Parallel Program improve web crawler achieving AI model self-learning

閻俊宇
Institute of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
yen.cs09g@nctu.edu.tw

蔡詠平
Institute of Bioinformatics and Systems Biology
National Chiao Tung University
Hsinchu, Taiwan
vbnmzxc9513.bi08g@nctu.edu.tw

黃威竣
Institute of Data Science and Engineering
National Chiao Tung University
Hsinchu, Taiwan
ym110204.iie08g@nctu.edu.tw

Abstract

我們設計一套 AI 自主學習的系統，在訓練的過程中，程式會評估較差的準確度的類別並爬蟲從網路下載新的訓練集自動進行訓練，這種方法不僅擴展了訓練集的侷限性，也提升了模型的表現。我們系統分成三部分 Web Crawler 、 Feature Extraction 和 SVM Training ，運用 Pthread 和 Cuda 對各階段進行加速，最後我們的平行程式相比於順序執行加速了 38.3 倍，大大提升了系統的效率。

1. Introduction

隨著機器學習技術日漸成熟，我們認為機器學習除了模型要判斷更加精準以外，訓練模式也應該要越來越聰明，甚至有自主學習能力。而增加數據資料來進行訓練是最簡單粗暴增強模型表現的方法，以往蒐集資料和標記資料主要都是仰賴人工來做，這樣的方式縮現了訓練資料的多樣性和變異度，人工增加數據以增強模型的方式讓系統看起來不太聰明。

圖一. 所辨識的食物圖片範例，上面三張圖片由左至右為蘋果派、巧克力蛋糕、甜甜圈，下面四張由左至右為漢堡、熱狗、冰淇淋、披薩

所以我們建立一套系統能夠讓機器自行判斷自己所不足之處並且自主蒐集需要的數據來訓練，這種訓練系統就好比模仿人類想要增強自己某方面的知識會針對該領域主動去搜尋相關的資料來學習的方式。而且令人振奮的是人類的平行多工能力是有限的，我們很難在同一時間學習多種知識，但是機器只要透過平行程

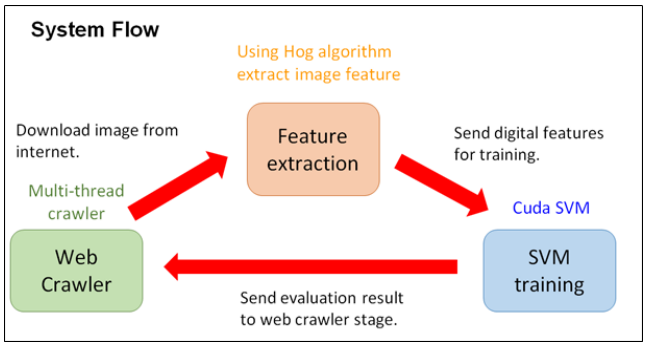
式就可以可以在同一時間處理大量訊息，讓學習這件事變得更有效率。

在這個研究中，我們將這種訓練模式運用在訓練食物分類模型，從 kaggle food-101 dataset 中選擇七個類別的食物作為我們的訓練集基礎訓練集，套用在我們所開發的系統中，輸入的圖片樣是如圖一，最終輸出該輸入圖片的類別為何。

訓練的流程主要可以分成三個階段，包含 Web Crawler 、 Feature Extraction 、 SVM training，針對這三個階段可平行的部分我們深入探討他們的瓶頸所在並且進行優化，平行方式包含 Pthread 還有 Cuda，經過優化後的系統相對於順序的執行有大幅度的提升。

在第二章節將會詳細介紹訓練過程每個環節如何運作。在過去的研究中，我們沒有發現與我們訓練方法類似的模式，因此這是一項嶄新的研究。

2. Proposed Solution



圖二. 系統起點為特徵擷取原基礎數據集影像，再來進行 SVM 訓練，然後爬蟲蒐集新影像增加數據集，回到起點接續循環執行。

我們的系統首先將數據集的影像進行特徵萃取，接著 SVM 模型進行訓練並且使用獨立測試集進行評估表現，系統將會記錄該次模型每一類別的準確度再分析表現較差的類別進行網路爬蟲增加額外的數據資料來增強該類別的於模型辨識的泛化能力，再將蒐集的圖片進行特徵擷取和模型建模評估，以此類推循環執行，大致流程如圖二所式。

