行人偵測之深度學習實作與分析

指導教授：王元凱 博士　　學生：王婕、石芷瑜、董紹偉

輔仁大學 電機工程學系 大學部專題生

背景摘要

行人偵測技術(Human Detection)是現今最受重視的應用之一，不論是加強行駛中駕駛與行人的安全、或者是要讓自動駕駛甚至無人駕駛能實現在一般道路上，都與此項技術有密不可分的關連。機器學習為幾十年來行人偵測最主要使用的方法，像是支援向量機(SVM)、決策樹(Decision Tree)、AdaBoost、類神經網路...等。但因為類神經網路早期超過三層以上的運算會發生梯度消失的問題，使得其效果有限，相較之下不如採用其它效果更好的機器學習算法。直到2012年成功使用GPU訓練多層神經網路——稱之為「深度學習 (Deep Learning)」，不僅大幅降低平均偵測錯誤率，也讓訓練時間成功下降10~20倍，成為深度學習運算硬體的關鍵；而影像識別的錯誤率從2007年的26%左右，在2012年下降至3.5%，超越人類的5%，發展可謂一日千里，也再次帶起一股人工智慧領域的革命。

研究目的

現今已有多種深度學習方法能成功執行行人偵測之任務，雖然深度學習的準確率已經能達到一定水準以上，但依然會多少受限於外在因素影響而導致誤判，例如：明暗程度、人體的不同姿勢或角度、物體大小遠近、是否具有遮蔽物等...，故本專題欲分析現有之技術，期望未來能改善甚至再更進一步提高偵測率。在眾多深度學習架構中，本專題選擇著墨於Faster R-CNN，它是近年研究出的新穎技術，其不僅僅擁有深度學習訓練速度較快的特性，也能在訓練完成後的使用上大幅減少輸入到輸出之間所需耗費的時間。在應用層面上，最大的優點在於能做到接近「即時性的偵測」，一邊拍攝的同時即可完成偵測與分類的工作，而不需花太多時間等待結果。

Faster-RCNN (Regions with CNN features)

深度學習現今有幾個重要框架，其中，Convolutional Neural Network (CNN)又為目前廣為使用的深度學習框架的基礎。CNN架構主要分兩部分，如Figure 1所示，前半部分是特徵學習，後半部分是分類。在特徵學習的部分，Convolutional Layer使用不同的filter來提取各種可能的特徵，並且計算特徵和圖片局部的相符程度。Rectified Linear Unit (ReLU)則將特徵圖片上的所有負數轉為0，減少梯度消失的問題和增加網路的稀疏性，並且提高網路訓練速度。Pooling Layer為一種降維採樣但仍然保留重要資訊的方法，改善耗費運算量的問題。在分類的部分，Fully Connected Layer將上一部分輸出的特徵圖片經過全層連接之後，使用Softmax Function將圖像歸納為幾種可能的類別，並給出可能機率值，最後選擇最大的機率值的種類做為最終的判定。

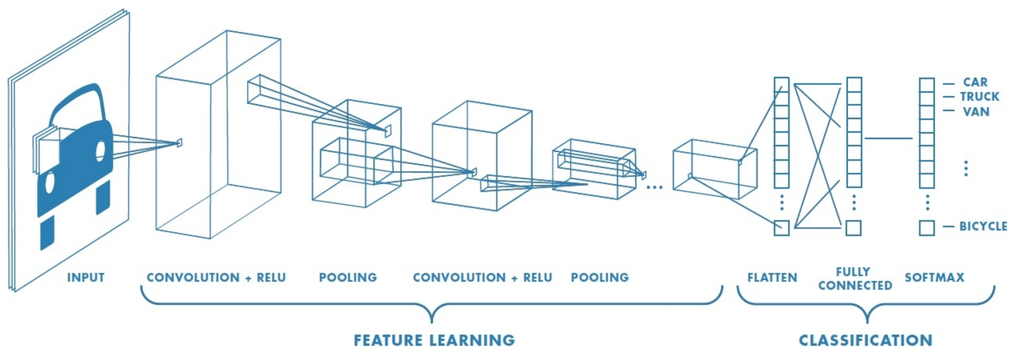


Figure 1、CNN架構

先前所及提，本專題欲著墨之Faster R-CNN，是以CNN為基礎作延伸。Faster R-CNN由S. Ren, K. He, R. Girshick and J. Sun等人在2015年發表於NIPS (Neural Information Processing Systems)，由一系列的演算法架構演變而來，相較於前幾代，其最大的改進在於將架構圖中所有步驟都放入CNN的運算之中，除了能共享特徵減少計算量，亦可使用GPU加速其效能。

