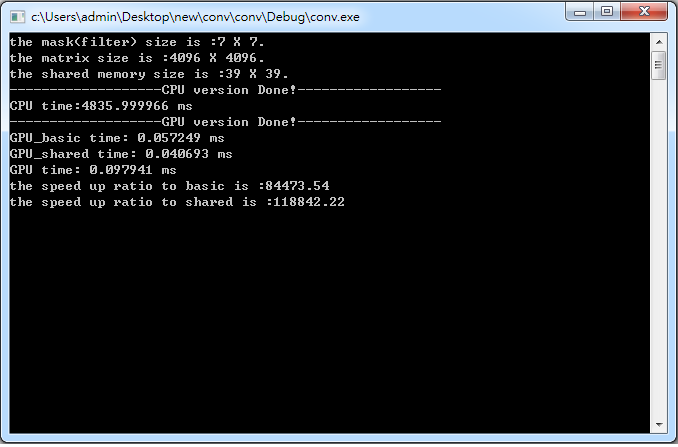
加速

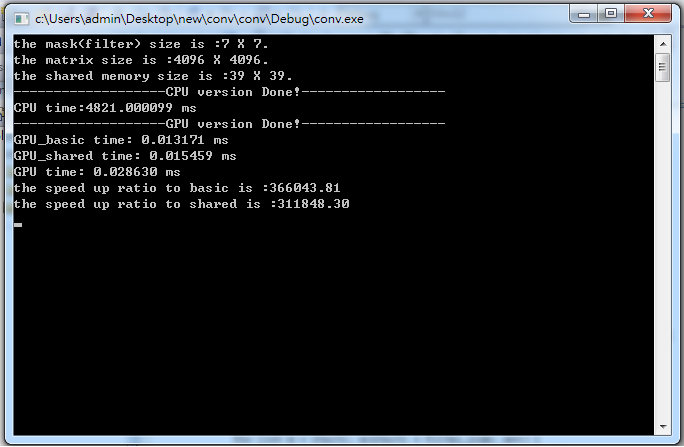
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 輸入\大小 | 256 | 512 | 1024 | 2048 | 4096 |
| 加速1 | 227960.6 | 148590 | 219051 | 249955 | 257303 |
| 加速2 | 269565.2 | 300970.9 | 300867.9 | 304899.7 | 309474.9 |

卷積核12X12 shared memory為32X32

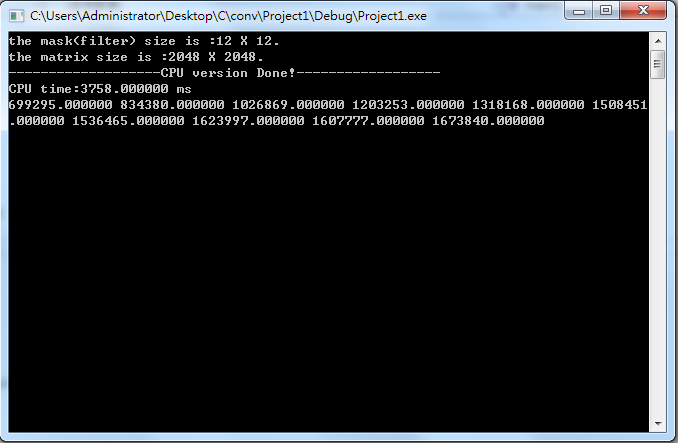
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 輸入\大小 | 256 | 512 | 1024 | 2048 | 4096 |
| CPU | 108 | 218 | 921 | 3635 | 14758 |
| GPU | 0.00022 | 0.000731 | 0.002245 | 0.010673 | 0.041756 |
| 加速 | 490909.1 | 298221.6 | 410318.1 | 340579 | 353434.2 |

一些實際運作的截圖





不同輸入的運行結果

由於在學校測試時最後把輸出註解掉在影片中沒有輸出 在此附上CPU版本的輸出

心得與感想:

一開始在做的時侯CPU的版本算是花了一小段時間去理解，因為一開始我是用很直覺的方法去寫，但遇到了些許問題因此去網路查詢，結果大多是用我最後的方法去寫。

到了GPU版本基本版也就只是單純把迴圈拆開，所以跟共用記憶體的版本一起寫，一開始遇到很多編譯上的問題，但還算順利解決，但意外的是基本版卻是有問題的反而是共享記憶體的版本可以運行，本來以為是把兩個版本放在一起有記憶體上的問題，額外把基本版拉出來執行卻還是不行，且經過測試函數本身並沒有運行，但也沒有報錯，因此我認為可能是在分配執行緒的部分有出了問題。

在經過老師點明後可以知道此加速是無意義的 因為測量到的GPU速度為呼叫指令的時間 (基本版)可能的問題是GPU過載 或超出界線