2.1 Web Crawler

在網頁爬蟲方面我們使用 selenium 來模擬人的操作行為如點擊、下拉、拖曳等動作，來爬取各個預測類別的 Google Search 的圖片，進而增強我們的資料集，同時為了避免短時間大量發送 requests 被認為是對網站的攻擊行為，我們在三台 AWS 虛擬機上架設 squid 代理伺服器,由於每個虛擬機都有獨立不同的 ip 可以幫助我們轉發的 requests，達到發送的 requsets 分流的目地，避免我們頻繁的爬取而被 ban 掉。

在這次時中我們比較一條 thread、multi-thread、multi-process 和是否使用 proxy 進行代

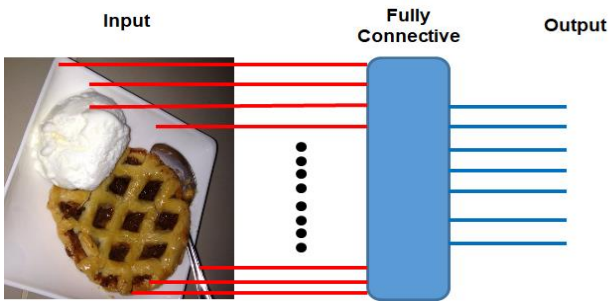
理執行爬取圖片的時間，來分析我們的加速是否是有效的。

2.2 Feature Extraction

得到可用的圖片資料集之後，為了要進行 SVM model 的 training，我們會需要將圖片的資料轉換，在這邊我們會將圖像的特徵萃取出來轉換成維度表示。

這邊採取的方式是使用 histogram of oriented gradient(HOG) method 進行萃取，在這個階段中，會將所有圖像轉成特徵向量，如果圖像的解析度越高，則轉變的時間愈多，這邊我們的解析度設成 512x512，由於我們所使用的資料量非常大，解析度也不低，因此在原先的資料集、之後擴增的資料集進行特徵萃取的動作會消耗過多的時間，造成整體流程的運作拖慢，因此在這邊會使用平行化的方式去進行特徵萃取，進行負載平衡，讓使用的每個 thread 的工作量平衡，減少此階段的運作時間

2.3 SVM Training



圖三. 全連層的輸入為影像的特徵，輸出為模型所計算出該類別的決策數值。

本次研究 SVM 模型是使用 Linear kernel，對特徵不進行額外的映射，這樣做的好處是模型較不會過度擬合訓練集數據。我們使用一層全連接層搭配 Hinge Loss 疊代進行反向傳播來調整模型參數，模型架構如圖三。

在疊代進行之前，由於我們設備的 GPU 記憶體很充足，所以我們一次將所有影像特徵還有 SVM 模型儲存在 GPU 記憶體中，運用 CUDA 計算資源執行正向傳遞和反向傳遞所需要的運算，只需要一次記憶體複製大大節省了設備之間溝通的花費，也讓整體模型訓練速度加快將近 3 倍。瓶頸的分析和優化的結果本專題會在第四章的部分和結果數據共同進行分析。

3.Experimental Methodology

本次實驗 SVM 框架使用 Pytorch 的包裹來實現，初始的 learning rate 為 0.001，batch size 為 2500，疊代次數為 6000，使用 3300 MB 的 GPU 記憶體。

實驗設備使用的 CPU 為 Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz ，核心數為 2* (12 cores 24 threads) ，GPU 為 Nvidia RTX 2080 Ti 12 GB，作業系統為 Centos 8。

系統的輸入資料初始為從 Kaggle Food

101 中所挑選的七類食物影像，每類食物影像共 1000 張，依照訓練集：獨立測試集=7:3 的比例進行分割，我們的實驗獨立對 Web Crawler 、 Feature Extraction 、 SVM training 這三個階段進行分析，並針對各階段分析程式執行各項任務在平行程式優化前的時間及平行程式優化後所花費的時間。