Faster R-CNN架構中之Convolutional Layers和Feature Maps即是完成「特徵學習」的工作，其所得到的特徵再共享至Region Proposal Network運算，得出建議的目標區域(Proposal)，通過RoI Pooling統一目標區域尺寸，最後同時進行Softmax分類及Bounding Box Regression執行方框的調整，即可完成偵測的任務。

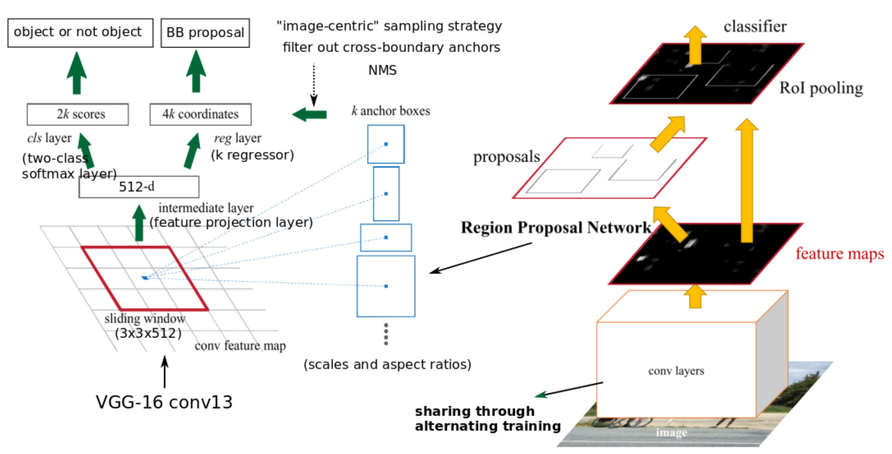


Figure 2、Faster R-CNN架構

實作方法

首先，本專題利用已訓練好的模型來觀察Faster R-CNN和過去常用之非深度學習框架的不同之處，另外再使用GitHub上所提供的TensorFlow Android Camera Demo，進行Faster R-CNN用於攝像即時偵測的實作與分析。由於深度學習模型訓練關鍵在於GPU與電腦系統的配合使用，故以下說明本專題實作時所使用之軟硬體規格：

① 繪圖卡型號：GeForce GTX 960M/PCIe/SSE2
② 作業系統：Ubuntu 14.04 LTS
③ GPU加速配置：NVIDIA 375.82、Cuda 8.0、CuDNN 5.1
④ 編譯環境：Python 2.7 (Anaconda 2 - 4.4.0)、TensorFlow GPU version

欲將深度學習的模型建置在Android裝置中，除了電腦規格需符合外，手機也有門檻上的要求——Android作業系統必須在5.0以上且具備後置攝像鏡頭。

建構好基本深度學習環境後，接著要在電腦上安裝開發套件用以整合裝置與軟體，需要SDK(Android Software Development Kit)開發工具，還有NDK(Native Development Kit)來自動將JAVA應用程式一起打包成apk，GitHub上提供一個完整的物件偵測模型，只要利用Bazel編譯就能製作.apk檔。最後使用Android SDK中的ADB工具，直接操作管理Android設備，手機進入開發人員選項後開啟USB偵錯模式，連接手機和電腦就能執行adb install，將模型安裝到手機上。

執行完成以上的前置作業，手機裡會出現4個應用程式，分別為TF Classify、TF Detect、TF Stylize、TF Speech，因本專題主要目的在於探討「行人偵測」的準確性，故這4個應用程式中，我們將以TF Detect作為主要分析工具。

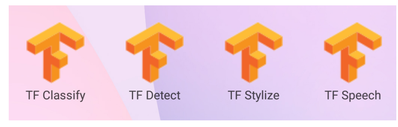


Figure 3、 TensorFlow Android Camera Demo

實作結果

為了比較深度學習和非深度學習的不同之處，我們自行蒐集了一些數據，將數張相同圖像分別輸入Faster R-CNN、HOG-SVM和Haar-Cascade觀察其中的差異，MR (Missing Rate)代表的是圖像中有人像卻沒有被標示出來的比率，FAR (False Alarm Rate)則代表沒有人像卻被判定為人的比率。結果如同預期地，非深度學習的兩個演算法HOG-SVM和Haar-Cascade在偵測物件的失誤率高於深度學習的Faster-RCNN許多。