4.Experimental Results

4.1 Web Crawler

對於網路爬蟲，我們進行的任務為七個類別各 100 張圖片，總計 700 張圖片的爬取，並比較單一 thread 、 multi-thread 、muluti-proces 是否使用 prxoy 代理的情況結果如下表一。

	one thread	multi-thread	multi-process
use proxy	824.05s	10.85s	12.1s
no use proxy	56.1s	8.961s	10.5s

表一. 比較使用不同的方法實作 web crawler 所花費時間。

根據表一的數據可以發現使用 proxy 會導致我們所花費的時間增加，尤其是 one thread 的情況下差距更是明顯，原因在於使用 proxy 代理必然會增加一個等待 proxy 回應的時間，若使用 one thread 來實作，每個請求都會有這個無法避免的 overhead。此外在表一中 multi-thread 和 multi-process 不使用 proxy 的方法雖然時間上較使用 proxy 代理來得快，但此種方式相當容易被當成攻擊，若頻率太頻繁，這種爬取方式經常會失敗。

4.2 Feature Extraction

首先在資料集的部分，以使用七個類別，每個類別各 100 個數量，一共 700 張圖片，

Threads	times
1	1290s
2	673s
4	353s
8	169s
16	86s
32	48s
48	28.5s

表二. 本表為各個任務在優化前和優化後所花費時間的比較。

每個 thread 處理的圖片數量即為 total number of image / thread number，接著在每個 thread 計算完成之後，進行特徵向量的寫檔，可以看出在這部分使用平行化進行處理後，將時間大幅縮短。由於每個 thread 完成自己任務的時間會不大一樣，因此進行對檔案 append

時，每次操作只能由 1 個 thread 進行操作，在這部分會使用 mutex 進行避免兩個或兩個以上的 thread 同時進行操作。

4.3 SVM Training

Flow \ type	Before opt	After opt
Initialization	0.987 s	0.987 s
Load data	60.995 s	0.713 s
Training	719.805 s	49.566 s
Evaluation	0.2608 s	0.2792 s

表三. 本表為各個任務在優化前和優化後所花費時間的比較。

從 SVM Training 階段各項任務包含 Initialization, Load data, Training, Evaluation 在平行程式優化前的程式所花費時間依序為 0.987, 60.995, 719.805 秒，在優化後程式所花費時間為 0.987, 0.713, 49.566, 0.2792 秒如表三所示。

可以明顯看出程式執行時間主要開銷是在 Load data 和 Training 的部分。由於我們會先將萃取的圖像特徵以檔案的方式記錄在本地端， Load data 的工作就是要從本地端的讀取記錄特徵的檔案轉換為 python 的 pandas datafram 格式，起初使用 csv 格式來記錄資料，讀取就相當費時，而我們優化的方式就是改以 .pkl 的格式儲存並讀取，速度也獲得大幅提升。Training 的部分我們利用 CUDA 進行平行優化，計算速度得到大幅度提升，從 719 秒進步到只需要 49.5 秒。但是當我們將 CUDA 平行運算的方式套用在 Evaluation 上的時候卻沒有進步，甚至有所退步，其原因是因為在 Evaluation 上我們所測試的資料不像 Training 時這麼多，而且測試時不進行反向傳遞所以計算量也較小，因為以上兩個原因平行帶來的效益也被縮小。

經過平行化改良程式，在 SVM training 階段總共所花費的時間從 782.05 秒減少至 51.54 秒，速度約加快 15.17 倍。

4.4 Model Evaluation

我們 SVM model 目標，是要提升較弱的 class 的準確度，以下是對於各個 class 的準確度評估結果，在第一階段表現最差準確度的 class 為 Hambuger，因此我們爬蟲會針對此 class 的圖片進行拓展，拓展完畢後接續 round 2，可以見到我們的 hamburger 的準確度從 18.5% 提升至 22%，在接續的 round 3 提升至 24%

Class	Round1 Accuracy	Round2 Accuracy	Round3 Accuracy
Chocolate_cake	31%	25%	27%
Donuts	25%	23%	25%
Ice_cream	47.5%	44.5%	42%
Hot_dog	29%	29.5%	31%
Hamburger	**18.5%**	**22%**	**24%**
Pizza accuracy	52%	54.5%	50%
apple_pie	21%	22%	21.5%

表四. 此表為系統訓練模型進行三次循環每個類別的準確度的變化。

5.Related Work

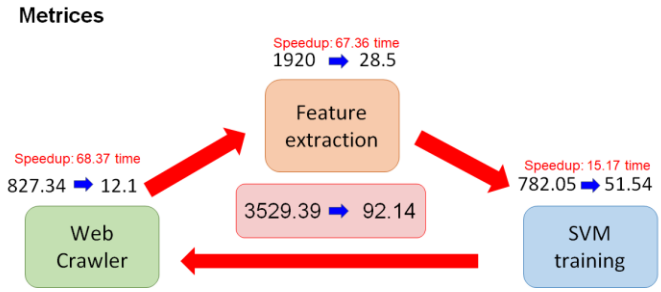
我們以 Libsvm [5]作為我們專題的比較對象。Libsvm 是一個使用起來很簡單、快速而且有效的 SVM 套件，重要的是這個套件在學術領域已經被廣泛運用，其強健性和穩定性普遍是受到肯定的。

將我們的 Cuda SVM 模型和 Libsvm 模型對同一數據集 (food-7) 進行訓練還有測試，最終我們模型對於每一類別的準確度都近似於 Libsvm 所甚至略好於 Libsvm 的預測結果。並且在比較所花費的時間上，Libsvm 花費 4284.5 秒才完成該數據集的訓練，而我們所提出的 Cuda SVM 模型僅需要 49.5 秒即完成訓練 (表五)，證實我們的 CUDA SVM 模型能在不影響訓練準確度的情況下，速度也能具有相當顯著的成效。

	libsvm	**cuda svm**
apple pie	**19.0 %**	**21.0 %**
chocolate cake	**29.5 %**	**31.0 %**
donuts	**26.5 %**	**25.0 %**
hamburger	**16.5 %**	**18.5 %**
hot dog	**20.5 %**	**29.0 %**
ice cream	**35.5 %**	**47.5 %**
pizza	**42.0 %**	**52.0 %**
mean acc	**27.1 %**	**32.0 %**
Spend time	**4284.5 s**	**49.5 s**

表五. libsvm 是公認穩定的 svm 版本，Cuda svm 是本專題所提出來的方法，從表顯示我們所提出的模型在不影響準確度的前題下，速度有顯著的提升

6.Conclusions



圖四. 各個階段於平行程式優化後的結果，整體系統單一循環所花費時間從 3529.39 秒加快到只需 92.14 秒

在這次專題中我們使用 multi-threading, cuda, pthread 等各種方法來加速我們的系統，相比於順序執行的情況，我們的平行化系統加速將近 35 倍，且對比於 Libsvm 套件所提供的 SVM 模型，我們所設計的 SVM 在有相似預測準確度的狀況下使用 CUDA 獲得更快的表現，如圖四。

在我們的實驗中透過爬蟲增加資料，增加模型的泛化能力，使模型可以看到更多面向的資料，雖然網路爬蟲的資料在相近時間點搜尋到的圖片有限，搜尋越多也越容易出現與預期獲得的圖片相差越大的圖片數據。但在我們的實驗中，這種訓練模式確實在有效率的情況下加強了模型辨識較難類別的能力，我們認為這種訓練方式隨著時間推移，所能蒐集的資料增加後訓練系統就能展現他的價值，也是這套系統的潛力所在。

7. Github

https://github.com/vbnmzxc9513/Parallel-self-learning-model

8. Reference

[1] Implementation of an efficient web crawler to search medicinal plants and relevant diseases
[2] Support Vector Machine
[3] food 101
[4] Docker Squid Proxy build
[5] Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2011, 2, 1–27.