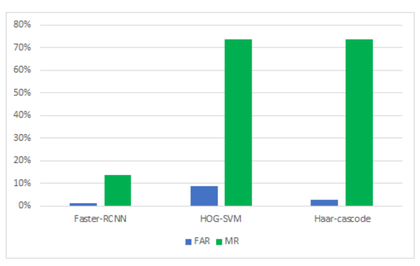


Figure 4、三種演算法的FAR、MR比率

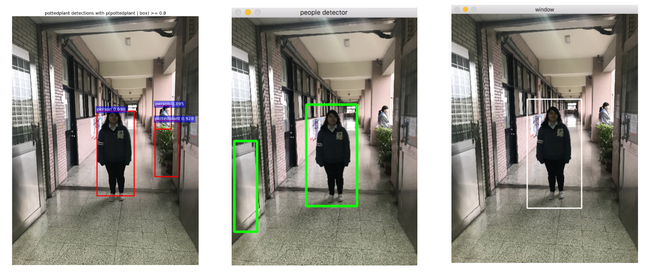
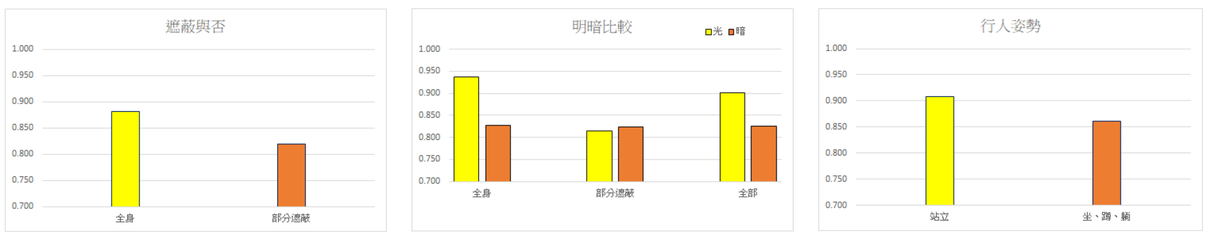


Figure 5、依序為Faster-RCNN、HOG-SVM、Haar-cascade實作結果

接著進行Faster R-CNN即時偵測的分析，本次實驗針對行人姿勢分為14個不同類別，分別對偵測數據進行分類探討。主要兩大類型，分為行人全身偵測與行人部分遮蔽偵測；行人全身偵測包括行人正面、側面、坐、蹲、躺五類，並且針對此五類分別做光線充足與光線不足之分析；行人部分遮蔽偵測包括上身遮蔽與下身遮蔽兩類，並且也針對此兩類分別做光線充足與光線不足之比較。

TF-detect會把偵測出來的行人標上框框與顯示偵測機率。我們將收集來的數據整理之後分成三大部分比較。

根據實驗結果，可見部分遮蔽較全身偵測之偵測機率低，明亮環境偵測機率又高於陰暗環境，而相較於站著的情況，其他姿勢偵測機率會稍微低一點。比較特別的是，在部分遮蔽的部分光與暗對偵測的影響就比較不明顯。



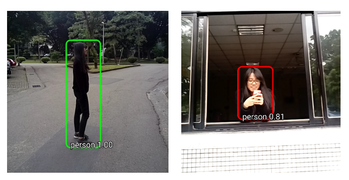


Figure 6、 遮蔽與否平均數據及其中一組對照圖

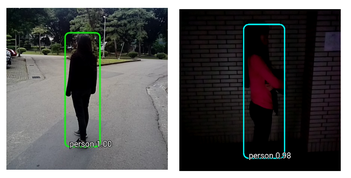


Figure 7、 明與暗平均數據及其中一組對照圖

Figure 8、 姿勢比較平均數據及其中一組對照圖

結論

深度學習之所以正確率較高，是因為傳統機器學習使用人已給定好的固定特徵來對比輸入的特徵相似性，而深度神經網路則會自主學習特徵，此種方法使得它的效果更佳。然而在眾多深度學習架構中正確率偏高、偵測速度快的Faster R-CNN模型，實務上仍然有可以改進的地方，如遮蔽情況以及昏暗的環境可能會降低偵測機率。

我們期望行人偵測能有更好的表現，在目前的能力所及內，以下列舉出近期之內本組專題組員的學習目標：

1. 使用ImageNet的其他資料庫進行訓練，並針對較差環境的圖片種類加強訓練的資料量
2. 修改訓練的神經網路深度或寬度
3. 精研更高深的深度學習演算法

2017 輔仁大學電機工程學系
大學部專題成果展